Using computational modelling to build robust synthetic genetic switches

Miriam Leon
Supervisor: Chris P Barnes
Department of Cell and Developmental Biology
University College London

0.1 Abstract

Creating synthetic devices that are robust to changing cellular contexts will be key to the success of synthetic biology. 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 and it is known that additional feedback interactions can increase robustness to extrinsic noise sources. The genetic toggle switch is examined, consisting of two mutually repressing transcription factors. It is known for exhibiting bistability, and is found in natural systems when binary fate decisions need to be made, such as in a developmental pathway. It can also be used in synthetic systems to sense and respond to different environments. Here a new tool is presented, called StabilityChecker, that takes advantage of sequential Monte Carlo to identify regions of parameter space capable of producing the desired stability while handling uncertainty in biochemical rate constants and initial conditions. StabilityChecker is tested against known systems. A collection of networks are examined, ranging from the simple genetic toggle switch to those containing different positive feedback connections. We verify StabilityChecker against known results of the simple toggle switch. We find that positive feedback and other regulatory interactions can significantly increase the robustness of these switching systems.

Contents

List of Figures

List of Tables

1 Modelling switches

1.1 Introduction

A challenge that synthetic biology is facing is the ability to build synthetic devices that are robust to changing cellular contexts. 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 et al. 2009). There has been great progress in the quantity and quality of devices being created(), but we are still lagging behind in our ability to rationally design a device and minimize the time-consuming and expensive experimental trial and error.

One of the most common devices used StabilityFinder in synthetic biology is the genetic toggle switch. A toggle switch consists of a set of transcription factors that mutually repress each other (Gardner et al. 2000). Toggle switches play a major role in binary cell fate decisions like stem cell differentiation, as they are capable of exhibiting bistable behaviour. Bistability is defined as the system being able to have 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.

Bistability ensures that the differentiating cell will go down one pathway, or the other, with no intermediate phenotypes being possible. 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 et al. 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 reinforce this dynamic and the system remains in the same state, until an external stimulus disturbs it (Ferrell 2002).

Despite their simplicity, toggle switches can be powerful building blocks with which to create complex responses in a synthetic network. They have been used in isolation() or in tandem to create complex networks and signalling cascades.

The toggle switches have been studied extensively and there are numerous studies based on a number of different methods of modelling and analysing the dynamics of this system. A summary can be found in Table 1.1. Numerous studies have concluded that cooperativity is a necessary condition for bistability to arise (). Lipshtat et al. (2006) found that stochastic effects can give rise to bistability even without cooperativity in three kinds of switch: In the exclusive switch, in which there can only be one repressor bound at any one time, the switch in which there is degradation of bound repressors or the switch in which free repressor proteins can form a complex which renders them inactive as transcription factors (Lipshtat et al. 2006). In their study, Ma et al. (2012) found that the stochastic fluctuation 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. In their system, the toggle switch was found to be tristable, as small number effects render the third unstable steady state stable. In the study conducted by Biancalani & Assaf (2015), they identified multiplicative noise as the source of bistability in the stochastic case. This bistability disappears of the noise source is reduced below a threshold, which in the toggle switch is represented by the repression strength. Thus if the repression strength falls below a certain threshold, the switch becomes monostable. In the genetic toggle switch the source of noise is the weakness of the repression so thus, increasing repression strength one decreases the source of noise and increases the stability of the switch (Warren & ten Wolde 2005). Warren & ten Wolde (2005) concluded that the exclusive switch is always more robust than the general switch,

since the free energy barrier is higher. 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 a 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 a bigger, more complex system, until its behaviour can be predicted otherwise designing such a system with predictable behaviour will be near impossible.

Table 1.1 Stability of switches found in literature

		Simple		Double positive autoregulation		
	Stability	*Reference*	*Notes*	*Stability*	*Reference*	*Notes*
Deterministic	Monostable	Loinger 2007	no cooperativity, exclusive & general	Bistable	Guantes 2008	
	Bistable	Gardner 2000	copperativity >2,	Tristable	Guantes 2008	
		Loinger 2007	bound repressor degradation	4 steady steates	Guantes 2008	
Stochastic	Monostable	Loinger 2007	no cooperativity, weak repression	Tristable	Lu 2014	
		Lu 2014				
	Bistable	Biancalani 2015	exclusive, controlled by noise strength			
		Lipshtat 2006	no cooperativity			
		Loinger 2007	no cooperativity, exclusive & bound repression degradation			
	Tristable	Loinger 2007	no cooperativity, strong repression			
		Ma 2012				

In order to solve this problem, we created a framework, StabilityFinder, that can be used to elucidate the stability each model is capable of, under conditions of parameter uncertainty. Using this framework one can study the different methods of modelling the switch and be able to attribute the differences in the results to each type of modelling. By using the same framework to compare the different modelling techniques we can get to the bottom of what the stability the switch is capable of and why each method produces a different result. We will use StabilityFinder to study three different models of the toggle switch. This methodology can also be used to uncover the design principles behind making a bistable switch, as well as those necessary to make a tristable switch. The methodology presented here can be used to study existing systems, toggle switches that exist in nature, as well as synthetic systems, if used as an aid to system design in synthetic biology. By identifying the parameter ranges that can give rise to the desired stability of a system, one can choose the parts of the genetic system accordingly. For example, if StabilityFinder dictates that gene expression must be low for a given stability, one can select a weak promoter or a low copy plasmid for their desired construct.

1.2 Methods

The algorithm we developed can be utilised to identify regions of parameter space capable of producing a desired stability while handling uncertainty in biochemical rate constants and initial conditions. It can be used to design new systems of desired stability or study the stability of existing systems. The method we are presenting here is based on the Approximate Bayesian Computation, Sequential Monte Carlo method (ABC SMC), a statistical inference method developed by Toni et al. (2009). This simulation-based method, using an iterative process can arrive at a distribution of parameter values that can give rise to a desired system behaviour.

ABC methods are used for inferring the posterior distribution in cases where it is too computationally hard to evaluate the likelihood function. Instead of calculating the likelihood, ABC methods simulate the data and then compare the desired to the simulated data (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. The generic ABC algorithm is as follows:

The simplest ABC algorithm is the ABC rejection sampler (Pritchard et al. 1999). In this method, parameters are sampled and simulated, and for each sample if the

Algorithm 1 Generic ABC 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 $d \leq \epsilon$, accept θ

distance from that to the desired behaviour is greater than a threshold, the sample is rejected, otherwise accepted. 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).

The method used here, developed by Toni et al. (2009) is the ABC SMC method that avoids both these issues faced by the above methods. It propagates the prior through a series of intermediate distributions in order to arrive to 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).

In the algorithm we developed the desired behaviour in ABC SMC is replaced by a desired stability that the simulated model has to have to be accepted. The ϵ measures the distance of the current stability to the desired stability. The advantages of this method is that it can handle stochastic as well as deterministic models, and can inherently handle parameter uncertainty (Barnes et al. 2011). The algorithm is summarised in Algorithm ?? below.

We packaged this algorithm into a python package, called StabilityFinder. The user provides the model file, in the form of SBML of cuda as well as the input file. The user input file contains all the necessary information to run the algorithm that is not contained in the model itself. The user specifies the desired stability, the total variance as well as the variance within each cluster that . In addition the user provides the tolerance for the distance from the desired behaviour necessary for the algorithm to terminate. The flow of execution is illustrated in Figure 1.1 below.

Algorithm 2 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: $$w_t^{(i)} = \begin{cases} 1, & \text{if } p = 0 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})}, & \text{if } p \geq 0. \end{cases}$$
15: Normalise weights
16: **repeat**steps 3 - 15
17: **until** $\epsilon \leq \epsilon_T$

