The Gillespie SSA applied to Queue Theory

MATTHEW ROBERT JAMES WARREN

1 INTRODUCTION

Stochastic systems occur throughout the physical sciences and form a vital part of probability theory. It is often necessary to numerically simulate systems in the physical science as analytic methods are either intractable or impossible. However, many systems at a fundamental level are inherently random (such as the famous Brownian Motion) thus deterministically and continuously simulating them is an inappropriate method which does not respect the system's intrinsic randomness. The realisation that deterministic and continuous simulations are not always appropriate has resulted in the development of many stochastic simulation algorithms which endeavour to mirror the random and discrete nature of many natural processes [1].

This project explores one of the first proposed methods of stochastic simulation, the Gillespie Stochastic Simulation Algorithm (SSA) [2]. Published in 1976 by Daniel Gillespie the Gillespie SSA method was published to deal with chemical reactions and much of the application of the method has been to chemical reactions and biological systems of many reactants or components.

1.1 Theoretical Groundwork

The deterministic, older methods of mathematical chemistry treat qualify chemical reactions by their rate of occurrence - according to [3, 4] - however, as this report is not focussed on chemistry this shall not be delved into too deeply. While the stochastic approach, unsurprisingly, focusses on a probability, reactions are quantified by their Stochastic Reaction Constant c_μ which differs for each specific reaction μ that is occurring in the system.

The Stochastic Reaction Constant (multiplied by a infinitesimal time interval dt) is defined to be the "average probability that a particular combination of reactants react according to a reaction μ in the next infinitesimal time interval dt". Gillespie then goes on to define a probability density function (pdf) $P(\tau,\mu)$ such that:

$$
\begin{aligned}
P(\tau,\mu)d\tau := \text{probability that a reaction of type } \mu \\
\text{occurs in the time interval } [t+\tau, t+\tau+d\tau]
\end{aligned}
$$

for a given state of reactant populations at time t.

The stochastic formulation revolves around the quantifying of different reactions by which the system evolves. Each reaction μ depends on the populations of the reactants in the system. For example, consider the dummy reaction:

$$X_1 + X_2 \longrightarrow X_3$$

Obviously if there are no X_1 reactant species in the mixture then these reactions cannot happen, or if there are a far greater number of one of the species then the reaction rate will be limited on this number. So for the above reaction we are interested in the amounts of X_1 and X_2 in the system. From this point on consider X_i to represent the amount of reactant i present in the system. Then the following important quantity h_μ is defined to be: h_μ := number of distinct reactant combinations available for a reaction μ; so for the above reaction with species 1 and species 2: $h_{dummy} = X_1X_2$ (1) . This allows for another important quantity - the propensity - to be defined: The propensity of a reaction μ is a_μ where $a_\mu dt = h_\mu c_\mu dt$, which in qualitative terms is the probability that a reaction μ will occur given the state $\{X_i\}$ of the system at a time t.

The explicit details of how the pdf $P(\tau,\mu)$ is derived are not required in order to appreciate the SSA algorithm, thus the pdf is simply given as:

$$
P(\tau,\mu) = \begin{cases} a_\mu e^{-a_0\tau} & if\, 0 \leq \tau < \infty \;\&\; \mu = 1, ..., M \\ 0 & \text{otherwise} \end{cases}
$$

In order to apply the algorithm the computer must randomly generate the values μ and τ. For any interesting system worth simulating there is likely to be several different reactions that can occur, consider there to be M reactions where $\mathcal{R}_\mu$ represents reaction μ and $\mu = 0, 1, 2, ..., M$. In the pdf above a_μ is the already introduced propensity while a_0 is the sum of each propensity, as each $\mathcal{R}_\mu$ has its own propensity, hence $a_0 = \sum_\mu^M a_\mu$.

Showing that the pdf P depends on all the reactions for any given μ.

