École polytechnique fédérale de Lausanne

Master project, 2018, Mathematics section

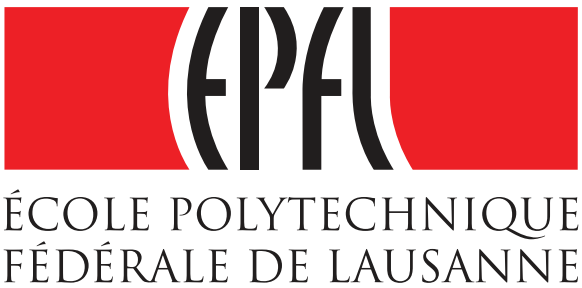
Master in Computational Science and Engineering

Computational Mathematics and Numerical Analysis Group

Probabilistic methods for differential equations: adaptivity and Bayesian inverse problems.

Author:
Aleksa Stanković

Supervisors:
Professor Assyr Abdulle
Giacomo Garegnani



Submitted on January 19, 2018

Abstract

Probabilistic numerical methods for ordinary differential equations have attracted research interest in recent years. In this thesis we provide an overview of the existing approaches which extend Runge-Kutta methods in order to describe the solution of ordinary differential equations as a random variable.

Particular attention is devoted to studying the uncertainty introduced by probabilistic methods. In particular, we analyze whether the output of these methods carries some information about the numerical error. In particular, an adaptive time-stepping scheme which relies solely on statistical information given by the probabilistic solver is introduced. While this adaptive scheme does not outperform standard approaches available in existing literature, it nevertheless preserves the convergence properties of the underlying Runge-Kutta method, and thus shows that random variables produced by probabilistic numerical methods carry information about the error.

Probabilistic numerical methods for ordinary differential equations are particularly effective in the framework of Bayesian inverse problems, which are discussed in this thesis. Since the solution of Bayesian inverse problems is given as a probability measure which is complex and often high dimensional, we consider Markov Chain Monte Carlo and Sequential Monte Carlo methods in order to sample from it. Furthermore, we discuss two Metropolis-Hastings samplers which are used in conjunction with probabilistic numerical methods for solving Bayesian inverse problems: the pseudo-marginal Metropolis-Hastings (PMMH) and the Monte Carlo within Metropolis (MCWM). To the best knowledge of the author, we provide the first result on the approximation error of the MCWM algorithm when used to sample solutions of Bayesian inverse problems for ODEs solved by probabilistic Runge-Kutta solvers with respect to the step-size employed for time integration.

Finally, we consider infinite-dimensional Bayesian inverse problems, and discuss the computational framework required for their solution.

Acknowledgements

I would like first to thank Professor Abdulle for finding an interesting topic which kept me captivated for months, and supporting me throughout the project's duration.

I am also very thankful to Giacomo Garegnani, who invested a lot of time in supervising my work and helping whenever it was necessary. His support and ideas were of invaluable significance for completing this project.

I am also very grateful to Prof. Dr. Marcus Grote, who has agreed to serve as my examiner.

I would like to express the gratitude to all of my fellows I have meet during the days I have spent in Lausanne. I have spent the most amazing two years with all of you, and I hope we stay in touch no matter what life paths we undertake.

Last but not least, I would like to thank my family for providing me with unfailing support and continuous encouragement throughout my whole life. Any my accomplishment would not have been possible without them. Thank you.

Table of contents

Chapter 1

Introduction

Probabilistic numerical methods for ordinary differential equations have been an active research area in recent years. Two major approaches have been developed so far: one that extends Runge-Kutta methods by introducing stochastic terms in their design [3, 8], and another which exploits Bayesian analysis on Gaussian processes [19].

In this thesis we focus on the first family of methods. In order to lay the basis of our discussion, in Chapter 2 we introduce Runge-Kutta methods, study their stability properties, and present two explicit stabilized Runge-Kutta schemes: RKC [27] and ROCK2 [4]. Furthermore, we outline an adaptive time-stepping scheme suitable for these methods.

Building on this discussion, in Chapter 3 we give an overview of probabilistic Runge-Kutta methods which describe the solution of an ordinary differential equation in form of a random variable. Firstly, we provide a general framework in which the existing methods can be cast into. Then, this setting is used to introduce the additive noise (AN-RK) [8] and the random time-stepping Runge-Kutta (RTS-RK) [3] methods. In the last section of this chapter we focus on adaptive time-stepping strategies for the RTS-RK method. We extend the adaptive algorithm from Chapter 2 for which satisfactory numerical results are obtained. Furthermore, an adaptive time-stepping scheme which uses only statistical information given by the RTS-RK method to adapt time-steps is analyzed. While the performance of this scheme is not advantageous to the existing approaches, it nonetheless conserves the convergence order of the underlying Runge-Kutta method.

Apart for uncertainty quantification, probabilistic numerical schemes for ODEs are particularly useful in the framework of Bayesian inverse problems. Therefore, in Chapter 4 we give a brief overview of Bayesian inverse problems. The exposition mostly follows the approaches given in [12, 18, 26]. Moreover, we introduce Markov Chain Monte Carlo and sequential Monte Carlo methods for sampling the solution of Bayesian inverse problem which is given as a probability measure $\mu^{\mathcal{Y}}$. In order to use probabilistic numerical solvers we introduce two more sampling schemes, i.e., the pseudo-marginal Metropolis-Hastings (PMMH) and the Monte Carlo within Metropolis (MCWM). We study the convergence properties of MCWM in Section 4.5.1 in the limit case $h \to 0$, where h is the step-size of the probabilistic numerical method.

Finally, in Chapter 5 we consider Bayesian problems in the infinite dimensional setting and introduce necessary adaptations for the approach from Chapter 4. Furthermore, we consider the parabolic Brusselator partial differential equation, and show how its solution can be approximated by the method of lines. In the last section of Chapter 5 we consider the problem of recovering the initial condition of this problem by using the methodology developed for Bayesian inverse problems.

Numerical experiments are discussed at the end of each chapter in order to corroborate theoretical results. The code which was used to generate these results can be found at https://github.com/StankovicAleksa/master-project.

Chapter 2

Runge-Kutta Methods and Stiff Ordinary Differential Equations

In this chapter we present Runge-Kutta methods, which are an important class of numerical schemes for solving ordinary differential equations (ODEs). The importance of ODEs stems from their application in describing various processes studied in applied and social sciences.

In Section 2.1 we introduce general terms and definitions related to ODEs and Runge-Kutta methods, which will be used in the rest this thesis.

In Section 2.2, we discuss the stability properties of Runge-Kutta methods and introduce stiff problems. Building on this exposition, in Section 2.3 we discuss explicit methods, which are convenient for solving dynamical systems whose Jacobian of the driving term has eigenvalues ”close” to the negative real axis in the complex plane.

In Section 2.4, we present adaptive time-stepping algorithms for Runge-Kutta methods. In particular, we discuss the embedding approach which is the most suitable in the context of stabilized explicit methods. Furthermore, we show how to construct embedded methods for the explicit stabilized integrators introduced in Section 2.3. We also explain how this local error estimate can be combined with conveniently chosen proportional–integral–derivative (PID) controller to devise an adaptive time-stepping algorithm for explicit stabilized methods.

2.1 Introduction

Consider a dynamical system

$$\dot{y}(t) = f(t, y), \tag{2.1}$$

where $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ and $y : \mathbb{R} \to \mathbb{R}^d$. If (2.1) is given an initial condition, it is called an initial value problem (IVP), i.e.,

$$\dot{y}(t) = f(t, y), \qquad y(t_0) = y_0. \tag{2.2}$$

In the most common case, the function f and the initial condition y_0 are known, and we are interested in finding $y(T)$, the solution to this problem at some time $T > t_0$.

For the sake of simplifying the notation, we assume $t_0 = 0$ in (2.2) and that f does not depend explicitly on t. Thus, we consider $f : \mathbb{R}^d \to \mathbb{R}^d$ and the dynamical system (2.1) reduces to

$$\dot{y}(t) = f(y), \tag{2.3}$$

with the initial value problem associated to it being

$$\dot{y}(t) = f(y), \qquad y(0) = y_0. \tag{2.4}$$

Generality is not lost under these assumptions because (2.2) is equivalent to the autonomous system

$$\dot{x}(t) = g(x), \qquad x(0) = x_0, \tag{2.5}$$

where $x(t) = [y(t + t_0)^\top, t + t_0]^\top$, $g(y) = [f(y)^\top, 1]^\top$, and $x_0 = [y_0^\top, t_0]^\top$.

In order to ensure existence and uniqueness of the solution to the IVP, throughout the rest of this work we make the following assumption.

Assumption 2.1. *The function f is locally Lipschitz continuous.*

It is convenient to express the solution of (2.4) in terms of the flow of the differential equation.

Definition 2.1. *Consider the dynamical system (2.3), and let Assumption 2.1 hold. The flow $\varphi : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ of (2.3) is defined as*

$$\varphi(t, y_0) := y(t), \tag{2.6}$$

where $y(t)$ is the solution of the initial value problem (2.4) at time $t \in \mathbb{R}$. The semi-group notation $\varphi_t(y_0) := \varphi(t, y_0)$ is used as well.

In most cases the solution $y(T)$ to the initial value problem (2.2) does not admit a closed-form expression, and therefore we rely on numerical schemes to estimate its value. Let us consider a numerical method that approximates the solution $y(t)$ at the points $t \in \{t_n\}_{n=1}^N$, $0 = t_0 < t_1 < \ldots < t_N = T$. In particular, we consider numerical methods which can be characterized by their numerical flow $\Psi(h, y) : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$, which is used to calculate the approximations y_n of the solution $y(t)$ at points $\{t_n\}_{n=1}^N$, by iteratively applying

$$y_{n+1} = \Psi(h_n, y_n), \qquad n = 0, 1, \ldots, N - 1, \tag{2.7}$$

with $h_n = t_{n+1} - t_n$ being the step-sizes of the scheme. We also employ the notation $\Psi_h(y) := \Psi(h, y)$ for the numerical flow. Since we have $t_N = T$, the last iteration gives us the value of interest, namely $y(T)$.

It is common to choose fixed step-sizes by setting N beforehand and defining $h_i = h = T/N, t_n = nh$. Note that in this case the values y_n can be expressed as $y_n = \Psi_h^n(y_0)$, where $\Psi_h^n = \Psi_h \circ \ldots \circ \Psi_h$ is the composition of n numerical flows Ψ_h.

Since we consider numerical integrator of ODEs which are uniquely defined by their flow, we will always introduce a method just by showing how $y_1 = \Psi_h(y_0)$ can be constructed. For example, the simplest Runge-Kutta method for ODEs, the Euler explicit method, can be formulated as

$$y_1 = y_0 + hf(y_0). \tag{2.8}$$

A fundamental property of numerical integrators for ordinary differential equations is their order of convergence.

Definition 2.2. *A numerical method $\Psi : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d$ has order of convergence p if and only if for any $T \in \mathbb{R}^+$ there exists a constant C independent of h such that*

$$\|\Psi_h^N(y_0) - \varphi_T(y_0)\| \leq Ch^p, \quad \forall y_0 \in \mathbb{R}^d, \tag{2.9}$$

where $h = T/N$ is sufficiently small.

The order of convergence is closely related to the local error of a method, as can be seen from the following result.

Theorem 2.1. *Consider the IVP (2.2) and a numerical method Ψ_h. If there exists a constant C, which does not depend on h, such that*

$$\|\Psi_h(y_0) - \varphi_h(y_0)\| \leq Ch^{p+1}, \quad \forall y_0 \in \mathbb{R}^d, \tag{2.10}$$

then the numerical method has order p.

This theorem is a practical tool for computing the order of convergence of any given method. For example, in the case of the Euler explicit method, the Taylor expansion around any point y_0 is

$$\varphi_h(y_0) = y_0 + hf(y_0) + O(h^2), \tag{2.11}$$

and therefore (2.10) holds for $p = 1$. Thus, by Theorem 2.1, the Euler explicit method has order of convergence 1.

The Euler explicit method belongs to a class of Runge-Kutta methods, which are defined as follows.

Definition 2.3. *Consider the autonomous system (2.3), let $s \in \mathbb{N}$ and let $a_{ij}, b_i \in \mathbb{R}, 1 \leq i, j \leq s$. An s-stage Runge-Kutta method is defined by*

$$\begin{aligned} k_i &= f\left(y_0 + h\sum_{j=1}^{s} a_{ij}k_j\right), \quad i = 1, \dots, s, \\ y_1 &= y_0 + h\sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{2.12}$$

The vectors $k_i \in \mathbb{R}^d, i = 1, \dots, s$, are called the internal stages of the method.

By setting $s = 1, a_{11} = 0, b_1 = 1$, we can see that the Euler explicit method (2.8) is indeed a member of this class. Another classical Runge-Kutta method is Heun's method

$$\begin{aligned} k_1 &= f(y_0), \\ k_2 &= f(y_0 + hk_1), \\ y_1 &= y_0 + \frac{h}{2}(k_1 + k_2), \end{aligned} \tag{2.13}$$

which is of order 2. For both Heun's (2.13) and explicit Euler (2.8) methods we can iteratively compute the stages k_i by evaluating the function f with known arguments. All Runge-Kutta methods endowed with this property are called explicit. The methods for which this is not possible are called implicit. The simplest example of an implicit Runge-Kutta method is the implicit Euler method, which is defined as

$$\begin{aligned} k_1 &= f(y_0 + hk_1), \\ y_1 &= y_0 + hk_1. \end{aligned} \tag{2.14}$$

We can see that for this method the value of k_1 cannot by computed by simple evaluation of the function f, since the argument in $f(y_0 + hk_1)$ depends on k_1 itself. Thus, in general, we need to employ some additional numerical scheme (typically Newton's method) to solve the possibly nonlinear equation (2.14), which makes the evaluation of the numerical flow of implicit methods computationally more expensive. However, implicit methods can possess advantageous stability properties that explicit schemes cannot be endowed with, which makes them more suitable for certain types of equations.

2.2 Stability of Numerical Methods

In the previous section we showed how the solutions of initial value problems can be approximated by Runge-Kutta methods. Some of these methods are sensitive to small perturbations coming from either round-off errors or from the approximation error of the initial condition. Due to this sensitivity, they tend to magnify the errors, which renders their output of no practical use. In this section we

introduce the concept of stability for Runge-Kutta methods, which quantifies robustness against such small errors.

The stability properties of numerical integrators for ODEs are especially important in the context of *stiff* problems. There is no precise definition of stiffness, but in general stiff problems tend to be particularly expensive for unstable numerical methods, since they require extremely small step-size in order to produce significant solutions. Many stiff problems are distinguished by the large ratio between the largest and the smallest eigenvalue of their Jacobian, or by rapidly varying solutions.

We now introduce a concept of stability for Runge-Kutta methods. Consider the following linearization of the dynamical system (2.3)

$$\dot{w}(t) = \frac{\partial f}{\partial x}(t)w(t), \tag{2.15}$$

where $\partial f/\partial x(t)$ is the Jacobian matrix. Fixing the time parameter t, and diagonalizing the Jacobian leads to the following problem

$$\dot{y} = \lambda y, \qquad y_0 = 1, \tag{2.16}$$

where $y, \lambda \in \mathbb{C}$. This equation gives a description of how the local error propagates and is known as the Dahlquist equation [11]. Since this method is used for studying the stability properties of numerical schemes for ODEs, equation (2.16) is commonly called the test equation. Its explicit solution is $y(t) = e^{\lambda t}$. We will always assume that $\mathrm{Re}(\lambda) < 0$, which implies that $y(t) \to 0$ as $t \to +\infty$.

Consider a numerical scheme applied to the test equation (2.16) with a fixed step-size h. This scheme produces approximations y_n of the solution $y(t)$ at the times $t_n = nh$. Hence, in the limit for $n \to +\infty$, we have that the exact solution is correct only if $y_n \to 0$. We are particularly interested in schemes which exhibit this behavior, since they do not tend to accumulate errors, and thus show favorable stability properties.

A simple iteration of any Runge-Kutta method applied to the test equation (2.16) can be expressed as a rational function as stated in the following theorem.

Theorem 2.2. *Consider an s-stage Runge-Kutta method applied to the test equation (2.16). Then the numerical flow $y_1 = \Psi_h(y_0)$ can be expressed as*

$$y_1 = R(z)y_0, \tag{2.17}$$

where $z = \lambda h$, $R(z) = P(z)/Q(z)$ is a rational function, and polynomials P, Q are such that $deg(P), deg(Q) \leq s$. Furthermore, in case the Runge-Kutta method is explicit we have $Q \equiv 1$.

For example, the numerical flow of the explicit Euler's method can be expressed as $y_1 = (1+z)y_0$, while the implicit Euler method is given by $y_1 = 1/(1-z)y_0$. Since the n-th approximation y_n produced by a Runge-Kutta method applied to the test equation is $y_n = R(z)^n y_0$ and thus $y_n \xrightarrow{n\to\infty} 0$ if and only if $|R(z)| < 1$, it is natural to connect areas where $|R(z)| < 1$ with the numerical stability of a method.

Definition 2.4. *The stability region of Runge-Kutta method is the set*

$$\{z \in \mathbb{C} | Re(z) < 0 \text{ and } |R(z)| < 1\}, \tag{2.18}$$

where $R(z)$ is the rational function associated to the method as per Theorem 2.2.

The stability regions of the implicit Euler and the explicit Euler method are shown in Figure 2.1. We use a stability region to describe stability properties of a given Runge-Kutta method. In particular, methods with larger stability regions have better stability properties. Furthermore, in order to enforce stability, we choose methods for which the eigenvalues of $\partial f/\partial x$ around the trajectory of the true solution belong to their stability domains.

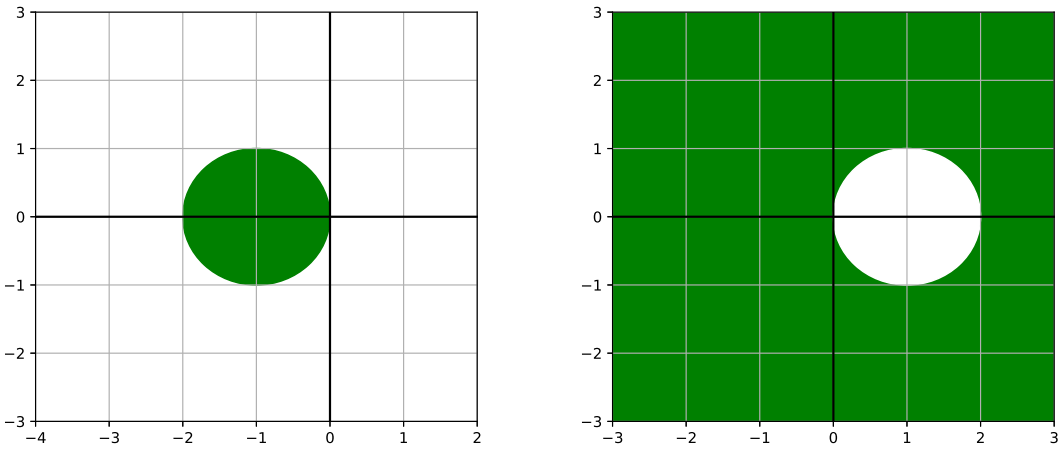


Fig. 2.1 Stability regions (green) of the Euler explicit method (left), and the Euler implicit method (right).

2.3 Explicit Stabilized Runge-Kutta Methods: RKC and ROCK2

As we have seen in the previous section, stability properties of Runge-Kutta methods are described by their stability domains. In this section we show how methods with extended stability domains around the negative real axis can be constructed. In particular, we focus on first and second order methods, for which an increase in the stage number s significantly enlarges the stability region around the negative real axis. We decide to cover only explicit methods, since their evaluation is much cheaper in comparison to implicit methods. These methods are a natural choice for approximating parabolic differential equations discretized in space, which we consider in Chapter 5.

Since we focus on explicit methods, the rational function $R(z)$ from Theorem 2.2 reduces to a polynomial $P(z)$. We will refer to $P(z)$ as stability polynomial in this chapter. Let us consider first order methods for now. Let us fix the number of stages s, and look for a method with the largest

possible interval $[-l_s, 0]$ included in its stability domain, where $l_s > 0$. It can be shown [27] that the biggest value $l_s = 2s^2$ is achieved in case the stability polynomial is a shifted Chebyshev polynomial

$$T_s\left(1 + \frac{z}{s^2}\right), \tag{2.19}$$

where polynomial T_s can be iteratively constructed with

$$\begin{aligned} T_0(x) &:= 1, \\ T_1(x) &:= x, \\ T_n(x) &:= 2xT_{n-1}(x) - T_{n-2}(x), \qquad n \geq 2. \end{aligned} \tag{2.20}$$

The method defined by this stability polynomial is known as the Runge-Kutta-Chebyshev (RKC) method [25, 27], and its stages can be computed with

$$\begin{aligned} k_0 &:= y_0, \\ k_1 &:= k_0 + \frac{h}{s^2} f(k_0), \\ k_j &:= \frac{2h}{s^2} f(k_{j-1}) + 2k_{j-1} - k_{j-2}, \quad j = 2, \ldots, s, \\ y_1 &:= k_s. \end{aligned} \tag{2.21}$$

The stability region of RKC method with $s = 5$ stages is plotted in Figure 2.2.

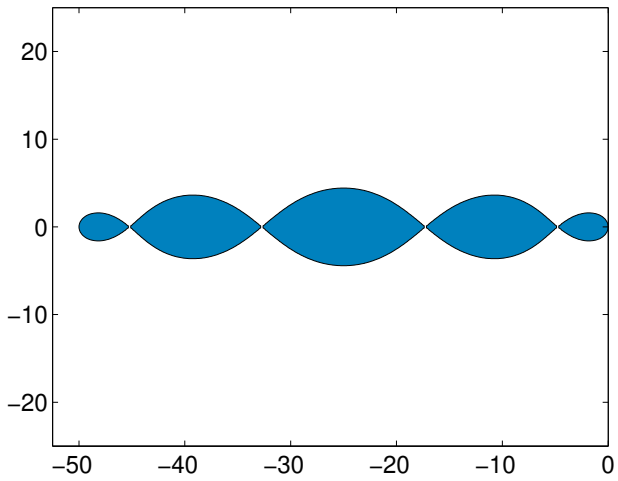


Fig. 2.2 Stability region of RKC method with $s = 5$ stages.

