Computational design and characterisation of synthetic genetic switches

Miriam Leon

A thesis submitted in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy of
University College London

2016

Primary supervisor:
Dr. Chris P Barnes

Secondary supervisor:
Prof. Geraint MH Thomas

I, Miriam Leon, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Contents

List of Figures

List of Tables

Abbreviations

ABC Approximate Bayesian Computation.

ATc anhydrotetracycline.

BSA Bovine Serum Albumin.

CS-LU Lu classic switch.

CS-MA Mass action classic switch.

DNA Deoxyribonucleic acid.

dNTPs Deoxynucleotide.

DP-LU Lu double positive switch.

DP-MA Mass action double positive switch.

GFP green fluorescent protein.

GPU Graphical Processing Units.

IPTG Isopropyl-beta-D-thiogalactopyranoside.

LB Lysogeny broth.

MCMC Markov Chain Monte Carlo.

MJP Markov jump process.

ODE Ordinary differential equation.

PCA Principal component analysis.

PCR Polymerase Chain Reaction.

QSSA quasi-steady state approximation.

SDE Stochastic differential equation.

SMC Sequential Monte Carlo.

SP-LU Lu single positive switch.

Acknowledgements

1 Introduction

1.1 Contents of this thesis

2 Background

2.1 Introduction to synthetic biology

Synthetic biology aims at the rational design and construction of biological parts, devices, and systems in order to engineer organisms to perform new tasks (Lu, Khalil, & Collins 2009; Andrianantoandro et al. 2014). A part is a basic unit, like a promoter or a ribosome binding site that when combined with other parts will make a functional unit, a device (Heinemann & Panke 2006). A device processes inputs, performs functions and produces outputs (Andrianantoandro et al. 2014). A system comprises of a collection of devices.

Emphasis is put on the use of engineering principles such as modularity, standardisation, use of predictive models and the separation of design and construction (Agapakis & Silver 2009; Heinemann & Panke 2006). A hierarchy similar to computer science is used, with cells, pathways and biochemical reactions acting as computers, modules and gates respectively (Andrianantoandro et al. 2014).

Numerous applications of synthetic biology have emerged, from altering existing metabolisms to producing synthetic drugs (Holtz & Keasling 2010) or creating new synthetic life forms (Agapakis & Silver 2009). Despite the successes there is still a lack of predictive power due to the stochasticity and lack of complete knowledge of the cellular environment (Andrianantoandro et al. 2014).

Synthetic biology is now entering an age where simple synthetic circuits have been built, such as toggle switches (Gardner, Cantor, & Collins 2000; Kramer et al. 2004; Isaacs et al. 2003; Ham et al. 2008; Deans, Cantor, & Collins 2007; Friedland et al. 2009), oscillators (Stricker et al. 2008; Fung et al. 2005; Tigges et al. 2009) and pulse generators (Basu et al. 2004), but larger circuits have proven more difficult (XXX). The leap from building low-level circuits to assembling them into complex networks has yet to be made successfully (Lu, Khalil, & Collins 2009), and predictable circuit behaviour remains challenging (XXX). Efforts to do so are plagued by intra-circuit

crosstalk and incompatibility, as well as cellular noise, which can render synthetic networks non-functional *in vivo* (XXX).

2.2 System design in synthetic biology

Creating synthetic devices that are robust to changing cellular contexts will be key to the success of synthetic biology. Unknown initial conditions and parameter values as well as the variability of the cellular environment, extracellular noise and crosstalk makes the majority of synthetic genetic devices non-functional (Chen, Chang, & Lee 2009). Designing devices robust to this environment will lead to reliable behaviour of the systems. When faced with a set of competing designs for a given genetic circuit, one is likely to choose the simplest possible model that can achieve the desired behaviour. However, simple systems are often the least robust. Feedback loops are well known key regulatory motifs (Brandman et al. 2005). Negative feedback loops are essential for homeostasis and buffering (Thomas, Thieffry, & Kaufman 1995) thus increasing robustness to extrinsic noise sources and positive feedback loops can generate multistationarity in a system (Thomas, Thieffry, & Kaufman 1995). Incorporating this kind of additional feedback interactions can make a design more robust and reliable. Maximising production is an important goal for a metabolic engineering project if it is to produce an economically viable substance (Holtz & Keasling 2010). Network topologies and parameter values of different toggle switch designs are explored here in order to identify the design that maximises robustness and distance between steady states. This ensures the reliable production of the product with the greatest distance between the on and off states of the switch. In the future, by selecting the system components accordingly, the parameter values can be adjusted *in vivo*. For example, the parameter value corresponding to the translation initiation rate can be chosen by selecting the appropriate RBS sequence which given a nucleotide sequence will produce the desired rate (Holtz & Keasling 2010), a method developed by Salis, Mirsky, & Voigt (2009). Another method to tweak the parameter values *in vivo* is to select the promoter to have the strength corresponding to the levels of gene expression and repression desired. Activity of each promoter can be measured and standardised (Kelly et al. 2009) making this process possible. For a system requiring more than one promoter, these can be efficiently selected from a promoter library using a genetic algorithm created by Wu, Lee, & Chen (2011). These standardised interchangeable components with known sequence and activity are what synthetic biology classes

as BioBricks (Kelly et al. 2009; Canton, Labno, & Endy 2008). These can be selected and used to construct a desired system and replicate the parameter values found in the scan presented here.

The first computational approach for the tuning of robust synthetic networks was that of Batt et al. (2007) where they examined the problem of finding a subset of the parameter set for which a given property was satisfied for all the parameters. Chen, Chang, & Lee (2009) used the fuzzy dynamic game method to solve the minimax regulation design problem of synthetic genetic networks. In that method the worst case effect of all disturbances is minimised for a given network. An evolutionary algorithm has also been used to solve the robust design problem by evolving the parameters of the system in order to make it more robust to cellular disturbances by **Chen:2011hj** The added value of the methodology presented here is that the network structure in addition to the network parameters are adjusted to select a network that can robustly create the desired behaviour.

2.3 Introduction to Biochemical Modelling

2.3.1 Graphical representation of biochemical systems

It is common to represent coupled biochemical reactions graphically. In a graph, as shown in Figure **??**, nodes represent the species and the edges represent an interaction between the species it connects, in which a transcription factor directly affects the transcription of a gene (Alon 2007). An arrow at the end of an arc represents activation, i.e. that when the transcription factor binds to the promoter the rate of transcription of the gene increases. A flat line perpendicular to the arc at the end of an arc represents repression, i.e. that when the transcription factor binds to the promoter the rate of transcription of the gene decreases (Alon 2007).

2.3.2 Deterministic and Stochastic modelling

Modelling attempts to describe the elements and dynamics of the biochemical system of interest. It is a tool used for integrating knowledge and experimental data as well as for making predictions about the behaviour of the system (Wilkinson 2006). When modelling a biochemical system it is generally assumed that the rates of a reaction are directly proportional to the concentration of the reactants, raised to the power of their stoichiometry (Wilkinson 2006). This is known as mass-action kinetics and is used in this work to model the various systems. There are two main ways

of modelling a system, deterministically and stochastically. Deterministic modelling utilises Ordinary differential equation (ODE) and models the concentrations of the species (proteins or other molecules) by time-dependent variables (de Jong 2002). Rate equations are used to model gene regulation where the rate of production of a species is a function of the concentrations of the other species (de Jong 2002). When modelling deterministically the model is viewed as a system which, with sufficient knowledge of the system, its behaviour is entirely predictable. Nevertheless we are still a long way away from having complete knowledge of a system of interesting size (Wilkinson 2006). Deterministic modelling also assumes a homogenous mixture where species concentrations vary continuously and deterministically, assumptions that often are not met *in vivo*. A cell is spatially and temporally separated, due to small molecule numbers and fluctuations in the timing of processes (de Jong 2002).

In stochastic modelling, species are measured in discrete amounts rather than concentrations and a joint probability distribution is used to express the probability that at time t the cell contains a number of molecules of each species (de Jong 2002). It takes uncertainty into account and does not assume a homogenous mix. It is thus often more appropriate for modelling cellular systems, although more computationally intensive. In stochastic systems the Gillespie algorithm is widely used to simulate the time-evolution of the state of the system (Wilkinson 2006). The algorithm, developed by Gillespie (1977) can be summarised in four steps:

1. Number of molecules in the system initialised
2. Two random numbers generated, one to determine which reaction will occur next and one to determine the time step
3. Time step increased and molecule counts updated according to Step 2
4. Repeat from Step 2 until total simulation time reached

2.3.3 Steady state and stability

In a steady state, the state of a system remains fixed. In non-linear systems, like the ones systems biology deals with, there is generally not an analytical solution thus the system has to be solved numerically. A stable steady state is defined as a fixed point whose nearby points approach the fixed point (Kaplan & Glass 1995). This means that after a small perturbation the system will quickly return to the steady

state. An unstable steady state is one which if the system is perturbed slightly then it moves away from the steady state (Konopka 2007).

2.4 The genetic toggle switch

One of the most common devices used in synthetic biology is the genetic toggle switch. A toggle switch consists of a set of transcription factors that mutually repress each other (Gardner, Cantor, & Collins 2000). Genetic switches play a major role in binary cell fate decisions like stem cell differentiation, as they are capable of exhibiting bistable behaviour. Bistability of a system is defined by the existence of two distinct phenotypic states but no intermediate state. Bistability is a property that is important in nature and a valuable resource to tap into in synthetic biology. It allows cells to alter their response to environmental cues and increases the overall population fitness by 'hedge-betting' the response of the population (XXX).

2.4.1 Importance in natural systems

In developmental processes, bistability ensures that the differentiating cell will follow one pathway, or the other, with no possible intermediate phenotypes. This is vital for the correct development of a cell in a specific pathway. One example is the trophectoderm differentiation pathway, in which a mutually inhibitory toggle switch exists between Oct3/4 and Cdx2. This determines whether an Embryonic Stem cell will differentiate into a Trophectoderm cell, if Cdx2 dominates the system, or an Inner Cell Mass cell if Oct3/4 dominates (Niwa et al. 2005). Bistability is critical in this system as a cell must differentiate into either a trophectoderm cell or an inner cell mass cell, thus the signal to do so must be straightforward. In the case of the GATA1 and PU.1 toggle switch, the transcription factor pair controls the fate of the common myeloid progenitors, and the two possible differentiation paths are erythroid and myeloid blood cells (Chickarmane, Enver, & Peterson 2009). The double-negative feedback loop created by the mutually repressive pair of transcription factors sustains the system in balance until an external stimulus causes one of the two transcription factors to increase in concentration. The increased concentration of one transcription factor causes the increased repression of the production of the antagonistic transcription factor, tipping the balance towards the dominance of the first transcription factor. The double negative feedback loop reinforces this dynamic and the system remains in the same state, until an external stimulus disturbs it (Ferrell 2002).

2.4.2 Uses in synthetic biology

Despite their simplicity, toggle switches can be powerful building blocks with which to create complex responses in a synthetic network. They can be used in isolation or in tandem to create complex networks and signalling cascades. The toggle switch has been used for the regulation of mammalian gene expression (Deans, Cantor, & Collins 2007; Kramer et al. 2004). Other synthetic applications of the toggle switch include the construction of a synthetic genetic clock (Atkinson et al. 2003), of a predictable genetic timer (Ellis, Wang, & Collins 2009), and the formation of biofilms in response to engineered stimuli (Kobayashi et al. 2004). These applications are modifications of the classical toggle switch (Gardner, Cantor, & Collins 2000), and to our knowledge no application made of a cascade or collection of the switch has been successful. This would make more complex applications possible and could be used to solve real-life problems. For example, an analog-to-digital converter to translate external stimuli like the concentration of an inducer into an internal digital response, or programmable bacteria to move from point to point up different chemical gradients (Lu, Khalil, & Collins 2009). For a review on current circuits see (Khalil & Collins 2010) and for possible future applications see (Lu, Khalil, & Collins 2009). This leap will be difficult to achieve before first being able to build robust and well characterised individual switches.

2.4.3 Modelling the genetic toggle switch

The toggle switch motif has been studied extensively and there are numerous studies based on a number of different methods of modelling and analysis of the dynamics, including both deterministic and stochastic approaches. Deterministic modelling utilises ordinary differential equations (ODE) and models the concentrations of the species (proteins or other molecules) by time-dependent variables (de Jong 2002). When modelling deterministically the model is viewed as a system whose behaviour is entirely predictable, given sufficient knowledge. In stochastic modelling, species are measured in discrete amounts rather than concentrations and a joint probability distribution is used to express the probability that at time t the cell contains a number of molecules of each species (**Wilkinson:2006**; de Jong 2002). It takes uncertainty into account and is thus often more appropriate for modelling cellular systems, although more computationally expensive. In stochastic systems the Gillespie algorithm is widely used to simulate the time-evolution of the state of the system (Warren & ten Wolde 2005).

The conclusions drawn about the stability and robustness of the toggle switch also vary between the different modelling approaches. Numerous studies have concluded that cooperativity is a necessary condition for bistability to arise (Gardner, Cantor, & Collins 2000; Walczak, Onuchic, & Wolynes 2005; Warren & ten Wolde 2004; Warren & ten Wolde 2005; Cherry & Adler 2000). However, Lipshtat et al. (2006) found that stochastic effects can give rise to bistability even without cooperativity in three kinds of switch; the exclusive switch, in which there can only be one repressor bound at any one time, a switch in which there is degradation of bound repressors, and the switch in which free repressor proteins can form a complex, which renders them inactive as transcription factors (Lipshtat et al. 2006). In another study, Ma et al. (2012) found that the stochastic fluctuations in a system involving such a small number of molecules, like the toggle switch, uncovers effects that can not be predicted by the fully deterministic case (Ma et al. 2012). In their system, the toggle switch was found to be tristable, as small number effects render the third unstable steady state stable. Biancalani & Assaf (2015) identified multiplicative noise as the source of bistability in the stochastic case (Biancalani & Assaf 2015). Warren & ten Wolde (2005) concluded that the exclusive switch is always more robust than the general switch, since the free energy barrier is higher (Warren & ten Wolde 2005). A summary of the toggle switch models is shown in Table 2.1. As is clear from above, there is yet to exist a consensus on the stability a switch is capable of, and the most appropriate method of modelling it. Different methods arrive at different conclusions, creating confusion on which behaviour to be expected by the experimentalist for even a simple system like the toggle switch, consisting of just two genes. The toggle switch cannot be used as a building block of larger, more complex systems until its behaviour can be predicted accurately. Until then, designing systems with predictable behaviour will be near impossible.

Table 2.1 Summary of stability for the toggle switch found via different modelling approaches

		Simple			Double positive autoregulation		
	Stability	*Reference*	*Notes*	*Stability*	*Reference*	*Notes*
Deterministic	Monostable	(Loinger et al. 2007)	no cooperativity, exclusive & general copperativity >2, bound repressor degradation	Bistable	(Guantes & Poyatos 2008)	
	Bistable	(Gardner, Cantor, & Collins 2000) (Loinger et al. 2007)		Tristable 4 steady steates	(Guantes & Poyatos 2008) (Guantes & Poyatos 2008)	
Stochastic	Monostable	(Loinger et al. 2007) (Lu, Onuchic, & Ben-Jacob 2014)	no cooperativity, weak repression	Tristable	(Lu, Onuchic, & Ben-Jacob 2014)	
	Bistable	(Biancalani & Assaf 2015)	exclusive, controlled by noise strength			
		(Lipshtat et al. 2006)	no cooperativity			
		(Loinger et al. 2007)	no cooperativity, exclusive & bound repression degradation			
	Tristable	(Loinger et al. 2007) (Ma et al. 2012)	no cooperativity, strong repression			

2.5 Introduction to Bayesian statistics

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta \frac{p(x|\theta)p(\theta)}{p(x)}}$$

because

$$p(x)p(\theta|x) = p(\theta)p(x|\theta)$$

where $p(x|\theta)$ is the likelihood, $p(\theta)$ is the prior, and $\int p(x|\theta)p(\theta)d\theta$ is the evidence. This is the normalisation.

Bayes factor:

$$B_{12} = \frac{\int p(x|\theta, M_1)p(\theta, M_1)d\theta}{\int p(x|\theta, M_2)p(\theta, M_2)d\theta}$$

In our case, O is the objective, and D is the design. Therefore:

$$p(O|D_1) = \int p(O|\theta, D_1)p(\theta|D_1)d\theta,$$

This is the robustness, or evidence or marginal likelihood

$$p(O|D_1) = \int p(O|\theta, D_1)p(\theta|D_1)d\theta,$$
$$p(O|D_1) = \iiint_{\underline{\Theta}} p(O|\underline{\Theta})p(\underline{\Theta}|D_1)d\underline{\Theta}$$

where $\underline{\Theta} = \{\theta_1, \theta_2, \theta_3\}$
Assuming the prior is uniform, and $a = 0$:

$$p(O|D_1) = \iiint_{\underline{\Theta}} p(O|\underline{\Theta})\frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}d\underline{\Theta}$$
$$p(O|D_1) = \frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}\iiint_{\underline{\Theta}} p(O|\underline{\Theta})d\underline{\Theta}$$

Assuming uniform likelihood:

$$p(O|D_1) = \frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}\iiint_{\underline{\Theta}_F} 1d\theta_1\theta_2\theta_3 + \frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}\iiint_{\underline{\Theta}_F} Od\underline{\Theta}$$

2.5.1 Bayes' theorem

2.5.2 Bayesian inference

2.5.3 Model checking

2.5.4 Prior selection

A circuit must be robust to a fluctuating cellular environment and its response and sensitivity must be able to be fine tuned in order to orchestrate a network of circuits that function together. A robust circuit can tolerate the compound stochasticity that a chain of circuits brings, and fine tuning of its response and sensitivity enables the researcher to make it sensitive to an upstream signal as well as influence a downstream subsystem. Parts can be fine tuned by developing component libraries (Lu, Khalil, & Collins 2009), but this will be of little use if the required parameter ranges for parts to make a functional complex network are unknown, and will only perpetuate the cycles of trial-and-error. A computational method to find the range of parameter values that will produce the behaviour of choice is crucial to the design process by enabling the informed selection of appropriate parts from the libraries. For example, if it is known that gene expression must be low for a given stability, one can select a weak promoter or a low copy plasmid for the desired construct.

Both analytical and computational approaches have been deployed for the study of the toggle switch. Analytical approaches are limited to simpler models and thus require a number of assumptions to be made. The system under consideration has to be reduced to very few equations and parameters in order to make the system solvable. This requires assumptions to be made about the system that cannot always be justified, such as the quasi-steady state approximation (QSSA). The QSSA assumes that the binding/unbinding processes are much faster than any other process (Loinger et al. 2007), thus the bound intermediate is assumed to always be in steady state. The QSSA assumption is met *in vitro* but often does not hold *in vivo* and its misuse can lead to large errors and incorrectly estimated parameters (Pedersen, Bersani, & Bersani 2007). Moreover, it is generally not possible to solve even simple stochastic models analytically, and these methods are restricted to deterministic models. The computational and graph-theoretic approaches developed for the study of multistationarity generally focus on deciding on whether a given system is incapable of producing multiple steady states (Conradi et al. 2007; Banaji & Craciun 2010; Feliu & Wiuf 2013). For example, Feliu & Wiuf (2013) developed an approach using chemical reaction theory and generalised mass action modelling

(Feliu & Wiuf 2013). No approach exists that can handle both deterministic and stochastic systems in an integrated manner.

For this purpose, I developed a computational framework based on sequential Monte Carlo that takes a model and determines whether it is capable of producing a given number of (stable) steady states and the parameter space that gives rise to the behaviour. Uniquely, this can be done for both deterministic and stochastic models, and also complex models with many parameters, thus removing the need for simplifying assumptions. This framework can be used for comparing the conclusions drawn by various modelling approaches and thus provides a way to investigate appropriate abstractions. I have made this framework into a python package, called Stability Finder.

I use this methodology to investigate genetic toggle switches and uncover the design principles behind making a bistable switch, as well as those necessary to make a tristable and a quadristable switch (4 steady states). I also demonstrate the ability of Stability Finder to examine more complex systems and examine the design principles of a three gene switch. The examples I used demonstrate that Stability Finder will be a valuable tool in the future design and construction of novel gene networks.

2.5.5 Model parametric Robustness

During this thesis I define robustness as the ability of a system to retain its function despite parameter perturbations (Stelling et al. 2004). The robustness of biological systems has been studied extensively (Barkai & Leibler 1997; Stelling et al. 2004; Prill, Iglesias, & Levchenko 2005; Kim et al. 2006; Kitano 2007; Hafner et al. 2009; Shinar & Feinberg 2010; Zamora-Sillero et al. 2011; Woods et al. 2015). and it is well known that feedback loops can increase the robustness of a system (**Doyle:2005ul**; Becskei & Serrano 2000).

The robustness of a model can be calculated by dividing the volume of its functional region by the volume of its priors. This is a measure of the volume of the posterior distribution is compared to the priors. It comes from Bayes' rule that:

$$f(\theta|x) = \frac{f(\theta)f(x|\theta)}{\int p(x|\theta)p(\theta)d\theta} \tag{2.1}$$

where $p(x|\theta)$ is the likelihood, $p(\theta)$ is the prior, and $\int p(x|\theta)p(\theta)d\theta$ is the evidence. The evidence is the normalisation added so that the distribution integrates to 1. For

a given model design D and objective O we define the functional region F as the region within the prior where O is satisfied. So within the prior we can assign 1 to any region that falls within F and 0 to any region outside that.

$$p(O|D_1) = \int p(O|\theta, D_1)p(\theta|D_1)d\theta, \tag{2.2}$$

For a design with three parameters this becomes:

$$p(O|D_1) = \iiint_{\underline{\Theta}} p(O|\underline{\Theta})p(\underline{\Theta}|D_1)d\underline{\Theta}, \tag{2.3}$$

where $\underline{\Theta}$ is a vector containing the three parameters = $\theta_1, \theta_2, \theta_3$. To calculate the robustness, or model evidence, we integrate this with respect to $\underline{\Theta}$. We assume all parameters $\theta_1, \theta_2, \theta_3$ are uniform, $p(\underline{\Theta}|D_1) \sim U(a, b)$. If we assume a = 0 this integral becomes:

$$p(O|D_1) = \iiint_{\underline{\Theta}} p(O|\underline{\Theta})\frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}d\underline{\Theta} \text{ , and} \tag{2.4}$$

$$p(O|D_1) = \frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}\iiint_{\underline{\Theta}} p(O|\underline{\Theta})d\underline{\Theta} \tag{2.5}$$

since $\frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}$ is a constant. Then assuming that the likelihood is uniform Equation2.5 becomes:

$$p(O|D_1) = \frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}\Big[\iiint_{\underline{\Theta}_F} 1d\underline{\Theta} + \cancelto{0}{\iiint_{\underline{\Theta}_{\not F}} 0d\underline{\Theta}}\Big] \tag{2.6}$$

$$\tag{2.7}$$

since we assign 1 to any region within F and 0 to any region outside it. This becomes:

$$p(O|D_1) = \overbrace{\frac{1}{b_1}\frac{1}{b_2}\frac{1}{b_3}}^{|P|}\underbrace{\iiint_{\underline{\Theta}_F} 1d\underline{\Theta}}_{|F|}, \tag{2.8}$$

$$\therefore p(O|D_1) = \frac{|F|}{|P|}, \tag{2.9}$$

where |P| is the volume of the prior P and |F| the volume of the functional region F. Therefore, in the case where both the prior and the likelihood are uniform, the robustness R of the design is the ratio of the volumes of the two.

If on the other hand we assume the likelihood is multivariate normal, with priors remaining uniform, Equation2.5 becomes:

$$p(O|D_1) = \frac{1}{|P|} \iiint_{\underline{\Theta}} f(\underline{\Theta}; \mu, \Sigma) d\underline{\Theta} \tag{2.10}$$

$$\therefore p(O|D_1) = \frac{1}{|P|} \underbrace{\times (2\pi)^{\frac{k}{2}} \times |\Sigma|^{\frac{1}{2}}}_{|F|} \tag{2.11}$$

$$\therefore p(O|D_1) = \frac{|F|}{|P|}, \tag{2.12}$$

We can use the Bayes' factor in order to compare the robustness between two model designs. The Bayes' factor is defined as follows:

$$B_{ab} = \frac{\int p(x|\theta, D_a) p(\theta, D_a) d\theta}{\int p(x|\theta, D_b) p(\theta, D_b) d\theta} \tag{2.13}$$

$$\therefore B_{ab} = \frac{|Fa|}{|Pa|} / \frac{|Fb|}{|Pb|} \tag{2.14}$$

Therefore, we can use the ratio of the two robustness measures to calculate the Bayes' factor. If two models have a different number of parameters, the robustness of the system will only increase if |F| increases by more than the proportion by which |P| increased (Woods et al. 2015). A model will be penalised for an additional if it does not increase the volume of the functional region by more than the volume that the added parameter added to the prior. This is true for nested models, where one model is wholly contained in the other.

2.6 Approximate Bayesian Computation (ABC)

2.6.1 ABC algorithms

Stability Finder is based on a statistical inference method which combines ABC with Sequential Monte Carlo (SMC) (Toni et al. 2009). This simulation-based method uses an iterative process to arrive at a distribution of parameter values that can give rise to observed data or a desired system behaviour (Barnes et al. 2011).

ABC methods are used for inferring the posterior distribution in cases where it is too computationally expensive to evaluate the likelihood function. Instead of calculating the likelihood, ABC methods simulate the data and then compare the simulated and observed data through a distance function (Toni et al. 2009). Given the prior distribution $\pi(\theta)$ we can approximate the posterior distribution, $\pi(\theta \mid$

$x) \propto f(x \mid \theta)\pi(\theta)$, where $f(x \mid \theta)$ is the likelihood of a parameter, θ, given the data, x. There are a number of different variations of the ABC algorithm depending on how the the approximate posterior distribution is sampled.

The simplest ABC algorithm is the ABC rejection sampler (Pritchard et al. 1999). In this method, parameters are sampled from the prior and data simulated through the data generating model. For each simulated data set, a distance from that of the desired behaviour is calculated, and if greater than a threshold, ε, the sample is rejected, otherwise it is accepted.

Algorithm 1 ABC rejection algorithm

1: Sample a parameter vector θ from prior $\pi(\theta)$
2: Simulate the model given θ
3: Compare the simulated data with the desired data, using a distance function d and tolerance ε. if $d \leq \varepsilon$, accept θ

The main disadvantage of this method is that if the prior distribution is very different from the posterior, the acceptance rate is very low (Toni et al. 2009). An alternative method is the ABC Markov Chain Monte Carlo (MCMC) developed by Marjoram et al. (2003). The disadvantage of this method is that if it gets stuck in an area of low probability it can be very slow to converge (Sisson, Fan, & Tanaka n.d.).

The method used here is based on Sequential Monte Carlo, which avoids both issues faced by the rejection and MCMC methods. It propagates the prior through a series of intermediate distributions in order to arrive at an approximation of the posterior. The tolerance, ε, for the distance of the simulated data to the desired data is made smaller at each iteration. When ε is sufficiently small, the result will approximate the posterior distribution (Toni et al. 2009).

ABC SMC can identify the parameter values within a predefined range of values that can achieve the desired behaviour. It works by first sampling at random from the initial range set by the user, i.e. form the prior distribution of values. Each sample from the priors is called a particle. It then simulates the model given those values and compares that to the target behaviour. If the distance between the simulation and the target behaviour is greater than a predefined threshold distance ε, then the parameter values that produced that simulation are rejected. This is repeated for a predefined number of samples which are collectively referred to as a population. Each particle in a population has a weight associated with it, which represents the probability of it producing the desired behaviour. At subsequent iterations the new samples are obtained from the previous populations and the ε is

set to smaller value, thus eventually reaching the desired behaviour. The algorithm proceeds as follows:

Algorithm 2 ABC SMC algorithm

1: Select ε and set population t = 0
2: Sample particles (θ). If t = 0, sample from prior distributions (P). If t > 0, sample particles from previous population.
3: If t > 0: Perturb each particle by ± half the range of the previous population (j) to obtain new perturbed population (i).
4: Simulate each particle to obtain time course.
5: Reject particles if d > ε.
6: Calculate the weight for each accepted particle. At the first population assign a weight equal to 1 for all particles. In subsequent populations the weight of a particle is equal to the probability of observing that particle divided by the sum of the probabilities of the particle arising from each of the particles in the previous population:
7: $w_t^{(i)} = \begin{cases} 1, & \text{if } n = 0 \\ \frac{P(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})}, & \text{if } n{>}0. \end{cases}$

This algorithm is implemented on a simple example for illustration. A simple model was used, consisting of one species, A converting to another, B. The model is described by two differential equations, where A is the reactant and B the product, produced at a rate p.

$$\frac{d[B]}{dt} = p[A] \tag{2.15}$$

$$\frac{d[A]}{dt} = -p[A] \tag{2.16}$$

The priors were set to $p \sim U(0, 10)$. Initial conditions for A and B were set to 1 and 0 respectively. The data to which the model was compared to was generated by simulating the same model with the parameter set to 1, as shown in Figure 2.1.