In order to generate the values of μ and τ the computer cannot just randomly choose them as this would make the construction of the pdf P pointless, they must be randomly chosen according to P. In order to do this - this is given without proof - first randomly generate two numbers from the uniform interval:

$$r_1, r_2 \in [0, 1] \tag{2}$$

Then according to the two following equations the reaction parameters τ, μ are produced:

$$\tau = \frac{1}{a_0} \ln \left(\frac{1}{r_1} \right) \tag{3}$$

$$\mu \in \mathbb{N} \; s.t. \sum_{\nu=1}^{\mu-1} a_\nu < r_2 a_0 \le \sum_{\nu=1}^{\mu} a_\nu \tag{4}$$

Following the construction as in [3, 4] the actual algorithm can be introduced as all the theoretical groundwork has been now laid.

1.2 The SSA

The Stochastic Simulation Algorithm as according to Gillespie's work can now be outlined in a few steps:

1.2.1 *Step 1 - Initiation.*

- Determine the M reactions that define the system and their respective Stochastic Reaction Constants $\{c_\mu\}$.
- Specify the initial quantities of each of the N reactants present in the system - the starting values for $\{X_i\}$ (where $i = 0, 1, ..., N$).
- Initialise the time elapsed as t to $t = 0$ and the reaction counter to g to $g = 0$.

1.2.2 *Step 2 - Initial Calculations of State.*

- Using the current population values contained in $\{X_i\}$ determine h_μ (as defined in 5) for each $\mathcal{R}_\mu$. Then calculate the propensity a_μ of each $\mathcal{R}_\mu$ for this particular transition/reaction g.
- Calculate a_0 as the sum of the individual propensities.

1.2.3 *Step 3 - Random Generation.*

- Use a random number generator to randomly generate the r_1, r_2 values from the uniform unit interval.
- Combine the random numbers r_1 and r_2 with the quantities calculated in the previous step to ascertain this generations τ and μ via equations (5) and (6).

1.2.4 *Step 4 - Apply the Reaction.*

- Using the parameters found in the previous step increase the time elapsed so that $t_{new} = t_{current} + \tau$ and apply $\mathcal{R}_\mu$ according to the generated μ.
- Change the population levels $\{X_i\}$ according to the instructions of $\mathcal{R}$, - the required change should be evident when the SSA is applied to a given context.
- Increase the reaction counter g by 1.

After running through step 1 to step 4 once repeat steps 2 to 4 as many times as desired/required.

It is also useful to define a state-change vector ν_μ for each $\mathcal{R}_\mu$. The state-change vector ν_μ encodes the effect of the reaction $\mathcal{R}_\mu$ on the population set $\{X_i\}$ allowing for an easy update of the system. Consider $\mathbf{x}$ to be the vector of initial reactant populations (representing $\{X_i\}$) at the start of a given step g then after a certain reaction $\mathcal{R}_\delta$ has been chosen by the SSA (step 3) the reactant populations become $\mathbf{x} + \nu_\delta$. The state-change vector together with the propensity functions completely define a reaction.

1.3 Exactness

While the SSA is a simulation it is an exact simulation and does not make any approximations. The elements of randomness that are present in the algorithm are reflective of the true nature of the processes in the system. The random parts of the algorithm are the selection of the parameter τ and μ, the rest of the algorithm is determined by the form of the reactions $\mathcal{R}_\mu$ which have no probabilistic element and are key to defining the system. Hence only the choices of τ and μ need to be justified in order to claim exactness.

Throughout this process it has been assumed that the system of interest is Markovian and hence the probabilities of the system attaining a certain state only depend on the state before. The Markovian assumption is reflected in the formulation of the SSA as only the populations of the previous state are carried over into the next state and there is no memory bank of further back populations or reactions that is used to influence the choices of the parameters. The probability density function P did not make any restrictive assumptions about the nature of the system and although the derivation by Gillespie was phrased in the context of chemical reactions this has not been imparted onto the pdf in any meaningful way. Thus the pdf P is very general.

