

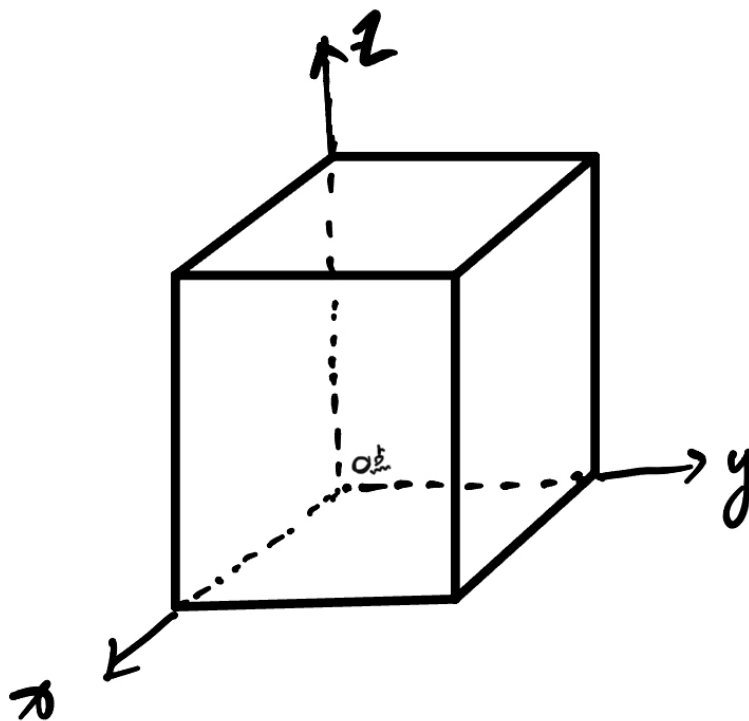
并行程序设计——期末大作业（版本1）

基本要求：

1. 可以1人单独完成，也可以2人或者3人组队完成，但不同人数的队伍要求不完全相同。
2. 2人或者3人组的队，其完成作业过程需要用Git软件管理方式来完成团队代码开发完成。完成组队后，小组成员需要创建gitee私有项目，并将两位助教和老师添加到小组成员内，以便我们查看小组内代码贡献的具体信息。
3. 编程语言可以选用Fortran，C或者C++完成。
4. 多文件编程，不能把所有程序都放在一个文件里，建议使用makefile或者cmake来管理程序。
5. 代码的关键部分需要用并行来加速，并行方式可以选择MPI，openMP或者CUDA的一种，也可以多种结合。

具体要求：

1. **确定长方体网格大小：**程序需要定义一个边长分别为 l_x 、 l_y 和 l_z 的三维网格长方体。
2. **生成均匀网格：**程序需要创建一个均匀分布的网格，其中每个方向上都包含 n_x 、 n_y 和 n_z 个点。这些格点之间的距离相等，因此整个网格是均匀的。
 1. 这些网格后面被用于计算空间上的积分。
 2. 共有 $ngrid = n_x \times n_y \times n_z$ 个格点
 3. 长方体网格坐标系的建立如下，以O点为原点，建立笛卡尔坐标系。



3. **读入两个定点坐标：**程序需要从用户输入文件 `POINTS.txt` 中读取两个定坐标点（笛卡尔坐标），这两个坐标点的范围落在这个长方体内，假设这两个坐标点为 (x_1, y_1, z_1) 和 (x_2, y_2, z_2) 。
4. **读入一维径向分布函数：**程序将从分布函数文件 `Distribution.txt` 中读取一维径向分布函数 $f_l(r)$ ，这里 `l` 取 `1`，附加题三中该取值可以大于1，读入后每个定点坐标都有一份 $f_l(r)$ 函数。接下来，以每一个定点坐标为球心，将 $f_l(r)$ 函数的 $r = 0$ 点放在定点坐标上，我们接下来会将这个一维径向分布函数扩展到三维均匀网格上，以待后续操作。

🕶️ 径向分布函数：

1. 一维函数，**自变量**是某点距离定坐标点的距离，**因变量**为分布函数的值。
2. 一维的分布函数有截断半径，超过截断半径函数值为0。
3. 在非周期性边界条件下，超出长方体网格范畴的分布函数可以当作0。

