УДК 539.3
КП
№ госрегистрации 0109U001389
Инв. №

Министерство образования и науки Украины
Сумский государственный университет
(СумГУ)
40007, г. Сумы, ул. Р.Корсакова, 2;
тел. 33 41 08, факс (542) 33 40 49

УТВЕРЖДАЮ
Проректор по научной работе,
д.ф.-м.н., профессор

_____ А. М. Чёрноус

2009.12.25

ОТЧЕТ

О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПОЛЕЙ В НАНОРАЗМЕРНЫХ И ФРАКТАЛЬНЫХ СТРУКТУРАХ

(промежуточный)

Начальник НИЧ
канд. техн. наук, доцент _____В.А. Осипов
2009.12.25

Руководитель НИР зав. кафедрой
прикладной и вычислительной математики
д-р физ.-мат. наук, профессор _____Л.А. Фильштинский
2009.12.25

2009

Рукопись закончена 22 декабря 2009 года

Результаты этой работы рассмотрены на заседании научного совета СумГУ,
протокол от 2009.12.24 № 6

СПИСОК АВТОРОВ

Руководитель НИР главный научный сотрудник, д-р физ.-мат. наук, профессор	Л.А. Фильштинский (реферат, введение, выводы, раздел 1, раздел 2)
Научный сотрудник	Т. Л. Мизина (раздел 1 (п. 1.3)
Инженер 1-й категории	Л. Л. Фильштинская (раздел 1 (п. 1.1)
Ассистент	Т. А. Киричек (раздел 1, введение)
Аспирант	Т. В. Мукомел (раздел 1 (п.1.1, 1.2)
Зав. лабораторией	А. В. Барсук (раздел 2 (п.2.1-2.3))
Аспирант	А. В. Бережный (раздел 2 (п.2.4-2.7))

РЕФЕРАТ

Отчет о НИР: 68 с., 22 рис., 4 табл., 78 ист.

Рассмотрен новый класс задач о тепло и массопереносе во фрактальных средах, чрезвычайно актуальный для полимерных структур, перколяционных кластеров, аморфных полупроводников, пористых материалов и т.п. Идеология рассмотрения указанных проблем вытекает из глубоких статистических, термодинамических соображений и в математическом отношении сводится к решению дифференциальных уравнений с производными дробного порядка по времени и пространственным переменным.

В данном отчете исследованы процессы аномального переноса типа субдиффузии и супердиффузии в одномерных и трехмерных телах для тех случаев, когда на поверхности тела действует импульсный тепловой поток.

Рассмотрены квазистатическая задача термоупругости с дробными производными Рисса и Капуто, а также обобщение задачи Даниловской для нелокального уравнения теплопроводности.

Получены новые фундаментальные закономерности в распределении тепловых и силовых полей во фрактальных средах в зависимости от порядка производной по времени α, характеризующей нелокальность процесса и порядка производной по пространственной переменной β, связанной с заменой нормального распределения Гаусса классического броуновского движения более общим вероятностным распределением Леви.

Намечена идеология параллельных вычислений применительно к проблемам указанного класса и родственных им.

ФРАКТАЛЬНАЯ СРЕДА, УРАВНЕНИЯ С ДРОБНОЙ ПРОИЗВОДНОЙ, АНОМАЛЬНАЯ ДИФФУЗИЯ (ТЕПЛОПРОВОДНОСТЬ), ТЕРМОУПРУГОСТЬ, ПРЕОБРАЗОВАНИЕ ЛАПЛАСА, ФУНКЦИЯ МИТТАГ - ЛЕФЛЕРА, КАЧЕСТВЕННЫЕ И КОЛИЧЕСТВЕННЫЕ ЗАКОНОМЕРНОСТИ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В механике сплошной среды рассматриваются тела, обладающие пространственно однородными свойствами. Если материалы неоднородны и нерегулярны, то процессы переноса, протекающие в них, не подчиняются законам классической механики. Процессы переноса частиц и энергии (диффузия и теплопроводность соответственно), возникающие в пористых материалах, аморфных полупроводниках, перколяционных кластерах, полимерных структурах, называют аномальными, а иногда и фрактальными из-за их связи с дробно-дифференциальным исчислением [1-5].Наиболее естественным и удобным математическим аппаратом описания процессов аномальной диффузии (теплопроводности) на некотором множестве являются уравнения в частных дробных производных как по пространственным координатам, так и по времени [6, 7].

Производная дробного порядка — это нелокальная характеристика функции: она зависит не только от поведения функции в окрестности рассматриваемой точки x, но и от принимаемых ею значений на всем интервале (a,x) (или (x,b)). Эта нелокальность означает, что изменение плотности потока частиц зависит не только от её значений в окрестности рассматриваемой точки (как это имеет место в случае нормальной диффузии либо классической теплопроводности), но и от её значений в удаленных точках пространства. Случайный процесс, скорость изменения плотности которого зависит от значений плотности в предшествующие моменты времени, называется эредитарным. Такие процессы удобно описывать уравнениями, содержащими дробную производную по времени.

Порядок производной по времени определяется величиной α, характеризующей топологию данного множества [8]. В работе [9] на основании экспериментального изучения фрактальных характеристик осадочных пород приводятся значения показателя α: $0,66 < \alpha < 0,909$. Различают два вида аномального переноса: субдиффузия и супердиффузия. Медленная диффузия (субдиффузия) характеризуется значениями показателя $0 < \alpha < 1$ и возникает на

фрактальных средах, так как частицы диффундирующего вещества вынуждены двигаться по узким каналам сложной конфигурации с тупиками, резкими поворотами и сужениями. Для быстрой диффузии (супердиффузии) показатель α принадлежит интервалу $\left(1;2\right)$. При $\alpha = 1$ имеет место нормальный диффузионный процесс, описывающийся законом Фика [10]. В работе [11] устанавливается связь между аномальной теплопроводностью и диффузией в одномерных системах. Авторы показывают, что случай $\alpha = 1$ соответствует классической теплопроводности, основанной на законе Фурье, супердиффузия соответствует аномальному процессу теплопроводности с расходящимся коэффициентом проводимости, а при $0 < \alpha < 1$ система ведет себя как термоизолятор в термодинамическом пределе.

В последние годы возник значительный интерес исследователей к дробному по времени уравнению диффузии (теплопроводности). Фундаментальное решение фрактального диффузионно-волнового уравнения в одномерном случае получил Ф. Маинарди [12], который также с помощью интегрального преобразования Лапласа решил задачу о распространении начального импульса [13]. В работах [14, 15] решена задача Коши для дробного диффузионного уравнения путем сведения её к интегро-дифференциальному уравнению. Многомерные дробные диффузионно-волновые уравнения рассмотрены в работах [16-19]. Авторы [20] используют дробное по времени диффузионное уравнение для изучения распространения волн напряжения в вязкоупругой среде применительно к акустике и сейсмологии. В статьях [21-23] приводятся решения диффузионных уравнений с дробной производной по времени и соответствующих, возникающих в процессе диффузии (теплопроводности) механических напряжений. В [24] рассматривается двумерное диффузионно-волновое уравнение с дробной производной по времени в смысле Капуто [25-27]. С помощью метода интегральных преобразований аналитически исследованы задачи Дирихле и Неймана, приведены численные расчеты. Условия существования решений граничных задач первого типа для дробного по времени диффузионного уравнения доказаны в работах [28, 29]. Ю. Лучко [30] рассматривает обобщенное диффузионное уравнение с дробной по времени производной Капуто и

применяет метод разделения переменных для решения соответствующей начально-краевой задачи. Этот же метод используется в [31] при решении однородных задач Дирихле и Неймана для диффузионного уравнения с дробными производными по времени и пространственной координате. Происхождение дробных производных Римана-Лиувилля и Капуто, их применение в построении различных моделей вязкоупругости, дается в обзорных статьях [25, 32]. В работе [32] также получено общее решение линейного дифференциального уравнения с дробной производной Капуто.

Что касается пространственной дробной производной, то она входит в состав дифференциального уравнения переноса, если распределение Гаусса классического броуновского движения заменить более общим устойчивым вероятностным распределением Леви [33-35]. Экспериментально доказано, что в некоторых неоднородных средах процессы переноса не могут быть описаны законом Фика и уравнением теплопроводности, экспериментальные данные свидетельствуют о наличии больших «хвостов», связанных с полетами Леви [36, 37]. Последние относятся к случайным блужданиям со смещениями частиц, распределенных в соответствии с устойчивыми законами Леви. Пространственно-дробное уравнение аномальной теплопроводности (диффузии) можно получить, исходя из универсального уравнения (Паули) [38], либо на основании модели случайных блужданий, переходя к макромасштабному пределу. Если имеет место масштабное соотношение $l^{\alpha}/\tau = K$ (l, τ – масштабы длины и времени соответственно), то вероятность обнаружения частицы в заданном интервале стремится к пределу, имеющему плотность распределения, удовлетворяющую дробно-дифференциальному уравнению

$$\partial_t P(x,t) = K \nabla^{\theta}_{x,\alpha} P(x,t).$$

Это означает, что поток частиц удовлетворяет дробному обобщению закона Фика (Фурье) [39, 40].

Пусть X_i, T_i – случайные величины, обозначающие смещение и время ожидания между последовательными прыжками (столкновениями). Тогда величина

X_i/l распределена по α-устойчивому закону $L_{\alpha,\theta}$ с показателем $1<\alpha<2$ и параметром ассиметрии θ. Поток частиц, пересекающих ось x за промежуток времени $[t,t+dt)$ определяется разностью двух выражений

$$Q_l^{\alpha,\theta}P(\cdot,t)(x)=Kl^{-\alpha}\left[\int_0^{+\infty}P(x-y,t)F_{\alpha,\theta}^d\left(\frac{y}{l}\right)dy-\int_0^{+\infty}P(x+y,t)F_{\alpha,\theta}^g\left(\frac{-y}{l}\right)dy\right],$$

где $F_{\alpha,\theta}^d\left(y/l\right)$ – вероятность того, что смещение частицы $y>l$.

Первое слагаемое определяет вероятность пересечения оси справа от x, второе — слева.

В термодинамическом пределе ($l,\tau\to 0$ при $l^\alpha/\tau=K$) оператор потока частиц, представляющих полеты Леви, будет иметь вид [41]

$$Q^*f=\frac{K}{\Gamma(2-\alpha)}\frac{d}{dx}\left[\frac{\sin\frac{\pi}{2}(\alpha-\theta)}{\sin\pi\alpha}\int_{-\infty}^{x}(x-y)^{1-\alpha}f(y)dy+\right.$$
$$\left.+\frac{\sin\frac{\pi}{2}(\alpha+\theta)}{\sin\pi\alpha}\int_{x}^{+\infty}(y-x)^{1-\alpha}f(y)dy\right],$$

где f – концентрация (температура) частиц в момент времени t.

