Опыт применения параллельного алгоритма LU-разложения для решения линейных систем уравнений в упругопластических задачах*

А.В. Коновалов, А.В. Толмачев, А.С. Партин

Представлены результаты применения параллельного алгоритма LU-разложения, основанного на принципе "разделяй и властвуй", для решения линейных систем уравнений в упругопластических задачах. Реализация алгоритма осуществлена с помощью библиотеки ScaLAPACK на кластере um64 Института математики и механики УрО РАН.

1. Введение

Упругопластическая задача с большими пластическими деформациями физически и геометрически существенно нелинейная и требует большого количества времени для ее решения на персональном компьютере. На решение двумерной задачи затрачивается несколько часов, для трехмерной задачи это время увеличивается до нескольких суток. Существенно сократить время вычислений можно с помощью техники параллельных вычислений, в частности, решая данные задачи на кластерных системах.

Решение упругопластических задач методом конечных элементов осуществляется шагами по нагрузке, и на каждом таком шаге состоит из трех основных этапов [1]:

1) расчет локальных матриц жесткости для конечных элементов и формирование матрицы A (глобальной матрицы жесткости) и вектора B правой части системы линейных алгебраических уравнений (СЛАУ)

$$AX = B \tag{1}$$

относительно искомого вектора X обобщенной скорости в узлах конечно-элементной сетки;

2) решение СЛАУ (1);

3) вычисление напряженно-деформированного состояния в конечных элементах в конце шага нагрузки.

Матрица A имеет ленточный вид. На каждом шаге нагрузки этап 1 выполняется один раз, а этапы 2 и 3 – от десяти до пятнадцати раз для удовлетворения итерационно с приемлемой точностью условию пластичности. При этом матрица жёсткости не меняется, а изменяется только правая часть системы уравнений.

Если этапы 1 и 3 легко распараллеливаются, то распараллеливание процесса решения СЛАУ является сложной задачей. Решение этой задачи итерационными методами рассмотрено в работе [2].

Целью работы является исследование возможностей параллельного алгоритма, основанного на LU-разложении матрицы A, для решения СЛАУ в упругопластических задачах на кластерной системе.

Все численные эксперименты проводились на решении методом конечных элементов задачи сжатия цилиндра плоскими плитами из упругопластического изотропного и изотропно-упрочняемого материала, постановка которой приведена в работе [2].

2. Обоснование выбора алгоритма LU-разложении для решения СЛАУ

2.1. Структура матрицы жесткости

Количество переменных n в СЛАУ равно произведению количества узлов в сетке на количество степеней свободы в каждом узле. Матрица жёсткости A является ленточной с полуши-

* Работа выполнена в рамках программы Президиума РАН "Интеллектуальные информационные технологии, математическое моделирование, системный анализ и автоматизация"

риной k. Для упрощения дальнейшего анализа будем рассматривать разбиение деформируемого тела регулярной сеткой с одинаковым количеством частей d на каждой координатной линии. Тогда для плоской сетки имеем

$$n = 2(d+1)^2, \qquad k = 2(d+3)-1. \tag{2}$$

Для объемной сетки эти соотношения равны

$$n = 3(d+1)^3, \qquad k = 3(d^2+3d+4)-1. \tag{3}$$

В табл. 1 приведены значения параметров n и k для различного числа разбиений d, рассчитанные по формулам (2) и (3).

Таблица 1. Значения параметров матрицы A

d	Тип сетки					
	Плоская сетка			Объёмная сетка		
	k	n	k/n	k	n	k/n
20	45	882	0,051	1391	27783	0,050
40	85	3362	0,025	5171	206763	0,025
60	125	7442	0,017	11351	680943	0,017
80	165	13122	0,013	19931	1594323	0,013
100	205	20402	0,010	30911	3090903	0,010
200	405	80802	0,005	121811	24361803	0,005

Из табл. 1 видно, что матрица A имеет неширокую ленту, и ее относительная ширина в двумерном и трехмерном случаях примерно одинаковая.

