SEQUENTIAL MONTE CARLO PARAMETER ESTIMATION FOR DIFFERENTIAL EQUATIONS

by

ANDREA ARNOLD

Submitted in partial fulfillment of the requirements
For the degree of Doctor of Philosophy

Thesis Advisors: Dr. Daniela Calvetti and Dr. Erkki Somersalo

Department of Mathematics, Applied Mathematics and Statistics
CASE WESTERN RESERVE UNIVERSITY

May 2014

CASE WESTERN RESERVE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Andrea Arnold

candidate for the Doctorate of Philosophy degree *

(signed) Dr. Daniela Calvetti

(chair of the committee)

Dr. Erkki Somersalo

Dr. Jenný Brynjarsdóttir

Dr. Mark Griswold

March 20, 2014

* We also certify that written approval has been obtained for any proprietary material contained therein.

DEDICATION

To Grandma June, for her daily prayers, love, and support in “whateva” I do.

Table of Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family for their constant love and encouragement. In particular, to my mother, Diane, my father, Tedd, and my brother, Nicholas – I could not ask for a better support system, and I love you with all of my heart.

My never-ending gratitude goes to my incredible advisors, Dr. Daniela Calvetti and Dr. Erkki Somersalo, without whose encouragement, enthusiasm and energy this thesis would not have been possible. DC, you have been such a great role model and positive influence on me throughout my graduate studies. Thank you for always reinforcing infinite confidence in me – I will never forget it. When I think of you, this image from [1] will always come to mind:

EJ, thank you for your sound advice over the years and for always managing to make me smile, especially when I've needed it the most. Your patience and calm disposition have immensely helped to see me through the most challenging times. I hope that you both know how much I appreciate everything you've done for me. I have such

fond memories of working with you over the course of my studies, and I am lucky to have you not only as mentors but also as friends. I look forward to working with you for years to come.

Many thanks to Dr. Jenný Brynjarsdóttir and Dr. Mark Griswold for their active participation as members of my thesis committee and for their invaluable comments and suggestions regarding my thesis. I greatly appreciate the time and effort you put forth in helping to improve my work.

I would like to thank everyone in the Department of Mathematics, Applied Mathematics and Statistics at Case Western Reserve University for helping to cultivate my education over the past five years. I am forever grateful for the kindness and support that all of the faculty members, staff, and students have shown me. Thank you to the graduate students, former and current, who have been the best friends I could ask for, both in and out of Yost Hall. To my fellow doctoral graduates of May 2014 – in the wise words of Elle Woods, "WE DID IT!!!"

In addition to my collaborators Albert Gjedde and Peter Iversen, I would like to acknowledge the following members of the "liver-brain" group at Aarhus University and the University of Copenhagen, who made the study possible from which we received the PET scan data: Kim Mouridsen PhD, Mikkel B. Hansen PhD, Svend B. Jensen PhD, Michael Sørensen MD PhD, Lasse K. Bak PhD, Helle Waagepetersen PhD, Arne Schousboe DSc, Peter Ott MD DSc, Hendrik Vilstrup MD DSc, and Susanne Keiding MD DSc.

Sequential Monte Carlo Parameter Estimation for Differential Equations

Abstract

by

ANDREA ARNOLD

A central problem in numerous applications is estimating the unknown parameters of a system of ordinary differential equations (ODEs) from noisy measurements of a function of some of the states at discrete times. Formulating this dynamic inverse problem in a Bayesian statistical framework, state and parameter estimation can be performed using sequential Monte Carlo (SMC) methods, such as particle filters (PFs) and ensemble Kalman filters (EnKFs).

Addressing the issue of particle retention in PF-SMC, we propose to solve ODE systems within a PF framework with higher order numerical integrators which can handle stiffness and to base the choice of the innovation variance on estimates of discretization errors. Using linear multistep method (LMM) numerical solvers in this context gives a handle on the stability and accuracy of propagation, and provides a natural and systematic way to rigorously estimate the innovation variance via well-known local error estimates.

We explore computationally efficient implementations of LMM PF-SMC by considering parallelized and vectorized formulations. While PF algorithms are known to be amenable to parallelization due to the independent propagation of each particle, by formulating the problem in a vectorized fashion, it is possible to arrive at an implementation of the method which takes full advantage of multiple processors.

We employ a variation of LMM PF-SMC in estimating unknown parameters of a tracer kinetics model from sequences of real positron emission tomography scan data. A combination of optimization and statistical inference is utilized: nonlinear least squares finds optimal starting values, which then act as hyperparameters in the Bayesian framework. The LMM PF-SMC algorithm is modified to allow variable time steps to accommodate the increase in time interval length between data measurements from beginning to end of the procedure, keeping the time step the same for each particle.

We also apply the idea of linking innovation variance with numerical integration error estimates to EnKFs by employing a stochastic interpretation of the discretization error in numerical integrators, extending the technique to deterministic, large-scale nonlinear evolution models. The resulting algorithm, which introduces LMM time integrators into the EnKF framework, proves especially effective in predicting unmeasured system components.

Chapter 1

Introduction

A central problem in numerous applications is the estimation of the unknown parameters of a governing system of differential equations from noisy measurements of some of the states, or more generally, functions of the solution, at discrete times. More precisely, we consider a system of differential equations

$$\frac{du}{dt} = f(t, u, \theta), \qquad u(0) = u_0 \tag{1.1}$$

where $u = u(t) \in \mathbb{R}^d$ is a vector containing the states of the system, $f : \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^d$ is the known model function, and $\theta \in \mathbb{R}^k$ is the vector of model parameters. While the model parameters θ, as well as the initial values u_0, may be partly or completely unknown, suppose that the measured observations are given by

$$b_j = g(u(t_j), \theta) + e_j \in \mathbb{R}^m, \qquad 0 < t_1 < t_2 < \ldots < t_T, \tag{1.2}$$

where e_j is the error term which we assume, for simplicity, to be additive, and $g : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^m$ is a known function. The problem can be stated as follows: Estimate $u(t)$ at given times, θ, and possibly u_0 from the measurements b_j.

Problems of this type, known as *filtering and parameter estimation problems*, arise in a variety of applications, including the modeling of cellular metabolism [2], viral infection in cell assays [3], atmospheric chemistry in remote sensing applications [4], and monitoring flow by indirect observations [5, 6].

1.1 Inverse problems

Filtering and parameter estimation problems belong to the field of *inverse problems*, since they aim to retrieve information regarding unknown quantities of interest from indirect, noisy observations. Inverse problems arise in many fields of science and engineering, including various areas of physics, mathematical biology, signal processing, medical imaging, and statistical inference; see [5, 7, 8, 9] for more examples. As the name suggests, inverse problems are the inverse of direct or forward problems [5]. More specifically, given a map f explicitly relating two variables x and y,

$$y = f(x), \tag{1.3}$$

the *direct* or *forward problem* is the computation of y from known x, while the inverse problem is the estimation of unknown x given y or a function of y. If y is an observed quantity, it may be corrupted by noise. In many applications of interest, the map f either does not have an inverse, or the inverse is not continuous, leading to the difficulties characteristically associated with inverse problems.

When considering the real-world application behind a mathematical model, there is often a natural distinction between the direct and inverse problem. For example, y in (1.3) can be thought of as an observation of a physical system and x as a physical parameter relevant to that system. Predicting the future behavior of a physical system from knowledge of its present state and physical parameters is a direct problem, while identifying physical parameters or initial conditions from observations of its state at a later time is an inverse problem. In general, inverse problems are concerned with determining causes for desired or observed effects [8].

According to Hadamard's definition, a *well-posed problem* satisfies the following three criteria [8, 9]:

1. Existence: For all admissible data, a solution exists.

2. Uniqueness: For all admissible data, the solution is unique.

3. Stability: The solution depends continuously on the data.

Conversely, a problem that violates at least one of the above criteria is said to be *ill-posed*. The inverse problems of interest in this thesis are ill-posed in the sense that small perturbations of the observed data can lead to arbitrarily large perturbations in the solution, thus violating the stability criterion. Furthermore, the uniqueness criterion may also be violated when insufficient data is available. Violation of the existence criterion follows if the noise or modeling errors cause the data not to be in the range of the forward map.

Regularization methods are necessary to stabilize ill-posed problems and to produce useful and stable solutions of a nearby problem. This process involves the introduction of additional information, typically in the form of a penalty term [9].

1.1.1 Bayesian approach to inverse problems

The Bayesian statistical approach to solving inverse problems involves modeling all unknown quantities as random variables, where the randomness, naturally accounting for uncertainty in the values of the quantities, is described in terms of probability distributions. The distribution of a random variable conveys what is known about its values and to what extent. This provides a natural link between inverse problems and *statistical inference*, where properties of an unknown distribution are inferred from its realizations.

From a Bayesian perspective, the solution of the inverse problem is therefore the *posterior density*, that is, the probability distribution of the quantity of interest x in (1.3), modeled as a random variable, conditioned on the observed data y, which expresses the degree of confidence about the quantity after the observation is made [7, 5]. The celebrated Bayes' formula links the posterior density of the random variable x to the likelihood of the data y and the density of x prior to taking the data into consideration. Regularization techniques in the Bayesian framework often correspond to choosing suitable prior distributions for x.

1.2 Filtering and parameter estimation problems in a Bayesian framework

In this thesis we formulate the filtering and parameter estimation problem using a Bayesian statistical approach. Within this general framework, a large number of both sequential and non-sequential methods have been proposed. Given discrete time instances

$$0 = t_0 < t_1 < t_2 < \ldots < t_T,$$

let u_j denote the state vector at time t_j, $j = 0, 1, \ldots, T$, and let b_j denote the data observed at time t_j, $j = 1, \ldots, T$, according to (1.2). The vector θ contains the parameters relating to the propagation of the model and/or the generation of the data. Denoting by D_j the accumulated data up to time $t = t_j$,

$$D_j = \{b_1, b_2, \ldots, b_j\}, \tag{1.4}$$

non-sequential methods aim to estimate the posterior probability density of the parameter vector, denoted by $\pi(\theta \mid D_T)$, from the complete data D_T, a process which involves building parametric predictive models based on the inferred values of θ. As will be detailed in later chapters, a popular approach for exploring the posterior density is to use Markov chain Monte Carlo (MCMC) sampling methods, most commonly utilizing some variation of the Metropolis-Hastings algorithm.

When the predictive models result in stiff systems of differential equations, integration over the entire time interval $[0, T]$ may be problematic. In these cases, *sequential methods* are a viable alternative to MCMC methods using all of the collected data in a single batch. The idea of sequential methods is to update the posterior probability density in a sequential fashion, that is,

$$\pi(u_j, \theta \mid D_j) \;\longrightarrow\; \pi(u_{j+1}, \theta \mid D_{j+1}),$$

where u_j is a discrete approximation of the state vector $u(t_j)$. One advantage of this approach is that the data need not be dealt with all at once, but rather the increasing

information is integrated gradually as the data arrive. We will demonstrate that this approach also has significant computational advantages.

Sequential methods for estimating only the state variable u include the classical Kalman filtering method in the case of linear equations and Gaussian models, and extended Kalman filtering, ensemble Kalman filtering, and particle filtering in the case of nonlinear and non-Gaussian models. Since both particle filters and ensemble Kalman filters are sequential methods which make use of ensemble statistics, we refer to these algorithms in particular as *sequential Monte Carlo* algorithms. These algorithms, which will be described in Chapter 4, serve as motivation for the filtering and sequential parameter estimation algorithms developed in this thesis.

1.3 Thesis organization

The topic of this thesis is the design and analysis of efficient and robust filtering algorithms for state and parameter estimation within a Bayesian inference framework. More specifically, we develop a method of linking the discretization error in numerical integration to the variance of the innovation term in the evolution equation of the prediction step in solving filtering problems and parameter estimation problems. We incorporate this technique into sequential Monte Carlo methods, specifically considering the particle filters and ensemble Kalman filters for nonlinear and non-Gaussian models.

In **Chapter 2** we review the mathematical background material relevant to the numerical time integration schemes used in the state evolution, emphasizing the key components needed for deriving the evolution innovation variance based on numerical discretization error. In **Chapter 3** we discuss the statistical tools necessary for formulating the filtering and parameter estimation problems in a Bayesian framework. We give the basics of Monte Carlo integration and motivate the use of MCMC techniques in exploring the posterior density.

A review of the literature regarding filtering and parameter estimation algorithms in a Bayesian statistical framework is given in **Chapter 4**, where we provide an

overview of non-sequential and sequential methods. We then focus on the sequential Kalman-type filters and particle filters that serve as motivation for the algorithms developed in the chapters that follow.

The novel contributions of this work are given in Chapters 5-8. In **Chapter 5** we develop a particle filter sequential Monte Carlo (PF-SMC) method employing linear multistep method (LMM) numerical time integrators, which was published in [10]. Using LMMs in this context affords stable and computationally feasible propagation, as well as the retaining of a high number of particles. The LMM solvers also provide a natural and systematic way to rigorously estimate the innovation variance through use of well-known local error estimates, thereby keeping the posterior variance of the state and parameter estimation as low as possible. We compare several strategies for the computationally efficient implementation of the LMM PF-SMC algorithm in **Chapter 6**, considering in particular parallelized and vectorized versions of the algorithm. Through applications to a suite of test problems, it is shown that both the size and structure of the problem considered determine whether the parallelized or vectorized version of the algorithm is more efficient.

In **Chapter 7** we apply a variant of the LMM PF-SMC algorithm in the estimation of unknown parameters of a model of tracer kinetics from sequences of real positron emission tomography (PET) scan data. More specifically, the algorithm is applied to dynamic PET images of $[1-^{11}\mathrm{C}]$-acetate-derived tracer accumulation in order to estimate the transport rates in a two-compartment model of astrocytic uptake and metabolism of the tracer for a group of 18 volunteers from three groups (healthy control individuals, cirrhotic liver patients, and hepatic encephalopathy patients). A combination of standard optimization and statistical inference approaches is employed by running a quasi-Newton nonlinear least squares algorithm to find optimal starting values for the particle filter, then treating the optimized values as a hyperparameters in the Bayesian statistical framework. A vectorized version of the LMM PF-SMC algorithm is modified to use variable time steps to account for the increase in interval length between data measurements from the beginning to the end of the procedure,

while keeping the time step the same for each particle. The resulting parameter distributions show a clear distinction in parameter values for the hepatic encephalopathy patients as compared to the other two groups.

We further the idea derived in [10] to ensemble Kalman filters in **Chapter 8**, employing a stochastic interpretation of the discretization error in numerical integrators and thereby extending the technique to deterministic, large-scale nonlinear evolution models, with innovation variance based on classic error estimates. The effectiveness of the resulting algorithm is demonstrated on the chaotic Lorenz-63 equations [11]. A variant of the algorithm, which introduces the use of a spatio-temporal prior in the context of ensemble Kalman filters, is used in an application to metabolism kinetics for skeletal muscle. We conclude the work and give final remarks in **Chapter 9**.

Chapter 2

Mathematical Background

In this chapter we discuss the main mathematical concepts that will be used in developing the time-integration schemes for the state evolution in the filtering and sequential parameter estimation algorithms to follow. Some basic concepts relating to numerical differential equations are presented in Section 2.1, and a review of linear multistep method numerical solvers is given in Section 2.2. The error control methods which will serve as a basis for choosing the variance of the innovation in time propagation are discussed in Section 2.3, while the variable time step linear multistep formulas are derived in Section 2.4.

2.1 Numerical differential equations

In the filtering and sequential parameter estimation algorithms central to this thesis, the time-evolution of the state is often garnered by a system of ordinary differential equations (ODEs). The following concepts relating to the numerical solution of ODEs stem from finite difference approximation techniques. The notation mainly follows that of [12, 13].

2.1.1 Finite difference approximations

A *finite difference method* for approximating solutions of differential equations at discrete points replaces the derivatives with their finite difference approximations. This changes the system of ODEs into a system of linear equations whose solution is the value of the solution of the ODE at selected grid points.

Let $u(x)$ be a function of one variable, assumed to be smooth enough to have derivatives of as high order as needed, and let $\bar{x}$ be a specific point in the domain of u. Based on the definition of the derivative of u at $\bar{x}$,

$$u'(\bar{x}) = \lim_{h \to 0} \frac{u(\bar{x}+h) - u(\bar{x})}{h},$$

the *forward difference approximation* of $u'(\bar{x})$ is given by

$$D_+ u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x})}{h} \tag{2.1}$$

for some small value of h; hence, $D_+ u(\bar{x})$ is the slope of the line interpolating u at the points $\bar{x}$ and $\bar{x}+h$. Replacing h with $-h$ yields the *backward difference approximation*

$$D_- u(\bar{x}) = \frac{u(\bar{x}) - u(\bar{x}-h)}{h}. \tag{2.2}$$

Note that both (2.1) and (2.2) are *one-sided* approximations of u', since u is evaluated at only one side of $\bar{x}$; i.e., at $x \geq \bar{x}$ for (2.1) and $x \leq \bar{x}$ for (2.2). The *central difference approximation*

$$D_0 u(\bar{x}) = \frac{u(\bar{x}+h) - u(\bar{x}-h)}{2h} \tag{2.3}$$

is a two-sided approximation of $u'(\bar{x})$ derived by simply averaging the two one-sided approximations (2.1) and (2.2). More general finite difference approximations can be derived using, e.g., the method of undetermined coefficients; higher order derivatives can be approximated in a similar fashion [12].

2.1.2 Initial value problems

An initial value problem (IVP) for a time-dependent system of ODEs is given by

$$\frac{du}{dt} = f(t, u(t)), \qquad t > t_0, \qquad u(t_0) = u_0, \tag{2.4}$$

where $u = u(t)$ is an $\mathbb{R}^d$-valued function, $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function whose components are (possibly nonlinear) functions of the entries of u, and $u_0 \in \mathbb{R}^d$ is a given initial condition. We assume for simplicity that $t_0 = 0$.

The order of an ODE is the highest derivative appearing in its formulation. Since an ODE of order m can be written as a system of m first-order ODEs by letting $u_j(t) = v^{(j-1)}(t), j = 1, \ldots, m$, any mth-order system of d equations can be written as a system of md first order equations in a similar fashion. Therefore, the formulation (2.4) is the standard form of an IVP. Further, equation (2.4) can be made *autonomous* by introducing an additional variable equal to t, thereby eliminating any explicit dependence of f on t.

In order to solve the IVP (2.4) numerically, we discretize the problem in time using a fixed time step $h > 0$. More specifically, we let $t_n = nh$ for $n \geq 0$ and denote the approximate solution by $u_n \approx u(t_n)$. Given an initial condition u_0 at time $t_0 = 0$, we devise a method to march forward in time and compute approximations $u_1, u_2, \ldots$ at consecutive discrete time instances $t_1, t_2, \ldots$.

There are many numerical schemes to compute the approximations u_n at each time instance. The simplest (and perhaps most well-known) is *Euler's method*, or *forward Euler*, given by

$$\frac{u_{n+1} - u_n}{h} = f(u_n),$$

which can be rewritten as

$$u_{n+1} = u_n + hf(u_n).$$

This method replaces $u'(t_n)$ with the forward difference approximation D_+u_n in (2.1). Forward Euler is a one-step, explicit, first-order *linear multistep method* (LMM), a

particular class of numerical integrators which will be discussed in detail in Section 2.2.

In general terms, a numerical scheme for solving an IVP is said to be *stable* if small perturbations in the initial condition lead to small variations in the solution. We will describe in more detail two specific types of stability, namely, zero-stability and absolute stability, relating to LMM time integrators in Sections 2.2.2 and 2.2.3, respectively.

2.1.3 Stiff equations

The notion of stiffness for a system of ODEs, while lacking a precise definition, is an important consideration in numerical integration. In general, an ODE system (2.4) is said to be *stiff* if its numerical solution by certain methods requires a significant decrease in the time step h to avoid instability. Many ODE systems arising in applications from chemical kinetics, mathematical biology, and weather prediction are stiff.

Stiffness seems to occur when the ODE system contains terms that lead to rapid variation in the solution; i.e., some components of the solution show a fast transient behavior [12]. Stiffness causes numerical difficulty since many methods will become unstable unless the time step is chosen small enough to adequately capture the rapid transient, which can in turn lead to the need of prohibitively small time steps that slow down or completely stall the solver.

In more quantitative terms, stiffness is often measured by the ratio of the largest and smallest (in modulus) eigenvalues of the Jacobian matrix of the system: If λ_j are the eigenvalues of the Jacobian $\mathcal{D}_u f(t,u)$ of system (2.4), then the *stiffness ratio* of (2.4) is given by

$$\frac{\max|\lambda_j|}{\min_{\lambda_k\neq 0}|\lambda_k|}. \tag{2.5}$$

A large stiffness ratio (2.5) implies that the eigenvalues of the Jacobian span several orders of magnitude, thus a wide range of time scales is present in the problem; the

latter is another characterization of stiff systems. However, not all systems with a high stiffness ratio turn out to be stiff (e.g., highly oscillatory problems), so care must be taken when interpreting the stiffness ratio.

A numerical approximation scheme can also be a source of stiffness in a problem; this is the case when discretizing parabolic partial differential equations (PDEs) into large systems of stiff ODEs via the method of lines approach, described in Section 2.1.4. We will see this more clearly when considering the advection-diffusion equation on a torus in Chapter 6. Since stiff systems arise in numerous important applications, various numerical methods for handling them effectively have been developed, including the backward differentiation formula class of LMMs which will be discussed in Section 2.2.6.

2.1.4 The method of lines

When considering finite difference methods for time-dependent PDEs, such as the parabolic advection-diffusion which will be discussed in Section 6.4, where spatial variations are related to temporal variations, both space and time must be discretized. Suppose that we seek a function $u(t)$ that satisfies a given parabolic PDE over some bounded spatial domain Ω. Along with an initial condition specifying the state at time $t_0 = 0$, conditions must be assigned all along the boundary $\partial\Omega$ of the domain. Standard boundary conditions include Dirichlet, when the value of the solution u is assigned at the boundary; Neumann, when information regarding the derivative u' rather than u itself is given; Robin, a mixture of Dirichlet and Neumann; and periodic boundary conditions.

Although there are schemes, e.g., the Crank-Nicolson method, to discretize both space and time simultaneously, one of the most popular approaches for the numerical solution of parabolic PDEs is the *method of lines* (MOL), in which spatial discretization is performed first, leaving a large system of ODEs where each component corresponds to the solution at some grid point as a function of time. The resulting *semidiscrete* system of ODEs is then solved using an appropriate numerical method,

so stability considerations must be taken into account. The MOL technique will be utilized in Section 6.4 in discretizing a two-dimensional advection-diffusion equation.

2.2 Linear multistep methods

LMMs [12, 13, 14] are an important class of methods for the numerical solution of initial value problems. Consider the IVP (2.4), where for simplicity the possible dependency of the righthand side on model parameters is not explicitly indicated, and let $u_n \approx u(t_n)$, where $t_n = nh$ for $n \geq 0$ and time step $h > 0$.

In an r-step LMM for the solution of (2.4), the value u_{n+r} is computed from the previous values $u_{n+r-1}, u_{n+r-2}, \ldots, u_n$ and function values at these points according to the formula

$$\sum_{j=0}^{r} \alpha_j u_{n+j} = h \sum_{j=0}^{r} \beta_j f(u_{n+j}, t_{n+j}). \tag{2.6}$$

Clearly the values $u_0, u_1, \ldots, u_{r-1}$ must be available.

The LMM (2.6) is explicit if $\beta_r = 0$; otherwise the method is implicit. The choice of the coefficients α_j and β_j define different families of LMMs. For example, the *Adams methods*, characterized by $\alpha_r = 1$, $\alpha_{r-1} = -1$, and $\alpha_j = 0$ for $j < r - 1$, are of the general form

$$u_{n+r} = u_{n+r-1} + h \sum_{j=0}^{r} \beta_j f(u_{n+j}, t_{n+j}), \tag{2.7}$$

where the coefficients β_j are chosen to maximize the order of accuracy of the method.

Explicit Adams methods, known as *Adams-Bashforth* (AB) methods, are obtained by setting $\beta_r = 0$ and choosing the coefficients $\beta_0, \ldots, \beta_{r-1}$ in (2.7) such that the

method has order r. The AB methods can also be derived by writing

$$\begin{aligned} u(t_{n+r}) &= u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} u'(t)dt \\ &= u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} f(u(t))dt \\ &\approx u(t_{n+r-1}) + h\sum_{j=0}^{r-1} \beta_j f(u(t_{n+j})), \end{aligned}$$

where the quadrature rule approximating the integral follows from interpolation of the function $f(u(t))$ by a polynomial of degree $r-1$ at the points $t_n, \ldots, t_{n+r-1}$ and integrating the interpolating polynomial. This will be further detailed in Section 2.4.

Freeing β_r in (2.7) and choosing the remaining coefficients so as to have optimal order gives the implicit *Adams-Moulton* (AM) methods of order $r+1$. The AM methods can also be derived through polynomial interpolation similar to that used in deriving the AB methods, but instead with a polynomial approximation of degree r at the points $t_n, \ldots, t_{n+r}$.

Another family of implicit LMMs called *backward differentiation formulae* (BDF), obtained by setting $\beta_0 = \beta_1 = \cdots = \beta_{r-1} = 0$ in (2.6), are particularly effective for the numerical solution of highly stiff problems. We will give additional details regarding the AB, AM and BDF families of LMMs in Sections 2.2.4, 2.2.5 and 2.2.6, respectively.

2.2.1 Local error

Following the terminology of [12], the *local one-step error* of an LMM, defined as

$$\mathcal{L}_n = u(t_{n+1}) - u_{n+1}, \tag{2.8}$$

is the error introduced in one time step assuming that $u_n, u_{n-1}, \ldots, u_{n-r+1}$ coincide with the exact values of the solution $u(t)$ at $t = t_n, t_{n-1}, \ldots, t_{n-r+1}$. This error is referred to as the *local truncation error* in much of the literature concerning numerical

methods for ODEs. For example, consider the one-step AB method, i.e., the forward Euler scheme,

$$u_{n+1} = u_n + hf(u_n).$$

To find the local error, we first suppose that $u_n = u(t_n)$ and then use this value to compute $u_{n+1} \approx u(t_{n+1})$:

$$\begin{aligned} u_{n+1} &= u(t_n) + hf(u(t_n)) \\ &= u(t_n) + hu'(t_n). \end{aligned}$$

Substituting in (2.8) yields

$$\begin{aligned} \mathcal{L}_n &= u(t_{n+1}) - u_{n+1} \\ &= u(t_{n+1}) - u(t_n) - hu'(t_n) \\ &= u(t_n + h) - u(t_n) - hu'(t_n), \end{aligned}$$

and Taylor expansion of $u(t_n + h)$ about the point t_n gives

$$\begin{aligned} \mathcal{L}_n &= \left(u(t_n) + hu'(t_n) + \frac{1}{2}h^2u''(t_n) + \mathcal{O}(h^3) \right) - u(t_n) - hu'(t_n) \\ &= \frac{1}{2}h^2u''(t_n) + \mathcal{O}(h^3). \end{aligned}$$

Therefore, the local error of the forward Euler scheme is $\mathcal{O}(h^2)$. This means that as we march from t_n to t_{n+1} using forward Euler with a sufficiently small time step h, we introduce an error of $\mathcal{O}(h^2)$.

The magnitude of the local error of a method, expressed as a function of the time step, determines its accuracy. To see this, we find a general formula for the local error of the LMM (2.6) by writing it in the form

$$\sum_{j=0}^{r} \alpha_j u_{n+j} - h \sum_{j=0}^{r} \beta_j f(u_{n+j}, t_{n+j}) = 0$$

and by replacing the approximation u_n with the exact solution $u(t_n)$ and $f(u(t_n))$ with $u'(t_n)$, given that $u(t)$ is the exact solution to the IVP (2.4). This yields the difference expression

$$\varphi(t,u) := \sum_{j=0}^{r} \alpha_j u(t+jh) - h\sum_{j=0}^{r} \beta_j u'(t+jh) \tag{2.9}$$

for the LMM in the exact solution, written as a function of t and u. The limit of $\varphi(t,u)$ in (2.9) as $h \to 0$ determines the *order of accuracy* of the LMM.

More precisely, the LMM (2.6) is of order $p \geq 1$ if and only if

$$\varphi(t,u) = \mathcal{O}(h^{p+1}), \qquad h \to 0 \tag{2.10}$$

for u sufficiently smooth. In other words, an LMM is of order p if it integrates exactly every polynomial up to degree p [13]. It turns out, as we will show, that for analytic functions, the condition (2.10) can be rewritten as

$$\rho(\zeta) - \sigma(\zeta)\ln\zeta = c(\zeta-1)^{p+1} + \mathcal{O}(|\zeta-1|^{p+2}), \qquad \zeta \to 1 \tag{2.11}$$

for $\zeta \in \mathbb{C}$ and some real constant $c \neq 0$, where

$$\rho(\zeta) := \sum_{j=0}^{r} \alpha_j \zeta^j \qquad \text{and} \qquad \sigma(\zeta) := \sum_{j=0}^{r} \beta_j \zeta^j \tag{2.12}$$

are the *characteristic polynomials* of the LMM. The constant c in (2.11) is known as the *(local) error constant* of the LMM.

The equivalence of statements (2.10) and (2.11) can be shown using the Taylor series expansion of $\varphi(t,u)$. Assuming that u is a real analytic function, we have that

$$\begin{aligned}
\varphi(t,u) &= \sum_{j=0}^{r} \alpha_j u(t+jh) - h\sum_{j=0}^{r} \beta_j u'(t+jh) \\
&= \sum_{j=0}^{r} \alpha_j \left(\sum_{k=0}^{\infty} \frac{1}{k!} u^{(k)}(t) j^k h^k \right) - h\sum_{j=0}^{r} \beta_j \left(\sum_{k=0}^{\infty} \frac{1}{k!} u^{(k+1)}(t) j^k h^k \right) \\
&= \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{j=0}^{r} j^k \alpha_j \right) h^k u^{(k)}(t) - \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{j=0}^{r} j^k \beta_j \right) h^{k+1} u^{(k+1)}(t).
\end{aligned}$$

By rearranging and collecting terms of the same order in h, it follows that

$$\begin{aligned}
\varphi(t,u) &= \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{j=0}^{r} j^k \alpha_j \right) h^k u^{(k)}(t) - \sum_{k=1}^{\infty} \frac{1}{(k-1)!} \left(\sum_{j=0}^{r} j^{k-1} \beta_j \right) h^k u^{(k)}(t) \\
&= \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{j=0}^{r} j^k \alpha_j \right) h^k u^{(k)}(t) - \sum_{k=1}^{\infty} \frac{1}{k!} \left(k \sum_{j=0}^{r} j^{k-1} \beta_j \right) h^k u^{(k)}(t) \\
&= \left(\sum_{j=0}^{r} \alpha_j \right) u(t) + \sum_{k=1}^{\infty} \frac{1}{k!} \left(\sum_{j=0}^{r} j^k \alpha_j - k \sum_{j=0}^{r} j^{k-1} \beta_j \right) h^k u^{(k)}(t).
\end{aligned}$$

Therefore, for the LMM to be of order p as $h \to 0$, it is necessary and sufficient that

$$\sum_{j=0}^{r} \alpha_j = 0, \qquad \sum_{j=0}^{r} j^k \alpha_j = k \sum_{j=0}^{r} j^{k-1} \beta_j \text{ for } k = 1, \ldots, p \tag{2.13}$$

and

$$\sum_{j=0}^{r} j^{p+1} \alpha_j \neq (p+1) \sum_{j=0}^{r} j^p \beta_j. \tag{2.14}$$

To connect $\varphi(t,u)$ with the characteristic polynomials ρ and σ defined in (2.12), assume that $u(t) = e^{\lambda t}$ for $\lambda \in \mathbb{C}$, such that $u'(t) = \lambda e^{\lambda t}$. Then

$$\begin{aligned}
\varphi(t,u) &= \sum_{j=0}^{r} \alpha_j u(t+jh) - h \sum_{j=0}^{r} \beta_j u'(t+jh) \\
&= \sum_{j=0}^{r} \alpha_j e^{\lambda(t+jh)} - h \sum_{j=0}^{r} \beta_j \lambda e^{\lambda(t+jh)} \\
&= e^{\lambda t} \sum_{j=0}^{r} \alpha_j e^{\lambda jh} - \lambda e^{\lambda t} h \sum_{j=0}^{r} \beta_j e^{\lambda jh} \\
&= e^{\lambda t} \left(\sum_{j=0}^{r} \alpha_j e^{\lambda jh} - \lambda h \sum_{j=0}^{r} \beta_j e^{\lambda jh} \right).
\end{aligned}$$

Letting $z = \lambda h$ yields

$$\begin{aligned}\varphi(t,u) &= e^{\lambda t}\left(\sum_{j=0}^{r}\alpha_j e^{jz} - z\sum_{j=0}^{r}\beta_j e^{jz}\right)\\ &= e^{\lambda t}\Big(\rho(e^z) - \sigma(e^z)z\Big),\end{aligned}$$

and so

$$e^{-\lambda t}\varphi(t,u) = \rho(e^z) - \sigma(e^z)z.$$

Noting that $z \to 0$ as $h \to 0$, using a Taylor series expansion of e^{jz} about 0 on the righthand side yields

$$\begin{aligned}\rho(e^z) - \sigma(e^z)z &= \sum_{j=0}^{r}\alpha_j e^{jz} - \left(\sum_{j=0}^{r}\beta_j e^{jz}\right)z\\ &= \sum_{j=0}^{r}\alpha_j\left(\sum_{k=0}^{\infty}\frac{1}{k!}j^k z^k\right) - \sum_{j=0}^{r}\beta_j\left(\sum_{k=0}^{\infty}\frac{1}{k!}j^k z^k\right)z.\end{aligned}$$

Changing the order of summation and collecting terms of the same order in z, we have

$$\begin{aligned}\rho(e^z) - \sigma(e^z)z &= \sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r}j^k\alpha_j\right)z^k - \sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r}j^k\beta_j\right)z^{k+1}\\ &= \sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r}j^k\alpha_j\right)z^k - \sum_{k=1}^{\infty}\frac{1}{(k-1)!}\left(\sum_{j=0}^{r}j^{k-1}\beta_j\right)z^k\\ &= \sum_{k=0}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r}j^k\alpha_j\right)z^k - \sum_{k=1}^{\infty}\frac{1}{k!}\left(k\sum_{j=0}^{r}j^{k-1}\beta_j\right)z^k\\ &= \sum_{j=0}^{r}\alpha_j + \sum_{k=1}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r}j^k\alpha_j - k\sum_{j=0}^{r}j^{k-1}\beta_j\right)z^k.\end{aligned}$$

It follows that

$$\begin{aligned}\rho(e^z) - \sigma(e^z)z &= \frac{1}{(p+1)!}\left(\sum_{j=0}^{r} j^{p+1}\alpha_j - (p+1)\sum_{j=0}^{r} j^p\beta_j\right)z^{p+1} \\ &\quad + \sum_{k=p+2}^{\infty}\frac{1}{k!}\left(\sum_{j=0}^{r} j^k\alpha_j - k\sum_{j=0}^{r} j^{k-1}\beta_j\right)z^k \\ &= cz^{p+1} + \mathcal{O}(z^{p+2}) \qquad (2.15)\end{aligned}$$

for some $c \neq 0$ if and only if the p-order conditions (2.13) and (2.14) hold. Substituting $\zeta = e^z$ and observing that $\zeta \to 1$ as $z \to 0$ gives (2.11), as desired.

Note that the choice of $u(t) = e^{\lambda t}$ in the above computation corresponds to the homogenous linear test problem

$$u'(t) = \mathsf{A}u(t), \qquad (2.16)$$

where A a constant, diagonalizable matrix in $\mathbb{R}^{d\times d}$. If $\lambda \in \mathbb{C}$ is an eigenvalue of A, then the ODE when restricted to the eigenspace of A reduces to $u'(t) = \lambda u(t)$. The stability of numerical integrators for ODEs is usually studied using the test problem (2.16). The significance in letting $z = \lambda h$ and $\zeta = e^z$ as above becomes clear in the following sections.

It is sometimes convenient to define $w := \zeta - 1$ so that $\zeta = w + 1$ and equation (2.11) becomes

$$\rho(w+1) - \sigma(w+1)\ln(w+1) = cw^{p+1} + \mathcal{O}(|w|^{p+2}), \qquad w \to 0.$$

For example, for the forward Euler scheme, where $\alpha_1 = 1$, $\alpha_0 = -1$, $\beta_1 = 0$ and $\beta_0 = 1$, the corresponding characteristic polynomials are given by

$$\rho(\zeta) = \alpha_0 + \alpha_1\zeta = -1 + \zeta$$

and

$$\sigma(\zeta) = \beta_0 + \beta_1\zeta = 1,$$

which implies that

$$\rho(w+1) = w \quad \text{and} \quad \sigma(w+1) = 1.$$

Using the Taylor series expansion of the natural logarithm of $w+1$ about 1, it follows that

$$\begin{aligned}
\rho(w+1) - \sigma(w+1)\ln(w+1) &= w - \ln(w+1) \\
&= w - \left(w - \frac{1}{2}w^2 + \frac{1}{3}w^3 - \dots \right) \\
&= \frac{1}{2}w^2 + \mathcal{O}(|w|^3),
\end{aligned}$$

from which we conclude that the forward Euler method is of order $p = 1$ with local error constant $c = \dfrac{1}{2}$.

If the order p and the coefficients α_j, β_j of the LMM are known *a priori*, the local error constant c can be computed via the formula

$$c = \frac{1}{(p+1)!}\left(\sum_{j=0}^{r} j^{p+1}\alpha_j - (p+1)\sum_{j=0}^{r} j^p \beta_j \right) \tag{2.17}$$

which follows directly from (2.15). For the forward Euler scheme, which has order $p = 1$ and LMM coefficients $\alpha_1 = 1$, $\alpha_0 = -1$, $\beta_1 = 0$ and $\beta_0 = 1$, according to (2.17) the local error constant is

$$\begin{aligned}
c &= \frac{1}{2}\left(\sum_{j=0}^{1} j^2\alpha_j - 2\sum_{j=0}^{1} j\beta_j \right) \\
&= \frac{1}{2}\left(\alpha_1 - 2\beta_1 \right) \\
&= \frac{1}{2},
\end{aligned}$$

in agreement with what was previously found.

2.2.2 Consistency, zero-stability and convergence

An LMM is said to be *consistent* if the local error (2.8) goes to zero as $h \to 0$. Thus, consistency requires that

$$\sum_{j=0}^{r} \alpha_j = 0 \qquad \text{and} \qquad \sum_{j=0}^{r} j\alpha_j = \sum_{j=0}^{r} \beta_j, \tag{2.18}$$

which implies that the LMM is at least first-order accurate as $h \to 0$. This is a consequence of formulas (2.13) derived in the previous section. Any LMM of order $p \geq 1$ is automatically consistent. The consistency conditions can be expressed using the characteristic polynomials (2.12) of the LMM: Since

$$\rho(1) = \sum_{j=0}^{r} \alpha_j \qquad \text{and} \qquad \rho'(\zeta) = \sum_{j=0}^{r} j\alpha_j \zeta^{j-1},$$

it follows that conditions (2.18) hold if

$$\rho(1) = 0 \qquad \text{and} \qquad \rho'(1) = \sigma(1).$$

The characteristic polynomial ρ is also related to the zero-stability of an LMM, which is the type of stability needed to guarantee convergence as $h \to 0$. As motivation for the definition of zero-stability, recall that in order to apply an r-step LMM (2.6) to the IVP (2.4), the values $u_0, u_1, \ldots, u_{r-1}$ must be available. While u_0 is given as the initial condition, the values $u_1, \ldots, u_{r-1}$ must be determined by other means: we can either assume a constant history or compute these values using an appropriate one-step method, e.g., a Runge-Kutta method [12, 13, 14]. It is important to take into account how numerical errors introduced in these starting values affect the future approximations u_j for $j \geq r$. To this end, we require a type of stability with respect to small perturbations in the starting values.

Following [14], an r-step LMM is said to be *zero-stable* if there exists a constant M independent of the time step h such that, for any two sequences $\{u_j\}_{j \geq r}$ and $\{v_j\}_{j \geq r}$

generated by the same method but with different starting values $u_0, u_1, \ldots, u_{r-1}$ and $v_0, v_1, \ldots, v_{r-1}$, respectively, as $h \to 0$ we have

$$|u_j - v_j| \leq M \max\{|u_0 - v_0|, |u_1 - v_1|, \ldots, |u_{r-1} - v_{r-1}|\}$$

for $t_j \leq T$. Therefore, zero-stability can be thought of as a guarantee that perturbations in the starting values do not grow in an unbounded fashion as the method proceeds.

An algebraically equivalent definition of zero-stability follows from checking a condition on the roots of the characteristic polynomial ρ. More specifically, an r-step LMM is zero-stable if the roots of $\rho(\zeta)$ satisfy

$$|\zeta_j| \leq 1 \text{ for } j = 1, \ldots, r \quad \text{and} \quad |\zeta_j| < 1 \text{ for zeros of higher multiplicity.} \tag{2.19}$$

Condition (2.19) is often referred to as the *root condition*.

For the Adams methods (2.7) with general form

$$u_{n+r} = u_{n+r-1} + h \sum_{j=0}^{r} \beta_j f(u_{n+j})$$

and

$$\rho(\zeta) = \zeta^r - \zeta^{r-1} = (\zeta - 1)\zeta^{r-1},$$

the roots of ρ are $\zeta_1 = 1$ and $\zeta_2 = \cdots = \zeta_r = 0$, which clearly satisfy the root condition (2.19). Therefore all Adams methods (both AB and AM) are zero-stable. However, it can be shown that BDF methods are zero-stable only for $r \leq 6$.

In order for an LMM to be convergent in the sense that the numerical solution converges to the exact solution as $h \to 0$, it must be both consistent and zero-stable, as stated in the following theorem [12, 14, 15]:

Theorem 1. (**Dahlquist Equivalence Theorem**) *For LMMs applied to the initial value problem for $u'(t) = f(u(t), t)$, convergence is equivalent to consistency and zero-stability.*

2.2.3 Absolute stability

The characteristic polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ defined in (2.12) can be combined to define the *stability polynomial*

$$\pi(\zeta; z) := \rho(\zeta) - z\sigma(\zeta), \quad z, \zeta \in \mathbb{C}, \tag{2.20}$$

which we use to introduce a notion of stability, stronger than zero-stability, that is necessary for determining whether or not a numerical scheme will yield reasonable results for a given time step $h > 0$. Note that the stability polynomial appeared in disguise in the formula (2.11) with $z = \ln \zeta$.

The stability polynomial (2.20) is derived by applying the LMM (2.6) to the linear test problem (2.16), where $\lambda \in \mathbb{C}$ is an eigenvalue of A, yielding the rth-order linear recursion equation

$$\sum_{j=0}^{r} (\alpha_j - \lambda h \beta_j) u_{n+j} = 0,$$

with characteristic polynomial

$$\pi(\zeta; \lambda h) = \sum_{j=0}^{r} (\alpha_j - \lambda h \beta_j) \zeta^j. \tag{2.21}$$

By letting $z = \lambda h$ and writing (2.21) in terms of ρ and σ, we obtain the stability polynomial $\pi(\zeta; z)$ in (2.20) as desired. We remark that $\pi(\zeta; z)$ is a polynomial in ζ, but its coefficients depend on z.

An LMM is said to be *absolutely stable* for a particular value of $z \in \mathbb{C}$ if an error introduced in a single time step does not accumulate in subsequent steps. This can be expressed as a condition about the roots of the stability polynomial. The *absolute stability region* of an LMM is the set of points z in the complex plane for which the zeros of the polynomial $\zeta \mapsto \pi(\zeta; z)$ satisfy the root condition (2.19):

$$|\zeta_j| \leq 1 \text{ for } 1 \leq j \leq r \quad \text{and} \quad |\zeta_j| < 1 \text{ for zeros of higher multiplicity.}$$

In other words, the set

$$\Omega = \{z \in \mathbb{C} : \pi(\zeta; z) \text{ satisfies the root condition } \} \subset \mathbb{C}$$

is the stability region of the method. An LMM is absolutely stable within the region Ω and unstable elsewhere. Note that an LMM is zero-stable if and only if $z = 0$ lies in the absolute stability region of the LMM.

For example, the stability polynomial for the forward Euler scheme,

$$\pi(\zeta; z) = \zeta - (1 + z),$$

has the root $\zeta = 1 + z$. Thus the absolute stability region of forward Euler is the set of $z \in \mathbb{C}$ for which the condition

$$|1 + z| \leq 1$$

holds, which is a closed disk in the complex plane of radius 1 centered at the point $z = -1$. This region is shown in Figure 2.1.

The size and shape of the absolute stability regions of the AB, AM, and BDF methods are important factors to be taken into consideration when choosing a numerical ODE solver, because they determine the maximum length of the time step. When solving a system of ODEs, the points $z \in \mathbb{C}$ are the products of the time step h and the eigenvalues λ of the constant-coefficient matrix A of the system in the linear case, or the Jacobian $\mathcal{D}_u f$ of f in the nonlinear case. Thus, in order for the numerical integrator to be stable, the product of every eigenvalue of the Jacobian and the time step h must fall inside the stability region of the method. The stability regions are plotted in the complex plane to account for the fact that the eigenvalues of the ODE system may take on complex values. For further details, see [12, 13, 14].

2.2.4 Adams-Bashforth methods

An r-step AB method is of order r and has local one-step error $\mathcal{O}(h^{r+1})$. The first four AB methods, which we denote by AB1-AB4, are given in Table 2.1, where, for simplicity, the formulas are written for the autonomous case when $f(u, t) = f(u)$, with

	Adams-Bashforth Method	Local Error	Error Constant
1-step:	$u_{n+1} = u_n + hf(u_n)$	$\mathcal{O}(h^2)$	$\frac{1}{2}$
2-step:	$u_{n+2} = u_{n+1} + \frac{h}{2}\big[-f(u_n) + 3f(u_{n+1})\big]$	$\mathcal{O}(h^3)$	$\frac{5}{12}$
3-step:	$u_{n+3} = u_{n+2} + \frac{h}{12}\big[5f(u_n) - 16f(u_{n+1}) + 23f(u_{n+2})\big]$	$\mathcal{O}(h^4)$	$\frac{3}{8}$
4-step:	$u_{n+4} = u_{n+3} + \frac{h}{24}\big[-9f(u_n) + 37f(u_{n+1}) - 59f(u_{n+2}) + 55f(u_{n+3})\big]$	$\mathcal{O}(h^5)$	$\frac{251}{720}$

Table 2.1: The first four AB methods.

the understanding that they can be generalized by setting $f(u_{n+j}) = f(u_{n+j}, t_{n+j})$. As previously mentioned, the one-step AB method is the forward Euler method. While the main appeal of the AB methods is that they are explicit, the absolute stability regions of AB1-AB4, as shown in Figure 2.1, become smaller as the order of the method increases.

2.2.5 Adams-Moulton methods

An r-step AM method has order $r + 1$; hence its local one-step error is $\mathcal{O}(h^{r+2})$. The formulas for the first four AM methods, which we denote by AM1-AM4, are presented in Table 2.2. In particular, the one-step AM method is the trapezoidal method. The AM methods are implicit, and the absolute stability regions for AM1-AM4, shown in Figure 2.1, are much larger than those of the AB rules, thus making this family better suited for solving stiff problems, where the eigenvalues of the Jacobian of f span several orders of magnitude.

	Adams-Moulton Method	Local Error	Error Constant
1-step:	$u_{n+1} = u_n + \frac{h}{2}\big[f(u_n) + f(u_{n+1})\big]$	$\mathcal{O}(h^3)$	$-\frac{1}{12}$
2-step:	$u_{n+2} = u_{n+1} + \frac{h}{12}\big[-f(u_n) + 8f(u_{n+1}) + 5f(u_{n+2})\big]$	$\mathcal{O}(h^4)$	$-\frac{1}{24}$
3-step:	$u_{n+3} = u_{n+2} + \frac{h}{24}\big[f(u_n) - 5f(u_{n+1}) + 19f(u_{n+2}) + 9f(u_{n+3})\big]$	$\mathcal{O}(h^5)$	$-\frac{19}{720}$
4-step:	$u_{n+4} = u_{n+3} + \frac{h}{720}\big[-19f(u_n) + 106f(u_{n+1}) - 264f(u_{n+2}) + 646f(u_{n+3}) + 251f(u_{n+4})\big]$	$\mathcal{O}(h^6)$	$-\frac{3}{160}$

Table 2.2: The first four AM methods.

2.2.6 BDF methods

An r-step BDF method is of order r and has local one-step error $\mathcal{O}(h^{r+1})$. The first four BDF methods, which we denote by BDF1-BDF4, are listed in Table 2.3. The one-step BDF method gives the backward Euler method. Like the AM methods, the BDF methods are implicit. The absolute stability regions of BDF1-BDF4, shown in Figure 2.1, are the largest of the three families considered, thus suggesting that the BDF methods are particularly well-suited for very stiff systems.

2.3 Error control

Estimating and controlling the local numerical error in the solution of (2.4) with LMMs is a central topic in the numerical solution of differential equations. In the context of particle filters and ensemble Kalman filters, the estimate of the local error will be used to assign the variance of the innovation term and is therefore

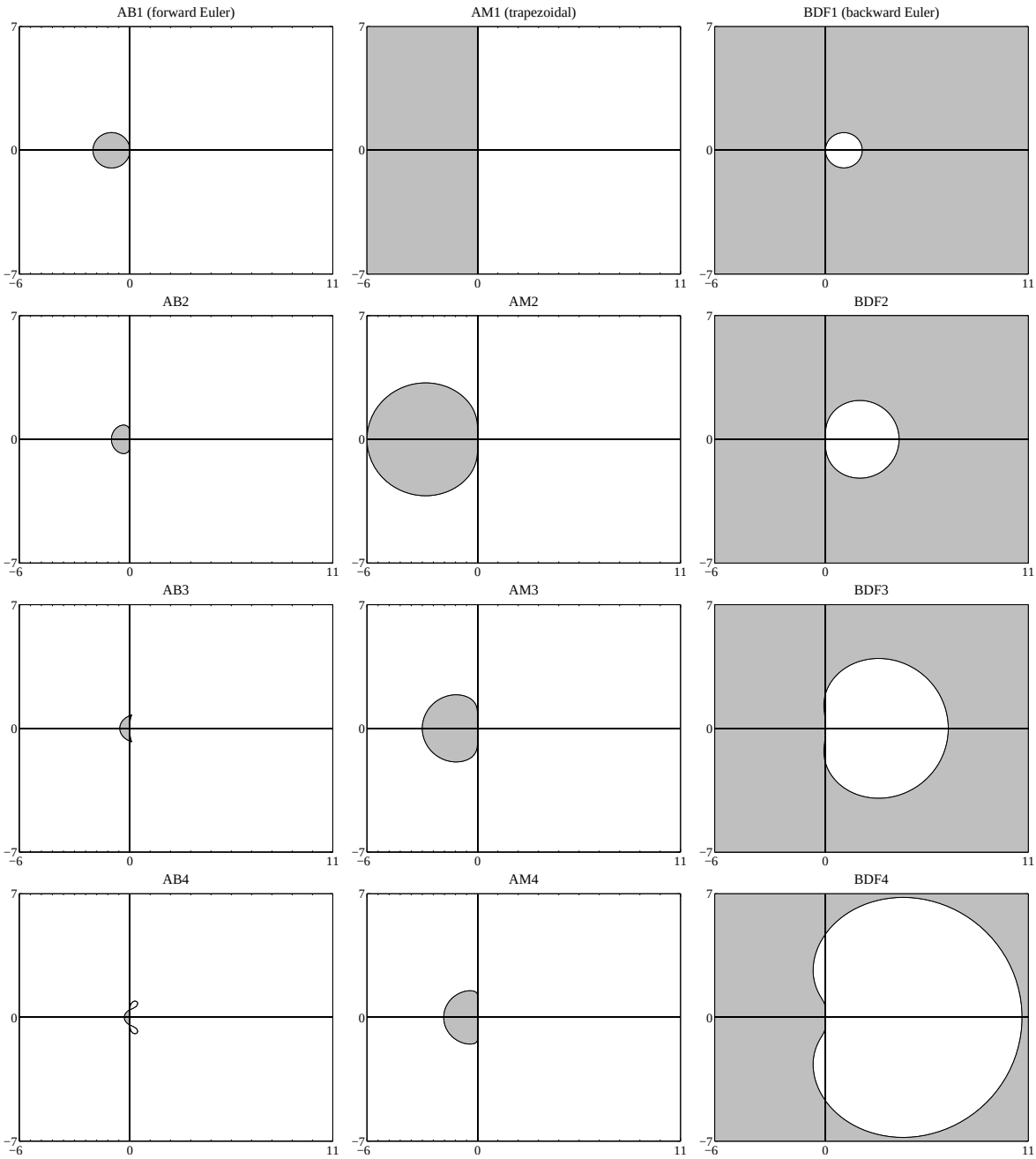

Figure 2.1: Absolute stability regions for the first four methods of three families of linear multistep methods: AB (left), AM (center), BDF (right). The shaded region (including the boundary) in each plot is the stable region.

	BDF Method	Local Error	Error Constant
1-step:	$u_{n+1} - u_n = hf(u_{n+1})$	$\mathcal{O}(h^2)$	$-\frac{1}{2}$
2-step:	$3u_{n+2} - 4u_{n+1} + u_n = 2hf(u_{n+2})$	$\mathcal{O}(h^3)$	$-\frac{2}{9}$
3-step:	$11u_{n+3} - 18u_{n+2} + 9u_{n+1} - 2u_n$	$\mathcal{O}(h^4)$	$-\frac{3}{22}$
	$= 6hf(u_{n+3})$		
4-step:	$25u_{n+4} - 48u_{n+3} + 36u_{n+2} - 16u_{n+1} + 3u_n$	$\mathcal{O}(h^5)$	$-\frac{12}{125}$
	$= 12hf(u_{n+4})$		

Table 2.3: The first four BDF methods.

of particular interest in this thesis. There are two classic techniques for estimating the local truncation error: one which employs an LMM of higher order to monitor the error, and another which couples two LMMs of the same order but with different error constants. These methods are given below; we will describe in Chapter 5 how the error estimates can be linked to the covariance matrix of the innovation term in filtering settings.

2.3.1 Higher order method error control

In the first local error estimation method, which follows the paradigm for Runge-Kutta methods described in, e.g., [13], the numerical solution of the ODE is computed with an LMM of order p and is concurrently computed to higher accuracy with an LMM of order $\widetilde{p} \geq p + 1$ from the same family. The difference between the results of the two integrators is attributed to the local error introduced by the lower accuracy method and is used as an error estimate. We call this technique *higher order method error control* (HOMEC). For simplicity, we only consider methods of order $\widetilde{p} = p + 1$ for error control. Thus if we propagate, e.g., with the 2-step BDF method of order 2,

we use the 3-step BDF method of order 3 to estimate the error.

Let u_{n+r} denote the candidate solution at time t_{n+r} computed by the LMM of order p, and let $\widehat{u}_{n+r}$ be the solution at t_{n+r} given by the higher order method. If u is the exact solution with initial value $u(t_n) = u_n$, then

$$\begin{aligned} u_{n+r} &= u(t_{n+r}) + \ell h^{p+1} + \mathcal{O}(h^{p+2}) \\ \widehat{u}_{n+r} &= u(t_{n+r}) + \mathcal{O}(h^{p+2}) \end{aligned}$$

as $h \to 0$, where ℓ is some vector depending on the solution $u(t)$ of (2.4) and on the propagation scheme but not on the time step h. Subtracting the equations side to side and neglecting higher order terms yields

$$\ell h^{p+1} \approx u_{n+r} - \widehat{u}_{n+r}, \tag{2.22}$$

which suggests that the absolute value of the difference (2.22) can be used to estimate the dominant part of the error in u_{n+r} at t_{n+r}.

2.3.2 The Milne device

A second method for estimating the local error, known as the *Milne device* [13], approximates the local error of an LMM by comparing its solution with that computed by another LMM of the same order but with a different error constant.

Let

$$\sum_{j=0}^{r} \alpha_j u_{n+j} = h \sum_{j=0}^{r} \beta_j f(u_{n+j}, t_{n+j}), \quad n = 0, 1, \ldots, \quad \alpha_r = 1 \tag{2.23}$$

be a convergent LMM of order p, and let

$$\sum_{j=q}^{r} \widetilde{\alpha}_j \widetilde{u}_{n+j} = h \sum_{j=q}^{r} \widetilde{\beta}_j f(\widetilde{u}_{n+j}, t_{n+j}), \quad n \geq \max\{0, -q\}, \quad \widetilde{\alpha}_r = 1 \tag{2.24}$$

be another convergent LMM of the same order. Here $q \leq r-1$ is an integer of either sign, and $\widetilde{u}_{n+j} = u_{n+j}$ for $j = \min\{0, q\},\ \min\{0, q\} + 1,\ \ldots, r-1$.

Assuming $u_n, u_{n+1}, \ldots, u_{n+r-1}$ satisfy (2.23) exactly, it follows from (2.11) that

$$u(t_{n+r}) - u_{n+r} = ch^{p+1}u^{(p+1)}(t_{n+r}) + \mathcal{O}(h^{p+2}), \qquad h \to 0, \tag{2.25}$$

and, similarly for (2.24),

$$u(t_{n+r}) - \widetilde{u}_{n+r} = \widetilde{c}h^{p+1}u^{(p+1)}(t_{n+r}) + \mathcal{O}(h^{p+2}), \qquad h \to 0, \tag{2.26}$$

where c and $\widetilde{c}$, $\widetilde{c} \neq c$, are the error constants of the methods. Subtracting (2.26) from (2.25) side to side and ignoring higher order terms, we get

$$\widetilde{u}_{n+r} - u_{n+r} \approx (c - \widetilde{c})h^{p+1}u^{(p+1)}(t_{n+r}),$$

and thus

$$h^{p+1}u^{(p+1)}(t_{n+r}) \approx \frac{1}{c - \widetilde{c}}(\widetilde{u}_{n+r} - u_{n+r}).$$

The latter expression, when substituted into (2.25), yields

$$u(t_{n+r}) - u_{n+r} \approx \frac{c}{c - \widetilde{c}}(\widetilde{u}_{n+r} - u_{n+r}), \tag{2.27}$$

which provides an estimate for the local error. In general, we want $|\widetilde{c}| < |c|$, so that the method used for error control is more accurate than the method used for propagation. The key quantity in the Milne device estimate (2.27) is

$$\kappa = \left|\frac{c}{c - \widetilde{c}}\right|, \tag{2.28}$$

which can be computed using the error constants listed in Tables 2.1, 2.2 and 2.3.

2.4 Variable time step LMMs

The coefficients of the LMM formulas given in Tables 2.1, 2.2 and 2.3 in Section 2.2 are derived assuming that the time step h is *fixed* and does not change as we march forward in time, i.e., $t_n = nh$ for $n \geq 0$ and fixed $h > 0$. In some applications, however, as will be seen with the PET problem in Chapter 7, it is advantageous

to allow the time step to change over time in order to guarantee that stability is maintained. Here we derive the general LMM formulas for the Adams and BDF methods using polynomial interpolation, and we describe how to compute the LMM coefficients over a given interval with variable time steps.