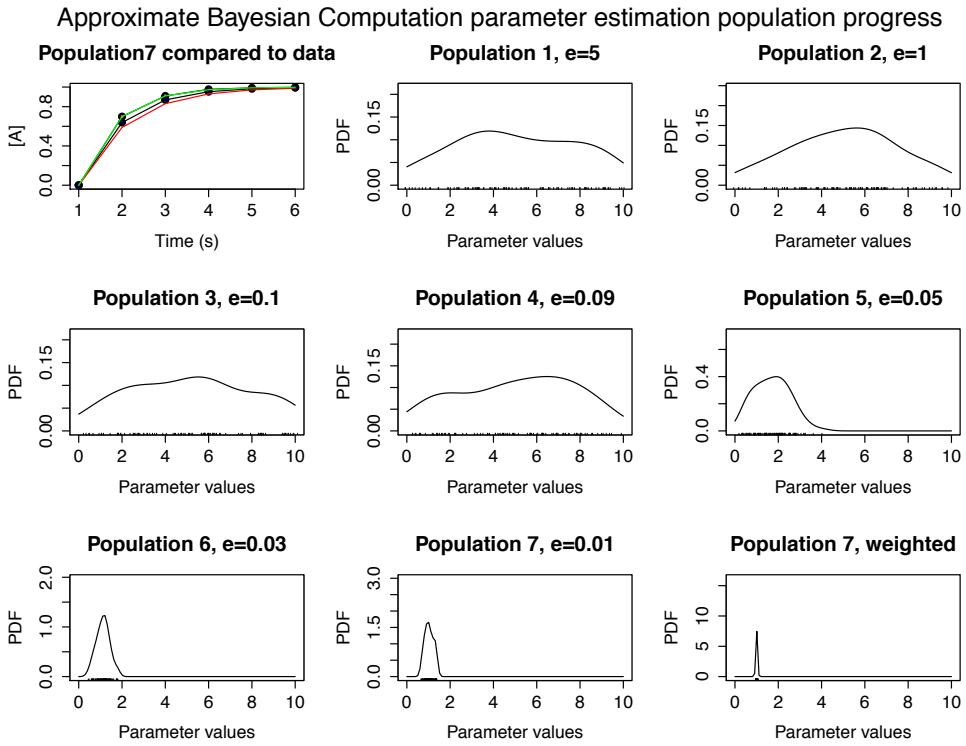

Figure 2.1 ABC SMC parameter inference. The posterior parameter is equal to 1 and its time course shown in red in the top left panel. The blue time course is that of the final population, green is the upper quartile and red is the lower quartile range of values. The progress of the selection process can be seen the ε schedule proceeds from the top left to the bottom right. The bottom far right panel is a density plot of ε= 0.01 with their weights taken into account.

Figure 2.1 demonstrates, using a simple example, that ABC SMC is capable of fitting a model to the data. During the course of 7 populations, the accepted distance ε of the simulated particles to the data is incrementally decreased. This leads to a final population where the distance of the data to the particles is very small, and there is a good agreement between the two. The algorithm concludes with a set of parameter values that produced this behaviour, which approximate the posterior distribution. The posterior distribution found in this model is in good agreement with the parameter value used to generate the data. This example successfully demonstrates the effectiveness of the ABC SMC algorithm in fitting models to data.

3 Bayesian model selection

3.1 Introduction to ABC-SysBio

3.2 Methods

3.2.1 ABC for parameter estimation

3.2.2 ABC for model selection

3.2.3 Particle sampling

For the first population, particles are sampled from the priors. Random samples are taken from the distribution specified by the user for each parameter.

For subsequent populations particles are sampled from the previous population. The weight of each particle in the previous population dictates the probability of it being sampled. The number of samples to be drawn is specified by the user in the input file.

3.2.4 Perturbation

Each sampled particle is perturbed by a kernel defined by the distribution of the previous population, as developed by Toni et al. (2009).

$$K_p(\theta|\theta*) = \theta * + U(+s_p, -s_p), \text{ where:} \tag{3.1}$$

$$s_p = \frac{1}{2}(max(\theta_{p-1}) - min(\theta_{p-1})) \tag{3.2}$$

If the θ* falls out of the limits of the priors then the perturbation is rejected and repeated until an acceptable θ* is obtained. This method is successful in perturbing

the particles by a small amount in order to explore the parameter space, but can be slow to complete.

3.2.5 Particle simulation

Each particle is simulated using cuda-sim (Zhou et al. 2011). The model is provided by the user in SBML format and is converted into CUDA® code by cuda-sim. The model in CUDA® code format can then be run on NVIDIA®. CUDA®. GPUs. This allows the user to take advantage of the speed of parallelised simulations without any CUDA® knowledge.

3.2.6 Distance function

3.2.7 Weight calculation

For the first population the weights are all given a value of 1, and then normalised over the number of particles. For subsequent populations the weights of the particles are calculated by considering the weights of the previous population (Toni et al. 2009).

$$w_t^{(i)} = \frac{P(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})} \text{ for n} > 0 \tag{3.3}$$

The weights are then normalised over the total number of particles.

3.3 Models of the genetic toggle switch

3.4 Results

3.4.1 Genetic toggle switch model selection

3.5 Conclusions

4 Toggle switch stability

4.1 Introduction

In this chapter, I aim to uncover the underlying principles that govern the stability of a given switch. To do this, I developed an algorithm, called Stability Finder, that can find the parameter value ranges that can produce the desired stability in a given model. I use this algorithm to examine a variety of switch architectures using different modelling abstractions.

Structurally, this chapter is organised as follows: In the first section I examine the current understanding of the stability landscape of the genetic toggle switch. Then, I discuss the development of Stability Finder, justify the choices made and the drawbacks of this method. In the sections following I apply Stability Finder to a variety of models and finally I discuss the implications our findings have to the overall understanding of the toggle switch stability.

4.2 Stability Finder algorithm

To investigate the multistable behaviour of systems, I had to make a number of extensions to existing approaches. Firstly, a wide range of initial condition samples are required in order to determine the stability of a system. For a given set of parameter values, sample points are taken across initial conditions using latin hypercube sampling (McKay, Beckman, & Conover 2000), and the ensemble system simulated in time until steady state. As a distance function I use the desired stability of the simulated model. An overview of the algorithm is given in section 4.2.1. Then each module in the algorithm is described in Sections 3.2.3-4.2.5.

4.2.1 Algorithm overview

The Stability Finder algorithm is summarised below. Stability Finder is available as a Python package, and can be downloaded from https://github.com/ucl-cssb/StabilityFinder.git.

Algorithm 3 StabilityFinder algorithm

1: Initialise ϵ
2: population p $\leftarrow 1$
3: **if** p $= 1$ **then**
4: Sample particles (θ) from priors
5: **else**
6: Sample particles from previous population
7: Perturb each particle by $\pm$ half the range of the previous population (j) to obtain new perturbed population (i).
8: **end if**
9: Sample initial conditions via latin hypercube sampling.
10: Simulate each particle to obtain steady state values.
11: Cluster steady state
12: Reject particles if d $> \epsilon$.
13: Calculate weight for each accepted θ
14: $w_t^{(i)} = \begin{cases} 1, & \text{if } p = 1 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})}, & and \text{if } p \geq 1. \end{cases}$
15: Normalise weights
16: Repeat steps 3 - 15 until $\epsilon \leq \epsilon_T$

The user provides an SBML model file and an input file that contains all the necessary information to run the algorithm, including the desired stability and the final tolerance ε, for the distance from the desired behaviour necessary for the algorithm to terminate. The flow of execution is illustrated in Figure 4.1E. Since the algorithm is computationally intensive, all deterministic and stochastic simulations are parallelised and performed using algorithms implemented on Graphical Processing Units (GPU)s.

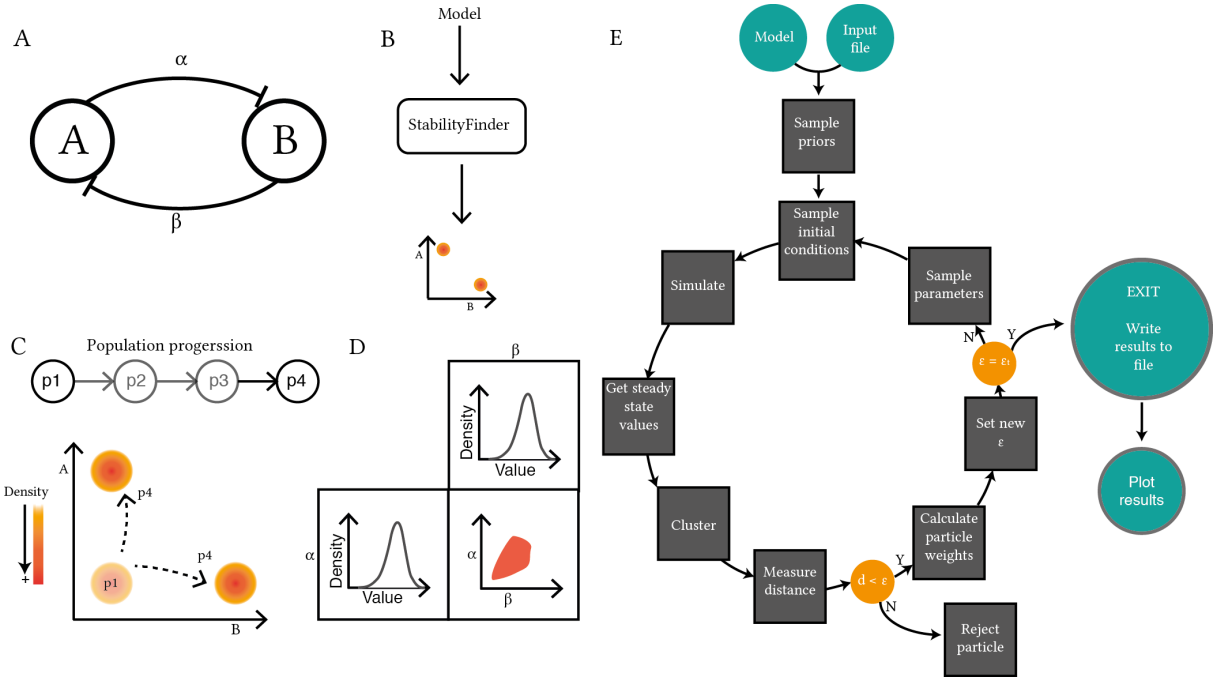

Figure 4.1 : Using sequential Monte Carlo to examine system stability. The algorithm takes as input a model (A) and evolves it to the stability of choice (C) via intermediate populations. In this example model shown in A, There are two species and two parameters. For the model to be bistable, the phase plot of the two species of interest must have two distinct densities, as shown in (C). The parameter space of the model is searched through our algorithm until the resulting simulations give rise to bistability. The parameter values for the model that demonstrated the desired behaviour are given as an output (D). The output consists of the accepted values for each parameter, as well as each density plotted against the other. This allows us to uncover correlations between parameter values. We made this algorithm into a python package, called Stability Finder. The overview of the algorithm is shown in (E).

4.2.2 Initial condition sampling

In Stability Finder, latin hypercube sampling is used to sample initial conditions (McKay, Beckman, & Conover 2000). This is used to ensure that the whole space is sampled uniformly. Latin hypercube sampling is done in two dimensions in Stability Finder. The uniform priors of the two species in consideration represent a rectangle space, which is subdivided into equal parts. Then a random sample is drawn from each sub-part. This ensures the whole space is evenly sampled.

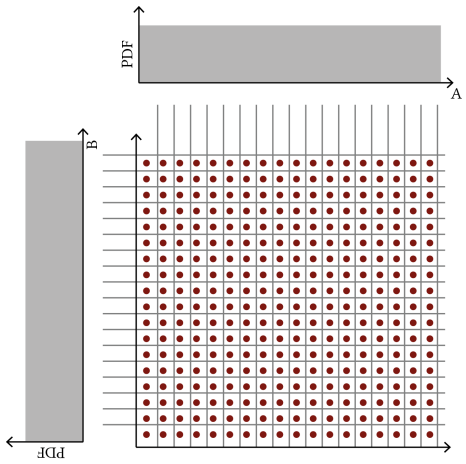

Figure 4.2 Latin hypercube sampling ensures that the whole space is sampled evenly. For the two species concerned, A and B, we assume uniform distributions, shown in grey. The joint space of the two distributions is divided into smaller equal parts and a random sample is drawn from within each subspace.

Stability Finder can only be used for stability analysis concerning two species. By that I refer to models where the phase plot is always and only two-dimensional. Stability landscapes involving more that two species are beyond the scope of this thesis.

4.2.3 Distance function

The distance function is the function the algorithm uses to compare the desired behaviour to the behaviour observed in each particle (Toni et al. 2009). In Stability Finder the distance function consists of three distances. The first one is the difference between the number of desired clusters and the number of clusters observed in the phase plot. For this distance metric the number of clusters in the phase plot must be calculated. The clustering methods used are outlined in section 4.2.3.1.

The other two distance metrics used in Stability Finder are the variance within each cluster and the overall, between cluster, variance. The within cluster variance ensures that the clusters are tight, and the between cluster variance is used to ensure the clusters are far apart from each other. In the context of this thesis, the ideal behaviour of a system is tight, widely separated clusters. This means that the genetic system has distinct steady states, and the difference in the protein levels between each steady state is observable.

4.2.3.1 Clustering methods

Whether the model was simulated using ODEs or the Gillespie algorithm (Gillespie 1977) dictated the method of clustering that we used. For the deterministic models I used an algorithm I developed, that will be referred to as the delta clustering algorithm in this thesis. This algorithm consists of defining the number of clusters by counting a new cluster every time a data point is more than a distance δ away from any existing clusters. The benefits of the delta clustering algorithm are that it is fast and can be used on deterministic solutions, where steady state values tend to be identical if all the particles have reached steady state.

Steady states of stochastic models are clustered using the K-means clustering (Lloyd 1982) and the number of clusters determined using the Gap statistic (Tibshirani, Walther, & Hastie 2001). This method is more suited to stochastic solutions, where the delta clustering method would fail as the steady state solutions tend to be more widely dispersed than in the deterministic case. The detailed algorithms used are shown in Appendix (XXX).

The method used for clustering can be altered by the user if he/she wants to add their own preferred clustering algorithm that might be more appropriate for their specific purposes. For the models I used here, the above methods were successful in clustering the steady state solutions.

4.2.4 Particle rejection

Once the distance from the desired behaviour has been calculated, the algorithm rejects any particles whose distance is farther than the current ε. The distances taken into account are the number of clusters (C), the between cluster variance (V_{bc}) and the within cluster variance (V_{wc}) as outlined in section 4.2.3. In addition to these distances I have included another two checks for the particles. Firstly, Stability Finder checks if the simulation of a particle has reached steady state. If the standard deviation of the last ten time points in the simulation is larger than a user-specified value, then the particle is rejected. This is to ensure that only particles that have reached steady state are considered. Secondly, there is a check for the minimum level of the steady states. This is to allow the user to select for steady states whose protein levels are above a certain threshold. This has to be added as an additional check as the steady state levels must be experimentally observable if they are to be used to design new systems. Two steady state levels of very low levels would be biologically indistinguishable and thus meaningless in an experimental setup. This check is optional to the user, and can be set to zero if not desirable.

4.2.5 Model checking

A problem that can arise by using this method with stochastic simulations is that the behaviour observed may not be the true behaviour but it might be a result of noise. We need to ensure that the resulting behaviour is reproducible. Therefore, I added model checking to the algorithm. Model checking consists of resampling from the posterior distribution and simulating each sample. If the resulting behaviour is the same as what we expected we can be confident that it is the true behaviour of the system and not a result of noise.

4.3 Calculating robustness

Unlike other ABC SMC methods, Stability Finder does not have model selection integrated into the method. This is because the purpose of Stability Finder is not necessarily to compare models for robustness but to elucidate the stability a given model is capable of. Nevertheless, robustness analysis is an outcome that Bayesian methods are well suited for. Therefore, here I discuss another algorithm I developed in order to extract robustness information from the results of Stability Finder and apply model selection.

As discussed in Section 2.5.5, two models can be compared for their robustness using Equation 2.14. This represents the ratio of the robustness measure of each model which in turn is defined as the ratio between the volume of functional region F and the volume of the prior P. In order to calculate the Bayes' factor we must first be able to approximate the volume of the viable parameter space. The viable parameter space is the space that approximates the posterior distribution that can give rise to the desired behaviour. I tested two methods of approximating the volume of the viable space, which are outlined in Algorithm 4. The first method is based on the method used by (Hafner et al. 2009), where the volume of the cuboid containing all the viable space is calculated. I modified this part of their method by only including the area of the viable space where the majority of the last population lies. Therefore only the 1^{st} and 99^{th} percentile of the viable space are taken into account. This is necessary in order to exclude outliers in the distribution that would skew the volume calculation significantly. Each parameter represents a side in the cuboid and since the volume of a cuboid is equal to the product of its sides, the volume of the viable space is equal to the product of the ranges of all the parameters. This cuboid method will be prone to overestimating robustness especially in cases of correlation between parameters. This caveat could be alleviated if a Principal component analysis (PCA) (Fukunaga 2013) is done on the data before the cuboid is calculated (Hafner et al. 2009). This would align the axes of the cuboid to the major axes of the distribution. This would still be a crude estimation of the volume, since if the posterior distribution is assumed to be normally distributed the volume would still be overestimated.

Thus we used a second method, where the volume of the viable space was represented by a hyper-ellipsoid, an ellipse in higher dimensions. This method should not be as prone to overestimation of robustness as the cuboid method as an ellipsoid can take correlation into account. For this method the distribution of the viable space is assumed to be normal. The method calculates the covariance matrix of the distribution, whose volume is given by Equation 4.1. Just as in the cuboid method, the 1^{st} and 99^{th} percentile of the data is ignored.

$$V = \frac{2\pi^{\frac{k}{2}}}{k\Gamma(\frac{k}{2})}\left[\chi^2_k(\alpha)\right]^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}, \tag{4.1}$$

where k is the number of dimensions, Γ is the Gamma function, α is the confidence interval required and $|\Sigma|$ is the determinant of the covariance matrix.

To validate these methods I compare them to ABC-SysBio model selection (Liepe et al. 2014). ABC-SysBio has been used extensively for model selection (Toni et al. 2009; Toni et al. 2011; Barnes et al. 2011) There is good agreement between the three methods as can be seen in Figures 4.3 and 4.4.

Algorithm 4 Approximating robustness

for each model m of M **do**
  Prior$\sim U(a,b)$
  $V^m_{prior} = \prod_{i=1}^{k}(i_b - i_a)$
  Get $1^{st} < data < 99^{th}$ percentiles
  if Cuboid calculation **then**
    $V^m_{post} = \prod_{i=1}^{k}(i_{max} - i_{min})$
  end if
  if Ellipsoid calculation **then**
    Calculate data covariance matrix
    $V^m_{post} = \frac{2\pi^{\frac{k}{2}}}{p\Gamma(\frac{k}{2})}\left[\chi^2_k(\alpha)\right]^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}$
  end if
  $R^m = \frac{V^m_{post}}{V^m_{prior}}$
  $R^m_{norm} = \frac{R^m_i}{\sum_{i=1}^{M} R^m_i}$
end for

We use the following two examples used in ABC-SysBio (Toni et al. 2009):

4.3.1 Case study 1: Infectious diseases

For the first case study utilizes the models used in autociteToni:2009tr. As described in Toni et al. (2009), the models describe the spread of an infectious disease through a population over time. The population is made up of susceptible, infected or recovered individuals, denoted as S, I and R respectively. Three models are compared for the robustness of their posterior distributions. The first model (Model 1), is the simplest model of the three. Each individual S or R can be infected once and then it can immediately infect other individuals (Toni et al. 2009).

$$\dot{S} = \alpha - \gamma SI - dS \tag{4.2}$$

$$\dot{I} = \gamma SI - vI - dI \tag{4.3}$$

$$\dot{R} = vI - dR, \tag{4.4}$$

where a denotes the birth rate, d the death rate, γ the infection rate, and v the recovery rate.

The second model, Model 2, includes a time delay between an individual getting infected and being infectious. δ denotes the rate of transition of a non-infectious infected individual to an infectious one.

$$\dot{S} = \alpha - \gamma SI - dS \tag{4.5}$$

$$\dot{L} = \gamma SI - \delta L - dL \tag{4.6}$$

$$\dot{I} = \delta L - vI - dI \tag{4.7}$$

$$\dot{R} = vI - dR, \tag{4.8}$$

Finally the third model, Model 3, extends Model 1 and includes the recovered individuals being able to become susceptible again. This is denoted by rate e.

$$\dot{S} = \alpha - \gamma SI - dS + eR \tag{4.9}$$

$$\dot{I} = \gamma SI - vI - dI \tag{4.10}$$

$$\dot{R} = vI - dR - eR, \tag{4.11}$$

The three models are simulated using ODEs. In ABC-SysBio model selection is used. Parameter inference is also used for each model separately without the use

of model selection. I used the two methods outlined in Algorithm 4 to calculate the robustness of the posterior distributions of all three models. This robustness measure was then compared to the result of ABC-SysBio model selection. As shown in Figure 4.3, there is good agreement between the three measures of robustness. The posterior distributions of all three models are also shown in Figure 4.3.

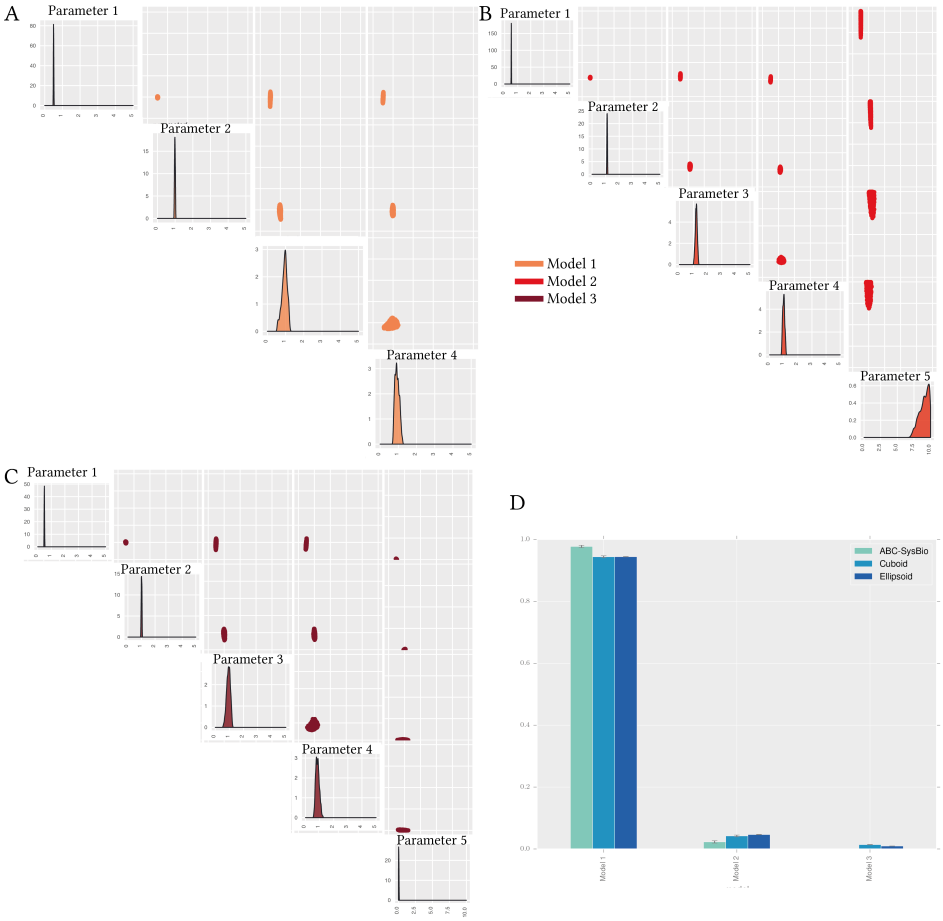

Figure 4.3 : Robustness analysis of the three models for the spread of infectious diseases. (A-C) The posterior distributions of the three models compared. (D) I use three methods to calculate robustness, ABC-SysBio model selection, the volume of the hyper-cuboid approximation of the posterior distribution and the volume of the hyper-ellipsoid approximation of the posterior distribution. Each analysis was repeated three times. The height of the bars indicate the mean robustness from the three repeats and the error bars represent the standard deviation. There is good agreement between all three methods. All three methods show that Model 1, the simplest model, is the most robust model.

4.3.2 Case study 2: Population growth

The second example I will use to demonstrate the effectiveness of the methods used here for robustness calculation is a population growth model. This is another example used in ABC-SysBio (Toni et al. 2009).

The data was obtained by simulating an immigration-death model shown in Equation 4.12. This model (referred to as Model 1) and a model of logistic growth are compared for robustness of their posterior distributions. Model 1:

$$\frac{dI}{dt} = \alpha - \beta I \tag{4.12}$$

Logistic growth, model 2:

$$\frac{dI}{dt} = \gamma - I(\delta - \epsilon I) \tag{4.13}$$

As in Section 4.3.1, two analyses were carried out on these two models. First, ABC-SysBio model selection was used to find the most robust model. Then parameter inference was done on each model. The resulting posterior distributions (shown in Figure 4.4), were compared for robustness using the cuboid and the ellipsoid approximation methods. All three robustness measures find that Model 1 is the most robust model. The analysis was repeated for ODE, MJP and SDE simulations, all arriving to the same result of Model 1 being the most robust. The results are shown in Figure 4.4.

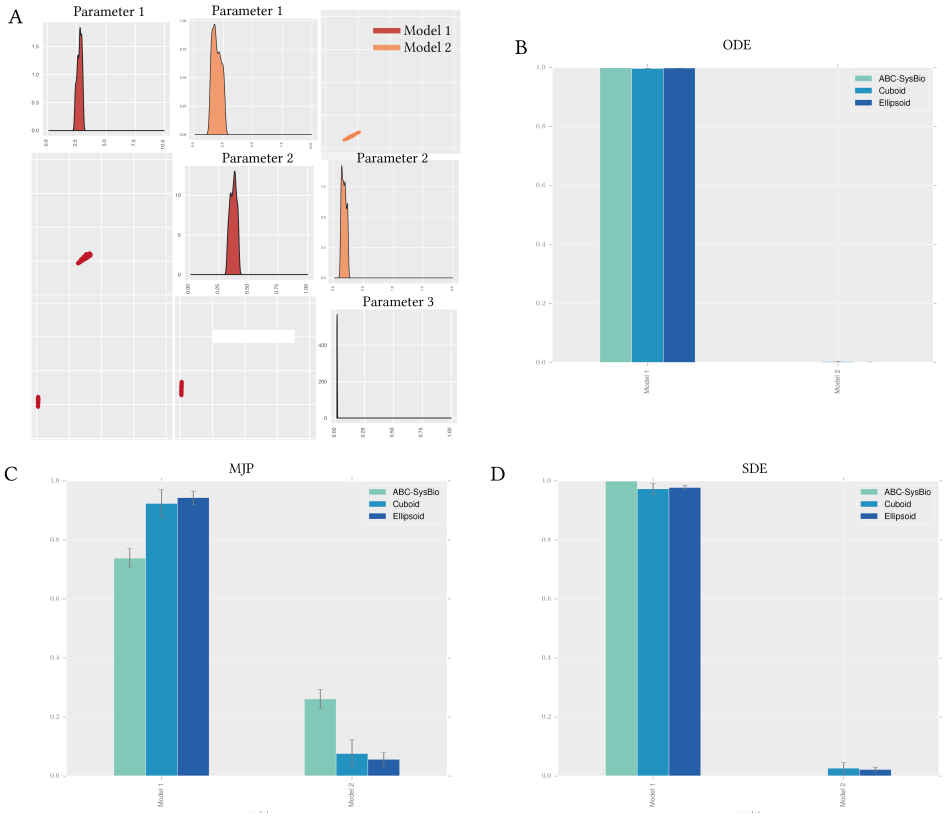

Figure 4.4 : Robustness comparison of two population growth models. (A) The posterior distributions of the two models. (B-D) The models were simulated using ODE, MJP and SDE. Both the cuboid and the ellipsoid approximations agree with ABC-SysBio model selection results. Each analysis was repeated three times. The height of the bars indicate the mean robustness from the three repeats and the error bars represent the standard deviation.