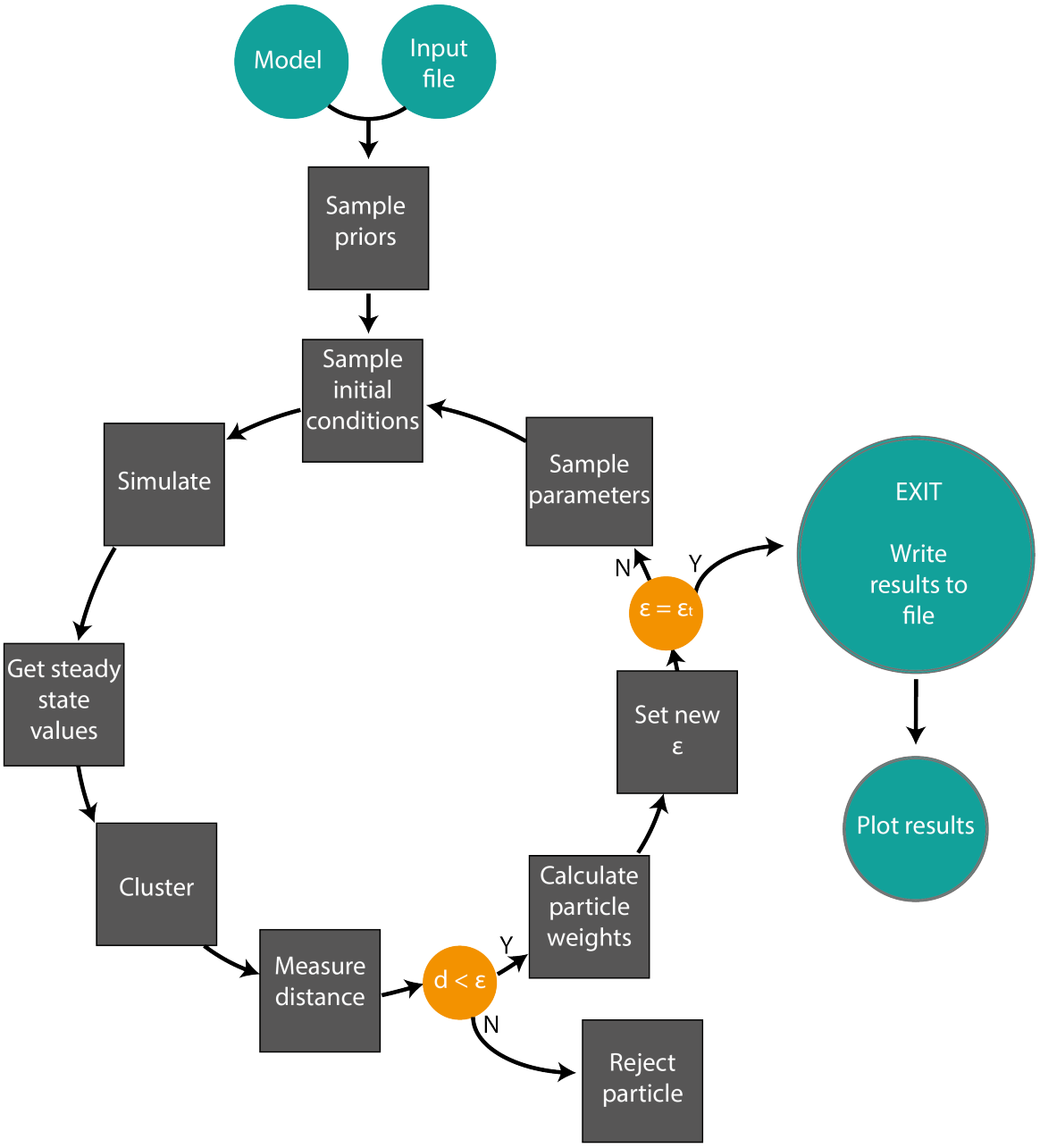

Figure 1.1 Flowchart of the algorithm used in StabilityFinder. ϵ stands for the threshold for the distance from the desired value.

StabilityFinder begins by sampling from the prior, by selecting a random value from within the user-specified range. Each sample from the priors is called a particle. The number of particles is specified in the user input file. The algorithm then proceeds by sampling initial conditions using Latin Hypercube sampling that ensures an even coverage of values from within the specified range. The model is then simulated for each parameter sample set, for each initial condition set. It uses cuda-sim software (Zhou et al. 2011) to simulate the models thus taking full advantage of GPUs. As soon as the simulations are complete, the steady state values for the two species of interest are clustered in order to determine the stability achieved by each parameter sample set. Whether the model has been simulated using ODEs or the Gillespie algorithm dictates the method of clustering used. The two algorithms are summarised in the Appendix. Once the number of clusters present for each parameter set has been determined, the distance from the desired values is measured. 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. The tolerated distance is first calculated from the samples taken, and then decreased at each iteration until it reaches the final accepted tolerance. 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.

Calculating robustness

We use the results from StabilityFinder to estimate system robustness, by comparing the volume of the posterior to the volume of the prior in a given model. Here we are calculating it using a Monte Carlo sampling accept reject algorithm. Taking a number of random samples from the prior, it keeps track of how many are also found in the functional region F of the posterior. The ones that are found in F are accepted and the rest rejected. Robustness is then defined as the number of accepted samples divided by the total number of samples.

Algorithm 3 Calculating robustness via Monte Carlo sampling rejection

1: Sample from priors
2: Get min and max boundaries of functional region of posterior (F)
3: **if** sample within F **then**
4: $\quad accepted+ = 1$
5: **end if**
6: $acceptance\_rate = \frac{accepted}{numberofsamples}$
7: **return** $acceptance\_rate$

1.3 Results

Testing StabilityFinder: The Gardner toggle switch

This toggle switch model was developed by T. S. Gardner (Gardner et al. 2000). This model consists of two mutually repressing transcription factors and is defined by the following ODEs:

$$\frac{du}{dt} = \frac{a_1}{1 + v^{\beta}} - u \tag{1.1}$$

$$\frac{dv}{dt} = \frac{a_2}{1 + u^{\gamma}} - v \tag{1.2}$$

Where a_1 and a_2 are the effective rates of synthesis of repressors 1 and 2 respectively. u is the concentration of repressor 1 and v the concentration of repressor 2. β is the cooperativity of repression of promoter 1 and γ of repressor 2. In their paper, T. S. Gardner (Gardner et al. 2000) state that there are two conditions for bistability of this toggle switch model. That a_1 and a_2 are balanced and that β and γ are >1. In order to test our methodology, we used StabilityFinder to find the posterior distribution for which this model exhibits bistable behaviour. So setting the desired behaviour to bistable, and using for priors a wide range of values that includes the values used in the Gardner paper, we test StabilityFinder to see if it finds the same conditions as the ones set by T. S. Gardner. The prior distributions used are shown in Table 1.2. The posterior distribution calculated by StabilityFinder is shown in Figure 1.2.

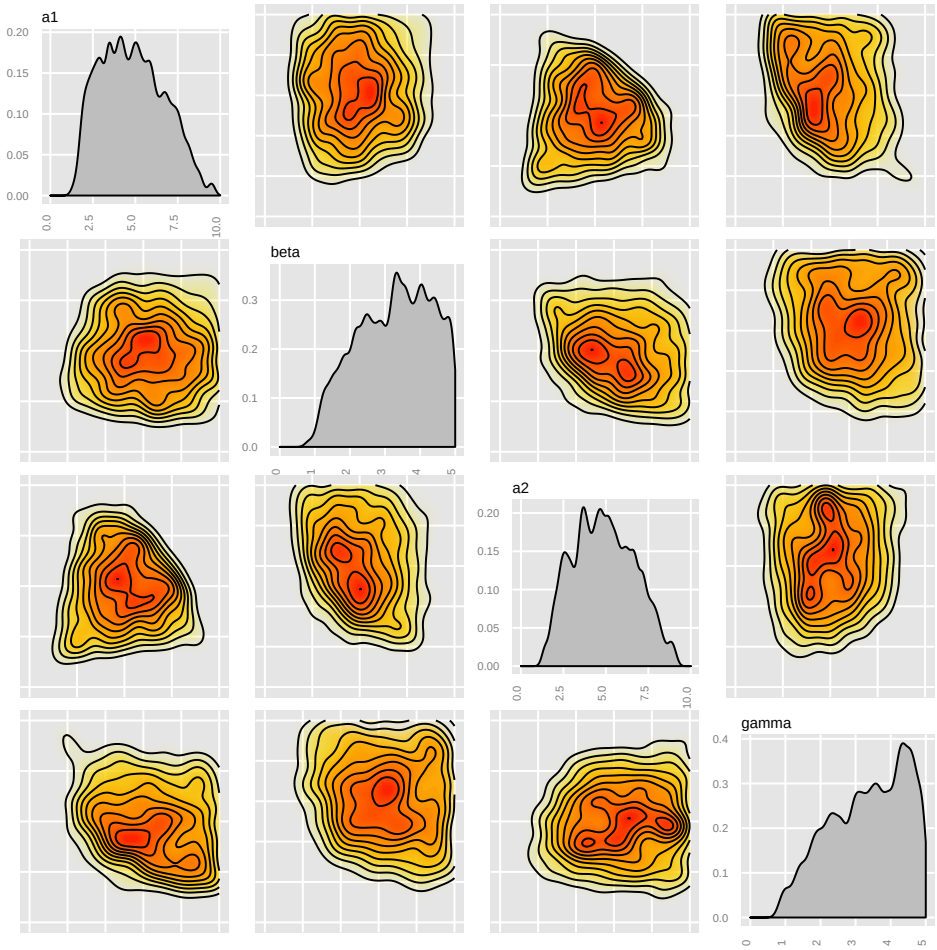

Figure 1.2 The posterior distribution of the bistable Gardner toggle switch. The parameters a_1, a_2 represent to the effective synthesis rate of repressors 1 and 2 respectively. Parameters β, γ represent the cooperativity on repressors 1 and 2 respectively. The marginal distributions of each parameter are found on the diagonal and pairwise joint distributions along the sides.

These results agree with the results reported by the original paper (Gardner et al. 2000) . For this switch to be bistable a_1 and a_2 must be balanced while β and γ must both be >1, as can be seen in the marginal distributions of β and γ in Figure 1.2. The conditions set by the original paper for parameters a_1 and a_2 are met, as the

Table 1.2 Gardner switch priors

Parameters				Species	
a_1	β	a_2	γ	s_1	s_2
0-10	0-5	0-10	0-5	0-100	0-100

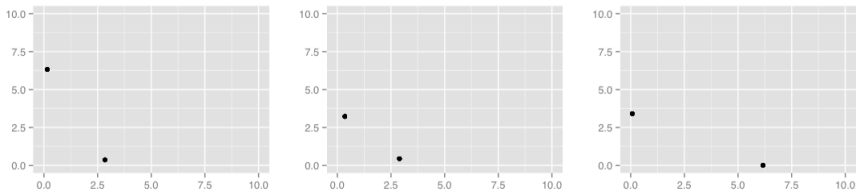

Figure 1.3 A sample of the phase plots produced from the final population of the Gardner switch.

joint distribution shown in Figure 1.2 matches the bifurcation lines calculated in the original paper. This successful test demonstrates that StabilityFinder can be used to find the stability a model is capable of as well as the parameter ranges that can produce that. This allows us to confidently apply the methodology to further models.

