Solving the Master Equation with Krylov Subspace Approximation

Joris de Jong
S4192044

April 16, 2021

Abstract

Reaction networks are used for modeling the dynamics of complex interactions between different groups. Modeling those dynamics is often attempted by solving the so-called master equation. In this report, the master equation was numerically approximated by a technique called Krylov subspace approximation. This technique was then compared to another method of modelling the dynamics of a reaction network, namely Monte Carlo simulation using the Gillespie algorithm. The Gillespie algorithm was used to sample trajectories from the master equation and Monte Carlo simulations were used to infer statistical behavior of the system. The two methods were compared in a case study in which opinion formation was modelled in a sociological network. The Krylov subspace approximation method yielded much more accurate and smoother results than the Monte Carlo method. Although, both methods have its shortcomings, in our case study, the KSA technique is preferred. This report can be seen as an introduction to both the KSA method and Monte Carlo simulations using the Gillespie algorithm.

1 Introduction

Complex networks of interactions lie at the heart of many problems of scientific interest. This has resulted in the interdisciplinary study of such reaction networks, e.g. chemical reaction networks, cellular networks, epidemiological networks, ecological networks, social networks.

Modelling the dynamics of such reaction networks is commonly approached by employing a system of first-order differential equations, known as the *master equation*. The master equation determines how the joint probability mass function of the system evolves over time. However, reaction networks can often display nonlinear behavior, which results in an inability of solving the master equation analytically. Therefore, modelling nonlinear reaction networks frequently requires numerical approaches for solving the master equation. One method is by viewing the master equation as the first-order differential equation $\frac{d\mathbf{p}}{dt} = \mathcal{P}\mathbf{p}(t)$, where $\mathbf{p}(t)$ is a vector containing the probabilities over all possible states at time t and $\mathcal{P}$ is a large sparse matrix. Solving this equation yields the solution $\mathbf{p}(t) = \exp(t\mathcal{P})\mathbf{p}(0)$. Since $\mathcal{P}$ is a large sparse matrix, an efficient algorithm is needed to solve this equation. The *Krylov Subspace Approximation* (KSA) method is a quick and efficient technique for solving exactly such equations. The KSA method will be compared to a Monte Carlo approach for modelling the dynamics of a reaction network, where many trajectories are sampled using the Gillespie algorithm.

The remainder of this report is structured as follows. First, the mathematical background of reaction networks is introduced as well as how to sample a trajectory with the Gillespie algorithm and how to solve the master equation using the KSA technique. Then, both methods are are applied to a case study where the differences between the Monte Carlo method and the KSA method are shown. In the case study we will have a look at the dynamics of public and private opinion formation in both a liberal and totalitarian political system. In the last sections, the results are discussed and an outlook is given on other methods.

This report serves as an introduction to reaction networks and how to model its dynamics using two different approaches, Monte Carlo simulations using the Gillespie algorithm and the KSA method. The case study is based on an example given in the article written by J. Goutsias and G. Jenkinson [1]. The code for this report was written in R and is available at https://github.com/jorispdejong/OpinionFormation.

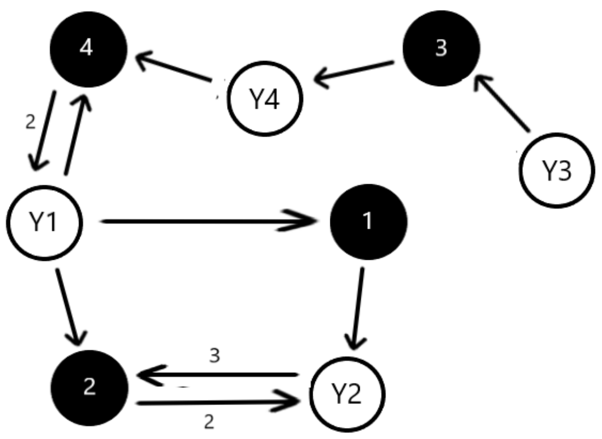

Figure 1: A bipartite, weighted, directed graph representing a reaction network as described in the text. The black dots represent the reactions and the white dots represent the species.

2 Reaction Networks

Networks of chemical reactions are used abundantly for modelling biochemical reactions in cells. Fortunately, the mathematical and computational properties of such chemical reactions networks can be extended to model other complex networks. Still, much of the nomenclature is rooted in biochemistry.

2.1 Mathematical Setup

A network of reactions comprises a group of species and a one or more reactions. The number of elements in each group of species is referred to as the population of the species. The reactions govern how the different species interact with each other and how the populations change over time. For each reaction, a group of species, called *reactants*, reacts to produce a different group of species, called *products*. Let N define the number of species $Y_1, Y_2, ..., Y_N$ and let K define the number of reactions. We can then write the interactions as

$$\sum_{n=1}^{N} r_{nk} X_x \longrightarrow \sum_{n=1}^{N} p_{nk} X_n, \quad k \in 1, ..., K. \tag{1}$$

The coefficients r_{nk} and p_{nk} are called the *stoichiometric coefficients* of the reactants and the products, respectively. They determine how the number of species change in the reactions. The difference between r_{nk} and v_{nk} determines how Y_n changes after a reaction. Therefore, this difference, $v_{nk} = r_{nk} - p_{nk}$, is called the *net* stoichiometric coefficient.

