Stochastic Simulation of Supply Chains

Sohail Suleman Zaveri
201681438
Supervised by Dr. Nadhir Ben Rached (Department of Mathematics)
and Hamish Steptoe (Met Office)

Submitted in accordance with the requirements for the module MATH5872M:
Dissertation in Data Science and Analytics
as part of the degree of

Master of Science in Data Science and Analytics

The University of Leeds, School of Mathematics
August 2023

1 Introduction to Supply Chains

A *supply chain* [4] is a network of all the parties involved in production, transportation, storage, distribution and delivery of products in order to fulfil a customer's request. Other than manufacturers and suppliers, a supply chain also includes warehouses, transporters, distributors, retailers and the customers themselves. A supply chain also includes the activities and functions that are aimed towards fulfilling a customer's demands. These functions include distribution, finance, quality assurance, information exchange and transportation.

We present a simple example of a supply chain that describes the functions of various stages when a customer buys a smartphone from a Walmart store.

Let us consider a customer purchasing a smartphone from Walmart. The supply chain that kicks-in to fulfil the customer's request includes - the customer, Walmart's store staff, Walmart's distribution centers, smartphone manufacturer, their suppliers, transportation partners and third-party organisations. The supply chains starts with the customer visiting Walmart to buy a smartphone. Walmart's store staff stocks its shelves from the store's inventory, that in turn is supplied from Walmart's distribution centers. Supplying goods requires involvement of transportation vehicles that might have been supplied from a third party. The distribution centers are stocked by supplies from smartphone manufacturers, that themselves use supplies from lower-tier suppliers. For example, the manufactures might depend on a supplier for its semiconductor demands and another supplier for its packaging needs. This network of stages involves information flow amongst parties such as - i) point-of-sales information about the purchase that is registered at Walmart store, ii) data on available stocks in the store's inventory that is communicated with Walmart's distribution centers, iii) sales information and additional orders of smartphones placed by Walmart to smartphone manufacturers, just to name a few.

Several stages that are involved in a typical supply chain, as described in [4], include the customers, retailers, distributors, manufacturers and raw material suppliers.

A manufacturer usually depends on more than one suppliers, and similarly other stages of a supply chain depend on more than one parties for proper functioning of the supply chain. Figure 1 depicts a supply chain with eight processes, where process 3 depends on supplies from supply 1 and process 1. Similarly, process 5 depends on output materials from processes 3 and 4.

1.1 Risk and disruption in supply chains

Supply chains across the globes faced one of the biggest disruptions in the form of COVID-19 pandemic in 2020-2021. Supply chains at local and global levels were effected drastically due to the near-total reduction in public mobility and ocean freight. Reduction in workforce at manufacturing plants, ports and road transportation posed challenges to supply chains at almost all levels. Many

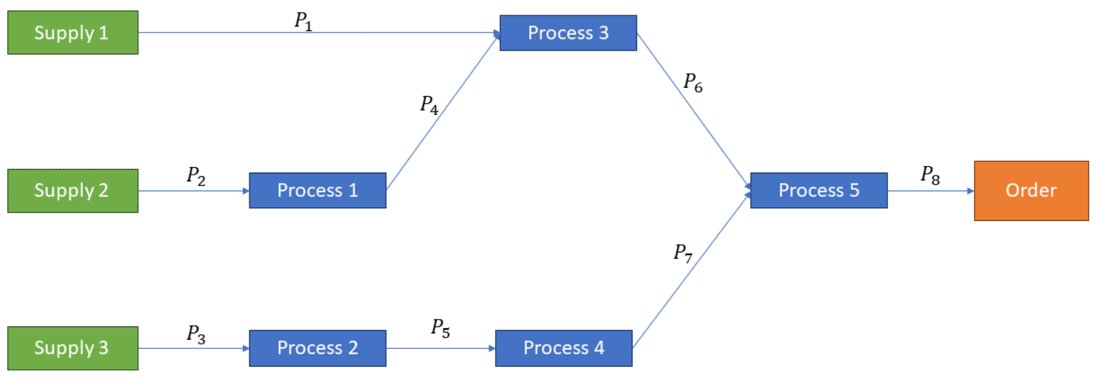


Figure 1: A supply chain with 3 supplies and 5 processes.

countries still face economic challenges as an aftermath of disrupted and restructured supply chains. Given the far-reaching impact of the COVID-19 pandemic, which led to substantial reductions in public mobility and disrupted various aspects of supply chains, it becomes crucial to discuss the risks and underlying causes of such disruptions during the design and implementation of supply chains.

With rising global temperatures, extreme weather events are posing higher and higher risks to the smooth functioning of global supply chains. High level of inter-dependence between countries for raw materials and manufactured products implies that even a single disruption caused by extreme weather can hamper global supply chains, and hence lead to serious economic consequences.

As global supply chains become increasingly complex, they are susceptible to multiple points of failure, rendering them more vulnerable and less resilient. Some of the risk factors, as discussed in [4], are categorised in table 1:

1.2 Resilience of supply chains

With good network designs, supply chains can mitigate many risk factors mentioned in Table 1. According to D. Mavatoor in [5], “The concept of resilience shouldn’t assume that you won’t fail, but rather that you should be able to get back up fast.” Flexibility of networks, amongst others, plays a crucial role in ensuring resilience of supply chains. An example of the effectiveness of flexible networks is the levels of impact a supplier disruption had on two companies, Nokia and Ericsson

Category	**Risk Drivers**
Delays	High capacity utilization at supply source, inflexibility of supply source, poor quality or yield at supply source
Disruptions	Pandemics, war, natural disaster, supplier bankruptcy, labor disputes
Systems Risk	Information infrastructure breakdown, system integration or extent of systems being networked
Forecast Risk	Inaccurate forecasts due to small customer base, seasonality, product variety, information distortion
Procurement Risk	Exchange-rate risk, price of inputs, fraction purchased from a single source, industry-wide capacity utilization
Receivables Risk	Number of customers, financial strength of customers
Inventory Risk	Rate of product obsolescence, inventory holding cost, product value, demand and supply uncertainty
Capacity Risk	Cost of capacity, capacity flexibility

Table 1: Categorization of risk factors in supply chains

in 2000. Supply chains of Nokia and Ericsson were disrupted when a plant owned by Royal Philips Electronics caught fire in Albuquerque, New Mexico in 2000. Nokia bounced back quickly because of presence of multiple backups in their network of suppliers, and faced minimal losses. On the other hand, Ericsson lost $400 million in revenues as it did not have a backup in their network, and hence could not bounce back fast.

Mitigation strategies come at a price and their own set of risks. Having multiple suppliers in the network increases cost of the supply chain due to increase in transportation, coordination, management and used resources. Having higher inventory incurs cost on storage, maintenance and raises risk of obsolescence. As discussed in [18], the trade-off in inventory control is between *producing*, which leads to increased inventory costs and management costs, and *idling*, which can lead to stock-outs and unsatisfied demands. Hence, customised mitigation strategies should be incorporated in designs of supply chains that achieve a trade-off between the risk mitigated and the incurred cost. Some of the tailoring strategies, as discussed in [4] are mentioned in table 2.

1.3 Types of processes in supply chains

The processes in supply chains can be categorised based on whether they operate in response to a customer's order, or in anticipation of customer orders [4]. *Push processes* are executed in anticipation of customer orders, whereas *pull processes* are triggered following a customer's order.

1.3.1 Push Processes

Push processes are executed in anticipation of customer orders. They are based on forecasts of product demands in the future, and hence, are also referred to as *speculative processes* because they

Risk Mitigation Strategy	**Tailored Strategies**
Increase capacity	Focus on low-cost, decentralized capacity for predictable demand. Build centralized capacity for unpredictable demand. Increase decentralization as the cost of capacity drops.
Get redundant suppliers	More redundant supply for high-volume products, less redundancy for low-volume products. Centralize redundancy for low-volume products in a few flexible suppliers.
Increase responsiveness	Favor cost over responsiveness for commodity products. Favor responsiveness over cost for short-life cycle products.
Increase inventory	Decentralize inventory of predictable, lower value products. Centralize inventory of less predictable, higher value products.
Increase flexibility	Favor cost over flexibility for predictable, high-volume products. Favor flexibility for unpredictable, low-volume products. Centralize flexibility in a few locations if it is expensive.
Pool or aggregate demand	Increase aggregation as unpredictability grows.
Increase source capability	Prefer capability over cost for high-value, high-risk products. Favor cost over capability for low-value commodity products. Centralize high capability in a flexible source if possible.

Table 2: Tailored strategies for risk mitigation

operate in response to speculated demand. An example of a push process is a bakery producing a variety of bread and pastries early in the morning and pushing them onto store shelves for customers to purchase. Upstream processes, such as production pf raw materials typically use push models based on forecasts of company demands.

1.3.2 Pull Processes

Pull processes are initiated in response to a customer order. These are also known are *reactive processes* as they react to an actual demand of a product. An example of a pull process is a company manufacturing products only when a customer order is received, ensuring that production is triggered by actual demand. Processes closer to end-customers, such as retailers and e-commercial websites use pull model as they operate only when a customer order is placed.

