

# Contents

1	Toggle switch stability . . . . .	3
1.1	Introduction . . . . .	3
1.2	Background . . . . .	3
1.2.1	ABC algorithms . . . . .	5
1.2.2	Robustness . . . . .	9
1.3	Stability Finder algorithm . . . . .	9
1.3.1	Algorithm overview . . . . .	9
1.3.2	Particle sampling . . . . .	13
1.3.3	Perturbation . . . . .	13
1.3.4	Initial condition sampling . . . . .	13
1.3.5	Particle simulation . . . . .	15
1.3.6	Distance function . . . . .	15
1.3.6.1	Clustering methods . . . . .	15
1.3.7	Particle rejection . . . . .	16
1.3.8	Weight calculation . . . . .	16
1.3.9	Model checking . . . . .	16
1.4	Calculating robustness . . . . .	16
1.4.1	Case study 1: Infectious diseases . . . . .	18
1.4.2	Case study 2: Population growth . . . . .	21
1.5	Applications of StabilityFinder . . . . .	23
1.5.1	Testing StabilityFinder . . . . .	23
1.5.2	Lu toggle switch models . . . . .	27
1.5.3	Mass Action switches . . . . .	36
1.6	Discussion . . . . .	41
1.7	Summary . . . . .	42
	Bibliography . . . . .	43



# 1 Toggle switch stability

## 1.1 Introduction

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

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

## 1.2 Background

Synthetic biology is now entering an age where simple synthetic circuits have been built, such as toggle switches (Kramer:2004kq; Ham:2008hh; Deans:2007cya; Friedland:2009ce; Gardner, Cantor, & Collins 2000; Isaacs et al. 2003), oscillators (Stricker:2008jqa; Fung:2005jd; Tigges:2009jx) and pulse generators (Basu:2004gn), but larger circuits have proven more difficult (XXX). The leap from building low-level circuits to assembling them into complex networks has yet to be made successfully (Lu:2009ez), and predictable circuit behaviour remains challenging (XXX). Efforts to do so are plagued by intra-circuit crosstalk and incompatibility, as well as cellular noise, which can render synthetic networks non-functional *in vivo* (XXX).

A circuit must be robust to a fluctuating cellular environment and its response and sensitivity must be able to be fine tuned in order to orchestrate a network of

#### 4 TOGGLE SWITCH STABILITY

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

Both analytical and computational approaches have been deployed for the study of the toggle switch. Analytical approaches are limited to simpler models and thus require a number of assumptions to be made. The system under consideration has to be reduced to very few equations and parameters in order to make the system solvable. This requires assumptions to be made about the system that cannot always be justified, such as the quasi-steady state approximation (QSSA). The QSSA assumes that the binding/unbinding processes are much faster than any other process (Loinger et al. 2007), thus the bound intermediate is assumed to always be in steady state. The QSSA assumption is met *in vitro* but often does not hold *in vivo* and its misuse can lead to large errors and incorrectly estimated parameters (Pedersen, Bersani, & Bersani 2007). Moreover, it is generally not possible to solve even simple stochastic models analytically, and these methods are restricted to deterministic models. The computational and graph-theoretic approaches developed for the study of multistationarity generally focus on deciding on whether a given system is incapable of producing multiple steady states (Conradi:2007jo; Banaji:2010fh; Feliu:2013dz). For example, Feliu:2013dz developed an approach using chemical reaction theory and generalised mass action modelling (Feliu:2013dz). No approach exists that can handle both deterministic and stochastic systems in an integrated manner.

Here we present a computational framework based on sequential Monte Carlo that takes a model and determines whether it is capable of producing a given number of (stable) steady states and the parameter space that gives rise to the behaviour. Uniquely, this can be done for both deterministic and stochastic models, and also complex models with many parameters, thus removing the need for simplifying assumptions. Our framework can be used for comparing the conclusions drawn

by various modelling approaches and thus provides a way to investigate appropriate abstractions. We have made this framework into a python package, called StabilityFinder. This methodology is used to investigate genetic toggle switches and uncover the design principles behind making a bistable switch, as well as those necessary to make a tristable switch. We find that degradation rates of transcription factors are important for bistability, and that the addition of positive autoregulation can create tristable behaviour and also significantly more robust bistability when feedback strength is well balanced. Modelling transcription and translation allows us to conclude that transcriptional bursting can inhibit bistability, and also that bistability can occur even when the assumptions of time scale separation in the repressor dynamics are not met. We also examine the design principles behind the design of bistable versus tristable switches and highlight the importance of including stochastic dynamics when modelling these systems. Finally we demonstrate the ability of the framework to examine more complex systems and examine the design principles of a three gene switch. These examples demonstrate that StabilityFinder will be a valuable tool in the future design and construction of novel gene networks.

### 1.2.1 ABC algorithms

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

---

**Algorithm 1** ABC rejection 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 main disadvantage of this method is that if the prior distribution is very different from the posterior, the acceptance rate is very low (Toni et al. 2009). An alternative method is the ABC Markov Chain Monte Carlo (MCMC) developed by Marjoram et al. (2003) (Marjoram et al. 2003). The disadvantage of this method is that if it gets stuck in an area of low probability it can be very slow to converge (Sisson:wf).

The method used here is based on sequential Monte Carlo, which avoids both

## 6 TOGGLE SWITCH STABILITY

issues faced by the rejection and MCMC methods. It propagates the prior through a series of intermediate distributions in order to arrive at an approximation of the posterior. The tolerance,  $\epsilon$  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).

The desired behaviour of the system is used as the data to which the candidate model output is compared to Barnes et al. 2011. Given a set of competing models, their associated prior estimates of their parameters and the design specifications, the algorithm is able to rank the models according to how well they describe the data and the posterior probabilities of the parameters Barnes et al. 2011.

ABC SMC can identify the parameter values within a predefined range of values that can achieve the desired behaviour. It works by first sampling at random from the initial range set by the user, i.e. form the prior distribution of values. Each sample from the priors is called a particle. It then simulates the model given those values and compares that to the target behaviour. If the distance between the simulation and the target behaviour is greater than a predefined threshold distance  $\epsilon$ , then the parameter values that produced that simulation are rejected. This is repeated for a predefined number of samples which are collectively referred to as a population. Each particle in a population has a weight associated with it, which represents the probability of it producing the desired behaviour. At subsequent iterations the new samples are obtained from the previous populations and the  $\epsilon$  is set to smaller value, thus eventually reaching the desired behaviour. The algorithm, without model selection, proceeds as follows:

---

**Algorithm 2** ABC SMC algorithm

---

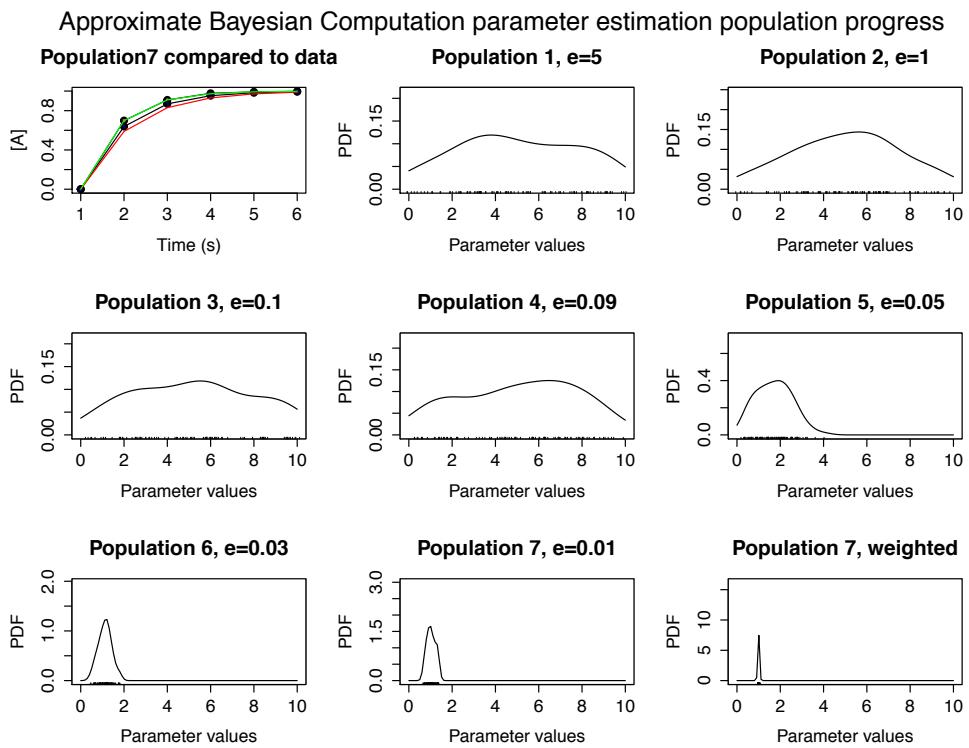
- 1: Select  $\epsilon$  and set population  $t = 0$
  - 2: Sample particles ( $\theta$ ). If  $t = 0$ , sample from prior distributions ( $P$ ). If  $t > 0$ , sample particles from previous population.
  - 3: If  $t > 0$ : Perturb each particle by  $\pm$  half the range of the previous population (j)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

This algorithm is implemented on a simple example for illustration. A simple model was used, consisting of one species,  $A$  converting to another,  $B$ . The model is described by two differential equations, where  $A$  is the reactant and  $B$  the product, produced at a rate  $p1$ .

$$\begin{aligned}\frac{d[B]}{dt} &= p1[A] \\ \frac{d[A]}{dt} &= -p1[A]\end{aligned}$$

The priors were set to  $p1 \sim U(0, 10)$ . Initial conditions for  $A$  and  $B$  were set to 1 and 0 respectively. The data to which the model was compared to was generated by simulating the same model with the parameter set to 1, as shown in Figure 1.1.