A reaction network can be graphically represented by a bipartite, directed, weighted graph. Figure 1 shows the graphical representation of the following reaction network:

$$\begin{aligned} Y_1 &\longrightarrow Y_2, \\ Y_1 + 3Y_2 &\longrightarrow 2Y_2, \\ Y_3 &\longrightarrow Y_4, \\ Y_1 + Y_4 &\longrightarrow 2Y_1. \end{aligned} \tag{2}$$

The black dots represent the reactions and the white dots represent the species. An arrows going from a white dot to a black dot represent the reactants and the arrows going from a black dot to a white dots represent the products. The labels on the arrows give the weight of that edge. They correspond to the stoichiometric coefficients of the reaction network.

Another way of compactly showing the information of a reaction network is via the matrices $R = \{r_{nk}\}$

and $P = \{p_{nk}\}$, where $n \in \{1,..,N\}$ and $k \in \{1,..,K\}$. In the example reaction network given by the reactions in (2), they are defined as

$$R = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 0 & 2 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Furthermore, how the population of the species changes over time can now be written in matrix form in the so-called *net-effect matrix* or *stoichiometric matrix*, which is given by $V = \{v_{nk}\}$ where $n \in \{1,..,N\}$ and $k \in \{1,..,K\}$. For the reaction network given in Equation 2 the net-effect matrix is given by

$$V = \begin{bmatrix} -1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

where the columns contain the reactions and the rows contain the species.

2.2 Population Processes

The stochastic dynamics on a reaction network can be described by either keeping track of the number of times a reaction occurs in a given time period or the populations at a given time. In case of the former, define $\mathbf{Z}(t)$ as a $K \times 1$-dimensional vector containing the elements $Z_k(t)$ for $k = 1, 2, ..., K$, which is the number of times the k-th reaction occurs in the time interval $[0, t)$. The stochastic dynamics of the reaction network are then given by the process $\{\mathbf{Z}(t), t \geq 0\}$. In case of the latter approach let $\mathbf{Y}(t) = [Y_1(t)\ Y_2(t)\ ...\ Y_N(t)]^T$ denote the vector containing the populations of the species at time t. The multivariate stochastic process $\{\mathbf{Y}(t), t \geq 0\}$ will be referred to as the *population process*. Note that $\mathbf{Y}(t)$ can be computed via $\mathbf{Y}(t) = \mathbf{Y}(0) + V\mathbf{Z}(t)$, where $\mathbf{Y}(0)$ is the starting populations of the species and V is the net-effect matrix.

To give a bit more intuition of the population process, assume that the population of the species at time t is given by $\mathbf{X}(t)$ and reaction k is triggered in the time interval $t + \delta t$, then the population of the species at time $t + \delta t$ will be given by $\mathbf{Y}(t + \delta t) = \mathbf{Y}(t) + V[, k]$, where $V[, k]$ denotes the k-th column of V.

2.3 Master Equation

The population process $\{\mathbf{Y}(t), t \geq 0\}$ is a Markov process [1], which means that in a small time interval $[t, t + \delta t]$ the probability of a reaction to occur is proportional to δt and $\mathbf{Y}(t)$. We can then write $\mathbb{P}[\text{a reaction } k \text{ is triggered in time interval } [t, t + \delta t] | \mathbf{Y}(t) = \mathbf{y}] = \pi_k(\mathbf{y})\delta t + o(\delta t)$, where $\pi_k(\mathbf{y})$ is the so-called *propensity function* and $o(\delta t)$ is a term that goes to zero faster than $\pi_k(\mathbf{y})\delta t$ [2]. Now define $p(\mathbf{y}, t) = \mathbb{P}[\mathbf{Y}(t) = \mathbf{y} | \mathbf{Y}(0) = \mathbf{y_0}]$, where the dependency on $\mathbf{y_0}$ is omitted for notational ease. Tt turns out that under the above conditions, $p(\mathbf{y}, t)$ evolves over time according to the following partial differential equation

$$\frac{\partial p(\mathbf{y}, t)}{\partial t} = \sum_{k=1}^{K} \pi_k(\mathbf{y} - V[, k]) p(\mathbf{y} - V[, k], t) - \sum_{k=1}^{K} \pi_k(\mathbf{y}) p(\mathbf{y}, t). \tag{3}$$

This equation is known as the *master equation*. Intuitively speaking, in an infinitesimal time interval of length δt, the first sum increases the probability $p(\mathbf{y}, t)$ from potential state changes that cause the state to change to $\mathbf{y}$, whereas the second sum decreases the probability $p(\mathbf{y}, t)$ because of the potential state changes that cause the state to change from $\mathbf{y}$. Conventionally, if a reaction cannot occur because of e.g. constraints on the populations or not enough reactants, then $p(\mathbf{y}, t) = 0$.

2.4 Gillespie Algorithm

A trajectory can be sampled from the master equation using the Gillespie algorithm. First, set the initial state of the system $\mathbf{y_0}$. Then repeat the following two steps for a given time period.

Step 1
Assume that the system is in state $\mathbf{y}(t)$, then we first sample the waiting time for the next reaction to occur. Let τ be the waiting time, then

$$\tau \backsim \exp\left(\sum_{k=1}^{K} \pi_k(\mathbf{y}(t))\right).$$

Increase the current time t by τ.

Step 2
Next, it needs to be decided which reaction will occur at time $t + \tau$. This is done by drawing a sample from the probability mass function

$$pmf(k,t) = \frac{\pi_k(\mathbf{y}(t))}{\sum_{k=1}^{K} \pi_k(\mathbf{y}(t))},$$

which yields a value $1 \geq j \geq K$. Add the j-th column of V to $\mathbf{y}(t)$.