1.3.3 Hybrid Systems

In a hybrid system, some processes operate on a push model, whereas other processes operate on a pull model. Using hybrid systems enable companies to tailor their strategies to meet their limitations and maximise customer satisfaction at the same time. Hybrid systems offer the potential to optimise inventory levels, enhance customer satisfaction, reduce lead times, and improve overall supply chain performance.

2 Chemical Reaction Modelling

Continuous-time Markov chains (CTMCs) are stochastic processes that follow the Markov property in which the system makes transitions in continuous time. Unlike discrete-time Markov chains where transitions can only take place at discrete time steps, CTMCs can transition through states at any point in time.

Stochastic Reaction Networks (SRNs) are a class of CTMCs that are used to model evolution of chemical systems where molecules of different chemical species can undergo a finite set of reactions [17]. SRNs are particularly useful when dealing with systems with small number of molecules as it helps in incorporating the inherent randomness of phase space of molecules in the evolution of the system.

In this chapter, we define CTMCs, SRNs, and discuss algorithms such as *Stochastic Simulation Algorithm* (SSA), *Next Reaction Method* (NRM), *Modified Next Reaction Method* (MNRM) and *tau-leap method*, that can be used to simulate SRN trajectories. SSA, NRM and MNRM are considered to be exact as the paths simulated by these methods follow correct statistical distributions. Tau-leap methods are used to simulate SRN paths using discrete time steps, in order to address some of the computational and performance related shortcomings of exact methods.

2.1 Continuous Time Markov Chains

Let $\mathbf{X} = \{X(t), t \in [0, \infty)\}$ be a continuous-time stochastic process with a countable state space $\mathbf{S}$. $\mathbf{X}$ is called a continuous-time Markov chain [20] if for any set of time indices $0 \leq t_1 < t_2 < \ldots < t_{s-1} < t_s < t$ and corresponding states $c_1, c_2, \ldots, c_{s-1}, c_s, c_t \in \mathbf{S}$ such that $P(X(t_1) = c_1, X(t_2) = c_2, \ldots, X(t_s) = c_s) > 0$, the following is true:

$$P(X(t) = c_t | X(t_1) = c_1, \ldots, X(t_s) = c_s) = P(X(t) = c_t | X(t_s) = c_s). \tag{1}$$

The property (1) is also called the Markov Property. We see from (1) that the probability of X being in a state c_t at time t is only conditioned on the state $X(t_s)$ at time index t_s, and does not depend on the states of the process in time indices $t_1, t_2, \cdots, t_{s-1}$.

2.2 Stochastic Reaction Networks

An SRN $\mathbf{X}$ is a type of CTMC that describes evolution of a homogeneous chemical mixture of molecules in continuous time.

Given an underlying probability space $(\Omega, \mathcal{F}, P)$, $X : [0, T] \times \Omega \to \mathbb{Z}^N_{\geq 0}$ describes the time-evolution of a system of molecules belonging to N different species $(S_1, S_2, \ldots, S_N)$ in continuous time $t \in [0, T]$.

Elements in $X(t) = (x_1(t), x_1(t), \ldots, x_N(t))$ indicate the abundances of the N species of molecules at time t. The molecules can undergo a finite set of reactions $\mathbf{R} = (\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_M)$, where a reaction $\mathcal{R}_j \in \mathbf{R}$ consists of a pair (ν_j, a_j). Each element in $\nu_j = (\nu_{j,1}, \nu_{j,2}, \ldots, \nu_{j,N}), 1 \leq j \leq, M$ corresponds to the change in the number of molecules of that particular species when the j^th reaction fires in the mixture of molecules. The vectors ν_j is also known as *stoichiometric vectors*. The *propensity* $a_j : \mathbb{Z}^N_{\geq 0} \to \mathbb{R}_{\geq 0}$ of a reaction $\mathcal{R}_j$ depends on the state $X(t)$ at time t and describes the reaction rate of $\mathcal{R}_j$ as follows:

$$P(X(t + \Delta t) = x + \nu_j | X(t) = x) = a_j(x)\Delta t + o(\Delta t). \tag{2}$$

A larger value of $a_j(X(t))$ at time t indicates faster rate of reaction $\mathcal{R}_j$ at time t. Probability of firing reaction $\mathcal{R}_j$ in interval $(t, t + \Delta t]$ increases as $a_j(X(t))$ increases.

The probability that no reaction fires in the time interval $(t + \Delta t]$ in the SRN $\mathbf{X}$ is given by:

$$P(X(t + \Delta t) = x | X(t) = x) = 1 - \sum_{j=1}^{M} a_j(x)\Delta t + o(\Delta t). \tag{3}$$

Defining a new quantity $a_0(x) := \sum_{j=1}^{M} a_j(x)$, (3) can be rewritten as:

$$P(X(t + \Delta t) = x | X(t) = x) = 1 - a_0(x)\Delta t + o(\Delta t). \tag{4}$$

Given $X(t) = x$ at time t, it can be derived from (4) that the time to fire next reaction follows an exponential distribution with parameter $a_0(x)$.

The stochastic mass-action kinetics principle can be represented by the diagram given below for some reaction $\mathcal{R}_j \in \mathbf{R}$:

$$\alpha_{j,1}S_1 + \ldots + \alpha_{j,n}S_n \to \beta_{j,1}S_1 + \ldots + \beta_{j,n}S_n. \tag{5}$$

As implied by (5), when a reaction $\mathcal{R}_j$ is fired, $\alpha_{j,1}, \cdots, \alpha_{j,N}$ molecules of the corresponding species $S_1, \cdots, S_N$ are consumed and $\beta_{j,1}, \cdots, \beta_{j,N}$ molecules are produced. The stoichiometric vector $\nu_j := (\beta_{j,1} - \alpha_{j,1}, \ldots, \beta_{j,n} - \alpha_{j,n})$ can contain both positive and negative integers, and hence $\nu_j \in \mathbb{Z}$.

Given a positive reaction constant c_j and state $X(t) = x = (x_1, \ldots, x_N)$ at time t, the propensity function $a_j(x)$ of reaction $\mathcal{R}_j$ as derived from mass-action kinetics is given by:

$$a_j(x) = c_j \prod_{i=1}^{N} \frac{x_i!}{(x_i - \alpha_{j,i})!} \mathbf{1}_{\alpha_{j,i} \leq x_i} \tag{6}$$

where $\alpha_{j,i}$ is the number of S_i molecules consumed when reaction R_j is fired and $\mathbf{1}_{\alpha_{j,i} \leq x_i}$ is an indicator function that can be described as below:

$$\mathbf{1}_{\mathbf{A}}(x) = \begin{cases} 1, & \text{if } x \in \mathbf{A}, \\ 0, & \text{otherwise.} \end{cases}$$

In the remaining literature, (6) will be used to calculate propensities of reactions given the state $X(t)$ at time t.

Random Time Change Representation

Let x_0 be the initial state of an SRN $\{X(t), t \geq 0\}$ with reaction channels $(\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_M)$. The random time-change representation of $\mathbf{X}$ as shown by T.G. Kurtz [16, 8, 15] is given as:

$$X(t) = x_0 + \sum_{j=1}^{m} \nu_j Y_j \left(\int_0^t a_i(X(s)) ds \right), \tag{7}$$

where $Y_j : \mathbb{R}_{\geq 0} \times \Omega \to \mathbb{Z}_{\geq 0}$ are independent unit-rate Poisson processes. The representation (7) models the path of the process $\{X(t), t \geq 0\}$ in continuous time as a combination of paths of M independent unit-rate Poisson processes. Each reaction is represented by a unit-rate Poisson process, and for $1 \leq j \leq M$, the value $\int_0^t a_j(X(s))ds$ is called the internal time of Y_j.

Later in the chapter, we will see how *Next Reaction Method* (NRM) and *Modified Next Reaction Method* (MNRM) make use of (7) to simulate exact paths of SRNs by sampling next-reation times and next-reaction internal times respectively.

Example - Gene transcription and translation

For illustration, we look at an example of an SRN that models gene transcription and translation. More details on this model can be found in [13]. The SRN, say $X(t), t \geq 0$, consists of three species (M, P, D) and five reactions as shown below:

- $\emptyset \to M$, Transcription of a gene into mRNA.
- $M \to M + P$, Translation of mRNA into proteins.
- $P + P \to D$, Production of stable *Dimers* from proteins.
- $M \to \emptyset, P \to \emptyset$, Degradation of mRNA and proteins respectively.

