university of groningen / faculty of science and engineering

Propagation of initial condition uncertainty for linear chemical reaction systems via the Liouville Equation

Bachelor's Project Chemistry
July 2020
Student: M.A. van Straaten
First supervisor: Dr. A. Milias-Argeitis
Second assessor: Prof. dr. M. Heinemann

1 Introduction

Dynamical systems are systems whose behaviour changes over time. (Bio)chemical reactions lead to changes in the concentration of reactants and products over time, hence they can be treated as dynamical systems. To better understand a given (bio)chemical reaction system, it is of interest to know more about dynamical systems and their behaviour. As a result of imperfect knowledge (e.g. initial conditions) and imperfections in the model formulation itself, mathematical models of dynamical systems are typically subject to uncertainty. To enable modelling of reactions over time, these uncertainties in dynamical systems need to be considered.

Systems of (bio)chemical reactions are often modelled via sets of ordinary differential equations (ODEs). A key component of an ODE-based model is the initial condition, which results in a specific solution for the system. For most (bio)chemical reactions, precisely measuring the initial condition is difficult. As a result, the initial condition of the system is often not precisely known. In order to still model (bio)chemical reactions, it is of interest to predict how uncertain initial conditions for dynamical systems evolve over time.

Probability density functions can help to represent initial condition uncertainty in dynamical systems. To see how these initial probability distributions evolve over time, this thesis derives a method to follow the evolution of so-called initial log-polynomial distributions of linear first-order dynamical systems.

First, the preliminaries necessary to derive the expression for the log-polynomial distribution are covered. Afterwards, the log-polynomial distributions are derived and inserted into the Liouville Equation to obtain the ordinary differential equations describing the change of the log-polynomial distribution parameters over time. Next, the Matlab code is explained and lastly the multivariate model is subjected to an unregulated gene expression system.

2 Preliminaries

In order to understand the concepts discussed in this thesis, some preliminary sections will follow. In section 2.1 ordinary differential equations will be discussed. Probability distributions and their density will be treated in section 2.2 and 2.3, and exponential families in section 2.4.

2.1 Ordinary Differential Equations

Dynamical systems are frequently described as a set of ordinary differential equations. The solution of these differential equations provides the evolution of the state of the system over time. Differential equations are widely used in chemical and biological models, since in many cases it is easier to write the rate of change of a system variable, than to obtain a function describing the variable over time [1]. The general form of a system of ordinary differential equations is given by

$$\dot{x} = f(x) \tag{1}$$

With initial condition

$$x(0) = x_0$$

Here, $x \in \mathbb{R}^n$ is a vector describing the state of the dynamical system. $\dot{x}$ represents the fist-order derivative of the vector with respect to t. The vector $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ consists of functions corresponding to the entries of the vector x. The initial state of the system at time $t = 0$ is $x(0)$ and is defined by initial condition $x_0 \in \mathbb{R}^n$, containing the initial values of all entries of x. A multivariate system of ordinary differenial equations can be seen in equation (2).

$$\dot{x} = A \cdot x \tag{2}$$

$A \in \mathbb{R}^{n \times n}$ is a matrix containing the coefficients multiplying $x \in \mathbb{R}^n$.

In this thesis, systems of linear ordinary differential equations are considered. Even though most chemical reaction systems are not described by a linear system, most nonlinear systems can be locally approximated by a linear dynamical system.

The initial condition x_0 determines the specific solution of a dynamic system. If the initial condition is uncertain, the time evolution of the state vector will also be uncertain, resulting in a system that cannot be solved exactly. To see how a not fully determined initial condition affects a system, we need to find a way to express this initial state uncertainty. This can be accomplished by using probability distributions.

2.2 Probability Distributions

The probability distribution of a continuous random variable X indicates how likely it is for X to obtain a value within a given set. For (bio)chemical dynamic models, a guess can often be made about the probability distribution of the initial condition. Therefore, probability distributions give a way to describe initial uncertainty in a mathematical way.

The probability distribution of a continuous random variable X is described by its Probability Density Function (PDF), denoted by $p(x)$. The area under the curve between two points a and b represents the probability of X taking a value between a and b. Hence, in order to get the probability of X taking a value between a and b, we calculate

$$P(a \leq X \leq b) = \int_a^b p(x)dx \tag{3}$$

If X is multivariate ($X \in \mathbb{R}^n, n > 1$), the PDF is a function of n variables $x_1, x_2, ..., x_n$. For a multivariate X we can calculate the probability of X taking a value inside the box

$[a_1, b_1] \times [a_2, b_2] \times ... \times [a_n, b_n]$ by calculating a multiple integral:

$$P(X \in [a_1, b_1] \times [a_2, b_2] \times ... \times [a_n, b_n]) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} ... \int_{a_n}^{b_n} p(x)\ dx_1 dx_2 ... dx_n \tag{4}$$

Another important aspect of probability density functions is that the total probability is defined to be 1 (equation (5) for a scalar X, equation (6) for a multivariate X).

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{5}$$

$$\int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} p(x)\ dx_1 dx_2 ... dx_n = 1 \tag{6}$$

In this thesis, probability distributions are used to represent the uncertain initial condition of a system. After $t = 0$, the distributions evolve to represent the state of to the system. The key question is how we can calculate the change of the probability density of the system state x over time.

2.3 Normal Distributions

In order to simulate the evolution of a system subject to a certain initial distribution, a specific initial distribution needs to be considered. Normal distributions are typically used to describe the initial condition uncertainty, since the normal distribution has many mathematical properties that make calculations straightforward. The two parameters which distinguish scalar normal distributions from each other are the mean $\mu \in \mathbb{R}$, and the variance $\sigma^2 \in \mathbb{R}$. For a random variable X that is normally distributed with mean μ, and variance σ^2, the common notation is $X \sim \mathcal{N}(\mu, \sigma^2)$. An example of a normal distribution where X is distributed as $X \sim \mathcal{N}(30, 25)$ can be found in figure 1.

