УДК 519.615.5

С. А. Ишанов, С. В. Клевцур,
К. С. Латышев, Д. И. Пялов

МНОГОПРОЦЕССОРНАЯ РЕАЛИЗАЦИЯ ОДНОГО ИТЕРАЦИОННОГО АЛГОРИТМА ДЛЯ СИСТЕМ С РАСПРЕДЕЛЁННОЙ ПАМЯТЬЮ

Рассмотрены многопроцессорные реализации α–β-итерационного алгоритма решения систем пятиточечных разностных уравнений.

Some multiprocessors realizations of the α–β-iteration algorithm for solution of five-dots difference equations systems are considered.

Ключевые слова: система линейных уравнений, разностная схема, α—β-итерация, пятиточечная схема, мультипроцессорная система.

Keywords: system of linear equations, difference scheme, α—β-iteration, five-dots scheme, multiprocessors system.

Введение

Использование численных методов в задачах математического моделирования приводит к необходимости решения систем линейных алгебраических уравнений (СЛАУ) больших порядков, что влечет за собой большие затраты вычислительных ресурсов. В работе [1] для решения двухмерных разностных уравнений предложен α—β-итерационный метод, не требующий априорной информации о границах спектра разностного оператора, малочувствительный к сильным изменениям коэффициентов разностной схемы и обладающий высокой скоростью сходимости по сравнению с остальными методами [2]. Несмотря на указанные достоинства метода, время вычислений заметно возрастает при увеличении числа узлов разностной сетки. Так, например, расчет модельной двухмерной задачи теплопроводности в области размером 500×500 узлов с 20 шагами по времени занимает около 100 с, а решение той же задачи на сетке 2000×500 с 10 шагами по времени на том же компьютере занимает уже порядка 600 с. В настоящее время для уменьшения времени расчетов широко используются технологии параллельных вычислений на многопроцессорных системах.

Рассмотрим один из вариантов адаптации α—β-итерационного метода для многопроцессорной системы с распределенной памятью. Основу данного метода составляют рекурсивные вычисления, поэтому параллельная реализация алгоритма имеет ряд недостатков, характерно проявляющихся при использовании систем с распределенной памятью. Однако некоторые особенности α—β-итерационного алгоритма позволяют получить заметное ускорение при использовании нескольких процессоров. Отметим, что идея параллельной адаптации данного

Вестник Российского государственного университета им. И. Канта. 2009. Вып. 10. С. 64 – 73.

метода была опубликована в работе [3], где приводится оценка ускорения вычислений в реализации для 32-транспьютерной системы.

Коротко рассмотрим $\alpha-\beta$-итерационный метод [1] на примере двумерного уравнения теплопроводности [4]

$$\frac{du}{dt} + div(D\,\mathrm{grad}\,u) = Q = L \qquad (1)$$

с граничными и начальными условиями $u\big|_{x\in G} = \varphi(x, t)$, $u\big|_{t=0} = u_0(x)$.

В прямоугольной системе координат уравнение (1) в случае стационарного коэффициента теплопроводности принимает вид

$$\frac{du(x, y, t)}{dt} = D\left(\frac{d^2u(x, y, t)}{dx^2} + \frac{d^2u(x, y, t)}{dy^2}\right) + F(x, y, t),$$

$$u\big|_{x=0} = \varphi_0(t), \quad u\big|_{x=a} = \varphi_a(t), \quad u\big|_{y=0} = \psi_0(t), \quad u\big|_{y=b} = \psi_b(t), \quad u\big|_{t=0} = u_0(x).$$

Для аппроксимации дифференциальных операторов используем простейшую разностную схему на пятиточечном шаблоне:

$$\frac{u_{i,j}^{\tau+1} - u_{i,j}^{\tau}}{\Delta t} = D\left(\frac{u_{i+1,j}^{\tau+1} - 2u_{i,j}^{\tau+1} + u_{i-1,j}^{\tau+1}}{\Delta x^2} + \frac{u_{i,j+1}^{\tau+1} - 2u_{i,j}^{\tau+1} + u_{i,j-1}^{\tau+1}}{\Delta x^2}\right) + F_{i,j}^{\tau+1}.$$

