Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 20, Number 1 (2024), pp. 161-167
1© Research India Publications
http://www.ripublication.com

# Parallel Programming in Computer Algebra Systems

**Kostas Zotos**

*South-west University "Neofit Rilski" Faculty of Science and MathematicsDepartment of Informatics*
*66 Ivan Michailov st. 2700 Blagoevgrad, Bulgaria*

**Abstract**

The acceptance of using mathematical software to solve problems in the physical, biological, and social sciences has increased dramatically because of technological advancements. Parallel programming is one of these advancements that has many benefits and that's why an increased use has in the new versions of mathematical packages. On the other hand, there are difficulties in its use which make it difficult to apply it to all mathematical problems. In this paper, we will examine how parallelization is done in some famous Computer Algebra Systems (CAS) and what it offers us in general.

**Keywords:** *Computer Algebra Systems; Parallel Programming; CAS performance; CAS Parallelism; MATLAB; Maple; Mathematical*

## Computer Algebra Systems

The programs that make use of Computational Algebra methods are called **Computer Algebra Systems (CAS)**. A CAS aims to automate laborious and challenging algebraic manipulation tasks. These systems differ widely from one another in terms of their specific functions and applications. For example, some of them provide a programming language for the user to define their procedures. CASs can be divided into two large categories according to their use: a) the General-purpose CAS or otherwise the systems that contain functions for most fields of Mathematics e.g., Maple, Mathematical, etc., and b) special-purpose CAS which specialize in specific areas of Mathematics e.g. PARI (Number Theory), DELiA (Differential Equations), etc.

The first CASs were created in the LISP programming language. Later, new CASs, such as the MAPLE and MATHEMATICA were created in the C programming language, which could better manage the computer's operational resources like memory. These new CASs consumed less memory and therefore could be used on smaller computer systems, which was the reason for their widespread.

## The Basic Components of a Cas

In the beginning of CAS history - the computer, justifying its name, acted as a powerful programmable calculator, able to quickly and automatically (according to a given program) perform complex and cumbersome arithmetic and logical operations in numbers. Advances in computational Mathematics and the continuous improvement of numerical methods make it possible to solve any mathematical problem in this way. It is important to note that the result of calculations in this case is represented by a finite number in numerical form that is, using decimal places. Sometimes the result is represented by a set (array, matrix) of such numbers, but the essence of the representation does not change from this - the result is in the form of a finite decimal numerical number. However, such a result often did not satisfy professional mathematicians, and here is why. Most results of non-trivial mathematical calculations in classical Mathematics are traditionally written in symbolic form: using special known numbers: e.g. *pi* and irrational values - using a root. It is believed that otherwise there is a fundamental loss of accuracy.

From the beginning mathematical science proceeded to introduce and use many symbols for processing and developing mathematical ideas, making their role decisive in its further development. With time only several sign symbols survived with the main cause, among others, the response to the laws of convenience, consistency, and functionality. The development of Mathematics is directly related to the development of different semiotic systems. According to **Duval** (1999), “Progress in Mathematics is linked to evolution (development) of different semiotic systems of the sensory systems: of language and image" [1]. The role of symbols in its mediation of expressing ideas and conceptualizing new ideas was a pervasive force in Mathematics and these symbols evolved as mathematical concepts that from empirical and concrete existence were transferred to general and abstract thought. Knowledge of symbols is the study of the structure of mathematical signs and symbols and the procedures involved in handling them objects into meaningful concepts, processes, and representations. More practically it intends to understand how symbols help us to know the way to do mathematical calculations.

Within the environment of mathematical thought the evolution of symbols starts from the understanding that symbols are tools that help us in observation and are part of larger systems that may evolve with human knowledge or are cultural artifacts that support mathematical ideas. Mathematical symbolism is one of the strongest means of expression in Mathematics. The use of symbols to represent mathematical ideas contributed significantly to the impressive development of mathematical science, helping us with a flexible and economical tool.

Of course, after the rapid improvement of computer systems, a person in computer calculations wanted more: why not make the computer perform transformations in the traditional ways for Mathematics (fractional-rational transformations, substitutions, simplifications, solving equations, differentiation-discrimination, and so on.). This led to the creation of computational systems of symbolic Mathematics, designed for a wide range of users - non-mathematical professionals. Thus began the CAS era in the mid-1960s.

**The basic components of a CAS are:**

- A **user interface** that lets the user insert and manipulate mathematical problems. The CAS's interfaces vary. There are many with a command line interface (CLI), others with a graphical user interface (GUI), and a few with only some libraries.
- A **programming language**, a **simplifier**, and an **interpreter**. Also, a canonical function that rewrites a formula to its canonical form.
- A **library** with many efficient algorithms implemented for "common" operations that let the user not "reinvent the wheel".
- An **inner arithmetic** (E.g. arbitrary precision) that can manipulate mathematical objects numerically, symbolically, and graphically.
- A **memory manager**.
- A **graphing editor** for the creation of two-dimensional and three-dimensional graphs of any complexity, visual diagrams, and not only for simple construction but also for connecting a graph with a formula, in which a change in a parameter is immediately reflected in the graph curve.
- The development of web documents and networking capabilities for sharing, receiving updates, and support.

