УДК 004.382

А.Л. Масюк, аспирант
Донецкий национальный технический университет «ДонНТУ»
(Украина, 85300, г. Покровск, пл. Шибанкова, 2,
e-mail: ars.masiuk@gmail.com)

Гибридный параллельный решатель уравнений методом прямых для задачи нормального воздухораспределения

Структура параллельного решателя уравнений расширена для задачи нормального воздухораспределения в шахтных вентиляционных сетях посредством объединения MIMD структуры со встроенными SIMD средствами современных процессоров. На основе SSE и AVX расширений усовершенствован алгоритм решателя уравнений методом прямых. Применение SIMD составляющей в полученной гибридной MIMD+SIMD структуре позволяет в несколько раз уменьшить число вычислительных итераций и тем самым ускорить процесс моделирования.

Ключевые слова: параллельный алгоритм, шахтная вентиляционная сеть, нормальное воздухораспределение, метод прямых, интегрированные средства SIMD.

Структуру паралельного розв'язувача рівнянь розширено для задачі нормального розподілу повітря в шахтних вентиляційних мережах за допомогою об'єднання MIMD структури з вбудованими SIMD засобами сучасних процесорів. На основі SSE і AVX розширень вдосконалено алгоритм вирішувача рівнянь методом прямих. Застосування SIMD у складі отриманої гібридної MIMD+SIMD структури дозволяє в декілька разів зменшити кількість обчислювальних ітерацій і тим самим прискорити процес моделювання.

Ключові слова: паралельний алгоритм, шахтна вентиляційна мережа, нормальний розподіл повітря, метод прямих, інтегровані засоби SIMD.

Исследования, связанные с решением уравнений для задачи нормального воздухораспределения в шахтной вентиляционной сети (ШВС) основаны на использовании параллельных вычислительных систем SIMD (Single Instruction Multiple Data) и MIMD (Multiple Instruction Multiple Data) архитектуры [1—4]. Разработаны алгоритмы, реализованные для языка Parallaxis с учетом SIMD архитектуры кластера, а также алгоритмы, основанные на стандарте MPI для выполнения на MIMD архитектурах [1, 5, 6].

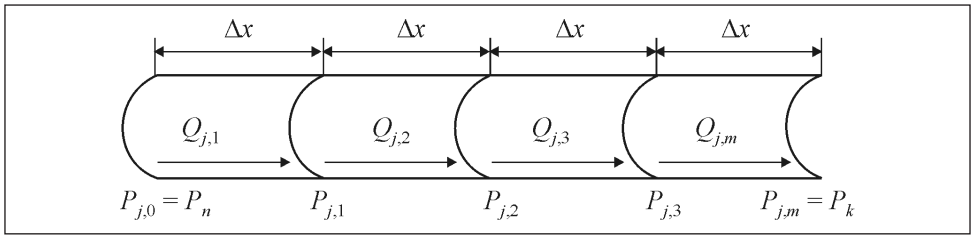

Рис. 1. Схема разбиения ветви на отрезки

Основная идея параллельного MIMD метода заключается в том, что каждая ветвь ШВС, состоящей из M ветвей, моделируется как отдельный процесс на каждом шаге итерации. В идеале каждый процесс должен быть запущен на отдельном физическом процессоре, что достигается при числе доступных процессоров $N = M$. В случае, когда $N < M$ (т.е. физических процессоров меньше, чем ветвей), происходит девиртуализация модели [3] (при этом один и тот же процессор последовательно занимается моделированием нескольких назначенных ему ветвей). Таким образом, каждая итерация выполняется параллельно на N процессорах. По окончании итерации управляющий процесс собирает результаты всех моделирующих процессов и принимает решение об окончании моделирования либо о подготовке и запуске последующей итерации.

Предложенный в работах [5, 7] метод прямых предполагает логическую декомпозицию каждой ветви на отдельные элементы. Ветвь Q_i разбивается на m отрезков с постоянным шагом дискретизации dx (рис. 1). Граничные значения давлений в начале и конце элемента обозначим $P_{i,\,j\,(0\ldots m)}$, а давление атмосферы — P_n, при этом $P_m = f(Q)$. Для каждого отрезка $Q_{i,j}$ решается система дифференциальных уравнений

$$-\frac{P_{i,\,j} - P_{i,\,j+1}}{\Delta x} = r_{i,\,j} Q_{i,\,j}^2 + \frac{\rho}{S_{i,\,j}} \frac{dQ_{i,\,j}}{dt},$$

$$-\frac{dP_{i,\,j+1}}{dt} = \frac{\rho a^2}{S_{i,\,j}} \frac{Q_{i,\,j} - Q_{i,\,j+1}}{\Delta x}, \qquad (1)$$

