关于李变换的机器实现

殷志云[1], 熊刚强[2]

(1. 湖南商学院, 湖南 长沙 410205;
2. 东莞市农业银行, 广东 东莞 511700)

摘要: 李变换是一种常用的求奇异摄动问题摄动解的方法. 在求弱非线性系统的有效渐近展开式解时, 利用李级数与李变换能得到一种较为完整和有效的方法. 计算机代数是应用数学的一个边缘学科, 基于采用 Lindstedt-Poincare 方法、多重尺度法、平均化方法自动求解问题, 应用 Mathematica 系统的强大的符号运算功能以及该系统提供的控制语句, 不但实现变量和任意函数到新变量的变换, 而且可以实现李变换及其正则系统的自动求解. 对李变换平均方法 Lie Transform[ ] 函数, 它虽然不能处理非自治问题的摄动问题的自动求解, 但李变换正则系统 Lie Transform[ ] 函数可以处理非自治系统的微分方程问题, 化简微分系统, 降低阶数, 并将该处理过程做成相关的软件包, 简便、实用.

关键词: 李变换; 有效渐近解; 正则系统
中图分类号: TP312 **文献标识码**: A **文章编号**: 1005-9792(2002)03-0317-05

计算机代数是应用数学的一门边缘学科, 现已实现了对 Lindstedt-Poincare 方法、多重尺度法和平均化方法的自动求解问题[1-3]. 在此, 作者借助李级数与李变换方法, 用计算机代数语言在求解弱非线性系统的有效渐近展开式解时, 用机器自动实现了变量和任意函数到新变量的变换, 并通过机器求解一部分摄动系统的有效渐近解.

1 李变换平均方法

考察弱非线性系统

$$\ddot{u}+\omega_0^2 u=\varepsilon f(u,\dot{u}) \tag{1}$$

的有效渐近展开式解, 把方程变成如下标准形式:

$$\dot{x}=f(x,\varepsilon)=\sum_{m=0}^{\infty}\frac{\varepsilon^m}{m!}f_m(x). \tag{2}$$

其中: $f_m(x)=\left.\frac{\partial^m f}{\partial\varepsilon^m}\right|_{\varepsilon=0}$, 这里 x 与 f 是具有 N 个分量的向量. 为分析方便, 引入近似恒等变换:

$$x=x(y,\varepsilon)=y+\varepsilon x_1(y)+\varepsilon^2 x_2(y)+\cdots. \tag{3}$$

式(2)可以变为

$$y=g(y,\varepsilon)=\sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}g_n(y). \tag{4}$$

$$\frac{\mathrm{d}x}{\mathrm{d}\varepsilon}=W(x,\varepsilon). \tag{5}$$

当 $\varepsilon=0$ 时, $x=y$. 把 W 叫做生成向量.

1.1 算法的构造与简化

设式(5)的解是式(3): $x=x(y,\varepsilon)$, 则有反函数, 设其反函数为 $y=y(x,\varepsilon)$, 则

$$\mathrm{d}x=X_y\mathrm{d}y,\ \mathrm{d}y=Y_x\mathrm{d}x. \tag{6}$$

其中: $X_y=\frac{\partial x_i}{\partial y_i}$, $Y_x=\frac{\partial y_i}{\partial x_i}$, 是雅可比矩阵.

式(6)可以写成如下形式:

$$\dot{y}=Y_y f|_{x=x(y,\varepsilon)}. \tag{7}$$

把式(7)的右边展开成 ε 的幂级数形式, 得

$$\dot{y}=\sum_{n=0}^{\infty}\frac{\varepsilon^n \mathrm{d}^n g}{n!\ \mathrm{d}\varepsilon^n}\Big|_{\varepsilon=0}. \tag{8}$$

$$\frac{\mathrm{d}g}{\mathrm{d}\varepsilon}=\left[\frac{\partial}{\partial x}(Y_y f)\right]\frac{\mathrm{d}x}{\mathrm{d}\varepsilon}+\frac{\partial}{\partial\varepsilon}(Y_y f)\Big]_{x=x(y,\varepsilon)}. \tag{9}$$

