A. V. Moryakov, S. S. Pylev, A. A. Sedov, A. S. Lubina, The method to get linear Caushy problem solution using parallel calculation,
Mat. Model., 2017, Volume 29, Number 2, 47–62

https://www.mathnet.ru/eng/mm3814

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ 2017 год, том 29, номер 2, стр. 47-62

МЕТОД РЕШЕНИЯ ЛИНЕЙНОЙ ЗАДАЧИ КОШИ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

© *2017 г.* ***А.В. Моряков, С.С. Пылев, А.А. Седов, А.С. Лубина***

НИЦ "Курчатовский институт", Москва
sailor@orc.ru, pylev_ss@nrcki.ru, Sedov_AA@nrcki.ru, Lubina_AS@nrcki.ru

Представлен метод нахождения решения линейной задачи Коши для систем обыкновенных дифференциальных уравнений большой размерности. Предложенный алгоритм для линейных систем дифференциальных уравнений первого порядка реализован в программе *EDELWEISS* с возможностью использования параллельных вычислений на суперкомпьютерах, применяющих *MPI* стандарт для обмена данными между параллельными процессами. Решение представляется в виде ряда по ортогональным полиномам на отрезке [0,1]. К особенностям алгоритма можно отнести его простоту и возможность решения нелинейных задач с поправкой оператора с учетом решения, полученного в итерационном процессе, а также возможность использования параллельных вычислений в процессе получения решения. Представлены тестовые задачи для нахождения решения нестационарного уравнения теплопроводности и сравнение результатов, полученных по программам *EDELWEISS, RELAP5/MOD3.3, ANSYS, LUCKY_HEATER*.

Ключевые слова: задача Коши, алгоритм, итерационный процесс, программа, компьютер, система уравнений, решение, пространство, вектор, тестовые задачи.

THE METHOD TO GET LINEAR CAUSHY PROBLEM SOLUTION USING PARALLEL CALCULATION

A.V. Moryakov, S.S. Pylev, A.A. Sedov, A.S. Lubina

NRC "Kurchatov Institute", Moscow

The method of solution the linear Cauchy problem for large systems of ordinary differential equations using parallel calculations is presented. The algorithm for linear systems of first order equations was realized by *EDELWEISS* computer code. This algorithm was developed especially for supercomputers that may use MPI technology to data exchange for parallel processes. The solution is presented as a row by orthogonal polynomials at [0,1] segment. Features of this algorithm are simplicity, opportunity to get solution by parallel calculations and also possibility to get a solution for non-linear tasks by changing the operator using the solution from iteration process. The test problems are presented for time-dependent heat transport equation solution. Results obtained by *EDELWEISS, RELAP5/MOD3.3, ANSYS* and *LUCKY_HEATER* codes are compared.

Key words: Cauchy problem, algorithm, iteration process, program, computer, system of equations, solution, space, vector, test tasks.

1. Введение. Многие задачи математической физики сводятся к решению задачи Коши для больших по размерности систем дифференциальных уравнений первого или второго порядка, точное и быстрое решение этой задачи представляет большой практический интерес.

Развитие вычислительной техники, ориентированной на использование параллельных вычислительных технологий, дает возможность эффективно решать эти задачи. Для использования всех возможностей суперкомпьютеров необходимы алгоритмы, способные применять все преимущества таких ЭВМ. Целью данной работы является тестирование алгоритма, представленного в [1], с возможностью параллельных вычислений при его реализации на ЭВМ на тестовых задачах получения решения нестационарного уравнения теплопроводности. Представлено сравнение результатов, полученных по программам *EDELWEISS* [1], *RELAP5/MOD3.3* [2], *ANSYS* [3] и *LUCKY_HEATER*.

Программа *EDELWEISS* может быть использована в качестве "решателя" для задач, сводимых к линейной задаче Коши, например, применялась для решения задач массопереноса радиоактивных продуктов деления по контурам ядерных энергетических установок.

2. Описание алгоритма.

Детальное описание алгоритма для решения задачи Коши для систем обыкновенных дифференциальных уравнений большой размерности с использованием параллельных вычислений представлено в [1]. Дадим основные положения этого алгоритма.

2.1. Постановка задачи. Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка с оператором $\mathrm{A}(t,\boldsymbol{\varphi}(t))$:

$$\frac{d\boldsymbol{\varphi}(t)}{dt} = \mathrm{A}(t,\boldsymbol{\varphi}(t)), \qquad (1)$$

начальные условия $\boldsymbol{\varphi}(0) = \boldsymbol{\vartheta}$, $t \in [0,1]$. Вектор $\boldsymbol{\varphi}(t)$ задан в пространстве R_0^n с расстоянием между элементами пространства $D(\mathbf{x},\mathbf{y}) = \max |x_j(t) - y_j(t)|$ для $\forall j = 1,...,n$.

Предполагается, что оператор $\mathrm{A}(t,\boldsymbol{\varphi}(t))$ удовлетворяет условию Липшица $|\mathrm{A}(t,\boldsymbol{\varphi}_1(t)) - \mathrm{A}(t,\boldsymbol{\varphi}_2(t))| \le M|\boldsymbol{\varphi}_1(t) - \boldsymbol{\varphi}_2(t)|$. Любая система, определенная на интервале времени $t \in [0,T]$, может быть сведена к системе, определенной на отрезке $[0,1]$ введением новой переменной $t = T \cdot s$, $s \in [0,1]$. Тогда $\frac{d\boldsymbol{\varphi}(s)}{ds} = \frac{d\boldsymbol{\varphi}(t)}{dt} \cdot \frac{dt}{ds} = \frac{d\boldsymbol{\varphi}(t)}{dt} \cdot T$, с учетом этого $\frac{d\boldsymbol{\varphi}(s)}{ds} = (\mathrm{A}(s,\boldsymbol{\varphi}(s)) \cdot T$.

2.2. Итерационный алгоритм для нахождения решения системы уравнений (1). Запишем уравнение (1) следующим образом:

$$\frac{d\boldsymbol{\varphi}(t)}{dt} = \mathrm{A}(t,\boldsymbol{\varphi}(t)) + r\boldsymbol{\varphi}(t) - r\boldsymbol{\varphi}(t),$$

введя некоторый положительный параметр r.