где r_i — удельное аэродинамическое сопротивление воздуховода; ρ — плотность воздуха; S_i — площадь сечения воздуховода; a —скорость звука в воздухе; j — индекс отрезка ветви Q_i. Полученные для каждой ветви результаты возвращаются в основной процесс, формируя общий результат по ветви, и при необходимости осуществляется запуск следующей итерации моделирования. Общая структура такого MIMD решателя представлена на рис. 2.

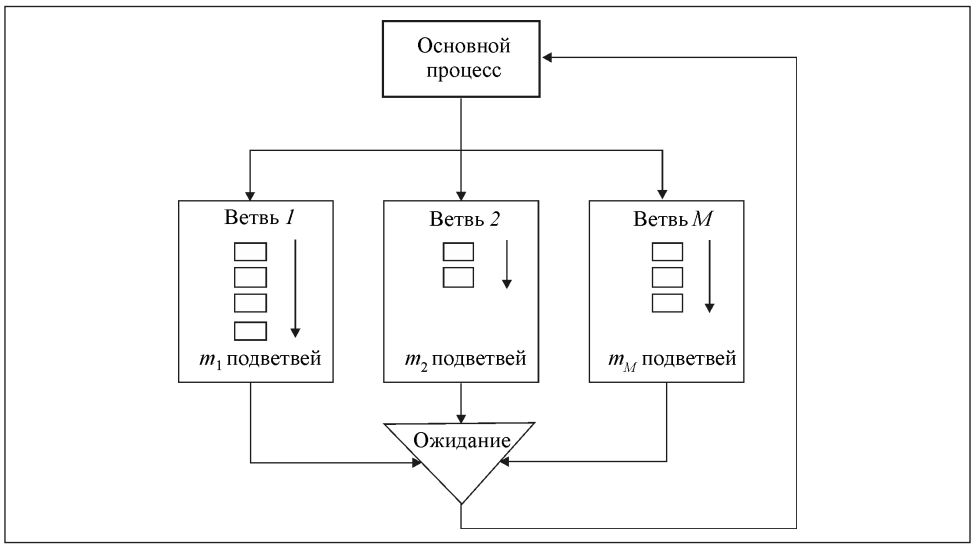

Рис. 2. Структура MIMD решателя уравнений

Основной недостаток данного подхода состоит в том, что время моделирования каждой ветви зависит от числа подветвей в ней (как правило, число отрезков прямо пропорционально длине L_i ветви). Следовательно, общее время моделирования одной итерации на N процессорах будет равно максимальному времени, затраченному каждым процессором в отдельности. При этом возникает проблема простоя процессоров, уже закончивших свою работу (например, ветви *2* и *M* на рис. 2). Данная проблема порождает целый класс задач, связанных с оптимальной балансировкой загрузки процессоров [8], решения которых часто являются нетривиальными и требуют значительных временных ресурсов и специальных знаний.

Предлагается гибридная схема MIMD+SIMD решателя уравнений, ориентированного на встроенные параллельные средства современных процессоров (SSE, AVX). При использовании гибридной схемы решение уравнений для каждого отрезка ветви предполагается выполнять, насколько это возможно, с помощью параллельного SIMD алгоритма, оптимизированного под встроенную SIMD архитектуру. При этом ускоряется расчет ветвей и минимизируется время, затраченное каждым процессорным элементом на ожидание остальных. В свою очередь, это приведет к уменьшению времени моделирования каждой итерации, а в итоге — к общему ускорению процесса моделирования. Гибридная структура решателя уравнений приведена на рис. 3.

