基于广义 Foliation 条件的非线性映射二维流形计算*

李慧敏† 樊养余 孙恒义 张菁 贾蒙

(西北工业大学电子信息学院, 西安 710072)

(2011 年 3 月 21 日收到; 2011 年 4 月 10 日收到修改稿)

主要研究非线性映射函数双曲不动点的二维流形计算问题. 提出了推广的 Foliation 条件, 以此来衡量二维流形上的一维流形轨道的增长量, 进而控制各子流形的增长速度, 实现二维流形在各个方向上的均匀增长. 此外, 提出了一种一维子流形轨道的递归插入算法, 该算法巧妙地解决了二维流形面上网格点的插入、前像搜索, 以及网格点后续轨道计算问题, 同时插入的轨道不必从初始圆开始计算, 避免了在初始圆附近产生过多的网格点. 以超混沌三维 Hénon 映射和具有蝶形吸引子的 Lorenz 系统为例验证了算法的有效性.

关键词: 非线性映射, 稳定和不稳定流形, 三维 Hénon 映射, Lorenz 系统

PACS: 95.10.Fh, 52.25.Gj, 82.40.Bj

1 引 言

稳定流形和不稳定流形对于分析非线性系统的动力学特性起着非常重要的作用. 它们与同宿、异宿以及混沌 [1] 等复杂动力学行为的出现有着密切关系, 例如, 双曲不动点的稳定和不稳定流形的同宿相交将引发混沌. 稳定流形构成了不动点的稳定区域的边界, 通过计算稳定流形, 可以估计系统的鲁棒性: 不动点距边界越近, 系统就越容易失稳, 反之则越稳定. 由于流形一般难以用解析的方法来计算, 所以其数值算法的研究就显得非常必要.

向量场的流形计算方法可参见文献 [2—10]. 本文主要研究非线性映射二维流形的计算问题. 非线性映射的流形计算比较困难, 与向量场的流形计算相比, 它的一个重要特点是轨道 (orbit) 上新网格点对轨道上已有的网格点存在依赖性, 新网格点必须是通过已有网格点或其插值经过映射后才能得到; 所以计算过程中必须不断搜索并保存流形网络点的 (前) 像位置. 这种依赖性的存在, 使得非线性映射的流形计算变得更为困难. 计算二维流形时需解决两个主要问题: 一是保证二维流形在各向上的均匀增长, 二是处理好流形曲面上的网格点插入问题. 对于二维流形, 如果要进行网络点的插入, 除了计算插值点之外, 还要考虑如何计算插值点之后的流形轨道, 并且因为后续流形的计算对于插值点之前的轨道存在 "依赖性", 所以同时必须对插值点之前原本不存在的轨道也进行插值, 从而使计算变得尤为复杂.

一维流形的计算方法可参见文献 [11—17]. 目前映射的二维流形的算法很少, 文献 [18] 中提出了一种应用 Foliation 条件的算法, 该算法比较好地解决了流形各向增长速度不均的问题, 实际上与向量场二维流形算法 [19] 是同一算法, 该算法默认新网格点的原像都处于当前流形边界上, 但大多数映射函数的二维流形并不满足这一条件, 所以该算法的应用将受到限制. 文献 [20] 提出了一种对二维流形进行外部估计的算法, 但计算量非常大, 而且计算结果也不太理想. 国内在相关方面的研究较少, 文献 [21] 提出一种预估流形增长速度的一维流形算法, 在此基础上, 提出了一种快速的二维流形计算方法, 但是该算法没有流形各向增长速度不均的问题进行处理.

本文算法的基本思想是用一维子流形来覆盖二维流形. 通过对 Foliation 条件进行推广, 保证了

* 国家自然科学基金 (批准号: 60872159) 资助的课题.

† E-mail: huimin_lee@126.com

二维流形在各个方向上的均匀增长. 并提出了在二维流形上插入一维子流形的算法, 解决了新网格点的原像搜索及对应轨道的插值问题.

2 流形的基本概念

设 $F: R^n \to R^n$ 为一个保向的微分同胚映射函数

$$x_{n+1} = F(x_n). \tag{1}$$

对于映射 (1), 若存在 x_0, 满足 $F(x_0) = x_0$, 则称 x_0 为映射的不动点. A 为 x_0 处的雅各比矩阵 $A = J_F(x_0) = [\partial f_i/\partial x_j](x_0)$. 若矩阵 A 的特征值的模都不等于 1, 那么 x_0 就是一个双曲不动点; 其中模小于 1 的特征值叫做稳定特征值, 其对应的特征向量 $\{v_1, v_2, \cdots, v_l\}$ 张成稳定特征空间 E^{s}; 模大于 1 的特征值叫做不稳定特征值, 它们对应的特征向量 $\{v_{l+1}, v_{l+2}, \cdots, v_n\}$ 张成不稳定特征空间 E^{u}.

定理 1 设 x_0 是保向微分同胚映射函数 F 的一个双曲不动点, 则在 x_0 的邻域 U 内存在局部稳定和不稳定流形

$$W^{\mathrm{s}}_{\mathrm{loc}}(x_0) = \{x \in U : \lim_{k\to+\infty} F^k(x) \to x_0\}, \tag{2}$$

$$W^{\mathrm{u}}_{\mathrm{loc}}(x_0) = \{x \in U : \lim_{k\to-\infty} F^k(x) \to x_0\}, \tag{3}$$

在 x_0 处, $W^{\mathrm{s}}_{\mathrm{loc}}(x_0), W^{\mathrm{u}}_{\mathrm{loc}}(x_0)$ 分别与 E^{s} 和 E^{u} 相切.

