# АНАЛИЗ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ЧИСЛЕННОГО РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ ИТЕРАЦИОННЫМИ МЕТОДАМИ

**Дмитриева О.А.**
Кафедра ПМиИ ДонГТУ
E-mail: dmitriv@r5.dgtu.donetsk.ua

***Abstract***

***Dmitrijeva O. The parallel algorithms analysis of the linear equations systems the numerical decision by iterative methods.*** *This article is devouted to the problem of the decision of the linear algebraic equations systems on computing structures of a type SIMD. The estimations of labour input and efficiency of the decision of systems by iterative methods Jacoby and Gauss-Zejdel are resulted. The information structure of algorithms is submitted by the graph of influence.*

## *Введение*

Расширение возможностей в конструировании вычислительной техники всегда оказывало влияние на развитие вычислительной математики - в первую очередь численных методов и численного программного обеспечения. В этой связи решение больших систем линейных уравнений принадлежит к кругу задач, для которых применение новейшей высокопроизводительной вычислительной техники дает максимальную выгоду. Идею конструирования эффективных алгоритмов для реализации решения систем линейных уравнений можно развить дальше в таком направлении, чтобы реализация этих алгоритмов разбивалась на каком-то достаточно продолжительном этапе на взаимно-независимые вычисления. Это обстоятельство может служить естественной основой "распараллеливания " алгоритмов на ЭВМ со специальной организацией вычислений. В качестве аппарата, позволяющего разбивать последовательные алгоритмы решения линейных систем на участки, вычисления в которых проводятся одновременно, можно использовать теорию расщепления информационных потоков В. В. Воеводина [1].

## *1. Реализация метода Якоби на SIMD структурах*

Пусть методом Якоби решается система линейных алгебраических уравнений большой размерности вида $Ax = f$ , (1)

где A - матрица коэффициентов системы $A = [a_{i,j}]$, i,j = 1,2,...,m, имеющая обратную матрицу, $x = (x_1, x_2, \ldots, x_m)^t$ - вектор неизвестных, $f = (f_1, f_2, \ldots, f_m)^t$ - вектор свободных членов. Для решения системы (1) методом Якоби, она преобразуется к виду

$$x_i^{(n+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j - \sum_{j=i+1}^{m} \frac{a_{ij}}{a_{ii}} x_j + \frac{f_j}{a_{ii}} \quad , \quad где \quad i = 1,2,\ldots m \qquad (2)$$

Или в векторной записи

$$Dx^{n+1} + (A_1 + A_2)x^n = f \quad , \quad откуда \quad x^{n+1} = D^{-1}f - D^{-1}(A_1 + A_2)x^n \qquad (3)$$

Матрицы D, $A_1$,$A_2$ - соответственно диагональная, нижняя треугольная и верхняя треугольная матрицы, причем $D + A_1 + A_2 = A$ (4)

Окончание итерационного процесса определяется либо заданием максимального числа итераций, либо задается условием

$$\max_{1\le i\le 14} |x_i^{n+1} - x_i^n| < \varepsilon \tag{5}$$

Для иллюстрации работы метода удобно ввести следующие обозначения. Пусть

$$G = -D^{-1}(A_1 + A_2) \quad и \quad B = D^{-1} f \tag{6}$$

или

$$G = \begin{bmatrix} 0 & \frac{-a_{12}}{a_{11}} & \dots & \frac{-a_{1m}}{a_{11}} \\ \frac{-a_{21}}{a_{22}} & 0 & \dots & \frac{-a_{2m}}{a_{22}} \\ \dots & \dots & \dots & \dots \\ \frac{-a_{m1}}{a_{mm}} & \frac{-a_{m2}}{a_{mm}} & \dots & 0 \end{bmatrix} \qquad B = \begin{bmatrix} \frac{f_1}{a_{11}} \\ \frac{f_2}{a_{22}} \\ \dots \\ \frac{f_m}{a_{mm}} \end{bmatrix} \tag{7}$$

Рассмотрим сначала случай, когда точно определен порядок выполнения всех операций. Построим граф влияния, поставив в соответствие его вершинам выполняемые операции и соединив вершины дугами, если для соответствующих операций результат выполнения одной из них влияет на результат смежной операции. Подробнее правила построения изложены в [1].

