Tree structures in Poisson series processors

Juan F. Navarro
University of Alicante (Spain)

jf.navarro@ua.es

Abstract

Much of the work concerned with the application of perturbation theories in celestial mechanics, and particularly in the development of analytical theories of the motion of celestial bodies, can be reduced to algebraic operations on Poisson series.

The aim of this contribution is to make a review on the use of tree structures for the storage of Poisson series. We analyse a type of structure based on maps, as well as its representation in the form of red–black tree. We also compare the complexity of some fundamental algorithms in their corresponding computer implementation as lists and red–black trees.

Keywords
Symbolic computation, Poisson series, Maps, Multimaps, Complexity

1 Introduction

Since the early sixties, investigators used computers to generate analytical expressions. The first Poisson series processors were born to deal with the theory of the Moon, considered as one of the hardest problems in celestial mechanics. Later, analytical theories for the rotation of the Earth (Kinoshita, 1977) were treated with the help of symbolic computation packages. Nowadays there are many open problems which requires massive symbolic computation.

Many Poisson series processors have been developed until now, as PSP (Broucke, 1970), MAO (Mechanized Algebraic Operations) (Rom, 1969), TRIGMAN (Trigonometric Manipulator) (Jefferys, 1970), MSNam (Henrard, 1986), PARSEC (Richardson, 1989), PSPC (Abad and San–Juan, 1993), and others. We also would like to mention that MSNam software (Manipulateur de Séries de Namur) was first written by H. Claes, J. Henrard, M. Moons and J.M. Zune. It was later improved by M. Moons (1993) and the last version in Fortran 90 was made by J. Henrard in 2004. In this version, the arguments and exponents of the series and the indication that the trigonometric expression is a cosine or a sine are coded and packed in a large array of integers.

Furthermore, several general purpose systems such as Mathematica, Macsyma, Reduce, Maple, Matlab and others have been designed to treat a wide range of problems from many branches of Science. Because of their universality, they are not as efficient as special purpose systems designed for solving some specific applications. In particular, high accuracy analytical problems of celestial mechanics involving perturbation methods require specific symbolic processors.

In this paper, we analyse the red–black tree structure and the way it can be used to represent computationally a Poisson series. This structure leads to best computational times in the basic operations with Poisson series, as addition and multiplication of Poisson series.

2 Poisson series as a symbolic object

In this section, we follow F. San–Juan and A. Abad (2001) to introduce the representation of a mathematical object in a computer. We will focus our attention on the set $\mathcal{P}_{n,m}$ of Poisson series with n polynomial variables $x_1, \ldots, x_n$, and m angular variables $\phi_1, \ldots, \phi_m$. A Poisson series is a map $P : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ such that

$$P(x_1, \ldots, x_n, \phi_1, \ldots, \phi_m) = \sum_{i_1,\ldots,i_n} \sum_{j_1,\ldots,j_m} C^{j_1,\ldots,j_m}_{i_1,\ldots,i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \frac{\cos}{\sin}(j_1\phi_1 + \cdots + j_m\phi_m)\,, \quad (1)$$

where $i_1, \ldots, i_n, j_1, \ldots, j_m \in \mathbb{Z}$. The set $\mathcal{P}_{n,m}$, with the addition of Poisson series and the multiplication of a Poisson series by a real number, is a vectorial space. The partial derivative of a Poisson series with respect to a polynomial or angular variable is also a Poisson series, as well as the multiplication of two Poisson series.

We look for a canonical representation for each equivalence class defined in $\mathcal{P}_{n,m}$. For that purpose, the following operations must be performed over each Poisson series:

1. The first non–zero coefficient of the angular variables must be positive. If not, we will apply the following rules:

$$\sin(-\lambda_i\phi_i + \cdots) = -\sin(\lambda_i\phi_i - \cdots)\,, \quad \cos(-\lambda_i\phi_i + \cdots) = \cos(\lambda_i\phi_i - \cdots)\,.$$