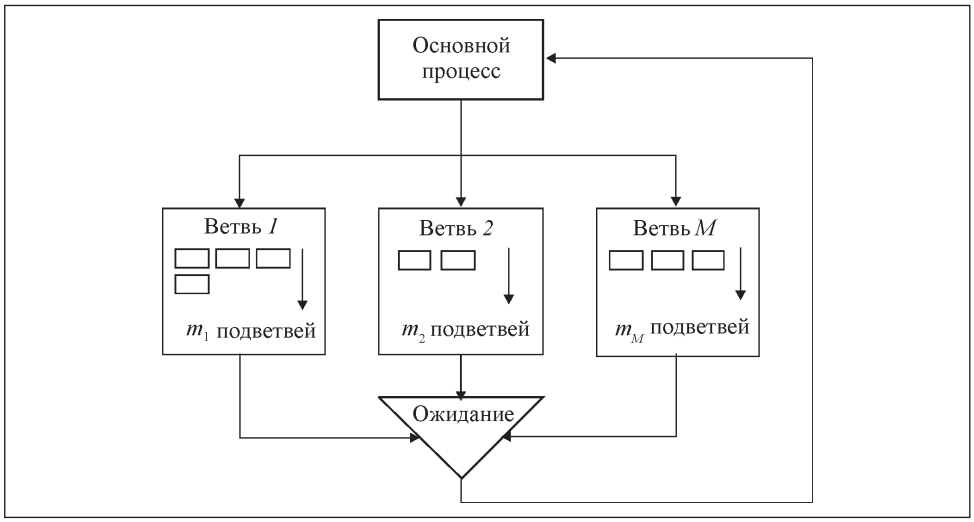

Рис. 3. Структура гибридного MIMD-SIMD решателя уравнений

Решение системы уравнений (1) относительно каждого отрезка $Q_{i,j}$ с помощью матричного решателя уравнений представлено в [7] в следующем виде:

$$\frac{dQ_{i,j}}{dt} = a*(P_{i,j-1} - P_{i,j}) - b*(Q_{i,j}*|Q_{i,j}|),$$
$$\frac{dP_{i,j}}{dt} = g*(Q_{i,j} - Q_{i,j+1}), \qquad (2)$$

где a, b, g — аэродинамические параметры, вычисляемые для всей ветви Q_i в пределах текущей итерации,

$$a = \frac{S}{\rho\Delta x}, \quad b = \frac{rS}{\rho}, \quad g = \frac{\rho a 2}{S\Delta x}.$$

Для $j = 1$ (первый отрезок) значения $dQ_{i,1}$ и $dP_{i,1}$ вычисляются с помощью следующей системы:

$$\frac{dQ_{i,1}}{dt} = a*(P_{i\text{н}} - P_{i,1}) - b*(Q_{i,1}*|Q_{i,1}|),$$
$$\frac{dP_{i,1}}{dt} = g*(Q_{i,1} - Q_{i,2}), \qquad (3)$$

где $P_{i\text{н}}$ — начальное значение давления в ветви Q_i, которое вычисляется в процессе генерирования уравнений в зависимости от входных коммутаций ветви.

Для $j = m$ (последний отрезок) формулы для вычислений $dP_{i,m}$ генерируются в зависимости от выходных коммутаций ветви Q_i. [4, 7]

Производительность параллельной программы находится в тесной взаимосвязи с представлением обрабатываемых данных. Для SIMD программ, как правило, наиболее эффективным отображением входных и выходных данных в памяти является концепция SoA (Structure of Arrays), предполагающая последовательное размещение данных, относящихся к одному набору.

Рассмотрим размещение локальных данных алгоритма для отрезка Q_j с помощью систем уравнений (2), (3). Структура переменных, описывающих каждый отрезок Q_j, имеет вид

$$D_1 = [dQ_1, dP_1, Q_1, P_1],$$
$$D_j = [dQ_j, dP_j, Q_j, P_j],$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$D_m = [dQ_m, dP_m, Q_m, P_m],$$

где D_j — набор локальных переменных для хранения входных данных (Q_j, P_j) и результатов вычислений (dQ_j, dP_j) по каждому отрезку Q_j. Таким образом, набор переменных для проведения расчетов по всей ветви Q содержит m структур типа D. Из (2), (3) видно, что значения dQ_j и dP_j для каждого отрезка, кроме последнего (для $j = 1, ..., m - 1$), зависят от величин a, b, g, которые являются постоянными величинами в пределах одной ветви и поэтому могут быть вычислены однократно перед параллельным циклом расчетов по отрезкам, и текущих значений Q_j и P_j, а также значения P_{j-1} предыдущего и значения Q_{j+1} следующего отрезка.