However, we can observe in Figure 2.2 that the stability region is very thin in the vertical direction at certain positions on the negative real axis. This can cause instabilities in case a small imaginary perturbation is introduced. We can overcome this issue via the technique of damping, at the expense

of slightly smaller l_s. We start by prescribing the damping parameter ν, $0 < \nu < 1$, and then modify the method such that it satisfies $|P(z)| < \nu$ in as big interval $[-l_s^\nu, -\varepsilon]$ as possible, where $l_s^\nu, \varepsilon \in \mathbb{R}$, and ε is small. In order to enforce these constraints the RKC method is modified as

$$\begin{aligned}
\mu &:= 1 - \nu, \qquad w_0 := 1 + \frac{\mu}{s^2}, \qquad w_1 := T_s(w_0)/T_s'(w_0), \\
k_0 &:= y_0, \\
k_1 &:= k_0 + h\frac{w_1}{w_0} f(k_0), \\
k_j &:= 2hw_1 \frac{T_{j-1}(w_0)}{T_j w_0} f(k_{j-1}) + w_0 \frac{T_{j-1}(w_0)}{T_j(w_0)} k_{j-1} - \frac{T_{j-2}(w_0)}{T_j(w_0)} k_{j-2}, \qquad 2 \le j \le s \\
y_1 &:= k_s.
\end{aligned} \tag{2.22}$$

The stability domain of the damped RKC method can be seen in Figure 2.3.

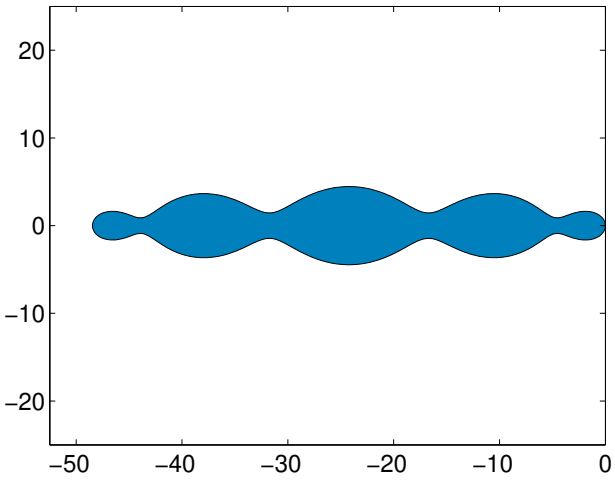


Fig. 2.3 Stability region of damped RKC method with $s = 5$ stages and damping parameter $\nu = 0.05$.

Let us now focus on second order explicit stabilized methods. In this case, the coefficients of the optimal stability polynomial $P(z)$ are given as elliptic integrals [17]. Methods which have approximations of these coefficients in their stability polynomials are known as Lebedev-type methods [20]. Alternatively, we can consider RKC2 [27] and ROCK2[1] [4] methods which have suboptimal stability polynomials, but they provide closed-form expressions for construction of stages in three-terms recurrence formulae.

We explore here ROCK2 method since it has larger stability region than RKC2 method. Furthermore, ROCK2 method has advantageous internal stability properties compared to Lebedev-type methods, since the three-term recurrence formula of ROCK2 method is internally stable. A detailed exposition of the ideas and derivation of ROCK2 method exceeds the scope of this thesis, so we here

[1] ROCK2 is an abbreviation for second order Orthogonal-Runge-Kutta-Chebyshev.

only give its definition

$$\begin{aligned} k_0 &:= y_0, \\ k_1 &:= k_0 + \mu_1 h f(k_0), \\ k_j &:= \mu_j h f(k_{j-1}) - \nu_j k_{j-1} - \kappa_j k_{j-2}, \qquad 2 \le j \le s-2 \\ k_{s-1} &:= k_{s-2} + 2\tau h f(k_{s-2}), \\ y_1 &:= k_{s-2} + (2\sigma - \frac{1}{2}) h f(k_{s-2}) + \frac{1}{2} h f(k_{s-1}). \end{aligned} \tag{2.23}$$

Values $\mu_j, \nu_j, \kappa_j, \sigma, \tau$ are coefficients which depend on s and are computed numerically. Let us remark that the ROCK2 method given in this form is already damped. However, in case we would like to fine-tune the stability of ROCK2 method, we can introduce a parameter $\alpha > 0$ and change the damping of ROCK2 method with

$$\begin{aligned} k_0 &:= y_0, \qquad k_1 := k_0 + \alpha\mu_1 h f(k_0), \\ k_j &:= \alpha\mu_j h f(k_{j-1}) - \nu_j k_{j-1} - \kappa_j k_{j-2}, \qquad 2 \le j \le s-2, \\ k_{s-1} &:= k_{s-2} + 2\tau_\alpha h f(k_{s-2}), \\ y_1 &:= k_{s-2} + (2\sigma_\alpha - \frac{1}{2}) h f(k_{s-2}) + \frac{1}{2} h f(k_{s-1}), \end{aligned} \tag{2.24}$$

where σ_α and τ_α satisfy

$$\sigma_\alpha = \frac{1-\alpha}{2} + \alpha\sigma, \quad \tau_\alpha = \frac{(\alpha-1)^2}{2} + 2\alpha(1-\alpha)\sigma + \alpha^2\tau. \tag{2.25}$$

The stability region of the ROCK2 method is shown in Figure 2.4.

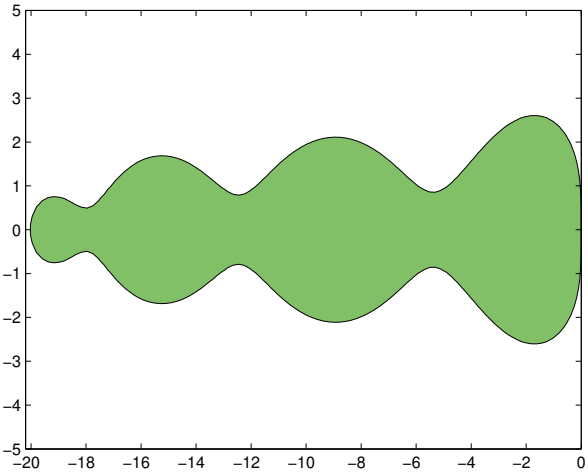


Fig. 2.4 Stability region of damped ROCK2 method with $s = 5$ and $\alpha = 1$.

2.4 Adaptive Time-Stepping

In the previous two sections we have introduced Runge-Kutta methods which approximate the solution $y(t)$ of an ODE by vectors $\{y_n\}_{n=1}^N$ at the points $\{t_n\}_{n=1}^N$. In the simplest case Runge-Kutta methods are used with a fixed step-size $h = T/N$, in which case $t_n = nh, n = 1, \dots, N$. In this case we can usually observe that the integration errors vary widely along the trajectory of the numerical solution. This is because the behavior of the equation is not uniform, i.e., certain parts of the domain are more problematic to integrate, and thus Runge-Kutta methods are less precise in these parts than in the others. Therefore, it makes sense to consider non-uniform step-sizes $\{h_n\}_{n=1}^N$, chosen in an adaptive fashion in order to control the numerical error. In particular, we design algorithms such that the error of the Runge-Kutta integrator is locally always close to a certain prescribed tolerance.

We start this section by considering embedded Runge-Kutta methods, which allow for the quantification of the error at every integration step. We focus on these methods since they are a particularly suitable choice for the explicit stabilized methods presented in Section 2.3. Then, we show how ideas from proportional–integral–derivative controller theory can be used to adapt step-sizes in order to construct effective adaptive time-stepping schemes. Finally, at the end of this section, we corroborate the exposition by considering a numerical test and showing that the adaptive time-stepping algorithm exhibits advantageous performance over the fixed time-stepping method.

Embedded Runge-Kutta methods provide at every step estimates y_n, y_n^*, given by two Runge-Kutta methods of order p and $p-1$, respectively. Then, the difference

$$E_n = |y_n - y_n^*| \tag{2.26}$$

between these estimates is used to approximate the error committed over a time-step. Intuitively, we consider the estimate of order p to be exact (or much closer to the exact solution) in comparison to the estimate of order $p-1$, and thus their difference indeed reflects the local error. Furthermore, the two estimates of embedded Runge-Kutta methods usually share most of the stages, which grants that the introduction of two estimates comes with only a minor expense in additional functional evaluations. Sometimes only the values of $\{b_i\}_{i=1}^s$ for the finishing procedures of the two Runge-Kutta methods are different, in which case embedding comes at no additional cost.

One example of an embedded method is given by the pair of Euler's explicit (2.8) and Heun's (2.13) method. In this case, we can reuse the stages of Heun's method to construct explicit Euler's method with no additional cost in function evaluations.

In case of ROCK2 method (2.23), we can add two more stages

$$k_{s-1}^{**} = k_{s-2} + 2\tau_\alpha h f(k_{s-2}), \tag{2.27}$$

$$y_1^* = k_{s-2} + (2 - \frac{1}{2}\frac{\sigma_\alpha}{\tau_\alpha})\sigma_\alpha h f(k_{s-2}) + \frac{1}{2}\frac{\sigma_\alpha^2}{\tau_\alpha} h f(k_{s-1}^{**}), \tag{2.28}$$

to construct an embedded method with two estimates y_1, y_1^* of order 2 and 1, respectively. It is important to note that estimate y_1^* shares the same stability properties of y_1.

Let us now show how we can control the step-sizes using the local error estimator. We follow the approach given in [1, 13], which uses ideas from theory of proportional–integral–derivative (PID) controllers to select/reject the time-steps and update step-sizes during the integration of ODEs.

We prescribe a tolerance $\tau > 0$ as an input to the adaptive time-stepping algorithm, in order to control the accuracy of the final output. In particular, decreasing the value of the tolerance τ gives a more precise output.

The adaptive time-stepping scheme chooses step-sizes $\{h_1, h_2, \dots\}$ iteratively. Alongside choosing the step-sizes, adaptive algorithm also constructs the approximations $\{y_1, y_2, \dots\}$. Thus, in order to construct the adaptive time-stepping algorithm we just need to explain how the step-size h_n is chosen given the approximations $y_1, y_2, \dots, y_n$ calculated with step-sizes $h_0, h_1, \dots, h_{n-1}$. Let us observe that since we are using an embedded Runge-Kutta method, we know the error estimates $\{E_i\}_{i=1}^n$ of the first n integration steps. Let us define the relative integration error err_i of i-th step, for $1 \le i \le n$, with

$$\mathrm{err}_i = \frac{E_n}{\sigma(\tau, h_{i-1}, y_{i-1}, y_i)}, \tag{2.29}$$

where σ is a function given by

$$\sigma(\tau, h_{i-1}, y_{i-1}, y_i) = \max(|y_{i-1}|, |y_i|)\tau + \tau. \tag{2.30}$$

The adaptive time-stepping algorithm proposes the value of the step-size h_n by calculating $h_{new} = \kappa(\mathrm{err}_n, \mathrm{err}_{n-1}, h_n, h_{n-1})$, with function κ being defined by

$$\kappa(\mathrm{err}_n, \mathrm{err}_{n-1}, h_n, h_{n-1}) = \mathrm{fac} \cdot h_n \left(\frac{1}{\mathrm{err}_n}\right)^{1/2} \frac{h_n}{h_{n-1}} \left(\frac{\mathrm{err}_{n-1}}{\mathrm{err}_n}\right)^{1/2}, \tag{2.31}$$

where $\mathrm{fac} = 0.8$ is a safety constant. Then, we propagate the approximation y_n using $y_{new} = \Psi_{h_{new}}(y_n)$, and calculate the relative integration error err_{new}. In case $\mathrm{err}_{new} < 1$ we accept the proposal by setting $h_n = h_{new}$, $y_{n+1} = y_{new}$, and move to the step $n+1$. Otherwise, we propose a smaller value h_{new} with

$$h_{new} := h_{new} \frac{1}{\sqrt{err_n}}, \tag{2.32}$$

recalculate $y_{new} = \Psi_{h_{new}}(y_n)$ and err_{new}, and check whether the condition for acceptance of step-size ($\mathrm{err}_{new} < 1$) is satisfied. Otherwise, we keep on proposing smaller values of h_{new} using (2.32) until $\mathrm{err}_{new} < 1$. The condition $\mathrm{err}_{new} < 1$ will be satisfied for sufficiently small h_{new}, because $E_{new} \to 0$ as $h_{new} \to 0$.

Let us also clarify the choice of proposal (2.31). Rearranging the terms in $h_{new} = \kappa(\mathrm{err}_n, \mathrm{err}_{n-1}, h_n, h_{n-1})$ gives us

$$\frac{h_{new}}{h_n} \left(\frac{\mathrm{err}_n}{1}\right)^{1/2} = fac \cdot \frac{h_n}{h_{n-1}} \left(\frac{\mathrm{err}_{n-1}}{\mathrm{err}_n}\right)^{1/2}. \tag{2.33}$$

Thus, under assumption that

$$\frac{h_n}{h_{n-1}}\left(\frac{\text{err}_{n-1}}{\text{err}_n}\right)^{1/2} \approx C, \tag{2.34}$$

where $C > 0$ is a constant, we will chose h_{new} such that $\text{err}_{new} \approx \text{fac}$. Thus, from this choice we expect that $\text{err}_{new} \approx fac$, in which case the step-size would be accepted immediately. The coefficient fac is set to be smaller than 1 in order to reduce the number of step-size rejections, thus reducing the computational cost. Let us remark that such a choice of the adaptive time-stepping algorithm will select the step-sizes such that $\text{err}_n \approx \text{fac}$, and thus the magnitude of the relative errors along the trajectory will be the same.

For clarification, we provide the pseudo-code in Algorithm 1. Let us remark that the adaptive time-stepping scheme is not well defined for $n = 0, 1$. In this case we can choose some very small value h_ε and calculate the first two approximations y_1, y_2 using the numerical integrator with step-size h_ε.

Algorithm 1 Adaptive time-stepping algorithm.

1: Calculate $y_1 = \Psi_{h_\varepsilon}(y_0)$, $y_2 = \Psi_{h_\varepsilon}(y_1)$, where $h_\varepsilon \ll 1$.
2: Take $t_2 = 2 \cdot h_\varepsilon$ and set $n = 1$.
3: If $t_n = T$, take $N = n$, and return y_N.
4: Propose $h_{new} = \kappa(err_n, err_{n-1}, h_n, h_{n-1})$.
5: If $t_n + h_{new} > T$, set $h_{new} = T - t_n$.
6: Calculate $y_{new} = \Psi_{h_{new}}(y_n)$.
7: Calculate local error E_{new}.
8: Calculate err_{new} as in (3.31).
9: If $\text{err}_{new} < 1$, set $t_{n+1} = t_n + h_{new}$, . Increment n and go to point 3.
10: Otherwise set $h_{new} := h_{new} \cdot (1/\text{err}_{new})^{1/2}$, and go to point 3.

In order to test the adaptive time-stepping algorithm we apply ROCK2 method to the stiff peroxide-oxide equation [21], given by

$$\begin{aligned}
a' &= k_7(a_0 - a) - k_3 aby, & a(0) &= 6,\\
b' &= k_8 b_0 - k_1 bx - k_3 aby, & b(0) &= 58,\\
x' &= k_1 bx - 2k_2 X^2 + 3k_3 aby - k_4 x + k_6 x_0, & x(0) &= 0,\\
y' &= 2k_2 x^2 - k_5 y - k_3 aby, & y(0) &= 0.
\end{aligned} \tag{2.35}$$

where the coefficients $\{k_i\}_{i=1}^8$ are given by

$$\begin{aligned}
k_1 &= 0.35, \quad & k_2 &= 250, \quad & k_3 &= 0.035, \quad & k_4 &= 20,\\
k_5 &= 5.35, \quad & k_6 &= 10^{-5}, \quad & k_7 &= 0.1, \quad & k_8 &= 0.825.
\end{aligned} \tag{2.36}$$

We are interested in the solution at final time $T = 200$. Results are shown in Figure 2.5, where we plot errors of fixed and adaptive time-stepping schemes against the number of function evaluations.

We use $N = 8 \cdot 10^5$ to get the reference value of $y(T)$, which we consider to be exact. Then, this value is used to estimate the error of both fixed and adaptive time-stepping schemes. Adaptive time-stepping scheme is used with tolerances $\tau_i = 10^{-7}2^{-i}, i = 1, \dots, 5$, while the fixed time-stepping scheme is used with step-sizes $h_i = T/(N \cdot 2^i)$, $N = 5 \cdot 10^4, i = 1, \dots, 4$. We can see that the adaptive time-stepping method is performing better than the fixed time-stepping method in this test case.

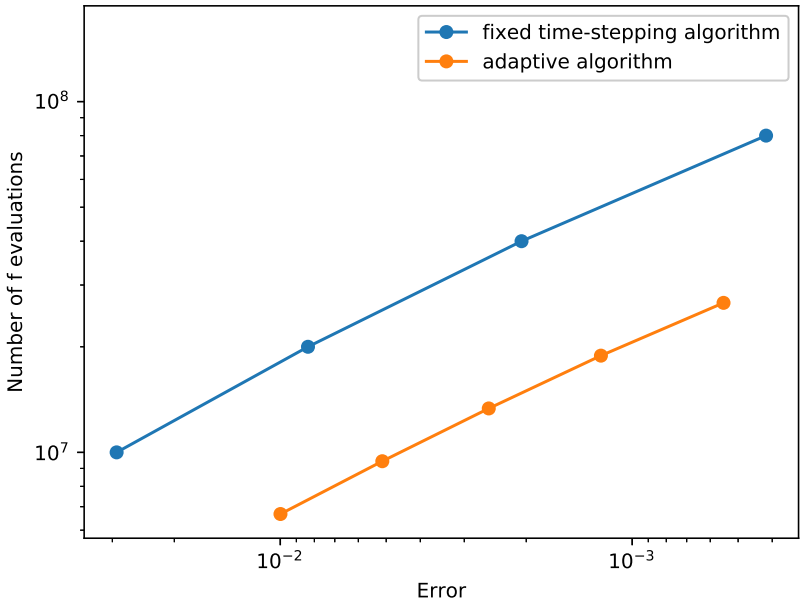


Fig. 2.5 Performance comparison of fixed time-stepping scheme and adaptive time-stepping scheme for the peroxide-oxide equation (2.35).

Chapter 3

Probabilistic Numerical Methods for Ordinary Differential Equations

In the previous chapter we introduced numerical methods which approximate the solution of an ODE with a pointwise estimate. While the approximation produced by these methods can be made arbitrarily precise by choosing sufficiently small step-size, these methods cannot measure the uncertainty of their solution except with error estimators. For chaotic problems, we observe that these numerical methods can give overconfident estimates of the error.

In recent years there has been an increased interest in numerical solvers which describe the solution of an equation in a probabilistic manner [3, 8, 19]. The methods presented in these papers use random variables to describe the exact solution, and thus give richer description about the uncertainty of the numerical solver than the error estimates coming from *classical* integrators (Runge-Kutta, multistep).

In this chapter we discuss probabilistic methods for integration of ordinary differential equations. In Section 3.1 we introduce general terms and notation which will be used for presenting and analyzing these methods.

In Section 3.2, we give a brief exposition of the method presented in [8], which at every step of a Runge-Kutta method adds a noise in a form of a suitably chosen random variable. Then, in Section 3.3, we present a recent approach [3] in which the randomization comes from the probabilistic perturbation of the step-sizes in a Runge-Kutta method.

Finally, in Section 3.4, we conclude the exposition of the probabilistic numerical methods for ODEs by providing numerical results which showcase the properties discussed in this chapter. We first show in Section 3.4.1 the results of numerical tests which confirm the convergence orders of methods presented in this chapter. Then, in Section 3.4.2 we study the application of probabilistic ODE solvers to chaotic problems. Finally, in Section 3.4.3 we extend the adaptive time-stepping algorithm introduced in Section 2.4, and show that it outperforms fixed time-stepping scheme.

Part of the research effort in this thesis was devoted to finding an efficient adaptive time-stepping scheme for the method introduced in Section 3.3 which uses information provided by random vari-

ables generated at every time-step. At the end of this chapter we show the approach that was tried during the project but which unfortunately shows worse performance than the fixed time-stepping scheme.

3.1 General setting

In this section we provide a general setting for probabilistic numerical methods which approximate the solution of equation $y(t)$ at times $\{t_n\}_{n=1}^N$ by random variables $\{Y_n\}_{n=1}^N$. The discussion given in this section will be used in Section 3.2 and Section 3.3 to describe two different classes of probabilistic ODE integrators.

The random variables $\{Y_n\}_{n=1}^N$ which represent the approximations produced by a probabilistic algorithm at times $t \in \{t_n\}_{n=1}^N$ form a Markov Chain, which means the distribution of the variable $Y_n, n = 1, \dots, N$ depends only on the value sampled from Y_{n-1}. The transition from the step Y_{n-1} to Y_n given Y_{n-1} can be represented with

$$Y_n = \Psi_h^s(Y_{n-1}, W_n), \quad n = 1, \dots, N, \tag{3.1}$$

where $\{W_n\}_{n=1}^N$ is a family of i.i.d. random variables defined over a probability space $(\Theta, \mathcal{F}, \nu)$. Usually, we have that $\Theta = \mathbb{R}^k, k \in \mathbb{N}$, and $\mathcal{F}$ is the Borel σ-algebra. When the choice of W_n is clear we sometimes omit the argument W_n in (3.1) and write $Y_n = \Psi_h^s(Y_{n-1})$ instead. Formally, the Markov property of the chain $\{Y_n\}_{n=1}^N$ can be stated as

$$\mathbb{E}\left(Y_k | Y_{k-1} = y_{k-1}, \dots, Y_0 = y_0\right) = \mathbb{E}\left(Y_k | Y_{k-1} = y_{k-1}\right), \quad k = 1, \dots, N. \tag{3.2}$$

Let us now consider the concept of strong convergence order which is used to study the path-wise precision of probabilstic numerical integrators.

Definition 3.1. *Consider the initial value problem (2.4) and a probabilistic numerical integrator which approximates the solution $y(t)$ at times $t \in \{t_n\}_{n=1}^N$, with $t_n = nh$ and $Nh = T$, by random variables $\{Y_n\}_{n=1}^n$. The numerical method has strong order p if and only if for all sufficiently small h we have*

$$\sup_{n=1,\dots,N} \mathbb{E}\left|Y_n - y(t_n)\right| \leq Ch^p, \tag{3.3}$$

where $C \in \mathbb{R}^+$ is a constant independent of h.

