UNIVERSITY OF LEICESTER

EXPLICIT REPRESENTATIONS OF PERIODIC SOLUTIONS OF NONLINEARLY PARAMETERIZED ORDINARY DIFFERENTIAL EQUATIONS AND THEIR APPLICATIONS TO INVERSE PROBLEMS

Thesis submitted for the degree of

Doctor of Philosophy

at the

University of Leicester

by

Jehan Mohammed Al-Ameri

Department of Mathematics

University of Leicester

October 2018

Abstract

Developing mathematical models involves joining theory and experimental or observational data. The models often depend on parameters which are not always known or measured. A major task in this process is therefore to determine parameters fitting empirical observations. In this work we consider the fundamental challenge of inferring parameters of systems of ordinary differential equations (ODEs) from the values of their solutions and/or their continuous mappings. To achieve this aim we developed a method for deriving computationally efficient representations of solutions of parametrized systems of ODEs. These representations depend on parameters of the system explicitly, as quadratures of some known parametrized computable functions. The method applies to systems featuring both linear and nonlinear parametrization, and time-varying right-hand side; which opens possibilities to invoke scalable parallel computations for numerical evaluation of solutions for various parameter values.

In the core of the methods the idea is to use availability of parallel computational streams offered by modern computational technology and hardware, such as GPUs. This, if used efficiently, drastically reduces the amount of time spent on solving direct problems. This opens up new possibilities for dealing with inverse problems by employing the methods that have not been possible to use to date due to massive computational costs involved.

We illustrate our method with parameter estimation problems for classical benchmark models of neuron cells, Hodgkin–Huxley and Morris–Lecar models. These applications enable to assess potential computational advantage, of the method relative to other procedures known in the literature; they also offer new ways to more forward.

Acknowledgements

As I near the end of a long academic journey, I should like to thank all those wonderful people who helped me along the way.

First, my supervisor Prof. Ivan Tyukin, who offered his invaluable experience. Thank you for your untiring patience and guidance at every stage of my work! Then, Mrs. Tatiana Tyukina, without whose caring support I would never have learnt Cuda programming!

My husband Akram Rodeen and my beautiful children Mohammed, Ismael and Ibrahim were always there for me, with their unquestioning faith in me, and their love!

I want to humbly, deeply express my thanks to all the members of my family - my parents, Mohammed Khudhir and Sadiyah Hadi; sisters, Eman, Jenan, Heba, Esraa and Soad; brothers, Asaad and Amjad, and my uncle, Miyah Rodeen. They have always prayed for success in my life, and in my studies. I would to acknowledge the kindness and guidance my brother Dr. Amjad Miyah Rodden provided throughout my studies and my life.

I am also deeply grateful to my teachers at Basrah university - Mathematics department: Ass.Prof. Basil Luqa, Ass.Prof. Abdul Nabi Ibrahim, Prof. Raad Mahdi, Prof. Husam Luti and Dr. Ayad Raysan. They always encouraged and supported me when I was an undergraduate student and when I was studying for my Master's. They have unceasingly been a source of wisdom, help and constancy, right until this moment, as I am completing my Ph.D.

Big thanks to the Programme administrator of Mathematics department in Leicester University, Miss. Charlotte Langley for her fast reply and help for any enquiry I have.

I am also greatly indebted to my sponsor - the Ministry of Higher Education in Iraq - for offering me this unique opportunity to study for a doctorate outside Iraq.

I would like to thank Prof. Alexander Gorban and Prof. Jeremy Levesley and everybody at Leicester university - Mathematics department. I would also to thank all my colleagues in the UK and Iraq, without whom it would have been a

hard and lonely time. They gave me so much - their companionship, their advice, their suggestions, ideas, inspiration and endless good humour.

To all of you, the warmest thanks imaginable, and my sincerest appreciation of everything you have done for me!

Yours,

Contents

List of Figures

List of Tables

"Remember to look up to the stars and not down at your feet. Try to make sense of what you see and wonder about what makes the universe exist. Be curious. And however difficult life may seem, there is always something you can do and succeed at. It matters that you don't give up."

—Stephen Hawking

Notation and Abbreviations

Notations

- Symbol $\mathbb{R}$ denotes the field of real numbers, and $\mathbb{R}^n$ stands for the $n-$dimensional real space, $\mathcal{C}^r$ denotes the space of continuous functions that are at least r times differentiable.
- $\|x\| = \sqrt{\sum_{i=1}^{n} |x_i|^2}$ is the Euclidean norm for $x = \text{col}(x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$.
- The notation $|\cdot|$ stands for the absolute value of a scalar.
- $L_{\infty,[t_0,t_0+T]}$ denotes the space of all functions $f : [t_0, t_0 + T] \to \mathbb{R}^n$ such that $\|f\|_{\infty,[t_0,t_0+T]} = \max_{\tau\in[t_0,t_0+T]} \|f(\tau)\| < \infty$, and $\|f\|_{\infty,[t_0,t_0+T]}$ stands for the $L_{\infty,[t_0,t_0+T]}$ norm of $f(\cdot)$.
- $\vartheta \in \Omega_\vartheta \subset \mathbb{R}^p$ denotes the unknown parameters of the system of ordinary differential equations $\dot{x} = f(t, x, \vartheta)$.
- Solutions (if they exist) of the system of ordinary differential equations $\dot{x} = f(t, x, \vartheta)$, where $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, passing through the point $x_0 \in \mathbb{R}^n$ at $t = t_0$ are denoted by $x(t; t_0, x_0, \vartheta)$.
- Suppose that we have a continuous functional π mapping a function $y : \mathbb{R} \to \mathbb{R}^n$ to an element of $\mathbb{R}$. We will refer to this functional as $\pi([y])$.

- The symbol I_n represents the identity matrix in $\mathbb{R}^{n\times n}$ while the system $0_{p\times 1}$ represents the zero vector in $\mathbb{R}^{p\times 1}$.

Abbreviations

ODEs	Ordinary Differential Equations
NLP	Non-Linear Programming Problem
BVP	Boundary Value Problem
IVP	Initial Value Problem
NM	Nelder Mead Algorithm
INM	Improved Nelder Mead Algorithm
RBFs	Radial Basis Functions
CPU	Central processing unit (standard computers).
GPU	Graphics processing unit.
CUDA	Programming parallel computing platform.

Chapter 1

Introduction

Many physical process are modelled by systems of linear or non-linear ordinary differential equations (ODEs). These equations generally depend on parameters, some of which have physical meaning. Ability to measure these parameters with known accuracy is therefore a prerequisite for scientific discovery and progress. Despite the substantial progress that has been made to date to improve our ability to measure physical variables which are relevant for more developments (for instance [83], [20], [1], [101]), the challenge of measuring parameters remains open; this is particularly true in areas such as cell physiology, high performance engineering, and studies of complex chemical reactions. In these areas, measuring all relevant quantities directly is either technically challenging (*e.g.* measuring pressure at the tip of the drilling bores) or too invasive (isolation of individual currents in neural membranes) [21]. In some cases, indirect observation for some variables and physical quantities is possible, on the basis of knowledge of underlying physical and mathematical models (*e.g.* [30], [29]). Therefore, it is necessary to know first if inference of their values is at all possible. If so then the challenge is to estimate these parameters precisely. Variables unavailable for direct observation may include the state and parameter values of the model. In the context of this thesis, such models are defined as systems of ODEs of the first order:

$$\begin{aligned} \dot{x} &= f(t, x, \vartheta);\ x(t_0) = x_0 \in \mathbb{R}^n \\ y &= h(x) \end{aligned} \tag{1.1}$$

In this system, $x \in \mathbb{R}^n$ denotes the state vector, $\vartheta \in \mathbb{R}^p$ is the vector of unknown parameters, x_0 is the initial condition and $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is a vector field describing the dynamics of the system, $y \in \mathbb{R}$ is the "measured" quantity. The "measurement" process is modelled by a continuous map $h : \mathbb{R}^n \to \mathbb{R}$.

Without imposing any conditions on the function f the differential equation (1.1) may not have a solution for all $t \geq t_0$, and the uniqueness of the solution of the system is not guaranteed. The existence and uniqueness the solution of the system can be guaranteed in $\mathcal{D}$ (open subset of $\mathbb{R}\times\mathbb{R}^n$), however, by requiring that the function f satisfies $\|f(t,x,\vartheta)-f(t,x',\vartheta)\| < L_{\mathcal{D}}\|x-x'\|$ for all (t,x) and (t,x') $\in \mathcal{D} \subset \mathbb{R}\times\mathbb{R}^n$ and for some fixed $L_{\mathcal{D}} \in \mathbb{R}_{>0}$, known as the Lipschitz continuity on the domain $\mathcal{D}$. A sufficient condition for a function f to be Lipschitz is that the Jacobian, $\partial f/\partial x$, is uniformly bounded in t for all (t,x) in $\mathcal{D}$ [6]. Unless stated otherwise we shall assume that these conditions hold.

Many strategies for addressing parameter estimation problems in this similar setting have been developed to date, including but not limited to shooting methods [14], sensitivity functions [8], splines [116] and adaptive observers [10], [75], [12], [31], [108], [106] (see also [65], [100] for system-identification take on the problem). All these methods involve fitting of modelled trajectories or variables to empirical data. However, such methods are often difficult to apply in practice because of various specific issues; examples of these issues include but are not limited to the necessity to access derivatives of x, general nonlinear parametrization, slow convergence, ill-conditioned problems, etc.

State and parameter estimation of linear and nonlinear systems has been an active research topic for many decades [64, 71, 72, 115, 50]. In the case of system parameters being unknown, an effective method is the utilisation of an adaptive observer[1] for simultaneous estimation of states and parameters. Consequently, over the last decades, a very active and extensive research area into the design of adaptive observers has arisen. In general, adaptive observers have two tasks to perform: a) to provide an estimation of the initial values of state variables and b)

[1] An observer is an auxilary system of differential equations $\dot{\hat{x}} = \tilde{f}(t,\hat{x});\ \hat{x}(t_0) = \hat{x}_0 \in \mathbb{R}^n$ whose outputs are the estimates of the state variables of the system as in (1.1). These outputs must converge to $x(\cdot;t_0,x_0)$ in forward time. In a special case, when these outputs coincide with $\hat{x}$, the latter condition can be formalized $\lim_{t\to\infty}(x(t;t_0,x_0)-\hat{x}(t;t_0,\hat{x}_0)) = 0$.

at the same time to provide an estimation of the unknown (constant) parameters. The use of such observers is especially significant in applications of a challenging nature, *i.e.* adaptive control and fault detection and isolation (see for instance [4] and [118]). The seminal contributions in adaptive observer design (see for instance [70] and [60]) have largely focused on linear time-invariant systems. Investigations into linear time varying systems have recently been carried out within deterministic and stochastic contexts [118], [88]. A variety of approaches have been employed to regarding the design of adaptive observers for nonlinear systems.

A large proportion of recent developments in the field of adaptive observers concerns a dynamic transformation of the original system into an observer canonical form [75, 76, 77, 74]. In [75] the author demonstrated that adaptive observer can be applied to solve a class of single-output nonlinear systems which are linear with respect to an unknown constant parameter vector. He proposed a simple adaptive observer that achieves convergence of the state estimate without requiring for persistent excitation (PE)[2] of the state variables explicitly. A filtered nonlinear transformation, namely a transformation that depends upon the unknown parameters to enlarge the class of nonlinear systems that can be turned into the adaptive observer form, has been given in [76]. It showed that any system in a (*global*) observer canonical form can be put into a (*global*) adaptive observer form, thus admits which is the so called Marino-Tomei observer. Another adaptive observer was proposed in [77] with the presentation of an arbitrary fast exponential rate of convergence for both parameters and state estimates. [74] presented robust adaptive observers for a class of nonlinear systems, subject to a number of sufficient conditions for a state estimate to converge asymptotically. Primarily, major body of work on adaptive nonlinear observer design has been focused on systems with linear parametrizations(see *i.g.* [55], [70], [102]); some results on nolinear parametrized systems can be found in [31, 66, 59, 58, 99, 108, 113].

Notwithstanding the progress, we first note that all these works (including the current thesis) are based on several fundamental assumptions: about models and their parameters. True values of model parameters are the ones that are

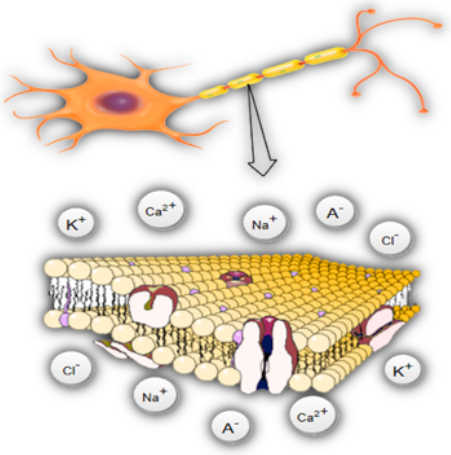
[2]Recall that a function $\Theta : \mathbb{R} \to \mathbb{R}^{\iota}$ is PE if there exist $L, \delta > 0$ such that $\int_t^{t+L} \Theta(\tau)\, \Theta^T(\tau) d\tau > \delta I_\iota$ for all t [67]. In this definition, the notation $\int_t^{t+L} \Theta(\tau)\, \Theta^T(\tau) d\tau > \delta I_\iota$ is to be understood that the matrix $\int_t^{t+L} \Theta(\tau)\, \Theta^T(\tau) d\tau - \delta I_\iota$ is positive-definite.

supposed to reflect relevant physical meaning or properties of the phenomenon captured by the model. These parameters can be given by experts or otherwise chosen. We do not like to question these choices here. We assume that they exist and are defined somehow. We also assume that, for these true parameter values, the model behaviour and the data match each other in some metrics. For example, measured data, if represented by a continuous function, is close to model trajectories in L_∞ or L_2 norm on some domain of relevance. These two assumptions form the basis on our subsequent mathematical problem statement.

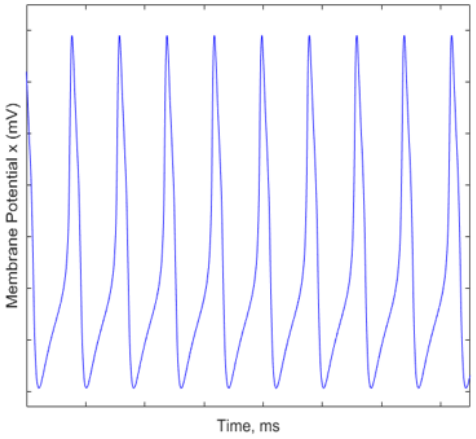
Consider an example of modelling steps illustrating these assumptions for the problem of parameter estimation. The diagram (a) in Figure 1.1 shows a basic phenomenological prescription of how currents propagate through a patch of the cell's membrane where there is number of voltage-dependent channels, such as for Ca, Na and K depicted in the figure. Recording currents through a single channel in the membrane is not always possible [106]. Thus they must be estimated from available measurements, such as the membrane potentials represented in Figure 1.1 (b). This figure represents the membrane potential of the neuron at time t for relevant true values of parameters; it exhibits oscillations to mimic the spiking behaviour. We consider Morris–Lecar system in (c) as a simple example of a mathematical model of neuron cells. Parameters of this model are not linked directly to kinetic parameters of the ion pumps. These parameters do, however, represent the pumps functionally. In this respect, there is a demand for robust methods to estimate parameters from data, even when these parameters are not "physical" variables or quantities. These parameters do, however, translate into physically meaningful behaviour as is shown in (d).

Modelling behaviour of such parametrized problems requires tools for obtaining accurate numerical solutions of systems of ordinary differential equations. Here we are dealing with uncertain systems in which the parameters in the right-hand side of the corresponding differential equations enter the equations nonlinearly. These are needed for both direct and inverse problems.

This problem alone, *i.e.* nonlinear parametrization, presents a major theoretical and practical issue. Despite this area has seen significant progress to



(a)



(b)

$$\dot{x} = g_{Ca} m_\infty(x)(x - E_{Ca}) + g_K q(x + E_K) + g_L(x + E_L) + I$$

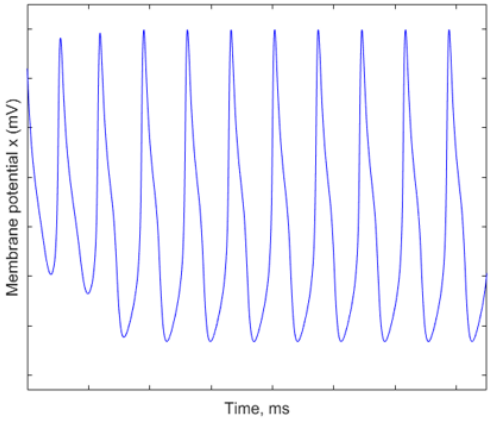
$$\dot{q} = -\frac{1}{\tau(x)} q + \frac{\omega_\infty}{\tau(x)}$$

$$\omega_\infty(x) = 0.5\left(1 + \tanh\left(\frac{x - v_1}{v_2}\right)\right)$$

$$m_\infty(x) = 0.5\left(1 + \tanh\left(\frac{x + v_3}{v_4}\right)\right)$$

$$\tau(x) = T_0 / \left(\cosh\left(\frac{x + v_3}{2v_4}\right)\right)$$

(c)



(d)

FIGURE 1.1: (a) and (b) show physical variables of cell's membrane and the measured membrane potential variable, respectively. (c) and (d) show Moris–Lecar example and the behaviour of membrane potential at some estimated values of parameters, respectively.

date ([108], [21], [5], [63], [31], etc), the existing solutions typically assume monotonicity of nonlinearities or their application is limited to functions in which the nonlinearity is "low-dimensional" (see *i.e.* [107] for more details on this particular approach and interpretation). Alternative yet popular approaches, such as swarm optimization [97], genetic algorithms [114] often lack sufficient mathematical rigor; sensitivity functions [42] are local and computationally expensive. All these methods, additionally require efficient solvers of the direct problem.

The main focus and motivation of my thesis are standing on the following hierarchical lines of inquiries:

Q1 - How can we efficiently estimate unknown parameters and state variables of nonlinear systems of ODEs (1.1) with nonlinearly parametrized so that the measured output of the model matches the data?

Q2 - How do we know that the estimate $\hat{\vartheta}$ of ϑ is closed enough to the unknown ϑ in some sense?

Q3 - How to employ scalable parallel computations for numerical evaluations of the representations solutions of nonlinear systems to make the calculations faster?

We attempt to answer all these inquiries in this thesis.

The approach we took can be expressed as "what if" scenario. If solutions of system of nonlinear ODEs would be known, as explicit and easily computable functions of parameters and initial conditions, then the problem would be reduced to standard nonlinear programming problems (NLP) with known cost functions. Finding a solution of a system of ODEs is often understood as determining a finite sum of elementary functions which satisfies the given ODE systems and initial conditions. This process involves integration of a given function (as is example the case for separable variables equations).

In the 19th century Liouville stated and proved an influential theorem which roughly states that if the integral of an elementary function is elementary, then it can be expressed using only functions that appear in the integrand and a linear combination of logarithms of such functions [103]. This theorem is now known as Liouville's theorem; its statement is presented bellow:

Theorem 1.1. *(Liouville)[56] If y and z are algebraic functions of x whose derivatives, $\frac{dy}{dx}$, $\frac{dz}{dx}$, are each algebraic functions of x,y and z, and if P is an algebraic function of x,y and z such that $\int P dx$ is a finite function, then*

$$\int P dx = a + A \log b + B \log c + \cdots + D \log d, \tag{1.2}$$

where $A, B, \cdots, D$ *are constants, and* $a, b, c, \cdots, d$ *are algebraic functions of* x,y *and* z.

An algebraic function $y = f(x_1, x_2, \cdots, x_n)$ can be defined as the root of an $(n+1)$ polynomial equation $\mathcal{P}(x_1, x_2, \cdots, x_n, y) = 0$; its values are determined by the values of n independent variables, $x_1, x_2, \cdots, x_n$. Algebraic functions are algebraic expressions with a finite number of terms, involving only the algebraic operations addition, subtraction, multiplication, division, and raising to a fractional power. The equation $\mathcal{P} = 0$ can be solved explicitly for y, but y is an algebraic function of the other quantities in the equation; examples of these quantities are (which we focus on in this thesis) parameters and initial points. We show some examples to what Liouville calls a finite function, and what we now call an elementary function. Liouville showed that some integrals can be expressed as finite sum of elementary functions, for example, $\int e^{x^2} y dx$ if y is an algebraic function in x. This, however, is impossible if the integrand P is not merely algebraic but rational in x,y, for example, $e^{x^2} y/x$ and $\sin xy/(1 + x^2)$ (further details can be found in [56]). During the rest of 19th and through the 20th centuries some developments were made on Liouville's theorem, such as [47], [92], [95], [93], [94]. Some of them provided algorithms for integrating functions. These, however, require multiple substitutions and can lead to large systems of equations to be solved symbolically. Additionally, other computational difficulties may arise as is exemplified in [95].

Despite of the "prohibitively" looking statement of the Liouville's theorem and its other developments over many years we have seen significant technological progress that radically effects our computational capability. We may not be able to write the solutions down but we can estimate them numerically. Parallel and cheap computing create unprecedented opportunities to speed up the calculations; seizing on this opportunities is the focus of this work. The advanced programming parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.

The aim of this thesis is to produce efficient derivation of solutions of system of ordinary differential equations, including linear and nonlinear equations.

The long-standing challenge relates to methods for representing the solution of such systems as integrals of known functions. This contrasts with Liouville's ambition to find finite-sum representation of solutions. But nevertheless, it is a step forward from numerical methods perspective. The motivation is to use the adaptive observer design to create our representations under the periodicity condition. Different representations of periodic solutions of such systems of nonlinear ordinary differential equations are suggested (see published papers [107] and [82]). We demonstrate that these methods are faster, if run on parallel computational streams, than similar-accuracy conventional iterative numerical routines. Furthermore, with the use of Radial Basis functions, we were able to offer next avenues for further improvements. We also show that our adaptive algorithm operates well from the point of view of accuracy, contrasting with estimator methods such as the shooting methods discussed in Chapter 2 regarding estimating the initial conditions of the state variables of ODE systems. Moreover, we show that the RBF approximations can be employed to numerically solve this problem efficiently (details of numerical simulations are given in Chapter 4). For this sort of problems, we did not aim at achieving the best possible accuracy of estimation. Instead, the aim was to demonstrate feasibility.

The thesis is organised as follows. In Chapter 2, we consider the numerical methods, single and multiple shooting methods, and illustrate their applications for solving relevant practical problems. These applications are considered for two biological models, Hodgkin–Huxley and ECG models. In Chapter 3 we proposed two different representations of periodic solution of systems of ODEs and explained in details. The first representation was published in [107], and the second one is presented in IFAC 2017 world Congress and published in [82]. These representations contrast with all the foregoing methods of observation. They allow for the reconstruction of parameters entering the model nonlinearly. In certain circumstances, these methods encompass not only parameters entering the model nonlinearly but also state variables. A higher computational cost will be involved, however, as constant computational re-evaluation of certain integrals is required as the exploration proceeds. In Section 3.4 the efficiency of the representations in solving systems of ODEs is illustrated in detail with the Hodgkin–Huxley and

Morris–Licar models. Results of the estimation for both observers are presented and compared. To show how the methods speed up the calculations, we implemented these methods by CUDA on GPU. In Chapter 4 we present an approach how Gaussian radial basis functions can be used to approximate some integrals in the final formulae.

1.1 Publications

The contribution of this research has been disseminated in the following papers:

1. Mohammed, J. A. and Tyukin, I. (2017). Explicit parameter-dependent representations of periodic solutions for a class of nonlinear systems. IFAC-PapersOnLine, 50(1):4001–4007.

2. Tyukin, I. Y., Gorban, A., Tyukina, T., Al-Ameri, J., and Korablev, Y. A. (2016). Fast sampling of evolving systems with periodic trajectories. Mathematical Modelling of Natural Phenomena, 11(4):73–88.

3. Adebayo, D., Al-Ameri, J., Tyukin, I., & Rona, A. (2018). Linear stability analysis of the flow between rotating cylinders of wide gap. European Journal of Mechanics-B/Fluids, 72, 567-575.

4. Tyukin, I. Y., Al-Ameri, J. M., Gorban, A. N., Levesley, J., & Terekhov, V. A. (2018, June). Fast Numerical Evaluation of Periodic Solutions for a Class of Nonlinear Systems and Its Applications for Parameter Estimation Problems. In International Conference on Optimization Problems and Their Applications (pp. 137-151). Springer, Cham.

Chapter 2

Numerical Methods for solving Non-linear Parametrized Systems Of Ordinary Differential Equations and Case Studies

2.1 Parameter Estimation for Ordinary Differential Equations

Consider a system of first-order ordinary differential equations:

$$\begin{aligned} \dot{x} &= f(t, x, \vartheta);\ x(t_0) = x_0 \in \mathbb{R}^n \\ y &= x_1, \end{aligned} \tag{2.1}$$

where x is the state variable, y is the output variable, f is a nonlinear function of the vector of model parameters ϑ, state variable x. It is assumed that f is Lipschitz in x, and continuous in ϑ and t, so that (2.1) has an unique solution.

The parameterized system can be represented in the extended form:

$$\begin{aligned} \dot{x} &= f(t, x, \vartheta);\ x(t_0) = x_0 \in \mathbb{R}^n \\ \dot{\vartheta} &= 0_p \\ y &= x_1. \end{aligned} \tag{2.2}$$

where 0_p is a $p \times 1$ zero vector. The objective is to find a set of parameter estimates, $\hat{\vartheta}$, given one measurement of the response outputs at the time $t \in [t_0, t_0 + T]$, $y = x_1(t)$, so that the output predicted by the model within the estimated parameters,

$\hat{\vartheta}$, is as close as possible to the true process response.

A common approach for determining $\hat{\vartheta}$ [78] is the so called Nonlinear Least-Squares (NLS) method in which the sum of squared deviations of the model predictions from the measured output is minimized:

$$\begin{aligned} &\hat{\vartheta} = \arg\min_{\vartheta}\{\sum_{i=1}^{N}(y(t_i) - \hat{y}(t_i, \vartheta))^2\} \\ &\text{subject to} \\ &\dot{x} = f(t, x, \vartheta);\ x(t_0) = x_0 \in \mathbb{R}^n \\ &\dot{\vartheta} = 0_p \\ &y = x_1 \end{aligned} \tag{2.3}$$

for some given $N \in \mathbb{Z}^+$. The sum could be replaced with an integral.

In NLS a nonlinear minimization technique is employed along with an ODE solver to find the optimal set of parameter estimates [90, 62, 37]. Popular sets of these methods are single and multiple shooting methods [43], sensitivity functions [7], and spline collocation methods [116].

We will briefly review these methods and give examples for some of them to illustrate difficulties that these methods may lead to.

2.2 Shooting Methods

2.2.1 Single Shooting Method

Single shooting methods are utilized to solve (2.1). In the single shooting method, we first use initial guesses for ϑ_0 and x_0 to numerically solve the initial value problem (forward problem) using single- or multi-step methods such as Runge Kutta methods [51, 18]. The least square error in (2.3) is computed as the next step, and then new values for ϑ_0 and x_0 can be found, depending on the optimization strategy, *for example*, gradient-based strategies such as Gauss-Newton methods or direct search approaches such as the Nelder-Mead Method which we will be explained in detail in section 2.5. This process continues until we find $\hat{\vartheta}$ and $\hat{x}_0$ which are expected to be within the same neighbourhood of optimal values, with

a pre-defined optimality tolerance (alternatively, they could be local minima). In practice, we never know the values, and hence some indirect stopping rules are used: errors between $y(\cdot, \vartheta)$ and $\hat{y}(\cdot, \hat{\vartheta})$ are small enough.

There could, however, be some issues with single shooting method. For example, the error may be large, the convergence can be quite slow, in addition to that the boundary value problem might be unstable, even when it is well-conditioned [119]. A solution of the initial value problem may not exist over the whole interval for a given ϑ_0 and x_0 . Resolving these disadvantages of the single shooting method can be carried out by utilising what is known as the multiple shooting method.

2.2.2 Multiple Shooting Method

Multiple shooting method divides the relevant time interval $[t_0, t_0 + T]$ for the model into K smaller subintervals, introduces K shooting nodes, $x_0(t_\kappa)$, $\kappa = 0, 1, \cdots, K-1$ for each subinterval, and solve the individual initial value problem (2.2) on each subinterval using single- or multi-step methods such as Runge–Kutta methods. As this method reformulates the problem as a constrained optimization (2.3) for every subinterval, it is more robust in comparison with the single shooting method because it matches many segments which decreases the error of fitting the nodes of the segments. However, there may be difficulties in finding initial guesses of $x(t_\kappa)$ for every segment. In practice, the last point in every segment is considered as a starting point in the following segment, except the first segment. The dimension of the space in which a direct search is performed, and a number of unknown variables are significant factors affecting performance of single and multiple shooting methods.

2.3 Derivative approximation method: B-spline Collocation Method

In the spline collocation method, the state trajectories are approximated by piecewise polynomial functions, where coefficients γ_j and a set of basis functions B_j

can be found to represent the solution globally for $j = 0, 1, ..., N_t$. Optimization problem (2.3) is formulated, so that its solutions are the coefficients γ_j for $j = 0, 1, ..., N_t$. Additionally, forcing the piecewise polynomial to satisfy the ordinary differential equations model at certain specified points, termed collocation points [3], determines constraints on the polynomial coefficients as well as the unknown parameters. There are many types of spline collocation methods exists, including the B-spline collocation, which may be linear, quadratic, cubic according to the given equation, orthogonal and others.

Numerical integration is used to represent the solution globally with a set of convenient basis functions $B_j(t), j = 1, 2, ..., N_t$ defined over the domain $[t_0, t_0 + T]$ with nodes t_j, $j = 0, 1, ..., N_t$. The design of these functions, termed B-spline functions, is to facilitate generalisation of polynomials for the approximation of a variety of functions. Subsequently, the collocation constraints at finite set of nodes, t_j, which reside on the Cartesian abscissa axis, should be satisfied by the said approximated solution.

We start to define the B-spline base functions for using in the solution procedure of Equation (2.1). Four additional nodes t_{-2},t_{-1},t_{N_t+1} and t_{N_t+2} are introduced such that $t_{-2} < t_{-1} < t_0$ and $t_{N_t} < t_{N_t+1} < t_{N_t+2}$. The B-splines $B_i(t)$, $i = -1, 0, 1, ..., N_t + 1$, at the nodes t_j are described over the interval $[t_0, t_0 + T]$ as

$$B_j(t) = \frac{1}{6h^3} \begin{cases} (t - t_{j-2})^3 & t_{j-2} < t \leq t_{j-1} \\ h^3 + 3h^2(t - t_{j-1}) + 3h(t - t_{j-1})^2 - 3(t - t_{j-1})^3 & t_{j-1} \leq t \leq t_j \\ h^3 + 3h^2(t_{j+1} - t) + 3h(t_{j+1} - t)^2 - 3(t_{j+1} - t)^3 & t_j \leq t \leq t_{j+1} \\ (t_{j+2} - t)^3 & t_{j+1} \leq t < t_{j+2} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where $h = T/N_t$ is the spacing between the nodes t_j.

