

# 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

0.1 Abstract . . . . .	2
<b>List of Figures</b>	<b>4</b>
<b>List of Tables</b>	<b>6</b>
<b>1 Modelling switches</b>	<b>7</b>
1.1 Introduction . . . . .	7
1.2 Methods . . . . .	9
Calculating robustness . . . . .	13
1.3 Results . . . . .	14
Testing StabilityChecker: The Gardner toggle switch . . . . .	14
Comparing the deterministic and stochastic cases . . . . .	16
1.4 Lu toggle switch models . . . . .	20
Classical model . . . . .	20
Self-activation . . . . .	23
Extracting the design principles of a tristable switch . . . . .	25
Positive auto-regulation increases robustness to parameter fluctuations . . . . .	27
1.5 Discussion . . . . .	32
<b>Bibliography</b>	<b>35</b>
<b>2 Appendix</b>	<b>39</b>
2.1 Clustering algorithms . . . . .	40
Deterministic case . . . . .	40
Stochastic case . . . . .	41
Gap statistic . . . . .	41
K-means clustering . . . . .	43

# List of Figures

1.1	Flowchart of the algorithm used in StabilityChecker . . . . .	11
1.2	The posterior distribution of the bistable Gardner toggle switch . . . . .	15
1.3	A sample of the phase plots produced from the final population of the Gardner switch. . . . .	16
1.4	The posterior distribution of the deterministic model of the Gardner toggle switch. . . . .	18
1.5	The posterior distribution of the stochastic model of the Gardner toggle switch. . . . .	18
1.6	Comparing the robustness of the deterministic and stochastic Gardner switch models . . . . .	19
1.7	Phase portrait of the Lu classical model with no self activation . . . . .	21
1.8	The posterior distribution of the Lu classical model with no self activation	22
1.9	A sample of the phase plots produced from the final population of the Lu classical model. . . . .	23
1.10	Phase portrait of the Lu model including double self-activation . . . . .	23
1.11	The posterior distribution of the Lu model with double self activation .	24
1.12	A sample of the phase plots produced from the final population of the Lu tristable model. . . . .	25
1.13	Extracting the tristable versus bistable switch design principles . . . . .	26
1.14	The phase plots produced by the last population of StabilityChecker. By changing only the parameters $g_x$ and $g_y$ in the Lu model with self-activation, the switch became a bistable switch, instead of a tristable switch. . . . .	26
1.15	The posterior distribution of the mass action simple toggle switch . . . . .	29
1.16	A sample of the phase plots produced from the final population of the mass action simple toggle switch. . . . .	29

1.17 The posterior distribution of the Mass action double positive feedback loop switch . . . . .	30
1.18 A sample of the phase plots produced from the final population of the mass action double positive feedback toggle switch. . . . .	30
1.19 The robustness of the double positive and the simple switch mass action models . . . . .	31

## List of Tables

1.1	Gardner switch priors . . . . .	15
1.2	Gardner switch priors . . . . .	17
1.3	Lu classical model parameter values . . . . .	21
1.4	Lu classical model priors . . . . .	22
1.5	Lu model with self-activation parameter values . . . . .	23
1.6	Lu model with double self-activation priors . . . . .	24
1.7	Mass Action switches priors . . . . .	29

# 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 a 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 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,

## 8 MODELLING SWITCHES

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. 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 if 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.

In order to solve this problem, we created a framework, StabilityChecker, 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 StabilityChecker 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 StabilityChecker 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,

## 10 MODELLING SWITCHES

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,  $\pi(\theta | x) \propto f(x | \theta)\pi(\theta)$ , where  $f(x | \theta)$  is the likelihood of a parameter,  $\theta$ , given the data,  $x$ . There are a number of different variations of the ABC algorithm. The generic ABC algorithm is as follows:

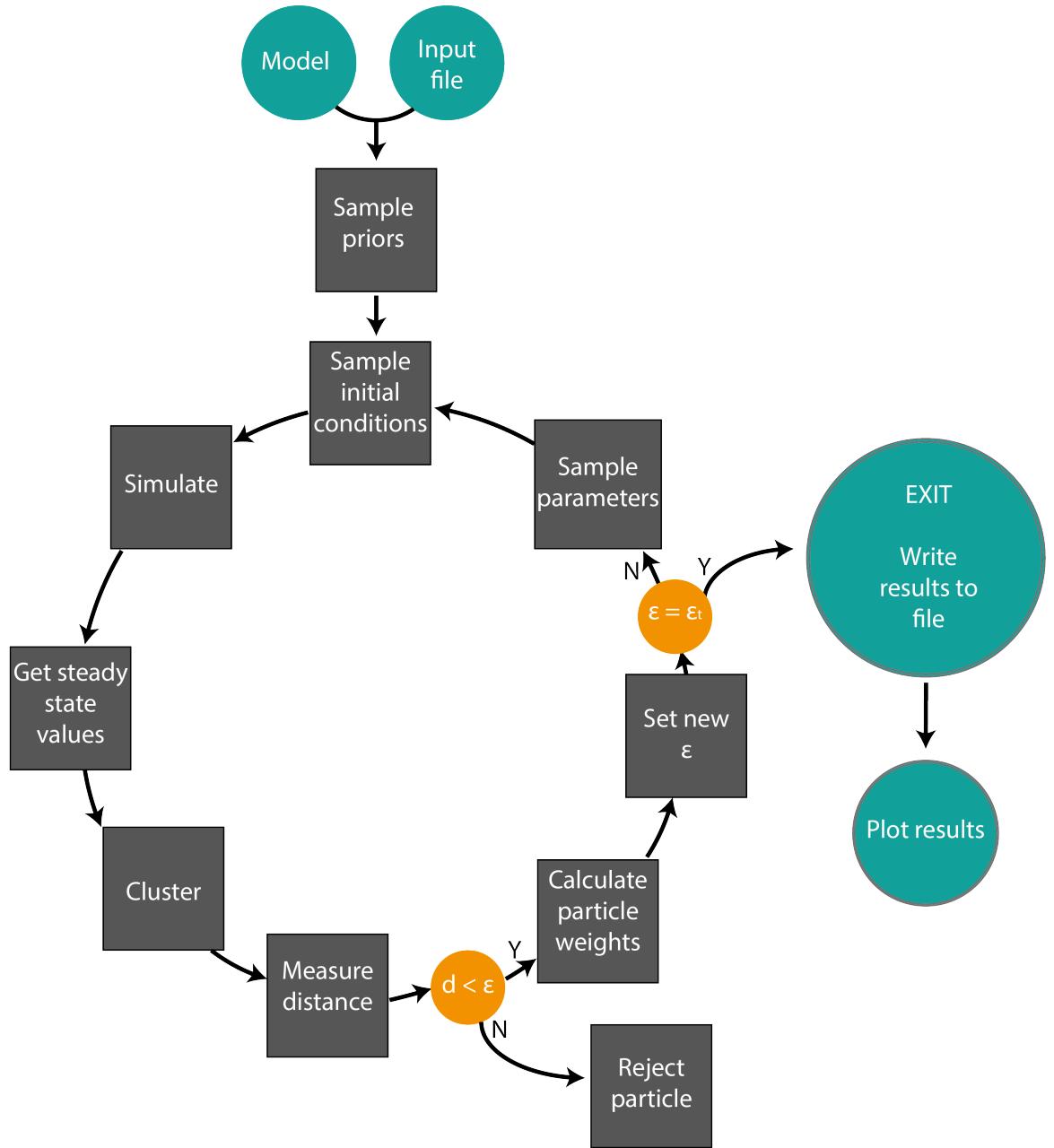
---

**Algorithm 1** Generic ABC algorithm

---

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

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 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,  $\epsilon$  for the distance of the simulated data to the desired data is made smaller at each iteration. When  $\epsilon$  is sufficiently small, the result will approximate the posterior distribution (Toni et al. 2009). In StabilityChecker the desired behaviour in ABC SMC is replaced by a desired stability that the simulated model has to have to be accepted. The  $\epsilon$  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 can be summarised as seen in Figure 1.1.



**Figure 1.1** Flowchart of the algorithm used in StabilityChecker.  $\epsilon$  stands for the threshold for the distance from the desired value.

## 12 MODELLING SWITCHES

The user provides the model file, in the form of SBML or 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 algorithm 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  $\epsilon$ , 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  $\epsilon$  is set to smaller value, thus eventually reaching the desired behaviour. The algorithm's pseudocode is shown in Algorithm ??.

---

**Algorithm 2** StabilityChecker

---

```

1: Initialise  $\epsilon$ 
2: population p  $\leftarrow$  1
3: if p = 1 then
4:     Sample particles ( $\theta$ ) 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  $\theta$ 
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  $\epsilon \leq \epsilon_T$ 