Another important part of a symbolic computer algebra system is if it uses the same language for data storage, manipulation, and implementation. There are CASs that the previous three are the same (E.g. Reduce, Axiom, and Sympy) and others that the core uses a different implementation language (E.g. Maple, and Mathematica). Finally, there are CASs with a clear distinction between these three (E.g. Cadabra) [2].

## Parallel Computing

**Parallel computing** is using multiple processors concurrently to solve a computational problem. The objective is to use as much as we can computer resources to solve a problem faster than using serial computation. Solving mathematical algorithms in a parallel way creates many problems that have to do with how to transform a serial problem into a parallel one, the design of an appropriate hardware architecture to support the parallel processing of the algorithms, round-off analysis, etc. Some metrics show us the degree of speed improvement and efficiency from parallelism [3].

Solving **linear systems** is one of the most important problems in scientific computing, as many well-known real-world applications and problems are modeled through them. Very often, the size of these systems is very large and therefore their solution is particularly time-consuming. Fatefully, the efficient parallelization of their solution becomes fundamental to be able to derive meaningful results in a satisfactory time.

The methods for solving linear systems are usually divided into systematic ones (which are slower to execute, are mainly suitable for dense coefficient matrices, and provide the solutions of the linear system, always, with straight mathematical calculations) and iterative/approximate ones. The latter are usually much faster in their execution (this is one of the main reasons why they were developed and applied in practice), they are suitable for sparse arrays, but their convergence is not certain, so they are not always applicable.

Parallelization usually reduces the overall completion time. However, the computation load is not equally distributed among the processors and many of them remain inactive for a significant period. For example, the division of the rows of an A table can be done in two ways:

**A)** With **sectional allocation** which is the simplest way of allocating table rows to processors. Here a subset of contiguous rows of the system coefficient table is assigned to each processor. Therefore each $P_i$ processor takes n/p contiguous lines and more specifically lines of order *i(n/p)* to *(i+1)(n/p)-1*. For example, if the number of processors is *P=4* and the number of system equations is *n=16* then the distribution of the lines will be as shown in the following Picture 1.

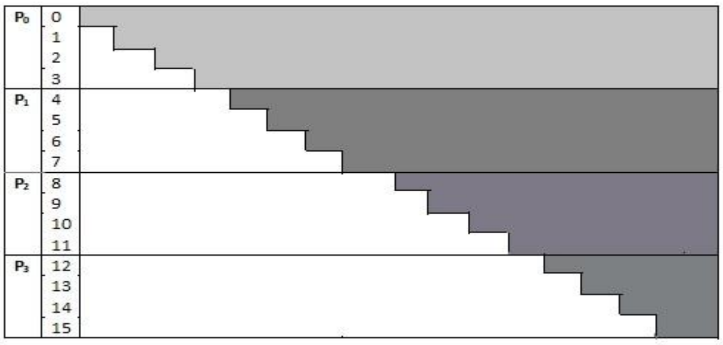

**Figure 1.** Sectional distribution of table rows

**B)** With a **circular distribution** where the lines of the table that each processor takes over are not contiguous. Each processor takes over lines spaced *p* apart. For example, the processor $P_i$ takes over the lines of order *i,i+p,i+2p*, etc. The assignment of the lines is done cyclically (each line of order I is allocated to the processor *I mod p*). For example, if the number of processors is P=4 and the number of system equations is n=16 then the distribution of the lines will be as shown in the following Picture 2.

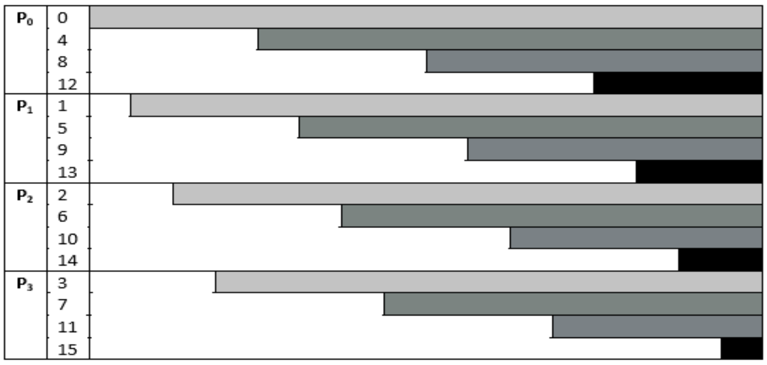

**Figure 2.** Circular distribution of table rows

Sectional distribution is easier to implement than circular distribution. On the other hand, however, the circular distribution leads to a fairer distribution of the rows of the table among the processors, based on the calculation load it entails. This can also be seen in the Pictures 1 and 2. As we can see with the partitioning the processors go down one by one earlier. In contrast to circular distribution, they remain active and participate in calculations for more steps.