Построим последовательность $\boldsymbol{\varphi}_k(t)$ для получения решения уравнения (1) согласно итерационному процессу по следующей схеме, стартуя с некоторого, например, $\boldsymbol{\varphi}_0(t) = \boldsymbol{\vartheta}$, k – номер итерации

$$\frac{d\boldsymbol{\varphi}_k(t)}{dt} = \mathrm{A}(t,\boldsymbol{\varphi}_{k-1}(t)) + r\boldsymbol{\varphi}_{k-1}(t) - r\boldsymbol{\varphi}_k(t).$$

Обозначим

$$\gamma_{k-1}(t) = \mathrm{A}(t, \varphi_{k-1}(t)) + r\varphi_{k-1}(t),$$

тогда система уравнений примет вид

$$\frac{d\varphi_k(t)}{dt} = -r\varphi_k(t) + \gamma_{k-1}(t). \qquad (2)$$

Фактически процесс получения решения производится посредством итераций по источнику $\gamma_{k-1}(t)$, зависящему от $\varphi_{k-1}(t)$.

Особенностью системы (2) является то, что в уравнениях неизвестные в векторе $\varphi_k(t)$ не связаны друг с другом (связь производится через $\gamma_{k-1}(t)$), следовательно, могут быть найдены с помощью параллельных вычислительных процессов. Система имеет простое аналитическое решение, можно назвать его "ведущим" решением $\varphi_k(t) = e^{-rt}\int_0^t e^{rt'}\gamma_{k-1}(t')dt' + e^{-rt}\vartheta$. Обозначим $\mathrm{B}\varphi(t) = e^{-rt}\int_0^t e^{rt'}(\mathrm{A}(t', \varphi(t')) + r\varphi(t'))dt'$, тогда $\varphi_k(t) = \mathrm{B}\varphi_{k-1}(t) + e^{-rt}\vartheta$. В [1] показано, что для любых элементов θ_1 и θ_2 пространства R_0^n имеем консервативную оценку

$$D(\mathrm{B}^k\theta_1, \mathrm{B}^k\theta_2) = \max|\mathrm{B}^k\theta_1 - \mathrm{B}^k\theta_2| \le (M+r)^k \frac{1^k}{k!}\max|\theta_1 - \theta_2|.$$

Для достаточно большого k получим, что величина $(M+r)^k \frac{1^k}{k!} < 1$. Тогда, начиная с некоторого k, отображение B^k будет сжимающим отображением [4], следовательно, описанный выше итерационный процесс будет сходиться к решению уравнения (1). Чем меньше величина $(M+r)$, тем быстрее будет сходиться итерационный процесс.

Таким образом, решение задачи Коши для системы дифференциальных уравнений (1) сводится к нахождению решения системы интегральных уравнений второго рода типа Вольтерра с вырожденным ядром:

$$\varphi(t) = e^{-rt}\int_0^t e^{rt'}(\mathrm{A}(t', \varphi(t')) + r\varphi(t'))dt' + e^{-rt}\vartheta.$$

Продифференцировав это уравнение, получим (1). При $r = 0$ уравнение примет вид $\varphi(t) = \int_0^t \mathrm{A}(t', \varphi(t'))dt' + \vartheta$. Это уравнение может быть сразу получено из (1), если проинтегрировать правую и левую часть (1) на интервале $[0, t]$.

Рассмотрим случай, когда оператор $\mathrm{A}(t)$ линейный и не зависит от решения $\varphi(t)$. Тогда задача $\frac{d\varphi(t)}{dt} = \mathrm{A}(t)\varphi(t) + \sum_{i=0}^{m}\alpha_i P_i(t)$ с начальными условиями $\varphi(0) = \vartheta$, $t \in [0,1]$, может быть сведена к задаче с нулевыми начальными условиями путем замены $\varphi(t) =$

$= \vartheta + \xi(t)$ с начальными условиями для $\xi(0) = 0$, с источником в уравнении, равным $A(t)\vartheta + \sum_{i=0}^{m} \alpha_i P_i(t)$. Предполагается, что источник разложен в ряд по ортогональным полиномам на этом отрезке по степеням *t*. Базисные полиномы могут быть получены из полиномов Лежандра, заданных на отрезке $[-1,1]$.

2.3. Интегрирование по частям как способ получения моментов от решения. Так как в дальнейшем решение (2) будет разложено в ряд по ортогональным полиномам, необходимо найти моменты искомого решения от этих полиномов. В качестве ортогональных полиномов используются полиномы, полученные из полиномов Лежандра, в построении которых используются степени *t*. Покажем, как можно аналитически получать моменты от искомого решения от таких полиномов.

Известна формула интегрирования по частям, которая имеет вид [5]

$$\int_0^b u dv = (uv)_{|_0^b} - \int_0^b v du .$$

Так как основу полиномов образуют степени *t*, необходимо научиться вычислять моменты от этих степеней (что позволит в дальнейшем легко находить моменты от самих полиномов). Покажем, как данная формула может быть использована для нахождения момента функции $f(t)$ от t^n.

Рассмотрим интеграл $\int_0^b t^n f(t)dt$ и применим к нему формулу интегрирования по частям

$$\int_0^b t^n f(t)dt = (t^n \int f(t)dt)_{|_0^b} - n\int_0^b t^{n-1}(\int f(t')dt')dt = (t^n I^0(t))_{|_0^b} - n\int_0^b t^{n-1} I^0(t)dt ,$$

где $I^0(t) = \int_0^t f(t')dt'$.

Применим формулу интегрирования по частям ко второму слагаемому в предыдущей формуле

$$\int_0^b t^{n-1} I^0(t)dt = (t^{n-1} \int I^0(t)dt)_{|_0^b} - (n-1)\int_0^b t^{n-2} I^1(t)dt ,$$

где $I^1(t) = \int_0^t I^0(t')dt'$.

Последовательно будем применять формулу интегрирования по частям ко второму слагаемому в предыдущей формуле и получим формулу, позволяющую находить момент функции $f(t)$ от функции t^n, зная n интегралов от функции $f(t)$. Формула будет иметь вид

$$\int_0^b t^n f(t)dt = \sum_{i=0}^{n} (-1)^i \left(t^{n-i} I^i(t) \frac{n!}{(n-i)!} \right)_{|_0^b} = \sum_{i=0}^{n} (-1)^i b^{n-i} I^i(b) \frac{n!}{(n-i)!} . \qquad (3)$$

Отметим, что для вычисления момента по этой формуле не надо знать значения функции $I^i(t)$ на всем интервале интегрирования, нужно знать ее значения только на границах интервала интегрирования.

Далее рассмотрим задачу Коши, представленную уравнением

$$\frac{d\varphi_j(t)}{dt} = -r\varphi_j(t) + t^j,\ t \in [0,1],\ \text{с начальными условиями}\ \varphi_j(0) = 0. \qquad (4)$$

Найдем $I_j^i(t) = \int_0^t I_j^{i-1}(t')dt'$ от решения уравнения (4), индекс j определяет значение степени t. Для этого получим уравнения для $I_j^i(t)$, а решив их, сможем найти эти функции.