Подставляя выражение для $Q^*P(x,t)$ в закон сохранения $\partial_t P=-\partial_x Q^*P$, получаем дробно-дифференциальное уравнение

$$\partial_t P(x,t)=K\nabla_{x,\alpha}^{\theta}P(x,t).$$

Здесь $\nabla_{x,\alpha}^{\theta}$ – дробная производная Рисса-Феллера [38].

Решению начально-краевых задач для уравнений, содержащих производную Рисса-Феллера, посвящены работы [42-46]. Наиболее популярными и хорошо изученными методами исследования таких задач являются метод интегральных преобразований [46] и численные аппроксимации оператора дробного дифференцирования [33, 42]. Аналитическое решение однородных задач Дирихле и Неймана для дробно-дифференциального уравнения с пространственно-симметричной производной Рисса ($\theta=0$) представлено в работе [31]. В статье [47] применяется метод декомпозиции Адомиана при решении начально-краевой задачи

для дробно-дифференциального уравнения с пространственной производной Капуто.

На протяжении последнего десятилетия все большее внимание исследователей стали привлекать задачи теории упругости на фрактальных средах. Они возникают, например, при рассмотрении моделей микрополярной упругости, градиентных теорий, различных микромеханических моделей. Аэрогели, коллоидные агрегаты, полимеры, некоторые типы композитных материалов, пористые среды и др. имеют мультифрактальную структуру в некоторой области специальных масштабов $R_0 < R < R_s$, где R_0 – минимальный размер частиц, кластеров, пор (микроскопический предел компонентов, образующих фрактальную структуру), R_s – корреляционная длина самоподобия — размер фракталов, похожих друг на друга. Прочность, упругость фрактальных структур являются весьма интересными как с фундаментальной, так и с практической точек зрения, поскольку определяют дебаевскую температуру теплопроводности, прохождение упругих волн и их затухание [48].

Феноменологическое рассмотрение упругих свойств мультифрактальной структуры изложены в [49] в предположении, что приложение внешней силы F к упругому изотропному фрактальному объекту приводит к деформации, осуществляемой на длине определенного масштаба. Таким образом, присутствие внешних напряжений приводит к возникновению необычной характеристической масштабной длины L_F, физически обозначающей размер молекулярных клубков в полимерах, характерный размер ячеек, среднее расстояние между включениями в композитных материалах и может служить характеристикой радиуса корреляции в случайной сетке или аэрогеле. Модуль упругости E и коэффициент Пуассона ν выражаются в виде [49]

$$E = \frac{GD}{d-1},\ \nu = \frac{D}{d-1} - 1,$$

где G – модуль сдвига, D – фрактальная размерность, d – размерность пространства. В настоящее время отсутствуют экспериментальные данные для проверки правильности приведенных выражений для мультифрактальных структур.

Однако установлено, что аэрогель SiO_2 имеет монофрактальную структуру с размерностью $D = 2,3 \pm 0,1$ и коэффициентом Пуассона $\nu = 0,15 \pm 0,05$, рассчитанным в [49]. Это значение довольно близко к экспериментальным данным, приведенным в [50] для аэрогеля SiO_2.

В работе [51] получены основные уравнения упругости для фрактальных сред. В [52] рассмотрено одномерное уравнение движения нелокального типа с дробными производными Капуто, получено его решение с помощью интегральных преобразований Лапласа и Фурье. Уравнения фрактальной термоупругости получены в работе [53]. Ю. Повстенко записал уравнения квазистатической термоупругости, основанные на дробно-диффузионном уравнении теплопроводности [21, 40, 54], а также рассмотрел несколько задач об определении напряжений во фрактальных средах [22, 23, 55].

В настоящее время особый интерес представляют граничные задачи фрактальной теплопроводности, термоупругости, диффузии. В литературе представлены, в основном, только численные методы их решения, основанные на аппроксимациях дробной производной Рисса-Феллера. Поэтому важной задачей является разработка аналитических либо численно-аналитических методов решения дробно-дифференциальных уравнений и соответствующих граничных задач.

1 ГРАНИЧНЫЕ ЗАДАЧИ ФРАКТАЛЬНОЙ ТЕПЛОПРОВОДНОСТИ И ТЕРМОУПРУГОСТИ

1.1 Одномерная начально-краевая задача для дробно-дифференциального уравнения теплопроводности

Рассмотрим дробно-дифференциальное уравнение теплопроводности [47]

$$\frac{\partial^\alpha T}{\partial t^\alpha} = \frac{\partial^\beta T}{\partial x^\beta},\ 0 < \alpha \le 2,\ 1 < \beta \le 2,\ 0 \le x \le 1,\ t > 0, \tag{1.1}$$

с однородными начальными

$$T|_{t=0} = 0,\ 0 < \alpha \le 1,$$

$$T|_{t=0} = \left.\frac{\partial T}{\partial t}\right|_{t=0} = 0,\ 1 < \alpha \le 2, \tag{1.2}$$

и граничными условиями

$$T(0,t) = H(t),\ \left.\frac{\partial T}{\partial x}\right|_{x=1} = 0. \tag{1.3}$$

Здесь $T(x,t)$ – температура, $\partial^\alpha / \partial t^\alpha$, $\partial^\beta / \partial x^\beta$ – операторы дробного дифференцирования Капуто [27], $H(t)$ – функция Хевисайда.

Применим к уравнению (1.1) и граничным условиям (1.3) преобразование Лапласа с учетом условий (1.2). Формула для преобразования Лапласа дробной производной в смысле Капуто имеет вид [12, 13]

$$\mathrm{L}\left[\frac{\partial^\alpha f(t)}{\partial t^\alpha}\right](s) = s^\alpha\,\mathrm{L}[f(t)] - \sum_{k=0}^{n-1} f^{(k)}\left(0^+\right) s^{\alpha-1-k},\ n-1 < \alpha < n.$$

Приходим к следующей граничной задаче

$$\frac{\partial^\beta \overline{T}}{\partial x^\beta} - s^\alpha \overline{T} = 0,\ \overline{T}(x,s) = \mathrm{L}[T(x,t)](s), \tag{1.4}$$

$$\overline{T}(x,s)\big|_{x=0} = \frac{1}{s},\ \left.\frac{\partial \overline{T}(x,s)}{\partial x}\right|_{x=1} = 0, \tag{1.5}$$

где L — оператор преобразования Лапласа, задаваемый интегралом

$$\mathrm{L}[f(t)](s) = \int_0^\infty f(t)e^{-st}dt.$$

Общее решение однородного уравнения (1.4) имеет вид [32]

$$\overline{T}(x,s) = C_1 u_1(\gamma x) + C_2 u_2(\gamma x),$$

где

$$u_1(x) = \int_0^\infty e^{-rx}K_{\beta,0}(r)dr + \frac{2}{\beta}e^{x\cos\frac{\pi}{\beta}}\cos\left[x\sin\frac{\pi}{\beta}\right],$$

$$u_2(x) = \int_0^\infty e^{-rx}K_{\beta,1}(r)dr + \frac{2}{\beta}e^{x\cos\frac{\pi}{\beta}}\cos\left[x\sin\left(\frac{\pi}{\beta}\right) - \frac{\pi}{\beta}\right],$$

$$K_{\beta,l}(r) = \frac{(-1)^l}{\pi}\frac{r^{\beta-1-l}\sin(\beta\pi)}{r^{2\beta} + 2r^\beta\cos(\beta\pi) + 1},\ \gamma = (-s^\alpha)^{1/\beta}.$$

Постоянные C_1 и C_2 определим из граничных условий (1.5).

$$\overline{T}(x,s)\big|_{x=0} = C_1 u_1(0) + C_2 u_2(0) = \frac{1}{s},$$

$$\left.\frac{\partial\overline{T}}{\partial x}\right|_{x=1} = C_1 u_1'(\gamma) + C_2 u_2'(\gamma) = 0$$

Отсюда находим

$$C_1 = \frac{1}{s},\ C_2 = -\frac{u_1'(\gamma)}{s\cdot u_2'(\gamma)} = -\frac{u_1'(\gamma)}{s\cdot u_1(\gamma)}, \tag{1.6}$$

где $u_1'(x) = \int_0^\infty e^{-rx}K_{\beta,-1}(r)dr + \frac{2}{\beta}e^{x\cos\frac{\pi}{\beta}}\cos\left[x\sin\left(\frac{\pi}{\beta}\right) + \frac{\pi}{\beta}\right].$

Окончательно получаем

$$\overline{T}(x,s) = \frac{1}{s}u_1(\gamma x) - \frac{u_1'(\gamma)}{s\cdot u_1(\gamma)}u_2(\gamma x). \tag{1.7}$$

Применяя процедуру обращения преобразования Лапласа [56] к функции (1.7), определяем значения температуры $T(x,t)$.

На рис. 1.1 – 1.5 представлены результаты расчета функции $T(x,t)$ для различных значений параметров α и β. Графики на рис. 1.1, 1.2 демонстрируют зависимость температуры от времени в точке $x = 0,5$ м (середина

рассматриваемого отрезка). При $\alpha = 0,5$, $\beta = 1.9$ (рис. 1.1) наблюдается медленный процесс теплопроводности — температура достигает равновесного значения при $t \approx 12\,c$, в то время как согласно классическому закону теплопроводности ($\alpha = 1$) — при $t \approx 2\,c$. При $\alpha = 1.9$ процесс теплопроводности приобретает волновой характер, в момент времени $t = 1,5\,c$ происходит отражение от границы.

Дробный показатель β учитывает пространственные корреляции, поэтому температура изменяется по-разному при различных значениях β (рис. 1.2), однако термодинамическое равновесие наступает в один и тот же момент времени $t \approx 2\,c$.