From the above reactions, we can construct ν_j for each reaction as follows:

$$\nu_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \nu_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \nu_3 = \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix}, \quad \nu_4 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad \nu_5 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}.$$

The propensity functions of the five reactions as functions of the state $X(t) = (m(t), p(t), d(t))^T$ at some time t are given by:

$$a_1(X(t)) = 25,\ a_2(X(t)) = 10^3 m(t),\ a_3(X(t)) = 10^{-3} p(t)(p(t)-1),$$
$$a_4(X(t)) = 0.1 m(t) \text{ and } a_5(X(t)) = p(t).$$

If the initial state $X(0) = x_0$, the process can be modeled with Kurtz' random time change representation (7) as:

$$\begin{aligned} X(t) = x_0 &+ Y_1\left(\int_0^t 25 ds\right) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + Y_2\left(\int_0^t 10^3 m(s) ds\right) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ &+ Y_3\left(\int_0^t 10^{-3} p(s)(p(s)-1) ds\right) \begin{bmatrix} 0 \\ -2 \\ 1 \end{bmatrix} + Y_4\left(\int_0^t 0.1 m(s) ds\right) \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \\ &+ Y_5\left(\int_0^t p(s) ds\right) \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \end{aligned}$$

Where Y_js are unit-rate Poisson processes with internal times given as $\int_0^t a_j(X(s))ds$.

2.3 Simulation of SRNs

Analytical and numerical solutions to differential equations involving SRNs are usually infeasible. Some challenges met while finding analytical solutions for an SRN, lets say $\mathbf{X}$, include: i) non-linearity of propensity functions a_js w.r.t. states of $\mathbf{X}$, ii) large number of state space of $X(t), t \geq 0$, to name a few. For this reason, simulating paths of SRNs using Monte Carlo method can help us circumvent these challenges and find approximate values of several quantities of interest including but not limited to time taken by process $X(t), t \geq 0$ to reach a specified subset of its state space and expected value of an observable g at some time T, i.e. $E[g(X(T))]$.

Exact Algorithms

Exact algorithms are methods that can be used to simulate paths of an SRN $X(t), t \geq 0$ that obey the probabilities described by (2) and (3). These paths have correct statistical distributions and hence can be used to estimate values such as expected value of an observable g at time T, i.e. $E[g(X(t))]$. We will discuss three exact methods: 1. *Stochastic Simulation Algorithm* (SSA), 2. *Next Reaction Method* (NRM) and 3. *Modified Next Reaction Method* (MNRM) [9, 1].

2.3.1 Stochastic Simulation Algorithm (SSA)

In [11], SSA was popularised by Gillepsie to simulate paths of SRNs that could be used to describe evolution of chemical reactions. The algorithms involves sampling two random numbers at each step: i) first random number is used to sample the time to next reaction, lets say h, and ii) second random number is used to sample which reaction takes place after time h. The algorithm is based on an observation from (2) that, given $X(t) = x$ at time t, the probability of $\mathcal{R}_j$ being the only reaction that takes place in time interval $(t, t+h)$ is given by $a_j(x) \times exp(-a_0(x)h)$ which can be split and rewritten as $(a_j(x)/a_0(x)) \times a_0(x)exp(-a_0(x)h)$.

As observed from (4), given $X(t) = x$ at time t, time to next reaction has an exponential distribution with parameter $a_0(x)$. Hence, the probability of firing reaction $\mathcal{R}_j$ in time interval $(t, t+h)$ can be split into two independent probabilities: probability of selecting reaction $\mathcal{R}_j$ from set of all the reactions, and probability density that follows an exponential distribution with parameter $a_0(x)$.

SSA involves the following sequence of steps to produce an exact simulation of a path of an SRN $X(t), t \geq 0$:

Algorithm 1 Stochastic Simulation Method

```
Algorithm: Stochastic Simulation Method
Initialize $\mathbf{x} = x_0$, $\mathbf{t} = 0$
Declare array $\mathbf{a}$
Declare constant $\mathbf{T}$, array $\nu$
while $\mathbf{t} < \mathbf{T}$ do
  for each reaction-channel $\mathbf{i}$ do
    Calculate propensity $\mathbf{a}[\mathbf{i}]$ using $x$
  end for
  Set $\mathbf{a_0} = \sum_j \mathbf{a}[\mathbf{j}]$
  Generate a random number $\mathbf{r_1}$ from $unif(0,1)$
  Set $\mathbf{h} = \frac{1}{\mathbf{a_0}} \ln\left(\frac{1}{\mathbf{r_1}}\right)$
  Generate a random number $\mathbf{r_2}$ from $unif(0,1)$
  Find $\mathbf{m}$ such that $\sum_{k=1}^{m-1} \mathbf{a}[\mathbf{k}] < \mathbf{a_0 r_2} \leq \sum_{k=1}^{m} \mathbf{a}[\mathbf{k}]$
  Set $\mathbf{x} = \mathbf{x} + \nu[\mathbf{m}]$, $\mathbf{t} = \mathbf{t} + \mathbf{h}$
end while
```

It can be observed from the algorithm that for each reaction that is fired in the simulation, two

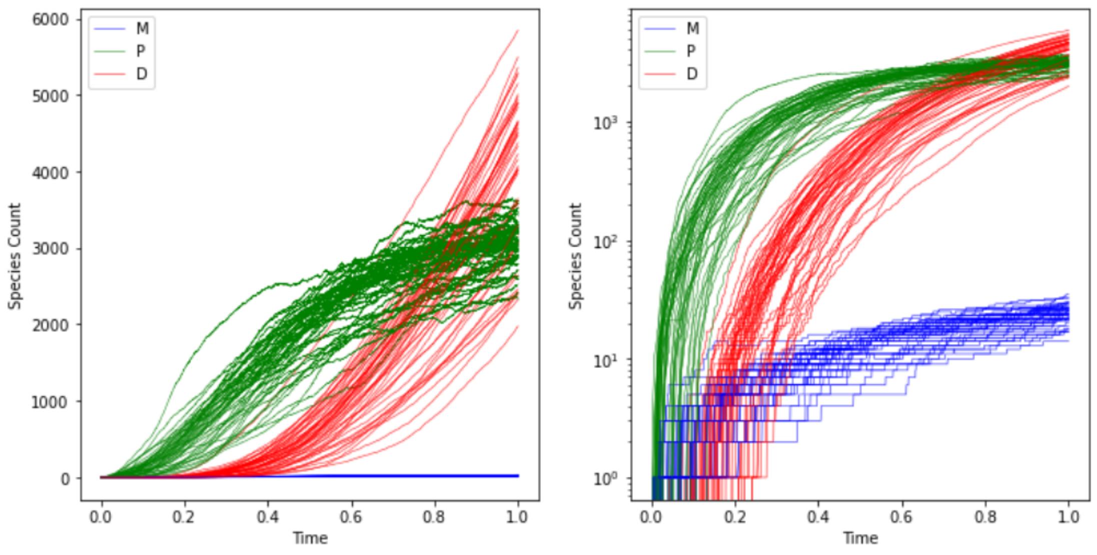


Figure 2: Fifty paths of gene transcription and translation SRN model simulated using SSA, with the left image showing a linear scale y-axis and the right image displaying a log scale y-axis.

random numbers are being generated, first to sample the time to next reaction and second to sample the reaction that should be fired based on their propensities. We will see in the following sections that NRM and MNRM require generation of only one random number each time a reaction is fired.

The plots in Figure 2 show 50 exact paths of the gene transcription and translation SRN discussed above using Stochastic Simulation Algorithm.

2.3.2 Next Reaction Method (NRM)

The representation (7) of SRN paths as shown by Kurtz is given as, assuming SRN $\mathbf{X}$ is initially at state x_0:

$$X(t) = x_0 + \sum_{j=1}^{m} \nu_j Y_j \left(\int_0^t a_i(X(s))ds \right).$$

The representation (7) separates the randomness of the system from the state of the system, as the randomness is only contained in the Y_js which are unit-rate independent Poisson processes that count the number of times the corresponding reactions $\mathcal{R}_j$ fired until time t. The intensities of the Y_js are given by $a_j(X(t))$. Hence, each Y_j comes with its own time-frame, apart from the actual time frame of the chemical system. We can define $T_j(t) = \int_0^t a_j(X(s))ds$, to represent the *internal times* of the corresponding Y_j. Using internal times $T_j(t)$ in simulating SRN paths requires solving two crucial challenges: i) how to sample firing times of each Y_j in its own time frame and ii) how to translate these firing times to the actual time of the chemical system. We will see that NRM and MNRM try to solve these challenges in two different ways but both essentially use the same property of separation of randomness from the state of the system as suggested in (7).

For an SRN with M reactions, NRM first calculates the firing times of all the reactions, say $\delta t_j, j \leq M$ at time $t = 0$ by sampling from exponential distributions with internal times of Y_js as parameters, based on the assumption the a_j remain unchanged until $j^t h$ reaction is fired. The reaction with the least firing time is fired first. After firing a reaction, the change in state of the system leads to change in propensities of the reaction-channels. This change in propensities requires the firing times to be updated, but the internal times of Y_j until their next firing remain the same. In the subsequent steps, the reaction with the nearest firing time is fired, and the firing times of the rest of the reactions are updated accordingly. For the reaction that got fired, new reaction time is sampled based on the updated propensity. The simulation is carried out until it is explicitly stopped or time limit of the simulation is reached.

