AN ALGORITHM FOR THE STOCHASTIC SIMULATION OF GENE EXPRESSION AND CELL POPULATION DYNAMICS

Daniel A. Charlebois

Thesis submitted to the Faculty of

Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree of

MSc Physics

Department of Physics

Faculty of Science

University of Ottawa

Ottawa-Carleton Institute for Physics

ABSTRACT

AN ALGORITHM FOR THE STOCHASTIC SIMULATION OF GENE EXPRESSION AND CELL POPULATION DYNAMICS

Daniel A. Charlebois

Department of Physics

MSc Physics

Over the past few years, it has been increasingly recognized that stochastic mechanisms play a key role in the dynamics of biological systems. Genetic networks are one example where molecular-level fluctuations are of particular importance. Here stochasticity in the expression of gene products can result in genetically identical cells in the same environment displaying significant variation in biochemical or physical attributes. This variation can influence individual and population-level fitness.

In this thesis we first explore the background required to obtain analytical solutions and perform simulations of stochastic models of gene expression. Then we develop an algorithm for the stochastic simulation of gene expression and heterogeneous cell population dynamics. The algorithm combines an exact method to simulate molecular-level fluctuations in single cells and a constant-number Monte Carlo approach to simulate the statistical character-

istics of growing cell populations. This approach permits biologically realistic and computationally feasible simulations of environment and time-dependent cell population dynamics. The algorithm is benchmarked against steady-state and time-dependent analytical solutions of gene expression models, including scenarios when cell growth, division, and DNA replication are incorporated into the modelling framework. Furthermore, using the algorithm we obtain the steady-state cell size distribution of a large cell population, grown from a small initial cell population undergoing stochastic and asymmetric division, to the size distribution of a small representative sample of this population simulated to steady-state. These comparisons demonstrate that the algorithm provides an accurate and efficient approach to modelling the effects of complex biological features on gene expression dynamics. The algorithm is also employed to simulate expression dynamics within 'bet-hedging' cell populations during their adaption to environmental stress. These simulations indicate that the cell population dynamics algorithm provides a framework suitable for simulating and analyzing realistic models of heterogeneous population dynamics combining molecular-level stochastic reaction kinetics, relevant physiological details, and phenotypic variability and fitness.

FORWARD

Throughout the master's program I had the opportunity to attend and present posters on my research at two conferences:

1. 2009 Progress in Systems Biology, April 23-24, Ottawa, Canada. 'Development of a parallel algorithm for the stochastic simulation of gene regulatory network dynamics in cells of a lineage and fixed populations'. Daniel A. Charlebois and Mads Kaern.

2. HiBi09 - 2009 International Workshop on High Performance Computational Systems Biology, October 14-16, Trento, Italy. 'Benchmarking a Parallel Algorithm for the Stochastic Simulation of Gene Expression and Population Dynamics'. Daniel A. Charlebois, Dawn Fraser, Jukka Intosalmi and Mads Kaern.

Additionally, an article titled 'An Algorithm for the Stochastic Simulation of Gene Expression and Heterogeneous Population Dynamics' [1] has been published in the journal *Communications in Computational Physics*. The major contribution of this article is the augmentation and validation of an accurate and efficient framework for simulating and analyzing biologically realistic models of heterogeneous population dynamics. This work embodies the results of my master's research and is presented in Section 4.

ACKNOWLEDGMENTS

The author would like to especially thank Dr. Mads Kaern and Dawn Fraser for their guidance and support, and also Nezar Abdennur for editing the thesis. This work was carried out and supported financially by the Dynamical Systems Biology Laboratory at the University of Ottawa, Canada, and the author would like to extend his thanks to the entire laboratory, whose members, in one way or another, enhanced his graduate experience.

This thesis is dedicated to my mother, Stephanie Marie Gillespie-Charlebois.

Ottawa, April 2010

Daniel Charlebois

Contents

List of Figures

List of Symbols

T	Promoter
T_A	Active promoter (also denoted by A)
T_R	Repressed promoter (also denoted by R)
M	Messenger RNA (mRNA)
P	Protein
s_X	Production rate of molecule X
δ_X	Decay rate of molecule X
V_i	Volume of cell i
V_0	Initial cell volume (i.e. volume following cell division)
V_{div}	Critical volume at which the cell divides (also represented by V_c)
t_{div}	Time since last cell division
t_{cdiv}	Cell division time interval
τ_0	Time interval between cell volume doublings
t_d	Time at which gene replication occurs (also denoted by t_{rep})
$t_{restore}$	Time interval at which population is restored to some fixed number of individuals
t_{sample}	Time interval at which data for in silico population is obtained
$[X]$	Number concentration of X
$\langle X \rangle$	Expected value of X
μ_X	Mean number of X
σ_X	Standard deviation of X
σ_X^2	Variance of X
η	Relative deviation from the average, or *noise*
CV	Coefficient of variation
Y^s	Steady-state value of quantity Y (also denoted by $\bar{Y}$)
p_k	Probability of the system to occupy each one of a discrete set of states k
E_i^k	Step operator which describes addition/removal of k molecules of species i
$\prod$	Probability density (as opposed to $\prod_{i=1}^{N}$ which represents the product of the N terms)
ξ	Gaussian noise term
τ	Time at which next reaction occurs (also used to denote dimensionless time)
R_μ	Next reaction (index μ) to occur
a_i	Propensity of reaction i
a_0	Sum of reaction propensities

Chapter 1

Introduction

1.1 Noise in Biochemical Reactions

Biochemical reactions are discrete events as molecular population numbers can only change by discrete integer amounts. Furthermore, reactions are stochastic since they occur as a result of collisions between randomly moving molecules. Consequently this leads to the non-deterministic timing of individual reactions and an inherently noisy time evolution of molecular population levels [2].

Specific physical reasons exist as to why the dynamics of chemically reacting systems are non-deterministic. Chemical systems are not usually mechanically isolated, but rather in contact with a 'heat bath' which keeps the system in thermal equilibrium at some temperature. When the system is in thermodynamic equilibrium, the molecules will at all times be distributed randomly and uniformly throughout the containing volume. These molecules move with erratic motions as a result of the uneven bombardment of underlying fluid molecules (Brownian motion), which themselves have essentially random motions due to thermal fluctuations from the environment and collisions with other fluid molecules (self-diffusion). These erratic

motions result in erratic collisions, and any resulting reactions occur in an essentially random manner [2–4]. Another reason for the non-deterministic time-evolution of a chemical system involves quantum indeterminacy. For example, in a unimolecular reaction it is impossible to know exactly when a molecule will transform itself into a different isometric form [3].

1.2 Stochasticity in Gene Expression

Advancement in experimental techniques for empirically measuring single cells and in corresponding theoretical methods have enabled the rigorous design and interpretation of experiments that provide incontrovertible proof that there are important endogenous sources of stochasticity that drive biological processes [5].

One example of particular importance is the stochastic expression of gene products (mRNA and protein) [5–9]. Specifically, the multistep processes that lead to the synthesis and degradation of messenger RNA (mRNA) and protein molecules are inherently stochastic due to the underlying binding events which occur as a result of the random collisions between small numbers of molecules (e.g. transcription factors binding to one or two copies of a gene) [6]. A model of the process of expressing a single gene is shown in Figure 1.1. Although this depiction is simple compared to the true complexity of gene expression, it captures the essential features including the switching of the DNA template between transcriptionally active (A) and repressed (R) states at rates that depend on the binding of transcriptional regulators to the promoter region of the gene (rates k_{on} and k_{off}), the synthesis of mRNA (M) from a single gene copy (at a rate s_R or s_A, depending on whether the promoter is in the R or A state, respectively), the synthesis of protein (P) from mRNA templates (rate s_P), and the decay of mRNA and protein molecules (rates δ_M and δ_P respectively)

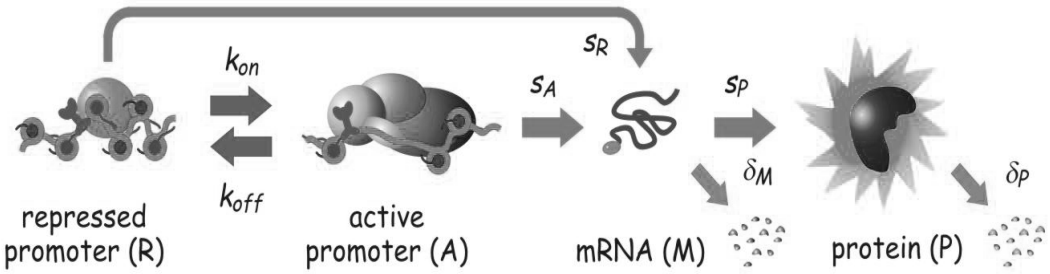

Figure 1.1 A simple model for the expression of a single gene (each step represents several biochemical reactions). All steps are modelled as first-order reactions with the indicated rate constants (units of inverse time) associated with these steps. Figure used with permission from Scott *et al.* [10].

[6, 10]. Note that $k_{on}/(k_{on} + k_{off})$ and $k_{off}/(k_{on} + k_{off})$ are the fractions of time that the gene spends in the active and repressed states, respectively. Although more complex models of gene expression have been developed (e.g. [11–14]), the simple model depicted in Figure 1.1 is sufficient for the purpose of this thesis.

1.3 Modelling Gene Expression

Traditionally, the time evolution of a chemical system is modelled as a deterministic process using a set of ordinary differential equations (ODEs). This approach is based on the empirical law of mass action, which provides a relation between reaction rates and molecular concentrations [15]. Generally, the instantaneous rate of a reaction is directly proportional to the concentration (which is in turn proportional to mass) of each reactant raised to the power of its stoichiometry.

In the deterministic description of the model shown in Figure 1.1, the cellular mRNA and protein concentrations ($[M]$ and $[P]$, respectively) are governed by the macroscopic rate equations

$$\frac{d[M]}{dt} = \frac{s_A}{V}\frac{k_{on}}{k_{on}+k_{off}} + \frac{s_R}{V}\frac{k_{off}}{k_{on}+k_{off}} - \delta_M[M], \tag{1.1}$$

$$\frac{d[P]}{dt} = s_P[M] - \delta_P[P], \tag{1.2}$$

where V is the cell volume; the terms $\delta_M[M]$ and $\delta_P[P]$ are the degradation rates for mRNA and proteins respectively; the term $s_P[M]$ is the rate of protein synthesis. Here the promoter is assumed to be in chemical equilibrium and consequently mRNA production occurs at a constant rate given by the weighted average of the activated (s_A) and repressed (s_R) mRNA synthesis rates (denoted by Λ). The steady-state concentrations are given by

$$[M^s] = \frac{\Lambda}{\delta_M}, \tag{1.3}$$

$$[P^s] = \frac{\Lambda s_P}{\delta_M \delta_P}, \tag{1.4}$$

and are related to the average steady-state number of M and P by V.

Note that the deterministic mathematical model (Eqs. (1.1) and (1.2)) was obtained by treating each step as a first-order chemical reaction and applying the law of mass action. The law of mass action was developed to describe chemical reactions under conditions where the number of each chemical species is so large that concentrations can be approximated as continuous variables without introducing significant error [10].

The conditions that need to be satisfied for the deterministic approach to provide a valid approximation of the exact stochastic description (see Section 2.1.1) are fast reaction kinetics (here large k_{on} and k_{off} values) and large system size in terms of large numbers of each species and system volume (here large s_R, s_A and V so that the number of expressed mRNA and protein molecules is high with their ratios s_R/V and s_A/V remaining constant) [6]. When these conditions are not satisfied, the

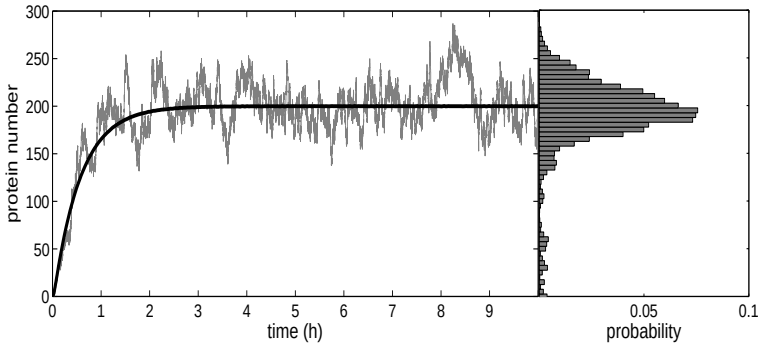

Figure 1.2 Time series of protein number resulting from constant gene expression generated by deterministic and stochastic simulations (black and gray curves, respectively). The histogram in the right-hand panel shows the probability that a cell will have a given intracellular protein level. Parameter were set to (units s^{-1}): $s_A = 0.02$, $s_R = 0$, $s_P = 0.05$, $\delta_M = 0.0005$, and $\delta_P = 0.01$. V was set to unity (a.u.).

effects of molecular noise can be significant. These conditions are not satisfied for gene expression, due to low copy number of genes, mRNAs, and transcription factors within the cell [16].

When the deterministic ODEs presented in Eqs. (1.1) and (1.2) are numerically simulated, although they can in certain parameter regimes capture the mean behavior of these types of systems, they cannot capture the fluctuations about the mean and therefore the resulting probability distributions (Fig. 1.2). Futhermore, when rates depend nonlinearly on randomly fluctuating components, macroscopic rate equations may be far off the mark even in their estimates of averages [17]. Reliable averages over cell populations can then only be found from ensembles of probabilistic single-cell descriptions as the behavior of the system at deviations below the average expression may not compensate for deviations above.

One case which emphasizes the importance of using a stochastic framework for

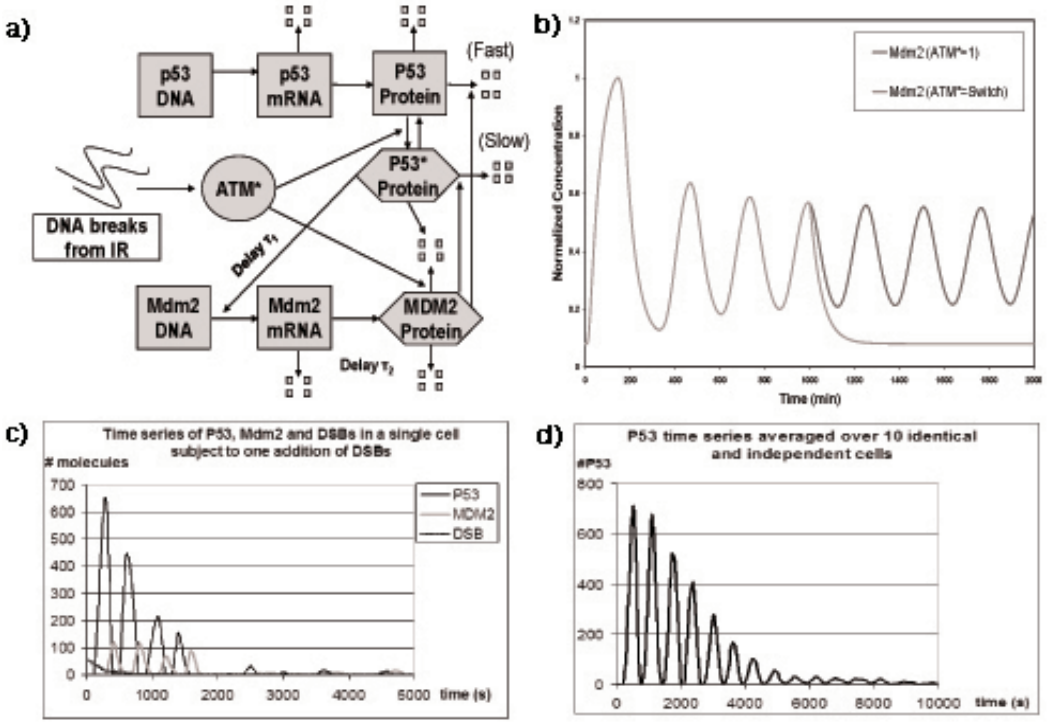

Figure 1.3 (a) P53-MDM2 network (reproduced from [11]). **(b)** Time series obtained from simulations of a deterministic P53-MDM2 model [11] - the number of oscillations either end abruptly after the damaged DNA is repaired and ATM is switched off, or they continue indefinitely. **(c)** Time series from a single realisation (representing one cell) of a stochastic P53-MDM2 model [25]. **(d)** Average number of P53 molecules obtained from 10 realisations (representing a population of cells) of the model used in **c**. Note that figures **(c)** and **(d)** were obtained from simulations carried out as part of a previous project [25]).

modeling biological systems is the P53-MDM2 network [18]. P53 is a human tumour suppressor protein (encoded by the p53 gene) which is activated (phosphorylated) by the presence of DNA damage (which can be induced by ionizing radiation). Activated P53 then upregulates the transcription of the Mdm2 gene, which in turn negatively regulates (degrades) the P53 protein (Fig. 1.3a). This negative feedback loop results in oscillatory dynamics in P53 and MDM2 concentrations until the DNA damage present in the system is repaired (Fig. 1.3b - when the DNA damage sensing protein kinase ATM is modelled as a switch, and Fig. 1.3c) [11, 19, 20]. Experiments show that the number of oscillations in response to DNA damage can vary among individual cells of a population, and as a result, a damped oscillation is observed at the population-level [19–22]. The deterministic models (e.g., [11, 23]) do not capture the full dynamics of the P53-MDM2 system as they inevitably fail to account for the observed heterogeneity in cellular response to DNA damage [18]. Specifically, in the deterministic models cellular P53 and MDM2 concentrations either oscillate a finite or indefinite number of times (Fig. 1.3b), with no variability in response to DNA damage across the cell population. This lack of agreement between the deterministic models and experimental results cannot be attributed to genetic or environmental effects, as these have been largely eliminated by experimental design [18]. On the other hand, stochastic models of the P53-MDM2 system (e.g., [24, 25]) can capture both the cell-to-cell variability (Fig. 1.3c) and population-level behavior that is observed experimentally (Fig. 1.3d).

Another advantage of using a stochastic framework (see Section 2.2) to simulate the model of gene expression under consideration (Fig. 1.1) can be seen in Figure 1.2. Specifically, the stochastic methods capture not only the mean protein concentration, but also the fluctuations in protein abundance. These fluctuations provide the information necessary for the histograms that describe the probability that a cell will

have a given level of a particular molecular species. Futhermore, population level probability distributions can be obtained from the ensemble of single-cell time-series, or when the ergodic hypothesis is satisfied, from the time-series of a single cell. Capturing these probability distributions is often required to describe and predict the dynamics of biological systems. Take for example the development of drug resistance during chemotherapy. When the drug Imatinib is used to treat chronic myeloid leukemia, the disease recurs with a frequency of 20-30 %. Even though numerous genetic mutations have been shown to render the drug ineffective, in two-thirds of the cases no mutations have been found. Instead, elevated levels of survival pathway proteins in Imatinib-resistant leukaemia cell lines were detected. The rapid rate of resistance development, its dose dependence and high frequency of upregulation of the correct pathways are consistent with non-genetic heterogeneity which generates enduring outlier cells with distinct phenotypes (i.e. any observable biochemical or physical attribute), some of which may be subject to selection [26]. This non-genetic heterogeneity, or variance within the population, can be captured in probability histograms (where outlier cells would be accounted for in the tails of the distribution), generated from a nondeterministic modeling framework.

1.4 Quantifying Noise in Gene Expression

The term 'noise' when used in the context of gene expression is a broad reference to observed variation in protein content among apparently identical cells exposed to the same environment [27]. This noise can be divided up into extrinsic and intrinsic components. Extrinsic noise can generally be defined as fluctuations and variability that arise in a system due to disturbances originating from its environment, and therefore depends on how the system of interest is defined [10]. Extrinsic gene expression

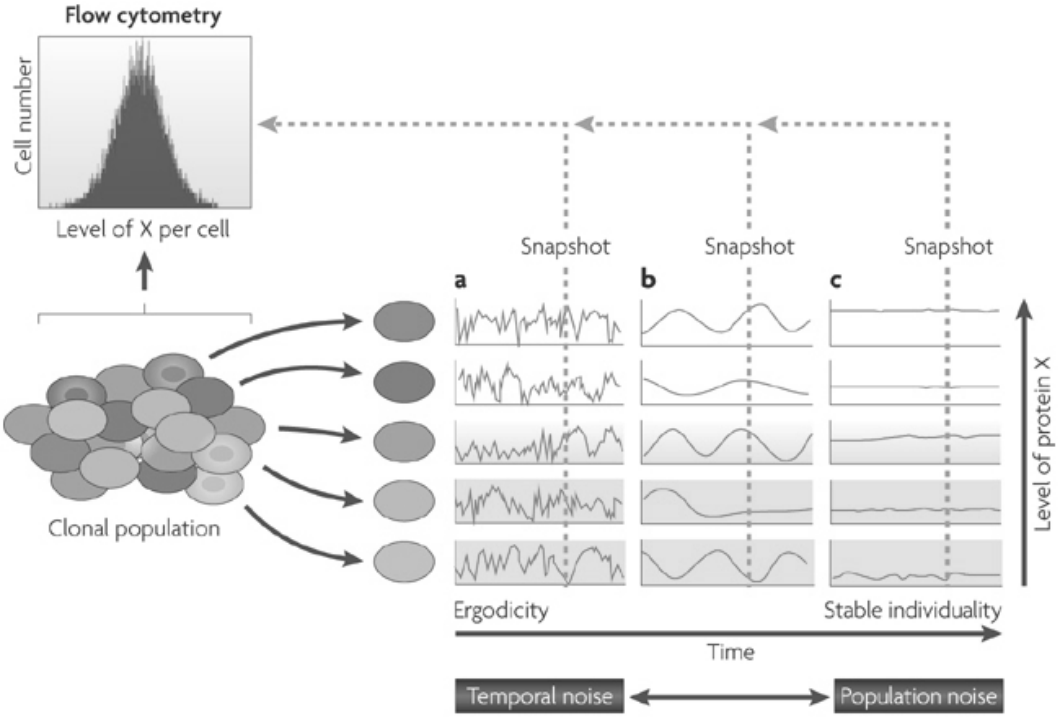

Figure 1.4 Schematic of flow cytometry revealing heterogeneity of phenotype (expression level of protein X) in a clonal cell population. Three idealized interpretations for the spread in the level of X obtained from a population snapshot are shown. All of them give rise to the distribution in the histogram (shown here as a Gaussian distribution, but other distributions are possible). (a) Fast random fluctuations sufficiently fast such that each cell visits all possible states. (b) Asynchronous (but deterministic) slow fluctuations such as oscillatory processes. (c) An extreme case in which each cell has one more or less stable (i.e. time-invariant) but cell-specific level of X. Figure used with permission from Brock *et al.* [26].

noise arises from several sources including: the metabolic state of the cell, cell-cycle phase, cell age, and variability in upstream signal transduction [6, 8, 13, 27–32]. Intrinsic expression noise, which is the focus of this thesis, refers to variation that arises from 'finite-number' molecular-level fluctuations inherent to reaction kinetics in the nanomolar range during the expression of individual alleles and is illustrated by the following example. At equilibrium, nuclear and cytoplasmic protein concentrations are, on average, equal. However, because the volume of the nucleus is much less than that of the cytoplasm, a larger fluctuation occurs as a result of a protein moving from the cytoplasm to the nucleus than vice-versa. So if 10 molecules are present in the nucleus and 1000 protein molecules in the cytoplasm, a translocation of a single protein molecule across the nuclear membrane will result in a 10 % change in the nuclear concentration and only a 0.1 % in the cytoplasmic protein concentration. This differential effect resulting from the difference in molecule numbers in the two compartments is referred to as the 'finite-number effect', which is perhaps the most commonly recognized manifestation of molecular-level noise in cellular regulation [6].