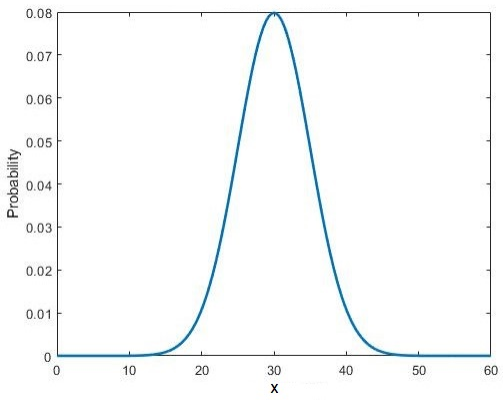


Figure 1: Probability density function with mean 30 and variance 25

The probability density function describing the normal distribution $X \sim \mathcal{N}(\mu, \sigma^2)$ is provided by equation (7). For $-\infty < x < \infty$

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x-\mu)^2}{2\sigma^2} \tag{7}$$

As with systems of ordinary linear differential equations (section 2.1) normal distributions can also be written in the multivariate form. Instead of a scalar random variable, a random vector is used. The random vector $X \in \mathbb{R}^n$ is a column vector, whose entries are scalar random variables.

The two variables which describe multivariate normal distributions are the mean vector $\mu \in \mathbb{R}^n$ and the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. The covariance matrix Σ is a square matrix consisting of the pairwise covariance between all entries of X. The pairwise covariance characterizes the behaviour of pairs of entries of X. The notation for the multivariate normal distribution is $X \sim \mathcal{N}(\mu, \Sigma)$.

The probability density function for a multivariate normal distribution is given by

$$f(x) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{8}$$

For two dimensional multivariate normal distributions, the graph looks similar to figure 2.

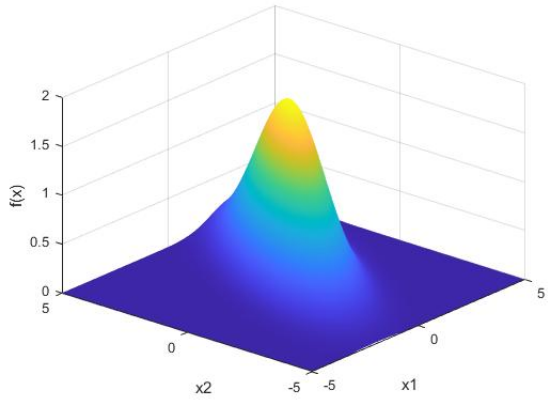


Figure 2: Example of a graph for a two-dimensional multivariate normal distribution

As we will explain below, when the normal distribution is used to represent the initial uncertainty of a linear dynamical system, then the probability density of the system state at any time can be calculated in a straightforward way. However, the normal assumption is quite restrictive for many linear dynamical systems, so we would like to consider a larger class of initial distributions. These are presented in the next section.

2.4 Exponential Families

In order to obtain mathematically tractable solutions to uncertainty propagation problems for linear systems, we will consider a special class of exponential families.

For a scalar random variable X, an exponential family is a set of parametrized probability distributions whose probability density function can be written in the form

$$f_X(x \mid \lambda) = g(\lambda) \exp(\eta(\lambda) \cdot T(x)) \tag{9}$$

Here, $\lambda \in \mathbb{R}^m$ is the parameter vector of the exponential family. Parameters λ can be set to different values, producing different distributions belonging to the same family. $T(x) : \mathbb{R} \to \mathbb{R}^p$ is a vector of sufficient statistics, meaning that it contains all necessary information about x with respect to the parameter values. $g(\lambda)$ is called the partition function. It is appropriately defined to ensure that the integral of the PDF is equal to 1. Finally, $\eta(\lambda) : \mathbb{R}^m \to \mathbb{R}^p$ contains the so-called natural parameters of the distribution, which can be functions of the original parameters λ.

To see that the normal distribution is an exponential family, let us define $\lambda = \begin{bmatrix} \mu & \sigma^2 \end{bmatrix}$. If we further define

$$g(\lambda) = \frac{1}{\sqrt{2\pi}\sigma} \exp(\frac{-\mu^2}{2\sigma^2}); \tag{10}$$

$$\eta(\lambda) = \begin{bmatrix} -\frac{1}{2\sigma^2} & \frac{\mu}{\sigma^2} \end{bmatrix}; \qquad T(x) = \begin{bmatrix} x^2 \\ x \end{bmatrix} \tag{11}$$

Then we find

$$\begin{aligned} f(x;\mu,\sigma^2) &= g(\lambda)\exp(\eta(\lambda)\cdot T(x)) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-\mu^2}{2\sigma^2} - \frac{x^2}{2\sigma^2} + \frac{x\mu}{\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \end{aligned}$$

Which is the same probability density function as equation (7). Therefore, all normal distributions belong to the same exponential family.

Defining this, we can also relate non-normal distributions to an exponential family. To see how this is possible, we can rewrite the normal distribution as follows: defining $\eta(\lambda) = \begin{bmatrix} \lambda_2 & \lambda_1 \end{bmatrix}$, and x such that $T(x) = \begin{bmatrix} x^2 \\ x \end{bmatrix}$ and deriving $g(\lambda)$ such that the total probability is 1, we find

$$f(x) = g(\lambda)\exp(\lambda_2 x^2 + \lambda_1 x)$$

Since $g(\lambda)$ can be treated as a lambda parameter, we write it as $\exp(\lambda_0)$, Then we can write the distribution in the form of (12), by putting all the lambda parameters in a single exponential.

$$f(x) = \exp(\lambda_2 x^2 + \lambda_1 x + \lambda_0) \tag{12}$$

Equation (12) is still a normal distribution, although with different notation. This notation will be used to define non-normal distributions later.

Henceforth, if the logarithm of a probability density function results in a polynomial, the PDF will be said to follow a log-polynomial distribution. We constrict the sufficient statistics $T(x)$ to only containing the monomials of x, ensuring a log-polynomial density.

