物理计算的保真与代数动力学算法*
——II. 代数动力学算法与其他算法计算结果的比较

王顺金** 张 华

(四川大学物理学院理论物理中心, 成都 610064)

摘要 基于常微分方程的偏微分形式的代数动力学精确解, 在 Taylor 级数表示的局域精确解的有限项截断近似下, 建立起代数动力学算法. 在四阶近似下实现了常微分方程的数值求解, 在12个典型的动力学系统的计算机实验中比较了三种算法的精度及其优缺点. 结果表明, 代数动力学算法是独立于辛几何算法和 Runge-Kutta 算法的第三种算法, 它有可能克服辛几何算法的动力学失真和 Runge-Kutta 算法的人为耗散, 在可预期和可控制的精度下兼顾运动学代数-几何保真和动力学守恒律保真.

关键词 常微分方程的代数动力学解法和代数动力学算法 三种算法计算结果的比较 运动学代数-几何保真和动力学守恒律保真

文献[1]介绍了代数动力学算法的数学理论基础, 包括常微分方程的代数动力学解法及其相关问题, 精确的、解析的 Taylor 级数解的截断近似和代数动力学算法设计, 代数动力学算法与辛几何算法和 Rounge-Kutta 算法的关系以及对三种算法的特点的分析. 本文将系统地报告代数动力学算法、辛几何算法和 Rounge-Kutta 算法对12个典型的动力学系统的数值求解的结果以及对这些结果的比较分析, 从而可以使人们对三种算法的优缺点有一个具体而定量的认识, 以便于根据具体情况选择合适的算法, 并在实际使用中进一步发展这些算法.

收稿日期: 2005-02-21; 接受日期: 2005-09-19
* 国家自然科学基金(批准号: 10375039, 90503008)、教育部博士点基金和兰州重离子加速器国家实验室核理论中心基金资助项目
** E-mail: sjwang@home.swjtu.edu.cn

1　三种数值计算方法

1.1　常微分方程的几种主要的数值计算方法

常微分方程的数值求解方法, 可以按 Hamilton 系统和非 Hamilton 系统分为两大类:

(ⅰ) Hamilton系统的算法. Hamilton系统具有辛代数-几何结构, 目前有下列三种算法可供使用: ⅰ) Runge-Kutta算法[2,3]. 这是最流行的算法, 已有 110 年的历史. ⅱ) 辛几何算法[4~6]. 这是中国科学院计算数学研究所的冯康、秦孟兆、唐贻发和孙耿等人, 针对Runge-Kutta算法不能保持Hamilton系统的辛几何结构和具有人为耗散等缺点而提出的. 中国科学院理论物理研究所郭汉英、吴可等人把辛算法推广用于场论系统, 发展了多辛算法和基于Lagrange变分原理的多辛格式[7~9]. 该算法已有 22 年的历史. ⅲ) 代数动力学算法. 这是本文介绍的一种不同于上述两种算法的第三种算法. 根据Hamilton系统有、无复合代数结构, 这一算法有单参数群代数动力学算法和复合代数动力学算法之分. 根据精度不同, 有朴素代数动力学算法(U_N)、辛代数动力学算法(sU_N)和高精度代数动力学算法(eU_N)之分.

(ⅱ) 非 Hamilton 系统的算法. 这类系统没有辛几何结构, 辛几何算法不能使用. 因此, 目前只有 Runge-Kutta 算法及其变种和代数动力学算法可供使用.

1.2　四阶辛几何算法、Runge-Kutta 算法和代数动力学算法设计

1.2.1　辛几何算法设计[4~6]

如文献[1]3.2节所述, 四阶辛几何算法的设计是基于时间演化算子的非正则表示的四阶截断近似, 即

$$U_4 \approx S_4 = \exp[a_1 hT]\exp[b_1 hV]\exp[a_2 hT]\exp[b_2 hV]\exp[a_2 hT]\exp[b_1 hV]\exp[a_1 hT], \tag{1.1}$$

其中

$$T = \frac{\partial H}{\partial p}\frac{\partial}{\partial q} = T_p \frac{\partial}{\partial q}, \quad V = -\frac{\partial H}{\partial q}\frac{\partial}{\partial p} = f(q)\frac{\partial}{\partial p}, \tag{1.2}$$

$$s = 2^{1/3}, \ a_1 = \frac{1}{2}\frac{1}{2-s}, \ a_2 = -\frac{1}{2}\frac{s-1}{2-s}, \ b_1 = \frac{1}{2-s}, \ b_2 = -\frac{s}{s-2}, \tag{1.3}$$

但实际计算中使用如下等价的迭代公式:

$$q_1 = q_n + a_1 hT[p_n], \ p_1 = p_n + b_1 hV[q_1], \ q_2 = q_1 + a_2 hT[p_1], \ p_2 = p_1 + b_2 hV[q_2],$$
$$T(p) = \frac{\partial H}{\partial p}, \ V(q) = -\frac{\partial H}{\partial q}, \ p_{n+1} = p_2 + b_1 hV[q_2], \ q_{n+1} = q_2 + q_1 hT[p_{n+1}].$$

对于 p, q 不可分离的 Hamilton 系统，可以采用欧拉中点算法:

$$p^{n+1}=p^n-\tau H_q\left(\frac{q^n+q^{n+1}}{2},\frac{p^n+p^{n+1}}{2}\right),\quad q^{n+1}=q^n+\tau H_p\left(\frac{q^n+q^{n+1}}{2},\frac{p^n+p^{n+1}}{2}\right)$$

研究表明，上述几种辛算法是辛Runge-Kutta算法[2,3].

1.2.2 Runge-Kutta算法设计[2,3]

运动方程、四阶速率迭代和轨道迭代格式如下:

$$\begin{aligned}&\frac{\mathrm{d}y}{\mathrm{d}t}=f(t,y),\\&k1=f(t,y^n),\quad k2=f\left(t+\frac{h}{2},y^n+\frac{h}{2}k1\right),\\&k3=f\left(t+\frac{h}{2},y^n+\frac{h}{2}k2\right),\quad k4=f(t+h,y^n+hk3),\\&y^{n+1}=y^n+\frac{h}{6}(k1+2k2+2k3+k4).\end{aligned}\tag{1.4}$$

1.2.3 代数动力学算法设计

如文献[1]中 2.3.2 和 3.1 节所述，按照常微分方程的代数动力学解法[1,10]，常微分方程及其解的四阶近似如下:

$$U_4=1+tL+\frac{t^2}{2!}L^2+\frac{t^3}{3!}L^3+\frac{t^4}{4!}L^4, z_4(z_0,t)=U_4(t)z_0,\tag{1.5}$$

如果加上六阶修正以提高保辛精度，则修正后的时间演化算子为 $U_{4-6}=U_4+\frac{1}{144}t^6L^6$. 如考虑提高能量精度的修正，则近似的时间演化算子应作如下修正：四阶近似轨道迭代格式为 $\begin{pmatrix}q_1^n\\p_1^n\end{pmatrix}=U_4\begin{pmatrix}q^{n-1}\\p^{n-1}\end{pmatrix}$，修正后的轨道迭代格式为: $q^n=q_1^n-\varepsilon_q, p^n=p_1^n-\varepsilon_p$，其中

$$\varepsilon_q=(H(q_1^n,p_1^n)-E_0)\left|\frac{\partial H(q,p)}{\partial q}\right|_{q_1^n p_1^n},$$

$$\varepsilon_p=(H(q_1^n,p_1^n)-E_0)\left|\frac{\partial H(q,p)}{\partial p}\right|_{q_1^n p_1^n}.$$

1.2.4 复合代数动力学算法

基于复合代数结构的代数动力学算法设计如前文[1]所述：从 H 的复合代数

结构 $H(q,p)=H[X_i(q.p)],\{X_i,X_j\}=C_{ij}{}^k X_k$ 和 $\{X_i\}$ 的运动方程 $\dot{X}_i=\dfrac{\partial X_i[X,t]}{\partial t}=\hat{L}[X]X_i$，$\hat{L}=\sum_i F_i[X]\dfrac{\partial}{\partial X_i}$，得到代数动力学解: $X_i[X_0,t]=\hat{U}[X_0,t]X_{i0}$，按其时间演化算子 $\hat{U}[X_0,t]=\mathrm{e}^{\hat{L}[X_0]t}$ 设计 $\hat{U}$ 的N阶算法 $\hat{U}_N$ 为 $\hat{U}_N=\sum_{n=0}^{N}\dfrac{t^n}{n!}\hat{L}^n$.

本文所用的朴素代数动力学算法是显式算法. 对 Hamilton 系统, 这种算法只能在一定的精度下近似保持辛几何结构. 最近的研究表明, 对 Hamilton 系统, 也可以根据设定的精度建立隐式代数动力学算法. 这种隐式代数动力学算法是严格保持局域辛结构的, 所以叫做辛代数动力学算法. 辛代数动力学算法建立起代数动力学算法与辛几何算法和辛 Runge-Kutta 算法的密切关系, 它可以改善辛几何算法的动力学失真, 克服 Runge-Kutta 算法的人为耗散问题, 提高它们的精度, 达到与代数动力学算法的相同的水平.

2　检验算法的典型问题

2.1　辛几何算法的 8 个考题

冯康等人[6]提出检验一个算法的8个考题如下: (ⅰ) 谐振子, (ⅱ) 非线性 Duffing动量相关势振子, (ⅲ) 惠更斯振子, (ⅳ) Cassini振子, (ⅴ) Toda格点模型, (ⅵ) Lissajous图形, (ⅶ) 椭球面测地线流, (ⅷ) 开普勒运动.

