基于异构计算的简单行走模型的吸引区域研究*

李清都[1)3)†] 周红伟[1)] 杨晓松[2)3)]

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) (华中科技大学数学与统计学院, 武汉 430074)

3) (重庆邮电大学非线性电路与系统研究所, 重庆 400065)

(2010 年 10 月 22 日收到; 2011 年 6 月 15 日收到修改稿)

被动行走机器人由于结构简单、能量利用率高而倍受青睐, 但其很容易跌倒, 因此准确把握最终步态与吸引区域成了关键. 由于面对非光滑系统, 大规模数值计算很难避免, 为此本文先提出基于 CPU+GPU 异构平台的 Poincaré映射算法. 该算法可发挥最新平台计算潜力, 比传统 CPU 上算法快上百倍. 得益于此, 本文针对双足被动行走的最基本模型, 大规模地选取样点进行计算, 不仅清晰地得出吸引区域的形状轮廓和细节特征, 揭示了其内在分形结构, 还得到系统吸引集和吸引区域随倾角 k 的变化关系, 发现了新的稳定三周期步态和倍周期分岔混沌现象, 并研究了吸引区域.

关键词: Poincaré映射, 被动行走, 双足机器人, 混沌

PACS: 05.45.−a, 45.40.Ln

1 引 言

被动行走机器人仅靠自身结构和所受重力就能实现在缓坡上行走. 由于它们结构简单, 更符合生物运动规律 [1], 因此能量利用率很高, 仅用重力就能补充足与地面碰撞和摩擦的能量损失. 若以简单驱动代替重力提供能量, 就能实现平地上的稳定行走. 因此被动行走能够突破传统机器人能耗过大的发展瓶颈, 为双足机器人的发展提供了一个新思路.

被动行走最早由 McGeer 于 1990 年提出 [2], 经过 Cornell, Delft, MIT 等的研究小组所做的大量动力学研究与样机实验, 逐步发展为机器人研究的前沿核心领域, 其关键是如何实现稳定行走. 实验表明, 这类模型能以稳定步态行走的概率很低, 小的扰动就会使其跌倒. 因此弄清其稳定步态, 以及吸引区域的大小和结构变得十分必要. 这不仅直接决定了受扰动后能否恢复到原有步态, 而且有助于设计出更合理的硬件结构与控制方式, 使其行走变得更稳定. 为此, 国内外学者做了大量研究. 对于直腿被动行走, Garcia 和 Goswami 等研究了简化情形时的周期轨与稳定性, 发现随着坡度逐渐增加, 会出现倍周期分岔而进入混沌 [3,4], Schwab 和 Wisse 则进一步计算了其吸引区域, 发现其很小 [5]; 近期, 国内柳宁等人研究了更一般情形时的周期步态和吸引区域, 发现了若干混沌步态 [6,7].

在方法上, 为处理双脚触地时的切换, 人们通常以切换流形为 Poincaré截面将模型离散化, 依靠 Poincaré映射计算进行研究. 为求解周期步态, 常采用 Newton- Raphson 迭代法 [7], 然而由于其收敛范围极小, 不得不进行大规模的尝试. 在吸引区域分析时, 多采用胞映射法 [6,7], 由于胞数随着分辨率和维数急剧增加, 为了得到更加清晰的吸引区域, Poincaré映射的大规模数值计算也无法避免. 因此如何快速完成这些计算成了关键. 不仅有助于更全面地搜索稳定步态, 更清晰地得出其吸引区域的形状与结构, 而且还能用在胞映射的其他非线性问题的研究上 [8]. 当今计算机构架正面临革新, 中央处理器 CPU 和图形处理器 GPU 都有计算能力, 前者能高效执行串行指令, 而后者并行计算能力突出. 目前 GPU 的计算能力高于 CPU 数十倍, 且增长速度快一倍, 计算潜力很大. 异构计算可发挥两者的

* 国家自然科学基金 (批准号: 61104150, 10972082), 重庆市科委项目 (批准号: cstcjjA40044) 和华中科技大学自主创新基金 (批准号: 011906), 资助的课题.

† E-mail: ql78@cornell.edu

© 2012 中国物理学会 Chinese Physical Society

各自优势, 实现整体计算能力的最大化利用, 成为未来发展趋势 [9]. 随着 GPU 用于异构计算的协议制定 [10], 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 GPU, 便能获得数十甚至数百倍的性能提升, 从而简单经济地搭建出个人计算中心, 从而快速完成海量计算任务 [11].

因此, 本文首先提出基于异构平台的 Poincaré计算方法, 解决大规模点的 Poincaré映射的快速计算问题, 然后针对双足被动行走的简化模型, 计算出稳定步态吸引区域的形状和结构, 进而更细致全面地研究其稳定步态行为.

