UNIVERSITY OF SUSSEX

DOCTORAL THESIS

A Bayesian framework for inverse problems for quantitative biology

Author:
Eduard CAMPILLO-FUNOLLET

Supervisor:
Prof. Anotida MADZVAMUSE

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Department of Mathematics

December 2018

Declaration of Authorship

I, Eduard CAMPILLO-FUNOLLET, declare that this thesis titled, "A Bayesian framework for inverse problems for quantitative biology" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

"Skill to do comes of doing."

Ralph Waldo Emerson

UNIVERSITY OF SUSSEX

Abstract

School of Mathematical and Physical Sciences

Department of Mathematics

Doctor of Philosophy

A Bayesian framework for inverse problems for quantitative biology

by Eduard CAMPILLO-FUNOLLET

In this thesis, we present a Bayesian framework to solve inverse problems in the context of quantitative biology. We present a novel combination of the Bayesian approach to inverse problems, suitable for infinite-dimensional problems, with a parallel, scalable Markov Chain Monte Carlo algorithm to approximate the posterior distribution. Both the Bayesian framework and the parallelised MCMC were already known but they were not used in this context in the past. Our approach puts together existing results in order to provide a tool to easily solve inverse problems. We focus on models given by partial differential equations. Our methodology differs from previous results in its approach: it aims to be as transparent and independent of the model as possible, in order to make it flexible and applicable to a wide range of problems emerging from experimental and physical sciences. We illustrate our methodology with three of such applications in the areas of theoretical biology and cell biology.

The first application deals with parameter and function identification within a Turing pattern formation model. To the best of our knowledge, our results are the first attempt to use Bayesian techniques to study the inverse problem for Turing patterns. In this example, we show how our implementation can deal with both finite- and infinite-dimensional parameters in the context of inverse problems for partial differential equations.

The second example studies the spatio-temporal dynamics in cell biology. The study provides an example that seeks to best-fit a mathematical model to experimental data finding in the process optimal parameters and credible regimes and regions. We present a new derivation of the model, that corrects the short-comings of previous approaches. We provide all the details from techniques for data acquisition to the parameter identification, and we show in particular how the mathematical model can be used as a proxy to estimate parameters that are difficult to measure in the experiments, providing an novel alternative to more indirect estimates that also require more complex experiments.

Finally, our third example illustrates the flexibility of our implementation of the methodology by using it to study traction force microscopy (TFM) data with a solver implemented independent of the Bayesian approach for parameter identification. We limit ourselves to the classical TFM setting, that we model as a two-dimensional linear elasticity problem. The results and methods generalise to more complex settings where quantitative modelling driven by biological observations is a requirement.

Acknowledgements

Firstly, I would like to acknowledge my main advisor Prof. Anotida Madzvamuse, for his continuous support and patience. This thesis would not have been possible without his motivation and knowledge. I could have not imagined a better advisor and mentor for my PhD.

I would like to thank Chandrasekhar Venkataraman, who was my co-supervisor during the first two years of my PhD, for his insightful comments and stimulating discussions.

I send my gratitude to my office mates for most of my PhD, Laura Murphy, Benard Kipchumba, Muflih Alhamzi and Bootan Rahman, and to past and present members of the Madzvamuse research group: Davide Cusseddu, Victor Juma and Wakil Sarfaraz.

I crossed paths with many people that in one way or another influenced and inspired me. A special mention to Kathryn Burrows for the *science socials* and great discussions about mathematics and physics, and to Rosanna Barnard, for all the hours shared as ATs.

My life as a PhD student has been much easier thanks to the great job of the technical and administration staff at the School of Mathematical and Physical Sciences and at the University of Sussex. My most sincere gratitude to all of them.

Last but not least, I would like to thank my wife, my brother and my parents, for supporting me despite all my failures. I could not have completed this thesis without their support and understanding.

My PhD was funded by the Leverhulme Trust (RPG-2014-149). I also received support for travel from the University of Sussex. I received support for participating in conferences from the EPSRC, EU Horizon 2020, IMA and NSF.

My most sincere gratitude to my examiners, for their comments and criticisms that helped improved this thesis a lot. They made the last part of this PhD voyage most enjoyable.

Contents

List of Figures

List of Tables

List of Abbreviations

ABC	**A**pproximate **B**ayesian **C**omputation
MCMC	**M**arkov **C**hain **M**onte **C**arlo
ODE	**O**rdinary **D**ifferential **E**equation
PDE	**P**artial **D**ifferential **E**equation
PDF	**P**robability **D**ensity **F**unction
TFM	**T**raction **F**orce **M**icroscopy

To Anna

Chapter 1

Introduction

Quantitative biology is evolving quickly due to both the advances in mathematical techniques and the modern experimental methods, that provide large amounts of high quality data. Therefore, there is a need for robust and efficient parameter identification techniques that can deal with the experimental data as well as with the complexity of the models.

Models of Partial differential equations (PDEs) are a good example of this situation. With experimental advances providing data in space and time, the mathematical models moved naturally to PDEs in order to incorporate all the available information. At the same time, the parameterisation of the model may also incorporate space or time dependent parameters that can reproduce accurately the biological system features.

It is clear that we require both theoretical and computational advances to tackle the inverse problems in the context of quantitative biology. The computational power available today makes it feasible to solve the mathematical model many times, for different parameter values, in order to extract as much information as possible from the data. Furthermore, a sound approach allows a feedback cycle with the experimentalists, where they use the model results, together with the new information about the parameters, to design new experiments and validate or improve the mathematical model.

This thesis aims to present a novel mathematical and computational approach for quantitative biology using a full Bayesian method. Within this approach, we present a methodology for parameter identification problems, that exploits the available computational power for parallelisation, and at the same time is flexible in the use of different mathematical models and implementations. Furthermore, our approach is robust in its interpretation, thus allowing a sound interpretation of the results across a wide range of disciplines and applications.

1.1 Quantitative biology

Despite the long history of applications of mathematics to biology, recent advances in experimental acquisition of data in multi-dimensions necessitate new quantitative methods to mirror such advances that have the capacity to deal with large and complex spatiotemporal series of data. In this thesis, we are interested in quantitative modelling, and how the models help the biologists in the study of biological systems. There are many different modelling approaches, depending among other things on the questions of interest and the available data, but we are interested in models defined by differential equations.

In many cases, a popular approach is to use ordinary differential equations (ODEs). Although in some cases this imply a simplification in terms of the spatial features of the physical system, the well-established analytical tools to study ODEs provide many insights on the characteristics of the system and its solutions.

But with the advance of experimental techniques, data containing spatial information became available, and therefore partial differential equations became more suitable. Furthermore, the advances in the theory of PDEs, both in analysis and numerics, contributed to the popularisation of these models.

Therefore, there is need to study the parameter identification problem for PDEs, in order to exploit at the same time the features of the model and the experimental data in a novel quantitative biology approach.

1.2 Inverse problems

Parameter identification is the problem of extracting information about the parameters of a model from the actual result of some measurement (Tarantola (2005)). This problem is known by many different names, depending on the context. In mathematics, specially when discussing the theoretical aspects, the problem is referred to as an inverse problem, thus considering as a direct problem solving the mathematical model when the value of the parameters is available. In other contexts, the same problem is known as model fitting or parameter fitting.

Let us introduce some basic notation, for the purposes of this introduction. We will provide more details in Chapter 2. Assume we have a mathematical model. In the context of this thesis, our model is always a system of partial differential equations. The model depends on certain parameters, e.g. diffusion rates or reaction rates. Let u be the parameter of the

model; in general, u is multidimensional. Let G be the solution operator for our model: given a parameter u, $G(u)$ is the solution to our system of PDEs. In general, in the actual experiment we observe some projection of G onto a finite dimensional space. Let us denote this projection by $\mathcal{G}(u)$.

Let y be the measurement from the experiment. If the mathematical model was a perfect match with the physical system, we would observe $\mathcal{G}$, that is

$$y = \mathcal{G}(u). \tag{1.1}$$

In reality, several errors affect the measurement. The simplest situation is when we assume that our model is a perfect match with the physical system, but we consider some measurement error in the form of additive noise. We then have

$$y = \mathcal{G}(u) + \eta, \tag{1.2}$$

where η represents the noise.

Using this notation, the parameter identification problem can be stated as follow: given a measurement y, find the parameter u that corresponds to such measurement.

There are two main approaches to this problem. On one hand, we have two quantities, the measurement y and the model solution $\mathcal{G}(u)$ that we want to match. The presence of noise implies that in general we will not have an exact solution—i.e. $\bar{u}$ such that (1.1) is satisfied. Therefore, we can try to minimise the distance between the data y and the solution $\mathcal{G}(u)$. We will refer to this approach as the optimisation or optimal control approach.

A second approach is the so called Bayesian approach. Here, the idea is to understand all the involved quantities as probability distributions. Therefore, given a parameter u, (1.2) characterises the conditional probability of observing y given u. By means of the Bayes' theorem, we can compute the reverse condition: the probability of the parameter u given the observation y. This is exactly our object of interest and the cornerstone of this thesis.

1.2.1 Optimisation approach

It is natural to tackle a parameter identification problem as an optimisation problem, by trying to minimise the distance between the data and the solution to the mathematical model. In this context, it is often necessary to include a regularisation term in order to address the difficulties

presented by the inverse problem, in particular to ensure that the distance between data and model solution is well-defined, and that the inverse problem is well-posed. A typical example of ill-posedness of the inverse problem is when we have two local minima. If the value of the local minima is close, a small change in the parameters, even if it only produces a small change in the value of the local minima, may change the location of the global minimum drastically. See Figure 1.1 for an illustration of this phenomenon.

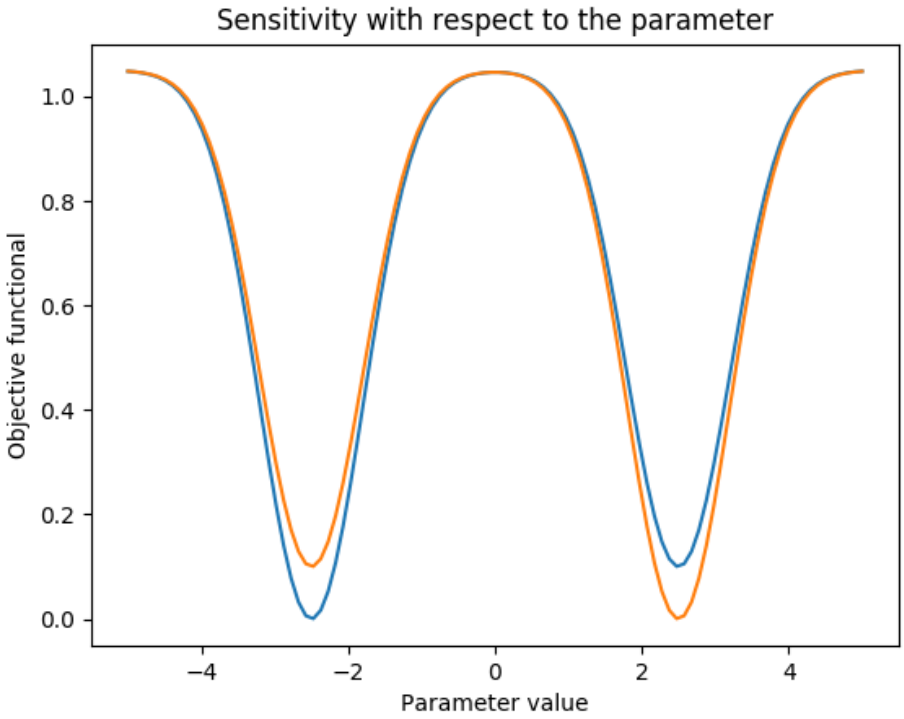


FIGURE 1.1: An illustration of how even for a well-posed forward problem, small changes in the parameter can lead to significant changes in the location of the global minimum. The two curves, representing for instance the distance from the model solution to the experimental data for each value of the parameter, are close to each other globally, yet the location of the minimum is drastically different.

The result of the optimisation problem is one value for the parameters, that we interpret as the best possible value, i.e. the parameter that best reproduces the data. Using the same example, two local minima with similar values in different locations imply that there are two different parameter values that reproduce the data with similar accuracy. Another quantity of interest when solving a parameter identification problem is the uncertainty of the result, in other words, with how much accuracy are we estimating the parameter, given the available data. From the optimisation point of view, the uncertainty can be estimated by studying the

Hessian of the distance as the parameter varies, but the technical choices—for instance, the distance or any regularisation included to ensure well-posedness— impact its value. Therefore, it is difficult to give a robust interpretation to this uncertainty.

1.2.2 Bayesian approach

The Bayesian approach to parameter identification starts with a change of paradigm: we do not interpret the parameter as a single value or function, but instead we want to model our *knowledge* about the parameter. The main premise of the Bayesian approach is to assume that a probability distribution models our knowledge about a physical quantity.

This is a very general statement, and leads to different approaches to solve the problem. It is important to notice that nowadays the term Bayesian refers to many different methods. All of these methods have in common, implicitly or explicitly, the basic assumption that the object of interest is a probability distribution, but they differ in how this probability is defined, characterised and interpreted.

Here, we use the experimental model introduced in Section 1.2. Equation (1.2) defines the relation between our physical measurement y, the parameter u and the noise η. Now, all these terms are drawn from probability distributions, and in consequence, (1.2) characterises the probability distribution of the measurement y if the parameter u is given. To solve the parameter identification problem, we are interested in the reverse condition, which we can compute by means of the Bayes' theorem (see for instance (Sanz-Solé, 1999)). We will provide the details in Chapter 2, but formally we have

$$\mathbb{P}(u|y) = \frac{\mathbb{P}(y|u)\mathbb{P}(u)}{\mathbb{P}(y)}. \tag{1.3}$$

Therefore, the knowledge about the parameter u given the data y is characterised in terms of the knowledge about the data y given the parameter u, which we have from (1.2); the knowledge about the parameter u regardless of the data, i.e. the marginal distribution of u; and the marginal distribution of y. We can interpret the latter as a normalisation term to ensure that the right hand side defines a probability. As we will discuss later, this term is in general difficult to compute, but we can use numerical methods to avoid its computation.

The marginal distribution of u is known as the prior, because it is the knowledge about the

parameter that we have *a priori*, before performing the experiment. It can be used to incorporate knowledge acquired in previous experiments, as well as constraints such as bounds on a parameter value. Our distribution of interest, the distribution of the parameter u given the data y, is known as the posterior.

We remark at this point that intuitively, we can expect to have well-posedness for this problem as long as the evaluation of the operator $\mathcal{G}$ is well-posed. Note that now our quantities of interest are probability distributions. Assuming that the prior for u is smooth in some sense, if G is smooth with respect to the parameters, a small change in u will produce a small change in the distribution of y given u (as a consequence of (1.2)), and in turn a small change in the distribution of the parameter u given the data y (as a consequence of (1.3)).

The Bayes' theorem characterises the distribution of the parameter given the data, but in general we will not be able to solve the problem analytically and we will need to rely on numerical methods to estimate the result. There are many available choices for the numerical approach, but we have to take into account now what kind of mathematical models we will be using, i.e. how do we evaluate $\mathcal{G}$.

1.3 Novel numerical methods

We focus on models given by partial differential equations. In general, we do not have analytical solutions to the relevant models, and we rely on numerical methods to compute approximate solutions. Note that even when a good numerical method is available, the implementation of the numerical solver in the computer may entail a significant amount of work due to the complexity of the mathematical model. Note that in our formulation of the inverse problem in (1.2), the evaluation of the operator $\mathcal{G}$ involves solving the mathematical model, and this is, in general, a costly operation in terms of computational resources.

At first sight, it seems reasonable to aim to improve the numerical methods to solve the PDEs. Since it is the most expensive operation, any improvement on that front will in turn provide a significant improvement on the estimation of the posterior distribution, but there are some limitations to this approach. First, it may simply not be possible to improve the numerical method, or it may take a long time. In particular, we may have solvers in place for a mathematical model that cannot be improved without re-implementing them from scratch. In this situation, it is natural to try to make use of the existing solver as it is.

Similarly, we can obtain more information about $\mathcal{G}$ from its differential, but computing the differential of $\mathcal{G}$ is typically an even more expensive operation than the evaluation of $\mathcal{G}$ itself, specially if the solver has not been originally implemented with this goal in mind.

Therefore, we need to find an alternative to improving the solver by itself. We keep in mind that our goal is to estimate the posterior distribution, and therefore what we need is an approach that allows us to use any existing solver. In this way, instead of improving every single evaluation of $\mathcal{G}$, we settle for improving the way we perform these evaluations.

The main improvement in computational resources in recent years did not come from the power of a single processor, but from the availability of parallel environments. Even with regular computers, it is not unusual to have 4 to 8 cores available, and in High Performance Computing (HPC) environments we can easily use a few hundred processors. We will exploit these observations to improve the computational efficiency of our algorithm.

There are numerical methods for PDEs exploiting the parallelism, but again we assume that either we have already a solver in place, or that the implementation of a parallel solver is not feasible for any other reason. Even if a parallel solver if available, we have to consider how does it scale in terms of the number of processors. In many cases, depending on the algorithm, the implementation and the computational environment itself, there is a limit on the improvement of the performance and it is not efficient to use more than a certain number of cores.

Therefore, the alternative is to aim for methods that allow evaluations of the operator $\mathcal{G}$ in parallel. This approach was introduced by Calderhead (2014). Instead of aiming to exploit the parallelism in every single evaluation, we would like to use many instances of an existing implementation of a solver for $\mathcal{G}$ and run them in parallel. We note that there are other Monte Carlo methods suitable for parallelisation. Particle filters (Sequential Monte Carlo, SMC) are a good example. See Del Moral (2004) for details, Kantas et al (2014) for an example of application to PDEs with a comparison between serial and parallelised implementations, and Das et al (2014); Murray et al (2016) for examples of parallelised particle filter algorithms.

In order to pursue this approach, our problems must meet several requirements. First, we assume that we have a solver for our system, that is robust with respect to the parameters. We will need to use the solver as a black box, that simply takes a parameter as input and gives the solution of the mathematical model as output. Any fine tuning of the solver must be either done in advance, or automated.

On the other hand, we also need to look for a method to approximate the posterior distribution that does not require anything else other than a solver for the PDEs. In particular, we cannot rely on methods that require the evaluation of the differential of $\mathcal{G}$. Therefore, we lose performance by restricting ourselves to simple methods, but we expect to compensate it with parallelism.

Finally, the characteristics of the computational environment limit the applicability of this approach. In principle, this approach will scale very well with the number of processors, and the main limitation will be how many of them can we use. From this point of view, there is an arbitrary decision on what is a reasonable maximum time to solve our problem. In general, we assume that computational times shorter than two days are acceptable.

1.4 Parameter estimation

In many situations, the parameters of the mathematical model are not scalar, but functions. A typical example involves time- or space-dependent parameters. It is therefore important that we use methodologies that allow the estimation of parameters lying in functional spaces.

One approach is to first discretise the parameters, and then apply the same methods that one would use to deal with scalar-valued parameters. This is for instance a common approach in statistics, where methods for estimating scalar parameters are well-developed. The main limitation of this approach is that usually it is not possible to prove any convergence result when the discretisation of the parameter is refined. This may lead to several issues, from a total lack of convergence to convergence to objects that do not have the intended features. Alternatively, one can devise a functional framework setting, and then apply the discretisation. Of course, we still need the convergence results, but now we know that the limit object exists, and at least in principle we are in a better position to look for convergence. We refer to Section 1.6 for references and a more complete literature review.

In terms of implementation, we must also keep in mind that the solver should be able to work with functional parameters independent of its discretisation, to avoid the need of re-implementation of the solver if we refine the parameter resolution.

1.5 On the interaction between mathematics and biology

The ideal interaction between mathematical modelling and biology is a cycle. We can start the cycle at almost any point, but to present the idea, let us assume that the experimentalist, after performing the experiment, have some data at hand, and some hypothesis about the underlying mechanisms on the system that they are studying.

The information about the underlying mechanisms defines the mathematical model. The mathematician uses the hypotheses of the biologist to derive a mathematical model. In many cases, the modelling process also needs some technical assumptions, that we assume reasonable from the biological point of view. With the model at hand, the mathematician can implement a numerical solver in order to compute the solution of the model. At this point, for any given numerical value for the parameters of the model, the mathematician is able the generate synthetic data. For parameters that have a physical interpretation, the biologist may provide estimates or constraints. In some cases, the parameters can be measured in independent experiments, but in many situations not much is known and one can only rely on rough estimates. Similarly, some parameters may not have a direct physical interpretation.

With this information at hand, the mathematician uses the experimental data to fit the parameters of the model, and reports back to the biologist. Depending on the questions of interest, the biologist evaluates this information in different ways. At the very least, the biologist have the knowledge to assess if the outputs of the model are reasonable and biologically meaningful, and also to check the feasibility of the values obtained for the parameters. In other situations, the biologist may go back to the laboratory to test the predictions of the models with new experiments. In any case, the biologist can report back to the mathematician, and the cycle start again, incorporating the new information.

We are specially interested in two characteristics of this cycle. First, as argued before, the modelling process, including the implementation of the model, is costly in terms of human resources. On the other hand, the parameter identification step is independent of the rest of the steps, in the sense that if the fitting method is general enough, it can be used with many different mathematical models, and when the cycle starts again, in can incorporate new information about the parameters.

The parameter identification stage is also critical for the cycle to work. Without it, mathematical models are limited to at most qualitative results about biological systems, and the biologist is limited in its tools to understand the experimental data from a quantitative point

of view. In order to facilitate the cycle, we aim to find a widely applicable methodology, that can make use of existing solvers without the need for re-implementation, and that provides as much information as possible about the parameters, including uncertainty quantification.

Our approach is to exploit parallelism in a suitable way, relying on existing theoretical results that ensure the robustness of the approach under general assumptions, therefore providing a tool to fulfill the parameter identification step in a manner that is as transparent as possible to the biologist. In this way, the biologist will be able to give a robust physical interpretation to the results.

1.6 A literature review on quantitative methods in biology

1.6.1 Optimisation methods and inverse problems

Optimisation methods and inverse problem frameworks are a natural candidate to provide quantitative methods for parameter fitting and model selection in experimental sciences. There are many available resources for the optimisation approach. The theoretical basis of optimisation problems with PDE constraints can be found, for instance, in Tröltzsch (2010). The book includes the basic results for both elliptic and parabolic systems of PDEs, but it is devoted to the general theory of optimal control. A good reference for the use of optimisation methods for parameter identification is Aster et al (2013). The book covers the main difficulties of parameter identification, both in the discrete and the continuous setting, and presents several regularisation techniques, as well and numerical algorithms to solve the problems. The book by Beck et al (1985) is slightly older, but offers a systematic presentation of the optimisation approach in the particular case of heat conduction problems, starting from the interpretation of the physical measurements and including a discussing of both analytical and numerical techniques. In terms of algorithms, Adby (2013) presents many of the most popular algorithms for optimisation, including Gauss-Newton optimisation and the Levenberg-Marquardt method.

When the properties of the solution operator are not well-known, or in general when its evaluation is not easy, there are randomised optimisation methods that are useful under general assumptions. A popular choice are the so-called genetic algorithms, inspired by the idea of natural evolution: one tries to optimise the fitness of the parameter by random changes—interpreted as the mutations of a living organism—and discarding the parameters that do not provide a good fit—in other words, eliminating the organisms that are not well adapted to the

environment. See Schmitt (2001) and Schmitt (2004) for a general introduction to the method and for examples of its applications.

In some situations, it is possible to perform an analytical parameter identification. For example, Friedman and Reitich (1992) present a method for parameter identification in a class of reaction-diffusion models. The applicability of this approach is very limited to particular classes of models. In contrast, we aim to provide methods that work in general settings to allow for a wider applicability of our computational framework.

1.6.2 The Bayesian approach to inverse problems

Our approach to parameter identification, including the idea of using a probability distribution as a model of the knowledge about a physical quantity, and to a certain extent, the idea of discretising the functional parameters at the end, can be found in Tarantola (2005). The book focus on the general ideas of the method and on the applications, specially to problems in geophysics, but it does not develop a sound theoretical framework.

Within the general framework of Bayesian methods, the book by Kaipio and Somersalo (2006) is a general reference for statistical methods for inverse problems. The theoretical basis for the application of Bayesian methods to inverse problem with PDEs can be found in Stuart (2010), and in a more detailed presentation, in Dashti and Stuart (2016). These are our two main references for the basis of our approach, and they provide all the necessary well-posedness results for the Bayesian inverse problem, as well as for the discretisation of functional parameters, and also for the study of the Markov Chain Monte Carlo methods (MCMC) to numerically approximate the posterior distribution.

There are many relevant aspects when solving inverse problems. Kaipio and Somersalo (2007) present a discussion of common issues when discretising an inverse problem, and in particular when one applies model reduction techniques. In particular, it includes the discussion of the so called inverse crime, when one tests an inverse problem method against synthetic data provided by the same model that one uses for the inverse problem. In this situation, the lack of experimental noise may lead to an overestimate of the performance of the method.

The estimation of the noise distribution is not trivial. Although in some cases the experimental noise can be approximated from the experimental data, other sources of error, for instance numerical error or simply the lack of accuracy of the model itself are more difficult to

incorporate. See for instance Huttunen and Kaipio (2007) for a discussion of the approximation errors in non-stationary inverse problems.

There are several considerations to take into account when defining the prior distributions for the parameters. When the prior information is scarce, the prior must model the lack of information. Kass and Wasserman (1996) discuss several rules for the choice of non-informative priors.

Note that our approach relies in particular on the idea that we have a good knowledge about the mathematical model and the measurement model, and in particular we assume that the noise distribution is known. When the noise distribution is completely unknown, or in general, when it is not possible to evaluate the solution operator efficiently, other Bayesian approaches can be suitable Wegmann et al (2009). See for instance Ross et al (2017) for an application of Approximate Bayesian Computation estimate parameters in a cell migration model, or Li et al (2017) for an application to evolutionary biology.

1.6.3 Estimation of the posterior distribution using Markov Chain Monte Carlo methods

A basic introduction to MCMC methods, in the context of estimating distributions in probability and statistics, can be found in Norris (1998). In Toni et al (2009), the MCMC method is presented and applied in the context of Approximate Bayesian Computation (ABC), in particular in the context of parameter identification for dynamical systems. Similarly, Brown and Sethna (2003) use MCMC methods in an statistical approach to parameter identification. The Metropolis-Hastings algorithm, which we will use for the MCMC, was presented first in Metropolis et al (1953) for symmetric proposal kernels, and later extended to general kernels in Hastings (1970). For a general discussion of the application of MCMC methods from the statistical point of view, see Kass et al (1998).

The proposal kernel is a crucial component of the Metropolis-Hastings algorithm. Cotter et al (2013) derive a number of proposal kernels by discretising a Langevin-type stochastic differential equation. The advantage of this approach is that it ensures the consistency of the kernel for infinite-dimensional parameters. On the other hand, efficient solvers may require the evaluation of the differential of the solution functional, and therefore lie out of our area of interest.

The choice acceptance probability of a proposed state depends on the proposal kernel to ensure that the MCMC approximates the desired probability. The choice of the acceptance probability is also discussed in Cotter et al (2013), and in general, the basic details of its properties in the context of MCMC methods can be found in Tierney (1998).

In the context of MCMC methods, there are several criteria to assess the convergence and quality of the Markov chain. Raftery and Lewis (1995) present some of the standard approaches.

A key component of our approach is the ability to parallelise the MCMC method. In order to do that, we will use the method presented in Calderhead (2014), where the details of the parallelisation approach are discussed, together with an application to systems of ODEs. The method is in turn based on the results from Tjelmeland (2004), although the focus there is different: Tjelmeland is interested on incorporating the information from rejected proposals in MCMC methods in order to improve the estimates of the mean value of the chain. Other Monte Carlo methods, such as Sequential Monte Carlo (SMC) also suitable for parallelisation; see Del Moral (2004); Kantas et al (2014); Das et al (2014); Murray et al (2016). We decided to use the parallel MCMC method proposed by Calderhead (2014) because it offers more flexibility in terms of implementation.

There are alternatives to the MCMC methodologies. For instance, Schwab and Stuart (2012) present the polynomial chaos technique in the context of Bayesian problems for PDEs. This approach has been discussed extensively, see for instance Lu et al (2015) for a discussion of the limitations of the approach, or the thesis Owen (2017) for an comparison between Gaussian processes and polynomial chaos for uncertainty quantification.

1.6.4 Application of Bayesian methods

Bayesian techniques are common for parameter fitting in statistics. The Bayesian approach is to a certain extent natural in statistics, because statistical models are defined in terms of probability distributions. For instance, Ma and Leijon (2011) present a Bayesian estimation method for beta mixtures. Fan et al (2012) present a variational learning approach, that in its basics is a parameter identification problem also defined in a Bayesian context. Similarly, the article by Ma et al (2015) includes a Bayesian method for matrix factorisation with bounded data. Although all these references apply the Bayesian techniques in a different context, and with a different interpretation, they provide the contrast necessary to understand our approach. As

a side note, the use of the term *Bayesian* is so popular that it becomes difficult to know what is the actual method or technique used without looking deeper into the results.

Closer to our problem of interest, Xun et al (2013) present a Bayesian approach to parameter estimation for PDE models. In the article, Xun et al represent the solution operator via a basis function expansion. Although the method is readily applicable, there is no guarantee that it will converge when the expansion is refined.

Although in a slightly different context, the book Robert et al (2014) presents a probabilistic approach to machine learning which is similar in spirit to our proposed approach to the Bayesian methods for parameter identification. In particular, it discusses several aspects of the discretisation of functional parameters, although without providing rigorous theoretical results. Similarly, Apte et al (2007) argue in favour of the Bayesian approach in the problem of data assimilation.

Uncertainty quantification from the posterior distribution is a well studied problem. See for instance Olbricht et al (1994) for an application to estimate thermodynamics properties of minerals, including the computation of credible regions as the regions with highest posterior density. The book by Soize (2017) provides a general introduction to the problem of uncertainty quantification. (Mitov and Stadler, 2017, 2018) implement a parallel methodology to evaluate uncertainty in models defined in terms of the Ornstein-Uhlenbeck stochastic process.

There are many examples of applications of Bayesian methods to PDEs in the literature. Outside the field of mathematical biology, Iglesias et al (2014) apply the Bayesian techniques in the sense of Stuart (2010) to the study inverse problems in subsurface flow. Similarly, Jiang and Ou (2017) combine model reduction techniques and Bayesian methods in the context of subsurface flow. Crestel et al (2017) develop a Bayesian method for the Helmholtz inverse problem. Perdikaris and Karniadakis (2016) use a different Bayesian approach for a parameter identification problem in the three-dimensional modelling of the human cardiovascular system.

1.6.5 Inverse problems in quantitative biology

Parameter identification problems are ubiquitous in mathematical biology. We note that even the examples of models given by ODEs provide interesting insights on the typical problems that one faces when performing parameter identification with experimental data from biological systems.

Ashyraliyev et al (2009) apply optimisation techniques to systems of ODEs modelling enzyme kinetics. In this case, uncertainty quantification is provided by means of confidence regions computed using the second derivatives of the cost functional. The problem of parameter identification for kinetic models is also studied in Gábor et al (2017), in this case focusing on large networks. Ashyraliyev et al (2009) show the importance of studying the correlations between parameters, using the example of a model for the *Drosopohila Melanogaster* gap gene. The study concludes that even when individual parameters cannot be identified, it is still possible to draw conclusions about the topology of the gene network. Note that an intrinsic advantage of the Bayesian approach, when applied to compute the posterior probability, is that one can observe the correlations between parameters in the joint probability distribution. These correlations may be invisible when one only computes the best value for the parameters, leading to an overestimation of the uncertainty (Sutton et al (2016)). Similarly, the correlations between parameters can suggest a non-dimensionalised form of the model to eliminate non-relevant parameters from the problem. A general overview of inverse problems in systems biology can be found in Engl et al (2009).

