J. Symbolic Computation (2001) **32**, 565–572
doi:10.1006/jsco.2000.0396
Available online at http://www.idealibrary.com on IDEAL®

Algebraic and Symbolic Manipulation of Poisson Series

FÉLIX SAN-JUAN[†] AND ALBERTO ABAD

Grupo de Mecánica Espacial, Universidad de La Rioja, 26004 Logroño, Spain

Efficiency in handling Poisson series is essential to obtain high-accuracy analytical theories in celestial mechanics and non-linear dynamics in general. A good knowledge of the mathematical structure of these objects is fundamental to create data structures to store and handle efficiently its equivalent computational object. In this paper we analyse the mathematical, symbolic and computational structure of Poisson series.

1. Introduction

High accuracy is one of the main requirements for theories of perturbations applied to non-linear mechanics or non-linear differential equations problems. In these problems expressions based on multivariate Fourier series, whose coefficients are multivariate Laurent series

$$\sum_{i_0,\ldots,i_{n-1},j_0,\ldots,j_{m-1}} C^{j_0,\ldots,j_{m-1}}_{i_0,\ldots,i_{n-1}} x_0^{i_0}\ldots x_{n-1}^{i_{n-1}} \binom{\sin}{\cos}(j_0y_0+\cdots+j_{m-1}y_{m-1}),$$

commonly named Poisson series (Deprit *et al.*, 1965; Deprit, 1981), are among the most characteristic mathematical objects. Indeed, they appear in quite involved problems such as lunar and planetary motion, in artificial satellite theories, etc.

After Herget and Musen (1959), most of the people working in these scientific areas needed to handle Poisson series by computer. The software to manipulate algebraically Poisson series was named the Poisson series processor (PSP). There exists a great variety of PSPs, not only because of the language in which they were written (PL1, FORTRAN, LISP, C), but for the computational data structures they use. Among the most important PSP we find: Broucke's PSP (Broucke and Garthwaite, 1969), MAO (Rom, 1970), TRIGMAN (Jefferys, 1970, 1972), Dasenbrock's PSP (Dasenbrock, 1982), MAO!! (Miller, 1988) and our PSPC (Abad and San-Juan, 1993; San-Juan, 1996).

When the requirements of the analytical theory increase, PSPs become more and more efficient compared with general symbolic processors such as *Mathematica*, Maple, Macsyma, etc. In a previous paper (Abad and San-Juan, 1997) we compared PSPs with respect to general symbolic processors and we described PSPCLink, a tool to be added to *Mathematica* in order to have a new kernel combining the efficiency of PSPC with the flexibility of *Mathematica*, that sends expressions that involve Poisson series to PSPCLink and the remaining ones to the kernel of *Mathematica*.

[†]E-mail: `juanfelix.sanjuan@dmc.unirioja.es`

In order to achieve better accuracies in the applications of analytical theories, it is necessary to reach high orders in these theories; this fact demands a continuous maintenance and revision of PSPs. Unfortunately, there is not a standard PSP. In fact, usually each user writes and uses its own PSP. Since the desired standard PSP does not exist, this scenario will continue and a better knowledge of the Poisson series will help new users without previous experience. The objective of this paper is to clarify the characteristics of the mathematical object called Poisson Series, in order to help in the creation of software tools to handle it.

In this respect, good algorithms adapted to a good definition of the data structures are fundamental to the efficiency of a PSP. In this paper, we analyse the mathematical structure of the Poisson series, but this analysis may be useful not only to current PSP users, but to people who are not specialists in computer algebra and need to work with special algebraic manipulators to handle Poisson series, as well as other similar mathematical objects. The ideas contained in this paper can help them to characterize the Poisson series in order to create efficient software tools.

In particular, we show here the data structures and part of the algorithms implemented by us in our PSPC. The software package PSPC can be obtained by sending an e-mail to the authors.

2. Representation of Mathematical Objects in a Computer

To represent a mathematical object with the aim of its symbolic treatment in a computer, three levels of abstraction need to be distinguished.

The first level is *the mathematical object* level. This is the pure mathematical level where the elements belong to a set, in which a number of operations have been defined, which gives an algebraic structure.