Comparing the deterministic and stochastic cases

There are two main ways of modelling a system, deterministically and stochastically. 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). 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 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 (Warren & ten Wolde 2005).

The toggle switch has been modelled both deterministically and stochastically,

with the two methods producing different results. Thus here we use StabilityFinder to compare the stabilities that the model is capable of, under the different simulation types. By using the same framework, and the same prior distributions for the models, one can directly compare the posterior distributions produced by the deterministic and the stochastic case, and uncover the effects that are due to the stochasticity of the system. This is relevant in a a biological model in which small molecule numbers and external noise are not negligible. Using StabilityFinder and using the same priors for the two models (Table 1.3), we calculated the posterior for each both bistable models. The results are shown in Figure 1.4. The prior ranges used are much wider than in the test case in order to allow more flexibility in both models and have a greater range over which to compare the models.

Table 1.3 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

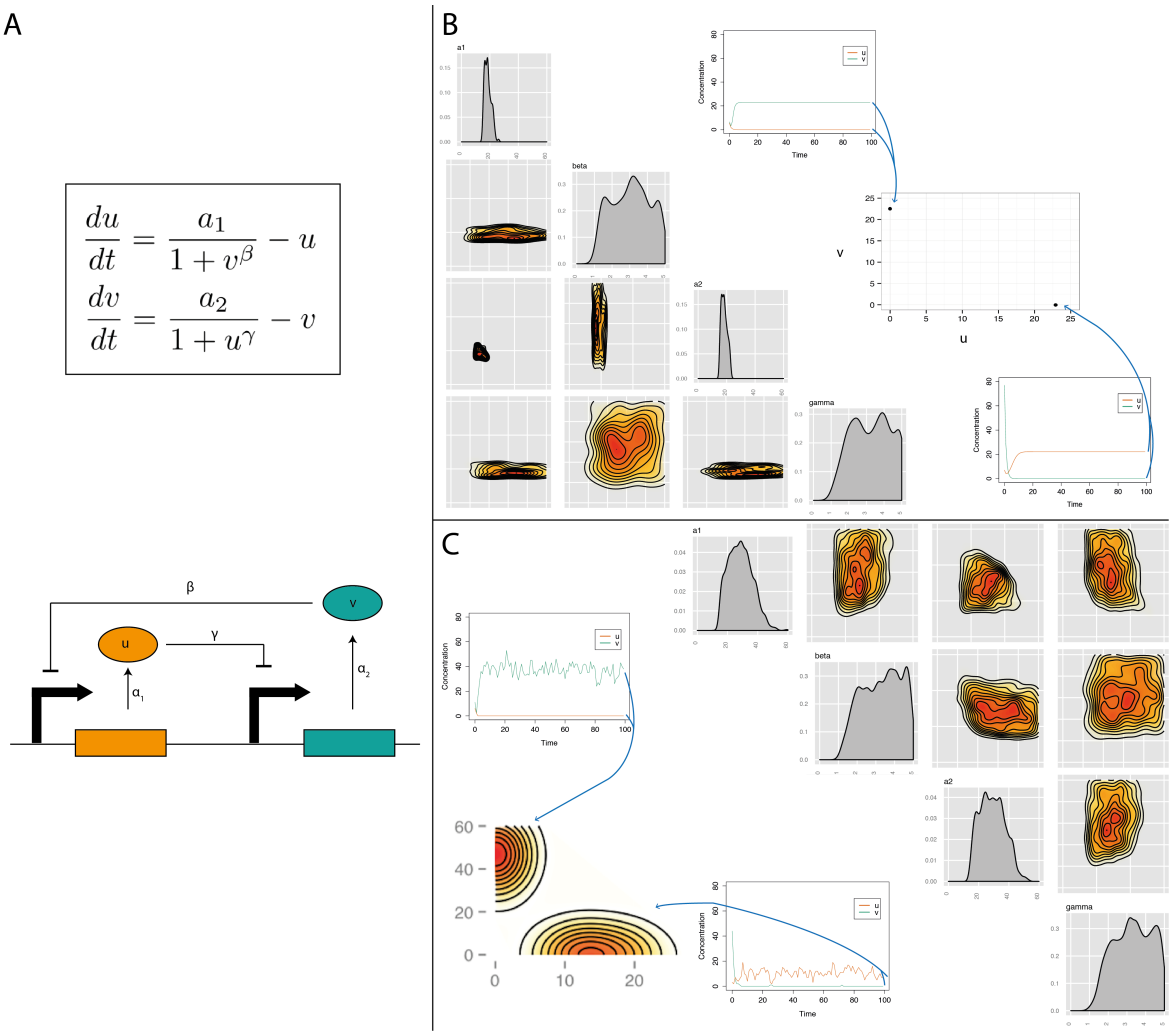

Figure 1.4 The Gardner toggle switch is modelled stochastically and deterministically. The model is shown graphically and mathematically in (A), The posterior and phase space of the deterministic model is shown in (B) and of the stochastic case in (C).

As can be seen in the above results, stochasticity has a big effect on the posterior. Firstly, a greater parameter range can produce a bistable behaviour when stochastic effects are taken into account. The condition set by T. S Gardner (Gardner et al. 2000) for the values of a_1 and a_2 to be balanced holds in both the deterministic and stochastic cases but in the stochastic case this is met by a wider range of values. The second conditions of β, γ >1 also needs to be met in the stochastic case. The posterior of the deterministic model shown in Figure 1.4B, parameters a_1 and a_2 also have an upper limit of values they can take and still create a bistable switch. In order to test this result, we find the roots of the system for large values of a_1 and a_2 in order to see if three roots are still found, two stable and one unstable. The results as shown in Figure indicate that the system is still bistable for increasing values of a_1 and a_2. This suggests that the upper limit found with StabilityFinder is an artefact of the variance limit imposed on the system. In order to find the steady states we impose an accepted distance from a given total variance for each model. When the two clusters of steady state values are too far removed, this increases total variance of the system and would consequently be rejected in StabilityFinder.

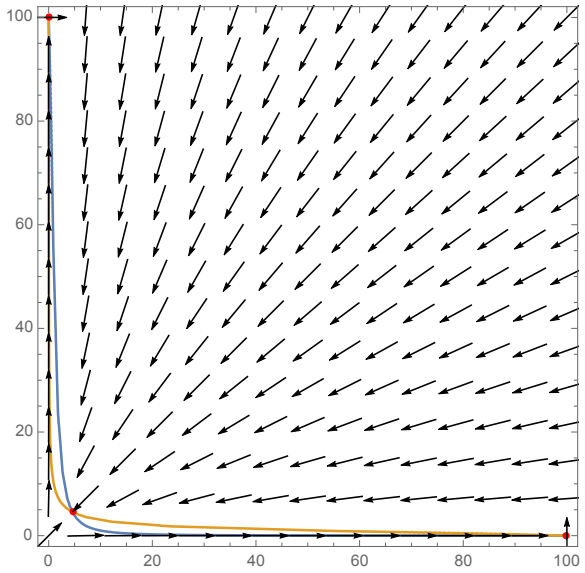

Figure 1.5 Solving the Gardner toggle switch. The parameters values used are: a_1, a_2 = 100 and β, γ = 2. The system has three roots, of which one was found to be unstable and the other two stable. This result disagrees with that found in StabilityFinder, that a_1 and a_2 have an upper limit of 30.

When stochasticity is taken into account, robustness increases significantly as seen in Figure 1.6. This indicates that stochasticity increases the ability of the model

to withstand fluctuations in parameter values and still produce the desired bistability. A deterministic model cannot predict this increased robustness.

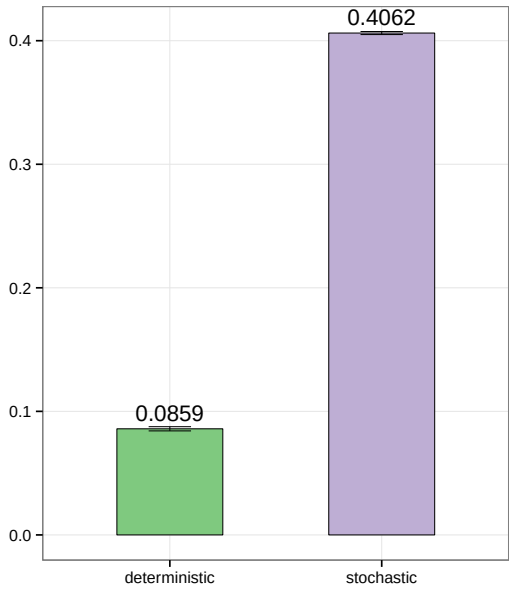

Figure 1.6 Comparing the robustness of the deterministic and stochastic Gardner switch models

In order to check if this increase in robustness is caused by the different clustering methods used in the stochastic and deterministic cases, we tested the deterministic case by running the exact same model but using the clustering algorithm used in the stochastic case. Then we compared the robustness using the method outlined above. These results are shown in Figure 1.7

This indicates that the clustering methods used have a big effect on the robustness measured. The increase in robustness seen in Figure 1.6 could be a bias of the clustering algorithms.