2.2　检验三种算法的 12 个典型问题

下面介绍三种算法在 12 个典型问题上的数值计算结果的比较, 这 12 个问题是: (ⅰ) 谐振子, (ⅱ) Duffing 动量相关势振子, (ⅲ) Cassini 振子, (ⅳ) 惠更斯振子, (ⅴ) Lissajous 图形系统, (ⅵ) 开普勒系统, (ⅶ) 扰动的开普勒系统, (ⅷ) Toda 格点模型, (ⅸ) Henon-Heiles 模型(Hamilton 混沌系统), (ⅹ) Lorentz 气象模型(非 Hamilton 混沌系统), (ⅹⅰ) 三维轨道角动量系统(复合代数动力学系统), (ⅹⅱ) 椭球面上的自由运动(测地线流).

3　三种算法对 12 个典型问题的计算结果

数值计算按上节呈述的三种四阶算法格式进行.

(ⅰ) 谐振子模型: $H(p,q)=\dfrac{1}{2}p^2+\dfrac{1}{2}q^2$. 参数选择: 初值选取为: $p_0=0.0$, $q_0=1.0$; 时间步长: $h=0.4$; 计算 10 万步. 计算使用 Pentium IV,2.4GHz,1G RAM, 计算效率比较见表 1, 结果显示代数动力学算法较慢.

表 1 计算 10 万步所需时间的比较(单位为 s)

步长	辛算法	代数动力学算法	RK4
0.4	0.0913	0.1589	0.1366
0.1	0.3858	0.6457	0.5410
0.02	1.8888	3.3279	2.7057

计算结果(步长 = 0.4):

(1) 轨道: 计算见图1. 代数动力学算法(alge)和 Runge-Kutta 算法(rk4)十分接近于精确解(q), 4 阶辛 Runge-Kutta 算法(symp)对于精确解有相位移动, 造成它对于精确解的较大偏离, 这是动力学失真的表现(图 2). 在同一四阶近似下, 代数动力学算法轨道的绝对误差的精度比其他算法高出约一个量级(大的误差是由于大步长造成的)(图 2~4).

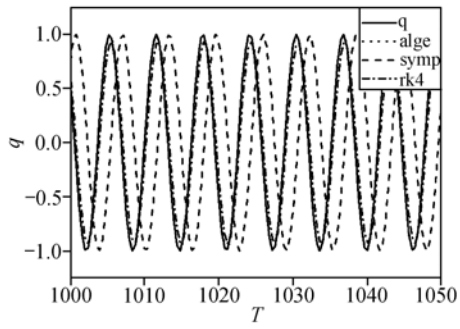


图 1 解析解和三种算法计算所得的轨道 $q(t)$

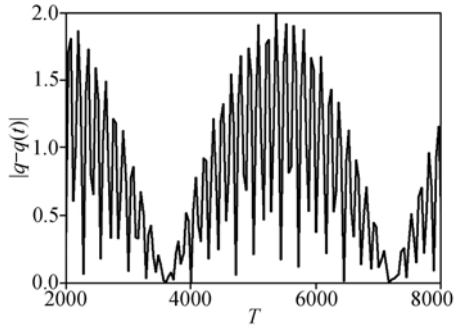


图 2 辛算法的轨道绝对误差

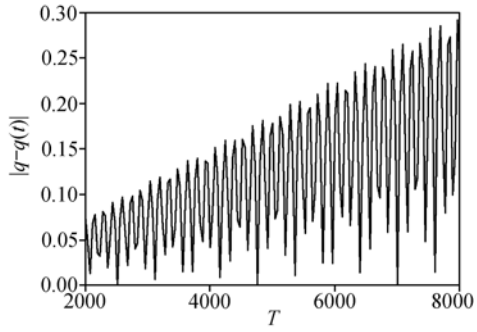


图 3 代数动力学算法轨道的绝对误差

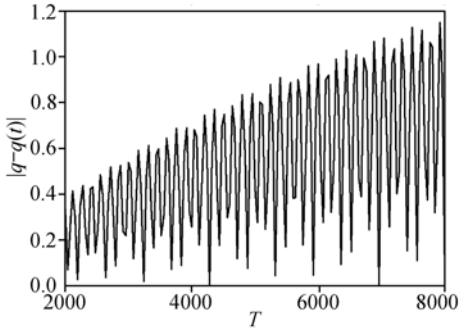


图 4 Runge-Kutta 算法轨道的绝对误差

(2) 相图: 在 10 万步计算中, Runge-Kutta 算法出现由于能量耗散导致的轨道收缩, 而辛算法和代数动力学算法并未显示出轨道的耗散性收缩(图 5~7).

(3) 能量相对误差: 代数动力学算法的能量相对误差的精度高出4个量级(图 8~10).

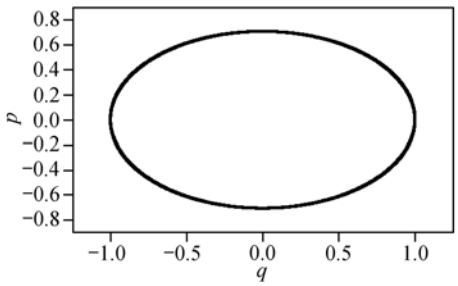


图 5　辛算法的相图

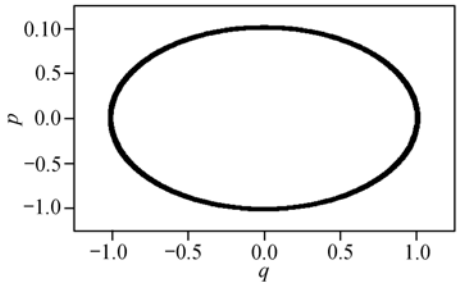


图 6　代数动力学算法的相图

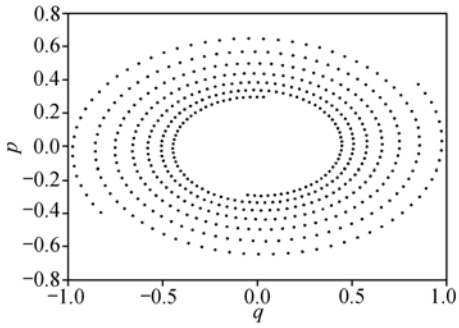


图 7　Runge-Kutta 算法计算的相图
(出现耗散性收缩)

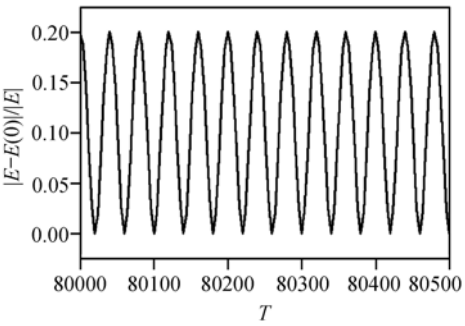


图 8　辛算法的能量相对误差

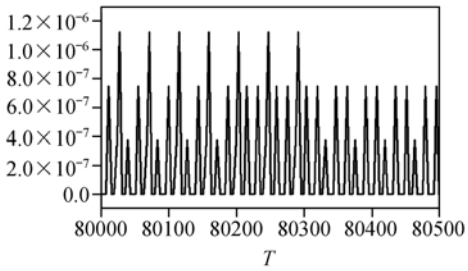


图 9　代数动力学算法的能量相对误差

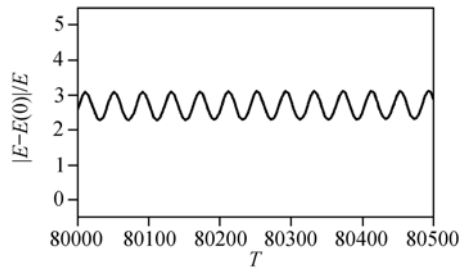


图 10　Runge-Kutta 算法的能量相对误差

(ⅱ) xp 藕合谐振子: 模型: $H(p,q)=\frac{1}{2}p^2+\frac{1}{2}q^2+\frac{1}{2}pq$. 参数选择: 初值选取为: $p_0=0.0$, $q_0=1.0$; 步长: h = 0.4, 使用 Pentium IV,2.4GHz,1G RAM, 计算 25 万步所需时间见表 2, 代数动力学算法为显示算法, 比其他两个隐式算法效率高. 计算结果(步长 h = 0.4):

(1) 轨道(图 11 和 12): 代数动力学算法(algeq)与精确解析解(rsol)最接近; 辛

表 2 计算效率比较(单位为 s)

步长	辛算法	代数动力学	Runge-Kutta
0.5	0.794895	0.251986	1.523053
0.4	0.972545	0.297514	1.902968
0.2	1.942686	0.594551	3.800498
0.1	3.890216	1.189868	7.611460

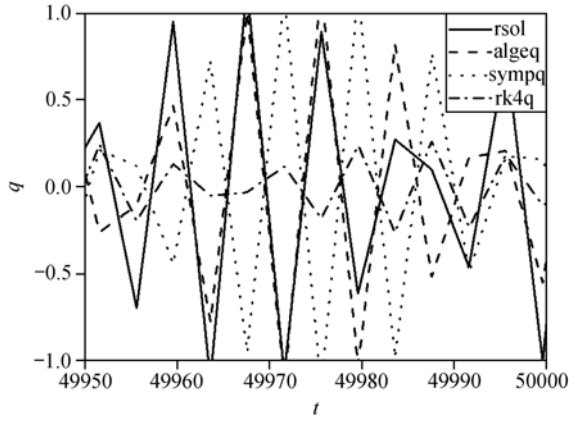


图 11 三种算法计算的轨道 $q(t)$与精确解的比较

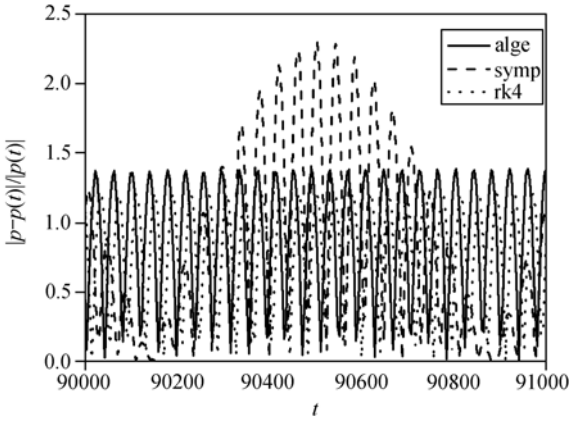


图 12 三种算法计算的动量 p 对精确解的相对误差

算法(sympq)与精确解的偏离由频率改变和相位移动造成, rk4q 为 Runge-Kutta 算法, 由于能量耗散, 其振幅减小很快. 图形的不光滑变化是由于每隔 100 点进行抽样造成的. 大的误差是由于大步长(0.4)造成的. 图 12 给出三种算法计算的动量 p 对精确解的相对误差.