A basis is generated for the functions defined over $[t_0, t_0 + T]$ by the set of B-splines

$$\{B_{-1}(t), B_0(t), B_1(t), \cdots, B_{N_t+1}(t)\}. \tag{2.5}$$

Thus, we can write an approximation solution $U(t)$ in terms of the cubic B-splines trial functions as:

$$U(t) = \sum_{j=-1}^{N_t+1} \gamma_j B_j(t). \tag{2.6}$$

From (2.4) we observe that for $t_{-1} < t < t_{N_t+2}$ in equation (2.6) reduces to

$$U(t) = \gamma_{j-1}B_{j-1}(t) + \gamma_j B_j(t) + \gamma_{j+1}B_{j+1}(t) + \gamma_{j+2}B_{j+2}(t), \quad j = 0, 1, ..., N_t. \tag{2.7}$$

Furthermore, evaluating at the nodes t_j, we have

$$\begin{aligned} U(t_j) &= \gamma_{j-1}B_{j-1}(t_j) + \gamma_j B_j(t_j) + \gamma_{j+1}B_{j+1}(t_j) \\ &= \tfrac{1}{6}\gamma_{j-1} + \tfrac{2}{3}\gamma_j + \tfrac{1}{6}\gamma_{j+1}, \quad j = 0, 1, ..., N_t. \end{aligned} \tag{2.8}$$

This can be written in matrix form as

$$\begin{pmatrix} U(t_{-1}) \\ U(t_0) \\ U(t_1) \\ \vdots \\ U(t_{N_t}) \\ U(t_{N_t+1}) \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 4 & 1 & 0 & \cdots & 0 \\ 1 & 4 & 1 & \cdots & 0 \\ 0 & 1 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 4 & 1 \\ 0 & \cdots & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} \gamma_{-1} \\ \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{N_t} \\ \gamma_{N_t+1} \end{pmatrix}, \tag{2.9}$$

where $U(t_{-1})$ and $U(t_{N_t+1})$ are auxiliary values used purely to give a well determined system, which determines the behaviour of U at the boundaries and can, for instance, be chosen to set the second derivative of U at the boundaries to 0 (a so-called natural cubic B-spline). The matrix in (2.9) is explicitly invertible. This shows that we can either parametrize the spline $U(t)$ using the coefficients γ_j, or using its value at the nodes points $U(t_0)$, $U(t_1)$, $\cdots$, $U(t_{N_t})$. This makes it straightforward to obtain a spline that interpolates a given set of points.

The B_j are piece-wise polynomials and hence using equation (2.6), we can easily differentiate U and substitute the derivative into equation (2.1), then have the following equation:

$$\sum_{j=-1}^{N_t+1} \gamma_j \frac{dB_j(t)}{dt} = f(t, \sum_{j=-1}^{N_t+1} \gamma_j B_j(t), \vartheta). \tag{2.10}$$

We can choose a finite number of so-called collocation points t_i, $i = 0, 1, \cdots, N$ in the interval $[t_0, t_0 + T]$ and require (2.10) to hold at these points.

Note that U, as solution of (2.2) depends on x_0, ϑ and t_0. If B_j are fixed for all $j = 1, 2, \cdots, N$, then the only way such dependence can be accounted for x_0, ϑ and t_0 through the coefficients γ_j. This implies that in Equation (2.10) γ_j depends on the values of x_0, ϑ and t_0.

The derivatives $\frac{dB_j(t)}{dt}$ can be easily computed for the given set of basis functions and collocation points, so that equation (2.10) represents a system of algebraic equations for $\gamma = \{\gamma_0, \gamma_1, \cdots, \gamma_{N_t}\}$, or for the values $U(t_0)$, $U(t_1)$, $\cdots$, $U(t_{N_t})$ via equation (2.9). We note that the way equation (2.10) was constructed is independent of the particular choice of spline, or indeed other basis functions.

Given an appropriate choice of splines and collocation points, the system (2.10) is well determined and can be solved to yield the coefficients of the approximate solution of the differential equation.

In the case of parameter fitting, the minimization of the error which has to be carried out simultaneously with the solution of equation (2.10) and is defined as follow:

$$E(\gamma) = \sum_{i=1}^{N} (y(t_i) - \sum_{j=-1}^{N_t+1} \gamma_j B_j(t_i))^2. \tag{2.11}$$

Hence, we have the following optimization problem:

$$\begin{aligned} &\min_{(\gamma)} E(\gamma) \\ &\text{subject to} \\ &\sum_{j=-1}^{N_t+1} \gamma_j \frac{dB_j(t)}{dt} = f(t, \sum_{j=-1}^{N_t+1} \gamma_j B_j(t), \vartheta). \end{aligned} \tag{2.12}$$

Necessary optimality tests for (2.12) follow Karush-Kuhn-Tucker (KKT) conditions [110]. In practice, however, solving (2.12) can be achieved via using e.g. the penalty functions approach [16] followed by the application of the Nelder–Mead algorithm for solving the resulted unconstrained optimization problem.

In order to solve the optimization problem initial guesses for γ and ϑ should be introduced.

2.4 Sensitivity Functions

Sensitivity functions [7] are used in mathematical modelling to investigate variations in the output of a model resulting from variations in the parameters and the initial conditions. For the purpose of quantifying the variation in the state variable $x(t)$ relating to changes in the parameter ϑ and the initial condition $x(t_0)$, as a natural consequence consideration must be given to sensitivity functions defined by the derivatives

$$\Upsilon_{\vartheta_i}(t) = \frac{\partial x}{\partial \vartheta_i}(t),\ i = 1, \cdots, p$$

and

$$\Upsilon_{x_{0l}}(t) = \frac{\partial x}{\partial x_{0l}}(t),\ l = 1, \cdots, n,$$

where x_{0l} is the $l-$th component of the initial condition x_0. If the function f of system (1.1) is sufficiently regular, the solution x is differentiable with respect to ϑ_i and x_{0l}, and therefore the sensitivity functions Υ_{ϑ_i} and Υ_{ϑ_i} are well defined.

From the sensitivity analysis theory for dynamical systems, one finds that $\Upsilon_1 = (\Upsilon_{\vartheta_1}, \cdots, \Upsilon_{\vartheta_p})$ is an $n \times p$ vector function that satisfies the ODE system

$$\begin{aligned} \dot{\Upsilon}_1(t) &= f_x(t, x(t), \vartheta)\Upsilon_1(t) + f_\vartheta(t, x(t), \vartheta) \\ \Upsilon_1(t_0) &= 0_{n\times p}. \end{aligned} \tag{2.13}$$

Here $f_x = \frac{\partial f}{\partial x}$, $f_\vartheta = \frac{\partial f}{\partial \vartheta}$ are the derivatives of f with respect to x and ϑ, respectively. In a similar manner, the sensitivity functions with respect to the components of the initial condition x_0 define an $n \times n$ vector function $\Upsilon_2 = (\Upsilon_{x_{01}}, \cdots, \Upsilon_{x_{0n}})$, which satisfies

$$\begin{aligned} \dot{\Upsilon}_2(t) &= f_x(t, x(t), \vartheta)\Upsilon_2(t) \\ \Upsilon_2(t_0) &= I_{n\times n}. \end{aligned} \tag{2.14}$$

The equations (2.13) and (2.14) are used in conjunction with equation (1.1) to numerically compute the sensitivities Υ_1 and Υ_2 for general cases when the function f is sufficiently complicated to prohibit a closed form solution by direct integration.

Determining Υ_1, Υ_2 requires solutions of (2.13), (2.14). In general, these are not available analytically but numerical integrates are used to derive Υ_1, Υ_2.

To define a cost function and perform an optimization problem, data y is

produced by evaluating the numerical solution of (2.1) with the actual value of parameters ϑ at t_i, $i = 0, 1, \cdots, N$. Hence, we consider an efficient optimization algorithm, such as Nelder–Mead and Newton methods, to minimize the cost function

$$\sum_{i=1}^{N}(y(t_i) - \hat{y}(t_i, \hat{\vartheta}))^2 \tag{2.15}$$

with respect to ϑ and using an initial guess ϑ_0 for the optimization algorithm, where $\hat{y}(t_i, \hat{\vartheta}) = x_1(t; t_0, x_0, \hat{\vartheta})$.

There are various uses for sensitivity information, such us in gradient-based optimization. Using such efficient method, gradient-based optimization, that accurately calculate sensitivities is extremely important. However, the calculation of gradients is often the most costly step in the optimization cycle, ; it is not efficient method for solving complex problems with a large number of variables and parameters, since it requires, however, to solve large number of equations, $(n \times n) + (n \times p)$. For example, if we consider the Hodgkin–Huxley system of 4 variables and 23 parameters, then this requires to solve $(4 \times 4) + (4 \times 23)$ equations in every step of estimation.

This method is not practicable in all real-time applications. In order to alter the unknown parameters a root finding method is used to set the sensitivity (derivative of the cost with respect to the parameters) to zero, allowing the determination of the effectiveness and rapidity of convergence for this method. For instance, in order to employ a steepest descent method [40] toward the minimum cost function it requires to determine an appropriate step size to alter the parameters with respect to computed gradients. Using Newton method [111] might achieve a more rapid convergence rate. Although it is a more robust method, the calculation of the Hessian (second vector derivative) of the cost function is required. So, the second derivative of the system regarding the parameters of interest is necessary. However, utilising the subsequent operation, as can be seen in the next derivation, would increase the computational cost of using such an algorithm significantly.

Several methods are similar in their behaviour to with either the method of steepest descent or first order methods of feasible directions, with slow convergence when faced with relatively ill-conditioned problems. Approaches for the

solution of non-differentiable problems need to be taken into consideration, and the purpose of this thesis is to consider a method the implementation of which is not complex, being quite broad in its scope, and reliant on a philosophy which diverges completely from those underlying methods already available. To be able to solve optimization problems with non-differentiable cost functional, particularly minimum problems, practical methods and approaches, such as the Nelder–Mead algorithm, are currently undergoing development. It is this algorithm, then, that will constitute a practical direct search method in all the examples of the works of this thesis.

2.5 Nelder-Mead Simplex algorithm

The Nelder-Mead is one of the derivatives-free known multidimensional unconstrained optimization methods, proposed by Nelder and Mead in 1965 [86]. Despite it has been proposed more than fifty years ago, it is still the method of choice in applications because it is easy to code and to use. Note though that its convergence is still an open question [46]. It belongs to a class of methods which do not require derivatives and which are often claimed to be robust for problems with discontinuities or where the function values are noisy [91]. This method can be tried for problems with non-smooth functions since no derivative information is required. It is used in cases when the function values are uncertain or subjected to noise [2]. The goal of the Nelder and Mead algorithm is to solve the following unconstrained optimization problem:

$$\min_{s} F(s),$$

where $s \in \mathbb{R}^p$, p is the number of unknown parameters and $F : \mathbb{R}^p \longrightarrow \mathbb{R}$ is the objective function.

This algorithm is based on the iterative update of a simplex made of $p+1$ points, $\{s_1, s_2, ..., s_{p+1}\}$, as explained in the following definition:

Definition 2.5.1 (Simplex). [11]: A simplex S in $\mathbb{R}^m$ is *the convex hull* of all $p+1$ vertices, that is, a simplex $S = \{s_J\}_{J=1}^{p+1}$ is defined by its $p+1$ vertices $s_J \in \mathbb{R}^p$ for $J = 1, 2, ..., p+1$.

Each point in the simplex is called a *vertex* and is associated with a function value $F(s)$ for all s point. If we consider the dimension p, the number of unknown parameters that need estimation, the $(p+1)$ vertices will arise. In each evolution the simplex may move, expand or shrink or contract. When all the vertices finally converge to a single point, the stopping criterion is satisfied. The number of variables of the system of equations affects the accuracy in finding the minimum point. It has been stipulated that there is relationship between the dimension of the algorithm and the mean number of the evaluations for convergence [61]. The major drawback of Nelder–Mead simplex method is that it does not define its moving directions well enough by simple geometrical movements in high dimensional cases. This explains why Nelder–Mead method is a simple and fast algorithm but is not stable in optimizing multi-dimensional problems. To illustrate this problem in this algorithm we will consider two different high dimensional parameter estimation problems in examples.

2.5.1 Statement of the algorithm

The Nelder–Mead algorithm uses four scalar parameters, the coefficient of reflection α, expansion β, contraction γ and shrinkage σ. When the expansion or contraction steps are performed, the shape and the size of the simplex is changed. These parameters should satisfy the following inequalities [11]:
$\alpha > 0,\ \beta > 1,\ \beta > \alpha,\ 0 < \gamma < 1$ and $0 < \sigma < 1$.
The standard choices for these parameters are:

$$\alpha = 1, \quad \beta = 2, \quad \gamma = 0.5 \quad and \quad \sigma = 0.5. \tag{2.16}$$

Fuchang Gao and Lixing Han [38] proposed to choose the expansion, contraction, and shrinkage parameters adaptively according to the problem dimension M to reduce the use of reflection steps for uniformly convex functions in high dimensional space. These parameters are chosen for large p in the following form:

$$\alpha = 1, \quad \beta = 1 + 2/p, \quad \gamma = 0.75 - 1/2p \quad and \quad \sigma = 1 - 1/p. \tag{2.17}$$

The new β can help prevent the simplex from bad distortion caused by expansion steps in high dimensions, using the new γ instead of 0.5 can alleviate the reduction of the simplex diameter when p is large and the purpose for using the new σ instead of 0.5 is to prevent the simplex diameter from sharp reduction when p is large. At the beginning of the first iteration we guess the initial point s_0 as a point in $\mathbb{R}^p$ and then find the other p simplex vertices by using the following parameters $P, Q > 0$ [11, 104]:

$$\begin{aligned} P_s &= (\sqrt{(p+1)} + p - 1)/(p\sqrt{2}) \\ Q_s &= (\sqrt{(p+1)} - 1)/(p\sqrt{2}). \end{aligned} \tag{2.18}$$

The first vertex of the simplex is the initial guess and the other vertex coordinates are subsequently defined depending on the length of all the edges of the simplex shape. Spendley, Hext and Himsworth [11] use a regular simplex by supposing that all the edges of the simplex shape have the same length $\ell > 0$ which keeps the shape regular. So the other vertices are defined by:

$$s_J(j) = s_0(j) + \begin{cases} \ell P_s & j = J - 1 \\ \ell Q_s & j \neq J - 1 \end{cases} \tag{2.19}$$

where $J = 2, 3, ..., p+1$ and $j = 1, 2, ..., p$. Then we have $p+1$ vertices $s_1, s_2, ..., s_{p+1}$ and every vertex is a vector of the order p. On the other hand, the axis-by-axis simplex explained in [11] does not depend on the parameters P_s, Q_s but it supposes that the edges of the simplex have different length and the other vertices are defined by:

$$s_J(j) = \begin{cases} s_0(j) + \ell_j & j = J - 1 \\ s_0(j) & j \neq J - 1 \end{cases} \tag{2.20}$$

for $J = 2, 3, ..., p + 1$ and $j = 1, 2, ..., p$. For example, for a problem in two-dimensional design space equations (2.18) and (2.19) or (2.20) lead to an equilateral triangle of side ℓ. After defining all the vertices $s_1, s_2, ..., s_{p+1}$ we can start using the algorithm where these vertices are changed by the following different steps:

1. **Order**: The vertices are sorted by increasing function values so that the best vertex has index 1 and the worst vertex has index $p + 1$ as follow: $F(s_1) \leq F(s_2) \leq \cdots \leq F(s_{p+1})$.

The s_1 vertex is called the best vertex because it is associated with the lowest function value $F(s_1)$ while the worst point s_{p+1} is associated with the highest function value $F(s_{p+1})$.

2. **Centroid**: Define s_c centroid of the p best vertices except the worst vertex s_{p+1} , which has the maximum value of F, by equation:

$$s_c = \frac{\sum_{J=1}^{p} s_J}{p}. \tag{2.21}$$

3. **Reflect**: We perform a reflection with respect to the worst vertex s_{p+1}, which creates the reflected point s_r from the equation:

$$s_r = s_c + \alpha(s_c - s_{p+1}), \tag{2.22}$$

and then compute the function value of the reflected point as $F_r = F(s_r)$. From that point, there are several possibilities, which are listed below by the next steps. Most steps try to replace the worst vertex s_{p+1} by a better point, which is computed depending on the context.

4. **Expand**: If $F_r \leq F_1$ calculate the expansion point s_e:

$$s_e = s_c + \beta(s_r - s_c), \tag{2.23}$$

and evaluate $F_e = F(s_e)$. If the expansion point allows to improve the function value, i.e $F_e \leq F_r$, the worst vertex (s_{p+1}) is rejected from the simplex and the expansion point s_e is accepted. This step is performed when the simplex is far away from the optimum vertex. Then the direction of descent is followed and the worst vertex is moved into that direction. While, if $F_r \geq F_1$ and $F_r < F_p$, then the reflection point s_r is accepted without needing to compute the expansion point.

5. **Contract**: If $F_r \geq F_p$, perform a contraction between s_c and the better of s_{p+1} and s_r. The contraction steps are performed when the simplex is near

the optimum, which allows to decrease the size of the simplex.

a- **Outside contraction**: If $F_r < F_{p+1}$, perform an outside contraction and calculate

$$s_{oc} = s_c + \gamma(s_r - s_c), \tag{2.24}$$

and evaluate $F_{oc} = F(s_{oc})$. If $F_{oc} \leq F_r$ we accept s_{oc} and replace s_{p+1} by s_r then perform contraction otherwise a shrink step (6) is performed where all vertices are moved toward the best vertex.

b- **Inside contraction**: The inside contraction vertex is calculated if $F_r \geq F_{p+1}$ by the equation:

$$s_{ic} = s_c + \gamma(s_{p+1} - s_c), \tag{2.25}$$

and evaluate $F_{ic} = F(s_{ic})$. If $F_{ic} \leq F_{p+1}$, we accept s_{ic} and replace s_{p+1} by s_{ic} otherwise a shrink step is performed.

6. **Shrink**: This step is confirmed after the contraction one and the shrink vertices is defined by:

$$s_J = s_1 + \sigma(s_J - s_1), \ J = 2, 3, ..., p + 1. \tag{2.26}$$

The unordered vertices of the simplex at the next iteration consist of s_1, s_2, s_3,$\cdots$, s_{p+1}.

Figure (2.1) presents the detailed situations in two dimension when each type of step occur. These figures been created in order to illustrate the following specific points of the algorithm. We suppose that s_w is the worst point and s_L and s_N are the lowest and the next worst (the second worst) vertices. The simplex steps are explained in Figure (2.2) which take search directions to reach the optimal vertex where the size of the simplex is increased by the contraction and shrinking steps.

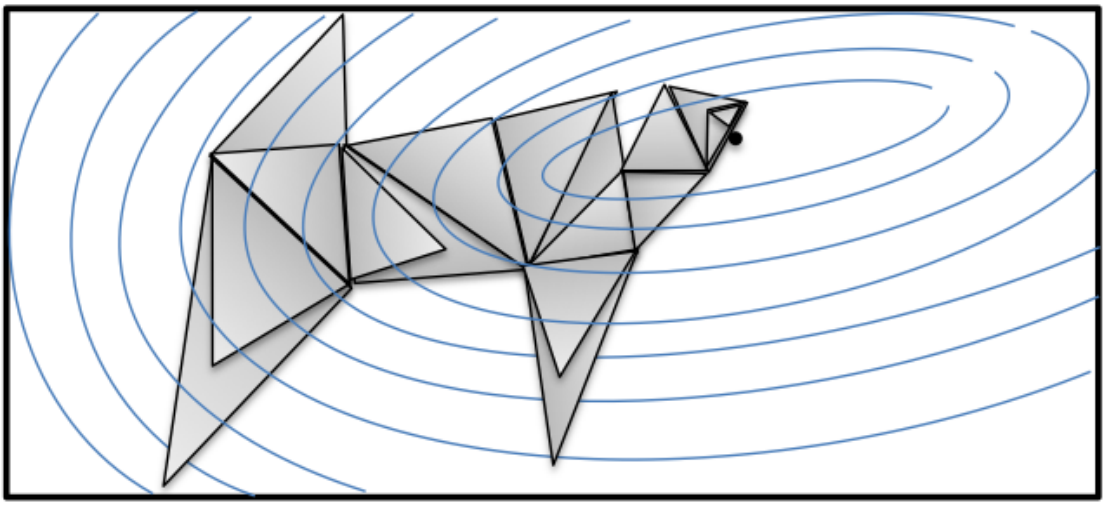


FIGURE 2.1: Nelder-Mead moves in two dimensions.

In every step the worst vertex should be replaced by the vertex which has the least evaluation and the algorithm should be repeated for a considerable number of iterations until the error is reduced. The standard error stopping criteria for the Nelder-Mead algorithm, proposed by Dennis and Woods in 1987 [25], is based on the size of the simplex. Thus, the shrinking steps in the algorithm work to satisfy this criterion [104]. The criterion is:

$$(1/\Delta)\max||s_J, s_1|| < \epsilon, \tag{2.27}$$

where the maximization is over all the points from 1 to p in the current simplex, $\Delta = \max(1, ||s_1||)$ and $\epsilon = 10^{-\varepsilon}$, $\varepsilon > 0$. Increasing the value of ε means that the simplex size decreases, with the distance between all the $p+1$ simplex vectors being shrunk. So the number of iterations depends on satisfying the Equation (2.27). The Nelder-Mead algorithm shrinks the simplex size automatically. Then the algorithm in the last iteration gives the best estimation for the parameters.

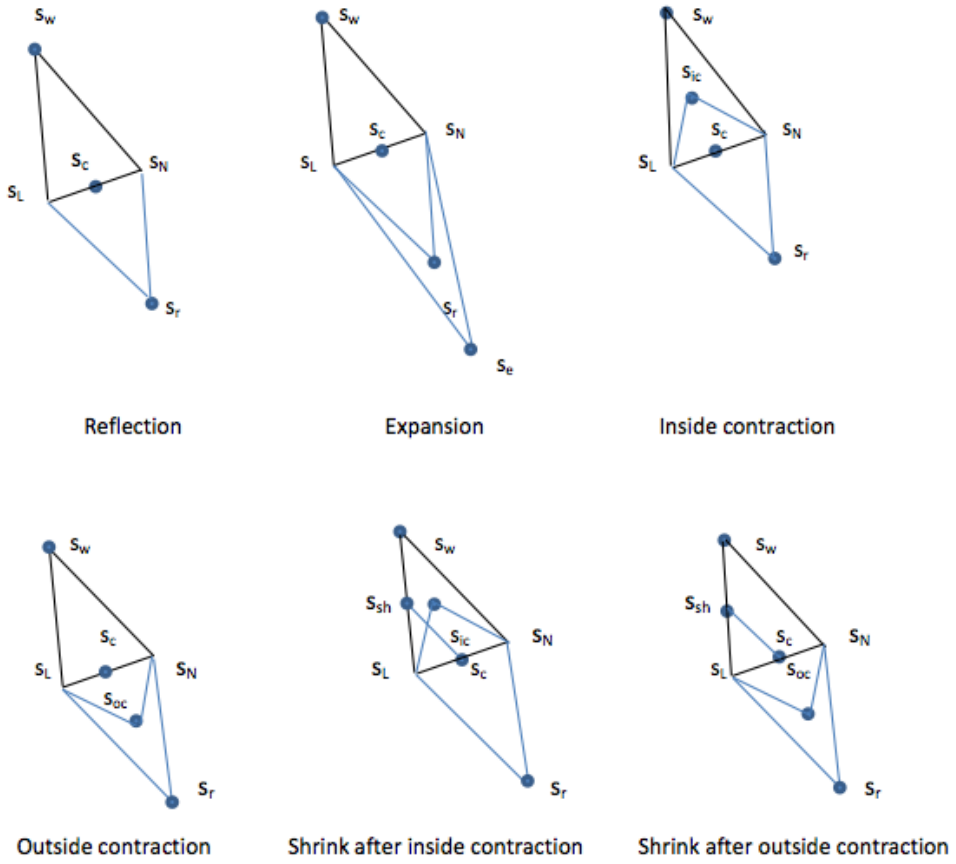


FIGURE 2.2: The six possible moves in the original Nelder-Mead algorithm are shown in two dimensions. The original simplex is surrounded by a black line, and its worst vertex s_{p+1} is labeled s_w. The point s_c is the average (centroid) of the two best vertices. The blue figures are Nelder-Mead simplices following refection, expansion, outside contraction, inside contraction, and shrink, respectively, where s_{ic} is the inside contraction vertex, s_r is the reflection Point, s_{oc} is the outside contraction vertex, s_e is the expansion vertex, s_N is the next worst vertex s_p, s_{sh} is the shrink vertex s and the best vertex (lowest vertex) is labelled s_L.

2.6 Case Studies

2.6.1 Hodgkin-Huxley model

Hodgkin–Huxley equations [17] are phenomenological model representing the current flow and voltage dynamics in neural membranes. The current is carried through the cell membrane due to the membrane capacitance or movement of ions through the conductances in parallel with the capacitance. The conductances are generally time varying and voltage dependent, and are modelled by ordinary

differential equations. The equations are presented as follows:

$$\begin{aligned}
\dot{v} &= (1/C_{mc})(I - (g_{Na}q_2^3 q_3(v - E_{Na}) + g_K q_1^4(v - E_K) + g_{leak}(v - E_{leak})) \\
\dot{q}_1 &= \alpha_1(v)(1 - q_1) - \beta_1(v)q_1 \\
\dot{q}_2 &= \alpha_2(v)(1 - q_2) - \beta_2(v)q_2 \\
\dot{q}_3 &= \alpha_3(v)(1 - q_3) - \beta_3(v)q_3,
\end{aligned} \tag{2.28}$$

where $v(t)$ is the measured voltage and the variables q_1, q_2 and q_3 dependent on both time and membrane potential, and represent the variability of the ion channels.

The functions $\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3 : \mathbb{R} \to \mathbb{R}$ are defined as:

$$\begin{aligned}
&\alpha_1 = 0.01(v + 50)/(1 - e^{(-0.1(v+50))}),\ \beta_1 = 0.125e^{(-(v+60)/80)} \\
&\alpha_2 = 0.1(v + 35)/(1 - e^{(-0.1(v+35))}),\ \beta_2 = 4e^{(-0.0556(v+60))} \\
&\alpha_3 = 0.07e^{(-0.05(v+60))},\ \beta_3 = 1/(1 + e^{(-0.1(v+30))}),
\end{aligned} \tag{2.29}$$

and the initial value of the variables q_1, q_2 and q_3 are defined by the following equations [98]:

$$\begin{aligned}
q_{1_0} &= \alpha_1(v_0)/(\alpha_1(v_0) + \beta_1(v_0)) \\
q_{2_0} &= \alpha_2(v_0)/(\alpha_2(v_0) + \beta_2(v_0)) \\
q_{3_0} &= \alpha_3(v_0)/(\alpha_3(v_0) + \beta_3(v_0)).
\end{aligned} \tag{2.30}$$

The parameters E_{Na}, E_K, E_{Leak} are the Nernst potentials for the calcium, potassium and leakage channels respectively, while g_{Na}, g_K, g_{Leak} represent the conductances for the calcium, potassium and leakage channels, respectively.

The system (2.28) was solved using a Runge-Kutta scheme of order 4 with an integration time step 0.01 by Siciliano in [98]. It was furthermore observed that the Runge-Kutta method gives the most accurate results when its solution compared with the exact solution after setting the sodium and the potassium conductances equal to zero for a step size of 0.04. This was performed because the model does not have an exact solution for other values of the conductances. The values $\vartheta_1 = g_{Na}$, $\vartheta_2 = E_{Na}$, $\vartheta_3 = E_K$, $\vartheta_4 = E_{leak}$ can, in certain circumstances, be regarded as typical but ultimately should be regarded as unknown parameters requiring observation since they are the result of a curve-fitting process. In our computer simulations, we will take capacitance $C_{mc} = 0.01$ and $I = 0.1$ and

assume that the states v, q_1, q_2 and q_3 are bounded in forward time.

For the purposes of illustration at least, we take g_{Na}, E_{Na}, E_K, $\vartheta_4 = E_{leak} \in \mathbb{R}$ as unknown values. The values of parameters $g_K, g_{Leak} \in \mathbb{R}$, however, may vary substantially from one cell to another, they depend upon the density of ion channels in a patch of the membrane. Hence, in order to model the dynamics of individual cells, we need to be able to obtain the unknown parameters from data.

In the simulation, we set the unknown and known parameters of the Hodgkin–Huxley model (2.28) to the following: $g_{Na} = 1.2$, $E_{Na} = 55.17$, $E_K = -72.14$, $E_{leak} = -49.42$, $g_K = 0.36$, $g_{Leak} = 0.003$.

We used the Runge–Kutta 4th order algorithm to find numerical solutions of the systems, and the Nelder–Mead algorithm as an optimization routine. A starting simplex was constructed around an initial parameter estimate V_0 by

$$\vartheta_0 = \vartheta_i + \varepsilon\iota_i, \tag{2.31}$$

and an initial value for x_0 by

$$v_0 = v(t_0) + \epsilon\iota_j, \tag{2.32}$$

where ε, ϵ are constants, and ι_i and ι_j are the unit vectors in the $i-$th, $j-$th coordinate directions, respectively. The initial value v_0 combined the parameters, *i.e.* (v_0, ϑ). By using this initial point the other variables q_1, q_2 and q_3 were calculated by Equation (2.30). The values of t were chosen from the 0.01-spaced points in $[0, 25]$ which shows two uniform spikes for the first variable V of the system. Table 2.1 shows the estimated values of the unknown parameters and v_0, and Figure 2.4 shows the fitting curves. The 2nd column of all the shown tables of this section represents the true values of the parameters and v_0.

Parameters	Estimated Values	Actual Values	Parameter Estimation Error
v_0	-60.00000	-60	5.86907e-6
g_{Na}	1.19999	1.2	6.95908e-7
E_{Na}	55.17004	55.17	3.56341e-5
E_K	-72.13999	-72.14	1.74518e-6
E_{leak}	-49.41998	-49.42	2.01932e-5

TABLE 2.1: Estimated values of the initial condition v_0 and the unknown parameters ϑ_1, ϑ_2, ϑ_3 obtained using a Nelder–Mead–based single shooting method starting with (v_0, ϑ_0); we put $\varepsilon = 3$ and $\epsilon = -40$. We used one node point and 2501 data points. The least square error is $3.503357e-9$ and the simplex size is $6.695736e-11$ obtained over 651 iterations.

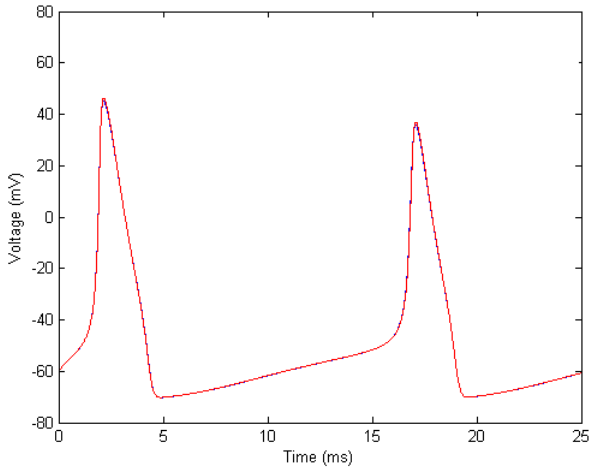


FIGURE 2.3: Voltage Change curves (Runge-Kutta method) at the nominal value of parameters ϑ_1, ϑ_2, ϑ_3 and ϑ_4 (blue curve), and at the estimates resulting from the single shooting method (red curve).

