并行计算实验报告 – 1

# 运行命令

python Random\_matrix.py 10 10

./Reduction 10 10 –t –p –d

mpiexec –n 5 ./Reduction\_mpi 10 10 –t –p –d

./Validation\_mpi -p

# 原算法思路

* 原算法接收最多5个参数：前两个为M和N表示矩阵的行数和列数，然后是-t，打印运行时间；-p打印结果矩阵；-d将结果矩阵分别存入文件。
* 原算法第43到52行为U\_t、Alphas、Betas、Gammas各申请了N\*N的空间，但65到70行的for循环中将数据读入到M\*N的矩阵中，而后面92到94行又将计算结果存入矩阵，访问的范围是M\*M（计算过程也是从M\*M的矩阵中获取数据）。因此，当M>N时原始算法不能正常运行，所以进行实验时假设原始算法正确并且保证M<N，在程序中进行检测，并在不满足时打印错误信息。
* 核心算法代码是78到97行的三层循环结构，且核心语句使用了全部三个循环变量i、j、k。

# 并行改造思路

核心思想就是将原始算法第78行的循环拆解，将对M行的计算工作平均分配给其他进程，每一进程需要计算的行数=(M + size – 1) / size。

在计算之前，主进程（MASTER进程，rank=0）利用MPI Broadcast将输入矩阵广播给其他所有进程，为了方便进程间数据传输，将所有矩阵声明为一维数组，即将点U\_t（i，j）映射到U\_t‘（i \* N + j）。另外，主进程开始计算任务之前先要使用异步接收函数MPI\_Irecv设置接收监听，以便其他进程计算完成之后能及时将数据回传给主进程汇总。

计算时，每个进程只计算rank \* rowPerRank到(rank + 1) \* rowPerRank之间的行的数据。

计算之后将自己计算过的行的数据回传给主进程。同时主进程要等待所有进程的数据都回传之后才会进行下一步操作（结束计时，输出结果等）。

# 运行结果