Optimisation techniques are also used in models given by PDEs. Garvie et al (2010) study the inverse problem of Turing patterns using an optimisation approach, in contrast to our Bayesian methods for the same problem as studied in Chapter 3 next. Similarly, Garvie and Trenchea (2014) study space-time distributed parameters in the Gierer-Meinhardt system, and Uzunca et al (2017) study the FitzHugh-Nagumo system. Stoll et al (2016) also study pattern formation problems using the optimal control approach. In Blazakis et al (2015) we find an application of the optimisation methods to a cell-tracking problem. The idea is to use optimal control techniques on geometric evolution laws for the cell shape. Croft et al (2015) discuss two alternative cost functionals for the identification of parameters in models of cell motility, using imaging data of migrating cells. Yang et al (2015) also consider the optimisation of geometric evolution laws in the context of cell migration, and presents a parallel, multigrid solver for the optimal control problem. Portet et al (2015) inspired our example in Chapter 4; they use a genetic algorithm to solve the problem of parameter identification for a family of reaction-diffusion models for the dynamics of the keratin network. They also use a model selection approach to find what are the relevant mechanisms in the system.

Bayesian techniques are widely applied in mathematical biology, although not so often with PDE models. There are many examples of applications in the context of statistical models

and ODEs. Engelhardt et al (2017) use a Bayesian approach to study hidden variables and model errors in molecular networks. Sgouralis et al (2017) apply the Bayesian methods to a topological data analysis problem, to reconstruct the trajectories of subcellular particles from microscopy imaging data. Hasenauer et al (2015) review the data-driven modelling approach to mathematical biology, with emphasis on multiscale models given by ordinary or partial differential equations, and also agent-based models. The article also discusses the pros and cons of optimisation vs Bayesian parameter identification techniques. Dewar et al (2010) present a Bayesian framework for parameter identification in a stochastic reaction-diffusion model for a morphogen. The study uses both synthetic and experimental data.

The uncertainty quantification for a discrete model of collective cell spreading is the main result of Vo et al (2015). In this work, the authors use imaging data and an Approximate Bayesian Computation (ABC) method to estimate the uncertainties in the model parameters.

ABC methods are very common. See for instance Wu et al (2014) for an application to stochastic models for complex regulatory networks, or Smith and Gröhn (2015) to an application in epidemiology.

Oden et al (2010) suggest a Bayesian inference approach for tumour growth models defined by PDEs, and in particular for diffuse interface models. The study remarks the importance of the parameter identification and uncertainty quantification for the validation of the mathematical model. Hawkins-Daarud et al (2013) review the Bayesian parameter identification problem for tumour growth models. Lima et al (2018) also addresses the parameter identification for a tumour growth model using Bayesian techniques, in this case fitting the model to a time-series of microscopy imaging data.

Perdikaris and Karniadakis (2016) study a model for the cardiovascular system using Bayesian techniques. The article presents a combination of Bayesian optimisation methods with multi-fidelity techniques, in order to reduce the number of evaluations of the solution operator of the complete model.

Similarly to the approach in Portet et al (2015), one can perform model selection using the results provided by Bayesian parameter identification. For instance, Vyshemirsky and Girolami (2008) present a Bayesian ranking method for biochemical models, and Toni et al (2009) use ABC to apply model selection methods to dynamical systems.

Lin et al (2016) present a Bayesian model selection to study growth patterns of fungi. In this case, the growth models are statistical models, and the authors use the model selection

methods to quantify the model error in three different cases. The study uses only synthetic data.

The book by Friedman (2018) includes a chapter on parameter identification in mathematical biology, but uses an analytical approach. As usual, the analytical techniques can only be applied to a limited class of models, yet they provide significant insights. Belgacem (2012) analyse a parameter identification problem in a class of advection-reaction-diffusion models. Although the study focuses on the theoretical aspects, the models studied there have many applications in mathematical biology. Similarly, Trillos and Sanz-Alonso (2017) provide the theoretical foundations for the Bayesian approach to parameter identification to fractional elliptic models.

1.6.6 Computational aspects of our Bayesian approach

For our implementation of the method, we will make an extensive use of the Python module *SciPy* Jones et al (2001–). The module provides tools for scientific computation, including the basic linear algebra routines and random number generators.

A general overview of the parallel programming paradigm can be found in Hamilton (2013). Although we will rely on standard modules for most of our parallel algorithms, it is necessary to understand the underlying parallel systems in order to design the algorithms.

Our approach is partly justified by the computational power that is available at present. It is important to keep in mind that only 10 years ago, it was not possible to apply the methods that we propose to obtain results in a reasonable time. See SIAM (2001) for a discussion of the impact of the increase of computational power in the sciences.

1.7 Thesis objectives

The main objective of this thesis is to develop a methodology for parameter identification problems, in particular in the context of quantitative biology, suitable for applications in a wide range of problems in a transparent manner, both in terms of the interpretation of the results and in terms of the requirements for the mathematical model.

In particular, we aim for a methodology that can use existing implementations of the numerical solver as a black box, with the only requirement to expose a routine that takes a parameter as an input, and produces the solution to the mathematical as an output. The solvers must be robust in order to evaluate the solution for a wide range of parameters.

Our methods must be suitable for models defined by PDEs, although they may be applied also in other situations such as ODEs or even of algebraic nature. Examples of the models that we have in mind include elliptic problems and reaction-diffusion systems. We also aim to be able to estimate functional parameters.

We are interested in providing uncertainty quantification for the parameters in a robust way. Here, robust means that we want to avoid arbitrary technical choices that may impact the final estimations. Furthermore, we aim to describe the theoretical results that justify the applicability of the methodology. In particular, we are interested in a method for which we can ensure well-posedness and convergence by using only results of the forward problem, i.e. results about the well-posedness of the mathematical model.

Finally, we want to show the applicability of the proposed methodology in different problems in the field of quantitative biology. We selected the applications to illustrate different aspects of our approach to relevant problems in mathematical biology.

In the first example, we study a parameter identification problem associated with Turing patterns. Turing patterns provide a theoretical mechanism for pattern formation in biological systems, and in some cases have been confirmed experimentally. See the references in Chapter 3 for details. We limit ourselves to synthetic data for the purpose of illustrating the applicability of the method, but otherwise we follow the same approach as one would follow with experimental data. The results also illustrate an application of our method to time-dependent parameters, in this case, the estimation of the time-dependent growth of the domain from the measurement of the Turing pattern at the final time.

Our second application is the study of the parameters of a model for the dynamics of the keratin network. Keratin is a protein responsible for many of the mechanical properties of the cell. Our parameter identification methods provide a method for estimating the assembly and disassembly rates of keratin, quantities that otherwise can only be measured indirectly. We present a detailed description of the data processing, from the raw data provided by the biologists to the tidy data that we use to fit the mathematical model. This illustrates how the

proposed methodology can deal with experimental data in a systematic manner. This application also illustrates the approach to define priors for functional parameters incorporating the hypotheses presented by the biologists.

Finally, we apply our techniques to the inverse problem of traction force microscopy (TFM). In TFM, the experimentalists measure the deformation of an elastic substrate when a cell moves, and the goal is to compute the force that the cell applies. It is a well-studied problem, but is usually approached by using either analytical techniques or an optimisation approach. In this case, we use again experimental data. Furthermore, in contrast with the previous examples, we use a solver implemented independently, to illustrate the ability of our implementation to invoke different solvers. The application is limited to the simplest experimental setting, that can be modelled as a two-dimensional, linear elasticity problem, but because of the flexibility of the method it could be extended to more complex experiments, such as three-dimensional TFM in anisotropic, non-linear media.

1.8 Outline of the methods

In order to accomplish our objectives, we use a Bayesian approach based mainly in the ideas presented in Tarantola (2005) and Stuart (2010). The Bayesian approach accomplishes naturally many of our objectives, in particular the uncertainty quantification, together with a transparent modelling that avoids as many technical choices as possible.

Stuart (2010) and Dashti and Stuart (2016) present the basic results to ensure the well-posedness of the approach, together with the MCMC methods to estimate the posterior distribution. We extend the MCMC from Stuart (2010) to parallel algorithms from Calderhead (2014). In this way, we provide a scalable algorithm that can be used in an HPC environment.

For the implementation of the methods, we use the programming language Python. The choice of the language is influenced by several factors. First, Python allows a quick development of the implementation, in a rapid cycle of test and improvement. Furthermore, Python can interact easily with programs implemented in other languages, therefore providing flexibility for the use of existing numerical solvers for the mathematical models. The main drawback of Python is its efficiency, but we use a carefully designed modular structure for our implementation. In this way, the components of our method can be re-implemented in the future to improve the efficiency of the program. Last but not least, the Python standard library

includes a parallel toolbox that provides simple tools for the parallel evaluation of functions. Therefore, for the first version of our solution, we can avoid dealing with the technical aspects of the parallel programming, such as distributed memory architectures.

1.9 Structure of the thesis

The rest of this thesis is organised as follows. In Chapter 2 we present our methodology in detail. We describe the Bayesian approach, its interpretation and the fundamental theoretical results that justify the robustness of the techniques. Then we proceed to present the parallelised MCMC method, and the description of the algorithm that we will use. We also discuss the details on the discretisation of functional parameters. We devote the last part of Chapter 2 to discuss the design of the algorithm implementation.

Chapters 3, 4 and 5 present the applications of our methods. These three chapters include detailed introductions to the corresponding problem and the discussion of the results.

Finally, Chapter 6 presents a brief discussion of the contribution of this thesis and the future work.

Chapter 2

Methods

2.1 General assumptions

For our methodology, we will follow the ideas in Tarantola (2005), Stuart (2010) and Dashti and Stuart (2016). Stuart (2010); Dashti and Stuart (2016) provide the theoretical results to make the approach rigorous, as well as the rigorous mathematical setting. In this chapter, we present our approach to these ideas, and describe the methodology that we will follow. All the results about the Bayesian approach that are presented in this section and the following are available in the references mentioned above.

The first step in to give a rigorous formulation of the parameter identification problem. We are interested, in particular, to give a clear interpretation of all the components of the formulation, in order to then be able to provide a robust interpretation of the solution. In particular, we want to limit the number of decisions that cannot be justified or understood in terms of the physical system of study.

Since one of our objectives is to solve inverse problems involving time- or space-dependent parameters, we shall frame our formulation in the context of Banach spaces. Finite dimensional parameters, although much simpler and requiring less assumptions, can be interpreted as a particular case of the Banach space setting. In this way, we keep our approach as general as possible.

To make the ideas precise, let U be a Banach space, that we will interpret as the parameter space; let X be a Banach space, the space of solutions to our mathematical model, and let $Y \subset \mathbb{R}^M$ be an Euclidean space, the space of the observations. We note that here we assume already that our observations will come from a physical, real-world measurement, and therefore they will always be finite dimensional.

For a given mathematical model, we denote by G the solution operator. G maps parameters to solutions of the mathematical model,

$$G : U \longrightarrow X. \tag{2.1}$$

For a given parameter $u \in U$, $G(u) \in X$ is the solution of the mathematical model, for example, the solution of the PDE.

In general, we cannot observe $G(u)$ in a physical experiment. The real world observation is given by a mapping $\mathcal{P} : X \to Y$, that maps solution to the mathematical model $x \in X$ to the physical observation $y \in Y$. In the simplest case, $\mathcal{P}$ represents a projection of X onto a finite dimensional space. This is the case, for instance, when our observation is a finite number of measurements of a function. $\mathcal{P}$ can also represent more complex measurements.

We denote by $\mathcal{G} : U \to Y$ the composition $\mathcal{P} \circ G$. We refer to $\mathcal{G}$ as the observation operator, and it maps a parameter $u \in U$ to a the physical observation of the solution of the mathematical model, $\mathcal{G}(u) = \mathcal{P}(G(u))$. Note that if we assume that $\mathcal{P}$ has good smoothness properties, the well-posedness of the mathematical model determines the properties of G, and in turn the properties of $\mathcal{G}$. In particular, when thinking about the general setting regardless of the functional setting, one can abuse the notation and assume that $\mathcal{G} = G$. The distinction becomes only important when setting the functional spaces for solutions and observations.

In a physical measurement, we will also find errors or noise. Therefore, we make the following assumption for the structure of the experimental measurement. Let $y \in Y$ be the result of a physical observation, and let $\eta \in Y$ be an error term. We assume

$$y = \mathcal{G}(u) + \eta, \tag{2.2}$$

for a certain parameter $u \in U$.

Note that we could include other types of noise. For a general model, we could let $\eta \in S$, where S is a suitable space, and define the observation operator as

$$\mathcal{G} : U \times S \longrightarrow Y. \tag{2.3}$$

Here, the noise η can affect the solution in non-linear ways. The interpretation of this error

term is also open to other possibilities rather than experimental noise. For instance, η can incorporate modelling errors in the inverse problem (see for instance the approach in Engelhardt et al (2017)).

There are some limitations to the use of additive noise. Some problems, for instance geometric evolution problems, are described by models that give as solution objects that do not lie in Banach spaces. Even if the solution can be embedded in Banach space, the choice of the Banach space must be informed by what addition will mean in terms of the perturbation of a measurement. We will not be studying this type of problems in this thesis.

In this work we are only interested in modelling the experimental error, we restrict ourselves to the additive noise as described in equation (2.2). We assume in particular that the experimental error is additive. Although this is a very general assumption and it does not restrict the applicability, it is important to keep it in mind to ensure the consistency of our modelling approach with the physical experiments.

Note that the additive noise is very suitable when we assume that our mathematical model is a model for the mean behaviour of the system of interest. In this case, from a series of physical measurements we will compute the average observation, and use it to fit the mathematical model. The Central Limit Theorem ensures that for a large enough number of measurements, we will have an additive Gaussian error in our average observation. Furthermore, we will be able to estimate the standard deviation of the error also from the physical measurements. In general, finding the noise distribution is not trivial (see for instance Kaipio and Somersalo (2007); Huttunen and Kaipio (2007)).

To sum up, equation (2.2) is our main assumption in terms of the experimental measurement, and it is well-defined in a Banach space setting, therefore allowing us to describe infinite-dimensional parameters. The next step is to incorporate in it the idea of modelling the knowledge of a physical quantity as a probability distribution. Stuart (2010) describes this setting in similar terms.

2.2 Bayesian framework

The main idea underlying our approach is that we can model our knowledge about a physical quantity as a probability distribution, as expressed in Tarantola (2005). This notion distinguishes the approach from a minimisation method, by giving a complete robust interpretation

to all the components of the framework, avoiding any arbitrary choices. If a quantity is known with infinite precision, then the probability distribution associated with this knowledge will be concentrated at one point—a Dirac delta; the less we know, the more spread the distribution will be.

Note that once we define the distributions of the noise η, and the parameter u, equation (2.2) gives a complete characterisation of the distribution of the physical observation y given a parameter u. Therefore, we discuss now the definition and interpretation of these distributions.

2.2.1 Parameters

Note that the distribution for the parameter u incorporates the knowledge about the parameter regardless of the measurement, and it is known as the prior. It is important to note that this distribution may incorporate information from independent measurements, but it must be independent of the observations that we intend to use for the parameter identification problem, otherwise we may be incorporating the same information twice.

In the applications, several pieces of information can be modelled in the prior distribution for the parameters. For example, if a parameter is known from independent measurements with a given precision, this leads naturally to a prior distribution. Similarly, the prior distribution can incorporate bounds on the parameter, or restrictions such as positivity. In the Chapters 3 and 4 we will see practical examples on how to model this features.

An more difficult question is how to model the *lack* of knowledge. Priors to model the absence of informations are known as non-informative priors. It is a subject of debate what is the actual interpretation of non-informative, that in turn is linked to the more philosophical question of the interpretation of probabilities, i.e. what do we mean exactly by "a probability distribution models our knowledge". We will not explore this questions, see for instance Williamson (2010) for a discussion.

A simple rule to define a non-informative prior is the principle of indiference, i.e. assign equal probabilities to all the possibilities. This leads to uniform distributions, when they are well-defined. There are also other approaches, for instance the use of priors that maximise the Shannon entropy of the probability distribution, relying on the interpretation that the larger the entropy, the less information is provided by the distribution (Jaynes (1968)).

Unless stated otherwise, we denote the prior distribution by $\mathbb{P}_0$, a probability distribution over the space U. When a probability density function (PDF) for $\mathbb{P}_0$ is available, we denote this PDF by μ_0.

In many situations, we can define the prior as a Gaussian distribution. A simple example of this situation is when we are defining a prior for a scalar quantity, that have been measured independently several times, and we define the prior by means of the Central Limit Theorem as the average of the measurements, with the corresponding standard deviation. Gaussian priors also have the advantage of being well-suited for the theoretical results.

2.2.2 Noise

Although technically noise is just another probability distribution, the interpretation of the noise is different from the interpretation of the priors, and in consequence the treatment is slightly different. Note in particular that if we would assume the general definition of the observation operator given in equation (2.3), at the mathematical level the role of the parameter and the noise term is similar.

In our methodology, we assume that the knowledge about the noise comes from the experimental setting. In the ideal situation, we will be able to estimate the noise also from the data, but in general, we assume that the noise distribution is informed by the experimental setting, and it is not defined following only technical criteria.

We denote by $\mathbb{Q}_0$ the probability distribution associated with the noise term η in equation (2.2). $\mathbb{Q}_0$ is a distribution over the space Y, and we can assume without loss of generality that it is a distribution centered at zero. Otherwise, we can include a translation in the observation operator $\mathcal{G}$. When a probability density is available, we denote by ρ the PDF of $\mathbb{Q}_0$.

In this work, we will restrict ourselves to Gaussian distributions for the noise. There are two main reasons for this restriction: first, although some of the results can be generalised easily to other distributions, Gaussian distributions are easy to define and manipulate, both in the theory and in the numerics; second, as indicated above, in many situations we have averaged data that leads naturally to a Gaussian noise distribution. This is the situation in the problems that we study in Chapters 3, 4 and 5.

Since we assume that the noise distribution $\mathbb{Q}_0$ is a centered Gaussian, it is completely determined by its covariance. Unless stated otherwise, we denote this covariance by Σ. This fact,

together with an explicit density, are the main advantages that motivate the use of Gaussian distributions.

So far, we have defined the distributions of the parameter and the noise, and together with equation (2.2) we have a relation between these two quantities and the experimental observations. The next step is to extract the information about the parameter u when the observation is given.

2.3 Bayes' theorem

Let us recall the basic idea from the Bayes' theorem. Given two events A, B and a probability $\mathbb{P}$, if we know the conditional probability of A given B, $\mathbb{P}(A|B)$, and the respective marginal distributions for A and B, $\mathbb{P}(A)$ and $\mathbb{P}(B)$, the Bayes' theorem characterises the reverse conditional probability,

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)}. \tag{2.4}$$

In our problem, equation (2.2) characterises the probability distribution of the observations, given a parameter u. Intuitively, the Bayes' theorem will therefore give us the reverse condition, the distribution of the parameter given an observation y. This is our object of interest, since this distribution models the knowledge about the parameter that we obtain when we incorporate the physical observations.

Let $\mathbb{Q}_u$ be the probability distribution of the observation y given a parameter u. From equation (2.2), $\mathbb{Q}_u$ is the probability distribution of the noise shifted by $\mathcal{G}(u)$. Assume that $\mathbb{Q}_u \ll \mathbb{Q}_0$ $\mathbb{P}_0$-almost surely. Then, there exist a potential $\Phi : U \times Y \to \mathbb{R}$ such that

$$\frac{\mathrm{d}\mathbb{Q}_u}{\mathrm{d}\mathbb{Q}_0}(y) = \exp\left(-\Phi(u, y)\right). \tag{2.5}$$

The existence of such potential is just the definition of the Radon-Nikodym derivative. Φ is known as the negative log-likelihood. We can now formulate the Bayes' theorem.

Theorem 1 (Bayes' theorem)**.** *Assume that the potential $\Phi : U \times Y \longrightarrow \mathbb{R}$ is measurable with respect to the product measure $\mathbb{P}_0 \times \mathbb{Q}_0$ and assume that*

$$Z := \int_U \exp\left(-\Phi(u, y)\right) \mathbb{P}_0(du) > 0.$$

Then, the conditional distribution $\mathbb{P}(p|y)$, denoted by $\mathbb{P}^y(p)$, exists, and it is absolutely continuous with respect to the prior distribution $\mathbb{P}_0$. Furthermore,

$$\frac{d\mathbb{P}^y}{d\mathbb{P}_0} = \frac{1}{Z}\exp\left(-\Phi(p;y)\right).$$

Proof. See Dashti and Stuart (2016), theorem 14. □

We remark that Theorem 1 provides a characterisation of our distribution of interest, $\mathbb{P}^y$, but in general this characterisation is not explicit and we will need to rely on numerical methods to approximate it. We will present the numerical methods in subsequent sections. Following the notation introduced above, we will denote by μ^y a PDF associated with $\mathbb{P}^y$.

A particular situation, when both the noise and the prior distributions are Gaussian, it turns out that the posterior distribution $\mathbb{P}^y$, that is, the Gaussian class of distributions is a Bayesian conjugate to itself. In particular, it is easy to show that if the prior is a Gaussian with center at $u_0 \in U$ and covariance Γ, and we denote by $\|\cdot\|_\Lambda$ the norm in Y weighted by Λ, we have

$$\Phi(u;y) = \|y - \mathcal{G}(u)\|_\Sigma^2 + \|u - u_0\|_\Gamma^2.$$

For example, if both the noise and the prior distribution are standard Gaussian distributions with unit or identity covariance, both norms in the previous expression are the standard L^2-norm.

Note that in the context of an optimisation approach, the goal will be to minimise exactly this cost functional, but here we can see that the Tykhonov regularisation, that in the optimisation approach is a technical addition, is actually defined by the prior distribution, i.e. about the knowledge *a prior* about the parameter.

Note that the constant Z in Theorem 1 can be interpreted as a normalisation constant. In general, Z could be very expensive to compute, since it is an integral over the whole parameter space, but we will see that there are suitable numerical algorithms that do not require the explicit computation of Z.

The formulation of Theorem 1 is general in order to be applied to Banach spaces, but the theorem reduces to the well-known finite dimensional Bayes' theorem in the case of finite-dimensional spaces. In terms of PDFs, following the notation presented above, the theorem

reads

$$\mu^y(u) \propto \rho\left(y - \mathcal{G}(u)\right) \mu_0(u).$$

2.3.1 Well-posedness

One of the main drawbacks of the optimisation approach is that it is typically difficult to prove the well-posedness of the inverse problem. As detailed in Chapter 1, even when the forward problem is well-posed, the inverse problem understood as a minimisation problem can be ill-posed.

Using only properties of the forward problem, the following theorem guarantees that the parameter identification problem is well-posed in the Bayesian framework: the posterior distribution exists, and it depends continuously on the data y. Let us denote by $B(0,r) \subset U$ the ball with centre at the origin and radius r, and by $\|\cdot\|_\Sigma$ the Euclidean norm with weight Σ.

Theorem 2 (Well-posedness of the Bayesian inverse problem)**.** *Assume that $\mathbb{Q}_0$ is a Gaussian distribution with covariance Σ, and that the potential is $\Phi(u;y) = \|y - G(u)\|_\Sigma^2$. Assume that the observation operator $\mathcal{G}$ satisfies the following conditions.*

i) For all ε, there exists $M \in \mathbb{R}$ such that for all $u \in U$,

$$\|\mathcal{G}(u)\|_\Sigma \leq \exp\left(\varepsilon \|u\|_U^2 + M\right).$$

ii) For all $r > 0$, there exists a K such that for all $u_1, u_2 \in B(0,r)$,

$$\|\mathcal{G}(u_1) - \mathcal{G}(u_2)\|_\Sigma \leq K \|u_1 - u_2\|_U.$$

Then, the distribution $\mathbb{P}^y$ defined in Theorem 1 exists, and depends continuously on the data y in the sense of the Hellinger distance.

Proof. See Stuart (2010), theorem 4.2. □

Note that the first condition is a bound on the growth of the solution with respect to the parameter u, whilst the second condition is the Lipschitz continuity of the observation operator.

We remark that both conditions in the theorem refer to the forward problem. Therefore, these results overcome at once many of the difficulties when studying the well-posedness of the inverse problem, by reducing it to the study of the well-posedness of the forward problem.

In any case, well-posedness of the forward problem is expected, otherwise we may not have a well-defined solution operator G. The existence of G is equivalent to the existence and uniqueness of solutions of the mathematical model, and the second condition, Lipschitz continuity of $\mathcal{G}$, is the problem of continuous—actually, Lipschitz—dependence with respect to the parameters of the model.

The first condition in Theorem 2 is not necessary to obtain well-posedness if the parameter space U is finite dimensional. This condition is required in the infinite-dimensional setting to ensure a fast enough decay of the probability distribution.

2.4 Sampling algorithms

Markov Chain Monte Carlo methods (MCMC) are a family of methods that produce a Markov chain with a given distribution (Norris (1998)). For the problem at hand, the target distribution is the posterior. Therefore, for each new state of the Markov chain, we obtain a sample of the posterior distribution.

MCMC methods are robust but slow. The distribution of the Markov chain converges, under general conditions, to the target distribution, but long chains are necessary to obtain good approximations (see for instance Norris (1998)). Also, the methods are inherently sequential. Furthermore, in the parameter identification framework, evaluation of the acceptance probability typically involves, at least, one evaluation of the potential Φ, and in consequence, one evaluation of the observation operator G. This is an expensive operation in terms of computing time in cases where the model is a system of PDEs because it entails solving the system.

To overcome these difficulties we shall use a parallel Metropolis–Hastings algorithm (Calderhead (2014); Tjelmeland (2004)). The key idea is to generate an *N-dimensional* Markov chain, such that its distribution is N copies of the target distribution. This can be done in a way that allows parallel evaluations of the potential Φ.

Let x_k be the current state of the Markov Chain, and let $k(x_k, x)$ be a proposal kernel—the probability to propose a state x if the present state is x_k. For ease of presentation assume that

the proposal kernel is symmetric, i.e. $k(x_k, x) = k(x, x_k)$. Let $a(x_k, x)$ be the probability of accepting a new state x if the present state is x_k.

The choice of the proposal kernel k is critical to the performance of the algorithm. Furthermore, the choice of the acceptance probability a depends on the proposal kernel in order to ensure that the Markov chain is reversible with respect to the target probability. Reversibility implies, in turn, that the Markov chain preserves the target probability. We refer to Tierney (1998) for details on the choice of a in general, and to Cotter et al (2013) for the case of infinite dimensional states.

In the present work, we restrict ourselves to simple proposal kernelss. Given the prior distribution $\mathbb{P}_0$, take the proposal kernel k satisfying

$$k(x_k, \cdot) \sim \mathbb{P}_0(\cdot).$$

Together with the acceptance probability given by

$$a(x_k, x) = \min\left\{1, \exp\left(\Phi(x_k) - \Phi(x)\right)\right\}, \tag{2.6}$$

this choice corresponds to the standard Metropolis-Hastings method known as independence sampler. We refer to Cotter et al (2013) for other choices suitable for infinite dimensional problems. Note that the acceptance probability in this case, as well as for many other popular choices, depends only on the potential Φ.

Although the independence sampler is useful for a general exploration of the parameter space, it does not offer any tunable parameters, and therefore it may work poorly when the values of the potential Φ have significant differences depending on where it is evaluated.

Another popular sampler is the preconditioned Crank-Nicholson (pCN). For a Gaussian prior with mean 0 and covariance Γ, the proposal kernel for the pCN is given by

$$k(x_k, x) \sim \mathcal{N}((1-\beta)^{\frac{1}{2}} x_k, \beta^2 \Gamma),$$

where $\mathcal{N}(m, C)$ denotes a Gaussian distribution with mean m and covariance C, and $\beta \in [0, 1]$. The acceptance probability is the same as in the independence sampler, (2.6). With the pCN, we can adjust the variance of the proposals by tuning the parameter β. The proposals account for the local information provided by the current state x_k. For $\beta = 1$, we recover the

independence sampler. Taking smaller values of β entails that the samples are closer to the current state and the acceptance probability increases.

Both independence sampler and pCN are suitable for infinite dimensional parameters. Cotter et al (2013) consistently derive these and other proposal kernels by discretising a Langevin-type stochastic differential equation.

We note that for some problems of interest, the evaluation of the potential Φ is computationally intensive, and therefore proposal kernels that require evaluations of quantities like $\nabla\Phi$ may not be feasible.

In the standard Metropolis-Hastings method, given a current state x_k, we generate a proposal x from the proposal kernel. We accept the new proposal with probability $a(x_k, x)$, i.e. $x_{k+1} = x$ with probability $a(x_k, x)$, and $x_{k+1} = x_k$ otherwise. Note that the evaluation of $a(x_k, x)$ involves the evaluation of $\Phi(x)$, which in turn involves the evaluation of the observation operator G. Since this evaluation of G is an expensive operation, the aim of the parallel Metropolis-Hastings method is to evaluate Φ in parallel for many proposals.

The parallel Metropolis–Hastings algorithm goes as follows. We generate N new proposals $\{x^j\}_{j=1}^N$ from the proposal kernel $k(x_k, \cdot)$. Take $x^0 = x_k$. Then, in parallel, we evaluate the potentials $\Phi(x^j)$, $j = 1 \dots N$. Note that this will be done by N instances of the PDE solver running in parallel, but the solver itself need not to be parallel. Parallelising the solver for the model will reduce the time spent solving the model system and therefore is, in some cases, worth implementing. With the values of $\Phi(x^j)$ at hand, the acceptance probability of each proposal is computed by finding the stationary distribution of a Markov chain with $N + 1$ states, given by the transition matrix

$$A(i,j) = \begin{cases} \dfrac{1}{N} a(x^i, x^j), & \text{if } j \neq i, \\ 1 - \displaystyle\sum_{j \neq i} A(i,j), & \text{if } j = i. \end{cases}$$

Finally, we sample N times from the stationary distribution to produce N new states (see Algorithm 1). Note that this approach is nonintrusive, in the sense that it does not require modifications of the solver of the PDEs.

There are many other sampling methods to approximate a probability distribution characterised by the Bayes' theorem. The parallelised Metropolis-Hastings method presented above is suitable when a PDE solver for the mathematical model is already available, and the only

Algorithm 1 One step of the parallel Metropolis–Hastings algorithm. This will generate N new samples.

1. Draw N new points $\{x^j\}_{j=1}^N$ from the proposal kernel $k(x_k, x^j)$. Take $x^0 = x_k$.
2. Evaluate, in parallel, $\Phi(x^j)$, $j = 1 \dots N$.
3. Compute the acceptance probabilities of the proposals, given by the stationary distribution of a Markov chain with transition matrix

$$A(i,j) = \begin{cases} \frac{1}{N} a(x^i, x^j), & \text{if } j \neq i, \\ 1 - \sum_{j \neq i} A(i,j), & \text{if } j = i. \end{cases}$$

4. Sample N times from the stationary distribution to produce $x_{k+1}, \dots, x_{k+N}$.

interest is in speeding-up the computations using a parallel HPC environment, without doing any changes to the solver. In other words, it is a parallel sampling algorithm that works unintrusively with a given PDE solver to evaluate the potential Φ. Furthermore, the Metropolis-Hastings methods work well for problems with infinite dimensional parameters (Stuart (2010); Dashti and Stuart (2016)).

We remark that in this context, we assume that we can explicitly compute the potential Φ, i.e. we can compute the likelihood. This is a consequence of (2.2), and allows us to incorporate in the modelling the information available on the experimental noise. In other problems, specially when the noise distribution is completely unknown, or when the likelihood cannot be computed in general, Approximate Bayesian Computation can be more suitable. See Ross et al (2017) for an example of application of Approximate Bayesian Computation in mathematical biology.

2.5 Discretisation of measures in Banach spaces

In this section, we describe the basic ideas to discretise probability measures on Banach spaces. Note that in principle there is no difficulty in order to describe probabilities for finite dimensional parameters. In contrast, we need to describe the probabilities for infnite-dimensional parameters taking into account that we may refine the discretisation and expect convergence to a limit probability, i.e. to a probability measure in a Banach space.

There is a natural approach to discretise an element of a Banach space: we fix a basis of the space, we express the element in this basis, and then we cut the expansion to a finite number of elements, in other words, we project it a finite dimensional subspace spanned by

a finite number of basic functions. Any element of the Banach space is characterised by the coefficients in the expansion.