2.4.1 Adams methods with variable time step

To derive the formula for the general Adams method of order r, consider the IVP (2.4) and discrete time instances

$$t_n < t_{n+1} < \cdots < t_{n+r},$$

not necessarily evenly spaced. It follows from the fundamental theorem of calculus that

$$u(t_{n+r}) = u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} f(t, u(t))dt.$$

Assume that the approximate values $u_j \approx u(t_j)$ are given for $j = n, n+1, \ldots, n+r-1$ and let $f_j = f(t_j, u_j)$.

To obtain the explicit Adams method of order r via polynomial interpolation, we require that the interpolation polynomial $P(t)$ of order $r-1$, $P(t) \in \mathbb{P}_{r-1}$ where $\mathbb{P}_{r-1}$ denotes the set of all polynomial functions of order $r-1$, satisfies the r conditions

$$P(t_j) = f_j \quad \text{for} \quad j = n, n+1, \ldots, n+r-1.$$

We can explicitly construct $P(t)$ using a basis of Lagrange polynomials p_j of order $r-1$,

$$p_j(t) = \prod_{\substack{\ell=n \\ \ell \neq j}}^{n+r-1} \frac{t - t_\ell}{t_j - t_\ell}, \qquad j = n, n+1, \ldots, n+r-1,$$

yielding

$$P(t) = \sum_{j=n}^{n+r-1} f_j p_j(t).$$

The explicit Adams formula for variable time step is then found by replacing $f(t, u(t))$ by its polynomial approximation,

$$\begin{aligned} u_{n+r} &= u_{n+r-1} + \int_{t_{n+r-1}}^{t_{n+r}} f(t, u(t))dt \\ &\approx u_{n+r-1} + \int_{t_{n+r-1}}^{t_{n+r}} P(t)dt \\ &= u_{n+r-1} + \sum_{j=n}^{n+r-1} f_j \int_{t_{n+r-1}}^{t_{n+r}} p_j(t)dt \\ &= u_{n+r-1} + \sum_{j=0}^{r-1} \beta_j^{n+r} f_{n+j}, \end{aligned}$$

where

$$\beta_j^{n+r} = \int_{t_{n+r-1}}^{t_{n+r}} p_{n+j}(t)dt$$

for $j = 0, \ldots, r-1$.

To derive the implicit Adams method of order r, we define a polynomial $P(t)$ of order r, $P(t) \in \mathbb{P}_r$, satisfying the $r+1$ conditions

$$P(t_j) = f_j \quad \text{for} \quad j = n, n+1, \ldots, n+r$$

and express it in terms of a basis of Lagrange polynomials

$$p_j(t) = \prod_{\substack{\ell=n \\ \ell \neq j}}^{n+r} \frac{t - t_\ell}{t_j - t_\ell}, \quad j = n, n+1, \ldots, n+r, \tag{2.29}$$

which yields

$$P(t) = \sum_{j=n}^{n+r} f_j p_j(t).$$

The implicit formula for variable time step is then given by

$$u_{n+r} = u_{n+r-1} + \sum_{j=0}^{r} \beta_j^{n+r} f_{n+j},$$

where

$$\beta_j^{n+r} = \int_{t_{n+r-1}}^{t_{n+r}} p_{n+j}(t)dt$$

for $j = 0, \ldots, r$.

When the time step is fixed, the coefficients β_j^{n+r} do not depend on the time interval indexed by $n + r$, and the fixed time step h may be factored out, yielding formula (2.7). However, when the time step varies, the coefficients in general do depend on $n + r$ and must be recomputed at each step.

In practice, to compute the β_j^{n+r} coefficients for the explicit Adams method of order r, we write the interpolation polynomial $P(t) \in \mathbb{P}_{r-1}$ in the form

$$P(t) = \sum_{\ell=0}^{r-1} \gamma_\ell (t_{n+r} - t)^\ell,$$

and we require that

$$P(t_j) = f_j \quad \text{for} \quad j = n, n+1, \ldots, n+r-1.$$

Letting

$$d_k = t_{n+r} - t_{n+r-k} \quad \text{for} \quad k = 1, 2, \ldots, r,$$

where the dependency of d_k on $n + r$ is suppressed to simplify the notation, the r conditions can be written as

$$\begin{aligned} P(t_{n+r-k}) &= \sum_{\ell=0}^{r-1} \gamma_\ell (t_{n+r} - t_{n+r-k})^\ell \\ &= \sum_{\ell=0}^{r-1} \gamma_\ell d_k^\ell \\ &= f_{n+r-k}, \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} 1 & d_r & d_r^2 & \cdots & d_r^{r-1} \\ 1 & d_{r-1} & d_{r-1}^2 & \cdots & d_{r-1}^{r-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & d_1 & d_1^2 & \cdots & d_1^{r-1} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{r-1} \end{bmatrix} = \begin{bmatrix} f_n \\ f_{n+1} \\ \vdots \\ f_{n+r-1} \end{bmatrix},$$

which we write in matrix notation concisely as

$$\mathsf{H}\boldsymbol{\gamma} = \mathbf{f}$$

with

$$\mathsf{H} \in \mathbb{R}^{r\times r}, \quad \boldsymbol{\gamma} \in \mathbb{R}^r, \quad \text{and} \quad \mathbf{f} \in \mathbb{R}^r.$$

The solution of this linear system, $\boldsymbol{\gamma} = \mathsf{H}^{-1}\mathbf{f}$, gives the coefficients γ_ℓ of the interpolation polynomial. Using this in the polynomial approximation of f, we have

$$\begin{aligned} \int_{t_{n+r-1}}^{t_{n+r}} f(t,u(t))dt &\approx \int_{t_{n+r-1}}^{t_{n+r}} P(t)dt \\ &= \sum_{\ell=0}^{r-1} \gamma_\ell \int_{t_{n+r-1}}^{t_{n+r}} (t_{n+r}-t)^\ell dt \\ &= \sum_{\ell=0}^{r-1} \gamma_\ell \frac{1}{\ell+1} d_1^{\ell+1}. \end{aligned}$$

Defining

$$q_\ell = \frac{1}{\ell+1} d_1^{\ell+1}$$

and

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{r-1} \end{bmatrix} \in \mathbb{R}^r,$$

we may write

$$\begin{aligned}\int_{t_{n+r-1}}^{t_{n+r}} f(t,u(t))dt &\approx \sum_{\ell=0}^{r-1}\gamma_\ell q_\ell \\ &= \mathbf{q}^{\mathsf{T}}\boldsymbol{\gamma} \\ &= \mathbf{q}^{\mathsf{T}}\mathsf{H}^{-1}\mathbf{f} \\ &= (\mathsf{H}^{-\mathsf{T}}\mathbf{q})^{\mathsf{T}}\mathbf{f},\end{aligned}$$

where $\mathsf{H}^{-\mathsf{T}}$ denotes the inverse of the transpose of H. Therefore,

$$\begin{aligned}u_{n+r} &= u_{n+r-1} + \int_{t_{n+r-1}}^{t_{n+r}} f(t,u(t))dt \\ &\approx u_{n+r-1} + (\mathsf{H}^{-\mathsf{T}}\mathbf{q})^{\mathsf{T}}\mathbf{f} \\ &= u_{n+r-1} + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{f},\end{aligned}$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{r-1} \end{bmatrix} = \mathsf{H}^{-\mathsf{T}}\mathbf{q} \in \mathbb{R}^r. \tag{2.30}$$

Therefore the vector (2.30) contains the β_j^{n+r} coefficients of the explicit Adams formula

$$u_{n+r} = u_{n+r-1} + \sum_{j=0}^{r-1}\beta_j^{n+r} f_{n+j}.$$

The β_j^{n+r} coefficients for the implicit Adams method of order r are computed in a similar manner, with the modification that the polynomial $P(t) \in \mathbb{P}_r$, written in the form

$$P(t) = \sum_{\ell=0}^{r}\gamma_\ell (t_{n+r} - t)^\ell,$$

must satisfy the additional condition

$$P(t_{n+r}) = f_{n+r}.$$

Therefore $\boldsymbol{\gamma} \in \mathbb{R}^{r+1}$, $\mathbf{f} \in \mathbb{R}^{r+1}$ and the matrix $\mathsf{H} \in \mathbb{R}^{(r+1)\times(r+1)}$ becomes

$$\mathsf{H} = \begin{bmatrix} 1 & d_r & d_r^2 & \cdots & d_r^{r-1} & d_r^r \\ 1 & d_{r-1} & d_{r-1}^2 & \cdots & d_{r-1}^{r-1} & d_{r-1}^r \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & d_1 & d_1^2 & \cdots & d_1^{r-1} & d_1^r \\ 1 & d_0 & d_0^2 & \cdots & d_0^{r-1} & d_0^r \end{bmatrix}$$

with the additional row and column. Note that $d_0 = 0$. The β_j^{n+r} coefficients of the implicit Adams formula

$$u_{n+r} = u_{n+r-1} + \sum_{j=0}^{r} \beta_j^{n+r} f_{n+j}$$

are thus given by the entries of the vector

$$\boldsymbol{\beta} = \mathsf{H}^{-\mathsf{T}}\mathbf{q} \in \mathbb{R}^{r+1}.$$

When the time step is fixed, this procedure will yield the coefficients given in Tables 2.1 and 2.2.

2.4.2 BDF methods with variable time step

The BDF method of order r is derived by approximating $u(t)$ rather than $f(t, u(t))$ by a polynomial. Suppose that the approximate values $u_j \approx u(t_j)$ are given for $j = n, n+1, \ldots, n+r$, and let $Q(t)$ be a polynomial of order r satisfying the $r+1$ conditions

$$Q(t_j) = u_j \quad \text{for} \quad j = n, n+1, \ldots, n+r.$$

The BDF formula for u_j is obtained by requiring

$$Q'(t_j) = f(t_j, u_j), \quad j = n, n+1, \ldots, n+r$$

so that

$$\begin{aligned} u(t_{n+r}) &= u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} f(t, u(t))dt \\ &\approx u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} Q'(t)dt. \end{aligned}$$

Using the Lagrange polynomials (2.29) as a basis, as for the derivation of the implicit Adams methods, the interpolation polynomial $Q(t)$ can be written as

$$Q(t) = \sum_{j=n}^{n+r} u_j p_j(t),$$

from which it follows that

$$Q'(t) = \sum_{j=n}^{n+r} u_j p_j'(t).$$

The general BDF formula then becomes

$$\sum_{j=n}^{n+r} u_j p_j'(t_j) = f(t_j, u_j),$$

or, adhering to the form used in the LMM literature,

$$\sum_{j=0}^{r} \alpha_j^n u_{n+j} = f(t_j, u_j),$$

where $\alpha_j^n = p_{n+j}'(t_j)$. When the time step is fixed and equal to h, it can be written as a factor multiplying the righthand side $f(t_j, u_j)$, as it appears in the formulas in Table 2.3.

In practice, to compute the coefficients α_j^n of the BDF formula of order r for variable time step, we write the interpolation polynomial $Q(t) \in \mathbb{P}_r$ as

$$Q(t) = \sum_{\ell=0}^{r} a_\ell (t_{n+r} - t)^\ell,$$

and we require that

$$Q(t_j) = u_j \quad \text{for} \quad j = n, n+1, \ldots, n+r.$$

Letting

$$d_k = t_{n+r} - t_{n+r-k} \quad \text{for} \quad k = 0, 1, 2, \ldots, r$$

and suppressing the dependency of d_k on $n+r$ for simplicity, the $r+1$ conditions become

$$\begin{aligned} Q(t_{n+r-k}) &= \sum_{\ell=0}^{r} a_\ell (t_{n+r} - t_{n+r-k})^\ell \\ &= \sum_{\ell=0}^{r} a_\ell d_k^\ell \\ &= u_{n+r-k}, \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} 1 & d_r & d_r^2 & \cdots & d_r^r \\ 1 & d_{r-1} & d_{r-1}^2 & \cdots & d_{r-1}^r \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & d_0 & d_0^2 & \cdots & d_0^r \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_r \end{bmatrix} = \begin{bmatrix} u_n \\ u_{n+1} \\ \vdots \\ u_{n+r} \end{bmatrix},$$

which we denote by

$$\mathsf{H}\mathbf{a} = \mathbf{u},$$

where

$$\mathsf{H} \in \mathbb{R}^{(r+1)\times(r+1)}, \quad \mathbf{a} \in \mathbb{R}^{r+1}, \quad \text{and} \quad \mathbf{u} \in \mathbb{R}^{r+1}.$$

Solving this linear system gives the coefficients a_ℓ of $Q(t)$, namely, $\mathbf{a} = \mathsf{H}^{-1}\mathbf{u}$. Since

$$Q'(t) = -\sum_{\ell=1}^{r} \ell a_\ell (t_{n+r} - t)^{\ell-1}$$

and, in particular,

$$\begin{aligned} Q'(t_{n+r}) &= -a_1 \\ &= -\mathbf{e}_2^{\mathsf{T}}\mathbf{a} \\ &= -\mathbf{e}_2^{\mathsf{T}}\mathsf{H}^{-1}\mathbf{u} \\ &= -(\mathsf{H}^{-\mathsf{T}}\mathbf{e}_2)^{\mathsf{T}}\mathbf{u} \end{aligned}$$

where

$$\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{r+1},$$

it follows that the α_j^n coefficients in the variable time step BDF formula

$$\sum_{j=0}^{r} \alpha_j^n u_{n+j} = f(t_j, u_j)$$

are given by

$$\boldsymbol{\alpha} = -\mathsf{H}^{-\mathsf{T}}\mathbf{e}_2 \in \mathbb{R}^{r+1}.$$

Note that if the time step is fixed, this procedure will yield the coefficients given in Table (2.3) with proper factoring of the fixed time step h.

Chapter 3

Statistical Background

As noted in Section 1.1.1, inverse problems can be reformulated as problems of statistical inference through use of Bayesian statistics. In this chapter we review the statistical concepts and results most relevant to the development of filtering and parameter estimation algorithms within a Bayesian inference framework. Some basic notions from probability and statistics are given in Section 3.1, and the Bayesian solution to inverse problems in formulated in Section 3.2. Monte Carlo integration is discussed in Section 3.3, motivating the MCMC sampling and Metropolis-Hastings algorithm described in Section 3.4.

3.1 Basic statistical notions

We begin with some basic concepts from probability theory, leading to the definition of a random variable and related notions. Here we primarily follow the notation of [7, 16], referring to [17, 18, 19, 20] for a more thorough treatment.

Let Ω be a given set. We say that a nonempty collection $\mathcal{F}$ of subsets of Ω is a *σ-algebra* on Ω if it has the following properties:

(i) $\emptyset \in \mathcal{F}$

(ii) If $F \in \mathcal{F}$, then $F^c \in \mathcal{F}$ where $F^c = \Omega \backslash F = \{\omega : \omega \in \Omega \text{ and } \omega \notin F\}$ is the *complement* of F in Ω

(iii) If $A_1, A_2, \cdots \in \mathcal{F}$, then $A := \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$

We call property (ii) closure under complements and property (iii) closure under countable unions.

The pair $(\Omega, \mathcal{F})$ is called a *measurable space*. A *probability measure* P on a measurable space $(\Omega, \mathcal{F})$ is a function

$$P : \mathcal{F} \to [0, 1]$$

such that

(a) $P(\emptyset) = 0, \quad P(\Omega) = 1;$

(b) If $\{A_i\}_{i=1}^{\infty}$ is a collection of mutually disjoint sets $A_i \in \mathcal{F}$, i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$, then

$$P\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} P(A_i).$$

The triple $(\Omega, \mathcal{F}, P)$ is called a *probability space* and is said to be *complete* if $\mathcal{F}$ contains all subsets G of Ω with P-outer measure zero, i.e.,

$$P^*(G) := \inf\{P(F) : F \in \mathcal{F}, G \subset F\} = 0.$$

Note that any probability space can be made complete by adding all sets of outer measure 0 to $\mathcal{F}$ and extending P accordingly.

The subsets F of Ω belonging to $\mathcal{F}$ are called $\mathcal{F}$-*measurable* sets, or *events* in a probability setting, in which case $P(F)$ is interpreted as the probability that the event F occurs. We require that $0 \leq P(F) \leq 1$, and if $P(F) = 1$, we say that F occurs with probability 1 or *almost surely* (a.s.).

For any family of subsets $\mathcal{U}$ of Ω, there is a unique smallest σ-algebra $\mathcal{M}_{\mathcal{U}}$ containing $\mathcal{U}$, namely, the intersection of all σ-algebras containing $\mathcal{U}$:

$$\mathcal{M}_{\mathcal{U}} = \bigcap\{\mathcal{M} : \mathcal{M} \text{ is a } \sigma\text{-algebra of } \Omega,\ \mathcal{U} \subset \Omega\}.$$

Note that the intersection of any family of σ-algebras on Ω is again a σ-algebra. $\mathcal{M}_{\mathcal{U}}$ is called the σ-algebra *generated by* $\mathcal{U}$.

If Ω is any topological space, e.g., $\Omega = \mathbb{R}^n$, the σ-algebra generated by the family of open sets in Ω or, equivalently, by the family of closed sets in Ω, is called the *Borel σ-algebra* on Ω and is denoted $\mathcal{B}$. The elements $B \in \mathcal{B}$ are called *Borel sets*. It follows from the properties of σ-algebras that $\mathcal{B}$ contains all open sets, closed sets, countable intersections of open sets, and countable unions of closed sets.

Given a probability space $(\Omega, \mathcal{F}, P)$, a function $Y : \Omega \to \mathbb{R}^n$ is said to be $\mathcal{F}$-*measurable* if

$$Y^{-1}(U) := \{\omega \in \Omega : Y(\omega) \in U\} \in \mathcal{F}$$

for all open sets $U \subset \mathbb{R}^n$, or, equivalently, for all Borel sets $U \subset \mathbb{R}^n$.

We are now ready to give the definition of a random variable. Let $(\Omega, \mathcal{F}, P)$ denote a complete probability space. A *(multivariate) random variable* X is an $\mathcal{F}$-measurable function

$$X : \Omega \to \mathbb{R}^n$$

that assigns to each element ω of Ω an n-dimensional real-valued vector $x = X(\omega)$ called a *realization* of X. Note that when $n = 1$, $X : \Omega \to \mathbb{R}$ and the random variable is said to be *univariate*. In the remainder of this thesis, we denote random variables by capital letters and their realizations by the corresponding lower case letters.

Every $\mathbb{R}^n$-valued random variable induces a probability measure μ_X on $\mathbb{R}^n$, defined by

$$\mu_X(B) = P(X^{-1}(B)) = P(X(\omega) \in B)$$

and called the *(probability) distribution* of X. Thus $\mu_X(B)$ is the probability of the event $x = X(\omega) \in B$.

Assuming that μ_X is absolutely continuous with respect to the Lebesgue measure, we can write the distribution of X as

$$\mu_X(B) = \int_B d\mu_X(x) = \int_B \pi_X(x)dx,$$

where π_X, the *probability density function* (pdf) of X, is a nonnegative function such that

$$\int_{\mathbb{R}^n} \pi_X(x)dx = 1$$

over the entire range of X. When the dependence on X is clear, we simply write

$$\pi_X(x) = \pi(x).$$

The *joint probability distribution* of two random variables $X, Y : \Omega \to \mathbb{R}^n$ is, by definition,

$$P(X \in A, Y \in B) = P(X^{-1}(A) \cap Y^{-1}(B)) = \mu_{XY}(A, B);$$

if μ_{XY} is absolutely continuous with respect to the Lebesgue measure over $\mathbb{R}^n \times \mathbb{R}^n = \mathbb{R}^{2n}$, we may write

$$\mu_{XY}(A, B) = \int \int_{A \times B} \pi_{XY}(x, y)dxdy.$$

The joint probability expresses the probability of the event that $x \in A$ and, simultaneously, $y \in B$, where $\pi_{XY}(x, y) = \pi(x, y)$ is the *joint pdf* of X and Y. Note that this is called a *bivariate distribution* in the case of two random variables, but the concept may be generalized to account for any number of random variables; thus the pdf of a multivariate random variable X is the joint pdf of its components. In the sequel, we assume the probability distributions to be absolutely continuous with respect to the Lebesgue measure.

Given two random variables X and Y with joint pdf $\pi(x, y)$, the *marginal density* of X, i.e., the probability of X regardless of the value of Y, is obtained by marginalizing the joint pdf with respect to X,

$$\pi(x) = \int_{\mathbb{R}^n} \pi(x, y)dy.$$

Likewise, the marginal density of Y is given by

$$\pi(y) = \int_{\mathbb{R}^n} \pi(x, y)dx.$$

Two random variables $X, Y : \Omega \to \mathbb{R}^n$ are said to be *independent* if their joint pdf is equal to the product of their respective marginal densities, i.e.,

$$\pi(x, y) = \pi(x)\pi(y).$$

The *conditional probability density* of X given Y,

$$\pi(x \mid y) = \frac{\pi(x, y)}{\pi(y)}, \quad \pi(y) \neq 0, \tag{3.1}$$

is the probability density of X assuming $Y = y$. It follows from the symmetry of the roles of X and Y that

$$\pi(x, y) = \pi(x \mid y)\pi(y) = \pi(y \mid x)\pi(x), \tag{3.2}$$

an important identity used in deriving *Bayes' theorem*, a fundamental result which will be discussed in Section 3.2.

If $\int_\Omega |X(\omega)| dP(\omega) < \infty$, then we define the *expectation*, or *mean*, of X with respect to the probability measure P as

$$E[X] := \int_\Omega X(\omega) dP(\omega) = \int_{\mathbb{R}^n} x d\mu_X(x) = \int_{\mathbb{R}^n} x\pi(x) dx,$$

which is sometimes denoted $\overline{x} = E[X] \in \mathbb{R}^n$. Note that the expectation operator E is linear, that is, for any two random variables $X, Y : \Omega \to \mathbb{R}^n$ and constants $a, b \in \mathbb{R}$,

$$E[aX + bY] = aE[X] + bE[Y],$$

which follows immediately from the definition

$$\begin{aligned} E[aX + bY] &= \int_\Omega (aX(\omega) + bY(\omega)) dP(\omega) \\ &= aE[X] + bE[Y] \end{aligned}$$

and the linearity of integration.

In general, if $f : \mathbb{R}^n \to \mathbb{R}$ is Borel measurable and $\int_\Omega |f(X(\omega))| dP(\omega) < \infty$, then we have that

$$E[f(X)] := \int_\Omega f(X(\omega)) dP(\omega) = \int_{\mathbb{R}^n} f(x) d\mu_X(x) = \int_{\mathbb{R}^n} f(x)\pi(x)dx.$$

Given a univariate random variable $X : \Omega \to \mathbb{R}$, we define its *variance* to be the expectation of the squared deviation from the expectation,

$$\text{var}(X) = \sigma_X^2 = E[(X - \overline{x})^2] = \int_{\mathbb{R}} (x - \overline{x})^2 \pi(x) dx \in \mathbb{R}.$$

The variance measures how widely the values taken on by the random variable are distributed around the expectation.

If $X, Y : \Omega \to \mathbb{R}$ are univariate random variables, the *covariance* of X and Y is defined as

$$\text{cov}(X, Y) = E[(X - \overline{x})(Y - \overline{y})],$$

or, equivalently,

$$\text{cov}(X, Y) = E[XY] - E[X]E[Y],$$

since

$$\begin{aligned} E[(X - \overline{x})(Y - \overline{y})] &= E[XY - \overline{x}Y - \overline{y}X + \overline{x}\ \overline{y}] \\ &= E[XY] - \overline{x}E[Y] - \overline{y}E[X] + \overline{x}\ \overline{y} \\ &= E[XY] - E[X]E[Y] - E[Y]E[X] + E[X]E[Y] \\ &= E[XY] - E[X]E[Y]. \end{aligned}$$

Two univariate random variables X and Y with non-vanishing variance have *correlation coefficient*

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

where

$$\sigma_X = \sqrt{\text{var}(X)} \quad \text{and} \quad \sigma_Y = \sqrt{\text{var}(Y)}.$$

The random variables X and Y are said to be *uncorrelated* if $\mathrm{corr}(X,Y) = 0$, or equivalently, if $\mathrm{cov}(X,Y) = 0$. Clearly, if X and Y are independent, they are uncorrelated since

$$\begin{aligned}
\mathrm{cov}(X,Y) &= E[(X-\overline{x})(Y-\overline{y})] \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} (x-\overline{x})(y-\overline{y})\pi(x,y)dxdy \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} (x-\overline{x})(y-\overline{y})\pi(x)\pi(y)dxdy \\
&= \left(\int_{\mathbb{R}} (x-\overline{x})\pi(x)dx\right)\left(\int_{\mathbb{R}} (y-\overline{y})\pi(y)dy\right) \\
&= E[(X-\overline{x})]E[(Y-\overline{y})] \\
&= (E[X]-\overline{x})(E[Y]-\overline{y}) \\
&= 0.
\end{aligned}$$

The converse, however, is not true: Two random variables can have vanishing correlation without being mutually independent.

Given a multivariate random variable $X : \Omega \to \mathbb{R}^n$, we define its *covariance matrix* to be

$$\mathrm{cov}(X) = \int_{\mathbb{R}^n} (x-\overline{x})(x-\overline{x})^{\mathsf{T}}\pi(x)dx \in \mathbb{R}^{n\times n},$$

which is defined componentwise as

$$\mathrm{cov}(X)_{ij} = \int_{\mathbb{R}} (x_i-\overline{x}_i)(x_j-\overline{x}_j)\pi(x)dx \in \mathbb{R}, \quad 1 \le i,j \le n.$$

The entries along the main diagonal of the covariance matrix are the variances of the individual components of X, i.e.,

$$\mathrm{cov}(X)_{ii} = \mathrm{var}(X_i), \quad 1 \le i \le n,$$

and the off-diagonal entries give the covariances between the corresponding components of X, i.e.,

$$\mathrm{cov}(X)_{ij} = \mathrm{cov}(X_i, X_j), \quad 1 \leq i, j \leq n, \quad i \neq j.$$

The covariance matrix can be written in terms of expectations as

$$\begin{aligned} \mathrm{cov}(X) &= E[(X - \overline{x})(X - \overline{x})^{\mathsf{T}}] \\ &= E[XX^{\mathsf{T}}] - E[X]E[X]^{\mathsf{T}}. \end{aligned}$$

The covariance matrix is symmetric and positive semi-definite. While the symmetry is implicit in the definition, to prove the positive semi-definiteness we show that, for any $v \in \mathbb{R}^n$, $v \neq 0$,

$$\begin{aligned} v^T \mathrm{cov(X)} v &= v^T \left(\int_{\mathbb{R}^n} (x - \overline{x})(x - \overline{x})^T \pi(x) dx \right) v \\ &= \int_{\mathbb{R}^n} [v^T (x - \overline{x})][(x - \overline{x})^T v] \pi(x) dx \\ &= \int_{\mathbb{R}^n} [v^T (x - \overline{x})]^2 \pi(x) dx \\ &\geq 0 \end{aligned}$$

due to the nonnegativity of $\pi(x)$.

Given two multivariate random variables $X, Y : \Omega \to \mathbb{R}^n$, the *cross covariance matrix* or *variance-covariance matrix* is defined as

$$\begin{aligned} \mathrm{cov}(X, Y) &= E[(X - \overline{x})(Y - \overline{y})^{\mathsf{T}}] \\ &= E[XY^{\mathsf{T}}] - E[X]E[Y]^{\mathsf{T}}. \end{aligned}$$

It follows from the definition that $\mathrm{cov}(X, Y)_{ij} = \mathrm{cov}(X_i, Y_j)$, where X_i is the ith univariate component of X and Y_j is the jth univariate component of Y. Furthermore, $\mathrm{cov}(Y, X) = \mathrm{cov}(X, Y)^{\mathsf{T}}$.

Of particular interest in this thesis are *normally distributed* or *Gaussian* random variables. A univariate random variable $X : \Omega \to \mathbb{R}$ is said to be normally distributed, or Gaussian, denoted as

$$X \sim \mathcal{N}(x_0, \sigma^2),$$

with mean $x_0 \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$, $\sigma^2 \neq 0$, if

$$P(X \leq t) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{t} \exp\left\{ -\frac{1}{2\sigma^2}(x - x_0)^2 \right\} dx,$$

i.e., if its pdf is of the form

$$\pi(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{1}{2\sigma^2}(x - x_0)^2 \right\}.$$

A univariate random variable X is said to have a *standard normal* distribution if it is Gaussian with zero mean and variance one, i.e., $X \sim \mathcal{N}(0, 1)$. The graph of the pdf of a standard normal univariate random variable is shown in Figure 3.1.

A multivariate random variable $X : \Omega \to \mathbb{R}^n$ is said to be normally distributed, or Gaussian, denoted as

$$X \sim \mathcal{N}(x_0, \Gamma),$$

with mean $x_0 \in \mathbb{R}^n$ and symmetric positive definite covariance matrix $\Gamma \in \mathbb{R}^{n \times n}$ if

$$\pi(x) = \left(\frac{1}{(2\pi)^n \det(\Gamma)} \right)^{1/2} \exp\left\{ -\frac{1}{2}(x - x_0)^{\mathsf{T}} \Gamma^{-1} (x - x_0) \right\}.$$

Note that the symmetric positive definiteness of Γ ensures its invertibility. A multivariate random variable X is said to have a standard normal distribution if it is Gaussian with zero mean and covariance matrix the $n \times n$ identity matrix, i.e., $X \sim \mathcal{N}(0, \mathsf{I}_n)$.

Gaussian random variables arise naturally in certain settings, e.g., the modeling of pressure and temperature, where the macroscopic phenomenon is a mean effect of microscopic processes. Mathematically, the central rule of the normal distribution can be attributed to the Central Limit Theorem (CLT), stated as follows for the multivariate case.

Theorem 2. (**Central Limit Theorem**) *If the multivariate random variables* $X_j : \Omega \to \mathbb{R}^n$, $j \geq 1$, *are independent and identically distributed, each with finite mean* μ *and covariance matrix* Γ, *then the distribution of the random variable*

$$Z_k = \frac{1}{\sqrt{k}}\Gamma^{-1/2}\left(\sum_{j=1}^{k} X_j - k\mu\right)$$

converges to the distribution of a standard normal random variable, that is,

$$\lim_{k\to\infty} P(Z_k \in B) = \frac{1}{(2\pi)^{n/2}} \int_B \exp\left\{ -\frac{1}{2}||z||_2^2 \right\} dz.$$

In this case, we say that Z_k converges in distribution to a standard normal random variable. In particular, if Y_k is the mean of k independent and identically distributed (i.i.d.) random variables $X_j : \Omega \to \mathbb{R}^n$ for $j = 1, \ldots, k$,

$$Y_k = \frac{1}{k}\sum_{j=1}^{k} X_j,$$

then for large k,

$$Y_k \overset{\text{approx}}{\sim} \mathcal{N}\Big(\mu, \frac{1}{k}\Gamma\Big).$$

The CLT therefore justifies the use of Gaussian approximations as a model for phenomena involving averaging.

A related result, the Law of Large Numbers (LLN), states that the averages of any ensemble of realizations of a sufficiently large number of i.i.d. random variables converge towards the mean. The strong formulation of the law for the multivariate case is stated as follows.

Theorem 3. (**Strong Law of Large Numbers**) *If the multivariate random variables* $X_j : \Omega \to \mathbb{R}^n$, $j = 1, 2, \ldots, k$, *are i.i.d. and have finite mean* μ, *then*

$$\lim_{k\to\infty} \frac{1}{k}\sum_{j=1}^{k} X_j = \mu$$

almost surely.

The LLN therefore justifies estimating the mean of a sample

$$\mathcal{S} = \{x_1, x_2, \ldots, x_N\}, \quad x_j \in \mathbb{R}^n, \quad j = 1, 2, \ldots, N$$

of N observed independent realizations of a random variable $X : \Omega \to \mathbb{R}^n$ with finite mean μ via the formula

$$\mu = E[X] \approx \frac{1}{N} \sum_{j=1}^{N} x_j = \widehat{\mu}.$$

Further, if X has covariance matrix Γ, we can use the estimate $\mu \approx \widehat{\mu}$ in the estimation of the covariance matrix of the sample $\mathcal{S}$ as follows:

$$\begin{aligned} \Gamma &= \operatorname{cov}(X) \\ &= E[(X-\mu)(X-\mu)^{\mathsf{T}}] \\ &\approx E[(X-\widehat{\mu})(X-\widehat{\mu})^{\mathsf{T}}] \\ &\approx \frac{1}{N} \sum_{j=1}^{N} (x_j - \widehat{\mu})(x_j - \widehat{\mu})^{\mathsf{T}} \\ &= \widehat{\Gamma}. \end{aligned}$$

The estimates

$$\widehat{\mu} = \frac{1}{N} \sum_{j=1}^{N} x_j \quad \text{and} \quad \widehat{\Gamma} = \frac{1}{N} \sum_{j=1}^{N} (x_j - \widehat{\mu})(x_j - \widehat{\mu})^{\mathsf{T}} \tag{3.3}$$

are often referred to as the *sample mean* and the *sample covariance*, respectively, and will be frequently employed in summarizing the statistics of the samples used in the filtering and parameter estimation algorithms derived in later chapters. Note that the sample mean $\widehat{\mu}$ is an *unbiased* estimator of μ since $E[\widehat{\mu}] = \mu$, whereas the sample covariance in (3.3) is a biased estimator of Γ. To eliminate the bias in the estimator, the covariance is sometimes estimated from the sample using the formula

$$\widehat{\Gamma} = \frac{1}{N-1} \sum_{j=1}^{N} (x_j - \widehat{\mu})(x_j - \widehat{\mu})^{\mathsf{T}},$$

where the multiplication by Bessel's correction term $N/(N-1)$ makes $\widehat{\Gamma}$ an unbiased estimator of Γ.

Another distribution useful in sampling is the uniform distribution. A univariate random variable $X : \Omega \to \mathbb{R}$ is said to be *uniformly distributed*,

$$X \sim \text{Uniform}(a, b),$$

on the interval $[a, b]$ where $a, b \in \mathbb{R}$, $a < b$, if its pdf is of the form

$$\pi(x) = \begin{cases} \dfrac{1}{b-a} & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}.$$

Letting $a = 0$ and $b = 1$ results in the *standard uniform distribution*, Uniform(0,1). The graph of the pdf of a standard uniform univariate random variable is shown in Figure 3.1.

Extending to the multivariate case, a multivariate random variable $X : \Omega \to \mathbb{R}^n$ is said to be uniformly distributed,

$$X \sim \text{Uniform}(a, b),$$

over the region $[a_1, b_1] \times \cdots \times [a_n, b_n]$ where $a, b \in \mathbb{R}^n$, $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_n)$, and $a_i < b_i$ for all $i = 1, \ldots, n$, if

$$\pi(x) = \prod_{i=1}^{n} \begin{cases} \dfrac{1}{b_i - a_i} & \text{if } x_i \in [a_i, b_i] \\ 0 & \text{otherwise} \end{cases}.$$

3.2 Bayesian solution of inverse problems

The celebrated *Bayes' theorem*

$$\pi(x \mid y) = \frac{\pi(y \mid x)\pi(x)}{\pi(y)}, \quad \pi(y) \neq 0 \tag{3.4}$$

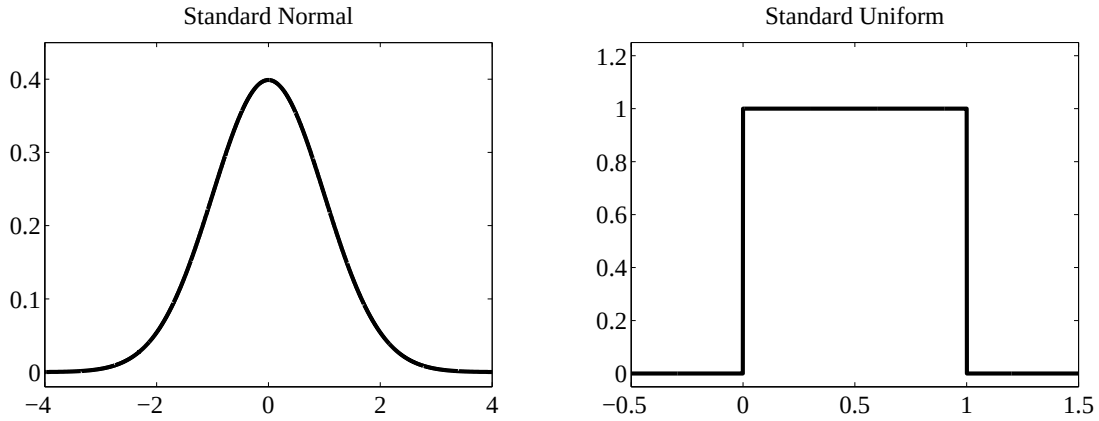

Figure 3.1: Probability density functions of a standard normal (left) and standard uniform (right) univariate random variable.

follows immediately from the definition of conditional probability (3.1) and the symmetry identity (3.2). Named after the Reverend Thomas Bayes, this formula plays an essential role in the Bayesian solution of inverse problems, as it links the three probability densities of interest: the likelihood, the prior, and the posterior. These densities are discussed in more detail below.

Let X be a random variable modeling the unknown quantity of primary interest, and let Y be a random variable modeling the observed data related to X. The *likelihood*, defined as

$$\pi_{\text{likelihood}}(y \mid x) = \pi(y \mid x), \quad y = y_{\text{observed}}, \tag{3.5}$$

is the density of the observations y conditioned on the value of the quantity of interest x. This density can be thought of as a measure indicating how likely it is that the available data are observed if x were known. The *prior*, denoted as

$$\pi_{\text{prior}}(x) = \pi(x), \tag{3.6}$$

is the density encoding any *a priori* information about the distribution of x, without taking the data y into consideration. Finally, the *posterior*, given by

$$\pi_{\text{posterior}}(x \mid y) = \pi(x \mid y), \tag{3.7}$$

is the density of the unknown quantity of interest x conditioned on the observed data y and, as mentioned in Section 1.1.1, the solution of the inverse problem. It follows from an application of Bayes' formula (3.4) that

$$\pi_{\text{posterior}}(x \mid y) = \frac{\pi_{\text{likelihood}}(y \mid x)\pi_{\text{prior}}(x)}{\pi(y)}, \quad \pi(y) \neq 0, \quad y = y_{\text{observed}},$$

or, simplifying the notation,

$$\pi(x \mid y) \propto \pi(y \mid x)\pi_{\text{prior}}(x), \quad y = y_{\text{observed}}, \tag{3.8}$$

where $\propto$ symbolizes proportionality up to the multiplicative constant $1/\pi(y)$. The posterior can be thought of as a correction to the prior after taking the data into account.

In some cases, the posterior density in (3.8) is summarized via simple point estimators and covariance estimates. One of the most popular point estimators is the *maximum a posteriori* (MAP) estimate, x_{MAP}, i.e., the posterior mode, which is defined as

$$x_{\text{MAP}} = \arg\max \pi(x \mid y)$$

if the maximum exists. When computing the MAP estimate, it is sometimes convenient to observe that

$$x_{\text{MAP}} = \arg\min V(x \mid y), \quad V(x \mid y) = -\log \pi(x \mid y)$$

and to proceed with the calculation using numerical optimization techniques. An alternative and also popular point estimator is the *conditional mean* (CM) estimate, i.e., the posterior mean,

$$x_{\text{CM}} = \int_{\mathbb{R}^n} x\pi(x \mid y)dx,$$

whose computation in some special cases can be carried out analytically, but more commonly requires estimation via numerical integration techniques. If the posterior distribution is Gaussian, the CM estimate coincides with the MAP estimate, and they

are equal to the mean of the Gaussian distribution. The CM estimate can be used for the estimation of the posterior covariance by

$$\Gamma_{\text{CM}} = \int_{\mathbb{R}^n} (x - x_{\text{CM}})(x - x_{\text{CM}})^{\mathsf{T}} \pi(x \mid y) dx.$$

Both the MAP and CM point estimates carry some information about the posterior distribution of the unknowns, and each present their own computational challenges. When the information provided by a point estimator is not sufficient, it is possible to explore the posterior density using sampling techniques, such as *Markov chain Monte Carlo* (MCMC), a Monte Carlo sampling method which will be motivated and described in the following sections.

3.3 Monte Carlo integration and MCMC motivation

Before introducing MCMC sampling, we first discuss the idea behind *Monte Carlo integration*, a numerical integration technique based on random sampling. Suppose that we want to estimate the integral

$$\mathcal{I} = \int f(x)\pi(x)dx, \tag{3.9}$$

where $\pi(x)$ is the pdf of a multivariate random variable $X : \Omega \to \mathbb{R}^n$ and $f(x)$ is a function integrable with respect to the probability measure $\pi(x)dx$.

The integral (3.9) can be approximated using a quadrature rule of the form

$$\mathcal{I} \approx \sum_{j=1}^{N} w_j f(x_j), \tag{3.10}$$

where x_j and w_j are the nodes and weights, respectively, of the specified quadrature rule associated with the density π; see [21] for details on particular quadrature rules, such as Newton-Cotes and the Gauss-type formulas.

Quadrature approximations such as (3.10) become problematic when the dimensionality of $x \in \mathbb{R}^n$ is high, since the total number of discretized grid points N grows exponentially with the need for suitable discretization in each coordinate direction; in fact, if we require k nodes per direction, the total number of nodes needed is $N = k^n$, which, for large n, can lead to an unreasonable computational burden. Moreover, if the supports of the density π and the function f are poorly known, this poses an extra challenge for the design of a reasonable quadrature scheme.

An alternate approach, particularly attractive for high dimensional problems, is to use Monte Carlo integration. More specifically, given a sample

$$\mathcal{S} = \big\{x_1, x_2, \ldots, x_N\big\}, \quad x_j \in \mathbb{R}^n, \quad j = 1, 2, \ldots, N \tag{3.11}$$

of realizations x_j generated independently from the density $\pi(x)$, the corresponding *Monte Carlo approximation* to (3.9) is

$$\mathcal{I} \approx \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \widehat{\mathcal{I}}_N, \tag{3.12}$$

and it follows from the LLN that

$$\lim_{N \to \infty} \widehat{\mathcal{I}}_N = \mathcal{I}$$

almost surely. Furthermore, the CLT states that

$$\sqrt{N}(\widehat{\mathcal{I}}_N - \mathcal{I}) \to \mathcal{N}\Big(0, \operatorname{var}(f(X))\Big)$$

in distribution, so

$$\operatorname{var}(\widehat{\mathcal{I}}_N - \mathcal{I}) \approx \frac{\operatorname{var}(f(X))}{N}$$

asymptotically. Therefore, the error in the Monte Carlo approximation (3.12) is $\mathcal{O}(N^{-1/2})$, hence it goes to zero like $1/\sqrt{N}$, regardless of the dimensionality of x.

The challenge in Monte Carlo integration is that direct generation of independent realizations from the target distribution $\pi(x)$ may not be feasible in many applications [22]. The crucial step in Monte Carlo integration therefore becomes how

to effectively draw a sample from $\pi(x)$. Some methods proposed in the literature for generating the sample, including rejection sampling, importance sampling, and sampling-importance-resampling (SIR) methods, start with samples drawn from a *trial distribution* $p(x)$ which is close to $\pi(x)$; for more details on these sampling techniques, see [22, 23].

In the following section, we focus our attention on MCMC sampling, which can be thought of as a Monte Carlo integration procedure in which the random samples are generated in a sequential manner through use of Markov chains [22]. In particular, we focus on the Metropolis-Hastings algorithm, which is the cornerstone of many MCMC techniques.

3.4 The Basics of MCMC

As noted in the previous section, MCMC sampling is a procedure stemming from the use of Markov chains in the generation of a random sample, for example, in the context of Monte Carlo integration. As a prelude to MCMC, we define the concepts of a random walk and a Markov chain. We then describe in detail the Metropolis-Hastings algorithm, an acceptance-rejection-like method upon which many MCMC algorithms are based.

3.4.1 Random walks and Markov chains

Following the notation used in [7], a *random walk* in $\mathbb{R}^n$ is a process of moving around in the state space by taking random steps, which can be performed as follows:

1. Start at a point $x_0 \in \mathbb{R}^n$, and set $k = 0$.
2. Draw a random vector $w_{k+1} \sim \mathcal{N}(0, \mathsf{I}_n)$ and set $x_{k+1} = x_k + \sigma w_{k+1}$.
3. Increment $k = k + 1$ and repeat from 2.

The position $x_k \in \mathbb{R}^n$ of the random walk at time k is a realization of the random variable $X_k : \Omega \to \mathbb{R}^n$. Therefore the random walk process defines a *chain*

$$\{X_0, X_1, \dots\} = \{X_k\}_{k=0}^{\infty}$$

of random variables generated by the evolution model

$$X_{k+1} = X_k + \sigma W_{k+1}, \quad W_{k+1} \sim \mathcal{N}(0, \mathsf{I}_n),$$

where the constant σ determines the length of the step taken from X_k to X_{k+1}. The conditional density of X_{k+1} given $X_k = x_k$ is

$$\pi_k(x_{k+1} \mid x_k) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} ||x_k - x_{k+1}||_2^2 \right) = q_k(x_k, x_{k+1}),$$

and the function $q_k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called the *transition kernel*. Since the length of the step is independent of k, i.e.,

$$q_0 = q_1 = \cdots = q_k = \cdots = q,$$

the transition kernel q is said to be *time invariant*.

The chain $\{X_k\}_{k=0}^{\infty}$ generated in this fashion forms a *discrete-time stochastic process*, i.e., it is an ordered set of random variables, whose index variable k takes on values in a discrete set, which has the *Markov property* in the sense that the probability distribution of each X_{k+1} depends on the past only through the previous element X_k,

$$\pi(x_{k+1} \mid x_0, x_1, \dots, x_k) = \pi(x_{k+1} \mid x_k). \tag{3.13}$$

The Markov property (3.13) is often expressed by saying that "the future depends on the past only through the present." A stochastic process satisfying property (3.13) is referred to as a *Markov process* or a *Markov chain*.

A classic problem in the theory of Markov chains is how to find the corresponding invariant density for a particular transition kernel. Given an arbitrary transition

kernel q and a random variable X with known pdf $\pi(x) = p(x)$, a new random variable Y can be generated via the kernel $q(x, y)$ by setting

$$\pi(y \mid x) = q(x, y).$$

The pdf of Y is then found through marginalization:

$$\pi(y) = \int \pi(y, x)dx = \int \pi(y \mid x)p(x)dx = \int q(x, y)p(x)dx.$$

If the probability density of Y is equal to the probability density of X, i.e., if

$$\int q(x, y)p(x)dx = p(y),$$

then the probability density p is said to be an *invariant density* of the transition kernel q.

A Markov chain with an invariant distribution is *irreducible* if it is possible to eventually move from every state in the state space to every set of positive measure with positive probability. Moreover, a Markov chain is called *periodic* if there are regions of the state space that can be reached only at certain regularly-spaced times; otherwise, it is *aperiodic*. A Markov chain is *ergodic* if it is both irreducible and aperiodic. The ergodicity property is important in guaranteeing that the chain converges to a unique limiting distribution; for further details on the convergence theory of Markov chains, see [24, 25, 26, 5]. In practice, these conditions are typically not checked unless the generated chain fails to converge to the desired distribution after a finite number of steps.

3.4.2 The Metropolis-Hastings algorithm

A classic problem in the theory of Markov chains is to find an invariant density of a given transition kernel. When Markov chains are used to sample from a given target density, however, the problem of interest becomes how to find a transition kernel such that its invariant density is the target density. Therefore, the goal of MCMC

algorithms is to produce a transition kernel q for which the target density $p = p(x)$ is an invariant density. This kernel can then be used to generate a sample that is distributed according to p. One of the most popular methods of constructing such a kernel is the *Metropolis-Hastings* (MH) algorithm [27, 28], which we describe in this section following closely [7]; see also [25, 29]. We remark that an alternative to MH, the *Gibbs sampler*, is also frequently used in MCMC algorithms; we do not present the details of the Gibbs sampler here but instead refer to [30, 22, 31, 26, 5].

Consider the following Markov process: Starting from the current point $x \in \mathbb{R}^n$, either

(a) move from x to a point y using a transition kernel $R(x, y)$, or

(b) stay put at x with probability $r(x)$, $0 \leq r(x) < 1$.

Since $R(x, y)$ is a transition kernel, the mapping $y \mapsto R(x, y)$ defines a probability density, and therefore

$$\int_{\mathbb{R}^n} R(x, y)dy = 1. \tag{3.14}$$

Let $\mathcal{A}$ denote the event of moving from x to y, and let $\neg\mathcal{A}$ denote the event of staying at x. The probabilities of these events are thus given by

$$P(\mathcal{A} \mid X = x) = 1 - r(x) \quad \text{and} \quad P(\neg\mathcal{A} \mid X = x) = r(x).$$

Given the current state $X = x$, we want to find the probability density of Y generated by the above process. It follows that the probability of the event $Y \in B$ for some $B \subset \mathbb{R}^n$ is

$$\begin{aligned} P(Y \in B \mid X = x) \;&=\; P(Y \in B \mid X = x, \mathcal{A})P(\mathcal{A} \mid X = x) \\ &\quad + P(Y \in B \mid X = x, \neg\mathcal{A})P(\neg\mathcal{A} \mid X = x) \end{aligned}$$

so Y can end up in B either by moving to B or by staying in B. The probability of arriving in B after a move is calculated using the transition kernel,

$$P(Y \in B \mid X = x, \mathcal{A}) = \int_B R(x, y)dy,$$

and the probability of arriving in B without moving, which relies on x already being in B, is equal to the probability that $x \in B$,

$$P(Y \in B \mid X = x, \neg\mathcal{A}) = \chi_B(x) = \begin{cases} 1 & \text{if } x \in B \\ 0 & \text{if } x \notin B \end{cases}$$

where $\chi_B(x)$ is the *characteristic function* of B. In summary, it follows that

$$\begin{aligned} P(Y \in B \mid X = x) &= P(Y \in B \mid X = x, \mathcal{A})P(\mathcal{A} \mid X = x) \\ &\quad + P(Y \in B \mid X = x, \neg\mathcal{A})P(\neg\mathcal{A} \mid X = x) \\ &= \left(\int_B R(x, y)dy \right)(1 - r(x)) + \chi_B(x)r(x) \\ &= \int_B (1 - r(x))R(x, y)dy + \chi_B(x)r(x). \end{aligned}$$

Marginalization over x gives the probability of $Y \in B$ regardless of the initial position,

$$\begin{aligned} P(Y \in B) &= \int P(Y \in B \mid X = x)p(x)dx \\ &= \int \left(\int_B (1 - r(x))R(x, y)dy + \chi_B(x)r(x) \right) p(x)dx \\ &= \int \left(\int_B (1 - r(x))R(x, y)dy \right) p(x)dx + \int \chi_B(x)r(x)p(x)dx \\ &= \int_B \left(\int (1 - r(x))R(x, y)p(x)dx \right) dy + \int_B r(x)p(x)dx \\ &= \int_B \left(\int (1 - r(x))R(x, y)p(x)dx + r(y)p(y) \right) dy, \end{aligned}$$

and since by definition of the probability density $\pi(y)$ of Y,

$$P(Y \in B) = \int_B \pi(y)dy,$$

it follows that

$$\pi(y) = \int (1 - r(x))R(x, y)p(x)dx + r(y)p(y).$$

Therefore we aim to find a kernel R such that $\pi(y) = p(y)$, i.e.,

$$p(y) = \int (1 - r(x))R(x, y)p(x)dx + r(y)p(y),$$

or, equivalently,

$$(1 - r(y))p(y) = \int (1 - r(x))R(x, y)p(x)dx. \tag{3.15}$$

Letting

$$K(x, y) = (1 - r(x))R(x, y),$$

it follows from (3.14) that

$$\int K(y, x)dx = (1 - r(y)) \int R(y, x)dx = 1 - r(y).$$

We can thus rewrite equation (3.15) as

$$\int p(y)K(y, x)dx = \int p(x)K(x, y)dx, \tag{3.16}$$

which is referred to as the *balance equation*. To guarantee that the balance equation (3.16) is satisfied, it suffices to require that

$$p(y)K(y, x) = p(x)K(x, y), \tag{3.17}$$

a condition known as the *detailed balance equation*. The detailed balance equation (3.17) says that the probability of moving from x to y is the same as the probability of moving from y to x. Markov chains that satisfy the detailed balance equation are *reversible* [22], meaning that for any given realization in the chain, it is impossible to decide if the chain reached it by moving forward or backward in time. The MH algorithm is a method of finding a kernel K such that (3.17) is satisfied.

In order to start the MH algorithm, it is necessary to select a *proposal distribution* or *candidate generating kernel* $q(x, y)$, also referred to as the MH proposal or MH

kernel, which is usually chosen so that the generation of a Markov chain from q is a straightforward (and simple) task. For this reason, Gaussian kernels are often selected.

If the candidate kernel q satisfies the detailed balance equation (3.17), i.e., if

$$p(y)q(y,x) = p(x)q(x,y), \tag{3.18}$$

then we let $r(x) = 0$ and $K(x,y) = R(x,y) = q(x,y)$ and we are done, as p is an invariant density for such a kernel, because

$$\int p(x)q(x,y)dx = \int p(y)q(y,x)dx = p(y)\int q(y,x)dx = p(y).$$

The detailed balance equation therefore ensures invariance, though the converse is not necessarily true [22].

In general, a chosen q will not satisfy (3.18), implying that one side of the equation is larger than the other. Suppose that

$$p(y)q(y,x) < p(x)q(x,y),$$

and define the kernel K as

$$K(x,y) = \alpha(x,y)q(x,y),$$

where the correcting factor $\alpha(x,y)$ is chosen so that the detailed balance equation (3.17) holds for K, i.e.,

$$p(y)\alpha(y,x)q(y,x) = p(x)\alpha(x,y)q(x,y). \tag{3.19}$$

Since the kernel α need not be symmetric, we can let

$$\alpha(y,x) = 1$$

and compute $\alpha(x,y)$ according to (3.19), which yields

$$\alpha(x,y) = \frac{p(y)q(y,x)}{p(x)q(x,y)} < 1.$$

If, instead,

$$p(y)q(y,x) > p(x)q(x,y),$$

we can reverse the roles of x and y above, letting $\alpha(x,y) = 1$ and solving for $\alpha(y,x)$.

Therefore it follows that by setting

$$K(x,y) = \alpha(x,y)q(x,y), \quad \alpha(x,y) = \min\left\{1, \frac{p(y)q(y,x)}{p(x)q(x,y)}\right\},$$

the kernel K satisfies the detailed balance equation (3.17). Random draws from K can be generated using a two-phase procedure along the lines of rejection sampling. An outline of this procedure for generating a sample chain of length N is given in Algorithm 1.

Algorithm 1 Metropolis-Hastings Algorithm

Given an initial point $x_0 \in \mathbb{R}^n$ and candidate kernel q, set $k = 0$ and choose the length N of the sample chain to be generated.

1. Draw a proposal point $y \in \mathbb{R}^n$ using the transition kernel $q(x_k, y)$.
2. Calculate the *acceptance ratio*

$$\alpha(x_k, y) = \frac{p(y)q(y,x_k)}{p(x_k)q(x_k,y)}.$$

3. "Flip the α-coin": Draw $t \sim \text{Uniform}(0,1)$.
   (a) If $\alpha > t$, accept the move to y and set $x_{k+1} = y$.
   (b) Otherwise, reject the move to y, remain at the current point and set $x_{k+1} = x_k$.
4. If $k < N$, set $k = k + 1$ and repeat from Step 1; otherwise, stop.

To see that the procedure in Algorithm 1 indeed effectuates the move according

to the kernel K, observe that

$$\begin{aligned}\pi(y, \mathcal{A} \mid x) &= \pi(y \mid \mathcal{A}, x)\pi(\mathcal{A} \mid x) \\ &= q(x, y)\alpha(x, y) \\ &= K(x, y),\end{aligned}$$

that is, the distribution of accepted moves is indeed given by $y \mapsto K(x, y)$.

There are several important issues to consider when implementing the MH algorithm. As previously noted, when not much is known about the shape of the target distribution $p(x)$, the MH candidate kernel $q(x, y)$ is typically chosen to be Gaussian, e.g., a Gaussian white noise random walk distribution of the form

$$q(x, y) \propto \exp\left\{ -\frac{1}{2\gamma^2}||x - y||_2^2 \right\},$$

that is, $X - Y \sim \mathcal{N}(0, \gamma^2 \mathsf{I}_n)$, where the parameter γ controls the length of the proposed steps. In this case, the transition kernel is symmetric, i.e., $q(x, y) = q(y, x)$, so the MH acceptance ratio simplifies to

$$\alpha(x, y) = \frac{p(y)}{p(x)},$$

as in the original Metropolis algorithm [27]; the generalization to non-symmetric transition kernels was later derived by Hastings [28].

The choice of a starting point $x_0 \in \mathbb{R}^n$ and the step length often are crucial to the success of the MH algorithm in adequately exploring the target density. Taking too small of steps results in a high acceptance rate, which often indicates that the chain is moving too conservatively and does not sufficiently explore the density, while taking excessively long steps leads to a low acceptance rate, hence the chain does not move much and the majority of proposed steps are rejected. It has been suggested in the literature that the step length should be tuned so that the acceptance rate falls between 20% and 30% [7] or 25% and 35% [32, 22]. Further, if the starting point

is chosen far from the numerical support of the target density, it may take a large number of steps to simply reach the support. These *burn-in* steps tell nothing about the target density and must be discarded from the chain in order to not affect the sample calculations. It is hard to judge *a priori* how many burn-in steps should be discarded; the length of the burn-in may affect the length of the chain needed to adequately sample the target distribution, with a longer chain needed to compensate for a long burn-in period.

Another key factor in deciding the length N of the Markov chain, or, equivalently, when to stop the sampling algorithm, is how well the chain represents its target distribution. This decision can be vital, e.g., in applications where the calculations for each additional point are expensive or time-consuming. Since there are no set rules for translating theoretical MCMC convergence results into practical stopping criteria, it is typical in practice to monitor the behavior of the already collected sample. To this end, it is important to remember that while Monte Carlo convergence results [33, 25] assume independent samples, this is not the case with MCMC points. In fact, each sample point generated by the MH algorithm is correlated with the previous point.

However, after a number of steps, this correlation becomes negligible, so it is a natural to ask how long it takes for an MCMC scheme to generate a sample with enough independent points to accurately represent the distribution. This question can be addressed by evaluating the *autocovariance function* (ACF) [34, 35, 36] of the sample and using it to compute the *integrated autocorrelation time* (IACT), which we denote by τ. The IACT can be interpreted as the time required for a given MCMC method to produce an independent sample. It has been shown that while the Monte Carlo approximation error for independence samplers goes to zero like $1/\sqrt{N}$, as discussed in Section 3.3, the MCMC sampler converges to its limiting distribution at a rate $1/\sqrt{N/\tau}$; see [34, 35] for further details.