The two case studies used above show that the cuboid and the ellipsoid approximation of model robustness agree with the results obtained from ABC-SysBio model selection. A point I must draw attention to is that for ABC-SysBio model selection where model selection is incorporated in the process, each model is also considered a particle with an associated weight (Toni et al. 2009). If a model is performing poorly it does not proceed in the algorithm and is dropped when the weight falls low enough so that the model is not sampled (Toni et al. 2009). This can save time in the analysis as computational resources are not wasted on 'dead' models, models that perform the required behaviour poorly. Using Stability Finder for model selection, each model must reach the given final ε in order for the cuboid and ellipsoid methods to be valid. This means that time and computational power will be spent on models that are potentially a bad fit, or that have posterior distributions so small compared to the prior that it will take a long time for Stability Finder to find it. Despite this, the results agree between the all three methods of model selection. This shows that the requirement for all models to reach the final ε does not affect the results for the models used in the above case studies. The potentially wasted computational resources on 'dead' models is a compromise we make in order to be able to run the models separately, as model selection is not the primary purpose of Stability Finder.

4.4 Applications of Stability Finder

In this section I apply Stability Finder to switch models in order to find the design principles underlying their stabilities. First I apply it to a simple model with known results, the Gardner, Cantor, & Collins (2000) toggle switch. This model can serve as a test for Stability Finder, as the conditions for bistability are derived in Gardner, Cantor, & Collins (2000).

4.4.1 Testing StabilityFinder

Gardner, Cantor, & Collins (2000) constructed the first synthetic genetic toggle switch (Gardner, Cantor, & Collins 2000). Their model consisted of two mutually repressing transcription factors, as shown in Figure 4.5A, and in the deterministic case is defined by the following ODEs:

$$\frac{du}{dt} = \frac{a_1}{1 + v^\beta} - u \tag{4.14}$$

$$\frac{dv}{dt} = \frac{a_2}{1+u^{\gamma}} - v, \tag{4.15}$$

where u is the concentration of repressor 1, v the concentration of repressor 2, a_1 and a_2 denote the effective rates of synthesis of repressors 1 and 2 respectively, β is the cooperativity of repression of promoter 1 and ɣ of repressor 2. Gardner, Cantor, & Collins (2000) studied the deterministic case and concluded that there are two conditions for bistability for this model; that a_1 and a_2 are balanced and that β, ɣ>1 (Gardner, Cantor, & Collins 2000). I test Stability Finder by using it to find the posterior distribution for which this model exhibits bistable behaviour. Therefore, the desired behaviour is set to two steady states, and using a wide range of values as priors as shown in Table 4.1, I used Stability Finder to find the parameter values necessary for bistability to occur. The posterior distribution calculated by Stability Finder for the Gardner deterministic case is shown in Figure 4.6A.

Table 4.1 Gardner switch priors in the deterministic and stochastic cases

Parameters				Species	
a_1	β	a_2	γ	s_1	s_2
0-60	0-5	0-60	0-5	0-100	0-100

These results agree with the results reported by Gardner, Cantor, & Collins (2000). For this switch to be bistable a_1 and a_2 must be balanced while β and ɣ must both be >1, as can be seen in the marginal distributions of β and ɣ in Figure 4.6A.

I next applied Stability Finder to the case of the Gardner switch under stochastic dynamics using the same priors as the deterministic case, and again searched the parameter space for bistable behaviour. The posterior distribution is shown in Figure 4.6B. We can see that the conditions on the parameters required for bistability in the deterministic case generally still stand in the stochastic case. There appears to be slightly looser requirements on the parameters of the stochastic model (wider marginal distributions). Some difference between the deterministic and stochastic posteriors is expected as different clustering algorithms are used for the stochastic and the deterministic cases. The Gap statistic is used in the case of the stochastic case, as it is capable of dealing with noisier data whereas a simpler and faster algorithm is used for clustering the deterministic solutions. These results demonstrate that Stability Finder can be used to find the parameter values that can produce a desired stability and can be confidently applied to more complex models.

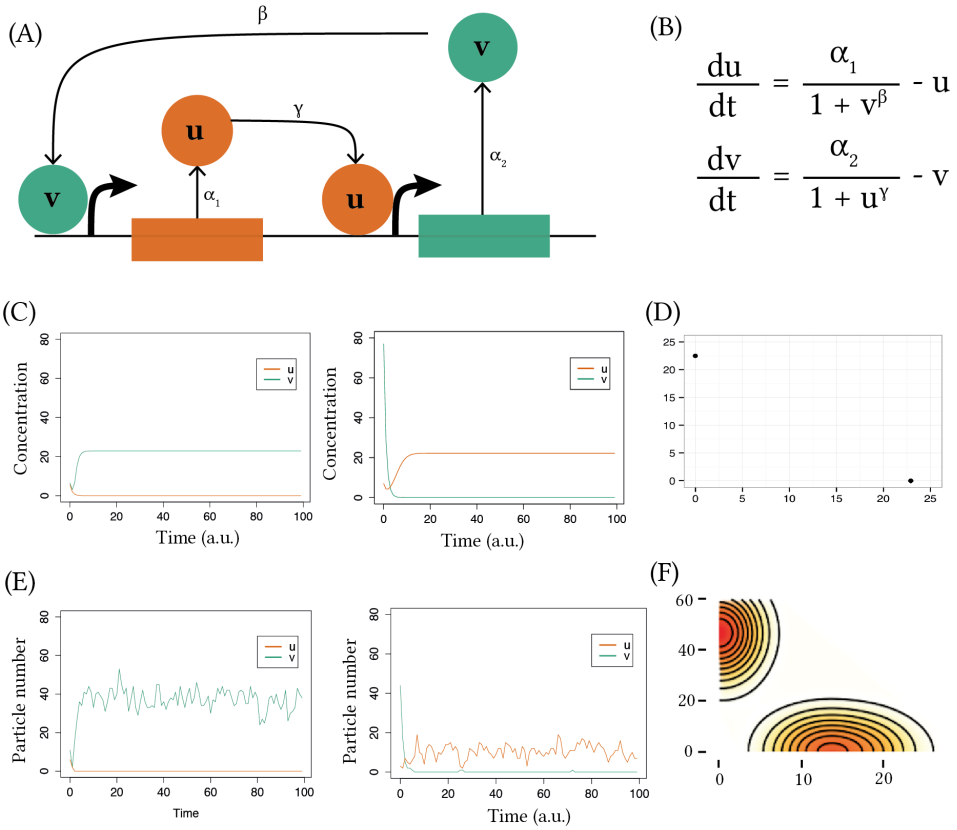

Figure 4.5 The Gardner switch model used to test Stability Finder. The Gardner model (A) consists of two mutually repressing transcription factors. It can be reduced to a two-equation system (B) , where u and v are the two transcription factors, a_1,a_2 are their effective rates of synthesis, u,v are their concentrations and β, γ represent the cooperativity of each promoter. (C) Two samples of deterministic simulated timecourses of the Gardner switch and (D) The resulting phase plot. (E) Two samples of timecourses of the stochastic simulations and (F) the resulting phase plot.

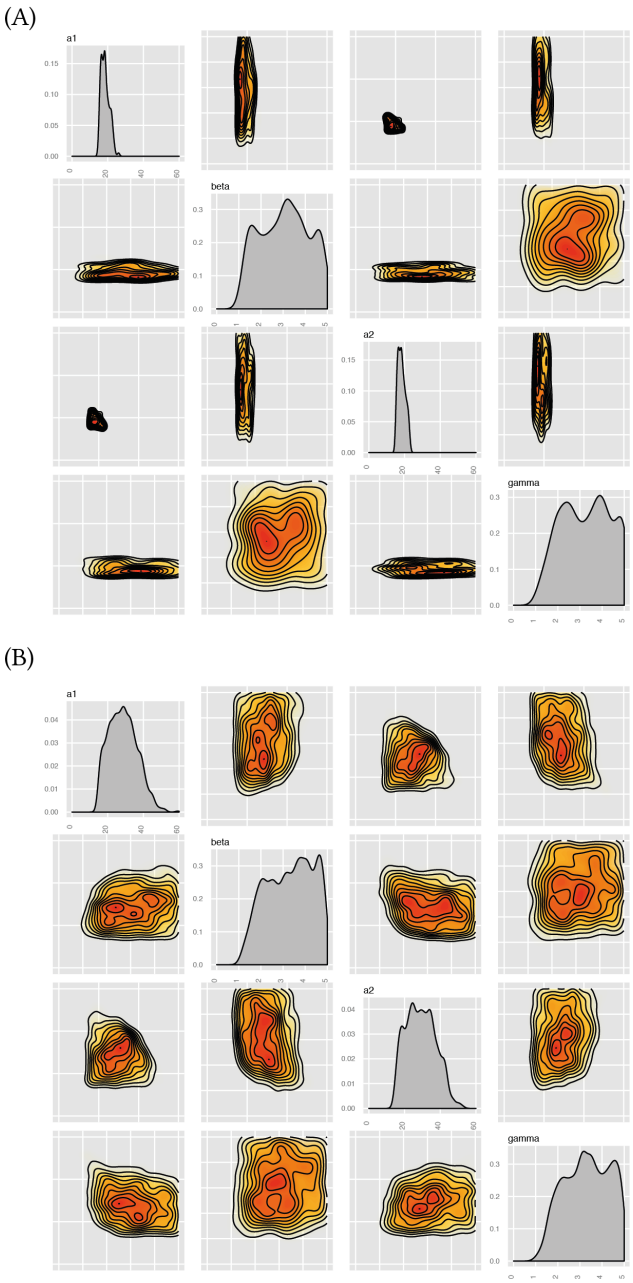

Figure 4.6 Elucidating the stability of the Gardner switch. The Gardner model has four parameters, for which I want to find the values for which this system is bistable. I use Stability Finder to find the posterior distribution of the bistable Gardner switch, deterministically (A) and stochastically (B). The posterior distributions are shown as the density plots of each parameter as well as each one plotted against the other.

4.4.2 Lu toggle switch models

Next I analyzed an extension of the Gardner switch model developed by Lu, Onuchic, & Ben-Jacob (2014). I use these models as they are of increased complexity from the Gardner model. Lu, Onuchic, & Ben-Jacob (2014) considered two types of switches, the classic switch consisting of two mutually repressing transcription factors (model CS-LU), as well as a Lu double positive switch (DP-LU). The CS-LU switch was found to be bistable given the set of parameters used, while the DP-LU switch was found to be tristable (Lu, Onuchic, & Ben-Jacob 2014). The CS-LU model used in their study is given by the following system of ODEs.

$$\dot{x} = g_x H^S_{xy}(y) - k_x x \tag{4.16}$$

$$\dot{y} = g_y H^S_{yx}(x) - k_y y, \tag{4.17}$$

where:

$$H^S_I(x) = H^-_I(x) + \lambda_I H^+_I(x) \tag{4.18}$$

$$H^-_I(x) = 1/\left[1 + (x/x_I)^{n_I}\right] \tag{4.19}$$

$$H^+_I(x) = 1 - H^-_I(x), \tag{4.20}$$

and the DP-LU model is given by

$$\dot{x} = f_x(x,y) = g_x H^S_{xy}(y)\, H^S_{xx}(x) - k_x x \tag{4.21}$$

$$\dot{y} = f_y(x,y) = g_y H^S_{yx}(x)\, H^S_{yy}(y) - k_y y, \tag{4.22}$$

g_I represents the production rate, k_I the degradation rate, n_I the Hill coefficient, x_I the Hill threshold concentration and λ_I the fold change of the transcription rates, and $I \in \{xy, yx, xx, yy\}$.

For the parameter values used in the Lu study, the CS-LU switch exhibits three steady states, two of which are stable and one is unstable. The CS-LU switch exhibits five steady states, of hich three are stable and two are unstable. This analysis of the stability of the switch steady states was carried out by Mae Woods, PhD. Bifurcation diagrams of the two Lu models are shown in Figure 4.7.

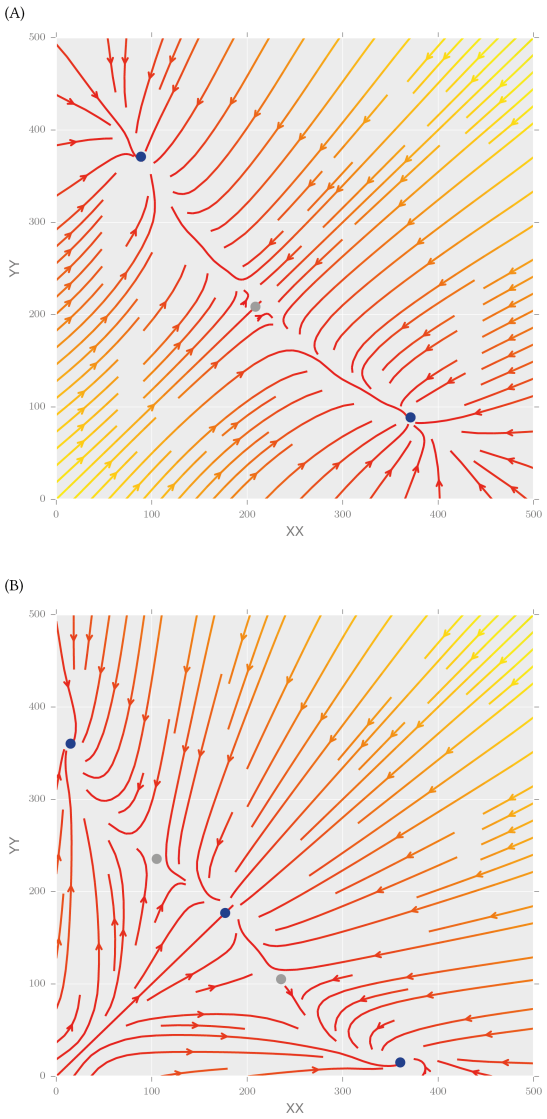

Figure 4.7 Stream plot of the vector plot of the (A) CS-LU and DP-LU switches. The colours indicate the magnitude of the vectors, with yellow indicating high and red low values. The blue points represent stable steady states and the grey points represent unstable steady states.

4.4.2.1 Extending the Lu models

I start the analysis of the Lu models by extending their analytical approach to solving the system. I use Stability Finder to explore a larger parameter space which allows us to distinguish between rare events and robust behaviours. The advantage of using Stability Finder over solving the system analytically is that the full parameter space is explored rather than solving the system for a single set of parameters. This allows us to deduce model properties that could not otherwise be identified. Robustness to parameter fluctuations can be explored, as well as parameter correlations and restrictions on the values they can take while still producing the desired behaviour.

It is known that that the addition of positive autoregulation to the classical toggle switch can induce tristability (Lu, Onuchic, & Ben-Jacob 2014). Here I investigate the interplay of positive autoregulation on the values of the other parameters in the model. I extended the analysis presented in Lu, Onuchic, & Ben-Jacob (2014) by including the switch with single positive autoregulation (model SP-LU), where an asymmetry of positive feedbacks is present between the two genes. The three switches considered in this analysis are shown in Figure 4.8.

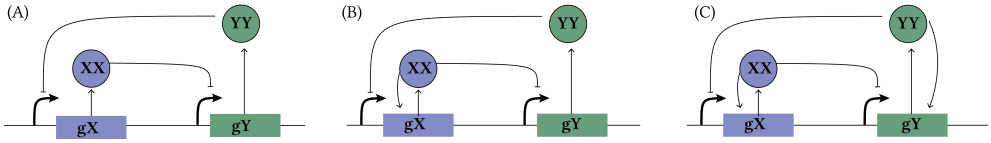

Figure 4.8 The three LU toggle switch models. (A) CS-LU, (B) SP-LU and (C) DP-LU.

The SP-LU switch is modelled using the following ODE system

$$\dot{x} = g_x H^S_{xy}(y)\, H^S_{xx}(x) - k_x x \tag{4.23}$$

$$\dot{y} = g_y H^S_{yx}(x) - k_y y. \tag{4.24}$$

Using Stability Finder with priors centred around the parameter values used in the original paper (see Table 4.2), we can identify the most important parameters for achieving the models' stability. The phase plots of the final populations of the models are shown in Figure 4.9 and the posterior distribution of these models are shown in Figure 4.10A. We find that the parameters representing the rates of degradation of the transcription factors in the system (k_x,k_y) must both be large in relation to the prior ranges for bistability to occur. Protein degradation rates have been shown to be important for many system behaviours including oscillations (Woods et al. 2015).

Table 4.2 Priors of the classical(CS-LU), single positive (SP-LU) and double positive (DP-LU) models.

Parameter	Symbol	CS-LU	SP-LU	DP-LU
Production rate	gx	30-50	1-2	1-100
	gy	30-50	20-25	1-100
Degradation rate	kx	0-0.5	50-55	0-1
	ky	0-0.5	48-52	0-1
Hill coefficient	nxy	1-5	30-35	0-10
	nyx	1-5	0.1-0.2	0-10
Hill thresholds concentration	xxy	100-300	2-3	100-1000
	xyx	100-300	0.4-0.6	100-1000
Transcription rate fold change	lxy	0-0.5	0.02-0.04	0-1
	lyx	0-0.5	0.02-0.04	0-0.2
Hill coefficient	nXX	-	25-30	0-10
	nYY	-	0.01-0.02	0-10
Hill thresholds concentration	xXX	-	0.4-0.5	50-500
	xYY	-	1-3	50-500
Transcription rate fold change	lXX	-	65-72	1-20
	lYY	-	0.02-0.04	1-20

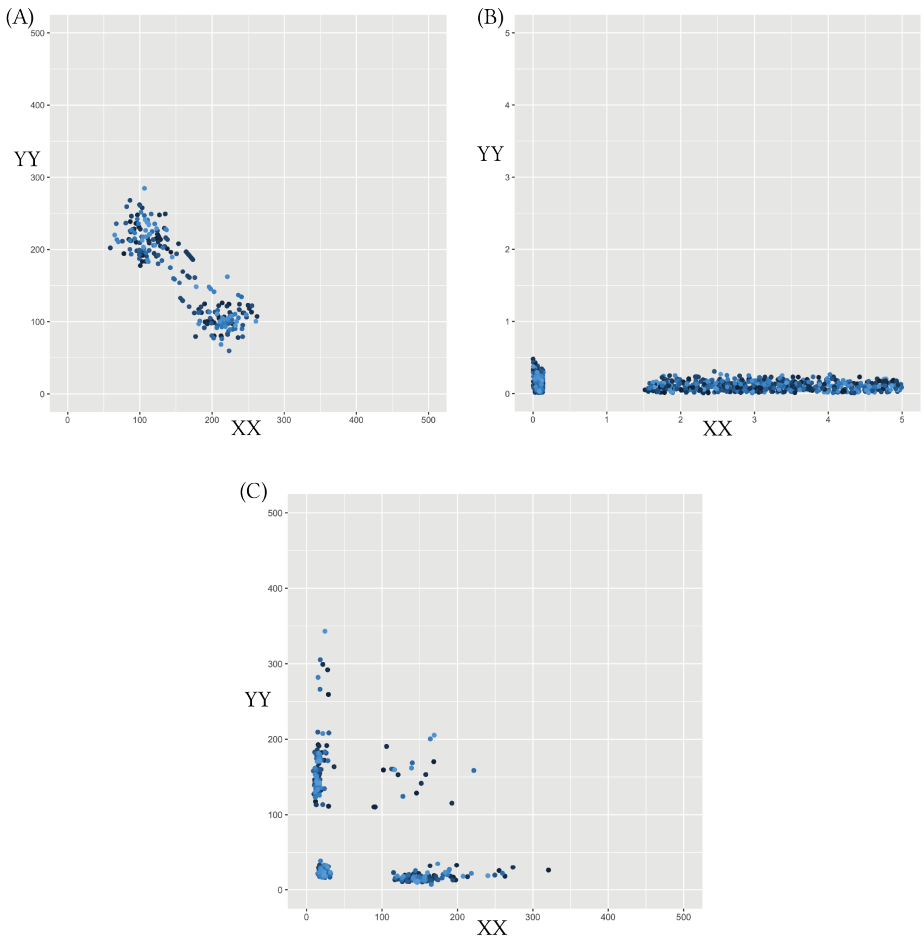

Figure 4.9 : The phase plots of 100 particles from the last population of the three Lu switches. (A) The bistable CS-LU (B) The bistable SP-LU and (C) The tristable DP-LU. We find two types of tristable behaviour, one where the third steady state is zero-zero and one where the third state is high (non-dead).

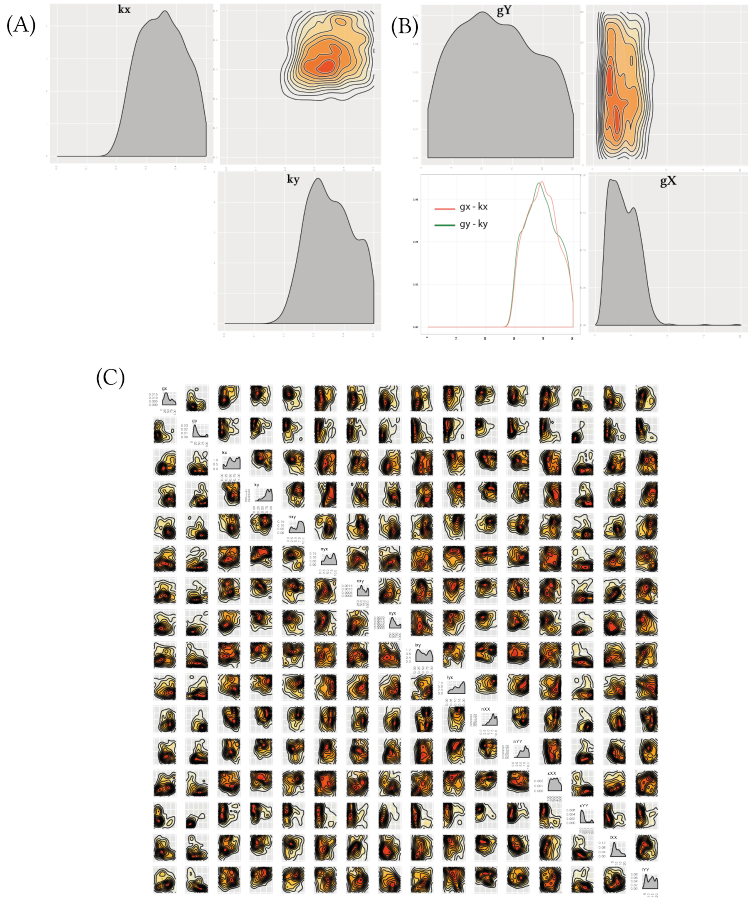

Figure 4.10 : The three variants of the Lu models. (A) The CS-LU switch is bistable. The most restricted parameters for this behaviour are kx and ky which both have to be high relative to the prior while the net protein production for X and Y myst be balanced. (B) The extended Lu model with a single positive autoregulation on X. This model is bistable when gx is small, but the net production of protein is equal for the two nodes.(C) The Lu model with double positive autoregulaiton is tristable, and its posterior distribution shown here.

We find that the switch with single positive autoregulation is capable of bistable behaviour as seen in Figure ??B, but this is only possible when the strength of the promoter under positive autoregulation, gx, is small (Figure 4.10). There appear to be no such constraints on the strength of the original, unmodified, promoter, gy.

Upon examination of the DP-LU model, we also find that tristability in the switch is relatively robust, as tristability is found across a large range of parameter values, with no parameters strongly constrained. Two types of tristable behaviour are identified, one where the third steady state is at (0,0) and and one where the third steady state has non-zero values, as seen in Figure 4.9. This result agrees with previous work by Guantes & Poyatos (2008), who found that a switch can exhibit two kinds of tristability, one in which the third steady state is high (III_H) and one in which it is low (III_L) (Guantes & Poyatos 2008).

4.4.2.2 Multistability in the Lu models

The DP switch is capable of both bistable and tristable behaviour as well as 4 coexisting states under deterministic dynamics (Guantes & Poyatos 2008). It is of great interest to understand the conditions under which these three behaviours occur. Doing a bifurcation analysis of the DP switch can give us an indication of the stabilities this model is capable of, and at which parameter ranges these are found.

Since the Lu models can be solved analytically, we can obtain the bifurcation diagram of the DP-LU by keeping all parameters constant apart from gene expression (gx). The result shown in Figure 4.11B, the system can exhibit 2, 3 or 4 steady states depending on the value of the gene expression rate. We observe that if $100 \leq gx \leq 120$ the system exhibits four steady states, if $9 \leq gx \leq 10$ the system is tristable and if $10 \leq gx \leq 100$ the system is bistable. I use the whole range tested above ($0 \leq gx \leq 140$) as prior distributions in Stability Finder and searched parameter space for 2, 3 and 4 steady states.

Using Stability Finder we obtain a more complex picture of the parameter space that can produce each behaviour. This is because, unlike the bifurcation analysis, Stability Finder does not require any of parameters to be fixed. Since there are no such restraints on the value each parameter can take we obtain a bigger range of parameters that can produce each behaviour than the ranges found during the bifurcation analysis. The priors used for each analysis are identical and include the whole range of values found in the bifurcation diagram, varying only the required number of steady states. In addition, unlike the bifurcation analysis the values for gx and gy are not forced to be equal in the analysis done on Stability Finder.

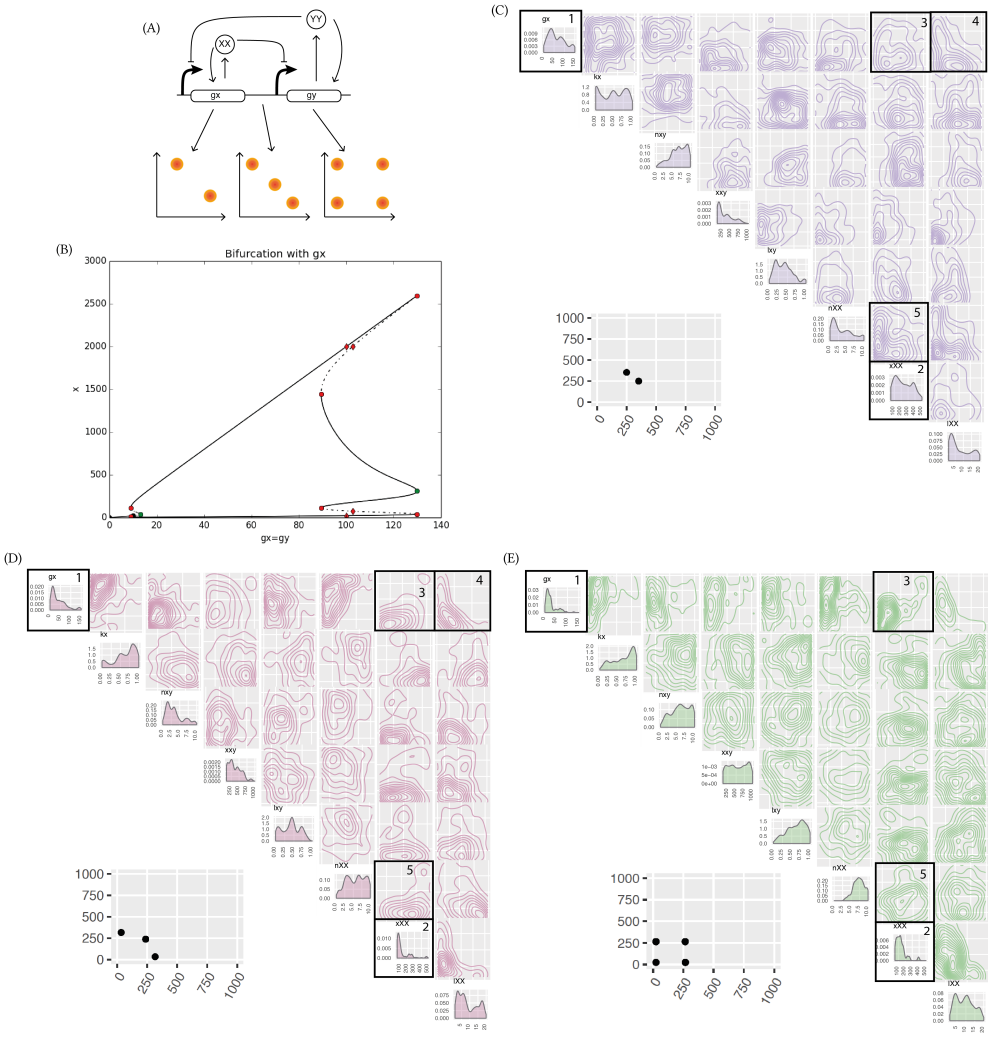

Figure 4.11 : Design principles of multistable switches. (A) Using the Lu model with added positive autoregulation we uncover the design principles dictating if a switch will be bistable, tristable, or will have 4 steady states. (B-D) By considering the bivariate distributions of the parameters we can uncover the differences in the parameters of a bistable switch compared to a tristable switch, compared to a quadristable switch . The posterior distribution of the bistable switch is shown in purple, of the tristable switch in pink and of a quadristable in green. The bivariate distributions for which a difference is observed between teh stabilities are in black boxes. An example of a phase plot from each behaviour is shown next to the corresponding posterior distribution.