2.2. Оценка метода LU-разложения

LU-разложение представляет собой один из способов решения СЛАУ (1). Для произвольной квадратной матрицы A существует разложение

$$A = LU, \tag{4}$$

где L и U - соответственно нижняя и верхняя треугольные матрицы [3]. С учетом (4) уравнение (1) принимает вид

$$LUX = B. \tag{5}$$

Если обозначить $Y = UX$, то решение уравнения (5) находится из последовательного решения двух систем с треугольными матрицами. Сначала решается система

$$LY = B,$$

Затем решается система линейных уравнений с верхней треугольной матрицей

$$UX = Y.$$

После получения разложения (4) для матрицы A, можно использовать его для решения системы уравнений с данной матрицей и разными значениями правой части B. Этот факт дает значительные преимущества LU-разложения по сравнению с методом Гаусса [4] для решения упругопластических задач, так как в этих задачах на каждом шаге нагрузки необходимо удовлетворить условию пластичности. Это осуществляется итерационно путем последовательного изменения правой части уравнения (1), по результатам решения системы на каждой итерации. Опыт вычислений показывает, что условие пластичности с приемлемой точностью выполняется после 10 – 15 итераций. В случае использования метода Гаусса необходимо на каждой итерации заново выполнять прямой и обратный ход данного метода. При использовании LU-разложения на каждой итерации необходимо решать только систему (5).

Приведём оценки количества арифметических операций для решения СЛАУ (1) методом Гаусса и с помощью LU-разложения. Количество арифметических операций при решении

СЛАУ методом Гаусса оценивается, как $O(n^3)$ [4], где n - количество переменных. Для ленточной матрицы с полушириной ленты k оценка принимает значение $O(nk^2)$. При решении системы на c итерациях имеем $O(cnk^2)$.

Оценка количества арифметических операций для ленточной матрицы, необходимых для реализации по приведенным в [3] формулам процедуры LU-разложения, составит $O(nk^2)$, а для решения на его основе системы (5) – $O(nk)$. В сумме для решения системы (1) с помощью LU-разложения в c итерациях потребуется выполнить $O(nk^2 + cnk)$ арифметических операций.

Для плоской сетки с учетом соотношения (2) для решения СЛАУ (1) методом Гаусса потребуется выполнить $O(cd^4)$ арифметических операций, а методом LU-разложения $O(d^4 + cd^3)$.

Отношение δ оценок временных затрат на решение СЛАУ методом Гаусса t_Γ и LU-разложением t_{LU} следующее

$$\delta = \frac{O(cd^4)}{O(d^4 + cd^3)} \approx O\left(\frac{c}{1 + c/d}\right). \qquad (6)$$

В табл. 2 приведены значения $\omega = \dfrac{c}{1 + c/d}$ и $\delta_э = \dfrac{t_\Gamma}{t_{LU}}$, полученные в численном эксперименте решения задачи сжатия цилиндра для $c = 15$.

Таблица 2. Отношение временных затрат методов Гаусса и LU-разложения при $c = 15$

d	10	20	30	40	50	60	70	80	90	100
ω	6,0	8,6	10,0	10,9	11,5	12,0	12,4	12,6	12,9	13,0
$\delta_э$	5,2	5,8	7,3	6,6	7,7	9,2	9,2	9,8	10,3	10,7

Из формулы (6) и данных табл. 2 видно, что временные затраты на решение СЛАУ, когда итерационно изменяется ее правая часть, методом LU-разложения существенно меньше по сравнению с методом Гаусса. Эти данные послужили основанием для выбора алгоритма распараллеливания решения СЛАУ на основе LU-разложения.

На рис. 1 представлены относительные доли времени $t_{отн}$ вычислений, приходящиеся на каждый из этапов решения на одном шаге нагрузки в задаче о сжатии цилиндра, при использовании метода LU-разложения для решения СЛАУ. Видно, что доля времени на решение СЛАУ увеличивается с ростом числа разбиений сетки и на мелких сетках занимает почти половину общего времени расчетов на шаге нагрузки.

