Monografías de la Real Academia de Ciencias de Zaragoza. **28:** 33–42, (2006).

Tratamiento simbólico de los objetos matemáticos de la Astrodinámica

A. Abad, I. Toda y E. Tresaco

Grupo de Mecánica Espacial. Universidad de Zaragoza, 50009, Zaragoza. Spain

Abstract

Languages like C++, based on the object oriented programming, add new possibilities to the creation of more modern and efficient symbolic software. In this paper we present the Lie transformations as a new computational object that complement to the Poisson Series. We characterize, from a computational point of view, not only the canonical and non-canonical Lie transforms, but other auxiliar mathematical objects needed in its implementation.

1 Introducción

La mayor parte de los problemas de la Mecánica Celeste y la Astrodinámica pueden ser expresadas por medio de series de Poisson. De hecho, los procesadores de series de Poisson, en adelante PSP, constituyen la herramienta más importante para la obtención de teorías analiticas simbólicas.

Habitualmente algunos objetos matemáticos, como las transformaciones de Lie, se integran en los procesadores de series de Poisson, programándose sus algoritmos a partir de las técnicas computacionales implementadas para el PSP pero sin diferenciar sus elementos de las series de Poisson y sus propiedades. Las nuevas técnicas de programación orientada a objetos, implementadas en lenguajes de programación como C++, añaden nuevas posibilidades a la creación de software simbólico más eficiente, por medio de la creación de librerias de objetos independientes que aprovechen al máximo las ventajas computacionales generadas de las propiedades particulares de dichos objetos.

En este trabajo se presenta un análisis computacional detallado de las transformaciones de Lie con vistas a su implementación en un futuro software que llamaremos SAT (Symbolic Astrodynamic Tools). Para ello revisaremos algunas de las características computacionales de las series de Poisson e intentaremos extenderlas a las transformaciones de

Lie. De esta forma, nos encontraremos con varios objetos matemáticos diferentes, necesarios para el tratamiento de las transformaciones de Lie y que de acuerdo con esta nueva forma de trabajo deben ser tratados también como objetos independientes.

La idea de la implementación de SAT fue ya analizada en parte en [4], aunque este software no ha sido todavía desarrollado. En este trabajo se utiliza *Mathematica*, en lugar de C++, como campo de trabajo pues únicamente se pretende analizar los objetos y comprobar las dificultades de implementación de sus métodos. Aunque *Mathematica* no constituye una buena herramienta para la programación orientada a los objetos, la hemos elegido por su mayor facilidad de uso. De esta forma hemos creado el paquete `MathSAT`, que constituye un prototipo de SAT, y que puede ser útil para todos aquellos que no precisen toda la potencia de un procesador de series de Poisson y se sientan más cómodos trabajando en el entorno de un procesador simbólico de caracter general. `MathSAT` incluye un procesador de series de Poisson como kernel y varios procesadores más uno para series de potencias, otro para vectores de series de Poisson y series de potencias y el último para transformaciones de Lie. Otra ventaja del uso de `MathSAT` es la de disponer de una herramienta que permita trasladar sin esfuerzo a SAT los algoritmos generados en ella.

2 Series de Poisson–Deprit

La aplicación del Algebra Computacional a la Mecánica Celeste ha estado asociada al uso de sistemas especializados en el tratamiento de las llamadas series de Poisson. El origen de los PSP tiene lugar en el año 1965 cuando Deprit y sus colaboradores, [5], definen las series de Poisson de manera formal y las identifican como el objeto matemático más general, con una estructura algebraica bien definida, que aparece en los problemas de la Mecánica Celeste, Astrodinámica y Dinámica No-Lineal. En el mismo artículo describen el primer procesador de series de Poisson, llamado MAO (Mechanized Algebraic Operations). La influencia que desde ese momento ha tenido A. Deprit en el tratamiento simbólico de los problemas de la Mecánica Celeste nos ha llevado a proponer un cambio de nombre para estas series que de aquí en adelante llamaremos series de Poisson–Deprit (PDS).