Using this idea, we can define a probability distribution in a Banach space by giving a basis and a probability distribution for the coefficient of each basic function. In this way, we can also naturally use the projection of this probability distribution onto a subspace spanned by a finite number of elements of the basis.

In general, let U be a separable Banach space, and let $\{u_i\}_{i\in\mathbb{N}}$ be a basis. We assume that for all $i \in \mathbb{N}$, $\|u_i\| = 1$, where $\|\cdot\|$ denotes the norm in U. Let $\{\omega_i\}_{i\in\mathbb{N}} \subset \mathbb{R}$ be a deterministic sequence, and let $\{\xi_i\}_{i\in\mathbb{N}} \subset \mathbb{R}$ be an independent, identically distributed random sequence. Later on, it will also useful to have a distinguished element $m_0 \in U$, that we will use to adjust the mean of our prior.

With this notation at hand, we can define a random element of U by

$$u = m_0 + \sum \omega_i \xi_i u_i. \tag{2.7}$$

The sum can be truncated to a finite number of terms to have a finite dimensional approximation of u.

Note that this approach does not actually require for $\{u_i\}_{i\in\mathbb{N}}$ to be a basis of U. Given a sequence $\{u_i\}$, not necessarily a basis of U, this formulation will provide elements of a separable subspace $U' \subset U$.

We refer to Dashti and Stuart (2016) for detailed constructions of several types of prior, together with convergence results to ensure the convergence of the sum in equation (2.7). We discuss here briefly only two examples, uniform and Gaussian priors.

We define uniform priors by assuming that $\xi_i \sim \mathcal{U}(-1, 1)$, i.e. a uniform distribution. Note that in this case, since ξ_i are bounded and we assumed that the basis functions are normalised, we can expect to have convergence in equation (2.7) as long as the weights ω_i decay quickly enough. In particular, if $\{\omega_i\}_{i\in\mathbb{N}} \in \ell_1$, the sum converges. See Dashti and Stuart (2016) for details.

To define Gaussian priors, we assume that $\xi_i \sim \mathcal{N}(0, 1)$. In this case, the limit measure is actually a Gaussian over U. In fact, for a Gaussian measure $\mathcal{N}(0, \mathcal{C})$, one could use the Karhunen-Loeve basis associated with $\mathcal{C}$ to a create the expansion in equation (2.7). Note that the Gaussian priors the equivalent to the so called Gaussian processes (Rasmussen (2004);

Murphy (2012)), popular in machine learning.

As discussed in Section 2.2.1, the prior distribution models our *a priori* knowledge about the parameters. For finite dimensional parameters, we can use the modelling approaches commonly used in statistics (see for instance Kass and Wasserman (1996); Gelman et al (2017)). In the case of priors for infinite-dimensional parameters, once we have decided for a family of priors—i.e. once we fixed the distribution ξ_i—, there are two objects that can be adjusted: the sequence of weights ω_i, and the actual Banach space, by the choice of the basis functions u_i. In Chapters 3 and 4 we see examples of how to perform this choices.

To sum up so far, we presented a theoretical framework for parameter identification, with a robust formulation that depends in principle only on the well-posedness of the forward problem. All the elements of the formulation have a clear physical interpretation, thus allowing for a physical interpretation of the results. We described scalable parallel algorithms to exploit the framework numerically, and we described the ideas to design priors in infinite-dimensional spaces. We will now describe design of the implementation.

2.6 Implementation

The language of choice for the implementation if Python. Python is a scripting language that allows for the quick development of applications, with the drawback of a low efficiency in comparison to compiled languages such as C (Ritchie et al (1988)) or field specific languages such as Julia (Bezanson et al (2012)). Our implementation is original, and incorporates the requirements to apply the framework presented in the previous sections together with the parallel algorithm by Calderhead (2014).

Python have a large module ecosystem that facilitates the quick implementation of common tasks. In particular, Scipy (Jones et al (2001–)) is a collection of libraries for scientific computation, including linear algebra routines and random number generation. The Standard Python Libraries include the module *multiprocessing* for the implementation of parallel routines. Note that traditionally Python struggled with parallel tasks due to the so called Global Interpreter Lock (GIL), a lock to protect Python programs from errors in memory management (Beazley (2010)). This lock effectively forbids the use of more than one thread, but the module *multiprocessing* overcomes this difficulty by creating multiple instances of the Python interpreter at the cost of a higher memory use.

2.6.1 Structure of the implementation

We have two main goals in mind when designing the implementation of our methodology. First, we want to keep the structure as modular as possible, to allow the re-implementation of the different components in the future. In this way, parts of the code could be rebuild without affecting the rest. The second goal is to keep the structure of the software as close as possible to the theoretical framework. This provides an advantage to the final user, that can clearly follow the logic of the software as long as she or he knows the theory.

In principle we assume that our software has access to a routine that evaluates the negative log-likelihood Φ. Such a routine has actually two distinct parts, the solver of the mathematical model and the actual computation of Φ, that compares the solution to the model with the experimental data. We consider this evaluation of Φ as only one routine because the output of the solver varies significantly depending on its implementation, whilst the results of the evaluation of Φ is a single real value. In this way, we can keep our implementation open to virtually any solver because we do not need to deal with any particular format for the solution of the mathematical model. The routine to evaluate Φ has only one argument, an array of real values that represent the parameter. The serialisation of the parameter, in other words, its representation as an array of real values, and in general, all the routines to handle the parameters, form another independent piece of the software.

We remark that Φ is the *only* routine that depends on the solver for the mathematical model, and therefore our implementation is as flexible as possible to the use of different solvers, as long as they can expose Φ.

The are three main classes in our implementation, linked to different elements of the theoretical framework. The class *mc* is the main data structure. It contains the Markov chain, and includes the methods to add new states to the chain, save to a file, and load it. It also contains metadata about the Markov chain, such as the seeds for the random number generators. In this way, it is possible to reproduce exactly the same Markov chain twice. Note that we also save in the Markov chain the value of all the evaluations of Φ, accepted or not. Since the evaluation of Φ is the most time-consuming part of the code, it is efficient to save it for possible re-use.

The class *pa* implements all the routines to handle the proposal and acceptance of new states in the Metropolis-Hastings algorithms. We implemented the independence sampler and the pre-conditioned Crank-Nicolson sampler. The reason to have the proposal and acceptance

routines as a separate class is that, as mentioned in Section 2.4, the acceptance depends on the proposal kernel. Therefore, it is a good design to allow the user to choose only the proposal method, and let the class *pa* adjust the acceptance accordingly. Note that for the proposals that we implemented that acceptance does not change, but we will implement new proposal methods in future versions of the software. An instance the *pa* class is configured with the prior.

Finally, the class *pmh* implements the parallel Metropolis-Hastings method. To create an instance of this class we require an instance of the *mc* and *pa* classes, together with metadata for the simulation such as maximum running times and target number of samples, and with the Φ evaluation routine.

As part of this thesis, we also implemented a post-processing module, that includes the routines to read the data from the Markov Chains and analyse it. This module allows for the computation of the mean, standard deviation and other statistics, the computation of credible regions in two and three dimensions, and it includes several tools for data visualisation.

2.7 Description of the methodology

We summarise here our approach to the parameter identification problems. Given a mathematical model, we begin by defining the priors for the parameters according to the information provided by the experimentalist. Using the experimental data, and the knowledge about the actual experiment, we define the noise distribution and in turn the log-likelihood.

If the solver for the mathematical is already implemented, we only need to implement the log-likelihood routine and the priors. We can then run the algorithms to produce MCMC. Typically we will need a few runs to adjust the proposal methods, and then we can proceed to generate several independent Markov Chains, to ensure a good exploration of the parameter space.

With the Markov Chains at hand, we can produce the results of interest for the biologist, either numerical or in the form of data visualisation for the parameters and their credible regions. Note that in principle we did not make any technical choice that could affect the results. At this point the biologist may update the hypothesis about the parameters or the model, and the cycle starts again.

Chapter 3

Parameter identification for Turing systems on stationary and evolving domains

3.1 Introduction

Turing systems are a family of reaction-diffusion systems introduced by Turing Turing (1952). Originally intended to model morphogenesis, they have since found many applications in mathematical biology and other fields Lacitignola et al (2017); Das (2016); Wang et al (2016b); Hu et al (2015); Guiu-Souto and Muñuzuri (2015). The main feature of these models is that small perturbations of the homogeneous steady states may evolve to solutions with non-trivial patterns Murray (2011, 2013). An example of such a model is the Schnakenberg system, also known as the activator-depleted substrate model Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979). This model has been widely studied, both analytically and numerically Garvie et al (2010); Venkataraman et al (2012); Madzvamuse et al (2005). Note that for some reaction-diffusion systems with classical reaction kinetics the Turing space can be obtained analytically, thereby offering us a bench-marking example for our theoretical and computational framework (see for instance Murray (2013)).

The parameter identification for Turing systems, using Turing patterns as data, was studied in Garvie et al (2010), both with Schnakenberg kinetics and with Gierer-Meinhardt kinetics, but using an optimal control approach. Garvie and Trenchea (2014) also used an optimal control approach to identify space-time distributed parameters in a system with Gierer-Meinhardt

kinetics. In contrast, here we use a Bayesian framework, that in particular allows the consistent computation of credible regions for the estimated parameters. To the best of our knowledge, this is the first study to address the inverse problem for Turing patterns in the Bayesian framework, and in particular, in growing domains. Uzunca et al (2017) studies a convective FitzHugh-Nagumo system which is similar to the system arising in growing domains, and identifies finite-dimensional parameters using optimisation techniques.

Our first goal is to identify a time-dependent growth function for a reaction-diffusion system posed on a one-dimensional growing domain. For this case, the rest of the model parameters are considered known and fixed. This example illustrates an application of our approach to infinite dimensional parameter identification. Our second example is that of parameter identification in a finite dimensional framework where we consider the reaction-diffusion system posed on a stationary two-dimensional domain. For both examples, we use synthetic patterns – computer generated – based on previous works using a fixed set of parameters Garvie et al (2010); Madzvamuse et al (2010).

3.2 Turing systems

We consider infinite (as well as finite) dimensional parameter identification for a well-known mathematical model, a reaction-diffusion system on evolving and stationary domains. As mentioned earlier, we take the Schnakenberg kinetics Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979) for illustrative purposes.

3.2.1 Reaction-diffusion system posed on uniform isotropic evolving domains

Let $\Omega_t \subset \mathbb{R}^m$ $(m = 1, 2)$ be a simply connected bounded evolving domain for all time $t \in I = [0, t_F]$, $t_F > 0$ and $\partial\Omega_t$ be the evolving boundary enclosing Ω_t. Also let $\mathbf{u} = (u(\mathbf{x}(t), t), v(\mathbf{x}(t), t))^T$ be a vector of two chemical concentrations at position $\mathbf{x}(t) \in \Omega_t \subset \mathbb{R}^m$. The growth of the domain Ω_t generates a flow of velocity $\mathbf{v}$. For simplicity, let us assume a uniform isotropic growth of the domain defined by $\mathbf{x}(t) = \rho(t)\mathbf{x}(0)$ where, $\mathbf{x}(0) \in \Omega_0$ is the initial domain and $\rho(t) \in C^1(0, 1)$ is the growth function (typically exponential, linear or logistic). We further assume that the flow velocity is identical to the domain velocity $\mathbf{v} := \frac{d\mathbf{x}}{dt}$. The evolution equations for reaction-diffusion systems can be obtained from the application

of the law of mass conservation in an elemental volume using Reynolds transport theorem. Since the domain evolution is known explicitly, a Lagrangian mapping from an evolving to a stationary reference initial domain yields the following non-dimensional reaction-diffusion system with time-dependent coefficients Crampin et al (2002); Mackenzie and Madzvamuse (2011); Madzvamuse and Maini (2007); Madzvamuse et al (2010, 2016)

$$\begin{cases} \begin{cases} u_t + \frac{m\dot{\rho}(t)}{\rho(t)}u = \frac{1}{\rho^2(t)}\Delta u + \gamma f(u,v), \\ v_t + \frac{m\dot{\rho}(t)}{\rho(t)}v = \frac{d}{\rho^2(t)}\Delta v + \gamma g(u,v), \end{cases} & \mathbf{x} \in \Omega_0, \\ \mathbf{n}\cdot\nabla u = \mathbf{n}\cdot\nabla v = 0, \ \mathbf{x}\,\text{on}\,\partial\Omega_0, \\ u(\mathbf{x},0) = u_0(\mathbf{x}), \ \text{and} \ v(\mathbf{x},0) = v_0(\mathbf{x}), \ \mathbf{x}\,\text{on}\,\Omega_0, \end{cases} \tag{3.1}$$

where Δ is the Laplace operator on domains and volumes, d is the ratio of the diffusion coefficients and $\dot{\rho} := \frac{d\rho}{dt}$. Here, $\mathbf{n}$ is the unit outward normal to Ω_t. Initial conditions are prescribed through non-negative bounded functions $u_0(\mathbf{x})$ and $v_0(\mathbf{x})$. In the above, $f(u,v)$ and $g(u,v)$ represent nonlinear reactions and these are given by the activator-depleted kinetics Crampin et al (1999); Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979)

$$f(u,v) = a - u + u^2 v, \quad \text{and} \quad g(u,v) = b - u^2 v. \tag{3.2}$$

To proceed, let us fix the parameters $a, b, d, \gamma \geq 0$, and use the Bayesian approach to identify the domain growth rate function $\rho(t)$, assuming $\rho(0) = 1$, $\rho(t) > 0$ for $t > 0$ and that the domain size at the final time, $\rho(T)$, is known.

Well posedness results for the system of equations (3.1), as well as stability results, can be found in Venkataraman et al (2012). Furthermore, the positivity of solutions is established. These results ensure that the conditions in Theorem 2 are satisfied, and thus we can conclude the well posedness of the parameter identification problem.

Proposition 1 (Well-posedness of the parameter identification problem)**.** *Let G be the observation operator associated with the reaction-diffusion system* (3.1) *with reaction-kinetics* (3.2)*. Assume that a, b, d, $\gamma > 0$, and $\rho \in C^1(0,T)$, $\rho > 0$. Then, the conditions in Theorem* 2 *are satisfied, and thus the parameter identification problem is well-posed.*

Proof. The result follows from the analysis in Venkataraman et al (2012), where well-posedness for the problem (3.1) is shown. In particular, the first condition in Theorem 2 is a consequence of the a priori estimates for the solution. More precisely, the estimates in Venkataraman et al (2012) show in particular that the L^2-norm of the solution to (3.1) is controlled by the square of the L^2-norm of the domain growth, a bound tighter than the one required in Theorem 2. As noted, this bound is only required for parameters in an infinite-dimensional space. Lipschitz continuity of the solutions with respect to the parameter follows from the *a priori* estimates in Venkataraman et al (2012) and the smoothness of the reaction terms with respect to the solution. □

Although this example is synthetic, it is biologically plausible. First, we generate the data by adding noise to a numerical solution of the system, but the parameter identification problem would be analogous in the case of experimental data (see for instance Vigil et al (1992) for an experiment that produces Turing patterns similar to the patterns that we use as data in Example 2).

Since the noise follows a Gaussian distribution, we evaluate the potential Φ according to Theorem 2. Let $\tilde{G}$ be the observation operator associated to the numerical solver, that maps a parameter p to the numerical approximation of the solution of (3.1). Let y be the data, and let Σ be the covariance of the noise. Then,

$$\Phi(p; y) = \|\tilde{G}(p) - y\|_{\Sigma}^2,$$

where $\|\cdot\|_{\Sigma}$ is the Euclidean norm weighted with the noise covariance Σ.

Example 1: Infinite dimensional parameter identification

Without any loss of generality, we restrict our first example to the one-dimensional case ($m = 1$) where we fix model parameters with standard values in the literature as Murray (2013)

$$a = 0.1,\ b = 0.9,\ d = 10,\ \gamma = 1000.$$

We want to identify the growth function $\rho(t)$ given synthetic data generated from two different growth profiles defined by the exponential and logistic functions

$$\rho_{exp}(t) = \exp(0.001t), \quad \text{and} \quad \rho_{log}(t) = \frac{\exp(0.01t)}{1 + \frac{\exp(0.01t)-1}{\exp(0.006)}}. \tag{3.3}$$

We fix the final time $T = 600$. Note that the size of the domain at time T is the same in both cases.

Initial conditions are taken as small fixed perturbations around the homogeneous steady state, $(u_*, v_*) = \left(a + b, \frac{b}{(a+b)^2}\right)$, and these are given by

$$u_0(x) = 1 + 0.005 \sum_{k=1}^{9} \sin(k\pi x), \quad \text{and} \quad v_0(x) = 0.9 + 0.005 \sum_{k=1}^{9} \sin(k\pi x). \tag{3.4}$$

The choice of the initial conditions is motivated by comparison with known results about Turing patterns, and also to ensure a unique steady state. The same initial condition is used in Garvie et al (2010) for parameter identification using an optimization approach, and therefore by using the same initial condition we can use their results as a benchmark for validation. Furthermore, the steady state pattern depends on the initial condition—more precisely, the Turing instability only ensures the certain wave numbers will grow, thus the spectrum of the initial condition affects the spectrum of the steady state pattern—. The problem of finding the initial condition given final pattern could also be solved as an inverse problem, but it is beyond the scope of this thesis.

We solve the system using the finite difference method both in space and time. The system of PDEs is solved on the mapped initial unit square domain using a finite difference scheme. Similar solutions can be obtained by employing finite element methods for example or any other appropriate numerical method. The algebraic linear systems are solved using a conjugate gradient method from the module *SciPy* Jones et al (2001–).

The time-stepping scheme is based on a modified implicit-explicit (IMEX) time-stepping scheme where we treat the diffusion part and any linear terms implicitly, and use a single Picard iterate to linearise nonlinear terms Ruuth (1995); Madzvamuse (2006); Venkataraman et al (2012). This method was analysed in Lakkis et al (2013) for finite element discretisation. It is a first order, semi-implicit backward Euler scheme, given by

$$\begin{aligned}\frac{u^{n+1}-u^n}{\tau}+\frac{\dot{\rho}^n}{\rho^n}u^{n+1} &= \frac{1}{\rho^{2n}}\Delta_h u^{n+1}+\gamma(a-u^{n+1}+u^n u^{n+1}v^n),\\ \frac{v^{n+1}-v^n}{\tau}+\frac{\dot{\rho}^n}{\rho^n}v^{n+1} &= \frac{d}{\rho^{2n}}\Delta_h u^{n+1}+\gamma(b-(u^n)^2 v^{n+1}),\end{aligned} \tag{3.5}$$

where Δ_h is the standard 3-point (or 5-point) stencil finite difference approximation of the Laplacian operator in 1-D (or 2-D) respectively, with Neumann boundary conditions. The parameters of the solver are $h = 10^{-2}$ and $\tau = 10^{-4}$ for all the computations.

We remark that our aim here is to illustrate the applicability of the Bayesian approach and the Monte Carlo methods for parameter identification to problems emanating from mathematical biology. More sophisticated solvers can be used and will improve the computational time Venkataraman et al (2013). The synthetic data is generated by solving the reaction-diffusion system (3.1) with reaction-kinetics (3.2) up to the final time $T = 600$, and then construct the synthetic data by perturbing the solution with Gaussian noise with mean zero and standard deviation equal to 5% of the range of the solution. The data is illustrated in Figure 3.1.

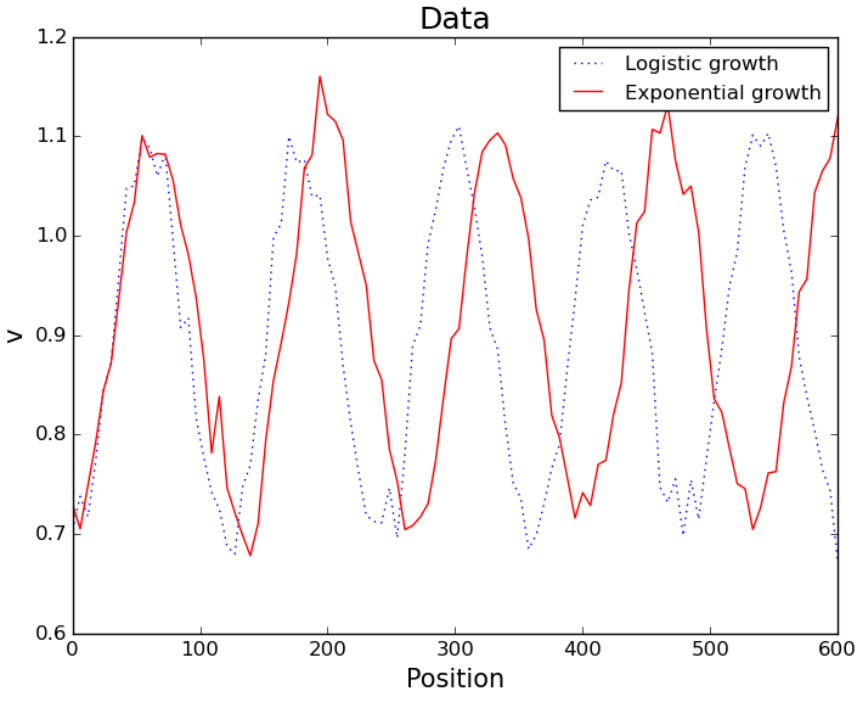


FIGURE 3.1: Synthetic data for the reaction-diffusion system (3.1) with reaction-kinetics (3.2) on a one-dimensional growing domain with exponential and logistic growth functions. The figure depicts only the v-component of the solution, the u component is 180 degrees out of phase. (Colour version online).

In order to identify a time-dependent parameter $\rho(t)$, we approximate it on a finite dimensional space by using a polynomial of degree four, with only three degrees of freedom. The coefficients of order zero and four are fixed in order to satisfy the conditions for $\rho(t)$ at times $t = 0$ and $t = T$, respectively. The priors for the coefficients of the polynomial approximation of $\rho(t)$ are Gaussian distributions, adjusted to ensure that 95% of the samples lay in the large shaded region in Figure 3.2 (yellow in colour version). Note that the results of Proposition 1, and in particular the Lipschitz continuity of the solutions with respect to ρ ensure that the dicretised measure converges upon refining of the discretisation. Therefore, we are effectively approximating an infinite dimensional parameter.

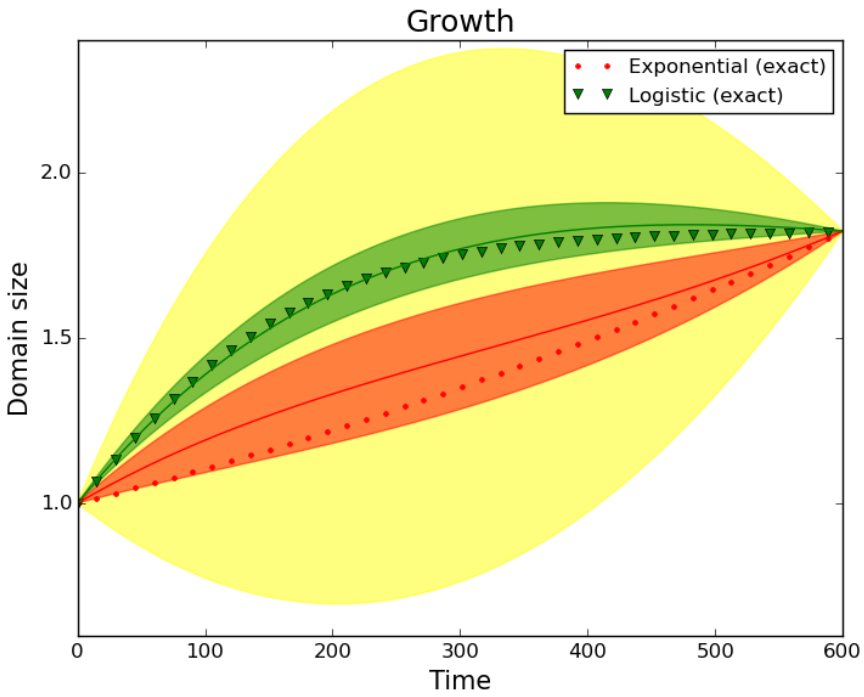


FIGURE 3.2: Regions where 95% of the samples of the prior (light colour, larger region) and the posteriors for exponential growth (darker colour, bottom region) and logistic growth (darker colour, upper region). The exact growth used to generate the data is traced with triangles (logistic) and circles (exponential). The solid lines are the growth rates computed using the mean of each posterior distribution (Colour version online).

The same prior is used for both the logistic and the exponential growth data sets. In Figure 3.2 we depict the regions where 95% of the samples from the posteriors lie, and also the region where 95% of the samples of the prior lie. Observe that we could also plot the credible regions for the coefficients of the finite dimensional approximation of ρ, but it is more difficult to

Parameter	Value
a	0.126779
b	0.792366
d	10
γ	1000

TABLE 3.1: Exact values of the parameters, used to generate the noisy data shown in Figure 3.3.

visualise the result from it. We compute these credible regions as highest posterior density regions, i.e. the smallest region of the parameter space has probability 0.95 with respect to the posterior distribution.

For the final time $T = 600$, we find that the region corresponding to the parameters identified from the exponential growth rate and the region corresponding to the logistic growth rate are completely separated: we can distinguish the type of growth from the solutions at the final time T.

Example 2: Finite dimensional parameter identification

Next we demonstrate the applicability of our approach to identifying credible regions for parameters which are constant and not space nor time-dependent. This is an often encountered problem in parameter identification. We will again use the reaction-diffusion system (3.1) with reaction-kinetics (3.2) in the absence of domain growth, i.e. $\rho(t) = 1$ for all time. Our model system is therefore posed on a stationary domain, for the purpose of demonstration, we assume a unit-square domain. We seek to identify a, b, d and γ. For ease of exposition, we will seek parameters in a pair-wise fashion.

Our "experimental data" are measurements of the steady state of the system. We fix the initial conditions as a given perturbation of the spatially homogeneous steady state given by $(a+b, \frac{b}{(a+b)^2})$. For the values of the parameters given in Table 3.1, the initial conditions are given by

$$\begin{aligned} u_0(x,y) &= 0.919145 + 0.0016\cos(2\pi(x+y)) + 0.01\sum_{j=1}^{8}\cos(2\pi jx), \\ v_0(x,y) &= 0.937903 + 0.0016\cos(2\pi(x+y)) + 0.01\sum_{j=1}^{8}\cos(2\pi jx). \end{aligned} \tag{3.6}$$

We let the system evolve until the L^2-norm of the discrete time-derivative is smaller than a certain threshold. For the numerical experiments presented here, the threshold is 10^{-5}. At that point, we assume that a spatially inhomogeneous steady state has been reached. We record the final time T and save the solution. To confirm that the solution is indeed stationary, we keep solving the system until time $t = 2T$ and check that the difference between the solutions at time T and $2T$ is below the threshold. To generate our synthetic measurement, we add Gaussian noise to the solution, as illustrated in Figure 3.3.

This synthetic experiment is a situation similar to what one will face in an actual experiment, although some assumptions, in particular the fixed known initial conditions, are not realistic. A detailed study of the dependence of the solution with respect to initial conditions will be necessary to drop this assumption. Alternatively, the initial conditions could be included as a parameter to identify.

Case 1: Credible regions for a and b with little knowledge

For our first example, we assume that the values of the parameters γ and d are known, and that we would like to find the values for the parameters a and b. In a first approach, we assume very little knowledge about a and b: only their order of magnitude is assumed to be known. This knowledge is modelled by a uniform prior distribution on the region given by $[0.1, 10]^2$. The data for this example has Gaussian noise with standard deviation 5% of the range of the solution. We can see in Figure 3.4 a 95% probability region for the posterior distribution. Observe how this region is concentrated around the exact value, in contrast to our original knowledge of a uniform distribution on the region $[0.1, 10]^2$. More precisely, the credible region is contained within the range $[0.126, 0.128] \times [0.789, 0.796]$. The length of the credible region in the b-direction is approximately 3.5 times larger than in the a-direction, although the size relative to the magnitude of the parameters is smaller in the b-direction. Intuitively, the larger credible region in the b-direction might be connected with the contrast in the diffusion coefficients for u and v. In order to have Turing patterns—i.e. Turing instability—in fixed domains, the ratio between v-diffusion and u-diffusion must be larger than 1, and after rescaling, we assumed without loss of generality that the diffusion coefficient for u is 1.

For the Schnakenberg reaction kinetics Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979), it is possible to compute the region that contains the parameters

that can lead to non-homogeneous steady states, the Turing space (see next example for details). We can see that our original prior covered an area much larger than the Turing space (almost 100 times larger), but the posterior is concentrated in a small region completely contained within it (see Figure 3.5).

Case 2: Credible regions for a and b using the Turing parameter space

Unlike the previous example, where we assume little knowledge of the prior, here we exploit the well-known theory for reaction-diffusion systems on stationary domains and use a much-more informed prior based on analytical theory of the reaction-diffusion system. On stationary domains, diffusion-driven instability theory requires reaction-diffusion systems to be of the form of *long-range inhibition, short-range activation* for patterning to occur (i.e. $d > 1$). More precisely, a necessary condition for Turing pattern formation is that the parameters belong to a well-defined parameter space Murray (2013) described by the inequalities (for the case of reaction-kinetics (3.2))

$$\begin{aligned} f_u + g_v = \frac{b-a}{b+a} - (a+b)^2 &< 0, \\ f_u g_v - f_v g_u = (a+b)^2 &> 0, \\ d f_u + g_v = d\left(\frac{b-a}{b+a}\right) - (a+b)^2 &> 0, \\ (d f_u + g_v)^2 - 4d(f_u g_v - f_v g_u) & \\ = \left(d\left(\frac{b-a}{b+a}\right) - (a+b)^2\right)^2 - 4d(a+b)^2 &> 0, \end{aligned} \tag{3.7}$$

where f_u, f_v, g_u and g_v denote the partial derivatives of f and g with respect to u and v, evaluated at the spatially homogeneous steady state $(a+b, \frac{b}{(a+b)^2})$. In Figure 3.5 we plot the parameter space obtained with Schnakenberg kinetics.

In this second example, we use data with added Gaussian noise with standard deviation 10% of the solution range. For the prior, we now use a uniform distribution on the Turing space of the system. In Figure 3.6 we can now see that despite the increased noise, the improved prior reduced the size of the 95% probability region dramatically.

Case 3: Credible regions for d and γ

In a third example, we assume that a and b are known, and we would like to find γ and d. To illustrate the use of different types of priors, here we assume a log-normal prior that ensures positivity of γ and d, which in the case of d is necessary to ensure a well-posed problem. We use the data with 5% noise, and the prior distribution of a log-normal with mean (5, 500) and standard deviation 0.95. A 95% probability region of the posterior is depicted in Figure 3.7. Note that the use of a log-normal or similar prior is essential here to ensure positivity of the diffusion coefficient which is required for the well posedness of the forward problem. The prior distribution covers a range of one order of magnitude for each parameter, and the posterior distribution suggests relative errors of order 10^{-3}.

Case 4: Credible regions for a, b, d and γ

Finally, we identify all four parameters a, b, d and γ from the set of data with 10% noise respectively. We use the priors from **Case 1** for a and b, and from **Case 3** for d and γ. We do not apply any further restrictions on the parameters.

In Figure 3.8 we depict the credible regions for the projection of the parameters to four different coordinate planes. The noise for this experiment is higher than in **Case 1** and **3**. Also note that compared to the previous experiments we assume here less knowledge *a priori* about the parameters. It must be noted that assuming that a parameter is known is equivalent in the Bayesian formulation to use a Dirac delta for the prior on the parameter; in comparison, the prior distributions for **Case 4** are more spread, therefore they represent less knowledge. Since the level of noise is higher, and the prior knowledge lower, the credible regions that we obtain are larger. In this case, the relative errors are of order 10^{-2} to 10^{-1}.

3.2.2 HPC computations

For each of the examples shown in Section 3.2, we generate 10 Markov chains for a total of approximately 10^6 samples. The mean and the correlation of each chain are examined, and used to decide the burn-in—the fraction of the chain discarded due to the influence of the initial value. The burn-in fraction is determined by checking the convergence of the mean for each chain. Finally, the chains are combined in a big set, that we use to generate the plots.

