

A Computational Approach for Cell Fate Reprogramming

i. Abstract

The notion of reprogramming cell fate is a direct challenge to the traditional view in developmental biology that a cell's phenotypic identity is sealed after undergoing differentiation. Direct experimental evidence, beginning with the somatic cell nuclear transfer experiments of the 20th century and culminating in the more recent breakthroughs in transdifferentiation and induced pluripotent stem cell (iPSC) reprogramming, have rewritten the rules for what is possible with cell fate transformation. Research is ongoing in the manipulation of cell fate for basic research in disease modeling, drug discovery, as well as in clinical therapeutics. In many of these cell fate reprogramming experiments, there is often little known about the genetic and molecular changes accompanying the reprogramming process. However, gene regulatory networks (GRNs) can in some cases be implicated in the switching of phenotypes, providing a starting point for understanding the dynamic changes that accompany a given cell fate reprogramming process. In this chapter, we present a framework for computationally analyzing cell fate changes by mathematically modeling these GRNs. We provide a user guide with several tutorials of a set of techniques from dynamical systems theory that can be used to probe the intrinsic properties of GRNs as well as study their responses to external perturbations.

I. Introduction

The probe into the theoretical question of how to reprogram cell fate can be aided by identification and understanding of the gene regulatory network(s) (GRNs) that are implied in the transformation between one phenotype and another. A common conceptualization of this problem is the Waddington view of cellular differentiation (Waddington, 1957) in which a ball rolls down a hilly landscape with several different valleys representing the multiple paths that undifferentiated cells may take in acquiring their differentiated identity (Figure 1). In an energetic landscape sense, differentiation is represented as a spontaneous 'downhill' process associated with typical development during an organism's periods of growth and maturation. While it was a long-held consensus that this development was a permanent one-way process, experimental manipulations of cell fate during the second half of the 20th century (such as somatic cell nuclear transfer (Briggs and King, 1952; Gurdon et al., 1958; Campbell et al., 1996; Wakayama et al., 1998) and transdifferentiation (Tapscott et al., 1988; Davis et al., 1987)) began chipping away at this notion. In 2006, it was shattered entirely when Yamanaka and colleagues discovered that mouse fibroblasts could be reverted to what was coined an induced pluripotent stem cell (iPSC) state (Yamanka et al., 2006), a transition tantamount to an 'uphill' climb in the Waddington landscape. Remarkably, this transformation was accomplished using the mere overexpression of a small cocktail of transcription factors (TFs): Oct4, Sox2, Klf4, and c-Myc (also known as 'OSKM factors'). It is understood that in successfully reprogrammed cells, the researchers had transformed the phenotype from fibroblast to iPSC by shifting the initial cell state, characterized by low levels of OSKM, to the iPSC state, characterized by higher levels of these factors (Radzishewska, 2013).

While Yamanaka's work and the variations that have followed (González et al., 2013) have been remarkable achievements in their own right, much remains to be answered about the

biochemical and biomolecular changes that accompany many reprogramming processes (Buganim et al, 2013; David and Polo, 2014; Back et al, 2014; Yao et al., 2017). Chief among these questions are those surrounding the low efficiencies, variable quality, and high latency of iPSC formation (Malik and Rao, 2013; Schlaeger and Daheron, 2015; Goh et al., 2013) that persist in the laboratory today. Yamanka himself, when describing the first clinical trial with iPSCs in 2016, remarked *‘what we learned from the first patient, in which we performed autologous iPS cell transplantation, is that the entire process is too expensive. It also took almost a year to make iPS cells from the patient’s own cells, to transfer the original cells, and to perform all the rigorous quality control tests. It was too expensive, and it took long’* (‘iPS Cells 10 Years Later’, 2016). Even when other cell fate transitions occur with greater efficiency in the laboratory, such as in certain transdifferentiation protocols (Zhou et al., 2008; Pang et al., 2011; Xie et al., 2004; Bussmann et al., 2009; Laiosa et al., 2006; Vierbuchen et al., 2010; Forsberg et al., 2010), many open questions surrounding the biomolecular changes that take place during the reprogramming remain.

As a result, iPSC reprogramming, and more generally cell fate reprogramming, is an active area of research. From a theoretical standpoint, understanding the complex dynamics associated with the GRNs involved in cell fate determination can be a promising approach to provide insight into current bottlenecks and suggest novel approaches. The task of creating mathematical models of GRN dynamics is today rendered possible by the vast amounts of genomic and molecular data available in the post-genomic era (Yu et al., 2004). In particular, for iPSC reprogramming, a key GRN was discovered by Boyer and colleagues in 2005 (Boyer et al., 2005). This GRN, which is considered to be the core pluripotency network, consists of a ‘fully-connected-triad’ of mutually activating transcription factors, Oct4, Sox2, and Nanog. Activating this network has been implicated as a signature of achieving pluripotency starting from differentiated cells, and the network itself is known to be part of an extended regulation network involving other TFs, various signaling components, differentiation genes, and chromatin remodelers (Boyer et al., 2005; Zhang and Wolynes. 2014; Orkin et al., 2008; Chickarmane and Peterson, 2009; Bieberich and Wang, 2013). Other well-studied cell fate reprogramming pathways include transdifferentiation pathways between different phenotypes in the hematopoietic stem cell (HSC) lineage. For instance, in the transdifferentiation between the megakaryocyte lineage and the granulocyte/macrophage lineage, a core GRN involving the transcription factors PU.1 and GATA-1 has been implicated, supporting the fact that the two lineages are characterized by high GATA-1, low PU.1 levels and high PU.1, low GATA-1 levels, respectively (Goldfarb, 2007; Friedman, 2007; Gupta et al, 2009).

When sufficient data about a cell fate reprogramming process exists, mathematical models of the GRN(s) in question may be constructed and thus used as a representation of the reprogramming process itself. The utility of such models lies in the extent to which they can be used to explain or predict experimental outcomes, shed light on biomolecular mechanisms taking place during the reprogramming process, or suggest new informative experiments. There is a plethora of models that have been devised for various cell fate reprogramming contexts. Given their importance at the top of the developmental hierarchy, several models have been constructed for the various TFs involved in establishing mammalian pluripotency, including those composing the core network of Oct4, Nanog, and Sox2 (the ‘triad’) discovered by Boyer and colleagues in 2005. Shortly after Boyer’s discovery, (Chickarmane et al. 2006) proposed a deterministic model

of the triad that demonstrated bistability. According to their work, when the triad is expressed (the pluripotent state, or ‘on’ state), self-renewal genes that establish pluripotency were also shown to be expressed while differentiation genes were off, and vice versa when the triad was not expressed. The model also predicted that if input to the system turned on the switch, the network could persist in the ‘on’ state without sustained input if the binding strength of Nanog to Oct4 and Sox2 was increased or the constitutive transcription level of Nanog was increased. (Chickarmane et al. 2009) also constructed a deterministic model of the triad along with two other TFs, Cdx2 and Gata6. Their model showed that antagonisms of Cdx2 and Gata6 with Oct4 and Nanog, respectively, are involved in controlling the murine pluripotent state’s differentiation into the trophectoderm and endoderm lineages, respectively. In addition, their work explored various reprogramming paths from these differentiated lineages back to the pluripotent state. We have also previously used a mathematical model for the triad to explore questions surrounding the outcome of reprogramming experiments and how to explain their prevalent failures (Del Vecchio et al., 2017; Abdallah et al., 2016). (Olariu et al, 2016) have studied the impact of demethylation on the iPSC reprogramming process, constructing a deterministic model of Oct4, Nanog, and Tet, which is a protein that demethylates the Oct4 promoter before it can be reactivated. This deterministic model was combined with a stochastic description of methylation and demethylation at the promoters of Oct4 and Nanog to form a multi-layered model that underscored the importance of demethylation in the iPSC reprogramming process.

Several mathematical models have also been published for the well-studied network involving the TFs GATA-1 and PU.1 that govern the decision of the multipotent common myeloid precursor (CMP) to differentiate into either the erythroid or myeloid lineages. (Huang et al, 2007) used a differential equation model to mathematically describe the differentiation of a ‘meta-stable’ CMP state into either a stable erythroid or myeloid lineage by way of stochastic fluctuations or directed deterministic perturbations. At a mathematical level, the bistability underlying this model hinged upon an assumption that the functions describing the interactions between the two TFs were sigmoidal. However, high levels of TF cooperativity, which is typically what gives these functions sigmoidal shape (Santillán, 2008), was not consistent with experimental results on how PU.1 and GATA-1 were interacting (Liew and Rand, 2006). A model by (Chickarmane et al, 2009) pointed this out and suggested the need for a potentially undiscovered cofactor mechanism to establish the bistability seen experimentally. (Tian and Smith-Miles, 2014) constructed a stochastic model of the GATA-1/PU.1 GRN with the TF GATA-2 incorporated as well, and showed the existence of three states that correspond to three different blood cell phenotypes that could be reached stochastically from the same starting conditions (as is the case in real-life development). Another example includes the work by (Zhou et al., 2011) which proposes a differential equation model of ten essential genes involved in pancreatic cell differentiation that predicts which artificial perturbations are needed to induce transitions among the various pancreatic cell types.

The purpose of this chapter is to illustrate in detail how practitioners of cell fate reprogramming (including iPSC reprogramming and transdifferentiation) can mathematically model their particular process in question if there is already a known GRN implied in the transformation between the phenotypes of interest. In Section II, we discuss how to use such GRNs to model the resulting dynamics of species in the network using ordinary differential equation (ODE) models (Figure 2, top panel). The validity of such modeling techniques rests on

some key assumptions which will be explained in detail as a prerequisite to the approaches applied.

In Section III, we introduce the notion of reprogramming in the state-space domain, while in Section IV we provide an introduction to analysis tools from the fields of dynamical systems (Strogatz, 2014) that can be used to *understand the intrinsic properties and behavior of the GRN* (Figure 2, middle panel). By moving the reprogramming problem to the state-space domain, it becomes possible to characterize the admissible steady state landscapes of the system, understand how robust these are to parameter changes, and determine which state transitions are possible under the influence of external perturbations. Most importantly, the comparison between the outcome of an experiment and the model prediction using these tools can often reveal biological mechanisms that are necessary to explain the data. Finally, this workflow will culminate in Section V, where we provide a discussion of the analysis of how a GRN responds to artificial perturbations, such as those imposed by reprogramming practices (Figure 2, bottom panel). Questions that can be addressed at this level of analysis include: how do the steady state landscape reconfigure after artificial perturbations? what inter-state transitions can or cannot be triggered with external perturbations, and what type of interventions are required to trigger a particular transition.

II. Constructing an Ordinary Differential Equation Model of a Gene Regulatory Network

As introduced above, viewing the problem of cell fate reprogramming through the lens of a mathematical model can lend many useful insights into the biology of a given reprogramming practice and potentially inform future experiments. In this section, we show by way of the example in Figure 2 how to construct an ODE model for a GRN. ODE models provide an appropriate starting point to gauge a deeper understanding of the GRN dynamics, and hence we focus this chapter on them. However, ODE models are suitable only when the molecular counts of species are sufficiently high such that the system can be described in terms of concentrations (see (Del Vecchio and Murray, 2014) and (Gillespie, 2009) for a detailed overview of assumptions behind ODE models of biomolecular systems). In such situations, the ODE models provide an accurate description of the mean behavior of the system dynamics. When one is interested in stochastic properties, such as determining the response of a system to noise, or the system is characterized by low molecular counts, other descriptions than ODEs should be considered. The Chemical Langevin Equation (CLE) is a stochastic differential equation that describes the system dynamics by adding a noise term to the ODE (Gillespie, 2000). Just like ODEs, CLEs can be used only when concentration is a meaningful way of describing the system. When this is not the case and molecular counts are very low, the Chemical Master Equation (CME) is the appropriate description (Van Kampen, 1992). In these cases, the behavior of an ODE model can be dramatically different from the mean behavior of the stochastic model (Al-Radhawi et al., 2017).

There are many different ODE models that one could write for a GRN of interest (Polynikis et al, 2009). In our representation, we describe a network involving n TFs, or n ‘nodes,’ using n ODEs, with each equation describing the rate of change of the concentration of a single TF. If $\mathbf{X} = (X_1, X_2, \dots, X_n)$ represents the n nodes in a GRN with concentrations $\mathbf{X} = (X_1, X_2, \dots, X_n)$ respectively, we write an ODE model for their concentrations as follows:

$$\frac{dX_i}{dt} = H_i(\mathbf{X}) - \gamma_i X_i,$$

where $H_i(\mathbf{X})$ is the production rate of TF X_i that lumps together the effects of all activators and repressors of gene of X_i and γ_i is the decay rate of the TF, which encompasses both active degradation and dilution (Del Vecchio and Murray, 2014). We will derive $H_i(\mathbf{X})$ ($i = 1,2$) for the simple GRN network with nodes X and Y in Figure 2 using a set of molecular interactions between these TFs themselves, the genes that encode for them, and the mRNA transcripts from which they are translated (we use the nodes X and Y rather than X_1 and X_2 to simplify notation). Though we decide upon these interactions and declare them to be true for illustrative purposes, in general these will have to be determined using a literature search surrounding the particular GRN being studied: different TFs have different binding behaviors to their gene targets, and this is one of the factors that affect the final functional form of $H_i(\mathbf{X})$. Moreover, there is also freedom in deciding which molecular species to include in a representation of the interactions based on the degree of detail one is interested in capturing. We begin describing these interactions for what will be our example in Figure 2 using a biochemical reaction model.

1. From Literature Data to Biochemical Reaction Model

Formation of Transcription Factor Multimers (“Multimerization”)

It is common for TFs in a GRN to act as multimers, or proteins consisting of multiple monomers. In general, a literature search should be conducted to determine the most probable multimerization state of each TF in the network under study. As will be shown, whether a TF is acting as a monomer, dimer, trimer, etc., will be captured by a key parameter during ODE construction. In our running example, for illustrative purpose we treat TFs X and Y as acting in their homo-dimerized forms, X_d and Y_d , respectively. The biochemical reactions corresponding to these reversible dimerizations are shown in Table 1A.

Transcriptional Level Activation

In the basic modeling formalism presented here, we assume that regulatory interactions, such as the activations between TFs X and Y in Figure 2, occur at the transcriptional level. Other regulatory mechanisms are possible, such as regulation of translation rate through microRNAs (Agrawal et al, 2003), and regulation of protein activity through covalent modification (Berg et al 2002). Here, for simplicity, we only consider transcriptional regulation since it is typically better characterized experimentally. In particular, we treat the arrow going from node X to node Y (node Y to node X) in the GRN as representing protein product X (Y) activating the promoter of gene Y (X). Likewise, the ‘auto-regulating’ arrow going from node X (Y) to X (Y) itself represents the TF X (Y) activating its own promoter.

There are a number of ways two TFs can co-bind to a common promoter (Gyorgy and Del Vecchio 2014).

Independent Binding: the two TFs bind to the promoter on separate sites and do not affect each other’s binding/unbinding.

Cooperative Binding: the second TF only binds to the promoter when the first TF is already bound, or vice versa.

Competitive Binding: the two TFs compete for the same binding site, which they mutually exclusively occupy. Hence, if one TF is bound to the promoter, the other TF cannot bind.

In general, the mode(s) of binding for each TF in a network will have to be determined from the literature. In our model, we treat TFs X and Y as acting independently, which means they can occupy their targets alone or together. These configurations are captured by the specific association and dissociation rate constants in Table 1B, where we have introduced free DNA promoters D_X and D_Y for genes X and Y, respectively. Based on the binding modes shown in Table 1, the promoters of gene X and gene Y can exist as:

- D_X : this species represents the unbound promoter of gene X.
- D_{XX} : this species represents D_X bound by the homodimer X_d alone.
- D_{XY} : this species represents D_X bound by the homodimer Y_d alone.
- D_{XXY} : this species represents D_X bound by the homodimers X_d and Y_d .
- D_Y : this species represents the unbound promoter of gene Y.
- D_{YY} : this species represents D_Y bound by the homodimer Y_d alone.
- D_{YX} : this species represents D_Y bound by the homodimer X_d alone.
- D_{YYX} : this species represents D_Y bound by the homodimers X_d and Y_d .

There are a variety of experimental strategies used for the discovery of DNA binding sites for a TF of interest (Geertz and Maerkl, 2010). Some common techniques for genome-wide binding site discovery are ChIP-on-Chip (Horak and Snyder, 2002) and ChIP-Seq (Park, 2009). In ChIP-on-Chip, the TFs of interest are cross-linked to chromatin that is then precipitated using antibodies specific for each TF, hence revealing their binding site to a particular DNA sequence. When done in a high-throughput manner using microarray chips, this is a practical way to reveal binding sites at the genome scale. In ChIP-Seq, TFs are similarly cross-linked and precipitated, and then the precipitated samples are run through high-throughput sequencing where peak-finding software tools are used to identify the presumed TF binding sites.

Once it is determined that a protein is acting as a TF, elucidating the direction of transcriptional regulation is commonly done by measuring the fold change of transcription in separate experiments in which the TF in question is knocked out or overexpressed. Once again, this can be done in a high-throughput manner using microarray chips or RNA-sequencing (Wang and Snyder, 2009) to determine transcription levels. In addition, experimenters may sometimes find the same TF acting in a context-dependent manner as both an activator and repressor. In such cases, further investigation is needed to determine if another mechanism, such as context-dependent cofactors, are influencing the TF's effect on transcription.

At the biochemical reaction level, the representation for activators and repressors (Figure 3) is similar. A repressor R can bind a free promoter D at its target to form a repressed promoter D_R . The fact that this DNA promoter is repressed rather than activated will be reflected in the fact that the relative transcription rate from the promoter configuration D_R is less than that from the non-repressor bound promoter, D. Likewise, if A is an activator, the transcription rate from the promoter configuration representing A bound to its target promoter, D_A , will be higher than the

transcription rate on the non-activated target promoter, D . We illustrate these details in the following section for our running example of Figure 2.

Transcription and Translation

Next we provide biochemical reaction models for transcription and translation. In prokaryotes and eukaryotes alike, transcription is an elaborate process involving several steps including initiation, elongation, and termination of an mRNA transcript, amongst others (Alberts et al., 2002). Likewise, translation consists of several steps including polypeptide formation and elongation and is followed by protein folding before the protein can reach its active form. A full mechanistic model of the transcriptional and translational apparatuses is laid out in equations 2.10 and 2.11 of Chapter 2 in (Murray and Del Vecchio, 2014). In modeling GRNs for cell fate reprogramming, however, this level of detail is not typically important for capturing the qualitative behavior of a network and its response to external perturbations. As derived in (Murray and Del Vecchio, 2014), we can treat transcription and translation as single step reactions with lumped rates that encompass the time it takes for all the steps in the elaborate machinery to complete.

For our particular GRN of Figure 2, these one-step biochemical reactions are shown in Table 1C, where each DNA promoter state established above produces an mRNA transcript, m_X or m_Y , from the genes of X and Y , respectively. Table 1C distinguishes between constitutive, or ‘leaky’ transcription that represents the basal rate of transcription of a gene without activation or repression, and activated transcription in our example. The degree to which constitutive transcription occurs will have to be determined from the literature for each gene in question; some genes will constitutively be active while others only transcribe mRNA upon activation. Moreover, the ‘activating’ property of these reactions will be encoded in the fact that the rates of transcription for activated DNA will be greater than the rate of constitutive transcription, as shown in the table.

To summarize Table 1C, the mRNA transcript, m_X , of gene X can be produced in 4 ways:

- From the constitutive promoter of gene X , D_X , at a rate α_x (unit time)⁻¹.
- From the singly-activated promoter D_{XX} at a rate α_x^1 of mRNA transcripts/unit time.
- From the singly-activated promoter D_{XY} at a rate α_x^2 of mRNA transcripts/unit time.
- From the dually-activated promoter D_{XXY} at a rate α_x^3 of mRNA transcripts/unit time.