Продемонстрируем прием, который можно использовать для линейных уравнений или систем линейных уравнений, решение которых может быть найдено как аналитически, так и численным методом.

Проинтегрируем уравнение (4) от 0 до t:

$$\int_0^t \left(\frac{d\varphi_j(t')}{dt'}\right)dt' = \int_0^t -r\varphi_j(t')dt' + \int_0^t (t')^j dt' \ \text{или}\ \int_0^t \left(\frac{d\varphi_j(t')}{dt'}\right)dt' = -r\int_0^t \varphi_j(t')dt' + \frac{t^{j+1}}{j+1}. \qquad (5)$$

Ранее определяли $I_j^0(t) = \int_0^t \varphi_j(t')dt'$. Очевидно, для любой функции $\varphi(t)$ можно записать $\int_0^t \left(\frac{d\varphi(t')}{dt'}\right)dt' = \frac{d}{dt}(\int_0^t \varphi(t')dt') - \varphi(0)$. С учетом данного соотношения и определения $I^0(t)$ уравнение (5) примет вид

$$\frac{dI_j^0(t)}{dt} = -rI_j^0(t) + \frac{i^{j+1}}{j+1}. \qquad (6)$$

Таким образом, получили уравнение для нахождения $I_j^0(t) = \int_0^t \varphi_j(t')dt'$, с начальным условием для $I_j^0(0) = 0.$ Интегрируя (6), будем иметь уравнение для определения

$$I_j^1(t) = \int_0^t I_j^0(t')dt', \qquad \frac{dI_j^1(t)}{dt} = -rI_j^1(t) + \frac{i^{j+2}}{(j+1)(j+2)}.$$

Аналогично получаем уравнение для нахождения соответственно интегралов от решения

$$\frac{dI_j^i(t)}{dt} = -rI_j^i(t) + \frac{i^{j+i+1}}{\prod_{l=0}^{i}(j+l+1)},\quad j = 0,...,m,\ i = 0,...,m,$$

где m – порядок разложения источника $\gamma_{k-1}(t)$ по ортогональным полиномам в уравнении (2).

Задачи Коши для нахождения всех $I_j^i(t)$, $I_j^i(t) = \int_0^t I_j^{i-1}(t')dt'$ будут иметь нулевые начальные условия. Найдем рекуррентную формулу для нахождения следующего интеграла $F_n = \int_0^1 e^{rt} t^n dt$:

$$F_0 = \int_0^1 e^{rt} dt = \frac{1}{r}(e^r - 1),$$

используя интегрирование по частям

$$F_1 = \int_0^1 e^{rt} t^1 dt = \left(\frac{t^1}{r} e^{rt} \right)_{|0}^{1} - \frac{1}{r} \int_0^1 e^{rt} dt = \frac{1}{r} e^r - \frac{1}{r} F_0 .$$

Таким образом, получаем

$$F_n = \int_0^1 e^{rt} t^n dt = \left(\frac{t^n}{r} e^{rt} \right)_{|0}^{1} - \frac{n}{r} \int_0^1 e^{rt} t^{n-1} dt = \frac{1}{r} e^r - \frac{n}{r} F_{n-1} . \quad (7)$$

Так как решение уравнения (4) есть

$$\varphi_j(t) = e^{-rt} \int_0^t e^{rt'} (t')^j dt' , \quad (8)$$

то $\varphi_j(1) = e^{-r} F_j$.

При проведении описанного в разделе 1 итерационного процесса в реализации на ЭВМ в уравнении (2) предполагается, что член $\boldsymbol{\gamma}_{k-1}(t)$ разложен в ряд $\boldsymbol{\gamma}_{k-1}(t) = \sum_{i=0}^{m} \boldsymbol{\mu}_i P_i(t)$. Здесь $P_i(t)$ – ортогональные полиномы на отрезке $t \in [0,1]$, полученные из полиномов Лежандра. Тогда решение задачи (2) (как следствие ее линейности) может быть представлено как сумма решений задач (2) с правой частью $\boldsymbol{\gamma}_{i,k-1}(t) = \boldsymbol{\mu}_i P_i(t)$, а эти задачи, в свою очередь, могут быть редуцированы к простейшим задачам $\frac{d\psi_i(t)}{dt} = -r\psi_i(t) + t^i$ с нулевыми начальными условиями. Используя формулы (3), (7), (8), можно аналитически получить решения и моменты от решения для этих задач и в итоге найти решение уравнения (2) в виде $\boldsymbol{\varphi}_k(t) = \sum_{i=0}^{m} \boldsymbol{\omega}_i P_i(t)$ и затем использовать это решение для вычисления следующего $\boldsymbol{\gamma}_k(t) = \sum_{i=0}^{m} \boldsymbol{\mu}_i P_i(t)$ в итерационном процессе.

Продемонстрируем алгоритм решения задачи (2). Так как элементы вектора $\boldsymbol{\varphi}(t)$ в уравнении не связаны друг с другом, рассмотрим уравнение

$$\frac{d\theta(t)}{dt} = -r\theta(t) + \sum_{i=0}^{m} D_i P_i(t) , \quad (9)$$

$\theta(t)$ – элемент вектора $\boldsymbol{\varphi}(t)$ с начальными условиями $\boldsymbol{\theta}(0)=0$, а $\gamma(t)=\sum_{i=0}^{m} D_i P_i(t)$ – элемент вектора $\boldsymbol{\gamma}(t)$. Уравнение линейно, и решение этого уравнения может быть представ-лено как $\theta(t)=\sum_{i=0}^{m} D_i f_i(t)$, где $f_i(t)$ – решение уравнения $df_i(t)/dt=-rf_i(t)+P_i(t)$ с начальными условиями $f_i(0)=0$. Найдем моменты $\vartheta_j=\int_0^1 P_j(t)\theta(t)dt=(P_j(t)\theta(t))$ от ортогональных полиномов для решения уравнения (9). С учетом разложения $P_j(t)=\sum_{l=0}^{m} p_{l,j}t^l$ можно записать

$$\vartheta_j=(P_j(t)\theta(t))=\sum_{i=0}^{m} D_i(P_j(t)f_i(t))=\sum_{i=0}^{m} D_i\sum_{k=0}^{m}(P_j(t)p_{i,k}g_k(t))=\sum_{i=0}^{m} D_i\sum_{k=0}^{m} p_{i,k}(P_j(t)g_k(t)),$$

где $g_k(t)$ – решение уравнения

$$dg_k(t)/dt=-rg_k(t)+t^k \tag{10}$$

с начальными условиями $g_k(0)=0.$ Тогда

$$\vartheta_j=\sum_{i=0}^{m} D_i\sum_{k=0}^{m} p_{i,k}(P_j(t)g_k(t))=\sum_{i=0}^{m} D_i\sum_{k=0}^{m} p_{i,k}\sum_{l=0}^{m} p_{l,j}(t^l g_k(t)),$$