2 大规模点 Poincaré映射的异构计算

对于 n 维自治系统

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

取 $S:\{\boldsymbol{x}|s(\boldsymbol{x})=0,\dot{s}(\boldsymbol{x})>0\}$ 为 Poincaré截面, 则相应 Poincaré映射 $\boldsymbol{g}:S\to S$ 定义为: 对于任意初值点 $\boldsymbol{x}\in S$, $\boldsymbol{g}(\boldsymbol{x})$ 表示从 x 出发在系统 (1) 的作用下再次回到 S 时的交点.

在研究 $\boldsymbol{g}$ 的不变集与性质时, 常常依靠大量点的映像计算, 其计算量往往随着 $\boldsymbol{S}$ 的维数指数增加. 例如, 在胞映射法时, 每个维度上选取 10^3 个胞, 则二维需计算 10^6 个点, 三维变为 10^9 个. 目前常微分方程的数值求解研究已经比较成熟, 在确保误差相同的情况下, 计算量很难再进一步降低. 在处理如此大规模数值计算时, 采用 CPU 计算的传统方法需要消耗许多宝贵的计算时间和计算成本, 甚至有时无法完成计算任务. 为了实现大量点的 Poincaré映射的高效计算, 下面采用 CPU 和 GPU 相互协调的异构计算方式进行加速.

根据新制定的 OpenCL 标准 [10], 我们在 CPU 上执行宿主程序, 用于配置和调度; 在 GPU 上执行内核程序, 为每点创建一个独立线程 (work-item) 进行并行计算. GPU 中有许多计算单元, 每个单元又含有许多线程处理器, 每个处理器同时只处理一个线程. 由于线程很多且相互独立, 根据目前 GPU 的线程粒度, 我们选取 64 个线程为一组 (work-group), 以便 OpenCL 运行时为其灵活分配计算单元.

在实际计算时, 一方面, 每个线程需要获取各自初值点的坐标向量. 对此, 我们先把它们按行依次存放于二维数组 $\boldsymbol{X}$ 中, 然后将其一次性复制到 GPU 显存 (global memory) 中, 并用 $\boldsymbol{X}_{\rm d}$ 表示该副本, 接着各线程按其编号从 $\boldsymbol{X}_{\rm d}$ 中读出坐标各自的初值点 $\boldsymbol{x}_{\rm d}$ 进行并行计算. 计算完成后, GPU 将结果回写至 $\boldsymbol{X}_{\rm d}$, CPU 再把其复制到 $\boldsymbol{X}$ 并输出. 另一方面, 这些线程公用一些常数参数, 比如误差界、最长求解时间、最小求解步长等参数, 还有系统方程参数、求解中所用公式系数等常量. 我们依次将其存放于一维常量数组 $\boldsymbol{P}$ 中, 然后把它复制到 GPU 常数内存 (constant memory) 中, 并用 $\boldsymbol{P}_{\rm d}$ 表示该副本. 在计算时供各线程反复从 $\boldsymbol{P}_{\rm d}$ 读取所需常数. 常数内存因采用广播机理而速度极快, 不会造成线程的等待.

本算法的具体流程如图 1 所示. 宿主程序主要包含 OpenCL 的初始化、内核对象的创建、数据参数的传输、内核的执行以及结果输出等几个主要步骤. 内核程序主要包括常微分方程数值求解、以及求取截面交点的 Newton-Raphson 迭代两个步骤. 为了控制计算误差, 在微分方程数值求解时, 本算法采用变步长的自适应 Runge-Kutta 法. 目前可供选择的数值公式很多, 其中最常用的是四阶五阶变步长法, 如 Runge–Kutta–Fehlberg 法、Dormand–Prince 法、Cash–Karp 法等. 如果对误差要求不严格, 也可对本文算法稍作修改, 采用固定步长法, 如经典 Runge-Kutta 四阶公式, 也可以采用线性多步法等.

新一代 GPU 均支持 FMA(Fused Multiply-Add) 运算, 即直到乘法和加法运算完成后才舍入到双精度, 且每个线程处理器每个时钟周期可以执行一次这种运算. 因此, 在算法实现时, 做好 GPU 双精度 FMA 运算优化, 不仅有利于提高计算精度, 而且还能逼近 GPU 计算能力的理论峰值. 本算法的主要计算任务集中在 $\boldsymbol{f}(\boldsymbol{x})$ 的求值和变步长公式套用上, 其中后者为典型的乘加运算, 因此需要对前者进行重点优化.