Because the species D_{XX} , D_{XY} , and D_{XXY} represent activated forms of D_X , the rates of transcription from these species will be greater than from constitutive transcription, i.e. $\alpha_x^1, \alpha_x^2, \alpha_x^3 > \alpha_x$. Since the GRN is symmetric, similar representations hold for the way m_Y is produced.

Note that if these promoters were being repressed rather than activated, the only change would be in the relative rates of transcription between free promoters and repressor-bound promoters, i.e. $\alpha_x^1, \alpha_x^2, \alpha_x^3 < \alpha_x$ and $\alpha_y^1, \alpha_y^2, \alpha_y^3 < \alpha_y$.

TABLE 1: Dimerization, DNA-Binding, and Transcription Reactions

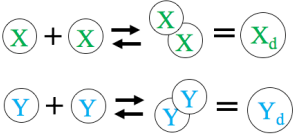
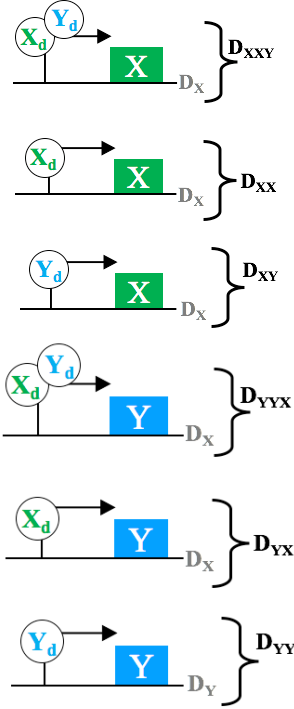
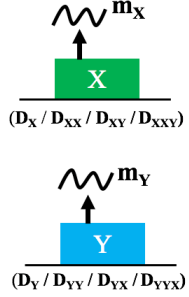
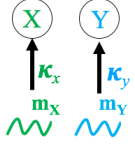
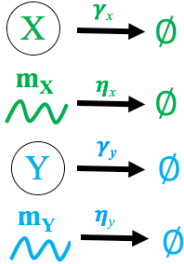
Reaction Type	Biochemical Reaction Model Representation		
A. Dimerization 	$X + X \xrightleftharpoons[d_x]{a_x} X_d$	$Y + Y \xrightleftharpoons[d_y]{a_y} Y_d$	
	Represents TF X reversibly forming a homo-dimer, X_d . Units of a_x : $(\text{concentration} \cdot \text{time})^{-1}$, d_x : $(\text{time})^{-1}$	Represents Y reversibly forming a homo-dimer, Y_d . Units of a_y : $(\text{concentration} \cdot \text{time})^{-1}$, d_y : $(\text{time})^{-1}$	
B. Reversible DNA Binding 	INDEPENDENT ACTIVATION		
	AUTO ACTIVATION	MUTUAL-ACTIVATION	
	$D_X + X_d \xrightleftharpoons[d_1]{a_1} D_{XX}$ Represents homo-dimer X_d reversibly binding its own free promoter, D_X , to form an activated promoter D_{XX} . $K_1 = d_1/a_1$	$D_X + Y_d \xrightleftharpoons[d_2]{a_2} D_{XY}$ Represents homo-dimer Y_d reversibly binding the free promoter of X, D_X , to form an activated promoter D_{XY} . $K_2 = d_2/a_2$	
	$D_Y + Y_d \xrightleftharpoons[d_4]{a_4} D_{YY}$ Represents homo-dimer Y_d reversibly binding to the promoter of Y, D_Y , to form activated promoter D_{YY} . $K_4 = d_4/a_4$	$D_Y + X_d \xrightleftharpoons[d_3]{a_3} D_{YX}$ Represents homo-dimer X_d reversibly binding to the promoter of Y, D_Y , to form activated promoter D_{YX} . $K_3 = d_3/a_3$	
	CO-ACTIVATION		
	$D_{XX} + Y_d \xrightleftharpoons[d_5]{a_5} D_{XXY}$ (left) Represents homo-dimer Y_d reversibly binding the promoter of X when X_d is already bound (i.e. D_{XX}) to form an independently co-activated promoter D_{XXY} . $K_5 = d_5/a_5$	$D_{XY} + X_d \xrightleftharpoons[d_6]{a_6} D_{XXY}$ (right) Represents homo-dimer X_d reversibly binding the promoter of X when Y_d is already bound (i.e. D_{XY}) to form an independently co-activated promoter D_{XXY} . $K_6 = d_6/a_6$	
	$D_{YX} + Y_d \xrightleftharpoons[d_7]{a_7} D_{YYX}$ (left) Represents homo-dimer Y_d reversibly binding the promoter of Y when X_d is already bound (i.e. D_{YX}) to form an independently co-activated promoter D_{YYX} . $K_7 = d_7/a_7$	$D_{YY} + X_d \xrightleftharpoons[d_8]{a_8} D_{YYX}$ (right) Represents homo-dimer X_d reversibly binding the promoter of Y when Y_d is already bound (i.e. D_{YY}) to form an independently co-activated promoter D_{YYX} . $K_8 = d_8/a_8$	
	Units of $a_i = (\text{concentration} \cdot \text{unit time})^{-1}$. Units of $d_i = (\text{unit time})^{-1}$ ($i \in \{1 \dots 8\}$)		
C. Transcription 	CONSTITUTIVE TRANSCRIPTION	ACTIVATED TRANSCRIPTION	
	$D_X \xrightarrow{\alpha_x} m_X + D_X$ $D_Y \xrightarrow{\alpha_y} m_Y + D_Y$	$D_{XX} \xrightarrow{\alpha_x^1} m_X + D_{XX}$ $D_{XY} \xrightarrow{\alpha_x^2} m_X + D_{XY}$ $D_{XXY} \xrightarrow{\alpha_x^3} m_X + D_{XXY}$	$D_{YY} \xrightarrow{\alpha_y^1} m_Y + D_{YY}$ $D_{YX} \xrightarrow{\alpha_y^2} m_Y + D_{YX}$ $D_{YYX} \xrightarrow{\alpha_y^3} m_Y + D_{YYX}$
	Transcription of genes X and Y without activation, i.e. “leaky” transcription.	Represents transcription of activated genes of X to form species m_X .	Represents transcription of activated genes of Y to form species m_Y .
	for activated DNA: $\alpha_x^i, \alpha_y^i > \alpha_{x_0}, \alpha_{y_0}$, for repressed DNA: $\alpha_x^i, \alpha_y^i < \alpha_{x_0}, \alpha_{y_0}$ ($i \in \{1,4\}$). Units for all rate constants: $(\text{unit time})^{-1}$		

TABLE 1 (cont.): Translation, Decay and Ectopic Overexpression Reactions					
Reaction Type	Biochemical Reactions				
D. Translation 	$m_X \xrightarrow{\kappa_x} m_X + X \qquad m_Y \xrightarrow{\kappa_y} m_Y + Y$ <p>Represents translation of mRNA transcript of gene X into the protein product X. Represents translation of mRNA transcript of gene Y into the protein product Y.</p>				
E. Decay 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">DEGRADATION</th> <th style="text-align: center;">DILUTION</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;"> $X \xrightarrow{\delta_x} \phi \quad Y \xrightarrow{\delta_y} \phi$ $m_X \xrightarrow{\eta_x} \phi \quad m_Y \xrightarrow{\eta_y} \phi$ <p>Represents the degradation of TFs X and Y and mRNA transcripts m_X and m_Y.</p> </td> <td style="text-align: center;"> $X \xrightarrow{\beta} \phi \quad Y \xrightarrow{\beta} \phi$ <p>Represents the dilution of TFs X and Y due to cell-division.</p> </td> </tr> </tbody> </table> <p>In total, the decay rates γ_x and γ_y of TFs X and Y, respectively, are equal to the sum of degradation and dilution rates for each TF: $\gamma_x = \delta_x + \beta, \gamma_y = \delta_y + \beta$.</p>	DEGRADATION	DILUTION	$X \xrightarrow{\delta_x} \phi \quad Y \xrightarrow{\delta_y} \phi$ $m_X \xrightarrow{\eta_x} \phi \quad m_Y \xrightarrow{\eta_y} \phi$ <p>Represents the degradation of TFs X and Y and mRNA transcripts m_X and m_Y.</p>	$X \xrightarrow{\beta} \phi \quad Y \xrightarrow{\beta} \phi$ <p>Represents the dilution of TFs X and Y due to cell-division.</p>
DEGRADATION	DILUTION				
$X \xrightarrow{\delta_x} \phi \quad Y \xrightarrow{\delta_y} \phi$ $m_X \xrightarrow{\eta_x} \phi \quad m_Y \xrightarrow{\eta_y} \phi$ <p>Represents the degradation of TFs X and Y and mRNA transcripts m_X and m_Y.</p>	$X \xrightarrow{\beta} \phi \quad Y \xrightarrow{\beta} \phi$ <p>Represents the dilution of TFs X and Y due to cell-division.</p>				
F. Ectopic Overexpression	$\phi \xrightarrow{u_x} m_X \qquad \phi \xrightarrow{u_y} m_Y$ <p>Represents the additional mRNA species m_X and m_Y formed via ectopic overexpression at rates u_x and u_y mRNA species / unit time.</p>				

Though we began with several DNA promoter states, these ultimately led to the same two mRNA transcripts, m_X and m_Y . At this point under the assumptions we have made, translation can also be represented by a one-step process from these transcripts into their protein products as shown in Table 1D.

Decay

While the reactions from above lead to the production of X and Y, we also model the decay of these proteins as well as their respective mRNA species. In general, these proteins and mRNA species will undergo decay at some rate that is a combination of both active degradation (δ_x and δ_y for proteins X and Y, respectively, and η_x and η_y for mRNA species m_X and m_Y , respectively) and dilution (β), as shown in Table 1E. For proteins X and Y: the total rate of decay is $\gamma_x = \delta_x + \beta$ and $\gamma_y = \delta_y + \beta$, respectively.

In general, the degradation parameters can be determined using the following relations (Del Vecchio and Murray, 2014):

$$\gamma_i = \ln(2) / t_{1/2}, \delta_i = \ln(2) / t_{1/2},$$

where $t_{1/2}$ represent the half-lives of each protein or mRNA transcript. The half-life of a species is the time it takes for its concentration to fall to half of its initial concentration (Zhou 2004), and can often be found from the literature. Care should be taken to use the half-life value in a cell environment that most closely resembles the phenotypes in question during the reprogramming

process under study, as half-lives can vary as a function of environment (Kuhar, 2009). Likewise, the dilution rate β can be calculated using the cell division rate, or doubling time, $t_{doubling}$ of the cell which the proteins are contained in: $\beta \approx \ln(2) / t_{doubling}$. The doubling time of prokaryotic cells such as *E. Coli* can be on the order of minutes (Sezonov, 2007) while that of eukaryotic cells can be approximated by the characteristic human cell cycle time of approximately 24 hours (Cooper, 2000).

Overexpression (Artificial Perturbation)

Thus far we have described the reactions that comprise the endogenous components of our GRN. Ultimately, the reprogramming framework we describe will include artificial perturbations to our endogenous GRN that can be captured in our model as follows. As described in Section I, reprogramming is typically done by artificially overexpressing TFs of the GRN through insertion of ectopic DNA (Takahashi and Yamanka, 2016). Therefore, we can model this by adding an additional production rate of the TF's mRNA from the ectopic DNA (Table 1F).

As seen above, this is the phase in the modeling process that will require a literature search to determine the multimerization state of the TFs in the GRN, the configurations in which they can bind the DNA they are regulating, as well as their half-lives and dilution rates. The online database bionumbers.org (Milo et al., 2010) can be a useful source for some of these parameters in many cases.

2. From Biochemical Reaction Model to ODE

With this biochemical reaction model established, we are ready to construct an ODE model for the GRN. Since ODE models are deterministic, they implicitly assume that the species in our biochemical reactions exist at sufficiently high copy numbers and in well-stirred volumes (Section 2.1, Del Vecchio and Murray, 2014).

The procedure for obtaining the ODE for each species introduced is relatively straightforward, with no expertise in differential equations required: one simply must go through every reaction listed in Table 1 and add appropriate terms to the differential equations of each species involved in that reaction (see Section 2.1 of Del Vecchio and Murray, 2014). For a given species S,

$$\text{Change in concentration of S} = \frac{d}{dt}(S) = \dot{S} = \Sigma (\text{all biochemical reaction rates involving S})$$

Doing so for our biochemical reaction network yields:

Table 2: Full 14D Model of GRN	
$\dot{X}_d = a_x X^2 - d_x X_d - a_1 D_X X_d + d_1 D_{XX} - a_3 D_Y X_d + d_3 D_{YX} - a_6 D_{XY} X_d + d_6 D_{XXY} - a_8 D_{YY} X_d + d_8 D_{YYX}$	(1)
$\dot{Y}_d = a_y Y^2 - d_y Y_d - a_2 D_X Y_d + d_2 D_{XY} - a_4 D_Y Y_d + d_4 D_{YY} - a_5 D_{XX} Y_d + d_5 D_{XXY} - a_7 D_{YX} Y_d + d_7 D_{YYX}$	(2)
$\dot{D}_X = -a_1 X_d D_X + d_1 D_{XX} - a_2 Y_d D_X + d_2 D_{XY}$	(3)
$\dot{D}_Y = -a_4 Y_d D_Y + d_4 D_{YY} - a_3 X_d D_Y + d_3 D_{YX}$	(4)
$\dot{D}_{XX} = a_1 D_X X_d - d_1 D_{XX} - a_5 D_{XX} Y_d + d_5 D_{XXY}$	(5)
$\dot{D}_{XY} = a_2 D_X Y_d - d_2 D_{XY} - a_6 D_{XY} X_d + d_6 D_{XXY}$	(6)

$\dot{D}_{YX} = a_3 D_Y X_d - d_3 D_{YX} - a_7 D_{YX} Y_d + d_7 D_{YYX}$	(7)
$\dot{D}_{YY} = a_4 D_Y Y_d - d_4 D_{YY} - a_8 D_{YY} X_d + d_8 D_{YYX}$	(8)
$\dot{D}_{XXY} = a_5 D_{XX} Y_d - d_5 D_{XXY} + a_6 D_{XY} X_d - d_6 D_{XXY}$	(9)
$\dot{D}_{YYX} = a_7 D_{YX} Y_d - d_7 D_{YYX} + a_8 D_{YY} X_d - d_8 D_{YYX}$	(10)
$\dot{m}_X = \alpha_x D_X + \alpha_x^1 D_{XX} + \alpha_x^2 D_{XY} + \alpha_x^3 D_{XXY} - \eta_x m_X + u_x$	(11)
$\dot{m}_Y = \alpha_y D_Y + \alpha_y^1 D_{YY} + \alpha_y^2 D_{YX} + \alpha_y^3 D_{YYX} - \eta_y m_Y + u_y$	(12)
$\dot{X} = \kappa_x m_X - \gamma_x X$	(13)
$\dot{Y} = \kappa_y m_Y - \gamma_y Y$	(14)

This 14-dimensional ODE model of the GRN contains a large number of parameters that makes it impractical to analyze for cell fate reprogramming. However, if we assume that dimerization, DNA binding/unbinding, and mRNA dynamics in Table 1A-1C occur sufficiently faster than protein production and decay in Table 1D-1E, the temporal derivatives of the respective species can be set to zero, indicating that the species concentration reaches its (quasi) steady state very quickly compared to the rest of the system. This is called the quasi-steady state (QSS) approximation and is widely used in modeling biochemical reaction systems (Del Vecchio and Murray, 2014). In our example, this amounts to setting the temporal derivatives in equations (1-12) to zero.

Using the QSS approximation and accounting for the fact that DNA is not destroyed, through the conservation laws $D_{TX} = D_X + D_{XX} + D_{XY} + D_{XXY}$ and $D_{TY} = D_Y + D_{YY} + D_{YX} + D_{YYX}$, we arrive at the 2D model:

$$\begin{aligned}\dot{X} &= \frac{\alpha_{x0} + \frac{a_{xx}}{K_1 K_X} X^2 + \frac{a_{yx}}{K_2 K_Y} Y^2 + \frac{b_x R}{K_1 K_2 K_X K_Y} X^2 Y^2}{1 + \frac{1}{K_1 K_X} X^2 + \frac{1}{K_2 K_Y} Y^2 + \frac{1}{K_1 K_2 K_X K_Y} X^2 Y^2} - \gamma_x X + \bar{u}_x, \\ \dot{Y} &= \frac{\alpha_{y0} + \frac{a_{xy}}{K_X K_3} X^2 + \frac{a_{yy}}{K_Y K_4} Y^2 + \frac{b_y R}{K_3 K_4 K_X K_Y} X^2 Y^2}{1 + \frac{1}{K_X K_3} X^2 + \frac{1}{K_4 K_Y} Y^2 + \frac{R}{K_3 K_4 K_X K_Y} X^2 Y^2} - \gamma_y Y + \bar{u}_y,\end{aligned}$$

where we have further assumed $K_2/K_5 = K_1/K_6 = K_4/K_7 = K_3/K_8 = R$, and the lumped parameters are equal to:

$\alpha_{x0} = D_{TX} \alpha_x / \eta_x$, $\alpha_{y0} = D_{TY} \alpha_y / \eta_y$, $a_{xx} = D_{TX} \alpha_x^1 / \eta_x$, $a_{yx} = D_{TX} \alpha_x^2 / \eta_x$, $b_x = D_{TX} \alpha_x^3 / \eta_x$, $a_{yy} = D_{TY} \alpha_y^1 / \eta_y$, $a_{xy} = D_{TY} \alpha_y^2 / \eta_y$, $b_y = D_{TY} \alpha_y^3 / \eta_y$, $K_X = d_x / a_x$, $K_Y = d_y / a_y$. Without loss of generality, if we take $R = 1$ and $K_X = K_Y = K_i = 1$ ($i \in \{1, 2, 3, 4\}$), we get the model:

$$\begin{aligned}\dot{X} &= \frac{\alpha_{x0} + a_{xx} X^2 + a_{yx} Y^2 + b_x X^2 Y^2}{1 + X^2 + Y^2 + X^2 Y^2} - \gamma_x \cdot X + \bar{u}_x = H_1(X, Y) - \gamma_x X + \bar{u}_x \\ \dot{Y} &= \frac{\alpha_{y0} + a_{xy} X^2 + a_{yy} Y^2 + b_y X^2 Y^2}{1 + X^2 + Y^2 + X^2 Y^2} - \gamma_y \cdot Y + \bar{u}_y = H_2(X, Y) - \gamma_y Y + \bar{u}_y\end{aligned}\quad (15)$$

In (15), the terms $H_1(X, Y)$ and $H_2(X, Y)$ are regulatory functions referred to as Hill functions (Santillán, 2008). In our regulatory functions, the variables X and Y being raised to the power 2

stems from fact that X and Y were assumed to act as homo-dimers. In fact, this exponent is more generally referred to as the Hill coefficient n (Santillán, 2008), where n = the number of subunits in the multimerized state of the TF (i.e. $n = 1$ for monomers, $n = 2$ for dimers, $n = 3$ for trimers, etc).

In summary, the assumptions we have made in arriving at the 2D model of our GRN in (15) are:

1. Regulations Take Place at the Transcriptional Level Only: as shown in Table 1B, the arrows between nodes in the GRN take place through TF protein products binding at the DNA promoters of their targets, where they activate or repress transcription.

2. mRNA Levels Produced via Transcription Reach Steady State Levels Sufficiently Fast. During reduction of the dynamics from 14D to 2D above, setting $\dot{m}_X = \dot{m}_Y = 0$ was predicated on the assumption that mRNA dynamics were sufficiently fast, where sufficiently fast is equivalent to assuming that the decay rates of mRNA are much larger than those of proteins (i.e. $\eta_x, \eta_y \gg \delta_x, \delta_y$). Protein half-lives are typically longer than mRNA half-lives (Schwanhäusser, 2013), and this is a good assumption when they are longer on the order of one magnitude or more.