Вычисление значений u на временном слое $\tau + 1$ при известных значениях u и F на слое τ сводится к разрешению пятиточечной схемы, представленной в общем виде уравнением

$$B_{i,n}U_{i,n-1} + K_{i,n}U_{i-1,n} - C_{i,n}U_{i,n} + E_{i,n}U_{i+1,n} + V_{i,n}U_{i,n+1} + F_{i,n} = 0 \qquad (2)$$

с граничными условиями

$$U_{1,n} = \varphi_{2,n}U_{2,n} + \psi_{2,n}, \quad U_{N_i,n} = \varphi_{N_i-1,n}U_{N_i-1,n} + \psi_{N_i-1,n},$$
$$U_{i,1} = \varphi_{i,2}U_{i,2} + \psi_{i,2}, \quad U_{i,N_n} = \varphi_{i,N_n-1}U_{i,N_n-1} + \psi_{i,N_n-1}.$$

Пусть решение исходной системы (2) удовлетворяет соотношениям

$$U_{i,n} = \alpha_{i+1,n}U_{i+1,n} + \beta_{i+1,n}, \quad U_{i,n} = \gamma_{i-1,n}U_{i-1,n} + d_{i-1,n},$$
$$U_{i,n} = \tilde{\alpha}_{i,n+1}U_{i,n+1} + \tilde{\beta}_{i,n+1}, \quad U_{i,n} = \tilde{\gamma}_{i,n-1}U_{i,n-1} + \tilde{d}_{i,n-1},$$

где $\alpha, \gamma, \tilde{\alpha}, \tilde{\gamma}, \beta, d, \tilde{\beta}, \tilde{d}$ — неизвестные коэффициенты прогонки. Метод $\alpha-\beta$-итераций, как уже отмечалось, подробно изложен в работе [1] и мы не будем приводить расчетные формулы подробно. С модификациями метода можно ознакомиться в работах [5; 6].

Метод распараллеливания

Среди методов, направленных на решение систем сеточных уравнений с использованием многопроцессорных систем, выделим методы, основанные на разбиении исходной области решения задачи на подобласти. Этот подход тесно связан с принципом геометрического параллелизма. Суть этого подхода заключается том, что исходная область разбивается на части, например на две (рис. 1), и в каждой из подобластей расчет ведет свой процессор.

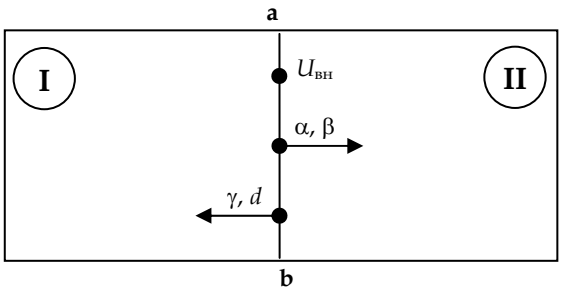

Рис. 1. Разбиение без перекрытия ($U_{вн}$ — значение на внутренней границе)

Возможны два способа разбиения области решения: без перекрытия и с перекрытием. В первом случае общими для подобластей являются лишь точки внутренней границы. При втором способе области перекрываются, имея общую часть. В этом случае сеточные функции в достаточно большом числе точек могут определяться одновременно в рабочем поле процессора I и II.

В случае перекрывающихся областей расчет начинается с постановки искусственных граничных условий на границах подобластей ab и cd (рис. 2). С этими граничными условиями производится расчет сеточной функции в подобластях I и II. После окончания расчета в области I определяются новые значения U на линии ab, в свою очередь, из решения задачи в области II определяются значения U на линии cd. После этого расчет в области I повторяется с новыми граничными значениями U на линии cd, а в области II — на линии ab. Процесс повторяется до сходимости итераций между подобластями.

