British Journal of Mathematics & Computer Science

22(5): 1-8, 2017; Article no.BJMCS.33945
ISSN: 2231-0851

Cyclic-union Operation to Obtain Latin Squares

M. I. García-Planas[1*] and D. Roca-Borrego[1]

[1] *Departament de Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain.*

Authors' contributions:

This work was carried out in collaboration between both authors. Author MIGP designed the study, introduced Latin squares from a historical point of view and managed literature searches. Author DRB make the description of methods for constructing new Latin squares and expose the code. Both authors searched the literature and read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2017/33945
Editor(s):
(1) Huchang Liao, Business School, Sichuan University, P. R. China.
Reviewers:
(1) Raul Manuel Falcon Ganfornina, University of Seville, Spain.
(2) Krasimir Yordzhev, South-West University, Bulgaria.
(3) Vishnu Namboodiri, India.
(4) Anthony B. Evans, Wright State University, USA.
Complete Peer review History: http://www.sciencedomain.org/review-history/19542

Received: 4^{th} ***May 2017***
Accepted: 8^{th} ***June 2017***
Published: 15^{th} ***June 2017***

Short communication

Abstract

With the power that has taken the information technologies, one has developed the study and research about cryptography, and cryptanalysis, in which Latin squares are ideal candidates for being used in cryptographic systems because the Cayley tables of the finite groups are Latin squares. This fact has awakened a new interest in the study of Latin squares by applying them to the study of code theory and error correcting codes. They also play a significant role in the statistical theory of experimental design.
In this work, we develop an algorithm for the generation of Latin squares based on the cyclic-union operation defined for effect.

Keywords: Latin square; cyclic matrices; Generation of Latin squares.

2010 Mathematics Subject Classification: 68N15, 15A36

**Corresponding author: E-mail: maria.isabel.garcia@upc.edu*

1 Introduction

The Latin squares have been studied for centuries; their origin goes back at least to the surroundings of the year 1000 in which the Arab and Indian communities used them as amulets or talismans. Also in the thirteenth century, the philosopher Ramon Llull (1232-1315) introduces in his text Ars Demostrativa (1283) four Latin squares of order 4, using as symbols fire, air, water and earth, in an attempt to explain the world to Through combinatorial numbers [1]. However, the first formal definition was given by Leonhard Euler [2], who was interested in solving the problem of the 36 officers.

The Latin squares appear naturally in algebra, for example, when constructing the summation tables on the sets of classes of remainder modulo m.

In recent years, the rise of information technologies has led to the development of cryptography, a science that studies the creation of secret codes, and cryptanalysis aimed at deciphering those codes, in which Latin squares are ideal candidates for Be used in cryptographic systems because the Cayley tables of the finite groups are Latin squares. This fact has awakened a new interest in the study of Latin squares by applying them to the study of code theory and error correcting codes. They also play a significant role in the statistical theory of experimental design.

Following Ritter in [3] a Latin square can be seen as a stream cipher combiner, Koscielny in [4] developed some routines using the Maple 7 package to generate 256-order Latin squares. For use in cryptography in addition to the Latin square-building algorithm, an algorithm for extracting symbols from one Random shape must be done. When it is intended to create a Latin square generation algorithm, to obtain an encryption or decryption algorithm for a secure communication protocol, it must be such that the Latin square is generated quickly and from time to time. This construction should not represent an overload of time or resources (memory, hard disk, etc.) in the protocol. Generally, simple Latin square generation algorithms are of exponential order, so the problem requires a certain complexity to be efficient. Gallego in [5], proposes two implementations, one giving weight to clarity versus efficiency and the other giving weight to efficiency versus clarity, highlighting the difficulty in achieving these two properties at the same time.

The exact number of Latin squares of order $n > 11$ has not been known so far (for more information, see [6][7] and [8] for example). However, there have been different lower bounds for this number. Therefore it is still interesting to create algorithms that generate Latin squares.

The objectives of this work are on the one hand to compile the existing information in the literature on the Latin squares and, on the other is to outline some techniques for a specific computer tool for the generation of Latin squares to they can be used for graduate students learning.

In [9] some constructions of Latin squares are summarized, in particular the construction of Latin squares preserving some subsquare conditions. By simulating an ergodic Markov chain with uniform stationary distribution over the space of n-order Latin squares, Jacobson and Matthews in [10], have discussed some methods to generate Latin squares.

