ISSN 1303-9709 G.U. Journal of Science 17(1): 37-47, 2004

A MULTIVARIATE NORMAL RANDOM VECTOR GENERATOR

Mustafa Y.ATA
Gazi Üniversitesi, Fen-Edebiyat Fakültesi, İstatistik Bölümü, 06570, Ankara, TÜRKİYE
e-mail:yavuzata@gazi.edu.tr

ABSTRACT

An explicit procedure for generating multivariate normal random vector is presented. Using a lower triangular matrix **L** decomposed from the covariance matrix Σ by the **CHOLESKY** method, the algorithm of the generator consists of the transformation of a p-dimensional standard normal variate **z**, elements of which are obtained by the Box-Muller procedure, into a *p*-dimensional normal random sample $\mathbf{x}=\mathbf{Lz}+\mu$ from the distribution $\mathbf{X}\sim N(\mu,\Sigma)$. The efficiency of the proposed procedure is exhibited by a Monte Carlo test of the algorithm which showed that the generator is highly reliable.

Key Words: Multivariate random vector generation, Cholesky, decomposition, Monte Carlo simulation.

ÇOK-DEĞİŞKENLİ NORMAL RASSAL BİR YÖNEY ÜRETECİ

ÖZET

Çok-değişkenli normal rassal bir yöneyin üretimi için açık bir yordam verildi. **CHOLESKY** yöntemi ile Σ kovaryans dizeyinden ayrıştırılmış **L** alt üçgen dizeyini kullanan üretecin algoritması, elemanları Box-Muller yöntemi ile elde edilen *p*-boyutlu standart normal **z** değişkeninin, $\mathbf{X}\sim N(\mu,\Sigma)$ dağılımından *p*-boyutlu rassal bir $\mathbf{x}=\mathbf{Lz}+\mu$ örneğine dönüştürülmesinden oluşmaktadır. Önerilen yordamın etkinliği, üretecin oldukça yüksek bir güvenirliği olduğunu gösteren bir Monte Carlo sınama ile sergilenmiştir.

*Anahtar Kelimeler: Çok-değişkenli rassal yöney üretme, **Cholesky** ayrıştırması, Monte Carlo benzetim.*

1.GİRİŞ

Her hangi bir çok-değişkenli istatistiksel tekniğin sağlamlığına ilişkin deneysel bir kanıt sağlama amacı ile yapılan Monte Carlo çalışmalarında sanal deney, kontrol grubu ile işlem grubunun beklenen değer yöneylerinin farklarının sınanmasında olduğu gibi tasarlanır. Böyle bir düzenlemede, kontrol grubu verileri değişmez bir biçimde çok-boyutlu hata terimlerinin bir birinden bağımsız, aynı ve normal dağıldığı standart varsayımının geçerli olduğu bir benzetim modelinden elde edilirken, işlem grubu verileri belli bir biçimde karıştırılarak bozulmuş çok-değişkenli normal dağılımlardan elde edilebilir. Dolayısı ile bu tür bir Monte Carlo çalışması için öncelikle, ulaşılabilir çok-değişkenli normal rassal bir yöney üretecinin varlığı ön koşuldur.

Her ne kadar (1) ve (2) gibi kimi standart matematiksel ve istatistiksel yazılım paketleri kullanılarak, çok-değişkenli normal bir dağılımdan rasgele örnekleme ve bir Monte Carlo deney programlanabilse de, Basic ya da Fortran gibi bir bilgisayar dilini iyi düzeyde bilen bir Monte Carlo deneyci, bu yazılımlara özgü programlama tekniğini öğrenmeye çaba harcamak yerine, tüm deneyi gerçekleştiren kendi programlarını özgürce yazmayı daha

1. INTRODUCTION

In the Monte Carlo studies with the aim to provide an empirical evidence on the robustness of some multivariate statistical techniques, the virtual experiment must be designed just like it is in testing the difference between the mean vector of the control group and that of the treatment group. In such a context, control group data are to be obtained invariably via a simulation model with the standard assumptions of independently, identically and normally distributed multidimensional error terms, whereas the data for treatment group might be obtained through the confounding of several multivariate normal distributions. So, for such a Monte Carlo study, the availability and the accessibility of a multivariate normal random vector generation procedure is a prerequisite.

Although it is evident that some of the standard statistical software packages are capable of sampling from a given multivariate normal distribution and of programming a Monte Carlo experiment, still some Monte Carlo experimenters having a good command of a low level computing language such as Basic, Fortran or C may find it more resonable and more joyfull to write their own programs performing the whole experiment, instead of

mantıklı ve zevkli bulabilir.

Şimdiki çalışmanın bu konuda bir ilk olmadığı açık. Bibliografik veri tabanlarında yer alan benzer çalışmaların sayısının oldukça kabarık olduğu görülüyor. (3-14), ve (15) bunlardan bir kaçı. Özellikle, (3), (5) ve (7)'deki yaklaşım, bu çalışmada benimsenen yaklaşımla belirgin bir benzerlik taşımaktadır. Ancak, bu tür çalışmalar bir bilgisayar yordamı üretmeye yönelik ve her kaynak program da onu yazanın özgün eseri olduğundan, aynı kuramsal temele dayalı olsalar bile, özgün niteliklerini ürünleri ile ortaya koyarlar. Bu nedenle, aynı amaca yönelik bu tür ürün sayısının çok olması, her alanda olduğu gibi bu ürünün tüketicileri açısından da olumlu bir durumdur. Şimdiki çalışma da, bu anlamda özgün olan bir ürünü kullanıcıların beğenisine ve hizmetine sunmaktadır. Şimdiki çalışmanın rakibi gibi görünen (3) ve (5)'in ürünlerine henüz ulaşamadığımızdan, (16) daki tür bir karşılaştırma olanağı bulunamadı. Bu nedenle, şimdiki ile birlikte sayıları dördü bulan ve kaynak kodu açık bir biçimde verilen çok-boyutlu normal rassal yöney üreteç seçeneklerini değerlendirerek aralarından en uygun olanını seçmek şimdilik kullanıcının kendisine kalmaktadır.

Öte yandan, yordamın dayandığı kuramsal bilgi; matematik, istatistik ve bilgisayar bilimlerinin kesişim bölgesindeki kaynaklarda ve dağınık olduğundan, bu temelin açık ve özlü olarak bir arada sunulması ile hiç olmazsa bu tür kaynaklara ulaşmada şimdiki ürün üreticisinin yaşadığı sıkıntıyı kullanıcının da aynı boyutta yaşamaması amaçlanmıştır. İzleyen bölümde, bundan böyle NRYJ olarak anılacak olan, çok-değişkenli Normal Rassal Yöney üretecinin kuramsal dayanağı, bir teorem biçiminde ifade edilecek. Daha sonra, NRYJ ve bağlı diğer alt yordamların algoritmaları sunulacak ve son olarak da, önerilen yordamın güvenirliği bir Monte Carlo sınamadan geçirilecektir.

Kullanıcıların uygun görecekleri bir dilde uyarlamasını kolayca yapabilecekleri biçimde algoritmaların verilmesine özen gösterildi. Monte Carlo sınama için tasarlanan deney QuickBasic dili kullanılarak programlandı ve derlendi. Monte Carlo deneye ilişkin ana programda alt-yordam olarak yer alan, NRYJ ve bağlı diğer alt yordamlarla, bu alt-yordamları kullanan bir ana-yordam örneğinin QuickBasic uyarlamaları açık biçimde **EK:B**'de verilmiştir. Monte Carlo sınamayı gerçekleştiren program "kendi başına çalışabilir derlenmiş bir dosya" biçiminde yazardan sağlanabilir ve istenirse burada verilen örnekten farklı parametre değerleri de girilerek, üretecin güvenilirliği ve etkinliği farklı deneysel tasarım noktalarında sınanabilir.