The second level is *the symbolic object*. At this level, the objects are considered as a sequence of characters or symbols. It is the usual way in which the objects are represented formally by the mathematician. One object may have different representations. For example, the polynomial $p(x) = x^2 + x - 2$ can be represented as:

$$\begin{aligned} p_1(x) &= x^2 + x - 2, & p_2(x) &= -2 + x + x^2, \\ p_3(x) &= (x+2)(x-1), & p_4(x) &= -2 + x(1+x). \end{aligned}$$

The choice of one representation or another depends on the context of the problem and on the goals wished to be reached in the manipulation of the object.

Finally, the *computational object* level is based on the way the computer memory is organized. At this level, the computer stores the chosen symbolic representation of the mathematical object using data structures (stacks, lists, trees, etc.).

As seen above, an algebraic expression or mathematical object can have different symbolic representations that are equivalents. This has created one of the greatest problems of computational algebra: *the simplification* (Moses, 1971). The term *simplify* itself induces confusion due to its ambiguity. In fact, why is $p_1(x)$ simpler than $p_2(x)$, $p_3(x)$ or $p_4(x)$? The simplification problem can be formulated in a more precise way by the concepts of *normal* and *canonical functions* (Geddes *et al.*, 1992).

Let E be a set of expressions (symbolic objects) and $\sim$ be an equivalent relation in E defined as follows:

$$a \sim b \qquad \text{if} \qquad a = b \quad \text{with} \quad a, b \in \mathrm{E}, \tag{2.1}$$

where the $=$ operator is considered as the equality on the mathematical object level. For example, $(x+1)^2 \sim x^2 + 2x + 1$ since $(x+1)^2 = x^2 + 2x + 1$. We denote $E/\sim$ as the quotient set.

On the other hand, a will be identical to b, represented by $a \equiv b$, if a and b are identical as symbolic objects. In the previous example $(x+1)^2 \not\equiv x^2 + 2x + 1$ since they are two different character chains.

Thus, we say "to simplify" when referring to looking for a transformation $f : \mathrm{E} \to \mathrm{E}$ such that for all $a \in \mathrm{E}$ the transformed expression corresponds to the same equivalence class as that of a in $E/\sim$.

Definition 2.1. Let E be a set of expressions and $\sim$ be an equivalence relation over E. We say that a function $f : \mathrm{E} \to \mathrm{E}$ is a *normal function* in $(\mathrm{E}, \sim)$ if for all $a \in \mathrm{E}$, $f(a) \sim a$.

Definition 2.2. A function $f : \mathrm{E} \to \mathrm{E}$ is called *canonical* in $(\mathrm{E}, \sim)$ if it is *normal* in $(\mathrm{E}, \sim)$ and moreover, for all a, $b \in \mathrm{E}$ such that $a \sim b \Rightarrow f(a) \equiv f(b)$.

As shown before, a canonical function provides not only symbolically equivalent objects but also identical objects. Thus, a canonical function assigns a unique sequence of characters or symbols to each equivalence class, and this canonical representative will be chosen as the symbolic representation of the object.

3. Poisson Series as Mathematical Objects

On the first level, i.e. considering the series as mathematical object, we have to analyse its algebraic properties. In particular, from the computational point of view it is worth while checking what operations are closed over the set of Poisson series. Operations that do not obey this property will be difficult to implement without increasing the number of mathematical objects to manipulate in the program, and therefore increasing its complexity.

Definition 3.1. We call Poisson series of n polynomials $\boldsymbol{x} = (x_1, \ldots, x_n)$ and m angular variables $\boldsymbol{y} = (y_1, \ldots, y_m)$, to a map $(\boldsymbol{x}, \boldsymbol{y}) \to \mathcal{S}(\boldsymbol{x}, \boldsymbol{y}) : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, defined by

$$\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{i} \in \mathcal{I}, \boldsymbol{j} \in \mathcal{J}} C_{\boldsymbol{i}}^{\boldsymbol{j}} P_{\boldsymbol{i}} T_{\boldsymbol{j}}, \qquad C_{\boldsymbol{i}}^{\boldsymbol{j}} \in \mathbf{R},$$

where $\mathrm{i} = (i_0, \ldots, i_{n-1})$ and $\mathrm{j} = (j_0, \ldots, j_{m-1})$ are elements belonging to $\mathbf{Z}^n$ and $\mathbf{Z}^m$, respectively, and furthermore

$$P_{\boldsymbol{i}} = x_0^{i_0} \ldots x_{n-1}^{i_{n-1}} \qquad \text{and} \qquad T_{\boldsymbol{j}} = \begin{pmatrix} \sin \\ \cos \end{pmatrix} (j_0 y_0 + \cdots + j_{m-1} y_{m-1}),$$

represent the polynomial and trigonometric parts, respectively.

