Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2012, Article ID 201340, 24 pages
doi:10.1155/2012/201340

Research Article

The Asymptotic Expansion Method via Symbolic Computation

Juan F. Navarro

Departamento de Matemática Aplicada, Universidad de Alicante, Carretera San Vicente del Raspeig s/n, 03690 San Vicente del Raspeig, Alicante, Spain

Correspondence should be addressed to Juan F. Navarro, jf.navarro@ua.es

Received 26 October 2011; Revised 25 March 2012; Accepted 26 March 2012

Academic Editor: B. V. Rathish Kumar

This paper describes an algorithm for implementing a perturbation method based on an asymptotic expansion of the solution to a second-order differential equation. We also introduce a new symbolic computation system which works with the so-called modified quasipolynomials, as well as an implementation of the algorithm on it.

1. Introduction

The origin of symbolic manipulation derives from the sheer magnitude of the work involved in the building of perturbation theories, which made inevitable that scientific community became interested in the possibility of constructing those theories with the help of computers.

Perturbation theories for differential equations containing a small parameter ϵ are quite old. The small perturbation theory originated by Sir Isaac Newton has been highly developed by many others, and an extension of this theory to the asymptotic expansion, consisting of a power series expansion in the small parameter, was devised by Poincaré (1892) [1]. The main point is that for the most of the differential equations, it is not possible to obtain an exact solution. In cases where equations contain a small parameter, we can consider it as a perturbation parameter to obtain an asymptotic expansion of the solution. In practice, the work involved in the application of this approach to compute the solution to a differential equation cannot be performed by hand, and algebraic processors seem to be a very useful tool.

As explained in [2], the first symbolic processors were developed to work with Poisson series, that is, multivariate Fourier series whose coefficients are multivariate Laurent series,

$$\sum_{i_1,\ldots,i_n} \sum_{j_1,\ldots,j_m} C_{i_1,\ldots,i_n}^{j_1,\ldots,j_m} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \frac{\cos}{\sin}\left(j_1\phi_1 + \cdots + j_m\phi_m\right), \tag{1.1}$$

where $C_{i_1,\ldots,i_n}^{j_1,\ldots,j_m} \in \mathbb{R}$, $i_1,\ldots,i_n$, $j_1,\ldots,j_m \in \mathbb{Z}$, and $x_1,\ldots,x_n$ and $\phi_1,\ldots,\phi_m$ are called polynomial and angular variables, respectively. These processors were applied to problems in nonlinear mechanics or nonlinear differential equations problems, in the field of Celestial Mechanics.

One of the first applications of these processors was concerned with the theory of the Moon. Delaunay invented his perturbation method to treat it and spent 20 years doing algebraic manipulations by hand to apply it to the problem. Deprit et al. [3, 4] prolongated the solution of Delaunay's work with the help of a special purpose symbolic processor, and Henrard [5] pushed it to order 25. This solution was improved by iteration by Chapront-Touzé [6], and planetary perturbations were also introduced by Chapront-Touzé [7]. At present, the most complete solution, ELP (Ephemeride Lunaire Parisien) contains more than 50000 periodic terms.

But the motion of the Moon is not the only application of algebraic processors. There are many problems where the facilities provided by Poisson series processors can lead rather quickly to very accurate results. As examples, we would like to mention planetary theories, the theory of the rotation of the Earth (e.g., [8]), and artificial satellite theories (AST). Abad et al. [9] have analysed the convenience of developing specific computer algebra systems in order to deal with AST. As it is explained in detail in their work, the series involved in the computation of the solution through the application of a Lie transformation have a total amount of almost 2000000 terms.

Nowadays, many general purpose computer algebra packages—as, for example, `Maple, Mathematica, and Matlab`—contain tools for the calculation of the solution of certain classes of ODEs. All these packages have the advantage of being very general, so they can deal with a lot of problems of different nature. However, if one is interested in higher order solutions, the most common perturbation methods tend to produce expressions containing thousands of terms, and their treatment with those general processors becomes inefficient.

To achieve better accuracies in the applications of analytical theories, high orders of the approximate solution must be computed, making a continuous maintenance and revision of the existing symbolic manipulation systems necessary, as well as the development of new packages adapted to the peculiarities of the problem to be treated.

In order to contribute to the solution of this problem, we have developed a symbolic computation package (called `MQSP`) based on the object-oriented philosophy which manipulates objects of the form

$$\sum_{\nu\geq 0} \tau_1^{\sigma_1} \times \cdots \times \tau_Q^{\sigma_Q}\ t^{n_\nu} e^{\alpha_\nu t}\left(\lambda_\nu \cos(\omega_\nu t) + \mu_\nu \sin(\omega_\nu t)\right), \tag{1.2}$$

where $n_\nu \in \mathbb{N}$, $\alpha_\nu, \omega_\nu, \lambda_\nu, \mu_\nu \in \mathbb{R}$, $\sigma_\nu \in \mathbb{Z}$, and $\tau_1,\ldots,\tau_Q$ are real constants with unknown value. We will refer to those elements as modified quasipolynomials [10]. The kernel of this symbolic processor has been developed in C++. The operations on series implemented in the manipulator are the usual operations of the Algebra of quasipolynomials: addition, subtraction, multiplication, multiplication by a scalar, differentiation with respect to t, and substitution of a quasipolynomial into an undetermined coefficient.

We have also constructed a set of subroutines to deal with the solution to the perturbed differential equation

$$\ddot{x} + a_1\dot{x} + a_0 x = u(t) + \epsilon f(x, \dot{x}), \qquad x(t_0) = x_0, \qquad \dot{x}(t_0) = \dot{x}_0, \tag{1.3}$$

where $a_0, a_1, t_0, x_0, \dot{x}_0 \in \mathbb{R}$, $u(t)$ is given by (1.2), and

$$f(x, \dot{x}) = \sum_{\kappa=0}^{M} \sum_{0 \le \nu \le \kappa} f_{\nu,\kappa-\nu}\ x^{\nu}\ \dot{x}^{\kappa-\nu}, \qquad f_{\nu,\kappa-\nu} \in \mathbb{R}. \tag{1.4}$$

In a previous contribution, the author employed the kernel of this processor to compute periodic solutions in equations of type (1.3) via the Poincaré-Lindstedt method [11, 12]. If the unperturbed equation ($\epsilon = 0$) has periodic solutions and ϵ is a measure of the size of the perturbing terms, then the trajectories for the full system will remain pretty close to those of the nonperturbed system, for any finite period of time $t_0 < t < t_0 + \alpha$ ($\alpha > 0$) with an error not larger than $O(\alpha)$. In general, even a small perturbation is enough to destroy periodicity, that is, nonlinearity will end with most of the periodic orbits of the unperturbed system, but some of them may persist. To calculate those periodic orbits, the solution and the modified frequency are expanded with respect to the small parameter, allowing to kill secular terms which appear in the recursive scheme.

The aim of this paper is to construct an algorithm to implement the asymptotic expansion method [13]. This new implementation is general and does not depend on the function $f(x, \dot{x})$ as given in (1.4), that is, the user does not need to programme the algorithm described in this paper, and only has to introduce the adequate parameters when calling the corresponding routine of the package. The code of this specific symbolic system is not available on internet but it can be provided by contacting its author.

2. Data Structure

The algorithms that can be implemented to perform the basic manipulation on a series and their efficiency depend on the way a series is coded. An overcoded structure that makes good use of memory generally requires complex algorithms, which increase the computational cost in terms of time. On the other hand, an undercoded computational representation of the terms generates simple algorithms, because the location of all the coefficients can be obtained directly. However, this scheme presents the inconvenience of being very wasteful in the memory resources required for the storage of the series [2].

In this section, we follow San-Juan and Abad [14] to introduce the representation of a mathematical object in a computer. To that purpose, let us introduce the concepts of normal and canonical functions. Let E be a set of symbolic objects, and let $\sim$ be an equivalence relation in E, defined as follows: $a \sim b$ if $a = b$, with $a, b \in E$. Here, the operator $=$ is considered as the equality on the mathematical object level. Moreover, $a \equiv b$ if a and b are identical as symbolic objects. A function $f : E \to E$ is said to be normal in $(E, \sim)$ if $f(a) \sim a$ for all $a \in E$, and f is said to be canonical in $(E, \sim)$ if it is normal and $a \sim b \Rightarrow f(a) = f(b)$ for all $a, b \in E$. Thus,

a canonical function provides identical objects when objects are equivalent, that is, when they represent the same mathematical object.

For the sake of simplicity, let us focus our attention on the set of quasipolynomials in the independent variable $Q[t]$. A quasipolynomial is a map $u : \mathbb{R} \to \mathbb{R}$, defined by

$$u(t) = \sum_{\nu \geq 0} t^{n_\nu} e^{\alpha_\nu t} \left(\lambda_\nu \cos(\omega_\nu t) + \mu_\nu \sin(\omega_\nu t)\right), \tag{2.1}$$

where $n_\nu \in \mathbb{N}$, α_ν, ω_ν, λ_ν, and $\mu_\nu \in \mathbb{R}$. Let us now consider the set of quasipolynomials in the independent variable t, $Q[t]$.

2.1. QuasiPolynomials as Symbolic Objects

We look for a canonical representation for each equivalence class defined in $Q[t]$. For that purpose, the following operations must be performed over each quasipolynomial.