Lets say, at some time t, we know $X(t)$, internal times $T_j = T_j(t)$ and propensities $a_j = a_j(X(t))$. Assuming a_j remains constant over the next firing time of Y_j, we can also derive δt_j for each reaction. Now, the next reaction after time t that has to be fired should be the one with the minimum δt_j. Let us denote this value, i.e. $min_j \delta t_j$ as Δ, and let μ be the value of j with minimum δt_j. The system can then be updated to $\bar{t} = t + \Delta$ and all the propensities can be updated to $\bar{a}_j = a_j(X(\bar{t}))$. For $j = \mu$, next firing time should be sampled using the updated propensity $a_\mu(X(\bar{t}))$. As the propensity functions have changed for all the reactions, the next firing time would not be same as old firing times for $j \neq \mu$. But it is to be noted that the internal times of the reactions until their next firing remain the same. Internal times that have passed until $\bar{t}$ are given by $T_j(\bar{t}) = T_j(t) + a_j \Delta$. Hence the amount of internal time that should pass before firing Y_j is given by

$$(T_j(t) + a_j \delta t_j) - (T_j(t) + a_j \Delta) = a_j(\delta t_j - \Delta).$$

Also, we note that the difference in internal times between t and $\bar{t}$ for Y_j are given by $\bar{a}_j \delta \bar{t}_j$. Hence, equating the differences between internal times for reactions, we get

$$\delta \bar{t}_j = \frac{a_j}{\bar{a}_j}(\delta t_j - \Delta).$$

NRM generates M random numbers during initialisation to sample the firing times of all the reactions. After the first step, it requires only one random variable per firing to sample the next firing time of the reaction that fired in the current step. NRM algorithm involves following sequence of steps:

Unlike SSA where two random variables were required in each step, NRM requires only one random number per firing of a reaction.

Algorithm 2 Next Reaction Method

```
Algorithm: Next Reaction Method
Initialize $\mathbf{x} \leftarrow x_0$, $\mathbf{t} \leftarrow 0$
Declare array $\mathbf{a}$, array $\tau$
Declare constant $\mathbf{T}$, array $\nu$
for each reaction-channel $\mathbf{i}$ do
  Calculate propensity $\mathbf{a}[\mathbf{i}]$ using $x$
  Generate random number $\mathbf{r}$ from $unif(0,1)$
  Set $\tau[\mathbf{i}] = (1/a[i])\ln(1/r)$
end for
while $\mathbf{t} < \mathbf{T}$ do
  Set $\mathbf{t_{next}} = \min_j \tau[\mathbf{j}]$, $\mu = \arg\min_j \tau[\mathbf{j}]$
  Set $\mathbf{x} = \mathbf{x} + \nu[\mu]$, $\mathbf{t} = \mathbf{t_{next}}$
  for each reaction-channel $\mathbf{i}$ do
    Calculate propensity $\bar{\mathbf{a}}[\mathbf{i}]$ using $x$
    if $\mathbf{i} \neq \mu$ then
      Set $\tau[\mathbf{i}] = \mathbf{t} + (\mathbf{a}[\mathbf{i}]/\bar{\mathbf{a}}[\mathbf{i}])(\tau[\mathbf{i}] - \mathbf{t})$
    end if
  end for
  Generate a random number $\mathbf{r}$ from $unif(0,1)$
  Set $\tau[\mu] = \mathbf{t} + (\mathbf{1}/\bar{\mathbf{a}}[\mu])\ln(\mathbf{1}/\mathbf{r})$
  for each reaction-channel $\mathbf{i}$ do
    Set $\mathbf{a}[\mathbf{i}] = \bar{a}[i]$
  end for
end while
```

2.3.3 Modified Next Reaction Method (MNRM)

MNRM is similar to NRM in the way that it uses Kurtz' representation of evolution of X to simulate paths of $\mathbf{X}$, but unlike NRM, it does not require time conversions and works explicitly with internal times to calculate next firing times of reactions. At some time t, let P_j denote the first firing internal time of Y_j that is strictly larger than T_j: $P_j = min_s\{s > T_j : Y_j(s) > Y_j(T_j)\}$, where T_js are the internal times of Y_js until t. Hence the next firing time of the reaction can be given as:

$$\delta t_j = (P_j - T_j)/a_j.$$

The internal time until the next firing of a reaction Y_j should remain constant irrespective of the evolution of the system until Y_j is fired, only the actual firing time changes based on the changes in propensities. MNRM uses this fact and lets us sample internal times, here P_j, directly instead of sampling actual next times of reactions. MNRM algorithm is as follows:

MNRM calculates M random numbers during initialisation to sample the internal times of first firings of all the Y_js, and subsequently uses only one random number per iteration to re-sample the internal time of the fired reaction. MNRM and NRM differ in the way they use random numbers. MNRM utilizes random numbers to sample the internal times until the first firing, eliminating the need for time-conversions as in NRM.

Algorithm 3 Modified Next Reaction Method

```
Algorithm: Modified Next Reaction Method
Initialize $\mathbf{x} = x_0$, $\mathbf{t} = 0$, array $\mathbf{P} = 0$, array $\mathbf{T} = 0$
Declare array $\mathbf{a}$, array $\delta t$
Declare constant $\mathbf{T}$, array $\nu$
for each reaction-channel i do
  Calculate propensity $\mathbf{a}[\mathbf{i}]$ using $\mathbf{x}$
  Generate random number $\mathbf{r}$ from $unif(0,1)$
  Set $\mathbf{P}[\mathbf{i}] = (1/a[i])\ln(1/r)$
end for
while $\mathbf{t} < \mathbf{T}$ do
  for each reaction-channel i do
    Set $\delta t[i] = (P[i] - T[i])/a[i]$
  end for
  Set $\Delta = \min_i \delta t[i]$, $\mu = \arg\min_i \delta t[i]$
  Set $\mathbf{t} = \mathbf{t} + \mathbf{\Delta}$, $\mathbf{x} = \mathbf{x} + \nu[\mu]$
  for each reaction-channel i do
    Set $\mathbf{T}[\mathbf{i}] = \mathbf{T}[\mathbf{i}] + \mathbf{a}[\mathbf{i}]\mathbf{\Delta}$
  end for
  Generate a random number $\mathbf{r}$ from $uniform(0,1)$
  Set $\mathbf{P}[\mu] = \mathbf{P}[\mu] + \ln(\mathbf{1}/\mathbf{r})$
  for each reaction-channel i do
    Calculate $\mathbf{a}[\mathbf{i}]$ using $\mathbf{x}$
  end for
end while
```

2.3.4 Approximate Algorithms - Tau-Leap Algorithm

Exact algorithms, although being useful in simulating paths of SRNs with correct distributions, may not be feasible for many reasons. High propensities can lead to lower and lower reaction times, and hence increase computation load of the algorithms. Approximate algorithms mitigate these challenges by simulating paths of SRNs over discrete time steps. One such algorithm is the Tau-leap algorithm [17].

In this algorithm, a small time interval τ is fixed and a path of an SRN $\mathbf{X}$ is simulated by calculating $X(t)$ at $t = 0, \tau, 2\tau, \ldots, N\tau$ where $N\tau$ is the time limit of the simulation. After k time-steps, when $t = k\tau$, the algorithm progresses by sampling the number of each kind of reaction that would take place in the interval $(k\tau, (k+1)\tau]$ and update the state of the system accordingly. Let the number of reactions that take place in the interval $(k\tau, (k+1)\tau]$ for M reaction channels be $(\mu_{k+1,1}, \mu_{k+1,2}, \ldots, \mu_{k+1,M})$, then the system is updated as follows:

$$X((k+1)\tau) = X(k\tau) + \sum_{j=1}^{M} \mu_{k+1,j}\nu_j.$$

If the initial state of the SRN is $X(0) = x_0$, and if the time interval $(0, T]$ is partitioned into N

time intervals, $(0, s_1], (s_1, s_2], \dots, (s_{N-1}, s_N]$, the integral $\int_0^t a_j(X(s))ds$ in Kurtz's representation (7) can be approximated as a summation over discrete time intervals so that we get

$$\int_0^t a_j(X(s))ds \approx \sum_{q=0}^{N-1} a_j(X(s_q))(s_{q+1} - s_q).$$

Let us represent the path simulated by approximate methods by $\overline{X}(t)$ to differentiate it from exact paths. The approximate path $\overline{X}$ that can be modeled by approximating the integrals in random time change representation (7) is thus given as:

$$\overline{X}(t) = x_0 + \sum_{j=1}^{m} Y_j \left(\sum_{q=0}^{N-1} a_j(\overline{X}(s_q))(s_{q+1} - s_q) \right) \nu_j. \tag{8}$$

While simulating $\overline{X}(t + \tau)$ given the state $\overline{X}(t)$, the second term in (8) becomes $Y_j \left(\sum_{q=0}^{N-1} a_j(\overline{X}(s_q))(s_{q+1} - s_q) + a_j(\overline{X}(t))\tau \right)$. Y_j becomes a Poisson process with internal time as sum of two terms. Hence, Y_j can be split into two terms: $Y_j \left(\sum_{q=0}^{N-1} a_j(\overline{X}(s_q))(s_{q+1} - s_q) \right)$ and a Poisson random Variable $\mathcal{P}_j(a_j(\overline{X}(t))\tau)$. The first term is already known as it was used to arrive at the state $\overline{X}(t)$. Hence the expression for $\overline{X}(t+\tau)$ becomes:

$$\overline{X}(t + \tau) = \overline{X}(t) + \sum_{j=1}^{M} \mathcal{P}_j(a_j(x)\tau)\nu_j,$$

where $\mathcal{P}_j(\lambda_j), j = 1, 2, \dots, M$, are independent Poisson processes with intensity λ_j.

