KADİR HAS UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF ENGINEERING AND NATURAL SCIENCES

DEMAND CLASSIFICATION FOR SPARE PARTS SUPPLY CHAINS IN THE PRESENCE OF THREE-DIMENSIONAL PRINTERS

ZÜLAL İŞLER

MASTER OF SCIENCE THESIS

ISTANBUL, JANUARY, 2022

DEMAND CLASSIFICATION FOR SPARE PARTS SUPPLY CHAINS IN THE PRESENCE OF THREE-DIMENSIONAL PRINTERS

ZÜLAL İŞLER

A thesis submitted to
the School of Graduate Studies of Kadir Has University
in partial fulfillment of the requirements for the degree of
Master of Science in
Industrial Engineering

Istanbul, January,2022

APPROVAL

This thesis titled DEMAND CLASSIFICATION FOR SPARE PARTS SUPPLY CHAINS IN THE PRESENCE OF THREE-DIMENSIONAL PRINTERS submitted by ZÜLAL İŞLER, in partial fulfillment of the requirements for the degree of Master of Science in Industrial Engineering, Kadir Has University by:

Asst. Prof. Mustafa Hekimoğlu (Advisor) …….
(Kadir Has University)

Assoc. Prof. Deniz Karlı …….
(Işık University)

Asst. Prof. Esra Ağca Aktunç …….
(Kadir Has University)

I confirm that the signature above belongs to the aforementioned faculty members.

_____

Prof. Dr. Mehmet Timur Aydemir

Director of the School of Graduate Studies

APPROVAL DATE: Day/Month/Year

DECLERATION ON RESEARCH ETHICS AND

PUBLISHING METHODS

I, ZÜLAL İŞLER; hereby declare

- that this Master of Science Thesis that I have submitted is entirely my own work and I have cited and referenced all material and results that are not my own in accordance with the rules;
- that this Master of Science Thesis does not contain any material from any research submitted or accepted to obtain a degree or diploma at another educational institution;
- and that I commit and undertake to follow the "Kadir Has University Academic Codes of Conduct" prepared in accordance with the "Higher Education Council Codes of Conduct".

In addition, I acknowledge that any claim of irregularity that may arise in relation to this work will result in a disciplinary action in accordance with the university legislation.

Zülal İşler

_____

27.01.2022

To the great power within me…

ACKNOWLEDGEMENT

I started my master September 2019 and studied for this dissertation for almost three years at Kadir Has University. I want to express my gratitude to those who contributed and helped me grow.

First of all, I would like to thank my thesis advisor, Assist. Prof. Mustafa Hekimoğlu, for always presenting new ideas and insights and sharing his experiences in the scientific research process and supporting me.

Furthermore, I would like to thank my professors Meltem Kıygı Çallı and Saadet Çetinkaya, who supported me every time I questioned myself and were always there for me. I am grateful to Meltem Kıygı Çallı for allowing me to do my first academic studies.

Last but not least, I am thankful to my teammates that I worked with during the project for their support and contribution to my learning process.

In addition to these contributions, I am grateful to my family, who stood by me in every decision I made, supported me and made me feel their love at all times.

Zülal İşler

Istanbul, January 2022

DEMAND CLASSIFICATION FOR SPARE PARTS SUPPLY CHAINS IN THE PRESENCE OF THREE-DIMENSIONAL PRINTERS

ABSTRACT

Three-dimensional printers (3DPs) are currently the source of the supply chain and are used to ensure spare parts supply in case of shortages. However, the reliability of the part produced in 3DP is lower than the original part supplied by the original equipment manufacturer (OEM). Failure of parts creates demand and the failure probability of original and printed part is different than each other. Thus, knowing the total demand distribution have great importance in optimizing the order quantity given to the OEM in the presence of 3DPs. In this study, the demand distribution of system failures has been determined by using the distribution classification methods put forward by Ord (1967) and Adan et al. (1995). In line with the results, according to study of Ord(1967), demand distribution is found as Hypergeometric and Binomial distribution. Discrete distribution family of Adan et al. (1995) gives Binomial distribution for the system demand. All results are tested with chi-square test and likelihood ratio test.

Keywords: 3D printers, supply chain, demand classification, distribution selection, Ord hypergeometric distribution family, Adan discrete distribution family

ÜÇ BOYUTLU YAZICILARIN VARLIĞINDA YEDEK PARÇA TEDARİK ZİNCİRLERİNDE TALEP SINIFLANDIRMASI

ÖZET

Üç boyutlu yazıcılar (ÜBY) tedarik zincirinin kaynağıdır ve eksiklik durumunda yedek parça teminini sağlamak için kullanılmaktadır. Ancak, ÜBY'de üretilen parçanın güvenilirliği, orijinal ekipman üreticisi tarafından sağlanan orijinal parçadan daha düşüktür. Sistemde bulunan parçaların bozulması talep yaratmaktadır ve orijinal ve yazılı parçaların bozulma olasılığı birbirinden farklıdır. Bu nedenle, ÜBY'nin varlığında original ekipman üreticisine verilen sipariş miktarının optimize edilmesinde toplam talep dağılımının bilinmesi büyük önem taşımaktadır. Bu çalışmada, Ord (1967) ve Adan vd. (1995) tarafından ortaya konulan dağılım sınıflandırma yöntemleri kullanılarak sistem bozulmalarının talep dağılımı belirlenmiştir. Sonuçlar doğrultusunda, Ord'un (1967) sınıflandırma yöntemine göre talep dağılımı Hipergeometrik ve Binom dağılımı olarak bulunmuştur. Adan vd. (1995) tarafından verilen ayrık dağıtım ailesine göre ise, sistemin talebi Binom dağılımını takipe etmektedir. Tüm sonuçlar ki-kare testi ve olabilirlik oranı testi ile test edilmiştir.

Anahtar Sözcükler: Üç boyutlu yazıcı, tedarik zinciri, talep sınıflandırması, dağılım seçimi, Ord hipergeometrik dağılım ailesi, Adan ayrık dağılım ailesi

TABLE of CONTENTS

LIST of FIGURES

LIST of TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

3DP: Three-dimensional printer

AM: Additive manufacturing

OEM: Original Equipment Manufacturing

1. INTRODUCTION

Manufacturing equipment, millions of dollars in capital assets, are often critical in sustaining business core processes, so asset owners invest heavily in managing system availability. Many organizations operate complex supply chains service with local spare parts inventory to protect themselves against their prolonged system shortages. However, spare parts inventory often means a considerable investment due to the number of parts involved, prices, and long production times. For example, the US coast guard holds more than 60,000 spare parts in stock, having a total inventory value of more than $700 million (Deshpande et al., 2006). Cost and the value of keeping inventory are more significant for remote locations, such as military equipment, oceangoing transport vessels, and mining equipment (Westerweel et al., 2021). The original equipment manufacturer (OEM) conducts the replacement of spare parts, which creates excellent dependence on OEM. However, three-dimensional printers enable companies with capital assets to produce their spare parts using additive manufacturing; instead of supplying from OEM.

Additive manufacturing (AM) is a digital technology which is used to produce physical objects layer by layer from a three-dimensional computer-aided file (Thomas, 2016). The technology was introduced as rapid prototyping and three-dimensional printer (3DP) for producing rough physical prototypes of the final products (Khajavi et al. 2014). Additive manufacturing is currently used in prototyping, but it is also used in producing functional parts. AM can be used in manufacturing operations and supply chains to exploit AM's benefits, which are speed, quality, materials range, and affordability (Thomas, 2016; Strong et al., 2018). AM brings the advantage of producing spare parts in remote geographic areas where spare part stock replenishment time is long (Westerweel et al., 2021), enabling companies to produce spare parts when the need arises. Thus, inventory cost decreases, and asset availability increases. However, one of the disadvantages of using AM is the reliability of a produced part (Kruth et al. 1998). Part produced with 3DP has lower reliability than those provided by OEM (Westerweel et al. 2021), leading to the quality difference between printed part and original part.

The spare parts produced with 3DP become a temporary solution for the system. OEM may not supply the spare parts immediately due to lead time. In this case, 3DP becomes the second supplier on hand and supplies the spare parts if there is no spare part inventory. Inventory management systems in which two supply sources are used simultaneously are called dual sourcing systems. In this study, from the perspective of the inventory planner, the failures of spare parts are defined as the demand for spare parts. For a capital product, the demand for spare parts will be fulfilled either with 3DP or inventory on hand which is supplied from OEM. The total demand of the system develops from failures of printed and original parts. In this case, the presence of original and printed parts in the system changes demand distribution. How much to order and when to order decisions are essential for an inventory control system; thus, knowing the demand distribution becomes crucial for the forecasts and ordering decisions to OEM.

In order to determine the demand distribution of the system, convolution of several parts distributions might be required. The characterization of failures in a finite machine environment when there are two different qualities in the system is defined as the Bernoulli process by Westerweel et al. (2021). In this study, the demand comes from Bernoulli failures. The demand distribution of the printed part is assumed to be Binomial distribution, and the demand distribution of the original part is assumed as Binomial distribution. Although it is possible to obtain the convolution of two discrete random variables with non-identical distributions by numerical methods, it is not known exactly which of the existing theoretical distributions can meet this sum. Approximate solutions for finding the exact distribution of convolution have been proposed in the literature (Norman et al., 2005; Jolayemi, 1992).

Besides, for many parts, the probability of failures increases with the age of the part. Therefore, the failure distribution of these parts has a memory, and failures occur in more than one stage (Hekimoğlu, 2015). In this case, the failure probability of a newly installed part will increase during the time. In other words, the probability of failure in the period in which the part is located is higher than the probability of failure in the previous period. At that time, the original part demand in the system in any period will only depend on the number of original parts in the previous period. If there is no inventory on hand, the printed part produced by a 3DP will be used instead of the original piece to keep the system running.

In this study, we answer the following research questions: 1) How does the demand distribution in the system change in the presence of 3DP? 2) How does the demand distribution change if the parts are written with an adjustable quality level on a 3DP? 3) How does the ageing of original parts affect the system? 4) How does the optimum order quantity for the original and the printed part change in the presence of a 3DP?

To answer the first two research questions, Studies of Ord (1967) and Adan et al. (1995) will be used. Ord (1967) uses Pearson's differential equation and develops a system of discrete distributions. The choice of distribution depends on only the first three central moments. Ord (1967) gives a two-dimensional plane for the distribution selection for a given first-three central moments. In addition to this, Adan et al. (1995) provide distribution selection based on the first two central moments. The distribution selection of Adan et al. (1995) is based on Erlang distributions. Accordingly, random variables for demand belonging to a countable set will be handled in this study. Since both studies use moments, the first-three moments of convolution are calculated. Then, a parameter set is generated, and distribution selection is conducted for a given parameter set. The chi-square test and likelihood ratio test are handled to validate the results.

The number of suppliers can be reduced drastically due to the flexibility which 3DP can deliver. The only supplier for the 3DP process in the production phase will be the material supplier (Chan et al., 2018). Since 3DP enables companies to keep less inventory of spare parts, optimal order quantity from OEM might change. Therefore, a mathematical model is created for a dual-sourcing system with both printed part and original part supply. Change in an optimal order quantity of original part and printed part is calculated with numerical experiments, answering the third research question. The findings of this study indicate that the convolution of two random variables has Binomial distribution or Hypergeometric distribution.

In this study, distribution selection of spare parts demand and change in optimal order quantity are provided in the existence of 3DP. The literature review for the study is described in Section 2. The mathematical model, moments of convolution, and parameter estimation is given in Section 3. After the results are presented alongside with discussions in Section 3, concluding remarks follow in Section 4.

2. LITERATURE REVIEW

3DP is commonly used in many areas such as e-commerce, online platforms, supply chain, manufacturing (Khajavi et al. 2014, Rayna et al. 2015). Despite of this, the current form of 3DP cannot replace entirely traditional manufacturing methods (Holweg, 2015). 3DP technology may offer quick customization and a make-to-order opportunity for businesses (Gao et al., 2015). The use of 3DP might be expensive for the industries; however, it lessens the cost of holding and lack of inventory (Chan et al. 2018, Liu et al., 2014). In the literature, some studies examine the inventory control strategy in the presence of 3DP (Westerweel et al. 2021, Song and Zhang, 2020). Westerweel et al. (2021) show that using 3DP on remote locations leads to significant operational cost savings due to less inventory in stock and increased asset availability.

One of the points in the use of 3DP in spare part production is the quality difference. Printed spare parts fail earlier than original spare parts (Hekimoğlu and Ulutan, 2020). Laser polishing increases the reliability of the part (Ma et al., 2017), which reduces the quality difference. Laser polishing density increases the durability of the part, and failure probabilities decrease (Hekimoğlu and Ulutan, 2020). Westerweel et al. (2021) assume printed part failure probability is equal to the original part. Knofius et al. (2019) discuss a system where a single part quality is written by multiple machines. This study considers the quality difference of printed parts for a single printer. Also, it is considered that the 3DPs write parts at different quality levels.

Distribution selection of convolutions and its application are used in many research. Guerrero-Salazar and Yevjevich (1975) examine droughts as a stationary and periodic-stochastic process; in order to determine droughts distribution, they use Ord (1967) classification. Vitanov et al. (2020) consider the motion of substance in a finite channel. They prove the obtained class of distributions contains all truncated discrete probability distributions of discrete random variables using Ord (1967). Also, Ord (1967) study leads many other distribution classification studies (Robertson et al. 2013, Adan et al. 1995, Korwar 1989).

Adan et al. (1995) use the first two moments to fit discrete distributions. Janssen et al. (1998) study on inventory model, and they model the demand using the study by Adan et

al. (1995). De Smidt-Destombes et al. (2006) examines maintenance policy under the decision between spare part inventory and repair capacity, and they compare computational analysis and simple approximation of Adan et al. (1995). Sleptchenko et al. (2002) approximates pipeline inventory distribution using Adan et al. (1995) and finds the first two moments of the backorders. Van Donselaar and Broekmeulen (2012) use fitting discrete distribution of Adan et al. (1995) to find demand distribution and conduct simulation test. Syntetos et al. (2011) study parametric assumptions towards stock control, relying on demand distribution assumptions and conducting goodness of fit tests. In this study, distribution families from Ord (1967) and Adan et al. (1995) are compared, and their validation is tested for spare parts inventory control system for printed and original part demand.

In all inventory control literature, including those related to 3DP, no studies found in which quality and inventory are dynamically controlled and optimized together. Song and Zhang (2020) assume that more than one part was produced on the same printer, but only one of the printed or original part inventory strategies was valid and does not consider dual sourcing. Westerweel et al. (2021) do not consider the quality difference in their study, but they use dynamic system control. This study contributes to the literature in terms of finding the distribution of spare parts demands in the use of 3DP and expressing the numerical results of dynamic programming, which considers quality difference and dual sourcing.

3. MATHEMATICAL MODEL

In a high-cost capital environment or a production line, spare part management of working machines are critical to sustaining business processes. In case of failures in a part of capital machines, OEM may not supply the part immediately, and 3DP enables companies to make on-demand production. In addition, printed parts fail faster than the original parts, but 3D printers allow production at different quality levels. The use of different quality levels changes the distribution of spare parts demand. In order to figure out the distribution, Ord (1967) and Adan et al. (1995) discrete distribution families are handled using the central moments. In this study, three cases are considered. First, there is one printer quality and OEM supplies. Second, OEM supply continues, and there are different quality levels (k) for the printed part. Third, there is one printer quality and original part ages in different levels (s) and failures observed at s^{th} level. Conditional distributions are considered based on Bernoulli failures.

Considering the convolution of printed and original part demands, the first three moments will be calculated in the following parts. Then, parameter estimation of Ord (1967) will be made, and the distribution selection algorithm of Adan et al. (1995) will be given. Further, limiting probabilities and moments of ageing system will be given. In the end, simulations will be conducted for generated parameter sets. In section 4, these simulation results will be discussed.

3.1. Moments of Spare Parts Demand

I consider a system with N identical machines in a discrete-time setting. In these machines, a spare part fails with probability p at every period. When an original spare part is broken, it is replaced with an original part if there is enough inventory. Otherwise, a part is printed and installed. It is assumed that each printed part fails with probability $\tilde{p}$ and $p < \tilde{p}$. The number of original parts is m_0 out of N machines and the number of printed parts is m_1 out of N, $m_0 + m_1 = N$. It is assumed that the demand of the original parts (D_o) follow the $Binomial(m_0, p)$ distribution. Similarly, the failures of the printed part (D_p) follow $Binomial(m_1, \tilde{p})$ distribution. The total demand for spare parts in a period consists of the convolution of two Binomial distributions.

Although, it is possible to obtain the convolution of two non-identical Binomial distributions by numerical experiments. It is also essential to know the theoretical distribution that approximates the convolution. Approximate solutions have been proposed in the literature for calculating central moments (Norman et al., 2005; Jolayemi, 1992). The two most relevant approaches are Kolmogorov's method and an adaptation of a distribution from the Pearson family (Liu and Quertermous, 2017). This study focuses on the Pearson approach. The most relevant reference on Pearson approximation addresses a characterization and more precisely selection criteria with the first three moments of some laws of random variables, whose density f follows the following Pearson differential equation $\frac{df}{dx} = \frac{(a-x)f}{b_0+b_1x+b_2x^2}$.