2. ÜRETECİNİN KURAMSAL DAYANAĞI

NRYJ üretecinin kuramsal dayanağı, aşağıda verilen teoremle ifade edilebilir:

Teorem: Eğer $\mathbf{Z} = (Z_1, \ldots\ldots, Z_p)'$,

spending so much effort to learn the special programming features of those softwares like as (1) and (2).

It is evident that the present work is not the first one on the subject, since the articles and texts dealing with the subject are numerous. (3-14), and (15) are a few of them just to be cited. Especially, the common approach of (3),(5) and (7), are quite similar with the one adopted in the present work. However, these kind of works are inclined to produce some sort of computer routines and even though they are based on the same theoretical base, they preserve their originality, since every source of programming is an original product of its author. In fact the multiplicity of the product is a preferable situation from the point of the consumers of this product as it is true for any product. The present work too submits a product which is original in this sense to the service and discrimination of the users. Since we have not yet reached the products of the likely competitors (3) and (5) of the present work, a comparative study of the competetive works like in (16) could not be carried out. Therefore, for the time being, it is up to the potential users to discriminate the optimal one among the four alternative multivariate normal random vector generators, including the present one, all of which them give the open source codes of the generator.

On the other hand, because the theoretical information on which the procedure is based on scatterred in the joint region of mathematics, statistics, and computer sciences sources, it is aimed at making it possible for at least the user in not having to go through the same difficulty that the producer of the present product had to practice by presenting this base in a clear, dense and integrated manner . In the following section, the theoretical base of the Normal Random Vector Generator will be referenced as NRYJ afterwards from now on. Then, the algorithms of NRYJ and the related procedures will be given, and finally the reliability of the proposed procedure will be passed through a Monte Carlo test.

Care was taken in giving the algorithms so that the users could easily implement them in a language that they prefered. The experiment designed for the Monte Carlo test was coded using QuickBasic language and compiled. A QuickBasic implementations of the NRYJ and the related subroutines which take place in the main of the Monte Carlo experiment together with a sample main using these subroutines was exhibited in the **Appendix:B**. The routine realizing the Monte Carlo experiment can be obtained from the author in the form of a stand-alone executable compiled program file and if it is liked, the reliability and efficiency of the generator can be tested in different design points by inputting parameter values different from the ones given in the example.

2.THEORETICAL BASE of the GENERATOR

The following theorem constitutes the theoretical base of the NRYJ algorithm:

Theorem: If $\mathbf{Z} = (Z_1, \ldots\ldots, Z_p)'$ is a random vector whose elements are random variables distributed independently and in accordance with the density function,

$$f_i(z_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2}, \qquad i = 1(1)p \qquad [1]$$

yoğunluk fonksiyonlu bir birinden bağımsız rassal değişkenlerin rassal bir yöneyi ise,

$$\mathbf{X} = \mathbf{LZ} + \boldsymbol{\mu} \qquad [2]$$

dönüşümü ile elde edilen $\mathbf{X} = (X_1, \ldots\ldots, X_p)'$ rassal yöneyinin olasılık yoğunluk fonksiyonu, $\boldsymbol{\mu}=\langle \mathbf{X} \rangle$, **X**'in p-boyutlu dikey beklenen değer yöneyi; $\boldsymbol{\Sigma}=\langle (\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})' \rangle$, i. satır ve j. sütun elemanı σ_{ij} olan pxp boyutlu pozitif tanımlı simetrik kovaryans dizeyi; ve **L**,

$$\boldsymbol{\Sigma} = \mathbf{LL}'. \qquad [3]$$

ilişkisini sağlayan pxp boyutlu alt üçgen dizeyi olmak üzere,

$$\varphi(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \qquad [4]$$

olur.

İspat: p tane birbirinden bağımsız z_i değişkeninin olasılık yoğunluk fonksiyonu [1] deki gibi olsun. O zaman, $\mathbf{z} = (z_1, \ldots\ldots, z_p)$ yöneyinin olasılık yoğunluk fonksiyonu,

$$g(\mathbf{z}) = \prod_{i=1}^{n} f_i(z_i) = \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\sum_{i=1}^{n} z_i^2}. \qquad [5]$$

olur. [2]'den $\mathbf{Z} = \mathbf{L}^{-1}(\mathbf{X}-\boldsymbol{\mu})$ dönüşümü tanımlanarak değişkenlerin dönüşüm yöntemi (17) uygulanırsa,

$$\begin{aligned} h(\mathbf{x}) &= g\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\left|\frac{\partial \mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})}{\partial \mathbf{x}}\right| \\ &= \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)'\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)} \left|\mathbf{L}^{-1}\right| \\ &= \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \\ &= \varphi(\mathbf{x}). \end{aligned}$$

olduğu görülür.

3. NRYJ'nün ALGORİTMASI

Yukarıdaki teoremin sonucu olarak, $\mathbf{X} \sim \mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ dağılımından rasgele örnekleme için önerilen NRYJ ve NRYJ için gerekli diğer alt-yordamların algoritmaları, bu alt-yordamların bir Monte Carlo Deney ana-yordamı içinde kullanılacağı göz önünde tutularak tasarlandı. Etkinlik açısından; $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, ve örnek çapı n gibi yöney, dizey, değişken ve sabit girdilerin, NRYJ ve NRYJ tarafından çağrılacak diğer alt-yordamlar tarafından ortak paylaşımının ana-yordam düzeyinde betimlenmesi daha uygundur. Bu betimleme ana-yordam düzeyinde yapıldıktan sonra, tüm ön girdi niteliğindeki değerlerin, Girdi adlı bir alt-yordam içinde atanmış olduğu varsayılarak söz konusu alt-yordamların algoritmaları açıklanacaktır.

NRYJ alt-yordamı, üst-yordam tarafından her

$$f_i(z_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2}, \qquad i = 1(1)p \qquad [1]$$

then the random vector $\mathbf{X} = (X_1, \ldots\ldots, X_p)'$ obtained by the transformation,

$$\mathbf{X} = \mathbf{LZ} + \boldsymbol{\mu} \qquad [2]$$

of which the density function is

$$\varphi(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \qquad [3]$$

where $\boldsymbol{\mu}=\langle \mathbf{X} \rangle$ is the p-dimensional column mean vector , $\boldsymbol{\Sigma}=\langle (\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})' \rangle$ is the positive definite symmetric pxp covariance matrix of which the i^{th} row and the j^{th} column element being σ_{ij}, and **L** is the unique Cholesky lower triangular pxp matrix which satisfies the relation

$$\boldsymbol{\Sigma} = \mathbf{LL}'. \qquad [4]$$

Proof: Let the density functions of the independent random p variables z_i be denoted by [1]. Then the density function of vector $\mathbf{z}=(z_1, \ldots\ldots, z_n)$ is ,

$$g(\mathbf{z}) = \prod_{i=1}^{n} f_i(z_i) = \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\sum_{i=1}^{n} z_i^2}. \qquad [5]$$

If the transformation technique of the variables (17) is applied by using the transformation $\mathbf{Z}=\mathbf{L}^{-1}(\mathbf{X}-\boldsymbol{\mu})$, defined from [2], it is seen that

$$\begin{aligned} h(\mathbf{x}) &= g\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\left|\frac{\partial \mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})}{\partial \mathbf{x}}\right| \\ &= \frac{1}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)'\left(\mathbf{L}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)} \left|\mathbf{L}^{-1}\right| \\ &= \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \\ &= \varphi(\mathbf{x}). \end{aligned}$$

3. ALGORITHM of the NRYJ

As a corollary of the above theorem, the proposed algorithms of NRYJ and the other subroutines required for NRYJ for random sampling from the distribution $\mathbf{X} \sim \mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ were designed so as to be used in a Monte Carlo Experiment main routine. From the point of efficiency, it is more appropriate to define at the modul level the commonly shared input vectors, matrices, variables and constants like $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, and sample size n to be called by NRYJ and by the subroutines called by NRYJ. After performing this definition at the main routine level, the related algorithms will be explained assuming all of the initial values have been inputted in a sub-routine, called Girdi.