Using StabilityFinder, we obtained posterior distributions for bistable, tristable and quadristable behaviours in the DP-LU model and then compared the posterior parameter distributions (Figure 4.11). Upon examination of the posterior distributions for all three switches we observe that a subset of the posterior parameter values is different under the three behaviours. We find differences in the univariate distribution of the parameters for gene expression, gx, as highlighted in Figure 4.11, box 1. This parameter must be small for a quadristable switch to occur but there are no such restraints for a bistable or a tristable switch. Furthermore, parameter xXX must be small for three and four steady states to be achieved but there are no such restraints for a bistable switch, as can be seen in Figure 4.11, box 2.

We also find a difference in the bivariate distributions in the posterior. Most notably, we find that parameters xXX and gX are tightly constrained in the tristable and the four steady state cases, where both parameters are required to be small, but less so in the bistable case (Figure 4.11, box 3). Another notable difference is between parameters xXX and nXX shown in Figure 4.11, box 5, where they are constrained and in the tristable and four steady state cases but not the bistable case. Interestingly, we also find parameter correlations conserved between the three behaviours, as seen in Figure 4.11, box 4, where parameters lXX and gx, positive autoregulation and gene expression are negatively correlated in both cases. This highlights the importance of treating unknown parameters as distributions rather than fixed values when studying the parameter values of a model, as they are capable of uncovering not only the ranges and values needed but also the correlations between parameters that would not have otherwise been detected.

I further analyse these models by studying the phase plots resulting from simulating the particles from the posterior distribution to steady state. The phase plots from 100 particles from each posterior are shown in Figure 4.12. We find that there is a strong conservation on the locations of the steady states between each particle. This indicates that the steady states in a two-node toggle switch tend to be symmetrical. This gives rise to the patterns seen in Figure 4.12. This is especially evident in the quadristable switch. For every steady state at (0,0) there is another steady state on its diagonal, at $XX = YY$. All the combinations of these two steady states form the straight line seen in Figure 4.12C. This indicates that two of the four steady states exist where XX=YY. The other two exist where one of the two proteins dominates the other.

This same principle can be seen in the bistable and the tristable switches. In the bistable switch the two steady states are also symmetrical and one never completely

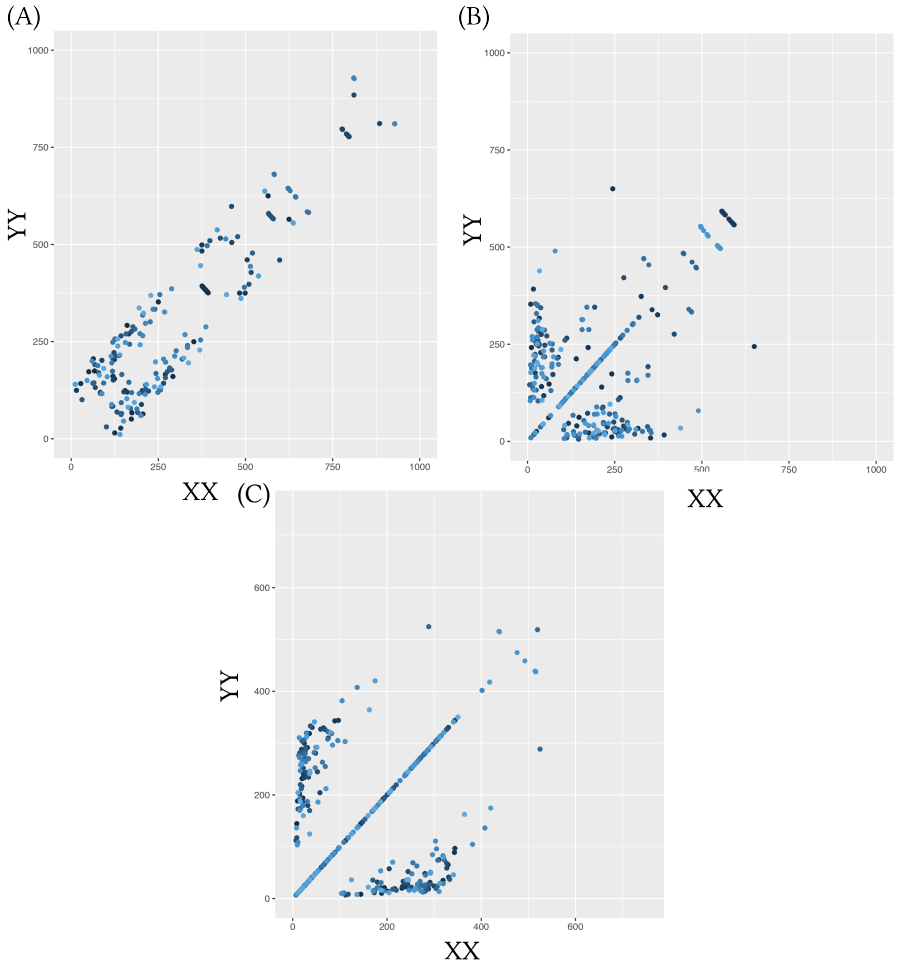

Figure 4.12 : The phase plots from 100 particles from each posterior. Each particle is represented by a different shade of blue. We find a strong conservation on the location of the steady states between particles.

dominates the other. For the tristable case we observe that two of the steady states exist where the levels of one protein is much larger than the other, and a third steady state exists where $XX = YY$. We also observe that the third steady state is not necessarily a 'dead' state, but they can exist over a range of values for XX and YY.

4.4.2.3 Extending the Lu switch to three nodes

To further demonstrate the flexibility of Stability Finder I investigated a system capable of higher stabilities. Multistability is found in differentiating pathways, like the myeloid differentiation pathway (Ghaffarizadeh, Flann, & Podgorski 2014; Cinquin & Demongeot 2005). I allow for these more complex dynamics by extending the DP-LU model by adding another gene, making it a three gene switch. This new system is depicted in Figure 4.13A. This model has symmetric parameters, which means that the parameters for equivalent reactions (e.g. gene expression) are the same. In Stability Finder I look for six steady states, the output being in nodes X and Y and using the priors shown in Table 4.3. We successfully find that the system is capable of six steady states, as shown in Figure 4.13C.

Table 4.3 Priors used in the three-node switch

Parameter	Symbol	Range
Production rate	gx	3-5
Degradation rate	kx	0-0.2
Hill coefficient	nxy	0-2
Hill thresholds concentration	xxy	140-160
Transcription rate fold change	lxy	0-0.2
Hill coefficient	nxx	2-4
Hill thresholds concentration	xxx	90-110
Transcription rate fold change	lxx	8-12

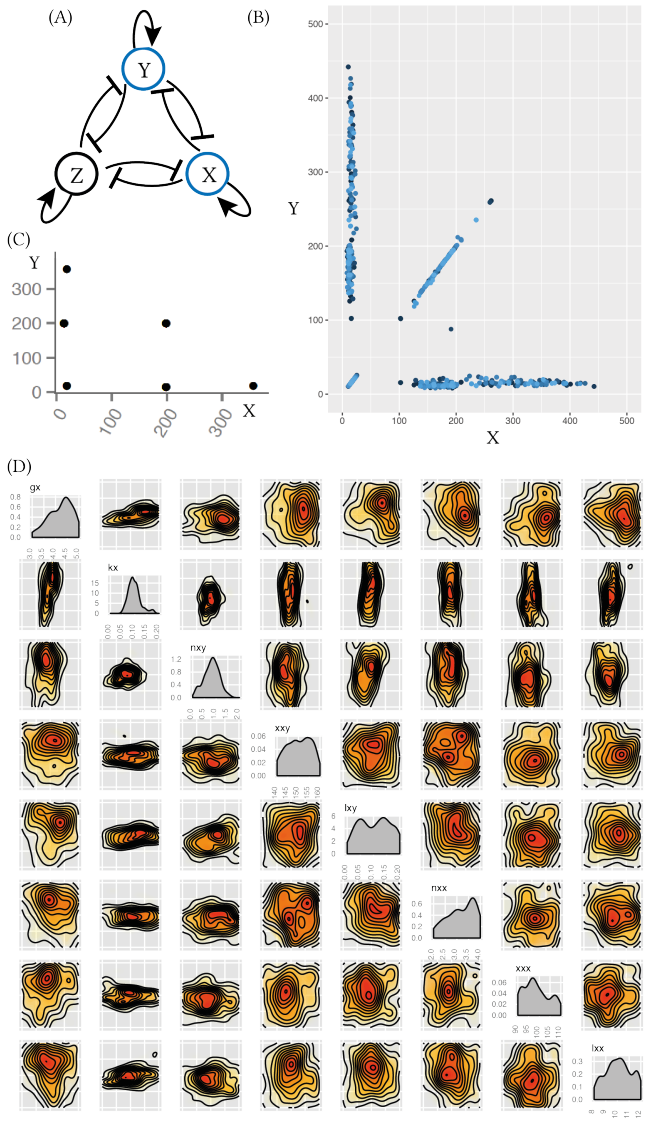

Figure 4.13 : The three-node mutual repression model, with added positive auto-regulation on each node. (A) The model. The model is studied in two dimensions using Stability Finder, for nodes X and Y. (B) The phase plot of 100 particles from the posterior found by Stability Finder. There are 6 steady states. (C) The posterior distribution of the 6-steady state three-node system. Parameters kx and nxy are the most constrained.

We find that the most constrained parameters for this behaviour are again the degradation rate of the proteins, kx. If they are too large or too small the system will not exhibit hexa-stability. Additionally we find that the Hill coefficients for the repressors, nxy, are constrained to be smaller than 1.5 as seen in Figure 4.13D. This example demonstrates that Stability Finder can be used to elucidate the dynamics of more complex network architectures, which will be key to the successful design and construction of novel gene networks as synthetic biology advances.

Consistently with the results found in section 4.4.2.2, we find that the steady states are symmetric (Figure 4.13B). Each of six steady states exists in symmetry with another one, in tightly constrained regions.

4.4.3 Mass Action switches

In order to study the switch system in a more realistic way, I developed an extension to the switches used in Sections 4.4.1 and 4.4.2. This new set of switches does not use the quasi-steady state approximation (QSSA) that is often used in modelling the toggle switch. Using mass action, this changes the two-equation system used in Gardner, Cantor, & Collins (2000) and Lu, Onuchic, & Ben-Jacob (2014) into a system of 8 ODEs and 10 parameters in the classical switch case with no autoregulation (model CS-MA). The equations describing the system are shown below.

$$\mathrm{gA} \xrightarrow{\mathrm{ge}} \mathrm{gA} + \mathrm{A}$$
$$\mathrm{gB} \xrightarrow{\mathrm{ge}} \mathrm{gB} + \mathrm{B}$$
$$\mathrm{A} + \mathrm{A} \xrightarrow{\mathrm{dim}} \mathrm{A2}$$
$$\mathrm{A2} \xrightarrow{\mathrm{dim\,r}} \mathrm{A} + \mathrm{A}$$
$$\mathrm{B} + \mathrm{B} \xrightarrow{\mathrm{dim}} \mathrm{B2}$$
$$\mathrm{B2} \xrightarrow{\mathrm{dim\_r}} \mathrm{B} + \mathrm{B}$$
$$\mathrm{gA} + \mathrm{B2} \xrightarrow{\mathrm{rep}} \mathrm{B2gA}$$
$$\mathrm{B2gA} \xrightarrow{\mathrm{rep\_r}} \mathrm{B} + \mathrm{gA}$$
$$\mathrm{gB} + \mathrm{A2} \xrightarrow{\mathrm{rep}} \mathrm{A2gB}$$
$$\mathrm{A2gB} \xrightarrow{\mathrm{rep\_r}} \mathrm{A2} + \mathrm{gB}$$
$$\mathrm{A} \xrightarrow{\mathrm{deg}} \emptyset$$
$$\mathrm{B} \xrightarrow{\mathrm{deg}} \emptyset$$
$$\mathrm{A2} \xrightarrow{\mathrm{deg\_dim}} \emptyset$$
$$\mathrm{B2} \xrightarrow{\mathrm{deg\_dim}} \emptyset$$

For the model with added double positive autoregulation (model DP-MA) the following equations are added to the system:

$$\begin{aligned}
A2 + gA &\xrightarrow{aut\ 1} A2gA \\
A2gA &\xrightarrow{aut\ 2} A + A2gA \\
A2gA &\xrightarrow{aut\ 3} A2 + gA \\
B2 + gB &\xrightarrow{aut\ 1} B2gB \\
B2gB &\xrightarrow{aut\ 2} B + B2gB \\
B2gB &\xrightarrow{aut\ 3} B2 + gB
\end{aligned}$$

The ODEs describing the above switches are shown in Appendix (XXX). These models are too complex to be solved analytically and ideal for use in Stability Finder. Thus, I use Stability Finder and fit the models to a bistable behaviour. The two models used and the resulting phase plots are shown in Figure 4.14 and the posterior distributions obtained are shown in Figure 4.15.

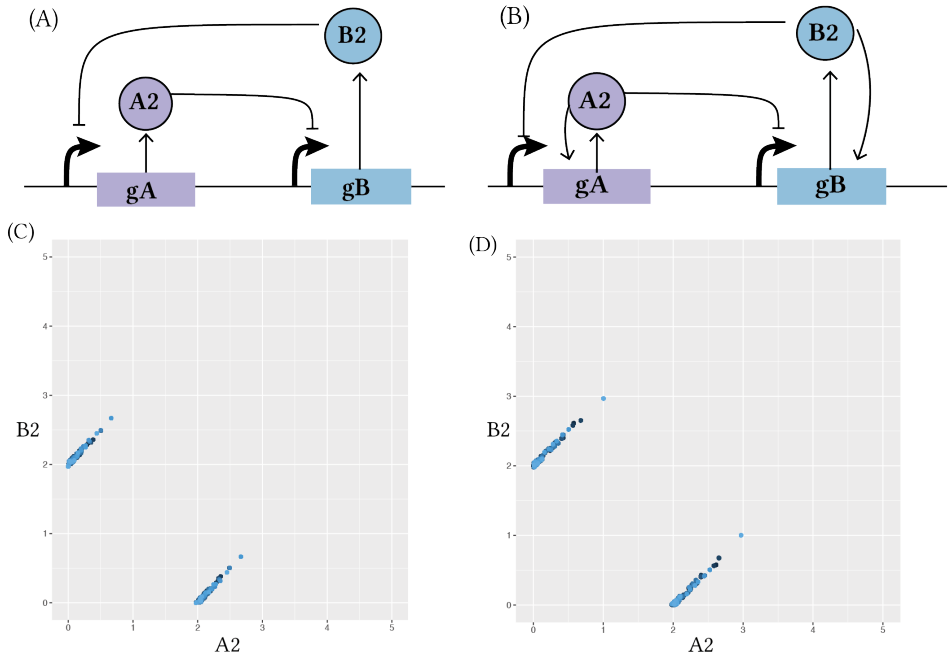

Figure 4.14 : The two mass action switches I developed. (A) The simple switch CS-MA (B) The switch with double positive autoregulation DP-MA. (C, D) The phase plots of 100 particles simulated from the posterior distributions of the bistable mass action switches.

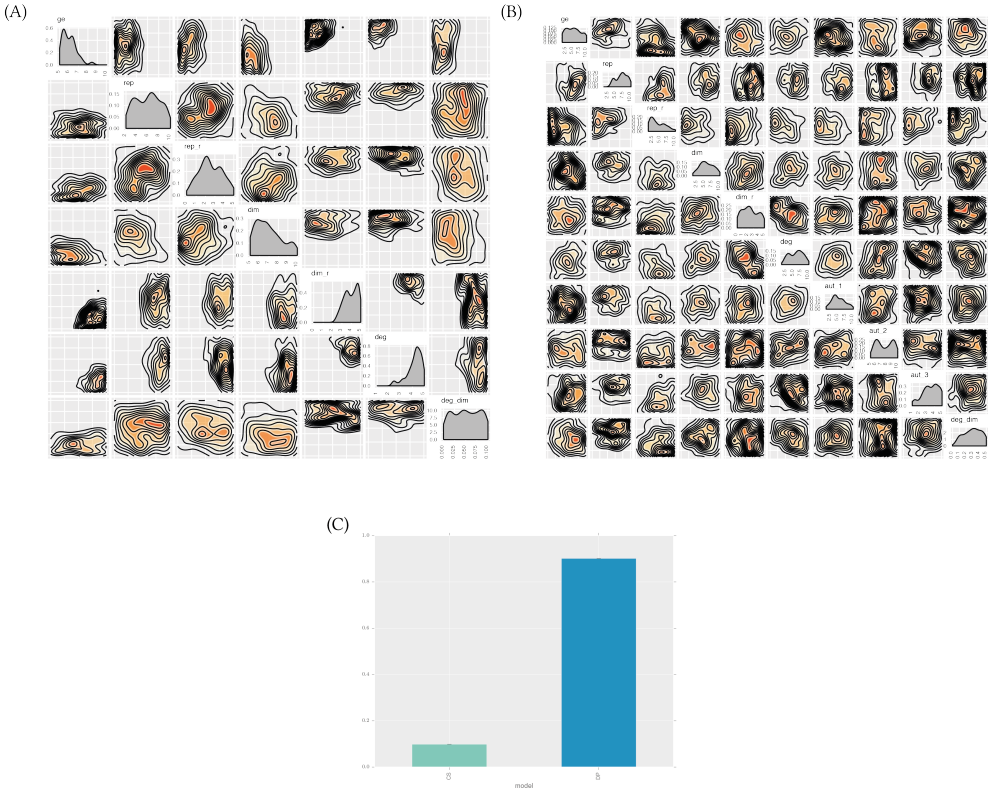

Figure 4.15 : The posterior distributions of the symmetric deterministic (A) CS-MA and (B) DP-MA switches. (C) Robustness comparison of the two models.

By examining the posterior distributions shown in Figure 4.15 we see that the CS-MA is much more constrained that the DP-MA switch. We find that gene expression must be low for bistability to occur in the CS-MA model but there is no such constraint in the DP-MA model. We also find that the monomerization rate *dim_r* and the monomer degradation rate *deg* must both be larger than 2. This is not found in the DP-MA model.

Next I compare the two models for robustness using the ellipsoid method described in Section 4.3. We find that the addition of positive feedback loops greatly increases the system's robustness to parameter fluctuations as seen in Figure 4.15C. Adding positive feedback loops to the model allows it to be bistable over a greater range of parameter values. This indicates that small fluctuations in parameters in the cellular environment will not flip the switch and thus makes it more suitable for use in synthetic biological applications where spontaneous and undesired switching might be detrimental. This makes it a better candidate for building new synthetic devices based on the toggle switch design. We identified the parameter region within which these models are bistable, information that is important when building such a device in the lab.

The models used in the above analysis assume the parameters for equivalent reactions are equal. This is a constraint that simplifies the model. When building this model into a synthetic system in the lab, this assumption is not necessarily justified. When choosing promoters to build this synthetic system two promoters can be chosen to have similar strength but their strength will not necessarily be identical. In order to study how this might affect the results, I further eliminate modelling assumptions made in the toggle switch by making the parameters representing gene expression (*ge*) and repression (*rep*), as well as the protein degradation parameters asymmetric (independent parameters for each protein, versus fixed to be equal). We find that the features of the posterior distributions of the symmetric and the asymmetric models remain the same. The posterior distribution of the asymmetric CS-MA model was more constrained than the posterior distribution of the asymmetric DP-MA model.

We further study the asymmetric mass action models by examining the QSSA aproximation. As stated above, the QSSA is a common analytical tool for model simplification. By examining the posterior distributions of the asymmetric CS-MA and DP-MA models we find that the QSSA does not necessarily hold for these models (Figure 4.16). That is, the systems function as switches even when the QSSA does not hold. These assumptions, necessary for the reduction of the model, are

therefore not always justified in this case.

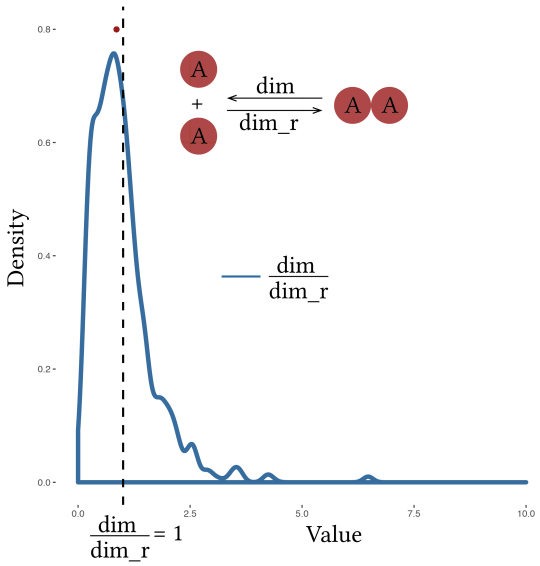

Figure 4.16 : The QSSA that the dimerization reaction (*dim*) is much faster than its reverse (*dim_r*) cannot be justified here. The median of the data (red) lies below the x=1 line (black dashed line) which indicates that in the majority of the particles in the posterior $\frac{dim}{dim\_r} < 1$

4.4.3.1 Multistability in the MA switces

To investigate how the level of abstraction affects switch design principles, I expand the analysis under the assumption of mass action kinetics and stochastic dynamics. The asymmetric CS-MA and DP-MA models are simulated using the Gillespie algorithm (Gillespie 1977).

Ma et al. (2012) found that the stochastic fluctuations in a system involving such a small number of molecules, like the toggle switch, uncovers effects that can not be predicted by the fully deterministic case (Ma et al. 2012). We find that in the stochastic case, both the simple switch, CS-MA , and positive autoregulation switch, DP-MA, are capable of both bistable and tristable behaviour. The fact that tristability can occur in the classical model is consistent with the effect of small molecule numbers; if gene expression remains low, it provides the opportunity for small number effects to be observed, and the third steady state to stabilise (Ma et al. 2012). In order to ensure that the tristable switches found in the stochastic case are truly tristable, I re-sample the posterior distributions and simulate to steady state. If the resulting phase plots are tristable then we know that the posterior truly represents tristability.

As can be seen in Figure 4.17, differences in the parameter values are observed between the bistable and tristable switches, in bothCS-MA and DP-MA models. We find that the simple switch is tristable when dimerisation rate is low and bistable when it is high. The degradation of the dimer proteins must have a low rate for bistability but there are no restraints in the case of the tristable switch. For the case of the DP switch, we find that the rates for dimerisation, degradation and dimer degradation are different for the bistable and tristable behaviours (Figure 4.17). The rate of dimerisation must be low for tristability to occur and large for bistability, as observed for the simple switch. The parameter for protein degradation must be low for tristability whereas there are no constraints for the bistable case. Finally, the parameter for dimer degradation must be low for bistability whereas it has no constraints for tristability, as observed in the simple switch. The design principles for both the CS-MA model and the DP-MA model are summarised in Table 4.4

Table 4.4 Design principles of bistable and tristable switches

	CS-MA		DP-MA	
	Bistable	*Tristable*	*Bistable*	*Tristable*
dimerisation	High	Low	High	Low
protein degradation	-	-	-	Low
dimer degradation	Low	-	Low	-

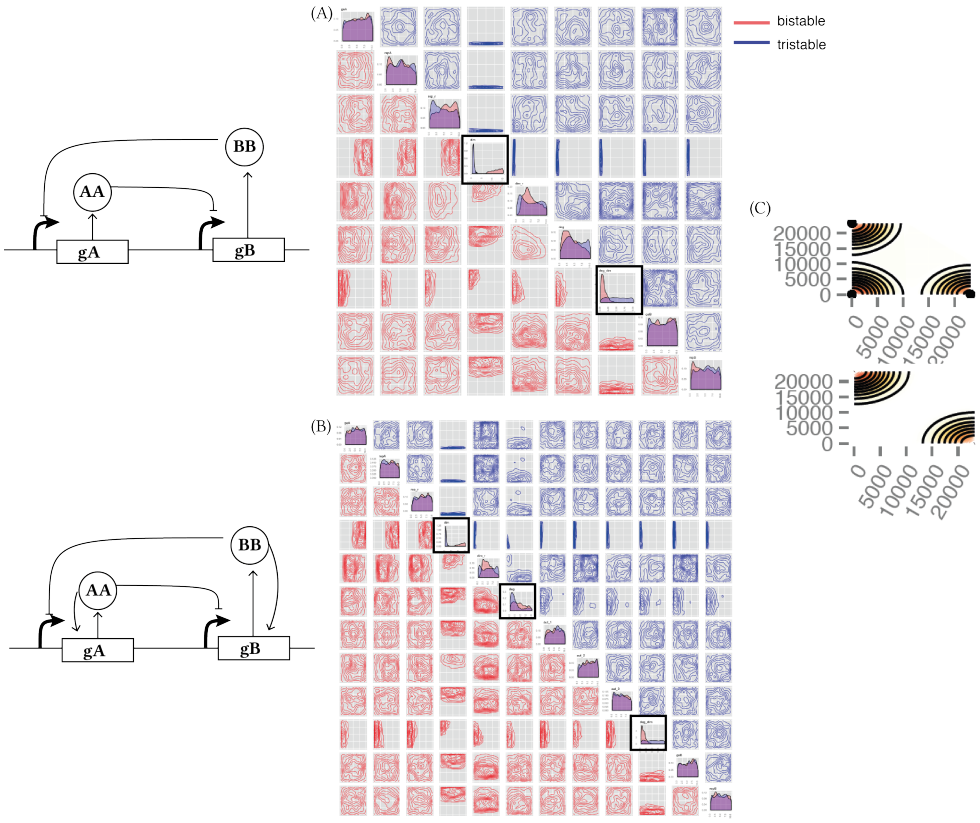

Figure 4.17 : Tristability is possible in the mass action toggle switch models only when simulated stochastically. (A) The simple toggle switch with no autoregulation can be both bistable and tristable. The two posteriors are shown, where the posterior distribution of the bistable switch is shown in red and of the tristable switch in blue. From the posterior distribution we can deduce the the dimerization parameter must be small for tristability to occur but large for bistability. The switch with double positive autoregulation and its posterior distributions for the bistable and tristable case are shown in (B). (C) A sample phase plot of a stochastic tristable and bistable mass action switch.

4.4.3.2 Robustness prior dependence

An important aspect of robustness that I must address is its dependence on the prior distributions. From Equation 2.14 we can expect that the measure for robustness will depend on the size of the prior.

I use the symmetric CS-MA model and Stability Finder to find the posterior distribution that makes this model bistable. This is repeated using the same model, but with larger prior ranges. The priors used are shown in Table 4.5. The posterior distributions are shown in Figure 4.18A and B. We find that for this model, if the prior ranges are very large, the robustness of the model will be much larger. If one of the parameters is able to have a larger value, then the constraints on the rest of the parameters are not necessary any more.

In order to test this observation, I test each parameter separately. All the priors are kept the same as the priors in the very narrow case (Figure 4.18A), except for one. Each run, one of the parameters has priors equal to the priors used in the wide case, Figure 4.18B. The priors are summarised in Table 4.5. Every time the robustness is calculated using the ellipsoid method. Since robustness is only meaningful when used in relation to another model, the robustness of the CS-MA model is compared to the robustness of the DP-MA model used in Figure 4.15. The same posterior distribution of the DP-MA model as used in Figure 4.15 are used every time. For the parameters this model shares with the CS-MA model I used the priors used in the narrow case in Table 4.5, for their shared parameters.