为了使本算法易于使用, 我们可在宿主程序做好与 MATLAB 接口, 用数组 $\boldsymbol{X}$ 和 $\boldsymbol{P}$ 作为其输入参数, 并编译为 MEX 文件. 对于不同问题, 由于函数 $\boldsymbol{f}$ 和 s 产生变化, 因此需要对内核的相应部分调整. 采用 OpenCL 编写生成的 MEX 文件, 其自身已具备内核源程序的自动编译能力. 因此, 我们只需要用文本编辑器修改 $\boldsymbol{f}(\boldsymbol{x})$, $s(\boldsymbol{x})$ 的宏定义即可实现对新问题的计算. 由于能够脱离开发编译环境, 因此使用方便.

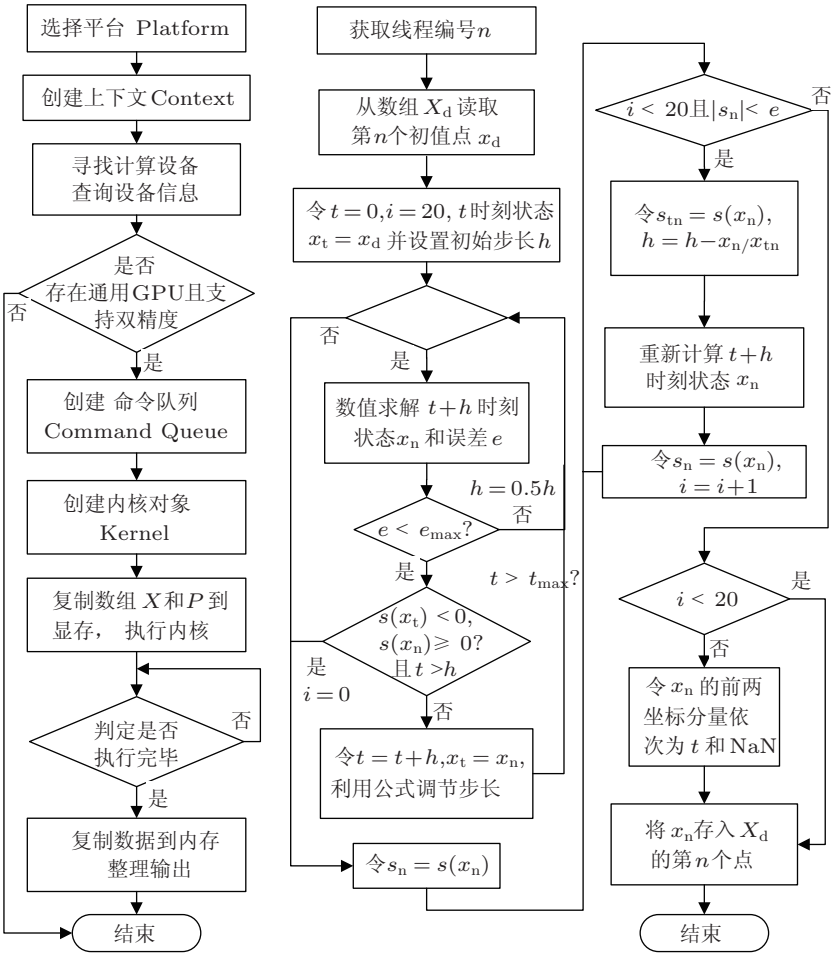

图 1 算法流程, 左图为宿主程序, 右图为内核程序

3 简单被动行走模型的稳定步态与吸引区域

最简单的被动行走模型如图 2 所示, 它由两条直腿构成, 其中一条着地, 为支撑腿 (stance leg), 另一条为摆动腿 (swing leg). 忽略腿部质量, 仅考虑两个脚质点 m 和一个髋质点 M, 那么该模型变成一个双摆模型. 从图 2 的单步行走过程可以看出, 当摆动腿的角度 φ 再次等于 2θ 时, 支撑腿与摆动腿需要切换, 开始下一步行走. 本模型的动力学行为可由以下切换系统来描述 [3]:

$$\ddot{\theta}=\sin(\theta-k),$$
$$\ddot{\varphi}=\sin(\varphi)[\dot{\theta}^2-\cos(\theta-k)]+\sin(\theta-k), \quad (2)$$

其中 k 表示斜坡的倾角. 这里的切换流形是

$$s(\theta,\varphi)=\varphi-2\theta=0, \quad (3)$$

切换函数是

$$\theta^+=-\theta^-,$$
$$\varphi^+=-2\theta^-,$$
$$\dot{\theta}^+=\cos(2\theta^-)\dot{\theta}^-,$$
$$\dot{\varphi}^+=\cos(2\theta^-)(1-\cos(2\theta^-))\dot{\theta}^-. \quad (4)$$

也就是说, 轨线一旦到达切换流形, 系统状态马上通过切换函数跳转到新的初始状态继续运行. 为了处理系统的不连续性, 我们直接选取切换流形作为系统 Poincaré截面. 由于本混合系统主要由时间连续系统 (2) 和切换函数 (4) 构成, 因此我们研究相应 Poincaré映射 g 和切换函数的复合映射 $\boldsymbol{H}$.