Las series de Poisson–Deprit pueden considerarse como series de Fourier multivariadas cuyos coeficientes son series de Laurent multivariadas en la forma

$$\sum_{i_0,\ldots,i_{n-1},j_0,\ldots,j_{m-1}} C_{i_0,\ldots,i_{n-1}}^{j_0,\ldots,j_{m-1}}\, x_0^{i_0}\ldots x_{n-1}^{i_{n-1}} \begin{pmatrix}\sin\\ \cos\end{pmatrix}(j_0y_0+\ldots+j_{m-1}y_{m-1}). \tag{1}$$

La estructura matemática, simbólica y computacional, cuyo conocimiento es imprescindible para una buena implementación de un procesador de series de Poisson, ha sido analizada en detalle en [3]. Basándonos en estas características construimos PSPC, [1], que es un procesador de series de Poisson escrito en lenguaje C mediante el cual se

creó ATESAT [2], una herramienta para obtener automáticamente teorías analíticas del movimiento del satélite artificial.

PSPC parte de la información básica que permite almacenar una PDS en el ordenador y que no es sino un conjunto ordenado de términos cada uno de los cuales se caracteriza por

- Los exponentes de las variables polinómicas $i_0, \ldots, i_{n-1}$ y los coeficientes de las variables trigonométricas $j_0, \ldots, j_{m-1}$.
- El coeficiente $C_{i_0,\ldots,i_{n-1}}^{j_0,\ldots,j_{m-1}}$
- Un indicador que determina si el término es un seno o un coseno.

Puesto que la serie es una sucesión de términos es necesario implementar la misma por medio de una estructura de datos que puede ser una lista unidimensional o bidimensional o bien una estructura matricial. Aunque en principio cualquier estructura es válida el uso de una u otra influye en la implementación de cada algoritmo y en el tiempo de cálculo del mismo, por lo que es preciso un detallado estudio antes de la elección de la estructura de datos. En PSPC se eligió una estructura de lista bidimensional, sin embargo el futuro SAT sustituirá ésta por una estructura matricial de mas fácil acceso a cada elemento.

Una vez definido el objeto computacional es necesario implementar los algoritmos que le doten de la estructura matemática adecuada, en este caso un álgebra de Poisson. De esta forma nos aseguramos una estructura cerrada en la que cada operación nos conduce a un elemento del mismo conjunto y que por tanto el ordenador es capaz de representar.

En el caso de las series de Poisson–Deprit la implementación en C (PSPC) y C++ (SAT) es muy similar con las diferencias derivadas del uso de dos lenguajes de programación diferentes y la introducción de las técnicas de programación orientada a objetos. Sin embargo, la implementación de `MathSAT`, del que pretendemos que sea un clon de SAT, tiene con respecto a los anteriores notables diferencias derivadas del hecho de que *Mathematica* no es un lenguaje adecuado a la programación orientada a objetos y a pesar de ello se ha pretendido emular este *objeto matemático* usando todas las características de *Mathematica*.

No pretendemos aquí describir `MathSAT`, pero puede ser útil dar unas breves ideas de cómo definir e introducir un `objeto` para que *Mathematica* lo identifique como tal y en cuanto lo detecte le aplique todas sus propiedades y algoritmos.

En primer lugar tengamos en cuenta que para almacenar una serie en `MathSAT` no se ha usado la misma estructura de datos que en PSPC, sino que se ha aprovechado la forma en que *Mathematica* almacena estos elementos, que aunque menos eficiente, permite aprovechar todos los algoritmos de *Mathematica* en el momento de implementar las operaciones.

Sin embargo, cuando en *Mathematica* realizamos una operación con dos PDS, por ejemplo el producto de la serie `Sin[x]` por ella misma, el resultado es `Sin[x]^2` que no es formalmente una PDS. Esto es debido a que *Mathematica* no tiene ninguna regla que le obligue pasar a ángulos múltiples esa expresión salvo que formalmente se escriba `Expand[TrigReduce[Sin[x]^2]]`, con lo que obtendremos `1/2 - Cos[2 x]/2`, que si es una serie de Poisson–Deprit.