To apply multiple shooting method the time interval $[0, 25]$ was divided into 5 subintervals where the first point of 501 points in each interval represents the shooting node. Table 2.2 shows the estimated values of v_0, ϑ_1, ϑ_2, ϑ_3, ϑ_4. Figure 2.4 and Table 2.2 show the results of applying multiple shooting method using Nelder-Mead algorithm.

Parameters	Estimated Values	Actual Values	Parameter Estimation Error
v_0	-60.00000	-60	1.65796e-10
g_{Na}	1.20000	1.2	4.41639e-7
E_{Na}	55.16997	55.17	2.60501e-6
E_K	-72.14000	-72.14	6.07513e-6
E_{leak}	-49.41996	-49.42	3.43414e-5

TABLE 2.2: Estimated values of the initial condition v_0 and the unknown parameters ϑ_1, ϑ_2, ϑ_3 obtained using a Nelder–Mead–based multiple shooting method starting with (v_0, ϑ_0); we put $\varepsilon = 1$ and $\epsilon = 1$. We used 5 node points and 501 points in every segment. The least square value is $2.952988e-10$ and the simplex size is $8.885229e-11$ obtained over 321 iterations.

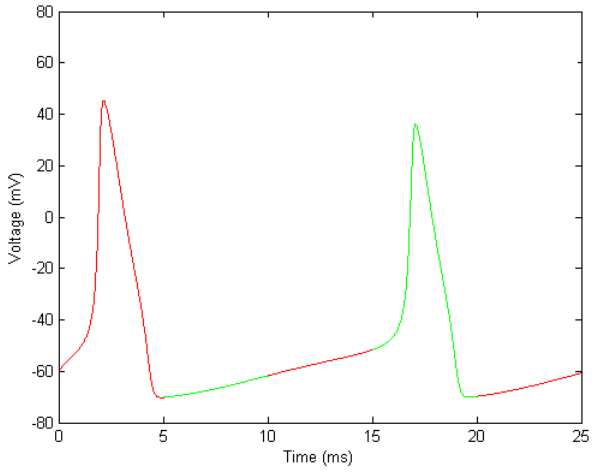


FIGURE 2.4: Voltage Change curves of Hodgkin Hukley Model (Runge–Kutta method) at the nominal value of parameters ϑ_1, ϑ_2, ϑ_3 and ϑ_4 (blue curve), and at the estimates resulting from the multiple shooting method (red and grean curves).

The above methods were implemented in Matlab2015. One choice of initial values of parameters and initial condition was applied to the 2501 time point dataset. Convergence to the global minimum occurred and these successful runs both started with the same initial values of parameters and v_0. The duration of the runs was consistently high, almost always taking more than half hour to converge. In order to provide the best chance of success, the test was performed with a large dataset; a reduction in the dataset would render the parameter space more complex and make convergence to the optimal solution harder.

We conclude that shooting methods using Nelder–Mead can give good estimates for parameters, but requires a lot of time tuning the minimization algorithm

for success. These conclusions will apply to any global minimization method that relies on single shooting to determine the error function. Multiple shooting can help to regularize the parameter space and generally provides a more robust algorithm.

2.6.2 ECG System

A single norm cycle is the ECG represents atrial depolarization or repolarization and ventricular depolarization or repolarization which occurs with every heartbeat. This can be approximately associated with the peaks and troughs of the ECG waveform labeled P, Q, R, S and T as shown in Figure 2.5 adopted from [79]. P, Q, R, S and T are waves in every cycle of the ECG which represent atrial depolarization, the first wave of the ventricular depolarization QRS, the initial positive deflection, the negative deflection following the R wave and ventricular repolarization, respectively [79].

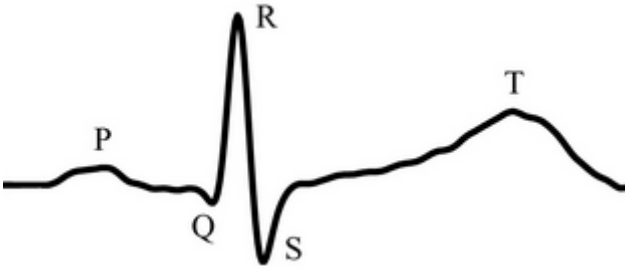


FIGURE 2.5: Morphology of a mean PQRST-complex of electrocardiogram signals (ECG).

The *horizontal axis* and *vertical axis* represent the time (ms) and voltage (mV). The dynamical model of ECG is based on three ordinary differential equations which is capable of generating realistic synthetic electrocardiogram signals. The dynamical equations are given by [79]:

$$\begin{aligned} \dot{x}^* &= \upsilon x^* - \omega y^* \\ \dot{y}^* &= \upsilon y^* + \omega x^* \\ \dot{z} &= -\sum_{\tau \in P,Q,R,S,T} a_\tau \Delta\theta_\tau e^{(} - \tfrac{\Delta\theta_\tau^2}{2b_\tau^2}) - (z - z_0), \end{aligned} \tag{2.33}$$

where $\upsilon = 1 - \sqrt{x^{*2} + y^{*2}}$, $\Delta\theta_\tau = (\theta - \theta_\tau) \mathrm{mod} 2\pi$, $\theta = \mathrm{atan2}(x^*, y^*)$ and ω is the angular velocity of the trajectory as it moves around the limit cycle. The (x^*, y^*, z) ECG trajectory generated by the model is illustrated in Figure 2.6 [79]. This figure illustrates the process of the waves P, Q, R, S, T around one unit circle in the (x^*, y^*) plane. These waves are placed at fixed angles $\theta_P, \theta_Q, \theta_R, \theta_S$ and θ_T along the unit circle. The small circles in the figure show the position of the P, Q, R, S, T which are represented by the values a_τ and b_τ in the model (2.33). We have the values of these parameters as shown in Table 2.3 where the position of the R-peak specifics the time and the angles θ. The variables x^* and y^* depend on which is called respiratory frequency f_2, the angular velocity ω and the time t as in the equations [79]:

$$x^* = \cos(tw + f_2)$$
$$y^* = \sin(tw + f_2).$$

The baseline value z_0 in (2.33) is coupled with the respiratory frequency f_2 using $z_0(t) = A \sin(2\pi f_2 t)$,
where $A = 0.15\ mV$.
If the respiratory frequency f_2 equals to 0, the initial value $z_0 = 0$. Hence the system is transformed into the system of one non-linear ordinary differential equation:

$$\dot{z} = -\sum_{\tau \in P,Q,R,S,T} a_\tau \Delta\theta_\tau \exp(-\frac{\Delta\theta_\tau^2}{2b_\tau^2}) - z. \tag{2.34}$$

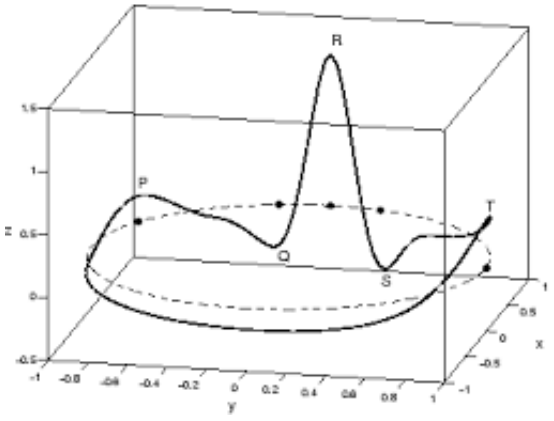


FIGURE 2.6: (x^*, y^*, z) trajectory generated by the dynamical model (2.33) in 3 dimensional space.

Parameter	P	Q	R	S	T
$time$	-0.2	-0.05	0	0.05	0.3
a_τ	1.2	-5.0	30.0	-7.5	0.75
b_τ	0.25	0.1	0.1	0.1	0.4
θ_τ	-1/3 π	-1/12 π	0	1/12 π	1/2 π

TABLE 2.3: Parameters of the ECG model.

The parameterized system of ECG is represented as follows:

$$\begin{aligned} \dot{z} &= -\sum_{\tau\in P,Q,R,S,T} a_\tau \Delta\theta_\tau \exp(-\tfrac{\Delta\theta_\tau^2}{2b_\tau^2}) - z \\ \dot{a}_\tau &= 0 \\ \dot{b}_\tau &= 0 \\ \dot{\theta}_\tau &= 0 \\ y &= z, \end{aligned} \tag{2.35}$$

where $\hat{a}_\tau = a_{\tau 0}$, $\hat{b}_\tau = b_{\tau 0}$ and $\hat{\theta}_\tau = \theta_{\tau 0}$ for all $\tau = 1, 2, 3, 4, 5$.

The methods applied in this example use some common steps for minimizing the least square error of the optimization problem (2.3) subject to the system (2.35).

Multiple shooting method was performed for 6 segments where every segment has 44 points. We found the initial values of the parameters by Equation (2.31), and the initial value z_0 as explained in Equation (2.32). The estimations of the parameters and z_0 are shown in Table 2.4, and Figure 2.7 represents the 6 segments of the the voltage by red and green colours.

Parameters	Estimated Values	Actual Values	Parameter Estimation Error
z_0	-0.00617	-0.0063	0.00013
a_1	1.26989	1.2	0.06989
a_2	-5.31679	-5.0	0.31679
a_3	30.22360	30.0	0.22360
a_4	-6.08688	-7.5	1.41312
a_5	0.71935	0.75	0.03065
b_1	0.25105	0.25	0.00105
b_2	0.09876	0.1	0.00124
b_3	0.10212	0.1	0.00212
b_4	0.11128	0.1	0.01128
b_5	0.41820	0.4	0.01820
θ_1	-1.29476	-1.2217	0.07306
θ_2	-0.30044	-0.2618	0.03864
θ_3	-0.00981	0	0.00981
θ_4	0.26204	0.2618	0.00024
θ_5	1.81601	1.7453	0.07071

TABLE 2.4: Estimated values of the initial condition z_0 and the unknown parameters a_τ, b_τ and θ_τ for $\tau = 1, 2, 3, 4, 5$ obtained using a Nelder–Mead–based multiple shooting method starting with (z_0, ϑ_0); we put different values of ε for parameters and $\epsilon = 0.5$. We used 6 node points and 44 points in every segment. The least square error is $4.701174e-9$ and the simplex size is $9.806092e-6$ obtained over 4475 iterations.

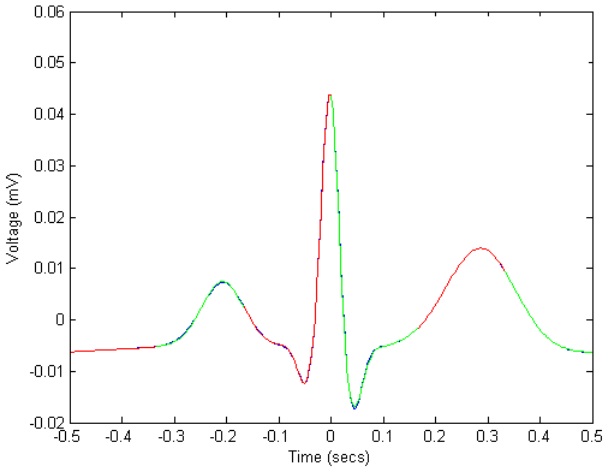


FIGURE 2.7: Voltage change curve of ECG Model (by Runge–Kutta method) at the nominal value of parameters (blue curve) and at the estimates resulting from the multiple shooting method (red curve).

On the other hand, we applied the cubic B-splines for the same system. We considered a partition of $[-0.5, 0.5]$ equally divided by nodes t_j into N subinterval

$[t_j, t_{j+1}]$, where $j = 0, 1, \cdots, N$. When the dataset is relatively large ($N = 258$ time points), we tried to obtain parameter estimates using N splines.

The approximation $U(t)$ based on the collocation approach, which can be expressed as:

$$U(t) = \sum_{j=-1}^{N+1} \gamma_j B_j(t). \tag{2.36}$$

The approximated solution can be substituted in the model to have the following equation:

$$\sum_{j=-1}^{N+1} \gamma_j B_j'(t) = -\sum_{\tau \in P,Q,R,S,T} a_\tau \Delta\theta_\tau \exp(-\tfrac{\Delta\theta_\tau^2}{2b_\tau^2}) - \sum_{j=-1}^{N+1} \gamma_j B_j(t). \tag{2.37}$$

To obtain the approximations of the solutions, the values of B_j and its derivative at the nodes are required. Since the values vanish at all other nodes, they are omitted from Table 2.5. Equation (2.38) explains the first derivative of B_j.

$$B_j' = \begin{cases} \frac{1}{2h^3}(t - t_{j-2})^2 & t_{j-2} < t \le t_{j-1} \\ \frac{1}{2h} + \frac{1}{h^2}(t - t_{j-1}) - 3\frac{1}{2h^3}(t - t_{j-1})^2 & t_{j-1} \le t \le t_j \\ -\frac{1}{2h} - \frac{1}{h^2}(t_{j+1} - t) + 3\frac{1}{2h^3}(t - t_{j-1})^2 & t_j \le t \le t_{j+1} \\ -\frac{1}{2h^3}(t - t_{j-2})^2 & t_{j+1} \le t < t_{j+2} \\ 0 & otherwise \end{cases} \tag{2.38}$$

	t_{j-2}	t_{j-1}	t_j	t_{j+1}	t_{j+2}
B_j	0	1/6	4/6	1/6	0
B_j'	0	$1/2h$	0	$-1/2h$	0

TABLE 2.5: Values of B_j and B_j'.

Then we have :

$$U(t_j) = \tfrac{1}{6}\gamma_{j-1} + \tfrac{2}{3}\gamma_j + \tfrac{1}{6}\gamma_{j+1}, \quad j = 0, 1, ..., N, \tag{2.39}$$

and

$$U'(t_j) = \frac{1}{2h}\gamma_{j-1} - \frac{1}{2h}\gamma_{j+1} = 129\gamma_{j-1} - 129\gamma_{j+1}, \quad j = 0, 1, ..., N. \tag{2.40}$$

Hence the ECG system will be presented as follows:

$$129\gamma_{j-1} - 129\gamma_{j+1} = -\sum_{\tau\in P,Q,R,S,T} a_\tau \Delta\theta_\tau \exp(-\tfrac{\Delta\theta_\tau^2}{2b_\tau^2}) - \tfrac{1}{6}\gamma_{j-1} - \tfrac{2}{3}\gamma_j - \tfrac{1}{6}\gamma_{j+1}. \tag{2.41}$$

We have 259×259 the system of linear equations and 261 unknown variables $\gamma_j,\ j = -1, 0, ..., 259$. Then we need to find the boundary conditions:

$$\begin{aligned} U(t_0) &= \tfrac{1}{6}\gamma_{-1} + \tfrac{2}{3}\gamma_0 + \tfrac{1}{6}\gamma_1 = -0.0063 \\ U(t_N) &= \tfrac{1}{6}\gamma_{N-1} + \tfrac{2}{3}\gamma_N + \tfrac{1}{6}\gamma_{N+1} = -0.0063 \end{aligned} \tag{2.42}$$

to have the matrix form of linear equations system (2.9) which is so easily to solve by supposing that the approximate spline polynomial (2.36) satisfies the equation (2.34) and then the coefficients $\gamma_j,\ j = -1, 0, \cdots, 259$ can be calculated.

The unknown parameters a_τ, b_τ and θ_τ for $\tau = 1, 2, 3, 4, 5$ and the coefficients γ are estimate by using Nelder-Mead method. The experimental data of the ECG system has 258 points defined on the domain $[-0.5, 0.5]$ with step size $h = 1/258$. The spline curve is represented in Figure 2.8 and the estimated values of the unknown parameters are obtained in Table 2.6.

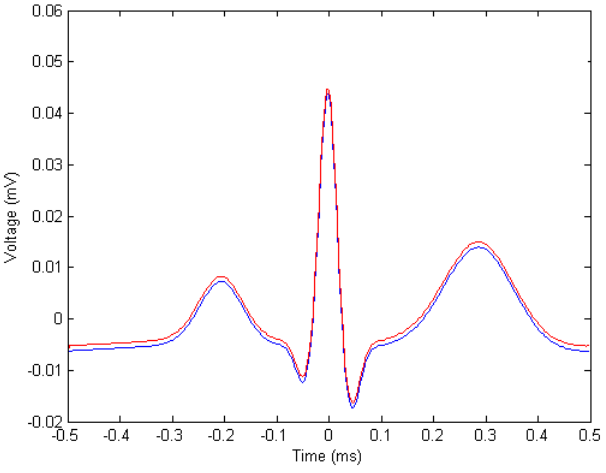


FIGURE 2.8: Voltage change for ECG model (by Runge–Kutta method) at the nominal values of the parameters a_τ, b_τ and θ_τ for $\tau = 1, 2, 3, 4, 5$ and the coefficients $\gamma_j,\ j = -1, 0, ..., 259$ (blue curve), and at the estimates resulting from the spline collocation method (red curve).

We noticed that spline collocation method based on Nelder-Mead algorithm spent more than one hour to estimate the parameters. This is not significantly

Parameters	Estimated Values	Actual Values	Parameter Estimation Error
a_1	1.1901	1.2	0.0099
a_2	-4.7099	-5.0	0.2901
a_3	31.0901	30.0	1.0901
a_4	-7.6099	-7.5	0.1099
a_5	0.8301	0.012247	0.8179
b_1	0.3451	0.25	0.0951
b_2	0.2001	0.1	0.1001
b_3	0.1911	0.1	0.0911
b_4	0.2101	0.1	0.1101
b_5	0.5201	0.4	0.1201
θ_1	-0.9099	-1.22173	0.3118
θ_2	-0.0599	-0.261799	0.2019
θ_3	0.1901	0	0.1901
θ_4	0.2401	0.261799	0.0217
θ_5	2.2972	1.74533	0.5519

TABLE 2.6: Estimated values of the initial condition z_0 and the unknown parameters a_τ, b_τ and θ_τ for $\tau = 1, 2, 3, 4, 5$ obtained using a Nelder–Mead–based shooting method starting with (z_0, ϑ_0); we put $\varepsilon = 0.001$ and $\epsilon = 0.5$. We used 6 node points and 44 points in every segment. The least square error is $\sum_{k=1}^{259} \| \sum_{j=1}^{261} \gamma_j B_j(t_k^*) - y(t_k^*) \|^2 = 0.208098$.

faster than using multiple shooting method, but the speed could be improved considerably as the amount of data or the number of splines is reduced. Notice that even shooting method was applied with very small amounts of segments, the estimates still reasonably more accurate than using spline collocation method as shown clearly in the figures 2.7 and 2.8.

Overviews and limitations: The classical and most commonly used methods, as reviewed, offer convenient ways to find state and parameters of a broad range of problems. They do, however, have shortcomings and limitations. Numerical methods require a considerable number of iterations in order to approach solutions. The solution usually is not exact, and therefore they provide approximated solutions which are required to use to estimate the unknown state and parameters of systems.

The more serious limitation in spline collocation method is that the polynomial degree is only one less than the number of collocation points. In a practical situation we may be given several thousand points which would require a polynomial curve of an impossibly high degree. To compute a point on a curve of degree

N_t requires a number of multiplications and additions that are at best proportional to N_t. If for example $N_t = 1000$, computer manipulations like plotting and interactive editing of the curve would be much too slow to be practical, despite the speed of computers developed over time. This was practically experienced in second example; we supposed to handle number of splines as the same number of collocation points.

In general, shooting and spline collocation methods cannot always be applied, and their numerical results cannot always be trusted. The disadvantage in the linear case arise because the IVPs integrated in the process could be unstable, even when the BVP is well-conditioned and hence the method is unstable. Moreover, if the initial values of the state variables are far away from the actual values, the methods using any direct search algorithm may fail or cost time to obtain accurate estimations. Notwithstanding the plausibility of numerical integration of (2.1) in algorithms for state and parameter estimation, this operation is an inherently sequential process. The amount of time needed for direct numerical integration of these systems over a grid of N points is at least $O(N)$. This constrains computational scalability of the problem, and as a result it imposes limitations on the time required to derive a solution [107]. Therefore, we need to construct an integral formula that allows for fast and efficient numerical evaluation of solutions of ODE systems which may invoke parallel computational frame to speed up the evaluation or solving parameterized problems.

In order to overcome this limitation we should propose to cast the inverse problem in an alternative integral form in which the model output is defined as a combination of indefinite integrals with known, explicit and computable kernels, possibly dependent on x_0, ϑ, and explicit functions of x_0, ϑ. The advantage of such integral formulations is that their computations are scalable and can be performed using conventional prefix sum algorithms of which the execution time is of order $O(\log_a(N))$, $a = 2, 3, \cdots$ [107]. The latter option compares favourably with $O(N)$.

Chapter 3

Explicit Parameter-dependent Representations of Periodic Solutions for a Class of Nonlinear Systems for Parameter Estimation

3.1 Identifiability of Mathematical Models

To determine parameters from measured projections/mappings, an assessment of identifiability has to be carried out first. Sometimes, the knowledge or the lack of identifiability of some parameters can lead to or even require a transformation yielding a globally identifiable model. Informally, the question of identifiability of a system can be posed as that of the uniqueness of the solution parametrization given initial conditions, model structure, and measured data.

To formally introduce notions of the identifiability, consider the following system of ODEs:

$$\dot{x} = f(t, x, \vartheta);\ x(t_0) = x_0 \in \mathbb{R}^n \tag{3.1}$$

where $x = \text{col}(x_1, \cdots, x_n)$ is the system's state vector, $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is the state evolution function which is assumed to be at least a piece-wise continuous function so that solution exists. The initial state x_0 can be known or unknown; in the latter case it must be estimated from experimental data as well. Let $[t_0, t_0 + T]$ be an interval on which the solution $x(t; t_0, x_0, \vartheta)$ of (3.1) is defined. Let us further suppose that the system's state, $x(t; t_0, x_0, \vartheta)$, is not accessible for direct

observation at any $t \in [t_0, t_0 + T]$. One can, however, observe the values

$$y(t; t_0, x_0, \vartheta) = h(t, x(t; t_0, x_0, \vartheta)) \tag{3.2}$$

for every $t \in [t_0, t_0 + T]$. For the sake of simplicity we will refer to these values as $y(t)$ or y if the other arguments, t, x_0, ϑ, are clear from the context. Let the problem be to find $\vartheta' \in \mathbb{R}^p$ and $x_0' \in \mathbb{R}^n$

$$y(t; t_0, x_0, \vartheta) = y(t; t_0, x_0', \vartheta'). \tag{3.3}$$

We note that y is a function of state x, which in turn, is a function of t, x_0, t_0 and parameters ϑ. If t_0 is fixed then each pair x_0, ϑ corresponds to an observed trajectory $y(t; t_0, x_0, \vartheta)$.

Deriving x, y governed by the initial value problem (3.1) is the so called forward problem. It concerns predicting and simulating the values of measurements or output variables for a given system with given parameters and initial conditions. Solving (3.2) is the inverse problem. In this problem one uses the measurements of some state or output variables to estimate the true values of parameters determining given measurements and, implicitly, characterizing the system. This is not a trivial problem especially for nonlinear ODE models without closed-form solutions.

Existence of unique solutions to the inverse problem is closely related to the notion of identifiability. Fundamental definitions of identifiability and unidentifiability of ODE systems are introduced in [26]; here we adopt these definitions for system (3.1) as follows:

Definition (Identifiability)

Consider system (3.1). Let $x(t; t_0, x_0, \vartheta)$ and $x(t; t_0, x_0', \vartheta')$ be two solutions of (3.1) corresponding to different parameter values and initial conditions defined over the time interval $[t_0, t_0 + T]$, then

1. the parameter ϑ of system (3.1) is said to be unidentifiable on $[t_0, t_0 + T]$ if for some given (x_0, ϑ) there exists an infinite number of solutions of (3.3) for ϑ.

2. the parameter ϑ of system (3.1) is said to be identifiable on $[t_0, t_0+T]$ if for every (x_0, ϑ) there exists a finite number of solutions of (3.3) for ϑ.

3. the parameter ϑ of system (3.1) is said to be uniquely identifiable on $[t_0, t_0+T]$ if for every (x_0, ϑ) there exists a unique solution of (3.3) for ϑ.

Similar notions have been introduced in [80]; in terms of global and local identifiability let us have the below remarks:

Remark 1: System (3.1) is globally identifiable if for any ϑ, ϑ' in $\mathbb{R}^p$, $y(t; t_0, x_0, \vartheta) = y(t; t_0, x_0', \vartheta')$ for all $t \in [t_0, t_0+T]$ if and only if $\vartheta = \vartheta'$.

Remark 2: System (3.1) is locally identifiable if there exists an open neighbourhood W of (x_0, ϑ) in $\mathbb{R}^{n+p}$ such that $y(t; t_0, x_0, \vartheta) = y(t; t_0, x_0', \vartheta')$ for (x_0, ϑ), (x_0', ϑ') in $\mathbb{R}^{n+p}$ if and only if $\vartheta = \vartheta'$.

Note that, in general, solving (3.3) for ϑ is not independent on x_0. As the examples below show that ϑ and x_0 in the inverse problem are related and linked.

Inferring true values of ϑ from output observations, $y(t; t_0, x_0, \vartheta)$, is not always possible, even if the system is linearly parametrized and no unmodeled dynamics are presented [108]. Let us explain this with two simple examples which are considered by Ivan *et al.* in [108]. The first example considers the following system of ODEs:

$$\begin{aligned} &\dot{x} = Ax + \begin{pmatrix} 1 \\ 1 \end{pmatrix}\theta + \begin{pmatrix} 1 \\ 0 \end{pmatrix}\lambda, A = \begin{pmatrix} -a_1 & 1 \\ -a_2 & 0 \end{pmatrix}, \quad a_1, a_2 \in \mathbb{R}_{>0} \\ &y = x_1. \end{aligned} \tag{3.4}$$

Let $x(t; t_0, x_0, (\theta, \lambda))$ and $x(t; t_0, x_0', (\theta', \lambda'))$ be two solutions of (3.1) corresponding to different parameter values and initial conditions, and let $e(t) = \mathrm{col}(e_1(t), e_2(t)) = x(t; t_0, x_0, (\theta, \lambda)) - x(t; t_0, x_0', (\theta', \lambda'))$. Picking $e_2(t_0) = -\theta + \theta'$, $e_1(t_0) = 0$ ensures that $e_1(t) = 0$ for all $t \geq t_0$. Another, albeit nonlinearly parameterized, example is

$$\begin{aligned} &\dot{x} = -x + \theta + [\sin^2(\lambda + \theta) + x^2 + 1]^{-1} \\ &y = x \end{aligned} \tag{3.5}$$

In this case $e(t) = x(t; t_0, x_0, (\theta, \lambda)) - x(t; t_0, x_0', (\theta', \lambda')) = 0$ for all $t \geq t_0$ if $\lambda' = \lambda + k\pi$, $k \in \mathbb{Z}$, $\theta' = \theta$.

The difficulty in estimating the parameters in a quantitative mathematical model is not so much how to compute them, but more how to assess the quality of the obtained parameters. Identifiability arises in conjunction with the question of observability, when the notion of states may be augmented to include both the actual state variables of the dynamic system and its parameters. This results in the formulation of a nonlinear augmented system even though the dynamic equations of motion of the original system might be linear. The consequence of a system being observable is that different states can be distinguished on the basis of measurements. Consequently, when the system is not observable, a system identification method cannot be expected to return the true value of the states and parameters.

In what follows we will present non local identifiability conditions for (1.1) and provide solution parametrizations that are a) suitable for scalable implementations and b) and unique in the sense that $y(t; t_0, x_0, \vartheta) = \hat{y}(t; t_0, x_0', \vartheta') \Leftrightarrow \vartheta = \vartheta'$, $x_0 = x_0'$.

3.2 Problem Formulation

3.2.1 System definition

Consider the following class of nonlinear systems

$$\begin{aligned} \dot{x} &= F(y,t)x + \Psi(y,t)\theta + g(y,\lambda,t) \\ y(t) &= C_1^T x;\ x(t_0) = x_0, \end{aligned} \tag{3.6}$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$ are the state and the output of the system, respectively, $F(y,t) \in \mathbb{R}^{n\times n}$ is a known matrix dependent on y and t; $\lambda \in \Omega_\lambda$, $\Omega_\lambda \subset \mathbb{R}^p$, $\theta \in \Omega_\theta$, $\Omega_\theta \subset \mathbb{R}^m$ are parameters, and $C_1 \in \mathbb{R}^n$: $C_1 = \mathrm{col}(1, 0, \cdots, 0)$. Other technical assumptions are detailed in Assumption 1 below.

Assumption 1. The following properties hold for (3.6):

1. the solution of (3.6) is defined for all $t \geq t_0$, and it is T-periodic, $T > 0$;

2. the function F is continuous, bounded, and $F(y(\cdot), \cdot)$ is T-periodic;

3. exact values of parameters λ and θ are unknown;

4. the values of $y(t)$ for $t \in [t_0, t_0 + T]$ are available and known;

5. the function $\Psi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{n \times m}$ is such that $\Psi(y(\cdot), \cdot)$ is T-periodic and is in $L_\infty[t_0, \infty) \cap \mathcal{C}^0$;

6. the function $g : \mathbb{R} \times \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}^n$ is such that $g(y(\cdot), \lambda, \cdot)$ is T-periodic and is in $L_\infty[t_0, \infty) \cap \mathcal{C}^1$ for all $\lambda \in \Omega_\lambda$;

7. the observability Gramian matrix

$$G(T, t_0) = \int_{t_0}^{t_0+T} \Phi_A(s, t_0) C C^T \Phi_A^T(s, t_0) ds,$$
$$C \in \mathbb{R}^{n+m},\ C = \mathrm{col}(1, 0, \ldots, 0),$$

where $\Phi_A(t, t_0)$, is the normalized (i.e. $\Phi_A(t_0, t_0) = I_{n+m}$) fundamental solution matrix of

$$\begin{aligned} \dot{\chi} &= A(y(t), t)\chi, \\ A(y(t), t) &= \begin{pmatrix} F(y(t), t) & \Psi(y(t), t) \\ 0 & 0 \end{pmatrix}, \end{aligned} \tag{3.7}$$

is of full-rank, i.e $\mathrm{rank}(G(T, t_0)) = n + m$.

Remark 3: Regarding condition 1, observe that T generally depends on the unknown x_0, θ, λ. We will, however, assume that the value of T is measured and known. This, in what follow next, obviates the need to account for such dependences within the scope of this work. We would, however, like to acknowledge this for clarity here.