(1) Let us consider a quasipolynomial $u(t) = \sum_{\nu \geq 0} t^{n_\nu} e^{\alpha_\nu t} (\lambda_\nu \cos(\omega_\nu t) + \mu_\nu \sin(\omega_\nu t))$. If $\omega_\nu < 0$, the following rules must be applied:

$$\sin(-\omega_\nu t) = -\sin(\omega_\nu t), \qquad \cos(-\omega_\nu t) = \cos(\omega_\nu t). \tag{2.2}$$

(2) The terms of a quasipolynomial will be ordered as follows: let us consider two term of a quasipolynomial: $\tau_1 = t^{n_1} e^{\alpha_1 t} (\lambda_1 \cos \omega_1 t + \mu_2 \sin \omega_1 t)$ and $\tau_2 = t^{n_2} e^{\alpha_2 t} (\lambda_2 \cos \omega_2 t + \mu_2 \cos \omega_2 t)$. We say that $\tau_1 < \tau_2$ if $(n_1 < n_2)$ or $(n_1 = n_2$ and $\alpha_1 < \alpha_2)$ or $(n_1 = n_2$, $\alpha_1 = \alpha_2$, and $\omega_1 < \omega_2)$ or $(n_1 = n_2, \alpha_1 = \alpha_2, \omega_1 = \omega_2$, and $\lambda_1 < \lambda_2)$ or $(n_1 = n_2, \alpha_1 = \alpha_2$, $\omega_1 = \omega_2$, $\lambda_1 < \lambda_2$, and $\mu_1 < \mu_2)$.

(3) The terms of a quasipolynomial

$$u(t) = \sum_{\nu \geq 0} t^{n_\nu} e^{\alpha_\nu t} \left(\lambda_\nu \cos(\omega_\nu t) + \mu_\nu \sin(\omega_\nu t)\right) \tag{2.3}$$

with identical values of n_ν, α_ν, and ω_ν must be grouped together.

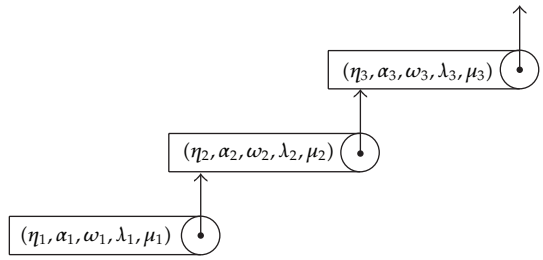
2.2. QuasiPolynomials as Computational Objects

In this section, we will consider the basic information which characterizes a quasipolynomial, as well as the data structure to store it in the computer. This must be done preserving the canonical representation we have chosen.

Each quasipolynomial is collected in a sorted dynamical list: a sorted list is one in which the order of the items is defined by some collating sequence. The codification of each term of the list contained in a quasipolynomial is statical, and given by the following elements.

(1) $\lambda, \mu \in \mathbb{R}$ are the coefficients of the term.

(2) $n \in \mathbb{N}$ is the degree of the monomial t.



Scheme 1

(3) $\alpha \in \mathbb{R}$ is the exponent of the exponential part of the term.

(4) $\omega \geq 0$ is the frequency of the periodic part.

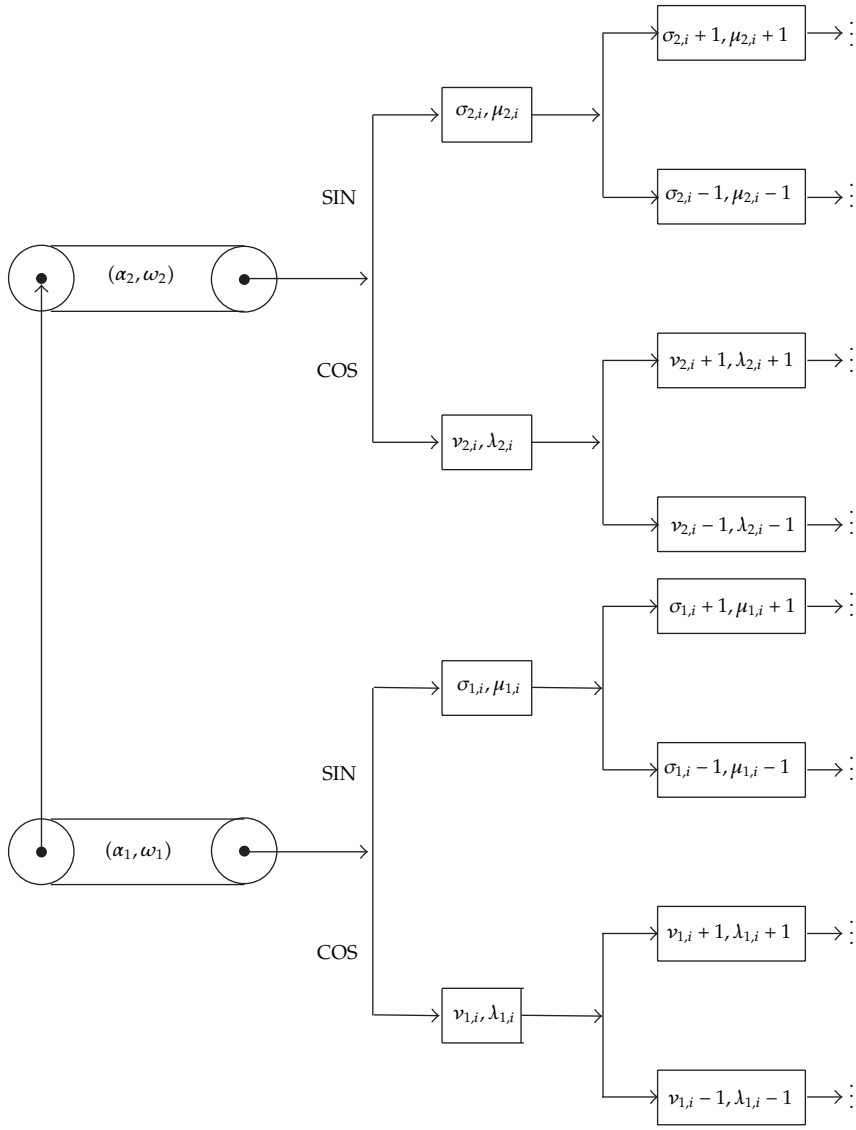
In Scheme 1 we represent the way a quasipolynomial is stored in memory by using a simple list.

As pointed out in [14], 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 quasipolynomials is equivalent to inserting each term of one series into the other one. So, a good choice of the data structure results in simple and efficient algorithms. The binary tree resulting seems to be 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), which can be visualized spatially as below the first node with one placed to the left and the other to the right. In this structure, the relationship between the linked leaves 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 greater 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: A node is either red or black. The root is black. All leaves are black, even when the parent is black. Both children of every red node are black. 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 efficient, for instance, the search-time results to be $O(\log n)$.

With the use of this structure, the complexity of the algorithms for addition, multiplication, derivation, and integration of quasipolynomials is significantly reduced. Unfortunately, this structure, which results to be ideal for Poisson series, cannot be applied directly in our case due to the fact that the numbers which identify each term of a quasipolynomial are not indexed arrays. However, an alternative aggrupation of terms in a quasipolynomial can be performed in order to introduce this balanced structure. To do that, let us express a quasipolynomial as follows:

$$u(t) = \sum_{\nu \geq 0} C_{p,\nu}(t) e^{\alpha_\nu t} \cos(\omega_\nu t) + S_{q,\nu}(t) e^{\alpha_\nu t} \sin(\omega_\nu t), \tag{2.4}$$

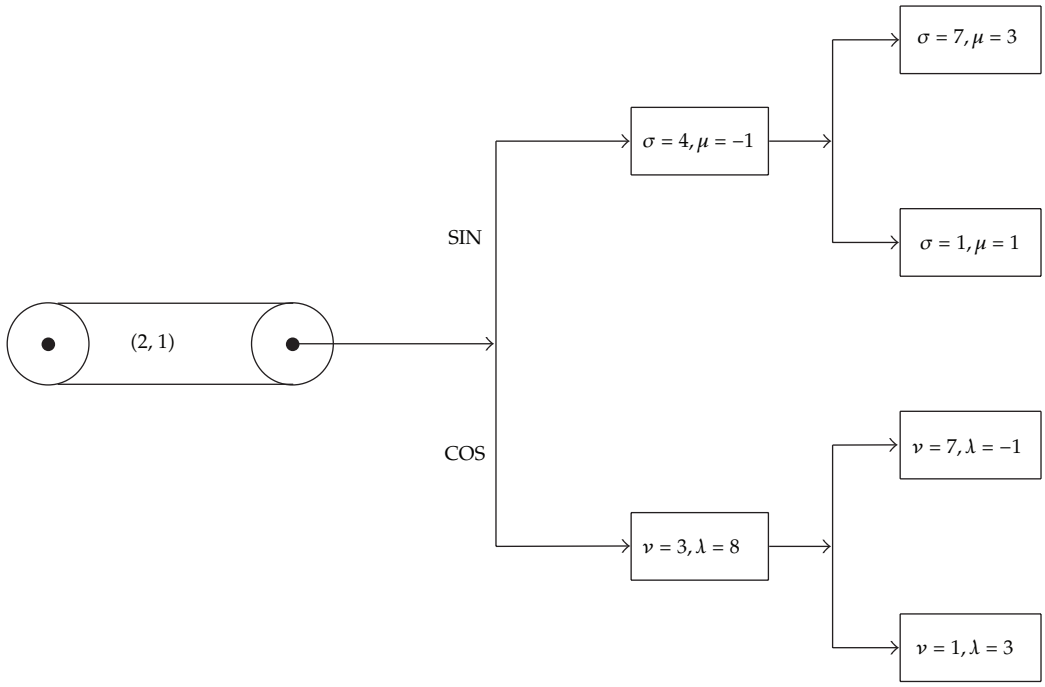


Scheme 2

where $C_{p,v}(t)$ and $S_{q,v}(t)$ are polynomials in the variable t with constant coefficients, of degree p and q respectively, being $p, q \in \mathbb{N}$:

$$\begin{aligned} C_{p,v}(t) &= \lambda_0 + \lambda_1 t + \lambda_2 t^2 + \cdots + \lambda_p t^p, \\ S_{q,v}(t) &= \mu_0 + \mu_1 t + \mu_2 t^2 + \cdots + \mu_q t^q, \end{aligned} \tag{2.5}$$

with $\lambda_0, \lambda_1, \ldots, \lambda_p,\ \mu_0, \mu_1, \ldots, \mu_q \in \mathbb{R}$. If we aggrupate terms of a quasipolynomial in such a manner, we can use a tree structure to store it, saving only significant terms. In Scheme 2 we show the tree structure in which the quasipolynomial is stored.



