Contents

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 Stability Finder, 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 Stability Finder, justify the choices made and the drawbacks of this method. In the sections following we apply Stability Finder 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

A circuit must be robust to a fluctuating cellular environment and its response and sensitivity must be able to be fine tuned in order to orchestrate a network of circuits that function together. A robust circuit can tolerate the compound stochasticity that a chain of circuits brings, and fine tuning of its response and sensitivity enables the researcher to make it sensitive to an upstream signal as well as influence a downstream subsystem. Parts can be fine tuned by developing component libraries (**Lu: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.

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

I use this methodology to investigate genetic toggle switches and uncover the design principles behind making a bistable switch, as well as those necessary to make a switch with 3 and 4 steady states. I also demonstrate the ability of Stability Finder to examine more complex systems and examine the design principles of a three gene switch. The examples I used demonstrate that Stability Finder will be a valuable tool in the future design and construction of novel gene networks.

1.2.1 ABC algorithms

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

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

Algorithm 1 ABC rejection algorithm

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

The main disadvantage of this method is that if the prior distribution is very different from the posterior, the acceptance rate is very low (Toni et al. 2009). An alternative method is the ABC Markov Chain Monte Carlo (MCMC) developed by Marjoram et al. (2003) (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 issues faced by the rejection and MCMC methods. It propagates the prior through a series of intermediate distributions in order to arrive at an approximation of the posterior. The tolerance, ε for the distance of the simulated data to the desired data is made smaller at each iteration. When ε is sufficiently small, the result will approximate the posterior distribution (Toni et al. 2009).

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

Algorithm 2 ABC SMC algorithm

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

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

$$\frac{d[B]}{dt} = p1[A] \tag{1.1}$$

$$\frac{d[A]}{dt} = -p1[A] \tag{1.2}$$

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.

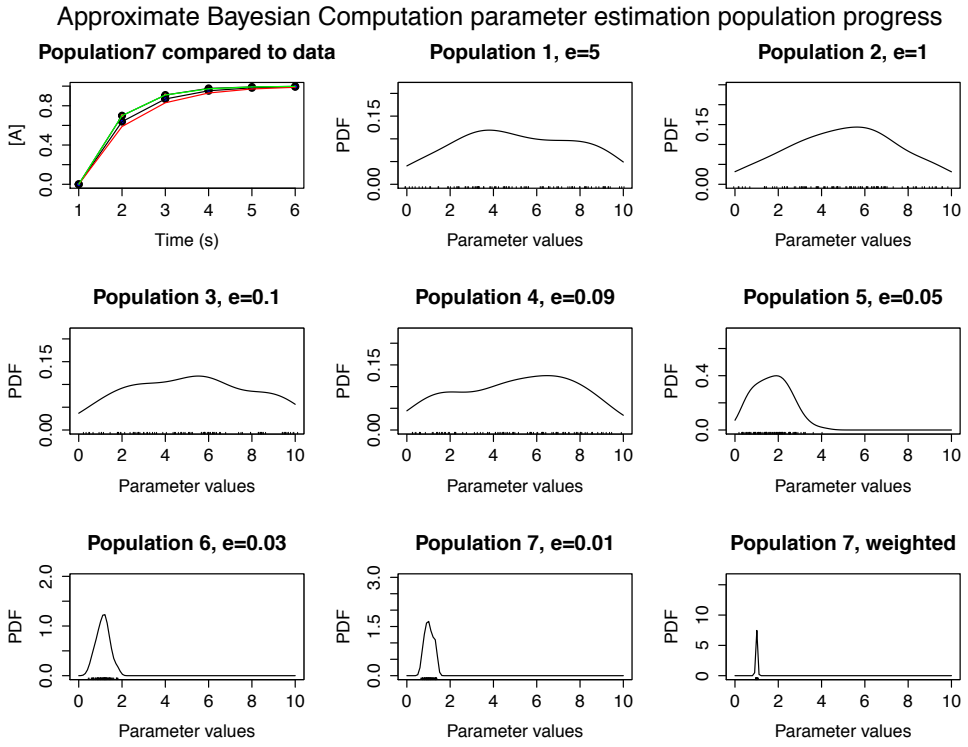

Figure 1.1 ABC SMC parameter inference. The posterior parameter is equal to 1 and its time course shown in red in the top left panel. The blue time course is that of the final population, green is the upper quartile and red is the lower quartile range of values. The progress of the selection process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

Figure 1.1 demonstrates, using a simple example, that ABC SMC is capable of fitting a model to the data. During the course of 7 populations, the accepted distance ε of the simulated particles to the data is incrementally decreased. This leads to a final population where the distance of the data to the particles is very small, and there is a good agreement between the two. The algorithm concludes with a set of parameter values that produced this behaviour, which approximate the posterior distribution. The posterior distribution found in this model is the same as the parameter value used to generate the data. This example successfully demonstrates the effectiveness of the ABC SMC algorithm in fitting models to data.

1.2.2 Robustness

During this thesis I use robustness in its parametric sense. I examine the robustness of a model to fluctuations in parameter values.

— TO DO —

1.3 Stability Finder algorithm

To investigate the multistable behaviour of systems, I had to make a number of extensions to existing approaches. Firstly, a wide range of initial condition samples are required. WHY? 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. An overview of the algorithm is given in section 1.3.1. Then each module in the algorithm is described in sections 1.3.2-1.3.9.

1.3.1 Algorithm overview

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

Algorithm 3 StabilityFinder algorithm

Initialise ϵ
population p $\leftarrow 1$
if p $= 1$ **then**
    Sample particles (θ) from priors
else
    Sample particles from previous population
    Perturb each particle by $\pm$ half the range of the previous population (j) to obtain new perturbed population (i).
end if
Sample initial conditions via Latin Hypercube Sampling.
Simulate each particle to obtain steady state values.
Cluster steady state
Reject particles if d $> \epsilon$.
Calculate weight for each accepted θ
Normalise weights
repeatsteps 3 - 15
until $\epsilon \leq \epsilon_T$

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 ε, for the distance from the desired behaviour necessary for the algorithm to terminate. The flow of execution is illustrated in Figure 1.2E. Since the algorithm is computationally intensive, all deterministic and stochastic simulations are parallelised and performed using algorithms implemented on Graphical Processing Unit (GPU)s (XXX).