The class of equations (3.6) accommodates a broad set of technical and natural systems ranging from models of [9], dynamics of populations [54], and neural membranes [84]. In case the solutions are periodic it also may, after suitable

modifications [107], include

$$\begin{aligned} \dot{x} &= F(y,t)x + \Psi(y,t)\theta + g(y,q,\lambda,t) \\ \dot{q} &= \upsilon(y,\lambda,t)q + \omega(y,\lambda,t) \\ y &= C_1^T x;\ x(t_0) = x_0,\ q(t_0) = q_0. \end{aligned} \tag{3.8}$$

Indeed, the variable q admits a closed-form solution

$$\begin{aligned} q(t;q_0,\lambda,y) = e^{\int_{t_0}^{t}\upsilon(y(\tau),\lambda,\tau)d\tau}q_0 + e^{\int_{t_0}^{t}\upsilon(y(\tau),\lambda,\tau)d\tau}\times \\ \int_{t_0}^{t} e^{-\int_{t_0}^{\tau}\upsilon(y(s),\lambda,s)ds}\omega(y(\tau),\lambda,\tau)d\tau \\ q_0 = (1 - e^{-\int_{t_0}^{t_0+T}\upsilon(y(s),\lambda,s)ds})^{-1}\int_{t_0}^{t_0+T} e^{-\int_{z}^{t_0+T}\upsilon(y(s),\lambda,s)ds}\omega(y(z),\lambda,z)dz. \end{aligned} \tag{3.9}$$

which can be explicitly satisfied into the first equation of (3.8).

In addition to the system (3.6) we will consider its special case of the form

$$\begin{aligned} \dot{x} &= F_0 x + b\phi(y,t)^T\theta + g(y,q,\lambda,t) \\ \dot{q} &= \upsilon(y,\lambda,t)q + \omega(y,\lambda,t) \\ y &= C_1^T x;\ x(t_0) = x_0 \end{aligned} \tag{3.10}$$

where $\theta \in \mathbb{R}^m$, $F_0 = \begin{pmatrix} 0 & I_{n-1} \\ 0 & 0 \end{pmatrix}$, $\phi : \mathbb{R}\times\mathbb{R} \to \mathbb{R}^m$ and $b = (1, b_1, \cdots, b_{n-1})$ is such that the polynomial $s^{n-1} + b_1 s^{n-2} + \cdots + b_{n-1}$ is Hurwitz and the properties $(1-6)$ of the assumption 1 hold for the system (3.10). As we shall show later, for this special class one can final a somewhat simpler solution of the inverse problem, and the corresponding conditions are easier to check.

For notational convenience (cf. [105]), in what follows, we will combine the state variable x and parameters θ entering the right-hand-side of (3.6) linearly into a single variable χ and rewrite the system accordingly:

$$\begin{aligned} \dot{\chi} &= A(y,t)\chi + \begin{pmatrix} g(y,\lambda,t) \\ 0 \end{pmatrix} \\ y(t) &= C^T\chi;\ \chi(t_0) = \chi_0. \end{aligned} \tag{3.11}$$

In (3.11) $\chi = (x, \theta)$ is the combined state vector, matrix $A(y, t)$ is defined as in (3.7), and $C \in \mathbb{R}^{n+m}$ is $C = \text{col}(1, 0, \cdots, 0)$. Let us now proceed with the formal definition of the problem.

3.2.2 Problem statement

Consider system (3.11), and suppose that the values of $y(t)$ for $t \in [t_0, t_0 + T]$ are known and available *a-priori*. These values will depend on the parameters λ and initial condition χ_0 which themselves are assumed to be *unknown*. The question is if there exists an operator $\mathcal{F}$ mapping $y(\cdot)$ over $[t_0, t_0 + T]$ into an efficiently computable quantity that does depend on the parameters λ explicitly? Formally we are seeking to find an $\mathcal{F}(\lambda, y, t)$ such that

$$\begin{aligned} & C^T \chi(t; t_0, \chi_0, \lambda) = \mathcal{F}(t, \lambda, [y]) \\ & \mathcal{F}(t, \lambda, [y]) = \pi(t, \lambda, [y]) + \int_{t_0}^{t} u(\tau, \lambda, y(\tau), [y]) d\tau \\ & \forall\ t \in [t_0, t_0 + T],\ \ \lambda \in \Omega_\lambda \end{aligned} \tag{3.12}$$

in which the functionals π and u are known and computable, e.g. in quadratures. In some special cases the functionals π, u may not depend on χ_0 as a parameter, but nevertheless have to ensure that the required representation (3.12) holds. In this thesis we will focus on these special cases. When such a representation is found one can employ numerous off-line numerical optimization techniques to infer the values of λ, θ, and initial conditions from the values of y in the interval $[t_0, t_0 + T]$. We will illustrate this step with examples in which the Nelder-Mead algorithm [86] will be used for this purpose.

In Figure 3.1 we show the work-flow and illustrate the advantages of the proposed method relative to the current state-of-the-art.

Now, the question is if there is an equivalent integral formulation such as e.g. (3.12) for (3.11) and (3.10)? If such an integral formulation exists then whether a reduced-complexity version of this formulation can be stated so that the dimension of the parameter vector in the reduced formulation is smaller than

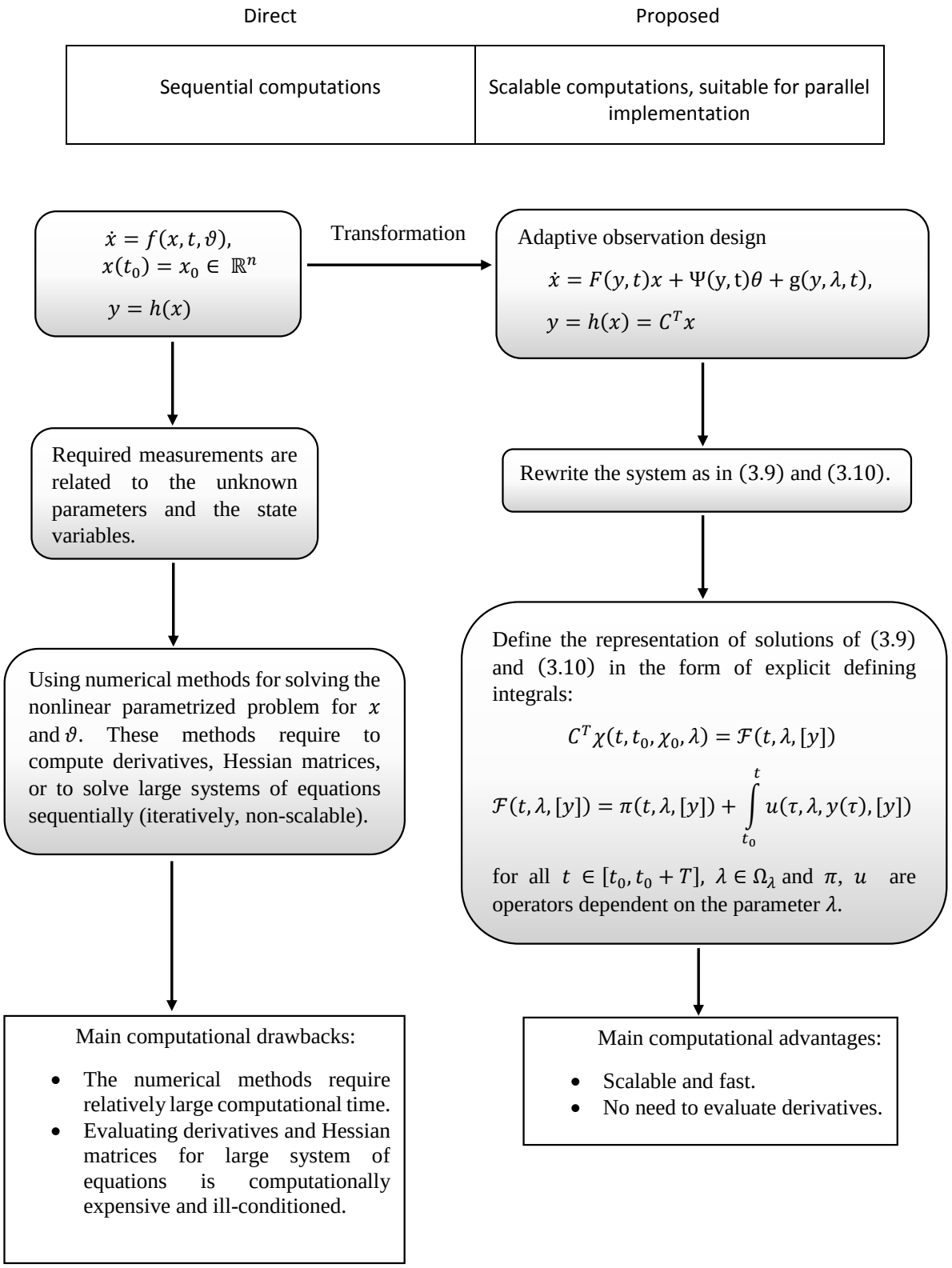


FIGURE 3.1: Workflow of the direct and proposed appraches.

that in the original problem? Answers to these questions are provided in the next section.

The method that we propose is for explicit integral reformulation of inverse problems for a class of nonlinear systems. The technique is aimed at using parallel computational streams to reduce the time of calculations. It has been shown that the method allows to reduce dimensionality of the problem to that of the dimension $n + p$ of the vector of parameters entering the right-hand side of the model nonlinearly.

This uncertainty reduction could be achieved in the proposed method if all functions in the right-hand side of (3.8) and (3.10) are T-periodic. Such periodicity assumptions may not always hold. In these cases the general approach still applies but the number of uncertainties becomes $n + m + p$.

Considering the representation form of solution (3.12) of the ODE problem raises several questions which are generally corresponding to whether the parametrization problem has finite solutions for λ? and if the solution is unique?. In Lemma 3.1 and Corollary 3.2 we will answer these questions.

Remark 4: Note that the proposed "transformed" system (3.11) has a very specific structure. In this structure the matrix F depends on y and y is a scalar. Whilst this presents a clear constraint on the applicability of the method, the class of systems (3.11) is reasonably broad. Moreover, the approach need not be generally confined to (3.11) as the major motivation for (3.11) was existence of an adaptive observer for x, ϑ for which a closed-form can be written down. If such an observer exists for other systems, the method will apply to those too.

3.3 Main Result

3.3.1 Indistinguishable parametrizations of (3.11) and (3.10)

Lemma 3.1. *Consider the following class of system*

$$\begin{aligned} \dot{\chi} &= A_0(t)\chi + u(t) + d(t), \\ y &= C^T\chi, \quad \chi(t_0) = \chi_0, \quad \chi_0 \in \mathbb{R}^\ell \end{aligned} \tag{3.13}$$

where

$$A_0(t) = \left(\begin{array}{c|ccccc} \alpha_1(t) & \beta_2(t) & \beta_3(t) & \cdots & \beta_\ell(t) \\ \hline \alpha_2(t) & & & & \\ \vdots & & A_0^*(t) & & \\ \alpha_\ell(t) & & & & \end{array}\right)$$

and $u, d, \alpha : \mathbb{R} \to \mathbb{R}^\ell$, $\beta : \mathbb{R} \to \mathbb{R}^{\ell-1}$, $u \in \mathcal{C}^1$, $d, \alpha, \beta \in \mathcal{C}$, $\alpha = \mathrm{col}(\alpha_1(t), \alpha_2(t), \ldots, \alpha_\ell(t))$, $\beta = (\beta_2(t), \beta_3(t), \ldots, \beta_\ell(t))$, *and assume that solutions of* (3.13) *are globally bounded in forward time. Let, in addition:*

1) $u, \dot{u}, d, \alpha, \beta$ *be bounded:* $\max\{\|u(t)\|, \|\dot{u}(t)\|\} \leq B, \|d(t)\| \leq \triangle_\xi, \|\alpha(t)\| \leq M_1, \|\beta(t)\| \leq M_2$ *for all* $t \geq t_0$.

2) there exist a $b : \mathbb{R} \to \mathbb{R}^{\ell-1}, b \in \mathcal{C}, \|b(t)\| \leq M_3$ *such that the zero solution of the system*

$$\dot{z} = \Lambda(t)z, \qquad \Lambda(t) = A_0^*(t) - b(t)\beta(t),$$

is uniformly exponentially stable, and let $\Phi_\Lambda(t, t_0)$ *be the corresponding fundamental solution:* $\Phi_\Lambda(t_0, t_0) = I_\ell$. *Then the following statements hold:*

1) If the solution of (3.13) *is globally bounded for all* $t \geq t_0$ *then, for* T *sufficiently large and for small positive* ϵ, *there are* $k_1, k_2 \in \mathcal{K}$ *:*

$$\|y(t)\|_{\infty,[t_0,t_0+T]} \leq \epsilon \Rightarrow \exists \;\; t'(\epsilon, x_0) \geq t_0 : \|h(t) + u_1(t)\|_{\infty,[t',t_0+T]} \leq k_1(\epsilon) + k_2(\triangle_\xi), \tag{3.14}$$

where $h(t) = \beta(t)z$,

$$\begin{aligned} &\dot{z} = \Lambda(t)z + Gu, \\ &G = \begin{pmatrix} -b(t) & I_{\ell-1} \end{pmatrix}, \quad z(t_0) = 0, \end{aligned} \tag{3.15}$$

2) If $d(t) \equiv 0$, *then* $y(t) = 0$ *for all* $t \in [t_0, t_0 + T]$ *implies existence of* $P \in \mathbb{R}^{\ell-1}$*:*

$$\beta(t)\Phi_\Lambda(t, t_0)P + h(t) + u_1(t) = 0 \tag{3.16}$$

for all $t \in [t_0, t_0 + T]$.

The proof of the lemma 3.1 is provided in the Appendix.

According to Lemma 3.1 the set of parameters:

$$\mathcal{E}(\lambda) = \{\lambda' \in \mathbb{R}^p | \ \exists \ P \in \mathbb{R}^{\ell-1} : \ \eta(t, P, \lambda', \lambda) = 0, \forall t \in [t_0, t_0 + T]\} \tag{3.17}$$

where

$$\begin{aligned} \eta(t, P, \lambda', \lambda) = \beta(t)\Phi_\Lambda(t, t_0)P + \beta(t) \int_{t_0}^{t} \Phi_\Lambda(t, \tau) \\ G(\tau) \begin{pmatrix} g(y(\tau), \lambda', \tau) - g(y(\tau), \lambda, \tau) \\ 0 \end{pmatrix} d\tau, \end{aligned}$$

and Λ is defined as in (3.15), contains parameters λ' producing measurements $y(t) = C^T\chi(t; t_0, \chi_0, \lambda')$ that are indistinguishable from $C^T\chi(t; t_0, \chi_0, \lambda)$ on the interval $[t_0, t_0 + T]$. If the set $\mathcal{E}(\lambda)$ contains more than one element then the system (3.11) may not be uniquely identifiable on $[t_0, t_0 + T]$. Notwithsdanding existence and possible utility of systems that are not uniquely identifiable, we will nevertheless focus on systems (3.11) that are uniquely identifiable on $[t_0, t_0 + T]$.

Thus we assume that the following holds:

Assumption 2. For every $\lambda \in \Omega_\lambda$, the set $\mathcal{E}(\lambda)$ consists of just one element.

Corollary 3.2. *Consider*

$$\begin{aligned} \dot{\chi} &= A_0 x + u(t) + d(t), \\ y &= C_1^T x, \quad x(t_0) = x_0, \quad x_0 \in \mathbb{R}^n \end{aligned} \tag{3.18}$$

where $A_0 = \left(\begin{array}{c|c} \alpha_1 & \\ \alpha_2 & I_{n-1} \\ \vdots & \\ \alpha_n & 0 \end{array}\right)$, $C_1 = \mathrm{col}(1, 0, \cdots, 0)$ *and* $x, u, d : \mathbb{R} \to \mathbb{R}^n$, $, \alpha \in \mathbb{R}^n$, $u \in \mathcal{C}^1$, $d \in \mathcal{C}$. *Let* $u(\cdot), \dot{u}(\cdot), d(\cdot)$ *be bounded:* $\max\{\|u(t)\|, \|\dot{u}(t)\|\} \leq B, \|d(t)\| \leq \triangle_\xi$ *for all* $t \geq t_0$. *Then the following hold:*

1) If the solution of (3.13) *is globally bounded for all* $t \geq t_0$ *then, for* T *sufficiently large, there are* $k_1, k_2 \in \mathcal{K}$*:* (3.14) *holds where* $z_1 = C_1 z$,

$$\begin{aligned} \dot{z} &= \Lambda z + Gu, \ \Lambda = \left(\begin{array}{cc} & \vdots \quad I_{n-2} \\ -b & \\ & \vdots \quad 0 \end{array}\right) \\ G &= \left(\begin{array}{cc} -b & I_{n-1} \end{array}\right), \quad z(t_0) = 0, \end{aligned} \tag{3.19}$$

and $b = (b_1, \cdots, b_{n-1})^T$*: real parts of the roots of* $s_{n-1} + b_1 s_{n-2} + \cdots + b_{n-1}$ *are negative.*

2) If $d(t) \equiv 0$, *then* $y(t) = 0$ *for all* $t \in [t_0, t_0 + T]$ *implies existence of* $P \in \mathbb{R}^{n-1}$*:*

$$C_1 e^{\Lambda(t-t_0)} P + z_1(t) + u_1(t) = 0 \tag{3.20}$$

for all $t \in [t_0, t_0 + T]$.

The corollary 3.2 is proved in the Appendix.

According to Corollary 3.2 the following sets of parameters, associated with every θ,λ, need special consideration. The first set is defined as

$$\begin{aligned} \mathcal{E}_0(\theta, \lambda) = \{(\theta', \lambda'), \theta' \in \mathbb{R}^m, \lambda' \in \mathbb{R}^p | & b\varphi(y(t), t)^T (\theta' - \theta) \\ & + g(y(\tau), \lambda', \tau) - g(y(\tau), \lambda, \tau) = 0, \forall t \in [t_0, t_0 + T]\}. \end{aligned}$$

The set $\mathcal{E}_0(\theta,\lambda)$ contains all parametrizations of (3.10) which are indistinguishable from each other providing that the values of x(t) are known for all $t \in [t_0, t_0+T]$. That is, if $x(t;t_0,x_0,\theta,\lambda) = x(t;t_0,x_0,\theta',\lambda')$ for all $t \in [t_0,t_0+T]$ then $(\theta',\lambda') \in \mathcal{E}_0(\theta,\lambda)$. Denote

$$\eta(t,p,\theta',\lambda',\theta,\lambda) = \varphi(y(t),t)(\theta'-\theta) + g_1(y(t),\lambda',t) - g_1(y(t),\lambda,t) + C_1^T e^{\Lambda(t-t_0)}p + z_1(t;t_0,\lambda') - z_1(t;t_0,\lambda),$$

where Λ,C_1,$z(t;t_0,\lambda')$ are defined as in (3.19) with $u(t)$ replaced by $g(y(t),\lambda',t)$. The second set is defined as

$$\mathcal{E}(\theta,\lambda) = \{(\theta',\lambda'), \theta' \in \mathbb{R}^m, \lambda' \in \mathbb{R}^p | \ \exists\ p \in \mathbb{R}^{\ell-1} :\ \eta(t,p,\lambda',\lambda) = 0, \forall t \in [t_0,t_0+T]\}. \tag{3.21}$$

In accordance with Corollary 3.2 the set $\mathcal{E}(\theta,\lambda)$ contains all parametrizations of (3.10) that are indistinguishable on the interval $[t_0,t_0+T]$ on the basis of accessing only the values of $y(x(t;t_0,x_0,\theta,\lambda))$. In other words, if $y(x(t;t_0,x_0,\theta,\lambda)) = y(x(t;t_0,x_0',\theta',\lambda'))$ for all $t \in [t_0,t_0+T]$ then $(\theta',\lambda') \in \mathcal{E}(\theta,\lambda)$. If $\mathcal{E}(\theta,\lambda)$ contains more than one element then (3.10) is not is not uniquely identifiable on $[t_0,t_0+T]$ [27].

Here, for simplicity, we will focus on systems (3.10) that are uniquely identifiable on $[t_0,t_0+T]$:

Assumption 3. Sets $\mathcal{E}_0(\theta,\lambda)$ and $\mathcal{E}(\theta,\lambda)$ are coincide and contain no more than one element.

3.3.2 Integral parametrization of periodic solutions of (3.10)

Before we proceed with presenting an equivalent integral formulation of Problem (3.10) let us first introduce several additional components and corresponding technical assumptions. Let $l \in \mathbb{R}^n$ be a vector satisfying the following condition:

$$P(F_0 + lC_1^T) + (F_0 + lC_1^T)^T P = -Q,\ \ Pb = C_1$$

where P, Q are some symmetric positive definite matrices. According to the Meyer-Kalman-Yakubovich-Popov lemma [22], such vector will always exist since the polynomial $s^{n-1} + b_1 s^{n-2} + \cdots + b_{n-1}$ is Hurwitz. Consider

$$\dot{\hat{\chi}} = \begin{pmatrix} F_0 + lC_1^T & b\phi(y(t),t)^T \\ -\phi(y(t),t)C_1^T & 0 \end{pmatrix} \hat{\chi} \tag{3.22}$$

and let $\Phi(t, t_0)$ be its corresponding normalized fundamental solution matrix: $\Phi(t_0, t_0) = I_{n+m}$.

Theorem 3.3. *Consider* (3.10) *and suppose that Assumption 3 holds. Let* $y(t)$, $\phi(y(t),t)$, $g(y(t), \lambda, t)$ *be* $T-$*periodic on* $[0, \infty)$ *for all* λ, *and the function* $\phi(y(t),t)$ *satisfy:*

$$\int_{t_0}^{t_0+T} \phi(y(\tau),\tau)\phi(y(\tau),\tau)^T d\tau \geq \delta I_{n+m}, \ \delta > 0. \tag{3.23}$$

Then the following statements are equivalent:

1) $\hat{y}(\lambda', t) = y(t)$ *for all* $[t_0, t_0 + T]$, *where* $\hat{y} : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$*:*

$$\hat{y}(\lambda', t) = (1\ 0\ \cdots\ 0)^T \big(\Phi(t, t_0)\hat{\chi}_0 + \int_{t_0}^{t} \Phi(t,\tau) \begin{pmatrix} g(y(\tau), \lambda, \tau) - ly(\tau) \\ y(\tau)\phi(y(\tau),\tau) \end{pmatrix} d\tau\big), \tag{3.24}$$

$$\hat{\chi}_0 = (I_{n+m} - \Phi(t_0 + T, t_0))^{-1} \int_{t_0}^{t_0+T} \Phi(t_0 + T, \tau) \begin{pmatrix} g(y(\tau), \lambda, \tau) - ly(\tau) \\ y(\tau)\phi(y(\tau),\tau) \end{pmatrix} d\tau. \tag{3.25}$$

2) $(1\ 0\ \cdots\ 0)x(t; t_0, x_0, \tilde{\theta}, \lambda') = y(t)$ *for all* $[t_0, t_0 + T]$.

Furthermore, the values of x_0, $\tilde{\theta}$ *satisfy*

$$\begin{pmatrix} x_0 \\ \tilde{\theta} \end{pmatrix} = \hat{\chi}_0. \tag{3.26}$$

Proof. Let us first show that 1) $\Rightarrow$ 2). Recall (see *e.g.* [68]) that assumptions of the theorem imply existence of positive numbers $\rho, D > 0$:

$$\|\Phi(t, t_0)\| \leq De^{-\rho(t-t_0)}$$

for all $t \geq t_0',\ t_0' \in [t_0, \infty)$.

Hence the matrix $I_{n+m} - \Phi(t_0+T, t_0)$ has no zero eigenvalues, and its inverse matrix, $(I_{n+m} - \Phi(t_0 + T, t_0))^{-1}$, exists. Consider $\chi = (\chi_1, \chi_2)$:

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} F_0 + lC_1^T & b\phi(y(t), t)^T \\ -\phi(y(t), t)C_1^T & 0 \end{pmatrix}\begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \\ + \begin{pmatrix} g(y(t), \lambda', t) - ly(t) \\ y(t)\phi(y(t), t) \end{pmatrix}. \end{aligned} \tag{3.27}$$

It is clear that solutions of (3.27) are defined for all $t \geq t_0$ providing that the definition of $y(\cdot)$, $g(y(t), \lambda', t)$, and $\phi(y(\cdot), \cdot)$ are extended (periodically) on the interval $[t_0, \infty)$. Introduce the function $\zeta(\cdot) = (x(\cdot, t_0, x_0, \tilde{\theta}, \tilde{\lambda}), \tilde{\theta})$ (in which the domain of the function $x(\cdot, t_0, x_0, \tilde{\theta}, \tilde{\lambda})$ definition is extended to $[t_0, \infty)$), and consider the difference

$$\xi = \chi - \zeta.$$

Dynamics of ξ satisfy (3.22) with $\xi_1(t_0) = \chi_1(t_0) - x(t_0)$, $\xi_2(t_0) = \chi_2(t_0) - \tilde{\theta}$. Moreover, $\hat{y}(\lambda', t) = C^T\chi_1(t)$ for all $[t_0, t_0+T]$ (or in $[t_0, \infty)$ if $\hat{y}(\lambda', \cdot)$ is periodically extended on $[t_0, \infty)$).

Let $\hat{y}(\lambda', t) \equiv y(t)$. This implies that $\chi_2 - \tilde{\theta}$ =const for all $t \in [t_0, t_0 + T]$. Hence according to Corollary 3.2 $(\chi_2(t_0), \lambda')$ belong to $\mathcal{E}(\tilde{\theta}, \tilde{\lambda}, T)$. Given that sets $\mathcal{E}(\tilde{\theta}, \tilde{\lambda}, T)$ and $\mathcal{E}_0(\tilde{\theta}, \tilde{\lambda}, T)$ coincide and contain just one element, $\tilde{\theta}, \tilde{\lambda}$, we conclude that $\chi_2(t_0) = \tilde{\theta}$, $\lambda' = \tilde{\lambda}$.

Notice that $\lim\limits_{t\to\infty} \xi(t) = 0$ for all $\chi_2(t_0)$, and that

$$\Phi(t, t_0)\hat{\chi}_0 + \int_{t_0}^{t} \Phi(t, \tau)\begin{pmatrix} g(y(\tau), \lambda', \tau) - ly(\tau) \\ y(\tau)\phi(y(\tau), \tau) \end{pmatrix} d\tau$$

is the unique exponentially stable periodic solution of (3.27). This implies that (3.26) holds.

Let us show that 2) $\Rightarrow$ 1). Let $\tilde{\theta}, \lambda'$ be parameters for which the following identity holds: $y(x(t; t_0, x_0, \theta, \lambda)) = y(t)$ for all $[t_0, t_0 + T]$. Consider the function $\zeta(\cdot)$ defined earlier. Given that (3.28) is the unique exponentially stable periodic solution of (3.27), that $\lim\limits_{t\to\infty} \zeta(t) = 0$ for arbitrary choice of initial conditions (*i.e.* vectors $x(t_0)$, $\tilde{\theta}$, and $\chi_1(t_0)$, $\chi_2(t_0)$) and that $\zeta(t) \equiv 0$ if $\chi_1(t_0) = x(t_0)$, $\chi_2(t_0) = \tilde{\theta}$, one includes that $\hat{y}(\lambda', t) = y(x(t; t_0, x_0, \theta, \lambda)) = y(t)$ for all $[t_0, t_0 + T]$. □

3.3.3 Integral parametrization of periodic solutions of (3.11)

In addition to (3.11) consider the following *auxiliary* system:

$$\begin{aligned}
&\dot{\hat{\chi}} = A(y(t), t)\hat{\chi} + \begin{pmatrix} g(y(t), \lambda', t) \\ 0 \end{pmatrix} + R^{-1}C(C^T\hat{\chi} - y), \\
&\dot{R} = -\delta R - A(y(t), t)^T R - RA(y(t), t) + CC^T \\
&\hat{\chi}(t_0) = \hat{\chi}_0 \in \mathbb{R}^{n+m},\ R(t_0) \in \mathbb{R}^{(n+m)\times(n+m)}
\end{aligned} \tag{3.28}$$

where $\hat{\chi} \in \mathbb{R}^{n+m}$ is the observer's state, $R(t_0)$ is a positive-definite symmetric matrix, and $\delta \in \mathbb{R}_{>0}$ is a positive parameter. Solutions of (3.28) are defined for all $t \geq t_0$ (see items (1), (2) in Assumption 1), and hence, [44], $R(t)$ is given by

$$\begin{aligned}
R(t) =& e^{-\delta(t-t_0)}\Phi_A(t_0, t)^T R(t_0)\Phi_A(t_0, t) + \\
&\int_{t_0}^{t} e^{-\delta(t-s)}\Phi_A(s, t)^T CC^T\Phi_A(s, t)ds.
\end{aligned} \tag{3.29}$$

It is clear that $R(t)$ is non-singular for all $t \geq t_0$, symmetric, and positive-definite. Furthermore, if the value of the parameter $\delta > 0$ is chosen so that

$$\|e^{-\delta(t-t_0)/2}\Phi_A(t_0, t)\| \leq De^{-a(t-t_0)},\ a > 0, \tag{3.30}$$

then $R(t)$ is bounded. In what follows the following additional assumption is instrumental:

Assumption 4. There exist $t_1 \geq t_0$ and $\alpha(\delta) > 0$ such that

$$\phi(t,\delta) = \int_{t_0}^{t} e^{-\delta(t-s)} \Phi_A(s,t)^T C C^T \Phi_A(s,t) ds \geq \alpha(\delta) I_{n+m}$$

for all $t \geq t_1$.

The next theorem specifies asymptotic behaviour of the observer system (3.28) (adapted from [44]).

Theorem 3.4. *Consider (3.28) and suppose that $\delta > 0$ be chosen so that both (3.30) and Assumption 4 hold, and $\lambda' = \lambda$. Then there exists a $t_2 \geq t_0$, such that:*

$$\|\hat{\chi}(t;\hat{\chi}_0) - \chi(t;\chi_0)\| \leq k e^{-\delta(t-t_0)}$$

for all $t \geq t_2$, where k is a constant dependent on δ, t_0, χ_0 and the initial state $\hat{\chi}_0$ of the observer system (3.28).

Theorem 3.4 states the variable $\hat{\chi}(t)$ asymptotically tracks $\chi(t)$, and that the difference between the two converges to zero exponentially. Here, however, we are interested in establishing finite-time relationships (3.12). To do so we need another technical result establishing sufficient conditions for the existence of unique periodic solutions of R. The result is provided in Lemma 3.5.

