连续时间系统二维不稳定流形的异构算法*

李清都[1)2)†] 谭宇玲[2)] 杨芳艳[2)]

1)(重庆邮电大学网络化控制与智能仪器仪表教育部重点实验室,重庆 400065)

2)(重庆邮电大学非线性系统研究所,重庆 400065)

(2010 年 5 月 12 日收到;2010 年 7 月 1 日收到修改稿)

非线性系统的二维流形通常具有复杂几何结构和丰富动力学信息,因此在流形计算与可视化时存在大量的不可避免的数值计算. 因此,如何高效地完成这些计算就成了关键问题. 鉴于当今计算机的异构发展趋势(包含多核 CPU 和通用 GPU),本文在兼顾精度和通用性的基础上,提出了适用于新一代计算平台的快速流形计算方法. 本算法将计算任务分为轨道延伸和三角形生成两部分,前者运算量大而单一适合 GPU 完成,后者运算量小而复杂适合 CPU 执行. 通过对 Lorenz 系统原点稳定流形的计算,表明本算法能充分发挥异构平台的综合性能,可大幅度提高计算速度.

关键词: 不稳定流形, 流形计算, 异构计算, Lorenz 系统

PACS: 02.40.Sf, 05.45.-a

1. 引 言

对稳定和不稳定流形的几何性质的把握一直是非线性动力系统的核心问题之一. 用计算机从数值上对流形进行计算,进而直观展示流形的几何结构、以及系统在其上的演化规律,有着极为重要的学术研究价值. 通过它人们可以研究系统的同异宿轨、横截栅栏、分岔与混沌,以及吸引子的吸引区域等,实现系统结构和行为的整体分析. 对于二维流形,国内外不少学者对此做了长期深入的研究[1—7],主要成果包括 Doedel 的连续边值问题(BVP)算法[1], Krauskopf 和 Osinga 的测地圆算法[2], Guckenheimer 和 Vladimirsky 的偏微分方程(PDE)算法[3], Henderson 的扩展轨迹算法[4],作者的两步算法等[6]. 然而这项研究却进展缓慢,原因是动力系统的流形极其复杂,大小上有全局和有界,演进速度上快慢不一,延伸方向上有分有合,表面上崎岖复杂,所以难度很大. 近期,作者在分析和借鉴过去算法的基础上,提出了一种新算法[7]. 在新算法中,由于借助了成熟的微分方程求解算法,利用了普遍满足的轨道不相交性,因此在兼顾精度和通用性的同时,降低了轨道求解量. 然而,二维流形的复杂几何结构和丰富动力学信息,导致在计算与可视化时,存在着无法避免的大规模的计算量. 因此,如何高效地完成这些计算任务,成了提高流形计算速度的关键.

当今计算机构架正面临革新,并行计算和异构计算的趋势越发明显[8]. 中央处理器 CPU 和图形处理器 GPU 都有计算能力,前者缓存丰富,能高效执行串行指令,而后者执行单元众多,并行计算能力突出. 目前 GPU 峰值可达每秒 2.72 万亿次浮点运算,高于常用 CPU 两个数量级,而且发展速度是 CPU 的两倍,科学计算潜力很大. 异构计算可发挥两者的各自优势,实现整体计算能力的最大化利用,成为未来高性能计算的趋势. 例如,我国研制的"天河一号"曾首次达到世界排名第五的好成绩;美国橡树岭国家实验室正在研制的基于 Fermi 内核的异构超级计算机,预计可达当今最快超级计算机的 10 倍. 随着 GPU 用于通用计算的相关技术与协议的发布(如 CUDA, OpenCL 等),个人科学计算迎来了黄金机遇. 通过直接在现有平台上扩展通用 GPU,便能获得上数十到数百倍的性能提升,从而简单经济地搭建出个人计算中心. 此外,GPU 作为图形处理器在计算结果的展示还有着天生优势.

若能充分利用这种异构系统的计算资源,则有助

* 国家自然科学基金(批准号:10926072,10972082),重庆市教委项目(批准号:KJ080515),重庆市科委项目(批准号:CSTC-2008BB2409)资助的课题.