Tau-Leap algorithm involves the following sequence of steps:

Algorithm 4 Tau-Leap Algorithm

```
Initialize $\mathbf{x} = x_0$, $\mathbf{t} = 0$
Declare array a, array p
Declare constant $\tau_0$, T, array $\nu$
while t < T do
  for each reaction-channel i do
    Calculate propensity a[i] using x
    Generate random number p[i] from Poisson distribution $\mathcal{P}(\mathbf{a}[\mathbf{i}]\tau)$
  end for
  Set $\mathbf{t} = \mathbf{t} + \tau, \mathbf{x} = \mathbf{x} + \sum_{\mathbf{j=1}}^{\mathbf{M}} \mathbf{p}[\mathbf{j}]\nu[\mathbf{j}]$
end while
```

The plots in Figure 3 show 50 approximate paths of gene transcription and translation SRN simulated using tau-leap algorithm.

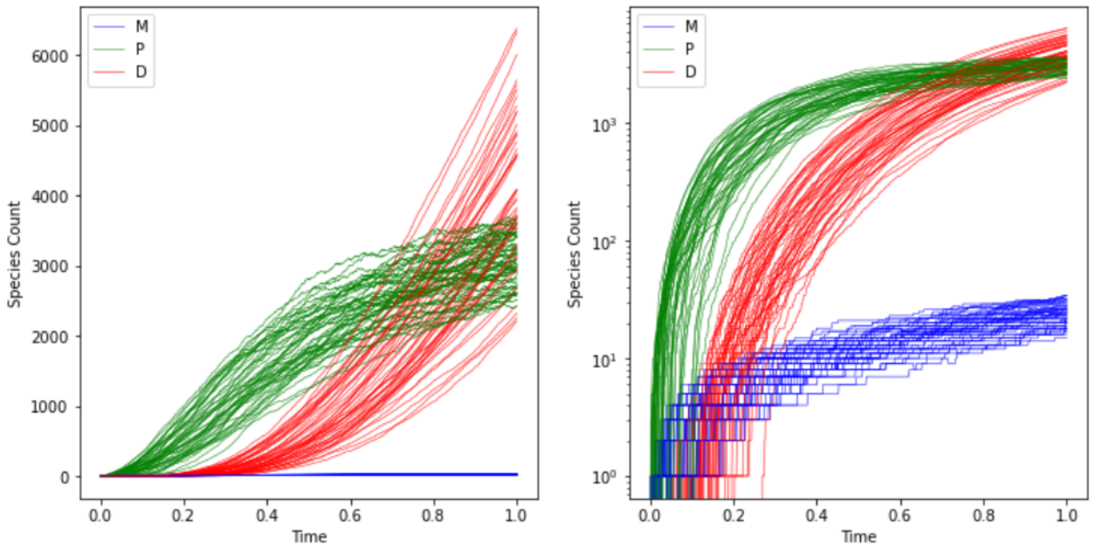


Figure 3: Fifty approximate paths of gene transcription and translation SRN model simulated using tau-leap method, with the left image showing a linear scale y-axis and the right image showing a log scale y-axis.

3 Analogy between Supply Chains and Chemical Reaction Modelling

As seen in chapter 2 - chemical reaction modelling, simulations can be useful in modelling the dynamic behaviour of systems with multiple moving parts (molecules) and processes (reactions). Supply chains involve coordinated transportation of materials from suppliers to consumers. As global supply chains grow more and more complex, simulations play an important role in designing of policies prior to implementation by performing sensitivity-analysis, and ensuring resiliency and sustainability by being able to incorporating several scenarios in the simulations.

One of the most widely used algorithms to exactly simulate supply chains is the discrete-event simulation (DES) algorithm, that involves instantaneous change in state of the system at distinct points in time [3]. Due to computational and performance related shortcomings of exact methods such as DES, approximate methods can be employed for simulations such as the time-bucket method, that involves simulating paths of the supply chain at discrete time intervals. These methods accelerate the simulations, as system state is updated only at fixed time-intervals, unlike exact methods where each distinct event leads to a change in state of the system.

In this chapter, we discuss DES, time-bucket method and L-Leap method, that is an extension of D-Leap method [2], and explore the similarities between supply chain modelling and chemical reaction modelling using SRNs. We will incorporate delays in the processes in supply chain simulations, following the instantaneous-consumption-delayed-production approach [3].

3.1 DES of supply chains

The DES [3] is a widely used algorithm to simulate logistics and supply chains. The *events* and *states* in DES can be modelled in accordance to the specific goal and use-case of the simulation. Each event taking place in a simulation results in an instantaneous change in the state of the system. One kind of DES is the next-event time-advance algorithm where the time leaps to the next nearest event among the list of future events. It is worth noting that next-event time-advance is analogous to SSA, that was mentioned in section (2.3.1). Events in the DES algorithm are analogous to the chemical reactions in SSA, and the states are analogous to the number of molecules in the chemical system being simulated by SSA. Similar to SSA and other exact methods of simulation SRNs, the complexity of DES depends on the number of events that take place within a time period. Using DES for simulating complex supply chains with high number of processes and high production rates can be computationally expensive. Approximate methods, such as the time-bucket method, accelerate simulations by calculating state after fixed time intervals. Approximate methods circumvent the computational challenges posed by exact methods by changing state of the system at fixed time intervals, and not separately for each discrete event.

3.2 Time-bucket method

In the time-bucket method [3], a time interval τ is fixed and the algorithm progresses by simulating the number of each event that takes place in the time intervals $(0, \tau], (\tau, 2\tau], \ldots$ until the time limit of the simulation is reached, unlike the DES where each discrete event is simulated at their exact simulated times. The time-bucket method is, thus, an approximation of the DES, similar to how the tau-leap algorithm (2.3.4) used in simulating evolution of chemical systems, is an approximation of the exact method SSA. Hence, time-bucket method is often useful in simulating complex supply chains with large number of events and states.

In the time-bucket method simulation of a supply chain, each event represents a process in the supply chain, and we consider $(P_1, P_2, \ldots, P_d)$ as the supply chain's parts that are manufactured and transported between different processes. These parts are analogous to the species of molecules $(S_1, S_2, \ldots, S_d)$ in a chemical reaction as discussed in section 2.2. Given J processes, each process $j \in \{j \mid j \in \mathbb{Z}, 1 \leq j \leq J\}$ can be represented as follows:

$$\sum_{i \in A_j} c_{ij} P_i \rightarrow \sum_{i \in B_j} p_{ij} P_i,$$

where A_j and B_j are the sets of parts consumed and produced respectively by process j, and c_{ij} and p_{ij} are the numbers of part P_i consumed and produced respectively by process j. If we consider c_{ij} and p_{ij} as the i^{th} elements of vectors $C_j \in \mathbb{Z}_+^d$ and $P_j \in \mathbb{Z}_+^d$ respectively, the vector $\nu_j = P_j - C_j$ is equivalent to the stoichiometric vector ν_j of a reaction (here process) j. Given state $X(t) = (x_1(t), \ldots, x_d(t))$ at time t, each process $j \in \{j \mid j \in \mathbb{Z}, 1 \leq j \leq J\}$ is also associated with a production rate $\lambda_j(X(t))$ that describes the rate of consumption and production of parts. The maximum value that $\lambda_j(X(t))$ can take is a fixed value represented by λ_j^{max}. The rate of production by process j at time t depends on the availability of all parts $P \in A_j$ at time t as well as possible disruptions during the time period of simulation. Less availability of the parts hampers the rate of the processes, leading to less number of occurrences of the processes in the time intervals.

In a single simulation of a supply chain using the time-bucket method, a time interval τ is fixed and the approximate state of the system $\overline{X}(t) \in \mathbb{Z}_+^d$ is simulated at $t = \tau, 2\tau, \ldots, N\tau$ where $T = N\tau$ is the time limit of the simulation. The state $\overline{X}(t+\tau)$, given the state $\overline{X}(t)$ at time t is updated as follows:

$$\overline{X}(t+\tau) = \overline{X}(t) + \sum_{j=1}^{J} \mathcal{P}_j(\lambda_j(\overline{X}(t))\tau)\nu_j, \tag{9}$$

where $\mathcal{P}_i$s are independent Poisson processes with rates given by $\lambda_j(\overline{X}(t))\tau$. At each time step, each $\overline{x}_i(t+\tau)$ in $\overline{X}(t+\tau) = (\overline{x}_1(t+\tau), \overline{x}_2(t+\tau), \ldots, \overline{x}_d(t+\tau))$ is checked for negativity, and the number of processes simulated in the time period $(t, t+\tau]$ are adjusted accordingly.