In many cases we are interested in measuring some quantity derived from the solution $y(t)$ of the ODE. For example, we could be interested in evaluating the energy of the dynamical system at final time. In particular, we are interested in estimating $Q(y(t))$ where Q is some sufficiently smooth function. Since the output of the probabilistic method at the final time T is a random variable Y_N, the quantity of interest is approximated by the random variable $Q(Y_N)$. The precision of the

probabilistic numerical integrator in this case can be studied through weak convergence, which is defined as follows.

Definition 3.2. *Consider the initial value problem (2.4) and a probabilistic numerical integrator which approximates the solution at times* $\{t_n = nh\}_{n=1}^N$ *by random variables* $\{Y_n\}_{n=1}^n$. *The numerical method has weak order p if and only if for every uniformly bounded function* $Q \in C^\infty(\mathbb{R}^d, \mathbb{R})$ *with all derivatives uniformly bounded we have*

$$\sup_{n=1,\dots,N} \left|\mathbb{E}Q(Y_n) - Q(y(t_n))\right| \leq Ch^p, \tag{3.4}$$

for all sufficiently small h and where $C \in \mathbb{R}^+$ *is a constant independent of h.*

In practice, the distribution of Y_N is not available and we resort to Monte-Carlo sampling to infer the information about it. Thus, we draw M different trajectories which give outputs $\{Y_N^{(i)}\}_{i=1}^M$, and estimate the value $Z = Q(y(T))$ by

$$\hat{Z} = \frac{1}{M}\sum_{i=1}^{M} Q\left(Y_N^{(i)}\right). \tag{3.5}$$

The approximation $\hat{Z}$ introduces error from two sources. One of them is the imprecision of probabilistic numerical integrator, which is captured by the weak convergence order, while the other one comes from sampling.

3.2 Additive Noise Runge-Kutta Methods

In this section we give a quick overview of the probabilistic numerical method introduced in [8]. This scheme extends Runge-Kutta methods by adding a noise term in form of a random variable at every time-step. This noise can be intuitively interpreted as the uncertainty of the numerical solver about the solution. We will refer to this family of methods as additive noise Runge-Kutta methods (AN-RK).

Consider a Runge-Kutta method defined by its numerical flow Ψ_h. For the AN-RK method we define the transition between Y_n and Y_{n+1} by

$$Y_{n+1} = Y_n + \Psi_h(Y_n) + \varepsilon_n(h_n), \tag{3.6}$$

where $\{\varepsilon_n\}_{n=1}^N$ are i.i.d. stochastic processes given by

$$\varepsilon_n(t) = \int_0^t \chi_n(\tau)d\tau, n = 1, \dots, N. \tag{3.7}$$

We denote with $\{\chi_n(\tau)\}_{n=1}^N$ i.i.d. random variables which are artificially introduced in order to represent the uncertainty of the solver at a specific time. Usually, we take $\{\chi_n(\tau)\}_{n=1}^N$ to be normally

distributed, i.e.

$$\chi_n(\tau) \sim \mathcal{N}(0, C^h), \tag{3.8}$$

where C^h is the covariance matrix which depends on the step-size h. For this choice of $\{\chi_n(\tau)\}_{n=1}^N$, we have that $\{\varepsilon_n(t)\}_{n=1}^N$ are zero mean normally distributed as well.

In most common cases only the value of $\varepsilon_n(h_n)$ is necessary for integration of ODE, in which case we can treat $\varepsilon_n(t)$ as a random variable. But for the purpose of analysis of this method (see [8]) we need to keep in mind that the approximation of solution for $t \in [t_n, t_{n+1}]$ is given with

$$Y(t) = Y_n + \Psi_{t-t_n}(Y_N) + \varepsilon_n(t - t_n). \tag{3.9}$$

It is also important to consider $\varepsilon_n(t)$ as a stochastic process when using adaptive time-stepping schemes, since in case the step-size proposal is rejected, the process $\varepsilon_n(t)$ needs to be sampled backwards in time.

Let us now consider the convergence properties of this method. Before stating the results, let us make some assumptions on the additive random noise $\{\chi_n(t)\}_{n=1}^N$.

Assumption 3.1. *Consider $\varepsilon_k = \int_0^t \chi_k(s)ds$ with $\chi_k \sim \mathcal{N}(0, C^h)$. Then, the following inequality holds*

$$\mathbb{E}^h|\varepsilon_k(t)\varepsilon_k(t)^\top|_F \leq Kt^{2q+1}, \tag{3.10}$$

where $|\cdot|$ denotes Frobenius norm in $\mathbb{R}^n$, $\mathbb{E}^h$ is expectation with respect to the i.i.d χ_k, $K \in \mathbb{R}^+$, and $p \in \mathbb{N}$. Furthermore, there exists a matrix $Q \in \mathbb{R}^{d\times d}$ which does not depend on h such that

$$\mathbb{E}^h[\varepsilon_k(h)\varepsilon_k(h)] = Qh^{2q+1}. \tag{3.11}$$

Furthermore, we need to make certain assumptions on the regularity of the dynamical system (2.3), and local convergence order of the deterministic numerical solver, as follows.

Assumption 3.2. *The function f from (2.3) and a sufficient number of its derivatives are uniformly bounded in $\mathbb{R}^d$. Thus, the function f is globally Lipschitz. Furthermore, the numerical flow map Ψ_h from (3.6) is of order p, such that its local truncation is uniform and of order $p+1$*

$$\sup_{y_0\in\mathbb{R}^d} |\Psi_h(y_0) - \varphi_h(y_0)| \leq Kh^{p+1}. \tag{3.12}$$

We can now state the theorem on the strong convergence order of the additive noise method.

Theorem 3.1. *Consider the additive noise method (3.6), and let Assumptions 3.1,3.2 hold. Then, we have*

$$\sup_{k=1,\dots,N} \mathbb{E}^h|y(t_k) - Y_k| \leq Kh^{\min(p,q)}, \tag{3.13}$$

where $K > 0$ is a constant which does not depend on h.

We can see that the strong convergence depends on the precision of deterministic solver, as well as on the variance of the additive noise. It is natural to choose $q \geq p$, in order to preserve the accuracy of the deterministic solver.

For the weak convergence results to hold we need to introduce additional regularity assumptions on the solution of the initial value problem (2.2), and the stochastic numerical flow Ψ_h^s, as follows.

Assumption 3.3. *Consider the initial value problem (2.2), and consider function $Q \in \mathcal{C}^\infty(\mathbb{R}^d, \mathbb{R})$. The function f belongs to $C^\infty(\mathbb{R}^d)$, and all of its derivatives are uniformly bounded in $\mathbb{R}^d$. Furthermore, the flow φ_h and the stochastic numerical flow Ψ_h^s of the AN-RK method satisfy*

$$\begin{aligned} \sup_{y_0 \in \mathbb{R}^d} \left| Q(\varphi_h(y_0)) \right| &\leq (1 + Lh) \sup_{y_0 \in \mathbb{R}^d} |Q(y_0)|, \\ \sup_{y_0 \in \mathbb{R}^d} \left| Q(\Psi_h^s(y_0)) \right| &\leq (1 + Lh) \sup_{y_0 \in \mathbb{R}^d} |Q(y_0)|. \end{aligned} \tag{3.14}$$

Let us now state the weak convergence result for the AN-RK method.

Theorem 3.2. *Consider the AN-RK method (3.6), and let Assumptions 3.1 and 3.3 hold. Then, for any function $Q \in C^\infty(\mathbb{R}^d, \mathbb{R})$ with all derivatives uniformly bounded we have that*

$$\left| Q(y(T)) - \mathbb{E}(Q(Y_N)) \right| \leq K h^{\min\{2q, p\}}, \tag{3.15}$$

where K is a constant which does not depend on h.

3.3 Randomized Time-Stepping Runge-Kutta Methods

In the previous section we extended Runge-Kutta methods to probabilistic numerical ODE solvers by employing additive noise to introduce uncertainty. In this section we consider a different scheme, in which uncertainty on the final solution is accounted for by a probabilstic perturbation of the step-sizes. We will refer to this scheme by RTS-RK.

In particular, we consider the stochastic numerical flow (3.1) defined as

$$Y_{n+1} = \Psi_{H_n}(Y_n), \quad n = 0, \dots, N-1, \tag{3.16}$$

where $\{H_n\}_{i=0}^{N-1}$ are i.i.d. random variables and Ψ_h is the numerical flow of a Runge-Kutta method. The choice of the random step-sizes $\{H_n\}_{n=0}^{N-1}$ is not arbitrary. In particular, we consider the following assumption.

Assumption 3.4. *The i.i.d. random variables $\{H_n\}_{n=0}^{N-1}$ satisfy*

- $H_n > 0$ *a.s.,*
- $\mathbb{E} H_n = h$, *for some $h \in \mathbb{R}^+$ independent of k,*

- *there exists* $q \in \mathbb{N}, q > 1$ *such that Var* $H_n = Ch^{2q}$.

There are numerous different choices of random variables $\{H_n\}_{n=0}^{N-1}$ which satisfy Assumption 3.4. One such choice is given by a random variable

$$H_n \sim \mathcal{U}(h - h^q, h + h^q), \quad 0 < h < 1, q \in \mathbb{N}, n = 0, \dots, N-1, \tag{3.17}$$

for which the Assumption 3.4 trivially holds. Assumption 3.4 is not sufficient to ensure strong convergence of the RTS-RK scheme, as we also need to assume smoothness of the deterministic Runge-Kutta integrator Ψ_h.

Assumption 3.5. *The numerical flow* $h \to \Psi_h(y)$ *of Runge-Kutta method belongs to* $C^2\left(\mathbb{R}^+, \mathbb{R}^d\right)$ *and is Liptchitz with constant* L_Ψ

$$|\Psi_t(y) - \Psi_s(y)| \leq L_\Psi |t - s|, \forall t, s \in \mathbb{R}^+. \tag{3.18}$$

Let us now state the strong convergence order result for the RTS-RK method [3, Theorem 3.5].

Theorem 3.3. *Consider the RTS-RK method (3.16), and let Assumptions 3.4,3.5 hold. Let us consider times* $t_n = nh, n = 0, \dots, N$, *with* $Nh = T$. *Then, the approximations* $\{Y_i\}_{i=1}^N$ *satisfy*

$$\sup_{i=1,\dots,N} \left(\mathbb{E}\|Y_i - y(t_i)\|\right) \leq Ch^{\min(p, q-1/2)}, \tag{3.19}$$

where p *is the convergence order of the underlying Runge-Kutta method, and* q *is as in Assumption 3.4.*

Proof. The proof is direct consequence of Theorem 4.3 from [3], and Cauchy-Schwarz inequality. □

In order to introduce weak convergence results, we need to consider additional stability assumption on the RTS-RK method.

Assumption 3.6. *Consider the initial value problem (2.4) and the RTS-RK method (3.16) with the stochastic numerical flow* Ψ_h^s. *For any function* $Q \in C^\infty(\mathbb{R}^d, \mathbb{R})$ *with all derivatives uniformly bounded there exists a constant* L *such that*

$$\sup_{u \in \mathbb{R}^d} \left|Q(\Psi_h^s(u))\right| \leq (1 + Lh) \sup_{u \in \mathbb{R}^d} |Q(u)|. \tag{3.20}$$

Let us now state the weak convergence result for RTS-RK method.

Theorem 3.4. *Consider the RTS-RK method (3.16), and let Assumptions 3.4,3.5 and 3.6 hold. Furthermore, assume that* $\mathbb{E}|H_0^4| < \infty$. *Let us consider times* $t_n = nh, n = 0, \dots, N$, *with* $Nh = T$. *Then, for any function* $Q \in C^\infty(\mathbb{R}^d, \mathbb{R})$ *with all derivatives uniformly bounded there exists a constant*

C independent of h such that

$$\sup_{i=1,\dots,N} \left|\mathbb{E}Q(Y_i) - Q(y(t_i))\right| \leq Ch^{\min(2p-1,q)}. \tag{3.21}$$

Proof. The proof is already given in Theorem 3.5 from [3]. □

From the convergence theorems for the RTS-RK method we can observe that it is enough to set $q \geq p + 1/2$ in order to preserve the accuracy of underlying Runge-Kutta scheme. Furthermore, it is reasonable to choose $q = p + 1/2$ since this choice of q gives the widest uncertainty quantification of numerical solver among all choices which preserve the precision of the underlying deterministic numerical integrator.

Let us also remark that the RTS-RK method inherits the geometric properties of the underlying Runge-Kutta method, which is not the case for the AN-RK method. This property proves to be particularly important for dynamical systems whose solutions conserve quadratic first integrals, and which naturally arise in Hamiltonian mechanics for example. In case the underlying Runge-Kutta method conserves these quantities, the RTS-RK method will preserve them too, and thus it will give better results than the AN-RK method. A wider discussion on this toipc can be found in [3]. For more detailed study of geometric numerical integrators, we refer to [14].

3.4 Numerical Experiments

In this section we provide numerical results for the methods presented in Sections 3.2, 3.3. We start by corroborating theoretical convergence results with numerical experiments in Section 3.4.1. Then, in Section 3.4.2, we study the approximation of the chaotic Lorenz system given by the RTS-RK method. In particular, we show that the RTS-RK method clearly indicates chaotic nature of the problem, and gives a bound on the uncertainty of numerical approximation.

Next, in Section 3.4.3, we show how the idea for adaptive time-stepping method from Section 2.4 can be extended to the RTS-RK method. Using numerical tests we show that this scheme shows advantageous performance in comparison to the fixed-timestepping scheme. One of the objectives in the master thesis project was to investigate whether an efficient adaptive time-stepping scheme for the RTS-RK method which uses only the variance of the time-steps Y_n can be designed. At the end of Section 3.4.3 we discuss methodology tried during the project, which does not show improvement over the fixed time-stepping scheme.

3.4.1 Convergence

In Theorems 3.1 and 3.3 we stated strong order of convergence results for AN-RK and RTS-RK methods, respectively. In this section we numerically show that these results hold, by applying these

methods to the FitzHugh-Nagumo equation [24]

$$\begin{aligned} \frac{dV}{dt} &= c\left(V - \frac{V^3}{3} + R\right), \\ \frac{dR}{dt} &= -\frac{1}{c}\left(V - a + bR\right), \end{aligned} \tag{3.22}$$

where coefficients a, b, c and the initial condition are given by

$$a = 0.2, b = 0.2, c = 3, V(0) = -1, R(0) = 1. \tag{3.23}$$

We look for a solution at the final time $T = 20$.

As an underlying Runge-Kutta method for both AN-RK and RTS-RK scheme we choose Heun's method (2.13). For the AN-RK method we select additive noise with $\varepsilon(\tau) = \mathcal{N}(0, \tau^{2p} I_2)$, where I_2 is two-dimensional identity matrix. For the RTS-RK method, we use H_k defined as in (3.17), with $q = p + 1/2 = 2.5$. We perform 20 numerical simulations for both methods using different

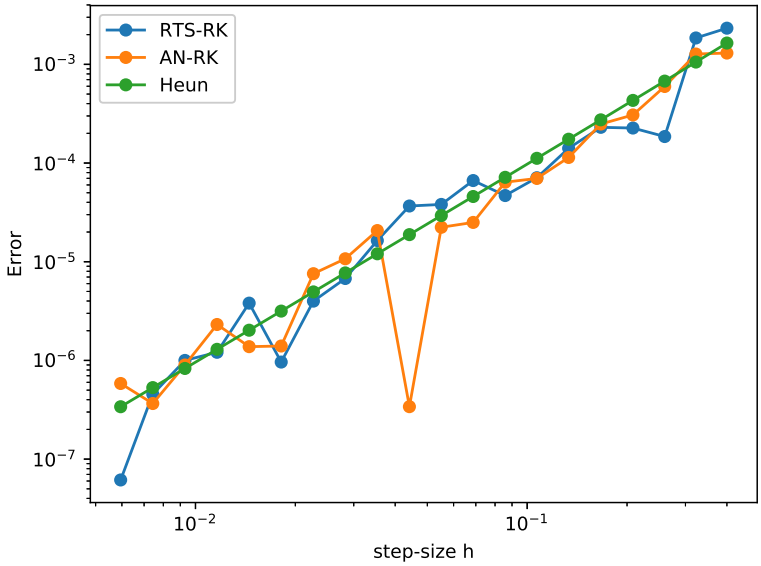


Fig. 3.1 Log-log plot of step-size h against the error for AN-RK, RTS-RK, and Heun's methods applied to FitzHugh-Nagumo equation (3.22).

step-sizes h_i, with the i-th simulation having $h_i = T/50 \cdot 0.8^i, i = 0, \ldots, 20$. For both methods we use 350 Monte-Carlo draws to obtain the estimate of the final solution. Furthermore, in order to compare the performance of probabilistic methods with the underlying Runge-Kutta integrator, we perform numerical simulations of the Heun's method with the same step-sizes $\{h_i\}_{i=0}^{20}$. The error of a simulation is calculated as an Euclidean distance between the simulation output and a reference solution which is calculated using Heun's method with a very small step-size.

It is also interesting to visualize the uncertainty of the solver about the solution. Thus, we plot the approximations of V from FitzHugh-Nagumo equation for 10 different trajectories of the RTS-RK method in Figure 3.2. The RTS-RK method is used with the values of H_k as in (3.17), for $h = T/N, N = 100$.

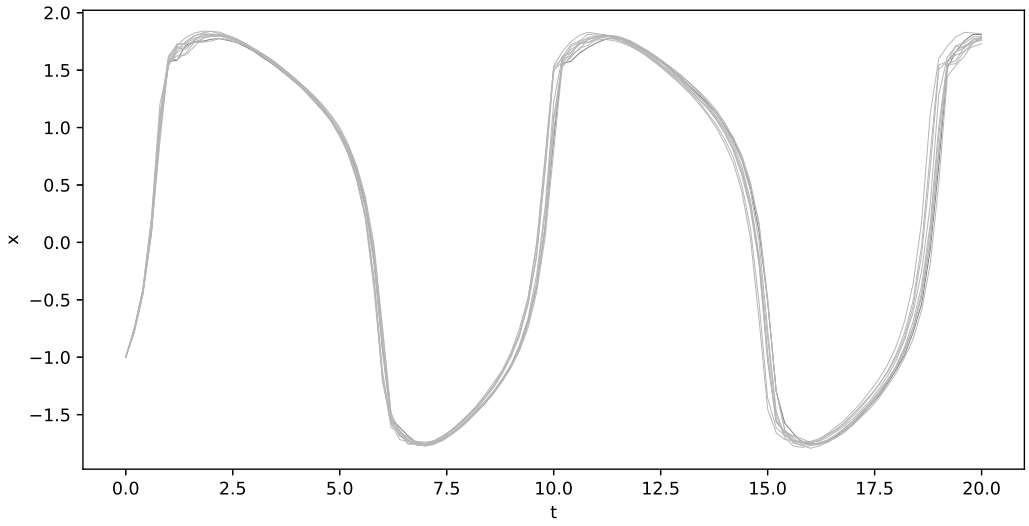


Fig. 3.2 Approximations of V from FitzHugh-Nagumo equation (3.22) for 10 different paths produced by the RTS-RK method with Heun's method as an underlying Runge-Kutta integrator.

3.4.2 Chaotic problems

Studying uncertainty of numerical solutions is especially interesting in the context of chaotic problems. In this subsection we consider application of the RTS-RK method from Section 3.3 to the Lorenz system

$$\begin{aligned} \dot{x} &= \sigma(y - x), & x(0) &= -10, \\ \dot{y} &= x(\rho - z) - y, & y(0) &= -1, \\ \dot{z} &= xy - \beta z, & z(0) &= 40, \end{aligned} \tag{3.24}$$

which shows chaotic behavior for parameters $\sigma = 10, \rho = 28, \beta = 8.0/3$. We search for a solution at final time $T = 40$. In Figure 3.3 we plot x-values of 10 random path realizations calculated using the RTS-RK scheme with Heun's method (2.13) as the underlying numerical integrator. For the average step-size h we take $h = T/N, N = 2000$. From this figure we can observe that the paths are "very close" to each other for $t < 5$, but then diverge from each other shortly after. Thus, we can intuitively assume that the solver is not very "confident" about the solution for values $t > 5$. Furthermore, we can observe that the uncertainty in $x(t)$ of the solver remains bounded for all values after $t > 5$, and thus we can intuitively conclude from the figure that $x(t)$ is between $[-20, 20]$ for $t \in [0, 40]$.

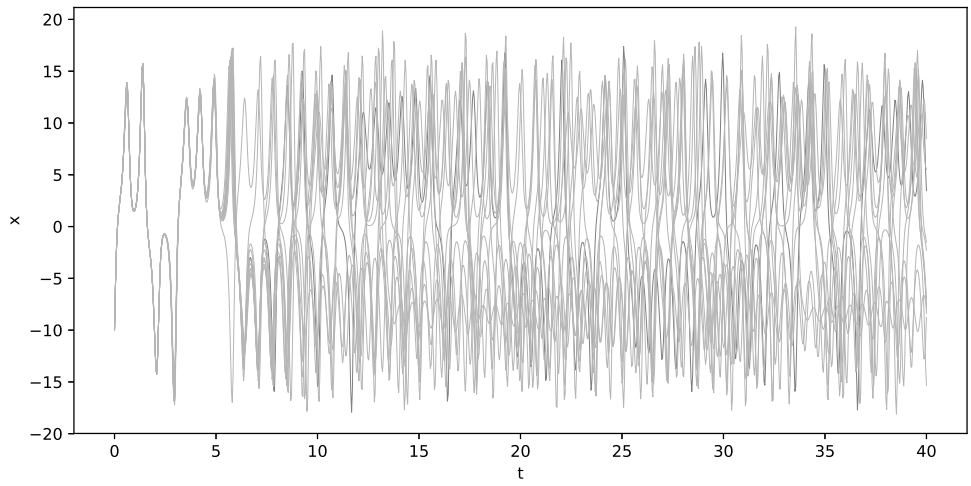


Fig. 3.3 Realizations of x-values for 10 different paths of Lorenz system.