Several noise measures are used to quantify the degree of heterogeneity in gene expression. The relative deviation from the average is measured by the ratio of the standard deviation σ to the mean μ, and is what is generally referred to as noise η. Although the term 'coefficient of variation' (CV) is sometimes used interchangeably with η, it will here be used to specifically to define the ratio σ^2/μ^2. Furthermore, unless otherwise specified, η will here refer to intrinsic noise. Another more sensitive measure of noise is known as the 'fano factor' ($\vartheta = \sigma^2/\mu$), can be used to uncover trends that would otherwise be obscured by the characteristic $1/\sqrt{N}$ scaling of the noise, where N is the average molecular abundance, arising from finite-number effects [6, 33].

Heterogeneity within a clonal population of cells of a single cell type can be mea-

sured using flow-cytometry analysis, which produce histograms for the abundance of a given protein per cell in a population of cells (see Figure 1.4). Within the peak of the histogram, the abundance of the protein in the cells with the lowest and highest expression level typically differs by three or more orders of magnitude; this spread far exceeds signal measurement noise [26].

1.5 Heterogeneous Cell Populations and Fitness

Genetically identical cells in the same environment can display significant variation in molecular content and thus exhibit significant variation in phenotypes [6, 34]. This heterogeneity in a cell population is probably the most apparent manifestation of stochastic gene expression. In the simplest case, the concentration of a constitutively expressed protein could display some variability from cell to cell [28, 35]; more interestingly, a cell population could split into two or more groups, each of which is characterized by a distinct state of gene expression [34]. For example, stochastic mechanisms can cause a population of isogenic bacteria, where the entire population is subject to the same environment, to exhibit diverse patterns of gene expression with the resulting phenotypic subpopulations displaying distinct growth rates [34]. These fluctuations in gene expression provide a mechanism for 'sampling' physiologically distinct states, which can increase the probability of survival during times of stress without the need for genetic mutation [6, 34].

Under fixed environmental conditions, the net growth rate (fitness) of the population is maximized when all cells are of the fastest growing phenotype. However, in a changing environment, it is thought that a statically heterogeneous population (i.e. a population where transitions between states are not influenced by environmental conditions) can deal with an uncertain future by hedging its bets, that is by generating

a broad distribution of phenotypes in the hope that some of these will remain viable after an external change [34]. In contrast, a dynamically heterogeneous population has a more reliable strategy: individuals in such populations sense and respond to external changes by actively switching to the fit state. In this case if the response rate is sufficiently rapid compared to the rate of environmental fluctuations, as is the case for many real systems, then transitions into the unfit state are actually detrimental and so bet-hedging is only beneficial if response rates are sufficiently low.

The probabilistic features arising from gene expression noise led to the hypothesis that evolution has fine-tuned noise-generating mechanisms and genetic architectures to derive beneficial population diversity [36–38]. Direct evidence that genome sequence contributes to cell-cell variability indicates that gene expression noise, like other genome-encoded traits, is inheritable and subject to selective pressures, and therefore evolvable. Specifically, large-scale proteomic studies in yeast have shown that genes associated with stress response pathways have elevated levels of intrinsic noise [39–41]. Stress-response genes have thus experienced positive pressure toward high population variability, presumably by providing a selective advantage during periods of stress.

1.6 Simulating the Dynamics of Heterogeneous Cell Populations

Biological systems can be modelled at multiple scales, from detailed physical descriptions of molecular interactions to phenomenological representations of populations of organisms. Due to the importance of noise in many biological systems, models involving stochastic formulations of chemical kinetics are increasingly being used to simulate and analyze cellular control systems [42]. In many cases, obtaining analytical

solutions for these models are not feasible due to the intractability of the corresponding system of nonlinear equations. Thus, Monte Carlo (MC) simulation procedures for numerically calculating the time evolution of a spatially homogeneous mixture of molecules are commonly employed (see Section 2.2). Among these procedures, the Gillespie stochastic simulation algorithm (SSA) is the *de-facto* standard for simulating biochemical systems in situations where a deterministic formulation may be inadequate [2, 43].

As the SSA tracks the molecular number of each species in the system, as opposed to the variation in concentrations in the deterministic framework, high network complexity, large separation of time-scales and high molecule numbers can result in computationally intensive executions. Another challenge is the need to simulate cell populations. In many cases, gene expression is measured for 10-100 thousand individuals sampled from an exponentially growing culture of continuously dividing cells. While the dynamics of these individual cells can be appropriately simulated by disregarding daughter cells, repeating such simulations for a fixed number of cells will not capture, for example, cell lineage dynamics. The alternative, tracking and simulating all cells within the population, is intractable beyond a few divisions due to an exponential increase in CPU demands as a function of time [44].

In order to address the issues outlined above, we present a flexible algorithm to enable simulations of heterogeneous cell population dynamics at single-cell resolution (Section 3). Deterministic and Langevin approaches to account for changes in intracellular content and the constant-number MC method [45, 46] were previously combined to simulate and analyze gene expression across cell populations [44, 47]. In these studies, extrinsic heterogeneity associated with stochastic division and partitioning mechanisms, and intrinsic heterogeneity associated with molecular reaction kinetics were considered. Specifically, Mantzaris [44, 47] demonstrated the importance of us-

ing a fully stochastic model that can quantitatively account for heterogeneity arising from noise at the single-cell and population levels (in gene expression and from the unequal partitioning of cellular content at cell division, respectively). Specifically, for a genetic network with a positive feedback architecture, the author found that accounting for both of these sources of stochasticity resulted in a shrinkage of the parameter space where bistability was observed and a decrease in average expression levels (when compared with a fully deterministic model, and a cell population balance model which accounted for population heterogeneity only in terms of unequal partitioning at cell division). Furthermore, the effect of intrinsic expression noise at the cell population level was found to be substantially different than at the single-cell level (namely bistable regime shrinkage at the population level and expansion at the single cell level), emphasizing the importance of simulating the dynamics of entire populations of growing a dividing cells to understand behavior at the population level. Our algorithm, which combines the exact SSA for single-cell molecular-level modeling and a constant-number MC method for population-level modeling, is designed to incorporate user-defined biologically relevant features, such as gene duplication and cell division, as well as single cell, lineage and population dynamics at specified sampling intervals. Additionally, the SSA, which can be replaced by approximate methods if desired, is implemented along with growth and division dynamics within a shared-memory CPU parallelization framework to reduce simulation run-times.

1.7 Summary

The transcription of DNA into mRNA and the translation of mRNA into protein are noisy processes and therefore stochastic models are used to theoretically investigate gene expression dynamics. However, often these models are analytically intractable

and therefore stochastic simulation algorithms are employed to obtain numerical solutions. Futhermore, in order to accurately simulate the dynamics of a population of growing and dividing cells, one must account for the gene expression dynamics of the daughter cells in addition to those of the mother cells. This presents a computational challenge as the number of cells then increases in an exponential fashion. Thus, we develop an algorithm which combines an exact stochastic simulation algorithm for simulating gene expression dynamics with a Monte Carlo method for simulating a fixed number of cells representing the exponentially growing population as a whole. In order to benchmark the accuracy of the algorithm, we compare simulation results with analytical solutions for several scenarios where the corresponding exact and approximate analytical solutions are available. To further benchmark the algorithm, we implement a coarse-grain two-state model of a bet-hedging yeast population in order to simulate fitness (growth-rate) dynamics under environmental stress - these results are compared with available experimental data. Finally, for the same yeast population, we simulate a fine-grain model which explicitly incorporates both gene expression and fitness in order to capture environmental effects on phenotype distributions - these simulations exemplify the utility of the present algorithm in a case where deterministic methods are unable to account for the full dynamics of the system.

In the next chapter, some of the exact and approximate analytical and simulation methods used in the modelling of stochastic gene expression are presented.

Chapter 2

Background

A small class of stochastic gene expression models can be solved using exact analytical methods (see Section 2.1.1). In many instances, the dynamics of the reaction system of interest are nonlinear and it is not possible to solve the corresponding system of equations exactly. In these cases, approximate analytical methods (see Section 2.1.2) or stochastic simulations (see Section 2.2) are required [10]. Stochastic simulation methods typically take into account explicitly the random formation and decay of single molecules and multi-component complexes and thus capture the potentially significant effects of factors that cause stochasticity in gene expression. Accordingly, stochastic models are being used increasingly in preference to deterministic models to describe biochemical network dynamics at the single-cell level [18]. As with analytical solutions, both exact and approximate methods exist, the tradeoff of the former being more computationally intense simulations and the latter in the exactness of the numerical results obtained.

Although the focus of this thesis is on the development of the cell population dynamics algorithm presented in Section 3.3 and its validation in cases where analytical solutions already exist, for completeness, here we provide an overview of some of the

exact and approximate methods that are commonly employed to obtain these analytical solutions. These analytical solutions are an essential tool for benchmarking algorithms used to simulate the dynamics of biological systems.

2.1 Analytically Solving the Chemical Master Equation

2.1.1 Exact Analytical Methods

Chemical Master Equation

A master equation (ME) is a set of first-order differential equations describing the time evolution of the probability of a system to occupy each one of a discrete set of states. These equations are useful as it is often desirable to predict the probability of a given system state without having to calculate it from repeated runs of a computer program (see Section 2.2). The ME can be expressed as the differential form of the Chapman-Kolmogorov equation for Markov processes, and is usually written in the following form when dealing with a discrete set of states k

$$\frac{dp_k(t)}{dt} = \sum_{k'} \left\{ W_{k' \to k} p_{k'}(t) - W_{k \to k'} p_k(t) \right\}. \tag{2.1}$$

Here $W_{k' \to k}$ is the transitional probability per unit time from k' to k and $p_{k'}$ the time-dependent probability associated with state k'. In this form it is clear that the ME is a gain-loss equation for the probabilities of the separate states k. The gain of state k due to the transitions from other states k' is represented by the first term, and the loss due to transitions from k into other states k' is represented by the second term [48].

In the stochastic formulation of chemical kinetics, the time evolution of a biochemi-

cal system is analytically described by a single differential-difference equation in which time and the N distinct reacting species populations, where $X_i : i \in \{1, 2, \ldots, N\}$ is the population of species $S_i : i \in \{1, 2, \ldots, N\}$, all appear as independent variables [2, 43]. The finite differential-difference equation in this context is commonly referred to as the chemical master equation (CME), and the function which satisfies it, namely $p(X_1, \ldots, X_N; t)$, is known as the grand probability function (GPF).

The GPF describes the probability that there will be X_1 molecules of S_1, X_2 molecules of $S_2, \ldots$, and X_N molecules of S_N, in a volume V at time t; the CME is the equation governing the time-evolution of this function. The state of the chemically reacting system can be described by the integer vector $\mathbf{x} = [X_1, \ldots, X_N]^T$. The system's state can change through any one of the M reactions $R_\mu : \mu \in \{1, 2, \ldots, M\}$. An R_μ reaction results in a state transition from $\mathbf{x}$ to $\mathbf{x} + \mathbf{s}_\mu$, where $\mathbf{s}_\mu$ is a vector which represents the changes in molecular species numbers that occurred as a result of R_μ. The propensity for a reaction to occur a_μ (units T^{-1}) is given by $a_\mu = c_\mu x_i$, where c_μ is a reaction parameter that characterises reaction R_μ, and x_i is the number of distinct molecular reactant combinations for reaction R_μ found to be present in V at t. The fundamental hypothesis of the stochastic formulation of chemical kinetics is that the probability that a particular combination of R_μ reactant molecules will react within the next infinitesimal time interval dt is given by $a_\mu dt$.

The CME and can be expressed as follows [2, 43]

$$\frac{\partial}{\partial t} p(\mathbf{x}, t) = \sum_{\mu=1}^{M} \left[a_\mu(\mathbf{x} - s_\mu) p(\mathbf{x} - s_\mu, t) - p(\mathbf{x}, t) a_\mu(\mathbf{x}) \right]. \tag{2.2}$$

The first and second moments of $p(\mathbf{x}, t)$ with respect to a species i are the average number $\langle n_i(t) \rangle$ of that molecule and variance $\sigma_i^2(t) = \langle n_i^2 \rangle - \langle n_i \rangle^2$. The intrinsic noise for species i is defined by $\eta_{int}(t) = \sigma_i(t) / \langle n_i(t) \rangle$ and is thus directly related to the moments of $p(\mathbf{x}, t)$ [10]. Note that another generalization of the CME is the

Fokker-Planck equation which describes the time evolution of a continuous probability distribution (see Section 2.1.2).

For example, the CME corresponding to the following birth and death process together with the associated transition probabilities: $n \to n+1 : k_c dt$, $n \to n-1 : nk_d dt$, $n \to n : 1-(k_c+nk_d)dt$, is as follows

$$\frac{\partial}{\partial t}p(n,t) = k_c p(n-1,t) + k_d(n+1)p(n+1,t) - (k_c+nk_d)p(n,t). \tag{2.3}$$

Note, since $nk_d dt$ is the probability per unit time for a degradation event given there are n molecules, the probability of the system moving from a state with n molecules to a state with $n-1$ molecules is $nk_d dt p_n$, where p_n is the time-dependent probability for having n molecules [17].

If we set $\frac{\partial}{\partial t}p(n,t) = 0$ then in some cases it is possible to directly obtain the stationary probability distribution $p^s(n)$. In the steady-state, the probability of transition from a state with n molecules to the state with $n+1$ molecules must be equal to the probability of transition from a state with $n+1$ molecules to the state with n molecules. Hence, $\frac{\partial}{\partial t}p(n,t) = 0$ is satisfied for $n \geq 0$ when

$$k_c p^s(n-1) = k_d n p^s(n) \tag{2.4}$$

and thus obtain

$$p^s(n) = \frac{k_c}{k_d n}p^s(n-1). \tag{2.5}$$

From here $p^s(n)$ can be found in an iterative fashion starting from the probability of having zero molecules in the reaction volume

$$p^s(n) = p^s(0)\prod_{m=1}^{n}\frac{k_c}{k_d m} = p^s(0)\frac{\left(\frac{k_c}{k_d}\right)^n}{n!} \tag{2.6}$$

Since the sum of the probabilities $\sum_{m=0}^{n} p^s(n)$ must equal one

$$p^s(0)\sum_{m=0}^{n}\frac{(\frac{k_c}{k_d})^m}{m!} = 1, \tag{2.7}$$

and the value of $p^s(0)$ can be obtained using the power series of the exponential function,

$$p^s(0) = e^{-k_c/k_d}. \tag{2.8}$$

The ratio $\frac{k_c}{k_d}$ is equal to the steady-state value n^s obtained from the macroscopic description and $p^s(0)$ is given by the Poisson distribution (Appendix A).

In order to construct the CME associated with the model of gene expression shown in Fig. 1.1 we proceed as follows:

To keep the CME in a compact form, we introduce a *step operator* $\mathbf{E}_i^k$ which describes the addition or removal of k molecules of species i when a particular reaction occurs. For a function $f(n_i, n_j)$ with two integer arguments, $\mathbf{E}_i^k$ increments n_i by an integer k, such that

$$\mathbf{E}_i^k f(n_i, n_j) = f(n_i + k, n_j). \tag{2.9}$$

Now we consider a change in the system due to degradation of mRNA. The change in probability is given by

$$\frac{dp(A, M, P, t)}{dt} = \delta_M(M+1)p(A, M+1, P, t) - \delta_M M p(A, M, P, t), \tag{2.10}$$

where the first and second terms describe, respectively, the flux in an out of state A, M, P, t due to the removal of one mRNA. Using the step operator and letting $p(A, M, P, t) = p$, the above equation can be expressed in compact form as

$$\frac{dp}{dt} = \delta_M(\mathbf{E}_{\mathbf{M}}^1 - 1)Mp. \tag{2.11}$$

The contributions to the probability flux due to single promoter binding is obtained in a similar way

$$\begin{aligned} \frac{dp}{dt} &= k_{on}(\mathbf{E}_{\mathbf{A}}^{-1} - 1)(1-A)p + k_{off}(\mathbf{E}_{\mathbf{A}}^{1} - 1)Ap \\ &\quad + s_A(\mathbf{E}_{\mathbf{M}}^{-1} - 1)Ap + s_P(\mathbf{E}_{\mathbf{P}}^{-1} - 1)Mp \\ &\quad + \delta_M(\mathbf{E}_{\mathbf{M}}^{1} - 1)Mp + \delta_P(\mathbf{E}_{\mathbf{P}}^{1} - 1)Pp. \end{aligned} \tag{2.12}$$

Because the CME is linear in the state variables A, M, P the moments of the probability distribution can be calculated using moment generating functions (Section 2.1.1). The first moment of p yields the average steady-state numbers of mRNA

$$\langle M^s \rangle = \langle A^s \rangle \frac{s_A}{\delta_M} \tag{2.13}$$

and protein molecules

$$\langle P^s \rangle = \langle M^s \rangle \frac{s_P}{\delta_P} = \langle A^s \rangle \frac{s_A s_P}{\delta_M \delta_P}, \tag{2.14}$$

where $\langle A^s \rangle = k_{on}/(k_{on} + k_{off})$ is the average probability of the promoter being active. The variances in the state variables can be obtained from the second moment of p, and thus an expression for the intrinsic noise in the steady-state protein abundance can be determined. Assuming that the protein decay is slow compared to the promoter kinetics ($\delta_P << k_{on} + k_{off}$) the coefficient of variation in steady-state protein number can be expressed as

$$CV_P^s = \left(\frac{1}{P^s} + \frac{1}{1+\phi} \frac{1}{M^s} \right) + \frac{\delta_M k_{off}/k_{on}}{(1+\phi)(k_{on} + k_{off})}, \tag{2.15}$$

where $\phi = \delta_M/\delta_P$ [10, 30]. Note that the first term in Eq. (2.15) arises from mRNA and protein kinetics and the second term from kinetics of the promoter [10]. Since $(\sigma_M^s)^2 = \langle M^s \rangle$ as described by Poisson statistics, the coefficient of variation in steady-state mRNA number CV_M^s is simply $1/\langle M^s \rangle$.

Although there are a few specific cases were the CME can be solved exactly, in general analytical solving or numerically simulating the master equation for a system of realistic size and complexity is not possible, and the only practical approach for gaining insight into system's dynamics is via a MC simulation of the CME (see Section 2.2) [2, 16, 43].

Moment-Generating Functions

The moments of a random variable can be calculated via moment-generating functions [49]. Noting that the expected value $\langle X \rangle$ of a continuous random variable X is given by

$$\langle X \rangle = \int_{-\infty}^{\infty} xp(x)dx, \tag{2.16}$$

where $p(x)$ is the probability density, and that by definition the moment-generating function $M_x(t)$ of X is the expected value of the function e^{tx}, where t is an auxiliary variable,

$$M_x(t) = \left\langle e^{tx} \right\rangle \tag{2.17}$$

and so,

$$M_x(t) = \int_{-\infty}^{\infty} e^{tx} p(x)dx. \tag{2.18}$$

The moments

$$\langle X^n \rangle = \int_{-\infty}^{\infty} p(x)x^n dx \tag{2.19}$$

can be written as

$$\langle X^n \rangle = lim_{t\to 0} \int_{-\infty}^{\infty} dxp(x) \left(\frac{d}{dt}\right)^n (e^{tx}) = lim_{t\to 0} \left(\frac{d}{dt}\right)^n M_x(t). \tag{2.20}$$

Thus, the moment $\langle X^n \rangle$ is the limit as $t \to 0$ of the n^{th} derivative of $M_x(t)$ with respect to the auxiliary variable t.

For the system described by Eq. (2.3), for a single species X_1 with n molecules per V, the moment generating function is

$$M_x(s,t) = \sum_{n}^{\infty} \prod s^n p(n,t), \tag{2.21}$$

where s is an integer [49]. Since

$$M_x(s,t)|_{s=1} = \sum_{n=0}^{\infty} p(n,t) = 1, \tag{2.22}$$

the first derivative generates the first moment

$$\partial_s M_x(s,t)|_{s=1} = \sum_{n=0}^{\infty} n(t)p(n(t),t) = \langle n \rangle = \mu_n \tag{2.23}$$

and the second derivative the second moment

$$\begin{aligned} \partial_{ss} M_x(s,t)|_{s=1} &= \sum_{n=0}^{\infty} n(t)^2 p(n(t),t) - \sum_{n=0}^{\infty} n(t)p(n(t),t) \\ &= \left\langle n(t)^2 \right\rangle - \langle n(t) \rangle = \langle n(t) \rangle_f^2 , \end{aligned} \tag{2.24}$$

where $\langle n(t)^2 \rangle_f$ is known as the *factorial moment*. Since $\langle n \rangle_f = \langle n \rangle$, one obtains

$$\partial_{ss} M_x(s,t)|_{s=1} = \sigma_n^2 = \langle n \rangle . \tag{2.25}$$

Note μ_s and σ_n^2 are as expected for this birth-death process (i.e. Poissonian - Section 2.1.1).

The moment-generating function can be used to find the mean and variance for the gene expression model presented in Fig. 1.1 as well as an approximative protein distribution as a function of time. The approximation is based on the assumption that the degradation of mRNA is fast compared to the degradation of proteins (i.e. $\delta_M/\delta_P \gg 1$). Consequently, the dynamics of mRNA are at the steady-state for most of a protein's lifetime. The essential steps of the derivation are as follows (see supplementary materials in [50] for a complete derivation):
Eq. (2.12), the CME describing the probability of having M and P for this system at time t, can also be written as

$$\begin{aligned} \frac{\partial p_{M,P}}{\partial t} &= s_A(p_{M-1,P} - p_{M,P}) + s_P M(p_{M,P-1} - p_{M,P}) \\ &\quad + \delta_M[(M+1)p_{M+1,P} - M p_{M,P}] \\ &\quad + \delta_P[(M+1)p_{M,P+1} - P p_{M,P}]. \end{aligned} \tag{2.26}$$

By defining the generating function $F(z^{'}, z)$, by $F(z^{'}, z) = \sum_{M,P} (z^{'})^M z^P p_{M,P}$, Eq. (2.26)

can be converted into a first-order partial differential equation, namely

$$\frac{1}{v}\frac{\partial F}{\partial \tau} + \frac{\partial F}{\partial v} - \gamma\left[b(1+u) - \frac{u}{v}\right]\frac{\partial F}{\partial u} = a\frac{u}{v}F, \tag{2.27}$$

where $a = s_A/\delta_P$, $b = s_P/\delta_M$, $\gamma = \delta_M/\delta_P$, and $\tau = \delta_P t$, and where $u = z^{'} - 1$ and $v = z - 1$ [50]. Using the method of characteristics, Eq. (2.27) can be solved to obtain

$$\frac{du}{dv} = -\gamma\left[b(1+u) - \frac{u}{v}\right]. \tag{2.28}$$

Integration of Eq. (2.28) yields the solution

$$u(v) = e^{-\gamma b v} v^{\gamma}\left[C - b\gamma\int^{v} dv^{'}\frac{e^{\gamma b v^{'}}}{v^{'\gamma}}\right] \tag{2.29}$$

for a constant C. By Taylor expansion of $e^{\gamma b v}$ such that $e^{\gamma b v} = \sum_n (\gamma b v)^n/n!$ the integral in Eq. (2.29) can be evaluated, and if Stirling's approximation is subsequently applied, $u(v)$ is found for $\gamma >> 1$ to obey

$$u(v) \cong \left(u_0 - \frac{b v_0}{1 - b v_0}\right) e^{-\gamma b(v - v_0)}\left(\frac{v}{v_0}\right)^{\gamma} + \frac{bv}{1 - bv} \tag{2.30}$$

or

$$u(v) \cong \frac{bv}{1 - bv}. \tag{2.31}$$

When $\gamma >> 1$, u tends rapidly to a fixed function of v and the generating function describing the distribution of proteins can be obtained from Eq. (2.27)

$$\frac{dF}{dv} \cong \frac{ab}{1 - bv}F. \tag{2.32}$$

Integrating Eq. (2.32) yields the probability distribution for protein number as a function of time

$$F(z,\tau) = \left[\frac{1 - b(z-1)e^{-\tau}}{1 + b - bz}\right]^{a}. \tag{2.33}$$

By definition of a generating function, expanding $F(z)$ in z yields

$$P_n(\tau) = \frac{\Gamma(a+n)}{\Gamma(n+1)\Gamma(a)}\left[\frac{b}{1+b}\right]^{n}\left[\frac{1 + be^{-\tau}}{1+b}\right]^{a} \times {}_2F_1\left[-n, -a, 1 - a - n; \frac{1+b}{e^{\tau} + b}\right], \tag{2.34}$$

where ${}_2F_1$ and Γ are the hypergeometric and the gamma function, respectively [50]. When the initial number of proteins n is set to zero, the mean and noise of the process are described, respectively by

$$\mu_P = ab(1 - e^{-\tau}), \tag{2.35}$$

$$\eta_P = \left[(1 + b + be^{-\tau})/ab(1 - e^{-\tau})\right]^{1/2}. \tag{2.36}$$

Note that these results are in agreement with the steady-state results found using the CME approach (Section 2.1.1) when the conditions $\tau >> 1$ (i.e. the system has reached steady-state) and $\delta_M >> \delta_P$ are satisfied.