为利用上述算法进行计算, 我们令 $\boldsymbol{x}=(x_1,x_2,x_3,x_4)^{\rm T}=(\theta,\varphi,\dot{\theta},\dot{\varphi})^{\rm T}$, 于是得

$$\boldsymbol{f}(\boldsymbol{x})=\begin{pmatrix} x_3 \\ x_4 \\ \sin(x_1-k) \\ \sin(x_2)[x_3^2-\cos(x_1-k)]+\sin(x_1-k) \end{pmatrix}, \quad (5)$$

$$\boldsymbol{s}(\boldsymbol{x})=x_2-2x_1 \text{ 和 } \quad \dot{\boldsymbol{s}}(\boldsymbol{x})=x_4-2x_3, \quad (6)$$

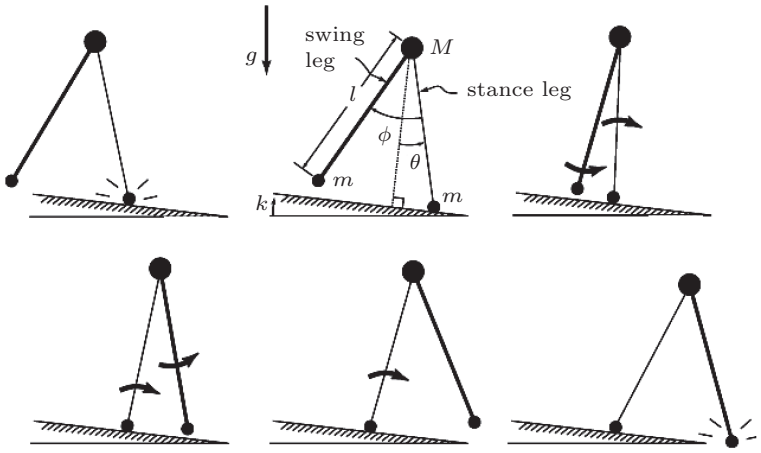

图 2 简单被动行走模型与单步行走过程

而切换函数变为

$$\boldsymbol{x}^{+} = \boldsymbol{r}(\boldsymbol{x}^{-}),$$

$$\boldsymbol{r}(\boldsymbol{x}) = \begin{pmatrix} -x_1 \\ -2x_1 \\ \cos(2x_1)x_3 \\ \cos(2x_1)(1-\cos(2x_1))x_3 \end{pmatrix}. \quad (7)$$

因此, 我们有

$$\boldsymbol{H} = \boldsymbol{r} \circ \boldsymbol{g}, \quad (8)$$

显然, 计算 $\boldsymbol{H}$ 的关键是计算 Poincaré映射 $\boldsymbol{g}$. 因为 $\boldsymbol{r}$ 的取值仅由 x_1 和 x_3 分量决定, $\boldsymbol{x}$ 和 $r(\boldsymbol{x})$ 均位于 S 上 (即其第二坐标分量 x_2 仅由 x_1 决定), 并且

$$x_4^{+} = \cos(2x_1^{-})x_3^{-}(1-\cos(2x_1^{-})) = x_3^{+}(1-\cos(2x_1^{+})),$$

因此, 在选择初值 $\boldsymbol{x}$ 时, 只要其 x_2 和 x_4 分量满足

$$x_2 = 2x_1,$$
$$x_4 = x_3(1-\cos(2x_1)), \quad (9)$$

那么复合映射 $\boldsymbol{H}$ 就可被看作是 x_1ox_3 平面的映射.

Garcia 研究发现, 该系统在坡度较小时有稳定的周期步态 $k < 0.0151$, 随着坡度 k 的逐渐增大, 出现倍周期分岔, 进入混沌步态, 如图 3 所示 [3]. Schwab 和 Wisse 的研究表明, 这些步态吸引区域面积很小 [5].