3.4.3 Time-Step Adaptivity for RTS-RK Method

In this section we discuss how the RTS-RK method can be combined with an adaptive time-stepping scheme in order to achieve performance gains over the simplest fixed time-stepping scheme. We first develop an adaptive time-stepping scheme which estimates local errors using the embedding approach. We refer to this scheme as E-ADAPT, and perform a numerical test which shows that this strategy is advantageous over the fixed time-stepping scheme. The E-ADAPT strategy is only a slight extension of the strategy from Section 2.4, which will be referred to as A-ADAPT in this section.

It is also interesting to study whether the statistical properties of random variables $\{Y_n\}_{n=1}^M$ which describe the solution at time t_n carry some information about the numerical errors. In case this is true, it should be possible to construct an adaptive algorithm which exploits this information to scale the step-size h of the next time-step accordingly and outperform the fixed time-stepping scheme. Part of the master project was dedicated to this problem, and even though satisfactory results were not obtained, we conclude this subsection by discussing the approach taken.

In the whole section we consider the RTS-RK scheme with Heun's method (2.13) as an underlying Runge-Kutta method. The probabilistic step-sizes $\{H_n\}_{n=0}^{N-1}$ are given by (3.17), with $q = p + 1/2 = 2.5$. We use RTS-RK method to propagate M paths $\{Y_N^{(i)}\}_{i=1}^M$ together in time. Thus, at the n-th step, we have calculated values $\{Y_n^{(i)}\}_{i=1}^M$. Occasionally, we use also the explicit Euler method alongside the Heun's method in order to form an embedded Runge-Kutta method underlying the RTS-RK scheme, and get an embedded error estimate for each stochastic numerical flow evaluation.

Let us first consider an extension of the A-ADAPT time-stepping scheme from Section 2.4, which we refer to as E-ADAPT. The adaptive time-stepping algorithm iteratively constructs step-sizes $\{h_0, h_1, \dots\}$ along with path realizations $\{\{Y_1^{(i)}\}_{i=1}^M, \{Y_2^{(i)}\}_{i=1}^M, \dots\}$ in an iterative fashion. The

values $\{\{Y_n^{(i)}\}_{i=1}^M$ for $n \geq 1$ are calculated with

$$Y_n^{(i)} = \Psi_{H_n^{(i)}}(Y_{n-1}^{(i)}), \quad i = 1, \dots, M, \tag{3.25}$$

where $H_n^{(i)}$ are values sampled from probabilistic step-size H_n which satisfies $\mathbb{E}(H_n) = h_n$ as per Assumption 3.4.

We consider using an embedded method as an underlying Runge-Kutta method of the RTS-RK scheme, which allows us to define an embedded error of paths as

$$E_n^{(i)} = |\Psi_{H_n^{(i)}}(Y_{n-1}^{(i)}) - \Psi^*_{H_n^{(i)}}(Y_{n-1}^{(i)})|, \quad i = 1, \dots, M, \tag{3.26}$$

where $\Psi_{H_n^{(i)}}, \Psi^*_{H_n^{(i)}}$ are the two Runge-Kutta methods which form the embedded method. At every step, the new step-size h_{new} is proposed using the same function κ as given in (2.31), where values err_n are extended with

$$\mathrm{err}_n = \max_{i=1}^{M} \frac{E_n^{(i)}}{\sigma\left(\tau, h_{i-1}, Y_{n-1}^{(i)}, Y_n^{(i)}\right)}. \tag{3.27}$$

The other parts of the adaptive time-stepping algorithm are the same as in the algorithm presented in Section 2.4.

Let us now test the performance of the E-ADAPT scheme. We apply the scheme to the FitzHugh-Nagumo equation (3.22), with tolerances $\tau \in 0.001 \cdot 2^{-i}, i = 1, \dots, 10$. In Figure 3.4 we plot the errors of the E-ADAPT scheme with $M = 100$ path realizations against the number of evaluations of function f. In this figure we also plot the errors of the fixed time-stepping scheme and the A-ADAPT scheme. For the fixed time-stepping scheme we choose step-sizes $h_i = T/(N \cdot 2^i), i = 1, \dots, 6, N = 25$. The results for the A-ADAPT scheme are obtained using the same set of tolerances as for the E-ADAPT scheme. In order to make the comparison close, we scale down the number of functional evaluations of E-ADAPT scheme by M. This will in particular show the cost of the E-ADAPT scheme as if only one path is evaluated, while in practice we used $M = 100$ paths.

Let us now study the statistical properties of the variables Y_n, and discuss unsuccessful adaptive time-stepping strategy which relies solely on the statistical properties of Y_n. We will refer to the scheme that will be presented in the next paragraphs as to VAR-ADAPT.

In the proof of Theorem 5.1. from [3] it is stated that the variance of the estimator $\hat{Z}$ is bounded by

$$\mathrm{Var}(\hat{Z}) \leq \frac{1}{M} C h^{2\min(p, q-1/2)}, \tag{3.28}$$

where C is a constant independent of h. Since we consider $p = q + 1/2 = 2$, we have that $\mathrm{Var}(\hat{Z}) \leq Ch^2$. Thus, we have that $\mathrm{tr}\left(\mathrm{Var}\, Y_n\right) \leq Ch^2$. The approximation error of Heun's method is also bounded by Ch^2, and therefore it is interesting to see whether $\mathrm{tr}\left(\mathrm{Var}\, Y_n\right)$ contains some information about the error of a numerical method.

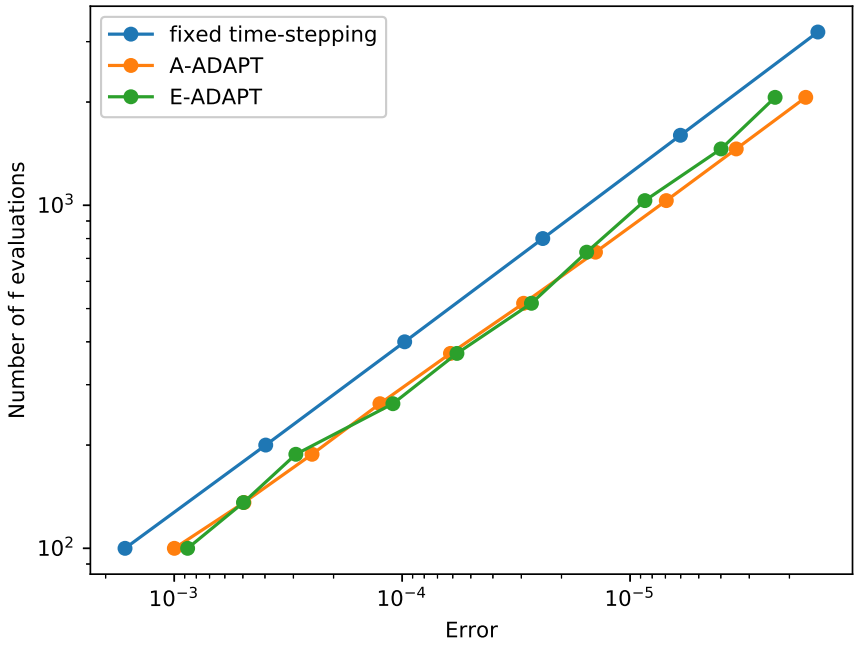


Fig. 3.4 Performance comparison of E-ADAPT, A-ADAPT, and fixed time-stepping schemes. The schemes are applied to the FitzHugh-Nagumo problem (3.22).

In order to check this we apply the RTS-RK method to the Lorenz system (3.24) with $M = 100$ paths. The estimations of $\sqrt{\mathrm{tr}\left(\mathrm{Var}\, Y_n\right)}, n = 1 \ldots, N$, produced by the RTS-RK method with a fixed step-size $h = T/N$, $N = 2000$, are showed in Figure 3.5. We also plot in this figure the embedded error estimates calculated by the underlying deterministic embedded method with the same step-size h. Furthermore, we also plot the *true* error, which is calculated as a difference between the expectation of the random path realizations and a reference solution calculated using numerical method with a very fine fixed step-size.

From Figure 3.5 we observe that $\sqrt{\mathrm{tr}\left(\mathrm{Var}\, Y_n\right)}$ indeed has the same order of magnitude as the true error, and therefore it *could* be a good indicator for the global error of the numerical method.

Let us now consider how we can use this information to construct the VAR-ADAPT adaptive time-stepping scheme. We use the same approach as for the E-ADAPT scheme, and only modify the calculation of errors E_n, err_n, as well as the new step-size proposition function κ.

Since we consider $\sqrt{\mathrm{tr}\left(\mathrm{Var}\, Y_n\right)}$ to be the global error of a numerical solver, it is intuitive to define the local error of an integration step at time t_{n-1} as

$$\bar{E}_n = \sqrt{\mathrm{tr}\left(\mathrm{Var}\, Y_n\right)} - \sqrt{\mathrm{tr}\left(\mathrm{Var}\, Y_{n-1}\right)}. \tag{3.29}$$

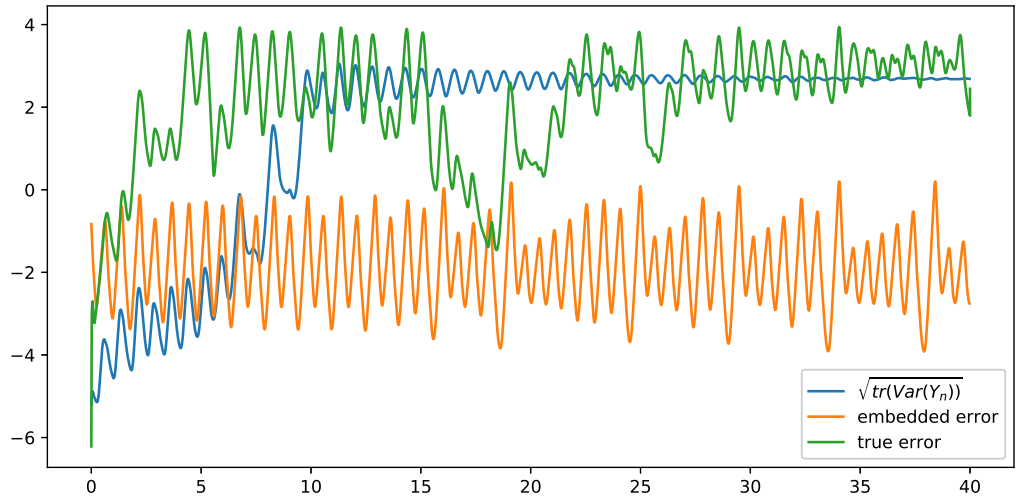


Fig. 3.5 Semi-log plot of error, $\sqrt{\text{tr}\left(\text{Var}\, Y_n\right)}$, and embedded error of Heun's method with fixed time-stepping scheme applied to the Lorenz system.

Since the value of the local error $\bar{E}_n$ defined in this way can also be negative, for the time-stepping algorithm we consider a slightly different version

$$\hat{E}_n = \max\left(\left|\sqrt{\text{tr}\left(\text{Var}\, Y_n\right)} - \sqrt{\text{tr}\left(\text{Var}\, Y_{n-1}\right)}\right|, \epsilon_{min}\right), \tag{3.30}$$

where ϵ_{min} is some safety constant which bounds the local error from below. In all numerical experiments this constant is taken to be $\epsilon_{min} = 10^{-10}$. In practice, the values $\text{tr}\left(\text{Var}\, Y_n\right)$ are estimated from the path realizations $\{Y_n^{(i)}\}_{i=1}^M$.

It is interesting to observe how the values of $\hat{E}_n$ compare to the true error and embedding errors. Thus, we apply the RTS-RK method to the FitzHugh-Nagumo equation (3.22), with a fixed time-step $h = T/N, N = 2000$, and plot the estimations of $\hat{E}_n$ in Figure 3.6. We also plot in this figure the embedded error estimates calculated by underlying deterministic embedded method with the same step-size h. Furthermore, we also plot the *true* error, which is calculated as a difference between the expectation of the random path realizations and a reference solution calculated using numerical method with very fine fixed step-size.

Now that we have defined the local integration error $\hat{E}_n$, let us define the relative error err_n with

$$\text{err}_n = \frac{\hat{E}_n}{\left(\max_{i=1}^M \sigma(\tau, h_n, Y_{n-1}^{(i)}, Y_n^{(i)})\right)}. \tag{3.31}$$

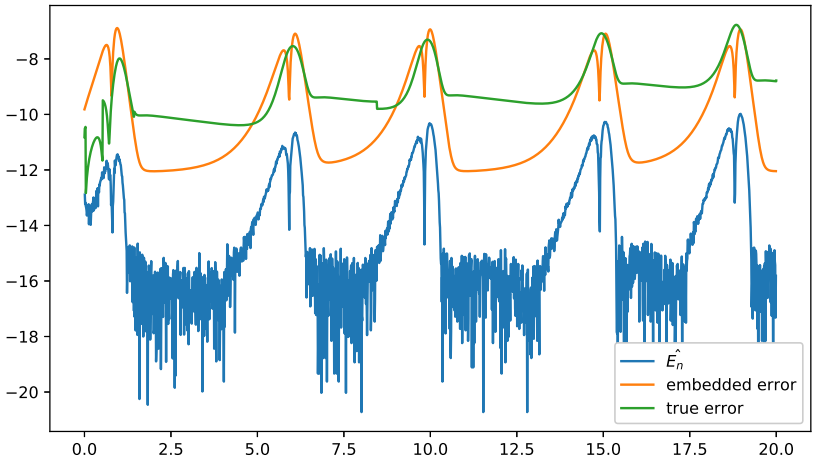


Fig. 3.6 Semi-log plot of error, $\hat{E}_n$, and embedded of the RTS-RK method error applied to the FitzHugh-Nagumo equation (3.22).

Finally, the proposal function κ is given with

$$\kappa(h_n, h_{n-1}, err_n, err_{n-1}) = fac \cdot h_n \left(\frac{1}{\text{err}_n}\right)^{1/4} \frac{h_n}{h_{n-1}} \left(\frac{\text{err}_{n-1}}{\text{err}_n}\right)^{1/4}. \tag{3.32}$$

Note that in (3.32) we use the fourth root of err_n instead of the square root as in (2.31). This is because the err_n is of order 1 in (2.31) (since embedded error is of order 1), while the order of err_n in (3.32) is 2, and therefore appropriate scaling is needed.

Let us now test the performance of the VAR-ADAPT scheme. We apply the scheme to the FitzHugh-Nagumo equation (3.22), with tolerances $\tau_i = 0.012^{-i}, i = 1, \dots, 20$. In Figure 3.7 we plot the errors of the VAR-ADAPT scheme with $M = 1000$ path realizations against the number of function evaluations. In this figure we also plot the errors of fixed time-stepping scheme and the A-ADAPT scheme. For the the fixed time-stepping scheme we choose step-sizes $h_i = T/(N \cdot 1.2^i), i = 1, \dots, 20$. The results for the A-ADAPT scheme are obtained using the same set of tolerances as for the E-ADAPT scheme. In order to make the comparison, we scale down the number of functional evaluations of the VAR-ADAPT scheme by M. This will in particular show the cost of the VAR-ADAPT scheme as if only one path is evaluated, while in practice we used $M = 1000$ paths.

From Figure 3.7 we can see that the VAR-ADAPT scheme has same the convergence order as the underlying Runge-Kutta method, but it is outperformed even by fixed time-stepping scheme.

It could be interesting to check the selection of step-sizes for VAR-ADAPT, A-ADAPT, and fixed time-stepping schemes used for obtaining the results in Figure 3.7. Therefore, in Figure 3.8 we plot the step-sizes h_n at the times t_n for each of the methods. The fixed time-stepping method is used with $h = T/N$, $N = 2000$, the A-ADAPT scheme is used with tolerance $\tau = 10^{-2}$, and the VAR-ADAPT

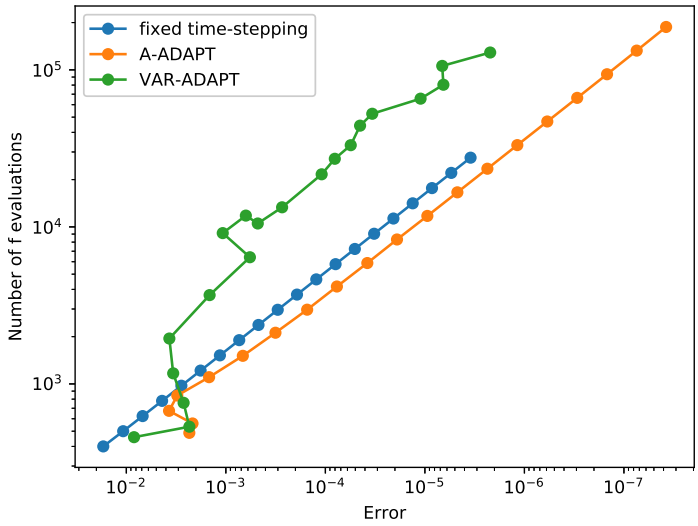


Fig. 3.7 Performance plot of fixed time-stepping, A-ADAPT and VAR-ADAPT schemes for FitzHugh-Nagumo model.

scheme is used with $M = 1000$ paths and tolerance $\tau = 10^{-3}$. From Figure 3.8 we can observe that the VAR-ADAPT scheme chooses similar step-sizes as the A-ADAPT scheme, but at certain areas it chooses too small step-sizes which spoils its performance.

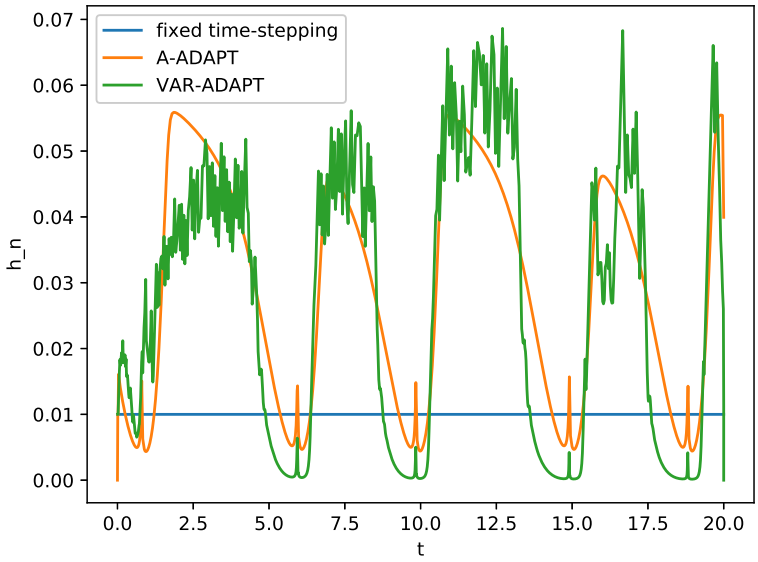


Fig. 3.8 Plot of the step-sizes against the times at which the are realized for VAR-ADAPT, A-ADAPT, and fixed time-stepping scheme.

Chapter 4

Bayesian Inverse Problems in Context of ODEs

One of the central questions of numerical mathematics is the development of efficient methods which approximate solutions of certain models. In abstract terms, in this thesis we represent a model with a mapping $\mathcal{G}(u) \rightarrow y$, where u represents parameters, and y the solution. Usually, the operator $\mathcal{G}$ is very complex, and numerical mathematics aims to find an approximation $\mathcal{G}_h$ of $\mathcal{G}$ which can be efficiently computed and for which the value $\mathcal{G}_h(u)$ is very close to solution $\mathcal{G}(u)$. The symbol h of the operator $\mathcal{G}_h$ represents some discretization parameter of the numerical scheme, such as for example the step-size of Runge-Kutta methods.

In this chapter we consider y to be given, for example by observing the results of an experiment, and aim to find the values of u which through $\mathcal{G}$ approximate the results as good as possible . Finding "suitable" values of u which the operator $\mathcal{G}$ maps near the observed y is known as the inverse problem. The definition of "suitable" values u is not clear, especially since the operator $\mathcal{G}$ is in general not invertible.

In this thesis we opt for the probabilistic approach which relies on Bayes' theorem and states the solution of an inverse problem in a clear and logical manner. Furthermore, the probabilistic approach can mitigate ill-posedness which is often found in inverse problems. In this chapter we give a rough overview of the methodology used in Bayesian inverse problems. For more detailed study we refer to [12, 18, 26].

We start this chapter by introducing inverse problems in Section 4.1. In this section we show how Bayes' theorem can be used to describe the solution of an inverse problem as a random variable, which we denote with $\pi^{\mathcal{Y}}$.

In Section 4.2, we consider the inverse problems for which the operator $\mathcal{G}$ is given by a solution of an ODE, while the u is a parameter of the equation. Since the solution of an ODE in general cannot be given as a closed-form expression, the operator $\mathcal{G}$ is in practice replaced with an operator $\mathcal{G}_h$ which approximates $\mathcal{G}$ using numerical solvers for ODEs, such as the ones discussed in Chapter 2. Since we use $\mathcal{G}_h$ instead of $\mathcal{G}$, the solution of the inverse problem is a random variable $\pi_h^{\mathcal{Y}}$ different from

$\pi^{\mathcal{Y}}$. Thus, it is interesting to see if the true solution $\pi^{\mathcal{Y}}$ is close to $\pi_h^{\mathcal{Y}}$ in case the operator $\mathcal{G}$ is well approximated by $\mathcal{G}_h$, which is addressed at the end of Section 4.2.

The solution $\pi^{\mathcal{Y}}$ of Bayesian inverse problems introduced in Section 4.1 has very complex probability density function and is often high dimensional in practice, which makes sampling from it non-trivial. Therefore, in Section 4.3 we introduce Markov Chain Monte Carlo methods, which are used for sampling solutions of Bayesian inverse problems.

Next, as an alternative to MCMC methods, we consider sequential Monte Carlo (SMC) methods in Section 4.4. While these methods tend to be expensive in comparison to MCMC methods, we include them in this master thesis for the completeness of the exposition, and also because they present an active area of research.

In Section 4.5 we consider replacing the $\mathcal{G}$ by a probabilistic operator $\mathcal{G}_h^s$ which for every input u outputs random variable $\mathcal{G}_h^s(u)$. This probabilistic operator naturally arises for example in Bayesian inverse problems for ODEs when the equation is solved with probabilistic Runge-Kutta methods as the ones introduced in Chapter 3. The operator $\mathcal{G}_h^s$ is then used by two different sampling schemes: Monte Carlo within Metropolis (MCWM) and pseudo-marginal Metropolis-Hastings (PMMH) algorithm. In Section 4.5.1 we give a bound on a difference between the solution of an inverse problem and approximation sampled by MCWM method. A similar result for PMMH algorithm is omitted since it was already given in [22]. As we will see from the numerical examples, probabilistic methods tend to account for the uncertainty better that the deterministic methods, and thus they usually give more precise posterior distributions.