In each call by an upper level routine, the NRYJ gets $\boldsymbol{\mu}$, **L**, and **z** as inputs and returns the vector $\mathbf{x} = \mathbf{Lz} + \boldsymbol{\mu}$ via

çağrılışında; μ, **L**, ve **z** girdilerini alıp, **X~N(μ,Σ)** dağılımından rasgele **x=Lz+μ** örnek birimlerini bulur. NRYJ alt-yordamının temel girdisi niteliğindeki,

- Kovaryans dizeyi Σ'dan ayrıştırılarak alt üçgensel **L** dizeyinin,
- **R** p-boyutlu standart tekdüze rassal bir yöney ve

$$\mathbf{F(Z \le z)} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_p} \int_{-\infty}^{z_{p-1}} \cdots\cdots \int_{-\infty}^{z_1} e^{-\frac{1}{2}\mathbf{z'z}} d\mathbf{z}$$

olmak üzere, $\mathbf{z} = \mathbf{F}^{-1}(\mathbf{r})$ ters dönüşümü ile **z** yöneyinin , ve

- **z** yöneyini verecek algoritmanın temel girdisi olarak da p-boyutlu standart tekdüze rassal **r** yöneyinin,farklı alt-yordamlarla bulunması uygundur. Bu üç yordamın her biri için temel alınabilecek çeşitli algoritmalar arasından, güvenilirlik ve hız açısından yeterli bulunan Cholesky(18), Box-Muller(19), ve Rand(20) algoritmaları seçildi ve küçük değişikliklerle , sırasıyla CHOLESKY, SNRYJ(Standart Normal Rassal Yöney Jenaratörü), ve STRYJ(Standart Tekdüze Rassal Yöney Jenaratörü) adı verilen alt-yordamlar oluşturuldu.

Daha önce de belirtildiği gibi, CHOLESKY ve NRYJ tarafından çağrılacak STRYJ, ve SNRYJ alt-yordamlarının karşılıklı girdi ve çıktılarının ortak paylaşımının ana-yordam düzeyinde betimlendiği ve tüm ön girdi niteliğindeki değerlerin, örneğin aşağıdaki gibi bir ana-yordam tarafından çağrılan Girdi adlı bir alt-yordam içinde atandığı ve yine ana-yordam düzeyinde çağrılan CHOLESKY alt-yordamı ile **L** dizeyinin Σ dizeyinden ayrıştırıldığı varsayılarak, söz konusu alt-yordamların açıklaması ve algoritmaları verilecektir.

Ana-yordam: MonteCarloDeney
Adım:01 Ortak paylaşımlı yöney,dizey,değişken ve sabitleri betimle.
Adım:02 $\mu = [\mu_i]_{px1}$, $\Sigma = [\sigma_{ij}]_{pxp}$, n ,
Tohum ← Alt-yordam : Girdi
Adım:03 $k = \lceil p/2 \rceil$ **:L** ← Alt-yordam : CHOLESKY
•
•
•

3.1 Alt-yordam: NRYJ

NRYJ alt-yordamının algoritması, bir üst-düzey yordam tarafından L ve μ girdileri verilerek çağrıldığında, STRYJ ve SNRYJ alt-yordamlarından r ve z yöneylerini alıp, [3] dönüşümüyle, verilen p-boyutlu normal bir dağılımdan, x örnek yöneyini elde eder.

Alt-yordam: NRYJ
Adım 01: **r** ← Alt-yordam : STRYJ
Adım 02: **z** ← Alt-yordam : SNRYJ
Adım 03: **x** ← **Lz+μ**

3.2. Alt-yordam: CHOLESKY

Kovaryans dizeyi Σ'yı ana-yordam düzeyinde girdi olarak alan, CHOLESKY alt-yordamının görevi, [3] deki tanıma uygun olarak Cholesky alt üçgen dizeyi L'yi Σ dizeyinden ayrıştırmaktır. Elemanları NRYJ alt-yordamının temel girdilerinden biri olan **L** dizeyi,

random sampling from the distribution **X~N(μ, Σ).**

It's more reasonable to obtain

- the lower triangular matrix **L** by decomposing covariance matrix Σ,
- the standard normal random vector **z**, by the inverse transformation $\mathbf{z} = \mathbf{F}^{-1}(\mathbf{r})$, where the **r** is a p-dimensional standard uniform vector and

$$\mathbf{F(Z \le z)} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_p} \int_{-\infty}^{z_{p-1}} \cdots\cdots \int_{-\infty}^{z_1} e^{-\frac{1}{2}\mathbf{z'z}} d\mathbf{z} \text{ , and}$$

- the p-dimensional standard uniform vector **r**, which is the input to the generating algorithm of vector **z**, in distinct sub-routines. Among the alternative algorithms available for each of the three procedures just described, Cholesky(18), Box and Muller(19), and the Rand(20) algorithms which are found sufficient from the point of reliability and speed were selected and adapted to the present context with some changes producing the sub-routines named in turn, CHOLESKY(Cholesky Lower Triangular Matrix), SNRYJ(Standard Normal Random Vector Generator), and STRYJ(Standard Uniform Random Vector Generator).

As it is stated before, the algorithms of these sub-routines will be presented and be explained below assuming the common sharing of the reciprocal inputs and outputs of the subroutines STRYJ and SNRYJ which are to be called by CHOLESKY and NRYJ, were defined at the module level, and the initial values were inputed in a sub-routine named Girdi, and the lower triangular matrix **L** is decomposed from matrix Σ in the sub-routine CHOLESKY which is to be called at the main-pocedure level.

The Main: MonteCarloExperiment
Step:01 Define the constants,variables,vector and matrix inputs and outputs to be shared .
Step:02 $\mu = [\mu_i]_{px1}$, $\Sigma = [\sigma_{ij}]_{pxp}$, n ,
Tohum ← Sub-routine : Girdi
Step:03 $k = \lceil p/2 \rceil$ **:L** ← Sub-routine: CHOLESKY
•
•
•

3.1. The Sub-routine: NRYJ

When it is called by an upper level procedure with the given L and μ inputs, it obtains r and z vectors from the subroutines STRYJ and SNRYJ and returns the sample x vector from the given p-dimensional multivariate normal distribution according to the transformation [2].

Algorithm of NRYJ
Step 01: **r** ← Sub-routine : STRYJ
Step 02: **z** ← Sub-routine : SNRYJ
Step 03: **x** ← **Lz+μ**

3.2. The Sub-routine: CHOLESKY

Taking the covariance matrix Σ as an input from the main algorithm, the task of the CHOLESKY algorithm is to decompose Σ and to get Cholesky lower triangular matrix **L** matrix defined in [4].