Первый отрезок ветви описывается системой уравнений (3), в которой в качестве дополнительной переменной используется входное значение давления $P_{i\text{н}}$ для расчета начальных значений dP_1 и dQ_1. Исходя из оптимального расположения данных в памяти, можно представить значение $P_{i\text{н}}$ в качестве элемента массива с нулевым индексом: $P[0] = P_{i\text{н}}$. На практике это означает, что для массивов Q, P, dP и dQ необходимо выделить на один элемент памяти больше, и в таком случае вычисление первого отрезка тоже может быть выполнено параллельным алгоритмом (при этом элементы Q_0, dP_0, dQ_0 не будут содержать необходимых для расчетов значений и результатов).

На рис. 4 представлено возможное размещение данных в памяти при применении SoA. Данные для вычисления dQ_1 и dP_1 — в одинарных рамках, данные для вычисления dQ_3 и dP_3 — в двойных рамках.

Представим вычисления $dP_{j=1,\ldots,m-1}$ в виде таблицы операций (табл. 1), где в столбцах приведены индексы текущего вычисления j, а в строках —

$Q_0 = ?$	Q_1	Q_2	Q_3	Q_4	...	Qm
$P_0 = P_н$	P_1	P_2	P_3	P_4	...	Pm
$dQ_0 = ?$	dQ_1	dQ_2	dQ_3	dQ_4	...	dQm
$dP_0 = ?$	dP_1	dP_2	dP_3	dP_4	...	dPm

Рис. 4. Зависимости значений при вычислении dQ_1 и dP_1

Таблица 1

Команда	Q_1	Q_2	Q_3	...	Q_{m-1}	Q_m
Загрузка $Q_{j+1}, ..., Q_m$	Q_2	Q_3	Q_4	...	Q_m	—
$Q_j - Q_{j+1}$	$Q_1 - Q_2$	$Q_2 - Q_3$	$Q_3 - Q_4$	...	$Q_{m-1} - Q_m$	—
Загрузка g	g	g	g	g	g	g
$dP_{j=1,...,m-1}$	$g*(Q_1 - Q_2)$	$g*(Q_2 - Q_3)$	$g*(Q_3 - Q_4)$	...	$g*(Q_{m-1} - Q_m)$	—

параметры, операции и результаты вычислений. Из табл. 1 видно, что алгоритм вычисления $dP_{j=1,...,m-1}$ может быть эффективно имплементирован с учетом SIMD архитектуры, поскольку представляет собой последовательное выполнение инструкций над значениями, не зависящими одно от другого. Это означает, что каждое значение $dP_{j=1,...,m-1}$ может быть вычислено параллельно с остальными:

1. Загрузка в SIMD регистр R_1 значений $Q_1, ..., Q_{m-1}$.

2. Загрузка в SIMD регистр R_2 значений $Q_2, ..., Q_m$.

3. Выполнение операции = $R_1 - R_2$, после чего в SIMD регистре R_3 будут находиться значения $Q_j - Q_{j+1}$.

4. Загрузка в SIMD регистр R_4 значений g методом broadcast (распределение g по всем разрядам).

5. Выполнение операции $R_5 = R_3 * R_4$, после чего в SIMD регистре R_5 будут находиться значения $g*(Q_j - Q_{j+1})$.

6. Считывание результатов dP_j из регистра R_5 и помещение их в память результатов.

Аналогично с помощью таблицы операций может быть представлен алгоритм для вычисления $dQ_{j=1,...,m-1}$ (табл. 2). Приведенный в табл. 2 параллельный SIMD алгоритм рассчитан на вычисление отрезков ветви с первой по предпоследнюю. Уравнения для решения последнего отрезка не могут быть представлены универсальным способом, поскольку число переменных в них зависит от конкретной топологии сети [4, 7] (коммутация текущей ветви с другими объектами может быть произвольной). Учитывая