3. DNA Binding and Unbinding Rates Are Much Faster than Protein Production and Decay: setting the ODEs corresponding to the DNA species to zero during model reduction from 14D to 2D was predicated on the assumption—known as the adiabatic limit (Zhang and Wolynes, 2014)—that DNA binding and unbinding rates in Table 1B are much larger than the production and decay rates of the protein products, i.e. $d_i, a_iX, a_iY \gg \gamma_x, \gamma_y, \kappa_x, \kappa_y$ for $i = 1, \dots, 8$. In prokaryotic cells, the adiabatic assumption is typically accurate (Alon, 2006; Del Vecchio and Murray, 2014), though in eukaryotic cells the time needed to demethylate DNA and unpack chromatin before TFs can bind may mean that the promoter kinetics are slower than assumed (Arjun et al., 2006; Yuan et al., 2016; To and Maheshri, 2010; Mariana et al, 2010). When they become slower than the protein production and decay, the predictions of the reduced model may not be entirely accurate, and it would be preferable to use the full 14D model (i.e. the model without reductions based on QSS assumptions). Nonetheless, the steady states of a reduced model will be in the same location as the steady states of the full model so we proceed with this assumption to illustrate what a first-pass analysis of the qualitative behavior of a GRN looks like.

4. The Species in Question Exist in Sufficiently High Molecular Counts: an assumption that has been held throughout this entire ODE modeling scheme is that the species in question exist at sufficiently high molecular counts.

Now that we have arrived at this reduced ODE model of our GRN, we are ready to represent the Waddington view of cell fate reprogramming from the mathematical perspective.

III. A Dynamical Systems View of Cell Fate Reprogramming

By developing a mathematical model for the cell fate reprogramming problem in Section II, we can now study its properties using dynamical systems tools. Within a dynamical systems description, a GRN consisting of the n TF species X_1, X_2, \dots, X_n has a “state” given by the tuple

of TF concentrations $\bar{X} = (X_1, X_2, \dots, X_n)$. The GRN can be described using n ODEs representing the evolution of these concentrations: $\dot{\bar{X}} = \bar{f}(\bar{X})$, in which we use $\dot{\bar{X}}$ for $d\bar{X}/dt$. We refer to the set of points (S^1, \dots, S^m) in state space such that $\bar{f}(S^i) = 0$ (for $i = 1 \dots m$) as the *steady states* of this dynamical system, which can be either stable or unstable (Strogatz, 2014). Qualitatively, stable steady states (SSS) are those that can withstand or damp out small amplitude disturbances, and are therefore associated to a phenotype in this context (Kauffman, 1973; Huang and Eichler, 2005; Huang, 2009)

The distinction between a stable and unstable steady state (USS) can be visualized in Figure 4A, which portrays the states of a GRN as a landscape of wells. In this landscape, SSSs are equivalent to valley-like depressions while USSs are equivalent to ‘peaks.’ On the one hand, under the influence of gravity pointing downwards in the figure, any arbitrarily small perturbation applied on a hypothetical ball that is resting on a peak will cause it to permanently lose its position on that peak (and hence that state is unstable). On the other hand, a ball resting in a depression can withstand some level of perturbation and still converge back into that well (and hence that state is stable).

In addition, SSSs have varying degrees of stability, which can in part be captured by the basin of attraction surrounding each SSS, which is the set of states (TF concentrations) starting from which the system converges autonomously to that SSS. For instance, in Figure 4A, if the SSS themselves are the bottom-most points in the wells, then the entire well is the basin of attraction of that SSS. Hence in the metaphor of this figure, cell fate reprogramming is akin to enforcing transitions from the basin of attraction of the starting phenotype (one well), to the basin of attraction of the final phenotype (another well). This is accomplished by artificially perturbing (i.e. applying an input to) the endogenous GRN just the right amount such that the landscape pushes the system’s state from the basin of attraction of the starting state towards the basin of attraction of the desired final state (Figure 4B).

In Figure 5, we depict this reprogramming concept for a prototypical two dimensional system consisting of two TFs, X and Y. The state space hence consists of concentrations of these TFs: X and Y . In Figure 5A, when no input is applied, the steady state landscape of the endogenous GRN alone consists of three SSSs: S^1 , S^2 , and S^3 . As introduced above, we view these states as corresponding to distinct phenotypes characterized by the relative concentrations shown, and with basins of attraction surrounding each SSS as highlighted in the figure. When an artificial perturbation is introduced for the purpose of reprogramming, the steady state landscape morphs into one that may have a different number of SSSs at different locations. (Figure 5B). Specifically, the original states S^1 , S^2 , and S^3 disappear and two new stable steady states, $S^{2'}$ and $S^{3'}$, arise as depicted. As a result, once the perturbation is applied on a system starting in the location which was originally S^1 , the system’s state will converge towards $S^{2'}$. Upon removal of input in Figure 5C (i.e. completion of the cell fate reprogramming experiment), the system’s state now at $S^{2'}$ is found in the basin of attraction of the original system’s SSS S^2 , and therefore the system’s state ultimately converges to S^2 . In essence, in a dynamical system representation, cell fate reprogramming corresponds to morphing the steady state landscape of the GRN through external perturbation such that the current system’s state (current phenotype) is pushed into the basin of attraction of the target state (target phenotype). It is in general non-trivial to determine external perturbations that are appropriate to trigger a desired state transition. However,

mathematical and computational analysis of the ODE model can reveal useful information in this regard, as we illustrate in the next sections.

To summarize, Figure 6 juxtaposes the different views of reprogramming at the three levels of abstraction discussed thus far: reprogramming as transitions up and across the Waddington landscape, as transitions in state space, and as an experimental change in the phenotype itself.

IV. Analysis of Intrinsic Properties of GRN using Tools from Dynamical Systems

Having introduced the notion of SSS as phenotypes, in this section we provide tools for mathematical characterization of the steady state landscape by continuing with our running example of Figure 2. In particular, we analyze the intrinsic properties of this GRN by analyzing the steady state landscape of our system in (15) without input (i.e. $\bar{u}_x = \bar{u}_y = 0$). Studying this steady state landscape is of utmost importance, as having a correspondence between the mathematical steady states of an ODE model and the phenotypes being studied in a particular cell fate reprogramming process speaks to the strength of a model and its ability to represent the phenotype switching in question.

In general, the mathematical steady states of an ODE model, regardless of its dimensionality, are the solutions to simultaneously setting the time derivatives of all species in the GRN to zero. In our running example of a GRN in Figure 2, the mathematical steady states of its 2D model in (15) are solutions to the equations $dX/dt = dY/dt = 0$. One of the benefits of having a 2D model is that these solutions can be viewed in a 2D-plane, as depicted in Figure 7. As shown, setting these time derivatives to zero yields two equations whose solutions are ‘nullclines’ in the X-Y plane. Viewed in this manner, the solutions to $dX/dt = dY/dt = 0$, and hence the steady states, are given by the intersection of these nullclines.

Stability of Steady States

Once the steady states of a system have been determined, there are different options for determining their stability, depending on the dimensionality of the model. For a system $\dot{\bar{X}} = \bar{f}(\bar{X})$ with $\bar{X} = (X_1, X_2, \dots, X_n)$ and $\bar{f}(\bar{X}) = (f_1(\bar{X}), \dots, f_n(\bar{X}))$, the vector $\bar{f}(\bar{X})$ is known as the vector field, which has a different length and direction as a function of \bar{X} . Qualitatively, plotting this vector at different points across the state space indicates the local direction and speed at which a point in state space would move under the dynamics of the system. Hence *arrows in a vector field point towards SSSs while they point away from USSs*. (Del Vecchio and Murray, 2014). For systems with dimension higher than 2, plotting the vector field may not be practical and therefore algebraic tools, such as the Jacobian matrix are more informative. We illustrate the vector field approach with the 2D model in (15), and then illustrate the Jacobian matrix approach using a 3D GRN model.

2D Geometric Analysis

With the choice of parameters shown in Figure 7, there are five intersections, or steady states. In the code block of the Figure, we use the Mathematica functions `ContourPlot` and `VectorPlot` to simultaneously plot these nullclines and the vector field, respectively. As shown, this allows for a very simple visual inspection of stability at each steady state, and we can

see that our system has three SSSs and two USSs. The same principles allow for simple visual inspection of stability for 1D systems, an example of which is shown in Appendix Figure 1.

n-D Computational Analysis

In GRNs with more than two TFs, such as the three node network shown in Figure 8A, the reduced ODE system that can be obtained through the workflow from Section II has greater than two dimensions. Equation (16) of Figure 8A shows the reduced 3D ODE model that can be obtained for the GRN shown. In these cases, the steady state landscape may be inferred using computational techniques outlined in Figure 9. This workflow is based on standard theory from dynamical systems and linear algebra (Strang, 1993; Strogatz, 2014), though in-depth knowledge of these concepts is not required for use of the code we provide.

The workflow is as follows: one can first solve for the steady states of an *n*-D system using an algebraic solver such as `vpasolve` in Matlab (Fig. 9A). Then, to determine the stability of these steady states, a standard method is to use a Jacobian matrix (that can be found using the function `jacobian` in Matlab, Fig. 9B), which results from linearization of the nonlinear system. The linearization of the system at a steady state of interest provides a good approximation of the system dynamics in the near proximity of the steady state. Since this approximation is linear, we can infer the stability of the linearization by determining the eigenvalues of the Jacobian matrix evaluated at that steady state (Fig. 9C-D). Stability (all eigenvalues with negative real part) or instability (some eigenvalue with positive real part) directly translates into stability or instability of the original system's steady state. Marginal stability of the linearization (when one eigenvalue has zero real part with all others having negative real parts) is inconclusive for the original nonlinear system (Del Vecchio and Murray, 2014; Khalil 2014).

In Figure 8C-8D, we show the results of this workflow, the list of steady states and their eigenvalues for the 3D model in equation (16). These results indicate that the system possesses two SSS and one USS. Figures 10A and 10B implement the Matlab code used to compute these steady states and eigenvalues, which can be adapted for other systems by simply adding the relevant ODEs and parameters. In Figure 8B, we also show in 3D how the steady states are distributed in the 3D state space, by plotting the surfaces obtained by setting the time derivatives of *X*, *Y*, and *Z* in (16) equal to zero (which are the 3D equivalents of nullclines), and seeing that intersections of these surfaces occur at the same points computed by `vpasolve`.

To summarize the Jacobian matrix method of determining stability, for each steady state:

- Case #1: if all eigenvalues of the Jacobian matrix evaluated at that steady state have negative real parts, the state is a ***stable steady state*** for the original nonlinear system.
- Case #2: if any eigenvalues of the Jacobian matrix evaluated at that steady state has a positive real part, the state is an ***unstable steady state*** for the original nonlinear system.
- Case #3: if none of the real parts of eigenvalues of the Jacobian matrix evaluated at that steady state are positive, but one or more are zero, the Jacobian Linearization method is inconclusive, and other methods must be used.

In the rare Case #3, more advanced methods from nonlinear dynamical systems theory, including Lyapunov Theory (Slotine and Li, 1991) or the Center Manifold Theorem (Wiggins, 2003; Carr, 1981) can be used to investigate stability.

Basins of Attraction

Once the SSSs of a model are identified, determining the basin of attraction around each one can be an informative next step in understanding the landscape. For our running example in equation (15), Figure 11 shows how to compute this basin of attraction numerically in Matlab. As shown in the figure, the basins of attraction around each of the three SSS S^1 , S^2 , and S^3 are the blue, pink, and purple-shaded regions, respectively. These represent the sets of concentrations of X and Y starting from which the system converges autonomously to each respective SSS.

Parameter Analysis

Thus far in our study of the steady state landscape, we have taken for granted a certain set of parameters in our model for the 2D GRN, though in reality the values of parameters used can have major consequences on the location, number, and stability of steady states. As discussed in Section II, when constructing a biochemical reaction model of a GRN, care should be taken at every step to use parameters from the literature, when available, that most accurately reflect the reactions that take place in reality. However, there will often be large variability on many parameter values and some reactions may have unknown parameters. It is therefore important to assess how variations in parameters affect the steady state landscape of the GRN.

Here, we describe three different tools that can be used to this end. Specifically, for a single parameter variation at a time, parameter bifurcation analysis is the most commonly used approach (Wiggins, 2003). To evaluate the effect of all parameters changing at the same time, local parameter sensitivity analysis (Khalil, 2014; Del Vecchio et al, 2014) can be performed, which provides insight for sufficiently small changes in parameters. This tool can also be used to locally determine the parameters to which the system is more sensitive. Finally, for the effect of changing all parameters at the same time within given ranges, global sensitivity analysis can be performed (Saltelli et al, 2008), which is a computationally intensive approach. In the examples that follow, we implement our code in Matlab for the sake of introducing the concepts. However, several software tools are available, especially for parameter bifurcation analysis, including AUTO (<http://indy.cs.concordia.ca/auto/>) and Oscill8 (<http://oscill8.sourceforge.net/>).

Single-Parameter Bifurcations

One way to assess the impact of a given parameter on the steady state landscape is to hold all other parameters constant and plot the location of the steady states as that parameter is changed, with attention given to how the number and stability of steady states change, or *bifurcates* (Strogatz, 2014). Figure 12 demonstrates this method for the parameter a_{xx} in equation (15). The two plots in this figure show the steady states of the system, (X^*, Y^*) , and their stability, as this *bifurcation parameter* a_{xx} is varied from the values 2 to 2.8, as an example. As shown, the system begins with 5 steady states at $a_{xx} = 2$ (the nominal value used in Figures 7 and 11), and loses two steady states at about $a_{xx} = 2.45$. Stability analysis using the Jacobian matrix method (implemented in the code block of the figure) shows that of the remaining 3 steady states, two

are stable while one is unstable. In this case, we say that the system goes through a bifurcation at $a_{xx} \approx 2.5$. In particular, for the intermediate stable and unstable steady states, this is a case where the stable and unstable steady states "collide" and annihilate each other, which is known as a saddle-node bifurcation (Wiggins, 2003). This is an important insight since, as we have described, SSSs represent phenotypes and so this model would indicate that at least one phenotype could potentially disappear if this parameter is somehow perturbed experimentally. These notions will become important when we discuss how some external perturbations to ODEs can be modeled by changes in parameters in Section V, which will lend insight into possible explanations for how external stimulation during reprogramming changes the steady state landscape.

Local Sensitivity Analysis

A more systematic way of assessing how the steady state landscape changes in response to small parameter perturbations is the use of sensitivity matrices. As depicted, in Figure 13A, the sensitivity matrix, $S_{\bar{X},\bar{\theta}}$, of a system with state $\bar{X} = (X_1, X_2, \dots, X_n)$ and parameters $\bar{\theta} = (\theta_1, \theta_2, \dots, \theta_m)$ at a stable steady state \bar{X}^* and nominal parameter values $\bar{\theta}_o$ represents the local change in the stable steady state coordinates when small changes are applied to the parameters. The theory behind this method is described in detail in Section 3.2 of (Del Vecchio and Murray, 2014), though the general result can be used directly if an ODE model for a GRN is given. For a general n dimensional system with ODEs described by $\dot{X}_1 = f_1(\bar{X}, \bar{\theta}), \dots, \dot{X}_n = f_n(\bar{X}, \bar{\theta})$, the sensitivity matrix within the vicinity of steady state \bar{X}^* and around a nominal set of parameters $\bar{\theta}_o$ is:

$$S_{\bar{X},\bar{\theta}} := \frac{d\bar{X}}{d\bar{\theta}} = - \left(\frac{\partial \bar{f}}{\partial \bar{X}} \Big|_{(\bar{X}^*, \bar{\theta}_o)} \right)^{-1} \frac{\partial \bar{f}}{\partial \bar{\theta}} \Big|_{(\bar{X}^*, \bar{\theta}_o)} \quad (17)$$

where $\bar{f} = [f_1, \dots, f_n]'$.

In the normalized version of this sensitivity matrix,

$$S_{\bar{X},\bar{\theta}}^* = (D^{\bar{X}^*})^{-1} \cdot S_{\bar{X},\bar{\theta}} \cdot D^{\bar{\theta}_o}$$

where $(D^{\bar{X}^*})^{-1} = \text{diag}\{\bar{X}^*\}$ and $D^{\bar{\theta}_o} = \text{diag}\{\bar{\theta}_o\}$, the entries of this matrix can be compared to each other to assess the parameters to which the system's steady state is most sensitive. In Figure 13B, a Matlab code block is provided that carries out this computation for our 2D system in equation (15). As seen in the figure, this matrix can be computed at different SSS (as specified by the parameter 'state_index'). For the nominal parameters used, ODE model (15) has three SSS (recall Figure 7). Using this code, the normalized sensitivity matrix at the SSS S^2 from Figure 7, for instance, is:

	α_{ox}	α_{oy}	a_{xx}	a_{yx}	b_x	a_{xy}	a_{yy}	b_y	γ_x	γ_y
X^*	0.0033	0.0273	2.1682	0.1437	0.3498	0.0728	0.7706	17.7313	0.1175	0.3814
Y^*	0.0004	0.0070	0.2290	0.0152	0.0370	0.0174	0.1844	4.5339	0.0300	0.0975

A quick glance at this matrix shows that both the X and Y components of S^2 are most sensitive to the parameter b_y compared to the rest of the parameters. Note that the sensitivity matrix can be computed only at steady state/parameter combinations where the Jacobian matrix $d\bar{f}/d\bar{X}$ is non-singular, and therefore cannot be computed at bifurcation points.

Global Sensitivity Analysis

Global sensitivity analysis can offer a wider glimpse into how parameters affect the salient properties of a system. In this type of analysis, Monte Carlo methods (Saltelli et al., 2008) can be used to randomly sample several parameters at a time over a sufficiently large number of repetitions. In doing so, any output that one may be interested in as these parameters are varied, such as the number of SSS, their locations, or the dominant eigenvalues associated with them, is observed in the aggregate across a statistically representative set of outputs.

The crucial part of this process is running a sufficient number of repetitions so that the parameter sets used are representative of the entire parameter space such that, in turn, the outputs realized are statistically representative of the output space. As the number of parameters increase, running a sufficient number of repetitions becomes more challenging due to the curse of dimensionality (Saltelli et al., 2008).

There are various computational techniques that can be used to control the sample generation process of Monte Carlo methods so that a statistically representative parameter set is formed without an intractable number of samples needed (McKay et al, 1979). Latin hypercube sampling (LHS) is one such method that partitions the parameter space into a grid and enforces sampling rules such that every sample resides in a unique row and column (Stein, 1987). Figure 14 depicts this with a simple example in 2D parameter space. A Latin hypercube is shown with five samples in the parameter space a_{xy} and a_{xx} . Without LHS (Fig. 14A), the 2D parameter space might not efficiently cover the space, while with LHS (Fig. 14B), there is some level of spread enforced by the fact that every sample is the only one in the row and column of the grid formed by partitioning the parameter space.

In Figure 14C, we provide Matlab code that generates the number of steady states in the model of equation (15) as the six parameters a_{xx} , a_{xy} , b_{xy} , a_{yy} , a_{yx} , b_y are collectively randomly sampled. The code uses the function `lhsnorm` to generate a 6D normal Latin hypercube, where each parameter is sampled from a normal distribution with the means at nominal values from Figure 7 and variances as specified in Figure 14. For $N = 1000$ samples, we see that our model has 3 total steady states around 70% of the time, 5 total steady states about 18% of the time, and 1 total steady state about 12% of the time.

We note that the distributions from which the parameters are being sampled can have a significant difference on the results of a global sensitivity analysis. If sampling from normal distributions, increasing variances of these distributions would include more extreme values away from the nominal means. In addition, the Matlab function `lhsdesign` offers LHS from uniform distributions, which may be used if those are considered better representative of the parameters in a model. Ultimately, these are decisions that one has to make in accordance with the degree of global sensitivity one is interested in capturing, and the claims made about global

sensitivity should be explained in the context of assumptions made on the parameter distributions used.