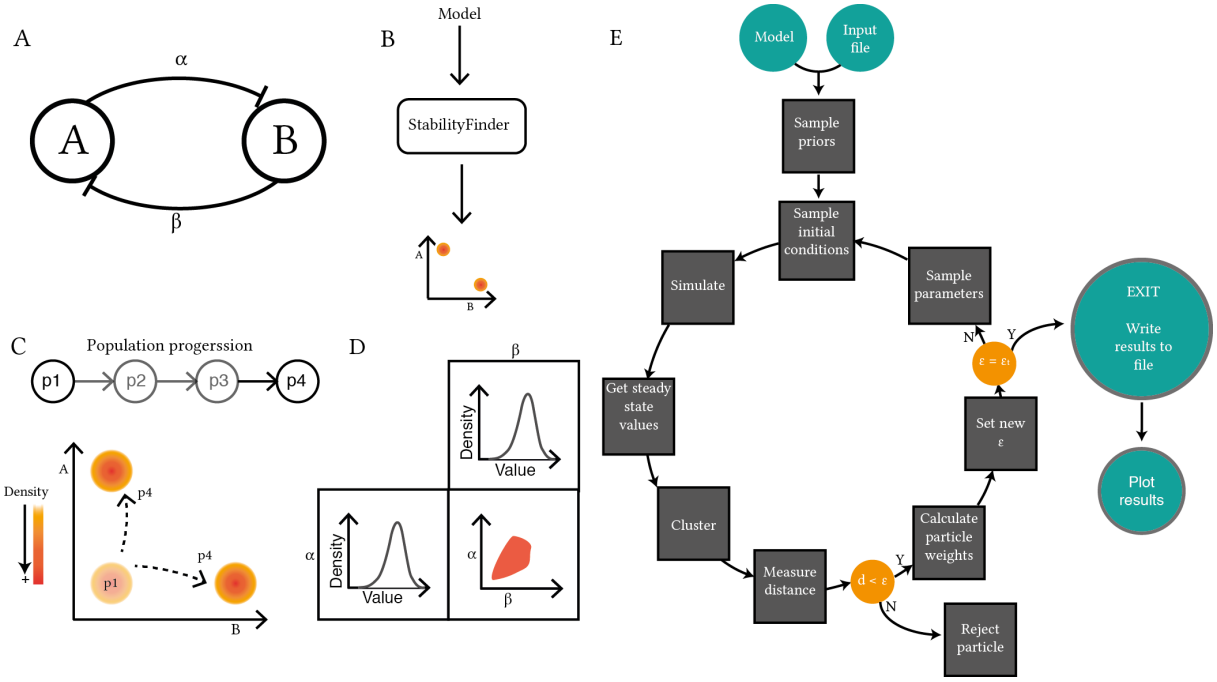

Figure 1.2 : Using sequential Monte Carlo to examine system stability. The algorithm takes as input a model (A) and evolves it to the stability of choice (C) via intermediate populations. In this example model shown in A, There are two species and two parameters. For the model to be bistable, the phase 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).

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. The number of samples to be drawn is specified by the user in the input file.

1.3.3 Perturbation

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

$$K_p(\theta|\theta*) = \theta * + U(+s_p, -s_p), \text{ where:} \tag{1.3}$$

$$s_p = \frac{1}{2}(max(\theta_{p-1}) - min(\theta_{p-1})) \tag{1.4}$$

If the θ* falls out of the limits of the priors then the perturbation is rejected and repeated until an acceptable θ* is obtained. This method is successful in perturbing the particles by a small amount in order to explore the parameter space, but can be slow to complete.

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 uniformly. Latin hypercube sampling is done in two dimensions in Stability Finder. The uniform priors of the two species in consideration represent a rectangle space, which is subdivided into equal parts. Then a random sample is drawn from each sub-part. This ensures the whole space is evenly sampled.

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

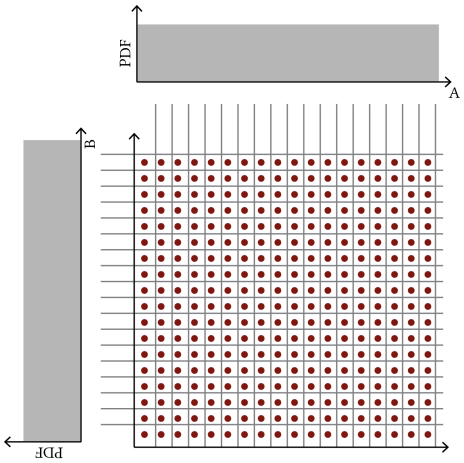

Figure 1.3 Latin hypercube sampling ensures that the whole space is sampled evenly. For the two species concerned, A and B, we assume uniform distributions, shown in grey. The joint space of the two distributions is divided into smaller equal parts and a random sample is drawn from within each subspace.

1.3.5 Particle simulation

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

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 Stability Finder the distance function consists of three distances. The first one is the difference between the number of desired clusters and the number of clusters observed in the phase plot. For this distance metric the number of clusters in the phase plot must be calculated. The clustering methods used are outlined in section 1.3.6.1.

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

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 δaway 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 detailed algorithms used are shown in Appendix (XXX).

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

1.3.7 Particle rejection

Once the distance from the desired behaviour has been calculated, the algorithm rejects any particles whose distance is farther than the current ε. The distances taken into account are the number of clusters (C), the between-cluster variance (V_{bc}) and the within-cluster variance (V_{wc}) as outlined in section 1.3.6, Apart from these I have included an additional two checks for the particles. Firstly, Stability Finder checks if the simulation of a particle has reached steady state. If the standard deviation of the last ten time points in the simulation is larger than a user-specified value (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. Secondly, there is a check for the minimum level of the steady states. This is to allow the user to select for steady states whose protein levels are above a certain threshold. This has to be added as an additional check as the steady state levels must be experimentally observable if they are to be used to design new systems. Two steady state levels of very low levels would be biologically indistinguishable and thus meaningless in an experimental setup. This check is optional to the user, and can be set to zero if not desirable. In summary, a particle can be rejected for any number of the following reasons:

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

1.3.8 Weight calculation

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

$$w_t^{(i)} = \frac{P(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})} \text{ for n} > 0 \tag{1.5}$$

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 as 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

Unlike other ABC SMC methods, Stability Finder 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 I discuss another algorithm I developed in order to extract robustness information from the results of Stability Finder and apply model selection.

In order to discuss the parametric robustness of a model we need to be able to approximate the volume of the viable parameter space. The viable parameter space is the space that approximates the posterior distribution that can give rise to the desired behaviour. I tested two methods of approximating the volume of the viable space, which are outlined in Algorithm 4. The first, crude, method is to calculate the volume of the cuboid that contains this viable space. The 1^{st} and 99^{th}percentile of the viable space (denoted as data in Algorithm 4) are taken into account. This is necessary in order to exclude outliers in the distribution that would skew the

volume calculation significantly. Each parameter represents a side in the cuboid and since the volume of a cuboid is equal to the product of its sides (XXX), the volume of the viable space is equal to the product of the ranges of all the parameters. This cuboid method will be prone to overestimating robustness especially in cases of correlation between parameters.

For the second method the volume of the viable space was represented by a hyper-ellipsoid, an ellipse in higher dimensions. This method should not be as prone to overestimation of robustness as the cuboid method as an ellipsoid can take correlation into account. For this method the distribution of the viable space is assumed to be normal. The method calculates the covariance matrix of the distribution, whose volume is given by Equation 1.6. Just as in the cuboid method, the 1^{st} and 99^{th} percentile of the data is ignored.

To validate these methods I compare them to ABC-SysBio model selection (XXX) Why compare it to abc sybio? why is this a good measure? . There is good agreement between the three methods as can be seen in Figures 1.4 and 1.5.

$$V = \frac{2\pi^{\frac{k}{2}}}{k\Gamma(\frac{k}{2})}\left[\chi^2_k(\alpha)\right]^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}, \tag{1.6}$$

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

Algorithm 4 Approximating robustness

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

We use the following two examples 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).

$$\dot{S} = \alpha - \gamma SI - dS \tag{1.7}$$

$$\dot{I} = \gamma SI - vI - dI \tag{1.8}$$

$$\dot{R} = vI - dR, \tag{1.9}$$

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

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

$$\dot{S} = \alpha - \gamma SI - dS \tag{1.10}$$

$$\dot{L} = \gamma SI - \delta L - dL \tag{1.11}$$

$$\dot{I} = \delta L - vI - dI \tag{1.12}$$

$$\dot{R} = vI - dR, \tag{1.13}$$

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

$$\dot{S} = \alpha - \gamma SI - dS + eR \tag{1.14}$$

$$\dot{I} = \gamma SI - vI - dI \tag{1.15}$$

$$\dot{R} = vI - dR - eR, \tag{1.16}$$

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 the two methods outlined in algorithm 4 to calculate the robustness of the posterior distributions of all three models. This robustness measure was then compared to the result of ABC-SysBio model selection. As shown in Figure 1.4, there is good agreement between the three measures of robustness. The posterior distributions of all three models are also shown in Figure 1.4.