定理的证明见文献 [22]. 全局稳定和不稳定流形定义为

$$\begin{aligned} W^{\mathrm{s}}(x_0) &= \{x \in R^n : \lim_{k\to+\infty} F^k(x) \to x_0\} \\ &= \bigcup_{i=0}^{-\infty} F^i(W^{\mathrm{s}}_{\mathrm{loc}}(x_0)), \end{aligned} \tag{4}$$

$$\begin{aligned} W^{\mathrm{u}}(x_0) &= \{x \in R^n : \lim_{k\to-\infty} F^k(x) \to x_0\} \\ &= \bigcup_{i=0}^{+\infty} F^i(W^{\mathrm{u}}_{\mathrm{loc}}(x_0)). \end{aligned} \tag{5}$$

3 二维流形的计算

二维流形面可以通过一维流形的覆盖来表示, 现在要解决的问题有两个: 一是如何确保二维流形在各个方向上的均匀增长, 二是流形曲面上的插值方法.

3.1 推广的 Foliation 条件

Foliation 条件在文献 [18] 中率先提出, 它通过使流形环面每步向外延伸相同的长度来实现二维流形在各个方向上的均匀增长. 由于二维流形与一个平面是拓扑等价的, 所以如果将二维流形" 展平", 则可以看出, Foliation 条件是以画圆的思想来保证二维流形在各个方向的均匀增长, 这个圆的中心为不动点, 而圆的半径每步向外增加相同长度. 但二维流形在本质上是由一维流形轨道" 铺成" 的, 而且映射系统流形计算具有" 依赖性", 所以在实际计算过程中直接利用 Foliation 条件非常不便. 下面我们将从控制流形轨道的增长速度入手, 提出推广的 Foliation 条件.

首先说明流形轨道上叶形的定义方法: 如图 1 所示, 双曲不动点 x_0 的二维局部 (不) 稳定流形面由两个 (不) 稳定特征值对应的特征向量 V_1 和 V_2 张成, $\boldsymbol{n} = V_1 \times V_2$ 为局部流形面的法向量, 对于流形轨道上任意一点 x_k, 由向量 $\overrightarrow{x_0x_k}$ 和 $\boldsymbol{n}$ 确定了过点 x_k 的平面, 即为该点处的叶形.

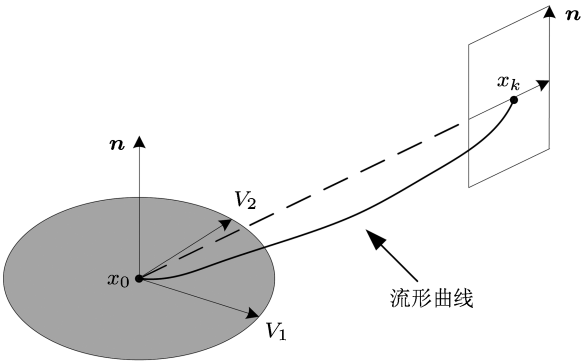

图 1 推广的 Foliation 条件中叶形的定义

推广的 Foliation 条件内容如下: 取流形轨道上任意一点, 对该点处的切向分量向该点处的叶形进行投影, 沿轨道对投影值进行积分, 所得结果就是该流形轨道相对于不动点 x_0 的 Foliation 弧长. 若对二维流形面上的所有一维流形轨道都计算相等的 Foliation 弧长, 就能够保证二维流形的初始部分的均匀增长, 但是当流形具有比较特殊的结构时, 控制效果将会变差, 这点我们将在仿真过程中进行详细讨论.

由于计算过程中流形轨道通常都是用一系列的离散点来表示, 是分段线性的, 所以 Foliation 弧长的积分可以用下面的公式来表示:

$$S_{\mathrm{Foliation}} = \int_L \left\| T(x) \times \frac{(n \times \overrightarrow{x_0x})}{\|(n \times \overrightarrow{x_0x})\|} \right\| \mathrm{d}s$$

$$\approx \sum_{i=0}^{m-1} \int_{x_i}^{x_{i+1}} \left\| T(x) \times \frac{(n \times \overrightarrow{x_0 x})}{\|(n \times \overrightarrow{x_0 x})\|} \right\| \mathrm{d}s, \tag{6}$$

其中 $T(x)$ 表示流形轨道在点 x 处的切分量. 在流形轨道的每个线性段上, 再使用梯形公式对原积分式进行近似计算, 可得

$$\int_{x_i}^{x_{i+1}} \left\| T(x) \times \frac{(n \times \overrightarrow{x_0 x})}{\|(n \times \overrightarrow{x_0 x})\|} \right\| \mathrm{d}s \approx \frac{1}{2} \sum_{j=0}^{1} \left\| (\overrightarrow{x_{i+1}} - \overrightarrow{x_i}) \times \frac{(n \times \overrightarrow{x_0 x_{i+j}})}{\|(n \times \overrightarrow{x_0 x_{i+j}})\|} \right\|. \tag{7}$$

3.2 二维流形面上的轨道插值

在计算二维流形的过程中, 若相邻两条一维子流形轨道之间的距离过大, 就要在两条轨道间插入一条新的一维子流形轨道, 以此来保证二维流形计算的精度. 下面将介绍轨道插值方法.

首先声明任意两条一维子流形轨道的 Foliation 弧长是相等的. 为了便于对两条轨道进行比较, 要先对原始轨道 (直接使用一维流形计算方法计算得到的轨道) 进行规整化, 方法如下: 以原轨道上的离散点为基础, 进行网格点的线性插值, 使得新网格点的 Foliation 弧长为 $k \times \mathrm{step}$ $(k = 0, 1, 2, \cdots)$, 其中 step 规定了网格的大小, 可以根据需要进行调整. 与此对应, 新网格点的 (原) 像位置也要进行相应的调整, 并同比插值出其 Foliation 弧长. 这样对轨道进行整理后, 任意两条轨道上的网格点就构成了一一对应关系, 从而使我们能够对两条轨道进行比较.