Scheme 3

In Scheme 2 we show how we store in memory the quasipolynomial

$$u(t) = C_1(t)e^{\alpha_1 t}\cos(\omega_1 t) + S_1(t)e^{\alpha_1 t}\sin(\omega_1 t) + C_2(t)e^{\alpha_2 t}\cos(\omega_2 t) + S_2(t)e^{\alpha_2 t}\sin(\omega_2 t), \tag{2.6}$$

where

$$\begin{aligned} C_1(t) &= \sum_{i\in\mathbb{I}_{C,1}} \lambda_{1,i}t^{\nu_{1,i}}, \qquad S_1(t) = \sum_{i\in\mathbb{I}_{S,1}} \mu_{1,\,i}t^{\sigma_{1,i}}, \\ C_2(t) &= \sum_{i\in\mathbb{I}_{C,2}} \lambda_{2,i}t^{\nu_{2,i}}, \qquad S_2(t) = \sum_{i\in\mathbb{I}_{S,2}} \mu_{2,i}t^{\sigma_{2,i}}, \end{aligned} \tag{2.7}$$

with $\mathbb{I}_{C,j}$ and $\mathbb{I}_{S,j}$ being sets of indices for any $j = 1, 2$, $\nu_{1,i} \geq 0$ and $\lambda_{1,i} \neq 0$ for any $i \in \mathbb{I}_{C,1}$, $\nu_{2,i} \geq 0$ and $\lambda_{2,i} \neq 0$ for any $i \in \mathbb{I}_{C,2}$, $\sigma_{1,i} \geq 0$ and $\mu_{1,i} \neq 0$ for any $i \in \mathbb{I}_{S,1}$, and $\sigma_{2,i} \geq 0$ and $\mu_{2,i} \neq 0$ for any $i \in \mathbb{I}_{S,2}$. Moreover, $\nu_{j,i-1} < \nu_{j,i} < \nu_{j,i+1}$ and $\sigma_{j,i-1} < \sigma_{j,i} < \sigma_{j,i+1}$ for any $j = 1, 2$ and i belonging to the corresponding set of indices.

To illustrate the way a quasipolynomial is stored by means of the tree structure described above, we show in Scheme 3 the structure associated to the quasipolynomial given by

$$u(t) = \left(3t + 8t^3 - t^7\right)e^{2t}\cos t + e^{2t}\sin t\left(t - t^4 + 3t^7\right). \tag{2.8}$$

Table 1: Some examples of representation of terms.

Term	λ	μ	σ_1	σ_2	σ_3	σ_4	n	α	ω
$3\tau_1^2\tau_4\cos(5t)$	3	0	2	0	0	1	0	0	5
$-\tau_1\tau_2\tau_3^2t^7e^{-t}\sin(3t)$	0	–1	1	1	2	0	7	–1	3
$2\tau_2\tau_3^2te^{2t}\cos(3t)$	2	0	0	1	2	0	1	2	3

2.3. *Modified QuasiPolynomials as Computational Objects*

In our symbolic system, we represent a modified quasipolynomial by an ordered and dynamical list of terms, keeping in memory only significant terms. The codification of a term is statical, and given by the following elements.

(1) $\lambda, \mu \in \mathbb{R}$ are the coefficients of the term (for cos and sin, resp.).

(2) $(\sigma_1, \ldots, \sigma_Q) \in \mathbb{Z}^Q$. For each $0 \leq \nu \leq Q$, σ_ν represents the exponent of the undetermined coefficient τ_ν.

(3) $n \in \mathbb{N}$ is the degree of the monomial t.

(4) $\alpha \in \mathbb{R}$ is the exponent of the exponential part of the term.

(5) $\omega \geq 0$ is the frequency of the periodic part.

We have included the two following additional vectors, which are common to all the terms of a modified quasipolynomial:

(1) $(m_1, \ldots, m_Q) \in \{0,1\}^Q$. For each $0 \leq \nu \leq Q$, $m_\nu = 0$ indicates that τ_ν is a coefficient with unknown value, while $m_\nu = 1$ implies that τ_ν has a real value assigned,

(2) $\tau = (\tau_1, \ldots, \tau_Q) \in \mathbb{R}^Q$. If the coefficient τ_ν has a real value assigned, its content is given by τ_ν, for each $0 \leq \nu \leq Q$.

It is also absolutely essential to store the number of undetermined coefficients that are currently in use, to generate correctly new undetermined constants if needed. In Table 1 we illustrate the way a term is coded with a few examples, where it has been assumed that $Q = 4$.

3. Design of the Symbolic System in C++

As pointed out in [15], object-oriented programming using C++ provides many advantages in the design of computer algebra systems, as this programming technique combines both the data and the functions that operate on that data into a single unit (called class). The main reasons given by Hardy et al. to use C++ to implement a symbolic system are as follows.

(1) C++ allows the introduction of abstract data types. Thus, we can define a modified quasipolynomial as an abstract data type.

(2) The language C++ supports encapsulation, inheritance, polymorphism, and operator overloading. Consequently, we can overload the operators +, −, and ∗ for modified quasipolynomials, as well as ∗ and/for multiplication and division of modified quasipolynomials by real numbers.

Some symbolic computation systems have been constructed in C++. `MuPAD` is a computer algebra system developed by the `MuPAD` research group at the University of

```
typedef Term *serie;
class MQ
{
   public:
        [definition of functions]
    private:
       serie first;
};
```

Algorithm 1: C++ code for the definition of a modified quasipolynomial.

Paderborn, in Germany. This symbolic system manipulates formulas symbolically and provides packages for linear algebra, differential equations, number theory, statistics, and functional programming, as well as an interactive graphic system that supports animations and transparent areas in 3D. `MuPAD` also offers a programming language that supports object-oriented programming and functional programming [16].

`Symbolic C++` also uses C++ to develop a computer algebra system. This package introduces the Symbolic class which is used for all symbolic computation, and provides almost all of the features required for symbolic computation including symbolic terms, substitution, noncommutative multiplication, and vectors and matrices.

Both symbolic systems could have been used to implement the algorithm over a general symbolic class. However, as the goal of the symbolic system we are developing is to handle modified quasipolynomial to apply perturbation methods to solve some types of differential equations, we have constructed it directly over C++, instead of using some other symbolic processor.

The specific symbolic processor designed is written in clean C++ code, is very portable, it can compile stand-alone, and is easily embeddable. It implements a new data type, called `MQ`, which represents a series of the form given by (1.2). The class `MQ` is defined as an ordered and dynamical list of terms. To that end, we have also defined a class associated to the structure of a term of a modified quasipolynomial, called `Term`. The definition of these two classes in C++ code is shown in Algorithms 1 and 2, respectively.

The set of routines developed includes: addition, subtraction and product of series, multiplication of a series by a-real number, differentiation with respect to t, and computation of the solution to a linear second-order differential equation of type

$$\ddot{x} + a_1\dot{x} + a_0 x = u(t), \tag{3.1}$$

where $u(t)$ is a modified quasipolynomial presenting undetermined coefficients, and computation of the solution to (1.3) via the asymptotic expansion method. These two algorithms are described in detail in Sections 4 and 5, respectively.

The basic algebra associated to quasipolynomials is easily implemented because of the undercoded data scheme chosen for their computational representation. Thus, for example, the addition of two quasipolynomials is performed by directly juxtaposing both quasipolynomials, arranging the resulting series, and joining terms with equal elements. In a similar way, the rest of algebraic operations are simply implemented.

```
 class Term
{
   public:
       Term ();
        Term (double L, double M,
              int *sigm, int m, double alph, double omeg,
              Term *nxt);
    ~Term ();
   private:
      double lambda, mu;
      int *sigma, n;
      double alpha, omega;
      Term *next;
   friend class MQ;
};
```

Algorithm 2: C++ code for the definition of a term of a modified quasipolynomial.

4. Solution of a Linear Second-Order ODE

The general solution to a nonhomogeneous differential equation can be expressed as the sum of general solutions to the corresponding homogenous, linear differential equation and any solution to the complete equation [17]. The solution to the homogeneous ODE is expressed in terms of the roots of the characteristic equation, $\alpha^2 + a_1\alpha + a_0 = 0$, and it is well-known. We will resume now the formulae that are required to construct a particular solution to a complete ordinary differential equation of second-order (3.1). Without loss of generality, we will assume that $u(t)$ is written as follows:

$$u(t) = \sum_{\nu\geq 0} e^{\alpha_\nu t}\left(p_{\nu,n}(t,\tau)\cos(\omega_\nu t) + q_{\nu,m}(t,\tau)\sin(\omega_\nu t)\right), \tag{4.1}$$

where $\alpha, \omega \in \mathbb{R}$, and $p_{\nu,n}(t,\tau)$, and $q_{\nu,m}(t,\tau)$ are nth and mth degree polynomials in t with undetermined coefficients respectively, of the form

$$p_{\nu,n}(t,\tau) = u_{\nu,0} + u_{\nu,1}t + \cdots + u_{\nu,n}t^n, \qquad q_{\nu,\,m}(t,\tau) = v_{\nu,0} + v_{\nu,1}t + \cdots + v_{\nu,m}t^m, \tag{4.2}$$

being

$$u_{\nu,\rho} = u^*_{\nu,\rho}\ \tau_1^{\sigma_{\nu,\rho,1}} \times \cdots \times \tau_Q^{\sigma_{\nu,\rho,Q}}, \qquad v_{\nu,\rho} = v^*_{\nu,\rho}\ \tau_1^{\sigma^*_{\nu,\rho,1}} \times \cdots \times \tau_Q^{\sigma^*_{\nu,\rho,Q}}, \tag{4.3}$$

with $u^*_{\nu,\rho}, v^*_{\nu,\rho} \in \mathbb{R}$, $\sigma_{\nu,\rho,i}, \sigma^*_{\nu,\rho,i} \in \mathbb{Z}$, $1 \leq i \leq Q$, $0 \leq \rho \leq n$ for $u_{\nu,\rho}$, and $0 \leq \rho \leq m$ for $v_{\nu,\rho}$.