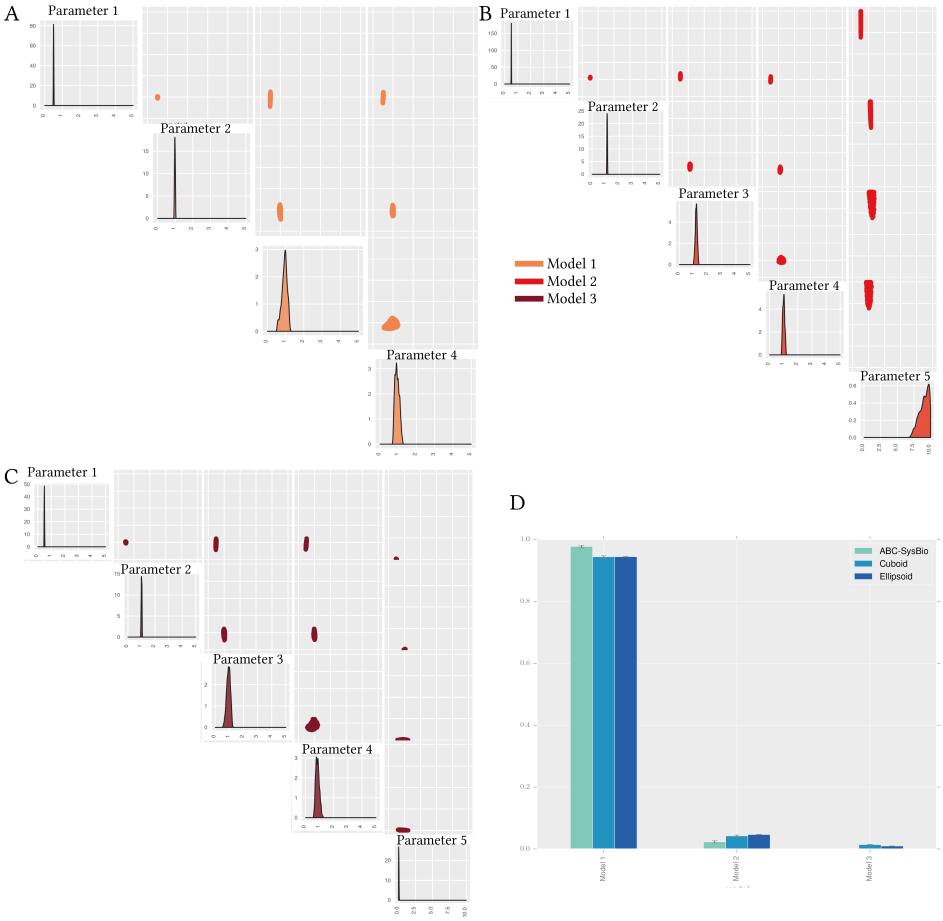

Figure 1.4 : Robustness analysis of the three models for the spread of infectious diseases. (A-C) The posterior distributions of the three models compared. (D) I use three methods to calculate robustness, ABC-SysBio model selection, the volume of the hyper-cuboid approximation of the posterior distribution and the volume of the hyper-ellipsoid approximation of the posterior distribution. Each analysis was repeated three times. The height of the bars indicate the mean robustness from the three repeats and the error bars represent the standard deviation. There is good agreement between all three methods. All three methods show that Model 1, the simplest model, is the most robust model.

1.4.2 Case study 2: Population growth

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

The data was obtained by simulating an immigration-death model shown in Equation 1.17. This model (referred to as Model 1) and a model of logistic growth are compared for robustness of their posterior distributions. Model 1:

$$\frac{dI}{dt} = \alpha - \beta I \tag{1.17}$$

Logistic growth, model 2:

$$\frac{dI}{dt} = \gamma - I(\delta - \epsilon I) \tag{1.18}$$

check these equations

As in Section 1.4.1, two analyses were carried out on these two models. First, ABC-SysBio model selection was used to find the most robust model. Then parameter inference was done on each model. The resulting posterior distributions (shown in Figure 1.5), 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 1.5.

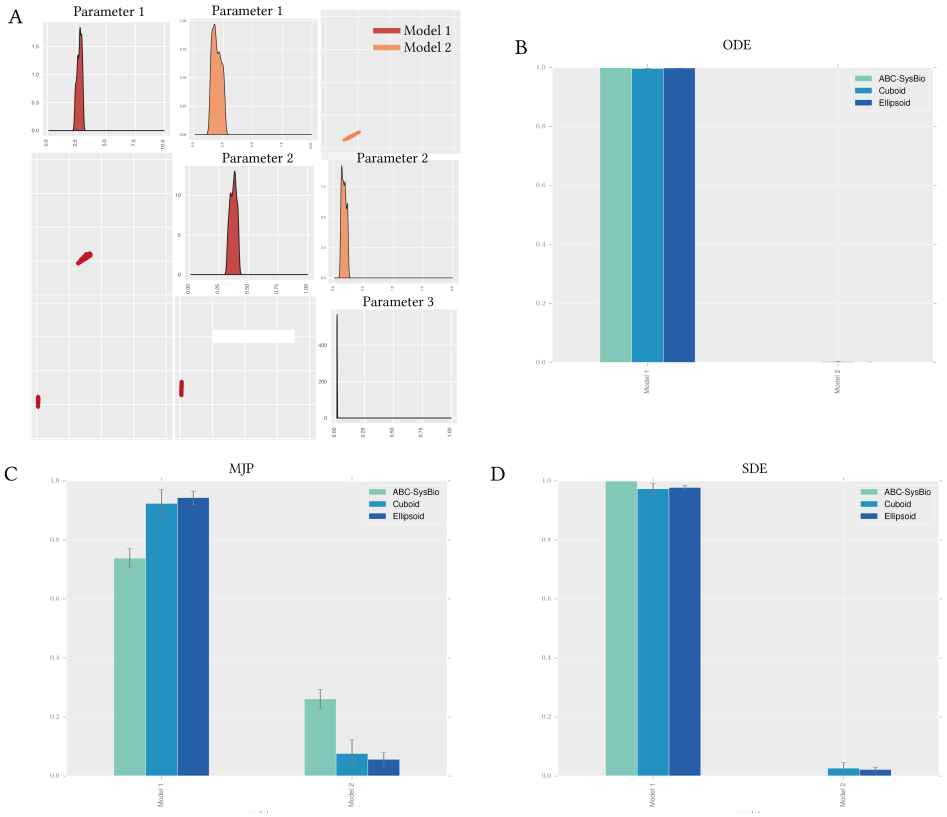

Figure 1.5 : Robustness comparison of two population growth models. (A) The posterior distributions of the two models. (B-D) The models were simulated using ODE, MJP and SDE. Both the cuboid and the ellipsoid approximations agree with ABC-SysBio model selection results. Each analysis was repeated three times. The height of the bars indicate the mean robustness from the three repeats and the error bars represent the standard deviation.

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

1.5 Applications of StabilityFinder

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

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 1.6A, and in the deterministic case is defined by the following ODEs:

$$\frac{du}{dt} = \frac{a_1}{1 + v^{\beta}} - u \tag{1.19}$$

$$\frac{dv}{dt} = \frac{a_2}{1+u^{\gamma}} - v, \tag{1.20}$$

where u is the concentration of repressor 1, v the concentration of repressor 2, a_1 and a_2 denote the effective rates of synthesis of repressors 1 and 2 respectively, β is the cooperativity of repression of promoter 1 and ϒof repressor 2. Gardner, Cantor, & Collins (2000) studied the deterministic case and concluded that there are two conditions for bistability for this model; that a_1 and a_2 are balanced and that β, ϒ>1 (Gardner, Cantor, & Collins 2000). I test Stability Finder by using it to find the posterior distribution for which this model exhibits bistable behaviour. Therefore, the desired behaviour is set to two steady states, and using a wide range of values as priors as shown in Table 1.1, I used Stability Finder to find the parameter values necessary for bistability to occur. The posterior distribution calculated by Stability Finder for the Gardner deterministic case is shown in Figure 1.7A.