Equation (3) is produced from the pdf P by a method known as the "Inversion Generating Method" which is a Monte Carlo technique and beyond the scope of this report[4]. It is sufficient to say that the Monte Carlo Inversion Generating Method is well known and reliable meaning that Equation (3) is entirely trustworthy. Therefore the method used to generate the parameters τ and μ represents and replicates the Markovian nature of the system. For example, in a system with inter-arrival times governed by an Exponential Distribution - which is inherently stochastic and therefore random - then the simulated parameters τ and μ are accurately sampled from such a distribution. As the system is inherently random there are no correct values for τ and μ to take but they are sampled in an exact way which does not make distributional assumptions outside of those already assumed by the system. Thus the Gillespie SSA is considered to be an exact method of simulation.

2 EXAMPLE

2.1 Formulation

To view the Gillespie SSA algorithm in action and to convince oneself of its efficacy consider a ***M/M/3/5*** queue which is used as a model of a coffee shop. This is a memoryless/Markovian system with an assumed Poisson distribution dictating the arrival and departure rates of queuing members - which are taken to be independent.

For the ***M/M/3/5*** queue the state transition diagram looks like:

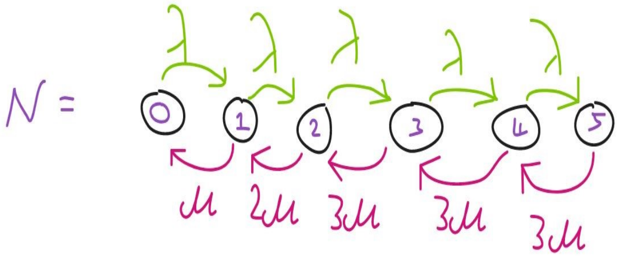


Fig. 1. State Transition Diagram

The state that the system is in determines the population of the queue, so the value of N in the queue state transition diagram that the queue is in at any given time is the state of the system. Hence each state can either be occupied or unoccupied and only one state can be occupied at any given time. For example the system being in state 4 means that there are four people in the queue, three of which will be being served and one shall be waiting to be served.

Defining $\mathbb{S}$ to be the set of possible queue/system states then $\mathbb{S} = \{0, 1, 2, 3, 4, 5\}$, then if the system is in state i for a certain state $i \in \mathbb{S}$ the population of the states $\mathbf{x}$ is given by:

$$\mathbf{x}_j = \begin{cases} 1, & j = i \qquad (5) \\ 0, & j \neq i \qquad (6) \end{cases}$$

for $i, j \in \mathbb{S}$

By substituting the values of $\lambda = 1$ per minute and $\mu = \frac{1}{2}$ per minute into the above state transition diagram the following rate matrix $\mathbf{\Lambda}$ can be deduced:

$$\mathbf{\Lambda} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{3}{2} & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{3}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{3}{2} & 0 \end{pmatrix} \qquad (7)$$

With i, j forming the rows and columns respectively so the ith,jth entry represents the transition from state i to state j with $i, j = 0, 1, 2, 3, 4, 5$. The diagonals are zero because a state cannot transition to itself, and all non-adjacent entries are zero because states can only transition to their neighbouring states due to the formulation of Poisson based queues. So, for example the value of $\mathbf{\Lambda}_{12}$ (If the first row of the matrix is counted as row 0 and similarly for the columns) is 1 and denotes a rate of 1 per minute for the queue to progress from state 2 to state 3 - as reflected by the state transition diagram.

