Метод Гаусса с выбором главного элемента по строке с использованием MPI

1 Постановка задачи

Рассмотрим систему линейных уравнений, записанную в матричном виде:

$$AX = B,\ A \in M_{n,n}(\mathbb{R}),\ X, B \in \mathbb{R}^n$$

.
Требуется решить ее методом Гаусса с выбором главного элемента по строке.

2 Индексация и переменные

Индексация любых массивов начинается с 0. Столбцы и строки матрицы всегда подразумеваются блочными, если не указано другого. Ниже представлены переменные, использованные для описания алгоритма.

$$A = \begin{pmatrix} A_{00}^{m\times m} & \dots & A_{0,l-1}^{m\times m} & A_{0,l}^{m\times s} \\ \vdots & \ddots & \vdots & \vdots \\ A_{l-1,0}^{m\times m} & \dots & A_{l-1,l-1}^{m\times m} & A_{l-1,l}^{m\times s} \\ A_{l,0}^{s\times m} & \dots & A_{l,l-1}^{s\times m} & A_{ll}^{s\times s} \end{pmatrix}$$

$$X = \begin{pmatrix} X_0^m \\ \vdots \\ X_{l-1}^m \\ X_l^s \end{pmatrix}$$

$$B = \begin{pmatrix} B_0^m \\ \vdots \\ B_{l-1}^m \\ B_l^s \end{pmatrix}$$

n — размер матрицы A.
m — размер стандартного блока.
$l = \left[\frac{n}{m}\right]$.
$s = n - m \cdot l$.
i_main — номер итерации алгоритма Гаусса.
$pivot$ — индекс столбца главного элемента на итерации i_main
p — количество запущенных процессов.
$rank$ — номер-идентификатор процесса.

$$sh0 = \begin{cases} 1, & \text{если } rank < i\_main \ \% \ p \\ 0, & \text{иначе} \end{cases}$$

$$sh1 = \begin{cases} 1, & \text{если } rank \leqslant i\_main \ \% \ p \\ 0, & \text{иначе} \end{cases}$$

3 Хранение

Матрица хранится в памяти процессов по блок-столбцам. А именно, процесс $rank$ хранит все столбцы с номерами $rank + k \cdot p$, где $k \in \mathbb{Z}$. Столбец присоединенной матрицы B хранится в процессе с $rank = 0$.

4 Прямой ход алгоритма Гаусса

4.1 Общее описание

В нижеследующих разделах описывается одна итерация прямого хода алгоритма Гаусса. Его задача - привести матрицу A к верхнетреугольному виду с единицами на диагонали. i_main пробегает все значения от 0 до l.

4.2 Поиск главного элемента

В процессе $rank$ в строке i_main, начиная со столбца $i\_main/p + sh0$, проводится поиск главного элемента. т. е. блока с наименьшей нормой обратной матрицы. Если обратной не существует, норму полагаем бесконечностью. Результат процедуры - структура с номером столбца, содержащего такой блок, и нормой матрицы, обратной к блоку. Если невырожденного блока не найдено, номеру присваивается значение -1.
Затем, вызывается функция MPI_Allreduce, после работы которой, все процессы получают глобальный индекс главного элемента.

4.3 Смена столбцов

Процессы с $rank = i\_main \ \% \ p$ и $rank = pivot \ \% \ p$ меняются столбцами с локальными номерами $i\_main \ / \ p$ и $pivot \ / \ p$ соответственно с помощью функции MPI_Sendrecv_replace. MPI обмена, конечно же, не происходит, если столбцы находятся в одном процессе. Этому преобразованию столбцов соответствует умножение матрицы A справа на матрицу этого преобразования: AI_i. Тогда система принимает вид $AI_iX = B$. Запомним эту перестановку столбцов и в конце алгоритма покажем, как вернуться к системе, равносильной исходной.

4.4 Умножение строки i_main

С помощью функции MPI_Bcast , вызванной в процессе $rank = i\_main \,\%\, p$, процессы получают блок A_{i_main,i_main}. Затем, блоки в строке i_main, начиная со столбца $i\_main \ / \ p \ + \ sh1$ умножаются слева на матрицу $A^{-1}_{i\_main,i\_main}$.
В процессе 0 так же умножается матрица B_{i_main}.