## 8 TOGGLE SWITCH STABILITY



**Figure 1.1** ABC SMC parameter inference. The posterior parameter is equal to 1 and its time course shown in red in the top left panel. The blue time course is that of the final population, green is the upper quartile and red is the lower quartile range of values. The progress of the selection process can be seen the eschedule proceeds from the top left to the bottom right. The bottom far right panel is a density plot of  $\epsilon = 0.01$  with their weights taken into account.

### 1.2.2 Robustness

Here we define robustness as..

Robustness coming out of Bayesian methods..

## 1.3 Stability Finder algorithm

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

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

### 1.3.1 Algorithm overview

To investigate the multistable behaviour of systems, a number of extensions to existing approaches are required. For a given set of parameter values, sample points are taken across initial conditions using latin hypercube sampling (XXX), and the ensemble system simulated in time until steady state. The distance function in ABC is replaced by a distance on the desired stability of the simulated model. To do this we cluster the steady state coordinates using Kmeans clustering and use the gap statistic to determine the number of clusters (XXX). The algorithm is summarised below.

## 10 TOGGLE SWITCH STABILITY

---

**Algorithm 3** StabilityFinder algorithm

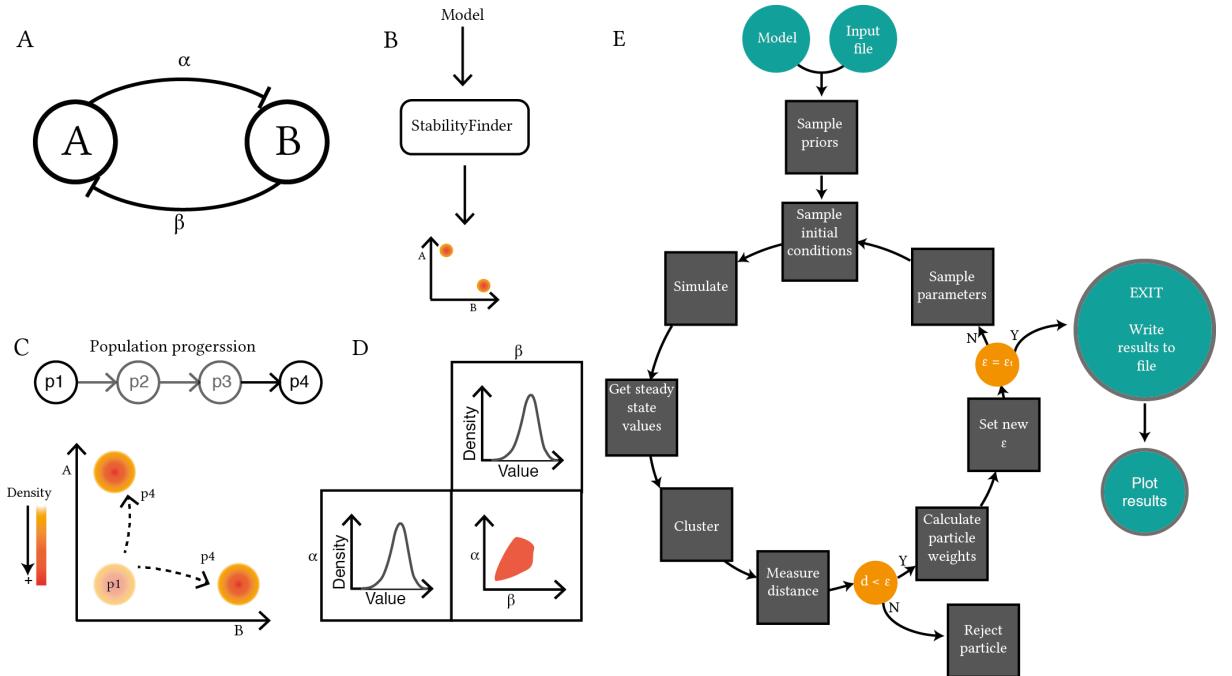
---

```
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: Normalise weights
15: repeat steps 3 - 15
16: until  $\epsilon \leq \epsilon_T$ 
```

---

This algorithm is available as a Python package, called StabilityFinder.

The user provides an SBML model file (XXX) and an input file that contains all the necessary information to run the algorithm, including the desired stability and the final tolerance,  $\epsilon$ , for the distance from the desired behaviour necessary for the algorithm to terminate. The flow of execution is illustrated in Figure 1.2. Since the algorithm is computationally intensive, all deterministic and stochastic simulations are performed using algorithms implemented on Graphical Processing Unit (GPU)s (XXX).



**Figure 1.2** : Using sequential Monte Carlo to examine system stability. The algorithm takes as input a model (A) and evolves it to the stability of choice (C) via intermediate populations. In this example model shown in A, There are two species and two parameters. For the model to be bistable, the phase plot of the two species of interest must have two distinct densities, as shown in (C). The parameter space of the model is searched through our algorithm until the resulting simulations give rise to bistability. The parameter values for the model that demonstrated the desired behaviour are given as an output (D). The output consists of the accepted values for each parameter, as well as each density plotted against the other. This allows us to uncover correlations between parameter values. We made this algorithm into a python package, called Stability Finder. The overview of the algorithm is shown in (E).

## 12 TOGGLE SWITCH STABILITY

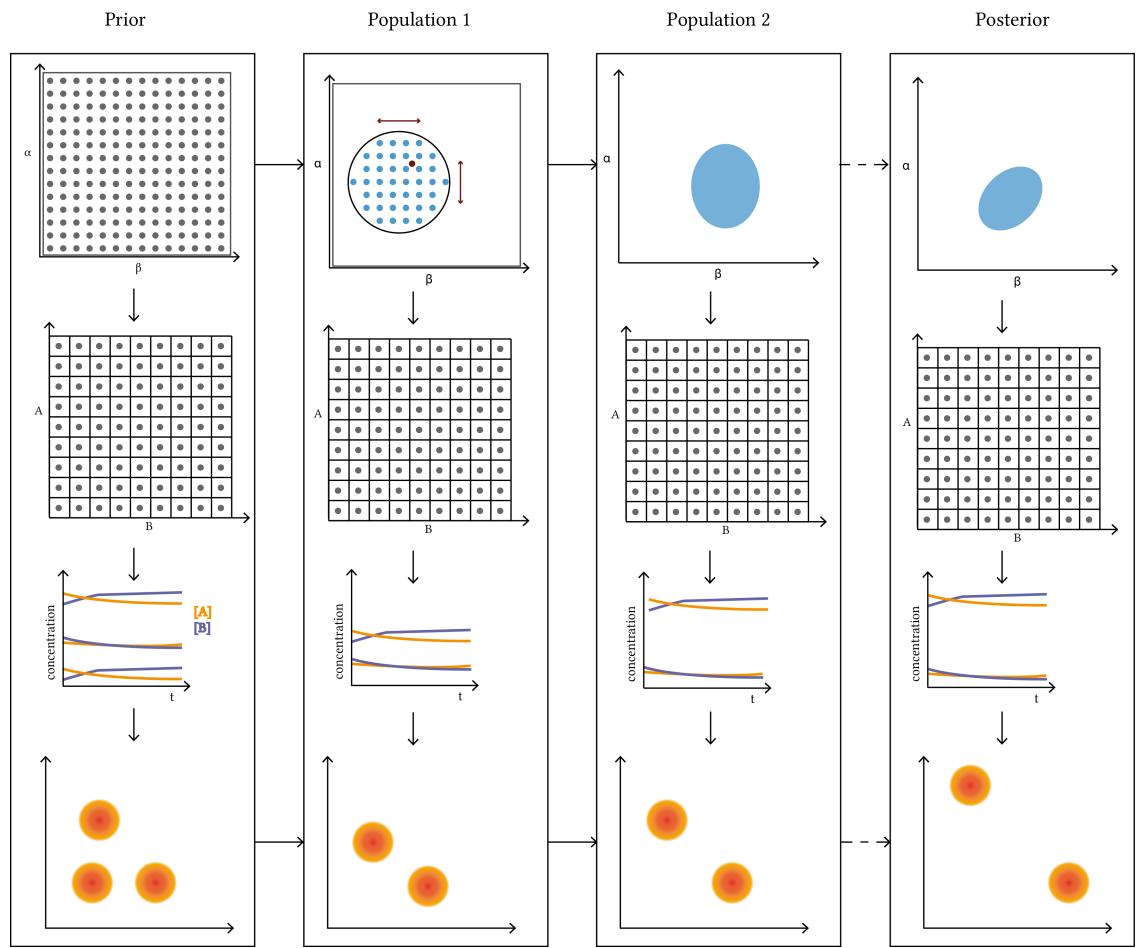


Figure 1.3 q

### 1.3.2 Particle sampling

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

For subsequent populations particles are sampled from the previous population. The weight of each particle in the previous population dictates the probability of it being sampled. Using Bernouli trials.