All the results shown are generated using the local HPC cluster provided and managed by the University of Sussex. This HPC cluster consists of 3000 computational units. The specifications of the computational units are AMD64, x86_64 or 64 bit architecture, made up of a mixture of Intel and AMD nodes varying from 8 cores up to 64 cores per node. Each unit is associated with 2GB memory space. Most of the simulations in this paper are executed using 8-48 units. The wall-clock computation time for one chain ranged from 1 to 4 days, CPU time ranged from 10 to 35 days for one chain. It must be noted that there are two levels of parallelisation: first the algorithm is parallelised and uses 8 of the available cores, and then many instances of the algorithm run at the same time to produce independent chains. We remark that we produce independent chains in order to test the convergence of the algorithm.

3.3 Conclusion

The Bayesian framework offers a mathematically rigorous approach that allows for the inclusion of prior knowledge about the parameters (or more generally functions). Furthermore, the well-posedness results for the identification problem are often possible to obtain under minimal assumptions other than those needed to ensure the well-posedness of the forward problem.

Although exploring a whole probability distribution can be computationally expensive, very useful information about the uncertainty or correlation of the parameters can be inferred from it. The use of a parallel Metropolis-Hastings algorithm makes the computations feasible using HPC facilities.

We studied the parameter identification problem for reaction-diffusion systems using Turing patterns as data for both scenarios: infinite and finite dimensional parameter identification cases. We presented the results that ensure the well-posedness of the parameter identification problem, and we performed several numerical simulations to find credible regions for the parameters. In particular, we provided numerical evidence that we can distinguish between different growth functions based solely on noisy observations of the data at the end time.

3.3.1 Further work

This approach opens many possibilities in the study of inverse problems connected with Turing patterns. As mentioned in Section 3.2.1, the initial conditions are not known in a real

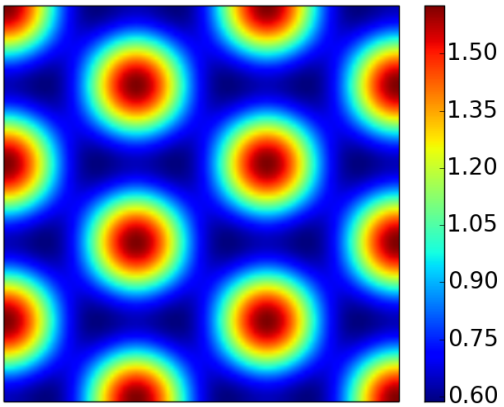
application with experimental data, and can be treated as a parameter as well.

We can also study qualitative properties of the patterns, by means of alternative definitions of the noise and the likelihood. For example, we could take data on the Fourier transform of the pattern in order to identify parameters for different modes—i.e. spots, stripes, etc.— in the patterns.

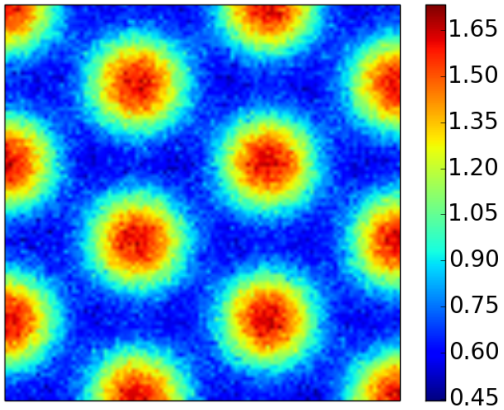
We used synthetic data in this work, but we presented the method keeping in mind the applicability for problems with experimental data, by trying to use for the parameter identification only data that could be measured in an experiment, for instance, the final pattern but not all the evolution of the system. Furthermore, with an adequate PDE solver, we can also apply this approach to problems posed on manifolds. A manuscript presenting the application of these methods with experimental data is in preparation.

A comparison between different methods to estimate posterior distributions in the context of problems involving PDEs and infinite-dimensional parameter would provide more insights on the performance of the different algorithms. For example, we could compare the method presented here with the Approximate Bayesian Computation.

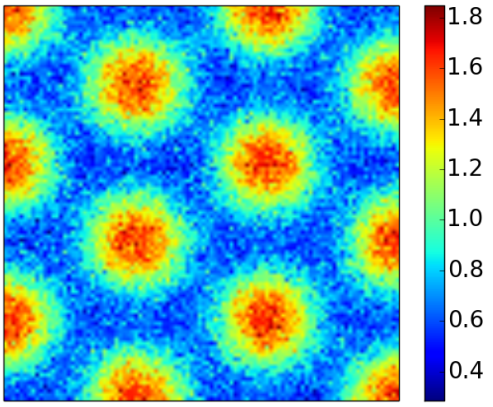
The comparison of the approach presented here with the optimal control approach would allow a better understanding of the trade-off between the information about the parameters and the computational cost of the method.



(a)



(b)



(c)

FIGURE 3.3: The u-component of the solution to the Schnakenberg system Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979), with added Gaussian noise with mean zero and standard deviation 5% (b) and 10% (c) of the range of the solution. Solutions of the v-component are 180 out-of-phase with those of u and as such their plots are omitted (Colour version online).

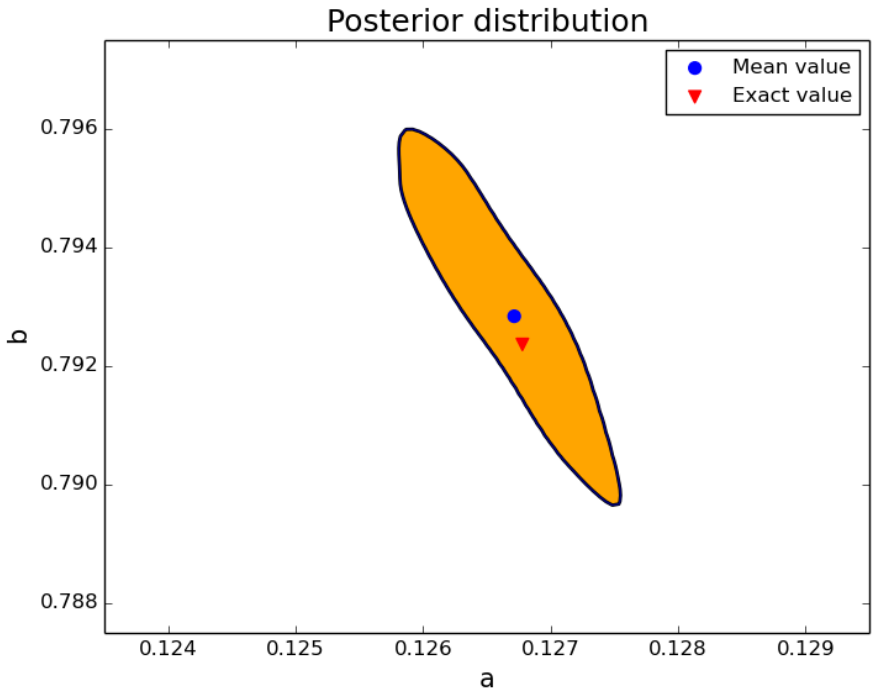


FIGURE 3.4: 95% credible region for the posterior distribution for the parameters a and b, using a uniform prior on the region $[0.1, 10]^2$. Note that for ease of visualisation, scales for a and b are different (Colour version online).

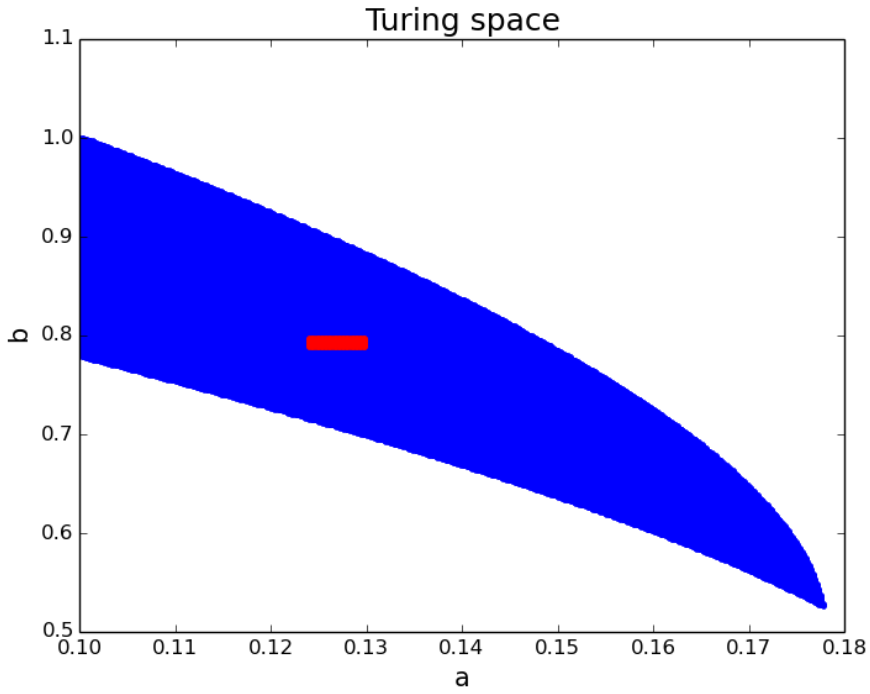


FIGURE 3.5: Darker region (blue in the online version), the Turing space for the parameters a and b of the Schnakenberg reaction kinetics Gierer and Meinhardt (1972); Prigogine and Lefever (1968); Schnakenberg (1979). Lighter region (red in the online version), the region plotted in Figure 3.4, depicting where the credible region is contained (Colour version online).

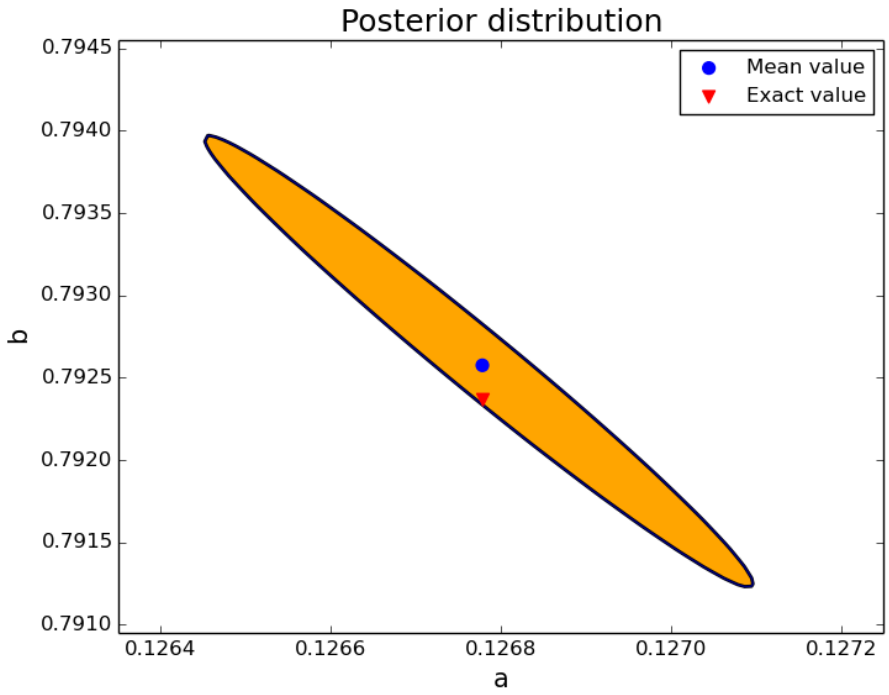


FIGURE 3.6: 95% credible region for the posterior distribution for the parameters a and b, using a uniform prior on the Turing space (Colour version online).

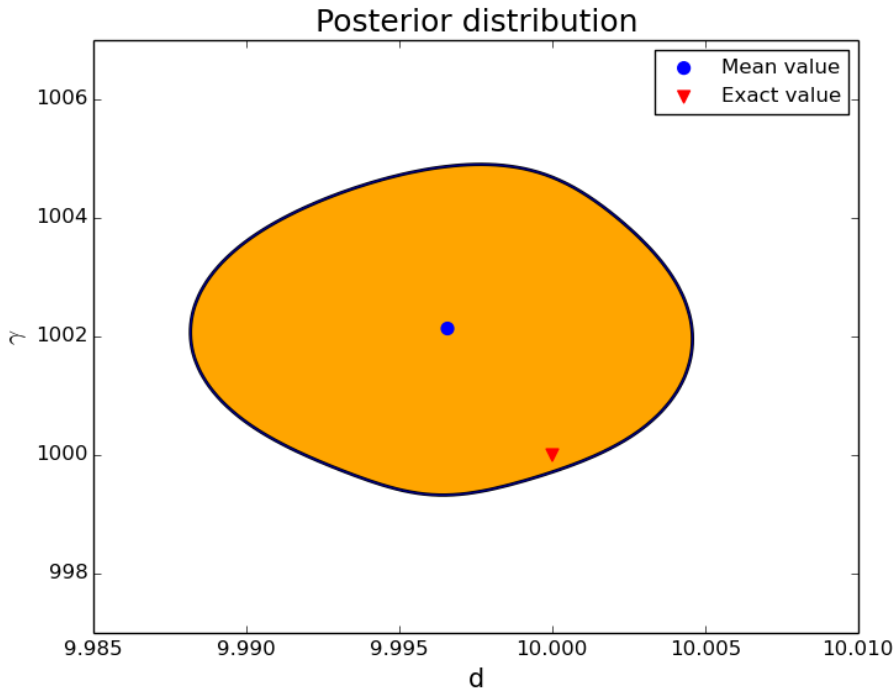


FIGURE 3.7: 95% credible region for the posterior distribution for the parameters d and γ, a log-normal prior with mean (5,500) and standard deviation 0.95 (Colour version online).

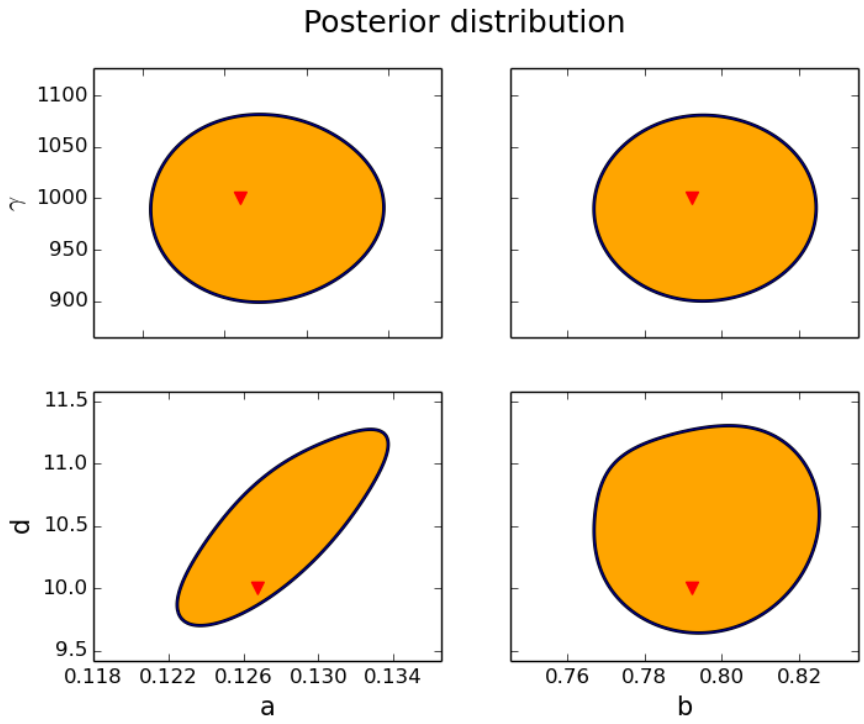


FIGURE 3.8: 95% credible region for the posterior distribution for the parameters a, b, d and γ. See **Case 1** and **3** for a description of the priors. The data has a noise of 10% of the range of the solution. Each subplot corresponds to the projection of the credible region onto a coordinate plane for two of the parameters, given by the row and the column of the subplot. The exact value of the parameters is marked with a triangle.(Colour version online).

Chapter 4

Parameter identification for the spatiotemporal organisation of keratin material

4.1 Introduction

Keratins are a family of fibrous structural proteins. The basic keratin monomer assembles in the form of intermediate filaments, one of the main components of the cytoskeleton. As part of the cytoskeleton, keratins can exhibit a very dynamic behaviour, with assembly and disassembly occurring in a continuous cycle; yet in other cases such as cornified tissues, keratin forms a passive, static structure. Antibodies for keratin can be used as a fluorescent marker, thus allowing the observation of the keratin structures in a microscope.

Keratins are a ubiquitous material. They are present in epithelial cells and in particular in keratinocytes, the predominant cell type in the epidermis. Keratin is also the main material in hair, nails, claws, horns and the whale baleen. In birds and reptiles, keratins are present in beaks, feather and scales (Wang et al (2016a)).

The mechanical properties of keratins are crucial to their main functions. As part of the cytoskeleton, keratins provide epithelial cells with resistance to stress and physical damage. In cornified tissue, keratins fill the cells almost completely, changing the mechanical properties of tissue (Tombolato et al (2010)). For example, calluses are formed by cornification of the outermost layer of the epidermis in response to mechanical stresses such as rubbing or pressure (Greenberg et al (1991)).

Malfunctions of the keratin can lead to several diseases. Mutations in the keratins genes cause Epidermolysis bullosa and other skin blistering diseases. Keratin is present in some types of cancer, and the analysis of keratin expression is useful in determining the origin of some metastases (Itakura et al (2001); Omary et al (2009)).

In this work, we are interested in the dynamics of the keratin network in cells. Our aim is to derive a mathematical model that includes several possible mechanisms of assembly, disassembly and transport of keratin, and to use experimental data to find the model parameters. The properties of the parameters, extracted from the posterior probability distribution, contain information about the plausible underlying mechanisms. In contrast to the model derivation in Portet et al (2015), we give a detailed description of the biological assumptions and derive the model from first principles. Furthermore, our derivation ensures conservation of the total amount of keratin, which is not guaranteed in the form of the model presented in Portet et al (2015).

The keratin network assembles near the membrane of the cell, and then it moves towards the nucleus. Near the nucleus, the network disassembles and keratin is transported back to the membrane. The exact mechanisms of transport, assembly and disassembly are not known.

There are several mathematical models for the keratin dynamics in the literature. Sun et al (2013) present a model for the average concentration of keratin in the cell. The model considers three different states of keratin: assembled, soluble, and precursor. Precursor keratin is an intermediate state between assembled and soluble, and it consists of short chains of keratin created near the boundary. A previous model (Portet and Arino (2009)) uses yet another intermediate state, in this case between the precursor and the assembled keratin, to model short filaments. Similarly, Sun et al (2017) use a model based on delayed ordinary differential equations to study the disassembly of keratin, by modelling three different states.

Portet et al (2015) presented a model that includes the spatial dynamics of the keratin. The model is a reaction-diffusion-advection system for two state variables: assembled and soluble keratin. The same paper includes a study of the parameter fitting to a set of experimental data, and they use an information theory approach to find the best model among a hierarchy of combinations of the biologically plausible mechanisms.

4.2 Experimental data for the structure of the Keratin material

Fluorescence microscopy permits the direct observation of keratin structure in living cells. Generally speaking, the technique consists of using a cell strain that expresses a modified version of the protein. The modified protein includes a fluorescent structure. During the experiment, the cell is excited with light at the right wavelength, and the fluorescent protein becomes visible under the microscope.

In particular, in order to observe keratin, a fluorescent structure is attached to the wild type keratins (Moch et al (2013)). Soluble keratin—the disassembled form—is very dilute in the cytosol, and in consequence it does not emit enough light in order to be observed. In contrast, the assembled keratin is clearly observable.

We shall use experimental data recorded in the experiments described in Moch et al (2013). For these experiments, the cells were keratinocytes expressing the fluorescent keratins. The experiment proceeded as follows: 24 hours after seeding the cells, Moch *et al.* recorded a first set of images of the keratin network. 24 hours later—48 hours since the beginning of the experiment—, they captured a second set of images (see Fig. 4.2). We will refer to these two data sets as *early* and *late*.

The data was pre-processed by the authors of the experimental study, following the methods described in Herberich et al (2010) and Ma and Plonka (2010). The main step in the preprocess of the images is the application of a curvelet-based denoising algorithm to reduce the image noise without blurring the structure of the keratin filaments.

Different cells have different shapes during the experiment. In order to facilitate the comparison of the data from different cells, Moch *et al.* normalised the data from the different geometries to a circle. The normalised cells are circles, of radius $22.5\mu m$; this is the average radius of the keratinocytes (Möhl et al (2012); Moch et al (2013)).

We receive the data in the form of two collections of images, 50 images of early cells—24 hours after seeding—, and 84 images of late cells—48 hours after seeding—. The center of the normalised cells coincides with the center of the image. Each image measures 1024 x 1024 pixels, and the cell radius, in pixels, is 313 pixels. The format of the images is TIFF, without data loss due to compression.

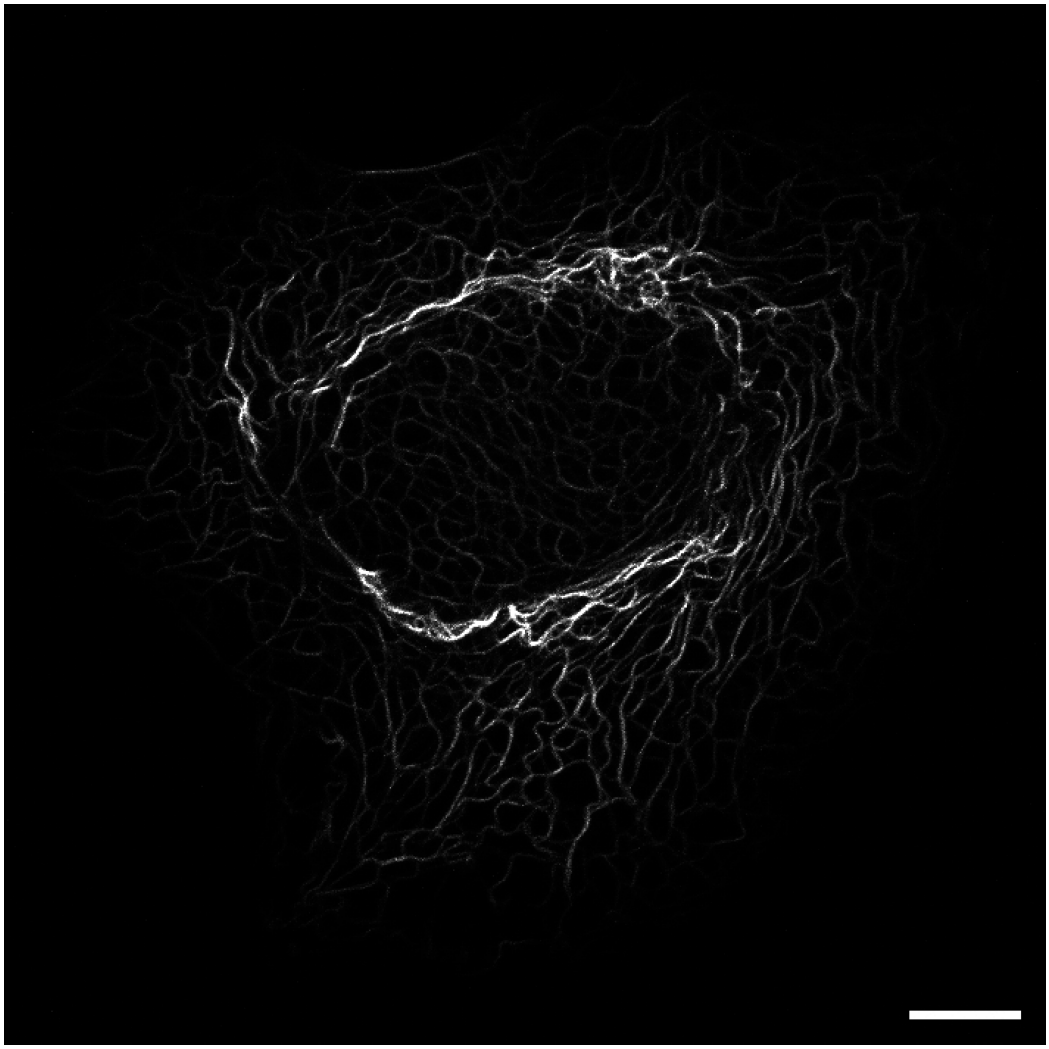


FIGURE 4.1: An image of a hepatocellular carcinoma cell expressing fluorescent keratin. The keratin network is sparser near the cell boundary, where it is assembled; it then moves towards and nucleus and becomes denser in the nuclear periphery. The image is part of a time-lapse. See the supporting information in Portet et al (2015) for a video. Image from Portet et al (2015).

4.3 A novel approach to data processing in quantitative biology

For the data, the cell shape is a circle. There is no significant polarisation in the cells, and therefore when we compare the data of two different cells, any possible direction is the same in terms of the data. As a consequence, any attempt of modelling this data requires a circular symmetry, and can be simplified to a one dimensional model, restricted to a radius of the cell. We choose to derive the model directly in one dimension, because it is simpler and it also

allows direct comparison with previous works on the topic (e.g. Portet et al (2015)).

As a result, our first goal is to reduce the images to two one-dimensional datasets, one for the cells measured at the early time, and one for the cells measured at a late time. We describe a novel data processing approach for the early time dataset. The process is completely analogous for the late time dataset.

We use the standard deviation of the intensity at each point in space to estimate the noise. The Central Limit Theorem (see for instance Dudley (2018)) guarantees that the mean intensity at a fixed distance of the center is distributed as a Gaussian, and that the sample average and standard deviation are good estimators of the mean and variance.

Let us consider that the cell centre is at the origin of coordinates. On the positive x direction of the radius, there are 313 pixels. This is the best resolution that we can obtain from the image. We shall bin the pixels in the dataset in 313 different bins, according to the distance to the centre of the image in pixels, rounded to the nearest integer.

Since the data in a cell of radius measures 313 pixels, we let $n \in \{1, \dots, 313\}$. Let $L_k(i,j)$ be the intensity of the pixel with coordinates (i,j) in the k-th image. The average m and the standard deviation s at a distance n from the center are given by

$$m = \sum_k \sum_{i^2+j^2=n^2} L_k(i,j)/N_n, \tag{4.1}$$

$$s = \sum_k \sum_{i^2+j^2=n^2} (L_k(i,j) - m)^2/(N_n - 1), \tag{4.2}$$

where N_n is the total number of pixels at distance n from the center of the image.

Since we measure the noise independently at each point in space, we are actually over-estimating the standard deviation. A more comprehensive approach would be to treat each cell radius in the data as a realisation of a Gaussian process, measured at the discrete space points. This approach would require a larger dataset in order to get a good estimate of the underlying Gaussian process. With the process that we just described, we obtain a dataset for the average intensity. In order to interpret the data in terms of keratin concentration, we make the following assumptions.

(A1) The luminosity $L(i,j)$ of a pixel (i,j) is proportional to the concentration of assembled keratin at the same point.

(A2) 95% of the keratin content of a given cell is in the assembled form.

(A3) The distribution of disassembled keratin at the initial time is proportional to the distribution of assembled keratin.

Previous studies (Moch et al (2013); Portet et al (2015)) on this topic used Assumption (A1). As long as the intensity does not reach the saturation limit of the camera sensor, this assumption corresponds to the fact that the light emitted by the fluorescent molecules is proportional to the number of molecules. Feng et al (2013) measured the average keratin content c_K of keratinocytes to be

$$c_K = 520\mu M.$$

Let c_L be the mean luminosity in the data set. We compute the proportionality constant ρ between keratin concentration and luminosity as

$$\rho = c_K / c_L, \tag{4.3}$$

and therefore the keratin concentration at a given point with luminosity l is given by ρl.

The second assumption corresponds to the experimental results reported in Chou et al (1993). Since our goal is to derive a dynamic model of the keratin networks, in order to perform simulations we need to prescribe initial conditions. As we will see in the sequel, the dependent variables in the model correspond to the assembled and disassembled keratin concentrations. Thus we need to prescribe initial conditions for both species, but the experimental data only provides information about the assembled keratin concentration. Using Assumptions (A2) and (A3), we can compute the distribution of disassembled keratin at the initial time as

$$g_0 = \frac{0.05}{0.95} f_0, \tag{4.4}$$

where f_0 is the distribution of assembled keratin, that we estimate from the experimental data.

Therefore, in order to obtain the data corresponding to our mathematical model, we first use (4.1) to obtain the mean luminosity over a radius, and then by means of (4.3) we convert the luminosity to keratin concentration. Since the observations correspond only to the assembled keratin, we finally apply (4.4) to obtain the concentration of the disassembled keratin.

Summing up, we obtain three datasets, corresponding to the average concentration of assembled and disassembled keratin at the early time, and to the concentration of assembled keratin at the late time. The concentrations at an early are the initial conditions for our model; the concentration of assembled keratin at the late time is the data to measure the fitness of the model.

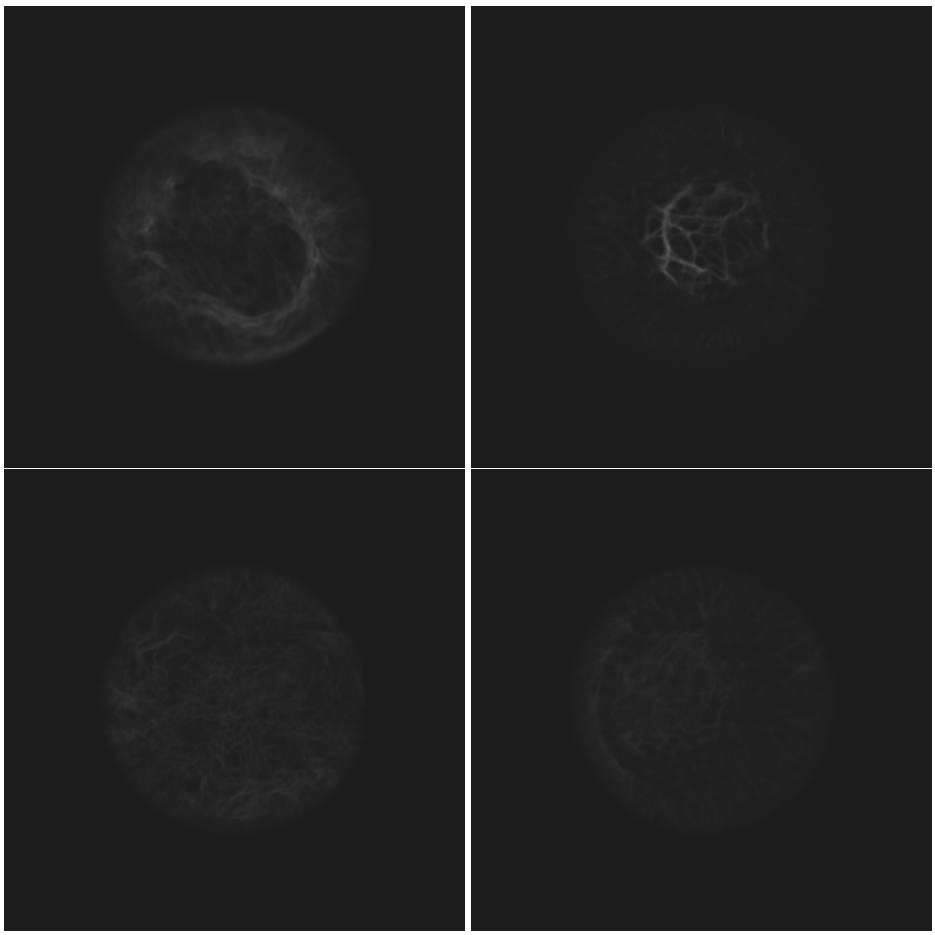


FIGURE 4.2: Images of the keratin network. The cell shape is normalised to a circle, according to the methods from Möhl et al (2012). The experimentalist provided us with two data sets, corresponding to two different times of the experiment: 24 hours after seeding (top row) and 48 hours after seeding (bottom row). The early-time data set contains 50 images, whilst the late-time data set 84 contains images.

4.4 Data-driven mathematical model

Let $L \in \mathbb{R}^+$ be the radius of a normalised cell, and let $\Omega = [-L, L]$ a domain that represents a cell diameter. Let $T_0, T \in \mathbb{R}^+$ be the initial and final times of the experiment, respectively.