Since the Gillespie algorithm merely samples from the master equation it will only give a possible trajectory. In order to infer statistical properties of the system we can use Monte Carlo sampling where we simulate S trajectories $\{\mathbf{y}^{(s)}(t), t \geq 0\}$ for $s = 1, 2, ..., S$ and then we compute the dynamics of the moments of the system using Monte Carlo estimators. For example, $p(\mathbf{y}, t)$ can be approximated by

$$\widehat{p}(\mathbf{y}, t) = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{\mathbf{y}^{(s)}=\mathbf{y}}, \tag{4}$$

where $\mathbb{1}_{\mathbf{y}^{(s)}=\mathbf{y}}$ is the indicator function which is 1 if $\mathbf{y}^{(s)} = \mathbf{y}$ and 0 otherwise.

Note that many trajectories need to be sampled in order to achieve an accurate approximation of $p(\mathbf{y}, t)$, which may be unfavorable. Nevertheless, the Monte Carlo sampling in combination with the Gillespie algorithm is an robust and simple way to quickly gain intuition about the dynamics of the reaction network.

2.5 Krylov Subspace Approximation

Instead of sampling trajectories to infer statistical behavior about the system, we can also attempt to solve the master equation. Since it is often not possible to solve the master equation analytically, we must resort to numerical approximations. Note that the master equation can be written as a linear system of coupled differential equation as follows:

$$\frac{d\mathbf{p}(t)}{dt} = \mathcal{P}\mathbf{p}(t), \tag{5}$$

where $\mathbf{p}(t) = [p(\mathbf{y_1}, t)\ p(\mathbf{y_2}, t)\ ...p(\mathbf{y_M}, t)]^T$ is a $M \times 1$-dimensional vector with $\mathbf{y_i}$, $i = 1, 2..., M$, being all the possible states $\mathbf{y}$ can attain, and $\mathcal{P}$ is a large sparse $M \times M$ matrix [3]. The elements of $\mathcal{P}$ are given by

$$\mathcal{P}_{ij} = \begin{cases} -\sum_{k=1}^{K} \pi_k(\mathbf{y_i}), & \text{if } i = j \\ \pi_k(\mathbf{y_i}), & \text{for all } j \text{ such that } \mathbf{y_j} = \mathbf{y_i} + V[, k] \\ 0, & \text{otherwise} \end{cases}$$

for $i, j = 1, 2, ..., M$. In our case study, we consider a small constrained network, where $-40 \leq y_1, y_2 \leq 40$, leading to $M = 81 \times 81 = 6561$ possible states. If we define the state space $\mathcal{Y}$ as the space in which the $\mathbf{y}$

live, then M is equal to the cardinality of the state space $\mathcal{Y}$. Therefore, solving equation 5 only works for finite M. Assuming that M is finite, the solution to Equation 5 is given by

$$\mathbf{p}(t) = \exp(t\mathcal{P})\mathbf{p}(0), \tag{6}$$

for $t > 0$. One technique for solving this equation is the *Krylov Subspace Approximation* (KSA) technique. In simulations it will be solved iteratively, meaning that

$$\mathbf{p}(t+\tau) = \exp(t\mathcal{P})\mathbf{p}(t) \tag{7}$$

for a small time step τ.

In order to understand the KSA method, first note that the matrix exponential can be expanded using Taylor, which yields

$$\widehat{\mathbf{p}}(t+\tau) = c_0\mathbf{p}(t) + c_1\tau Q\mathbf{p}(t) + ... + c_{m-1}(\tau Q)^{K_0-1}\mathbf{p}(t), \tag{8}$$

where the coefficients are given by $c_i = (i!)^{-1}$ for $i = 0, 1, ..., m-1$. Every polynomial of order m-1 has an optimal solution in the *Krylov subspace* $\mathcal{K}_m(t\mathcal{P}, \mathbf{p}(t)) = \text{span}\{\mathbf{p}(t), (t\mathcal{P})\mathbf{p}(t), (t\mathcal{P})^2\mathbf{p}(t), ..., (t\mathcal{P})^{m-1}\mathbf{p}(t)\}$ [4]. Therefore, we will attempt to find the optimal solution $\widehat{\mathbf{p}}(t+\tau)$ in $\mathcal{K}_m(t\mathcal{P}, \mathbf{p}(t))$. We start by computing an orthogonal basis $\{v_1, v_2, ..., v_m\}$ for $\mathcal{K}_m(t\mathcal{P}, \mathbf{p}(t))$ via the Arnoldi procedure. The Arnoldi procedure computes the matrix $V_m \in \mathbb{R}^{N\times m}$, whose columns contains the orthogonal basis vectors $v_1, v_2, ..., v_m$, which satisfies the Arnoldi decomposition

$$AV_m = V_{m+1}H_{m+1,m}, \tag{9}$$

where $H_{m+1,m} \in \mathbb{R}^{m+1\times m}$ is an upper-Hessenberg matrix [1]. Let $H_{m,m}$ denote the first m rows of $H_{m+1,m}$ and $h_{m+1,m}$ the element of $H_{m+1,m}$ on the $m+1$-th row and m-th column. Then we can rewrite Equation 9 as

$$AV_m = V_mH_{m,m} + h_{m+1,m}v_{m+1}\mathbf{e}_m^T, \tag{10}$$

where $\mathbf{e}_m^T$ is the m-dimensional vector $[0\ ...\ 0\ 1]^T$. If $h_{m+1,m}v_{m+1}e_m^T$ is small, the Krylov subspace is close to an invariant subspace of $t\mathcal{P}$, meaning that if we multiple any vector in the Krylov subspace by $t\mathcal{P}$ it will again be a vector in the Krylov subspace plus a small addition.