```

---

## Calculating robustness

We use the results from StabilityChecker 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:    $accepted += 1$ 
5: end if
6:  $acceptance\_rate = \frac{accepted}{numberofsamples}$ 
7: return  $acceptance\_rate$ 

```

---

### 1.3 Results

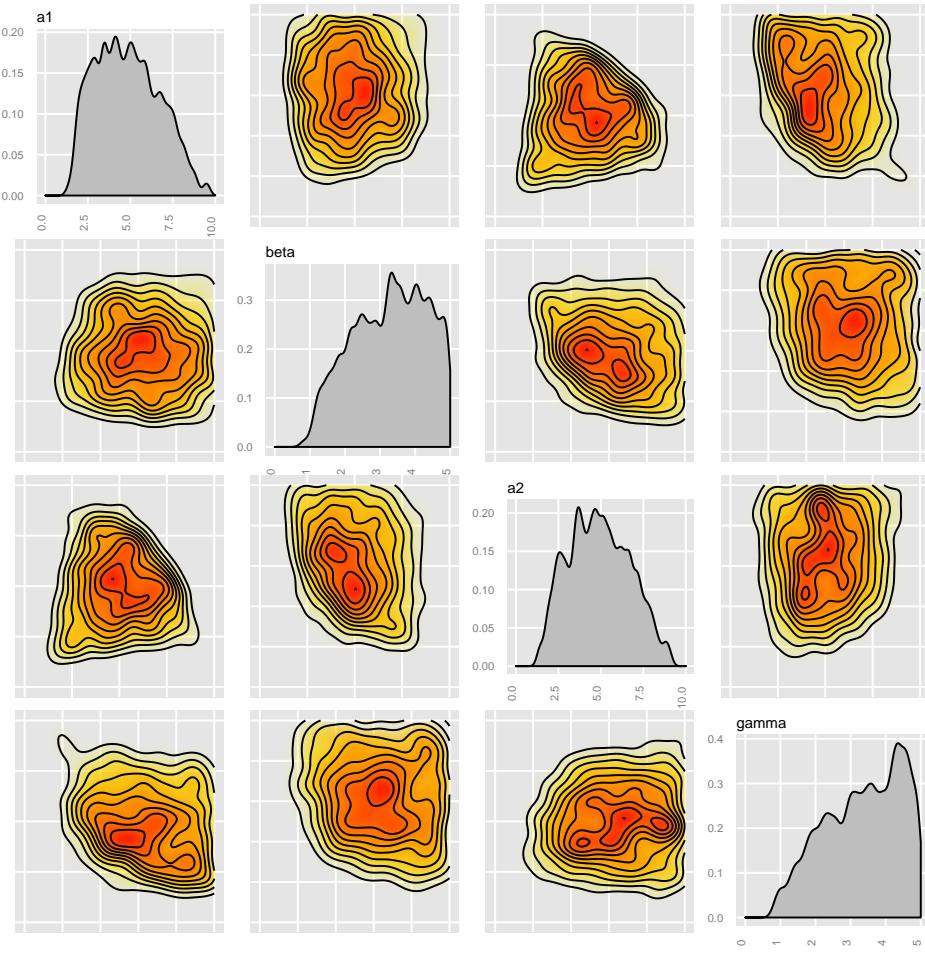
#### Testing StabilityChecker: The Gardner toggle switch

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:

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

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

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.  $\beta$  is the cooperativity of repression of promoter 1 and  $\gamma$  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  $\beta$  and  $\gamma$  are  $>1$ . In order to test our methodology, we used StabilityChecker 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 StabilityChecker to see if it finds the same conditions as the ones set by T. S. Gardner. The prior distributions used are shown in Table 1.1. The posterior distribution calculated by StabilityChecker 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 the effective synthesis rate of repressors 1 and 2 respectively. Parameters  $\beta$ ,  $\gamma$  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  $\beta$  and  $\gamma$  must both be  $>1$ , as can be seen in the marginal distributions of  $\beta$  and  $\gamma$  in Figure 1.2. The conditions set by the original paper for parameters  $a_1$  and  $a_2$  are met, as the

**Table 1.1** Gardner switch priors

Parameters				Species	
$a_1$	$\beta$	$a_2$	$\gamma$	$s_1$	$s_2$
0-10	0-5	0-10	0-5	0-100	0-100

## 16 MODELLING SWITCHES

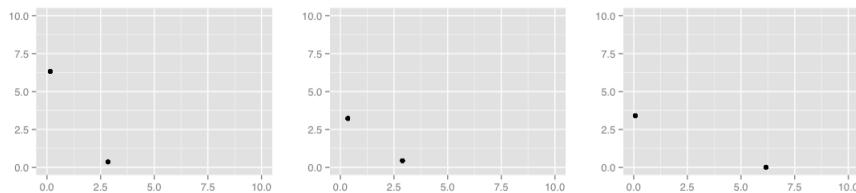


Figure 1.3 A sample of the phase plots produced from the final population of the Gardner switch.

joint distribution shown in Figure 1.2 matches the bifurcation lines calculated in the original paper. This successful test demonstrates that StabilityChecker 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).

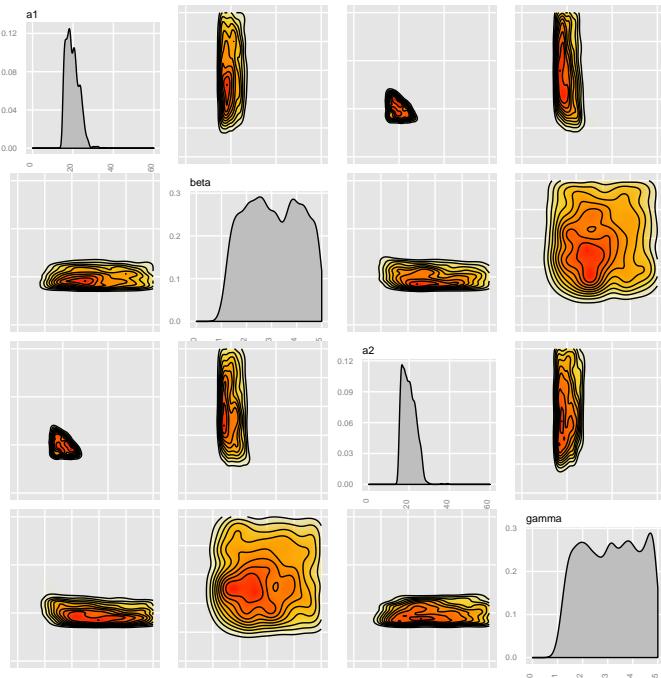
The toggle switch has been modelled both deterministically and stochastically,

with the two methods producing different results. Thus here we use StabilityChecker 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 biological model in which small molecule numbers and external noise are not negligible. Using StabilityChecker and using the same priors for the two models (Table 1.2), we calculated the posterior for each both bistable models. The results are shown in Figures 1.4, 1.5. 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.