Así pues la expresión `Sin[x]` no es reconocida como una PDS por *Mathematica*. Para que se reconozca como tal escribiremos `ser = ToPDS[Sin[x]]`, con lo que se almacenará en `ser` la expresión `PDS[1]`, esto es una expresión con cabecera `PDS` y un número entero n, en este caso 1, como argumento. Este número es generado por `MathSAT` e indica el número de orden con el que `MathSAT` identifica la serie que es almacenada en la expresión `PDS$[1]`. De esta forma el elemento de cabecera `PDS` es reconocido de aquí en adelante como serie de Poisson y la operación `ser^2` devuelve otra serie de Poisson `PDS[2]` que representa simbólicamente el resultado `1/2 - Cos[2 x]/2`, que se ha guardado en `PDS$[2]`.

De esta forma el módulo `PDS` de `MathSAT` implementa toda la estructura algebraica de las series de Poisson–Deprit además de otras operaciones importantes en un PSP como son los reemplazamientos de una variables por un número u otra serie y la partición de series en otras según sus propiedades. Asímismo se implementa un sistema automático de borrado de resultados intermedios innecesarios que minimiza el problema de llenado rápido de memoria cuando se trabaja con problemas de orden alto.

3 Transformaciones de Lie

Las teorías analíticas desarrolladas en ATESAT están basadas en el método de Deprit [6] que busca transformaciones canónicas de Lie que simplifiquen el hamiltoniano o lo hagan integrable. Aunque dichas teorías han sido programadas usando PSPC, basado únicamente en series de Poisson–Deprit, el tratamiento dentro del software de dichas transformaciones con un objeto independiente, con sus propiedades y métodos permitirá una simplificación en el uso computacional de métodos como el de Deprit.

Un detallado estudio de las transformaciones de Lie, tanto canónicas como no canónicas, puede verse en [7], [8] y [9]. En este trabajo presentaremos un análisis de las mismas desde el punto de vista computacional y determinamos la necesidad de añadir otros objetos matemáticos necesarios para su implementación.

Una transformación de Lie canónica puede definirse como la solución formal

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y}, \epsilon) = \sum_{i=0}^{n} \frac{\epsilon^i}{i!} \boldsymbol{x}_n(\boldsymbol{y}),$$

de la ecuación diferencial

$$\frac{d\boldsymbol{x}}{d\epsilon} = -\mathcal{J}\nabla_{\boldsymbol{x}} W(\boldsymbol{x}, \epsilon), \quad W(\boldsymbol{x}, \epsilon) = \sum_{i=0}^{n} \frac{\epsilon^i}{i!} W_{n+1}(\boldsymbol{x}),$$

con las condiciones iniciales siguientes: $\boldsymbol{x}(\boldsymbol{y}, \epsilon = 0) = \boldsymbol{y}$.

La caracterización simbólica de la transformación viene dada por el conjunto de símbolos $\boldsymbol{x}, \boldsymbol{y}, \epsilon$, que representan respectivamente las variables canónicas originales, las transformadas y el parámetro de la transformación.

Por otro lado, la función generatriz $W(\boldsymbol{x}, \epsilon)$ contiene toda la información matemática necesaria para realizar las tres operaciones fundamentales de las transformaciones de Lie que constituirá el módulo de software que manipule dicho objeto:

1. Expresar la función $F(\boldsymbol{x}, \epsilon)$ en términos de las variables transformadas, esto es, obtener $F(\boldsymbol{y}, \epsilon)$.

2. Obtener la inversa de una transformación.

3. Componer dos transformaciones.

La función generatriz se representa por una serie de potencias de parámetro ϵ, pero tradicionalmente se ha tratado como una serie de Poisson–Deprit. De esta forma, el parámetro pasa a ser una variable más de la serie y se pierde la ventaja derivada de trabajar directamente con los algoritmos de las series de potencias. Para ver esta ventaja consideremos un problema elemental. Supongamos que trabajamos en orden uno y tenemos una serie de potencias $a + \epsilon b$. Si la serie es representada como una serie de Poisson al elevarla al cuadrado obtendremos la serie $a^2 + \epsilon(2ab) + \epsilon^2 b^2$, de la que deberemos borrar el término $\epsilon^2 b^2$, pues pertenece al orden dos. Sin embargo, trabajando con los algoritmos de las series de potencias dicho término no llega a ser construido, con el consiguiente ahorro de tiempo y memoria.

