应用数学和力学编委会编
重 庆 出 版 社 出 版

偏微分方程组的 Lie 群与高阶对称群的 Taylor 多项式逐步精化算法*

张鸿庆① 朝 鲁② 唐立民③

(1996 年 7 月 5 日收到, 1997 年 10 月 5 日收到修改稿)

摘 要

本文基于生成函数的 Taylor 展开式及逐步简化步骤, 提出了计算偏微分方程组的 Lie 群与高阶对称群的 Taylor 多项式算法, 把标准算法中的求解超定偏微分方程组的问题转化为求解代数方程组的问题, 降低了求解的难度, 提高了计算效率, 并且易用计算机代数系统在计算机上全过程实现, 并得到重要的对称群•

关键词 Lie 群 高阶对称群 Taylor 多项式 计算机代数 确定方程 逐步精化法
中图分类号 O175

§1. 引 言

偏微分方程组的对称群在数学物理及力学方程的研究中有着重要的应用[1], 如: 1. 从已知解求得新的解• 把偏微分方程的对称群作用在已知的解上, 产生一组新的解• 2. 守恒律的构造• 利用对称群可以构造在数学物理中非常重要的许多守恒律• 3. 偏微分方程的约化• 对称群可用来减少偏微分方程组的未知函数及自变量的个数, 如, 把具有一个未知函数, 两个自变量的方程降为常微分方程• 4. 偏微分方程的分类, 具有同样对称群的偏微分方程(组)分为等价类[6], 便宜统一研究• 5. 偏微分方程的解的渐进性• 因为偏微分方程组的解渐进趋向于由对称群约化而得到的低维方程组的解, 从而用对称群而得到的特解能够解释重要的物理现象, 如解的渐进性及 Blow_up 性等问题[2]• 6. 数值算法的检测• 偏微分方程对称群可用来设计求解偏微分方程数值解的有效算法[3], 也可用对称精确解来检验一些数值算法的正确性和可靠性•

对称群虽有众多的理论和实际的重要性, 但其繁重的计算量及最后求解超定偏微分方程组—确定方程等问题的困难, 严重限制了其应用和发展• 八十年代中期开始, 随着计算机速度的提高, 存储量的增大及强有力的计算机代数系统(符号运算)的实现和完善, 对称群的计算问题得到部分解决, 但本质上的困难尚未得到克服• 对偏微分方程求解其对称群主要有两个步

* 国家自然科学基金及数学机械化中心资助项目

① 大连理工大学数学所, 大连 116024
② 内蒙古工业大学, 呼和浩特 010062
③ 大连理工大学力学系, 大连 116024

骤: 1. 产生生成函数满足的超定偏微分方程组, 即所为确定方程组; 2. 求解确定方程组• 标准的产生确定方程组的方法是下述的无穷小变换方法(细节见[1])•

设

$$(\Delta)\quad \begin{cases} \Delta_1(x, u, u^{(\alpha)}) = 0 \\ \qquad\vdots \qquad (|\alpha| \leqslant k) \\ \Delta_l(x, u, u^{(\alpha)}) = 0 \end{cases}$$

为 k 阶偏微分方程组• 其中 $x = (x_1, x_2, \cdots, x_p)$ 表示自变量, $u = (u_1, u_2, \cdots, u_q)$ 表示未知函数, $\alpha = (\alpha_1, \alpha_2, \cdots \alpha_p)$ 为多重指标, 且 $|\alpha| = \sum_{i=1}^{p}\alpha_i$, $u^{(\alpha)}$ 表示 u 的所有导数 $u_{i,\alpha} = \partial^{|\alpha|} u_i / \partial x_1^{\alpha_1}\partial x_2^{\alpha_2}\cdots\partial x_p^{\alpha_p}$ $(1 \leqslant |\alpha| \leqslant k, i = 1, 2, \cdots, k)$ 构成的集合•

我们求(Δ)的对称群的如下不变向量场

$$X = \sum_{i=1}^{p}\xi_i\partial_{x_i} + \sum_{j=1}^{q}\phi_j\partial_{u_j} \tag{1.1}$$