Variations of the MH algorithm have been developed to better account for the shape of the invariant density, which becomes an issue particularly for problems that are strongly nonlinear and/or of high-dimensionality. Methods such as adaptive

MH automatically adjust the MH kernel while the algorithm proceeds, taking into account the size and shape of the underlying target density. Select variations of the MH algorithm will be discussed in the following chapter.

Chapter 4

A Review of Filtering and Parameter Estimation Algorithms

In this chapter we review the relevant literature regarding filtering and parameter estimation algorithms in a Bayesian statistical framework. We establish the terminology used in such problems in Section 4.1 and give an overview of both non-sequential and sequential Bayesian methods for solving parameter estimation problems in Section 4.2. The general evolution-observation model and updating formulas for sequential Bayesian filtering of discrete-time stochastic processes is described in Section 4.3. The classical, extended, and ensemble Kalman filtering algorithms for state estimation are the topic of Section 4.4, along with an overview of related solution methods. The ensemble Kalman filter, a variation on the Kalman filter with a Monte Carlo twist, is particularly relevant to the algorithm developed in Chapter 8. Sequential Monte Carlo (SMC) particle filtering algorithms are discussed in Section 4.5, with particular emphasis on an algorithm for state and parameter estimation proposed by Liu and West [37] which serves as a basis for the algorithm developed in Chapter 5.

4.1 Filtering, smoothing, and parameter estimation

We begin by establishing some standard notation and terminology that will be used throughout the remainder of this thesis. The continuous-time evolution model (1.1) is discretized by developing a marching scheme as mentioned in Section 2.1.2 using discrete times t_j. Without loss of generality, we assume that the time discretization is uniform. We further assume that the observation model follows (1.2). To avoid using double indexing for the propagation steps and the observation times, we adhere to the indexing of the propagation steps with the convention that

$$b_j = \emptyset \text{ if no data arrives at } t = t_j.$$

Following this notation, $b_j = \emptyset$ implies that $D_j = D_{j-1}$, where D_j is defined as in (1.4).

It is customary in the state and parameter estimation literature to address three separate types of problems with varying complexity: *filtering*, *smoothing*, and *parameter estimation*, the last being the most challenging and most relevant in the solution of inverse problems. We now briefly review each of the aforementioned problem types.

Filtering problems in continuous time usually assume an evolution model based on stochastic differential equations (SDEs). In this thesis, however, the emphasis is on deterministic evolution models. Therefore, in the framework on non-stochastic models, filtering problems assume that the system of differential equations describing the evolution of the states is known exactly, but the initial state may be poorly known, and the state is incompletely observed. The former condition can be expressed by replacing the initial condition u_0 by $\pi_0(u_0)$, where π_0 is a given probability density defining the distribution of the initial states. The observations, corrupted with additive noise, are assumed to depend only on some components of the solution. More specifically, if the vector-valued function u is partitioned as $u = (u^{(1)}, u^{(2)}) \in \mathbb{R}^p \times \mathbb{R}^{d-p}$, $p < d$, and we assume that the data depend only on components in $u^{(1)}$, that is,

$$b_j = g(u^{(1)}(t_j)) + w_j \in \mathbb{R}^m, \qquad 0 < t_1 < t_2 < \ldots < t_T$$

for some function $g : \mathbb{R}^p \to \mathbb{R}^m$, the filtering problem, or state estimation problem, is to estimate the posterior density of u at $t = t_j$. Of particular interest is the prediction of the components not directly observed, which we refer to as the *blind* states of the system. As will be explained in Section 4.4.1, the classical Kalman filter is a special instance of such filtering algorithms [38].

In the present context, the smoothing problem seeks to find the initial state of the system, u_0, from which it is possible to generate any successive state $u(t)$ simply by propagation. This problem is sometimes referred to as history matching. Therefore, starting from an *a priori* distribution $\pi_0(u_0)$ of u_0, we aim to find the posterior distribution $\pi(u_0 \mid D_T)$, which is the solution of the smoothing problem. We do not further elaborate on the smoothing problem here, because it can be formulated as a special case of the parameter estimation problem, as will be described momentarily.

In the sequential estimation of the unknown parameters of the evolution model, we collect all unknown parameters, which may include the initial state u_0, in the vector $\theta \in \mathbb{R}^k$. Since in our treatment of both smoothing and sequential parameter estimation problems the state $u(t_j)$ is always estimated, the filtering problem can be treated as a special case of smoothing and sequential parameter estimation problems.

The filtering and sequential parameter estimation problems belong to the category of *nonstationary* inverse problems, where the quantities of interest are time-dependent and the data depend on measurements of these quantities at different times. In the following sections, we will address the relevant methods for solving these problems in a Bayesian framework.

4.2 An overview of Bayesian methods for state and parameter estimation

In this section we give an overview of both sequential and non-sequential methods for solving the state and parameter estimation problem in a Bayesian statistical framework. The non-sequential methods include several variations of the MH algorithm

described in Section 3.4.2, while the sequential methods include the Kalman-type filters and particle filtering algorithms, which are of particular interest in this thesis and will be the topic of the sections that follow.

4.2.1 Non-sequential methods

Non-sequential Bayesian methods for solving filtering and parameter estimation problems use all available data in one batch and resort to the use of MCMC methods, most often employing a variation of the MH algorithm, to explore the posterior density

$$\pi(\theta \mid D_T), \qquad D_T = \big\{b_1, b_2, \ldots, b_T\big\},$$

which requires the forward map $\theta \mapsto \big\{b_1, b_2, \ldots, b_T\big\}$, or, in practice, a numerical integrator of the system (1.1). The success of the MCMC method depends crucially on how effective the proposal distribution, or the MH kernel, is at producing good mixing and independent samples. The choice of an appropriate candidate kernel is especially important in high-dimensional problems where the large number of parameters calls for rather sophisticated modification of the MH proposals.

Among the proposed variants, we mention the adaptive MH algorithms, where the MH kernel is adjusted as the algorithm proceeds in order to better account for the size and shape of the target distribution p. Adaptive MH algorithms were introduced in [39, 40, 41, 42, 43]; for an overview of adaptive MCMC techniques, see [44] and references therein.

When the MH algorithm is used to generate a sample from a target distribution about which not much is known *a priori*, the initial "tuning" of the MH proposal can take a long time. In the adaptive proposal (AP) algorithm [39], the MH proposal is updated periodically as the chain progresses using the sample covariance calculated from a fixed number of previous points in the chain, thereby locally adapting the MCMC process to the target distribution. More specifically, assuming that the target density $p(x)$ is d-dimensional and that at time k a sample

$$\big\{x_0, x_1, \ldots, x_{k-M+1}, \ldots, x_{k-1}, x_k\big\}, \quad x_j \in \mathbb{R}^d, \quad j = 0, 1, \ldots, k$$

of at least M points has accumulated, where M is referred to as the memory parameter, the proposal distribution q_k for drawing the next proposed step y is chosen by

$$q_k(y \mid x_0, x_1, \ldots, x_k) \sim \mathcal{N}(x_k, c_d^2 R_k) \tag{4.1}$$

where R_k is the $d \times d$ sample covariance matrix determined by the M points

$$x_{k-M+1}, \ldots, x_k$$

and c_d is a scaling factor depending only on the dimension d. In the AP algorithm, the covariance is not updated with every step but only after a prescribed finite number of steps η, where η is called the update frequency, and kept fixed in between.

Practical considerations when using the AP algorithm thus include selecting the values of the parameters M, η, and c_d, with the choice of the covariance scaling factor c_d being of particular significance. In [45] it is shown that, in the case of Gaussian target distributions, setting $c_d = 2.4/\sqrt{d}$ improves results over using no scaling, i.e., $c_d = 1$. The total length N of the chain must also be assigned, and it must increase with the dimensionality d. The optimal values of the memory parameter M and updating frequency η also increase with d; see [39] for a more detailed analysis of these parameter choices. Though the AP algorithm is not strictly ergodic, it can be used as an effective adaptive burn-in procedure for ergodic MCMC methods.

The adaptive Metropolis (AMet) algorithm [40] is an extension of the AP algorithm in which the proposal is continuously adapted to the target distribution, using the full information accumulated up to that point. In other words, while the AP algorithm updates the covariance of the Gaussian proposal distribution (4.1) locally using only a fixed number M of previous states, the AMet algorithm does so globally using the entire chain. The covariance update is done via a simple recursion formula which takes advantage of the definition of the sample covariance matrix.

One advantage of the AMet algorithm is that it starts adapting right away, ensuring that the search is more efficient in the early stages of simulation. Yet, as with the AP algorithm, better mixing properties result when the adaptation is performed

after some prescribed number of steps η. Due to the fact that the updated proposal depends on the full history of the chain, the AMet algorithm is neither Markovian nor reversible. However, while the limiting distribution of the AP algorithm differs slightly from the target, the AMet algorithm is shown to preserve asymptotically the correct ergodic properties. The proof of the ergodicity of the AMet algorithm, presented in [40], is based on the assumptions that the target density has compact support and is bounded from above. Moreover, the Gaussian proposal could be replaced by, e.g., uniform distributions on a parallelepiped, retaining the same ergodicity results.

A variant of the AMet algorithm where the adaptation is performed componentwise, called the single component adaptive Metropolis (SCAM) algorithm [41], has been shown to work well for problems of dimensions up to 1,000 for which updating the sample covariance with standard AMet can be time-consuming. The SCAM algorithm can be thought of as applying the AMet algorithm on each component separately. For example, the ith coordinate $x_k^{(i)}$ at time k is determined as follows:

1. Draw $y^{(i)}$ from a one-dimensional Gaussian proposal $q_k^{(i)} \sim \mathcal{N}(x_{k-1}^{(i)}, v_k^{(i)})$;

2. Accept the candidate point $y^{(i)}$ with probability

$$\min\left\{1, \frac{p(x_k^{(1)}, \ldots, x_k^{(i-1)}, y^{(i)}, x_{k-1}^{(i+1)}, \ldots, x_{k-1}^{(d)})}{p(x_k^{(1)}, \ldots, x_k^{(i-1)}, x_{k-1}^{(i)}, \ldots, x_{k-1}^{(d)})}\right\}.$$

The above steps are similar to those performed in the single component Metropolis algorithm, with the difference that the variances $v_k^{(i)}$ are time-dependent.

As was the case for the AMet algorithm, while not Markovian, the SCAM algorithm has been shown to have the correct ergodic properties. If the components are highly correlated, to avoid poor mixing the proposal distribution may be rotated so that the sampling occurs in the order of the principal vectors of the covariance matrix; for more details, see [46].

Another adaptive method aiming to reduce the number of outright rejected proposals by allowing for more than one candidate step per iteration is the delayed

rejection (DR) algorithm [42]. The DR algorithm was developed to overcome the problem that when a Markov chain remains at the same position for too long, which occurs when the proposed steps are rejected time and again, the estimates obtained by averaging over the chain are less efficient [47]. The DR algorithm remedies this situation by inserting a "delaying process" into the MH framework, which allows for a successive chain of candidate steps, referred to by stages, to be considered within the same iteration. Therefore, when a candidate step is rejected, instead of automatically staying at the current position and advancing time, a new candidate step can be proposed with an adjusted proposal and acceptance probability based on the current step and the first rejected step. More specifically, assuming that the value of the chain at the current time k is x_k, a first-stage candidate step y_1 is generated from the kernel $q_1(x_k, y_1)$ and accepted with probability

$$\begin{aligned} \alpha_1(x_k, y_1) &= \min\left\{1, \frac{p(y_1)q_1(y_1, x_k)}{p(x_k)q_1(x_k, y_1)}\right\} \\ &= \min\left\{1, \frac{T_1}{B_1}\right\} \end{aligned}$$

as in the standard MH algorithm. If y_1 is rejected, suggesting that the proposal distribution is a bad fit locally, an adapted kernel $q_2(x_k, y_1, y_2)$ for a new proposal y_2 can be constructed, and the acceptance probability of the second-stage candidate y_2 can be computed to retain the stationary distribution. This can be achieved by imposing the detailed balance equation (3.17) separately at each stage and computing the acceptance probability that maintains reversibility. Thus the second-stage candidate is accepted with probability

$$\begin{aligned} \alpha_2(x_k, y_1, y_2) &= \min\left\{1, \frac{p(y_2)q_1(y_2, y_1)q_2(y_2, y_1, x_k)[1 - \alpha_1(y_2, y_1)]}{p(x_k)q_1(x_k, y_1)q_2(x_k, y_1, y_2)[1 - \alpha_1(x_k, y_1)]}\right\} \\ &= \min\left\{1, \frac{T_2}{B_2}\right\}, \end{aligned}$$

and since y_1 was rejected, it follows that $T_1 < B_1$ and therefore the $\alpha_1(x_k, y_1)$ term

in the expression for B_2 can be replaced by T_1/B_1, yielding

$$\begin{aligned}\alpha_2(x_k, y_1, y_2) &= \min\left\{1, \frac{T_2}{q_2(x_k, y_1, y_2)[p(x_k)q_1(x_k, y_1) - p(y_1)q_1(y_1, x_k)]}\right\} \\ &= \min\left\{1, \frac{T_2}{q_2(x_k, y_1, y_2)[B_1 - T_1]}\right\}.\end{aligned}$$

The delaying process can continue in this manner for a fixed or random number of stages at each time step.

The DR algorithm therefore locally adapts the proposal distribution while remaining both Markovian and reversible. It is worthwhile to note that the computation time for one iteration of DR may be considerably longer than one iteration of standard MH due to the multiple candidates at each iteration.

A non-Markovian and ergodic method combining the DR and AMet algorithms, called delayed rejection adaptive Metropolis (DRAM), was proposed in [43] to merge the two algorithms at the benefit of both. In fact, AMet enhances the success of DR in situations when good proposals are not available, while DR speeds up the exploration of the target density when AMet has a slow start. There are a number of ways to combine the DR and AMet algorithms. One simple strategy nests the AMet algorithm within an r-stage DR algorithm in the following manner:

1. Adapt the first-stage proposal of DR as in AMet, i.e., compute the first-stage covariance R^1 using all previous sample points in the chain via the AMet recursion formula.

2. Compute the covariance R^i of the ith stage, $i = 2, \ldots, r$, as a scaled version of the first-stage covariance, i.e., $R^i = \gamma_i R^1$ for some scaling factor γ_i.

The scaling factor γ_i can be chosen such that higher stages have smaller or larger variances than the lower stages. It has been shown beneficial to decrease the variance as the stages increases, thereby starting with a variance that is "too big" and subsequently scaling it down.

While adaptive methods can substantially reduce the time spent on tedious pilot runs to adjust the step size, it is easy to construct examples in which the adaptation covariance goes astray. A different class of MCMC samplers, referred to as Metropolis adjusted Langevin algorithms (MALA) [48], generates the sample by solving a Langevin diffusion SDE with a drift term based on the gradient of the logarithm of the target density. To guarantee convergence to the invariant density of the time discretized version, a Hastings-type acceptance check must be included. Another class of algorithms, known as Hamiltonian Monte Carlo (HMC) [49], generates the sample by solving a Hamiltonian system with the target density as a potential function. In order to compensate for the error introduced by the use of numerical schemes to solve the associated differential equations, an acceptance step is again introduced to guarantee the correct limit distribution. For recent contributions to these methods, see [50].

In the literature, all of the aforementioned algorithms are applied to the type of problem that we are interested in, where the forward model requires the solution of a large and stiff system of ODEs for which, ideally, a high-order numerical integrator would be used to reduce the errors due to numerical approximation. Unfortunately, since the stiffness of the system (1.1) depends on the parameter vector θ, random proposals in high dimensions, even with sophisticated adaptation schemes, easily result in systems so stiff to create serious problems when it comes to their numerical solution, causing standard numerical solvers to struggle or even fail to converge. To exacerbate the situation, it may happen that proposals leading to systems of differential equations requiring a long integration are rejected. This problem can be partially mitigated through use of surrogate move, or delayed acceptance, MCMC algorithms, as in, e.g., [22, 51, 2], where a surrogate transition kernel $S(x, y)$ is chosen such that a reasonable approximation $p^*(x)$ of the target density $p(x)$ remains invariant, satisfying the detailed balance equation

$$p^*(x)S(x, y) = p^*(y)S(y, x).$$

A candidate step y is then drawn from S and accepted with probability

$$\min\left\{1, \frac{p(y)p^*(x)}{p^*(y)p(x)}\right\}$$

where x is the current position of the chain. Computational efficiency for surrogate move algorithms may remain an issue nonetheless.

The numerical solution of the differential equation has been identified by many authors as a true bottleneck, thus motivating the development of parameter estimation approaches that completely avoid the numerical solution of the system [52, 53, 54]. In the generalized smoothing approach [52], which has its roots in [53], the solution is approximated in an appropriate basis, e.g., a spline basis, and the coefficients of the expansion are estimated along with the system parameters θ. A Bayesian version of the algorithm is described in [54].

4.2.2 Sequential methods

Instead of using all the collected data at once, sequential methods update the posterior density from one time instance to the next. The advantage of this approach is that the numerical integrators do not have to propagate the solution through the full time interval $[0, t_T]$ but just from one observation time instant to the next. As mentioned in Section 1.2, sequential Bayesian methods for estimating only the state variable u include the classical Kalman filter when the model is linear and Gaussian, and the extended Kalman filter, ensemble Kalman filter, and particle filters when the model is nonlinear and/or non-Gaussian. For an overview of such sequential methods, we refer to [5, 23, 55, 56, 57].

Kalman-type filtering algorithms and particle filtering algorithms play a major role in the scope of this thesis, therefore we describe them in detail in the sections that follow. While the classical Kalman filter allows an explicit solution of the conditional densities, the particle filtering and ensemble Kalman filtering techniques of interest in this work make use of random samples and ensemble statistics. Hence, we say that

particle filters and ensemble Kalman filters fall into the category of SMC methods, being sequential algorithms which make use of Monte Carlo integration strategies.

A main goal of this thesis is the estimation of the model parameter vector θ along with the state vector, for which several SMC techniques have been suggested. The most relevant reference with respect to particle filter SMC is the algorithm of Liu and West [37], which integrates the auxiliary particle technique of Pitt and Shephard [58] with the approximation of the posterior density of the parameter vector by a Gaussian mixture, or an ensemble of particles drawn from the density [59, 60]. A similar approach is suggested in [61]. These algorithms will be described in detail in Section 4.5.

4.3 The evolution-observation model and updating formulas

The Kalman-type filters and particle filters mentioned in Section 4.2.2 can be viewed as Bayesian filtering methods for nonstationary inverse problems. In this section we define the Bayesian filtering problem for discrete-time stochastic processes, as described in [5], and develop the general evolution-observation model. We focus our attention to discrete-time evolution models, though a continuous-time formulation using the Itô interpretation of systems of SDEs is given, e.g., in [16]. We then derive the two formulas needed to sequentially update the posterior density.

Consider the stochastic processes $\{X_k\}_{k=0}^{\infty}$ and $\{Y_k\}_{k=1}^{\infty}$, where the random variables $X_k \in \mathbb{R}^d$ represent the unknowns of primary interest, or states, and $Y_k \in \mathbb{R}^m$ the measurements, or observations, at time k. Suppose that the processes satisfy the following three assumptions:

1. The process $\{X_k\}_{k=0}^{\infty}$ is a Markov process, i.e.,

$$\pi(x_{k+1} \mid x_0, x_1, \ldots, x_k) = \pi(x_{k+1} \mid x_k). \tag{4.2}$$

2. The process $\{Y_k\}_{k=1}^{\infty}$ is a Markov process with respect to the history of X_k

values, i.e.,

$$\pi(y_k \mid x_0, x_1, \ldots, x_k) = \pi(y_k \mid x_k). \tag{4.3}$$

3. The process $\{X_k\}_{k=0}^{\infty}$ depends on past observations only through its own history, i.e.,

$$\pi(x_{k+1} \mid x_k, y_1, y_2, \ldots, y_k) = \pi(x_{k+1} \mid x_k). \tag{4.4}$$

A pair of stochastic processes $\{X_k\}_{k=0}^{\infty}$ and $\{Y_k\}_{k=1}^{\infty}$ satisfying the properties (4.2), (4.3) and (4.4) form an *evolution-observation model*.

In order for an evolution-observation model to be fully specified, we need to know

(a) The pdf of the initial state X_0,

(b) The Markov transition kernels $\pi(x_{k+1} \mid x_k)$, $k = 0, 1, 2, \ldots$,

(c) The likelihood functions $\pi(y_k \mid x_k)$, $k = 1, 2, \ldots$.

Note that the transition kernels $\pi(x_{k+1} \mid x_k)$ and the likelihood functions $\pi(y_k \mid x_k)$ may change with time. That is,

$$\pi(x_{k+1} \mid x_k) = \pi_k(x_{k+1} \mid x_k), \quad \pi(y_k \mid x_k) = \pi_k(y_k \mid x_k),$$

which is implicity assumed in the notation used here.

The general form of an evolution-observation model is given by

$$X_{k+1} = F_{k+1}(X_k, W_{k+1}), \quad k = 0, 1, 2, \ldots \tag{4.5}$$

$$Y_k = G_k(X_k, V_k), \quad k = 1, 2, \ldots \tag{4.6}$$

where the *state evolution equation* (4.5) is a Markov model describing the evolution of the states, and the *observation equation* (4.6) is a model for the observations depending on the current state. The functions F_{k+1} and G_k are assumed to be known, and the random variables W_{k+1} and V_k are called the *state noise* and *observation noise*, respectively. We assume that for $k \neq \ell$ the state noise vectors W_k and W_ℓ are mutually independent, which we denote by $W_k \perp W_\ell$, and that the observation

noise vectors V_k and V_ℓ are mutually independent, $V_k \perp V_\ell$, and independent of the initial state X_0. Moreover, we assume that the noise vectors W_k and V_ℓ are mutually independent, $W_k \perp V_\ell$, for all k, ℓ.

Denoting by D_k the accumulated observations up to time k,

$$D_k = \{y_1, y_2, \ldots, y_k\},$$

we aim to sequentially update the posterior distribution $\pi(x_k \mid D_k)$ using the following scheme:

$$\pi(x_k \mid D_k) \longrightarrow \pi(x_{k+1} \mid D_k) \longrightarrow \pi(x_{k+1} \mid D_{k+1}).$$

The first step, called the prediction step or time evolution update, relies on the state evolution equation (4.5) and can be formulated as follows: Given $\pi(x_k \mid D_k)$, find $\pi(x_{k+1} \mid D_k)$ using the Markov transition kernel $\pi(x_{k+1} \mid x_k)$. The second step, called the observation update or the analysis step in filtering literature, is based on the observation equation (4.6): Given $\pi(x_{k+1} \mid D_k)$, find $\pi(x_{k+1} \mid D_{k+1})$ using the new observation y_{k+1} and the likelihood function $\pi(y_{k+1} \mid x_{k+1})$.

The updating formula for the prediction step can be interpreted as a marginalization via the Chapman-Kolmogorov formula. First note that from the definition of conditional probability densities we have

$$\begin{aligned}
\pi(x_{k+1}, x_k, D_k) &= \pi(x_{k+1} \mid x_k, D_k)\pi(x_k, D_k) \\
&= \pi(x_{k+1} \mid x_k, D_k)\pi(x_k \mid D_k)\pi(D_k) \\
&= \pi(x_{k+1} \mid x_k)\pi(x_k \mid D_k)\pi(D_k),
\end{aligned}$$

where the last equality follows from the Markov property (4.4). By marginalizing with respect to x_k,

$$\int \pi(x_{k+1}, x_k, D_k) dx_k = \pi(x_{k+1}, D_k),$$

we obtain

$$\begin{aligned}\pi(x_{k+1} \mid D_k) &= \frac{\pi(x_{k+1}, D_k)}{\pi(D_k)} \\ &= \frac{\int \pi(x_{k+1}, x_k, D_k) dx_k}{\pi(D_k)},\end{aligned}$$

and therefore the time evolution update is given by the Chapman-Kolmogorov formula,

$$\pi(x_{k+1} \mid D_k) = \int \pi(x_{k+1} \mid x_k)\pi(x_k \mid D_k) dx_k. \tag{4.7}$$

The formula for the observation update is an application of Bayes' theorem (3.4). Consider the joint density

$$\begin{aligned}\pi(x_{k+1}, D_{k+1}) &= \pi(y_{k+1}, x_{k+1}, D_k) \\ &= \pi(y_{k+1} \mid x_{k+1}, D_k)\pi(x_{k+1}, D_k) \\ &= \pi(y_{k+1} \mid x_{k+1}, D_k)\pi(x_{k+1} \mid D_k)\pi(D_k) \\ &= \pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)\pi(D_k),\end{aligned}$$

where the last equality follows from observation equation (4.6) and the mutual independence of V_k, implying that y_{k+1} is conditionally independent of the previous observations given that x_{k+1} is known. It follows from the definition of conditional probability that

$$\begin{aligned}\pi(x_{k+1} \mid D_{k+1}) &= \frac{\pi(x_{k+1}, D_{k+1})}{\pi(D_{k+1})} \\ &= \frac{\pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)\pi(D_k)}{\pi(D_{k+1})},\end{aligned}$$

and since

$$\pi(D_{k+1}) = \pi(y_{k+1}, D_k) = \pi(y_{k+1} \mid D_k)\pi(D_k),$$

we have that

$$\begin{aligned}\pi(x_{k+1} \mid D_{k+1}) &= \frac{\pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)\pi(D_k)}{\pi(D_{k+1})} \\ &= \frac{\pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)\pi(D_k)}{\pi(y_{k+1} \mid D_k)\pi(D_k)} \\ &= \frac{\pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)}{\pi(y_{k+1} \mid D_k)}.\end{aligned}$$

Therefore the observation update is given by

$$\pi(x_{k+1} \mid D_{k+1}) = \frac{\pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k)}{\pi(y_{k+1} \mid D_k)}. \tag{4.8}$$

Combining the time evolution update (4.7) and the observation update (4.8) and neglecting constant factors yields the following updating formula for the posterior density:

$$\pi(x_{k+1} \mid D_{k+1}) \propto \pi(y_{k+1} \mid x_{k+1}) \int \pi(x_{k+1} \mid x_k)\pi(x_k \mid D_k)dx_k. \tag{4.9}$$

In the following sections, we review some well-known algorithms for special cases of the evolution-observation model (4.5)–(4.6) and derive the corresponding updating formulas.

4.4 Kalman-type filtering algorithms

In this section we review three of the main Kalman-type filtering algorithms for state estimation: the classical Kalman filter (KF), the extended Kalman filter (EKF), and the ensemble Kalman filter (EnKF). We also briefly describe some related filters and variations of these methods. There exists an extensive literature on Kalman-type filtering methods; any details omitted in the descriptions of the algorithms that follow can be found in [5, 62, 63, 64, 65].

4.4.1 Classical Kalman filter

Named after Rudolf E. Kalman for his contribution to the development of the algorithm in the classic work [66], the KF algorithm is a special case of the Bayesian filtering method described in Section 4.3 when the evolution-observation model (4.5)–(4.6) is linear and Gaussian. More specifically, denote the current state vector by the random variable $X_k \in \mathbb{R}^d$ and the updated state vector by X_{k+1}. We assume that the states are linked via the linear evolution equation

$$X_{k+1} = \mathsf{F}_{k+1} X_k + W_{k+1}, \quad k = 0, 1, 2, \ldots, \tag{4.10}$$

where F_{k+1} is a known matrix and the innovation term W_{k+1} is a normally distributed random variable with a known mean, which, without loss of generality, is assumed to vanish, and with a known covariance,

$$W_{k+1} \sim \mathcal{N}(0, \mathsf{C}).$$

We assume the linear observation model

$$Y_k = \mathsf{G}_k X_k + V_k, \quad k = 1, 2, \ldots, \tag{4.11}$$

where $Y_k \in \mathbb{R}^m$ denotes the observation, G_k is a known matrix, and the observation error V_k is normally distributed with zero mean and known covariance,

$$V_k \sim \mathcal{N}(0, \mathsf{D}).$$

Note that while the covariance matrices C and D may be k-dependent, here we consider the time-invariant cases. Moreover, we assume that the noise vectors are mutually independent, i.e.,

$$W_k \perp W_\ell, \quad V_k \perp V_\ell, \quad k \neq \ell$$

and

$$W_k \perp V_\ell.$$

We also assume that the distribution of the initial state X_0, $\pi(x_0) = \pi(x_0 \mid D_0)$, where D_0 is interpreted as an empty observation, is known and Gaussian.

To derive the time evolution update, we introduce some notation which follows [5, 63]. Let $X_{k|k}$ denote the random variable with probability $\pi(x_k \mid D_k)$ representing the current state at time k as an estimate of the true state $x_k \in \mathbb{R}^d$, and let $X_{k+1|k}$ denote the prior state at time $k+1$ obtained by propagating $X_{k|k}$ by the formula

$$X_{k+1|k} = \mathsf{F}_{k+1} X_{k|k} + W_{k+1}, \tag{4.12}$$

which follows from the evolution equation (4.10). We call $X_{k+1|k}$ the predictor of the state x_{k+1}.

If we assume that $X_{k|k}$ is normally distributed with known mean $\overline{x}_{k|k}$ and covariance $\mathsf{\Gamma}_{k|k}$,

$$X_{k|k} \sim \mathcal{N}(\overline{x}_{k|k}, \mathsf{\Gamma}_{k|k}), \tag{4.13}$$

which implies that the density $\pi(x_k \mid D_k)$ is of the form

$$\pi(x_k \mid D_k) \propto \exp\left\{ -\frac{1}{2}(x_k - \overline{x}_{k|k})^{\mathsf{T}} \mathsf{\Gamma}_{k|k}^{-1} (x_k - \overline{x}_{k|k}) \right\},$$

and that $X_{k|k}$ is independent of W_{k+1}, then before the new data are taken into consideration, we have that

$$X_{k+1|k} \sim \mathcal{N}(\overline{x}_{k+1|k}, \mathsf{\Gamma}_{k+1|k}),$$

where

$$\overline{x}_{k+1|k} = \mathsf{F}_{k+1} \overline{x}_{k|k} \tag{4.14}$$

and

$$\mathsf{\Gamma}_{k+1|k} = \mathsf{F}_{k+1} \mathsf{\Gamma}_{k|k} \mathsf{F}_{k+1}^{\mathsf{T}} + \mathsf{C}. \tag{4.15}$$

Indeed, since by (4.12) the predictor $X_{k+1|k}$ is the sum of two normally distributed random variables, it follows immediately that $X_{k+1|k}$ is normally distributed, and the

expression for the mean (4.14) can be derived as follows:

$$\begin{aligned}
\overline{x}_{k+1|k} &= E[X_{k+1|k}] \\
&= E[\mathsf{F}_{k+1}X_{k|k} + W_{k+1}] \\
&= \mathsf{F}_{k+1}E[X_{k|k}] + E[W_{k+1}] \\
&= \mathsf{F}_{k+1}\overline{x}_{k|k}.
\end{aligned}$$

The formula for the covariance (4.15) can be found in a similar way, since

$$\begin{aligned}
\mathsf{\Gamma}_{k+1|k} &= E[X_{k+1|k}X_{k+1|k}^{\mathsf{T}}] - \overline{x}_{k+1|k}\overline{x}_{k+1|k}^{\mathsf{T}} \\
&= E[(\mathsf{F}_{k+1}X_{k|k} + W_{k+1})(\mathsf{F}_{k+1}X_{k|k} + W_{k+1})^{\mathsf{T}}] - \mathsf{F}_{k+1}\overline{x}_{k|k}(\mathsf{F}_{k+1}\overline{x}_{k|k})^{\mathsf{T}} \\
&= E[\mathsf{F}_{k+1}X_{k|k}X_{k|k}^{\mathsf{T}}\mathsf{F}_{k+1}^{\mathsf{T}} + W_{k+1}X_{k|k}^{\mathsf{T}}\mathsf{F}_{k+1}^{\mathsf{T}} + \mathsf{F}_{k+1}X_{k|k}W_{k+1}^{\mathsf{T}} + W_{k+1}W_{k+1}^{\mathsf{T}}] \\
&\quad - \mathsf{F}_{k+1}\overline{x}_{k|k}\overline{x}_{k|k}^{\mathsf{T}}\mathsf{F}_{k+1}^{\mathsf{T}} \\
&= \mathsf{F}_{k+1}\big(E[X_{k|k}X_{k|k}^{\mathsf{T}}] - \overline{x}_{k|k}\overline{x}_{k|k}^{\mathsf{T}}\big)\mathsf{F}_{k+1}^{\mathsf{T}} + E[W_{k+1}W_{k+1}^{\mathsf{T}}] \\
&= \mathsf{F}_{k+1}\mathsf{\Gamma}_{k|k}\mathsf{F}_{k+1}^{\mathsf{T}} + \mathsf{C}.
\end{aligned}$$

Therefore the prior density of the propagated state, $\pi(x_{k+1} \mid D_k)$, is of the form

$$\pi(x_{k+1} \mid D_k) \propto \exp\left\{ -\frac{1}{2}(x_{k+1} - \overline{x}_{k+1|k})^{\mathsf{T}}\mathsf{\Gamma}_{k+1|k}^{-1}(x_{k+1} - \overline{x}_{k+1|k})\right\}, \tag{4.16}$$

where the prediction mean $\overline{x}_{k+1|k}$ is given in equation (4.14) and the prediction covariance $\mathsf{\Gamma}_{k+1|k}$ is given in equation (4.15).

As in Section 4.3, the update of the prior in light of the new observed data is an application of Bayes' theorem. Since $V_{k+1} \sim \mathcal{N}(0, \mathsf{D})$, the observation model (4.11) defines a likelihood density

$$y_{k+1} \mid x_{k+1} \sim \mathcal{N}(\mathsf{G}_{k+1}x_{k+1}, \mathsf{D}),$$

hence the conditional density of y_{k+1} given x_{k+1} is of the form

$$\pi(y_{k+1} \mid x_{k+1}) \propto \exp\left\{ -\frac{1}{2}(y_{k+1} - \mathsf{G}_{k+1}x_{k+1})^{\mathsf{T}}\mathsf{D}^{-1}(y_{k+1} - \mathsf{G}_{k+1}x_{k+1})\right\}. \tag{4.17}$$

Combining the Gaussian likelihood (4.17) and prior (4.16), via Bayes' theorem, it follows that the posterior state density is a Gaussian density of the form

$$\begin{aligned}\pi(x_{k+1} \mid D_{k+1}) &\propto \pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k) \\ &= \exp\Big\{ -\frac{1}{2}(y_{k+1} - \mathsf{G}_{k+1}x_{k+1})^{\mathsf{T}}\mathsf{D}^{-1}(y_{k+1} - \mathsf{G}_{k+1}x_{k+1}) \\ &\quad -\frac{1}{2}(x_{k+1} - \overline{x}_{k+1|k})^{\mathsf{T}}\mathsf{\Gamma}_{k+1|k}^{-1}(x_{k+1} - \overline{x}_{k+1|k})\Big\}.\end{aligned}$$

The posterior state mean $\overline{x}_{k+1|k+1}$ and posterior error covariance $\mathsf{\Gamma}_{k+1|k+1}$ can be obtained by maximizing $\pi(x_{k+1} \mid D_{k+1})$ with respect to x_{k+1}, which is equivalent to minimizing the cost function

$$\begin{aligned}\mathcal{J}(x) &= (y_{k+1} - \mathsf{G}_{k+1}x)^{\mathsf{T}}\mathsf{D}^{-1}(y_{k+1} - \mathsf{G}_{k+1}x) \\ &\quad + (x - \overline{x}_{k+1|k})^{\mathsf{T}}\mathsf{\Gamma}_{k+1|k}^{-1}(x - \overline{x}_{k+1|k}),\end{aligned} \tag{4.18}$$

i.e.,

$$\overline{x}_{k+1|k+1} = \arg\min_x \mathcal{J}(x).$$

Letting $x_c = x - \overline{x}_{k+1|k}$ and $y_c = y_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k}$, the cost function (4.18) becomes

$$\mathcal{J}(x_c) = (y_c - \mathsf{G}_{k+1}x_c)^{\mathsf{T}}\mathsf{D}^{-1}(y_c - \mathsf{G}_{k+1}x_c) + x_c^{\mathsf{T}}\mathsf{\Gamma}_{k+1|k}^{-1}x_c.$$

After rearranging and combining terms, we have

$$\mathcal{J}(x_c) = x_c^{\mathsf{T}}(\mathsf{\Gamma}_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}\mathsf{G}_{k+1})x_c - 2x_c^{\mathsf{T}}\mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}y_c + y_c^{\mathsf{T}}\mathsf{D}^{-1}y_c.$$

Differentiation with respect to x_c yields

$$\frac{\partial \mathcal{J}}{\partial x_c} = 2(\mathsf{\Gamma}_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}\mathsf{G}_{k+1})x_c - 2\mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}y_c,$$

and setting $\dfrac{\partial \mathcal{J}}{\partial x_c} = 0$, we find the critical point

$$x_c = (\mathsf{\Gamma}_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}\mathsf{G}_{k+1})^{-1}\mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}y_c,$$

which is a minimizer since $\dfrac{\partial^2 \mathcal{J}}{\partial (x_c)^2} = 2(\Gamma_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1})$ is positive definite.

Therefore, recalling how we defined x_c, it follows that the posterior state mean is

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - \mathsf{G}_{k+1} \overline{x}_{k+1|k}), \tag{4.19}$$

where the matrix K_{k+1}, known as the *Kalman gain*, is

$$\mathsf{K}_{k+1} = (\Gamma_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1})^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1}, \tag{4.20}$$

or, equivalently,

$$\mathsf{K}_{k+1} = \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}} (\mathsf{G}_{k+1} \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}. \tag{4.21}$$

The equivalence of formulas (4.20) and (4.21) is shown using the matrix identity

$$(\mathsf{I} + \mathsf{PQ})^{-1} \mathsf{P} = \mathsf{P} (\mathsf{I} + \mathsf{QP})^{-1} \tag{4.22}$$

for invertible matrices $\mathsf{I} + \mathsf{PQ}$ and $\mathsf{I} + \mathsf{QP}$, which is derived by multiplying the equivalence expression

$$\mathsf{P}(\mathsf{I} + \mathsf{QP}) = \mathsf{P} + \mathsf{PQP} = (\mathsf{I} + \mathsf{PQ})\mathsf{P}$$

on the left by $(\mathsf{I} + \mathsf{QP})^{-1}$ and on the right by $(\mathsf{I} + \mathsf{PQ})^{-1}$. From formula (4.20), we have that

$$\begin{aligned} (\Gamma_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1})^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} &= \left[(\mathsf{I} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1} \Gamma_{k+1|k}) \Gamma_{k+1|k}^{-1} \right]^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \\ &= \Gamma_{k+1|k} (\mathsf{I} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1} \Gamma_{k+1|k})^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1}, \end{aligned}$$

and applying identity (4.22) with the matrix $\mathsf{G}_{k+1}^{\mathsf{T}}$ playing the role of P and $\mathsf{D}^{-1} \mathsf{G}_{k+1} \Gamma_{k+1|k}$ the role of Q, we get that

$$(\Gamma_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1})^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} = \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}} (\mathsf{I} + \mathsf{D}^{-1} \mathsf{G}_{k+1} \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}})^{-1} \mathsf{D}^{-1}.$$

By combining the inverses on the righthand side, the above expression further simplifies to

$$\begin{aligned} (\Gamma_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} \mathsf{G}_{k+1})^{-1} \mathsf{G}_{k+1}^{\mathsf{T}} \mathsf{D}^{-1} &= \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}} \left[\mathsf{D} (\mathsf{I} + \mathsf{D}^{-1} \mathsf{G}_{k+1} \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}}) \right]^{-1} \\ &= \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}} (\mathsf{D} + \mathsf{G}_{k+1} \Gamma_{k+1|k} \mathsf{G}_{k+1}^{\mathsf{T}})^{-1}, \end{aligned}$$

which proves the equivalence of (4.20) and (4.21), as desired. Note that identity (4.22) can be similarly used to derive the Sherman-Morrison-Woodbury formula, sometimes referred to as the Matrix Inversion Lemma, for computing the inverse of a low-rank update of an invertible matrix; see, e.g., [67, 62]. The advantage of writing the Kalman gain in the form (4.21) for $m << n$ is that the inverse of $\mathsf{D} + \mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}$ has dimensions $m \times m$, whereas the inverse of $\mathsf{\Gamma}_{k+1|k}^{-1} + \mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{D}^{-1}\mathsf{G}_{k+1}$ in the form (4.20) has dimensions $n \times n$. Form (4.21) may therefore be computationally more efficient.

To find the posterior error covariance, we start by considering

$$\begin{aligned} e_{k+1|k+1} &= x_{k+1} - \overline{x}_{k+1|k+1} \\ &= x_{k+1} - \overline{x}_{k+1|k} - \mathsf{K}_{k+1}(y_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k}) \\ &= x_{k+1} - \overline{x}_{k+1|k} - \mathsf{K}_{k+1}(\mathsf{G}_{k+1}x_{k+1} + v_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k}) \\ &= x_{k+1} - \overline{x}_{k+1|k} - \mathsf{K}_{k+1}(v_{k+1} + \mathsf{G}_{k+1}(x_{k+1} - \overline{x}_{k+1|k})), \end{aligned}$$

and letting

$$e_{k+1|k} = x_{k+1} - \overline{x}_{k+1|k},$$

we have that

$$\begin{aligned} e_{k+1|k+1} &= e_{k+1|k} - \mathsf{K}_{k+1}(v_{k+1} + \mathsf{G}_{k+1}e_{k+1|k}) \\ &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})e_{k+1|k} - \mathsf{K}_{k+1}v_{k+1}. \end{aligned}$$

It then follows that

$$\begin{aligned} \mathsf{\Gamma}_{k+1|k+1} &= \mathrm{cov}(e_{k+1|k+1}) \\ &= \mathrm{cov}[(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})e_{k+1|k} - \mathsf{K}_{k+1}v_{k+1}] \\ &= \mathrm{cov}[(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})e_{k+1|k}] + \mathrm{cov}(\mathsf{K}_{k+1}v_{k+1}) \\ &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\mathrm{cov}(e_{k+1|k})(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}} + \mathsf{K}_{k+1}\mathrm{cov}(v_{k+1})\mathsf{K}_{k+1}^{\mathsf{T}}, \end{aligned}$$

and we arrive at the expression

$$\begin{aligned}\Gamma_{k+1|k+1} &= (\mathsf{I}-\mathsf{K}_{k+1}\mathsf{G}_{k+1})\Gamma_{k+1|k}(\mathsf{I}-\mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}}+\mathsf{K}_{k+1}\mathsf{D}\mathsf{K}_{k+1}^{\mathsf{T}}\\ &= \Gamma_{k+1|k}-\mathsf{K}_{k+1}\mathsf{G}_{k+1}\Gamma_{k+1|k}-\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{K}_{k+1}^{\mathsf{T}}\\ &\quad +\mathsf{K}_{k+1}(\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}+\mathsf{D})\mathsf{K}_{k+1}^{\mathsf{T}}.\end{aligned}$$

Using formula (4.21) for the Kalman gain K_{k+1}, we have that

$$\mathsf{K}_{k+1}(\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}+\mathsf{D})\mathsf{K}_{k+1}^{\mathsf{T}}=\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}\mathsf{K}_{k+1}^{\mathsf{T}},$$

which we substitute in the expression for $\Gamma_{k+1|k+1}$ to obtain, after simplification,

$$\begin{aligned}\Gamma_{k+1|k+1} &= \Gamma_{k+1|k}-\mathsf{K}_{k+1}\mathsf{G}_{k+1}\Gamma_{k+1|k}\\ &= (\mathsf{I}-\mathsf{K}_{k+1}\mathsf{G}_{k+1})\Gamma_{k+1|k}. \qquad (4.23)\end{aligned}$$

Therefore *a posteriori* we have that

$$x_{k+1}\mid y_{k+1}\sim\mathcal{N}(\overline{x}_{k+1|k+1},\Gamma_{k+1|k+1}),$$

where the posterior mean $\overline{x}_{k+1|k+1}$ and the posterior error covariance $\Gamma_{k+1|k+1}$ are as in formulas (4.19) and (4.23), respectively. The classical KF algorithm as derived here is summarized in Algorithm 2. Note that the classical KF algorithm is optimal in the least squares sense.

We remark that by introducing the notation

$$||z||_{\mathsf{M}}^2=z^{\mathsf{T}}\mathsf{M}^{-1}z$$

for $z\in\mathbb{R}^n$ and $\mathsf{M}\in\mathbb{R}^{n\times n}$ a symmetric positive definite matrix, we can write the cost function (4.18) as

$$\mathcal{J}(x)=||y_{k+1}-\mathsf{G}_{k+1}x||_{\mathsf{D}}^2+||x-\overline{x}_{k+1|k}||_{\Gamma_{k+1|k}}^2.$$

Hence,

$$\begin{aligned}x_{k+1} &= \arg\min_x \mathcal{J}(x)\\ &= \arg\min_x\left\{||y_{k+1}-\mathsf{G}_{k+1}x||_{\mathsf{D}}^2+||x-\overline{x}_{k+1|k}||_{\Gamma_{k+1|k}}^2\right\},\end{aligned}$$

Algorithm 2 Classical Kalman Filter

Given an initial prior distribution $\pi(x_0) = \pi(x_0 \mid D_0) \sim \mathcal{N}(\overline{x}_{0|0}, \Gamma_{0|0})$, set $k = 0$.

1. *Prediction step*: Update the prior mean and covariance using the formulas

$$\overline{x}_{k+1|k} = \mathsf{F}_{k+1}\overline{x}_{k|k}$$

and

$$\Gamma_{k+1|k} = \mathsf{F}_{k+1}\Gamma_{k|k}\mathsf{F}_{k+1}^{\mathsf{T}} + \mathsf{C}.$$

2. *Observation update*: After observing y_{k+1}, update the posterior mean and error covariance via the formulas

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k})$$

and

$$\Gamma_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\Gamma_{k+1|k},$$

where

$$\mathsf{K}_{k+1} = \Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}(\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

3. If $k < T$, set $k = k + 1$ and repeat from Step 1; otherwise, stop.

a formula that will become useful in later sections.

We further note that if the linear evolution-observation model (4.10)–(4.11) is time-invariant, i.e., if $\mathsf{F}_{k+1} \equiv \mathsf{A}$ and $\mathsf{G}_k \equiv \mathsf{B}$ where A and B are constant matrices, then the formulas for the time evolution and observation updates of the classical Kalman filter summarized in Algorithm 2 simplify to those in Algorithm 3. In fact, if both the observation matrix B and the observation noise covariance matrix D are constant, it may happen that the prediction covariance $\Gamma_{k+1|k}$ converges over time to a steady-state value Γ, such that the Kalman gain also converges to a steady-state value,

$$\mathsf{K}_{k+1} = \Gamma_{k+1|k}\mathsf{B}^{\mathsf{T}}(\mathsf{B}\Gamma_{k+1|k}\mathsf{B}^{\mathsf{T}} + \mathsf{D})^{-1} \quad \longrightarrow \quad \Gamma\mathsf{B}^{\mathsf{T}}(\mathsf{B}\Gamma\mathsf{B}^{\mathsf{T}} + \mathsf{D})^{-1} = \mathsf{K},$$

making it only necessary to compute

$$\begin{aligned}\overline{x}_{k+1|k+1} &= \overline{x}_{k+1|k} + \mathsf{K}(y_{k+1} - \mathsf{B}\overline{x}_{k+1|k}) \\ &= \mathsf{A}\overline{x}_{k|k} + \mathsf{K}(y_{k+1} - \mathsf{BA}\overline{x}_{k|k}) \\ &= \mathsf{K}y_{k+1} + (\mathsf{I} - \mathsf{KB})\mathsf{A}\overline{x}_{k|k} \\ &= \mathsf{K}y_{k+1} + \widehat{\mathsf{A}}\overline{x}_{k|k}\end{aligned}$$

at each time instance from a certain step onward [5].

Algorithm 3 Classical Kalman Filter (Time-Invariant Case)

Given an initial prior distribution $\pi(x_0) = \pi(x_0 \mid D_0) \sim \mathcal{N}(\overline{x}_{0|0}, \mathsf{\Gamma}_{0|0})$, set $k = 0$.

1. *Prediction step*: Update the prior mean and covariance using the formulas

$$\overline{x}_{k+1|k} = \mathsf{A}\overline{x}_{k|k}$$

and

$$\mathsf{\Gamma}_{k+1|k} = \mathsf{A}\mathsf{\Gamma}_{k|k}\mathsf{A}^{\mathsf{T}} + \mathsf{C}.$$

2. *Observation update*: After observing y_{k+1}, update the posterior mean and error covariance via the formulas

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - \mathsf{B}\overline{x}_{k+1|k})$$

and

$$\mathsf{\Gamma}_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{B})\mathsf{\Gamma}_{k+1|k},$$

where

$$\mathsf{K}_{k+1} = \mathsf{\Gamma}_{k+1|k}\mathsf{B}^{\mathsf{T}}(\mathsf{B}\mathsf{\Gamma}_{k+1|k}\mathsf{B}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

3. If $k < T$, set $k = k + 1$ and repeat from Step 1; otherwise, stop.

A continuous-time version of the Kalman filter, known as the Kalman-Bucy filter, can be found in [68].

4.4.2 Extended Kalman filter

A limitation of the classical KF algorithm is the linearity assumption: the fact that Gaussian densities remain Gaussian under linear transformations allows for straight-

forward updating of the densities by simply updating the means and covariances. The EKF algorithm is a generalization of classical KF designed to accommodate nonlinear models, where nonlinear functions are replaced by their linearizations. Consider the evolution-observation model

$$X_{k+1} = F_{k+1}(X_k) + W_{k+1}, \quad k = 0, 1, 2, \ldots, \tag{4.24}$$

$$Y_k = G_k(X_k) + V_k, \quad k = 1, 2, \ldots, \tag{4.25}$$

where F_{k+1} and G_k are nonlinear, differentiable functions, and the noise processes W_{k+1} and V_k are mutually independent, zero-mean Gaussian random variables. We assume that the initial prior density $\pi(x_0) = \pi(x_0 \mid D_0)$ is given.

As in [5], let π_G denote a Gaussian approximation to the density π, meaning that π_G is the Gaussian density with the same mean and covariance as π. Assuming that we know

$$\pi_G(x_k \mid D_k) \sim \mathcal{N}(\overline{x}_{k|k}, \Gamma_{k|k}), \tag{4.26}$$

the time evolution update is given by

$$\begin{aligned} \pi(x_{k+1} \mid D_k) &= \int \pi(x_{k+1} \mid x_k)\pi(x_k \mid D_k)dx_k \\ &\approx \int \pi(x_{k+1} \mid x_k)\pi_G(x_k \mid D_k)dx_k. \end{aligned}$$

The Gaussian approximation of the transition kernel $\pi(x_{k+1} \mid x_k)$ is obtained by taking a first-order Taylor series expansion of the nonlinear operator F_{k+1} centered at $\overline{x}_{k|k}$, yielding

$$\begin{aligned} X_{k+1} &= F_{k+1}(X_k) + W_{k+1} \\ &\approx F_{k+1}(\overline{x}_{k|k}) + \mathcal{D}F_{k+1}(\overline{x}_{k|k})(X_k - \overline{x}_{k|k}) + W_{k+1}, \end{aligned}$$

where $\mathcal{D}F$ denotes the Jacobian of F. To simplify the notation, from this point forward we write

$$\mathcal{D}F_{k+1} = \mathcal{D}F_{k+1}(\overline{x}_{k|k}).$$

It follows from the prediction step of the classical KF that the Gaussian approximation prior to the observation update is

$$\pi_G(x_{k+1} \mid D_k) \sim \mathcal{N}(\overline{x}_{k+1|k}, \Gamma_{k+1|k}),$$

where

$$\overline{x}_{k+1|k} = F_{k+1}(\overline{x}_{k|k}) \tag{4.27}$$

and

$$\Gamma_{k+1|k} = \mathcal{D}F_{k+1}\Gamma_{k|k}\mathcal{D}F_{k+1}^{\mathsf{T}} + \mathsf{C}. \tag{4.28}$$

The observation update follows from Bayes' theorem:

$$\begin{aligned}
\pi(x_{k+1} \mid D_{k+1}) &\propto \pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k) \\
&\approx \pi(y_{k+1} \mid x_{k+1})\pi_G(x_{k+1} \mid D_k) \\
&\propto \exp\Big\{ -\frac{1}{2}(y_{k+1} - G_{k+1}(x_{k+1}))^{\mathsf{T}}\mathsf{D}^{-1}(y_{k+1} - G_{k+1}(x_{k+1})) \\
&\qquad -\frac{1}{2}(x_{k+1} - \overline{x}_{k+1|k})^{\mathsf{T}}\Gamma_{k+1|k}^{-1}(x_{k+1} - \overline{x}_{k+1|k})\Big\}.
\end{aligned}$$

A Gaussian approximation to this density is obtained by linearizing G_{k+1} about the point $\overline{x}_{k+1|k}$ using the first-order Taylor polynomial

$$G_{k+1}(x_{k+1}) \approx G_{k+1}(\overline{x}_{k+1|k}) + \mathcal{D}G_{k+1}(\overline{x}_{k+1|k})(x_{k+1} - \overline{x}_{k+1|k}),$$

where $\mathcal{D}G$ is the Jacobian of G and we write

$$\mathcal{D}G_{k+1} = \mathcal{D}G_{k+1}(\overline{x}_{k+1|k}),$$

so that the Gaussian approximation to the posterior density is given by

$$\pi_G(x_{k+1} \mid D_{k+1}) \sim \mathcal{N}(\overline{x}_{k+1|k+1}, \Gamma_{k+1|k+1}),$$

where

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - G_{k+1}(\overline{x}_{k+1|k})), \tag{4.29}$$

$$\Gamma_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathcal{D}G_{k+1})\Gamma_{k+1|k}, \tag{4.30}$$

and the Kalman gain is

$$\mathsf{K}_{k+1} = \mathsf{\Gamma}_{k+1|k}\mathcal{D}G_{k+1}^{\mathsf{T}}(\mathcal{D}G_{k+1}\mathsf{\Gamma}_{k+1|k}\mathcal{D}G_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}. \tag{4.31}$$

Similarly to what was done for the classical KF, the EKF posterior updating formulas (4.29) and (4.30) are obtained by minimizing the cost function

$$\begin{aligned} \mathcal{J}(x) &= ||y_{k+1} - G_{k+1}(x)||^2_{\mathsf{D}} + ||x - \overline{x}_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}} \\ &\approx ||y_{k+1} - G_{k+1}(\overline{x}_{k+1|k}) - \mathcal{D}G_{k+1}(x - \overline{x}_{k+1|k})||^2_{\mathsf{D}} \\ &\quad + ||x - \overline{x}_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}}, \end{aligned} \tag{4.32}$$

so that

$$\begin{aligned} x_{k+1} &= \arg\min_x \mathcal{J}(x) \\ &= \arg\min_x \Big\{ ||y_{k+1} - G_{k+1}(\overline{x}_{k+1|k}) - \mathcal{D}G_{k+1}(x - \overline{x}_{k+1|k})||^2_{\mathsf{D}} \\ &\quad + ||x - \overline{x}_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}} \Big\}. \end{aligned}$$

A summary of the EKF algorithm is given in Algorithm 4.

A few remarks are in order. First, we note that EKF is suboptimal due to the errors introduced using first-order Taylor series approximations of the nonlinear operators F and G. Moreover, numerical computation of the Jacobian matrices $\mathcal{D}F$ and $\mathcal{D}G$ can be very costly, especially when dealing with large nonlinear systems, as in numerical weather prediction [63]. The optimal interpolation (OI) strategy proposed in [69, 70] attempts to remedy this issue by replacing the prior covariance matrix $\mathsf{\Gamma}_{k+1|k}$ with a constant matrix R, thereby removing the cost associated with computing $\mathcal{D}F$ in formula (4.28). As illustrated in [71], the OI method is equivalent to variational analysis techniques such as spectral statistical interpolation (SSI) [72] and the three-dimensional variational approach (3D-Var) [73], which uses iterative schemes such as conjugate gradient [74, 75, 12, 7] or quasi-Newton methods [76, 77] to numerically

Algorithm 4 Extended Kalman Filter

Given an initial prior distribution $\pi(x_0) = \pi(x_0 \mid D_0)$:

1. *Initialization*: If $\pi(x_0)$ is not Gaussian, replace it by a Gaussian approximation $\pi_G(x_0) \sim \mathcal{N}(\overline{x}_{0|0}, \Gamma_{0|0})$. Set $k = 0$.

2. *Prediction step*: Update the prior mean and covariance using the formulas

$$\overline{x}_{k+1|k} = F_{k+1}(\overline{x}_{k|k})$$

and

$$\Gamma_{k+1|k} = \mathcal{D}F_{k+1}\Gamma_{k|k}\mathcal{D}F_{k+1}^{\mathsf{T}} + \mathsf{C}$$

where

$$\mathcal{D}F_{k+1} = \mathcal{D}F_{k+1}(\overline{x}_{k|k}).$$

3. *Observation update*: After observing y_{k+1}, update the posterior mean and error covariance via the formulas

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - G_{k+1}(\overline{x}_{k+1|k}))$$

and

$$\Gamma_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathcal{D}G_{k+1})\Gamma_{k+1|k},$$

where

$$\mathcal{D}G_{k+1} = \mathcal{D}G_{k+1}(\overline{x}_{k+1|k})$$

and

$$\mathsf{K}_{k+1} = \Gamma_{k+1|k}\mathcal{D}G_{k+1}^{\mathsf{T}}(\mathcal{D}G_{k+1}\Gamma_{k+1|k}\mathcal{D}G_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

4. If $k < T$, set $k = k + 1$ and repeat from Step 2; otherwise, stop.

minimize the cost function (4.32). However, the success of these methods depends heavily on the choice of the matrix R. Finally, when computational time is not an issue, a variant of the EKF algorithm can be implemented which includes an inner iteration loop in the observation update; see [5] for details.

4.4.3 Ensemble Kalman filter

As anticipated in the previous section, a major issue when using the EKF algorithm on a large, nonlinear system with evolution-observation model (4.24)–(4.25) is the cost of computing the Jacobian matrices $\mathcal{D}F$ and $\mathcal{D}G$ needed for updating the prior covariance matrix $\mathsf{\Gamma}_{k+1|k}$ in equation (4.28) and the posterior error covariance matrix $\mathsf{\Gamma}_{k+1|k+1}$ in equation (4.30), respectively. Moreover, the error introduced by using first-order Taylor series approximations, and thereby ignoring all third and higher order moments, of the nonlinear operators F and G can lead to further computational difficulties.

A related filtering method for nonlinear systems which sidesteps some of these problems is the EnKF algorithm, first proposed by Geir Evensen in [78]. The EnKF algorithm is based on the use of Monte Carlo techniques and ensemble statistics within the classical KF framework. More specifically, the idea behind EnKF is to approximate both $\mathsf{\Gamma}_{k+1|k}$ and $\mathsf{\Gamma}_{k+1|k+1}$ with sample covariance matrices computed using the sample mean of a random sample of realizations of the state variable. By integrating this ensemble of realizations forward in time and by properly updating the ensemble when data arrive, the error statistics needed in the EKF algorithm can be computed directly from the ensemble with no additional computational expenses.

