Московский Государственный Университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по заданию №4 в рамках курса "Суперкомпьютерное моделирование и технологии"

Некрасов Николай Витальевич, 609 группа

Математическая постановка задачи

В области $D \subset \mathbb{R}$, ограниченной контуром γ, рассматривается дифференциальное уравнение Пуассона
$-\Delta u = 1$, где D - трапеция, с вершинами в точках (0, 0), (0, 3), (3, 0), (2, 3).
Пусть также задано граничное условие типа Дирихле:
$u(x, y) = 0, (x, y) \in \gamma.$
Требуется найти функцию $u(x, y)$, удовлетворяющую уравнению и условию Дирихле на границе.

Численный метод

Для приближенного решения был использован метод фиктивных областей. В качестве прямоугольника П был использован квадрат с вершинами в точках (0, 0), (0, 3), (3, 0), (3, 3). Далее была определена функция:
$F(x, y) = \begin{cases} 1, (x, y) \in D \\ 0, (x, y) \notin D \end{cases}$
В прямоугольнике П рассматривается задача Дирихле:
$-\frac{\partial(k(x,y)\frac{\partial u}{\partial x})}{\partial x} - \frac{\partial(k(x,y)\frac{\partial u}{\partial y})}{\partial y} = F(x, y)$
$u(x, y) = 0, (x, y) \in \Gamma$ с кусочно-постоянным коэффициентом
$k(x, y) = \begin{cases} 1, (x, y) \in D \\ 1/\epsilon, (x, y) \notin D \end{cases}$
Требуется найти непрерывную в замыкании П функцию $u(x, y)$, удовлетворяющую дифференциальному уравнению всюду внутри $\Pi \backslash \gamma$, равную нулю на границе Г прямоугольника, и такую, чтобы вектор потока имел непрерывную нормальную компоненту на общей части криволинейной области D и прямоугольника П.
Для численного решения задачи применим метод конечных разностей. В замыкании прямоугольника П введем равномерную сетку с шагом h на каждой оси. Дифференциальное уравнение задачи во всех внутренних точках сетки аппроксимируется разностным уравнением:
$-\frac{1}{h}\left(a_{i+1j}\frac{w_{i+1j}-w_{ij}}{h} - a_{ij}\frac{w_{ij}-w_{i-1j}}{h}\right) - \frac{1}{h}\left(b_{ij+1}\frac{w_{ij+1}-w_{ij}}{h} - b_{ij}\frac{w_{ij}-w_{ij-1}}{h}\right) = F_{ij}$
$i = \overline{1, M-1}, j = \overline{1, N-1}$, где:
$a_{ij} = \frac{1}{h}\int_{y_{j-0.5}}^{y_{j+0.5}} k(x_{i-0.5}, t)dt$
$b_{ij} = \frac{1}{h}\int_{x_{j-0.5}}^{x_{j+0.5}} k(t, y_{i-0.5})dt$
$F_{ij} = \frac{1}{h^2}\iint_{\Pi_{ij}} F(x, y)dxdy$
Отсюда получим СЛАУ с самосопряженной и положительно определенной матрицей, которая будет решена методом наименьших невязок.

Отчет по созданию OpenMP-программы

Для написания OpenMP-программы были использованы конструкции #pragma omp parallel и #pragma omp parallel reduction в функциях scalar_product, multiply_vector_by_matrix, subtract_vector, add_vector, multiply_vector_by_number..