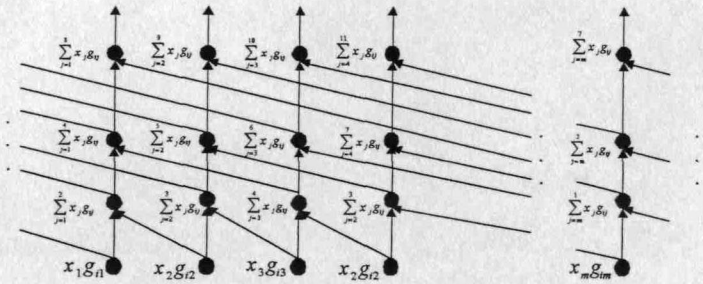

Рис.1. Граф влияния метода Якоби

С помощью графов влияния решается основная задача исследования структуры алгоритмов и программ применительно к параллельным вычислительным системам, состоящая в отыскании множеств операций, которые можно выполнять параллельно.

Граф влияния, изображенный на рис. 1, демонстрирует получение решения системы линейных уравнений на одной итерации. Модель, на которую ориентируется решение, имеет следующие особенности: используется процессор SIMD-структуры с квадратной сеткой, содержащей m×m процессорных элементов (ПЭ); каждый процессорный элемент может выполнить любую арифметическую операцию за один такт; временные затраты, связанные с обращением к запоминающему устройству, отсутствуют. Для эффективной работы метода Якоби в каждый процессорный элемент, имеющий индексы $i,j$, пересылается соответствующий элемент матрицы $G = [g_{i,j}]$. Из

вектора неизвестных $x$ формируется новая матрица U (8), элементы которой также пересылаются в соответствующие процессорные элементы. Таким образом, в каждом процессорном элементе с индексами $i,j$ оказывается соответствующее значение $g_{i,j}$. и элемент матрицы $u_j$ ( элементы $u_j$ одинаковы во всех строках). Первоначально выполняется умножение $g_{i,j}$. * $u_j$ во всех процессорных элементах. Затем осуществляется одиночный циклический сдвиг полученных значений и сложение содержимого процессорных элементов. Количество позиций, на которое производится очередной сдвиг, определяется элементами следующего ряда: $2^0, 2^1, 2^2, \ldots, 2^{k-1}$, где $k =$ ближайшее целое сверху $[\log_2 m]$.

Легко видеть, что на такой сетке в каждой ячейке $i$-ой строки получается новое значение $x_i^{n+1}$. Вычисление всех значений $x_i^{n+1}$ по строкам осуществляется параллельно с использованием алгоритма сдваивания. Таким образом, время получения значений вектора неизвестных $x^1$ определяется временем, затрачиваемым на одно умножение $t_{умн}$, суммой времен, необходимых для осуществления сдвигов и сложения сдваиванием $\sum_{s=0}^{k-1}(t_{сл} + 2^s * t_{сдв})$, а также временем $t_{сл}$, затрачиваемым на добавление в каждый процессорный элемент, находящийся в $i$-ой строке, элемента вектора $B = [b_i]$. Но необходимо учесть, что это время характеризует только нахождение значений вектора неизвестных на первой итерации. Это обусловлено тем, что значения, получающиеся в сетке процессорных элементов, расположены не в том порядке, как при изначальной засылке (8), а именно, следующим образом

$$U = \begin{bmatrix} x_1 & x_2 & \ldots & x_m \\ x_1 & x_2 & \ldots & x_m \\ \ldots & \ldots & \ldots & \ldots \\ x_1 & x_2 & \ldots & x_m \end{bmatrix} \quad (8) \qquad U = \begin{bmatrix} x_1 & x_1 & \ldots & x_1 \\ x_2 & x_2 & \ldots & x_2 \\ \ldots & \ldots & \ldots & \ldots \\ x_m & x_m & \ldots & x_m \end{bmatrix} \quad (9)$$

