Алгоритм Якоби для решения уравнения Лапласа

Алгоритм Якоби для решения уравнения Лапласа в данном контексте основывается на итерационном обновлении значений внутри двумерной сетки с целью достижения стационарного состояния. Он применяется для численного решения уравнения Лапласа, которое в дискретной форме имеет вид:

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - f(i,j)\right)$$

где:

- $u_{i,j}^{(k)}$ - значение внутренней ячейки сетки на итерации k в точке (i,j),
- $f(i,j)$ - функция правой части уравнения Лапласа в точке (i,j).

1. **Инициализация:**
    - Создание двумерной сетки размера $N \times N$.
    - Установка начальных значений внутренних ячеек сетки.
    - Установка граничных условий.
2. **Итерации:**
    - Повторение следующих шагов заданное количество раз (например, 5000 итераций):
        - Для каждой внутренней ячейки сетки применяется итерационная формула Якоби, обновляя ее значение на основе значений соседних ячеек и текущего приближения.
3. **Сходимость:**
    - Алгоритм повторяется до тех пор, пока изменения внутри сетки остаются небольшими или до достижения максимального числа итераций.

4. **Решение:**
    - После завершения итераций, значения внутренних ячеек сетки приближенно соответствуют решению уравнения Лапласа в заданных граничных условиях.
5. **Обмен граничными значениями:**
    - На каждой итерации мастер-процесс обменивается граничными значениями с соседними процессами, чтобы учесть влияние граничных условий внутри параллельной среды MPI.
6. **Завершение:**
    - По завершении итераций алгоритм завершается, и полученные результаты могут быть использованы для анализа или дальнейших вычислений.

Для оценки степени параллелизма можно использовать формулу для ускорения:

$$S = \frac{T_{\text{sequential}}}{T_{\text{parallel}}}$$

где *T*sequential - время выполнения на одном ядре, *T*parallel - время выполнения на *N* ядрах.

Степень параллелизма (*P*) может быть выражена как:

$$P = \frac{S}{N}$$

где *N* - количество ядер.

Для размера сетки 1000x1000 и 5000 итераций:

- 2 ядра: $S = \frac{67.8132}{35.1349} \approx 1.931, P \approx \frac{1.931}{2} \approx 0.9655$ или 96.55%
- 4 ядра: $S = \frac{67.8132}{19.4511} \approx 3.486, P \approx \frac{3.486}{4} \approx 0.8715$ или 87.15%
- 5 ядер: $S = \frac{67.8132}{16.95} \approx 4.001, P \approx \frac{4.001}{5} \approx 0.8002$ или 80.02%

Для размера сетки 2000x2000 и 5000 итераций:

- 2 ядра: $S = \frac{270.696}{138.151} \approx 1.960, P \approx \frac{1.960}{2} \approx 0.9800$ или 98.00%
- 4 ядра: $S = \frac{270.696}{77.2793} \approx 3.499, P \approx \frac{3.499}{4} \approx 0.8747$ или 87.47%
- 5 ядер: $S = \frac{270.696}{69.345} \approx 3.907, P \approx \frac{3.907}{5} \approx 0.7814$ или 78.14%

Оценка трудоемкости алгоритма $\Theta(n^2)$ => чтобы повысить трудоемкость в m раз нужно увеличить n в $\sqrt{m}$

Исходя из тестирования на процессоре AMD Ryzen 4600H @ 3.00GHz 4.00GHz (6 ядер 12 потоков):

- по строкам: размерность задачи
- по столбцам: количество процессоров
- время выполнения в секундах

	1	2	4	6
1000 (3000)	40.71	21.01	11.91	9.39
1414 (3000)	81.55	41.05	23.13	18.65
2000 (3000)	162.23	82.57	45.94	37.44
2450 (3000)	242.81	123.97	70.54	54.45

Эти значения представляют оценку степени параллелизма. Значения около 100% указывают на хорошую эффективность параллелизации, но это приблизительные оценки. Реальная эффективность может варьироваться в зависимости от конкретной архитектуры процессора, характеристик задачи и других факторов.