### 1.3.3 Perturbation

Each sampled particle is perturbed by a kernel defined by the distribution of the previous population.

$$K_p(\theta|\theta^*) = \theta^* + U(+s_p, -s_p)$$

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

If the  $\theta^*$  falls out of the limits of the priors then the perturbation is rejected and repeated until an acceptable  $\theta^*$  is obtained. The drawback of this perturbation kernel is that it can be computationally costly.

### 1.3.4 Initial condition sampling

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

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

14 TOGGLE SWITCH STABILITY

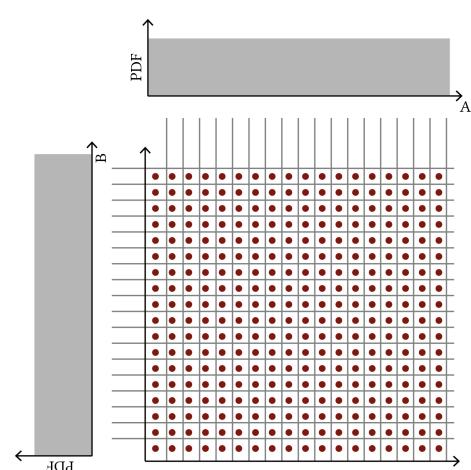


Figure 1.4 Latin Hypercube Sampling

### 1.3.5 Particle simulation

Each particle is simulated using cudasim (XXX). Cudasim is

### 1.3.6 Distance function

The distance function is the function the algorithm uses to compare the desired behaviour to the behaviour observed in each particle (XXX). In StabilityFinder the distance function consists of three distances. The first one is the difference between the number of desired clusters and the number of clusters observed in the phase plot. For this distance metric the number of clusters in the phase plot must be calculated. The clustering methods used are outlined in section 1.3.6.1.

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

#### 1.3.6.1 Clustering methods

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

Steady states of stochastic models are clustered using the K-means clustering (XXX) and the number of clusters determined using the Gap statistic (XXX). This method is more suited to stochastic solutions, where the Delta clustering method would fail as the steady state solutions tend to be more widely dispersed than in the deterministic case.

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

### 1.3.7 Particle rejection

A particle can be rejected for any number of the following reasons:

1. Particle cluster distance  $>\varepsilon_c$
2. Particle  $V_{bc}$  distance  $>\varepsilon_{Vbc}$
3. Particle  $V_{wc}$  distance  $>\varepsilon_{Vwc}$
4. Particle  $SS_v >\varepsilon_{ssv}$
5. Particle  $SS_l >\varepsilon_{ssl}$

I included a check for whether the model simulation had reached steady state. If the standard deviation of the last ten time points in the simulation is larger than a user-specified value (default is set to 0.0001), then the particle is rejected. This is to ensure that only particles that have reached steady state are considered.

### 1.3.8 Weight calculation

The weights of the particles are calculated by considering the previous population. The numerator  $\prod_{p=1}^R (\max(perturbation) - \min(perturbation))^{-1}$ . The denominator is.

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

### 1.3.9 Model checking

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

## 1.4 Calculating robustness

HOW DO WE DEFINE ROBUSTNES ETCETERA ETCETERA ETCETERA

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

First we start by the simplest case, where the posterior distribution is represented by a cuboid. This will be prone to overestimating robustness especially in cases of correlation between parameters. The next step is to approximate the posterior by a hyper-ellipsoid, an ellipse in higher dimensions. I compare both methods to ABC-SysBio model selection (XXX). There is good agreement between the three methods as can be seen in Figure 1.5 and 1.6.

If there is correlation between the parameters the volume was approximated by an ellipsoid encompassing 99.7% of the volume.

$$V = \frac{2\pi^{\frac{k}{2}}}{k\Gamma(\frac{k}{2})} [\chi_k^2(\alpha)]^{\frac{k}{2}} |\Sigma|^{\frac{1}{2}}, \quad (1.1)$$

where  $k$  is the number of dimensions,  $\Gamma$  is the Gamma function,  $\alpha$  is the confidence interval required and  $|\Sigma|$  is the determinant of the covariance matrix. In this case  $p = 3$ ,  $\alpha = 0.025$

---

**Algorithm 4** Approximating robustness

---

```

1: for each model  $m$  of  $M$  do
2:   Prior  $\sim U(a, b)$ 
3:    $V_{prior}^m = \prod_{i=1}^k (i_b - i_a)$ 

4:   Get  $1^{st} < data < 99^{th}$  percentiles
5:   if Cuboid calculation then
6:      $V_{post}^m = \prod_{i=1}^k (i_{max} - i_{min})$ 
7:   end if
8:   if Ellipsoid calculation then
9:     Calculate data covariance matrix
10:     $V_{post}^m = \frac{2\pi^{\frac{k}{2}}}{p\Gamma(\frac{k}{2})} [\chi_k^2(\alpha)]^{\frac{k}{2}} |\Sigma|^{\frac{1}{2}}$ 
11:   end if

12:    $R^m = \frac{V_{post}^m}{V_{prior}^m}$ 
13:    $R_{norm}^m = \frac{R^m}{\sum_{i=1}^M R_i^m}$ 
14: end for

```

---

We use the following two examples from ABC-SysBio (XXX):

#### 1.4.1 Case study 1: Infectious diseases

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

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

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

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

The second model, Model 2,

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

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

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

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

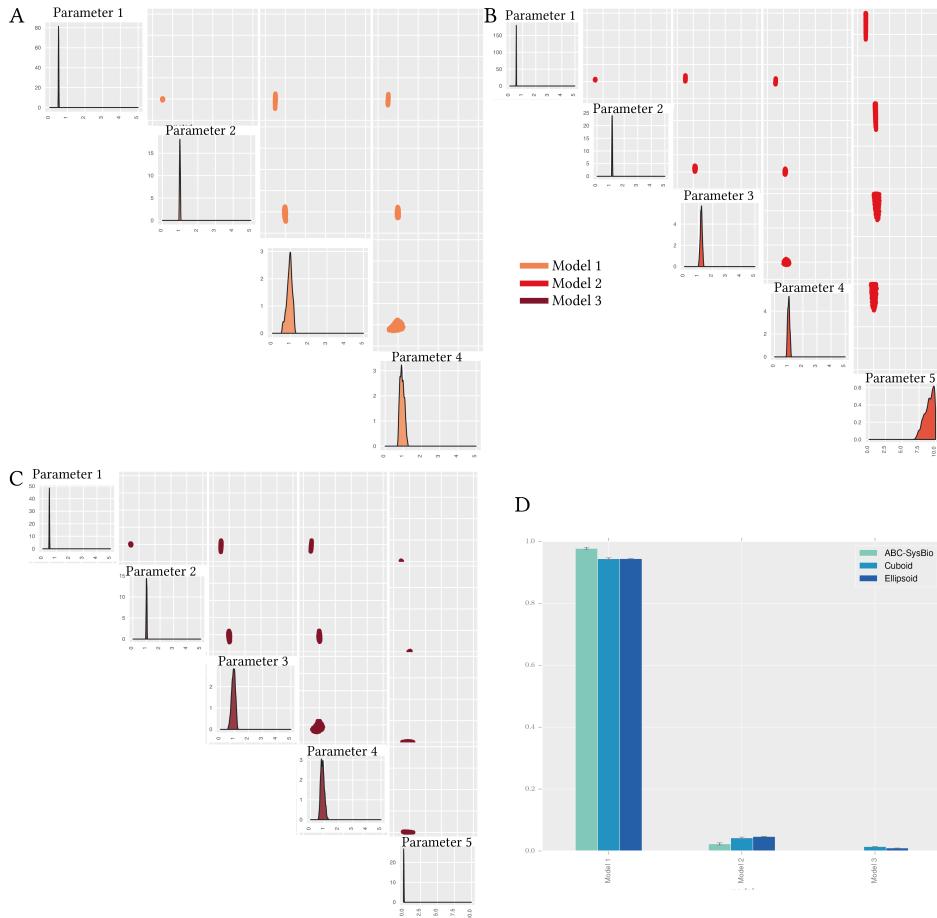
Finally the third model, Model 3,

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

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

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

The three models are simulated using ODEs. In ABC-SysBio (XXX) model selection is used. Parameter inference is also used for each model separately without the use of model selection. I used algorithm ?? to calculate the robustness of the posterior distributions of all three models. This robustness measure was then compared to the result of ABC-SysBio model selection. As shown in Figure 1.5, there is good agreement between the three measures of robustness. The posterior distributions of all three models are also shown in Figure 1.5.



**Figure 1.5** : Robustness analysis of the three models for the spread of infectious diseases. (A-C) The posterior distributions of the three models compared. (D) I use three methods to calculate robustness, ABC-SysBio model selection, the volume of the hyper-cuboid approximation of the posterior distribution and the volume of the hyper-ellipsoid approximation of the posterior distribution. There is good agreement between all three methods. All three methods show that Model 1, the simplest model, is the most robust model.