Table 4.5 Priors used for studying the effect of priors to robustness

	Very narrow	Narrow	Wide
ge	5 - 10	1 - 10	1 - 100
rep	2 - 10	1 - 10	1 - 100
rep_r	0 - 5	1 - 10	0 - 10
dim	5 - 10	1 - 10	0 - 10
dim_r	0 - 5	0 - 5	0 - 10
deg	1 - 5	0 - 10	0 - 100
deg_dim	0 - 0.1	0 - 0.5	0 - 1

We can see from Figure 4.18C that when the priors for *ge* are much larger, the Bayes' factor increases significantly. This is due to the fact that when the rest of the parameters are constrained to being within a very narrow range, gene expression must be small for bistability to occur. This has the effect of lower robustness since the prior volume added to the system is much larger than the volume of the

functional region, thus greatly decreasing the robustness.

Robustness of the CS-MA model is increased when the priors for degradation and dimerisation have wider priors (Figure 4.18C). Therefore, if degradation and dimerisation can take larger values, the system becomes more robust.

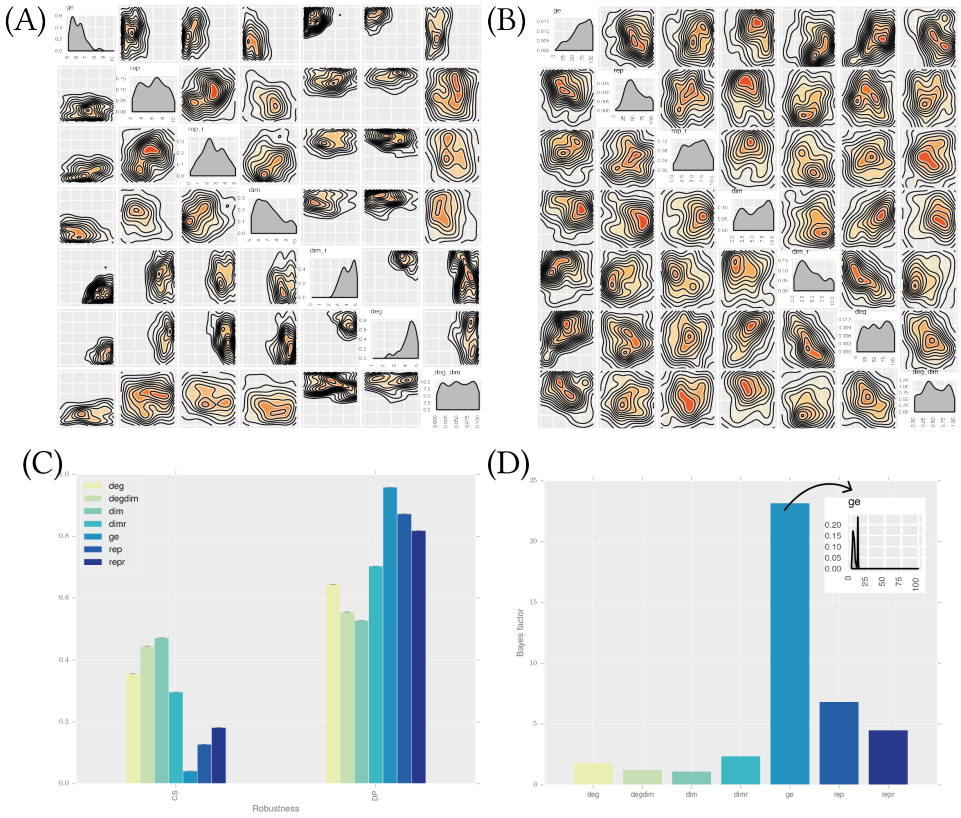

Figure 4.18 : The volume of the priors has an effect on the posterior distribution obtained. (A) The CS-MA model with narrow priors and (B) the CS-MA with wide priors. (C) The increase in robustness seen is due to the gene expression parameter being able to be large.

In Section 4.4.3 we found that the DP-MA is more robust than the CS-MA model. Here I want to test whether this result still remains when the priors of the models are made wider. I change the prior ranges of both models and measure their robustness each time. The results are shown in Figure 4.19A. We find that the robustness measure changed significantly as the priors of the models changed. When both models have very narrow or when both models have wide priors then their robustness measures are very similar. When both models have vary narrow priors the Bayes' factor is equal to 1.32 and when the priors are wide the Bayes' factor is equal to 1.06. In both these cases there is no significant difference in robustness between the two models. When the priors for both models are narrow the Bayes' factor is equal to 2.25. Most notably, when the priors for the CS-MA model are very narrow and the priors of the DP-MA model are narrow, the Bayes' factor is at 9.14.

It is evident that the robustness measure depends on the prior volume. It is therefore useful to think of the Bayes' factor in terms of the difference in the volume of the priors of the models that are being compared. I carry out this analysis for the above priors and the results are shown in Figure 4.19B. Here we see that even though the prior difference is within the same order of magnitude, the Bayes' factor increases significantly. This point corresponds to the case where the priors of CS-MA are very narrow and the priors of DP-MA are narrow, and there is where we see the Bayes' factor between the two models maximise. We can take advantage of this observation to design a switch with double positive autoregulation that is significantly more robust than the Gardner toggle switch. By choosing the parameter values carefully we can maximise the gain of robustness that adding two positive feedback loops gives.

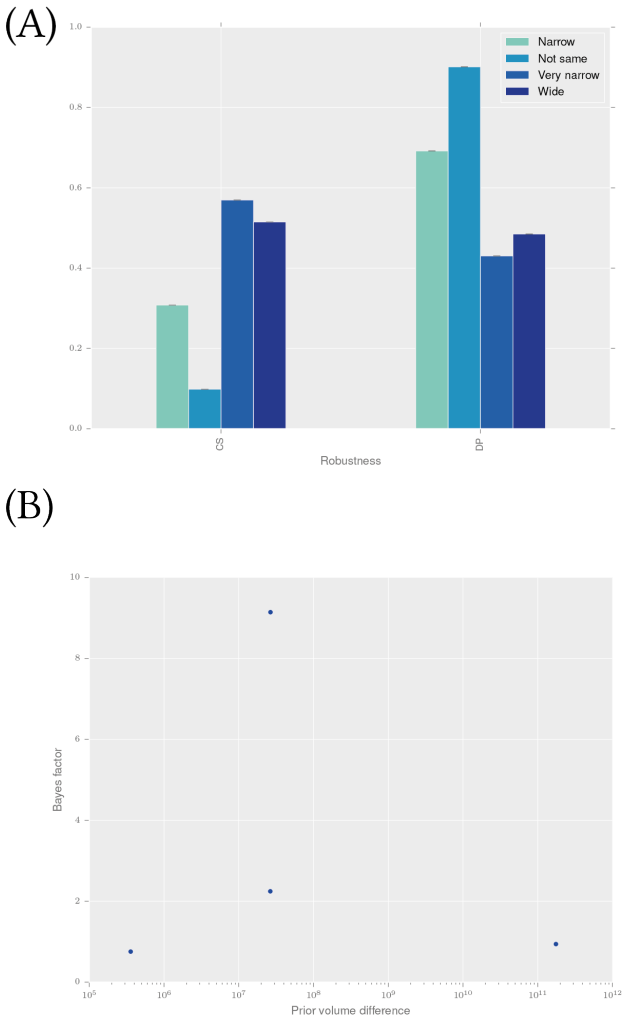

Figure 4.19 : Changing the priors in both models affects the robustness measure. (A) Using different prior ranges for the CS-MA and the DP-MA models yields different robustness for each. (B) The Bayes' factor as a function of the prior volume difference.

4.5 Discussion

Here I developed a novel framework, Stability Finder, that can be used to infer parameter values that can produce a desired behaviour. The novelty in the framework I developed over existing methodology is that complex models can be analyzed assuming both deterministic and stochastic dynamics. I have used Stability Finder to uncover the design principles of a bistable, a tristable and a quadristable switch. I found key parameters that are important in determining the number of steady states a model is capable of. This is critical to the design of novel synthetic switches, where the stability of a system has to be well defined and predictable. A bistable, a tristable or a quadristable switch could each be used for different functions within a synthetic system. Being able to predetermine the stability a system is capable of is vital for the design of new systems. This is especially true when multiple such systems are used together, and the success of the whole system depends on all the parts working as expected.

Tools that can identify parameter regions that give rise to specific behaviours will be key for the success of synthetic biology. In the future, by selecting the system components accordingly, the parameter values can be adjusted *in vivo*. For example, the desired level of gene expression can be accomplished by selecting the appropriate RBS sequence (Salis, Mirsky, & Voigt 2009). Another method to modify the parameter values *in vivo* is to select the promoter to have the strength corresponding to the levels of gene expression and repression desired. Activity of each promoter can be measured and standardised (Kelly et al. 2009) making this process possible. For a system requiring more than one promoter, these can be efficiently selected from a promoter library using a genetic algorithm (Wu, Lee, & Chen 2011). These standardised interchangeable components with known sequence and activity (Kelly et al. 2009; Canton, Labno, & Endy 2008) can be selected and used to construct a desired system and replicate the parameter values found using Stability Finder.

The methodology used here can only be used to study the presence of a given stability and not its absence. If the algorithm is not converging it cannot be concluded that the given model is not capable of the desired stability under these priors. For example, the mass action switches were found to be both bistable and tristable when stochastic effects were taken into account. Using deterministic dynamics the algorithm did not converge using priors within the ranges used in this work. Nevertheless this does not permit the conclusion of absence of tristability in the deterministic classic or double positive mass action switches. The methodology

presented here only permits the interpretation of models that have converged to a given stability.s

The methodology presented here can also be used to study the topology of more complex multistable switches that exist in natural biological systems such as developmental pathways. I also limited this framework to the objective behaviour of a given number of stable steady states. This could be extended to examine systems with a given switching rate or systems robust to a particular set of perturbations, both of which could be of great importance for building more complex genetic circuits.

Importantly I find that the prior distributions used during such an analysis greatly affect the robustness observed. More generally, the assumptions made when building a model can have a significant effect on the predictions made. This is consistent with current understanding (Babtie, Kirk, & Stumpf 2014) and highlight the importance of a programme of experimental work, combined with systems modelling, in order to understand the rules of thumb for abstraction in model based design of synthetic biological systems.

4.6 Summary

In this chapter I discussed the algorithm I developed and demonstrated how it can identify the parameter regions necessary for a model to achieve a given number of stable steady states. I used it to uncover the underlying principles that govern the stability of a given switch.

I first tested Stability Finder on a known switch and then proceeded to apply it to more complex models. I uncovered the design principles that make the Lu switch bistable, tristable or quadristable. I extended the Lu models to a three-node switch and showed how it can achieve 6 steady states.

Furthermore, I built two novel models of the toggle switch which do not use the QSSA and showed that the QSSA cannot be justified in these models. Using these models I studied the effect positive autoregulation has on the robustness of a model. I also studied the effect the priors have on the posteriors and on the robustness of a model. Finally, using stochastic modelling I showed that these switch models are capable of both bistable and tristable behaviour.

In the next chapter I study the genetic toggle switch in the lab and fit the toggle switch models used here to experimental data.

5 Characterising the genetic toggle switch

5.1 Introduction

In this chapter, I aim to study the genetic toggle switch experimentally. This chapter is organised as follows: In the first section I provide an overview of the circuit used and then outline the methods used for the experiments carried out. In the subsequent section I investigate the effect that the switch has on the growth rate of the bacteria. Then I examine the concentrations of the inducers and the time needed to flip the switch.

5.2 Circuit overview

The toggle switch plasmid I used here was provided by Litcofsky et al. (2012). All the switch components were contained in one plasmid, pKDL071. An overview of the plasmid is shown in Figure 5.1A and the sequence given in Appendix (XXX). The circuit consists of two promoters, Ptrc2 and PLtetO-1 (Lutz & Bujard 1997). Ptrc2 is a constitutive promoter, repressible by LacI. PLtetO-1 is also a constitutive promoter, repressible by TetR, as shown in Figure 5.1B. mCherry (Shaner et al. 2004) and GFP (Shimomura, Johnson, & Saiga 1962) are fluorescent proteins, that were added under the control of the same promoters as the repressors, and thus reflect the levels of TetR and LacI in the system. The plasmid contains kanamycin antibiotic resistance and is high copy (ColE1 origin of replication).

This system is capable of two states, GFP high and mCherry high. When IPTG is added to the system, it represses the repression of TetR and mCherry and thus the cells end up in the mCherry high state. When ATc is added to the system, it represses the repression of LacI and GFP and thus the cells end up in the GFP high

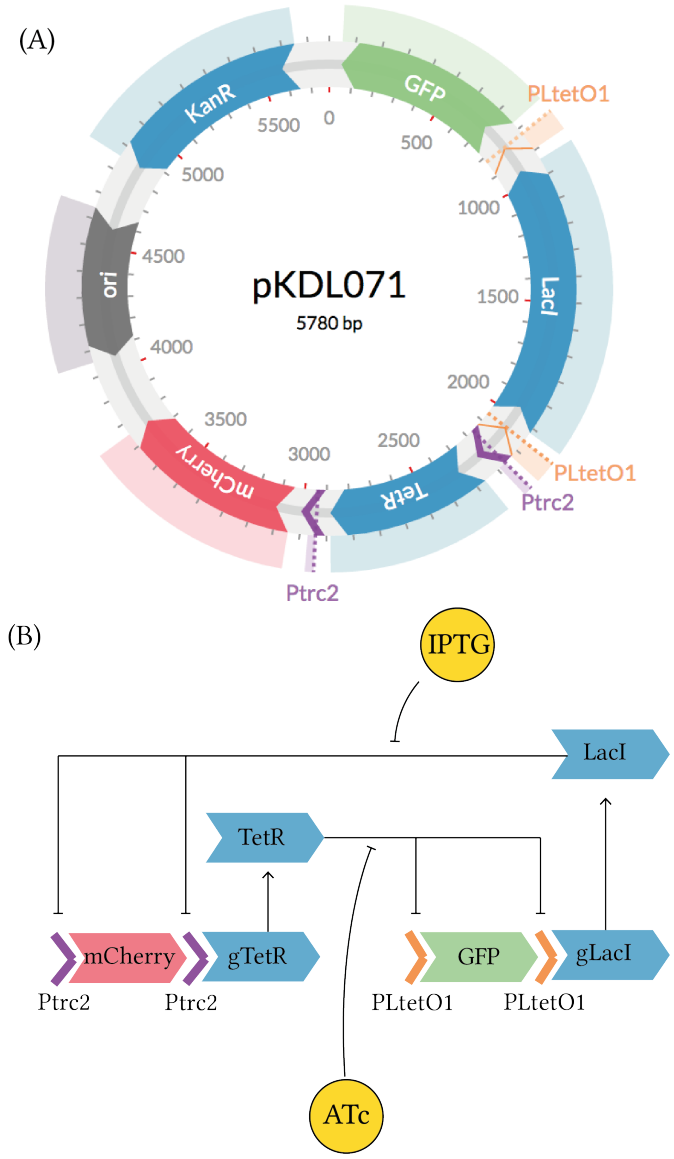

Figure 5.1 : The genetic toggle switch circuit used in this chapter. (A) The plasmid map of pKDL071, the plasmid containing the genetic toggle switch used in Litcofsky et al. (2012) (B) The interactions between each element of the circuit.

state. If no inducer is added to the system it will randomly go to one or the other.

5.3 Methods

The toggle switch plasmid was provided by the James J Collins lab in the form of a stab culture in *E. coli* K-12 MG1655.

5.3.1 *Escherichia coli* culturing conditions

Lysogeny broth (LB) was made by diluting LB in deionized water to a concentration of $25\,\mathrm{g\,L^{-1}}$ and subsequently autoclaved. LB agar plates were made by adding bacteriological agar to the above solution to a concentration of $45\,\mathrm{mg\,mL^{-1}}$ before autoclaving. The solution was then cooled down to 55 °C using a water bath. If antibiotic was required it was added to the correct concentration to the cooled solution. The solution was then aliquoted to plates and left to solidify in room temperature. The plates were stored in the fridge for up to 1 month.

Overnight cultures were made by picking a single colony from a static culture in an agar plate. Each colony was placed in 15 mL Falcon tubes (Fisher Scientific, MA, U.S.A) with 5 mL LB with kanamycin antibiotic at a concentration of $50\,\mathrm{\mu g\,mL^{-1}}$. The tubes were then screwed loosely and taped securely in order to allow for aeration. The falcon tubes were put in an incubator at 37 °C with orbital shaking at 200 rpm for 12-16 hours.

5.3.2 Inducers

Anhydrotetracycline (ATc) solution was made by diluting ATc from Cayman Chemical Company in 100 % ethanol to a concentration of $1\,\mathrm{mg\,mL^{-1}}$. Isopropyl-beta-D-thiogalactopyranoside (IPTG) solution was made by dissolving IPTG in deionized water to a concentration of 1 M. The solution was sterilised by passing the solution through a 0.22 µm syringe filter. Both inducers were stored in 1 mL aliquots at −20 °C.

5.3.3 Glycerol stock preparation

To preserve the transformed cultures long term glycerol stocks were made. 5 mL LB and Kanamycin overnight cultures were made as described in Section 5.3.1. The cultures were kept on ice and 70 % glycerol was added to the cultures in a ratio of glycerol to culture of 1:7. These were aliquoted into cryovials and transferred to a −80 °C freezer for long-term storage.

5.3.4 Revival

For subsequent revival of the frozen cultures, a 1.5 mL eppendorf tube was removed from the −80 °C freezer and put on ice. Small amount was streaked onto an agar plate containing LB and kanamycin. The plates were stored in an incubator at 37 °C overnight. Then the plates were sealed using parafilm and stored at 4 °C for up to two weeks.

5.3.5 Plasmid construction

I constructed two plasmids in order to used them for the flow cytometry experiments. The first plasmid, pSEVA281G contains the promoter PLtetO-1 and GFP and the other, pSEVA281C, contains the promoter Ptrc2 and mCherry from PKDL071, shown in Figure 5.2. I used these two plasmids to determine the appropriate voltages for the lasers that excite GFP and mCherry.

I constructed pSEVA281G by digesting pKDL071 and pSEVA281 using the protocol outlined in Section 5.3.5.2. pSEVA281 is a plasmid backbone containing kanamycin resistance, a high copy origin of replication and a multiple cloning site. I isolated the digested fragments using gel purification (Section 5.3.5.3) and then ligated the isolated fragments (Section 5.3.5.4). I then transfected *Escherichia coli* Dh5α with each plasmid (Section 5.3.5.5). pSEVA281C was constructed via PCR cloning. I carried out a PCR using the pKDL071 plasmid as a template DNA using the protocol outlined in Section 5.3.5.1. I chose the primers so that Ptrc2 and mCherry were copied and a HindIII restriction enzyme recognition sequence added to the fragment. The primers are listed in Appendix (XXX). I purified the amplified DNA using the Qiagen PCR cleanup kit (Qiagen, Crawley, U.K) and then carried out the rest of the cloning procedure as per plasmid pSEVA281G.

Following construction, I isolated each plasmid using the QIAprep Spin Miniprep Kit (Qiagen, Crawley, U.K). I determined plasmid concentration using the Thermo Scientific NanoDrop 1000 Spectrophotometer (Fisher Scientific, MA, U.S.A).

5.3.5.1 Polymerase Chain Reaction

In order to amplify DNA and add the restriction enzyme sites required, I carried out a Polymerase Chain Reaction (PCR) reaction with mutagenic primers. A list of primers can be found in Appendix (XXX). I used the Q5® DNA Polymerase (NEB, MA, U.S.A) with its associated buffer, dNTPs and Q5® enhancer, as specified in

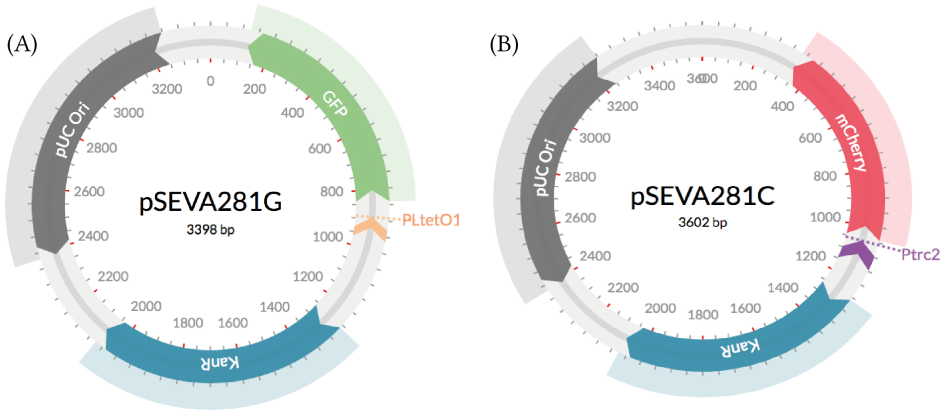

Figure 5.2 : The plasmids used to calibrate GFP and mCherry fluorescence. (A) pSEVA281G plasmid map (B) pSEVA281C plasmid map.

Table 5.1. PCR reactions were run in a T100^{TM} thermal cycler (Bio-Rad Laboratories, Inc., UK) as per the Q5® recommendations, and as outlined in Tables 5.1 and 5.2.

Table 5.1 PCR recipe

Reagent	Final concentration	50 µL reaction
Q5® buffer 5X	1X	10 µL
dNTPs	200 mM each	1 µL
Forward primer	0.5 µM	2.5 µL
Reverse primer	0.5 µM	2.5 µL
Template DNA	2 µg/50 µL	-
Q5® DNA polymerase	0.02 U µL^{-1}	0.5 µL
Q5® enhancer	1X	10 mL
H_2O	-	to 50 µL

Table 5.2 Thermocycling conditions

Step	Cycles	Temperature	Time
Initiation	1	98 °C	30 s
Denaturation		98 °C	10 s
Annealing	30	72 °C	20 s
Extension		72 °C	2 min
Final extension	1	72 °C	2 min
Hold	1	4 °C	∞

5.3.5.2 Digestion

All enzymes, buffers and Bovine Serum Albumin (BSA) were supplied by NEB. I carried out digestion controls by adding H_2O instead of DNA in the digestion reaction. Additionally, during agarose gel electrophoresis uncut plasmid was run alongside the digested plasmid in order to detect the difference.

I set up 2 µg digests by mixing the plasmid with 0.5 µL of each restriction enzyme, 3 µL 10x buffer and 3 µL 10x BSA. I added H_2O to make the reaction to 20 µL. The recipe used is shown in Table 5.3. I placed the reactions in an incubator at 37 °C for 4 hours. Finally, I analysed the solutions using agarose gel electrophoresis (Section 5.3.5.3).

Table 5.3 Digestion recipe

Reagent	Volume
PstI	0.5 µL
HindIII	0.5 µL
NEB Buffer 2.1	2 µL
BSA	0.2 µL
DNA	1 µg
H_2O	to 20 µL

5.3.5.3 Agarose gel electrophoresis

To make a 0.8% agarose gel, I diluted 0.4 g agarose in 50 mL 1X TAE buffer. It was further dissolved by microwaving for 1-3 minutes. I left the solution to cool for 5 minutes and then added 1.5 µL gel red. I prepared gel trays by putting the well comb in place and taping the ends shut. I then poured the solution into the prepared gel trays and left to solidify for 20-30 minutes at room temperature.

I carried out agarose gel electrophoresis by placing the poured gels into the gel tanks. I then flooded the tank with 1X TAE buffer. I prepared the DNA to be analysedI use by adding 4 µL loading dye to 20 µL sample. I also used a negative control with H_2O instead of sample. I prepared the DNA ladder of choice by adding 1 µL H_2O and 1 µL dye to 2 µL ladder. I added each sample to a well by pipetting. The agarose gel was ran at 90 V until the dye was 80% of the way down the gel, approximately 1 hour.

To purify the fragments from the agarose gel, I placed the gel was in a UV box. Using a sterile razor blade, I cut out the desired fragment and placed in a clean

eppendorf tube. I isolated the DNA from the gel using the QIAquick Gel Extraction Kit.

5.3.5.4 Ligation

I used a ratio of 3:1 of insert to recipient plasmid, 1 µL T4® DNA ligase (NEB, MA, U.S.A) and 2 µL ligase buffer. I added H_2O to make the reaction up to 20 µL. The controls I used for each ligation reaction, are shown in Table 5.4. Control 1 is used to detect competent cell viability, control 2 background due to uncut vector, control 3 contamination and control 4 vector re-circularization.

I placed the ligation reactions at 4 °C for 12 hours. I then placed the reactions at 65 °C for 10 minutes to heat inactivate the T4 DNA ligase enzyme. I then carried out a transfection as per Section 5.3.5.5.

Table 5.4 Ligation controls

	Control 1	Control 2	Control 3	Control 4
Vector	Uncut	✓	✓	✗
Insert	✗	✗	✗	✓
Buffer	✓	✓	✓	✓
H_2O	✓	✓	✓	✓
Ligase	✗	✗	✓	✓

5.3.5.5 Transfection

I transfected thermocompetent *E.coli* Dh5αwith the constructed plasmids. I added each ligation reaction to 50 µL of thawed competent cells. The cells were subsequently kept on ice for 30 minutes, then I placed them at a 42 °C water bath for 45 s. I then placed the cells back on ice for 15 minutes. Then I added 500 µL of Super Optimal broth with Catabolite repression (SOC) to each ligation and placed in a 37 °C shaking incubator for 3 hours. I subsequently pipetted 500 µL and 50 µL of each ligation onto petri dishes with LB agar and the appropriate antibiotic. The plates were incubated at 37 °C for 12-16 hours. I used two controls for the transfection protocol, a positive control with no antibiotic in the LB agar and non-transfected cells and a negative control of non-transfected cells and LB agar with antibiotic. These ensure that the cells are viable and not contaminated respectively.

Finally, I counted the number of colonies on each plate. I selected individual colonies from each transfection and grew each separately in 5 mL LB medium for

12-16 hours at 37 °C, 200 rpm. I prepared glycerol stocks from each culture, as per Section 5.3.3.

5.3.5.6 Colony PCR

In order to determine if the fragment was successfully inserted into the vector DNA plasmid, I carried out diagnostic colony PCR. I designed primers that amplified the multiple cloning site of the vector DNA plasmid. These can be found in Appendix (XXX). I made a PCR master mix for the number of colonies to be amplified, 32, with an added 10% to account for pipetting error. I used GoTaq® Flexi DNA polymerase (Promega Corp., WI, U.S.A.) with its associated buffer, dNTPs and $MgCl_2$and H_2O. The recipe for the master mix is shown in Table 5.5.

Table 5.5 Colony PCR master mix recipe

Reagent	Final concentration	Master mix
GoTaq® green Flexi buffer	1X	141 µL
dNTPs	200 mM each	14.1 µL
Forward primer	0.5 µM	1.4 µL
Reverse primer	0.5 µM	1.4 µL
GoTaq® Flexi polymerase	0.02 U μL^{-1}	3.5 µL
$MgCl_2$	1X	42.2 µL
H_2O	-	465 µL

I then added 19 µL from the master mix to each PCR tube. I lifted each of the colonies from the transformation from the agar plate using a 20 µL pipette tip and added it to a PCR mix by mixing. I subsequently used the pipette tip to make a scratch into a clean agar plate, and labelled it. I then carried out a PCR according to GoTaq® Flexi polymerase recommendations, and as shown in Table 5.6.

Table 5.6 Thermocycling conditions for colony PCR

Step	Cycles	Temperature	Time
Cell lysis	1	95 °C	10 minutes
Denaturation		95 °C	30 s
Annealing	35	50 °C	1 minute
Extension		72 °C	1 min
Final extension	1	72 °C	5 min
Hold	1	4 °C	∞

Finally I carried out a diagnostic agarose gel electrophoresis as outlined in Section 5.3.5.3.

5.3.5.7 Sequencing

In order to confirm plasmid identity, all plasmids were sequenced using Source Bioscience, Cambridge UK. I submitted 10 µL of each plasmid DNA at a minimum of 100 ng µL^{-1} as per the requirements. I submitted primer sequences that were manufactured by Source Bioscience. Primers can be found in Appendix (XXX).

5.3.6 Growth rate measurement

I carried out a plate reader analysis in order to measure the growth of *E.coli* over time. I made overnight cultures using the method shown in Section 5.3.1. I then diluted the overnight cultures by a 1:1000 ratio into a 5 mL LB + kanamycin solution. The diluted cultures were grown at 37 °C with shaking at 200rpm for 1 hour. I then further diluted these cultures by a 1:100 ratio. I transferred 200 µl aliquots of the dilutions to a clear bottom, black-walled 96-well plate. I also added wells with only LB and kanamycin in order to be used as blanks. I then sealed the plate using a gas permeable membrane and placed it in BMG FLUOstat OPTIMA plate reader to measure absorbance. I set the plate reader to a constant 37 °C, with 30 seconds orbital shaking at 150 rpm and 4 mm shaking width every ten minutes. Absorbance was measured at 540 nm. Data was exported as a CSV file and analysed using Python.