The definition of an exponential family can be generalized by using the random vector $X \in \mathbb{R}^n$ instead of the scalar random variable X. This also changes the definition of the vector of sufficient statistics into a vector function where $T(x) : \mathbb{R}^n \to \mathbb{R}^p$. Then, the multivariate normal distribution is also a member of an exponential family. For the two dimensional multivariate distribution, using $\Sigma = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$, $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ we find that equation (8) becomes

$$\begin{aligned} p(X) = \frac{1}{\sqrt{2\pi|\Sigma|}} \cdot \exp(-\frac{1}{2}\frac{1}{s_{11}s_{22} - s_{12}s_{21}} \cdot s_{22}x_1^2 + s_{11}x_2^2 + (-2\mu_1 s_{22} + \mu_2 s_{12} + \mu_2 s_{21})x_1 \\ +(\mu_1 s_{12} - 2\mu_2 s_{11} + \mu_1 s_{21})x_2 + (-s_{12} - s_{21})x_1 x_2 + \mu_1^2 s_{22} - \mu_1\mu_2 s_{12} + \mu_2^2 s_{11} - \mu_1\mu_2 s_{21}) \end{aligned}$$

Thus, the multivariate normal exponential family is defined by

$$g(\lambda) = \frac{1}{\sqrt{2\pi|\Sigma|}}\exp(\mu_1^2 s_{22} - s_{12}\mu_1\mu_2 + s_{11}\mu_2^2 - s_{21}\mu_1\mu_2);$$

$$\eta(\lambda) = \begin{bmatrix} s_{22} \\ s_{11} \\ -2\mu_1 s_{22} + \mu_2 s_{12} + \mu_2 s_{21} \\ \mu_1 s_{12} - 2\mu_2 s_{11} + \mu_1 s_{21} \\ -s_{12} - s_{21} \end{bmatrix}; \qquad T(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 \\ x_2 \\ x_1 x_2 \end{bmatrix}$$

Just like the univariate case, the parameter λ is defined by the parameters of the multivariate distribution, $\lambda = \begin{bmatrix} \mu & \Sigma \end{bmatrix}$. Also the two dimensional multivariate normal distribution can be written as a log-polynomial density function:

$$p(x) = \exp(\lambda_1 x_1^2 + \lambda_2 x_2^2 + \lambda_{11} x_1 + \lambda_{22} x_2 + \lambda_{12} x_1 x_2 + \lambda_0) \tag{13}$$

Naturally, this can also be done for higher order multivariate distributions. See for instance equation (14) for the general exponential family for the fourth order multivariate distribution.

$$p(x) = \exp(\lambda_{4,0}x_1^4 + \lambda_{3,0}x_1^3 + \lambda_{2,0}x_1^2 + \lambda_{1,0}x_1 + \lambda_{0,4}x_2^4 + \lambda_{0,3}x_2^3 + \lambda_{0,2}x_2^2 \tag{14}$$

$$+\lambda_{0,1}x_2 + \lambda_{3,1}x_1^3x_2 + \lambda_{2,1}x_1^2x_2 + \lambda_{1,3}x_1x_2^3 + \lambda_{1,2}x_1x_2^2 + \lambda_{2,2}x_1^2x_2^2 + \lambda_{1,1}x_1x_2 + \lambda_0)$$

One thing to keep in mind when constructing higher order exponential families is that the order of the polynomial in the exponential term must always be an even number. In order for a PDF to be valid, the probability density cannot go to infinity, considering that the PDF has to integrate to 1 (equations (5) and (6)). The asymptotic behavior of polynomials is determined by the highest order terms. In equations (15) and (16), the limit of different univariate higher order terms are considered. When a limit goes to infinity, that polynomial order cannot define a PDF for the aforementioned reasons. When n is even:

$$\lim_{x\to\pm\infty} \exp(x^n) = \infty \qquad \lim_{x\to\pm\infty} \exp(-x^n) = 0 \tag{15}$$

When n is odd:

$$\lim_{x\to\infty} \exp(x^n) = \infty \qquad \lim_{x\to-\infty} \exp(-x^n) = \infty \tag{16}$$

Hence, from equations (15) and (16) it can be concluded that the order of the polynomial in the exponent can only be even, in addition to the fact that the coefficient of the highest order term has to be negative. Moreover, it must be kept in mind that from e.g. equation (13) the x_1x_2 term is also a second order term, and must therefore have a negative coefficient as well. The same goes for the $x_1^nx_2^m$ terms in equation (14) for which $n + m = 4$.

Using these exponential families, we have a way of describing various variations of initial probabilities. Next, we can start looking at how we can model the change of the initial distribution over time.

3 Describing the evolution of initial distributions over time

To recap, we have found a general expression of the initial distribution for a dynamical system (equations (12), (13) and (14)). The question remains how we can model the change in the initial distributions over time, such that we can find the probability density function of the system at any given time.

3.1 Normal probability distribution over time

To illustrate how probability distributions in a dynamical system change over time, we go back to the univariate normal distribution (section 2.3). Considering a scalar linear system $\dot{x} = ax$, with initial uncertainty $x(0) = x_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$, the distribution of the system at $t > 0$ will always be a normal distribution. This is shown as follows:

$$\begin{aligned}
\dot{x} = \frac{dx}{dt} &= ax \\
\int \frac{1}{ax} dx &= \int dt \\
\frac{1}{a}\log(ax) &= t + C \\
\log(ax) &= a(t + C) \\
ax &= \exp(a(t + C) \\
x(t) &= \frac{1}{a}\exp(a(t + C)
\end{aligned}$$

For $C' = \frac{1}{a}\exp(aC)$, we find

$$x(t) = C'\exp(at)$$

Since $x(0) = x_0$,

$$\begin{aligned}
x(0) = C'\exp(0) &= x_0 \\
C' &= x_0 \\
x(t) &= x_0\exp(at)
\end{aligned}$$

At any time t, the term $\exp(at)$ is a constant. Hence, the probability density at time $t > 0$ is just a linear transformation of a normally distributed random variable. This means that $x(t)$ is also normally distributed, more specifically $x(t) \sim \mathcal{N}(e^{at}\mu_0, e^{2at}\sigma_0^2)$ [2]. So, if the initial probability distribution is univariate normal, then the probability at $t > 0$ can be written as a normal distribution too.