The Arnoldi procedure lets the first basis vector v_1 be such that it is the normalized $\mathbf{p}(t)$, so $\mathbf{p}(t) = ||\mathbf{p}(t)||_2V_m\mathbf{e}_1$, where $\mathbf{e}_1 = [1\ 0\ ...\ 0]^T \in \mathbb{R}^m$. Now, let $\mathbf{p}_{opt}(t+\tau)$ be the optimal solution to Equation 7, which can be stated as finding the vector $\mathbf{y}_{opt} \in \mathbb{R}^m$ such that $\mathbf{p}_{opt}(t+\tau) = V_m\mathbf{y}_{opt}$. We can find $\mathbf{y}_{opt}$ by minimizing the linear least squares problem

$$\mathbf{y}_{opt} = \min_{\mathbf{y}\in\mathbb{R}^m}||V_m\mathbf{y} - \exp(t\mathcal{P})\mathbf{p}(t)||_2^2, \tag{11}$$

which gives

$$\mathbf{y}_{opt} = (V_m^TV_m)^{-1}V_m^T\exp(t\mathcal{P})\mathbf{p}(t) = V_m^T\exp(t\mathcal{P})\mathbf{p}(t), \tag{12}$$

where $V_m^TV_m = I$. Then $\mathbf{p}_{opt}(t+\tau)$ can be expressed as

$$\mathbf{p}_{opt}(t+\tau) = V_m\mathbf{y}_{opt} = ||\mathbf{p}(t)||_2V_m(V_m^T\exp(t\mathcal{P})V_m)\mathbf{e}_1. \tag{13}$$

This still requires $\exp(t\mathcal{P})$ to be computed, which is what we want to avoid. Therefore, we approximate $V_m^T\exp(t\mathcal{P})V_m \approx \exp(tV_m^T\mathcal{P}V_m) = \exp(tH_{m,m})$ [5]. This gives

$$\mathbf{p}_{opt}(t+\tau) \approx ||\mathbf{p}(t)||_2V_m\exp(tH_{m,m})\mathbf{e}_1 = \mathbf{p}_{m,opt}(t+\tau). \tag{14}$$

Notice how $\mathbf{p}_{m,opt}(t+\tau)$ is computationally much cheaper than $\mathbf{p}_{opt}(t+\tau)$, because m is usually much smaller than M. The computation of the matrix exponential in $\mathbf{p}_{m,opt}(t+\tau)$ can be done by classical methods such as the Padé method.

[1]An upper-Hessenberg matrix is basically an upper triangular matrix with an extra non-zero diagonal, so all the entries h_{ij} are zero for $i > j+1$.

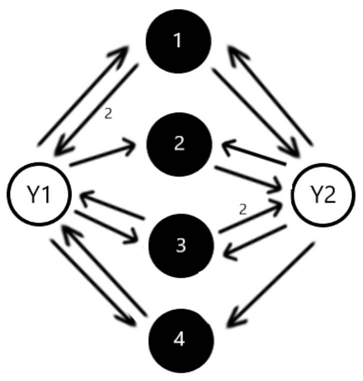

Figure 2: Graphical representation of the reactions that govern opinion formation as described in (15). The black dots represent the reactions and the white dots represent the species.

3 Case Study: Opinion Formation

In this case study, a social reaction network is analyzed to showcase the difference between the Monte Carlo method and the KSA method. The model is one of opinion formation in society. Let an individual possess two different opinions on the current reigning ideology of the system, a private opinion and a public opinion, where each opinion can either be in favor or against. In a liberal system the public and private opinion should not differ, but they may differ in a totalitarian system, where there is pressure inflicted on the population.

Let there exist $2L$ individuals who all each have a public and a private opinion. If the opinion is in favor of the ruling ideology then it takes the value $+0.5$ and if it is against then it takes the value -0.5. In order to measure the overall opinion of the individuals, define Y_1 as the net public opinion and Y_2 as the net private opinion, where the net opinion is simply the sum over all opinions. This means that $-L \leq Y_1, Y_2 \leq L$, where $Y_1 = -L$ means that everyone disagrees publicly with the ruling ideology and $Y_1 = L$ means that everyone agrees publicly with the ruling ideology and similarly for Y_2, but then privately. Now, there are 2 species, namely Y_1 and Y_2, and there are 4 possible reactions. An individual can change there private opinion from in favor to against and vice versa, and an individual can change there public opinion from in favor to against and vice versa. This amount to 4 reactions that can be written as follows:

$$\begin{aligned}
Y_1 + Y_2 &\longrightarrow 2Y_1 + Y_2, \\
Y_1 + Y_2 &\longrightarrow Y_2, \\
Y_1 + Y_2 &\longrightarrow Y_1 + 2Y_2, \\
Y_1 + Y_2 &\longrightarrow Y_1.
\end{aligned} \tag{15}$$

The first reaction governs how the private opinion can influence an individual to change their public opinion from against to in favor of the ruling beliefs, whereas in the second reaction the private opinion sways an individual to change their public opinion from in favor to against. The third and fourth reactions are similar, but instead the public opinion influences an individual to change their private opinion. The graph of the reactions in (15) is shown in Figure 2. It can be shown that the propensity functions for these reactions are

$$\begin{aligned}
\pi_1(\mathbf{y}) &= \kappa_1(L - y_1)\exp(a_1 y_1 + a_2 y_2), \\
\pi_2(\mathbf{y}) &= \kappa_1(L + y_1)\exp(-a_1 y_1 - a_2 y_2), \\
\pi_3(\mathbf{y}) &= \kappa_1(L - y_2)\exp(a_3 y_1), \\
\pi_4(\mathbf{y}) &= \kappa_1(L + y_2)\exp(-a_3 y_1),
\end{aligned} \tag{16}$$

where $\kappa_1, \kappa_2 > 0$ are the reaction parameters, $a_1 \geq 0, a_2 > 0, a_3$ are the sociological model parameters, and y_1, y_2 are the net public and private opinion, respectively [1]. To give a little intuition into the parameters, a_1 determine the amount of pressure from the public opinion, e.g. a_1 is 0 in a system of free speech, but a_1