Table 1.1 Gardner switch priors in the deterministic and stochastic cases

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

These results agree with the results reported by Gardner, Cantor, & Collins (2000) (Gardner, Cantor, & Collins 2000). For this switch to be bistable a_1 and a_2 must be balanced while β and ϒ must both be >1, as can be seen in the marginal distributions of β and ϒ in Figure 1.7A.

I 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 distribution is shown in Figure 1.7B. 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 Stability Finder can be used to find the parameter values that can produce a desired stability and can be confidently applied to more complex models.

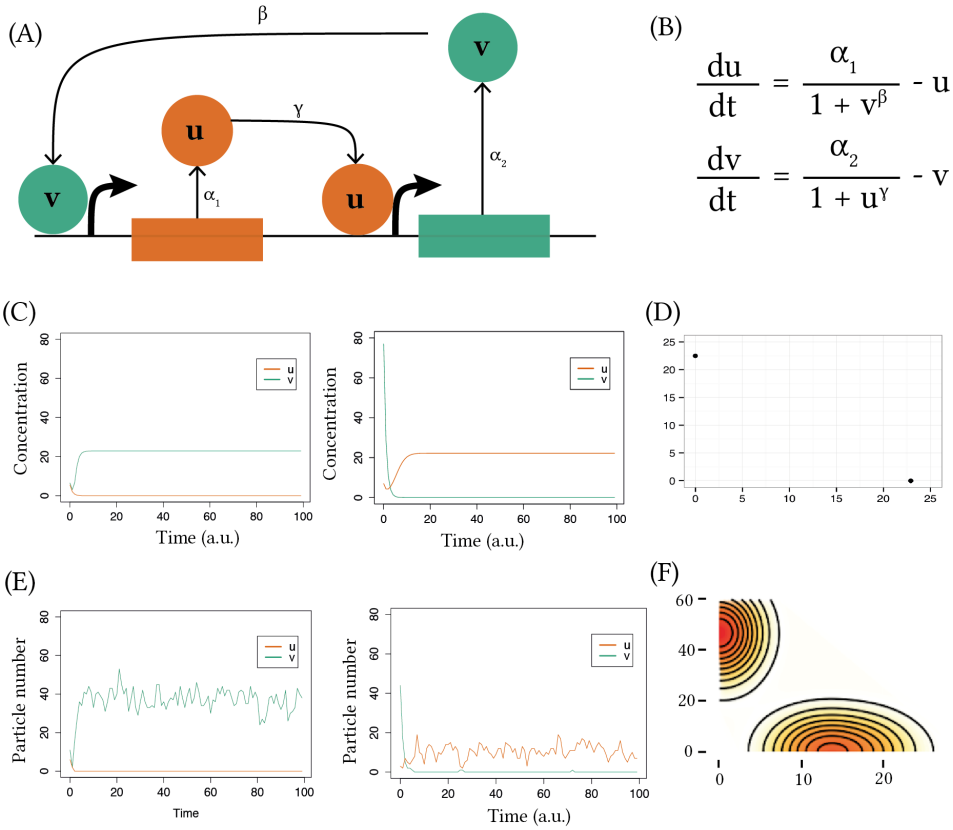

Figure 1.6 The Gardner switch model used to test Stability Finder. The Gardner model (A) consists of two mutually repressing transcription factors. It can be reduced to a two-equation system (B) , where u and v are the two transcription factors, a_1,a_2 are their effective rates of synthesis, u,v are their concentrations and β, γ represent the cooperativity of each promoter. (C) Two samples of deterministic simulated timecourses of the Gardner switch and (D) The resulting phase plot. (E) Two samples of timecourses of the stochastic simulations and (F) the resulting phase plot.

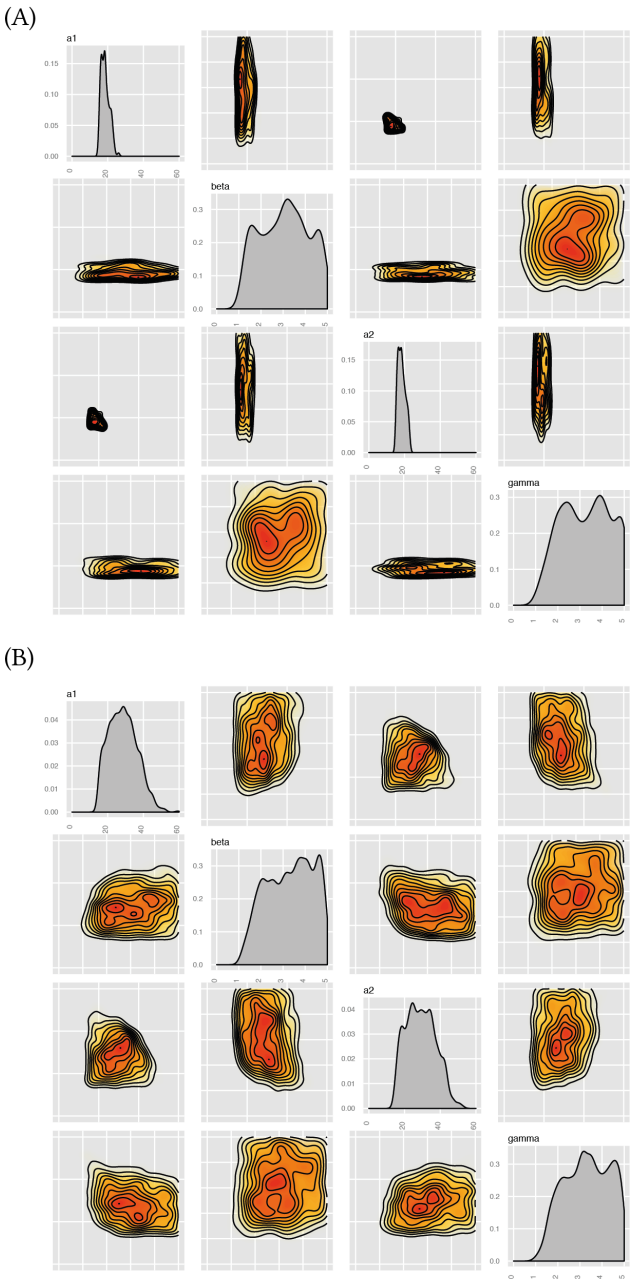

Figure 1.7 Elucidating the stability of the Gardner switch. The Gardner model has four parameters, for which I want to find the values for which this system is bistable. I use Stability Finder to find the posterior distribution of the bistable Gardner switch, deterministically (A) and stochastically (B). The posterior distributions are shown as the density plots of each parameter as well as each one plotted against the other.

1.5.2 Lu toggle switch models

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

$$\dot{x} = g_x H^S_{xy}(y) - k_x x \tag{1.21}$$

$$\dot{y} = g_y H^S_{yx}(x) - k_y y, \tag{1.22}$$

where:

$$H^S_I(x) = H^-_I(x) + \lambda_I H^+_I(x) \tag{1.23}$$

$$H^-_I(x) = 1/\left[1 + (x/x_I)^{n_I}\right] \tag{1.24}$$

$$H^+_I(x) = 1 - H^-_I(x), \tag{1.25}$$

and the DP-LU model is given by

$$\dot{x} = f_x(x,y) = g_x H^S_{xy}(y) H^S_{xx}(x) - k_x x \tag{1.26}$$

$$\dot{y} = f_y(x,y) = g_y H^S_{yx}(x) H^S_{yy}(y) - k_y y, \tag{1.27}$$

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

For the parameter values used in the Lu study, the classical switch exhibits three steady states, two of which are stable and one is unstable. Bifurcation diagrams of the two Lu models are shown in Figure 1.8.