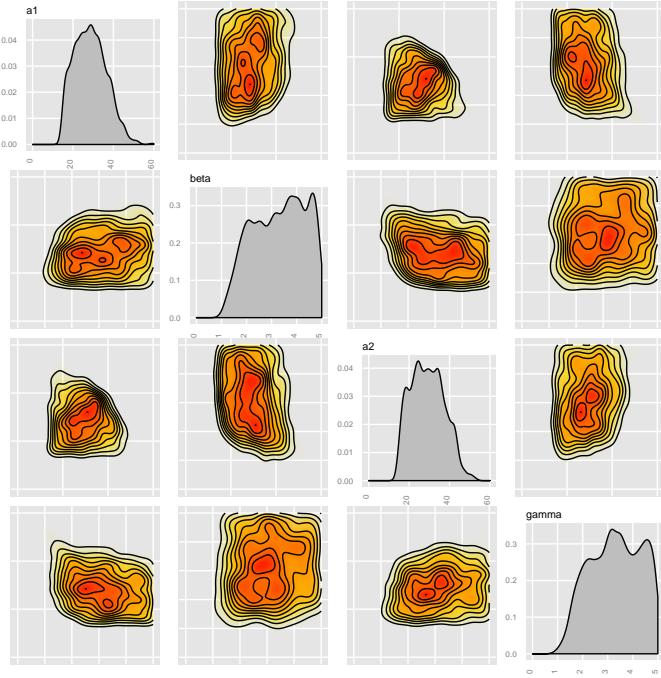
**Table 1.2** Gardner switch priors

Parameters				Species	
$a_1$	$\beta$	$a_2$	$\gamma$	$s_1$	$s_2$
0-60	0-5	0-60	0-5	0-100	0-100

## 18 MODELLING SWITCHES

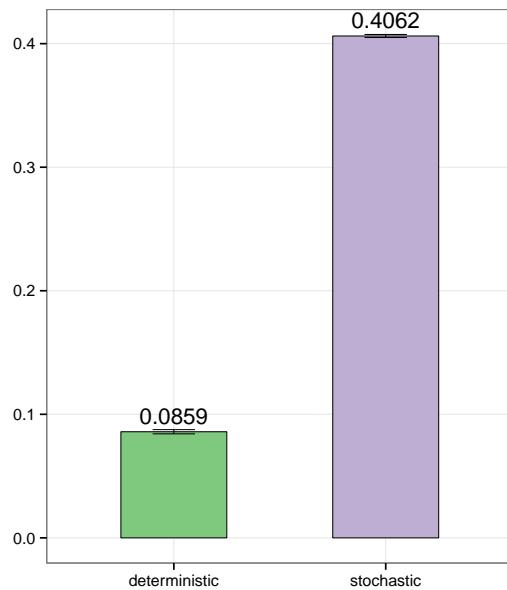


**Figure 1.4** The posterior distribution of the deterministic model of the Gardner toggle switch.



**Figure 1.5** The posterior distribution of the stochastic model of the Gardner toggle switch.

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  $\beta, \gamma > 1$  also needs to be met in the stochastic case.



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

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.

## 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 StabilityChecker we can determine the design principles that make a tristable versus a bistable switch. This is another example of a use for StabilityChecker. The system used in their study is defined by two dynamical systems:

$$\dot{x} = f_x(x, y) = g_x H_{xy}^S(y) H_{xx}^S(x) - k_x x \quad (1.3)$$

$$\dot{y} = f_y(x, y) = g_y H_{yx}^S(x) H_{yy}^S(y) - k_y y \quad (1.4)$$

$$H_{xx}^S = H_x^-(x) + \lambda_x H_x^+(x) \quad (1.5)$$

$$H_x^-(x) = 1 / [1 + (x/a_x)^{n_x}] \quad (1.6)$$

$$H_x^+(x) = 1 - H_x^-(x) \quad (1.7)$$

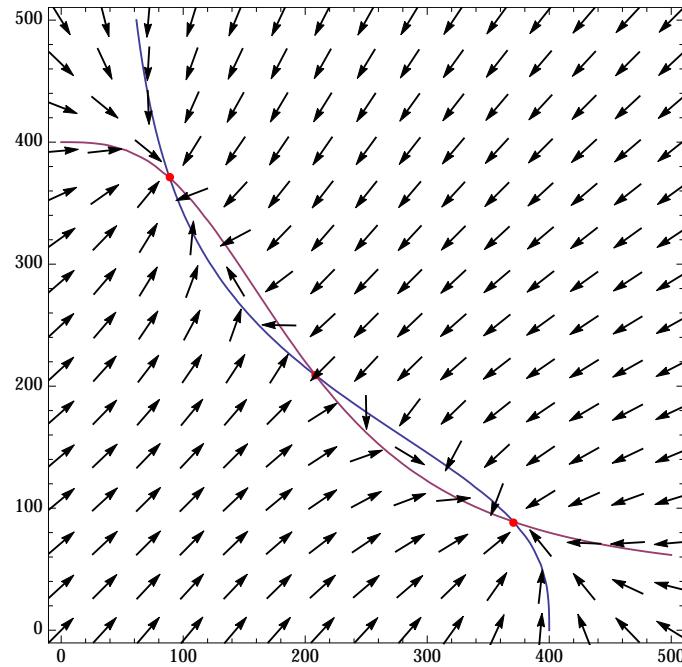
### Classical model

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

$$\dot{x} = f_x(x, y) = g_x H_{xy}^S(y) - k_x x, \quad (1.8)$$

$$\dot{y} = f_y(x, y) = g_y H_{yx}^S(x) - k_y y \quad (1.9)$$

For the parameter values used in the Lu study, as shown in Table 1.3, the system exhibits three steady states (Figure 1.7), of which two are stable and one is unstable.



**Figure 1.7** Phase portrait of the Lu classical model with no self activation. There are two stable steady states and one unstable steady state.

**Table 1.3** Lu classical model parameter values

gx	gy	kx	ky	nxy	nyx	xx	xy	Ixy	Iyx
40	40	0.1	0.1	3	3	200	200	0.1	0.1

## 22 MODELLING SWITCHES

Using StabilityChecker with priors centred around the parameter values used in the original paper (Table 1.4), we can find the robustness of this bistable behaviour, as well as identify the most important parameters for bistability.