5. **插值：**根据格点到读入定点的距离，在三维网格上进行插值得到每个格点上对应的分布函数值 $f_l(r)$ 。
 - a. 对于三维均匀网格中的某个格点 (x_0, y_0, z_0) ，计算其到某个定点 (x_1, y_1, z_1) （该定点不一定正好在网格坐标上）的径向距离 $r = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$
 - b. 通过计算出来的r值，从给定的一维径向分布函数中插值出数值： $f_l(r)$

c. 推荐的插值方法是三次样条插值，也可用其它插值方法

6. **读入格点上的函数** $V(x, y, z)$ ：程序将从输入文件中读取一个三维函数 $V(x, y, z)$ ，一共有 `ngrid` 个值，类型为double，格点数 $ngrid = nx \times ny \times nz$ 。

7. **计算积分**：对于两个定点A和B，选取他们的函数 $f_l(r)$ ，分别记为 $f_a(r)$ 和 $f_b(r)$ ，计算积分（可以通过离散化的方式进行积分） $\int f_a(r)V(r)f_b(r) dr^3 \approx \sum_n f_{an}V_n f_{bn} dv$ 。通过该积分得到实对称矩阵的矩阵元 h_{ab} ，例如两个定点，就会计算得到4个矩阵元，组成2*2的实对称矩阵。



提示：

1. `i=j` 的时候，也需要计算。
2. 对于 $f_i(r) = 0$ 的格点，事实上不需要再计算其积分项 $f_{in}V_n f_{jn}$ （因为等于0），可以减少计算量，提高计算效率，但如果由于特殊算法或者并行需求，也可以算。
3. 因为是实对称矩阵，事实上只需要算上三角阵，或者下三角阵。

8. **构造矩阵**：遍历完所有定点的配对，得到一个实数对称矩阵计算出实对称矩阵 `H`

9. **求特征对**：将实对称矩阵 `H` 对角化，求出所有特征值和特征向量。



可以调用Lapack，也可以自己写对角化求解器。（必须实现一种对角化方式）

如果你写的对角化求解器，是Fortran的程序，可以学习C++/Fortran混合编译。

10. **输出结果**：输出特征值和特征向量到 `eigenvalues.log` 和 `eigenvectors.log` 文件中。

输入文件解析：

用户输入文件 `INPUT.txt`



用户输入文件 `INPUT.txt`（用户/开发者提供）：

- a. `isHexahedral`：默认为0

如果选择做附加题四（处理平行六面体），则 `isHexahedral` 还可以取1。

- `isHexahedral` =1，表示程序处理的为并行六面体网格
- `isHexahedral` =0，表示程序处理的为长方体网格

b. `lx`：默认1000

c. `ly`：默认1000

d. `lz`：默认1000

e. `thetaxy`：默认为0（单位为度，例如设成90代表x和y轴垂直）

f. `thetayz`：默认为0（单位为度，例如设成90代表x和y轴垂直）

g. `thetaxz`：默认为0（单位为度，例如设成90代表x和y轴垂直）

- `isHexahedral` =0，表示程序处理的为长方体网格，不需要 `thetaxy`、`thetayz`、`thetaxz`，因此默认为0。
- `isHexahedral` =1，表示程序处理的为并行六面体网格，`thetaxy`、`thetayz`、`thetaxz`，不能为0。

h. `support_SH`：默认为0

如果选择做附加题三，则 `support_SH` 还可以取1。

- `support_SH` =1，表示支持球谐函数操作
- `support_SH` =0，表示不支持球谐函数操作

i. `diago_lib`：默认为 `lapack`

不考虑附加题二的话，`diago_lib` 取 `lapack` 或者 `mylib`（仅需支持 `lapack` 或者 `mylib` 其中一种即可）。

- `diago_lib` = `lapack`，对角化实对称矩阵的函数是调用Lapack库实现。
- `diago_lib` = `mylib`，对角化实对称矩阵的函数是自己写的。