For a multivariate system $\dot{X} = AX$, with initial uncertainty $X(0) = X_0 \sim \mathcal{N}(\mu, \Sigma)$, the specific solution is

$$X(t) = \exp(At) \cdot X_0$$

$\exp(At) \in \mathbb{R}^{n \times n}$ is a matrix depending on t. Therefore, for any time $t > 0$ the matrix will consists of constants. The components of $X(t)$ for any specific time will therefore be a linear transformation of a multivariate normally distributed random variable, which results again in a multivariate normal distribution for $X(t)$, with $X(t) \sim \mathcal{N}(e^{At}\mu, e^{At}\Sigma e^{At^T})$ [2]. Hence, if the initial uncertainty for a multivariate linear dynamical system is multivariate normal, the distribution of the system at time $t > 0$ is also multivariate normal.

With this knowledge, we modify the exponential families for the univariate and multivariate normal distributions, making the lambda-parameters dependent on time. The change of the time-dependent lambda-parameters represents the change of the normal distribution

over time. This results in equation (17) for a scalar system and equation (18) for a two dimensional vector system.

$$p(t,x) = \exp(\lambda_2(t)x^2 + \lambda_1(t)x + \lambda_0(t)) \tag{17}$$

$$p(t,x) = \exp(\lambda_1(t)x_1^2 + \lambda_2(t)x_2^2 + \lambda_{11}(t)x_1 + \lambda_{22}(t)x_2 + \lambda_{12}(t)x_1x_2 + \lambda_0(t)) \tag{18}$$

Not a lot of initial probability distributions for dynamical systems follow a normal distribution. Therefore, the question arises whether other types of initial distributions for linear dynamical systems will also keep describing the system after $t = 0$.

3.2 Log-polynomial distributions over time

In order to describe other distributions whose form is maintained over time, we generalize the log-polynomial distribution of the normal distributions from section 3.1. Doing this, we look at a bigger exponential family of which the normal distribution is also a part of:

$$p(x) = \exp(\lambda_n(t)x^n + \lambda_{n-1}(t)x^{n-1} + ... + \lambda_1(t)x + \lambda_0(t)) \tag{19}$$

Note that if we put $\lambda_n(t), \lambda_{n-1}(t), ..., \lambda_3(t)$ equal to zero for all t, we end up with the univariate normal distribution (equation (17)). The same we do for the multivariate case, e.g. for the fourth order:

$$\begin{aligned} p(x) = \exp(&\lambda_{4,0}(t)x_1^4 + \lambda_{3,0}(t)x_1^3 + \lambda_{2,0}(t)x_1^2 + \lambda_{1,0}(t)x_1 + \lambda_{0,4}(t)x_2^4 \\ &+\lambda_{0,3}(t)x_2^3 + \lambda_{0,2}(t)x_2^2 + \lambda_{0,1}(t)x_2 + \lambda_{3,1}(t)x_1^3x_2 + \lambda_{2,1}(t)x_1^2x_2 \\ &+\lambda_{1,3}(t)x_1x_2^3 + \lambda_{1,2}(t)x_1x_2^2 + \lambda_{2,2}(t)x_1^2x_2^2 + \lambda_{1,1}(t)x_1x_2 + \lambda_0(t)) \end{aligned} \tag{20}$$

This can also be done for higher order multivariate distributions, however the log-polynomial becomes increasingly longer for higher orders, hence only the fourth order is shown. As with the normal case, equations (19) and (20) describe the evolution of the probability distribution of the system thanks to the time-dependency of the lambda parameters.

3.3 Liouville Equation

Having found general descriptions for the initial uncertainty of a linear system (eq. (19) and (20)), the lambda parameters are still unknown after $t = 0$. It would be ideal if it was possible to find expressions for the rate of change of the lambdas over time. Having a system of ordinary differential equations describe the lambdas would solve this problem. Then it would be possible to find the probability density of the system at any point in time.

Fortunately, the Liouville Equation [3] can be used to describe the time evolution of probability density functions. In equation (21) the general definition of the Liouville equation is given.

$$\frac{\partial p(t,x)}{\partial t} + \sum_{k=1}^{n} f_k(x)\frac{\partial p(t,x)}{\partial X_k} + \sum_{k=1}^{n} \frac{\partial f_k(x)}{\partial X_k} p(t,x) = 0 \tag{21}$$

n is the dimension of the state vector x. $p(t,x)$ is the probability density function of the system, depending on x and t, where the time dependence is expressed in the lambdas. $f(x) \in \mathbb{R}^n$ is the vector containing the functions which represent the derivative (equation (1)).

The Liouville Equation allows us to derive ordinary differential equations describing the evolution of the lambdas over time. When inserting the time-varying distribution (17) to the Liouville Equation, for an arbitrary scalar system $\dot{x} = ax, a \in \mathbb{R}$, we find:

$$p(t,x) \cdot (\dot{\lambda_2}x^2 + \dot{\lambda_1}x + \dot{\lambda_0}) + ax \cdot p(t,x) \cdot (2\lambda_2 x + \lambda_1) + a \cdot p(t,x) = 0$$

Since the probability density function $p(t,x)$ cannot be zero, we find

$$\dot{\lambda_2}x^2 + \dot{\lambda_1}x + \dot{\lambda_0} + ax \cdot (2\lambda_2 x + \lambda_1) + a = 0$$

$$(\dot{\lambda_2} + 2a\lambda_2)x^2 + (\dot{\lambda_1} + a\lambda_1)x + \dot{\lambda_0} + a = 0$$