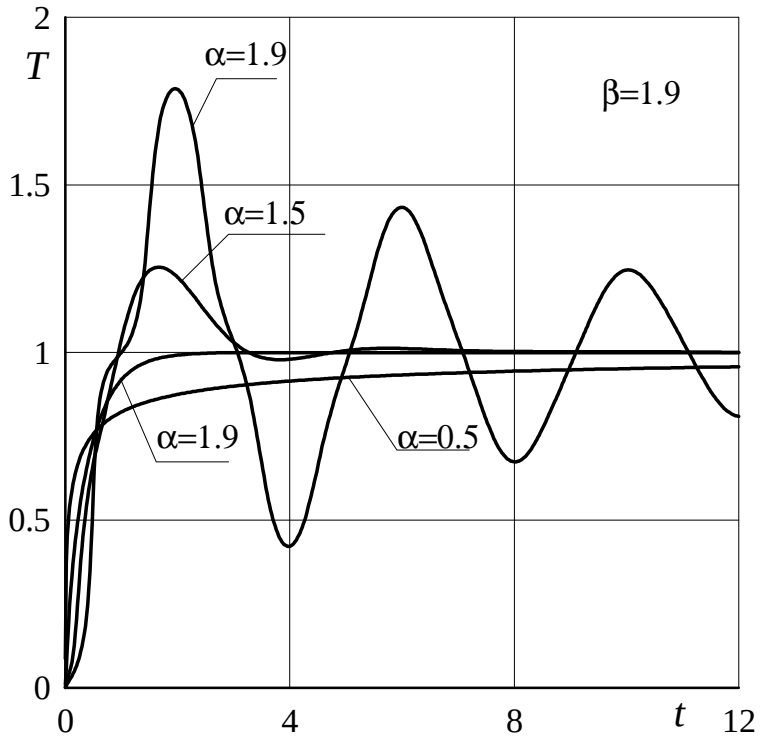

Рисунок 1.1 – Зависимость температуры от времени в точке $x = 0,5$ м при различных значениях α

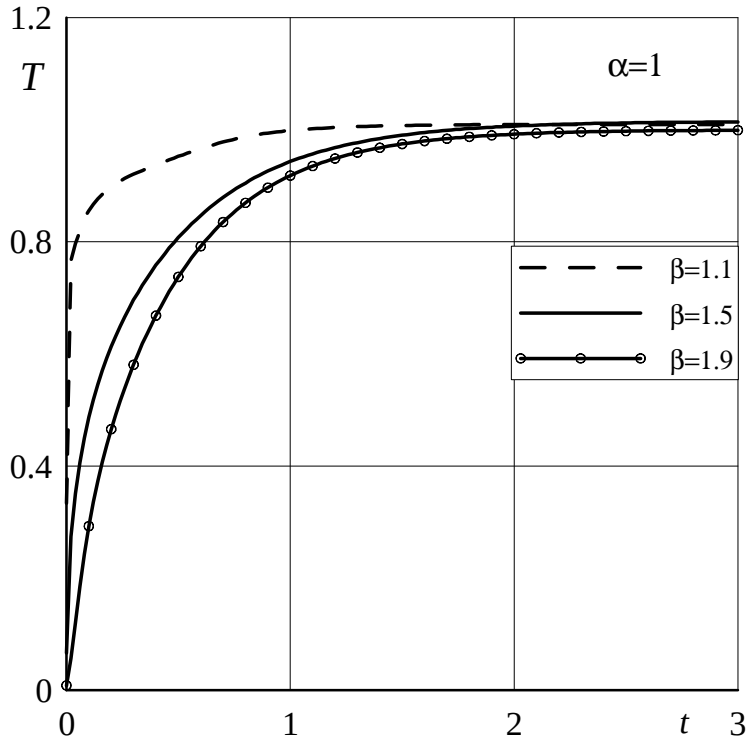

Рисунок 1.2 – Зависимость температуры от времени в точке $x = 0,5$ *м при различных значениях* β

На рис. 1.3-1.5 приведены графики распределения температуры вдоль отрезка $\left[0;1\right]$. При $\alpha = 0,5, \beta = 1.9$ процесс близок к «диффузионному», поэтому в моменты времени $t = 0,5\,c$ (рис. 1.3) и $t = 1,5\,c$ (рис. 1.4) происходит уменьшение температуры при движении вдоль рассматриваемого отрезка. При $x = 1$ м выполняется граничное условие: $T = 1\,K$.

При $\alpha = 1,5$, $\alpha = 1,9$ процесс теплопроводности близок к «волновому». В момент времени $t = 0,5\,c$ температурный нагрев достигает середины отрезка, а в момент $t = 1,5\,c$ в эту точку ($x = 0,5$ м) приходит отраженная от границы $x = 1$ м волна. Поэтому на графиках рис. 1.4 при $\alpha = 1,5$, $\alpha = 1,9$ происходит увеличение температуры.

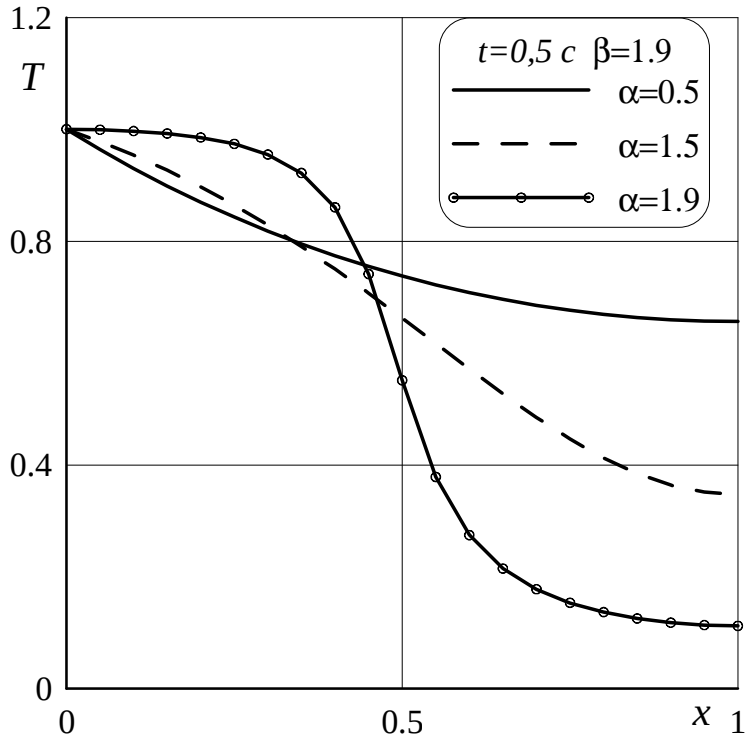

Рисунок 1.3– Распределение температуры вдоль координаты x в момент времени $t = 0,5\,c$ при различных значениях α

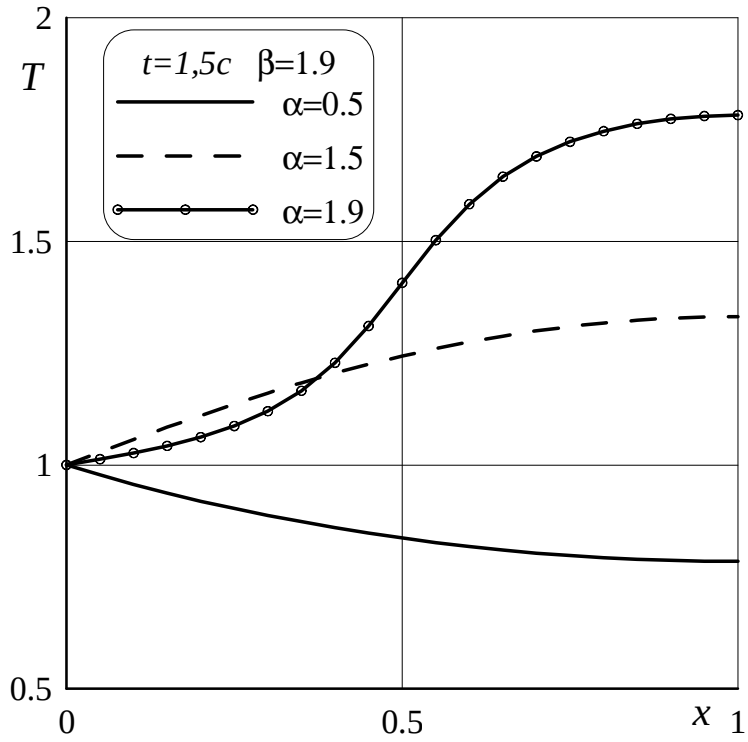

Рисунок 1.4 – Распределение температуры вдоль координаты x в момент времени $t = 1,5\,c$ при различных значениях α

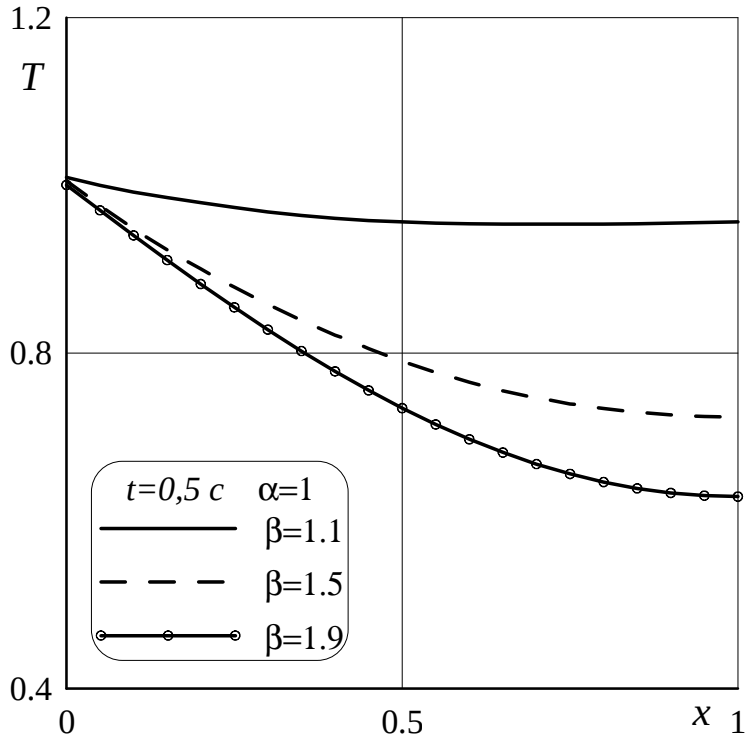

Рисунок 1.5 – Распределение температуры вдоль координаты x в момент времени $t = 0,5\,c$ при различных значениях β

1.2 Решение трехмерной граничной задачи для уравнения теплопроводности с дробной производной по времени

В данном разделе предлагаются аналитические и численные процедуры решения новой трехмерной граничной задачи фрактальной теплопроводности для слоя со сквозной цилиндрической полостью. Приведены численные результаты, характеризующие эволюцию температуры во времени для различных значений порядка производной при действии на поверхности тела импульсной нагрузки.

В прямолинейной декартовой системе координат $Ox_1x_2x_3$ рассмотрим слой $-\infty < x_1, x_2 < \infty$, $|x_3| \le h$, содержащий сквозную цилиндрическую полость $-h \le x_3 \le h, 0 < \rho \le R$. Будем предполагать, что на поверхности полости задан тепловой поток, а на основаниях слоя поддерживается нулевая температура.