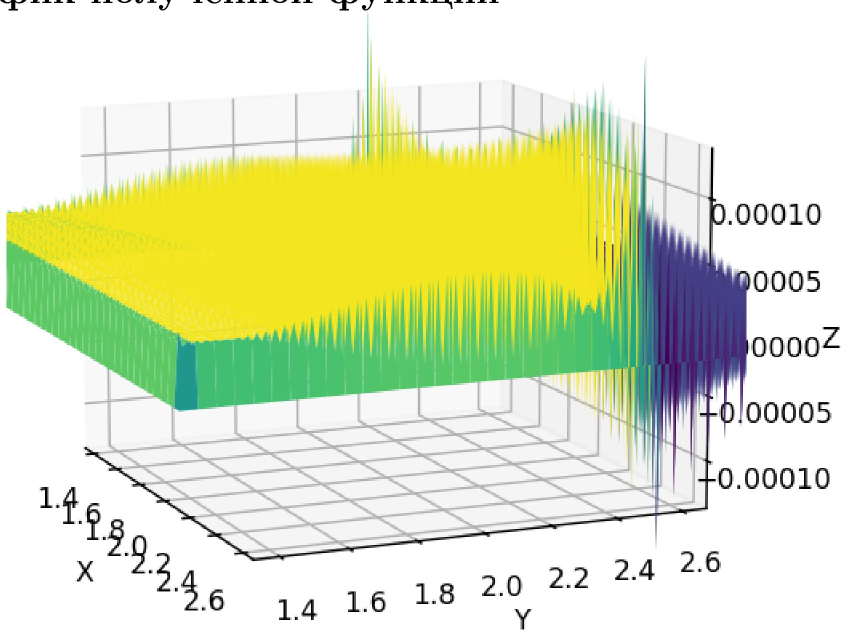
Результаты работы программы

Результаты работы программы для случая 10, 20, 40, 80, 160 точек находятся в файлах result10.txt, result20.txt, result40.txt, result80.txt, result160.txt соответственно.

Таблица 1: Таблица с результатами расчетов на ПВС IBM Polus (OpenMP код).

Число OpenMP-нитей	Число точек сетки $M \times N$	Время решения	Ускорение
2	80×80	71	1.8
4	80×80	45	2.9
8	80×80	25	5.24
16	80×80	15	8.7
4	160×160	108	2.78
8	160×160	65	4.63
16	160×160	38	7.92
32	160×160	25	12

График полученной функции



Графики ускорений

График ускорения для случая, когда кол-во узлов равно 80:

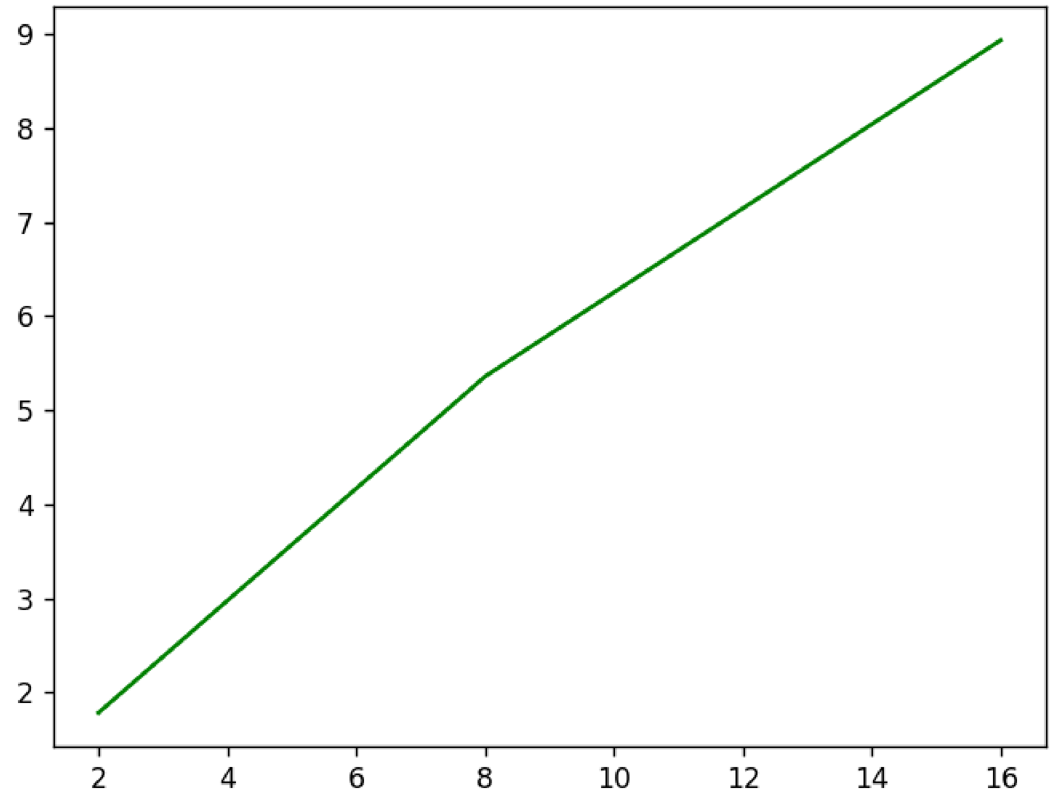
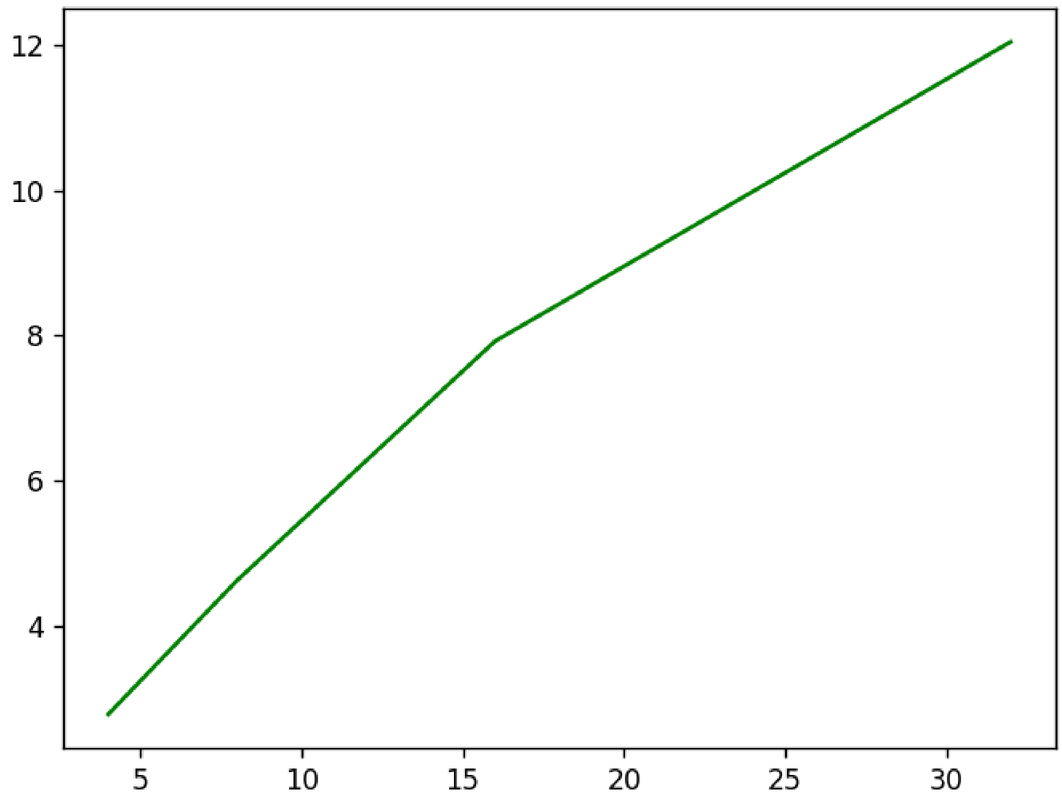


График ускорения для случая, когда кол-во узлов равно 160:



Результаты работы гибридной программы

Таблица 2: Таблица с результатами расчетов на ПВС IBM Polus (MPI+OpenMP код).