### 1.4.2 Case study 2: Population growth

The second example I will use to demonstrate the effectiveness of the method used here for robustness calculation is a population growth model.

The data was obtained by simulating an immigration-death model (XXX). Then this model and a model of logistic growth are compared for robustness of their posterior distributions. Model 1:

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

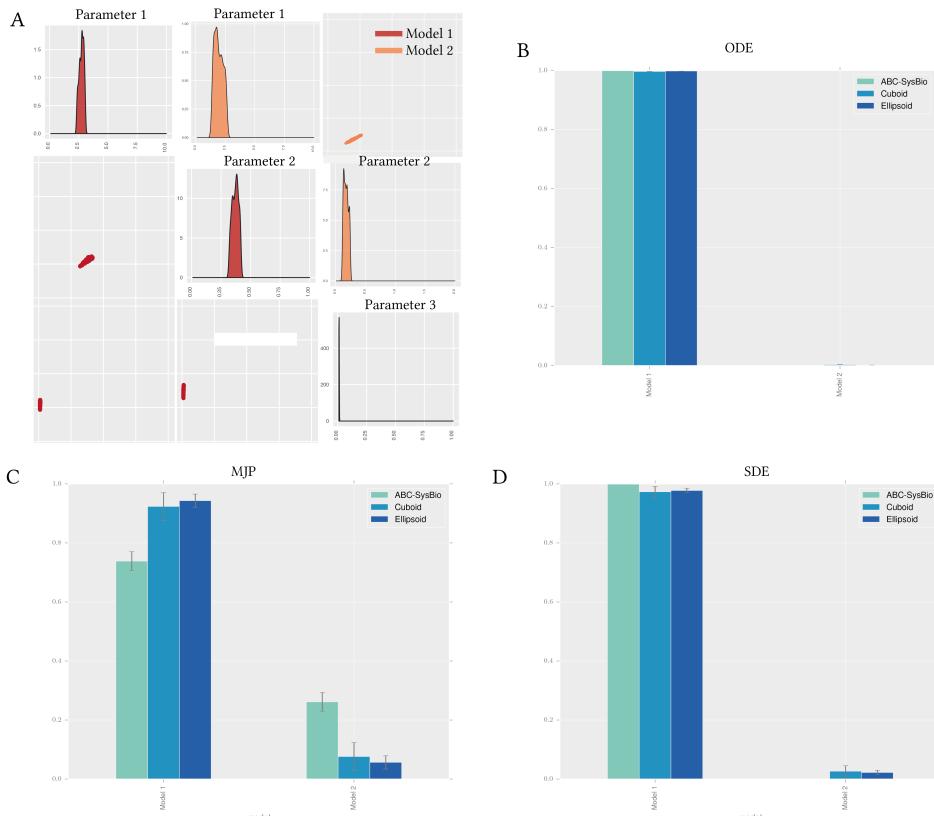
Model 2:

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

**check these equations**

As done in Section 1.4.1, two analyses were done on these two models. First, ABC-SysBio model selection was used to find the most robust model. Then parameter inference was done on each model. The resulting posterior distributions (shown in Figure 1.6), were compared for robustness using the cuboid and the ellipsoid approximation methods. All three robustness measures find that Model 1 is the most robust model. The analysis was repeated for ODE, MJP and SDE simulations, all arriving to the same result of Model 1 being the most robust. The results are shown in Figure 1.6.

## 22 TOGGLE SWITCH STABILITY



**Figure 1.6** : Robustness comparison of two population growth models. (A) The posterior distributions of the two models. (B-D) The models were simulated using ODE, MJP and SDE. Both the cuboid and the ellipsoid approximations agree with ABC-SysBio model selection results.

The two case studies used above show that the hyper-cuboid and the hyper-ellipsoid approximation of model robustness agree with the results obtained from ABC-SysBio model selection.

The drawbacks of our method compared to model selection are that each model stability analysis must reach completion in order to be used. In Bayesian model selection, where model selection is incorporated in the process, each model is also considered a particle (XXX) with an associated weight. If a model is performing poorly it does not proceed in the algorithm and is dropped when the weight falls low enough so that the model is not sampled (XXX). This can save time in the analysis as computational resources are not wasted on models that perform the required behaviour poorly. Using StabilityFinder for model selection, we will inevitably fall in this problem, since each model must reach the given final  $\epsilon$  in order to be comparable. This means that time will be spent in running models that are potentially a bad fit, or that have posteriors so small compared to the prior that it will take a long time for StabilityFinder to find it. This is a compromise we make in order to be able to run the models separately, as model selection is not the primary purpose of this software.

Even though this method of calculating robustness requires all the models' parameter inference to come to completion, the results still agree with ABC-SysBio model selection where that is not required (see Figures 1.5 and 1.6).

## 1.5 Applications of StabilityFinder

In this section we apply StabilityFinder to switch models in order to find the design principles underlying their stabilities. First we apply it to a simple model with known results, the Gardner, Cantor, & Collins (2000) toggle switch. This allows us to test StabilityFinder before applying it to further models.

### 1.5.1 Testing StabilityFinder

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

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

## 24 TOGGLE SWITCH STABILITY

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

where  $u$  is the concentration of repressor 1,  $v$  the concentration of repressor 2,  $a_1$  and  $a_2$  denote the effective rates of synthesis of repressors 1 and 2 respectively,  $\beta$  is the cooperativity of repression of promoter 1 and  $\gamma$  of repressor 2. Gardner *et al* studied the deterministic case and concluded that there are two conditions for bistability for this model; that  $a_1$  and  $a_2$  are balanced and that  $\beta, \gamma > 1$  (Gardner, Cantor, & Collins 2000). In order to test StabilityFinder we used it to find the posterior distribution for which this model exhibits bistable behaviour. We therefore set the desired behaviour to two steady states, and using a wide range of values as priors as shown in Table 1.1, we used StabilityFinder to find the parameter values necessary for bistability to occur. The posterior distribution calculated by StabilityFinder for the Gardner deterministic case is shown in Figure 1.7B.

**Table 1.1** Gardner switch priors in the deterministic and stochastic cases

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

These results agree with the results reported by Gardner, Cantor, & Collins (2000) (Gardner, Cantor, & Collins 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 ??B. The conditions set by the original paper for parameters  $a_1$  and  $a_2$  are met.

We next applied StabilityFinder to the case of the Gardner switch under stochastic dynamics using the same priors as the deterministic case, and again searched the parameter space for bistable behaviour. The posterior is shown in Figure 1.7C. We can see that the conditions on the parameters required for bistability in the deterministic case generally still stand in the stochastic case. There appears to be slightly looser requirements on the parameters of the stochastic model (wider marginal distributions), which is expected due to the nature of clustering deterministic steady states versus stochastic steady states. The gap statistic is used in the case of the stochastic case, as it is capable of dealing with noisier data whereas a simpler and faster algorithm is used for clustering the deterministic solutions. These results demonstrate that StabilityFinder can be used to find the parameter values that can produce a desired stability and allow us to confidently apply the methodology to more complex models.