下面主要介绍二维稳定流形的计算, 二维不稳定流形只需考虑逆映射 F^{-1} 即可. 如图 2 所示, 相邻的两条一维子流形轨道 Orbit_1 和 Orbit_2 的距离大于 $\mathrm{SIZE}_{\max}$, 不满足精度条件, 要在两轨之间插入一条轨道 Orbit_{12}. 步骤如下:

1) 对齐轨道起点: 当 Orbit_1 和 Orbit_2 的轨道起始点的 Foliation 弧长相等时, 这一步是可以省略的. 但有些情况下, 插入 Orbit_{12} 后, 轨道间的距离还是不能满足条件, 需要在 Orbit_{12} 和 Orbit_1 之间, 或者是 Orbit_{12} 和 Orbit_2 之间继续插入轨道, 这种情况下的插值的特殊之处在于, 两条轨道的出发点的 Foliation 弧长并不一定是相等的, 不能对两条轨道直接进行比较. 为了统一这两种情况, 要首先对齐两条轨道的起点. 若一条轨道的起点处的 Foliation 弧长为 $k \times \mathrm{step}$, 而另一条轨道的起点处的 Foliation 弧长为 $(k + \mathrm{shift}) \times \mathrm{step}$, 那么就要用第一条轨道的第 $k + \mathrm{shift}$ 个点和第二条轨道的第 k 个点对应进行比较, 即把第一条轨道上的第 $k + 1$ 个点当作轨道的起点.

2) 找插值起点: 从对齐的轨道起点开始, 依次计算两条轨道上对应点之间的距离, 若第 i 个点对的距离大于精度值 d_{insert}, 例如图 2 中点 A 和 B 构成的点对, 取这两个点的中点 C 作为插值起点.

3) 对 "最小相关区间" 进行插值: 如本文开始所提到的, 映射系统的流形计算对已有流形段具有依赖性, 点 C 之后的轨道的计算是依赖于它之前的轨道的, 因此要对它之前的轨道进行插值. 这里我们提出 "最小相关区间" 这个概念, 称点 C 和点 $F(C)$ 之间的轨道为点 C 之后的轨道计算的 "最小相关区间", 很简单就可以证明, 只要对 "最小相关区间" 进行插值, 而无需对点 C 之前的整条轨道进行插值, 后续轨道就能够完整地计算.

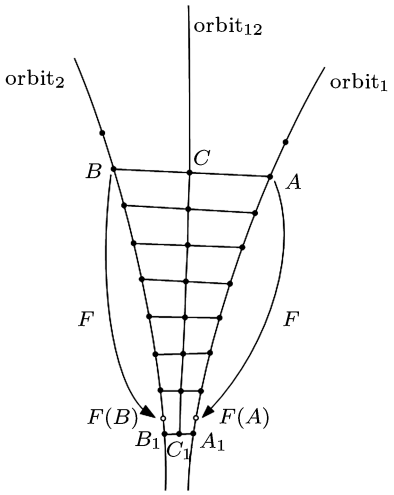

图 2 "最小相关区间" 的插值

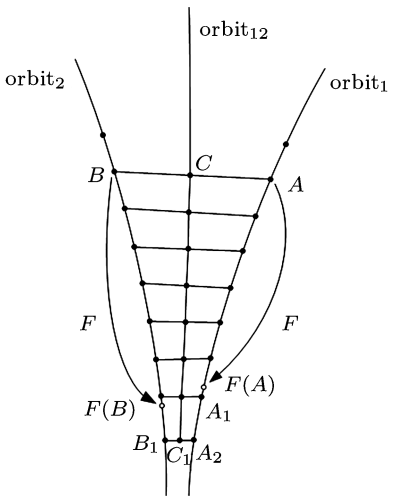

图 3 "最小相关区间" 的插值

如图 2 所示, 我们分别找到点 A 和点 B 的像 $F(A)$ 和 $F(B)$, 并确定它们所处的区间段, 选取所处区间的 Foliation 弧长较小的端点 A_1 和 B_1, 取中点 C_1, 然后对 C_1 和 C 之间的流形段进行插值 (由 Orbit_1 和 Orbit_2 进行插值), 并标注相应的 Foliation 弧长, 相应的, 它们的像以及像的 Foliation 弧长都要进行计算和插值. C_1 和 C 之间的流形段就包含了 "最小相关区间".

有时候会出现点 $F(A)$ 和 $F(B)$ 不在同一个 Foliation 弧长区间上. 如图 3 所示, 点 $F(B)$ 所处的 Foliation 弧长区间点 $F(B)$ 靠前, 为了保证包含 "最小相关区间", 则要从 A_2 和 B_1 的位置开始插值.

4) 使用一维流形算法计算点 C 之后的流形轨道, 使最终的 Foliation 弧长与 Orbit_1 和 Orbit_2 的相等, 随后对新轨道进行规整化, 使轨道各离散点的 Foliation 弧长为 step 的整数倍.

5) 判断 Orbit_1 和 Orbit_{12} 之间的距离是否满足精度条件, 若不满足则按上述方法在这两条轨道之间继续插入轨道.

6) 判断 Orbit_2 和 Orbit_{12} 之间的距离是否满足精度条件, 若不满足则按上述方法在这两条轨道之间继续插入轨道.

上述 6 个步骤可以用一个递归程序来实现, 伪代码如下:

```
New_Orbit = ORBIT_INSERT(Orbit1,Orbit2)
    对齐轨道起点;
    找插值起点;
    对 "最小相关区间" 进行插值;
    计算轨道 Orbit12;
    if Orbit1 和 Orbit12 之间的距离不满足精度条件
    Orbit_left= ORBIT_INSERT(Orbit1,Orbit12);
    end
    if Orbit2 和 Orbit12 之间的距离不满足精度条件
    Orbit_left= ORBIT_INSERT(Orbit1,Orbit12);
    end
    New_Orbit={Orbit_left, Orbit12, Orbit_right};
    函数返回
```

3.3 二维流形计算步骤

以二维稳定流形的计算为例. 由于二维流形实际上就是许多条一维流形轨道的集合, 一维流形的计算是二维流形计算的基础, 不同于单纯的一维流形计算, 在二维流形中计算一维流形轨道时要按照 (6) 式标出轨道上离散点的 Foliation 弧长, 并同时要记录每个离散点的像的位置以及像所处的 Foliation 弧长.

