

基于 GPU 的混沌弱信号检测临界阈值确定^{*}

陈 鹏^a, 芮国胜^b, 王 林^a, 张 嵩^b, 吴 芳^b

(海军航空工程学院 a. 信号与信息处理山东省重点实验室; b. 电子信息工程系, 山东 烟台 264001)

摘要: 混沌检测系统临界阈值的确定是建立混沌检测系统的核心问题, 临界阈值的精度决定了可检测信号的精度。目前相轨迹图观察法已经无法满足快速确定精确的系统临界阈值的需求, 利用 Lyapunov 指数等量化指标检测临界阈值的方法计算量大、算法复杂、时间消耗大, 且消耗大量计算资源, 无法在 GPU 上实现并行程序设计; 系统相轨迹过零周期数相变判别算法在检测相同精度阈值情况下较 Lyapunov 指数算法有相同的检测精度, 同时更易于利用现代高性能计算工具 GPU 实现并行程序设计。因此, 在系统相轨迹过零周期数阈值判别算法的基础上提出了基于 GPU 的混沌弱信号检测临界阈值并行检测算法, 实验显示, GPU 并行系统临界阈值检测算法在幅值递增步长 10^{-6} 下较 CPU 串行算法可以实现近 90 倍的加速比, 使高精度临界阈值能够在短时间内准确确定。

关键词: 混沌微弱信号检测; 临界阈值; GPU; 过零周期数

中图分类号: TN911.7

文献标志码: A

文章编号: 1001-3695(2014)04-1051-04

doi:10.3969/j.issn.1001-3695.2014.04.023

Determination of threshold value of chaos-based weak signal detecting based on GPU

CHEN Peng^a, RUI Guo-sheng^b, WANG Lin^a, ZHANG Song^b, WU Fang^b

(a. Signal & Information Processing Provincial Key Laboratory in Shandong, b. Dept. of Electronic Engineering, Naval Aeronautical & Astronautical University, Yantai Shandong 264001, China)

Abstract: Determination of the threshold value is the key to chaos-based weak signal detection. The precision of the threshold value invariably defined the precision of the detection. Currently, determining the threshold value by observing the phase space trajectory could hardly satisfy the demand for an accurate threshold value in a short time, while the Lyapunov index method was complicated and it took a lot of time and computing resources, which was not convenient to program on the GPU platform. Comparably, the detecting method based on zero-crossing number could achieve an accurate threshold value with lower computing resources and complexity, which could well applied to the modern high performance programmable GPU. By using the zero-crossing number criterion, this paper proposed a new parallel algorithm of detecting the accurate threshold value based on GPU platform. Experiments show that the parallel algorithm based on GPU can accelerate the determination of the threshold value up to 90 times compared with the single-thread algorithm at the amplitude step length of 10^{-6} . This parallel algorithm brings light to the quick detection of the threshold value of different chaos detecting system.

Key words: chaos-based weak signal detection; threshold value; GPU; zero-crossing number

0 引言

自 1992 年 Birk 等人^[1]首次提出将混沌理论应用于微弱信号检测至今, 国内外许多学者都对该技术进行了深入研究。其中利用 Duffing 混沌振子检测微弱信号的研究取得了许多的研究成果。在利用 Duffing 混沌振子进行混沌微弱信号检测中, 一个十分重要的参数就是混沌振子的相变临界阈值。调节策动力幅值使检测系统状态处于临界混沌状态是利用混沌振子检测微弱信号的关键环节。准确地确定混沌系统的相变临界阈值, 将直接影响混沌系统的可检测信号强度、检测精度以及检测稳定性。确定系统相变临界阈值关键在于系统运动状态判别, 目前常用的方法包括 Melnikov 方法、Lyapunov 指数判

别法、相轨迹观察法等。Melnikov 方法通过理论推导得到临界阈值近似值, 但在实际应用中往往存在较大误差, 无法精确确定系统状态跃变阈值^[2]; Lyapunov 指数判别法是广泛应用于判断系统混沌特性的量化指标, 但求解系统 Lyapunov 指数计算量大、算法复杂度高, 当系统参数发生微小改变往往也无法十分快速地确定新的系统参数下系统的相变临界阈值^[3~5]; 在实验室仿真中常用的方法是在不同参数下根据经验选定一个幅值, 并在微调幅值的同时观察相轨迹图来确定系统相变临界阈值。随着对混沌微弱信号检测研究的深入, 往往需要对各种参数设置下的混沌微弱信号检测系统建模。此时一种能够快速、准确地检测系统相变临界阈值的方法已经成为研究中亟待解决的问题。