In the Gillespie SSA paradigm transitions between states are caused by reactions, each arrow in the state diagram represents a reaction, hence there are ten reactions in this system. These reactions are:

- $\mathcal{R}_0$:= State 0 ⟶ State 1
- $\mathcal{R}_1$:= State 1 ⟶ State 0
- $\mathcal{R}_2$:= State 1 ⟶ State 2
- $\mathcal{R}_3$:= State 2 ⟶ State 1
- $\mathcal{R}_4$:= State 2 ⟶ State 3
- $\mathcal{R}_5$:= State 3 ⟶ State 2
- $\mathcal{R}_6$:= State 3 ⟶ State 4
- $\mathcal{R}_7$:= State 4 ⟶ State 3
- $\mathcal{R}_8$:= State 4 ⟶ State 5
- $\mathcal{R}_9$:= State 5 ⟶ State 4

To mathematically characterise the above reactions the state-change vector ν and the propensity function $a(\mathbf{x})$ need to be defined for each reaction.

For illustrative purposes consider the first reaction $\mathcal{R}_0$, the state-change vector ν_0 for this reaction is a six dimensional column vector given by:

$$\nu_0 = \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{8}$$

Each row in ν_0 represents a state of the queue system, and as $\mathcal{R}_0$ moves the system from state 0 to state 1 it follows that its corresponding state-change vector ν_0 reduces the population of state 0 by one (the only allowed value by formulation) and increases the population of state 1 by one. Computing the remaining state-change vectors similarly and combining them into one matrix denoted as $\boldsymbol{\nu}$ which has ten columns corresponding to the ten reactions, and six columns. The full state-change matrix is:

$$\boldsymbol{\nu} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \tag{9}$$

For each reaction there is also the propensity function which for this system is fairly simple. The first piece of information necessary is the scalar propensity value. These scalar values can be determined by considering the rate matrix $\mathbf{\Lambda}$, to get the propensity value for the first reaction $\mathcal{R}_0$ find the value in the rate matrix which gives the rate at which the system moves from state 0 to state 1, this is the value of $\mathbf{\Lambda}_{01}$ which is 1.

The second piece of information required to build the propensity function is the population of the reaction's initial state, so for $\mathcal{R}_0$ the state of interest is state 0. In this system the states can only have a population of 0 or 1 and if one state has population 1 the other states must all have a population of 0 as shown in (5). Therefore, for reaction $\mathcal{R}_0$ the population of interest is $\mathbf{x}_0$. Multiplying $\mathbf{x}_0$ by $\mathbf{\Lambda}_{01}$ gives the propensity function for $\mathcal{R}_0$ so $\boldsymbol{a}_0 = \mathbf{\Lambda}_{01} \cdot \mathbf{x}_0 = \mathbf{x}_0$.

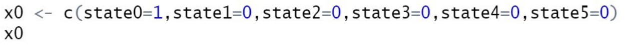
The other propensity functions can be found in a similar way and the full propensity function vector is a ten dimensional row vector with each column giving the propensity value of the corresponding reaction. The propensity vector $\boldsymbol{a}$ is hence

$$\boldsymbol{a} = \Big(1 \cdot \mathbf{x}_0,\ \frac{1}{2} \cdot \mathbf{x}_1,\ 1 \cdot \mathbf{x}_1,\ 1 \cdot \mathbf{x}_2,\ 1 \cdot \mathbf{x}_2,\ \frac{3}{2} \cdot \mathbf{x}_3,\ 1 \cdot \mathbf{x}_3,\ \frac{3}{2} \cdot \mathbf{x}_4,\ 1 \cdot \mathbf{x}_4,\ \frac{3}{2} \cdot \mathbf{x}_5\Big)$$

The $\boldsymbol{\nu}$ state-change matrix and $\boldsymbol{a}$ propensity vector completely specify the reactions/transitions of the queue system. Clearly, at any given time the majority of the propensity vector is zero and at most there can only be two non-zero entries in the vector, reflecting the neighbouring transitions only nature of the queue system.