The elements of **L** which are the basic inputs of the

(18),(20), ve (21) de çeşitli biçimleri verilen aşağıdaki Cholesky denklemlerinden elde edilebilir.

$$\ell_{ij} = \begin{cases} 0, & i < j, \\ \dfrac{\sigma_{1}}{\sqrt{\sigma_{11}}}, & i = 1 \wedge 1 \le i \le p, \\ \sqrt{\sigma_{ii} - \sum_{k=1}^{i-1} \ell_{ik}^2}, & i = j \wedge 1 < i \le p, \\ \dfrac{\sigma_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}}{\ell_{jj}}, & 1 \wedge j < i \le p. \end{cases} \quad [6]$$

CHOLESKY alt-yordamının algoritması, doğrudan [6]'da verilen denklemlere dayalı olarak oluşturuldu.

Alt-yordam: CHOLESKY

Adım 01: i ← 1 : a ← $\sqrt{\sigma_{11}}$,

Adım 02: ℓ_{i1} ← σ_{i1} / a : i ← i+1,

Adım 03: i ≤ p ise, Adım:01 'ye dön;

Adım 04: $\ell_{22} \leftarrow \sqrt{\sigma_{22} - \ell_{21}^2}$: i ← 3,

Adım 05: j ← 2,

Adım 06: $\ell_{ij} \leftarrow \dfrac{\sigma_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}}{\ell_{jj}}$: j ← j + 1,

Adım 07: j < i ise, Adım:06'ya dön ;

Adım 08: $\ell_{ii} \leftarrow \sqrt{\sigma_{ii} - \sum_{j=1}^{i-1} \ell_{ii}^2}$: i ← i + 1,

Adım 09: i ≤ p ise, Adım:05'e dön.

3.3. Alt-yordam: STRYJ

Her programlama dilinin hazır rassal sayı üreten fonksiyonu olmakla beraber, her hangi bir Monte Carlo deneyde kullanılacak rassal sayı üretecinin sahip olduğu periyodun deneyci tarafından bilinmesi gerekir. Bu nedenle, Monte Carlo deneyci periyodunu bildiği kendi üretecini yeğliyebilir. Örneğin (20)'deki gibi modülü 2^{31}-1 ve periyodu yaklaşık $2x10^9$ olan çarpımsal uyumlu sözde-rassal bir sayı üreteci, ortalama bir Monte Carlo deney için oldukça yeterlidir. A çarpanlı ve M=2^{31}-1 modüllü,

$$r_i = A\, r_{i-1} \ \mathbf{mod}\ M \quad [7]$$

üreteci bir alt-yordam biçiminde programlanırken, [7] tanımının doğrudan bir algoritma olarak kullanılması durumunda, kimi Ar_{i-1} değerlerinin hesaplanmasındaki tamsayı aritmetik işlem sonuçları, kelime uzunluğu 32-bit olan bilgisayarlarda, 2^{31}-1 değerini aşacağından taşma hatası verir. Aritmetik işlemleri parçalayarak taşma hatasını önleyen aşağıdaki algoritma, [7]'deki algoritmaya denktir (20):

subroutine NRYJ can be computed from the following Cholesky equations, of which the several versions have been given in (18),(20), and (21).

$$\ell_{ij} = \begin{cases} 0, & i < j, \\ \dfrac{\sigma_{1}}{\sqrt{\sigma_{11}}}, & i = 1 \wedge 1 \le i \le p, \\ \sqrt{\sigma_{ii} - \sum_{k=1}^{i-1} \ell_{ik}^2}, & i = j \wedge 1 < i \le p, \\ \dfrac{\sigma_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}}{\ell_{jj}}, & 1 \wedge j < i \le p. \end{cases} \quad [6]$$

The algorithm of the subroutine CHOLESKY is based directly on [6] :

Sub-routine: CHOLESKY

Step 01: i ← 1 : a ← $\sqrt{\sigma_{11}}$,

Step 02: ℓ_{i1} ← σ_{i1} / a : i ← i+1,

Step 03: If i ≤ p , then return to Step:01;

Step 04: $\ell_{22} \leftarrow \sqrt{\sigma_{22} - \ell_{21}^2}$: i ← 3,

Step 05: j ← 2,

Step 06: $\ell_{ij} \leftarrow \dfrac{\sigma_{ij} - \sum_{k=1}^{j-1} \ell_{ik}\ell_{jk}}{\ell_{jj}}$: j ← j + 1,

Step 07: If j < i then return to Step:06;

Step 08: $\ell_{ii} \leftarrow \sqrt{\sigma_{ii} - \sum_{j=1}^{i-1} \ell_{ii}^2}$: i ← i + 1,

Step 09: If i ≤ p then return to Step:05.

3.3. The Sub-routine: STRYJ

Although each programming language has its own random number generating function, the period of random number generator that will be used in a Monte Carlo work is ought to be known by the experimenter. Therefore, the Monte Carlo experimenter may prefer his own generator with the a prior known period. A multiplicative congruential pseudorandom number generator with modulus 2^{31}-1, such as the one given in (20) has a period of $2x10^9$ approximately which is highly sufficient for a moderate Monte Carlo experiment. In programming the random number generator

$$r_i = A\, r_{i-1} \ \mathbf{mod}\ M \quad [7]$$

with the multiplier A and modulus M=2^{31}-1 , if a direct algorithm of [7] is adopted it may result with an overflow error for some integer arithmetic in computing some of A r_{i-1} since the result may be larger than 2^{31}-1 in the computers whose word size is 32-bit. The following algorithm is an equivalent form of [7],and eliminates the overflow error through partitioning arithmetic operations (20).

$$r_i = \left\{ \begin{array}{l} (A(\mathrm{mod}\,2^{16})r_{i-1}(\mathrm{mod}\,2^{31}) \\ +(A - A(\mathrm{mod}\,2^{16}))r_{i-1} \\ +\left\lfloor \dfrac{A(\mathrm{mod}\,2^{16})r_{i-1}}{2^{31}} \right\rfloor \\ +\left\lfloor \dfrac{(A - A(\mathrm{mod}\,2^{16}))r_{i-1}}{2^{31}} \right\rfloor \end{array} \right\} (\mathrm{mod}\,2^{31}-1) \quad [8]$$

Her çağrılışta k = ⌈p/2⌉2 boyutlu standart tekdüze rassal bir yöney üretecek olan STRYJ alt-yordamının algoritması, çarpan A=950706376 ve modül M=2^{31}-1 olmak üzere [8]'deki tanıma göre oluşturuldu:

Alt-yordam: STRYJ
Adım 01: A←950706376:A_1←41160:A_2← 950665216,
Adım 02: {i← 1(1)k; r_i← 0} : i← 0,
Adım 03: r← Tohum,
Adım 04: i← i + 1 : r← ⌊r⌋ : r_1← rA_1 : r_2← rA_2,
Adım 05: B_1 ←⌊r_1 / 2^{31}⌋ : B_2 ←⌊r_2 / 2^{31}⌋ ,
Adım 06: r_1 ← $r_1 - 2^{31}B_1$: r_2 ← $r_2 - 2^{31}B_2$,
Adım 07: r← $r_1 + r_2 + B_1 + B_2$: B ←⌊ r /(2^{31} −1)⌋ ,
Adım 08: r← r − B(2^{31} −1) : Tohum ← r,
Adım 09: r_i← r / (2^{31} −1) ,
Adım 10: i < k ise, Adım:03'e dön.