收稿日期: 2013-04-18; 修回日期: 2013-06-05 基金项目: “泰山学者”建设工程专项经费资助项目; 海军航空工程学院“研究生创新基金”资助项目

作者简介: 陈鹏(1989-), 男(土家族), 湖北鹤峰人, 硕士研究生, 主要研究方向为混沌微弱信号检测、GPU 高性能计算及其应用等(chenpeng89@foxmail.com)。

金涛等人^[6]提出了基于过零周期数判别混沌相变判别方法,该方法在混沌检测系统临界阈值的确定上具有高精度、低复杂度的特点。本文基于过零周期数判别方法,提出了 GPU 架构下对策动力幅值进行遍历计算统计对应过零周期数的并行阈值检测算法,达到实现快速、高效、高精度地检测不同系统参数下的系统临界阈值的目的。实验表明,该算法与传统串行算法相比具有几十倍的加速比,能够使不同精度要求混沌系统的临界阈值在短时间内被确定。

1 基于过零周期数的混沌相变临界阈值检测算法

在微弱信号检测领域应用较为广泛的 Holmes 型 Duffing 方程标准形式如下:

$$\ddot{x}(t) + k\dot{x}(t) - x(t) + x^3(t) = \gamma \cos(t) \quad (1)$$

其中: $\gamma \cos(t)$ 为系统周期策动力, k 为阻尼比, $-x(t) + x^3(t)$ 为非线性恢复力。为了能对不同频率的信号进行检测,将式(1)变换为微分方程组形式:

$$\begin{cases} \dot{x} = \omega y \\ \dot{y} = \omega(-\delta y + x - x^3 + \gamma \cos(\omega t)) \end{cases} \quad (2)$$

当参数 ω, δ 确定后,策动力幅值 γ 在逐渐由 0 增大的过程中,系统相轨迹 (x, y) 将历经同宿轨道、分叉、混沌轨迹、临界状态,最后进入大尺度周期状态。在微弱信号检测中主要利用的是临界混沌状态和大尺度周期状态,如图 1 所示。

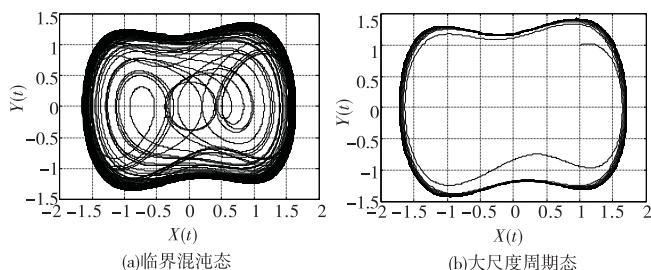


图1 临界混沌态与大尺度周期态的系统相图

过零周期统计法是基于大尺度周期状态,系统相轨迹收敛于一条确定轨迹,系统数值解 $x(t)$ 具有恒定周期,表现为相平面轨迹以恒定周期通过相平面坐标轴。图 2、3 所示为相平面和时域范围内 $x(t)$ 的过零点示意图,图 4 所示为仿真时间 0.005 s, $k=0.5$, $\omega=(2\pi \times 60000)$ rad/s 时不同幅值下 $x(t)$ 过零周期数。

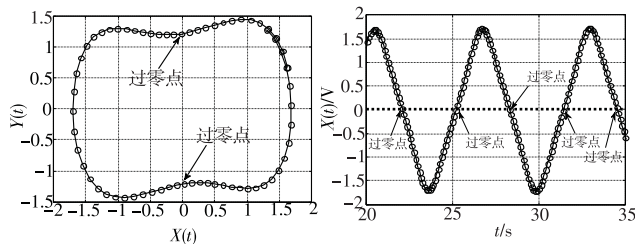


图2 相轨迹平面 $x(t)$ 过零点示意图 图3 $x(t)$ 时域波形过零点示意图

$x(t)$ 具有与策动力相同的频率,其过零周期数随策动力幅值的增加呈现递增趋势并稳定于定值 N ,时间 T 内过零点数为 $N = T \times f \times 2$, f 为策动力频率。在混沌状态,系统过零周期数呈现大幅度波动状态,当系统进入大尺度周期状态,系统过零周期数将处于一个稳定水平。在数值求解系统微分方程的同时通过记录过零周期数即可快速判断系统混沌和大尺度周期状态。