(2) 相图: 在 25 万步计算中, Runge-Kutta 算法出现由于能量耗散导致的轨道收缩, 而辛算法和代数动力学算法并未显示出轨道的耗散性收缩(图 13~15). 此外, 四阶隐式辛算法中非线性方程迭代求解有时不收敛.

(3) 能量: 对这一模型, 辛算法保持能量精确守恒, Runge-Kutta 算法出现能量耗散, 代数动力学算法能量的相对精度达 10^{-7}(见图 16~18).

(iii) Cassini 振子: 模型: $H(x,p)=(x^2+p^2)^2-2c^2(x^2-p^2)$. 参数选择: $c=1$, 分别计算能量为 $E=-0.9599$, 1.0736, 2 等 4 条轨道. 对 $x=1$, 则对应为 $P=0.1$ (−0.1), 0.485868, 0.681519, −0.8036.

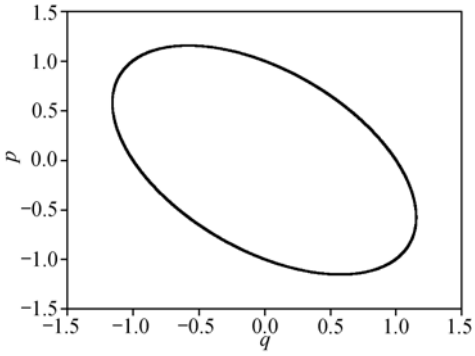


图 13　Euler 中点辛算法的相图

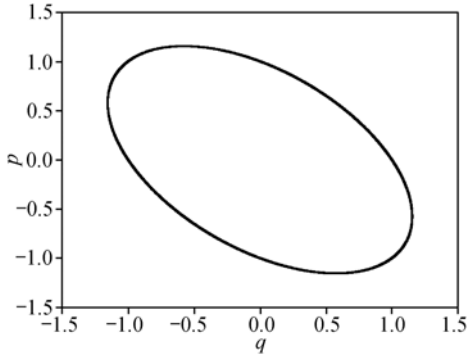


图 14　代数动力学算法的相图

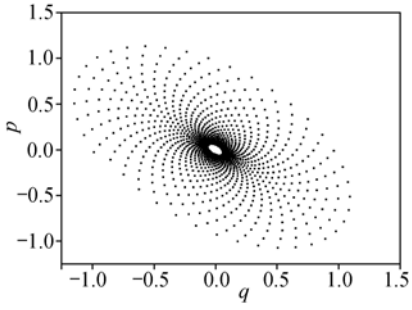


图 15　Runge-Kutta 算法的相图
(出现耗散性收缩)

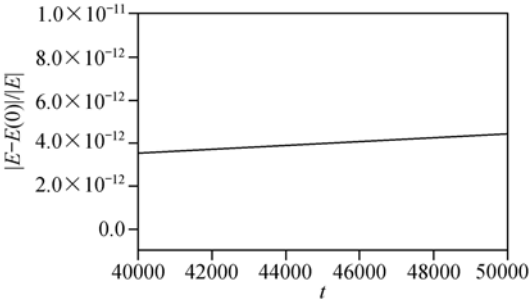


图 16　Euler 中点辛算法的能量相对误差
(能量精确守恒)

对于辛算法, 采用隐式 Euler 中点算法, 用 Newton-Rapson 方法求解非线性方程(其误差估计为 10^{-10}), 选取步长为 0.02 , 计算 1 万步, 在 Pentium IV 1.7 GHz 机上计算时间为: 代数动力学: 2.097141 (seconds); 辛算法: 5.031093(seconds); Runge-Kutta 算法: 3.839443 (seconds). 计算结果:

(1) 相图: 对于非线性动量相关势模型, 当能量接近零(临界点)时, 辛算法遇

到困难，不能给出连续的相图(图 19).

(2) 能量：代数动力学算法的能量相对误差比其他两种算法高出两个量级(图 22~24).

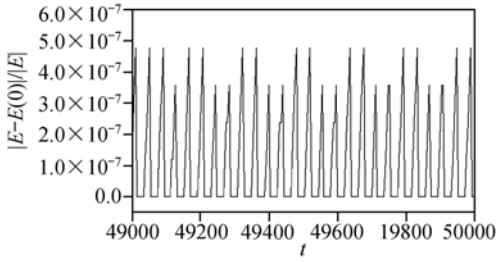


图 17 代数动力学算法的能量相对误差

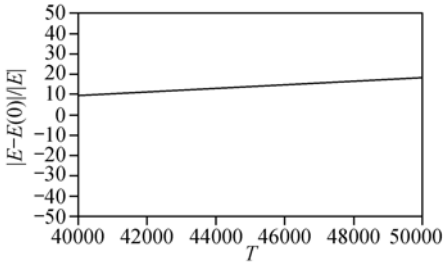


图 18 Runge-Kutta 算法的能量相对误差

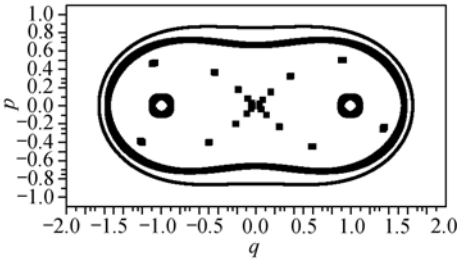


图 19 辛算法的相图($E\approx 0$ 时不连续)

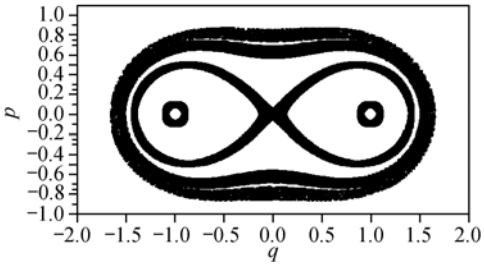


图 20 代数动力学算法的相图

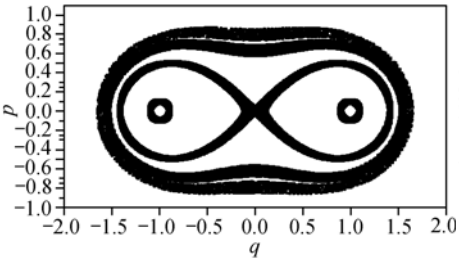


图 21 Runge-Kutta 算法的相图

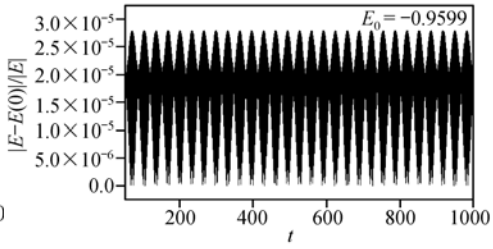


图 22 辛算法的能量相对误差

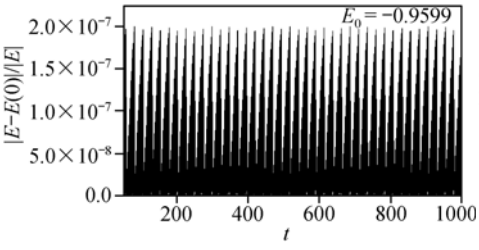


图 23 代数动力学算法的能量相对误差

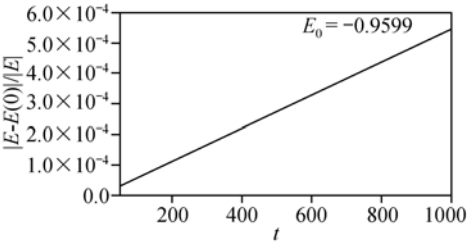


图 24 Runge-Kutta 算法的能量相对误差

(ⅳ) Huygens 振子: 模型: $H(p,q)=\frac{1}{2}p^2-4q^2+2q^4$. 参数选择: 分别计算 3 个能量点{1.625, –0.47, –0.47, –0.00024}对应的轨道, 步长为 0.15, 在 Pentium IV 1.7GHz 机上运行 10 万步, 计算时间分别为: 代数动力学: 7.8557(seconds); 辛算法: 3.7815 (seconds); Runge-Kutta 算法: 5.3497 (seconds).

计算结果(步长为 0.15):

(1) 相图: 代数动力学算法给出很好的相图, 辛 Runge-Kutta 算法的相图在临界点附近(E~0, p, q~0)很不好. Runge-Kutta 算法的相图很混乱, 出现耗散性收缩(图 25~27).

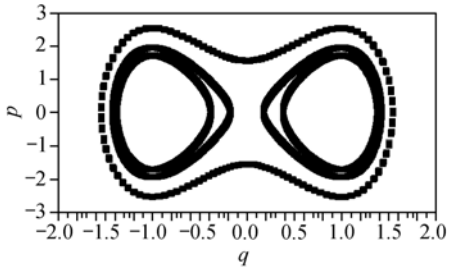


图 25　辛算法的相图

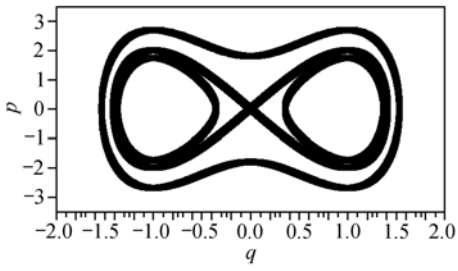


图 26　代数动力学算法的相图

(2) 能量: 代数动力学算法的能量相对误差高出 4~5 个量级(图 28~30).