† E-mail: liqd@cqupt.edu.cn

于快速完成流形计算和可视化中的海量计算任务. 因此,本文借鉴文献[7]中的算法,在兼顾精度和通用性上优势的同时,采用所有轨道并行延伸的方式,研究异构系统上的流形计算方法,充分发挥 CPU 和 GPU 的整体优势,提高运算速度数十倍以上.

2. 二维不稳定流形的异构计算

设 n 维自治微分动力系统

$$\dot{x} = f(x), \tag{1}$$

其中 $x \in R^n$,映射 $f: R^n \to R^n$ 充分光滑,则鞍点 x_0 处的不稳定流形定义为

$$W^u(x_0) = \{x \in R^n \mid \lim_{x \to -\infty} \phi_t(x) = x_0\},$$

这里的 ϕ_t 代表方程(1)的流. 由于该系统自治且满足解的存在和唯一性,因此相空间中轨道具有不相交性.

由于 x_0 处的不稳定流形与雅克比矩阵 $Df(x_0)$ 的不稳定特征空间 $E^u(x_0)$ 相切,因此要获得二维不稳定流形 $W^u(x_0)$,我们可以在 $E^u(x_0)$ 上的 x_0 充分小邻域内,选取一个含 x_0 的单连通区域作为初始流形面 W_0;然后以其边界 ∂W_0 上各点为初始值,通过轨道求解延伸,轨道经过的区域即为所求流形. 但实际上,由于构成流形面的轨道有无穷多,构成轨道的点也有无穷多,所以无法对所有轨道和点进行求解. 合理的做法是:先在流形面上均匀地选取一些轨道;然后在轨道上均匀地选取一些样点,以时间顺序连接样点,用所得曲线来逼近实际轨道;接着,在相邻轨道之间,根据其样点的几何关系绘制三角形带;最后,拼接这些三角形带,得到由大量三角形面元组成的曲面,而该曲面即为实际流形面的有效逼近.

这里的数值流形面 W 是由许多样点和连接样点的三角形面元构成,为了节约存储空间,降低数据交互量,我们将这些样点和相关信息存放在长度为 N_P 的结构体数组 P 中. 为了便于论述,下文中我们直接用下标来简记 P 中的点,即第 k 个点 P_k 简记为点 k,它具有以下信息元素: x 为该点的坐标位置;v 为轨道前一点指向该点的单位向量;t 为从 B_0 出发到该点,沿轨道经历的时间;L 为从 B_0 出发到该点,沿轨道经历的长度;u 为该点所处轨道的下一点的下标;l 为该点相邻左点的下标;r 为该点相邻右点的下标.

对于点 k 上述任一元素 $\cdot$,下文中我们记为 $k.\cdot$,例如它的轨道下一点 u 记为 $k.u$. 在这里 u, l, r 表示点 k 的相邻点(无效时取 -1),其好处是:在计算时,它们不仅可用来确定计算边界、追踪附近的点,还能压缩记录待绘制三角形面元,节约存储空间,例如三点 $k, k.u, k.l$ 和三点 $k, k.u, k.r$ 可表示两个三角形;在计算完成后,我们不仅可利用各点的两个三角形(如果存在的话)来绘制流形面,而且这种网络连接关系为我们进一步分析流形面的几何结构、动力学行为打下基础.

为了便于论述,我们令 B 为已算流形面的计算边界,它是由处于边界上或边界以外的 n 条相邻的轨道尾段 $\Gamma_1, \Gamma_2, \cdots, \Gamma_n$ 连接而成. 令 $\underline{\Gamma}_i$ 和 $\overline{\Gamma}_i$ 分别表示轨道尾段 Γ_i 的起点和终点,令 $\overline{B}$ 表示由所有轨道终点构成的点列 $\overline{\Gamma}_1, \overline{\Gamma}_2, \cdots, \overline{\Gamma}_n$. 对于 B 上任意两条相邻轨道尾段 Γ_l 和 Γ_r,其边界 B 在标准状态下的连接方式只有 $\overline{\Gamma}_l \leftrightarrow \overline{\Gamma}_r$, $\overline{\Gamma}_l \leftrightarrow \underline{\Gamma}_r$ 和 $\underline{\Gamma}_l \leftrightarrow \overline{\Gamma}_r$ 三种. 如图 1 所示.