The application of the Pearson differential equation given above with discrete distribution has been discussed by Ord (1967). To provide an example for Pearson's law fit of discrete distributions, let X be a random variable defined in the probability space (Ω, F, P) and the parameter $B(n,p)$ follows Binomial distribution. In this case, $\forall k \in [0,n], P(X = k) = \binom{n}{k} p^k (1-p)^{(n-k)}$. Calculations show that the density follows the equation below, which is close to the Pearson equation (Katz, 1948).

$$P(X = k+1) - P(X = k) = \left(\frac{(n-k)p}{(k+1)(1-p)} - 1\right) P(X = k) \tag{3.1}$$

Let (Ω, F, P) be probability space, and X be a random variable following Binomial distribution, denoted by $Binom(n,p)$.

Proposition 1. For all $r \in \mathbb{N}, X \in (\Omega, F, P)$, and $E[X^r] = \sum_{k=0}^{r} \frac{n!}{(n-k)!} S(r,k) p^k$, with $S(r,k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^r$

Proof. The fact that, for all $r \in \mathbb{N}, X \in (\Omega, F, P)$ is trivial. Later, the proof of the equations given in the proposition is issued by Gupta and Singh (1981) and can be found using factorial moments.

Using *Proposition 1*, the first three central moments of X are given by:

$$E[X] = np \tag{3.2}$$

$$E[(X - E[X])^2] = np(1-p) \tag{3.3}$$

$$E[(X - E[X])^3] = np(1-p)(1-2p) \tag{3.4}$$

If X and Y are two independent random variables defined on the same probability space (Ω, F, P) with discrete finite support $[0, n]$ and $[0, m]$ respectively. Then, the random variable $Z = X + Y$, has the following distribution, called convolution:

$$\forall k \in [0, m+n], P(X+Y=k) = \sum_{0 \le j < k} P(X+Y=k, Y=j) \tag{3.5}$$

$$= \sum_{0 \le j < k} P(X=k-j)P(Y=j) \tag{3.6}$$

Proposition 2. Let X and Y be two independent random variables defined on a probability space (Ω, F, P) and assume that X and Y are independent, with support in $\mathbb{N}$. Let $\forall r \in \mathbb{N}, X \in (\Omega, F, P)$ and $Y \in (\Omega, F, P)$. For, $\forall (k, p) \in \mathbb{N}$, then $E[X^k Y^p] = E[X^k]E[Y^p]$.

Proposition 3. Let $(X_1, \dots, X_n)$ be n random variables defined on a probability space (Ω, F, P). Assume that $(X_1, \dots, X_n)$ are independent and $\forall k \in [1, \dots, n]$, $X_k \in (\Omega, F, P)$.

Then we have:

$$E\left[\left(\sum_{i=1}^{n} X_i - E\left[\sum_{i=1}^{n} X_i\right]\right)^2\right] = \sum_{i=1}^{n} E\left[\left(X_i - E(X_i)\right)^2\right] \tag{3.7}$$

$$E\left[\left(\sum_{i=1}^{n} X_i - E\left[\sum_{i=1}^{n} X_i\right]\right)^3\right] = \sum_{i=1}^{n} E\left[\left(X_i - E(X_i)\right)^3\right] \tag{3.8}$$

Proof. We have used the multinomial theorem:

$$E\left[\left(\sum_{i=1}^{n} X_i - E\left[\sum_{i=1}^{n} X_i\right]\right)^3\right] = E\left[\left(\sum_{i=1}^{n} (X_i - E[X_i])\right)^3\right] \tag{3.9}$$

$$= E\left[\sum_{i=1}^{n} 3! \prod_{i=1}^{k} \frac{1}{k_i!} (X_i - E[X_i])^{k_i}\right] \tag{3.10}$$

$$= \sum_{i=1}^{n} E\left[\left(X_i - E(X_i)\right)^3\right] + E\left[\sum_{i=1}^{n} 3! \prod_{i=1}^{k} \frac{1}{k_i!} (X_i - E[X_i])^{k_i}\right] \tag{3.11}$$

Therefore, the independence assumption is used and $E\left[X_i - E[X_i]\right] = 0, \forall i \in [1, \dots, n]$

$$E\left[\sum_{i=1}^{n} 3! \prod_{i=1}^{k} \frac{1}{k_i!} (X_i - E[X_i])^{k_i}\right] = 0 \tag{3.12}$$

■

Proposition 4. $(X_1, \dots, X_n)$ n independent random variables defined on a probability space (Ω, F, P). Assume that, $\forall i \in \{1, \dots, n\}, X_i \sim Binom(n_i, p_i)$ and $p_i \in (0,1)$. Then the three first central moments are given by:

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} n_i\, p_i \tag{3.13}$$

$$E\left[(\sum_{i=1}^{n} X_i - E[\sum_{i=1}^{n} X_i])^2\right] = \sum_{i=1}^{n} n_i\, p_i(1 - p_i) \tag{3.14}$$

$$E\left[(\sum_{i=1}^{n} X_i - E[\sum_{i=1}^{n} X_i])^3\right] = \sum_{i=1}^{n} n_i\, p_i(1 - p_i)(1 - 2p_i) \tag{3.15}$$

Proof. It is a straight consequence of Proposition 1 and 3.

In the next step, using the moments discussed above, I will find out which distribution the Binomial convolutions evolve for the Ord (1967) and Adan et al. (1995) distribution families and explain them. Also, I will make parameter estimations for the distributions obtained.

Using the first three central moments of the Binomial convolutions given above, the distribution that provides the best approximation to the total demand for spare parts in a machine park consisting of parts of different quality will be selected.

3.2. Parameter Estimation of Distribution

The distribution of the printed and original parts in the system will be found using the first three central moments. But in this case, the parameters of the obtained distribution are unknown. For this reason, parameter estimation will be made at this stage of the study by using the moment approach.

3.2.1. Ord's hypergeometric distribution family

Ord (1967) showed that the discrete distributions hypergeometric distribution family fit the Pearson equation. The family of hypergeometric distributions includes commonly known distributions such as Binomial, Negative Binomial, Poisson, Geometric distributions; besides, compound distributions such as Beta-Binomial and Beta-Pascal. The Beta–Binomial distribution is a mixture of binomial distributions, with the binomial probability parameter (p) having a beta distribution. Beta-Pascal distribution arises as a beta mixture of negative binomial distributions, also called beta-negative binomial distribution (Johnson et al., 2005). Theoretical distributions in this large family are classified based on their first three central moments (Ord, 1967). Define μ_r as the r^{th} central moment of a distribution. Moment generating function for binomial coefficients is given as $\mu_r = E[(X - E[X])^r] \mid= \sum_{j=0}^{r} (-1)^j \binom{r}{j} \mu'_{r-j} \mu^j$ (Johnson et al. 2005).

Figure 1. Ord (1967) Discrete Distribution Family

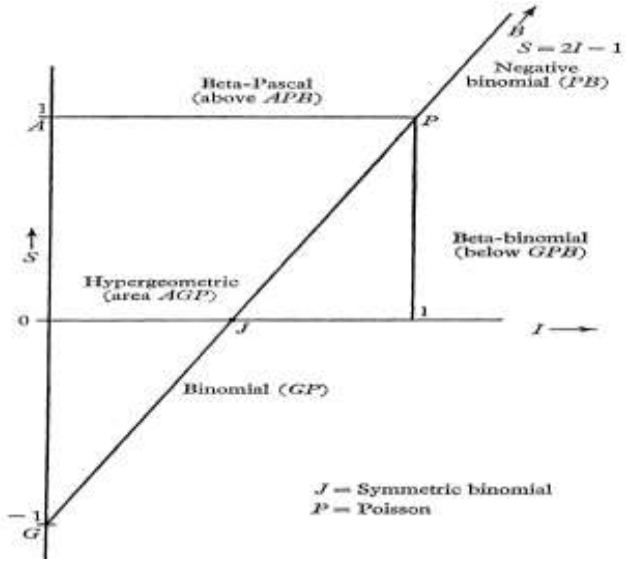


Figure 1 shows Ord (1967) discrete distribution family. Horizontal line indicates I which is equal to $\frac{\mu_2}{\mu_1}$ and vertical line denotes S which is equal to $\frac{\mu_3}{\mu_2}$. All the points in the GP line in Figure 1 give Binomial distribution. For a given parameter set, points of the AGP area belong to Hypergeometric distribution. The equation of the GB line is $S = 2I - 1$.

The characterization of the demand distribution is made using the discrete distribution family created by Ord (1967) for the number of original and printed parts in the system. It is assumed that the failures in the system follow Binomial distribution. In this case, the system's total demand for spare parts (D) consists of the convolution of the Binomial distribution equal to the sum of the original parts (D_o) and printed parts (D_p), $D = D_o + D_p$. The simulation results show that for a given parameter set, the convolution of the two Binomial distributions belongs to the hypergeometric distribution or the Binomial distribution. For this reason, we need parameter estimation only for Hypergeometric and Binomial distribution.

The hypergeometric distribution has its types, and the general hypergeometric distribution provides these types with certain conditions (Johnson et al., 2005). Therefore, parameter estimation for the hypergeometric distribution is more challenging than the Binomial distribution. The probability mass function of general hypergeometric distribution is $Pr[X = x] = \binom{a}{x}\binom{b}{n-x}/\binom{a+b}{n}$. a is the number of success states in the population, n is the number of draws, x is the number of observed successes, and b is the number of failed states in the population. Ord (1967) uses the classical hypergeometric distribution in his study. The probability mass function of classical hypergeometric distribution is $Pr[X = x] = \binom{Np}{x}\binom{N-Np}{n-x}/\binom{N}{n}$. N represents the population, and p is the probability of success. For this reason, parameter estimates of the classical hypergeometric distribution were made using the first three central moment formulas given by Ord (1967). Equation 3.16 – 3.18 shows the first three central moments Ord (1967) provided.

$$\mu_1 = \frac{ab}{e} \tag{3.16}$$

$$\mu_2 = \frac{ab(a+e)(b+e)}{e^2(e-1)} \tag{3.17}$$

$$\mu_3 = \frac{ab(a+e)(b+e)(2a+e)(2b+e)}{e^3(e-1)(e-2)} \tag{3.18}$$

Ord (1967) states in his study that $a = -m,\ b = -Nk,\ e = N$ should be written for the hypergeometric distribution. When we substitute the given expressions, the first three moments for the classical hypergeometric distribution are obtained in Equation 3.19 – 3.21.

$$\mu_1 = mk \tag{3.19}$$

$$\mu_2 = \mu_1 \frac{(N-m)(1-k)}{N-1} \tag{3.20}$$

$$\mu_3 = \mu_2 \frac{(N-2m)(1-2k)}{N-2} \tag{3.21}$$

Parameter estimation is made using the moment approach. For this reason, the moments of the classical hypergeometric distribution given above are used. Taking $\rho_1 = \frac{\mu_2}{\mu_1}$ and $\rho_2 = \frac{\mu_3}{\mu_2}$ gives $\rho_1 = \frac{(N-m)(1-k)}{N-1}$ and $\rho_2 = \frac{(N-2m)(1-2k)}{N-2}$. Parameter estimation formulas for the hypergeometric distribution are given in Equation 3.22 – 3.24.

$$N = \frac{2\rho_2 - 2\rho_1 + 2\mu_1}{\rho_2 - 2\rho_1 + 1} \tag{3.22}$$

$$0 = m^2 - m(N + \mu_1 - N\rho_1 + \rho_1) + N\mu_1 \tag{3.23}$$

$$k = \frac{\mu_1}{n} \tag{3.24}$$

In the following parts of the study, the parameter space will be created for simulations, and the moments at each point of the parameter space will be calculated. R program will be used for the simulations. Johnson et al. (2005) give the general hypergeometric distribution conditions that satisfy the different hypergeometric distribution types. Therefore, the classical hypergeometric distribution is transformed into the general hypergeometric distribution according to the condition given by Johnson et al. (2005). Equation 3.25 – 3.27 gives the first three moments for the general hypergeometric distribution.

$$\mu_1 = E[X] = \frac{na}{a+b} \tag{3.25}$$

$$\mu_2 = Var[X] = \frac{nab(a+b-n)}{(a+b)^2(a+b-1)} \tag{3.26}$$

$$\mu_3 = \frac{\mu_2(b-a)(a+b-2n)}{(a+b)(a+b-2)} \tag{3.27}$$

The general hypergeometric distribution is equal to the classical hypergeometric distribution for the following conditions:

$$n-b-1<0,\ n \text{ integer, and } \ 0<n-1\ <a$$

$$n-b-1<0,\ a \text{ integer, and } \ 0<a-1\ <n$$

If we write $b = N - Np$ and $a = Np$, the first three moments of the general hypergeometric distribution will equal the first three moments of the classical hypergeometric distribution. Equation 3.28-3.30 reveals the connection between the classical hypergeometric distribution and the general hypergeometric distribution (Johnson et al., 2005).

$$\mu_1 = \frac{n.Np}{N} = np \tag{3.28}$$

$$\mu_2 = \frac{n.Np.(N-Np)(N-n)}{N^2(N-1)} = \frac{n.Np.N(1-p).(N-n)}{N^2(N-1)} = \frac{np(1-p)(N-n)}{N-1} \tag{3.29}$$

$$\mu_3 = \mu_2 \frac{(N-Np-Np)(Np+N-Np-2n)}{(Np+N-Np-2)} = \mu_2 \frac{(N-2n)(1-2p)}{N-2} \tag{3.30}$$

3.2.2. Adan's discrete distribution family

In addition to the hypergeometric distribution family, a distribution selection based on the mixture of Erlang distributions was also proposed by Adan et al. (1995). This method is based on selecting one of the Binomial mixtures, Geometric mixtures, Poisson, and Negative Binomial mixtures distributions using the first two moments of a random variable. The Adan et al. (1995) method is commonly used in inventory control literature

to select spare parts demand distribution (Axsater, 2006). Below is the demand and parameter estimation are given by Adan et al. (1995).

$$a = \frac{c_x^2 - 1}{\mu_1}, \qquad and \quad c_x = \frac{\sqrt{\mu_2}}{\mu_1} \tag{3.31}$$

Distribution selection is based on the following conditions: Y is a random variable that comes from convolution, and k is the number of trials. Also, parameter estimations are given below by Adan et al. (1995).

1. If $\frac{-1}{k} \leq a \leq \frac{-1}{k+1}$ for certain $k = 1,2,3,\ldots$ then,

$$Y = \begin{cases} Binom(k,p)\ w.p.\ \ q, \\ Binom(k+1,p)\ w.p.\ \ 1-q. \end{cases} \tag{3.32}$$

where,

$$q = \frac{1 + a(1+k) + \sqrt{-ak(1+k) - k}}{a+1} \tag{3.33}$$

$$p = \frac{\mu_1}{k + 1 - q} \tag{3.34}$$

2. If $a = 0$, then $Y = Pois(\lambda)$ with $\lambda = \mu_1$
3. If $\frac{1}{k+1} \leq a \leq \frac{1}{k}$ for certain $k = 1,2,3,\ldots$ then,

$$Y = \begin{cases} NBinom(k,p) & w.p. \quad q, \\ NBinom(k+1,p) & w.p. \quad 1-q \end{cases} \tag{3.35}$$

where,

$$q = \frac{a(1+k) - \sqrt{-ak(1+k) - k}}{1+a} \tag{3.36}$$

$$p = \frac{\mu_1}{k + 1 - q + \mu_1} \tag{3.37}$$

4. If $a \geq 1$, then

$$Y = \begin{cases} Geo(p_1) & w.p.\ q_1, \\ Geo(p_2) & w.p.\ q_2 \end{cases} \tag{3.38}$$

where,

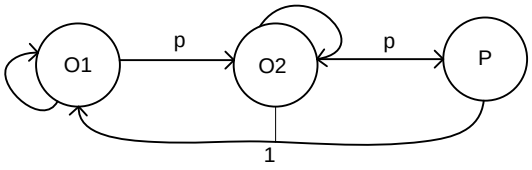
$$p_1 = \frac{\mu_1\left[1 + a + \sqrt{a^2 - 1}\right]}{2 + \mu_1\left[1 + a + \sqrt{a^2 - 1}\right]}, p_2 = \frac{\mu_1\left[1 + a - \sqrt{a^2 - 1}\right]}{2 + \mu_1\left[1 + a - \sqrt{a^2 - 1}\right]} \quad (3.39)$$

$$q_1 = \frac{1}{1 + a + \sqrt{a^2 - 1}}. q_2 = \frac{1}{1 + a - \sqrt{a^2 - 1}} \quad (3.40)$$

3.3. Ageing of Parts

For many parts, the probability of failure increases with the age of the part. This can be interpreted as failure distribution of these parts has a memory, and the failure occurs in more than one stage. To model multi-stage components as opposed to Bernoulli failures in a production facility with N identical machines, we define a Markov chain. To make the model simple, there are one original part and one printed part in the Markov chain and the original part ages at two levels. Figure 2 shows the transition diagram of Markov chain. O1 and O2 indicate the original part at ageing level 1 and 2, respectively. P indicates the printed part. The demand of the original part will occur at ageing level 2 where O2 fails. When O2 fails, the original part will be replaced with a printed part. When P fails, the printed part will be replaced with an original part. As a rule of the Markov chain, conditional expectation consists of the probability of aging of the part in the next period and the part remaining in the same condition in the next period.

Figure 2. Transition Diagram



The matrix below represents the transition probability matrix for the given transition diagram. States are respectively ordered as (2,0,0), (1,1,0), (0,2,0), (1,0,1), (0,1,1), and (0,0,2).

$$P = \begin{bmatrix} (1-p)^2 & 2p(1-p) & p^2 & 0 & 0 & 0 \\ 0 & (1-p)^2 & p(1-p) & p(1-p) & p^2 & 0 \\ 0 & 0 & (1-p)^2 & 0 & 2p(1-p) & p^2 \\ 1-p & p & 0 & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In order to compute the first three moments of the Markov chain, limiting probabilities are calculated.