Let us assume that $\overline{X}^{temp}(t+\tau) = (\overline{x}_1^{temp}(t+\tau), \ldots, \overline{x}_d^{temp}(t+\tau))$ is the updated state obtained after (9) in the time interval $(t, t+\tau]$, and let $\overline{x}_i^{temp}(t+\tau)$ be the least negative number in $\overline{X}^{temp}(t+\tau)$. For the process j that consumes c_{ij} number of parts P_i in a single occurrence, the number of simulated occurrences of process j in the interval $(t, t+\tau]$ is reduced by -$\lfloor \overline{x}_i^{temp}(t+\tau)/c_{ij} \rfloor$. Hence the number of parts of P_i at time $t+\tau$ is updated as follows:

$$\overline{x}_i^{temp}(t+\tau) = \overline{x}_i^{temp}(t+\tau) - c_{ij}\lfloor \overline{x}_i^{temp}(t+\tau)/c_{ij} \rfloor.$$

It should be noted that $-c_{ij}\lfloor \overline{x}_i^{temp}(t+\tau)/c_{ij} \rfloor$ is a positive number that, when added to $\overline{x}_i^{temp}(t+\tau)$, makes $\overline{x}_i^{temp}(t+\tau)$ a non-negative number. This process is repeated until all the negative numbers in state $\overline{X}^{temp}(t+\tau)$ have been eliminated, and finally $\overline{X}(t+\tau)$ is set equal to $\overline{X}^{temp}(t+\tau)$. $\overline{X}(t+\tau)$ can then be used in the subsequent time step in the time-bucket algorithm.

3.3 Logistic-Leap (L-Leap) Method

Each process in a supply chain is usually associated with a delay. A delay for a process j is the amount of time that passes between consumption and production of parts by that process. L-Leap method [3] extends D-Leap method [2] used in simulating chemical reactions with delays, by incorporating several features of a supply chain such as transportation, production time, inventory management and push and pull systems. The simulation of supply chains using time-bucket method can be split into two phases: instantaneous consumption and delayed production. This is based on the idea that each process in a supply chain consumes parts instantaneously and takes some finite non-zero time period to produce parts.

3.3.1 Consumption

Each occurrence of any process j is assumed to consume parts instantaneously and produce parts with some delay. Let $C_j = (c_{j1}, c_{j1}, \ldots, c_{jd})$ represent the vector containing number of parts consumed during a single occurrence of a process j. During a simulation, given state $\overline{X}(t)$ at time t, consumption is simulated by calculating the number of each process j in time interval $(t, t+\tau]$ and updating the state as follows:

$$\overline{X}(t+\tau) = \overline{X}(t) - \sum_{j=1}^{J} \mathcal{P}_j(\lambda_j(\overline{X}(t))\tau)C_j. \tag{10}$$

Each element in $\overline{X}(t+\tau)$ is checked for negativity, and the number of processes simulated in $(t, t+\tau]$ are updated to ensure non-negativity of $\overline{X}(t+\tau)$. The $\mathcal{P}_j$s are independent Poisson processes with rates $\lambda_j(\overline{X}(t))\tau$ respectively.

Each process $j \leq J$ can be associated with a range $[t_j^{min}, t_j^{max}]$ that denotes the range of time delays the process j can have between consumption and production. With time delays in processes, the state of the system is influenced not only by its immediate predecessor but also by delayed processes,

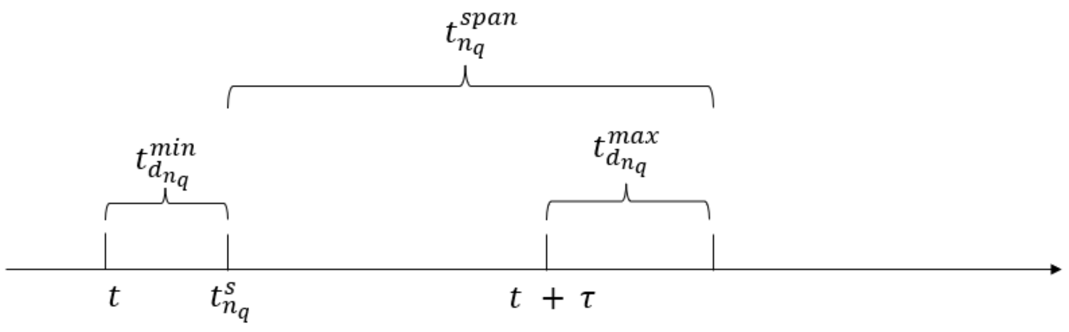


Figure 4: Timeline for processes simulated during time interval $(t, t + \tau]$.

rendering the process non-Markovian. In order to make the system Markovian, the algorithm in [3] extends the state by incorporating each delayed process simulated in the time interval $(t, t + \tau]$ into the state of the system. This is required in order to update the state of the system appropriately when these processes produce their respective parts after delays. To achieve this, in [3], a queue structure is maintained where each entry stores the following information, assuming q_j occurrences of process j are simulated in time interval $(t, t + \tau]$:

- The index of delayed process $d_{n_q} = j$.
- Number of delayed processes $Q_{n_q} = q_j$.
- Earliest time of completion of delayed process d_{n_q} represented by $t^s_{n_q} = t + t_j^{min}$.
- Time span of completion of Q_{n_q} processes represented by $t^{span}_{n_q} = t + \tau + t_j^{max} - (t + t_j^{min}) = \tau + t_j^{max} - t_j^{min}$.

3.3.2 Production

After adjusting the state of the system by incorporating consumption by processes simulated in the interval $(t, t + \tau]$, the algorithm iterates through to queue to determine the number of processes that completed in the current time bucket. For an entry n_q, if $t^s_{n_q} > t + \tau$, i.e. earliest time of completion of process d_{n_q} does not lie in the interval $(t, t + \tau]$, then the entry is skipped. Similarly, an entry n_q is skipped if $Q_{n_q} = 0$. For entries not lying in the above criteria, a fraction of the Q_{n_q} processes will be completed in the current time bucket. Let $P_j = (p_{j1}, p_{j1}, \ldots, p_{jd})$ represent the vector containing number of parts produced during a single occurrence of a process j. The following updates are made to the queue entries and the state of the system:

- Number of completed processes K_{n_q} are modeled by a Binomial distribution with parameters

Q_{n_q} and $min\left(\frac{t+\tau-t^s_{n_q}}{t^{span}_{n_q}}, 1\right)$.

- Number of delayed processes is updated: $Q_{n_q} = Q_{n_q} - K_{n_q}$.
- Earliest time of completion is updated: $t^s_{n_q} = t + \tau$.
- Span is updated: $t^{span}_{n_q} = max(0, t^{span}_{n_q} - (t + \tau - t^s_{n_q}))$.
- Finally, the state of system is updated as follows:

$$\overline{X}(t+\tau) = \overline{X}(t) + K_{n_q} P_{n_q}.$$

3.4 Monte Carlo methods in supply chain management

Monte Carlo methods [7] are computational algorithms that rely on random sampling to analyse complex systems and obtain numerical results, that are otherwise difficult to obtain analytically. These are particularly useful in systems that have input parameters as random variables, following some probability distributions. Monte Carlo methods are used to simulate the paths of systems by repeatedly sampling from probability distributions, and the quantities of interest are calculated by aggregating over the simulated paths. As the method depends on randomness to produce numerical results for the quantities of interest, the two most prominent errors that are considered while simulating a system using MC methods are:

- Sampling error - Error that arises due to the inherent random nature of the MC method. As parameters are repeatedly sampled from probability distributions while simulating paths of a system, each path can give different numerical result for the quantity of interest, and only the aggregated value over a number of simulations can give a reliable value. This error can be minimised by aggregating over a large number of simulated paths.
- Bias error - Error that occurs due to the approximations incorporated in the simulations. This error usually arises when the simulation algorithm does not represent a real-world system. In case of time-bucket method, the length of time-interval affects the bias error. We recall that time-bucket method and tau-leap method are approximations of exact methods such as DES and SSA.

If the aim of simulations is to carry out sensitivity analysis based on input parameters, then another layer of simulations have to be performed to assess the effect of different input parameters on the output quantities. This approach is known as Multi Level Monte Carlo (MLMC) method [10].

While simulating supply chains, the system is usually simulated a large number of times to minimise the sampling error and to obtain a reliable value for the quantity of interest. We see an example of using MC method to simulate a supply chain with three supply parts and five interdependent processes below.