下面我们通过大规模样点的 $\boldsymbol{H}$ 映像计算, 更细致更全面地去研究该吸引区域的形状和内部结构. 鉴于映射 $\boldsymbol{H}$ 由 Poincaré映射 $\boldsymbol{g}$ 和切换映射 $\boldsymbol{r}$ 构成. 对于前者, 由 (2) 式不难推导出, x_1 和 x_3(即 θ 和 $\dot{\theta}$) 恒满足

$$E = 0.5x_3^2 + \cos(x_1 - k), \quad (10)$$

因此, 在 (2) 式的数值求解过程中 E 保持不变, 且仅由初值点的 x_1 和 x_3 分量决定. 利用恒等式 (11), 我们可以确保 $\boldsymbol{g}$ 数值求解的准确性. 在具体计算时, 我们对 $\boldsymbol{x}_{\rm d}$ 只取初值点的 x_1 和 x_3 分量 (缩小了一半数据传输量), 在内核开头利用 (9) 式将 $\boldsymbol{x}_{\rm d}$ 扩展成四个分量的 $\boldsymbol{x}_t$; 在内核结尾, 我们再将 $r(\boldsymbol{x}_n)$ 的 x_1 和 x_3 分量赋值给 $\boldsymbol{x}_{\rm d}$.

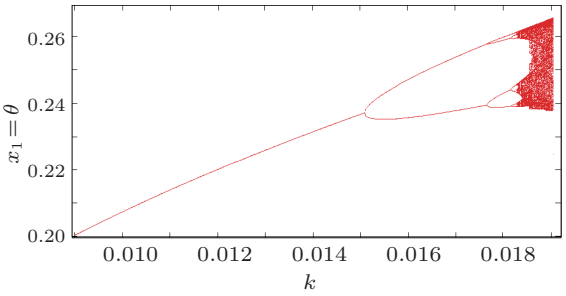

图 3 被动行走稳定步态的倍周期分岔 $k = [0.009, 0.0192]$

为比较和验证上述算法的速度和精度, 我们选取 $k = 0.009$. 这时 $\boldsymbol{H}$ 存在一个稳定平衡点 $\boldsymbol{x}_{\rm o} = (x_1, x_3) = (0.200310900687949, -0.19983247317447)$, 我们在其邻域内随机选取大量的初始点进行计算. 在 MATLAB 环境中, 我们先用 ode45 计算, 然后用 C 语言 MEX 程序来加速, 最后用本文方法来计算. 为了便于比较, 上述计算中均利用 Dormand–Prince 求解公式, 分别采取三种单步误差. 结果如表 1 所示, 其中 CPU 为 AMD Phenom II X4 965@3.4GHz, GPU 为 HD5830. 显然, 常规 ode45 求解速度较慢, 用 C 程序能获可观提速, 而本算法又比 C 程序提速上百倍, 结果本算法相对 ode45 提高了数万倍. 在精度上, 本算法与 ode45 和 C 程序的最终结果差距均在单步误差限内. 因此, 本算法能够在保证精度的情况下大幅度提升计算速度. 这

使对大规模点的 Poincaré映射计算成为可能, 从而产生多方面好处: 1) 我们可以扩大分析区域去把握全局行为; 2) 我们可以通过稠密取样来提高分辨率, 使研究细致入微; 3) 我们还能实时分析系统随参数的变化规律, 有利于分析系统的结构和分岔.

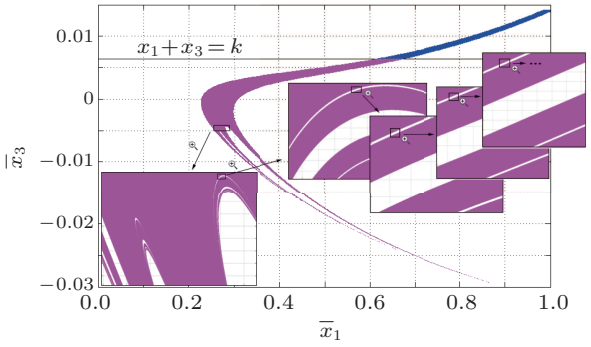

图 4 $k=0.009$ 时的稳定周期解吸引区域

Schwab 和 Wisse 通过胞映射法, 用 200×250 个样点估算了 $\boldsymbol{H}$ 的吸引区域, 发现其形状为一细长区域. 该区域被直线 $x_1+x_3=k$ 分为线上和线下两部分, 其中线上部分由于输入能量不足以克服切换损失而向后倒, 因此线下部分是我们的研究重点. 由于该部分形状细长且靠近直线, 为了便于观察, 我们引入变换