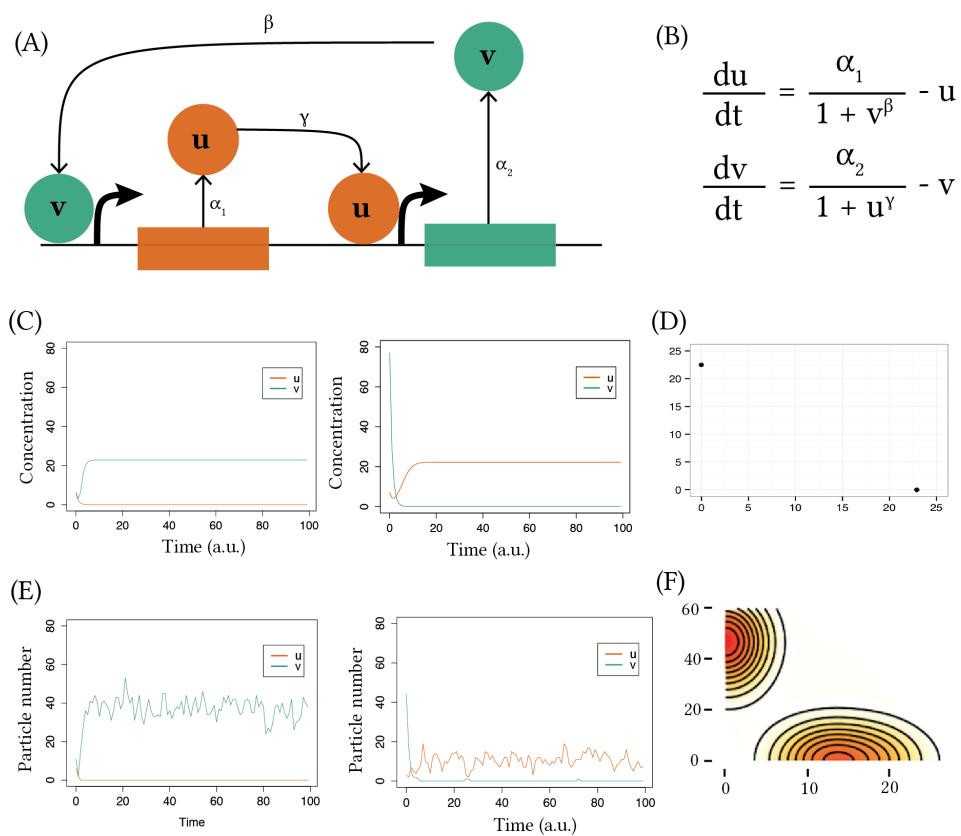


Figure 1.7

## 26 TOGGLE SWITCH STABILITY

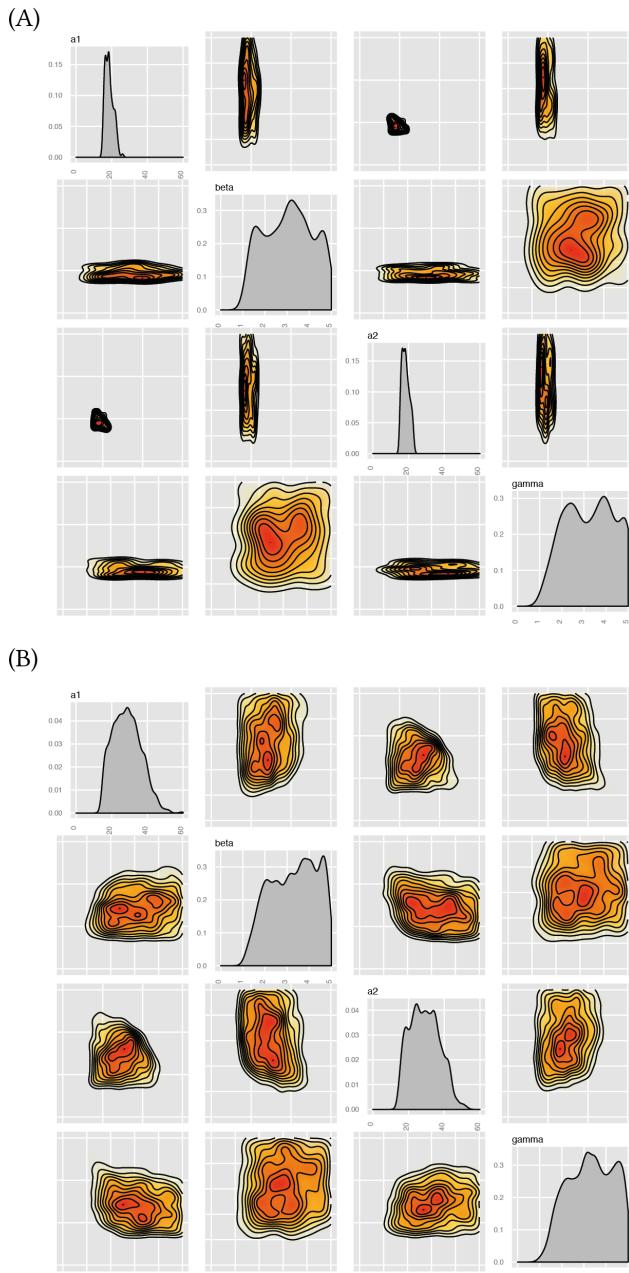


Figure 1.8

### 1.5.2 Lu toggle switch models

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

$$\begin{aligned}\dot{x} &= g_x H_{xy}^S(y) - k_x x \\ \dot{y} &= g_y H_{yx}^S(x) - k_y y,\end{aligned}$$

where

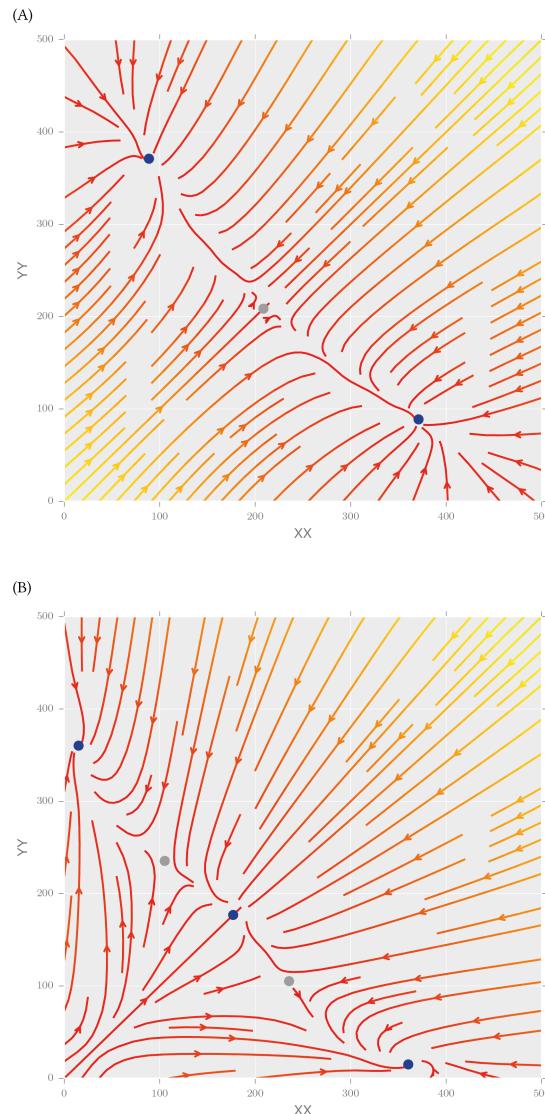
$$\begin{aligned}H_I^S(x) &= H_I^-(x) + \lambda_I H_I^+(x) \\ H_I^-(x) &= 1 / [1 + (x/x_I)^{n_I}] \\ H_I^+(x) &= 1 - H_I^-(x),\end{aligned}$$

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

For the parameter values used in the Lu study, the classical switch exhibits three steady states (Figure 1.13), two of which are stable and one is unstable. Using StabilityFinder with priors centred around the parameter values used in the original paper (see Supplementary Information), we can identify the most important parameters for achieving bistability. The posterior distribution of these models are shown in Figure 1.13A. We find that the parameters representing the rates of degradation of the transcription factors in the system ( $k_x, k_y$ ) must both be large in relation to the prior ranges for bistability to occur. Protein degradation rates have been shown to be important for many system behaviours including oscillations (XXX). StabilityFinder

It is known that the addition of positive autoregulation to the classical toggle switch can induce tristability (XXX; Lu, Onuchic, & Ben-Jacob 2014). Here we investigate the interplay of positive autoregulation on the values of the other parameters in the model. We extended the analysis presented in Lu, Onuchic, & Ben-Jacob (2014) by including the switch with single positive autoregulation (model SP-LU),

## 28 TOGGLE SWITCH STABILITY



**Figure 1.9** Stream plot of the vector plot of the (A) CS-LU and DP-LU switches.

The colours indicate the magnitude of the vectors, with yellow indicating high and red low values. The blue points represent stable steady states and the grey points represent unstable steady states.

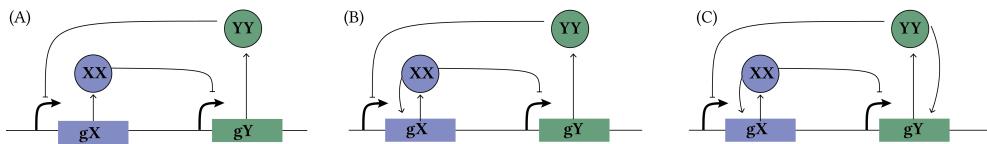
where an asymmetry of positive feedbacks is present between the two genes. The advantage of using StabilityFinder over solving the system analytically is that the full parameter space is explored rather than solving the system for a single set of parameters. This allows us to deduce model properties that could not otherwise be identified. Robustness to parameter fluctuations can be explored, as well as parameter correlations and restrictions on the values they can take while still producing the desired behaviour.

The DP-LU model is given by

$$\begin{aligned}\dot{x} &= f_x(x, y) = g_x H_{xy}^S(y) H_{xx}^S(x) - k_x x \\ \dot{y} &= f_y(x, y) = g_y H_{yx}^S(x) H_{yy}^S(y) - k_y y,\end{aligned}$$

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

$$\begin{aligned}\dot{x} &= g_x H_{xy}^S(y) H_{xx}^S(x) - k_x x \\ \dot{y} &= g_y H_{yx}^S(x) - k_y y.\end{aligned}$$



**Figure 1.10** The three LU toggle switch models. (A) CS-LU, (B) SP-LU and (C) DP-LU.

We find that the switch with single positive autoregulation is capable of bistable behaviour as seen in Figure 1.13B, but this is only possible when the strength of the promoter under positive autoregulation,  $g_x$ , is small. There appear to be no such constraints on the strength of the original, unmodified, promoter,  $g_y$ .