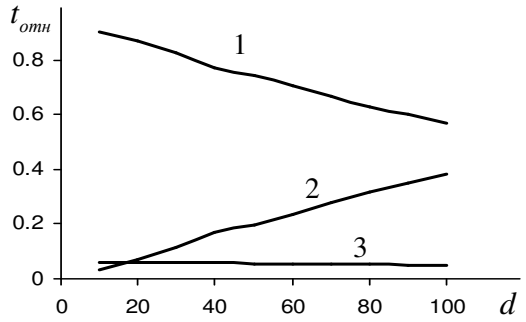

Рис. 1. Относительные доли времени вычислений этапов решения на одном шаге нагрузки в задаче сжатия цилиндра: 1- формирование матрицы A, 2 – решение СЛАУ, 3 – вычисление напряженно-деформированного состояния

3. Результаты распараллеливания решения СЛАУ

Для исследования был взят параллельный алгоритм вычисления LU-разложения и решения системы линейных уравнений без выбора ведущего элемента, основанный на парадигме "разделяй и властвуй", описанный в работах [5, 6] и реализованный в библиотеке ScaLAPACK [7].

3.1. Суть алгоритма вычисления LU-разложения и решения системы линейных уравнений без выбора ведущего элемента с использованием парадигмы "разделяй и властвуй"

Для решения системы линейных уравнений (1) матрица A распределяется на p процессоров следующим образом

$$\begin{pmatrix} A_1 & B_1^U & & & & & \\ B_1^L & C_1 & D_2^U & & & & \\ & D_2^L & A_2 & B_2^U & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & B_{p-1}^L & C_{p-1} & D_p^U & \\ & & & & D_p^L & A_p \end{pmatrix} \begin{pmatrix} x_1 \\ \xi_1 \\ x_2 \\ \vdots \\ \xi_{p-1} \\ x_p \end{pmatrix} = \begin{pmatrix} b_1 \\ \beta_1 \\ b_2 \\ \vdots \\ \beta_{p-1} \\ b_p \end{pmatrix}, \qquad (7)$$

где A_i - матрица размерности $n_i \times n_i$, $i = 1,..., p$; C_i - матрица размерности $k \times k$; x_i, b_i - векторы размерности n_i; ξ_i, β_i - векторы размерности k; $\sum_{i=1}^{p} n_i + (p-1)k = n$, k - полуширина ленты матрицы. Шаблон матрицы в уравнении (7) для p=4 показан на рис. 2 а. Такое трёхдиагональное блочное разбиение возможно только в случае, если $\forall i \; n_i > k$. Это условие ограничивает предельно возможный уровень параллелизма, а именно: максимальное количество процессоров p, которое может быть использовано для параллельного исполнения, должно удовлетворять условию $p < (n+k)/(2k)$.

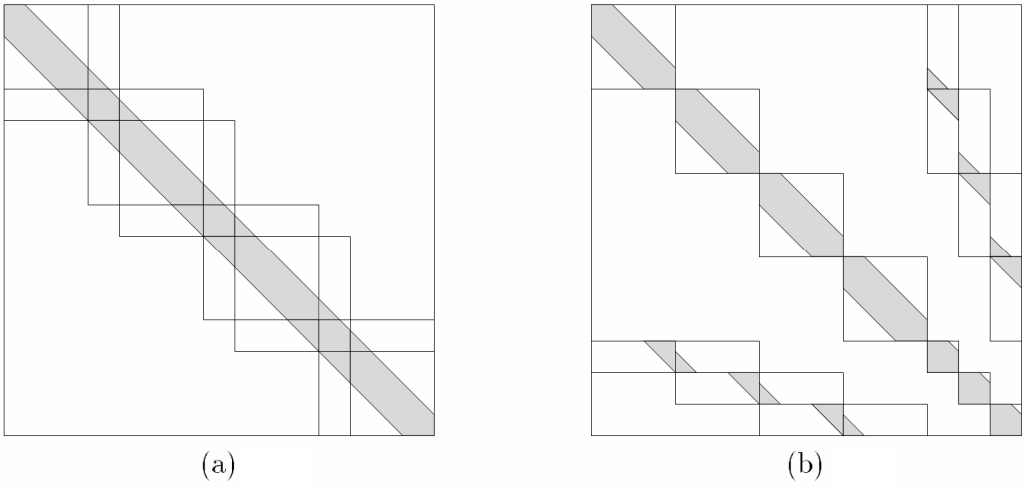

Рис. 2. Структура исходной (а) и подверженной блочной чётно-нечётной перестановке (б) матриц