Lemma 1. Limiting probability of given states are $\pi_0 = \frac{1}{(p+2)^2}$, $\pi_0 = \pi_2, \pi_1 = 2\pi_2, \pi_3 = \pi_4 = 2p\pi_2, \pi_5 = p^2\pi_2$.

Proof. Limiting probability equations are given below:

$$\pi_0 = P_{00}\pi_0 + P_{30}\pi_3 + P_{50}\pi_5 \tag{3.41}$$

$$\pi_1 = P_{01}\pi_0 + P_{11}\pi_1 + P_{13}\pi_3 + P_{14}\pi_4 \tag{3.42}$$

$$\pi_2 = P_{02}\pi_0 + P_{12}\pi_1 + P_{22}\pi_2 \tag{3.43}$$

$$\pi_3 = P_{13}\pi_1 + P_{43}\pi_4 \tag{3.44}$$

$$\pi_4 = P_{14}\pi_1 + P_{24}\pi_2 \tag{3.45}$$

$$\pi_5 = P_{25}\pi_2 \tag{3.46}$$

Write the probabilities in place, we get the following equations:

$$p(2-p)\pi_0 = \pi_3 - p\pi_3 + p^2\pi_2 \tag{3.47}$$

$$p(2-p)\pi_1 = 2p\pi_0 - 2p^2\pi_0 + p\pi_3 + \pi_4 - p\pi_4 \tag{3.48}$$

$$p(2-p)\pi_1 = p^2\pi_0 + p\pi_1 - p^2\pi_4 \tag{3.49}$$

$$\pi_3 = p\pi_1 - p^2\pi_1 + p\pi_4 \tag{3.50}$$

$$\pi_4 = p^2\pi_1 + 2p\pi_2 - 2p^2\pi_2 \tag{3.51}$$

$$\pi_5 = p^2\pi_2 \tag{3.52}$$

In order to solve the equations, write equation 3.51 in equation 3.50 in place:

$$\pi_3 = p\pi_1 - p^2\pi_1 + p^3\pi_1 + 2p^2\pi_2 - 2p^3\pi_2 \tag{3.53}$$

Write the equation above in equation 3.47 in place:

$$p(2-p)\pi_0 = p\pi_1 - p^2\pi_1 + p^3\pi_1 + 2p^2\pi_2 - 2p^3\pi_2 + p^3\pi_1 - p^4\pi_1 - 2p^3\pi_2 + 2p^4\pi_2 + p^2\pi_2 \tag{3.54}$$

$$\pi_0 = \frac{(1-2p+2p^2-p^3)\pi_1 + (3p-4p^2+2p^3)\pi_2}{2-p} \tag{3.55}$$

Write π_0 above in equation 3.49 in place:

$$p(2-p)\pi_2 = p^2\left[\frac{(1-2p+2p^2-p^3)\pi_1 + (3p-4p^2+2p^3)\pi_2}{2-p}\right] + p(1-p)\pi_1 \quad (3.56)$$

$$(2p)^2\pi_2 = p\big((1-2p+2p^2-p^3)\pi_1 + (3p-4p^2+2p^3)\pi_2\big) + (1-p)(2-p)\pi_1 \quad (3.57)$$

$$(4-4p+p^2-3p^2+4p^3-2p^4)\pi_2 = (2-3p+p^2+p-2p^2+2p^3-p^4)\pi_1 \quad (3.58)$$

$$\pi_1 = \frac{(4-4p-2p^2+4p^3-2p^4)\pi_2}{2-2p-p^2+2p^3-p^4} \rightarrow \pi_1 = 2\pi_2 \quad (3.59)$$

Write π_1 in equation 3.51 in place:

$$\pi_4 = 2p^2\pi_2 + 2p\pi_2 - 2p^2\pi_2 \quad (3.60)$$

$$\pi_4 = 2p\pi_2 \quad (3.61)$$

Write π_4 in equation 3.50 in place:

$$\pi_3 = 2p\pi_2 - 2p^2\pi_2 + 2p^2\pi_2 \quad (3.62)$$

$$\pi_3 = 2p\pi_2 \quad (3.63)$$

Write π_3 in equation 3.47 in place:

$$p(2-p)\pi_0 = 2p\pi_2 - 2p^2\pi_2 + p^2\pi_2 \quad (3.64)$$

$$p(2-p)\pi_0 = 2p\pi_2 - p^2\pi_2 \quad (3.65)$$

$$p(2-p)\pi_0 = p(2-p)\pi_2 \quad (3.66)$$

$$\pi_0 = \pi_2 \quad (3.67)$$

Boundary condition; $\sum_i \pi_i = 1$ implies.

$$\pi_2 + 2\pi_2 + \pi_2 + 2p\pi_2 + 2p\pi_2 + p^2\pi_2 = 1 \quad (3.68)$$

$$\pi_2(4+4p+p^2) = 1 \quad (3.69)$$

$$\pi_2 = \frac{1}{(p+2)^2} \quad (3.70)$$

$$\pi_0 = \frac{1}{(p+2)^2}, \pi_1 = \frac{2}{(p+2)^2}, \pi_3 = \frac{2p}{(p+2)^2}, \pi_4 = \frac{2p}{(p+2)^2}, \quad (3.71)$$

$$\pi_1 = \frac{2p^2}{(p+2)^2}$$

Lemma 2. The first moment $\mu_1 = \sum_i \pi_i \, E[D_i|W_i] = E[D]$ equals to $\frac{2p}{(p+2)^2}(4+7p)$.

Proof:

$$E[D \mid w_i = (2,0,0)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (2,0,0)\} \cdot k \tag{3.72}$$

$$= 1 \cdot \binom{2}{1} p'(1-p)' + 2 \cdot \binom{2}{2} p^2 = 2p^2 + 2p(1-p) = 2p(1-p+p) \tag{3.73}$$

$$= 2p$$

$$E[D \mid w_i = (1,1,0)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (1,1,0)\} \cdot k \tag{3.74}$$

$$= 1 \cdot \binom{1}{1} \cdot p \cdot \binom{1}{0}(1-p) + \binom{1}{0}(1,p) \cdot \binom{1}{1} p + 2 \cdot p^2 = 2p^2 + 2p(1-p) \tag{3.75}$$

$$= 2p$$

$$E[D \mid w_i = (0,2,0)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (0,2,0)\} \cdot k \tag{3.76}$$

$$= 1 \cdot \binom{2}{1} \cdot p(1-p) + 2 \cdot \binom{2}{2} p^2 = 2p \tag{3.77}$$

$$E[D \mid w_i = (1,0,1)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (1,0,1)\} \cdot k \tag{3.78}$$

$$= 1 \cdot (1) + 2 \cdot (1 \cdot (1) \cdot p) = 1 + \ 2\mathrm{p} \tag{3.79}$$

$$E[D \mid w_i = (0,1,1)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (0,1,1)\} \cdot k \tag{3.80}$$

$$= 1 \cdot (1) + 2 \cdot \left(1 \cdot \binom{1}{1} \cdot p\right) = 1 + 2p \tag{3.81}$$

$$E[D \mid w_i = (0,0,2)] = \sum_{k=0}^{N} \Pr\{D = k \mid w_i = (0,0,2)\} \cdot k \ = \ 2 \tag{3.82}$$

$$E[\mathrm{D}] = \sum_i \pi_i E[\mathrm{D} \mid w_i] \tag{3.83}$$

$$= \pi_0 E[D \mid w_0] + \pi_1 E[D \mid w_1] + \pi_2 E[D \mid w_2] + \pi_3 E[D \mid w_3]$$

$$+ \pi_4 E[D \mid w_4] + \pi_5 E[D \mid w_5]$$

$$= \frac{1}{(p+2)^2} \cdot 2p + \frac{2}{(p+2)^2} \cdot 2p + 2p \cdot \frac{1}{(p+2)^2} + \frac{2p}{(p+2)^2} \cdot (1+2p) \tag{3.84}$$

$$\cdot + \frac{2p}{(p+2)^2} \cdot (1+2p) + \frac{p^2}{(p+2)^2} \cdot 2$$

$$= \frac{2p}{(p+2)^2} \cdot (4+7p) = E[D] \tag{3.85}$$

■

Lemma 3. The second moment of the given Markov chain is $\mu_2 = E[D^2] - (E[D])^2$ which equals to $\frac{2p}{(p+2)^2}(6 + 14p)$.

Proof:

$$E[D_i^2 \mid w_i = (2,0,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (2,0,0)\}\, k^2 \tag{3.86}$$

$$= 1^2 \binom{2}{1} \cdot p \cdot (1-p) + 2^2 \binom{2}{2} \cdot p^2 = 2p(1-p) + 4p^2 = 2p(2p+1-p) \tag{3.87}$$

$$= 2p(1+p)$$

$$\mathrm{E}[D_i^2 | w_i = (1,1,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (1,1,0)\} k^2 \tag{3.88}$$

$$= 1^2 \left[\binom{1}{1} p \binom{1}{0} (1-p) + \binom{1}{0} (1-p) \binom{1}{1} p\right] + 2^2 \binom{1}{1} \mathrm{p} \binom{1}{1} \mathrm{p} \tag{3.89}$$

$$= 2\mathrm{p}(1-p) + 4p^2 = 2p(1+p)$$

$$E[D_i^2 | w_i = (0,2,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,2,0)\} k^2 \tag{3.90}$$

$$= 1^2 \binom{2}{1} p(1-p) + 2^2 \binom{2}{2} p^2 \tag{3.91}$$

$$E[D_i^2 | w_i = (1,0,1)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (1,0,1)\} k^2 \tag{3.92}$$

$$= 1 + 2^2(p) = 1 + 4p \tag{3.93}$$

$$E[D_i^2 | w_i = (0,1,1)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,1,1)\} k^2 \tag{3.94}$$

$$= 1 + 2^2(p) = 1 + 4p \tag{3.95}$$

$$E[D_i^2 | w_i = (0,0,2)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,0,2)\} k^2 = 2^2 . 1 \ = 4 \tag{3.96}$$

$$E[D^2] = \pi_0\big(2p(1+p)\big) + \pi_1\big(2p(1+p)\big) + \pi_2\big(2p(1+p)\big) + \pi_3(1+4p) + \pi_4(1+4p) + \pi_5 4 \tag{3.97}$$

$$= \frac{1}{(p+2)^2}(2p(1+p) + 2(2p(1+p)) + 2p(1+p) + 2p(1+4p)2p(1+4p) + 4p^2) \tag{3.98}$$

$$= \frac{2p}{(p+2)^2}(6+14p) \tag{3.99}$$

■

Lemma 4. The third moment of the given Markov chain is $\mu_3 = E[(D - E[D])^3]$ which is equal to $\frac{2p}{(p+2)^2}(6+32p)$.

Proof:

$$E[D_i^3 \mid w_i = (2,0,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (2,0,0)\} k^3 \tag{3.100}$$

$$= 1^3 \binom{2}{1} \cdot p \cdot (1-p) + 2^3 \binom{2}{2} \cdot p^2 = 2p(1-p) + 8p^2 = 2p(1+3p) \tag{3.101}$$

$$E[D_i^3 | w_i = (1,1,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (1,1,0)\} k^3 \tag{3.102}$$

$$= 1^3 \left[\binom{1}{1} p \binom{1}{0}(1-p) + \binom{1}{0}(1-p)\binom{1}{1} p\right] + 2^3 \binom{1}{1} p \binom{1}{1} p \tag{3.103}$$

$$= 2p(1-p) + 8p^2 = 2p(1+3p)$$

$$E[D_i^3 | w_i = (0,2,0)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,2,0)\} k^3 \tag{3.104}$$

$$= 1^3 \binom{2}{1} p(1-p) + 2^3 \binom{2}{2} p^2 = 2p(1+3p) \tag{3.105}$$

$$E[D_i^3 | w_i = (1,0,1)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (1,0,1)\} k^3 \tag{3.106}$$

$$= 1 + 2^3(p) = 1 + 8p \tag{3.107}$$

$$E[D_i^3 | w_i = (0,1,1)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,1,1)\} k^3 \tag{3.108}$$

$$= 1 + 2^3(p) = 1 + 8p \tag{3.109}$$

$$E[D_i^3 | w_i = (0,0,2)] = \sum_{k=1}^{N} Pr\{D_i = k \mid w_i = (0,0,2)\} k^3 = 2^3.1 = 8 \tag{3.110}$$

$$E[D^3] = \pi_0\big(2p(1+3p)\big) + \pi_1\big(2p(1+3p)\big) + \pi_2\big(2p(1+3p)\big) + \pi_3(1+8p) + \pi_4(1+8p) + \pi_5 8 \tag{3.111}$$

$$E[D^3] = \frac{1}{(p+2)^2}(2p(1+p) + 2(2p(1+p)) + 2p(1+p) + 2p(1 + 4p)2p(1+4p) + 4p^2) = \frac{2p}{(p+2)^2}(6+32p) \tag{3.112}$$

■

Lemma 1 supports Lemma 2, Lemma 3 and Lemma 4. Moments obtained from Lemma 2, Lemma 3 and Lemma 4 will be used to classify demand distribution of Markov chain given in Figure 2 using Ord (1967) and Adan et al. (1995) studies.

4. SIMULATION TESTS

In this part of the study, simulations are made using the obtained moments and parameter estimates above. First of all, the parameter space required for the simulations is created. First, Ord (1967) classification is conducted for this parameter space, and Hypergeometric and Binomial distribution are found. The elements of the parameter space giving the Binomial and Hypergeometric distribution are separated and named as Binomial parameter space and Hypergeometric parameter space in the next step of the study. The elements of these parameter spaces are also classified using the distribution family given by Adan et al. (1995). The Binomial distribution is obtained for all the elements of the parameter spaces. The distributions selected from the distribution classes by Adan et al. (1995) and Ord (1967) are then tested using the chi-square test and the likelihood-ratio test. R programming code is given in Appendix A. In addition, the C++ code that gives the numerically optimum order quantity is given in Appendix B.

4.1. Single Stage Failures with Single Quality Level

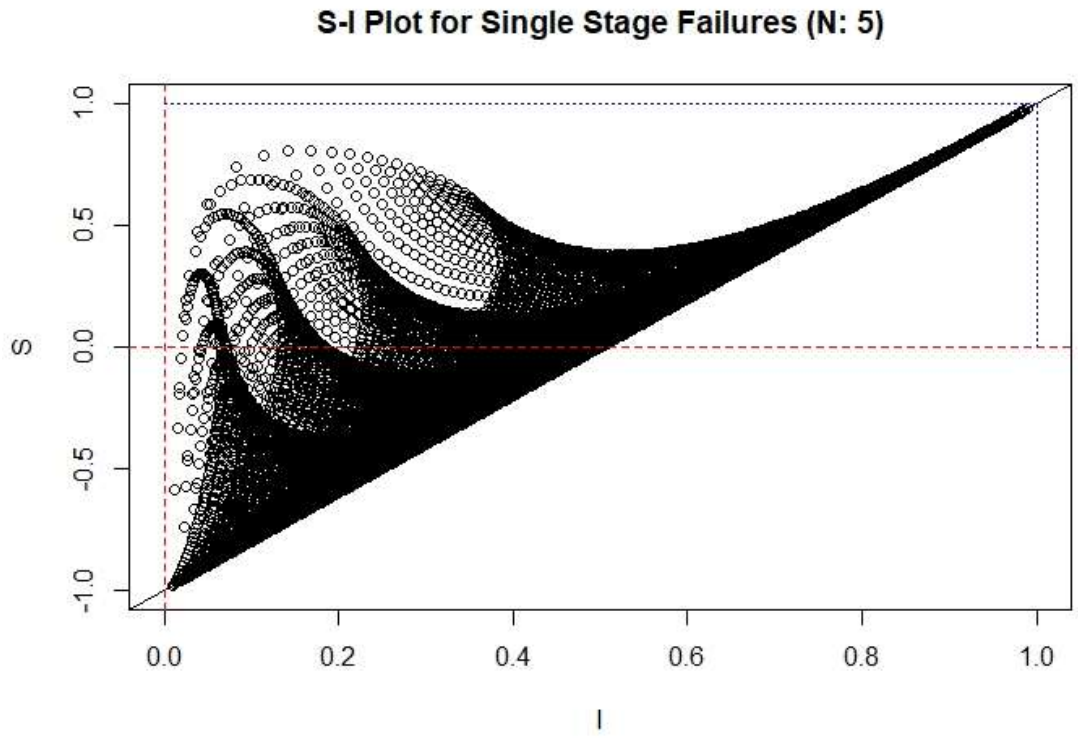
The characterization of the demand distribution is made using the discrete distribution family created by Ord (1967) for the number of original and printed parts in the system. Ord (1967) characterizes the discrete distributions using the first three central moments. In the mathematical models in Section 3, we characterize spare parts demand for a given set of system parameters using their moments. The parameter space is created for the characterization of the demand distribution. Parameter space includes the number of machines in the system (N), the number of original parts in the system (m_0), the failure probability of the original part (p) and the failure probability of printed part $\tilde{p}$. The number of machines in the system (N) is taken as 5ℓ, and the number of original parts in the system is taken as $m_0 \in \{1\ell, 2\ell, 3\ell, 4\ell\}$, $\ell \in \{1,2,10,20\}$, $N \in \{5,10,50,100\}$. Failure probabilities are determined as p for the original part and $\tilde{p}$ for the printed part, and the parameter space satisfies the assumption of $p \leq \tilde{p}$. In addition, the parameter space has been rearranged, so that number of original and printed spare parts in the system does not exceed the number of machines in the system. That is, the parameter space satisfies the $m_0 + m_1 \leq N$ condition. Ord (1967) uses the Discrete Distribution Family in Figure 1 in his study, and the GB line gives the Binomial distribution. While

performing the calculations and distribution characterization, an interval (ϵ) is used around the GB line, and the studies are repeated for different interval lengths. Table 1 shows the components of parameter space and the values taken for the parameter spaces.