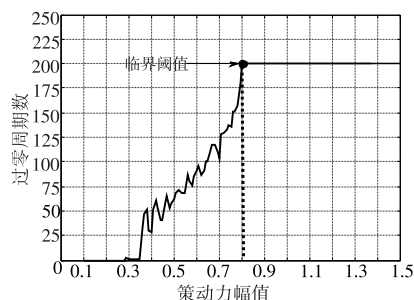


图4 过零周期数与策动力幅值关系

1.1 算法精度分析

Lyapunov 指数是衡量系统动力学特性的一个重要定量指标,它表征了系统在相空间中相邻轨道收敛或发散的平均指数率。通过判断最大 Lyapunov 指数是否大于零可以直观地定量描述系统是否处于混沌状态。

为了验证基于过零周期数的混沌相变判别算法的精度可靠性,本文以 Lyapunov 指数的阈值判别结果作为系统临界阈值的精确解,在不同策动力幅值步长下对系统幅值遍历求解系统微分方程并统计过零周期数与其进行比较。

图 5(a) 所示为在幅度递增步长为 10^{-4} 时不同策动力幅值下系统的过零周期数;图 5(b) 所示为相同条件下,系统最大 Lyapunov 指数的变化图。从图中可以看出,过零周期数算法确定的系统临界阈值为 0.7943, Lyapunov 指数算法确定的系统临界阈值为 0.7945,相对误差为 0.025%。图 6 所示是策动力幅值为 0.7943 和 0.7944 时系统相轨迹图,相图上直观地反映出了阈值左右系统相轨迹由临界混沌状态到大尺度周期状态的变化。

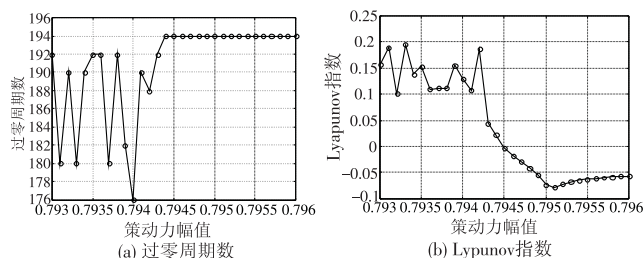


图5 不同策动力幅值下过零周期数与 Lyapunov 指数变化图

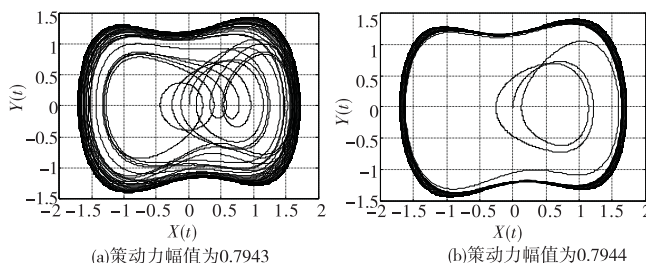


图6 不同策动力幅值系统相轨迹图

表 1 不同幅值递增步长下两种算法确定的系统相变阈值比较

幅值递增步长	过零周期数算法 相变临界阈值/V	Lyapunov 指数算法 相变临界阈值/V	相对误差/%
10^{-3}	0.795	0.795	0
10^{-4}	0.794 3	0.794 5	0.025
10^{-5}	0.794 39	0.794 48	0.011
10^{-6}	0.794 394	0.794 480	0.011

由表 1 可以看出,在不同幅值递增步长下,过零周期数算法与 Lyapunov 指数算法相比均表现出很高的精度,能够实现不同精度下的相变阈值确定。

1.2 算法复杂度分析

本文通过在CPU为主计算单元的计算机上利用MATLAB软件对(0,1)范围内不同幅度递增步长下两种算法进行计算比较,其运算时间如图7所示,结果显示两种算法运行时间相差约10倍。同时注意到,两种算法在确定精度更高的阈值时均面临庞大的计算规模,耗时巨大,无法快速确定高精度的临界阈值。一种能够快速、准确确定系统临界阈值的并行检测算法成为研究亟待解决的问题。