Для того, чтобы выполнить следующую итерацию, нужно потратить какое-то время на переупорядочивание элементов, для того, чтобы привести их к виду (8). На это потребуется $m-1$ циклических сдвигов по столбцам ( в первом -0, во втором 1, ..., в $m$ столбце необходимо выполнить - $2^{k-1}$ сдвигов), и столько же по строкам. Итого дополнительно потребуется времени $2*(m-1)*t_{сдв}$ . Необходимость перехода на вычисление следующей итерации осуществляется путем оценки абсолютной величины разности (5). На это потребуется время, характеризуемое одним вычитанием $t_{выч}$, $2*\sum_{s=0}^{k-1}(t_{сравн} + 2^s * t_{сдв})$ сравнениями и сдвигами сначала в каждой строке процессорных элементов, а затем в столбце, состоящем из первых элементов всех строк ( если сравнения выполняются с помощью алгоритма сдваивания). Тогда можно окончательно записать, что время, затрачиваемое на выполнение одной итерации

$$T_x^{(n+1)} = t_{умн} + \sum_{s=0}^{k-1}(t_{сл} + 2^s * t_{сдв}) + t_{сл} + 2*(m-1)*t_{сдв} + t_{сл} + 2*\sum_{s=0}^{k-1}(t_{сравн} + 2^s * t_{сдв}) \quad (10)$$

Используя систолический алгоритм [2], можно несколько уменьшить время, затрачиваемое на сдвиги при приведении матрицы U к исходному виду после каждой итерации. Достаточно будет провести циклические сдвиги только по столбцам. Восстановление первоначального вида матрицы неизвестных можно осуществить за время $(m-1)*t_{сдв}$, а не за $2*(m-1)*t_{сдв}$, как ранее. Тогда общее время, затрачиваемое на одну итерацию

$$T_x^{(n+1)} = t_{умн} + 2(m-1)t_{сдв} + 3t_{сл}(\log_2 m + 1) \quad (11)$$

Для оценок качества рассмотренного алгоритма вычисления очередной итерации используем наиболее распространенные критерии [3]: коэффициент ускорения $S_p(N) = \frac{T_1(N)}{T_p(N)}$ и коэффициент эффективности $E_p(N) = \frac{S_p(N)}{p} \le 1$, где $T_1(N)$ - время вычисления вектора $X^{(n+1)}$ на однопроцессорной ЭВМ с быстродействием арифметического процессора и объемом ОЗУ, равным суммарному объему всех ЗУ процессорных элементов, и с необходимым числом внешних устройств, имеющих скорости обмена такие же, как в многопроцессорной вычислительной системе типа SIMD. При реализации одной итерации метода Якоби на однопроцессорной ЭВМ

$$T_1 = m*[(m-1)t_{умн} + (m-1)t_{сл} + t_{сл} + t_{вы+} + t_{сравн}] = m*[(m-1)(t_{умн} + t_{сл}) + 3t_{сл}] \quad (12)$$

Чтобы оценить параллелизм алгоритма, примем, как и в [3], что $t_{оп} = t_{умн} = t_{сл} = 10t_{сдв}$. Тогда

$$S_{m^2} = \frac{m*[(m-1)2t_{оп} + 3t_{оп}]}{0.2(m-1)t_{оп} + (3\log_2 m + 4)t_{оп}} \approx 10*m \quad (13)$$

Эффективность метода Якоби при решении на SIMD-структуре

$$E_{m^2} = \frac{10*m}{m^2} \approx \frac{10}{m} \quad (14)$$

Полученные результаты значительно отличаются от характеристик потенциального параллелизма метода Якоби, приведенных в [4] (ускорение $\approx m^2$, эффективность $\approx 1$), поскольку при расчете этих характеристик не учитывались времена, затрачиваемые на сдвиги, хотя, как оказалось, эти времена существенно влияют на характеристики параллелизма.