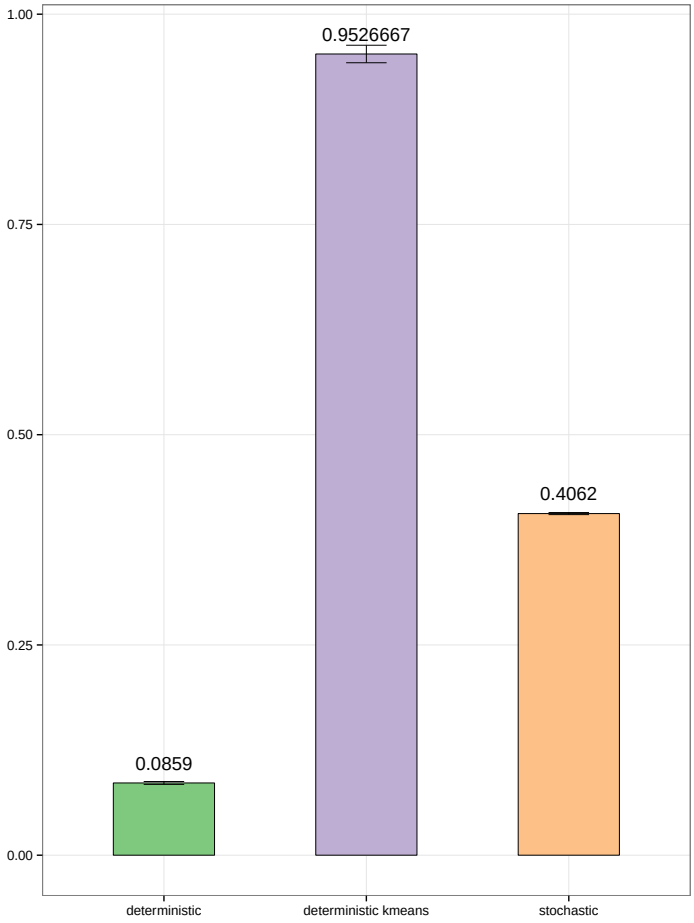

Figure 1.7 The effect of the clustering methods on robustness measurements

1.4 Lu toggle switch models

In their study, Lu et al. 2013 explored the effect of white Gaussian and shot noise on the multi-state switches. They found that the classical toggle switch, with the repressing transcription factors has two steady states and the toggle switch with added double positive auto-regulation has three steady states. By extending the analysis on these models by using StabilityFinder we can determine the design principles that make a tristable versus a bistable switch. This is another example of a use for StabilityFinder. The system used in their study is defined by two dynamical systems:

$$\dot{x} = f_x(x,y) = g_x H^S_{xy}(y) H^S_{xx}(x) - k_x x \tag{1.3}$$

$$\dot{y} = f_y(x,y) = g_y H^S_{yx}(x) H^S_{yy}(y) - k_y y \tag{1.4}$$

$$H^S_{xx} = H^-_x(x) + \lambda_x H^+_x(x) \tag{1.5}$$

$$H^-_x(x) = 1/\left[1 + (x/a_x)^{n_x}\right] \tag{1.6}$$

$$H^+_x(x) = 1 - H^-_x(x) \tag{1.7}$$

The three switches are illustrated in Figure 1.8 below.

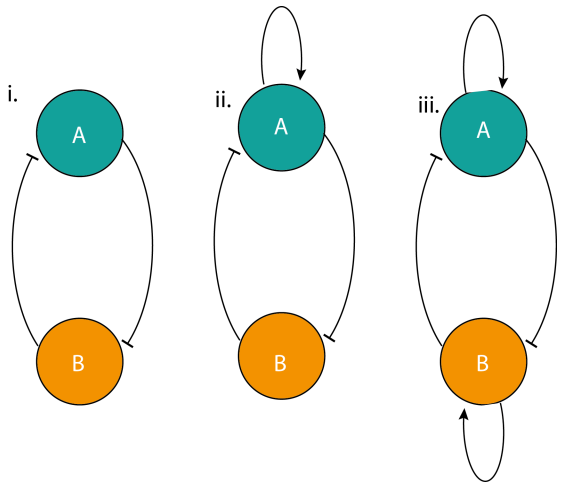

Figure 1.8 The three Lu switches studied. (i) is the classical model with no self activation, (ii) has a single positive autoregulation and (iii) has double positive autoregulation.

Classical model

For the classical model, in which no self-activation is present, the system reduces to the following equations:

$$\dot{x} = f_x(x,y) = g_x H^S_{xy}(y) - k_x x, \tag{1.8}$$

$$\dot{y} = f_y(x,y) = g_y H^S_{yx}(x) - k_y y \tag{1.9}$$

For the parameter values used in the Lu study, as shown in Table 1.4, the system exhibits three steady states (Figure 1.9), of which two are stable and one is unstable.

Table 1.4 Lu classical model parameter values

gx	gy	kx	ky	nxy	nyx	xxy	xyx	Ixy	Iyx
40	40	0.1	0.1	3	3	200	200	0.1	0.1

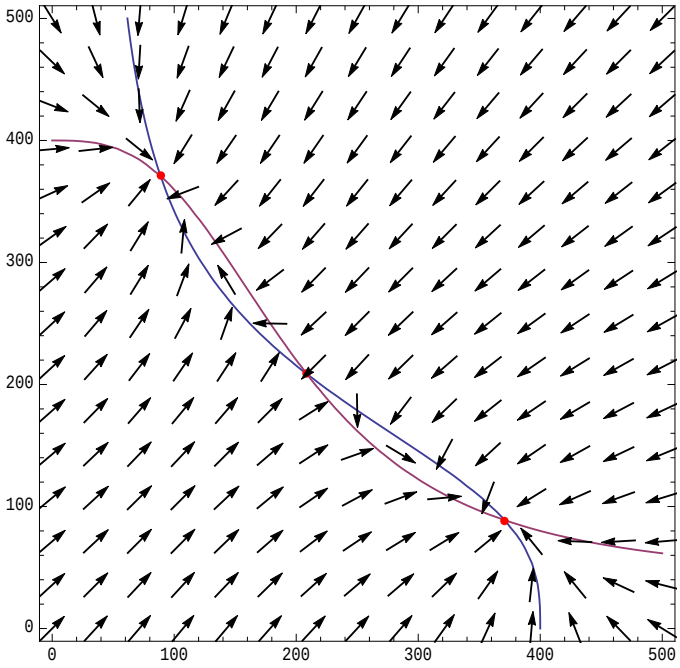

Figure 1.9 Phase portrait of the Lu classical model with no self activation. There are two stable steady states and one unstable steady state.

Using StabilityFinder with priors centred around the parameter values used in the original paper (Table 1.5), we can find the robustness of this bistable behaviour, as well as identify the most important parameters for bistability.

Table 1.5 Lu classical model priors

gx	gy	kx	ky	nxy	nyx	xxy	xyx	Ixy	Iyx
35-45	35-45	0-0.2	0-0.2	2-4	2-4	150-250	150-250	0-0.2	0-0.2

The posterior distribution of this model is shown in Figure 1.10. As can be seen from the posterior, the most restrained parameters are kx and ky, the parameters responsible for the degradation of the species involved. This indicates that the rate of degradation of the species is critical for the desired dynamic to occur.

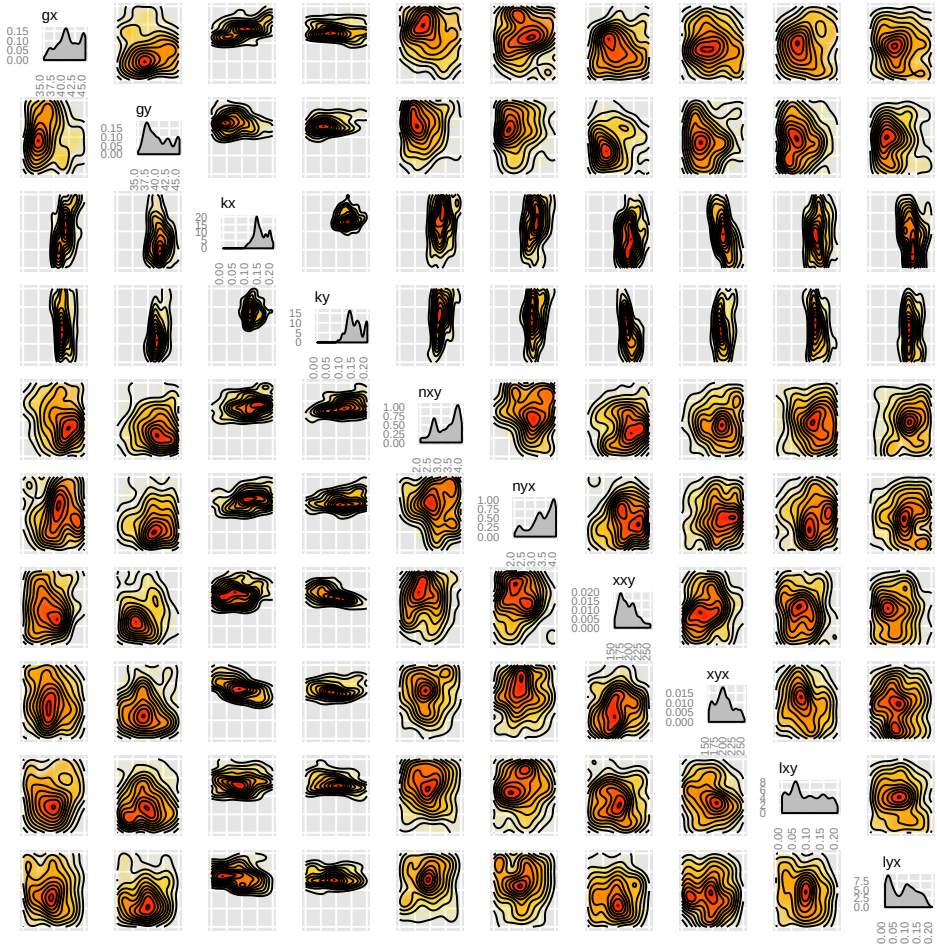

Figure 1.10 The posterior distribution of the Lu classical model with no self activation. kx and ky are the most constrained parameters.