The principle of superposition is applied to calculate the particular solution, so we can focus our attention on the computation of a particular solution to the equation

$$\ddot{x} + a_1\dot{x} + a_0 x = e^{\alpha t}\left(p_n(t,\tau)\cos(\omega t) + q_m(t,\tau)\sin(\omega t)\right), \tag{4.4}$$

where

$$p_n(t,\tau) = u_0 + u_1 t + \cdots + u_n t^n, \qquad q_m(t,\tau) = v_0 + v_1 t + \cdots + v_m t^m, \tag{4.5}$$

being

$$u_\rho = u_\rho^* \, \tau_1^{\sigma_{\rho,1}} \times \cdots \times \tau_Q^{\sigma_{\rho,Q}}, \qquad v_\rho = v_\rho^* \, \tau_1^{\sigma^*_{\rho,1}} \times \cdots \times \tau_Q^{\sigma^*_{\rho,Q}}, \tag{4.6}$$

with $u_\rho^*, v_\rho^* \in \mathbb{R}$, $\sigma_{\rho,i}, \sigma^*_{\rho,i} \in \mathbb{Z}$, $1 \le i \le Q$, $0 \le \rho \le n$ for u_ρ, and $0 \le \rho \le m$ for v_ρ.

At this point, we will distinguish two cases depending on if $\omega = 0$ or $\omega \neq 0$.

Case 1. Let us consider the second-order ODE (4.4), with $\omega = 0$. Then, the equation is written as

$$\ddot{x} + a_1 \dot{x} + a_0 x = p_n(t,\tau) e^{\alpha t}, \tag{4.7}$$

where $p_n(t,\tau)$ is given by (4.5).

Subcase 1.1. If $\alpha^2 + a_1\alpha + a_0 \neq 0$, the particular solution to the complete differential equation is expressed as

$$x(t) = \left(\alpha_0(\tau) + \alpha_1(\tau) t + \alpha_2(\tau) t^2 + \cdots + \alpha_n(\tau) t^n\right) e^{\alpha t}. \tag{4.8}$$

Thus, substituting $x(t)$ and its derivatives in (4.7), we get that

$$\alpha_n(\tau) = \frac{u_n(\tau)}{\alpha^2 + a_1\alpha + a_0}, \qquad \alpha_{n-1}(\tau) = \frac{u_{n-1}(\tau) - (a_1 + 2\alpha) n \alpha_n(\tau)}{\alpha^2 + a_1\alpha + a_0}, \tag{4.9}$$

and, for any $p < n - 1$,

$$\alpha_p(\tau) = \frac{u_p(\tau) - (a_1 + 2\alpha)(p+1)\alpha_{p+1}(\tau) - (p+1)(p+2)\alpha_{p+2}(\tau)}{\alpha^2 + a_1\alpha + a_0}. \tag{4.10}$$

Subcase 1.2. If $\alpha^2 + a_1\alpha + a_0 = 0$ and $a_1^2 \neq 4a_0$ (i.e., $\alpha \neq -a_1/2$), the particular solution can be written as

$$x(t) = \left(\alpha_0(\tau) t + \alpha_1(\tau) t^2 + \alpha_2(\tau) t^3 + \cdots + \alpha_n(\tau) t^{n+1}\right) e^{\alpha t}. \tag{4.11}$$

Now, the substitution of $x(t)$ and its derivatives into (4.7) leads to the following formula for $\alpha_n(\tau)$,

$$\alpha_n(\tau) = \frac{u_n(\tau)}{(2\alpha + a_1)(n+1)}, \tag{4.12}$$

and, for any $p \leq n-1$,

$$\alpha_p(\tau) = \frac{u_p(\tau) - (p+1)(p+2)\alpha_{p+1}(\tau)}{(2\alpha + a_1)(p+1)}. \tag{4.13}$$

Subcase 1.3. If $\alpha^2 + a_1\alpha + a_0 = 0$ and $a_1^2 = 4a_0$ (i.e., $\alpha = -a_1/2$), the particular solution can be written as

$$x(t) = \left(\alpha_0(\tau)t^2 + \alpha_1(\tau)t^3 + \alpha_2(\tau)t^4 + \cdots + \alpha_n(\tau)t^{n+2}\right)e^{\alpha t}. \tag{4.14}$$

Now, the substitution of $x(t)$ and its derivatives into (4.7) leads to the following formula for $\alpha_p(\tau)$,

$$\alpha_p(\tau) = \frac{u_p(\tau)}{(p+1)(p+2)}, \quad p = 0, \ldots, n. \tag{4.15}$$

Case 2. Let us consider the second-order differential equation (4.4), where now $\omega \neq 0$. There are two possible situations:

(1) $\alpha \pm i\omega$ is not a root of the characteristic equation, that is, $\alpha \neq -a_1/2$ or $\omega \neq \sqrt{4a_0 - a_1^2}/2$,

(2) $\alpha \pm i\omega$ is a root of the characteristic equation, that is, $\alpha = -a_1/2$ and $\omega = \sqrt{4a_0 - a_1^2}/2$.

Subcase 2.1. Let us call $N = \max\{n, m\}$. Then,

$$\begin{aligned} x(t) = {} & \left(\alpha_0(\tau) + \alpha_1(\tau)t + \cdots + \alpha_N(\tau)t^N\right)e^{\alpha t}\cos(\omega t) \\ & + \left(\rho_0(\tau) + \rho_1(\tau)t + \cdots + \rho_N(\tau)t^N\right)e^{\alpha t}\sin(\omega t). \end{aligned} \tag{4.16}$$

Now, if we assume $p_N(t,\tau)$ and $q_N(t,\tau)$ to be

$$p_N(t,\tau) = u_0(\tau) + u_1(\tau)t + \cdots + u_N(\tau)t^N, \qquad q_N(t,\tau) = v_0(\tau) + v_1(\tau)t + \cdots + v_N(\tau)t^N, \tag{4.17}$$

and substitute $x(t)$ and its derivatives into (4.4), we get that

$$\alpha_N(\tau) = \frac{1}{\Delta}\begin{vmatrix} u_N(\tau) & \omega(a_1+2\alpha) \\ v_N(\tau) & a_0+a_1\alpha+\alpha^2-\omega^2 \end{vmatrix},$$
$$\rho_N(\tau) = \frac{1}{\Delta}\begin{vmatrix} a_0+a_1\alpha+\alpha^2-\omega^2 & u_N(\tau) \\ -\omega(a_1+2\alpha) & v_N(\tau) \end{vmatrix}, \tag{4.18}$$

where $\Delta = (a_0+a_1\alpha+\alpha^2-\omega^2)^2+\omega^2(a_1+2\alpha)^2$. Note that $\Delta \neq 0$, because $a_0+a_1\alpha+\alpha^2-\omega^2 \neq 0$ or $\omega(2\alpha+a_1) \neq 0$. Now, we can compute $\alpha_{N-1}(\tau)$ and $\rho_{N-1}(\tau)$ by solving the system

$$\begin{aligned} &u_{N-1}(\tau) - N(a_1+2\alpha)\ \alpha_N(\tau) - 2N\omega\rho_N(\tau) \\ &\quad = \left(a_0+a_1\alpha+\alpha^2-\omega^2\right)\alpha_{N-1}(\tau) + (a_1\omega+2\alpha\omega)\rho_{N-1}(\tau), \\ &v_{N-1}(\tau) - N(a_1+2\alpha)\rho_N(\tau) + 2N\omega\alpha_N(\tau) \\ &\quad = -\omega(a_1+2\alpha)\alpha_{N-1}(\tau) + \left(a_0+a_1\alpha+\alpha^2-\omega^2\right)\ \rho_{N-1}(\tau). \end{aligned} \tag{4.19}$$

As before, this system can be solved by applying the Cramer's rule. Finally, for any $p < N-1$, we have to solve the system in $\alpha_p(\tau)$ and $\rho_p(\tau)$, as follows:

$$\begin{aligned} &u_p(\tau) - (p+2)(p+1)\ \alpha_{p+2}(\tau) - (p+1)(a_1+2\alpha)\ \alpha_{p+1}(\tau) - 2(p+1)\omega\rho_{p+1}(\tau) \\ &\quad = \left(a_0+a_1\alpha+\alpha^2-\omega^2\right)\alpha_p(\tau) + \omega(a_1+2\alpha)\rho_p(\tau), \\ &v_p(\tau) - (p+2)(p+1)\ \rho_{p+2}(\tau) - (p+1)(a_1+2\alpha)\ \rho_{p+1}(\tau) + 2(p+1)\omega\alpha_{p+1}(\tau) \\ &\quad = -\omega(a_1+2\alpha)\alpha_p(\tau) + \left(a_0+a_1\alpha+\alpha^2-\omega^2\right)\rho_p(\tau). \end{aligned} \tag{4.20}$$

Subcase 2.2. Now we consider the case where $\alpha = -a_1/2$ and $\omega = \sqrt{4a_0-a_1^2}/2$. This implies that $a_1+2\alpha = 0$ and $\alpha^2-\omega^2+a_1\alpha+a_0 = 0$. In this case, the particular solution adopts the form

$$\begin{aligned} x(t) = &\left(\alpha_0(\tau)t + \alpha_1(\tau)t^2 + \alpha_2(\tau)t^3 + \cdots + \alpha_N(\tau)t^{N+1}\right)e^{\alpha t}\cos(\omega t) \\ &+ \left(\rho_0(\tau)t + \rho_1(\tau)t^2 + \rho_2(\tau)t^3 + \cdots + \rho_N(\tau)t^{N+1}\right)e^{\alpha t}\sin(\omega t), \end{aligned} \tag{4.21}$$

and substituting $x(t)$, $\dot{x}(t)$, and $\ddot{x}(t)$ into (4.4), we obtain

$$\rho_N(\tau) = \frac{u_N(\tau)}{2(N+1)\omega}, \qquad \alpha_N(\tau) = -\frac{v_N(\tau)}{2(N+1)\omega},$$
$$\rho_p(\tau) = \frac{u_p(\tau) - (p+1)(p+2)\alpha_{p+1}}{2(p+1)\omega}, \qquad \alpha_p(\tau) = \frac{v_p(\tau) - (p+1)(p+2)\rho_{p+1}}{-2(p+1)\omega}, \tag{4.22}$$

for any $p < N$.