3.4 Alt-yordam: SNRYJ

z yöneyinin k=⌈p/2⌉2 tane birbirinden bağımsız standart normal değişken değerinden oluşan elemanlarını üretmek için, Box ve Muller(19)'in iki-değişkenli kutupsal üretecinin yinelemeli bir biçimi kullanılabilir. STRYJ alt-yordamının ürettiği r yöneyinin ardışık her iki elemanını girdi olarak alacak olan Box-Muller algoritması bir birinden bağımsız iki standart normal rassal değişken değeri üreteceğinden, k çift bir sayı olmak üzere,

$$z_i = \begin{cases} Sin(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}, & i = 1(2)k-1 \\ Cos(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}, & i = 2(2)k \end{cases} \quad [9]$$

biçimindeki bir düzenleme ile z yöneyi elde edilebilir. p-boyutunun tek sayı olması durumunda, STRYJ alt-yordamı tarafından üretilen r yöneyinin son elemanı r_k , SNRYJ alt-yordamı tarafından kullanılsa da, r_k ve z_k değerlerinin Monte Carlo deneyde yer alamayacağı açıktır. Böyle bir durumda, Monte Carlo deneyin kullanacağı standart tekdüze rassal değişken değerleri dizisi, STRYJ alt-yordamının ürettiği dizinin her k adımda bir ürettiği değerleri içermeyecektir. Ancak bu, Monte Carlo deneyin rassallığını bozmaz.

Bu açıklamaların ışığında, SNRYJ alt-yordamı [9]'daki tanıma dayalı aşağıdaki algoritmaya göre yazıldı:

Alt-yordam: SNRYJ
Adım 01: {i← 1(1)k; z_i← 0} : i← 0,
Adım 02: i = i + 1 ,
Adım 03: $z_i = Sin(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}$,

$$r_i = \left\{ \begin{array}{l} (A(\mathrm{mod}\,2^{16})r_{i-1}(\mathrm{mod}\,2^{31}) \\ +(A - A(\mathrm{mod}\,2^{16}))r_{i-1} \\ +\left\lfloor \dfrac{A(\mathrm{mod}\,2^{16})r_{i-1}}{2^{31}} \right\rfloor \\ +\left\lfloor \dfrac{(A - A(\mathrm{mod}\,2^{16}))r_{i-1}}{2^{31}} \right\rfloor \end{array} \right\} (\mathrm{mod}\,2^{31}-1) \quad [8]$$

The sub-routine STRYJ which will generate a standard uniform random vector of dimension k=⌈p/2⌉2 on each call, was constructed according to equation [8], with the multiplier A=950706376 and modulus M=2^{31}-1

Sub-routine: STRYJ
Step 01: A←950706376:A_1←41160:A_2← 950665216,
Step 02: {i← 1(1)k; r_i← 0} : i← 0,
Step 03: r← Tohum,
Step 04: i← i + 1 : r← ⌊r⌋ : r_1← rA_1 : r_2← rA_2,
Step 05: B_1 ←⌊r_1 / 2^{31}⌋ : B_2 ←⌊r_2 / 2^{31}⌋ ,
Step 06: r_1 ← $r_1 - 2^{31}B_1$: r_2 ← $r_2 - 2^{31}B_2$,
Step 07: r← $r_1 + r_2 + B_1 + B_2$: B ←⌊ r /(2^{31} −1)⌋ ,
Step 08: r← r − B(2^{31} −1) : Tohum ← r,
Step 09: r_i← r / (2^{31} −1) ,
Step 10: If i < k then , return to Step:03.

3.4 The Sub-routine: SNRYJ

To generate the independently distributed standart normal k=⌈p/2⌉2 elements of vector **z**, a generating algorithm such as that of Box and Muller's(19) bivariate standard normal polar generator can be utilized recursively. For each pair of independent standart uniform deviates from STRYJ, the Box-Muller algorithm returns a pair of independent standard normal deviates; thus the following arrangement of Box-Muller transformation produces the elements of **z**, assuming k is an even number

$$z_i = \begin{cases} Sin(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}, & i = 1(2)k-1 \\ Cos(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}, & i = 2(2)k \end{cases} \quad [9]$$

In case of p is an odd number, even though in each call of STRYJ the last standart uniform deviate r_k of vector **r** is used by SNRYJ, it is clear that , r_k and z_k values are not used in the Monte Carlo experiment. In such a case, the sequence of the uniform deviates to be used by the Monte Carlo experiment will not cover the values produced by sub-routine STRYJ at each of the k^{th} step. However it does not violate the randomness of the Monte Carlo experiment.

Under this setting, the algorithm of SNRYJ based on [9] is:

Sub-routine: SNRYJ
Step 01: {i← 1(1)k; z_i← 0} : i← 0,
Step 02: i = i + 1 ,
Step 03: $z_i = Sin(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}$,

Adım 04: $z_{i+1} = Cos(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}$,
Adım 05: i < k-1 ise, Adım:02'ye dön .

4. NRYJ'nin MONTE CARLO SINAMASI

Önerilen NRYJ üretecinin sınanmasında (22)'deki yaklaşım kullanılabilir. Ancak bu tür üreteçlerin güvenirliğini ortaya koymanın en kısa ve etkin yolu,(23)'teki gibi tüm parametrelerin kuramsal değerlerinin önceden bilindiği bir ortamda, parametrelerin Monte Carlo tahminlerini elde etmeye yönelik sanal bir deneyle sınamaktır. Böylece, parametrelerin gerçek değerleri ile tek bir sanal deneyin Monte Carlo tahminleri arasıdaki fark güvenirliğin doğrudan bir ölçütü olarak alınabilir.

NRYJ üretecini (ve aynı zamanda diğer alt-yordamları da) sınamak üzere tasarlanan Monte Carlo sanal deneyin benzetim-modelindeki parametre değerleri, (23)'den aynen alınmıştır. Buna göre verilecek örnekte, 10-boyutlu standart normal rassal bir **X** değişkeninin, kovaryans dizeyi,

$$\Sigma = \begin{bmatrix} 1 & -0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.8 & 1 \end{bmatrix} \quad [10]$$

beklenen değer yöneyi,

$$\mu = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)' \quad [11]$$

üst-sınır değerleri yöneyi,

$$\mathbf{h} = (1.7\ 0.8\ 5.1\ 3.2\ 2.4\ 1.8\ 2.7\ 1.5\ 1.2\ 2.6)' \quad [12]$$

ve

$$\Phi(\mathbf{x} \le \mathbf{h}) = \frac{1}{\sqrt{(2\pi)^{10}|\Sigma|}} \int_{-\infty}^{\mathbf{h}} e^{-\frac{1}{2}\mathbf{x}\Sigma^{-1}\mathbf{x}} d\mathbf{x} = 0.58300606 \quad [13]$$

olduğu biliniyor.

Beklenen değer yöneyi μ, kovaryans dizeyi Σ, ve $\Phi(\mathbf{X} \le \mathbf{h})$ olasılığının Monte Carlo tahminlerini elde etmek için tasarlanan sanal deneyin i. denemesinde, NRYJ alt-yordamı ile, $\mathbf{X} \sim N(\mu, \Sigma)$ dağılımından rasgele seçilen 10-boyutlu $\{\mathbf{x}^{(i)}_j : j=1(1)25\}$ birimlerinin oluşturduğu s=25 çaplı örneklerin ortalama yöneyi ve kovaryans dizeyi hesaplanarak; "ζ-yakınsak denemeli ve 2ε-luk tahmin hata aralığında yakınsamalı" Monte Carlo tahmin kıstası sağlanıncaya kadar deneye devam edilecektir. Bu amaçla düzenlenen Monte Carlo deneyin sanal gözlemlerine uygulanacak işlemlerin temel ögeleri aşağıda tanımlanmıştır. [15],[18],ve [21]'de simgesel olarak tanımlanan Monte Carlo tahminler ile biraz önce belirtilen Monte Carlo tahmin kıstası ayrıntıları ile (24)'de açıklandığından, bu konuya burada daha fazla

Step 04: $z_{i+1} = Cos(2.\pi.r_i)\sqrt{-2.ln(r_{i+1})}$,
Step 05: If i < k-1 then return to Step:02.