In summary, the collection of tools we have provided a tutorial for in this section can be used towards understanding the properties of arbitrary GRNs that abide by the assumptions at the end of Section II. Using the Matlab and Mathematica implementations of these various tools for our running GRN example, practitioners can adapt the code we provide according to the architecture of their networks.

V. Modeling Cell Fate Reprogramming: Response of GRNs to Artificial Perturbations

The dynamical systems tools discussed in Section IV were introduced as a means of garnering insight into the intrinsic properties of GRNs. In this section, we use these tools to analyze the response of GRNs to *artificial perturbations* that can capture the manipulations that experimentalists exert during cell fate reprogramming.

In the overexpression reactions of Table 1F for our example GRN in Figure 2, we modeled these artificial perturbations as constant production terms, u_x and u_y , on the mRNA species m_x and m_y , respectively. Following reduction of our model to two dimensions in equation 15, the protein species X and Y display constant production terms \bar{u}_x and \bar{u}_y (proportional to the original overexpression terms), respectively, in their dynamics. When modeling TF-mediated cell fate reprogramming, these constant overexpression terms are responsible for the change in the steady state landscape in such a way as to induce the desired transition between two SSSs. However, with nonlinear systems such as ours, it is in general non-trivial to determine what types and levels of artificial perturbation can trigger a desired transition.

To begin elucidating the ways in which artificial perturbations might be applied to trigger a certain transition, we can observe the qualitative change in the steady state landscape as we change parameters that correspond to potential artificial perturbations, assuming these stay approximately constant once applied. We will consider ectopic overexpression and enhanced degradation of TFs as the possible artificial perturbations to our model in (15).

Open-Loop Overexpression

In Figure 15, we use the nullclines and bifurcation tools discussed in Section IV to observe the response of the steady state landscape of the system to increasing values of \bar{u}_x and \bar{u}_y . In Figure 15A, we see that the landscape again begins with three SSS: S^1 , S^2 , S^3 . From a control design point of view (Astrom and Murray, 2016), we can regard prefixed overexpression of TFs as an "open loop" control strategy, wherein the input of the system (overexpression level) is prefixed at the beginning of the experiment and is not adjusted based on the TF levels throughout the experiment (Figure 15B). In panels C and D, we note the form in which the nullclines change as a result of increasing \bar{u}_x and \bar{u}_y , respectively. And though it is difficult, given the nonlinearity of the system, to pinpoint systematically how the nullclines will change shape or move across the X - Y plane, we can qualitatively see that increasing \bar{u}_x (perturbation #1) causes S^1 to disappear first followed by S^2 while increasing \bar{u}_y (perturbation #2) causes S^2 to disappear first, followed by S^1 (Figure 15C-F). In either case, sufficiently high overexpression will leave only one SSS,

$S^{3'}$, in the proximity of S^3 (i.e. in the basin of attraction of S^3 ; see bifurcation plots in figure Fig. 15E-F for \bar{u}_x and \bar{u}_y very high). This is important insight for the reprogramming strategy. Specifically, if the starting state (phenotype) is S^1 and our objective is to trigger a transition into S^2 , overexpression of \bar{u}_x may be a preferable approach to overexpression of \bar{u}_y .

In panels G-H of Figure 15, we simulate a reprogramming experiment using open-loop overexpression with an intermediate level of \bar{u}_x to facilitate a transition from S^1 to S^2 . Starting with the system at S^1 (Fig. 15G), we apply $\bar{u}_x = 0.2$ (a level informed by the bifurcation plots of Fig. 15E) and see that the nullclines and vector field change in a fashion that forces the state to transition to $S^{2'}$. Upon removal of this input in Fig. 15H, we see that the system transitions from $S^{2'}$ to S^2 , and hence the reprogramming from S^1 to S^2 is complete. We note that even though overexpression was possible with this intermediate level of \bar{u}_x , very high overexpression of either TF will not be a successful strategy to triggering transitions from S^1 to S^2 as the system's state will ultimately approach S^3 instead (see bifurcation plots in Fig. 15E-F). At best with \bar{u}_x there is a finite window of overexpression that is required for triggering the desired transition and it may not be experimentally trivial to set the overexpression level precisely in that window. So, overexpression may in practice fail for this example GRN.

Cooperative Monotone Systems: Fundamental Limitations on Re-programmability

Our model makes some potentially useful predictions about possible reprogramming strategies using open-loop overexpression. In particular, it indicates that there may be a fundamental flaw in these types of perturbations for certain transitions, such as the one from S^1 to S^2 . There is a fundamental reason why certain transitions from states characterized by lower concentrations (S^1) to states characterized by higher, but not maximal, concentrations (S^2) cannot be guaranteed with the open-loop overexpression described in Figure 15. It is because our example GRN of Figure 2 only has activating arrows between the TFs. At an intuitive level, this means that both genes are always upregulating themselves as well as each other, so stimulating these nodes artificially (via open-loop ectopic overexpression) very easily sets off an upregulation positive-feedback cascade that pushes concentrations of both to higher and higher levels so that the system becomes monostable at maximal (i.e. non-intermediate) values. In fact, reprogramming GRNs to intermediate states is a difficult task to achieve in an even more general class of GRNs that can be described as ‘cooperative¹ monotone’ systems (Del Vecchio et al, 2017), of which our example is a type. Hence, it is not possible using these reprogramming paradigms to guarantee that reprogramming from states with lower concentrations to states with intermediate concentrations (such as the transition from S^1 to S^2) will be successful.

Enhanced Degradation

Another way to experimentally perturb our GRN is by adding proteases or microRNA (Agrawal, 2003; to artificially enhance the degradation rate of TFs X and Y. In our model in (15), this is akin to increasing the values of the overall decay rates γ_x and γ_y , respectively. In Figure 16, we repeat the analysis from above, this time analyzing the effect of increasing these decay rates from their nominal values in Figure 7. As demonstrated by the changing nullclines and bifurcation

¹ Cooperative networks should not be confused with cooperative co-binding modes discussed in Section II.

plots in panels 16C-H, we see that enhancing the degradation rate of γ_x (perturbation #3) causes the SSS S^2 to disappear first, followed by S^3 . Likewise, enhancing the degradation rate of γ_y (perturbation #4) causes the SSS S^3 to disappear first, and maintains the SSSs in the proximity of S^1 and S^2 .

Based on the way increasing degradation rates changes the steady state landscape, another perturbation we might consider for use in transitioning from S^1 to S^2 is appropriately combining overexpression with enhanced degradation (perturbations 1 and 4) so that both S^1 and S^3 disappear while keeping a state $S^{2'}$ in the vicinity of S^2 . Figure 17 shows the results of simulating this experiment, in which we use the same level of intermediate overexpression as in the experiment above, in addition to increasing the degradation on TF Y enough to make S^3 disappear (informed by the bifurcation plots of Fig. 16F). This causes the system to transition to a state $S^{2'}$ near S^2 (Fig. 17C), so that upon cessation of both artificial perturbations (Fig. 17D), the system transitions to S^2 , and reprogramming is complete.

Closed-Loop Overexpression

The crucial conclusion from these theoretical results is that open-loop overexpression, whether or not it is combined with enhanced degradation, cannot be guaranteed to be a winning strategy for certain reprogramming tasks. To address this, in previous work (Del Vecchio et al., 2017) we have proposed an entirely new paradigm called closed-loop, or feedback overexpression (Figure 18A), in which the level of ectopic overexpression is not merely set once at the beginning of an experiment. Rather, if TF X in our system needs to be steered to a target concentration X^* (where X^* may be the concentration seen in a target phenotype in a reprogramming experiment) the actual concentration is measured throughout the experiment and the level of overexpression or degradation \bar{u}_x is altered in proportion to the distance between the current state and target state, i.e. $\bar{u}_x = G(X^* - X)$. Our theory shows that with high enough values of G , known as the *gain* of the feedback controller, any starting state can in principle be steered to arbitrary levels X^* of concentrations for each node in the network that is overexpressed via feedback overexpression. This is true of any system regardless of the initial shape of the nullclines, which demonstrates that this strategy is not dependent upon the specific dynamics or parameters of a GRN.

In Figure 18B, we demonstrate what this means for the nullclines for our model in (15). As the figure shows, as the values of gain G are increased, the nullclines morph into increasingly straight lines that intersect at the target state (X^*, Y^*) . In panels C-D of Figure 18, we simulate a reprogramming experiment in our running example from S^1 to S^2 via closed-loop overexpression.

Realizing Genetic Feedback Control

In the feedback overexpression paradigm we have presented, which contrasts with the status quo of ‘open-loop’ overexpression paradigms used in most TF-mediated reprogramming experiments, transitions to intermediate states can in fact be theoretically guaranteed (demonstrated in the theory shown in (Del Vecchio et al, 2017)). In addition, we have shown that one way to realize this closed-loop overexpression without requiring continuous measurements of concentration, (which may be difficult experimentally) is by use of a synthetic genetic feedback controller that simultaneously ectopically overexpresses and degrades the mRNA of a

species under control using short-interfering RNA (siRNA) (Figure 19). By the appropriate balancing of these two artificial perturbations using inducers that control the expression levels of mRNA using a synthetic gene and the expression levels of the siRNA, the system can be in principle steered to arbitrary concentrations independent of network dynamics and parameters. Moreover, we have shown in detail in (Del Vecchio et al, 2017) that the high gain G needed according to the theory can be achieved by increasing the copy number of the ectopic DNA on which the synthetic circuit is encoded. At an intuitive level, this is tantamount to having a sufficient amount of the synthetic circuit inside the cell being reprogrammed that it dominates the effect of endogenous dynamics and takes over the network to steer it to arbitrary concentration levels.

VI. Conclusion

The topics we have discussed in this chapter comprise the basic steps in the computational approach to cell fate reprogramming. At the root of the approach we have described is the existence of a GRN that has been implicated in the cell fate change of interest. Moreover, in order to obtain an ODE model, we have made certain kinetic assumptions about how the species of that GRN interact. By way of a running example of a GRN with two TFs mutually and auto-activating, we have gone through a series of tutorials on methods from dynamical systems theory that can be used to probe the intrinsic properties of GRNs, and shown their implementations using Mathematica and Matlab. These include methods that identify the number and location of stable steady states of a GRN (corresponding to phenotypes) as well as various ways to understand the model's dependence on parameters. The code blocks we have provided can in many cases straightforwardly be adapted to suit practitioners' ODE models corresponding to GRNs with various topologies. We then provided a discussion for how to use these tools to understand GRNs' responses to artificial perturbations, which model the stimulations that experimenters apply to a GRN during cell fate reprogramming. Ultimately, this is the purpose of these models: to understand the response of a GRN *in silico* to various types of artificial perturbations so that one can inform *in vitro* and *in vivo* experiments (see Sections III-V).

Throughout this chapter, the differential equation models we have used were deterministic. As noted in Section II, this is because ODE models are an appropriate starting point for a first pass understanding of the qualitative behavior of GRNs. However, they are only appropriate to describe the mean behavior of species' dynamics when molecular counts are high enough to be stated in terms of concentrations. The next step after understanding GRNs in the deterministic realm is typically to use stochastic differential equation (SDE) models, such as the Chemical Langevin Equation (CLE), which can capture the effects of the prevalent intrinsic and extrinsic noise in biological systems (Swain and Elowitz, 2002; Elowitz et al., 2002). The Chemical Master Equation (CME) is most appropriate when molecular counts are low while the Chemical Langevin Equation (CLE) is appropriate when concentration is still a meaningful description of molecular counts. When molecular counts are low, practitioners may find that the GRN behaviors observed in an ODE model are dramatically different from the mean behavior of a stochastic model (Al-Radhawi et al., 2017).

Another dimension that we have not considered in this chapter is the epigenetic transformation that might often take place during a cell fate reprogramming process. In the past decade, it has

become increasingly clear that the unpacking of genes from their condensed chromatin states, as well as methylome reprogramming of DNA base pairs on the promoters of GRNs are key events (Allis et al., 2015; Bagci and Fisher, 2013; De Carvalho, 2010; Huang and Fan, 2017) that take place during what has typically been assumed to be genetic reprogramming alone. In general, the degree of epigenetic transformation that takes place during a given cell fate reprogramming process should be investigated and potentially included in mathematical models of reprogramming.

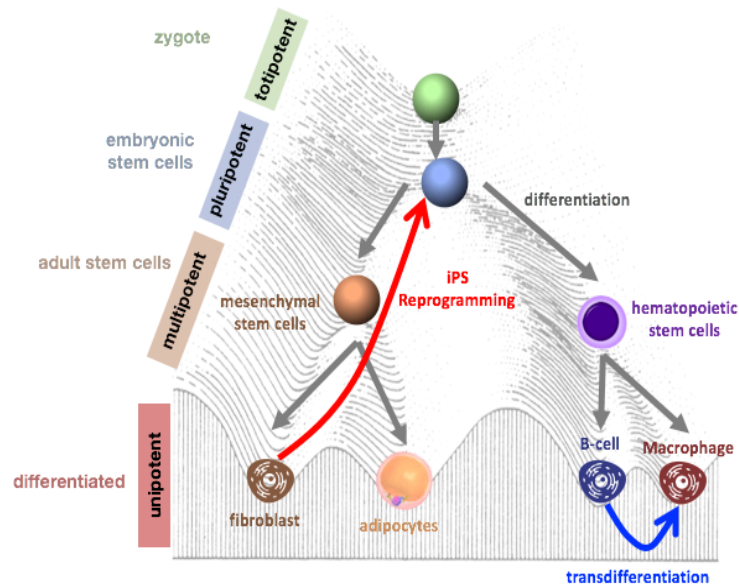


Figure 1: Waddington Landscape of Cell Differentiation. In this metaphor for differentiation, cell fate specification is akin to a marble rolling down a hill with several different valleys that represent the diverse fates that cells with the same genetic encoding ultimately adopt. Along the way to differentiation, these cells are often lumped under the umbrella term of ‘stem cells,’ though they have different potencies depending on their degree of differentiation. Zygotes and embryonic stem cells (ESCs) in the first few divisions after fertilization are *totipotent*, and can give rise to an entire organism including the placenta and umbilical cord. When ESCs become *pluripotent*, they can continue to give rise to all three germ layers and thus an entire organism. *Multipotent* cells are adult stem cells that retain some potential for further differentiation, which is typically limited to a particular tissue type (e.g. haematopoietic cells, neural stem cells, mesenchymal stem cells) (Mitalipov and Wolf, 2009). iPS reprogramming as depicted is tantamount to an ‘uphill’ movement from a somatic state (e.g. fibroblast) to the pluripotent state. Transdifferentiation as depicted is tantamount to directly transforming from one lineage to another (e.g. B-cell to macrophage, as shown in Xie et. al, 2004).

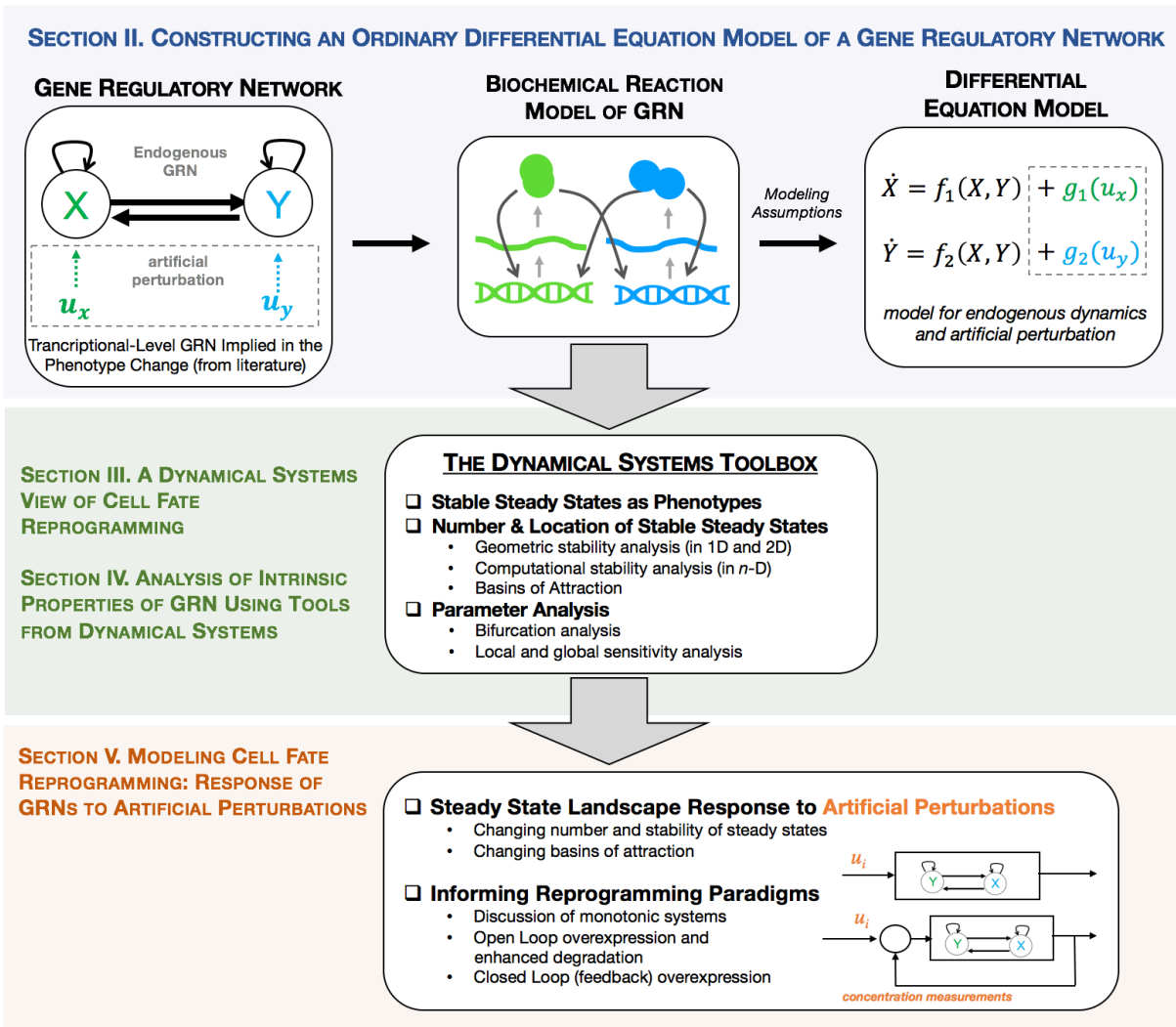
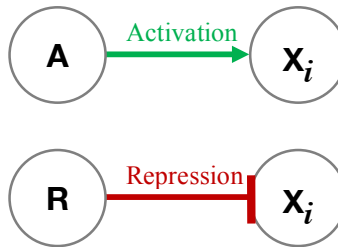


Figure 2: The Dynamical Systems Approach to Cell Fate Reprogramming (top panel) If a known GRN is implied during a phenotype change in a cell fate transformation experiment, a biochemical reaction model of the network can be constructed. Under certain assumptions about this biochemical reaction model that are detailed in Section II, ODE models can be constructed that describe the endogenous dynamics of the system and also incorporate the effect of artificial perturbations. (middle panel) Once an ODE model is constructed, the dynamical systems toolbox can be applied towards the goal of understanding the intrinsic properties of the GRN in question. Notably, these tools seek to understand the number and location of stable steady states implied by the mathematical model, including an assessment of how these depend on the model's parameters. (bottom panel) The response of the GRN to artificial perturbations is the ultimate goal of formulating the reprogramming problem into these mathematical terms. By understanding how the number, location, and extent of basins of attraction surrounding the stable steady states determined in step (II) will change in response to artificial perturbations such as ectopic overexpression, practitioners can understand the advantages and limitations of different reprogramming paradigms, which may be used to inform experiments.