Let $\mathcal{P}_{n,m}$ be the set of the Poisson series above defined. Thus, we can define the sum of two Poisson series and prove that $(\mathcal{P}_{n,m}, +)$ forms a commutative group whose zero element is the series $0x_1^0 \dots x_n^0 \cos(0y_1 + \cdots + 0y_n)$ and the inverse element of a given series is the series itself with their coefficients $C_{\boldsymbol{i}}^{\boldsymbol{j}}$ changed in signs.

On the other hand, taking into account the trigonometrical relations, we can define the product of two Poisson series as an internal, associative, and commutative operation with the unit element.

On the other hand, we can define an external operation with coefficient field $\mathbf{R}$, $(t, \mathcal{S}(\boldsymbol{x}, \boldsymbol{y})) \to t\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}) : \mathbf{R} \times \mathcal{P}_{n,m} \to \mathcal{P}_{n,m}$, such that

$$t\mathcal{S}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{i} \in \mathcal{I}, \boldsymbol{j} \in \mathcal{J}} (tC_{\boldsymbol{i}}^{\boldsymbol{j}}) P_{\boldsymbol{i}} T_{\boldsymbol{j}}.$$

The $\mathcal{P}_{n,m}$ set with the two operations defined, $+$ and $\cdot_{\mathbf{R}}$, is a vectorial space that together with the ring structure carry out $\mathcal{P}_{n,m}$ a commutative algebra.

Note that the previous definitions are also valid when the external coefficient field is the set of the complex numbers $\mathbf{C}$.

The algebraic structure of the Poisson series determines the mathematical framework within which we shall work. However, the type of problems we will deal with by using these series requires study of another type of operation that is related to the differential and integral calculus.

It is not difficult to prove that the partial derivative of a Poisson series with respect to a polynomial or an angular variable is always another Poisson series. Besides, the total derivative $d\mathcal{S}(\boldsymbol{x}, \boldsymbol{y})/dt$ of a Poisson series with respect to an independent variable t, is a Poisson series if the derivatives of each variable x_i and y_j with respect to t can be expressed as Poisson series.

The integral of a Poisson series with respect to a polynomial variable is another Poisson series provided that there is no term whose exponent is -1. In this case, a term with a $\ln(x)$ factor would be obtained and therefore the integral would not be a Poisson series.

4. Poisson Series as Symbolic Objects

Let $\mathcal{P}_{n,m}$ be the set of Poisson series with n polynomial and m angular variables, and let $\sim$ be the equivalence relation defined in (2.1). We are looking for canonical functions that will provide a canonical representation of each equivalence class, that is, the symbolic object that represents the Poisson series.

As in the polynomial case, the symbolic representation of a non-performed operation can be used as a representation of the resultant object. Thus, for example, $(x-1)(x+1)$ represents $x^2 - 1$. From now on, this type of representation will not be considered, and all the operations will be assumed to be performed in such a way that for the Poisson series we will always start with a representative of the form

$$\sum_{\boldsymbol{i},\boldsymbol{j}} C_{\boldsymbol{i}}^{\boldsymbol{j}} P_{\boldsymbol{i}} T_{\boldsymbol{j}}, \tag{4.1}$$

where the summations appear without any predetermined order.

To obtain a canonical representative, it is necessary to make the following operations.

(1) If the first non-zero coefficient of the angular variables is negative, the following rules will be applied:

$$\sin(-a_{\boldsymbol{k}} y_{\boldsymbol{k}} + \cdots) = -\sin(a_{\boldsymbol{k}} y_{\boldsymbol{k}} - \cdots),$$
$$\cos(-a_{\boldsymbol{k}} y_{\boldsymbol{k}} + \cdots) = \cos(a_{\boldsymbol{k}} y_{\boldsymbol{k}} - \cdots).$$

The duplicity of symbolically different but mathematically equal trigonometrical terms is avoided with this operation.

(2) The terms with identical polynomial and trigonometric parts will be grouped together.