$(t^l g_k(t))$ – моменты решений задачи (10), которые могут быть найдены через интегрирование по частям. Если рассматривать ϑ_j как элемент вектора $\vartheta=(\vartheta_0,...,\vartheta_m)$, тогда можно записать $\vartheta=\mathbf{ZD}$, где вектор $\mathbf{D}=(D_0,...,D_m)$ из уравнения (9), а $\mathbf{Z}=\|z_{i,j}\|$ есть матрица размерностью $m\times m$,

$$z_{i,j}=\sum_{k=0}^{m} p_{i,k}\sum_{l=0}^{m} p_{l,j}(t^l g_k(t)) \tag{11}$$

– элементы этой матрицы. Таким образом, коэффициенты разложения по ортогональным полиномам решения задачи (9) с нулевым начальным условием могут быть найдены в виде произведения матрицы на вектор $\vartheta=\mathbf{ZD}$. В итоге имеем для $\theta(t)=\sum_{i=0}^{m} D_i f_i(t)=$

$=\sum_{i=0}^{m}\vartheta_i P_i(t)$. Отметим, что матрица $\mathbf{Z}$ зависит только от r и коэффициентов полиномов порядка m и может быть вычислена один раз в рамках принятой модели для аппроксимации решения. Изначально предполагалось, что r один и тот же для всех элементов вектора $\boldsymbol{\varphi}(t)$, но формально для каждого $\varphi_j(t)=\theta_j(t)$ элемента вектора $\boldsymbol{\varphi}(t)$ можно использовать свой r_j (конечно, с коррекцией получения уравнения (2)).

2.4. Выбор параметра *r***.** Если оператор A ограничен величиной M_1 на отрезке $t\in[0,T]$, то интервал времени T всегда можно разбить на N частей с шагом Δt ($M = M_1\Delta t$ на интервале [0,1]) таким образом, чтобы величина $(M+r)<1$ была достаточно малой величиной (выбрав $r<1$), тогда уже оператор B^1 будет сжимающим отображением, а задача нахождения решения на отрезке $t\in[0,T]$ – сводиться к последовательности задач Коши с известными начальными условиями для $t_i = (i-1)\Delta t$, $i = 1,...,N$.

Несмотря на то что формально сходимость итерационного процесса возможна при любом r, фактически параметр r должен быть сравним с M, в этом случае при формировании $\boldsymbol{\gamma}_{k-1}(t)$ в итерационном процессе будет вноситься сравнимое с решением "возмущение" в правую часть системы уравнений (2). Шаг по времени при решении практических задач выбирался так, чтобы $0<(M+r)<20$ при условии $r = M$. Кроме того, еще одним критерием выбора шага является требование "слабой" зависимости решения от величины шага (сравнение результатов с шагом Δt и $2\Delta t$). Параметр r и порядок разложения решения по ортогональным полиномам m однозначно определяют матрицу $\mathbf{Z} = \| z_{i,j} \|$, при вычислении которой используются рекуррентные формулы, полученные через интегрирование по частям.

Почему выбрано уравнение (2) для получения "ведущего" решения? Решение ищется посредством итераций по $\boldsymbol{\gamma}_{k-1}(t)$, зависящему от $\boldsymbol{\varphi}_{k-1}(t)$. Система уравнений (2) имеет точное аналитическое решение, что дает возможность точно вычислять моменты от решения с использованием матрицы $\mathbf{Z}$, т.е. сразу получаются коэффициенты разложения по ортогональным полиномам для решения. В итоге окончательное решение для системы (2) (для каждой компоненты вектора) будет получено именно через аналитическое решение (в рамках принятого порядка разложения по m) дифференциального уравнения вида (2) со "сформировавшейся" в процессе итераций своей правой частью $\boldsymbol{\gamma}_{k-1}(t)$.

3. Программа *EDELWEISS*

Предложенный выше алгоритм был использован для решения задачи $d\boldsymbol{\varphi}(t)/dt = \mathbf{A}\boldsymbol{\varphi}(t) + \sum_{i=0}^{m} \boldsymbol{\alpha}_i t^i$, с начальными условиями $\boldsymbol{\varphi}(0) = \boldsymbol{\vartheta}$, $t\in[0,T]$, $\mathbf{A}$ – линейный оператор (матрица большой размерности n). Алгоритм реализован в программе *EDELWEISS*. Программа *EDELWEISS* использует *MPI* [6] технологию для организации параллельного вычислительного процесса и ориентирована на современные вычислительные комплексы.

Рассматриваемая система уравнений, заданная в пространстве размерностью n, путем замены переменной приводится к системе с нулевыми начальными условиями с отображением на отрезок $t\in[0,1]$:

$$\frac{d\boldsymbol{\varphi}(t)}{dt} = \mathbf{A}\boldsymbol{\varphi}(t) + \sum_{l=1}^{m} \boldsymbol{\beta}_l P_l(t). \qquad (12)$$

Возможно использование программы в качестве "решателя" при решении различных задач, сводимых к задаче (12). *EDELWEISS* работает с матрицей, упакованной по определенному закону, исключающему наличие нулевых элементов. Программа опреде-

ляет необходимые связи при организации обмена между параллельными процессами, оптимизируя *MPI* обмен данными.

3.1. Параллельная схема решения задачи (12). Система уравнений (12) разбивается на подсистемы размерностью N_i. Число подсистем равно N. Тогда $n = \sum_{i=1}^{N} N_i$. Желательно делать равномерную разбивку на подсистемы $N_i \approx n / N$, таким образом, число операций на каждом вычислительном ядре будет примерно одинаковым, это необходимо для получения лучшей эффективности организованного параллельного процесса.

Связь между подсистемами осуществляется посредством пересчитываемого на итерациях членов "связи" для каждой подсистемы. Член "связи" отражает влияние других подсистем на данную подсистему. Уравнение для подсистемы *i* имеет вид

$$\frac{d\boldsymbol{\varphi}_i^k(t)}{dt} = \mathbf{A}_i\boldsymbol{\varphi}_i^{k-1}(t) + r\boldsymbol{\varphi}_i^{k-1}(t) - r\boldsymbol{\varphi}_i^k(t) + \sum_{l=1}^{m}\boldsymbol{\beta}_{i,l}P_l(t) + \sum_{j=1}^{N-1}\boldsymbol{\chi}_{i,j\neq i}^{k-1}(t), \qquad (13)$$

$t \in [0,1]$, $\sum_{l=1}^{m}\boldsymbol{\beta}_{i,l}P_l(t)$ – член источника, $\boldsymbol{\varphi}_i(0) = 0$ – начальные условия, *k* – индекс номера итерации, $\boldsymbol{\chi}_{i,j}^{k-1}(t) = \mathbf{A}_{j,i}\boldsymbol{\varphi}_j^{k-1}(t)$ – член, связывающий подсистему *i* с подсистемой *j*.