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

deterministic dynamics (Guantes & Poyatos 2008). The classical switch is also capable of both bistable and tristable behaviour when stochastic dynamics capture

**Table 1.2** Priors of the classical(CS-LU), single positive (SP-LU) and double positive (DP-LU) models.

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

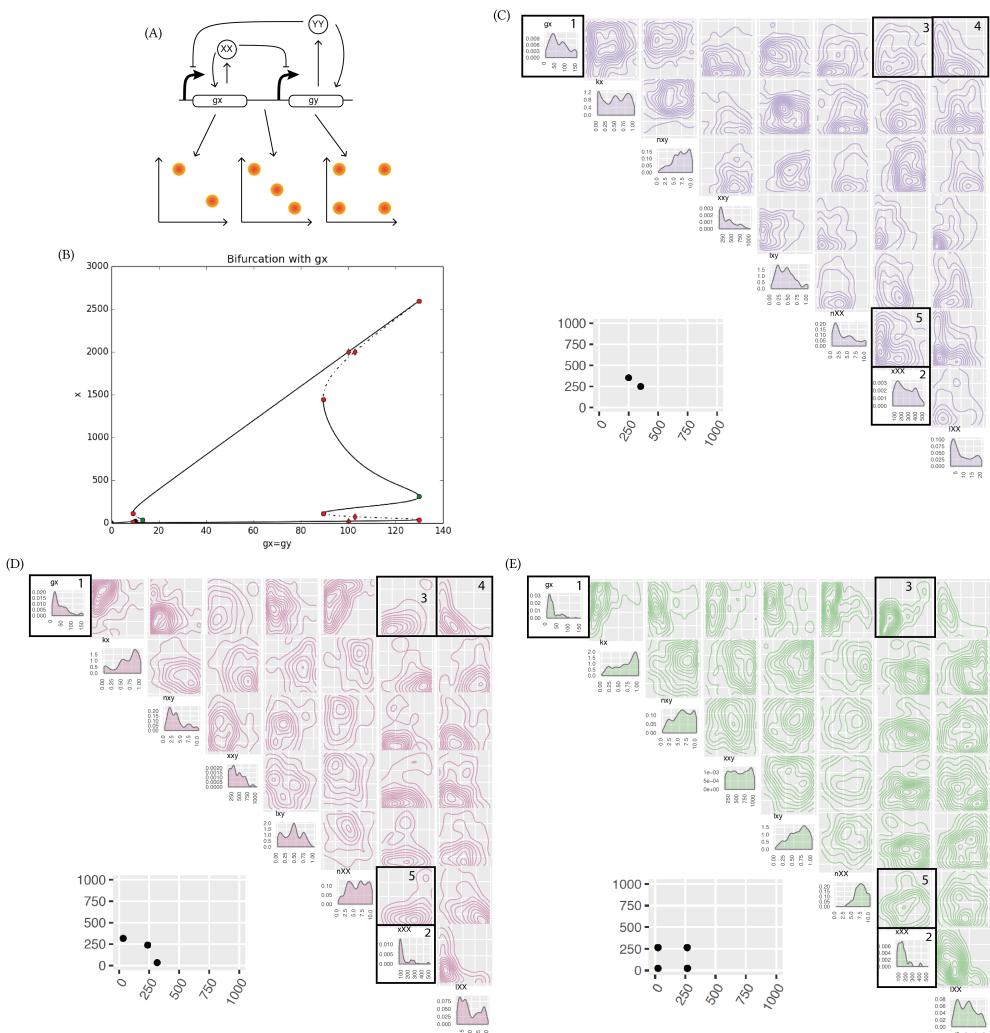
small number effects (Ma et al. 2012). It is therefore of great interest to understand the conditions under which these two behaviours occur in both stochastic and deterministic scenarios.

In order to do this using StabilityFinder, we first obtained posterior distributions for bistable and tristable behaviours in the deterministic case (DP-LU model) and then compared the individual parameter distributions (Figure ?? and Supplementary Information). From analysis of the one dimensional marginal distributions it appears that there is no difference in the parameter values that allow a switch to be bistable versus tristable (Figure ??B). However, upon examination of the two dimensional marginal distributions we find that the correlation structure of a small subset of the posterior parameter values is in fact different under the two behaviours (Figure ??C). Most notably, we find differences in the bivariate distribution of the two parameters for gene expression,  $gx$  and  $gy$ , as highlighted in Figure ??C, box 1. In the tristable case the distribution is more constrained than in the bistable case, as both parameters must be small for tristability to arise.

**Table 1.3** Priors used in the three-node switch

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

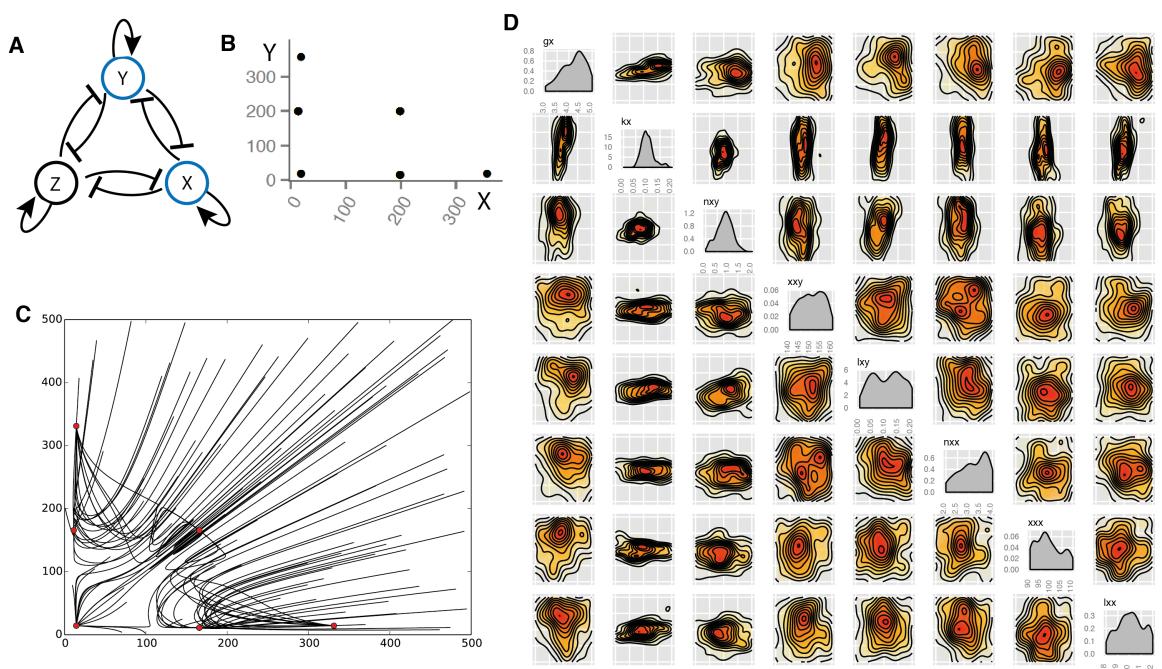
## 32 TOGGLE SWITCH STABILITY



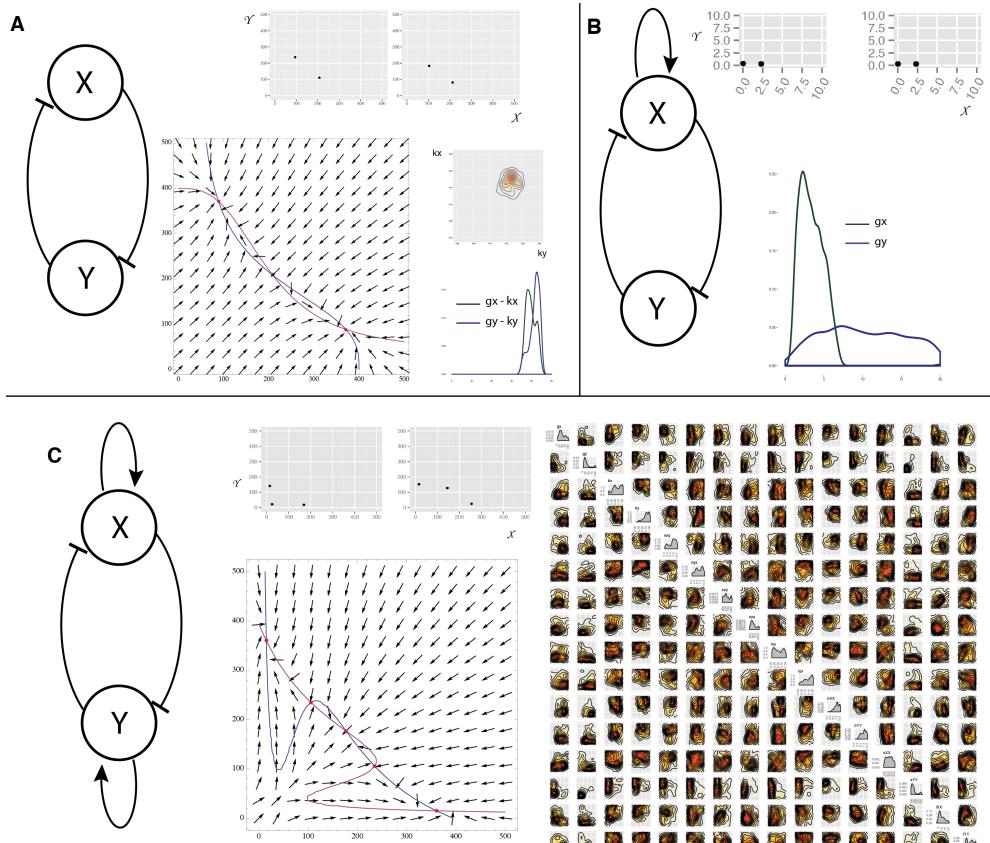
**Figure 1.11 :**

Parameters  $xYY$  and  $gy$  are tightly constrained in the tristable case and both required to be small, but less so in the bistable case (Figure ??C, box 3). Another notable difference is between parameters  $xXX$  and  $IXX$  shown in Figure ??C, box 4, where they are constrained in the bistable case but not the tristable case. Interestingly, we also find parameter correlations conserved between the bistable and tristable case, as seen in Figure ??C, box 2, where parameters  $IXX$  and  $gx$ , positive autoregulation and gene expression are negatively correlated in both cases. This highlights the importance of treating unknown parameters as distributions rather than fixed values when studying the parameter values of a model, as they are capable of uncovering not only the ranges and values needed but also the correlations between parameters that would not have otherwise been detected.