Table 1. Parameter Set of Simulation Tests for Single Stage Failures

ϵ	N	m_0, m_1	$p, \tilde{p}$
0.001	5	$m_0, m_1 = (1,2,3,4)$	$p, \tilde{p} = (0.01: 0.99)$
0.01	10	$m_0, m_1 = (2,4,6,8)$	
0.1	50	$m_0, m_1 = (10,20,30,40)$	
	100	$m_0, m_1 = (20,40,60,80)$	

Figure 3. S-I Plot for Single Stage Failures



Using the moments in Section 3.1, S and I are calculated $\left(S = \frac{\mu_3}{\mu_2} and I = \frac{\mu_2}{\mu_1}\right)$ and Ord (1967) classification is conducted. Figure 3 shows the S-I graph when $N = 5$ and $\epsilon = 0.001$. According to Ord's classification, the elements of the parameter space follow the Hypergeometric distribution and the Binomial distribution. Table 2 shows the dimensions

of the parameter spaces and the dimensions of the incoming demand distributions. For all $N \in \{5,10,50,100\}$ values, the dimensions of the parameter spaces and the number of elements of the incoming distributions are the same. This situation is already expected because the parameter space is systematically divided into equal parts, and as N increases, the number of parts in the system m_0, m_1 also increases equally.

Table 2. Selected Distributions from Hypergeometric Distribution Family for Single Stage Failures

ϵ	Parameter Space	Hypergeometric Distribution	Binomial Distribution
0.001	19800	18250	1550
0.01	19800	15292	4508
0.1	19800	7893	11907

Ord's (1967) discrete distribution family gives hypergeometric and binomial distribution for the prepared parameter space. After that, parameter space is also classified according to Adan et al.'s (1995) study, and results indicate only the Binomial Distribution. The chi-square test is performed after parameter estimations are made for Hypergeometric and Binomial distributions. The Chi-square test is used to compare observed demand in the simulation with the expected demand taken from the parameter estimation. Since the studied distributions are discrete, the Kolmogorov Smirnov test is not applied in the study. In order to compare Ord (1967) and Adan et al. (1995) distribution selection models, the likelihood ratio test is applied. Likelihood ratio test statistics approximately follows a chi-square distribution. The likelihood ratio test requires both estimations under the null and alternative hypotheses, which is the main reason use of the likelihood ratio test in the study.

Table 3. Chi-Square and Likelihood Ratio Test Results for Single Stage Failures

	Chi-Square Test				Likelihood Ratio	
	Binomial		Hypergeometric			
$\epsilon = 0.001$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.69	0.87	0.27	0.53	0.95	0.41
N=10	0.40	0.63	0.18	0.31	0.94	0.45
N=50	0.04	0.09	0.01	0.02	0.96	0.51
N=100	0.02	0.04	0.003	0.003	0.98	0.56
	Binomial		Hypergeometric			
$\epsilon = 0.01$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.48	0.87	0.19	0.48	0.72	0.33
N=10	0.25	0.60	0.13	0.26	0.70	0.38
N=50	0.02	0.06	0.01	0.01	0.73	0.44
N=100	0.004	0.009	0.01	0.002	0.58	0.49
	Binomial		Hypergeometric			
$\epsilon = 0.1$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.22	0.79	0.04	0.26	0.43	0.17
N=10	0.12	0.53	0.04	0.08	0.35	0.26
N=50	0.01	0.04	0.004	0.004	0.48	0.32
N=100	0.002	0.01	0.0006	0.0001	0.58	0.35

Looking at the chi-square test results, it is observed that the demand model given by Adan et al. (1995) gives better results than the demand distribution given by Ord (1967). While $\epsilon = 0.001$, for both Binomial distribution space and Hypergeometric distribution space, the acceptance rates by Ord (1967) and Adan et al. (1995) decrease as N increases. As interval length (ϵ) increases, the relevant elements of the parameter space move from the hypergeometric distribution to the binomial distribution. The area between the GP line in Figure 1 increases as interval length increases. To explain implicitly, when $\epsilon = 0.001$, an element of parameter space might have Hypergeometric distribution. When we increase the interval $\epsilon = 0.1$, the same element might be included in the Binomial distribution. This shift between assigned distribution affects the test statistics. In Figure 6, it is possible to see the change of binomial and hypergeometric distributions when $\epsilon = 0.1$. Similar results come for $N = 10, N = 50, N = 100$, so the graphs are not included in the study.

We observe that the data in the Hypergeometric distribution has a high acceptance rate by Ord (1967) at points close to the Binomial line (GP in Figure 1) and the chi-square statistic at intermediate points, which is away from the GP line rejects the Ord classification. Figure 4 shows Ord's chi-square test results for $N = 5, (m_0, m_1) = (1,4), \ \epsilon = 0.001$. The red dots represent the parameters that the chi-square test statistic rejects, and the blue dots stand for the parameters that the chi-square statistic cannot reject the null hypothesis.

Figure 4. Ord Chi-Square Results for $\epsilon = 0.001$ and Single Stage Failures

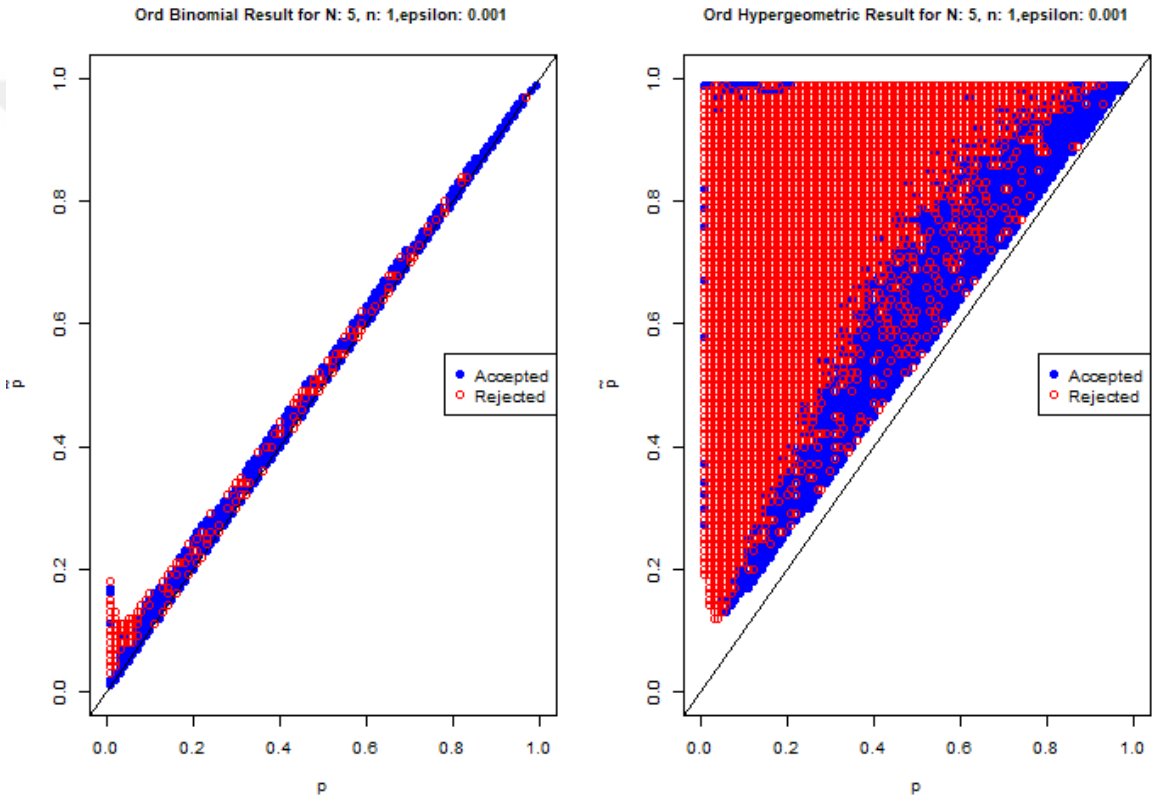


Figure 5. Adan Chi-Square Results for $\epsilon = 0.001$ and Single Stage Failures

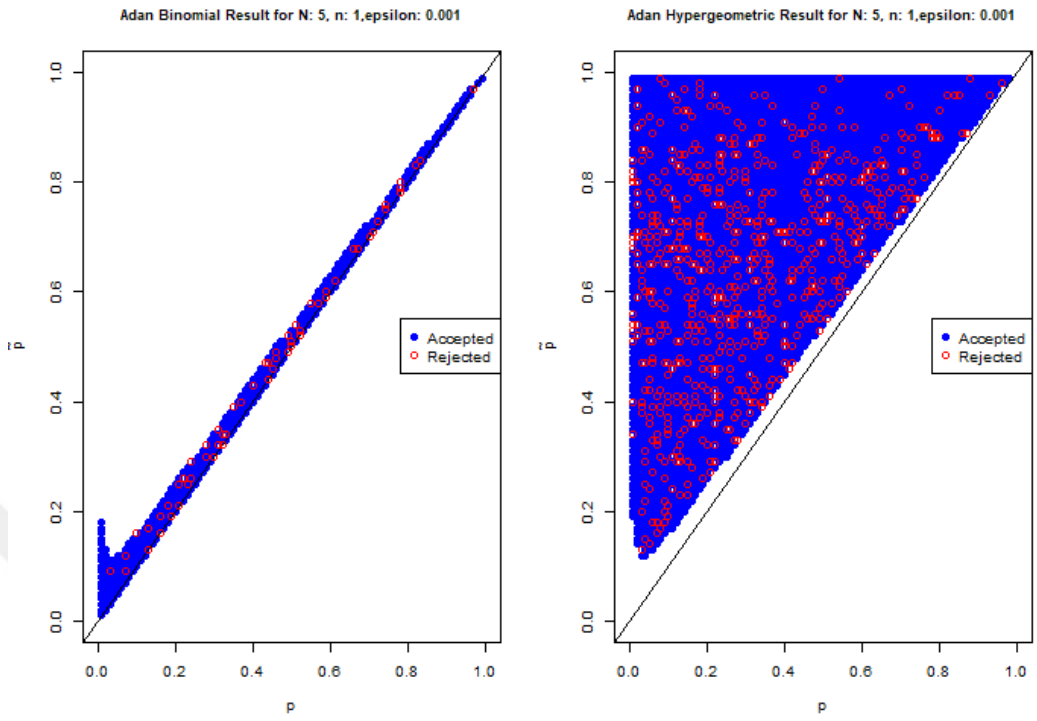
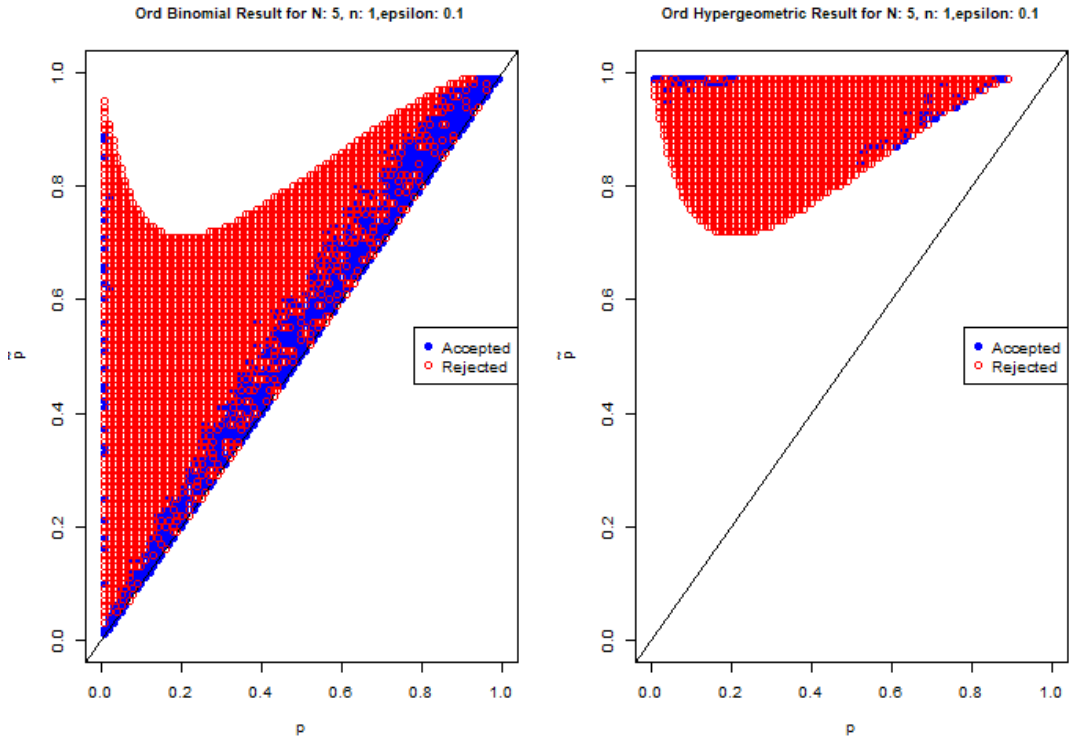
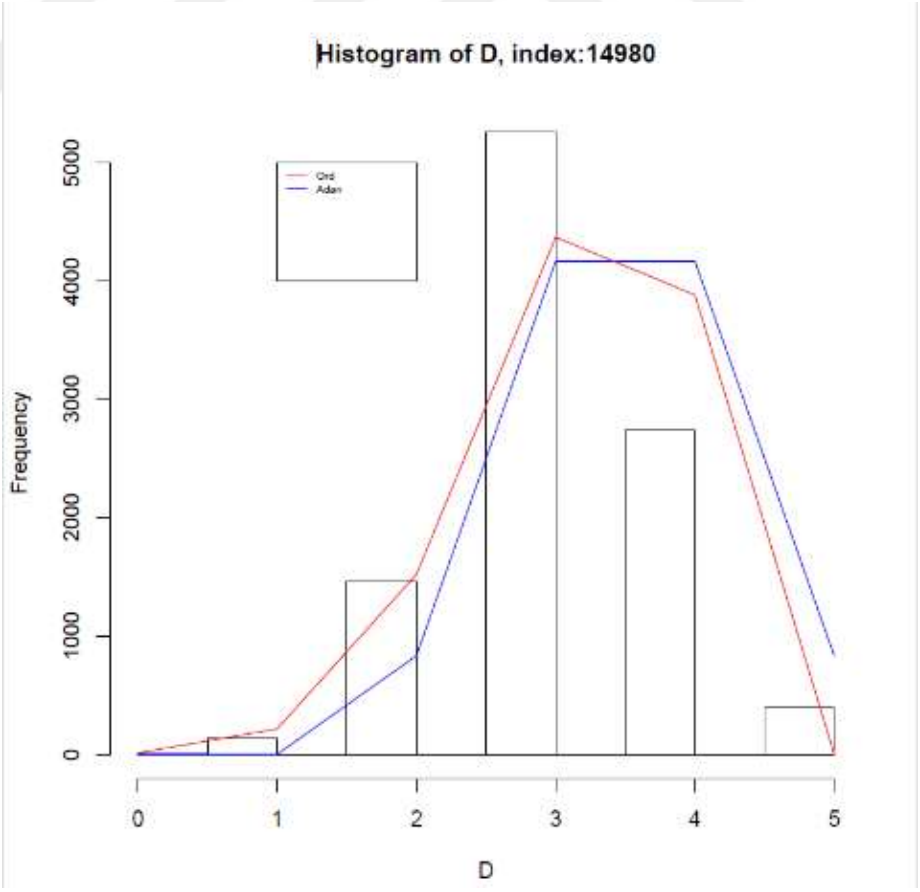


Figure 6. Ord Chi-Square Results for $\epsilon = 0.1$ and Single Stage Failures



While the chi-square test statistics obtained for Ord (1967) distribution characterization are lower than the chi-square test statistics obtained for Adan et al. (1995) classification, the likelihood ratio test results show that the characterization of hypergeometric distribution family given by Ord gives better results than Adan et al. (1995). This is due to the fact that the distribution characterization is given by Ord (1967) rejects the chi-square test at some points. Still, the likelihood ratio test gives better results than the classification given by Adan et al. (1995). An example of this situation is given in Figure 7. It shows the difference between the actual demand distribution and the convergence of the demand distribution given by the Ord (1967) family of discrete distributions and the convergence of the demand distribution given by Adan et al. (1995) classification. The red line shows the Ord distribution, while the blue line shows the Adan distribution. When we plot the histogram of actual demand and add Ord (1967) and Adan et al. (1995) forecast demand to Figure 7, Ord's forecast better reflects the change in demand than Adan's demand forecast.