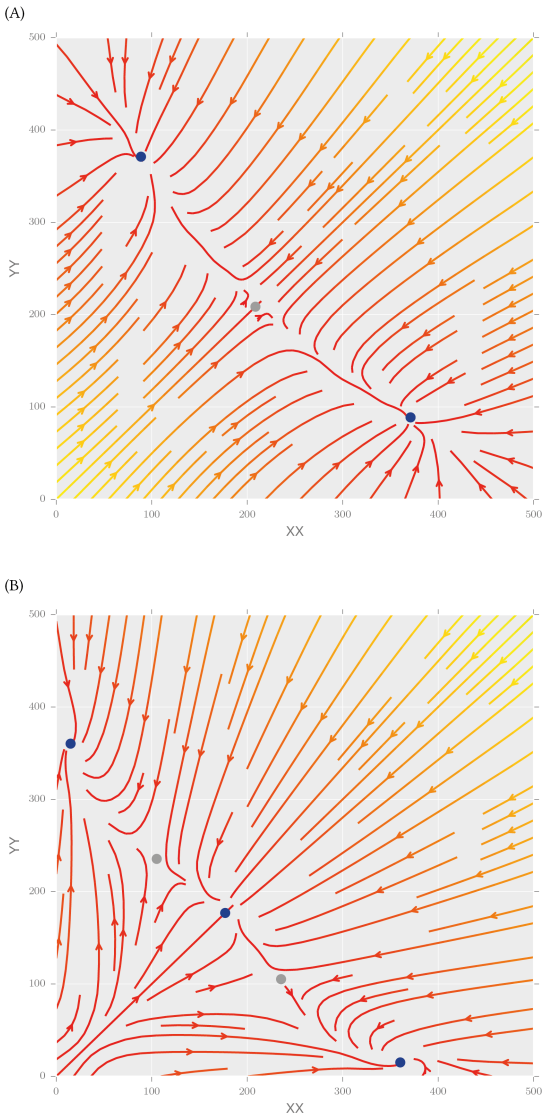

Figure 1.8 Stream plot of the vector plot of the (A) CS-LU and DP-LU switches. The colours indicate the magnitude of the vectors, with yellow indicating high and red low values. The blue points represent stable steady states and the grey points represent unstable steady states.

1.5.2.1 Extending the Lu models

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

It is known that that the addition of positive autoregulation to the classical toggle switch can induce tristability (**XXX**; Lu, Onuchic, & Ben-Jacob 2014). Here I investigate the interplay of positive autoregulation on the values of the other parameters in the model. I extended the analysis presented in Lu, Onuchic, & Ben-Jacob (2014) by including the switch with single positive autoregulation (model SP-LU), where an asymmetry of positive feedbacks is present between the two genes. The three switches considered in this analysis are shown in Figure 1.9.

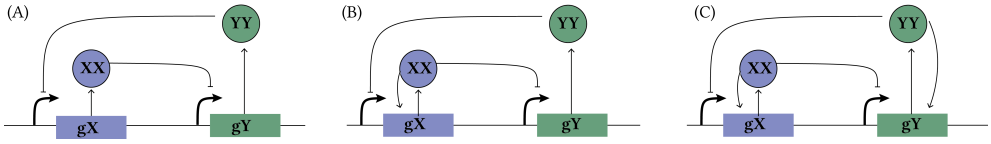

Figure 1.9 The three LU toggle switch models. (A) CS-LU, (B) SP-LU and (C) DP-LU.

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

$$\dot{x} = g_x H^S_{xy}(y)\, H^S_{xx}(x) - k_x x \tag{1.28}$$

$$\dot{y} = g_y H^S_{yx}(x) - k_y y. \tag{1.29}$$

Using Stability Finder with priors centred around the parameter values used in the original paper (see Table 1.2), we can identify the most important parameters for achieving bistability. The posterior distribution of these models are shown in Figure 1.10A. We find that the parameters representing the rates of degradation of the transcription factors in the system (k_x,k_y) must both be large in relation to the prior ranges for bistability to occur. Protein degradation rates have been shown to be important for many system behaviours including oscillations (**XXX**).

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

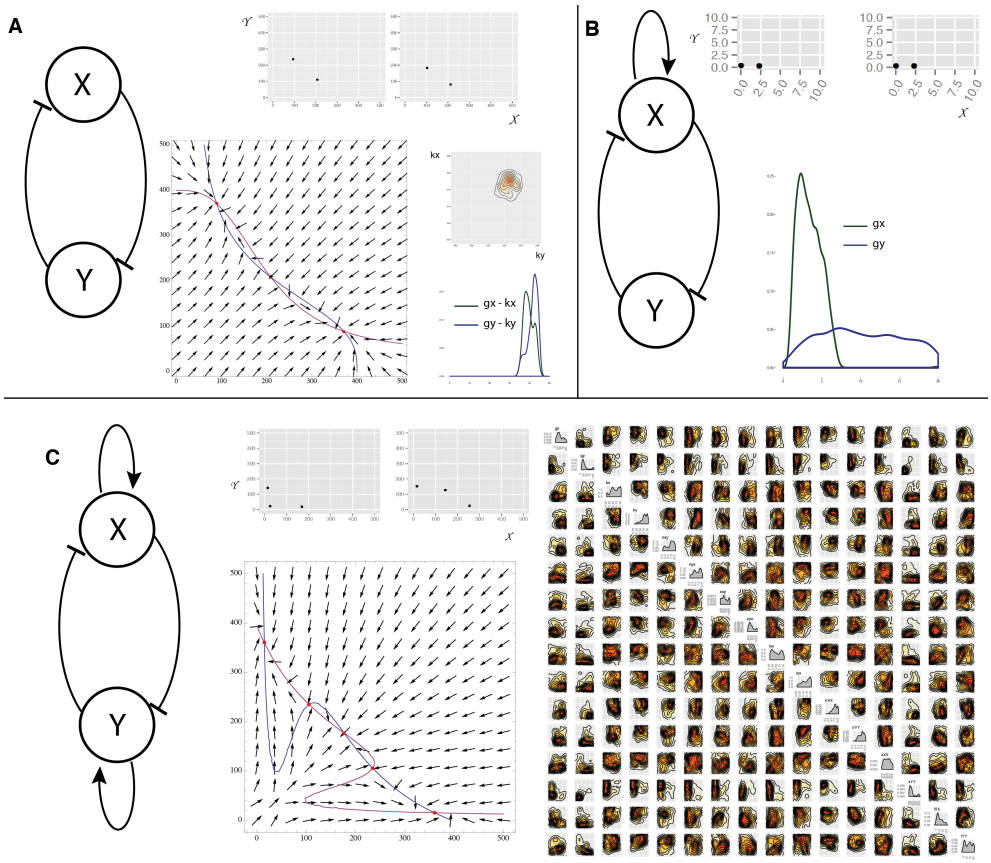

Figure 1.10 : The three variants of the Lu models. (A)**CS (deterministic)**. The classic switch with no autoregulation is bistable. The most restricted parameters for this behaviour are kx and ky which both have to be high relative to the prior while the net protein production for X and Y myst be balanced. (B)**SP (deterministic)**.The extended Lu model with a single positive autoregulation on X. This model is bistable when gx is small. (C) **DP (deterministic)**. The Lu model with double positive autoregulaiton 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).