From (4.8)–(4.22), it is a straightforward task; the derivation of an algorithm for the computation of the solution to (4.4).

5. Computation of the Solution to the Perturbed Problem

The symbolic package is thought to compute the solution to (1.3). The standard approach [13] is to try a power series solution of the form

$$x(t,\epsilon) = x_0(t) + x_1(t)\ \epsilon + x_2(t)\ \epsilon^2 + \cdots. \tag{5.1}$$

This series is inserted into the governing equation and initial conditions, and coefficients of same powers of ϵ are then grouped to obtain a collection of equations for the coefficient functions $x_i(t)$, which are then solved in a sequential manner. The resulting series need not converge for any value of ϵ; nevertheless, the solution $x(t,\epsilon)$ can be useful in approximating the function when ϵ is small.

Considering the zero-order term in ϵ yields

$$\ddot{x}_0 + a_1\dot{x}_0 + a_0x_0 = u(t), \qquad x(t_0) = x_0, \qquad \dot{x}(t_0) = \dot{x}_0, \tag{5.2}$$

the so-called nonperturbed problem. The symbolic manipulation system calculates the solution to a differential equation of the form (5.2), and arranges it as a quasipolynomial, as it has been described in detail in the previous section.

The coefficient $x_q(t)$ of the solution to the order $q \geq 1$ is computed by solving the equation

$$\ddot{x}_q + a_1\dot{x}_q + a_0x_q = \sum_{\kappa=0}^{M} \sum_{0\leq\nu\leq\kappa} f_{\nu,\kappa-\nu}\ \left(x^\nu\ \dot{x}^{\kappa-\nu}\right)_{q-1}, \qquad x_q(t_0) = 0, \qquad \dot{x}_q(t_0) = 0, \tag{5.3}$$

where the notation $(x^\nu\dot{x}^{\kappa-\nu})_q$ refers to the qth order term of the series $x^\nu\dot{x}^{\kappa-\nu}$.

At each order of the solution, the series $(x^\nu)_q$, $(\dot{x}^\nu)_q$, and $(x^\nu \dot{x}^{\kappa-\nu})_q$ must be computed once the order q has been solved, for each $q \geq 0$, following the formulae

$$\begin{aligned}
(x^\nu)_q &= \sum_{0\leq p\leq q} \left(x^{\nu-1}\right)_p (x)_{q-p},\\
(\dot{x}^\nu)_q &= \sum_{0\leq p\leq q} \left(\dot{x}^{\nu-1}\right)_p (\dot{x})_{q-p},\\
\left(x^\nu \dot{x}^{\kappa-\nu}\right)_q &= \sum_{0\leq p\leq q} (x^\nu)_p \left(\dot{x}^{\kappa-\nu}\right)_{q-p}.
\end{aligned} \tag{5.4}$$

According to this, the algorithm to apply the asymptotic expansion method to solve the initial value problem given by (1.3), consists of the following steps.

(1) Define a three-dimensional array of quasipolynomials $X(\rho_1, \rho_2, q)$, where $\rho_1, \rho_2, q \in \mathbb{N}$, $0 \leq \rho_1, \rho_2 \leq M$, and $0 \leq q \leq Q$, Q being the order of the asymptotic expansion.

(2) Define an array of quasipolynomials $x(\rho)$, where $0 \leq \rho \leq Q$.

(3) Initialize $X(0,0,0) = 1$, and proceed as follows:

(3.1) Compute $X(1,0,0)$ as the solution to (5.2).

(3.2) Compute $X(0,1,0) = d/dt(X(1,0,0))$.

(3.3) Calculate, for each ρ such that $2 \leq \rho \leq M$,

$$\begin{aligned}
X(\rho,0,0) &= X(1,0,0) \times X(\rho-1,0,0),\\
X(0,\rho,0) &= X(0,1,0) \times X(0,\rho-1,0).
\end{aligned} \tag{5.5}$$

(3.4) For each ρ_1, ρ_2 such that $1 \leq \rho_1, \rho_2 \leq M$, determine the quasipolynomial

$$X(\rho_1,\rho_2,0) = X(\rho_1,0,0) \times X(0,\rho_2,0). \tag{5.6}$$

(4) For each q such that $1 \leq q \leq Q$, do the following.

(4.1) Compute the quasipolynomial

$$U = \sum_{\kappa=0}^{M} \sum_{0\leq\rho\leq\kappa} f_{\rho,\kappa-\rho} \times X(\rho, \kappa-\rho, q-1)\ . \tag{5.7}$$

(4.2) Calculate $X(1,0,q)$ as the solution to (5.3),

$$\ddot{x}_q + a_1 \dot{x}_q + a_0 x_q = U, \qquad x_q(t_0) = 0, \qquad \dot{x}_q(t_0) = 0. \tag{5.8}$$

(4.3) Compute $X(0,1,q) = d/dt(X(1,0,q))$.

(4.4) Calculate, for each ρ such that $2 \le \rho \le M$,

$$\begin{aligned} X(\rho,0,q) &= \sum_{0\le p\le q} X(\rho-1,0,p)\times X(1,0,q-p),\\ X(0,\rho,q) &= \sum_{0\le p\le q} X(0,\rho-1,p)\times X(0,1,q-p). \end{aligned} \tag{5.9}$$

(4.5) For each ρ_1, ρ_2 such that $1 \le \rho_1,\ \rho_2 \le M$, compute

$$X(\rho_1,\rho_2,q) = \sum_{0\le p\le q} X(\rho_1,0,p)\times X(0,\rho_2,q-p). \tag{5.10}$$

(5) For each ρ such that $0 \le \rho \le Q$,

$$x(\rho) = X(1,0,\rho), \qquad x(t) = \sum_{0\le\rho\le Q} \epsilon^{\rho} x(\rho). \tag{5.11}$$

The input arguments of the algorithm consist of the order Q of the asymptotic approximation, the coefficients a_1, a_0 of the differential equation, the real values t_0, x_0, and $\dot{x}_0$ which define the initial conditions of the problem, the quasipolynomial $u(t)$, and the perturbed part of the equation $f(x,\dot{x})$. As f can be written as given in (1.4),

$$f(x,\dot{x}) = \sum_{\kappa=0}^{M} \sum_{0\le\nu\le\kappa} f_{\nu,\kappa-\nu}\ x^{\nu}\ \dot{x}^{\kappa-\nu}, \tag{5.12}$$

it can be specified by a real $(M+1)\times(M+1)$ matrix,

$$f(x,\dot{x}) = \begin{pmatrix} 1 & x & \cdots & x^{M} \end{pmatrix} \begin{pmatrix} f_{00} & f_{01} & \cdots & f_{0M} \\ f_{10} & f_{11} & \cdots & f_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M0} & f_{M1} & \cdots & f_{MM} \end{pmatrix} \begin{pmatrix} 1 \\ \dot{x} \\ \vdots \\ \dot{x}^{M} \end{pmatrix}. \tag{5.13}$$

The parameter M is also required as input.

The output argument of the algorithm is the array $x(\rho)$ containing the coefficients of the asymptotic expansion. The asymptotic approximation to the order Q is given by

$$x(t) = \sum_{0\le\rho\le Q} \epsilon^{\rho} x(\rho). \tag{5.14}$$

```
MQ xdx [M + 1] [M + 1] [ORDER + 1];
```
Define a three-dimensional array of quasi polynomials $X(\rho_1, \rho_2, q)$, where $\rho_1, \rho_2, q \in N$, $0 \le \rho_1, \rho_2 \le M$ and $0 \le q \le Q$, being Q the order of the asymptotic expansion.

```
MQ *s;
s=new MQ [ORDER + 1];
```
Define an array of quasipolynomials $x(\rho)$, where $0 \le \rho \le Q$.

```
int i, j, k, p, q, nu;
MQ U;
```
Definition of auxiliary variables.

Algorithm 3: Implementation of the algorithm. Steps (1) and (2).

```
xdx [0] [0] [0].add
(1., 0., 0, 0., 0.);
```
Initialize $X(0,0,0) = 1$.

```
xdx [1] [0] [0] = u.solvePVI
(a1, a0, t0, x0, dx0);
xdx [0] [1] [0] = xdx [1] [0] [0].der ();
```
Compute $X(1,0,0)$ as the solution to (5.2), and $X(0,1,0) = d/dt(X(1,0,0))$.