4. MONTE CARLO TEST of the NRYJ

In testing the reliability of the proposed generator NRYJ, the approach of (22) might be followed. However, the most efficient shortest way of measuring the reliability of such generators is to test by a Monte Carlo experiment to estimate the parameters whose exact theoretical values are known a priori. Hence, the discrepancy between the Monte Carlo estimates obtained from a single virtual experiment and the exact values might be used as a direct measure of reliability of the algorithm.

The parameter values in the simulation model of the Monte Carlo virtual experiment designed for testing the NRYJ generator (and at the same time the other sub-routines) were taken from (23). So, in the example that is to be presented, it is known that covariance matrix of the 10-dimensional standard normal random variable **X** is

$$\Sigma = \begin{bmatrix} 1 & -0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.8 & 1 \end{bmatrix} \quad [10]$$

the expected mean vector is

$$\mu = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)' \quad [11]$$

the vector of upper bound values is

$$\mathbf{h} = (1.7\ 0.8\ 5.1\ 3.2\ 2.4\ 1.8\ 2.7\ 1.5\ 1.2\ 2.6)' \quad [12]$$

and

$$\Phi(\mathbf{x} \le \mathbf{h}) = \frac{1}{\sqrt{(2\pi)^{10}|\Sigma|}} \int_{-\infty}^{\mathbf{h}} e^{-\frac{1}{2}\mathbf{x}\Sigma^{-1}\mathbf{x}} d\mathbf{x} = 0.58300606 \quad [13]$$

Computing the mean vector and covariance matrix of a sample composed by 10-dimensional units $\{\mathbf{x}^{(i)}_j : j=1(1)25\}$ of a sample size of s=25 which are randomly sampled by the subroutine NRYJ from the distribution $\mathbf{X} \sim N(\mu, \Sigma)$, at the i^{th} trial of the virtual experiment designed to obtain the Monte Carlo estimates of the expected mean vector μ, covariance matrix Σ and the probability of $\Phi(\mathbf{X} \le \mathbf{h})$, the experiment will continue on until the Monte Carlo convergence criterion of the "ζ-step convergent trials within 2ε-estimation error interval" is satisfied. The fundamental steps of the procedures to be applied to the virtual observations of the Monte Carlo experiment designed for this target have been defined below. Since the Monte Carlo estimates and the Monte Carlo estimation criterion defined symbolically in [15],[18], and [21] have

değinmeyeceğiz.

$$\bar{x}_\ell^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x_\ell^{(i)}, \qquad \ell = 1(1)10 \qquad [14]$$

$$\lim_{n\to N}\left(\bar{x}_l^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{x}_l^{(n)}\right\}-\min\left\{\bar{x}_l^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \hat{\mu}_l^{MC} \pm \varepsilon, \qquad [15]$$

$$l = 1(1)10.$$

$$\hat{\sigma}_{\ell,m}^{(i)} = \frac{\sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_\ell^{(j)} - \bar{x}_\ell^{(n)})(x_m^{(j)} - \bar{x}_m^{(n)})}{\sqrt{\sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_\ell^{(j)} - \bar{x}_\ell^{(n)})^2 \sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_m^{(j)} - \bar{x}_m^{(n)})^2}}, \quad i = 1(1)n; l = 1(1)10; m = 1(1)10, \qquad [16]$$

$$\bar{\hat{\sigma}}_{\ell,m}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} \hat{\sigma}_{\ell,m}^{(i)}, \qquad [17]$$

$$\lim_{n\to N}\left(\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right\}-\min\left\{\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \hat{\sigma}_\ell^{MC} \pm \varepsilon, \qquad [18]$$

$$\ell, m = 1(1)10.$$

$$w_i^{(\ell)} = \begin{cases} 1, & x_i \le h \\ 0, & x_i > h \end{cases}, \qquad i = 1(1)n \qquad [19]$$

$$\bar{w}_\ell^{(n)} = \frac{1}{n}\sum_{i=1}^{n} w_i^{(\ell)}, \qquad [20]$$

$$\lim_{n\to N}\left(\bar{w}_\ell^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{w}_\ell^{(n)}\right\}-\min\left\{\bar{w}_\ell^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \Phi_\ell^{MC} \pm \varepsilon, \qquad [21]$$

$$\ell = 1(1)10.$$

Tasarlanan Monte Carlo deney için QuickBasic dilinde yazılan program; Tohum=0.005336789 , s=25 , ve Monte Carlo tahmin kıstası:{ζ=250, 2ε=0.005}alınarak çalıştırıldı. N=1189[ci] deneme sonunda, Monte Carlo birikimli örnek istatistikleri μ^{MC} ve Σ^{MC} ile, yakınsanan $\Phi(X \le h)$ olasılığının Monte Carlo tahmini **EK.A**'da söz konusu programın ekran çıktısı biçiminde verilmiştir.

5. SONUÇ

Görüldüğü üzere, gerçek kuramsal $\Phi(X \le h)$=0.58300606 değeri ile Monte Carlo tahmin $\hat{\Phi}^{MC}(X \le h)$ =0.581688 değeri arasındaki fark ε=0.0025'den küçüktür. Son ζ=250 denemenin, 2ε=0.005'lik bir yakınsama aralığında gerçekleşmiş olması ise, Büyük Sayılar Yasası ışığında,

$$Ol\{0.579188 \le \Phi(X \le h) \le 0.584588 \mid n \ge 1200, \varepsilon = 0.0025, \zeta = 250\} \approx 1$$

olarak yorumlanabileceğinden NRYJ'nün güvenilir bir üreteç olduğu söylenebilir.

been explained in detail in (24), this subject will not be dealt with more.

$$\bar{x}_\ell^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x_\ell^{(i)}, \qquad \ell = 1(1)10 \qquad [14]$$

$$\lim_{n\to N}\left(\bar{x}_l^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{x}_l^{(n)}\right\}-\min\left\{\bar{x}_l^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \hat{\mu}_l^{MC} \pm \varepsilon, \qquad [15]$$

$$l = 1(1)10.$$

$$\hat{\sigma}_{\ell,m}^{(i)} = \frac{\sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_\ell^{(j)} - \bar{x}_\ell^{(n)})(x_m^{(j)} - \bar{x}_m^{(n)})}{\sqrt{\sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_\ell^{(j)} - \bar{x}_\ell^{(n)})^2 \sum_{j=\left\lfloor \frac{i}{S}\right\rfloor S+1}^{Sj} (x_m^{(j)} - \bar{x}_m^{(n)})^2}}, \quad i = 1(1)n; l = 1(1)10; m = 1(1)10, \qquad [16]$$

$$\bar{\hat{\sigma}}_{\ell,m}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} \hat{\sigma}_{\ell,m}^{(i)}, \qquad [17]$$

$$\lim_{n\to N}\left(\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right\}-\min\left\{\bar{\hat{\sigma}}_{\ell,m}^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \hat{\sigma}_\ell^{MC} \pm \varepsilon, \qquad [18]$$

