Метод Жордана с выбором главного элемента по столбцу.

Гвоздев Михаил

15 февраля 2020 г.

1 Алгоритм блочный

Задаётся размер блока m, матрица A теперь имеет вид:

$$\begin{pmatrix} A_{0,0}^{m\times m} & A_{0,1}^{m\times m} & A_{0,2}^{m\times m} & \dots & A_{0,k}^{m\times p} \\ A_{1,0}^{m\times m} & A_{1,1}^{m\times m} & A_{1,2}^{m\times m} & \dots & A_{1,k}^{m\times p} \\ . & . & . & \ddots & . \\ A_{k,0}^{p\times m} & A_{k,1}^{p\times m} & A_{k,2}^{p\times m} & \dots & A_{k,k}^{p\times p} \end{pmatrix}$$

где $n = k * m + p$.

Вектор свободных членов B тоже делится на блоки по m элементов, последний блок будет состоять из p элементов.

Норму матрицы возьмём

$$||A|| = \max_{i=0,\dots,n-1} \sum_{j=0}^{n-1} |a_{i,j}|$$

Также заведем массив длины k для восстановления положения блоков вектора B после всех операций.

Искать будем блок норма обратного которого минимальна (это уменьшает погрешность при вычислениях).

0-й шаг алгоритма.
Для блоков 0-ого столбца с номерами $< k$, найдем обратные матрицы. Если ни одна из них не посчиталась, то метод неприменим. Иначе переставляем строку, содержащую обратный блок с минимальной нормой, с нулевой строкой и заносим соответственное изменение в массив индексов. Далее:

$A_{0,i}^{m\times m} = (A_{0,0}^{-1})^{m\times m} * A_{0,i}^{m\times m}, i = 1, \dots, k-1$
$A_{0,k}^{m\times p} = (A_{0,0}^{-1})^{m\times m} * A_{0,k}^{m\times p}$
$B_0^{m\times 1} = (A_{0,0}^{-1})^{m\times m} * B_0^{m\times 1}$
$A_{0,0}^{m\times m} = E^{m\times m}$

Получим:

$$\begin{pmatrix} E & (A_{0,1}^{m\times m})^{(1)} & \dots & (A_{0,k}^{m\times p})^{(1)} \\ A_{1,0}^{m\times m} & A_{1,1}^{m\times m} & \dots & A_{1,k}^{m\times p} \\ . & . & \ddots & . \\ A_{k,0}^{p\times m} & A_{k,1}^{p\times m} & \dots & A_{k,k}^{p\times p} \end{pmatrix}$$

Продолжение $0-$ого шага. Вычтем из каждой строки $0-$вую, умноженную слева на $0-$вой элемент данной строки:

$A_{i,j}^{m\times m} = A_{i,j}^{m\times m} - A_{0,j}^{m\times m} * A_{i,0}^{m\times m}, i = 1, \dots, k-1, \quad j = 1, \dots, k-1$
$A_{i,k}^{m\times p} = A_{i,k}^{m\times p} - A_{i,0}^{m\times m} * A_{0,k}^{m\times p}, i = 1, \dots, k-1$
$A_{k,i}^{p\times m} = A_{k,i}^{p\times m} - A_{k,0}^{p\times m} * A_{0,i}^{m\times m}, i = 1, \dots, k-1$
$A_{k,k}^{p\times p} = A_{k,k}^{p\times p} - A_{k,0}^{p\times m} * A_{0,k}^{m\times p}$
$B_i^{m\times 1} = B_i^{m\times 1} - A_{i,0}^{m\times m} * B_0^{m\times 1}, i = 1, \dots, k-1$