Figure 7. Histogram of Demand for Single Stage Failures



4.2. Multiple Quality Levels of Printed Parts

Advances in additive manufacturing technologies allow printing parts at different quality levels depending on the needs. In this case, while the total number of parts working in the system is N, under the assumption that there are parts printed at v different levels in the system, the total demand for spare parts will consist of $v+1$ units of Binomial convolutions. So, a new parameter space is created for $v=2$, and analyzes are made for this parameter space. Table 4 shows the parameter space for multiple quality levels. m_0 is the original part. $\overline{m_1}$ and $\overline{m_2}$ are printed parts with quality level 1 and 2, respectively. p is the failure probability of the original part, $\widetilde{p_1}$ is the failure probability of the printed part at quality level 1, $\widetilde{p_2}$ is the failure probability of the printed part at quality level 2, and $p \leq \widetilde{p_1} \leq \widetilde{p_2}$. $m_0 + \overline{m_1} + \overline{m_2} = N$.

Table 4. Parameter Space for Multiple Quality Levels

ϵ	N	$m_0, \overline{m_1}, \overline{m_2}$	$p, \widetilde{p_1}, \widetilde{p_2}$
0.001	5	$m_0, \overline{m_1}, \overline{m_2} = (1,2,3,4)$	$p, \widetilde{p_1}, \widetilde{p_2} = (0.01:0.99)$
0.01	10	$m_0, \overline{m_1}, \overline{m_2} = (2,4,6,8)$	
0.1	50	$m_0, \overline{m_1}, \overline{m_2} = (10,20,30,40)$	
	100	$m_0, \overline{m_1}, \overline{m_2} = (20,40,60,80)$	

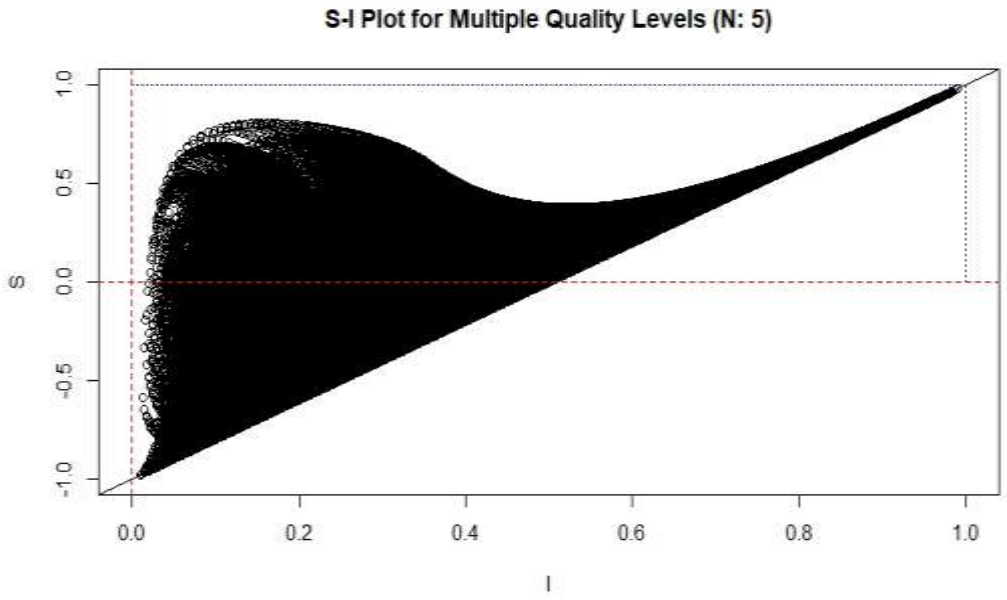
As handled in Section 4.1, Ord (1967) and Adan et al. (1995) studies are made for parameter space in this part of the study. Then, the chi-square test and likelihood ratio test are applied. Table 5 shows the size of the parameter space and the dimensions of the demand distribution space. The dimensions of the parameter space and distribution spaces are the same for all $N \in \{5,10,50,100\}$. For the parameter space, Ord (1967) characterization gives Hypergeometric and Binomial distributions. Later, Adan et al. (1995) classification is applied, and Binomial distribution is obtained for the entire parameter space.

Table 5. Selected Distributions from Hypergeometric Distribution Family for Multiple Quality Levels

ϵ	Parameter Space	Hypergeometric Distribution	Binomial Distribution
0.001	1666500	1569928	96572
0.01	1666500	1361204	305296
0.1	1666500	748641	917859

Figure 8 shows the S-I graph obtained while performing the Ord (1967) classification for multiple quality levels, when $N = 5, \epsilon = 0.001$. The structure of Figure 8 is similar to the S-I graph in Section 4.1.

Figure 8. S-I Plot for Multiple Quality Levels

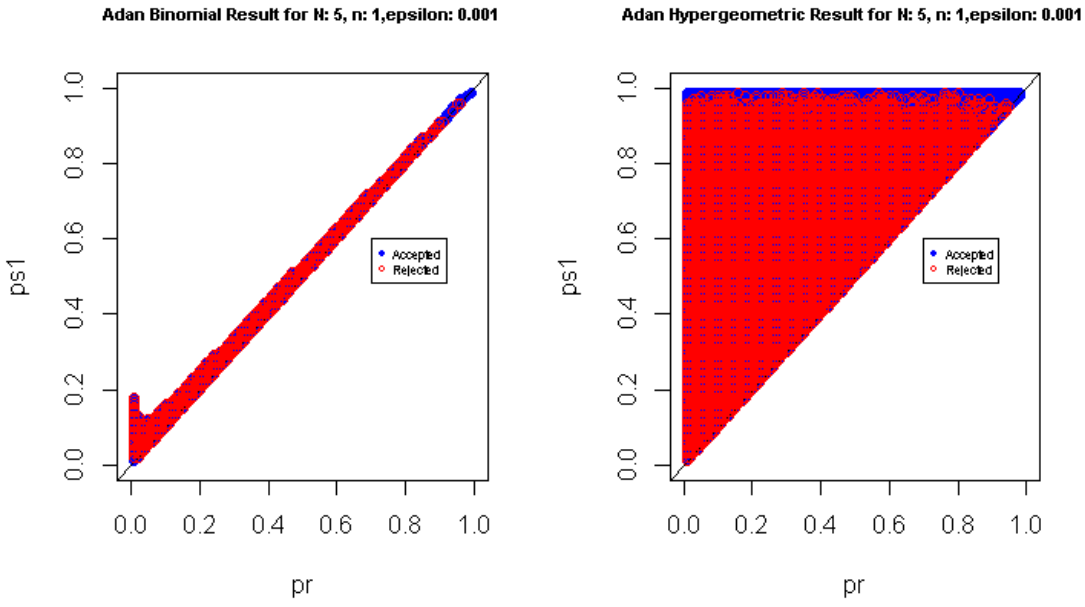


Similar to the results in Section 4.1, the chi-square test statistics for the classification by Adan et al. (1995) is better than the chi-square tests obtained from Ord (1967) characterization. Table 6 shows the results of the chi-square and likelihood-ratio tests for multiple quality levels of printed parts. As the number of machines (*N*) in the system increases, the chi-square test statistics for both Ord (1967) and Adan et al. (1995) classification decrease. However, the likelihood-ratio test results increase. Figure 9 shows the chi-square test results for Adan et al. (1995) classification.

Table 6. Chi-Square and Likelihood Ratio Test Results for Multiple Quality Levels

	Chi-Square Test				Likelihood Ratio	
	Binomial		Hypergeometric			
$\epsilon = 0.001$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.67	0.88	0.21	0.44	0.91	0.35
N=10	0.39	0.65	0.13	0.26	0.91	0.40
N=50	0.05	0.10	0.01	0.01	0.94	0.46
N=100	0.02	0.03	0.0009	0.0012	0.97	0.53
	Binomial		Hypergeometric			
$\epsilon = 0.01$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.40	0.87	0.14	0.37	0.61	0.29
N=10	0.21	0.62	0.09	0.20	0.60	0.35
N=50	0.02	0.07	0.004	0.01	0.70	0.40
N=100	0.01	0.02	0.0005	0.0006	0.79	0.47
	Binomial		Hypergeometric			
$\epsilon = 0.1$	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.16	0.74	0.03	0.15	0.26	0.19
N=10	0.09	0.49	0.03	0.05	0.29	0.28
N=50	0.01	0.03	0.0025	0.0018	0.47	0.31
N=100	0.003	0.01	0.0002	0.0002	0.60	0.35

Figure 9. Adan Chi-Square Results for $\epsilon = 0.001$ and Multiple Quality Levels



4.3. Multistage Failures and Multiple Quality Levels of Printed Parts

The aging conditions of the parts working in the system are considered, and it is assumed that the failures occur gradually. In this part of the study, both the original and printed parts have the ageing level(s). For original and printed parts, aging will take place in 3 levels, $(s = 3)$. The first level is the stage that the part is currently replaced. The second level is the intermediate level. The third level is the last stage, where the spare part fails. The number of original parts is m_0, m_1, m_2 and the number of printed parts is given as $\overline{m_1}, \overline{m_2}, \overline{m_3}$. Since failures follow the Binomial distribution, the total demand for spare parts will come from 2s Binomial convolutions. Table 7 shows parameter sets, and Table 8 shows the size of the parameter space. Parameter space constitutes of original parts $(\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3)$ and printed parts $(\overline{\mathrm{m}_1}, \overline{\mathrm{m}_2}, \overline{\mathrm{m}_3})$. Failure probability of original part at level 1 which is the new condition of the part is $\mathrm{p}_1 = \mathrm{p}$. Failure probability of original part at level 2 is $\mathrm{p}_2 = \mathrm{p}/2$. Failure probability of original part at level 3 is $\mathrm{p}_3 = \mathrm{p}/3$. Also, failure probability of printed part at level 1 which is the new condition of the part is $\widetilde{\mathrm{p}_1} = \tilde{\mathrm{p}}$. Failure probability of printed part at level 2 which is the new condition of the part is $\widetilde{\mathrm{p}_2} = \tilde{\mathrm{p}}/2$. Failure probability of printed part at level 3 which is the new condition of the part is $\widetilde{\mathrm{p}_3} = \tilde{\mathrm{p}}/3$. In this study, the first three moments are calculated according to Proposition 4 under the assumption of the conditional distribution.

Table 7. Parameter Space for Multistage Failures

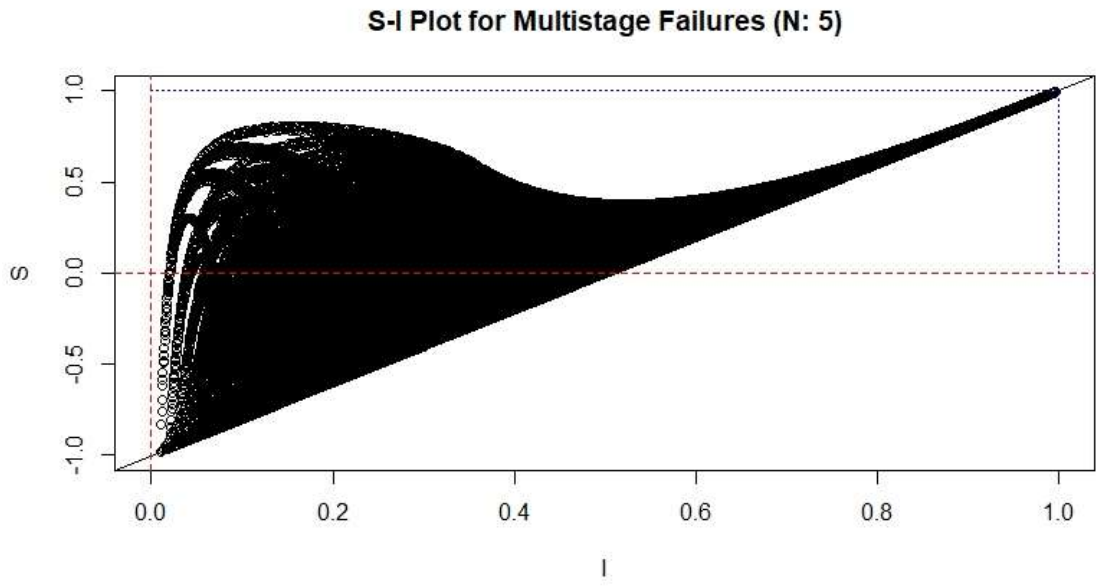
$\boldsymbol{\epsilon}$	**N**	$\boldsymbol{m_0, m_1, m_2, \overline{m_1}, \overline{m_2}, \overline{m_3}}$	$\boldsymbol{p, \tilde{p}}$
0.001	5	$m_{0,1,2}, \overline{m_{1,2,3}} = (1,2,3,4)$	$p, \tilde{p} = (0.01: 0.99)$
0.01	10	$m_{0,1,2}, \overline{m_{1,2,3}} = (2,4,6,8)$	
0.1	50	$m_{0,1,2}, \overline{m_{1,2,3}} = (10,20,30,40)$	
	100	$m_{0,1,2}, \overline{m_{1,2,3}} = (20,40,60,80)$	

Table 8. Magnitudes of Parameter and Distribution Spaces for Multistage Failures

ϵ	Parameter Space	Hypergeometric Distribution	Binomial Distribution
0.001	1217700	1149627	68073
0.01	1217700	921409	296291
0.1	1217700	435164	782536

Figure 10 shows the S-I graph obtained while performing the Ord (1967) distribution characterization, when $N = 5, \epsilon = 0.001$. The structure of Figure 9 is similar to the S-I graph that emerged in Section 4.1.

Figure 10. S-I Plot for Multistage Failures



In this part of the study, the chi-square test results are close to Ord (1967) and Adan et al. (1995). Besides, the acceptance rates are low compared to Section 4.1 and Section 4.2. However, the likelihood-ratio test results show that the Ord (1967) discrete distribution family performs better. Table 9 shows the test statistics for multistage failures and multiple printed parts. For all the number of parts in the system $N \in \{5,10,50,100\}$, the chi-square test acceptance rates and likelihood-ratio test results are close to each other. On the other hand, while the chi-square test acceptance rate for $N = 100$ and $\epsilon =$

0.001 decreases, the likelihood-ratio test results show a significant increase for the Ord discrete distribution family.

Table 9. Chi-Square Test and Likelihood Ratio Test Results for Multistage Failures

	Chi-Square Test				Likelihood Ratio	
	Binomial		Hypergeometric			
ϵ=0.001	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.05	0.08	0.18	0.40	0.60	0.30
N=10	0.05	0.07	0.13	0.30	0.63	0.33
N=50	0.04	0.05	0.01	0.03	0.62	0.46
N=100	0.11	0.19	0.00	0.00	0.97	0.54
	Binomial		Hypergeometric			
ϵ =0.01	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.05	0.12	0.10	0.29	0.54	0.23
N=10	0.05	0.11	0.07	0.21	0.58	0.25
N=50	0.03	0.06	0.00	0.01	0.59	0.38
N=100	0.01	0.03	0.00	0.00	0.60	0.46
	Binomial		Hypergeometric			
ϵ =0.1	Ord	Adan	Ord	Adan	Binomial	Hypergeometric
N=5	0.06	0.27	0.02	0.06	0.43	0.15
N=10	0.06	0.24	0.01	0.01	0.48	0.18
N=50	0.02	0.06	0.00	0.00	0.58	0.22
N=100	0.01	0.02	0.00	0.00	0.64	0.26

5. CONCLUSION

In the view of an inventory planner, the failure of a machine creates a demand for a spare part. Replacement of spare parts is made with the inventory supplied from OEM. 3DP becomes a supporter of inventory control management if OEM does not deliver spare parts immediately. 3DP enables companies to fulfill their needs by on-demand production. Knowing demand distribution is vital for demand forecasts and inventory planning, and joint use of original and printed parts in a system changes the demand distribution. For this reason, this study aims to figure out the demand distribution in the existence of three-dimensional printers and establishes parameter estimation using the moment approach. In order to find the demand distribution, the hypergeometric distribution family characterized by Ord (1967) and the discrete distribution fitting method of Adan et al. (1995) are used. Both studies give the classification based on the central moments.

Under the three scenarios considered during the study, the demand distributions obtained are Hypergeometric and Binomial distributions. The first of these scenarios is a system with original and written parts operating at a single quality level. The created parameter space was run for different interval lengths and the percentages of hypergeometric distribution for epsilon=0.001, epsilon=0.01 and epsilon=0.1 are 92.2%, 77.2% and 40.6%, respectively. Laser polishing technology increases the durability of a product, and in the second case this technology is considered. There is only one quality level for the original part. There are two different quality levels for the printed part. In the last scenario, the aging of the original and written parts over time is considered. The chi-square test and likelihood ratio tests show that the demand classification is given by Ord (1967) and Adan et al. (1995) becomes insufficient as the system complexity increases.

In the multistage failures part of this study, simulations are made using the moments of the conditional distribution. For future research, simulations of multistage failures can be conducted using the moments of the Markov chain. Also, the Markovian diagram in this study assumes that failures of the original part occur at two-level and replacement of the original part is done with the printed part. Replacement of printed part is made with the original part. However, the Markovian diagram can be revised with the change of

replacement policy. In a real-life scenario, the replacements of the original and printed part can be done with both the original and the printed parts.

BIBLIOGRAPHY

Adan, I., van Eenige, M., & Resing, J. (1995). Fitting discrete distributions on the first two moments. *Probability in the engineering and informational sciences*, 9(4), 623-632.

Axsäter, S. (2006). *Inventory control*. Springer.

Boxiang Liu, and Thomas Quertermous. Approximating the Sum of Independent Non-Identical Binomial Random Variables, *Arxiv*:1712.01410v1, 2017.