Table 1.4 Lu classical model priors

$g_x$	$g_y$	$k_x$	$k_y$	$n_{xy}$	$n_{yx}$	$x_{xy}$	$x_{yx}$	$I_{xy}$	$I_{yx}$
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 1.8. As can be seen from the posterior, the most restrained parameters are  $k_x$  and  $k_y$ , 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.

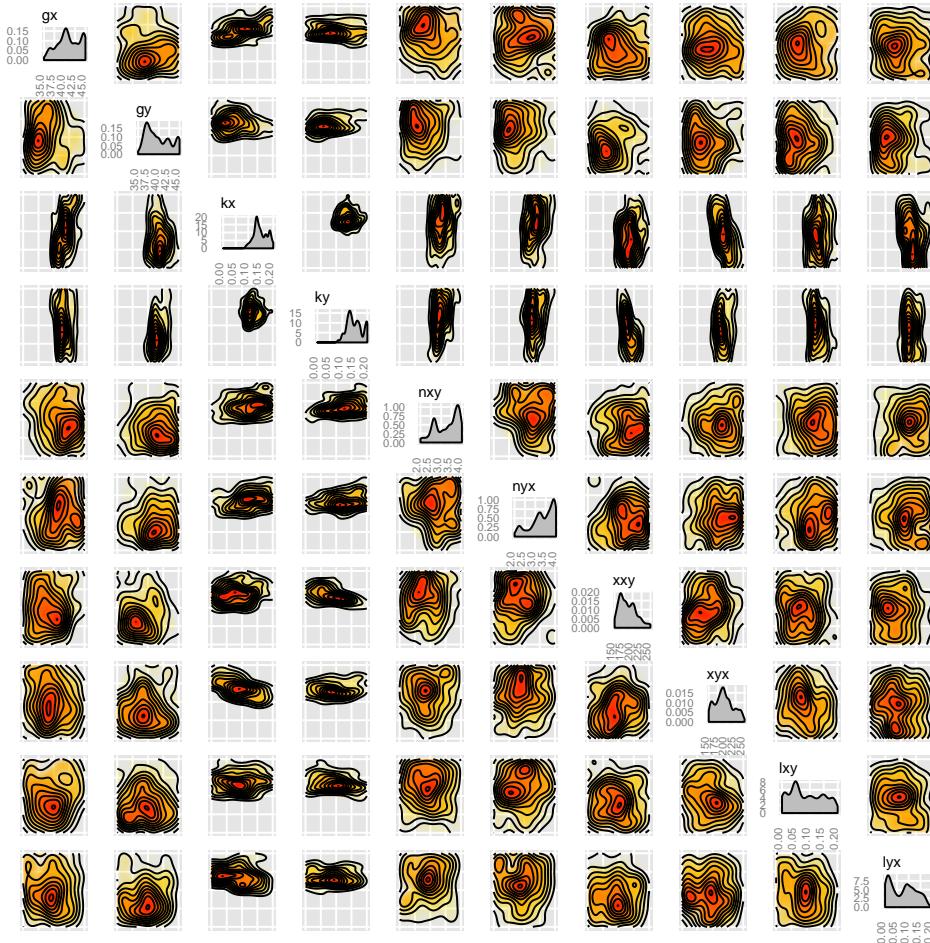
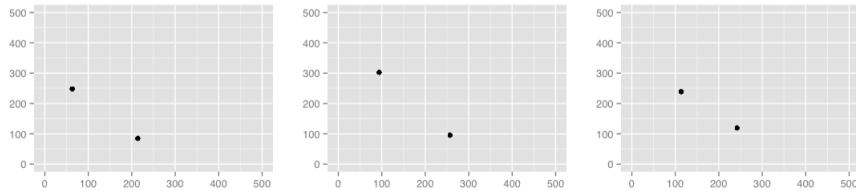


Figure 1.8 The posterior distribution of the Lu classical model with no self activation.  $k_x$  and  $k_y$  are the most constrained parameters.



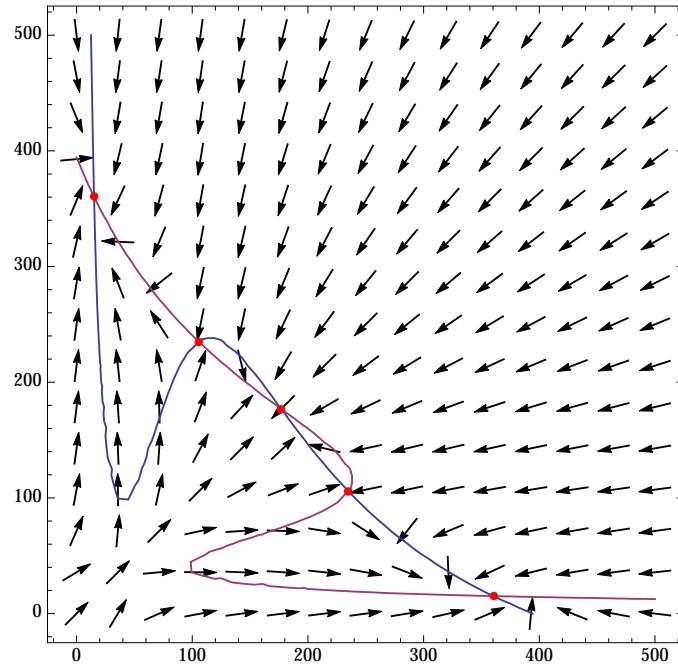
**Figure 1.9** A sample of the phase plots produced from the final population of the Lu classical model.

### Self-activation

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 1.5 are assigned to the parameters for the self-activating model, the system exhibits five steady states as seen in Figure 1.10.

**Table 1.5** 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 1.10** Phase portrait of the Lu model including double self-activation. The model had three stable steady states and two unstable steady states.

## 24 MODELLING SWITCHES

Using StabilityChecker and priors centred around the original values, shown in Table 1.6, we can explore the robustness of this behaviour, in a similar way as done for the classical model.

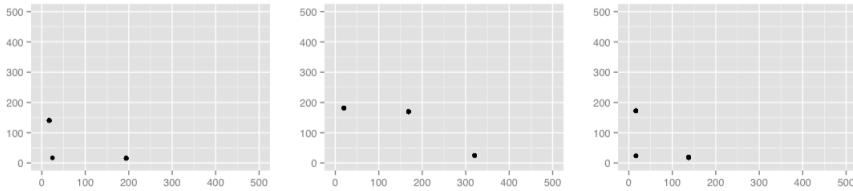
**Table 1.6** Lu model with double self-activation priors

gx	gy	kx	ky	nxy	nyx	xx	xy	Ixy	Iyx
3-5	3-5	0-0.2	0-0.2	0-2	0-2	150-250	150-250	0-0.2	0-0.2

The posterior is shown in Figure 1.11. The most constrained parameters are kx and ky, similar as in the classic toggle switch case.



**Figure 1.11** The posterior distribution of the Lu model with double self activation. kx and ky are the most constrained parameters.

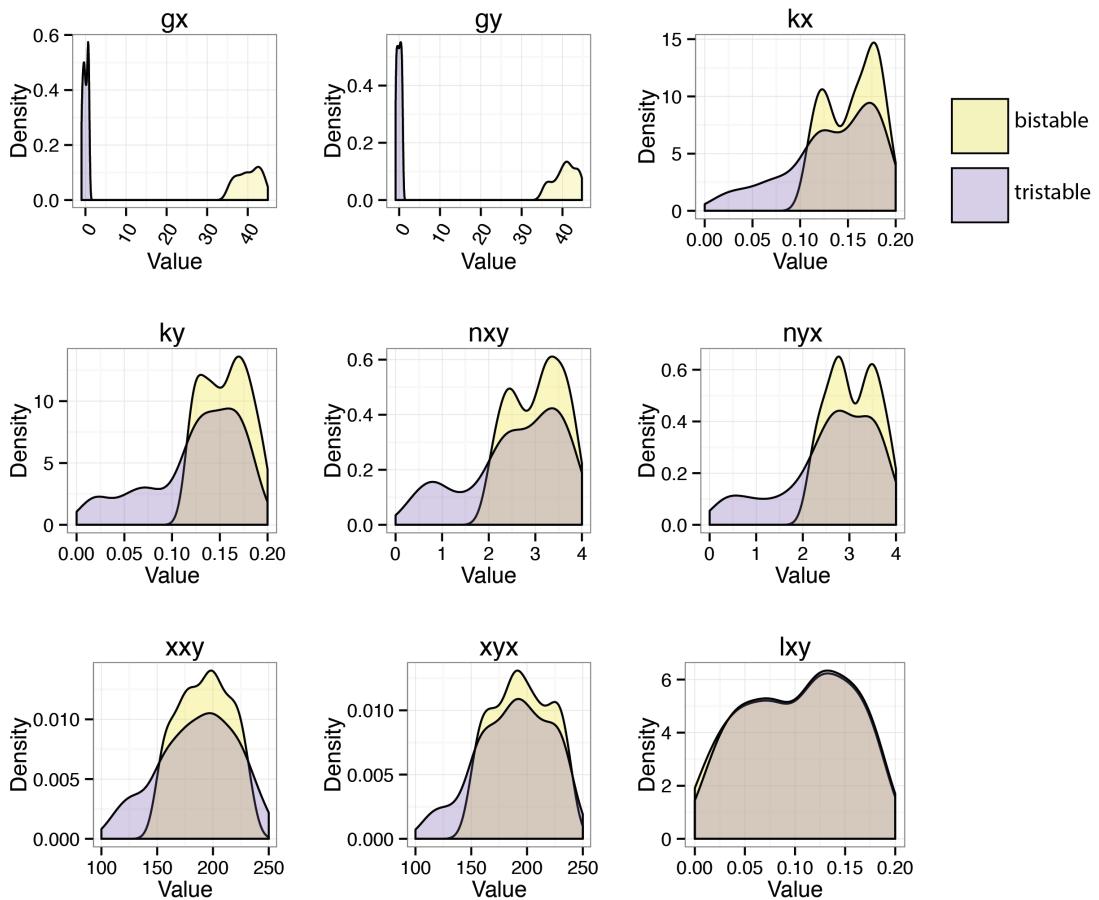


**Figure 1.12** A sample of the phase plots produced from the final population of the Lu tristable model.

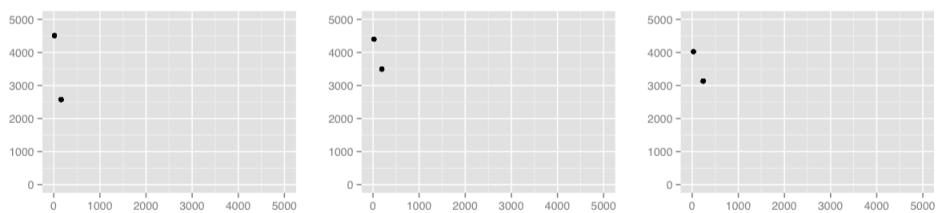
### Extracting the design principles of a tristable switch

Next, we went on to study the design principles that make a switch tristable vs bistable. To do this we implemented an algorithm, as shown in the Appendix. Samples were taken from posterior distribution of the Lu tristable switch. Then we separate the values, for each parameter, that 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 1.13.