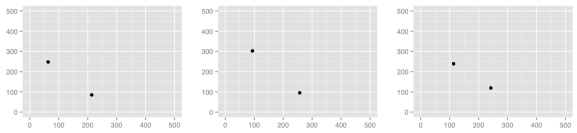

Figure 1.11 A sample of the phase plots produced from the final population of the Lu classical model.

Single positive autoregulation

Table 1.6 Lu model with single self-activation priors

parameter	range
gx	1-2
gy	20-25
kx	50-55
ky	48-52
nxy	30-35
nyx	0.1-0.2
xxy	2-3
xyx	0.4-0.6
lxy	0.02-0.04
lyx	0.02-0.04
nXX	25-30
nYY	0.01-0.02
xXX	0.4-0.5
xYY	1-3
lXX	65-72
lYY	0.02-0.04

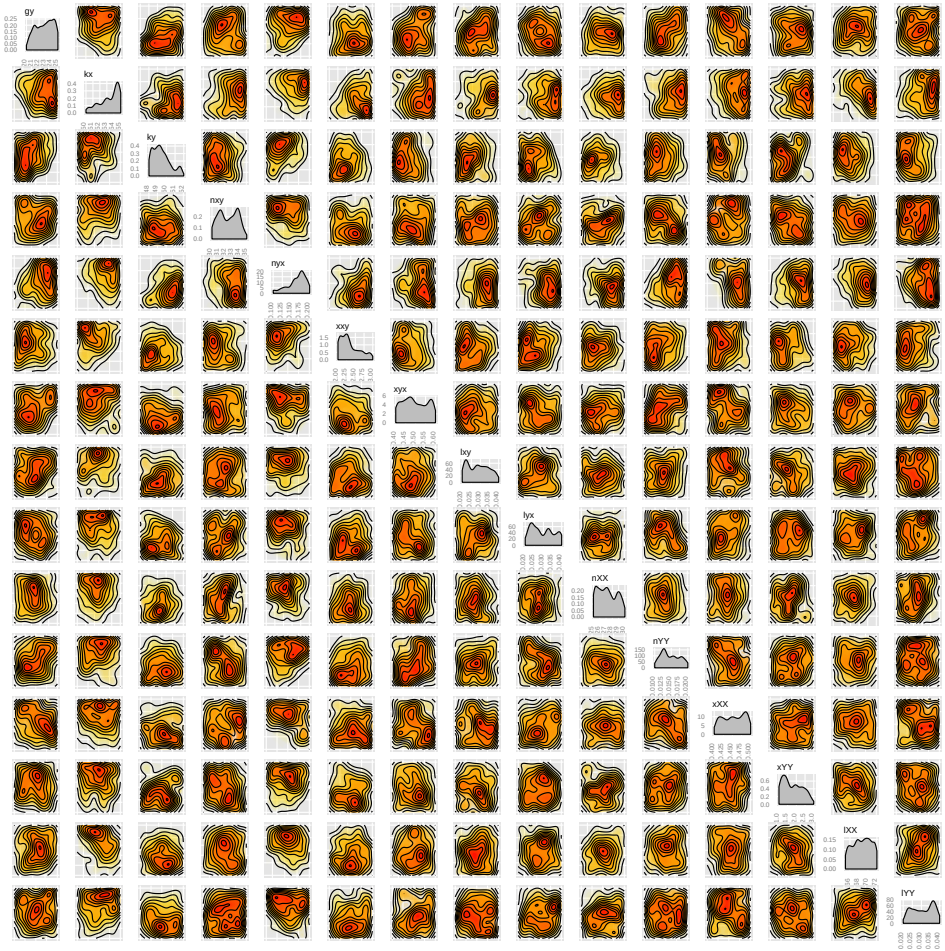

Figure 1.12 The posterior distribution of the Lu model with asymmetric self activation

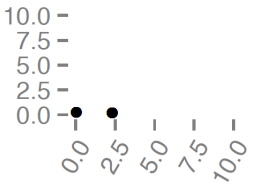

Figure 1.13 Phase plot of asymmetric Lu toggle switch

Double positive autoregulation

If self-activation is included, then the system is that presented in equations 1.3 and 1.4 . When values that are presented in Table 1.7 are assigned to the parameters for the self-activating model, the system exhibits three stable and two unstable steady states as seen in Figure 1.14.

Table 1.7 Lu model with self-activation parameter values

gx	gy	kx	ky	nxy	nyx	xxy	xyx	Ixy	Iyx
4	4	0.1	0.1	1	1	200	200	0.1	0.1

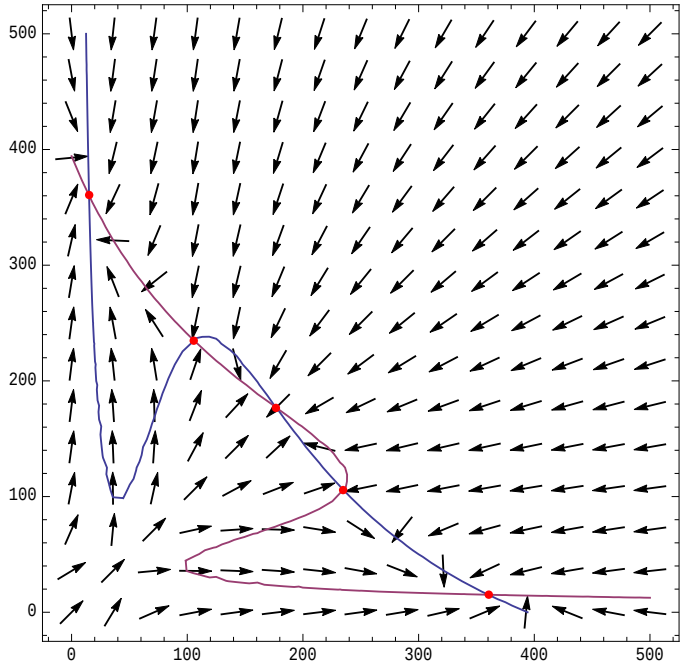

Figure 1.14 Phase portrait of the Lu model including double self-activation. The model had three stable steady states and two unstable steady states.

Using StabilityFinder and wide priors around the original values, shown in Table 1.8, we can explore the robustness of this behaviour, in a similar way as done for the classical model.

Table 1.8 Lu model with double self-activation priors

parameter	range
gx	1-100
gy	1-100
kx	0-1
ky	0-1
nxy	0-10
nyx	0-10
xxy	100-1000
xyx	100-1000
lxy	0-1
lyx	0-0.2
nXX	0-10
nYY	0-10
xXX	50-500
xYY	50-500
lXX	1-20
lYY	1-20

The posterior is shown in Figure 1.15.

Figure 1.15 The posterior distribution of the Lu model with double self activation.

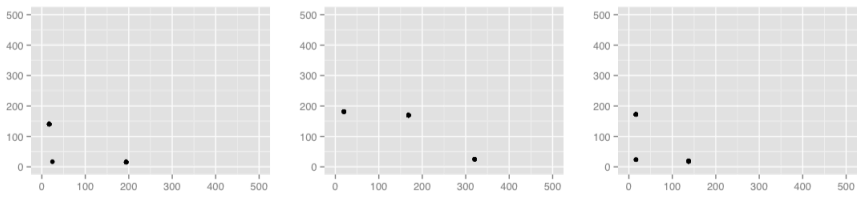

Figure 1.16 A sample of the phase plots produced from the final population of the Lu tristable model.

Extracting the design principles of a tristable switch

Next, we went on to study the design principles that make a switch tristable vs bistable. In order to do that, first we use StabilityFinder to find the parameter space that gives rise to a bistable switch, given the same model and priors used above to find a tristable switch, as shown in Figure 1.17.

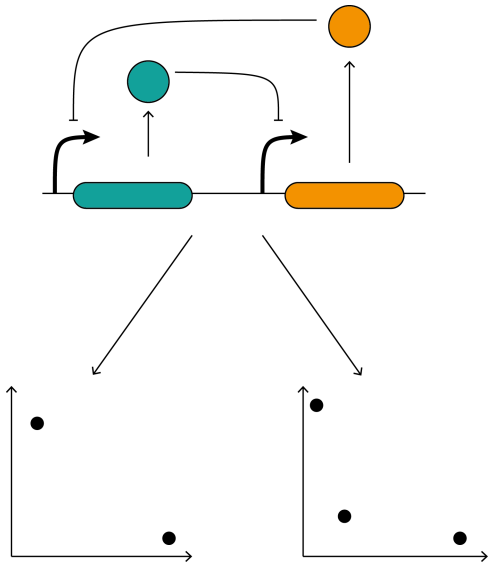

Figure 1.17 Finding the parameter spaces that make the Lu double positive model bistable and tristable.

By plotting the resulting posterior with that of the tristable switch shown in Figure 1.15 we can examine any separation of values between a bistable and a tristable switch, as shown in Figure 1.18.

Figure 1.18 Comparing the posteriors of the tristable Vs bistable Lu switch.