其中, ξ_i, ϕ_j $(i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 称为生成函数, 当求(Δ)的 Lie 群时, 它们依赖 x, u, 当求高阶对称群时依赖 x, u 和 $u^{(\alpha)}$•

X 成为(Δ)的不变向量场的充分必要条件是无穷小方程组

$$P_r^{(k)}X(\Delta)\,|_{\Delta=0} = 0 \tag{1.2}$$

成立, 其中 $P_r^{(k)}X$ 表示 X 的 k 阶延拓[1]•

取(1.2)中独立变量 $u_{i,\alpha}^{n_\alpha}u_{j,\beta}^{n_\beta}\cdots u_{k,\gamma}^{n_\gamma}$ 的系数为零, 即得到 ξ_i, ϕ_j 满足的超定偏微分方程组一确定方程组• 虽然以上计算过程是完全代数化的, 但它涉及到极繁琐的代数及多元函数求高阶全导数等众多的运算, 使它难于人工处理• 为此已发展了有关的软件[4], 计算机自动产生确定方程• 对确定方程也有些处理方法[1]、[5]、[6]• 但均不能彻底求解, 最后阶段上仍剩有"规模"不小的超定偏微分方程组只有人工求解, 而求解超定偏微分方程组的困难是众所周知的• 所以改进这些算法和发现有效的新算法是目前对称群理论得以应用和发展所面临的最重要的问题• 现有的算法都求解给定偏微分方程组全体对称群为目标的一般性方法, 其工作量和难度较大• 但事实上在应用中偏微分方程组的某些非平凡对称群在研究, 如方程的约化, 构造守恒律等时起着重要的作用• 故从应用的角度上来讲寻找非平凡特殊对称群是非常重要的• 据笔者所知还没有计算特殊对称群的有效而实用算法•

本文中作者基于以上理由及绝大多数偏微分方程组的对称群的生成函数为其自变量的多项式这个事实, 提出了一个求解对称群的 Taylor 多项式算法• 其中采取了逐步精简等措施, 有效地控制了中间表达使迅速膨胀, 一定程度上克服了符号运算面临的难题• 由于此算法中 ξ_i, ϕ_j 直接与多项式形式参与符号计算, 所以宜易用计算机代数在计算机上全过程实现, 即一般方法中的人工去求解超定偏微分方程组的问题转化为用计算机解代数方程组的问题, 自动得到最后结果•

§2. 算法的叙述及逐步精化法

我们的算法是非常直观的, 就是假设(1.1)中的生成函数 ξ_i, ϕ_j $(i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 有下面的 Taylor 多项式形式展开

$$(\mathrm{CS})\quad \begin{cases} \xi_i(x,u,v)=\displaystyle\sum_{\substack{i_1,i_2,\cdots,i_p\\ j_1,j_2,\cdots,j_q\\ k_1,k_2,\cdots,k_s}}^{M} a^{i}_{i_1i_2\cdots i_pj_1j_2\cdots j_qk_1k_2\cdots k_s}x_1^{i_1}x_2^{i_2}\cdots x_p^{i_p}u_1^{j_1}u_2^{j_2}\cdots u_q^{j_q}v_1^{k_1}v_2^{k_2}\cdots v_s^{k_s} \\ \phi_j(x,u,v)=\displaystyle\sum_{\substack{i_1,i_2,\cdots,i_p\\ j_1,j_2,\cdots,j_q\\ k_1,k_2,\cdots,k_s}}^{M} b^{j}_{i_1i_2\cdots i_pj_1j_2\cdots j_qk_1k_2\cdots k_s}x_1^{i_1}x_2^{i_2}\cdots x_p^{i_p}u_1^{j_1}u_2^{j_2}\cdots u_q^{j_q}v_1^{k_1}v_2^{k_2}\cdots v_s^{k_s} \\ i=1,2,\cdots,p;\ j=1,2,\cdots,q \end{cases}$$

及结合以下叙述的逐步精化策略• 其中 M 是适当选择的正整数, v 代表 u 的导数项, 对求解 Lie 群的情形(CS) 中 v 项不出现, 即 $k_i=0(i=1,2,\cdots,s)$, $a^{i}_{i_1i_2\cdots k_s}$ 和 $b^{j}_{i_1i_2\cdots k_s}$ 为常数•