Algorithm 4: Implementation of the algorithm. Steps (3.1) and (3.2).

```
for (i = 2; i <= m; i + +){
   xdx [i] [0] [0] =
        xdx [1] [0] [0] *xdx [i − 1] [0] [0];
   xdx [0] [i] [0] =
        xdx [0] [1] [0] *xdx [0] [i − 1] [0];

}
for (i=1; i<=m; i++)
  for (j=1; j<=m; j++)
    xdx [i] [j] [0] =
        xdx [i] [0] [0]*xdx [0] [j] [0];
```
Calculate, for each ρ such that $2 \le \rho \le M$,

$X(\rho,0,0) = X(1,0,0) \times X(\rho - 1,0,0)$,

$X(0,\rho,0) = X(0,1,0) \times X(0,\rho - 1,0)$.

For each ρ_1, ρ_2 such that $1 \le \rho_1, \rho_2 \le M$, determine the quasipolynomial

$X(\rho_1,\rho_2,0) = X(\rho_1,0,0) \times X(0,\rho_2,0)$.

Algorithm 5: Implementation of the algorithm. Steps (3.3) and (3.4).

6. On the Implementation of the Algorithm

In Algorithms 3, 4, 5, 6, and 7 we show the C++ code for the implementation of the algorithm as described in Section 5, omitting error control sentences for the sake of simplicity. The head of the definition of the function that implements the algorithm is shown below.

```
MQ *solveAM (double a1, double a0, double t0, double x0, double dx0, MQ
u, double **f, int m, int order)
```

```
for (q=1; q<=order; q++)
{

  U=U*0.;
   for (k=0; k<=m; k++)
     for (nu=0; nu<=k; nu++)

       U = U + xdx [nu] [k-nu] [q-1]
               *f [nu] [k-nu];

  xdx [1] [0] [q] = U.solvePVI
  (a1, a0, t0, 0., 0.);
  xdx [0] [1] [q] = xdx [1] [0] [q].der ();



  for (i=2; i<=m; i++)
    for (nu=0; nu<=q; nu++) {
      xdx [i] [0] [q] = xdx [i] [0] [q]
         + xdx [i - 1] [0] [nu]
           *xdx [1] [0] [q - nu];
      xdx [0] [i] [q] = xdx [0] [i] [q]
           + xdx [0] [i - 1] [nu]
            *xdx [0] [1] [q - nu];
    }

  for (i=1; i<=m; i++)
    for (j=1; j<=m; j++)
      for (nu=0; nu<=q; nu++)
        xdx [i] [j] [q] =
          xdx [i] [j] [q]
          + xdx [i] [0] [nu]
             *xdx [0] [j] [q - nu];
}
```

For each q such that $1 \le q \le Q$, do:

Compute the quasipolynomial

$$U = \sum_{\kappa=0}^{M} \sum_{0 \le \rho \le \kappa} f_{\rho,\kappa-\rho} \times X(\rho, \kappa - \rho, q-1).$$

Calculate $X(1,0,q)$ as the solution to (5.3),

$$\ddot{x}_q + a_1 \dot{x}_q + a_0 x_q = U,$$
$$x_q(t_0) = 0, \ \dot{x}_q(t_0) = 0,$$
and $X(0,1,q) = d/dt(X(1,0,q))$.

Calculate, for each ρ such that $2 \le \rho \le M$,

$$X(\rho,0,q) = \sum_{0 \le p \le q} X(\rho - 1, 0, p) \times X(1,0,q-p),$$
$$X(0,\rho,q) = \sum_{0 \le p \le q} X(0, \rho - 1, p) \times X(0,1,q-p).$$

For each ρ_1, ρ_2 such that $1 \le \rho_1, \rho_2 \le M$, compute

$$X(\rho_1, \rho_2, q) = \sum_{0 \le p \le q} X(\rho_1, 0, p) \times X(0, \rho_2, q-p).$$

Algorithm 6: Implementation of the algorithm. Step (4).

7. Example

Let us consider the initial value problem given by

$$\ddot{x} + x = \epsilon \left(-x^3 - \dot{x}\right) = \epsilon \, f(x, \dot{x}), \qquad x(0) = 1, \qquad \dot{x}(0) = 0, \tag{7.1}$$

where $\epsilon = 10^{-1}$. Here, $a_1 = 0$, $a_0 = 1$, $t_0 = 0$, $x_0 = 1$, $\dot{x}_0 = 0$, $M = 3$, and f is defined by the following 4×4 matrix:

$$F = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}. \tag{7.2}$$

```
for (i=0; i<=order; i++){
  s [i] = xdx [1] [0] [i];
   s [i].normalice ();
  s [i].order ();
  s [i].join ();
  s [i].neglect (PREC);
}
```

For each ρ such that $0 \leq \rho \leq Q$,

$x(\rho) = X(1, 0, \rho)$, and $x(t) = \sum_{0 \leq \rho \leq Q} \epsilon^{\rho} x(\rho)$.

Algorithm 7: Implementation of the algorithm. Step (5).

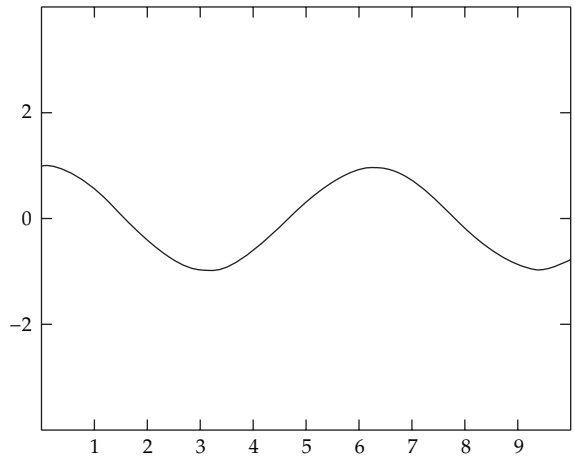


Figure 1: Comparison between a fourth order symbolic and a numerical solution.

In Table 2 we show the output generated from the symbolic manipulator, just to order nine for the sake of simplicity, as the number of terms of the asymptotic expansion increases very quickly with the order of the expansion.

In Figure 1, we show a comparison of the solution computed through the symbolic method (to the fourth-order of the solution) presented in this paper, and the numeric solution to the problem calculated by a Runge-Kutta fourth-order method with a step of $h = 0.001$. Let us stress that the difference between both solutions at any time is smaller than 10^{-7}. The numerical solution has been calculated with the only purpose of validating the symbolic solution, as it is not the goal of this paper to develop a numerical tool.

In Table 3, we compare the CPU time of the algorithm implemented with `MQSP` and `Maple` running on an iMac 2.8 GHz Intel Core 2 Duo. For the test problem described by (7.1), we have calculated the solution up to order 14. In Table 3, we show the CPU time for the different orders of the solution from $x_0(t)$ (nonperturbed problem) to $x_{14}(t)$. The computation time of $x_4(t)$ with the use of the `dsolve` command of `Maple` in the implementation of the algorithm is larger than 100 000 seconds. This is due to the fact that `dsolve` is a general solver which handles different types of ordinary differential equations. We have also computed the symbolic solution taking into account the linearity of (7.1) and implementing a routine following the algorithm detailed in Section 4. These CPU times are given in the fourth column of Table 3. For orders of the solution larger than 11, `Maple` does not give any response. We can see that the difference between `MQSP` and `Maple` in terms of CPU time is significant.

Table 2: Some coefficients of the asymptotic expansion of the solution to (7.1).

$x_0 =$	$+1.000000\cos(t)$		
$x_1 =$	$-0.031250\cos(t)$	$+0.031250\cos(3t)$	$+0.500000\sin(t)$
	$-0.500000t\cos(t)$	$-0.375000t\sin(t)$	
$x_2 =$	$+0.022461\cos(t)$	$-0.023438\cos(3t)$	$+0.000977\cos(5t)$
	$+0.226563\sin(t)$	$+0.070313\sin(3t)$	$-0.390625t\cos(t)$
	$-0.046875t\cos(3t)$	$-0.031250t\sin(t)$	$-0.035156t\sin(3t)$
	$+0.054688t^2\cos(t)$	$+0.375000t^2\sin(t)$	
$x_3 =$	$+0.047760\cos(t)$	$-0.046326\cos(3t)$	$-0.001465\cos(5t)$
	$+0.000031\cos(7t)$	$+0.122640\sin(t)$	$-0.005859\sin(3t)$
	$+0.003988\sin(5t)$	$-0.201660t\cos(t)$	$+0.079102t\cos(3t)$
	$-0.002441t\cos(5t)$	$-0.133057t\sin(t)$	$-0.059692t\sin(3t)$
	$-0.001831t\sin(5t)$	$+0.103760t^2\cos(t)$	$+0.015381t^2\cos(3t)$
	$+0.140625t^2\sin(t)$	$+0.070313t^2\sin(3t)$	$+0.084635t^3\cos(t)$
	$-0.194336t^3\sin(t)$		
$x_4 =$	$+0.031557\cos(t)$	$-0.025802\cos(3t)$	$-0.005688\cos(5t)$
	$-0.000069\cos(7t)$	$+0.000001\cos(9t)$	$+0.314784\sin(t)$
	$+0.001236\sin(3t)$	$-0.003563\sin(5t)$	$+0.000181\sin(7t)$
	$-0.329605t\cos(t)$	$+0.018219t\cos(3t)$	$+0.009552t\cos(5t)$
	$-0.000107t\cos(7t)$	$-0.441519t\sin(t)$	$+0.023666t\sin(3t)$
	$-0.006212t\sin(5t)$	$-0.000080t\sin(7t)$	$+0.340805t^2\cos(t)$
	$-0.113068t^2\cos(3t)$	$+0.001335t^2\cos(5t)$	$+0.323486t^2\sin(t)$
	$-0.007141t^2\sin(3t)$	$+0.005493t^2\sin(5t)$	$-0.113566t^3\cos(t)$
	$+0.031860t^3\cos(3t)$	$-0.168498t^3\sin(t)$	$-0.064362t^3\sin(3t)$
	$-0.081533t^4\cos(t)$	$+0.060547t^4\sin(t)$	
$x_5 =$	$+0.101090\cos(t)$	$-0.103018\cos(3t)$	$+0.002327\cos(5t)$
	$-0.000396\cos(7t)$	$-0.000003\cos(9t)$	$+0.000000\cos(11t)$
	$+0.567049\sin(t)$	$+0.011443\sin(3t)$	$-0.003304\sin(5t)$
	$-0.000300\sin(7t)$	$+0.000007\sin(9t)$	$-0.630124t\cos(t)$
	$+0.042347t\cos(3t)$	$+0.004305t\cos(5t)$	$+0.000649t\cos(7t)$
	$-0.000004t\cos(9t)$	$-0.861960t\sin(t)$	$-0.136819t\sin(3t)$
	$+0.014968t\sin(5t)$	$-0.000407t\sin(7t)$	$-0.000003t\sin(9t)$
	$+0.769419t^2\cos(t)$	$+0.056821t^2\cos(3t)$	$-0.019625t^2\cos(5t)$
	$+0.000082t^2\cos(7t)$	$+0.653250t^2\sin(t)$	$-0.009047t^2\sin(3t)$
	$-0.002311t^2\sin(5t)$	$+0.000320t^2\sin(7t)$	$-0.271820t^3\cos(t)$
	$+0.077614t^3\cos(3t)$	$+0.003465t^3\cos(5t)$	$-0.512211t^3\sin(t)$
	$+0.066607t^3\sin(3t)$	$-0.007243t^3\sin(5t)$	$+0.084625t^4\cos(t)$
	$-0.054769t^4\cos(3t)$	$+0.110530t^4\sin(t)$	$+0.027557t^4\sin(3t)$
	$+0.046019t^5\cos(t)$	$-0.005018t^5\sin(t)$	
$x_6 =$	$+0.041699\cos(t)$	$-0.035635\cos(3t)$	$-0.006540\cos(5t)$
	$+0.000498\cos(7t)$	$-0.000022\cos(9t)$	$-0.000000\cos(11t)$
	$+0.000000\cos(13t)$	$+1.753506\sin(t)$	$-0.124680\sin(3t)$
	$+0.000293\sin(5t)$	$-0.000409\sin(7t)$	$-0.000018\sin(9t)$
	$+0.000000\sin(11t)$	$-1.732879t\cos(t)$	$+0.361830t\cos(3t)$
	$-0.007317t\cos(5t)$	$+0.000416t\cos(7t)$	$+0.000035t\cos(9t)$