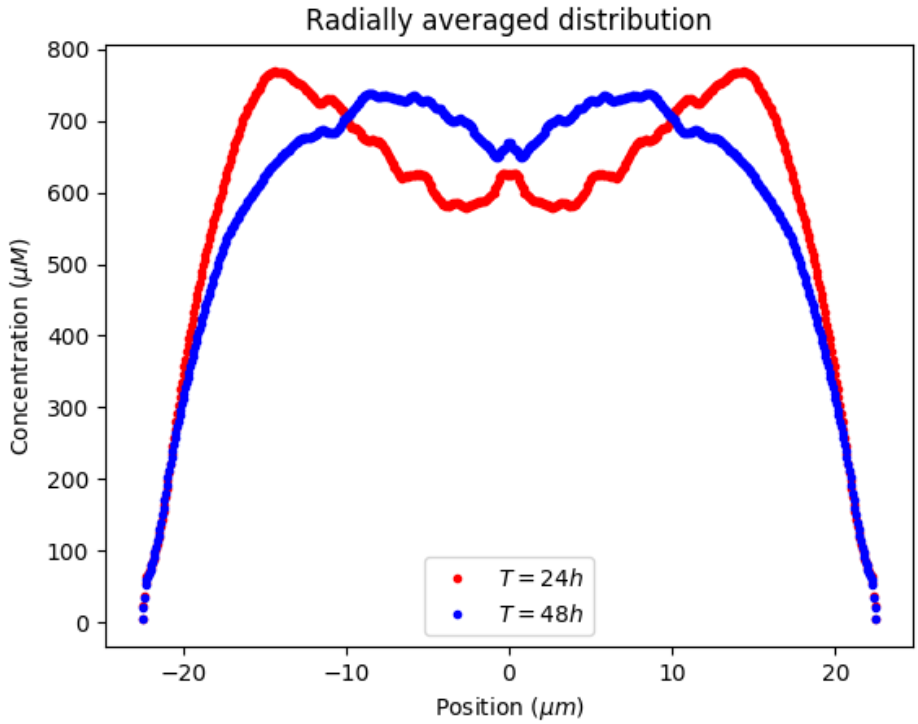


FIGURE 4.3: Tidy data set obtained from the images of cells normalised to a circle. We average the intensity over a radius, for all the images taken at same time; by means of (4.3) and (4.4) we convert the intensity values to keratin concentration. Since the mathematical model provides solutions along a diameter, for purposes of representation, we symmetrise the data with respect to the origin.

For our modelling purposes, we consider two forms of keratin: insoluble and soluble. Soluble keratin represents keratin disolved in the cytosol, whilst insoluble keratin represents the assembled keratin network. Let $u_I(x,t)$ and $u_S(x,t)$ be the concentrations of insoluble and soluble keratin, respectively, at $x \in \Omega$, $t \in [0,T]$. Similarly, let $\mathbf{J}_I(x,t)$ and $\mathbf{J}_S(x,t)$ be the fluxes insoluble and soluble keratin.

In order to derive the mathematical model for the dynamics of the keratin network, we make the following assumptions.

(A1) Keratin is locally conserved.

(A2) There is no flux of keratin through the boundaries.

(A3) Keratin is only present in assembled and disassembled forms.

The assumption on local conservation (A1) implies that we are not modelling the synthesis and degradation of keratin. Synthesis and degradation of keratin occurs at a small rate compared with the transport dynamics that we are modelling (Jaitovich et al (2008)).

With (A1) we are also not taking into account the keratin transport in the direction perpendicular to the view plane. The cell is a three-dimensional entity, and keratin may move up and down; in the data, keratin transported in the direction perpendicular to the image will disappear, and vice versa, keratin coming into the view plane will seem to appear. The reason is that as soon as keratin is displaced in the perpendicular direction, it moves out of focus and cannot be observed in the current configuration of the microscope.

The local conservation of keratin (A1) allows as to write a conservation law for the concentration of keratin (see for instance Eck et al (2017)). For $j \in \{I, S\}$, we have

$$\partial_t u_j(x,t) + \operatorname{div}(\mathbf{J_j}(\mathbf{x},\mathbf{t})) = R_j(x,t), \tag{4.5}$$

where R_j is a source term accounting for the creation or destruction of keratin.

Keratin does not flow through the cellular membrane, thus (A2). Let ν be the normal direction at the boundary of the domain Ω. For $j \in \{I, S\}$, (A2) is equivalent to

$$\mathbf{J}_j \cdot \nu = 0 \text{ at } \partial\Omega. \tag{4.6}$$

Assumption (A3) is a modelling assumption. We could also include intermediate states between soluble (disassembled) keratin and the insoluble (assembled) network, see Sun et al (2013, 2017). Since we have experimental data only for the concentration of assembled keratin, we choose a minimum model.

If we sum (4.5) for soluble and insoluble keratin, integrate over the entire domain, and use (4.6), we deduce that $R_I + R_S = 0$. We can therefore define $R = R_I$, and we have $R_S = -R$.

Therefore, the concentrations $u_I(x,t)$ and $u_J(x,t)$ satisfy the following system of partial differential equations:

$$\begin{cases} \begin{cases} \partial_t u_I(x,t) + \operatorname{div}(\mathbf{J}_I(x,t)) = R(x,u_I,u_S), \\ \partial_t u_S(x,t) + \operatorname{div}(\mathbf{J}_S(x,t)) = -R(x,u_I,u_S), \end{cases} & (x,t) \in \Omega \times [T_0,T], \\ \mathbf{J}_I \cdot \nu = \mathbf{J}_S \cdot \nu = 0, \qquad (x,t) \in \partial\Omega \times [T_0,T], \\ u_I(x,T_0) = u_{I_0}(x), u_S(x,T_0) = u_{S_0}(x), \quad (x,t) \in \Omega \times \{0\}, \end{cases} \tag{4.7}$$

where u_{I_0} and u_{S_0} represent the initial concentrations of insoluble and soluble keratin.

The soluble keratin diffuses through the cytosol. We do not account for any other transport mechanism of the soluble pool, therefore we define the flux J_S as

$$\mathbf{J}_S = -D_S \nabla u_S, \tag{4.8}$$

where D_S is the diffusion rate of soluble keratin in the cytosol. Results from independent experiments provide an estimate for the value of D_S, see section 4.5.1 for details.

On the other hand, we assume an active transport of the assembled keratin network. There is experimental evidence that keratin filaments interact with the molecular motors attached to the microtubules and actin filaments, producing the movement of the assembled keratin along the microtubules (Windoffer et al (2011); Windoffer and Leube (1999); Helfand et al (2004); Robert et al (2014); Yoon et al (2001)). We represent this active transport of the assembled keratin by a transport velocity $\mathbf{v}$, according to

$$\mathbf{J}_I = \mathbf{v} u_I. \tag{4.9}$$

We assume a nonlinear reaction term R, in order to account for a limit reaction rate in both the assembly and the disassembly reactions. We note that this reactions are a simplification of the actual biological process. There are no conclusive results on the entire reaction pathway for these reactions, but it is reasonable to assume that this reactions are controlled by other compounds in the cell, available only in a limited amount. Furthermore, our results suggest than a linear reaction rate would not give a better fit (see section 4.7 for details). In consequence, our

reaction rate is

$$R(x, u_I, u_S) = \frac{k_a(x)}{k_S + u_S} u_S - \frac{k_d(x)}{k_I + u_I} u_I. \tag{4.10}$$

The scalar parameters k_S and k_I control the saturation limit of the reaction. They are linked to the actual reaction, and therefore we assume that they are constant across the cell. The reaction rates k_a and k_d, are respectively the assembly and disassembly reaction rates. We assume that they depend on the position x in the cell. The results in Moch et al (2013) suggest already that assembly and disassembly reactions occur at characteristic regions of the cell.

The insoluble keratin flux J_I, as defined above, leads to an hyperbolic equation that may develop fronts. In order to ensure well-posedness of the model, we regularise the equation for the insoluble keratin with the addition of a small diffusion term,

$$\mathbf{J}_I = \mathbf{v} u_I - D_I \nabla u_I. \tag{4.11}$$

We shall set the diffusion rate D_I to be always smaller than the diffusion rate D_S, e.g $D_I = 10^{-4} D_S$. In this way, we limit the actual error in the model introduced by the regularisation. Summing up, the model for the evolution of the insoluble and soluble keratin concentration becomes

$$\begin{cases} \begin{cases} \partial_t u_I(x,t) + \operatorname{div}(\mathbf{J}_I) = R(x, u_I, u_S), \\ \partial_t u_S(x,t) + \operatorname{div}(\mathbf{J}_S) = -R(x, u_I, u_S), \end{cases} & (x,t) \in \Omega \times [0,T], \\ \mathbf{J}_I \cdot \nu = \mathbf{J}_S \cdot \nu = 0, \qquad (x,t) \in \partial\Omega \times [0,T], \\ u_I(x,0) = u_{I_0}(x), u_S(x,0) = u_{S_0}(x), \quad (x,t) \in \Omega \times \{0\}. \end{cases} \tag{4.12}$$

In this form, (4.12) is a reaction-diffusion system, a system of parabolic partial differential equations with smooth, bounded reaction terms. Well-posedness results for similar models can be found in Smoller (1982). Note that since $\mathbf{v}$ is bounded, and the reaction terms are bounded for positive solutions, we can use a regularised system to prove existence of solutions, and with enough regularity, positivity and uniqueness.

4.5 Inverse problem

With the data at hand, and the mathematical model, our goal is to apply the inverse problem framework from Chapter 2 to find the probability distributions for the parameters given the data. The model includes both scalar and space-dependent parameters.

The parameters of the model are the diffusion coefficient D_S, the reaction rates k_a and k_d, the limiting constants k_S and k_I, and the transport velocity $\mathbf{v}$. The following proposition shows the well-posedness of the inverse problem, under general assumptions on the parameters.

Proposition 2 (Well-posedness of the parameter identification problem)**.** *Let G be the observation operator associated with the reaction-diffusion system (4.12). Assume that $k_I, k_S, D_S > 0$, and k_a, k_d and $\mathbf{v}$ are smooth, bounded and positive. Then, the conditions in Theorem 2 are satisfied, and thus the parameter identification problem is well-posed.*

We omit the details of the proof. The general theory of reaction-diffusion systems ensures well-posedness of the problem, including Lipschitz continuity with respect to the parameters. Solutions are bounded and controlled by k_S and k_I, which in turn implies the bound on the growth of the solution with respect to the parameters. See Hundsdorfer and Verwer (2013) and the references therein for the general results on the theory of reaction-diffusion equations with transport terms.

4.5.1 Prior distribution of the parameters

In order to proceed with the parameter identification, we shall define the prior distributions for the parameters, i.e. the probability distributions *a priori*, regardless the data.

Diffusion rates

The diffusion rate for the soluble keratin, D_S, is a scalar parameter. Kölsch et al (2010) measured D_S experimentally, obtaining a value of $0.88\mu m^2 s^{-1}$, with a standard deviation of $0.08\mu m^2 s^{-1}$. Therefore, we use a Gaussian prior for D_S, with mean $0.88\mu m^2 s^{-1}$, and standard deviation $0.08\mu m^2 s^{-1}$.

The diffusion rate for the insoluble keratin does not have a direct physical interpretation. We include a diffusion term in the insoluble keratin equation for technical reasons, namely to regularise the problem and simplify the analysis of the forward problem. In order to limit the

effect of this regularisation, we take D_I small. More precisely, we set

$$D_I = 10^{-4} D_S.$$

We remark that this is the same approach of the study by Portet et al (2015), although the justification is different.

Limit reaction disassembly and assembly rates

The scalar parameters k_I and k_S limit the disassembly and assembly reaction rates. There are no experimental results on their values. Therefore, we use a conservative prior given by a uniform distribution in the interval $[250, 1500](\mu M)$.

For the rest of parameters, we follow the approach presented in Chapter 2 to define the prior distributions of infinite-dimensional parameters. Since all the parameters k_a, k_d and $\mathbf{v}$ must be positive, we will define the priors for the logarithm of the parameter, and then take the exponential of the samples to find the parameter values. Recall that the general form of this priors is (see Section 2.5):

$$\exp\left(m_0(x) + \sum_{i=1}^{N} \xi_i w_i f_i(x)\right).$$

The disassembly rate

The parameter k_d controls the disassembly rate of keratin. It is not directly measurable in these experiments, and therefore we use only mild assumptions for its prior. We assume that $k_d : [-L, L] \to \mathbb{R}^+$ is smooth, and by construction of the model, it must be symmetric with respect to the origin.

The current biological models (Moch et al, 2013) assume that the disassembly rate peaks near the nucleus, and decays towards the boundary. We incorporate this hypothesis by assuming that k_d has only one local maximum in $[0, L]$. We also assume that at the origin the disassembly rate vanishes, because of the presence of the nucleus. Note that according to the biologists, we cannot assume in general that k_d vanishes towards the boundary. For comparison, all these assumptions are included in the reaction rates modelled in Portet et al (2015).

In order to incorporate the assumptions mentioned above, we define the prior for k_d as an expansion over a basis of sine functions, with exponentially decaying weights. The exponential decay accomplishes two purposes: first, it ensures that the first mode (i.e. lowest frequency base function) dominates, and therefore k_d will have only one maximum in $[0, L]$; secondly, it guarantees convergence if the expansion is refined. The numerical values of the mean and the weights are adjusted to cover a reasonable range of reaction rates, and in particular, to include the family of reaction rates from Portet et al (2015). The values of the coefficients of the prior for k_d, according to (4.5.1) are

$$\begin{aligned} \xi_i &\sim \mathcal{U}[-1,1], \\ m_0(x) &= -2, \quad \forall x \in [-L, L], \\ w_i &= 2e^{-i}, \\ f_i(x) &= \sin\left(\frac{(i+1)\pi(|x|-3)}{25.5}\right). \end{aligned} \tag{4.13}$$

We include a shift in the base functions to allow for the possibility of non-vanishing disassembly rates at the boundary of the domain. Figure 4.5 depicts one thousand samples from the prior for k_d. Note that the shift in the base functions produces an artifact at the origin. This artifact is not accurate biologically, but nevertheless the prior includes the biologically correct reaction rates that vanish at the origin. We will see later that this artifact is not present in the posterior distribution, and therefore it is not necessary to add complexity to the prior to remove it.

The assembly rate

The modelling of the assembly reaction rate k_a follows the same steps as the modelling of the prior for the disassembly reaction rate k_d. In analogy to the assumptions for the disassembly rate, we assume that $k_a : [-L, L] \to \mathbb{R}^+$ is a smooth function, symmetric with respect to the origin.

The biological models imply that k_a has only one local maximum, near the boundary, and decays to zero. We assume that it vanishes at the origin, but as before, we cannot assume that it vanishes at the boundary. The prior for k_a is defined by the following values:

$$\begin{aligned} \xi_i &\sim \mathcal{U}[-1,1], \\ m_0(x) &= -2, \quad \forall x \in [-L,L], \\ w_i &= 5e^{-i}, \\ f_i(x) &= \sin\left(\frac{(i+1)\pi(|x|-3)}{25.5}\right). \end{aligned} \tag{4.14}$$

As for the disassembly rate k_d, the shift of the base functions produces an artifact at the origin, but we will see that it is not present in the posterior distribution.

Modelling transport speed

The transport velocity $\mathbf{v}$ always points towards the nucleus, i.e. towards the origin in our model. Let $s : [-L,L] \to [-1,1]$ be a smooth approximation of the sign function, and let $v : [-L,L] \to \mathbb{R}^+$ be a smooth function, symmetric with respect to the origin. We define

$$\mathbf{v} = s(x)v(x).$$

Therefore, to define a prior for $\mathbf{v}$ we shall define a prior, according to (4.5.1), for v. The assumptions for the reaction rates are also valid here: v can have at most one peak at each side of the origin, and it must vanish at the origin due to the presence of the nucleus. The values for the prior of v are the following:

$$\begin{aligned} \xi_i &\sim \mathcal{U}[-1,1], \\ m_0(x) &= -10, \quad \forall x \in [-L,L], \\ w_i &= 4e^{-i}, \\ f_i(x) &= \sin\left(\frac{1}{2}(i+1)\pi\frac{\sqrt[3]{\frac{|x|}{27.5}}}{27.5}\right). \end{aligned} \tag{4.15}$$

In the case of the speed, we use a different set of base functions to avoid the artifact at the origin. Although the argument about the artifact not showing in the posterior would still be valid in this case, however the artifact at the origin can create numerical instabilities in the solver for the system of PDEs.

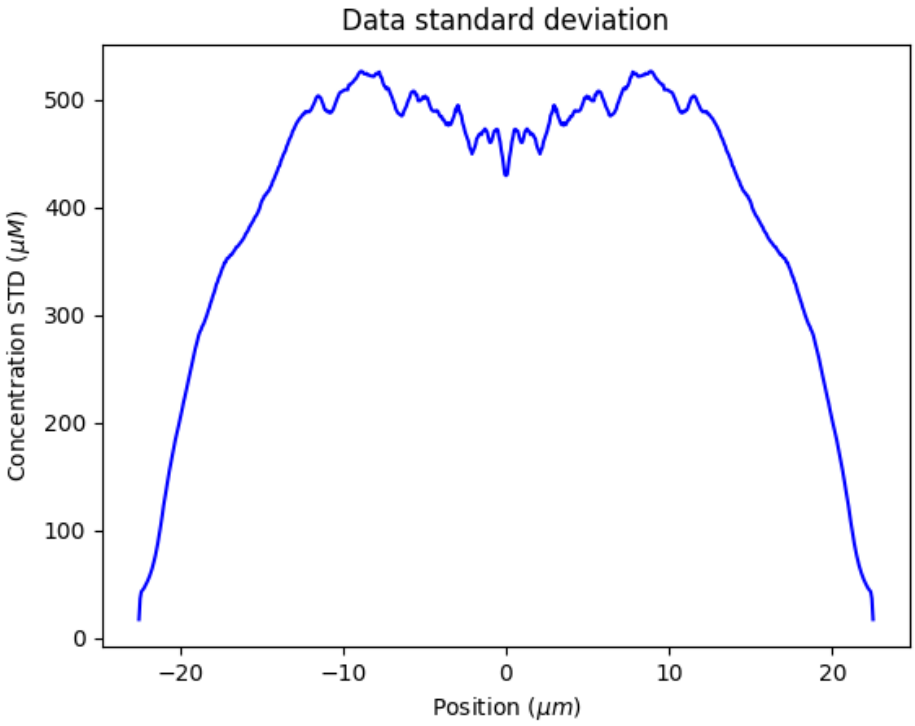


FIGURE 4.4: Standard deviation of the mean keratin concentration, computed from the raw data sets. Every point in the tidy data corresponds to the average of at least 50 data points (84 for the late-time data), and it is therefore reasonable to approximate the noise in the measurements by a Gaussian distribution with covariance matrix given by the sample standard deviation.

4.6 Numerical results

We perform the parameter identification for (4.12) using the algorithms described in Chapter 2 and the priors defined in Section 4.5.1. We run the algorithm using the University of Sussex HPC cluster, using a maximum of 80 cores. The wall clock computation time is about 70 hours. The algorithm converges in mean with a tolerance of 10^-8. We solve the system of PDEs by means of the finite element method described in Skeel and Berzins (1990).

A 95% credible region for the posterior is represented in Figure 4.6. In order to depict the credible region, we sample 1000 parameter values from the posterior distribution for the parameters, and we colour the region that contains the graph of the corresponding solutions.

Near the boundaries of the domain, the credible region is narrower. Towards the perinuclear region, the credible region has more variance, to a maximum of a range of about $100\mu M$. Figure 4.6 also presents the experimental data obtained in Section 4.2. The credible region contains almost all the experimental data points, suggesting that the model is able to explain

the data within the error range given by our knowledge of the experimental noise and the parameters.

The red line in Figure 4.6 corresponds to the solution of (4.12) using the mean value of the posterior distribution as the parameter. The mean solution reproduces the data qualitatively well, specially near the boundaries and in the nuclear region. It fails to reproduce the features of the data at position $x = \pm 10$, where the experimental results show a increase in concentration.

Figure 4.7 depicts the posterior probability distribution for the reaction rates k_a and k_d, and for the velocity $\mathbf{v}$. The mean value, represented by the red line, shows two peaks, at distance $19.27\mu m$ from the center of the cell. The assembly rate decays rapidly to zero; it is smaller than $0.5\mu M$ in the region $x \in [-9.70, 9.70]$. Note that the prior for the assembly rate included the possibility for a smaller zero-assembly region, compare with the samples in Figure 4.5. Note also that the rate assembly is not zero at the cell boundary.

The orange area in Figure 4.7 (top panel) represents the 95% credible region for the posterior distribution of the assembly rate k_a. The credible region is very narrow around the mean in the inner region of the cell. In the region $x \in [-15, 15]$ the maximum range of the credible region is $0.34\mu M$. The credible region allows for more variability in the outer region of the cell, with a maximum range of $2.79\mu M$ at $x = \pm 20.49$, near the peaks, and a range of $1.81\mu M$ at the boundary.

Although the variability in the maximum assembly rate is high, the location of the peaks is constrained. The maximum assembly rates for the reaction rates within the 95% credible region are located in the intervals $[-21, -19]$ and $[19, 21]$.

The second panel in Figure 4.7 corresponds to the posterior distribution of the disassembly rate. The disassembly rate vanishes in the nuclear region, and it is small near the boundary of the cell. The mean value of the posterior distribution for the disassembly rates peaks at $x = \pm 13.44$.

The credible region for the disassembly rate shows a low absolute variability in the distribution, although the relative variability is significant. The credible region is narrow were the disassembly rates is small. Near the peaks of the reaction rate, the credible region has a range of $14.59\mu M$. The location of the maximum disassembly rate varies; within the 95% credible region, it lies in the intervals $[-15, -12]$ and $[12, 15]$.

In contrast to the assembly rate, the location of the maximum disassembly rate is closer to

the nucleus, and at the boundary, the assembly rate does not vanish. Note also the different scales of the top and middle panels in Figure 4.7. The disassembly rate is low in comparison to the assembly rate.

The last panel in Figure 4.7 shows the posterior distribution for the speed $\mathbf{v}$. The posterior mean value, represented with a red solid line, has peaks at $x = \pm 17.18$, and it decreases to zero towards the nucleus. Although the prior includes the possibility for very sharp decays, and almost constant values away from the nucleus, we see in the posterior that the speed has two clear peaks.

The credible region shows variability near the peaks and towards the boundary, and it is narrow in the nuclear region. The maximum value for the speed ranges from $0.0019 \mu m s^{-1}$ to $0.0026 \mu m s^{-1}$ within the 95% credible region. At the boundary, the speed is in the interval $[0.0013, 0.0018] (\mu m s^{-1})$.

4.7 Conclusion and interpretation of the results

We derive the mathematical model from first principles based on experimental observations, accounting for the biological hypotheses provided by the biological experts. Our derivation ensures global conservation of keratin, which was a short-coming of the previous models. The required assumptions to obtain the model are detailed explicitly, and therefore allow for a clear understanding of what to take into account.

We justify the main limitation of the model—the restriction to only one dimension—in two ways. First, the lack of spatial orientation of the data implies that spatial heterogeneity could be either intrinsic, reflecting some characteristic spatial features of the cell, or a random event, i.e. it could happen in any direction. With the information at hand, we cannot distinguish between the two scenarios. If we assume the second case—random events—, then we could average the data *as it is*, and derive a two dimensional model on the disc, using the assumptions similar to the assumptions in Section 4.4. But there would be the possibility that we will model as noise features that actually reflect an intrinsic, deterministic characteristic of the cells, and the model will not allow us to distinguish between these scenarios. Furthermore, the amount of data is reduced in this case. With the approach that we present here, for each location in the one-dimensional domain, we get data from each radius, in each image. In particular, we only get as many data points as images in the data set for the center of the cell

(radius 0), but we get as much as 42588 data points for the locations near the cell membrane. More precisely, we get 25350 data points near the membrane for the early time data set, and 42588 for the late time data set.

Furthermore, the consensus from the experimentalists, supported by other studies, is that the keratin dynamics is predominantly in the radial direction. This justifies not taking into account mechanisms for displacement in the tangential direction. For the assembled keratin network, this is also supported by the observation that the keratin filaments move along other cell structures aligned with the radial direction. Diffusion is not naturally limited in that direction, but for simplicity we account only for diffusion in the radial direction. Since diffusion is only significant for disassembled keratin, and given that the concentrations are low, we would not expect significant changes in the results if we incorporate tangential diffusion. The effect of tangential diffusion will be higher near the origin, where in any case the accuracy of the model is lower due to the presence of the nucleus. Only in a full three dimensional model we could model the nucleus and the movement of keratin around it.

The processing of the experimental data requires only a few explicit assumptions. We extract information not only about the mean values, but also about the noise distribution. Therefore, our estimates for the noise are consistent with the experiments, and in turn we get consistent credible regions for the posterior distributions. This is one of the main features of the Bayesian framework. Since we treat the parameters not as deterministic values but as probability distribution, the level of knowledge about them is incorporated in the mathematical setting, and in consequence we can give an interpretation of the posterior probability distributions in terms of level of knowledge about the results. Note that the variability that comes from the priors is expected to vanish provided we have enough data, according to the Bernstein-von Mises theorem (see for instance Lu et al (2017)).

The results on Figure 4.6 show a great degree of accuracy. The only significant feature of the data that is not present in the solution are the local maxima at $x = \pm 10$. In contrast, the model reproduces accurately the concentration of keratin in the nuclear region. We know that the model is in fact less accurate near the nucleus, but we did not incorporate any noise from the model inaccuracy. Therefore, a plausible explanation for the discrepancies at $x = \pm 10$ is an over fitting of the model in the nuclear region.

Although the representation of the posterior distribution via the image of the solution operator is symmetric with respect to the mean, the data is located at the top half of the credible

region (see Figure 4.6). This suggests that the model is systematically underestimating the concentration of keratin.

The posterior distributions for the reaction rates confirm the hypothesis that assembly takes place near the cell membrane, and disassembly takes place near the nucleus. This can be seen in the relative position of the peaks of the assembly and disassembly reaction rates. As expected, the assembly reaction peak rate is not at the boundary, because the vertical of the cell is smaller there, and therefore there is less physical space for the keratins to assemble.

Note that our results are in agreement with the results in Moch et al (2013) with a completely different approach; in the cited paper, Moch *et al.* use an image processing, and a time sequence of images, to approximate the assembly and disassembly rates as the rates of change in luminosity. Similarly, they estimate the speed of the keratin network by tracking features of the network in the images. For comparison, see Figure 4 in Moch et al (2013) or Figure 2 in Portet et al (2015).

Portet et al (2015) present a similar model for the dynamics of the keratin network, using a different approach to model the parameters. In their paper, they define a hierarchy of models, accounting for different mechanisms: linear versus nonlinear reactions, localised versus non localised reaction rates, constant versus non constant speed. In total, they fit 36 models to the experimental data, and they use an information theory approach (the Akaike Information Criteria) to find the most suitable model, and the most relevant mechanisms in the dynamics.

Our approach, with parameters defined in a functional space, covers all the models in the hierarchy defined in Portet et al (2015), with the only exception of the linear reaction rates. We note that if linear reaction rates were a better fit, we would expect the reaction saturation parameters k_I and k_S to become larger. We do not observe this behaviour, and we are therefore confident that the Michaelis-Menten reaction type is more accurate than a linear reaction term.

For comparison, we performed the model fitting for the best model in Portet et al (2015) using our Bayesian techniques. The results, depicted in Figure 4.8 are similar to the results in Portet et al (2015). Not that although the fitted parameters minimise the distance to the data, the solution exhibits oscillations that are not present in the experimental results.

The data set in Portet et al (2015) comes from the same experiments as the data set that we use here. The data processing method is different, in particular the data is reduced to one dimension by taking the intensities on a diameter of the cell, instead of a radius. This leads to a non-symmetric data set. Our model, as well as the models in Portet et al (2015) use parameters

that are symmetric with respect to the origin, and therefore solutions will tend to a symmetric distribution, even more so when the initial data is symmetric or nearly symmetric. Therefore, the model cannot reproduce the asymmetric features of the data. This can be seen in Figure 4.8, as well as in Portet et al (2015). The solution corresponding to the posterior mean parameters fits the left hand side of the data better than the right hand side.

Note also the credible region is much narrower than in our solutions. The main reason is that the experimental noise is not modelled in Portet et al (2015). More precisely, they objective functional for minimisation is the squared (Euclidean) distance between the data and the solution, i.e. Portet et al (2015) use a least-squares method. If we interpret the least-squares objective functional in the Bayesian framework, it corresponds to assuming a Gaussian distribution for the noise, with mean zero and unit covariance. This is a much lower covariance than the values we extracted from the experimental data, suggesting that the range of the credible region in Figure 4.8 is underestimated.

In the past, researchers in biology obtained results on the dynamics of keratin assembly in disassembly using different techniques. One approach was to measure the dynamics by direct observation of the fluorescent keratin. For example, a technique known as fluorescence recovery after photobleaching (FRAP), in which a region of the cell is photobleached—the fluorescence of keratin is eliminated, and it is then possible to observe the keratin dynamics by observing how the fluorescent keratin from other parts of the cell invades the photobleached region (see for example Kölsch et al (2010); Windoffer et al (2004)).

A second approach, presented in Moch et al (2013), relies on image analysis tools to extract information from time-lapse fluorescence recordings. This approach overcomes some of the limitations of the FRAP-type approaches. In particular, the image analysis approach provides some quantification of the assembly and disassembly regions.

The results presented here offer a new approach to the study of the assembly and disassembly dynamics of keratin. The mathematical model is derived using precise assumptions, based on the current knowledge in biology. Then, by fitting the model to the experimental data, we obtain information about the parameters, and in particular, information about the assembly and disassembly regions is extracted from the posterior distributions for the assembly and disassembly rates.

There are some limitations in the mathematical model presented in this chapter. Since we only have access to data at the initial and final times, but not at the intermediate times,

we have not attempted to model the transient dynamics of keratin. At this point, it is not clear if it would be necessary to incorporate more features in the model, such as the vertical displacement of the keratin network—i.e. assembled keratin moving out of focus—in order to fit the transient dynamics; it is possible that on average, the vertical displacement of keratin does not have any effect.