由以上假设, (1.2)产生的确定方程将成为对 x, u 和 v 恒成立的多项式方程组, 进而得到 $a^{i}_{i_1i_2\cdots k_s}$ 及 $b^{j}_{i_1i_2\cdots k_s}$ 满足的超定代数方程组, 我们称它为代数确定方程组• 这样, 标准方法中解超定偏微分方程组的问题转化为求解代数方程组的问题, 大大降低了求解的难度•

实现算法时, 我们采取了以下几个策略:

1. 逐步精化法• 由内存及中间表达式“爆炸”(符号运算面临的困难之一)等问题造成确定方程不易一次求出时, 可以采用下述的逐步精化法• 若系统较大, 把系统分解成较小部分, 再对每个部分先确定某些特定的“优先”确定方程, 如最高阶导数项对应的确定方程, 由此求得 $a^{i}_{i_1i_2\cdots k_s}$ 与 $b^{j}_{i_1i_2\cdots k_s}$ 的某些关系, 如许多 $a^{i}_{i_1i_2\cdots k_s}$ 或 $b^{j}_{i_1i_2\cdots k_s}$ 为零, 利用此信息简化 ξ_i, $\phi_j(i=1,\cdots,p;j=1,\cdots,q)$ 的表达式, 再计算下一个“优先”确定方程, 进一步得到 $a^{i}_{i_1i_2\cdots k_s}$ 与 $b^{j}_{i_1i_2\cdots k_s}$ 的关系, 再次简化 ξ_i, ϕ_j• 如此下去经有限步之后, 最终将得到 $a^{i}_{i_1i_2\cdots k_s}$, $b^{j}_{i_1i_2\cdots k_s}$ 满足的代数确定方程•

2. 用特征函数• 在计算高阶对称群时, 由于(1.1)与其特征形式的发展向量场(见[1]):

$$(\mathrm{CH})\qquad V_Q=\sum_{i=1}^{q}Q_i(x,u,v)\partial_{u_i}$$

的等价性, 我们选 Q_i 的 Taylor 多项式形式

$$Q_i(x,u,v)=\sum_{\substack{i_1,i_2,\cdots,i_p\\ j_1,j_2,\cdots,j_q\\ k_1,k_2,\cdots,k_s}}^{M} c_{i_1i_2\cdots i_pj_1j_2\cdots j_qk_1k_2\cdots k_s}x_1^{i_1}x_2^{i_2}\cdots x_p^{i_p}u_1^{j_1}u_2^{j_2}\cdots u_q^{j_q}v_1^{k_1}v_2^{k_2}\cdots v_s^{k_s}$$

及上述逐步法来计算, 将提高计算速度和节省内存• 其中 $c^{i}_{i_1i_2\cdots k_s}$ 是常数•

3. 整数 M• M 的选择依赖于方程 Δ 的阶数 k(如, $M=k+1$)或根据计算环境等情况来确定• 因为算法的计算量随 M 的增大而增加•

4. 优化群参数• 由于我们的假设(CS) 或(CH) 非一般情况, 所以 X 中可能含有平凡的对称群• 必要时可以去掉它, 使群参数变的较少• 确认 X 中平凡部分的一种方法是根据[1]中关于平凡发展向量场的讨论, 验证 $Q_i\partial u_i$ 是否为平凡, 即(0, 0, …, 0, Q_i, 0, …, 0)是否满足方程 Δ• 对线性方程的情况选择 ξ_i, $\phi_j(i=1,2,\cdots,p;j=1,2,\cdots,q)$ 关于 u, v 线性来避免 X 中平凡项[5]• 但除非 X 是恒平凡的, 否则 X 中所含平凡项不影响获得非平凡的对称群•

此算法有以下三个优点:

1. 由于 $\xi_i, \phi_j (i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 的直接与多项式的具体形式参与计算, 易用任何一种计算机代数系统, 如 Mathematica, REDUCE 等来计算机上全过程实现(以往的算法做不到这一点), 并加快运行速度和节省内存• 也易移植到现有的计算对称群的程序中•