2. The terms of a Poisson series will be ordered following a lexicographical order, as follows: let us condider two terms of a Poisson series, τ_1 and τ_2, given by

$$\begin{aligned}
\tau_1 &= C^{i^{(1)}_{n+1},\ldots,i^{(1)}_{n+m}}_{i^{(1)}_1,\ldots,i^{(1)}_n} x_1^{i^{(1)}_1}\, x_2^{i^{(1)}_2} \cdots x_n^{i^{(1)}_n} T^{(1)}(i^{(1)}_{n+1}\phi_1 + \cdots + i^{(1)}_{n+m}\phi_m)\,,\\
\tau_2 &= C^{i^{(2)}_{n+1},\ldots,i^{(2)}_{n+m}}_{i^{(2)}_1,\ldots,i^{(2)}_n} x_1^{i^{(2)}_1}\, x_2^{i^{(2)}_2} \cdots x_n^{i^{(2)}_n} T^{(2)}(i^{(2)}_{n+1}\phi_1 + \cdots + i^{(2)}_{n+m}\phi_m)\,.
\end{aligned}$$

   We say that $\tau_1 < \tau_2$ if for the first $k \in \{1, \ldots, n, n+1, \ldots, n+m\}$ such that $i^{(1)}_k \neq i^{(2)}_k$, then $i^{(1)}_k < i^{(2)}_k$ is verified or, if for all $k \in \{1, \ldots, n+m\}$, $i^{(1)}_k = i^{(2)}_k$, and $T^{(1)} = \cos$ and $T^{(2)} = \sin$.

3. The terms of a Poisson series with identical polynomial and angular part must be grouped together.

3 Red–black trees and maps

As pointed out in (San–Juan and Abad, 2001), most of the operations involving a series are based on navigating and searching through the structure that represents the series. For example, the addition of two Poisson series is equivalent to insert each term of one series into the other one. Thus, a good choice of the data structure cause simple and efficient algorithms. In this section, we introduce two objects (red–black trees and maps) which have resulted to be very useful in the representation of a Poisson series.

3.1 Red–black trees

The binary tree is a very useful data structure for rapidly storing sorted data and rapidly retrieving saved data. A binary tree is composed of parent nodes, or leaves, each of which stores data and also links to up to two other child nodes (leaves), one of them placed to the left and the other one placed to the right. In this structure, the relationship between the leaves linked to and the linking leaf makes the binary tree an efficient data structure: the leaf on the left has a lesser key value, and the leaf on the right has an equal or larger key value.

A special type of tree is the red–black tree. In a red–black tree, each node has a color attribute, the value of which is either red or black. In addition to the ordinary requirements imposed on binary search trees, the following additional requirements of any valid red–black tree apply:

1. A node is either red or black.
2. The root is black.
3. All leaves are black, even when the parent is black.
4. Both children of every red node are black.
5. Every simple path from a node to a descendant leaf contains the same number of black nodes.

A critical property of red–black trees is enforced by these constraints: the longest path from the root to a leaf is no more than twice as long as the shortest path from the root to a leaf in that tree. The result is that the tree is roughly balanced. Since operations such as inserting,

deleting, and finding values requires worst case time proportional to the height of the tree, this fact makes the red–black tree be an efficient data structure. For instance, the search–time results to be $O(\log n)$. As we will discuss later, the use of this structure reduces significantly the complexity of the algorithms for addition and multiplication.

As mentioned above, in a binary tree, each node stores a key value (which must be unique in our case) and some associated data. For every node in the tree, all keys in the left subtree are smaller than the key of the node, and all keys in the right subtree are larger than the key of the node. So, each node is comprised of a key, a value, and a reference to the left (smaller keys) and right (larger keys) subtrees. This means that the key is the way to introduce the lexicographical order in the tree structure.

3.2 Maps

A map is an indexed data structure, similar to a vector. However, maps differ from vectors in two important points:

1. In a map, the index values (key values) can be any ordered data type, that is, any data type for which a comparison operator can be defined can be used as a key.
2. A map is an ordered data structure, elements are maintained in sequence, the ordering being determined by key values.

4 Poisson series as computational objects

Now, we will consider the basic information which characterizes a Poisson series, as well as the data structure to store it in the computer. This must be done preserving the canonical representation we have chosen. Let us consider a term of a Poisson series,

$$\tau = C x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} T(i_{n+1}\phi_1 + \cdots + i_{n+m}\phi_m)\,.$$

The information associated to each term of a Poisson series is given by the following elements:

1. A real number $C \in \mathbb{R}$ for representing the coefficient of the term.
2. A set of n integers $i_1, \ldots, i_n \in \mathbb{Z}$ for representing the exponents of the polynomial part.
3. A set of m integers $i_{n+1}, \ldots, i_{n+m} \in \mathbb{Z}$ for representing the coefficients of the angular part.
4. An integer $t \in \mathbb{Z}$ ($t = 0$ if $T = \cos$ and $t = 1$ if $T = \sin$).