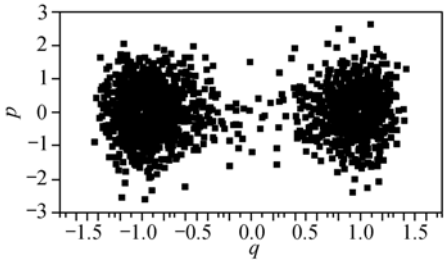


图 27　Runge-Kutta 算法的相图

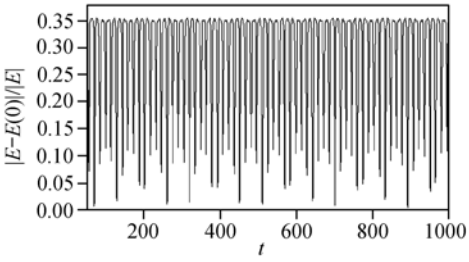


图 28　辛算法的能量相对误差

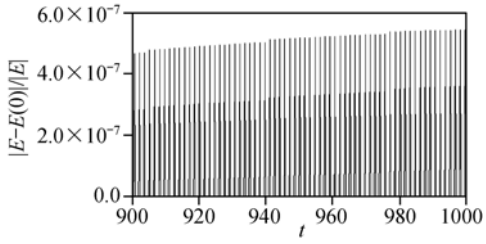


图 29　代数动力学算法的能量相对误差

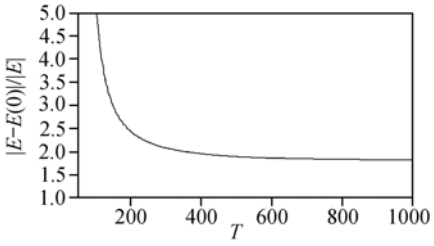


图 30　Runge-Kutta 算法的能量相对误差

(ⅴ) Lissajou 图形: 模型: $H(p,q)=\frac{1}{2}{p_1}^2+\frac{1}{2}{p_2}^2+k_1{q_1}^2+k_2{q_2}^2$. 参数选择: k_1 = 0.1, k_2 = 0.2. 初值为: $\{p_1=0.0, p_2=1.0, q_1=1.0, q_2=0.0\}$, 长为 0.5, 在 Pentium IV 1.7 GHz 机上运行 10 万步, 记录间隔为每隔 10 步记录一次.

计算结果:

(1) 轨道: 代数动力学算法与解析解基本重合, 其轨道的绝对误差最小, 辛算法有较大相位移动, 造成其轨道运动对于解析解的较大偏离(图 31 和 32).

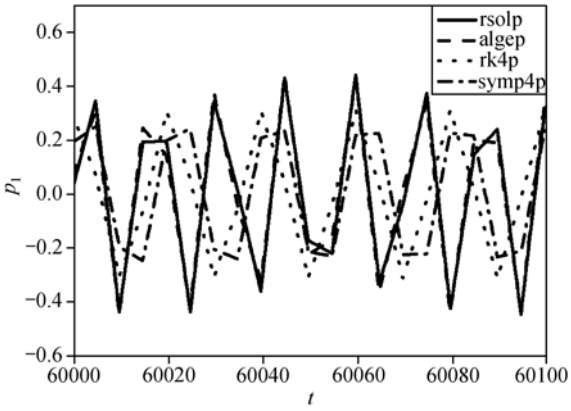


图 31 三种算法和解析解 p_1 的比较

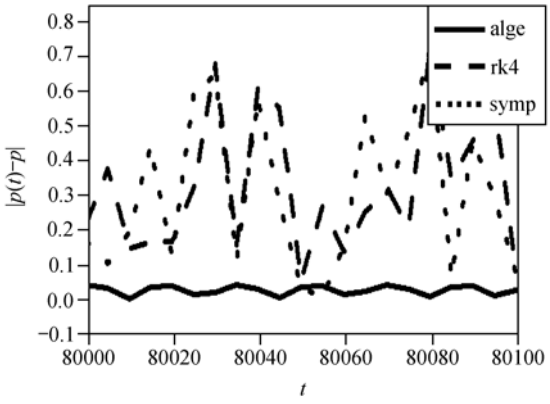


图 32 三种算法计算 p_1 的绝对误差

(2) 相图: 三种算法生成的相图的宏观结构相似, 由于精度不同, 其微观结构不同(图 33~35).

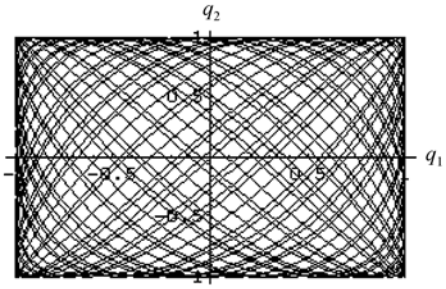


图 33　辛算法的相图

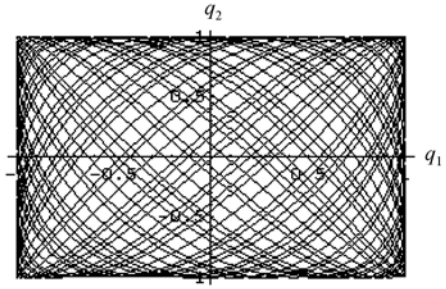


图 34　代数动力学算法的相图

(3) 能量: 代数动力学算法的能量相对误差比其他两种算法高出 4 个量级(图 36~38).

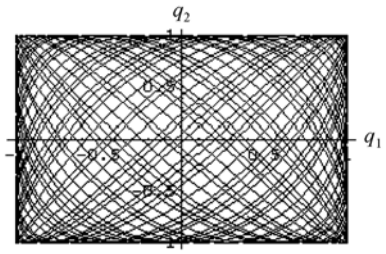


图 35　Runge-Kutta 算法的相图

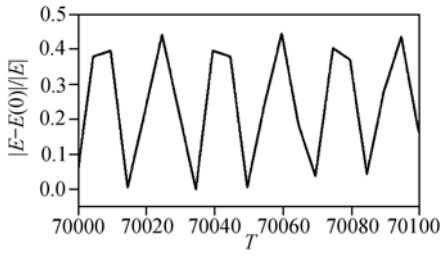


图 36　辛算法的能量相对误差

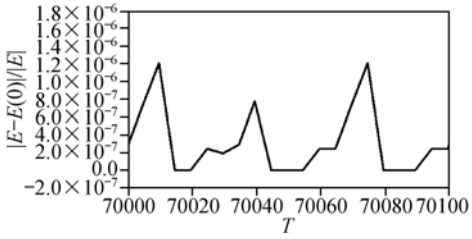


图 37　代数动力学算法的能量相对误差

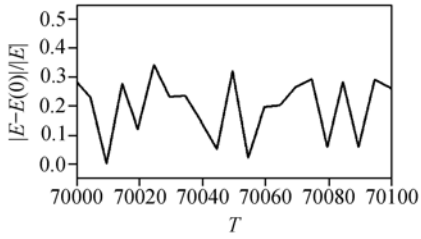


图 38　Runge-Kutta 算法的能量相对误差

(vi) Kepler 问题: 模型: $H(p,q)=\frac{1}{2}{p_1}^2+\frac{1}{2}{p_2}^2-\frac{1}{\sqrt{{q_1}^2+{q_2}^2}}$. 参数选择: 偏心率为 $\frac{11}{20}$, 初值 $p_1(0)=0, p_2(0)=\frac{\sqrt{31}}{3}, q_1(0)=\frac{9}{20}, q_2(0)=0$, 能量 E = –0.5. 使用 Pentium IV 2.4GHz CPU 机上运行 1 万步, 计算效率见表 3.

表 3 计算效率比较(单位为 s)

步长	RK4	代数动力学	辛算法
0.4	0.0224	0.1799	0.0150
0.2	0.0449	0.3832	0.0299
0.1	0.0899	0.7588	0.0595

计算结果:

(1) 轨道: 代数动力学算法的精度比辛算法高出 4~5 个量级, 这是由于辛算法相位移动造成的(图 39). 图 39 以代数动力学算法为基准(其精度为 10^{-8}), 步长为 0.01.

(2) 相图: 步长为 0.0365 时, 辛算法计算所得的轨道不闭合, 代数动力学算法计算所得的轨道基本闭合, Runge-Kutta 算法计算所得的轨道有耗散(图 40~42).

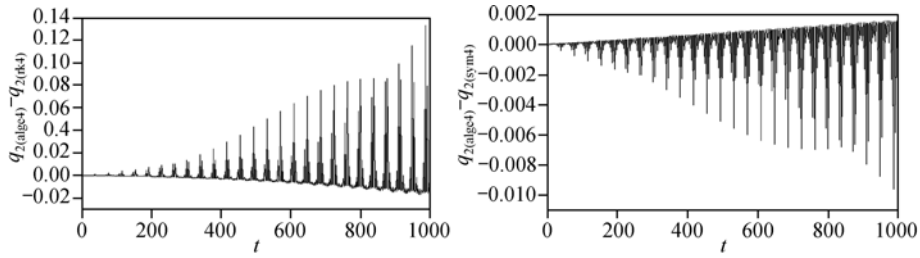


图 39 代数动力学算法与 Runge-Kutta 算法和辛算法所得轨道的比较

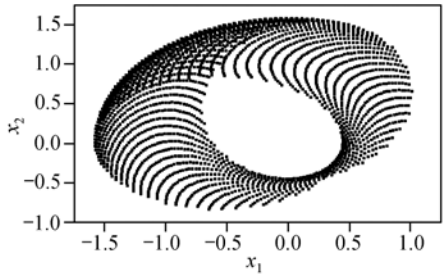


图 40 辛算法的轨道

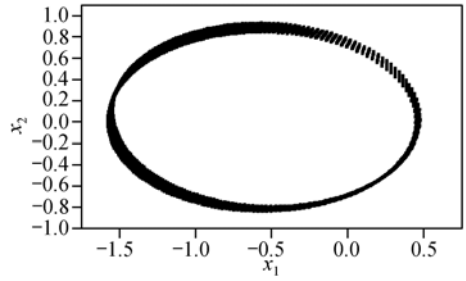


图 41 代数动力学算法的轨道

(3) 能量: 步长为 0.0365 时, 代数动力学算法的能量绝对误差的精度比辛算法高一个量级, 比 Runge-Kutta 算法高 4 个量级(图 43~45).