To further demonstrate the flexibility of our framework we investigated a system capable of higher stabilities. Multistability is found in differentiating pathways, like the myeloid differentiation pathway (Ghaffarizadeh:2014bt; Cinquin:2005go). We allow for these more complex dynamics by extending the Lu DP model by adding another gene, making it a three gene switch. This new system is depicted in Figure 1.12A. In StabilityFinder we look for six steady states, the output being in nodes  $X$  and  $Y$  and using the priors shown in Table 1.3. We successfully find that the system is capable of six steady states, as shown in Figure 1.12C and as predicted mathematically and shown in Figure 1.12B. Interestingly, and consistent with the results presented above, we find that the most constrained parameters for this behaviour are again the degradation rate of the proteins,  $kx$ . Additionally we find that the Hill coefficients for the repressors,  $nxy$ , are tightly constrained to be smaller than 1.5 as seen in Figure 1.12D. This example demonstrates that Stability Finder can be used to elucidate the dynamics of more complex network architectures, which will be key to the successful design and construction of novel gene networks as synthetic biology advances.



**Figure 1.12 :** The three-node mutual repression model, with added positive auto-regulation on each node. (A) The model. The model is studied in two dimensions using StabilityFinder, X and Y. (B) The phase plot of the resulting final population of StabilityFinder. There are 6 steady states. (C) The posterior distribution of the 6-steady state three-node system. Parameters  $k_x$  and  $n_{xy}$  are the most constrained.

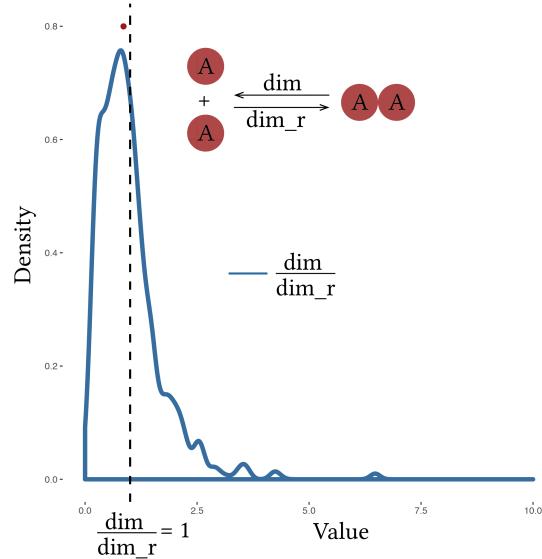


**Figure 1.13** : The three variants of the Lu models. (A)CS (deterministic). The classic switch with no autoregulation is bistable. The most restricted parameters for this behaviour are  $k_x$  and  $k_y$  which both have to be high relative to the prior while the net protein production for  $X$  and  $Y$  must be balanced. (B)SP (deterministic). The extended Lu model with a single positive autoregulation on  $X$ . This model is bistable when  $g_x$  is small. (C) DP (deterministic). The Lu model with double positive autoregulation is tristable, and its posterior distribution shown here. We find two types of tristable behaviour, one where the third steady state is zero-zero and one where the third state is high (non-dead).

### 1.5.3 Mass Action switches

Although the DP switch can achieve tristability, we investigated how the addition of positive autoregulation alters the robustness of the switch for bistable behaviour. Here we define a robust system as a device that can withstand fluctuations in parameter values and still produce the desired behaviour (parametric robustness). Feedback loops are well known key regulatory motifs (Brandman et al. 2005). Although negative feedback loops are essential for homeostasis and buffering (Thomas, Thieffry, & Kaufman 1995) and can increase robustness to extrinsic noise sources, we focussed on the addition of positive feedback because this has been shown to increase robustness in other systems (XXX). We use StabilityFinder to compare the two models, and determine whether its worthwhile building a bistable circuit with added positive feedback.

In order to study the switch system in the most realistic way, we avoid using the quasi-steady state approximation (QSSA) that is often used in modelling the toggle switch. Using mass action, this changes the two-equation system used in Lu, Onuchic, & Ben-Jacob (2014) (Lu, Onuchic, & Ben-Jacob 2014) into a system of 18 equations and 7 parameters in the classical switch case. These are shown in the Supplementary Information.



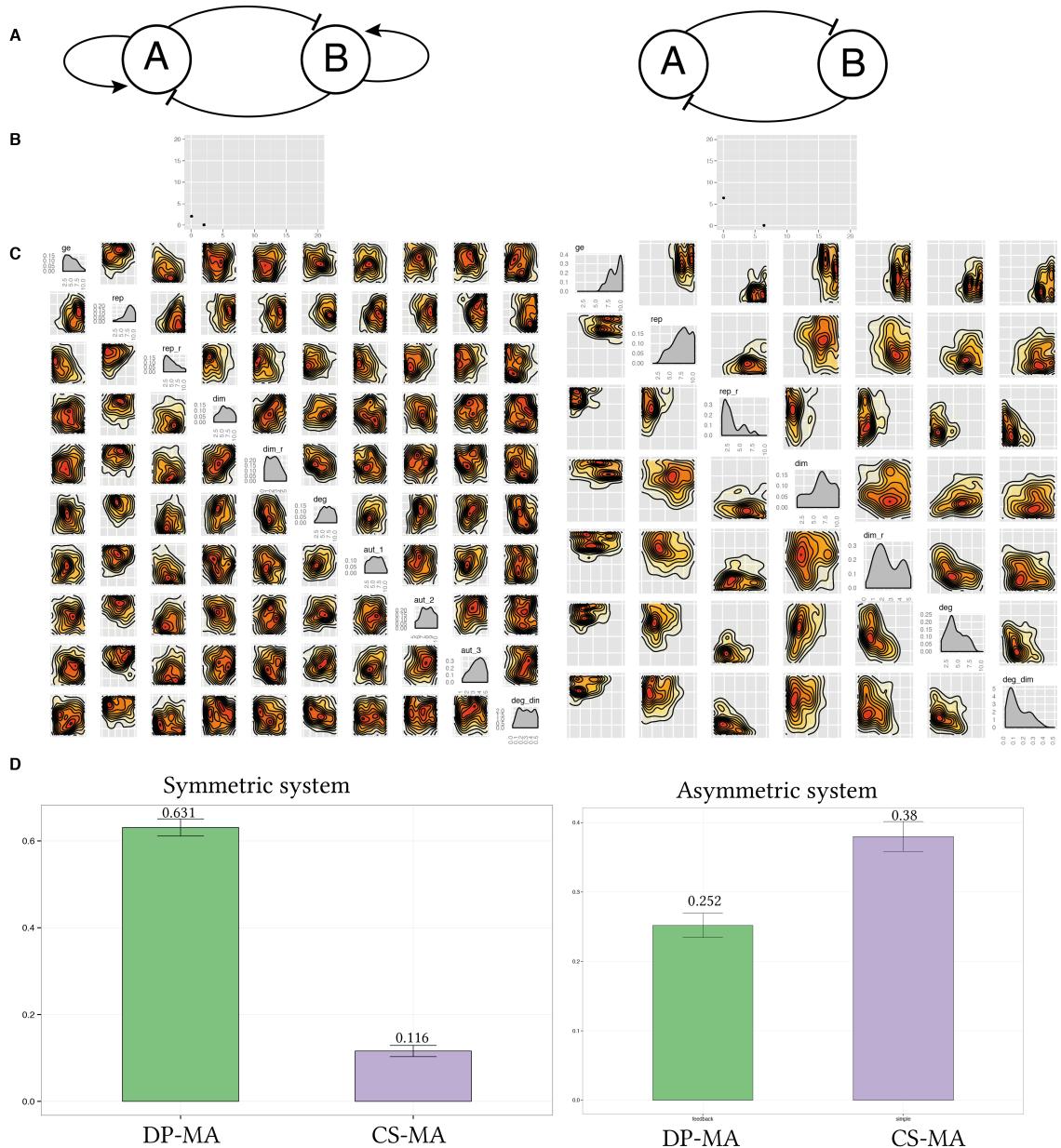
**Figure 1.14** : (A) Separating transcription and translation in the mass action model. (B) Transcriptional bursting does not allow for a bistable switch. (C) The Quassi Steady state approximation assumption that the dimerization reaction ( $dim$ ) is much faster than its reverse ( $dim_r$ ) cannot be justified here. The median of the data (red) lies below the  $x=1$  line (black dashed line) which indicates that in the majority of the particles in the posterior  $\frac{dim}{dim_r} < 1$  (D) No difference was found in the robustness to parameter fluctuations for the two models tested here.