$B_k^{p\times 1} = B_k^{p\times 1} - A_{k,0}^{p\times m} * B_0^{m\times 1}$
$A_{i,0}^{m\times m} = 0^{m\times m}, i = 1, \ldots, k-1$
$A_{k,0}^{p\times m} = 0^{p\times m}$
Получим :

$$\begin{pmatrix} E & (A_{0,1}^{m\times m})^{(1)} & (A_{0,2}^{m\times m})^{(1)} & \ldots & (A_{0,k}^{m\times p})^{(1)} \\ 0 & (A_{1,1}^{m\times m})^{(1)} & (A_{1,2}^{m\times m})^{(1)} & \ldots & (A_{1,k}^{m\times p})^{(1)} \\ . & . & . & \ddots & . \\ 0 & (A_{k,1}^{p\times m})^{(1)} & (A_{k,2}^{p\times m})^{(1)} & \ldots & (A_{k,k}^{p\times p})^{(1)} \end{pmatrix}$$

На $(t-1)$-м шаге будем иметь

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \ldots & A_{0,t}^{m\times m} & \ldots & A_{0,k}^{m\times p} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \ldots & A_{1,t}^{m\times m} & \ldots & A_{1,k}^{m\times p} \\ \vdots & \vdots & \ddots & \vdots & \ldots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \ldots & A_{t,t}^{m\times m} & \ldots & A_{t,k}^{m\times p} \\ \vdots & \vdots & \ldots & \vdots & \ddots & \vdots \\ *_{k,0}^{p\times m} & *_{k,1}^{p\times m} & \ldots & A_{k,t}^{p\times m} & \ldots & A_{k,k}^{p\times p} \end{pmatrix}$$

и переход после перемены строк местами и занесения изменений в массив индексов $(t < k)$:

$A_{t,i}^{m\times m} = (A_{t,t}^{-1})^{m\times m} * A_{t,i}^{m\times m}, i = t+1, \ldots, k-1$
$A_{t,k}^{m\times p} = (A_{t,t}^{-1})^{m\times m} * A_{t,k}^{m\times p}$
$B_t^{m\times 1} = (A_{t,t}^{-1})^{m\times m} * B_t^{m\times 1}$
$A_{t,t}^{m\times m} = E^{m\times m}$

Вычтем из каждой строки $t-$ую, умноженную слева на $t-$ый элемент данной строки.

$A_{i,j}^{m\times m} = A_{i,j}^{m\times m} - A_{i,t}^{m\times m} * A_{t,j}^{m\times m}, i = 0, \ldots, k-1, \quad t \neq i$
$j = (t+1), \ldots, k-1$
$A_{i,k}^{m\times p} = A_{i,k}^{m\times p} - A_{i,t}^{m\times m} * A_{t,k}^{m\times p}, i = 0, \ldots, k-1, \quad t \neq i$
$A_{k,i}^{p\times m} = A_{k,i}^{p\times m} - A_{k,t}^{p\times m} * A_{t,i}^{m\times m}, i = 0, \ldots, k-1, \quad t \neq i$
$A_{k,k}^{p\times p} = A_{k,k}^{p\times p} - A_{k,t}^{p\times m} * A_{t,k}^{m\times p}$
$B_i^{m\times 1} = B_i^{m\times 1} - A_{i,t}^{m\times m} * B_t^{m\times 1}, i = 0, \ldots, k-1, \quad t \neq i$
$B_k^{p\times 1} = B_k^{p\times 1} - A_{k,t}^{p\times m} * B_t^{m\times 1}$
$A_{i,t}^{m\times m} = 0^{m\times m}, i = 0, \ldots, k-1, \quad t \neq i$
$A_{k,t}^{p\times m} = 0^{p\times m}$
На $(k-1)$-м шаге получим:

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \ldots & *_{0,t}^{m\times m} & \ldots & A_{0,k}^{m\times p} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \ldots & *_{1,t}^{m\times m} & \ldots & A_{1,k}^{m\times p} \\ \vdots & \vdots & \ddots & \vdots & \ldots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \ldots & *_{t,t}^{m\times m} & \ldots & A_{t,k}^{m\times p} \\ \vdots & \vdots & \ldots & \vdots & \ddots & \vdots \\ *_{k,0}^{p\times m} & *_{k,1}^{p\times m} & \ldots & *_{k,t}^{p\times m} & \ldots & A_{k,k}^{p\times p} \end{pmatrix}$$

где $*$ либо 0, либо E (почти).

