Метод Жордана нахождения обратной матрицы с выбором главного элемента по строке

Попов Илья

14 февраля 2025 г.

1 Блочный вариант нахождения обратной матрицы методом Жордана с поиском главного элемента по строке, параллельный MPI

Постановка задачи

Находим матрицу обратную к данной

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Пусть m - размер блока, тогда поделим n - размер матрицы на m с остатком $n = m * k + l$ тогда матрицу можно представить в виде:

$$A = \begin{pmatrix} A_{11}^{m\times m} & A_{12}^{m\times m} & \cdots & A_{1,k}^{m\times m} & A_{1,k+1}^{m\times l} \\ A_{21}^{m\times m} & A_{22}^{m\times m} & \cdots & A_{2,k}^{m\times m} & A_{2,k+1}^{m\times l} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{k,1}^{m\times m} & A_{k,2}^{m\times m} & \cdots & A_{k,k}^{m\times m} & A_{k,k+1}^{m\times l} \\ A_{k+1,1}^{l\times m} & A_{k+1,2}^{l\times m} & \cdots & A_{k+1,k}^{l\times m} & A_{k+1,k+1}^{l\times l} \end{pmatrix}$$

1.1 Разеделение данных на свои и чужие

Каждый процесс получает блочные строки матрицы с номерами $num + p * counter$, где num это номер потока, p это общее число потоков и $num + p * counter <= (k+1)$, $counter$ это натуральные числа и 0
В присоеденненной матрице разделение на потоки происходит в точности таким же образом
Каждый процесс имеет доступ к своим данным и не имеет доступа к чужим

1.1.1 Локальная и глобальная нумерации

```
j_glob = j_loc
int local_to_global(int m, int p, int k, int i_loc){
int i_loc_m = i_loc/m;
int i_glob_m = i_loc_m * p + k;
return i_glob_m * m + i_loc%m;
}
int global_to_local(int m, int p, int i_glob){
```

int i_glob_m = i_glob/m; int i_loc_m = i_glob_m/p; k = i_glob_m%p; return i_loc_m * m + i_glob%m; }
Хоть хранение матрицы и блочное но эти формулы нужны в случае заполнения матрицы по формуле и в этом случае они верны

1.2 Формулы в локальной нумерации

Шаг с номером h, где $h <= k$: (то есть h это номера блочных строк в глобальной нумерации)

1. Если блочная строчка с номером h приадлежит потоку с номером num $(num = h\%p)$ то этот поток рассылает часть это блочной строчки остальным функцией $MPI\_Scatter$, а именно : если $begin$ это указатель на начало блочной строчки, то в качестве $sendbuf$ в функцию подается $begin + h*m*m$, размер отпарвляемых данных : $(k+1-h+p-1)/p$ таким образом каждый процесс получит часть текущей блочной строки. В присоеденненной матрице рассылается между процессами первые $(h/p)*p$ блоков той же функцией $MPI\_Scatter$, размер отпрваляемых данных $(h+1+p-1)/p$

2. Каждый процесс получил q блоков исходной матрицы в свой буффер и среди них ищет блок с наименьшей нормой обратного, а именно :
среди блоков $Buf1_{0,g}, g = p, p <= q$

3. С момощю функции $MPI\_Allreduce$ потоки находят блок с наименьшей нормой обратного среди всех блоков данной строки - блок с номером w (номера столбцов совпадают в локальной и глобальной нумерации) если такого блока нет, то алгоритм не применим

4. Каждый поток в своих строчках меняет местами k и w столбцы

5. MPI_Bcast блока с наименьшей нормой всем остальным от потока который его нашел - блок S. Размер отправляемых данных - $m*m$

6. Каждй процесс домножает свои блоки в буффере на S^{-1} слева
$Buf1_{0,g}, g = p, p <= q$
$Buf2_{0,g}, g = p, p <= (h+1+p-1)/p)$
, где Buf2 - буффер от присоеденненной матрицы

7. Функцией $MPI\_Allgather$ все процессы собирают блочную строчку исходной матрицы и меняют местами блоки w и h. Размер отпрваляемых данных - блочная строка из $k-h+1$ блоков. Функцией $MPI\_Allgather$ все процессы собирают блочную строчку присоеденнёной матрицыю Размер отправляемых данных - блочная строка из $h+1$ блоков.

