

Juxtaposition operation to obtain Latin squares

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 juxtaposition operation defined for effect.

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

2010 Mathematics Subject Classification: 68N15; 15A36

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 Demonstrativa* (1283) four Latin squares of order 4, using as symbols fire, air, water and earth, in an attempt to explain the world through combinatorial numbers [Bonner (1985)]. However, the first formal definition was given by Leonhard Euler [Euler (1782)], 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 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.

The exact number of Latin squares of order $n \geq 10$ has not been known so far. 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 to develop a specific computer tool for the generation of 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.

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 $n \leq 11$ [McKay, Wanless (2005)]. 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 [OEIS-A002860 (2011)].

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 *juxtaposition operation* introduced in the following manner.

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

Definition 2.1. The juxtaposition of these squares is defined as the matrix:

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

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 juxtaposition of the n -order Latin squares $A_1 * A_2 * \dots * A_r$ is a Latin square

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

Given two Latin squares A_1 and A_2 of respective orders n_1 and n_2 and base set $\{a_1, \dots, a_{n_1}\}$ and $b_1, \dots, b_{n_2}\}$ respectively

Definition 2.2. The Kronecker product of this Latin squares is defined in the following manner:
Writting

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

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

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

Proposition 2.2. The Kronecker product of two Latin squares $A_1 \circ A_2$ of orders n_1 and n_2 respectively, is an $n_1 n_2$ -order Latin square.

Remark 2.1. The basis set 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 $((a_1, b_1) \rightarrow 1, \dots, (a_1, b_{n_2}) \rightarrow n_2, \dots)$.

For more details on these operations see [Roca-Borrego (2017)].

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 [Yordzhev (2016)], based on bitwise operations.

Thus, the development of optimisation in the calculation of a random Latin square has been based both on the code previously mentioned and on the theorem of the juxtaposition of Latin squares.

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 juxtaposition 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 juxtaposition 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 renmbre(){
```

```

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 juxtaposition of the calculated Latin squares, in this case, L, L2, L3 and L4.

The juxtaposition 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

4	8	1	2	131072	524288	262144	65536	256	2048	512	1024	64	16	32	128
8	4	2	1	524288	262144	65536	131072	2048	1024	256	512	128	32	16	64
1	2	4	8	65536	131072	524288	262144	512	256	1024	2048	16	64	128	32
2	1	8	4	262144	65536	131072	524288	1024	512	2048	256	32	128	64	16
64	16	32	128	4	8	1	2	131072	524288	262144	65536	256	2048	512	1024
128	32	16	64	8	4	2	1	524288	262144	65536	131072	2048	1024	256	512
16	64	128	32	1	2	4	8	65536	131072	524288	262144	512	256	1024	2048
32	128	64	16	2	1	8	4	262144	65536	131072	524288	1024	512	2048	256
256	2048	512	1024	64	16	32	128	4	8	1	2	131072	524288	262144	65536
2048	1024	256	512	128	32	16	64	8	4	2	1	524288	262144	65536	131072
512	256	1024	2048	16	64	128	32	1	2	4	8	65536	131072	524288	262144
1024	512	2048	256	32	128	64	16	2	1	8	4	262144	65536	131072	524288
131072	524288	262144	65536	256	2048	512	1024	64	16	32	128	4	8	1	2
524288	262144	65536	131072	2048	1024	256	512	128	32	16	64	8	4	2	1
65536	131072	524288	262144	512	256	1024	2048	16	64	128	32	1	2	4	8
262144	65536	131072	524288	1024	512	2048	256	32	128	64	16	2	1	8	4

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 $O(n^3 + nr)$

References

- Bonner, A. (1985). Selected Works of Ramon Llull, Princeton: Princeton University Press, 1 305–568.
- Euler, L. (1782). Recherches sur une nouvelle espece de quarres magiques. Verh. v. h. Zeeuwsch Genootsch. der Wetensch, Vlissingen, 9 85–239.
- McKay, B. D. and Wanless, I. M. (2005). On the number of Latin squares. Annals of Combinatorics 9(3), 335–344.
- OEIS. The On-Line Encyclopedia of Integer Sequences. A002860 - Number of Latin squares of order n . Available: <http://oeis.org/A002860>, <http://oeis.org/A002860> (Last accessed on April 30, 2017 at 12:45).
- Roca-Borrego, D. (2017). Estudio y analisis de los cuadrados latinos para la optimizacin del proceso de obtencion. Universitat Politècnica de Catalunya, Spain *Inpublished TFG thesis*.
- K. Yordzhev, K. (2016). Bitwise Operations in Relation to Obtaining Latin Squares. British Journal of Mathematics & Computer Science 5, 1–7.