One of the biggest challenges to achieving the best parallel program performance is usually synchronization and communication among the various subtasks. But there is another factor that exists when we deal with big data in parallel *energy consumption*. In new versions of CASs, there are tools to assess power efficiency. So, you can easily evaluate and report power and energy consumption.

Launching and managing numerous Wolfram Language kernel processes from within a single master Wolfram Language is the foundation of Wolfram Language parallel computing. Numerous methods are available in the Wolfram Language for running parallel workers, including locally on the same machine or remotely across a network. Furthermore, the network may consist of a heterogeneous grid or a homogeneous grid managed by a specific management application [4].

With the Parallel Computing Toolbox of MATLAB, you can use computer clusters, GPUs (using *gpuArray*), and multicore processors to solve computationally and data-intensive problems. Also, you can run applications on workers (locally running MATLAB computational engines) and take full advantage of the processing power of multicore desktops [5].

Parallelism is made possible in Maple by the Task Programming Model, which runs several tasks inside of a single process, and with the Grid package which launches several processes (with its independent memory) to enable parallelism [6].

**Advantages/disadvantages of Parallel Computing**

When a CAS is programmed in parallel, it can process and solve problems more efficiently by utilizing its resources. Complex problems can be broken down into smaller tasks by parallel programs, which can process each task independently at the same time. On the other hand, parallelization is not a magic bullet that works for every mathematical problem. It also brings up certain issues that must be resolved, like compatibility, complexity, and communication. The following Table 1 summarizes the advantages and disadvantages of parallelism.

**Table 1.** Advantages/disadvantages of Parallel Computing

| Advantages | Disadvantages |
|---|---|
| **Enhanced Output** The capacity of parallel computing to greatly enhance performance is one of its main benefits. Parallel computing handles complex calculations and data-intensive operations far faster than sequential computing because it divides tasks across multiple processing units. This in Mathematics is very helpful, especially for jobs like data analysis, scientific simulations, and producing high-quality graphics. | **Intricacy** Parallel computing implementation can be difficult and complex. Carefully planning and taking into account task dependencies are necessary when creating algorithms and programs that can be parallelized efficiently. Furthermore, testing and debugging concurrent programs can be more difficult than sequential ones. |
| **Quick process** Parallelism can handle the demands of real-time processing. | **Cost** Specialized hardware, such as multi-core CPUs, GPUs, or clusters of linked computers, is frequently needed for parallel computing. Such infrastructure can be expensive to acquire and maintain. |
| **The ability to scale** Excellent scalability is a feature of parallel computing, allowing it to effectively manage increasing workloads as the number of processing units rises. Parallel computing can fully utilize these resources as technology develops and more potent processors become accessible, allowing for faster and more effective data and task processing. | **The Amdahl Law** This law tells us that the sequential part of the algorithm determines how much faster a computation can be made overall by parallelizing it. Put differently, the benefits of parallel computing may not materialize as expected if a sizable portion of the computation needs to be performed sequentially. This limits the amount of speedup that can be achieved through parallel computing. |
| **Maxim use of resources** By using several processing units at once, parallel computing maximizes resource usage. It maximizes the effectiveness of hardware resources by ensuring that no processing power is wasted. | **The communication cost** Tasks in parallel computing frequently require synchronization and communication with one another. Performance as a whole may be impacted by the complexity and possible bottlenecks this communication overhead may introduce. To lessen these problems, load balancing and effective synchronization techniques are crucial. |

## CONCLUSIONS

There is a trend toward the development of mathematical software tools utilized in science, research, and engineering. The aforementioned areas offer a variety of subdivisions for mathematical software applications; however, the selection of software necessary for mathematical analyses depends on the purpose of the research or the problem being studied. Due to the limitations of these software programs, updated versions frequently complement or enhance the features found in one type, thereby improving the program's multitasking capabilities.

When a CAS is programmed in parallel, it can process and solve problems more efficiently by utilizing its resources. Complex problems can be broken down into smaller tasks by parallel programs, which can process each task independently at the same time. Modern CASs can operate more quickly thanks to parallel processing, which divides more complex computational problems into smaller ones and processes them simultaneously.

On the other hand, parallelization is not a magic bullet that works for every mathematical problem. It also brings up certain issues that must be resolved, like compatibility, complexity, and communication. Data and messages are exchanged between processors or cores working on different subproblems during communication, which can impact performance and accuracy and add overhead and latency. Sometimes, the complexity of an algorithm makes its development, testing, and debugging challenging and time-consuming.

## REFERENCES

**[1]** Duval, R. (1999) Representation, Vision and Visualization: Cognitive Functions in Mathematical Thinking. Basic Issues for Learning. Proceedings of the Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education, Cu-ernavaca, Morelos, México.
**[2]** https://cadabra.science/
**[3]** https://mathworld.wolfram.com/ParallelComputing.html
**[4]** https://reference.wolfram.com/language/ParallelTools/tutorial/Introduction.html#71761281
**[5]** https://www.mathworks.com/products/parallel-computing.html
**[6]** https://www.maplesoft.com/support/help/maple/view.aspx?path=ProgrammingGuide%2FChapter15