Additionally, this equation needs to hold for any value x within the state space of the system. This implies that the coefficient of each power of x needs to be equal to zero, giving rise to a set of ordinary differential equations describing the time evolution of the lambdas:

$$\dot{\lambda_2}(t) = -2a\lambda_2(t) \tag{22}$$

$$\dot{\lambda_1}(t) = -a\lambda_1(t) \tag{23}$$

$$\dot{\lambda_0}(t) = -a \tag{24}$$

For multivariate distributions a similar approach can be made. For the two dimensional system $\dot{x} = A \cdot x$, where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, we find as a result from the Liouville Equation:

$$p(t,x) \cdot (\dot{\lambda_1}x_1^2 + \dot{\lambda_2}x_2^2 + \dot{\lambda_{11}}x_1 + \dot{\lambda_{22}}x_2 + \dot{\lambda_{12}}x_1x_2 + \dot{\lambda_0})$$

$$+(a_{11}x_1 + a_{12}x_2)p(t,x)(2\lambda_1 x_1 + \lambda_{11} + \lambda_{12}x_2)$$

$$+(a_{21}x_1 + a_{22}x_2)p(t,x)(2\lambda_2 x_2 + \lambda_{22} + \lambda_{12}x_1) + (a_{11} + a_{22})p(t,x) = 0$$

Again using the properties that $p(t,x)$ and x cannot be zero, we find

$$\begin{aligned}
\dot{\lambda_1} &= -2a_{11}\lambda_1 - a_{21}\lambda_{12} \\
\dot{\lambda_2} &= -2a_{22}\lambda_2 - a_{12}\lambda_{12} \\
\dot{\lambda_{11}} &= -a_{11}\lambda_{11} - a_{21}\lambda_{22} \\
\dot{\lambda_{22}} &= -a_{22}\lambda_{22} - a_{12}\lambda_{11} \\
\dot{\lambda_{12}} &= -a_{11}\lambda_{12} - 2a_{12}\lambda_1 - 2a_{21}\lambda_2 - a_{22}\lambda_{12} \\
\dot{\lambda_0} &= -a_{11} - a_{22}
\end{aligned}$$

These ordinary differential equations can be derived for any log-polynomial initial probability density. With the systems of ordinary differential equations and the initial condition of the lambdas, we can find the specific solution of the dynamical system at any point in time. Therefore, the next goal is to simulate the change in distribution over time.

4 Numerical implementation

4.1 Overview

Matlab was used to solve the systems of ordinary differential equations describing the lambda parameters. Both for the scalar and two dimensional systems a script was written to solve for the lambdas numerically, using the Matlab function *ode45*. Moreover, a script was written which would automatically write the equations for the lambdas for higher dimensions.

4.2 Solving for the lambdas

For both the scalar and vector script, the initial values and the scalar a or matrix A have to be specified. The ending time of interest, alongside the number of time steps taken can be adjusted to the interest of the user. The x values for which the lambdas would be calculated each time can also be changed, to give a more accurate result. Using *ode45*, the values of the lambdas at each specified time was calculated. Afterwards, each lambda value was connected to the corresponding x monomial, resulting in the probability density function for that particular time. Finally, the probability density functions were plotted in sequence, such that the evolution of the probability density could be seen.

For these systems the ordinary differential equations describing the lambdas still had to be derived by hand, hence it was investigated whether a script could execute this time consuming step.

4.3 Writing the differential equations

In order to have Matlab compose the differential equations for the lambdas, a script was made for a two dimensional system, with a maximum degree of 4 in the polynomial. For higher dimensions and degrees the script can easily be adjusted to obtain the differential equations for the lambdas for those log-polynomials.

In order to write the differential equations describing the lambdas, first the monomials of the specified dimension were generated. Next, the log-polynomial is put together. The Liouville Equation is used to obtain the system of ordinary differential equations. These are solved using the *ode45* function, resulting in the values for the lambdas at the specified points in time. Using this code, the step of manually deriving the equations for the lambdas can be skipped.

5 A numerical example: unregulated gene expression

To see whether the written scripts can be used in further research, their application to dynamical systems should be analyzed. Therefore, the multivariate script was applied to an unregulated gene expression dynamical system [4]. The mRNA-protein system is schematically shown in figure 3.

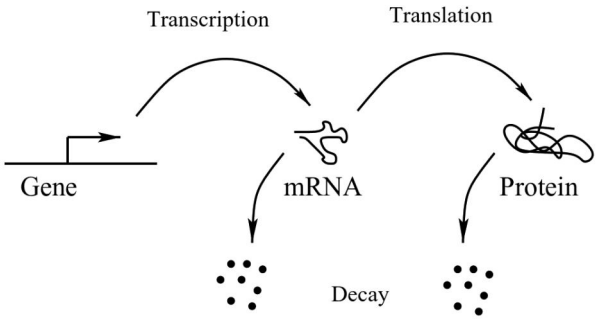


Figure 3: Gene expression system, where transcription of the gene gives mRNA, which after translation results in protein

The system is described by the following differential equations:

$$\dot{m} = k_0 - \delta_m m \tag{25}$$

$$\dot{p} = k_1 m - \delta_p p \tag{26}$$

Where m and p represent the concentration of mRNA and protein, respectively. The transcription rate is given by k_0, and k_1 accounts for the rate of translation. Also the decay of mRNA and protein is taken into account, with the respective terms δ_m and δ_p.

The model can be used to describe prokaryotic gene expression, since for eukaryotic gene expression additional steps need to be considered, such as the transport of the mRNA out of the nucleus. The model assumes all the other processes regarding gene expression to be constant, hence the model is highly simplified.

For simplicity, the terms not describing the mRNA or protein concentration were assumed to be constant. k_0 and k_1 were assumed to be 0.1, δ_m was set to be 0.1 and δ_p was 0.01.