Así pues, la definición de la estructura de datos que almacena las transformaciones de Lie canónicas exige la implementación de las series de potencias como nuevo objeto. Estas series de potencias pueden representarse como una expresión

$$\mathcal{S} = \sum_{i=0}^{n} \frac{\epsilon^i}{i!} \mathcal{S}_i$$

que queda caracterizada por los siguientes elementos:

- Un símbolo ϵ que representa el parámetro.

- Un número n que representa el orden de la serie.

- $n + 1$ series de Poisson–Deprit $\mathcal{S}_0, \mathcal{S}_1, \ldots \mathcal{S}_n$.

La implementación de las operaciones que determinan la estructura de álgebra de este conjunto de elementos completa el tratamiento computacional del objeto.

Una transformación de Lie no canónica es la solución formal

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{y}, \epsilon) = \sum_{i=0}^{n} \frac{\epsilon^i}{i!} \boldsymbol{x}_n(\boldsymbol{y}),$$

de la ecuación diferencial

$$\frac{d\boldsymbol{x}}{d\epsilon} = \boldsymbol{W}(\boldsymbol{x}, \epsilon) = \sum_{i=0}^{n} \frac{\epsilon^i}{i!} \boldsymbol{W}_{n+1}(\boldsymbol{x}),$$

con las condiciones iniciales siguientes: $\boldsymbol{x}(\boldsymbol{y}, \epsilon = 0) = \boldsymbol{y}$.

Esta definición se diferencia de la correspondiente a una transformación canónica en el carácter vectorial de la función generatriz. Así pues, en este caso debemos añadir un nuevo objeto: los vectores de series de potencias o de series de Poisson–Deprit. No vamos a hablar de este objeto que es considerado en `MathSAT` dentro del módulo `VSAT`, así como las series de potencias lo han sido en el módulo `PWS`.

La estructura de datos que define una transformación de Lie queda finalmente caracterizada por

- El conjunto de símbolos originales $\boldsymbol{x}$ y el de las variables transformadas $\boldsymbol{y}$.
- El símbolo ϵ que representa el parámetro.
- Un vector de series de potencias $\boldsymbol{W}(\boldsymbol{x}, \epsilon)$ para una transformación no canónica.
- Una serie de potencias $W(\boldsymbol{x}, \epsilon)$ para una transformación canónica.

4 Triángulo de Lie

La implementación de objetos matemáticos como las transformaciones de Lie ha hecho aparecer otros objetos como las series de potencias y los vectores de series de Poisson–Deprit o de potencias cuya implementación facilita un adecuado tratamiento simbólico de las transformaciones de Lie. Estos objetos auxiliares pertenecen a una estructura algebraica bien definida que los hace fácilmente identificables, sin embargo una inspección detallada de los algoritmos que tratan las transformaciones de Lie nos permite identificar otro objeto que no es fundamental en este desarrollo pero que ilustra claramente como se puede simplificar el software simbólico descomponiendo algoritmos complejos en operaciones sencillas.

Para ilustrar esto pensemos en el método de Deprit que consiste en buscar una transformación de Lie canónica de generador $W(\vec{x}, \epsilon)$ que transforme el hamiltoniano $\sum_{i=0}^{n} \frac{\epsilon^i}{i!} \mathcal{H}_{i,0}(\vec{x})$, en otro $\sum_{i=0}^{n} \frac{\epsilon^i}{i!} \mathcal{H}_{0,i}(\vec{y})$.

Para ello debemos aplicar el algoritmo del triángulo de Lie basado en la relación

$$\mathcal{H}_{p,q} = \mathcal{H}_{p+1,q-1} + \sum_{k=0}^{p} \binom{p}{k} \left(\mathcal{H}_{p-k,q-1}, W_{k+1}\right). \tag{2}$$

Si reunimos todas las expresiones obtenidas a partir de la relación anterior haciendo $p+q=n$, obtendremos la ecuación homológica

$$\mathcal{H}_{0,n} = \widetilde{\mathcal{H}}_{0,n} + \left(\mathcal{H}_{0,0}, W_n\right),$$

que constituye la base del método de Deprit. El orden n de dicho método puede describirse en tres pasos que se derivan de la ecuación homológica:

1. Obtener el valor del término $\widetilde{\mathcal{H}}_{0,n}$ que reune todos los términos conocidos de las ecuaciones del triángulo de Lie para este orden.