We find that the switch with single positive autoregulation is capable of bistable behaviour as seen in Figure 1.10B, but this is only possible when the strength of the promoter under positive autoregulation, gx, is small. There appear to be no such constraints on the strength of the original, unmodified, promoter, gy.

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

1.5.2.2 Multistability in the Lu models

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

Since the Lu models can be solved analytically, we can obtain the bifurcation diagram of the DP-LU by keeping all parameters constant apart from gene expression (gx). The result shown in Figure 1.11B, the system can exhibit 2, 3 or 4 steady states depending on the value of the gene expression rate. We observe that if $100 \leq gx \leq 120$ the system exhibits four steady states, if $9 \leq gx \leq 10$ the system is tristable and if $10 \leq gx \leq 100$ the system is bistable. I use the whole range tested above ($0 \leq gx \leq 140$) as prior distributions in Stability Finder and searched parameter space for 2, 3 and 4 steady states.

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

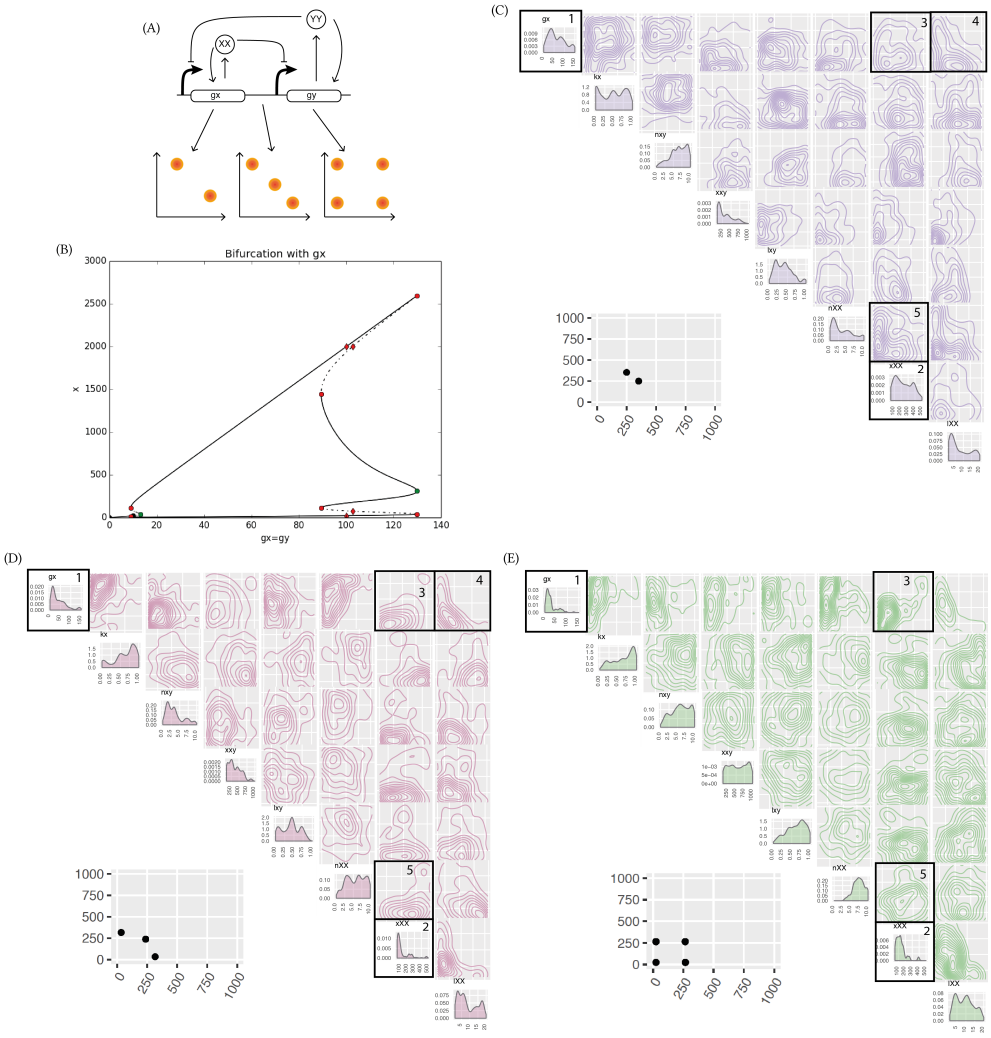

Figure 1.11 : Design principles of multistable switches. (A) Using the Lu model with added positive autoregulation we uncover the design principles dictating if a switch will be bistable, tristable, or will have 4 steady states. (B-D) By considering the bivariate distributions of the parameters we can uncover the differences in the parameters of a bistable switch compared to a tristable switch, compared to a quadristable switch . The posterior distribution of the bistable switch is shown in purple, of the tristable switch in pink and of a quadristable in green. The bivariate distributions for which a difference is observed between teh stabilities are in black boxes. An example of a phase plot from each behaviour is shown next to the corresponding posterior distribution.

Using StabilityFinder, we obtained posterior distributions for bistable, tristable and quadristable behaviours in the DP-LU model and then compared the posterior parameter distributions (Figure 1.11). Upon examination of the posterior distributions for all three switches we observe that a subset of the posterior parameter values is different under the three behaviours. We find differences in the univariate distribution of the parameters for gene expression, gx, as highlighted in Figure 1.11, box 1. This parameter must be small for a quadristable switch to occur but there are no such restraints for a bistable or a tristable switch. Furthermore, parameter xXX must be small for three and four steady states to be achieved but there are no such restraints for a bistable switch, as can be seen in Figure 1.11, box 2.

We also find a difference in the bivariate distributions in the posterior. Most notably, we find that parameters xXX and gX are tightly constrained in the tristable and the four steady state cases, where both parameters are required to be small, but less so in the bistable case (Figure 1.11, box 3). Another notable difference is between parameters xXX and nXX shown in Figure 1.11, box 5, where they are constrained and in the tristable and four steady state cases but not the bistable case. Interestingly, we also find parameter correlations conserved between the three behaviours, as seen in Figure 1.11, box 4, where parameters lXX and gx, positive autoregulation and gene expression are negatively correlated in both cases. This highlights the importance of treating unknown parameters as distributions rather than fixed values when studying the parameter values of a model, as they are capable of uncovering not only the ranges and values needed but also the correlations between parameters that would not have otherwise been detected.

I further analyse these models by studying the phase plots resulting from simulating the particles from the posterior distribution to steady state. The phase plots from 100 particles from each posterior are shown in Figure 1.12. We find that there is a strong conservation on the locations of the steady states between each particle. This indicates that the steady states in a two-node toggle switch tend to be symmetrical. This gives rise to the patterns seen in Figure 1.12. This is especially evident in the quadristable. For every steady state at (0,0) there is another steady state on its diagonal, at $XX = YY$. All the combinations of these two steady states form the straight line seen in Figure 1.12C. This indicates that two of the four steady states exist where XX=YY. The other two exist where one of the two proteins dominates the other.