In order to separate the values in a more meaningful way, we implement an algorithm as outlined in Appendix. By taking samples from the posterior of a bistable switch, we keep track of the values for each parameter that are also found in the posterior of the tristable switch, and vice versa. The results are shown in Figure 1.19 below:

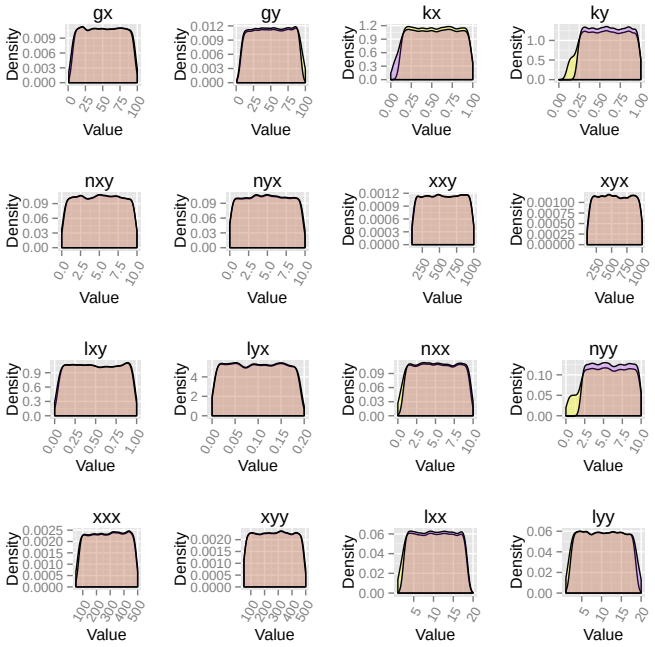

Figure 1.19 Design principles of the tristable Lu switch.

From the results above we can conclude that there is no significant difference between a bistable and a tristable switch in parameter values.

1.5 Mass Action switches

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. Here we define a robust device as a device that can withstand fluctuations in parameter values and still produce the desired behaviour. Feedback loops are well known key regulatory motifs (Brandman et al. 2005). Negative feedback loops are essential for homeostasis and buffering (Thomas et al. 1995) thus increasing robustness to extrinsic noise sources and positive feedback loops can generate multistationarity in a system (Thomas et al. 1995).

Incorporating this kind of additional feedback interactions can make a design more robust and reliable. Here we use StabilityFinder to compare the robustness of two switches, a simple toggle switch and a switch with added positive auto-regulation.

In order to study this system in the most realistic way, we avoid using the quasi-steady state approximation (QSSA) that is often used in modelling the toggle switch. 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 is often used since it reduces the dimensions of the system thus reducing computational time (Pedersen et al. 2007). The QSSA assumption is met in vitro but often does not hold in vivo. Its misuse can lead to large errors and incorrectly estimated parameters (Pedersen et al. 2007). Parameter estimation is central to this project and since StabilityChekcker is able to analyse models with numerous equations and parameters, there is no need for this assumption, which approximates a solution. Using Mass Action, the two models used are shown below:

Standard toggle switch:

$$\mathrm{gA} \xrightarrow{\mathrm{ge}} \mathrm{gA} + \mathrm{A}$$
$$\mathrm{gB} \xrightarrow{\mathrm{ge}} \mathrm{gB} + \mathrm{B}$$
$$\mathrm{A} + \mathrm{A} \xrightarrow{\mathrm{dim}} \mathrm{A2}$$
$$\mathrm{A2} \xrightarrow{\mathrm{dim\,r}} \mathrm{A} + \mathrm{A}$$
$$\mathrm{B} + \mathrm{B} \xrightarrow{\mathrm{dim}} \mathrm{B2}$$
$$\mathrm{B2} \xrightarrow{\mathrm{dim\_r}} \mathrm{B} + \mathrm{B}$$
$$\mathrm{gA} + \mathrm{B2} \xrightarrow{\mathrm{rep}} \mathrm{B2gA}$$
$$\mathrm{B2gA} \xrightarrow{\mathrm{rep\_r}} \mathrm{B} + \mathrm{gA}$$
$$\mathrm{gB} + \mathrm{A2} \xrightarrow{\mathrm{rep}} \mathrm{A2gB}$$
$$\mathrm{A2gB} \xrightarrow{\mathrm{rep\_r}} \mathrm{A2} + \mathrm{gB}$$
$$\mathrm{A} \xrightarrow{\mathrm{deg}} \emptyset$$
$$\mathrm{B} \xrightarrow{\mathrm{deg}} \emptyset$$
$$\mathrm{S} + \mathrm{A2} \xrightarrow{\mathrm{rep\_dim}} \mathrm{SA2}$$
$$\mathrm{SA2} \xrightarrow{\mathrm{rep\_dim\_r}} \mathrm{S} + \mathrm{A2}$$
$$\mathrm{R} + \mathrm{B2} \xrightarrow{\mathrm{rep\_dim}} \mathrm{RB2}$$
$$\mathrm{RB2} \xrightarrow{\mathrm{rep\_dim\_r}} \mathrm{R} + \mathrm{B2}$$
$$\mathrm{R} \xrightarrow{\mathrm{deg}} \emptyset$$
$$\mathrm{S} \xrightarrow{\mathrm{deg}} \emptyset$$

Double positive autoregulation:

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

Deterministic case

Using the same priors for their shared parameters, we used StabilityFinder for both models in order to compare their robustness. The corresponding posterior distributions can be seen in Figures 1.20 and 1.22.

Table 1.9 Mass Action switches priors

ge	rep	rep_r	dim	dim_r	deg	deg_dim	aut_1	aut_2	aut_3
1-10	1-10	1-10	1-10	0-5	1-10	0-0.5	1-10	5-10	1-5

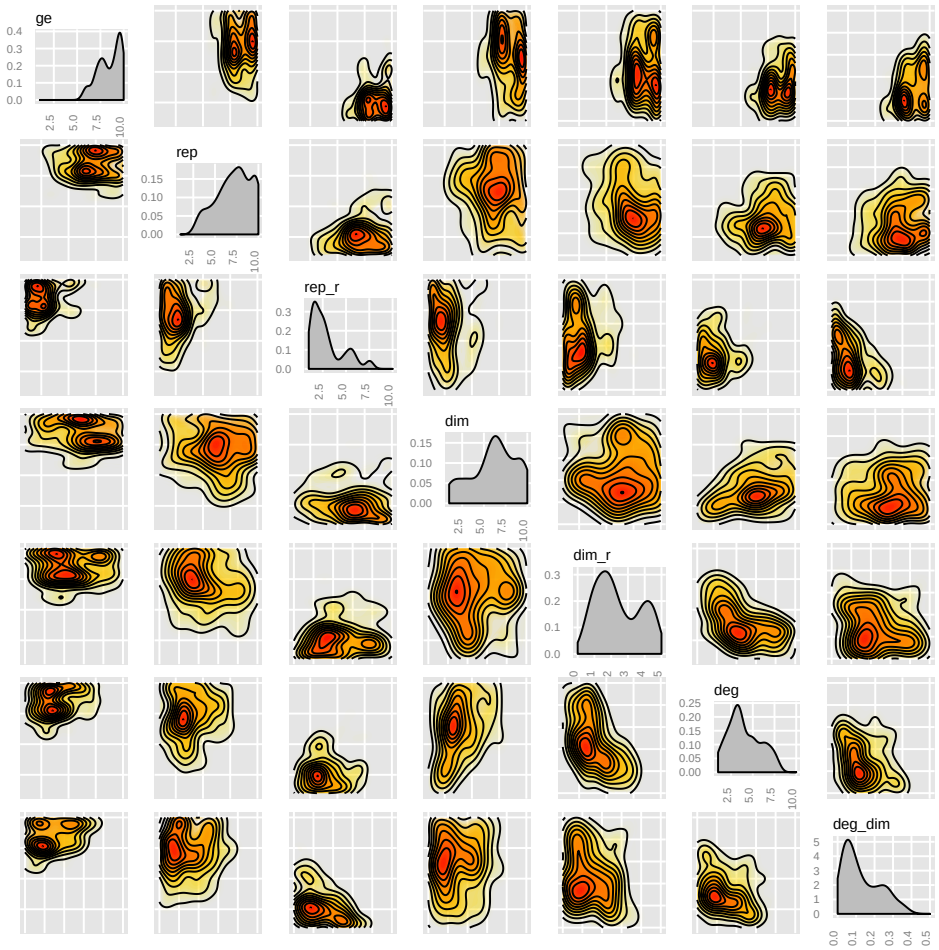

Figure 1.20 The posterior distribution of the mass action simple toggle switch

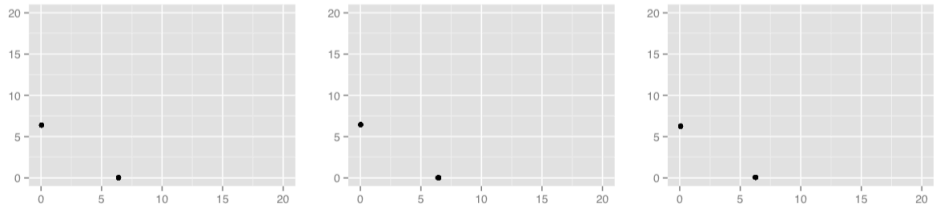

Figure 1.21 A sample of the phase plots produced from the final population of the mass action simple toggle switch.