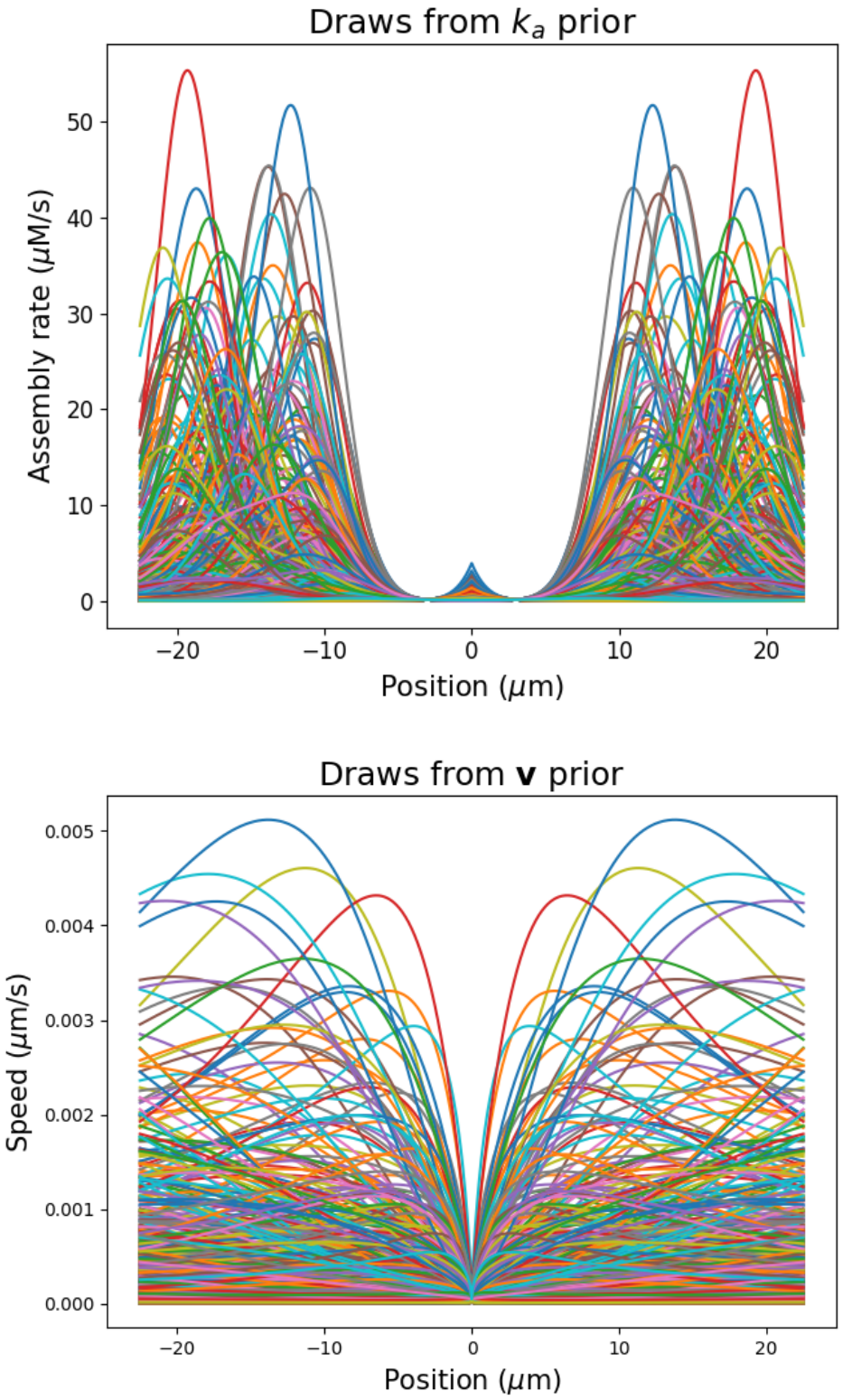


FIGURE 4.5: One thousand samples from the prior for the assembly rate k_a and the speed $\mathbf{v}$. Note that the prior for k_a produces samples with an artifact at the origin due to the shift in space of the prior base functions. The artifact is not consistent with the biology. Although it is possible to eliminate the it by refining the base functions, we can keep as long as it does not show in the posterior distributions, i.e. as long as it does not provide a better fitting than the samples consistent with the biology.

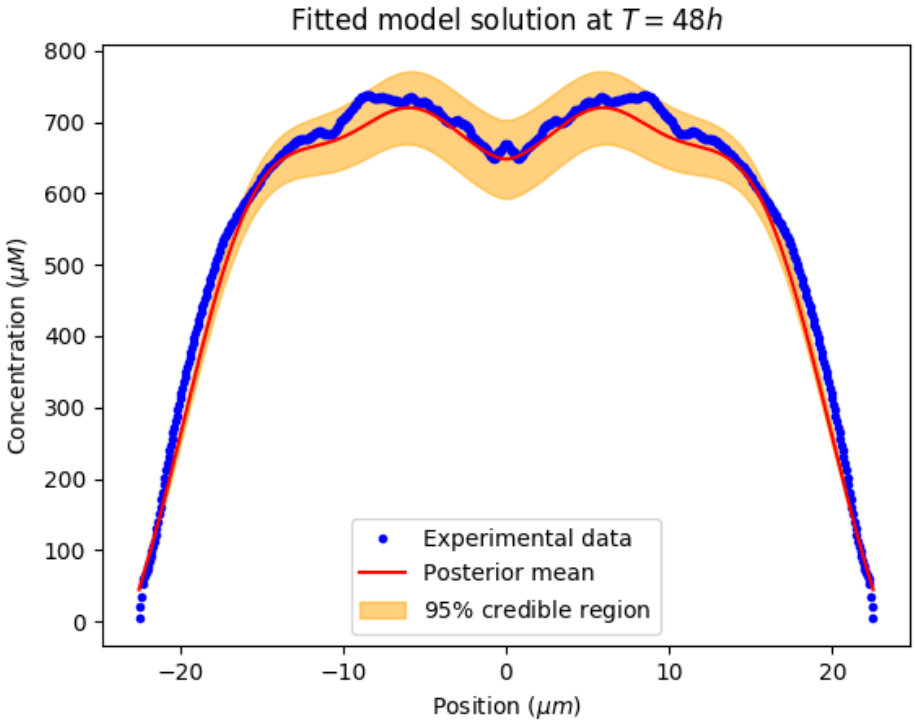


FIGURE 4.6: Solution to the model with the parameters set to the posterior mean. The credible region corresponds to the region covered by the solutions to the model within the 95% credible region of the posterior distribution. We include the experimental data for comparison. The solution to the mathematical model is close to the data both in quantitative and qualitative terms, and the data is almost completely included in the credible region.

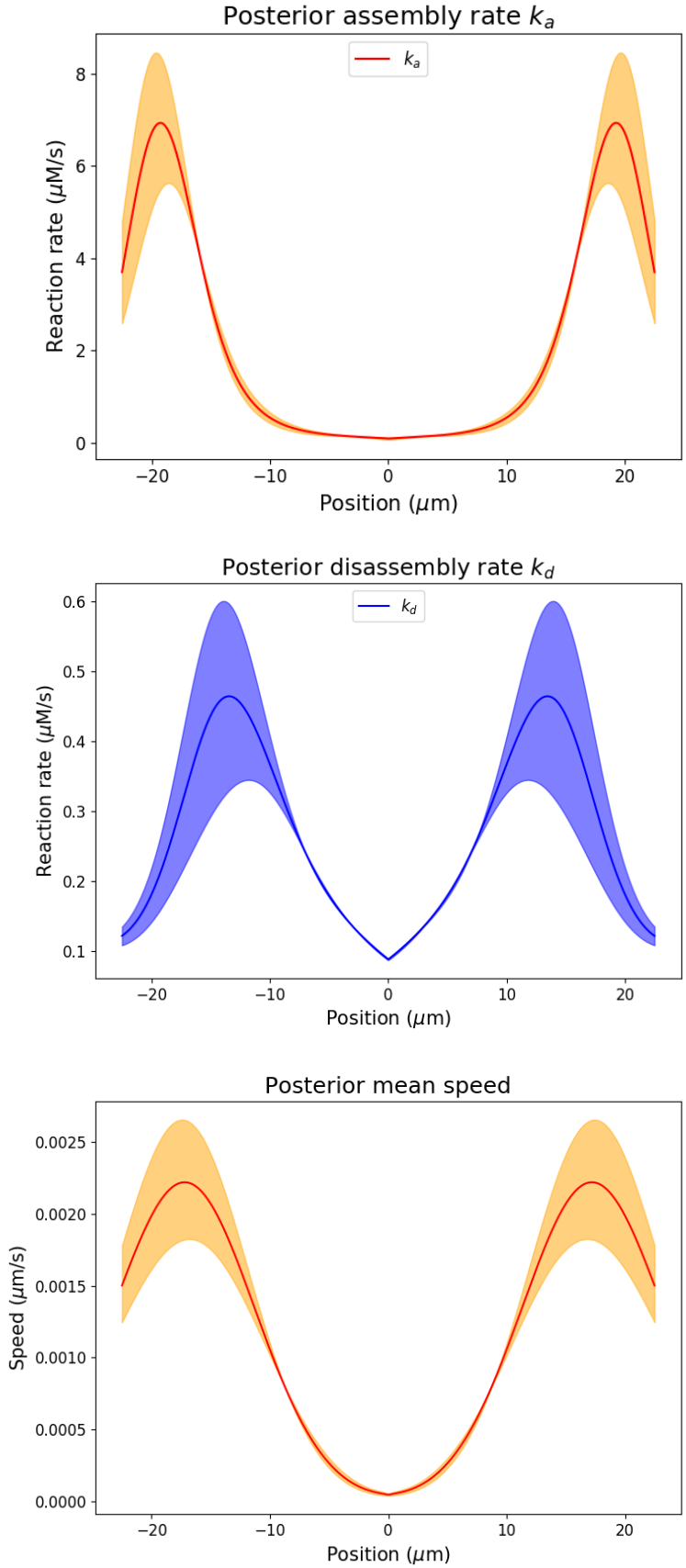


FIGURE 4.7: Mean and 95% credible region for the parameters. The parameters cannot be measured experimentally, but they agree with the estimates for the parameters obtained by means of image processing techniques, as reported in Portet et al (2015).

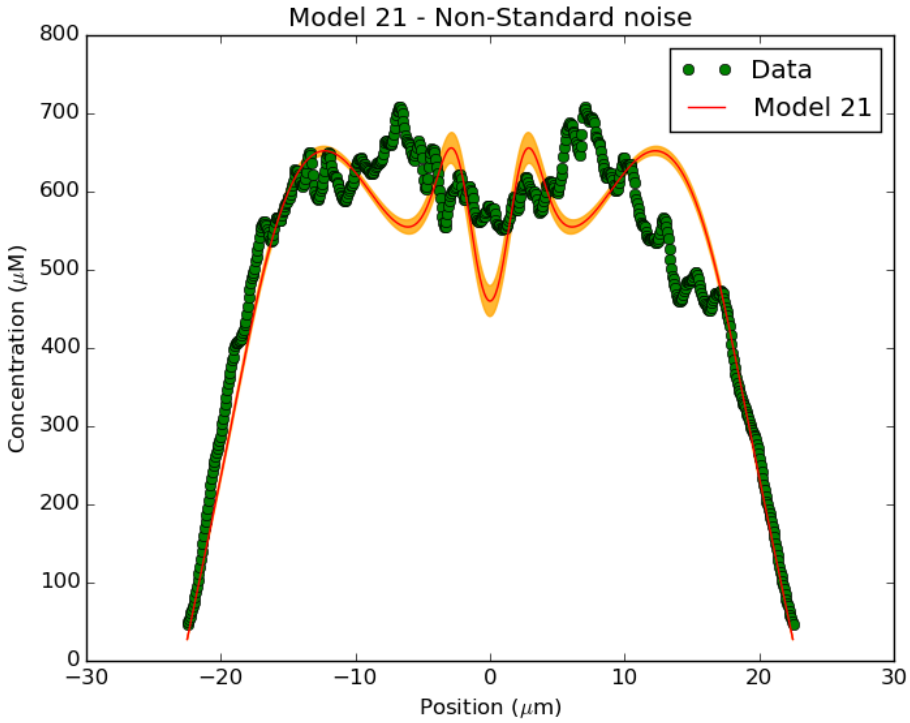


FIGURE 4.8: Using the methods presented here, we reproduce the model fitting from Portet et al (2015). "Model 21" is the model that gets the best score according to the Akaike Information Criterion. Note that the structure of the model is the same, but the priors for the parameters are different. The experimental data set is the same, but the data processing is different. The fitting is not able to reproduce the qualitative features of the data.

Chapter 5

Bayesian analysis of traction force microscopy data

5.1 Introduction

Traction force microscopy is a technique for measuring the traction forces generated by a moving cell on a substrate or surface. In its more basic setting, the cells are placed on an elastic substrate, that contains fluorescent beads. When the cell moves, it applies a force to the substrate, producing a displacement that can be measured by observing the positions of the fluorescent beads. See Wang and Lin (2007) for a general overview of the technique.

The idea of observing the mechanical effects of cell migration on the substrate to study cell locomotion goes back to Harris et al (1980), where cells are cultured on thin silicone sheets. Cell migration produces wrinkles in the silicone, that can be observed by measuring the distortion when light crosses the substrate. However, this experimental setting is hard to model mathematically, due to the inherent randomness of the wrinkling.

In contrast, Dembo and Wang (1999) and Munevar et al (2001) introduced the idea of embedding fluorescent microspheres in the substrate, and measuring their displacement. This approach is much more suitable to data analysis and mathematical modelling. Since the problem can be reduced to linear elasticity, and the mechanical properties of the substrate are known, one can use the Green's function method to explicitly solve the problem (see for instance Gould (1994)).

Traction force microscopy has many applications. In cancer biology, the experimental data

suggests that there is a strong correlation between the force that the cells generates and its invasiveness: cells that generate large forces are more invasive (Koch et al (2012); Peschetola et al (2013)). These results suggest that traction forces can be used as a proxy to estimate the invasiveness of a cancer cell line. Other applications include the optimal design of scaffolds for tissue engineering (Pasqualini et al (2018)) and regeneration of various tissues (López-Fagundo et al (2014); George et al (2014); Vedula et al (2014)). See Style et al (2014) for application in other fields, such as physics and chemistry.

Although in this work we restrict ourselves to the two-dimensional problem, the experimental techniques can also provide three-dimensional data (see for instance Koch et al (2012)). Note that our approach can be readily applied in this setting as well as in the case of anisotropic substrates, where the analytical methods have a limited applicability.

The inverse problem of computing the traction force for the displacement data is a linear inverse problem that can be studied using a range of techniques, both analytically and numerically. Schwarz and Soiné (2015) review the computational techniques for this problem. See for instance Vitale et al (2012) for an analysis of the inverse problem accounting for the pointwise observations. In the applications, most biologists rely in the modules included in popular image analysis software such as ImageJ (Abràmoff et al (2004)). These modules are typically based on the original ideas of Dembo and Wang (1999), that use the explicit solutions of the problem to identify the traction force. Kozawa et al (2012) use a Bayesian approach to introduce priors for the cell shape and improve the accuracy of the estimated traction force. Their results differ from our approach in the available data: we do not estimate the cell shape because we have information available on the focal adhesions, and therefore we know—i.e. we have data about—how the forces are distributed. Huang et al (2018) apply a Bayesian method to avoid the image filters and regularisation that are usually required when solving the problem using analytical techniques. In this thesis, we use already processed data provided by the experimentalist. Huang et al (2018) deals with the problem of how to do this process in a robust manner, and in particular they use Bayesian techniques to remove background noise from the images.

When the cell migrates, it attaches itself to the substrate at discrete locations, the focal adhesions. The location of the focal adhesions can be observed by means of immunofluorescent coloration of vinculin, a protein associated with attachment sites (Humphries et al (2007)).

Using our methodology, we compute not only the traction force but we also provide estimates on the uncertainty of the result. In the following we will illustrate this by performing the force identification and uncertainty quantification with experimental data that includes the location of the focal adhesions.

5.2 Mathematical model for the recovery of traction forces

The biological system—a cell applying a force on an elastic substrate—, corresponds to a well studied problem in continuum mechanics: linear elasticity (see for instance Gould (1994)). We assume that the substrate is isotropic, and enough regularity on the boundary to have smooth solutions. Both assumptions are based on the actual experimental setting. In particular, the substrate is large in comparison to the area of influence of a cell, and the problem of regularity of the boundary has no actual effects on the displacement produced by the cell traction.

Let $\Omega \subset \mathbb{R}^2$ be the domain, and let $\mathbf{u}(x)$ represent the displacement of the point $x \in \Omega$ under a force $f(x)$. Then, $\mathbf{u}$ satisfies

$$\operatorname{div}(\mathbb{C} : \nabla \mathbf{u}) = f. \tag{5.1}$$

The system is closed by prescribing boundary conditions. Since we consider measurements of isolated cells, we assume homogeneous Dirichlet boundary conditions on the domain, representing a fixed boundary. The product $\mathbb{C} : \nabla \mathbf{u}$ represents the stress tensor, and it can be described in terms of the Lamé parameters λ and μ such that

$$\mathbb{C} : \nabla \mathbf{u} = 2\mu \nabla \mathbf{u} + \lambda \operatorname{tr}(\nabla \mathbf{u}) I.$$

The goal of parameter identification in traction force microscopy is to compute the force f given the experimental measurement of the displacement $\mathbf{u}$. Using the notation introduced in Chapter 2, the observation operator $\mathcal{G}$ maps a force f to the discrete measurements of the displacement, that corresponds to the values of the solution $\mathbf{u}$ to Equation (5.1) evaluated at the locations of the fluorescent beads.

In this setting, $\mathcal{G}$ is linear with respect to the parameter f, and this suffices to conclude the well-posedness of the inverse problem, provided that f lives in a sufficiently smooth space.

Furthermore, in this case, if the prior distribution for the force f and the noise distribution are Gaussian, then the posterior distribution is also Gaussian (see for instance Stuart (2010)).

5.3 Data, noise and priors

During the experiment, the cells are placed on an elastic substrate with embedded fluorescent beads. The positions of the beads at rest are measured at the end of the experiment. The position of the beads is measured also when a cell is applying a force. By computing the difference between these two measurements, one obtains the displacement of each bead due to the cell force. Note that a drift correction is applied. The drift is computed in a bead in the boundary of the domain, where we assume that the cell force does not cause displacement.

In the same experiment, the position of the focal adhesions is measured, using cells that exhibit fluorescence at the focal adhesions. Since the focal adhesions are the points of attachment of the cell to the substrate, we assume that the force is applied only at focal adhesions.

The data is represented in Figure 5.1. The displacement field is represented with black arrows, scaled for convenience of visualisation. The red dots correspond to the focal adhesions.

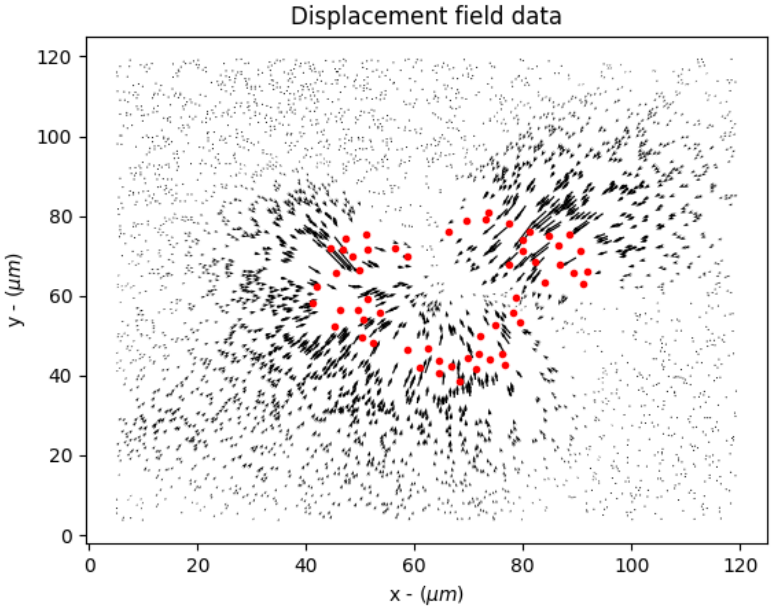


FIGURE 5.1: Measured displacement field and positions of the focal adhesions.

The data is captured in one picture, and hence the experimental error is a combination of resolution limit and optical error. We will not attempt to model its precise distribution, but rather assume a Gaussian noise with a conservative estimate of the covariance. More precisely, we assume that the standard deviation of the measurement of the position of the fluorescent beads is $2\mu m$.

We do not assume any restriction on the force direction in the prior. The prior consists of a collection of independent Gaussian variables centered at the focal adhesions. At each focal adhesion, the standard deviation of the force is $4 \cdot 10^{-9} N/\mu m^2$. This corresponds to the force to be applied at only one point to produce the largest measured displacement.

The typical size of a focal adhesion is of the order of $1\mu m$. Therefore, we smooth the force applied at each focal adhesion over a circle of radius $5\mu m$, by multiplying by a smooth cut-off function defined as

$$\phi(x) = \begin{cases} \frac{\exp\left(\frac{-1}{1-|x-x_0|^2/5^2}\right)}{e^{-1}}, & \text{if } |x - x_0| < 5, \\ 0 \quad \text{otherwise,} \end{cases} \tag{5.2}$$

where x_0 is the location of the corresponding focal adhesion. Note that this is a smooth mollifier with support in the ball of radius 5 around the focal adhesion.

5.4 Numerical methods

The partial differential equation (5.1) is solved numerically using a Finite Element Method (see for instance Ern and Guermond (2013)) using the standard Lagrange elements. The equation is discretized over a grid of 6400 elements (80×80), refined around focal adhesions to a total of $13,520$ elements. The solver is implemented in Python using Fenics (Dupont et al (2003)).

The posterior probability distribution is approximated using a Markov-chain Monte Carlo method. A first exploration is done using an independence sampler (i.e. proposals are independent samples from the prior). A pre-conditioned Crank-Nicholson method (Cotter et al (2013)) is used to provide a detailed approximation of the posterior. A total of $500,000$ states of the Markov-Chain are computed, and then they are thinned to $25,000$ independent samples. All the numericals experiments are performed on a desktop computed with the specifications Intel(R) i7-4790 CPU @ 3.60GHz, 8 cores, 64 bits, 16G RAM.

5.5 Computational results

Figure 5.2 represents the displacements corresponding to the posterior mean for the force. The posterior mean force is represented in Figure 5.3. Note that despite not imposing any

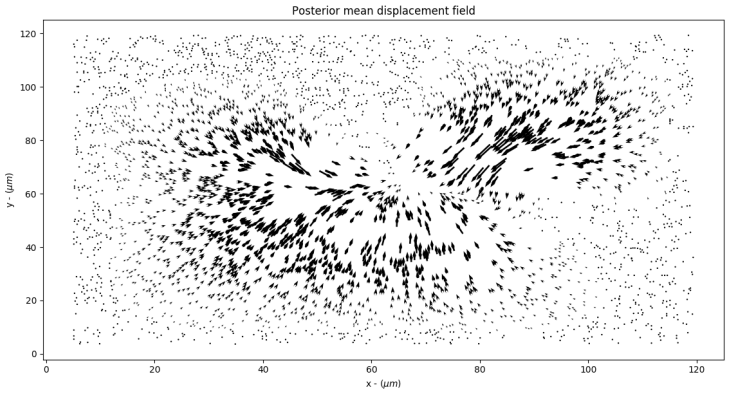


FIGURE 5.2: Displacement field corresponding to the posterior mean force.

restriction on the direction of the force, the forces tend to point to the central region of the cell. The uncertainty for the force is low, and cannot be appreciated at the scale of the cell. In

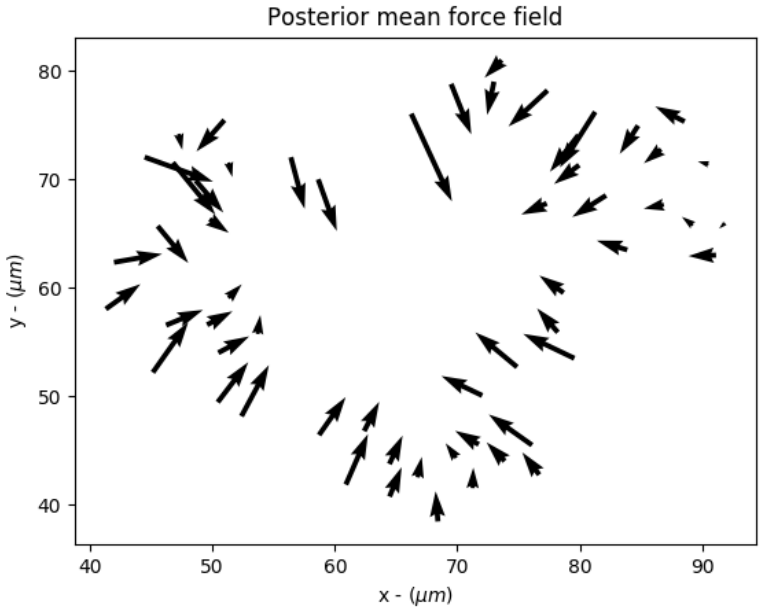


FIGURE 5.3: Force field corresponding to the posterior mean force.

Figure 5.4 we represent the prior and posterior credible regions corresponding to 3 standard

deviations for the focal adhesion located at $(41.297\mu m, 58.043\mu m)$. As mentioned above, the posterior credible region is not centered at the focal adhesion, but rather displaced towards the central region of the cell.

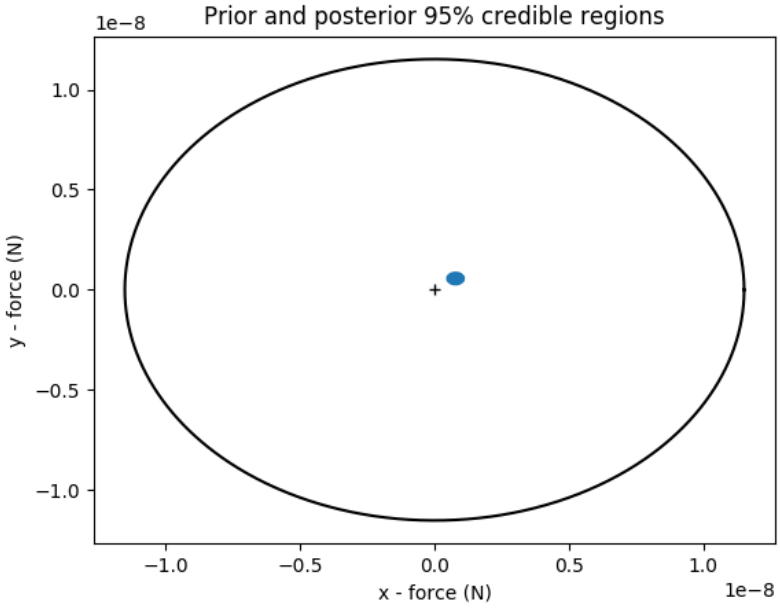


FIGURE 5.4: Prior (black solid line) and posterior (blue) credible regions for the focal adhesion at $(41.297\mu m, 58.043\mu m)$.

5.6 Discussion

A common application of Bayesian methods in traction force motility is to use the priors to restrict the direction of the cell force. This idea is based on empirical observations and mechanical considerations: since the cell produces traction by contraction of polymer fibers, it cannot easily produce expanding forces.

Here we proceeded without imposing any distinguished direction in the priors, and we can in turn conclude that according to the data, the forces point towards the central region of the cell. We can also observe clearly the different force intensities at different focal adhesions.

The credible region for the posterior distribution shows how robust this methodology is. Even with our very conservative estimate of the error, the posterior distribution is very concentrated, both in the direction and in the magnitude. An interesting question for a future

project is how this uncertainty would change with a more accurate derivation of the noise distribution.

The uncertainty will also be reduced if a more informed prior is available. In Figure 5.4 we can see that the uncertainty reduces about ten times from the prior to the posterior. The low uncertainty suggests that this techniques can be applied to infer characteristics of individual cells, without the need of averaging cell population data.

Note that as discussed Chapter 2, we do not need to impose any technical condition to solve the problem. In particular, we have not applied any regularisation to the data. The only manipulation of the data previous to the parameter identification is the drift correction. Although the drift correction is quite precise, we could incorporate the drift as a parameter in the model, with a precise prior defined by the experimental measurement.

From a methodology point of view, for this problem we use a numerical solver implemented in Fenics, which is completely independent of our parameter identification approach. This illustrates the flexibility of our implementation. Note also that in this case, we do not use an HPC environment for the computations. A regular desktop PC with 8 cores is enough to obtain this results in a few hours. Although this is a significantly longer time in comparison to solving the inverse problem using analytical techniques, we apply more transparent methods—in particular, we do not use any regularisation—, and we obtain the uncertainty quantification results.

5.6.1 Future work

The model that we use here is simple, but the general methodology remains the same if the model is changed to a three-dimensional domain. Another extension will be to include anisotropic materials, very common in the applications of TFM to tissue engineering.

Even with our simple model, there are a number of questions that we could study if there was enough experimental data. First, we could not use the information about the focal adhesions, and instead used a functional prior for a force distributed inside the contour of the cell. The cell contour itself could be modelled as a parameter, following the ideas in Kozawa et al (2012). There are two interesting questions in this case: to what extent do we observe a predominant central directionality of the traction forces, and how well can we locate the focal adhesions using the inverse problem approach and validating the results against the experimental data. The first question relates to the importance of using priors restricting the force,

whilst the second question may offer an alternative way to observe the location of the focal adhesions without the need of immunofluorescence experiments.

Chapter 6

Conclusion

In this thesis, we proposed a methodology to apply a Bayesian framework to parameter identification problems in quantitative biology. The methodology is robust and based on existing theoretical results, and it is designed to be as transparent as possible to the biologist. Although both the Bayesian framework that we use, and the parallel MCMC algorithm are known results, their combination and implementation is a novel contribution from this thesis. The examples in Chapters 3, 4 and 5 illustrate the novelty and power of our approach.

In some situations, there might be concerns about the efficiency of our methods. As discussed in 1, we pay a price in terms of efficiency due to our interest in keeping the parameter identification algorithms independent of the solver. This places the main limit to the applicability of our approach, but the parallel scalability of our algorithms ensures that we can study a wide range of problems. Again, the examples in Chapters 3, 4 and 5 show that our methods can be applied in a medium-size HPC facility, or even with a desktop PC in the case of 5.

Another common criticism comes from the statistics field. This has more to do with the fact that the term *Bayesian* is an umbrella term for many different techniques, and in turn different people have different interpretations of the Bayesian framework. In statistics, Bayesian techniques are well established and have been applied to numerous problems, including PDEs. Our approach is slightly different in philosopy—the interpretation of the distributions, and the idea relies on using expert information for the definitions, instead of technical decisions—, and also in the applications to PDEs, it is based on rigorous theoretical results that are usually not available in statistics.

The results in this work help to close the gap between experimental results and mathematical models. Even when good mathematical models are in place, many times they remain *applicable* but not *applied* due to the difficulties in incorporating the experimental knowledge and

data in the modelling. A methodology that is both satisfactory for the mathematicians—i.e. it is based on rigorous results—and for the biologist is essential to bridge this gap. Furthermore, as we discussed in Chapter 1, a feedback cycle between the mathematician and experimentalist provides improvements in both fields, with better, more accurate mathematical models and also deeper knowledge for the biologist.

The results in Chapter 3 are already published (Campillo-Funollet et al, 2018), together with a general description of our approach, whilst manuscripts for the results in Chapters 4 and 5 are in preparation.

6.1 Other applications in quantitative biology

We are already applying the techniques presented here to other problems. For example, with V. Juma, S. Portet, L. Dehmelt and A. Madzvamuse, we are using our methodology to an ODE model for Rho GTPases. The inverse problem in this case includes a constraint on the eigenvalues of the linearised system. Our methods are flexible enough to incorporate this constraint.

We are also currently applying this approach to new models for genome replication. In this case, the parameters are in general functions of the position along the DNA molecule. The mathematical model is given by a one-dimensional, hyperbolic PDE.

Also in the field of genomics, we plan to apply our methods to study the replication of the kinetoplast DNA (kDNA) in trypanosomes. In this case, we are still in the early stages of the model development, but we expect to also have space-dependent parameters in connection to the topology of the kDNA.

6.2 Future work

In future works, we plan to release our software as an open source application. In view of the applications that we presented here, we believe that this software will be well received by the experimentalists in various fields. There are several tasks to complete before being able to publish the software for general use. First, the documentation is limited now, and it needs to be improved to enhance the final user experience. Some parts of the code, specially the sampling routines within the parallel Metropolis-Hastings methods can be reimplemented for efficiency.

The parallel structure of the code can also be improved to allow runtime management of the available cores. The idea here will be to allow the code to assign more processors to an instance of the solver when necessary. This of course requires solvers that are more efficient when several cores are available.

Another feature that can be included in the software is model selection methods. The information provided by the posterior distribution can be used in several model selection approaches and these could be easily implemented to allow the comparison of different mathematical models. Similarly, we can use our approach to build robust classifiers, that instead of using only one estimated value for a sample, use its posterior distribution, thus allowing a more accurate description of the classification.

In Chapters 3, 4 and 5 we already mentioned some extensions of the work. Inverse problems related to the Turing patterns have plenty of applications. There are experimental results that pinpoint the exact Turing mechanism underlying pattern formation, see for instance Glover et al (2017), or for a more controlled setting, the chemical reactions in Castets et al (1990). The application of our methodologies to experimental data for this model paradigm would provide nice insights on the actual parameter regimes in the systems.

Our results on the keratin dynamics suggest a promising approach to use mathematical models as a proxy to measure parameters of a biological system. A next step to ensure the robustness of the approach is to improve the analytical results on the model, in particular to ensure positivity of solutions. A more challenging question is the extension of the models to use two-dimensional data. Ideally, we would like to be able to fit a model without any normalisation of the cell shape, and without any averaging.