5.3.7 Flow cytometry

I carried out flow cytometry experiments in order to get fluorescent levels in single cells. Flow cytometry allows us to gather this information for thousands of single cells. Flow cytometry data was exported as FCS files and analysed using the R bioconductor package.

5.3.7.1 Concentration assays

I carried out concentration assays to determine the concentration of each inducer (ATc and IPTG) at which the switch flips. I prepared separate overnight cultures as per Section 5.3.1 with added IPTG at a concentration of 1 mM or added ATc at a concentration of 100 ng mL^{-1} (Litcofsky et al. 2012). I then diluted the cultures by 1:1000 into fresh LB medium with varying concentrations of the opposite inducer

than what the cells were grown in overnight. The concentrations used are shown in Table 5.7. For each concentration I made three replicates cultures.

Table 5.7 Concentrations used for flow cytometry assay

ATc (ng/ml)	IPTG (M)
0.05	1e-7
0.06	6e-7
0.07	1e-6
0.08	6e-6
0.09	1e-5
0.1	1e-3
1.0	0.1

I placed the cultures in an incubator at 37 °C, 200rpm for 5 hours. I then placed the cultures in a centrifuge and spun at 13,000rpm for 5 minutes. I discarded the supernatant and replaced it with 1 mL PBS solution. I used the The BD LSRFortessa^{TM} cell analyzer (Becton, Dickinson and Company) at the St. Mary's Flow Cytometry Core Facility at Imperial College London for flow cytometry analysis. GFP was excited using the 488 nm laser and detected using the 533/30 filter. mCherry was excited using the 561 nm laser and detected using the 620/10 filter. Data was obtained at n=10000 events per experiment.

5.3.7.2 Time course assays

I carried out time course assays to measure the time it takes for the switch to flip to each side. I prepared separate overnight cultures of pKDL071 as per Section 5.3.1 with added IPTG at a concentration of 1 mM or added ATc at a concentration of 100 ng mL^{-1} (Litcofsky et al. 2012). I also made overnight cultures of pSEVA281G and pSEVA281C. I then diluted the cultures by a ratio of 1:1000 into fresh LB medium. I made separate cultures for each time point, in triplicates. For cultures grown overnight in IPTG, I added ATc at a concentration of 100 ng mL^{-1} and for cultures grown overnight in ATc, I added IPTG at a concentration of 1 mM. All cultures were placed at 37 °C, 200rpm incubator. At 30 minutes, 1 hour and then every hour up to 6 hours I carried out flow cytometry to the corresponding cultures. I removed three replicates for each induction from the incubator and placed them in a centrifuge at 13,000rpm for 10 minutes. I discarded the supernatant and replaced it with 1 mL PBS solution. I then analysed these cultures in an Attune^{TM} NxT Flow Cytometer (Thermo Fisher Scientific) at University College London. GFP was excited using the

488 nm laser and detected using the 533/30 filter. mCherry was excited using the 561 nm laser and detected using the 620/10 filter. Data was obtained at n=10000 events per experiment. I used pSEVA281G and pSEVA281C cultures to set the laser voltages and pKDL071 cultures to detect the bacteria population.

5.4 Growth rate investigation

I carried out a growth rate analysis to determine whether the ATc or IPTG added to pKDL071 or pSEVA281G *E. coli* cultures affected the growth of the bacteria. Cultures were grown without any inducer overnight as described in Section 5.3.6. I ran assays for the cultures with and without added inducers. As can be seen in Figure 5.3, there is no difference between the conditions. The addition of either ATc or IPTG does not affect the growth rate of *E. coli* K-12 MG1655. Additionally, ATc does not affect the growth rate of *E. coli* Dh5α. Since the addition of ATc flips the switch to the GFP high state, and IPTG to the mCherry high state, we can also conclude that the growth rate of the chassis is not affected by which side of the switch is in the high state. The growth rate of *E. coli* Dh5αwas consistently lower than that of *E. coli* K-12 MG1655.

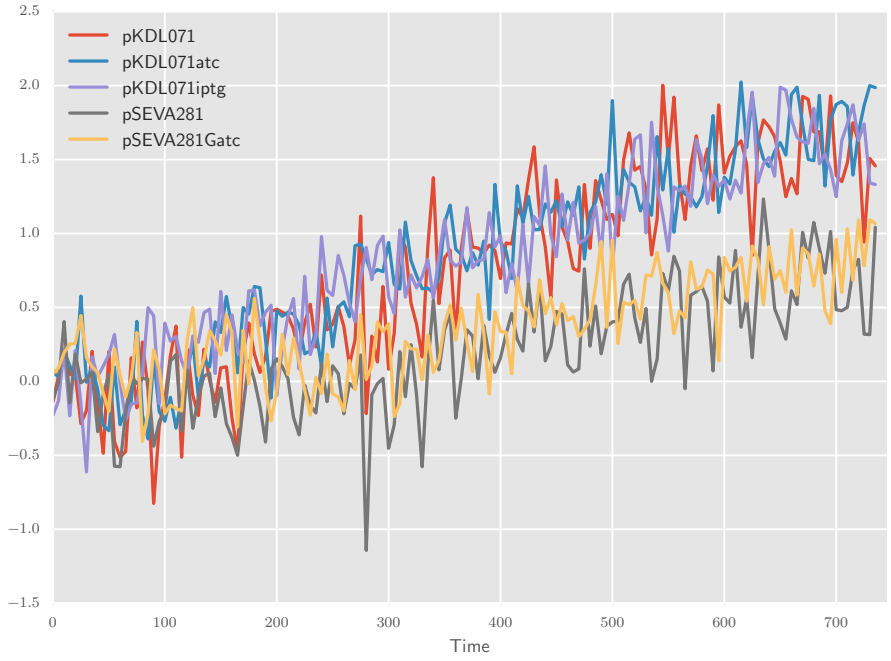

Figure 5.3 : Growth rate analysis of *E. coli* K-12 MG1655 pKDL071 and *E. coli* Dh5αpSEVA281G cultures with and without inducers. The inducers do not affect the growth of the bacteria.

5.5 Toggle switch concentration assays

Here I aim to identify the inducer concentration at which the pKDL071 toggle switch changes state. In order to do that I carry out a concentration assay using flow cytometry, as described in Section 5.3.7.1. As can be seen in Figure 5.4A, during ATc induction the switch flips to a GFP high state when ATc concentration is at $0.09\,\mathrm{ng\,mL^{-1}}$ or higher. We observe a bimodal distribution at concentrations $0.07\,\mathrm{ng\,mL^{-1}}$ and $0.08\,\mathrm{ng\,mL^{-1}}$, which indicates that the switching has begun at these concentrations. Thats why part of the population has switched to the GFP high state but complete switching is not observed until the concentration of ATc is at $0.09\,\mathrm{ng\,mL^{-1}}$. In the case of IPTG induction (Figure 5.4B) we find that the switch flips to the mCherry high state when the concentration of IPTG is higher or equal to 0.001M. A decrease in GFP fluorescence is also observed. We do not observe a bimodal distribution in this case.

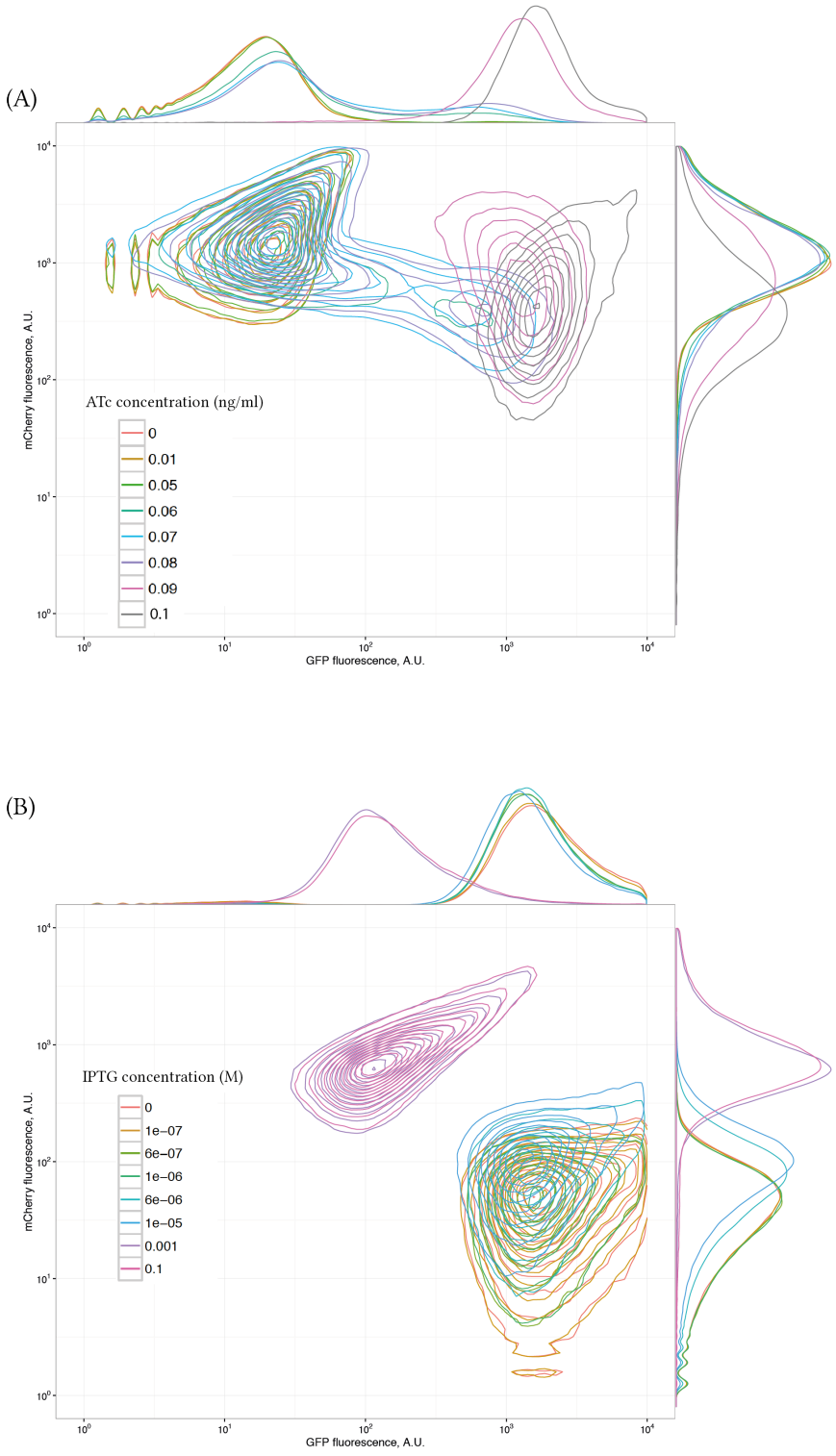

Figure 5.4 : (A) ATc induction at various concentrations (B) IPTG induction at various concentrations.

By taking into account the two induction curves of the switch turning to each high state, we can see the dynamic ranges of pKDL071 in *E.coli*. We can see in Figure 5.5 there is an approximately 100-fold change in fluorescent units during IPTG and ATc induction.

A Hill function was used to model the characterisation curves shown in Figure 5.5. The model used is the following:

$$F = Pmin + (Pmax - Pmin)\frac{\left(\frac{[I]}{Kd}\right)^n}{1 + \left(\frac{[I]}{Kd}\right)^n}, \tag{5.1}$$

where F is the median fluorescent unit and [I] is the concentration of inducer. Pmin and Pmax are the minimum and maximum fluorescence respectively, and Kd and n are the dissociation constant, and Hill coefficient. I fit Hill function models using maximum likelihood estimation to the response curves. The values for parameters Pmin, Pmax, Kd, and n are 8, 1600, 0.1, 1.8 respectively for the ATc induction and 8, 700, 0.08, 2.5 for IPTG induction.

For the case of the ATc induction we observe a sharp switch between the GFP low to the GFP high state, as can be seen in the characterisation curve in Figure 5.5B. This is a clear indication of the bistability of this switch.

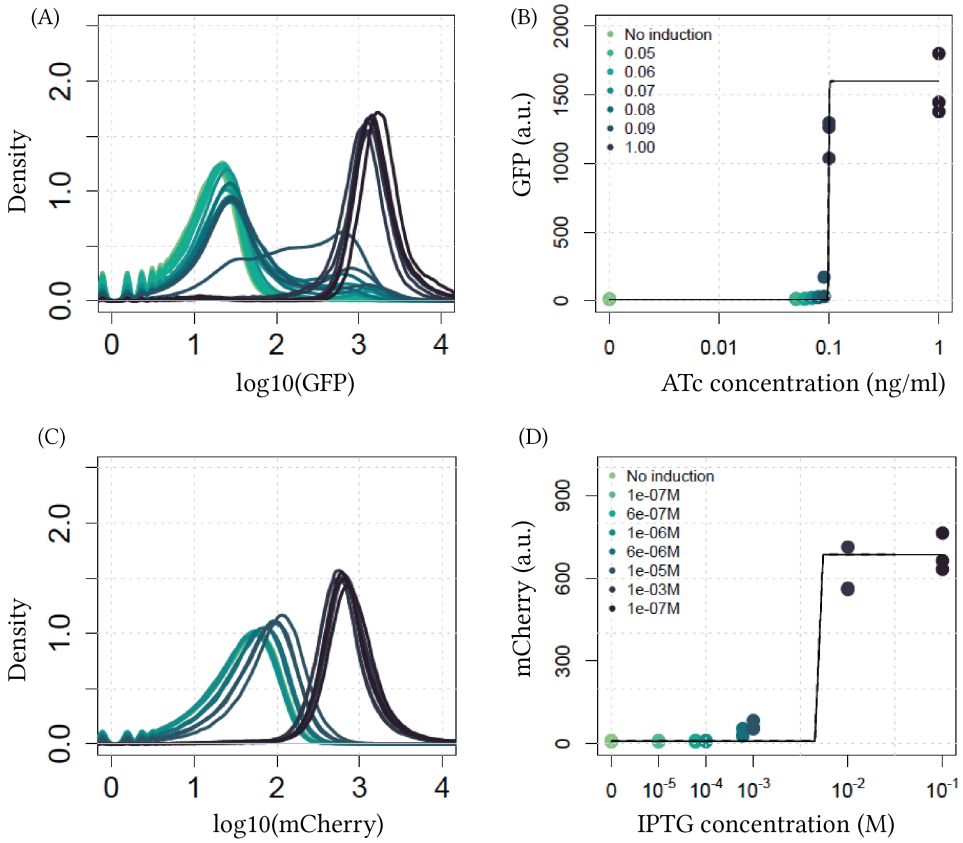

Figure 5.5 : (A, B) ATc induction of pKDL071. (C, D) IPTG induction of pKDL071.

5.6 Toggle switch time course assay

I further analysed the pKDL071 toggle switch by investigating the time it takes for it to switch from one high state to the other. To do that I used the method outlined in Section 5.3.7.2. I obtained separate time courses for the IPTG and ATc inductions.

As can be seen in Figure 5.6 pKDL071 ATc induction begins switching 1 hour after induction. Complete induction is seen at 6 hours. During the IPTG induction (Figure 5.7) we see a bimodal distribution at 4 hours, and induction is complete at 6 hours. We observe that during ATc induction there is an increase in GFP fluorescence and a decrease in mCherry fluorescence, in the case of IPTG induction the increase in mCherry fluorescence is not as prominent. A decrease in GFP fluorescence is observed during IPTG induction.

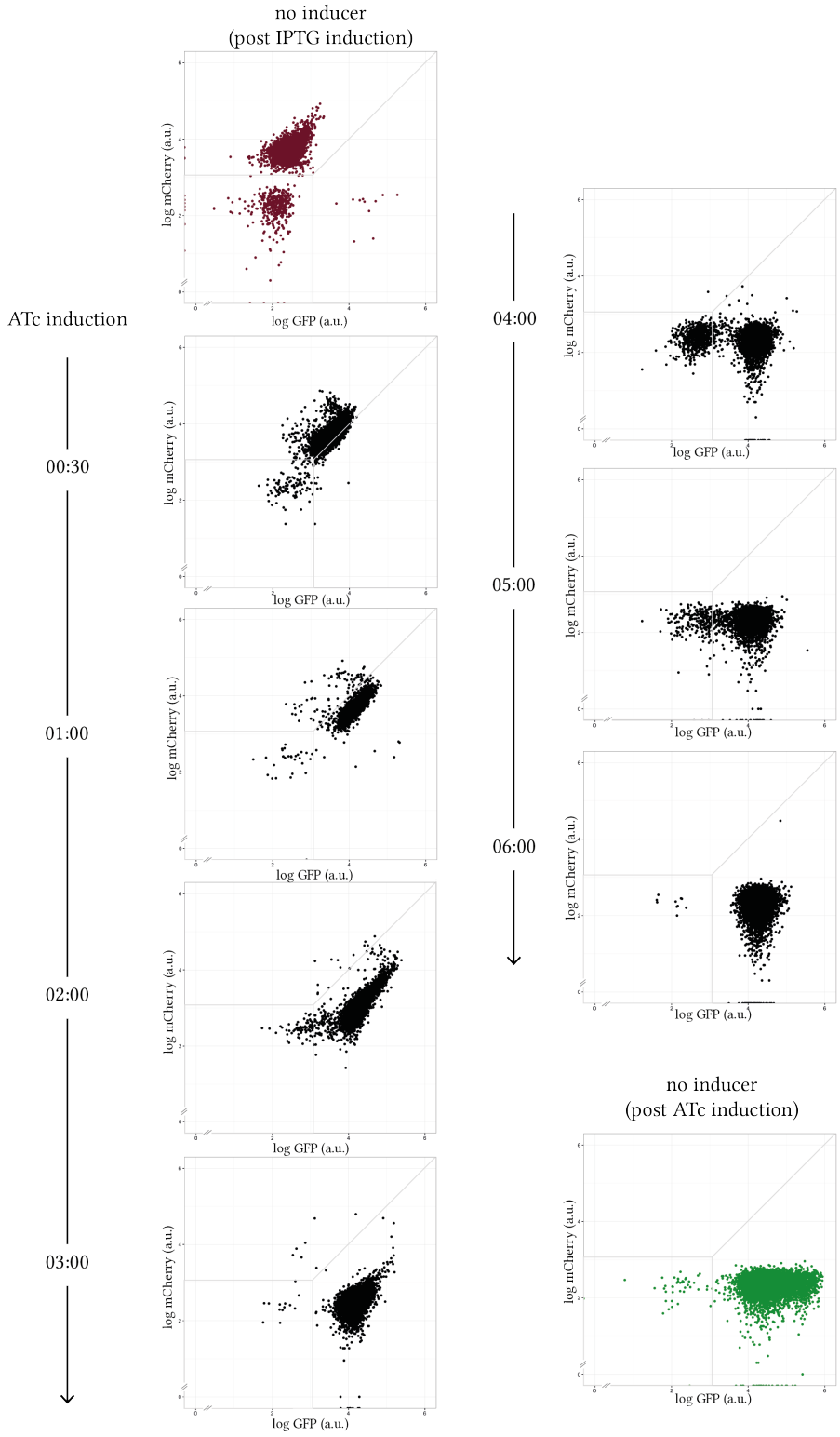

Figure 5.6 : ATc induction of pKDL071 over time

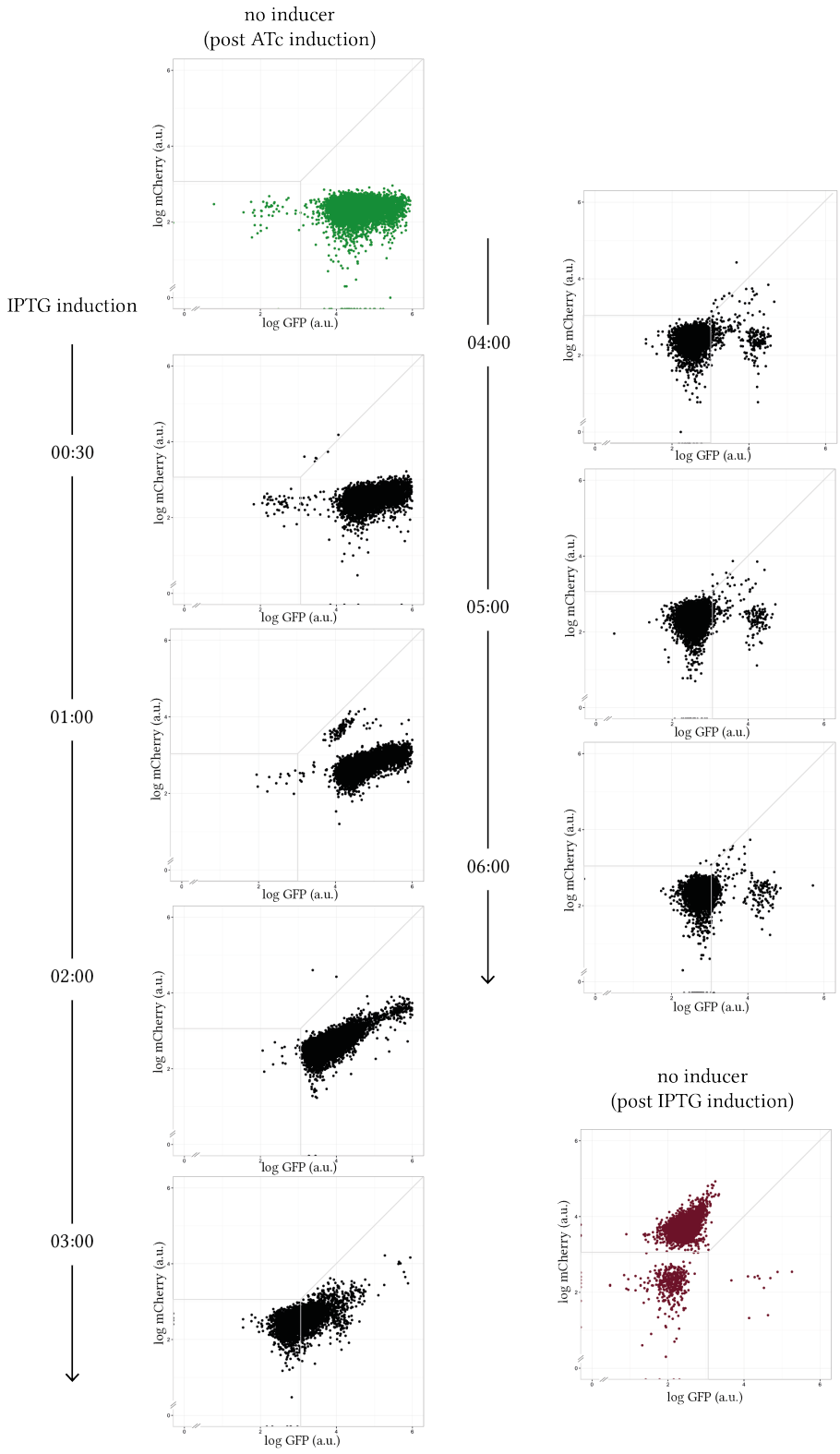

Figure 5.7 : IPTG induction of pKDL071 over time

5.7 Discussion

Here I characterised the genetic toggle switch experimentally. First I study the effect of the two inducers ATc and IPTG on the growth rate of the selected chassis *E. coli* K-12 MG1655. I find that there is no detrimental effect to the bacterium by the inducers. I also find that which state the switch is on has no effect on the growth rate of the bacteria. In order for this toggle switch to be used in a synthetic biology application, it is important that both sides of the switch have an equivalent burden onto the chassis. If one of the steady states creates a larger burden and slows down the growth of the bacteria, this can create an imbalance in the population. If the toggle switch-bearing bacterial population exists in an environment with competing bacteria, for example the gut microbiome, and one of the two states creates a larger burden, this would cause the switch-bearing population to become less competitive compared to the non switch-bearing population. It is therefore crucial that the state of the switch does not affect the competitiveness of the chassis.

I further characterised the switch by determining the minimum inducer concentration necessary to change the state of the switch. I find that for ATc induction, a minimum of $0.09\,\mathrm{ng\,mL^{-1}}$ is required to cause the switch to go to a GFP high state. For IPTG induction I find that a minimum of $0.001\,\mathrm{M}$ is required to flip the switch to an mCherry high state. This information is critical for using this switch in other applications. Both sides of the switch are very sensitive to inducer concentrations, as the concentrations required to observe a change in fluorescence are very small.

Furthermore I find that this toggle switch, pKDL071, is faster to respond to a change in ATc concentration that to a change in IPTG concentration. For IPTG induction we observe a change in fluorescence after 3-4 hours of induction. For ATc induction we can see a difference within an hour of induction. This result is in agreement with Litcofsky et al. (2012). This difference in response times must be taken into account when using the pKDL071 switch for other applications. This difference could be due to maturation times of the fluorescent proteins. Macdonald, Chen, & Mueller (2012) found that mCherry half-maturation time is 150 mins, whereas the GFP variant used here, GFPmut3b has been especially mutated for fast action (Cormack, Valdivia, & Falkow 1996). Cormack, Valdivia, & Falkow (1996) found that whereas wild type GFP is detectable 1-2 hours after induction, GFPmut3b is detectable 8 minutes after induction. This difference could account for the different response times observed here, but further investigation is required.

5.8 Summary

In this chapter I summarised the experiments carried out for the analysis of the genetic toggle switch. I used the pKDL071 plasmid and characterised its switching behaviour over various inducer concentrations and over time. I found the concentration of each inducer necessary to flip the switch as well as the time it takes for the change to be observed. Furthermore, I investigated the effect of the inducers on the growth rate of the chassis and found that they have no effect. In the next chapter I use the data collected in the chapter to fit to the more realistic toggle switch models used in Chapter (XXX).

6 ABC-Flow

6.1 Introduction

6.2 Methods

Algorithm 5 ABC-Flow

1: Read input file
2: **if** ABC-Rejection **then**
3: Sample from priors
4: Simulate model
5: Convert signal to intensity
6: Measure distance to data
7: Reject particles if d $> \epsilon$.
8: **if** number of accepted particles == number of particles **then**
9: Exit
10: **else**
11: Return to step 3.
12: **end if**
13: **end if**

14: **if** ABC-SMC **then**
15: Initialise ϵ
16: population p $\leftarrow 1$
17: **if** p $= 1$ **then**
18: Sample particles (θ) from priors
19: **else**
20: Sample particles from previous population
21: Perturb each particle by $\pm$ half the range of the previous population (j) to obtain new perturbed population (i).
22: **end if**
23: Simulate model
24: Convert signal to intensity
25: Measure distance to data
26: Reject particles if d $> \epsilon$.
27: Calculate weight for each accepted θ
28: $$w_t^{(i)} = \begin{cases} 1, & \text{if } p = 0 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})}, & \text{if } p \geq 0. \end{cases}$$
29: Normalise weights
30: **repeat** steps 17 - 29
31: **until** $\epsilon \leq \epsilon_T$
32: **end if**

6.3 Results

6.3.1 Distance Calculations

Algorithm 6 Distance calculation

1: Grid ← min(data):max(data):ngrid
2: kD = kernel density estimation(data)
3: kS = kernel density estimation(simulations)
4: fD = kD(xx)
5: fS = kS(xx)
6: $d = \sum((fD - fS)^2)$

6.3.2 Comparing 1D and 2D distances

In order to compare the 1D and 2D fitting to the data in ABC-Flow, we must first find out how comparable the distance measures are. Here we simulate two normal distributions, with identical mu and sigma, and calculate the distance between the two using the distance measure used in ABC-Flow. Doing this 1000 times, we then plot the distribution of epsilon. By doing that we can calculate the variance of the epsilon distribution, and find out the error that can be expected when measuring the distance in ABC-Flow. By doing that in 1D and in 2D we can compare the epsilon variances.

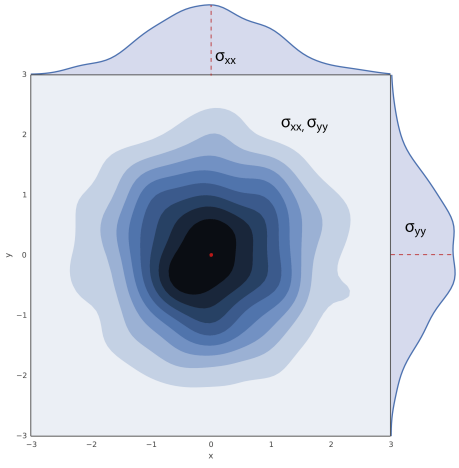

Figure 6.1 Comparing 1D and 2D distributions.