其中:

$$\frac{\partial}{\partial\varepsilon}(Y_y f)=Y_x\frac{\partial f}{\partial\varepsilon}+\frac{\partial f}{\partial\varepsilon}(Y_x)f=Y_x\frac{\partial f}{\partial\varepsilon}+\frac{\partial}{\partial x}\left(\frac{\partial y}{\partial\varepsilon}\right)f. \tag{10}$$

即

$$\frac{\partial}{\partial\varepsilon}(Y_y f)=Y_x\frac{\partial f}{\partial\varepsilon}-\frac{\partial}{\partial x}(Y_x W)f. \tag{11}$$

收稿日期: 2001-04-08
基金项目: 湖南省普通高校首届学科带头人专项培养基金
作者简介: 殷志云(1960-), 男, 湖南益阳人, 湖南商学院教授, 博士, 主要从事计算机代数与应用的研究.

由式(5)和式(11),将式(9)改写成:

$$\frac{\partial g}{\partial \varepsilon}=\left[\frac{\partial}{\partial x}(\boldsymbol{Y}_x f)\boldsymbol{W}+\boldsymbol{Y}_x\frac{\partial f}{\partial \varepsilon}-\frac{\partial}{\partial x}(\boldsymbol{Y}_x\boldsymbol{W})f\right]_{x=x(y,\varepsilon)}=\left[\boldsymbol{Y}_x Df\right]_{x=x(y,\varepsilon)}. \tag{12}$$

其中: $Df=\dfrac{\partial f}{\partial \varepsilon}+f_x\boldsymbol{W}-\boldsymbol{W}f_x$. 这里 $\boldsymbol{W}$ 是生成向量.

通过递推关系,由式(12)得:

$$\frac{\mathrm{d}^n g}{\mathrm{d}\varepsilon^n}=\left[\boldsymbol{Y}_x D^n f\right]_{x=x(y,\varepsilon)}. \tag{13}$$

这里 $\boldsymbol{Y}_x|_{\varepsilon=0}=I$. 得

$$\frac{\mathrm{d}^n g}{\mathrm{d}\varepsilon^n}\Big|_{\varepsilon=0}=D^n f|_{x=y,\varepsilon=0}. \tag{14}$$

如果 $\boldsymbol{W}$ 可展开成

$$\boldsymbol{W}=\sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}\boldsymbol{W}_{n+1}. \tag{15}$$

则由式(3)和式(15),得:

$$Df=\sum_{k=0}^{\infty}\frac{\varepsilon^k}{k!}f_k^{(1)}. \tag{16}$$

式中:
$$f_k^{(1)}=f_{k+1}+\sum_{m=0}^{\infty}C_k^m L_{m+1}f_{k-m}; \tag{17}$$

$$L_m g=\frac{\partial g}{\partial x}\boldsymbol{W}_m-\frac{\partial \boldsymbol{W}_m}{\partial x}g. \tag{18}$$

通过递堆,由式(16),(17),(18),得

$$D^n f=\sum_{k=0}^{\infty}\frac{\varepsilon^k}{k!}f_k^{(n)}. \tag{19}$$

其中:

$$f_k^{(n)}=f_{k+1}^{(n-1)}+\sum_{m=0}^{k}C_k^m L_{m+1}f_{k-m}^{(n-1)},\ (f_k^{(0)}=f_k). \tag{20}$$

于是,

$$\dot{y}=\sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}f^{(n)},\ f^{(n)}=f_0^{(n)}|_{x=y}, \tag{21}$$

$$\frac{\mathrm{d}^n E}{\mathrm{d}\varepsilon^n}=\sum_{k=0}^{\infty}\frac{\varepsilon^k}{k!}F_k^{(n)},\quad F(x,\varepsilon)=\sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}\frac{\partial^n E}{\partial \varepsilon^n}\Big|_{\varepsilon=0}. \tag{22}$$

为了简化李变换算法,需要把式(20)改写成

$$f_k^{(n)}=f_{k-1}^{(n+1)}-\sum_{m=0}^{k-1}C_{k-1}^m L_{m+1}f_{k-m-1}^{(n)}. \tag{23}$$

