

课 程 大 作 业

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓 名:** | 张凯珺  李宗宇  拓欣 | | **学 号：** | 923130830237  923130830223  922114740127 |
| **课 程:** | 基于解释性语言的高效科学计算 | | | |
| **题 目:** | 求复杂函数在积分 | | | |
| **任课教师:** | 周毅 | 能源与动力工程学院 | | |

2024 年 4 月

# 基于蒙特卡洛法使用Numba和Julia两种方式实现题目三

我组选择题目三：求复杂函数在积分，数学方法为蒙特卡洛法，代码实现方法为Numba和Julia。两种语言的源代码详见附件。

下面我们将介绍我们的三种算法实现以及我们在工程上采取的一些手段。

## 方法一：蒙特卡洛法的一般实现

对于蒙特卡洛法的一些简单的知识老师在课上有讲到过，当时我们求的是圆周率，我们把这种方法推广到更一般的情况，考虑到一个关于x的一元函数，若要求出其在区间内的面积，我们必须求其的积分，但当其原函数不能被一个已知的连续函数或一个收敛的无穷级数表示时，我们使用原有的微积分算法就会陷入僵局，但我们可以采用数值积分。

根据中值定理和积分的定义可知，直角坐标系内连续函数在其区间内的面积等于区间上面积的平均值乘以区间长度，即：

我们可以浅显的理解其为我们的要求的积分值，我们的算法也将围绕着其展开。

接下来解释代码(以Julia为例)：

*文本

描述已自动生成*

第一个函数func(x)定义了我们的原函数。

第二个函数共有四个参数，第一个参数*func*为使用的原函数，第二个参数为区间最小值，第三个参数为区间最大值，第四个参数为总共进行的次数。

函数的第一行里我们使用了*Uniform*函数，生成了*times*个均匀分布[bottom,top]区间内的数并将其整体赋值给了*random\_x*变量，这里我们就将这一问题**向量化**了，按照传统的思维，我们要通过一个循环逐个地将随机值插入到list中，但Julia这类科学语言以及Numpy等科学计算库都提供了向量化的函数，避免了提高时间复杂度。函数第二行亦是如此，我们将整个*random\_x*传入到原函数中并得到统一组织的返回值。

这里的times指数则代表着生成的随机数总数，数量越多，算法执行用时越长，图为该算法在10^4、10^5、10^6、10^7、10^8次情况下的运行时间图。

图表, 折线图

描述已自动生成

第三行则更容易理解，这里用到了数学的**第一积分中值定理**，得到的值即可看作积分的近似值。

总结：

这个函数的主要目的是计算一个数学函数在指定区间[bottom, top]上的积分的近似值。它通过以下步骤实现：

1. 生成大量（*times*个）在指定区间内均匀分布的随机数*x*。

2. 使用*func*函数计算这些*x*值对应的*y*值。

3. 计算*y*值的平均值。

4. 将*y*的平均值乘以区间的长度（*top - bottom*），得到积分的近似值。

## 方法二：重要性采样法

在蒙特卡洛法的基本原理基础之上，还存在着两种非常有名的算法实现：重要性采样法和分层采样法。

接下来解释代码(以Julia为例)：

*文本

描述已自动生成*

这个函数前两行和第一种算法一致。

第三行主要应用了概率统计与数理统计的专业知识，期望值使用积分形式。1/2pi代表着均匀分布， 应用到被积函数值与对应概率密度的比值，它们的值乘以区间长度则为积分的近似值。图表, 折线图

描述已自动生成

总结：

该方法专业性较强，可能涉及到专门的采样方法，比如逆变换采样、接受-拒绝采样或者马尔科夫链蒙特卡洛方法（MCMC）等，但对于该问题来说就是使用正常期望值的计算方法即可，我们亦受数学知识限制无法深入解释。

## 方法三：分层采样法

分层采样法可以看作是把蒙特卡洛法进一步微分，把一个任务拆分成了数个任务，在目前使用的三种算法中，这种方法在样本量足够大时是速度最快、计算值最接近答案的方法，也是我们接下来在工程角度上着重优化的算法。

接下来解释代码(以Julia为例)：

*文本

描述已自动生成*

这个函数新增了一个参数layers，即为分层层数，与样本量成正比时可以相对提升算法速度，层数越多结果越接近，当然计算量和时间时间也会增开。

我们提前在外部初始化dist和width，避免在循环内重复计算，增加性能开支。

循环内部可以看作是一个小的蒙特卡洛实现，只不过他是在一小块的区域上进行计算，所以值要除以分层数。

在第12行，我们将layers个层的积分累加，最后返回的就是整个区间的积分了。

图表, 折线图

描述已自动生成

总结：

分层采样的基本思想是将整个积分区间分成若干个子区间，然后在每个子区间内独立地生成随机样本，并计算每个子区间的积分近似值，最后将这些近似值加权平均得到整个区间的积分近似值，目前其性能及准确率都很让人满意，因此也是我们接下来优化的核心算法。