Chan, H. K., Griffin, J., Lim, J. J., Zeng, F., & Chiu, A. S. (2018). The impact of 3D Printing Technology on the supply chain: Manufacturing and legal perspectives. *International Journal of Production Economics*, 205, 156-162.

Cohen, A. C. (1951). Estimation of parameters in truncated Pearson frequency distributions. *The Annals of Mathematical Statistics*. 22, 256-65.

de Smidt-Destombes, K. S., van der Heijden, M. C., & Van Harten, A. (2006). On the interaction between maintenance, spare part inventories and repair capacity for a k-out-of-N system with wear-out. *European journal of operational research*, 174(1), 182-200.

Deshpande, V., Iyer, A. V., & Cho, R. (2006). Efficient supply chain management at the US Coast Guard using part-age dependent supply replenishment policies. *Operations Research*, 54(6), 1028-1040.

Guerrero-Salazar, P. L. A., & Yevjevich, V. M. (1975). Analysis of drought characteristics by the theory of runs (Doctoral dissertation, Colorado State University. Libraries).

Gupta, P. L., & Singh, J. (1981). On the moments and factorial moments of a MPSD. *In Statistical Distributions in Scientific Work* (pp. 189-195). Springer, Dordrecht.

Hekimoğlu, M. (2015). *Spare parts management of aging capital products* (No. EPS-2015-368-LIS).

Hekimoğlu, M., & Ulutan, D. (2020). Optimum Laser Polishing Decision-Making for On-Demand Additive Manufacturing of Spare Parts: An Exploratory Study. *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 24(2), 516-525.

Holweg, H., 2015. The limits of 3D printing. Harv. Bus. Rev Available at: https://hbr.org/ 2015/06/the-limits-of-3d-printing, Accessed date: 02 September 2021.

J. K. Ord, On a System of Discrete Distributions. *Biometrika*, Vol. 54, No. 3/4 (Dec., 1967),pp. 649-656.

Janssen, F., Heuts, R., & de Kok, T. (1998). On the (R, s, Q) inventory model when demand is modelled as a compound Bernoulli process. *European journal of operational research*, 104(3), 423-436.

Katz, L. (1948). Frequency functions defined by the Pearson difference equation. *In Annals of Mathematical Statistics* (Vol. 19, No. 1, pp. 120-120).

Jolayemi, K. A unified approximation scheme for the convolution of independent binomial variables. *Applied Mathematics and Computation*, 49(2-3):269–297, June 1992.

Khajavi, S. H., Partanen, J., & Holmström, J. (2014). Additive manufacturing in the spare parts supply chain. *Computers in industry*, 65(1), 50-63.

Knofius, N., van der Heijden, M. C., & Zijm, W. H. (2019). Consolidating spare parts for asset maintenance with additive manufacturing. *International journal of production economics*, 208, 269-280.

Korwar, R. M. (1989). On a new system of discrete distributions and characterizations of several discrete distributions by equality of distributions. *Annals of the Institute of Statistical Mathematics*, 41(2), 305-321.

Koščová, M., Mačutek, J., & Kelih, E. (2016). A data-based classification of Slavic languages: Indices of qualitative variation applied to grapheme frequencies. *Journal of Quantitative Linguistics*, 23(2), 177-190.

Kruth, J. P., Leu, M. C., & Nakagawa, T. (1998). Progress in additive manufacturing and rapid prototyping. *Cirp Annals*, 47(2), 525-540.

Liu, P., Huang, S. H., Mokasdar, A., Zhou, H., & Hou, L. (2014). The impact of additive manufacturing in the aircraft spare parts supply chain: supply chain operation reference (scor) model-based analysis. *Production Planning & Control*, 25(13-14), 1169-1181.

Ma, H., Li, H., & Han, Z. (2012, March). A framework of frequency oscillation in power grid: Epidemic propagation over social networks. In 2012 Proceedings *IEEE INFOCOM Workshops* (pp. 67-72). IEEE.

Ma, C. P., Guan, Y. C., & Zhou, W. (2017). Laser polishing of additive manufactured Ti alloys. *Optics and Lasers in Engineering*, 93, 171-177.

Martináková, Z., Mačutek, J., Popescu, I. I., & Altmann, G. (2009). Ord´ s criterion in musical texts. *Glottotheory*, 2(1), 86-98.

Norman L Johnson, Adrienne W Kemp, and Samuel Kotz. *Univariate Discrete Distributions. Johnson Univariate Discrete Distributions*. John Wiley Sons, Inc., Hoboken, NJ, USA, January 2005.

Rayna, T., Striukova, L., & Darlington, J. (2015). Co-creation and user innovation: The role of online 3D printing platforms. *Journal of Engineering and Technology Management*, 37, 90-102.

Rayner, J. C., Thas, O., & Best, D. J. (2009). *Smooth tests of goodness of fit: using R*. John Wiley & Sons.

Robertson, B., Fung, T., & Weber, N. (2013). An Alternative Estimator for The Shape Parameter In The Negative Binomial Distribution. *Mathematical Scientist*, 38(1).

Schneider, D. C., & Duffy, D. C. (1985). Scale-dependent variability in seabird abundance. *Marine ecology progress series. Oldendorf*, 25(3), 211-218.

Sleptchenko, A., van der Heijden, M. C., & van Harten, A. (2002). Effects of finite repair capacity in multi-echelon, multi-indenture service part supply systems. *International Journal of Production Economics*, 79(3), 209-230.

Song, J. S., & Zhang, Y. (2020). Stock or print? Impact of 3-D printing on spare parts logistics. *Management Science*, 66(9), 3860-3878.

Strong, D., Kay, M., Conner, B., Wakefield, T., & Manogharan, G. (2018). Hybrid manufacturing–integrating traditional manufacturers with additive manufacturing (AM) supply chain. *Additive Manufacturing*, 21, 159-173.

Syntetos, A. A., Babai, M. Z., Lengu, D., & Altay, N. (2011). Distributional assumptions for parametric forecasting of intermittent demand. In *Service Parts Management* (pp. 31-52). Springer, London.

Westerweel, B., Basten, R., den Boer, J., & van Houtum, G. J. (2021). Printing spare parts at remote locations: Fulfilling the promise of additive manufacturing. *Production and Operations Management*, 30(6), 1615-1632.

APPENDIX A

Appendix A.1. R Code for Distribution Selection

```
source("D:/MH/1001 Project/AllFunctions.R")  ##### CREATE VALUE SPACE #####
N=5       ## Number of capital product in the system
n=c(1,2,3,4) ## number of original product in the system
pr=(1:99)/100  ##failure probability of original part
ps=(1:99)/100  ##failure probability of printed part
tab = expand.grid(N,n,pr,ps) ##build value space
cond=((tab$Var1<tab$Var2)+(tab$Var3>tab$Var4)) ##eliminate undesired situations.
value.space=tab[cond==0,] ; names(value.space)=c("N","n","pr","ps")
value.space=cbind(value.space,0,0,0)
names(value.space)[5:7]=c("binom.mu1","binom.mu2","binom.mu3")
S=NULL ; I=NULL
for (i in 1:dim(value.space)[1])
{ N=value.space[i,1] ;   n=value.space[i,2] ; pr=value.space[i,3] ; ps=value.space[i,4]
  theo.first.center.moment=n*pr+(N-n)*ps ; value.space[i,5]=theo.first.center.moment
  theo.second.center.moment=n*pr*(1-pr) + (N-n)*ps*(1-ps)
  value.space[i,6]=theo.second.center.moment
  theo.third.center.moment=n*pr*(1-pr)*(1-2*pr) + (N-n)*ps*(1-ps)*(1-2*ps)
  value.space[i,7]=theo.third.center.moment
  S[i]=theo.third.center.moment/theo.second.center.moment
  I[i]=theo.second.center.moment/theo.first.center.moment }
plot(I,S,ylim=c(-1,1),xlim=c(-0,1),xlab="I",ylab="S",main=sprintf("S-I  Plot  for  N: %s",N))
abline(a=-1,b=2) ; abline(v=0,lty=2,col="red") ;abline(h=0,lty=2,col="red")
segments(0,1,1,1,col="blue",lty=3) ; segments(1,0,1,1,col="blue",lty=3)
value.space=cbind(value.space,S,I) ; epsilon=0.001; dist.vect=0
for(i in 1:dim(value.space)[1])
{  S=value.space[i,8] ;  I=value.space[i,9] ;cond1=((abs(S-1) + abs(I-1))<epsilon)
  cond2 = (abs(S-2*I+1)<epsilon) & (S>1) & (I>1)
```

```
 cond3=(-1<S & S<1) & ((S-2*I+1)>epsilon) & (I<1)
 cond4=(abs(S-2*I+1)<epsilon) & (S<1) & (I<1)
 if(cond1){dist.vect[i]="Pois"} ; if(!cond1 & cond2){dist.vect[i]="NBinom"}
 if(cond3){dist.vect[i]="Hypergeo"} ;  if(cond4){dist.vect[i]="Binom"}
 if((cond1+cond2+cond3+cond4)==0)   { print("Error")     break}}
value.space=cbind(value.space,dist.vect)
hyper.space=value.space[(value.space$dist.vect=="Hypergeo"),]
hyper.space=cbind(hyper.space,0,0,0,0,0)
names(hyper.space)[c(11:15)]=c("N.hyper","p1","n.root1","p2","n.root2")
for(i in 1:dim(hyper.space)[1])
{  N=hyper.space[i,1] ; n=hyper.space[i,2];pr=hyper.space[i,3];ps=hyper.space[i,4]
 binom.mu1=hyper.space[i,5]; binom.mu2=hyper.space[i,6]
 binom.mu3=hyper.space[i,7] ; rho1=binom.mu2/binom.mu1
 rho2=binom.mu3/binom.mu2
 NN=(2*rho2-2*rho1+2*binom.mu1)/(rho2-2*rho1+1); hyper.space[i,11]=NN
 alpha=1; beta=rho1*(NN-1)-NN-binom.mu1 ; gamma=NN*binom.mu1
 delta=beta*beta-4*alpha*gamma
 if(delta>=0)
 {   m.root1=(-beta-sqrt(delta))/(2*alpha) ;m.root2=(-beta+sqrt(delta))/(2*alpha)
   p1=binom.mu1/m.root1 ; p2=binom.mu1/m.root2 ;hyper.space[i,12]=p1
   hyper.space[i,13]=m.root1;     hyper.space[i,14]=p2;    hyper.space[i,15]=m.root2
 }}
chivect.hyper.ord=0; chivect.hyper.adan=0; pval.chi.hyper.ord=0; pval.chi.hyper.adan=0
LR.val.hyper=0; pval.LR.hyper=0 ; Adan.class.hyper=0; test.stat=0;
size=10000; replicsize=100
for (i in 1:dim(hyper.space)[1])
{  if(hyper.space$n.root1[i]==0)
 {    chivect.hyper.ord[i]=0;  chivect.hyper.adan[i]=0;   pval.chi.hyper.ord[i]=0
   pval.chi.hyper.adan[i]=0;  LR.val.hyper[i]=0;   pval.LR.hyper[i]=0
   Adan.class.hyper[i]=0 }
 if(hyper.space$n.root1[i]>0)
```

```
{ N=hyper.space[i,1];  n=hyper.space[i,2]; pr=hyper.space[i,3];    ps=hyper.space[i,4]
 N.hyper=round(hyper.space[i,11]);  n.root1=round(hyper.space[i,13])
 p1=hyper.space[i,12];   n.root2=round(hyper.space[i,15])
 p2=hyper.space[i,14];  binom.mu1=hyper.space[i,5];    binom.mu2=hyper.space[i,6]
 a.root1=round(N.hyper*p1); a.root2=round(N.hyper*p2);  avect=c(a.root1,a.root2)
 pvect=c(p1,p2);  index=which(avect==max(avect))
 if(a.root1==a.root2)    {      index=1    }
 a=avect[index];  b=round(N.hyper-a); n.hyper=round(binom.mu1/pvect[index])
 if(n.hyper>b)
 {      if(index==1)
  {       index2=index+1;  a=avect[index2];   b=round(N.hyper-a)
    n.hyper=round(binom.mu1/pvect[index2]) }
  if(index==2) {   index2=index-1;   a=avect[index2];   b=round(N.hyper-a)
    n.hyper=round(binom.mu1/pvect[index2])} }
 Dr=rbinom(size,n,pr) #original part demand
 Dp=rbinom(size,N-n,ps) #printed part demand;    D=Dr+Dp #total demand
 freqvect=as.numeric(table(D)/size);   freqvect.head=as.numeric(row.names(table(D)))
 if(length(freqvect)<=N)
 {      freqvect.head=c(freqvect.head,rep(0,N+1-length(freqvect.head)))
  for(k in 0:N)
  {       if(freqvect.head[k+1]!=k)
    {    freqvect=append(freqvect,0,after=k)
      freqvect.head=append(freqvect.head,k,after=k)       }
  }}
 xval = c(0:N);    probvect=dhyper(c(0:N),a,b,n.hyper);    expected= size*probvect
 res.HYP=consolidate.chisq.v2(expected, freqvect,xval,FALSE)
 expected.ord=res.HYP$expected.consol
 freqvect.ord=res.HYP$freq.consol*size
 chi2.hyper.ord=sum((freqvect.ord-expected.ord)^2/(expected.ord))
 chivect.hyper.ord[i]=chi2.hyper.ord
 df.hyper=3;  pval.hyper.ord=1-pchisq(chi2.hyper.ord,df.hyper)
```

```
  pval.chi.hyper.ord[i]=pval.hyper.ord
  adanvect=adan.dist.v2(c(0:N),binom.mu1,binom.mu2)
  expected.adan=adanvect$prob*size
  res.adan=consolidate.chisq.v2(expected.adan,freqvect,xval,FALSE)
  expected.adan=res.adan$expected.consol;   freqvect.adan=res.adan$freq.consol*size
  chi2.hyper.adan=sum((freqvect.adan-expected.adan)^2/(expected.adan))
  chivect.hyper.adan[i]=chi2.hyper.adan;   df.binommix=3
  pval.hyper.adan=1-pchisq(chi2.hyper.adan,df.binommix)
  pval.chi.hyper.adan[i]=pval.hyper.adan;   probvect.Ord=dhyper(D,a,b,n.hyper)
  Adan.vect=adan.dist.v2(D,binom.mu1,binom.mu2)
  Adan.class.hyper[i]=Adan.vect$dist;    probvect.Adan=Adan.vect$prob
  k.Adan=Adan.vect$k;   probvect.Ord[probvect.Ord==0]=0.0001
  ##probvect.Ord[D>n.hyper]=0.0001;   probvect.Adan[D>(k.Adan+1)]=0.0001
  h0=sum(log(probvect.Ord));   h1=sum(log(probvect.Adan))
  LR.val.hyper[i]=-2*(h0-h1);   pval.LR.hyper[i]=1-pchisq(LR.val.hyper[i],df.hyper)
 }}
hyper.space=cbind(hyper.space,pval.chi.hyper.ord,chivect.hyper.ord,Adan.class.hyper,p
val.chi.hyper.adan,chivect.hyper.adan,LR.val.hyper,pval.LR.hyper)
hyper.accept.ord=hyper.space[hyper.space$pval.chi.hyper.ord>=0.05,]
hyper.accept.adan=hyper.space[hyper.space$pval.chi.hyper.adan>=0.05,]
hyper.mu1.root1=NULL; hyper.mu2.root1=NULL; hyper.mu3.root1=NULL
hyper.mu1.root2=NULL; hyper.mu2.root2=NULL;hyper.mu3.root2=NULL
for(i in 1:dim(hyper.space)[1])
{  NN=hyper.space[i,11];   m.root1=hyper.space[i,13];  p1=hyper.space[i,12]
 m.root2=hyper.space[i,15];  p2=hyper.space[i,14]
 hyper.mu1.root1[i]=m.root1*p1
 hyper.mu2.root1[i]=hyper.mu1.root1[i]*(((NN-m.root1)*(1-p1))/(NN-1))
 hyper.mu3.root1[i]=hyper.mu2.root1[i]*(((NN-2*m.root1)*(1-2*p1))/(NN-2))
 hyper.mu1.root2[i]=m.root2*p2
 hyper.mu2.root2[i]=hyper.mu1.root2[i]*(((NN-m.root2)*(1-p2))/(NN-1))
 hyper.mu3.root2[i]=hyper.mu2.root2[i]*(((NN-2*m.root2)*(1-2*p2))/(NN-2)) }
```

```
hyper.space=cbind(hyper.space,hyper.mu1.root1,hyper.mu2.root1,hyper.mu3.root1,hyp
er.mu1.root2,hyper.mu2.root2,hyper.mu3.root2)
binom.space=value.space[(value.space$dist.vect=="Binom"),]
binom.space=cbind(binom.space,0,0)
names(binom.space)[c(11:12)]=c("N.binom","p.binom")
for(i in 1:dim(binom.space)[1])
{ N=binom.space[i,1];  n=binom.space[i,2]  pr=binom.space[i,3];  ps=binom.space[i,4]
  binom.mu1=binom.space[i,5] ;  binom.mu2=binom.space[i,6]
  binom.mu3=binom.space[i,7]
  p.binom=1-(binom.mu2/binom.mu1); binom.space[i,12]=p.binom
  N.binom=binom.mu1/p.binom;  binom.space[i,11]=N.binom }
chivect.binom.ord=0; chivect.binom.adan=0; pval.chi.binom.ord=0
pval.chi.binom.adan=0; pval.LR.binom=0; LR.val.binom=0
Adan.class.binom=0; test.stat=0;size=10000; replicsize=100
for (i in 1:dim(binom.space)[1])
{   N=binom.space[i,1];  n=binom.space[i,2];   pr=binom.space[i,3]  ps=binom.space[i,4]
  binom.mu1=binom.space[i,5];  binom.mu2=binom.space[i,6]
  N.binom=round(binom.space[i,11]);  p.binom=binom.space[i,12]
  Dr=rbinom(size,n,pr) #original part demand
  Dp=rbinom(size,N-n,ps) #printed part demand;  D=Dr+Dp #total demand
  freqvect=as.numeric(table(D)/size);  freqvect.head=as.numeric(row.names(table(D)))
  if(length(freqvect)<=N)
  {     freqvect.head=c(freqvect.head,rep(0,N+1-length(freqvect.head)))
    for(k in 0:N)
    {  if(freqvect.head[k+1]!=k)
      {  freqvect=append(freqvect,0,after=k)
        freqvect.head=append(freqvect.head,k,after=k)
      }    }   }
  xval = c(0:N);  probvect=dbinom(c(0:N),N.binom,p.binom); expected= size*probvect
  res.HYP=consolidate.chisq.v2(expected, freqvect,xval,FALSE)
  expected.ord=res.HYP$expected.consol; freqvect.ord=res.HYP$freq.consol*size
```

```
  chi2.binom.ord=sum((freqvect.ord-expected.ord)^2/(expected.ord))
  chivect.binom.ord[i]=chi2.binom.ord;   df.binom=2
  pval.binom.ord=1-pchisq(chi2.binom.ord,df.binom)
  pval.chi.binom.ord[i]=pval.binom.ord
  adanvect=adan.dist.v2(c(0:N),binom.mu1,binom.mu2)
expected.adan=adanvect$prob*size
  res.adan=consolidate.chisq.v2(expected.adan,freqvect,xval,FALSE)
  expected.adan=res.adan$expected.consol; freqvect.adan=res.adan$freq.consol*size
  chi2.binom.adan=sum((freqvect.adan-expected.adan)^2/(expected.adan))
  chivect.binom.adan[i]=chi2.binom.adan; df.binommix=3
  pval.binom.adan=1-pchisq(chi2.binom.adan,df.binommix)
  pval.chi.binom.adan[i]=pval.binom.adan;   probvect.Ord=dbinom(D,N.binom,p.binom)
  Adan.vect=adan.dist.v2(D,binom.mu1,binom.mu2)
  Adan.class.binom[i]=Adan.vect$dist
  probvect.Adan=Adan.vect$prob; k.Adan=Adan.vect$k
  probvect.Ord[probvect.Ord==0]=0.0001 ; probvect.Adan[D>(k.Adan+1)]=0.0001
  h0=sum(log(probvect.Ord)); h1=sum(log(probvect.Adan))
  LR.val.binom[i]=-2*(h0-h1);   pval.LR.binom[i]=1-pchisq(LR.val.binom[i],df.binom)
}
binom.space=cbind(binom.space,pval.chi.binom.ord,chivect.binom.ord,Adan.class.bino
m,pval.chi.binom.adan,chivect.binom.adan,LR.val.binom,pval.LR.binom)
binom.accept.ord=binom.space[binom.space$pval.chi.binom.ord>=0.05,]
binom.accept.adan=binom.space[binom.space$pval.chi.binom.adan>=0.05,]
binom.mu1.est=NULL; binom.mu2.est=NULL; binom.mu3.est=NULL
for(i in 1:dim(binom.space)[1])
{  N.binom=binom.space[i,11];  p.binom=binom.space[i,12]
  binom.mu1.est=N.binom*p.binom;  binom.mu2.est=N.binom*p.binom*(1-p.binom)
  binom.mu3.est=N.binom*p.binom*(1-p.binom)*(1-2*p.binom) }
binom.space=cbind(binom.space,binom.mu1.est,binom.mu2.est,binom.mu3.est)
```