如果选择做附加二，则 `diago_lib` 还可以取 `scalapack`。

- `diago_lib` = `scalapack`，对角化实对称矩阵的函数是支持并行求解的，调用ScaLAPACK库实现。

j. `support_Periodic_Boundary`：默认为0

如果选择做附加题六的话，`support_Periodic_Boundary` 还可以取1。

- `support_Periodic_Boundary` =1，表示考虑周期性边界条件
- `support_Periodic_Boundary` =0，表示不考虑周期性边界条件

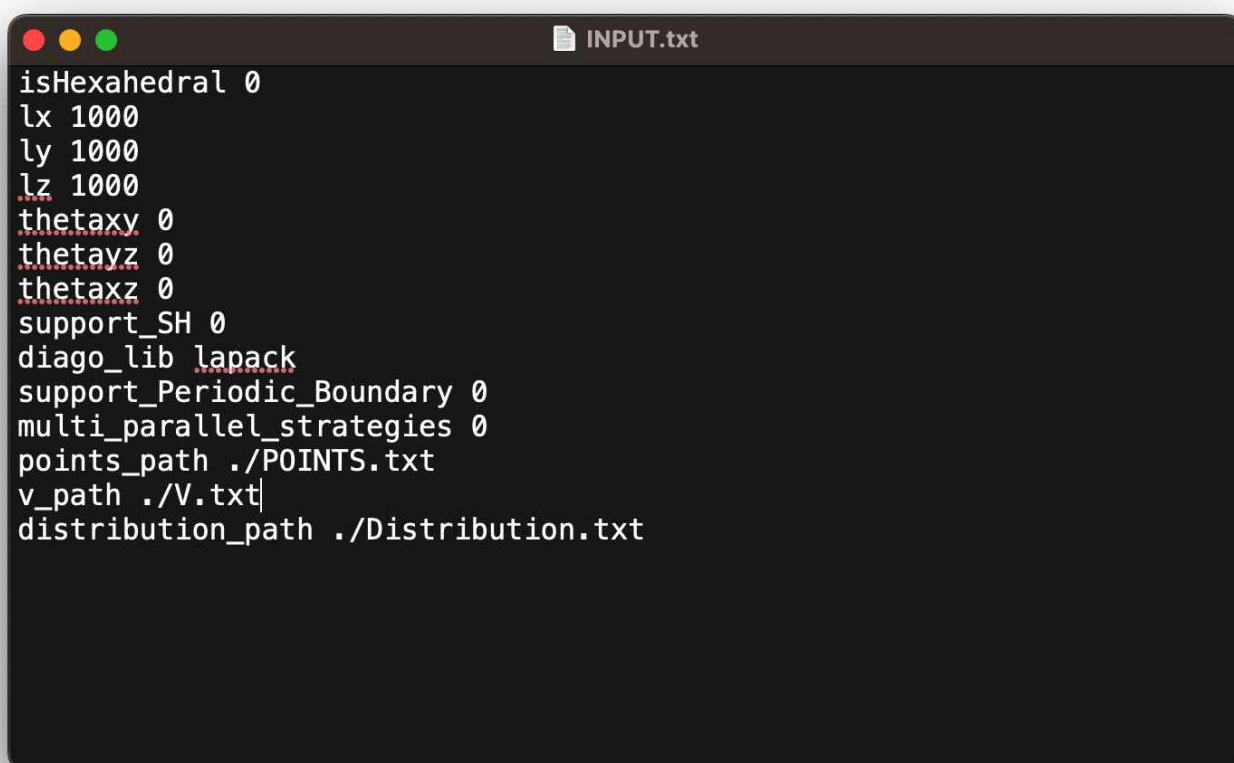
k. `multi_parallel_strategies`：默认为0

如果选择做附加题五的话，`multi_parallel_strategies` 的取值需要你自己命名你所支持的并行划分策略（例如：行、列、块），并在文档中进行解释说明。

`multi_parallel_strategies = 0`，表示程序只支持默认的一种并行划分策略。

- l. `points_path`：定坐标点文件路径
- m. `v_path`：格点上的函数文件路径
- n. `distribution_path`：分布函数文件路径