2.1.2 Approximate Analytical Methods

Linear Noise Approximation

The LNA [51, 52] is an efficient method for estimating (and in some cases it is exact, e.g., the CV expression presented in this section for the model of single gene expression shown in Fig. 1.1) the internal and external variability in nonlinear oscillatory systems (e.g. genetic networks [53–55]). Furthermore, it allows the possibility of calculating the intrinsic noise in systems with nonlinear reaction rates where moment generating functions cannot be used, or where due to feasibility, stochastic simulations may not be desirable [10].

In the LNA, the discrete state space (number of molecules) is smoothed into a continuum (macroscopic concentrations) and additional terms describing the fluctuations about the macroscopic trajectory are calculated. Specifically, the number of molecules $n_i(t)$ of a species i is approximated as a continuous variable in terms of the concentration $c_i(t)$, and a term $\alpha_i(t)$ describing the deviation from $c_i(t)$. Since the impact of this fluctuation is expected to scale with the square root of system size Ω,

the number of molecules is approximated by

$$n_i(t) \approx \Omega c_i(t) + \sqrt{\Omega}\alpha_i(t). \tag{2.37}$$

The LNA assumes that the step operator (Eq. (2.9)) associated with the transition probabilities in the CME is well described by a Taylor series for large Ω. The basis for this assumption lies in the fact that an integer change in molecule number will have a negligible effect on the concentration in the macroscopic limit. If k molecules are added to the system, the value of α_i in Eq. (2.37) increases by $k/\sqrt{\Omega}$. Therefore, for large Ω, a single reaction brings about only a small change in concentration and the operator E_i^k is well approximated by a truncated Maclaurin series

$$E_i^k \approx \left[1 + \frac{k}{\sqrt{\Omega}}\partial_i + \frac{k^2}{2\Omega}\partial_i^2\right], \tag{2.38}$$

where $\partial_i = \partial/\partial\alpha_i$ and $\partial_i^2 = \partial^2/\partial\alpha_i^2$. From here it is necessary to derive an equation that describes the fluctuations $\vec{\alpha}(t) = \alpha_1, \ldots, \alpha_d$, where d is the number of distinct species in the system, in order to calculate the average fluctuations $\langle\alpha_i\rangle$ and the variances $\langle\alpha_i\alpha_j\rangle$. To do this the probability density

$$\prod = \prod(\vec{\alpha}, t) = \Omega^{d/2} p(\mathbf{n}, t) \tag{2.39}$$

is introduced, where the scaling $\Omega^{d/2}$ ensures that $\prod$ is normalized appropriately. If Eqs. (2.37) and (2.38) are inserted into the CME, and the terms in which Ω appears with the same order are collected [52], an approximation of the equations that governs the evolution of $\prod$ can be obtained [10]. The equation that governs the evolution of $\prod(\vec{\alpha}, t)$ contains terms multiplied by $1/\sqrt{\Omega}$; the general form of a linear Fokker-Planck equation (FPE) results when these terms are collected

$$\frac{\partial \prod}{\partial t} = -\sum_{i,j} A_{ij}\partial_i(\alpha_j \prod) + \frac{1}{2}\sum_{i,j} B_{ij}\partial_{ij} \prod, \tag{2.40}$$

where A_{ij} are the drift (or dissipation) terms, and B_{ij} are the diffusion terms. Note that the matrix $\mathbf{A}$ reflects the local stability of the macroscopic system to small perturbations and the matrix $\mathbf{B}$ describes the strength of the local fluctuations. Also, since the time-dependent coefficient matrices $\mathbf{A}$ and $\mathbf{B}$ are independent of the fluctuations $\vec{\alpha}$, which appear only linearly in Eq. (2.40), the solution $p(\mathbf{n}, t)$ is normal and completely characterized by the first two moments [10].

If Eq. (2.40) is multiplied by α_i and then is integrated over all $\vec{\alpha}$, evolution equations describing the the means $\langle\alpha_i\rangle$ and the variances $\langle(\alpha_i - \langle\alpha_i\rangle)(\alpha_j - \langle\alpha_j\rangle)\rangle$ can be obtained. For the means

$$\frac{d}{dt}\langle\vec{\alpha}\rangle = \mathbf{A}\langle\vec{\alpha}\rangle \tag{2.41}$$

and for the variances $C_{ij} = \langle\alpha_i\alpha_j\rangle$, if one assumes that $\langle\alpha\rangle = 0$, a set of equations are given in compact matrix-vector form

$$\frac{d\mathbf{C}}{dt} = \mathbf{AC} + \mathbf{CA}^T + \mathbf{B}. \tag{2.42}$$

The following example obtained from Scott *et al.* [10] demonstrates how the LNA can be used to calculate the intrinsic noise in the expression of a single gene shown in Fig. 1.1. When mRNA synthesis occurs with a constant rate s_A, the CME given in Eq. (2.12) can be expressed as follows

$$\begin{aligned}\frac{dp}{dt} &= s_A A(\mathbf{E}_M^{-1} - 1)p + s_P(\mathbf{E}_P^{-1} - 1)Mp \\ &\quad + \delta_M(\mathbf{E}_M^1 - 1)Mp + \delta_P(\mathbf{E}_P^1 - 1)Pp.\end{aligned} \tag{2.43}$$

Eq. (2.43) can then be expressed in terms of $\prod = \prod(\alpha_1, \alpha_2, t)$ by applying the chain rule [52],

$$\frac{dp}{dt} = \frac{1}{\sqrt{\Omega}}\left[\frac{1}{\sqrt{\Omega}}\frac{\partial\prod}{\partial t} - \frac{dM}{dt}\frac{\partial\prod}{\partial\alpha_M} - \frac{dP}{dt}\frac{\partial\prod}{\partial\alpha_P}\right]. \tag{2.44}$$

Inserting Eq. (2.37) in the form $M = \Omega([M] + \alpha_M/\sqrt{\Omega})$ and $P = \Omega([P] + \alpha_P/\sqrt{\Omega})$, and the approximate step-operator (Eq. (2.38)), into Eq. (2.43), yields an intermittent

equation

$$\begin{aligned}\frac{1}{\sqrt{\Omega}}\frac{\partial \prod}{\partial t} &= \frac{dM}{dt}\partial_M \prod + \frac{dP}{dt}\partial_P \prod + s_A\left[\frac{\partial_M^2}{2\sqrt{\Omega}} - \partial_M\right]\prod \\ &+ \left(s_P\left[\frac{\partial_P^2}{2\sqrt{\Omega}} - \partial_P\right] + \delta_M\left[\partial_M + \frac{\partial_M^2}{2\sqrt{\Omega}}\right]\right)\left(M + \frac{\alpha_M}{\sqrt{\Omega}}\right)\prod \\ &+ \delta_P\left[\partial_P + \frac{\partial_P^2}{2\sqrt{\Omega}}\right]\left(P + \frac{\alpha_P}{\sqrt{\Omega}}\right)\prod, \end{aligned} \tag{2.45}$$

where M and P are now, until the end of this section, being used to denote $[M]$ and $[P]$. Isolating terms entering Eq. (2.45) independent of Ω, corresponding to zeroth order in $1/\sqrt{\Omega}$, yields

$$\frac{dM}{dt}\partial_M \prod + \frac{dP}{dt}\partial_P \prod = [s_A - \delta_M M]\,\partial_M \prod + [s_P M - \delta_P P]\,\partial_P \prod \tag{2.46}$$

Equating the coefficients of $\partial \prod / \partial \alpha_i$ then yields the macroscopic rate equations in Eqs. (1.1)-(1.2).

A first order approximate evolution equation for $\prod(\alpha_M, \alpha_P)$ is obtained by isolating terms entering Eq. (2.45) with first order in $1/\sqrt{\Omega}$

$$\begin{aligned}\frac{d\prod}{dt} &= \frac{s_A}{2}\partial_M^2 \prod + \frac{s_P M}{2}\partial_P^2 \prod - s_P \partial_P(\alpha_M \prod) + \delta_M \partial_M(\alpha_M \prod) \\ &+ \frac{\delta_M M}{2}\partial_M^2 \prod + \frac{\delta_P P}{2}\partial_P^2 \prod + \delta_P \partial_P(\alpha_P \prod), \end{aligned} \tag{2.47}$$

which is a FPE with the coefficient matrices

$$\mathbf{A} = \begin{bmatrix} -\delta_M & 0 \\ s_P & -\delta_P \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} s_A + \delta_M M & 0 \\ 0 & s_P M + \delta_P P \end{bmatrix}.$$

The intrinsic noise in the steady-state can be found by setting the LHS of Eq. (2.42) to zero and solving for $\langle \alpha_i \alpha_j \rangle$ yielding the following set of equations

$$-2\delta_M \langle \alpha_M \alpha_M \rangle + s_A + \delta_M M^s = 0,$$

$$-(\delta_M + \delta_P)\langle \alpha_M \alpha_P \rangle + s_P \langle \alpha_M \alpha_M \rangle = 0,$$

$$-2\delta_P \langle \alpha_P \alpha_P \rangle + s_P \langle \alpha_M \alpha_P \rangle + s_P M^s + \delta_P P^s = 0, \tag{2.48}$$

the solution of which is found to be

$$\mathbf{C}^s = \begin{bmatrix} M^s & \frac{P^s}{1+\phi} \\ \frac{P^s}{1+\phi} & \left(P^s + \frac{(P^s)^2}{M^s(1+\phi)}\right) \end{bmatrix},$$

where $\phi = \delta_M/\delta_P$. Using the definition $\langle \alpha_P \alpha_P \rangle^s = C^s_{22}$ the coefficient of variation for the protein concentration is calculated to be

$$CV = \frac{1}{\Omega}\left[\frac{1}{P^s} + \frac{1}{M^s(1+\phi)}\right], \tag{2.49}$$

in agreement with the exact analytical methods (Section 2.1.1), in the limit of fast promoter kinetics.

Stochastic Differential Equations

A set of stochastic differential equations (SDEs) can also be used to model biochemical reactions [15, 56]. A SDE can be thought of as a way of constructing a realisation of X from a realisation of Brownian motion (Wiener process) $W(t) = W$ [16]. Although like ODEs this approach is based on the law of mass action, here the stochastic Itô process (stochastic processes described through Itô integrals, i.e. integrals of the form $Y(t) = \int_o^t H(s)dW(s)$) is used to describe the time evolution of the system.

The following example of how to obtain an analytical solution to a one-dimensional SDE was obtained from [57–59]:

We begin by assuming that we have an ODE, namely

$$\frac{dX}{dt} = \mu X, \quad \mu X \in \Re, \tag{2.50}$$

and we want to incorporate stochasticity into the constant μ. This can be accomplished by incorporating a noise term ξ_t with a parameter $\nu \in \Re$ to μ, which results

in a SDE of the form

$$dX = (\mu + \nu\xi)Xdt = \mu Xdt + \nu X\xi dt, \quad \mu X, \nu X \in \Re. \tag{2.51}$$

Since the random fluctuation at a certain moment can often be described by a Gaussian random variable, ξ_t is often referred to as a Gaussian noise term (or process). However, from a mathematical point of view, as the mathematical formalism to handle a Gaussian noise process as part of an ODE does not exist, one must resort to using fractions of a suitable Gaussian process as a noise term, namely Brownian motion [57]. Thus Eq. (2.51) can be expressed as follows

$$\frac{dX}{dt} = \mu X + \nu X dW, \tag{2.52}$$

where dW is now a fraction of Brownian motion. In order to obtain an analytical solution to the case discussed above, we proceed as follows considering an Itô process of the form

$$X = X(0) + \int_0^t \mu X(s)ds + \int_0^t \nu X(s)dWds, \tag{2.53}$$

which is equivalent to Eq. (2.52). Rearranging Eq. (2.52) we obtain

$$\frac{dX}{X} = \mu dt + \nu dWdt, \tag{2.54}$$

hence,

$$\int_0^t \frac{dX}{X} = \int_0^t \mu dt + \int_0^t \nu dWdt = \mu t + \nu W. \tag{2.55}$$

Evaluating the integral on the L.H.S. of Eq. (2.55) using the Itô formula yields [57]

$$\frac{dX}{X} = d\ln(X) + \frac{1}{2}\nu^2 dt. \tag{2.56}$$

From Eqs. (2.55) and (2.56) we obtain

$$\int_0^t d\ln(X) + \int_0^t \frac{1}{2}\nu^2 ds = \mu t + \nu W$$

$$\ln X - \ln X(0) + \frac{1}{2}\nu^2 t = \mu t + \nu W$$
$$\ln \frac{X}{X(0)} = \mu t + \nu W - \frac{1}{2}\nu^2 t$$
$$\ln \frac{X}{X(0)} = (\mu - \frac{1}{2}\nu^2)t + \nu W \tag{2.57}$$

or equivalently,

$$X = X(0) \exp\left[(\mu - \frac{1}{2}\nu^2)t + \nu W\right]. \tag{2.58}$$

Similarly, multi-dimensional SDEs can be constructed by adding noise terms in a set of ODEs. Here we consider the model of gene expression depicted in Fig. 1.1, again where transcription occurs with a constant rate s_A such that the CME describing the system is given by Eq. (2.43). The Fokker-Planck approximation corresponding to Eq. (2.43) is as follows [60, 61]

$$\begin{aligned}\frac{\partial p_{M,P}}{\partial t} &= -\frac{\partial[(s_A - \delta_M M)p_{M,P}]}{\partial M} - \frac{\partial[(s_P - \delta_P P)p_{M,P}]}{\partial P} \\ &\quad + 2^{-1}\frac{\partial^2[(s_A + \delta_M M)p_{M,P}]}{\partial M^2} + 2^{-1}\frac{\partial_P^2[(s_P M + \delta_P P)p_{M,P}]}{\partial P^2}.\end{aligned} \tag{2.59}$$

Eq. (2.59) is equivalent to the following set of nonlinear SDEs (or Chemical Langevin Equations) [62]

$$\begin{aligned}\frac{dM}{dt} &= s_A - \delta_M M + \sqrt{s_A + \delta_M M}\eta_{M,t}, \\ \frac{dP}{dt} &= s_P M - \delta_P P + \sqrt{s_P M + \delta_P P}\eta_{P,t},\end{aligned} \tag{2.60}$$

where $\eta_{M,t}$ and $\eta_{P,t}$ are Gaussian white noise processes. If we are close to the steady-state, then for the system described by Eq. (2.60) $s_A \approx \delta_M \langle M^s \rangle$ and $s_P \langle M^s \rangle \approx \delta_P \langle P^s \rangle$ and therefore can be described by the following system of linear-approximations

$$\begin{aligned}\frac{dM}{dt} &= s_A - \delta_M M + \sqrt{q_M}\eta_{M,t}, \\ \frac{dP}{dt} &= s_P M - \delta_P P + \sqrt{q_P}\eta_{P,t},\end{aligned} \tag{2.61}$$

where $q_M = 2s_A$ and $q_P = 2s_P M^s$. The general integral solution to Eq. (2.61) is given by

$$\begin{aligned} M &= M_0 e^{-\delta_M t} + e^{-\delta_M t} \int_0^t (s_A + \eta_{M,s}) e^{\delta_M s} ds, \\ P &= P_0 e^{-\delta_P t} + e^{-\delta_P t} \int_0^t (s_P M_s + \eta_{P,s}) e^{\delta_P s} ds. \end{aligned} \tag{2.62}$$

Defining the Fourier transforms of the variables M and P as $M_w^s = \int_{-\infty}^{\infty} M e^{-iwt} dt$ and $P_w^s = \int_{-\infty}^{\infty} P e^{-iwt} dt$, then Eq. (2.62) can be rewritten in the frequency domain as follows

$$\begin{aligned} M_w^s &= 2\pi s_A (\delta_M + iw)^{-1} (\delta(w) + \eta_{M,w}), \\ P_w^s &= (\delta_P + iw)^{-1} (s_P 2\pi s_A (\delta_M + iw)^{-1} (\delta(w) + \eta_{M,w}) + \eta_{P,w}), \end{aligned} \tag{2.63}$$

where the noise terms are defined as $\eta_{M,w} = \int_{-\infty}^{\infty} \eta_{M,t} e^{-iwt} dt$ and $\eta_{P,w} = \int_{-\infty}^{\infty} \eta_{P,t} e^{-iwt} dt$. Using Eq. (2.63) the mean steady-state values for the number of mRNA and protein molecules can be obtained [62]

$$\begin{aligned} \langle M^s \rangle &= s_A \int_{-\infty}^{\infty} (\delta_M + iw)^{-1} \delta(w) e^{iwt} dw = \frac{s_A}{\delta_M}, \\ \langle P^s \rangle &= s_P \int_{-\infty}^{\infty} (\delta_P + iw)^{-1} (s_A (\delta_M + iw)^{-1} \delta(w)) e^{iwt} dw = \frac{s_A s_P}{\delta_M \delta_P}, \end{aligned} \tag{2.64}$$

which, when the gene is constitutively expressed, are in agreement with the analytical results found in Section 2.1.1. Similarly, the stationary state variance is found to be

$$\begin{aligned} \left\langle (M^s)^2 \right\rangle - \langle M^s \rangle^2 &= q_M/(2\delta_M), \\ \left\langle (P^s)^2 \right\rangle - \langle P^s \rangle^2 &= q_P/(2\delta_P) + s_P^2 q_M/(2\delta_M \delta_P (\delta_M + \delta_P)). \end{aligned} \tag{2.65}$$

The CV can be obtained from Eqs. (2.64) and (2.65) [62]

$$\begin{aligned} CV_M^s &= \delta_M / s_M = \frac{1}{\langle M^s \rangle}, \\ CV_P^s &= \delta_M \delta_P / (s_A + s_P) + \delta_M \delta_P / (s_A (\delta_M + \delta_P)) \\ &= \frac{1}{\langle P^s \rangle} + \delta_P / s_A, \end{aligned} \tag{2.66}$$

and is in agreement with the solutions found using exact methods in Section 2.1.1.

In the following sections we present an overview of the exact and approximate methods commonly employed to simulate models of gene expression.

2.2 Stochastically Simulating the Chemical Master Equation

2.2.1 Exact Simulation Methods

Direct and First-Reaction Methods

The physical basis of the stochastic formulation of chemical kinetics is a consequence of the fact that collisions in a system of molecules in thermal equilibrium are essentially a random process. This stochasticity is correctly accounted for by the Gillespie stochastic simulation algorithm (SSA) [2, 43], a Monte Carlo (MC) procedure to numerically simulate the time evolution of chemical and biochemical reaction systems. While based on an assumption of intracellular homogeneity and mass-action kinetics, it is the standard for simulations of gene expression. This method is extremely important as it is rarely possible to solve the chemical master equation (CME) (see Section 2.1.1), either analytically or numerically, except for simplest of chemical systems [63].

The SSA tracks the molecular number of each species in the system as opposed to the variation in concentrations in the deterministic framework. The SSA however does not try to simulate the CME numerically but rather the very Markov process that the CME describes using rigorously derived MC techniques. The SSA is fully equivalent to the CME, even though the CME itself is never used [2, 43].

In the *direct method* Gillespie SSA, M chemical reactions $\{R_1, \ldots, R_M\}$ with re-

action parameters $c_1, ..., c_M$ among N chemical species $X_1, ..., X_N$, are simulated one reaction event at a time. The fundamental hypothesis of the stochastic formulation of chemical kinetics is that the average probability of a given reaction i occurring in the next infinitesimal time interval dt is given by $a_i dt$, where a_i is the reaction propensity obtained by multiplying c_i by the number of reactants (for first order reactions) or reactant combinations (for second order and higher reactions) h_i available for reaction R_i. The next reaction to occur (index μ) and its timing τ are determined by calculating the M reaction propensities $a_1, ..., a_M$ to obtain an appropriately weighted probability for each reaction. The SSA determines when ($\tau = \ln(1/r_1)/a_0$) and which ($\min\{\ \mu \mid \sum_{i=1}^{\mu} a_i \geq r_2 a_0\}$) reaction will occur next, using uniformly distributed random numbers r_1 and r_2, and the sum of the reaction propensities a_0. See Section 3.1 for an algorithmic implementation of the Gillespie direct method.

One alternative, but equivalent method to the direct SSA method, is the *first-reaction method*. In this method, the τ for each of the M reaction channels is computed directly and then the reaction with the smallest τ is selected as the next reaction to occur. As the τ for each i of the M reactions must be computed (via $\tau = \ln(1/r_1)/a_i$) at each SSA step, this method is computationally less efficient than the direct method when $M \geq 3$ (since in this case more than the two random numbers required in the direct method need to be generated) and is therefore not commonly employed [3, 63].

We note that the SSA can be extremely computationally intensive since the step size τ becomes very small when the total number of molecules is high or the fastest reaction occurs on a time-scale that is much shorter than the time-scale of interest. Thus, the exact nature of the SSA comes at the expense of efficiency and it can be incapable of simulating larger systems because of computational inefficiency [3, 42]. It is therefore useful to develop techniques that can be used to speed up the

simulation. This can be done, for example, using approximate simulation methods (see Section 2.2.2). Additionally, since many independent realisations are required in order to compute population statistics, parallel computing can be used to further optimize simulation run-times (see Section 3.3).

Next-Reaction Method

The *next-reaction method* is a reformulation of the SSA which significantly decreases simulation times as compared with the direct and first-reaction methods for a large number of molecular species N and reactions M [64]. In this method, as in the first-reaction method, the putative times τ_i for each reaction to occur (i.e. a time each reaction would occur if no other reaction occurred first) are generated. However, the next-reaction method does away with each of the following three steps (each of which takes a time proportional to M) which occur during each iteration of the Gillespie loop in the first-reaction method: (1) updating all M of the a_is; (2) generating M τ_is; (3) identifying the smallest τ_i. Specifically, each τ_i is stored along with each a_i and is recalculated only if it changes as a result of the previously selected reaction. This allows for only one uniform random number to be used in each iteration of the Gillespie loop in contrast to the two required by the direct method or the M required by the first-reaction method.