4.5 Обнуление столбца, содержащего главный элемент

С помощью функции MPI_Bcast , вызванной в процессе $rank = i\_main \,\%\, p$, процессы получают часть (со строки $i\_main + 1$) столбца с глобальным индексом i_main. Этот столбец необходим каждому процессу, чтобы произвести вычитание

$$\widetilde{A}_{jh} = A_{jh} - A_{j,i\_main}\widetilde{A}_{i\_main,h},$$

$$\forall\ h \in \{i\_main\ +\ 1, \ldots, l\},\ \forall\ j \in \{i\_main\ +\ 1, \ldots, l\}$$

В процессе $rank$ вычитание проводится со строками строго ниже i_main, со столбцами с индексом от $i\_main\ /\ p + sh1$.
В процессе 0 вычитания так же проводятся со столбцом B.

5 Обратный ход алгоритма Гаусса

5.1 Общее описание

К этому моменту, логически матрица A имеет вид (для простоты $l = 4$):

$$\begin{pmatrix} E & A_{01} & A_{02} & A_{03} & A_{0l} \\ 0 & E & A_{12} & A_{13} & A_{1l} \\ 0 & 0 & E & A_{23} & A_{2l} \\ 0 & 0 & 0 & E & A_{3l} \\ 0 & 0 & 0 & 0 & E \end{pmatrix}$$

В следующем разделе описывается одна итерация обратного хода алгоритма Гаусса. Его задача - логически привести матрицу A к единичной. Фактические вычисления проводятся только со столбцом B в процессе 0. i_main пробегает все значения от l до 0.

5.2 Итерация обратного хода

С помощью функции MPI_Send, процесс с $rank\ =\ i\_main \,\%\, p$ передает процессу 0 часть (со строки i_main до строки 0) столбца с глобальным индексом i_main, локальным $i\_main\ /\ p$.
Процесс 0 с помощью полученного столбца вычисляет

$$\widetilde{B_i} = B_i - A_{i,i\_main}B_{i\_main},$$

$$\forall i \in \{i\_main\ -\ 1, \ldots,\ 0\}$$

.

6 Вычисление решения системы

Преобразованиям строк матрицы A соответсвует умножение ее слева на произведение матриц элементарных преобразований(R). Перестановке столбцов($I = I_1 I_2 \dots I_l$) — умножение ее справа. Таким образом, $E = RAI$. А значит, системе $AX = B$ будет равносильна $RAX = RB \Leftrightarrow RAI \cdot (IX) = RB \Leftrightarrow IX = RB$. Тогда $X = I^{-1}RB$. То есть к полученному столбцу B после обратного шага алгоритма нужно применить перестановку строк, соответствующую I^{-1}. Массив соответствующей перестановки хранится в процессе 0. Он же проводит обратную перестановку для вычисления решения системы.

7 Оценка количества и объема пересылок

Пересылка в 4.2 происходит на каждой итерации с помощью функции MPI_Allreduce, причем объем передаваемых данных не зависит от n и m, а значит, пренебрежимо мал.
Количество же пересылок равно l.
Пересылка в 4.3 происходит с помощью функции MPI_Sendrecv_replace на каждой итерации в объеме nm.
Количество пересылок равно l.
Суммарный объем пересылаемых данных: n^2.
Пересылка в 4.4 происходит с помощью функции MPI_Bcast на каждой итерации в объеме m^2.
Количество пересылок равно l.
Суммарный объем пересылаемых данных: nm
Пересылка в 4.5 происходит с помощью функции MPI_Bcast на каждой итерации в объеме $(l - i\_main - 1) \cdot m^2$.
Количество пересылок равно l.
Суммарный объем пересылаемых данных:

$$\frac{l(l-1)m^2}{2} = \frac{1}{2}n^2 - \frac{1}{2}nm$$

.
Пересылка в 5.2 происходит с помощью функции MPI_Send на тех итерациях обратного хода, где $i\_main \neq k \cdot p, k \in \mathbb{Z}$ в объеме $i\_main \cdot m^2$.
Количество пересылок равно $l - l \ / \ p$.
Суммарный объем пересылаемых данных не больше

$$\frac{l(l-1)m^2}{2} = \frac{1}{2}n^2 - \frac{1}{2}nm$$

.
Тогда, оценка общего количества пересылок:

$$5\frac{n}{m}$$

.
Оценка суммарного объема всех пересылок:

$$2n^2$$

.