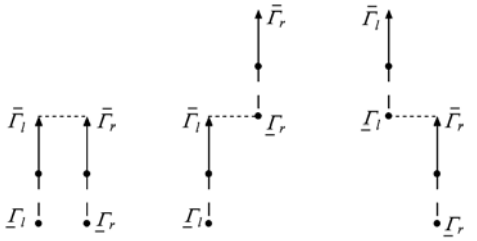

图 1 边界上相邻轨道尾段的连接方式

在实际计算时,先在初始流形面 W_0 的边界上适当选取一些点作为初始计算边界 B_0,然后要对边界上的所有轨道并行求解向外延伸 δ 长度,显然,只需处理序列 $\overline{B}_0$ 上各点即可. 在延伸后,对于以任意相邻两个新尾段 Γ_l 和 Γ_r,由于各增加了一个点作为新的终点,如图 2 所示,因此不能满足图 1 所示的连接关系. 假设点 k_l 和点 k_r 之间的连线处于 Γ_l 和 Γ_r 之间的计算边界 B_0 上,那么需要我们处理的是从 k_l 到 $\overline{\Gamma}_l$ 和从 k_r 到 $\overline{\Gamma}_r$ 的轨道间隙区域. 在处理

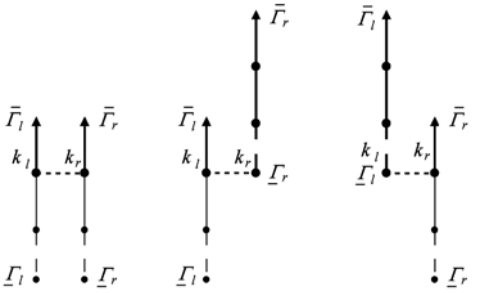

图 2 边界上相邻轨道尾段的连接方式,需要处理的是加粗的轨道间隙区域

时,首先,通过填充新的三角形面元,使其相邻连接关系恢复至图 1;接着,根据 Γ_l 和 Γ_r 在最后一个三角形上的距离,若距离过大则 k_l 和 k_r 间在插入新轨道,若距离过小,则选择其中一个轨道,并停止对其继续延伸. 所有轨道间隙处理完成后,我们得到新的点列 $\bar{B}_1$. 我们再对 $\bar{B}_1$ 延伸,然后再处理轨道间隙区域……如此循环下去,就能得到所需流形面.

在轨道延伸时,需要对 $f(x)$ 做大量运算,例如,采用 Runge-Kutta 四阶算法时每个时间步长需四次运算,每个新样点的产生则需要上百个时间步长. 在生成三角形面元时(调节计算边界到标准状态时),每个轨道样点最多对应两个三角形面元,仅需要少数向量内积运算确定邻近样点的几何关系即可. 因此在运算量上,轨道延伸远远大于三角形生成. 由于轨道数目多,程序上几乎没有分支,计算单一,非常适合 GPU 并行运算. 而计算边界 B 向标准状态调节时(确定三角形面元),尽管运算量小,但存在很多分支,处理复杂,适合 CPU 计算. 因此,采用这种思想能够充分发挥异构系统的整体性能,大幅度提高执行速度.