6.3.2.1 Normal distribution

Here, we simulate two normal distributions, with $\mu = 0$ and $\sigma = 1$ 1000 times and measure the distance. In the 2D case, we simulate two multivariate normal distributions, with $\mu = 0$ and covariance=[0.5 0, 0 0.5]. In the 1D case the median of the epsilon distribution was 0.00196 and the variance 0.0016. In the 2D case the median was 3.41e-14 and the variance 1.97e-07.

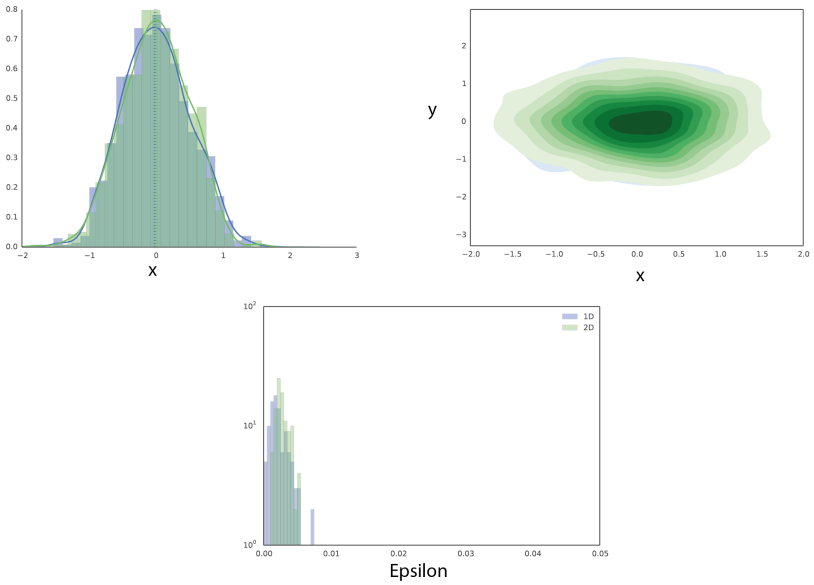

Figure 6.2 Epsilon distribution for 1D (blue) and 2D (green) distances.

Next we study the effect that the number of data points have on the variance of the epsilons.

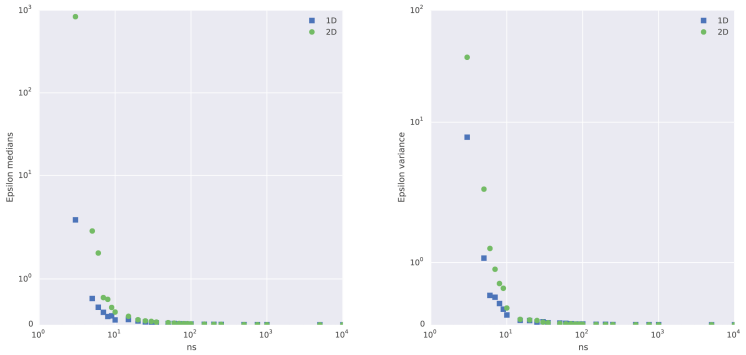

Figure 6.3 Epsilon distribution medians and variance over number of data points.

As the number of data points increases, the distance calculation becomes more precise and more accurate. The median distance approaches zero, and the variance of the epsilon distribution decreases to zero. From Figure 6.3 we conclude that the minimum number of data points that need to be used to calculate the distance between the distributions in ABC-Flow is 100.

The next parameter to be optimised is the bin size used in the distance calculation. In both 1D and 2D, the space is divided into bins, and the distance between corresponding bins in the data and the simulated data is calculated. The overall distance between to distributions is the sum of the distances between corresponding bins.

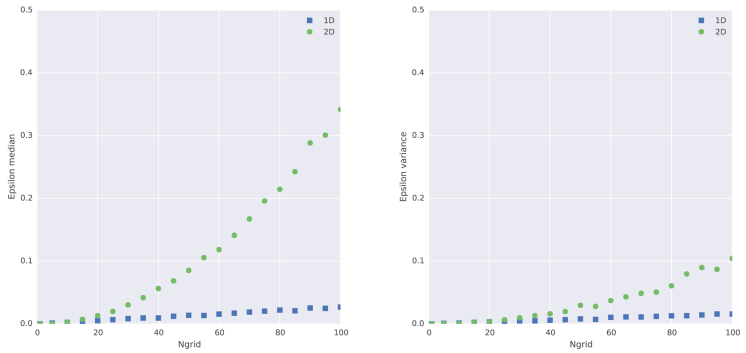

Figure 6.4 Epsilon distribution medians and variance vary with the size of bins used.

We find that the 2D distance is more sensitive to the bin size than the 1D distance. From Figure 6.4 we conclude that for a sample size of 100 data points, the optimal bin size for 1D and 2D distance calculation is 10. The above optimizations have been made by using standard normal distributions, of $\mu = 0$ and $\sigma = 1$. Here we investigate whether the distance calculation depends on the μ and *sigma* of the distributions.

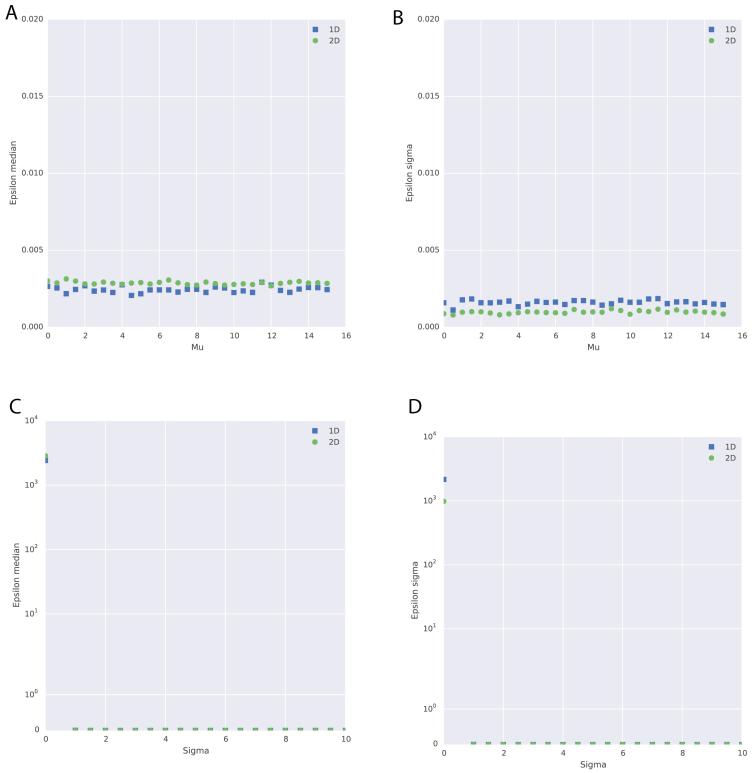

Figure 6.5 The distance calculations do not depend on the μ and σ of the distributions. The median (A) and variance (B) of the epsilons remains constant with increasing μ (C,D) as well as with increasing σ.

Next, by varying the amount of μ and σ by which the two normals differ, we can find out the dynamical range of the epsilons. Whether two distributions are identical or vary by a large amount, we can get an estimate of how much the epsilon will vary from one to the other, both in 1D and 2D.

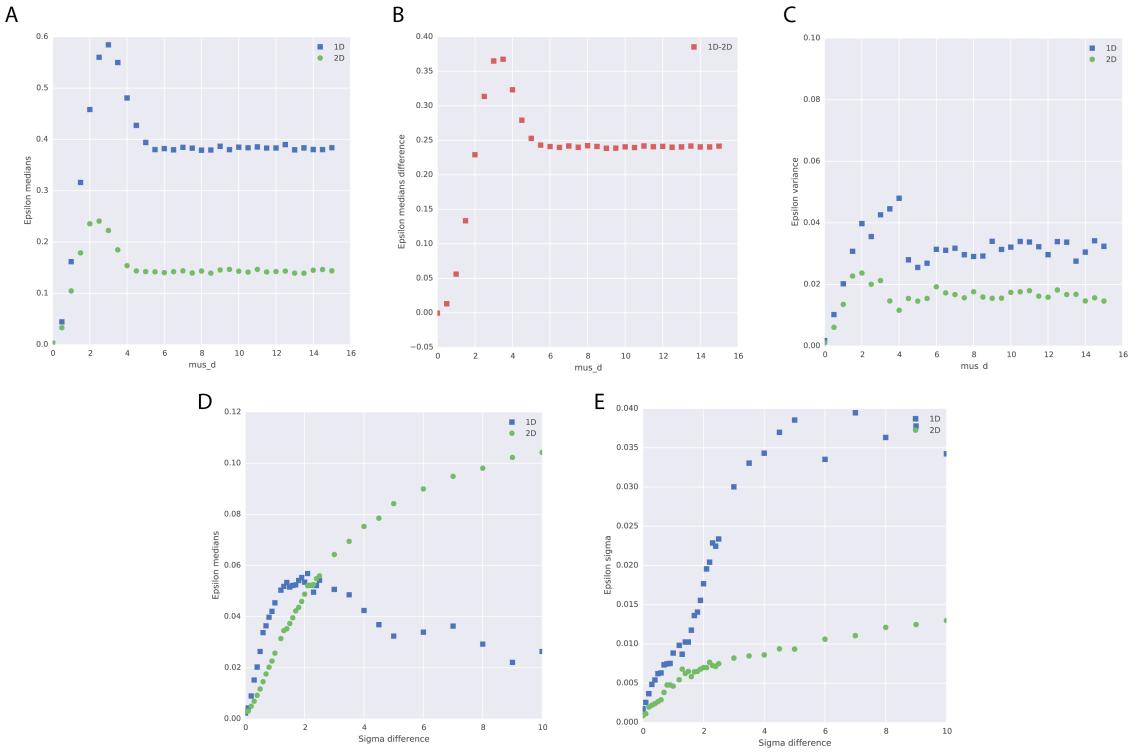

Figure 6.6 (A)The range by which epsilon varies as the difference between the μ of the distributions increases. (B) The difference between the epsilon distribution medians in the 1D and 2D case is not constant with increasing μ difference between the data sets. (C)The variance of the epsilon distributions remains relatively constant with increasing μ. (D)The median of the epsilon distributions varies by a small amount with increasing difference in the σ of the distributions, but the variance (E) remains relatively constant.

From Figure 6.6 we find that as the difference in μ increases the epsilon medians reach a plateau. We find that beyond a difference of 4 in μ, the distance calculation cannot further separate the distances. This can be caused by the fact that when first dividing the space into bins, the range of the data is used to define the grid. If all the simulated data is located outside that grid, the algorithm can no longer distinguish between them, and will only allocate them as outside the range. The variance of the epsilon distributions does not vary significantly with increasing difference in μ. As the difference in σ increases between the distributions, we find that the median in the 2D distance calculation increases but not in the 1D. Note, that the range of the difference in the epsilon medians is small and thus we conclude that differences in the μ of a distribution are much better detected than the differences in the σ. The variance of the epsilon distribution when σ is varied does not change significantly with increasing σ difference.

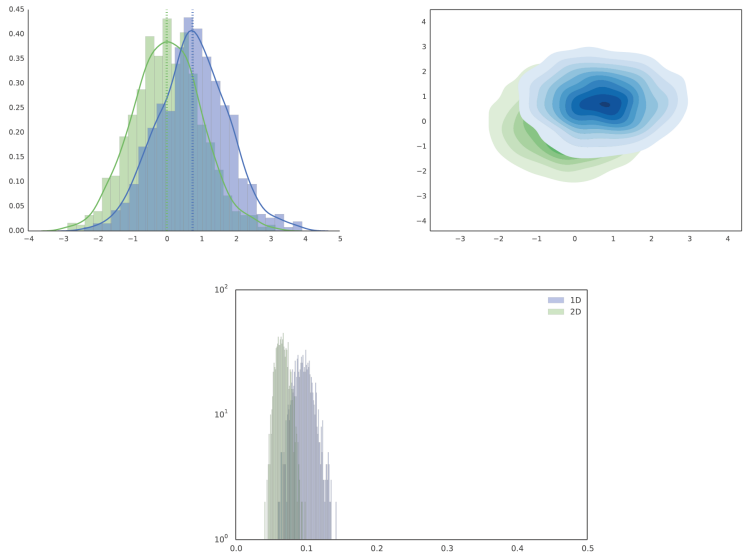

Figure 6.7 The difference in distributions when epsilon median is smaller than 0.1 in 1D and 2D

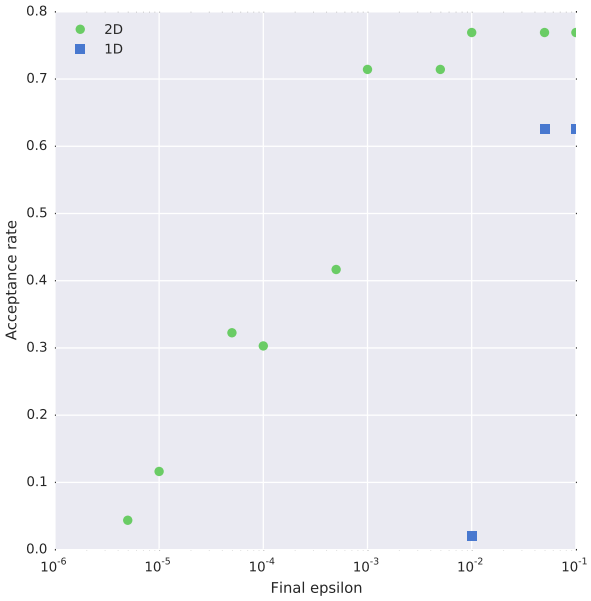

Figure 6.8 Acceptance rate drops rapidly in the 1D case

6.3.2.2 Uniform distribution

We study the epsilon distribution variance in the uniform distribution, [0, 1].

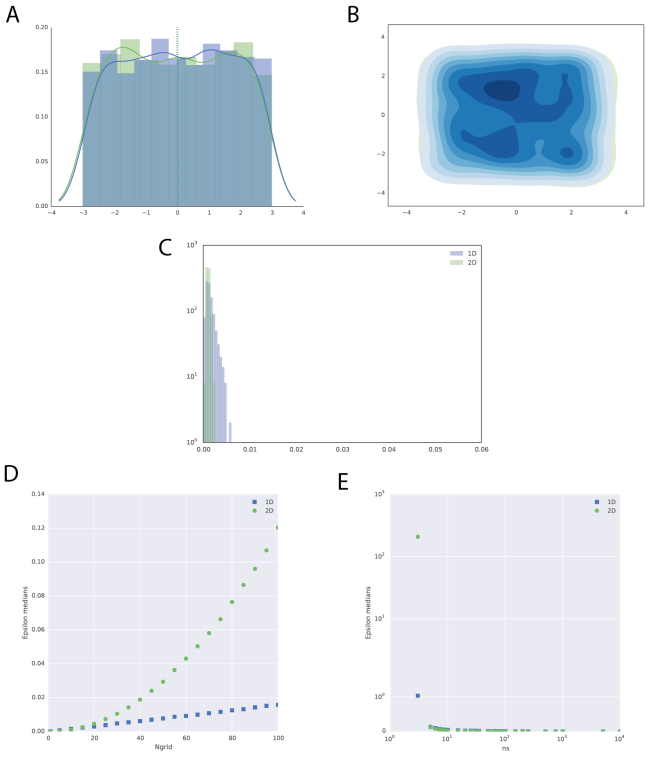

Figure 6.9 Comparing the 1D and 2D distances between uniform distributions. (A) and (B) show samples of the uniform distributions compared in 1D and 2D respectively. (C) The epsilon distributions in the 1D and 2D distances are equivalent (D,E) Epsilon medians and variance varies with how dense the grid is in the 2D case.

6.3.2.3 Comparing uniform and normal distributions

When comparing a normally distributed data set to a uniformly distributed data set, we don't see a great difference between the 1D and 2D epsilons.

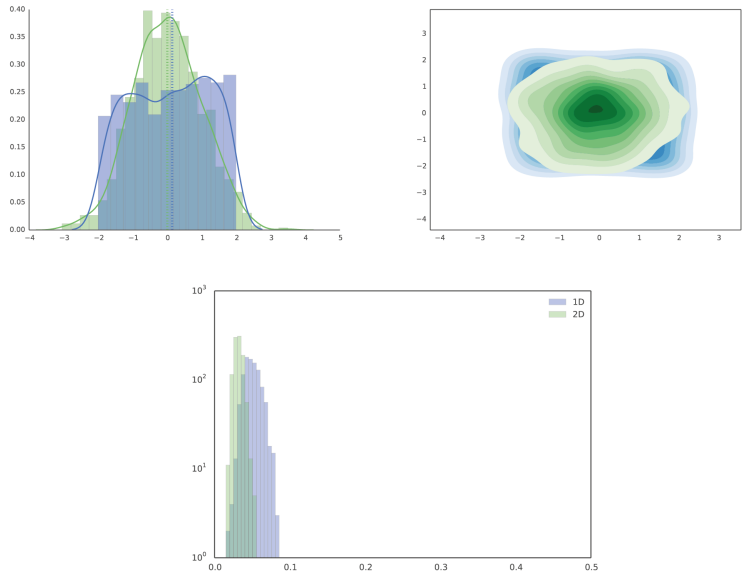

Figure 6.10 Comparing normally distributed data to uniformly distributed simulations.

6.3.2.4 Bimodal distributions

Another type of distribution that is commonly found in Flow cytometry experiments, is a bimodal distribution. Here we test whether the 1D and 2D distances are equivalent when measuring distances in this type of distributions.

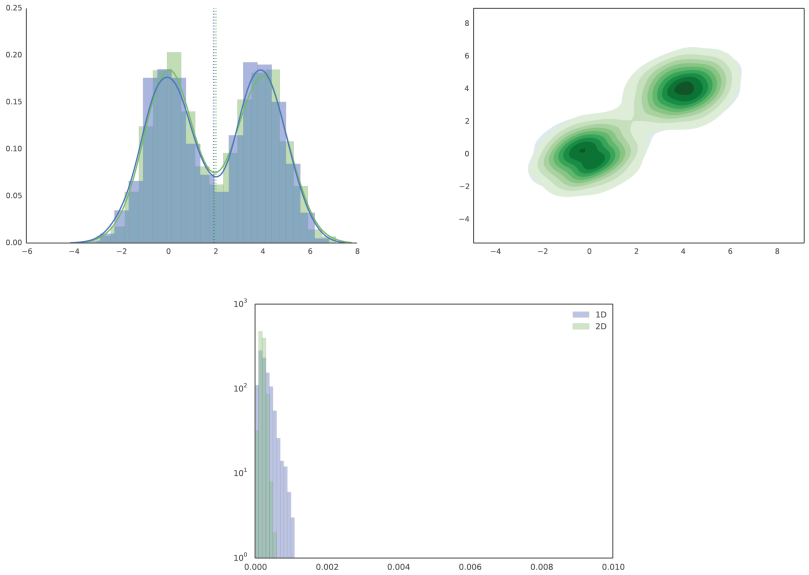

Figure 6.11 Comparing the 1D and 2D distances between bimodal distributions.

We find that the two distances are comparable when dealing with this kind of distibution. Next, we examine how the distance between bimodal distributions with different μ varies in the 1D and 2D cases. We find a similar behaviour like the one observed when comparing normal distributions with increasinglt different μ (Figure 6.6).

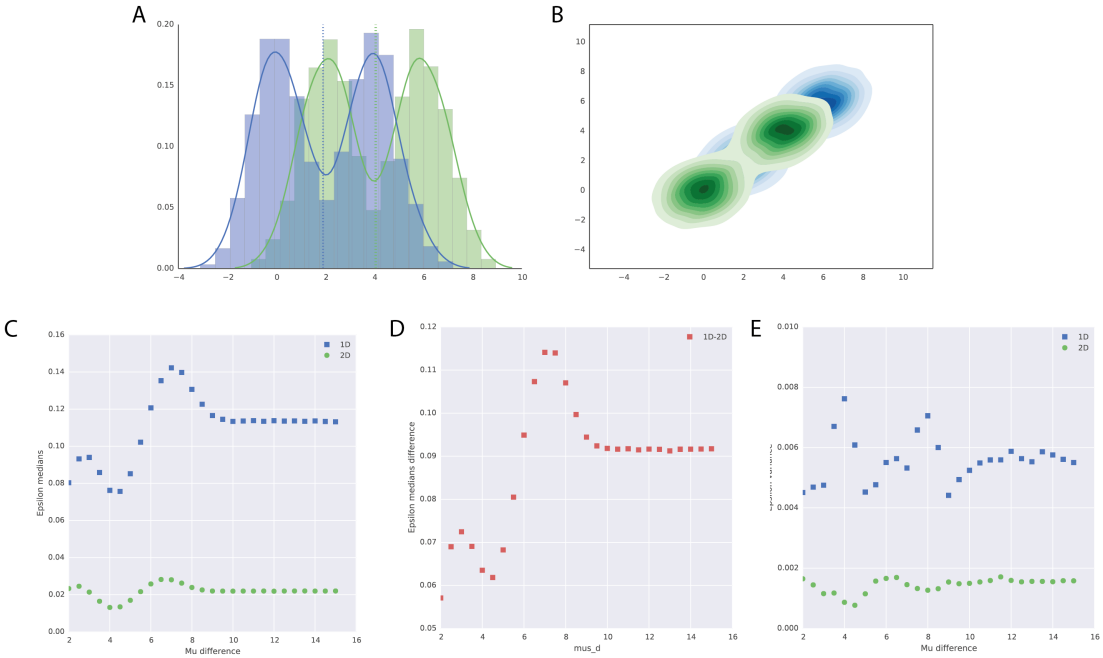

Figure 6.12 Comparing the 1D and 2D distances between bimodal distributions. (A) and (B) show samples of the bimodal distributions compared in 1D and 2D respectively with a μ difference of 4 between simulations and data. (C,D) The range by which epsilon median and variance varies as the difference between the μ of the distributions increases.

6.3.2.5 Comparing bimodal and normal distributions

Finally, we study how the distances perform when comparing a bimodal with a normal distribution. We test the distances by using a bimodal distribution and a series of normal distributions with increasing μ, in 1D and 2D. We find that epsilon is the lowest when the μ of the normal distribution corresponds to the μ of one of the two peaks in the bimodal distribution and the highest when there is no overlap between the distributions.

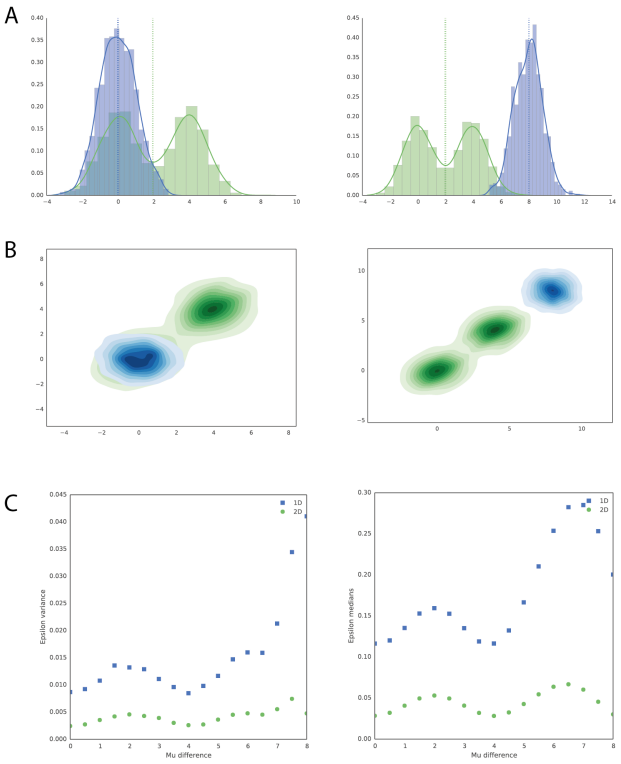

Figure 6.13 Comparing a multimodal to a normal distribution, in 1D and 2D. (A, B) We vary the *mu* of the normal distribution from equal to the μ of the first peak of the bimodal distribution to beyond the range of the bimodal distribution. (C) We find that epsilon median and variance are at the lowest when the μ of the normal distribution is equal to the μ of one of the peaks of the bimodal distribution.

6.3.3 Applying ABC-Flow to simulated Gardner data

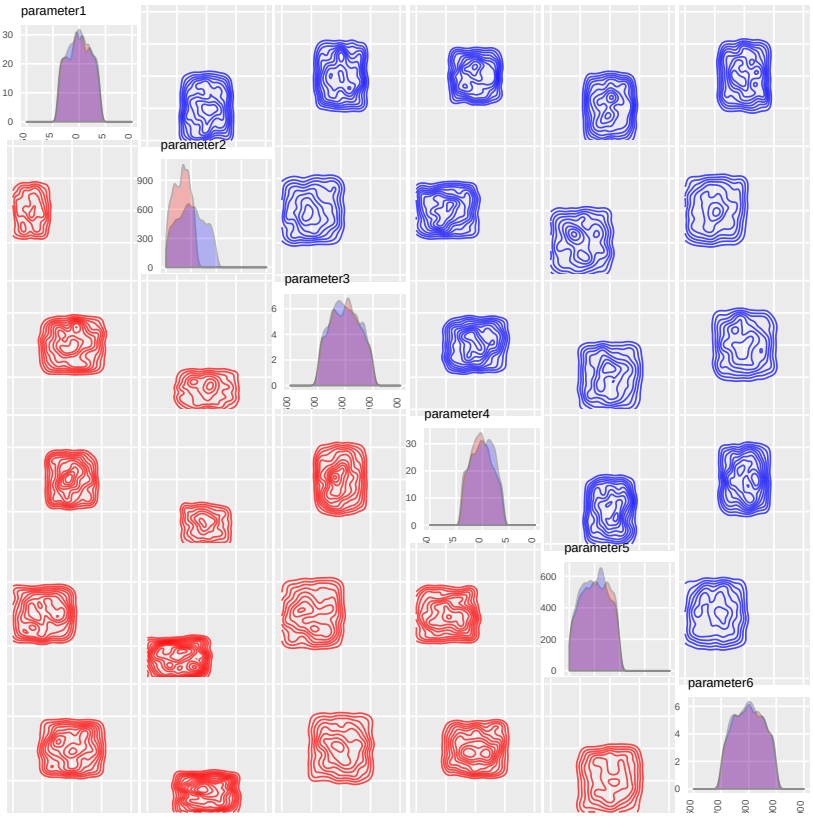

Figure 6.14 Comparing the fit in 1D (red) Vs 2D (blue) in ABC-Flow, when ϵ=0.1.

In order to improve the identifiability of the parameters in the 2D case, we lower the final ϵ: We find increased identifiability for parameters 2 and 5. Next we will

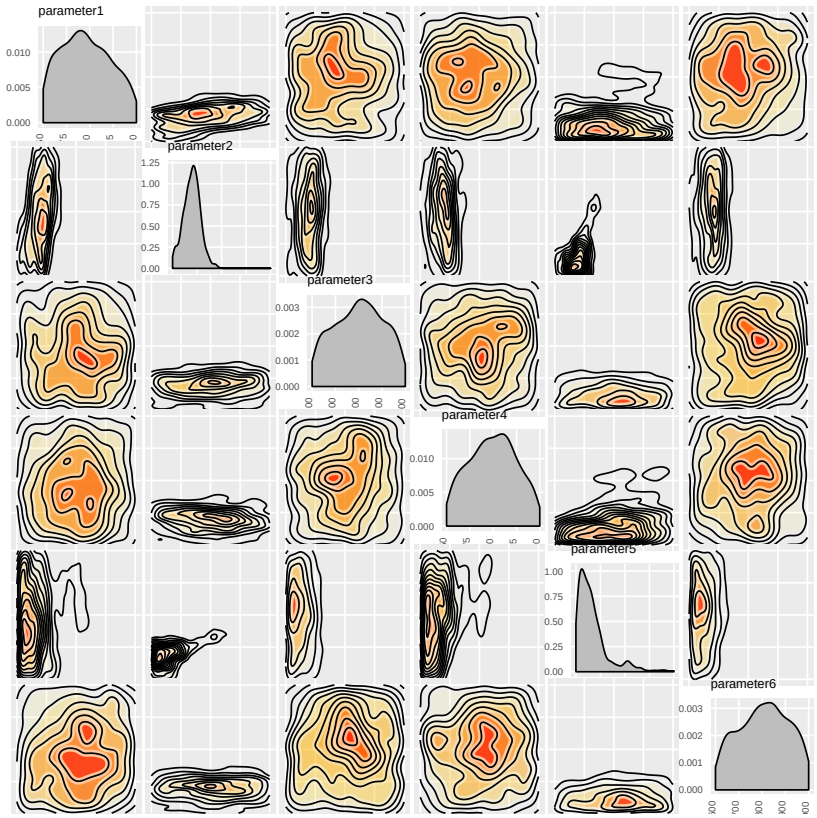

Figure 6.15 ϵ=0.00001.

use this epsilon in the 1D case.