$$(\bar{x}_1,\bar{x}_3)=Q(x_1,x_3),$$

其中

$$Q=\frac{1}{\sqrt{2}}\begin{pmatrix}1&-1\\1&1\end{pmatrix},$$

将 x_1ox_3 面顺时针旋转 $\pi/4$. 对于 $\bar{x}_1o\bar{x}_3$ 平面, 我们选择一个稠密网格, 其中在 $\bar{x}_1$ 方向从 0 到 1 选取 1920 个格点, 在 $\bar{x}_3$ 方向从 -0.035 到 0.015 选取 1080 个格点, 总计 1920×1080 个样点. 尽管胞映射法计算可以避免大量的重复计算, 但它在一定程度上牺牲了准确性. 由于本文算法极快, 所以我们用直接迭代法来计算吸引区域高清图像, 以便准确把握吸引区域形状和结构. 对于吸引区域的周期轨道, $\partial\boldsymbol{H}$ 在 $\boldsymbol{x}_{\rm o}$ 处的特征值之模小于 1, 因此存在 $\boldsymbol{x}_{\rm o}$ 的某个 $e_{\rm o}$ 邻域, 该邻域内任意点均收敛于 $\boldsymbol{x}_{\rm o}$. 因此, 若格点落入该邻域, 我们则认为该格子位于 $\boldsymbol{x}_{\rm o}$ 的吸引区域.

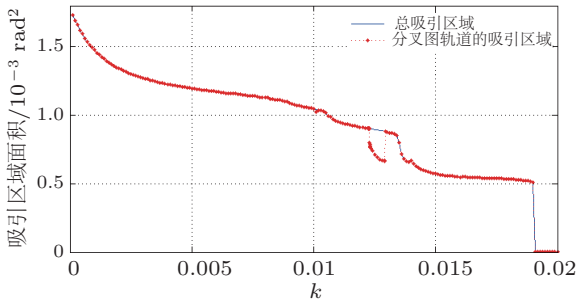

图 5 两种方法得到的不同的吸引区域面积

下面, 我们首先计算 $k=0.009$ 时单周期点 $\boldsymbol{x}_{\rm o}$ 的吸引区域. 选取 $e_{\rm o}=5\times10^{-5}$, 单步绝对误差 10^{-9} 进行计算, 结果如图 4 所示. 有效吸引区域 (直线下面部分) 总面积为 0.001091 rad^2, 其边界看起来非常复杂. 为了弄清其中结构, 我们对某些局部, 做了放大重新计算, 发现蕴含着自相似的嵌套结构, 如图 4 所示, 这说明该吸引区域有着分形结构.

表 1 本算法与 ode45 和 C 语言 MEX 程序的计算结果对比

点数	单步绝对误差	ode45 耗时/s	C 程序耗时/s	本算法耗时/s	与 C 程序结最大差距	与 C 程序速度比 (倍)	与 ode45 结果最大差距	与 ode45 速度比 (倍)
1024×1024	10^{-6}	4393	13.87	0.190	6.455×10^{-7}	73.2	6.396×10^{-7}	23175
	10^{-9}	10093	45.16	0.366	6.787×10^{-10}	123.5	6.569×10^{-10}	27598
	10^{-12}	32965	170.64	0.945	6.621×10^{-13}	180.5	5.783×10^{-13}	34875
2048×4096	10^{-6}	35256	110.55	1.086	6.402×10^{-7}	101.8	6.261×10^{-7}	32461
	10^{-9}	80257	364.29	2.204	6.557×10^{-10}	165.3	6.679×10^{-10}	36422
	10^{-12}	258832	1346.16	6.772	6.309×10^{-13}	198.8	6.193×10^{-13}	38221

鉴于吸引区域大小对本模型的重要物理意义, 下面我们以 0.0001 为步长, 研究其面积与参数 k 的变化关系. 我们采用两种方法进行计算. 首先, 我们直接反复迭代所有格点, 剔除不收敛的格点, 最终我们将得到总极限集和总吸引区域. 结果如图 5 所示, 随着坡度 k 的逐步增大, 总吸引区域几乎单调递减至 0, 这说明在较小的坡度上容易进入稳定步态. 然后, 我们用计算图 4 的方法来计算分岔图 3 中所示步态的吸引区域. 结果在 $k\in[0.0123,0.0129]$ 时, 其吸引区域骤然下降, 明显小于总吸引区域. 因此, 这里除了 Garcia 在文献 [3] 中发现的图 3 中的稳定步态之外, 应该还存在新的吸引子 (稳定步态),

并且其吸引区域就是两个吸引区域之差.