Переход (строки местами не меняем, т.к. квадратная матрица только в углу) $(t == k)$:

$B_k^{p\times 1} = (A_{k,k}^{-1})^{p\times p} * B_k^{p\times 1}, i = 0, \ldots, k-1, \quad k \neq i$
$A_{k,k}^{p\times p} = E_{k,k}^{p\times p}$

Вычтем из каждой строки $k-$ую, умноженную слева на $k-$ый элемент данной строки.

$B_i^{m\times 1} = B_i^{m\times 1} - A_{i,k}^{m\times p} * B_k^{m\times 1}, i = 0, \ldots, k-1, \quad k \neq i$
$A_{i,k}^{m\times p} = 0^{m\times p}, i = 0, \ldots, k-1, \quad k \neq i$

Получим вектор B с перемешанными координатами, которые приводятся в правильный порядок при помощи массива индексов, который мы завели в самом начале.

1.1 Описание основных функций работы с блоками

```
void get_block(int i, int j, int width, int hight, int real_size_c, int n, double *c, doub
/*i,jкоординаты верхнего левого угла блока с в матрице а
width, hight ширина и высота блокаа с
n длина стороны матрицы а */
{
    int p=0;
    p=i+j;

    for(int t=0;t<hight;t++)
    {
        for( int r = 0 ; r < width ; r ++ )
        {
            c[ t * real_size_c + r ] = a[ p + r ];
        }

        p+=n;
    }
}

void put_block(int i, int j, int width, int hight, int real_size_c , int n, double *c, dou
{
    int p;
    p=i+j;

    for(int t=0;t<hight;t++)
    {
        for(int r=0;r<width;r++)
        {
            a[p+r]=c[t*real_size_c+r];
        }

        p+=n;
    }
}
```

1.2 Сложность

1.На нахождение обратной матрицы к $m\times m$ методом Жордана нужно $(2m-i-1)$ умножение на обратный элемент, $(m-1)(2m-i-1)$ умножение на соответсвующий элемент каждой строки и столько же вычитаний. Всего:

$$\sum_{i=0}^{m}((2m-i-1)+2(m-1)(2m-i-1))=\frac{(2m-1)(3m-1)m}{2}=3m^3+O(m^2)$$

арифметических операций.

2. На умножение матрицы $m\times m$ на матрицу $m\times p$ требуется $2m*pm=2pm^2$ арифметических операций (так как чтобы получить 1 элемент итоговой матрицы $m\times p$ нужно провести m умножений и столько же сложений).
3. Разность матриц $m\times m$ имеет сложность m^2.
4. Итоговая сложность для $n=km+p$ равносильна сложности $n=km$ при $(n\longrightarrow\infty)$. Тогда

$$\sum_{j=0}^{k-1}(k-j)(\frac{(2m-1)(3m-1)m}{2}+(k-1)(2m^3+m^2))=$$

$$=\frac{k(k+1)}{2}(\frac{(2m-1)(3m-1)m}{2}+(k-1)(2m^3+m^2))=$$

$$=n^3(1+\frac{1}{2m})+n^2(\frac{3}{2}m-\frac{5}{4}+\frac{1}{4m})+n(\frac{1}{2}m^2-\frac{1}{2}m-1)=$$

$$=n^3+O(n^2)$$

при $(n\longrightarrow\infty)$

1.3 Частные случаи

$m=1$

$$S(n,1)=\frac{3}{2}n^3+O(n^2)$$

при $(n\longrightarrow\infty)$

$m=n$

$$S(n,n)=n^3(1+\frac{1}{2n})+n^2(\frac{3}{2}n-\frac{5}{4}+\frac{1}{4n})+n(\frac{1}{2}n^2-\frac{1}{2}n-1)=$$

$$=n^3(1+\frac{3}{2}+\frac{1}{2})+n^2(\frac{1}{2}-\frac{5}{4}-\frac{1}{2})+n(\frac{1}{4}-1)=$$