A motivation for the development of the EnKF algorithm comes from a multilayer nonlinear quasi-geostrophic (QG) ocean circulation model [79, 80]. In [79], where a QG ocean model with closed boundary conditions is considered, it is shown that the nonlinearity of the problem has a strong effect on the propagation of the error covariance, as well as on the data assimilation process. In fact, linearization of the evolution equation introduces an unbounded growth in error variance, and an

abundance of data or use of higher order moments is required to control the instability. Such use of higher order moments can cause EKF to be computationally unfeasible in this setting. In [80] the application of EKF to the QG ocean model further includes the possibility of open boundaries with inflow and outflow, complicating the error covariance propagation equation and requiring the implementation of stable and consistent open boundary conditions. The approximate boundary schemes introduced result in the loss of positive definiteness of the error covariance matrix over long integration times due to use of a non-symmetrical update, making it necessary to add a multiple of the identity to guarantee positive definiteness. Therefore, the general aim in deriving the EnKF in [78] was to find a better method for computing the error covariance.

In our outline of the general EnKF algorithm, we follow [78, 81, 82, 63, 64]. Consider the evolution-observation model

$$X_{k+1} = F_{k+1}(X_k) + W_{k+1}, \quad k = 0, 1, 2, \ldots, \tag{4.33}$$

$$Y_k = G_k(X_k) + V_k, \quad k = 1, 2, \ldots, \tag{4.34}$$

where F_{k+1} and G_k are nonlinear operators, and the noise processes W_{k+1} and V_k are mutually independent. The model (4.33)–(4.34) is similar to the model (4.24)–(4.25) used in the derivation of the EKF algorithm, except for the fact that now the distributions of the noise processes W_{k+1} and V_k are not required to be Gaussian, although it is often assumed that the prior distribution and the observation likelihood are Gaussian in the observation update. Thus, for the sake of definiteness, we will assume $V_k \sim \mathcal{N}(0, \mathsf{D})$ later in our discussion.

Suppose that the distribution of the random variable X_k is given in the form of a random sample, referred to as the *state ensemble*, comprising N realizations drawn from the distribution,

$$\mathcal{S}_k = \left\{x_k^1, x_k^2, \ldots, x_k^N\right\},$$

which, using our previous notation, we denote by

$$\mathcal{S}_{k|k} = \left\{x_{k|k}^1, x_{k|k}^2, \ldots, x_{k|k}^N\right\}.$$

In the time evolution step, instead of propagating the mean and covariance as done in KF and EKF, we propagate each state ensemble member $x^n_{k|k}$, $n = 1, 2, \ldots, N$, via the evolution equation (4.33), obtaining

$$x^n_{k+1|k} = F_{k+1}(x^n_{k|k}) + w^n_{k+1}, \quad n = 1, 2, \ldots, N, \tag{4.35}$$

where each w^n_{k+1} is an independently drawn realization of the random variable W_{k+1}. Note that no approximation of F_{k+1} is needed. The mean $\overline{x}_{k+1|k}$ and covariance $\Gamma_{k+1|k}$ prior to the observation update are computed from the prediction ensemble

$$\mathcal{S}_{k+1|k} = \left\{x^1_{k+1|k}, x^2_{k+1|k}, \ldots, x^N_{k+1|k}\right\} \tag{4.36}$$

via the ensemble formulas

$$\overline{x}_{k+1|k} = \frac{1}{N}\sum_{n=1}^{N} x^n_{k+1|k} \tag{4.37}$$

and

$$\Gamma_{k+1|k} = \frac{1}{N-1}\sum_{n=1}^{N}(x^n_{k+1|k} - \overline{x}_{k+1|k})(x^n_{k+1|k} - \overline{x}_{k+1|k})^{\mathsf{T}}, \tag{4.38}$$

respectively. Introducing the $d \times N$ matrix

$$\mathsf{U}_{k+1|k} = \frac{1}{\sqrt{N-1}}\left[\; x^1_{k+1|k} - \overline{x}_{k+1|k}, \quad x^2_{k+1|k} - \overline{x}_{k+1|k}, \quad \cdots, \quad x^N_{k+1|k} - \overline{x}_{k+1|k} \;\right], \tag{4.39}$$

we can write (4.38) compactly as

$$\Gamma_{k+1|k} = \mathsf{U}_{k+1|k}\mathsf{U}^{\mathsf{T}}_{k+1|k}. \tag{4.40}$$

The observation update for the EnKF algorithm is performed by first generating an ensemble of virtual observed values at the current time. More specifically, letting y_{k+1} denote the actual observation at time $k+1$ and recalling that the observations are linked to the state variables via equation (4.34), we generate an ensemble of fictitious observations about y_{k+1} by the formula

$$y^n_{k+1} = y_{k+1} + v^n_{k+1}, \quad n = 1, 2, \ldots, N,$$

where each v_{k+1}^n is an independently drawn realization of V_{k+1}. We refer to the sample

$$\left\{y_{k+1}^1, y_{k+1}^2, \ldots, y_{k+1}^N\right\} \tag{4.41}$$

as the *observation ensemble*. Here, for simplicity, the size of the observation ensemble is the same as the size of the state ensemble, but this is not needed in general. As explained in [81], the use of an observation ensemble is vital in ensuring that the updated ensemble does not have a variance that is too low, which will be demonstrated later in this section.

The elements of the posterior state ensemble satisfy

$$x_{k+1|k+1}^n = \arg\min_x \left\{ ||y_{k+1}^n - G_{k+1}(x)||_{\mathsf{D}}^2 + ||x - x_{k+1|k}^n||_{\mathsf{\Gamma}_{k+1|k}}^2 \right\} \tag{4.42}$$

for $n = 1, 2, \ldots, N$. If the observation model is linear and Gaussian, i.e., when $G_{k+1} = \mathsf{G}_{k+1}$ for some constant matrix G_{k+1} and $V_{k+1} \sim \mathcal{N}(0, \mathsf{D})$, we can generate the posterior state ensemble from the formula

$$x_{k+1|k+1}^n = x_{k+1|k}^n + \mathsf{K}_{k+1}(y_{k+1}^n - \mathsf{G}_{k+1} x_{k+1|k}^n), \quad n = 1, 2, \ldots, N, \tag{4.43}$$

where the Kalman gain is (4.21), as for the classical KF algorithm, with the caveat that now $\mathsf{\Gamma}_{k+1|k}$ is computed using ensemble statistics, and the same Kalman gain is used to update each ensemble member. It follows from (4.43) that the posterior ensemble mean $\overline{x}_{k+1|k+1}$, which is the EnKF new state estimate, is given by

$$\overline{x}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(\overline{y}_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k}), \tag{4.44}$$

where $\overline{x}_{k+1|k}$ is the prediction ensemble mean (4.37) and $\overline{y}_{k+1}$ is the observation ensemble mean, which asymptotically approaches the actual observation y_{k+1}.

As done in the classical KF algorithm, the posterior error covariance matrix $\mathsf{\Gamma}_{k+1|k+1}$ is derived by considering

$$\begin{aligned} e_{k+1|k+1}^n &= x_{k+1|k+1}^n - \overline{x}_{k+1|k+1} \\ &= x_{k+1|k}^n - \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1}^n - \overline{y}_{k+1}) - \mathsf{K}_{k+1}\mathsf{G}_{k+1}(x_{k+1|k}^n - \overline{x}_{k+1|k}), \end{aligned}$$

which, if we let

$$e^n_{k+1|k} = x^n_{k+1|k} - \overline{x}_{k+1|k}$$

and gather like terms, can be expressed as

$$e^n_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})e^n_{k+1|k} + \mathsf{K}_{k+1}(y^n_{k+1} - \overline{y}_{k+1})$$

for each $n = 1, 2, \ldots, N$. Since

$$\begin{aligned}\mathsf{\Gamma}_{k+1|k+1} &= \operatorname{cov}(e_{k+1|k+1}) \\ &= E[e_{k+1|k+1}e^{\mathsf{T}}_{k+1|k+1}],\end{aligned}$$

substituting the expression for $e_{k+1|k+1}$ and simplifying, we find that

$$\begin{aligned}\mathsf{\Gamma}_{k+1|k+1} &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})E[e_{k+1|k}e^{\mathsf{T}}_{k+1|k}](\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}} \\ &\quad + \mathsf{K}_{k+1}E[(y^n_{k+1} - \overline{y}_{k+1})(y^n_{k+1} - \overline{y}_{k+1})^{\mathsf{T}}]\mathsf{K}^{\mathsf{T}}_{k+1} \\ &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\mathsf{\Gamma}_{k+1|k}(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}} + \mathsf{K}_{k+1}\mathsf{D}\mathsf{K}^{\mathsf{T}}_{k+1} \\ &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\mathsf{\Gamma}_{k+1|k} - \mathsf{\Gamma}_{k+1|k}\mathsf{G}^{\mathsf{T}}_{k+1}\mathsf{K}^{\mathsf{T}}_{k+1} + \mathsf{K}_{k+1}(\mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}^{\mathsf{T}}_{k+1} + \mathsf{D})\mathsf{K}^{\mathsf{T}}_{k+1}.\end{aligned}$$

Moreover, from

$$\mathsf{\Gamma}_{k+1|k}\mathsf{G}^{\mathsf{T}}_{k+1}\mathsf{K}^{\mathsf{T}}_{k+1} = \mathsf{K}_{k+1}(\mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}^{\mathsf{T}}_{k+1} + \mathsf{D})\mathsf{K}^{\mathsf{T}}_{k+1}$$

and the definition of K_{k+1} in (4.21), it follows that the posterior error covariance is given by

$$\mathsf{\Gamma}_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\mathsf{\Gamma}_{k+1|k}, \tag{4.45}$$

which, like the Kalman gain, is of the same form as the posterior error covariance update (4.23) for the classical KF algorithm. Therefore, by updating each ensemble member via (4.43) with the Kalman gain computed in (4.21), the resulting posterior ensemble

$$\mathcal{S}_{k+1|k+1} = \left\{x^1_{k+1|k+1}, x^2_{k+1|k+1}, \ldots, x^N_{k+1|k+1}\right\}$$

has the same error covariance as would be obtained from (4.45), implying that the EnKF algorithm, in the limit as the ensemble size N tends to infinity, yields the same

posterior error results as the KF and EKF algorithms [82]. We can therefore obtain the posterior error covariance by computing the sample error covariance,

$$\Gamma_{k+1|k+1} = \mathsf{U}_{k+1|k+1}\mathsf{U}_{k+1|k+1}^{\mathsf{T}}, \tag{4.46}$$

where the formula for $\mathsf{U}_{k+1|k+1}$ is analogous to (4.39), replacing k with $k+1$. Note that this result hinges on the use of the observation ensemble (4.41), which is demonstrated as follows.

Consider the case when the observation ensemble is not employed and therefore the actual observation y_{k+1} is appears in the posterior ensemble generating formula

$$\widetilde{x}_{k+1|k+1}^n = x_{k+1|k}^n + \mathsf{K}_{k+1}(y_{k+1} - \mathsf{G}_{k+1}x_{k+1|k}^n)$$

for each $n = 1, 2, \ldots, N$. The posterior ensemble mean is then given by

$$\overline{\widetilde{x}}_{k+1|k+1} = \overline{x}_{k+1|k} + \mathsf{K}_{k+1}(y_{k+1} - \mathsf{G}_{k+1}\overline{x}_{k+1|k}),$$

and so the posterior difference is

$$\begin{aligned} \widetilde{e}_{k+1|k+1}^n &= x_{k+1|k}^n - \overline{x}_{k+1|k} - \mathsf{K}_{k+1}\mathsf{G}_{k+1}(x_{k+1|k}^n - \overline{x}_{k+1|k}) \\ &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})e_{k+1|k}^n. \end{aligned}$$

The corresponding posterior error covariance matrix

$$\widetilde{\Gamma}_{k+1|k+1} = (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\Gamma_{k+1|k}(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}}, \tag{4.47}$$

is the desired error covariance (4.45) where the factor of $(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}}$ on the right does not simplify because of the missing $\mathsf{K}_{k+1}\mathsf{D}\mathsf{K}_{k+1}^{\mathsf{T}}$ term in the derivation. The presence of $(\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^{\mathsf{T}}$ reduces the covariance too much, as illustrated in [81] using a simple scalar example in which $\Gamma_{k+1|k} = 1$ and $\mathsf{D} = 1$. In that case, the Kalman gain is

$$\begin{aligned} \mathsf{K}_{k+1} &= \frac{\Gamma_{k+1|k}\mathsf{G}_{k+1}}{\mathsf{G}_{k+1}^2\Gamma_{k+1|k} + \mathsf{D}} \\ &= \frac{\mathsf{G}_{k+1}}{\mathsf{G}_{k+1}^2 + 1}, \end{aligned}$$

therefore

$$\begin{aligned}
\Gamma_{k+1|k+1} &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})\Gamma_{k+1|k} \\
&= 1 - \frac{\mathsf{G}_{k+1}^2}{\mathsf{G}_{k+1}^2 + 1} \\
&= \frac{1}{\mathsf{G}_{k+1}^2 + 1},
\end{aligned}$$

whereas from (4.47),

$$\begin{aligned}
\widetilde{\Gamma}_{k+1|k+1} &= (\mathsf{I} - \mathsf{K}_{k+1}\mathsf{G}_{k+1})^2\Gamma_{k+1|k} \\
&= \left(1 - \frac{\mathsf{G}_{k+1}^2}{\mathsf{G}_{k+1}^2 + 1}\right)^2 \\
&= \frac{1}{(\mathsf{G}_{k+1}^2 + 1)^2}.
\end{aligned}$$

The choice of $\mathsf{G}_{k+1} = 1$ shows that $\Gamma_{k+1|k+1} = 0.5$ while $\widetilde{\Gamma}_{k+1|k+1} = 0.25$, a simple demonstration of how the posterior variance is lower than desired when an observation ensemble is not used. Note that the posterior ensemble mean (4.44) is not affected by the use of an observation ensemble.

A summary of the steps of the standard EnKF algorithm for a linear and Gaussian observation model is presented in the form of Algorithm 5. When actually implementing the EnKF algorithm, several issues need to be addressed. One such consideration is the size N of the ensemble. Although there are no set rules, it is generally acknowledged that N should be chosen to be at least equal to the dimensions of the state vector. However, in weather prediction models, for example, this condition is impossible to meet within computing time requirements, since the propagation step is expensive.

Another consideration, which simplifies the calculations, is that since the prior covariance matrix $\Gamma_{k+1|k}$ is always multiplied by the transpose of the linear operator

Algorithm 5 Ensemble Kalman Filter with a Linear Observation Model

Given an initial prior distribution $\pi(x_0) = \pi(x_0 \mid D_0)$:

1. *Initialization*: Draw the initial state ensemble
$$\mathcal{S}_0 = \mathcal{S}_{0|0} = \left\{x_{0|0}^1, x_{0|0}^2, \ldots, x_{0|0}^N\right\}$$
from $\pi(x_0)$. Set $k = 0$.
2. *Prediction step*: Using the current state ensemble $\mathcal{S}_{k|k}$,
   (a) Generate the prediction ensemble $\mathcal{S}_{k+1|k}$ using the nonlinear evolution equation
   $$x_{k+1|k}^n = F_{k+1}(x_{k|k}^n) + w_{k+1}^n, \quad n = 1, 2, \ldots, N,$$
   where each w_{k+1}^n is a randomly drawn realization of W_{k+1}.
   (b) Compute the prior mean and covariance using the ensemble statistics
   $$\overline{x}_{k+1|k} = \frac{1}{N}\sum_{n=1}^{N} x_{k+1|k}^n$$
   and
   $$\Gamma_{k+1|k} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{k+1|k}^n - \overline{x}_{k+1|k})(x_{k+1|k}^n - \overline{x}_{k+1|k})^{\mathsf{T}}.$$

(Continued on the next page)

Algorithm 5 Ensemble Kalman Filter with a Linear Observation Model (Cont.)

3. *Observation update*: After observing y_{k+1},
   (a) Generate the observation ensemble

$$\left\{y_{k+1}^1, y_{k+1}^2, \ldots, y_{k+1}^N\right\}$$

   via the formula

$$y_{k+1}^n = y_{k+1} + v_{k+1}^n, \quad n = 1, 2, \ldots, N,$$

   where each v_{k+1}^n is a randomly drawn realization of $V_{k+1} \sim \mathcal{N}(0, \mathsf{D})$.
   (b) Generate the posterior state ensemble $\mathcal{S}_{k+1|k+1}$ using the updating formula

$$x_{k+1|k+1}^n = x_{k+1|k}^n + \mathsf{K}_{k+1}(y_{k+1}^n - \mathsf{G}_{k+1} x_{k+1|k}^n), \quad n = 1, 2, \ldots, N,$$

   where

$$\mathsf{K}_{k+1} = \mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}(\mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

   (c) Compute the posterior mean and error covariance using the ensemble statistics

$$\overline{x}_{k+1|k+1} = \frac{1}{N}\sum_{n=1}^{N} x_{k+1|k+1}^n$$

   and

$$\mathsf{\Gamma}_{k+1|k+1} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{k+1|k+1}^n - \overline{x}_{k+1|k+1})(x_{k+1|k+1}^n - \overline{x}_{k+1|k+1})^{\mathsf{T}}.$$

4. If $k < T$, set $k = k + 1$ and repeat from Step 2; otherwise, stop.

matrix G_{k+1}, we only need the product $\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}$, which can be computed directly from the prediction ensemble via the formula

$$\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{k+1|k}^{n} - \overline{x}_{k+1|k})[\mathsf{G}_{k+1}(x_{k+1|k}^{n} - \overline{x}_{k+1|k})]^{\mathsf{T}}.$$

Although the use of an observation ensemble in the EnKF algorithm yields an asymptotically correct posterior error covariance, the random perturbations of the data can introduce additional sampling error for small ensembles, thereby resulting in suboptimal filtering [83, 63]. In order to avoid unwarranted tightening of the prior distribution, which may cause the observations to essentially be ignored, leading to filter divergence [84], the prior covariance matrix can be artificially widened, e.g., by multiplying $\mathsf{\Gamma}_{k+1|k}$ by a factor $1+\delta$ where $\delta > 0$. This commonly used technique, called multiplicative variance inflation [85, 83], can be implemented in terms of the matrix $\mathsf{U}_{k+1|k}$ defined in (4.39) by setting

$$\mathsf{U}_{k+1|k} \leftarrow (\sqrt{1+\delta})\mathsf{U}_{k+1|k},$$

from which it follows that

$$\mathsf{\Gamma}_{k+1|k} \leftarrow (1+\delta)\mathsf{U}_{k+1|k}\mathsf{U}_{k+1|k}^{\mathsf{T}},$$

where δ usually is found empirically [85]. Other variance inflation strategies include additive variance inflation [86] and adaptive inflation [87]. Localization methods, e.g., [88, 89, 90], are an alternative way to deal with the sampling error.

The posterior state ensemble updating formula (4.43) corresponding to the linear observation model works well for data of low dimensionality. In high dimensional situations, when explicitly computing the inverse matrix in the Kalman gain formula (4.21) may become unfeasible, the posterior state ensemble can be generated using the formula

$$x_{k+1|k+1}^{n} = x_{k+1|k}^{n} + \mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}\xi_{k+1}^{n}, \quad n = 1, 2, \ldots, N,$$

where ξ_{k+1}^n is the solution to the linear system

$$(\mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})\xi_{k+1}^n = y_{k+1}^n - \mathsf{G}_{k+1}x_{k+1|k}^n.$$

As remarked in [78], the EnKF algorithm is well-suited for parallelization, since each ensemble member can be integrated independently. The EnKF algorithm is also amenable to a vectorized implementation for certain problems; this comment will be elaborated upon in Chapter 8, when we exploit these properties further and test them on the Lorenz-63 system [11]. For additional suggestions regarding EnKF implementation strategies, including potential approximation techniques when the observation model is nonlinear, see, e.g., [82, 63].

Since its introduction the EnKF algorithm has been applied to a variety problems arising in science and engineering, particularly in the filtering of turbulent nonlinear dynamical systems with numerous degrees of freedom, such as models for weather and climate prediction [91]. The EnKF methodology has been applied to hydrological models for dual state-parameter estimation [92], continuous updating of reservoir models [93, 94, 95], doppler radar observations [96], and numerical weather prediction [97]. With the aim of making EnKF schemes operational in real atmospheric prediction settings, an investigation in [98] discusses how to choose an effective ensemble size, maintain balance in the data assimilation cycle, and account for model error; this was done by applying EnKF to a low-resolution version of the global forecast model used by the Canadian Meteorological Centre, assimilating simulated radiosonde, satellite thickness, and aircraft observations into a global primitive-equation model with simple forcing and dissipation. In [99] an EnKF scheme is applied to atmospheric data assimilation using real observations.

Numerous variations of the EnKF algorithm have been proposed in the literature, where these ensemble-type filters have been compared to variational analysis techniques such as the four-dimensional variational approach (4D-Var) [97, 100]. We discuss some algorithms related to Kalman-type filtering methods in the following section.

4.4.4 Related filters

In this section we describe some variants of the Kalman-type filtering algorithms. While acknowledging a vast literature on this topic, we restrict our overview to a limited selection of well-known variants, pointing to [82, 64, 63, 22] as references which include pertinent algorithms.

The Kalman-type filtering algorithms that we have described are often compared to data assimilation algorithms based on variational analysis methods. Two popular data assimilation techniques of this type are the 3D-Var and 4D-Var algorithms; see, e.g., [73, 101, 102]. These algorithms are well-known in the atmospheric and oceanic communities and are used operationally in weather forecasting [103, 104, 105, 106], where EnKF is thought to be a potentially viable alternative scheme. The aim of the 3D-Var algorithm is to solve the nonlinear Bayesian filtering problem of the type (4.24)–(4.25) by numerically minimizing the cost function

$$\mathcal{J}(x) = ||y_{k+1} - G_{k+1}(x)||^2_{\mathsf{D}} + ||x - \overline{x}_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}}$$

through use of iterative schemes like the conjugate gradient algorithm or various quasi-Newton methods. As is common with numerical minimization problems of this type, 3D-Var must address the issue regarding the computational cost of repeatedly computing the gradient of $\mathcal{J}$, especially due to the nonlinear function $G_{k+1}(x)$. A formulation which replaces $G_{k+1}(x)$ with its first-order Taylor series approximation about the predictive mean $\overline{x}_{k+1|k}$ is often implemented in practice. A hybrid assimilation scheme combining 3D-Var and EnKF is presented in [107].

While 3D-Var only considers the data at the analysis step, 4D-Var adds time as a dimension, accounting for multiple time levels, and thereby minimizes the cost function

$$\mathcal{J}(x) = \sum_{\ell=1}^{s} ||y_{k+\ell} - G_{k+\ell}(x)||^2_{\mathsf{D}_\ell} + ||x - \overline{x}_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}},$$

where $G_{k+\ell}$ is a nonlinear function describing the model forecast from time k to time $k+\ell$, $\ell = 1, \ldots, s$. The computational simplification designed for 3D-Var can be

adapted for 4D-Var methods as well. A combination of 4D-Var and EnKF methods is proposed in [108].

As previously noted, there is a wide variety of modified EnKF algorithms. A double EnKF algorithm using two state ensembles, where the statistics of one ensemble are used to update the other, is presented in [109]. This variation was intended to reduce the effects of possible inbreeding in the analysis phase, which may occur when updating the ensemble with a gain calculated from the same ensemble, but has led to some dispute; see the comments in [110] and the response in [111]. An adaptive version of the double EnKF algorithm is presented in [88].

The class of ensemble square root filters (EnSRFs) generates a posterior ensemble via the formula

$$\begin{aligned} x^n_{k+1|k+1} &= \overline{x}_{k+1|k+1} + \mathsf{U}^n_{k+1|k+1} \\ &= \overline{x}_{k+1|k+1} + \left(\sqrt{(N-1)\mathsf{\Gamma}_{k+1|k+1}}\right)^n, \end{aligned} \tag{4.48}$$

where the n index in the second term of the righthand side denotes the nth column of the matrix. Here the posterior mean $\overline{x}_{k+1|k+1}$ and error covariance $\mathsf{\Gamma}_{k+1|k+1}$ are updated using the formulas (4.29) and (4.30), respectively, with the Kalman gain computed by (4.31), as in the EKF algorithm described in Section 4.4.2. The name EnSRF stems from taking the "square root" of the covariance matrix, which follows from the factorization

$$\mathsf{\Gamma}_{k+1|k+1} = \frac{1}{N-1}\mathsf{U}_{k+1|k+1}\mathsf{U}^{\mathsf{T}}_{k+1|k+1},$$

thus implying, with a slight abuse of notation,

$$\mathsf{U}_{k+1|k+1} = \sqrt{N-1}\mathsf{\Gamma}^{1/2}_{k+1|k+1} = \sqrt{(N-1)\mathsf{\Gamma}_{k+1|k+1}}.$$

An EnSRF algorithm which avoids perturbing the observations, such that no observation ensemble is generated, is presented in [83]. This algorithm, based on redefining

the Kalman gain as

$$\begin{aligned}\widetilde{K} &= \Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}\left[\left(\sqrt{\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}+\mathsf{D}}\right)^{-1}\right]^{\mathsf{T}} \\ &\quad\times\left[\sqrt{\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}+\mathsf{D}}+\sqrt{\mathsf{D}}\right]^{-1},\end{aligned}$$

can be interpreted as a Monte Carlo version of a square root filter [112].

Ensemble transform Kalman filters (ETKFs) [113, 114, 115] aim to find a *transition matrix* T such that

$$\mathsf{U}_{k+1|k+1} = \mathsf{U}_{k+1|k}\mathsf{T}$$

and

$$\mathsf{U}_{k+1|k}\mathsf{T}(\mathsf{U}_{k+1|k}\mathsf{T})^{\mathsf{T}} = (K-1)\Gamma_{k+1|k+1},$$

where $\Gamma_{k+1|k+1}$ is the sample posterior covariance matrix. Different choices of the transformation matrix T have been suggested; see, e.g., [113, 116]. Since the posterior ensemble is generated as in (4.48), the ETKF algorithms fall into the EnSRF category.

Another member of the EnSRF class is the ensemble adjustment Kalman filter (EAKF) [117], similar in form to ETKF, but where the aim is to find an *adjustment matrix* A such that

$$\mathsf{U}_{k+1|k+1} = \mathsf{A}\mathsf{U}_{k+1|k}$$

and

$$\mathsf{A}\mathsf{U}_{k+1|k}(\mathsf{A}\mathsf{U}_{k+1|k})^{\mathsf{T}} = (K-1)\Gamma_{k+1|k+1}.$$

Computation of the adjustment matrix A requires two eigenvalue decompositions; see [117, 63]. It should be noted that, along with the EnSRF proposed in [83], the ETKF and EAKF algorithms do not require perturbing of the observations. Instead, these algorithms derive different linear operators to replace the traditional Kalman gain in order to counteract the over-reduced covariance. For a summary of these EnSRF algorithms, see [118].

Other related algorithms include, e.g., the ensemble smoother (ES) [119], the ensemble Kalman smoother (EnKS) [120], the singular evolutive extended Kalman

(SEEK) filter [121, 122, 123], the reduced rank square root (RRSQRT) filter [124], and error subspace statistical estimation (ESSE) [125]. We also mention the related Gaussian sum filters [62], iterated extended Kalman filters [84], and the mixture Kalman filter (MKF) [22, 126].

4.5 Particle filtering algorithms

An alternative to Kalman-type filtering schemes are the particle filter SMC (PF-SMC) methods, also referred to as bootstrap filters [22], frequently used for filtering nonlinear and/or non-Gaussian evolution-observation models. The terms "particle filter" and "sequential Monte Carlo" are sometimes used synonymously in the literature. However, in this thesis, we use the term PF-SMC as unifying terminology to describe both state and parameter estimation algorithms using sequential data and particle filtering methodology.

Like with EnKF schemes, PF-SMC methods avoid linearization of nonlinear operators by instead using Monte Carlo techniques to simulate distributions with random samples. The aim of PF-SMC algorithms for state estimation is to sequentially produce random samples

$$\mathcal{S}_k = \left\{x_k^1, x_k^2, \ldots, x_k^N\right\}, \quad x_k^n \in \mathbb{R}^d, \quad n = 1, 2, \ldots, N,$$

one distributed according to the conditional density $\pi(x_{k+1} \mid D_k)$ obtained after the time evolution update, referred to as the prediction sample, and the other distributed according to the conditional density $\pi(x_{k+1} \mid D_{k+1})$ obtained after the observation update, which we will call the posterior sample. In the PF-SMC literature, the state vectors x_k^n are called the *particles* of the sample. PF-SMC algorithms generally consist of two main steps:

(1) Given a sample distributed according to the current density $\pi(x_k \mid D_k)$, generate a prediction sample distributed according to the conditional density $\pi(x_{k+1} \mid D_k)$.

(2) Given a prediction sample distributed according to the conditional density $\pi(x_{k+1} \mid D_k)$, generate a posterior sample distributed according to the posterior density $\pi(x_{k+1} \mid D_{k+1})$.

These steps are performed by embedding Monte Carlo integration techniques into the time evolution and observation updating formulas (4.7) and (4.8) derived in Section 4.3.

More specifically, assume that we have a sample of particles drawn from the current state density, where each particle is assumed to be an equally likely realization of the random variable X_k. We denote this sample by

$$\mathcal{S}_k = \big\{(x_k^1, w_k^1), (x_k^2, w_k^2) \ldots, (x_k^N, w_k^N)\big\}, \quad x_k^n \in \mathbb{R}^d, \quad w_k^n \in [0,1], \quad n = 1, 2, \ldots, N,$$

where each particle x_k^n is coupled with its probability of being drawn, w_k^n, for $n = 1, 2, \ldots, N$. The w_k^n values are often referred to as the *weights* of the particles, similar to the terminology used in the context of quadrature rules. In this case, we set

$$w_k^n = \frac{1}{N},$$

such that all N particles have equal weights, i.e., if sampled from $\mathcal{S}_k$ with replacement, all N particles are equally likely to be drawn. The Monte Carlo approximation to the time evolution update formula (4.7) is then given by

$$\begin{aligned} \pi(x_{k+1} \mid D_k) &= \int \pi(x_{k+1} \mid x_k)\pi(x_k \mid D_k)dx_k \\ &\approx \sum_{n=1}^{N} \pi(x_{k+1} \mid x_k^n)w_k^n, \end{aligned}$$

where the approximation

$$\widehat{\pi}(x_{k+1} \mid D_k) = \sum_{n=1}^{N} \pi(x_{k+1} \mid x_k^n)w_k^n \tag{4.49}$$

is referred to as the empirical prediction density, following [58]. The density (4.49) can be interpreted as a weighted mixture of the individual densities $\pi(x_{k+1} \mid x_k^n)$, each dependent on a single particle x_k^n. The use of mixture distributions, particularly Gaussian mixture models, in the context of Bayesian filtering has been examined in the references [59, 60].

Substituting the empirical prediction density (4.49) into the observation update formula (4.8), we arrive at the following approximation to the posterior updating formula (4.9):

$$\begin{aligned} \pi(x_{k+1} \mid D_{k+1}) &\propto \pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid D_k) \\ &\approx \pi(y_{k+1} \mid x_{k+1})\widehat{\pi}(x_{k+1} \mid D_k) \\ &= \widehat{\pi}(x_{k+1} \mid D_{k+1}), \end{aligned}$$

where

$$\widehat{\pi}(x_{k+1} \mid D_{k+1}) = \pi(y_{k+1} \mid x_{k+1}) \sum_{n=1}^{N} \pi(x_{k+1} \mid x_k^n) w_k^n \tag{4.50}$$

is called the empirical filtering density. This formula is starting point of most PF-SMC algorithms.

A basic implementation of PF-SMC which makes use of sampling-importance-resampling (SIR) [127] is outlined in Algorithm 6; see also [61, 22, 5]. We briefly mentioned the SIR algorithm in Section 3.3 on Monte Carlo integration. Note that the prediction in Step 2 of Algorithm 6 is nothing more than a realization of the approximation

$$\int f(x_k)\pi(x_k \mid D_k)dx_k \approx \frac{1}{N}\sum_{n=1}^{N} f(x_k^n)$$

with $f(x_k) = \pi(x_{k+1} \mid x_k)$. In the initial sampling step, the SIR model draws from the prior without taking into account the observation y_{k+1}. Steps 3–5 are referred to as importance sampling, since the predictive sample particles are first blindly drawn, then assigned probabilities based upon the observation likelihood function.

The values w_{k+1}^n computed in Step 4 are the updated weights corresponding to the $\widetilde{x}_{k+1}^n$ particles after the data is taken into account. The process of drawing one particle from each of the individual densities $x_{k+1} \mapsto \pi(x_{k+1} \mid x_k^n)$, $n = 1, 2, \ldots, N$, in Step 2 is called layered sampling. Although not essential in the initial sampling step, as we could simply draw each new particle from the approximated density in Step 2, layered sampling is often preferable in terms of generating a predictive sample that better covers the support of the density.

Algorithm 6 SIR PF-SMC Algorithm

Given an initial distribution $\pi(x_0) = \pi(x_0 \mid D_0)$:

1. Draw a random sample of N particles,
$$\mathcal{S}_0 = \left\{x_0^1, x_0^2, \ldots, x_0^N\right\},$$
from $\pi(x_0)$. Set $k = 0$.

2. *Prediction step*: Approximate the time evolution update integral (4.7) via Monte Carlo integration,
$$\pi(x_{k+1} \mid D_k) \approx \frac{1}{N} \sum_{n=1}^{N} \pi(x_{k+1} \mid x_k^n).$$

3. *Sampling step*: Sample from the predicted density by drawing one new particle $\widetilde{x}_{k+1}^n$ from the individual density $\pi(x_{k+1} \mid x_k^n)$ for $n = 1, 2, \ldots, N$.

4. *Observation update*: Calculate the relative likelihoods
$$w_{k+1}^n = \frac{1}{W} \pi(y_{k+1} \mid \widetilde{x}_{k+1}^n), \quad W = \sum_{n=1}^{N} \pi(y_{k+1} \mid \widetilde{x}_{k+1}^n).$$

5. *Resampling step*: Generate a posterior sample $\mathcal{S}_{k+1}$ by drawing x_{k+1}^n from the predictive sample,
$$\widetilde{S}_k = \left\{\widetilde{x}_k^1, \widetilde{x}_k^2, \ldots, \widetilde{x}_k^N\right\},$$
for $n = 1, 2, \ldots, N$, where w_{k+1}^n is the probability of drawing the particle $\widetilde{x}_k^n$.

6. If $k < T$, set $k = k + 1$ and repeat from Step 2; otherwise, stop.

Practical considerations when implementing PF-SMC algorithms include selecting an appropriate sample size N such that the filter converges to the limiting distribution of the posterior reasonably, and to avoid *thinning* or *impoverishment* of the sample, which occurs when the weights w_{k+1}^n are unevenly distributed, thereby leading to multiple resampling of the same few prediction particles with large weight. The former can usually be assessed by monitoring the sample variance of the PF-SMC estimate as the algorithm proceeds; the latter tends to happen when the likelihood $\pi(y_{k+1} \mid x_{k+1})$ is very narrow or when the evolution model used does not match well the actual evolution of the system, causing the prediction sample to go astray, and has been considered extensively in the literature; we refer to, e.g., [22, 37, 128, 129, 130] and additional articles within [23].

A particularly effective method for combatting sample impoverishment is known as the auxiliary particle technique. As pointed out in [58], one weakness of PF-SMC algorithms is that in the presence of an outlier, the weights w_{k+1}^n become very unevenly distributed, and therefore a large number of sample points is needed for the draws to be even remotely close to a sample from the empirical filtering density (4.50). The auxiliary particle approach addresses this issue by sampling from the joint density $\pi(x_{k+1}, n \mid D_{k+1})$, where n is the index of the components of the mixture in the empirical prediction density (4.49). From the observation that

$$\pi(x_{k+1}, n \mid D_{k+1}) \propto \pi(y_{k+1} \mid x_{k+1})\pi(x_{k+1} \mid x_k^n)w_k^n, \qquad n = 1, \ldots, N, \tag{4.51}$$

it follows that drawing from the joint density (4.51) and discarding the index n produces the desired sample from the empirical filtering density (4.50). The index n is called an *auxiliary variable* because it is not of direct interest. Introducing the auxiliary variable in this manner effectually takes the likelihood into account before the sample is drawn.

Sampling methods such as SIR, rejection sampling, and MCMC can be used to efficiently sample the density $\pi(x_{k+1}, n \mid D_{k+1})$ in (4.51). For example, using SIR,

first M proposals

$$\left(x_{k+1}^j, n^j\right) \sim g(x_{k+1}, n \mid D_{k+1}), \qquad j = 1, \ldots, M,$$

are drawn, then they are assigned the new weights

$$w_{k+1}^j = \frac{1}{W} \frac{\pi(y_{k+1} \mid x_{k+1}^j)\pi(x_{k+1}^j \mid x_k^{n^j})}{g(x_{k+1}^j, n^j \mid D_{k+1})}, \qquad W = \sum_{j=1}^{M} w_{k+1}^j.$$

In the problem of drawing the M proposals, the proxy density g can be designed as needed to make the weights even, and it can depend on y_{k+1} and x_k^j, thus making this approach very adaptable and flexible. One convenient choice of g suggested in [58] is

$$g(x_{k+1}, n \mid D_{k+1}) \propto \pi(y_{k+1} \mid \mu_{k+1}^n)\pi(x_{k+1} \mid x_k^n),$$

where the auxiliary particle μ_{k+1}^n is the mean, the mode, a draw or some other likely value associated with the density $\pi(x_{k+1} \mid x_k^n)$. In the PF-SMC algorithms which will be derived in Chapter 5, we use the propagated value

$$\mu_{k+1}^n = F_{k+1}(x_k^n)$$

as the auxiliary predictor. This choice of g is made so that $g(n \mid D_{k+1}) \propto \pi(y_{k+1} \mid \mu_{k+1}^n)$, and therefore the new weights are of the form

$$w_{k+1}^j = \frac{1}{W} \frac{\pi(y_{k+1} \mid x_{k+1}^j)}{\pi(y_{k+1} \mid \mu_{k+1}^{n^j})}, \qquad W = \sum_{j=1}^{M} w_{k+1}^j.$$

Proposals can also be obtained by means of rejection sampling and MCMC in a similar fashion. Note that the sample drawn from the empirical filtering density can be of size M larger than N, then subsequently resampled to size N.

PF-SMC methods have been advocated in various applications, including mathematical finance, biomedical imaging, computational biology, and process tomography; see, e.g., [58, 37, 131, 132, 133, 6]. In many typical particle filter settings, the model consists of an SDE, whose solution is discretized and propagated in time by a simple

Euler(-Maruyama) scheme; see [134]. First-order explicit time integration methods are also used frequently for models which would admit other options. For example, in [6], where the primary equation describing the fluid flow is a classical advection-diffusion equation with poorly known coefficients and initial condition, stochastic noise is added to account for modeling errors. Since the MOL discretization of this problem transforms the underlying PDE into a system of stiff ODEs, in order to maintain stability, which is guaranteed by the Courant-Friedrichs-Lewy (CFL) condition [12], an excruciatingly small time step may be needed for the Euler method, thus increasing enormously the computation time.

Though many variants of the basic particle filtering algorithm exist, the most pertinent reference for this thesis is the algorithm proposed by Jane Liu and Mike West in [37], which introduces a method for sequentially estimating the model parameters of a system along with the system states. It is the parameter estimation capability which clearly distinguishes this PF-SMC variant from most Kalman-type filtering algorithms, which traditionally focus on state estimation only and assume fixed model parameters. The algorithm for combined parameter and state estimation derived in [37] extends the standard Bayesian filtering approach for state estimation by including, along with the state particles and their corresponding weights, a sample cloud of parameters that is artificially evolved over time.

More specifically, consider the evolution-observation model

$$X_{k+1} = F_{k+1}(X_k, W_{k+1}, \theta), \quad k = 0, 1, 2, \ldots, \tag{4.52}$$

$$Y_k = G_k(X_k, V_k, \theta), \quad k = 1, 2, \ldots, \tag{4.53}$$

where the functions F_{k+1} and G_k are assumed to be known, and the state noise W_{k+1} and observation noise V_k are mutually independent. Model (4.52)–(4.53) is similar to (4.5)–(4.6), except that now the dependence of the model on the unknown parameters θ is made clear.

Assume that we have a combined sample

$$\mathcal{S}_k = \left\{(x_k^1, \theta_k^1, w_k^1), (x_k^2, \theta_k^2, w_k^2) \ldots, (x_k^N, \theta_k^N, w_k^N)\right\},$$

with

$$x_k^n \in \mathbb{R}^d, \qquad \theta_k^n \in \mathbb{R}^q, \qquad w_k^n \in [0,1], \qquad n = 1, 2, \ldots, N,$$

drawn from the current joint density $\pi(x_k, \theta \mid D_k)$, where each particle x_k^n is associated with a set of model parameters θ_k^n as well as a corresponding weight w_k^n. We remark that the time index k on the θ samples indicates that these parameters are from the same posterior density as their corresponding particles, not that θ is time-varying. The updating formula for generating a sample from the posterior density $\pi(x_{k+1}, \theta \mid D_{k+1})$ is derived similarly to (4.9), that is,

$$\begin{aligned} \pi(x_{k+1}, \theta \mid D_{k+1}) &\propto \pi(y_{k+1} \mid x_{k+1}, \theta)\pi(x_{k+1}, \theta \mid D_k) \\ &\propto \pi(y_{k+1} \mid x_{k+1}, \theta)\pi(x_{k+1} \mid \theta, D_k)\pi(\theta \mid D_k), \end{aligned}$$

where the form of the last equation highlights the importance of the density $\pi(\theta \mid D_k)$ in the updating of the posterior. If θ is known and therefore fixed, it can be dropped from the above equations, leading to the usual updating formula for state estimation, which in [37] uses auxiliary particles [58]. In the case when θ is not known, an approximation of the density $\pi(\theta \mid D_k)$ is also needed.

A method for approximating $\pi(\theta \mid D_k)$ known as artificial evolution treats the model parameters as if they were time-evolving, thereby replacing θ by θ_k at time k and including θ_k in an augmented state vector. At each time, the parameter vector is then evolved according to the rule

$$\theta_{k+1} = \theta_k + \xi_{k+1}, \qquad \xi_{k+1} \sim \mathcal{N}(0, \mathsf{W}_{k+1})$$

for some specified covariance matrix W_{k+1}, where θ_k and ξ_{k+1} are conditionally independent given D_k. This idea is based on the roughening penalties approach for state particles suggested in [61]. While this treatment of the parameters provides a way to generate a new cloud of parameters at each time instance, thus avoiding the sample impoverishment issues corresponding to fixed parameter values, the artificial evolution process implies a loss of information between time steps, resulting in posterior distributions that are far from the actual posteriors.

An alternative approach proposed in [37] modifies the artificial evolution method by implementing a kernel smoothing approach as in [60], extending the mixture modeling techniques analyzed in [59] in the following manner. Suppose that at time k we have current parameter samples θ_k^n and corresponding weights w_k^n, $n = 1, 2, \ldots, N$, which provide an approximation of the density $\pi(\theta \mid D_k)$ in sampled form, where the subscript k in θ_k simply relates it to the posterior corresponding to the kth observation and does not imply time variance of the parameters. Denoting by $\overline{\theta}_k$ and C_k the mean and covariance of the parameter sample,

$$\overline{\theta}_k = \sum_{n=1}^{N} w_k^n \theta_k^n, \qquad \mathsf{C}_k = \sum_{n=1}^{N} w_k^n \big(\theta_k^n - \overline{\theta}_k\big)\big(\theta_k^n - \overline{\theta}_k\big)^{\mathsf{T}},$$

the smooth kernel density advocated in [59, 60] is given by

$$\pi(\theta \mid D_k) \approx \sum_{n=1}^{N} w_k^n \mathcal{N}(\theta \mid \overline{\theta}_k^n, s^2 \mathsf{C}_k), \tag{4.54}$$

where the kernel locations $\overline{\theta}_k^n$ are determined from the sampled parameters via the shrinkage rule

$$\overline{\theta}_k^n = a\theta_k^n + (1-a)\overline{\theta}_k.$$

The factor a is chosen such that $0 < a < 1$ and $a^2 + s^2 = 1$, which guarantees that the Gaussian mixture in (4.54) has the same mean and covariance as the parameter sample, thereby avoiding the over-widening of the variance which occurs when no shrinkage is performed. Combining this approach with artificial evolution yields the conditional parameter evolution density,

$$\pi(\theta_{k+1} \mid \theta_k) \sim \mathcal{N}(\theta_{k+1} \mid a\theta_k^n + (1-a)\overline{\theta}_k, s^2 \mathsf{C}_k), \qquad s^2 = 1 - a^2,$$

and an updating formula for $\pi(\theta_{k+1} \mid D_k)$ of the form (4.54). The combined parameter and state estimation PF-SMC technique derived in [37] will provide the foundation for the algorithm developed in the following chapter.

Chapter 5

LMM Particle Filter Sequential Monte Carlo

In this chapter we detail a novel contribution to the particle filtering methodology for deterministic dynamical systems, where we propose a systematic method for defining the innovation term in the time evolution update of PF-SMC algorithms based on an estimate of the approximation errors in numerical propagation. More precisely, we propose to carry out the time integration in the evolution step using the LMM numerical solvers described in Section 2.2. The choice of LMMs in this context is motivated by the fact that their stability properties are well-known, and good estimates for the accumulating discretization error exist, thereby providing a basis for estimating rigorously the innovation variance.

In the proposed approach, the innovation variance estimation is related naturally to classical error analysis by predictor-corrector methods for numerical ODE solvers, in particular, the HOMEC and Milne device error estimates described in Section 2.3. Ultimately, a good control of the innovation variance is expected to reduce the posterior variance of the state and parameter estimation. Moreover, by controlling the complexity of the propagation step, unexpected slow-down due to difficulty in satisfying a preassigned integration accuracy is avoided: While poor parameter proposals may lead to decreased accuracy of the solution, this is accounted for by increasing

the innovation variance, without slowing down the computations. The selection of the LMM to be used for the time evolution step depends on the characteristics of the system, the AM and BDF families being the methods of choice for stiff problems. Additionally, as the time propagation for each particle entails exactly the same mutually independent computational steps, the algorithm is particularly well-suited for parallelization, making particle filters and SMC methods competitive for parameter estimation problems. Computationally efficient implementations of the algorithm for parallel and vectorized computing environments will be discussed in Chapter 6.

This chapter is organized as follows. In Section 5.1, we show that discrete-time propagation with LMMs in the Bayesian filtering framework retains the Markov properties necessary for the updating formulas derived in Section 4.3 to hold. We describe how to assign the variance of the innovation term in propagation based on LMM error control methods in Section 5.2. The LMM PF-SMC algorithms for nonlinear filtering and parameter estimation are derived in Section 5.3, and the effectiveness of the algorithm with respect to parameter estimation is demonstrated in Section 5.4, where the method is applied to a model problem with some of the characteristics of the dynamical systems encountered in metabolic models. The LMM PF-SMC algorithm and related results have been published in the article [10].

5.1 Discrete-time propagation

We begin by reformulating the parameter estimation problem defined in Section 4.1 within the framework of discrete-time propagation. Consider the following IVP over the time interval $[t_j, t_{j+1}]$,

$$\frac{du}{dt} = f(t, u, \theta), \quad t_j < t < t_{j+1}, \quad u(t_j) = u_j,$$

and assume that the exact solution can be written using the formal exact propagation operator ψ^{exact},

$$u(t_{j+1}) = \psi^{\text{exact}}(t_{j+1}, u_j, \theta).$$

In many applications, however, ψ^{exact} is not accessible and is replaced by a numerical scheme: We define a grid function associated with an explicit r-step numerical solver ψ by assuming that its value at $t = t_{j+1}$ is

$$u_{j+1} = \psi(u_j, u_{j-1}, \ldots, u_{j-r+1}, \theta, h),$$

where $h > 0$ is the constant time step, and, for the time being, the values u_j with $j < 0$ are assumed to be known. This formula, which is exact for the grid function u_n, when applied to the discretized values of the true solution of the differential equation, must be modified by adding an approximation error, or local truncation error, yielding

$$u(t_{j+1}) = \psi(u(t_j), \ldots, u(t_{j-r+1}), h) + w_{j+1}.$$

If we model the solution $u(t_j)$ at the specified times as a random variable U_j, this formula defines an r-Markov model

$$U_{j+1} = \psi(U_j, \ldots, U_{j-r+1}, \theta, h) + W_{j+1} \tag{5.1}$$

for the stochastic process $\{U_j\}_{j=0}^{T}$, where $\{W_j\}_{j=1}^{T}$ is the associated innovation process.

The r-Markov model (5.1) can be reformulated in vectorial 1-Markov form by introducing the multivariate random variables

$$X_j = \begin{bmatrix} U_j \\ \vdots \\ U_{j-r+1} \end{bmatrix}, \qquad V_{j+1} = \begin{bmatrix} W_{j+1} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

and defining

$$\Psi(X_j, \theta, h) = \begin{bmatrix} \psi(X_j, \theta, h) \\ U_j \\ \vdots \\ U_{j-r+2} \end{bmatrix}.$$

After this change of variables, the stochastic process $\{X_j\}_{j=0}^T$ obeys the 1-Markov model

$$X_{j+1} = \Psi(X_j, \theta, h) + V_{j+1}. \tag{5.2}$$

In the case where ψ is an implicit method, in order to apply the 1-Markov formula (5.2) we must call upon the implicit function theorem.

If $\{Y_j\}_{j=1}^T$ is the stochastic process modeling the observations of U_j, then

$$Y_j = g(U_j) + E_j = G(X_j) + E_j, \tag{5.3}$$

where g is a nonlinear function relating the observations and the state variable, and $\{E_j\}_{j=1}^T$ represents the measurement noise, with the convention that if no measurement is done at $t = t_j$, the variable Y_j is interpreted as an empty measurement, $Y_j = \emptyset$.

Equations (5.2)–(5.3) constitute the discrete-time evolution-observation model on which our state and parameter estimation algorithm is based. Unlike in the standard filtering literature, the innovation V_{j+1} in (5.2) represents the numerical approximation error due to the propagation scheme ψ, therefore it is reasonable to base the assignment of its covariance on an estimate of the error introduced by the numerical solver. Observe that the error typically depends on the parameter θ as well. In the following section, we describe a natural way to base the covariance of V_{j+1} on classical time integration error estimates when using LMMs in the propagation step.

5.2 LMM error-based innovation covariance

Given the discrete-time propagation model (5.2), we want to derive a systematic procedure for assigning the covariance of the innovation term V_{j+1} in the particle filter from local truncation error estimates. In particular, we will restrict our attention to the cases where the error is estimated with the HOMEC and Milne device methods detailed in Section 2.3.

More precisely, if the local truncation error is estimated with the HOMEC method, we model the innovation as

$$V_{j+1} \sim \mathcal{N}(0, \Gamma_{j+1}),$$

where the entries of the covariance matrix Γ_{j+1},

$$\Gamma_{j+1} = \mathrm{diag}(\gamma), \quad \gamma_i = \tau^2 (u_{j+1} - \widehat{u}_{j+1})_i^2, \quad i = 1, \ldots, d,$$

follow from the error estimate (2.22), and the factor $\tau > 1$ is introduced to compensate for the omission of the higher order terms. In the computed examples presented in Section 5.4, we use $\tau = 1.2$. This definition of Γ_{j+1} is based on the assumption that the estimated error determines the standard deviation of the innovation. We remark that while the innovation is not independent of X_j and the parameter θ, this assignment preserves the Markov property of the model.

The innovation covariance model based on the Milne device estimate (2.27) is constructed in a similar fashion, with

$$\Gamma_{j+1} = \mathrm{diag}(\gamma), \quad \gamma_i = \tau^2 \kappa^2 (u_{j+1} - \widetilde{u}_{j+1})_i^2, \quad i = 1, \ldots, d,$$

where the constant κ, defined in (2.28), is computed using the error constants of the pair of selected LMMs, listed in Tables 2.1, 2.2 and 2.3 in Chapter 2.

5.3 The LMM PF-SMC algorithms for filtering and parameter estimation

In this section we briefly refresh the basic principles of particle filters, including the main ideas of SMC and Bayesian filtering presented in Chapter 4, and we describe how the LMM-based propagation approach is integrated into this framework. In the previous sections, we showed that the LMM gives us an evolution-observation model

$$X_{j+1} = \Psi(X_j, \theta, h) + V_{j+1}, \qquad V_{j+1} \sim \mathcal{N}(0, \Gamma_{j+1}(X_j, \theta)), \tag{5.4}$$

$$Y_j = G(X_j) + E_j, \qquad E_j \sim \mathcal{N}(0, \Sigma_j), \tag{5.5}$$

where the innovation covariance $\Gamma_{j+1}(X_j, \theta)$ is estimated using the error control of the LMM scheme determining the propagator Ψ. Here we proceed to design a particle filtering algorithm that allows us to update the joint conditional probability densities of the states and parameters,

$$\pi(x_j, \theta \mid D_j) \;\longrightarrow\; \pi(x_{j+1}, \theta \mid D_{j+1}), \tag{5.6}$$

in sample form. Given that the pairs (x_j^n, θ_j^n), $n = 1, \ldots, N$, have been drawn from the posterior density $\pi(x_j, \theta \mid D_j)$ with relative probabilities w_j^n, we want to effectuate the sample update

$$\mathcal{S}_j \;\longrightarrow\; \mathcal{S}_{j+1}, \qquad \mathcal{S}_j = \left\{(x_j^n, \theta_j^n, w_j^n)\right\}_{n=1}^N, \tag{5.7}$$

thereby obtaining a sample $\mathcal{S}_{j+1}$ from $\pi(x_{j+1}, \theta \mid D_{j+1})$.

For simplicity, we assume first that the parameter θ is known and can therefore be dropped from the notation in (5.6). Because the model (5.4)–(5.5) is such that

(a) The state X_{j+1} depends on the past data D_j only through the previous state X_j, that is,

$$\pi(x_{j+1} \mid x_j, D_j) = \pi(x_{j+1} \mid x_j); \tag{5.8}$$

(b) The observation Y_{j+1} depends on the past only through the current state X_{j+1}, that is,

$$\pi(y_{j+1} \mid x_{j+1}, D_j) = \pi(y_{j+1} \mid x_{j+1}), \tag{5.9}$$

the updating of the posterior density can be carried out in two steps,

$$\pi(x_j \mid D_j) \;\longrightarrow\; \pi(x_{j+1} \mid D_j) \longrightarrow \pi(x_{j+1} \mid D_{j+1}),$$

the first step based on the evolution equation (5.4) and the second step based on the observation model (5.5).

The prediction step can be formulated in terms of the Chapman-Kolmogorov formula,

$$\begin{aligned} \pi(x_{j+1} \mid D_j) &= \int \pi(x_{j+1} \mid x_j, D_j)\pi(x_j \mid D_j)dx_j \\ &= \int \pi(x_{j+1} \mid x_j)\pi(x_j \mid D_j)dx_j, \end{aligned} \tag{5.10}$$

where the last expression follows from property (5.8). The second step reduces to an application of Bayes' theorem conditioned on D_j, followed by an application of identity (5.9):

$$\begin{aligned} \pi(x_{j+1} \mid D_{j+1}) &= \pi(x_{j+1} \mid y_{j+1}, D_j) \\ &\propto \pi(y_{j+1} \mid x_{j+1}, D_j)\pi(x_{j+1} \mid D_j) \\ &= \pi(y_{j+1} \mid x_{j+1})\pi(x_{j+1} \mid D_j), \end{aligned} \tag{5.11}$$

Combining (5.10) and (5.11), we arrive at the updating formula

$$\pi(x_{j+1} \mid D_{j+1}) \propto \pi(y_{j+1} \mid x_{j+1}) \int \pi(x_{j+1} \mid x_j)\pi(x_j \mid D_j)dx_j, \tag{5.12}$$

which is equal to equation (4.9) in Section 4.3.

Assuming that we have a sample $\mathcal{S}_j$ as in (5.7) drawn from the posterior probability density $\pi(x_j \mid D_j)$ with fixed parameter θ, and using a Monte Carlo approximation of the integral in (5.12), we arrive at the formula

$$\pi(x_{j+1} \mid D_{j+1}) \propto \pi(y_{j+1} \mid x_{j+1}) \sum_{n=1}^{N} w_j^n \pi(x_{j+1} \mid x_j^n), \tag{5.13}$$

which serves as the starting point in most particle filter algorithms.

In the spirit of auxiliary particle methods, let $\overline{x}_{j+1}^n$ denote the expectation of X_{j+1} conditioned on $X_j = x_j^n$. This auxiliary particle is a predictor of the value of X_{j+1} given the initial value x_j^n, and in view of the model (5.4),

$$\overline{x}_{j+1}^n = \Psi(x_j^n, h).$$

We write equation (5.13) as

$$\pi(x_{j+1} \mid D_{j+1}) \propto \sum_{n=1}^{N} \underbrace{w_j^n \, \pi(y_{j+1} \mid \overline{x}_{j+1}^n)}_{=g_{j+1}^n} \frac{\pi(y_{j+1} \mid x_{j+1})}{\pi(y_{j+1} \mid \overline{x}_{j+1}^n)} \pi(x_{j+1} \mid x_j^n), \tag{5.14}$$

where the factor g_{j+1}^n, which we will refer to as the *fitness* of the nth predictor, measures how well the predictor agrees with the new data. Equation (5.14) can be interpreted as a mixture model [59, 60], and we can draw the sample $\mathcal{S}_{j+1}$ from it using the LMM particle filtering procedure outlined in Algorithm 7, where we set $\pi(x \mid y_{j+1}) = 1$ if $Y_{j+1} = \emptyset$, i.e., if at the end of the time step no new data arrives. Observe that if $Y_{j+1} = \emptyset$, the resampling is reduced to a reweighting of the particles, so that at the end of Step 5 we have $w_{j+1}^n = 1/N$ for all n.

Next we explain how the procedure can be modified to combine parameter and state estimation, as in [37]. When the differential equation depends on the unknown parameter vector θ, we must change (5.12) to

$$\begin{aligned} \pi(x_{j+1}, \theta \mid D_{j+1}) \quad &\propto \quad \pi(y_{j+1} \mid x_{j+1}, \theta) \\ &\qquad \times \int \pi(x_{j+1} \mid x_j, \theta) \pi(x_j \mid \theta, D_j) \pi(\theta \mid D_j) dx_j, \end{aligned} \tag{5.15}$$

to account for the fact that θ is unknown and thus modeled as a random variable. Furthermore, assume that a sample (5.7) has been drawn from the conditional density $\pi(x_j, \theta \mid D_j)$, and let $\overline{\theta}_j$ and C_j denote the mean and covariance of the parameter particles,

$$\overline{\theta}_j = \sum_{n=1}^{N} w_j^n \theta_j^n, \qquad \mathsf{C}_j = \sum_{n=1}^{N} w_j^n \big(\theta_j^n - \overline{\theta}_j\big)\big(\theta_j^n - \overline{\theta}_j\big)^{\mathsf{T}}.$$

We approximate the marginal probability density $\pi(\theta \mid D_j)$ of θ by a Gaussian mixture model,

$$\pi(\theta \mid D_j) \approx \sum_{n=1}^{N} w_j^n \mathcal{N}\big(\theta \mid \overline{\theta}_j^n, s^2 \mathsf{C}_j\big)$$

Algorithm 7 LMM PF-SMC for State Estimation

Given the initial probability density $\pi_0(x_0)$:

1. *Initialization:* Draw the particle ensemble from $\pi_0(x_0)$:

$$\mathcal{S}_0 = \big\{(x_0^1, w_0^1), (x_0^2, w_0^2), \ldots, (x_0^N, w_0^N)\big\}, \quad w_0^1 = w_0^2 = \ldots = w_0^N = \frac{1}{N}.$$

   Set $j = 0$.