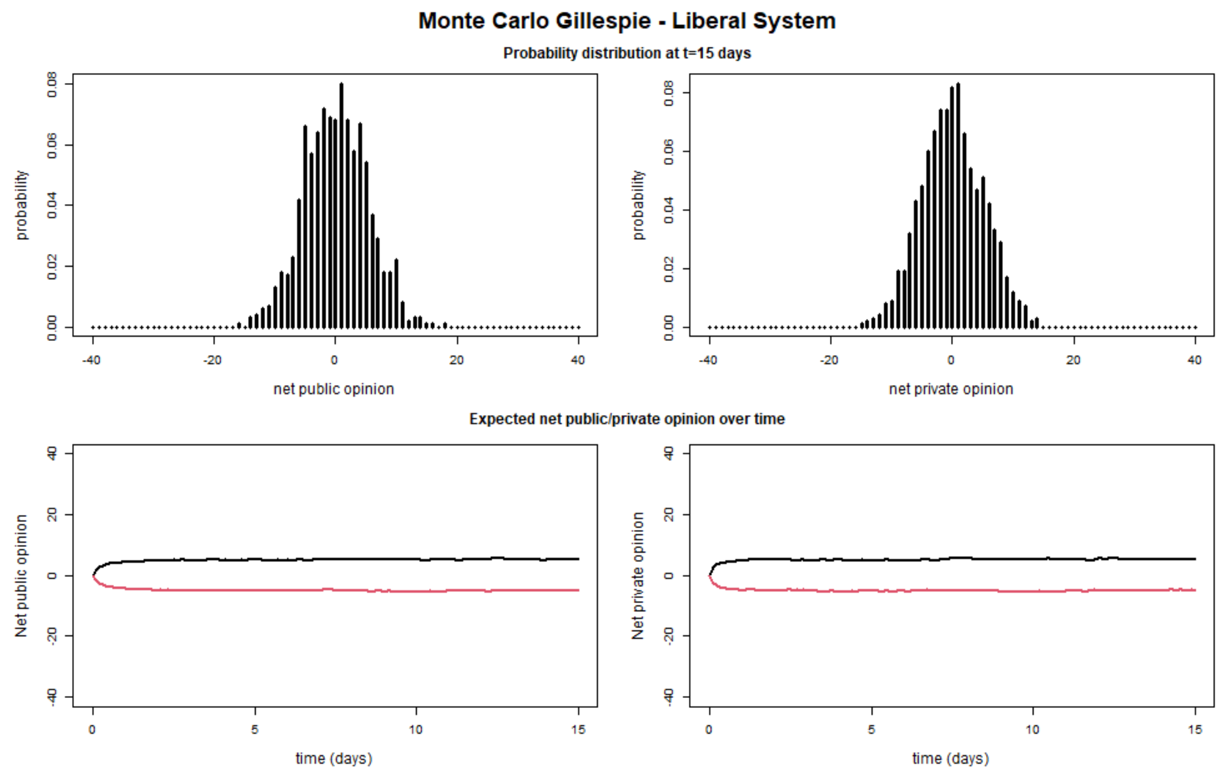

Figure 3: Results of Monte Carlo simulations using the Gillespie algorithm for the liberal system. In the upper left/right plots the probability distribution is plotted against the net public/private opinion at $t = 15$ days. In the lower left/right plots the expected value plus and minus the standard deviation of the net public/private opinion is plotted over time. The black line is the mean plus the standard deviation and the red line is the mean minus the standard deviation.

is large for a totalitarian system. a_2 controls the pressure from the private beliefs on the public opinion and a_3 decides how challenging the private beliefs are towards the ruling ideology, which means that $a_3 > 0$ if the individuals are positive about the ruling ideology and $a_3 < 0$ if the ruling ideology will be opposed.

3.1 Simulation Setup

We will be simulating two different systems: A liberal system and a totalitarian system. Let $L = 40$, meaning that there are 80 individuals in the system. The two systems can be fully defined by the parameters a_1, a_2 and a_3. For the liberal system the parameters will be set to $a_1 = 0$, $a_2 = 1/2L$, $a_3 = 1/2L$ and for the totalitarian system the parameters are set to $a_1 = 3/2L$, $a_2 = 1/L$, $a_3 = -1/4L$. Furthermore, the reaction parameters are set to $\kappa_1 = 20/L$ and $\kappa_2 = 40/L$. The total simulation time $total_t ime = 15$ and time step $dt = 0.1$ will be in days. Both the net public and private opinion are initialized to 0. The number of Monte Carlo simulations is set to 1000 and the Krylov subspace dimension is set to $m = 10$. The For the Krylov subspace dimension, different values were tried and $m = 10$ seems to be a small value that still yields good results.

The simulations are implemented in R. The code for the Monte Carlo simulations and the Gillespie algorithm is based mostly on the lecture notes and requires only the R package *pracma*. The code for the KSA method is based partly on code provided by [1] and partly on the matlab package *Expokit*, and requires only the R packages *pracma* and *Matrix*. Expokit is a matlab implementation of the KSA technique for solving equations of the form $w = \exp{(tA)}v$ for a large sparse matrix A. We have rewritten the function *expv* to R language.