It is clear from the above results that the values for  $g_x$  and  $g_y$  are very different for producing a bistable versus a tristable switch. In order to further test this result, we used the model with self activation, with the same priors as used in the tristable case (Table 1.5), except for the values of  $g_x$  and  $g_y$ , which we swapped for the values found to produce a bistable switch. This new model was then analysed using StabilityChecker and was found to be bistable. This indicates that the values of  $g_x$  and  $g_y$ , representing gene expression, are critical for the design of a switch with a given stability, and changing their values can make a tristable switch bistable. This can be seen in the phase plots produced, shown in Figure 1.14



**Figure 1.13** Extracting the tristable versus bistable switch design principles



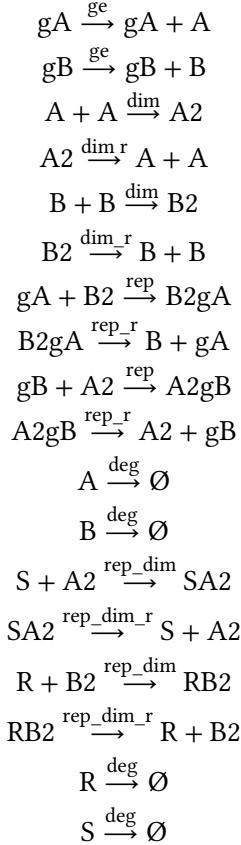
**Figure 1.14** The phase plots produced by the last population of StabilityChecker.  
By changing only the parameters  $gx$  and  $gy$  in the Lu model with self-activation, the switch became a bistable switch, instead of a tristable switch.

## Positive auto-regulation increases robustness to parameter fluctuations

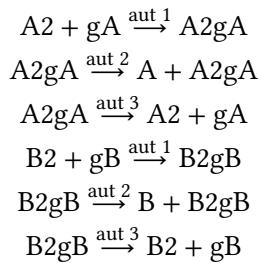
When faced with a set of competing designs for a given genetic circuit, one is likely to choose the simplest possible model that can achieve the desired behaviour. However, simple systems are often the least robust. Here we define a robust device as a device that can withstand fluctuations in parameter 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 StabilityChecker 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:



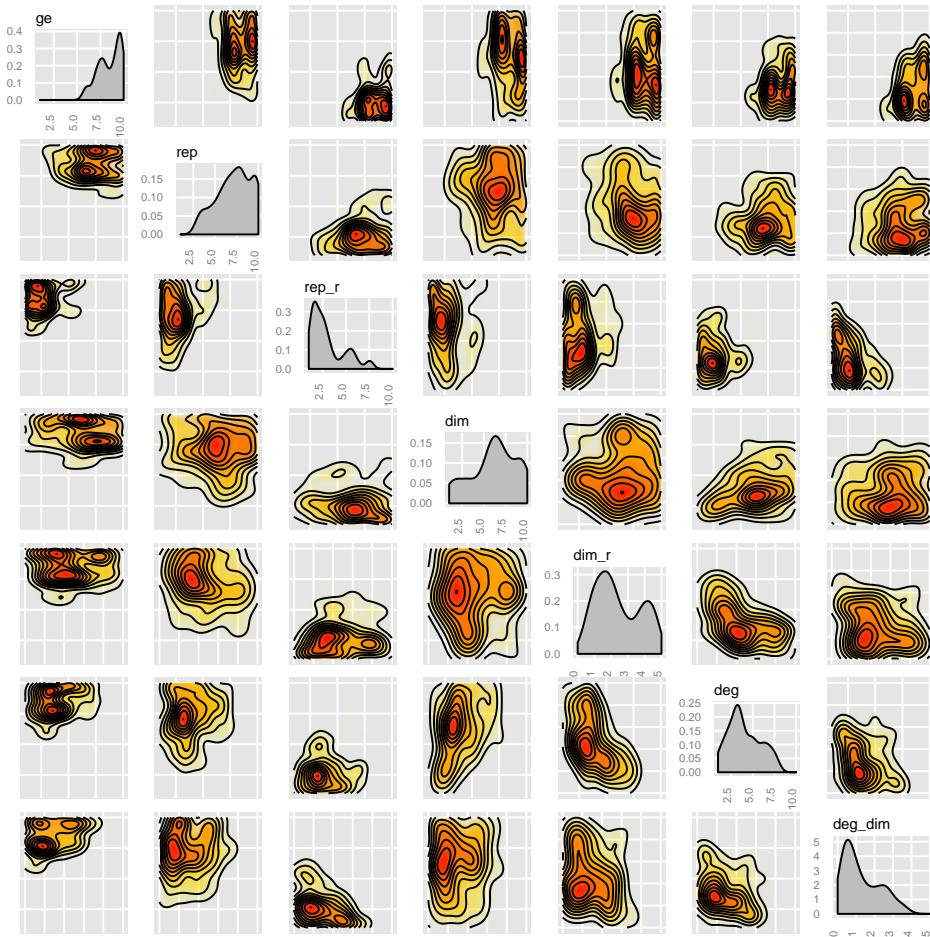
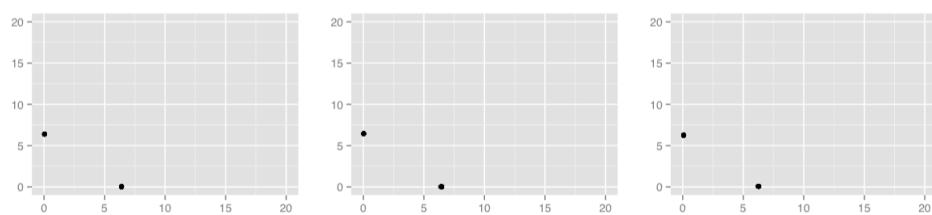
Double positive autoregulation:

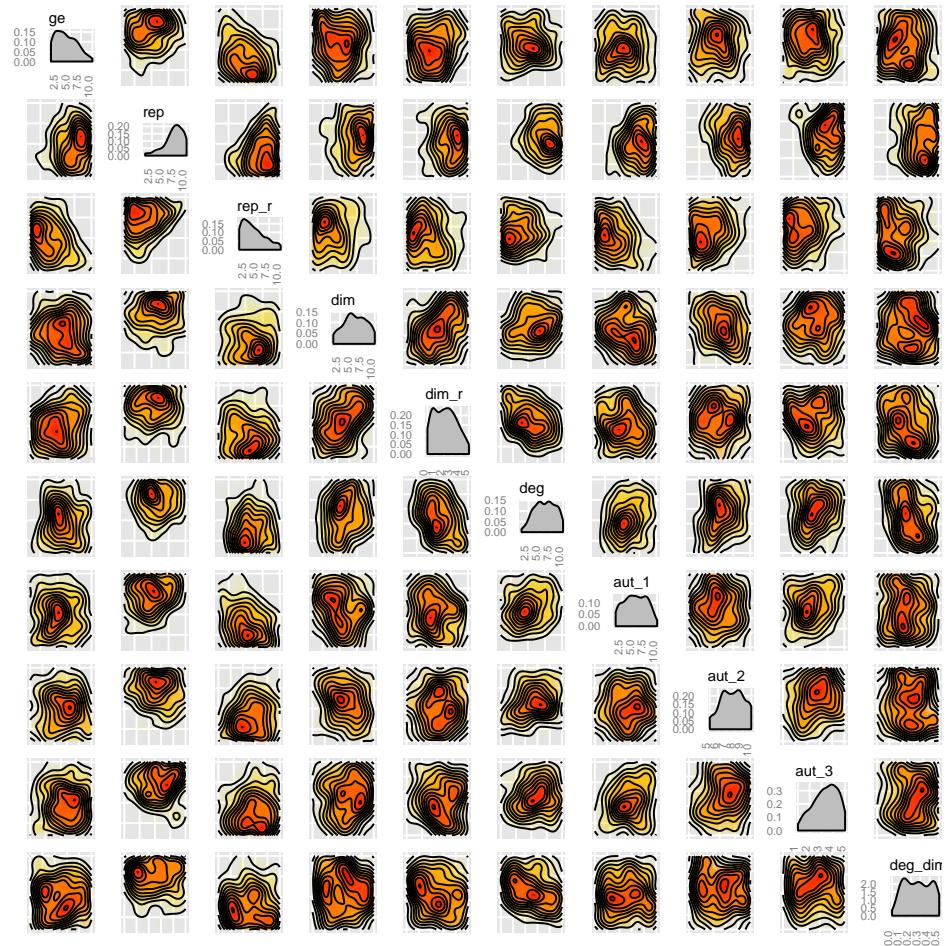


Using the same priors for their shared parameters, we used StabilityChecker for both models in order to compare their robustness. The corresponding posterior distributions can be seen in Figures 1.15 and 1.17.

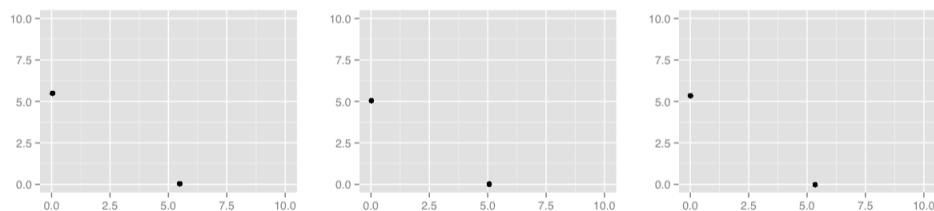
**Table 1.7** Mass Action switches priors

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 1.15** The posterior distribution of the mass action simple toggle switch**Figure 1.16** A sample of the phase plots produced from the final population of the mass action simple toggle switch.