2. *Propagation:* Compute the predictor using LMM:

$$\overline{x}_{j+1}^n = \Psi(x_j^n, h), \quad 1 \le n \le N.$$

3. *Survival of the fittest:* For each n:

   (a) Compute the fitness weights

$$g_{j+1}^n = w_j^n \, \pi(y_{j+1} \mid \overline{x}_{j+1}^n), \quad g_{j+1}^n \leftarrow \frac{g_{j+1}^n}{\sum_n g_{j+1}^n};$$

   (b) Draw indices with replacement $\ell_n \in \big\{1, 2, \ldots, N\big\}$ using probabilities $\mathsf{P}\{\ell_n = k\} = g_{j+1}^k$;

   (c) Reshuffle

$$x_j^n \leftarrow x_j^{\ell_n}, \quad \overline{x}_{j+1}^n \leftarrow \overline{x}_{j+1}^{\ell_n}, \quad 1 \le n \le N.$$

4. *Innovation:* For each n:

   (a) Using LMM error control, estimate $\Gamma_{j+1}^n = \Gamma_{j+1}(x_j^n)$;

   (b) Draw $v_{j+1}^n \sim \mathcal{N}(0, \Gamma_{j+1}^n)$;

   (c) Proliferate

$$x_{j+1}^n = \overline{x}_{j+1}^n + v_{j+1}^n.$$

5. *Weight updating:* For each n, compute

$$w_{j+1}^n = \frac{\pi(y_{j+1} \mid x_{j+1}^n)}{\pi(y_{j+1} \mid \overline{x}_{j+1}^n)}, \quad w_{j+1}^n \leftarrow \frac{w_{j+1}^n}{\sum_n w_{j+1}^n}.$$

6. If $j < T$, set $j = j + 1$ and repeat from Step 2.

with

$$\overline{\theta}_j^n = a\theta_j^n + (1-a)\overline{\theta}_j,$$

where a is a shrinkage factor, $0 < a < 1$, and $s^2 = 1 - a^2$. It is a straightforward matter to verify that the mean and variance of the mixture model coincide with the empirical mean and variance of the sample of parameter vectors. Implicitly, here we assume that the particle θ_j^n was drawn from the nth Gaussian density in the mixture, therefore justifying the discrete approximation of (5.15) by the sum

$$\pi(x_{j+1}, \theta \mid D_{j+1}) \propto \sum_{n=1}^{N} w_j^n \pi(y_{j+1} \mid x_{j+1}, \theta)\pi(x_{j+1} \mid x_j^n, \theta)\mathcal{N}(\theta \mid \overline{\theta}_j^n, s^2\mathsf{C}_j),$$

which we write as

$$\begin{aligned}\pi(x_{j+1}, \theta \mid D_{j+1}) \;\propto\; & \sum_{n=1}^{N} \underbrace{w_j^n \pi(y_{j+1} \mid \overline{x}_{j+1}^n, \overline{\theta}_j^n)}_{=g_{j+1}^n} \\ & \times \frac{\pi(y_{j+1} \mid x_{j+1}, \theta)}{\pi(y_{j+1} \mid \overline{x}_{j+1}^n, \overline{\theta}_j^n)} \pi(x_{j+1} \mid x_j^n, \theta)\mathcal{N}(\theta \mid \overline{\theta}_j^n, s^2\mathsf{C}_j), \quad (5.16)\end{aligned}$$

where the coefficient g_{j+1}^n is the fitness of the predictor

$$\left(\overline{x}_{j+1}^n, \overline{\theta}_j^n\right) = \left(\Psi(x_{j+1}^n, \overline{\theta}_j^n, h), \overline{\theta}_j^n\right).$$

The approximation (5.16) is the basis for the LMM PF-SMC algorithm for combined parameter and state estimation presented in Algorithm 8. We remark that, while it is suggested in [37] that the value θ_j^n is used in the propagation step, following [131] we perform the contraction $\theta_j^n \to \overline{\theta}_j^n$ before the propagation.

The choice of the value of the shrinkage factor a follows from [59, 60]. To understand the role of this parameter, we consider the extreme cases: If $a = 1$ and thus $s = 0$, the auxiliary particles $\overline{\theta}_j^n$ coincide with the particles θ_j^n, and we have an SIR algorithm, which is known to suffer from sample impoverishment, or data thinning, as discussed in Section 4.5. At the other extreme, $a = 0$ leads to a Gaussian

Algorithm 8 LMM PF-SMC for Combined Parameter and State Estimation

Given the initial probability density $\pi_0(x_0, \theta)$:

1. *Initialization:* Draw the particle ensemble from $\pi_0(x_0, \theta)$:

$$\mathcal{S}_0 = \big\{(x_0^1, \theta_0^1, w_0^1), (x_0^2, \theta_0^2, w_0^2), \ldots, (x_0^N, \theta_0^N, w_0^N)\big\},$$
$$w_0^1 = w_0^2 = \ldots = w_0^N = \frac{1}{N}.$$

   Compute the parameter mean and covariance:

$$\overline{\theta}_0 = \sum_{n=1}^{N} w_0^n \theta_0^n, \quad \mathsf{C}_0 = \sum_{n=1}^{N} w_0^n \big(\theta_0^n - \overline{\theta}_0\big)\big(\theta_0^n - \overline{\theta}_0\big)^{\mathsf{T}}.$$

   Set $j = 0$.

2. *Propagation:* Shrink the parameters

$$\overline{\theta}_j^n = a\theta_j^n + (1-a)\overline{\theta}_j, \quad 1 \le n \le N,$$

   by a factor $0 < a < 1$. Compute the state predictor using LMM:

$$\overline{x}_{j+1}^n = \Psi(x_j^n, \overline{\theta}_j^n, h), \quad 1 \le n \le N.$$

3. *Survival of the fittest:* For each n:
   (a) Compute the fitness weights

$$g_{j+1}^n = w_j^n \pi(y_{j+1} \mid \overline{x}_{j+1}^n, \overline{\theta}_j^n), \quad g_{j+1}^n \leftarrow \frac{g_{j+1}^n}{\sum_n g_{j+1}^n};$$

   (b) Draw indices with replacement $\ell_n \in \big\{1, 2, \ldots, N\big\}$ using probabilities $\mathsf{P}\{\ell_n = k\} = g_{j+1}^k$;
   (c) Reshuffle

$$x_j^n \leftarrow x_j^{\ell_n}, \quad \big(\overline{x}_{j+1}^n, \overline{\theta}_j^n\big) \leftarrow \big(\overline{x}_{j+1}^{\ell_n}, \overline{\theta}_j^{\ell_n}\big), \quad 1 \le n \le N.$$

(Continued on the next page)

Algorithm 8 LMM PF-SMC for Combined Parameter and State Estimation (Cont.)

4. *Proliferation:* For each n:
   (a) Proliferate the parameter by drawing
   $$\theta_{j+1}^n \sim \mathcal{N}\big(\overline{\theta}_j^n, s^2 \mathsf{C}_j\big), \quad s^2 = 1 - a^2;$$
   (b) Using LMM error control, estimate $\mathsf{\Gamma}_{j+1}^n = \mathsf{\Gamma}_{j+1}(x_j^n, \theta_{j+1}^n)$;
   (c) Draw $v_{j+1}^n \sim \mathcal{N}(0, \mathsf{\Gamma}_{j+1}^n)$;
   (d) Repropagate using LMM and add innovation:
   $$x_{j+1}^n = \Psi(x_j^n, \theta_{j+1}^n, h) + v_{j+1}^n.$$
5. *Weight updating:* For each n, compute
   $$w_{j+1}^n = \frac{\pi(y_{j+1} \mid x_{j+1}^n, \theta_{j+1}^n)}{\pi(y_{j+1} \mid \overline{x}_{j+1}^n, \overline{\theta}_j^n)}, \quad w_{j+1}^n \leftarrow \frac{w_{j+1}^n}{\sum_n w_{j+1}^n}.$$
6. If $j < T$, update
   $$\overline{\theta}_{j+1} = \sum_{n=1}^N w_{j+1}^n \theta_{j+1}^n, \quad \mathsf{C}_{j+1} = \sum_{n=1}^N w_{j+1}^n \big(\theta_{j+1}^n - \overline{\theta}_{j+1}\big)\big(\theta_{j+1}^n - \overline{\theta}_{j+1}\big)^{\mathsf{T}},$$
   set $j = j + 1$ and repeat from Step 2.

approximation of the particle density. In our computed examples, we use a value of a close to 1, namely, $a \approx 0.97$.

For the smoothing problem, we estimate the probability density of initial values that leads to realizations that are fit enough to survive. Therefore, we set

$$\theta_0^n = x_0^n,$$

or, more generally,

$$\theta_0^n = \widehat{T}(x_0^n),$$

where $\widehat{T} : \mathbb{R}^k \to \mathbb{R}^k$, $k = d$, is an appropriate one-to-one transformation and the particles θ_j^n are not initial values per se, but their ensemble represents the distribution of successful initial values. In the present framework, the solution to the smoothing problem is a hybrid between Algorithms 7 and 8 in the sense that since the propagation after the first time step does not depend explicitly on θ, the repropagation in Step 4(d) can be skipped, as the propagated value before adding innovation coincides with the predictor.

5.4 Computed examples

To illustrate the effectiveness of the LMM PF-SMC method outlined in Algorithm 8 for parameter estimation, we consider the following system of ODEs, which may be thought of as the governing equations for a compartment model comprising three biochemical species, S_1, S_2 and S_3, whose concentrations x_1, x_2 and x_3 are changed in response to enzymatic reactions:

$$\begin{aligned} \frac{dx_1}{dt} &= \Phi(t) - V_1 \frac{x_1}{x_1 + k_1} \\ \frac{dx_2}{dt} &= V_1 \frac{x_1}{x_1 + k_1} - V_2 \frac{x_2}{x_2 + k_2} \\ \frac{dx_3}{dt} &= V_2 \frac{x_2}{x_2 + k_2} - \lambda(x_3 - c_0) \end{aligned} \tag{5.17}$$

Parameter	V_1	V_2	T_1	T_2	k_1	k_2	λ	c_0	τ	A	A_0
Value	1.0	0.5	0.1	1.0	0.1	0.5	1.5	0.5	0.5	e/τ	0.1

Table 5.1: True parameter values used in generating the data. The units are arbitrary.

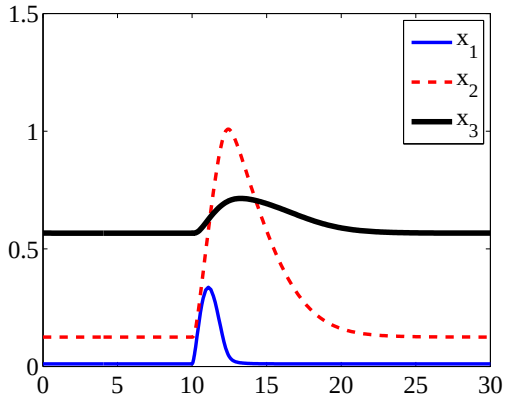

Figure 5.1: Reference solution curves for components x_1, x_2 and x_3 of system (5.17).

Here the parameters λ and c_0 are assumed to be known, as is the input function

$$\Phi(t) = A_0 + A(t - t_0)\exp(-(t - t_0)/\tau)$$

and its parameters A_0, A, t_0 and τ. The input function Φ can be thought of as a bolus of substance S_1, while the last term in the third equation models the passive diffusion of substance S_3 out of the system. The variables V_j and k_j, $j = 1, 2$, which are the maximum reaction rates and affinity constants, respectively, of a Michaelis-Menten reaction flux model are the unknown variables of interest. The ratios $T_j = k_j/V_j$, $j = 1, 2$, represent the time constants of the reactions. We refer to x_1, x_2 and x_3 as the states or components of the system.

To generate the data we solve the system (5.17) over the time interval $[0, 30]$ using `ode45` with default settings, fixing the parameters to the values listed in Table 5.1; the solution curves for components x_1, x_2 and x_3 are shown in Figure 5.1. More precisely, the data consist of noisy measurements at a few time instances of either all or some of the states, corrupted by additive zero mean Gaussian noise. We also assume that

at the start of the experiment the system is at steady state, and therefore the initial history needed as input for the higher order LMM solvers is constant.

The problem considered here is to estimate sequentially the unknown parameters V_1, V_2, k_1 and k_2 given observations of the states of the system and some *a priori* bounds for the parameters. The *a priori* probability density for the four unknown model parameters specifies their upper and lower bounds. We choose 0.5 times the true parameter values given in Table 5.1 as the lower bounds and 1.5 times the true parameter values as the upper bounds. To better accommodate the prior information with the Gaussian mixture model used in the sequential algorithm, we apply a logit transformation of the parameters: Using the notation $\kappa = (V_1, V_2, k_1, k_2)$ with $\kappa_{i,\min} \leq \kappa_i \leq \kappa_{i,\max}$, we define the model parameter vector $\theta \in \mathbb{R}^4$ with components

$$\theta_i = \log\left(\frac{s_i}{1 - s_i}\right),$$

where

$$\kappa_i = (1 - s_i)\kappa_{i,\min} + s_i\kappa_{i,\max}, \quad i = 1, \ldots, 4.$$

The initial sample $\{\theta_0^1, \ldots, \theta_0^N\}$ is drawn from a zero mean normal distribution with the variance adjusted so that for each θ_i, the corresponding s_i is approximately uniformly distributed over $[0, 1]$.

We use the third-order method BDF3 for propagation and fourth-order method BDF4 for error control, selecting the innovation variance based on the HOMEC technique, while noting that for all of the examples that follow, similar results are obtained if we instead use the third-order method AM2 for error control and select the innovation variance based on the Milne device technique. Since the BDF solvers are implicit, a system of nonlinear equations has to be solved at each time step in order to compute the solution: this is done with five iterates of Newton's method for nonlinear systems. In all computed examples we propagate $N = 10,000$ particles with fixed time step $h = 0.05$, and we begin propagating at time $t = 5$.

In the first computed example, we take 100 measurements of each of the three states, x_1, x_2 and x_3, and we add zero mean Gaussian noise with standard deviation

0.01. Figures 5.2 and 5.3 show the time series estimates and resulting smooth histograms of the parameters V_1, V_2, k_1 and k_2, respectively.

Since the data do not contain much information about the dynamics of the system over the time interval $[5, 10)$, the parameter clouds shown in Figure 5.2 remain widespread, suggesting that many parameter values could work. However, starting around time $t = 10$, the data starts to contain information regarding the underlying dynamics of the system, and the filter is able to learn through these dynamics and significantly narrow the pool of potential parameter values. As the dynamics begin to slow around time $t = 20$, the posterior uncertainty stays about the same.

We expect that the measurement gives information about the time constants of the two reactions. As the time constant T_j is related to parameters V_j and k_j via the formula $T_j = k_j/V_j$, we expect a positive correlation between the estimated values of V_j and k_j. Conversely, the joint probability density of V_j and k_j can be used to estimate a distribution for T_j. The scatter plots in Figure 5.4 display the correlation between V_1 and k_1 and the correlation between V_2 and k_2, respectively, along with the estimated distributions for T_1 and T_2. We see that, as expected, V_j and k_j are strongly positively correlated for $j = 1, 2$, and we are able to get accurate estimates of the time constants T_j, $j = 1, 2$.

Since the loss of particles can be a major problem when dealing with particle filters, we monitor how many particles we retain at each measurement by computing the *particle retention rate* (PRR), which is the percentage of predictor particles deemed fit enough to proliferate; i.e., the number of fit particles out of the total number of particles. Figure 5.5 displays the PRR at the arrival of each datum. Note that around the 22nd data arrival, which occurs at time $t = 10.5$, there is a significant drop in particle retention, suggesting that the model learns something new at this point and therefore rejects a higher number of particles.

In the second computed example, we decrease the sampling frequency by a factor of two and take 50 measurements of each of the three states. The time series estimates and resulting smooth histograms of the parameters are shown in Figures 5.6 and 5.7,

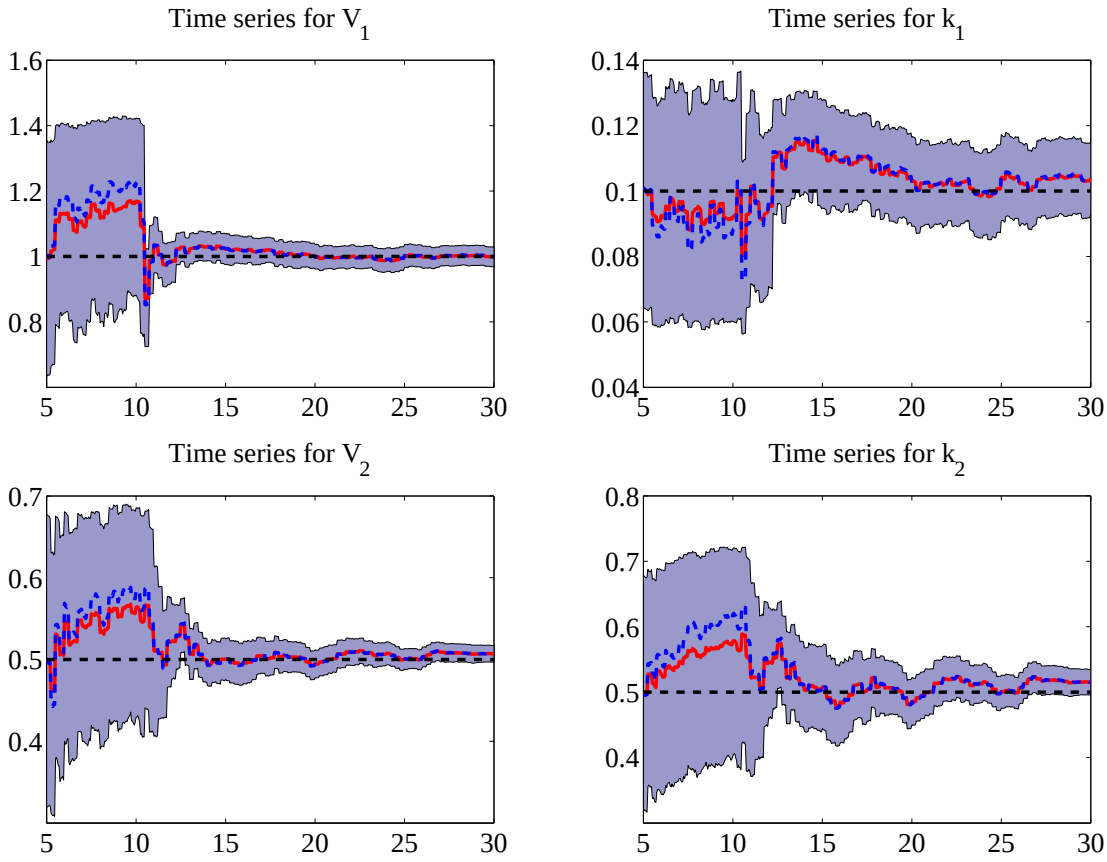

Figure 5.2: Time series estimates of parameters V_1, V_2, k_1 and k_2 obtained by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_1, x_2 and x_3. The solution envelopes display one standard deviation upper and lower bounds. In each figure, the true parameter value is marked as a dashed black line, the mean as a solid red curve, and the median as a dashed blue curve.

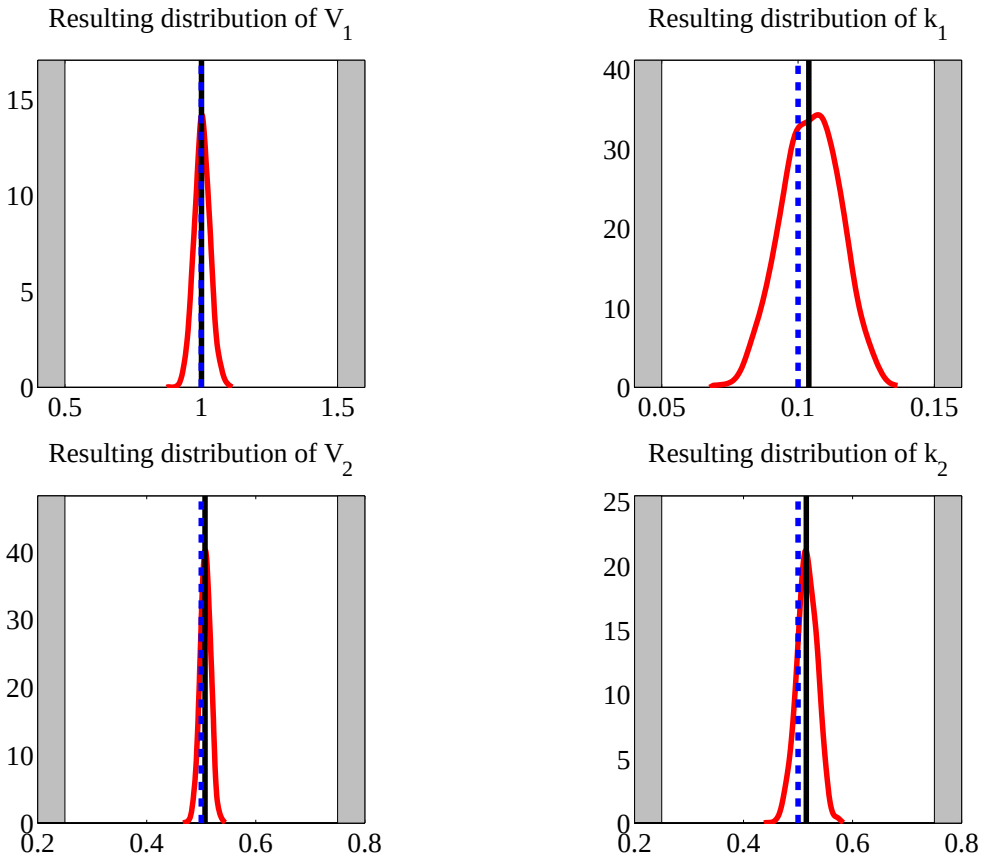

Figure 5.3: Smooth histograms of parameters V_1, V_2, k_1 and k_2 as determined by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_1, x_2 and x_3. In each figure, the true value of the parameter is marked as a dashed blue line, the distribution as a solid red curve, and the mean value as a solid black line. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

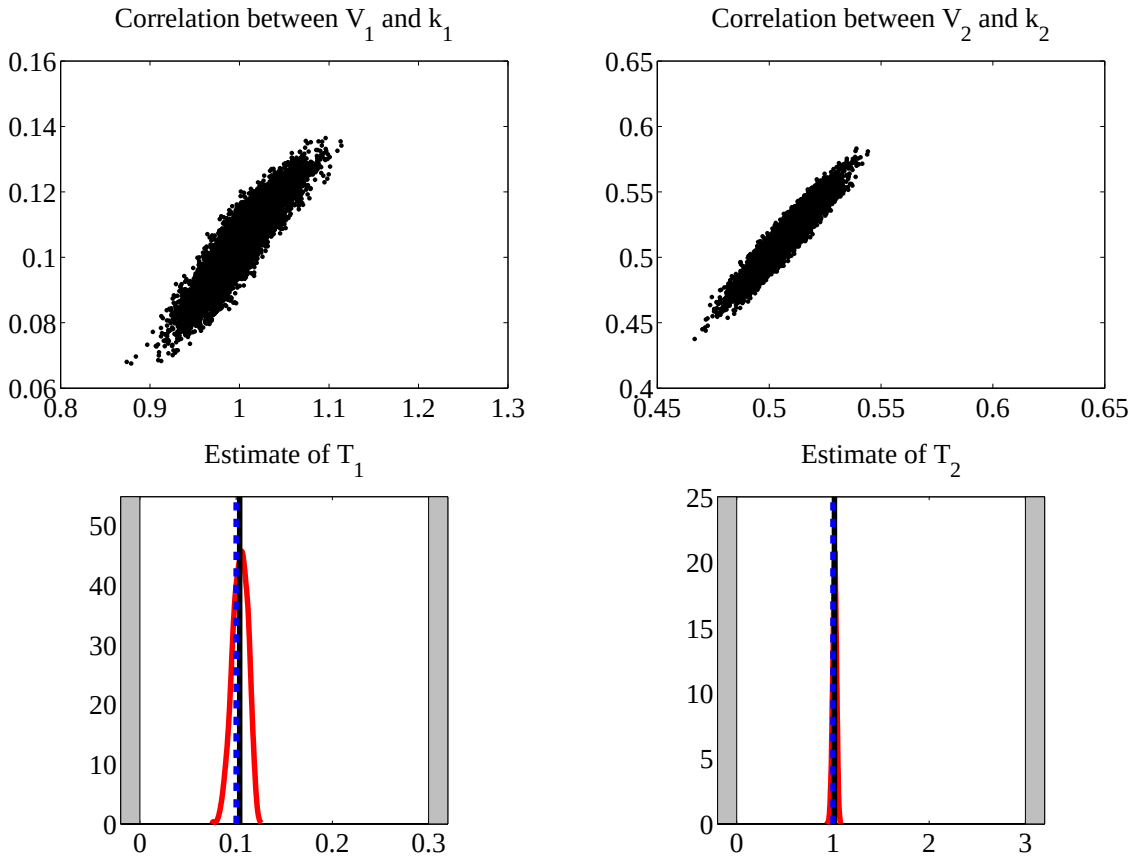

Figure 5.4: Top: Scatter plots of V_1 and k_1 (left) and V_2 and k_2 (right) obtained by LMM PF-SMC using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_1, x_2 and x_3. Bottom: Estimated smooth histograms of time constants T_1 (left) and T_2 (right). In each figure, the true value of the time constant is plotted in a dashed blue line, the estimated distribution in solid red, and the mean value in solid black. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

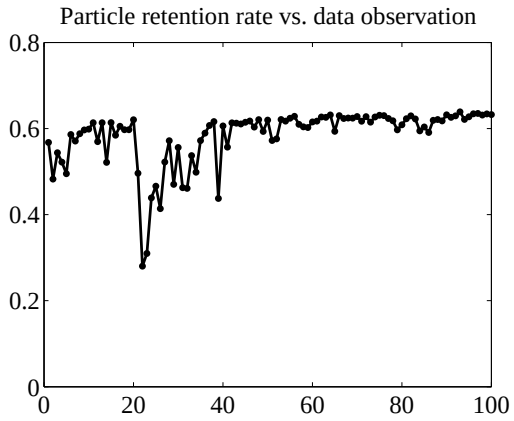

Figure 5.5: Particle retention rate at each datum observation. Here the data consist of 100 measurements of components x_1, x_2 and x_3.

respectively. Figure 5.8 displays the scatter plots of V_1 and k_1 and of V_2 and k_2, along with the estimated distributions for T_1 and T_2, while Figure 5.9 gives the PRR at each datum arrival.

The larger posterior uncertainty bounds in Figure 5.6 as compared to Figure 5.2 demonstrate the effect of having less data on the parameter estimation. We still see positive correlation between V_j and k_j, $j = 1, 2$, however the estimate of the time constant T_1 is less accurate here.

For the third computed example, we take 100 measurements of two of the states, namely, x_2 and x_3. Thus the data contain no information about x_1, which we refer to as a blind component of the system. The time series estimates and resulting smooth histograms of the parameters are shown in Figures 5.10 and 5.11, respectively. Figure 5.12 shows the V_1–k_1 and V_2–k_2 scatter plots, along with the estimated distributions for T_1 and T_2, while Figure 5.13 gives the PRR at each datum arrival.

Figure 5.10 shows the severe effect that having no data for x_1 has on the filter's ability to recover k_1, though V_1 and the other two parameters are well-recovered. The correlation plots in Figure 5.12 show that while positive correlation is retained between V_2 and k_2, it is lost between V_1 and k_1. Thus while the time constant T_2 can still be well-estimated, the uncertainty in the estimate for T_1 is much larger.

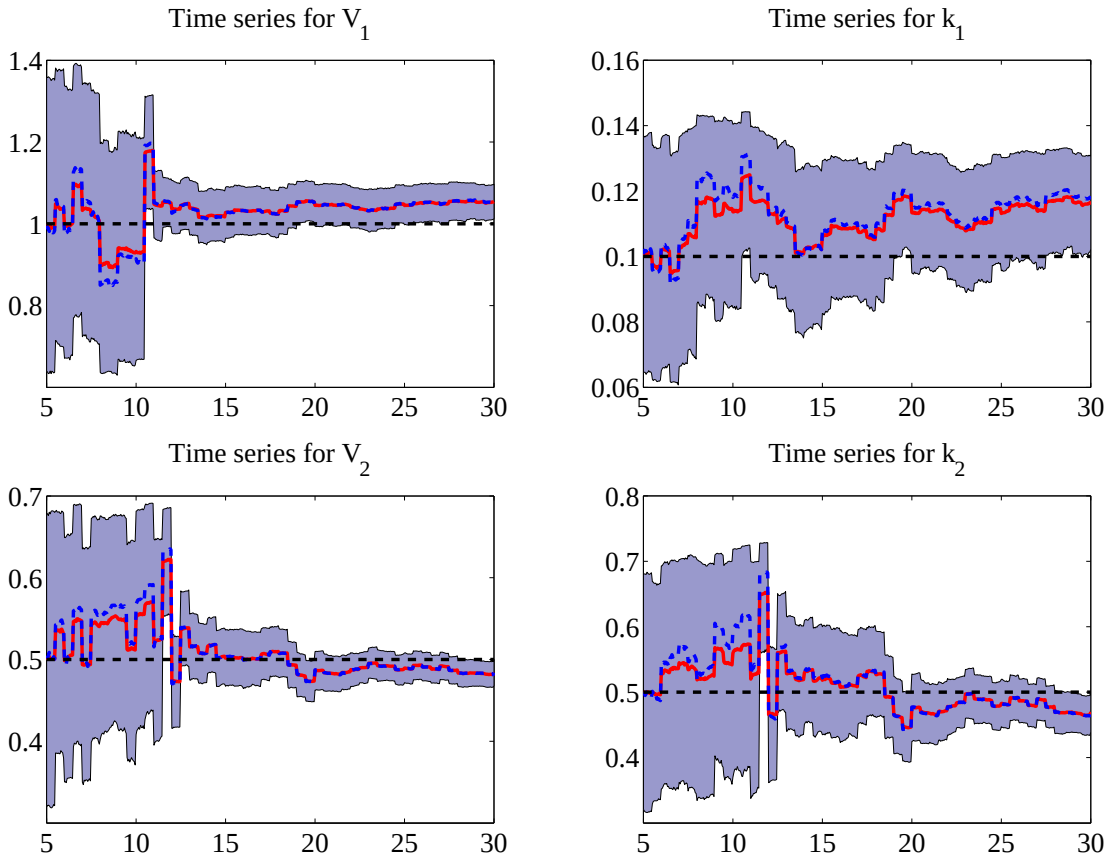

Figure 5.6: Time series estimates of parameters V_1, V_2, k_1 and k_2 obtained by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 50 measurements of components x_1, x_2 and x_3. Solution envelopes display one standard deviation upper and lower bounds. In each figure, the true parameter value is plotted as a dashed black line, the mean as a solid red curve, and the median as dashed blue curve.

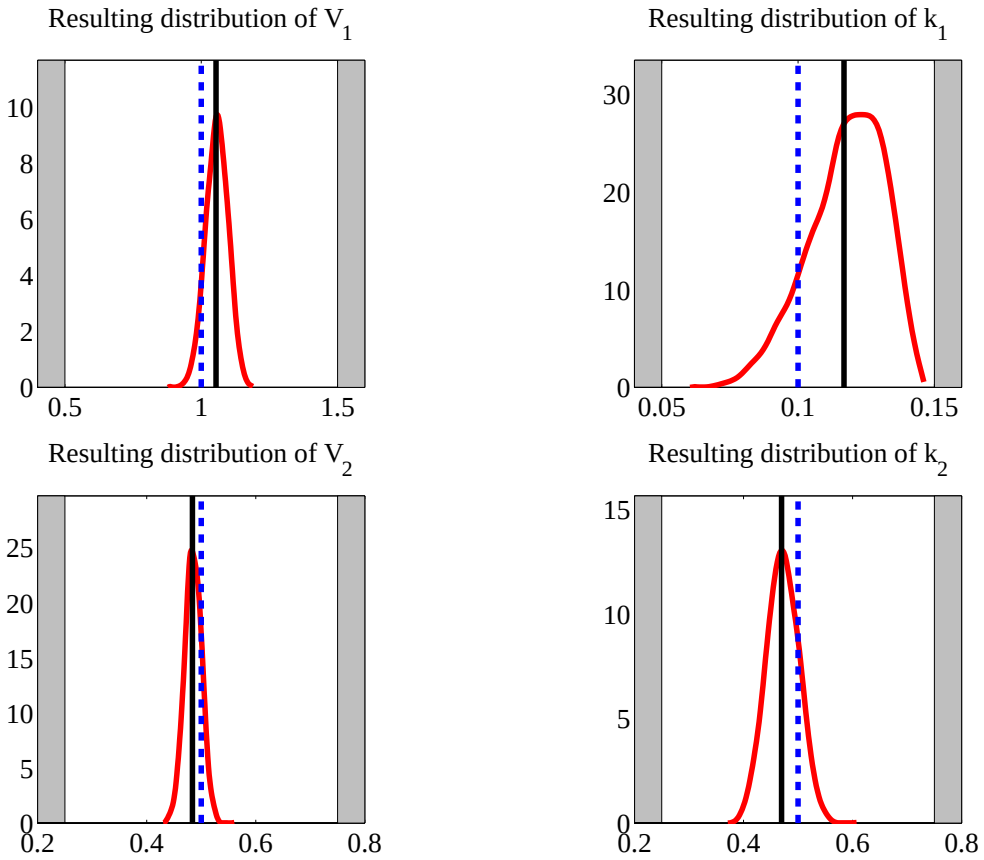

Figure 5.7: Smooth histograms of parameters V_1, V_2, k_1 and k_2 as determined by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 50 measurements of components x_1, x_2 and x_3. In each figure, the true value of the parameter is plotted as a dashed blue line, the distribution as a solid red curve, and the mean value as a solid black line. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

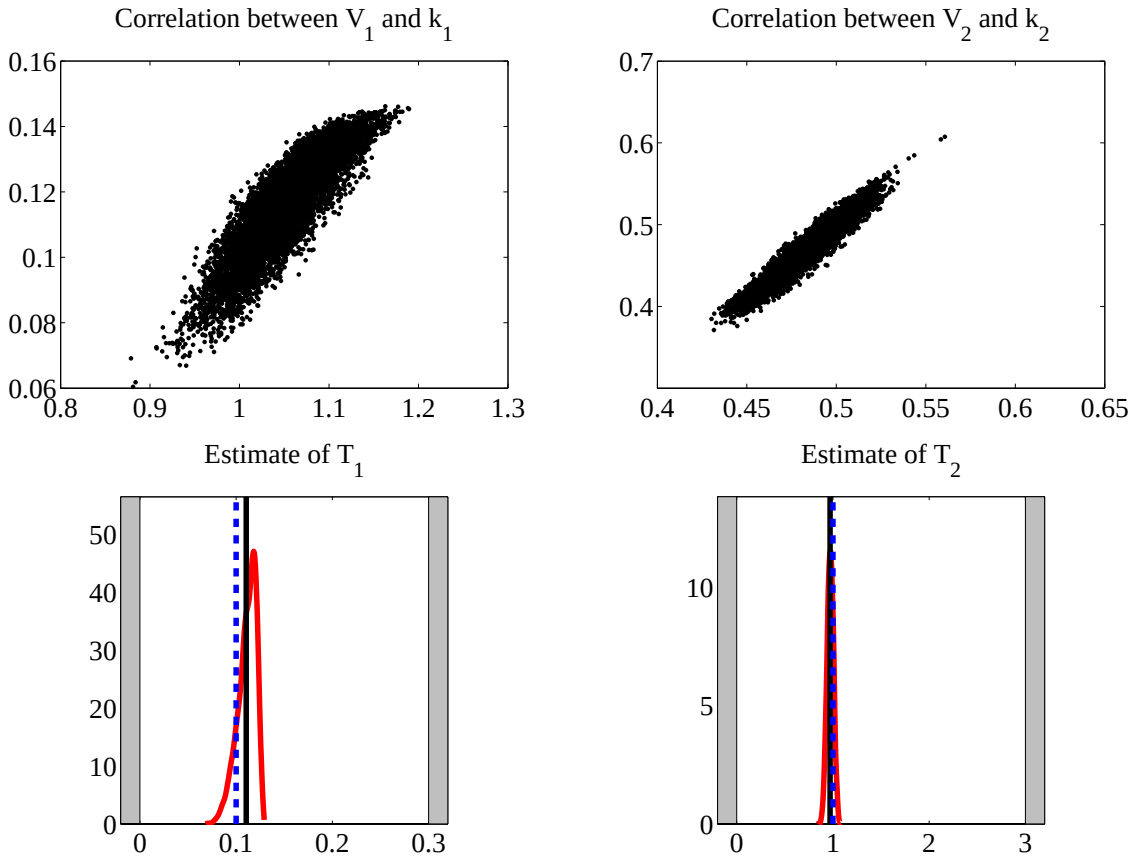

Figure 5.8: Top: Scatter plots of V_1 and k_1 (left) and V_2 and k_2 (right) obtained by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 50 measurements of components x_1, x_2 and x_3. Bottom: Estimated smooth histograms of time constants T_1 (left) and T_2 (right). In each figure, the true value of the time constant is plotted as a dashed blue line, the distribution as a solid red curve, and the mean value as a solid black line. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

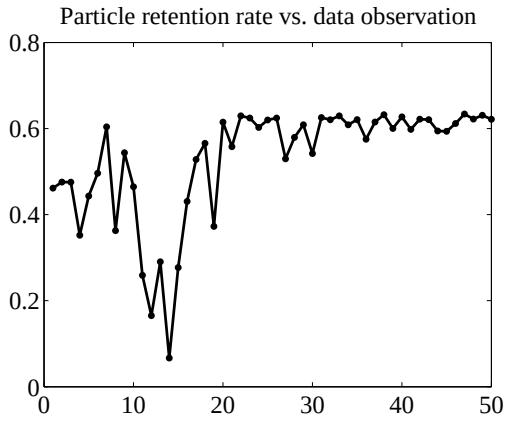

Figure 5.9: Particle retention rate at each datum observation. Here the data consist of 50 measurements of components x_1, x_2 and x_3.

The present chapter addresses the questions of stability and accuracy of combined state and parameter estimation problems when the underlying dynamical model is a non-stochastic ODE with ill-determined initial values and/or coefficients. It is shown that by using suitable LMM solvers, it is possible to treat the problem using existing sequential estimation methods with the advantage of having good control of the stability and the accuracy, and that the numerical scheme provides a natural way to set the innovation variance. The computed examples above suggest that both the frequency of data and the number of measured components of the underlying system of differential equations have an effect on the resulting parameter estimates.

The LMM PF-SMC approach is particularly appealing when the underlying system of differential equations is stiff. The stiffness of the system may be difficult to determine in particular when the system contains unknown parameters that are determined sequentially, since random draws of the parameter vector from the underlying distribution may easily contain parameter combinations that increase the stiffness uncontrollably. This is the case for many nonlinear problems that arise in applications.

An additional benefit of the proposed LMM PF-SMC methodology is that the algorithm is easily and efficiently parallelized, which allows for computational speedup.

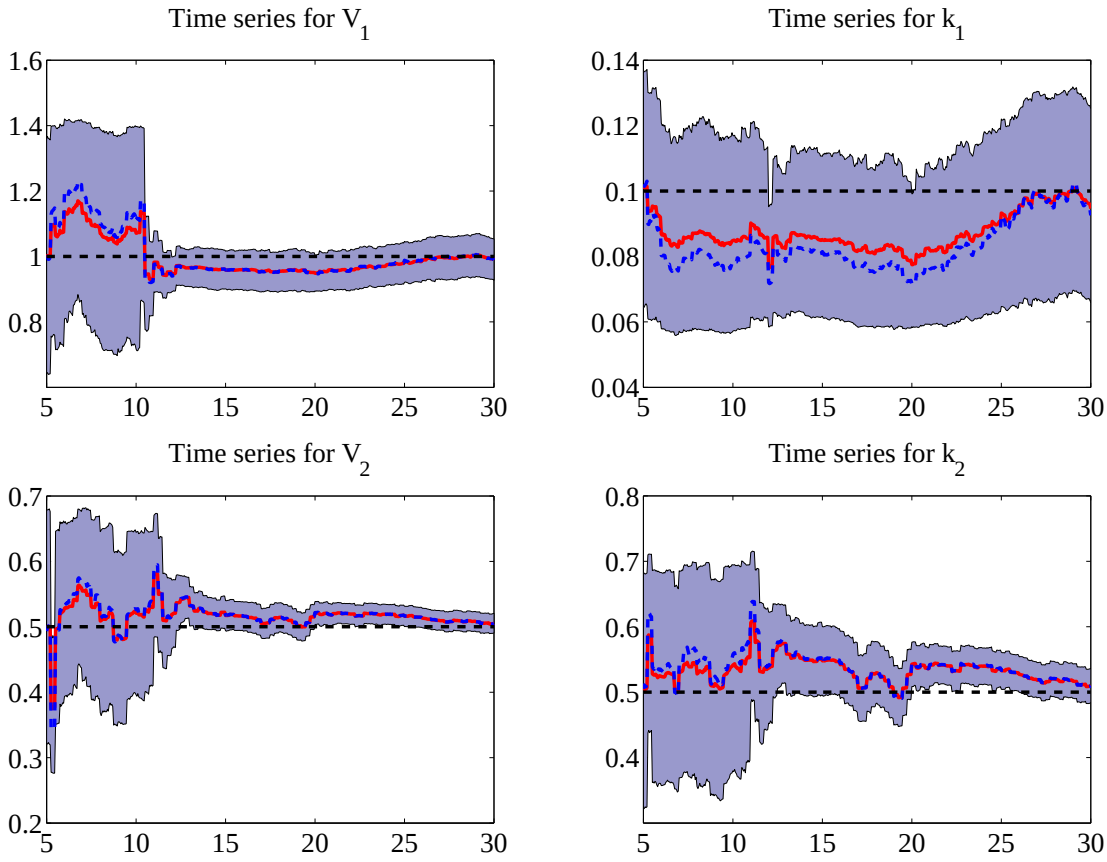

Figure 5.10: Time series estimates of parameters V_1, V_2, k_1 and k_2 obtained by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_2 and x_3. Solution envelopes display one standard deviation upper and lower bounds. In each figure, the true parameter value is marked as a dashed black line, the mean as a solid red curve, and the median as dashed blue curve.

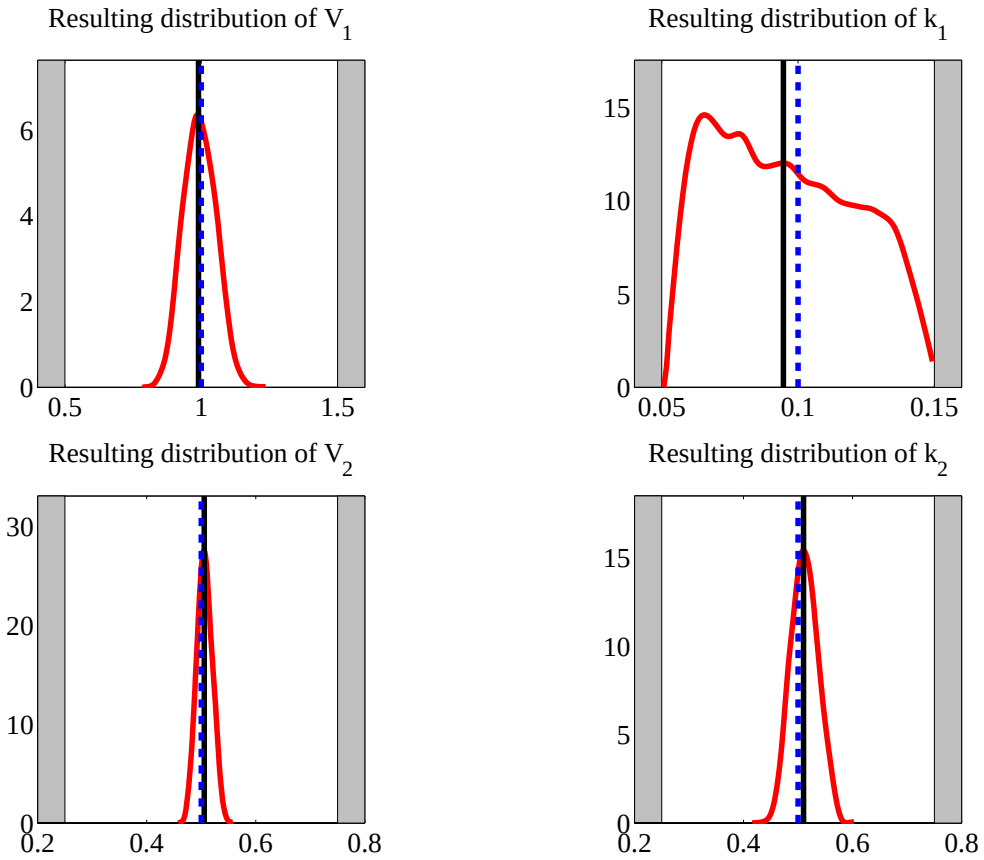

Figure 5.11: Resulting smooth histograms of parameters V_1, V_2, k_1 and k_2 as determined by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_2 and x_3. In each figure, the true value of the parameter is marked as a dashed blue line, the distribution as a solid red curve, and the mean value as a solid black line. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

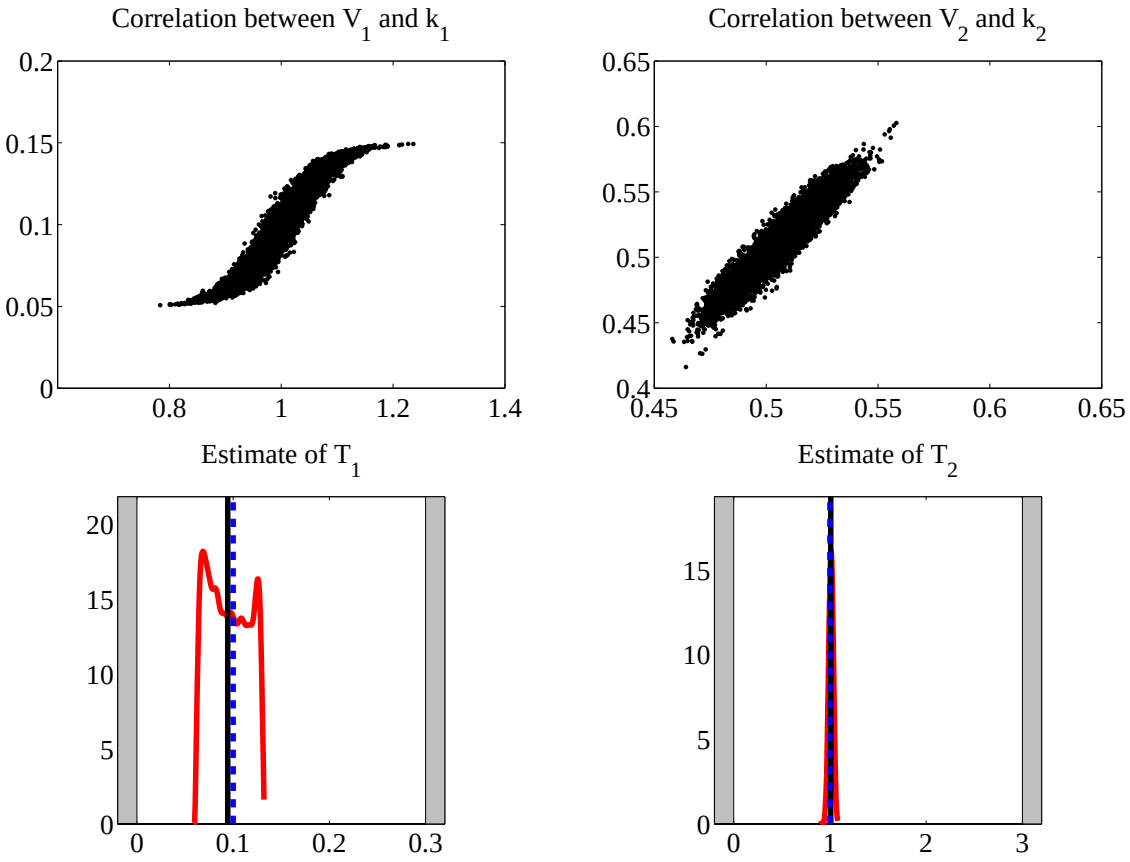

Figure 5.12: Top: Scatter plots of V_1 and k_1 (left) and V_2 and k_2 (right) obtained by the LMM PF-SMC algorithm using BDF3 for propagation and BDF4 for error control with 100 measurements of components x_2 and x_3. Bottom: Estimated smooth histograms of time constants T_1 (left) and T_2 (right). In each figure, the true value of the time constant is marked as a dashed blue line, the estimated distribution as a solid red curve, and the mean value as a solid black line. Shaded regions to the left and right indicate the lower and upper bounds used for the prior, respectively.

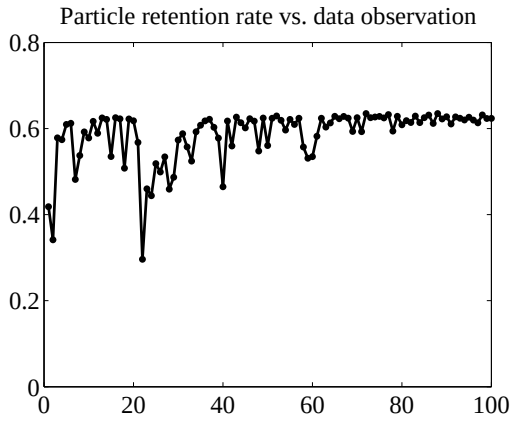

Figure 5.13: Particle retention rate at each datum observation. Here the data consist of 100 measurements of components x_2 and x_3.

This is particularly useful when propagating a large number of particles. We explore two computationally efficient implementation strategies for LMM PF-SMC, namely, parallelization and vectorization, in Chapter 6.

Chapter 6

Implementation Strategies for LMM PF-SMC

In the previous chapter, we presented a new variant of PF-SMC in which the time integration is performed using LMM numerical solvers; this algorithm was originally proposed in [10], where we derived an LMM PF-SMC algorithm for combined parameter and state estimation, outlined in Algorithm 8, inspired by [37]. Our aim in the current chapter is to explore the computational efficiency of the aforementioned LMM PF-SMC algorithm by testing different implementation strategies.

The propagation time in the sequential implementation of the algorithm at each step is determined by the sum of the propagation times of the individual particles, as each particle needs to be propagated before the algorithm moves forward. One way to potentially speed up this process is by parallelizing the algorithm. It is well known that particle filtering algorithms are amenable to parallelization due to the independent propagation of each particle, affording a straightforward way to split particles equally among processors. Such implementation can be especially useful when propagating a large number of particles.

However, it is important to recognize that in order for the parallelization to be efficient, the propagation of each particle should require approximately the same time: this is guaranteed in the present LMM-based approach, since each particle is

propagated using the same fixed time step, but it may fail to be true if, e.g., variable step size propagation with prescribed accuracy is used. Furthermore, parallelization is not the only option: in fact, by formulating the problem in a vectorized fashion, it is possible to arrive at an implementation of the LMM PF-SMC algorithm which takes full advantage of the computer architecture, achieving a high level of efficiency.

In Section 6.1, we set the stage for the parallelized and vectorized formulations of the LMM PF-SMC algorithm, which will be implemented and tested for the problem considered for the computed examples in Chapter 5. We derive the parallelized version of the algorithm and give the corresponding central processing unit (CPU) timing results for parallel MATLAB implementations in Section 6.2; we do the same for the vectorized version of the algorithm in Section 6.3. In Section 6.4, we compare the computational efficiency of parallelized and vectorized implementations of LMM PF-SMC as applied to a highly stiff system of ODEs arising from the spatial discretization of a two-dimensional advection-diffusion problem. The results are summarized in Section 6.5, where we show that both the size and structure of the problem determine whether the parallelized or vectorized version of the algorithm is more efficient.

Both the parallelized and vectorized versions of the LMM PF-SMC algorithm, along with a careful comparison of the computation times versus that of the sequential version for a suite of test problems in MATLAB, are presented in the manuscript [135].

6.1 To parallelize or vectorize?

In this chapter, we address which implementation of the LMM PF-SMC algorithm for combined parameter and state estimation derived in Chapter 5 is superior with respect to computational efficiency in a high-level language on rather standard computing equipment which, in this case, means a Dell Alienware Aurora R4 desktop computer with 16 GB RAM and an Intel® Core™ i7-3820 processor (CPU @ 3.60GHz) with 8 virtual cores, i.e., 4 cores and 8 threads with hyper-threading capability. Thus we set the local cluster of this computer to have a maximum of 8 MATLAB workers available. The baseline computing time to which the different approaches will be

compared is the execution time of the LMM PF-SMC algorithm with a given number of particles when using a single processor in MATLAB R2013a.

We test the effects of different implementation strategies on the computational time of the LMM PF-SMC combined parameter and state estimation algorithm on the system of nonlinear ODEs (5.17), which was used in the computed examples in Section 5.4. In our tests for computational efficiency, the data consist of noisy observations of all three of the components x_1, x_2 and x_3 at 50 time instances, and we aim to estimate the unknown parameters V_j and k_j, $j = 1, 2$. In the following sections, we describe how to parallelize and vectorize the algorithm, comparing the computational times with the sequential version of the algorithm.

6.2 Parallelization (via parallel loops)

Particle filter methods are known to be amenable to parallelization, and this applies naturally to our LMM PF-SMC algorithm: each particle can be independently propagated at the beginning of Step 2, but the whole ensemble is needed for the computation of the fitness weights and the generation of the daughter particles. Subsequently, each particle can be repropagated independently in Step 4. Thus the propagation and repropagation steps lend themselves trivially to parallelization, if several processors are available.

We can propagate/repropagate each particle sequentially on a single processor in MATLAB through use of `for`-loops, e.g.,

```
for j = 1:npart
    xi(:,j) = Psi(input(j));
end
```

where `Psi` denotes an LMM time integrator whose corresponding `input` parameters depend on the loop index `j`, `xi` denotes the propagated/repropagated particles, and `npart` is the number of particles.

The main reorganization of the algorithm for parallelization concerns these `for`-loops which, if the execution for each index is independent of all the others, can be partitioned and distributed amongst the available processors (or workers) in the pool. This is done in MATLAB using the command `parfor`:

```
parfor j = 1:npart
    xi(:,j) = Psi(input(j));
end
```

which triggers each worker to concurrently execute some portion of the iterations of the `parfor`-loop and communicate the results back to the client machine. Available workers are explicitly invoked in MATLAB by use of the command

```
matlabpool open
```

and can be accessed locally from the client machine or remotely from a cluster, depending on the choice of scheduler. In this thesis, we focus on the use of local resources.

The best performance is achieved when all workers take approximately the same time to complete the task, as the slowest execution time determines the speed of the parallel loop. This observation highlights the advantage of using the same time step in the numerical time integration of all particles in a parallel setting. On the other hand, most ODE solvers for stiff and non-stiff problems, including the MATLAB built-in time integrators, control the accuracy in the time integration by adaptively choosing the time step, with the result that the propagation of two different particles may require very different times, depending on the stiffness induced by the different parameter values. This violates the principle of equal load on the workers that is essential for a good parallel performance. Propagation of all particles by LMMs with the same time steps, on the other hand, ensures that the time required for each

particle is the same, thus guaranteeing that each worker will complete the assigned task at the same instance, reducing idle time.

To discuss the timing results of our computed examples, we introduce two important concepts in parallel computing: speedup and efficiency. The *speedup*, S_P, using P processors is defined as

$$S_P = \frac{T_1}{T_P},$$

where T_1 is the execution time of the sequential algorithm and T_P is the execution time of the parallel algorithm on P processors. Linear, or ideal, speedup is obtained when $S_P \approx P$, and super-linear speedup occurs when $S_P > P$, although this is rare in practice. The parallel *efficiency*, E_P, using P processors is defined as

$$E_P = \frac{S_P}{P},$$

where S_P is the speedup using P processors. Efficiency is a performance measure used to estimate how well the processors are utilized in running a parallel code. Note that $E_P = 1$ for algorithms with linear speedup and, trivially, for algorithms run sequentially, i.e., on a single processor. For further details, we refer to [136].

Tables 6.1 and 6.2 list the CPU times (in seconds) for solving the parameter estimation problem for the system (5.17) with LMM PF-SMC using the first three time integrators of each LMM family for the propagation with $N = 5,000$ and $N = 50,000$ particles, respectively, both sequentially and in parallel with 8 workers, along with the corresponding speedup and efficiency. When computing the efficiency, we interpret the number of processors as the number of virtual cores, which here matches the number of MATLAB workers used in the parallel computation, while recognizing that efficiency is traditionally calculated using the number of physical cores. The fixed time step $h = 0.05$ is chosen so that all the methods are stable. Here the innovation variance is computed using the HOMEC technique, as described in Section 5.2.

The results in Tables 6.1 and 6.2 suggest that parallelizing the LMM PF-SMC algorithm via parallel loops for test problem (5.17) shows a speedup when using 8

LMM	Sequential	8 Workers	S_8	E_8
AB1	9.1767e+02	4.8311e+02	1.8995	0.2374
AB2	1.6672e+03	6.8397e+02	2.4375	0.3047
AB3	2.3921e+03	8.7218e+02	2.7426	0.3428
AM1	5.1506e+03	1.6512e+03	3.1192	0.3899
AM2	6.5278e+03	2.0474e+03	3.1883	0.3985
AM3	7.7925e+03	2.4821e+03	3.1395	0.3924
BDF1	4.1441e+03	1.4419e+03	2.8740	0.3592
BDF2	4.7411e+03	1.7234e+03	2.7510	0.3439
BDF3	5.4694e+03	2.0309e+03	2.6931	0.3366

Table 6.1: CPU times (in seconds) sequentially and in parallel with 8 workers, along with the corresponding speedup S_8 and efficiency E_8, for solving the parameter estimation problem for system (5.17) with LMM PF-SMC using the first three LMM time integrators of each family for propagation and HOMEC innovation with time step $h = 0.05$ and $N = 5,000$ particles.