以不动点 x_0 为圆心, 在局部不稳定流形上以 δ 为半径作圆, 在圆上均匀取 N 个点. 然后从这 N 个点出发, 计算出相应的一维流形轨道, 并使各轨道的 Foliation 弧长均为 ARC. 计算相邻轨道间的距离 (通过计算两条轨道上 Foliation 弧长相等的离散点对间的距离来衡量), 若其中某两条轨道间的距离大于网格最大半径 $\text{SIZE}_{\max}$, 则要进行轨道的插入, 这里分为两种情况来处理: 若两轨在初始圆上的距离大于阈值 precision, 则在初始圆上的两个出发点中间插入一个新点, 然后计算出一条新轨道; 若两轨在初始圆上的距离小于于阈值 precision, 则按照上一小节介绍的方法在这两条轨道之间再插入新的轨道.

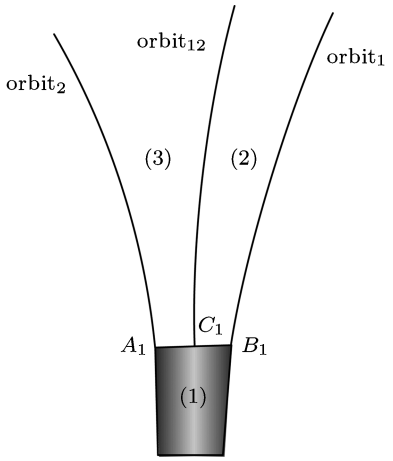

图 4 二维流形面的可视化

相比之下, 如果直接从初始圆开始计算所以插入的轨道, 则似乎不需要遵循前一小节所述的 "繁琐" 的步骤, 但是这样处理的话存在一个很严重的问题: 每条轨道在初始圆上都有一个出发点, 在弱流形方向上轨道分离比较厉害, 为了很好地覆盖二维流形, 在这个方向上就必须添加很多条轨道, 从而使得其出发点的间距变得非常小, 当间距小于计算机所能达到的精度时, 我们就无法在两条距离过大的轨道间继续插入轨道了, 从而无法完整地覆盖二维流形. 所以采用分而处之的策略, 也就是上一段提到的, 当初始圆上的间距较大时, 直接计算轨道, 而当初始圆上的间距较小时, 采用所介绍的新插值方法.

接下来讨论计算结果进行可视化, 将二维流形表示为一个曲面. 可视化的过程可以和前边的递归

程序结合起来一块进行. 借助图 4, 将前述递归过程重写为

New_Orbit = ORBIT_INSERT(Orbit$_1$,Orbit$_2$)
对齐轨道起点;
找插值起点;
对 “最小相关区间” 进行插值;
对区域 (1) 进行网络化;
计算轨道 Orbit$_{12}$;
if Orbit$_1$ 和 Orbit$_{12}$ 之间的距离不满足精度条件
Orbit_left= ORBIT_INSERT(Orbit$_1$,Orbit$_{12}$);
else 对区域 (2) 进行网络化;
end
if Orbit$_2$ 和 Orbit$_{12}$ 之间的距离不满足精度条件
Orbit_right= ORBIT_INSERT(Orbit$_1$,Orbit$_{12}$);
else 对区域 (3) 进行网络化;
end
New_Orbit={Orbit_left, Orbit$_{12}$, Orbit_right};
函数返回

3.4 关于插值精度的讨论

插值的误差主要来源于新插入的网格点, 以及以此网格点为基础进行计算时所引起的传播误差, 所以由文献 [18] 中的证明可知, 插值误差保持有界, 并且当所取精度参数趋于零时, 计算误差也趋于零.

4 算例分析

4.1 三维 Héon 映射的二维稳定流形

三维 Héon 映射 [23] 的表达式如下:

$$F\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} y \\ z \\ M_1 + Bx + M_2 y - z^2 \end{pmatrix}.$$

当 $M_1 = 1.4$, $M_2 = 0.2$, $B = 0.1$ 时, 该系统具有混沌吸引子, 见图 5, 这点与著名的二维 Hénon 映射类似. 此时系统的不动点为 $x_0 = (x^*, y^*, z^*)$, 其中 $x^* = y^* = z^* = 0.8839$. F^2 在 x_0 处的 Jacobian 矩阵为

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0.1 & 0.2 & -1.7678 \\ -0.1768 & -0.2536 & 3.3251 \end{bmatrix}.$$

矩阵 A 具有 3 个实特征值: $\lambda_1 = 3.4106$, $\lambda_2 = 0.0386$, $\lambda_3 = 0.0759$, 所以 x_0 是一个双曲不动点, 且具有一维不稳定流形和二维稳定流形.

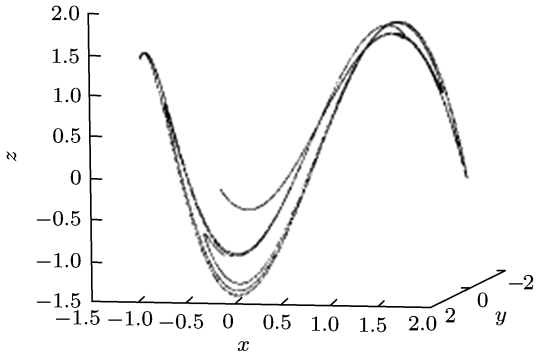

图 5 三维 Hénon 映射的混沌吸引子

计算过程中采用的精度参数如下: $\delta = 0.4$, $\mathrm{SIZE}_{\max} = 1$, $d_{\mathrm{insert}} = 0.1$, precision $= 0.001$.