Lemma 3.5. *Consider (3.28) with $A(y(t),t)$ being T-periodic. Then, for sufficiently large $\delta > 0$, there exists a unique symmetric $R(t_0)$ ensuring that the function $R(t)$ defined by* (3.29) *is T-periodic. If, in addition, (3.30) and Assumption 4 hold then $R(t_0)$ is positive-definite.*

Proof. Consider $R(t+T)$ and its derivative wrt. t:

$$\dot{R}(t+T) = -\delta R(t+T) - A(y(t+T),t+T)^T R(t+T) - R(t+T) A(y(t+T),t+T) + CC^T.$$

Since $A(y(t+T),t+T) = A(y(t),t)$ for all $t \geq t_0$, we have

$$\dot{R}(t+T) = -\delta R(t+T) - A(y(t),t)^T R(t+T) - R(t+T) A(y(t),t) + CC^T. \quad (3.31)$$

Denoting $E(t) = R(t+T) - R(t)$ and invoking (3.31) we obtain:

$$\dot{E} = -\delta E - A(y(t),t)^T E - EA(y(t),t). \tag{3.32}$$

If $R(t_0) = R(t_0+T)$ then $E(t) = \mathbf{0}$ is the unique $(n+m)$ zero matrix solution of (3.32). This implies that $R(t) = R(t+T)$ for all $t \geq t_0$. Let us show that such $R(t_0)$ exists. For $R(t_0) = R(t_0+T)$ to hold $R(t_0)$ must satisfy

$$\begin{gathered} R(t_0) = e^{-\delta T}\Phi_A(t_0, t_0+T)^T R(t_0)\Phi_A(t_0, t_0+T) + \\ \int_{t_0}^{t_0+T} e^{-\delta(T+t_0-s)}\Phi_A(s, t_0+T)^T CC^T \Phi_A(s, t_0+T)ds. \end{gathered} \tag{3.33}$$

Let us rewrite (3.33) as:

$$R(t_0) - H_1 R(t_0) H_2 = B, \tag{3.34}$$

where

$$\begin{aligned} H_1 &= e^{-\delta T/2}\Phi_A(t_0, t_0+T)^T, \\ H_2 &= e^{-\delta T/2}\Phi_A(t_0, t_0+T) \end{aligned}$$

and

$$B = \int_{t_0}^{t_0+T} e^{-\delta(T+t_0-s)}\Phi_A(s, t_0+T)^T CC^T \Phi_A(s, t_0+T)ds.$$

The matrices H_1, H_2 are non-singular by construction, and hence (3.34) is equivalent

$$H_1^{-1} R(t_0) - R(t_0) H_2 = H_1^{-1} B. \tag{3.35}$$

Moreover, $H_1 = H_2^T$. The latter implies that if $R(t_0)$ is a solution of (3.34) then so is $R(t_0)^T$:

$$R(t_0)^T = (H_1 R(t_0) H_2)^T + B^T = H_2^T R(t_0)^T H_1^T + B.$$

Further, (3.35) is the Sylvester equation [24]; it has a unique solution if the spectra of $(n+m)\times(n+m)$ matrices H_1^{-1} and H_2 are disjoint (i.e. H_1^{-1} and H_2 have no

common eigenvalues). Note that

$$H_1^{-1} = e^{\delta T/2}(\Phi_A(t_0, t_0+T)^T)^{-1},$$

and let $\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_{n+m}$ and $\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_{n+m}$ be the eigenvalues of $(\Phi_A(t_0, t_0 + T)^T)^{-1}$ and $\Phi_A(t_0, t_0+T)$, respectively. The moduli of eigenvalues α_i of H_1^{-1} and eigenvalues β_i of the matrix $-H_2$ are:

$$|\alpha_i| = e^{\delta T/2}|\tilde{\alpha}_i|,$$
$$|\beta_i| = e^{-\delta T/2}|\tilde{\beta}_i|$$

for all $i = 1, 2, \cdots, n+m$. Denote

$$\alpha_{\max} = \max_i\{|\tilde{\alpha}_i|\}, \quad \alpha_{\min} = \min_i\{|\tilde{\alpha}_i|\}$$
$$\beta_{\max} = \max_i\{|\tilde{\beta}_i\}, \quad \beta_{\min} = \min_i\{|\tilde{\beta}_i\}.$$

Given $\alpha_{\min} \neq 0$ one can pick the value of δ so large that

$$e^{\delta T} > \frac{\beta_{\max}}{\alpha_{\min}}.$$

Doing so implies that

$$e^{\delta T/2}\alpha_{\min} > e^{-\delta T/2}\beta_{\max}.$$

This, in turn, results in

$$|\alpha_i| > |\beta_j|, \forall\ i, j = 1, \ldots, n+m.$$

Hence

$$\alpha_i \neq \beta_j, \forall\ i, j = 1, \ldots, n+m,$$

and there is a symmetric matrix $R(t_0)$ satisfying (3.35) and, consequently, (3.33). Finally, let us show that if (3.30) and Assumption 4 hold then the corresponding $R(t_0)$ is positive-definite. Let N be a non-negative integer. Given that $R(t_0) = R(t_0 + NT)$ we see that

$$R(t_0) = e^{-\delta NT}\Phi_A(t_0, t_0+NT)^T R(t_0)\Phi_A(t_0, t_0+NT) + \phi(t_0+NT, \delta). \quad (3.36)$$

According to (3.30) the norm

$$\|e^{-\delta NT}\Phi_A(t_0, t_0+NT)^T R(t_0)\Phi_A(t_0, t_0+NT)\|$$

can be made arbitrarily small if N is large enough. At the same time, Assumption 4 guarantees that $\phi(t_0+NT, \delta) \geq \alpha(\delta)$ in (3.36) for all N that are sufficiently large. Since the value of N in (3.36) can be chosen arbitrary large we conclude that $R(t_0)$ is positive-definite too. □

For notational convenience, let us rewrite auxiliary observer equations (3.28) as:

$$\begin{aligned} \dot{\hat{\chi}} &= (A(t) - R^{-1}CC^T)\hat{\chi} + \begin{pmatrix} g(y(t), \lambda', t) \\ 0 \end{pmatrix} + R^{-1}Cy(t) \\ \dot{R} &= -\delta R - A(y(t), t)^T R - RA(y(t), t) + CC^T \\ \hat{\chi}(t_0) &= \hat{\chi}_0 \in \mathbb{R}^{n+m},\ R(t_0) \in \mathbb{R}^{(n+m)\times(n+m)} \end{aligned} \tag{3.37}$$

and additionally consider dynamics of the linear part of the first equation:

$$\xi = \left(A(y(t), t) - R^{-1}(t)CC^T \right) \xi. \tag{3.38}$$

Let $\Phi(t, s)$ be the normalized fundamental solution matrix of (3.28), i.e. $\Phi(t, t) = I_{n+m}$ and $\Phi(s, t) = \Phi(t, s)^{-1}$.

Theorem 3.6. *Consider system (3.28) and suppose that Assumptions 1 and 2 hold. In addition, suppose that condition (3.30) hold and the values of δ and the initial condition $R(t_0)$ in (3.37) is chosen such that $R(t) > 0$ is T-periodic. Consider the function $\hat{y} : \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}$:*

$$\begin{aligned} \hat{y}(\lambda', t) = C^T\big(\Phi(t, t_0)\hat{\chi}_0 + \int_{t_0}^{t} \Phi(t, \tau) \times \\ \left(R^{-1}(\tau)Cy(\tau) + \begin{pmatrix} g(y(\tau), \lambda', \tau) \\ 0 \end{pmatrix} \right) d\tau\big) \end{aligned} \tag{3.39}$$

where

$$\hat{\chi}_0 = (I_{n+m} - \Phi(t_0+T, t_0))^{-1} \int_{t_0}^{t_0+T} \Phi(t_0+T, \tau) \times \left(R^{-1}(\tau) C y(\tau) + \begin{pmatrix} g(y(\tau), \lambda', \tau) \\ 0 \end{pmatrix} \right) d\tau. \tag{3.40}$$

Then

$$\hat{y}(\lambda', t) = C\chi(t; t_0, \chi_0, \lambda) \ \forall \ t \in [t_0, t_0+T] \Leftrightarrow \lambda = \lambda'.$$

Proof. Sufficiency, i.e. implication $\Rightarrow$. Assumption 1 implies that Assumption 4 holds along the solution of (3.37). This together with condition (3.30) assure that there are positive constants $\rho, D > 0$ such that

$$\|\Phi(t, t_0)\| \leq De^{-\rho(t-t_0)}.$$

Hence the matrix $I_{n+m} - \Phi(t_0+T, t_0)$ has no zero eigenvalues, and its inverse matrix, $(I_{n+m} - \Phi(t_0+T, t_0))^{-1}$, exists. Thus $\hat{y}(\lambda', t)$ described by (3.39), (3.40) is defined for all $t \in [t_0, t_0+T]$. Periodicity of $R(t)$ implies that

$$\hat{\chi}(t; t_0, \hat{\chi}_0, \lambda') = \Phi(t, t_0)\hat{\chi}_0 + \int_{t_0}^{t} \Phi(t, \tau) \left(R^{-1}(\tau) C y(\tau) + \begin{pmatrix} g(y(\tau), \lambda', \tau) \\ 0 \end{pmatrix} \right) d\tau$$

with $\hat{\chi}_0$ defined by (3.40) is the unique asymptotically stable periodic solution of the $\hat{\chi}$-subsystem in (3.37). On this solution we have: $CC^T\hat{\chi}(t) - Cy(t) = 0$ for all $t \in [t_0, t_0+T]$. Thus

$$\dot{\hat{\chi}} = A(y(t), t)\hat{\chi} + \begin{pmatrix} g(y(t), \lambda', t) \\ 0 \end{pmatrix}, \ C^T\hat{\chi}(t) = y(t).$$

Consider $e = \hat{\chi} - \chi$:

$$\dot{e} = A(y(t), t)e + \begin{pmatrix} g(y(t), \lambda', t) \\ 0 \end{pmatrix} - \begin{pmatrix} g(y(t), \lambda, t) \\ 0 \end{pmatrix}.$$

According to Lemma 3.13 and Assumption 2 the set of indistinguishable parametrizations $\mathcal{E}(\lambda)$ of (3.11) comprises of a single element, and hence $\lambda' = \lambda$.

Necessity, $\Leftarrow$.

Let $\lambda = \lambda'$. According to assumptions of the theorem dynamics of $\hat{\chi} - \chi$ satisfies (3.38). The zero solution of the latter is globally asymptotically stable, and hence $\lim_{t\to\infty} \hat{\chi}(t) - \chi(t) = 0$. Noticing that (3.41) is the unique exponentially stable periodic solution of the $\hat{\chi}$-subsystem in (3.37) we obtain that $\hat{\chi}(t; t_0, \hat{\chi}_0, \lambda') = \chi(t; t_0, \chi_0, \lambda)$ for all $t \in [t_0, t_0 + T]$, and hence $\hat{y}(\lambda', t) = C^T \chi(t; t_0, \chi_0, \lambda)$. □

Remark 5: In the proposed methods, instead of dealing with continuous-time signals, $y(t)$, one may re-formulate the above results in the setting in which model responses and data are compared at mere N discrete points $\{t_i\}$ in $[t_0, t_0+T]$. In this case sets of parameters $\mathcal{E}_0$ and $\mathcal{E}$ in Corollary 3.2 and $\mathcal{E}$ in Lemma 3.1 will need to be re-defined so that the corresponding identities hold at the given finite number of points $\{t_i\}$ rather than for all $t \in [t_0, t_0 + T]$. Discrete extension of the theorems allows straightforward formulation of the inference problem as

$$\hat{\lambda} = \arg\min_{\lambda\in\mathbb{R}^p} \{\sum_{i=1}^{N} (y(t_i) - \hat{y}(t_i, \lambda))^2\} \tag{3.41}$$

Remark 6: To deal with discrete measurements using continuous-time models, one can employ a suitable interpolation scheme providing that the outcome of such an interpolation is phenomenologically adequate.

Remark 7: The methods, as formulated in Theorems 3.3 and 3.6, require periodicity of $y(t)$ as a function of t. Similar representations can be obtained for models that do not necessarily produce periodic signals. This, however, will bear additional costs. In absence of periodicity, $\hat{\chi}_0$ in (3.24) and (3.39) will generally be replaced by an unknown vector. Yet, due to the asymptotic stability of the solutions of the observer, if the interval of observation is long enough then relative contribution of this unknown term in $y(\lambda, t)$ for t sufficiently large will be small. Thus $y(\lambda, t)$ becomes an approximation of the measured $y(t)$ rather than an exact match.

3.4 Examples

3.4.1 Predator-Prey system

The mathematical problem of Predator Prey is considered in the following system of equations:

$$\begin{aligned} \dot{x} &= p_1 x(1 - \tfrac{x}{p_2}) - \tfrac{p_3 z x}{p_4 + x} \\ \dot{z} &= \tfrac{p_5 z x}{p_4 + x} - p_6 z, \end{aligned} \tag{3.42}$$

where x, z are population densities of prey and predator, respectively, and $p = (p_1, p_2, \cdots, p_6)$ are parameters that are subjected to evolutionary modifications. Suppose that parameter values correspond to the unique stable limit cycle, and for simplicity we assume that the system evolves on the cycle (or in its sufficiently small vicinity). The corresponding parameter values and initial conditions are set as follows:

$$\begin{aligned} &p_1 = 1,\ p_2 = 1.3,\ p_3 = 1,\ p_4 = 1, p_5 = 3,\ p_6 = 0.1, \\ &(x_0, z_0) = (0.0053, 0.2536). \end{aligned} \tag{3.43}$$

Note that system (3.42) is not in the class of systems specified by (3.10) to which Theorem 3.3 applies. It may, however, be transformed into this class as follows. Given that $p_3, p_5 \neq 0$ we will first introduce a new variable

$$q = x + \frac{p_3}{p_5} z.$$

Thus, in accordance with (3.42):

$$\dot{q} = p_1 x(1 - \frac{x}{p_2}) - \frac{p_3 z x}{p_4 + x} + \frac{p_3 z x}{p_4 + x} - p_6 \frac{p_3}{p_5} z, \tag{3.44}$$

and the system equations in the new coordinates become:

$$\begin{aligned} \dot{x} &= p_1 x(1 - \frac{x}{p_2}) - \frac{p_5 (q - x) x}{p_4 + x} \\ \dot{q} &= p_1 x(1 - \frac{x}{p_2}) + p_6 x - p_6 q. \end{aligned} \tag{3.45}$$

Equation (3.45) is in the original form of (3.10) before transformation. The latter, in turn, can be transformed into (3.10) by means of the following closed form expression for the variable q:

$$q(t,p_1,p_2,p_6) = e^{-p_6(t-t_0)}q_0(p_1,p_2,p_6)+e^{-p_6 t}\int_{t_0}^{t} e^{p_6\tau}\left(p_1 x(\tau)(1-\frac{x(\tau)}{p_2}) +p_6 x(\tau)\right)d\tau. \tag{3.46}$$

The observed variable, $x(\cdot)$, is periodic with period $T = 34.05$, and $p_6 = 0.1 \neq 0$. Therefore

$$q(t,p_1,p_2,p_6) = (1-e^{-p_6 T})^{-1}e^{-p_6 T}\int_{t_0}^{t_0+T} e^{p_6\tau}\left(p_1 x(\tau)(1-\frac{x(\tau)}{p_2}) \ +p_6 x(\tau)\right)d\tau. \tag{3.47}$$

Hence dynamics of x obeys

$$\dot{x} = p_1 x - \frac{p_1}{p_2}x^2 + \frac{p_5 x^2}{p_4+x} - \frac{p_5 x}{p_4+x}q(t,p_1,p_2,p_6) \tag{3.48}$$

which is of class (3.10) with

$$\tilde{\lambda} = (p_1,p_2,p_4,p_5,p_6).$$

Thus Theorem 3.3 applies, and observed periodic trajectory $x(\cdot)$ of system (3.42) can be represented as an explicit integral.

Notice that the number of parameters in (3.48) is reduced to just 5 as compared to 6 in the original equations. Moreover, since the right-hand side of (3.48) is purely nonlinearly parameterized, there is no $\phi(\cdot)$ in (3.24), $F_0 = 0$, and the fundamental solution matrix, $\Phi(t,t_0)$, becomes

$$\Phi(t,t_0) = e^{l(t-t_0)},\ l \in \mathbb{R},\ l < 0. \tag{3.49}$$

For the sake of simplicity we set $l = -1$. The corresponding expression for $\hat{y}(\tilde{\lambda}, t)$ is

$$\begin{aligned}\hat{y}(\tilde{\lambda}, t) &= e^{-(t-t_0)}\chi_0(\tilde{\lambda}) + \int_{t_0}^{t} e^{-(t-\tau)}\Big(x(\tau) + p_1 x(\tau) - \frac{p_1}{p_2}x^2(\tau) \\ &\quad + \frac{p_5 x^2(\tau)}{p_4 + x(\tau)} - \frac{p_5 x(\tau)}{p_4 + x(\tau)} q(t, p_1, p_2, p_6)\Big) d\tau \\ \chi_0(\tilde{\lambda}) &= (1 - e^{-T})^{-1} \int_{t_0}^{t_0+T} e^{-(T-\tau)}\Big(x(\tau) + p_1 x(\tau) - \frac{p_1}{p_2}x^2(\tau) \\ &\quad + \frac{p_5 x^2(\tau)}{p_4 + x(\tau)} - \frac{p_5 x(\tau)}{p_4 + x(\tau)} q(t, p_1, p_2, p_6)\Big) d\tau.\end{aligned} \tag{3.50}$$

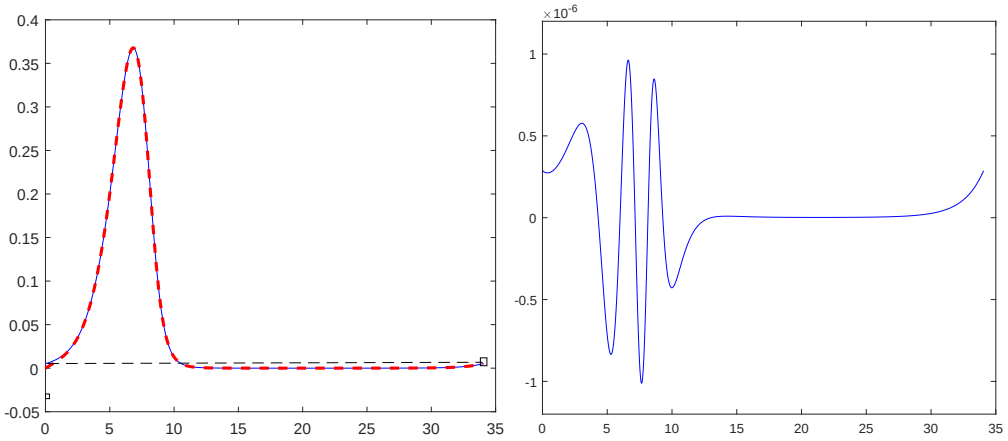


FIGURE 3.2: Left panel: the values of $y(t) = x(t; t_0, (x_0, z_0), \lambda)$ and $\hat{y}(\tilde{\lambda}, t)$ as functions of t for $\lambda = (p_1, p_2, \cdots, p_6)$ and initial conditions specified by (3.43). Black circles indicate starting and ending points of the periodic trajectory $y(t)$. The values of $y(t)$ (blue curve) were obtained by numerical integration of (3.42) by Runge Kutta of 4th order method with integration step 0.001. The values of $\hat{y}(\tilde{\lambda}, t)$ (dashed red curve) have been computed from representation (3.50) numerically by simple right-hand rectangular integration with the same integration step. Right panel: the values of relative error, $e(t) = (\hat{y}(\tilde{\lambda}, t) - y(t))/\|y(t)\|_{\infty,[t_0,t_0+\infty]}$ as a function of t.

Trajectories $y(\cdot)$ and $\hat{y}(\tilde{\lambda}, \cdot)$ are shown in Figure 3.2. Notice that trajectories $y(\cdot)$ and $\hat{y}(\tilde{\lambda}, \cdot)$ nearly coincide with discrepancies of the order of 10^{-6} that are due to the differences in numerical integration. Here the representation of (3.50) is defined for x instead of using χ since there is not θ parameters defined in the system (3.45), implies that $\chi_0 = x_0$.

In order to illustrate how our method works for this class of systems let us suppose that $y(t_i) = x(t_i; t_0, (x_0, z_0), p),\ i = 0, 1, \cdots, N$ be the measured data,

p_1	p_2	p_4	p_5	p_6
1	1.3	1	3	0.1
0.2	0.2	0.2	1.0	0.01
0.99998	1.30178	0.99911	2.9966	0.1

TABLE 3.1: True (first row), Initial (second row), and Estimated (third row) parameter values of (3.45).

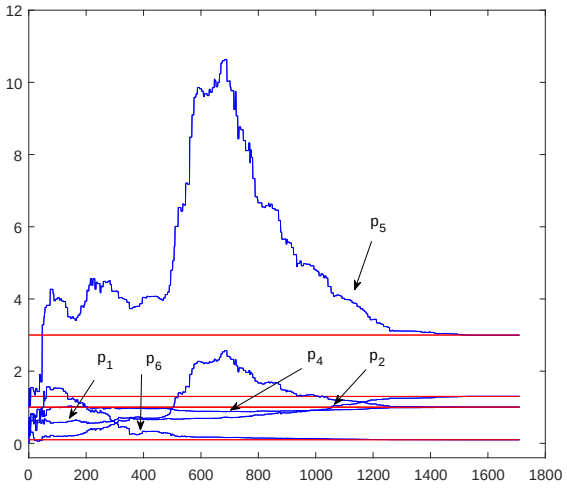


FIGURE 3.3: Estimates and true values of p_1, p_2, p_4, p_5, p_6.

and values of p_1, p_2, p_4, p_5, p_6 be unknown. For this particular simulation t_i formed an equispaced grid in $[t_0, t_0 + T]$ with $t_{i+1} - t_i = 0.001$ for all $i = 0, 1, \cdots, N-1$.

The values $y(t_i),\ i = 0, 1, \cdots, N$ were derived using the Runge Kutta 4th order method. As a measure of closeness between $y(\cdot)$ and $\hat{y}(\tilde{\lambda}, \cdot)$ we used the sum $\sum_{i=0}^{N}(y(t_i) - \hat{y}(\tilde{\lambda}, t_i))^2$, and as a parameter estimation routine we used the Nelder Mead algorithm [86]. Behavior of parameter estimates are shown in Figure 3.3, and their initial and final values are provided in Table 3.1. As one can see from the table and plots, after roughly 1710 steps the estimates are already in close proximity of true values of model parameters. This particular simulation took 27.229262 seconds in Matlab *R2015b*. Let us now consider another relevant model that will enable us not only to demonstrate existence of explicit integral representation of its solutions but also to illustrate the point that sometimes the overall number of unknown parameters can be reduced as a result of the proposed integral representation.

Integral representation (3.24)

CPU	GPU	Ratio
3.218992	0.000433	7434.16167

Runge Kutta integration

CPU	GPU	Ratio
6.408423	0.000003	2136141

TABLE 3.2: Time for 1000 evaluations of y.

In order to assess potential computational advantage of the proposed integral representations we compared the time needed for 1000 evaluations of $y(t)$ on CPU and GPU by CUDA programming over the interval $[t_0, t_0 + T]$ for 1000 randomly chosen parameter values. The results are summarized in Table 3.2 (upper table). CUDA with C++ is just one of the ways that we can create massively parallel computations. It uses the powerful C++ programming language to develop high performance algorithms accelerated by thousands of parallel threads running on GPUs. We see that GPU implementation of the same procedure resulted in the 39-fold performance gain. Our second set of experiments assessed the time needed for running 1000 Runge Kutta integrations of 4th order over the same period and for the same parameter values. The results are shown in Table 3.2. These experiments showed that explicit integral representations, if implemented on GPU, are approximately 14800 times faster than Runge Kutta integration on CPU. We also notice that 1000 model evaluations using Runge Kutta integration on GPU is approximately 144 times faster than the proposed integral implementations in this problem, but, on CPU, our representation of y is approximately 2 times faster Runge Kutta integration. Yet, our proposed scheme returns the estimates of all initial conditions and parameters that enter the right-hand side linearly (no θ parameters and one initial condition for this example). Furthermore, it enables to consolidate all computational power of the GPU into a single stream of computations which will be advantageous for local and inherently iterative optimization methods such as *e.g.* gradient-based search. In this regards comparing amount of time spent in integrating the corresponding sensitivity functions system with our explicit integral representation would be a fairer setting. This will be done in our future work.

3.4.2 Hodgkin-Huxley system

Consider the following ordinary differential equations of Hodgkin Huxley system which is describing the behaviour of nerve cells in a squid giant axon [48]:

$$\begin{aligned}
\dot{x} &= (1/C_{mc})(I - g_{Na}q_2^3 q_3(x - E_{Na}) - g_K q_1^4(x - E_K) - g_L(x - E_L)) \\
\dot{q}_1 &= (1 - q_1)\alpha_1(x) - \beta_1(x))q_1 \\
\dot{q}_2 &= (1 - q_2)\alpha_2(x) - \beta_2(x))q_2 \\
\dot{q}_3 &= (1 - q_3)\alpha_3(x) - \beta_3(x))q_3
\end{aligned} \tag{3.51}$$

where

$$\begin{aligned}
&\alpha_1(x) = 0.01(x + V_1)/(1 - e^{(-0.1(x+V_1))}),\ \beta_1(x) = 0.125e^{-(x+V_2)/80} \\
&\alpha_2(x) = 0.1(x + V_3)/(1 - e^{(-0.1(x+V_3))}),\ \beta_2(x) = 4e^{-(0.0556(x+V_4))} \\
&\alpha_3(x) = 0.07e^{(-0.05(x+V_5))},\ \beta_3(x) = 1/(1 + e^{-(0.1(x+V_6))})
\end{aligned} \tag{3.52}$$

where $x(t)$ is the measured voltage and $q_1(t), q_2(t), q_3(t)$ are the giant variables. This model describes the change of the voltage $x(t)$ in the electrical potential in the neuron cells over time depending on some parameters C_{mc}, E_K, E_{Na}, E_L, g_K, g_{Na}, I, g_L, V_1, V_2, V_3, V_4, V_5, V_6. Some of these parameters are assumed to be known: $C_{mc} = 1$, $E_K = -110.14$, $E_{Na} = 55.17$, $E_L = 49.49$; other parameters may vary from cell to another, thus are considered unknown.

The explicit solutions of the linear equations of (3.51) are in the following equations:

$$\begin{aligned}
q_i(t, \lambda, [y]) &= e^{-\int_{t_0}^{t}(\alpha_{q_i}(x(s))+\beta_{q_i}(x(s)))ds} q_i(t_0, \lambda, [y]) + \\
&\int_{t_0}^{t} e^{-\int_{z}^{t}(\alpha_{q_i}(x(s))+\beta_{q_i}(x(s)))ds} \alpha_{q_i}(x(z))dz \\
q_i(t_0, \lambda, [y]) &= (1 - e^{-\int_{t_0}^{t_0+T}(\alpha_{q_i}(x(s))+\beta_{q_i}(x(s)))ds})^{-1} \times \\
&\int_{t_0}^{t_0+T} e^{-\int_{z}^{t}(\alpha_{q_i}(x(s))+\beta_{q_i}(x(s)))ds} \alpha_{q_i}(x(z))dz
\end{aligned}$$

for all $i = 1, 2, 3$, where $T = 9.15$ is the period of oscillations. This brings the system (3.51) to be in the forms (3.10) and (3.8) with parameters $\lambda =$(g_K, g_{Na}, V_1, V_2, V_3, V_4, V_5, V_6) and $\theta = (x_0, g_L, I)$. In particular, the estimation of I is

not the value of θ_2 but rather is $(I - E_L g_L)$ and $g_L = -\theta_1$ where $g(t, \lambda, [y]) = g_{Na} m^3 h(x - E_{Na}) + g_K n^4 (x - E_K)$. T-periodic trajectories of $y(t)$ and $\hat{y}(\lambda, t)$ (by the observer (3.39)) are shown in Figure 3.4. The values of $\hat{y}(\lambda, t)$ have been computed from the representation (3.39) numerically by simple right hand side integration with the integration step size 0.0004, and $y(t)$ is numerically solved by the 4th order Runge-Kutta method with the same integration step at the nominal values of parameters.

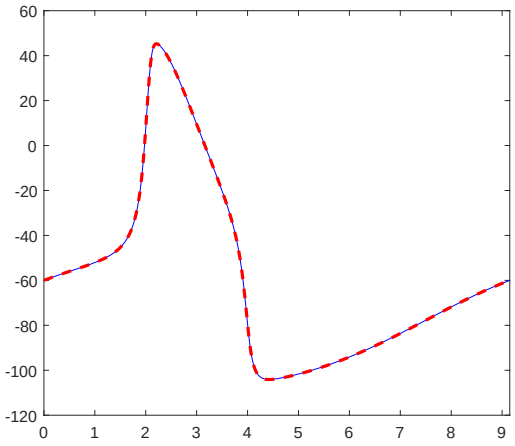


FIGURE 3.4: T-periodic trajectories of $y(t)$ (blue curve) and $\hat{y}(\lambda, t)$ (dashed red curve).

In order to evaluate $\hat{y}(\lambda, t)$ in Equation (3.39) as a function of parameters λ one we need to find the fundamental solution matrices $\Phi_A(t, t_0)$ of the homogeneous system $\dot{\chi} = A(y(t), t)\chi$ for all $t \in [t_0, t_0 + T]$ where $A(y(t), t) = \begin{pmatrix} F(t) & 1 & y(t) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $F(t) = 0$. We set $\delta = 2$ and used numerical estimates of the variable $R(t)$. The fundamental solution matrices $\Phi(t, t_0)$ of the homogeneous system (3.38) are computed for all $t \in [t_0, t_0 + T]$. These matrices were constructed numerically using the improved Euler method with step size 0.0004 from linearly independent solutions of the homogeneous systems $\dot{z}_1 = A(y(t), t) z_1$ and $\dot{z}_2 = (A(y(t), t) - R^{-1}C)\, z_2$ (starting from $(1, 0, 0), (0, 1, 0), (0, 0, 1)$). The value of $R(t_0)$ in (3.29) was chosen to be the unique solution of the Sylvester equation (3.35) (see Lemma 3.5).

The parameterized representations (3.39) and (3.24) were later used, in combination with the Nelder-Mead algorithm to recover the values of parameters λ and θ. Results are provided in Table 3.3 and Figure 3.6 for parameters. The

Vector $\lambda = (g_K, g_{Na}, V_1, V_2, V_3, V_4, V_5, V_6)$

g_K	g_{Na}	$V1$	$V2$	$V3$	$V4$	$V5$	$V6$
36	120	50	60	35	60	60	30
36.00115	119.68690	49.99090	60.02785	35.00185	59.84273	59.99556	30.00437
36	120	50	60	35	60	60	30

x_0 and vector $\theta = (g_L, I)$

x_0	g_L	I
-60	0.3	0.1
-59.99236	0.29849	0.27877
-60.00067	0.2999493	0.1018595

TABLE 3.3: True (first row) and Estimated (second row and third row) of λ and θ, and the initial value x_0 by the representations (3.24) and (3.39), respectively.

time of calculations and the number of iterations are shown in Tables 3.4 and 3.5.

Equation	(3.39)	(3.24)
Time(minute)	18	21
Iterations	14352	18244

TABLE 3.4: Time and number of iterations on a standard PC in Matlab R2015a.

Equation	(3.39)	(3.24)
Time(second)	12	45
Iterations	2790	9690

TABLE 3.5: Time and number of iterations on GPU.

However, to achieve these results it was required to select different initial points to start the estimation. We selected the initial points (10, 50, 10, 20, 10, 20, 35, 15), $(22, 108, 30, 45, 25, 42, 45, 12)$ on CPU and $(20, 109, 35, 49, 26, 47, 43, 20)$, (26, 102, 40, 50, 25, 50, 50, 20) for the requirements of evaluating y by the representations (3.39) and (3.24), receptively.

Vector $\lambda = (g_K, g_{Na}, V_1, V_2, V_3, V_4, V_5, V_6)$

g_K	g_{Na}	$V1$	$V2$	$V3$	$V4$	$V5$	$V6$
20	109	35	49	26	47	43	20
36.04396	120.75294	50.02992	59.76529	34.98947	60.02058	60.22761	30.05980

x_0 and vector $\theta = (g_L, I)$

x_0	g_L	I
-59.988889	0.341330	-4.628210

TABLE 3.6: Initial (first row) and Estimated (second row) values of λ in the above table by the representation (3.24). The estimated values of θ, and the initial value x_0 are in the below table. The least square error at the estimates is 0.0009204794.