2 Preliminaries

A Latin square is an $n \times n$-order matrix L whose elements belong to a finite set A of cardinal n and each of them appears exactly once in each row and each column of L. The set A is usually considered as the set of the firsts n natural numbers and receives the Base set name of the square and n its order.

Example 2.1. *A 6-order Latin square can be:*

1	2	3	4	5	6
4	5	6	1	2	3
6	1	2	3	4	5
3	4	5	6	1	2
5	6	1	2	3	4
2	3	4	5	6	1

A special case of Latin squares is the so-called Latin reduced squares. That are those in which its first row and its first column the numbers from 1 to n taken as symbols, appear in the natural order.

There is a Latin square of order n for any positive integer n. It is easy to construct a Latin square of any order n by placing in the first row the elements of the base set (for example the first n natural numbers) and the other rows we obtain them by cyclically displacing the elements of the base set. Thus far it is known the number $l(n)$ of all Latin reduced squares of order n, where $n \leq 11$ [11]. There is a relation between the number of reduced Latin squares and Latin squares $L(n)$ given by $L(n) = n!(n-1)!l(n)$ (when $n \geq 2$). Then, the knowledge of the number of reduced Latin squares, permit us to count the exact number of all Latin squares of order $n \leq 11$. The explicit number can be found in [12].

2.1 Generation of latin squares

Given an n-order Latin square, it is possible to obtain from it, Latin squares of order any multiple of the order of the given one, using *cyclic-union operation* introduced in the following manner.

Let $A_1, \ldots, A_r$, be r Latin squares of order n where the respective base sets are renamed them of the form $\{1, \ldots, n\}$, $\{n+1, \ldots, 2n\}$, ..., $\{(r-1)n+1, \ldots, rn\}$.

Definition 2.1. The cyclic-union of these squares is defined as the matrix:

$$A_1 * A_2 * \ldots * A_r = \begin{array}{|c|c|c|c|c|} \hline A_1 & A_2 & \ldots & A_{r-1} & A_r \\ \hline A_r & A_1 & \ldots & A_{r-2} & A_{r-1} \\ \hline \vdots & \vdots & \ldots & \vdots & \vdots \\ \hline A_2 & A_3 & \ldots & A_r & A_1 \\ \hline \end{array}$$

That is, each row is obtained by cyclically rotating the Latin squares of the previous row, passing the first Latin square to the last place. By construction, this square is a Latin square, and the order is nr.

Proposition 2.1. *The cyclic-union of the n-order Latin squares $A_1 * A_2 * \ldots * A_r$ is a Latin square*

Example 2.2. *Let $A_1 = A_2 = A_3$ three identical two-order square Latin with*

1	2
2	1

Renaming we obatin

$$A_1 = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 2 & 1 \\ \hline \end{array} \quad A_2 = \begin{array}{|c|c|} \hline 3 & 4 \\ \hline 4 & 3 \\ \hline \end{array} \quad A_3 = \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 6 & 5 \\ \hline \end{array}$$

Then

$$A_1 * A_2 * A_3 = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 2 & 1 & 4 & 3 & 6 & 5 \\ \hline 5 & 6 & 1 & 2 & 3 & 4 \\ \hline 6 & 5 & 2 & 1 & 4 & 3 \\ \hline 3 & 4 & 5 & 6 & 1 & 2 \\ \hline 4 & 3 & 6 & 5 & 2 & 1 \\ \hline \end{array}$$

Another way to generate easily computable Latin squares is the operation that we will call *Kronecker's composition* because of its similarity to the left Kronecker product of matrices operation.

Given two square matrices A_1 and A_2 of respective orders n_1 and n_2 respectively.

Definition 2.2. The Kronecker commposition of these square matrices is defined in the following manner:

Writting

$$A_1 = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n_1} \\ a_{21} & a_{22} & \dots & a_{2n_1} \\ \vdots & \vdots & & \vdots \\ a_{n_1 1} & a_{n_1 2} & \dots & a_{n_1 n_1} \end{pmatrix}, \quad \text{and} \quad A_2 = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n_2} \\ b_{21} & b_{22} & \dots & b_{2n_2} \\ \vdots & \vdots & & \vdots \\ b_{n_2 1} & b_{n_1 2} & \dots & b_{n_1 n_1} \end{pmatrix}$$