## 工程手段一：Numba JIT及多线程

我们选择的课题两种实现方式之一就是Numba，所以我们去了解了一下Numba的特性，为以上算法开启了JIT、nopython、nogil、parallel模式，速度提升极大。

图片包含 文本

描述已自动生成

在Numba的加持下，我们的代码执行速度可以接近原生C/CPP执行速度，相较于原理速度提升了十余倍。手机屏幕截图

中度可信度描述已自动生成

## 工程手段二：通过NVIDIA CUDA使用显卡进行计算

## 文本 中度可信度描述已自动生成

由于GPU精于浮点运算，这正迎合了我们计算亿个以上浮点数的需求，因此我们选择了可以和Numpy“一键转换”的CuPy库，在算法不变的基础上将运算过程移植到了GPU上，在NVIDIA RTX 3060 Laptop显卡下进行了测试，性能提升数个量级。

可以看到在样本量足够大时，CUDA的计算速度可以达到CPU的50倍。

而这三种算法在CUDA的加持下速度差距也越来越大，使用了CUDA的分层算法比不使用时速度提升了200多倍，重要性采样算法的速度提升了700倍，而事实上我们只是把全局的*Numpy*改成了*cupy*就可以实现如此恐怖的提升。

图表, 条形图

描述已自动生成图表, 折线图

描述已自动生成

## 工程手段三：算法向量化

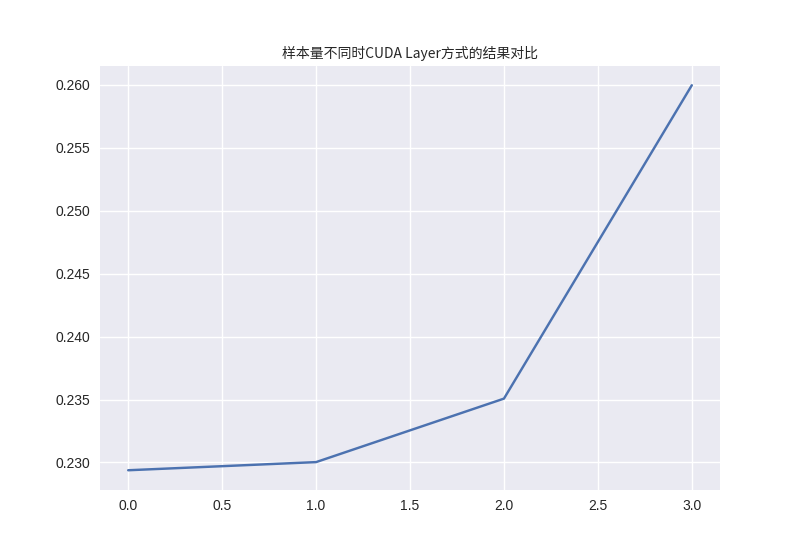
图表

描述已自动生成 在此之前我们的前两种算法亦使用了向量化的思想，而向量化的思想在分层采样中的优势则更大了，下图为我们的两种分层采样Python语言实现。

相较于*for*循环，我们使用的*cp.arange*函数将这个问题向量化，将一个时间问题转换成了数学问题，同样的算法在向量化后运行速度提升了400倍，可见在科学计算中向量化思维的重要性。

文本

描述已自动生成

图表, 折线图

描述已自动生成

## 总结

在我们研究这一题目的过程中，我们接触到了许多过去未曾接触到的东西。最初我们实现了相关算法，但是计算速度却相当的慢，虽然我们按照老师课上所教授的那样运用了Numba的*nopython=true*功能，但是得到的速度提升却很有限。在查看了电脑的CPU使用情况后我们发现只有一个核在跑，于是我们去查询资料并知道了*parallel*等功能，在开启后程序的执行速度突飞猛进，所有的CPU核心都被使用。

图表, 条形图

描述已自动生成在解决了工程上的一些问题后，我们又开始通过翻阅专业书籍和搜集论文等手段了解到了许多基于蒙特卡洛法的积分计算改进方法，例如我们已经实现的重要性采样法和分层采样法，借助这些算法我们的计算速度和结果精确程度得到了很大的提升。

除此之外，我们还了解到了通过遗传算法提升结果准确率、通过神经网络迭代等方法，但因为时间不足且我们目前大一、大二所学到的知识不足于让我们深入理解相关算法故我们没有进行实现。

图表, 折线图

描述已自动生成于是我们又开始着手于研究通过工程手段进一步提升计算量，通过增大计算量来提升结果精度并减少计算时间，受到人工智能技术的启发，我们尝试了通过NVIDIA CUDA来进行计算并且成果斐然，同时我们也利用向量化的思维改进了算法，诸如此类的方式我们还有待进一步探索。

我们相信一定还有更多方法可以用来提升精度以及减少计算用时，我们的作业代码已经开源到<https://github.com/Hobr/monte-carlo-integration>供大家互相学习交流以及提出改进意见，我们也希望以此为契机来让我们更加深入地学习和参与到科学计算这一领域中来。在此我们非常的感谢老师您的指导。