Finally, in Section 4.6 we give results of various numerical experiments in order to illustrate the concepts discussed in this chapter. First, we test MCMC and SMC methods on a very simple ODE in order to verify the correctness of both methods. Next, to showcase the power of probabilistic approach, we apply the same methods on the FitzHugh-Nagumo problem, for which satisfactory results are obtained. Furthermore, it is interesting to see how probabilistic ODE solvers impact the solution of inverse problems. Thus, at the end of Section 4.6 we show that usage of probabilistic solvers in inverse problems indeed helps in capturing uncertainties of the numerical solver, which makes them more suitable than the deterministic Runge-Kutta solvers. Unfortunately, "correcting" behavior of probabilistic solvers is not guaranteed, which is highlighted in the last experiment of this section.

4.1 Introduction

Consider the equation

$$y = \mathcal{G}(u), \tag{4.1}$$

where u and y are vectors in Hilbert spaces X and Y respectively. We refer to $\mathcal{G} : X \to Y$ as the forward operator[1]. The output y is occasionally termed as *data* or observations. We denote by $\|.\|_X$

[1]The operator $\mathcal{G}$ is sometimes reffered to as an observational operator as well, see e.g. [26].

and $\|.\|_Y$ the norms defined over the spaces X and Y, respectively. The task of finding y given the input u is called the *forward problem*. In this chapter we are mostly interested in finding the value of u, given the observed values y, which is known as the *inverse problem*. Since we consider the value y to be fixed, and in order to avoid the confusion with the solution of an ODE $y(t)$, we will now use the formulation of inverse problem

$$\mathcal{Y} = \mathcal{G}(u), \tag{4.2}$$

instead.

In the general case, the inverse problem given as in (4.2) does not need to be well-posed. For example, the observations $\mathcal{Y}$ may not be in the image of the forward operator $\mathcal{G}$, in which case there is no solution to the inverse problem. Furthermore, it is also possible that the solutions of inverse problem are very sensitive to small changes in observations $\mathcal{Y}$. For all these reasons inverse problems require careful treatment.

One straightforward approach for solving (4.2) is to consider finding a solution to

$$\arg\min_{u \in X} \|y - \mathcal{G}(u)\|_Y, \tag{4.3}$$

which is a problem that can be solved using various global optimization techniques. Unfortunately, the formulation of the inverse problem as given in (4.3) can still be ill-posed. A common way to solve this issue is to introduce regularization to (4.3) with

$$\arg\min_{u \in E} \left(\|y - \mathcal{G}(u)\|_Y + \|u - m_0\|_E \right), \tag{4.4}$$

where $E \subset X, m_0 \in E$, are carefully chosen in order to alleviate ill-posedness. While this approach can give good results with well chosen E and m_0, the choice of proper E and m_0 can be difficult and ambiguous. Furthermore, in many cases we are interested not only in extracting the input u, but we also want to infer some information about the quality of the inverse problem's solution. For example, we might be interested in the sensitivity of the solution to the changes in observations $\mathcal{Y}$, or we might want to know whether some other choices of the input u are possible as well.

Due to these reasons, in this thesis we adopt a probabilistic approach, which provides a much richer description of the solution. In order to do so, we first consider observations to be given with some noise η, so that the problem (4.2) becomes

$$\mathcal{Y} = \mathcal{G}(u) + \eta. \tag{4.5}$$

We take η to be a zero mean random variable which can be intuitively understood as a term that captures measurement and modeling errors. The random variable η is commonly taken to be Gaussian, with covariance $\mathcal{C}$. From now on we will restrict ourselves to the case when X and Y are finite-dimensional Euclidean spaces, $X = \mathbb{R}^{d_1}, Y = \mathbb{R}^{d_2}$. Usual choice for covariance in this case is $\mathcal{C} = \sigma^2 I$, and thus $\eta \sim \mathcal{N}(0, \sigma^2 I)$. Let us denote with ρ the probability density function of η.

We model the *belief* about the input by considering it to be a random variable with a probability density function $\pi_0 : X \to \mathbb{R}^+$. Let us denote with π the probability density function of $\mathcal{G}(u) + \eta$. Given u, the likelihood of observing $\mathcal{Y}$ is given by

$$\pi(\mathcal{Y}|u) = \rho(\mathcal{Y} - \mathcal{G}(u)). \tag{4.6}$$

Informal application of Bayes' rule gives

$$\pi_0(u|\mathcal{Y}) = \frac{\rho(\mathcal{Y} - \mathcal{G}(u))\,\pi_0(u)}{\int_X \rho(\mathcal{Y} - \mathcal{G}(v))\,\pi_0(v)dv}. \tag{4.7}$$

Thus, we use observations $\mathcal{Y}$ to correct our belief about the input π_0 and obtain a posterior $\pi^{\mathcal{Y}}(\cdot) = \pi_0(\cdot|\mathcal{Y})$ using (4.7). In particular, up to a normalization constant, the p.d.f. of the posterior $\pi^{\mathcal{Y}}$ is given by

$$\pi^{\mathcal{Y}}(u) \propto \rho(\mathcal{Y} - \mathcal{G}(u))\,\pi_0(u). \tag{4.8}$$

Let us denote with $\mu_0, \mu^{\mathcal{Y}}$ the probability measures with densities $\pi_0, \pi^{\mathcal{Y}}$. From (4.8) we can link μ_0 and $\mu^{\mathcal{Y}}$ by Radon-Nikodym derivative as

$$\frac{d\mu^{\mathcal{Y}}}{d\mu_0} \propto \rho(\mathcal{Y} - \mathcal{G}(u)). \tag{4.9}$$

Introduction of the potential $\Phi(u, y) = -\log(\rho(y - \mathcal{G}(u)))$ allows us to rewrite (4.9) as

$$\frac{d\mu^{\mathcal{Y}}}{d\mu_0} \propto \exp(-\Phi(u, \mathcal{Y})). \tag{4.10}$$

The importance of (4.10) stems from the fact that it gives a solution of an inverse problem in terms of probability measures, and thus it generalizes to the case when X and Y are infinite-dimensional.

In order to ensure well-posedness of inverse problems[2], it is common to make the following assumptions on the potential Φ.

Assumption 4.1. *For the function $\Phi : X \times Y \to \mathbb{R}$ following holds.*

1. *For every $\varepsilon > 0$ and $r > 0$ there is $M \in \mathbb{R}$ such that for all $u \in X$ and all $y \in Y, \|y\|_Y < r$ we have*
$$\Phi(u, y) \geq M - \varepsilon\|u\|_X^2. \tag{4.11}$$

2. *For every $r > 0$ there is a constant $K > 0$ such that for all $u \in X, \|u\|_X < r$ and all $y \in Y, \|y\|_Y < r$ we have*
$$\Phi(u, y) \leq K. \tag{4.12}$$

[2]See for example Assumption 2.6 and Theorems 4.2, 4.6 from [26].

3. *For every $r > 0$ there is a constant $L > 0$ such that for all $u_1, u_2 \in X, \|u_1\|_X < r, \|u_2\|_X < r$, and every $y \in Y, \|y\|_Y < r$ we have*

$$|\Phi(u_1, y) - \Phi(u_2, y)| \leq L\|u_1 - u_2\|_X. \tag{4.13}$$

4. *For every $\varepsilon > 0$ and $r > 0$ there is a constant $C \in \mathbb{R}$ such that for every $u \in X$ and every $y_1, y_2 \in Y, \|y_1\|_Y < r, \|y_2\|_Y < r$ we have*

$$|\Phi(u, y_1) - \Phi(u, y_2)| \leq Ce^{\varepsilon\|u\|_X^2 + C}\|y_1 - y_2\|_Y. \tag{4.14}$$

These assumptions on operator Φ are not very strong, and they hold for wide variety of inverse problems. In particular, they give an upper and a lower bound on operator Φ, as well as Lipchitz-like properties on the first and second argument of Φ.

4.2 Inverse Problems for ODEs

In order to define inverse problems for ordinary differential equations, we enrich the definition (2.4) from Section 2.1 by considering a parameterized initial value problem given by

$$\dot{y}(t) = f_u(y), \qquad y(0) = y_0, \tag{4.15}$$

where $u \in X$ is a parameter of the function f. Let us also define the observation times to be $0 < \tau_1 < \tau_2 < \ldots < \tau_K = T, K \in \mathbb{N}$. We assume that the parametrized initial value problem (4.15) has a unique solution y_u for every $u \in X$, and define the forward operator $\mathcal{G}(u)$ by

$$\mathcal{G}(u) = (y_u(\tau_1), \ldots, y_u(\tau_K)). \tag{4.16}$$

Since the solution of the initial value problem (4.15) in general does not have a closed-form expression, we use a numerical method with a time-step h to approximate the solution $y_u(t)$ at times $t \in \{\tau_i\}_{i=1}^K$. Let us denote the approximations of solution at times $\{\tau_i\}_{i=1}^K$ given by the numerical method by $\{y_u^i\}_{i=1}^K$. Thus, instead of forward operator $\mathcal{G}$ we use in practice an operator $\mathcal{G}_h$ defined by

$$\mathcal{G}_h(u) = (y_u^1, y_u^2, \ldots, y_u^K). \tag{4.17}$$

Since we are replacing the operator $\mathcal{G}$ with an approximation $\mathcal{G}_h$, the inverse problem (4.5) becomes

$$\mathcal{Y} = \mathcal{G}_h(u) + \eta, \tag{4.18}$$

and has a different solution $\pi_h^{\mathcal{Y}}$. Anyway, in case $\mathcal{G}$ is well approximated by $\mathcal{G}_h$, in the sense that

$$|\mathcal{G}(u) - \mathcal{G}_h(u)| \leq Ch^p, \forall u \in X, \tag{4.19}$$

for some $p \in \mathbb{N}$ and $C > 0$ independent of h, the solution $\pi_h^{\mathcal{Y}}$ is a good approximation of $\pi^{\mathcal{Y}}$, which is stated in Theorem 4.6 from [26]. In particular, in case η from (4.5) is Gaussian, we have that

$$d_{Hell}(\mu, \mu_h) < Ch^{2p}, \tag{4.20}$$

where $C > 0$ is a constant independent of h and d_{Hell} is a Hellinger distance defined between two probability measures as

$$d^2_{Hell}(\mu_1, \mu_2) = \frac{1}{2} \int \left(\sqrt{\frac{d\mu_1}{d\mu}} - \sqrt{\frac{d\mu_2}{d\mu}} \right)^2 d\mu, \tag{4.21}$$

where μ is a probability measure chosen such that both μ_1 and μ_2 are absolutely continuous with respect to μ. Thus, as $h \to 0$, the approximation given by $\pi_h^{\mathcal{Y}}$ converges towards $\pi^{\mathcal{Y}}$.

4.3 Markov Chain Monte Carlo Methods

In this section we show how the posterior posterior random variable $\pi^{\mathcal{Y}}$ defined in (4.7) can be sampled efficiently. Since the expression for probability density function (4.7) is complex and often high-dimensional, it is common to use Markov Chain Monte Carlo (MCMC) methods to sample from it. Furthermore, MCMC methods are a good choice since they are able to sample from a random variables for which the p.d.f. is known up to normalizing constant. This is useful since the normalization constant of the posterior (4.7) is given as an integral

$$\int_X \rho\left(\mathcal{Y} - \mathcal{G}(v)\right) \pi_0(v) dv, \tag{4.22}$$

which generally does not have a closed-form expression, and numerical approximation of it is very expensive.

Let us now introduce MCMC methods. Consider a probability measure μ defined over some vector space X. We use a Markov Chain Monte Carlo method to construct an M-length array of samples $\{x_0, x_1, \ldots, x_M\}$ such that for any μ-measurable function ℓ we have

$$\int_X \ell(x)\mu(dx) \approx \frac{1}{M} \sum_{i=0}^{M} \ell(x_i). \tag{4.23}$$

The approximation can be made arbitrarily close by choosing sufficiently large M. The values $\{x_i\}_{i=0}^M$ are sampled from the first M random variables of Markov chain $\{X_i\}_{i=0}^\infty$. In particular, the chain $\{X_i\}_{i=0}^\infty$ consists of random variables such that for every $i > 0$, the distribution of X_i depends only on the sample from X_{i-1}. Thus, for defining MCMC methods it is enough to prescribe a transition probability kernel, i.e. the distribution of X_i given the sample from X_{i-1}. In case the transition kernel

is known, MCMC method is usually provided with the initial value x_0, while the other values x_i are sampled iteratively.

Let us consider the case when probability measure μ is defined over space $X = \mathbb{R}^k, k \in \mathbb{N}$. Consider Borel σ-algebra $\mathcal{B}(\mathbb{R}^k)$, and let us define the probability transition kernel as a mapping $P : \mathbb{R}^k \times \mathcal{B} \to [0, 1]$ for which the following holds.

- For each Borel set $B \in \mathcal{B}$, the mapping $P(\cdot, B)$ is a measurable function.
- For each $x \in \mathbb{R}^k$, the mapping $P(x, \cdot)$ is a probability distribution.

The MCMC method is usually given the first element of the chain X_0, and then we use the transition kernel to build define the rest of the Markov chain $\{X_j\}_{j=1}^{\infty}$. In particular, if we denote with $\{\mu_{X_j}\}_{j=0}^{\infty}$ the probability measures of $\{X_j\}_{j=0}^{\infty}$, the random variables $\{X_j\}_{j=1}^{\infty}$ are iteratively defined with

$$\mu_{X_{j+1}}(B) = \mu_{X_j} P(B) = \int_X P(t, B)\mu_{X_j}(dt), \qquad j > 0, \forall B \in \mathcal{B}. \tag{4.24}$$

Let us now introduce some important properties of the transition kernel P. The transition kernel P preserves a probability measure μ if and only if

$$\mu P = \mu. \tag{4.25}$$

The expression (4.25) should be understood as

$$\int_X P(x, B)\mu(dx) = \mu(B), \qquad \forall x \in X, \forall B \in \mathcal{B}. \tag{4.26}$$

In case (4.25) is satisfied, we also say that μ is an invariant measure of P.

The powers of transition kernels can be defined by the following recurrence formula

$$P^{k+1}(x, B) = \int_X P(t, B)P^k(x, dt), \forall x \in X, B \in \mathcal{B}, k > 1. \tag{4.27}$$

In particular, we have that $\mu_{X_j}(B) = \mu_{X_0} P^{(j)}(B)$, for every $j > 0$ and every $B \in \mathcal{B}$.

The transition kernel P is irreducible (with respect to some measure μ) if for every $x \in X$ and every $B \in \mathcal{B}$ there exists $j > 0$ such that $P^j(x, B) > 0$. Intuitively, the transition kernel P is irreducible if it visits every Borel set B with non-zero probability, regardless of the starting point x.

Finally, let us introduce another property of transition kernels. The irreducible transition kernel P is periodic if there exists a disjoint union of Borel sets $\{B_k\}_{k=0}^{n-1}$, where $n \geq 2$, such that $\bigsqcup B_k = X$ and

$$P(x, B_{k+1 \bmod n}) = 1, \qquad \forall x \in B_k, k = 0, \dots, n-1. \tag{4.28}$$

Intuitively, the transition kernel P is periodic if we can split X into more that one set B_k, such that the transition kernel moves the points from B_k to $B_{(k+1) \bmod n}$ almost surely. A transition kernel is aperiodic if it is not periodic.

Let us now state the theorem which formalizes the approximation formula (4.23).

Theorem 4.1. *Consider a probability measure μ and a Markov chain $\{X_j\}_{j=0}^{\infty}$ with an irreducible and aperiodic transition kernel P. Furthermore, let μ be an invariant measure of P. Then, we have that*

$$\lim_{N\to\infty} P^N(x, B) = \mu(B), \quad \forall B \in \mathcal{B}, \forall x \in X. \tag{4.29}$$

Moreover, for any $\ell \in L(\mu(dx))$ we have that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{j=1}^{N} \ell(X_j) = \int_X \ell(x)\mu(dx) \tag{4.30}$$

almost certainly.

Thus, in order to sample the probability measure μ with the MCMC method, we need to construct an irreducible aperiodic transition kernel P which preserves μ. One construction of such kernel is given by the Metropolis-Hastings (MH) algorithm, which is used in this thesis.

Let us now show how the transition kernel of Metropolis-Hastings algorithm is constructed. Consider a function $h : X \times X \to \mathbb{R}^+$, such that

$$\int_X h(x, y)dy = 1, \quad \forall x \in X. \tag{4.31}$$

This means that for every x the function $h(x, \cdot)$ is a probability density function of some random variable. We refer to h as a proposal distribution or a candidate-generating kernel. For every point $x \in X$, we use $h(x, \cdot)$ to propose a move from from a point x to a point $y \in X$, where the proposal y is sampled from a random variable with the probability density function given by $h(x, \cdot)$. Then, the Metropolis-Hastings algorithm accepts the move from x to y with a probability

$$\alpha(x, y) = \min\left(1, \frac{\pi(y)h(y, x)}{\pi(x)h(x, y)}\right), \tag{4.32}$$

where π is defined by $\pi(x)dx = \mu(dx)$. In case MH algorithm rejects the proposal, we stay in the point x, and sample the new proposal y. The transition kernel of Metropolis-Hastings algorithm can thus be represented with

$$P(x, B) = \int_X \alpha(x, y)h(x, y)dy + \left(1 - \int_X \alpha(x, y)h(x, y)dy\right)\chi_B(x), \quad \forall x \in X, \forall B \in \mathcal{B}, \tag{4.33}$$

where χ_B is a characteristic function of the set B. The pseudo-code for the MH method is given in Algorithm 2. The proposal distribution $h(x, \cdot)$ is sometimes chosen such that $h(x, \cdot)$ does not depend on x, or formally

$$h(x_1, y) = h(x_2, y), \quad \forall x_1, x_2, y \in X, \tag{4.34}$$

in which case the MCMC method is referred to as the independent sampler. Otherwise, the MCMC method is referred to as the random walk Metropolis (RWM). Common choice for $h(x, y)$ of RWM is such that $h(x, \cdot)$ is the p.d.f. of a normal random variable $\mathcal{N}(x, C)$.

Algorithm 2 Metropolis-Hastings Algorithm

1: **for** $n = 0$ to I **do**
2: Draw $v \sim h(\cdot|u_n)$,
3: Set $u_{n+1} = v$ with probability $\min(1, \alpha(v, u_n))$,
4: where $\alpha(v, u) = \frac{\pi(v)h(u|v)}{\pi(u)h(v|u)}$.
5: Otherwise, set $u_{n+1} = u_n$.

4.4 Sequential Monte-Carlo Methods for Inverse Problems

In this section we consider two different methods for inverse problems which approximate a probability measure μ with a weighted sum of Dirac deltas δ as

$$\mu(v) \approx \sum_{i=1}^{M} w_i \delta(v - \hat{v}_i), \qquad \sum_{i=1}^{M} w_i = 1, \tag{4.35}$$

where $M \in \mathbb{N}$ is the number of particles, and the values $\{\hat{v}_i\}_{i=1}^{M}$ are the discretization points. The first algorithm discussed in Section 4.4.1 gives an alternative way of sampling the posterior from (4.2). On the other side, the algorithm presented in Section 4.4.2 is specific to the case when the forward operator $\mathcal{G}$ is given as a solution to some ODE at multiple times $0 = \tau_0 < \tau_1 < \ldots < \tau_M$.

4.4.1 S-SMC Algorithm

In this section we introduce the sequential Monte Carlo algorithm, as presented in [12]. We only discuss the main ideas necessary for the implementation of the algorithm, and refer to [12] for more detailed analysis of the method. We will use an abbrevation S-SMC for the method discussed in this section.

Let us consider an integer $J > 1$, let $\tau = 1/J$, and define sequence of measures $\{\mu_j\}_{j=1}^{J}$ by a Radon-Nykodim derivative with respect to the prior μ_0 by

$$\begin{aligned} \frac{d\mu_j}{d\mu_0}(u) &= \frac{1}{Z_j} \exp(-j\tau\Phi(u)), \\ Z_j &= \int_X \exp(-j\tau\Phi(u))\mu_0(du), j = 0, \ldots, J. \end{aligned} \tag{4.36}$$

Observe that for $j = J$ we have that μ_J is actually the posterior $\mu^{\mathcal{Y}}$ as given in (4.10). S-SMC algorithm starts from the known measure μ_0, and then iteratively builds measures μ_j to arrive at the

solution μ_J. The key observation is that, while the prior μ_0 is very far from the posterior μ_J, the pairs of measures $\{\mu_j, \mu_{j+1}\}_{j=0}^{J=1}$ are very close to each other for a sufficiently small τ, and thus it is easier to construct the values μ_j iteratively than to construct the solution μ_J directly from the prior μ_0.

Let us now explain how we can use information about the measure μ_j to construct the measure μ_{j+1}. Using the chain rule for Radon-Nykodim derivatives $d\mu_j/d\mu_0$ and $d\mu_{j+1}/d\mu_0$ we get

$$\frac{d\mu_{j+1}}{d\mu_j} = \frac{Z_j}{Z_{j+1}} \exp(-\tau\Phi(u)). \tag{4.37}$$

Let us denote with L an operator that acts as an application of a Bayes theorem to the inverse problem with likelihood $\exp(-\tau\Phi(u))$. In particular, given some probability measure ν_1, the map $\nu_2 = L(\nu_1)$ is defined with

$$\frac{d\nu_2}{d\nu_1}(u) = \frac{\exp(-\tau\Phi(u))}{\int_X \exp(-\tau\Phi(v))\nu_1(dv)}. \tag{4.38}$$

Consider the probability measure $\bar{\mu}_{j+1} = L(\mu_j)$. We have that

$$\frac{d\bar{\mu}_{j+1}}{d\mu_j}(u) = \frac{\exp(-\tau\Phi(u))}{\int_X \exp(-\tau\Phi(u))\nu_j(du)}. \tag{4.39}$$

Since the non-constant expressions on the right hand side of equations (4.37) and (4.39) are the same, and because $\mu_j, \mu_{j+1}, \bar{\mu}_{j+1}$ are all probability measures, we have that $\mu_{j+1} = \bar{\mu}_{j+1}$. In particular, we have $\mu_{j+1} = L\mu_j$.