逐次消去右边的函数,便可以得到如下表达式:

$$f_k^{(n)}=f^{(n+k)}-\sum_{j=1}^{k}C_j^k G_j f^{(n+k-j)}. \tag{24}$$

式中: G_j 是一个线性算子,它是 $L_j, L_{j-1}, \cdots, L_1$ 的函数,且有如下的递推关系:

$$G_j=L_j-\sum_{m=1}^{j-1}C_{j-1}^{m-1}L_m G_{j-m},\quad 1\leqslant j\leqslant n. \tag{25}$$

由式(23)可得:

$$\begin{cases} f^{(k)}=f_k+\sum_{j=1}^{k}C_j^k f_{j,k-j}, & n=0\text{ 时;}\\ f_k^{(1)}=f^{(k+1)}-\sum_{j=1}^{k}C_j^j f_{j,k-j+1}, & n=1\text{ 时.}\end{cases} \tag{26}$$

其中: $f_{j,i}=G_j f^{(i)}=L_j F^{(i)}-\sum_{m=1}^{j-1}C_{j-1}^{m-1}L_m f_{j-m,i}$

Kamel A A 给出了更方便的形式[4]:

$$f^{(k)}=f_k+\sum_{j=1}^{k-1}\left[C_{k-1}^{j-1}L_j f_{k-j}+C_j^{k-1}f_{j,k-j}\right]+L_k f_0. \tag{27}$$

根据上述推导方法,构造 $x=y+\sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}x^{(n)}(y)$ 的算法,可得如下关系式:

$$\begin{cases} x^{(k)}=\boldsymbol{W}_k+\sum_{j=1}^{k-1}C_{k-1}^j x_{j,k-j},\\ x_{j,i}=\pounds_j x^{(i)}-\sum_{m=1}^{j-1}C_{j-1}^{m-1}\pounds_m x_{j-m,i}.\end{cases} \tag{28}$$

其中: $k\geqslant 1$.

1.2 算法的实现

用李变换平均方法求方程式(1)的摄动解,关键是构造近似恒等变换式(3)与(2)的简单形式(4),求出李变换表达式(15),在自动求解过程中实现 3 个变换过程和 1 个分离长周期项与短周期项的过程. 即用式(18)或式(26)和式(27),求出式(4)与式(15);用式(28)求出式(3);用式(22)求出其有效渐近解.

下面是李变换平均方法在机器上运算的主要过程:

a. 输入函数表达式 $f[u[t], u'[t]]$;

b. 用标准形式的微分方程组生成变量表,调用子函数 Parameterpartition[ ];

c. 生成分量表 flist= $\{f_0, f_1, \cdots, f_n\}$, f_0 为 ε 的 0 次项系数, f_1 为 ε 的一次项系数,依此类推;

d. 生成第二个分量表 glist(表示式(4)中的每一个分量), wlist(表示式(15)中的每一个分量);

e. 调用 Transform$G$[ ],逐个求出 g_k 与 $\boldsymbol{W}_k$;

f. 调用 Transform$F$[ ]产生有效渐近展开式解的每个分量;

g. 调用 Inverse Trans$X$[ ],求出变量的近似恒等变换与近似恒等的逆变换;

h. 输出结果,结果中包含有渐近展开式解,李变换的具体表达式,变量 x 到 y 的近似恒等变换及逆变换.

值得注意的是:在调用子函数中,进行参数变易法,采用 Parameterpartition[ ] 函数,而 CF[ ] 函数用于

求组合数, *LF*[ ] 函数的目的是实现算子: $L\frac{\partial F}{\partial x}G-\frac{\partial G}{\partial x}F$. 要实现李变换平均方法的自动运算, 还要用到 Jaccob[ ] 函数, 即用于求函数向量的 Jaccob 矩阵. 而 *FF*[ ] 函数实现 $f_{j,k}=L_jg_k-\sum_{m=1}^{j-1}C_{j-1}^{m-1}L_mf_{j-m,k}$ 的符号求值过程, *GF*[ ] 函数实现 $g_i=f_i+\sum_{j=1}^{i}C_i^jf_{j,i-j}$ 的符号求值过程. 用 *TG*[ ] 函数进行迭代求解, *LF*1[ ] 函数实现算子 £, *TF*[ ] 函数得到渐近展开式解, *TX*[ ] 函数得到变量 x 到 y 的近似恒等变换, 而主函数为 *LT*[ ] 函数. 输出结果是一个表. 第一项为李变换表达式, 第二项即近似恒等变换, 其次分别是变换后的微分系统和渐近展开式的解.