**Table 1.4** Design principles of bistable and tristable switches

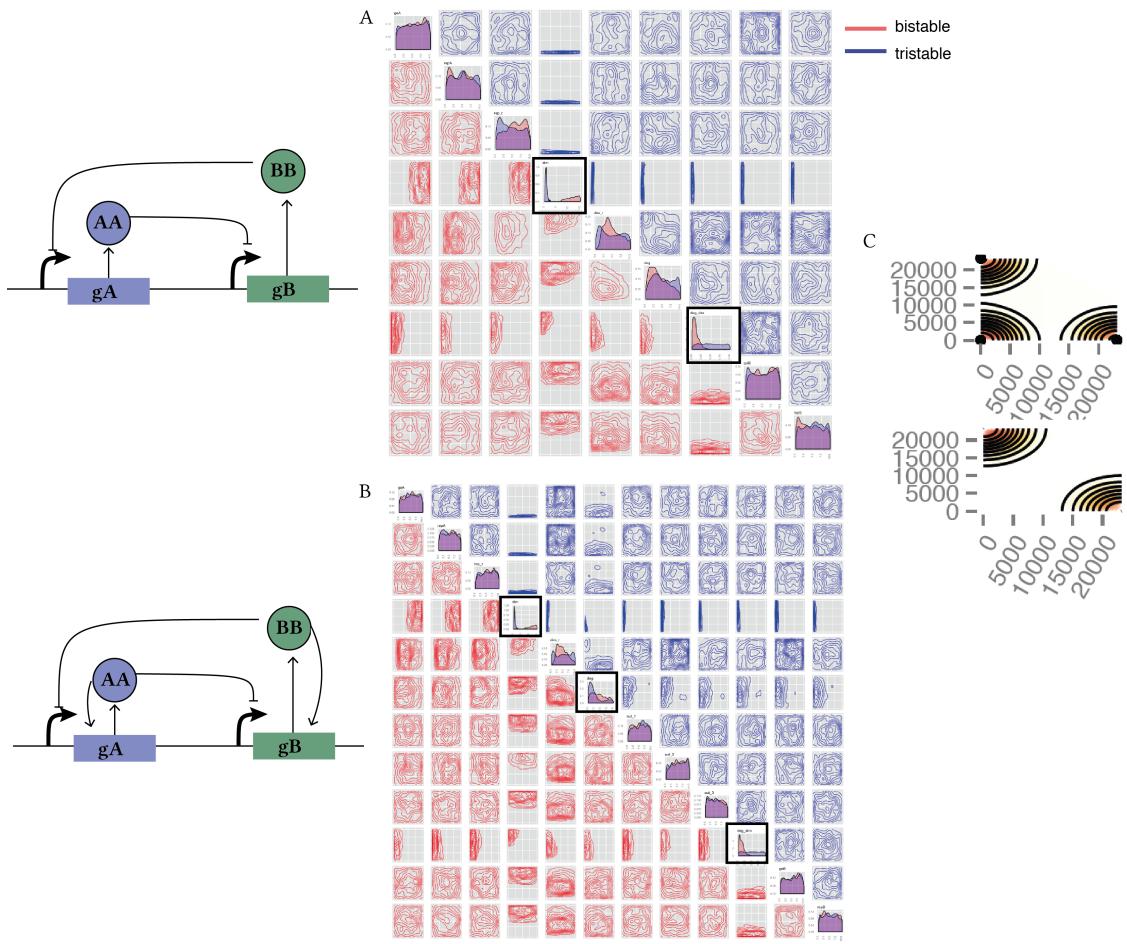
	CS-MA		DP-MA	
	<i>Bistable</i>	<i>Tristable</i>	<i>Bistable</i>	<i>Tristable</i>
dimerisation	High	Low	High	Low
protein degradation	-	-	-	Low
dimer degradation	Low	-	Low	-

Given the posterior distributions shown in Figure 1.15, we find that the parameter for transcription factor degradation (*deg*) has to be high, a result that reflects the findings for the Lu models above [Do we really see that??](#). We also find that the addition of positive feedback loops greatly increases the system's robustness to parameter fluctuations as seen in Figure 1.15D. Adding positive feedback loops to the model allows it to be bistable over a greater range of parameter values, but only when the system is symmetric. When this constraint is removed we no longer see a difference in robustness between the CS-MA and DP-MA models. [What does this mean?](#)

We find that in the stochastic case, both the simple switch, S-MA, and positive autoregulation switch, DP-MA, are capable of both bistable and tristable behaviour. The fact that tristability can occur in the classical model is consistent with the effect of small molecule numbers; if gene expression remains low, it provides the opportunity for small number effects to be observed, and the third steady state to stabilise (Ma et al. 2012). In order to ensure that the tristable switches found in the stochastic case are truly tristable, we re-sample the posterior distributions and simulate to steady state. If the resulting phase plots are tristable then we know that the posterior truly represents tristability. As can be seen in Figure 1.16B, differences in the parameter values are observed between the bistable and tristable switches, in both CS-MA and DP-MA models. The design principles for both the CS-MA model and the DP-MA model are summarised in Table 1.4



**Figure 1.15** : Adding positive feedback increases robustness to parameter fluctuations. We compare the posterior distributions of the simple mass action switch and the switch with added double positive autoregulation (A). The two models were both capable of bistable behaviour (B). The two posterior distributions are shown in (C). Comparing the functional region  $F$  of the posterior distributions (D) we find that there is a five-fold increase in robustness when positive autoregulation is added. This is not the case for the asymmetric switch in which there is no significant difference in robustness between the CS-MA and DP-MA models.



**Figure 1.16 :** Tristability is possible in the mass action toggle switch models only when simulated stochastically. (A) the simple toggle switch with no autoregulation can be both bistable and tristable. The two posteriors are shown, where the posterior distribution of the bistable switch is shown in red and of the tristable switch in blue. From the posterior distribution we can deduce the the dimerization parameter must be small for tristability to occur but large for bistability. The switch with double positive autoregulation and its posterior distributions for the bistable and tristable case are shown in (B). A sample phase plot of a stochastic tristable and bistable mass action switch is shown in (C). Comparing the robustness of the switch with and without positive autoregulation, we don't find a significant difference (D).

## 1.6 Discussion

We have developed an algorithm that can identify the parameter regions necessary for a model to achieve a given number of stable steady states. The novelty in our framework over existing methodology is that complex models can be analyzed assuming both deterministic and stochastic dynamics. We have shown that the algorithm can be used to infer parameter ranges and compare robustness between models. We found that adding positive feedback increases the robustness of a toggle switch model to parameter fluctuations. We also found that stochastic modelling can enable a model to achieve stabilities not possible in the deterministic case. We uncovered the design principles that make a tristable switch and finally we found that a three-node switch is capable of hexastability.

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

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

More generally our results highlight that changing the level of abstraction included in the model can significantly alter the observed multistationarity. This is further exacerbated by a change in the dynamics from deterministic to stochastic. These results for the case of multistationarity, assuming they translate to systemic behaviour more generally, suggest that a programme of experimental work, com-

bined with systems modelling, is required to understand the rules of thumb for abstraction in model based design of synthetic biological systems.

## 1.7 Summary

In this chapter...

# 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.
- 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.
- Gardner, T. S., Cantor, C. R., & Collins, J. J. (2000). ‘Construction of a genetic toggle switch in *Escherichia coli*’. *Nature* **403**(6767), 339–342.
- Guantes, R. & Poyatos, J. F. (2008). ‘Multistable decision switches for flexible control of epigenetic differentiation.’ *PLoS Computational Biology* **4**(11), e1000235.
- Isaacs, F. J., Hasty, J., Cantor, C. R., & Collins, J. J. (2003). ‘Prediction and measurement of an autoregulatory genetic module’. *Proceedings of the National Academy of Sciences of the United States of America* **100**(13), 7714–7719.
- 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.
- Loinger, A., Lipshtat, A., Balaban, N. Q., & Biham, O. (2007). ‘Stochastic simulations of genetic switch systems’. *Physical Review E*.
- Lu, M., Onuchic, J., & Ben-Jacob, E. (2014). ‘Construction of an Effective Landscape for Multistate Genetic Switches’. *Physical review letters* **113**(7), 078102.
- 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.
- Marjoram, P., Molitor, J., Plagnol, V., & Tavaré, S. (2003). ‘Markov chain Monte Carlo without likelihoods.’ *Proceedings of the National Academy of Sciences of the United States of America* **100**(26), 15324–15328.

#### 44 BIBLIOGRAPHY

- 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.
- 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.