The classical way to work with Poisson series is to define an order relation, which takes into account simultaneously the polynomial and the trigonometric parts of the series. By doing so, a canonical symbolic representation of the form (4.1) is obtained. However, we have chosen a different representation in which the polynomial and the trigonometric terms are ordered separately. This representation agrees better with the computational one chosen for the Poisson series. In order to do that let us follow the next steps.

(3) The terms of the set $\{P_{\boldsymbol{i}}, \mathrm{i} \in \mathcal{I}\}$ will be ordered with the lexicographical ordering defined as follows. Let $\mathrm{i}^{(1)}$, $\mathrm{i}^{(2)} \in \mathcal{I}$ we say that $\mathrm{i}^{(1)} \prec \mathrm{i}^{(2)}$ if for the first k-index such that $i_k^{(1)} \neq i_k^{(2)}$ then $i_k^{(1)} < i_k^{(2)}$ is verified.

By i we denote the natural number that represents the $\boldsymbol{i}$ index order after performing the lexicographical ordering described before. The set of polynomial terms can be now represented by $\{P_i, i = 1, \ldots, p\}$, with p equal to the number of polynomial terms.

(4) The terms of the set $\{T_{\boldsymbol{j}}, \mathrm{j} \in \mathcal{J}\}$ will be ordered following the lexicographical ordering defined as follows. Let $\mathrm{j}^{(1)}$, $\mathrm{j}^{(2)} \in \mathcal{J}$ we say that $\mathrm{j}^{(1)} \prec \mathrm{j}^{(2)}$ if for the first k-index such that $j_k^{(1)} \neq j_k^{(2)}$ then $j_k^{(1)} < j_k^{(2)}$ is verified, or if for all k, $j_k^{(1)} = j_k^{(2)}$ and $T_{\boldsymbol{j}^{(1)}}$ represents a cosine and $T_{\boldsymbol{j}^{(2)}}$ a sine.

By j we denote the natural number that represents the $\boldsymbol{j}$ index order once the previous lexicographical ordering is performed. The set of trigonometric terms can be now represented by $\{T_j, j = 1, \ldots, t\}$, with t equal to the number of trigonometric terms.

By calling P the row vector of dimension p whose elements are the ordered terms P_i, T the column vector with dimension t whose elements are the ordered terms T_j, and C the matrix of coefficients C_i^j, the series can be written in the following matrix form

$$S(\boldsymbol{x}, \boldsymbol{y}) = P \cdot C \cdot T. \tag{4.2}$$

This is the best canonical representative from the computational point of view, but not from the point of view of a user that has to read and write series. Owing to this, we will define a *more convenient* representation obtained by applying a new operation to the previous representation.

(5) The terms with a common trigonometric part will be grouped together

$$S(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{t} \left(\sum_{i=1}^{p} C_i^j P_i \right) T_j. \tag{4.3}$$

This will be the canonical representation used for the PSPC input and output functions, that is, for the user–computer communications.

5. Poisson Series as a Computational Object

In this section we will consider the basic information that characterizes a Poisson series and the data structures through which the series is stored in the computer, in such a way that the chosen canonical representation for the given object is reflected.

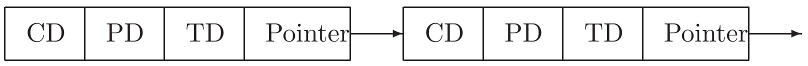

Figure 1. Unidimensional structure.

Each term of a Poisson series is characterized by the following basic information.

(i) A number C_i^j that represents the coefficients.
Using rational arithmetic, with multiple precision numbers, is preferable from an analytical point of view. However, when we want to numerically evaluate the series, floating point arithmetic is sufficient. A data structure CD stores rational or real coefficients.

(ii) n integers $(i_0, \ldots, i_{n-1})$ that represent the exponents of the polynomial part P_i.
This information is contained in a data structure PD.

(iii) m integers $(j_0, \ldots, j_{m-1})$ that represent the angular variable coefficients of the trigonometrical part T_j, plus an integer, or simply a bit, equal to 0 or 1 which says if the term is a sine or a cosine. The data structure TD contains all the information about the trigonometric part.