作为例子, 求 Van der pol 方程的二阶渐近展开式解.

```
In[1]:=
LieTransform[(1-u[t]^2)*u'[t],u[t],t,l,e,l]
Out[1]=
{{W[1]'[e]->e*((-7*a^3*cos[2*phi])/64+(3*a^5*cos[2*phi])/128+(a^3*cos[4*phi])/64
+(a^5*cos[6*phi])/384)-(a*sin[2*phi])/4+(a^3*sin[4*phi])/32;
W[2]'[e]->-cos[2*phi]/4+(a^2*cos[2*phi])/8+(a^2*cos[4*phi])/32+e*(-(a^2*sin
[2*phi])/32-(a^4*sin[2*phi])/64-(a^2*sin[4*phi])/64+(a^4*sin[4*phi])/256-(a^4*sin[6*
phi])/384)};
{a->(aa*(384-42*aa^2*e^2*cos[2*psi]+9*aa^4*e^2*cos[2*psi]+6*aa^2*e^2*cos[4*psi]
+aa^4*e^2*cos[6*psi]-96*e*sin[2*psi]+12*aa^2*e*sin[4*psi]))/384,
phi->psi-(e*cos[2*psi])/4+(aa^2*e*cos[2*psi])/8+(aa^2*e*cos[4*psi])/32-(aa^2*e^2*
sin[2*psi])/32-(aa^4*e^2*sin[2*psi])/64-(aa^2*e^2*sin[4*psi])/64+(aa^4*e^2*sin[4*psi])/
256-(aa^4*e^2*sin[6*psi])/384};
{a'->e*(a/2-a^3/8-(a*cos[2*phi])/2+(a^3*cos[4*phi])/8),phi'->1+e*(sin[2*phi]/2-
(a^2*sin[2*phi])/4-(a^2*sin[4*phi])/8)};
{aa'->(aa/2-aa^3/8)*e,psi'->1+((-32+48*aa^2-11*aa^4)*e^2)/256};
{u->aa*cos[psi]+(aa*e^2*cos[psi])/16-(aa^3*e^2*cos[psi])/16+(15*aa^5*e^2*cos[psi])/
1024-(aa^3*e^2*cos[3*psi])/16+(5*aa^5*e^2*cos[3*psi])/512-(5*aa^5*e^2*cos[5*psi])/
1536-(aa*e*sin[psi])/4+(aa^3*e*sin[psi]/16-(aa^3*e*sin[3*psi])/32}}.
```

在 CPU: INTEL (R) CELERON (TM)-MMX. 266 MHz, 内存为 64M 的微机上, LieTransform[ ] 函数的运行情况是 6 S, 在内存稍高的微机上运行时间不超过 4 s, 且结果与文献[5]所报道的相符.

2　李变换正则系统

用李变换化简哈密尔顿系统, 主要思路是找一个变量变换, 在新的变量之下, 使其变得更为简单.

先考察下面形式的哈密尔顿函数:

$$H(q_1,p_1)=\frac{1}{2}(p_1^2+\omega_0^2q_1^2). \tag{29}$$

对应方程为

$$\begin{cases}\dfrac{\mathrm{d}q_1}{\mathrm{d}t}=p_1,\\[2ex]\dfrac{\mathrm{d}p_1}{\mathrm{d}t}=-\omega_0^2q_1.\end{cases} \tag{30}$$

设有 2 个函数 $f(Q_i,P_i)$, $g(Q_i,P_i)$, 则 f 与 g 的泊松括号式(Poisson bracket) 为

$$\{f,g\}=\sum_{i=1}^{n}\left(\frac{\partial f}{\partial Q_i}\frac{\partial g}{\partial P_i}-\frac{\partial f}{\partial P_i}\frac{\partial g}{\partial Q_i}\right). \tag{31}$$

定义 2 个算子 L_n 与 S_n:

$$L_n=\{\boldsymbol{W}_n,H_{n-1}\}, \tag{32}$$

$$S_n=\frac{1}{n}\sum_{m=0}^{n-1}L_{n-m}S_{m,n}. \tag{33}$$

$n=1,2,3,\cdots$, 当 $n=0$ 时, S_0 为恒等算子.