A Poisson series is computationally considered as a hierarchic structure ordered by a key. This means that the adequate object to store a Poisson series is a map. Each term of the Poisson series corresponds to a node in the map structure. The data associated to each node of the map is a real number representing the coefficient of the corresponding term (C), and the key of each node is given by the set $(i_1, \ldots, i_n, i_{n+1}, \ldots, i_{n+m}, t)$.

The most common option for the storage of a Poisson series is the linked list, where each node contains the main characteristics of a separate term of the series. However, the most of the operations involving a series are based on navigating and searching through the structure that represents the series. In order to minimize the searching, deleting and inserting times of a term in a Poisson series, we have adopted the red–black tree as the structure to store a series in the computer. As we have already mentioned above, inserting, deleting, and finding values requires worst case time proportional to the height of the tree. Thus, we will represent a Poisson series as a red–black tree.

Moreover, as pointed out in (San–Juan and Abad, 2011), Poisson series involved in problems of celestial mechanics present small values for indices $i_1, \ldots, i_{n+m}$, normally in the interval $[-127, 128]$. On the other hand, the most of the integers $i_1, \ldots, i_{n+m}$ are zero. These two considerations should be taken into account in the way the structure for the representation of the key of each node, and the series itself, is coded.

If we store the key of a term in a vector structure, and assuming n and m polynomial and angular variables respectively, the complexity of the comparison of the keys is $O(n + m)$. We can reduce this complexity by storing keys in red–black trees. For each term of a Poisson series, we store

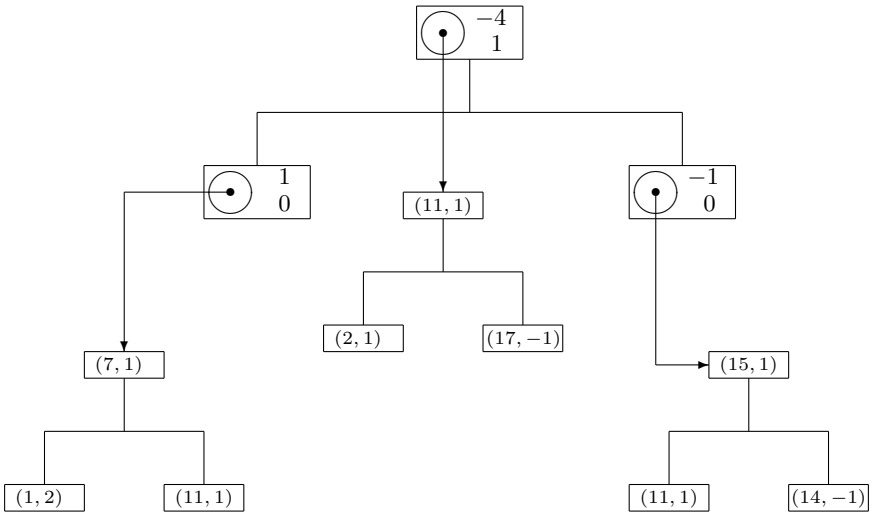

Figure 1: Red–black tree for representing a Poisson series P belonging to $\mathcal{P}_{10,12}$, being $P = x_1^2 x_7 \cos\phi_1 - 4x_2 \sin(\phi_1 - \phi_7) - \cos(\phi_1 + \phi_5 - \phi_4)$.

pairs (ν, i_ν) and (ν, j_ν). Distinction between angular and polynomial variables can be established taking polynomial variables for values of the index between 1 and n, and angular variables for values of the index between $n+1$ and $n+m$. Thus, the complexity of comparison between terms is reduced from $O(n+m)$ to $O(\log_2(n+m))$ in the worst case scenario.

If the keys associated to two different terms belonging to different Poisson series have different size, that means that both terms are not equal and can not be collected. This fact helps also to reduce the computation time. Moreover, it is not necessary to compare the entire key in case one index fails.

Thus, from a computational point of view, a Poisson series will be represented by a red–black tree with keys stored in red–black trees. In Figure 1, we show the representation of Poisson series

$$P = x_1^2 x_7 \cos\phi_1 - 4x_2 \sin(\phi_1 - \phi_7) - \cos(\phi_1 + \phi_5 - \phi_4)\,,$$

just to clarify the way red–black trees are used to store a Poisson series.