Далее над матрицей производится первый шаг блочно-циклической редукции, при котором переставляются местами строки и столбцы таким образом, чтобы сначала шли все нечётные строки и столбцы, а потом все чётные. Получаем, так называемую чётно-нечётную перестановку

$$\left(\begin{array}{cccccc|cccccc}
A_1 & & & & & & B_1^U & & & & & \\
& A_2 & & & & & D_2^L & B_2^U & & & & \\
& & \ddots & & & & & \ddots & \ddots & & & \\
& & & A_{p-1} & & & & & \ddots & B_p^U & & \\
& & & & A_p & & & & & D_p^L & & \\
\hline
B_1^L & D_2^U & & & & & C_1 & & & & & \\
& B_2^L & \ddots & & & & & C_2 & & & & \\
& & \ddots & \ddots & & & & & \ddots & & & \\
& & & B_{p-1}^L & D_p^U & & & & & C_{p-1} & &
\end{array}\right)\left(\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_{p-1} \\ x_p \\ \hline \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{p-1} \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_{p-1} \\ b_p \\ \hline \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{p-1} \end{array}\right). \qquad (8)$$

Шаблон матрицы в уравнении (8) представлен на рис. 5 б. Равенство (8) можно записать в более общей блочной форме

$$\begin{pmatrix} \tilde{A} & B^U \\ B^L & C \end{pmatrix}\begin{pmatrix} x \\ \xi \end{pmatrix} = \begin{pmatrix} b \\ \beta \end{pmatrix}. \qquad (9)$$

LU-разложение матрицы $\tilde{A}$ имеет вид:

$$\begin{pmatrix} \tilde{A} & B^U \\ B^L & C \end{pmatrix} = \begin{pmatrix} L & 0 \\ B^L U^{-1} & 1 \end{pmatrix}\begin{pmatrix} U & L^{-1}B^U \\ 0 & S \end{pmatrix}, \qquad (10)$$

где $S = C - B^L \tilde{A}^{-1} B^U$ - дополнение Шура [3] для матрицы $\tilde{A}$.

Для экономии памяти матрицы $B_i^L U_i^{-1}, L_i^{-1}B_i^U, D_i^U R_i^{-1}$ и $L_i^{-1}D_i^L$ помещаются в те же места памяти, которые выделены для матриц B_i^L, B_i^U, D_i^U и D_i^L соответственно. Так как в процессе вычислений в матрицах D_i^U и D_i^L происходит дополнительное заполнение, как показано на рис. 3 тёмным цветом, то требуется дополнительное место в памяти для $2kn$ чисел с плавающей точкой. Общее требование к памяти примерно в два раза больше, чем для последовательного алгоритма.

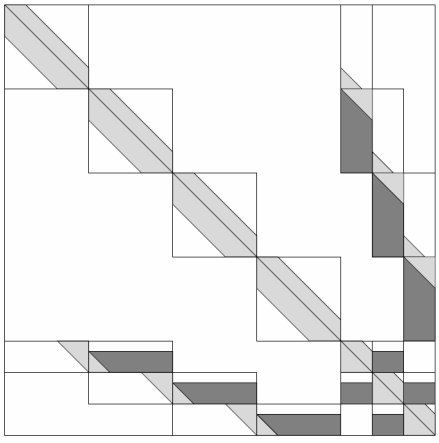

Рис. 3. Заполнение матрицы СЛАУ, получающееся в процессе LU-разложения

Дополнение Шура S в матрице A для $\tilde{A}$ представляет собой блочную трёхдиагональную матрицу размерности $(p-1)\times(p-1)$ с размером блоков $k \times k$:

$$S = \begin{pmatrix} T_1 & U_2 & & & \\ V_2 & T_2 & U_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & U_{p-1} \\ & & & V_{p-1} & T_{p-1} \end{pmatrix},$$

где $T = C_i - B_i^L A_i^{-1} B_i^U - D_{i+1}^U A_{i+1}^{-1} D_{i+1}^L = C_i - \left(B_i^L R_i^{-1}\right)\left(L_i^{-1} B_i^U\right) - \left(D_{i+1}^U R_{i+1}^{-1}\right)\left(L_{i+1}^{-1} D_{i+1}^L\right)$,

$U_i = -\left(D_i^U R_i^{-1}\right)\left(L_i^{-1} B_i^U\right)$, $V_i = -\left(B_i^L R_i^{-1}\right)\left(L_i^{-1} D_i^L\right)$.