Let us denote with $\{P_j\}_{j=0}^{J}$ the family of Markov transition kernels, such that the measure μ_j is invariant measure of P_j. One specific construction of such transition kernels is given by the Metropolis-Hastings algorithm which is studied in Section 4.3. Since we have that $P_j\mu_j = \mu_j$ we can use the following equality

$$\mu_{j+1} = LP_j\mu_j, j = 0, \dots, J-1, \tag{4.40}$$

to construct a probability measure μ_{j+1} from μ_j.

At every step of the S-SMC algorithm we approximate measures $\{\mu_j\}_{j=0}^{J}$ with Dirac sums $\{\mu_j^M\}_{j=0}^{J}$ as in (4.35). In particular, for every $j = 1, \dots, J$, we write

$$\mu_j \approx \mu_j^M(v) = \sum_{n=1}^{M} w_j^{(n)} \delta(v - \hat{v}_j^{(n)}). \tag{4.41}$$

The first approximation μ_0^M is generated by sampling from μ_0. Let us split (4.40) into two steps, as

$$\begin{aligned} \hat{\mu}_{j+1} &= P_j\mu_j, \\ \mu_{j+1} &= L\hat{\mu}_{j+1}. \end{aligned} \tag{4.42}$$

Note that because μ_j is invariant to P_j, we have that

$$\frac{d\mu_{j+1}}{d\hat{\mu}_{j+1}} = \frac{Z_j}{Z_{j+1}} \exp(-\tau\Phi(u)). \tag{4.43}$$

Since every $\mu_j^M(v)$ is defined only through its set of weights $\{w_j^{(n)}\}_{n=1}^M$ and discretization points $\{\hat{v}_j^{(n)}\}_{n=1}^M$, for the construction of the method we only need to show how to update the pairs $\{\hat{v}_j^{(n)}, w_i^{(n)}\}_{n=1}^M$ using (4.40). Let us now consider that we are at the step $0 \leq j < J$, so that we have calculated μ_j^M. The result of the first step $\hat{\mu}_{j+1}^M = P_j \mu_j^M$ from (4.42) is given with

$$\hat{\mu}_{j+1}^M(v) = \sum_{n=1}^{M} w_j^{(n)} \delta(v - \hat{v}_{j+1}^{(n)}), \tag{4.44}$$

where values $\{\hat{v}_{j+1}^{(n)}\}_{n=1}^M$ are sampled from

$$\hat{v}_{j+1}^{(n)} \sim P_j(\hat{v}_j^{(n)}, \cdot), \quad n = 1, \dots, M. \tag{4.45}$$

Next, given $\hat{\mu}_{j+1}^M$, we calculate the variable $\bar{\mu}_{j+1}^M$ by

$$\bar{\mu}_{j+1}^M = \sum_{n=1}^{M} \bar{w}_{j+1}^{(n)} \delta(v - \hat{v}_{j+1}^{(n)}), \tag{4.46}$$

where weights $\bar{w}_{j+1}$ are calculated with

$$\bar{w}_{j+1}^{(n)} = \exp\left(-\tau\Phi(\hat{v}_{j+1}^{(n)})\right) w_j^{(n)}. \tag{4.47}$$

Finally, we normalize the weights $\{\bar{w}_{j+1}^{(n)}\}_{n=1}^M$ by

$$w_{j+1}^{(n)} = \frac{\bar{w}_{j+1}^{(n)}}{\sum_{i=1}^{M} \bar{w}_{j+1}^{(i)}}, \quad n = 1, \dots, M, \tag{4.48}$$

to get the probability measure for the next iteration

$$\mu_{j+1}^M = \sum_{n=1}^{M} w_{j+1}^{(n)} \delta(v - \hat{v}_{j+1}^{(n)}), \tag{4.49}$$

In case the number of particles M is sufficiently big, the result of S-SMC method μ_J^M well approximates the solution of inverse problem 4.5. In order to quantify the similarity of μ_J to the output of S-SMC method μ_J^M, let us introduce the total variation distance between measures μ, ν defined over

the set of events $\mathcal{F}$ by

$$d_{TV}(\mu, \nu) = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|. \tag{4.50}$$

Then, if we assume that there exist some $\varphi^-, \varphi^+ \in \mathbb{R}$ such that the potential Φ satisfies

$$\varphi^- \leq \Phi(u) \leq \varphi^+, \quad \forall u \in X, \tag{4.51}$$

the distance between measures $\mu^{\mathcal{Y}}$ and μ_J is bounded [3] by

$$d_{var}(\mu_J^M, \mu_J) \leq \sum_{j=1}^{J} (2\kappa^{-2})^j \frac{1}{\sqrt{M}}, \tag{4.52}$$

where κ is a constant, $\kappa \in [0, 1]$.

4.4.2 RPF-SMC Algorithm

In this section we introduce another SMC method which can be used for parameter estimation of ODEs. The method is a minor modification of particle filtering PF-SMC method given in [2]. Since the modification relies on usage of a randomized ODE method, we will refer to the method proposed in this section as RPF-SMC. Let consider the parametrized ODE (4.15) and let us denote the observations at times $t \in \{\tau_k\}_k^K$ with vectors $\mathcal{Y}_k \in \mathbb{R}^d, k = 1, \ldots, K$. Furthermore, let us denote with π_0 a probability density of the prior measure μ_0 such that $\mu_0(du) = \pi_0(u)du$. Finally, we consider the observational noise to be given with $\eta = \mathcal{N}(0, \sigma^2 I)$, where I is the identity matrix, so that the observational noise at the times $\{\tau_k\}_{k=1}^K$ is given by $\eta_k = \mathcal{N}(0, \sigma^2)$. Let us denote by $\{\rho_k\}_{k=1}^K$ the probability density function of observational noise $\{\eta_k\}_{k=1}^K$.

The algorithm can be explained in the following 6 steps.

1. Initialization: Draw $M \in \mathbb{N}$ particles from $\pi_0(u)$

$$\begin{aligned} \mathcal{S} &= \left\{(y_0^1, u_0^1, w_0^1), (y_0^2, u_0^2, w_0^2), \ldots, (y_0^M, u_0^M, w_0^M))\right\}, \\ w_0^1 = w_0^2 = \ldots = w_0^M &= 1/M, \quad y_0^1 = y_0^2 = \ldots = y_0^M = y_0. \end{aligned} \tag{4.53}$$

Define the mean and covariance of the parameters $\{u_0^i\}_{i=1}^M$ as

$$\bar{u}_0 = \frac{1}{M} \sum_{n=1}^{M} w_0^n u_0^n, \qquad \mathcal{C}_0 = \sum_{n=1}^{M} w_0^n (u_0^n - \bar{u}_0)(u_0^n - \bar{u}_0)^\top. \tag{4.54}$$

Let $j = 0$.

2. Propagation Shrink the parameters

$$\bar{u}_j^n = a u_j^n + (1 - a)\bar{u}_j, \quad j = 1, \ldots, M, \tag{4.55}$$

[3] See Theorem 5.13 in [12].

where $a \in [0, 1]$ is a real number which is given as an input to RPF-SMC algorithm. Usual choice for a is $a = 0.97$.

Compute new state predictions using a deterministic Runge-Kutta method

$$\bar{y}_{j+1}^n = \Psi_h(y_j^n, \bar{u}_j^n), \quad n = 1, \dots, M, \tag{4.56}$$

where for every $n = 1, \dots, M$ the map $\Psi_h(y_j^n, \bar{u}_j^n)$ denotes a numerical flow of an parametrized initial value problem (4.15) with an initial value y_j^n and parameter $u = \bar{u}_j^n$.

3. Survival of the fittest.

- Compute the fitness of each particle with

$$\hat{g}_{j+1}^n = w_j^n \rho_{j+1}(\mathcal{Y}_{j+1} - \bar{y}_{j+1}^n), \quad n = 1, \dots, M. \tag{4.57}$$

- Normalize the fitness weights

$$g_{j+1}^n = \frac{\hat{g}_{j+1}^n}{\sum_{i=1}^{M} \hat{g}_{j+1}^i}, \quad n = 1, \dots, M. \tag{4.58}$$

- Define random variable V_{j+1} with

$$P\left(V_{j+1} = n\right) = g_{j+1}^n, \quad n = 1, \dots, M, \tag{4.59}$$

  and make M draws $l_1, \dots, l_n$ from V_{j+1}.

- Shuffle

$$y_j^n = y_j^{l_n}, \quad (\bar{y}_{j+1}^n, \bar{u}_j^n) = (\bar{y}_{j+1}^{l_n}, \bar{u}_j^{l_n}), \quad n = 1, \dots, M. \tag{4.60}$$

4. Proliferation. For $n = 1, \dots, N$ do the following.

- Proliferate the parameters by

$$u_{j+1}^n \sim \mathcal{N}(\bar{u}_j^n, s^2 \mathcal{C}_j), \quad s^2 = 1 - a^2. \tag{4.61}$$

- Repropagate the states using the RTS-RK method

$$y_{j+1}^n = \Psi_h^s(y_j, u_{j+1}^n), \quad n = 1, \dots, M, \tag{4.62}$$

  where for every $n = 1, \dots, M$ the map $\Psi_h^s(y_j^n, \bar{u}_j^n)$ denotes a probabilistic numerical flow of the RTS-RK method for a parameterized initial value problem (4.15) with the initial value y_j^n and the parameter $u = \bar{u}_j^n$.

5. Update weights. Compute

$$\bar{w}_{j+1}^n = \frac{\rho_{j+1}\left(\mathcal{Y}_{j+1} - y_{j+1}^n\right)}{\rho_{j+1}\left(\mathcal{Y}_{j+1} - \bar{y}_{j+1}^n\right)}, \quad n = 1, \dots, N, \tag{4.63}$$

and normalize the weights with

$$w_{j+1}^n = \frac{\hat{w}_{j+1}^n}{\sum_{i=1}^N \hat{w}_{j+1}^i}, \quad n = 1, \dots, N. \tag{4.64}$$

6. Estimate mean and variance of parameters. Compute

$$\bar{u}_{j+1} = \sum_{n=1}^N w_{j+1}^n u_{j+1}^n, \quad \mathcal{C}_{j+1} = w_{j+1}^n (u_{j+1}^n - \bar{u}_{j+1})(u_{j+1}^n - \bar{u}_{j+1})^\top. \tag{4.65}$$

If $j = K - 1$, return $\bar{u}_{j+1}$ as a solution of algorithm, otherwise set $j := j + 1$ and go to step 2.

This finishes our exposition of the RPF-SMC algorithm. Let us remark that the solution given by RPF-SMC does not converge to the solution $\pi^{\mathcal{Y}}$ of inverse problem given in Section 4.1, but it can nonetheless be a good choice for ODE parameter estimation problems.

4.5 Inverse Problems with probabilistic forward operator

In this section we consider using randomized Runge-Kutta integrators for inverse problems in which the forward operator $\mathcal{G}$ is given by a solution of an ODE. For randomized methods, instead of the forward operator $\mathcal{G}_h$ which gets evaluated by a deterministic Runge-Kutta method, we consider the operator

$$\mathcal{G}_h^s : X \times \Omega \to Y, \tag{4.66}$$

which for some parameter u produces a random variable $G_h^s(u, \cdot)$. We use H to denote a member of the sample space Ω, and ν to denote a probability measure on Ω, $\nu(\Omega) = 1$.

Furthermore, we consider the operator $\mathcal{G}_h^s$ to have strong order p

$$\mathbb{E}_{H\sim\nu} \|\mathcal{G}_h^s(u, H) - \mathcal{G}(u)\| < Ch^p, \tag{4.67}$$

where $C \in \mathbb{R}^+$ is a constant which depends only on u. For example, the operator $\mathcal{G}_h^s$ is given by AN-RK or RTS-RK method. Let us fix a number M, and for a parameter u and vector $H = \{H_1, \dots, H_M\}$

of values $\{H_i\}_{i=1}^M$ sampled from ν let us define an unbiased estimator $\hat{\pi}(u, H)$ by

$$\hat{\pi}(u, H) = \frac{1}{M} \sum_{i=1}^{M} \rho\left(\mathcal{Y} - \mathcal{G}_h^s(u, H_i)\right) \pi_0(u). \tag{4.68}$$

In order to relax the notation, let us define ν^M as a product of M measures ν. In particular, we have

$$\nu^M = \underbrace{\nu \times \ldots \times \nu}_{\text{M times}}. \tag{4.69}$$

Then, we can say that the vector H is sampled from ν^M. Replacing the $\pi^{\mathcal{Y}}(u)$ with $\hat{\pi}(u, H)$ in Metropolis-Hastings scheme gives rise to "Monte Carlo within Metropolis" (MWCM) algorithm. For clarity, we provide the pseudo-code for MCWM method in Algorithm 3.

Algorithm 3 MCWM Algorithm

1: **for** $n = 1$ to I **do**
2: Draw $v \sim h(\cdot|u_n)$, and draw H_1, H_2 from ν^M.
3: Take $u_{n+1} = v$ with a probability $\min(1, \hat{\alpha}(v, u, H_1, H_2))$,
4: where $\hat{\alpha}(v, u, H_1, H_2) = \frac{\hat{\pi}(v, H_2) h(u|v)}{\pi(u, \hat{H}_1) h(v|u)}$.
5: Otherwise, set $u_{n+1} = u_n$.

Another approach to incorporate stochastic forward operator in the framework of Bayesian problems is to use pseudo-marginal Metropolis-Hastings (PMMH) algorithm proposed by Beaumont [7], and studied in more detail by Andrieu and Roberts [6]. In particular, the PMMH method (Algorithm 4) samples from a random variable with probability density function

$$\pi_s^Y(u) \propto \int_\Omega \rho(\mathcal{Y} - \mathcal{G}_h^s(u, H)) \nu(dH) \pi_0(u), \tag{4.70}$$

which approximates the exact solution $\pi^{\mathcal{Y}}$. The PMMH algorithm is very similar to the MCWM algorithm; the only difference comes from the fact that the PMMH algorithm recycles the value given by the estimator $\hat{\pi}$ in the previous step, while MCWM recomputes it.

Algorithm 4 PMMH Algorithm

1: Draw u from prior μ_0, H from μ^M and compute $\pi = \hat{\pi}(u, H)$.
2: **for** $n = 1$ to I **do**
3: Draw $v \sim h(\cdot|u_n)$,
4: Draw H from ν^M, and define $\pi^n = \hat{\pi}(v, H)$.
5: Take $(u_{n+1} = v, \pi = \pi^n)$ with probability $\min(1, \bar{\alpha}(v, u, \pi^n, \pi))$,
6: where $\bar{\alpha}(v, u, \pi^n)) = \frac{\pi^n h(u|v)}{\pi h(v|u)}$.
7: Otherwise, set $u_{n+1} = u_n$.

4.5.1 Bounds on Difference of Probabilities Produced by MCWM and MH Method

It is interesting to investigate the solution given by MCWM and PMMH algorithms in the limit case $h \to 0$. Since in this case the stochastic forward operator $\mathcal{G}_h^s$ is converging towards $\mathcal{G}$, we expect that the solutions of both MCWM and PMMH algorithms converge to the exact solution of the inverse problem $\pi^{\mathcal{Y}}$ given in (4.7). The solution of PMMH algorithm can be indeed pushed arbitrarily close to $\pi^{\mathcal{Y}}$ by choosing sufficiently small h, as it is showed in [22]. To the best knowledge of the author, this result was not available for MCWM algorithm so far. In this section we give a bound on the difference between the probability measure given by MCMC and MCWM methods, which solves this question.

In order to prove the main theorem, let us make the following assumptions.

Assumption 4.2. *The following statements hold true.*

- ***(A1)*** *Consider the forward operator $\mathcal{G}$, and the randomized forward operator $\mathcal{G}_h^s$. There exists a constant $C_\infty < \infty$ such that*

$$\|\mathcal{G}(X)\|_Y < C_\infty, \qquad \|\mathcal{G}_h^s(X, \Omega)\|_Y < C_\infty. \tag{4.71}$$

  In other words, operators $\mathcal{G}$ and $\mathcal{G}_h^s$ are uniformly bounded.

- ***(A2)*** *Consider the observational noise η from 4.5, with the probability density function ρ. We assume that*

$$\sup_{x,y \in Y} \left| 1 - \frac{\rho(x)}{\rho(y)} \right| < C_\rho \|x - y\|_Y \cdot r(x, y), \tag{4.72}$$

  where $r : Y \times Y \to \mathbb{R}$ is some continuous function.

Assumption (A2) is not very strong, and it holds for any zero mean normal random variable, as it is highlighted in the following lemma.

Lemma 4.1. *The probability density function ρ of a d-dimensional normal random variable $\mathcal{N}(0, \mathcal{C})$ satisfies Assumption (A2).*

Proof. The probability density function $\rho(x)$ is given by

$$\rho(x) = \frac{\exp(-1/2 x^\top \mathcal{C}^{-1} x)}{\sqrt{(2\pi)^d |\mathcal{C}|}}. \tag{4.73}$$

Thus, we have that

$$\left| 1 - \frac{\rho(u)}{\rho(v)} \right| = \left| 1 - \frac{\exp(-1/2 x^\top \mathcal{C}^{-1} x)}{\exp(-1/2 y^\top \mathcal{C}^{-1} y)} \right|. \tag{4.74}$$

Then, using

$$|1 - e^x| \leq |x| \exp(|x|), \forall x \in \mathbb{R}, \tag{4.75}$$

along with Cauchy-Schwarz inequality we can further bound (4.74) with

$$\left|1 - \frac{\exp(-1/2x^\top \mathcal{C}^{-1}x)}{\exp(-1/2y^\top \mathcal{C}^{-1}y)}\right| \leq \|x-y\|_Y \cdot g(x,y)\exp(\|x-y\|_Y \cdot g(x,y)), \tag{4.76}$$

where $g(x,y)$ is given with

$$g(x,y) = \frac{1}{2}\left(\|x^\top \mathcal{C}^{-1}\|_Y + \|\mathcal{C}^{-1}y^\top\|_Y\right). \tag{4.77}$$

Thus, the bound (4.72) of Assumption (A2) directly follows from (4.77) with $r(x,y) = g(x,y)\exp(\|x-y\|_Y \cdot g(x,y))$. □

The proof of the main theorem in this chapter relies on a theorem which is a minor adaptation of a result from [5].

Theorem 4.2. *Consider the Metropolis-Hastings Markov chain with the transition kernel P, and consider the transition kernel $\hat{P}$ of the MCWM algorithm. Let us assume that P is uniformly ergodic, i.e*

$$\sup_{u\in X} \|\delta_u P^n - \pi\| \leq C\gamma^n, \tag{4.78}$$

and let us assume that $\alpha(v,u)$ and $\hat{\alpha}(v,u,H_1,H_2)$ satisfy

$$\mathbb{E}_{H_1,H_2\sim\nu^M}|\alpha(v,u) - \hat{\alpha}(v,u,H_1,H_2)| \leq \delta(u,v), \tag{4.79}$$

where $\delta : X\times X \to \mathbb{R}$ is a function. Then, we have for any starting point u_0, and any $n\in\mathbb{N}$

$$\|\delta_{u_0}P^n - \delta_{u_0}\hat{P}^n\| \leq \left(\lambda + \frac{C\rho^\lambda}{1-\gamma}\right)\sup_{u\in X}\int h(v|u)\delta(u,v)dv, \tag{4.80}$$

where $\lambda = \left\lceil \frac{log(1/C)}{log\gamma}\right\rceil$.

Proof. The proof of this theorem is a straightforward extension of Corollary 2.3 from [5], which relies on the results from [23]. □

Under the assumption of uniform ergodicity, we need only to find a suitable function $\delta(u,v)$ such that (4.79) holds true, and use it in (4.80) to bound the difference between the probability measures produced by MCMC and MCWM methods. In the following theorem we propose a function $\delta(u,v)$ which satisfies the inequality (4.79).

Theorem 4.3. *Assume that the stochastic forward operator $\mathcal{G}_h^s$ has strong order p. Furthermore, assume that (A1) and (A2) hold. Then, one possible bound δ from (4.79) is given by*

$$\delta(u,v) = \frac{\pi_0(v)}{\pi_0(u)}\cdot C_\alpha h^p, \tag{4.81}$$

where $C_\alpha > 0$ is a constant independent on u and h.