作业基本要求对应的 用户输入文件 `INPUT.txt` 示例：（不考虑附加题）



```
isHexahedral 0
lx 1000
ly 1000
lz 1000
thetaxy 0
thetayz 0
thetaxz 0
support_SH 0
diago_lib lapack
support_Periodic_Boundary 0
multi_parallel_strategies 0
points_path ./POINTS.txt
v_path ./V.txt
distribution_path ./Distribution.txt
```

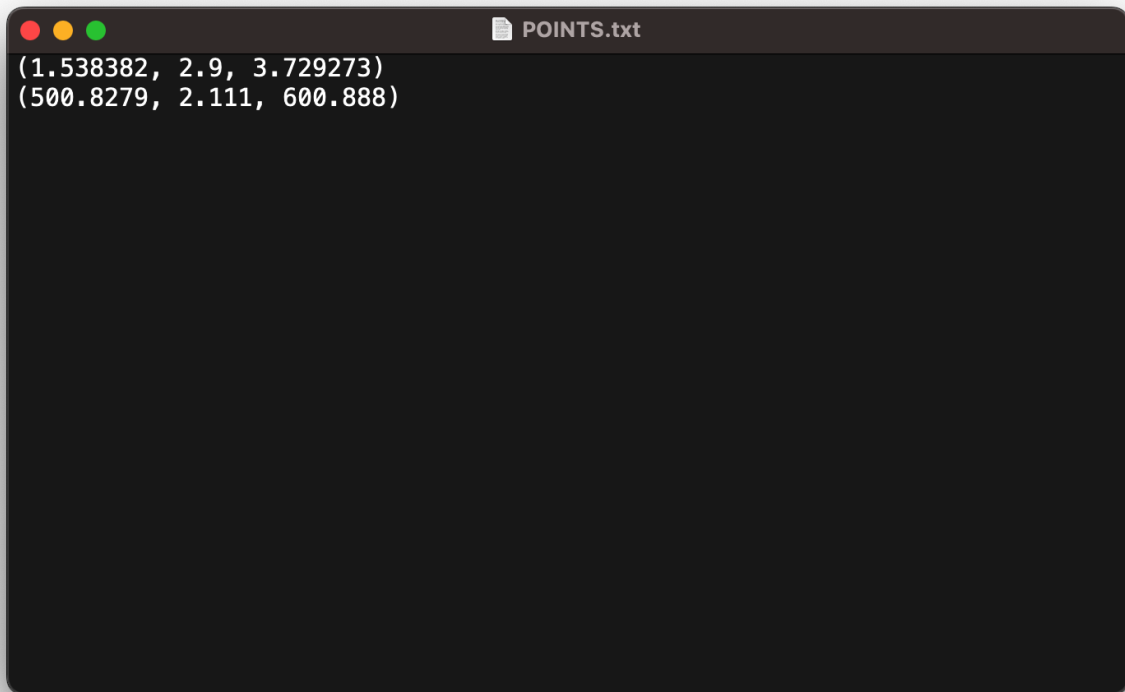
定坐标点文件 `POINTS.txt`

⚽ 定坐标点文件 `POINTS.txt`（用户/开发者提供）：

- 默认是2个坐标（笛卡尔坐标）
- 每个坐标的书写格式：例如： `(1, 2, 3)`
- 每个坐标单独占一行

- 如果完成了附加题二，该文件则包含 n 个定坐标点

定坐标点文件 `POINTS.txt` 示例：



```
(1.538382, 2.9, 3.729273)
(500.8279, 2.111, 600.888)
```

格点上的函数文件 `V.txt`

👍 格点上的函数文件 `V.txt` （助教提供）：

- `nx`、`ny` 和 `nz`
- 长方体网格中每个点的函数值 `v`
 - 函数值的个数为：总格点数 $n_{grid} = nx \times ny \times nz$
 - 每个函数值之间用空格 相隔
 - 每行的第一个函数值前面也有一个空格
 - 格点上的函数值，`z` 坐标变化最快，接下来依次是 `y`、`x`，6个数据一行，每走完 `z` 的一次循环换一行。例如下图中，走完 `z` 的一次循环对应的格点数为50个格点数。