$$A_1 \circ A_2 = \begin{pmatrix} a_{11}A_2 & a_{12}A_2 & \dots & a_{1n_1}A_2 \\ a_{21}A_2 & a_{22}A_2 & \dots & a_{2n_1}A_2 \\ \vdots & \vdots & & \vdots \\ a_{n_1 1}A_2 & a_{n_1 2}A_2 & \dots & a_{n_1 n_1}A_2 \end{pmatrix} =$$

$$\begin{pmatrix} a_{11}, b_{11} & \dots & a_{11}, b_{1n_2} & & a_{1n_1}, b_{11} & \dots & a_{1n_1}, b_{1n_2} \\ \vdots & & \vdots & \dots & \vdots & & \vdots \\ a_{11}, b_{n_2 1} & \dots & a_{11}, b_{n_1 n_1} & & a_{1n_1}, b_{n_2 1} & \dots & a_{1n_1}, b_{n_1 n_1} \\ & \vdots & & & & \vdots & \\ a_{n_1 1}, b_{11} & \dots & a_{n_1 1}, b_{1n_2} & & a_{n_1 1}, b_{11} & \dots & a_{n_1 1}, b_{1n_2} \\ \vdots & & \vdots & \dots & \vdots & & \vdots \\ a_{n_1 1}, b_{n_2 1} & \dots & a_{n_1 1}, b_{n_1 n_1} & & a_{n_1 n_1}, b_{n_2 1} & \dots & a_{n_1 n_1}, b_{n_1 n_1} \end{pmatrix}$$

Proposition 2.2. *Let A_1, A_2 be two matrices of orders n_1 and n_2 respectively, corresponding to Latin squares. Then The Kronecker composition of these matrices $A_1 \circ A_2$, is an $n_1 n_2$-order Latin square.*

Proof. Suppose that $a_{1i}b_{1j} = a_{\ell k}b_{rs}$, then $a_{1i} = a_{\ell k}$ and $b_{1j} = b_{rs}$, but taking into account that A_1 and A_2 are Latin squares we have that $(1, i) = (\ell, k)$ and $(1, j) = (\ell, k)$ $(1, j) = (r, s)$ □

Remark 2.1. If the base sets of A_1 and A_2 are $\{a_1, \dots, a_{n_1}\}$ and $\{b_1, \dots, b_{n_2}\}$, respectively then, the basis set of $A_1 \circ A_2$ is $\{1, 2, \dots, n_1 n_2\}$ obtained after renaming the set

$$\{(a_1, b_1), \dots, (a_1, b_{n_2}), (a_2, b_1), \dots, (a_2, b_{n_2}), \dots, (a_{n_1}, b_1), \dots, (a_{n_1}, b_{n_2})\}$$

following natural order $((a_1, b_1) \to 1, \dots, (a_1, b_{n_2}) \to n_2, \dots)$.

For more details on these operations see [13].

In 1974 Dénes and Keedwell [9], define the product of two Latin squares, using right Kronecker

product operation and also define the non-uniform product of Latin rectangles. In fact, our operations could be deduced of the constructions realised by these authors.

More information about distinct methods to generate Latin squares can be found in [10].

Remark 2.2. We want to observe by means an example, the little difference between Kronecker composition and cyclic-union os Latin squares.

Let $A_1 = \begin{array}{|c|c|}\hline 1&2\\\hline 2&1\\\hline\end{array}$, $A_2 = \begin{array}{|c|c|}\hline 3&4\\\hline 4&3\\\hline\end{array}$ and $A_3 = \begin{array}{|c|c|}\hline 5&6\\\hline 6&5\\\hline\end{array}$.

Then

$$A_1 * A_2 * A_3 = \begin{array}{|c|c|c|c|c|c|}\hline 1&2&3&4&5&6\\\hline 2&1&4&3&6&5\\\hline 5&6&1&2&3&4\\\hline 6&5&1&2&3&4\\\hline 3&4&5&6&1&2\\\hline 4&3&6&5&2&1\\\hline\end{array}$$

and

$$A_1 \circ A_2 \circ A_3 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 1&2&3&4&5&6&7&8\\\hline 2&1&4&3&6&5&8&7\\\hline 3&4&1&2&7&8&5&6\\\hline 5&6&7&8&1&2&3&4\\\hline 6&5&7&8&1&2&3&4\\\hline 6&5&8&7&2&1&4&3\\\hline\end{array}$$

That, they are clearly different.