2. Determinar $\mathcal{H}_{0,n}$ a partir de $\widetilde{\mathcal{H}}_{0,n}$ por algún criterio previamente establecido.

3. Encontrar una integral primera W_n de la ecuación en derivadas parciales $\left(\mathcal{H}_{0,0}, W_n\right) = \mathcal{H}_{0,n} - \widetilde{\mathcal{H}}_{0,n}$.

La implementación tradicional de este método consiste en un algoritmo que iterativamente, orden a orden, va realizando las tres operaciones anteriores.

Otra aproximación a éste método puede hacerse definiendo otro objeto, que no representa una estructura algebraica como los anteriores, sino que consiste en un mero recipiente de los elementos que aparecen en este método. Así llamaremos *Triángulo de Lie de orden* n a un conjunto de $(n+2)(n+1)/2$ series de Poisson–Deprit representadas en la figura siguiente

$$\begin{array}{|lllll|}
\hline
\mathcal{H}_{0,0} & \mathcal{H}_{0,1} & \mathcal{H}_{0,2} & \mathcal{H}_{0,3} & \cdots \\
\mathcal{H}_{1,0} & \mathcal{H}_{1,1} & \mathcal{H}_{1,2} & \cdots & \\
\mathcal{H}_{2,0} & \mathcal{H}_{2,1} & \cdots & & \\
\mathcal{H}_{3,0} & \cdots & & & \\
\cdots & & & & \\
\hline
\end{array}$$

La estructura de datos para almacenar un objeto de este tipo consta únicamente de esas $(n+2)(n+1)/2$ series de Poisson–Deprit.

Para implementar un objeto computacional es preciso además identificar las operaciones que se realizarán con dicho objeto. Para ilustrar esto pensemos en un triángulo de Lie de orden 3, formado por tanto por 10 series de Poisson–Deprit. Las operaciones a realizar en el triángulo de Lie se enumeran en los siguientes apartados.

4.1 Creación

La primera operación consiste en la *creación* de un objeto de este tipo para lo cual creamos espacio para 10 PDS vacías o iguales a cero.

0	0	0	0
0	0	0	
0	0		
0			

4.2 Inicialización

La segunda operación, que llamaremos *inicialización*, consiste en llenar la primera columna con los elementos que constituyen los distintos órdenes del hamiltoniano original.

$\mathcal{H}_{0,0}$	0	0	0
$\mathcal{H}_{1,0}$	0	0	
$\mathcal{H}_{2,0}$	0		
$\mathcal{H}_{3,0}$			

4.3 Llenado de la diagonal

Si hemos completado el orden $n-1$ son conocidos los elementos $\mathcal{H}_{p,q}$, $p+q<n$, así como $W_1,\ldots,W_{n-1}$. En nuestro ejemplo supongamos que hemos completado hasta el orden 2. Para completar el orden 3 debemos aplicar sucesivamente la relación (2), esto es calcular $\mathcal{H}_{p,q}$, $p+q=3$ para los valores $p=2,1,0$. Esta operación llena la tercera diagonal, esto es pasa del triángulo de la izquerda de la figura al de la derecha

$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	0		$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	$\mathcal{H}_{0,3}$
$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	0			$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	$\mathcal{H}_{1,2}$	
$\mathcal{H}_{2,0}$	0				$\mathcal{H}_{2,0}$	$\mathcal{H}_{2,1}$		
$\mathcal{H}_{3,0}$					$\mathcal{H}_{3,0}$			
	W_1	W_2	W_3			W_1	W_2	W_3

En el caso anterior se ha supuesto conocido toda la función generatriz, incluida W_3, es decir hemos hecho uso del algoritmo en su forma directa, sin embargo si estamos aplicando el método de Deprit W_3 no es conocido. Para aplicar en este caso el algoritmo supondremos $W_3=0$, con lo que llenamos la diagonal calculando unos elementos $\widetilde{\mathcal{H}}_{p,q}$, $p+q=3$, que contienen toda la información de $\mathcal{H}_{p,q}$ excepto la que depende de W_3 y $\mathcal{H}_{0,3}$. Estos son los elementos que aparecen en la ecuación homológica. De hecho, una vez calculado $\widetilde{\mathcal{H}}_{0,3}$ podemos obtener, por métodos que no dependen de las propiedades del *triángulo de Lie*, las expresiones de $\mathcal{H}_{0,3}$ y W_3.