Table 2: Continued.

	$-0.000000t\cos(11t)$	$-1.950362t\sin(t)$	$+0.090323t\sin(3t)$
	$-0.009679t\sin(5t)$	$+0.001762t\sin(7t)$	$-0.000022t\sin(9t)$
	$-0.000000t\sin(11t)$	$+1.753296t^2\cos(t)$	$-0.260570t^2\cos(3t)$
	$+0.014763t^2\cos(5t)$	$-0.001785t^2\cos(7t)$	$+0.000004t^2\cos(9t)$
	$+1.911049t^2\sin(t)$	$+0.270222t^2\sin(3t)$	$-0.022385t^2\sin(5t)$
	$-0.000220t^2\sin(7t)$	$+0.000016t^2\sin(9t)$	$-1.003710t^3\cos(t)$
	$-0.074659t^3\cos(3t)$	$+0.018743t^3\cos(5t)$	$+0.000255t^3\cos(7t)$
	$-1.212436t^3\sin(t)$	$-0.124235t^3\sin(3t)$	$+0.019212t^3\sin(5t)$
	$-0.000552t^3\sin(7t)$	$+0.305366t^4\cos(t)$	$-0.009847t^4\cos(3t)$
	$-0.008565t^4\cos(5t)$	$+0.511480t^4\sin(t)$	$-0.077788t^4\sin(3t)$
	$+0.004177t^4\sin(5t)$	$-0.061540t^5\cos(t)$	$+0.046341t^5\cos(3t)$
	$-0.067702t^5\sin(t)$	$+0.007443t^5\sin(3t)$	$-0.018889t^6\cos(t)$
	$-0.007806t^6\sin(t)$		
$x_7 =$	$-0.030141\cos(t)$	$+0.026721\cos(3t)$	$+0.003535\cos(5t)$
	$-0.000160\cos(7t)$	$+0.000046\cos(9t)$	$-0.000001\cos(11t)$
	$-0.000000\cos(13t)$	$+0.000000\cos(15t)$	$+5.481497\sin(t)$
	$+0.136811\sin(3t)$	$-0.022934\sin(5t)$	$+0.000391\sin(7t)$
	$-0.000032\sin(9t)$	$-0.000001\sin(11t)$	$+0.000000\sin(13t)$
	$-5.726072t\cos(t)$	$-0.099462t\cos(3t)$	$+0.047957t\cos(5t)$
	$-0.002159t\cos(7t)$	$+0.000029t\cos(9t)$	$+0.000002t\cos(11t)$
	$-0.000000t\cos(13t)$	$-5.240757t\sin(t)$	$+0.347041t\sin(3t)$
	$+0.020306t\sin(5t)$	$-0.000669t\sin(7t)$	$+0.000136t\sin(9t)$
	$-0.000001t\sin(11t)$	$-0.000000t\sin(13t)$	$+4.898568t^2\cos(t)$
	$-0.633927t^2\cos(3t)$	$-0.017751t^2\cos(5t)$	$+0.002044t^2\cos(7t)$
	$-0.000122t^2\cos(9t)$	$+0.000000t^2\cos(11t)$	$+5.910751t^2\sin(t)$
	$-0.423093t^2\sin(3t)$	$+0.042183t^2\sin(5t)$	$-0.003593t^2\sin(7t)$
	$-0.000015t^2\sin(9t)$	$+0.000001t^2\sin(11t)$	$-3.289085t^3\cos(t)$
	$+0.621827t^3\cos(3t)$	$-0.037683t^3\cos(5t)$	$+0.002208t^3\cos(7t)$
	$+0.000015t^3\cos(9t)$	$-3.616263t^3\sin(t)$	$-0.288847t^3\sin(3t)$
	$+0.002661t^3\sin(5t)$	$+0.002305t^3\sin(7t)$	$-0.000034t^3\sin(9t)$
	$+1.238058t^4\cos(t)$	$-0.002983t^4\cos(3t)$	$-0.002782t^4\cos(5t)$
	$-0.000822t^4\cos(7t)$	$+1.664171t^4\sin(t)$	$+0.252574t^4\sin(3t)$
	$-0.030580t^4\sin(5t)$	$+0.000408t^4\sin(7t)$	$-0.311598t^5\cos(t)$
	$-0.037610t^5\cos(3t)$	$+0.009709t^5\cos(5t)$	$-0.443419t^5\sin(t)$
	$+0.045286t^5\sin(3t)$	$+0.002005t^5\sin(5t)$	$+0.041874t^6\cos(t)$
	$-0.023862t^6\cos(3t)$	$+0.045263t^6\sin(t)$	$-0.022910t^6\sin(3t)$
	$+0.005561t^7\cos(t)$	$+0.006555t^7\sin(t)$	
$x_8 =$	$+0.707697\cos(t)$	$-0.737652\cos(3t)$	$+0.029647\cos(5t)$
	$+0.000299\cos(7t)$	$+0.000006\cos(9t)$	$+0.000003\cos(11t)$
	$-0.000000\cos(13t)$	$-0.000000\cos(15t)$	$+17.859287\sin(t)$
	$+0.537553\sin(3t)$	$+0.020622\sin(5t)$	$-0.001769\sin(7t)$
	$+0.000063\sin(9t)$	$-0.000002\sin(11t)$	$-0.000000\sin(13t)$
	$+0.000000\sin(15t)$	$-19.388600t\cos(t)$	$-0.105739t\cos(3t)$

Table 2: Continued.