2. 理论上此算法虽不能产生给定偏微分方程组的全部对称群, 但能够产生非平凡的重要特殊(甚至全部)群•

3. 在实际操作中由于 $\xi_i, \phi_j (i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 直接展成多项式形式及其较短的符号的参与, 此方法适合于求解高维及大系统的对称群问题•

§ 3. 算法的实现和算例

作者用计算机代数系统"Mathematica 1.2 for DOS"[7]编制了求偏微分方程对称群的通用程序, 实现了本文建议的算法• 本程序可用两重方式使用•

1. 机器自动产生 $\xi_i, \phi_j (i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 的 Taylor 展示(CS)或(CH), 直接计算或用上面所述的逐步精化法得到最后的代数确定方程组•

2. 用逐步精化先产生某些特定'优化'的一般确定方程(偏微分方程), 然后把 $\xi_i, \phi_j (i = 1, 2, \cdots, p; j = 1, 2, \cdots, q)$ 或 $Q_i (i = 1, 2, \cdots, s)$ 的 Taylor 展式(CS)或(CH)代入这些'优先'方程中, 来简化 ξ_i, ϕ_j 或 Q_i 的表达式• 再计算下一个较'优先'方程, 进一步简化 ξ_i, ϕ_j 或 Q_i, 依次有限步之后得到 $a^i_{i_1 i_2 \cdots k_s}$, $b^j_{i_1 i_2 \cdots k_s}$ 或 $c^i_{i_1 i_2 \cdots k_s}$ 的代数确定方程• 从而求解这个代数方程组得到最后结果•

为说明本文提出的算法的有效性, 下面给出一个算例•

例 1 计算非线性 Shrodinger 方程

$$i \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + |u|^2 u$$

的 Lie 群的无穷小不变向量场•

由于篇幅所限, 我们仅给出部分计算中间过程(见附录)•

设 $u = u[1] + iu[2]$, 其中 i 为虚数单位, $u[1], u[2]$ 为实函数, 则方程变为

$$\begin{cases} \dfrac{\partial u[2]}{\partial t} + \dfrac{\partial^2 u[1]}{\partial x^2} + \dfrac{\partial^2 u[1]}{\partial y^2} + u[1](u[1]^2 + u[2]^2) = 0 \\ -\dfrac{\partial u[1]}{\partial t} + \dfrac{\partial^2 u[2]}{\partial x^2} + \dfrac{\partial^2 u[2]}{\partial y^2} + u[2](u[1]^2 + u[2]^2) = 0 \end{cases} \tag{s}$$

我们求(s)的如下无穷小不变向量场:

$$\begin{aligned} X = {} & \xi_1(t, x, y, u[1], u[2])\partial_t + \xi_2(t, x, y, u[1], u[2])\partial_x \\ & + \xi_3(t, x, y, u[1], u[2])\partial_y + \phi_1(t, x, y, u[1], u[2])\partial_{u[1]} \\ & + \phi_2(t, x, y, u[1], u[2])\partial_{u[2]} \end{aligned}$$

第一步: 为了避免内存溢出, 分别计算第一和第二方程产生的确定方程组, 并先取 $\left\{ \dfrac{\partial u[2]}{\partial t}^2, \dfrac{\partial u[2]}{\partial y}\dfrac{\partial u[2]}{\partial t}, \left\{ \dfrac{\partial u[2]}{\partial x}^2, \cdots \right.\right.$ 的系数构成的'优先'1确定方程组得到 ξ_1 只依赖 t, ξ_2 与 $u[1], u[2]$ 无关, $\phi_i (i = 1, 2)$ 关于 $u[1], u[2]$ 线性等 12 个简化结论• 以此结论为变换规则, 产生全部一般确定方程组, 得到 18 个方程(附录), 显然不易直接去解• 下面用本文的方法求解二阶多项式形式不变向量场• 为此机器自动产生符合上述简化结论的 $\xi_i (i = 1, 2,$