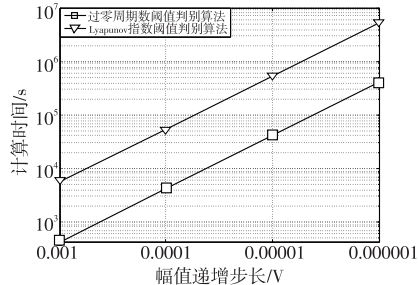


图7 算法运行时间比较

考虑到Lyapunov指数算法需要对混沌微分方程数值解集的Jacobi矩阵进行大规模乘法运算并求其特征值,需要消耗大量计算机资源不适合在GPU中进行并行运算。而过零周期数算法较Lyapunov指数算法复杂度更低,但同时又有较好的计算精度,并行程序设计中无须保存大量数值解,在求解系统数值解的同时即可进行过零周期数的统计,在计算机资源的利用上更高效,更便于GPU并行程序处理,本文在系统相轨迹过零周期数阈值判别算法的基础上提出了基于GPU的混沌弱信号检测临界阈值并行检测算法。

2 CPU-GPU 异构并行计算架构

可编程图形处理单元(GPU)的快速发展使得GPU已经逐渐成为主流高性能计算单元,在CPU-GPU异构计算平台下,利用硬件的可编程性和并行性来实现一些复杂模型的快速、通用计算已经成为当今的研究热点之一^[7]。

2007年NVIDIA公司推出的CUDA(compute unified device architecture,统一计算设备架构)是一个全新的用于GPU高性能计算的软硬件架构。CUDA架构中,编程及执行模型如图8所示^[6]。当程序执行到并行部分,由CPU主线程调用GPU设备端进行数据拷贝、运算,并在运算结束后将结果传递回CPU主机端。

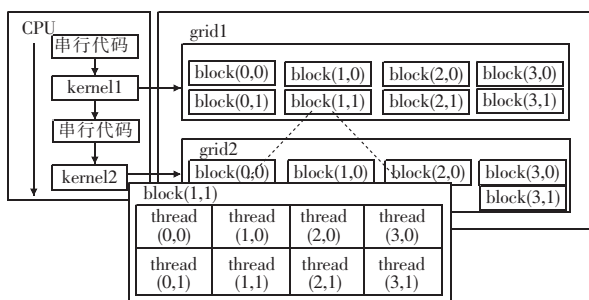


图8 CUDA编程和执行模型

CUDA架构的线程组织结构分为三层。其中最小单位为线程(thread),一组线程组成一个线程块(block),一组线程块组成一个线程网格(grid),并以一维、二维或者三维的索引标示线程,使得程序员能够对各种域(向量、矩阵,或者空间)中

的数据进行更直观、自然的划分和运算^[8-10]。

除了编程和执行模型,CUDA也有自己的存储器模型,如图9所示。

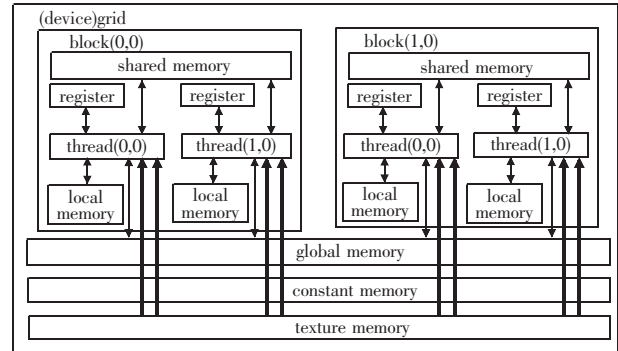


图9 CUDA中的多级存储器模型

通过合理地组织线程规模、内存分配和程序调度,在GPU为协处理器的异构平台下,对大规模数据处理往往能够获得巨大的加速比。这一快速高效而低成本的高性能计算模式正逐渐成为现代科学实验的方法。

3 基于CPU-GPU异构平台的并行算法实现

3.1 串行算法流程

下列所示是过零周期数阈值判别算法的串程序流程,通过for循环内部对幅度值进行遍历,在每个幅度值下统计系统过零周期数并对系统状态进行判别,确定以系统过零周期数由波动进入持续稳定状态的跃变点对应幅值作为系统相变临界阈值。