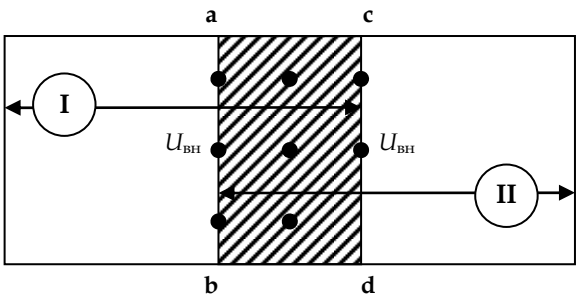

Рис. 2. Разбиение с перекрытием

Очевидно, что, с одной стороны, чем больше перекрытие областей, тем быстрее сходятся итерации между подобластями, а с другой — чем больше общих точек, тем ниже эффективность параллелизма, так как процессоры делают больше общей работы. Кроме того, увеличение числа точек в каждой подобласти приводит к замедлению итерационного процесса. Это также приводит к заметному снижению эффективности параллельных вычислений.

Вариант разбиения без перекрытия свободен от указанных недостатков. Однако в этом случае остается непонятным механизм формиро-

вания граничного условия на внутренней границе ab в процессе итераций между подобластями.

Итерационный $\alpha-\beta$-метод дает простую возможность формирования условий на внутренней границе в процессе итераций между подобластями. В самом деле, в ходе итераций внутри каждой подобласти вычисляются не значения сеточной функции U, а значения прогоночных коэффициентов. После окончания расчета в области I на внутренней границе ab получаются новые значения коэффициентов α и β. А после окончания расчета в области II на этой границе определяются значения коэффициентов γ и d. Полученные значения коэффициентов α и β послужат новыми граничными условиями для области II, а значения коэффициентов γ и d — граничными условиями для области I. Построенный таким образом алгоритм нахождения внутренних граничных условий служит основой параллельного варианта.

Отметим, что рекурсивность формул $\alpha-\beta$-итерационного метода является основной проблемой эффективной реализации многопроцессорной программы. Рассмотрим этапы решения задачи $\alpha-\beta$-итерационным методом.

I. Выделение необходимой памяти.

II. Задание начальных условий.

III. Процесс по t:

1) задание коэффициентов схемы и граничных условий;

2) α-итерационный процесс:

а) первая половина α-процесса (вычисление $\alpha, \gamma, \tilde{\alpha}$),

б) вторая половина α-процесса (вычисление $\alpha, \gamma, \tilde{\gamma}$),

в) проверка сходимости;

3) β-итерационный процесс:

а) первая половина β-процесса (вычисление $\beta, d, \tilde{\beta}$),

б) вторая половина β-процесса (вычисление $\beta, d, \tilde{d}$),

в) проверка сходимости;

4) вычисление значения сеточной функции в момент времени t;

5) сохранение результатов.

При разбиении области решения по координате x получаем различную степень параллелизма отдельных стадий вычислительного процесса. Отметим, что этапы I, II, III.1, III.2в, III.3в и III.5 могут выполняться параллельно, независимо от других процессоров. Вычисление значений сеточной функции (этап III.4) тоже может проводиться автономно. На этих этапах теоретическая эффективность может достигать 100 %. Стоит отметить, что этап III.5 связан с записью на жесткий диск, что требует больше времени, нежели работа с оперативной памятью.

Рассмотрим более детально работу системы на стадиях III.2а, III.2б, III.3а и III.3б. Организация вычислительного процесса на каждом из четырех этапов одинакова, поэтому рассмотрим только первую половину α-процесса (вычисление $\alpha, \gamma, \tilde{\alpha}$). Для произведения расчетов одному процессору необходимо наличие краевых условий для α или γ либо уже

вычисленных значений α и γ. Тогда алгоритм работы каждого процессора можно представить в виде схемы (рис. 3).