Задача (12) решается с использованием *N* параллельных вычислительных процессов. Каждый вычислительный процесс обеспечивает нахождение решения задачи (13). Задача (13) решается по алгоритму, описанному в разделе 2. Получение решения задачи (12) осуществляется через следующий итерационный процесс.

1. С использованием начального приближения параллельно получаются решения для каждой подсистемы $\boldsymbol{\varphi}_i^0(t)$ (верхний индекс – номер итерации) в виде разложения в ряд по ортогональным полиномам.

2. Рассчитывается источник, связывающий подсистемы друг с другом $\sum_{j=1}^{N-1}\boldsymbol{\chi}_{i,j\neq i}^{0}(t)$ в виде ряда по ортогональным полиномам с использованием передачи данных по *MPI* стандарту. Итерация считается завершенной.

3. Параллельно получаются новые решения для задачи (13) $\boldsymbol{\varphi}_i^1(t)$ и итерационный процесс продолжается по описанной схеме до получения решения задачи (12) с заданной относительной точностью для двух последовательно полученных решений. В [1] показана сходимость этого параллельного итерационного процесса к решению задачи (12).

В данной модели предполагается связь каждого подпространства с каждым. Полное количество связей в системе в общем случае равно N^2. Алгоритм нахождения решения задачи (12) достаточно хорошо распараллеливается, так как все операции при вычислении решения сведены к умножению матрицы на вектор.

Следует отметить, что при наличии полного количества связей в системе при большом значении N рассматриваемая параллельная схема будет мало интересна, так как итерационный процесс получения решения будет длительным.

Для задач, где оператор $\mathbf{A}$ описывает конечно-разностную аппроксимацию уравнений в частных производных первого или второго порядка по пространственным коор-

динатам в прямоугольной системе координат *X,Y,Z* и число пространственных подобластей (подпространств) равно N, связь подобластей будет только с соседними подобластями. Таким образом, полное число связей в системе будет не более $6N$. Для таких задач эффективность параллельного процесса будет значительной величиной, даже при больших значениях *N*. Вычислительную модель можно масштабировать (по числу используемых параллельных процессов, уменьшая вычислительную нагрузку на процесс) до тех пор, пока время обмена данными будет сравнимо со временем вычисления решения на итерации.

3.2. Оценка необходимого числа операций и выбор параметра *m*. Оценим необходимое число операций при проведении итерационного процесса. При найденной заранее матрице $\mathbf{Z} = \| z_{i,j} \|$ размерностью $m \times m$ (именно эта матрица позволяет сразу находить решение системы (2), используя коэффициенты разложения $\boldsymbol{\gamma}_{k-1}(t)$ в виде разложения по ортогональным полиномам), и матрице **A** размерностью *n*×*n*, например, в случае использования одного вычислительного ядра, необходимое число операций при проведении одной итерации равно $2m^2n + 2n^2m - mn$. Величина m определяет порядок аппроксимации решения по переменной *t*, значение этого параметра определяется требованием к точности решения для конкретных задач. Достаточность принятой модели для аппроксимации решения по времени обычно определяется сравнением результатов, полученных с различными временными шагами. Отметим, что число операций по сравнению со стандартным методом (простая итерация) приблизительно в m раз больше, если $n >> m$. Тем не менее, использование $m > 1$ позволяет иметь более высокий порядок аппроксимации решения по t наряду с гарантированной сходимостью итерационного параллельного процесса для $\mathbf{A}$ общего вида.

При решении практических задач, чтобы избежать потери точности при вычислениях не следует выбирать большую величину *m*. При использовании 8 байт для представления числа на ЭВМ можно взять *m* не более 5 (с учетом того, что при вычислении **Z** будут участвовать степени по *t* до 2*m*).

4. Уравнения второго порядка

Система дифференциальных уравнений второго порядка

$$\frac{d^2\boldsymbol{\varphi}(t)}{dt^2} = \mathbf{A}(t)\boldsymbol{\varphi}(t) + \sum_{i=0}^{m} \boldsymbol{\alpha}_i P_i(t), \tag{14}$$

$t \in [0,1]$, с начальными условиями $\boldsymbol{\varphi}(0) = \boldsymbol{\vartheta}$ и $\boldsymbol{\varphi}'(0) = \boldsymbol{\theta}$ путем замены переменных $\boldsymbol{\varphi}(t) = \boldsymbol{\vartheta} + \boldsymbol{\theta}t + \boldsymbol{\xi}(t)$ может быть сведена к системе с нулевыми начальными условиями $\boldsymbol{\xi}(0) = 0$, $\boldsymbol{\xi}'(0) = 0$. Если сделать замену переменных $\boldsymbol{\xi}(t) = \int_0^t \boldsymbol{\psi}(t')dt'$, то (14) может быть представлена как система дифференциальных уравнений первого порядка:

$$\frac{d\boldsymbol{\psi}(t)}{dt} = \mathrm{A}(t)\int_0^t \boldsymbol{\psi}(t')dt' + \sum_{i=0}^{m} \boldsymbol{\beta}_i P_i(t), \tag{15}$$

$t \in [0,1]$, с начальными условиями $\boldsymbol{\psi}(0) = 0$.

Полагая, что оператор $\mathbf{A}(t)$ удовлетворяет условию Липшица

$$| \mathrm{A}(t)\psi_1(t) - \mathrm{A}(t)\psi_2(t) | \le M | \psi_1(t) - \psi_2(t) |$$

и с учетом того, что $0 \le t \le 1$, можно сделать вывод, что

$$| \mathbf{A}(t)\int_0^t \boldsymbol{\psi}_1(t')dt' - \mathbf{A}(t)\int_0^t \boldsymbol{\psi}_2(t')dt' | = | \int_0^t \mathbf{A}(t)(\boldsymbol{\psi}_1(t) - \boldsymbol{\psi}_2(t))dt' | \le \int_0^t |\mathbf{A}(t)(\boldsymbol{\psi}_1(t) - \boldsymbol{\psi}_2(t))| \, dt' \le$$

$$\le tM | \boldsymbol{\psi}_1(t) - \boldsymbol{\psi}_2(t) | \le M | \boldsymbol{\psi}_1(t) - \boldsymbol{\psi}_2(t) | .$$

Следовательно, для получения решения системы (15) можно использовать алгоритм, рассмотренный в разделах 2-3.