$$\ell, m = 1(1)10.$$

$$w_i^{(\ell)} = \begin{cases} 1, & x_i \le h \\ 0, & x_i > h \end{cases}, \qquad i = 1(1)n \qquad [19]$$

$$\bar{w}_\ell^{(n)} = \frac{1}{n}\sum_{i=1}^{n} w_i^{(\ell)}, \qquad [20]$$

$$\lim_{n\to N}\left(\bar{w}_\ell^{(n)}\right) \xrightarrow{\left\{\max\left\{\bar{w}_\ell^{(n)}\right\}-\min\left\{\bar{w}_\ell^{(n)}\right\}\le 2\varepsilon : n=(N-\zeta(1)N\right\}} \Phi_\ell^{MC} \pm \varepsilon, \qquad [21]$$

$$\ell = 1(1)10.$$

The program coded in QuickBasic language for the designed Monte Carlo experiment was run for Tohum=0.005336789 , s=25 , and Monte Carlo estimation criterion:{ζ=250, 2ε=0.005}. The converged Monte Carlo estimate of the probability of $\Phi(X \le h)$ and the Monte Carlo cumulative sample statistics μ^{MC} and Σ^{MC} have been presented in **Appendix.A** as the output of the program at the end of N=1189[th] trial.

5. CONCLUSION

As it is observed the discrepancy between the actual $\Phi(x \le h)$=0.58300606 and its Monte Carlo estimate $\hat{\Phi}^{MC}(X \le h)$ =0.581688 is less than ε=0.0025. Since the fact that the last ζ=250 trials have resulted within a convergence band with a width of 2ε=0.005 can be interpreted as

$$\Pr\{0.579188 \le \Phi(X \le h) \le 0.584588 \mid n \ge 1200, \varepsilon = 0.0025, \zeta = 250\} \approx 1$$

under the light of Law of Large Numbers, it can be asserted that the NRYJ is a reliable generator.

REFERENCES/ KAYNAKLAR

1. Burn, D.A., " Advanced Simulation and Statistics Package-IBM Professional Version", ***The American Statistician***,41(4): 324-327(1987).

2. Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W., " Linpack Users Guide", ***Society of Industrial and Applied Mathematics***, Philadelphia (1979).

3. Aguinis, H., "A QuickBasic Program for Generating Correlated Multivariate Random Normal Scores", ***Educational and Psychological Measurement***, 54(3):687-689 (1994).

4. Alliger, G.M., "Generating Correlated Bivariate Random Normal Standard Scores in QuickBasic," ***Educational and Psychological Measurement***, 52(1):107-108(1992).

5. Cheng, R.C.H., "Generation of Multivariate Normal Samples with Given Sample Mean and Covariance Matrix", ***Journal of Statistical Computation and Simulation***, 21: 39 - 49(1985).

6. Fernandez, J.F-Criado,C., "Algorithm for Normal Random Numbers", ***Physical Review E***, 60(3): 3361-3365(1999).

7. Ghosh, A., Kulatilake, P.H.S.W., " A Fortran Program for Generation of Multivariate Normally Distributed Random-Variables", ***Computers and Geosciences***, 13(3): 221-233 (1987).

8. Johnson, M. E., Wang, C.R., John S., "Generation of Continuous Multivariate Distributions for Statistical Applications", ***American Journal of Mathematical and Management Sciences***, 4:225-248 (1984).

9. Marsaglia, G., MacLaren, M.D., Bray, T.A., "A Fast Procedure for Generating Normal Random Variables", ***Communications of the ACM***, 7(1): 4-10 (1964).

10. Marsaglia, G., Zaman, A., Marsaglia, J.C.W., "Rapid Evaluation of the Inverse of the Normal-Distribution Function", ***Statistics and Probabilty Letters***, 19(4) :259-266 (1994).

11. Meyer, D.L., "Methods of Generating Random Normal Numbers", ***Educational and Psychological Measurement***, 29(1): 193- (1969).

12. Mickey, M.R., Chen, E.H., "How Normal does a Random Normal Generator Need to Be", ***Biometrics***, 31(1) : 257 (1975).

13. Odell, P.L., "On Generating Normal Random Vectors", ***The American Mathematical Monthly***, 72(4): 454 (1965).

14. Scheuer, E. M., Stoller, D.S., "On the Generation of Normal Random Vectors", ***Technometrics***, 4: 278-281 (1962).

15. Woodward, J.A., Overall,J.E., "Computer Generation of Random Normal Deviates for Monte Carlo Work", ***Perceptual and Motor Skills***, 39(1): 31-37 (1974).

16. Edwards, L. K., "On Comparative Accuracy of Multivariate Nonnormal Random Number Generators", Science and Statistics: Proceedings of the 20th Symposium on the Interface, 618-623 (1988).

17. Anderson,T.W., "An Introduction to Multivariate Statistical Analysis", ***John Wiley and Sons Inc.***, New York,10-11 (1958).

18. Thisted, R. A., "Elements of Statistical Computing-Numerical Computation", ***Chapman and Hill***, New York, 81-84 (1988).

19. Box, G.E.P., Muller, M.E., "A note on the generation of random normal deviates", ***The Annals of Mathematical Statistics***, 29 : 610-611 (1958).

20. Fishman, G.S., "Monte Carlo-Concept, Algorithms, Applications", ***Springer-Verlag***, New York, 603-604 (1996).

21. Shoup, T.E., "Applied Numerical Methods for the Microcomputer", ***Prentice-Hall,Inc.***, Englewood Cliffs, New Jersey, 53-57 (1984).

22. Loh,W.Y., "Testing Multivariate Normality by Simulation", ***Journal of Statistical Computation and Simulation***, 26, 243-252 (1986).

23. Deák, I., "Random Number Generators and Simulation", Akademia Kiado, Budapest , 317 (1990).

24. Ata, M. Y., "Sanal Deneylerde Monte Carlo Tahminler için Deneysel Bir Yakınsama Kıstası", ***Teknik Rapor No:1***, İstatistik Bölümü, Fen-Edebiyat Fakültesi, Gazi Üniversitesi, Ankara (2002).

Received/ Geliş Tarihi: 04.04.2003 Accepted/Kabul Tarihi: 29.09.2003

APPENDICES/ EKLER

A.. *NRYJ Üretecinin Monte Carlo Sınaması İçin Düzenlenen Örnek Sanal Deneyin Sonucu*
(The Output of the Sample Virtiual Experiment Designed for the Monte Carlo Test of NRYJ Generator.)

```
Deneme No: 1186      Yakınsak Deneme Sayısı : 250      Sanal Deney Süresi :227"
(Trial No.)          (Convergent Trial Number)         (Duration of Virtual Experiment)

MC Birikimli Örnek Beklenen Değer Yöneyi (MC Cumulative Sample Mean Vector):
 -0.0008 -0.0020 -0.0076 -0.0100 -0.0004 –0.0027 0.0006 0.0105 0.0054 –0.0083

MC Birikimli Örnek Kovaryans Dizeyi (MC Cumulative Sample Covariance Matrix):
 0.9995 -0.5962 0.0017 0.0025 -0.0006 0.0024 0.0128 0.0003 -0.0043  0.0025
-0.5962 0.9974 0.0037 0.0005 -0.0043 -0.0034 -0.0126 -0.0038 0.0068 -0.0087
 0.0017 0.0037 1.0030 0.9046 -0.0070 0.0000 -0.0016 0.0021 0.0073  -0.0060
 0.0025 0.0005 0.9046 1.0068 -0.0037 0.0019 -0.0057 -0.0028 0.0078  -0.0044
-0.0006 -0.0043 -0.0070 -0.0037 1.0014 0.3963 -0.0176 -0.0024 0.0105 -0.0033
 0.0024 -0.0034 0.0000 0.0019  0.3963 0.9923 -0.0063 -0.0048 -0.0000 0.0042
 0.0128 -0.0126 -0.0016 -0.0057 -0.0176 -0.0063 0.9905 0.1948 0.0062  -0.0082
 0.0003 -0.0038 0.0021 -0.0028 -0.0024 -0.0048 0.1948 0.9938 0.0032  0.0041
-0.0043 0.0068 0.0073 0.0078 0.0105 -0.0000 0.0062 0.0032  1.0021 -0.7910
 0.0025 -0.0087 -0.0060 -0.0044 -0.0033 0.0042 -0.0082 0.0041  -0.7910 0.9884

Ol(x<=h)'nin MC Tahmini (MC Estimate of Pr(x<=h))= 0.581688± 0.0025
```