Although the next-reaction method can be significantly faster than the direct and first-reaction methods, it is more challenging to implement algorithmically [3]. The reason for this is because in order to know whether a_i has changed or not without recalculating it and comparing it to its old value, one needs to analyze the set of reactions beforehand and determine which reactions change which a_is. Specifically, a data structure, known as a dependency graph, needs to be introduced in order to allow the minimum number of a_is to be updated. Furthermore, one also needs

to generate an indexed priority queue to store the time and index of each reaction channel [64].

2.2.2 Approximate Simulation Methods

Tau-Leaping Method

The stochastic simulation methods described in Section 2.2.1, although essentially exact procedures for numerically simulating the time evolution of well-stirred chemically reacting system, are often very demanding in terms of the computer time required to simulate a desired amount of system time [18, 63]. The *tau-leaping* method [63], can in some cases, produce significant reductions in simulation times with acceptable losses in accuracy. This method accelerates the SSA by calculating a time step τ which advances the system through possibly many reaction events. A *leap condition*, an accuracy-assuring restriction, requires that τ be small enough that no propensity function changes by a significant amount during the infinitesimal time interval $[t, t+\tau)$.

More specifically, in order to implement the tau-leaping method we again consider a system of N species that react through M reaction channels, where the number of i species molecules at time t is a random variable described by the vector $\mathbf{X}(t) = [X_1(t), \ldots, X_N(t)]$. Here, a propensity function $a_j(\mathbf{X})$ and a state change vector $\mathbf{v}_j$ specify the dynamics of each reaction channel R_j. The quantity $a_j(\mathbf{X})\tau$ represents the probability that a reaction of type R_j will occur in the time interval $[t, t+\tau)$, and the ith element $v_{j,i}$ of $\mathbf{v}_j$ denotes the change in the number of i species molecules as a result of one R_j reaction. A value for tau is chosen such that $|a_j(\mathbf{X}+\mathbf{v}_j) - a_j(\mathbf{X})|$ is 'effectively infinitesimal' [63]. The basic tau-leaping formula can be expressed by [3]:

$$\mathbf{X}(t+\tau) = \mathbf{x} + \sum_{j=1}^{M} Po_j(a_j(\mathbf{x})\tau)\mathbf{v}_j, \tag{2.67}$$

where $\mathbf{x} = \mathbf{X}(t)$ and $Po_j(\lambda)$ is a Poisson random variable.

Numerically Simulating Stochastic Differential Equations

Biochemical reactions can also be modelled stochastically using a system of stochastic differential equations (SDEs) [57]. Although the SDE model sacrifices discreteness, it takes into account natural fluctuations and is therefore able to capture the dynamics of certain systems in cases where the ODE model fails to capture the temporal behavior (e.g. the Lotka-Volterra predator-prey system). In the physical sciences, SDEs are commonly written as *Langevin equations* in the form

$$\frac{dx}{dt} = f(x, \mu) + \xi(t) \tag{2.68}$$

where $f(x, \mu)$ is a linear or non-linear function, μ is a parameter, and $\xi(t) = \xi$ is some noise process. As random fluctuations at a given moment can often be described by a Gaussian random variable, ξ is referred to as a 'Gaussian noise process'.

SDEs are rarely explicitly solvable (see Section 2.1.2 for a simple case which permits analytical solution) and therefore numerical methods for solving these equations have been developed. The most common approach used in solving SDEs numerically involves the simulation of sample paths over discretized times. The *Euler-Maruyama method*, a generalization of the *Euler method* for solving ODEs to SDEs, is one such method for obtaining the approximate numerical solution to a SDE [57].

As an example of the Euler-Maruyama method the SDE, Eq. (2.52) is considered on the discrete time interval $[t_0, T]$ such that $t_0 = \tau_0 < \tau_1 < \ldots < \tau_n < \ldots < \tau_N = T$ [57]. Here W represents Brownian motion. If the process has the initial value X_0 then an Euler-Maruyama approximation of the process X_t is a continuous time stochastic process Y_t satisfying the iterative equation

$$Y_{n+1} = Y_n + \mu(Y_n)(\tau_{n+1} - \tau_n) + \nu(Y_n)(W_{n+1} - W_n), \tag{2.69}$$

with the initial value $Y_0 = X_0$, where $n = 0, 1, 2, \ldots, N-1$. Eq. (2.69) can be expressed in a simplified form as

$$Y_{n+1} = Y_n + \mu(Y_n)\Delta\tau + \nu(Y_n)\Delta W, \tag{2.70}$$

where $\Delta\tau = \tau_{n+1} - \tau_n$ and $\Delta W = W_{n+1} - W_n$. The increments of Brownian motion ΔW are independent and identically distributed normal random variables with expected value zero and variance $\sigma^2 = (T - t_0)/N > 0$ (for N equal subintervals). Although the Euler-Maruyama method is defined to be a continuous time stochastic process, only values at the discrete times are evaluated and values not belonging to the discretization are usually determined by interpolation [57].

2.2.3 Stochastic Simulation Algorithm Augmentations

Several augmentations to the SSA have been made in recent years. For example, Roussel and Zhu [14] extended the the SSA to include time delayed reactions (*delayed SSA*). While also accounting for the stochastic nature of chemical reactions, the delayed-SSA, unlike the original SSA, can model multi-step processes in a single step by accounting for the time duration required for these events to be completed. Specifically, delayed output events (e.g. production of a protein from an mRNA) are stored on a waitlist, each to be released some time after the reaction that produced them occurred.

Another augmentation to the SSA is the incorporation of extrinsic noise, that is the variability in factors considered to be external to the system of interest (discussed briefly in Section 1.4), into the simulation framework. In [13] the authors show how the total variation in the level of expression of a given gene can be decomposed into its intrinsic and extrinsic components and integrated within a single framework. In models of gene expression, extrinsic-noise sources such as cell-to-cell variation in

growth rates and gene expression capacity can be incorporated by varying the rate constants (i.e. drawing the values for rate constants from appropriate distributions) associated with these events [6].

Stochastic simulation software packages based on the original or augmented SSA have started to appear in the literature. One example is *CellLine*, a simulator of the dynamics of gene regulatory networks in cells of a lineage, that I co-developed [25]. From user-defined reactions and initial substance quantities, *CellLine* can generate cell lineages, where each cell's dynamics is driven by the delayed SSA. Furthermore, cells of the lineage can be individually subject to 'perturbations', such as gene deletion, duplication and mutation. External interventions, such as adding or removing a substance at a given time can also be specified. These augmentations allow the modelling of, for example, cell differentiation lineages. In the cell population dynamics algorithm presented in Section 3.3, the time to next division, which can be calculated for each cell the moment it is 'created', is also a delayed event that can be listed on a stack as is done in the delayed SSA for delayed reaction events.

In the next chapter, an algorithm we developed for simulating stochastic effects on population dynamics is presented.

Chapter 3

Algorithm

The Gillespie stochastic simulation algorithm (SSA) is a Monte Carlo (MC) simulation of the chemical master equation and is the standard for simulating models of stochastic gene expression [2, 43]. The constant-number MC method, used in the simulation of particle systems to keep the number of particles constant throughout the simulation, can be employed to simulate the time-dependent statistical characteristics of growing and dividing cell populations [45, 46]. In this chapter, we begin by presenting implementations of these two methods (Sections 3.1 and 3.2, respectively). Then in Section 3.3, we present a parallel algorithm that combines the SSA and the constant-number MC method within a single framework, in order to stochastically simulate the heterogeneous genotypic and phenotypic dynamics of a cell population.

3.1 Implementation: Stochastic Simulation Algorithm

The direct method Gillespie SSA discussed in Section 2.2.1 was implemented via the following pseudocode [2, 43]:

1: **if** $t < t_{end}$ and $a_0 = \sum_{i=1}^{M} a_i \neq 0$ **then**
2: **for** $i = 1, M$ **do**
3: Calculate a_i and $a_0 = \sum_{v=1}^{i} a_v$
4: **end for**
5: Generate uniformly distributed random numbers (r_1,r_2)
6: Determine when ($\tau = \ln(1/r_1)/a_0$) and which ($\min\{ \mu \mid \sum_{i=1}^{\mu} a_i \geq r_2 a_0\}$) reaction will occur
7: Set $t = t + \tau$
8: Update $\{X_i\}$
9: **end if**

The SSA can be augmented to incorporate biologically relevant features, such as changes in the volume of the cell during growth, the partitioning of cell volume and content at division, and gene duplication (see e.g. [25, 65, 66]). Changes in cell volume may have significant effects on reaction kinetics. First order reactions have deterministic rate constants (w_M) and stochastic rate constants (c_M) that are equal and independent of volume [67]. However, for higher order reactions, it is necessary to incorporate changing cell volume $V(t)$ into calculation of reaction propensities in order to perform an exact SSA simulation. For example, the stochastic rate constant for a bimolecular second order reaction R_μ at time t is given by

$$c_\mu = \frac{w_\mu}{N_A V_k(t)}, \tag{3.1}$$

where N_A is Avogadro's number [65]. This scaling of higher-order reaction rates by the current (i.e. at the moment of the reaction) cell volume before calculating reaction propensities (known as the adiabatic time-dependent Gillespie approach)

was previously shown to provide a good approximation when the cell volume changes slowly as compared with the time scale of the fastest chemical reaction [65].

Once the SSA incorporates a continuously increasing cell volume, it is necessary also to specify rules that govern cell division. One option is 'sloppy cell-size control' [68] where the cell division is treated as a discrete random event that take place with a volume-dependent probability. Another simpler option is to assume that division occurs once the cell has exceeded a critical size V_{div} corresponding to one doubling of its initial volume, $V_{div} = 2V_0$. The volume doubling time τ_0 then becomes cell division time and t becomes the time since the last division t_{div}. When cell division is triggered, that is when $V_k(t_{div}) \geq V_{div}$, additional rules must be specified to model the partitioning of cellular content between mother and daughter cells. For example, asymmetric cell division can be modeled by setting $V_{daughter} < V_{mother}$. The molecules of the cell can then be partitioned probabilistically between the two volumes [13, 50, 67, 69].

The SSA can accommodate additional discrete events. For example, the G2/M cell cycle checkpoint which ensures proper duplication of the cell's chromosomes before division, can be modeled by defining a variable representing the completion of DNA replication such that cell division is delayed until the DNA content of the cell has doubled. The replication of individual genes, which doubles the maximum rate of gene transcription by doubling the number of corresponding DNA templates, can be modeled as a discrete event that occurs at a fixed time t_{rep} after cell division, that is when $t_{div} \geq t_{rep}$, or as a random event that occurs with some variable probability. In both cases, the DNA-replication event can be placed in a cell-specific stack of future events that is compared against t_{div} (or t in the above pseudocode) following each SSA step. Events in the stack scheduled to occur before this time are then executed and removed from the stack. This can be incorporated into the above pseudocode by

inserting the following two lines:

8a: **if** $length(t_{event}) \geq 0$ **then** (there are scheduled events)

8b: **if** $t > t_{event}(i)$ **then** execute event(i) and delete $t_{event}(i)$ from stack

This approach also provides a convenient basis for simulating the effects of time-delays [14, 25].

The exact SSA can be extremely computationally intensive since the step size τ becomes very small when the total number of molecules is high or the fastest reaction occurs on a time-scale that is much shorter than the time-scale of interest. It is therefore useful to develop techniques that can be used to speed up the simulation. This can be done, for example, using approximate simulation methods (discussed in Section 2.2.2) such as the tau-leaping procedure in which each time step τ advances the system through possibly many reaction events [3]. Additionally, since many independent runs are required to compute population statistics, parallel computing can be used to further optimize simulation run-times. We have chosen to combine the later method with the direct method SSA in order to increase computational efficiency while preserving the statistical accuracy of the simulation results (see Section 3.3).

3.2 Implementation: Constant-Number Monte Carlo Method

Implementations of the augmented SSA presented in the previous section that track only one of the two cells formed during cell division may introduce artifacts in the calculation of population characteristics. For example, growth and reproductive rates may be influenced by the accumulation of genetic mutations within a specific cell lin-

eage or by the current levels of gene expression within individual cells. By discarding the daughter cells in these cases, it is not only no longer possible to track cell lineages, but the full dynamics of these populations may not be captured. Thus to simulate stochastic models of gene expression incorporating such features, it is necessary to couple the SSA with simulation techniques used in studies of population dynamics.

In the modeling of dispersed systems (e.g. colloidal suspensions, cell populations, etc.) the population balance equation (PBE) accounts for all the processes that generate and remove particles, or more generally individuals, from a system of interest [70]. In a general molecular-dynamics framework, the PBE contains terms due to coagulation, fragmentation, and so forth, and is mathematically represented by an integro-differential equation that typically must be solved numerically to obtain particle size distributions as a function of time [46]. Specifically, one can use MC methods to sample a finite subset of a system in order to infer its properties and study finite-size effects and local fluctuations not captured by a mean field approximation [3, 45, 46, 70]. Futhermore, a MC method is appropriate as its discrete nature lends itself naturally to growth processes involving discrete events, and can simulate growth over arbitrarily long times with finite numbers of simulation particles while maintaining constant statistical accuracy [45].

One approach for constructing a reliable and efficient algorithm to simulate cell populations, is to adopt the constant-number MC method to simulate the birth-death processes that take place within such populations [44–47]. Here, by choosing at random a sufficient number N of individuals, one obtains a representative sample in such a way that population-wide dynamics (e.g. epigenetic effects) are captured. Thus the constant-number MC approach permits the modeling of growing populations using a fixed number of cells while avoiding the alternative (i.e. an infinitely growing population) by sampling N particles representing the population as a whole. Specifically,

individuals are stored in an array during simulation; when the actual processes result in net gain (e.g. particle fragmentation or cell division), positions in the array are randomly selected and are overwritten with the 'surplus' individuals. When the process results in net loss (e.g. particle coagulation or cell death), individuals are selected at random and are placed in the vacated positions of the array. This essentially amounts to expanding and contracting the physical volume represented by the simulation as required to continuously maintain a constant number of individuals [45].

The constant-number MC approach has been successfully applied to a variety of non-biological particulate processes [45, 46, 71] as well as cell population dynamics [44, 47]. However, it should be noted that when one employs the constant-number MC method, a fixed number of individuals, which are randomly chosen, are being used to represent the population as a whole and therefore there is error associated with the method. This error was shown to be $(\ln x)^{0.89}/\sqrt{2N}$, where x is the extent of growth [46].

In our implementation of the constant-number MC, we keep track of individual mother and daughter cells in two separate arrays. Each time a cell divides, the daughter cell is placed in the daughter array and the time of birth recorded. Then, at specified intervals, cells within the mother array are replaced one at a time, with the oldest daughter cells being inserted first. Because every mother cell is equally likely to be replaced during the sample update, the size distribution of the population remains intact for sufficiently large populations [46]. In our case, the size distribution corresponds to the distribution of cell ages (or volumes) across the population.

Our implementation of the constant-number MC method can be described by the following pseudocode:

1: **if** $t > t_{restore}$ and $NC_{daughter} \geq 1$ **then**

2: **for all** $NC_{daughter}$ **do**

3: Randomly select mother cell

4: Replace mother cell with oldest available daughter cell

5: **end for**

6: **end if**

Here, $t_{restore}$ is the interval between population updates and $NC_{daughter}$ the number of daughter cells born since the last update. To avoid simulating the daughters of daughter cells, $t_{restore}$ is chosen such that mother cells divide at most once, and daughter cells not at all, during a particular $t_{restore}$ interval.

3.3 Cell Population Dynamics Algorithm

Simulations using the cell population dynamics algorithm are carried out from an initial population distribution. Gene expression in each cell is described by a user defined set of equations, and population statistics are obtained at a specified sampling interval. Here, the direct method SSA [2,43] is used for stochastic simulation, however any stochastic simulation method can be implemented. The volume of each cell k is modeled using an exponential growth law

$$V_k(t_{div}) = V_0 \exp\left[\ln(2)\left(\frac{t_{div}}{\tau_0}\right)\right], \tag{3.2}$$

where V_0 is the cell volume at the time of its birth, t is the time and τ_0 is the interval between volume doublings.

Parallelism is implemented across the simulation, as a large number of independent simulations need to be performed when simulating the dynamics of a cell population, in a shared memory multiprocessor environment.

The present algorithm can be expressed by the flow diagram (Fig. 3.1) and the

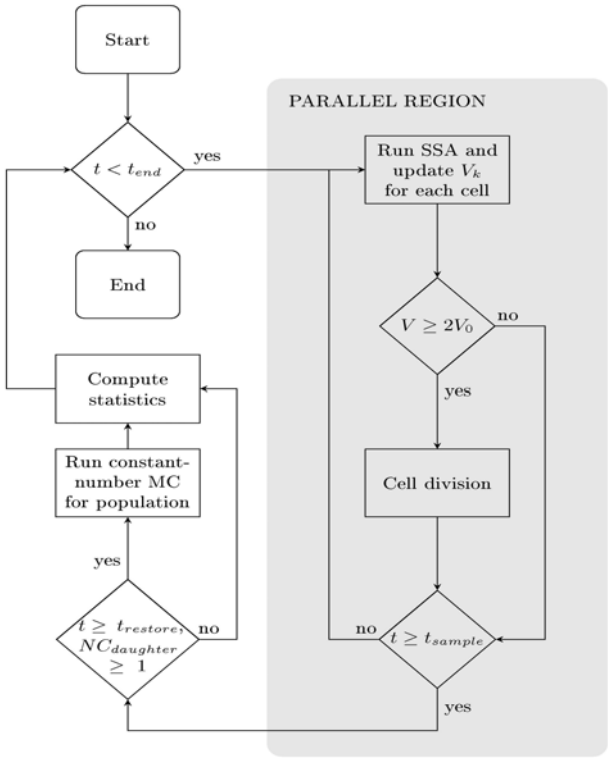

Figure 3.1 Flow diagram of the present algorithm for the parallel stochastic simulation of gene expression and heterogeneous population dynamics.

following pseudocode (see Appendix B for the full Fortran 90 code):

1: **while** $t < t_{end}$ **do**
2: *begin parallel region*
3: **for all** $NC_{population}$ such that $t < t_{sample}$ **do**
4: Gillespie SSA (see pseudocode in Section 3.1)
5: Update V_k via Eq. 3.2
6: Execute events in stack with $t_{event} < t_{div}$
7: **if** $V_k(t_{div}) \geq V_{div}$ **then**
8: Execute cell division
9: Increment $NC_{daughter}$
10: **end if**
11: **end for**
12: Update t_{sample}
13: *end parallel region*
14: Execute constant-number MC (see pseudocode in Section 3.2)
15: Compute statistics
16: **end while**

Here, $NC_{population}$ is the total number of cells in the population, V_k the volume of cell k, and t_{sample} the user defined population sampling interval.

The algorithm can execute simulations of considerable size in reasonable times. For example, an IBM with 2 quad-core processors (1.86GHz cores) and 2.0GB of RAM completed a $10^5 s$ simulation of the network presented in Section 4.1.2 for 8000 cells in $81s$ when $v_0 = 0.3s^{-1}$, $v_1 = 0.05s^{-1}$, $d_0 = 0.05s^{-1}$, $d_1 = 5 \times 10^{-5}s^{-1}$, $t_{div} = 3600s$, and $t_{restore} = 3300s$.

Chapter 4

Results

In this section, we present the numerical results obtained from the stochastic simulation of gene expression, where models describing increasingly realistic biological features are considered. Specifically, in order to benchmark the algorithm, we compare these simulation results with steady-state and time-dependent analytical solutions for several scenarios, including steady-state and time-dependent gene expression and the effects of cell growth, division, and DNA replication.

Details corresponding to methods used to obtain analytical solutions can be found in Sections 2.1.1 and 2.1.2 and the references provided within this chapter. Part of the motivation for developing the algorithm is the anticipation that analytical solutions corresponding to models incorporating complex biochemical reaction network and cellular physiology will be intractable.

We begin in Subsection 4.1.1 with the simplest case - the steady-state expression of gene (production of monomer from a gene) in the absence of cell growth and division. In Subsection 4.1.2, we consider time-dependent gene expression, that is, transcription (production of RNA from a gene) and translation (production of protein from an mRNA), and benchmark this scenario against the corresponding time-dependent

analytical distributions. In Subsection 4.1.3 we consider both time-dependent and time-independent gene expression using a model that incorporates the effects of gene duplication and cell division on gene expression dynamics in individual cells using the constant-number Monte Carlo (MC) method. All simulations statistics were obtained from populations consisting of 8000 cells.

4.1 Numerical Results

4.1.1 Steady-State Validation

Kepler and Elston [72] presented a model of constant gene expression in which the state space consists of the number of gene product monomers M (integer variable representing the combination of mRNA and protein), and a binary variable to represent the occupancy of the gene's promoter region by a regulatory protein. This system is described by reactions [72]:

$$M \xrightarrow{\delta} \oslash \tag{4.1}$$

$$\vartheta_0 \xrightarrow{k_0 K} \vartheta_1 \tag{4.2}$$

$$\vartheta_1 \xrightarrow{k_1 K} \vartheta_0 \tag{4.3}$$

$$\oslash \xrightarrow{\alpha_s} M \tag{4.4}$$

Eqs. (4.1) and (4.4) describe, respectively, the degradation and production of the gene product, where δ is the degradation rate, α_s the rate for protein production (s = 0 or

1, depending on the chemical state of the operator), and $\oslash$ the protein sink. Eqs. (4.2) and (4.3) account for the spontaneous transitions between operator states (ϑ_0 and ϑ_1 denote occupied and unoccupied operator states respectively). In these equations, K sets the time scale and k_0 and k_1 are dimensionless constants constrained to obey $k_0 + k_1 = 1$.

This model was shown to have the following equations for steady-state values (denoted by overbars) of the mean $\overline{\mu}$ (Eq. (4.5)), variance $\overline{\sigma^2}$ (Eq. (4.6)), and the coefficient of variation $\overline{CV}$ (Eq. (4.7)), corresponding to M:

$$\overline{\mu}_M = \frac{1}{\delta}(\alpha_0 k_1 + \alpha_1 k_0) \tag{4.5}$$

$$\overline{\sigma^2}_M = \overline{\mu}_M + k_0 k_1 \left[\frac{\alpha_0 - \alpha_1}{\delta}\right]^2 \frac{\delta}{\delta + K} \tag{4.6}$$

$$\overline{CV}_M \equiv \frac{\overline{\sigma^2}_M}{\left(\overline{\mu}_M\right)^2} = \frac{1}{\overline{\mu}_M} + k_0 k_1 \frac{\delta}{\delta + K}\left[\frac{(\alpha_0 - \alpha_1)}{\alpha_0 k_1 + \alpha_1 k_0}\right]^2 \tag{4.7}$$

In order to validate our parallel implementation of the SSA, we simulated Eqs. (4.1)-(4.4) and compared the results to the corresponding steady-state analytical solutions (Eqs. 4.5 and 4.7). In Figure 4.1, the steady-state simulation and analytical results obtained for $\overline{\mu}_M$ and $\overline{CV}_M$ are plotted for a range of α_0 and α_1 values. The simulation results and analytical solutions were found to be in excellent agreement.

4.1.2 Time-Dependent Population Distributions

Population-based simulation algorithms have the advantage of yielding time-dependent population-distributions as the output. To evaluate the accuracy of our approach in this respect, further validation against a time-dependent distribution is of interest.