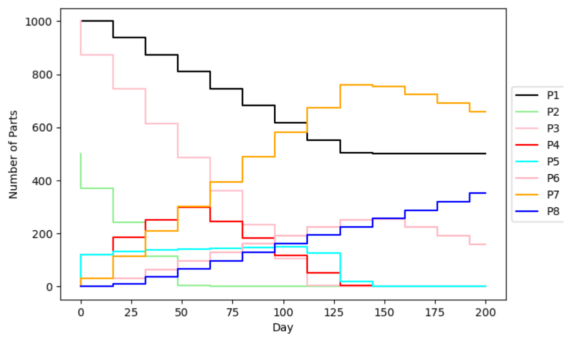
Example - Simulating a supply chain with three supply parts and five processes

We use MC method to simulate the supply chain represented in Figure 1 with eight parts $P_1, P_2, \ldots, P_8$, and five processes given as:

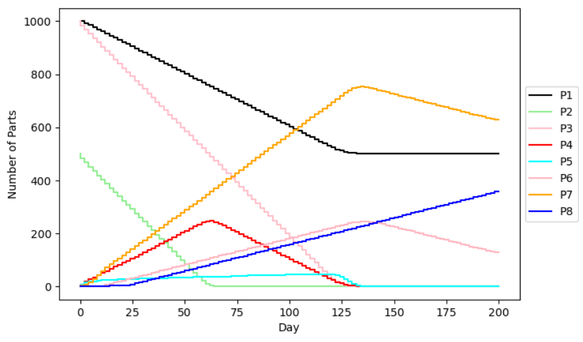
$$1 : P_2 \rightarrow P_4,\ 2 : P_3 \rightarrow P_5,\ 3 : P_1 + P_4 \rightarrow P_6,$$

$$4 : P_5 \rightarrow P_7 \textit{ and } 5 : P_6 + P_7 \rightarrow P_8.$$

The initial state is assumed to be: $x_1(t=0) = 1000,\ x_2(t=0) = 500,\ x_3(t=0) = 1000,\ x_i(t=0) = 0\ \forall\ i\ \in \{4,5,6,7,8\}$. P_1, P_2 and P_3 are the initial supplies of the supply chain, that are consumed by processes $3, 1$ and 2 respectively. The production rates of the processes are assumed to be $\lambda_1 = 8, \lambda_2 = 8, \lambda_3 = 4, \lambda_4 = 8$ and $\lambda_5 = 2$. It is also assumed that the time delays of production by the processes are deterministic, i.e. $t_j^{min} = t_j^{max}$ for $j \in \{j \mid j \in \mathbb{Z}, 1 \le j \le 5\}$. In this example, the time delays are assumed to be $t_1^{min} = 1, t_2^{min} = 1, t_3^{min} = 10, t_4^{min} = 1$ and $t_5^{min} = 10$.



(a) Time history with $\tau = 16$ days.



(b) Time history with $\tau = 2$ days.

Figure 5: Time histories of the supply chain represented in Figure 1, 200 MC simulations.

It can be seen from Figure 5 that both time-buckets, i.e. $\tau = 2$ and $\tau = 16$ are able to capture dynamics of the supply chain. The consumption of P_1 stops at $x_1 = 500$ units, which is right after entire 500 parts of P_2 are consumed. Moreover, x_8 increases linearly after some waiting period, which is attributed to the production delays of the processes that are involved in production of parts of P_8.

3.5 Inventory Management

Inventory management plays an important role in a push system. Maintaining a larger inventory carries the burden of higher storage and maintenance costs while also increasing the risk of items becoming obsolete over time. As highlighted in the study by Paschalidis [18], the challenge in inventory control lies in striking a balance between the costs associated with *producing* more goods, which raises inventory and management expenses, and *idling*, which may result in stock-outs and unmet customer demands.

Safety stock is a popular strategy used to mitigate challenges posed by disruptions in supply chain operations, such as uncertainties in the supply, or variations in product demand. One of the methods is to fix a replenishment amount that has to be ordered every time the quantity of a supply/raw material goes below a threshold value. Let $x_p^b(t)$ denote the replenishment order of part p at time t, then $x_p^b(t)$ is calculated as follows:

$$x_p^b(t) = \begin{cases} S_p, & \text{if } x_p(t) \leq x_p^s, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where x_p^s is known as the safety stock of supply p. If the stock of part p goes below the threshold, i.e. x_p^s, a replenishment order of S_p amount of part p is made to restock the supply part p. This order, often referred to as back order, usually has a time delay, let up denote it as t_p^b.

It is essential for a supply chain to strike a correct balance between S_p, x_p^s and t_p^b so that the intermediate parts along with the final product do not run out of stock, and the final product is produced at a predictable rate. It is also important to ensure that no intermediate part is getting accumulated because of excess of raw materials and fast intermediate processes. Large stock piles incur costs on maintenance and storage. Strategy to implement inventory control is usually problem specific, and some works can be found in [6, 12, 19].

In the time-bucket simulation of supply chains, the following algorithm can be used to update the system based on replenishment orders, after updating the system based on consumption and production of parts. For a given simulation time T, a time stamp $M > T$, back order time delays $\{t_p^b\}$, safeguard orders $\{S_p\}$, set of supplies/raw materials $\mathbf{S}$ and safety stocks $\{x_p^s\}$, the safety stock method is given in algorithm 5.

3.6 Disruptions in Supply Chains

Disruptions are events in a process that can completely bring down the production of a process for some amount of time. Disruptions are modelled as local events in processes, using two-state continuous Markov chains with states 0 and 1, where 0 represents the state when the process is going through a disruption event, whereas 1 is the normal state of the process. If a process $j \in \{j \mid 1 \leq j \leq J\}$ goes into state 0 at time t, the consumption and production of parts by process j stops completely until the process again goes into the normal state. Hence, if a process j is in a disrupted state at time t, then $\lambda_j(t) = 0$.

The rate of disruption events for a process j is controlled by a matrix Q_j given as follows:

$$Q_j = \begin{pmatrix} q_{j;0,0} & q_{j;0,1} \\ q_{j;1,0} & q_{j;1,1} \end{pmatrix}$$

Algorithm 5 Inventory Control - Safety Stock Algorithm

```
1: **Declare constant** $\bar{t}_p$, $M$, array $S_p$
2: **while** $t < T$ **do**
3:     Update system based on consumption and production algorithms
4:     **for** each $p \in \mathbf{S}$ **do**
5:         Compute the backorder quantity $x_p^b$ using:
6:             if $x_p(t) \leq x_p^s$ :
7:                 $x_p^b = S_p$
8:             else:
9:                 $x_p^b = 0$
10:        **if** $t \geq \bar{t}_p$ **then**
11:            Backorder arrived. Add it to the state vector: $x_p \leftarrow x_p + x_p^b$
12:            Reset the next arrival time: $\bar{t}_p \leftarrow M$
13:        **else if** $t < \bar{t}_p$ and $\bar{t}_p = M$ and $x_p^b > 0$ **then**
14:            Compute the next backorder arrival time: $\bar{t}_p \leftarrow t + t_d^p$
15:        **end if**
16:    **end for**
17: **end while**
```

where $q_{j;0,0} = -q_{j;0,1}$ and $q_{j;1,1} = -q_{j;1,0}$. The transition between states 0 and 1 are is modelled by simulating the times to transition to next state, also known as sojourn times, using the elements of matrix Q. This is done as follows: if the process j transitions to a state a at time t, then the time to transition to next state is sampled from an exponential random variable E_a where $E_a\ exp(1/q_{j;a,b})$ such that $a \neq b$. Thus, higher the value of $q_{j;1,0}$, lesser the time the process takes to transition to the disrupted state, and hence, higher the proportion of time it spends in the disrupted state.

Disruption events in a process j are simulated as follows: the process j starts in an undisrupted state, i.e. state 1 at $t = 0$. We sample w_1, the transition time to the disrupted state 0, from an exponential random variable with rate $q_{j;1,0}$. The process stays in the undisrupted state for w_1 amount of time and at $t = w_1$, the state of process is updated to 0. Now, we sample w_2, the transition time to the normal state 1 from exponential random variable with rate $q_{j;0,1}$, and the state is updated to 1 after spending w_2 amount of time in the disrupted state. These steps are repeated until the time limit of the simulation is reached. All the transition times and state history of the process is recorded, so that it can be incorporated in the simulation of supply chains in the time-bucket method. For a time bucket $(t_n, t_n + \tau]$, let $\delta D_{j,t_n}, 0 \leq \delta D_{j,t_n} \leq 1$ be the proportion of time the process j spends in the undisrupted state, i.e. state 1. The rate of the process j in time bucket $(t_n, t_n + \tau]$ is then calculated as follows:

$$\lambda_{j,t_n} = \lambda_j^{max} \delta D_{j,t_n},$$

where λ_j^{max} is the maximum production rate of process j. The number of occurrences of process j in time bucket $(t_n, t_n + \tau]$ is, thus, sampled as the Poisson random variable with rate equal to $\lambda_{j,t_n} \tau$.