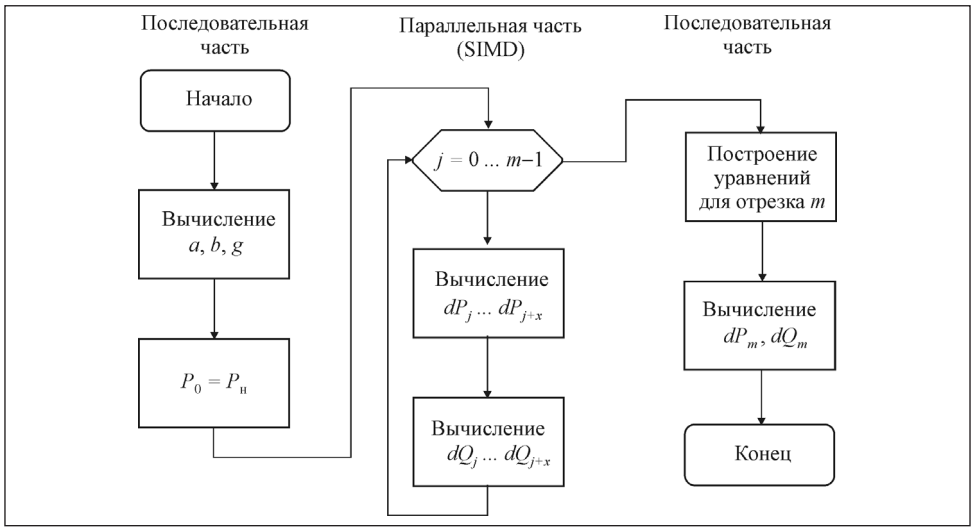

Рис. 5. Блок-схема вычисления ветви Q_i с помощью гибридного алгоритма

Таблица 2

Команда Параметр	Q_1	Q_2	Q_3	...	Q_{m-1}
Модуль $\|Q_j\|$	$\|Q_1\|$	$\|Q_2\|$	$\|Q_3\|$	...	$\|Q_{m-1}\|$
$Q_j * \|Q_j\|$	$Q_1 \|Q_1\|$	$Q_2 \|Q_2\|$	$Q_3 \|Q_3\|$	...	$Q_{m-1} \|Q_{m-1}\|$
Загрузка b	b	b	b	b	b
$b * Q_j \|Q_j\|$	$b Q_1 \|Q_1\|$	$b Q_2 \|Q_2\|$	$b Q_3 \|Q_3\|$	...	$b Q_{m-1} \|Q_{m-1}\|$
Загрузка P_{j-1}	P_0	P_1	P_2	...	P_{m-2}
Загрузка P_j	P_1	P_2	P_3	...	P_{m-1}
$P_{j-1} - P_j$	$P_0 - P_1$	$P_1 - P_2$	$P_2 - P_3$	...	$P_{m-2} - P_{m-1}$
Загрузка a	a	a	a	a	a
$a*(P_{j-1} - P_j)$	$a(P_0 - P_1)$	$a(P_1 - P_2)$	$a(P_2 - P_3)$	...	$a(P_{m-2} - P_{m-1})$
$a*(P_{j-1} - P_j) - b*Q_j^* \|Q_j\|$	$a(P_0 - P_1) - b Q_1 \|Q_1\|$	$a(P_1 - P_2) - b Q_2 \|Q_2\|$	$a(P_2 - P_3) - b Q_3 \|Q_3\|$	...	$a(P_{m-2} - P_{m-1}) - b Q_{m-1} \|Q_{m-1}\|$

данный факт, более целесообразным можно считать использование последовательных алгоритмов для генерации уравнений по последнему отрезку и для их расчета.

На рис. 5 представлена общая блок-схема вычисления отдельной ветви с помощью гибридного алгоритма. Как указано выше, последний отрезок ветви Q_i вычисляется на основе формул, полученных на этапе генерирования уравнений. Если они вычисляются с помощью последовательных

алгоритмов, то с учетом данных ограничений общее число итераций для расчета одной ветви будет $I_i = \text{ceil}((m_i - 1)/d) + 1$, где m_i — общее число подветвей ветви Q_i; d — число операндов одного SIMD регистра; ceil — функция округления значения в большую сторону до ближайшего целого. Поскольку для последовательного алгоритма число последовательных итераций I_i вычислений для расчета одной ветви равно m_i, алгоритм, основанный на SSE-командах, потребует на dI_i меньше итераций:

$$dI_i = m_i - \text{ceil}((m_i - 1)/d) - 1. \quad (4)$$

Например для числа отрезков ветви $m_i = 10$ при использовании регистров с двумя операндами (d = 2) по формуле (4) получим $dI_i = 10 - \text{ceil}(9/2) - 1 = 4$, что соответствует уменьшению числа итераций на 40 %. При $d = 4$ $dI_i = 10 - \text{ceil}(9/4) - 1 = 6$, что соответствует уменьшению числа итераций на 60 %. При значительном увеличении числа отрезков ветви ($m_i \geq 100$) можно пренебречь расчетами последнего отрезка, а также редуцировать функцию ceil. Тогда формулу (4) можно записать в виде

$$dI_i = m_i - m_i / d. \quad (5)$$

Следовательно, при использовании инструкций SSE теоретический предел уменьшения числа итераций при $d = 2$ составляет 50 %, а при $d = 4$ — 75 %, что соответствует идеальному случаю распараллеливания (без учета таких затратных по времени операций, как обращение к памяти).