Appendix A.2. C++ code for Numeric Experiments

```
#include <math.h>
#include <conio.h>
#include <fstream>
#include <algorithm>
#include <functional>
#include <queue>
#include <vector>
#include <time.h>
#include <stdio.h>
#include <iostream>
#include <random>
#include <string>
#include <sstream>
#include <ctime>
#include <ios>
#include <cmath>
#include <ctime>
//#include "\Users\Mustafa Hekimoglu\source\Mylib\sqlite3.h"
#include<Windows.h>
#include <thread>
#include<process.h>
#include"C:\Users\Zülal\Downloads\boncuk.h"

using namespace std;
int calculatestatespace(int i, int maxDmax, int lt, int v, int N, int yvect[], int mvect[]);
int calculateindex(int yvect[], int mvect[], int lt, int v, int N, int maxDmax);
double calculateprobvect(int v, int mvect[], double pvect[], double probvect[], int distsupport);
double functionL(int y, int totqp, int mvect[], double probvect[], double holdrate, double backlograte);
```

```
double singlepercost3Dprint(int v, int y, int mvect[], int qr, int qpvect[], double cr, double cpvect[], double hold, double backlog, int d);

void convolution2araydist(double dist1[], double dist2[], double targetdist[], int maxsupport);

void calculatemvectdouble(int y, int mvect[], int mvectdoublebar[]);

long double binompdf2(int k, int n, double p);

const int v = 3;

const int LT = 1;

const int N = 5;

const int M = 10000000; //very large number;

double holdrate = 0.25;

double backlograte = 3;

double substitutionrate = 0;

double gamma = 0.5;

long double qrcost;

long double qpcost;

long double hc;

long double bc;

const double discountfactor = 0.995;

int setup = 1;

const double phi = 0.999; //0.929; // 0.65; //

const double eta = 0.01; //0.032; //0.9; //

char dbname[30];

char filename1[30];

char filename2[30];

const char* errMSG;

const char* tail;

char* zErrMsg = 0;

bool testflg = FALSE;

int horizon = 5;

struct states {
```

```
        int st;
        int per;
        long double optcost;
        int optqr;
        int optqpvect[v];
        int yvect[LT + 1];
        int mvect[v + 1];


        long double gammacost;
        float sales;
        float lostsales_exp;
        float carryc_exp;
        float eip_exp;
        float wos_exp;
        float salvagecost_exp;
};
int letter2ind(string letters);
void readparams(int ind, int& param1, double& param2, double& param3, double& param4, double& param5);
//int main(int argc, char* argv[])
int main()
{
        cout << "=====================================NEW RUN=============================================\n\n" << endl;
                cout << indexx << " " << horizon << " " << backlograte << " " << substitutionrate << " " << holdrate << " " << gamma << endl;
        int i = 0, j, jj;
        char c;
        int d;
        int d1, d2, d3;
```

```
        long double prob0, prob1, prob2, prob3;
        float mu = 0.8, sigma = 1, epsilon = 0.001;
        int yvect[LT + 1];
        int yvectnew[LT + 1];
        int mvect[v + 1];
        int mvectdoublebar[v + 1];
        int mvectnew[v + 1];
        double ptildevect[v + 1];
        double xivect[v] = { 100,200,300 }; //{ 100,200,300 }; //{ 50, 250, 450 };   //100;
//{ 100,200,300,400,500 };
        double cpvect[v] = { 100,200,300 };
        double cr = 100;
        double cp0 = 40;
        double alpha = 3, r = 0.01, s = -0.00001;
        long double probvect[N + 1];
        long double probvect2[N + 1];
        long double targetprobvect[N + 1];
        long double costper;
        long double gammacost;
        long double opttotcost, totcost;
        long double nextpercost;
        long double nextpergammacost;
        long double hold = cr * holdrate;
        long double backlog = backlograte * cr;
        long double substitutioncost = substitutionrate * cr;
        int maxDmax = maxdemandnormal(mu, sigma, epsilon);
        int maxyupperbound = (N + 1) * (LT + 1); //(N + 1) * (v + 1);
        int statesize = pow((double)N + 1, v) * pow(maxyupperbound, (LT + 1));
//pow((double)N + 1, v)*pow(N + 1, (LT + 1));
        int statesize2 = pow((double)N + 1, v) * pow(N + 1, (LT + 1) * (LT + 1));
        cout << "Total State Size Per Period is: " << statesize << endl;
```

```
int qr;
int qpvect[v];
double optcost;
int optqr;
int optqvect[v];
int searchspace = pow((double)N + 1, v);
int orderuptoqr;
long double ps1, ps2, ps3;
long double pr;
int totprintedparts;
int totprintedpartsafterchange;
int totorgpartsafterchange;
int totprintedpartsnew;
int totaldemand;
time_t now = time(NULL);
char dt[26];
states* statevect = new states[statesize];
states* statevectnextper = new states[statesize];
if (setup == 1)
{
	ptildevect[0] = 0.5;
	for (i = 1; i <= v; i++)
	{
		ptildevect[i] = ptildevect[0] + 1 / (3 + 0.01 * xivect[i - 1] - 0.000001 * pow(xivect[i - 1], 2));
		cpvect[i - 1] = 10 * pow(xivect[i - 1], gamma);
	}
}
std::ostringstream paramheader;
paramheader << "Parameters: Horizon: " << horizon << " HoldingRate:" << holdrate << " BacklogRate:" << backlograte << " SubstitutionRate: " << substitutionrate
```

```
<< " xivect=(" << xivect[0] << "," << xivect[1] << "," << xivect[2] << "," << xivect[3] << ")";

        paramheader << " alpha=" << alpha << ", r=" << r << ", s=" << s << ", ptilde.vect = (" << ptildevect[0] << "," << ptildevect[1] << "," << ptildevect[2] << "," << ptildevect[3] << ")";

        paramheader << " gamma= " << gamma << " cpr.vect = (" << cr << "," << cpvect[0] << "," << cpvect[1] << "," << cpvect[2] << ")";

        std::string parameterheader = paramheader.str();

        cout << parameterheader;

        cout << "\n\nLaser Energy Density: \t xi.1=" << xivect[0] << "xi.2=" << xivect[1] << "xi.3=" << xivect[2] << endl;

        cout << "Failure Rates: \t\t p.org=" << ptildevect[0] << " p.1=" << ptildevect[1] << " p.2=" << ptildevect[2] << " p.3=" << ptildevect[3] << endl;

        cout << "Acquisition Costs: \t cr=" << cr << " cp1=" << cpvect[0] << " cp2=" << cpvect[1] << " cp3=" << cpvect[2] << endl;

        ofstream outputfile, outputsub, resultfile, logfile, resultfile0;

                resultfile0.open("resultfile0.txt", std::ofstream::out | std::ofstream::app);

        resultfile0 << "horizon" << "\t" << "yvect[0]" << "\t" << "yvect[LT]" << "\t" << "mvect[0]" << "\t" << "mvect[1]" << "\t" << "mvect[2]" << "\t" << "mvect[3]" << "\t" << "d" << "\t" << "d1" << "\t" << "d2" << "\t" << "d3" << "\t" << "totaldemand" << "\t" << "qr" << "\t" << "qpvect[0]" << "\t" << "qpvect[1]" << "\t" << "qpvect[2]" << "\t" << "hc" << "\t" << "bc" << "\t" << "qpcost" << "\t" << "qrcost" << "\t" << "prob.Dr" << "\t" << "prob.Dp1" << "\t" << "prob.Dp2" << "\t" << "prob.Dp3" << "\t" << "costper" << "\t" << "gammacost" << "\t" << "nextpercost" << "\t" << "nextpergammacost" << "\t" << "totcost" << endl;

        ofstream testfile;

        testflg = true;

        if (testflg)

        {

                testfile.open("filetest.txt", std::ofstream::out | std::ofstream::app);

                testfile << setprecision(30) << "cpvect[0]" << "\t" << cpvect[0] << "cpvect[1]" << "\t" << cpvect[1] << "cpvect[2]" << "\t" << cpvect[2] << endl;

                testfile << "i" << " \t " << "j" << "\t" << "jj" << "\t" << "yvect[0]" << "\t" << "yvect[LT]" << " \t " << "mvect[0]" << "\t" << "mvect[1]" << "\t" << "mvect[2]" << "\t" << "mvect[3]" << endl;

                for (i = 0; i < statesize; i++)

                {
```

```
                    calculatestatespace(i, maxyupperbound, LT, v, N + 1, yvect, mvect);

                    mvect[0] = N; totprintedparts = 0;
                    for (j = 1; j <= v; j++)
                        totprintedparts += mvect[j];
                    mvect[0] -= totprintedparts;
                    if (totprintedparts > N)
                        continue;
                    jj = calculateindex(yvect, mvect, LT, v, N + 1, maxyupperbound);
                    testfile << i << " \t : \t " << j - 3 << "\t" << jj << "\t" << yvect[0] << "\t" << yvect[LT] << " \t == \t" << mvect[0] << "\t" << mvect[1] << "\t" << mvect[2] << "\t" << mvect[3] << endl;
                }
            }
            outputsub.open(filename2);
            outputsub << parameterheader << endl;

            //LAST PERIOD
            std::cout << "\n\n" << "Period " << horizon << " starts!" << endl;
            for (i = 0; i < statesize; i++)
            {
                calculatestatespace(i, maxyupperbound, LT, v, N + 1, yvect, mvect);
                mvect[0] = N;
                totprintedparts = 0;
                if (yvect[0] + yvect[LT] > maxyupperbound)
                    continue;

                for (j = 1; j <= v; j++)
                    totprintedparts += mvect[j];
                if (totprintedparts > N)
                    continue;
```

```
mvect[0] -= totprintedparts;
optqr = N;
optqvect[0] = 0; optqvect[1] = 0; optqvect[2] = 0;
opttotcost = M;
for (orderuptoqr = 0; orderuptoqr <= maxyupperbound; orderuptoqr++)  // qr search başlangıcı
{
    gammacost = 0;
    qr = maxoftwo(orderuptoqr - yvect[0] - yvect[LT], 0);
    totcost = qr * cr; //+ minoftwo(y, totprintedparts) * substitutioncost;
    qrcost = totcost;
    hc = 0;
    bc = 0;
    for (d = 0; d <= mvect[0]; d++)
    {
        prob0 = binompdf2(d, mvect[0], ptildevect[0]);
        for (d1 = 0; d1 <= mvect[1]; d1++)
        {
            prob1 = binompdf2(d1, mvect[1], ptildevect[1]);
            for (d2 = 0; d2 <= mvect[2]; d2++)
            {
prob2 = binompdf2(d2, mvect[2], ptildevect[2]);
                for (d3 = 0; d3 <= mvect[3]; d3++)
                {
    prob3 = binompdf2(d3, mvect[3], ptildevect[3]);
                    totaldemand = d + d1 + d2 + d3;
                    optcost = M;
                    for (j = 0; j < searchspace; j++)
                    {
        qpvect[0] = j % (N + 1);
```

```
                qpvect[1] = ((int)(j / pow(N + 1, 1))) % (N + 1);
                qpvect[2] = ((int)(j / pow(N + 1, 2))) % (N + 1);
if (qpvect[0] + qpvect[1] + qpvect[2] != maxoftwo(0, totaldemand - yvect[0] - yvect[LT]))
                        continue;
costper = qpvect[0] * cpvect[0] + qpvect[1] * cpvect[1] + qpvect[2] * cpvect[2];
costper += hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand) + backlog * maxoftwo(0, totaldemand - yvect[0] - qpvect[0] - qpvect[1] - qpvect[2] - yvect[LT]);
hc = hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand);
bc = backlog * maxoftwo(0, totaldemand - yvect[LT] - yvect[0] - qpvect[0] - qpvect[1] - qpvect[2]);
                        if (costper < optcost)
                        {
                optqvect[0] = qpvect[0];
                optqvect[1] = qpvect[1];
                        optqvect[2] = qpvect[2];
                                optcost = costper;
                        }
                pr = prob0; ps1 = prob1; ps2 = prob2; ps3 = prob3;
        qpcost = qpvect[0] * cpvect[0] + qpvect[1] * cpvect[1] + qpvect[2] * cpvect[2];

                        } //END OF qpvect SEARCH
        totcost += optcost * prob0 * prob1 * prob2 * prob3;
                        if (qr == 0)
                        {
gammacost += (optqvect[0] * cpvect[0] + optqvect[1] * cpvect[1] + optqvect[2] * cpvect[2] + hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand) + backlog * maxoftwo(0, totaldemand - yvect[LT] - yvect[0] - qpvect[0] - qpvect[1] - qpvect[2])) * (prob0 * prob1 * prob2 * prob3);
                        }
                } //CONDITIONAL PROBABILITY
        } //CONDITIONAL PROBABILITY
```

```
                } //CONDITIONAL PROBABILITY
        } //CONDITIONAL PROBABILITY

        if (qr == 0)
        {
                statevect[i].gammacost = gammacost;
        }

        if (opttotcost > totcost)
        {
                optqr = qr;
                opttotcost = totcost;
        }
}  //END OF qr SEARCH
statevect[i].optqr = optqr;
statevect[i].optcost = opttotcost;
statevect[i].per = horizon;
statevect[i].yvect[0] = yvect[0];
statevect[i].yvect[1] = yvect[1];
statevect[i].mvect[0] = mvect[0];
statevect[i].mvect[1] = mvect[1];
statevect[i].mvect[2] = mvect[2];
statevect[i].mvect[3] = mvect[3];
statevect[i].optqpvect[0] = optqvect[0];
statevect[i].optqpvect[1] = optqvect[1];
statevect[i].optqpvect[2] = optqvect[2];

if ((i % 500) == 0)
{
        now = time(NULL);
        ctime_s(dt, sizeof dt, &now);
```

```
                cout << "Period " << horizon << " State Index: " << i << ": State=
(" << yvect[0] << " " << yvect[LT] << " || " << mvect[0] << " " << mvect[1] << " " <<
mvect[2] << " " << mvect[3] << ") " << " qr=" << optqr << " Cost=" << opttotcost << "
Time: " << dt;

            }

        } //END OF WHILE LOOP

        std::cout << "Period " << horizon << " is complete! Writing Starts!" << endl;

        outputfile.open("outputfile.txt", std::ofstream::out | std::ofstream::app);

        outputfile << "horizon" << "\t" << "yvect[0]" << "\t" << "yvect[LT]" << "\t" <<
"mvect[0]" << "\t" << "mvect[1]" << "\t" << "mvect[2]" << "\t" << "mvect[3]" << "\t"
<< "optqr" << "\t" << "optqpvect[0]" << "\t" << "optqpvect[1]" << "\t" << "optqpvect[2]"
<< "\t" << "optcost" << "\t" << "gammacost" << endl;

        for (i = 0; i < statesize; i++)

        {

            calculatestatespace(i, maxyupperbound, LT, v, N + 1, yvect, mvect);

            mvect[0] = N; totprintedparts = 0;

            if (yvect[0] + yvect[LT] > maxyupperbound)

                continue;

            for (j = 1; j <= v; j++)

                totprintedparts += mvect[j];


            if (totprintedparts > N)

                continue;

            mvect[0] -= totprintedparts;

            outputfile << statevect[i].per << "\t" << statevect[i].yvect[0] << "\t" <<
statevect[i].yvect[1] << "\t" << statevect[i].mvect[0] << "\t" << statevect[i].mvect[1] <<
"\t" << statevect[i].mvect[2] << "\t " << statevect[i].mvect[3] << "\t" << statevect[i].optqr
<< "\t" << statevect[i].optqpvect[0] << "\t" << statevect[i].optqpvect[1] << "\t" <<
statevect[i].optqpvect[2] << "\t" << statevect[i].optcost << "\t" << statevect[i].gammacost
<< endl;

        }

        horizon--;

        //NEXT PERIOD

        while (horizon >= 1)

        {
```

```
cout << "Period " << horizon << " starts!" << endl;
for (i = 0; i < statesize; i++)
{