5. Тестовые задачи

Несколько тестов для программы *EDELWEISS* представлены в [1]. В настоящей работе рассматривается одномерная нестационарная задача теплопроводности для пластины толщиной *l*

$$\rho C_p \frac{\partial T(x,t)}{\partial t} = \lambda \frac{\partial^2 T(x,t)}{\partial x^2} + Q_v \qquad (16)$$

с начальными условиями $T(x,t_0) = T_0$ и граничными условиями $T(0,t) = T_0$ и $T(l,t) = T_0$, где Q_v – удельный источник тепловыделения. На сетке размерностью n $(i = 0....n+1)$ имеем конечно-разностную аппроксимацию

$$\rho C_p \frac{\partial T_i(t)}{\partial t} = \lambda \frac{T_{i-1}(t) - 2T_i(t) + T_{i+1}(t)}{h^2} + Q_{v,i}\,,\ T_{i=0}(t) = T_0\,,\ T_{i=n+1}(t) = T_0\,,$$

h – шаг пространственной сетки. Задача имеет аналитическое асимптотическое решение при $t \to \infty$, $T(x) = \frac{1}{\lambda}(-\frac{Q_v x^2}{2} + \frac{x Q_v l}{2} + T_0\lambda)$.

Результаты расчета (профили температуры по толщине пластины) задачи (16) при *n*=19 по программам *EDELWEISS* и *RELAP5/MOD3.3* для моментов времени *t*=200 *c* (конец стадии разогрева); 300 *c* (100 с после стадии разогрева); 500 *c* (300 *c* после стадии разогрева) при следующих начальных и граничных значениях: толщина пластины *l* = =10 *см*, температура на границах пластины – T_0 =320*K*, свойства материала приняты не зависящими от температуры – $\lambda = 16.9\ Вт / м \cdot K$, $\rho C_p = 3.79 \cdot 10^6\ Дж / м^3 \cdot K$, объемное энерговыделение: Q_v = 0 на интервале времени от 0 до 100 с, $Q_v = 1.0 \cdot 10^5\ Вт / м^3$, на интервале времени от 100 *c* до 200 *c* и Q_v = 0 на интервале времени от 200 *c* до 500 *c* представлены на рис.1. Относительная погрешность результатов от аналитического решения не превышает 0.002. Относительный критерий точности для выхода из итераций для *EDELWEISS eps_ou t*=10^{-8}.

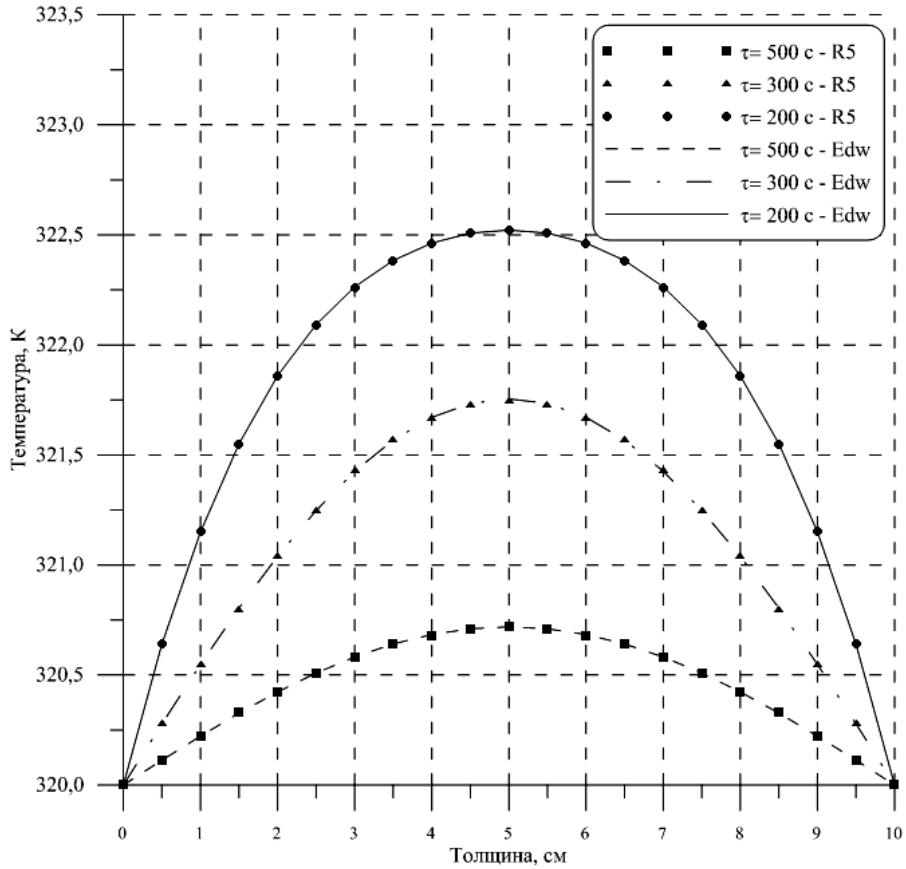
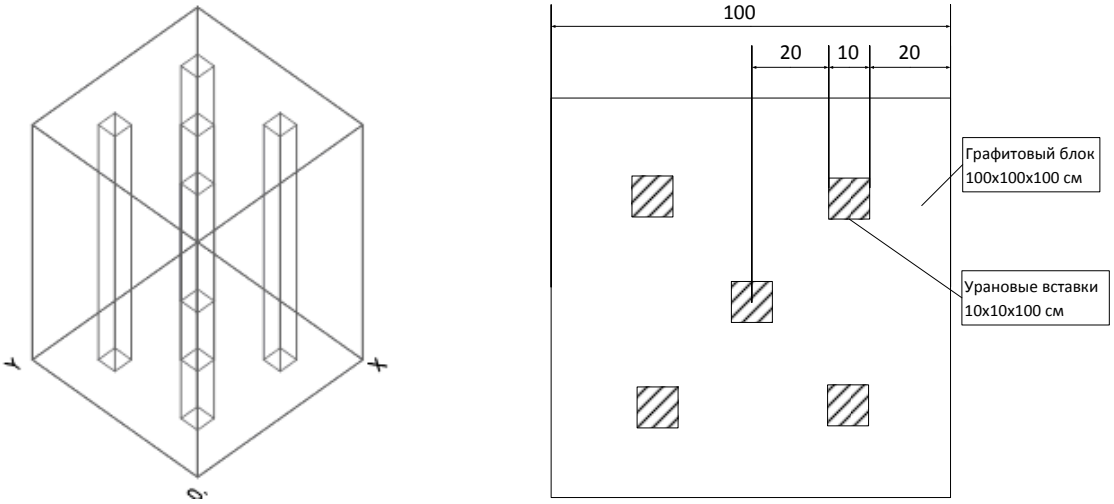


Рис.1. Профиль температуры по толщине пластины при различных значениях времени.