**Figure 3: Representing
Activation and Repression
Between Nodes in a GRN**

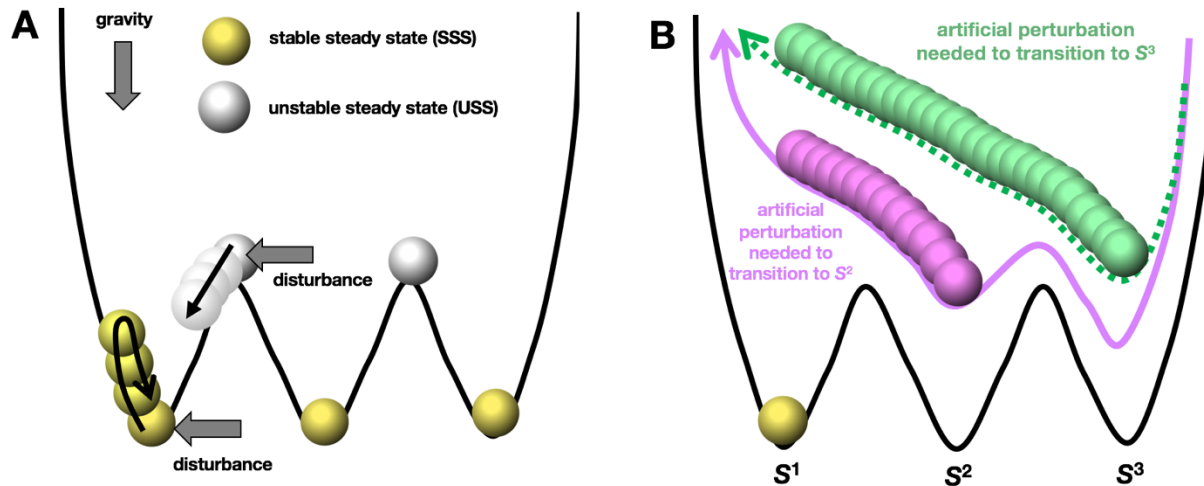


Figure 4: Visual Depiction of Stable and Unstable Steady States. (A) Unstable steady states (USS) are analogous to the 'peaks' in a valley landscape while stable steady states (SSS) are analogous to well-like depressions in the landscape. If a ball is resting in a SSS, it can withstand some degree of disturbance and still return to the SSS autonomously. However, if it is resting on a peak (USS), then any nonzero disturbance will permanently push it off that peak and hence out of that steady state. (B) Cell fate reprogramming is akin to transitioning between wells by altering the landscape through artificial perturbation. If a ball begins at state S^1 , it can be forced to transition to S^2 or S^3 by appropriately changing the valley landscape as shown.

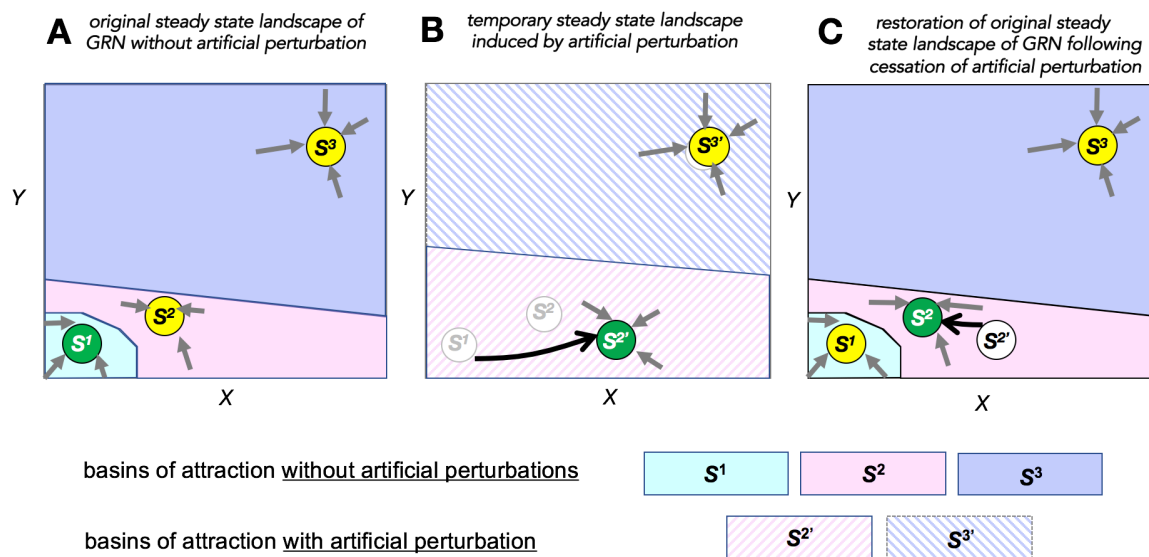


Figure 5: Cell Fate Reprogramming as Morphing the Steady State Landscape (A) Without any artificial perturbation, the system possesses three stable steady states (SSS): S^1 , S^2 , and S^3 that represent phenotypes characterized by relative concentration levels of the TFs X and Y. The basin of attraction of each SSS (phenotype) is the set of states (TF concentrations) starting from which the system converges autonomously to the corresponding SSS (phenotype), as depicted by the shaded regions. (B) When a certain degree of artificial perturbation to the network is applied, the steady state landscape changes shape so as to push the system starting from S^1 to a location $S^{2'}$ in the basin of attraction of the original state S^2 of the system without overexpression. (C) Upon removal of the artificial perturbation, the system's state moves from $S^{2'}$ to S^2 , and the transition from S^1 to S^2 has been completed.

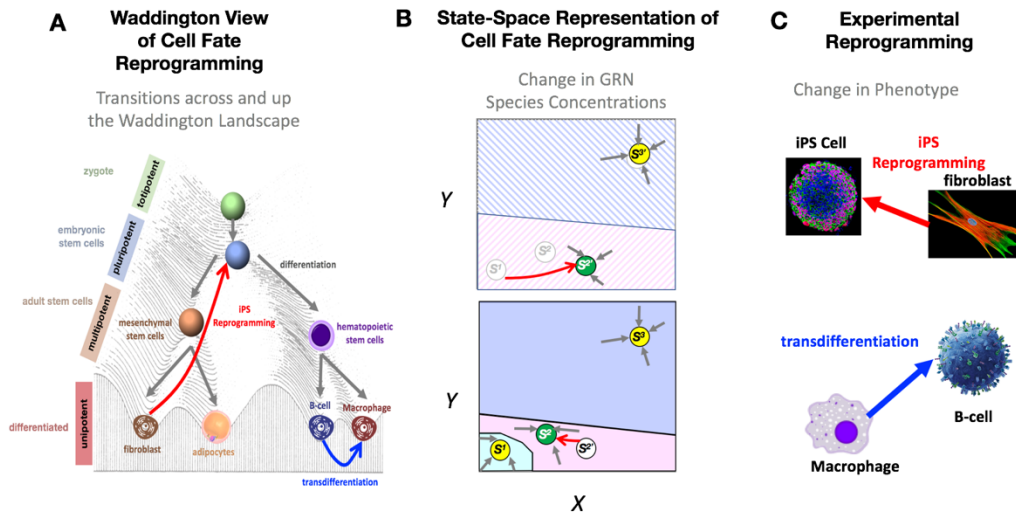


Figure 6: The Reprogramming Concept Viewed at Different Levels of Abstraction_(A) At a high-level, cell fate reprogramming is akin to traveling ‘uphill’ in the Waddington landscape. (B) Identification of a GRN motif controlling the relevant cell fates allows for the framing of reprogramming as enforcing a transition between stable steady states in the ‘state-space’ of an ODE model of that GRN. These stable steady states are characterized by relative concentration levels of TFs in the GRN that in principle map to relative concentration levels seen in the phenotypes being represented. (C) Ultimately, the dynamic model of reprogramming is meant to inform improved experimental cell fate reprogramming strategies, which transform one phenotype to another.

2D Geometric Analysis

ODEs

$$\frac{d}{dt}(X) = \frac{\alpha_{x0} + a_{xx}X^2 + a_{yx}Y^2 + b_xX^2Y^2}{1 + X^2 + Y^2 + X^2Y^2} - \gamma_x X + \bar{u}_x$$

$$\frac{d}{dt}(Y) = \frac{\alpha_{y0} + a_{xy}X^2 + a_{yy}Y^2 + b_yX^2Y^2}{1 + X^2 + Y^2 + X^2Y^2} - \gamma_y Y + \bar{u}_y$$

$$\begin{aligned} dX/dt &= 0 \\ dY/dt &= 0 \end{aligned}$$



Nullclines

$$Y = \frac{\sqrt{-\bar{u}_x - X^2(\bar{u}_x + a_{xx}) - \alpha_{x0} + \gamma_x(X + X^3)}}{\sqrt{\bar{u}_x + (\bar{u}_x + b_x)X^2 + a_{yx} - \gamma_x(X + X^3)}}$$

$$X = \frac{\sqrt{-\bar{u}_y - Y^2(\bar{u}_y + a_{yy}) - \alpha_{y0} + \gamma_y(Y + Y^3)}}{\sqrt{\bar{u}_y + (\bar{u}_y + b_y)Y^2 + a_{xy} - \gamma_y(Y + Y^3)}}$$

Mathematica Code Block

```

1  Solve[(alphax0 + axx*X^2 + ayx*Y^2 +
2  bx*(X^2)*(Y^2))/(1 + X^2 + Y^2 + X^2*Y^2) -
3  gammax*X + uxbar == 0, Y]
4  Solve[(alphay0 + axy*X^2 + ayy*Y^2 +
5  by*(X^2)*(Y^2))/(1 + X^2 + Y^2 + X^2*Y^2) -
6  gammay*Y + uybar == 0, X]
7
8  alphax0 = 0.01; alphay0 = 0.01; axx = 2; ayx = 10;
9  axy = 0.17; ayy = 1.8;
10  bx = 7.5; by = 2; gammax = 1; gammay = 1; uxbar =
11  0.00; uybar = 0.00;
12
13  dXdt = (alphax0 + ayx*Y^2 + axx*X^2 +
14  bx*X^2*Y^2)/(1 + X^2 + Y^2 + X^2*Y^2) - gammax*X
15  + uxbar;
16
17  dYdt = (alphay0 + ayy*Y^2 + axy*X^2 +
18  by*X^2*Y^2)/(1 + X^2 + Y^2 + X^2*Y^2) - gammay*Y
19  + uybar;
20
21  nullclines = ContourPlot[{dYdt == 0, dXdt == 0},
22  {X, 0, 7}, {Y, 0, 1.5}];
23  vectorField = VectorPlot[{dXdt, dYdt}, {X, 0, 7},
24  {Y, 0, 1.5}, VectorStyle -> Arrowheads[0.03]];
25
26  Show[nullclines, vectorField]
```

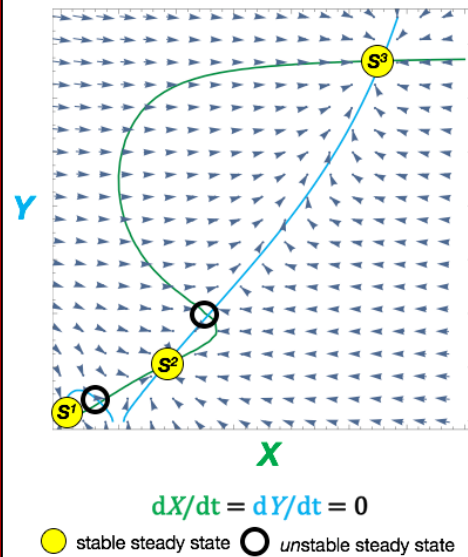


Figure 7: Steady states of a 2D ODE and Their Stability (top) One way to find the steady states of our ODE model is to set the time derivatives of each species to zero and solve for the curves in the plane that result. (bottom left) [lines 1-6] The `Solve` function in Mathematica can be used to solve for nullclines if there is an explicit solution. These could then be plotted using the `Plot` function in Mathematica (not shown). [lines 8-11] Parameter value definitions. [lines 13-24] The `ContourPlot` function in Mathematica can also be used to plot nullclines, whether or not there is an explicit solution, as the two ODEs are specified as `{dX/dt == 0, dY/dt == 0}` in the input. The `VectorPlot` function, can be used to plot the vector field of the system. The two ODEs are specified as `{dX/dt, dY/dt}` in the input. [line 26] The `Show` function can be used to overlay the nullclines and vector field.

Note that the plot shown includes styling not specified in the code shown. The optional parameters that can be used to format plots can be found in the documentation of the `ContourPlot` and `VectorPlot` functions.

Documentation for `Solve`: <http://reference.wolfram.com/language/ref/Solve.html>

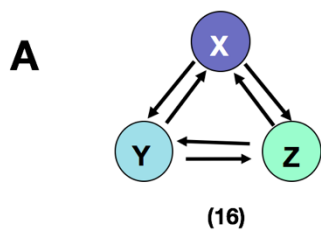
Documentation for `ContourPlot` in Mathematica: <http://reference.wolfram.com/language/ref/ContourPlot.html>

Documentation for `VectorPlot` in Mathematica: <http://reference.wolfram.com/language/ref/VectorPlot.html>

Note that overlaid nullcline and vector plot could also be generated using the `plot` and `quiver` functions, respectively, in Matlab.

Documentation for `plot` in Matlab: <https://www.mathworks.com/help/matlab/ref/plot.html>

Documentation for `quiver` in Matlab: <https://www.mathworks.com/help/matlab/ref/quiver.html>



$$\frac{d}{dt}X = \frac{\alpha_{x0} + a_{zx}Z^2 + a_{yx}Y^2}{1 + Y^2 + Z^2} - \gamma_x X = H_1(Y, Z) - \gamma_x X$$

$$\frac{d}{dt}Y = \frac{\alpha_{y0} + a_{xz}X^2 + a_{zy}Z^2}{1 + X^2 + Y^2} - \gamma_y Y = H_2(X, Z) - \gamma_y Y$$

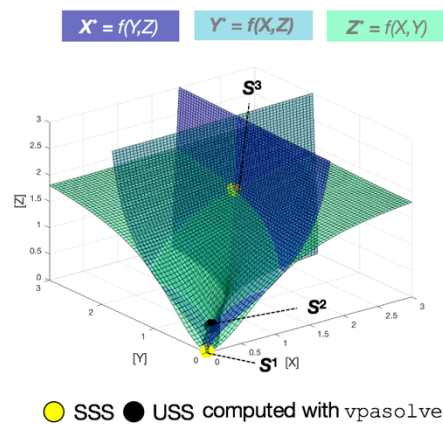
$$\frac{d}{dt}Z = \frac{\alpha_{z0} + a_{xz}X^2 + a_{yz}Y^2}{1 + X^2 + Y^2} - \gamma_z Z = H_3(X, Y) - \gamma_z Z$$

C

```
steady_states =
    0.2760    0.2760    0.2760 = (x1*, y1*, z1*) = S2
    1.7099    1.7099    1.7099 = (x2*, y2*, z2*) = S3
    0.0141    0.0141    0.0141 = (x3*, y3*, z3*) = S1
```

output of code in Figure 10A

B



D

Lambda =			USS
-1.3694	-1.3694	0.4888	= $\lambda_{11}, \lambda_{12}, \lambda_{31}$
-0.8587	-0.8587	-0.5326	= $\lambda_{21}, \lambda_{22}, \lambda_{23}$
-0.7921	-0.7921	-0.6659	= $\lambda_{31}, \lambda_{32}, \lambda_{33}$

SSS

output of code in Figure 10B

Figure 8: Steady State Landscape Analysis of a 3D GRN: (A) The ODEs in (16) represent the reduced system that would be obtained by applying the workflow and assumptions of Section II on the GRN shown, with three mutually activating TFs: X, Y, Z (B) The surfaces X^*, Y^*, Z^* are obtained by setting the time derivatives of X, Y, Z , respectively, equal to zero and solving for each variable. Intersections of all three surfaces represent the steady states of the system, of which there are three as shown. (C) These same three steady states can be equivalently computed using `vpasolve` in Matlab, as implemented in Figure 10A. (D) Using Jacobian matrices as explained in Figure 9 and implemented in Figure 10B, the stability of each steady state can be determined using the eigenvalues of the Jacobian matrix evaluated at that steady state.

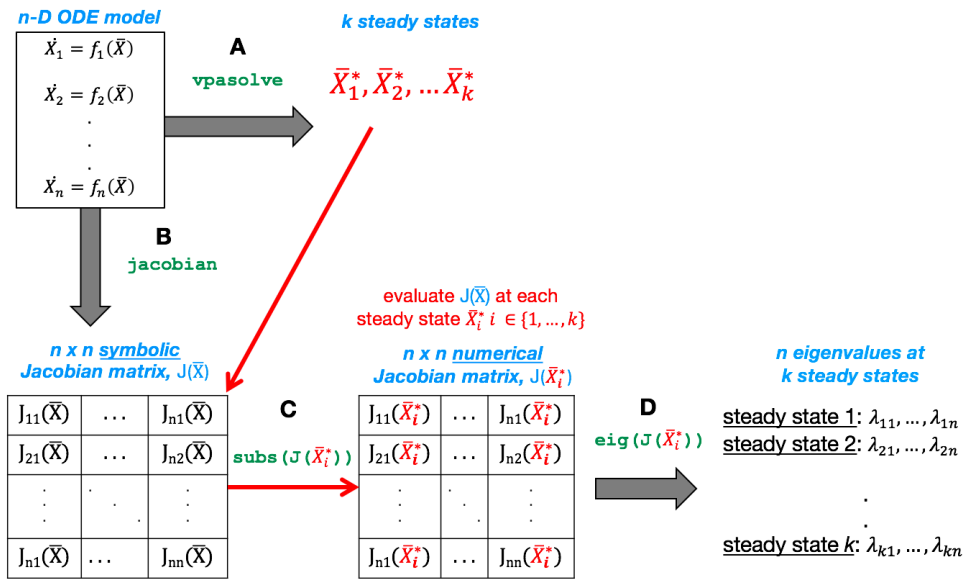


Figure 9: Using Jacobian Matrices to Determine Stability of Steady States in n-D Models. (A) The steady states of an n -D ODE model can be computed using the `vpasolve` function in Matlab. (B) The Jacobian matrix, J , of the n -D ODE model can also be computed in symbolic terms using the `jacobian` function in Matlab. (C)-(D) The outputs of results from (A) and (B) can be combined to evaluate the symbolic J at each of the k steady states using the symbolic substitution function `subs`, and the n eigenvalues of each Jacobian matrix can be calculated using the function `eig` (note: some eigenvalues may be repeated).

n-D Computational Analysis - Find All Steady States - surf, vpasolve

```
1 colormap winter % set coloring for 3D plots
2 a = 1.5; ayx = a; azx = a; azy = a; axy = a; axz = a; ayz = a; % set parameters
3 g = .75; gammaX = g; gammaY = g; gammaZ = g; % decay rates of proteins X,Y,Z
4 alpha0x=0.01;alpha0y=0.01;alpha0z=0.01;%leaky transcription rates of mX,mY,mZ
5 % plot the surface X* = f(Y,Z)
6 dp = 0.05; max = 3;
7 [Y,Z] = meshgrid(0:dp:max,0:dp:max);
8 Xnullcline = (1/gammaZ)*(alpha0x + azx*(Z.^2) + ayx*(Y.^2))./(1+ Y.^2 + Z.^2);
9 C = .25*ones(length(Y),length(Z)); % set color
10 sx = surf(Xnullcline,Y,Z,C); hold on;
11 % plot the surface Y* = f(X,Z)
12 [X,Z] = meshgrid(0:dp:max,0:dp:max);
13 Ynullcline = (1/gammaY)*(alpha0y + azx*(Z.^2) + axy*(X.^2))./(1+ X.^2 + Z.^2);
14 C = 0.55*ones(length(X),length(Z)); % set color
15 sy = surf(X,Ynullcline,Z,C); hold on;
16 % plot the surface Z* = f(X,Y)
17 [X,Y] = meshgrid(0:dp:max,0:dp:max);
18 Znullcline = (1/gammaZ)*(alpha0z + azx*(X.^2) + ayz*(Y.^2))./(1+ X.^2 + Y.^2);
19 C = 0.75*ones(length(X),length(Y));
20 sz = surf(X,Y,Znullcline,C); hold on;
21 % format plots and add axes labels
22 xlabel('X','FontSize',15);ylabel('Y','FontSize',15);zlabel('Z','FontSize',15)
23 alpha(sx,0.4); alpha(sy,0.4); alpha(sz,0.4); % apply transparency on the surfaces
24 % overlay numerically computed solutions on nullclines
25 % (Figure 9A) calculate steady states using vpasolve
26 syms X Y Z % define variables and dynamics
27 dXdt = (alpha0x + azx*(Z.^2) + ayx*(Y.^2))./(1+ Y.^2 + Z.^2) - gammaX*X;
28 dYdt = (alpha0y + azx*(Z.^2) + axy*(X.^2))./(1+ X.^2 + Z.^2) - gammaY*Y;
29 dZdt = (alpha0z + ayx*(X.^2) + azy*(Y.^2))./(1+ X.^2 + Y.^2) - gammaZ*Z;
30 [sol_X, sol_Y, sol_Z] = vpasolve([dXdt == 0, dYdt == 0, dZdt == 0],...
31 [X, Y, Z], 'random', true);
32 % extract indices of real solutions solutions
33 indices1 = find(imag(sol_Y)==0);
34 indices2 = find(imag(sol_Y)==0);
35 indices3 = find(imag(sol_Z)==0);
36 % store real solutions in matrix and save
37 steady_states=double([sol_X(indices1(3:5)),sol_Y(indices2(3:5)), sol_Z(indices3(3:5))])
38 % plot solutions in 3D space (overlay)
39 for i=1:3
40 scatter3(steady_states(i,1),steady_states(i,2),steady_states(i,3),300,'k','filled');
41 hold on;
42 end;
```

Figure 10A: Finding Steady States using surf and vpasolve in Matlab: [lines 1-23] Use surf to plot 3D surfaces using the 3D 'nullclines' obtained by solving for X,Y,Z after setting the temporal derivatives of these variables to zero in equation (16) of Figure 8A. [lines 25-38] Use vpasolve to numerically compute steady states of model in eq. 16. Note that in line 37, we use the solution indices 3-5 since the solutions corresponding to indices 1 and 2 are incorrect (verified manually). Since vpasolve is only a numeric solver, solutions should always be cross-checked by plugging them back into the ODEs. [lines 39-42] Overlay the steady states computed numerically using vpasolve on top of the intersecting surfaces. The steady states returned from this code-block, stored in the variable 'steady_states,' are shown in Figure 8C.