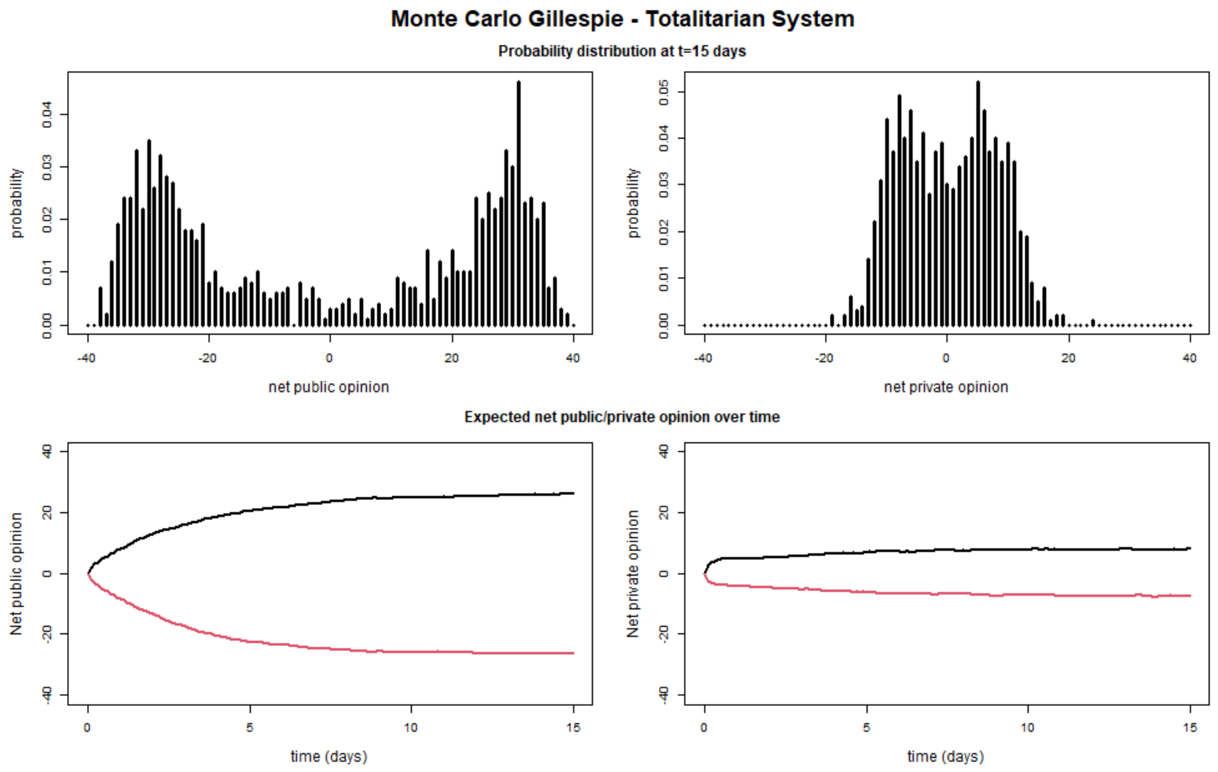

Figure 4: Results of Monte Carlo simulations using the Gillespie algorithm for the totalitarian system. In the upper left/right plots the probability distribution is plotted against the net public/private opinion at $t = 15$ days. In the lower left/right plots the expected value plus and minus the standard deviation of the net public/private opinion is plotted over time. The black line is the mean plus the standard deviation and the red line is the mean minus the standard deviation.

3.2 Monte Carlo Using the Gillespie Algorithm

The Gillespie algorithm is used to sample 1000 trajectories. In each run, there are $1 + 15/0.1 = 151$ time steps. Since the Gillespie algorithm will sample the waiting time from the exponential distribution it will not coincide with the predetermined time steps. However, in order to compute the Monte Carlo estimators we interpolate the values to each time step using nearest neighbor interpolation. This allows for all the runs of the Monte Carlo simulation to be compared to each other. After running the Monte Carlo simulation, the expected mean and expected standard deviation are computed for each time step. Also, the probability of the net public and private opinion at the final time step for every possible value of y_1 and y_2 is computed. These simulations are run for both the liberal system and totalitarian system and took approximately 38.44 and 34.56 seconds, respectively. The resulting 2 plots are shown in Figure 3 and 4.

The upper plots in Figure 3 show that on the final time step the liberal system has an expected net public and private opinion centered around 0. The standard deviation is approximately 5 for both the net public and private opinion. These results can be interpreted as there probably being approximately as many individuals in favor of the ruling ideology as there are individuals against it. The system is mono-stable, because this behavior continues once it has been established.