In the following section, we will anaylise the complexity of the most basic algorithms to be implemented in a Poisson series processor.

5 Basic manipulation of Poisson series

5.1 Addition and substraction of Poisson series

Let us consider two Poisson P and Q series containing N terms each of them. The implementation of this algorithm in a list structure requires: concatenating both lists ($O(1)$), sorting the resulting structure ($O(N \log_2 N)$), and collecting like terms ($O(N)$). The Quicksort algorithm has an average complexity of $O(N \log_2 N)$, but in the worst case scenario, the complexity is $O(N^2)$, and this case happens when the initial list is already ordered. When adding two Poisson series, both lists are initially sorted after being concatenated. Thus, the resulting list is quasi–sorted, and the complexity of the sorting algorithm results to be closer to $O(N^2)$ than to $O(N \log_2 N)$.

If both series are stored in a red–black tree, addition (or substraction) of Poisson series implies insertion of each term of Q in P: insertion when the key of the term is not contained in Q, and modification if the key of the term is contained in Q. In both cases, the complexity is $O(\log_2 N)$ for each term. Thus, complexity is $O(N \log_2 N)$. This algorithm provides the best case when all terms of Q appear also in P, with a complexity of $O(N \log_2 N)$. In the worst case scenario, the complexity is $\sum_{i=1}^{N} \log_2(N+i) = O(N \log_2(N))$. This occurs when all terms of Q must be inserted in P as new elements.

5.2 Multiplication of Poisson series

Let us consider two Poisson series P and Q with N_P and N_Q terms respectively. The multiplication of these two series can adopt the form of a Poisson series taking into account the following relations:

$$2\cos\lambda\cos\mu = \cos(\lambda+\mu)+\cos(\lambda-\mu)\,, \qquad 2\sin\lambda\sin\mu = \cos(\lambda-\mu)-\cos(\lambda+\mu)\,,$$
$$2\sin\lambda\cos\mu = \sin(\lambda+\mu)+\sin(\lambda-\mu)\,, \qquad 2\cos\lambda\sin\mu = \sin(\lambda+\mu)-\sin(\lambda-\mu)\,,$$

which can be applied when $\lambda = i_{n+1}\phi_1 + \cdots + i_{n+m}\phi_m$ and $\mu = i'_{n+1}\phi_1 + \cdots + i'_{n+m}\phi_m$. The implementation of this algorithm with a list structure has a complexity of $O(N^2M^2(n+m))$. For each term of series P, we have to visit each term in Q and then, compute and insert the resulting terms in the product series.

If we use a structure base on red–black trees, the cost of the insertion is then $\log_2(NM)$ instead of NM. So, the total complexity is reduced to $O(NM\log_2(NM)\log_2(n+m))$.

6 Conclusions

We have analysed how the adoption of a red–black tree structure to store Poisson series can reduce the computational cost of basic algorithms, as addition and multiplication of Poisson series. This structure leads to best computational times because basic operations like searching, insertion and deletion have logarithmic cost.

References

[1] Abad, A., San-Juan, J. F. PSPC, A Poisson series processor coded in C. *Dynamics and Astrometry of Natural and Artificial Celestial Bodies*, Poz'nan, Poland, pp. 383-389.

[2] Broucke, R. How to assemble a Keplerian processor, *Celest. Mech.*, 2, 9–20 (1970)

[3] Henrard, J. Algebraic manipulation on computers for lunar and planetary theories, *Proceedings of the IAU Symposium*, 114, Reidel Kovalevsky, J. and Brumberg, V. (eds.), 59–62 (1986)

[4] Jefferys, W. H. A Fortran–based list processor for Poisson series, *Celest. Mech.*, 2, 474–480 (1970)

[5] Kinoshita, H. Theory of the rotation of the rigid Earth, *Celest. Mech.*, 15, 277–326 (1977)

[6] Moons, M. Averaging approaches, *Proceedings of the "Artificial Satellite Theory Workshop"*, USNO, Washington, DC, 201 (1993)

[7] Richardson, D. L. PARSEC: An interactive Poisson series processor for personal computing systems, *Celest. Mech. Dyn. Astron.*, 45, 267–274 (1989)

[8] Rom, A. Mechanized algebraic operations (MAO), *Celest. Mech.*, 1, 301–319 (1969)

[9] San–Juan, F. and Abad, A. Algebraic and Symbolic Manipulation of Poisson Series, *J. Symbolic Computation*, 32, 565–572 (2001)