显然, 有如下等式成立:

$$q_i=[S_0+\varepsilon S_1+\varepsilon^2S_2+\cdots]Q_i; \tag{34}$$

$$p_i=[S_0+\varepsilon S_1+\varepsilon^2S_2+\cdots]P_i; \tag{35}$$

$$k_0=H_0; \tag{36}$$

$$k_1=H_1+\frac{\partial\boldsymbol{W}_1}{\partial t}+\{\boldsymbol{W}_1,H_0]; \tag{37}$$

$$k_n=H_n+\frac{1}{n}\left[\frac{\partial\boldsymbol{W}_n}{\partial t}+\{\boldsymbol{W}_n,H_0\}\right]+\frac{1}{n}\sum_{m=1}^{n-1}[L_{n-m}k_m+mS_{n-m}H_m]. \tag{38}$$

对一般弱非线性系统式(1), 当 $\varepsilon=0$ 时, 其哈密尔顿函数形式如式(29). 引入 action-angle 变量, 则

$$\begin{cases}q_i=\sqrt{2J_1/\omega_0}\sin\varphi_1,\\P_i=\sqrt{2J_1/\omega_0}\cos\varphi_1.\end{cases} \tag{39}$$

这时, 式(29)变为 $H(\varphi_1, J_1) = \omega_0 J_1$. (40)

从式(31)至式(38), 用 Mathematica 系统较容易实现:

a. 输入 Horiginal, Plist, Qlist, N, Ntrunc, T, e;

b. 生成 Hlist, Klist, Jlist, Philist, Ilist, Psilist 以及 Wgenlist;

c. 调用 Poissonbracket[ ]、L[ ]、S[ ], 计算出式(31)至式(33)的结果;

d. 调用 EQ[ ], 计算式(38), 分离出长短周期项, 使 W_n 取短周期项, 简化 K_n;

e. 调用 main Transform[ ]进行迭代求解, 并返回 W_n 与 K_n 的值;

f. 化简哈密尔顿系统, 生成函数的表达式以及变量变换的表达式.

以 Duffing 方程[1]为例, 令 $P = \dot{u}$, $q = u$, 相应地有哈密尔顿函数 $H = \frac{1}{2}p^2 + \frac{1}{2}q^2 + \varepsilon\frac{q^4}{4}$, 调用 LieHamilton[ ]函数.

```
In[1]:=
LieHamilton[(p^2+ q^2)/2+ e* q^4/4, {p}, {q}, 1, 1, t, e]
out[1]=
{{k-> I1+ (3* e* I1^2)/8}, {wgen[1]-> (I1^2* sin[2* psi1])/4(I1^2* sin[4* psi1])/32},
{J1-> I1+ e* (I1^2* cos[2* psi1])/2-(I1^2* cos[4* psi1])/8},
{phil-> psil+ e* (-(I1* sin[2* psil])/2+ (I1* sin[4* psil])/16)}}.
```

在 PENTIUM-S CPU 100 MHz, 内存为 32M 的微机上的运行时间为 3 s, 运算结果与文献[6]相符.

对一个含有 2 个自由度的哈密尔顿系统

$$H = \frac{1}{2}p_1^2 + \frac{1}{2}q_1^2 + \frac{1}{1}p_2^2 + \frac{1}{2}q_2^2 + \varepsilon\frac{K}{4}[q_1^4 + q_2^4] + \frac{\varepsilon}{4}[q_1 - q_2]^4. \quad (41)$$

调用 LieHamilton[ ]函数, 微机运行时间为14 s.