Используя разложение (10), из уравнения (9) получаем:

$$\begin{pmatrix} R & L^{-1}B^U \\ & S \end{pmatrix}\begin{pmatrix} x \\ \xi \end{pmatrix} = \begin{pmatrix} L & \\ B^L R^{-1} & I \end{pmatrix}^{-1}\begin{pmatrix} b \\ \beta \end{pmatrix} = \begin{pmatrix} L^{-1} & \\ -B^L R^{-1} L^{-1} & I \end{pmatrix}\begin{pmatrix} c \\ \gamma \end{pmatrix},$$

где части c_i и β_i векторов c и β задаются, как

$$c_i = L_i^{-1} b_i$$

$$\gamma_i = \beta_i - B_i^L R_i^{-1} c_i - D_{i+1}^U R_{i+1}^{-1} c_{i+1}$$

До этого момента не требуется межпроцессорной коммуникации, поскольку каждый процессор независимо вычисляет разложение блока диагонали $A_i = L_i R_i$ и вычисляет блоки $B_i^L U_i^{-1}, L_i^{-1} B_i^U, D_i^U R_i^{-1}, L_i^{-1} D_i^L$ и $L_i^{-1} b_i$. Каждый процессор формирует свою часть редуцированной системы $S\xi = \gamma$, где

$$S_i = \begin{pmatrix} -D_i^U R_i^{-1} L_i^{-1} D_i^L & -D_i^U R_i^{-1} L_i^{-1} B_i^U \\ -B_i^L R_i^{-1} L_i^{-1} D_i^L & C_i - B_i^L R_i^{-1} L_i^{-1} B_i^U \end{pmatrix}$$

$$\gamma_i = \begin{pmatrix} -D_i^U R_i^{-1} c_i \\ \beta_i - B_i^L R_i^{-1} c_i \end{pmatrix}.$$

Матрица S_i имеет размерность $2k \times 2k$, векторы ξ_i и γ_i имеют размерность $2k$.

После выполнения локальных вычислений для завершения построения матрицы T_i процессор с номером i посылает процессору с номером $i+1$ матрицу $C_i - B_i^L R_i^{-1} L_i^{-1} B_i^U$.

Чётно-нечётная перестановка, вычисление разложения и формирование редуцированной системы повторятся до тех пор, пока матрица редуцированной системы не станет заполненной матрицей размерностью $k \times k$. В этом случае получившаяся система может быть решена на одном процессоре. Так как на каждой итерации редукции размерность матрицы уменьшается в два раза, то для достижения желаемого результата, необходимо выполнить $\log_2 p - 1$ шагов.

После нахождения векторов ξ_i, процессоры вычисляют свою часть вектора x:

$$x_1 = R_1^{-1}\left(c_1 - L_i^{-1} B_1^U \xi_1\right)$$

$$x_i = R_i^{-1}\left(c_i - L_i^{-1} D_1^L \xi_{i-1} - L_i^{-1} B_i^U \xi_i\right)$$

$$x_p = R_p^{-1}\left(c_p - L_{p-1}^{-1} D_p^L \xi_{p-1}\right)$$

3.2. Результаты вычислительных экспериментов

Вычислительные эксперименты проводили на кластере um64 Института Математики и Механики УрО РАН. Кластер состоит из 32 двухпроцессорных двухядерных модулей на базе процессоров AMD Operton с тактовой частотой 2.6 ГГц. Для вычислительных экспериментов опи-

санного в п. 3.1 алгоритма использовали библиотеку ScaLAPACK , реализованную в Intel MKL версии 10.0.010, и MPI, реализованную в библиотеке OpenMPI версии 1.3.3 на сети InfiniBand.

Для тестирования были взяты сетки с параметрами, представленными в табл.3, где d - количество разбиений в сетке, k - полуширина матрицы жёсткости, p - максимальное количество процессоров, которые можно использовать для решения системы уравнений.

Таблица 3. Параметры сеток, использовавшихся в вычислительных экспериментах

d	k	n	k/n	p
70	145	10082	0,0144	35
100	205	20402	0,0100	50
120	245	29282	0,0084	60
150	305	45602	0,0067	75
200	405	80802	0,0050	100

На рис. 4 представлены полученные значения коэффициента ускорения a_p при распараллеливании вычислений LU-разложения матрицы A системы (1). Здесь и далее коэффициент ускорения $a_p = \frac{t_1}{t_p}$, где t_1 - время выполнения последовательного алгоритма на одном процессоре, t_p - время выполнения параллельного алгоритма на многопроцессорной вычислительной системе с числом процессоров p, $p > 1$.