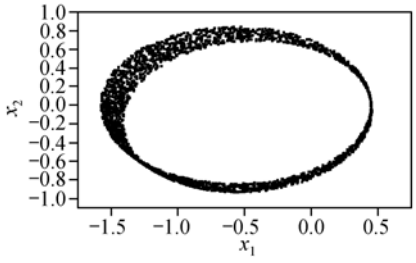


图 42 Runge-Kutta 算法的轨道

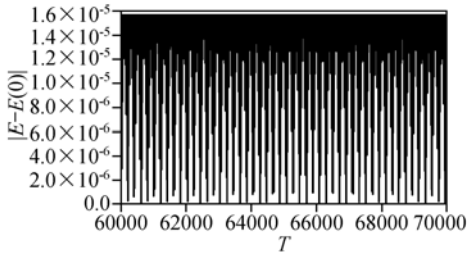


图 43 辛算法的能量绝对误差

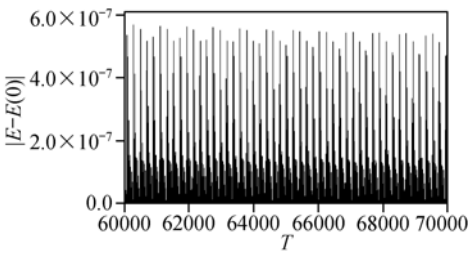


图 44　代数动力学算法的能量绝对误差

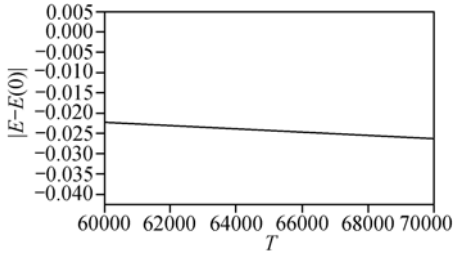


图 45　Runge-Kutta 算法的能量绝对误差

(vii) 扰动的 Kepler 问题:

模型:　$H(p,q)=\frac{1}{2}{p_1}^2+\frac{1}{2}{p_2}^2-\frac{1}{\sqrt{{q_1}^2+{q_2}^2}}-\frac{1}{400}\frac{1}{({q_1}^2+{q_2}^2)^{3/2}}$.

参数选择: 初始坐标为　$p_1=0, p_2=2, q_1=0.4, q_2=0$，能量为−0.53906，角动量为 0.8.

步长为 0.1，在 Pentium IV 1.7GHz 机上运算计算 1 万步，所需时间为: 代数动力学: 3.100358 s; 辛算法: 0.777701 s; Runge-Kutta 算法: 0.797791 s.

计算结果:

(1) 相图: 由于系统的势能有扰动项，辛算法和代数动力学的相图不闭合，

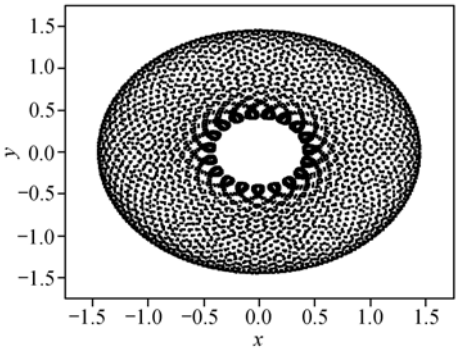


图 46　辛算法的相图

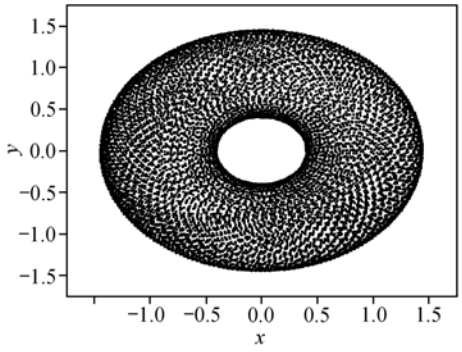


图 47　代数动力学算法的相图

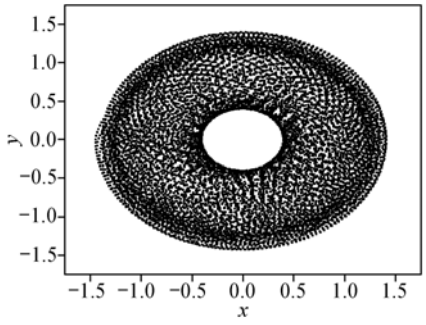


图 48　Runge-Kutta 算法的相图

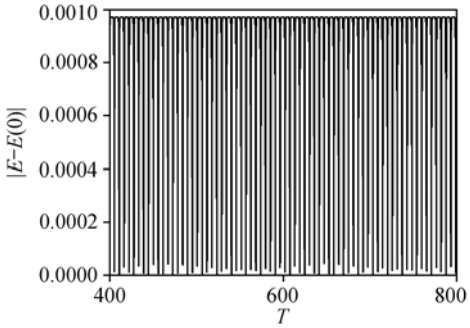


图 49　辛算法的能量绝对误差

Runge-Kutta 算法的相图还有耗散性收缩(图 46~48).

(2) 能量: 代数动力学算法的能量绝对误差的精度比辛算法高一个量级, 比 Runge-Kutta 算法高两个量级(图 49~51).

(3) 角动量: 对这个模型, 轨道保辛使辛算法计算的角动量精确守恒(图 52~54).

(viii) Toda 格点模型: 模型

$$H(p,q)=\frac{1}{2}(p_1^{\,2}+p_2^{\,2})+\frac{1}{24}(\exp(2q_2+2\sqrt{3}q_1)+\exp(2q_2-2\sqrt{3}q_1)+\exp(-4q_2))-\frac{1}{8}.$$

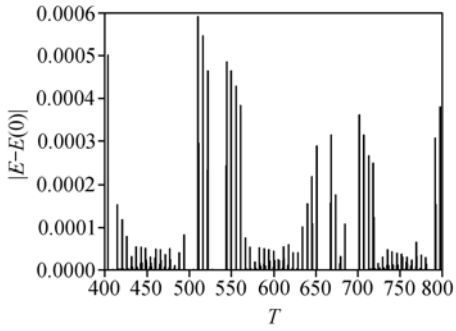


图 50 代数动力学算法的能量绝对误差

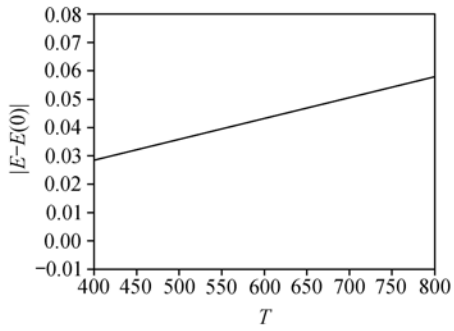


图 51 Runge-Kutta 算法的能量绝对误差

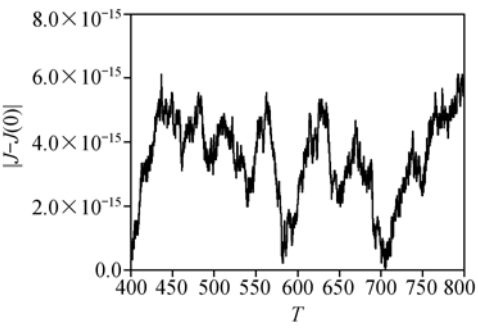


图 52 辛算法的角动量的绝对误差

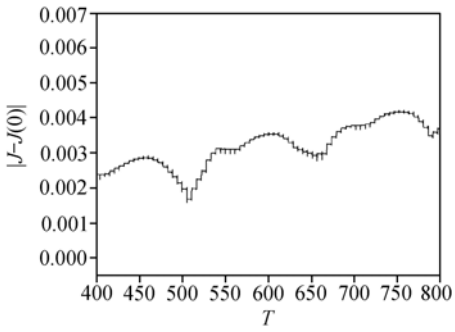


图 53 代数动力学算法的角动量的绝对误差

参数选择: 取初始值为 $p_1=0.1, p_2=1.4, q_1=0.1, q_2=0.0$, 步长取为 0.1, 计算 poincare 截面, 当 $(p_1>0$且$q_1=0)$ 时记录 (q_2,p_2), 记录 5000 个 poincare 点. 在 Pentium IV 1.7GHz 机上运行, 计算耗费时间为: 代数动力学: 16.270767 s; 辛算法: 2.337313 s; Runge-Kutta 算法: 3.376811 s.

计算结果:

(1) poincare 截面: 辛算法和代数动力学算法所得的 poincare 截面是光滑的闭合曲线, 而 Runge-Kutta 算法所得的 poincare 截面不连续(见图 55~57).

(2) 能量: 代数动力学算法的能量相对精度比辛算法高一个量级, 比 Runge-Kutta 算法高 3 个量级(图 58~60).