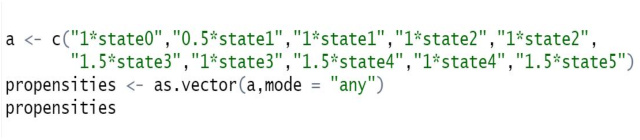
2.2 R Simulation

Now that the reactions have been completely specified the queue can be simulated using **R** and the **GillespieSSA** package which provides a simple way to use the Gillespie SSA algorithm. The difficult part of the simulation - namely formulating the $\boldsymbol{\nu}$ state-change matrix and $\boldsymbol{a}$ propensity vector - has been completed and they just needed to be entered into **R**. The last step before running the simulation is to define an initial state vector, assuming that when the coffee shop opens there is nobody already waiting then the queue can begin in state 0. The initial vector and reaction vectors are shown in the following screenshots from **R**:



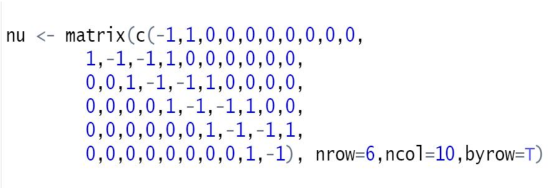
```
x0 <- c(state0=1,state1=0,state2=0,state3=0,state4=0,state5=0)
x0
```

Fig. 2. Initial State Vector in R



```
a <- c("1*state0","0.5*state1","1*state1","1*state2","1*state2",
       "1.5*state3","1*state3","1.5*state4","1*state4","1.5*state5")
propensities <- as.vector(a,mode = "any")
propensities
```

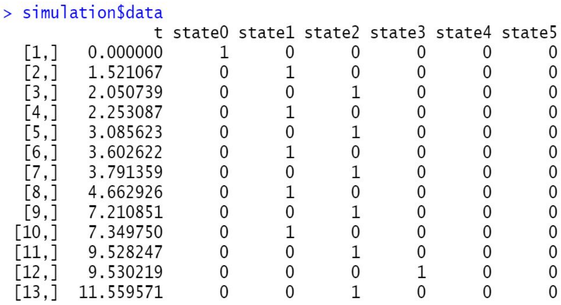
Fig. 3. Propensity Vector in R



```
nu <- matrix(c(-1,1,0,0,0,0,0,0,0,0,
        1,-1,-1,1,0,0,0,0,0,0,
        0,0,1,-1,-1,1,0,0,0,0,
        0,0,0,0,1,-1,-1,1,0,0,
        0,0,0,0,0,0,1,-1,-1,1,
        0,0,0,0,0,0,0,0,1,-1), nrow=6,ncol=10,byrow=T)
```

Fig. 4. State Transition Matrix in R

These vectors can be used as the arguments to the **ssa** function which will simulate the coffee shop queue for as long a time period as is specified. Using the **ssa** function to simulate the queue for 100 minutes produces output data which should look similar to:



```
> simulation$data
                t state0 state1 state2 state3 state4 state5
 [1,]   0.000000      1      0      0      0      0      0
 [2,]   1.521067      0      1      0      0      0      0
 [3,]   2.050739      0      0      1      0      0      0
 [4,]   2.253087      0      1      0      0      0      0
 [5,]   3.085623      0      0      1      0      0      0
 [6,]   3.602622      0      1      0      0      0      0
 [7,]   3.791359      0      0      1      0      0      0
 [8,]   4.662926      0      1      0      0      0      0
 [9,]   7.210851      0      0      1      0      0      0
[10,]   7.349750      0      1      0      0      0      0
[11,]   9.528247      0      0      1      0      0      0
[12,]   9.530219      0      0      0      1      0      0
[13,]  11.559571      0      0      1      0      0      0
```

Fig. 5. Simulated Queue Data

The t column represents the elapsed time since the start of the simulation in terms of simulation units - not real elapsed time. The λ and μ rates were defined in terms of minutes so so too are the t values in the t column. Selecting row 5 at random to explore, this is the fourth transition since the queue began (as row 1 represents the initial state) and this transition takes the queue from state 1 to state 2. This transition represents someone joining the queue, but as there are fewer than three people in the system this new customer shall be served straight away. Row 5 shows that the fourth transition occurred roughly 3.09 minutes after the coffee shop opened.