LMM	Sequential	8 Workers	S_8	E_8
AB1	1.4862e+04	3.9523e+03	3.7603	0.4700
AB2	2.2382e+04	5.7519e+03	3.8912	0.4864
AB3	2.9011e+04	7.4767e+03	3.8801	0.4850
AM1	5.6307e+04	1.5125e+04	3.7227	0.4653
AM2	6.9272e+04	1.9292e+04	3.5907	0.4488
AM3	8.2632e+04	2.3145e+04	3.5702	0.4463
BDF1	4.6328e+04	1.3761e+04	3.3666	0.4208
BDF2	5.3489e+04	1.6592e+04	3.2238	0.4030
BDF3	5.8351e+04	1.9762e+04	2.9527	0.3691

Table 6.2: CPU times (in seconds) sequentially and in parallel with 8 workers, along with the corresponding speedup S_8 and efficiency E_8, for solving the parameter estimation problem for system (5.17) with LMM PF-SMC using the first three LMM time integrators of each family for propagation and HOMEC innovation with time step $h = 0.05$ and $N = 50,000$ particles.

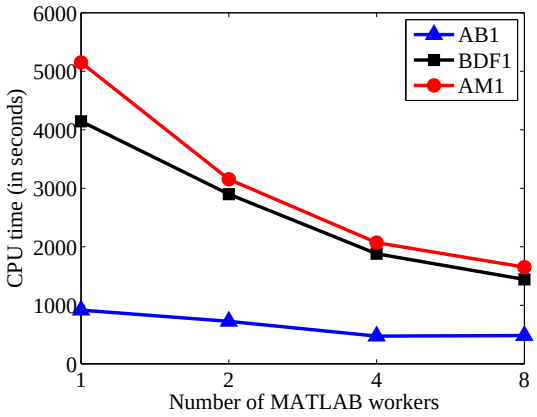

Figure 6.1: CPU times (in seconds) for solving the parameter estimation problem for system (5.17) in parallel with LMM PF-SMC using the solvers AB1, BDF1 and AM1 for propagation with $N = 5,000$ particles against the number of MATLAB workers assigned during parallel computation.

workers, more so when propagating $N = 50,000$ particles than $N = 5,000$ particles. In turn, the efficiency is higher for the sample of $N = 50,000$ particles. This could be due to the overhead of the communication time between workers. Not surprisingly, the results also show that it takes more time to propagate with the implicit (AM and BDF) solvers than with the explicit (AB) solvers, both sequentially and in parallel, and that the CPU time also increases with the order of the method. For both sample sizes, the AM methods take the longest time to propagate but benefit significantly from parallelization.

Although we assign the number of MATLAB workers in the above experiments to match the number of virtual cores of the computer, it is possible to use less parallel workers and still see a speedup. Figure 6.1 plots the CPU times (in seconds) for solving the parameter estimation problem for the system (5.17) with LMM PF-SMC using the solvers AB1, BDF1 and AM1 for propagation with $N = 5,000$ particles against the number of MATLAB workers assigned during parallel computation. The plot demonstrates that, for this example, the largest speedup does occur when 8 workers are used, particularly for the implicit solvers BDF1 and AM1.

6.3 Vectorization

One of the strengths of MATLAB is its adeptness at performing operations on block arrays, where it addresses vectors, matrices, or more general array structures. In fact, starting with the R2007a version, MATLAB has supported multithreaded computation for several built-in operations and functions that are combinations of element-wise functions. This is the case, e.g., for all element-wise vector operations and for the solution of linear systems by the backslash (`mldivide`) operator. Implicit multithreading is automatically initiated if MATLAB identifies an operation as amenable to speedup with the use of multiple processors, and does not require any special command or alteration of the code, unlike in the explicit parallelization described above.

To test the performance of the vectorized LMM PF-SMC on the system (5.17), we assemble a stacked column vector with $3N$ entries, corresponding to the three state variables for the N particles. To maximize the number of vector-vector or matrix-vector operations, in the implementation of Newton's method for the solution of the nonlinear system required for the implicit AM and BDF methods, we define an aggregate block diagonal Jacobian matrix, whose diagonal blocks are the Jacobian matrices of the systems corresponding to each particle. Since this $3N \times 3N$ matrix contains mostly zeros, we take advantage of its sparsity by using the `sparse` structure of MATLAB, which stores the matrix in the form of three vectors, containing the rows, columns, and values of the nonzero entries. Because the structure of each diagonal block is the same, we assign all the corresponding nonzero entries at once, by means of element-wise operations between vectors with N entries. For the problem at hand, the code for computing the aggregate Jacobian matrix for the BDF methods is as follows:

```
% Compute Jacobian entries for each block

% Entry (1,1)
col_ind = 1;
```

```
row_ind = 1;
cols = col_ind:m:m*N-m+col_ind;
rows = row_ind:m:m*N-m+row_ind;
temp = theta(1,:).*theta(3,:)./(x(1,:)+theta(3,:)).^2;
vals = num + coeff*temp;

% Entry (2,1)
row_ind = 2;
cols = [cols col_ind:m:m*N-m+col_ind];
rows = [rows row_ind:m:m*N-m+row_ind];
vals = [vals -coeff*temp];

% Entry (2,2)
col_ind = 2;
cols = [cols col_ind:m:m*N-m+col_ind];
rows = [rows row_ind:m:m*N-m+row_ind];
temp = theta(2,:).*theta(4,:)./(x(2,:)+theta(4,:)).^2;
vals = [vals, num+coeff*temp];

% Entry (3,2)
row_ind = 3;
cols = [cols col_ind:m:m*N-m+col_ind];
rows = [rows row_ind:m:m*N-m+row_ind];
vals = [vals -coeff*temp];

% Entry (3,3)
col_ind = 3;
cols = [cols col_ind:m:m*N-m+col_ind];
rows = [rows row_ind:m:m*N-m+row_ind];
vals = [vals, num+coeff*flow*ones(1,N)];
```

Note that the parameters `num` and `coeff` determine the order of the BDF method chosen for propagation. The computation of the aggregate Jacobian matrices for the AM methods follows similarly.

Tables 6.3 and 6.4 list the CPU times (in seconds) for solving the parameter estimation problem for system (5.17) with vectorized LMM PF-SMC using the first

LMM	Vectorized	$S_{\text{vec}}^{\text{seq}}$	$S_{\text{vec}}^{\text{par8}}$
AB1	6.5093e+01	14.0978	7.4218
AB2	6.5947e+01	25.2801	10.3715
AB3	6.6747e+01	35.8375	13.0668
AM1	8.3622e+01	61.5938	19.7463
AM2	8.6723e+01	75.2724	23.6091
AM3	8.6397e+01	90.1948	28.7286
BDF1	8.3499e+01	49.6304	17.2690
BDF2	8.4162e+01	56.3323	20.4773
BDF3	8.5340e+01	64.0901	23.7976

Table 6.3: CPU times (in seconds) for solving the parameter estimation problem for system (5.17) with the vectorized LMM PF-SMC algorithm using the first three LMM time integrators of each family for propagation and HOMEC innovation with time step $h = 0.05$ and $N = 5,000$ particles, along with the corresponding speedups $S_{\text{vec}}^{\text{seq}}$ and $S_{\text{vec}}^{\text{par8}}$ over the sequential and parallel times shown in Table 6.1, respectively.

three LMM time integrators of each family for the propagation steps and the HOMEC technique for assigning the innovation variance with $N = 5,000$ and $N = 50,000$ particles, respectively. The corresponding speedups over the sequential and parallel times given in Tables 6.1 and 6.2 are listed in the columns $S_{\text{vec}}^{\text{seq}}$ and $S_{\text{vec}}^{\text{par8}}$, respectively.

The results in Tables 6.3 and 6.4 suggest that vectorizing the LMM PF-SMC algorithm for problem (5.17) yields significant speedup over the sequential and even parallel implementations. For example, when propagating the particles with AM3, the vectorized version is about 13.5 times faster than the sequential and 3.8 times faster than the parallel when $N = 50,000$, and about 90.2 times faster than the sequential and 28.7 times faster than the parallel when $N = 5,000$. Moreover, the computing time for the vectorized LMM PF-SMC is essentially insensitive to the choice of LMM family, to the order of the method, and to the method used to estimate the discretization error.

Choosing BDF2 as the LMM for the propagation step, we also demonstrate that using the HOMEC technique for computing the innovation results in similar timing as when using the Milne device. Table 6.5, Column 2 lists the CPU times for solving the parameter estimation problem for system (5.17) with vectorized LMM PF-SMC using

LMM	Vectorized	$S_{\text{vec}}^{\text{seq}}$	$S_{\text{vec}}^{\text{par8}}$
AB1	6.0498e+03	2.4566	0.6533
AB2	6.0040e+03	3.7278	0.9580
AB3	6.0124e+03	4.8251	1.2435
AM1	6.1911e+03	9.0948	2.4431
AM2	6.1865e+03	11.1972	3.1184
AM3	6.1318e+03	13.4760	3.7746
BDF1	6.1722e+03	7.5058	2.2295
BDF2	6.1413e+03	8.7096	2.7017
BDF3	6.2543e+03	9.3298	3.1597

Table 6.4: CPU times (in seconds) for solving the parameter estimation problem for system (5.17) with the vectorized LMM PF-SMC algorithm using the first three LMM time integrators of each family for propagation and HOMEC innovation with time step $h = 0.05$ and $N = 50,000$ particles, along with the corresponding speedups $S_{\text{vec}}^{\text{seq}}$ and $S_{\text{vec}}^{\text{par8}}$ over the sequential and parallel times shown in Table 6.2, respectively.

BDF2 for propagation and BDF3 for error control, computing the innovation term via the HOMEC technique, as the number of particles N increases from $N = 5,000$ to $N = 200,000$. The timings when coupling BDF2 with AM1 for error control, computing the innovation term via the Milne device technique, are listed in Table 6.5, Column 3. We remark that the timings for N particles when using the HOMEC method are almost identical to the timings when using the Milne device, which suggests that the choice of innovation technique has negligible effect on the CPU time of the problem. Figure 6.2 plots these results in log-scale, where a log-linear relationship between the CPU time and the number of particles is visible.

6.4 Computed examples with a 2D advection-diffusion model

To test the relative performance of the parallelized and vectorized implementations of the LMM PF-SMC algorithm on a different type of problem, we consider a two-dimensional (2D) advection-diffusion problem similar to that used in [137], which can be thought of as modeling, e.g., the spreading of contaminants. Consider the

N	HOMEC	Milne
5,000	8.4162e+01	7.9449e+01
10,000	2.8262e+02	2.7757e+02
20,000	1.0478e+03	1.0410e+03
50,000	6.1413e+03	6.1857e+03
100,000	2.4470e+04	2.3931e+04
200,000	9.5259e+04	9.5961e+04

Table 6.5: CPU times (in seconds) for solving the parameter estimation problem for system (5.17) with the vectorized LMM PF-SMC algorithm using BDF2 for propagation with HOMEC innovation and with Milne device innovation for time step $h = 0.05$ and N particles.

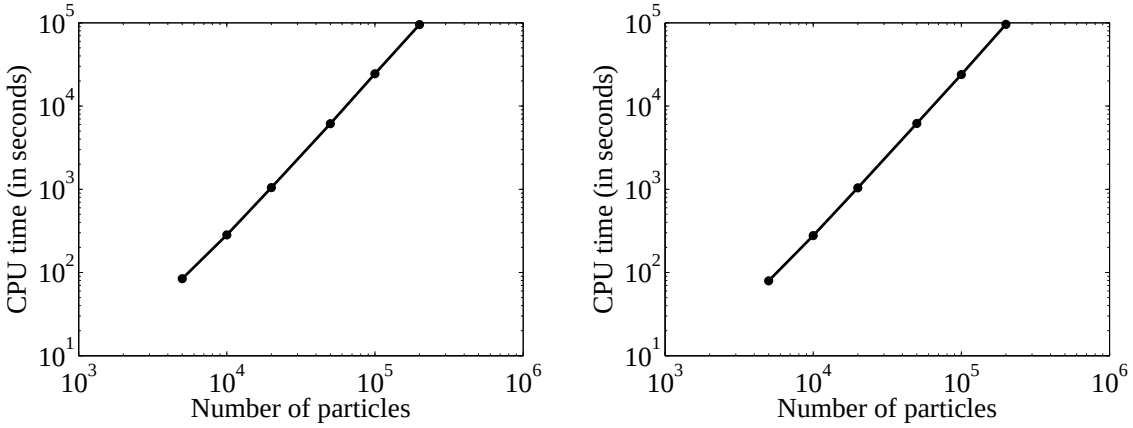

Figure 6.2: CPU time (in seconds) versus the number of particles for solving the parameter estimation problem for system (5.17) with the vectorized LMM PF-SMC algorithm using BDF2 with HOMEC innovation (left) and Milne device innovation (right), plotted in log-scale. See Table 6.5 for the corresponding data values.

i	1	2	3	4	5	6
γ_i	0.04	0.08	0.07	0.10	0.05	0.06
x_i^c	0.20	0.30	0.40	0.50	0.50	0.70
y_i^c	0.80	0.40	0.40	0.50	0.60	0.50

Table 6.6: Standard deviations γ_i and centers (x_i^c, y_i^c) for the six Gaussian plumes summed to form the initial condition (6.2) used to generate the simulated data for the discretized 2D advection-diffusion problem (6.1).

time-dependent PDE

$$\frac{\partial u}{\partial t} = \nabla \cdot (\mathsf{D}\nabla u) + \mathbf{c} \cdot \nabla u \tag{6.1}$$

over the square domain

$$\Omega = [0,1] \times [0,1] = \big\{(x,y) : 0 \le x \le 1, 0 \le y \le 1\big\}$$

with periodic boundary conditions. Here $\nabla u = \big(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\big)$, $\nabla\cdot$ is the divergence operator, D is a 2×2 matrix of constant diffusion coefficients and $\mathbf{c}$ is a 2×1 velocity vector describing the advection in the x and y directions.

The spatial domain is discretized into an $n \times n$ linearly-spaced square grid. Initially, at time $t = 0$, u is the sum of six Gaussian plumes,

$$u_0(x,y) = \sum_{i=1}^{6} \frac{1}{\gamma_i\sqrt{2\pi}} \exp\left\{ -\frac{1}{2\gamma_i^2}\big((x - x_i^c)^2 + (y - y_i^c)^2\big)\right\}, \tag{6.2}$$

whose standard deviations γ_i and centers (x_i^c, y_i^c) are listed in Table 6.6.

Since we assume periodic boundary conditions, we compute the solution on the grid of size $\widetilde{n} \times \widetilde{n}$, where $\widetilde{n} = n - 1$, represented in the form of the matrix

$$\mathsf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1,\widetilde{n}} \\ u_{21} & u_{22} & \cdots & u_{2,\widetilde{n}} \\ \vdots & \vdots & & \vdots \\ u_{\widetilde{n},1} & u_{\widetilde{n},2} & \cdots & u_{\widetilde{n},\widetilde{n}} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_{\widetilde{n}} \\ | & | & & | \end{bmatrix}_{\widetilde{n}\times\widetilde{n}}$$

which, for computational purposes, we write as a stacked column vector

$$U = \mathsf{U}(:) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{\widetilde{n}} \end{bmatrix}_{\widetilde{n}^2 \times 1}.$$

We assume that the 2×2 diffusion matrix

$$\mathsf{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

is symmetric positive definite and that its Cholesky decomposition is given by $\mathsf{D} = \mathsf{K}^T\mathsf{K}$, where

$$\mathsf{K} = \begin{bmatrix} k_1 & k_3 \\ 0 & k_2 \end{bmatrix}.$$

It is straightforward to verify that

$$d_{11} = k_1^2, \quad d_{12} = d_{21} = k_1 k_3, \quad \text{and} \quad d_{22} = k_2^2 + k_3^2.$$

We want to estimate the parameters k_1, k_2, k_3, and the components c_1 and c_2 of the velocity vector $\mathbf{c}$ using the LMM PF-SMC algorithm.

To discretize (6.1), we introduce the matrices

$$\begin{aligned} \frac{\partial}{\partial x} &\rightarrow B_1 = I_{\widetilde{n}} \otimes \ell \\ \frac{\partial}{\partial y} &\rightarrow B_2 = \ell \otimes I_{\widetilde{n}} \end{aligned}$$

where $I_{\widetilde{n}}$ is the $\widetilde{n} \times \widetilde{n}$ identity matrix, ℓ is an $\widetilde{n} \times \widetilde{n}$ finite difference matrix of the form

$$\ell = \begin{bmatrix} 1 & & & -1 \\ -1 & \ddots & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}_{\widetilde{n} \times \widetilde{n}}$$

and $\otimes$ denotes the Kronecker product. Therefore

$$\begin{aligned}
\nabla &\rightarrow \left[\begin{array}{c} B_1 \\ B_2 \end{array} \right]_{2\widetilde{n}^2 \times \widetilde{n}^2}, \\
\nabla u &\rightarrow \left[\begin{array}{c} B_1 \\ B_2 \end{array} \right] U, \\
\mathsf{D}\nabla u &\rightarrow \left[\begin{array}{cc} d_{11} & d_{12} \\ d_{21} & d_{22} \end{array} \right] \left[\begin{array}{c} B_1 \\ B_2 \end{array} \right] U = \left[\begin{array}{c} d_{11}B_1 + d_{12}B_2 \\ d_{21}B_1 + d_{22}B_2 \end{array} \right] U.
\end{aligned}$$

Since the entries of D are constant and $d_{12} = d_{21}$, we have that

$$\begin{aligned}
\nabla \cdot (\mathsf{D}\nabla u) &\rightarrow -\left[\begin{array}{cc} B_1^T & B_2^T \end{array} \right] \left[\begin{array}{c} d_{11}B_1 + d_{12}B_2 \\ d_{21}B_1 + d_{22}B_2 \end{array} \right] U \\
&= -\Big\{ B_1^T (d_{11}B_1 + d_{12}B_2) \\
&\qquad + B_2^T (d_{21}B_1 + d_{22}B_2) \Big\} U \\
&= -\Big\{ d_{11}B_1^T B_1 + d_{12}B_1^T B_2 + d_{21}B_2^T B_1 \\
&\qquad + d_{22}B_2^T B_2 \Big\} U \\
&= -\Big\{ d_{11}B_1^T B_1 + d_{12}(B_1^T B_2 + B_2^T B_1) \\
&\qquad + d_{22}B_2^T B_2 \Big\} U \\
&= -\Big\{ k_1^2 B_1^T B_1 + k_1 k_3 (B_1^T B_2 + B_2^T B_1) \\
&\qquad + (k_2^2 + k_3^2) B_2^T B_2 \Big\} U.
\end{aligned}$$

Similarly,

$$\mathbf{c} \cdot \nabla u \rightarrow \Big\{ c_1 B_1 + c_2 B_2 \Big\} U.$$

Thus the spatial discretization of the problem is given by

$$\nabla \cdot (\mathsf{D}\nabla u) + \mathbf{c} \cdot \nabla u \rightarrow \mathsf{L}U,$$

where the operator matrix

$$\begin{aligned} \mathsf{L} &= -\Big\{k_1^2 B_1^T B_1 + k_1 k_3 (B_1^T B_2 + B_2^T B_1) \\ &\qquad + (k_2^2 + k_3^2) B_2^T B_2 \Big\} + c_1 B_1 + c_2 B_2 \end{aligned} \tag{6.3}$$

is written in terms of the five parameters of interest. The matrices B_1, B_2, $B_1^T B_1$, $B_2^T B_2$ and $B_1^T B_2 + B_2^T B_1$ are sparse and need only be computed once. We now have a spatially discretized system of ODEs

$$\frac{dU}{dt} = \mathsf{L}U \tag{6.4}$$

which we can propagate forward in time via numerical integration. Note that since the system (6.4) obtained after spatial discretization with the MOL tends to exhibit stiffness, its numerical solution requires the use of stiff solvers.

6.4.1 Data generation

To generate the simulated data, we let

$$\mathsf{K} = \begin{bmatrix} 9 & 6 \\ 0 & 4 \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} 2.5 \\ -1.5 \end{bmatrix} \tag{6.5}$$

and propagate (6.4) over the time interval $[0, 30]$ in MATLAB using the built-in stiff ODE solver `ode15s` with relative tolerance 10^{-8}. The initial value, computed according to (6.2), and the reference solutions at 10, 20 and 30 seconds when $n = 40$ are shown in Figure 6.3. The solution is measured at 20 randomly selected, fixed spatial locations every one time unit, and we add Gaussian noise with

$$\text{noise level} \approx \frac{||\text{data at time } t = 0||_\infty}{50}$$

to the observations, thus yielding 30 noisy observations.

Observe that here the spatial discretization in both the data generation and the inverse model is the same, a practice that usually is referred to as an "inverse

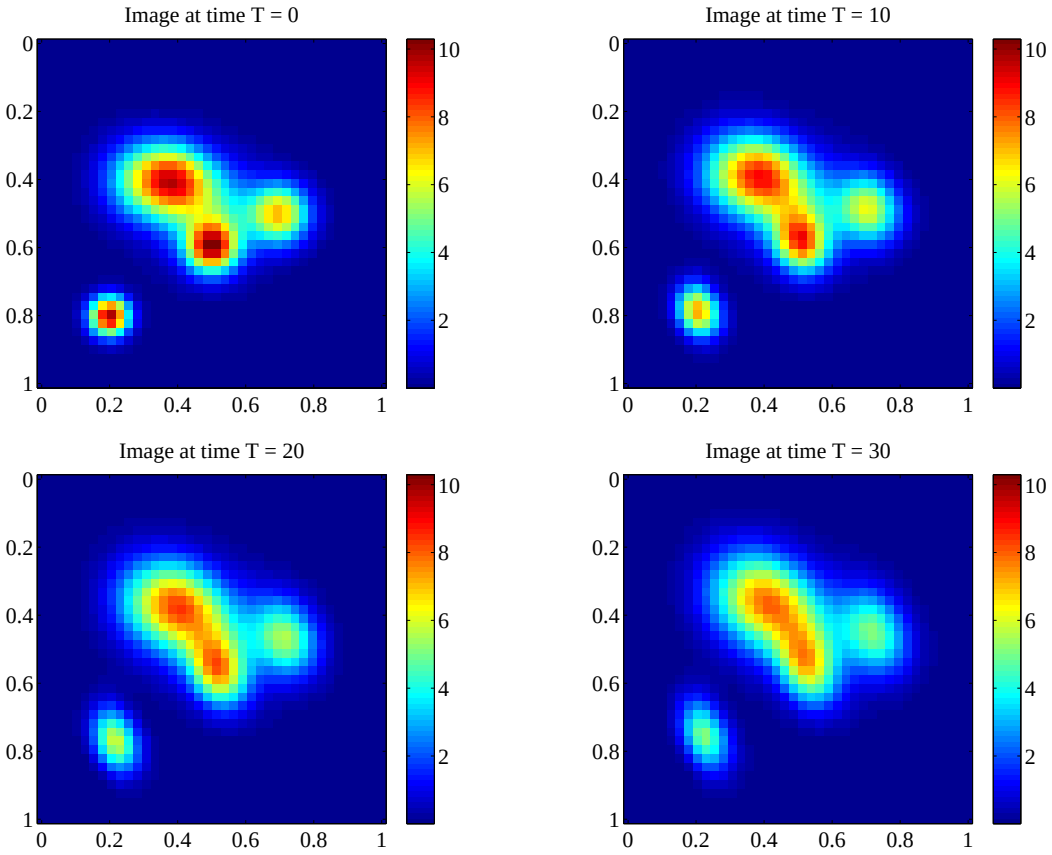

Figure 6.3: The initial image and reference solutions at 10, 20 and 30 seconds generated for the 2D advection-diffusion problem with discretization size $n = 40$ and the fixed parameter values given in (6.5).

crime." However, in this work, the advection-diffusion problem serves as a model for generating a large, stiff system of ODEs to test the parallel and vectorized performance of our algorithm.

6.4.2 Parallelized vs. vectorized scheme

The parallel implementation of the LMM PF-SMC algorithm for the system (6.4) follows the description in Section 6.2. Since each particle can be propagated/repropagated independently, the operator matrix L defined by (6.3) is built for each particle individually from its corresponding set of parameters in the `parfor` loop. We remark that because the matrix L depends on the five parameters of interest, it changes from particle to particle.

The vectorized implementation for this example requires more effort because separate L matrices must be built for all of the N particles from their corresponding sets of parameters at once. This is achieved by building a sparse block diagonal matrix of size $N\widetilde{n}^2 \times N\widetilde{n}^2$ encompassing all of the operator matrices corresponding to all of the individual particles, that is,

$$\mathsf{L} = \begin{bmatrix} \mathsf{L}^{(1)} & & & \\ & \mathsf{L}^{(2)} & & \\ & & \ddots & \\ & & & \mathsf{L}^{(N)} \end{bmatrix}_{N\widetilde{n}^2 \times N\widetilde{n}^2}, \tag{6.6}$$

where $\mathsf{L}^{(j)}$ is the $\widetilde{n}^2 \times \widetilde{n}^2$ operator matrix corresponding to the jth particle.

This block matrix can be built in MATLAB in several different ways, including the use of cell functions or Kronecker products. For the former, invoking the command

```
L1 = cellfun(@(x) x*B1B1, num2cell(d11), 'UniformOutput', false);
```

calls a cell function which takes each element of the vector `d11` (containing the constant d_{11} elements for each particle) and multiplies it by the sparse matrix `B1B1` (which denotes the matrix $B_1^T B_1$). Then the command

```
L1 = sparse(blkdiag(L1{:}));
```

yields a sparse block diagonal matrix of the form

$$\begin{bmatrix} d_{11}^{(1)} B_1^T B_1 & & & \\ & d_{11}^{(2)} B_1^T B_1 & & \\ & & \ddots & \\ & & & d_{11}^{(N)} B_1^T B_1 \end{bmatrix}_{N\widetilde{n}^2 \times N\widetilde{n}^2},$$

which is the first term in the expression for `L`. After computing the other terms in a similar fashion, we have that

```
L = -(L1 + L2 + L3) + L4 + L5;
```

Alternatively, we can construct `L1` with the command

```
L1 = kron(spdiags(d11',0,npart,npart),B1B1);
```

which places the elements of the vector `d11` along the main diagonal of an $N \times N$ sparse diagonal matrix and computes its Kronecker product with the sparse matrix `B1B1`. Since both factors are sparse and the Kronecker product preserves sparsity, the result is a sparse matrix. As above, each component of `L` can be computed in a similar fashion and summed to form the operator matrix.

Table 6.7 lists the CPU times for estimating the five parameters k_1, k_2, k_3, c_1 and c_2 of the 2D advection-diffusion problem (6.1) when the LMM PF-SMC algorithm applied to (6.4) is implemented sequentially, in parallel with 8 workers, and vectorized, respectively. The results reported were obtained using BDF2 for propagation and BDF3 for error control, computing the innovation term via the HOMEC technique, with $N = 5,000$ particles for discretization sizes $n = 20$ and $n = 40$, respectively. The

Size	Sequential	Parallel	Vectorized
$n = 20$	3.0769e+04	7.2048e+03	3.2571e+04
$n = 40$	1.6088e+05	3.3933e+04	1.6368e+05

Table 6.7: CPU times (in seconds) when applying LMM PF-SMC to the 2D advection-diffusion problem of discretization size n sequentially, in parallel with 8 workers, and vectorized using BDF2 for the particle propagation/repropagation and HOMEC innovation with time step $h = 0.1$ and $N = 5,000$ particles.

time series solution envelopes and smooth histograms of the resulting distributions for the five parameters obtained solving (6.4) sequentially when $n = 20$ are shown in Figures 6.4 and 6.5.

The results in Table 6.7 show that, for the 2D advection-diffusion problem (6.1), parallelization of the LMM PF-SMC algorithm with 8 workers yields a speedup of 4.2706 and efficiency of 0.5338 when $n = 20$, and a speedup of 4.7411 and efficiency of 0.5926 when $n = 40$. However, the vectorized implementation of the algorithm using implicit methods does not perform better than the sequential implementation when applied to this problem.

For comparison, we list in Table 6.8 the corresponding CPU times for estimating the five parameters when, instead of a fixed time step LMM, we use MATLAB's stiff solver `ode15s`, set to use BDF methods with maximum order 2 and with default relative tolerance 10^{-3}, for propagation within a similar PF-SMC algorithm, both sequentially and in parallel with 8 workers, assigning the innovation variance as the product of the relative tolerance and the number of integration steps taken by `ode15s` from one data arrival to the next. Since `ode15s` adjusts the time step as the integration proceeds, the number of steps taken may vary for each particle.

The results in Table 6.8 show that when $n = 20$ for the 2D advection-diffusion problem, propagating with `ode15s` using the BDF option and basing the innovation variance on the relative tolerance and number of integration steps yields a parallel speedup of 2.6379 and efficiency of 0.3297. Compared to the corresponding times using fixed time step LMM PF-SMC with implicit methods, while the parallel imple-

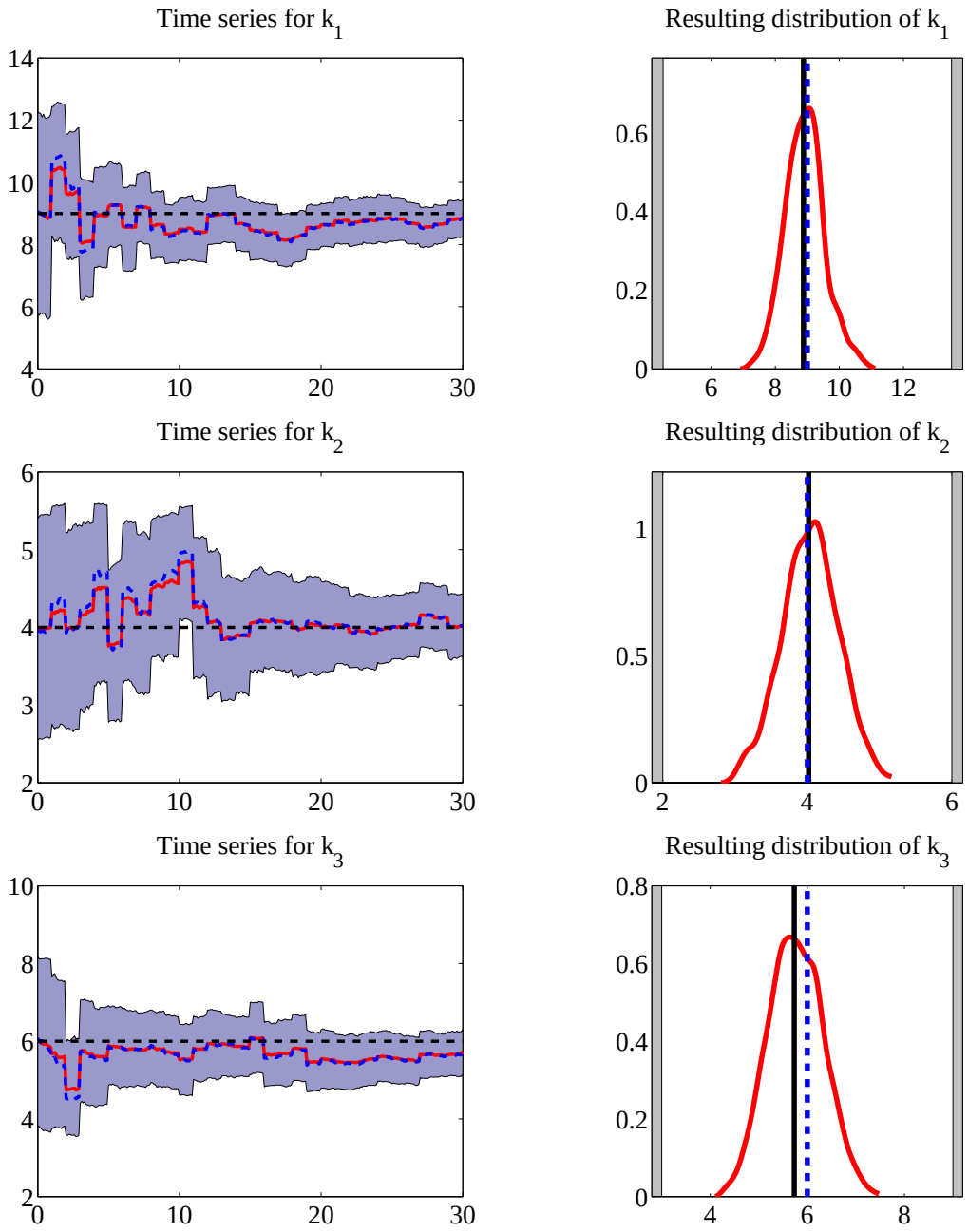

Figure 6.4: Time series estimates (left) and smooth histograms of the last posterior sample (right) of parameters k_1, k_2 and k_3 for the 2D advection-diffusion problem with discretization size $n = 20$.

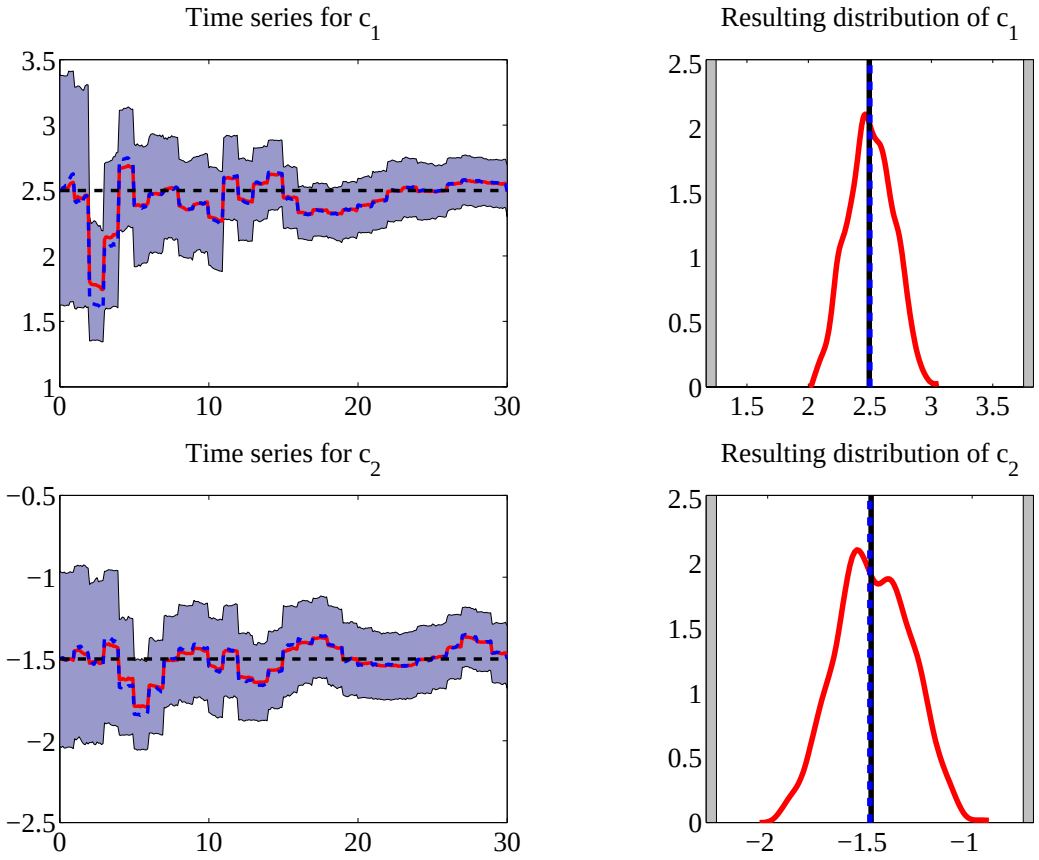

Figure 6.5: Time series estimates (left) and smooth histograms of the last posterior sample (right) of parameters c_1 and c_2 for the 2D advection-diffusion problem with discretization size $n = 20$.

Size	Sequential	Parallel
$n = 20$	1.2607e+04	4.7792e+03
$n = 40$	1.7106e+05	8.7722e+04

Table 6.8: CPU times (in seconds) when applying PF-SMC to the 2D advection-diffusion problem of discretization size n sequentially and in parallel with 8 workers using `ode15s` for the particle propagation/repropagation with $N = 5,000$ particles.

mentation using `ode15s` results in slightly faster CPU times both sequentially and in parallel with 8 workers, the resulting speedup is smaller, making it less efficient than parallel LMM PF-SMC in this sense.

Moreover, when $n = 40$ for the 2D advection-diffusion problem, the algorithm using `ode15s` with the BDF option results in slower CPU times both sequentially and in parallel with 8 workers than the corresponding times for fixed time step LMM PF-SMC with implicit methods, despite the fact that the LMM PF-SMC algorithm requires an additional propagation of all N particles per iteration in order to compute the innovation term. The parallel implementation using `ode15s` is also less efficient in this case, with a speedup of 1.9500 and efficiency of 0.2437.

We remark that when using the implicit BDF method for propagation, the full matrix `L` in (6.6) is needed for the computation of the Jacobian, which here is

```
J_val = num*EL - coeff*L;
```

where the constants `num` and `coeff` are known and depend on the BDF method used, and `EL` is the sparse identity matrix of size $N\widetilde{n}^2 \times N\widetilde{n}^2$. If we propagated the particles with an explicit AB method, the Jacobian computation would not be necessary, hence `L` would not need to be assembled. In that case, we could compute the terms of righthand side `L`U in (6.4) separately, e.g.,

```
L1  = B1B1*u_c*spdiags(d11',0,npart,npart);
```

where `u_c` is the current value of U, and then sum them together. The piecewise construction of `L`U is very efficient and leads to significant computational speedup of the vectorized LMM PF-SMC algorithm, up to 10 times faster than when `L` is required using the same time step; however, AB methods are not well-suited for the stiff problem unless a prohibitively small time step is used.

6.5 Summary of results

The use of stable, fixed time step LMM solvers in PF-SMC algorithms lends itself in a natural way to both parallelizing and vectorizing the computations, thus providing a competitive alternative to running independent parallel chains in Monte Carlo simulations [138, 139].

The results in Tables 6.1 and 6.2 for test problem (5.17) show that the parallelization of the LMM PF-SMC algorithm via parallel loops becomes efficient when using 8 workers, and more so when propagating $N = 50,000$ particles than $N = 5,000$ particles. This could be due to the overhead of the communication time between workers. Not surprisingly, it takes more time to propagate with the implicit (AM and BDF) solvers than with the explicit (AB) ones, and the CPU time also increases with the order of the method.

Tables 6.3 and 6.4 show that vectorizing the LMM PF-SMC algorithm for problem (5.17) shows significant speedup over the sequential and even parallel implementations. For example, when propagating the particles with AM3, the vectorized version is about 13.5 times faster than the sequential and 3.8 times faster than the parallel when $N = 50,000$, and about 90.2 times faster than the sequential and 28.7 times faster than the parallel when $N = 5,000$. Moreover, for the vectorized version the CPU times when using implicit and explicit methods are closer, and increasing the order of the method has little effect on the CPU time. The plots in Figure 6.2 show a log-linear relationship between the CPU time and the number of particles.

The vectorized implementation of LMM PF-SMC using implicit methods does not perform better than the sequential implementation when applied to the 2D advection-diffusion problem (6.1), as shown in Table 6.7. For that example, however, parallelization of the algorithm with 8 workers yields a speedup of 4.2706 and efficiency of 0.5338 when $n = 20$, and a speedup of 4.7411 and efficiency of 0.5926 when $n = 40$.

The results in Table 6.8 show that when $n = 20$ for the 2D advection-diffusion problem, propagating with `ode15s` using the BDF option and basing the innovation

variance on the relative tolerance and number of integration steps produces slightly faster CPU times both sequentially and in parallel with 8 workers than the corresponding times using fixed time step LMM PF-SMC with implicit methods. However, the parallel implementation using `ode15s` results in a speedup of 2.6379 and efficiency of 0.3297, making it less efficient than parallel LMM PF-SMC in this sense.

Furthermore, when $n = 40$ for the 2D advection-diffusion problem, the algorithm using `ode15s` with the BDF option results in slower CPU times both sequentially and in parallel with 8 workers than the corresponding times for fixed time step LMM PF-SMC with implicit methods, despite the additional propagation of all N particles per iteration required by the LMM PF-SMC algorithm in order to compute the innovation term. The parallel implementation using `ode15s` is also less efficient in this case, with a speedup of 1.9500 and efficiency of 0.2437.

Chapter 7

Application of LMM PF-SMC to Dynamic PET

In this chapter, we describe the application of a variant of the LMM PF-SMC algorithm for combined parameter and state estimation proposed in [10] for estimating the unknown parameters of a model of tracer kinetics from sequences of real positron emission tomography (PET) scan data, which is presented in the article [140]. The parameters to be estimated are important in understanding the neurochemistry and metabolic processes of the brain, and their distributions for different cohorts may carry valuable information when assessing the state of an individual's brain function from measured data. As will be detailed in the sections that follow, we combine standard optimization and statistical inference techniques in a novel way within a vectorized version of the LMM PF-SMC algorithm. The original LMM PF-SMC algorithm is also modified to use variable time steps to account for the increase in interval length between data measurements from the beginning to the end of the procedure, while keeping the time step the same for each particle.

We describe the mathematical compartment model and parameters of interest in Section 7.1, and we formulate the problem in a Bayesian framework in Section 7.2. The modifications made to the LMM PF-SMC algorithm are described in Section 7.3, and the results of computed examples with real data are presented in Section 7.4.

This chapter follows closely the manuscript [140].

7.1 The tracer kinetics problem

Studies of metabolic processes in brain routinely rely on indirect measurements, such as dynamic PET recordings. Although PET imaging, that is, the passage from photon counting data to activity map in the brain, is a well established practice, the process of obtaining quantitative kinetic model parameters from imaging data is less well developed and understood. In vivo neurodynamics is usually described in terms of parametric compartment models [141], and the distribution of parameter values carries important information about potential diseased states of the brain; see, e.g., [142, 143]. The estimation of parameters for the kind of parametric models used to study tracer kinetics from PET data counts can be approached from a standard optimization point of view, by fitting the model to the observations, or by reframing it as a problem of statistical inference. The advantage of the latter approach is that it naturally provides a framework for including uncertainties in the modeling paradigm, including model inputs such as the arterial tracer concentration and measurement errors. As discussed in earlier chapters, the Bayesian statistical framework allows qualitative complementary information to be encoded in the prior distribution, and it produces an output naturally equipped with a reliability estimate. Our analysis of the quantitative PET imaging problem uses sequential Bayesian estimation employing a new variant of the LMM PF-SMC algorithm.

More specifically, we consider a kinetic model of [1$-^{11}$C]-acetate imaging, which carries important information about the metabolism of astrocytes in brain [144, 145, 146]. Experimental data collection and the associated inverse problem can be summarized as follows: A radioactive tracer is administered into the blood stream of the subject, and the tracer concentration in arterial blood is estimated from direct arterial measurements of the radioactivity. The blood flow carries the tracer to the brain, where it enters the tissue, and the time-dependent positron emission activity, proportional to the tracer mass, is measured with a PET device and mapped to a

series of tomographic images, yielding an estimate of the time-dependent activity distribution in the tissue.

The mathematical compartment model describing the mass balance of the tracer in the brain tissue is governed by a system of differential equations [141] of the form

$$\frac{dm_1}{dt}(t) = K_1 c(t) - (k_2 + k_3) m_1(t) \tag{7.1}$$

$$\frac{dm_2}{dt}(t) = k_3 m_1(t) - k_5 m_2(t) \tag{7.2}$$

where $c(t)$ is the arterial tracer concentration, or *arterial input function* (AIF), estimated from measured blood samples drawn during the measurement process, and $m_1(t)$ and $m_2(t)$ represent the mass of tracer content in the precursor and product compartments, respectively. The coefficients k_2, k_3 and k_5 are transport rates of the tracer between the compartments, and K_1 is the clearance of the tracer from blood to tissue, weighted with the capillary compartment's virtual volume V_0. In this model, k_4, which would correspond to back-transport to the precursor compartment, is assumed to be vanishingly small because of the unidirectionality of the biochemical reaction represented by k_3. A schematic representation of the compartment model is shown in Figure 7.1. The initial value of the solution is $m_1(0) = m_2(0) = 0$, and the PET image provides an estimate of the time course of the total tracer content in the tissue volume of interest,

$$m(t) = V_0 c(t) + m_1(t) + m_2(t). \tag{7.3}$$

This estimate, which is refered to as the *time-activity curve* (TAC) of the tissue, is the data for the present problem. Our objective is to estimate the model parameters V_0, K_1 and k_ℓ, where $\ell = 2, 3, 5$, from the PET image data using a variant of the LMM PF-SMC method.

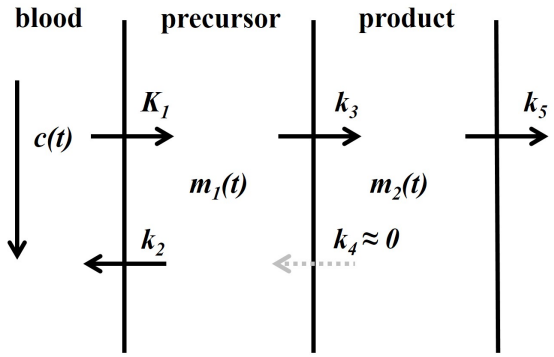

Figure 7.1: Schematic representation of the compartment model describing the mass balance of the tracer in the brain tissue, governed by equations (7.1)–(7.2).

7.2 PET in a Bayesian framework

Here we begin by introducing some notation. The unknown parameters describing the tracer dynamics in tissue are collected in the vector θ,

$$\theta = (V_0, K_1, k_2, k_3, k_5) \in \mathbb{R}^5.$$

The input function $c(t)$ obtained from plasma radioactivity measurements is assumed to be known. Integrating the system (7.1)–(7.2) starting with vanishing initial values gives the pair $(m_1(t,\theta), m_2(t,\theta))$ and the idealized model for the errorless observation,

$$m(t,\theta) = \theta_1 c(t) + m_1(t,\theta) + m_2(t,\theta), \tag{7.4}$$

while the observed data consist of noisy samples at discrete times t_j,

$$d_j = m(t_j,\theta) + e_j, \quad 1 \le j \le T, \tag{7.5}$$

where, for simplicity, the noise e_j is modeled as additive error. Figure 7.2 shows a plot of the input function and observed data for one patient, where the observed data is scaled so that its peak height matches that of the input function.

The vector e_j represents the aggregation of measurement errors, imprecisions in the model as well as model discrepancy. Measurement error in the context of dynamic PET imaging pertains to the photon emission of the volume element in

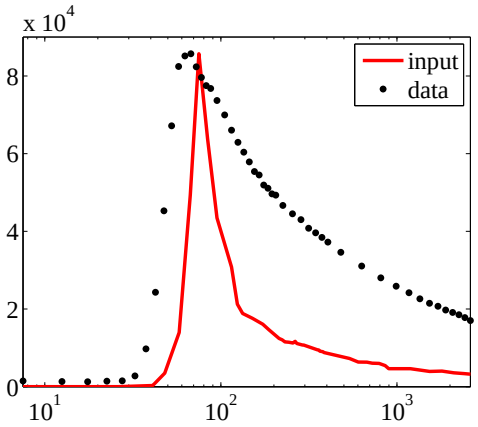

Figure 7.2: The input function and observed data for one of the cirrhotic liver patients from the study, where the observed data is scaled by a factor of ≈ 5.5, so that its peak height matches that of the input function. Note that the time axis is plotted in log-scale.

question, calculated from the PET counting data. By imprecisions in the model we refer to uncertainties that have an effect on reliability of the estimates, most notably uncertainties related to the AIF; see, e.g., [147, 148, 149, 150, 151].

In the present application, we address one possible error source, related to a incompletely known offset of the AIF and TAC curves, as the blood sample data is collected independently of the PET scan, and the delay between the time when the tracer arrives at the brain and at the blood sample site varies. This delay, which can be seen in Figure 7.2, may affect, in particular, the estimates based on data collected at the very beginning of the estimation process.

We address this issue by assuming that the convection time from the brain to the site where the blood sample is drawn contains an unknown estimation error τ, and we include this error in the model by replacing (7.5) with

$$d_j = m(t_j + \tau, \theta) + e_j, \quad j = 1, \ldots, T. \tag{7.6}$$

While the offset τ could be added as one of the unknown parameters of the model that needs to be estimated from the data, here we treat it as a nuisance parameter. To simplify the discussion, we approximate $m(t, \theta)$ by its first order Taylor polynomial

centered at (t_j, θ) so that

$$d_j \approx m(t_j, \theta) + m_t(t_j, \theta)\tau + e_j = m(t_j, \theta) + n_j + e_j, \tag{7.7}$$

where m_t denotes the derivative of m with respect to t. Clearly, the offset error n_j depends on the unknown θ itself. To overcome this difficulty, we propose the following hierarchical procedure.

Assuming that $\widehat{\theta}$ represents a typical value of the parameter θ, we define a conditional prior by setting

$$\pi_{\text{prior}}(\theta_k \mid \widehat{\theta}_k) \sim \text{Uniform}(\alpha\widehat{\theta}_k, A\widehat{\theta}_k)$$

for some scalars $0 < \alpha < 1 < A$, and we approximate the offset error term by

$$n_j = m_t(t_j, \theta)\tau \approx m_t(t_j, \widehat{\theta})\tau,$$

assuming that the only source of randomness is the random offset τ, which we will model as a zero mean Gaussian random variable.

To find a typical value $\widehat{\theta}$, we consider the model (7.5), assuming that the offset vanishes. Assuming for simplicity a Gaussian measurement error with constant variance σ^2, we have the likelihood model

$$\pi(d_j \mid \theta) \propto \exp\left(-\frac{1}{2\sigma^2}|d_j - m(t_j, \theta)|^2\right),$$

and further, by assuming conditionally independent measurements, the likelihood of the full data vector d becomes

$$\pi(d \mid \theta) = \prod_{j=1}^{T} \pi(d_j \mid \theta) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{T}|d_j - m(t_j, \theta)|^2\right). \tag{7.8}$$

Introducing a weakly informative prior which imposes the nonnegativity of the parameters θ_k and some large upper bound, Θ_k, it follows that

$$\pi_{\text{prior}}(\theta) \propto \prod_{k=1}^{K} 1_{[0,\Theta_k]}(\theta_k),$$

where $1_{[0,\Theta_k]}$ is the indicator function of the interval $[0, \Theta_k]$, and the posterior density becomes

$$\pi(\theta \mid d) \propto \prod_{k=1}^{K} 1_{[0,\Theta_k]}(\theta_k) \exp\left(-\frac{1}{2\sigma^2} \sum_{j=1}^{T} |d_j - m(t_j, \theta)|^2 \right). \tag{7.9}$$

Maximizing the posterior density is tantamount to solving for the constrained least squares solution,

$$\widehat{\theta} = \arg\min_{\theta} \left\{ \sum_{j=1}^{T} |d_j - m(t_j, \theta)|^2 \right\} \quad \text{subject to} \quad 0 \leq \theta_k \leq \Theta_k. \tag{7.10}$$

The approximate solution to this constrained nonlinear minimization problem can be found via a quasi-Newton algorithm with bound constraints, which will be described in Section 7.4.1. We note that estimating hyperparameters from data is called *empirical Bayes*.

7.3 Modified LMM PF-SMC

To assess the sensitivity of the θ estimated by solving (7.10) to modeling errors, a more comprehensive analysis is necessary. An extensive MCMC run for sampling the posterior is one option; a less thorough, but computationally less demanding alternative is to use parametric bootstrapping, which explores the posterior density assuming that the prior is non-informative. The alternative approach which we advocate here is based on a variant of the LMM PF-SMC algorithm for combined parameter and state estimation, described in Algorithm 8, joined together with the following error remodeling approach:

1. Estimate θ using the simple likelihood model (7.8), and compute $\widehat{\theta}$ by solving (7.10).

2. Write a conditional prior for each component θ_k, of the form

$$\pi_{\text{prior}}(\theta_k \mid \widehat{\theta}_k) \sim \text{Uniform}(\alpha\widehat{\theta}_k, A\widehat{\theta}_k)$$

for some scalars $0 < \alpha < 1 < A$, and the conditional likelihood

$$\pi(d_j \mid \theta, \widehat{\theta}) \sim \mathcal{N}(m(t, \theta), \sigma^2 + m_t(t_j, \widehat{\theta})^2 \mathrm{var}(\tau)).$$

3. Use the modified LMM PF-SMC algorithm to explore the conditional posterior density,

$$\pi(\theta \mid d, \widehat{\theta}) \propto \pi_{\mathrm{prior}}(\theta \mid \widehat{\theta}) \pi(d \mid \theta, \widehat{\theta}).$$

As mentioned above, changes to the LMM PF-SMC algorithm were made for the current application to meet the challenges posed by the initial, very rapid ramping of the tracer's mass and the 100 fold increase in the time intervals between data measurements from the beginning to the end of the procedure, as seen, e.g., in Figure 7.2. More precisely, instead of keeping the time step fixed to the same value for each particle for the entire duration of the experiment, we adjust the time step according to the frequency of the data collection, keeping it the same for all particles. Specifically, when propagating with an r-step LMM, we compute the maximum possible step size between consecutive data points, i.e., if T_j is the length of the time interval between data arrivals at times t_{j-1} and t_j, we set the maximum step size $h_j = T_j/r$. We initialize the time step to the maximum value and compute the corresponding LMM coefficients in the manner described in Section 2.4. If necessary, the step size may decreased for stability purposes, and the LMM coefficients recomputed in accordance. We emphasize that while the step size may change as the algorithm proceeds, it is always kept the same for each particle.

Before the algorithm begins, we adjust the data by shifting them down by a factor $\eta = \min\{d_1, d_2, d_3\}$ to account for noise in the measurement instrument estimated using the data prior to the arrival of the tracer. During the steep up-ramping phase of the experiment, the PRR may become prohibitively low: To avoid a too drastic impoverishment of the particle pool, if the PRR drops below 60%, we resample until the PRR is above the threshold or we reach a maximum number of regenerations.

The complexity of the problem and the need for a large number of particles make the use of the PF-SMC algorithm computationally demanding. It was shown in

Chapter 6 that it is possible to reformulate the problem in so as to take advantage of vectorized and parallelized computing environments, significantly reducing the computing time. The results in Section 7.4.2 were obtained using a vectorized version of the modified LMM PF-SMC algorithm just described.

7.4 Results

The algorithm was applied to data collected from a cohort of 18 subjects from three different groups: a healthy control group (HC) of five individuals, a group of seven individuals suffering from cirrhotic liver (CL), and a group of six individuals with the condition of hepatic encephalopathy (HE). The motivation for the study was to determine whether the liver condition of the patients in the last two groups, through a perturbed ammonium level in their circulation, significantly affected the astrocytic metabolism at the level of acetate uptake and clearance. Mathematically, we aim to distinguish any significant differences between the values of the model parameters V_0, K_1 and k_ℓ, $\ell = 2, 3, 5$, as described in Section 7.1, for the three groups.

The details about the cohort of patients, the scanning procedures and the reconstruction of the data can be found in [143, 152]. The [1$-^{11}$C]-acetate PET protocol lasted 45 minutes, and the data corresponds to 50 time frames, with the interval between them increasing in time: every 5 seconds for the first 18 frames, every 10 seconds for the next 12 frames, then every 30 seconds for the subsequent 7 frames, followed by one frame after 120 seconds, and finishing with 12 frames every 180 seconds. During the [1$-^{11}$C]-acetate recording, arterial blood samples of 0.5 ml each were collected manually at the following intervals: every 5 seconds for the first 18 times, every 10 seconds for the next 9 times, then every 60 seconds for the subsequent 12 times, and finally every 300 seconds for the last 6 times. Total blood ^{11}C radioactivity concentrations were measured using a well counter (Packard Instrument Company, Meriden, CT, USA), cross-calibrated with the tomograph, and corrected for radioactive decay to the start of the scan. The volume of interest in our computed examples is that of the whole brain, and the corresponding time-activity

curves were extracted from the dynamic PET scan. The study was performed in accordance with the Helsinki II Declaration and approved by the ethics committee of Aarhus County. Informed consent was obtained from each subject. No complications were observed during the procedures.

We label the 18 patients with an index p, $1 \leq p \leq 18$, and we identify the index sets with the corresponding group, writing

$$\mathrm{HC} = \{p : 1 \leq p \leq 5\}, \quad \mathrm{CL} = \{p : 6 \leq p \leq 12\}, \quad \mathrm{HE} = \{p : 13 \leq p \leq 18\}.$$

For each individual p, we record the optimized estimate and the parameter sample with corresponding weights as

$$\widehat{\theta}_{(p)}, \quad (\theta^n_{(p)}, w^n_{(p)}), \quad n = 1, \ldots, N.$$

7.4.1 Optimized hyperparameters

In Section 7.2, we formulated a Bayesian hierarchical update (7.9) for the parameter vector θ, where the hyperparameter $\widehat{\theta}$ is the solution to the constrained nonlinear least squares problem (7.10). We compute $\widehat{\theta}_{(p)}$ for each patient, $1 \leq p \leq 18$, by solving (7.10) via a bound-constrained quasi-Newton global optimization algorithm, which fits the parameter values to the data by minimizing the sum of the squared residuals. The details of this type of algorithm can be found, e.g., in [76, 77]. Here we compute the Jacobian using the known form of the problem, and we employ a combination of global convergence strategies, such as a backtracking line-search and a model trust region approach, when the Gauss-Newton step is rejected by failing to satisfy the Armijo condition [153].

Table 7.1 lists the resulting optimized parameter estimates for each patient, sorted by group. Since the aim of our analysis is to find whether there are any significant differences in parameter values between the three patient groups, we use data mining techniques to group the hyperparameters in order to see if any preliminary distinctions are visible. In particular, we make use of k-means groupings and principal component

Patient	V_0	K_1	k_2	k_3	k_5
1	8.8914e-03	9.1126e-03	2.4863e-02	5.7240e-03	5.0372e-03
2	2.7843e-02	8.8373e-03	2.3209e-02	3.4826e-03	3.0598e-03
3	9.6508e-02	9.7631e-03	1.8257e-02	2.2531e-03	2.9420e-03
4	8.3579e-02	5.6257e-03	1.3614e-02	1.7553e-03	1.6701e-03
5	7.4945e-02	3.4092e-03	1.2487e-02	4.3138e-03	1.3619e-03
6	6.2424e-02	8.9456e-03	2.2924e-02	2.5699e-03	3.2486e-03
7	8.6474e-02	6.3450e-03	1.3867e-02	2.0623e-03	2.0401e-03
8	1.2194e-01	6.8327e-03	1.5541e-02	1.3862e-03	2.0483e-03
9	7.9554e-02	4.6056e-03	1.2285e-02	2.4207e-03	1.3916e-03
10	4.5701e-02	6.6120e-03	2.1954e-02	3.5441e-03	2.5974e-03
11	1.6253e-01	3.7216e-03	1.1151e-02	1.5924e-03	1.3767e-03
12	6.8065e-02	6.7645e-03	1.6687e-02	1.9993e-03	2.2676e-03
13	4.4654e-02	4.4032e-03	1.1005e-02	1.3250e-03	1.2227e-03
14	1.6110e-02	4.2687e-03	1.0831e-02	1.6558e-03	1.9719e-03
15	3.1157e-02	4.6739e-03	1.1737e-02	2.1741e-03	1.8991e-03
16	2.7991e-02	2.7334e-03	1.6681e-02	5.5866e-03	9.8447e-04
17	3.9232e-02	3.3517e-03	1.0546e-02	3.5964e-03	1.6859e-03
18	4.6142e-02	4.4364e-03	1.0484e-02	1.5789e-03	1.8681e-03

Table 7.1: The optimized parameter results for each patient organized by group. Patients 1 through 5 correspond to the HC group, 6 through 12 to the CL group, and 13 through 18 to the HE group.

analysis (PCA) techniques; see [154] for details. Figures 7.3, 7.4 and 7.5 show the scatter plots for each of the ten possible pairings of parameters projected in 2-space. The leftmost plots group the values by their *a priori* known patient type, the middle by the computed 2-means groupings, and the rightmost by the computed 3-means groupings. It is interesting to note that in some of the plots, it seems more natural to group the values into 2-means groups rather than 3-means (see, e.g., k_2 vs. k_5). Figure 7.6 plots the patient data in the directions of the first and second principal components, where there is a clear separation between the HE group versus the HC and CL groups.