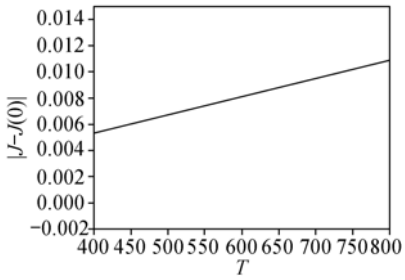


图 54　Runge-Kutta 算法的角动量的绝对误差

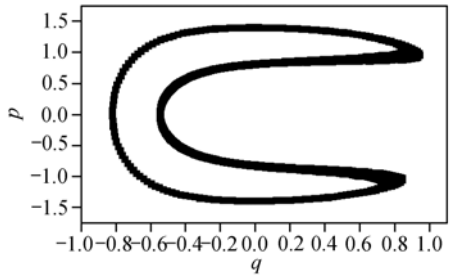


图 55　辛算法的 poincare 截面

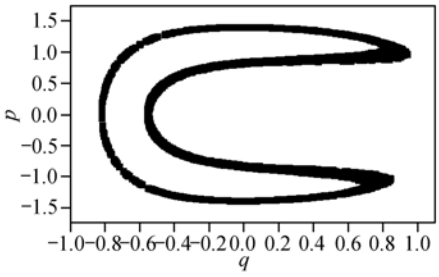


图 56　代数动力学算法的 poincare 截面

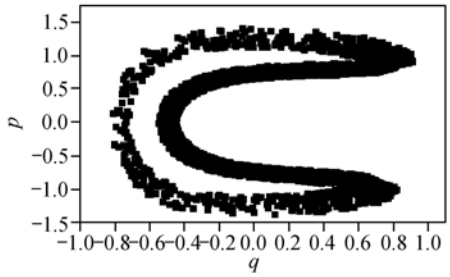


图 57　Runge-Kutta 算法的 poincare 截面

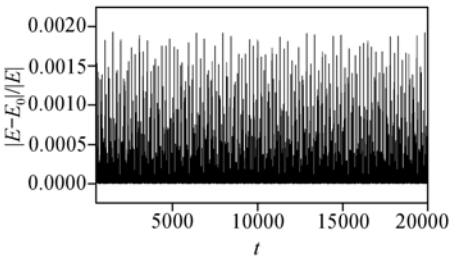


图 58　辛算法的能量相对误差

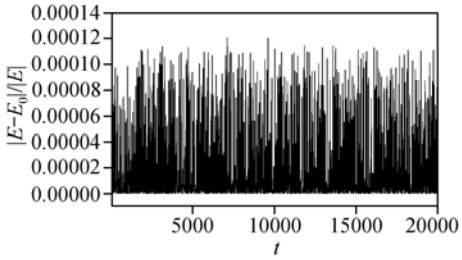


图 59　代数动力学算法的能量相对误差

(ix) Henon-Heiles 模型: 模型: $H(p,q)=\frac{1}{2}(p_1^2+p_2^2)+\frac{1}{2}(q_1^2+q_2^2+2q_1^2q_2-\frac{2}{3}q_2^3)$. 计算混沌临界点处的 poincare 截面和能量. 步长为 0.05, 计算 6000 个点的 poincare 截面. 初始值为: $p_1=0.4764951408$, $p_2=0.1$, $q_1=-0.2$, $q_2=0.2$, $E_0=1/8$. 在 Pentium IV 1.7GHz 机上三种算法耗时分别为: 代数动力学: 13.847015 s; 辛算法: 3.839117 s; Runge-Kutta 算法: 7.692915 s. 计算结果:

(1) 轨道: 代数动力学算法的精度高出辛算法 3 个量级(与辛算法相位移动和动力学失真有关, 见图 61, 以代数动力学算法为基准, 其精度为 10^{-7}).

(2) poincare 截面：三种算法生成的 poincare 截面的宏观结构相似，由于精度不同，其微观结构不同(图 62~64).

(3) 能量：代数动力学算法的能量相对误差精度比辛算法高一个量级，比 Runge-Kutta 算法高三个量级(图 65~67).

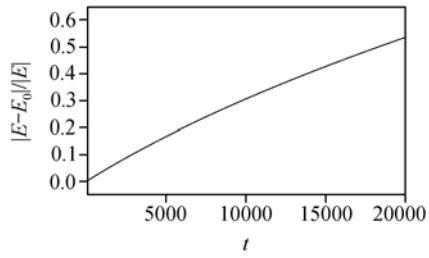


图 60 Runge-Kutta 算法的能量相对误差

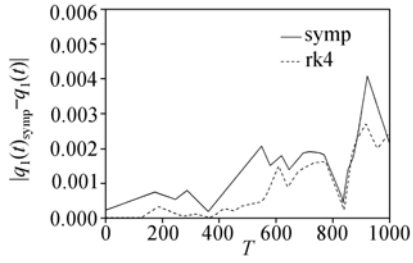


图 61 三种算法轨道之差的比较

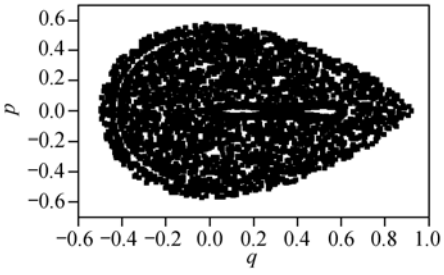


图 62 辛算法的 poincare 截面

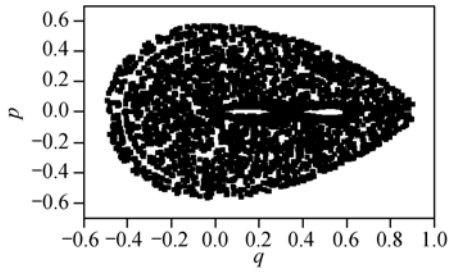


图 63 代数动力学算法的 poincare 截面

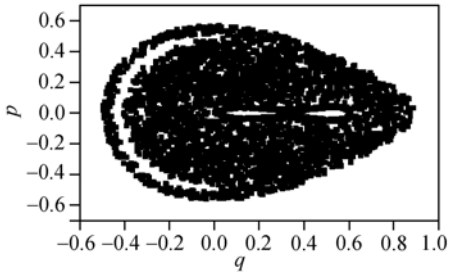


图 64 Runge-Kutta 算法的 poincare 截面

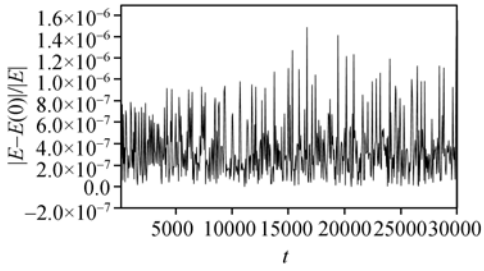


图 65 辛算法的能量相对误差

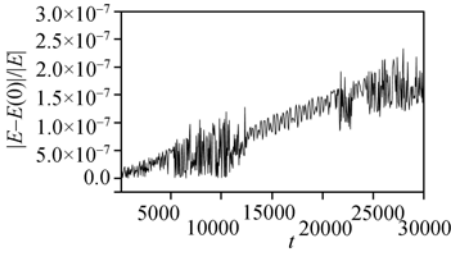


图 66 代数动力学算法的能量相对误差

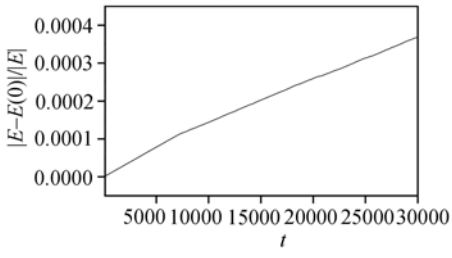


图 67 Runge-Kutta 算法的能量相对误差

(ⅹ) Lorenz 系统: 模型: $\frac{\mathrm{d}x}{\mathrm{d}t}=-\sigma(x-y)$, $\frac{\mathrm{d}y}{\mathrm{d}t}=-xz+rx-y$, $\frac{\mathrm{d}z}{\mathrm{d}t}=xy-bz$. 分别计算规则运动和混沌运动的轨道. 对于规则运动, 参数为 $\sigma=10, b=\frac{8}{3}, r=150$; 对于混沌运动, 参数为 $\sigma=10, b=\frac{8}{3}, r=28$. 初值选取为($x$ = 0.1, y = 1.0, z = 1.0). 步长为 0.01, 分别使用 Runge-Kutta 算法和代数动力学算法计算, 计算 1 万步. 计算时间, 对于代数动力学算法和 Runge-Kutta 算法分别为: 0.3222 s 和 0.0938 s. 计算结果:

(1) 规则运动区: Runge-Kutta 算法和代数动力学算法计算所得的轨道宏观上相似, 但数值细节差别很大. 代数动力学算法的精度为 10^{-7}, Runge- Kutta 算法对代数动力学算法的偏离很大(图 68~71).