На базе параллельной вычислительной платформы (вычислителя *EDELWEISS,* в котором в качестве оператора **A** применена конечно-разностная аппроксимация уравнения теплопроводности и геометрического модуля программы *LUCKY* [7-9]) создана программа *LUCKY_HEATER* для расчета нестационарных полей температур в прямоугольной геометрии.

В качестве 3*D* теста рассмотрен куб из графита ($\lambda = 170\ Bm/м\cdot K$, $\rho C_p = 1.224\cdot 10^6\ Дж/м^3 K$) с ребром 1 *м*. В кубе имеются пять каналов с прямоугольным сечением 10×10 *см* высотой 1 *м* из металлического урана ($\lambda = 25\ Bm/м\cdot K$, $\rho C_p = 2.229\cdot 10^6\ Дж/м^3 K$). Геометрия задачи представлена на рис.2. Граничные условия по всем граням куба T =300*K*, начальное приближение $T(t_0,x,y,z)$=300*K* во всем расчетном объеме. Энерговыделение в каналах $Q = 10^6\ Bm/м^3$. Результатом расчета является поле температур на момент времени $t = 100\ c$. Расчет проводился с шагом по времени $\Delta t = 0.156986\ c$ (637 временных интервалов на 100 *c*). Использовалась пространственная сетка по всем осям с шагом $\Delta = 0.5\ см$, таким образом, полная размерность задачи N = 200·200·200 = 8 000 000 пространственных интервалов. Относительный критерий точности для выхода из итераций 10^{-8}, число членов ряда для аппроксимации решения по времени равно 2. Задача решалась с использованием программ *ANSYS* и *LUCKY_HEATER*, причем результат по *LUCKY_HEATER* был получен с использованием различного числа вычислительных ядер с применением метода пространственной декомпозиции области

расчета (решение для каждой пространственной подобласти вычислялось на отдельном вычислительном ядре). Относительная погрешность полей температур по этим программам не превышает 0.01%. Время расчета на одном ядре по *ANSYS* в 1.5 раза меньше, чем по *LUCKY_HEATER*. На рис.3 дано распределение температуры по диагонали в среднем сечении куба (высота 0.5 *м*), полученное по программам *ANSYS* и *LUCKY_HEATER*.



а) – изометрия расчетной модели б) – размеры в поперечном сечении

Рис.2. Геометрия для расчета 3*D* теста.

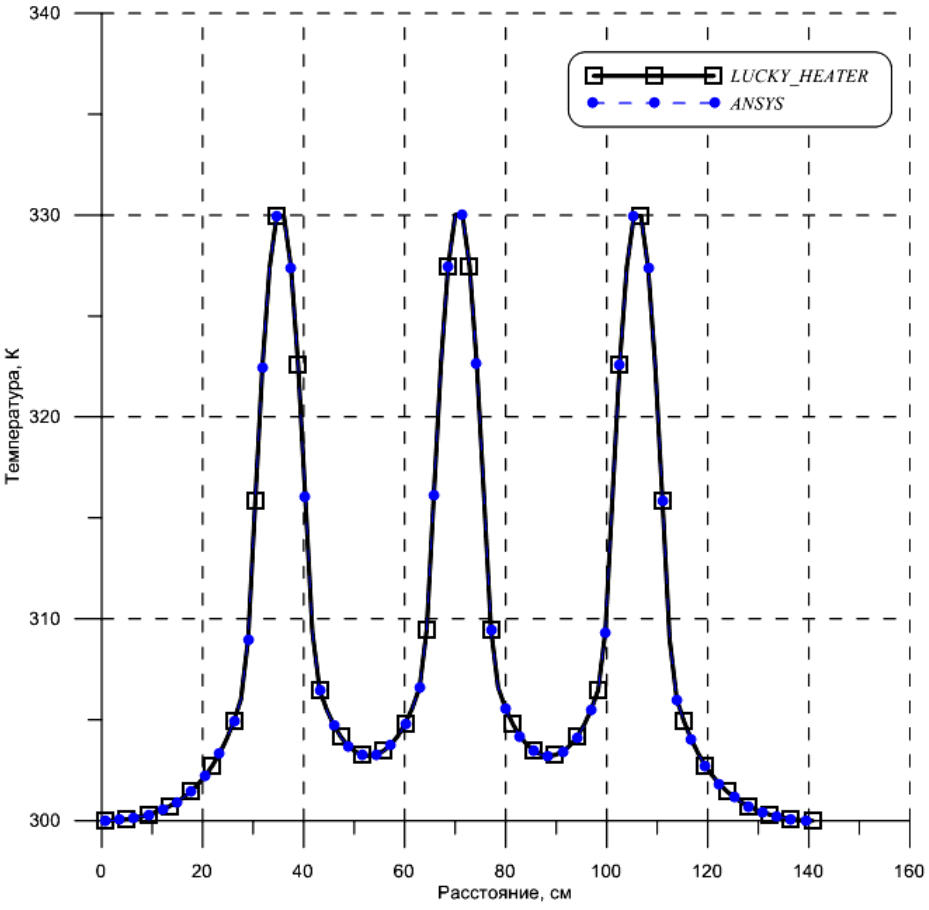


Рис.3. Распределение температур, полученное по программам *ANSYS* и *LUCKY_HEATER* по диагонали в среднем сечении куба на момент времени $t = 100\,c$.

Введем величину эффективности параллельного процесса, определив ее как $E_p = t_1/(t_n n)$, где t_1 – время решения задачи на одном вычислительном ядре, t_n – время решения задачи с использованием n вычислительных ядер. E_p определяет выигрыш по времени на одно используемое вычислительное ядро. В табл.1 представлены величины: эффективность E_p, время решения задачи t_n и доля времени на вычислительный процесс δ_n (время расчета без времени на обмен данными между вычислительными ядрами при получении решения, отнесенное к полному времени расчета) в зависимости от числа используемых вычислительных ядер. Пространственная сетка для всех расчетов была одинаковой, но уменьшалась для каждого ядра кратно используемому числу вычислительных ядер.