In traction force microscopy, the information on uncertainty quantification, together with the fact that we do not regularise the data, can be useful to make robust predictions on the cell state and behaviour based on the tractions. For example, the classification of cells for cancer invasiveness based on its tractions can be improved by means of the credible regions of the posterior.

6.2.1 Applications in other fields

We presented a methodology with the goal of applying it to problems in quantitative biology, but the methods that we presented are general and can be applied to many other fields.

For instance, we successfully applied the approach that we presented in this thesis to the study of the crime dynamics in Cape Town, South Africa. In this case, the mathematical model is an ODE model, and our parameter identification method allows us to show that the crime in an average police station is in an increasing regime, towards a stable steady state. The credible regions for the parameters allow us to show that significant changes in policy would be required to change this tendency. A manuscript of this study is in preparation.

Appendix A

The *pmh* module

```
"""
Parallel Metropolis-Hastings methods and Optimal Control methods for \
    parameter identification.

by Eduard Campillo-Funollet

May 2015 to August 2017.

"""

import numpy as np #Numerical tools
from scipy.linalg import eig #Eigenvector calculation.
import scipy.optimize as sco #Optimization routines.
import pickle as pk #Save/Load to file
import os #OS tools
import multiprocessing as mp#Parallelism
import logging #Logs used to monitorize tasks
import time #Time methods

def setSeed(seed):
        """
        Sets the seed for all number generations to be done. For \
        reproducibility.
```

```
        """

        np.random.seed(seed=seed)


class mc(object):
        """
        Markov chain and related information

        id - list -lookup table- to find state given id.
        Contains pairs [chunk_number,indext within chunk]
        states - list of states that appear (or not!) in the chain
        logl - list of log-likelihoods of states
        chain - list of indices corresponding to the states in the list \
        states.
        The actual Markov Chain
        accepted - list of true and false values.
        True means the state comes from an acceptace step.

        Note: The states are numpy arrays, but "states","logl",... \
        are Python lists.

        Note: It is optimized to work fast for adding and iterative \
        reading.

        Filename codification: basename_C, where C is the chunk number
        -states,logl-. basename_data -chain,accepted,id-
        """

        def __init__(self,basename="testmc",limSize=10000):
                #Attributes initialization
                self.id = []
```

```
            self.reset_chunk() #Sets states and logl to empty lists.
            self.chain = []
            self.accepted = []

        #Current number of states. It is used to generate ids.
            self.counter = 0
            self.current_chunk = 0
            self.last_chunk = 0

            #Store arguments
            self.basename = basename
            self.limSize = limSize
            #Load the chain if already exists.
            if os.path.isfile(self.getDataName()):
                    self.load()

        def reset_chunk(self):
            """
            Reset the loaded chunk
            """
            self.states = []
            self.logl = []

        def getChunkName(self,c):
            """
            Filename for chunk c
            """
            return self.basename+"_"+str(c)+".p"

        def getDataName(self):
            return self.basename+"_data.p"
```

```
    def save_data(self):
            """
            Save id,chain and accepted. Also last chunk.
            """
            data = {"id":self.id,"chain":self.chain,\
        "accepted":self.accepted, "last":self.last_chunk,\
        "limSize":self.limSize}

            pk.dump(data,open(self.getDataName(),"wb"))

    def load_data(self):
            """
            Loads the data.
            """
            data = pk.load(open(self.getDataName(),"rb"))

            self.id, self.chain, self.accepted,self.last_chunk = \
        data["id"],data["chain"],data["accepted"],data["last"]
            self.limSize = data["limSize"]


    def save_chunk(self):
            """
            Saves the current chunk to a file.
            """
            pk.dump({"states":self.states,"logl":self.logl},\
        open(self.getChunkName(self.current_chunk),"wb"))
            self.save_data() #Save the current data.

    def load_chunk(self,c):
            """
            Load the chunk number c
```

```
            WARNING: It does not save the current chunk before \
        replacing it!
            """

            chunk = pk.load(open(self.getChunkName(c),"rb"))
            self.states,self.logl = chunk["states"],chunk["logl"]
            self.current_chunk = c

    def load(self):
            """
            Loads data and first chunk.
            """
            self.load_data()
            self.load_chunk(0)

    def save(self):
            """
            Saves data and current chunk
            """
            #self.save_data() #Now data is saved when a chunk is saved,
    #so save_chunk is enough.
            self.save_chunk()

    def addBlock(self,states,logl,chain,accepted):
            """
            Adds a new block of states to the chain.
            Note that states in self.states may be duplicated!
            """

            #Integrity check
            if len(states) != len(logl) or len(chain) != len(accepted):
```

```
                raise ValueError("Non-matched dimensions in \
                        the new block.")

        #Set current chunk to last
        if self.current_chunk != self.last_chunk:
                self.save_chunk()
                self.load_chunk(self.last_chunk)
                self.current_chunk = self.last_chunk

        shift = len(states)
        L = len(self.states)
        id_shift = len(self.id)
        newIds = [[self.current_chunk,L + k] for k in range(shift)]

        #By now the id referes to an index in the self.states list.
        self.id += newIds
        self.states += states
        self.logl += logl
        self.chain += [state_index+id_shift for state_index in chain]
        self.accepted += accepted

        if self.getSize() > self.limSize:
                self.save_chunk()
                self.reset_chunk()
                self.last_chunk += 1
                self.current_chunk = self.last_chunk

def getState(self,id):
        """
        Get state with given id

        WARNING
```

```
            It does not save the current chunk if \
        it has to load a different one.
            """
            c,i = self.id[id][0],self.id[id][1]

            if self.current_chunk != c:
                    self.load_chunk(c)

            return self.states[i]

    def getLogl(self,id):
            """
            Get logl with given id

            WARNING
            It does not save the current chunk \
        if it has to load a different one.
            """
            c,i = self.id[id][0],self.id[id][1]

            if self.current_chunk != c:
                    self.load_chunk(c)

            return self.logl[i]

    def getChain(self):
            """
            Returns the actual Markov chain
            """

            return [self.getState(id) for id in self.chain]
```

```
    def getAcceptance(self):
            """
            Returns acceptance/rejection list.
            """
            return self.accepted
    def __iter__(self):
            """
            Allows iteration over an instance of the class
    as iteration over the chain.
            """
            return iter(self.getChain())


    def __getitem__(self,i):
            """
            Allows indexing.
            """
            return self.getState(self.chain[i])


    def __len__(self):
            """
            Allows len(mc)
            """
            return len(self.chain)


    def getSize(self):
            """
            Returns the approximate memory size of the states attribute.


            It will be used as a criteria for file storage.


            Note: It requires the states to be numpy arrays!
            """
```

```
                if len(self.states) == 0:
                        return 0

                return self.states[0].nbytes * len(self.states)

        def getStateLogl(self):
                """
                Returns two Python lists,
                -the MC, i.e. list of states
                -the logl of the states in the MC
                """

                pairs = [ [self.getState(id),self.getLogl(id)] \
            for id in self.chain]

                return [p[0] for p in pairs],[p[1] for p in pairs]



class pa(object):
        """
        Proposal - acceptance class. Implements different samplers.
        is - Independence sampler
        pCN - preconditioned Crank - Nicholson
        gCN - pCN with generic sampler.
        """

        def __init__(self,sampler,*args,**kwargs):
                """
                sampler is a string.
```

```
        "is" - independence sampler. args[0]=prior sampler.
        "pCN" - preconditioned Crank-Nicolson. args[0]=covariance of
    prior (np array (matrix)). args[1]=beta
                Note: pCN assumes normal prior with 0 mean.
    For other means, shift the problem.
        """
        if "cutoff" in kwargs:
                self.cutoff = lambda x,f=kwargs["cutoff"]: f(x)
        else:
                self.cutoff = None

        if sampler == "is":
                self.proposer = lambda x,f=args[0]: f(x)
                self.compute_prob = self.compute_acceptance_std
                self.sample = self.sample_acum



        elif sampler == "pCN":
                self.proposer = lambda x,C=args[0],\
        b=args[1]:self.proposal_pcn(C,b,x)
                self.compute_prob = self.compute_acceptance_std
                self.sample = self.sample_acum

                if "range" in kwargs:
                        self.range = kwargs["range"]
                        self.cutoff = self.in_range
                else:
                        self.range = None

        elif sampler == "wp":
                self.proposer = lambda x,f=args[0],\
        b=args[1]: self.proposal_wp(f,b,x)
```

```
                self.compute_prob = self.compute_acceptance_std
                self.sample = self.sample_acum

        elif sampler == "bp":
                self.proposer = lambda x,f=args[0],\
        r=args[1]: self.proposal_bp(f,r,x)
                self.compute_prob = self.compute_acceptance_std
                self.sample = self.sample_acum

        self.proposal = lambda x: self._proposal(self.proposer,x,\
    self.cutoff)

def _proposal(self,proposer,x,cutoff):
        """
        Proposal wrapper, to take cutoff into account.
        """
        if not cutoff:
                return proposer(x)

        prop = proposer(x)

        while not cutoff(prop):
                prop = proposer(x)

        return prop


def compute_acceptance_std(self,logl):
        """
        Computes the acceptance probability of each proposal.
        Standard: applies to independence sampler and pCN.
```

```
            """

            rows = [] #Rows of transition matrix.

            for i in range(len(logl)):
        #Mask to distinguish the "current" state.
                mask = range(len(logl))
                mask.remove(i) #Remove the current state from the mask.
                row = np.array(logl)

                row[mask] = row[i] - row[mask] #phi(u)-phi(v)
        #Take minimum (i.e. cutoff to 1 in the exp).
                row[mask] = np.array([min(0,row[k]) for k in mask])
                row[mask] -= np.log(len(row)-1) #When exp, division by N
                row[mask] = np.exp(row[mask]) #Compute exponential.
                row[i] = 1 - sum(row[mask])

                rows.append(row)

            A = np.vstack(rows) #Transition matrix

            w,v = eig(A,left=True,right=False)

    #Eigenvector of eigenvalue (closest to) 1
            p = v[:,np.argmin(abs(1.-w))]

            p = p / sum(p) #Normalization

            self.compute_acum(p)

    def in_range(self,prop):
            """
```

```
            self.range is an iterable with pairs [xmin,xmax],[ymin,ymax],...
            """
            for r,p in zip(self.range,prop):
                    if p < r[0] or p > r[1]:
                            return False

            return True

    def proposal_wp(self,prior,beta,current):
            """
            Weighted average.
            beta = 0 -> we stay at current state.
            beta = 1 -> independence sampler
            """
            return (1-beta)*current + beta*prior(current)

    def proposal_bp(self,prior,radius,current):
            """
            Proposals in a ball of maximum radius radius.
            """
            prop = prior(current)
            norm = np.linalg.norm(prop-current)

            if norm == 0:
                    return current

            return current + \
        np.random.uniform(0,radius)*(prop-current)/norm


    def proposal_pcn(self,covar,beta,current):
            """
```

```
            Generates a sample using the pCN sampler \
        with the given parameters

            Note: Assumes mean 0. In any other case, shift the problem.
            """

            return  np.sqrt(1-beta**2)*current + \
        beta*np.random.multivariate_normal(np.zeros(len(covar)),\
                                           covar)

    def compute_acum(self,logl):
            """
            Compute acumulated probability.
            """

            self.acum = np.zeros(len(logl))

            self.acum[0] = logl[0]

            for k in range(1,len(logl)):
                    self.acum[k] = self.acum[k-1] + logl[k]

    def sample_acum(self):
            """
            Samples from self.acum
            """

            r = np.random.uniform(0,1)

            k=0

            while self.acum[k] < r:
```

```
                        k = k + 1

                return k

class pmh(object):
        """
        Main class.
        Run the Parallel Metropolis Hastings method.

        Requires:
                A log-likelihood function phi.
                A proposal acceptance.
                A mc instance to store the chain.
                Initial state.

        Parameters of the Parallel Metropolis Hastings method:
        Npool - number of workers. If "all", uses all available processors.
                                                        If 0<Npool<1, uses that fracti
                                the available processors.
        N - #states to be proposed each step
        (i.e. N+1 states in total, N proposals + 1 current)
        M - #states to be sampled each step (default M=N)
        K - Evaluations per worker.
        If defined, overrides the value of N and sets N = Npool*K + 1

        maxT - Maximum number of seconds to run (wallclock time)
        maxDate - Deadline for the process to run (POSIX time)
        targetIt - target number of states to generate.

        Notes:
        Logs are saved using the Markov Chain basename.log
        Defaults are to a 1 worker - standard Parallel MH.
```

```
        The stopping conditions can be set to a negative value \
        to be ignored.  If all of them are negative, \
        an error is raised.
        """

    def __init__(self,pa,mc,initialState,Npool=1,N=1,M=-1,K=-1,\
        maxT=-1, maxDate=-1,targetIt=-1):
            #Check that a stopping condition is provided
            if maxT < 0 and maxDate < 0 and targetIt < 0:
                    raise ValueError("Invalid stoppping condition.")

            #Save stopping conditions
            self.stopConditions = {"maxT":maxT,"maxDate":maxDate,\
                            "targetIt":targetIt}

            #Set the number of workers
            if Npool == "all":
                    self.Npool = mp.cpu_count()
            elif 0<Npool and Npool<1:
                    self.Npool = int(Npool*mp.cpu_count())
            else:
                    self.Npool = Npool

            #Set the number of states.
            if K > 0:
                    self.N = self.Npool*K
            else:
                    self.N = N

            #Set the number of samples per step
            if M > 0:
                    self.M = M
```

```
            else:
                    self.M = self.N

            #Basic attributes.
            self.pa = pa
            self.mc = mc
            self.initialState = initialState

            #Configure logging
            logging.basicConfig(filename=self.mc.basename+".log",\
                        level=logging.DEBUG)


    def getPresentConditions(self,startT,it):
            """
            Returns a dictionary to use to check the stopping conditions.
            """
            return {"maxT":time.time()-startT,\
            "maxDate":time.time(),"targetIt":it}

    def checkStopping(self,presentConditions):
            """
            Check the stopping conditions. \
    Returns True if it is time to stop.
            """
            for cond in self.stopConditions:
                    if self.stopConditions[cond] > 0 and \
            presentConditions[cond]>self.stopConditions[cond]:
                            return True

            #If we arrive here, the conditions are not satisfied.
            return False
```

```
def getConfigString(self):
        """
        Header of the log file.
        """

        tx="Configuration: \nNumber of workers: "+ \
    str(self.Npool)+"\nNumber of states: "+ \
    str(self.N+1)+"\nNew states per step: "+ \
    str(self.M)+"\n\nStopping conditions: \n"

        if self.stopConditions["targetIt"] > 0:
                tx += "Target chain length: "\
        +str(self.stopConditions["targetIt"])+"\n"
        if self.stopConditions["maxT"] > 0:
                tx += "Maximum running time: "\
        +str(self.stopConditions["maxT"])+"\n"
        if self.stopConditions["maxDate"] > 0:
                tx += "Deadline: "\
        +str(self.stopConditions["maxDate"])+"\n"

        tx+="\nInitial state: "+str(self.initialState)+"\n"

        return tx

def getStepString(self,stepTime,nstates,startT):
        """
        Debug of each step.
        Includes stepTime and stepTime based estimations of
    remaining time and candidate stopping condition.
        """
        tx = "Last step: "+str(stepTime)+"\n"
```

```
            timeToGoals = []
            stop = []

            if self.stopConditions["targetIt"]>0:
                    timeToGoals.append((self.stopConditions["targetIt"]\
            -nstates)*stepTime/self.M)
                    stop.append("target chain length will be reached.")

            if self.stopConditions["maxT"] > 0:
                    timeToGoals.append(self.stopConditions["maxT"]-\
            (time.time()-startT))
                    stop.append("maximum running time will be reached.")

            if self.stopConditions["maxDate"] > 0:
                    timeToGoals.append(self.stopConditions["maxDate"]\
             - time.time())
                    stop.append("deadline will be reached.")

            t = min(timeToGoals)
            cause = stop[timeToGoals.index(t)]

            tx+="Remaining time "+str(t)+" because "+cause

            return tx


    def run(self,phi):
            """
            Runs the algorithm.
            """
            #Log the configuration.
```

```
        logging.info(self.getConfigString())

        #Initializes the pool of workers.
        pool = mp.Pool(self.Npool)

        #Initializes proposals and logl lists
        proposals = [None]*(self.N+1)
        logl = [None]*(self.N+1)
        proposals[0] = self.initialState
        #logl[0] = self.phi(self.initialState)
        logl[0] = phi(self.initialState)
        i = 0 #Current state.

        #Store the start time. Initializes number of states counter.
        startT = time.time()
        nstates = 0

        while not self.checkStopping(\
                self.getPresentConditions(startT,nstates)):
    #For logging and estimation of total time.
                step_startT = time.time()
                #Prepare the mask to the proposals.
                mask = range(self.N+1)
                mask.remove(i)

                #Generate proposals
                save = proposals[i]
                proposals = [self.pa.proposal(proposals[i])\
         for k in range(self.N)]
                proposals.insert(i,save)
                #Now the parallelized evaluation of the log-likelihood.
                save = logl[i]
```

```
                logl = pool.map(phi,[proposals[k] for k in mask])
                logl.insert(i,save)

                #The instance of pa takes care of the computations.
                self.pa.compute_prob(logl)

                #Generate the new block of states and acceptance vector.
                newState = [self.pa.sample()]
                if newState[0] == i:
                        acc = [False]
                else:
                        acc = [True]

                for m in range(self.M-1):
                        newState.append(self.pa.sample())
                        if newState[-1] == newState[-2]:
                                acc.append(False)
                        else:
                                acc.append(True)

                #Add the new block to the chain.
                self.mc.addBlock(proposals,logl,newState,acc)

                #Update current state, nstates
                i = newState[-1]
                nstates += self.M

                logging.debug(self.getStepString(\
        time.time()-step_startT,nstates,startT))

        #Save the chain to file.
        self.mc.save()
```

```
#Close the pool
pool.close()
pool.join()

logging.info("Total running time: "+str(time.time()-startT))
```

Appendix B

The *pp* module

```
"""
Postprocessing module for the PMH.

Included as a different module to avoid import matplotlib \
    in when not necessary.

Requires:
        pmh

by Eduard Campillo-Funollet
May 2015 to September 2015
"""

import pmh
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
from scipy import stats,interp

class pp(object):
        """
        Class to encapsulate postprocessing methods.
        It receives an instance of mc when instanciated \
```

```
        or a string with the basename.
        """

    def __init__(self,mc,preload=True):
            if isinstance(mc,basestring):
                    self.mc = pmh.mc(basename=mc)
            else:
                    self.mc = mc

            if preload == True:
        #Loads the chain to avoid excesive file IO.
                    self.chain = self.mc.getChain()
            else:
                    self.chain = None

    def getLen(self):
            """
            Returns lengnth of the chain.
            """

            if self.chain is not None:
                    return len(self.chain)
            else:
                    return len(self.mc)

    def getDim(self):
            """
            Returns the dimension of the states.
            """
            if self.chain is not None:
                    return len(self.chain[0])
```

```
        return len(self.mc[0])


    def scatter2D(self,c1,c2,mask=None,style='.',**kwargs):
        """
        Scatter plot component c1 vs. c2
        """
        if self.chain is not None:
            if mask == None:
                chain = np.array([[s[c1],s[c2]] for s in self.chain])
            else:
                chain = np.array([[s[c1],s[c2]] \
                for s in [self.chain[k] for k in mask]])
        else:
            if mask == None:
                chain = np.array([[s[c1],s[c2]] for s in self.chain])
            else:
                chain = np.array([[s[c1],s[c2]] \
                for s in [self.chain[k] for k in mask]])

        plt.figure()
        plt.plot(chain[:,0],chain[:,1],style,**kwargs)
        plt.show(block=False)

    def hist(self,c,mask=None,**kwargs):
        """
        Plots the histogram of the coordinate c of the chain.
        kwargs passed to plt.hist.
        mask allows to use only part of the chain.
        """
        plt.figure()

        if self.chain is not None:
```

```
                        if mask == None:
                                plt.hist([s[c] for s in self.chain],**kwargs)
                        else:
                                plt.hist([s[c] for s in [self.chain[k] for k in mask]],**k
                else:
                        if mask == None:
                                plt.hist([s[c] for s in self.mc],**kwargs)
                        else:
                                plt.hist([s[c] for s in [self.mc[k] for k in mask]],**kwar

                plt.show(block=False)

        def hist2d(self,c1,c2,mask=None,noPlot=False,**kwargs):
                """
                Plots 2D-histogram of the coordinates c1,c2.
                kwargs passed to plt.hist2d
                mask allows to use only part of the chain.
                noPlot = True -> returns counts, xedges, yedges,
            and does not show the plot.
                """
                plt.figure()

                if self.chain is not None:
                        if mask == None:
                                counts,xedges,yedges,img = plt.hist2d([s[c1] \
                    for s in self.chain],[s[c2] \
                        for s in self.chain],**kwargs)
                        else:
                                counts,xedges,yedges,img = plt.hist2d([s[c1] \
                    for s in [self.chain[k] for k in mask]],[s[c2] \
                        for s in [self.chain[k] for k in mask]],**kwargs)
                else:
```

```
                        if mask == None:
                                counts,xedges,yedges,img = plt.hist2d([s[c1] \
                    for s in self.mc],[s[c2] \
                        for s in self.mc],**kwargs)
                        else:
                                counts,xedges,yedges,img = plt.hist2d([s[c1] \
                    for s in [self.mc[k] for k in mask]],[s[c2] \
                        for s in [self.mc[k] for k in mask]],**kwargs)



                if noPlot:
                        return counts,xedges,yedges
                else:
                        plt.show(block=False)

        def kde(self,c,mask=None,res=100,**kwargs):
                """
                Plots Kernel Density Estimation for the coordinate c.
                kwargs passed to scipy.stats.gaussian_kde
                mask allows to use only part of the chain
                res - resolution, number of points.
                """

                if self.chain is not None:
                        if mask == None:
                                chain = [s[c] for s in self.chain]
                        else:
                                chain = [s[c] for s in [self.chain[k] for k in mask]]
                else:
                        if mask == None:
                                chain = [s[c] for s in self.mc]
                        else:
```

```
                chain = [s[c] for s in [self.mc[k] for k in mask]]


        X = np.linspace(min(chain),max(chain),res)
        kernel = stats.gaussian_kde(chain,**kwargs)
        Z = kernel(X)

        plt.figure()
        plt.plot(X,Z)
        plt.show(block=False)

    def kde2d(self,c1,c2,mask=None,res=100j,noPlot=False,**kwargs):
        """
        Plots Kernel Density Estimation for the coordinates c1,c2.
        kwargs passed to scipy.stats.gaussian_kde
        mask allows to use only part of the chain
        res - resolution, number of points.
        noPlot - returns chain,X,Y,Z instead of plotting.
        """

        if self.chain is not None:
            if mask == None:
                chain = np.array([[s[c1],s[c2]] for s in self.chain])
            else:
                chain = np.array([[s[c1],s[c2]] \
                for s in [self.chain[k] for k in mask]])
        else:
            if mask == None:
                chain = np.array([[s[c1],s[c2]] for s in self.chain])
            else:
                chain = np.array([[s[c1],s[c2]] \
                for s in [self.chain[k] for k in mask]])
```

```
            xmin,xmax,ymin,ymax = \
        min(chain[:,0]),max(chain[:,0]),\
            min(chain[:,1]),max(chain[:,1])
            X,Y = np.mgrid[xmin:xmax:res,ymin:ymax:res]
            positions = np.vstack([X.ravel(),Y.ravel()])
            kernel = stats.gaussian_kde(chain.T,**kwargs)
            Z = np.reshape(kernel(positions).T,X.shape)

            if noPlot:
                    return chain,X,Y,Z
            else:
                    plt.figure()
                    plt.imshow(np.rot90(Z),\
            extent=[xmin,xmax,ymin,ymax],aspect="auto")
                    plt.show(block=False)

    def contours(self,c1,c2,mask=None,res=100j,\
            fractions=[0.3,0.6,0.9], \
            noPlot=False,labels=True,**kwargs):
            """
            Plots contour curves for components c1,c2 distribution,
        at levels fractions. kwargs to gaussian_kde.

            noPlot = True, then return axes,contours \
        and does not show the plot.

            """
            chain,X,Y,Z = self.kde2d(c1,c2,mask=mask,\
        res=res,noPlot=True,**kwargs)

            plt.figure()
```

```
            axes = plt.axes()

            contours = axes.contour(X,Y,Z,200,extend='both')

            levs = contours.levels
            fracs = np.array(fracs_inside_contours(chain[:,0],\
            chain[:,1],contours))
            sortinds = np.argsort(fracs)
            levs = levs[sortinds]
            fracs = fracs[sortinds]

            levels = interp(fractions,fracs,levs)

            for coll in contours.collections:
                    coll.remove()

            contours.__init__(axes,X,Y,Z,levels)

            if noPlot:
                    return axes,contours
            else:
                    if labels:
                            frac_label_contours(chain[:,0],chain[:,1],contours)

                    plt.show(block=False)

    def credible_region(self,c1,c2,mask=None,res=100j,cred=0.95,\
    method="kde",ret=False,**kwargs):
            """
            Plots credible region of credibility cred.
```

```
        Method kde uses kde, hist uses the histogram.
        ret == True, then returns the figure or axes.
        """
        if method == "kde":
                axes,contours = self.contours(c1,c2,mask=mask,res=res,\
        fractions = [cred],noPlot=True,labels=False,**kwargs)

                for (icollection,collection) \
        in enumerate(contours.collections):
                        path = collection.get_paths()[0]
                        patch = patches.PathPatch(path,\
            facecolor='orange',lw=2)
                        axes.add_patch(patch)

                plt.show(block=False)

                if ret:
                        return axes

        elif method == "hist":
                counts,xedges,yedges = \
        self.hist2d(c1,c2,mask=mask,noPlot=True,**kwargs)

                barea = (xedges[1]-xedges[0])*(yedges[1]-yedges[0])
    #Mass normalized to 1.
                ncounts = counts / tmass(counts,barea)
                region_mass = 0.

                region = np.zeros(ncounts.shape)

                while region_mass < cred:
                        #print "Region mass ", region_mass
```

```
                imax,jmax = max_counts(ncounts) #Find max mass element.
        #Add the mass of the element to the total
                region_mass +=ncounts[imax][jmax]*barea
                region[jmax][imax] = 1. #This point is in the region.
        #This point does not contribute anymore.
                ncounts[imax][jmax] = 0.

            xcoord = midedge(xedges) #Coordinates of x axis
            ycoord = midedge(yedges) #y axis

            fig = plt.figure()
            plt.pcolormesh(np.array(xcoord),np.array(ycoord),region)
            plt.show(block=False)

            if ret:
                return fig

        else:
            print "Unknown method."



    def getAcceptance(self):
        """
        Returns acceptance rate.
        """
        return float(sum(self.mc.getAcceptance()))/self.getLen()

    def getRejection(self):
        """
        Returns rejection rate.
        """
```

```
            return 1 - self.getAcceptance()


    def plotAcceptance(self,*args):
            """
            Plots acumulated number of accepted states.
            If an argument is passed,
        it is used as acceptance chain \
        (to be used to plot rejections).
            """
            if len(args)==0:
                    acc = self.mc.getAcceptance()
            else:
                    acc = args[0]

            s = [sum(acc[0:1])] #1 if first element is true, 0 otherwise.

            for i in range(1,len(acc)):
                    s.append(s[i-1]+acc[i])

            plt.figure()
            plt.plot(range(len(acc)),s)
            plt.show(block=False)

    def plotRejection(self):
            """
            Plots acumulated number of rejected states.
            """
    #Negate acceptance -> rejection list.
            rej = [not s for s in self.mc.getAcceptance()]

            self.plotAcceptance(rej)
```

```
    def plotState(self,c,mask=None):
            """
            Evolution of coordinate c
            """

            if self.chain is not None:
                    if mask == None:
                            y = [s[c] for s in self.chain]
                    else:
                            y = [s[c] for s in [self.chain[k] for k in mask]]
            else:
                    if mask == None:
                            y = [s[c] for s in self.mc]
                    else:
                            y = [s[c] for s in [self.mc[k] for k in mask]]




            plt.figure()
            plt.plot(range(len(self.mc)),y)
            plt.show(block=False)

    def getMean(self,c,mask=None):
            """
            Computes the mean of component c
            mask allows to use only part of the chain.
            """

            if self.chain is not None:
                    if mask == None:
                            res = np.mean([s[c] for s in self.chain])
                    else:
```

```
                        res = np.mean([s[c] \
                for s in [self.chain[k] for k in mask]])
            else:
                    if mask == None:
                        res = np.mean([s[c] for s in self.mc])
                    else:
                        res = np.mean([s[c] \
                for s in [self.mc[k] for k in mask]])


            return res

    def getStateMean(self,mask=None):
            """
            Return a list of the mean of each component
            """

            res = []
            for c in range(self.getDim()):
                    res.append(self.getMean(c,mask=mask))

            return res

    def getCorr(self,c,mask=None):
            """
            Computes the normalized autocorrelation of component c

            Autocorrelation(X)[k] = 1/(sqrt(N*Var(X))sum(x_i * x_{i+k})

            mask allows to use only part of the chain.
            """
```

```
            if self.chain is not None:
                    if mask == None:
                            ch = [s[c] for s in self.chain]
                    else:
                            ch = [s[c] for s in [self.chain[k] for k in mask]]
            else:
                    if mask == None:
                            ch = [s[c] for s in self.mc]
                    else:
                            ch = [s[c] for s in [self.mc[k] for k in mask]]

            #Normalize
            ch = (np.array(ch) \
            - np.mean(ch))/np.sqrt(len(ch)*np.var(ch-np.mean(ch)))

            return np.correlate(ch,ch,mode='full')

    def getStateCorr(self,mask=None):
            """
            Returns a list of the normalized \
        autocorrelation of each component
            """

            res = []
            for c in range(self.getDim()):
                    res.append(self.getCorr(c,mask=mask))

            return res

    def getMode2D(self,c1,c2,mask=None,res=100j,method="kde",**kwargs):
            """
            Returns the maximum of the probability density, \
```

```
        approximated by method.
            """
            if method == "kde":
                    chain,X,Y,Z = self.kde2d(c1,c2,mask=mask,res=res,\
            noPlot=True,**kwargs)

                    argmax = np.where(Z == Z.max())

                    return X[argmax[0][0],argmax[1][0]],Y[argmax[0][0],\
            argmax[1][0]]

            elif method == "hist":
                    counts,xedges,yedges = self.hist2d(c1,c2,mask=mask,\
            noPlot=True,**kwargs)

                    barea = (xedges[1]-xedges[0])*(yedges[1]-yedges[0])
        #Mass normalized to 1.
                    ncounts = counts / tmass(counts,barea)

                    imax,jmax = max_counts(ncounts) #Find max mass element.

                    xcoord = midedge(xedges) #Coordinates of x axis
                    ycoord = midedge(yedges) #y axis

                    return xcoord[imax],ycoord[jmax]

    def plotMeans(self,c,res = 10, mask=None, **kwargs):
            """
            Divides the chain in res parts and plot the means of each part.
            """
            if self.chain is not None:
                    if mask == None:
```

```
                        ch = [s[c] for s in self.chain]
                else:
                        ch = [s[c] for s in [self.chain[k] for k in mask]]
        else:
                if mask == None:
                        ch = [s[c] for s in self.mc]
                else:
                        ch = [s[c] for s in [self.mc[k] for k in mask]]

        l = len(ch) // res #Points per part.

        means = [np.mean(ch[k*l:(k+1)*l]) for k in range(res)]

        plt.figure()
        plt.plot(range(len(means)),means,**kwargs)
        plt.show(block=False)

def plotCumMean(self,c,res = 100, mask = None, **kwargs):
        """
        Divides the chain in res parts \
    and plot the mean up to that point.
        """
        if self.chain is not None:
                if mask == None:
                        ch = [s[c] for s in self.chain]
                else:
                        ch = [s[c] for s in [self.chain[k] for k in mask]]
        else:
                if mask == None:
                        ch = [s[c] for s in self.mc]
                else:
                        ch = [s[c] for s in [self.mc[k] for k in mask]]
```

```
                l = len(ch) // res #Points per part.

                means = [np.mean(ch[0:(k+1)*l]) for k in range(res)]

                plt.figure()
                plt.plot(range(len(means)),means,**kwargs)
                plt.show(block=False)




def chain_recombination(chains,bname,burns=0.,ret=False):
        """
        Combines the chains in a new chain with name basename.
        chains can be either a (Python) list of strings \
        or a list of mc instances.
        bname is a string, basename of the new chain.
        burns can be either a float or a list of floats.
        It is the fraction of the chain to be burned.
        If burns is a float, the same burn out will \
        be applied to each chain.
        If ret is True, the mc instance with the new chain is returned.
        """

        #Init
        #Create mc instances of the chains.
        for i in range(len(chains)):
                if isinstance(chains[i],basestring):
                        chains[i] = pmh.mc(basename=chains[i])

        #Fill the burns list.
```

```
        if not isinstance(burns,list):
                burns = [burns]*len(chains)

        #Error handling
        if len(chains) != len(burns):
                raise ValueError("Number of chains must be \
                            equal to number of burns")

        #The new chain.
        mc = pmh.mc(basename=bname)

        for chain,burn in zip(chains,burns):
                #Get states and logls
                states,logls = chain.getStateLogl()

                #Burn out
                states = [states[i] \
            for i in range(int(burn*len(chain)),len(chain))]
                logls = [logls[i] \
            for i in range(int(burn*len(chain)),len(chain))]

                mc.addBlock(states,logls,range(len(states)),\
            [True]*len(states))

        #Save the chain, return if necessary.
        mc.save()

        if ret:
                return mc


#Auxiliary methods. Adapted from github/roban/plot2Ddist
```

```
def frac_inside_poly(x,y,path):
    """Calculate the fraction of points x,y inside path .


        path
    """
    xy = np.vstack([x,y]).transpose()
    return float(sum(path.contains_points(xy)))/len(x)


def fracs_inside_contours(x, y, contours):
    """Calculate the fraction of points x,y \
        inside each contour level.


    contours -- a matplotlib.contour.QuadContourSet
    """
    fracs = []
    for (icollection, collection) in enumerate(contours.collections):
        path = collection.get_paths()[0]
        frac = frac_inside_poly(x,y,path)
        fracs.append(frac)
    return fracs


def frac_label_contours(x, y, contours, format='%.3f'):
    """Label contours according to the fraction of points x,y inside.
    """
    fracs = fracs_inside_contours(x,y,contours)
    levels = contours.levels
    labels = {}
    for (level, frac) in zip(levels, fracs):
        labels[level] = format % frac
    contours.clabel(fmt=labels)


#Auxiliary functions for histogram based credible region.
```

```
def tmass(counts,barea):
    """
    Compute the total mass
    Assumes equidistant edges.
    """
    totalmass = 0.

    for i in range(len(counts)):
        for j in range(len(counts[i])):
            totalmass += counts[i][j]*barea

    return totalmass

def max_counts(ncounts):
    """
    Returns the coordinates i,j of the max in ncounts.
    """
    indmax = []
    maxrows = []

    for i in range(len(ncounts)):
        maxrows.append(max(ncounts[i]))
        indmax.append(np.where(ncounts[i] == max(ncounts[i]))[0][0])

    imax = maxrows.index(max(maxrows))
    jmax = indmax[imax]

    return imax,jmax

def midedge(edges) :
    """
    Returns a vector of length len(edges)-1, \
```

```
        with the midpoints of edges
    """
    medges = []

    for i in range(len(edges)-1):
        medges.append((edges[i]+edges[i+1])/2.)

    return medges

#Test chain generator.
def gaussian(x):
        """
        Gaussian distribution mean 0, variance 1.
        """
        return np.linalg.norm(x)**2

def uniform2d(x):
        """
        Uniform proposal in -10,10 square.
        """
        return np.array([np.random.uniform(-10,10),\
        np.random.uniform(-10,10)])

def testchain(name,length=1000):
        """
        Creates a MC for testing purposes.

        Gaussian, two components explored using independence sampler.
        """

        c = pmh.mc(basename=name)
        pa = pmh.pa("is",uniform2d)
```

```
    mh = pmh.pmh(pa,c,uniform2d(0),Npool="all",targetIt=length)

    mh.run(gaussian)

    c.save()

    return c
```