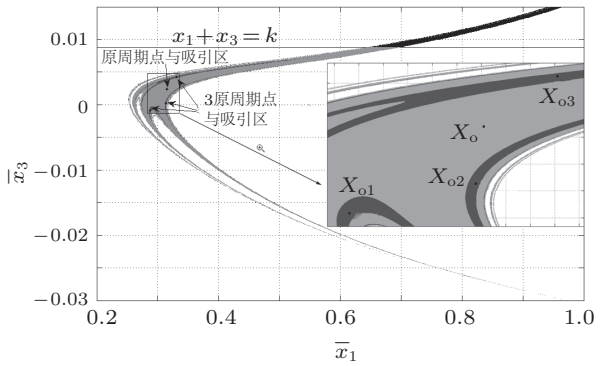

图 6 $k=0.0125$ 时的吸引子与吸引区域

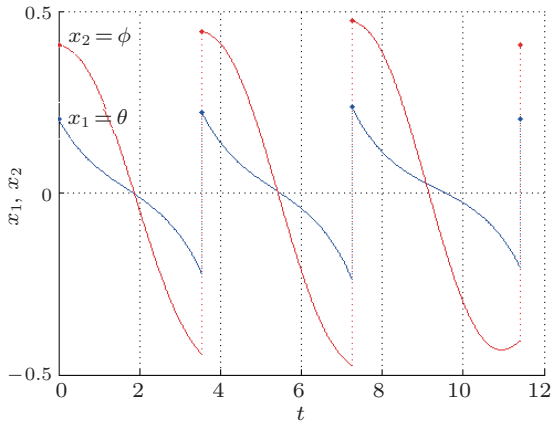

图 7 $k=0.0125$ 时的吸引三周期步态

下面我们对这些新的吸引子和吸引区域进行研究. 首先我们选取 $k=0.0125$, 这时原吸引子为单周期点 $\boldsymbol{x}_{\rm o}\approx(0.22477993,-0.22139495)$, 其吸引区域如图 6 所示. 我们在迭代极限集中排除该点后, 仅剩以下三点:

$$\boldsymbol{x}_{\rm o1}\approx(0.20312090,-0.20429593),$$

$$\boldsymbol{x}_{\rm o2}\approx(0.22211583,-0.22173218),$$

$$\boldsymbol{x}_{\rm o3}\approx(0.23671673,-0.23067387).$$

数值仿真表明 $\boldsymbol{x}_{\rm o2}=\boldsymbol{H}(\boldsymbol{x}_{\rm o1})$, $\boldsymbol{x}_{\rm o3}=\boldsymbol{H}(\boldsymbol{x}_{\rm o2})$ 且 $\boldsymbol{x}_{\rm o1}=\boldsymbol{H}(\boldsymbol{x}_{\rm o3})$, 说明这是一个三周期轨, 仿真步态如图 7 所示. 它们的吸引区域如图 6 所示, 其相互交织在一起, 边界模糊.

为了进一步弄清这个三周期步态的起源于走向, 我们需要计算其随参数 k 的关系. 首先, 我们以 0.000001 为步长, 将 k 从 0.0125 向下递减, 发现在 k 减至 0.01228 时该三周期轨突然消失, 系统回到只有单个吸引周期点的状态. 然后, 我们再以同样的步长, 将 k 从 0.0125 向上递增, 这时出现了倍周期分岔, 如图 8 所示, 并最终进入混沌状态, 其步态仿真如图 9 所示. 然而, 在 k 增至 0.01293 时混沌状态突然消失, 系统回到只有单个吸引周期点的情形. 这说明对于这种新步态, 参数 k 的区间大约为 $[0.01229,0.01292]$. 通过计算吸引区域, 且比较后, 我们发现了面积基本上随 k 的增加而增大, 其占总吸引区域比例约 12%—25%之间. 因此参数 k 在此范围时, 有较大的概率进入这种新步态, 所以不能忽略.