Таблица 1. Зависимость величин, характеризующих параллельный вычислительный процесс, от числа используемых ядер.

n – число ядер в параллельном вычислительном процессе	E_p – эффективность параллельного вычислительного процесса	δ_n – доля времени на вычислительный процесс	t_n – время расчета задачи, с
1	1	1	13817
4 (2·2·1)*)	0.887	0.974	3890
8 (2·2·2)	0.795	0.992	2172
16 (2·2·4)	0.572	0.924	1509
32 (4·4·2)	0.570	0.947	757
64 (4·4·4)	0.557	0.933	387
125 (5·5·5)	0.545	0.891	203
250 (5·5·10)	0.497	0.784	111
512 (8·8·8)	0.408	0.595	66
1000 (10·10·10)	0.242	0.309	57
2000 (10·10·20)	0.094	0.147	73
*) – в скобках приведено разбиение пространственной области на подобласти по координатным осям *X, Y, Z*			

Анализ полученных результатов (табл.1) показывает, что алгоритм достаточно хорошо распараллеливается $E_p \approx 0.5$ до тех пор, пока время обмена данными меньше времени вычисления (времени на проведение итерационного процесса). Вычислительный процесс можно масштабировать по числу используемых параллельно работающих вычислительных ядер до тех пор, пока время расчета итерации на ядре не будет сравнимо со временем обмена данными между параллельными вычислительными процессами.

Немонотонность зависимости δ_n от n обусловлена особенностями архитектуры суперкомпьютера (изменение времени на обмен данными между процессами в зависимости от распределения выделенных для расчета вычислительных узлов, загруженности компьютера при старте задачи). Несомненно, интересным будет использование вычислительного гибрида *MPI+GPU* (*CUDA*) для предложенного алгоритма.

6. Выводы

1. Предложен алгоритм получения решения линейной задачи Коши в виде ряда по ортогональным полиномам с возможностью использования параллельных вычислений

на современных вычислительных комплексах. Показана сходимость параллельного итерационного процесса к решению задачи.

2. Для использования на суперкомпьютерах создана программа *EDELWEISS*, где реализован предложенный алгоритм для решения систем линейных дифференциальных уравнений первого порядка большой размерности.

Алгоритм хорошо распараллеливается с возможностью использования *GPU* (графических процессоров) и *MPI* технологии, так как большинство вычислительных операций разделены и представляют собой умножение матрицы на вектор.

Следует отметить, что найденное по данному алгоритму решение задачи Коши является приближенным. Точность полученного решения определяется рамками вычислительной модели – порядком m разложения в ряд по ортогональным полиномам "ведущего" решения в процессе проведения итераций.

К особенностям предложенного алгоритма можно отнести:

– возможность решать нелинейные задачи с пересчетом оператора $\mathbf{A}(\varphi(t),t)$ при проведении итерационного процесса;

– простоту и возможность аналитически (точно) вычислять моменты от решения;

– оператор **Z**, найденный по формуле (11) один и тот же для всех элементов вектора и может быть вычислен заранее один раз в рамках выбранной вычислительной модели и определяется значением r и порядком m разложения решения в ряд по ортогональным полиномам;

– использование параллельных вычислений в процессе получения решения;

– главное требование к оператору – удовлетворение условию Липшица.

Результаты расчета тестовой задачи по программе *EDELWEISS* адекватно согласуются с результатами, полученными по программе *RELAP5/MOD3.3*, что дает основание полагать корректность реализации в программе *EDELWEISS* предложенного алгоритма. На примере 3*D* теста показано, что предложенный алгоритм достаточно хорошо распараллеливается на достаточно большом числе вычислительных ядер для слабосвязанных систем.

Все расчеты проводились на суперкомпьютере МВС-10П в Межведомственном Суперкомпьютерном центре Российской академии наук [10]. Авторы выражают благодарность МСЦ за возможность проведения расчетов и выделенные ресурсы.

СПИСОК ЛИТЕРАТУРЫ

1. *А.В. Моряков.* Алгоритм решения линейной задачи Коши для систем обыкновенных дифференциальных уравнений большой размерности с использованием параллельных вычислений // ВАНТ. Сер. Физика ядерных реакторов, 2015, вып.2;
   A.V. Moryakov. Algoritm resheniia lineinoi zadachi Koshi dlia sistem obyknovennykh differentsialnykh uravnenii bolshoi razmernosti s ispolzovaniem parallelnykh vychislenii // VANT. Ser. Fizika iadernykh reaktorov, 2015, vyp.2.
2. *"RELAP5/MOD3.3* Code Manual" Vol.1-8, NUREG/CR-5535/Rev.P3, ISL, Inc., Rockwell, MaryLand, Idaho. USA, March 2003.
3. ANSYS, Release 12.1 // ANSYS Inc., Southpointe 275 Technology Drive, Canonsburg, PA 15317, November 2009, ISO 9001:2008, http://www.ansys.com.
4. *А.Н. Колмогоров, С.В. Фомин.* Элементы теории функций и функционального анализа. – М.: Наука, 1972;

A.N. Kolmogorov, S.V. Fomin. Elementy teorii funktsii i funktsionalnogo analiza. – M.: Na-uka, 1972.

5. *Г. Корн, Т. Корн.* Справочник по математике. – М.: Наука, 1970;
   G. Korn, T. Korn. Spravochnik po matematike. – M.: Nauka, 1970.
6. *В.В. Воеводин, Вл.В. Воеводин.* Параллельные вычисления. – СПб: БХВ-Петербург, 2002;
   V.V. Voevodin, Vl.V. Voevodin. Parallelnye vychisleniia. SPb: BXB-Peterburg, 2002.
7. *А.В. Моряков.* Программа *LUCKY*. Решение уравнения переноса нейтронов и гамма излучения с использованием параллельных технологий // ВАНТ. Сер. Физика ядерных реакторов, 2010, вып.4, с.18-29;
   англ. пер.: *A.V. Moryakov.* The *LUCKY* program for solving the transport equation for neutrons and gamma radiation with the use of parallel technologies // Physics of Atomic Nuclei, 2011, v.74, №14, p.1891.
8. *А.В. Моряков.* Алгоритм получения угловых потоков в ячейке для многопроцессорных программ *LUCKY* и *LUCKY_C* // ВАНТ. Сер. Физика ядерных реакторов, 2011, вып.1, с.3-7;
   англ. пер.: *A.V. Moryakov.* Algorithm for obtaining angular fluxes in a cell for the *LUCKY* and *LUCKY_C* multiprocessor programs // Physics of Atomic Nuclei, 2012, v.75, №14, p.1627.
9. *А.В. Моряков.* Алгоритм пространственно-энергетического распараллеливания для нахождения решения критической задачи, реализованный в многопроцессорной программе LUCKY_C // ВАНТ. Сер. Физика ядерных реакторов, 2012, вып. 1,с. 24-27;
   англ. пер.: *A.V. Moryakov.* Algorithm intended for space-energy parallelization in solving the criticality problem and implemented in the LUCKY_C multiprocessor program // Physics of Atomic Nuclei, 2013, v.76, №13, p.1569.
10. https://www.jscc.com

Поступила в редакцию 03.09.15