为了说明本文提出的推广的 Foliation 条件能够很好地控制二维流形在初始部分的均匀增长, 先计算一段 ARC = 5 的二维稳定流形, 结果如图 6 所示. 其中不同灰度的环带之间的距离为 1, 图中的实线为实际计算过程中得到的一维流形轨道, 易见它们是 “弯曲” 着向外进行增长的, 并且不同方向的一维子流形的欧氏弧长也是不相等的. 轨道上的点则是抽取的 Foliation 弧长为 step 整数倍处的点, 容易看出推广的 Foliation 条件能够很好地保证二维流形在各个方向上的均匀增长.

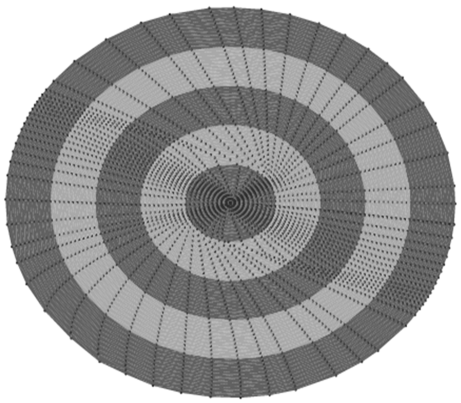

图 6 三维 Hénon 映射二维稳定流形的起始部分

下面计算 ARC = 9 的二维稳定流形, 结果如图 7 所示. 其中不同灰度的环带之间的距离为 0.8.

图 7 从不同视角展示了二维流形的计算结果. 由图 7 可以看出, 当二维流形具有比较复杂的几何结构时, 推广的 Foliation 条件并不能一直保证二维流形的均匀增长, 例如图 7(a) 右下角部分, 相邻流

形环之间的距离就非常不均匀. 这主要是由于推广的 Foliation 条件只利用了单个流形轨道的增长信息, 而未综合流形面的增长, 当流形面的增长方向与叶形接近于垂直时, 其增长量在叶形上的投影约等于零, 所以难以反映流形面的实际增长量. 但在流形的初始部分, 推广的 Foliation 条件还是能够很好地控制二维流形面的均匀增长.

由于在实际中没有必要保持流形面在各向上的绝对均匀增长, 而且二维流形面上的轨道分离主要发生在流形面的起始部分, 随着流形面的增长, 这种分离所造成的影响将减弱, 只要单纯的利用等欧氏弧长增长就可以保证流形面比较均匀的增长. 基于此, 我们在流形的起始平面部分使用 Foliation 弧长, 而在超出平面的部分采用欧氏弧长, 结合二者的优势, 这样就可以达到比较满意的效果了. 计算结果如图 8 所示.

图 7 三维 Hénon 映射的二维稳定流形 (a) 视角一; (b) 视角二

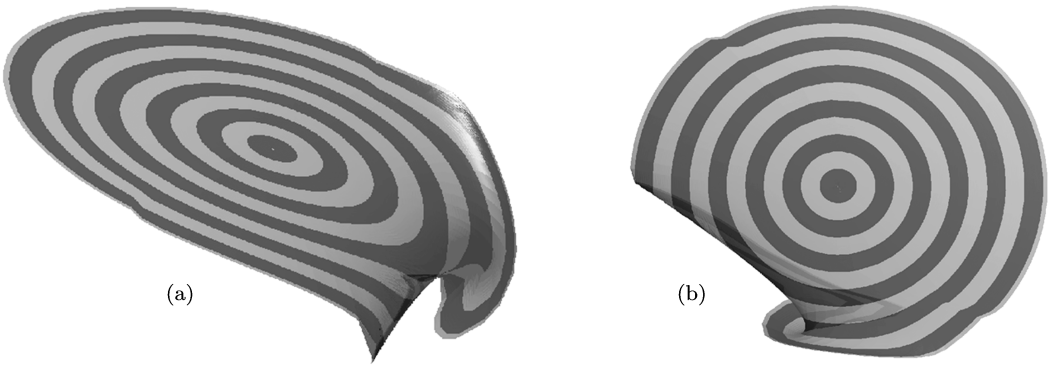

图 8 三维 Hénon 映射的二维稳定流形, 综合使用 Foliation 弧长和欧氏弧长 (a) 视角一; (b) 视角二

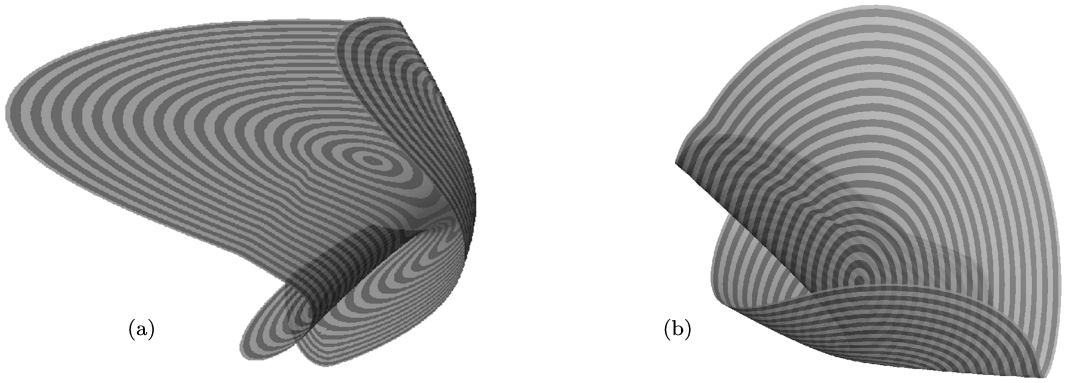

图 9 三维 Hénon 映射的二维稳定流形, 弧长为 30

与图 7 相比, 图 8 的计算结果对于流形增长的控制有了较大地改善, 总体效果要好于图 7. 下面用同样方法计算 ARC = 30 的二维稳定流形, 结果如图 9 所示.

二维稳定流形的形状类似于将一张圆形纸张进行了两次对折. 由图可见, 当流形的弧长增加时, 本文的算法仍然能够较好地控制二维流形在各个方向上的增长速度, 总体的控制效果还是不错的.