calculatestatespace(i, maxyupperbound, LT, v, N + 1, yvect, mvect);
        mvect[0] = N;
        totprintedparts = 0;
        if (yvect[0] + yvect[LT] > maxyupperbound)
                continue;
        for (j = 1; j <= v; j++)
                totprintedparts += mvect[j];

        if (totprintedparts > N)
                continue;
        mvect[0] -= totprintedparts;
        optqr = N;
        optqvect[0] = 0; optqvect[1] = 0; optqvect[2] = 0;
        opttotcost = M;
        for (orderuptoqr = 0; orderuptoqr < maxyupperbound; orderuptoqr++)  //N * (LT + 1)
        {
                qr = maxoftwo(orderuptoqr - yvect[0] - yvect[LT], 0);
                gammacost = 0;
    totcost = qr * cr;// +minoftwo(y, totprintedparts) * substitutioncost;
                qrcost = totcost;
                hc = 0;
                bc = 0;
                for (d = 0; d <= mvect[0]; d++) //totorgpartsafterchange
                {
                        prob0 = binompdf2(d, mvect[0], ptildevect[0]);

        for (d1 = 0; d1 <= mvect[1]; d1++) //mvectdoublebar
```

```
                                        {
                    prob1 = binompdf2(d1, mvect[1], ptildevect[1]);
                                                for (d2 = 0; d2 <= mvect[2]; d2++)
                                                {
                                prob2 = binompdf2(d2, mvect[2], ptildevect[2]);
                                                        for (d3 = 0; d3 <= mvect[3]; d3++)
                                                        {
                                prob3 = binompdf2(d3, mvect[3], ptildevect[3]);
                                                        totaldemand = d + d1 + d2 + d3;
                                                                optcost = M;
                                                for (j = 0; j < searchspace; j++)
                                                                {
                                                                qpvect[0] = j % (N + 1);
                                                        qpvect[1] = ((int)(j / pow(N + 1, 1))) % (N + 1);
                                                        qpvect[2] = ((int)(j / pow(N + 1, 2))) % (N + 1);
if (qpvect[0] + qpvect[1] + qpvect[2] != maxoftwo(0, totaldemand - yvect[0] - yvect[LT]))
                                                                                continue;
costper = qpvect[0] * cpvect[0] + qpvect[1] * cpvect[1] + qpvect[2] * cpvect[2];
costper += hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand) + backlog * maxoftwo(0, totaldemand - yvect[0] - qpvect[0] - qpvect[1] - qpvect[2] - yvect[LT]);
hc = hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand);
bc = backlog * maxoftwo(0, totaldemand - yvect[LT] - yvect[0] - qpvect[0] - qpvect[1] - qpvect[2]);
                                                                        yvectnew[LT] = qr;
                                                yvectnew[0] = maxoftwo(0, yvect[0] + yvect[LT] - totaldemand);


                    mvectnew[0] = mvect[0] - d + min(yvect[0] + yvect[LT], totaldemand);
                    mvectnew[1] = mvect[1] - d1 + qpvect[0];
                    mvectnew[2] = mvect[2] - d2 + qpvect[1];
                    mvectnew[3] = mvect[3] - d3 + qpvect[2];
jj = calculateindex(yvectnew, mvectnew, LT, v, N + 1, maxyupperbound);
```

```
nextpercost = statevect[jj].optcost;
nextpergammacost = statevect[jj].gammacost;
costper += nextpercost * discountfactor * phi + nextpergammacost * discountfactor * (1 - phi);
                                                    if (costper < optcost)
                                                    {
                optqvect[0] = qpvect[0]; optqvect[1] = qpvect[1]; optqvect[2] = qpvect[2];
                                                optcost = costper;
                                                    }
                        pr = prob0; ps1 = prob1; ps2 = prob2; ps3 = prob3;
qpcost = qpvect[0] * cpvect[0] + qpvect[1] * cpvect[1] + qpvect[2] * cpvect[2];
                                                } //End of qp search
        mvectnew[0] = mvect[0] - d + min(yvect[0] + yvect[LT], totaldemand);
                        mvectnew[1] = mvect[1] - d1 + optqvect[0];
                                mvectnew[2] = mvect[2] - d2 + optqvect[1];
                                    mvectnew[3] = mvect[3] - d3 + optqvect[2];
jj = calculateindex(yvectnew, mvectnew, LT, v, N + 1, maxyupperbound);
nextpercost = statevect[jj].optcost;
nextpergammacost = statevect[jj].gammacost;
totcost += optcost * prob0 * prob1 * prob2 * prob3;
                                                if (qr == 0)
                                                {
                                                    gammacost += (optqvect[0] * cpvect[0] + optqvect[1] * cpvect[1] + optqvect[2] * cpvect[2] + hold * maxoftwo(0, yvect[0] + yvect[LT] - totaldemand) + backlog * maxoftwo(0, totaldemand - yvect[0] - optqvect[0] - optqvect[1] - optqvect[2] - yvect[LT]) + nextpercost * discountfactor * eta + nextpergammacost * discountfactor * (1 - eta)) * (prob0 * prob1 * prob2 * prob3);
                                                }
                                            }//CONDITIONAL PROB
                                    } //CONDITIONAL PROBABILITY
                                } //CONDITIONAL PROBABILITY
                            } //CONDITIONAL PROBABILITY
```

```
                    if (qr == 0)
                    {
                        statevectnextper[i].gammacost = gammacost;
                    }
                    if (opttotcost > totcost)
                    {
                        optqr = qr;
                        opttotcost = totcost;
                    }
                }//END OF QR SEARCH

                statevectnextper[i].optqr = optqr;
                statevectnextper[i].optcost = opttotcost;
                statevectnextper[i].per = horizon;
                statevectnextper[i].yvect[0] = yvect[0];
                statevectnextper[i].yvect[1] = yvect[1];
                statevectnextper[i].mvect[0] = mvect[0];
                statevectnextper[i].mvect[1] = mvect[1];
                statevectnextper[i].mvect[2] = mvect[2];
                statevectnextper[i].mvect[3] = mvect[3];
                if ((i % 500) == 0)
                {
                    now = time(NULL);
                    ctime_s(dt, sizeof dt, &now);
                    cout << "Period " << horizon << " State Index: " << i << ": State= (" << yvect[0] << " " << yvect[LT] << " || " << mvect[0] << " " << mvect[1] << " " << mvect[2] << " " << mvect[3] << ") " << " qr=" << optqr << " Cost=" << opttotcost << " Time: " << dt;
                }
            }
            std::cout << "Period " << horizon << " is complete! Writing Starts!" << endl;
```

```
		for (i = 0; i < statesize; i++)
		{
			statevect[i].optqr = statevectnextper[i].optqr;
			statevect[i].optcost = statevectnextper[i].optcost;
			statevect[i].per = statevectnextper[i].per;
			statevect[i].gammacost = statevectnextper[i].gammacost;
	calculatestatespace(i, maxyupperbound, LT, v, N + 1, yvect, mvect);
			mvect[0] = N; totprintedparts = 0;
			if (yvect[0] + yvect[LT] > maxyupperbound)
				continue;
			for (j = 1; j <= v; j++)
				totprintedparts += mvect[j];
			if (totprintedparts > N)
				continue;
			mvect[0] -= totprintedparts;
			outputfile << statevect[i].per << "\t" << statevect[i].yvect[0] << "\t" << statevect[i].yvect[1] << "\t" << statevect[i].mvect[0] << "\t" << statevect[i].mvect[1] << "\t" << statevect[i].mvect[2] << "\t " << statevect[i].mvect[3] << "\t" << statevect[i].optqr << "\t" << statevect[i].optcost << "\t" << statevect[i].gammacost << endl;
		}
		horizon--;
	}
	outputfile.close();
	testfile.close();
	resultfile0.close(); outputsub.close();
	resultfile.open("resultfile.txt", std::ofstream::out | std::ofstream::app);
	resultfile << indexx << "\t" << parameterheader << "\t" << statevect[0].per << "\t" << statevect[0].optcost << "\t" << statevect[0].optqr << endl;
	resultfile.close();
	delete[] statevect;
	delete[] statevectnextper;
	return 0;
```

```
}
double calculateprobvect(int v, int mvect[], double pvect[], double probvect[], int distsupport)
{
	int i, j;
	double dist1[N + 1];
	double dist2[N + 1];
	double targetdist[N + 1];
	for (j = 0; j <= N; j++)
	{
		dist1[j] = binompdf(j, mvect[0], pvect[0]);
		dist2[j] = binompdf(j, mvect[1], pvect[1]);
	}
	convolution2araydist(dist1, dist2, targetdist, N);
	for (i = 3; i <= v; i++)
	{
		for (j = 0; j <= N; j++)
		{
			dist1[j] = binompdf(j, mvect[i], pvect[i]);
			dist2[j] = targetdist[j];
		}
		convolution2araydist(dist1, dist2, targetdist, N);
	}
	for (j = 0; j <= N; j++)
		probvect[j] = targetdist[j];
	return 0;
}
void calculatemvectdouble(int y, int mvect[], int mvectdoublebar[])
{
	int i, j;
	int mtemp;
```

```
	mtemp = mvect[1];
	mvectdoublebar[1] = (int)maxoftwo(mtemp - y, 0);
	mtemp = mvect[2];
	mvectdoublebar[2] = (int)maxoftwo((int)mvect[2] - (int)maxoftwo(y - (int)mvect[1], 0), 0);
	mtemp = mvect[1] + mvect[2];
	for (i = 3; i <= v; i++)
	{
		mvectdoublebar[i] = maxoftwo(mvect[i] - maxoftwo(y - mtemp, 0), 0);
		mtemp += mvect[i];
	}
	mvectdoublebar[0] = mvect[0];
}

double functionL(int y, int totqp, int mvect[], double probvect[], double holdrate, double backlograte)
{
	int s = 0;
	double expectedcost = 0;
	for (s = 0; s <= N; s++)
	{
expectedcost += holdrate * maxoftwo(y - s, 0) * probvect[s] + maxoftwo(s - y - totqp, 0) * probvect[s] * backlograte;
	}
	return expectedcost;
}

double singlepercost3Dprint(int v, int y, int mvect[], int qr, int qpvect[], double cr, double cpvect[], double hold, double backlog, int d)
{
	double cost = 0;
		int totm = 0;
	int totqp = 0;
```

```
	for (int i = 0; i < v; i++)
	{
		totm += mvect[i];
		cost += (double)qpvect[i] * cpvect[i];
		totqp += qpvect[i];
	}
	cost += hold * maxoftwo(y - d, 0) + maxoftwo(d - y - totqp, 0) * backlog;
	return cost;
}
int calculatestatespace(int i, int maxDmax, int lt, int v, int N, int yvect[], int mvect[])
{
	int lengthd = maxDmax;  //+1
	int lengthv = N;
	int l;
	int num = lengthd, denom = 1;
	int num2 = pow(lengthd, lt + 1) * lengthv, denom2 = pow(lengthd, lt + 1);
	if (lt == 1)
	{
		for (l = 0; l < lt; l++)
		{
			yvect[lt - l - 1] = (i % num) / denom;
			denom = num;
			num *= lengthd;
		}
		yvect[lt] = (i % num) / denom;
		for (l = 0; l < v; l++)
		{
			mvect[l + 1] = (i % num2) / denom2;
			denom2 = num2;
			num2 *= lengthv;
```

```
                }
        }
        return 0;
}
int calculateindex(int yvect[], int mvect[], int lt, int v, int N, int maxDmax)
{       int multp1 = 0, multp2 = 0;
        int lengthd = maxDmax;
        int lengthv = N;
        int l, res = 0;
        if (lt == 1)
        {
                for (l = 0; l < lt; l++)
                {
                        multp1 += yvect[lt - l - 1] * (int)pow(lengthd, l);
                }
                multp1 += (int)pow((double)lengthd, (double)lt) * yvect[lt];
                multp2 = (int)pow((double)lengthd, (double)lt + 1);

                for (l = 0; l < v; l++)
                {
                        multp1 += multp2 * mvect[l + 1];
                        multp2 *= lengthv;
                }
        }
        /* if (lt == 3)
        { //RECALL THAT y[0] is IL with dmax  and y[1] and y[2]
        multp1 = yvect[0] + yvect[1] * (1 + (double)dmax) + yvect[2] * (1 + (double)dmax)*(1 + (double)dmaxper);
        multp2 = (int)pow((double)(1 + dmaxper), (double)(lt - 1))*(1 + dmax)*(horizon - per);
        }
```

```
	*/
	res = multp1;
	return res;
}
long double binompdf2(int k, int n, double p)
{
	long double res;
	if (k <= n)
		res = pow(p, (long double)k) * pow(1 - p, (long double)(n - k)) * comb(n, k);
	if (k > n)
		res = 0;
	return res;
}
void convolution2araydist(double dist1[], double dist2[], double targetdist[], int maxsupport)
{	int i = 1, k, j;
	float temp[bignum];
	// float tempscalar;
	for (k = 0; k <= maxsupport; k++)
	{
		temp[k] = 0;
		for (j = 0; j <= k; j++)
			temp[k] += dist1[j] * dist2[k - j];
	}
	for (k = 0; k <= maxsupport; k++)
		targetdist[k] = temp[k];
}
int letter2ind(string letters)
{	int a1 = letters[0];
	int a2 = letters[1];
```

```
	int a3 = letters[2];
	int a4 = letters[3];
	int a5 = letters[4];
	int num = (a1 - 97) * 10000 + (a2 - 97) * 1000 + (a3 - 97) * 100 + (a4 - 97) * 10 + (a5 - 97);
	cout << "Run index is: " << num << endl;
	return num; }
//readparams(indexx, horizon, backlograte, substitutionrate, holdrate, gamma);
void readparams(int ind, int& param1, double& param2, double& param3, double& param4, double& param5)
{	int trial, i, j; char c;
	double param[6];
	double temp;
	ifstream input;
	input.open("params.txt");
	if (input.is_open()) //ADDED. LOD INDEX
		cout << "Parameter file is opened successfully." << endl;
	else
	{
		cout << "Unable to open the parameter file" << endl;
		cout << "quitting...";
		cout << '\a';
		cin >> c;
		exit(0);
	}
	for (i = 1; i <= ind; i++)
	{
		if (i != ind)
		{
			for (j = 0; j < 5; j++)
				input >> temp;
```

```
        }
        else
        {
                for (j = 0; j < 5; j++)
                {
                        input >> param[j];
                }
        }
}
input.close()
for (j = 0; j < 5; j++)
{               switch (j)
        {
        case 0:
        {
                param1 = (int)param[j]; break;
        }
        case 1:
        {
                param2 = param[j]; break;
        }
        case 2:
        {
                param3 = param[j]; break;
        }
        case 3:
        {
                param4 = param[j]; break;
        }
        case 4:
        {
```

```
                    param5 = param[j];
                }
                }
            }
        }
```

CURRICULUM VITAE

Personal Information:

Name and Surname: ZÜLAL İŞLER

Academic Background

Bachelor's Degree Education

Business Administration (2013-2019) at Kadir Has University

Industrial Engineering (2015-2019) at Kadir Has University

Post Graduate Education

Master of Science (2019-) in Industrial Engineering at Kadir Has University

Foreign Languages

English: Advanced

German: Intermediate

Work Experience

Research Assistant (2019-) at Kadir Has University