If we wanted to show some relation between the operations, we should use an auxiliary Latin square X. To obtain from $A_1 * A_2 * A_3$ the Latin square $A_1 \circ A_2 \circ A_3$, we add X to the end of the first row-block of $A_1 * A_2 * A_3$ and in permuting with the neighbouring square in each one of the next row-blocks until you reach the first place.

3 Description of the Algorithm

As mentioned, the calculation of the Latin squares continues to be a reason for investigation and analysis. In this line, it is worth mentioning the development of a new algorithm presented by [14], based on bitwise operations.

We rely on the bitwise working method for its performance and efficiency in the consumption of the processor. As well as, it is much more efficient and fast than other methods. It is due to the operations are treated directly by the low-level instructions on the adjacent processor.

To make the process more comprehensible, we present here the development of the algorithm for the cyclic-union of 4 Latin squares of order 2 and order 4, obtaining in this way Latin squares of 8x8 and 16x16 orders. We do this to maintain a small order due to the high execution time required by the calculation with high values of n, and thus it can be easily tested.

To perform the cyclic-union of the matrices is needed a module to perform the renaming of the symbols of the base set of the Latin squares. This code has been implemented using the bitwise operations technique. Below is the code developed to perform such renaming.

```
void renombre(){
    int desplazamiento =N;
    int desplazamiento2 = N≪1;
    int desplazamiento3= N≪2;
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; i++){
            L2 [i] [j] = L2 [i] [j] ≪ desplazamiento;
            L3 [i] [j] = L3 [i] [j] ≪ desplazamiento2;
            L4 [i] [j] = L4 [i] [j] ≪ desplazamiento3;
        }
    }
}
```

Once the symbols have been renamed, we proceed to the coherent cyclic-union of the calculated Latin squares, in this case, L, L2, L3 and L4.

The cyclic-union operation is then performed

```
void yux () {
Renombre ();
    for (int i=0; i<N; i++){
        F [i] [j] = L [i] [j];
        F [j+2] = L[i] [j];
        F [i+4] [j+4] = L[i] [j];
        F [i+6] [j+6] = L [i] [j];

        F [i+2] [j] = L2 [i] [j];
        F [i+4] [j+2] = L2 [i] [j];
        F [i+6] [j+4] = L2 [i] [j];
        F [i] [j+6] = L2 [i] [j];

        F [i+4] [j] = L3 [i] [j];
        F [i+6] [j+2] = L3 [i] [j];
        F [i] [j+4] = L3 [i] [j];
        F [i+2] [j+6] = L3 [i] [j];

        F [i+6] [j] = L4 [i] [j];
        F [i] [j+2] = L4 [i] [j];
        F [i+2] [j+4] = L4 [i] [j];
        F [i+4] [j+6] = L4 [i] [j];

        }
    }
}
```

4 Conclusions

With the help of algorithms we obtain Latin squares, for example

2^2	2^3	1	2	2^{17}	2^{19}	2^{18}	2^{16}	2^8	2^{11}	2^9	2^{10}	2^6	2^4	2^5	2^7
2^3	2^2	2	1	2^{19}	2^{18}	2^{16}	2^{17}	2^{11}	2^{10}	2^8	2^9	2^7	2^5	2^4	2^6
1	2	2^2	2^3	2^{16}	2^{17}	2^{19}	2^{18}	2^9	2^8	2^{10}	2^{11}	2^4	2^6	2^7	2^5
2	1	2^3	2^2	2^{18}	2^{16}	2^{17}	2^{19}	2^{10}	2^9	2^{11}	2^8	2^5	2^7	2^6	2^4
2^6	2^4	2^5	2^7	2^2	2^3	1	2	2^{17}	2^{19}	2^{18}	2^{16}	2^8	2^{11}	2^9	2^{10}
2^7	2^5	2^4	2^6	2^3	2^2	2	1	2^{19}	2^{18}	2^{16}	2^{17}	2^{11}	2^{10}	2^8	2^9
2^4	2^6	2^7	2^5	1	2	2^2	2^3	2^{16}	2^{17}	2^{19}	2^{18}	2^9	2^8	2^{10}	2^{11}
2^5	2^7	2^6	2^4	2	1	2^3	2^2	2^{18}	2^{16}	2^{17}	2^{19}	2^{10}	2^9	2^{11}	2^8
2^8	2^{11}	2^9	2^{10}	2^6	2^4	2^5	2^7	2^2	2^3	1	2	2^{17}	2^{19}	2^{18}	2^{16}
2^{11}	2^{10}	2^8	2^9	2^7	2^5	2^4	2^6	2^3	2^2	2	1	2^{19}	2^{18}	2^{16}	2^{17}
2^9	2^8	2^{10}	2^{11}	2^4	2^6	2^7	2^5	1	2	2^2	2^3	2^{16}	2^{17}	2^{19}	2^{18}
2^{10}	2^9	2^{11}	2^8	2^5	2^7	2^6	2^4	2	1	2^3	2^2	2^{18}	2^{16}	2^{17}	2^{19}
2^{17}	2^{19}	2^{18}	2^{16}	2^8	2^{11}	2^9	2^{10}	2^6	2^4	2^5	2^7	2^2	2^3	1	2
2^{19}	2^{18}	2^{16}	2^{17}	2^{11}	2^{10}	2^8	2^9	2^7	2^5	2^4	2^6	2^3	2^2	2	1
2^{16}	2^{17}	2^{19}	2^{18}	2^9	2^8	2^{10}	2^{11}	2^4	2^6	2^7	2^5	1	2	2^2	2^3
2^{18}	2^{16}	2^{17}	2^{19}	2^{10}	2^9	2^{11}	2^8	2^5	2^7	2^6	2^4	2	1	2^3	2^2