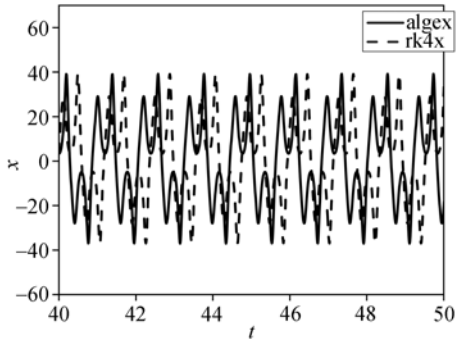


图 68　在规则区两种算法计算的 $x(t)$

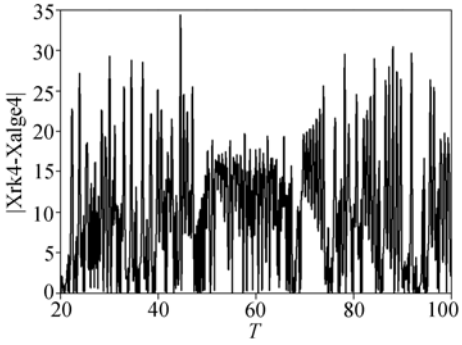


图 69　两种算法计算的 $x(t)$之差

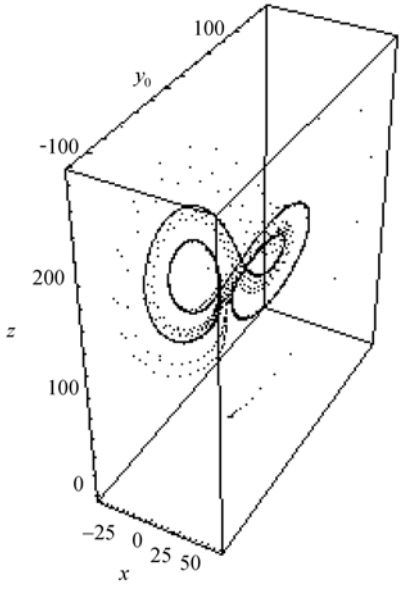


图 70　Runge-Kutta 算法的 x, y, z 三维轨道

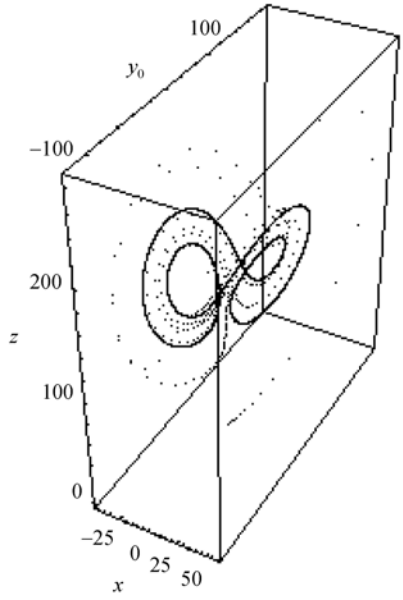


图 71　代数动力学算法的 x, y, z 三维轨道

(2) 混沌运动区: Runge-Kutta 算法和代数动力学算法所得的轨道宏观上相似, 但数值细节差别更大. 代数动力学算法的精度为 10^{-7}, Runge-Kutta 算法对代数动力学算法的偏离很大(图 72~75).

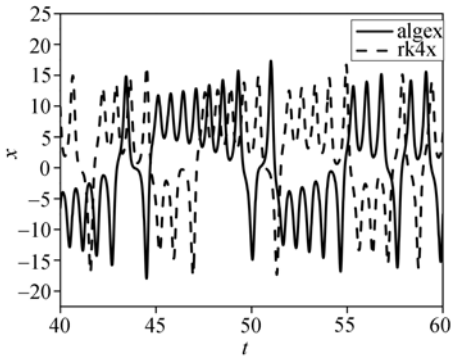


图 72 在混沌区两种算法计算的 $x(t)$

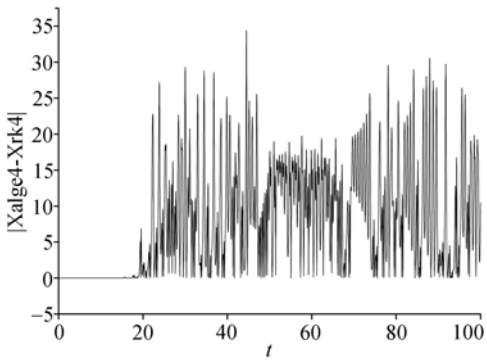


图 73 两种算法计算的 $x(t)$之差

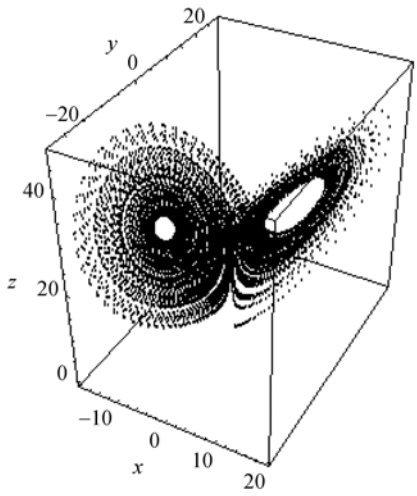


图 74 Runge-Kutta 算法的 x, y, z 三维轨道

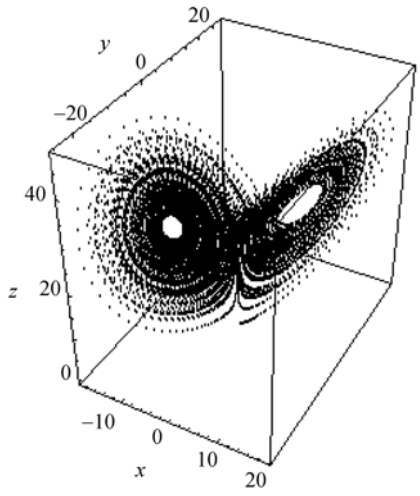


图 75 代数动力学算法的 x, y, z 三维轨道

(xi) 复合代数系统: 模型: $H(j)=\frac{1}{2}\sum_{i=1}^{3} j_i{}^2+\frac{\varepsilon}{2}\sum_{i\neq j=1}^{3} j_i j_j$. 其中 $j_1=zp_y-yp_z$, $j_2=xp_z-zp_x$, $j_3=yp_x-xp_y$. 初值选取: 坐标为 $p_1=1.0$, $p_2=0.0$, $p_3=1.0$, $q_1=1.0$, $q_2=-1.0, q_3=0.5$, 能量 $E=1.35$. 步长为 0.4, 计算 100 万步, 计算时间为: 代数动力学: 3.464695 s; Runge-Kutta: 2.491180 s; 辛中点 Euler: 22.636590 s. 计算结果:

(1) 轨道: Runge-Kutta 算法和代数动力学算法与解析解只有微小差别, 辛算法的轨道相位发生较大移动, 对精确解的偏离较大(图 76).

(2) 能量: 代数动力学算法能量误差接近机器误差, 辛算法的能量误差来源于隐式算法采用的迭代算法(Newton-Rapson 算法)(图 77~80).

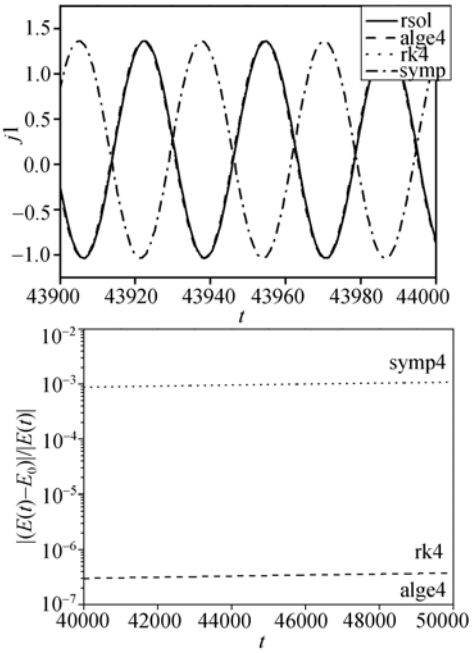


图 76　三种算法计算的角量 $j_1(t)$与精确解的比较

图 77　三种算法能量的相对误差比较

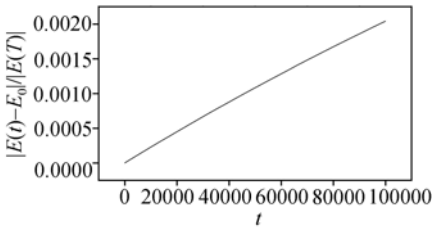


图 78　辛算计算的能量相对误差

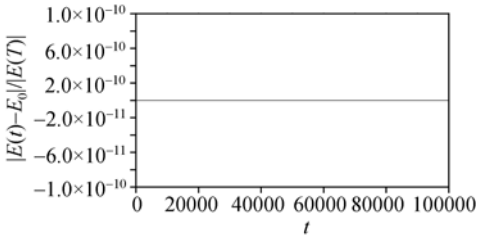


图 79　代数动力学算法的能量相对误差

(xii) 椭球测地线方程:

模型: Lagrange 为 $L=T-V=\frac{1}{2}(\dot{x}^2+\dot{y}^2+\dot{z}^2)-\lambda\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}-1\right)$, 由变分原理得到运动方程:

$$\ddot{x}=-\frac{2\lambda}{a^2}x=-{\omega_x}^2x,\ \ \ddot{y}=-\frac{2\lambda}{b^2}y=-{\omega_y}^2y,\ \ \ddot{z}=-\frac{2\lambda}{c^2}z=-{\omega_z}^2z,\ \ \frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1.$$

参数选择: 选取有理数频率比: $2\omega_x=2\omega_y=\omega_z$. 初值为: $x=0,\ \dot{x}=1.0,\ y=1.0,\ \dot{y}=0.0, z=0, \dot{z}=1.0$. 步长 = 0.5, 计算 10 万步的时间为: 辛算法: 0.87 s; 代数动力学: 1.47 s; Runge-Kutta: 1.32 s. 计算结果:

(1) 轨道: 代数动力学算法与解析解基本重合, 辛算法有相位移动, 偏离较大(图 81~82).

(2) 相图: 辛算法和代数动力学算法计算的 x-y 平面相图宏观相似, 但与 Runge-Kutta 算法计算的 x-y 平面相图差别大. 相图消去了时间, 掩盖了三者所固有的较大的轨道动力学细节差别(图 83~85).

(3) 能量: 代数动力学算法的能量相对精度极高, 辛算法的能量相对精度低(误差大与其相位移动和动力学失真有关, 图 86~88).