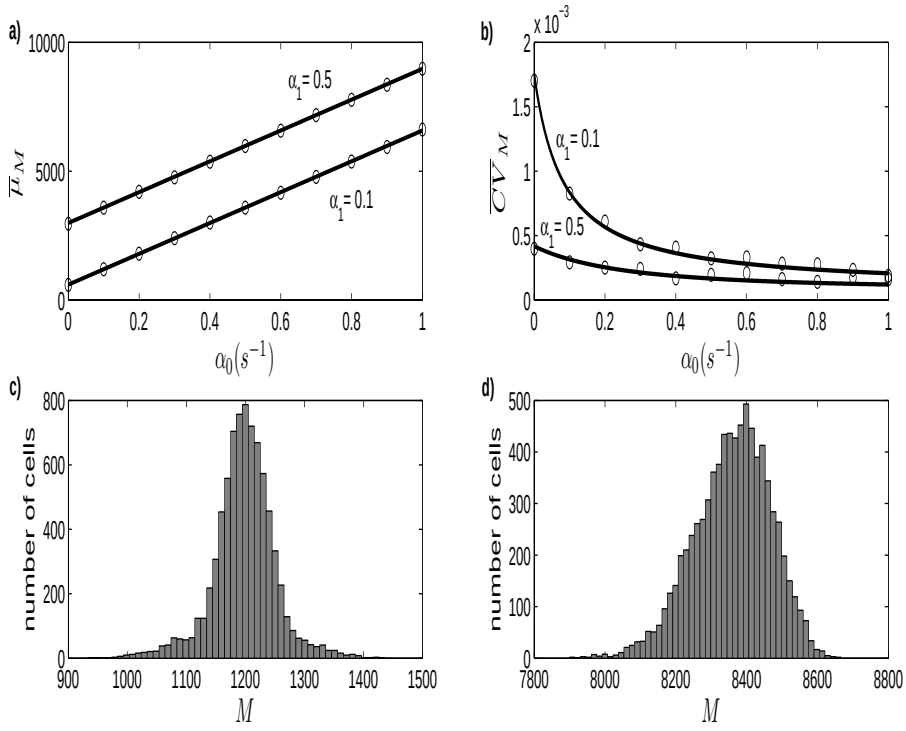

Figure 4.1 Comparison of analytical solutions and simulation results. (a) Gene product monomer mean steady-state ($\overline{\mu}_M$) and (b) coefficient of variation ($\overline{CV}_M$) are plotted for a range of transcription/translation parameter values (α_0 and α_1). Open circles indicate simulation results and black curves analytical solutions [72]. Protein population distributions corresponding to (a) are shown in (c) for $\alpha_0 = \alpha_1 = 0.1$ and (d) for $\alpha_0 = 0.9$ and $\alpha_1 = 0.5$.

For this purpose, we simulated a two-stage gene expression model consisting of the following biochemical reactions:

$$T \xrightarrow{v_0} mRNA \tag{4.8}$$

$$mRNA \xrightarrow{d_0} \oslash \tag{4.9}$$

$$mRNA \xrightarrow{v_1} P \tag{4.10}$$

$$P \xrightarrow{d_1} \oslash \tag{4.11}$$

Eq. (4.8) describes transcription at a rate v_0, Eq. (4.9) the degradation of the mRNA at a rate d_0, Eq. (4.10) translation at a rate v_1, and Eq. (4.11) the protein degradation at a rate d_1. Here it is assumed that the promoter T is always active and thus the model has two stochastic variables, the number of mRNA and the number of proteins P.

Shahrezaei and Swain [50] studied the system described by Eqs. (4.8)-(4.11) and derived an approximate protein distribution as a function of time (Eq. 2.34). The approximation is based on the assumption that the degradation of mRNA is fast compared to the degradation of proteins. Consequently, the dynamics of mRNA are at the steady-state for the most of a protein's lifetime. Here, when the initial number of proteins n is set to zero, equations describing protein mean and noise (Eqs. (2.35)-(2.36), respectively) can be obtained (see Section 2.1.1).

To benchmark the ability of the algorithm to accurately generate time-dependent population distributions, we simulated Eqs. (4.8)-(4.11) under conditions where the assumptions of Eq. (2.34) are satisfied, and compared the resulting distributions with corresponding time-dependent analytical distributions. Figure 4.2 shows the simulated and analytical distributions at two different values of dimensionless time τ. The population statistics, namely μ_P and η_P, as a function of τ are shown in Figure 4.3.

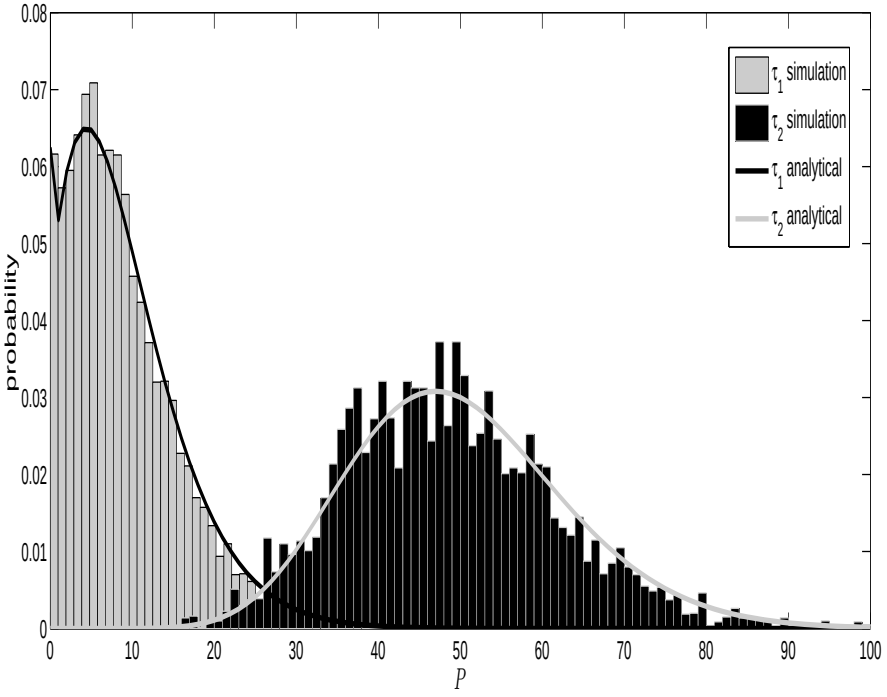

Figure 4.2 Simulation results and time-dependent analytical solutions of a two-stage model of gene expression [50]. The distributions of protein numbers for a population of cells at two different dimensionless times, $\tau = 0.2$ and $\tau = 10$, are shown.

In both cases, the simulated protein distributions and statistics are in good agreement with analytical results (Eqs. (2.35) and (2.36)).

4.1.3 Gene Duplication, Cell Division, and Time-Dependent Validation

To explore the accuracy of the algorithm when simulating models incorporating cell growth, division, and DNA replication, we implemented the simplified reaction network presented in Swain *et al.* [13]. The reduced reaction network was obtained from a model of gene expression consisting of 8 molecular species and 11 chemical

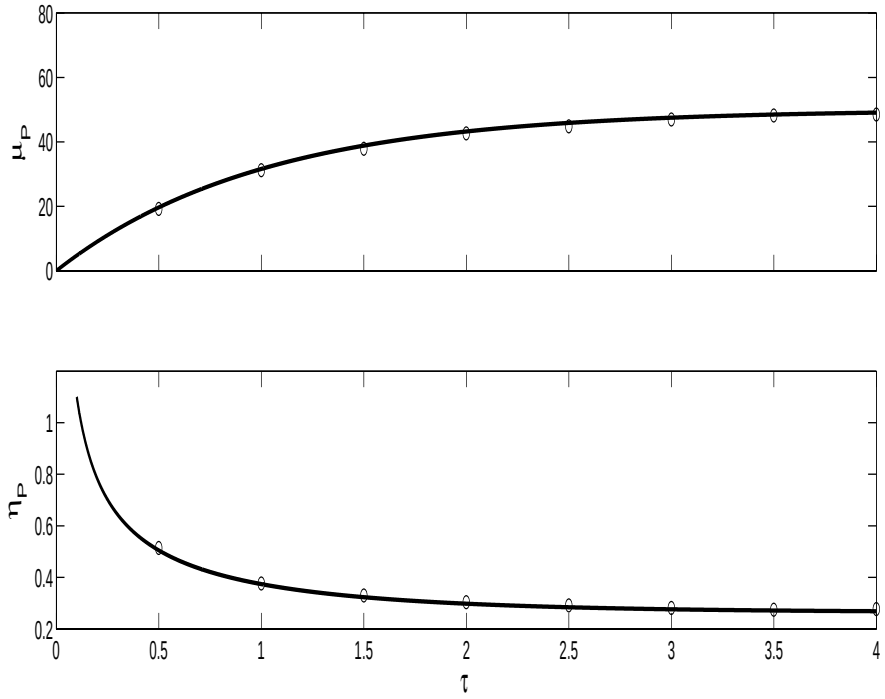

Figure 4.3 Simulation results and time-dependent analytical solutions of a two-stage model of gene expression [50]. Mean protein μ_P (top) and noise η_P (bottom) are plotted as a function of dimensionless time τ. Open circles indicate simulation results and black curves analytical solutions [50].

reactions. For this simplified network, it is possible to derive time-dependent analytical results for the mean protein number and coefficient of variation in protein number. Importantly, by making the appropriate approximations, the effects of gene replication and cell division can be included in the analytical solutions. The reduced model has two components - one described by the reactions in Eqs. (4.8)-(4.11) (note that the reaction rates v_1 and d_0 can be directly related to $v_1^{'}$ and $d_0^{'}$ in the original model [13]), and another describing pre-transcription kinetics. This component captures the reversible binding of RNA polymerase to the promoter (rate constants b_0 and f_0), and the formation of an open promoter complex (rate constant k_0). These steps are described by the reactions

$$D \quad \underset{b_0}{\overset{f_0}{\rightleftharpoons}} \quad C \tag{4.12}$$

$$C \quad \xrightarrow{k_0} \quad D + T \tag{4.13}$$

where D, C and T represent the promoter with polymerase unbound, the promoter with polymerase bound and the open promoter complex, respectively. Since the total number n of DNA molecules is conserved before and after replication, D and C can be constrained by

$$n_0 + n_1 = n, \tag{4.14}$$

where n_0 and n_1 are the number of promoter copies in state D and C respectively.

To derive an analytical solution, the authors invoked the assumption that the distributions of C, T, and $mRNA$ can be approximated by their steady state distributions. While this assumption thus ignores the transient dynamics of these species, it is expected to introduce a minimal error since the protein degradation rate d_1 is much smaller compared to the other reaction rates. As a consequence, the mean and coefficient of variation protein P are time-dependent while the moments of the distributions of the other species are constant. Even with this approximation, the derivation of

the analytical solutions for the mean and coefficient of variation is rather arduous. The derivation consists of three separate stages: the derivation of time-dependent expression for the population mean and noise (analogous to the derivation of time-dependent moments in Section 2.1.1), the incorporation of gene replication, and the addition of cell division. The complete derivation can be found in the supplementary material of Swain *et al.* [13] and the corresponding steady-state and time-dependent analytical solutions are as follows

$$\overline{\mu}_{mRNA} = \frac{f_0 k_0 n}{d_0^{'} l} \tag{4.15}$$

$$\overline{CV}_{mRNA} = \frac{(\overline{\mu}^2_{mRNA}) - (\overline{\mu}_{mRNA})^2}{(\overline{\mu}_{mRNA})^2} = \frac{1}{\overline{\mu}_{mRNA}} - \frac{d_0^{'} v_0 (d_0^{'} + l + v_0)}{n(d_0^{'} + l)(l + v_0)(d_0^{'} + v_0)} \tag{4.16}$$

$$\mu_P(t) = \frac{v_1^{'}}{d_1^{'}} \overline{\mu}_{mRNA} \phi_0(t) \tag{4.17}$$

where ϕ_0 is a continuous function of t,

$$\phi_0(t) = 1 - \frac{e^{-d_1(T - t_d + t)}}{2 - e^{-d_1 T}},\ for\ 0 \leq t \leq t_d \tag{4.18}$$

$$\phi_0(t) = 2\left[1 - \frac{e^{-d_1(t - t_d)}}{2 - e^{-d_1 T}}\right],\ for\ t_d \leq t \leq T_{cdiv} \tag{4.19}$$

$$CV_P(t) = \frac{1}{\mu_P(t)} + \frac{1}{\overline{\mu}_{mRNA}}\left[1 - \frac{f_0 k_0}{l^2}\right]\frac{d_1}{d_0^{'}}\phi_1(t) \tag{4.20}$$

with,

$$\phi_1(t) = \frac{2 - e^{-d_1 T}}{2 + e^{-d_1 T}} \times \frac{4 - e^{-2d_1 T} - 2e^{-2d_1 t} - e^{-2d_1(T + t - t_d)}}{\left(2 - e^{-d_1 T} - e^{-d_1(T + t - t_d)}\right)^2},\ for\ 0 \leq t \leq t_d \tag{4.21}$$

and,

$$\phi_1(t) = \frac{2 - e^{-d_1 T}}{2 + e^{-d_1 T}} \times \frac{4 - e^{-2d_1 T} - e^{-2d_1 t} - 2e^{-2d_1(t - t_d)}}{2\left(2 - e^{-d_1 T} - e^{-d_1(t - t_d)}\right)^2},\ for\ t_d \leq t \leq T_{cdiv} \tag{4.22}$$

Eq. (4.15) describes the mean mRNA number before gene duplication ($t < t_d$), noting

that the value is twice this result after gene replication ($t > t_d$), and Eq. (4.16) gives the mRNA coefficient of variation. Here, $l = f_0 + b_0 + k_0$.

It is noted that Eqs. (4.15) and (4.16) are time independent. This is due to an assumption by the authors that the RNA is in a quasi-steady state proportional to the gene copy number n, and that all other time dependencies are absorbed into the protein distribution. The mean protein number as a function of time is then described by Eqs. (4.17) and (4.18) before gene duplication, and by Eqs. (4.17) and (4.19) after duplication. Similarly, Eqs. (4.20) and (4.21) describe the time-dependent coefficient of variation in protein number before duplication, and Eqs. (4.20) and (4.22) describe the coefficient of variation after duplication.

Our simulation results are compared to the corresponding steady-state and time-dependent analytical solutions Figures 4.4-4.6. In these simulations, we use the same assumptions as in [13]; the cell volume increases linearly up to time of cell division T_{cdiv}, gene replication occurs at $t_{rep} = 0.4T_{cdiv}$ and cell division is symmetric with binomial partitioning of molecules. Simulated protein number and concentration, as well as mRNA number dynamics, for single cells (Fig. 4.4) are comparable with the simulation results obtained by Swain *et al.* [13]. Figures 4.5 and 4.6 further compare population characteristics estimated from simulations to those predicted by the corresponding steady-state analytical solutions. Both RNA ($\overline{\mu}_{mRNA}(n)$ and $\overline{CV}_{mRNA}(n)$, Fig. 4.5) and protein (μ_P Fig. 4.6) characteristics are in good agreement with analytical results in Eqs. (4.15)-(4.22). The average percent error between the simulation and analytical results in η_P shown in Figure 4.6 was calculated to be 1.84% and 0.85% for $d_1 = 6.42$ x 10^{-5} and $d_1 = 6.42$ x 10^{-6}, respectively. These errors can be explained by the fact that the corresponding analytical solutions (Eqs. (4.20)-(4.22)) are approximate; simlar results were obtained in Swain *et al.* [13].

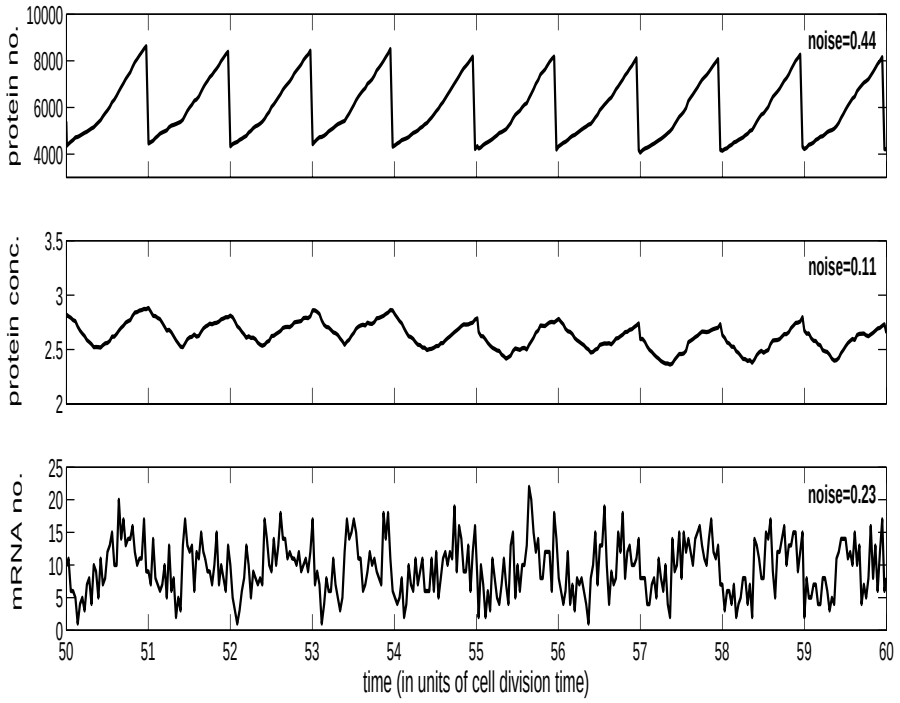

Figure 4.4 Time series of a single cell within a growing and dividing population. Protein number (top) and concentration (middle), and mRNA number (bottom), were obtained and found to be in agreement with a model of translation provided in [13]. Gene duplication occurs every $t_d = 0.4T_{cdiv}$ into the cell cycle and results in an increased rate of protein production until the next cell division event where the number of genes prior to duplication is restored.

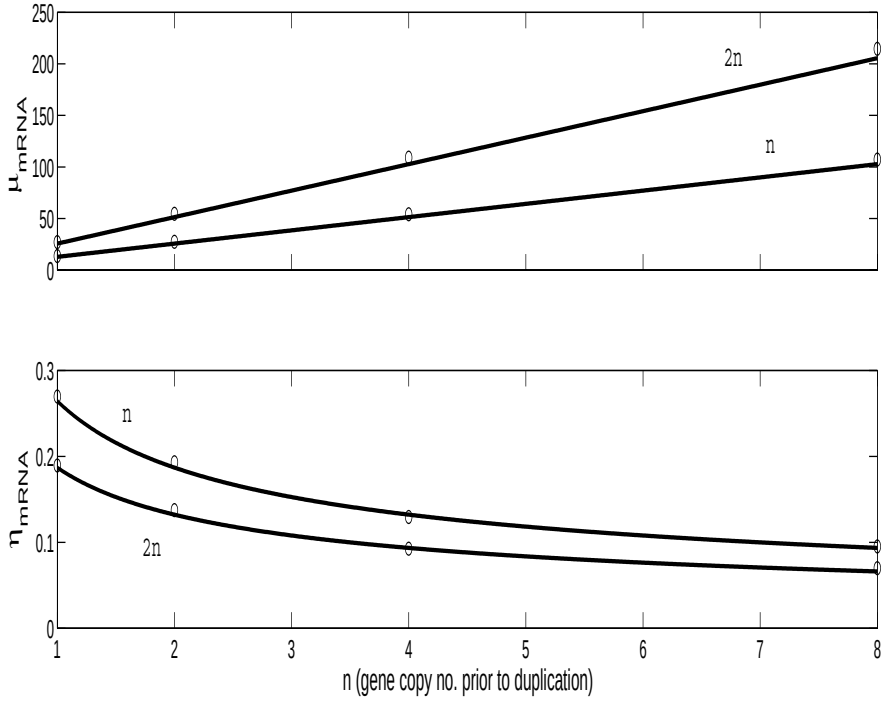

Figure 4.5 Comparison of simulation results and analytic solutions. Mean mRNA values are plotted as a function of gene copy number n (top). The noise in mRNA number is also plotted as a function of n (bottom). Note that mean mRNA values increase and the noise decreases after gene duplication as expected. Black curves indicate analytical values [13] and open circles simulation results.

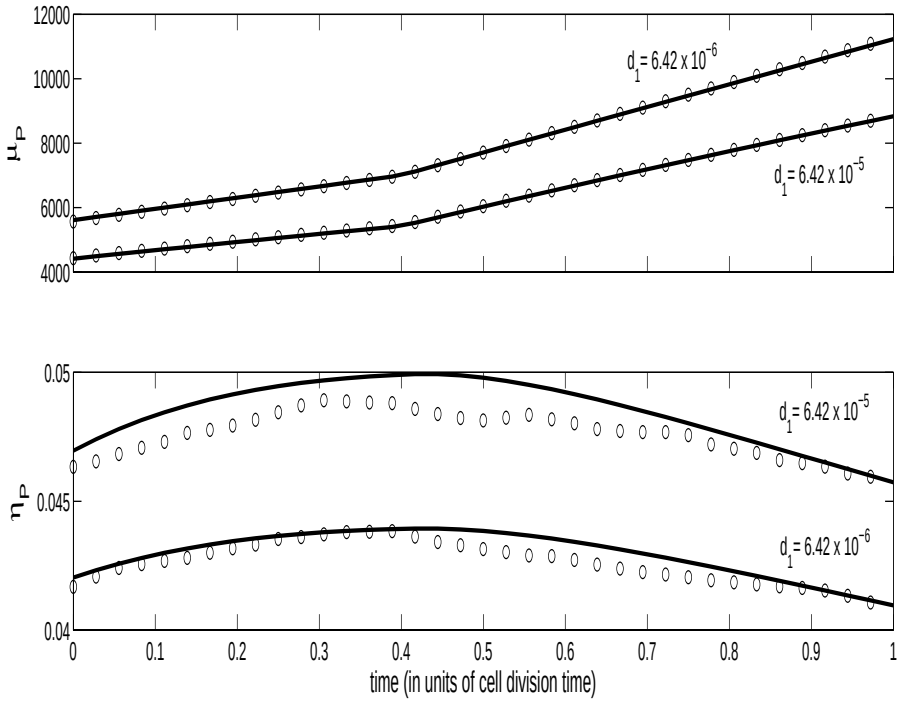

Figure 4.6 Comparison of simulation results and analytic solutions. Mean protein number (top) and noise (bottom) as a function of time t for two different values of the protein degradation parameter d_1. Note the increase in protein production rate and decrease in noise levels that occur after gene duplication at $t = 0.4$. Open circles indicate simulation results and black curves analytical values [13].

4.2 Simulating Complex Population Dynamics

4.2.1 Asymmetric Cell Division

To investigate sources of external variability in eukaryotic gene expression, Volfson *et al.* [32] combined computational modelling with fluorescence data. As part of this study, the authors simulated the distribution of cell sizes within a population of *Saccharomyces cerevisiae* (budding yeast). In these simulations, cells grew exponentially until they reached a critical volume V_c where they divide. The volume at division was drawn from a normal distribution with a mean specified as a function of genealogical age and coefficient of variation 0.15. Following division, the mother cell retained 70 % of the volume ($V_0 = 0.7V_c$) while daughter cells were correspondingly smaller ($V_0 = 0.3V_c$). The resulting distribution of cell sizes obtained from an initial population of 1000 cells allowed to grow to 100000 cells was found to be in agreement with experimental and analytical results [32].

The model by Volfson *et al.* [32] is ideally suited for benchmarking the constant-number MC method. As in Volfson *et al.* [32], we first simulated the growth of a population initially consisting of 1000 cells and obtained the steady-state size distribution once the population grew to 100000 cells (Fig. 4.7a). Next, we repeated the simulations using the constant-number MC method to estimate the size distribution from a representative sample (8000 cells) of this cell population (Fig. 4.7b). A plot of the probabilities for the sample population against the probabilities of the 'true' population shows that the difference between these variables is minimal (Fig. 4.7c). These results complement previous studies [44–47, 71] demonstrating the ability of the constant-number MC method to capture complex population dynamics.