$$3n^3-\frac{5}{4}n^2-\frac{3}{4}n=3n^3+O(n^2)$$

при $(n\longrightarrow\infty)$

2 Алгоритм блочный. Параллельный

Задаётся размер блока m, матрица A теперь имеет вид:

$$\begin{pmatrix} A_{0,0}^{m\times m} & A_{0,1}^{m\times m} & A_{0,2}^{m\times m} & \dots & A_{0,k}^{m\times p} \\ A_{1,0}^{m\times m} & A_{1,1}^{m\times m} & A_{1,2}^{m\times m} & \dots & A_{1,k}^{m\times p} \\ . & . & . & \ddots & . \\ A_{k,0}^{p\times m} & A_{k,1}^{p\times m} & A_{k,2}^{p\times m} & \dots & A_{k,k}^{p\times p} \end{pmatrix}$$

где $n = k * m + p$.

Вектор свободных членов B тоже делится на блоки по m элементов, последний блок будет состоять из p элементов.

Норму матрицы возьмём

$$||A|| = \max_{i=0,\dots,n-1} \sum_{j=0}^{n-1} |a_{i,j}|$$

Также заведем массив длины k для восстановления положения блоков вектора B после всех операций.

Искать будем блок норма обратного которого минимальна (это уменьшает время вычислений).

2.1 Разделение данных между потоками

Пусть у нас f потоков, тогда поделим работу и память меж ними построчно так ($i = 0, \dots, k - j - f * i \geq 0$) :
0-й (родительский) работает с $k - f * i$-й строками блоков матрицы A и вектора B, также он заносит изменения в массив индексов

j-й работает с $k - j - f * i$ строками блоков матрицы A и вектора B $j = 1, \dots, t - 1$

На t-м шаге будем иметь(начали с 0го шага)

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \dots & A_{0,t}^{m\times m} & \dots & A_{0,k}^{m\times p} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \dots & A_{1,t}^{m\times m} & \dots & A_{1,k}^{m\times p} \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \dots & A_{t,t}^{m\times m} & \dots & A_{t,k}^{m\times p} \\ \vdots & \vdots & \dots & \vdots & \ddots & \vdots \\ *_{k,0}^{p\times m} & *_{k,1}^{p\times m} & \dots & A_{k,t}^{p\times m} & \dots & A_{k,k}^{p\times p} \end{pmatrix}$$

Для блоков t-ого столбца, найдем обратные матрицы. Каждый поток вычисляет их для своих строк. Потоки выбирают наименьшую норму из обратившихся матриц и обмениваются ее значением и номером строки в *точке синхронизации*. Если все матрицы вырождены, то метод неприменим. Иначе 0й поток выбирает наименьшую норму из переданных и заносит изменение в массив индексов. Далее:
t-й столбец A менять не будем. Поток r ,которому принадлежит $(A_{t,t}^{-1})^{m\times m}$, делает:

$A_{t,i}^{m\times m} = (A_{t,t}^{-1})^{m\times m} * A_{t,i}^{m\times m}, i = t+1, \dots, k-1$
$A_{t,k}^{m\times p} = (A_{t,t}^{-1})^{m\times m} * A_{t,k}^{m\times p}$
$B_{t}^{m\times 1} = (A_{t,t}^{-1})^{m\times m} * B_{t}^{m\times 1}$

Все потоки вычитают из своих строк $t-$ую, умноженную слева на $t-$ый элемент данной строки. Каждый поток работает со своими строками. Общие формулы для r-того потока будут выглядеть:

$A_{i,j}^{m\times m} = A_{i,j}^{m\times m} - A_{i,t}^{m\times m} * A_{t,j}^{m\times m}$
$B_{i}^{m\times 1} = B_{i}^{m\times 1} - A_{i,t}^{m\times m} * B_{t}^{m\times 1}$
$i = k - r - fi(\geq 0), \dots, k-2r, k-r, \quad t \neq i, \quad j = (t+1), \dots, k-1$