Число процессов MPI	Количество OpenMP-нитей в процессе	Число точек сетки ($M \times N$)	Время решения	Ускорение
2	1	80×80	69.5	1.87
2	2	80×80	44.5	2.92
2	4	80×80	24.4	5.31
2	8	80×80	14.6	8.9
4	1	160×160	104	2.78
4	2	160×160	62.5	4.63
4	4	160×160	35.7	7.92
4	8	160×160	24.2	12

Графики ускорений для гибридной программы

График ускорения для случая, когда кол-во узлов равно 80:

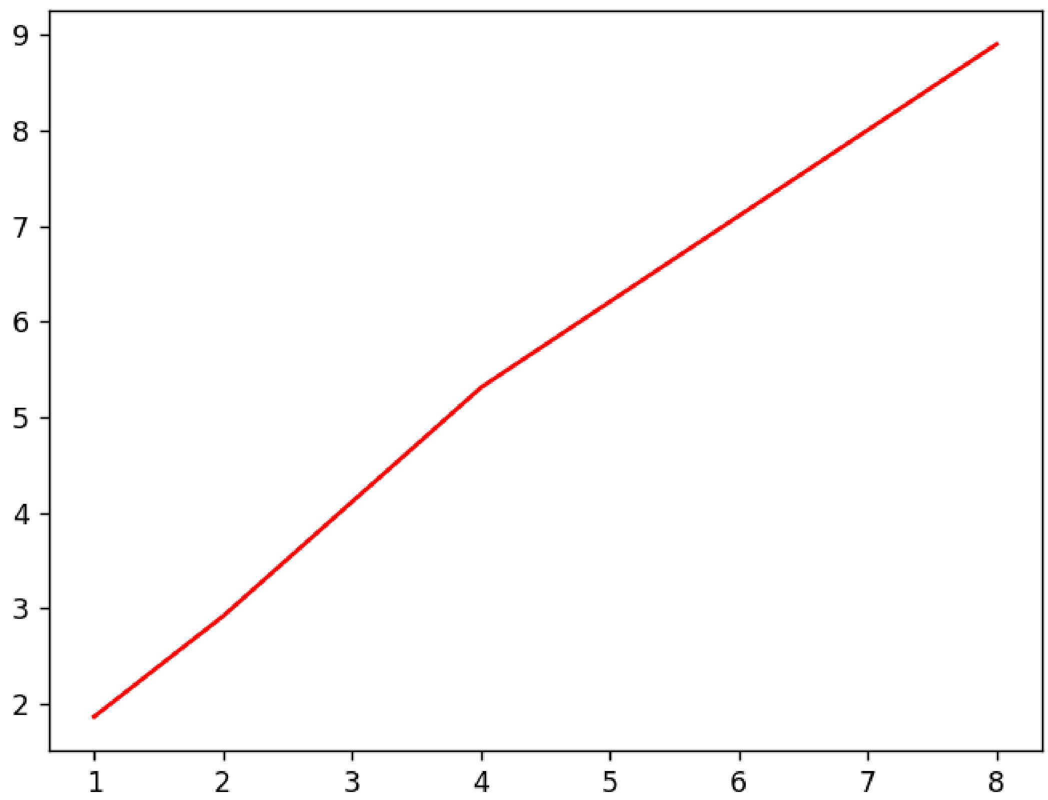
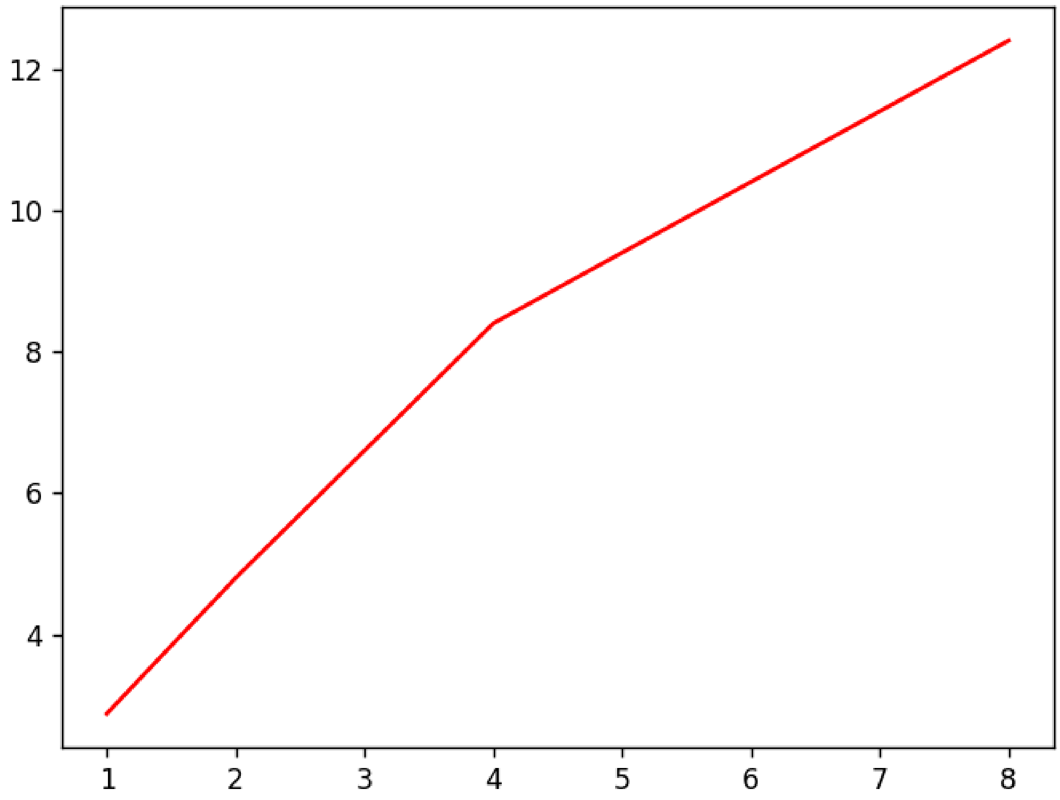