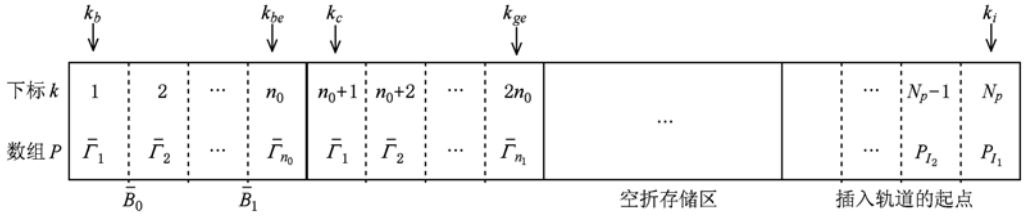

图 3 数组 P 中存放的内容,以及初始下标示意图

在数组 P 的数据存储上,每次我们只需要存放新样点即可. 因此 P 的数据结构如图 3 所示,从前往后依次存放 $\bar{B}_0,\bar{B}_1,\bar{B}_2,\cdots$,从后向前依次存放插入轨道的起点 $P_{I_1},P_{I_2},\cdots$,其中,n_0 表示初始轨道数目. 为了便于定位 P 中存放的点,我们定义变量如下:

k_b 和 k_{be} 分别表示当前 $\bar{B}$ 在 P 中起始和结束位置;k_{ge} 表示需要 GPU 求解延伸的点在 P 中的结束位置;k_c 表示当前 $\bar{B}$ 上 CPU 正在处理的点;k_i 表示当前可用于插入新轨道起点的位置.

为了便于理解,下面介绍基于 GPU 和 CPU 交替计算的算法,具体算法如下.

2.1. 算法开始

2.1.1. 初始化数值流形面和计算边界

在 ∂W_0 上均匀顺序地选取 n_0-1 个样点 $P_1,P_2,\cdots,P_{n_0-1}$. 为了得到完整流形面,令 $P_{n_0}=P_1$,顺序连接 $P_1,P_2,\cdots,P_{n_0}$ 得初始计算边界 B_0,其所围区域为初始流形面. 对于任意 $1\leqslant k\leqslant n_0$,令 P_k 的 $t=0$,$le=0$,$u=-1$,$l=k-1$,$r=k+1$. 最后,令 P_1 的 $l=-1$,P_{n_0} 的 $r=-1$.

B_0 选取时要确保系统的流向外或相切穿过它,根据 $E^u(x_0)$ 对应的特征值,可采取下述过程. 若这两个特征值为实数,令 U 和 V 为相应单位特征向量;若为一对共轭复数,令 U 和 V 分别为其中一个复单位特征向量的实部和虚部. 从理论上不难得出 P_k 的坐标选取公式:

$$\theta = 2\pi(k-1)/(n_0-1),$$

$$x = R_0\cos\theta\cdot U + R_0\sin\theta\cdot U + x_0,$$

在这里,R_0 决定了初始流形所处 x_0 邻域的大小. B_0 上每个轨道只有一点,所以 $P_k=\bar{\Gamma}_k$,即 $\bar{B}_0=B_0$. 接着,令 $k_b=1$,$k_{be}=n_0$,$k_{ge}=2n_0$,$k_i=N_P$. 最后,初始化即将被 GPU 延伸的点,对于 $k_{be}+1\leqslant k\leqslant k_{ge}$,标记 P_k 的轨道前一点是 $k-n_0$,即令 P_k 的 $r=k-n_0$.

2.1.2. 对计算边界 B 上所有轨道长度延伸 N 次,生成不稳定流形面

主循环开始:

1) 用 GPU 并行延伸 P 中 $k_{be}+1$ 与 k_{ge} 之间的点

对于其中任意一点 k,在 P 中读取其轨道前一点 $k.r$ 的 x,t,并将轨道延伸 δ 长度,更新点 k 的 x,t,L,v,令 k 和其前点的 u 分别为 $-1k,k.l=k.r$.

2)用 CPU 调节新计算边界(确定三角形面元),使其恢复到标准状态循环前初始化:$k_b=k_{be}+1$,$k_{be}=k_{ge}$.

k_c 从 k_b 取到 $k_{be}-1$ 做子循环:

A. 根据图 2 的三种情形,确定以 k_c,k_c+1 为相邻轨道终点 $\bar{\Gamma}_l,\bar{\Gamma}_r$ 的间隙区域,即定位 k_l 和 k_r.

B. 在间隙区域更新边界,记录三角形面元.

B1 循环直到 k_l 或 k_r 到达轨道终点:

通过这两点间的单位向量与它们下一点的信

息 v 的内积,比较轨道与边界线段 k_lk_r 的内夹角 $\angle(k_l.u)k_lk_r$ 和 $\angle(k_r.u)k_rk_l$,若前者小则记录三角形面元 $(k_l.u)k_lk_r$,即令 $k_l=k_l.u, k_l.r=k_r, k_r.l=k_l$;否则记录面元 $(k_r.u)k_rk_l$,即令 $k_r=k_r.u, k_l.r=k_r, k_r.l=k_l$.