The initial distribution for the start of the system was taken to be

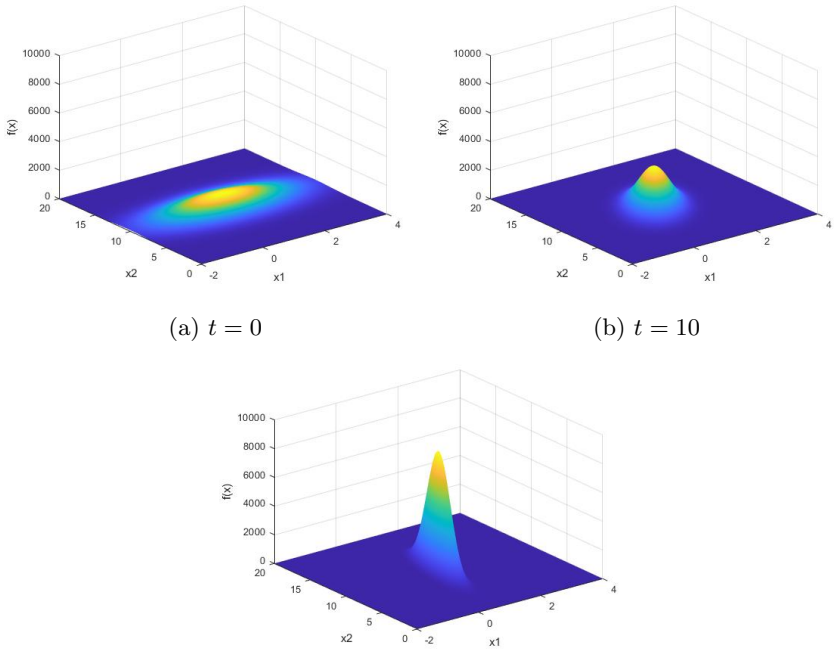
$$\log(p(x)) = -0.25x_1^2 - 0.125x_2^2 + 0.5x_1 + 2.5x_2 + \log\left(\frac{1}{\sqrt{2\pi}}\right) - 5$$

6 Results & Discussion

6.1 Unregulated Gene Expression

The initial state of the unregulated gene expression model is seen in figure 4a. The distribution over time (figures 4b and 4c) shows the distribution slowly becoming narrower over time. Simulteneously, the graph becomes higher, indicating that the distribution will still integrate to 1 and thus adhering to a fundamental property of probability distributions.

Considering that we have constant parameters describing the change in mRNA and protein concentration, it would be expected that at some point the mRNA and protein concentration reach equilibrium. Due to the narrowing of the distribution, it seems that the system attains an equilibrium state where the concentrations of mRNA and protein do not change. Hence, the generated results of the unregulated gene expression model make intuitively sense.



(a) $t = 0$

(b) $t = 10$

(c) $t = 20$

Figure 4: Evolution over time of the initial distribution for an unregulated gene expression model

6.2 Scalar Dynamical system

As for the multivariate model, the scalar model is also expected to reach an equilibrium, as no input uncertainty for other variables is considered. To see how an initial distribution evolves in our model, we looked at the scalar system $\dot{x} = -2x$, with bimodal initial distribution $-0.5x^4 - 0.1x^3 + 2x^2 - 2.742717515$, visible in figure 5.

When looking at figure 5, it is visible that over time the distribution narrows. The two peaks grow higher, while the distance between them narrows. Hence, the model indicates an equilibrium being approached. Therefore, the model seems to work as expected for a scalar dynamical system.

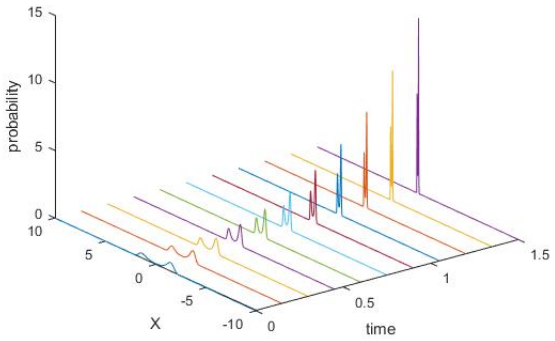


Figure 5: Time evolution of initial probability distribution for a scalar system

6.3 Writing Ordinary Differential Equations

For scalar and multivariate systems, the ordinary differential equations describing the time evolution of the lambda parameters still had to be derived by hand. For scalar and two dimensional systems this is not too time-consuming. For higher dimensions and degrees of initial distributions, deriving the ordinary differential equations for the lambdas by hand will become a time expensive challenge. Therefore, we looked into a system which would write the equations for the lambdas by itself. The aim was that only the dimension of the state vector x, the degree of the log-polynomial and the constant matrix $A \in \mathbb{R}^{n \times n}$ needed to be specified in order for the Matlab script to work.

For this, Mathlab's symbolic toolbox had to be used. However, this complicates the script quite a bit. Especially the order of the symbolic lambdas would be mixed when applying different functions of the toolbox. It proved complicated to obtain the ordinary differential equations for the lambda-parameters for arbitrary systems. However, a script was written for the two dimensional fourth degree log-polynomial case. This script is easy to adjust for higher degrees and dimensions, which does still make it useful for writing the lambda equations.

7 Conclusion

In this thesis, a way to model the time evolution of initial distributions in first-order linear dynamical systems is derived using the Liouville Equation. The multivariate model was tested for a system with defined inputs, resulting into the intuitively expected evolution of the initial distribution. Also the scalar model evolved as expected. Hence, these models can be used to construct the probability distribution of a dynamic system at any time.

The script to derive the ordinary differential equations describing the lambda-parameters works for two dimensions and a fourth degree polynomial. However, the script is very intuitive and can easily be changed to satisfy any other combination of dimension and polynomial degree.

Most dynamical systems are not first-order linear systems. Although nonlinear systems can be locally approximated by linear systems, it would be preferable to have a way to see how initial distributions evolve for these systems too. However, nonlinear systems would not allow us to derive the log-polynomial distributions, so it remains to be investigated if a similar approach can be implemented for nonlinear systems.