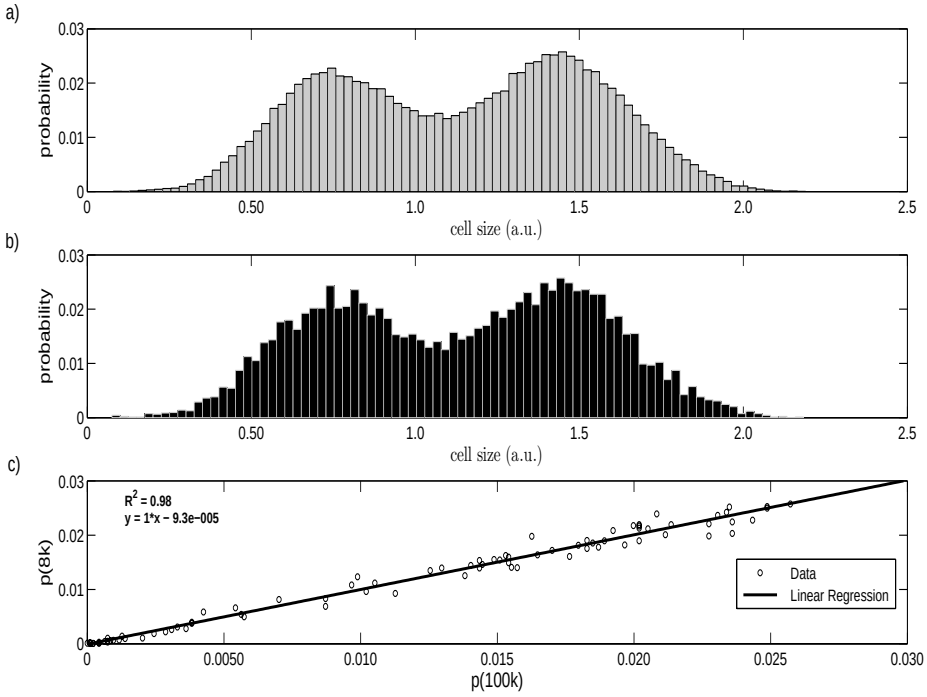

Figure 4.7 Simulation of a stochastic population dynamics model [32] of a *Saccharomyces cerevisiae* population undergoing stochastic (size at division) and asymmetric (partitioning of cell volume) division. (a) Steady-state distribution of cell sizes for a population of 100000 cells. (b) Steady-state size distribution of a representative sample (8000 cells) obtained using the constant-number Monte Carlo method [45,46] of the 'true' population shown in (a). (c) Plot of the probabilities population shown in (b) against the probabilities of the population shown in (a) along with linear regression.

4.2.2 Bet-Hedging Cell Populations

One of the most interesting potential applications of the simulation algorithm described in Section 3.3 is the investigation of interactions between environmental changes, population dynamics, and gene expression in individual cells. For example, this algorithm can be used to study the optimization of fitness in fluctuating environments, which is a classic problem in evolutionary and population biology [73–76]. Acar *et al.* [77] experimentally investigated how stochastic switching between phenotypes in changing environments affected growth rates in fast and slow-switching populations by using the galactose utilization network in *Saccharomyces cerevisiae* (budding yeast). Specifically, a strain was engineered to randomly transition between two phenotypes, ON and OFF, characterized respectively by high or low expression of a gene encoding the Ura3 enzyme necessary for uracil biosynthesis. Each phenotype was designed to have a growth advantage over the other in one of the two environments. In the first environment (E_1) which lacks uracil, cells in the ON phenotype have an advantage. In the second environment (E_2), cells in the OFF phenotype have an advantage due to the presence of a drug (5-FOA) which is converted into a toxin by the Ura3 enzyme. In this environment, which also contains uracil, cells expressing Ura3 will have low viability while cells not expressing Ura3 will grow normally.

We first follow the approach that was used in Acar *et al.* [77] to describe the dynamics of phenotype switching, where cells are in either the ON or the OFF state:

$$ON \underset{k_2}{\overset{k_1}{\rightleftharpoons}} OFF \tag{4.23}$$

In this scenario, cells randomly switch between the high and low expressing states at rates k_1 and k_2 (see [77] for parameter values corresponding to slow and fast-switching cells). The growth rate (Eq. (3.2)) of fit cells was set higher than the corresponding

growth rate for unfit cells in the same environment. In order to avoid synchronization in the population level dynamics, we set $V_{div} = 2V_0 + \xi$, where ξ is a small random number drawn from a normal distribution with zero mean and 0.2 variance.

Figure 4.8 shows the growth rates obtained from simulations of slow and fast-switching cell populations, where cells were transfered from E2 to E1, and vice versa, at $t = 0$. Growth rates show a transition period and a steady-state region. In agreement with experimental results [77], the transient corresponding to the E2 to E1 change (Fig. 4.8a) is shorter than the transient corresponding to the E1 to E2 change (Fig. 4.8b), and slow-switching cells have a higher steady-state growth after recovery from either of the environmental changes.

Next, we implemented a full model of gene expression described by the following biochemical reaction scheme [6]:

$$T_A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} T_R, \tag{4.24}$$

$$T_A \xrightarrow{v_{0,A}} T_A + mRNA \tag{4.25}$$

$$T_R \xrightarrow{v_{0,R}} T_R + mRNA \tag{4.26}$$

$$mRNA \xrightarrow{d_0} \oslash \tag{4.27}$$

$$mRNA \xrightarrow{v_1} mRNA + P \tag{4.28}$$

$$P \xrightarrow{d_1} \oslash \tag{4.29}$$

Eq. (4.24) describes the transitions to the active (upregulated level of gene expression) T_A and repressed (basal level of gene expression) T_R promoter states at rates k_1 and k_2 respectively, Eqs. (4.25) and (4.26) the mRNA production from the T_A (at a rate $v_{0,A}$) and T_R (at a rate $v_{0,R}$) states respectively, Eq. (4.28) the protein production

from mRNA at a rate v_1, and Eqs. (4.27) and (4.29) respectively the mRNA (at a rate d_0) and protein (at a rate d_1) degradation. The fitness w_k of each cell k, which is here defined as a function of the environment and cellular protein concentration $[P]$, was described by a Hill function

$$w_k(E,P) = \begin{cases} \frac{[P]^n}{[P]^n+K^n}, & if\ E = E1 \\ \frac{K^n}{K^n+[P]^n}, & if\ E = E2. \end{cases} \tag{4.30}$$

This equation describes partitioning of cells into fit ($w_k(E,P) \geq 0.5$) and unfit ($w_k(E,P) < 0.5$) phenotypes corresponding to whether or not their $[P]$ in a particular environment is above or below a particular value given by the Hill coefficient K. The volume of each cell was described by Eq. (3.2), except here $\tau_0 = \tau_\phi/w$, where τ_ϕ is the cell division time in absence of selection. To incorporate the effect of fitness on gene expression, the value of transcription rate parameter v_0 depended on whether or not a cell was fit in either $E1$ or $E2$ (see Fig. 4.9 and [77] for parameters).

The population distributions obtained for this model are shown in Figure 4.9. Specifically, we first obtained the steady-state protein concentration distributions for cells in $E1$ and $E2$ (Fig. 4.9a and 4.9b, respectively). Here, the majority of cells either fell within a distribution centered at higher value characterizing the ON cells, or a distribution centered at a lower value of P characterizing the OFF cells, in $E1$ or $E2$ respectively. The rest of the cells fell within the distribution capturing the unfit subpopulation in both environments. These results were found experimentally in [77] and are expected, as higher levels of the uracil enzyme are either favorable or unfavorable with respect to the fitness of the cells depending on the environment. Additionally, the time-dependent population distributions after the transition to $E1$ from $E2$, and vice versa, were obtained (Fig. 4.9a and 4.9b, respectively). Here, the dynamics of the two distinct subpopulations of cells in transition between the steady-

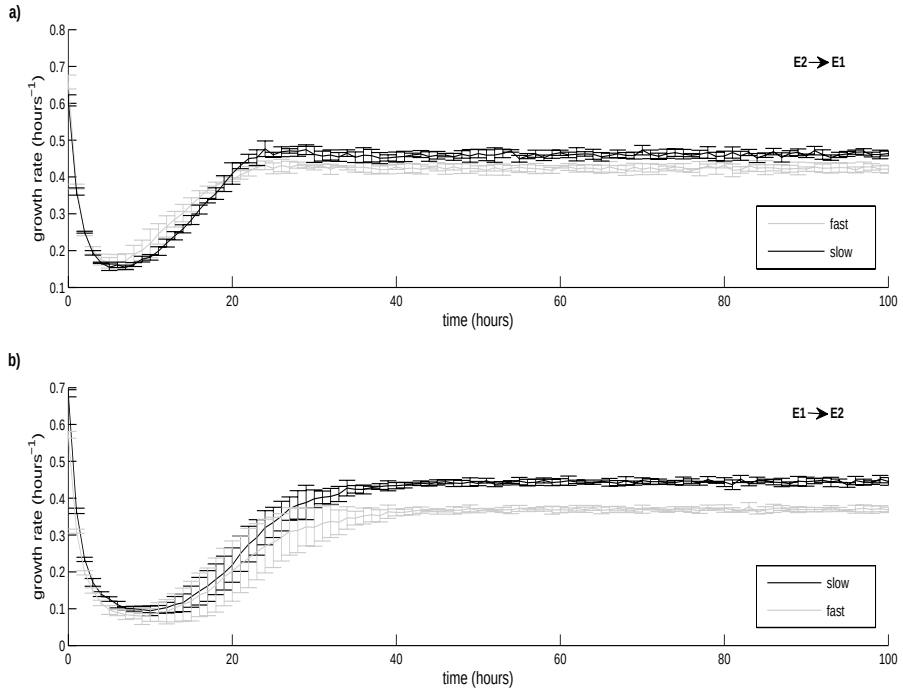

Figure 4.8 Simulations of populations of slow and fast-switching cells (20 realisations). (a) Growth rates of cells transfered from an environment containing uracil and 5-FOA (E2) to one containing no uracil (E1) at $t = 0$. (b) Growth rates of cells transfered from E1 to E2 at $t = 0$. Note that the transient before the steady-state region is shorter in (a) than in (b), and that the slow-switching cells have a higher steady-state growth after recovery from the environmental change, in agreement with experimental results found in [77].

states are visible. As time progresses after the environmental transition, fewer and fewer of the cells are in the unfit state (ON in Fig. 4.9a and OFF in Fig. 4.9b), as the cells in the more fit state (OFF in Fig. 4.9a and ON in Fig. 4.9b) grow and divide at a faster rate and therefore come to dominate the population in terms of absolute numbers.

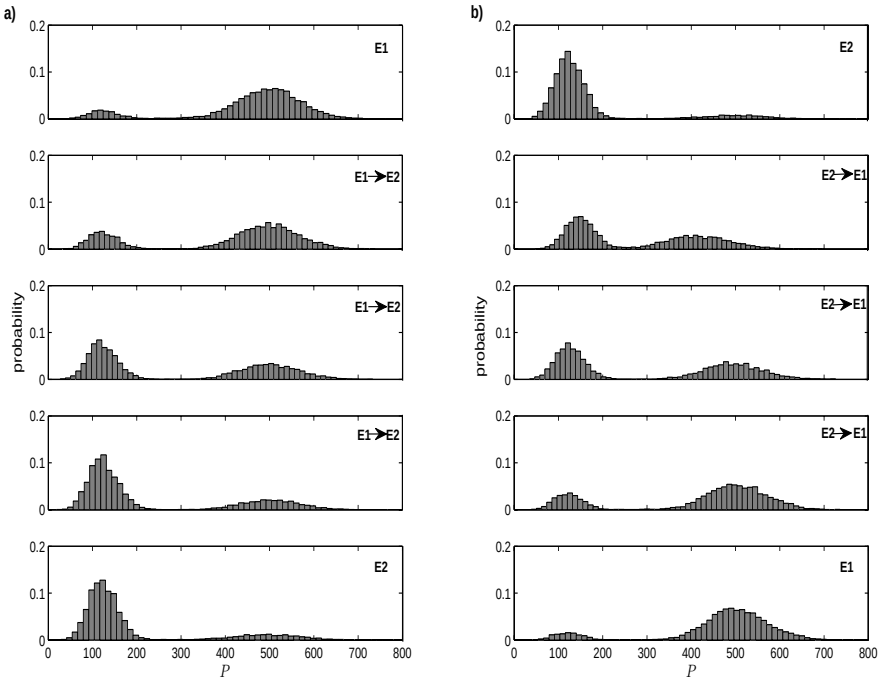

Figure 4.9 Simulations of environmental effects on phenotypic distribution. (a) Steady-state (top and bottom figures) and time-dependent (middle figures) protein distributions of cells resulting from an environment change from E1 to E2. (b) Steady-state (top and bottom figures) and time-dependent (middle figures) protein distributions of cells resulting from an environment change from E2 to E1. Note that when a sufficient amount of time has elapsed after the environmental transition from either E1 to E2 or vice versa, cells with either the OFF (represented in each pannel by the distribution with the lower mean protein, P, value) or ON (represented in each pannel by the distribution with the higher mean P value) phenotype proliferate, respectively, in agreement with experimental results found in [77]. The following parameters were used: $d_0 = 0.005s^{-1}$, $v_1 = 0.1s^{-1}$, $d_1 = 0.008s^{-1}$, $K = 200$, $n = 10$. For fit cells in E1 $v_{0,A} = 0.2$ and for unfit cells $v_{0,R} = 0.05$ - vice versa in E2. Additionally τ_ϕ was set to the mean doubling time (MDT) of 1.5 hours for *Saccharomyces cerevisiae* [78].

Chapter 5

Conclusion

Our understanding of the origins and consequences of stochasticity in gene expression has advanced significantly in recent years. This advance has been fueled by theoretical developments enabling biological hypothesis formulation using stochastic process and dynamical systems theory, as well as experimental breakthroughs in measurements of gene expression at the single cell level [10].

Noise in gene expression was originally viewed as being detrimental in terms of cellular function due to the corruption of intracellular signals negatively affecting cellular regulation with possible implications for disease. However, noisy gene expression can also be advantageous, providing the flexibility needed by cells to adapt to stress such as a changing environment [27, 34, 77]. Stochasticity in gene expression provides a mechanism for the occurrence of heterogeneous populations of genetically identical cells, in terms of phenotypic and cell-type diversity, which can be established during cellular growth and division [5, 6, 26]. Furthermore, it has been suggested that intrinsic stochasticity in gene expression is an evolvable trait [35, 41].

It is important to note that heterogeneity exists in all biological systems, not just those commonly discussed in the literature involving single-celled organisms such as

bacteria and yeast. This is likely to become increasingly important in the context of human genetics and diseases in the near future. For example, it was recently proposed by Brock *et al.* [26] that non-genetic variability resulting from gene expression noise can contribute to the somatic evolution of cancer cells, thus accelerating tumour progression independently of genetic mutation.

The construction of models describing biological systems is a fundamental part of systems biology. Traditionally, diagrammatic models were used to summarize a mechanistic understanding of a system obtained from experimental observations. These models, despite their benefits, yield a rather static picture of cellular processes. The need to translate these models into more dynamic forms that can capture time-dependent processes, together with increases in the model's scale and complexity, has prompted scientists to harness computational resources (e.g. parallel programming and cluster computing) to build and analyze ever-larger models. The long-term vision is that these large-scale models will revolutionise the way biological and biomedical research is conducted, ultimately enabling the design of new therapies [79]. Building the capacity to simulate cell populations at the single cell resolution is an important step towards harnessing this potential.

In this thesis we explored the theoretical background required to analytically derive solutions and perform simulations of models of stochastic gene expression. This theory was then used to develop an algorithm for the stochastic simulation of heterogeneous population dynamics. Specifically, we combined the stochastic formulation of chemical kinetics that is commonly used to simulate complex interaction networks in biological cells, and a MC method to sample a finite subset of cells that preserves population-level statistics of fluctuating intrinsic variables. The accuracy of the proposed method was verified by comparing simulation results of gene expression models with corresponding steady-state and time-dependent analytical solutions and

experimental results. Parallel execution of the algorithm was found to significantly decrease run-times in comparison to simulations run on a single processor, and did not introduce errors in numerical results.

The algorithm was also shown to be capable of simulating and capturing the dynamics of a cell population in a fluctuating environment, where phenotypic variability strongly influences gene expression dynamics. Agreement between this framework and the experimental and theoretical results obtained using a deterministic reaction-rate method in Acar *et al.* [77], serves as a further benchmark for the proposed method. Furthermore, the algorithm's ability to capture the steady-state and time-dependent phenotypic distributions in this system exemplifies the utility of this approach, as these distributions cannot be obtained using a deterministic framework.

Current cellular population simulation methods, including the present algorithm, treat the extracellular environment as homogeneous (e.g. the spatial-temporal concentration profile of a nutrient required for growth is held constant). This prohibits, for example, the inclusion of competition for a limiting resource in the present implementation. However, it is possible to model feedback between cells and their environment. The simplest approach would be to assume that the environment is constant over short time intervals. The change in total population cell volume at the end of each interval could then be used to calculate how much nutrients have been consumed and the parameters describing the environment adjusted accordingly. Since the time intervals would have to be sufficiently short so that the change in concentration of the nutrient during any particular interval is negligible, the computational workload would increase substantially. The focus of future work will be on developing and benchmarking accurate and efficient augmentations that permit population simulators to handle these and other more complex scenarios.

References

[1] D.A. Charlebois, J. Intosalmi, D. Fraser, M. Kaern, 'An Algorithm for the Stochastic Simulation of Gene Expression and Heterogeneous Population Dynamics', Commun. Comput. Phys. 9 (2011) 89-112.

[2] D.T. Gillespie, 'Exact stochastic simulation of coupled chemical reactions', J. Phys. Chem. 81 (1977) 2340-2361.

[3] D.T. Gillespie, 'Stochastic Simulation of Chemical Kinetics', Annu. Rev. Phys. Chem. 58 (2007) 35-55.

[4] D.T. Gillespie, 'Markov processes: an introduction for physical scientists', London: Academic Press Limited, 1992.

[5] M.S. Samoilov, G. Price, A.P. Arkin, 'From fluctuations to Phenotypes: The Physiology of Noise', Sci. STKE 366 (2006) re17.

[6] M. Kaern, T.C. Elston, W.J. Blake, J.J. Collins, 'Stochasticity in gene expression: From theories to phenotypes', Nat. Rev. Genet. 6 (2005) 451-464.

[7] B.B. Kaufmann, A. van Oudenaarden, 'Stochastic gene expression: from single molecules to the proteome', Curr. Opin. Genet. Dev. 17 (2007) 107-112.

[8] N. Maheshri, E.K. O'Shea, 'Living with noisy genes: how cells function reliably with inherent variability in gene expression', Annu. Rev. Biophys. Biomol. Struct. 36 (2007) 413-434.

[9] J. Paulsson, 'Summing up the noise in gene networks', Nature 427 (2004) 415-418.

[10] M. Scott, B. Ingalls, M. Kaern, 'Estimations of intrinsic and extrinsic noise in models of nonlinear genetic networks', Chaos 16 (2006) 026107.

[11] L. Ma, J. Wagner, J.J. Rice, H. Wenwei, J.L. Arnold, G.A. Stolovitzky, 'A plausible model for the digital response of p53 to DNA damage', PNAS 102 (2005) 14266-14271.

[12] A.S. Ribeiro, R. Zhu, S.A. Kauffman, 'A General Modeling Strategy for Gene Regulatory Networks with Stochastic Dynamics', J. Comput. Biol. 13 (2006) 1630-1639.

[13] P.S. Swain, M.B. Elowitz, E.D. Siggia, 'Intrinsic and extrinsic contributions to stochasticity in gene expression', PNAS 99 (2002) 12795-12800.

[14] M. Roussel, R. Zhu, 'Validation of an algorithm for the delay stochastic simulation of transcription and translation in prokaryotic gene expression', Phys. Biol. 3 (2006) 274-284.

[15] T.E. Turner, S. Schnell, K. Burrage, 'Stochastic approaches for modelling in vivo reactions', Comput. Biol. Chem. 28 (2004) 165-178.

[16] D.J. Wilkinson, 'Stochastic Modelling for Systems Biology', Boca Raton: Chapman & Hall, 2006.

[17] J. Paulsson, 'Noise in a minimal regulatory network: plasmid copy number control', Quart. Rev. Biophys. 34 (2001) 1-59.

[18] D.J. Wilkinson, 'Stochastic modelling for quantitative description of heterogeneous biological systems', Nat. Rev. Genet. 10 (2009) 122-133.

[19] R.L. Bar-Or, R. Maya, L.A. Segel, U. Alon, *et al.*, 'Generation of oscillations by the p53-Mdm2 feedback loop: A theoretical and experimental study', PNAS 97 (2000) 11250-11255.

[20] N. Geva-Zatorsky, N. Rosenfeld, S. Itzkovitz, R. Milo, *et al.*, 'Oscillations and variability in the p53 system', Mol. Syst. Biol. 2 (2006) doi:10.1038/msb4100068.

[21] G. Lahav, N. Rosenfeld, A. Sigal, N. Geva-Zatorski, *et al.*, 'Dynamics of the p53-mdm2 feedback loop in individual cells', Nat. Genet. 36 (2004) 147-150.

[22] N.A.M. Monk, 'Oscillatory Expression of Hes1, p53, and NFκB Driven by Transcriptional Time Delays', Curr. Biol. 13 (2003) 1409-1413.

[23] L.J. Zhang, S.W. Yan, Y.Z. Zhuo, 'A dynamical model of DNA-damage derived p53mdm2 interaction', Acta Physica Sinica 56 (2007) 2442-2447.

[24] K. Puszynski, B. Hat, T. Lipniacki, 'Oscillations and bistability in the stochastic model of p53 regulation', J. Theor. Biol. 254 (2008) 452-465.

[25] A.S. Ribeiro, D.A. Charlebois, J. Lloyd-Price, 'CellLine, a stochastic cell lineage simulator', Bioinf. 23 (2007) 3409-3411.

[26] A. Brock, H. Chang, S. Huang, 'Non-genetic heterogeneity - a mutation-independent driving force for the somatic evolution of tumours', Nat. Rev. Genet. 10 (2009) 336-342.

[27] D. Fraser, M. Kaern, 'A chance at survival: gene expression noise and phenotypic diversification strategies', Molec. Microbiol. 71 (2009) 1333-1340.

[28] M.B. Elowitz, A.J. Levine, E.D. Siggia, P.S. Swain, 'Stochastic gene expression in a single cell', Science 297 (2002) 1183-1186.

[29] J.M. Pedraza, A. van Oudenaarden, 'Noise propagation in gene networks', Science 307 (2005) 1965-69.

[30] J.M. Raser, E.K. O'Shea, 'Control of stochasticity in eukaryotic gene expression', Science 304 (2004) 1811-1814.

[31] J.M. Raser, E.K. O'Shea, 'Noise in gene expression: origins, consequences, and control', Science 309 (2005) 2010-2013.

[32] D. Volfson, J. Marciniak, W.J. Blake, N. Ostroff, *et al.*, 'Origins of extrinsic variability in eukaryotic gene expression', Nature 439 (2006) 861-64.

[33] M. Thattai, A. van Oudenaarden, 'Intrinsic noise in gene regulatory networks', PNAS 98 (2001) 8614-8619.