Row 5 is not particularly special or interesting and in order to understand the queue consider the following diagrams:

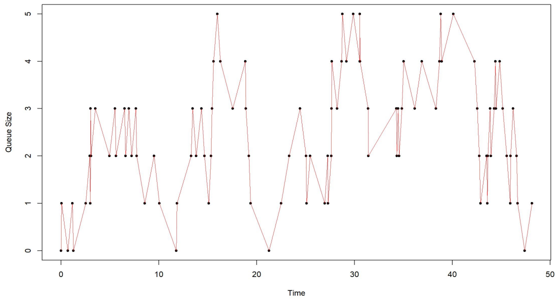


Fig. 6. Queue Size Against Time Steps

Figure 6 displays the first 100 transitions of the simulation and plots the state the system is in against the time since the coffee shop opened in minutes. Transitions are assumed to happen instantaneously so the length of time the system inhabits a state for is given by the horizontal length from that state to the next (right-hand) state.

As can be seen from Figure 6 the queue is fairly random - as to be expected - and this is shown both in the lack of discernible patterns and the width of the time steps. The most obvious evidence of the randomness in the time steps comes from the transitions $i = 1 \longrightarrow i = 0 \longrightarrow i = 1$ around the $t = 20$ minutes mark, which is a very wide transition compared to, say, the transitions between $t = 40 - 50$ minute interval. This is encouraging as it reflects the underlying Exponential distribution in the times between transitions which should be followed by the ***M/M/3/5*** queue. The lack of pattern in Figure 6 is also desirable and reflects the stochastic nature of the system.

From Figure 6, the system appears to spend the majority of its time in the middle states ($i = 1, 2, 3, 4 \ : \ i \in \mathbb{S}$), with only six instances of the system being in the $i = 1$ state and six instances again for the $i = 5$ state. Of course the exact number of occurrences for each state and when these occurrences occur will differ each type the simulation is ran, however, the general pattern of "avoiding" the outer states should be consistent between simulations. The clustering of the system around the central states is fairly expected too and comes from the Poisson Rate Equations that govern the system.

The histogram of Figure 7 shows the frequency density of time intervals between transitions for the first 100 transitions of the simulation.

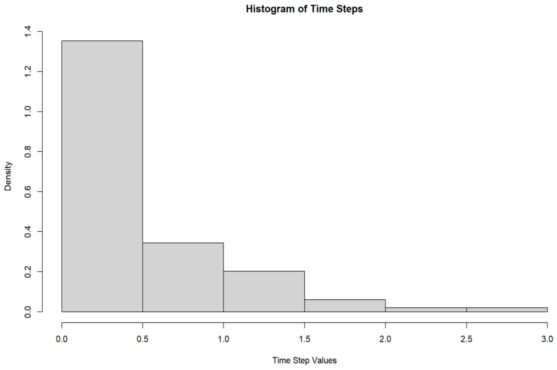


Fig. 7. Histogram of Time Step Sizes

The time intervals are found easily using **R** and the histogram displays the expected behaviour. This shape is expected due to the time intervals between transitions being governed by the Exponential Distribution, given by :

$$f(x, \alpha) = \begin{cases} \alpha e^{-\alpha x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Crudely taking the rate α which governs the distributions of time intervals to be the reciprocal of the mean of the first 100 time step transitions and plotting the resulting Exponential Distribution over the above histogram produces:

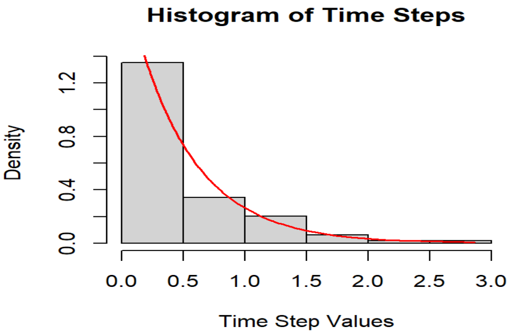


Fig. 8. Histogram of Time Step Sizes with Exponential Line

The Exponential Distribution curve - for the most part - convincingly passes through the histogram as desired.