To minimize the size of the data structures that contain the terms of the series we make use of two properties of the Poisson series when they appear in problems of non-linear dynamics. Since the integers $(i_0, \ldots, i_{n-1})$ and $(j_0, \ldots, j_{m-1})$ are usually small numbers, in practice we consider these numbers inside the interval $[-127, 128]$. In each term most of the integers $(i_0, \ldots, i_{n-1})$, $(j_0, \ldots, j_{m-1})$ are zero. We save a great amount of memory with data structures PD, TD that store the integers packed in groups of four and avoid the storage of zeros.

Throughout PSP evolution, the chosen way to store information on Poisson series and data structures has been affected by the computer limitations as well as the languages used. The designs used in the main PSPs, can be found in the Henrard (1989) and Laskar (1989) reviews. Abad and Sein-Echaluce (1988) analysed the solutions given by Broucke and Garthwaite (1969) and Dasenbrock (1982).

The Dasenbrock work is the direct predecessor of PSPC. His PSP is built using unidimensional, unidirectional and ordered term lists.

Dasenbrock also pointed out in his work the limitations of this structure, due mainly to the repetition of polynomial and trigonometric terms and suggested that a bidimensional structure would give better results.

Although a bidimensional structure is more difficult to be implemented, we have chosen it as the storage system since it ensures a faster access to each term. The data that determine the series are stored in three different types of nodes: polynomial (PN), trigonometrical (TN), and coefficient nodes (CN). These nodes contain the data (CD, PD, TD) and the necessary pointers. The graphic representation of this structure is shown in Figure 2.

The trigonometric nodes shown in the lower horizontal line form a bidimensional list, ordered according to the lexicographical ordering described in the latter part. The polynomial nodes appearing on the left vertical line also form an ordered bidirectional list. The coefficient nodes form horizontal bidirectional lists associated to a polynomial node, and vertical bidirectional lists associated to a trigonometrical node. A structure stored

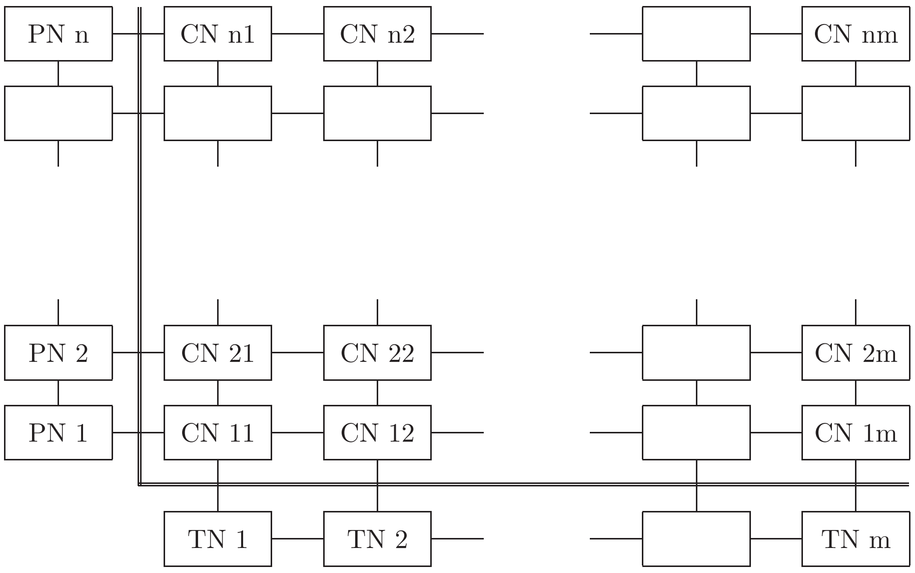

Figure 2. Bidimensional structure.

in such a way reflects reliably the symbolic representation (4.2). The representation of the coefficients through bidimensional lists, instead of a matrix, enables the building of a sparse representation in which the zeros are not stored. This represents a remarkable saving in memory when the series has a lot of terms.

This type of structure also allows us to retrieve the two canonical representation aforementioned. First, the structure reflects representation (4.1) since each coefficient node is linked to a polynomial and trigonometrical node. The three nodes constitute a term of the series. Secondly, representation (4.3) is obtained by considering each node of the list of trigonometrical terms and, obtaining the polynomial that acts as the coefficient of this node by multiplying the column of coefficients associated to this trigonometrical node with the corresponding list of polynomial nodes.

We find two basic operations when handling Poisson series. (1) Navigate through the different lists of the bidimensional structure applying or not functions to each node of the list, and (2) search the position of a term into a series.