B2 循环直到 k_l 到达轨道终点:

若 $\angle(k_l.u)k_lk_r$ 为锐角,则记录面元 $(k_l.u)k_lk_r$;否则跳出本循环.

B3 循环直到 k_r 到达轨道终点:

若 $\angle(k_r.u)k_rk_l$ 为锐角,则记录面元 $(k_r.u)k_rk_l$;否则跳出本循环.

C. 根据最后一个三角形估算的 Γ_l 和 Γ_r 距离 μ 的大小,插入或删除轨道如果 $\mu>\delta_I$,则在点 k_lk_r 中间插入新点 k_i.

记录 k_c 为轨道延伸点,即若 $k_c.u<0$ 则令 $k_{ge}=k_{ge}+1, k_{ge}.r=k_c$. 取 k_i 的 x,t,L 为 k_l 和 k_r 的平均值,并更新边界,即令 $k_i.u=-1, k_i.l=k_l, k_i.r=k_r, k_l.r=k_r.l=k_i$. 最后,记录 k_i 为轨道延伸点,并令 $k_i=k_i-1$.

否则,如果 $\mu<\delta_D$ 且 Γ_l 的左相邻轨道没有被合并,则将 Γ_l 和 Γ_r 合并为一条轨道若 k_l 不是轨道终点,则记录面元 $(k_l.u)k_lk_r$ 后令 $k_r.u=k_l$;若 k_r 不是轨道终点,则记录面元 $(k_r.u)k_rk_l$ 后令 $k_l.u=k_r$;若 k_l 和 k_r 都在 $\bar{B}$ 上,则比较向量 $k_l.v$ 和 $k_r.v$ 与线段 k_lk_r 的内夹角,若前者小,则令 $k_r.u=k_l$,否则令 $k_l.u=k_r$.

如果 k_c 所处轨道没被终止,即 $k_c.u<0$,则记录 k_c 为轨道延伸点.

子循环结束.

记录 k_{be} 为轨道延伸点.

主循环结束.

2.2. 算法结束

在本算法中,GPU 的作用为轨道延伸,CPU 的作用为生成三角形并调节轨道疏密. 如果当前计算边界的终点序列 $\bar{B}$ 较长,那么我们可以先让 GPU 延伸最开始的一部分,完成即可启动 CPU 的调节任务生成下个终点序列,于此同时,我们让 GPU 继续延伸其余部分. 这样以来,CPU 将不断产生新的待延伸的轨道终点供 GPU 处理,GPU 也不断将新的延伸结果返回给 CPU 调节,如此循环就能实现 CPU 和 GPU 的完全并行执行. 通过增加一些新的位置标记变量,如供 GPU 延伸的轨道终点序列的起始位置、可供 CPU 处理的序列结束位置等,就能很容易地将本算法修改为这种异构并行算法. 这样以来,在 CPU 和 GPU 的执行中,时间消耗较快的将被隐藏,速度由较慢的决定. 在时间消耗相同的情况下,计算速度将翻倍.

3. 算法的应用实例与速度比较

Lorenz 混沌系统具有深刻的物理背景,人们对其在理论和应用上做了广泛深入的研究,包括行为分析[9]、参数估计[10,11]、边界估计[12]、混沌轨道控制[13]、混沌镇定[14]、混沌同步[15,16]、吸引子改造[17]、超混沌化与同步[18—20]、分数阶 Lorenz 系统[21,22]、混沌电路[23,24]、混沌加密[25]等. 鉴于该系统的重要性、以及其状态空间结构的复杂性,在流形计算的文献中,常常通过计算它在原点的稳定流形来验证算法的有效性. 下面我们也同样对此进行计算,并做出比较. 由于稳定流形可以看作是不稳定流形的时间反向,下面我们研究时间反向的 Lorenz 系统,其方程为

$$\begin{aligned}\dot{x}_1 &= 10(x_1-x_2),\\ \dot{x}_2 &= -28x_1+x_2+x_1x_3,\\ \dot{x}_3 &= -x_1x_2+\frac{8}{3}x_3.\end{aligned}\tag{2}$$