***n*-D: Computational Analysis - Determine Stability: jacobian, subs, eig**

```
49 % (Figure 9B) find jacobian, in symbolic terms, for 3D system defined above
50 J_sym = jacobian([dXdDt,dYdt,dZdt],[X,Y,Z]);
51 Lambda = zeros(size(steady_states)); %create matrix to store eigvals at each SS
52 for i =1:length(steady_states) %loop through steady states
53     x = steady_states(i,1);
54     y = steady_states(i,2);
55     z = steady_states(i,3);
56     %(Figure 9C) evaluate Jacobian matrix at each steady state
57     J_numerical = subs(J_sym,[X,Y,Z],[x,y,z]);
58     %(Figure 9D) compute eigenvalues of Jacobian evaluated at each steady state
59     eigVals = eig(J_numerical);
60     %(store eigenvalues at each SSS)
61     Lambda(i,1) = eigVals(1);
62     Lambda(i,2) = eigVals(2);
63     Lambda(i,3) = eigVals(3);
64 end
65
66 Lambda % display all eigenvalues
67
--
```

Figure 10B: Computationally Determining the Stability of Numerically Computed Steady States. This code block continues from Figure 10A and implements the Jacobian matrix method outlined in Figure 9. Output of this code block, a matrix 'Lambda' where each row is the set of eigenvalues of a steady state, is shown in Figure 8D.

Documentation for `jacobian` in Matlab: <https://www.mathworks.com/help/symbolic/vpasolve.html>

Documentation for `eig` in Matlab: <https://www.mathworks.com/help/symbolic/eig.html>

Documentation for `subs` in Matlab: <https://www.mathworks.com/help/symbolic/subs.htm>

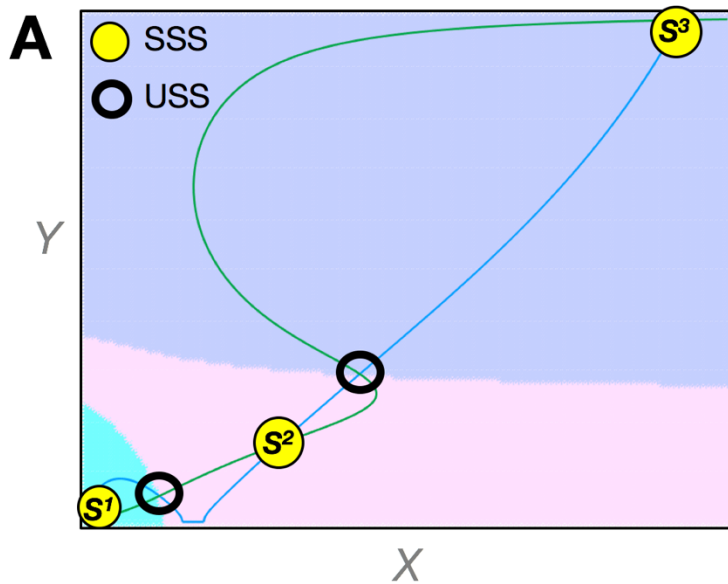


Figure 11: Basins of Attraction for 2D System: (A) In Matlab, the basins of attraction around each SSS can be found numerically by repeatedly simulating the system using one of Matlab's ODE solvers, and starting from a grid of points surrounding the SSSs (B) We implement this numerical calculation of basins of attraction using the `ode23s` solver to simulate the system starting from each point in the grid. In addition, we overlay nullclines on top of the numerically computed basins using the `plot` function in Matlab. (C), We define the ODE function 'ODE_model' passed to the solver on line (33) of panel (B) in a separate file of the same name.

Documentation for available Matlab solvers:
mathworks.com/help/matlab/math/choose-an-ode-solver.html

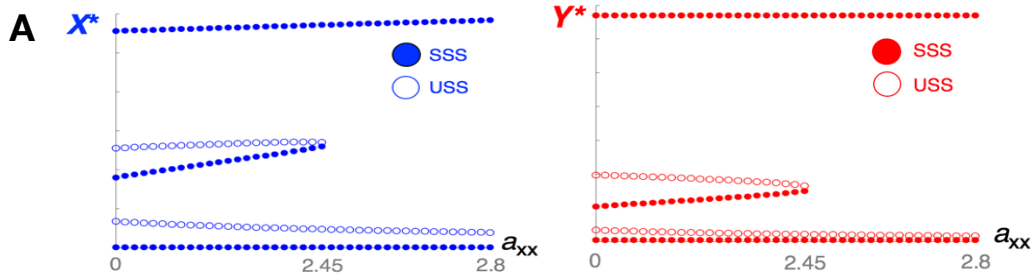
B – Compute Basin of Attraction Numerically – ode23s, plot

```
1 % Part 1: compute steady states of 2D system in equation (15) with vpasolve
2 % define variables, parameters and ODEs
3 syms X Y;
4 alpha0x = 0.01; alpha0y = 0.01; axx = 2; ayx = 10; bx = 7.5;
5 axy = 0.17; ayy = 1.8; by = 2; gammaX = 1; gammabar = 1; uxbar = 0; uybar = 0;
6 dXdT = (alpha0x + ayx*Y^2 + axx*X^2 + bx*(X^2)*(Y^2)) ...
7 /(1 + X^2 + Y^2 + (X^2)*(Y^2)) - gammaXbar*X + uxbar;
8 dYdT = (alpha0y + ayy*Y^2 + axy*X^2 + by*(X^2)*(Y^2)) ...
9 /(1 + X^2 + Y^2 + (X^2)*(Y^2)) - gammaYbar*Y + uybar;
10 % find numerical solutions and extract real ones
11 [sol_X, sol_Y] = vpasolve([dXdT == 0, dYdT == 0],[X, Y]);
12 % find real solutions and put in matrix 'steady_states'
13 sol_X = sort(sol_X(find(imag(sol_X)==0))); sol_Y =
14 sort(sol_Y(find(imag(sol_Y)==0)));
15 steady_states = [sol_X, sol_Y]
16 x1 = steady_states (1,1); y1 = steady_states (1,2); %coordinates of S1
17 x2 = steady_states (3,1); y2 = steady_states (3,2); %coordinates of S2
18 x3 = steady_states (5,1); y3 = steady_states (5,2); %coordinates of S3
19 % Part 2: Compute basins of attraction around each SSS
20 %set colors for each SSS's basin of attraction
21 S3_col = [198 208 255]/256; S2_col = [255 224 255]/256; S1_col = [115 254 255]/256;
22 %set range of values to simulate system from (extent of basin of attraction)
23 maxX = 6; maxY = 1.35; deltax = 0.05; deltay = 0.01;
24 Xrange = [0:deltax:maxX]; Yrange = [0:deltay:maxY];
25 for i=1:length(Xrange) % loop through X values
26     x_start = Xrange(i);
27     for j = 1:length(Yrange) % loop through Y values
28         y_start = Yrange(j);
29         initial_states = [double(x_start), double(y_start)];
30         t_end = 100; % length of simulation
31         p = [alpha0x, alpha0y, axx, ayx, bx, axy, ayy, ...
32             by, gammaXbar, gammaYbar, uxbar, uybar]; %put params in vector
33         % run system using solver (ode23s)
34         [t,out]=ode23s(@(t,out) ODE_model(t,out,p), [0 t_end], initial_states);
35         x_trajectory = out(:,1); xsim = x_trajectory(end);
36         y_trajectory = out(:,2); ysim = y_trajectory(end);
37         thres = 0.05; % threshold for comparison
38         % compare final state to all basins and mark I.C. according to SSS
39         if abs((xsim - x1)) < thres & abs((ysim - y1)) < thres
40             plot(x_start,y_start,'*', 'Color',S1_col); hold on;
41         elseif abs((xsim - x2)) < thres & abs((ysim - y2)) < thres
42             plot(x_start,y_start,'*', 'Color',S2_col); hold on;
43         elseif abs((xsim - x3)) < thres & abs((ysim - y3)) < thres
44             plot(x_start,y_start,'*', 'Color',S3_col); hold on;
45         else
46             display('none of these');
47         end
48     end
49 end
50 % overlay nullclines on top of basin of attractions
51 X = [0:0.005:maxX];
52 Ystar = sqrt((axx*X.^2 + alpha0x - gammaXbar.*(X + X.^3))./...
53 (-ayx - bx*X.^2 + gammaXbar*(X+X.^3)));
54 Y = [0:0.005:maxY];
55 Xstar = sqrt((ayy*Y.^2 + alpha0y - gammaYbar.*(Y + Y.^3))./...
56 (-axy - by*Y.^2 + gammaYbar*(Y+Y.^3)));
57 plot(X,Ystar, 'LineWidth',1.5, 'Color',blue); hold on;
58 plot(Xstar,Y, 'LineWidth',1.5, 'Color',green); hold on;
```

C – ODE Function Used by ode23s

```
1 function dydt = ODE_model(t,y,params)
2     % load the parameters passed as input
3     alpha0x = params(1); alpha0y = params(2);
4     axx = params(3); ayy = params(4);
5     bx = params(5); by = params(6);
6     ayy = params(7); by = params(8);
7     gammaXbar = params(9);
8     gammaYbar = params(10);
9     uxbar = params(11);
10    uybar = params(12);
11    % define dynamic variables
12    X = y(1);
13    Y = y(2);
14    % define differential equations
15    numStates = 2; dydt = zeros(numStates,1);
16    dydt(1) = (alpha0x + ayy*Y^2 + axx*X^2 + bx*(X^2)*(Y^2)) ...
17              /(1 + X^2 + Y^2 + (X^2)*(Y^2)) - gammaXbar*X + uxbar;
18    dydt(2) = (alpha0y + ayy*Y^2 + axy*X^2 + by*(X^2)*(Y^2)) ...
19              /(1 + X^2 + Y^2 + (X^2)*(Y^2)) - gammaYbar*Y + uybar;
```

ODE_model.m



B – Single Parameter Bifurcation

```

1  %% bifurcate along axx
2  axx_list = [2:0.02:2.8]; % parameter range to loop through
3  for i=1:length(axx_list)
4      axx = axx_list(i); % parameters
5      uxbar = 0; uybar = 0; alpha0x = 0.01; alpha0y = 0.01; ayx = 10;
6      bx = 7.5; axy = 0.17; ayy = 1.8; by = 2; gammaX = 1; gammaY = 1;
7      % compute symbolic Jacobian with these parameters
8      syms X Y
9      dXdt = (alpha0x+axx*X^2+ayx*Y^2+bx*(X^2)*(Y^2)) ...
10     /(1+X^2+Y^2+(X^2)*(Y^2))-gammaX*X + uxbar;
11     dYdt = (alpha0y+axy*X^2+ayy*Y^2+by*(X^2)*(Y^2)) ...
12     /(1+X^2+Y^2+(X^2)*(Y^2))-gammaY*Y + uybar;
13     J_sym = jacobian([dXdt,dYdt],[X,Y]);
14     [sol_X, sol_Y] = vpasolve([dXdt==0,dYdt==0],[X, Y]);%numerical solutions
15     % find real solutions
16     sol_X = double(sort(sol_X(find(imag(sol_X)==0))));
17     sol_Y = double(sort(sol_Y(find(imag(sol_Y)==0))));
18     % store solutions in a matrix
19     steady_states = [sol_X, sol_Y];
20     % loop through the steady states just computed, determine stability
21     for j = 1:length(sol_X)
22         x = steady_states(j,1); y = steady_states(j,2);
23         % evaluate jacobian at steady state and calculate eigenvalues
24         J_numerical = double(subs(J_sym,[X,Y],[x,y]));
25         eigVals = eig(J_numerical);
26         anypositives = find(eigVals > 0); % check if any eigenvalues are
27         +
28         if length(anypositives) > 0 % unstable
29             figure(1); scatter(axx,sol_X(j),80,'b'); hold on;
30             figure(2); scatter(axx,sol_Y(j),80,'r'); hold on;
31         else % stable
32             figure(1); scatter(axx,sol_X(j),80,'b','filled'); hold on;
33             figure(2); scatter(axx,sol_Y(j),80,'r','filled'); hold on;
34         end
35     end
36 end
37

```

Figure 12: Bifurcation with Parameter a_{xx} : (A) Bifurcation plot for the steady state (X^* , Y^*) for the ODE model in equation (15). For values of a_{xx} under approximately 2.45, there are a total of 5 steady states, three of which are stable and two of which are unstable. The system bifurcates above $a_{xx} \approx 2.45$, as a stable and an unstable steady state collide and annihilate each other (also, called a saddle node bifurcation: Strogatz, 2014). (B) Matlab code for looping through a value range for the parameter a_{xx} while holding all other parameters constant (lines 5-6). To bifurcate along any other parameter, define a parameter range in line 1 and adjust lines 4-6 appropriately.

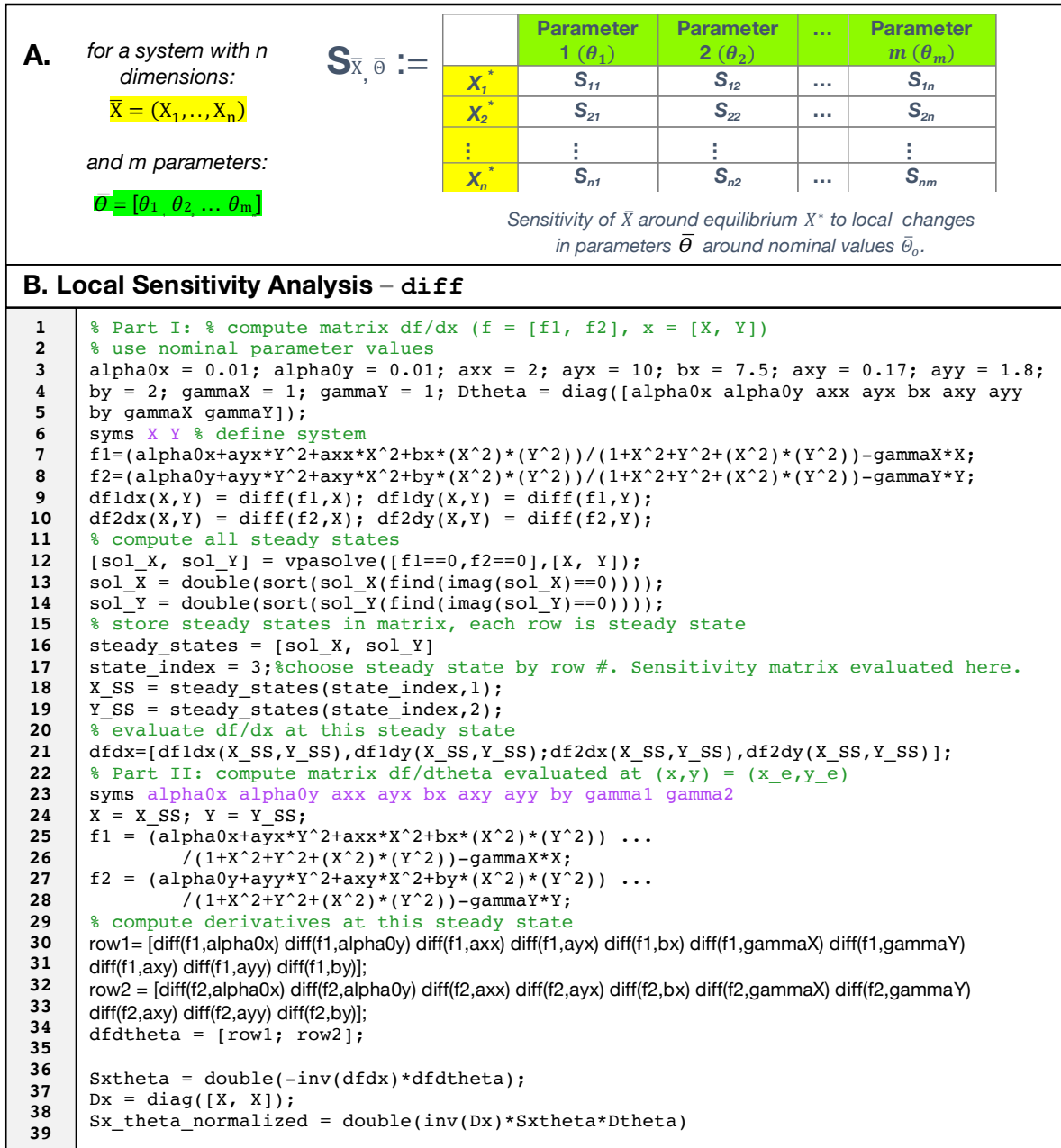
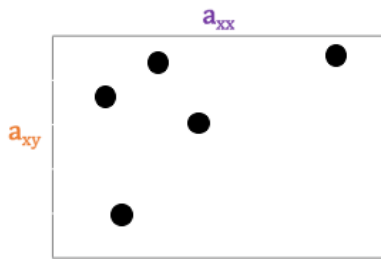
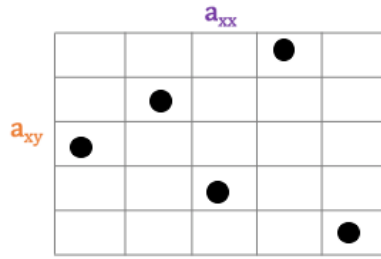


Figure 13: Sensitivity Matrices for Local Sensitivity Analysis At Steady States and Nominal Parameters
 (A) Depiction of the information captured in a sensitivity matrix. (B) Code used to implement equation 17 for the 2D system in equation 15.