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

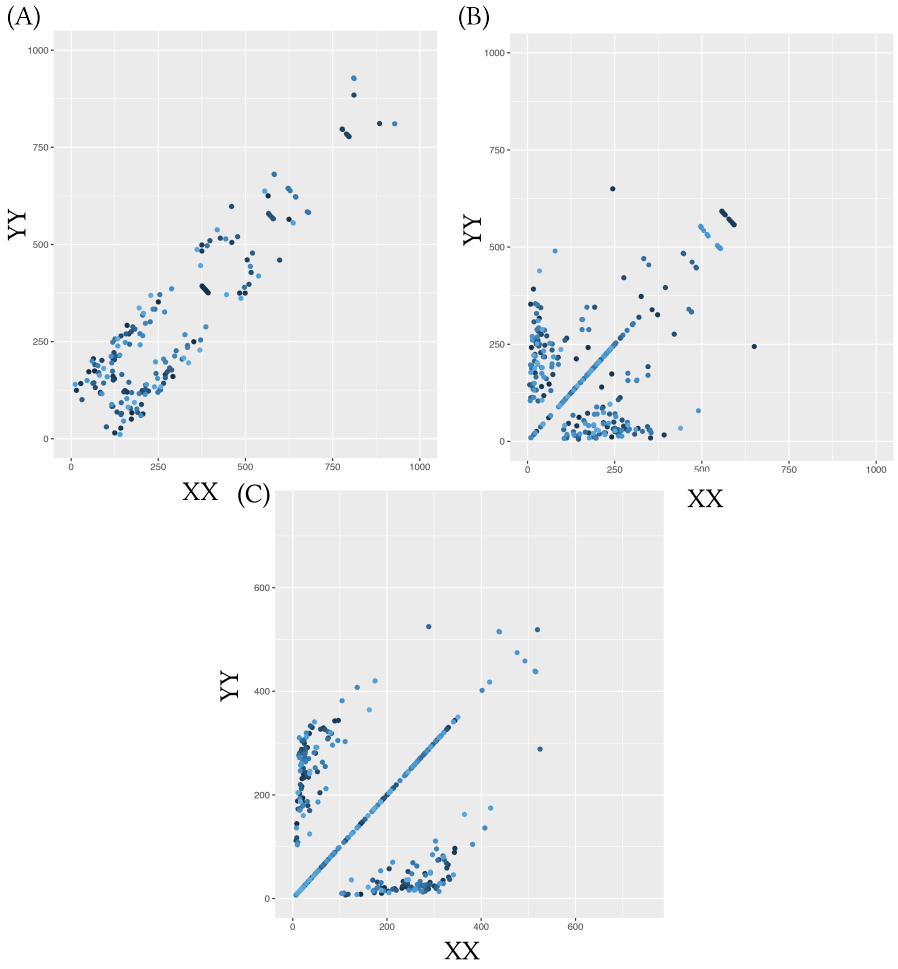

Figure 1.12 : The phase plots from 100 particles from each posterior. Each particle is represented by a different shade of blue. We find a strong conservation on the location of the steady states between particles.

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

1.5.2.3 Extending the Lu switch to three nodes

To further demonstrate the flexibility of Stability Finder I investigated a system capable of higher stabilities. Multistability is found in differentiating pathways, like the myeloid differentiation pathway (**Ghaffarizadeh:2014bt**; **Cinquin:2005go**). I allow for these more complex dynamics by extending the DP-LU model by adding another gene, making it a three gene switch. This new system is depicted in Figure 1.13A. In Stability Finder I 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 1.13C.

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	xxy	140-160
Transcription rate fold change	lxy	0-0.2
Hill coefficient	nxx	2-4
Hill thresholds concentration	xxx	90-110
Transcription rate fold change	lxx	8-12

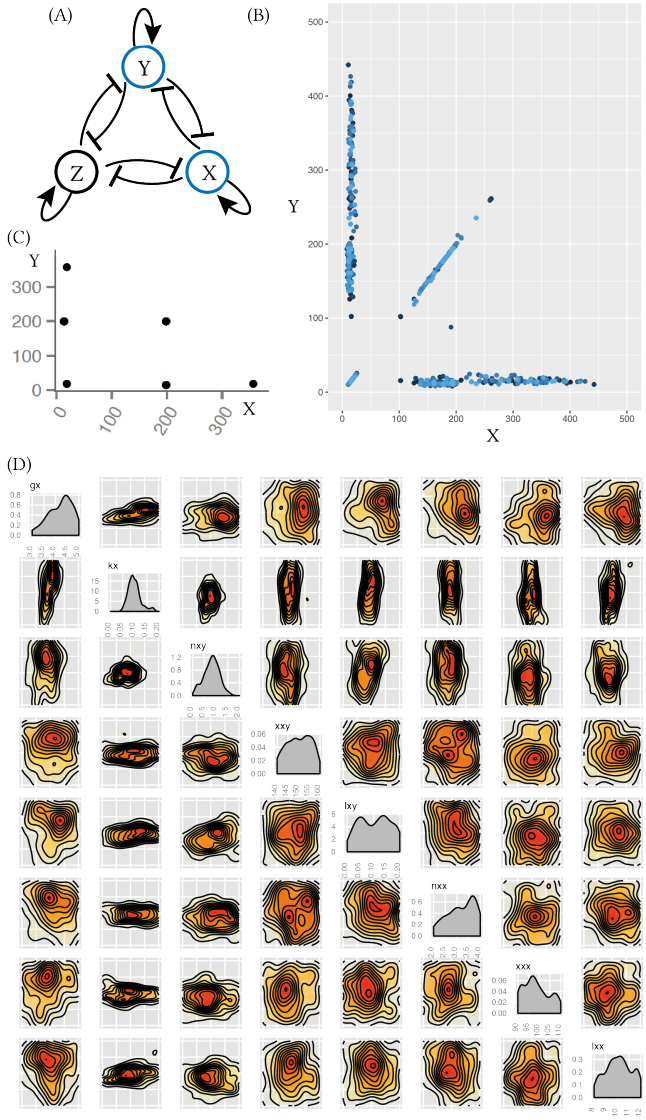

Figure 1.13 : The three-node mutual repression model, with added positive autoregulation on each node. (A) The model. The model is studied in two dimensions using Stability Finder, for nodes X and Y. (B) The phase plot of 100 particles from the posterior found by Stability Finder. There are 6 steady states. (C) The posterior distribution of the 6-steady state three-node system. Parameters kx and nxy are the most constrained.

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 1.13D. This example demonstrates that Stability Finder can be used to elucidate the dynamics of more complex network architectures, which will be key to the successful design and construction of novel gene networks as synthetic biology advances.

Consistently with the results found in section 1.5.2.2, we find that the steady states are consistently symmetric (Figure 1.13B). Each of six steady states exists in symmetry with another one, in tightly constrained regions. This is exaggerated in this model where parameters have been forced to be symmetric.

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.

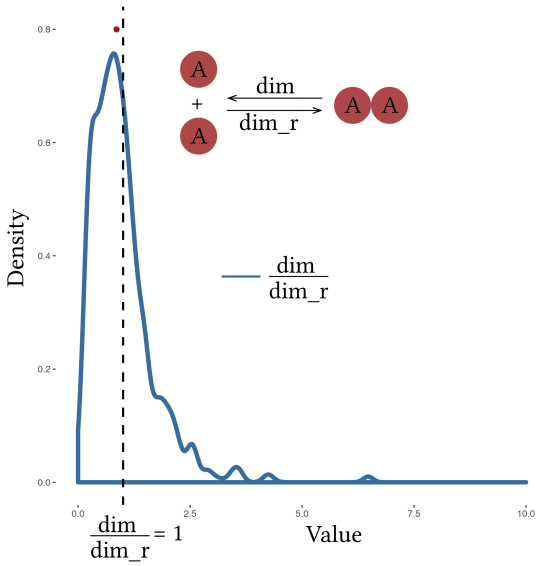

Figure 1.14 : (A) Separating transcription and translation in the mass action model. (B) Transcriptional bursting does not allow for a bistable switch. (C) The Quassi Steady state approximation assumption that the dimerization reaction (*dim*) is much faster than its reverse (*dim_r*) cannot be justified 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	
	Bistable	*Tristable*	*Bistable*	*Tristable*
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