Система уравнений, описывающая решение поставленной задачи, имеет вид уравнение теплопроводности [21, 24]

$$\frac{\partial^\alpha T}{\partial t^\alpha} = a\Delta T\,,\ 0 < \alpha \le 2; \tag{1.8}$$

начальные условия

$$T|_{t=0} = 0, \quad 0 < \alpha \le 2\,, \tag{1.9}$$

$$\left.\frac{\partial T}{\partial t}\right|_{t=0} = 0, \quad 1 < \alpha \le 2\,; \tag{1.10}$$

граничные условия на основаниях слоя

$$T|_{x_3=\pm h} = 0; \tag{1.11}$$

граничные условия на поверхности полости [20, 21, 26]

$$-kI^{\alpha-1}\left.\frac{\partial T}{\partial n}\right|_\sigma = P(t), \quad 1 \le \alpha \le 2\,,$$

$$-kD_{\mathrm{RL}}^{1-\alpha}\left.\frac{\partial T}{\partial n}\right|_\sigma = P(t), \quad 0 < \alpha \le 1. \tag{1.12}$$

В соотношениях (1) – (5) $T = T(x_1, x_2, x_3, t)$ – распределение температуры; a – коэффициент температуропроводности, k – коэффициент теплопроводности; $P(t)$ – заданная поверхностная плотность теплового потока, действующего на поверхности σ (поверхность кругового цилиндра); Δ – оператор Лапласа в R^3; $\partial^\alpha \big/ \partial t^\alpha$ – дробная производная Капуто, $I^{\alpha-1}$ и $D_{\mathrm{RL}}^{1-\alpha}$ – операторы дробного интегрирования и дифференцирования Римана-Лиувилля соответственно [13, 25, 27].

Решение граничной задачи проведено с применением интегрального преобразования Лапласа по времени и процедуры метода однородных решений [54]. Асимптотическое решение (при малых временах) определено в замкнутом виде. Решение для произвольного момента времени строилось путем численного обращения преобразования Лапласа [57].

Применим к соотношениям (1.8), (1.11), (1.12) преобразование Лапласа по времени при однородных начальных условиях (1.9), (1.10). Приходим к следующей граничной задаче

$$a\Delta\Theta - s^\alpha\Theta = 0\,,\ \Theta = L[T](s); \tag{1.13}$$

$$\Theta|_{x_3=\pm h} = 0; \tag{1.14}$$

$$\left.\frac{\partial\Theta}{\partial n}\right|_{\sigma} = -\frac{s^{\alpha-1}}{k}Q(s),\ Q(s) = L[P(t)]. \tag{1.15}$$

Симметричные однородные решения задачи (1.13), (1.14) будут иметь вид [54]

$$\Theta = \sum_{j=0}^{\infty}\varphi_j(x_1,x_2)\cos(\mu_j x_3), \tag{1.16}$$

где $\mu_j = \frac{\pi}{2h}(2j+1)$, $\varphi_j(x_1,x_2)$ – метагармонические функции, удовлетворяющие уравнению

$$(\nabla^2 - \nu_j^2)\varphi_j = 0,\ \nu_j^2 = \mu_j^2 + \frac{s^{\alpha}}{a},\ \nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}. \tag{1.17}$$

Раскладывая в (1.15) функцию $Q(s)$ в ряд Фурье по собственным функциям однородного решения (1.16), получим

$$\left.\frac{\partial\varphi_j}{\partial\rho}\right|_{\rho=R} = -\frac{s^{\alpha-1}}{k}Q_j,\ Q = \sum_{j=0}^{\infty}Q_j(x_1,x_2)\cos(\mu_j x_3). \tag{1.18}$$

Таким образом, задача состоит в определении функций $\varphi_j(\rho)$ из уравнения (1.17) и граничных условий (1.18).

С учетом осевой симметрии рассматриваемой задачи уравнение (1.17) можно записать следующим образом

$$\left(\frac{\partial^2}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial}{\partial\rho} - \nu_j^2\right)\varphi_j = 0,\ \rho \geq R.$$

Его решение представимо в виде

$$\varphi_j(\rho,s) = C_1(s)I_0(\nu_j\rho) + C_2(s)K_0(\nu_j\rho),$$

где $\mathrm{Re}\,\nu_j > 0$, I_0 и K_0 – модифицированные функции Бесселя 1-го и 2-го рода соответственно [58].

Так как температурное поле должно на бесконечности обращаться в ноль, то $C_1(s) = 0$, а коэффициенты $C_2(s)$ определим из условия (1.18).

Окончательно получаем

$$\Theta = \sum_{j=0}^{\infty} \frac{s^{\alpha-1}}{k\nu_j} Q_j \frac{K_0(\nu_j \rho)}{K_1(\nu_j R)} \cos(\mu_j x_3).$$

Для определения температурной функции T по образу Θ необходимо выполнить обратное преобразование Лапласа.

Асимптотика решения при малых значениях времени ($t \to 0$). Известно, что при преобразовании Лапласа малым значениям временного параметра t соответствуют большие значения параметра преобразования s (т.е. $s \to \infty$). Воспользовавшись этим замечанием, представим числа ν_j приближенно

$$\nu_j^2 = \mu_j^2 + \frac{s^\alpha}{a} \approx \frac{s^\alpha}{a} \Rightarrow \nu_j \approx \frac{s^{\alpha/2}}{\sqrt{a}},\ a > 0{,}1 \le \alpha \le 2.$$

Используя данное представление и асимптотические разложения функций Макдональда при больших значениях аргумента, получим следующее выражение для функции $\Theta(s)$

$$\Theta = \sum_{j=0}^{\infty} \frac{\sqrt{a}}{k} Q_j \sqrt{\frac{\rho}{R}} s^{\alpha/2-1} \exp\left(-s^{\alpha/2}\tau\right) \cos(\mu_j x_3),\ \tau = \frac{\rho - R}{\sqrt{a}}. \qquad (1.19)$$

Пусть на поверхности полости действует прямоугольный по времени и распределенный по параболе вдоль "толщинной" координаты импульс

$$P(t, x_3) = \left(1 - \frac{x_3^2}{h^2}\right)(H(t) - H(t - L)),$$

где $H(t)$ – единичная функция Хевисайда [58], L – длина импульса.

Тогда

$$Q_j(s) = \frac{4 \cdot (-1)^{j+1}}{\mu_j^3} \cdot \frac{(1 - \exp(-sL))}{s},$$

а представление (1.19) приобретает вид

$$\Theta = \sum_{j=0}^{\infty} \frac{\sqrt{a}}{k} \cdot \sqrt{\frac{\rho}{R}} \frac{4 \cdot (-1)^{j+1}}{\mu_j^3} \cdot \frac{(1 - \exp(-sL))}{s} \times$$

$$\times s^{\alpha/2-1} \exp\left(-s^{\alpha/2}\tau\right) \cos(\mu_j x_3). \qquad (1.20)$$

Рассмотрим несколько частных случаев.

1. Классическое (параболическое) уравнение теплопроводности $(\alpha = 1)$.

Функцию (1.20) запишем теперь в виде

$$\Theta = \sum_{j=0}^{\infty} \frac{\sqrt{a}}{k} \cdot \sqrt{\frac{\rho}{R}} \frac{4 \cdot (-1)^{j+1}}{\mu_j^3} \cdot \frac{(1 - \exp(-sL))}{s\sqrt{s}} \cdot \exp(-\tau\sqrt{s}) \cos\left(\mu_j x_3\right).$$

Обращение преобразования Лапласа последнего выражения можно выполнить в замкнутом виде. Получим

$$T(t, x_3, \rho) = \sum_{j=0}^{\infty} T_j (f(t) - f(t-L) H(t-L)) \cos\left(\mu_j x_3\right) \tag{1.21}$$

где

$$f(t) = \frac{1}{\sqrt{\pi}} \sqrt{t} \exp\left(\frac{-\tau^2}{4t}\right) - \tau \operatorname{Erfc}\left(\frac{\tau}{2\sqrt{t}}\right), \; \tau = \frac{\rho - R}{\sqrt{a}},$$

$$T_j = \frac{\sqrt{a}}{k} \cdot \frac{4 \cdot (-1)^{j+1}}{\mu_j^3} \cdot \sqrt{\frac{\rho}{R}}.$$

2. Волновое уравнение $(\alpha = 2)$.

В этом случае решение получаем в виде

$$T(t, x_3, \rho) = \sum_{j=0}^{\infty} T_j (H(t-\tau) - H(t-L-\tau) H(t-L-\tau)) \cos\left(\mu_j x_3\right) \tag{1.22}$$

На рис. 1.6, 1.7 представлены расчеты температурной функции в зависимости от времени, выполненные по асимптотическим (при малых значениях времени) формулам (1.21), (1.22) соответственно. На рис. 1.8, 1.9 приведены кривые эволюции температуры для различных дробных значений α. Решение для произвольного момента времени строилось путем численного обращения преобразования Лапласа. Алгоритм основан на разложении изображения в ряд Фурье и аппроксимации с помощью цепных дробей [57].

При расчетах задавались следующие параметры: $x_3 = 0, h = 1, R = 1, \rho = 1, L = 1$.

Анализируя графики, изображенные на рис. 1.8, можно сделать вывод, что распространение тепла (нагрев) в слое с полостью для сред с показателем $\alpha < 1$ осуществляется медленнее в отличие от сред, для которых $\alpha = 1$. При $1 < \alpha \leq 2$

(рис. 1.9) процесс, происходящий во фрактальной среде, можно охарактеризовать как процесс перехода от классической теплопроводности к волновому поведению.