The steady state of the CTMC gives the limiting distribution π of the system, which indicates the

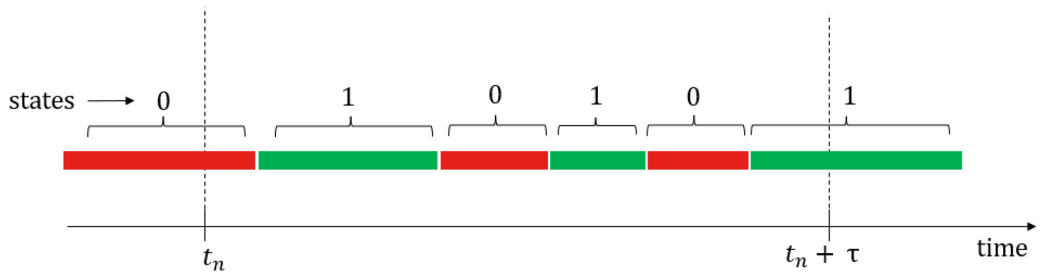


Figure 6: Incorporating disruption events simulation into time-bucket algorithm. The red states represent disrupted state and green state represents normal state of the process.

portions of time spent by the process in each state:

$$\pi = \begin{bmatrix} \frac{q_{j;1,0}}{q_{j;1,0}+q_{j;0,1}} & \frac{q_{j;0,1}}{q_{j;1,0}+q_{j;0,1}} \end{bmatrix}$$

The limiting distribution of disruption events in processes can be estimated from observed data, such as shipping logs, port records etc., which in turn can be used to estimate $q_{j;1,0}$ and $q_{j;0,1}$.

3.6.1 A basic supply chain with disruptions

We present a simple supply chain with one process, that represents import of a supply from one destination to other. The delay of the import process is assumed to be deterministic and equal to $t_1^{min} = t_1^{max} = 10$, while the maximum rate of the process is assumed to be $\lambda_1 = 10$. The initial supply of the system in the simulation is taken to be $x_1(t = 0) = 1000$. Let us consider the following Q matrix to control rate of disruptions in the process:

$$Q_j = \begin{pmatrix} -0.5 & 0.5 \\ 0.1 & -0.1 \end{pmatrix}$$

We perform 500 MC simulations for three scenarios:

1. Simulating supply chain without disruption.
2. Simulating supply chain with disruption rate $q_{1,0} = 0.1$.
3. Simulating supply chain with double disruption rate, i.e. $q_{1,0} = 0.2$.

When $q_{1,0} = 0.1$ and $q_{0,1} = 0.5$, the steady state vector of the system is given by $\pi = [1/6, 5/6]$, which indicates that the process spends 16% of the time in disrupted state and 84% of the time

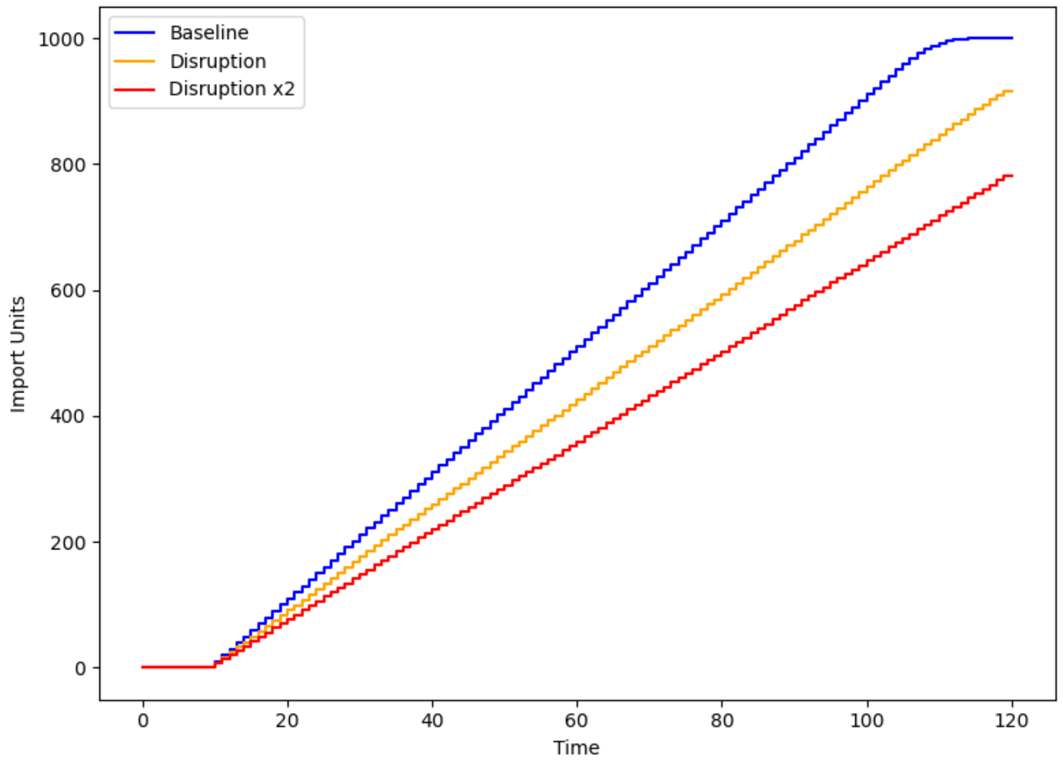


Figure 7: Comparison of supply chain dynamics for 500 simulations without disruption (blue), and with two frequencies of disruption events: a baseline level of disruption (orange) and with disruption events that are twice a frequent (red). For the simulations without disruption (blue) the supply (of 1000 units) is fully consumed by 110 time units, as implied by the flattening of the curve.

in the normal state. On the other hand, when $q_{1,0} = 0.2$, the steady state vector is given by $\pi = [0.28, 0.72]$. Figure 7 shows the aggregation of number of imports of 500 MC simulations for the three scenarios described above. It is evident from the plot that there is a significant decrease in rate of imports when disruptions are incorporated in the supply chains. Moreover, increasing the rate of disruptions further decreases the rate of the imports.

References

[1] David F. Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *Department of Mathematics, University of Wisconsin-Madison*, 2007. Received 20 June 2007; accepted 26 September 2007; published online 6 December 2007; publisher error corrected 28 January 2008.

[2] Basil Bayati, Philippe Chatelain, and Petros Koumoutsakos. D-leaping: Accelerating stochastic simulation algorithms for reactions with delays. *Unknown*, Unknown. Author links open overlay panel.

[3] Nai-Yuan Chiang, Yiqing Lin, and Quan Long. Efficient propagation of uncertainties in manufacturing supply chains: Time buckets, l-leap, and multilevel monte carlo methods. *Operations Research Perspectives*, Unknown. Journal homepage: `www.elsevier.com/locate/orp`.

[4] Sunil Chopra. *Supply Chain Management: Strategy, Planning, and Operation*. Pearson, Upper Saddle River, NJ, fifth edition edition, 2012. Kellogg School of Management.

[5] Joaquim Duarte Oliveira Vlad Filippov Contributors: Deepak Mavatoor, Peter Lidell. 2023 supply chain outlook: Expert advice on thriving in times of change. Report, 2023.

[6] JSR Daniel and C. Rajendran. A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research*, 12:101–127, 2005.

[7] L.A. Deleris and F. Erhun. Risk management in supply networks using monte-carlo simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 7 pp.–, 2005.

[8] S. Ethier and T. Kurtz. *Markov Processes: Characterization and Convergence (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2nd edition, 2005.

[9] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[10] Mike B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24:259–328, 2015.

[11] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[12] Ignacio E. Grossmann. Enterprise-wide optimization: a new frontier in process systems engineering. *AIChE Journal*, 51:1846–1857, 2005.

[13] Kirsty Hey, Hiroshi Momiji, Karen Featherstone, Julian Davis, Mike White, David Rand, and Bärbel Finkenstädt. Inference for a transcriptional stochastic switch model from single-cell imaging data. 2021.

[14] June Young Jung, Gary Blau, Joseph F. Pekny, Gintaras V. Reklaitis, and David Eversdyk. A simulation based optimization approach to supply chain management under demand

uncertainty. *Computers Chemical Engineering*, 28(10):2087–2106, 2004. Special Issue for Professor Arthur W. Westerberg.

[15] H. Koeppl, D. Densmore, G. Setti, and M. di Bernardo, editors. *Design Analysis of Biomolecular Circuits*. Springer, 2011.

[16] T. G. Kurtz. *Approximation of Population Processes (CBMS-NSF Regional Conference Series in Applied Mathematics)*. Society for Industrial and Applied Mathematics, 1987.

[17] Alvaro Moraes. *Simulation and Statistical Inference of Stochastic Reaction Networks with Applications to Epidemic Models*. Doctoral dissertation, Name of the University, 2023.

[18] Ioannis Ch. Paschalidis and Yong Liu. Large deviations-based asymptotics for inventory control in supply chains. *Journal of Optimization Theory and Applications*, 158(2):463–486, 2013.

[19] Jeremy F. Shapiro. *Modeling the Supply Chain*. Cengage Learning, 2 edition, 2006.

[20] G. George Yin and Qing Zhang. *Continuous-Time Markov Chains and Applications: A Two-Time-Scale Approach*, volume 37 of *Stochastic Modelling and Applied Probability*. Springer, 2006. New chapters added on backward equations and LQG control problems.