Appendix A

Scalar Dynamical System

```
a = -2; % this is the parameter a in the equation dxdt = a*x
% Here we define the initial values for the lambdas.
%We assume that the log-density is of the form
% log[p(x)]=lambda_n*x^n+lambda_(n-1)*x^(n-1)+...+lambda_1*x+lambda_0.
% The vector initial_values contains the initial values of the lambdas,
% starting from lambda_0 and going up. So, if initial_values has three
% elements, we know that these are [lambda_0 lambda_1 lambda_2]
initial_values = [-0.25+log(1/(2*sqrt(pi))) 0.5 -0.25];

n = length(initial_values)-1; % this is the degree of the log-density ...
    polynomial
t_end = 1; % final simulation time
steps = 20; %number of time steps taken
T = linspace(0,t_end,steps); % this line creates the vector of simulation times

[Tout,L]=ode45(@(t,l) L_ODE(t,l,a),T,initial_values)
xsteps=100000;
X=linspace(-10,10,xsteps);%the X's we take at which the pdf is evaluated
Y=zeros([xsteps,steps]);
% constructing the log-polynomial probability density function
for i=1:length(Tout)
    P=L(i,1);
    for k=2:n+1;
        P=P+(L(i,k)*(X.^(k-1)));
    end
    Y(:,i)=exp(P)';
    %Y(:,i) contains the values of the distribution at the evaluated X's
    %for time Tout(i)
end

Time=zeros([xsteps,steps]);
for i=1:steps

Time(:,i)=Tout(i); %making a matrix where each column consists of one ...
    particular time, s.t. we can plot the distributions
end
for i=1:steps
    plot3([Time(:,i)],X,Y(:,i));%plotting the distributions at different ...
        times in the same 3d plot
    hold on
end
hold off
xlabel('time')
ylabel('X')
zlabel('probability')
```

```
function dlambdadt = L_ODE(t,l,a)
%This function is used to calculate each individual lambda from the
%Lambda_script_scalar script.
n = length(l)-1; % we need to figure out what degree our log-density ...
    polynomial is
coeffs = [1:n]'; % here we define the coefficients of lambda_n, ...
    lambda_(n-1) etc. in the differential equations

dlambdadt = [-a;-coeffs.*l(2:end)*a];
end
```

Vector Dynamical System

```
A= [-1,2;0,-2]; % this is the matrix in dXdt=AX
a11=A(1,1);
a12=A(1,2);
a21=A(2,1);
a22=A(2,2);

% Here we use the initial state to determine the initial lambda values.
% For X_0 ~ N([mu_1; mu_2],[s11, s12; s21, s22]), we have:
mu_0=[1,1]; %initial mean vector
sigma_0=[1,0;0,1]; %initial covariance matrix
mu_1=mu_0(1);
mu_2=mu_0(2);
s11=sigma_0(1,1);
s12=sigma_0(1,2);
s21=sigma_0(2,1);
s22=sigma_0(2,2);

% Here we define the initial values for the lambdas. We assume that the ...
    log-density is of the form
% log[p(x)]=lambda_1*x(1)^2+lambda_2*x(2)^2+lambda_22*x(2)
% +lambda_11*x(1)+lambda_12*x(1)*x(2)+lambda_0

% The vector 'initial_values' contains the initial values of the lambdas,
% in the order of [lambda_0 lambda_1 lambda_2 lambda_11 lambda_22 lambda_12].

% the following automatically determines the initial distribution if X ...
    follows a normal distribution
s=s11*s22-s21*s12;
l_0=-1/(2*s)*(mu_1^2*s22-mu_1*mu_2*s12+s11*mu_2^2 ...
    -s21*mu_1*mu_2+log(1/(2*pi*sqrt(s))));
l_1=-1/(2*s)*(s22);
l_2=-1/(2*s)*(s11);
l_11=-1/(2*s)*(-2*mu_1*s22+s12*mu_2+s21*mu_2);
l_22=-1/(2*s)*(-2*mu_2*s11+mu_1*s12+s21*mu_1);
l_12=-1/(2*s)*(-s12-s21);

initial_values = [l_0 l_1 l_2 l_11 l_22 l_12];
% For a non-normal distribution the initial values can just be plugged in.

t_end = 5; % final simulation time
steps = 20; %number of steps taken
T = linspace(0,t_end,steps); % this line creates the vector of simulation times
[Tout,L]=ode45(@(t,l) multivariate_ode(t,l,a11,a12,a21,a22),T,initial_values);

xsteps=2500;
x1=linspace(-0.5,0.5,xsteps);%the X's we take at which the pdf is evaluated
x2=linspace(-1,1,xsteps);
Y=zeros([xsteps,xsteps]);

%constructing the log-polynomial pdf and plotting the distributions over time
for i=1:length(Tout)
    for k=1:xsteps %going through all values of x1
        for j=1:xsteps %going through all values of x2
            P=L(i,1)+L(i,2)*(x1(k).^2)+L(i,3)*(x2(j).^2)+L(i,4)*x1(k) ...
                +L(i,5)*x2(j)+L(i,6)*x1(k)*x2(j);
            Y(k,j)=exp(P); % the matrix conatining the value of the pdf at ...
                each x1 and x2
        end
    end
    s=surf(x1,x2,Y)
    s.EdgeColor='none';
    pause(0.5)
end
```

```
function dlambdadt = multivariate_ode(t,l,a11,a12,a21,a22)
%This function is used to calculate each individual lambda from the ...
    lambda_script_multivariate script.
dlambdadt = ...
    [-a11-a22;-2*a11*l(2)-a21*l(6);-a12*l(6)-2*a22*l(3);-a11*l(4)-a21*l(5);
-a22*l(5)-a12*l(4);-a11*l(6)-2*a12*l(2)-2*a21*l(3)-a22*l(6)];
% the ode's that need to be solved in order of lambda_0, lambda_1, ...
    lambda_2, lambda_11, lambda_22, lambda_12.

% the resulting L matrix will have the order [lambda_0 lambda_1 lambda_2 ...
    lambda_11 lambda_22 lambda_12]
end
```