Если $p \neq 0$ для $0-$го потока верны формулы

$A_{i,k}^{m\times p} = A_{i,k}^{m\times p} - A_{i,t}^{m\times m} * A_{t,k}^{m\times p}$
$A_{k,i}^{p\times m} = A_{k,i}^{p\times m} - A_{k,t}^{p\times m} * A_{t,i}^{m\times m}$
$A_{k,k}^{p\times p} = A_{k,k}^{p\times p} - A_{k,t}^{p\times m} * A_{t,k}^{m\times p}$
$B_{k}^{p\times 1} = B_{k}^{p\times 1} - A_{k,t}^{p\times m} * B_{t}^{m\times 1}$

Точка синхронизации, сохранение изменений, конец шага алгоритма

На k-м шаге получим:

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \dots & *_{0,t}^{m\times m} & \dots & A_{0,k}^{m\times p} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \dots & *_{1,t}^{m\times m} & \dots & A_{1,k}^{m\times p} \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \dots & *_{t,t}^{m\times m} & \dots & A_{t,k}^{m\times p} \\ \vdots & \vdots & \dots & \vdots & \ddots & \vdots \\ *_{k,0}^{p\times m} & *_{k,1}^{p\times m} & \dots & *_{k,t}^{p\times m} & \dots & A_{k,k}^{p\times p} \end{pmatrix}$$

где $*$ исходный блок матрицы или преобразованный в ходе решения СЛУ (почти).

0й поток обратит $A_{k,k}^{p\times p}$ и выполнит формулы:

$B_{k}^{p\times 1} = (A_{k,k}^{-1})^{p\times p} * B_{k}^{p\times 1}$
$B_{i}^{m\times 1} = B_{i}^{m\times 1} - A_{i,k}^{m\times p} * B_{k}^{m\times 1}, i = 0, \dots, k-1, \quad k \neq i$

Получим вектор B с перемешанными координатами, которые приводятся 0м потоком в правильный порядок при помощи массива индексов, который мы завели в самом начале.

2.2 Сложность

1.На нахождение обратной матрицы к $m\times m$ методом Жордана нужно $(2m-i-1)$ умножение на обратный элемент, $(m-1)(2m-i-1)$ умножение на соответсвующий элемент каждой строки и столько же вычитаний. Всего:

$$\sum_{i=0}^{m}((2m-i-1) + 2(m-1)(2m-i-1)) =$$

$$= \frac{(2m-1)(3m-1)m}{2} = 3m^3 + O(m^2)$$

арифметических операций.

2. На умножение матрицы $m \times m$ на матрицу $m \times p$ требуестся $2m * pm = 2pm^2$ арифметических операций (так как чтобы получить 1 элемент итоговой матрицы $m \times p$ нужно провести m умножений и столько же сложений).

3. Разность матриц $m \times m$ имеет сложность m^2.

4. Всего каждому потоку принадлежит $i \leq \frac{k-r}{f}$ строк

5. Итоговая сложность для r-ого потока (кроме дополнительной сложности родительского)

$$\sum_{j=0}^{[\frac{k-r}{f}]} (k-r-fj)(\frac{(2m-1)(3m-1)m}{2} + (k-1)(2m^3+m^2)) =$$

Пусть $r = 0$ и $k - r$ кратно f, для упрощения расчетов

$$= \frac{k(k+f)}{2f}(\frac{(2m-1)(3m-1)m}{2} + (k-1)(2m^3+m^2)) =$$

$$= \frac{n^3}{f}(1+\frac{1}{2m}) + n^2(m(\frac{1}{2f}+1) + \frac{1}{2}(1-\frac{7}{2f}) + \frac{1}{4mf}) + n(\frac{m^2}{2} - \frac{m}{2} - 1) =$$

$$= \frac{n^3}{f} + n^2 m(\frac{1}{2f}+1) + O(n^2)$$

при $(n \longrightarrow \infty)$

2.3 Точки синхронизации

Всего будет k шагов алгоритма в ,которых нужны точки синхронизации, по 2 на каждый шаг. Значит всего понадобится $2k$ точек синхронизации.