При имплементации алгоритма с учетом AVX архитектуры разрядность регистров и значение d возрастают вдвое относительно SSE, соответственно число итераций может быть уменьшено на 75 % для операндов с двойной точностью (double $d = 4$) и на 87,5 % для операндов с одинарной точностью (float d = 8). При использовании наиболее современной архитектуры AVX-512 разрядность SIMD регистров увеличивается до 512 бит.

Таблица 3

Алгоритм	Float, мс	Ускорение для отрезков 100/2000	Double, мс	Ускорение для отрезков 100/2000
No SSE (plain C)	$\frac{7,7}{152}$	1/ 1	$\frac{7,7}{154}$	1/ 1
SSE	$\frac{2,1}{54}$	3,6/ 2,8	$\frac{4,1}{93}$	1,9/ 1,6
AVX	$\frac{0,9}{36}$	8,5/ 4,2	$\frac{1,8}{126}$	4,3/ 1,2

Примечание. Целевая платформа: Intel Core i3-5020U CPU @ 2.20 GHz, 4 GB SODIMM RAM @ 1600 MHz.

При этом по формуле (5) получаем максимально возможное уменьшение числа итераций: для операндов типа double (d = 8) — до 87,5 %, для операндов типа float (d = 16) — до 93,75 %.

Результаты экспериментальных исследований, выполненных для синтетических ветвей, в целом подтверждают результаты теоретических расчетов. В ряде экспериментов установлено, что на производительность алгоритма, использующего встроенные SIMD средства, значительное влияние оказывает размер и быстродействие целевого процессора, а также большое значение имеет применение дополнительных техник (таких, как loop unrolling) при написании программного кода.

В табл. 3 приведены результаты измерений времени выполнения алгоритма для сети, состоящей из 100000 ветвей, каждая из которых была разбита на 100 (над чертой) и 2000 (под чертой) отрезков. Для каждого типа операндов указано отношение времени выполнения параллельного алгоритма (SSE, AVX) к последовательному (No SSE).

Выводы

Предложенная гибридная MIMD+SIMD структура решателя уравнений для задачи нормального воздухораспределения в ШВС дает возможность уменьшить число вычислительных итераций. Разработанный параллельный алгоритм моделирования воздухораспределения в отдельных ветвях по методу прямых позволяет значительно увеличить быстродействие сравнительно с последовательным алгоритмом.

СПИСОК ЛИТЕРАТУРЫ

1. *Перерва А.А.* Генератор и решатель уравнений проблемно ориентированной параллельной моделирующей среды для сетевых объектов с сосредоточенными параметрами // Наук. праці Донецького національного технічного університету. Серія «Проблеми моделювання та автоматизації проектування» (МАП-1999). Випуск: 10. — Донецьк: ДонНТУ, 1999. — С. 164—169.
2. *Святний В.А.* Паралельне моделювання складних динамічних систем // Междунар. конф. Моделирование — 2006. — Киев, 2006. — С. 83—90.
3. *Feldmann L.P., Svjatnyj V.A., Resch M., Zeitz M.* Forschungsgebiet: parallele Simulationstechnik / DonNTU, FRTI-Werke, Reihe “Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen”. Band 9 (150). — Donezk, 2008. — S. 9—36.
4. *Святний В.А., Смагін О.М., Солонін О.М.* Методи розпаралелювання вирішувача рівнянь MIMD-моделі мережного динамічного об’єкта / Наук. праці ДонДТУ. Серія «Інформатика, кібернетика та обчислювальна техніка». Вип. 70. — Донецьк, ДонДТУ, 2003. — С. 20—30.
5. *Смагин А.Н.* Эффективность MIMD решателей уравнений аэродинамики шахтных вентиляционных сетей на базе MPI и OPENMP стандартов// Матеріали IV науково-практичної конф. Донбас-2020: наука і техніка виробництву, 27—28 травня 2008 р. — Донецьк: ДонНТУ, 2008. — С. 432—436.