对非自治系统的哈密尔顿系统

$$\ddot{u} + (\delta + e\cos t)u + e\alpha u^3 = 0. \quad (e \text{ 为小参数}) \quad (42)$$

```
In[2]:=
LieHamilton[P1^2/2+ q1^2/2+ p2^2/2+ q2^2/2+ e* k* (q1^4+ q2^4)/4+ e* (q1- q2)^4/4, {p1, p2}, {q1, q2}, 2, 1, t, e];
out[2]=
{{K-> I1+ I2+ (3* e* (I1^2+ 4* I1* I2+ I2^2+ I1^2* k+ I2^2* K- 4* I1^(3/2)* I2^(1/2)* cos[psi1- psi2]- 4* I1^(1/2)* I2^(3/2)* cos[psi1- psi2]+ 2* I1* I2* cos[2* (psi1- psi2)]))/8};
{wgen[1]-> (I1^2* sin[2* psi1])/4+ (3* I1* I2* sin[2* psi1])/4+
(I1^2* k* sin[2* psi1])/4- (I1^2* sin[4* psi1])/32
- (I1^2* k* sin[4* psi1])/32+ (I1^(1/2)* I2^(3/2)* sin[psi1- 3* psi2)]/4-
(I1^(3/2)* I2^(1/2)* sin[2* psi1- 2* psi2])/4-
(I1^(3/2)* I2^(1/2* sin[3* psi1- psi2])/4+ (3* I1* I2* sin[2* psi2])/4+ (I2^2* sin[2* psi2])/4+
(I2^2* k* sin[2* psi2])/4- (I2^2* sin[4* psi2])/32
-
(I2^2* k* sin[4* psi2])/32- (3* I1^(3/2)* I2^(1/2)* sin[psi1+ psi2)/4]-
(3* I1^(1/2)* I2^(3/2)* sin[psi1+ psi2])/4+
(I1^(3/2)* I2^(1/2)* sin[3* Psi1+ psi2])/8-
(3* I1* I2* sin[2* psi1+ 2* psi2])/16+
(I1^(1/2)* I2^(3/2)* sin[psi1+ 3* psi2])/8},
{J1-> I1+ e* ((I1^2* cos[2* psi1])/2+ (3* I1* I2* cos[2* psi1])/2+
(I1^2* k* cos[2* psi1])/2- (I1^2* cos[4* psi1])/8
- (I1^2* k* cos[4* psi1])/8+
(I1^(1/2)* I2^(3/2)* cos[psi1- 3* psi2])/4-
(I1^(3/2)* I2^(1/2)* cos[2* psi1- 2* psi2])/2-
(I1^(3/2)* I2^(1/2)* cos[psi1- psi2])/4-
(3* I1^(3/2)* I2^(1/2)* cos[3* psi1- psi2])/4-
(3* I1^(3/2)* I2^(1/2)* cos[psi1+ psi2])/4-
(3* I1^(3/2)* I2^(1/2)* cos[3* psi1+ psi2])/8-
(3* I1* I2* cos[2* psi1+ 2* psi2])/8+
(I1^(1/2)* I2^(3/2)* cos[psi1+ 3* psi2])/8);
{J2-> I2+ e* ((-3* I1^(1/2)* I2^(3/2)* cos[psi1- 3* psi2])/4+
(I1^(3/2)* I2^(1/2)* cos[2* psi1- 2* psi2])/2+
(I1^(3/2)* I2^(1/2)* cos[psi1- psi2])/4+
(I1^(3/2)* I2^(1/2)* cos[3•psi1- psi2])/4+
(3* I1* I2* cos[2* psi2])/2+ (I2^2* cos[2* psi2])/2+
(I2^2* k* cos[2* psi2])/2- (I2^2* cos[4* psi2])/8
- (I2^2* k* cos[4* psi2])/8-
(3* I1^(3/2)* I2^(1/2)* cos[psi1+ psi2])/4-
(3* I1^(3/2)* I2^(/1)* cos[psi1+ psi2])/4+
(I1^(3/2)* I2^(1/2)* cos[3* psi1+ psi2])/8-(3* I1* I2* cos[2* psi1+ 2* psi2])/8+(3* I1^(1/2)*
```

```
I2^(3/2) * cos[psi1+ 3* psi2])/8},
{phi1-> psi1+ e* (-(I1* sin[2* psi1])/2-(3*
I2* sin[2* psi1])/4-
(I1* k* sin[2* psi1])/2+(I1* sin[4* psi1])/16+
(I1* k* sin[4* psi1])/16-
(I2^(3/2) * sin[psi1-3* psi2])/(8* I1^(1/2))+
(3* I1^(1/2) * I2^(1/2) * sin[2* psi1-2* psi2])/8
+
(3* I1^(1/2) * I2^(1/2) * sin[psi1- psi2])/8+
(3* I1^(1/2) * I2^(1/2) * sin[3* psi1- psi2])/8-
(3* I2* sin[2* psi2])/4+
(9* I1^(1/2) * I2^(1/2) * sin[psi1+ psi2])/8+
(3* I2^(3/2) * sin[psi1+ psi2])/(8* I1^(1/2))-
(3* I1^(1/2) * I2^(1/2) * sin[3* psi1+ psi2])/16+
(3* I2* sin[2* psi1+ 2* psi2])/16-
(I2^(3/2) * sin[psi1+ 3* psi2])/(16* I1^(1/2))),
phi2-> psi2+ e* ((-3* I1* sin[2* psi1])/4-
(3* I1^(1/2) * I2^(1/2) * sin[psi1-3* psi2])/8+
(I1* (3/2) * sin[2* psi1-2* psi2])/(8* I2^(1/2))
+
(I1* (3/2) * sin[psi1- psi2])/(8* I2^(1/2))+
(I1* (3/2) * sin[3* psi1- psi2])/(8* I2^(1/2))-
(3* I1* sin[2* psi2])/4-((I2* sin[2* psi2]/2-
(I2* k* sin[2* psi2])/2+(I2* sin[4* psi2])/16+
(I2* k* sin[4* psi2]/16+(3* I1^(3/2) * sin[psi1+
psi2])/(8* I2^(1/2)+
(9* I1^(1/2) * I2^(1/2) * sin[psi1+ psi2])/8-
(I1* (3/2) * sin[3* psi1+ psi2])/(16* I2^(1/2))+
(3* I1* sin[2* psi1+ 2* psi2])/16-
(3* I1^(1/2) * I2^(1/2) * sin[psi1+ 3* psi2])/
16)}}.
```