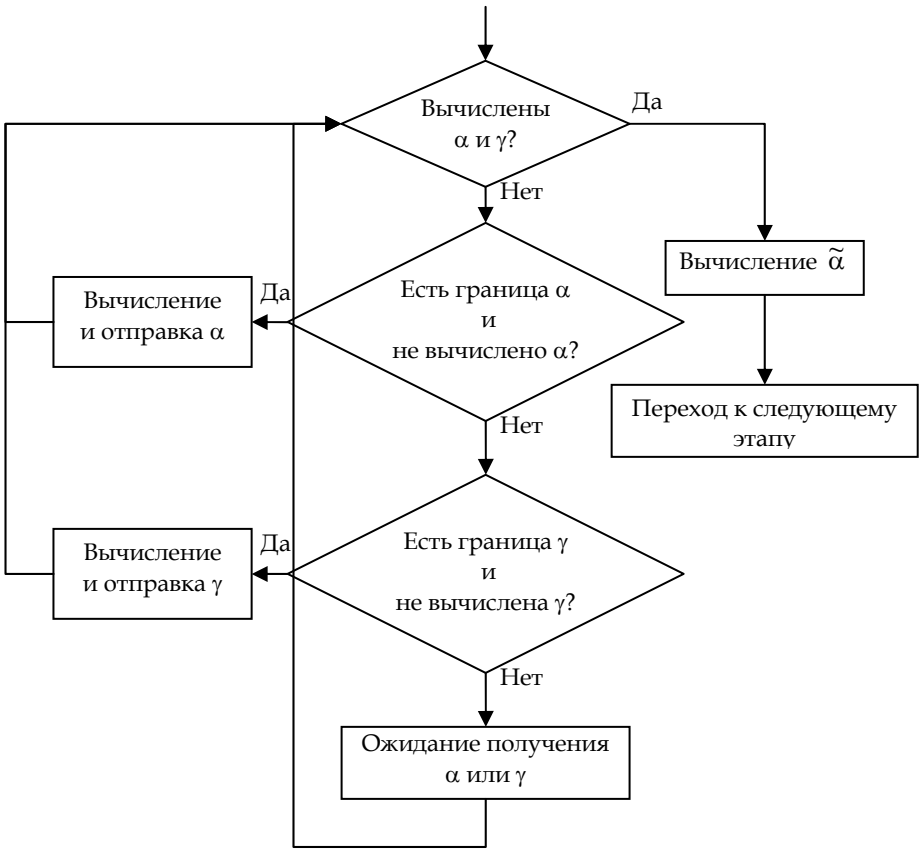

Рис. 3. Схема работы первой половины α-процесса

Данная схема (рис. 3) подразумевает линейную топологию связи процессоров в системе. Первый и последний процессор имеют граничные условия для α и γ соответственно, поэтому они сразу начинают работу. В случае процессоров с одинаковой производительностью вычисления прогоночных коэффициентов α и γ первыми заканчивают процессоры, находящиеся в середине топологической схемы, они же первыми начинают вычисление коэффициентов $\tilde{\alpha}$. К моменту, когда первый и последний процессор закончат вычисление $\tilde{\alpha}$, остальные процессоры будут ожидать начала следующего этапа.

Сделаем оценку для ускорения и эффективности параллельной реализации первой половины α-процесса. Рассмотрим вычислительный процесс в системе с четным количеством процессоров одинаковой производительности. Отметим, что четность количества процессоров не сильно сказывается на общей оценке, а лишь упрощает дальнейшие рассуждения. Как было сказано выше, для распараллеливания исполь-

зуем геометрическое деление области задачи по координате x с перекрытиями, что позволяют получить граничные значения прогоночных коэффициентов для области каждого процессора. Разобьем нашу область на n равных частей (допустим, что такое разбиение возможно), где n — количество вычислительных единиц в системе. Затем расширим область данных каждого процессора на b элементов влево и вправо за исключением первого и последнего процессоров. Величину b назовем радиусом перекрытия. Введем дополнительное значение