串程序算法:对不同幅值进行循环遍历

初始化

for $i = 1 : N$ // N 为遍历循环幅值数目

$\gamma = i \times \Delta k$ // Δk 为幅度递增步长

for $j = 1 : M$ // M 为迭代次数

// h 为迭代时间步长

$x(j+1) = x(j) + \omega h y(j)$

$y(j+1) = \omega h (-\delta y(j) + x(j) - x^3(j) + \gamma \cos(\omega h j))$

if $x(j+1) \times x(j) \leq 0$

$\text{num}[i] = \text{num}[i] + 1$

end

end

end

分析数据,寻找过零周期数跃变的幅值

3.2 并行算法流程

CUDA采用的SIMT(single instruction, multiple thread,单指令多线程)执行模型可以实现对for循环内部数据处理具有独立性的部分进行并行多线程处理,达到更加高效的目的。在异构平台下CPU主线程将待检测幅值数据传递给GPU协处理器并分配至每一个计算核心,GPU并发多线程在不同幅值下独立计算出各种条件下系统过零周期数并保存至全局寄存器,最后将结果传递给CPU主机端进行筛选。GPU并行程序算法流程如下所示。

并行程序算法:一个计算核心对应一个幅值进行并行检测

//计算每个计算核心的标示号

$\text{id} = \text{blockIdx}.x \times \text{blockDim}.x + \text{threadIdx}.x$

从全局存储器内读入微分方程各参数及存储幅值的数组 A

//将每个计算核心处理的幅值存入寄存器

$\gamma = A[\text{id}]$

初始化计数器

for $j = 1 : M$

```

x(1) = x(0) + ωh y(0)
y(1) = ωh(-δy(0) + x(0) - x³(0) + γ cos(ωh))
if x(1) × x(0) ≤ 0
    num = num + 1
end
x(0) = x(1)
y(0) = y(1)
end
//将计数器数值保存至全局存储器
count[id] = num

```

3.3 线程安排及内存分配

按照 CUDA 的执行模型,线程网格中的各个线程块会被分配到 GPU 的各个 SM 中执行,实际运行中线程块会被分割为更小的线程束(warp)。在 Tesla 架构的 GPU 中,一个线程束由 32 个线程组成,每个 SM 至少需要 6 个 active warp 才能有效隐藏流水线延迟。所以在线程块和线程网格的维度设计上要尽量保证其数值为 32 的整数倍^[11]。

通常在线程组织上,根据问题规模,在确定线程块的尺寸和维度后再确定线程网格的尺寸和维度。可采用以下方法计算(以 x 轴为例), N 为问题计算规模。

$$\text{gridDim}_x = \frac{N + \text{blockDim}_x - 1}{\text{blockDim}_x} \quad (3)$$

式(3)保证线程网格在线程块的分配上是在已设定的线程块规模和给定问题总规模下的最小整数。

在内存分配上,Tesla C2050 GPU 为每个线程块分配了 32 768 个寄存器文件,为了高效使用寄存器文件,避免 GPU 设备与主机之间产生大量数据交流,程序设计中的变量主要以私有寄存器为主。通过显存将遍历计算的幅度值传递给每条线程的私有寄存器,在 GPU 内部完成微分方程的迭代求解并统计过零周期数,最后将统计结果由显存传递回计算机 CPU 主机端处理。程序流程中只进行两次在设备和主机之间的数据传递,密集型计算由 GPU 完成。

由 CUDA 套件中的 Nsight Monitor 软件对每条线程分析显示,每条线程层占用 49 个寄存器文件,因此在每个线程块中受寄存器个数的限制,能够安排的线程上限为 668。考虑线程块尺寸尽量应为 32 的整数倍,文章中实验确定的线程块维度安排为 $640 \times 1 \times 1$,根据具体数据规模和式(3)可以确定线程网格的尺寸。

4 仿真实验

4.1 仿真环境

本实验采用的曙光 W580 工作站配备了 Intel Xeon 12 核 CPU 和两块 NVIDIA Tesla C2050 GPU,系统内存为 24 GB。软件方面采用 CUDA C 语言,开发环境使用 Visual Studio 2010。

4.2 仿真结果分析

表 2 为实验仿真结果,实验基于前文所述串行和并行算法,分别在 CPU 和 CPU-GPU 异构平台上利用 C 语言和 CUDA C 语言编程,对 $k=0.5$, $\omega=(2\pi \times 60000)$ rad/s 时混沌系统的幅值进行不同精度下的遍历计算。从计算结果中选择过零周期数稳定的起始点作为系统临界阈值,其阈值检测结果与 3.2 节 GPU 并行程序算法一致。由表 2 的计算结果可以看出,CPU 单线程下和 GPU 多线程下,对不同幅值精度下的阈值计算结果是一致的。同时在不同的幅值递增步长下得到的阈值精度不同,即对混沌检测系统而言系统对微弱信号的检测门限