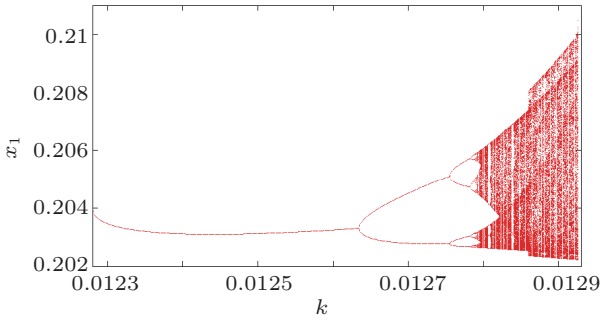

图 8 新三周期步态的分岔演化

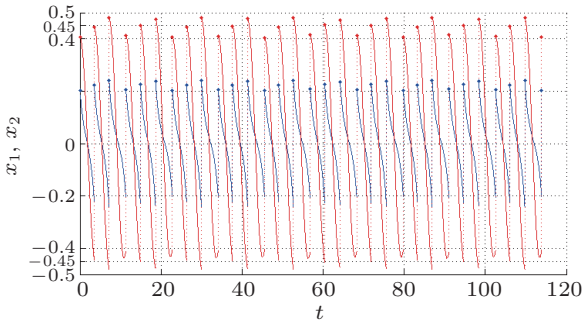

图 9 $k=0.0129$ 时的新混沌步态

4 结 论

本文提出了适合 CPU+GPU 异构平台的 Poincaré映射计算方法. 它能提高计算速度百倍以上, 非常适合大规模点的快速计算, 可以用在动力系统全局行为的分析上. 利用本算法的快速大规模计算能力, 我们重点研究了双足被动行走模型的吸引区域. 首先, 通过计算 $k=0.009$ 时, 吸引区域的高清图像, 我们发现了边界的分形结构. 然后, 我们调节参数 k, 采用两种计算方法, 一方面研究了整个系统的吸引集的吸引区域面积, 发现其随坡度 k 的增大而减小; 另一方面研究了单周期吸引轨道, 在分岔时, 其吸引区域面积的变化曲线. 两者的不一致导致我们发现在参数 $k\in[0.01229,0.01292]$ 时出现的新步态. 该步态

随着 k 的增加, 由三周期步态开始发生倍周期分岔, 进而出现混沌步态. 这些新步态不能忽略, 其吸引区域可达总吸引区域的 1/4. 本文研究有助于认识和把握被动行走模型的物理机理, 避免机器人跌倒. 另外, 由于 CPU+GPU 异构平台是计算机的未来发展方向, 因此本文算法应用前景广阔, 可望解决一些 Poincaré大规模计算问题, 使数值研究变得更加细致全面, 促使研究者发现新现象、探索新规律, 本文中新步态的发现正是得益于此.

[1] Winter D 1991 *Biomechanics and Motor Control of Human Movement* (2nd ed) (New York: Wiley)

[2] McGeer T 1990 *I. J. Robotic Res.* **9** 62

[3] Garcia M, Chatterjee A, Ruina A, Coleman M 1998 *J. Biomech. Eng.* **120** 281

[4] Goswami A, Thuilot B, Espiau B 1998 *I. J. Robotic Res.* **17** 1282

[5] Schwab A, Wisse M 2001 Basin of attraction of the simplest walking model. In: *International Conference on Noise and Vibration*, (Pennsylvania: ASME)

[6] Liu N, Li J F, Wang T S 2008 *Mechanics in Engineering* **30** 18 (in Chinese)[柳宁, 李俊峰, 王天舒 2009 力学与实践 **30** 18]

[7] Liu N, Li J F, Wang T S 2009 *Acta Phys. Sin.* **58** 3772 (in Chinese)[柳宁, 李俊峰, 王天舒 2009 物理学报 **58** 3772]

[8] Hong L 2010 *Chin. Phys.* B **19** 030513

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

[10] Munshi A 2010 The OpenCL Specification Version 1.1. (Khronos OpenCL Working Group)

[11] Li Q, Tan Y, Yang F 2011 *Acta Phys. Sin.* **60** 030206 (in Chinese)[李清都, 谭宇玲, 杨芳艳 2011 物理学报**60** 030206]

A study of basin of attraction of the simplest walking model based on heterogeneous computation*

Li Qing-Du[1)3)†] Zhou Hong-Wei[1)] Yang Xiao-Song[2)3)]

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

2) (*School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan* 430074, *China*)

3) (*Institute for Nonlinear Circuits and Systems, Chongqing University Posts and Telecommunications, Chongqing* 400065, *China*)

(Received 22 October 2010; revised manuscript received 15 June 2011)

Abstract

Passive dynamic walking becomes an important development for walking robots due to its simple structure and high energy efficiency, but it often falls. The key to this problem is to ascertain its stable gaits and basins of attraction. In order to handle the discontinuity, massive numerical computation is unavoidable. In this paper, we first propose an algorithm to compute Poincaré maps in heterogeneous platforms with CPU and GPU, which can take the best performance of the newest heterogeneous platforms and improve the computing speed by more than a hundred times. With this algorithm, we study the simplest walking model by sampling massive points from the state space. We obtain high resolution images of the basin of attraction, and reveal its fractal structure. By computing the relation between the stable gaits and their basins and by varying the slop k, we find a new three-period stable gait and a period-doubling route to chaos, and we also study the new gait and its basin.

Keywords: Poincaré map, passive dynamic walking, bipeds, chaos

PACS: 05.45.−a, 45.40.Ln

* Project supported by the National Natural Science Foundation of China (Grant Nos. 61104150, 10972082), the Natural Science Foundation Project of CQ CSTC (Grant No. cstcjjA40044), and the Independent Innovation Foundation of HUST(Grant No. 011906).

† E-mail: ql78@cornell.edu