Lambda Ordinary Differential Equations

```
d = 2; % we are considering 2-D systems
X = {};
maxdeg = 4; % this is the maximum degree of a polynomial in two variables

% We need to generate all monomials of d variables of degree less than or
% equal to k. The loop below does this operation

for k = 0:maxdeg
    x=sym('x',[d,1]);
    m = nchoosek(k+d-1,d-1); %the number of combinations of k+d-1 items ...
        taken d-1 at a time
    dividers = [zeros(m,1),nchoosek((1:(k+d-1))',d-1),ones(m,1)*(k+d)];
    a = diff(dividers,1,2)-1; %calculating the first-order difference ...
        between the columns, where 1 is substracted from the second columns ...
        evaluated each time
    PBase = cell(1, size(a,1));
    for i = 1:size(a,1)
        PBase{i} = prod(x.' .^ a(i,:));
    end
    X = [X PBase]; %monomials
end

% Building the log-density function logP, a polynomial of degree maxdeg in ...
    d variables

L = sym('L',[length(X),1]); % we generate as many lambda coefficients as ...
    monomials
syms P real
logP = 0;
for n = 1:length(X)
    logP = logP + L(n)*X{n};
end

% We define a linear dynamical system in two dimensions, needs to change ...
    for higher dimensions
syms a11 a12 a21 a22  real
A = [[a11 a12 ];[a21 a22]];
F = A*x; % this line writes the right-hand side of the system equations

% With the log-density and the system equations ready, we calculate the
% term F(x)*dlogP/dx which comes from the Liouville equation

Dp = expand(F(1)*diff(logP,x(1))+F(2)*diff(logP,x(2)));

[Coefs_dp,Mono_dp] = coeffs(Dp,x); % extract the monomial coefficients of Dp

% build the right-hand side of the lambda ODEs
% -F(x)*dlogP/dx - trace(A)
Coefs_dp = [-trace(A) -Coefs_dp]';
```

```
%%
% up to here, all calculations were done with symbols. Now we need to
% substitute the actual system parameters in the equations...
Coefs_dp = subs(Coefs_dp,{a11,a12,a21,a22},[-1 0.1 0.2 -2]); %the righthand ...
    side in the system of ODEs to solve for the lambdas
% and finally we convert the vector Coefs_dp into a Matlab function
% which takes the vector of Lambdas as input and returns the value of the
% right-hand side of the lambda ODEs.
syms t real

% CAREFUL: order of L elements needs to correspond to the order of the ...
    monomials (according to line 26)!
Lambda_fun = ...
    matlabFunction(Coefs_dp,'Vars',{t,[L(1);L(15);L(14);L(10);L(13);L(9); ...
    L(6);L(12);L(8);L(5);L(3);L(11);L(7);L(4);L(2)]}); %L(:) order is the ...
    corresponding to the order of Mono_dp mononomials.

% The function Lambda_fun takes an argument L (a set of coefficients of the
% initial log-density logP) and returns the right-hand side of the lambda
% ODEs. It can be called by writing Lambda_fun(lambda0), where lambda0 is a
% column vector of lambda values.

% Note: The order of the lambda coefficients is provided by the vector
% Mono_dp. This is useful for building the initial density of the system
% (i.e. the initial values of the lambdas).
% This order of lambdas should also be kept in mind when defining the
% initial condition below

lambda0=[0;0;0;0;0;0;-1;0;0;-0.2;0;0;0;-1;-2];

t_end = 3; % final simulation time (let's assume that t0 = 0 for simplicity)
steps = 20; %number of steps to take
T = linspace(0,t_end,steps); % this line creates the vector of simulation times

[Tout,La]=ode45(Lambda_fun,T,lambda0);
%% Simulating the density graph
if d==2
    xsteps=1000;
    xleft=-3;
    xright=3;
    x1=linspace(xleft,xright,xsteps); %the X's we take at which the pdf is ...
        evaluated
    x2=linspace(xleft,xright,xsteps);
    [X1,X2] = meshgrid(x1,x2);

    for i=1:length(Tout) % fill in here until which time we want to see the ...
        distribution evolve
    % Another point to pay attention: when reconstructing P, the La's are ...
        used in the order in which they
    % appear in the La vector. In this way La(i,2) corresponds to L15, ...
        which is the correct
    % coefficient for x1^4
        P = La(i,1)+La(i,2)*(X1.^4)+La(i,3)*((X1.^3).*X2)+La(i,4)*(X1.^3) ...
            +La(i,5)*((X1.^2).*(X2.^2))+La(i,6)*((X1.^2).*X2)+La(i,7)*(X1.^2) ...
            +La(i,8)*(X1.*(X2.^3))+La(i,9)*(X1.*(X2.^2))+La(i,10)*(X1.*X2) ...
            +La(i,11)*X1+La(i,12)*(X2.^4)+La(i,13)*(X2.^3)+La(i,14)*(X2.^2) ...
            +La(i,15)*X2;
        Y = exp(P); % the matrix containing the value of the pdf at each x1 ...
            and x2
        s = surf(X1,X2,Y);
        shading interp
        view(2)
        pause(0.1)
    end
end
```

References

[1] Virginia W. Noonburg. *Differential equations : from calculus to dynamical systems*, volume 43. Providence, Rhode Island : MAA Press, an imprint of the American Mathematical Society, 2019.

[2] Matthew A. Carlton and Jay L. Devore. *Probability with Applications in Engineering, Science, and Technology*. Springer New York, 2014.

[3] Martin Ehrendorfer. The liouville equation in atmospheric predictability. In *Seminar on Predictability of weather and climate, 9-13 September 2002*, pages 47–81, Shinfield Park, Reading, 2003. ECMWF, ECMWF.

[4] Brian P Ingalls. *Mathematical modeling in systems biology : an introduction*. Cambridge, MA : The MIT Press, 2013.