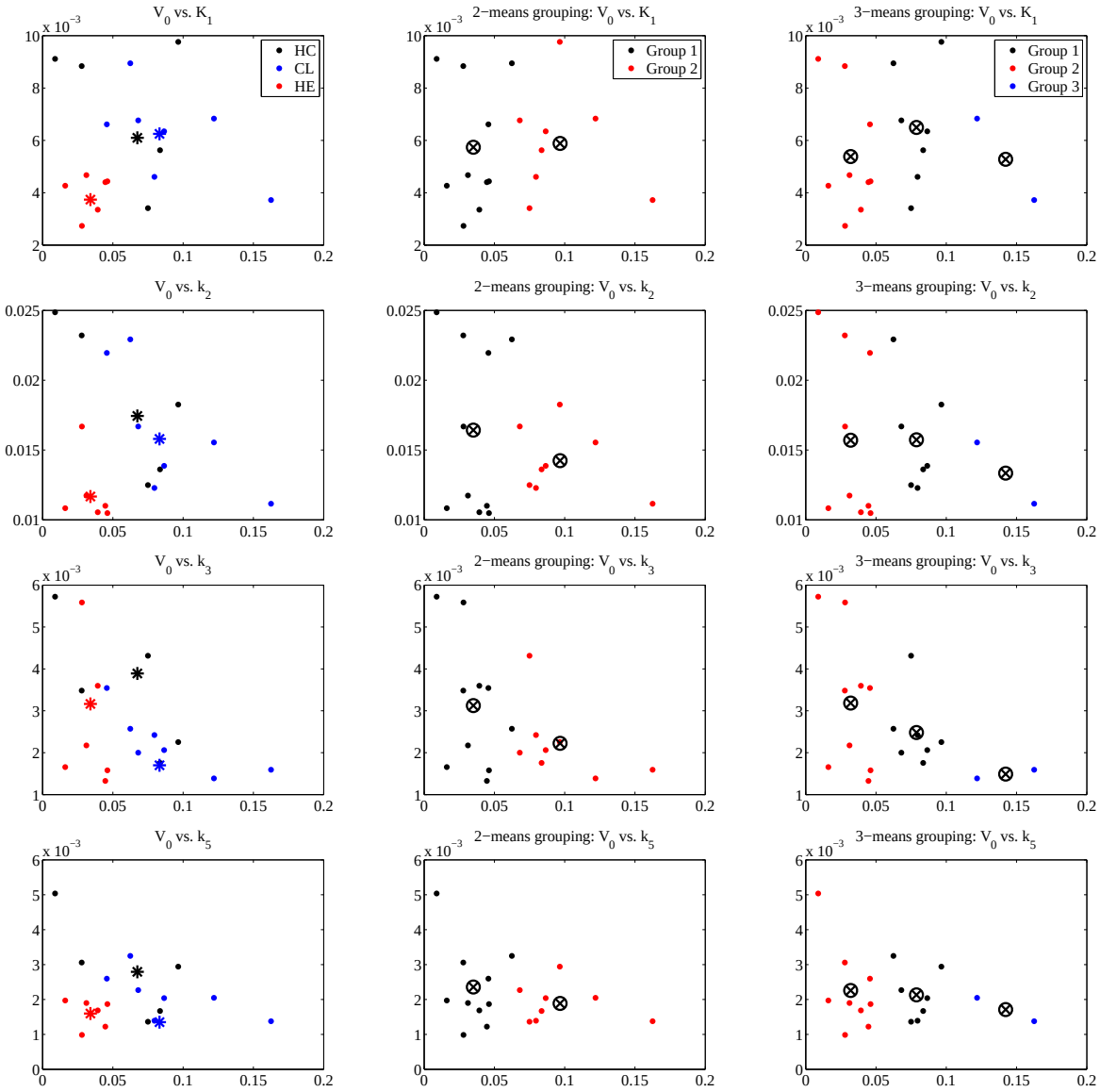

Figure 7.3: Scatter plots comparing the 2D projections of each pair of NLLS estimated parameters for the three groups of patients. The plots in the left column show the known patient-type groupings, where the optimized parameter value for each respective group is marked with an asterisk. The plots in the middle column show the computed 2-means groupings, and the plots in the right column show the computed 3-means groupings.

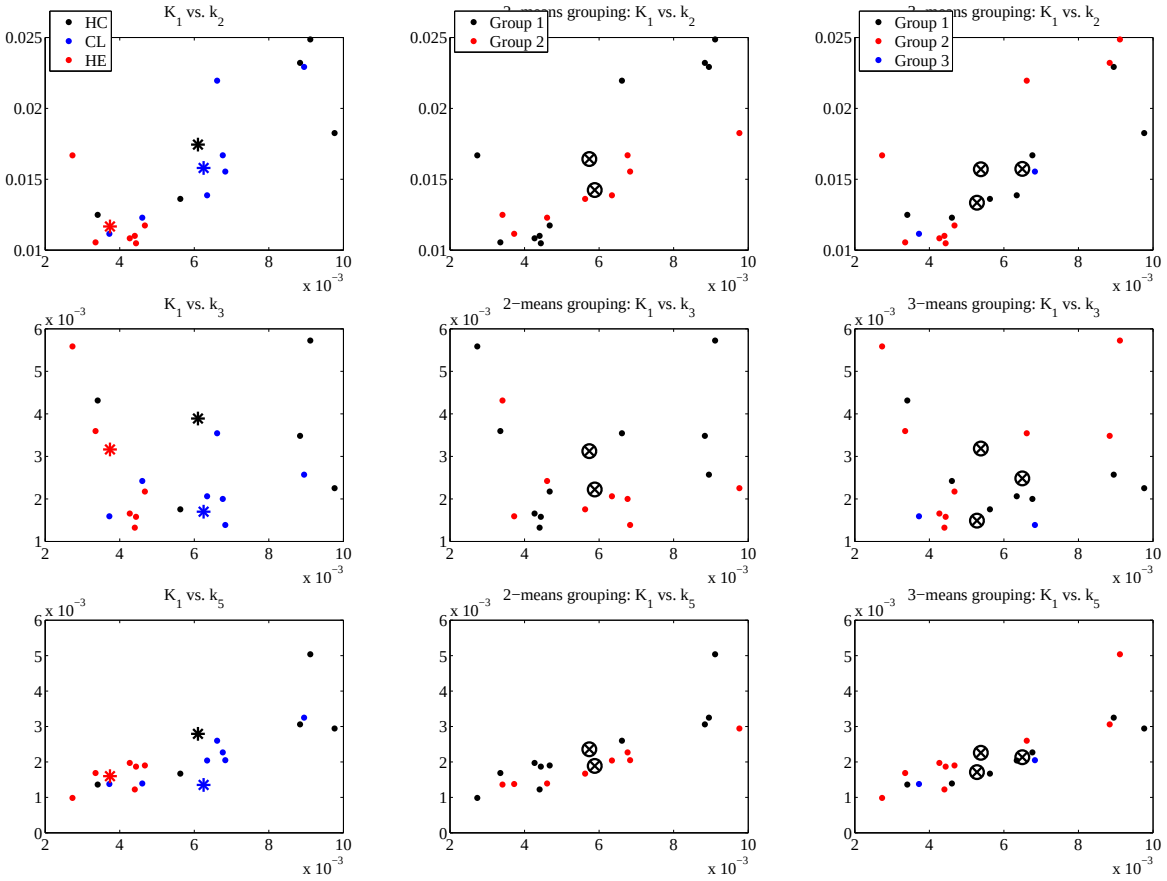

Figure 7.4: Scatter plots comparing the 2D projections of each pair of NLLS estimated parameters for the three groups of patients. The plots in the left column show the known patient-type groupings, where the optimized parameter value for each respective group is marked with an asterisk. The plots in the middle column show the computed 2-means groupings, and the plots in the right column show the computed 3-means groupings.

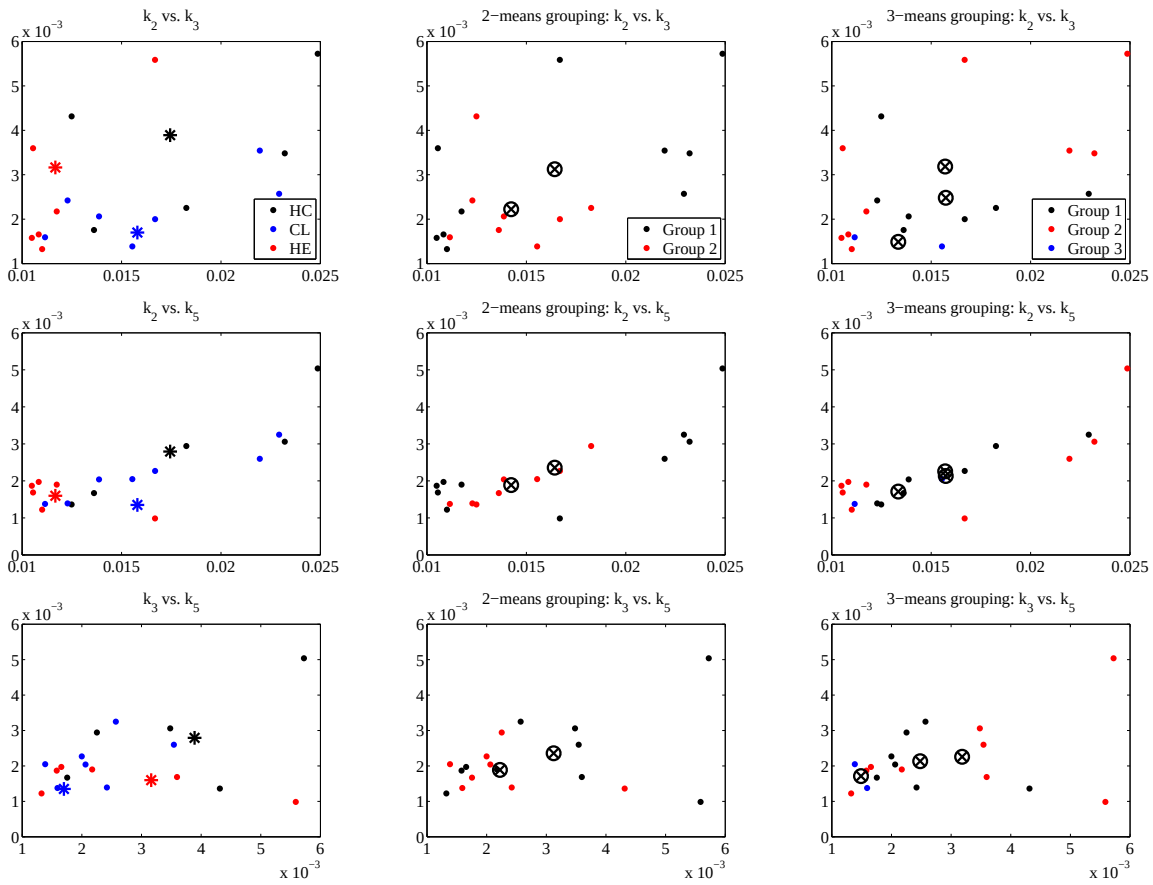

Figure 7.5: Scatter plots comparing the 2D projections of each pair of NLLS estimated parameters for the three groups of patients. The plots in the left column show the known patient-type groupings, where the optimized parameter value for each respective group is marked with an asterisk. The plots in the middle column show the computed 2-means groupings, and the plots in the right column show the computed 3-means groupings.

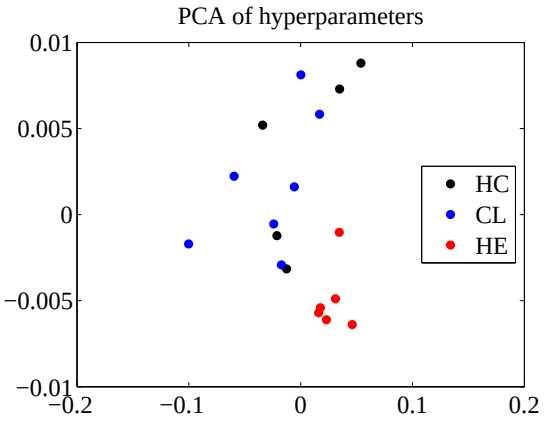

Figure 7.6: Patient data for the three respective groups plotted in the directions of the first and second principal components.

7.4.2 LMM PF-SMC estimates

In the sequential Monte Carlo approach, for each individual p we run the hierarchical, variable time step PF-SMC algorithm, filtering a sample size of $N = 100,000$ particles, using the third-order implicit Adams-Moulton scheme AM2 for the time propagation and the fourth-order implicit method AM3 for the error estimation via the HOMEC technique. The resulting distributions of the parameters are summarized in Figure 7.7, where the box plots indicate the 50% and 90% credibility intervals of the conditional distributions $\pi(\theta_{(p)} \mid \widehat{\theta}_{(p)}, d_{(p)})$, where $d_{(p)}$ is the data vector of the pth patient. The results, which are also presented in tabular form, organized by patient group with the HC patients listed in Table 7.2, the CL patients in Tables 7.3 and 7.4, and the HE patients in Tables 7.5 and 7.6, indicate that while there is no significant difference in the estimated parameters between the groups HC and CL, in HE patients the parameters related to acetate uptake, in particular, that is, V_0 and K_1, show a clear offset.

To quantify the differences between the groups beyond a mere visual inspection, we perform the following statistical test. Let X denote a chosen reference group (HC, CL or HE) and Y a selected test group. We start by fitting a Gaussian distribution $\mathcal{N}(\theta \mid \mu_X, \mathsf{C}_X)$ to the combined sample $\{\theta^n_{(p)} \mid p \in X, 1 \leq n \leq N\}$. Given the

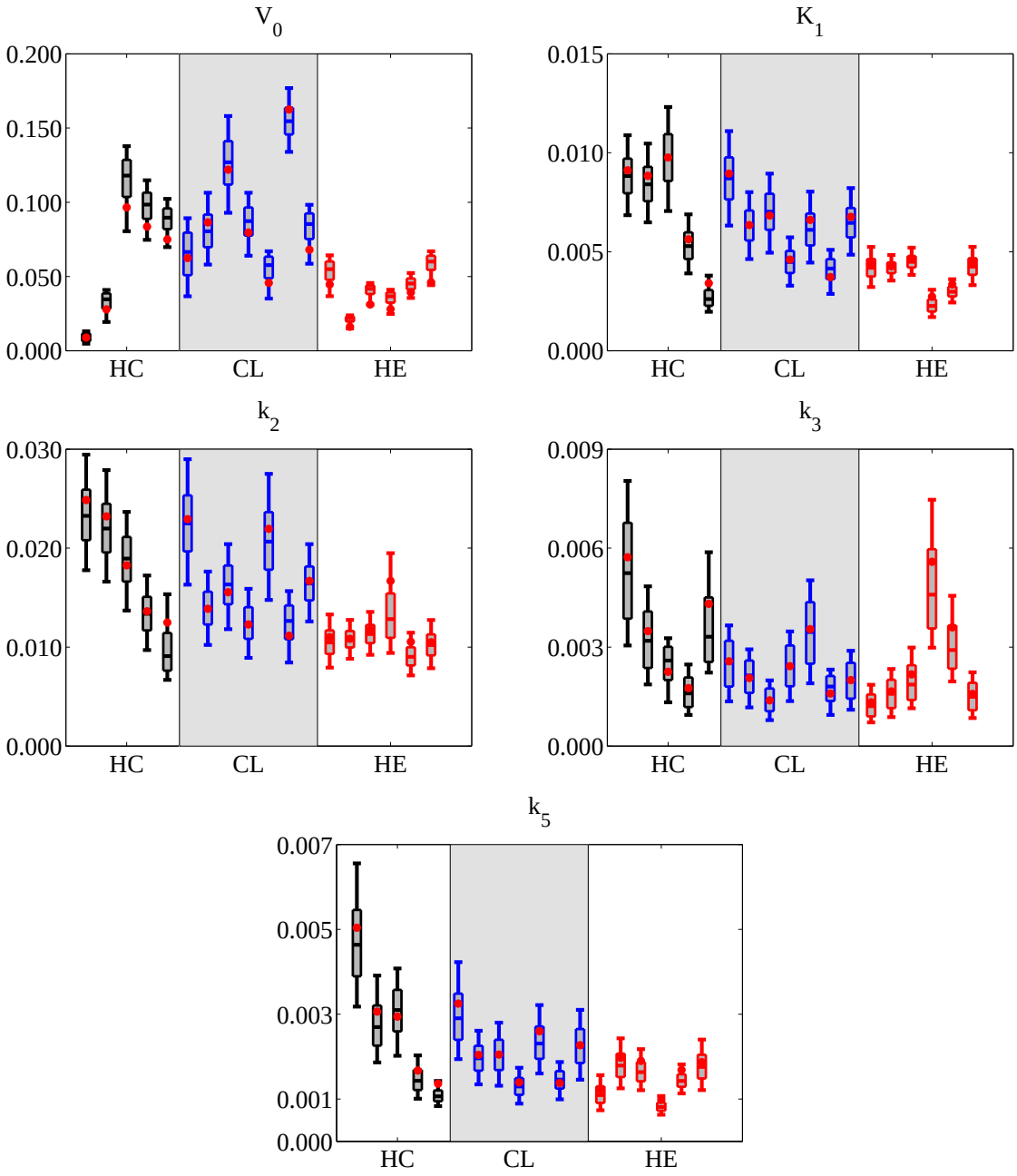

Figure 7.7: Box plots indicating the 50% and 90% credibility intervals for each patient with respect to the five parameters. Box plots for the HC group are shown in black, CL in blue and HE in red, respectively. The red dot on each box plot marks the NLLS estimate of the corresponding parameter and patient.

Patient 1

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	8.8914e-03	9.1126e-03	2.4863e-02	5.7240e-03	5.0372e-03
Posterior mean	8.9489e-03	8.8374e-03	2.3401e-02	5.3588e-03	4.7153e-03
Upper quartile	1.1578e-02	9.7043e-03	2.5913e-02	6.7686e-03	5.4572e-03
Lower quartile	6.3371e-03	7.9528e-03	2.0803e-02	3.8626e-03	3.8935e-03

Patient 2

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	2.7843e-02	8.8373e-03	2.3209e-02	3.4826e-03	3.0598e-03
Posterior mean	3.2851e-02	8.4330e-03	2.2079e-02	3.2498e-03	2.7650e-03
Upper quartile	3.8617e-02	9.2887e-03	2.4475e-02	4.0786e-03	3.2039e-03
Lower quartile	2.8162e-02	7.5562e-03	1.9564e-02	2.3685e-03	2.2597e-03

Patient 3

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	9.6508e-02	9.7631e-03	1.8257e-02	2.2531e-03	2.9420e-03
Posterior mean	1.1467e-01	9.7486e-03	1.8837e-02	2.4727e-03	3.0761e-03
Upper quartile	1.2852e-01	1.0940e-02	2.1130e-02	3.0072e-03	3.5728e-03
Lower quartile	1.0360e-01	8.5796e-03	1.6624e-02	2.0034e-03	2.5951e-03

Patient 4

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	8.3579e-02	5.6257e-03	1.3614e-02	1.7553e-03	1.6701e-03
Posterior mean	9.7014e-02	5.3257e-03	1.3399e-02	1.6429e-03	1.4641e-03
Upper quartile	1.0644e-01	5.9777e-03	1.5076e-02	2.0786e-03	1.6888e-03
Lower quartile	8.8837e-02	4.6361e-03	1.1686e-02	1.1827e-03	1.2124e-03

Patient 5

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	7.4945e-02	3.4092e-03	1.2487e-02	4.3138e-03	1.3619e-03
Posterior mean	8.8266e-02	2.7094e-03	9.8134e-03	3.6135e-03	1.0863e-03
Upper quartile	9.5864e-02	3.0568e-03	1.1424e-02	4.5033e-03	1.2022e-03
Lower quartile	8.1943e-02	2.2705e-03	7.6249e-03	2.5508e-03	9.4770e-04

Table 7.2: LMM PF-SMC parameter results for each patient $p = 1, \ldots, 5$ in the HC group, labeled by patient number. The optimized parameter value (computed via NLLS) as well as the posterior mean of the filtered sample distribution and the upper (75%) and lower (25%) quartiles are given.

Patient 6

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	6.2424e-02	8.9456e-03	2.2924e-02	2.5699e-03	3.2486e-03
Posterior mean	6.4943e-02	8.7107e-03	2.2542e-02	2.5062e-03	2.9698e-03
Upper quartile	7.9495e-02	9.7667e-03	2.5332e-02	3.1902e-03	3.4833e-03
Lower quartile	5.0891e-02	7.6447e-03	1.9674e-02	1.8031e-03	2.3962e-03

Patient 7

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	8.6474e-02	6.3450e-03	1.3867e-02	2.0623e-03	2.0401e-03
Posterior mean	8.1059e-02	6.3240e-03	1.3928e-02	2.0990e-03	1.9654e-03
Upper quartile	9.1732e-02	7.0845e-03	1.5591e-02	2.5934e-03	2.2537e-03
Lower quartile	6.9716e-02	5.5673e-03	1.2283e-02	1.6111e-03	1.6681e-03

Patient 8

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	1.2194e-01	6.8327e-03	1.5541e-02	1.3862e-03	2.0483e-03
Posterior mean	1.2634e-01	6.9997e-03	1.6248e-02	1.3980e-03	2.0432e-03
Upper quartile	1.4114e-01	7.9292e-03	1.8245e-02	1.7431e-03	2.3962e-03
Lower quartile	1.1194e-01	6.1090e-03	1.4336e-02	1.0566e-03	1.6836e-03

Patient 9

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	7.9554e-02	4.6056e-03	1.2285e-02	2.4207e-03	1.3916e-03
Posterior mean	8.6554e-02	4.4903e-03	1.2433e-02	2.4304e-03	1.3011e-03
Upper quartile	9.6469e-02	5.0371e-03	1.4032e-02	3.0483e-03	1.4936e-03
Lower quartile	7.7539e-02	3.9271e-03	1.0840e-02	1.8147e-03	1.0996e-03

Patient 10

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	4.5701e-02	6.6120e-03	2.1954e-02	3.5441e-03	2.5974e-03
Posterior mean	5.5152e-02	6.1530e-03	2.0826e-02	3.4431e-03	2.3465e-03
Upper quartile	6.3375e-02	6.9295e-03	2.3644e-02	4.3590e-03	2.7102e-03
Lower quartile	4.8836e-02	5.3202e-03	1.7818e-02	2.4984e-03	1.9483e-03

Table 7.3: LMM PF-SMC parameter results for patients $p = 6, \ldots, 10$ in the CL group, labeled by patient number. The optimized parameter value (computed via NLLS) as well as the posterior mean of the filtered sample distribution and the upper (75%) and lower (25%) quartiles are given.

Patient 11

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	1.6253e-01	3.7216e-03	1.1151e-02	1.5924e-03	1.3767e-03
Posterior mean	1.5490e-01	4.0900e-03	1.2413e-02	1.7288e-03	1.4476e-03
Upper quartile	1.6371e-01	4.6239e-03	1.4211e-02	2.1220e-03	1.6551e-03
Lower quartile	1.4573e-01	3.6094e-03	1.0790e-02	1.3669e-03	1.2471e-03

Patient 12

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	6.8065e-02	6.7645e-03	1.6687e-02	1.9993e-03	2.2676e-03
Posterior mean	8.2760e-02	6.4842e-03	1.6449e-02	1.9803e-03	2.2527e-03
Upper quartile	9.2506e-02	7.2070e-03	1.8144e-02	2.5240e-03	2.6489e-03
Lower quartile	7.5241e-02	5.7296e-03	1.4722e-02	1.4388e-03	1.8481e-03

Table 7.4: LMM PF-SMC parameter results for patients $p = 11, 12$ in the CL group, labeled by patient number. The optimized parameter value (computed via NLLS) as well as the posterior mean of the filtered sample distribution and the upper (75%) and lower (25%) quartiles are given.

parameter $\theta_{(p)}$ for an individual from the test group, $p \in Y$, we can evaluate the mean likelihood, averaged over the conditional posterior,

$$\begin{aligned} L(p \mid X, \widehat{\theta}_{(p)}) &= \int \mathcal{N}(\theta_{(p)} \mid \mu_X, \mathsf{C}_X)\pi(\theta_{(p)} \mid \widehat{\theta}_{(p)}, d_{(p)})d\theta_{(p)} \\ &\approx \sum_{n=1}^{N} w_{(p)}^n \mathcal{N}(\theta_{(p)}^n \mid \mu_X, \mathsf{C}_X). \end{aligned}$$

This quantity expresses the average likelihood of getting the observation $\theta_{(p)}$ if it would come from the distribution calculated from the reference group data.

To average over the test group, we fit the sample $\{\widehat{\theta}_{(p)} \mid p \in Y\}$ with a uniform distribution, which we assume represents the distribution of the hyperparameter $\widehat{\theta}$ over the test group Y, denoted $\pi(\widehat{\theta} \mid Y)$. The uniform distribution is justified by the small sample size and the rejection of a multivariate normal distribution via Mardia's test for skewness and kurtosis; see [155, 156] for test details. The likelihood averaged

Patient 13

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	4.4654e-02	4.4032e-03	1.1005e-02	1.3250e-03	1.2227e-03
Posterior mean	5.3271e-02	4.2031e-03	1.0523e-02	1.2412e-03	1.1120e-03
Upper quartile	6.0117e-02	4.6359e-03	1.1689e-02	1.5647e-03	1.2983e-03
Lower quartile	4.7824e-02	3.7525e-03	9.3077e-03	9.0020e-04	9.0731e-04

Patient 14

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	1.6110e-02	4.2687e-03	1.0831e-02	1.6558e-03	1.9719e-03
Posterior mean	2.0956e-02	4.1881e-03	1.0793e-02	1.5888e-03	1.8064e-03
Upper quartile	2.3107e-02	4.4592e-03	1.1625e-02	2.0094e-03	2.0693e-03
Lower quartile	1.9617e-02	3.9188e-03	9.9625e-03	1.1517e-03	1.5186e-03

Patient 15

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	3.1157e-02	4.6739e-03	1.1737e-02	2.1741e-03	1.8991e-03
Posterior mean	4.0486e-02	4.5001e-03	1.1340e-02	1.9457e-03	1.6556e-03
Upper quartile	4.3950e-02	4.7844e-03	1.2243e-02	2.4494e-03	1.8711e-03
Lower quartile	3.8186e-02	4.2048e-03	1.0404e-02	1.3962e-03	1.4190e-03

Patient 16

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	2.7991e-02	2.7334e-03	1.6681e-02	5.5866e-03	9.8447e-04
Posterior mean	3.5249e-02	2.3085e-03	1.3433e-02	4.8445e-03	8.2596e-04
Upper quartile	3.9306e-02	2.5910e-03	1.5423e-02	5.9660e-03	9.1559e-04
Lower quartile	3.2354e-02	1.9808e-03	1.0935e-02	3.5671e-03	7.2420e-04

Table 7.5: LMM PF-SMC parameter results for patients $p = 13, \ldots, 16$ in the HE group, labeled by patient number. The optimized parameter value (computed via NLLS) as well as the posterior mean of the filtered sample distribution and the upper (75%) and lower (25%) quartiles are given.

Patient 17

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	3.9232e-02	3.3517e-03	1.0546e-02	3.5964e-03	1.6859e-03
Posterior mean	4.4772e-02	2.9977e-03	9.1167e-03	3.0473e-03	1.4430e-03
Upper quartile	4.8628e-02	3.2354e-03	9.9861e-03	3.6583e-03	1.5816e-03
Lower quartile	4.1349e-02	2.7418e-03	8.1459e-03	2.3450e-03	1.2868e-03

Patient 18

Parameter	V_0	K_1	k_2	k_3	k_5
Optimized value	4.6142e-02	4.4364e-03	1.0484e-02	1.5789e-03	1.8681e-03
Posterior mean	5.8406e-02	4.2594e-03	1.0239e-02	1.5104e-03	1.7735e-03
Upper quartile	6.3940e-02	4.6723e-03	1.1291e-02	1.9177e-03	2.0431e-03
Lower quartile	5.4438e-02	3.8351e-03	9.1618e-03	1.0871e-03	1.4834e-03

Table 7.6: LMM PF-SMC parameter results for patients $p = 17, 18$ in the HE group, labeled by patient number. The optimized parameter value (computed via NLLS) as well as the posterior mean of the filtered sample distribution and the upper (75%) and lower (25%) quartiles are given.

over the combined posterior density,

$$\begin{aligned} L(Y,X) &= \int L(p \mid X, \widehat{\theta}_p)\pi(\widehat{\theta}_p \mid Y)d\widehat{\theta}_p \\ &\approx \frac{1}{n_Y}\sum_{p=1}^{n_Y} L(p \mid X, \widehat{\theta}_{(p)}), \end{aligned}$$

can be thought of as an average measure for the probability of the sample of group Y as a whole, assuming that it came from the distribution of the reference group X. To obtain a meaningful relative measure of how well the groups fit, we compute a version of a Bayes ratio, defining

$$R(Y,X) = \frac{L(Y,X)}{L(X,X)},$$

where a reference value $R(Y,X) = 1$ would indicate a perfect match.

Table 7.7 lists the Bayes ratios R for all possible pairs X and Y, while Table 7.8 gives the Bayes ratios comparing HC with the combined diseased group, CL + HE.

Y \ X	HC	CL	HE
HC	1.0000	0.5821	0.0042
CL	0.9591	1.0000	0.0034
HE	0.2205	0.0444	1.0000

Table 7.7: Bayes ratios for all possible pairs of control groups X and test groups Y.

Y \ X	HC	CL + HE
HC	1.0000	0.5285
CL + HE	0.6182	1.0000

Table 7.8: Bayes ratios comparing HC with the combined diseased group, CL + HE.

While these ratios do not suggest any stark differences between HC and CL or the combined diseased group, Table 7.7 shows a clear distinction between HE and the other groups, both when HE is treated as the control group and as the test group.

Chapter 8

LMM Ensemble Kalman Filter

We have seen in the previous chapters that numerical integration error estimates are a viable means of assigning the innovation variance in particle filtering algorithms. A natural question, therefore, is how the idea of linking the innovation variance with numerical integration error estimates can be applied in the context of the EnKF algorithms detailed in Section 4.4.3. It will be shown in this chapter that an approach similar to that formulated in [10] can be used to systematically estimate the variance of the innovation term in the time evolution update of the EnKF algorithm for some classes of problems. More specifically, a stochastic interpretation of the discretization error in numerical integrators can be employed to extend the technique to deterministic, large-scale nonlinear evolution models, with innovation variance based on classical error estimates. The resulting algorithm, which introduces LMM time integrators into the EnKF framework, proves especially useful in the prediction of blind system components.

This chapter is organized as follows. In Section 8.1, we derive the LMM EnKF algorithm for linear observation models, and we test its efficiency on the classic Lorenz-63 model [11] with chaotic parameters in Section 8.2. We describe a variant of the EnKF algorithm which introduces spatio-temporal priors in Section 8.3 and demonstrate its effectiveness with regards to LMM EnKF on an application to metabolism kinetics in Section 8.4. The LMM EnKF algorithm, along with its

application to estimating blind components of a two-compartment model for skeletal muscle metabolism, is presented in the manuscript [157].

8.1 The LMM EnKF algorithm

In this section we describe how to integrate the LMM-based propagation approach, described in the context of particle filters in Chapter 5, into the general EnKF framework reviewed in Section 4.4.3. Specifically, we discuss how to link the variance of the noise process W_{k+1} in the nonlinear evolution equation (4.33) to the local error of the LMM, which provides a novel and self-consistent way of estimating the approximation error. We end the section by formally stating the LMM EnKF algorithm for linear observation models.

The main modifications needed for the LMM EnKF algorithm are in the generation of the prediction ensemble (4.36) during the time evolution update: First, we specify that the nonlinear operator F_{k+1} in (4.35) is approximated using a chosen LMM time integration scheme, which we denote by Ψ, as detailed in Section 2.2. Secondly, we draw the innovation term w_{k+1}^n in (4.35) from a Gaussian distribution $\mathcal{N}(0, \mathsf{C}_{k+1}^n)$, where the covariance matrix C_{k+1}^n is estimated using an LMM error estimation procedure for each ensemble member, as described in Section 5.2. Since the implementation details of the LMM propagation and error control are similar to those described in Chapter 5, we illustrate the technique in the context of EnKF time evolution with an example.

Suppose the LMM propagator Ψ is the second-order implicit method BDF2, and we wish to estimate the time evolution innovation variance using the HOMEC technique, as described in Section 2.3.1. Assume that at the current time k, the state ensemble is given by

$$\mathcal{S}_{k|k} = \left\{x_{k|k}^1, x_{k|k}^2, \ldots, x_{k|k}^N\right\},$$

where

$$x_{k|k}^n \in \mathbb{R}^d, \quad n = 1, 2, \ldots, N.$$

Each ensemble member n is propagated forward with BDF2 to obtain an LMM predictor

$$x^n_{k+1|k,\Psi} = \Psi(x^n_{k|k}), \quad n = 1, 2, \ldots, N,$$

while simultaneously performing the propagation with the third-order method BDF3, which we denote by $\widehat{\Psi}$, to get a higher order LMM predictor

$$x^n_{k+1|k,\widehat{\Psi}} = \widehat{\Psi}(x^n_{k|k}), \quad n = 1, 2, \ldots, N.$$

We estimate the covariance matrix C^n_{k+1} using the local error estimate

$$\left| x^n_{k+1|k,\Psi} - x^n_{k+1|k,\widehat{\Psi}} \right|,$$

such that

$$\mathsf{C}^n_{k+1} = \mathrm{diag}(\gamma), \quad \gamma_i = \tau^2 \left(x^n_{k+1|k,\Psi} - x^n_{k+1|k,\widehat{\Psi}} \right)^2_i, \quad i = 1, \ldots, d,$$

where $\tau > 1$ is a safeguard factor. We then draw w^n_{k+1} from the distribution $\mathcal{N}(0, \mathsf{C}^n_{k+1})$ in order to generate the prediction ensemble via the equation

$$x^n_{k+1|k} = \Psi(x^n_{k|k}) + w^n_{k+1} = x^n_{k+1|k,\Psi} + w^n_{k+1}, \quad n = 1, 2, \ldots, N.$$

Similarly, if we wish instead to use the Milne device estimate (2.27), we choose the error control propagator $\widehat{\Psi}$ to be the second-order method AM1 and set

$$\mathsf{C}^n_{k+1} = \mathrm{diag}(\gamma), \quad \gamma_i = \tau^2 \kappa^2 \left(x^n_{k+1|k,\Psi} - x^n_{k+1|k,\widehat{\Psi}} \right)^2_i, \quad i = 1, \ldots, d,$$

where the constant κ in (2.28) is computed using the local error constants of the LMMs.

Since propagating with LMM time integrators may introduce additional time steps in between datum arrivals, an observation update does not necessarily occur at every time step. Therefore, at the time steps in between observations, we assume that the prediction ensemble produced during the time evolution update is the posterior state

Algorithm 9 LMM EnKF for Linear Observation Models

Given an initial prior distribution $\pi(x_0) = \pi(x_0 \mid D_0)$:

1. *Initialization*: Draw the initial state ensemble
$$\mathcal{S}_0 = \mathcal{S}_{0|0} = \left\{x_{0|0}^1, x_{0|0}^2, \ldots, x_{0|0}^N\right\}$$
from $\pi(x_0)$. Set $k = 0$.
2. *Prediction step*: Using the current state ensemble $\mathcal{S}_{k|k}$,
   (a) Propagate $\mathcal{S}_{k|k}$ using the LMM time integrator Ψ,
   $$x_{k+1|k,\Psi}^n = \Psi(x_{k|k}^n, h), \quad n = 1, 2, \ldots, N.$$
   (b) Using the LMM error estimate, compute
   $$\mathsf{C}_{k+1}^n = \mathsf{C}_{k+1}(x_{k|k}^n), \quad n = 1, 2, \ldots, N.$$
   (c) Draw $w_{k+1}^n \sim \mathcal{N}(0, \mathsf{C}_{k+1}^n)$, $n = 1, 2, \ldots, N$.
   (d) Generate the prediction ensemble $\mathcal{S}_{k+1|k}$ via the evolution equation
   $$x_{k+1|k}^n = x_{k+1|k,\Psi}^n + w_{k+1}^n, \quad n = 1, 2, \ldots, N.$$
   (e) Compute the prior mean and covariance using the ensemble statistics
   $$\overline{x}_{k+1|k} = \frac{1}{N}\sum_{n=1}^{N} x_{k+1|k}^n$$
   and
   $$\mathsf{\Gamma}_{k+1|k} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{k+1|k}^n - \overline{x}_{k+1|k})(x_{k+1|k}^n - \overline{x}_{k+1|k})^{\mathsf{T}}.$$
   (f) Inflate the prior covariance by a multiplicative factor $(1 + \delta)$,
   $$\mathsf{\Gamma}_{k+1|k} = (1 + \delta)\mathsf{\Gamma}_{k+1|k},$$
   for some $\delta > 0$. If no inflation is desired, set $\delta = 0$.

(Continued on the next page)

Algorithm 9 LMM EnKF for Linear Observation Models (Cont.)

3. *Observation update*: If an observation y_{k+1} arrives,
   (a) Generate the observation ensemble

$$\left\{y_{k+1}^1, y_{k+1}^2, \ldots, y_{k+1}^N\right\}$$

   via the formula

$$y_{k+1}^n = y_{k+1} + v_{k+1}^n, \quad n = 1, 2, \ldots, N,$$

   where $v_{k+1}^n \sim \mathcal{N}(0, \mathsf{D})$.
   (b) Compute the Kalman gain

$$\mathsf{K}_{k+1} = \Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}(\mathsf{G}_{k+1}\Gamma_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

   (c) Generate the posterior state ensemble $\mathcal{S}_{k+1|k+1}$ using the updating formula

$$x_{k+1|k+1}^n = x_{k+1|k}^n + \mathsf{K}_{k+1}(y_{k+1}^n - \mathsf{G}_{k+1}x_{k+1|k}^n), \quad n = 1, 2, \ldots, N.$$

   Otherwise, set $\mathcal{S}_{k+1|k+1} = \mathcal{S}_{k+1|k}$.
4. *Posterior ensemble calculations*: Compute the posterior mean and error covariance using the ensemble statistics

$$\overline{x}_{k+1|k+1} = \frac{1}{N}\sum_{n=1}^{N} x_{k+1|k+1}^n$$

   and

$$\Gamma_{k+1|k+1} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{k+1|k+1}^n - \overline{x}_{k+1|k+1})(x_{k+1|k+1}^n - \overline{x}_{k+1|k+1})^{\mathsf{T}}.$$

5. If $k < T$, set $k = k + 1$ and repeat from Step 2; otherwise, stop.

ensemble, i.e., we set $\mathcal{S}_{k+1|k+1} = \mathcal{S}_{k+1|k}$, and continue the algorithm. Performing the time evolution in this manner, we arrive at an LMM-based variant of the traditional EnKF algorithm for linear observation models, which we summarize in Algorithm 9.

In Step 2(f) of Algorithm 9, we include the option of inflating the prior covariance matrix, though this may not be necessary in all situations. The empirical choice of an optimal inflation parameter δ is explored in the computed examples with the chaotic Lorenz-63 system in Section 8.2. If the computation of the Kalman gain matrix in Step 3(b) is not feasible due to the dimensionality of the problem, as with the standard EnKF, we can instead solve the linear system

$$(\mathsf{G}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}} + \mathsf{D})\xi_{k+1}^{n} = y_{k+1}^{n} - \mathsf{G}_{k+1}x_{k+1|k}^{n}$$

and replace the state ensemble update in Step 3(c) by

$$x_{k+1|k+1}^{n} = x_{k+1|k}^{n} + \mathsf{\Gamma}_{k+1|k}\mathsf{G}_{k+1}^{\mathsf{T}}\xi_{k+1}^{n}, \quad n = 1, 2, \ldots, N.$$

Similarly to the standard EnKF algorithm, the LMM EnKF algorithm is amenable to parallelization, since each ensemble member is independently propagated. Moreover, for problems such as the Lorenz-63 model considered in Section 8.2, a vectorized version of the LMM EnKF algorithm is possible.

8.2 Computed example: Lorenz-63 model

We test the LMM EnKF algorithm for linear observation models on the well-known Lorenz-63 model [11], a simplified approximation to the Navier-Stokes equations describing atmospheric convection, governed by the following three-dimensional system of nonlinear ODEs:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= \rho x - y - xz \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{8.1}$$

where x, y and z are the states, or components, of the system and σ, ρ and β are the parameters.

The Lorenz-63 model is an example of a deterministic dynamical system known for exhibiting chaotic behavior for certain combinations of the initial values and the parameters σ, ρ and β. In particular, letting $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$ leads the system to behave chaotically. This system has been analyzed and tested extensively in the literature, especially in regard to nonlinear filtering algorithms, e.g., [158, 159, 160, 100, 63, 64].

In our numerical simulations, we generate the reference state from the initial condition

$$(x_0, y_0, z_0) = (1.508870, -1.531271, 25.46091)$$

using `ode45` with relative tolerance 10^{-8}, propagating over the time interval $[0, 40]$, where the time units are arbitrary. Our filtering goal is to recover the blind components of the system from partial, noisy observations of the measured components. To this end, observations of the y and z components of (8.1) are obtained at given times, as specified below, corrupted by additive Gaussian noise with mean 0 and variance 2, such that the observation noise level is $\sqrt{2}$. Note that this simulation protocol is in agreement with the computed examples for this system which have been reported in the literature; e.g., see [158, 63]. The x component is not measured and therefore is treated as a blind component of the system. A cloud of initial values is generated for the x component by adding Gaussian noise of mean 0 and variance 8, such that the standard deviation is $2\sqrt{2}$, to the point x_0; the initial values for y and z are given by y_0 and z_0, respectively.

We want to recover the time series of the x component of the Lorenz-63 system (8.1) with chaotic parameters from noisy measurements of the observed states y and z at given times. We quantify the accuracy of the filtered solution $\widehat{x}$ by computing the root mean square error (RMSE) of the x component as compared to the reference

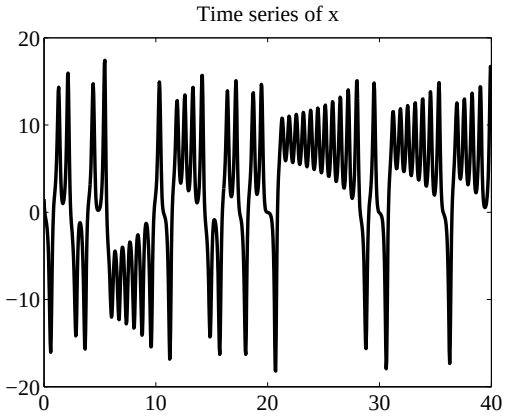

Figure 8.1: The reference solution for the x component of the chaotic Lorenz-63 system (8.1).

state x^{ref} at the observation points, i.e.,

$$\text{RMSE}(x) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widehat{x}_i - x_i^{\text{ref}})^2},$$

where M is the number of observations. In line with what has been done in the literature, e.g., in [100], the RMSE should be lower than the observation noise level, which here is $\sqrt{2} \approx 1.4142$, in order for the performance of the filter to be considered reasonable. A plot of the reference solution x^{ref} is given in Figure 8.1. In the following numerical experiments, we compute the RMSE of the x component obtained by filtering with LMM EnKF using six different LMMs for propagation, computing the innovation variance in the time evolution step via the HOMEC method, for seven different time steps h and with ensembles of size $N = 10$ and $N = 100$.

In our first numerical experiment, we observe the y and z components every 0.08 time units and filter using the LMM EnKF algorithm, as described in Algorithm 9, without inflating the prior covariance, i.e., setting the inflation parameter $\delta = 0$. Table 8.1 lists the resulting RMSEs of the x component obtained by LMM EnKF with different time step sizes h and ensemble size $N = 10$, and Table 8.2 reports the analogous results for ensemble size $N = 100$. Figure 8.2 shows plots of the LMM

EnKF time series estimates of x obtained by propagating with methods AB1, BDF1 and AM1 with time step $h = 0.04$ and ensemble size $N = 100$ against the reference solution, as well as the corresponding absolute values of the difference between the LMM EnKF estimated solution and the reference solution plotted against the LMM EnKF posterior error standard deviation estimate.

The plots in Figure 8.2 show that, for the given time step and ensemble size, LMM EnKF using AB1 and AM1 outperform BDF1. This is especially clear in the error plots, where the maximum absolute error using BDF1 is about 5 times that of AB1 and 10 times that of AM1. Further, the estimated error standard deviation obtained from the posterior error covariance matrix using BDF1 significantly underestimates the actual absolute error. In this regard, propagation with AM1 gives the best results, because its estimated error standard deviation matches well the corresponding absolute error. The poor performance of BDF1 in this case is not too surprising in light of the strong damping effects of BDF methods, as can be seen clearly in the time series estimate.

Based on standard knowledge of the properties of LMMs, intuitively we would expect that (a) as the time step gets smaller, the RMSE should decrease or at least not increase, and (b) higher order methods should perform better than or at least comparably to their lower order counterparts. While the results shown in Table 8.2 for ensemble size $N = 100$ are mostly in agreement with this intuition, the results in Table 8.1 are not: the behavior of the RMSE is somewhat erratic as h decreases, and higher order LMMs often have significantly larger RMSE values than their lower order counterparts. This suggests that the size of the ensemble plays a role in the accuracy of the filtered solution. Moreover, when propagating with AB1 for both ensemble sizes, at some point, as h goes from 0.02 to 0.01, making the step size smaller systematically increases the RMSE. Such behavior seems to point to a problem with filter divergence. In fact, when the step size is smaller, there are more time steps in between data arrivals. Therefore, if the filter starts to diverge in the sense that the prior covariance becomes too confident, the additional time steps between observation

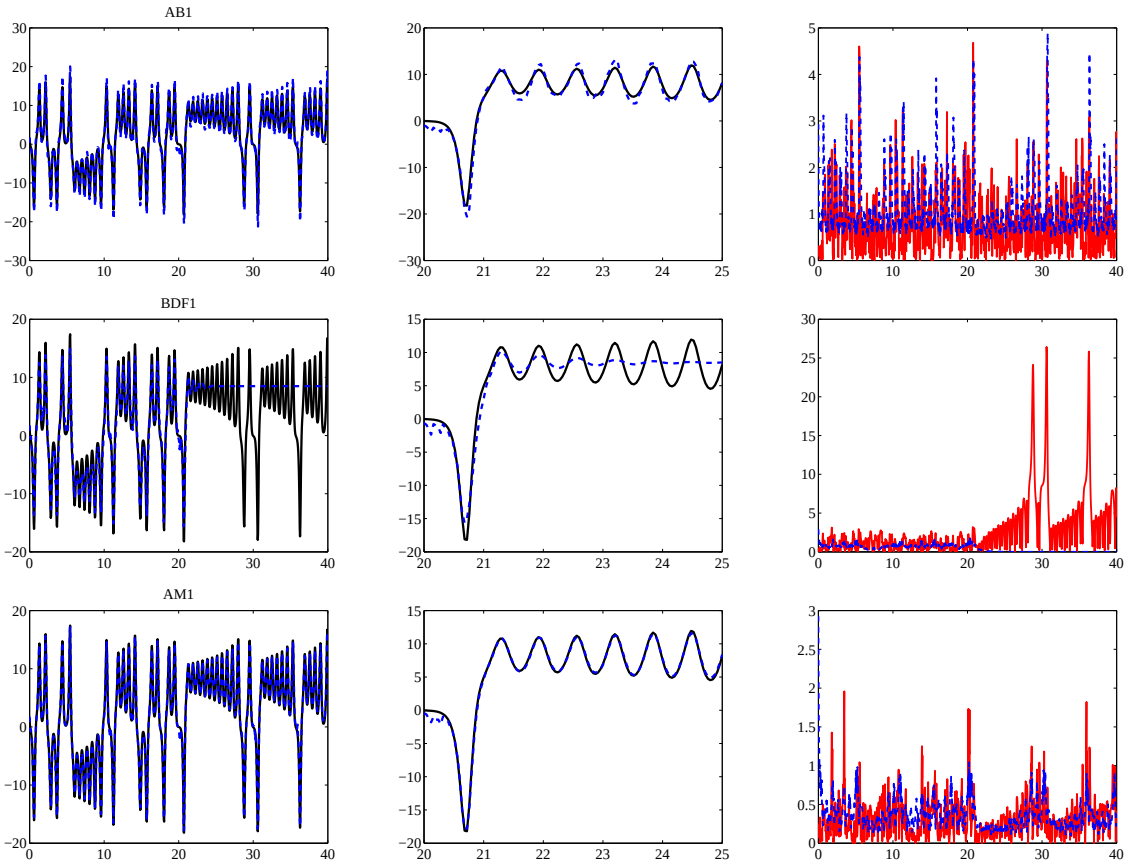

Figure 8.2: The LMM EnKF time series estimates plotted with the reference solution over the full time interval [0,40] (left) and on the interval [20,25] (center), as well as the absolute values of the difference between the LMM EnKF estimated solutions and the reference solution plotted along with the LMM EnKF posterior error standard deviation estimates (right), for blindly filtering the x component of the chaotic Lorenz-63 system (8.1) propagating with AB1 (top), BDF1 (middle) and AM1 (bottom). In the left and center plots, the LMM EnKF estimated solution is plotted as a dashed blue line, while the reference solution is plotted as a solid black line. In the plots on the right, the absolute difference is plotted in solid red and the LMM EnKF error standard deviation estimate in dashed blue.

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	3.9242	6.5421	4.7409	1.4801	0.7918	0.7199
0.0400	1.2447	0.9585	4.6328	0.7390	0.4277	0.6457
0.0200	1.1513	0.6497	12.5864	0.5824	0.3222	7.5628
0.0100	2.2736	1.1689	4.5254	0.6130	2.0088	0.2962
0.0050	7.0365	0.3081	1.4021	0.6386	0.2657	0.2935
0.0025	8.7215	0.5721	0.6429	0.6970	9.2821	4.3645
0.0010	7.9368	3.0537	0.4825	0.6054	5.2152	6.7874

Table 8.1: RMSEs of the x component of the chaotic Lorenz-63 system (8.1) obtained by LMM EnKF with ensemble size $N = 10$ for various time steps h using no prior covariance inflation. Noisy observations of only the y and z states of the system are made every 0.08 time units.

updates in which we do nothing to correct the prior will make the situation worse. This is one reason for exploring the effects of inflating the prior covariance in the LMM EnKF algorithm using a nonzero inflation parameter $\delta > 0$.

Motivated by the results reported above, in our second numerical experiment, we investigate the effects on the resulting RMSE of x when we inflate the prior covariance by a factor $(1 + \delta)$ for some $\delta > 0$. In the literature, the so-called "optimal" value of the inflation parameter $\delta > 0$ is commonly chosen empirically; see, e.g., [85]. In this spirit, we run the LMM EnKF using the aforementioned six LMMs for propagation for seven different time steps, varying the δ parameter incrementally by 0.02 between the values of 0 and 0.3, i.e., inflating the covariance by an amount between 0% and 30%. The optimal value of δ is the one, within this discrete set, that results in the lowest RMSE for a given LMM propagator and given step size. The ensembles that we consider are of sizes $N = 10$ and $N = 100$. The optimal δ values, determined by the empirical test as described, and corresponding RMSEs when the ensemble size is $N = 10$ are presented in Tables 8.3 and 8.4, respectively; the results with ensemble size $N = 100$ are given in Tables 8.5 and 8.6.

Table 8.3 shows that when $N = 10$ almost all of the different LMM EnKF filters benefited from the inclusion of nonzero multiplicative variance inflation of the prior.

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	1.9328	4.1646	1.7106	1.2196	0.6566	0.5846
0.0400	0.9665	0.7540	4.5582	0.6161	0.3900	0.3847
0.0200	0.8645	0.5246	4.5369	0.5955	0.2990	0.2623
0.0100	1.2507	0.3559	4.4967	0.6416	0.2755	0.2704
0.0050	7.2411	0.2880	0.9574	0.6593	0.2823	0.2723
0.0025	4.7467	0.2786	0.4969	0.6671	0.2582	0.2813
0.0010	4.8850	0.2867	0.4087	0.6863	0.2668	0.2676

Table 8.2: RMSEs of the x component of the chaotic Lorenz-63 system (8.1) obtained by LMM EnKF with ensemble size $N = 100$ for various time steps h using no prior covariance inflation. Noisy observations of only the y and z states of the system are made every 0.08 time units.

The optimal value of δ changes with the LMM propagator and the time step. The RMSE results when using these optimal δ values to inflate the prior covariance, as shown in Table 8.4, confirm our original intuition regarding the behavior of the LMM EnKF algorithm. The results in Tables 8.5 and 8.6, in line with our earlier comments, show that the prior covariance inflation is not as crucial to obtain a minimum RMSE when $N = 100$. This seems to suggest that variance inflation is has a bigger impact on the accuracy of the filter for smaller ensemble sizes. Moreover, the RMSE results when propagating with AB1 for small time steps do improve somewhat, suggesting that the use of variance inflation does partially alleviate the issue of filter divergence.

8.3 EnKF with spatio-temporal prior

As noted in Sections 4.3 and 4.4, in the general Bayesian filtering setting, the time evolution update serves as prior information for the observation update. This is true in particular for the standard EnKF algorithm given in Section 4.4.3 and the LMM-based EnKF variant derived in Section 8.1. That said, a time evolution model with poorly known initial values may not provide enough prior information for some components of the state variable $X \in \mathbb{R}^d$, leading to poor performance of the EnKF algorithm, especially when the coupling of the components with the observed com-

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	0.16	0.10	0.28	0.28	0.30	0.28
0.0400	0.26	0.10	0.14	0.12	0.18	0.22
0.0200	0.30	0.08	0.28	0.06	0.16	0.26
0.0100	0.30	0.16	0.22	0.08	0.14	0.14
0.0050	0.18	0.06	0.28	0	0.10	0.22
0.0025	0.14	0.26	0.14	0.10	0.22	0.18
0.0010	0.12	0.02	0.12	0.08	0.26	0.12

Table 8.3: The optimal inflation parameters δ with respect to computing the RMSE of the x component of the chaotic Lorenz-63 system (8.1) using LMM EnKF with ensemble size $N = 10$ for various time steps h. Noisy observations of only the y and z states of the system are made every 0.08 time units. The corresponding RMSEs are given in Table 8.4.

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	2.7467	5.2169	2.0508	1.2884	0.7047	0.6364
0.0400	1.1206	0.8630	1.4827	0.7021	0.3952	0.4463
0.0200	0.8998	0.5633	1.3415	0.5833	0.2943	0.3177
0.0100	1.4233	0.3573	1.8657	0.6472	0.2649	0.2518
0.0050	4.6654	0.2568	0.7164	0.6295	0.2353	0.2623
0.0025	4.7320	0.2624	0.5331	0.6500	0.2614	0.2492
0.0010	2.2941	0.2620	0.4181	0.6263	0.2761	0.2322

Table 8.4: The RMSEs of the x component of the chaotic Lorenz-63 system (8.1) corresponding to the optimal inflation parameters δ given in Table 8.3 using LMM EnKF with ensemble size $N = 10$ for various time steps h. Noisy observations of only the y and z states of the system are made every 0.08 time units.

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	0.02	0.26	0.30	0.18	0.28	0.28
0.0400	0.30	0.14	0.30	0	0.20	0.16
0.0200	0.30	0.08	0.28	0	0.10	0.04
0.0100	0.30	0.06	0.20	0.02	0.18	0.08
0.0050	0.30	0	0.26	0.02	0.04	0
0.0025	0.18	0.04	0.28	0	0.02	0.12
0.0010	0.28	0.08	0.04	0	0.08	0.22

Table 8.5: The optimal inflation parameters δ with respect to computing the RMSE of the x component of the chaotic Lorenz-63 system (8.1) using LMM EnKF with ensemble size $N = 100$ for various time steps h. Noisy observations of only the y and z states of the system are made every 0.08 time units. The corresponding RMSEs are given in Table 8.6.

h \ LMM	AB1	AB2	BDF1	BDF2	AM1	AM2
0.0800	1.7210	3.5364	1.5989	1.1506	0.6190	0.5667
0.0400	0.8709	0.7229	1.0795	0.6353	0.3835	0.3610
0.0200	0.7490	0.5118	4.4963	0.5912	0.2904	0.2674
0.0100	0.9843	0.3239	1.0058	0.6373	0.2734	0.2657
0.0050	1.7677	0.2743	0.5786	0.6756	0.2604	0.2590
0.0025	3.5796	0.2633	0.4630	0.6783	0.2642	0.2662
0.0010	0.7801	0.2674	0.4055	0.6914	0.2652	0.2656

Table 8.6: The RMSEs of the x component of the chaotic Lorenz-63 system (8.1) corresponding to the optimal inflation parameters δ given in Table 8.5 using LMM EnKF with ensemble size $N = 100$ for various time steps h. Noisy observations of only the y and z states of the system are made every 0.08 time units.

ponents is weak. The uncontrolled behavior of such components is often easy to identify *a posteriori* if there is some indication that the behavior predicted by the EnKF algorithm is in conflict with what one expects to see *a priori*. In other words, one may have *a priori* knowledge that the state vector X_k is distributed according to the prior density $\pi_{\text{prior}}(x_k)$, outside the context of the time evolution. It is then of interest to find a way to include this prior knowledge of the system behavior into the model, so to avoid potential conflict with the EnKF predicted values as the algorithm progresses.

In order to include prior information about the distribution of X_k into the transition kernel for the time evolution update to X_{k+1}, we call upon Bayes' theorem (3.4) and write

$$\pi(x_{k+1} \mid x_k) = \frac{\pi(x_k \mid x_{k+1})\pi_{\text{prior}}(x_{k+1})}{\pi(x_k)}, \quad \pi(x_k) \neq 0,$$

where

$$\pi(x_k) = \int \pi_{\text{prior}}(x_{k+1})\pi(x_k \mid x_{k+1})dx_{k+1}.$$

We can think of the probability density $\pi(x_k \mid x_{k+1})$ as a backward transition density from x_{k+1} to x_k.