在微机上运行时间为 27 s, 结果与手工计算结果相符.

李变换正则系统 Lie Transform[ ] 函数可以处理非自治系统的问题, 其主要目的是化简微分系统. 但它不能处理非自治系统的摄动问题的自动求解, 因而 Lie Transform[ ] 函数需要进一步改进.

参考文献:

[1] 殷志云, 熊刚强. Lindstedt Poincare 方法的机器实现[J]. 益阳师专学报, 2000, 17(6): 1-6.

[2] 殷志云, 熊刚强. 多重尺度法的机器实现[J]. 湘潭师范学院学报(自然科学版), 2001, 16(2): 27-31.

[3] 殷志云, 熊刚强. 平均化方法的机器实现[J]. 湘潭师范学院学报(自然科学版), 2002, 22(2): 39-43.

[4] Kamel A A. Lie transforms and the Hamiltonization of non-Hamiltonian systems[J]. Celestial Mech, 1971(4): 397-405.

[5] Naytch A H. 王辅俊, 徐钧涛译. 摄动方法[M]. 上海: 上海科技出版社, 1984: 70-95.

[6] Richard H. Topics in nonlinear dynamics with computer algebras[M]. 1994: 40-89.

[7] 裘宗燕. Mathematica 数学软件系统的应用及其程序设计[M]. 北京: 北京大学出版社, 1994: 46-81.

On automatical solutions of the Lie transformation

YIN Zhi-yun[1], XIONG Gang-qiang[2]

(1. Hunan Business College, Changsha 410205, China;
2. Agricultural Bank of Dongguan City, Dongguan 511700, China)

Abstract: Lie transfromation is a usual method of the perturbed solution that has solved the singular perturbed problems. By using Lie series and Lie transformation an effective technique is obtained on some solutions of effectively asymptotic expansions to a weakly nonlinear system in this paper, and some automatically solving problems of the Lie transformation are considered which would use powerful symbolic operation and control sentence provided by mathematica system. Based on automatical solutions of Lindstedt-Poincare method, multiple scale method and averaging method, Lie transformation[ ] function of the regular system of Lie transformation can treat the problems from higher dimensions to lower dimensions of the ordinary differential equations. The authors compile these programs and integrated into packages, which can be expanded, and is considered as teaching softwave about perturbation method.

Key words: Lie transformation; effectively asymptotic solutions; regular system