3), $\phi_j(j = 1, 2)$ 的 Taylor 多项式, 共有 273 个待定常数•

第二步: 利用前三个方程计算, 确定出 63 个常数, 简化 $\xi_i(i = 1, 2, 3)$, $\phi_j(j = 1, 2)$;

第三步: 利用第 4~ 7 方程, 54 个待定常数被确定, 再简化 ξ_i, ϕ_j;

第四步: 利用第 8~ 10 方程, 86 个待定常数被确定, 再次简化 ξ_i, ϕ_j, 至此 ξ_i, ϕ_j 中只含有 70 个待定常数•

第五步: 计算第 11~ 16 方程, 知它们恒成立, 未得到进一步的简化• 说明原确定方程组中这些方程是其它方程的线性组合•

最后通过第 17~ 18 方程得到最后结果:

$$\xi_1 = c_1 - 2c_5 t - c_9 t^2;$$

$$\xi_2 = c_2 - 2c_8 t + (-c_5 - c_9 t)x - c_4 y;$$

$$\xi_3 = c_3 - 2c_7 t + (-c_5 - c_9 t)x + c_4 y;$$

$$\phi_1 = u[1](c_5 + c_9 t) + u[2](-c_6 - c_8 x - c_9 x^2/4 - c_7 y - c_9 y^2/4);$$

$$\phi_2 = u[2](c_5 + c_9 t) + u[1](c_6 + c_8 x + c_9 x^2/4 + c_7 y + c_9 y^2/4);$$

容易验证 X 构成 Lie 代数

附 录

记: kes[i] = ξ_i, phai[i] = ϕ_i,

kes[i, { i_1, i_2, i_3, i_4, i_5 }] = $d\xi_i / dt^{i_1} dx^{i_2} dy^{i_3} du[1]^{i_4} du[2]^{i_5}$.

phai[i, { i_1, i_2, i_3, i_4, i_5 }] = $d\phi_i / dt^{i_1} dx^{i_2} dy^{i_3} du[1]^{i_4} du[2]^{i_5}$

确定方程为:

equation[1] = - 2* kes[2, {20, 1, 0, 0, 0 2] + 2* kes[3, { 0, 0, 1, 0, 0] ;

equation[2] = - 2* kes[2, { 0, 0, 1, 0, 0] - 2* kes[3, { 0, 1, 0, 0, 0] ; [u

equation[3] = 2* phai[1, { 0, 0, 0, 1, 1] ;

equation[4] = - kes[3, { 1, 0, 0, 0, 0 }] + 2* phai[1, { 0, 0, 1(0, 1 }] ;

equation[5] = - kes[3, { 0, 0, 2, 0, 0] - kes[3, { 0, 2, 0, 0, 0 }] + 2* phai[1, { 0, 0, 1, 1, 0] ;

equation[6] = - kes[2, { 1, 0, 0, 0, 0 }] + 2* phai[1, { 0, 1, 0, 0, 1 }] ; [

equation[7] = - kes[2, { 0, 0, 2, 0, 0] - kes[2, { 0, 2, 0, 0, 0 }] + 2* phai[1, { 0, 1, 0, 1, 0] ;

equation[8] = kes[1, { 1, 0, 0, 0, 0 }] - 2* kes[3, { 0, 0, 1, 0, 0 }] - phai[1, { 0, 0, 0, 1, 0 #] + phai[2, { 0, 0, 0, 0, 1] ; [u (s

equation[9] = - kes[1, { 1, 0, 0, 0, 0] + 2* kes[3, { 0, 0, 1, 0, 0] - phai[1, { 0, 0, 0, 1, 0]