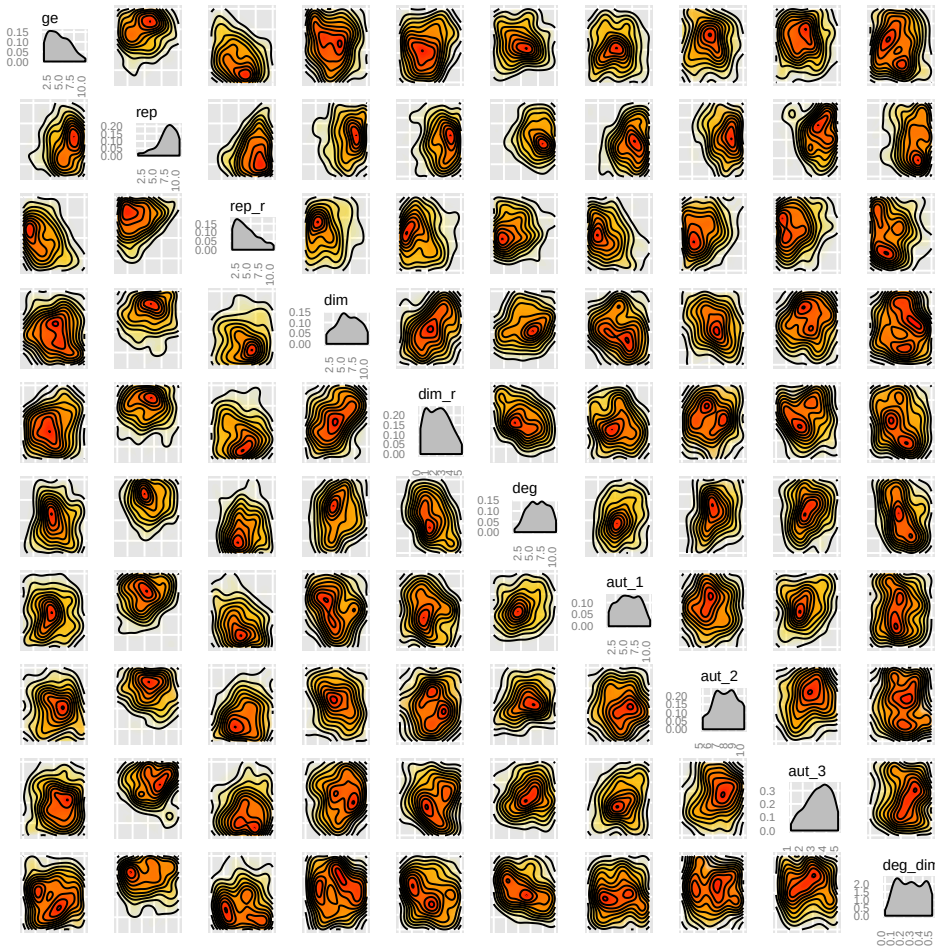

Figure 1.22 The posterior distribution of the Mass action double positive feedback loop switch

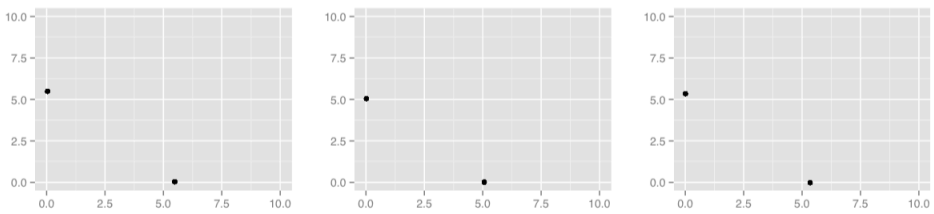

Figure 1.23 A sample of the phase plots produced from the final population of the mass action double positive feedback toggle switch.

In order to directly compare the robustness of the two models, we used the algorithm outlined in the Methods as a measure of how wide each posterior is given the priors. The results are shown in Figure 1.24.

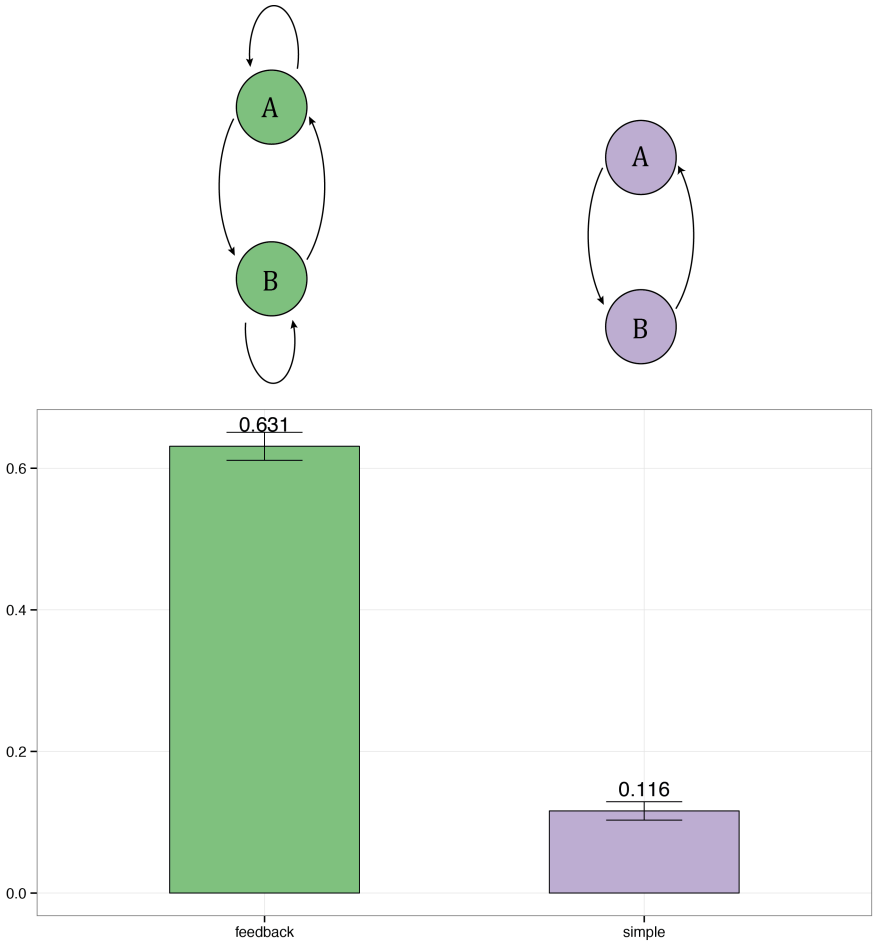

Figure 1.24 The switch with double positive autoregulation is more robust to parameter fluctuations that the simple switch. This is evident from the fact that the switch with added feedback has a larger posterior distribution under which it is bistable, compare to the simple switch. Robustness was calculated using a Monte Carlo accept reject algorithm.

As seen in Figure 1.24, the toggle switch with double positive autoregulation is more robust than the simple switch with no feedback. 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.

Stochastic case

Bistable

The model consists of asymetric parameters for gene expression and repression. We search for the parameter space that makes this model tristable. Example phase plots of the resulting tristable switches are shown below. 1000 particles were used in these simulations.

Table 1.10 priors of bistable switches

parameter	range
geA	0-10
repA	0-10
rep_r	0-10
dim	7-15 (0-20 in double pos)
dim_r	0-10
deg	0-10
deg_dim	0-1
geB	0-10
repB	0-10
aut_1	0-10
aut_2	0-10
aut_3	0-10

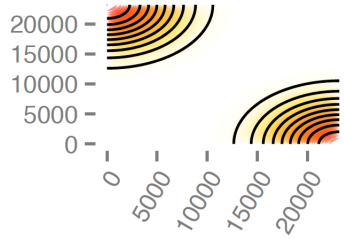

Figure 1.25 Phase plot of the bistable switch.

The posterior distributions of the two swicthes:

Figure 1.26 Posterior of simple switch

Figure 1.27 Posterior of switch with double positive autoregulation

Tristable

The model consists of asymmetric parameters for gene expression and repression. We search for the parameter space that makes this model tristable. Example phase plots of the resulting tristable switches are shown below. 1000 particles were used in these simulations.

Table 1.11 priors of tristable switches

parameter	range
geA	0-10
repA	0-10
rep_r	0-10
dim	0-20
dim_r	0-10
deg	0-10
deg_dim	0-1
geB	0-10
repB	0-10
aut_1	0-10
aut_2	0-10
aut_3	0-10

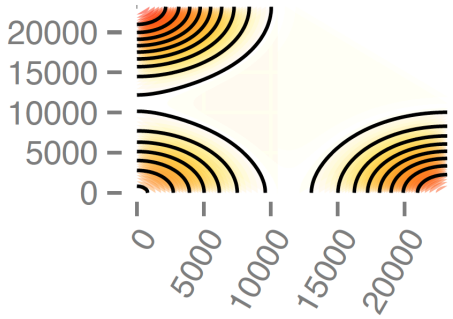

Figure 1.28 Phase plot of the tristable switch.

