README

工学院 2100011029 渠成锐

2023 年 6 月 24 日

目录

1 并行技术 **2**

2 编译安装运行指南 **2**

2.0.1 编译链接依赖库 2
2.0.2 openmpi 2
2.0.3 scalapack 2
2.0.4 lapack 3
2.1 编译运行指令 3

3 文件结构 **3**

3.1 代码结构 4
3.2 输入文件结构 4
3.3 输出文件结构 4

4 数据结构的设计 **5**

5 优化工作 **5**

1 并行技术

本项目设计了 openmp 版本与 cuda 版本

openmp 版本: 在 main 函数及相关库中仅使用 openmp, 在附加题二的相关文件 scalapack.cpp 中使用 MPI。

备注：由于有内存限制的要求，在积分时利用 MPI 进行数据传输容易导致运行时最大内存超过 12g，且代码会较为复杂，起到的效果反而不如仅使用 openmp，故在求积分时只使用了 openmp。

cuda 版本： 见 cuda_version 文件夹中的 README

2 编译安装运行指南

2.0.1 编译链接依赖库

2.0.2 openmpi

在命令行依次运行如下命令：

1. 下载最新发行：wget https://download.open-mpi.org/release/open -mpi/v4.1/openmpi-4.1.5.tar.gz
2. 解压安装包：tar -zxvf openmpi-4.1.5.tar.gz
3. 进入安装包目录：cd openmpi-4.1.5
4. 修改安装地址：./configure –prefix=/home/Username/software/openmpi-4.1.5 –enable-shared
5. 安装：make install -j12
6. 添加路径：export PATH=/home/Username/software/openmpi-4.1.5/bin:$PATH
7. export LD_LIBRARY_PATH=/home/Username/software/openmpi-4.1.5/lib:$LD_LIBRARY_PATH
8. 运行 which mpicxx 检验是否安装成功

2.0.3 scalapack

利用包管理工具进行安装，在命令行运行 apt install libscalapack-openmpi-dev

2.0.4 lapack

利用包管理工具进行安装，在命令行运行 apt install install liblapack-dev liblapacke-dev

2.1 编译运行指令

本项目采用 cmake 进行构建，进入项目目录后在命令行运行如下指令：

1. cmake -B build
2. cmake –build build
3. ./build/big (需要先将输入文件放入 input 文件夹中)

main 函数中采用 lapacke 接口对得到的矩阵，并将特征值和特征向量输出至 output 文件夹中，同时生成 matrix.txt 文件记录积分得到的实对称矩阵及维数。

如果要修改使用的线程数，只需在 main.cpp 中将 nthreads 的值修改为对应值即可。对于附加题 2，另外设计了利用 scalapack 接口实现的并行对角化，在命令行中运行如下指令进行调用：

1. mpicxx scalapack.cpp -lscalapack-openmpi
2. mpirun -np 4 ./a.out

该程序将读入 matrix.txt 并利用 scalapack 接口进行对角化，将特征值和特征向量输出至 output 文件夹中。（注：本程序默认用 4 核进行对角化）

3 文件结构

1. input(文件夹)：存放输入文件
2. output(文件夹)：存放输出文件
3. tools(文件夹)：存放用到的库

3.1 代码结构

1. main.cpp: 主函数
2. input.h:input 类的定义，读入文件输入
3. input.cpp:input 类的实现
4. timer.h:timer 类的定义，实现记录函数运行时间的功能
5. timer.cpp:timer 类的实现
6. spline.h:spline 类的定义，实现三次样条插值
7. spline.cpp:spline 类的实现
8. diago.h:diago 类的定义，实现实对称矩阵对角化（调用 lapacke 接口）
9. diago.cpp:diago 类的实现
10. scalapack.cpp: 调用 scalapack 接口，采用并行方式对实对称矩阵对角化

3.2 输入文件结构

1. INPUT.txt: 由 input 类读取的输入文件，设置计算任务及输入矩阵文件路径
2. 调试时将空间相关输入文件放置在 input 文件夹中，并将 INPUT.txt 中的路径设置为相应路径即可。

3.3 输出文件结构

输出文件均存放至 output 文件夹中

1. matrix.txt：积分得到的矩阵，第一行为维数，后面为矩阵元
2. eigenvalues.log: 调用 lapacke 接口对角化得到的特征值，从大到小排列
3. eigenvectors.log: 调用 lapacke 接口对角化得到的特征向量，顺序和特征值对应（按行排列）

4. scal_eigenvalues.log: 调用 scalapack 接口对角化得到的特征值，从大到小排列

5. scal_eigenvectors.log: 调用 scalapacke 接口对角化得到的特征向量，顺序和特征值对应 (按行排列)

4 数据结构的设计

项目中在 input 类的对象实例中存储空间的相关信息，利用一维数组存储 V 的分布（这是最占存储空间的部分），在 diago 类的对象实例中存储利用追赶法求解得到的样条插值程序的各节点导数值（一个比较小的数组），在 main 函数中动态分配内存存储积分得到的矩阵（占用空间也不大）

5 优化工作

为了加快积分速度，本项目在积分前先进行了一遍预处理，生成一维的 flag 矩阵，取值为 0 或 1，如果给定两个点之间的距离大于 2 倍的截断值，则将相应的 flag 矩阵元置为 1, 在积分循环中直接跳过，减少计算量。

对于多定点的情形，本项目进行特别的优化，将定点放在外层循环，内层循环是对空间格点的循环。在内层循环中，循环 i 指标时，格点到定点的 x 方向上的距离平方已经能计算出来了，若其大于截断值的平方，则直接跳过 (比较距离的平方是由于 sqrt 函数计算开销比较大，会减慢积分时间)；对于 j,k 指标的循环有类似的处理，由于最后得到的矩阵具有很大的稀疏性，上述优化可以大大加快计算速度，对于 50 个定点，V-512 的情形，用 8 个线程跑，积分用时在 0.7s 左右；V-1024 的情形，用 8 个线程跑，积分用时在 6s 左右。

值得一提的是，上述优化工作没有增加任何计算量，中间计算出来的某一方向上距离的平方均为最后需要插值计算的势函数的中间值，在最内层循环都能直接用上。唯一的开销是多了 4 个 double 型值的存储和 3 个判断语句，对于存储空间和运行时间的影响几乎可忽略。

另外, 为了加快积分的速度，在循环时采用动态调度，并利用规约子句进行求和操作（这个会比原子操作和临界区操作快一些，因为规约是每个线程创造属于自己的副本，同时这里 Hamilton 矩阵的大小不大，创建多个副本几乎不会影响内存空间大小）