## 2. Реализация метода Гаусса – Зейделя на SIMD-структурах

Рассмотрим решение системы (1) методом Гаусса-Зейделя на той же вычислительной структуре и при отмеченных ранее соглашениях. Система при этом приводится к виду [5]

$$(D + A_1)x^{n+1} + A_2 x^n = f \quad , \quad откуда \quad x^{n+1} = (D + A_1)^{-1} f - (D + A_1)^{-1} A_2 x^n \quad (15)$$

Решение рассмотрим уже с применением систолического алгоритма умножения матриц, благодаря которому можно ускорить формирование матрицы U. Первоначальные результаты получим, решая задачу как и в методе Якоби, но для матриц вида (16,17). В результате выполненных преобразований и решения формируется значение только одной переменной $x_1^{(n+1)}$. Все остальные значения переменных требуют доопределения. Для этого на каждом шаге при вычислении очередного значения $x_i^{(n+1)}$ необходимо выполнять следующую последовательность действий: *i-1* строку сохранить, строки с индексами $\le$ *i-1* перезаписать в соответствующие процессорные элементы, выполнить умножение переменных в строках, предшествующих *i*-той на соответствующие коэффициенты, осуществить сдвиги и сложение сдваиванием.

$$G=\begin{bmatrix} 0 & \frac{-a_{12}}{a_{11}} & \frac{-a_{13}}{a_{11}} & \dots & \frac{-a_{1m-1}}{a_{11}} & \frac{-a_{1m}}{a_{11}} \\ 0 & \frac{-a_{23}}{a_{22}} & \frac{-a_{24}}{a_{22}} & \dots & \frac{-a_{2m}}{a_{22}} & 0 \\ 0 & \frac{-a_{34}}{a_{33}} & \frac{-a_{35}}{a_{33}} & \dots & 0 & 0 \\ \dots & & & \dots & & \dots \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \quad (16) \qquad U=\begin{bmatrix} x_1 & x_2 & \dots & x_m \\ x_2 & x_3 & \dots & x_1 \\ \dots & \dots & \dots & \dots \\ x_m & x_1 & \dots & x_{m-1} \end{bmatrix} \quad (17)$$

Трудоемкость метода Гаусса- Зейделя при его реализации на SIMD-структурах оценивается на начальном этапе : $t_{умн}+t_{сл}\log_2 m+t_{сдв}(m-1)$. На этапе доопределения переменных $(m-1)t_{умн}+\log_2 m\,(t_{сл}+t_{сдв})$, на этапе преобразования матрицы неизвестных U $t_{сдв}(m-1)$ и на этапе проверки окончания итерационного процесса $t_{выч}+t_{сравн}\log_2 m+t_{сдв}(m-1)$. Таким образом, необходимое количество операций будет оцениваться как

$$T_x^{(m+1)}=mt_{умн}+2t_{сл}\log_2 m+3(m-1)t_{сдв}+\log_2 m\,(t_{сл}+t_{сдв})+t_{выч} \quad (18)$$

Трудоемкость метода Гаусса- Зейделя при решении на однопроцессорной ЭВМ будет такой же, как и у метода Якоби (12). Оценим параллелизм, используя приведенные ранее соотношения. Тогда ускорение метода Зейделя

$$S_{m^2}=\frac{m*\left[2(m-1)t_{оп}+3t_{оп}\right]}{(m+1)t_{оп}+3.1*\log_2 m*t_{оп}+0.3*(m-1)*t_{оп}}\approx 2m \quad (19)$$

Эффективность метода Зейделя при решении на SIMD-структуре

$$E_{m^2}=\frac{2m}{m^2}\approx\frac{1}{m} \quad (20)$$

Полученные значения существенно отличаются от оценок потенциального параллелизма ввиду указанных ранее причин. Кроме того, показатели ускорения и эффективности метода Гаусса- Зейделя при решении на $m^2$ процессорных элементах гораздо хуже, чем у метода Якоби .