图 10 为二维流形的一部分, 其中浅色轨道是从初始圆开始计算的, 深色的轨道是用 3.2 节所提出的算法计算得到的. 由控制参数 precision = 0.001 可知, 相邻轨道的最近距离为 0.001, 图 9 中轨道总数为 369 条, 其中从初始圆出发的轨道数目为 246 条. 与此相比, 如果单纯从初始圆开始计算所要插入的轨道, 实验数据显示轨道间的最近距离约为 10^{-5}, 从初始圆出发的轨道数目为 452 条. 由此可见, 本文算法一定程度上降低了初始圆上点的密集程度, 并减少了所要计算的轨道的总数目. 在计算较大弧长的二维流形时, 本文算法的优势将更明显.

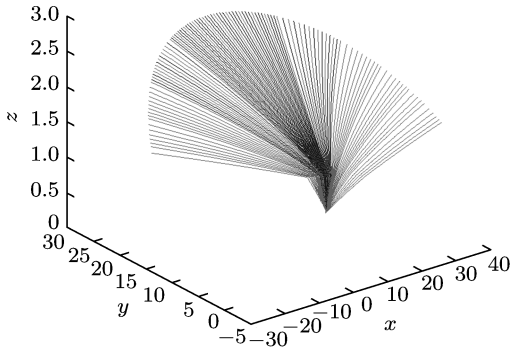

图 10 插入的轨道

4.2 Lorenz 系统的二维稳定流形

Lorenz 系统是一个大气对流模型, 以其著名的蝶形混沌吸引子而备受关注. 该模型的表达式如下:

$$\dot{x} = \sigma(y - x),$$
$$\dot{y} = \rho x - y - xz,$$
$$\dot{z} = xy - \beta z.$$

当 $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ 时, 系统的蝶形吸引子如图 11(a) 所示. 该模型是用微分方程表示的, 是一个连续系统, 我们以差分的形式将其离散化为

$$\frac{x_{n+1} - x_n}{T} = \sigma(y_n - x_n),$$
$$\frac{y_{n+1} - y_n}{T} = x_n(\rho - z_n) - y_n,$$
$$\frac{z_{n+1} - z_n}{T} = x_n y_n - \beta z_n.$$

整理后得

$$x_{n+1} = T\sigma(y_n - x_n) + x_n,$$
$$y_{n+1} = Tx_n(\rho - z_n) - Ty_n + y_n,$$
$$z_{n+1} = T(x_n y_n - \beta z_n) + z_n.$$

为了尽可能地保持原系统的性质, 需要对采样间隔 T 进行优化选取. 通过实验, 发现当 $T = 0.01$ 时, 离散系统具有与原连续系统类似的混沌吸引子 (见图 11(b)), 而且系统的演化速度适中, 比较好地保持了原系统的性质.

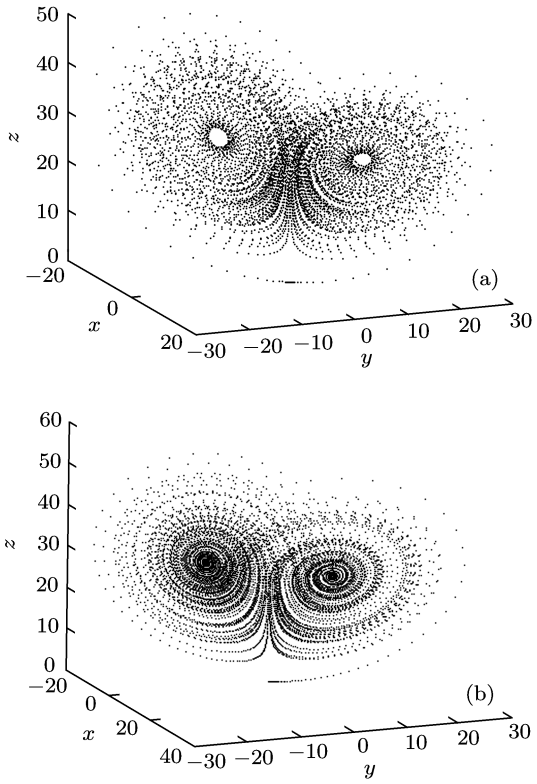

图 11 Lorenz 系统的混沌吸引子 (a) 原 Lorenz 系统混沌吸引子; (b) $T = 0.01$ 时, 离散化后的 Lorenz 系统混沌吸引子

原点是离散化后的 Lorenz 系统的一个不动点, 系统在原点处的 Jacobian 矩阵为

$$A = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.28 & 0.99 & 0 \\ 0 & 0 & 0.9733 \end{bmatrix}.$$

矩阵 A 具有 3 个实特征值: $\lambda_1 = 0.7717$, $\lambda_2 = 1.1183$, $\lambda_3 = 0.9733$, 所以 x_0 是一个双曲不动点, 且具有一维不稳定流形和二维稳定流形,

这点与原 Lorenz 系统类似. 计算过程中采用的精度参数如下: ARC = 100, $\delta = 0.5$, $\mathrm{SIZE}_{\max} = 1$, $d_{\mathrm{insert}} = 0.1$, precision = 0.001, step = 1.

计算结果如图 12 所示, 该图只使用 Foliation 弧长来控制流形的增长, 不同灰度带之间的距离为 5. 由图易见二维稳定流形总体上在各个方向上的增长是比较均匀的, 但也出现了局部增长不均的情况, 例如该图中部靠上的部分, 这也是由于流形的 “垂直” 增长造成的.

当综合使用了 Foliation 弧长和欧氏弧长时, 计算结果如图 13 所示. 与图 12 相比, 虽然流形的下半部分的控制效果有些变差, 但对于复杂曲面部分的增长有了更好地控制, 总体来说, 效果是好于图 12 的.