Vector $\lambda = (g_K, g_{Na}, V_1, V_2, V_3, V_4, V_5, V_6)$

g_K	g_{Na}	$V1$	$V2$	$V3$	$V4$	$V5$	$V6$
26	102	40	50	25	50	50	20
36.00115	118.91595	49.99954	60.00571	35.00677	59.98465	59.42014	30.02750

x_0 and vector $\theta = (g_L, I)$

x_0	g_L	I
-60.000684	0.300011	0.102571

TABLE 3.7: Initial (first row) and Estimated (second row) values of λ in the above table by the representation (3.39). The estimated values of θ, and the initial value x_0 are in the below table. The least square error at these estimated values is 0.01025706.

The least square error $\sum_{i=0}^{N}(y(t_i)-\hat{y}(\lambda,t_i))^2$ was measured between $y(t)$ and $\hat{y}(\lambda,t)$. Since there is deferences in numerical integration we notice that trajectories of $y(t)$ and $\hat{y}(\lambda,t)$ are nearly coincide with maximum relative error $e(t) = (\hat{y}(\lambda,t)-y(t))/\|y\|_{\infty,[t_0,t_0+\infty]}$ approximately 10^{-4} as it is shown in right panel of Figure 3.5. Nelder-Mead algorithm [86] played the role of estimation and the values of reflection, expansion, contraction and shrinking parameters were set to $1, 2, 0.5$ and 0.5, respectively. In order to assess potential computational advantage of the proposed integral form of equation (3.39) we compared the time needed for 1000 evaluations of $y(t)$ on CPU by (3.39) and improved Euler integration over the interval $[t_0, t_0+T]$. The total time of running 1000 times is accounted for the same values of chosen parameter values for both of the integration forms. The results are shown in Table 3.8. This experiment shows that the integral presentation of equation (3.39) on CPU is approximately 10 times faster than improved Euler integration.

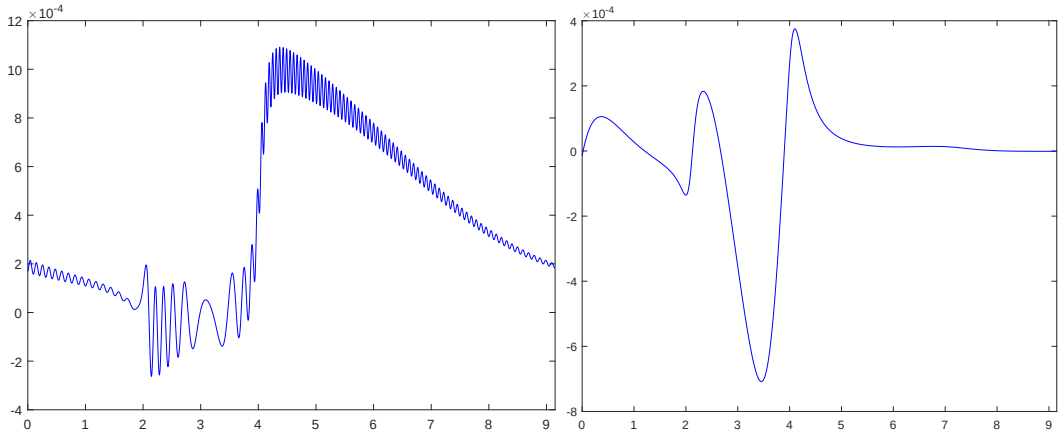


FIGURE 3.5: The values of relative error $e(t) = (\hat{y}(\lambda, t) - y(t))/\|y\|_{\infty,[t_0,t_0+\infty]}$ as a function of t by the representations (3.24)(left panel) and (3.39)(right panel).

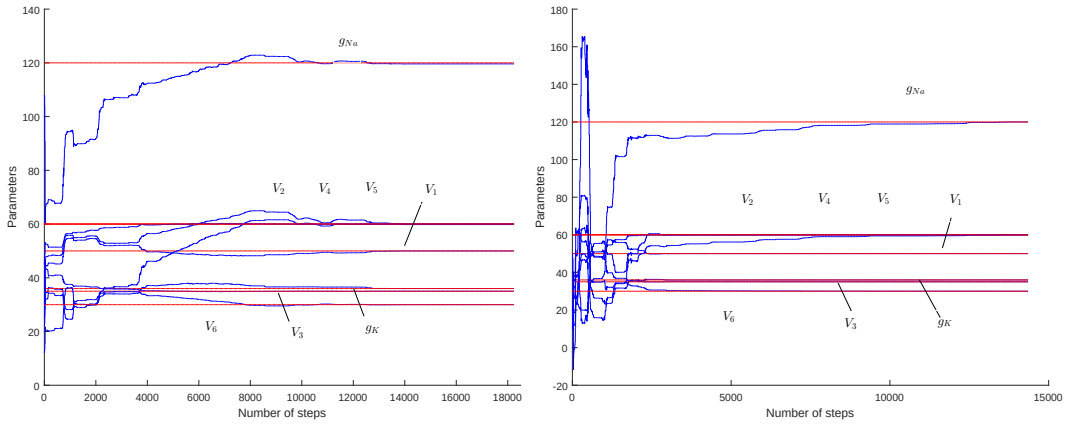


FIGURE 3.6: Estimated (blue) and true (red) values of the parameters g_K, g_{Na}, V_1, V_2, V_3, V_4, V_5, V_6.

Eq. (3.39)	Improved Euler method	Ratio
1.13308 minutes	8.80552 minutes	9.977401

TABLE 3.8: Time for 1000 evaluations of y

3.4.3 Morris-Lecar system

Consider the following simple point model of neural membrane activity [84]:

$$\begin{aligned} \dot{x} &= g_{Ca} m_\infty(x)(x - E_{Ca}) + g_K q(V + E_K) + g_L(x + E_L) + I \\ \dot{q} &= \frac{-1}{\tau(x)} q + \frac{\omega_\infty(x)}{\tau(x)} \end{aligned} \tag{3.53}$$

$$y = x,$$

$$
\begin{aligned}
m_\infty(x) &= 0.5\left(1+\tanh\left(\frac{x-V_1}{V_2}\right)\right)\\
\omega_\infty(x) &= 0.5\left(1+\tanh\left(\frac{x+V_3}{V_4}\right)\right)\\
\tau(x) &= T_0/\left(\cosh\left(\frac{x+V_3}{2V_4}\right)\right).
\end{aligned}
\tag{3.54}
$$

Here x is the measured voltage, q is the recovery variable. Parameters E_{Ca}, E_K, E_L are the Nernst potentials of which the nominal values are assumed to be *known*: $E_{Ca} = 55.17, E_K = -110.14, E_L = 49.49$; other parameters may vary from one cell to another and thus are considered *unknown*. Assume that the model operates in the oscillatory regime which corresponds to periodic solutions of (3.53). For practically relevant values of T_0, V_3, V_4 the integral

$$\int_{t_0}^{t_0+T} -\frac{1}{\tau(s)}ds < 0$$

where T is the period of oscillations. Given that $x(\cdot)$ is T-periodic, the variable q can be expressed as:

$$
\begin{aligned}
q(t) &= e^{\int_{t_0}^{t} -\frac{1}{\tau(x(s))}ds} q_0 + \int_{t_0}^{t} e^{\int_{z}^{t} -\frac{1}{\tau(x(s))}ds} \frac{\omega_\infty(x(z))}{\tau(x(z))}dz\\
q_0 &= \left(1 - e^{\int_{t_0}^{t_0+T} -\frac{1}{\tau(x(s))}ds}\right)^{-1} \int_{t_0}^{t_0+T} e^{\int_{z}^{t_0+T} -\frac{1}{\tau(x(s))}ds} \frac{\omega_\infty(x(z))}{\tau(x(z))}dz.
\end{aligned}
$$

This leads the equations of (3.51) to be in the forms (3.10) and (3.8). Denoting $g(t,\lambda,[y]) = g_{Ca}m_\infty(x)(x-E_{Ca}) + g_K q(x+E_K)$, $\Psi(t,y) = (y(t),1)$, and combining parameters as $\theta = (g_L, I)$, $\lambda = (V_1, V_2, V_3, V_4, T_0, g_{Ca}, g_K)$ we can rewrite (3.53) in the form of equation (3.11) with

$$A(y(t),t) = \begin{pmatrix} 0 & y(t) & 1\\ 0 & 0 & 0\\ 0 & 0 & 0 \end{pmatrix}.$$

For this system and given nominal parameter values, the period of oscillations is $T = 15.169$, and hence for convenience the integration interval is chosen as $[0, 15.169]$. In what follows, numerical evaluation of integrals and solutions of all auxiliary differential equations was performed on equi-spaced grids with the step

size of 0.0004. Figure 3.7 shows T-periodic trajectories of $y(t)$ and $\hat{y}(\lambda, t)$ by the observer (3.39). The estimation of I is not the value of θ_2 but rather is the sum $(I + E_L g_L)$ and $g_L = \theta_1$ for the both observers (3.39) and (3.24).

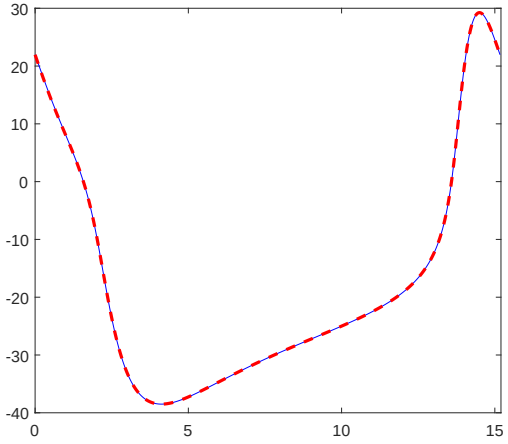


FIGURE 3.7: T-periodic trajectories of $y(t)$ (blue curve) and $\hat{y}(\lambda, t)$ (dashed red curve).

The parameterized representations (3.39) and (3.24) were later used, in combination with the Nelder-Mead algorithm [86] to recover the values of parameters λ and θ. Results are provided in Table 3.10 and Figure 3.9 for parameters. The time of calculations and number of iterations are shown in Table 3.9.

We selected the initial points $(0, 0, 0, -4, 4.5, -0.5, 0)$, $(-1, 0.01, 0.01, -8, 11, -1.7, 2.4)$ for the requirements of evaluating y by the representations (3.39) and (3.24), receptively.

Equation	(3.39)	(3.24)
Time(minute)	8	46
Iterations	2661	12775

TABLE 3.9: Time and number of iterations on a standard PC in Matlab R2015a.

According to Theorem 3.6, explicit parameter-dependent representation of the observed quantity, $\hat{y}(\lambda, t)$, is defined by (3.39), where $C = (1, 0, 0)$, $\chi = col(x, \theta)$, and the fundamental solution (3×3)-matrices $\Phi(t, t_0)$ and $\Phi_A(t, t_0)$ are computed for the linear systems $\dot{\chi} = (A(y(t), t) - R^{-1}(t)CC^T)\chi$, $\dot{R} = -\delta R - A(y(t), t)^T R - RA(y(t), t) + CC^T$, and $\dot{\chi} = A(y(t), t)\chi$, respectively, by the Improved Euler method for $t \in [0, 15.169]$. The value of δ was set as $\delta = 2$, and numerical approximations of matrices $\Phi_A(t, t_0)$ were used to compute the matrices

$R(t)$ in accordance with equation (3.29). The value of $R(t_0)$ in (3.29) was chosen to be the unique solution of the Sylvester equation (3.35) (see Lemma 3.5).

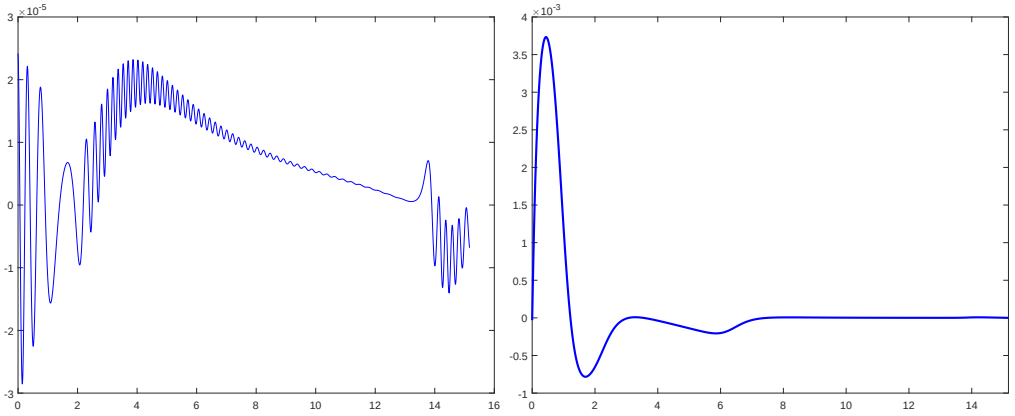


FIGURE 3.8: The values of relative error $e(t) = (\hat{y}(\lambda, t) - y(t))/\|y\|_{\infty,[t_0,t_0+\infty]}$ as a function of t by the representations (3.24)(left panel) and (3.39)(right panel).

We noticed that the estimating by using (3.24) required to start with an initial point closed enough to the real values of unknown parameters. The results are shown in Table 3.10(second row) and the Figure 3.9(left panel).

Figure 3.8 shows the relative errors, $e(t) = (\hat{y}(\lambda, t) - y(t))/\|y\|_{\infty,[t_0,t_0+\infty]}$, between the proposed numerical representations (3.39), (3.24) and simulated $y(t)$ (Runge-Kutta, step size 0.0004) for nominal parameter values. Since there is deferences in numerical integration we notice that trajectories of $y(t)$ and $\hat{y}(\lambda, t)$ are nearly coincide with maximum relative error $e(t) = (\hat{y}(\lambda, t) - y(t))/\|y\|_{\infty,[t_0,t_0+\infty]}$ with approximately 10^{-5} as shown in Figure 3.8(right panel).

Vector $\lambda = (g_K, g_{Ca}, V_1, V_2, V_3, V_4, T_0)$

g_K	g_{Ca}	V_1	V_2	V_3	V_4	T_0
-2	-1.1	-1	15	-10	14.5	3
-1.75755	-1.08292	-0.98639	14.72194	-8.03302	14.53612	2.64769
-2	-1.1	-1	15	-10.00000	14.50000	3.00000

x_0 and vector $\theta = (g_L, I)$

x_0	g_L	I
21.96388	-0.5	10
21.96641	-0.44589	9.00839
21.96477	-0.49982	9.99345

TABLE 3.10: True (first row) and Estimated (second row and third row) of λ and θ, and the initial value x_0 by the representations (3.39) and (3.24), respectively.

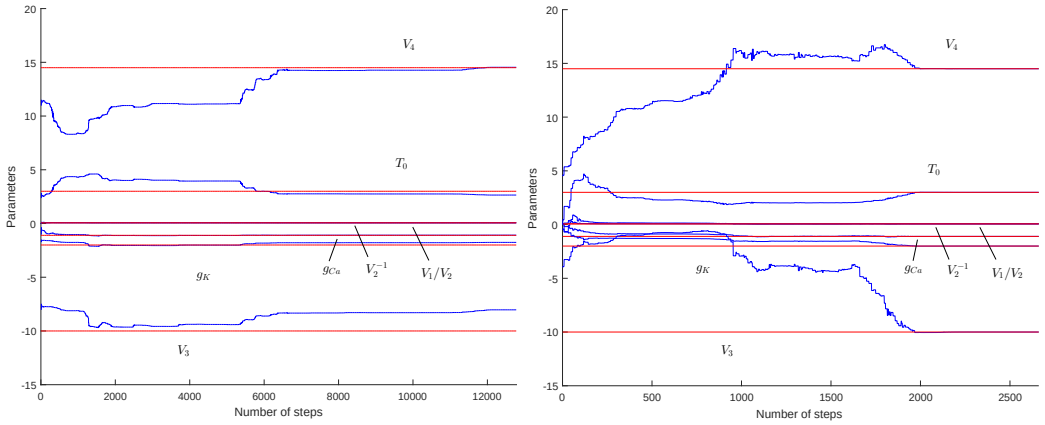


FIGURE 3.9: Estimated(blue) and true(red) values of the parameters g_K, g_{Ca}, V_1, V_2, V_3, V_4, T_0.

Eq. (3.39)	Improved Euler method	Ratio
2.21311 minutes	10.43818 minutes	4.71652

TABLE 3.11: Time for 1000 evaluations of y

We compared the time required for 1000 evaluations of $y(t)$ in Matlab expressed as in (3.39) and then computed by the Improved Euler method over the interval $[t_0, t_0 + T]$. The results are summarized in Table 3.11. This experiment shows that evaluation of the proposed representation, (3.39), in Matlab on CPU is approximately and on average 5 times faster than the Improved Euler integration.

Chapter 4

Approximating periodic solutions of linear Integral Equations Based on the RBFs

Representations (3.11) and (3.10) involve integrals of linear ODEs of system (3.11). Straightforward, evaluating of these integrals, albeit scalable, could still be computationally demanding.

Computational improvements might be possible and are practically viable if certain variables in the representations are replaced by their reasonable sparse Radial Basis Function approximations.

One of the key steps justifying incorporation of relevant class of equations specified by (3.8) into the setting focusing on (3.6) was an assumption that the variable $q(t; q_0, \lambda, y)$ is expressible as a known function of parameters, initial conditions, and t. For example, if q relates to a single first-order equation then such function can be computed as follows:

$$\begin{aligned} q(t; q_0, \lambda, y) = e^{\int_{t_0}^{t} \upsilon(y(\tau),\lambda,\tau)d\tau} q_0 + e^{\int_{t_0}^{t} \upsilon(y(\tau),\lambda,\tau)d\tau} \times \\ \int_{t_0}^{t} e^{-\int_{t_0}^{\tau} \upsilon(y(s),\lambda,s)ds} \omega(y(\tau),\lambda,\tau)d\tau \\ q_0 = (1 - e^{-\int_{t_0}^{t_0+T} \upsilon(y(s),\lambda,s)ds})^{-1} \int_{t_0}^{t_0+T} e^{-\int_{z}^{t_0+T} \upsilon(y(s),\lambda,s)ds} \omega(y(z),\lambda,z)dz. \end{aligned} \tag{4.1}$$

If the original problem is governed by (3.8) then availability of $q(t; q_0, \lambda, y)$ is required in our explicit parameter-dependent representation proposed, as explained in Chapter 3. One way to resolve the problem is to numerically evaluate all integrals involved. This, however, may not always be optimal. An alternative could

be to use computationally efficient approximations of $q(t; q_0, \lambda, y)$ instead.

A possible way to derive such approximations is to approximate the function using some "well- behaved" approximants. "Well-behaved" here is understood as a property that the derivatives of the approximating function could be controlled in some sense.

A conventional approximation task is to construct a function $\hat{q}(t; q_0, \lambda, y)$ which coincides in some sense with given measurements (data) at the corresponding locations (data sites). Normally an approximant is a continuous mapping from $\mathbb{R}^d$ to $\mathbb{R}$ (d is the dimension of the data). If the region on which the data sites from a uniform or a regular grid then the process is called grid or mesh data approximation, otherwise, it's called scattered data approximation.

To better control/regulate behaviour of the function between nodes one may relax the approximation requirement at all or some nodes, and allow for some errors there. We will discus this later in the chapter. Sections 4.1–4.3 introduce the radial basis function (RBF) approximation of scattered data points. Section 4.4 presents a new algorithm for evaluating $q(t, \lambda)$ with their RBF approximations in a generic optimization routine for inferring the values of θ and λ. In the last section the method is illustrated with an example involving the Morris–Lecar equations.

4.1 Scattered Data Approximation Problem

Definition 4.1.1. (Scattered Data approximation) Let $X = \{X_1, X_2, \cdots, X_N\} \subset \mathbb{R}^d$, for those points, given pairwise data $(X_i; q(X_i)), i = 1, \cdots, N$, where $q : \mathbb{R}^d \to \mathbb{R}$. The scattered data approximation problem is to find a function $S_q : \mathbb{R}^d \to \mathbb{R}$ such that $S_q(X_i) = q(X_i)$, for all or some $i = 1, \cdots, N$, where S_q is called the approximant to the data. If $S_q(X_i) = q(X_i)$ for all $i = 1, \cdots, N$, then the problem is referred to as the scattered data interpolation.

Here $X_i, i = 1, \cdots, N$ are the measurements locations, and $q(X_i)$ are the corresponding target function values at these locations, and the data set is hence the pairs $(X_i, q(X_i))$. A convenient approach for solving the scattered data approximation problem is assuming that the approximant S_q is a linear combination

of certain functions, $\phi_k(X)$[1], $k = 1, \cdots, N$, i.e.,

$$S_q(X) = \sum_{k=1}^{N} \omega_k \phi_k(X), \ X \in \mathbb{R}^d. \tag{4.2}$$

Solving *e.g.* the interpolation problem under this assumption, requires that

$$S_q(X_i) = q(X_i); i = 1, \cdots, N \tag{4.3}$$

holds. Alternatively

$$A\omega = q, \tag{4.4}$$

where the entries of the matrix A are given by $A_{i,j} = \phi_j(X_i)$ where $j, i = 1, \cdots, N$ and $\omega = [\omega_1, \cdots, \omega_N]$, $q = [q_1, \cdots, q_N]$.

If the matrix A is non singular, the unique solution of the problem exists. The non-singularity of the matrix A is guaranteed under some mild restrictions i.e., constant shape parameters and usually by adding a low order polynomial $P(X)$[81]: as *e.g.*,

$$S_q(X) = \sum_{k=1}^{N} \omega_k \phi_k(X) + P(X), \ X \in \mathbb{R}^d. \tag{4.5}$$

4.2 Radial Basis Function and Approximation Principle

Radial basis functions (RBFs) are traditional and powerful tools for the meshless interpolation and approximation of scattered data. These functions are real-valued functions which depend only on the distance from the fixed center point. More precisely, let us consider an univariate function:

$$\phi : [0, \infty] \to \mathbb{R}. \tag{4.6}$$

[1]Basis functions are elements of a particular basis for a function space such that Radial basis, polynomial basis and Fourier basis, where every continuous function in the function space can be represented as a linear combination of basis functions.

Then the radial basis functions $\Phi_c : \mathbb{R}^d \to \mathbb{R}$ are defined as:

$$\Phi_c(X) = \phi(r) = \phi(\|X - X_c\|), \tag{4.7}$$

where $X_c = \{X_{c_1}, X_{c_k}, \cdots, X_{c_M}\} \subset \mathbb{R}^d$ is a set of M different points which called centers and $\|\cdot\|$ is some norm on $\mathbb{R}^d$ (typically the Euclidean norm). For this finite set of centers, Φ_c can be defined at any point $X \in \mathbb{R}^d$ as constant at any given distance from the centers, *i.e.*

$$\|X_1\| = \|X_2\| \;\Rightarrow\; \Phi_c(X_1) = \Phi_c(X_2),\; X_1, X_2 \in \mathbb{R}^d. \tag{4.8}$$

Thus, Φ_c is radially (or spherically) symmetric around its center. Example of a such function can be seen in Figure 4.1.

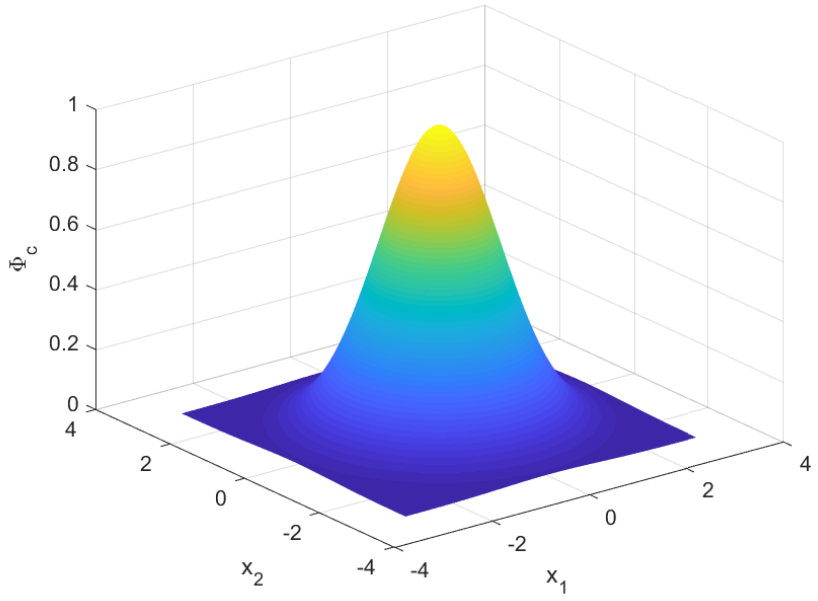


FIGURE 4.1: Example of RBF

The function ϕ can be termed as a basic function whereas Φ_c as a basis function, where one single basic function generates all of the basis functions which we have used in the expansion (4.2)[23]. Radial functions become especially useful for applications as the approximation problem becomes insensitive to the dimension d of the space wherein the data sites are found [117]; the same univariate function ϕ for all choices of d can be utilised, rather than having to work with the

multivariate function ϕ (whose complexity grows together with the growing space dimension d).

Table 4.1 presents some widely utilised globally supported radial basis functions.

Infinitely smooth RBFs	Functional Form, $\phi(r)$	Parameters
Plyharmonic Spline	r^k	$k > 0, k \notin 2\mathbb{N}$
Gaussian	$e^{-(\alpha r)^2}$	$\alpha > 0$
Multiquadric(MQ)	$(1 + \alpha^2 r^2)^{k/2}$	$k > 0, k \notin 2\mathbb{N}, \alpha > 0$
Inverse multiquadric	$(1 + \alpha^2 r^2)^{k/2}$	$k < 0, k \notin 2\mathbb{N}, \alpha > 0$

TABLE 4.1: Some commonly used radial basis functions.

Since the Gaussian has comparatively high accuracy and is infinitely differentiable, many authors have a preference to utilise this in the literature and show the accuracy, stability and ease of implementation , for example, [33, 49, 89]. In this chapter, Gaussian function is regarded as our basic function too. The Gaussian function is defined as follows:

$$\phi(\|X - X_c\|_2) = e^{-(\alpha\|X - X_c\|)^2}, \tag{4.9}$$

where α, the shape parameter, has a huge influence on the approximation quality. For the above presentations, different choices of shape parameters can lead to different shapes of RBFs, from peak to flat.

Let $X = (t, \lambda) \in \mathbb{R}^d$ be a vector accommodating relevant measurement parameters, i.e. t and λ. The centres X_c could be selected from the given data samples or derived via clustering algorithms.

Let

$$S_q(X) = \sum_{j=1}^{M} \omega_j \phi(\|X - X_{c_j}\|), \ X \in \mathbb{R}^d \tag{4.10}$$

be an RBF approximation of $q(t; q_0, \lambda, y)$ or simply $q(t, \lambda)$, where ω_j are unknown coefficients that need to be determined. It is well-known that, for a broad range of $\phi(\cdot)$, any continuous function on a bounded domain can be approximated by sums (4.10) with arbitrary accuracy in $L_{\mathbf{p}}$-norm, $\mathbf{p} > 1$, subject to the choice of parameters X_{c_j}, ω_j, and M [87].

The coefficients, ω, are chosen by enforcing the interpolation condition (4.3) at a set of nodes that typically coincides with the centres. Enforcing the interpolation condition at M centres results in a $N \times M$ linear system

$$
\begin{aligned}
&A\omega = q, \\
&\begin{pmatrix}
\phi(\|X_1 - X_{c_1}\|_2) & \cdots & \phi(\|X_1 - X_{c_M}\|_2) & \\
\phi(\|X_2 - X_{c_1}\|_2) & \cdots & \phi(\|X_2 - X_{c_M}\|_2) & \\
\vdots & \vdots & \vdots & \vdots \\
\phi(\|X_N - X_{c_1}\|_2) & \cdots & \phi(\|X_N - X_{c_M}\|_2) &
\end{pmatrix}
\begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_M \end{pmatrix}
= \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{pmatrix},
\end{aligned} \tag{4.11}
$$

where the number of rows is $N \ll M$. The $N \times M$ matrix A is called the interpolation matrix or the system matrix and consists of constants serving as the basis of the approximation space.

The linear system of equations (4.11) can be solved by the Gauss elimination method or the LU decomposition, but when the number of data samples exceeds the number of sites, the approximation problem can be cast as the unconstrained quadratic programming problem:

$$\min \|A\omega - q\|_2^2, \tag{4.12}$$

where the objective function represents the L_2-norm of the linear system (4.11) and its solution is:

$$\omega = (A^T A)^{-1} A^T q. \tag{4.13}$$

Practically, the quadratic error functionals demonstrated many weaknesses including high sensitivity to error/outliers and the curse of dimensionality. To remedy these L_2-norm is replaced with L_1-norms or fractional norms $L_{\mathbf{p}}(0 < \mathbf{p} < 1)$. This generalizes the optimization problem (4.12) to

$$\min \|A\omega - q\|_{\mathbf{p}} \tag{4.14}$$

for $\mathbf{p} = 1$ or $0 < \mathbf{p} < 1$. One of the drawbacks of these approaches is an increase in computational costs for optimization and loss of complexity. In 2016, Gorban *et al* [41] developed a theory and basic universal data approximation algorithms

(K-means, principal components, principal manifolds and graphs, regularized and sparse regression), based on piece-wise quadratic error potentials of subquadratic growth (PQSQ potentials). The development is presented by implementing a new and universal framework to minimize arbitrary sub-quadratic error potentials using an algorithm with guaranteed fast convergence to the local or global error minimum. Thus solving (4.14) breaks down to a sequence of quadratic problems (4.12). This enables us to consider (4.12) as a suitable problem for the purposes of our case-study in which we focus on computational efficiency.

As mentioned earlier, the shape of the RBF depends on the shape parameter α. When the shape parameter is too large or too small (which can not be easily quantified in practice), the results lose credibility. For α large the system becomes too ill-conditioned and hence unstable. Furthermore, even though a low value of α ($\alpha \longrightarrow 0$) can offer a well-conditioned linear system, the resultant solution is also inaccurate, as affirmed by Driscoll and Fornberg in [28]. Such observations possibly indicate that there should be a reliable region for their shape parameters for different RBFs with shape parameters. In 1995, Schaback [96] stated that a balancing point between accuracy and good condition exists, but both cannot be guaranteed. Later, applying the Contour-Padé algorithm [36] and the RBF-QR method [35] was proposed for a more stable management of flatter RBFs. Two significant developments then occurred: Fasshauer and Mccourt's [32] introduced a stable method with flat Gaussian kernels and Fornberg, Larsson and Flyer's [34] extended the RBF-QR approach to three dimensions. Recently, in [15] a Variably Scaled Kernels (VSK) method in order to achieve a reduction of the condition number by treating the shape parameter as an extra space variable was put forward. To date, the optimal shape parameter remains an unsolved problem in RBF research.

To select x_cs when the member of data sites is large, one can use clustering algorithms such as *e.g.* K–Means. Its basic description for consistency is provided in Section 4.3.