不同,精度越高,系统可检测信号门限越低。随着精度的提高,计算机需要处理的数据规模增大,在 CPU 串行算法程序中处理时间消耗成倍增长,在 GPU 并行处理算法下的计算时间增长并不成比例,主要原因是 GPU 适合对大规模高密度型数据进行处理。在幅值递增步长较小时,数据处理规模不大,并没有充分利用 GPU 的所有计算单元,在计算时间上相较于 CPU 串行算法加速比不大。随着递增步长的减小,计算规模增大,逐渐填充了 GPU 所有计算单元,加速比逐渐增大,此时又受限于 GPU 硬件的计算单元限制使加速比不再随着计算规模的增大而不断增长了。在本文实验中,GPU 并行算法能相较于 CPU 串行算法有效实现近 90 倍的加速比,使高精度阈值能在短时间内被准确确定。

表 2 不同计算平台下计算结果比较

幅值递增步长	GPU 计算结果		CPU 计算结果		加速比
	相变临界阈值/V	计算时间/s	相变临界阈值/V	计算时间/s	
10^{-3}	0.795	0.59	0.795	5.13	8.67
10^{-4}	0.794 3	1.08	0.794 3	50.14	46.60
10^{-5}	0.794 39	5.94	0.794 39	501.09	84.31
10^{-6}	0.794 394	55.79	0.794 394	4 987.59	89.41

5 结束语

本文在系统相轨迹过零周期数阈值判别算法的基础上提出了基于 GPU 的混沌弱信号检测临界阈值并行检测算法,充分利用 GPU 的高性能计算单元对待测系统策动力幅值遍历并统计过零周期数进而确定不同参数下系统的相变临界阈值。研究表明,该并行算法能有效确定不同精度需求下的系统临界阈值,且在幅值精度为 10^{-6} 下,相较于 CPU 单线程算法具有近 90 倍的加速比,能够使得高精度的临界阈值在短时间内被准确确定。这对于进一步深入研究不同参数条件下的混沌微弱信号检测带来了更高效的仿真手段,对自适应阈值判断、混沌自适应信号检测系统等研究都将带来积极的促进作用。

参考文献:

- [1] BIRX D L, PIPEMBERG S J. Chaotic oscillators and complex mapping feed forward networks (CMMFFNS) for signal detection in noisy environments[C]//Proc of IEEE International Joint Conference on Neural Networks. 1992:881-888.
- [2] 李亚峻. 混沌判据—Melnikov 算法的研究[D]. 吉林:吉林大学,2004.
- [3] 杨红英,叶昊,王桂增,等. Duffing 振子的 Lyapunov 指数与 Floquet 指数研究[J]. 仪器仪表学报,2008,29(5):927-931.
- [4] 程凤芹,曲娜. Floquet 指数确定混沌检测系统阈值的新方法[J]. 气象水文海洋仪器,2011(2):20-23.
- [5] 张宾,李月,马海涛. 微弱信号混沌检测临界阈值的 Lyapunov 指数算法[J]. 地球物理学进展,2003,18(4):748-751.
- [6] 金涛,柏逢明. 基于过零周期技术的混沌检测系统状态阈值判据研究[J]. 长春理工大学学报:自然科学版,2010,33(1):67-69.
- [7] OWENS J D, HOUSTON M, LUEBKE D, et al. GPU computing[J]. Proceedings of the IEEE, 2008, 96(5):879-899.
- [8] NVIDIA CUDA C programming guide version 4.2[K]. Santa Clara: NVIDIA Corporation, 2012.
- [9] 甘新标,沈立,王志英. 基于 CUDA 的并行全搜索运动估计算法[J]. 计算机辅助设计与图形学报,2010,22(3):457-460.
- [10] 方旭东. 面向大规模科学计算的 CPU-GPU 异构并行技术研究[D]. 长沙:国防科学技术大学,2009.
- [11] FARBER R. 高性能 CUDA 应用设计与开发[M]. 北京:机械工业出版社,2013:11-13.