8. Каждый процесс вычитает из своих строчек строчку из буффера в исходной и присоеденнёной матрице
домноженную слева на ведущий блок этой строки:
$$A_{i,j} = A_{i,j} - Buf1_{0,j} * A_{i,h}, \quad i < rows, i >= 0, i! = h\%p \quad j = h+1, .., k+1$$
$$B_{i,j} = B_{i,j} - A_{i,h} * Buf2_{0,j}, \quad i < rows, i >= 0, i! = h\%p \quad j = 1, ..., h$$
, где $rows$ - это число блочных строчек на данный поток

На шаге с номером $k+1$ ищем обратный блок к блоку $A_{k+1,k+1}$ (глобальной) -> $A_{rows-1,rows-1}$ (локально) $= S$ в процессе с номером $num = (k+1)\%p$, где $rows$ - число блочных строк в процессе num
Если блок не обратим то алгоритм не применим для данного m.
MPI_Bcast S^{-1} всем остальным процессам
Проводим выше описанный алгоритм только для присоеденненной матрицы и только для последней блочной строчки

Так как в матрице A мы меняли мастами стобцы для нахождения главного элемента, то в матрице присоеденённой B мы меняем мастими соответствующие строки.

1.3 Точки коммуникации

Были явно описаны в ходе описания алгоритма

1.4 Формула сложности

1.4.1 Число обменов

$$C(n,m,p) = 6 * \frac{n}{m}$$

1.4.2 Объем обменов

$$V = m^2 * \frac{n}{m} + 2 * \frac{1}{p} * \frac{n}{m} * \frac{p*m*(p*m-1)}{2} +$$

$$2 * \sum_{i=1}^{\frac{n}{p*m}} p*i*m^2$$

$$+n^2 = n*m + n*m*p - n + m*n + \frac{n^2}{p} = 2*n*m + n*m*p - n + \frac{n^2}{p} + n^2$$

$$V(n,m,p) = n^2 * (1 + \frac{1}{p}) + 2*n*m + n*m*p - n$$

1.4.3 Сложность

Для рассчета формулы сложности считаем, что $n\%m == 0, \frac{n}{m}\%p == 0$
$n = q*m,\ p$ - число потоков

1.4.4 Для исходной матрицы

$$\sum_{k=1}^{q} \lceil \frac{k}{p} \rceil * (2*m^3 - \frac{m^2}{2} - \frac{m}{2}) + \sum_{k=1}^{q-1} \lceil \frac{q-k}{p} \rceil * (2*m^3 - m^2) + \sum_{k=1}^{q-1} (q-k) * 2*m^3 * (\lceil \frac{q-1}{p} \rceil)$$

Так как $\sum_{i=1}^{q} \lceil \frac{i}{p} \rceil = \sum_{i=1}^{p} 1 + \sum_{i=p+1}^{2*p} 2 + ... = p * \sum_{i=1}^{\frac{n}{m*p}} 1 = \frac{q^2}{2*p} + \frac{q}{2}$
То получаем, что

$$(2*m^3 - \frac{m^2}{2} - \frac{m}{2}) * (\frac{q^2}{2*p} + \frac{q}{2}) + (2*m^3 - m^2) * (\frac{q^2}{2*p} - \frac{q}{2}) + m^3 * (q-1)^2 * \frac{q}{p} =$$

$$= \frac{n^3}{p} + \frac{n*m^2}{p} + O(n^2 + n*m + m^2)$$

1.4.5 Для присоеденнёной

$$\sum_{k=1}^{q}(\lceil\frac{k-1}{p}\rceil)*(2*m^3-m^2)+\lceil\frac{q-1}{p}\rceil*\sum_{k=1}^{q}k*2*m^3=$$

$$=(2*m^3-m^2)*(\frac{q^2}{2*p}-\frac{q}{2})+2*m^3*(q^2-1)*\frac{q}{p}=\frac{n^3}{p}+\frac{n^2*m}{p}-2*\frac{n*m2}{p}+O(n^2+n*m+m^2)$$

Суммируя получаем итогувую сложность :

$$S_p(n,m,p)=2*\frac{n^3}{p}+\frac{n^2*m}{p}-\frac{n*m^2}{p}+O(n^2+n*m+m^2)$$
$$S_p(n,m,1)=S(n,m)$$