Most of the operations involving Poisson series can be reduced applying the two previous operations. For instance, to add two series is equivalent to inserting each term of one of the series into the other one, and insertion is essentially based on the searching algorithm. The partial derivative of a series with respect to a polynomial variable corresponds to going through the list of polynomial nodes and extracting the exponent of the variable, subtracting a unit from this exponent and going to the horizontal list of coefficient nodes multiplying each coefficient by the exponent.

We have very carefully looked for a good searching algorithm. The best one, in our experience, is binary in nature; unfortunately, it cannot be applied to lists because lists are not indexed arrays. However, the use of the bidirectional lists, together with the knowledge of the number of terms of the list, permits navigating forward and backward and emulates the binary search, improving the performance of this critical operation.

Among the problems we had to face was the exchange of information with other computer programs. We solved the most pressing ones by creating functions (routines) for input and output in various formats. In particular, PSPC makes it possible to convert Poisson series to LaTeX and *Mathematica* output. Another important capability is to create automatically, from a series, a C or FORTRAN program to evaluate it. The navigation algorithm of Coffey and Deprit (1980) has been included in the evaluation code in order to obtain fast evaluations of very big series.

Acknowledgements

We are very grateful to J. C. Agnesse for reporting bugs, testing routines, etc. A. Elipe for his valuable suggestions, and I. Tijera for translating the manuscript. This work has been supported in part by the Ministerio de Educación y Ciencia (DGES Project # PB98-1576), the University of La Rioja (Project API-98/A11) and the Department of Space Mathematics of Centre National d'Etudes Spatiales (France).

References

Abad, A., San-Juan, J. F. (1993). PSPC. A Poisson series processor coded in C. In *Dynamics and Astrometry of Natural and Artificial Celestial Bodies, Poznan, Poland*, pp. 383–389.

Abad, A., San-Juan, J. F. (1997). PSPCLink: A cooperation between general symbolic and Poisson series processors. *J. Symb. Comput.*, **24**, 113–122.

Abad, A., Sein-Echaluce, M. L. (1988). Manipuladores algebráicos en Mecánica Celeste. *Rev. Acad. Cienc. Zaragoza*, **43**, 117–127.

Broucke, R., Garthwaite, K. (1969). A programing system for analytical series expansions on a computer. *Celest. Mech.*, **1**, 271–284.

Coffey, S. L., Deprit, A. (1980). Fast evaluation of Fourier series. *Astron. Astrophys.*, **81**, 310–315.

Dasenbrock, R. R. (1982). A FORTRAN-based program for computerized algebraic manipulation. Technical Report, National Research Laboratory Report 8611.

Deprit, A. (1981). Celestial mechanics: Never say no to a computer. *J. Guid. Control*, **4**, 577–581.

Deprit, A., Deprit, E. (1990). Processing Poisson series in parallel. *J. Symb. Comput.*, **10**, 179–201.

Deprit, A. *et al.* (1965). The symbolic manipulation of Poisson series. Technical Report. Boeing Scientific Research Laboratories Document, Seattle, WA.

Geddes, K. O. *et al.* (1992). *Algorithms for Computer Algebra.* Massachusetts, Kluwer Academic Publishers.

Henrard, J. (1989). A survey of Poisson series processors. *Celest. Mech.*, **45**, 245–253.

Herget, P., Musen, P. (1959). The calculation of literal expansions. *Astron. J.*, **64**, 11–20.

Jefferys, W. H. (1970). A FORTRAN-based list processor for poisson series. *Celest. Mech.*, **2**, 474–480.

Jefferys, W. H. (1972). A precompiler for the formula manipulation system trigman. *Celest. Mech.*, **6**, 117–124.

Laskar, J (1989). In Manipulation des séries*, Editions Frontières, Gif-sur-Yvette*, pp. 90–106.

Miller, B. R. (1988). *MAO!!* Private communication.

Moses, J. (1971). Algebraic simplification: A guide for the perplexed. *Commun. ACM*, **14**, 527–537.

Rom, A. (1970). Mechanized algebraic operations (MAO). *Celest. Mech.*, **1**, 301–319.

San-Juan, J. F. (1996). Manipulación algebraica de series de Poisson. Aplicación a la teoría del satélite artificial. Ph.D. Thesis, Universidad de Zaragoza.

Originally Received 9 April 2000
Accepted 9 June 2000