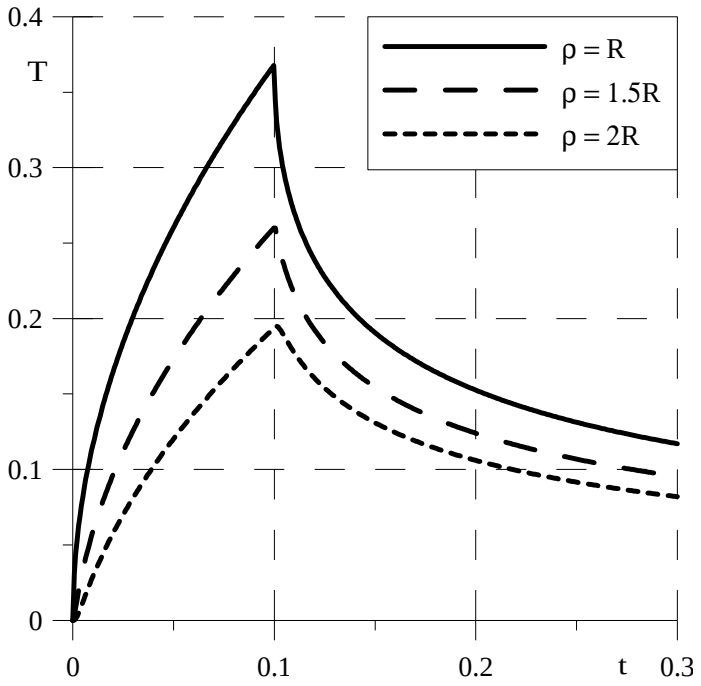

Рисунок 1.6. – Изменение температуры во времени при $\alpha = 1$ *(асимптотическое решение)*

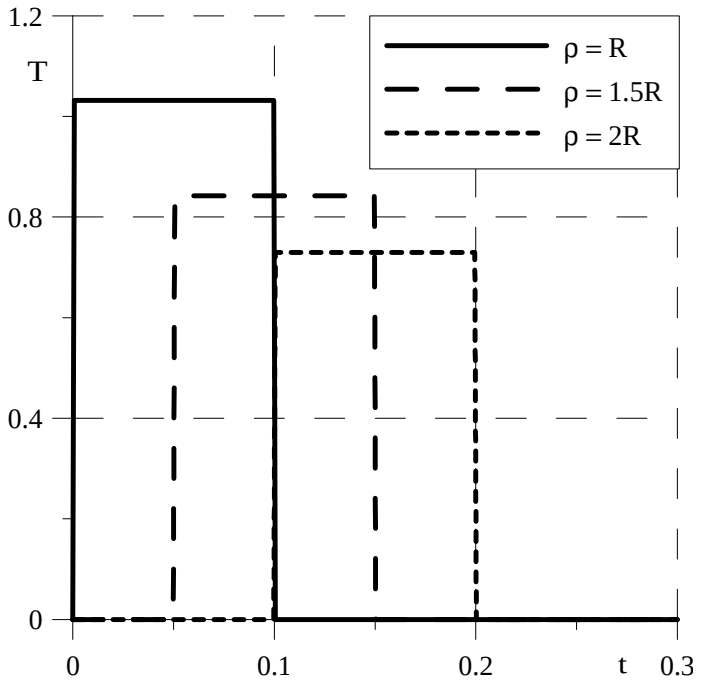

Рисунок 1.7 – Изменение температуры во времени при $\alpha = 2$ *(асимптотическое решение)*

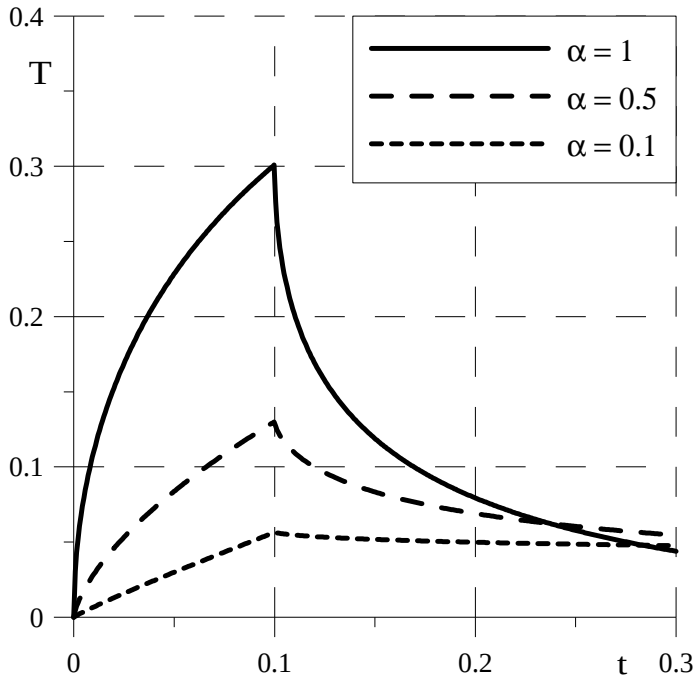

Рисунок 1.8 – Зависимость температуры от времени при различных значениях $0 < \alpha \leq 1$

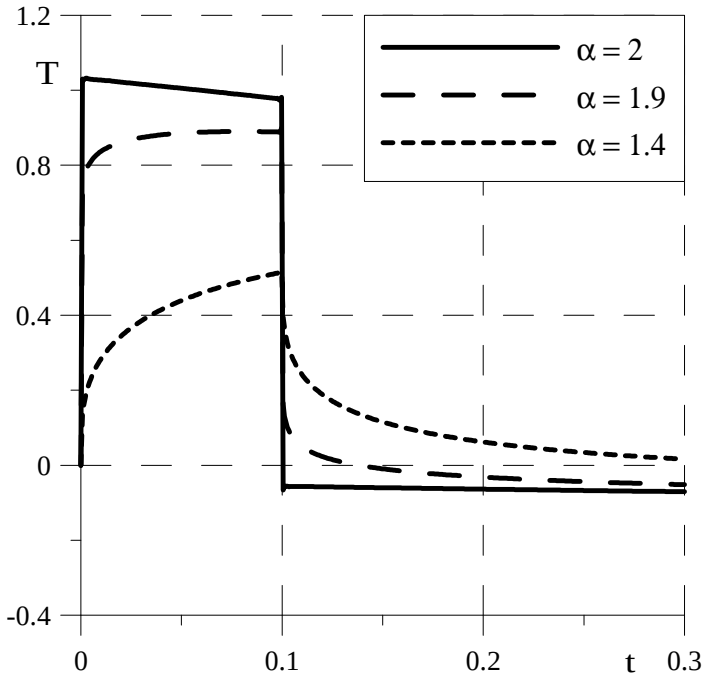

Рисунок 1.9 – Зависимость температуры от времени при различных значениях $1 < \alpha \leq 2$

1.3 Квазистатическая задача термоупругости для дифференциального уравнения теплопроводности с производными Рисса и Капуто

Рассмотрим полупространство, граница которого нагревается скачкообразно. При этом предполагается, что начальные условия для температуры однородны. Тогда уравнения соответствующей начально-краевой задачи квазистатической термоупругости имеют вид [21, 40]

уравнение теплопроводности

$$\frac{\partial^\alpha T}{\partial t^\alpha} - a\,\mathrm{R}^\beta_{x_1}(T(x_1,t)) = 0,\ 0<\alpha<2,\ 1<\beta<2,\ x_1>0,\ t>0, \tag{1.23}$$

уравнения движения

$$\mu\frac{\partial^2 u_i}{\partial x_j^2} + (\lambda+\mu)\frac{\partial^2 u_j}{\partial x_j \partial x_i} = \gamma\frac{\partial T}{\partial x_i}, \tag{1.24}$$

закон Гука

$$\sigma_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \delta_{ij}\lambda div\vec{u} - \gamma T\ ,\ i,j = 1,2,3 \tag{1.25}$$

начальные условия

$$T(x_1,0) = 0,\ 0<\alpha\le 1$$

$$T(x_1,0) = \left.\frac{\partial T(x_1,t)}{\partial t}\right|_{t=0} = 0,\ 1<\alpha\le 2, \tag{1.26}$$

граничные условия

$$T(0,t) = P(t). \tag{1.27}$$

Здесь $T(x_1,t)$ – температурная функция, u_i – компоненты вектора перемещений, λ,μ – постоянные Ламе, δ_{ij} – символ Кронекера, a – коэффициент температуропроводности, γ – коэффициент теплового расширения, σ_{ij} – компоненты тензора напряжений, $\partial^\alpha/\partial t^\alpha$ – оператор дробной производной в смысле Капуто [13, 27], $\mathrm{R}^\beta_{x_1}$ – оператор дробной конечной производной Рисса [26, 59], $P(t)$ – заданная функция, скачкообразно изменяющаяся в нуле.

Вводя потенциал термоупругого перемещения $\Phi(x_1,t)$ [60], из уравнений (1.24), (1.25) получим

$$u_i = \frac{\partial \Phi}{\partial x_i},\ \Delta\Phi = \frac{\gamma}{\lambda+2\mu}T,\ \sigma_{ij} = 2\mu\left(\frac{\partial^2}{\partial x_i \partial x_j} - \delta_{ij}\Delta\right)\Phi.$$

Тогда для данной задачи имеем

$$\sigma_{11} = \sigma_{12} = \sigma_{13} = \sigma_{23} = 0,$$

$$\sigma_{22} = \sigma_{33} = -2\mu\frac{d^2\Phi}{dx_1^2} = -2\mu m T,\ m = \frac{\gamma}{\lambda+2\mu}. \tag{1.28}$$

Таким образом, для получения напряжений в рассматриваемой области необходимо определить температуру из уравнения (1.23), начальных условий (1.26) и граничных условий (1.27).

Уравнение теплопроводности.

Применим преобразование Лапласа по t к уравнению (1.23) и условию (1.27) при выполнении начальных условий (1.26).

Приходим к следующей граничной задаче

$$a\,\mathrm{R}_{x_1}^{\beta} T - s^{\alpha}\overline{T} = 0, \tag{1.29}$$

$$\overline{T}(0,s) = \overline{P}(s), \tag{1.30}$$

где $\overline{f}(x_1,s) = \mathrm{L}[f(x_1,t)](s)$.

Далее обратимся к разложению конечной производной Рисса, которое дается в работе [59]

$$\mathrm{R}_x^{\beta}\varphi(x) = \frac{\mathrm{I}_{0+}^{2-\beta}\dfrac{d^2\varphi}{dx^2} + \mathrm{I}_{-}^{2-\beta}\dfrac{d^2\varphi}{dx^2}}{2\cos\dfrac{(2-\beta)\pi}{2}},\ 1<\beta<2, \tag{1.31}$$

где

$$\mathrm{I}_{0+}^{\alpha}\varphi(x) = \frac{1}{\Gamma(\alpha)}\int_0^x (x-\zeta)^{\alpha-1}\varphi(\zeta)\,d\zeta,$$

$$\mathrm{I}_{-}^{\alpha}\varphi(x) = \frac{1}{\Gamma(\alpha)}\int_x^{\infty} (\zeta-x)^{\alpha-1}\varphi(\zeta)\,d\zeta$$

— операторы дробного интегрирования Римана-Лиувилля, $\alpha > 0$.