График ускорения для случая, когда кол-во узлов равно 160:



Отчет по созданию CDVMH-программы

Оценка корректности параллельной программы осуществлялась посредством сравнения результата ее работы с результатом работы последовательной программы.

Приведенные ниже данные были получены при запуске программы со значением переменной DVMH_LOGLEVEL = 4.

Строка цикла	**Performance**
42	2.68943e + 08
145	1.51085e + 08
241	1.87503e + 09
250	1.33391e + 09
260	93787.5
276	1.7038e + 09
286	4.77055e + 09
295	2.30882e + 09
304	1.60367e + 09
313	1.81965e + 09
323	4.75698e + 09
337	4.19803e + 09
346	94189.7
361	1.39099e + 09

Таблица 1: Производительность каждого параллельного цикла

Графики ускорений для CDVMH-программы в сравнении с Open-MP

Будем рассматривать решение краевой задачи на 160 точках.
График ускорения CDVMH-программы по сравнению с последовательной:

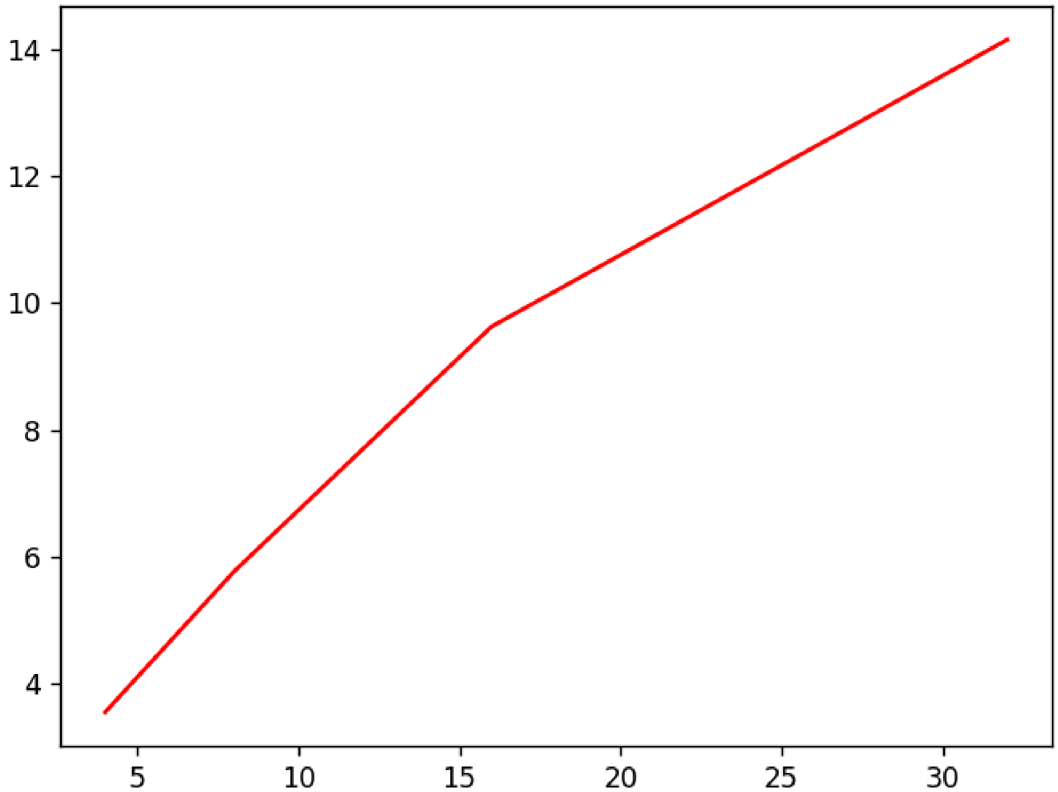
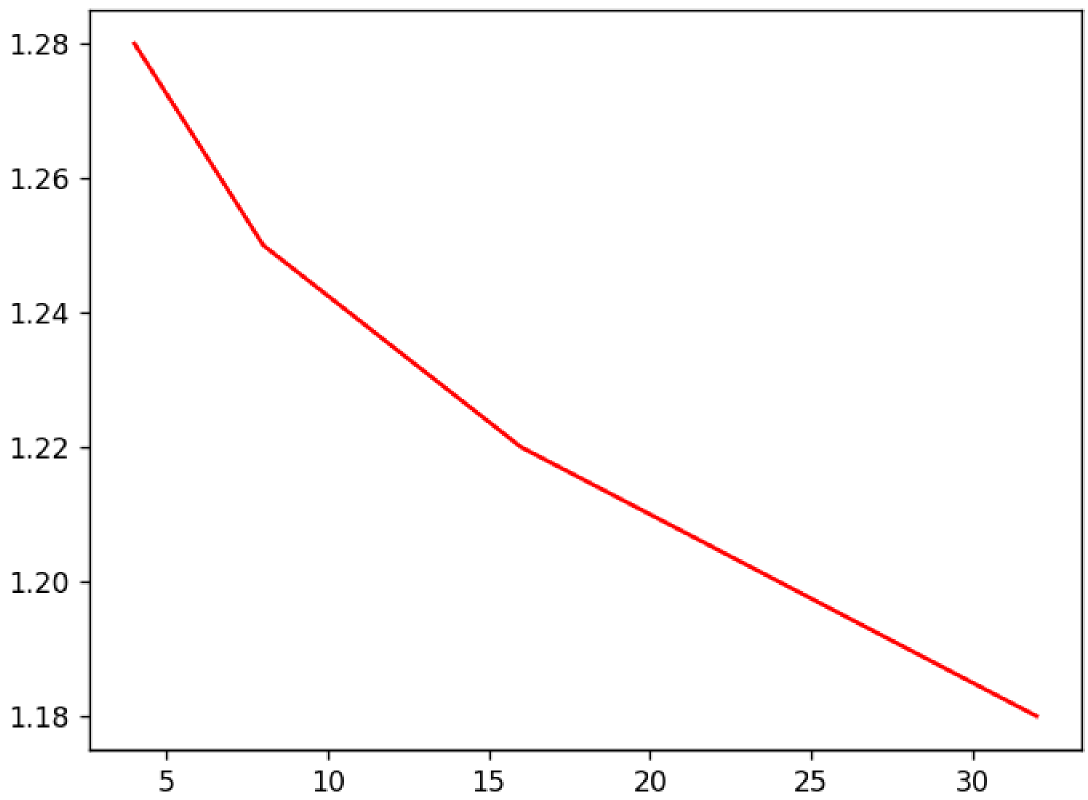
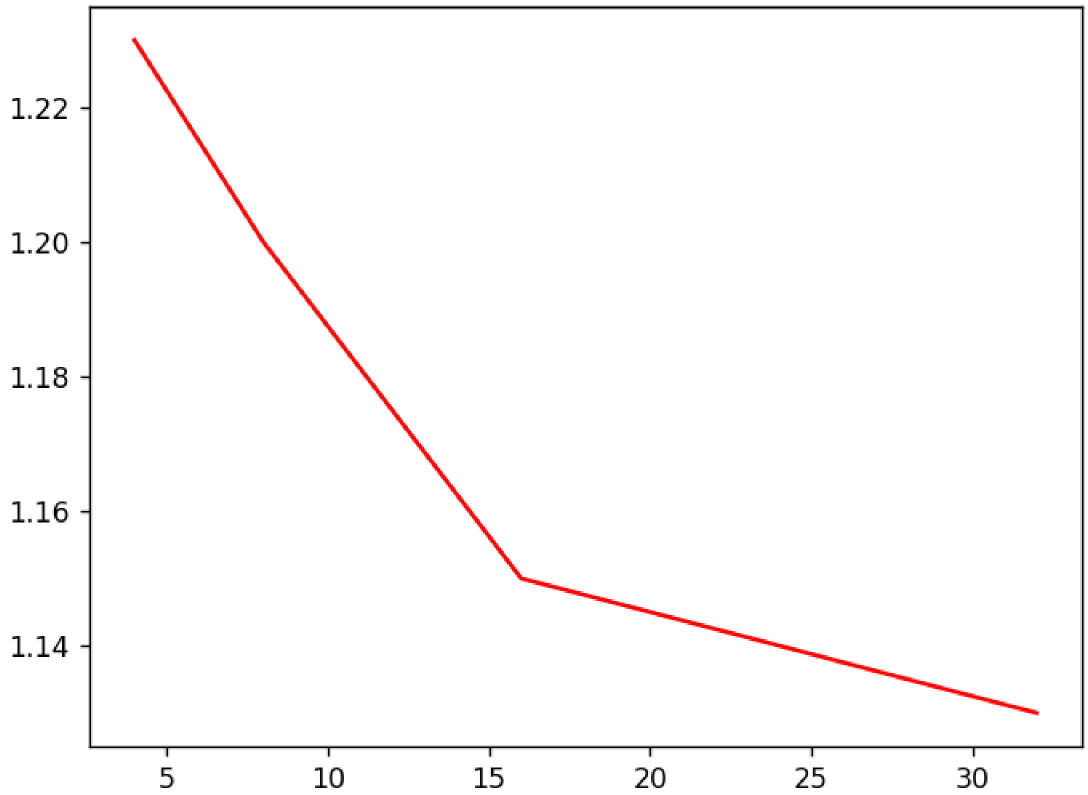


График ускорения CDVMH-программы по сравнению с Open-MP:



На первый взгляд столь сильно ускорение может показаться аномальным, однако оно является ожидаемым, т.к. в Open-MP(как и в MPI) версии программы не распараллелен этап создания матрицы системы.

График ускорения CDVMH-программы по сравнению с гибридной реализацией:


1.22
1.20
1.18
1.16
1.14
5
10
15
20
25
30