Proof. Expanding the expression from (4.79) gives

$$\mathbb{E}_{H_1,H_2\sim\nu^M}|\alpha(v,u)-\hat{\alpha}(v,u,H_1,H_2)| = \int\int\left|\frac{\pi(v)}{\pi(u)}-\frac{\hat{\pi}(v,H_1)}{\hat{\pi}(u,H_2)}\right|\nu^M(dH_1)\nu^M(dH_2). \tag{4.82}$$

Let us define

$$D_1 = \frac{\pi(v)}{\pi(u)}-\frac{\hat{\pi}(v,H_1)}{\pi(u)}, \quad D_2 = \frac{\hat{\pi}(v,H_1)}{\pi(u)}-\frac{\hat{\pi}(v,H_1)}{\hat{\pi}(u,H_2)}. \tag{4.83}$$

We can now express (4.82) as

$$\int\int\left|D_1+D_2\right|\nu^M(dH_1)\nu^M(dH_2). \tag{4.84}$$

Using the triangular inequality, we get

$$\begin{aligned}&\mathbb{E}_{H_1,H_2\sim\nu^M}|\alpha(v,u)-\hat{\alpha}(v,u,H_1,H_2)| \leq \\ &\int\int\left|D_1\right|\nu^M(dH_1)\nu^M(dH_2)+\int\int\left|D_2\right|\nu^M(dH_1)\nu^M(dH_2).\end{aligned} \tag{4.85}$$

Now, let us define

$$A = \int\int\left|D_1\right|\nu^M(dH_1)\nu^M(dH_2), \quad B = \int\int\left|D_2\right|\nu^M(dH_1)\nu^M(dH_2), \tag{4.86}$$

and give bounds on A, B. We start by giving a bound on A.

$$\begin{aligned}A = \int\int\left|\frac{\pi(v)}{\pi(u)}-\frac{\hat{\pi}(v,H_1)}{\pi(u)}\right|\nu^M(dH_1)\nu^M(dH_2) = \int\left|\frac{\pi(v)}{\pi(u)}-\frac{\hat{\pi}(v,H_1)}{\pi(u)}\right|\nu^M(dH_1) = \\ \frac{\pi(v)}{\pi(u)}\cdot\int\left|1-\frac{\hat{\pi}(v,H_1))}{\pi(v))}\right|\nu(dH_1).\end{aligned} \tag{4.87}$$

Let us bound $\pi(v)/\pi(u)$ first. Let us expand the expression $\pi(u) = \pi_0(u)\rho\left(\mathcal{Y}-\mathcal{G}(u)\right)$ and consider

$$\frac{\pi(v)}{\pi(u)} \leq \frac{\pi_0(v)}{\pi_0(u)}\left|1-\frac{\rho\left(\mathcal{Y}-\mathcal{G}(v)\right)}{\rho\left(\mathcal{Y}-\mathcal{G}(u)\right)}-1\right| \leq \frac{\pi_0(v)}{\pi_0(u)}\left(\left|1-\frac{\rho\left(\mathcal{Y}-\mathcal{G}(v)\right)}{\rho\left(\mathcal{Y}-\mathcal{G}(u)\right)}\right|+1\right). \tag{4.88}$$

Using Assumption (A2) we can further bound (4.88) by

$$\frac{\pi(v)}{\pi(v)} \leq \frac{\pi_0(v)}{\pi_0(u)}\left(1+\|\mathcal{G}(u)-\mathcal{G}(v)\|_Y r\left(\mathcal{Y}-\mathcal{G}(v),\mathcal{Y}-\mathcal{G}(u)\right)\right). \tag{4.89}$$

Now, since by Assumption (A1) the operator $\mathcal{G}$ is uniformly bounded and $r(u,v)$ is continous, we have that the expression

$$\left(1+\|\mathcal{G}(u)-\mathcal{G}(v)\|_Y r\left(\mathcal{Y}-\mathcal{G}(v),\mathcal{Y}-\mathcal{G}(u)\right)\right) \tag{4.90}$$

in (4.89) is uniformly bounded as well by a constant $C_1 \in \mathbb{R}^+$. Therefore, we have that

$$\frac{\pi(v)}{\pi(u)} \leq C_1 \frac{\pi_0(v)}{\pi_0(u)} \tag{4.91}$$

Let us now bound the integral

$$\int \left| 1 - \frac{\hat{\pi}(v, H_1))}{\pi(v))} \right| \nu^M(dH_1), \tag{4.92}$$

from (4.87). Let us recall that H_1 can be expanded as $H_1 = \{H_1^1, H_1^2, \dots, H_1^M\}$ and write

$$\begin{aligned} &\int \left| 1 - \frac{\hat{\pi}(v, H_1))}{\pi(v))} \right| \nu^M(dH_1) \leq \int \left| \frac{1}{M} \sum_{i=1}^{M} 1 - \frac{\rho\left(\mathcal{Y} - \mathcal{G}_h^s(v, H_1^i)\right)}{\rho\left(\mathcal{Y} - \mathcal{G}(v)\right)} \right| \nu^M(dH_1) \leq \\ &\frac{1}{M} \sum_{i=1}^{M} \int \left| 1 - \frac{\rho\left(\mathcal{Y} - \mathcal{G}_h^s(v, H_1^i)\right)}{\rho\left(\mathcal{Y} - \mathcal{G}(v)\right)} \right| \nu(dH_1^i) = \int \left| 1 - \frac{\rho\left(\mathcal{Y} - \mathcal{G}_h^s(v, H)\right)}{\rho\left(\mathcal{Y} - \mathcal{G}(v)\right)} \right| \nu(dH), \end{aligned} \tag{4.93}$$

where in the last equality we used the fact that all random variables $H_1^1, H_1^2, \dots, H_1^M$ are independent and identically distributed as $H \sim \nu$. Finally, let us bound (4.93) using Assumption (A2) as

$$\int \left| 1 - \frac{\rho(\mathcal{Y} - \mathcal{G}_h^s(v, H))}{\rho(\mathcal{Y} - \mathcal{G}(v))} \right| \nu(dH) \leq \int \left\| \mathcal{G}(u) - \mathcal{G}_h^s(u, H) \right\|_Y \cdot r(\mathcal{Y} - \mathcal{G}_h^s(v, H), \mathcal{Y} - \mathcal{G}(v)) \nu(dH). \tag{4.94}$$

Because the operators $\mathcal{G}, \mathcal{G}_h^s$ are uniformly bounded as per Assumption (A1), and since $r(\cdot, \cdot)$ is continous, we can bound

$$r(\mathcal{Y} - \mathcal{G}_h^s(v, H), \mathcal{Y} - \mathcal{G}(v)) \tag{4.95}$$

from above by some constant C_2. Thus, we can further bound (4.94) with

$$\begin{aligned} \int \left\| \mathcal{G}(u) - \mathcal{G}_h^s(u, H) \right\|_Y \cdot r\left(\mathcal{Y} - \mathcal{G}_h^s(v, H), \mathcal{Y} - \mathcal{G}(v)\right) \nu(dH) &\leq C_2 \int \left\| \mathcal{G}(u) - \mathcal{G}_h^s(u, H) \right\|_Y \nu(dH) \\ &\leq C_2 C h^p, \end{aligned} \tag{4.96}$$

where in the last inequality we used the fact that $\mathcal{G}_h^s$ is of strong order p. Thus, A can be bounded with

$$A \leq C_1 C_2 C h^p \frac{\pi_0(v)}{\pi_0(u)}. \tag{4.97}$$

We can bound B in similar fashion, and we skip that proof for the sake of brevity. This finishes our proof. □

Substituting δ obtained in Theorem 4.3 to the expression (4.80) gives

$$\|\delta_{u_0} P^n - \delta_{u_0} \hat{P}^n\| \leq h^p C \sup_{u \in X} \int h(v|u) \frac{\pi_0(v)}{\pi_0(u)} dv, \tag{4.98}$$

where $C < \infty$ is a constant independent of h. If we assume that π_0 is supported in some compact set Θ, we can replace $\sup_{u\in X}$ from Theorem 4.2 with $\sup_{u\in\Theta}$. Then, if we assume that

$$\frac{1}{C_\pi} < \pi_0(u) < C_\pi, \quad \forall u \in \Theta, \tag{4.99}$$

where $C_\pi > 0$ is a constant independent of u, we can bound expression (4.98) with

$$\|\delta_{u_0} P^n - \delta_{u_0} \hat{P}^n\| < C_c h^p, \quad \forall u_0 \in \Theta, \tag{4.100}$$

where $C_c < \infty$ is a constant which does not depend on h. Now, since $\delta_{u_0} P^n \to \pi^{\mathcal{Y}}$ as $n \to \infty$, we have that

$$\limsup_{n\to\infty} \|\pi^{\mathcal{Y}} - \delta_{u_0} \hat{P}^n\| \leq C_c h^p, \quad \forall u_0 \in \Theta. \tag{4.101}$$

4.6 Numerical Experiments

In this section we provide numerical tests for the methods discussed in this chapter. In Section 4.6.1 we verify the correctness of MCMC, S-SMC and RPF-SMC methods by using them to estimate the parameter λ of the test equation (2.16). Next, in Section 4.6.2 we apply these methods to a more complicated inverse problem for which the forward operator $\mathcal{G}$ is given as a solution to the FitzHugh-Nagumo equation (3.22). Finally, in Section 4.6.3 we conduct a test on inverse problems in which the forward operator $\mathcal{G}$ is replaced by probabilistic map $\mathcal{G}_h^s$. In particular, we test PMMH and MCWM methods. Next, we consider the limit case when the variance of η from 4.5 is very small, where we show that the RTS-RK within PMMH scheme gives better results than the deterministicaly driven MCMC algorithm. RTS-RK within PMMH scheme is not always advantageous to deterministicaly driven MCMC methods, which is showed in the last experiment of this section.

4.6.1 MCMC, S-SMC, and RPF-SMC: Basic Tests

In this section we check the correctness of the methods presented in Sections 4.3 and 4.4, by applying them to the simple test equation 2.16. Let us recall the test equation

$$\dot{y} = \lambda y, \qquad y_0 = 1. \tag{4.102}$$

In the spirit of the discussion from Section 4.2, we take the unknown parameter u to be λ, and take the prior to be $u \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 0.05$. Let us consider observations at times $\tau_n = nT/K$, for $n = 1, \ldots, K, K = 100$. The forward operator G_h is evaluated using the explicit Euler method with $h = T/N, N = 5K$. The observations $\mathcal{Y}$ are generated by evaluating $\mathcal{G}_h(u)$ for $u = 1$ and with a very small step-size $h = T/N, N = 500K$. The noise η from inverse problem formulation

$$\mathcal{Y} = \mathcal{G}_h(u) + \eta, \tag{4.103}$$

is taken to be $\mathcal{N}(0,(0.01)^2 I_K)$, where I_K is K-dimensional identity matrix. The probability measures which describe the solutions given by Metropolis-Hastings, S-SMC and RPF-SMC methods are given in Figure 4.1. For the starting point of MCMC algorithm we have taken $X_0 = 2$. The S-SMC algorithm is used with $J = 20$ iteration steps and $M = 200$ particles. The RPF-SMC algorithm was used with $M = 10^4$ particles.

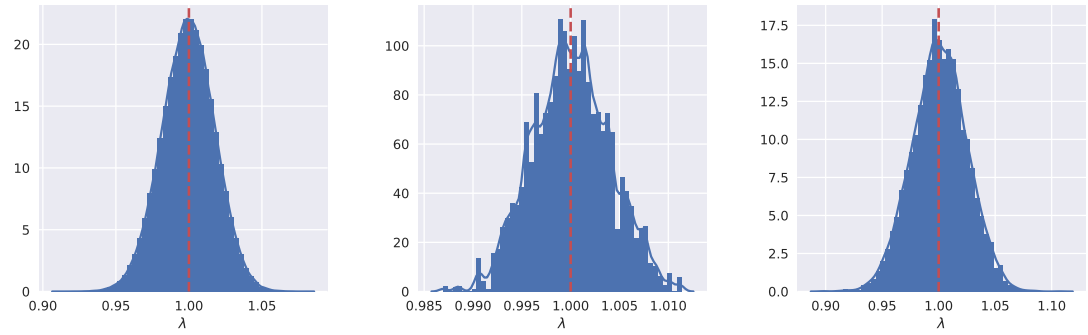


Fig. 4.1 Distribution of parameter inference for test equation calculated by Metropolis-Hastings (left), S-SMC (center) and RPF-SMC (right) methods.

4.6.2 MCMC, S-SMC and RPF-SMC: Bayesian Inference for FitzHugh-Nagumo Equation

In this section we use MCMC, S-SMC and RPF-SMC methods for estimating the parameters of FitzHugh-Nagumo equation. Let us recall the FitzHugh-Nagumo equation with

$$\begin{aligned} \frac{dV}{dt} &= c\left(V - \frac{V^3}{3} + R\right), \\ \frac{dR}{dt} &= -\frac{1}{c}\left(V - a + bR\right), \end{aligned} \tag{4.104}$$

For the initial condition we take $V(0) = -1$, $R(0) = 1$. In the spirit of Section 4.2, we take the forward operator $\mathcal{G}$ to be a solution of the equation at the observed times $\tau_n = nT/K, n = 1, \ldots, K, K = 100$. We consider the parameter u to be

$$u = (\log(a), \log(b), \log(c))^\top, \tag{4.105}$$

in order to ensure the that the values a, b, c are positive for every $u \in \mathbb{R}^3$. The approximation $\mathcal{G}_h$ of the forward operator $\mathcal{G}$ is provided by Heun's method (2.13), with the step-size $h = T/N$, $N = 5K$.

The observations $\mathcal{Y}$ are generated by evaluating $\mathcal{G}_h(u)$ for $u = (\log(0.2), \log(0.2), \log(3))^\top$ and a very small step-size $h = T/N$, $N = 500K$. The noise η from the inverse problem formulation

$$\mathcal{Y} = \mathcal{G}_h(u) + \eta, \tag{4.106}$$

is taken to be $\mathcal{N}(0, (0.05)^2 I_{2K})$, where I_{2K} is $2K$-dimensional identity matrix. The probability measures which describe the inferred probability of the parameter c given by Metropolis-Hastings, S-SMC and RPF-SMC methods are given in Figure 4.2. For the starting point of MCMC algorithm we have taken $X_0 = (0, 0, 0)$. The S-SMC algorithm is used with $J = 200$ iteration steps and $M = 2000$ particles. The RPF-SMC algorithm was used with $M = 10^5$ particles.

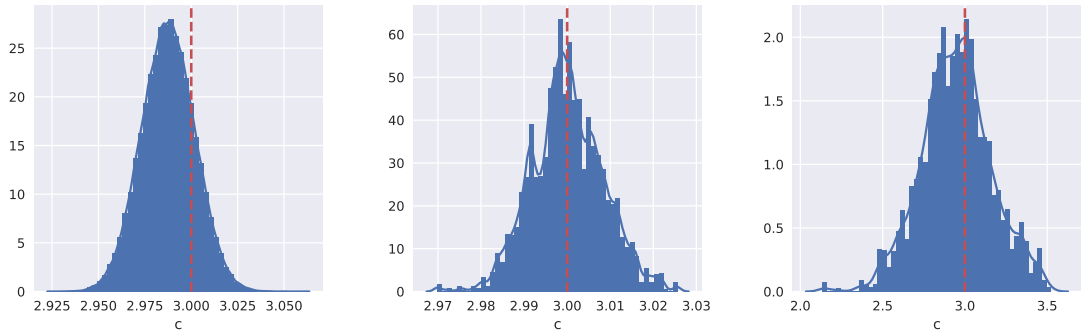


Fig. 4.2 Histogram for parameter c of FitzHugh-Nagumo model (3.22), produced by Metropolis-Hastings (left), S-SMC (center), and RPF-SMC (right) algorithm. The distribution of c is marginalized over other two parameters (a, b).

4.6.3 Randomized Runge-Kutt Methods and Inverse Problems: Numerical Tests

In this section we numerically study the solutions of inverse problems when the forward operator $\mathcal{G}$ is approximated by the stochastic operator $\mathcal{G}_h^s$. We first compare the solutions given by Monte-Carlo within Metropolis (MCWM) and pseudo-marginal Metropolis-Hastings algorithms. Then, we numerically show that the stochastic forward operator $\mathcal{G}_h^s$ within PMMH algorithm can account better for the uncertainty of the numerical solver and thus give better results than the deterministic $\mathcal{G}_h$ within Metropolis-Hastings algorithm. Unfortunately, this does not always needs to be the case, which we show in the last numerical test of this section.

In this section we consider the test equation (4.102). We make observations only at the final time, so therefore we have $K = 1$ and $\tau_1 = T$. We take the unknown parameter u to be λ, and model the belief about u with $u \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 1$. For each test in this section we consider the forward operator $\mathcal{G}_h$ to be evaluated with the explicit Euler method, and probabilistic forward operator $\mathcal{G}_h^s$ to be evaluated with RTS-RK method with the explicit Euler method as an underlying Runge-Kutta scheme. We always use $h = T/N$, $N = 20$.

Let us first consider the solutions provided by PMMH and MCWM methods. We take the observational noise η to be $\eta \sim \mathcal{N}(0, (0.1)^2)$. The results produced by PMMH and MCWM methods are given in Figure 4.3. From this figure we can observe the typical difference between the solutions produced by PMMH and MCWM methods: the probability measure generated by MCWM method tends to be much wider than the one produced by PMMH algorithm.

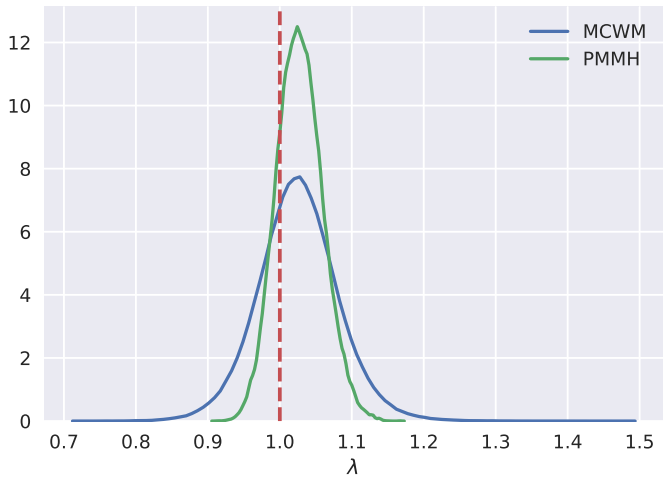


Fig. 4.3 Histograms for parameter λ of test-equation model (4.102) produced by PMMH (green) and MCWM (blue).

Let us now compare the solution given by the PMMH scheme with the stochastic forward operator $\mathcal{G}_h^s$ to the solution produced by MH with the deterministic forward operator $\mathcal{G}_h$. We consider the case when the variance of the noise η is much smaller; in particular, $\eta \sim \mathcal{N}(0, (0.03)^2)$. The results are given in Figure 4.4.

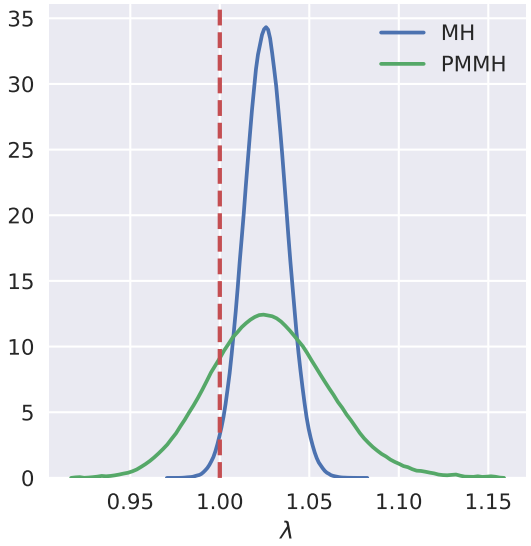


Fig. 4.4 Histograms for parameter λ of test-equation model (4.102) produced by PMMH (green) and MH (blue) schemes. The forward operator $\mathcal{G}_h^s$ is evaluated by the RTS-RK method with the explicit Euler as an underlying Runge-Kutta scheme.

From Figure 4.4 we can see that PMMH gives better results than MH algorithm. Unfortunately, this is not always the case. To show this, let us consider the observations times $\tau_n = nT/K, n = 1, \dots, K, K = 10$, and consider the noise η to be $\eta \sim \mathcal{N}(0, (0.03)^2 I_K)$. We use the PMMH method with the RTS-RK scheme with the probabilistic step-sizes $\{H_k\}_{k=1}^N$ chosen with $H_k = h + \zeta^h, k = 1, \dots, N$, where ζ^h is a random variable with probability density function f_{ζ^h} defined as

$$f_{\zeta^h} = \begin{cases} b/(a+b), & -ah < x < 0, \\ a/(a+b), & 0 \leq x \leq hb, \\ 0, & \text{otherwise}, \end{cases} \tag{4.107}$$

where $a, b \in \mathbb{R}^+$ and $a < 1/h$. Let us denote that this choice of $\{H_k\}_{k=1}^N$ is a generalization of (3.17), since the choice (3.17) is obtained for $a = b = 1$. It is straightforward to check that the variables H_k satisfy Assumption 3.4, and thus by Theorem 3.3 the RTS-RK method with this choice of H_k converges with strong order 1. The comparison of the PMMH scheme with such choice of RTS-RK method to the solution produced by MH algorithm is given in Figure 4.5. For ζ^h we choose $a = 0.1$ and $b = 10$.

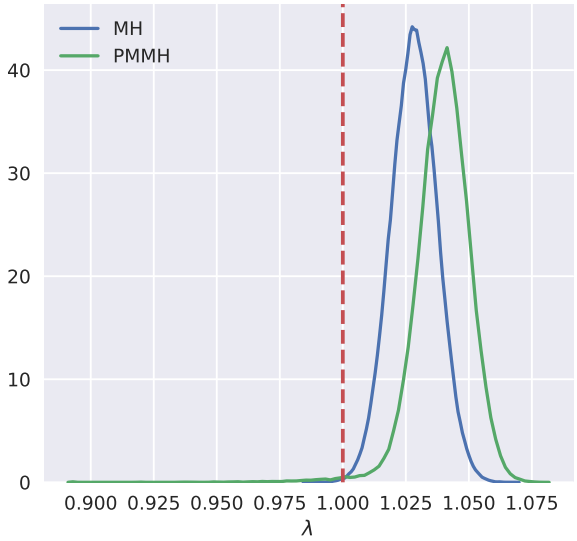


Fig. 4.5 Comparison of PMMH and RTS-RK method, with RTS-RK method being driven by the explicit Euler method and skewed choice of random step-sized given by $H_k = h + \zeta^h$.

Chapter 5

Bayesian Inverse Problems in Infinite-Dimensional Setting

In the previous chapter we have introduced Bayesian inverse problems. The discussion presented was restricted to the finite-dimensional case, which allowed us to motivate the approach in a clear way, and also use the Metropolis-Hastings method for sampling the solution of the inverse problem which was given as probability measure $\mu^{\mathcal{Y}}$. In this chapter we extend the topic by addressing necessary adaptations required for the infinite-dimensional case.

In Section 5.1 we discuss the changes to the approach from the previous chapter necessary for dealing with infinite-dimensional Bayesian inverse problems. In particular, we define the solution of infinite-dimensional problem, and discuss how the prior measure μ_0 can be specified. Furthermore, we introduce the preconditioned Crank-Nicolson method (pCN) [9], which is an MCMC method used to sample from the solution given as a probability measure $\mu^{\mathcal{Y}}$.

Then, in Section 5.2 we consider the parabolic Brusselator partial differential equation, and show how this equation can be suitably solved with the method of lines. Finally, in Section 5.3 we use the framework developed in Section 5.1 to perform numerical tests for the Brusselator problem.