The posterior distributions of the two switches:

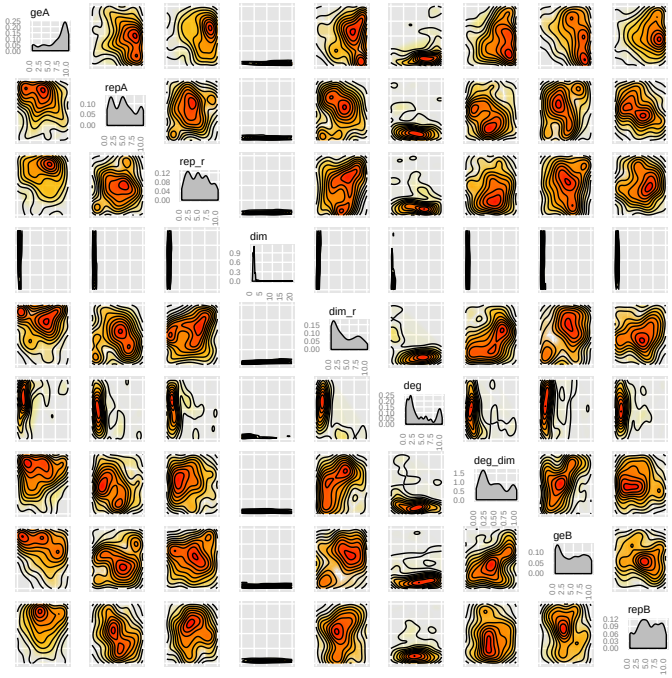

Figure 1.29 Posterior of simple switch

b]

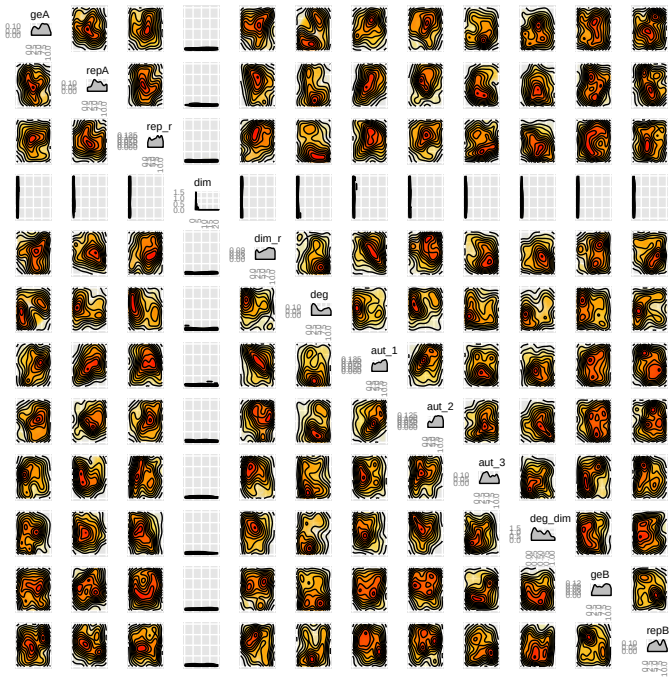

Figure 1.30 Posterior of switch with double positive autoregulation

It is clear that in both cases of the switch, the parameter for dimerisation (dim) is very constrained to low values. On the other and, the values for the reverse reaction, the unbinding of dimerisation is not constrained and can take up larger values.

The robustness of the two switches was compared and no difference was found between the two:

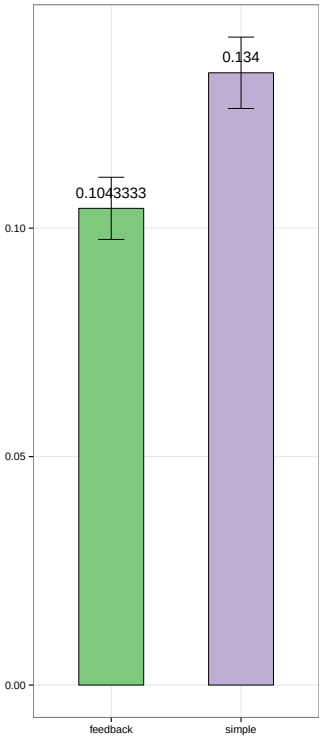

Figure 1.31 Robustness comparison of the tristable switches. There is no significant difference between the two.

Extracting the design principles of a tristable switch

Next, we went on to study the design principles that make a switch tristable vs bistable. To do this we implemented an algorithm, as shown in the Appendix. Samples were taken from the posterior distribution of the stochastic double positive mass action tristable switch. Then we separated the values that, for each parameter, were also found in the posterior of the bistable switch. The same procedure was done but taking samples from the bistable posterior and looking for them in the tristable posterior. This can give an indication of the separation of parameter values that can give rise to a bistable vs a tristable switch. The results are shown in Figure 1.32.

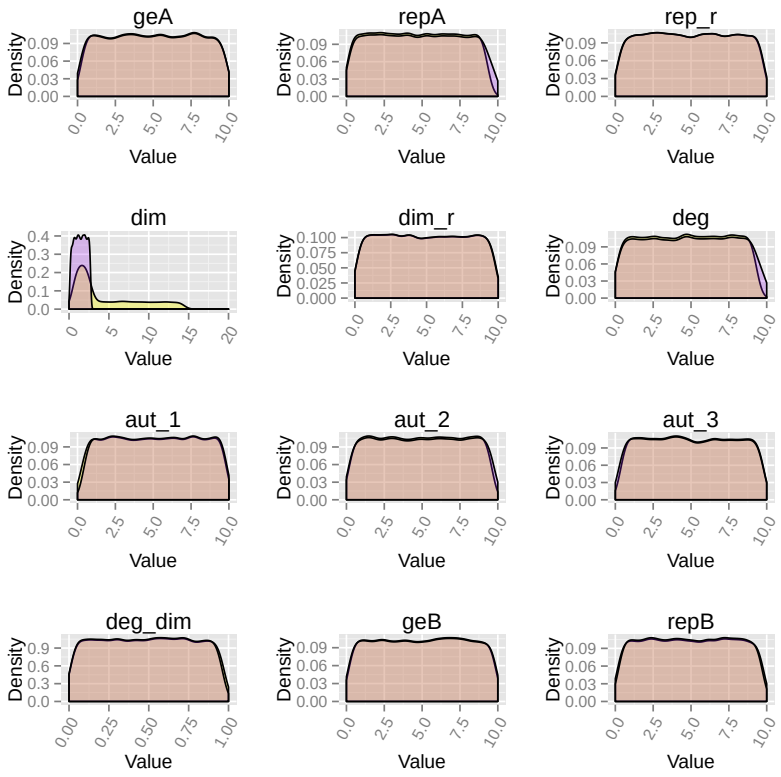

Figure 1.32 Extracting the tristable versus bistable switch design principles. Yellow represents a bistable switch and red a tristable

From the above results it can be seen that the only parameter that is different in the two switches is the one representing dimerisation (dim). For both a bistable and a tristable switch the value for dim must be small, although this condition is more

constrained in a tristable switch.

1.6 Discussion

Here we presented a methodology, StabilityFinder, which can identify the region of parameter space that can produce the stability of choice. We demonstrated its use on some known models and extracted stability and robustness information from them. We compared the stability profile of the Gardner toggle switch when modelled deterministically and stochastically in order to uncover the differences that arise from the addition of noise in the modelling.

We then applied StabilityFinder to the Lu switches in order to uncover the design principles that make a switch bistable versus tristable.

We also applied StabilityFinder to a synthetic biology design problem. We used two models of the switch, one simple model consisting of two mutually repressive transcription factors and a model with added double positive auto-regulation. Comparing the two models, both capable of bistable behaviour, using StabilityFinder we found that the model with added double positive feedback loops is more robust to parameter fluctuations. This makes it a better candidate for building new synthetic devices based on the toggle switch design. We identified the parameter region within which this models are bistable, information that is important when building such a device in the lab. 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 (Salis et al. 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 et al. (2011). These standardised interchangeable components with known sequence and activity are what synthetic biology classes as BioBricks (Kelly et al. 2009; Canton et al. 2008). These can be selected and used to construct a desired system and replicate the parameter values found using StabilityFinder.

The methodology we presented here can be applied to a variety of problems as demonstrated. It can be applied to any problem of finding the parameter values

that can produce a desired stability between two species. It can be used to design new systems of desired stability and help identify the appropriate parts to use by identifying the rates within which these parts need to operate. It can also be used to examine existing systems and give an insight on the underlying mechanisms that allow for the given stability to occur.

Bibliography

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.

Biancalani, T. & Assaf, M. (2015). 'Genetic Toggle Switch in the Absence of Cooperative Binding: Exact Results', 1–5.

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.

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.

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

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.

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

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

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.

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

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

Lu, M., Jolly, M. K., Gomoto, R., Huang, B., Onuchic, J., & Ben-Jacob, E. (2013). 'Tristability in cancer-associated microRNA-TF chimera toggle switch.' *The Journal of Physical Chemistry B* **117**(42), 13164–13174.

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.

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.

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.

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.

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.

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.

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.

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 Clustering algorithms

Deterministic case

Algorithm 4 Clustering the steady state deterministic simulation results

```
for each data point do
    if first point then
        Make first cluster
        cluster counter = 1
    else
        for each cluster do
            if cluster within cluster means ± delta then
                Add to existing cluster
                Update means of clusters
            end if
            if reached_end and not assigned to cluster then
                cluster counter += 1
                Add new cluster
            end if
        end for
    end if
end for
```

Stochastic case

Gap statistic

Algorithm 5 Choosing the optimal number of clusters

function Wk(clusters, cluster_centres)
    for each cluster **do**
        for each point in cluster **do**
            a = *matrixnorm* (cluster_centre − point)
        end for
        $dk = \sum((a)^2) \times$ (2× number of points in cluster)
    end for
    $wk = \frac{\sum(dk)}{2\times(numberofpointsincluster)}$
    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
        $Wkb = \frac{\sum(BWk)}{10}$
        $sk = \sqrt{\sum(\frac{(BWk-Wkb)^2}{10})}$
    end for
    $sk = sk \times \sqrt{1+\frac{1}{B}}$
    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 = $gaps[i] \geq (gaps[i+1] - sk[i+1])$
    return cluster_counter, clusters_means
end function

K-means clustering

Algorithm 6 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
```