[34] M. Thattai, A. van Oudenaarden, 'Stochastic Gene Expression in Fluctuating Environments', Genetics 167 (2004) 523-530.

[35] E.M. Ozbudak, M. Thattai, I. Kurtser, A.D. Grossman, *et al.*, 'Regulation of noise in the expression of a single gene', Nat. Genet. 31 (2002) 69-73.

[36] L. Lopez-Maury, S. Marguerat, J. Bahler, 'Tuning gene expression to changing environments: from rapid response to evolutionary adaptation', Nat. Rev. Gen. 9 (2008) 583-593.

[37] W.K. Smits, O.P. Kuipers, J.W. Veening, 'Phenotypic variation in bacteria: the role of feedback regulation', Nat. Rev. Microbiol. 4 (2006) 259-271.

[38] J.W. Veening, W.K. Smits, O.P. Kuipers, 'Bistability, epigenetics, and bet-hedging in bacteria', Annu. Rev. Microbiol. 62 (2008) 193-210.

[39] A. Bar-Even, J. Paulsson, N. Maheshri, M. Carmi, *et al.*, 'Noise in protein expression scales with natural protein abundance', Nat. Genet. 38 (2006) 636-643.

[40] J.R.S. Newman, S. Ghaemmaghami, J. Ihmels, D.K. Breslow, *et al.*, 'Single-cell proteomic analysis of *S. cerevisiae* reveals the architecture of biological noise', Nature 441 (2006) 840-846.

[41] H.B. Fraser, A.E. Hirsh, G. Giaever, J. Kumm, *et al.*, 'Noise minimization in eukaryotic gene expression', PLoS Biol. 2 (2004) e137.

[42] D.T. Gillespie, 'Stochastic Chemical Kinetics', Published as Section 5.11 of Handbook of Materials and Modeling, S. Yip, Ed., Berlin, Heidelberg, New York: Springer Dochdrecht, 2005.

[43] D.T. Gillespie, 'A general method for numerically simulating the stochastic time evolution of coupled chemical reactions', J. Comput. Phys. 22 (1976) 403-434.

[44] N.V. Mantzaris, 'From Single-Cell Genetic Architecture to Cell Population Dynamics: Quantitatively Decomposing the Effects of Different Population Heterogeneity Sources for a Genetic Network with Positive Feedback Architecture', Biophys. J. 92 (2007) 4271-4288.

[45] Y. Lin, K. Lee, T. Matsoukas, 'Solution of the population balance equation using constant-number Monte Carlo', Chem. Eng. Sci. 57 (2002) 2241-2252.

[46] M. Smith, T. Matsoukas, 'Constant-number Monte Carlo simulation of population balances', Chem. Eng. Sci. 53 (1998) 1777-1786.

[47] N.V. Mantzaris, 'Stochastic and deterministic simulations of heterogeneous cell population dynamics', J. Theor. Biol. 241 (2006) 690-706.

[48] N.G. van Kampen, 'Stochastic Processes in Physics and Chemistry', Amsterdam: North-Holland, 1992.

[49] D.S. Lemons, 'An Introduction to Stochastic Processes in Physics', Baltimore and London: Johns Hopkins University Press, 2002.

[50] V. Shahrezaei, P.S. Swain, 'Analytical distributions for stochastic gene expression', PNAS 105 (2008) 17256-17261.

[51] N.G. van Kampen, 'A Power Series Expansion of the Master Equation', Can. J. Phys. 39 (1961) 551.

[52] N.G. van Kampen, 'The expansion of the master equation', Adv. Chem. Phys. 34 (1976) 245.

[53] J. Elf, M. Ehrenberg, 'Fast Evaluation of Fluctuations in Biochemical Networks With the Linear Noise Approximation', Genome Res. 13 (2003) 2475-2484.

[54] J. Elf, J. Paulsson, O.G. Berg, M. Ehrenberg, 'Near-Critical Phenomena in Intracellular Metabolite Pools', Biophys. J. 84 (2003) 154-170.

[55] R. Tomioka, H. Kimura, T.J. Kobayashi, K. Aihara, 'Multivariate analysis of noise in genetic regulatory networks', J. Theor. Biol. 229 (2004) 501-521.

[56] T. Manninen, M.-L. Linne, K. Ruohonen, 'Developing Itô stochastic differential equation models for neuronal signal transduction pathways', Compt. Biol. Chem. 30 (2006) 280-291.

[57] J. Intosalmi, 'On Stochastic Differential Equations: Theory and Biochemical Applications', Published Thesis (MSc), University of Tampere, Tampere, Finland, 2007.

[58] I. Karatzas, S.E. Shreve, 'Brownian Motion and Stochastic Calculus', New York: Springer-Verlag, 1988.

[59] B. Øksendal, 'Stochastic Differential Equations', Berlin: Springer, 2007.

[60] H. Risken, 'Fokker-Plank Equations', Berlin: Springer, 1992.

[61] C.W. Gardiner, 'Handbook of Stochastic Methods', Berlin: Springer, 1992.

[62] R. Murugan, 'Multiple Stochastic Point Processes in Gene Expression', J. Stat. Phys. 131 (2008) 153-165.

[63] D.T. Gillespie, 'Approximate accelerated stochastic simulation of chemically reacting systems', J. Chem. Phys. 115 (2001) 1716-1733.

[64] M.A. Gibson, J. Bruck, 'Exact stochastic simulation of chemical systems with many species and many channels', J. Phys. Chem. 105 (2000) 1876-1889.

[65] T. Lu, D. Volfson, L. Tsimring, J. Hasty, 'Cellular growth and division in the Gillespie algorithm', Syst. Biol. 1 (2004) 121-128.

[66] D. Adalsteinsson, D. McMillen, T.C. Elston, 'Biochemical Network Stochastic Simulator (BioNetS): software for stochastic modeling of biochemical networks', BMC Bioinfo. 5 (2004) 24.

[67] A.M. Kierzek, 'STOCKS: STOChastic Kinetic Simulations of biochemical systems with Gillespie algorithm', Bioinf. 18 (2002) 470-481.

[68] J.J. Tyson, O.J. Diekmann, 'Sloppy size control of the cell division cycle', Theor. Biol. 118 (1986) 405-426.

[69] N. Rosenfeld, T.J. Perkins, U. Alon, M.B. Elowitz, P.S. Swain, 'A Fluctuation Method to Quantify In Vivo Fluorescence Data', Biophys. J. 91 (2006) 759-766.

[70] D. Ramkrishna, 'The status of population balances', Rev. Chem. Engng. 3 (1985) 49-95.

[71] K. Lee, T. Matsoukas, 'Simultaneous coagulation and break-up using constant-N Monte Carlo', Powder Technol. 110 (2000) 82-89.

[72] T.B. Kepler, T.C. Elston, 'Stochasticity in Transcriptional Regulation', Biophys. J. 81 (2001) 3116-3136.

[73] D. Cohen, 'Optimizing reproduction in a randomly varying environment', J. Theor. Biol. 12 (1966) 119-129.

[74] R. Levins, 'Evolution in Changing Environments: some Theoretical Explorations', New Jersey: Princeton University Press, 1968.

[75] W.M. Schaffer, 'Optimal efforts in fluctuating environments', Am. Nat. 108 (1974) 783-790.

[76] S.C. Stearns, 'Life-history tactics: a review of the ideas', Q. Rev. Biol. 51 (1976) 3-47.

[77] M. Acar, J.T. Mettetal, A. van Oudenaarden, 'Stochastic switching as a survival strategy in fluctuating environments', Nat. Genet. 40 (2008) 471-475.

[78] B.J. Brewer, E. Chlebowicz-Sledziewska, W.L. Fangman, 'Cell Cycle Phases in the Unequal Mother/Daughter Cell Cycles of *Saccharomyces cerevisiae*', Mol. Cell. Biol. 4 (1984) 2529-2531.

[79] J. Fisher, T.A. Henzinger, 'Executable cell biology', Nat. Biotechnol. 25 (2007) 1239-1249.

[80] A. Longtin, 'Physique numérique stochastique', lecture notes, University of Ottawa, 2009.

[81] J. Paulsson, 'Models of stochastic gene expression', Phys. Life Rev. 2 (2005) 157-75.

Appendices

Appendix A: Poisson Process

An understanding of the Poisson process is key to understanding more sophisticated stochastic processes. For example, the stochastic processes of transcription and translation can under certain conditions (i.e. a fixed number of active promoters and mRNAs, respectively) be described by a birth-death process whose steady-state can be described by the Poisson distribution. The Poisson distribution, a discrete probability distribution (a limit of a binomial distribution when the number of trials $n \to \infty$ and the probability of individual success $p \to 0$ in such a way that the product np remains constant) of the number of independent events found in a limited region, is particularly important in the stochastic modelling of biochemical networks, as the number of reaction events occurring in a short time interval is approximately Poisson. In physics, this distribution has often been used to describe radioactive decay experiments where there may be a very large number of trials (e.g., a reading for each microsecond when the counter is on), but a low probability of a decay event occurring within this one microsecond [16, 48, 80].

A Poisson random variable, X with parameter λ is written as

$$X \sim Po(\lambda) \tag{1}$$

and the corresponding probability mass function (PMF) of X as

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k = 0, 1, 2, \ldots \tag{2}$$

A PMF is defined for any discrete random variable X to be the function which gives the probability of each $k \in S_X$, where S_X is the sample space. Note that for smaller values of λ (e.g. $\lambda = 2$ shown in Fig. 1a) can be quite asymmetric and in this way quite different from a Gaussian distribution (which it approaches for $\lambda >> 1$ - this can be seen in Fig. 1b where $\lambda = 8$). If $X \sim Po(\lambda)$, the expectation and variance

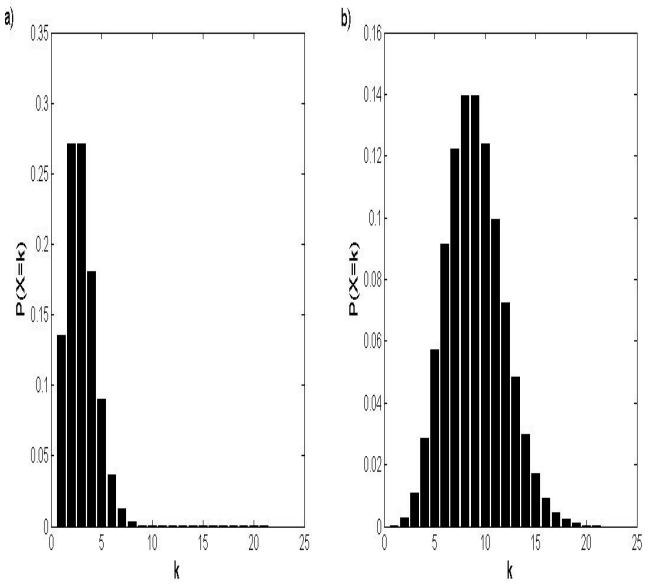

Figure 1 Probability mass function for a Poisson ($\lambda = 2$) distribution (a) and Poisson ($\lambda = 8$) distribution (b).

can be computed directly from the PMF, or can be obtained from the binomial limit, and are as follows [16]

$$E(X) = \lambda \text{ and } Var(X) = \lambda \tag{3}$$

That is, the mean and variance are both found to be λ.

A sequence of timed observations follows a Poisson process with rate λ if the number of observations, X, in any interval of length t is such that

$$X \sim Po(\lambda t) \tag{4}$$

and the numbers of events in disjoint intervals are independent of each another.

Transcription and translation are often assumed to follow Poisson processes where the production probabilities per unit time are proportional to the number of active genes and mRNAs [81]. Thus, the size of fluctuations in the number of a certain

molecule type (e.g. mRNA) is assumed to be of the order of the square root of the average (e.g. $\sqrt{\overline{mRNA}}$), as is often approximately the case at thermodynamic equilibrium [17]. Thus the probability of observing a significant deviation from the average decreases as the number of molecules in the reaction volume increases.

Appendix B: Fortran 90 Code

This parallel Fortran 90 program simulates the population dynamics (includes single cell tracking and statistics) of gene regulatory networks (default GRN, a single gene with no feedback in *E. coli.*, and default theoretical formulas for benchmarking are from Swain *et al.* [13]) using the Gillespie direct method stochastic simulation algorithm [2, 43] and the constant-number Monte Carlo method [45, 46].

Parallelism is here implemented across the simulation and designed for execution in a shared memory multiprocessor environment using OpenMP. The parallel region includes the execution of the SSA and growth and division for each cell of the population.