A $N = 5$ Samples without Latin Hypercube Sampling



B $N = 5$ Samples using Latin Hypercube Sampling



C. Latin Hypercube Sampling – lhsnorm

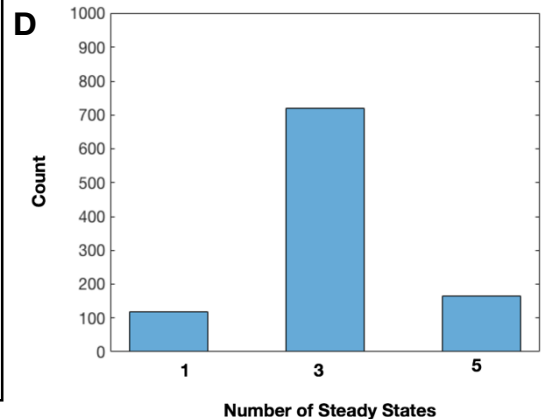
```

1  N = 1000; % # of samples (partitions in each direction)
2  % set means and std devs of normal distributions
3  mu_axx = 2; mu_ayx = 10; mu_bx = 7.5;
4  mu_axy = 0.17; mu_ayy = 1.8; mu_by = 2;
5  mu_list = [mu_axx, mu_ayx, mu_bx, mu_axy, mu_ayy, mu_by];
6  sigma_axx = 0.2; sigma_ayx = 0.2; sigma_bx = 0.2;
7  sigma_axy = 0.01; sigma_ayy = 0.2; sigma_by = 0.2;
8  sigma_list = diag([sigma_axx, sigma_ayx, ...
9                    sigma_bx, sigma_axy, sigma_ayy, sigma_by]);
10 % generate a latin hypercube sample
11 hypercube = lhsnorm(mu_list, sigma_list, N)
12 totals = []; % vector to store # total steady states
13 for i = 1:N % loop through hypercube
14     syms X Y; % define variables
15     % constant params
16     alpha0x = 0.01; alpha0y = 0.01;
17     gammaXbar = 1; gammaYbar = 1;
18     uxbar = 0; uybar = 0;
19     % params sampled from LHS space
20     tuple = hypercube(i, :);
21     axx = tuple(1); ayx = tuple(2); bx = tuple(3);
22     ayy = tuple(4); ayy = tuple(5); by = tuple(6);
23     % define ODEs
24     dXdt = (alpha0x + ayx * Y^2 + axx * X^2 + bx * (X^2) * (Y^2)) ...
25            / (1 + X^2 + Y^2 + (X^2) * (Y^2)) - gammaXbar * X + uxbar;
26     dYdt = (alpha0y + ayy * Y^2 + ayy * X^2 + by * (X^2) * (Y^2)) ...
27            / (1 + X^2 + Y^2 + (X^2) * (Y^2)) - gammaYbar * Y + uybar;
28     % find numerical solutions
29     [sol_X, sol_Y] = vpasolve([dXdt == 0, dYdt == 0], [X, Y]);
30     % find real solutions & store total # of steady states
31     sol_X = sort(sol_X(find(imag(sol_X) == 0)));
32     sol_Y = sort(sol_Y(find(imag(sol_Y) == 0)));
33     totals(i) = (length(sol_X)); %add totals to vector
34 end
35 histogram(totals) %display totals vector with histogram

```

Figure 14: Global Sensitivity Analysis using Latin Hypercube Sampling (LHS) (A)-(B) The filling of a 2D-parameter space is conceptually shown for 5 samples. Without LHS, the samples may not efficiently cover the entire parameter space. LHS enforces sampling rules such that each sample must be the only one in its row and column in a grid generated by the partitioning of the parameter space, resulting in a more uniform covering of the sample space while still being random. Partitioning of the parameter space is accomplished by partitioning each individual distribution from which each parameter is sampled (C) Using the `lhsnorm` function in Matlab, we implement a Latin hypercube sampling of the parameters a_{xx} , a_{xy} , b_{xy} , a_{yy} , a_{yx} , b_y of our 2D example in equation (15) and calculate the number of total steady states in the model across 1000 repetitions. Each parameter is sampled from a normal distribution centered at its nominal value from Figure 7, with variances as shown. (D) For the distributions used here, the system possesses three steady states most frequently. Note that this could change as the means and variances are changed, and that uniform distributions could even be used with the `lhsdesign` function in Matlab.

Documentation for `lhsnorm` in Matlab:
<https://www.mathworks.com/help/stats/lhsnorm.html>
 Documentation for `lhsdesign` in Matlab:



ARTIFICIAL PERTURBATION: OVEREXPRESSION

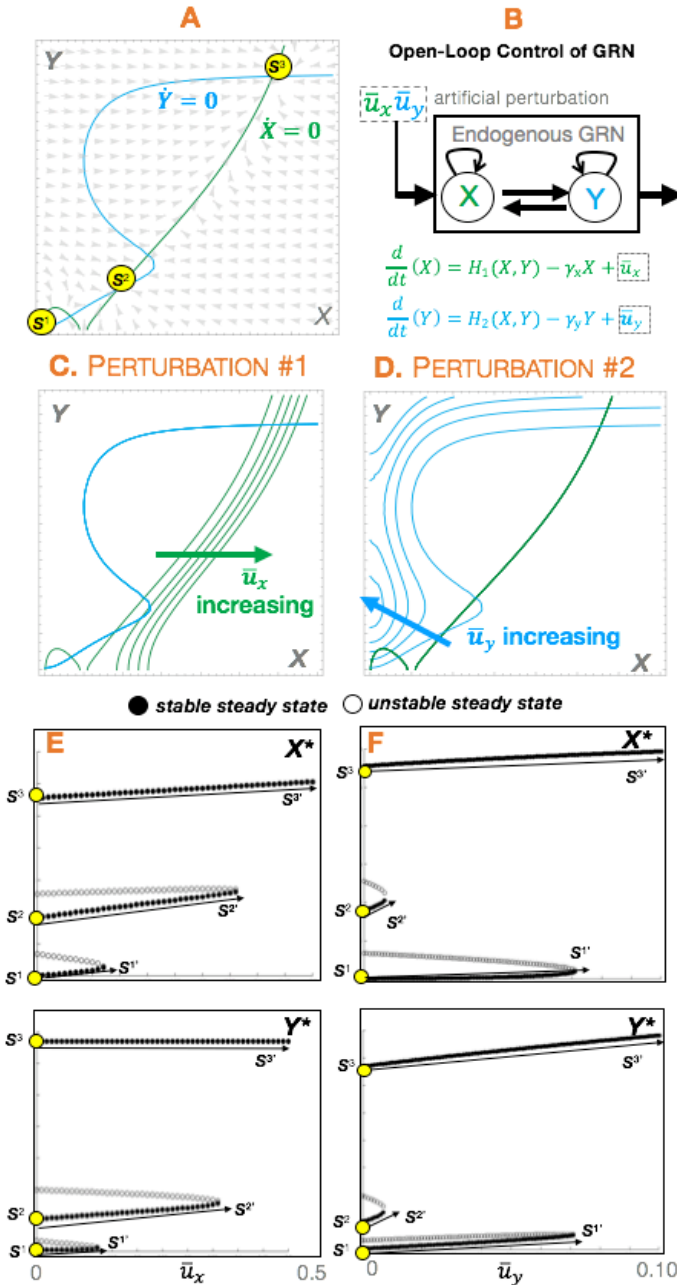
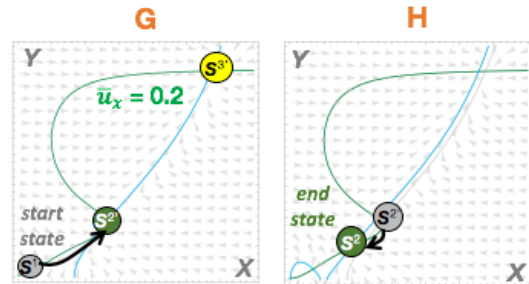


Figure 15: Educating Overexpression Strategies in Reprogramming

(A) In the steady state landscape of our model in (15), there are three SSS when the model is not artificially perturbed (same parameters used as in Fig. 7). (B) From a control systems point of view, we can regard ectopic overexpression of TFs as ‘open-loop’ control in the sense that the perturbation is applied at constant, preset, levels (\bar{u}_x and \bar{u}_y) that are not changed in response to how the concentrations of TFs change during the experiment (C) The effect of increasing \bar{u}_x is to cause the nullcline $\dot{X} = 0$ to change in a fashion that causes the steady state S^1 to disappear before S^2 and S^3 (D) The effect of increasing \bar{u}_y is to cause the nullcline $\dot{Y} = 0$ to change in a fashion that causes S_2 to disappear first, followed by the disappearance of S^1 and the persistence of only one SSS in the proximity of S^3 (E) Bifurcation plots confirm that S^1 , followed by S^2 , disappears as \bar{u}_x is increased. (F) Bifurcation plots confirm that as \bar{u}_y is increased, S^2 disappears followed by S^1 , with only one state remaining in the proximity of S^3 . (G) If the system starts at S^1 , applying an intermediate level overexpression, $\bar{u}_x = 0.2$, temporarily changes the landscape such that a transition to $S^{2'}$ is induced. (H) Upon removal of this artificial overexpression, the system transitions from $S^{2'}$ to S^2 .

Reprogramming via Open-Loop Overexpression



ARTIFICIAL PERTURBATION: ENHANCED DEGRADATION

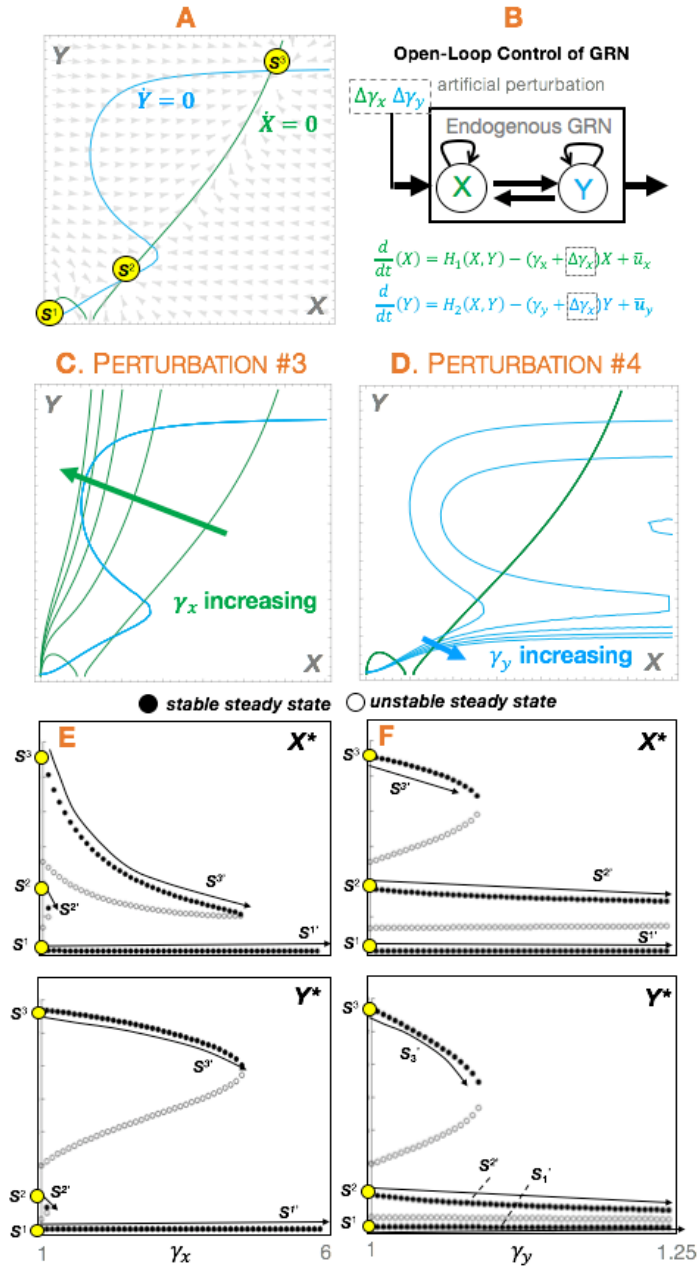


Figure 16: Educating Enhanced Degradation Strategies in Reprogramming (A) In the steady state landscape of our model in (15), there are three SSS when the model is not artificially perturbed (same parameters used as in Fig. 7). (B) We can model enhanced degradation of TFs X and Y via artificial additions of proteases or microRNA as akin to increasing the decay rates $\Delta\gamma_x$ and $\Delta\gamma_y$, respectively. (C) The effect of increasing γ_x is to cause the nullcline $\dot{X} = 0$ to change in a fashion that causes the steady state S^2 to disappear first, followed by S^3 . (D) The effect of increasing γ_y is to cause the nullcline $\dot{Y} = 0$ to change in a fashion that causes only S^3 to disappear, with two states in the proximity of S^1 and S^2 remaining. (E)-(F) Bifurcation plots confirm these observations.

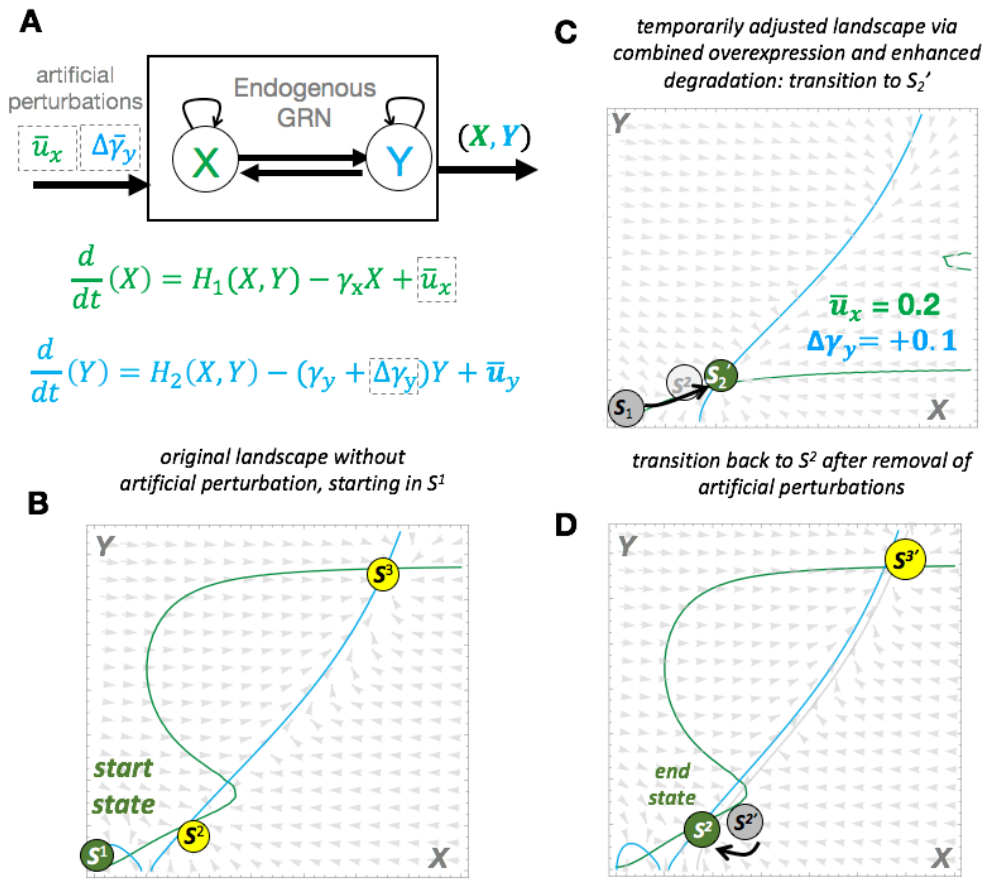
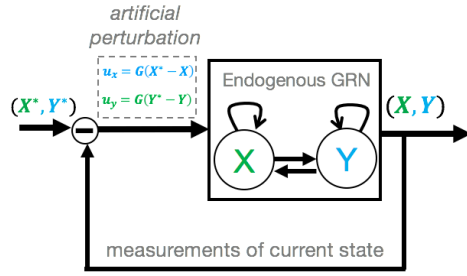


Figure 17 Reprogramming via Combination of Overexpression and Enhanced Degradation: (A) Overexpression of TF X and enhanced degradation of TF Y are modeled as the addition of constant term to the dynamics of X (\bar{u}_x) and an increased degradation ($\Delta\gamma_y$) in the dynamics of Y, respectively. (B) Without artificial perturbations, the landscape possesses the SSSs S^1 , S^2 , S^3 . (C) If the system starts at S^1 , applying an intermediate level of overexpression, $\bar{u}_x = 0.2$, combined with an increase in degradation of Y, $\Delta\gamma_y = +0.1$, temporarily changes the landscape such that a transition to $S^{2'}$ is induced. (C) Upon removal of this artificial overexpression and enhanced degradation, the system transitions from $S^{2'}$ to S^2 .

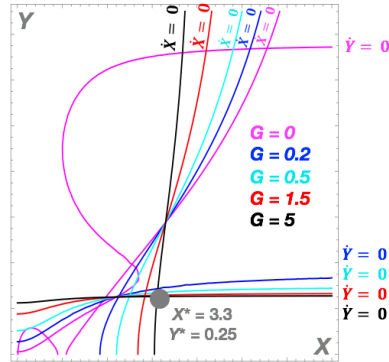
A FEEDBACK (CLOSED-LOOP) OVEREXPRESSION PARADIGM



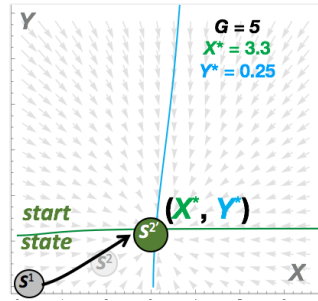
$$\frac{d}{dt}(X) = H_1(X, Y) - \gamma_x X + G \cdot (X^* - X)$$

$$\frac{d}{dt}(Y) = H_2(X, Y) - \gamma_y Y + G \cdot (Y^* - Y)$$

B REPROGRAMMING VIA FEEDBACK OVEREXPRESSION



C temporarily adjusted landscape via feedback overexpression, transition to S^{2'}



D transition back to S² after removal of feedback overexpression

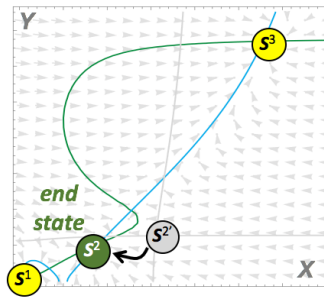


Figure 18: Reprogramming via Closed-Loop Feedback Overexpression (A) In closed-loop overexpression, measurements of the state (X, Y) are taken throughout the experiment, and the overexpression is adjusted in proportion to the distance between the current state (X, Y) and the target state (X^*, Y^*) . (B) As the values of gain G are increased, the nullclines increasingly morph into straight lines that intersect at the target state. (C) If a feedback reprogramming experiment begins in state S^1 , and there is high enough gain ($G = 5$), then feedback overexpression can be used to steer the system to any desired point $S^{2'}$ given by the coordinates (X^*, Y^*) in the basin of attraction of S^2 . (D) Upon removal of feedback overexpression, the system transitions from $S^{2'}$ to S^2 .

$$\textcircled{X} \leftarrow u_x = G(X^* - X) = GX^* - GX$$

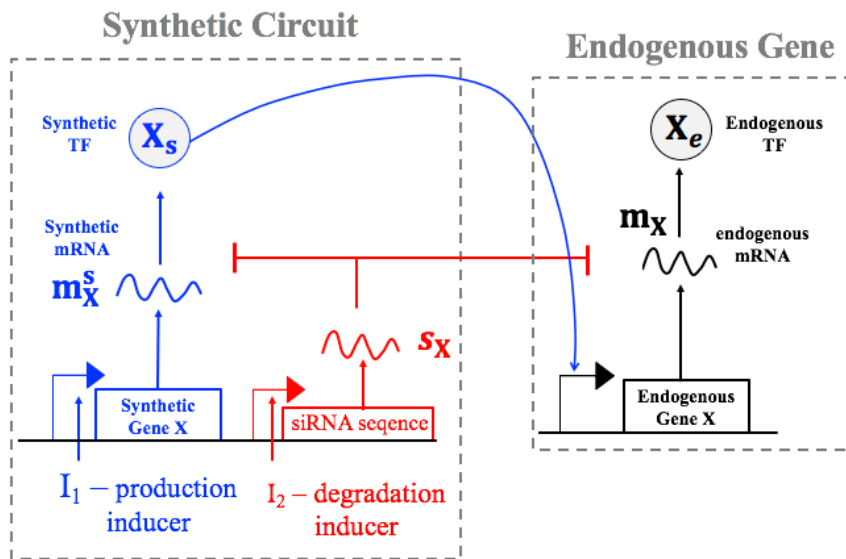
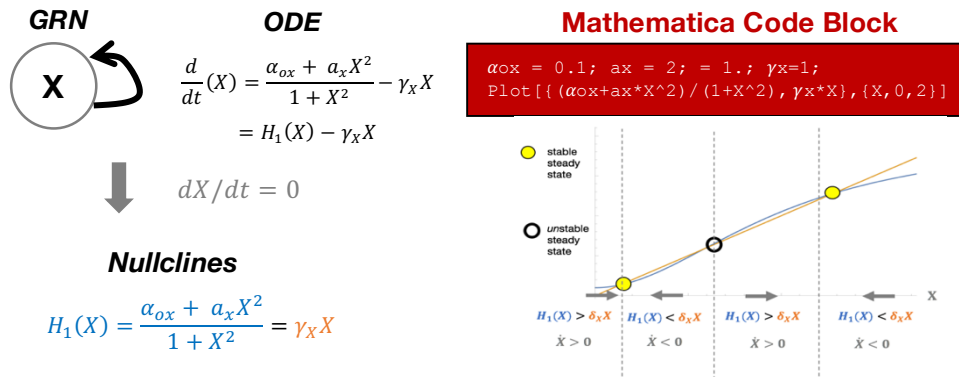


Figure 19: Realizing Feedback Overexpression with a Synthetic Genetic Circuit Feedback overexpression of a TF X can be realized with a synthetic construct consisting of inducer-activated synthetic gene X along with an inducer-activated siRNA sequence that degrades the mRNA of X. With high enough copy number for the synthetic circuit, the right balance of these inducers allows for steering the system's state to arbitrary concentration levels of X, as explained in detail in (Del Vecchio et al, 2017).