Для того, чтобы получить формулу $\sin$-преобразования Фурье производной Рисса, запишем следующие соотношения [27]

$$\mathrm{F}_s\left(\mathrm{I}_{0+}^{\alpha}\varphi\right) = \xi^{-\alpha}\left(-\sin\frac{\alpha\pi}{2}\mathrm{F}_c\,\varphi + \cos\frac{\alpha\pi}{2}\mathrm{F}_s\,\varphi\right),$$

$$\mathrm{F}_s\left(\mathrm{I}_{-}^{\alpha}\varphi\right) = \xi^{-\alpha}\left(\sin\frac{\alpha\pi}{2}\mathrm{F}_c\,\varphi + \cos\frac{\alpha\pi}{2}\mathrm{F}_s\,\varphi\right),\ 0 < \alpha < 1,$$

$$\mathrm{F}_s(\varphi) = \int_0^{\infty}\varphi(x)\sin\xi x dx,\ \mathrm{F}_c(\varphi) = \int_0^{\infty}\varphi(x)\cos\xi x dx. \tag{1.32}$$

Применим к обеим частям (1.31) $\sin$-преобразование Фурье, с учетом формул (1.32) получим

$$\mathrm{F}_s\left[\mathrm{R}_{x_1}^{\beta}\varphi(x)\right] = -\xi^{\beta}\,\mathrm{F}_s[\varphi] + \xi^{\beta-1}\varphi(0),\ 1 < \beta < 2. \tag{1.33}$$

Теперь преобразуем уравнение (1.29) с помощью синус-преобразования Фурье. Применяя полученное соотношение (1.33) и граничное условие (1.30), приходим к следующему алгебраическому уравнению

$$\xi^{\beta}\,\mathrm{F}_s\left[\overline{T}(x_1,s)\right] + \frac{s^{\alpha}}{a}\mathrm{F}_s\left[\overline{T}(x_1,s)\right] = \xi^{\beta-1}\overline{P}(s).$$

Решение последнего уравнения дает

$$\mathrm{F}_s\left[\overline{T}(x_1,s)\right] = \frac{\xi^{\beta-1}a\overline{P}(s)}{a\xi^{\beta}+s^{\alpha}}. \tag{1.34}$$

Для того, чтобы получить искомую температурную функцию, необходимо к правой части (1.34) применить обратные операторы F_s^{-1}, L^{-1}.

Обозначим

$$f(t) = \mathrm{L}^{-1}\left[\frac{1}{a\xi^{\beta}+s^{\alpha}}\right].$$

Тогда

$$\mathrm{L}^{-1}\left[\frac{\overline{P}(s)}{a\xi^{\beta}+s^{\alpha}}\right] = \int_0^{t} f(\tau)P(t-\tau)d\tau \tag{1.35}$$

Имеет место соотношение [24, 31]

$$\mathrm{L}^{-1}\left[\frac{1}{s^{\alpha}+b}\right]=t^{\alpha-1}E_{\alpha,\alpha}(-bt^{\alpha}),\ b>0,\ \alpha>0, \tag{1.36}$$

где $E_{\alpha,\beta}(x)$ – обобщенная функция Миттаг-Лефлера [61].

Выполняя в (1.34) обращение преобразования Лапласа и синус-преобразования Фурье, с учетом (1.35) (1.36) получаем

$$T(x_1,t)=\frac{2a}{\pi}\int_0^{\infty}\xi^{\beta-1}\left(\int_0^{t}\tau^{\alpha-1}E_{\alpha,\alpha}\left(-a\xi^{\beta}\tau^{\alpha}\right)P(t-\tau)\,d\tau\right)\sin\xi x_1 d\xi\,. \tag{1.37}$$

Данное выражение и дает решение поставленной задачи. Несобственный интеграл в правой части (1.37) является сходящимся, однако его вычисление вызывает некоторые трудности, так как ряд, представляющий функцию $E_{\alpha,\alpha}\left(-a\xi^{\beta}\tau^{\alpha}\right)$, плохо суммируется при больших значениях ее аргумента. Дело в том, что каждый член данного знакопеременного ряда имеет большое абсолютное значение, в то время как значение обобщенной функции Миттаг-Лефлера (суммы данного ряда) — величина малая. Подобная проблема возникает при вычислении функции Райта [62]. В данной работе при вычислении функции (1.37) используется асимптотическое представление функции Миттаг-Лефлера.

Численная реализация.

В качестве примера рассмотрим следующую задачу. Пусть граница полупространства нагревается скачкообразно в начальный момент времени, т.е.

$$P(t)=\begin{cases}1, & t\geq 0,\\ 0, & t<0.\end{cases} \tag{1.38}$$

Функция Миттаг-Лефлера, входящая в состав (1.37), представима рядом

$$E_{\alpha,\alpha}(-z)=\sum_{n=0}^{\infty}\frac{(-z)^n}{\Gamma(\alpha n+\alpha)},\ z>0\,. \tag{1.39}$$

При больших абсолютных значениях аргумента имеет место асимптотическое представление [61]

$$E_{\alpha,\alpha}(-z) = -\sum_{n=2}^{N} \frac{(-z)^{-n}}{\Gamma(\alpha - \alpha n)} + O\left(z^{-N}\right),\ z > 0,\ z \to \infty. \tag{1.40}$$

С учетом (1.39), (1.40) получаем

$$E_{\alpha,\alpha}(-z) \approx \begin{cases} \sum_{n=0}^{M} \frac{(-z)^{n}}{\Gamma(\alpha n + \alpha)}, & z < z_0, \\ -\sum_{n=2}^{N} \frac{(-z)^{-n}}{\Gamma(\alpha - \alpha n)}, & z \geq z_0. \end{cases} \tag{1.41}$$

Значение z_0 определяется экспериментально и зависит от значения параметра α. Далее, подставляя представление (1.41) и функцию (1.38) в (1.37), получаем следующие формулы для приближенного вычисления температуры

$$T(x_1,t) \approx \frac{2a}{\pi} \int_0^{\infty} K(t,\xi)\,\xi^{\beta-1} \sin \xi x_1 d\xi,\ x_1 > 0,\ t > 0,$$

$$K(t,\xi) = \begin{cases} t^{\alpha} \sum_{n=0}^{M_1} \frac{\left(-a\xi^{\beta}t^{\alpha}\right)^{n}}{(\alpha + \alpha n)\Gamma(\alpha + \alpha n)}, & -a\xi^{\beta}t^{\alpha} < z_0, \\ \frac{1}{-a\xi^{\beta}} \left[\sum_{n=0}^{M_2} \frac{z_0^{n+1}}{(\alpha + \alpha n)\Gamma(\alpha + \alpha n)} + \sum_{n=0}^{M_3} \frac{z_0^{-n+1}}{(\alpha - \alpha n)\Gamma(\alpha - \alpha n)} \right] - \\ \qquad - t^{\alpha} \sum_{n=2}^{M_4} \frac{\left(-a\xi^{\beta}t^{\alpha}\right)^{-n}}{(\alpha - \alpha n)\Gamma(\alpha - \alpha n)}, & -a\xi^{\beta}t^{\alpha} \geq z_0. \end{cases} \tag{1.42}$$

Константы M_1, M_2, M_3, M_4 определяются необходимой точностью вычислений и скоростью сходимости соответствующих рядов.

Ниже представлены графики зависимости температуры T и напряжения σ_{22} от времени и пространственной координаты при разных значениях α и β.

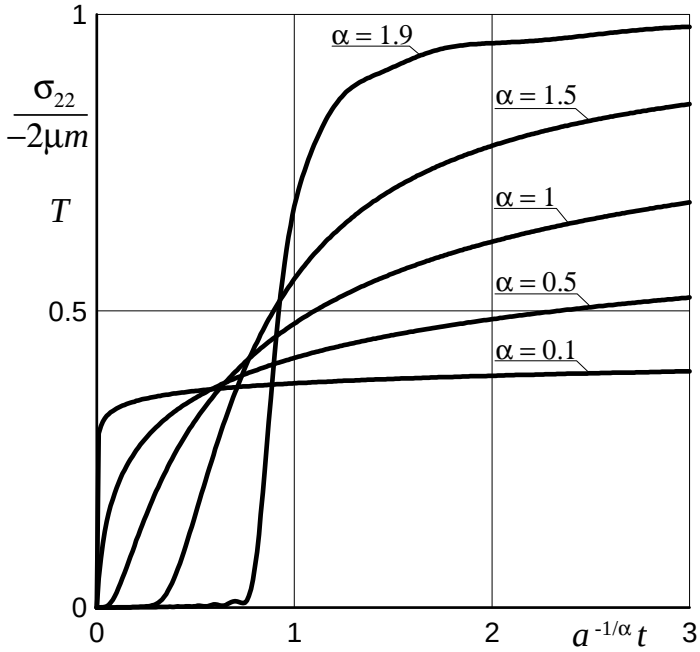

Рисунок 1.10 – Зависимость температуры и напряжения от времени при различных значениях α

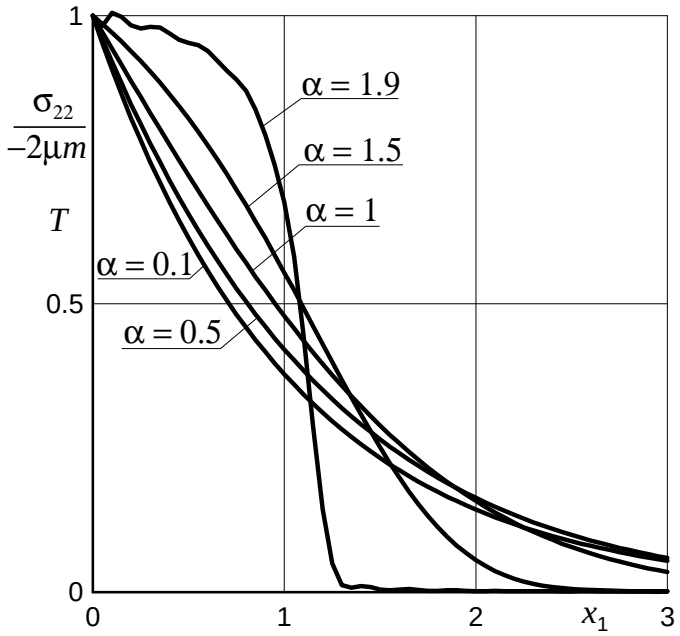

Рисунок 1.11 – Распределение температуры и напряжения вдоль пространственной координаты при различных значениях α

На рис. 1.10 показано изменение температуры по времени на расстоянии 1 м от границы полупространства при $\beta = 0.99$. Эти условия соответствуют такому процессу теплопереноса, при котором учитывается влияние памяти: температура определяется её эволюцией в течение всего предшествующего периода теплопроводности. Поэтому при малых α происходит мгновенное нагревание и установление граничной температуры (равной единице), а при $\alpha \to 2$ нагревание в

рассматриваемой точке происходит только через некоторое время. То же наблюдается и на графиках рис. 1.11, только речь идет об уменьшении температуры при удалении от границы в данный момент времени наблюдения t = 1 с. В работе [11] указано, что при $0 < \alpha < 1$ система ведет себя как термоизолятор, а при $1 < \alpha < 2$ мы имеем аномальную теплопроводность с коэффициентом проводимости, стремящемся к бесконечности в термодинамическом пределе. Случай $\alpha = 1$ соответствует нормальной (классической) теплопроводности, описанной законом Фурье.