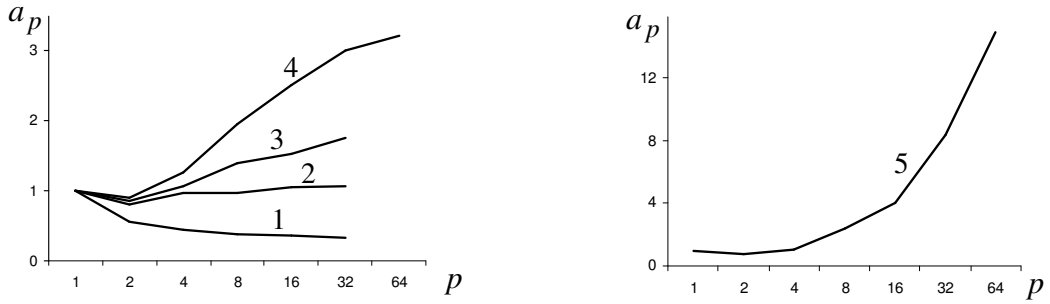

Рис. 4. Значения коэффициента ускорения a_p при выполнении LU-разложения матрицы СЛАУ в зависимости от числа процессоров p и параметра разбиения сетки d : 1- d =70, 2- d =100, 3- d =120, 4- d =150, 5- d =200

Из рис. 4 видно, что ускорение вычислений при распараллеливании появляется на сетках с количеством разбиений больше 100. При меньшем числе разбиений происходит замедление вычислений за счёт того, что время на передачу данных превышает время непосредственно затрачиваемое на вычисление на процессорах. Скачок времени на двух процессорах получается из-за того, что объём вычислений в рассматриваемом алгоритме примерно в 2 раза больше, чем в последовательной схеме, и дополнительно прибавляются затраты на передачу данных по сети. С увеличением количества разбиений уменьшается относительная ширина ленты матрицы A, и коэффициент ускорения становится больше 1. Его величина тем больше, чем мельче сетка, однако рост коэффициента ускорения замедляется с увеличением количества процессоров.

На рис. 5 приведены значения коэффициента ускорения a_p при распараллеливании решения системы (5). Видно, что при всех рассматриваемых значениях параметра разбиения сетки происходит замедление решения системы (5) с ростом числа процессоров, причем чем мельче сетка, тем скорость замедления меньше.

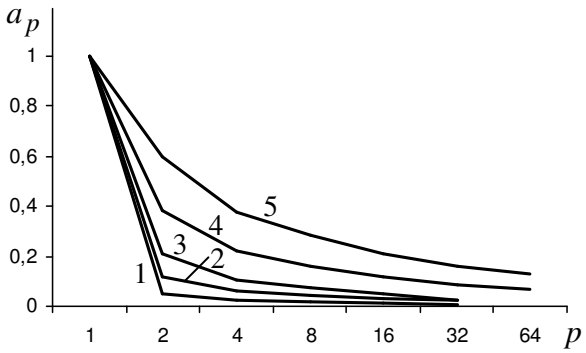

Рис. 5. Значения коэффициента ускорения a_p при распараллеливании решения системы (5) в зависимости от числа процессоров p и параметра разбиения сетки d : 1- d =70, 2- d =100, 3- d =120, 4- d =150, 5- d =200

В табл. 4 представлены значения времени, затрачиваемого на одно решение системы (5).

Таблица 4. Время, затрачиваемое на одно решение системы (5), мс

d	Количество процессоров						
	1	2	4	8	16	32	64
30	1,3	236,0	471,3	746,4			
50	5,6	258,0	466,8	743,2	1138,7		
70	12,6	264,7	494,5	757,3	1112,1	1475,4	
100	33,8	281,4	539,3	762,2	1083,2	1526,5	
120	57,0	273,3	557,0	786,6	1138,3	2076,8	
150	127,3	331,2	569,4	787,6	1108,4	1495,1	1886,6
200	250,3	418,6	664,3	874,4	1183,7	1549,2	1919,9

Несмотря на то, что ускорение расчетов не наблюдается даже на сетках размером 200x200, время, затрачиваемое на решение системы при одном и том же числе процессоров, практически не увеличивается с увеличением размера задачи (параметра разбиения сетки d), что свидетельствует о том, что практически всё время, которое тратится на решение системы уравнений (5), определяется задержкой сети.