4.3 K-Means Clustering Algorithm

Clustering involves the task of dividing data points into homogeneous clusters so that items in the same class are as similar as possible and items in different classes are as dissimilar as possible. Clustering can also be thought of as a form of data compression, where a large number of samples are converted into a small number of representative clusters. A loose definition of clustering could be the process of organizing objects into groups whose members are similar in some way. A cluster is therefore a collection of objects which are 'similar' between them and are 'dissimilar' to the objects belonging to other clusters [112, 85].

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem [45]. The procedure of the K-means algorithm follows a simple and an easy way to classify a given data set through a certain number of clusters (assume K clusters) fixed a priori. The main idea is to define K centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no points remain, the first step is completed. At this point we need to re-calculate K new centroids for the clusters resulting from the previous step.

Algorithm 1 K-means clustering

1. pick the number M of centres and randomly assign the data points t_{c_p} to M subsets.
2. Use the procedure of minimizing the sum squared clustering function

$$J = \sum_{i=1}^{M} \sum_{p \in S_j} \|t_{c_p} - \mu_j\|_2^2 \tag{4.15}$$

to end up with a partition of the data points into M disjoint sub-sets or clusters S_j containing N_j data points, where $\mu_j = \frac{1}{N_j} \sum_{p \in S_j} t_{c_p}$, $j = 1, \cdots, M$.

Algorithm 1 shows the main process of finding the centre points. The algorithm calculates the nearest mean μ_j to each data point t_{c_p}, reassigns the

data points to the associated clusters S_j, and then recomputes the cluster means μ_j. The clustering process terminates when no more data points switch from one cluster to another. Multiple runs can be carried out to find the local minimum with lowest J.

For further information on clustering and clustering algorithms, see [57, 19, 53, 52].

4.4 Parameter Inference with Approximated Variables of Linear Equations by The Radial Basis Approximation

The method works with points scattered throughout the domain of interest, and the RBF approximation of q is a linear combination of RBFs centred at the scattered data points of X as follow:

$$S_q(X) = \sum_{j=1}^{M} \omega_j e^{-(\alpha \|X - X_{c_j}\|_2)^2}, \tag{4.16}$$

where $X_{c_j} = (t_{c_j}, \lambda_{c_j}), j = 1, \cdots, M$ and λ_{c_j} are the M points of the parameters λ.

The coefficients ω are determined by Equation (4.13) and then the approximant (4.16) is evaluated:

$$S_q(t_i, \lambda) = \sum_{j=1}^{M} \omega_j e^{-(\alpha \|(t_i - t_{c_j}, \lambda - \lambda_{c_j}\|_2)^2} \tag{4.17}$$

for all $i = 1, \cdots, N$.

To calculate M, k values of the m parameters, λ, are randomly selected, implying $M = k^m$; therefore M clusters for (t, λ) are formed.

The identification procedure of the RBF includes estimating all or some of parameters of the system. Using a nonlinear parameter optimization algorithm to estimate parameters would not involve a large amount of computation because it never invokes integrals specified by Equation (4.1) even when the number of

the unknown parameters is large. The following heuristics is proposed to replace repeat evaluations (4.1) of $q(t, \lambda)$ with their RBF approximations in a generic optimization routine for inferring the values of θ and λ.

Algorithm 2 Parameter inference with approximated variables

1. Initialization: set $\hat{\lambda}$ as an initial guess of λ.
2. A set of M samples $X_i = (t_{n_i}, \lambda_{m_i})$ is randomly drawn from a relevant domain. The domain, in general, may depend on $\hat{\lambda}$. The values of t_{n_i}, λ_{m_i} are taken from their relevant grids.
3. Group spatially close points using a suitable clustering algorithm (e.g. [57, 19, 53, 52]), and set the centers X_{cj} as the centres of these clusters.
4. Determine parameters ω_j in (4.10) as the minimizer of $\sum_{i=1}^{N}(S(X_i) - q(t_i, \lambda_i))^2$, $N > 0$. Note that adjustments of the shape parameter, α, might be needed to ensure good approximation.
5. Using representation (3.39) and approximant (4.10) define:

$$\begin{array}{l} \tilde{y}(\hat{\lambda}, t) = F(t, t_0, \theta, \hat{\lambda}, \hat{q}(\hat{\lambda}, t)) \\ \hat{q}(\hat{\lambda}, t) = \sum_{k=1}^{M} \omega_k \phi(\|(t, \hat{\lambda}) - (t_{c_k}, \lambda_{c_k})\|). \end{array} \tag{4.18}$$

   The function $\tilde{y}(\hat{\lambda}, t)$ is an approximation of $\hat{y}(\hat{\lambda}, t)$.
6. Use $\tilde{y}(\hat{\lambda}, t)$, to produce a refined guess of $\hat{\lambda}$ and return to Step 1 if required.

In order to study the efficiency of the algorithm with regard to grids of the points for t and λ, it will be useful to compare the results with that for the representation $\hat{y}(\hat{\lambda}, t)$ in Equation (3.39) defined in chapter 3.

In the next section we illustrate an application the method with Algorithm 2 to the problem of parameter estimation for the Morris-Lecar system.

4.5 Experimental Results of RBF Approximation

To show feasibility of RBF approximations in this problem we repeated the experiment for the Morris–Lecar system but this time with the variable q replaced with its RBF approximation inside the optimization routine (Nelder-Mead). To produce such approximations we followed steps of Algorithm 2.

Recall that the Morris–Lecar equation is:

$$\begin{aligned} \dot{x} &= g_{Ca} m_\infty(x)(x - E_{Ca}) + g_K q(V + E_K) + g_L(x + E_L) + I \\ \dot{q} &= \frac{-1}{\tau(x)} q + \frac{\omega_\infty(x)}{\tau(x)} \end{aligned} \tag{4.19}$$

where

$$\begin{aligned} m_\infty(x) &= 0.5\left(1 + \tanh\left(\frac{x - V_1}{V_2}\right)\right) \\ \omega_\infty(x) &= 0.5\left(1 + \tanh\left(\frac{x + V_3}{V_4}\right)\right) \\ \tau(x) &= T_0 / \left(\cosh\left(\frac{x + V_3}{2V_4}\right)\right). \end{aligned} \tag{4.20}$$

Parameters E_{Ca}, E_K, E_L are the Nernst potentials of which the nominal values are assumed to be *known*: $E_{Ca} = 55.17, E_K = -110.14, E_L = 49.49$; other parameters may vary from one cell to another and thus are considered *unknown*.

The variables q and x, numerical evaluations of integrals and numerical solutions, are generated by the scheme of $4th$ order Runge–Kutta method with step size $\delta t = 0.002$. This data could be considered as an actual solution of the system for comparison purposes.

Note that the variable q depends only on 3 components of the vector λ, i.e. T_0, V_3, and V_4. And hence all steps of the algorithm related to approximation apply to these 3 relevant components and the variable t only, which results $M = 8$. We considered an extremely sparse setting, in which each of the three parameters have been sampled at 2 points per each relevant sample of t. The values of t where chosen from the grid of 0.002-spaced points in $[0, 15.1692]$ ($N = 7586$ points in the grid). The shape parameter α was set to 0.2.

In Table 4.2, for different number of samples of parameters, we check how well $S_q(t_i, \lambda)$ approximates $q(t_i, \lambda)$ as a function of t_i at the same sample of parameters using the following simple criterion:

$$LS = \sum_{i=1}^{N} (q(t_i, \lambda) - S_q(t_i, \lambda))^2. \tag{4.21}$$

Samples	M	Least Square Error	Time(seconds)
2	8	0.53065	0.25031
4	64	0.31278	1.1116
7	343	0.31417	4.54909
10	1000	0.39148	14.19380
14	2744	0.14929	37.04459
18	5832	0.17759	87.18906

TABLE 4.2: The least square error $LS = \sum_{i=1}^{N}(q(t_i,\lambda) - S_q(t_i,\lambda)))^2$ and the consumed time of RBFs interpolating for different number of samples of parameters.

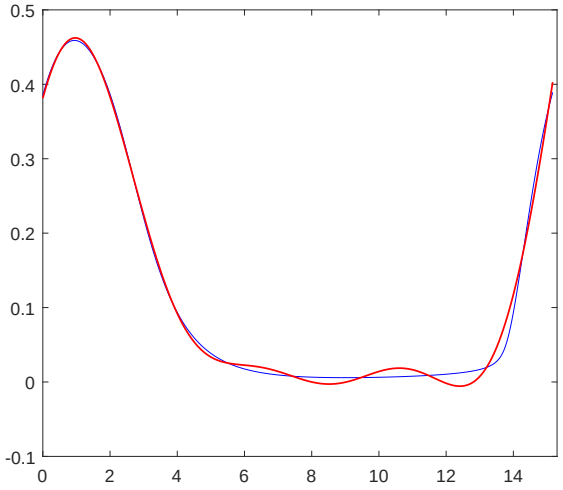


FIGURE 4.2: The data curve (blue) and the fitted curve (red) of q by the interpolation (4.16).

In order to judge the efficiency of the approach we run the algorithm 1000 times and recorded empirical errors between λ_i and their estimates $\hat{\lambda}_i$, and computed their L_2 distances as:

$$d_1(\nu) = \sqrt{\sum_{i=1}^{7}(\lambda_i - \hat{\lambda}_i(\nu))^2}, \tag{4.22}$$

where $\nu = 1, \cdots, 1000$ is the number of the experiment. Initial guesses for λ were selected randomly in the n-cube $[0,1] + \lambda_i$, $i = 1, 2, \cdots, 7$, where λ_i are the nominal values. Figure 4.3 shows histograms of (4.23), (4.22) at the initial step of the algorithm.

To see how well $S_q(t_i, \hat{\lambda})$ approximates $q(t_i, \lambda)$ as a function of t_i the following simple criterion has been used:

$$LS = \sum_{i=1}^{N} (q(t_i, \lambda) - S_q(t_i, \hat{\lambda}))^2. \tag{4.23}$$

Figure 4.4 shows histograms of the distributions of distances between λ and $\hat{\lambda}$ and the least square errors after the application of Nelder-Mead method with Algorithm 2 used to approximate $q(t, \hat{\lambda})$. We observe a pronounced shift of the histograms to the left, where they concentrate around zero. This contrasts sharply with the initial distributions of errors seen in Figure 4.3.

The overall estimation accuracy, including parameters x_0, g_L, I has also improved dramatically. Initially the estimates of $(x_0, \theta) = (x_0, g_L, I)$ varied widely in the intervals $[0.0027, 0.2393]$, $[0.1269, 1.0254]$, $[0.0632, 13.3767]$, respectively. After the optimization, the distributions of

$$d_2(\nu) = \sqrt{\sum_{i=1}^{2} (\theta_i - \hat{\theta}_i(\nu))^2 + (x_0 - \hat{x}_0(\nu))^2} \tag{4.24}$$

shrunk and shifted to the left. The corresponding histograms are shown in Figure 4.5.

Based on the results shown in Figure 4.4 can we conclude that RBF approximation is a viable alternative method to solve linear ODEs.

For the 7-parameter example, we showed that the RBF approximation method coupled with the Nelder–Mead algorithm offers reasonably accurate estimations for parameters. However, for a computationally demanding Morris–Lecar with 7 parameters, we noticed a drop in accuracy; this can be attributed to increased errors of estimation of (4.1). A list of other particular issues when using RBF approximation is summarized below.

1. The set of samples is randomly selected from the training set and the positions of the centers of the basis functions are set according to the samples. The initial selection may affect the outcomes of K-means clustering.

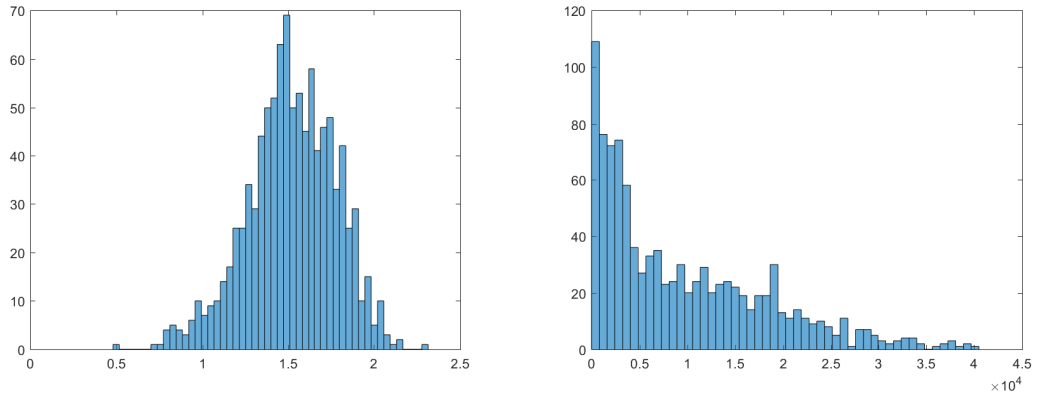


FIGURE 4.3: Histograms of the distributions of $d(\nu)$, $\nu = 1, \cdots, 1000$ (left panel), and least square errors $LS = \sum_{i=1}^{N}(q(t_i, \lambda) - S_q(t_i, \hat{\lambda}))^2$ (right panel) prior to any estimation.

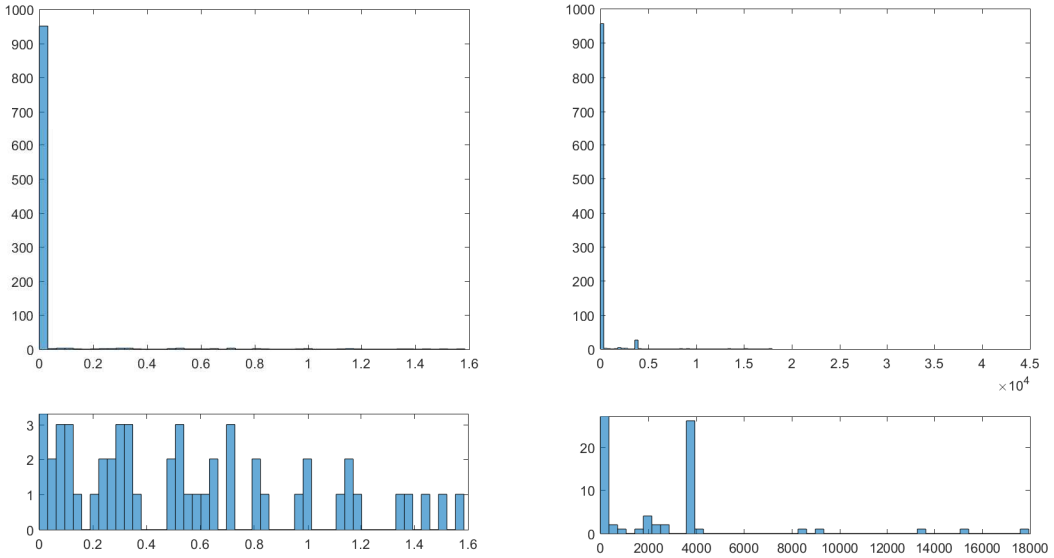


FIGURE 4.4: Histograms of the distributions of $d_1(\nu)$, $\nu = 1, \cdots, 1000$ (left panel) and $LS = \sum_{i=1}^{N}(q(t_i, \lambda) - S_q(t_i, \hat{\lambda}))^2$ (right panel) after optimization. To see finer detail of the tails zoomed-in version of the histograms are shown under the original ones, respectively.

2. The matrix A of the linear system may become ill-conditioned for large samples of parameters.

3. The shape parameter α for Gaussian basis function was determined randomly; it provided acceptable solutions and estimation for parameters. However, from the experiment, it is difficult to say that the solutions are accrate enough representations of the linear part in the Morris–Lecar system; this is illustrated with the "nonoptimal" estimation of the parameters λ in Figure 4.4.

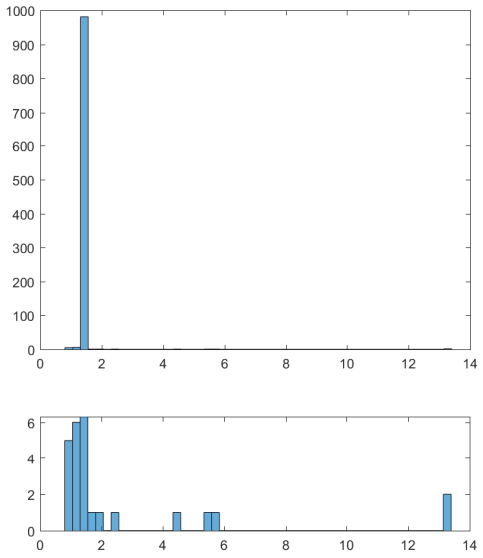


FIGURE 4.5: Histogram of the distribution of the distances $d_2(\nu)$, $\nu = 1, \cdots, 1000$ for the real and the estimated values of parameters g_L, I and the initial point x_0.

These, however, open possibilities for future work and motivates developments of efficient methods for solving linear ODEs. These methods, RBF-based or not, will have major influence on speed, efficiency, accuracy and robustness of the approach overall.

Chapter 5

Conclusion, Discussion and Future Challenges

5.1 Conclusion

In this thesis, instead of finding numerical solutions of IVPs and matching the results to an observed data, we searched for a representation of the problem as in the general form (3.12). In Chapter 3, we presented two methods for computationally efficient and explicit parameter-dependent representation of periodic solutions of systems of nonlinear ODEs. These methods are rooted in the ideas from adaptive observers theory. In this work we focused on two "canonical" observer forms determining the final representations. Different integral representations can be obtained for other observer structures, including e.g. [69], followed by replacement of condition (7) in Assumption 1 with the requirement of uniform persistency of excitation of corresponding terms.

The computational advantage of the method is in a possible parallel implementation of calculations. CPUs and GPUs have significantly different architectures. These different architectures are optimized for different tasks. GPU can handle large amounts of data in many streams simultaneously, and significantly different techniques are required to program GPUs. These different techniques include new programming languages, modifications to existing languages, such as CUDA, and new programming paradigms that are better suited to expressing a computation as a parallel operation to be performed by many stream processors. In addition to offering scalability and making use of parallel computations, the

method offers reduction of dimensionality of the problem due to incorporating linearly parameterized part of the model into internal variables of the proposed representations. These internal variables are uniquely determined by parameters entering the model nonlinearly and are computed as a part of the representation.

The viability of the methods was tested in two examples. The method performed well in the example problems involving parameter estimation for Hodgkin–Huxley and Morris–Lecar equations. We have also shown that the proposed approach greatly benefits from parallel implementation of explicit numerical integration involved. It would be interesting to see if the same approach could be applied to a broader range of systems.

An interesting possibility to further improve computational efficiency of the approach is to invoke RBF approximations in order replace numerical derivation of some auxiliary integrals in the scheme. Viability of these combined approaches in this setting has been demonstrated with a numerical example. We conclude that overall, despite explicit numerical integration of (4.1) could provide better precision than the RBF approximation, the RBF approximation method still delivers feasible accuracy at a reduced computational cost. In this latter case we observed that accuracy of the estimation depends significantly on the choice of initial conditions. The smaller distance from "nominal values" gives a better accuracy. Dealing with this issue is a possible direction of my future study.

Other future research venues include the study of this technique as an effective scheme for the surface representation of large sets of scattered data and parameters. We have not analyzed the theoretical properties of the RBF approximation for the one-dimensional of linear equation of Morris–Lecar problem. Global RBF approximations as the ones used here are competitive for problems in one or two space dimensions, but the computational cost can become prohibitive for higher dimensional problems.

5.2 Discussion and Future Challenges

There are several issues that deserve further study.

- It is important to understand the influence of the scaling parameters we arrange for estimation and how to estimate them for a given data set.

- One should be aware of the possible misspecification of the models. Increasing the model complexity might deteriorate the estimates, due to that the problem may become ill-conditioned.

- An automated procedure for finding initial estimates of parameters and state variables for the procedure would improve applicability of the overall approaches. This, however, is a general issue in many optimization scheme.

Moreover, particular improvements are required about using RBF approximation method for solving linear differential equations. Chapter 4 of this thesis is limited to the Gaussian RBF. This might be extended to other types of RBFs, such as the multiquadric and inverse multiquadric. Another extension to this work is to use radial basis neural networks. Further development can be made on how to choose/find the best shape parameter which could improve the approximation.

The first future study is to apply an interpolation technique based on RBFs proposed and implemented by Usta and Levesley in 2010 [109] for interpolating high-dimensional functions. Their study introduced a new quasi-multilevel sparse interpolation (Q-MuSIK); the algorithm is generally superior to the algorithm of multilevel sparse kernel-based interpolation (MuSIK) in terms of run time in high-dimensional interpolation problems. This method yields a solution directly with no need to solve large algebraic systems.

We will try to solve linear ODEs with Q-MuSIK by considering the unknown parameters ϑ instead of $\boldsymbol{x}$. Even though we introduced two representations of solutions of nonlinear ODEs, we still would need an alternative method like Q-MuSIK method to solve linear ODEs. This method may provide an interpolated data which, then, may improve the estimation of parameters.

Another interesting direction for solving linear ODEs to use what is called "Magic Points". Empirical Magic Point Interpolation method has been developed in [39] in 2017 by Maximilian *et al.* to approximate parametric integrals of the form

$$I(h_\vartheta) = \int_\Omega h_\vartheta(z)dt, \quad \text{for } \vartheta \in \Theta,\ z \in \Omega \tag{5.1}$$

Magic point integration is a quadrature rule for integrating parametric functions, where the interpolation nodes are chosen in a precomputation phase according to a set of integrands $h_\vartheta(z)$ [73]. This set of integrands is associated with Θ as follow:

$$\mathcal{U} = \{h_\vartheta : \Omega \to \mathbb{R} | \vartheta \in \Theta\} \tag{5.2}$$

For $M \in \mathbb{N}$, a mapping I_M defined from $\mathcal{U}$ to a tensor is denoted by $I_M(h_p)(\vartheta, z)$, and the Magic Point Integration with M points is denoted by $\mathcal{I}_M(h_p)(z)$. The purpose of this study is to test the empirical interpolation process in well-documented situations in order to first measure where magic points, z_1^*, z_2^*, $\cdots$, z_M^* $\in \Omega$, stand with respect to some optimal results. In other words, if I_M denotes the Magic Point Integration with M points, the algorithm chooses u_M as the function in the set $\mathcal{U}$ which is worst represented by the approximation with the previously identified $M-1$ magic points:

$$u_M = \arg\max_{u \in \mathcal{U}} \|u - I_M(z)\|_\infty \tag{5.3}$$

and the Mth magic point is defined:

$$z_M^* = \arg\max_{z \in \Omega} \|u_M(z) - I_M(u_M)(z)\|_\infty \tag{5.4}$$

Hence, for $m = 1, 2, \cdots, M$, the integral presented by linear combinations snapshot integrands $h_{p_m^*}$ with coefficients β_1^m, β_2^m, $\cdots$, β_M^m is defined:

$$\mathcal{I}_M(h)(\vartheta) = \sum_{m=1}^{M} h_p(z_m^*) \sum_{j=1}^{M} \beta_j^m \int_\Omega h_{p_m^*}(z) dz \tag{5.5}$$

where $p^* \in \Theta$ identifying u_M in (5.3) are recalled as magic parameters.

This final equation shows that Magic Point Integration is an interpolation method for parametric integrals in the parameter space.

This study shows that under suitable analyticity conditions, the approximation error of Magic Point Integration decays exponentially in M [39], whereas standard integration routines suffer from the curse of dimensionality of the integration domain.

Our next test, then, is to employ this method to integrate linear ODEs over the time interval $[t_0, t_0 + T]$.

Appendix A. Auxiliary Technical Results

Lemma .1. *Consider* $\dot{y} = k(t)y + u(t) + d(t)$, $k, u, d : \mathbb{R}_{\geq t_0} \to \mathbb{R}, u \in \mathcal{C}^1, d \in \mathcal{C}^0$, *and let* $\max\{|u(t)|, |\dot{u(t)}|\} \leq B, |d(t)| \leq \triangle_\xi$, $|k(t)| \leq \rho$. *Finally, let* T, ϵ *be a non-negative real numbers such that* $T > \sqrt{\epsilon}$. *Then*

$$\|y\|_{\infty,[t_0,t_0+T]} \leq \epsilon \Rightarrow \|u\|_{\infty,[t_0,t_0+T)} \leq 2\sqrt{\epsilon}(\ e^{\rho\sqrt{\epsilon}} + B) + \triangle_\xi, \quad \forall\ t \leq t_0 + T.$$

Proof. Let L be an arbitrary element of $[0, T]$. Note that for all $t \geq t_0 + L$ the variable $y(t)$ can be expressed as:

$$y(t) = y(t-L)e^{\int_{t-L}^{t} k(\tau)d\tau} + \int_{t-L}^{t} e^{\int_{\tau}^{t} k(\tau_1)d\tau_1}(u(\tau) + d(\tau))d\tau.$$

According to the mean value theorem there is a $\tau' \in [t-L, t]$:

$$\begin{aligned}
&y(t) - y(t-L)e^{\int_{t-L}^{t} k(\tau)d\tau} = L\ e^{\int_{\tau'}^{t} k(\tau_1)d\tau_1}(u(\tau') + d(\tau')) \\
&\Rightarrow \quad y(t)e^{-\int_{\tau'}^{t} k(\tau_1)d\tau_1} - y(t-L)e^{\int_{t-L}^{\tau'} k(\tau)d\tau} = L\ (u(\tau') + d(\tau')) \\
&\Rightarrow |y(t)e^{-\int_{\tau'}^{t} k(\tau_1)d\tau_1}| + |y(t-L)e^{\int_{t-L}^{\tau'} k(\tau)d\tau}| \geq L\ |u(\tau') + d(\tau')|
\end{aligned}$$

Given that:

$$\begin{aligned}
|e^{\int_{t}^{\tau'} k(\tau_1)d\tau_1}| \quad &\leq |e^{\int_{t}^{\tau'} \rho d\tau_1}| \triangleq e^{\rho(\tau'-t)} \\
\Rightarrow |e^{-\int_{\tau'}^{t} k(\tau_1)d\tau_1}| \quad &\leq e^{\rho(\tau'-t)} \leq e^{\rho L} \\
\Rightarrow |e^{\int_{t-L}^{\tau'} k(\tau_1)d\tau_1}| \quad &\leq |e^{\int_{t-L}^{\tau'} \rho d\tau_1}| \triangleq e^{\rho(\tau'-(t-L))} \\
&\leq e^{\rho(\tau'-(t-L))} \leq e^{\rho L}
\end{aligned}$$

and invoking the mean value theorem we conclude that $\exists\ \tau'' \in [\tau', t]$:

$$\begin{aligned} |u(t)| &= |u(\tau) - u(\tau) + u(t)| \\ &= |u(\tau') + u'(\tau'')(t - \tau') - d(\tau') + d(\tau')| \\ &\leq |u(\tau') + d(\tau')| + BL + \triangle_\xi \\ \Rightarrow |u(\tau') + d(\tau')| &\geq |u(t)| - BL - \triangle_\xi \end{aligned}$$

Hence

$$|u(t)| \leq BL + \triangle_\xi + \frac{2\ \epsilon}{L}\ e^{\rho L}, \quad \forall\ t \leq t_0 + L$$

Given that L can be chosen arbitrary in the interval $[0, T]$, we let $L = \sqrt{\epsilon}$, and thus $|u(t)|\frac{2\ \epsilon}{\sqrt{\epsilon}}\ e^{\rho\sqrt{\epsilon}} + \leq BL + \triangle_\xi$.
Finally, given that $|\dot{u}(t)| \leq B$ for all $t \in [t_0, t_0 + T]$ including in the interval $[t_0, t_0 + \sqrt{\epsilon}]$, we conclude that

$$|u(t)| \leq 2\sqrt{\epsilon}(\ e^{\rho\sqrt{\epsilon}} + B) + \triangle_\xi, \quad \forall\ t \leq t_0 + T$$

□

Corollary .2. *Consider* $\dot{y} = ky + u(t) + d(t)$, $k \in \mathbb{R}$, $u, d : \mathbb{R}_{\geq t_0} \to \mathbb{R}, u \in \mathcal{C}^1, d \in \mathcal{C}^0$, *and let* $\max\{|u(t)|, |\dot{u(t)}|\} \leq B, |d(t)| \leq \triangle_\xi$. *Finally, let* T, ϵ *be a non-negative real numbers such that* $T > \sqrt{\epsilon}$. *Then*

$$|y|_{\infty,[t_0,t_0+T]}\ \leq \epsilon \Rightarrow |u|_{\infty,[t_0,t_0+T)} \leq 2\sqrt{\epsilon}(1 + e^{|k|\sqrt{\epsilon}} + B) + \triangle_\xi, \quad \forall\ t \leq t_0 + T.$$

Proof. Let L be an arbitrary element of $[0, T]$. Noticing that $y(t)$ for $t \geq t_0 + L$, $L > 0$, can be expressed as: $y(t) = y(t - L)e^{kL} + \int_{t-L}^{t} e^{k(t-\tau)}(u(\tau) + d(\tau))d\tau$ and using the mean value theorem we obtain:

$$y(t) - y(t - L)e^{kL} = L\ e^{k(t-\tau')}(u(\tau') + d(\tau')),\ \tau' \in [t - L, t].$$

Hence $\varepsilon(1 + e^{kL}) \geq Le^{k(t-\tau')}(|u| - LB - \triangle_\xi)$, and

$$\triangle_\xi + LB + \frac{\varepsilon(1 + e^{kL})}{L\min\{1, e^{kL}\}} \geq \triangle_\xi + LB + \frac{\varepsilon(1 + e^{kL})}{L\min\{1, e^{k(t-\tau')}\}} \geq |u(t)|.$$

Given that L can be chosen arbitrarily in the interval $[0, T]$ we let $L = \sqrt{\varepsilon}$ and thus

$$|u(t)| \leq \sqrt{\varepsilon}(1 + e^{k\sqrt{\varepsilon}}) \max\{1 + e^{-k\sqrt{\varepsilon}}\} + B\sqrt{\varepsilon} + \triangle_\xi \leq \sqrt{\varepsilon}(1 + e^{|k|\sqrt{\varepsilon}} + B) + \triangle_\xi$$

for all $t \in [t_0 + \sqrt{\varepsilon}, t_0 + T]$. Finally, given that $|\dot{u}(t)| \leq B$ for all $t \in [t_0, t_0 + T]$, including in the interval $t \in [t_0, t_0 + \sqrt{\varepsilon}]$, we include that

$$|u(t)| \leq \sqrt{\varepsilon}(1 + e^{|k|\sqrt{\varepsilon}} + B) + \triangle_\xi$$

for all $t \in [t_0, t_0 + T]$. □

Proof of Lemma 3.1

Let us rewrite the system (3.13) as

$$\begin{aligned} \dot{y} &= C^T \dot{\chi} = \dot{\chi}_1 \\ &= \alpha_1(t)\chi_1 + \beta(t)\tilde{\chi} + u_1(t) + d_1(t) \end{aligned} \tag{6}$$

$$\begin{aligned} \dot{\tilde{\chi}} &= A_0^*(t)\tilde{\chi} + \tilde{\alpha}(t)\chi_1 + b(t)u_1(t) - b(t)u_1(t) + \tilde{u}(t) + \tilde{d}(t) \\ &= A_0^*(t)\tilde{\chi} + \tilde{\alpha}(t)y + b(t)u_1(t) + G(t)u(t) + \tilde{d}(t) \end{aligned} \tag{7}$$

where $G(t) = (-b(t) \quad I_{\ell-1})$, $\tilde{\alpha}(t) = \text{col}(\alpha_2(t), \ldots, \alpha_\ell(t))$, $\beta(t) = (\beta_1(t), \ldots, \beta_\ell(t))$, $\tilde{d}(t) = \text{col}(d_2(t), \ldots, d_\ell(t))$ and $\tilde{\chi} = \text{col}(\chi_2, \ldots, \chi_\ell)$.