B. *NRYJ, CHOLESKY, STRYJ, SNRYJ Alt-Yordamları ve Bu Alt-Yordamları Kullanan Bir Ana-Yordam Örneğinin QuickBasic Dilinde Uyarlaması (The implementation of the sub-routines NRYJ, CHOLESKY, STRYJ, SNRYJ and a sample main procedure using these sub-routines in the QuickBasic Language.)*

Main Procedure-Ana Yordam

```
'BU ANA-YORDAM, 9-BOYUTLU BEKLENEN DEĞER YÖNEYİ ve
'KOVARYANS DİZEYİ VERİLEN 9-BOYUTLU NORMAL BİR DAĞILIMDAN
'RASGELE 3 ÖRNEK SEÇER ve EKRANA YAZAR. (This main procedure
'samples randomly from a 9-dimensional normal distribution with 'given
covariance 'matrix and expected mean vector.)
DECLARE SUB CHOLESKY (BDD(), PP)
DECLARE SUB Girdi ()
DECLARE SUB NRYJ (MU(), L(), K)
DECLARE SUB STRYJ(TOHUM#, PP))
DECLARE SUB SNRYJ(STRY#(), PP)
COMMON Boyut,ORTALAMA(),BDD(),L(),NRY#(), SNRY#(),STRY#()
CLS:CALL Girdi: CALL CHOLESKY(BDD(), 9)
FOR I = 1 TO 9:PRINT USING "########"; I;:NEXT I: PRINT
FOR I = 1 TO S:CALL NRYJ(ORTALAMA(), L(),9):PRINT USING "###"; I;
FOR J = 1 TO Boyut: PRINT USING "#####.###"; NRY#(J); NEXT J: PRINT:NEXT
I
END
DATA     9,    3,    .005336789#
DATA     0,    0,    0,    0,    0,    0,    0,    0,    0
DATA     1,-0.6,    0,    0,    0,    0,    0,    0,    0
DATA  -0.6,    1,    0,    0,    0,    0,    0,    0,    0
DATA     0,    0,    1,  0.9,    0,    0,    0,    0,    0
DATA     0,    0,  0.9,    1,    0,    0,    0,    0,    0
DATA     0,    0,    0,    0,    1,  0.4,    0,    0,    0
DATA     0,    0,    0,    0,  0.4,    1,    0,    0,    0
DATA     0,    0,    0,    0,    0,    0,    1,  0.2,    0
DATA     0,    0,    0,    0,    0,    0,  0.2,    1,    0
DATA     0,    0,    0,    0,    0,    0,    0,    0,    1
```

The Output of the Main Program: (Ana Programın Çıktısı)

```
       1     2     3     4     5     6     7     8     9

  1 -1.559  1.178 -1.375 -1.478  1.072 -0.278  1.508 -0.517 -0.351
  2 -1.495  1.286 -0.059 -0.048  1.658  0.061  0.369 -0.735 -1.587
  3 -0.450  0.102 -0.347 -0.124 -1.025 -0.330  0.107  0.117  0.847
```

Sub-routines Alt-Yordamlar

```
SUB Girdi
SHARED Boyut,ORTALAMA(),BDD(),L(),NRY#(),SNRY#(), STRY#(), S
READ Boyut, S ,STRY#(K)
IF INT(Boyut/2)*2<>Boyut THEN K=Boyut+1 ELSE K=Boyut:P=Boyut
DIM ORTALAMA(K), BDD(K, K), L(K, K), NRY#(K),STRY#(K), SNRY#(K)
FOR I = 1 TO P: READ ORTALAMA(I): NEXT I
FOR I = 1 TO P:FOR J=1 TO P:READ BDD(I,J):NEXT J:PRINT:NEXT I
END SUB

SUB CHOLESKY(BDD(), PP)
SHARED L()
FOR I = 1 TO PP: FOR J = 1 TO PP
IF I < J THEN
   L(I, J) = 0
ELSEIF J = 1 THEN L(I,J)=BDD(I,J)/SQR(BDD(J, J))
ELSEIF I = J THEN
FOR K=1 TO I-1:TOPLAM =TOPLAM + L(I, K)^ 2:NEXT K
     L(I, J) = SQR(BDD(I, J) - TOPLAM):TOPLAM = 0
     ELSEIF J < I THEN
FOR K = 1 TO J - 1
 TOPLAM =TOPLAM+(L(I,K)^2)*(L(J,K)^ 2)
NEXT K
 L(I,J)=(BDD(I,J)-TOPLAM)/L(J,J):TOPLAM= 0
END IF
NEXT J: NEXT I
END SUB

SUB NRYJ(MU(), L(), K)
SHARED NRY#(), SNRY#(), STRY#()
IF INT(K / 2) * 2 <> K THEN K = K + 1
CALL SNRYJ(STRY#(), K)
FOR I = 1 TO K:FOR J = 1 TO K
TOP = TOP + L(I, J) * SNRY#(J)
NRY#(I) = TOP + MU(I)
NEXT J:TOP = 0:NEXT I
END SUB

SUB STRYJ (TOHUM#, K)
SHARED STRY#()
PAYDA# = 2 ^ 31: MODUL# = 2 ^ 31 - 1 :Z# = TOHUM# * PAYDA#
a1# = 41160              '=950706376 Mod (2^16)
a2# = 950665216          '=950706376-41160
FOR I = 1 TO K
Z# = INT(Z#): Z1# = Z# * a1#: Z2# = Z# * a2#
KISIM1# = INT(Z1# / PAYDA#): KISIM2# = INT(Z2# / PAYDA#)
Z1# = Z1# - KISIM1# * PAYDA#: Z2# = Z2# - KISIM2# * PAYDA#
Z# = Z1# + Z2# + KISIM1# + KISIM2#: KISIM# = INT(Z# / MODUL#)
Z# = Z# - KISIM# * MODUL#: STRY#(I) = Z# / MODUL#: TOHUM# = STRY#(I)
NEXT I
END SUB

SUB SNRYJ(STRY#( ), K)
SUB SHARED SNRY#()
PI = 3.141592654#:CALL STRYJ(STRY#(K), K)
FOR I = 1 TO K STEP 2
SNRY#(I)=SQR(-2*LOG(STRY#(I)))*SIN(2*PI*STRY#(I + 1))
SNRY#(I+1)=SQR(-2*LOG(STRY#(I)))*COS(2*PI*STRY#(I + 1))
NEXT I
END SUB
```