格点上的函数文件 `V.txt` 示例：

有空格

```
V.txt
nx 50
ny 50
nz 50
V:
-2.066e+01 -1.980e+01 -1.749e+01 -1.421e+01 -1.056e+01 -7.304e+00
-4.884e+00 -3.350e+00 -2.316e+00 -1.590e+00 -1.007e+00 -7.187e-01
-5.492e-01 -3.300e-01 -1.691e-01 -2.463e-02 6.363e-02 1.635e-01
2.230e-01 2.815e-01 3.160e-01 3.420e-01 3.645e-01 3.714e-01
3.808e-01 3.740e-01 3.808e-01 3.714e-01 3.645e-01 3.420e-01
3.160e-01 2.815e-01 2.230e-01 1.635e-01 6.363e-02 -2.463e-02
-1.691e-01 -3.300e-01 -5.492e-01 -7.187e-01 -1.007e+00 -1.590e+00
-2.316e+00 -3.350e+00 -4.884e+00 -7.304e+00 -1.056e+01 -1.421e+01
-1.749e+01 -1.980e+01
-1.980e+01 -1.900e+01 -1.679e+01 -1.362e+01 -1.013e+01 -7.029e+00
-4.732e+00 -3.267e+00 -2.266e+00 -1.559e+00 -9.821e-01 -7.057e-01
-5.429e-01 -3.220e-01 -1.673e-01 -2.226e-02 7.084e-02 1.628e-01
2.197e-01 2.743e-01 3.108e-01 3.394e-01 3.614e-01 3.711e-01
3.832e-01 3.808e-01 3.832e-01 3.711e-01 3.614e-01 3.394e-01
3.108e-01 2.743e-01 2.197e-01 1.628e-01 7.084e-02 -2.226e-02
-1.673e-01 -3.220e-01 -5.429e-01 -7.057e-01 -9.821e-01 -1.559e+00
-2.266e+00 -3.267e+00 -4.732e+00 -7.029e+00 -1.013e+01 -1.362e+01
-1.679e+01 -1.900e+01
```

分布函数文件 Distribution.txt



分布函数文件 Distribution.txt (助教提供) :

- 截断半径 `cutoff`
- 粒度 `dr`
- 径向分布函数 `f` 的数值个数 `mesh`
 - 在该文件中径向分布函数 `f` 包含的数值个数为: `cutoff / dr + 1`
 - 为什么加 1? 还有距离为 0 处的分布函数值
- 定点对应的分布函数个数 `1` (注意: 所有定点对应相同数量的分布函数个数)
- 径向分布函数 `f`
 - 个数为 `mesh`

分布函数文件 Distribution.txt 示例:


```
Distribution.txt
cutoff 80
dr 0.1
mesh 801
l 1
f:
4.059785035997653, 4.719276638648823, -9.366156861899658,
-2.743825809211142, 2.1927585989548106, -9.496775317781024,
4.1411614994444825, 6.33129413347168, -4.106651524251031,
3.4957203405001085, -0.4267858953635564, 12.93140660387067,
5.077070208116361, -5.705568516027717, -3.184288586257777,
14.987541999942188, 0.7886894667217543, 0.4105164584551666,
3.9226761901290654, 11.29075176489113, 3.4804408575753207,
8.658103221406488, -6.6131980079007135, -6.690788129169505,
-5.7581665301286025, 8.58029207723245, -7.062380000003415,
-7.194441501941107, 10.156075422675563, -5.2143442582952195,
1.8204739656024174, -9.498328279238097, 2.6324714787616763,
-0.771554792578657, 6.654922456032317, -7.3449149114755805,
-5.911741110045934, -0.5947879973881687, 11.393025761793332,
12.223116872005104, 14.634076133488598, 6.381571898643038,
-8.338274113923351, 2.259742688853029, 12.755886830721249,
-9.794554688038456, 10.518757926703994, -8.3872809557978,
4.6012437869415255, 8.236783448342784, -5.685459193418024,
```