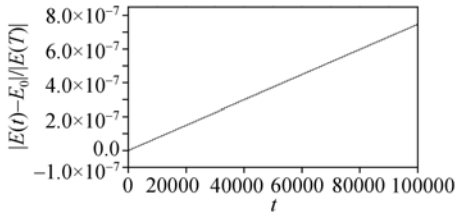


图 80 Runge-Kutta 算法算法的能量相对误差

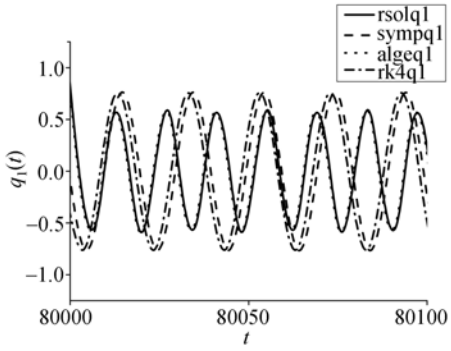


图 81 三种算法的 $x(t)$与精确解的比较

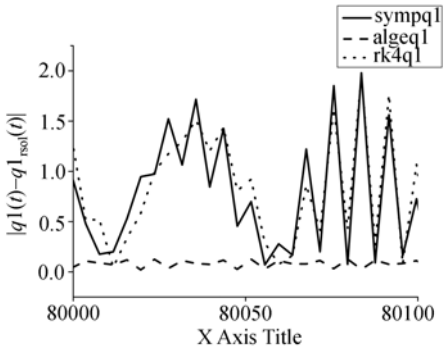


图 82 三种算法的 x(t)的绝对误差

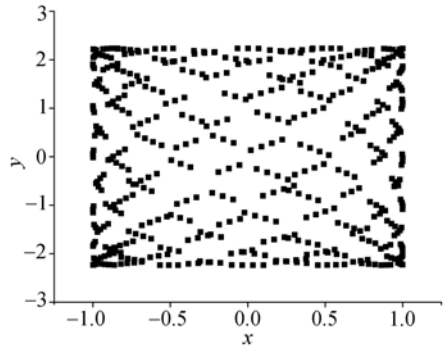
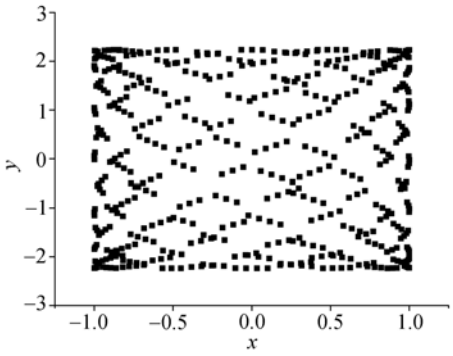


图 83 辛算法的相图



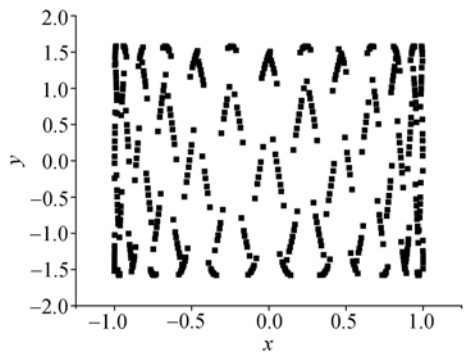
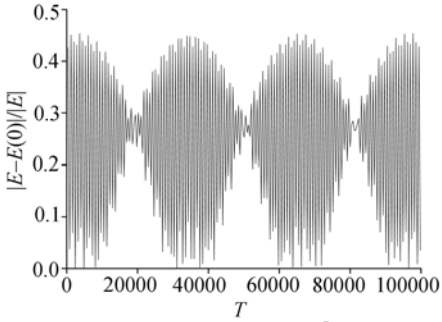


图 84　代数动力学算法的相图

图 85　Runge-Kutta 算法的相图



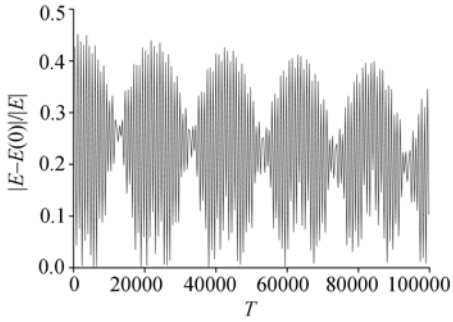


图 88　Runge-Kutta 算法的能量相对误差

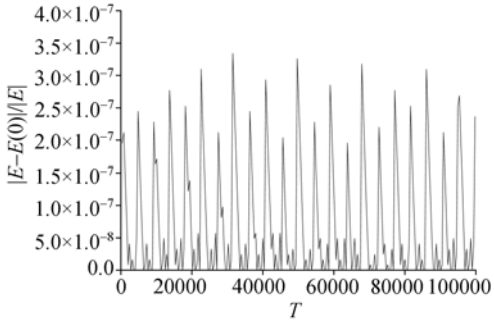


图 86　辛算法的能量相对误差　　图 87　代数动力学算法的能量相对误差

4　讨论

4.1　算法的科学要求与代数动力学算法

在科学的数值计算中，除了计算结果的科学保真外，还应从算法的简易性和效率等方面考虑. 从科学计算的上述要求，考察一下代数动力学算法. (ⅰ) 科学性. 科学算法应尽可能保持时间平移微分算子和时间演化整体积分算子的完整性，尽可能保持其精度，代数动力学算法强调了这点. (ⅱ) 保结构性. 动力学方

程的某些重要结构的保持, 如辛结构的保持, 可以基于时间平移算子 $\hat{L}$, 设计出特殊的、严格保结构的、近似的时间演化算子来实现, 如辛代数动力学算法 sU_N [1)] 就是基于 $\hat{L}$ 的完整性、具有 N 阶精度的、严格的保辛算法. (iii) 精度可估算、可控制性. 用时间平移算子的 Taylor 级数表示精确解, 就可以估算 Taylor 级数的收敛半径和由余项决定的计算精度. (iv) 算法设计可机械化. 用时间平移算子的 Taylor 级数表示精确解, 就可以用计算机进行 Taylor 级数系数函数的计算, 实现算法设计的机械化与自动化. (v) 算法的高效率. 代数动力学算法由于计算 Taylor 级数系数函数较费时间, 它与显式辛算法和显式 Runge-Kutta 算法相比, 要慢一些. 但可以通过算法格式的优化提高实际计算的效率. 近期的研究表明, 在高精度代数动力学算法 eU_N [2)]的基础上, 可以设计出速度与快速 Runge-Kutta 算法一样、输出信息量比它大 3 倍的、可以在速度和精度上自由选择的代数动力学算法 [1)].

4.2 对上述 12 个模型的计算结果的比较分析

(i) 对 Hamilton 系统的 10 个考题, 代数动力学算法总的结果好一些, 精度高一些, 并且能估计和控制计算精度.

(ii) 辛算法强调轨道几何保真, 其代价是相位和动力学失去真, 对动力学临界点附近(E~0)和高阶动量相关势, 结果不好. 但对于只强调运动轨道几何构形的科学问题, 辛算法仍是一种很好的算法. 研究表明, 通过隐式辛代数动力学算法, 可以减小相位失真, 提高精度.

(iii) Runge-Kutta 算法不能控制复杂的余项, 造成人为耗散和较大的轨道误差与较差的能量精度. 但 Runge-Kutta 算法简易、复杂性低、速度快, 对于动力系统短期行为的研究, 仍是一个广泛被使用的算法.

(iv) 代数动力学算法能在较高精度下兼顾轨道几何保真和速度、加速度、能量等动力学量保真, 对动力学临界点附近和速度相关势, 结果也较好.

(v) 代数动力学算法可用于具有复合代数结构的系统和非 Hamilton 系统.

(vi) 代数动力学算法设计可实现计算机编程自动化.

(vii) 代数动力学算法还需要进一步优化计算 Taylor 级数系数的程序, 以提高计算速度. 与此同时, 朴素代数动力学算法 U_N, 辛代数动力学算法 sU_N 和高精

1) 王顺金, 张华. 物理计算的保真与代数动力学算法: 辛代数动力学算法. 2006(待发表)

2) 王顺金, 张华. 物理计算的保真与代数动力学算法: 高精度(全保真)代数动力学算法, 四川大学内部报告

1) 王顺金, 张华. 物理计算的保真与代数动力学算法: 兼顾速度和精度的代数动力学算法, 四川大学内部报告

度代数动力学算法 eU_N, 应当根据实际问题的要求, 相互配合使用.

致谢　感谢中国科学院理论物理研究所郭汉英教授对我们这项工作的关心和支持;中国科学院计算数学研究所的秦孟兆、唐贻发、孙耿教授, 中国科学院理论物理研究所吴可和杜孟利教授, 中国科学院高能物理研究所鞠长胜教授, 四川大学数学系的李安民、吕涛和张伟年教授, 对我们的工作提出许多中肯的批评和有益的建议, 作者在此对他们表示衷心的感谢.

参　考　文　献

1 王顺金, 张　华. 物理计算的保真与代数动力学算法: I 动力学系统的代数动力学解法与代数动力学算法. 中国科学, G辑, 2005, 35(6): 573~608[摘要]

2 Sun G. A simple way constructing symplectic Runge-Kutta methods. J Comput Math, 2000, 18: 61

3 Sun G. Construction of high order symplectic Runge-Kutta methods. J Comput Math, 1993, b(11): 365

4 Feng K. Difference scheme for Hamiltonian formalism and symplectic geometry. J Comput Math, 1986, 4(3): 279~289

5 Feng K, Wu H M, Qin M Z, et al. Construction of canonical difference scheme for Hamiltonian formalism via generating functions. J Comput Math, 1989, 7(1): 71~96

6 冯　康, 秦孟兆. 辛几何算法. 杭州: 浙江科学技术出版社, 2003

7 Guo H Y, Li Y Q, Wu K, et al. Difference discrete variational principles, Euler-Lagrange cohomology and symplectic, multisymplectic structures I: difference discrete variational principle. Commun Theor Phys, 2002, 37: 1

8 Guo H Y, Li Y Q, Wu K, et al. Difference discrete variational principle, Euler-Lagrange cohomology and symplectic, multisymplectic structures II: Euler-Lagrange cohomology commun. Theor Phys, 2002, 37: 129

9 Guo H Y, Li Y Q, Wu K, et al. Difference discrete variational principles, Euler-Lagrange cohomology and symplectic, multisymplectic structures III: application to symplectic and multisymplectic algorithms. Commun Theor Phys, 2002, 37: 257

10 王顺金. 人造量子系统的理论研究与代数动力学. 物理学进展, 1999, 19(4): 3313~3373