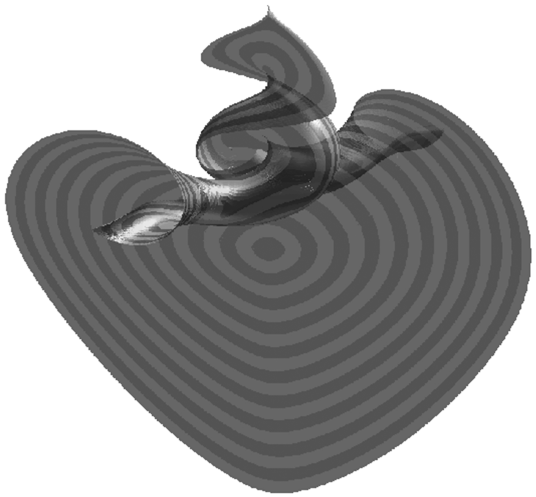

图 12 离散化后 Lorenz 系统的二维稳定流形, 以流形曲面来表示

图 14 为图 13 中流形所包含的轨道, 为了显示出细节, 分为上下两部分来表示, 其中浅色轨道是从初始圆计算出发的, 深色轨道为 3.2 节所介绍的算法计算得出的. 由图可知, 相邻两条浅色轨道间插入了不止一条深色轨道, 这说明本文的递归插入算法是有效的. 相邻轨道间的最小距离为 precision = 0.001. 弧长为 80 时, 计算的轨道总数为 1020 条, 其中从初始圆出发的轨道数目为 25. 如果不采用 3.2 节的方法, 所有的轨道都从初始圆开始计算, 此时从初始圆上出发的轨道数目为 1263 条, 实验所得的初始圆上的最小距离为 10^{-24}; 当弧长增大为 100 时, 由于相邻轨道在初始圆上的间距过小, 接近于计算所能达到的极限度值, 在原有精度条件下已经无法继续计算了. 对比可以发现, 本文的算法不仅减少了所要计算的轨道的数目, 而且有效降低了初始圆附近网格点的密集程度. 所以, 计算弧长较大的二维流形时, 本文的算法还是很具有优势并且是十分必要的.

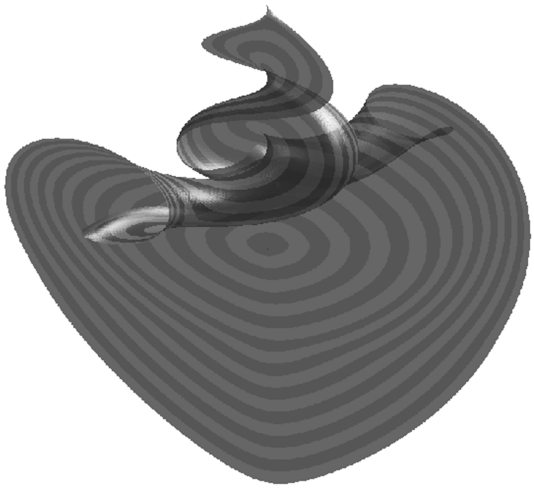

图 13 离散化后 Lorenz 系统的二维稳定流形, 综合使用 Foliation 弧长和欧氏弧长

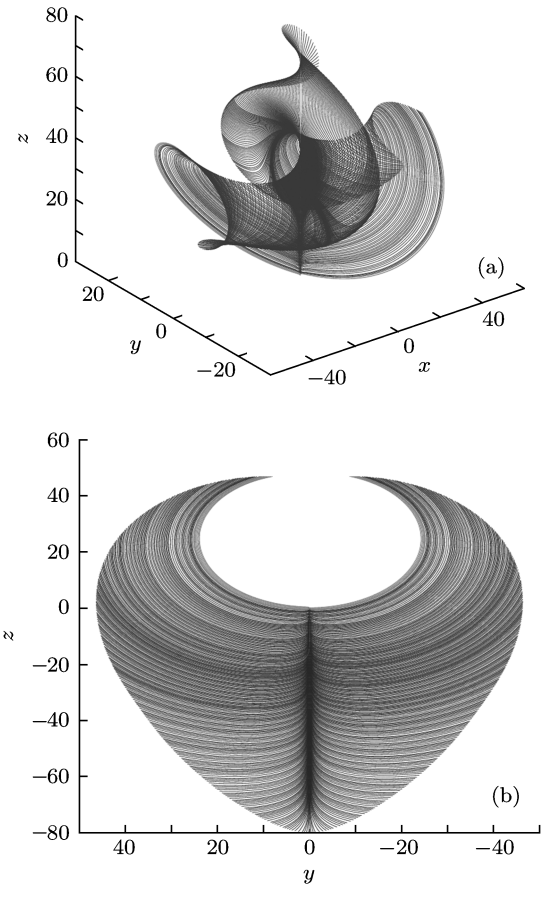

图 14 二维流形的轨道表示

5 结 论

本文研究映射的双曲不动点的二维流形计算问题. 由于映射的流形计算对已有流形存在 “依

赖性", 所以与向量场的二维流形计算相比, 更具有难度.

以文献 [18] 中提出的 Foliation 条件为基础, 通过对叶形重新进行定义, 提出了基于流形轨道的 Foliation 条件, 仿真结果表明, 推广的 Foliation 条件在二维流形的初始部分能够很好地控制流形在各个方向上的增长速度, 但当流形具有比较特殊的结构, 特别是流形局部发生 "垂直" 增长时, 单纯使用 Foliation 弧长进行控制的话效果会变差. 鉴于此, 我们在二维流形的初始平面部分使用 Foliation 弧长来控制, 当流形增长到非平面区后, 使用普通的欧氏弧长来控制, 仿真结果表明, 综合使用这两种控制手段得到的计算结果比仅使用 Foliation 弧长有了较大地改进, 整体上能够保证二维流形均匀地增长. 与原文献 [18] 中的 Foliation 条件相比, 本文的控制方法虽然不能保证二维流形在各向上的绝对均匀增长, 但整体上的控制效果还是比较理想的, 本文的控制办法实现起来更灵活、更简便.