To distinguish the prior density $\pi_{\text{prior}}(x_{k+1})$ from the prior associated with the time evolution equation, we call $\pi_{\text{prior}}(x_{k+1})$ the *spatial prior*, following the notion that x_{k+1} is a random field with components referring to different points in space, like pixels in an image. This interpretation is not restrictive, however, as will be demonstrated in the computed example in Section 8.4.

Assume, for simplicity, that we expect the random variable X_{k+1} to obey a Gaussian distribution,

$$X_{k+1} \sim \mathcal{N}(x_0, \Gamma_0),$$

such that

$$\begin{aligned}\pi_{\text{prior}}(x_{k+1}) &\propto \exp\left\{ -\frac{1}{2}(x_{k+1}-x_0)^{\mathsf{T}}\mathsf{\Gamma}_0^{-1}(x_{k+1}-x_0)\right\} \\ &= \exp\left\{ -\frac{1}{2}||x_{k+1}-x_0||^2_{\mathsf{\Gamma}_0}\right\}.\end{aligned}$$

Incorporating the spatial prior into the EnKF analysis step (4.42) yields

$$x^n_{k+1|k+1} = \arg\min_x \left\{ ||y^n_{k+1} - G_{k+1}(x)||^2_{\mathsf{D}} + ||x - x^n_{k+1|k}||^2_{\mathsf{\Gamma}_{k+1|k}} + ||x-x_0||^2_{\mathsf{\Gamma}_0}\right\},$$

which is the updating formula for the posterior ensemble, $n = 1, 2, \ldots, N$. One way to interpret this formula, keeping the modifications of the algorithm to a minimum, is to view x_0 as a fictitious observation of X_k; i.e., we define an extended observation model,

$$\widehat{y}_{k+1} = \begin{bmatrix} y_{k+1} \\ x_0 \end{bmatrix} = \begin{bmatrix} G_{k+1}(x_{k+1}) \\ x_k \end{bmatrix} + \widehat{v}_{k+1},$$

where

$$\widehat{v}_{k+1} \sim \mathcal{N}(0, \mathsf{\Delta}), \qquad \mathsf{\Delta} = \begin{bmatrix} \mathsf{D} & 0 \\ 0 & \mathsf{\Gamma}_0 \end{bmatrix}.$$

In the case of a linear observation model, we define

$$\mathsf{P}_{k+1} = \begin{bmatrix} \mathsf{G}_{k+1} \\ \mathsf{I} \end{bmatrix},$$

and the only change is in the Kalman gain, which becomes

$$\mathsf{K}_{k+1} = \mathsf{\Gamma}_{k+1|k}\mathsf{P}^{\mathsf{T}}_{k+1}(\mathsf{P}_{k+1}\mathsf{\Gamma}_{k+1|k}\mathsf{P}^{\mathsf{T}}_{k+1} + \mathsf{\Delta})^{-1},$$

while the rest of the EnKF algorithm remains intact.

A further extension of the model can be obtained if we assume prior information that is not captured by the evolution model. In its simplest form, we may assume that the spatial prior depends on the previous state estimate x_k, so that

$$X_{k+1} \sim \mathcal{N}\big(x_0(x_k), \mathsf{\Gamma}_0(x_k)\big),$$

i.e., the current spatial prior is a function of the previous state. We refer to the combination of the spatial prior with the time evolution equation as a *spatio-temporal* prior. Such a condition would appear naturally if, e.g., in addition to the time evolution model

$$X_{k+1} = F_{k+1}(X_k) + W_{k+1},$$

we have a reason to believe that

$$X_{k+1} = X_k + U_{k+1},$$

where U_{k+1} is an innovation process independent of W_{k+1}, assumed to be Gaussian with zero mean and covariance matrix Γ_0. This model extension can be handled either by assuming x_k to represent a fictitious observation of X_{k+1}, or by extending the propagation model to be

$$X_{k+1} = H_{k+1}(X_{k+1}) + \widetilde{W}_{k+1} = \begin{bmatrix} F_{k+1}(X_k) \\ X_k \end{bmatrix} + \begin{bmatrix} W_{k+1} \\ U_{k+1} \end{bmatrix}.$$

These two different interpretations lead to slightly different modifications of the EnKF algorithm: In the latter interpretation, the innovation to x_0 is added after the propagation, thus affecting the predicted covariance, while in the former, the innovation is added to x_0 when the fictitious observation ensemble is generated. For more details on spatial priors in the context of general Bayesian filtering, see [5, 161, 162].

8.4 Application to metabolism kinetics

Combining the ideas in the previous sections, we apply the LMM EnKF algorithm with a spatio-temporal prior to study the following problem:

> Given select observed metabolite concentrations in the blood and in the tissue, obtained by biopsies of MR spectroscopy, predict the dynamics of the several blind metabolites.

The metabolite kinetics are typically analyzed in the context of well-mixed, lumped multi-compartment models. In our example, we consider a compartment model for a skeletal muscle comprising two compartments, blood and tissue. The metabolic network in the tissue compartment consists of a simplified glycolysis and β-oxidation of fatty acids, connected to the oxidative phosphorylation via the tricarboxylic acid (TCA) cycle. Furthermore, the tissue has a creatine-phosphocreatine buffer cycle, as well as glycogen a triglyceride reservoirs that can be either replenished or depleted according to the substrate availability. The metabolism kinetics are described in terms of Michaelis-Menten dynamics for enzymatic reactions and facilitated transports between blood and tissue. In the blood compartment, both oxygen (O_2) and carbon dioxide (CO_2) appear either as free or bound to hemoglobin, and in the tissue they appear as either free or bound to myoglobin. This two-compartment model comprises a total of 39 metabolite concentrations, which are the model states, and 96 parameters. A detailed description of the model is given in the reference [2].

The metabolic kinetics model can be written concisely as

$$\frac{dC}{dt} = f(t, C, \theta), \quad C(0) = C_0,$$

where $C \in \mathbb{R}^{39}$ contains the concentrations of the different metabolites, each metabolite counted with the multiplicity of the compartment where it appears, so that if the same metabolite is present in blood and tissue, it contributes two concentration components. To guarantee the positivity of the concentrations in our model, we define the auxiliary variables $X_j \in \mathbb{R}^{39}$,

$$C_j(t) = e^{X_j(t)}, \quad 1 \leq j \leq 39. \tag{8.2}$$

The auxiliary variables satisfy the differential equation

$$\frac{dX_j}{dt} = \frac{1}{C_j}\frac{dC_j}{dt} = \frac{f_j(t, C, \theta)}{C_j(t)},$$

and by making the substitution (8.2), the system can be expressed in terms of X alone as

$$\frac{dX}{dt} = f(t, e^X, \theta) \odot e^{-X} = F(t, X, \theta), \quad (e^{\pm X})_j = e^{\pm X_j},$$

where the symbol "$\odot$" denotes componentwise multiplication.

Assume that the data consist of observations of a few concentrations at some discrete times. Denoting by $d_k^* \in \mathbb{R}^m$ the noiseless data vector at time $t = t_k$, we have

$$d_k^* = \begin{bmatrix} C_{\ell_1}(t_k) \\ \vdots \\ C_{\ell_m}(t_k) \end{bmatrix} = \mathsf{B}C(t_k),$$

where $\mathsf{B}_{ij} = \delta_{j,\ell_i}$. To guarantee that the data remains non-negative, we model the observation error using multiplicative noise, that is, the data d_k are given by

$$d_k = d_k^* \odot \alpha_k,$$

where the components of the noise vector $\alpha_k \in \mathbb{R}^m$ are positive. Further, by writing

$$\alpha_k = e^{\log \alpha_k} = e^{w_k},$$

we arrive at the logarithmic observation model

$$y_k = \log d_k = \log d_k^* + w_k = \mathsf{B}X(t_k) + w_k,$$

where we assume, for convenience, that the noise w_k is zero mean Gaussian noise with i.i.d. components,

$$w_k \sim \mathcal{N}(0, \sigma^2 \mathsf{I}_m).$$

Note that two standard deviations for the noise corresponds to a relative error of $e^{\pm 2\sigma}$, i.e.,

$$|(w_k)_j| \le 2\sigma \quad \text{ if and only if } \quad e^{-2\sigma} \le \frac{(d_k)_j}{(d_k^*)_j} \le e^{2\sigma}.$$

This noise model allows us to perform both the time evolution update and the observation update of the LMM EnKF algorithm in terms of the logarithmic variables.

In our computed example, we use this dynamical system to simulate an episode of ischemia by heavily reducing the blood flow starting at the time $t = t_0$, and, after awhile, slowly letting it recover to the initial value. The model for the blood flow dynamics is given by

$$Q(t) = Q_0 - \Delta Q \frac{(t - t_0)_+}{\tau} e^{-(t-t_0)/\tau},$$

where $(t - t_0)_+$ denotes the non-negative part of the linear function $(t - t_0)$. The parameter τ is a time constant of the ischemia, and ΔQ is related to the minimum value of the blood flow, $Q_{\min}$, as

$$Q_{\min} = Q(t_0 + \tau) = Q_0 - \frac{\Delta Q}{e}.$$

The simulation is started with the initial values taken to be the posterior mean values from the MCMC sample generated in [2], and the system is first run for 100 minutes to guarantee that it attains a near stationary state. Note that stationary state is not necessarily a steady state: we allow the reservoir concentrations glycogen and triglyceride to increase or decrease with a near constant rate. After this preliminary run, we reset the time to $t = 0$ and start the simulation, using the blood flow profile shown in Figure 8.3.

8.4.1 The need for spatio-temporal priors

The data are assumed to consist of the measured concentrations in the blood. With no additional information, we expect that the LMM EnKF algorithm as proposed in Section 8.1 will encounter difficulties in identifying the concentration time courses, due to the following factors, among others.

(i) First, consider the "futile cycle," sometimes referred to as the glycogen shunt,

$$\mathrm{G6P} + \mathrm{ATP} \rightarrow \mathrm{GLY} + \mathrm{ADP} + 2\,\mathrm{Pi} \rightarrow \mathrm{G6P} + \mathrm{ADP} + \mathrm{Pi},$$

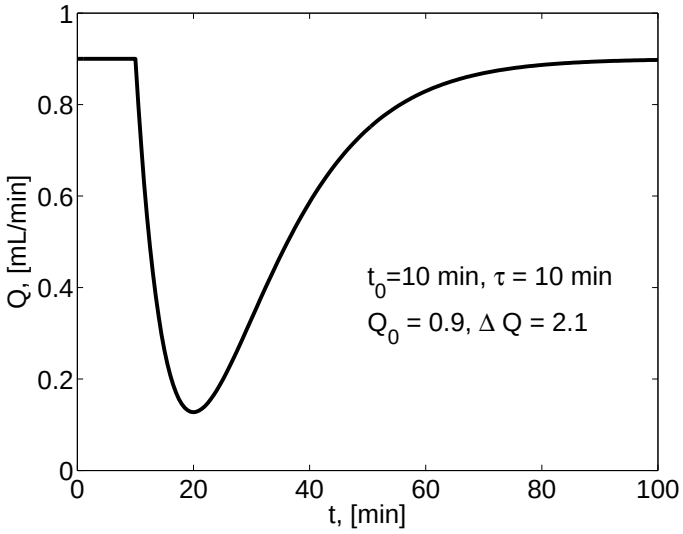

Figure 8.3: The blood flow dynamics used in the skeletal muscle metabolism simulation. At $t = t_0$, the normal blood flow Q_0 is constrained, then is slowly let to return to normal. The minimum is attained at time $t = t_0 + \tau$, the minimum value being $Q_0 - \Delta Q/e \approx 0.13$.

which, in effect, is equivalent to the ATP hydrolysis,

$$\text{ATP} \rightarrow \text{ADP} + \text{Pi}.$$

With no information about the glycogen level, there is an ambiguity between the ATP hydrolysis and glycogen shunt, since one can be freely traded to the other without changing anything else in the system.

(ii) Another cycle,

$$\text{PCr} + \text{ADP} \rightarrow \text{Cr} + \text{ATP} \rightarrow \text{PCr} + \text{ADP},$$

can freely change the phosphorylation state of the system simply by adjusting correspondingly the ratio between PCr and Cr. The phosphorylation state, however, has an effect on other reactions, and therefore, if no prior information about PCr and Cr is available, this ambiguity may have global consequences to the system. In particular, one may expect that if the TCA cycle is out of control, the problem propagates via the redox state.

(iii) A third cycle, the triglyceride-lipolysis cycle,

$$\mathrm{TGL} + 3\,\mathrm{ATP} \to \mathrm{GLC} + 3\,\mathrm{FFA} + 2\,\mathrm{ATP} \to \mathrm{TGL} + 3\,\mathrm{ADP} + 3\,\mathrm{Pi},$$

is effectively equivalent to the ATP hydrolysis,

$$3\,\mathrm{ATP} \to 3\,\mathrm{ADP} + 3\,\mathrm{Pi}.$$

The aforementioned cycles (i) and (iii) can disturb the system in the following manner: Assume that, in cycle (i), for some reason, the GLY level settles to a lower value than the true value, but the ATP hydrolysis keeps the phosphorylation level at the correct value; then the G6P concentration also needs to be lower than the true value, otherwise the glycogen shunt is not in equilibrium. However, too low a G6P level slows down the rest of the glycolysis, and this problem starts to propagate, leading possibly to a global problem. Similar reasoning applies to the triglyceride-lipolysis cycle (iii).

For these reasons, it seems reasonable that a prior concerning the glycogen, triglyceride, ATP hydrolysis, and creatine/phosphocreatine should be included in the LMM EnKF algorithm. Information concerning the ATP hydrolysis corresponds to specifying the energetic need of the muscle, i.e., an estimate of how much ATP is burned.

Furthermore, some of the time constants in the system may be short, making it possible that the concentrations react rather quickly to external changes in substrate availability. Such rapid reactions are not disruptive if the blood flow is changing rapidly: in that case, the system needs to adjust itself quickly. However, when the blood flow is changing slowly, we do not expect to see wild variations in the concentrations. Therefore, in the slowly-changing blood flow regime, it seems reasonable to implement a smoothness prior of the type

$$X(t_{j+1}) = X(t_j) + \Delta t W, \qquad W \sim \mathcal{N}(0, \gamma^2 \mathsf{I}_n)$$

for some choice of variance coefficient γ^2.

8.4.2 Results

We apply the LMM EnKF algorithm with the spatio-temporal prior to the skeletal muscle metabolism problem as described above. More specifically, we aim to recover time series estimates for the 39 model concentrations from noisy measurements of 8 concentrations in the blood at 11 time instances over the time interval [0,100] minutes. The observed concentrations in the blood compartment are GLU, LAC, ALA, TGL, GLC, FFA, CO_2 and O_2. We propagate with the LMM BDF1 and compute the innovation term in the evolution equation via the HOMEC technique. We choose the time step $h = 0.1$, which corresponds to 6 seconds, and an ensemble size of $N = 150$ members.

In defining the initial distribution for the state ensemble, we generate a cloud of normally distributed random variables in which the mean is shifted from the initial log concentrations used for synthetic data generation by a factor ϵu, where $u \sim \text{Uniform}(-1/2, 1/2)$, and the covariance is given by a multiple proportional to the time step and a fraction of the log concentrations. More precisely, for each concentration $j = 1, \ldots, 39$, we generate a cloud

$$X_j^n(0) = \left[1 + \epsilon u_j^n + 0.001 h \xi_j^n\right] \log C_j(0), \quad n = 1, \ldots, N,$$

where $u_j^n \sim \text{Uniform}(-1/2, 1/2)$, $\xi_j^n \sim \mathcal{N}(0, 1)$, and $\epsilon = 0.005$.

Furthermore, instead of fixing all of the 96 model parameters as is traditionally done in the EnKF setting, we "free" certain parameters by generating a cloud of parameter values in a similar fashion to the generation of the initial concentration ensemble. In particular, we free the 26 $V_{\max}$, 26 K, 12 μ and 10 ν parameters of the Michaelis-Menten-type reaction terms, for a total of 74 free parameters. If we write these parameters concisely in the vector

$$\theta = (V_{\max}, K, \mu, \nu) \in \mathbb{R}^{74},$$

then for each parameter $\ell = 1, \ldots, 74$, we generate a cloud

$$\theta_\ell^n = \left[1 + \epsilon u_\ell^n + 0.005 h \xi_\ell^n\right] \theta_\ell, \quad n = 1, \ldots, N,$$

where $u_\ell^n \sim \text{Uniform}(-1/2, 1/2)$, $\xi_\ell^n \sim \mathcal{N}(0,1)$, and $\epsilon = 0.005$. Allowing for a cloud of values for these parameters is more realistic than choosing a fixed value in this setting, as often the exact values of such parameters are not known. Note that after drawing the initial cloud, we do not artificially evolve the parameters in time; we simply assign each state ensemble member with its own corresponding set of parameters.

After each time evolution update, we inflate the prior covariance matrix by 20%. We incorporate the spatio-temporal prior as a fictitious observation of the state concentrations as described. The resulting LMM EnKF time series estimates and ± 2 estimated error standard deviation envelopes for the 39 model concentrations are plotted in Figures 8.4, 8.5, 8.6 and 8.7, along with the simulated noiseless time series obtained by propagating the true initial log concentrations using `ode15s`. Since PYR in the blood is disconnected from the other concentrations, the result shown for PYR in Figure 8.6 is simply numerical noise. We remark that these results, obtained using a cloud of parameter values as opposed to the fixed parameter assumption implicit to standard EnKF algorithms, serve as preliminary results for combined parameter and state estimation using the LMM EnKF algorithm, which is developed in the manuscript [157].

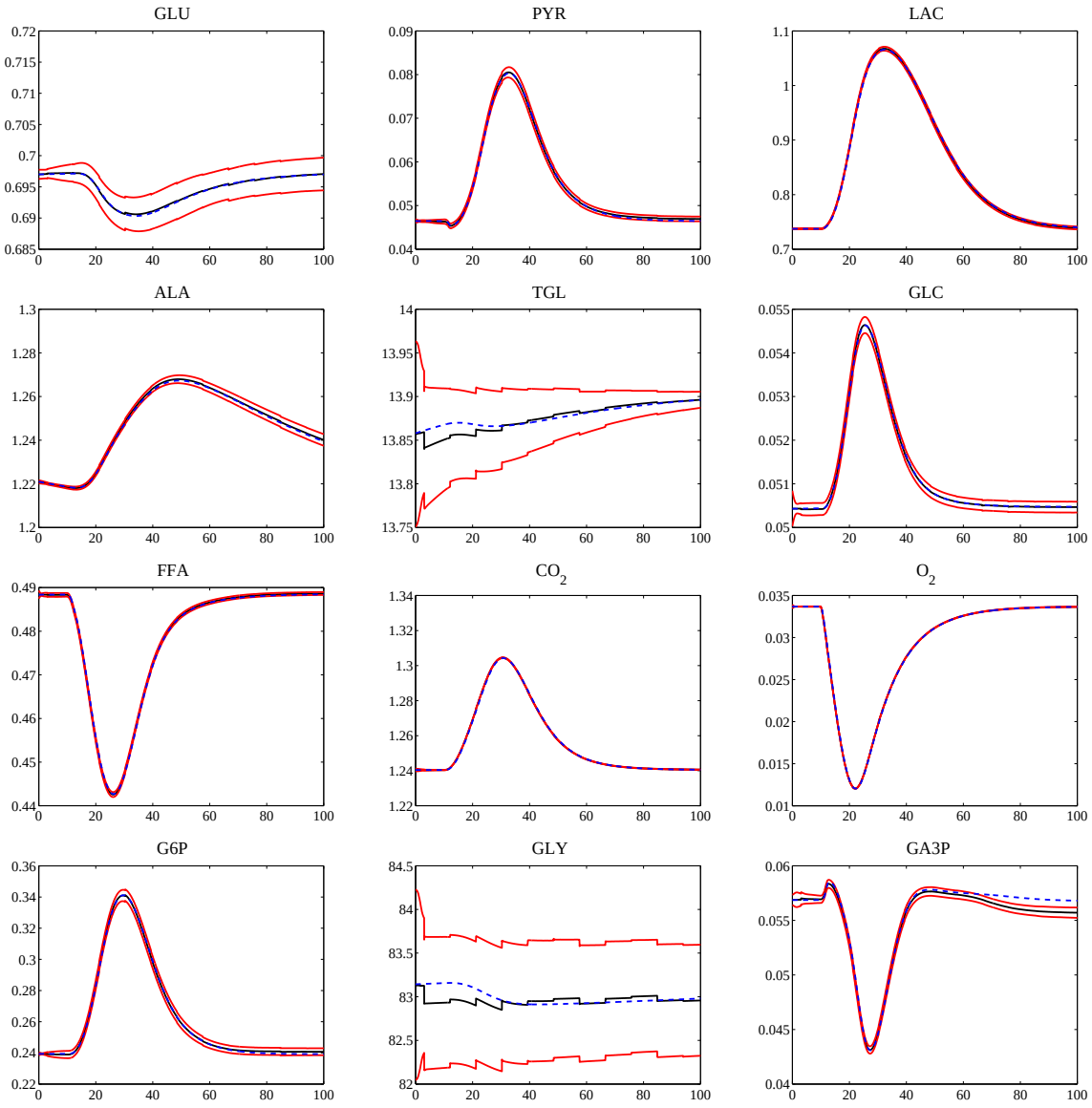

Figure 8.4: LMM EnKF time series estimates for concentrations 1-12 of the skeletal muscle metabolism problem. The LMM EnKF estimate is plotted as a solid black curve, and the ± 2 estimated error standard deviation curves are plotted in red. The expected solution is plotted as a dashed blue curve.

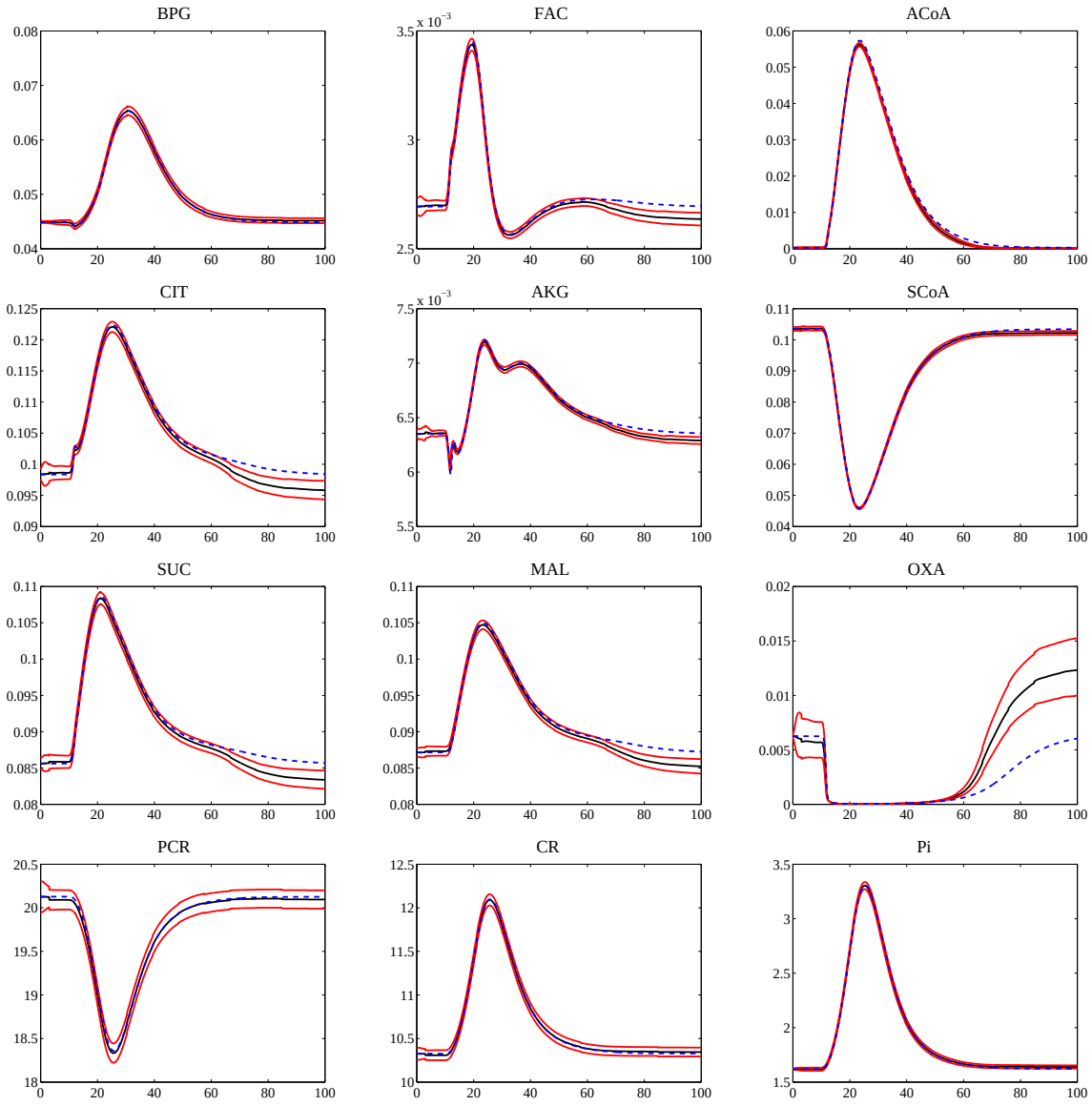

Figure 8.5: LMM EnKF time series estimates for concentrations 13-24 of the skeletal muscle metabolism problem. The LMM EnKF estimate is plotted as a solid black curve, and the ± 2 estimated error standard deviation curves are plotted in red. The expected solution is plotted as a dashed blue curve.

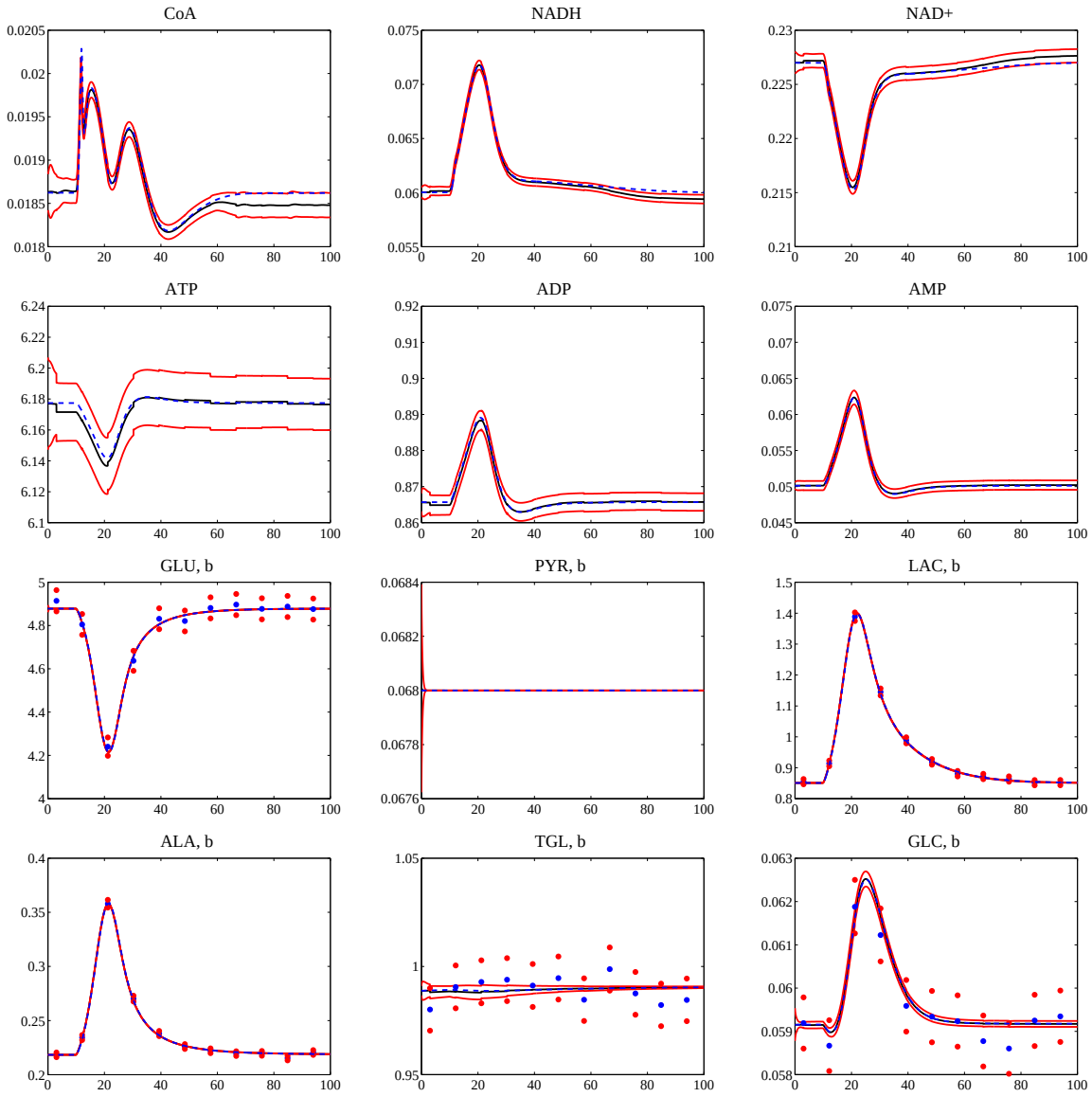

Figure 8.6: LMM EnKF time series estimates for concentrations 25-36 of the skeletal muscle metabolism problem. The LMM EnKF estimate is plotted as a solid black curve, and the ± 2 estimated error standard deviation curves are plotted in red. The expected solution is plotted as a dashed blue curve. The observed values ± 2 standard deviations are additionally plotted for measured concentrations.

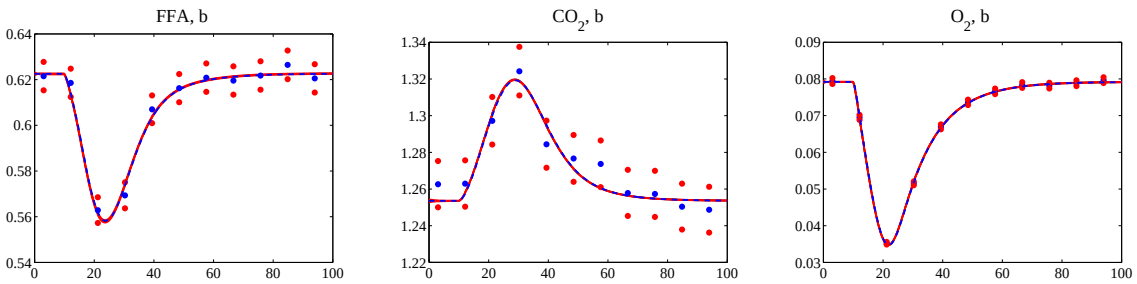

Figure 8.7: LMM EnKF time series estimates for concentrations 37-39 of the skeletal muscle metabolism problem. The LMM EnKF estimate is plotted as a solid black curve, and the ± 2 estimated error standard deviation curves are plotted in red. The expected solution is plotted as a dashed blue curve. The observed values ± 2 standard deviations are additionally plotted for measured concentrations.

Chapter 9

Conclusions

This thesis considers the problem of estimating the unknown parameters of a governing system of differential equations from noisy measurements of some of the states, or more generally, functions of the solution, at discrete times. By formulating the dynamic, ill-posed inverse problem in a Bayesian statistical framework, we discussed both sequential and non-sequential methods for solving state and parameter estimation problems for dynamical systems. We focused on the use of sequential Monte Carlo techniques, which approximate the integration required in the time evolution step using random samples, particularly considering particle filtering and ensemble Kalman filtering methods.

In Chapter 5, we addressed the questions of stability and accuracy of combined state and parameter estimation problems when the underlying dynamical model is a non-stochastic ODE with ill-determined initial values and/or coefficients. We proposed a method for carrying out the numerical propagation and systematically assigning the variance of the innovation term in the time evolution update by using LMM numerical solvers in the context of particle filtering. It was shown that by using suitable LMMs, it is possible to treat the problem using existing sequential estimation methods with the advantage of having good control of the stability and the accuracy, and that the numerical scheme provides a natural way to set the innovation variance. Though in this work we assign diagonal innovation covariances matrices, the method

could be extended to account for possible correlation between components. The LMM PF-SMC algorithm developed was shown to accurately estimate the parameters of a problem with characteristics of those found in biochemical applications under different testing conditions.

The approach was shown to be particularly appealing when the underlying system of differential equations is stiff. The stiffness of the system may be difficult to determine in particular when the system contains unknown parameters that are determined sequentially, since random draws of the parameter vector from the underlying distribution may easily contain parameter combinations that increase the stiffness uncontrollably. This is true especially for many nonlinear problems that arise in applications.

In Chapter 6, we investigated two computationally efficient implementations of the fixed time step LMM PF-SMC algorithm: parallelization and vectorization. Through applications to a suite of test problems in MATLAB, it was shown that while both implementations are effective in certain situations, the size and structure of the problem considered determine whether the parallelized or vectorized version of the algorithm is more efficient. In future work, we aim to explore the potential speedups gained by running the LMM PF-SMC algorithm in parallel on graphics processing units (GPUs), whose highly parallel structure may make them more effective for this type of algorithm than standard CPUs.

We applied a variant of the LMM PF-SMC algorithm in the estimation of unknown parameters of a model of tracer kinetics from sequences of real PET scan data in Chapter 7. A combination of standard optimization and statistical inference approaches was employed by running a quasi-Newton nonlinear least squares algorithm to find optimal starting values for the particle filter, then treating the optimized values as hyperparameters in the Bayesian statistical framework. A vectorized version of the LMM PF-SMC algorithm was modified to use variable time steps to account for the increase in interval length between data measurements from the beginning to the end of the procedure, while keeping the time step the same for each particle. By applying

the algorithm to dynamic PET images of $[1-^{11}C]$-acetate-derived tracer accumulation to estimate the transport rates in a two-compartment model of astrocytic uptake and metabolism of the tracer for a group of 18 volunteers from three groups (healthy control individuals, cirrhotic liver patients, and hepatic encephalopathy patients), the resulting parameter distributions displayed a clear distinction in parameter values for the hepatic encephalopathy patients as compared to the other two groups.

In Chapter 8, we furthered the idea derived in [10] to ensemble Kalman filters, employing a stochastic interpretation of the discretization error in numerical integrators and thereby extending the technique to deterministic, large-scale nonlinear evolution models, with innovation variance based on classical error estimates. The resulting algorithm was shown to be effective in recovering the blind components of the chaotic Lorenz-63 equations [11]. A variant of the algorithm, which introduces the use of a spatio-temporal prior in the EnKF framework, was applied to a problem of metabolism kinetics for skeletal muscle, where it was shown that by freeing some of the reaction parameters, we are able to estimate the unmeasured metabolite concentrations. The results serve as motivation for the combined parameter and state estimation LMM EnKF algorithm derived in the work [157].

Overall, this thesis provides novel contributions in the formulation of sequential Monte Carlo techniques for parameter estimation in differential equations. By deriving a systematic method of estimating the innovation term in the time evolution update of the Bayesian filtering problem, we are able to accurately estimate blind system components as well as unknown parameters of governing systems of ODEs within the framework of particle filters and ensemble Kalman filters.

Bibliography

[1] R. V. M. Zahar, editor. *Approximation and Computation: A Festschrift in Honor of Walter Gautschi (Proceedings of the Purdue Conference, December 25, 1993)*, volume 119 of *International Series of Numerical Mathematics*. Birkhauser, Boston, 1994.

[2] D. Calvetti and E. Somersalo. Large-scale statistical parameter estimation in complex systems with an application to metabolic models. *Multiscale Model. Simul.*, 5:1333–1366, 2006.

[3] H. Miao, C. Dykes, L. M. Demeter, and H. Wu. Differential equation modeling of HIV viral fitness experiments: model identification, model selection and multimodel inference. *Biometrics*, 65:292–300, 2009.

[4] H. Haario, M. Laine, M. Lehtinen, E. Saksman, and J. Tamminen. Markov chain Monte Carlo methods for high dimensional inversion in remote sensing. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 66:591–607, 2004.

[5] J. P. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Applied Mathematical Sciences. Springer, New York, 2005.

[6] A. Seppänen, M. Vauhkonen, P. J. Vauhkonen, E. Somersalo, and J. P. Kaipio. State estimation with fluid dynamical evolution models in process tomography - an application to impedance tomography. *Inverse Problems*, 17:467–483, 2001.

[7] D. Calvetti and E. Somersalo. *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing*. Surveys and Tutorials in the Applied Mathematical Sciences. Springer, New York, 2007.

[8] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Mathematics and Its Applications. Kluwer Academic Publishers, Boston, 2000.

[9] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Mathematical Modeling and Computation. SIAM, Philadelphia, 1998.

[10] A. Arnold, D. Calvetti, and E. Somersalo. Linear multistep methods, particle filtering and sequential Monte Carlo. *Inverse Problems*, 29(8):085007, 2013.

[11] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[12] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, Philadelphia, 2007.

[13] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations, Second Edition*. Cambridge Texts in Applied Mathematics. Cambridge University Press, New York, 2009.

[14] E. Süli and D. F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, New York, 2003.

[15] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Mathematica Scandinavica*, 4:33–53, 1956.

[16] B. Øksendal. *Stochastic Differential Equations: An Introduction with Applications, Fifth Edition*. Springer-Verlag, New York, 2003.

[17] P. Billingsley. *Probability and Measure, Third Edition*. John Wiley & Sons, New York, 1995.

[18] G. Casella and R. L. Berger. *Statistical Inference, Second Edition*. Duxbury, Pacific Grove, CA, 2002.

[19] G. B. Folland. *Real Analysis: Modern Techniques and Their Applications, Second Edition*. John Wiley & Sons, New York, 1999.

[20] W. J. Krzanowski. *Principles of Multivariate Analysis: A User's Perspective*. Oxford Statistical Science. Oxford University Press, New York, 1988.

[21] G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010.

[22] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. Springer, New York, 2001.

[23] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, New York, 2001.

[24] E. Nummelin. MCs for MCMCists. *Int. Stat. Rev.*, 70:215–240, 2002.

[25] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22:1701–1728, 1994.

[26] A. F. M. Smith and G. O. Roberts. Bayesian computation via Gibbs sampler and related Markov chain Monte Carlo methods. *J. Roy. Stat. Soc. B*, 55:3–23, 1993.

[27] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1091, 1953.

[28] W. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[29] A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *J. Amer. Stat. Assoc.*, 85:398–409, 1990.

[30] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[31] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods, Second Edition*. Springer Texts in Statistics. Springer, New York, 2004.

[32] G. O. Roberts and W. R. Gilks. Convergence of adaptive direction sampling. *Journal of Multivariate Analysis*, 49:287–298, 1994.

[33] J. Besag and P. Green. Spatial statistics and Bayesian computation. *J. Roy. Stat. Soc. B*, 55:25–37, 1993.

[34] C. Geyer. Practical Markov chain Monte Carlo. *Stat. Sci.*, 7:473–511, 1992.

[35] D. Calvetti and E. Somersalo. Statistical methods in imaging. In O. Scherzer, editor, *Handbook of Mathematical Methods in Imaging*, pages 913–957. Springer, New York, 2011.

[36] W. A. Woyczyński. *A First Course in Statistics for Signal Analysis, Second Edition*. Birkhäuser, Boston, 2011.

[37] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 197–223. Springer, New York, 2001.

[38] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer-Verlag, New York, 1986.

[39] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Comput. Statist.*, 14:375–395, 1999.

[40] H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7:223–242, 2001.

[41] H. Haario, E. Saksman, and J. Tamminen. Componentwise adaptation for high dimensional MCMC. *Comput. Statist.*, 20:265–273, 2005.

[42] A. Mira. On Metropolis-Hastings algorithm with delayed rejection. *Metron*, 59:231–241, 2001.

[43] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: Efficient adaptive MCMC. *Stat. Comput.*, 16:339–354, 2006.

[44] C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Stat. Comput.*, 18:343–373, 2008.

[45] A. G. Gelman, G. O. Roberts, and W. R. Gilks. Efficient Metropolis jumping rules. In J. M. Bernardo, J. O. Berger, A. F. David, and A. F. M. Smith, editors, *Bayesian Statistics V*, pages 599–608. Oxford University Press, Oxford, 1996.

[46] W. R. Gilks and G. O. Roberts. Strategies for improving MCMC. In W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, pages 75–88. Chapman & Hall, London, 1995.

[47] P. H. Peskun. Optimum Monte Carlo sampling using Markov chains. *Biometrika*, 60:607–612, 1973.

[48] G. O. Roberts and O. Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodol. Comput. Appl. Probab.*, 4:337–358, 2003.

[49] R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X. L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–163. CRC, New York, 2010.

[50] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 73:1–37, 2011.

[51] A. Christen and C. Fox. MCMC using an approximation. *J. Comp. Graph. Stat.*, 14:795–810, 2005.

[52] J. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 69:741–796, 2007.

[53] J. M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM J. Sci. Comput.*, 3:28–46, 1982.

[54] B. Calderhead, M. Girolami, and N. D. Lawrence. Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. *Adv. Neural Inf. Process. Syst.*, 21:217–224, 2009.

[55] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *IEEE Proceedings*, 95:899–924, 2007.

[56] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*, pages 656–704. Oxford University Press, Oxford, 2011.

[57] P. Fearnhead. Computational methods for complex stochastic systems: a review of some alternatives to MCMC. *Statist. Comput.*, 18:151–171, 2008.

[58] M. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filters. *J. Amer. Statist. Assoc.*, 94:590–599, 1999.

[59] M. West. Approximating posterior distributions by mixtures. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 55:409–422, 1993.

[60] M. West. Mixture models, Monte Carlo, Bayesian updating and dynamic models. In J. H. Newton, editor, *Computing Science and Statistics: Proceedings*

of the 24th Symposium on the Inteface, pages 325–333. Interface Foundation of America, Fairfax Station, VA, 1993.

[61] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140:107–113, 1993.

[62] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[63] A. J. Majda and J. Harlim. *Filtering Complex Turbulent Systems*. Cambridge University Press, New York, 2012.

[64] G. Evensen. *Data Assimilation: The Ensemble Kalman Filter, Second Edition*. Springer-Verlag, New York, 2009.

[65] D. Calvetti and E. Somersalo. Subjective knowledge or objective belief? An oblique look to Bayesian methods. In L. Biegler, G. Biros, O. Ghattas, M. Heinkenschloss, D. Keyes, B. Mallick, Y. Marzouk, L. Tenorio, B. van Bloemen Waanders, and K. Willcox, editors, *Large-Scale Inverse Problems and Quantification of Uncertainty*, pages 33–70. John Wiley & Sons, Chichester, UK, 2010.

[66] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.*, 82:35–45, 1960.

[67] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23:53–60, 1981.

[68] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Trans. ASME J. Basic Eng.*, 83:95–108, 1961.

[69] A. Eliassen. Provisional report on calculation of spatial covariance and autocorrelation of the pressure field. In L. Bengtsson, M. Ghil, and E. Kallen,

editors, *Dynamic Meteorology: Data Assimilation Methods*, pages 319–330. Springer-Verlag, New York, 1954.

[70] L. S. Gandin. *Objective analysis of meteorological fields*. Gidrometerologicheskoe Izdatelstvo. (English translation by Israeli Program for Scientific Translations, Jerusalem, 1965), 1963.

[71] A. C. Lorenc. Analysis methods for numerical weather prediction. *Quart. J. Roy. Meteorol. Soc.*, 112:1177–1194, 1986.

[72] D. F. Parrish and J. C. Derber. The National Meteorological Center's spectral statistical-interpolation analysis system. *Monthly Weather Rev.*, 120:1747–1763, 1992.

[73] P. Courtier, E. Andersson, W. Heckley, J. Pailleux, D. Vasiljevic, M. Hamrud, A. Hollingsworth, F. Rabier, and M. Fisher. The ECMWF implementation of three dimensional variational assimilation (3D-Var). I: Formulation. *Quart. J. Roy. Meteorol. Soc.*, 124:1783–1807, 1998.

[74] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[75] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. SIAM, Philadelphia, 2003.

[76] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. SIAM, Philadelphia, 1996.

[77] C. T. Kelley. *Iterative Methods for Optimization*. SIAM, Philadelphia, 1999.

[78] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99:10143–10162, 1994.

[79] G. Evensen. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. *Journal of Geophysical Research*, 97:17905–17924, 1992.

[80] G. Evensen. Open boundary conditions for the extended Kalman filter with a quasi-geostrophic model. *Journal of Geophysical Research*, 98:16529–16546, 1993.

[81] G. Burgers, P. J. van Leeuwen, and G. Evensen. Analysis scheme in the ensemble Kalman filter. *Monthly Weather Rev.*, 126:1719–1724, 1998.

[82] G. Evensen. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[83] J. S. Whitaker and T. M. Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Rev.*, 130:1913–1924, 2002.

[84] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, San Diego, CA, 1970.

[85] J. L. Anderson and S. L. Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Rev.*, 127:2741–2758, 1999.

[86] E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corrazza, E. Kalnay, and J. A. Yorke. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus A*, 56:415–428, 2004.

[87] J. L. Anderson. An adaptive covariance inflation error correction algorithm for ensemble filters. *Tellus A*, 59:210–224, 2007.

[88] H. L. Mitchell and P. L. Houtekamer. An adaptive ensemble Kalman filter. *Monthly Weather Rev.*, 128:416–433, 2000.

[89] T. M. Hamill, J. S. Whitaker, and C. Snyder. Distance-dependent filtering of background-error covariance estimates in an ensemble Kalman filter. *Monthly Weather Rev.*, 129:2776–2790, 2001.

[90] P. L. Houtekamer and H. L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Rev.*, 129:123–137, 2001.

[91] A. J. Majda, J. Harlim, and B. Gershgorin. Mathematical strategies for filtering turbulent dynamical systems. *Discrete and Continuous Dynamical Systems*, 27:441–486, 2010.

[92] H. Moradkhani, S. Sorooshian, H. Gupta, and P. Houser. Dual state-parameter estimation of hydrological models using ensemble Kalman filter. *Advances in Water Resources*, 28:135–147, 2005.

[93] Y. Gu and D. S. Oliver. The ensemble Kalman filter for continuous updating of reservoir simulation models. *Journal of Energy Resources Technology*, 128:79–87, 2006.

[94] G. Naevdal, L. M. Johnsen, S. I. Aanonsen, and E. H. Vefring. Reservoir monitoring and continuous model updating using ensemble Kalman filter. *SPE Journal*, 10:66–74, 2005.

[95] L. Heidari, V. Gervais, M. Le Ravalec, and H. Wackernagel. History matching of petroleum reservoir models by the ensemble Kalman filter and parameterization methods. *Computers & Geosciences*, 55:84–95, 2013.

[96] C. Snyder and F. Zhang. Assimilation of simulated doppler radar observations with an ensemble Kalman filter. *Monthly Weather Rev.*, 131:1663–1677, 2003.

[97] A. C. Lorenc. The potential of the ensemble Kalman filter for NWP - a comparison with 4D-Var. *Q.J.R. Meteorol. Soc.*, 129:3183–3203, 2003.

[98] H. L. Mitchell, P. L. Houtekamer, and G. Pellerin. Ensemble size, balance, and model-error representation in an ensemble Kalman filter. *Monthly Weather Rev.*, 130:2791–2808, 2002.

[99] P. L. Houtekamer, H. L. Mitchell, G. Pellerin, M. Buehner, M. Charron, L. Spacek, and B. Hansen. Atmospheric data assimilation with an ensemble Kalman filter: results with real observations. *Monthly Weather Rev.*, 133:604–620, 2005.

[100] E. Kalnay, H. Li, T. Miyoshi, S. Yang, and J. Ballabrera-Poy. 4-D-Var or ensemble Kalman filter? *Tellus*, 59:758–773, 2007.

[101] P. Courtier, J. N. Thpaut, and A. Hollingsworth. A strategy for operational implementation of 4D-Var using an incremental approach. *Quart. J. Roy. Meteor. Soc.*, 120:1367–1387, 1994.

[102] P. Courtier. Dual formulation of variational assimilation. *Q. J. R. Meteorol. Soc.*, 123:2449–2461, 1997.

[103] E. Andersson, P. Courtier, C. Gaffard, J. Haseler, F. Rabier, P. Undé, and D. Vasiljevic. 3D-Var – the new operational analysis scheme. *ECMWF Newsletter*, 71:2–5, 1996.

[104] P. Gauthier, C. Charette, L. Fillion, P. Koclas, and S. Laroche. Implementation of a 3D variational data assimilation system at the Canadian Meteorological Centre. Part I: The global analysis. *Atmos.Ocean*, 37:103–156, 1999.

[105] S. Laroche, P. Gauthier, M. Tanguay, S. Pellerin, and J. Morneau. Impact of the different components of 4DVAR on the global forecast system of the Meteorological Service of Canada. *Monthly Weather Rev.*, 135:2355–2364, 2007.

[106] J. D. Kepert. Maths at work in meteorology. *Gazette of the Australian Mathematical Society*, 34:150–155, 2007.

[107] T. M. Hamill and C. Snyder. A hybrid ensemble Kalman filter–3D variational analysis scheme. *Monthly Weather Rev.*, 128:2905–2919, 2000.

[108] J. A. Hansen and L. A. Smith. Probabilistic noise reduction. *Tellus A*, 53:585–598, 2001.

[109] P. L. Houtekamer and H. L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Montly Weather Rev.*, 126:796–811, 1998.

[110] P. J. van Leeuwen. Comment on "Data assimilation using an ensemble Kalman filter technique". *Monthly Weather Rev.*, 127:1374–1377, 1999.

[111] P. L. Houtekamer and H. L. Mitchell. Reply. *Montly Weather Rev.*, 127:1378–1379, 1999.

[112] P. S. Maybeck. *Stochastic Models, Estimation, and Control, Volume 1.* Mathematics in Science and Engineering. Academic Press, New York, 1979.

[113] C. H. Bishop, B. J. Etherton, and S. J. Majumdar. Adaptive sampling with the ensemble transform Kalman filter. Part I: theoretical aspects. *Montly Weather Rev.*, 129:420–436, 2001.

[114] S. J. Majumdar, C. H. Bishop, B. J. Etherton, I. Szunyogh, and Z. Toth. Can an ensemble transform Kalman filter predict the reduction in forecast-error variance produced by targeted observations? *Q. J. R. Meteorol. Soc.*, 127:2803–2820, 2001.

[115] B. R. Hunt, E. J. Kostelich, and I. Szunyogh. Efficient data assimilation for spatiotemporal chaos: a local ensemble transform Kalman filter. *Physica D*, 230:112–126, 2007.

[116] X. Wang, C. H. Bishop, and S. J. Julier. Which is better, an ensemble of positivenegative pairs or a centered spherical simplex ensemble? *Monthly Weather Rev.*, 132:1590–1605, 2004.

[117] J. L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Rev.*, 129:2884–2903, 2001.

[118] M. K. Tippett, J. L. Anderson, C. H. Bishop, T. M. Hamill, and J. S. Whitaker. Ensemble square-root filters. *Monthly Weather Rev.*, 131:1485–1490, 2003.

[119] P. J. van Leeuwen and G. Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Rev.*, 124:2898–2913, 1996.

[120] G. Evensen and P. J. van Leeuwen. An ensemble Kalman smoother for nonlinear dynamics. *Monthly Weather Rev.*, 128:1852–1867, 2000.

[121] D. T. Pham, J. Verron, and M. C. Roubaud. A singular evolutive extended Kalman filter for data assimilation in oceanography. *J. Marine Sys.*, 16:323–340, 1998.

[122] P. Brasseur, J. Ballabrera, and J. Verron. Assimilation of altimetric data in the mid-latitude oceans using the SEEK filter with an eddy-resolving primitive equation model. *J. Marine Sys.*, 22:269–294, 1999.

[123] V. Carmillet, J. M. Brankart, P. Brasseur, H. Drange, and G. Evensen. A singular evolutive extended Kalman filter to assimilate ocean color data in a coupled physical-biochemical model of the North Atlantic. *Ocean Modelling*, 3:167–192, 2001.

[124] M. Verlaan and A. W. Heemink. Nonlinearity in data assimilation applications: a practical method for analysis. *Monthly Weather Rev.*, 129:1578–1589, 2001.

[125] P. F. J. Lermusiaux and A. R. Robinson. Data assimilation via error subspace statistical estimation. Part I: theory and schemes. *Monthly Weather Rev.*, 127:1385–1407, 1999.

[126] R. Chen and J. S. Liu. Mixture Kalman filters. *J. Royal Statistical Soc. B*, 62:493–508, 2000.

[127] D. B. Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm. *Journal of the American Statistical Association*, 82:543–546, 1987.

[128] T. Li, T. P. Sattar, and S. Sun. Deterministic resampling: unbiased sampling to avoid sample impoverishment in particle filters. *Signal Processing*, 92:1637–1645, 2012.

[129] P. Stavropoulos and D. M. Titterington. Improved particle filters and smoothing. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 295–317. Springer, New York, 2001.

[130] C. Musso, N. Oudjane, and F. Le Gland. Improving regularised particle filters. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 247–271. Springer, New York, 2001.

[131] C. M. Carvalho and H. F. Lopes. Simulation-based sequential analysis of Markov switching stochastic volatility models. *Comput. Statist. Data Anal.*, 51:4526–4542, 2007.

[132] E. Somersalo, A. Voutilainen, and J. Kaipio. Non-stationary MEG by Bayesian filtering. *Inverse Problems*, 19:1047–1064, 2003.

[133] A. Golightly and D. J. Wilkinson. Bayesian sequential inference for stochastic kinetic biochemical network models. *J. Comp. Biol.*, 13:838–851, 2006.

[134] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations, Third Edition*. Springer-Verlag, Berlin, 1999.

[135] A. Arnold, D. Calvetti, and E. Somersalo. Particle filter SMC algorithms for vectorized and parallel environments. *Submitted*, 2014.

[136] L. R. Scott, T. Clark, and B. Bagheri. *Scientific Parallel Computing*. Princeton, Princeton, NJ, 2005.

[137] C. Lieberman and K. Willcox. Goal-oriented inference: approach, linear theory, and application to advection diffusion. *SIAM Review*, 55:493–519, 2013.

[138] D. J. Wilkinson. Parallel Bayesian computation. In E. J. Kontoghiorghes, editor, *Handbook of Parallel Computing and Statistics*, pages 477–508. Chapman & Hall/CRC, Boca Raton, FL, 2005.

[139] R. Ren and G. Orkoulas. Parallel Markov chain Monte Carlo simulations. *The Journal of Chemical Physics*, 126:211102, 2007.

[140] A. Arnold, D. Calvetti, and E. Somersalo. Astrocytic tracer dynamics estimated from [1$-^{11}$C]-acetate PET measurements. *Submitted*, 2014.

[141] A. Gjedde, W. R. Bauer, and D. F. Wong. *Neurokinetics: The Dynamics of Neurobiology In Vivo*. Springer Verlag, New York, 2011.

[142] A. Gjedde, S. Keiding, H. Vilstrup, and P. Iversen. No oxygen delivery limitation in hepatic encephalopathy. *Metab. Brain Dis.*, 25:57–63, 2010.

[143] P. Iversen, M. Sørensen, L. K. Bak, H. S. Waagepetersen, M. S. Vafaee, P. Borghammer, K. Mouridsen, S. B. Jensen, H. Vilstrup, A. Schousboe, P. Ott, A. Gjedde, and S. Keiding. Low cerebral oxygen consumption and blood flow in patients with cirrhosis and an acute episode of hepatic encephalopathy. *Gastroenterology*, 136:863–871, 2009.

[144] M. T. Wyss, P. J. Magistretti, A. Buck, and B. Weber. Mini review: labeled acetate as a marker of astrocytic metabolism. *J. Cereb. Blood Flow Metab.*, 31:1668–1674, 2011.

[145] L. Hertz, L. Peng, and G. A. Dienel. Energy metabolism in astrocytes: high rate of oxidative metabolism and spatiotemporal dependence on glycolysis/glycogenolysis. *J. Cereb. Blood Flow Metab.*, 27:219–249, 2007.

[146] R. A. Wanievski and D. L. Martin. Preferential utilization of acetate by astrocytes is attributable to transport. *J. Neurosci.*, 18:5225–5233, 1998.

[147] H. Iida, I. Kanno, S. Miura, M. Murakami, K. Takahashi, and K. Uemura. Error analysis of quantitative cerebral blood flow measurement using $H_2$15O autoradiography and positron emission tomography, with respect to the dispersion of the input function. *J. Cereb. Blood Flow Metab.*, 6:536–545, 1986.

[148] H. Iida, T. Jones, and S. Miura. Modeling approach to eliminate the need of separate arterial plasma in oxygen-15 inhalation positron emission tomography. *J. Nucl. Med.*, 34:1333–1340, 1993.

[149] N. Kudomi, H. Watabe, T. Hayashi, and H. Iida. Separation of input function for rapid measurement of quantitative $CMRO_2$ and CBF in a single PET scan with a dual tracer administration method. *Phys. Med. Biol.*, 52:1893–1908, 2007.

[150] E. Meyer. Simultaneous correction for tracer arrival delay and dispersion in CBF measurements by the $H_2$15O autoradiographic method and dynamic PET. *J. Nucl. Med.*, 30:1069–1078, 1989.

[151] K. Mouridsen, K. Friston, N. Hjort, L. Gyldensted, L. Østergaard, and S. Kirbel. Bayesian estimation of cerebral perfusion using a physiological model of microvasculature. *NeuroImage*, 33:570–579, 2006.

[152] P. Iversen, K. Mouridsen, M. B. Hansen, S. B. Jensen, M. Sørensen, L. K. Bak, H. Waagepetersen, A. Schousboe, P. Ott, H. Vilstrup, S. Keiding, and A. Gjedde. Oxidative metabolism of astrocytes is not reduced in hepatic encephalopathy: A PET study with acetate in humans. *Journal of Cerebral Blood Flow and Metabolism*, Submitted, 2014.

[153] L. Armijo. Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific J. Math*, 16:1–3, 1966.

[154] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. SIAM, Philadelphia, 2007.

[155] K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57:519–530, 1970.

[156] K. V. Mardia. Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies. *Sankhyā: The Indian Journal of Statistics, Series B*, 36:115–128, 1974.

[157] A. Arnold, D. Calvetti, and E. Somersalo. Parameter estimation for deterministic dynamical systems via ensemble Kalman filter. *In preparation*, 2014.

[158] G. Evensen. Advanced data assimilation for strongly nonlinear dynamics. *Monthly Weather Rev.*, 125:1342–1354, 1997.

[159] G. Evensen and N. Fario. A weak constraint variational inverse for the Lorenz equations using substitution methods. *J. Meteor. Soc. Japan*, 75:229–243, 1997.

[160] R. N. Miller, M. Ghil, and F. Gauthiez. Advanced data assimilation in strongly nonlinear dynamical systems. *J. Atmos. Sci.*, 51:1037–1056, 1994.

[161] D. Baroudi, J. P. Kaipio, and E. Somersalo. Dynamical electric wire tomography: time series approach. *Inverse Problems*, 14:799–813, 1998.

[162] J. P. Kaipio and E. Somersalo. Nonstationary inverse problems and state estimation. *J. Inv. Ill-Posed Probl.*, 7:273–282, 1999.