$$a = \frac{N}{n} - 2b, \tag{3}$$

где N — количество разностных узлов по оси x. Для всех процессоров, кроме первого и последнего, величина a равна размеру не перекрывающейся области. Для первого и второго процессоров размер не перекрывающейся области равен $a + b$. Отметим, что будем вести рассуждения с максимальной погрешностью в один элемент сетки для величин a и b, что не повлияет на нашу оценку при больших размерах задачи по оси x. Возьмем за единицу время, необходимое на вычисление одного прогоночного коэффициента α, второй — γ — потребует $\frac{n}{2}(a+2b)+b$ единиц времени. Тогда до момента, когда первый процессор закончит вычисление γ, а последний — вычисления α (остальные процессоры в это время будут заняты вычислением $\tilde{\alpha}$), пройдет еще $\frac{n}{2}(a+2b)+b$ единиц времени, а через $(a + 3b)$ единиц времени крайние процессоры завершат расчет $\tilde{\alpha}$. В сумме на вычисление прогоночных коэффициентов α, γ, $\tilde{\alpha}$ потребуется

$$n(a + 2b) + a + 5b \tag{4}$$

единиц времени. Подставляя значение (3) в выражение (4), получим формулу для относительного времени, необходимого для расчета прогоночных коэффициентов на одном слое по y:

$$T_n = N(1 + \frac{1}{n}) + 3b.$$

Следует добавить, что в данном алгоритме величина радиуса пересечения не оказывает влияния на сходимость прогоночных коэффициентов, так как оценка сходимости производится по всему массиву коэффициентов. Следовательно, имеет смысл минимизировать величину b, взяв ее равной 1. Для оценки ускорения и эффективности при больших значениях N величиной b можно пренебречь. Получим новую упрощенную формулу:

$$T_n = N(1 + \frac{1}{n}).$$

Для нахождения эталонного времени вычисления прогоночных коэффициентов на одном процессоре полученная формула непримени-

ма, однако вывод ее очевиден, время расчета складывается из времени определения коэффициентов $\alpha, \gamma, \tilde{\alpha}$, т.е.

$$T_1 = 3N.$$

Тогда теоретическое ускорение параллельных вычислений для прогоночных коэффициентов составит

$$R_n = 3\frac{n}{n+1}, \tag{5}$$

а теоретическая эффективность

$$P_n = \frac{3}{n+1}. \tag{6}$$

Формулы (5) и (6) имеют смысл при $n \geqslant 2$ и наиболее точны для четных значений n. Из формулы (5) видно, что теоретическое ускорение для α- и β-процессов не превосходит 3 и стремится к 3 при увеличении n. Максимальная теоретическая эффективность достигается при $n = 2$ и уменьшается при увеличении n. К сожалению, α- и β-процессы составляют основу алгоритма и добиться более высокого ускорения тяжело. Отметим, что достаточно ресурсоемкие этапы III.4 и III.5 позволяют увеличить это значение. Общая же оценка ускорения параллельной реализации сопряжена с большими трудностями. Это связано с неоднородностью вычислительного процесса, однако суммарная оценка эффективности алгоритма может быть получена экспериментальным путем. Для этого рассмотрим следующее уравнение теплопроводности:

$$u_t = \Delta u + f, \quad u\big|_{t=0} = 0,$$

$$f = \frac{4}{1+\sqrt{x^2+y^2}} - \frac{3}{1+\sqrt{(x-0{,}5)^2+(y-0{,}5)^2}}.$$

В качестве граничных условий взято $u\big|_{(x,y)\in G} = 0$, что соответствует бесконечно мощному холодильнику на границе области. Задача была решена (рис. 4) на сетке разного размера и с разным числом шагов по времени: 1) 500×500, 20 шагов по t; 2) 2000×500, 10 шагов по t.