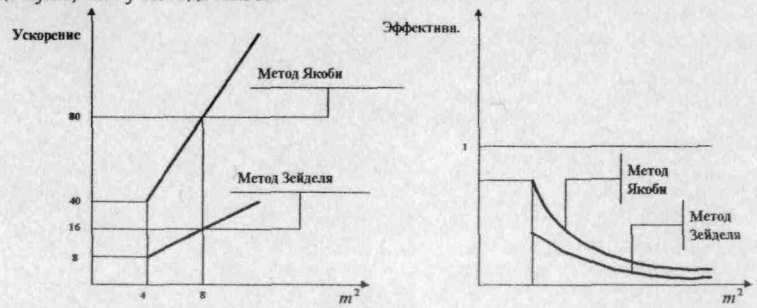


Рис.2 Сравнительные характеристики ускорения и эффективности итерационных методов на одной итерации

Это обусловлено тем, что все приведенные выше исследования оценивали только одну итерацию решения. Однако следует учитывать то, что скорость сходимости

метода Зейделя значительно превосходит сходимость метода Якоби. Количество же самих итераций существенно зависит от вида исходной матрицы $A$. Так, для плохо обусловленных матриц при оптимальных выборах параметров метод Гаусса- Зейделя асимптотически сходится в $2\sqrt{p}$ раз быстрее метода Якоби [4], где $p(A)$ - число обусловленности симметричной и положительно определенной матрицы $A$

$$p(A) = \frac{\beta(A)}{\alpha(A)} >> 1 \qquad (21)$$

$\alpha$ - минимальное, а $\beta$ - максимальное собственные числа матрицы $A$. Поэтому, если для исходной матрицы А значение $p(A)$ будет больше отношения $m^2/4$, то ускорение, полученное при решении задачи методом Зейделя на SIMD-структуре, будет превосходить ускорение метода Якоби.

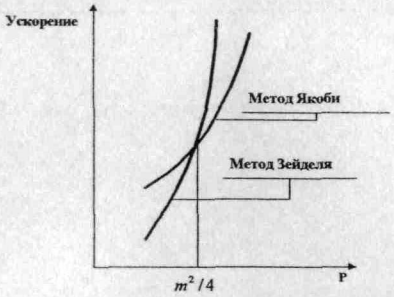


Рис. 3 Сравнительные характеристики ускорения итерационных методов

### *Заключение*

Таким образом, при анализе решения систем линейных уравнений на параллельных вычислительных структурах типа SIMD был проведен сравнительный анализ эффективности алгоритмов и возможности их распараллеливания; рассчитаны характеристики параллельных итерационных алгоритмов при нескольких характерных соотношениях между числом неизвестных и числом процессорных элементовов (что позволило выявить тенденции в изменении эффективности рассмотренных параллельных итерационных методов с ростом числа процессорных элементов); выявлено предельное распараллеливание итерационных алгоритмов.

### *Литература*

1. Воеводин В.В. Информационная структура алгоритмов.- М.: Изд-во МГУ, 1997.- 139с.
2. Фельдман Л.П., Параллельные алгоритмы моделирования динамических систем, описываемых обыкновенными дифференциальными уравнениями. //Научные труды ДонГТУ. Серия: Информатика, кибернетика и вычислительная техника, (ИКВТ-99) вып. 6.- Донецк: ДонГТУ, 1999.-с. 24-29.
3. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем: Пер. с англ. - М.: Мир, 1991. - 367с.
4. Марчук Г.И. Методы вычислительной математики. - М.: Наука. Гл. ред. физ.-мат. лит., 1989.- 608 с.
5. Самарский А.А., Гулин А.В. Численные методы. - М.: Наука. Гл. ред. физ.-мат. лит., 1989.- 432 с.