The totalitarian system on the other hand shows a different behavior. Figure 4 shows that on the final time step the totalitarian system stabilizes into a state where the weakly opposing individuals may hold an private opinion that is in disagreement with the ruling ideology, but the strong pressure on the public opinion ensures obedience. This results in a public opinion that is mostly approving of the ruling ideology

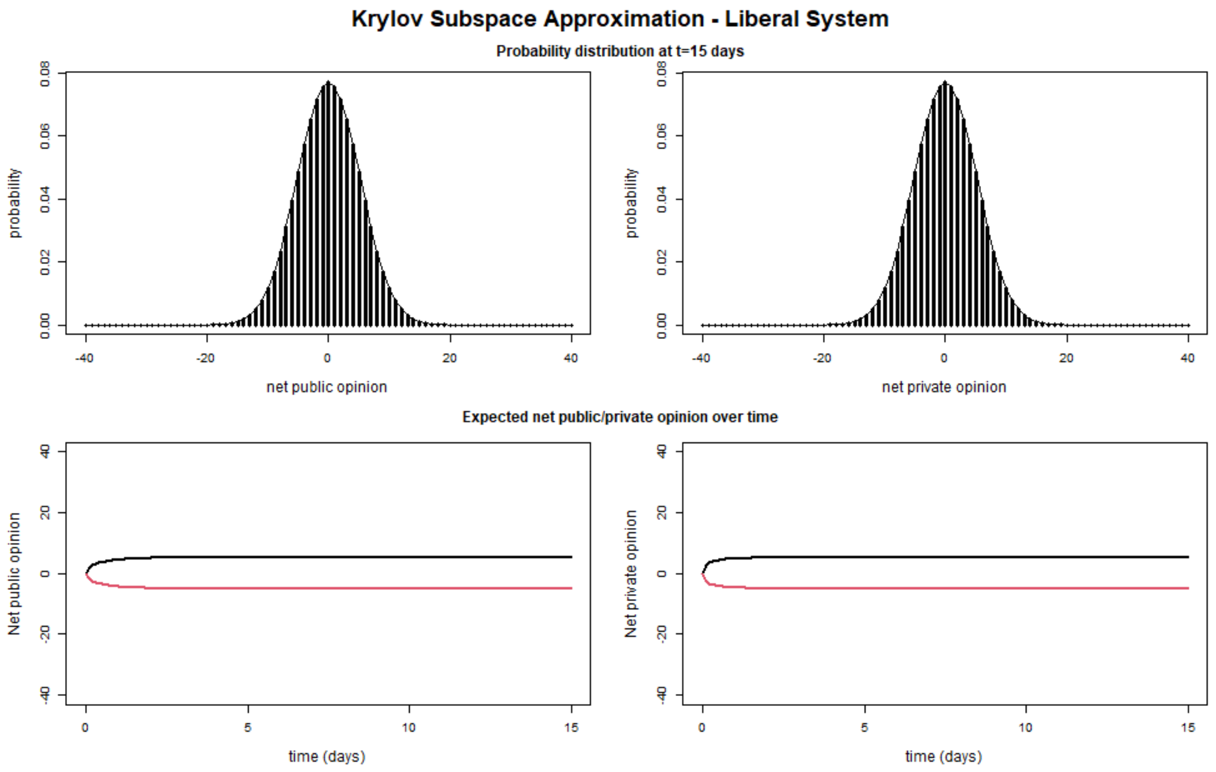

Figure 5: Results of KSA method for the liberal system. In the upper left/right plots the probability distribution is plotted against the net public/private opinion at $t = 15$ days. In the lower left/right plots the expected value plus and minus the standard deviation of the net public/private opinion is plotted over time. The black line is the mean plus the standard deviation and the red line is the mean minus the standard deviation.

and a private opinion is weakly disapproving of it, but not enough to overthrow the forced public opinion.

Even though the upper plots in Figure 3 and 4 give an overall intuition of the probability distribution, it is quite irregular. This could be smoothed out by increasing the number of Monte Carlo simulations, however, that would also increase the computational cost.

3.3 KSA

Similarly to the Monte Carlo simulations, the simulation time is set to 15, which is in days, with time step $dt = 0.1$. First, the matrix $\mathcal{P}$ is created, which holds the propensities of every possible value $-L \leq y_1, y_2 \leq L$. Then, Equation 7 will be solved where $\mathbf{p_0}$ is 0 everywhere except for $y_1 = 0$ and $y_2 = 0$, where it is 1. The simulation starts with $\mathbf{p}_{old} = \mathbf{p_0}$, then $\mathbf{p}_{new}$ is computed and saved, and finally $\mathbf{p}_{old}$ is set to $\mathbf{p}_{new}$. This is procedure is repeated for all the time steps. The total computation time for the liberal and totalitarian system simulations was approximately 40.75 and 38.64 seconds, respectively. Plots similar to Figures 3 and 4 were generated and the results are shown in Figure 5 and 6.

The probability distributions in the upper plots of Figures 5 and 6 are extremely smooth compared to probability distributions in Figures 3 and 4. In fact, it seems that Figures 5 and 6 give the results for Monte Carlo simulations where the number of simulations goes to infinity.