3 MPI реализация

Пусть у нас p процессов, матрица размера $n \times n$, разбитая на блоки размера $m \times m$, и $n == k * m + l$ и ветор B размера $n \times 1$, где p,k,n,m,l натуральные $\cup 0$.

3.1 Разделение данных между процессами

Данные между процессами делим по блочным строкам. i-ый процесс будет хранить $i, i + p, \ldots, i + p * j \leq n$ строки блочной матрицы и вектора B. Также 0-й процесс будет хранить последнюю строчку матрицы из блоков размера $l \times m$.

0	0	0	...	0	0	0
1	1	1	...	1	1	1
2	2	2	...	2	2	2
3	3	3	...	3	3	3
...	...	...	...	...	...	...
0	0	0	...	0	0	0
1	1	1	...	1	1	1
2	2	2	...	2	2	2
0	0	0	...	0	0	0

Каждый процесс будет хранить массив index длины k заполненый (-1), в котором будут отображаться позиции j строк на jом шаге алгоритма.

3.2 Шаг алгоритма

На t-м шаге алгоритма нужно найти главный элемент t-го столбца. (Матрица в логическом представлении)

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \ldots & A_{0,t}^{m\times m} & \ldots & A_{0,k}^{m\times l} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \ldots & A_{1,t}^{m\times m} & \ldots & A_{1,k}^{m\times l} \\ \vdots & \vdots & \ddots & \vdots & \ldots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \ldots & A_{t,t}^{m\times m} & \ldots & A_{t,k}^{m\times l} \\ \vdots & \vdots & \ldots & \vdots & \ddots & \vdots \\ *_{k,0}^{l\times m} & *_{k,1}^{l\times m} & \ldots & A_{k,t}^{l\times m} & \ldots & A_{k,k}^{l\times l} \end{pmatrix}$$

Каждый процесс обращает матрицы t-го столбца среди своих строк и находит номер строки с минимальной нормой. Для этого в цикле по массиву `index` при услови `(index[i] == -1 && i % p == j)`, процесс с номером j обращает свой $A_{[i/p],q}$ блок (т.е. относительно своей внутренней нумерации).

В результате каждый прцесс составит пару `(1/norma,ind)`, где norma - наименьшая норма обратного блока для данного процесса, а ind - номер строки в логическом прорядке, в которой достигается norma. Если все блоки процесса вырождены то пара будет `(0,ind)`.

Затем используем функции `MPI_Allreduce` с `MPI_MAXLOC` и получаем в каждом процессе пару с максимумом по первому значению. Они вносят t на место `index[ind]`.

По условию `(ind % p == j )` j -ый процесс с помощью функции `MPI_Bcast` , отправляет строку блоков $A_{[ind/p],d}$ (d = t,...,k) и кусочек B всем процессам (у каждого процесса будет в итоге массив save длины $m \times (n+1)$, n+1 тк нужно место под кусочек пересланного вектора B),а сам j-й процесс работает с исходной строкой матрицы.

Теперь каждый процесс ищет обратную матрицу к ведущему блоку (save[t]) и домножает слева на все блоки полученной строки (save) (в том числе тот, где лежат кусочки B), кроме ведущего, его полагаем E.

В итоге получим матрицу (в логическом представлении) :

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \dots & A_{0,t}^{m\times m} & \dots & A_{0,k}^{m\times l} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \dots & A_{1,t}^{m\times m} & \dots & A_{1,k}^{m\times l} \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \dots & E_{ind,t}^{m\times m} & \dots & A_{ind,k}^{m\times l} \\ \vdots & \vdots & \dots & \vdots & \ddots & \vdots \\ *_{k,0}^{l\times m} & *_{k,1}^{l\times m} & \dots & A_{k,t}^{l\times m} & \dots & A_{k,k}^{l\times l} \end{pmatrix}$$