$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	0	$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	$\widetilde{\mathcal{H}}_{0,3}$
$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	0		$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	$\widetilde{\mathcal{H}}_{1,2}$	
$\mathcal{H}_{2,0}$	0			$\mathcal{H}_{2,0}$	$\widetilde{\mathcal{H}}_{2,1}$		
$\mathcal{H}_{3,0}$				$\mathcal{H}_{3,0}$			
	W_1	W_2	0		W_1	W_2	W_3

4.4 Refresco de la diagonal

Para continuar el orden siguiente es necesario conocer el valor de los elementos $\mathcal{H}_{p,q}$, en lugar de $\widetilde{\mathcal{H}}_{p,q}$ para lo cual basta tener en cuenta que la relación entre ellos vienen dada por $\mathcal{H}_{p,q} = \widetilde{\mathcal{H}}_{p,q} + (\mathcal{H}_{0,0}, W_{p+q})$. En nuestro caso debe sumarse el valor de $(\mathcal{H}_{0,0}, W_3)$ a cada elemento de la diagonal, excepto $\mathcal{H}_{3,0}$, operación que hemos llamado refresco de la diagonal.

$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	$\widetilde{\mathcal{H}}_{0,3}$	$\mathcal{H}_{0,0}$	$\mathcal{H}_{0,1}$	$\mathcal{H}_{0,2}$	$\mathcal{H}_{0,3}$
$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	$\widetilde{\mathcal{H}}_{1,2}$		$\mathcal{H}_{1,0}$	$\mathcal{H}_{1,1}$	$\mathcal{H}_{1,2}$	
$\mathcal{H}_{2,0}$	$\widetilde{\mathcal{H}}_{2,1}$			$\mathcal{H}_{2,0}$	$\mathcal{H}_{2,1}$		
$\mathcal{H}_{3,0}$				$\mathcal{H}_{3,0}$			

4.5 Extracción y borrado

Dos operaciones más completan la implementación de este objeto. En primer lugar la extracción de la primera fila, que contiene los distintos órdenes del hamiltoniano transformado. Finalmente el borrado de este objeto, esto es de las $(n+2)(n+1)/2$ series, para eliminar del ordenador aquellos elementos no necesarios que consumen su memoria.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el proyecto BFM2003-02137 (MCYT).

Referencias

[1] Abad, A. and San Juan, J. F.: 1993, PSPC: A Poisson series processor coded in C, *Dynamics and Astrometry of Natural and Artificial Celestial Bodies, Poznan, Poland*, pp. 383–389.

[2] Abad, A., Elipe, A., Palacián and San Juan, J. F.: 1998, ATESAT: A symbolic Processor for Artificial Satellite Theory *Mathematics and Computers in Simulation*, **45**. pp. 497–510.

[3] Abad, A. and San Juan, J. F.: 2001, Algebraic and symbolic manipulation of Poisson Series *Journal of Symbolic Computation*, **32**. pp. 565-572.

[4] Abad, A. and San Juan, J. F.: 2002, Nuevas perspectivas en el tratamiento de series de Poisson, en *Métodos de Dinámica Orbital y Rotacional*, (Eds. Ferrer, López-Moratalla y Vigueras). Publicaciones de la Universidad de Murcia, pp. 31-38.

[5] Danby,J.M., Deprit, A. y Rom R.M...: 1965, The symbolic manipulation of Poisson Series *BSRL*, note **423**.

[6] Deprit A..: 1969, Canonical transformations depending on a small parameter *Celestial Mechanics*, **1** pp. 12–30.

[7] Henrard, J.: 1970, On a perturbation theory using Lie transforms *Celestial Mechanics*, **3** pp. 107–120.

[8] Kamel, A. A.: 1970, Perturbation method in the theory of nonlinear oscillations *Celestial Mechanics*, **3** pp. 90–106.

[9] Meyer, K.R..: 1991, Lie transform tutorial II *Computer aided proofs in analysis*, Ed. K.R. Meyer and D.S. Schmidt, The IMA Volumes in Mathematics and Its Applications, **28**. Springer-Verlag, New York pp. 190–210.