原点 O 是它的一个鞍点,$\mathrm{D}f(O)$ 的三个特征值分别为 $\lambda_1^u=\frac{8}{3}, \lambda_2^u\approx 22.8, \lambda^s\approx -11.8$,其中不稳定特征值对应的单位特征向量为 $U=[0,0,1]^{\mathrm{T}}, V\approx -[-0.61481, 0.78867, 0]^{\mathrm{T}}$. 该点的不稳定流形(原系统的稳定流形)具有非常复杂的几何结构,它沿一个蝶形吸引子向内螺旋延伸,对原系统的混沌行为起主导作用. 下面我们在联想笔记本 Y450 上(CPU 为 T6600,GPU 为 GT240M),用本算法来计算和展示该流形. 为了使结果具有可比性,我们选取相同的基本参数 $n_0=13, R_0=4, \delta_I=1.5\delta, \delta_D=0.5\delta$,以及不同的轨道长度 $N\delta$ 和延伸步长 δ,利用 CPU 的单线程和 GPU 的多线程在不同计算环境上进行计算,结果如表 1 所示.

本文算法与文献[7]算法相比,均基于普遍满足的轨道不相交性和成熟的初值问题求解算法,通过在相邻轨道间填充三角形来生成流形面,因此具有相似的精度、通用性和计算总量. 主要区别是,本算法采用并行的方式将所有轨道同时向外延伸,而文献[7]的算法采用完全串行的方式每次只延伸一条轨道. 为此,我们采用相同的计算方

式,先比较它们在 MATLAB 上的单线程执行效率. 结果如表 1 的序号 1 和序号 2 所示,本算法是文献[7]算法效率的 3.4 倍. 单线程效率的提升来自于以下两方面:1)从局部三角形生成上来看,如图 4 所示,本文算法生成的三角形多为大小相对均匀的锐角三角形,更为有效,因此所用三角形个数降低了 12%;2)数组 P 的使用和三角形面元的精简记录,使数据交互量大幅度降低. 此外,在序号 2 中,轨道延伸和边界调节耗时之比约为 48. 由于前者可并行加速,根据阿姆达尔定律,还能再次提升四十多倍,从而超出文献[7]速度的一百多倍.

表 1 算法测试结果与对比

序号	算法	计算环境	$N\delta$	δ	三角形数	耗时/s
1	文献[7]	MATLAB,CPU	120	2	20307	115.081
2	本算法	MATLAB,CPU	120	2	17803	34.252
3	本算法	CUDA C,CPU + GPU	120	2	17979	0.033
4	本算法	CUDA C,CPU + GPU	300	1	902483	0.443
5	本算法	CUDA C,CPU + GPU	500	1	7332526	2.948

为了测试本算法的异构计算潜力,我们利用 CUDA C 编程在 CPU + GPU 上执行[8],其结果如表 1 中的序号 3 至序号 5 所示. 其中序号 3 采用与前两次相同的参数,然而时间仅消耗了 0.033 s. 对于更大、更细的流形面,序号 4 和 5 分别仅用了 0.443 和 2.948 s. 这时的流形面变得越来越复杂,如图 4 所示,某些位置趋于重合(看起来破碎的位置),说明原系统在这些区域对扰动极为敏感. 此外,这里的计算速度与国内外其它文献[5,7]相比,均超过了两个数量级.