Bibliography

Abràmoff MD, Magalhães PJ, Ram SJ (2004) Image processing with imagej. Biophotonics international 11(7):36–42

Adby P (2013) Introduction to Optimization Methods. Springer Science & Business Media

Apte A, Hairer M, Stuart AM, Voss J (2007) Sampling the posterior: An approach to non-Gaussian data assimilation. Physica D: Nonlinear Phenomena 230(1):50–64, DOI 10.1016/j.physd.2006.06.009

Ashyraliyev M, Fomekong-Nanfack Y, Kaandorp JA, Blom JG (2009) Systems biology: Parameter estimation for biochemical models

Aster RC, Borchers B, Thurber CH (2013) Parameter Estimation and Inverse Problems. Academic Press

Beazley D (2010) Understanding the python gil. In: PyCON Python Conference. Atlanta, Georgia

Beck JV, Blackwell B, Jr CRSC (1985) Inverse Heat Conduction: Ill-Posed Problems. James Beck

Belgacem FB (2012) Identifiability for the pointwise source detection in fishers reaction–diffusion equation. Inverse problems 28(6):065,015

Bezanson J, Karpinski S, Shah VB, Edelman A (2012) Julia: A fast dynamic language for technical computing. arXiv preprint arXiv:12095145

Blazakis KN, Madzvamuse A, Reyes-Aldasoro CC, Styles V, Venkataraman C (2015) Whole cell tracking through the optimal control of geometric evolution laws. Journal of Computational Physics 297:495–514

Brown KS, Sethna JP (2003) Statistical mechanical approaches to models with many poorly known parameters. Phys Rev E 68(2):021,904, DOI 10.1103/PhysRevE.68.021904

Calderhead B (2014) A general construction for parallelizing Metropolis-Hastings algorithms. Proceedings of the National Academy of Sciences 111(49):17,408–17,413, DOI 10.1073/pnas.1408184111

Campillo-Funollet E, Venkataraman C, Madzvamuse A (2018) Bayesian parameter identification for turing systems on stationary and evolving domains. Bulletin of mathematical biology pp 1–24

Castets V, Dulos E, Boissonade J, De Kepper P (1990) Experimental evidence of a sustained standing turing-type nonequilibrium chemical pattern. Physical Review Letters 64(24):2953

Chou CF, Riopel CL, Rott LS, Omary MB (1993) A significant soluble keratin fraction in simpleepithelial cells. lack of an apparent phosphorylation and glycosylation role in keratin solubility. Journal of Cell Science 105(2):433–444

Cotter SL, Roberts GO, Stuart AM, White D, others (2013) MCMC methods for functions: Modifying old algorithms to make them faster. Statistical Science 28(3):424–446

Crampin E, Hackborn W, Maini P (2002) Pattern formation in reaction-diffusion models with nonuniform domain growth. Bulletin of mathematical biology 64(4):747–769

Crampin EJ, Gaffney EA, Maini PK (1999) Reaction and diffusion on growing domains: Scenarios for robust pattern formation. Bulletin of mathematical biology 61(6):1093–1120

Crestel B, Alexanderian A, Stadler G, Ghattas O (2017) A-optimal encoding weights for nonlinear inverse problems, with application to the helmholtz inverse problem. Inverse problems 33(7):074,008

Croft W, Elliott C, Ladds G, Stinner B, Venkataraman C, Weston C (2015) Parameter identification problems in the modelling of cell motility. Journal of Mathematical Biology 71(2):399–436, DOI 10.1007/s00285-014-0823-6

Das D (2016) Turing pattern formation in anisotropic medium. Journal of Mathematical Chemistry pp 1–14

Das SK, Mazumdar C, Banerjee K (2014) Gpu accelerated novel particle filtering method. Computing 96(8):749–773

Dashti M, Stuart AM (2016) The bayesian approach to inverse problems. Handbook of Uncertainty Quantification pp 1–118

Del Moral P (2004) Feynman-kac formulae. In: Feynman-Kac Formulae, Springer, pp 47–93

Dembo M, Wang YL (1999) Stresses at the cell-to-substrate interface during locomotion of fibroblasts. Biophysical journal 76(4):2307–2316

Dewar MA, Kadirkamanathan V, Opper M, Sanguinetti G (2010) Parameter estimation and inference for stochastic reaction-diffusion systems: application to morphogenesis in d. melanogaster. BMC Systems Biology 4(1):21

Dudley RM (2018) Real Analysis and Probability: 0. Chapman and Hall/CRC

Dupont T, Hoffman J, Johnson C, Kirby RC, Larson MG, Logg A, Scott LR (2003) The fenics project. Tech. rep., Tech. Rep. 2003–21, Chalmers Finite Element Center Preprint Series

Eck C, Garcke H, Knabner P (2017) Mathematical Modeling. Springer

Engelhardt B, Kschischo M, Fröhlich H (2017) A bayesian approach to estimating hidden variables as well as missing and wrong molecular interactions in ordinary differential equation-based mathematical models. Journal of The Royal Society Interface 14(131):20170,332

Engl HW, Flamm C, Kügler P, Lu J, Müller S, Schuster P (2009) Inverse problems in systems biology. Inverse Problems 25(12):123,014

Ern A, Guermond JL (2013) Theory and practice of finite elements, vol 159. Springer Science & Business Media

Fan W, Bouguila N, Ziou D (2012) Variational learning for finite Dirichlet mixture models and applications. IEEE transactions on neural networks and learning systems 23(5):762–774

Feng X, Zhang H, Margolick JB, Coulombe PA (2013) Keratin intracellular concentration revisited: Implications for keratin function in surface epithelia. The Journal of investigative dermatology 133(3):850

Friedman A (2018) Mathematical Biology: Modeling and Analysis, vol 127. American Mathematical Soc.

Friedman A, Reitich F (1992) Parameter identification in reaction-diffusion models. Inverse Problems 8(2):187

Gábor A, Villaverde AF, Banga JR (2017) Parameter identifiability analysis and visualization in large-scale kinetic models of biosystems. BMC systems biology 11(1):54

Garvie MR, Trenchea C (2014) Identification of space-time distributed parameters in the gierer–meinhardt reaction-diffusion system. SIAM Journal on Applied Mathematics 74(1):147–166

Garvie MR, Maini PK, Trenchea C (2010) An efficient and robust numerical algorithm for estimating parameters in Turing systems. Journal of Computational Physics 229(19):7058–7071, DOI 10.1016/j.jcp.2010.05.040

Gelman A, Simpson D, Betancourt M (2017) The prior can often only be understood in the context of the likelihood. Entropy 19(10):555

George J, Aratyn-Schaus Y, Nesmith AP, Pasqualini FS, Alford PW, Parker KK (2014) The contractile strength of vascular smooth muscle myocytes is shape dependent. Integrative Biology 6(2):152–163

Gierer A, Meinhardt H (1972) A theory of biological pattern formation. Kybernetik 12(1):30–39, DOI 10.1007/BF00289234

Glover JD, Wells KL, Matthäus F, Painter KJ, Ho W, Riddell J, Johansson JA, Ford MJ, Jahoda CA, Klika V, et al (2017) Hierarchical patterning modes orchestrate hair follicle morphogenesis. PLoS biology 15(7):e2002,117

Gould PL (1994) Introduction to linear elasticity. Springer

Greenberg CS, Birckbichler PJ, Rice RH (1991) Transglutaminases: multifunctional cross-linking enzymes that stabilize tissues. The FASEB Journal 5(15):3071–3077

Guiu-Souto J, Muñuzuri AP (2015) Influence of oscillatory centrifugal forces on the mechanism of Turing pattern formation. Physical Review E 91(1):012,917

Hamilton SL (2013) An Introduction to Parallel Programming. Lulu. com

Harris AK, Wild P, Stopak D (1980) Silicone rubber substrata: a new wrinkle in the study of cell locomotion. Science 208(4440):177–179

Hasenauer J, Jagiella N, Hross S, Theis FJ (2015) Data-driven modelling of biological multi-scale processes. Journal of Coupled Systems and Multiscale Dynamics 3(2):101–121

Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1):97–109

Hawkins-Daarud A, Prudhomme S, van der Zee KG, Oden JT (2013) Bayesian calibration, validation, and uncertainty quantification of diffuse interface models of tumor growth. Journal of mathematical biology 67(6-7):1457–1485

Helfand BT, Chang L, Goldman RD (2004) Intermediate filaments are dynamic and motile elements of cellular architecture. Journal of cell science 117(2):133–141

Herberich G, Würflinger T, Sechi A, Windoffer R, Leube R, Aach T (2010) Fluorescence microscopic imaging and image analysis of the cytoskeleton. In: Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on, IEEE, pp 1359–1363

Hu X, Liu Y, Xu X, Feng Y, Zhang W, Wang W, Song J, Wang Y, Zhao W (2015) Spatiotemporal evolution of a cosine-modulated stationary field and Kerr frequency comb generation in a microresonator. Applied optics 54(29):8751–8757

Huang Y, Schell C, Huber TB, Simsek AN, Hersch N, Merkel R, Gompper G, Sabass B (2018) Traction force microscopy with optimized regularization and automated bayesian parameter selection for comparing cells. arXiv preprint arXiv:181005848

Humphries JD, Wang P, Streuli C, Geiger B, Humphries MJ, Ballestrem C (2007) Vinculin controls focal adhesion formation by direct interactions with talin and actin. The Journal of cell biology 179(5):1043–1057

Hundsdorfer W, Verwer JG (2013) Numerical solution of time-dependent advection-diffusion-reaction equations, vol 33. Springer Science & Business Media

Huttunen JM, Kaipio JP (2007) Approximation errors in nonstationary inverse problems. Inverse problems and imaging 1(1):77

Iglesias MA, Lin K, Stuart AM (2014) Well-posed bayesian geometric inverse problems arising in subsurface flow. Inverse Problems 30(11):39, DOI 10.1088/0266-5611/30/11/114001

Itakura E, Tamiya S, Morita K, Shiratsuchi H, Kinoshita Y, Oshiro Y, Oda Y, Ohta S, Furue M, Tsuneyoshi M (2001) Subcellular distribution of cytokeratin and vimentin in malignant rhabdoid tumor: three-dimensional imaging with confocal laser scanning microscopy and double immunofluorescence. Modern Pathology 14(9):854

Jaitovich A, Mehta S, Na N, Ciechanover A, Goldman RD, Ridge KM (2008) Ubiquitin-proteasome-mediated degradation of keratin intermediate filaments in mechanically stimulated a549 cells. Journal of Biological Chemistry

Jaynes ET (1968) Prior probabilities. IEEE Transactions on systems science and cybernetics 4(3):227–241

Jiang L, Ou N (2017) Multiscale model reduction method for bayesian inverse problems of subsurface flow. Journal of Computational and Applied Mathematics 319:188–209

Jones E, Oliphant T, Peterson P, others (2001–) SciPy: Open Source Scientific Tools for Python. [Online; accessed 2016-01-22]

Kaipio J, Somersalo E (2006) Statistical and Computational Inverse Problems. Springer Science & Business Media

Kaipio J, Somersalo E (2007) Statistical inverse problems: Discretization, model reduction and inverse crimes. Journal of Computational and Applied Mathematics 198(2):493–504, DOI 10.1016/j.cam.2005.09.027

Kantas N, Beskos A, Jasra A (2014) Sequential monte carlo methods for high-dimensional inverse problems: A case study for the navier–stokes equations. SIAM/ASA Journal on Uncertainty Quantification 2(1):464–489

Kass R, Wasserman L (1996) The Selection of Prior Distributions by Formal Rules. Journal of the American Statistical Association 91(435):1343–1370, DOI 10.1080/01621459.1996.10477003

Kass R, Carlin B, Gelman A, Neal R (1998) Markov Chain Monte Carlo in Practice: A Roundtable Discussion. The American Statistician 52(2):93–100, DOI 10.1080/00031305.1998.10480547

Koch TM, Münster S, Bonakdar N, Butler JP, Fabry B (2012) 3d traction forces in cancer cell invasion. PloS one 7(3):e33,476

Kölsch A, Windoffer R, Würflinger T, Aach T, Leube RE (2010) The keratin-filament cycle of assembly and disassembly. J Cell Sci 123(13):2266–2272

Kozawa S, Sakumura Y, Toriyama M, Inagaki N, Ikeda K (2012) An estimation of cell forces with hierarchical bayes approach considering cell morphology. In: International Conference on Neural Information Processing, Springer, pp 501–508

Lacitignola D, Bozzini B, Frittelli M, Sgura I (2017) Turing pattern formation on the sphere for a morphochemical reaction-diffusion model for electrodeposition. Communications in Nonlinear Science and Numerical Simulation

Lakkis O, Madzvamuse A, Venkataraman C (2013) Implicit–Explicit Timestepping with Finite Element Approximation of Reaction–Diffusion Systems on Evolving Domains. SIAM Journal on Numerical Analysis 51(4):2309–2330

Li J, Zeng W, Zhang Y, Ko AMS, Li C, Zhu H, Fu Q, Zhou H (2017) Ancient dna reveals genetic connections between early di-qiang and han chinese. BMC evolutionary biology 17(1):239

Lima E, Ghousifam N, Ozkan A, Oden J, Shahmoradi A, Rylander M, Wohlmuth B, Yankeelov T (2018) Calibration of multi-parameter models of avascular tumor growth using time resolved microscopy data. Scientific reports 8(1):14,558

Lin X, Terejanu G, Shrestha S, Banerjee S, Chanda A (2016) Bayesian model selection framework for identifying growth patterns in filamentous fungi. Journal of theoretical biology 398:85–95

López-Fagundo C, Bar-Kochba E, Livi LL, Hoffman-Kim D, Franck C (2014) Three-dimensional traction forces of schwann cells on compliant substrates. Journal of The Royal Society Interface 11(97):20140,247

Lu F, Morzfeld M, Tu X, Chorin AJ (2015) Limitations of polynomial chaos expansions in the bayesian solution of inverse problems. Journal of Computational Physics 282:138–147

Lu Y, Stuart A, Weber H (2017) Gaussian approximations for probability measures on rˆd. SIAM/ASA Journal on Uncertainty Quantification 5(1):1136–1165

Ma J, Plonka G (2010) The curvelet transform. IEEE signal processing magazine 27(2):118–133

Ma Z, Leijon A (2011) Bayesian estimation of beta mixture models with variational inference. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(11):2160–2173

Ma Z, Teschendorff AE, Leijon A, Qiao Y, Zhang H, Guo J (2015) Variational bayesian matrix factorization for bounded support data. IEEE transactions on pattern analysis and machine intelligence 37(4):876–889

Mackenzie J, Madzvamuse A (2011) Analysis of stability and convergence of finite-difference methods for a reaction–diffusion problem on a one-dimensional growing domain. IMA journal of numerical analysis 31(1):212–232

Madzvamuse A (2006) Time-stepping schemes for moving grid finite elements applied to reaction–diffusion systems on fixed and growing domains. Journal of Computational Physics 214(1):239–263, DOI 10.1016/j.jcp.2005.09.012

Madzvamuse A, Maini PK (2007) Velocity-induced numerical solutions of reaction-diffusion systems on continuously growing domains. Journal of computational physics 225(1):100–119

Madzvamuse A, Maini P, Wathen A (2005) A Moving Grid Finite Element Method for the Simulation of Pattern Generation by Turing Models on Growing Domains. Journal of Scientific Computing 24(2):247–262, DOI 10.1007/s10915-004-4617-7

Madzvamuse A, Gaffney EA, Maini PK (2010) Stability analysis of non-autonomous reaction-diffusion systems: The effects of growing domains. Journal of mathematical biology 61(1):133–164

Madzvamuse A, Ndakwo HS, Barreira R (2016) Stability Analysis of Reaction-Diffusion Models on Evolving Domains: The effects of Cross-Diffusion. Dynamical Systems 36(4):2133–2170

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculations by fast computing machines. The journal of chemical physics 21(6):1087–1092

Mitov V, Stadler T (2017) Fast and robust inference of phylogenetic ornstein-uhlenbeck models using parallel likelihood calculation. bioRxiv p 115089

Mitov V, Stadler T (2018) Parallel likelihood calculation for phylogenetic comparative models: the splitt c++ library. bioRxiv p 235739

Moch M, Herberich G, Aach T, Leube RE, Windoffer R (2013) Measuring the regulation of keratin filament network dynamics. Proceedings of the National Academy of Sciences 110(26):10,664–10,669

Möhl C, Kirchgessner N, Schäfer C, Hoffmann B, Merkel R (2012) Quantitative mapping of averaged focal adhesion dynamics in migrating cells by shape normalization. J Cell Sci 125(1):155–165

Munevar S, Wang Yl, Dembo M (2001) Traction force microscopy of migrating normal and h-ras transformed 3t3 fibroblasts. Biophysical journal 80(4):1744–1757

Murphy KP (2012) Machine learning: a probabilistic perspective. MIT press

Murray JD (2011) Mathematical Biology: I. An Introduction. Springer Science & Business Media

Murray JD (2013) Mathematical Biology II: Spatial Models and Biomedical Applications. Springer New York

Murray LM, Lee A, Jacob PE (2016) Parallel resampling in the particle filter. Journal of Computational and Graphical Statistics 25(3):789–805

Norris JR (1998) Markov Chains. Cambridge University Press, Cambridge

Oden JT, Hawkins A, Prudhomme S (2010) General diffuse-interface theories and an approach to predictive tumor growth modeling. Mathematical Models and Methods in Applied Sciences 20(03):477–517

Olbricht W, Chatterjee ND, Miller K (1994) Bayes estimation: A novel approach to derivation of internally consistent thermodynamic data for minerals, their uncertainties, and correlations. part i: Theory. Physics and Chemistry of Minerals 21(1-2):36–49

Omary MB, Ku NO, Strnad P, Hanada S (2009) Toward unraveling the complexity of simple epithelial keratins in human disease. The Journal of clinical investigation 119(7):1794–1805

Owen NE (2017) A comparison of polynomial chaos and gaussian process emulation for uncertainty quantification in computer experiments

Pasqualini FS, Agarwal A, O'Connor BB, Liu Q, Sheehy SP, Parker KK (2018) Traction force microscopy of engineered cardiac tissues. PloS one 13(3):e0194,706

Perdikaris P, Karniadakis GE (2016) Model inversion via multi-fidelity Bayesian optimization: A new paradigm for parameter estimation in haemodynamics, and beyond. Journal of The Royal Society Interface 13(118):20151,107

Peschetola V, Laurent VM, Duperray A, Michel R, Ambrosi D, Preziosi L, Verdier C (2013) Time-dependent traction force microscopy for cancer cells as a measure of invasiveness. Cytoskeleton 70(4):201–214, DOI 10.1002/cm.21100

Portet S, Arino J (2009) An in vivo intermediate filament assembly model. Mathematical Biosciences and Engineering 6(1):117–134

Portet S, Madzvamuse A, Chung A, Leube RE, Windoffer R (2015) Keratin Dynamics: Modeling the Interplay between Turnover and Transport. PLoS ONE 10(3), DOI 10.1371/journal.pone.0121090

Prigogine I, Lefever R (1968) Symmetry breaking instabilities in dissipative systems. II. The Journal of Chemical Physics 48(4):1695–1700

Raftery AE, Lewis SM (1995) The Number of Iterations, Convergence Diagnostics and Generic Metropolis Algorithms. In: In Practical Markov Chain Monte Carlo (W.R. Gilks, D.J. Spiegelhalter and, Chapman and Hall, pp 115–130

Rasmussen CE (2004) Gaussian processes in machine learning. In: Advanced lectures on machine learning, Springer, pp 63–71

Ritchie DM, Kernighan BW, Lesk ME (1988) The C programming language. Prentice Hall Englewood Cliffs

Robert A, Herrmann H, Davidson MW, Gelfand VI (2014) Microtubule-dependent transport of vimentin filament precursors is regulated by actin and by the concerted action of rho-and p21-activated kinases. The FASEB Journal 28(7):2879–2890

Ross RJ, Baker RE, Parker A, Ford M, Mort R, Yates C (2017) Using approximate bayesian computation to quantify cell–cell adhesion parameters in a cell migratory process. NPJ systems biology and applications 3(1):9

Ruuth S (1995) Implicit-explicit methods for reaction-diffusion problems in pattern formation. Journal of Mathematical Biology 34(2):148–176, DOI 10.1007/BF00178771

Sanz-Solé M (1999) Probabilitats, vol 28. Edicions Universitat Barcelona

Schmitt LM (2001) Theory of genetic algorithms. Theoretical Computer Science 259(1-2):1–61

Schmitt LM (2004) Theory of genetic algorithms ii: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. Theoretical Computer Science 310(1-3):181–231

Schnakenberg J (1979) Simple chemical reaction systems with limit cycle behaviour. Journal of Theoretical Biology 81(3):389–400, DOI 10.1016/0022-5193(79)90042-0

Schwab C, Stuart AM (2012) Sparse deterministic approximation of bayesian inverse problems. Inverse Problems 28(4):045,003

Schwarz US, Soiné JR (2015) Traction force microscopy on soft elastic substrates: A guide to recent computational advances. Biochimica et Biophysica Acta (BBA)-Molecular Cell Research 1853(11):3095–3104

Sgouralis I, Nebenfuhr A, Maroulas V (2017) A bayesian topological framework for the identification and reconstruction of subcellular motion. SIAM Journal on Imaging Sciences 10(2):871–899

SIAM (2001) Graduate Education in Computational Science and Engineering. SIAM Review pp 163–177

Skeel RD, Berzins M (1990) A method for the spatial discretization of parabolic equations in one space variable. SIAM journal on scientific and statistical computing 11(1):1–32

Smith RL, Gröhn YT (2015) Use of approximate bayesian computation to assess and fit models of mycobacterium leprae to predict outcomes of the brazilian control program. PloS one 10(6):e0129,535

Smoller J (1982) Shock Waves and Reaction-Diffusion Equations. Springer, New York

Soize C (2017) Uncertainty Quantification. Springer

Stoll M, Pearson JW, Maini PK (2016) Fast solvers for optimal control problems from pattern formation. Journal of Computational Physics 304:27–45

Stuart AM (2010) Inverse problems: A Bayesian perspective. Acta Numerica 19:451–559, DOI 10.1017/S0962492910000061

Style RW, Boltyanskiy R, German GK, Hyland C, MacMinn CW, Mertz AF, Wilen LA, Xu Y, Dufresne ER (2014) Traction force microscopy in physics and biology. Soft matter 10(23):4047–4055

Sun C, Leube R, Windoffer R, Portet S (2013) A mathematical model for the keratin cycle of assembly and disassembly. The IMA Journal of Applied Mathematics 80(1):100–114

Sun C, Arino J, Portet S (2017) Intermediate filament dynamics: Disassembly regulation. International Journal of Biomathematics 10(01):1750,015

Sutton JE, Guo W, Katsoulakis MA, Vlachos DG (2016) Effects of correlated parameters and uncertainty in electronic-structure-based chemical kinetic modelling. Nat Chem advance online publication

Tarantola A (2005) Inverse Problem Theory and Methods for Model Parameter Estimation. Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics

Tierney L (1998) A note on metropolis-hastings kernels for general state spaces. Annals of applied probability pp 1–9

Tjelmeland H (2004) Using all Metropolis–Hastings proposals to estimate mean values. Statistics (4)

Tombolato L, Novitskaya EE, Chen PY, Sheppard FA, McKittrick J (2010) Microstructure, elastic properties and deformation mechanisms of horn keratin. Acta biomaterialia 6(2):319–330

Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP (2009) Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. Journal of The Royal Society Interface 6(31):187–202, DOI 10.1098/rsif.2008.0172

Trillos NG, Sanz-Alonso D (2017) The bayesian formulation and well-posedness of fractional elliptic inverse problems. Inverse Problems 33(6):065,006

Tröltzsch F (2010) Optimal control of partial differential equations: theory, methods, and applications, vol 112. American Mathematical Soc.

Turing AM (1952) The Chemical Basis of Morphogenesis. Philosophical Transactions of the Royal Society of London B: Biological Sciences 237(641):37–72, DOI 10.1098/rstb.1952.0012

Uzunca M, Küçükseyhan T, Yücel H, Karasözen B (2017) Optimal control of convective fitzhugh–nagumo equation. Computers & Mathematics with Applications 73(9):2151–2169

Vedula SRK, Hirata H, Nai MH, Toyama Y, Trepat X, Lim CT, Ladoux B, et al (2014) Epithelial bridges maintain tissue integrity during collective cell migration. Nature materials 13(1):87

Venkataraman C, Lakkis O, Madzvamuse A (2012) Global existence for semilinear reaction–diffusion systems on evolving domains. Journal of mathematical biology 64(1-2):41–67

Venkataraman C, Lakkis O, Madzvamuse A (2013) Adaptive finite elements for semilinear reaction-diffusion systems on growing domains. In: Numerical Mathematics and Advanced Applications 2011, Springer, pp 71–80

Vigil RD, Ouyang Q, Swinney HL (1992) Turing patterns in a simple gel reactor. Physica A: Statistical Mechanics and its Applications 188(1):17–25

Vitale G, Preziosi L, Ambrosi D (2012) Force traction microscopy: an inverse problem with pointwise observations. Journal of Mathematical Analysis and Applications 395(2):788–801

Vo BN, Drovandi CC, Pettitt AN, Simpson MJ (2015) Quantifying uncertainty in parameter estimates for stochastic models of collective cell spreading using approximate bayesian computation. Mathematical biosciences 263:133–142

Vyshemirsky V, Girolami MA (2008) Bayesian ranking of biochemical system models. Bioinformatics 24(6):833–839, DOI 10.1093/bioinformatics/btm607

Wang B, Yang W, McKittrick J, Meyers MA (2016a) Keratin: Structure, mechanical properties, occurrence in biological organisms, and efforts at bioinspiration. Progress in Materials Science 76:229–318

Wang J, Zhang M, Li M, Wang Y, Liu D (2016b) On the control of the microresonator optical frequency comb in turing pattern regime via parametric seeding. In: OptoElectronics and Communications Conference (OECC) Held Jointly with 2016 International Conference on Photonics in Switching (PS), 2016 21st, IEEE, pp 1–3

Wang JH, Lin JS (2007) Cell traction force and measurement methods. Biomechanics and modeling in mechanobiology 6(6):361

Wegmann D, Leuenberger C, Excoffier L (2009) Efficient approximate bayesian computation coupled with markov chain monte carlo without likelihood. Genetics

Williamson J (2010) Review of B. de finetti, philosophical lectures on probability. Philosophia Mathematica 18(1):130–135

Windoffer R, Leube RE (1999) Detection of cytokeratin dynamics by time-lapse fluorescence microscopy in living cells. J Cell Sci 112(24):4521–4534

Windoffer R, Woll S, Strnad P, Leube RE (2004) Identification of novel principles of keratin filament network turnover in living cells. Molecular biology of the cell 15(5):2436–2448

Windoffer R, Beil M, Magin TM, Leube RE (2011) Cytoskeleton in motion: the dynamics of keratin intermediate filaments in epithelia. The Journal of cell biology 194(5):669–678

Wu Q, Smith-Miles K, Tian T (2014) Approximate bayesian computation schemes for parameter inference of discrete stochastic models using simulated likelihood density. BMC bioinformatics 15(12):S3

Xun X, Cao J, Mallick B, Maity A, Carroll RJ (2013) Parameter estimation of partial differential equation models. Journal of the American Statistical Association 108(503):1009–1020

Yang F, Venkataraman C, Styles V, Madzvamuse A (2015) A parallel and adaptive multigrid solver for the solutions of the optimal control of geometric evolution laws in two and three dimensions. In: 4th International Conference on Computational and Mathematical Biomedical Engineering - CMBE2015, pp 1–4

Yoon KH, Yoon M, Moir RD, Khuon S, Flitney FW, Goldman RD (2001) Insights into the dynamic properties of keratin intermediate filaments in living epithelial cells. The Journal of cell biology 153(3):503–516