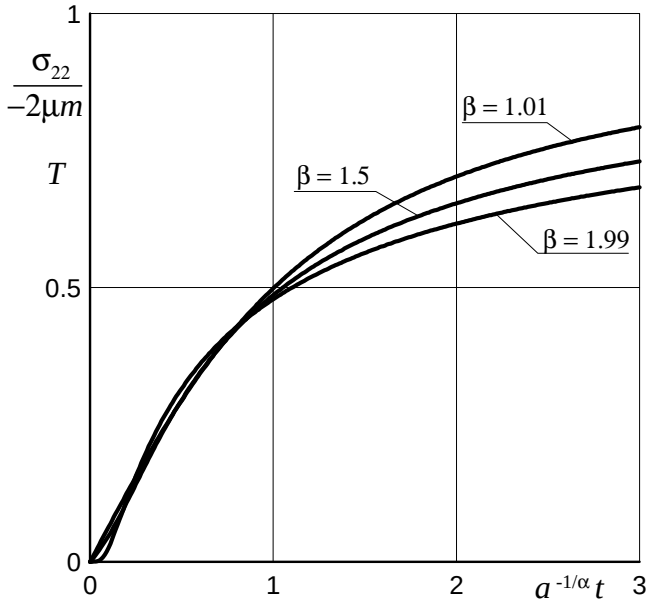

Рисунок 1.12 - Зависимость температуры и напряжения от времени при различных значениях β

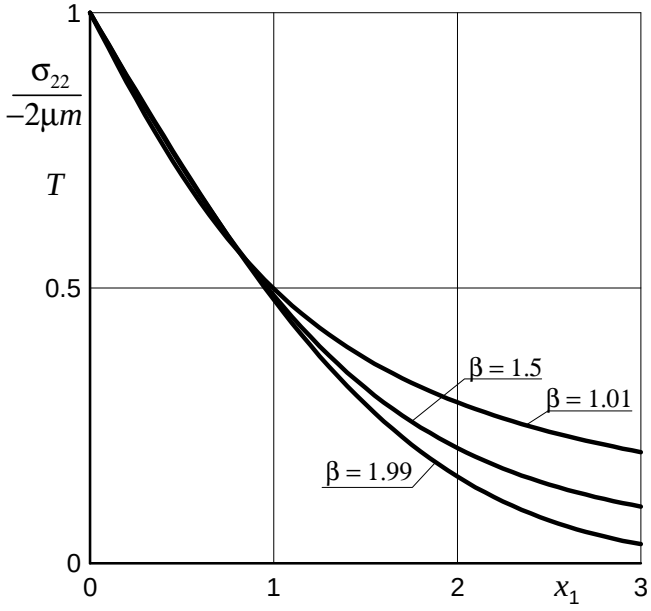

Рисунок 1.13 – Распределение температуры и напряжения вдоль пространственной координаты при различных значениях β

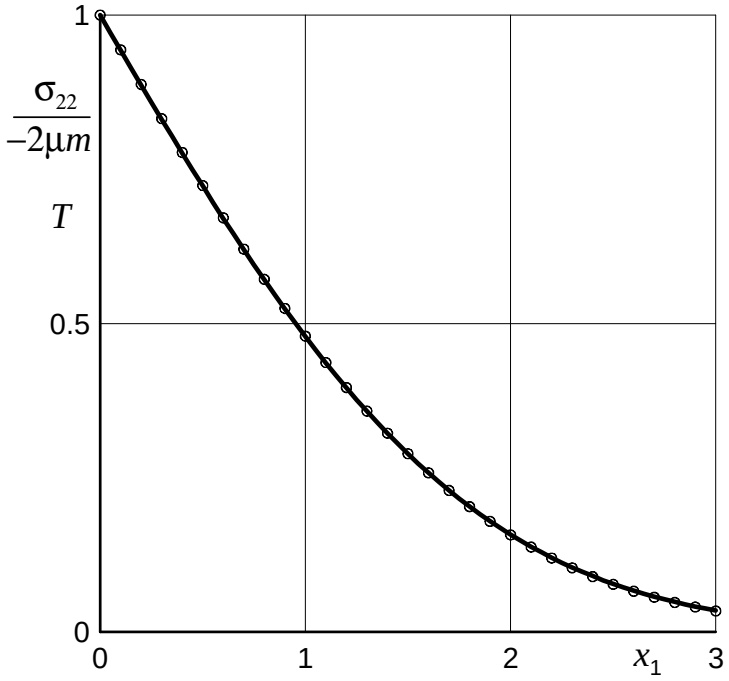

Рисунок 1.14 – Сравнение решений классической и фрактальной задач термоупругости

На рис. 1.12, 1.13 представлены графики зависимости температуры от времени и координаты при различных β и $\alpha = 1$. Здесь рассматривается процесс теплопроводности, учитывающий пространственную нелокальность, т.е. изменение температуры зависит не только от ее значений в окрестности рассматриваемой точки, но и от значений в удаленных точках пространства. Поэтому на приведенных графиках различия, в основном, сказываются при установлении процесса (по времени), либо на удалении от границы полупространства (при $x_1 > 1$).

При $\alpha = 1,\ \beta = 2$ уравнение (1.23) переходит в классическое уравнение теплопроводности, основанное на законе теплопроводности Фурье. Его решение при выполнении начальных условий (1.26) и граничного условия (1.27), (1.38), имеет вид [60]

$$T(x_1, t) = \operatorname{erfc}\left(\frac{x_1}{2\sqrt{at}}\right),\ t > 0\,. \tag{1.43}$$

На рис. 1.14 показано сравнение решения (1.42) при $\alpha = 1,\ \beta \to 2$ с известным решением (1.43), которое нанесено точками.

1.4 Обобщение задачи Даниловской

Первой публикацией по динамическим задачам термоупругости была задача Даниловской [63]. В ней рассматривается внезапное нагревание границы упругого полупространства. В момент $t = 0$ плоскость $x_1 = 0$, ограничивающая упругое полупространство $x_1 \geq 0$, внезапно нагревается до температуры T_0, которая затем остается постоянной. Этот процесс обычно называется тепловым ударом [60]. При этом предполагается, что плоскость $x_1 = 0$ свободна от напряжений и что начальные условия для температуры и перемещений однородны. Под влиянием внезапного нагревания плоскости $x_1 = 0$ в упругом полупространстве распространяется одномерная термоупругая волна. Задача состоит в определении напряжений и температуры из системы дифференциальных уравнений

нелокальное уравнение теплопроводности [21]

$$\frac{\partial^{\alpha} T}{\partial t^{\alpha}} - a\frac{\partial^2 T}{\partial x_1^2} = 0,\ T = T(x_1,t),\ 0 < \alpha \leq 1, \tag{1.44}$$

нелокальное волновое уравнение для потенциала термоупругого перемещения [3]

$$\left(\frac{\partial^2}{\partial x_1^2} - \frac{1}{c_1^2}\frac{\partial^{\alpha+1}}{\partial t^{\alpha+1}}\right)\Phi(x_1,t) = mT,\ u_i = \frac{\partial \Phi}{\partial x_i},\ m = \frac{\gamma}{\lambda + 2\mu}, \tag{1.45}$$

начальных условий

$$T(x_1,0) = 0,\ u_i(x_1,0) = 0, \tag{1.46}$$

граничных условий

$$\sigma_{11}(0,t) = 0,\ T(0,t) = T_0 H(t), \tag{1.47}$$

где $\partial^{\alpha}/\partial t^{\alpha}$ – оператор дробного дифференцирования Капуто [27], a – коэффициент температуропроводности, T – температура, σ_{ij} – тензор напряжений, u_i – компоненты вектора перемещений $\left(i, j = 1,2,3\right)$, λ, μ – постоянные Ламе, γ – коэффициент теплового расширения, c_1 – скорость распространения упругой волны расширения (сжатия), $H(t)$ – функция Хевисайда.

В рамках поставленной задачи напряжения выражаются через потенциал $\Phi(x_1,t)$ следующим образом

$$\sigma_{11} = \rho\frac{\partial^{\alpha+1}\Phi}{\partial t^{\alpha+1}},\ \sigma_{22} = \sigma_{33} = -2\mu m T + \frac{\lambda}{c_1^2}\frac{\partial^{\alpha+1}\Phi}{\partial t^{\alpha+1}}, \tag{1.48}$$

а граничное условие для σ_{11} в (1.47) будет иметь вид

$$\left.\frac{\partial^{\alpha+1}\Phi(x_1,t)}{\partial t^{\alpha+1}}\right|_{x_1=0} = 0. \tag{1.49}$$

Применим к уравнениям (1.44), (1.45) и граничным условиям (1.47), (1.49) преобразование Лапласа при выполнении начальных условий (1.46). Получим

$$\overline{T}''_{x_1} - \frac{s^\alpha}{a}\overline{T} = 0,\ \overline{T}(0,s) = T_0, \tag{1.50}$$

$$\overline{\Phi}''_{x_1} - \frac{s^{\alpha+1}}{c_1^2}\overline{\Phi} = m\overline{T}, \tag{1.51}$$

$$s^{\alpha+1}\overline{\Phi}(0,s) = 0, \tag{1.52}$$

где $\overline{f} = \mathrm{L}[f(x_1,t)](s) = \int\limits_0^\infty f(x_1,t)e^{-st}dt$.