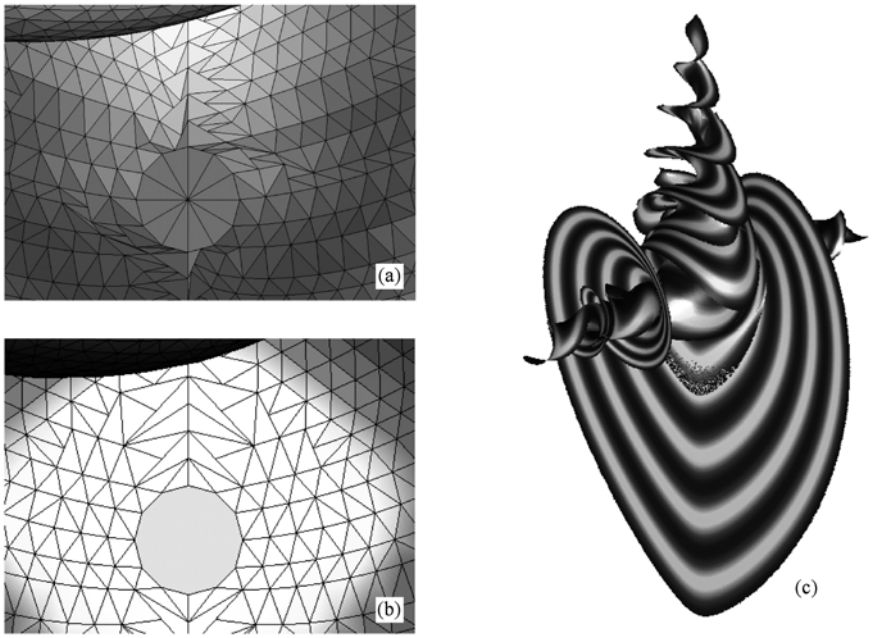

图 4 (a 序号 1 的三角形构成;(b)序号 2 的三角形构成;(c)序号 4 的计算结果

4. 结 论

本文提出了一种二维不稳定流形的快速算法,该算法在兼顾精度和通用性的同时,采用轨道并行延伸的方法,使计算得到并行化处理,有利于新一代异构计算机执行. 通过对 Lorenz 系统原点稳定流形的计算,表明本算法能充分发挥异构平台的综合性能,可以使二维流形的计算速度提高百倍以上,因此能够有效解决流形计算时一些无法避免的大

规模计算问题. 此外,本文提出的流形高效算法还能从以下三方面推动动力系统研究:1)能够对系统的流形进行实时计算和展示(像序号 3 的计算每秒可刷新 30 次),从而有利于研究系统相空间结构随参数的变化关系;2)可以通过增加轨道的求解长度,计算更大面积的流形,从而扩大我们的研究范围;3)可以通过提高计算精度,使曲面的面元更加精细,从而使我们的研究变得严格而又细致入微. 这些优点将有助于非线性系统的研究者们发现新现象、探索新规律.

[1] Doedel E J, Champneys A R 1997 ftp://ftp.cs.concordia.ca/pub/doedel/auto/

[2] Krauskopf B, Osinga H M 2003 *SIAM J. Appl. Dyn. Sys.* **2** 546

[3] Guckenheimer J, Vladimirsky A A 2004 *SIAM J Appli. Dyn. Sys.* **3** 232

[4] Henderson M 2005 SIAM *Journal on Applied Dynamical Systems* **4** 832

[5] Krauskopf B, Osinga H 2005 *Int. J. Bifurcation and Chaos* **15** 763

[6] Li Q D, Yang X S 2005 *Computational Physics* 22 549 (in Chinese) [李清都、杨晓松 2005 计算物理 **22** 549]

[7] 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]

[8] Kirk D, Hwu W 2010 *Programming Massively Parallel Processors* (Burlington: Elsevier)

[9] He W P, Feng G L, Gao X Q, Chou J F 2006 *Acta Phys. Sin.* **55** 3175 (in Chinese) [何文平、封国林、高新全、丑纪范 2006 物理学报 **55** 3175]

[10] Li L X, Peng H P, Yang Y X, Wang X D 2007 *Acta Phys. Sin.* **56** 51 (in Chinese) [李丽香、彭海朋、杨义先、王向东 2007 物理学报 **56** 51]