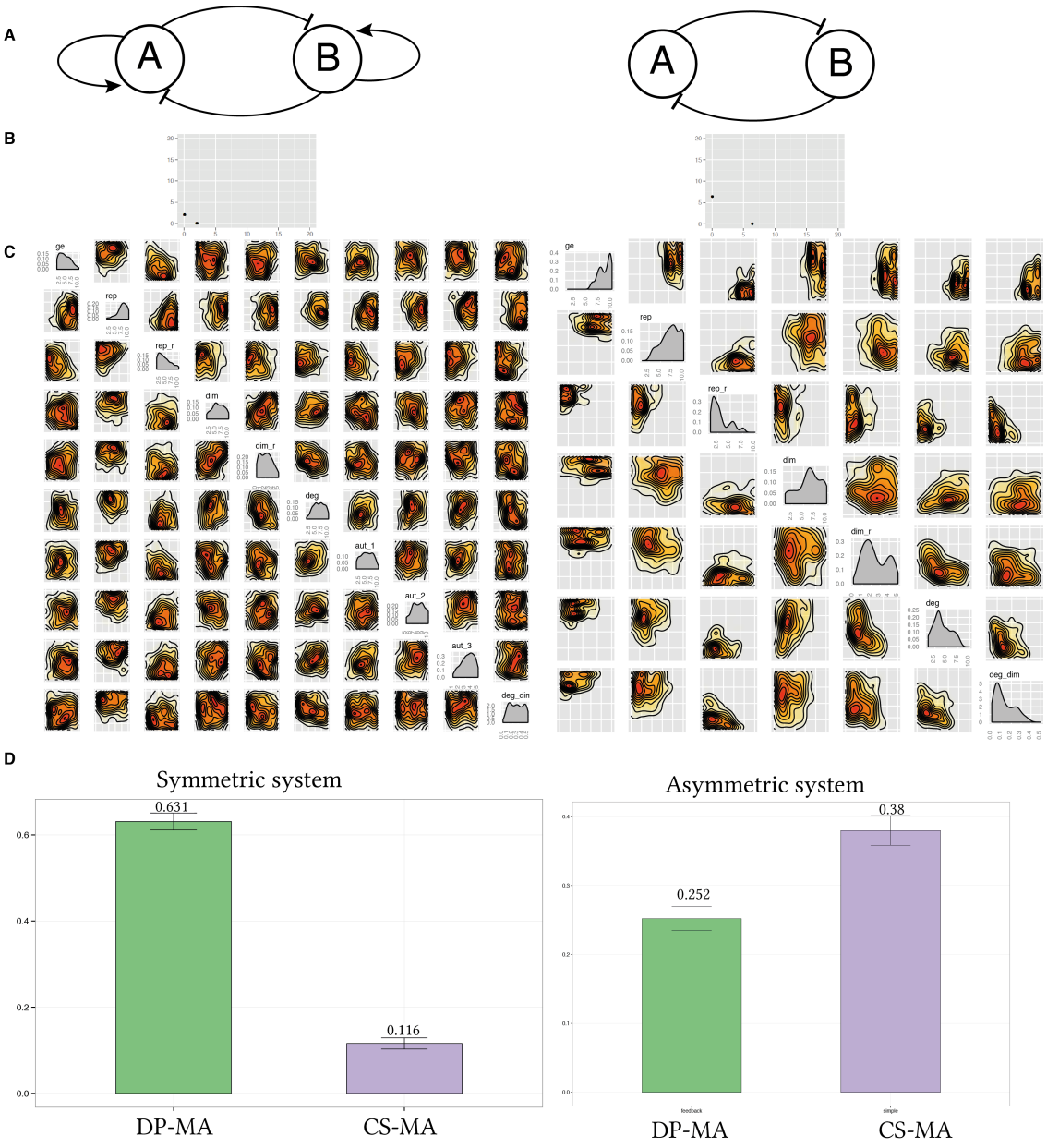

Figure 1.15 : Adding positive feedback increases robustness to parameter fluctuations. We compare the posterior distributions of the simple mass action switch and the switch with added double positive autoregulation (A). The two models were both capable of bistable behaviour (B). The two posterior distributions 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.

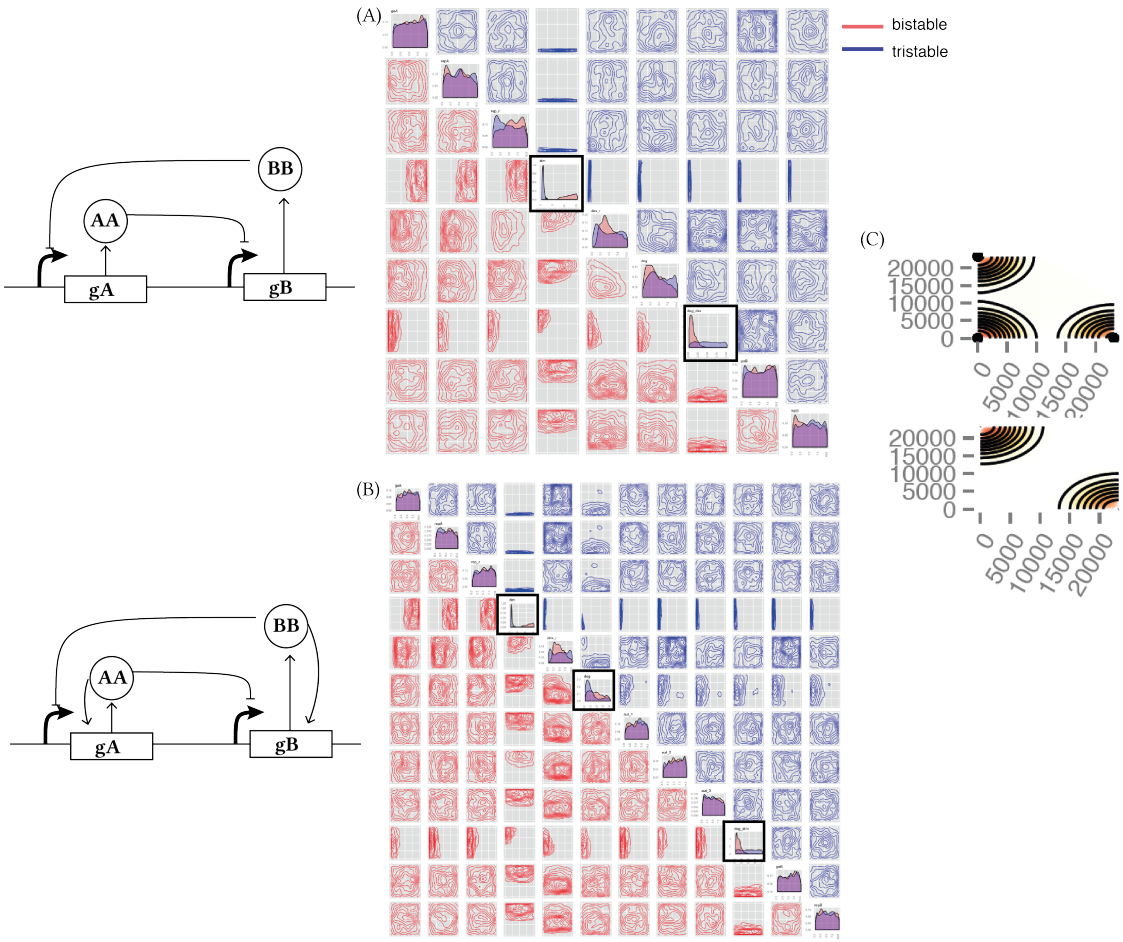

Figure 1.16 : Tristability is possible in the mass action toggle switch models only when simulated stochastically. (A) the simple toggle switch with no autoregulation can be both bistable and tristable. The two posteriors are shown, where the posterior distribution of the bistable switch is shown in 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.

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

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.