	$-0.072738t\cos(5t)$	$+0.004113t\cos(7t)$	$-0.000258t\cos(9t)$
	$+0.000002t\cos(11t)$	$+0.000000t\cos(13t)$	$-0.000000t\cos(15t)$
	$-16.767762t\sin(t)$	$-0.906868t\sin(3t)$	$+0.089594t\sin(5t)$
	$+0.000388t\sin(7t)$	$-0.000052t\sin(9t)$	$+0.000008t\sin(11t)$
	$-0.000000t\sin(13t)$	$-0.000000t\sin(15t)$	$+16.120089t^2\cos(t)$
	$+0.421931t^2\cos(3t)$	$-0.092803t^2\cos(5t)$	$+0.001418t^2\cos(7t)$
	$+0.000192t^2\cos(9t)$	$-0.000007t^2\cos(11t)$	$+0.000000t^2\cos(13t)$
	$+19.608050t^2\sin(t)$	$-0.420458t^2\sin(3t)$	$-0.128984t^2\sin(5t)$
	$+0.005901t^2\sin(7t)$	$-0.000351t^2\sin(9t)$	$-0.000001t^2\sin(11t)$
	$+0.000000t^2\sin(13t)$	$-11.042356t^3\cos(t)$	$+0.875871t^3\cos(3t)$
	$+0.101426t^3\cos(5t)$	$-0.006986t^3\cos(7t)$	$+0.000187t^3\cos(9t)$
	$+0.000001t^3\cos(11t)$	$-12.224352t^3\sin(t)$	$+0.801292t^3\sin(3t)$
	$-0.047084t^3\sin(5t)$	$+0.002053t^3\sin(7t)$	$+0.000194t^3\sin(9t)$
	$-0.000002t^3\sin(11t)$	$+4.715957t^4\cos(t)$	$-0.906645t^4\cos(3t)$
	$+0.033974t^4\cos(5t)$	$-0.000385t^4\cos(7t)$	$-0.000061t^4\cos(9t)$
	$+5.809016t^4\sin(t)$	$+0.172731t^4\sin(3t)$	$+0.039748t^4\sin(5t)$
	$-0.004671t^4\sin(7t)$	$+0.000031t^4\sin(9t)$	$-1.468495t^5\cos(t)$
	$+0.132440t^5\cos(3t)$	$-0.017574t^5\cos(5t)$	$+0.001171t^5\cos(7t)$
	$-1.830821t^5\sin(t)$	$-0.280024t^5\sin(3t)$	$+0.025814t^5\sin(5t)$
	$+0.000249t^5\sin(7t)$	$+0.278969t^6\cos(t)$	$+0.046180t^6\cos(3t)$
	$-0.006166t^6\cos(5t)$	$+0.377937t^6\sin(t)$	$-0.001896t^6\sin(3t)$
	$-0.006983t^6\sin(5t)$	$-0.028889t^7\cos(t)$	$+0.004692t^7\cos(3t)$
	$-0.029867t^7\sin(t)$	$+0.021274t^7\sin(3t)$	$-0.000816t^8\cos(t)$
	$-0.003373t^8\sin(t)$		
$x_9 =$	$+2.175721\cos(t)$	$-2.094928\cos(3t)$	$-0.084502\cos(5t)$
	$+0.003726\cos(7t)$	$-0.000018\cos(9t)$	$+0.000001\cos(11t)$
	$+0.000000\cos(13t)$	$-0.000000\cos(15t)$	$-0.000000\cos(17t)$
	$+64.332742\sin(t)$	$+0.333348\sin(3t)$	$+0.044040\sin(5t)$
	$+0.002384\sin(7t)$	$-0.000104\sin(9t)$	$+0.000006\sin(11t)$
	$-0.000000\sin(13t)$	$-0.000000\sin(15t)$	$+0.000000\sin(17t)$
	$-68.359632t\cos(t)$	$+2.878461t\cos(3t)$	$-0.079121t\cos(5t)$
	$-0.008828t\cos(7t)$	$+0.000335t\cos(9t)$	$-0.000022t\cos(11t)$
	$+0.000000t\cos(13t)$	$+0.000000t\cos(15t)$	$-0.000000t\cos(17t)$
	$-64.047243t\sin(t)$	$-1.871954t\sin(3t)$	$-0.188447t\sin(5t)$
	$+0.009868t\sin(7t)$	$-0.000169t\sin(9t)$	$-0.000004t\sin(11t)$
	$+0.000000t\sin(13t)$	$-0.000000t\sin(15t)$	$-0.000000t\sin(17t)$
	$+61.716104t^2\cos(t)$	$-0.768222t^2\cos(3t)$	$+0.295936t^2\cos(5t)$
	$-0.011360t^2\cos(7t)$	$+0.000383t^2\cos(9t)$	$+0.000014t^2\cos(11t)$
	$-0.000000t^2\cos(13t)$	$+0.000000t^2\cos(15t)$	$+69.633323t^2\sin(t)$
	$+2.498056t^2\sin(3t)$	$-0.113525t^2\sin(5t)$	$-0.012762t^2\sin(7t)$
	$+0.000674t^2\sin(9t)$	$-0.000026t^2\sin(11t)$	$-0.000000t^2\sin(13t)$
	$+0.000000t^2\sin(15t)$	$-40.085044t^3\cos(t)$	$-0.028980t^3\cos(3t)$
	$+0.047373t^3\cos(5t)$	$+0.009129t^3\cos(7t)$	$-0.000811t^3\cos(9t)$
	$+0.000013t^3\cos(11t)$	$+0.000000t^3\cos(13t)$	$-46.345112t^3\sin(t)$

Table 2: Continued.

$+0.014214t^3\sin(3t)$	$+0.328854t^3\sin(5t)$	$-0.010719t^3\sin(7t)$
$+0.000291t^3\sin(9t)$	$+0.000013t^3\sin(11t)$	$-0.000000t^3\sin(13t)$
$+18.758768t^4\cos(t)$	$-1.045383t^4\cos(3t)$	$-0.199353t^4\cos(5t)$
$+0.009593t^4\cos(7t)$	$-0.000040t^4\cos(9t)$	$-0.000004t^4\cos(11t)$
$+21.493321t^4\sin(t)$	$-0.827863t^4\sin(3t)$	$-0.028074t^4\sin(5t)$
$+0.005286t^4\sin(7t)$	$-0.000478t^4\sin(9t)$	$+0.000002t^4\sin(11t)$
$-6.218301t^5\cos(t)$	$+0.995799t^5\cos(3t)$	$+0.007797t^5\cos(5t)$
$-0.003375t^5\cos(7t)$	$+0.000105t^5\cos(9t)$	$-7.652998t^5\sin(t)$
$+0.017785t^5\sin(3t)$	$-0.070424t^5\sin(5t)$	$+0.004954t^5\sin(7t)$
$+0.000022t^5\sin(9t)$	$+1.549574t^6\cos(t)$	$-0.229644t^6\cos(3t)$
$+0.028038t^6\cos(5t)$	$-0.000898t^6\cos(7t)$	$+1.852341t^6\sin(t)$
$+0.213697t^6\sin(3t)$	$-0.008050t^6\sin(5t)$	$-0.001065t^6\sin(7t)$
$-0.230798t^7\cos(t)$	$-0.028635t^7\cos(3t)$	$+0.000743t^7\cos(5t)$
$-0.309074t^7\sin(t)$	$-0.025324t^7\sin(3t)$	$+0.008028t^7\sin(5t)$
$+0.019906t^8\cos(t)$	$+0.004745t^8\cos(3t)$	$+0.017987t^8\sin(t)$
$-0.012705t^8\sin(3t)$	$-0.000299t^9\cos(t)$	$+0.001329t^9\sin(t)$

Table 3: CPU time (in seconds) for the implementation with the `MQSP`, `dsolve`, and `Maple`.

x_n	`MQSP`	`Maple dsolve (using dsolve)`	`Maple`
x_0	0.000187	0.059	0.042
x_1	0.000847	0.131	0.078
x_2	0.004637	0.196	0.147
x_3	0.020543	0.300	0.299
x_4	0.070672	—	0.622
x_5	0.209268	—	1.330
x_6	0.538049	—	2.694
x_7	1.255629	—	5.426
x_8	2.711981	—	10.715
x_9	5.421418	—	21.052
x_{10}	10.367602	—	39.304
x_{11}	18.494896	—	73.471
x_{12}	31.738276	—	—
x_{13}	52.468991	—	—
x_{14}	83.748298	—	—

8. Conclusions

In this paper, we have described an algorithm for the computation of the solution to a perturbed second-order differential equation through the asymptotic expansion technique. This algorithm has been implemented via a symbolic computation system which handles quasipolynomials.

References

[1] H. Poincaré, *Les Méthodes nouvelles de la Mécanique Céleste, Tome I*, Gauthier-Villars, Paris, France, 1892.

[2] J. Henrard, "A survey of Poisson series processors," *Celestial Mechanics*, vol. 45, no. 1–3, pp. 245–253, 1988.

[3] A. Deprit, J. Henrard, and A. Rom, "La théorie de la lune de Delaunay et son prolongement," *Comptes Rendus de l'Academie des Sciences Paris A*, vol. 271, pp. 519–520, 1970.

[4] A. Deprit, J. Henrard, and A. Rom, "Analytical lunar ephemeris: Delaunay's theory," *The Astronomical Journal*, vol. 76, pp. 269–272, 1971.

[5] J. Henrard, "A new solution to the main problem of Lunar theory," *Celestial Mechanics*, vol. 19, no. 4, pp. 337–355, 1979.

[6] M. Chapront-Touzé, "La solution ELP du problème central de la Lune," *Astronomy & Astrophysics*, vol. 83, p. 86, 1980.

[7] M. Chapront-Touzé, "Progress in the analytical theories for the orbital motion of the Moon," *Celestial Mechanics*, vol. 26, no. 1, pp. 53–62, 1982.

[8] J. F. Navarro and J. M. Ferrándiz, "A new symbolic processor for the Earth rotation theory," *Celestial Mechanics & Dynamical Astronomy*, vol. 82, no. 3, pp. 243–263, 2002.

[9] A. Abad, A. Elipe, J. F. San-Juan, and S. Serrano, "Is symbolic integration better than numerical integration in satellite dynamics?" *Applied Mathematics Letters*, vol. 17, no. 1, pp. 59–63, 2004.

[10] V. I. Arnold, *Ordinary Differential Equations*, The Massachusetts Institute of Technology, Cambridge, Mass, USA, 1973.

[11] J. F. Navarro, "On the implementation of the Poincaré-Lindstedt technique," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 183–189, 2008.

[12] J. F. Navarro, "Computation of periodic solutions in perturbed second-order ODEs," *Applied Mathematics and Computation*, vol. 202, no. 1, pp. 171–177, 2008.

[13] N. Minorsky, *Nonlinear Oscillations*, Krieger, Boca Raton, Fla, USA, 1987.

[14] F. San-Juan and A. Abad, "Algebraic and symbolic manipulation of Poisson series," *Journal of Symbolic Computation*, vol. 32, no. 5, pp. 565–572, 2001.

[15] Y. Hardy, K. S. Tan, and W. H. Steeb, *Computer Algebra with SymbolicC++*, World Scientific, Hackensack, NJ, USA, 2008.

[16] B. C. Ikoki, M. J. Richard, M. Bouazara, and S. Datoussaïd, "Symbolic treatment for the equations of motion for rigid multibody systems," *Transactions of the Canadian Society for Mechanical Engineering*, vol. 34, no. 1, pp. 37–55, 2010.

[17] P. Hartman, *Ordinary Differential Equations*, John Wiley & Sons Inc., New York, 1964.