5.1 Bayesian Inverse Problems in Infinite-Dimensional Setting

Similar to Chapter 4, let us consider the equation

$$\mathcal{Y} = \mathcal{G}(u) + \eta, \tag{5.1}$$

where the forward operator $\mathcal{G} : X \to Y$ is a map between possibly infinite dimensional Hilbert spaces X, Y. We consider the observations $\mathcal{Y}$ to be given, η to be a random variable, and search for *suitable* values of u. Under Assumption 4.1 on the potential function $\Phi : X \times Y \to \mathbb{R}$ we can show

that the expression (4.10) is well defined, and thus we consider the probability measure $\mu^{\mathcal{Y}}$ given by

$$\frac{d\mu^{\mathcal{Y}}}{d\mu_0} \propto e^{-\Phi(u,\mathcal{Y})}, \tag{5.2}$$

to be the solution of the inverse problem.

Let us first discuss how the prior μ_0 can be specified. The prior μ_0 is usually taken to be a Gaussian measure $\mathcal{N}(m_0, \mathcal{C})$, where $m_0 \in X$ is a mean, and $\mathcal{C}$ is a covariance operator. The covariance operator $\mathcal{C}$ is taken to be compact and positive-definite, which by spectral theorem allows for specification of an orthonormal basis $\{\phi_i\}_{i=1}^{\infty}$ as

$$\mathcal{C}\phi_i = \lambda_i^2 \phi_i, \quad i = 1, 2, \dots. \tag{5.3}$$

Furthermore, we assume that $\mathcal{C}$ is a trace-class, so that eigenvalues $\{\lambda_i^2\}_{i=1}^{\infty}$ satisfy

$$\sum_{i=1}^{\infty} \lambda_i^2 < \infty. \tag{5.4}$$

Such a choice of covariance $\mathcal{C}$ allows for a representation of the prior μ_0 via a Karhunen–Loéve expansion with

$$\mu_0(x) = m_0 + \sum_{i=1}^{\infty} \lambda_i \phi_i \varepsilon_i, \tag{5.5}$$

where $\{\varepsilon_i\}_{i=1}^{\infty}$ are i.i.d. random variables, $\varepsilon_i \sim \mathcal{N}(0, 1)$. Since by spectral theorem we have $\lambda_i \to 0$ as $i \to \infty$, the measure μ_0 can approximated by the projection $\mathcal{P}^{d_u}$ of Karhunen–Loéve expansion (5.5) onto the first d_u nodes with

$$\mathcal{P}^{d_u}(\mu_0) = m_0 + \sum_{i=1}^{d_u} \lambda_i \phi_i \varepsilon_i. \tag{5.6}$$

Moreover, since the eigenvalues $\{\lambda_n\}_{n=1}^{\infty}$ in most common cases converge quickly towards 0 (usually $\lambda_n \sim n^{-2}$), the measure μ_0 is well approximated by $\mathcal{P}^{d_u}$ for small values of d_u (usually $d_u > 10$ is sufficient).

Let us now consider the problem of sampling the posterior $\mu^{\mathcal{Y}}$ using the formula (5.2) and specification of the prior as in (5.5). The main idea is to consider algorithms which are well defined for infinite-dimensional representation of the prior (5.5), since these algorithms are robust under the mesh refinement $d_u \to \infty$. One such algorithm is the pCN method, which belongs to the class of MCMC methods and is studied in [9]. Let us recall from the discussion on MCMC methods in Section 4.3 that any MCMC method can be defined by specifying its transition kernel.

In order to motivate construction of the transition kernel for pCN method, let us first consider the Langevin stochastic partial differential equation

$$\frac{du}{ds} = -\mathcal{K}(\mathcal{L}u + \gamma D\Phi(u)) + \sqrt{2\mathcal{K}}\frac{dW}{ds}, \tag{5.7}$$

where $\mathcal{L} = \mathcal{C}^{-1}$, $\mathcal{K} = \mathcal{C}$ or $\mathcal{K} = I$, and W is a Brownian motion. The square root of $\mathcal{K}$ can be defined through diagonalization of $\mathcal{C}$ in Karhunen–Loéve basis. The Langevin equation preserves the measure μ_0 for $\gamma = 0$, and the measure $\mu^{\mathcal{Y}}$ for $\gamma = 1$ [10, 15, 16]. Let us fix $\gamma = 0$ and $\mathcal{K} = \mathcal{C}$, for which the equation (5.7) becomes

$$\frac{du}{ds} = -u + \sqrt{2\mathcal{K}}\frac{dW}{ds}. \tag{5.8}$$

Different choices of γ or $\mathcal{K}$ are used in derivation of other schemes, such as Metropolis-adjusted Langevin (MALA) algorithm, which are discussed in [9]. Approximating du/ds with finite differences and considering Crank-Nicolson approximation of the linear part of (5.8), we get

$$\frac{v-u}{\delta} = \frac{u+v}{2} + \sqrt{2\mathcal{K}}\frac{dW}{ds}. \tag{5.9}$$

Consider $\delta \in [0, 2]$, and let us define $\beta^2 = 8\delta/(2+\delta)^2$. Then we can rewrite (5.9) as

$$v = \sqrt{1-\beta^2}u + \beta w, \tag{5.10}$$

where $w \sim \mathcal{N}(0, \mathcal{C})$. The equation (5.10) is used to propose a move from a point u to a point v in the pCN method. Actually, the equation (5.10) introduces the candidate-generating kernel $h(u, v)$ of the pCN method. Finally, to conclude the introduction of the pCN method we define accept-reject formula with

$$\alpha(u, v) = \min\left(1, \alpha(u, v)\right), \tag{5.11}$$

where $\alpha : X \times X \to \mathbb{R}^+$ is given with

$$\alpha(u, v) = \exp(\Phi(u) - \Phi(v)). \tag{5.12}$$

Let us remark that due to the careful construction of the candidate-generating kernel, there is no need to include the prior in the accept-reject formula for (5.12), as shown by Theorem 6.2 in [9]. The pseudo-code for the pCN method is given in Algorithm 5.

Algorithm 5 pCN Algorithm

1: Draw u_0 from the truncated prior $\mathcal{P}^{d_u}(\mu_0)$.
2: **for** $n = 0$ to I **do**
3: Propose $v_n \sim \sqrt{1-\beta^2}u + \beta\varepsilon_n, \varepsilon_k \sim \mathcal{N}(0, \mathcal{C})$
4: Set $u_{n+1} = v_n$ with probability $\min(1, \alpha(v_n, u_n))$,
5: where $\alpha(v, u) = \exp(\Phi(u) - \Phi(v))$.
6: Otherwise, set $u_{n+1} = u_n$.

5.2 Brusselator Problem

We now consider the Brusselator problem

$$\begin{aligned} \frac{\partial U}{\partial t} &= a + U^2V - (1+b)U + \nu\frac{\partial^2 U}{\partial x^2}, \\ \frac{\partial V}{\partial t} &= bU - U^2V + \nu\frac{\partial^2 V}{\partial x^2}, \end{aligned} \tag{5.13}$$

where $x \in (0, 1)$, the parameters take values $a = 1, b = 3, \nu = 0.01$. The boundary conditions we consider are

$$U(0,t) = U(1,t) = 1, \qquad V(0,t) = V(1,t) = 3, \tag{5.14}$$

with the initial conditions

$$U(x,0) = 1 + sin(2\pi x), \qquad V(x,0) = 3. \tag{5.15}$$

We solve this problem using the method of lines (MOL), first by discretizing the equation (5.13) in space using $k+1$ evenly spaced nodes $\{x_n\}_{n=0}^k$, with $x_n = n\Delta_x, \Delta_x = 1/k$. By introducing the notation $U_i = U(t, x_i), V_i = V(t, x_i)$ and approximating $\partial^2 U/\partial x^2$ and $\partial^2 V/\partial x^2$ with centered finite differences

$$\begin{aligned} \frac{\partial^2 U_i}{\partial x^2} &\sim \frac{U_{i-1} - 2U_i + U_i}{\Delta_x^2}, \quad i = 1, \dots, k-1, \\ \frac{\partial^2 V_i}{\partial x^2} &\sim \frac{V_{i-1} - 2V_i + V_i}{\Delta_x^2}, \quad i = 1, \dots, k-1, \end{aligned} \tag{5.16}$$

the equation (5.13) becomes

$$\begin{aligned} \frac{\partial U_i}{\partial t} &= a + U_i^2V_i - (1+b)U_i + \nu\frac{U_{i-1} - 2U_i + U_i}{\Delta_x^2}, \\ \frac{\partial V_i}{\partial t} &= bU_i - u_i^2V_i + \nu\frac{V_{i-1} - 2V_i + V_i}{\Delta_x^2}, \quad i = 1, \dots, k-1. \end{aligned} \tag{5.17}$$

The values on the boundary nodes are fixed by the boundary condition $U_0 = U_k = 1, V_0 = V_k = 3$. We can now use Runge-Kutta integrator to solve (5.17) forward in time. Since the stiffness of the problem increases with the number of nodes, explicit stabilized Runge-Kutta methods such as RKC

or ROCK2 are a good choice for integration of this equation in time. In Figure 5.1 we plot the result of a numerical simulation with $k = 30$ discretization nodes. For integrating (5.17) we used ROCK2 method (2.23) with step-size $h = T/N$, $N = 200$.

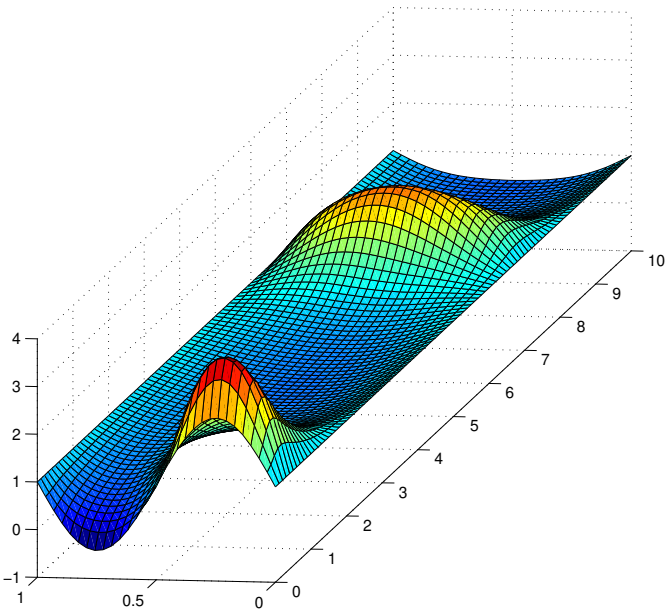


Fig. 5.1 Numerical solution of Brusselator problem (5.13) solved by method of lines and ROCK2 method. Space variable x is uniformly discretized on 30 nodes, while the time is discretized on 200 nodes.

5.3 Bayesian Inference Tests

In this section we consider inverse Bayesian problem in which the forward operator $\mathcal{G}(u) \to \mathcal{Y}$ is given as a solution to the Brusselator problem (5.13). We consider the unknown prior u to be initial condition of Brusselator equation,

$$u = (U(\cdot, 0), V(\cdot, 0)) . \tag{5.18}$$

The belief about the prior u is modeled with a probability measure $\mu_0 = \mu_U \times \mu_V$, where $\mu_U \sim \mathcal{N}(1, \mathcal{C})$ and $\mu_V \sim \mathcal{N}(3, \mathcal{C})$ are Gaussian priors on functions $U(\cdot, 0)$ and $V(\cdot, 0)$, respectively. For the covariance operator $\mathcal{C}$ we consider $\mathcal{C} = \gamma^2 \cdot (-\triangle)^{-2}$, where $\triangle$ is Laplacian operator and $\gamma \in \mathbb{R}^+$. Since the domain for both $U(\cdot, 0)$ and $V(\cdot, 0)$ is $(0, 1)$, the eigenvectors of the negative Laplacian are

$$\left\{ \sqrt{2} \sin(2n\pi x) \right\}_{n=1}^{\infty} , \left\{ \sqrt{2} \cos(2n\pi x) \right\}_{n=1}^{\infty} , \tag{5.19}$$

with eigenvalues $\{(2n\pi)^2\}_{n=1}^{\infty}, \{(2n\pi)^2\}_{n=1}^{\infty}$, respectively. Thus, the eigenvectors of the operator $\mathcal{C} = (-\triangle)^{-2}$ are

$$\left\{\sqrt{2}\sin(2n\pi x)\right\}_{n=1}^{\infty}, \left\{\sqrt{2}\cos(2n\pi x)\right\}_{n=1}^{\infty}, \tag{5.20}$$

with eigenvalues being $\{1/(2n\pi)^4\}_{n=1}^{\infty}, \{1/(2n\pi)^4\}_{n=1}^{\infty}$ respectively. Then, the prior μ_U can be expressed using a Karhunen–Loève expansion with

$$\mu_U = 1 + \gamma \cdot \sum_{n=1}^{\infty} a_n\sqrt{2}\sin(2n\pi x)\chi_i + b_n\sqrt{2}\cos(2n\pi x)\varepsilon_i, \tag{5.21}$$

where $\chi_i, \varepsilon_i \sim \mathcal{N}(0,1)$ are i.i.d. random variables, and $a_n = b_n = 1/(2\pi n)^2, n \in \mathbb{N}$. Since the boundary condition is fixed, we drop the cosine eigenvectors in Karhunen–Loéve basis (5.21), and thus the prior measure for function $U(\cdot,0)$ becomes

$$\mu_U = 1 + \gamma \cdot \sum_{n=1}^{\infty} a_n\sqrt{2}\sin(2n\pi x)\chi_i. \tag{5.22}$$

Similar to μ_U, we model the prior μ_V on the initial condition $V(\cdot,0)$ with a Karhunen–Loéve expansion

$$\mu_V = 3 + \gamma \cdot \sum_{i=1}^{\infty} a_n\sqrt{2}\sin(2n\pi x)\varepsilon_i. \tag{5.23}$$

In all computations the Karhunen–Loéve bases are truncated to the first $d_u = 10$ modes.

The forward operator $\mathcal{G}$ is approximated by the method of lines with $k = 30$ discretization nodes, and ROCK2 method for integration in time t with a step-size $h = T/N, N = 200$. The observations $\mathcal{Y}$ are calculated at times $\tau_k = kT/K, k = 1,\dots,K, K = 10$ on the discretization nodes $\{U_i\}_{i=1}^{k-1}, \{V_i\}_{i=1}^{k-1}$, using the ROCK2 method with a very fine step-size h. For the noise we choose $\eta \sim \mathcal{N}(0,(0.02)^2 I)$. We visualize the application of the pCN method in this setting in Figure 5.2, where we plot the mean of steps $U_n, n = 1000,\dots,I$ on discretization nodes $x_i, i = 0,\dots,k$ for the function U of Brusselator problem. Furthermore, we plot the standard deviations at every node, as well as the exact initial condition for $U(\cdot,0)$ from (5.15). Let us remark that the first 1000 proposals are not considered in plotting the pCN results, since the beginning of the MCMC chain usually poorly represents the distribution. Results are obtained with $I = 10^5$ proposals. The acceptance rate for pCN method is close to 10%, with $\beta = 0.1$.

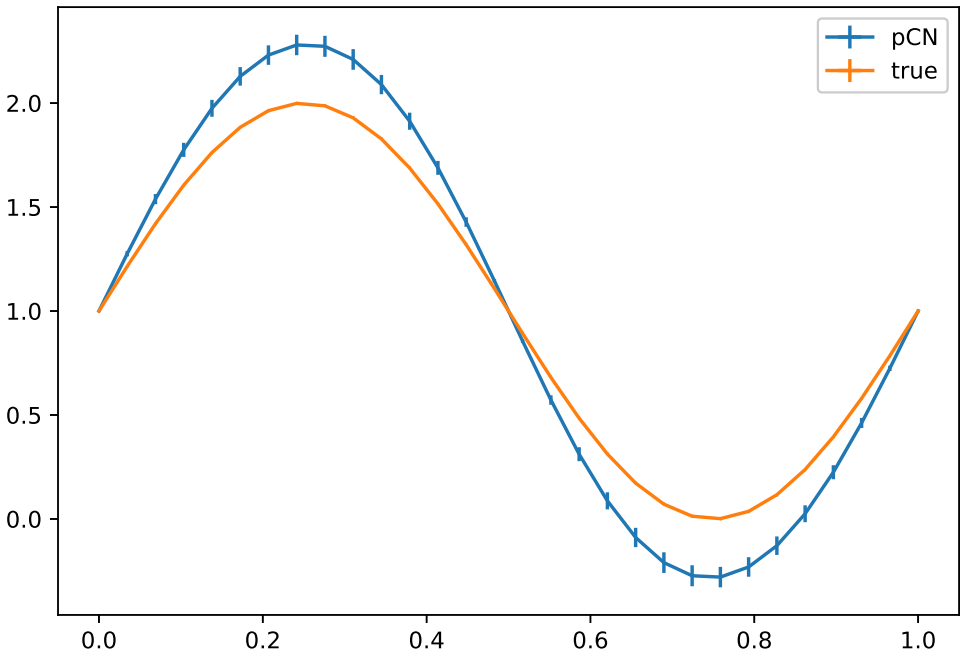


Fig. 5.2 Brusselator Bayesian inference of initial condition, generated by pCN method. Orange line shows the value of true input, while the blue line shows estimated values.

Chapter 6

Conclusion

In this thesis we studied Runge-Kutta methods for solving ordinary differential equations. We have particularly considered stiff problems, and discussed how they can be solved using explicit stabilized Runge-Kutta methods. We have also presented an adaptive time-stepping scheme which improves efficiency of Runge-Kutta methods.

Part of the study was dedicated to probabilistic numerical solvers for ODEs. We have covered recent results in the literature, in particular the additive noise Runge-Kutta method (AN-RK) [8] and the random time-stepping Runge-Kutta method (RTS-RK) [3]. Furthermore, we have devised an efficient adaptive strategy for the RTS-RK method. We also studied adaptive strategy for the RTS-RK method which uses only statistical information to quantify the local integration error, which did not show good performance.

Central research topic of this thesis were Bayesian inverse problems. We first focused on the finite-dimensional setting, for which we defined the solution as a random variable. We have then discussed Markov Chain Monte Carlo (MCMC) methods, which were used to sample the solution. Apart studying MCMC methods, we have presented two particle filtering methods. We have also formulated ODE parameter estimation problems as Bayesian inverse problems. Particular attention was devoted to using probabilistic numerical integrators such as AN-RK and RTS-RK method in Bayesian inverse problems for ODEs. We have introduced two MCMC methods which are used for probabilistic numerical integrators: pseudo-marginal Metropolis-Hastings (PMMH) and Monte Carlo within Metropolis (MCWM). To this best knowledge of the author, in this thesis a new result on convergence properties of MCWM method is presented.

In the final chapter of this thesis we have considered infinite dimensional Bayesian inverse problems. We have discussed what are the necessary changes required for dealing with infinite dimensional case. We have introduced pCN method, which was used in inverse problem of recovering the initial condition of the Brusselator problem.

References

[1] A. Watts, H. (1984). Step size control in ordinary differential equation solvers. 1:15–25.

[2] A.Arnold (2014). *Sequential Monte Carlo parameter estimation for differential equations*. PhD thesis, Case Western Reserve University, Cleveland, OH.

[3] Abdulle, A. and Garegnani, G. (2017). Random time step probabilistic methods for uncertainty quantification in chaotic and geometric numerical integration. *publication*.

[4] Abdulle, A. and Medovikov, A. A. (2001). Second order chebyshev methods based on orthogonal polynomials. *Numerische Mathematik*, 90(1):1–18.

[5] Alquier, P., Friel, N., Everitt, R., and Boland, A. (2016). Noisy monte carlo: convergence of markov chains with approximate transition kernels. *Statistics and Computing*, 26(1):29–47.

[6] Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient monte carlo computations. *Ann. Statist.*, 37(2):697–725.

[7] Beaumont, M. A. (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160.

[8] Conrad, P. R., Girolami, M., Särkkä, S., Stuart, A., and Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, 27(4):1065–1082.

[9] Cotter, S. L., Roberts, G. O., Stuart, A. M., and White, D. (2013). Mcmc methods for functions: Modifying old algorithms to make them faster. *Statist. Sci.*, 28(3):424–446.

[10] Da Prato, G. and Zabczyk, J. (2014). *Stochastic Equations in Infinite Dimensions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition.

[11] Dahlquist, G. G. (1963). A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43.

[12] Dashti, M. and Stuart, A. M. (2013). The Bayesian Approach To Inverse Problems. *ArXiv e-prints*.

[13] Gustafsson, K. (1994). Control-theoretic techniques for stepsize selection in implicit runge-kutta methods. *ACM Trans. Math. Softw.*, 20(4):496–517.

[14] Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations; 2nd ed.* Springer, Dordrecht.

[15] Hairer, M., Stuart, A. M., and Voss, J. (2007). Analysis of spdes arising in path sampling part ii: The nonlinear case. *Ann. Appl. Probab.*, 17(5/6):1657–1706.

[16] Hairer, M., Stuart, A. M., Voss, J., and Wiberg, P. (2005). Analysis of spdes arising in path sampling. part i: The gaussian case. *Communications in Mathematical Sciences*, 3(4):587–603.

[17] I. Lebedev, V. (1994). Zolotarev polynomials and extremum problems. 9:231–264.

[18] Jari Kaipio, E. S. (2005). *Statistical and Computational Inverse Problems*. Springer-Verlag New York.

[19] Kersting, H. and Hennig, P. (2016). Active uncertainty calibration in bayesian ode solvers. *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, pages 309–318.

[20] Lebedev, V. I. (1994). How to solve stiff systems of differential equations by explicit methods. 9:231–263.

[21] L.F.Olsen (1983). An enzyme reaction with a strange attractor. *Phys. Lett. A*, (94):454–457.

[22] Medina-Aguayo, F. J., Lee, A., and Roberts, G. O. (2016). Stability of noisy metropolis–hastings. *Statistics and Computing*, 26(6):1187–1211.

[23] Mitrophanov, A. Y. (2005). Sensitivity and convergence of uniformly ergodic markov chains. *J. Appl. Probab.*, 42(4):1003–1014.

[24] Ramsay, J. O., Hooker, G., Campbell, D., and Cao, J. (2007). Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796.

[25] Sommeijer, B., Shampine, L., and Verwer, J. (1998). Rkc: An explicit solver for parabolic pdes. *Journal of Computational and Applied Mathematics*, 88(2):315 – 326.

[26] Stuart, A. M. (2010). Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559.

[27] van Der Houwen, P. J. and Sommeijer, B. P. (1980). On the internal stability of explicit, m-stage runge-kutta methods for large m-values. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 60(10):479–485.