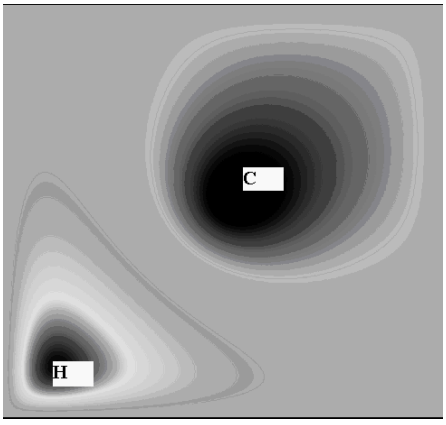

Рис. 4. Пример распределения тепла (H — горячая область, C — холодная)

Проведенные вычислительные эксперименты на одном процессоре показали высокую точность совпадения аналитического и численного решения, что позволило перейти к оценке эффективности распараллеливания. Эффективность работы программы зависит в первую очередь от размера задачи по оси x. Это связано с горизонтальным разбиением области данных. Была проведена серия тестов на одном— четырех процессорах по три испытания для каждого размера задачи. В первом варианте расчётов (рис. 5) время решения не сильно отличается при изменении числа процессоров и даже превышает эталонное в случае четырех процессоров, а наилучший результат достигается в случае двух процессоров. Это связано с тем, что при таком горизонтальном размере задачи каждый процессор загружен не на полную мощность, поэтому затраты на вычисление прогоночных коэффициентов не покрывают расходы на пересылку граничных значений.

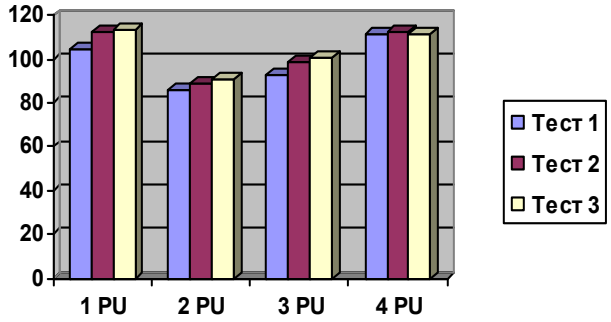

Рис. 5. Результаты задачи размером 500×500, 20 шагов по t

Однако во втором случае (рис. 6) ситуация кардинально меняется. Процессоры оказываются достаточно загруженными, и при переходе от одного процессора к двум время вычислений уменьшается почти в два раза. При переходе к четырем процессорам скачок менее заметен, в связи с тем что нагрузка на процессоры ослабевает.

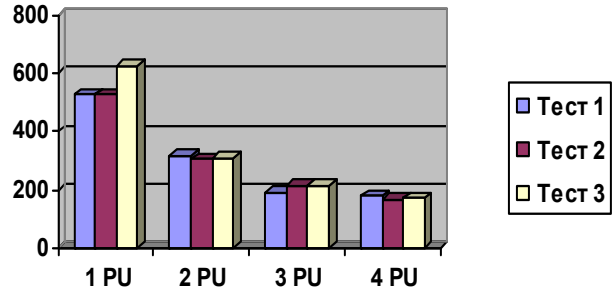

Рис. 6. Результаты задачи размером 2000×500, 10 шагов по t

На сводной диаграмме (рис. 7) наглядно изображено изменение среднего времени вычисления при увеличении числа процессоров.

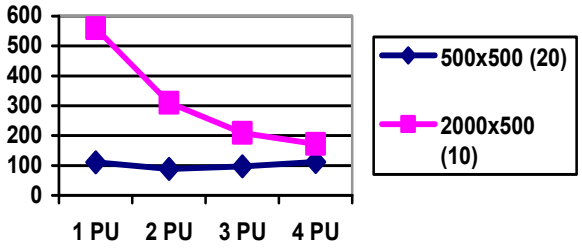

Рис. 7. Среднее время вычислений.

Из полученных результатов можно получить оценку для ускорения и эффективности работы алгоритма в многопроцессорной системе.

Как уже было отмечено выше, в случае первой задачи мы наблюдаем (рис. 8) максимальное ускорение при использовании двух процессоров, и при дальнейшем увеличении числа процессоров ускорение падает. В случае второй задачи ускорение растет с увеличением числа задействованных процессоров, но можно заметить, что при переходе к четырем процессорам темпы роста уменьшаются. Из диаграммы эффективности (рис. 9) можно сделать выводы об использовании ресурсов системы, предполагая, что программа задействует ресурсы максимально эффективно при выполнении на одном процессоре.