На рис. 6 приведены значения коэффициента ускорения при распараллеливании решения

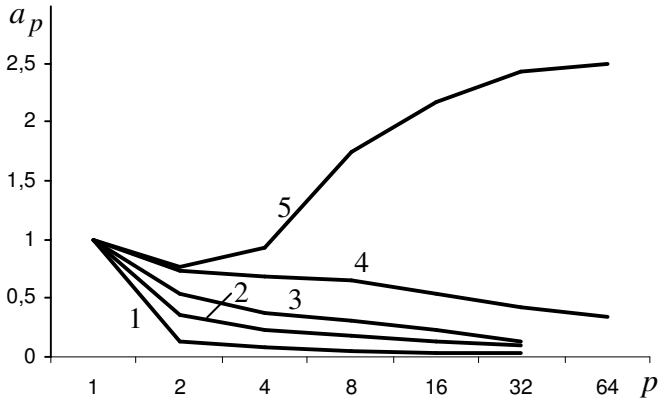

Рис. 6. Значения коэффициента ускорения a_p при распараллеливании решения СЛАУ (1) и c =15 в зависимости от числа процессоров p и параметра разбиения сетки d : 1 - d =70, 2 - d =100, 3 - d =120, 4 - d =150, 5 - d =200

на шаге нагрузки СЛАУ (1), когда имеет место LU-разложение матрицы системы A и решение системы (5). При этом по условию упругопластической задачи система (5) решалась 15 (c=15) раз для удовлетворения условию пластичности, матрицы L и U не изменялись, а изменялась только матрица B в ее правой части. Видно, что при наличии ускорения для LU-разложения матрицы A (см. рис.4) и его отсутствии при решении системы (5) (см. рис. 5) в сумме ускорение появляется на сетках с количеством разбиений 200. Это объясняется тем, что время решения системы уравнений (5) для больших систем на несколько порядков меньше, чем время, затрачиваемое на LU-разложение матрицы, поэтому в сумме наблюдается выигрыш по времени при распараллеливании вычислений. Поскольку объём арифметических операций для LU-разложения есть асимптотически самая большая величина в решаемой задаче, то при дальнейшем увеличении количества разбиений сетки ускорение будет увеличиваться.

4. Выводы

Рассмотренный в работе алгоритм решения СЛАУ в упругопластических задачах можно использовать для решения упругопластических задач с относительной шириной ленты менее 0,005.

Литература

1. Поздеев А.А., Трусов П.В., Няшин Ю.И.. Большие упруго-пластические деформации. М., Наука, 1986 - 232с.
2. Демешко И.П., Акимова Е.Н., Коновалов А.В. Применение параллельных алгоритмов для решения системы линейных алгебраических уравнений с ленточной матрицей итерационными методами на кластерной системе // Труды международной конференции "Параллельные Вычислительные технологии (ПаВТ'2009)", Нижний Новгород, 30 марта – 3 апреля 2009. – С. 444 – 449. – Челябинск: Изд. ЮУрГУ, 2009.– 839 с. (электронное издание).
3. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ. М., Вильямс, 2008. – 1296 с.
4. Бахвалов Н. С. Численные методы. М., Наука, 1975. – 632 с.
5. Cleary A., Dongarra J. Implementation in ScaLAPACK of Divide-and-Conquer Algorithms for Banded and Tridiagonal Linear Systems // Computer Science Dept. Technical Report CS-97-358, 1997: [http://www.netlib.org/lapack/lawns/downloads].
6. Arbenz P., Cleary A., Dongarra J., Hegland M. A. Comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems // Tech. Report 312, ETH Zurich, Computer Science Department, 1999: [http://www.netlib.org/lapack/lawns/downloads].
7. Blackford L. S., Choi J., Cleary A., D'Azeuedo E. at all. ScaLAPACK User's Guide. 1997: [http://www.netlib.org/scalapack/slug].