Note that the version of the code that follows was used to obtain the results in Section 4.1.3 and incorporates gene duplication, cell growth and division, and time-dependent analytical solutions for benchmarking, and can be modified to reproduce any of the results presented in this thesis.

```
!**MAIN CALLER PROGRAM**

!‘CELL_PASSED.f90’ is the main caller program which seeds the random number
!generator and calls the SSA from module ‘SSA_Parallel.f90’ in order to
!simulate the dynamics of a GRN and output the results at the user defined
!sampling interval.

!************************************************
PROGRAM CELL_PASSED_Parallel
use SSA_Parallel
implicit none
integer, allocatable :: iseed(:)
integer :: idate(8), isize
integer :: nc=8000 !number of cells
real :: t_cumm=0 !simulation start time

call DATE_AND_TIME(VALUES=idate)
print *, "Simulation start date/time = ", idate

!obtain random seed from date_and_time intrinsic
call random_seed(SIZE=isize)
allocate(iseed(isize))
call random_seed(GET=iseed)
iseed = iseed * (idate(8)-500) !idate(8) contains millisecond
call random_seed(PUT=iseed)
```

```
deallocate(iseed)

call main(nc,t_cumm)

call DATE_AND_TIME(VALUES=idate)
print *, "Simulation end date/time = ", idate

END PROGRAM CELL_PASSED_Parallel
!*************************************************

!**SSA_Parallel MODULE**

!This module contains the constant-number MC in subroutine 'main' which
calls the SSA in subroutine 'next_reaction'. Subroutine 'next_reaction'
in turn calls the module 'reactions_Parallel' in order to simulate gene
expression for the user defined GRN after it determines the next reaction
from the reaction propensities calculated from a call to the subroutine
'propensities'. Note that module 'results_Parallel' is called every
'sample_intervl' in order to calculate statistics and output/save results.

!*************************************************
MODULE SSA_Parallel
use reactions_Parallel
use globals_Parallel
use results_Parallel
implicit none
```

```
contains


SUBROUTINE main(nc,t_cumm)
use globals_Parallel
!random number arrays for r1 and r2
real, dimension(n_threads,num_rand) :: rand_array1,rand_array2
!array to store random number counters for each thread
real, dimension(n_threads) :: rand_cnt_array
!array to store data for cell(s) with index
![t_sample,P,M,D,k,mu,cell_vol,t_last_div_array(k),t_cumm,cell_age,mother_ID]
real, dimension(nc,11) :: u
!temporary mother and daughter cell arrays
real, dimension(11) :: u_temp_mothers, u_temp_daughters
real, dimension(2,11) :: u_daughters_temp
!array to store data for cell(s):
real, dimension(nc,11) :: u_daughters
!array to store molecular species values: P,Act,unAct
real, dimension(1,3) :: s
!array to store reaction propensities
real, dimension(1,4) :: a
!array to store intial last division times
real, dimension(nc)  :: t_last_div_array, cell_vol_array
!sample interval,simulation time,next reaction time
real :: t_sample=0,t_cumm,dt=0,t_marker
!initial conditions
real :: D,M=0,P=0,cell_age=0,mother_ID=0
```

```
!sum of reaction propensities,value holder for next reaction
real :: a0=0, amu=0
!counter for random numbers,cell I.D.,next reaction marker
integer :: num_rand,k=0,mu=0,i=0,result_cnt,nc,rand_cnt=0
!statistics run counter,OpenMP variables
integer :: stat_cnt=0,TID,OMP_GET_THREAD_NUM
!logical to indicate to SSA whether to simulate mother or daughter cells
integer :: mother_sim,daughter_sim
!counter for number of births since last constant-number MC restore event
integer :: daughter_cnt,d_count
!Constant-number algorithm variables
real, dimension(10*nc)  :: CNA_array
integer, dimension(nc) :: CNA_rand_index_array
real :: CNA_rand
integer :: j,jj,l,ii,t_restore_cnt,CNA_rand_index,CNA_rand_index_cnt
integer :: CNA_cnt1,CNA_cnt2,mRNA_cnt_mother=0,mRNA_cnt_daughter=0
integer :: protein_cnt_mother=0,protein_cnt_daughter=0

!file to store time series for selected cell(s)
  open(1,file='time_series_Parallel.dat')
  !file to store analytical results
  open(2,file='analytical_stats_Parallel.dat')
  !file to store population statistics
  open(3,file='population_stats_Parallel.dat')

  !obtain theoretical values for cell(s)
```

```
  call stats_theoretical()

  !initialize arrays
  rand_array1=0;rand_array2=0;rand_cnt_array=1;a=0;s=0;u=0;u_temp_mothers=0;
  u_daughters=0;u_temp_daughters=0

  !initialize logicals
mother_sim=1; daughter_sim=0

  !intialize misc.
t_restore_cnt=1;d_count=0;daughter_cnt=0

  !generate uniform distribution of time since last division for initial pop.
call random_number(t_last_div_array)
t_last_div_array=t_div*t_last_div_array

  !intial values for cell(s)
do k=1,nc
  !calculate initial cell volumes (Linear)
cell_vol_array(k)=init_cell_vol !*(1+t_last_div_array(k)/t_div)
!calculate initial cell volumes (Exponential)
!cell_vol_array(k)=init_cell_vol*EXP(log(2.)*t_last_div_array(k)/t_div)
u(k,1)=0; u(k,2)=0; u(k,3)=0; u(k,4)=1; u(k,5)=k; u(k,6)=0;
u(k,7)=cell_vol_array(k); u(k,8)=t_last_div_array(k); u(k,9)=t_cumm;
u(k,10)=cell_age;u(k,11)=mother_ID
!u(k,8)=0 !note: to start all cell in phase
```

```
end do

  !fill random number arrays and initialize counter
call random_number(rand_array1)
call random_number(rand_array2)
rand_cnt=1

!obtain initial values for time series and statistics for cell(s)
!call time_series(u,u_daughters,nc,daughter_cnt,t_cumm)
stat_cnt=stat_cnt+1
call pop_stats(u,u_daughters,nc,daughter_cnt,t_cumm)

t_sample=sample_intervl

  !*note: ensure t_sample is set such that t_cumm will always be < t_sample*
do while (t_cumm < t_end)
  !note: t_marker is time at begining of this t_sample (so that each cell
  !starts at the beggining of interval each iteration)
t_marker=t_cumm
if (t_cumm < t_sample) then
!PARALLEL REGION: simulate dynamics of mother cells
!$OMP PARALLEL PRIVATE(D,M,P,k,t_cumm,dt,s,a,a0,amu,mu,i,u_temp_mothers,
u_temp_daughters,mRNA_cnt_mother,mRNA_cnt_daughter,protein_cnt_mother,
protein_cnt_daughter) NUM_THREADS(n_threads)
!$OMP DO
do k=1,nc
```

```
u_temp_mothers = u(k,1:11)
call next_reaction(t_cumm,t_sample,t_marker,D,M,P,rand_array1,rand_array2,
k,nc,rand_cnt_array,dt,s,a,a0,amu,mu,i,u_temp_mothers,u_temp_daughters,
u_daughters,daughter_cnt,mother_sim,daughter_sim,d_count,mRNA_cnt_mother,
mRNA_cnt_daughter,protein_cnt_mother,protein_cnt_daughter)
u(k,1:11) = u_temp_mothers
end do
!$OMP END DO
!$OMP END PARALLEL

!set logicals
mother_sim=0; daughter_sim=1

!PARALLEL REGION: simulate dynamics of daughter cells
if (daughter_cnt >= 1) then
!$OMP PARALLEL PRIVATE(D,M,P,k,t_cumm,dt,s,a,a0,amu,mu,i,u_temp_mothers,
u_temp_daughters,mRNA_cnt_mother,mRNA_cnt_daughter,
protein_cnt_mother,protein_cnt_daughter) NUM_THREADS(n_threads)
!$OMP DO
do d_count=1,daughter_cnt
u_temp_daughters = u_daughters(d_count,1:11)
call next_reaction(t_cumm,t_sample,t_marker,D,M,P,rand_array1,rand_array2,
k,nc,rand_cnt_array,dt,s,a,a0,amu,mu,i,u_temp_mothers,u_temp_daughters,
u_daughters,daughter_cnt,mother_sim,daughter_sim,d_count,mRNA_cnt_mother,
mRNA_cnt_daughter,protein_cnt_mother,protein_cnt_daughter)
u_daughters(d_count,1:11) = u_temp_daughters
```

```
end do
!$OMP END DO
!$OMP END PARALLEL
end if

!reset logicals
mother_sim=1;daughter_sim=0

t_cumm=t_sample
t_sample=t_sample+sample_intervl
!obtain time series for cell(s)
!call time_series(u,u_daughters,nc,daughter_cnt,t_cumm)

stat_cnt=stat_cnt+1
!obtain population statistics for cell(s)
call pop_stats(u,u_daughters,nc,daughter_cnt,t_cumm)
end if

!Constant Number Algorithm
if (daughter_cnt >= 1) then
if (t_restore*t_restore_cnt <= t_cumm) then
!sort daughter cell array (oldest to youngest)
u_daughters_temp=0
if (daughter_cnt >= 2) then
ii=1
do while (ii <= daughter_cnt-1)
```

```
CNA_cnt1=u_daughters(ii,10)
CNA_cnt2=u_daughters(ii+1,10)
if (CNA_cnt2 > CNA_cnt1) then
do k=1,11
u_daughters_temp(1,1:11)=u_daughters(ii,1:11)
u_daughters_temp(2,1:11)=u_daughters(ii+1,1:11)
end do
u_daughters(ii,1:11)=u_daughters_temp(2,1:11)
u_daughters(ii+1,1:11)=u_daughters_temp(1,1:11)
ii=1
else
ii=ii+1
end if
end do
end if

!replace random cells in mother cell array with cells from daughter
!array (oldest daughter cells being inserted first)
CNA_array=0; call random_number(CNA_array); CNA_rand_index_cnt=0
do j=1,daughter_cnt
CNA_rand_index_cnt=CNA_rand_index_cnt+1
     CNA_rand_index_array(j)=nint(CNA_array(CNA_rand_index_cnt)*nc)
!ensure that random index array is non-zero
!(i.e. there must be a cell to replace!)
do while (CNA_rand_index_array(j)==0)
CNA_rand_index_cnt=CNA_rand_index_cnt+1
```

```
CNA_rand_index_array(j)=nint(CNA_array(CNA_rand_index_cnt)*nc)
!code to track first cell for Swain validation
!do while (CNA_rand_index_array(j)==1)
! CNA_rand_index_cnt=CNA_rand_index_cnt+1
! CNA_rand_index_array(j)=nint(CNA_array(CNA_rand_index_cnt)*nc)
!end do
end do
u(CNA_rand_index_array(j),1:11)=u_daughters(j,1:11)
end do

daughter_cnt=0; u_daughters=0
t_restore_cnt=t_restore_cnt+1
end if
end if

end do

close(1); close(2); close(3)

END SUBROUTINE main
!*********************************

!*********************************
SUBROUTINE next_reaction(t_cumm,t_sample,t_marker,D,M,P,rand_array1,
rand_array2,k,nc,rand_cnt_array,dt,s,a,a0,amu,mu,i,u_temp_mothers,
u_temp_daughters,u_daughters,daughter_cnt,mother_sim,daughter_sim,d_count,
```

```
mRNA_cnt_mother,mRNA_cnt_daughter,protein_cnt_mother,protein_cnt_daughter)
use globals_Parallel
implicit none
real, dimension(n_threads,num_rand) :: rand_array1, rand_array2
real, dimension(n_threads) :: rand_cnt_array
real, dimension(nc,11) :: u_daughters
real, dimension(11) :: u_temp_mothers, u_temp_daughters
real, dimension(1,3) :: s
real, dimension(1,4) :: a
real :: dt,t_cumm,t_sample,t_marker
real :: D,M,P,a0,amu,g1,g2,aa=0.0,bb=0.5
integer :: k,i,nc,mu,idate(8),TID,OMP_GET_THREAD_NUM
integer :: mother_sim,daughter_sim,daughter_cnt,d_count,mRNA_cnt_mother,
mRNA_cnt_daughter,protein_cnt_mother,protein_cnt_daughter

!Mothers
if (mother_sim==1) then
P=u_temp_mothers(2)
M=u_temp_mothers(3)
!Gene Duplication 'On' (Swain)
if (u_temp_mothers(8) >= 0.4*t_div) then
D=2*D_init
elseif (u_temp_mothers(8) < 0.4*t_div) then
D=D_init
end if
!D=D_init !Gene Duplication 'Off'
```

```
t_cumm=t_marker
end if

!Daughters
if (daughter_sim==1) then
P=u_temp_daughters(2)
M=u_temp_daughters(3)
!Gene Duplication 'On' (Swain)
if (u_temp_daughters(8) >= 0.4*t_div) then
D=2*D_init
elseif (u_temp_daughters(8) < 0.4*t_div) then
D=D_init
end if
!D=D_init !Gene Duplication 'Off'
t_cumm=u_temp_daughters(9)
end if

TID = OMP_GET_THREAD_NUM()

do while (t_cumm < t_sample)
s(1,1)=P
s(1,2)=M
s(1,3)=D

call propensities(s,a)
a0 = sum(a)
```

```
!check and replenish random number arrays as required
if (rand_cnt_array(TID+1) >= num_rand) then
!print *, 'Replenishing rand_arrays', k
call random_number(rand_array1(TID+1,num_rand))
call random_number(rand_array2(TID+1,num_rand))
rand_cnt_array(TID+1)=1
end if

!calculate time of next reaction
dt = log(1./rand_array1(TID+1,rand_cnt_array(TID+1)))/a0

!update time and time since last division
t_cumm = t_cumm+dt

if (mother_sim==1) then
u_temp_mothers(8)=u_temp_mothers(8)+dt
u_temp_mothers(9) = u_temp_mothers(9)+dt
u_temp_mothers(10)=u_temp_mothers(10)+dt
else
u_temp_daughters(8)=u_temp_daughters(8)+dt
u_temp_daughters(9) = u_temp_daughters(9)+dt
u_temp_daughters(10)=u_temp_daughters(10)+dt
end if

!cell volume
```

```
if (mother_sim==1) then
  !Linear growth (Swain)
u_temp_mothers(7)=init_cell_vol*(1+(u_temp_mothers(8)/t_div))
!Exponential growth (Kaern)
!u_temp_mothers(7)=init_cell_vol*EXP(log(2.)*u_temp_mothers(8)/t_div)
!division based on cell volume
if (u_temp_mothers(7) >= 2*init_cell_vol) then
!division based on fixed division time (t_div)
!if (u_temp_mothers(8) >= t_div+sample_intervl) then
!check and replenish random number arrays as required
if (rand_cnt_array(TID+1)+1 >= num_rand) then
!print *, 'Replenishing rand_arrays', k
call random_number(rand_array1(TID+1,num_rand))
call random_number(rand_array2(TID+1,num_rand))
rand_cnt_array(TID+1)=1
end if
daughter_cnt=daughter_cnt+1
!rand_cnt_array(TID+1)=rand_cnt_array(TID+1)+1
!Box-Muller transform
!g1=sqrt(-2*log(rand_array1(TID+1,rand_cnt_array(TID+1))))
!*COS(2*pi*rand_array2(TID+1,rand_cnt_array(TID+1)+1))
!g1=(aa*g1)+bb

!conservation of mass equations
!if (mod(u_temp_mothers(2),2.)==0) then
P=u_temp_mothers(2)
```

```
!Uniform
!u_temp_mothers(2)=nint(rand_array1(TID+1,rand_cnt_array(TID+1))
!*P)u(k,8)=t_last_div_array(k)
!u_daughters(daughter_cnt,2)=nint((1-rand_array1
!(TID+1,rand_cnt_array(TID+1)))*P)
!Gaussian
!u_temp_mothers(2)=nint(g1*P)
!u_daughters(daughter_cnt,2)=nint((1-g1)*P)
!P=u_temp_mothers(2)
  !Binomial
  protein_cnt_mother=0
  protein_cnt_daughter=0
  do i=1,P
  !check and replenish random number arrays as required
  if (rand_cnt_array(TID+1) >= num_rand) then
   call random_number(rand_array1(TID+1,num_rand))
  rand_cnt_array(TID+1)=1
  end if
  rand_cnt_array(TID+1)=rand_cnt_array(TID+1)+1
  if (rand_array1(TID+1,rand_cnt_array(TID+1))>0.5) then
  protein_cnt_mother=protein_cnt_mother+1
  else
  protein_cnt_daughter=protein_cnt_daughter+1
  end if
  end do
  u_temp_mothers(2)=protein_cnt_mother; P=u_temp_mothers(2)
```

```
  u_temp_daughter(2)=protein_cnt_daughter


!elseif (mod(u_temp_mothers(2),2.)/=0) then
        ! if (nint(rand_array1(TID+1,rand_cnt_array(TID+1)))==0) then
! u_temp_mothers(2)=nint(0.5*u_temp_mothers(2)); P=u_temp_mothers(2)
! u_daughters(daughter_cnt,2)=P-1
! else
! u_temp_mothers(2)=nint(0.5*u_temp_mothers(2))-1; P=u_temp_mothers(2)
! u_daughters(daughter_cnt,2)=P+1
! end if
!end if


!if (mod(u_temp_mothers(3),2.)==0) then
!rand_cnt_array(TID+1)=rand_cnt_array(TID+1)+1
!g2=sqrt(-2*log(rand_array1(TID+1,rand_cnt_array(TID+1))))
!*SIN(2*pi*rand_array2(TID+1,rand_cnt_array(TID+1)+1))
!g2=(aa*g2)+bb
M=u_temp_mothers(3)
!Uniform
!u_temp_mothers(3)=nint(rand_array1(TID+1,rand_cnt_array(TID+1))*M)
!u_daughters(daughter_cnt,3)=nint((1-rand_array1(TID+1,rand_cnt_array(TID+1)))*M)
!Gaussian
!u_temp_mothers(3)=nint(g2*M)
!u_daughters(daughter_cnt,3)=nint((1-g2)*M)
!M=u_temp_mothers(3)
!Binomial
```

```
mRNA_cnt_mother=0
mRNA_cnt_daughter=0
do i=1,M
!check and replenish random number arrays as required
if (rand_cnt_array(TID+M) >= num_rand) then
call random_number(rand_array1(TID+1,num_rand))
call random_number(rand_array2(TID+1,num_rand))
rand_cnt_array(TID+1)=1
end if
rand_cnt_array(TID+1)=rand_cnt_array(TID+1)+1
if (rand_array1(TID+1,rand_cnt_array(TID+1))>0.5) then
mRNA_cnt_mother=mRNA_cnt_mother+1
else
mRNA_cnt_daughter=mRNA_cnt_daughter+1
end if
end do
u_temp_mothers(3)=mRNA_cnt_mother; M=u_temp_mothers(3)

!else
! if (nint(rand_array2(TID+1,rand_cnt_array(TID+1)))==0) then
! u_daughters(daughter_cnt,3)=nint(0.5*u_daughters(daughter_cnt,3))
!   M=u_temp_mothers(3)
! u_daughters(daughter_cnt,3)=M-1
! else
! u_temp_mothers(3)=nint(0.5*u_temp_mothers(3))-1; M=u_temp_mothers(3)
! u_daughters(daughter_cnt,3)=M+1
```

```
! end if
!end if

u_temp_mothers(4)=u_temp_mothers(4)/2 !gene duplication 'on'
u_temp_mothers(7)=init_cell_vol
u_temp_mothers(8)=0

u_daughters(daughter_cnt,1)=u_temp_mothers(1)
u_daughters(daughter_cnt,4)=u_temp_mothers(4)
!note: correct daughter cell number (may not be same as 'k')
u_daughters(daughter_cnt,5)=daughter_cnt
u_daughters(daughter_cnt,6)=u_temp_mothers(6)
u_daughters(daughter_cnt,7)=u_temp_mothers(7)
u_daughters(daughter_cnt,8)=0
u_daughters(daughter_cnt,9)=t_cumm
u_daughters(daughter_cnt,10)=0
u_daughters(daughter_cnt,11)=k
!snap shot at moment of division
!print*, 'mother', u_temp_mothers
!print*, 'daughter', u_daughters(daughter_cnt,1:11)
end if
else
  !Linear growth (Swain)
u_temp_daughters(7)=init_cell_vol*(1+(u_temp_daughters(8)/t_div))
!Exponential growth (Kaern)
!u_temp_daughters(7)=init_cell_vol*EXP(log(2.)*u_temp_daughters(8)/t_div)
```

```
!division based on cell volume
if (u_daughters(d_count,7) >= 2*init_cell_vol) then
!division based on fixed division time (t_div)
!if (u_daughters(d_count,8) >= t_div) then
write(1,*) 'DAUGHTER CELL HAS DIVIDED!
SET t_restore < SHORTEST CELL DIVISION TIME!'
end if
end if

rand_cnt_array(TID+1)=rand_cnt_array(TID+1)+1
!calculate which reaction occurs next (if outside of dimension of
!rand_array2 then replenish with random numbers)
i=1; mu=0; amu=0 !set i=1 for each cyle
do while (amu < rand_array2(TID+1,rand_cnt_array(TID+1))*a0)
        mu=mu+1
     do while (i <= mu)
     amu=amu+a(1,i)
     i=i+1
     end do
end do

!carry out reaction
call rxns(mu,M,P) !call rxns subroutine in 'reactions_Parallel' module

!save cell specific data to u_temp
if (mother_sim==1) then
```

```
u_temp_mothers(1) = t_sample
u_temp_mothers(2) = P
u_temp_mothers(3) = M
u_temp_mothers(4) = D
u_temp_mothers(5) = k
u_temp_mothers(6) = mu
end if

!save data to u_daughters
if (daughter_sim==1) then
u_temp_daughters(1) = t_sample
u_temp_daughters(2) = P
u_temp_daughters(3) = M
u_temp_daughters(4) = D
u_temp_daughters(5) = d_count
u_temp_daughters(6) = mu
end if

end do

END SUBROUTINE next_reaction
!*********************************

!*********************************
SUBROUTINE propensities(s,a)
use globals_Parallel
```

```
implicit none
real, dimension(1,3) :: s
real, dimension(1,4) :: a
real ::  P,M,D

!conversion into P,M,D from 's' array
P=s(1,1)
M=s(1,2)
D=s(1,3)

  !compute reaction propensities
a(1,1)= D*v0
a(1,2)= M*d0_prime
a(1,3)= M*v1_prime
a(1,4)= P*d1

end subroutine propensities
!*********************************

END MODULE SSA_Parallel
!*************************************************

!**MODULE reactions_Parallel**

!This module contains the reactions for the user defined gene regulatory network.
```

```
!*************************************************
MODULE reactions_Parallel
implicit none
contains

!*********************************
SUBROUTINE rxns(mu,M,P)
use globals_Parallel
implicit none
integer :: mu
real :: M,P

if (mu==1) then
M=M+1
end if
if (mu==2) then
M=M-1
end if
if (mu==3) then
P=P+1
end if
if (mu==4) then
P=P-1
end if

END SUBROUTINE rxns
```

```
!*********************************

END MODULE reactions_Parallel
!************************************************

!**Module globals_Parallel**

!This module contains the global parameters.

!************************************************
MODULE globals_Parallel

real, parameter :: sample_intervl=100 !population sampling interval
real, parameter :: t_end=216000 !simulation end time (default ‘100000’)
integer, parameter :: num_rand=100000 !number of random numbers
real, parameter :: log2=log10(2.), pi=3.14159265
!note: (default ‘1’ --> represents 1.0*10^-15 liters)
real, parameter :: init_cell_vol=2.5 !initial/following division cell volume
real, parameter :: t_div=3600 !cell division time (default ‘6000’) !
real, parameter :: t_restore=3300  !cell population restore time (default ‘5500’)
real, parameter :: D_init=1 !initial gene copy number on chromosome
integer, parameter :: n_threads=8 !number of threads

!parameters for <mRNA(n)> results
real, parameter :: v0=0.3,d0=0.1,d0_prime=0.0221,v1=0.048,v1_prime=0.16,
real, parameter :: d1=0.0000642,f0=0.42,b0=0.1,n=8,k0=0.1
```

```
real, parameter :: k1=0.3,mb1=0.4,mf0=0.114,mf1=4.0

!parameters for <protein(t)>
!real, parameter :: v0=0.3,d0=0.1,d0_prime=0.0221,v1_prime=0.075,v1=0.048,k1=0.3
!real, parameter :: d1=0.00000642,f0=0.42,b0=0.1,n=1,k0=0.1,mb1=0.4,mf0=0.114,mf1=

!parameters for single-cell
!real, parameter :: v0=0.3,d0=0.1,d0_prime=0.05,v1=0.048,v1_prime=0.16,
!real, parameter  :: d1=0.0000642,f0=0.42,b0=0.1,n=1,k0=0.1
!real, parameter :: k1=0.3,mb1=0.4,mf0=0.114,mf1=4.0

END MODULE globals_Parallel
!*************************************************

!**MODULE results_Parallel**

!This module obtains simulation results and statistics,
!and calculates analytical results from subroutines
!‘time_series’, ‘pop_stats’, and ‘stats_theoretical’.

!*************************************************
MODULE results_Parallel
implicit none
contains

!*********************************
```

```
SUBROUTINE time_series(u,u_daughters,nc,daughter_cnt,t_cumm)
use globals_Parallel
implicit none
real, dimension(nc,11) :: u
real, dimension(nc,11) :: u_daughters
real :: t_cumm
integer :: j,k,nc,daughter_cnt

!obtain time series for mother cells
write(1,*) 'Mother Cells'
do k=1,nc
  !obtain values for a single cell (comment for all cell values)
if (k==1) then
!obtain values for select cells
!if (k==etc.or.3.or.2.or.1) then
write(1,*) (u(k,1)),(u(k,2)),(u(k,3)),(u(k,4)),(u(k,5)),(u(k,6)),
(u(k,7)),(u(k,8)),(u(k,9)),(u(k,10)),(u(k,11))
end if
end do

!obtain time series for daughter cells
  if (daughter_cnt >= 1) then
  write(1,*) 'Daughter Cells'
do j=1,daughter_cnt
!if (j==1) then
!if (j==etc.or.3.or.2.or.1) then
```

```
write(1,*) (u_daughters(j,1)),(u_daughters(j,2)),(u_daughters(j,3)),
(u_daughters(j,4)),(u_daughters(j,5)),(u_daughters(j,6)),
(u_daughters(j,7)),(u_daughters(j,8)), (u_daughters(j,9)),
(u_daughters(j,10)), (u_daughters(j,11))
!end if
end do
end if

END SUBROUTINE time_series
!*********************************

!*********************************
SUBROUTINE pop_stats(u,u_daughters,nc,daughter_cnt,t_cumm)
use globals_Parallel
implicit none
real, dimension(nc,11) :: u
real, dimension(nc,11) :: u_daughters
real :: avg_mRNA,tot_mRNA,tot_mRNA_mothers,tot_mRNA_daughters,avg_protein,
real :: tot_protein,tot_protein_mothers,tot_protein_daughters,var_cnt_mRNA_mothers
real :: var_cnt_mRNA_daughters,var_mRNA_population,noise_mRNA_population,t_cumm
real :: var_cnt_protein_mothers, var_cnt_protein_daughters, var_protein_population
real :: noise_protein_population
integer :: nc,i,j,k,l,daughter_cnt

!calculate mean population mRNA (includes mother and daughter cells)
tot_mRNA=0;tot_mRNA_mothers=0;tot_mRNA_daughters=0;avg_mRNA=0
```

```
do i=1,nc
tot_mRNA_mothers=tot_mRNA_mothers+u(i,3)
end do
!if (daughter_cnt >= 1) then
! do k=1,daughter_cnt
! tot_mRNA_daughters=tot_mRNA_daughters+u_daughters(k,3)
! end do
!end if
tot_mRNA=tot_mRNA_mothers+tot_mRNA_daughters
avg_mRNA=tot_mRNA/(nc+daughter_cnt)

!calculate variance population mRNA (includes mother and daughter cells)
var_cnt_mRNA_mothers=0; var_cnt_mRNA_daughters=0; var_mRNA_population=0
do i=1,nc
var_cnt_mRNA_mothers=var_cnt_mRNA_mothers+((u(i,3)-avg_mRNA)**2)
end do
!if (daughter_cnt >= 1) then
! do k=1,daughter_cnt
! var_cnt_mRNA_daughters=var_cnt_mRNA_daughters+((u_daughters(k,3)
!   -avg_mRNA)**2)
! end do
!end if
var_mRNA_population=(var_cnt_mRNA_mothers+var_cnt_mRNA_daughters)
!/(nc+daughter_cnt-1)

!calculate noise population mRNA (includes mother and daughter cells)
```

```
noise_mRNA_population=0
noise_mRNA_population=sqrt(var_mRNA_population/(avg_mRNA**2))

!calulate mean population protein (includes mother and daughter cells)
tot_protein=0;tot_protein_mothers=0;tot_protein_daughters=0;avg_protein=0
do j=1,nc
tot_protein_mothers=tot_protein_mothers+u(j,2)
end do
!if (daughter_cnt >= 1) then
! do l=1,daughter_cnt
! tot_protein_daughters=tot_protein_daughters+u_daughters(l,2)
! end do
!end if
tot_protein=tot_protein_mothers+tot_protein_daughters
avg_protein=tot_protein/(nc+daughter_cnt)

!calculate variance population protein (includes mother and daughter cells)
var_cnt_protein_mothers=0; var_cnt_protein_daughters=0; var_protein_population=0
do i=1,nc
var_cnt_protein_mothers=var_cnt_protein_mothers+((u(i,2)-avg_protein)**2)
end do
!if (daughter_cnt >= 1) then
! do k=1,daughter_cnt
! var_cnt_protein_daughters=var_cnt_protein_daughters+((u_daughters(k,2)
!   -avg_protein)**2)
! end do
```

```
!end if
var_protein_population=(var_cnt_protein_mothers+var_cnt_protein_daughters)
/(nc+daughter_cnt-1)

!calculate noise population protein (includes mother and daughter cells)
noise_protein_population=0
noise_protein_population=sqrt(var_protein_population/(avg_protein**2))

!write statistics to file
write(3,*) 't_sample = ', t_cumm
write(3,*) 'avg mRNA sim = ', avg_mRNA, 'noise mRNA sim = ',
noise_mRNA_population, 'avg protein sim = ', avg_protein,
'noise protein sim = ', noise_protein_population

END SUBROUTINE pop_stats
!*********************************

!*********************************
SUBROUTINE stats_theoretical()
use globals_Parallel
implicit none
real, dimension(1,t_div) :: avg_protein_array, protein_noise_array
real :: avg_mRNA_theor,mRNA_noise_theor,avg_mRNA_theor_dupl,mRNA_noise_theor_dupl
real :: l,l0,l1,d0_prime_theor,v1_prime_theor,b
integer :: i,j,n_duplicated
```

```
!Analytical values via Swain et al. formulas

!Compute mean mRNA and mRNA noise - *before gene duplication*
l=f0+b0+k0
l1=k1+mb1+mf0+mf1
l0=sqrt((l1**2)-(4*mf0*(k1+mb1)))
d0_prime_theor=log10(2.)*((l1-l0)/2)
!print*, 'd0_prime_theor = ', d0_prime_theor
avg_mRNA_theor=(f0*k0*n)/(d0_prime_theor*l)

!Compute mean number of proteins produced per transcript
b=(1/mf0)*((k1*mf1)/(k1+mb1))

v1_prime_theor=b*d0_prime_theor
!print*, 'v1_prime_theor = ', v1_prime_theor

!Compute mRNA noise
!mRNA noise squared
mRNA_noise_theor=(1/avg_mRNA_theor)-((d0_prime_theor*v0*
(d0_prime_theor+l+v0))/(n*(d0_prime_theor+l)*(l+v0)*(d0_prime_theor+v0)))

!write(2,*) 'Before gene duplication'
!write(2,*) 'avg mRNA theor = ', avg_mRNA_theor,
'avg protein per transcript = ', b, 'mRNA noise theor = ',
 sqrt(mRNA_noise_theor)
```

```
!Compute mean mRNA and mRNA noise - *after gene duplication*
n_duplicated=2*n
avg_mRNA_theor_dupl=(f0*k0*n_duplicated)/(d0_prime_theor*l)

!Compute mRNA noise
mRNA_noise_theor_dupl=(1/avg_mRNA_theor_dupl)-((d0_prime_theor*v0
*(d0_prime_theor+l+v0))/(n_duplicated*(d0_prime_theor+l)*(l+v0)
*(d0_prime_theor+v0))) !mRNA noise squared

!write(2,*) 'After gene duplication'
!write(2,*) 'avg mRNA theor = ', avg_mRNA_theor_dupl,
'avg protein per transcript = ', b, 'mRNA noise theor = ',
sqrt(mRNA_noise_theor_dupl)

!Compute mean protein number and noise (note <mRNA> before gene duplication
is used for (0,t_dvi) as specified in Swain et al.)
j=0
do i=0,t_div,100
!before gene duplication
if (i <= 0.4*t_div) then
j=j+1
avg_protein_array(1,j)=(v1_prime_theor/d1)*(avg_mRNA_theor)
*(1-((exp(-d1*(t_div-(0.4*t_div)+i)))/(2-(exp(-d1*t_div)))))
protein_noise_array(1,j)=(1/avg_protein_array(1,j))
+(1/avg_mRNA_theor)*(1-(f0*k0/(l**2)))*(d1/d0_prime_theor)
*((2-exp(-d1*t_div))/(2+exp(-d1*t_div)))*((4-exp(-2*d1*t_div)
```

```
-2*exp(-2*d1*i)-exp(-2*d1*(t_div+i-(0.4*t_div))))/((2-exp(-d1*t_div)
-exp(-d1*(t_div+i-(0.4*t_div))))**2)) !protein noise squared
!after gene duplication
elseif (i > 0.4*t_div) then
j=j+1
avg_protein_array(1,j)=(v1_prime_theor/d1)*(avg_mRNA_theor)
*(2*(1-((exp(-d1*(i-(0.4*t_div))))/(2-(exp(-d1*t_div))))))
protein_noise_array(1,j)=(1/avg_protein_array(1,j))+(1/avg_mRNA_theor)
*(1-(f0*k0/(l**2)))*(d1/d0_prime_theor)*((2-exp(-d1*t_div))
/(2+exp(-d1*t_div)))*((4-exp(-2*d1*t_div)-exp(-2*d1*i)
-2*exp(-2*d1*(i-(0.4*t_div))))/(2*(2-exp(-d1*t_div)
-exp(-d1*(i-(0.4*t_div))))**2)) !protein noise squared
end if
write(2,*) i, avg_protein_array(1,j), sqrt(protein_noise_array(1,j))
end do

END SUBROUTINE stats_theoretical
!*********************************

END MODULE results_Parallel
!************************************************
```