6.3.4 Applying ABC-Flow to experimental toggle switch data

6.3.4.1 ATc induction

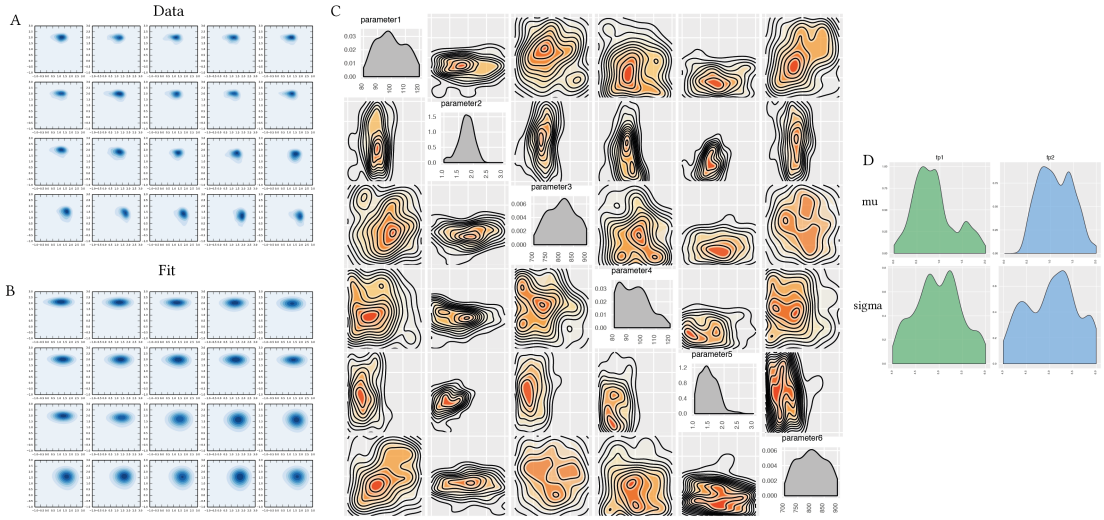

Figure 6.16 Real data 2D fit

6.3.4.2 IPTG induction

6.3.5 Conclusions

7 Designing new switches

7.1 Circuit overview

7.2 Parts

8 Conclusions

8.1 Evaluation

8.2 Future work

Bibliography

Agapakis, C. M. & Silver, P. A. (2009). 'Synthetic biology: exploring and exploiting genetic modularity through the design of novel biological networks'. *Molecular BioSystems* **5**(7), 704.

Alon, U. (2007). *An introduction to systems biology: Design principles of biological circuits*. Chapman and Hall/CRC.

Andrianantoandro, E., Basu, S., Karig, D. K., & Weiss, R. (2014). 'Synthetic biology: new engineering rules for an emerging discipline.' *Molecular systems biology* **2**(1), 2006.0028.

Atkinson, M. R., Savageau, M. A., Myers, J. T., & Ninfa, A. J. (2003). 'Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in Escherichia coli'. *Cell*.

Babtie, A. C., Kirk, P., & Stumpf, M. P. H. (2014). 'Topological sensitivity analysis for systems biology.' *Proceedings of the National Academy of Sciences of the United States of America* **111**(52), 18507–18512.

Banaji, M. & Craciun, G. (2010). 'Graph-theoretic criteria for injectivity and unique equilibria in general chemical reaction systems'. *Advances in Applied Mathematics* **44**(2), 168–184.

Barkai, N. & Leibler, S. (1997). 'Robustness in simple biochemical networks.' *Nature* **387**(6636), 913–917.

Barnes, C. P., Silk, D., Sheng, X., & Stumpf, M. P. H. (2011). 'Bayesian design of synthetic biological systems.' *Proceedings of the National Academy of Sciences of the United States of America* **108**(37), 15190–15195.

Basu, S., Mehreja, R., Thiberge, S., Chen, M.-T., & Weiss, R. (2004). 'Spatiotemporal control of gene expression with pulse-generating networks.' *Proceedings of the National Academy of Sciences of the United States of America* **101**(17), 6355–6360.

Batt, G., Yordanov, B., Weiss, R., & Belta, C. (2007). 'Robustness analysis and tuning of synthetic gene networks.' *Bioinformatics (Oxford, England)* **23**(18), 2415–2422.

Becskei, A. & Serrano, L. (2000). 'Engineering stability in gene networks by autoregulation.' *Nature* **405**(6786), 590–593.

Biancalani, T. & Assaf, M. (2015). 'Genetic Toggle Switch in the Absence of Cooperative Binding: Exact Results'. *Physical review letters*.

Brandman, O., Ferrell, J. E., Li, R., & Meyer, T. (2005). 'Interlinked fast and slow positive feedback loops drive reliable cell decisions.' *Science* **310**(5747), 496–498.

Canton, B., Labno, A., & Endy, D. (2008). 'Refinement and standardization of synthetic biological parts and devices.' *Nature Biotechnology* **26**(7), 787–793.

Chen, B.-S., Chang, C.-H., & Lee, H.-C. (2009). 'Robust synthetic biology design: stochastic game theory approach.' *Bioinformatics (Oxford, England)* **25**(14), 1822–1830.

Cherry, J. L. & Adler, F. R. (2000). 'How to make a biological switch.' *Journal of Theoretical Biology* **203**(2), 117–133.

Chickarmane, V., Enver, T., & Peterson, C. (2009). 'Computational modeling of the hematopoietic erythroid-myeloid switch reveals insights into cooperativity, priming, and irreversibility.' *PLoS Computational Biology* **5**(1), e1000268–e1000268.

Cinquin, O. & Demongeot, J. (2005). 'High-dimensional switches and the modelling of cellular differentiation'. *Journal of Theoretical Biology* **233**(3), 391–411.

Conradi, C., Flockerzi, D., Raisch, J., & Stelling, J. (2007). 'Subnetwork analysis reveals dynamic features of complex (bio)chemical networks'. *PNAS* **104**(49), 19175–19180.

Cormack, B. P., Valdivia, R. H., & Falkow, S. (1996). 'FACS-optimized mutants of the green fluorescent protein (GFP)'. *Gene* **173**(1), 33–38.

De Jong, H. (2002). 'Modeling and simulation of genetic regulatory systems: a literature review.' *Journal of Computational Biology* **9**(1), 67–103.

Deans, T. L., Cantor, C. R., & Collins, J. J. (2007). 'A Tunable Genetic Switch Based on RNAi and Repressor Proteins for Regulating Gene Expression in Mammalian Cells'. *Cell* **130**(2), 363–372.

Ellis, T., Wang, X., & Collins, J. J. (2009). 'Diversity-based, model-guided construction of synthetic gene networks with predicted functions.' *Nature Biotechnology* **27**(5), 465–471.

Feliu, E. & Wiuf, C. (2013). 'A computational method to preclude multistationarity in networks of interacting species.' *Bioinformatics (Oxford, England)* **29**(18), 2327–2334.

Ferrell Jr, J. E. (2002). 'Self-perpetuating states in signal transduction: positive feedback, double-negative feedback and bistability'. *Current opinion in cell biology* **14**(2), 140–148.

Friedland, A. E., Lu, T. K., Wang, X., Shi, D., Church, G., & Collins, J. J. (2009). 'Synthetic gene networks that count.' *Science* **324**(5931), 1199–1202.

Fukunaga, K. (2013). *Introduction to Statistical Pattern Recognition*. Academic Press.

Fung, E., Wong, W. W., Suen, J. K., Bulter, T., Lee, S. G., & Liao, J. C. (2005). 'A synthetic gene–metabolic oscillator '. *Nature* **435**(7038), 118–122.

Gardner, T. S., Cantor, C. R., & Collins, J. J. (2000). 'Construction of a genetic toggle switch in Escherichia coli'. *Nature* **403**(6767), 339–342.

Ghaffarizadeh, A., Flann, N. S., & Podgorski, G. J. (2014). 'Multistable switches and their role in cellular differentiation networks.' *BMC Bioinformatics* **15 Suppl 7**, S7.

Gillespie, D. T. (1977). 'Exact Stochastic Simulation of Coupled Chemical-Reactions'. *Journal of Physical Chemistry* **81**(25), 2340–2361.

Guantes, R. & Poyatos, J. F. (2008). 'Multistable decision switches for flexible control of epigenetic differentiation.' *PLoS Computational Biology* **4**(11), e1000235.

Hafner, M., Koeppl, H., Hasler, M., & Wagner, A. (2009). ''Glocal' Robustness Analysis and Model Discrimination for Circadian Oscillators'. *PLoS Computational Biology* **5**(10), e1000534.

Ham, T. S., Lee, S. K., Keasling, J. D., & Arkin, A. P. (2008). 'Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory'. *PLoS ONE* **3**(7), e2815.

Heinemann, M. & Panke, S. (2006). 'Synthetic biology–putting engineering into biology'. *Bioinformatics (Oxford, England)* **22**(22), 2790–2799.

Holtz, W. J. & Keasling, J. D. (2010). 'Engineering Static and Dynamic Control of Synthetic Pathways'. *Cell* **140**(1), 19–23.

Isaacs, F. J., Hasty, J., Cantor, C. R., & Collins, J. J. (2003). 'Prediction and measurement of an autoregulatory genetic module'. *Proceedings of the National Academy of Sciences of the United States of America* **100**(13), 7714–7719.

Kaplan, D. & Glass, L. (1995). *Understanding nonlinear dynamics*. Springer-Verlag.

Kelly, J. R., Rubin, A. J., Davis, J. H., Ajo-Franklin, C. M., Cumbers, J., Czar, M. J., de Mora, K., Glieberman, A. L., Monie, D. D., & Endy, D. (2009). 'Measuring the activity of BioBrick promoters using an in vivo reference standard.' *Journal of Biological Engineering* **3**(1), 4–4.

Khalil, A. S. & Collins, J. J. (2010). 'Synthetic biology: applications come of age'. *Nature Publishing Group* **11**(5), 367–379.

Kim, J., Bates, D. G., Postlewaite, I., Ma, L., & Iglesias, P. A. (2006). 'Robustness analysis of biochemical network models'. *Systems biology*.

Kitano, H. (2007). 'Towards a theory of biological robustness'. *Molecular systems biology* **3**.

Kobayashi, H., Kaern, M., Araki, M., Chung, K., Gardner, T. S., Cantor, C. R., & Collins, J. J. (2004). 'Programmable cells: interfacing natural and engineered gene networks.' *Proceedings of the National Academy of Sciences of the United States of America* **101**(22), 8414–8419.

Konopka, A. (2007). *Systems Biology, Principles, methods and concepts*. CRC Press.

Kramer, B. P., Viretta, A. U., Daoud-El-Baba, M., Aubel, D., Weber, W., & Fussenegger, M. (2004). 'An engineered epigenetic transgene switch in mammalian cells.' *Nature Biotechnology* **22**(7), 867–870.

Liepe, J., Kirk, P., Filippi, S., Toni, T., Barnes, C. P., & Stumpf, M. P. H. (2014). 'A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation.' *Nature Protocols* **9**(2), 439–456.

Lipshtat, A., Loinger, A., Balaban, N. Q., & Biham, O. (2006). 'Genetic toggle switch without cooperative binding.' *Physical review letters* **96**(18), 188101.

Litcofsky, K. D., Afeyan, R. B., Krom, R. J., Khalil, A. S., & Collins, J. J. (2012). 'Iterative plug-and-play methodology for constructing and modifying synthetic gene networks.' *Nature Methods* **9**(11), 1077–1080.

Lloyd, S. P. (1982). 'Least squares quantization in PCM'. *Information Theory*.

Loinger, A., Lipshtat, A., Balaban, N. Q., & Biham, O. (2007). 'Stochastic simulations of genetic switch systems'. *Physical Review E*.

Lu, M., Onuchic, J., & Ben-Jacob, E. (2014). 'Construction of an Effective Landscape for Multistate Genetic Switches'. *Physical review letters* **113**(7), 078102.

Lu, T. K., Khalil, A. S., & Collins, J. J. (2009). 'Next-generation synthetic gene networks'. *Nature Biotechnology* **27**(12), 1139–1150.

Lutz, R. & Bujard, H. (1997). 'Independent and tight regulation of transcriptional units in Escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements.' *Nucleic Acids Research* **25**(6), 1203–1210.

Ma, R., Wang, J., Hou, Z., & Liu, H. (2012). 'Small-number effects: a third stable state in a genetic bistable toggle switch'. *Physical review letters* **109**(24), 248107.

Macdonald, P. J., Chen, Y., & Mueller, J. D. (2012). 'Chromophore maturation and fluorescence fluctuation spectroscopy of fluorescent proteins in a cell-free expression system.' *Analytical Biochemistry* **421**(1), 291–298.

Marjoram, P., Molitor, J., Plagnol, V., & Tavare, S. (2003). 'Markov chain Monte Carlo without likelihoods.' *Proceedings of the National Academy of Sciences of the United States of America* **100**(26), 15324–15328.

McKay, M. D., Beckman, R. J., & Conover, W. J. (2000). 'A comparison of three methods for selecting values of input variables in the analysis of output from a computer code'. *Technometrics* **42**(1), 55–61.

Niwa, H., Toyooka, Y., Shimosato, D., Strumpf, D., Takahashi, K., Yagi, R., & Rossant, J. (2005). 'Interaction between Oct3/4 and Cdx2 Determines Trophectoderm Differentiation'. *Cell* **123**(5), 13–13.

Pedersen, M. G., Bersani, A. M., & Bersani, E. (2007). 'Quasi steady-state approximations in complex intracellular signal transduction networks – a word of caution'. *Journal of Mathematical Chemistry* **43**(4), 1318–1344.

Prill, R. J., Iglesias, P. A., & Levchenko, A. (2005). 'Dynamic properties of network motifs contribute to biological network organization.' *PLoS Biology* **3**(11), e343–e343.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., & Feldman, M. W. (1999). 'Population growth of human Y chromosomes: A study of Y chromosome microsatellites'. *Molecular Biology and Evolution* **16**(12), 1791–1798.

Salis, H. M., Mirsky, E. A., & Voigt, C. A. (2009). 'Automated design of synthetic ribosome binding sites to control protein expression.' *Nature Biotechnology* **27**(10), 946–950.

Shaner, N. C., Campbell, R. E., Steinbach, P. A., Giepmans, B. N. G., Palmer, A. E., & Tsien, R. Y. (2004). 'Improved monomeric red, orange and yellow fluorescent proteins derived from Discosoma sp. red fluorescent protein.' *Nature Biotechnology* **22**(12), 1567–1572.

Shimomura, O., Johnson, F. H., & Saiga, Y. (1962). 'Extraction, purification and properties of aequorin, a bioluminescent protein from the luminous hydromedusan, Aequorea.' *Journal of Cellular and Comparative Physiology* **59**, 223–239.

Shinar, G. & Feinberg, M. (2010). 'Structural Sources of Robustness in Biochemical Reaction Networks'. *Science* **327**(5971), 1389–1391.

Sisson, S. A., Fan, Y., & Tanaka, M. M. 'Sequential Monte Carlo without likelihoods'.

Stelling, J., Sauer, U., Szallasi, Z., Doyle, F. J., & DOYLE, J. (2004). 'Robustness of cellular functions'. *Cell* **118**(6), 675–685.

Stricker, J., Cookson, S., Bennett, M. R., Mather, W. H., Tsimring, L. S., & Hasty, J. (2008). 'A fast, robust and tunable synthetic gene oscillator'. *Nature* **456**(7221), 516–519.

Thomas, R., Thieffry, D., & Kaufman, M. (1995). 'Dynamical behaviour of biological regulatory networks–I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state.' *Bulletin of mathematical biology* **57**(2), 247–276.

Tibshirani, R., Walther, G., & Hastie, T. (2001). 'Estimating the number of clusters in a data set via the gap statistic'. *Journal of the Royal ….*

Tigges, M., Marquez-Lago, T. T., Stelling, J., & Fussenegger, M. (2009). 'A tunable synthetic mammalian oscillator.' *Nature* **457**(7227), 309–312.

Toni, T., Jovanovic, G., Huvet, M., Buck, M., & Stumpf, M. P. H. (2011). 'From qualitative data to quantitative models: analysis of the phage shock protein stress response in Escherichia coli.' *BMC systems biology* **5**, 69–69.

Toni, T., Welch, D., Strelkowa, N., Ipsen, A., & Stumpf, M. P. H. (2009). 'Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems.' *Journal of the Royal Society, Interface / the Royal Society* **6**(31), 187–202.

Walczak, A. M., Onuchic, J. N., & Wolynes, P. G. (2005). 'Absolute rate theories of epigenetic stability'. *Proceedings of the National Academy of Sciences of the United States of America* **102**(52), 18926–18931.

Warren, P. B. & ten Wolde, P. R. (2004). 'Enhancement of the Stability of Genetic Switches by Overlapping Upstream Regulatory Domains'. *Physical review letters* **92**(12), 128101.

Warren, P. B. & ten Wolde, P. R. (2005). 'Chemical models of genetic toggle switches'. *The Journal of Physical Chemistry B* **109**(14), 6812–6823.

Wilkinson, J. D. (2006). *Stochastic modelling for systems biology*. CRC Press.

Woods, M., Leon, M., Perez-Carrasco, R., & Barnes, C. P. (2015). 'A statistical approach reveals designs for the most robust stochastic gene oscillators'. *bioRxiv*, 025056.

Wu, C.-H., Lee, H.-C., & Chen, B.-S. (2011). 'Robust synthetic gene network design via library-based search method'. *Bioinformatics (Oxford, England)* **27**(19), 2700–2706.

Zamora-Sillero, E., Hafner, M., Ibig, A., Stelling, J., & Wagner, A. (2011). 'Efficient characterization of high-dimensional parameter spaces for systems biology.' *BMC systems biology* **5**, 142.

Zhou, Y., Liepe, J., Sheng, X., Stumpf, M. P. H., & Barnes, C. (2011). 'GPU accelerated biochemical network simulation.' *Bioinformatics (Oxford, England)* **27**(6), 874–876.

*

A Appendix

A.1 Ordinary differential equations

A.1.1 CS-MA

$$\frac{\mathrm{d}([\mathrm{A}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = +2\cdot V_{\mathrm{cell}}\cdot(\mathrm{dim\_r}\cdot[\mathrm{A2}]) - 2\cdot V_{\mathrm{cell}}\cdot(\mathrm{dim}\cdot[\mathrm{A}]\cdot[\mathrm{A}]) + V_{\mathrm{cell}}\cdot(\mathrm{geA}\cdot[\mathrm{gA}]) - V_{\mathrm{cell}}$$

$$\frac{\mathrm{d}([\mathrm{gA}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = +V_{\mathrm{cell}}\cdot(\mathrm{rep\_r}\cdot[\mathrm{B2gA}]) - V_{\mathrm{cell}}\cdot(\mathrm{repA}\cdot[\mathrm{gA}]\cdot[\mathrm{B2}])$$

$$\frac{\mathrm{d}([\mathrm{B}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = +2\cdot V_{\mathrm{cell}}\cdot(\mathrm{dim\_r}\cdot[\mathrm{B2}]) - 2\cdot V_{\mathrm{cell}}\cdot(\mathrm{dim}\cdot[\mathrm{B}]\cdot[\mathrm{B}]) + V_{\mathrm{cell}}\cdot(\mathrm{geB}\cdot[\mathrm{gB}]) - V_{\mathrm{cell}}\cdot$$

$$\frac{\mathrm{d}([\mathrm{gB}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = +V_{\mathrm{cell}}\cdot(\mathrm{rep\_r}\cdot[\mathrm{A2gB}]) - V_{\mathrm{cell}}\cdot(\mathrm{repB}\cdot[\mathrm{gB}]\cdot[\mathrm{A2}])$$

$$\frac{\mathrm{d}([\mathrm{A2}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = -V_{\mathrm{cell}}\cdot(\mathrm{dim\_r}\cdot[\mathrm{A2}]) + V_{\mathrm{cell}}\cdot(\mathrm{dim}\cdot[\mathrm{A}]\cdot[\mathrm{A}]) - V_{\mathrm{cell}}\cdot(\mathrm{deg\_dim}\cdot[\mathrm{A2}]) + V_{\mathrm{cell}}\cdot($$
$$- V_{\mathrm{cell}}\cdot(\mathrm{repB}\cdot[\mathrm{gB}]\cdot[\mathrm{A2}])$$

$$\frac{\mathrm{d}([\mathrm{B2}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = -V_{\mathrm{cell}}\cdot(\mathrm{dim\_r}\cdot[\mathrm{B2}]) + V_{\mathrm{cell}}\cdot(\mathrm{dim}\cdot[\mathrm{B}]\cdot[\mathrm{B}]) - V_{\mathrm{cell}}\cdot(\mathrm{deg\_dim}\cdot[\mathrm{B2}]) + V_{\mathrm{cell}}\cdot(\mathrm{r}$$
$$- V_{\mathrm{cell}}\cdot(\mathrm{repA}\cdot[\mathrm{gA}]\cdot[\mathrm{B2}])$$

$$\frac{\mathrm{d}([\mathrm{A2gB}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = -V_{\mathrm{cell}}\cdot(\mathrm{rep\_r}\cdot[\mathrm{A2gB}]) + V_{\mathrm{cell}}\cdot(\mathrm{repB}\cdot[\mathrm{gB}]\cdot[\mathrm{A2}])$$

$$\frac{\mathrm{d}([\mathrm{B2gA}]\cdot V_{\mathrm{cell}})}{\mathrm{d}t} = -V_{\mathrm{cell}}\cdot(\mathrm{rep\_r}\cdot[\mathrm{B2gA}]) + V_{\mathrm{cell}}\cdot(\mathrm{repA}\cdot[\mathrm{gA}]\cdot[\mathrm{B2}])$$

A.1.2 DP-MA

$$\frac{d([A]\cdot V_{cell})}{dt} = -V_{cell}\cdot(deg\cdot[A])+2\cdot V_{cell}\cdot(dim\_r\cdot[A2])-2\cdot V_{cell}\cdot(dim\cdot[A]\cdot[A])+V_{cell}\cdot$$
$$+V_{cell}\cdot(aut\_2\cdot[A2gA])$$

$$\frac{d([gA]\cdot V_{cell})}{dt} = -V_{cell}\cdot(aut\_1\cdot[A2]\cdot[gA])+V_{cell}\cdot(rep\_r\cdot[B2gA])-V_{cell}\cdot(repA\cdot[gA]\cdot[B2])$$
$$+V_{cell}\cdot(aut\_3\cdot[A2gA])$$

$$\frac{d([B]\cdot V_{cell})}{dt} = +V_{cell}\cdot(aut\_2\cdot[B2gB])-V_{cell}\cdot(deg\cdot[B])+2\cdot V_{cell}\cdot(dim\_r\cdot[B2])-2\cdot V_{cell}\cdot($$
$$+V_{cell}\cdot(geB\cdot[gB])$$

$$\frac{d([gB]\cdot V_{cell})}{dt} = +V_{cell}\cdot(aut\_3\cdot[B2gB])-V_{cell}\cdot(aut\_1\cdot[B2]\cdot[gB])+V_{cell}\cdot(rep\_r\cdot[A2gB])$$
$$-V_{cell}\cdot(repB\cdot[gB]\cdot[A2])$$

$$\frac{d([A2]\cdot V_{cell})}{dt} = -V_{cell}\cdot(aut\_1\cdot[A2]\cdot[gA])+V_{cell}\cdot(rep\_r\cdot[A2gB])-V_{cell}\cdot(repB\cdot[gB]\cdot[A2])$$
$$+V_{cell}\cdot(dim\cdot[A]\cdot[A])+V_{cell}\cdot(aut\_3\cdot[A2gA])$$
$$-V_{cell}\cdot(deg\_dim\cdot[A2])$$

$$\frac{d([B2]\cdot V_{cell})}{dt} = +V_{cell}\cdot(aut\_3\cdot[B2gB])-V_{cell}\cdot(aut\_1\cdot[B2]\cdot[gB])+V_{cell}\cdot(rep\_r\cdot[B2gA])$$
$$-V_{cell}\cdot(repA\cdot[gA]\cdot[B2])-V_{cell}\cdot(dim\_r\cdot[B2])+V_{cell}\cdot(dim\cdot[B]\cdot[B])-V_{cell}$$

$$\frac{d([B2gA]\cdot V_{cell})}{dt} = -V_{cell}\cdot(rep\_r\cdot[B2gA])+V_{cell}\cdot(repA\cdot[gA]\cdot[B2])$$

$$\frac{d([A2gB]\cdot V_{cell})}{dt} = -V_{cell}\cdot(rep\_r\cdot[A2gB])+V_{cell}\cdot(repB\cdot[gB]\cdot[A2])$$

$$\frac{d([B2gB]\cdot V_{cell})}{dt} = -V_{cell}\cdot(aut\_3\cdot[B2gB])+V_{cell}\cdot(aut\_1\cdot[B2]\cdot[gB])$$

$$\frac{d([A2gA]\cdot V_{cell})}{dt} = +V_{cell}\cdot(aut\_1\cdot[A2]\cdot[gA])-V_{cell}\cdot(aut\_3\cdot[A2gA])$$

A.2 Sequences

A.2.1 pKDL071

A.3 Algorithms

A.3.1 Clustering algorithms

A.3.1.1 Deterministic case

Algorithm 7 Clustering the steady state deterministic simulation results

```
1:  for each data point do
2:      if first point then
3:          Make first cluster
4:          cluster counter = 1
5:      else
6:          for each cluster do
7:              if cluster within cluster means ± delta then
8:                  Add to existing cluster
9:                  Update means of clusters
10:             end if
11:             if reached_end and not assigned to cluster then
12:                 cluster counter += 1
13:                 Add new cluster
14:             end if
15:         end for
16:     end if
17: end for
```

A.3.1.2 Stochastic case

Gap statistic

Algorithm 8 Choosing the optimal number of clusters

```
function Wk(clusters, cluster_centres)
    for each cluster do
        for each point in cluster do
            a = matrix norm (cluster_centre − point)
        end for
        dk = ∑((a)^2) × (2× number of points in cluster)
    end for
    wk = ∑(dk) / (2×(numberofpointsincluster))
    return wk
end function

function Gap_statistic(data, cutoff)
    ks = [1,2,3,4]
    for k in ks do
        cluster_centres, clusters = Kmeans(data, k, cutoff)
        Wk = log(Wk(clusters, cluster_centres))
        Create references datasets
        for each references dataset do
            cluster_centres, clusters = Kmeans(data, k, cutoff)
            BWk = log(Wk(clusters, cluster_centres))
        end for
        Wkb = ∑(BWk) / 10
        sk = sqrt(∑((BWk−Wkb)^2 / 10))
    end for
    sk = sk × sqrt(1 + 1/B)
    return ks, Wk, Wkb, sk, data_centres, clusters
end function

function Distance(data, cutoff)
    ks, logWks, logWkbs, sk, clusters_means, clusts = gap_statistic(data, cutoff)
    gaps = logWks − logWkbs
    optimum number of clusters = gaps[i] ≥ (gaps[i + 1] − sk[i + 1])
    return cluster_counter, clusters_means
end function
```

Algorithm 9 Clustering stochastic case

```
1:  function KMEANS CLUSTERING(data, k, cutoff)

2:      function UPDATE_CENTRES(old_centres, values)
3:          centre_coords = mean for each dimension
4:          shift = GETDISTANCE(centre_coords, old_centres)
5:          return shift, centre_coords
6:      end function

7:      function GETDISTANCE(a, b)
8:          dist = sqrt((a[x] − b[x])^2 + (a[y] − b[y])^2)
9:          return dist
10:     end function

11:     while True do
12:         for each point in data do
13:             for each cluster do
14:                 dist =GETDISTANCE(point, cluster centre)
15:             end for
16:             Find cluster with minimum distance
17:             Repopulate clusters
18:         end for
19:         biggest_shift ← 0
20:         for as many times as there are clusters do
21:             shift, cluster centres = UPDATE_CENTRES(old_centres, clusters)
22:             biggest_shift = max between shift, biggest_shift
23:         end for
24:         if biggest_shift ≤ cutoff then
25:             break
26:         end if
27:     end while
28:     return cluster_centres, clusters
29: end function
```

K-means clustering