Finally, we want to remark that the algorithm presented, optimises the time of obtaining a Latin square of order $N = nr$. The generation time of a Latin square is at least polynomial order $O(N^3)$ (depending on the algorithm), while the presented algorithm is of the order $O(n^3 + nr)$.

Remark that, the cost of choosing n symbols and placing them in n rows implies n^2 operations, but if we want to obtain a Latin square, we can not have collisions involving a cost in execution. In the standard approach for the generation of random Latin squares (to be useful in cryptography) of order n, obtained by applying movements to the one obtained by cyclic displacement of the base set, n^3 changes are required, (see [10], [5]). Therefore, if we want to obtain a Latin square of order rn we need $(rn)^3$ movements. But if we construct it from a Latin square of order n, renaming it r times and juxtaposing it we only require $n^3 + rn$.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Bonner A. Selected works of ramon llull. Princeton: Princeton University Press. 1985;1:305-568.

[2] Euler L. Recherches sur une nouvelle espece de quarres magiques. Verh. v. h. Zeeuwsch Genoottsch. der Wetensch, Vlissingen. 1782;9:85-239.

[3] Ritter T. Latin squares: A literature survey. Research Comments from Ciphers; 2003. Available: http://www.ciphersbyritter.com/RES/LATSQ.HTM

[4] Koscielny C. Generating quasigroups for cryptographic applications. Int. J. Appl. Math. Comput. Sci. 2002;12(4):559-569.

[5] Gallego I. Análisis de algoritmos para generación de cuadrados latinos aleatorios para criptografía TFM. Universidad Nacional de la Plata; 2014.

[6] Hulpke A, Kaski P, Östergård PRJ. The number of Latin squares of order 11. Math. Comp. 2011;80:1197-1219.

[7] Kolesova G, Lam CWH, Thiel L. On the number of 8×8 Latin squares. J. Combin. Theory Ser. A. 1990;54:143-148.

[8] McKay BD, Meynert A, Myrvold W. Small Latin squares. Quasigroups, and loops, J. Combin. Des. 2007;15:98-119.

[9] Dénes J, Keedwell AD. Latin squares and their applications. Academic Press, New York-London; 1974.

[10] Jacobson M, Matthews P. Generating uniformly distributed Latin squares. J. Combin. Des. 1996;4(6):405-437.

[11] McKay BD, Wanless IM. On the number of Latin squares. Annals of Combinatorics. 2005;9(3):335-344.

[12] OEIS. The On-Line Encyclopedia of Integer Sequences. A002860 - Number of Latin squares of order n. (Last accessed on April 30, 2017 at 12:45).
Available: http://oeis.org/A002860

[13] Roca-Borrego D. Estudio y análisis de los cuadrados latinos para la optimización del proceso de obtención. Universitat Politècnica de Catalunya, Spain; 2017. (In published TFG thesis).

[14] Yordzhev K. Bitwise operations in relation to obtaining latin squares. British Journal of Mathematics & Computer Science. 2016;5:1-7.

© *2017 García-Planas and Roca-Borrego; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.*

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/19542