There are seemingly very few transitions which occur a whole two minutes after the previous one, corresponding to the last two bins of the histogram. In order to quantify this value much longer simulations were ran. Ten simulations each running up to 100,000 simulation time units (minutes in the coffee shop context) were ran and the times t extracted from each simulation. For each of the 10 very long vectors of times - roughly 180,000 entries in each - the transition times were calculated and the number of transition times greater than 2 (minutes) were calculated. The probability of a transition taking longer than 2 minutes is then simply the previously calculated number divided by the number of total transitions. Once all ten probabilities had been worked out, the mean average was taken. This is all documented in the attached **R** code. The resulting value was found to be 0.030.

$$P(\text{Time between transitions} > 2\,\text{mins}) = 0.030 = 0.030\%$$

As mentioned the **R** work was relatively simple once the ten reactions of the system had been defined. However, this method of simulating the queue would struggle to deal with infinite capacity queues and queues with a large maximal capacity would be time-consuming with respect to defining the state-change matrix and propensity vectors. It is likely that using each state to represent the population for a large capacity queue would be an unwieldy method, despite its intuitiveness for this small queue.

3 CONCLUSION

The example demonstrated the efficacy of the SSA, once one becomes familiar with the state-change vectors and propensity function the SSA becomes intuitive. The **R** code required to implement the simulation is very straightforward. The SSA certainly has three main positives: its simplicity once the theoretical background has been understood; its generality - it can applied to many classes of stochastic problem, and whilst it was formulated to describe chemical reactions it has been successfully applied to queue theory in this project and much further afield in proper research papers; It is also an exact method resulting in highly reliable results which many other methods are not capable of replicating.

The exactness of the SSA does not imply that each SSA simulation will recreate the same state transitions - the SSA is still random but it exactly represents the stochastic nature of the system it is simulating. This exactness comes at a cost however, as it means each time step/transition in the simulation must be performed sequentially. For long simulations which require many many transitions the exact method of the SSA becomes increasingly slow [5]. There are methods built off the "direct" SSA which bundle together several transitions between states. The number of transitions which can be bundled together is limited by the Leap Condition which prevents any statistically significant change happening to the propensity functions. These bundled SSA methods forego the exactness of the direct SSA for faster computational time. The precise details of the bundled SSAs is not important but they highlight that the SSA is a very diverse and popular method for simulating stochastic processes.

REFERENCES

[1] Dedicated Tools for Stochastic Simulation, Chapter 12 - Software Tools for Systems Biology, Systems Biomedicine Concepts and Perspectives, 2010, Pages 289-314, Herbert M.SauroFrank T.Bergmann, https://www.sciencedirect.com/science/article/pii/B9780123725509000122 Methods

[2] Stochastic simulation, Accessed 21/04/2022, https://en.wikipedia.org/wiki/Stochastic_simulation#:~:text=A%20stochastic%20simulation%20is%20a,a%20model%20of%20the%20system.

[3] Exact Stochastic Simulation of Coupled Chemical Reactions,Daniel T. Gillespie, May 12 1977, https://vle.exeter.ac.uk/pluginfile.php/3088529/mod_resource/content/1/gillespie_orig.pdf

[4] A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions, Daniel T. Gillespie, JOURNAL OF COMPUTATIONAL PHYSICS 2, 403-434 (1976), https://www.sciencedirect.com/science/article/pii/0021999176900413

[5] GillespieSSA: Implementing the Stochastic Simulation Algorithm in R, Mario Pineda-Krch, University of California, Journal of Statistical Software, April 2008, Volume 25, Issue 12. https://vle.exeter.ac.uk/pluginfile.php/3088530/mod_resource/content/1/v25i12.pdf