6. *Гусєва Г.Б., Молдованова О.В.* MIMD-паралельний вирішувач рівнянь для мережного динамічного об'єкту з розподіленими параметрами//Наук. праці Донецького національного технічного університету. Серія «Проблеми моделювання та автоматизації проектування». — Донецьк: ДонНТУ, 2007. — С. 149—158 .
7. *Святний В.А., Молдованова О.В.* Генератор уравнений параллельной модели сетевого динамического объекта с распределенными параметрами // Наукові праці Донецького національного технічного університету. Серія «Проблеми моделювання та автоматизації проектування» (МАП-1999). Вип. 10. — Донецьк: ДонНТУ, 1999. — С. 135—141.
8. *Миков А.И., Замятина Е.Б.* Балансировка нагрузки в распределенных системах. — Национальный открытый институт «Интуит». Режим доступа: http://www.intuit.ru/department/algorithms/distras/9/1.html

Поступила 11.01.17

REFERENCES

1. Pererva, A.A. (1999), "Equation generator and solver of the problem-oriented parallel modeling environment for network objects with non-distibuted parameters", *Naukovi pratsi Donetskogo Natsionalnogo Tekhnichnogo Universytetu. Seria: "Problemy modelyuvannya ta avtomatyzatsii proektuvannya"*, Iss. 10, pp. 164-169.
2. Svjatnyj, V.A. (2006), "Parallel modeling of the complex dynamic systems", *Mezhdunarodnaya konferentsiya. Modelirovanie-2006* [International conference. Simulation], Kyiv, 2006, pp. 83-90.
3. Feldmann, L.P, Svjatnyj, V.A., Resch, M. and Zeitz, M. (2008), Forschungsgebiet: parallele Simulationstechnik (German), *Reihe "Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen", DonNTU, FRTI-Werke,* Vol. 9 (150), pp. 9-36.
4. Svjatnyj, V.A., Smagin, O.M. and Solonin, O.M (2003), "Methods of parallelisation of the equation solver for MIMD-model of a network dynamic object", *Naukovi pratsi Donetskogo Natsionalnogo Tekhnichnogo Universytetu. Seria: Informatika, kibernetika ta obchyslyuvalna tekhnika*, Iss. 70, pp. 20-30.
5. Smagin, A.N. (2008), "Efficiency of MPI and OPENMP standards-based MIMD equation solvers for air dynamics of mine airing networks", *Materialy IV Naukovo-praktychnoi konferentsii Donbas-2020: Nauka i tekhnika vyrobnytstvu* [Materials of IV Scientific-Practical Conference Donbas-2020: Science and Technology for Industry], Donetsk, May 27-28, 2008, pp. 432-436.
6. Guseva, G.B. and Moldovanova, O.V. (2007), "MIMD parallel equation solver of the network dynamic object with distributed parameters", *Naukovi pratsi Donetskogo Natsionalnogo Tekhnichnogo Universytetu. Seria: "Problemy modelyuvannya ta avtomatyzatsii proektuvannya"*, pp. 149-158.
7. Svjatnyj, V.A. and Moldovanova O.V. (1999), "Equation generator for a parallel model of the network dynamic object with distributed parameters", *Naukovi pratsi Donetskogo Natsionalnogo Tekhnichnogo Universytetu. Seria: "Problemy modelyuvannya ta avtomatyzatsii proektuvannya"*, Iss. 10, pp. 135-141.
8. Mikov, A.I. and Zamyatina, E.B. "Loading balance in the distributed systems", *Natsionalny otkryty institut "Intuit"*, available at: http://www.intuit.ru/department/algorithms/distrsa/9/1.html

Received 11.01.17

A.L. Masyuk

HYBRID PARALLEL EQUATION SOLVER FOR NORMAL
AIR DISTRIBUTION` BASED ON DIRECT METHOD

Structure of the parallel equation solver for the problem of normal air distribution in mine ventilation network is analyzed and improved in the way of merging the existing MIMD structure with the integrated SIMD facilities of the modern CPUs. The algorithm of direct method-based equation solver has been improved by SSE and AVX extensions. Usage of SIMD component of the hybrid MIMD+SIMD structure allows decreasing the amount of the computational iterations several times, thus accelerating the whole simulation process.

K e y w o r d s: parallel algorithm, mine ventilation network, normal air distribution, direct method, integrated SIMD facilities.

МАСЮК Арсений Леонидович, аспирант кафедры компьютерной инженерии факультета компьютерных наук и технологий Донецкого национального технического университета, который окончил в 2002 г. Область научных исследований — параллельные вычислительные системы, интерактивные диалоговые алгоритмы.