文献 [18] 的算法默认新网格点的原像处于已有流形的边界上, 而实际上并非所有映射的二维流形都满足该条件, 所以该算法在应用中会有所局限. 而本文的方法是以一维流形轨道曲线的计算为基础的, 应用中不存在这个限制. 本文提出的插入轨道的算法, 巧妙地解决了二维流形面上网格点的插入、前像搜索, 以及网格点后续轨道计算问题, 该算法不要求插入的轨道必须从初始圆处开始计算, 从而不会在初始圆附近产生过多的网格点, 进而避免相邻轨道靠得太近, 这一点在计算较大弧长的二维流形时尤为重要. 不仅如此, 通过仿真发现, 本文算法还有助于减少所要计算的轨道的总数目. 并且由于本文的插入算法是以递归形式给出的, 所以即使两条轨道间要插入一条以上的轨道时, 本文算法也可自动实现. 由于一维流形计算方法是二维流形计算的基础, 所以当有快速的一维流形计算方法出现时, 按照本文的步骤, 可以立即被应用到二维流形的计算当中, 从而直接加快二维流形的计算速度.

[1] Xu, Liu C X, Yang T 2010 *Acta Phys. Sin.* **59** 131 (in Chinese) [许, 刘崇新, 杨韬 2010 物理学报 **59** 131]

[2] Johnson M E, Jolly M S, Kevrekidis I G 1997 *Numer. Algorithms.* **14** 125

[3] Guckenheimer J, Worfolk P 1993 Kluwer Academic Publishers 241

[4] Henderson M E 2005 *SIAM J. Appl. Dyn. Syst.* **4** 832

[5] Krauskopf B, Osinga H M 1999 *Chaos* **9** 768

[6] Krauskopf B, Osinga H M, Doedel E J, Henderson M E, Guckenheimer J, Vladimirsky A, Dellnitz M, Junge O 2005 *Bifur. Chaos. Appl. Sci. Engrg.* **15** 763

[7] Guckenheimer J, Vladimirsky A 2004 *SIAM J. Appl. Dyn. Syst.* **3** 232

[8] Li Q D, Yang X S 2010 *Acta Phys. Sin.* **59** 1416 (in Chinese) [李清都, 杨晓松 2010 物理学报 **59** 1416]

[9] Jia M, Fan Y Y, Li H M 2010 *Acta Phys. Sin.* **59** 7686 (in Chinese) [贾蒙, 樊养余, 李慧敏 2010 物理学报 **59** 7686]

[10] Li Q D, Yang X S 2010 *Acta Phys. Sin.* **59** 1416 (in Chinese) [李清都, 杨晓松 2010 物理学报 **59** 1416]

[11] Hobson D 1991 *J. Comput. Phys.* **104** 14

[12] You Z, Kostelich E J, Yorke J A 1991 *Int. J. Bifurc. Chaos Appl. Sci. Eng.* **1** 605

[13] Simó C 1989 Benest D, Froeschlé C (eds.) Les Méthodes Modernes de la Mécanique Céleste 285

[14] Parker T S, Chua L O 1989 *Practical Numerical Algorithms for Chaotic Systems* (Berlin: Springer)

[15] Krauskopf B, Osinga H M 1997 http://www.ima.umn.edu/preprints/OCT97/1517.ps.gz

[16] Krauskopf B, Osinga H M 1998 *J. Comput. Phys.* **146** 406

[17] England J P, Krauskopf B, Osinga H M 2004 *SIAM J. Appl. Dyn. Syst.* **3** 161

[18] Krauskopf B, Osinga H M 1998 *Int. J. Bifurc. Chaos* **8** 483

[19] Krauskopf B, Osinga H M 1999 *Chaos.* **9** 768

[20] Dellnitz M, Hohmann A 1997 *Numer. Math* **75** 293

[21] Li Q D, Zhou L, Zhou H W 2010 *Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition)* **22** 339 (in Chinese) [李清都, 周丽, 周红伟 2010 重庆邮电大学学报 (自然科学版) **22** 339]

[22] Palis J, Melo W D 1982 *Geometric Theory of Dynamical Systems* (New York: Springer-Verlag)

[23] Gonchenko S V, Ovsyannikov I I, Simo C, Turaev D 2005 *Internat. J. Bifur. Chaos Appl. Sci. Engrg.* **15** 3493

Growing two-dimensional manifold of nonlinear maps based on generalized Foliation condition*

Li Hui-Min† Fan Yang-Yu Sun Heng-Yi Zhang Jing Jia Meng

(*School of Electronics and Information, Northwestern Polytechnical University, Xi'an* 710072, *China*)

(Received 21 March 2011; revised manuscript received 10 April 2011)

Abstract

In this paper we present an algorithm of computing two-dimensional (2D) stable and unstable manifolds of hyperbolic fixed points of nonlinear maps. The 2D manifold is computed by covering it with orbits of one-dimensional (1D) sub-manifolds. A generalized Foliation condition is proposed to measure the growth of 1D sub-manifolds and eventually control the growth of the 2D manifold along the orbits of 1D sub-manifolds in different directions. At the same time, a procedure for inserting 1D sub-manifolds between adjacent sub-manifolds is presented. The recursive procedure resolves the insertion of new mesh point, the searching for the image (or pre-image), and the computation of the 1D sub-manifolds following the new mesh point tactfully, which does not require the 1D sub-manifolds to be computed from the initial circle and avoids the over assembling of mesh points. The performance of the algorithm is demonstrated with hyper chaotic three-dimensional (3D) Hénon map and Lorenz system.

Keywords: nonlinear map, stable and unstable manifold, 3D Hénon map, Lorenz system

PACS: 95.10.Fh, 52.25.Gj, 82.40.Bj

* Project supported by the National Natural Science Foundation of China (Grant No. 60872159).

† E-mail: huimin_lee@126.com