Далее каждый процесс вычитает полученную строку из остальных своих строк.

```
for(int i = 0; i < k; i++)
{
    if( i%p == my_rank && i!=t)
    {
        for( int j = t+1 ; j < k ; j++)
        {
            a[int(i/p),j] -= a[int(i/p),j]*save[ j ];
        }
        b[int(i/p)] -= a[int(i/p),t]*save[k+1];
        // в save[k+1] лежит пересланный кусочек B
    }
}
for(int i = 0 ; i < k ; i ++ )
{
    if(i!=ind)
    {
        a[int(i/p),t]=0;
    }
    else    a[int(ind/p),t]=1;
}
```

Получим матрицу (в логическом представлении)

$$\begin{pmatrix} *_{0,0}^{m\times m} & *_{0,1}^{m\times m} & \dots & 0_{0,t}^{m\times m} & \dots & A_{0,k}^{m\times l} \\ *_{1,0}^{m\times m} & *_{1,1}^{m\times m} & \dots & 0_{1,t}^{m\times m} & \dots & A_{1,k}^{m\times l} \\ \vdots & \vdots & \ddots & \vdots & \dots & \vdots \\ *_{t,0}^{m\times m} & *_{t,1}^{m\times m} & \dots & E_{ind,t}^{m\times m} & \dots & *A_{ind,k}^{m\times l} \\ \vdots & \vdots & \dots & \vdots & \ddots & \vdots \\ *_{k,0}^{l\times m} & *_{k,1}^{l\times m} & \dots & 0_{k,t}^{l\times m} & \dots & A_{k,k}^{l\times l} \end{pmatrix}$$

После k шагов алгоритма процессы восстанавливают ответ с помощью массива `index` и `MPI_Sendrecv`

3.3 Количество пересылок данных

На каждый шаг алгоритма нужна 1 пересылка на пары (double,int), 1 пересылка строки save и еще 1 пересылка при восстановлении матрицы. В итоге $3k$ пересылок

3.4 Объем пересылаемых данных

Продолжая предыдущий раздел будет 1 пересылка по 12 байт, 1 пересылка $m(n+1)*8$ байт на 1 шаг алгоритма и в худшем случае $\frac{n}{p}*8$ байт. В итоге $k(m(n+1)*8+12+\frac{n}{p}*8) \approx n^2*8$ байт

3.5 Сложность

1.На нахождение обратной матрицы к $m \times m$ методом Жордана нужно $(2m-i-1)$ умножение на обратный элемент, $(m-1)(2m-i-1)$ умножение на соответсвующий элемент каждой строки и столько же вычитаний. Всего:

$$\sum_{i=0}^{m}((2m-i-1)+2(m-1)(2m-i-1))=$$

$$=\frac{(2m-1)(3m-1)m}{2}=3m^3+O(m^2)$$

арифметических операций.

2. На умножение матрицы $m \times m$ на матрицу $m \times l$ требуестся $2m*lm=2lm^2$ арифметических операций (так как чтобы получить 1 элемент итоговой матрицы $m \times l$ нужно провести m умножений и столько же сложений).

3. Разность матриц $m \times m$ имеет сложность m^2.

4. Всего каждому процессу принадлежит $i \le \frac{k-r}{p}$ строк

5. Итоговая сложность для r-ого процесса (кроме дополнительной сложности 0-ого)

$$\sum_{j=0}^{[\frac{k-r}{p}]}(k-r-pj)(\frac{(2m-1)(3m-1)m}{2}+(k-1)(2m^3+m^2))=$$

Пусть $r=0$ и $k-r$ кратно p, для упрощения расчетов

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

$$=\frac{n^3}{p}(1+\frac{1}{2m})+n^2(m(\frac{1}{2p}+1)+\frac{1}{2}(1-\frac{7}{2p})+\frac{1}{4mp})+n(\frac{m^2}{2}-\frac{m}{2}-1)=$$

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

при $(n \longrightarrow \infty)$

3.6 Частные случаи

m=1:

$$\frac{3n^3}{2p}+n^2(\frac{3}{2}-\frac{1}{p})+O(n)$$

m=n:

$$n^3(\frac{3}{2}+\frac{1}{p})+O(n^2)$$

p=1:

$$n^3+O(n^2)$$