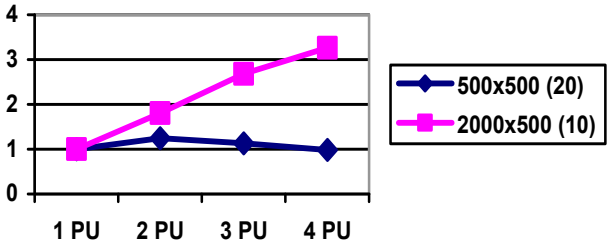

Рис. 8. Ускорение вычислений

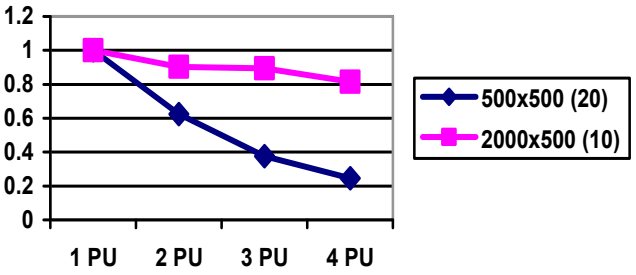

Рис. 9. Эффективность вычислений

Эффективность вычислений падает в обоих случаях, что связано с уменьшением загрузки процессоров, однако в случае второй задачи темпы падения эффективности заметно меньше.

Заключение

Задачи математического моделирования требуют больших вычислительных затрат. Эффективно увеличить производительность вычислительной системы можно только путем увеличения числа вычислительных единиц. Однако это требует модификации исходного алгоритма с учетом новой специфики системы. Для рекурсивных алгоритмов реализация параллельного алгоритма затруднена, в особенности для систем с распределенной памятью. Но даже в этом случае использование многопроцессорной системы может уменьшить время решения задачи в разы, в особенности для больших задач. В данной работе был реализован многопроцессорный вариант для α–β-итерационного метода, основу которого составляют рекуррентные вычисления, проведена теоретическая оценка для ускорения и эффективности параллельной реализации. Практические результаты исследования показывают, что даже в случае сложного итерационного алгоритма можно получить заметный выигрыш в скорости вычислений за счет использования многопроцессорных систем.

Работа выполнена при поддержке гранта РФФИ № 09-01-00628-А.

Список литературы

1. *Четверушкин Б. Н.* Математическое моделирование задач динамики излучающего газа. М., 1985.

2. *Самарский А. А., Николаев Е. С.* Методы решения сеточных уравнений. М., 1978.

3. *Четверушкин Б. Н.* Кинетически-согласованные схемы в газовой динамике: новая модель вязкого газа, алгоритмы, параллельная реализация, приложения. М., 1999.

4. *Тихонов А. Н., Самарский А. А.* Уравнения математической физики. М., 1999.

5. *Клевцур С. В., Латышев К. С., Четверушкин Б. Н.* Циклический вариант α–β-итерационного алгоритма // Дифференциальные уравнения. 1988. Т. 24. №7. С. 1213—1218.

6. *Ишанов С. А., Клевцур С. В., Латышев К. С.* Алгоритм α–β-итераций в задачах моделирования ионосферной плазмы // Матем. моделирование. 2009. Т. 21. №1. С. 33—45.

Об авторах

С. А. Ишанов — канд. физ.-мат. наук, проф., РГУ им. И. Канта, e-mail: math@dekan. albertina. ru.

С. В. Клевцур — канд. физ.-мат. наук, доц., РГУ им. И. Канта.

К. С. Латышев — д-р физ.-мат. наук, проф., РГУ им. И. Канта.

Д. И. Пялов — асп., РГУ им. И. Канта.

Authors

Dr S. A. Ishanov — professor, IKSUR, e-mail: math@dekan. albertina. ru.

Dr S. V. Klevtsur — assistant professor, IKSUR.

Professor K. S. Latyshev — IKSUR.

D. I. Pylov — PhD student, IKSUR.