解二 + phai[2, { 0, 0, 0, 0, 1 }] ;

equation[10] = - phai[1, { 0, 0, 0, 0, 1 }] - phai[2, { 0, 0, 0, 1, 0 = ,

```
                2  + 2* kes[3,{0,0,1,0,0}]* u[1]* u[2]^2- phai[1,{0,0,0,1,0}]* u[1]* u[2]^2
                - phai[1,{0,0,0,0,1}]* u[2]^3;
(
equation[18]= - phai[1,{1,0,0,0,0}]+ phai[2,{0,0,2,0,0}]+ phai[2,{0,2,0,0,0}]        只含
              + phai[2]* u[1]^2- phai[2,{0,0,0,1,0}]* u[1]^3+ 2* phai[1]* u[1]* u[2]    组中
              + 2* kes[3,{0,0,1,0,0}]* u[1]^2* u[2]- phai[2,{0,0,0,0,1}]* u[1]^2* u[2]
              + 3* phai[2]* u[2]^2- phai[2,{0,0,0,1,0}]* u[1]* u[2]^2+ 2kes[3,{0,0,1,0,0}]* u
              [2]^3
              - phai[2,{0,0,0,0,1}]* u[2]^3;
```

经过前 4 步的简化得到含 70 个待定常数的生成函数:

```
kess[1]= c[1]+ 2* c[4]* x[1]- 4* c[44]* x[1]^2;
kess[2]= c[2]- 2* c[40]* x[1]- c[54]* x[1]^2+ (c[4]- 4* c[44]* x[1])* x[2]- c[5]* x[3];
kess[3]= c[3]- 2* c[37]* x[1]- c[51]* x[1]^2+ c[5]* x[2]+ (c[4]- 4* c[44]* x[1])* x[3];
phaii[1]= c[6]+ c[15]* x[1]+ c[24]* x[1]^2+ u[1]* (c[34]+ c[48]* x[1]+ c[61]* x[1]^2)
  + (c[9]+ c[18]* x[1]+ c[27]* x[1]^2)* x[2]+ (c[12]+ c[21]* x[1]+ c[30]* x[1]^2)* x[2]^2
  + (c[7]+ c[16]* x[1]+ c[25]* x[1]^2+ (c[10]+ c[19]* x[1]+ c[28]* x[1]^2)* x[2]
  + (c[13]+ c[22]* x[1]+ c[31]* x[1]^2)* x[3]+ (c[8]+ c[17]* x[1]+ c[26]* x[1]^2
  + (c[11]+ c[20]* x[1]+ c[29]* x[1]^2)* x[2]+ (c[14]+ c[23]* x[1]
  + c[32]* x[1]^2)* x[2]^2)* x[3]^2+ u[2]* (- c[35]- c[49]* x[1]- c[62]* x[1]^2
  + (- c[40]- c[54]* x[1])* x[2]- c[44]* x[2]^2+ (- c[37]- c[51]* x[1])* [3]
  - c[44]* x[3]^2);
phaii[2]= c[33]+ c[47]* x[1]+ c[60]* x[1]^2+ u[2]* (c[34]+ c[48]* x[1]+ c[61]* x[1]^2)
  + (c[39]+ c[53]* x[1]+ c[65]* x[1]^2)* x[2]+ (c[43]+ c[57]* x[1]+ c[68]* x[1]^2)* x[2]^2
  + (c[36]+ c[50]* x[1]+ c[63]* x[1]^2+ (c[41]+ c[55]* x[1]+ c[66]* x[1]^2)* x[2]
  + (c[45]+ c[58]* x[1]+ c[69]* x[1]^2)* x[2]^2)* x[3]+ (c[38]+ c[52]* x[1]
  + c[64]* x[1]^2+ (c[42]+ c[56]* x[1]+ c[67]* x[1]^2)* x[2]+ (c[46]+ c[59]* x[1]
  + c[70]* x[1]^2)* x[2]^2)* x[3]^2+ u[1]* (c[35]+ c[49]* x[1]+ c[62]* x[1]^2
  + (c[40]+ c[54]* x[1])* x[2]+ c[44]* x[2]^2+ (c[37]+ c[51]* x[1])* x[3]
  + c[44]* x[3]^2);
```