Let $\|y(t)\|_{\infty,[t_0,t_0+T]} \leq \epsilon$ and denote $e(t) = \beta(t)\tilde{\chi} + u_1(t)$.

According to Lemma .1, there are $v_1, v_2 \in \mathcal{K}$ such that $\|e(t)\| = \|\beta(t)\tilde{\chi} + u_1(t)\| \leq v_1(\epsilon) + v_2(\triangle_\xi)$ for all $t \in [t_0, t_0 + T]$.

Using the notation above one obtains:

$$\begin{aligned} \dot{\tilde{\chi}} &= (A_0^*(t) - b(t)\beta(t))\tilde{\chi} + \tilde{\alpha}(t)y + G(t)u(t) + b(t)e(t) + \tilde{d}(t) \\ &= \Lambda(t)\tilde{\chi} + \tilde{\alpha}(t)y + G(t)u(t) + b(t)e(t) + \tilde{d}(t). \end{aligned} \tag{8}$$

Therefore,

$$\begin{aligned} \|u_1(t) + h(t)\| &= \|u_1(t) + \beta(t)\tilde{\chi} - \beta(t)\tilde{\chi} + h(t)\| \\ &\leq \|u_1(t) + \beta(t)\tilde{\chi}\| + \|\beta(t)\tilde{\chi} - h(t)\|. \end{aligned}$$

The solutions of (3.15) and (8) are:

$$z(t) = \Phi_\Lambda(t, t_0)z_0 + \int_{t_0}^{t} \Phi_\Lambda(t, \tau)G(\tau)u(\tau)d\tau = \int_{t_0}^{t} \Phi_\Lambda(t, \tau)G(\tau)u(\tau)d\tau,$$

$$\tilde{\chi}(t) = \Phi_\Lambda(t, t_0)\tilde{\chi}_0 + \int_{t_0}^{t} \Phi_\Lambda(t, \tau)(\tilde{\alpha}(\tau)(\tau)y + G(\tau)u(\tau) + b(\tau)e(\tau) + \tilde{d}(\tau))d\tau.$$

Hence

$$\int_{t_0}^{t} \Phi_\Lambda(t, \tau)G(\tau)u(\tau))d\tau = \tilde{\chi}(t) - \Phi_\Lambda(t, t_0)\tilde{\chi}_0 - \int_{t_0}^{t} \Phi_\Lambda(t, \tau)(\tilde{\alpha}(\tau)y + b(\tau)e(\tau) + \tilde{d}(\tau))d\tau$$

Since the system $\dot{z} = \Lambda(t)z$ is uniformly exponentially stable, there are $D, k \in \mathbb{R}_{>0}$ such that $\|\Phi(t, t_0)\| \leq De^{-k(t-t_0)}$.
Therefore,

$$\begin{aligned}\|\beta(t)\tilde{\chi}(t) - \beta(t)z(t)\| &= \|\beta(t)\Phi_\Lambda(t, t_0)\tilde{\chi}_0 - \beta(t)\int_{t_0}^{t} \Phi_\Lambda(t, \tau)(M_1 y + b(\tau)e(\tau) + \tilde{d}(\tau))d\tau\| \\ &\leq M_2 De^{-k(t-t_0)}\|\tilde{\chi}_0\| + \frac{DM_2}{k}\ (1 - e^{-k(t-t_0)})(M_1\epsilon + M_3(v_1(\epsilon) \\ &\quad + v_2(\triangle_\xi)) + \triangle_\xi)\end{aligned}$$

Noticing that $h(t) = \beta(t)\int_{t_0}^{t} \Phi_\Lambda(t, \tau)Gu(\tau)d\tau$, denoting $\kappa(\epsilon) = 2\frac{DM_2}{k}(M_1\epsilon + M_3 v_1(\epsilon)) + v_1(\epsilon)$, $\kappa_2(\triangle_\xi) = 2\frac{DM_2}{k}(\triangle_\xi + M_3 v_2(\triangle_\xi)) + v_2(\triangle_\xi)$, and letting

$$t'(\epsilon, \chi_0) = t_0 + \frac{1}{k}\ln\left(\frac{DM_2\|\chi_0\|}{\epsilon}\right)$$

we conclude that there is a $t'(\epsilon, \chi_0) \geq t_0$ such that

$$\begin{aligned}\|u_1(t) + h(t)\|_{\infty,[t_0,t_0+T]} &= \kappa(\epsilon) + \epsilon + \kappa_2(\triangle_\xi) \\ &\leq \kappa_1(\epsilon) + \kappa_2(\triangle_\xi)\end{aligned}$$

for all $t \in [t'(\epsilon, \chi_0), t_0 + T] : T$ is sufficiently large to satisfy $t_0 + T > t'(\epsilon, \chi_0)$.

Noticing that if $d(t) \equiv 0$, then $y \equiv 0 \Rightarrow \dot{y} \equiv 0$ and we obtain that $\exists$ an initial point $P \in \mathbb{R}$:

$$\begin{aligned} e(t) &= \beta(t)\tilde{\chi} + u_1(t) \\ &= \beta(t)\Phi_\Lambda(t, t_0)P + \beta(t)\int_{t_0}^{t} \Phi_\Lambda(t, \tau)Gu(\tau)d\tau + u_1(t) \\ &= \beta(t)\Phi_\Lambda(t, t_0)P + h(t) + u_1(t) \end{aligned}$$

then from equation (6) we get $e(t) = 0$ which ensures that (3.16) holds.

Proof of Corollary 3.2

Let us rewrite the system (3.18) as

$$\begin{aligned} \dot{y} &= C_1^T \dot{x} = \dot{x}_1 \\ &= \alpha_1(t)y + \tilde{C}\tilde{\chi} + u_1(t) + d_1(t) \\ \dot{\tilde{x}} &= \tilde{A}_0\tilde{x} + \tilde{\alpha}x_1 + bu_1(t) - b\tilde{u}(t) + \tilde{d}(t) \\ &= \tilde{A}_0\tilde{x} + \tilde{\alpha}(t)y + bu_1(t) + Gu(t) + \tilde{d}(t) \end{aligned}$$

where $\tilde{\alpha} = \mathrm{col}(\alpha_2, \ldots, \alpha_n)$, $\tilde{C}_1 = \mathrm{col}(1, 0, \cdots, 0)$, $\tilde{d}(t) = \mathrm{col}(d_2(t), \ldots, d_\ell(t))$, and

$$G(t) = (-b \quad I_{n-1}),\ \tilde{A}_0 = \left(\begin{array}{c|c} 0 & I_{n-2} \\ 0 & 0 \end{array}\right).$$

Let $\|y(t)\|_{\infty,[t_0,t_0+T]} \leq \epsilon$ and denote $e(t) = \tilde{C}_1^T\tilde{x} + u_1(t)$.
According to Corollary .2, there are $v_1, v_2 \in \mathcal{K}$ such that $\|e(t)\| = \|\tilde{C}_1^T\tilde{x} + u_1(t)\| \leq v_1(\epsilon) + v_2(\triangle_\xi)$ for all $t \in [t_0, t_0 + T]$.
Using the notation above we obtain:

$$\dot{\tilde{x}} = (\tilde{A}_0 - b\tilde{C}_1^T)\tilde{x} + \tilde{\alpha}(t)y(t) + Gu(t) + be(t) + \tilde{d}(t) \tag{9}$$

Matrix $\tilde{A}_0 - b\tilde{C}_1^T = \Lambda$ is Hurwitz, and hence there are $D, k \in \mathbb{R}_{>0}$ such that $\|e^{\Lambda(t-t_0)}\| \leq De^{-k(t-t_0)}$.

Therefore

$$\|\tilde{C}_1^T\tilde{x}(t) - \tilde{C}_1^T\int_{t_0}^{t} e^{\Lambda(t-\tau)}Gu(\tau)d\tau\| \leq De^{-k(t-t_0)}\|\tilde{x}(t_0)\| + \frac{D}{k}(\|a\|\varepsilon + \|b\|(v_1(\varepsilon) + v_2(\triangle_\xi)) + \triangle_\xi).$$

Noticing that $z_1 = \tilde{C}_1^T\int_{t_0}^{t} e^{\Lambda(t-\tau)}Gu(\tau)d\tau$, denoting $\kappa(\varepsilon) = 2\frac{D}{k}(\|a\|\varepsilon + \|b\|v_1(\varepsilon) + v_1(\varepsilon)$, $\kappa_2(\triangle_\xi) = 2\frac{D}{k}(\triangle_\xi + \|b\|v_2(\triangle_\xi)) + v_2(\triangle_\xi)$, and

$$t'(\epsilon, x_0) = t_0 + \frac{1}{k}\ln\left(\frac{D\|x_0\|}{\epsilon}\right)$$

we can conclude that there is a $t'(\epsilon, x_0) \geq t_0$ such that

$$\|z_1(\tau) + u_1(\tau)\|_{\infty,[t_0,t_0+T]} \leq \kappa(\varepsilon) + \varepsilon + \kappa_2(\triangle_\xi) = \kappa_1(\varepsilon\kappa_2(\triangle_\xi)).$$

for all $t \in [t'(\epsilon, x_0), t_0 + T]$, providing that T is sufficiently large to satisfy $t_0 + T > t'(\epsilon, x_0)$. Noticing that $y(t) \equiv 0 \Rightarrow e(t) \equiv 0$ ensures that (3.20) holds too.

Appendix B. Parallel Prefix Sum Algorithm for Integral Computation

Consider the finite integral defined over the interval $[t_0, t_0 + T]$:

$$I = \int_{t_0}^{t_0+T} u(\tau, \lambda, y) d\tau, \tag{10}$$

and included in the explicit solution form (3.12).

In many practical situations, we do not have a formula for the integrand, and in fact the value of u may only be known at a finite set of N data points, *i.e*

$$(t_0, u_0), (t_1, u_1), \cdots, (t_{N-1}, u_{N-1})$$

Matlab has commands for automatically computing "*cumulative integrals*". These are cumtrapz and trapz which take the same kind of input arguments, i.e. arrays of t_i and $u(t_i)$ values, and outputs a sequence of values approximating the cumulative integral for u from t_0 up to t_0, t_1, $\cdots$, t_{N-1}, for example, cumsum results in

$$[u_0, (u_0 + u_1), \cdots, u_0 + u_1 + \cdots + u_{N-1}] \tag{11}$$

Prefix sum scan offers possibilities to invoke parallel processes to speed up the calculations. One way to implement (11) is to add term sequentially. An alternative is the so called parallel scan algorithm which which we describe in detail below.

The sequential prefix sum of a sequence of N values is a new sequence of N values where the value at position i is the sum of all the values in the input sequence up to position i. If the sum includes the input value at position i, it is an inclusive prefix sum. Otherwise, it is an exclusive prefix sum. Algorithm 3

shows sequential implementation computes the inclusive prefix sum of the values in array u and stores the result back into the output array *sum*:

Algorithm 3 Sequential Implementation of prefix sum of order $O(N)$.

sum[0] = u[0]
for i from 0 to $N1$ do
sum[i] = sum[i-1] + u[i]

This algorithm performs $O(N)$ computations on N data. Due to the loop-carried dependency, each iteration can only be executed after the previous iteration has finished, making the code inherently sequential. However, several approaches for computing prefix sums in parallel are known [13]. The order of their computational complexity is significantly less than $O(N)$. Algorithm 4 and Figure 1 show prefix sum computations of order $\log_2(N)$; all the commutations are executed in parallel.

Algorithm 4 Parallel prefix computation of order $\log_2(N)$.

for j from 0 to $(\log_2(N) - 1)$
in parallel for $k = 0$ to $N - 1$ by 2^j
$u[j + 2^{k+1} - 1] = u[j + 2^k - 1] + u[j + 2^{k+1} - 1]$

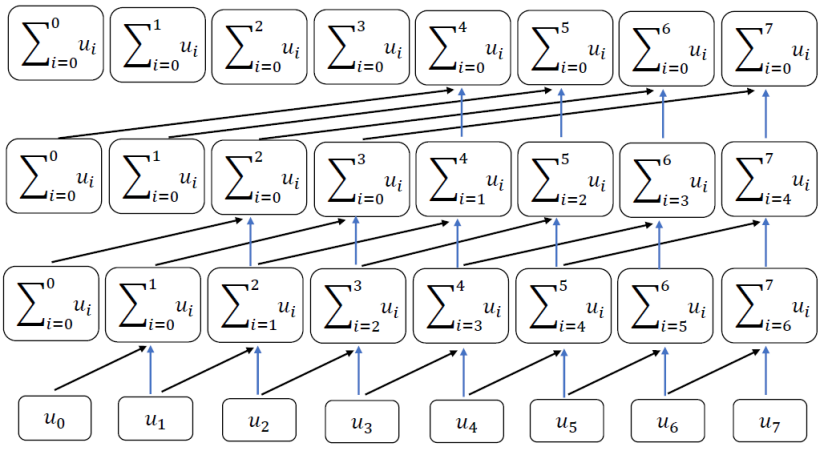


FIGURE 1: Prefix sum on array of eight elements.

Bibliography

[1] Abarbanel, H. D., Brown, R., Sidorowich, J. J., and Tsimring, L. S. (1993). The analysis of observed chaotic data in physical systems. *Reviews of modern physics*, 65(4):1331.

[2] Anderson, E. J. and Ferris, M. C. (2001). A direct search algorithm for optimization with noisy function evaluations. *SIAM Journal on optimization*, 11(3):837–857.

[3] Ascher, U. M., Mattheij, R. M., and Russell, R. D. (1994). *Numerical solution of boundary value problems for ordinary differential equations*, volume 13. Siam.

[4] Astolfi, A., Karagiannis, D., and Ortega, R. (2007). *Nonlinear and adaptive control with applications*. Springer Science & Business Media.

[5] Astolfi, A. and Ortega, R. (2003). Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems. *IEEE Transactions on Automatic control*, 48(4):590–606.

[6] Aström, K. J. and Murray, R. M. (2010). *Feedback systems: an introduction for scientists and engineers*. Princeton university press.

[7] Banks, H., Dediu, S., and Ernstberger, S. L. (2007). Sensitivity functions and their uses in inverse problems. *Journal of Inverse and Ill-posed Problems jiip*, 15(7):683–708.

[8] Banks, H. T., Robbins, D., and Sutton, K. (2012). Theoretical foundations for traditional and generalized sensitivity functions for nonlinear delay differential equations. *Math. Biosci. Eng. CRSC-TR12-14*.

[9] Bastin, G. and Dochain, D. (1990). *On-line Estimation and Adaptive Control of Bioreactors*. Elsevier.

[10] Bastin, G. and Gevers, M. (1988). Stable adaptive observers for nonlinear time-varying systems. *IEEE Trans. on Automatic Control*, 33(7):650–658.

[11] Baudin, M. (2009). Nelder mead user's manual.

[12] Besançon, G. (2000). Remarks on nonlinear adaptive observer design. *Systems & control letters*, 41(4):271–280.

[13] Bilgic, B., Horn, B. K., and Masaki, I. (2010). Efficient integral image computation on the gpu. In *Intelligent vehicles symposium (IV), 2010 IEEE*, pages 528–533. IEEE.

[14] Bock, H. G., Kostina, E., and Schlöder, J. P. (2007). Numerical methods for parameter estimation in nonlinear differential algebraic equations. *GAMM-Mitteilungen*, 30(2):376–408.

[15] Bozzini, M., Lenarduzzi, L., Rossini, M., and Schaback, R. (2014). Interpolation with variably scaled kernels. *IMA Journal of Numerical Analysis*, 35(1):199–219.

[16] Bryan, K. and Shibberu, Y. (2005). Penalty functions and constrained optimization. *Dept. of Mathematics, Rose-Hulman Institute of Technology. http:// www. rosehulman. edu/˜ bryan/lottamath/penalty. pdf*.

[17] Buhry, L., Saighi, S., Giremus, A., Grivel, E., and Renaud, S. (2008). Parameter estimation of the hodgkin-huxley model using metaheuristics: application to neuromimetic analog integrated circuits. In *Biomedical Circuits and Systems Conference, 2008. BioCAS 2008. IEEE*, pages 173–176. IEEE.

[18] Butcher, J. C. and Wanner, G. (1996). Runge-kutta methods: some historical notes. *Applied Numerical Mathematics*, 22(1-3):113–151.

[19] Capoyleas, V., Rote, G., and Woeginger, G. (1991). Geometric clusterings. *Journal of Algorithms*, 12(2):341–356.

[20] Carraro, T. (2005). *Parameter estimation and optimal experimental design in flow reactors*. PhD thesis.

[21] Cole, D. R. F. (2014). *Asymptotic state and parameter observation for dynamical systems with nonlinear parameterisation*. PhD thesis, Department of Mathematics.

[22] Collado, J., Lozano, R., and Johansson, R. (2001). On kalman-yakubovich-popov lemma for stabilizable systems. *IEEE Transactions on Automatic Control*, 46(7):1089–1093.

[23] De Marchi, S. (2018). Lectures on radial basis functions.

[24] De Terán, F. and Dopico, F. M. (2011). Consistency and efficient solution of the sylvester equation forâŃĘ-congruence. *Electron. J. Linear Algebra*, 22:849–863.

[25] Dennis, J. and Woods, D. J. (1987). Optimization on microcomputers: The nelder-mead simplex algorithm. *New computing environments: microcomputers in large-scale computing*, 11:6–122.

[26] Distefano, J. and Cobelli, C. (1980a). On parameter and structural identifiability: Nonunique observability/reconstructibility for identifiable systems, other ambiguities, and new definitions. *IEEE Transactions on Automatic Control*, 25(4):830–833.

[27] Distefano, J. and Cobelli, C. (1980b). On parameter and structural identifiability: Nonunique observability/reconstructibility for identifiable systems, other ambiguities, and new definitions. *IEEE Transactions on Automatic Control*, 25(4):830–833.

[28] Driscoll, T. A. and Fornberg, B. (2002). Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3):413–422.

[29] Eigen, M. (1967). Immeasurably fast reactions. *Nobel Lecture*, 11:1963–1979.

[30] Einstein, A. (1905). On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat. *Annalen der physik*, 17:549–560.

[31] Farza, M., M'Saad, M., Maatoug, T., and Kamoun, M. (2009). Adaptive observers for nonlinearly parameterized class of nonlinear systems. *Automatica*, 45(10):2292–2299.

[32] Fasshauer, G. E. and McCourt, M. J. (2012). Stable evaluation of gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing*, 34(2):A737–A762.

[33] Fornberg, B., Larsson, E., and Flyer, N. (2011a). Stable computations with gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2):869–892.

[34] Fornberg, B., Larsson, E., and Flyer, N. (2011b). Stable computations with gaussian radial basis functions. *SIAM Journal on Scientific Computing*, 33(2):869–892.

[35] Fornberg, B. and Piret, C. (2007). A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80.

[36] Fornberg, B. and Wright, G. (2004). Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers & Mathematics with Applications*, 48(5):853–867.

[37] Gábor, A. and Banga, J. R. (2015). Robust and efficient parameter estimation in dynamic models of biological systems. *BMC systems biology*, 9(1):74.

[38] Gao, F. and Han, L. (2012). Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277.

[39] Gaß, M., Glau, K., and Mair, M. (2017). Magic points in finance: Empirical integration for parametric option pricing. *SIAM Journal on Financial Mathematics*, 8(1):766–803.

[40] Goldstein, A. A. (1965). On steepest descent. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 3(1):147–151.

[41] Gorban, A. N., Mirkes, E. M., and Zinovyev, A. (2016). Piece-wise quadratic approximations of arbitrary error functions for fast and robust machine learning. *Neural Networks*, 84:28–38.

[42] Gupta, N. and Mehra, R. (1974). Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *IEEE Transactions on Automatic Control*, 19(6):774–783.

[43] Hamilton, F. (2011). Parameter estimation in differential equations: A numerical study of shooting methods.

[44] Hammouri, H. and de Morales, L. (1990). Observer synthesis for state-affine systems. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 784–785. IEEE.

[45] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[46] Han, L. and Neumann, M. (2006). Effect of dimensionality on the nelder–mead simplex method. *Optimization Methods and Software*, 21(1):1–16.

[47] Hardy, G. H. (1916). *The integration of functions of a single variable*. Number 2. University Press.

[48] Hassard, B. (1978). Bifurcation of periodic solutions of the hodgkin-huxley model for the squid giant axon. *Journal of Theoretical Biology*, 71(3):401–420.

[49] Hon, Y.-C. and Mao, X.-Z. (1999). A radial basis function method for solving options pricing models. *Journal of Financial Engineering*, 8:31–50.

[50] Hou, M. and Muller, P. (1992). Design of observers for linear systems with unknown inputs. *IEEE Transactions on automatic control*, 37(6):871–875.

[51] Immler, F. and Hölzl, J. (2013). Numerical analysis of ordinary differential equations.

[52] Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37.

[53] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

[54] Jing, Z. and Chen, L. (1984). The existence and uniqueness of limit cycles in general predator-prey differential equations. *Chineese Sci. Bull.*, 9:521–523.

[55] Karabutov, N. and Karabutov, P. (2009). Adaptive observers for linear dynamic systems. *Measurement Techniques*, 52(8):813–820.

[56] Kasper, T. (1980). Integration in finite terms: the liouville theory. *ACM SIGSAM Bulletin*, 14(4):2–8.

[57] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

[58] Kojić, A. and Annaswamy, A. M. (2002). Adaptive control of nonlinearly parameterized systems with a triangular structure. *Automatica*, 38(1):115–123.

[59] Kojić, A., Annaswamy, A. M., Loh, A.-P., and Lozano, R. (1999). Adaptive control of a class of nonlinear systems with convex/concave parameterization. *Systems & control letters*, 37(5):267–274.

[60] Kreisselmeier, G. (1977). Adaptive observers with exponential rate of convergence. *IEEE transactions on automatic control*, 22(1):2–8.

[61] Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147.

[62] Liang, H., Miao, H., and Wu, H. (2010). Estimation of constant and time-varying dynamic parameters of hiv infection in a nonlinear differential equation model. *The annals of applied statistics*, 4(1):460.

[63] Liu, X., Ortega, R., Su, H., and Chu, J. (2009). Adaptive control of nonlinearly parameterized nonlinear systems. In *American Control Conference*, pages 11–13. July, San Luis, Mo., USA.

[64] Ljung, L. 1999, system identification: Theory for the user.

[65] Ljung, L. (1987). *System Identification – Theory for the User*. Prentice Hall, Englewood Cliffs, New Jersy, USA.

[66] Loh, A.-P., Annaswamy, A. M., and Skantze, F. P. (1999). Adaptation in the presence of a general nonlinear parameterization: An error model approach. *IEEE Transactions on Automatic Control*, 44(9):1634–1652.

[67] Lorıa, A. (2004). Explicit convergence rates for mrac-type systems. *Automatica*, 40(8):1465–1468.

[68] Lorıa, A. and Panteley, E. (2003). Uniform exponential stability of linear time-varying systems: revisited. *Systems & Control Letters*, 47(1):13–24.

[69] Loria, A., Panteley, E., and Zavala, A. (2009). Adaptive observers for robust synchronization of chaotic systems. *IEEE Trans. on Circ. Syst. I: Regular Papers*, 56(12):2703–2716.

[70] Luders, G. and Narendra, K. (1973). An adaptive observer and identifier for a linear system. *IEEE Transactions on Automatic Control*, 18(5):496–499.

[71] Luenberger, D. (1966). Observers for multivariable systems. *IEEE Transactions on Automatic Control*, 11(2):190–197.

[72] Luenberger, D. (1971). An introduction to observers. *IEEE Transactions on automatic control*, 16(6):596–602.

[73] Maday, Y., Nguyen, N. C., Patera, A. T., and Pau, G. S. (2007). A general, multipurpose interpolation procedure: the magic points.

[74] Marine, R., Santosuosso, G. L., and Tomei, P. (2001). Robust adaptive observers for nonlinear systems with bounded disturbances. *IEEE Transactions on automatic control*, 46(6):967–972.

[75] Marino, R. (1990). Adaptive observers for single output nonlinear systems. *IEEE Transactions on Automatic Control*, 35(9):1054–1058.

[76] Marino, R. and Tomei, P. (1992). Global adaptive observers for nonlinear systems via filtered transformations. *IEEE Transactions on Automatic Control*, 37(8):1239–1245.

[77] Marino, R. and Tomei, P. (1995). Adaptive observers with arbitrary exponential rate of convergence for nonlinear systems. *IEEE Transactions on Automatic Control*, 40(7):1300–1304.

[78] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441.

[79] McSharry, P. E., Clifford, G. D., Tarassenko, L., and Smith, L. A. (2003). A dynamical model for generating synthetic electrocardiogram signals. *IEEE transactions on biomedical engineering*, 50(3):289–294.

[80] Miao, H., Xia, X., Perelson, A. S., and Wu, H. (2011). On identifiability of nonlinear ode models and applications in viral dynamics. *SIAM review*, 53(1):3–39.

[81] Micchelli, C. A. (1984). Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, pages 143–145. Springer.

[82] Mohammed, J. A.-A. and Tyukin, I. (2017). Explicit parameter-dependent representations of periodic solutions for a class of nonlinear systems. *IFAC-PapersOnLine*, 50(1):4001–4007.

[83] Moles, C. G., Mendes, P., and Banga, J. R. (2003). Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research*, 13(11):2467–2474.

[84] Morris, C. and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophys. J.*, 35:193–213.

[85] Munnoli, S. P. and Bapat, A. (2013). Clustering algorithms for radial basis function neural network. *Journal of Transactions on Electrical and Electronics Engineering*, 1(1):113–116.

[86] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

[87] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257.

[88] Perabò, S. and Zhang, Q. (2009). Adaptive observers for linear stochastic time-variant systems with disturbances. *International Journal of Adaptive Control and Signal Processing*, 23(6):547–566.

[89] Pettersson, U., Larsson, E., Marcusson, G., and Persson, J. (2008). Improved radial basis function methods for multi-dimensional option pricing. *Journal of Computational and Applied Mathematics*, 222(1):82–93.

[90] Poyton, A., Varziri, M. S., McAuley, K. B., McLellan, P., and Ramsay, J. O. (2006). Parameter estimation in continuous-time dynamic models using principal differential analysis. *Computers & chemical engineering*, 30(4):698–708.

[91] Price, C. J., Coope, I. D., and Byatt, D. (2002). A convergent variant of the nelder–mead algorithm. *Journal of optimization theory and applications*, 113(1):5–19.

[92] Risch, R. H. (1969). The problem of integration in finite terms. *Transactions of the American Mathematical Society*, 139:167–189.

[93] Risch, R. H. (1970). The solution of the problem of integration in finite terms. *Bulletin of the American Mathematical Society*, 76(3):605–608.

[94] Risch, R. H. (1979). Algebraic properties of the elementary functions of analysis. *American Journal of Mathematics*, pages 743–759.

[95] Rothstein, M. (1977). A new algorithm for the integration of exponential and logarithmic functions. In *Proceedings of the 1977 MACSYMA users conference*, pages 263–274.

[96] Schaback, R. (1995). Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264.

[97] Schwaab, M., Biscaia Jr, E. C., Monteiro, J. L., and Pinto, J. C. (2008). Nonlinear parameter estimation through particle swarm optimization. *Chemical Engineering Science*, 63(6):1542–1552.

[98] Siciliano, R. (2012). The hodgkin-huxley model.

[99] Skantze, F. P., Kojić, A., Loh, A.-P., and Annaswamy, A. M. (2000). Adaptive estimation of discrete-time systems with nonlinear parameterization. *Automatica*, 36(12):1879–1887.

[100] Soderstrom, T. and Stoica, P. (1988). *System Identification*. Prentice Hall, Englewood Cliffs, New Jersy, USA.

[101] Stanway, R., Sproston, J., and Stevens, N. (1987). Non-linear modelling of an electro-rheological vibration damper. *Journal of Electrostatics*, 20(2):167–184.

[102] Sugimoto, M., Ohmori, H., and Sano, A. (2000). Continuous-time adaptive observer for linear system with unknown time delay. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 2, pages 1104–1109. IEEE.

[103] Terelius, B. (2009). *Symbolic Intergration*. Skolan för datavetenskap och kommunikation, Kungliga Tekniska högskolan.

[104] Tomick, J. J. (1995). On convergence of the nelder-mead simplex algorithm for unconstrained stochastic optimization. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH.

[105] Torres, L., Besançon, G., Georges, D., and Verde, C. (2012). Exponential nonlinear observer for parametric identification and synchronization of chaotic systems. *Mathematics and Computers in Simulations*, 82:836–846.

[106] Tyukin, I. (2011). *Adaptation in dynamical systems*. Cambridge University Press.

[107] Tyukin, I. Y., Gorban, A., Tyukina, T., Al-Ameri, J., and Korablev, Y. A. (2016). Fast sampling of evolving systems with periodic trajectories. *Mathematical Modelling of Natural Phenomena*, 11(4):73–88.

[108] Tyukin, I. Y., Steur, E., Nijmeijer, H., and Van Leeuwen, C. (2013). Adaptive observers and parameter estimation for a class of systems nonlinear in the parameters. *Automatica*, 49(8):2409–2423.

[109] Usta, F. and Levesley, J. (2018). Multilevel quasi-interpolation on a sparse grid with the gaussian. *Numerical Algorithms*, 77(3):793–808.

[110] Wallace, B. (2004). Constrained optimization: Kuhn-tucker conditions. *Royal Holloway, Egham*.

[111] Wright, S. and Nocedal, J. (1999). Numerical optimization. *Springer Science*, 35(67-68):7.

[112] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

[113] Yan, X.-G., Spurgeon, S. K., and Edwards, C. (2013). State and parameter estimation for nonlinear delay systems using sliding mode techniques. *IEEE Transactions on Automatic Control*, 58(4):1023–1029.

[114] Yao, L. and Sethares, W. A. (1994). Nonlinear parameter estimation via the genetic algorithm. *IEEE Transactions on signal processing*, 42(4):927–935.

[115] Zeitz, M. (1987). The extended luenberger observer for nonlinear systems. *Systems & Control Letters*, 9(2):149–156.

[116] Zhan, C. and Yeung, L. F. (2011). Parameter estimation in systems biology models using spline approximation. *BMC systems biology*, 5(1):1.

[117] Zhang, Q. (2016). *Multilevel Adaptive Radial Basis Function Approximation using Error Indicators*. PhD thesis, Department of Mathematics.

[118] Zhang, Q. and Clavel, A. (2001). Adaptive observer with exponential forgetting factor for linear time varying systems. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 4, pages 3886–3891. IEEE.

[119] Zhang, W. (2012). Improved implementation of multiple shooting for bvps. *Computer Science Department, University of Toronto*.