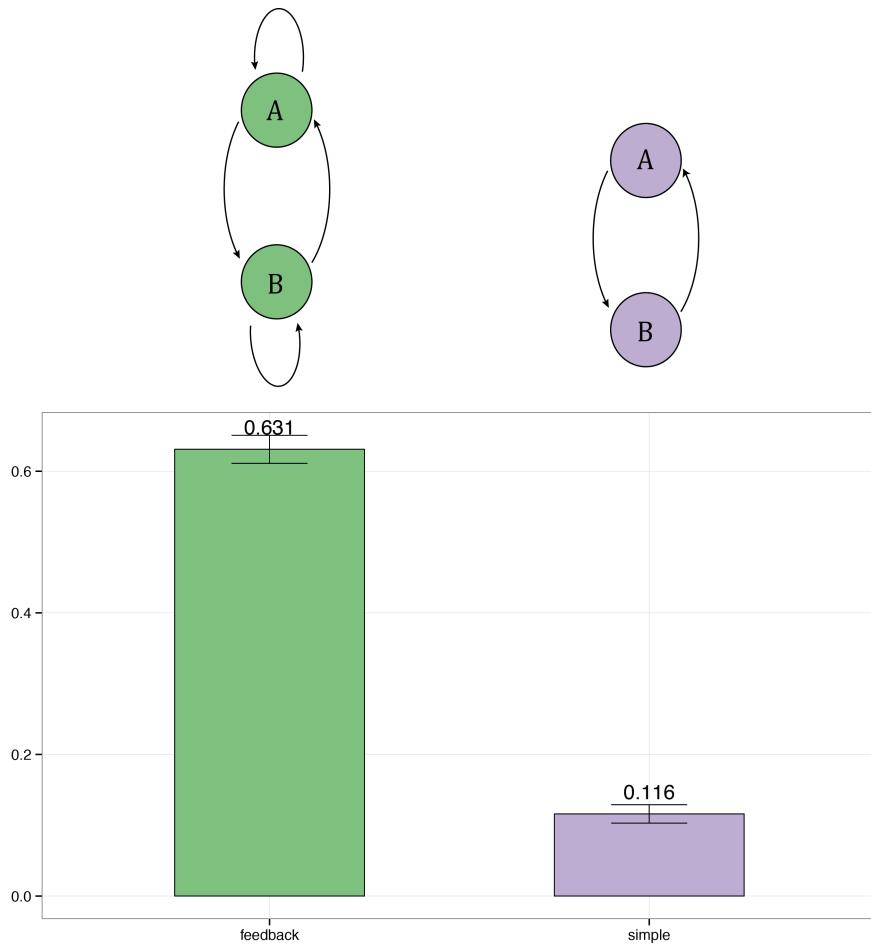


**Figure 1.17** The posterior distribution of the Mass action double positive feedback loop switch



**Figure 1.18** 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 1.19.



**Figure 1.19** The switch with double positive autoregulation is more robust to parameter fluctuations than 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 1.19, 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.

## 1.5 Discussion

Here we presented a methodology, StabilityChecker, 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 found that the stochastic model showed increased robustness to noise.

We then applied StabilityChecker to the Lu switches in order to uncover the design principles that make a switch bistable versus tristable. We found that the gene expression parameters were critical for making a tristable switch bistable. This would be in agreement with the dynamics of a tristable switch, in which the third steady state occurs when there is a deadlock situation between the two proteins. When there are small numbers of both proteins involved, one represses the production of the other resulting in both promoters being repressed (Ma et al. 2012). Given our result we can extrapolate that a higher rate of protein production eliminates the possibility of this deadlock situation happening. Once a promoter is free to produce protein it will produce it in a fast enough rate so that that protein dominates the system and represses the antagonizing promoter before it has the chance to repress it. This dynamic would explain the fact that when all the priors remained the same but gene expression was increased by an order of magnitude, the tristable switch became bistable.

We also applied StabilityChecker 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 StabilityChecker 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 StabilityChecker.

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.



## 2 Appendix

## 2.1 Clustering algorithms

### Deterministic case

---

**Algorithm 4** 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  $\pm$  delta then
8:         Add to existing cluster
9:
10:      Update means of clusters
11:    end if
12:    if reached_end and not assigned to cluster then
13:      cluster counter += 1
14:      Add new cluster
15:    end if
16:  end for
17: end if
18: end for