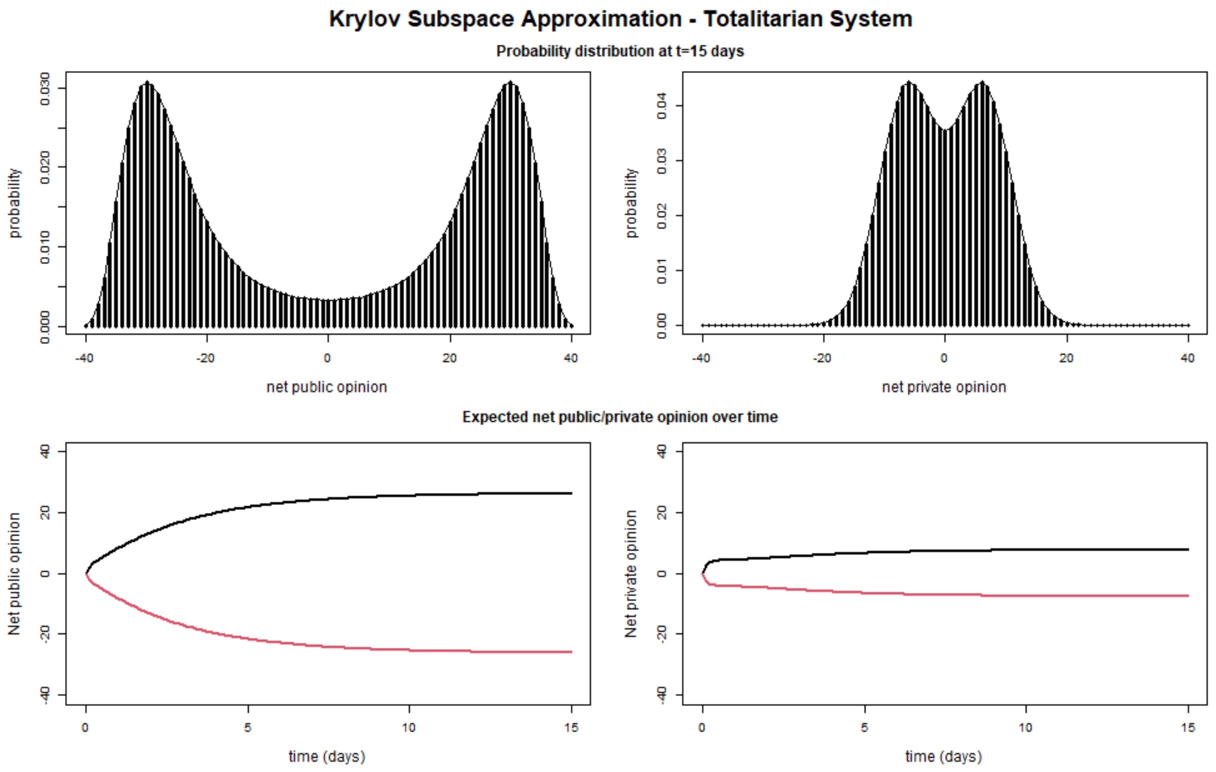

Figure 6: Results of KSA method for the totalitarian system. In the upper left/right plots the probability distribution is plotted against the net public/private opinion at $t = 15$ days. In the lower left/right plots the expected value plus and minus the standard deviation of the net public/private opinion is plotted over time. The black line is the mean plus the standard deviation and the red line is the mean minus the standard deviation.

4 Discussion

The results in Figures 3 and 4, immediately show the capability and shortcomings of the Gillespie algorithm with Monte Carlo simulations to model the dynamics of a reaction network. The method is robust and straightforward, hence easy to implement. Furthermore, the results quickly give intuition into the dynamics of the system and it is possible to achieve high accuracy by increasing the number of simulated trajectories. However, there are a few weaknesses. Increasing the number of trajectories comes at a high computational cost and it may be difficult to estimate beforehand how many simulated trajectories are necessary for convergence. Also, modelling more complex dynamics, such as higher order moments, gives rise to an increasingly larger numerical errors.

The KSA method, on the other hand, is more difficult to understand, hence also harder to implement. However, in the case of this case study, it yields much more accurate results than the Monte Carlo simulations, while its computational costs remains small. Nevertheless, it must be noted that the KSA method scales worse than the Monte Carlo simulations and it often cannot be used due to a the complexity of the underlying reaction network. In this case study, the number of possible states was small and constrained, but in the case that the number of possible states is large ,the KSA method is not suitable.

All in all, the KSA method is undoubtedly preferred over the Monte Carlo simulations for this case study. Mainly, because the results are significantly better, while the computational cost is approximately the same. The steeper learning curve of the KSA method does not weigh up against the increase in accuracy and we therefore deem it worthwhile.

5 Outlook

There a many more techniques for solving the master equation. Firstly, the Gillespie algorithm is not the only algorithm for sampling trajectories. In terms of Monte Carlo simulations there has been much work focused on improving computational efficiency whilst maintaining the same accuracy. Some examples are: *Langevin Approximation*, *Poisson Approximation*, *Maximum Entropy Approximation* [1]. Secondly, there has also been made an improvement on the KSA method, which is a technique known as *Implicit Euler* [6]. This technique is computationally superior over and more accurate than the KSA technique, but it can only be used in specific cases. Lastly, there are other, different techniques for numerically approximating the solution of the master equation. A few examples are *Moment Approximation* and *Linear Noise Approximation*.

The choice of which technique to use depends largely on the complexity of the underlying reaction network and the goal of the simulation. If the goal is to solve the master equation for a small constrained network, then the KSA method works really well. However, if the goal is to quickly gain some inside about the dynamics of the system, then Monte Carlo simulations may be more appropriate. If the goal of the simulation is more difficult or the complexity of the system is higher then more advanced techniques will be required and there will always be a trade-off between accuracy and computational efficiency.

References

[1] J. Goutsias and G. Jenkinson. Markovian dynamics on complex reaction networks. *Physics Reports*, 529(2):199–264, Aug 2013.

[2] Daniel T. Gillespie. The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

[3] B. Munsky and M. Khammash. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of chemical physics*, 124 4:044104, 2006.

[4] Mikhail A. Bochev. *A short guide to exponential Krylov subspace time integration for Maxwell's equations*. Number 1992 in Memorandum. University of Twente, Department of Applied Mathematics, September 2012.

[5] Expokit. https://www.maths.uq.edu.au/expokit/guide.html. Accessed: 2021-04-12.

[6] Garrett Jenkinson and John Goutsias. Numerical integration of the master equation in some models of stochastic epidemiology. *PloS one*, 7:e36160, 05 2012.