1D Geometric Analysis (Mathematica)



Appendix Figure 1: Geometric Analysis of Steady State Landscape in 1D. (left) The GRN consists of a single TF, X, auto-activating itself. Using the workflow in Section II, the 1D ODE model shown can be derived to describe the change in concentration of X over time. The location of steady states for the value of X can be found by setting the time derivative equal to zero, which is equivalent to finding the intersections between the Hill function, $H_1(X)$, and the line $\gamma_X X$. (right) These intersections mark the boundaries at which the sign of \dot{X} changes, and divide the line into discrete regions in which the sign of \dot{X} should be determined as shown. This allows for a picture of the ‘flow’ of the system along the line X as follows: when $\dot{X} > 0$, flow is to the right (X increasing) and when $\dot{X} < 0$, flow is to the left (X decreasing). Steady states which have arrows pointing towards them are stable while steady states which have arrows pointing away from them are unstable.

Documentation for Plot in Mathematica: <http://reference.wolfram.com/language/ref/Plot.html>

References

iPS Cells 10 Years Later (2016) *Cell* Volume 166, Issue 6, Pages 1356-1359
<https://doi.org/10.1016/j.cell.2016.08.043>

Abdallah H, Del Vecchio D, Qian Y, Collins JJ. (2016). A Dynamical Model for the Low Efficiency of Induced Pluripotent Stem Cell Reprogramming. Proceedings of American Control Conference.

Agrawal N, Dasaradhi PV, Mohmmmed A, Malhotra P, Bhatnagar RK, Mukherjee SK. (2003). RNA interference: biology, mechanism, and applications. *Microbiol. Mol. Biol. Rev.* 67(4): 657-685

Akashi K, Traver D, Miyamoto T, Weissman IL. (2000). A clonogenic common myeloid progenitor that gives rise to all myeloid lineages. *Nature*, 404(6774):193-197.

Alberts B, Johnson A, Lewis J, et al. (2002). *Molecular Biology of the Cell*. 4th edition. New York: Garland Science; From DNA to RNA. Available from:
<https://www.ncbi.nlm.nih.gov/books/NBK26887/>

Allis D. C., Caparros M.-L., Reinberg D., and Lachlan M. (2015). *Epigenetics*. Cold Spring Harbor, New York: Cold Spring Harbor Laboratory Press, second ed.

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRC press.

Al-Radhawi MA, Sontag E, Del Vecchio D. (2017). Multi-modality in gene regulatory networks with slow gene binding. *arXiv:1705.02330*

Astrom KJ, Murray, RM. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton UP.

Back W de, Zimm R, Bruschi L. (2013). Transdifferentiation of pancreatic cells by loss of contact-mediated signaling. *BMC Systems Biology* 7:77. doi:10.1186/1752-0509-7-77.

Bagci H. and Fisher AG. (2013). Dna demethylation in pluripotency and reprogramming: The role of tet proteins and cell division. *Cell Stem Cell*, vol. 13, pp. 265-269, 9.

Berg JM, Tymoczko JL, Stryer L. *Biochemistry*. 5th edition. New York: W H Freeman; 2002. Section 10.4, Covalent Modification Is a Means of Regulating Enzyme Activity.

Bieberich E. and Wang G. and Bhartiya D. and Lenka N. (2013). *Molecular Mechanisms Underlying Pluripotency*. Ch. 08

Boyer LA, Lee TI, Cole MF, Johnstone SE, Levine SS, Zucker JP, Guenther MG, Kumar RM, Murray HL, Jenner RG, Gifford DK, Melton DA, Jaenisch R, and Young RA. (2005). Core

transcriptional regulatory circuitry in human embryonic stem cells. *Cell*, vol. 122, no. 6, pp. 947–956

Briggs R and King T. J. (1952). Transplantation of Living Nuclei From Blastula Cells into Enucleated Frogs' Eggs *Proceedings of the National Academy of Sciences of the United States of America* **38** 455--463

Buganim Y., Faddah DA, Jaenisch R; (2013). Mechanisms and models of somatic cell reprogramming. *Nat Rev Genet.* June ; 14(6): 427–439. doi:10.1038/nrg3473

Bussmann LH, Schubert A, Vu Manh TP, De Andres L, Desbordes SC, Parra M, Zimmermann T, Rapino F, Rodriguez-Ubreva J, Ballestar E, et al. (2009). A robust and highly efficient immune cell reprogramming system. *Cell Stem Cell* 5: 554–566.

Campbell, K. H., McWhir, J., Ritchie, W. A., and Wilmut, I. (1996). Sheep cloned by nuclear transfer from a cultured cell line. *Nature* 380, 64–66. doi: 10.1038/380064a0

Carr J. Applications of Centre Manifold Theory. Springer, 1981.

Chickarmane, V., Enver, T. & Peterson, C. (2009). Computational modeling of the hematopoietic erythroid-myeloid switch reveals insights into cooperativity, priming, and irreversibility. *PLoS Comput. Biol.* 5, e1000268

Chickarmane V, Troein C, Nuber UA, Sauro HM, Peterson C (2006) Transcriptional Dynamics of the Embryonic Stem Cell Switch. *PLoS Comput Biol* 2(9): e123. <https://doi.org/10.1371/journal.pcbi.0020123>

Cooper GM. (2000.). *The Cell: A Molecular Approach*. 2nd edition. Sunderland (MA): Sinauer Associates; *The Eukaryotic Cell Cycle*. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK9876/>

David L, Polo JM (2014) Review: Phases of reprogramming. *Stem Cell Research* Volume 12, Issue 3, Pages 754-761

Davis RL, Weintraub H, Lassar AB. (1987). Expression of a single transfected cDNA converts fibroblasts to myoblasts. *Cell* 51: 987–1000.

De Carvalho D. D., You J. S., and Jones P. A. (2010). Dna methylation and cellular reprogramming. *Trends in cell biology*, vol. 20, pp. 609-617, 10

Del Vecchio D, Murray RM, *Biomolecular Feedback Systems*. Princeton UP, 2014.

Del Vecchio D, Abdallah H, Qian Y, Collins JJ. (2017). A Blueprint for a Synthetic Genetic Feedback Controller to Reprogram Cell Fate. *Cell Systems*.

Elowitz M.B., Levine A.J., Siggia E.D., and Swain P.S. (2002). Stochastic gene expression in a single cell. *Science*, 297.

Forsberg M, Carlén M, Meletis K, Yeung MSY, Barnabé-Heider F., and Persson M. A. A., Aarum J., Frisén J. (2010). Efficient reprogramming of adult neural stem cells to monocytes by ectopic expression of a single gene *Proceedings of the National Academy of Sciences* **107** 14657-14661

Friedman AD. (2007). Transcriptional control of granulocyte and monocyte development. *Oncogene*, 26(47):6816-6828.

Geertz M, Maerkl SJ. (2010). Experimental strategies for studying transcription factor–DNA binding specificities. *Briefings in Functional Genomics* 9(5-6):362-373. doi:10.1093/bfpg/elq023.

Gillespie, DT. (2000). The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306

Gillespie, DT. (2009). Deterministic Limit of Stochastic Chemical Kinetics *The Journal of Physical Chemistry B* 113 (6), 1640-1644 DOI: 10.1021/jp806431b

Goh P. A., Caxaria S., and Casper. (2013). A systematic evaluation of integration free reprogramming methods for deriving clinically relevant patient specific induced pluripotent stem (ips) cells. *PLoS ONE*, vol. 8, no. 11

Goldfarb AN (2007). Transcriptional control of megakaryocyte development. *Oncogene*.

González F. and Boué S. and Belmonte J. C. I. (2011). Methods for making induced pluripotent stem cells: reprogramming à la carte. *Nature Reviews Genetics* **12** 231 EP

Gupta P, Gurudutta GU, Saluja D, Tripathi RP. (2009). PU.1 and partners: regulation of haematopoietic stem cell fate in normal and malignant haematopoiesis. *J Cell Mol Med*, 13(11-12):4349-4363

Gurdon, JB, Elsdale, TR, Fischberg, M. (1958). Sexually mature individuals of *Xenopus laevis* from the transplantation of single somatic nuclei. *Nature* 182, 64–65. doi: 10.1038/182064a0

Gurdon J. B. and Byrne J. A. (2003). The first half-century of nuclear transplantation *Proceedings of the National Academy of Sciences of the United States of America* **100** 8048--8052

Gyorgy A, Del Vecchio D. (2014) Modular Composition of Gene Transcription Networks. *PLoS Comput Biol* 10(3): e1003486.

Horak CE, Snyder M. (2002). ChIP-chip: a genomic approach for identifying transcription factor binding sites. *Meth Enzymol* 350:469–83.

- Huang, S., G. Eichler, D. Ingber. (2005). Cell fates as high-dimensional attractor states of a complex gene regulatory network. *Phys. Rev. Lett.* 94:128701.
- Huang K. and Fan G. (2010). Dna methylation in cell differentiation and reprogramming: an emerging systematic view. *Regenerative Medicine*, vol. 5, pp. 531-544.
- Huang, S. (2009). Reprogramming cell fates: reconciling rarity with robustness. *BioEssays*. 31:546–560.
- Huang S, Guo YP, May G, Enver T. (2007). Bifurcation dynamics in lineage-commitment in bipotent progenitor cells. *Dev Biol.* 305(2):695-713.
- Kauffman, S. (1973). Control circuits for determination and transdetermination. *Science*. 181:310–318.
- Kim, Jonghwan et al. (2008). An extended transcriptional network for pluripotency of embryonic stem cells. *Cell*, Volume 132, Issue 6, 1049 – 1061
- Kuhar MJ. (2010). Measuring *levels* of proteins by various technologies: can we learn more by measuring turnover? *Biochemical pharmacology* 79(5):665-668. doi:10.1016/j.bcp.2009.09.029.
- Laiosa, Catherine V. et al. (2006). Reprogramming of Committed T Cell Progenitors to Macrophages and Dendritic Cells by C/EBP α and PU.1 Transcription Factors. *Immunity*, Volume 25, Issue 5, 731 – 744.
- Liew CW, Rand KD, Simpson RJ, Yung WW, Mansfield RE, et al. (2006) Molecular analysis of the interaction between the hematopoietic master transcription factors GATA-1 and PU.1. *J Biol Chem* 281: 28296–28306.
- Malik N. and Rao M. S. (2013). A review of the methods for human ipsc derivation,” *Methods in molecular biology* (Clifton, N.J.), vol. 997, pp. 23–33.
- Mariani, L, Schulz EG, Lexberg MH, Helmstetter C, Radbruch A, Lohning M, and Hofer T. (2003). Short-term memory in gene induction reveals the regulatory principle behind stochastic il-4 expression. *Molecular systems biology*, 6(1):359, 2010.
- Mckay and Beckman (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code;
- Milo, R., Jorgensen, P., Moran, U., Weber, G., and Springer, M. (2010). BioNumbers—the database of key numbers in molecular and cell biology. *Nucleic Acids Res.* 38: D750–D753
- Mitalipov S., Wolf D. (2009) Totipotency, Pluripotency and Nuclear Reprogramming. In: Martin U. (eds) *Engineering of Stem Cells. Advances in Biochemical Engineering / Biotechnology*, vol 114. Springer, Berlin, Heidelberg

- Olariu V. and Lövkvist C. and Sneppen K. (2016). Nanog, Oct4 and Tet1 interplay in establishing pluripotency. *Scientific Reports* **6** 25438 EP
- Orkin SH, Wang J, Kim J, Chu J, Rao S, Theunissen TW, Shen X, Levasseur DN. (2008). The Transcriptional Network Controlling Pluripotency in ES Cells *Cold Spring Harbor Symposia on Quantitative Biology* **73** 195--202
- Pang Z. P. and Yang N. and Vierbuchen T. and Ostermeier A. and Fuentes D. R. and Yang T. Q. and Citri A. and Sebastiano V. and Marro S. and Südhof T. C. and Wernig M. (2011). Induction of human neuronal cells by defined transcription factors *Nature* **476** 220 EP
- Park PJ. (2009). ChIP-seq: advantages and challenges of a maturing technology. *Nat Rev Genet.*10:669–80.
- Polynikis A, Hogan SJ, di Bernardo M. (2009). Comparing different ODE modelling approaches for gene regulatory networks. *J Theor Biol.* 261(4):511-30.
- Radzishenskaya A, Chia Gle B, dos Santos RL, Theunissen TW, Castro LF, Nichols J, Silva JC. (2013). A defined Oct4 level governs cell state transitions of pluripotency entry and differentiation into all embryonic lineages. *Nat Cell Biol.* 2013 Jun;15(6):579-90.
- Raj A, Peskin CS, Tranchina D, Vargas DY, and Tyagi S. (2006). Stochastic mRNA synthesis in mammalian cells. *PLoS Biol*, 4(10):e309
- Santillán, M. (2008). On the Use of the Hill Functions in Mathematical Models of Gene Regulatory Networks. *Mathematical Modelling of Natural Phenomena*, 3(2), 85-97. doi:10.1051/mmnp:2008056
- Schlaeger T. M. and Daheron. (2015). A comparison of non-integrating reprogramming methods,” *Nat Biotech*, vol. 33, no. 1, pp. 58–63.
- Schwanhäusser B., Busse D., Li N., Dittmar G., Schuchhardt J., Wolf J, Chen W, Selbach, M. (2011). Global quantification of mammalian gene expression control. *Nature* **473** 337 EP
- Sezonov G, Joseleau-Petit D, D'Ari R. (2007). Escherichia coli physiology in Luria-Bertani broth. *J Bacteriol.* 189(23):8746-9.
- Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics*. Vol. 29, No. 2, pp. 143–151. Correction, Vol. 32, p. 367.
- Strang, G. (2009). Introduction to Linear Algebra, Wellesley-Cambridge Press.
- Strogatz, S. (2014). Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering (Studies in Nonlinearity) 2nd Edition

Swain P. S., Elowitz M. B., and Siggia E.D. (2002). Intrinsic and extrinsic contributions to stochasticity in gene expression. In Proc. Natl Acad. Sci. USA 99, 2002.

Takahashi K. and Yamanaka S. (2006). Induction of pluripotent stem cells from mouse embryonic and adult fibroblast cultures by defined factors. *Cell*, vol. 126, no. 4, pp. 663–676.

Takahashi K. and Yamanaka S. (2016). A decade of transcription factor-mediated reprogramming to pluripotency. *Nature Reviews Molecular Cell Biology* **17** 183 EP

Tapscott SJ, Davis RL, Thayer MJ, Cheng PF, Weintraub H, Lassar AB (1988). MyoD1: a nuclear phosphoprotein requiring a Myc homology region to convert fibroblasts to myoblasts *Science* 21 405-411

To T-L, Maheshri N. (2010). Noise can induce bimodality in positive transcriptional feedback loops without bistability. *Science*, 327(5969):1142–1145.

Tian, T., Smith-Miles, K. (2014). Mathematical modeling of GATA-switching for regulating the differentiation of hematopoietic stem cell. *BMC Syst Biol.* 2014; 8(Suppl 1): S8.

Van Kampen, NG. (1992). Stochastic processes in physics and chemistry, volume 1. Elsevier.

Vierbuchen T. and Ostermeier A. and Pang Z. P. and Kokubu Y. and Südhof T. C. and Wernig M. (2010). Direct conversion of fibroblasts to functional neurons by defined factors *Nature* **463** 1035 EP

Waddington, C.H. (1957). *The Strategy of Genes* (New York: Routledge).

Wang Z, Gerstein M, Snyder M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews Genetics* 10(1):57-63. doi:10.1038/nrg2484.

Wakayama, T., Perry, A. C., Zuccotti, M., Johnson, K. R., and Yanagimachi, R. (1998). Full-term development of mice from enucleated oocytes injected with cumulus cell nuclei. *Nature* 394, 369–374. doi: 10.1038/28615

Wiggins S. (2003). *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer.

Xie H, Ye M, Feng R, Graf T. (2004). Stepwise reprogramming of B cells into macrophages. *Cell*. May 28;117(5):663-76.

Yao E, Lin C, Wu Q, Zhang K, Song H, Chuang PT. (2017). Notch Signaling Controls Transdifferentiation of Pulmonary Neuroendocrine Cells in Response to Lung Injury. *Stem Cells*.

Yuan L, Chan GC, Beeler D, Janes L, Spokes KC, Dharaneeswaran H, Mojiri A, Adams WJ, Sciuto T, Garcia-Cardena G, et al. (2016). A role of stochastic phenotype switching in generating mosaic endothelial cell heterogeneity. *Nature communications*, 7:10160

Yu Ungsik, Lee SH, Kim YJ and Kim S. (2004). Review: *Bioinformatics in the Post-genome Era*. *Journal of Biochemistry and Molecular Biology*, Vol. 37, No. 1, January 2004, pp. 75-82

Zhang, B., and P. Wolynes. (2014). Stem cell differentiation as a manybody problem. *Proc. Natl. Acad. Sci. USA*. 111:10185–10190.

Zhou J.X., Bruschi, L., and Huang, S. (2011). Predicting Pancreas Cell Fate Decisions and Reprogramming with a Hierarchical Multi-Attractor Model. *PLoS One*.

Zhou P. (2004). Determining Protein Half-Lives. *Signal Transduction Protocols* pp 67-77

Zhou Q, Brown J, Kanarek A, Rajagopal J, Melton DA. (2008). In vivo reprogramming of adult pancreatic exocrine cells to b-cells. *Nature* 455: 627–632.