```

---

**Stochastic case**

**Gap statistic**

---

**Algorithm 5** Choosing the optimal number of clusters
 

---

```

1: function WK(clusters, cluster_centeres)
2:   for each cluster do
3:     for each point in cluster do
4:       a = matrix norm (cluster_center - point)
5:     end for
6:     dk =  $\sum((a)^2) \times (2 \times \text{number of points in cluster})$ 
7:   end for
8:    $wk = \frac{\sum(dk)}{2 \times (\text{number of points in cluster})}$ 
9:   return wk
10: end function

11: function BOUNDING_BOX(data)
12:   xmin, xmax = min and max of x axis in data
13:   ymin, ymax = min and max of y axis in data
14:   return ((xmin, xmax), (ymin, ymax))
15: end function

16: function GAP_STATISTIC(data, cutoff)
17:   (xmin, xmax), (ymin, ymax) = BOUNDING_BOX(data)
18:   ks = [1, 2, 3, 4]
19:   for k in ks do
20:     cluster_centeres, clusters = KMEANS(data, k, cutoff)
21:     Wk = log(WK(clusters, cluster_centeres))
22:     Create references datasets:
23:     for i in range(10) do
24:       for n in range(length of data) do
25:         Xb = random value between xmin, xmax and ymin, ymax
26:       end for
27:       cluster_centeres, clusters = KMEANS(data, k, cut - off)
28:       BWk = log(WK(clusters, cluster_centeres))
29:     end for
30:     Wkb =  $\frac{\sum(BWk)}{10}$ 
31:     sk =  $\sqrt{\sum\left(\frac{(BWk - Wkb)^2}{10}\right)}$ 
32:   end for
33:   sk = sk  $\times \sqrt{1 - \frac{1}{B}}$ 
34:   return ks, Wk, Wkb, sk, data_centeres, clusters
35: 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  $\leftarrow$  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  $\leq$  cutoff then
25:      break
26:    end if
27:  end while
28:  return cluster_centres, clusters
29: end function

```

---