[11] Gao F, Li Z Q, Tong H Q 2008 *Chin. Phys.* B **17** 1196

[12] Zheng Y, Zhang X D 2010 *Chin. Phys.* B **19** 010505

[13] Yu J Z, Su N, Vincent T L 1998 *Acta Phys. Sin.* **47** 397 (in Chinese) [余建祖 1998 物理学报 **47** 397]

[14] Li S H, Tian Y P 2003 *Chin. Phys.* **12** 590

[15] Niu Y J, Xu W, Rong H W, Wang L, Feng J Q 2009 *Acta Phys. Sin.* **58** 2983 (in Chinese) [牛玉俊、徐 伟、戎海武、王 亮、冯进钤 2009 物理学报 **58** 2983]

[16] Li X J Xu Z Y Xie Q C Wang B 2010 *Acta Phys. Sin.* **59** 1532 (in Chinese) [李小娟、徐振源、谢青春、王 兵 2010 物理学报 **59** 1532]

[17] Giuseppe G 2008 *Chin. Phys.* B **17** 3247

[18] Chen G P, Hao J B 2009 *Acta Phys. Sin.* **58** 2914 (in Chinese) [陈光平、郝加波 2009 物理学报 **58** 2914]

[19] Wang X Y, Wang M J 2007 *Acta Phys. Sin.* **56** 5136 (in Chinese) [王兴元、王明军 2007 物理学报 **56** 5136]

[20] Han X J, Jiang B, Bi Q S 2009 *Acta Phys. Sin.* **58** 6006 (in Chinese) [韩修静、江 波、毕勤胜 2009 物理学报 **58** 6006]

[21] Zhang R X, Yang S P 2009 *Chin. Phys.* B **18** 3295

[22] Zhao L D, Hu J B, Liu X H 2010 *Acta Phys. Sin.* **59** 2305 (in Chinese) [赵灵冬、胡建兵、刘旭辉 2010 物理学报 **59** 2305]

[23] Cang S J, Chen Z Q, Wu W J 2009 *Chin. Phys.* B **18** 1792

[24] Wang G Y, Zheng Y, Liu J B 2007 *Acta Phys. Sin.* **56** 3113 (in Chinese) [王光义、郑 艳、刘敬彪 2007 物理学报 **56** 3113]

[25] Hao J H, Sun Z H, Xu H B 2007 *Acta Phys. Sin.* **56** 6857 (in Chinese) [郝建红、孙志华、许海波 2007 物理学报 **56** 6857]

A heterogeneous computing algorithm for two-dimensional unstable manifolds of time-continuous systems*

Li Qing-Du[1)2)†] Tan Yu-Ling[2)] Yang Fang-Yan[2)]

1) (*Key Laboratory of Network Control & Intelligent Instrument of Ministry of Education, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)

2) (*Institute for Nonlinear Systems, Chongqing Univ. Posts and Telecommunications, Chongqing* 400065, *China*)

(Received 12 May 2010; revised manuscript received 1 July 2010)

Abstract

Two-dimensional manifolds usually contain many nonlinear behaviors in complicate structures, which implies that much numerical calculation must be done during computing. Therefore, how to accomplish the work efficiently is a key problem. Since today's computers tend to heterogeneous platforms including multi-core CPUs and general purpose GPUs, this paper proposes a fast manifold computing algorithm, which is not only of high precision and versatility, but also very suited to the new generation of computers. The algorithm contains two kinds of computation: extending trajectories and generating triangles. The former is large and simple, which is suitable for GPU; the later is small and complicate, which is suitable for CPU. The computation for the stable manifold of the Lorenz system at the origin shows that this algorithm ensures the best performance of heterogeneous platforms and improve the computing speed greatly.

Keywords: unstable manifold, manifold computation, heterogeneous computing, Lorenz system

PACS: 02.40.Sf, 05.45.-a

* Project supported by the National Natural Science Foundation of China (Grant Nos. 10926072, 10972082), Chongqing Municipal Education Commission (Grant No. KJ080515) and Natural Science Foundation Project of CQ (Grant No. CSTC-2008BB2409).

† E-mail: liqd@cqupt.edu.cn