计算最后两个方程得到

```
{{c[4]- > - c[34], c[6]- > 0, c[7]- > 0, c[8]- > 0, c[9]- > 0, c[10]- > 0,
  c[11]- > 0, c[12]- > 0, c[13]- > 0, c[14]- > 0, c[15]- > 0, c[16]- > 0,
  c[17]- > 0, c[18]- > 0, c[19]- > 0, c[20]- > 0, c[21]- > 0, c[22]- > 0,
  c[23]- > 0, c[24]- > 0, c[25]- > 0, c[26]- > 0, c[27]- > 0, c[28]- > 0,
  c[29]- > 0, c[30]- > 0, c[31]- > 0, c[32]- > 0, c[33]- > 0, c[36]- > 0,
  c[38]- > 0, c[39]- > 0, c[41]- > 0, c[42]- > 0, c[43]- > 0,
  c[44]- > c[48]/4, c[45]- > 0, c[46]- > 0, c[47]- > 0, c[49]- > 0,
  c[50]- > 0, c[51]- > 0, c[52]- > 0, c[53]- > 0, c[54]- > 0, c[55]- > 0,
  c[56]- > 0, c[57]- > 0, c[58]- > 0, c[59]- > 0, c[60]- > 0, c[61]- > 0,
  c[62]- > 0, c[63]- > 0, c[64]- > 0, c[65]- > 0, c[66]- > 0, c[67]- > 0,
  c[68]- > 0, c[69]- > 0, c[70]- > 0}}
```

最后结果为 9 个参数的不变向量场:

```
kess[1]= c[1]- 2* c[5]* x[1]- c[9]* x[1]^2;
kess[2]= c[2]- 2* c[8]* x[1]+ (- c[5]- c[9]* x[1])* x[2]- c[4]* x[3];
kess[3]= c[3]- 2* c[7]* x[1]+ c[4]* x[2]+ (- c[5]- c[9]* x[1])* x[3];
```

```
phaii[1]= u[1]*(c[5]+c[9]*x[1])+u[2]*(-c[6]-c[8]*x[2]-(c[9]*x[2]^2)/4
  -c[7]*x[3]-(c[9]*x[3]^2)/4);
phaii[2]= u[2]*(c[5]+c[9]*x[1])+u[1]*(c[6]+c[8]*x[2]+(c[9]*x[2]^2)/4
  +c[7]*x[3]+(c[9]*x[3]^2)/4)
```

参 考 文 献

1 P. J. Olver, Applications of Lie Groups to Differential Equations, Springer_Verlag, New York, Berlin, Heideberg, Tokyo (1986).

2 R. O. Weber and S. I. Barry, Finite_time blow_up in reaction diffiusion equations, Math. Comput. Modelling, **18** (10) (1993), 163—168.

3 Robert Floreanini and Luc. Vinet, Lie symmetries of finite_difference equations, J. Math. Phys., **36** (12), (1995).

4 F. Schwtz, Automatically determining symmetries of differential equations, Computing, **74** (1985), 91—106.

5 G. Bluman and S. Kumei, Symmetries and Differential Equations, Springer_Verlag, New York, Berlin, Heideberg, Tokyo (1989).

6 L. V. Ovsiannikov, Group Analysis of Differential Equations, Academic Press, New York (1982).

7 沈凤仙,《"Mathematica"使用手册》,海洋出版社 (1991).

Taylor Polynomial Stepwise Refinement Algorithm for Lie and High Symmetries of Partial Differential Equations

Zhang Hongqing Tang Limin

(Institute of Mathematics, Dalian Univ. of Tech., Dalian 116024, P. R. China)

Chao Lu

(Basic Science Department of Inner Mongolia Polyt. Univ., Huhhot 010062, P. R. China)

Tang Limin

(Department of Mechanics, Dalian Univ. of Tech., Dalian 116024, P. R. China)

Abstract

In this article, based on the Taylor expansions of generating functions and stepwise refinement procedure, authors suggest a algorithm for finding the Lie and high (generalized) symmetries of partial differential equations (PDEs). This algorithm transforms the problem having to solve over_determining PDEs commonly encountered and difficulty part in standard methods into one solving to algebraic equations to which one easy obtain solution. So, it reduces significantly the difficulties of the problem and raise computing efficiency. The whole procedure of the algorithm is carried out automatically by using any computer algebra system. In general, this algorithm can yields many more important symmetries for PDEs.

Key words Lie groups, high symmetries, Taylor polynomial, computer algebra, determining equations, stepwise refinement