Решением граничной задачи (1.50) является функция

$$\overline{T}(x_1,s) = T_0\exp\left(-\frac{s^{\alpha/2}}{\sqrt{a}}x_1\right). \tag{1.53}$$

После подстановки (1.53) в неоднородное уравнение (1.51) получаем

$$\overline{\Phi}''_{x_1} - \frac{s^{\alpha+1}}{c_1^2}\overline{\Phi} = mT_0\exp\left(-\frac{s^{\alpha/2}}{\sqrt{a}}x_1\right). \tag{1.54}$$

Запишем общее решение соответствующего однородного уравнения

$$\overline{\Phi}^0(x_1,s) = A\exp(k_1x_1) + B\exp(-k_1x_1),\ k_1 = \frac{s^{(\alpha+1)/2}}{c_1}.$$

Константы A и B определим из граничного условия (1.52) и условия затухания на бесконечности. Получим

$$\overline{\Phi}^0(x_1,s) = \exp(-k_1x).$$

Частное решение неоднородного уравнения (1.54) будем искать в виде

$$\overline{\Phi}^{*}(x_1,s) = C\exp(-k_2 x),\ k_2 = \frac{s^{\alpha/2}}{\sqrt{a}}.$$

Подставляя последнее выражение в уравнение (1.54), находим константу C. Тогда решение (1.54) примет вид

$$\overline{\Phi}(x_1,s) = \frac{mT_0c_1^2}{\left(s^{\alpha}\cdot c_1^2/a - s^{\alpha+1}\right)}\left(e^{-k_1x_1} - e^{-k_2x_1}\right). \qquad (1.55)$$

С помощью представлений (1.48) и найденной функции (1.55) получаем выражения для напряжений в плоскости преобразования Лапласа

$$\overline{\sigma}_{11} = \rho s^{\alpha+1}\overline{\Phi},\ \overline{\sigma}_{22} = \overline{\sigma}_{33} = -2\mu m\overline{T} + \frac{\lambda}{c_1^2}s^{\alpha+1}\overline{\Phi}.$$

Теперь, выполняя обращение преобразования Лапласа, получаем значения напряжений.

На рис. 1.15 представлено изменение напряжения σ_{11} в зависимости от расстояния x_1 при $t = a/c_1^2$, $T_0 = 1K$.

При $\alpha = 1$ получаем частный случай задачи В.И. Даниловской. В момент времени $t = x_1/c_1$ напряжение σ_{11} получает разрыв и изменяет знак. После прохождения фронта волны напряжение приближается к квазистатическому значению. При $\alpha = 0.95$ величина скачка уменьшается, а примерно при $\alpha < 0,75$ разрыв напряжения исчезает.

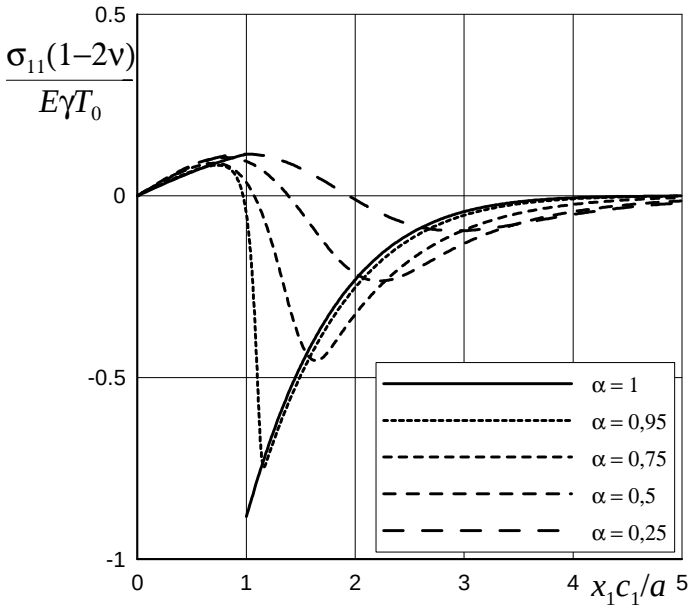

Рисунок 1.15 – Зависимость напряжения σ_{11} от пространственной координаты x_1

ПЕРЕЧЕНЬ ССЫЛОК

1. R. Metzler, J. Klafter. The random walk’s guide to anomalous diffusion: a fractional dynamics approach // Phys. Rep. – 2000. – 339, p. 1-77.
2. S. Lepri, R. Livi, A. Politi. Anomalous heat conduction, in book: Anomalous transport: foundations and applications edited by R. Klages, G. Radons, I.M. Sokolov, — Wiley, VCH(Berlin), 2008, – 584 p.
3. А.М. Нахушев. Дробное исчисление и его применение. — М.: Физматлит, 2003. — 272 с.
4. Учайкин В.В. Автомодельная аномальная диффузия и устойчивые законы. // УФН, – 2003. – т. 173, №8, – С. 847-876.
5. Смирнов Б.М. Энергетические процессы в макроскопических фрактальных структурах. // УФН. – 1991. – т. 161, №6. – С. 171-200.
6. Бейбалаев В.Д. Математическая модель теплопереноса в средах с фрактальной структурой. // Математическое моделирование. — 2009. – 21: 5. – С. 55-62.
7. Boyadjiev L., Scherer R. Fractional extensions of the temperature field problem in oil strata. // Kuwait. J. Sci. Eng. — 2004. – 31 (2). – p. 15-32.
8. Л.М. Зеленый, А.В. Милованов. Фрактальная топология и странная кинетика: от теории к проблемам космической электродинамики // УФН. — 2004. – т. 74, №8. – С. 810-853.
9. В.В. Филатов. О некоторых обратных задачах во фрактальных средах. // Международная конференция «Обратные и некорректные задачи математической физики». — 2007. – www.math.nsc.ru/conference/ipmp07.
10. Старк Д.П. Диффузия в твердых телах. — Мир, 1980. – 240с.
11. Li B., Wang J. Anomalous heat conduction and anomalous diffusion in one dimensional systems. // Phys. Rev. Lett. — 2003. – 91(4). – p. 044301.
12. Mainardi F. The fundamental solutions for the fractional diffusion-wave equation. // Appl. Math. Lett. —1996. – 9. – p. 23-28.
13. Mainardi F. Fractional relaxation-oscillation and fractional diffusion-wave phenomena. // Chaos, Solitons and Fractals. — 1996. – 7. – p. 1461-1477.

14. Schneider W.R., Wyss W. Fractional diffusion and wave equations. // J. Math. Phys. — 1989. – 30. – p. 134-144.
15. Fujita Y. Integrodifferenial equation which interpolates the heat equation and the equation. // Osaka J. Math. — 1990. – 27. – p. 309-321, 797-804.
16. Hanyga A. Multidimensional solutions of time-fractional diffusion-wave equations. // Proc. R. Soc. Lond. — 2002. – A 458. – p. 933-957.
17. Agrawal O.P. A general solution for the fourth-order fractional diffusion-wave equation. // Fract. Calculus Appl. Anal. — 2000. – 3. – p. 1-12.
18. Agrawal O.P. A general solution for the fourth-order fractional diffusion-wave equation defined in a bounded domain. // Comp. Struct. — 2001. – 79. – p. 1497-1501.
19. Beghin L. Pseudoprocesses governed by higher-order fractional differential equations. // Electronic journal of probability. — 2008. – vol. 13, n. 16. – p. 467-485.
20. Mainardi F., Paradisi P. Fractional diffusive waves. // J. Comput. Acoustics. — 2001. – 9. – p. 1417-1436.
21. Povstenko Y.Z. Fractional heat conduction equation and associated thermal stresses. // J. Thermal Stresses. — 2005. – 28. – p. 83-102.
22. Povstenko Y.Z. Stresses exerted by a source of diffusion in a case of a non-parabolic diffusion equation. // Int. J. Eng. Sci. — 2005. – 43. – p. 977-991.
23. Povstenko Y.Z. Two-dimensional axisymmetric stresses exerted by instantaneous pulses and sources of diffusion in an infinite space in a case of time-fractional diffusion equation. // Int. J. Solids Struct. — 2007. – 44. – p. 2324-2348.
24. Povstenko Y. Signalling problem for time-fractional diffusion-wave equation in a half-plane. // Fractional calculus and applied analysis. — 2008. – vol. 11, 3. – p. 329-352.
25. Mainardi F., Gorenflo R. Time-fractional derivatives in relaxation processes: a tutorial survey. // Fractional calculus and applied analysis. — 2007. – vol. 10, 3. – p. 269-308.
26. Kilbas A., Srivastava H.M., Trujillo J.J. Theory and applications of fractional differential equations. — North-Holland, Mathematics studies 204, 2006, – 524 p.

27. Вірченко Н.О., Рибак В.Я. Основи дробового інтегродиференціювання. Навч. посібник. — К., 2007. — 364 с.
28. Benchohra M., Hamani S., Ntouyas S.K. Boundary value problems for differential equations with fractional order. // Surveys in mathematics and its applications. — 2008. – vol. 3. – p. 1-12.
29. Rabha W.I. Existence results for fractional boundary value problem. // Int. J. Contemp. Math. Sciences. — 2008. – vol. 3, n. 36. – p. 1767-1774.
30. Luchko Y. Initial-boundary-value problems for the generalized time-fractional diffusion equation, to appear.
31. Yang Q., Turner I., Liu F. Analytical and numerical solutions for the time and space-symmetric fractional diffusion equation. // ANZIAM J. — 2009. – 50. – p. C800-C814.
32. Gorenflo R., Mainardi F. Fractional calculus: integral and differential equations of fractional order in A. Carpinteri, F. Mainardi (Editors): Fractals and fractional calculus in continuum mechanics, Springer Verlag, Wien and New York. — 1997. – p. 223-276.
33. M. Ciesielski, J. Leszczynski. Numerical solutions to boundary value problem for anomalous diffusion equation with Riesz-Feller fractional operator. // Journal of theoretical and applied mechanics. — 2006. – 44, 2. – p. 393-403.
34. W. Chen, S. Holm. Fractional Laplacian time-space models for linear and nonlinear lossy media exhibiting arbitrary frequency dependency. // J. Acoustic Society of America. — 2004. – 115(4). – p. 1424-1430.
35. Trefan G., Floriani E., West B.J. Grigolini P. Dynamical approach to anomalous diffusion: response of Levy processes to a perturbation. // Phys. Rev. E. — 1994. – p. 2564-2579.
36. Benson D.A., Wheatcraft S.W., Meerschaert M.M. The fractional-order equation of Levy motion, Water Resour. Res. — 2000. – 36. – p. 1413-1424.
37. Benson D.A., Schumer R., Meerschaert M.M., Wheatcraft S.W. Fractional dispersion, Levy motion, and the MADE tracer tests // Transp. Por. Med. — 2001. – 42. – p. 211-240.

38. Scalas E., Gorenflo R., Mainardi F. Uncoupled continuous-time random walks: solutions and limiting behaviour of the master equation, Phys. Rev. E. — 2004. – 692. – 011107.
39. Paradisi P., Cesari R., Mainardi F., Tampieri F. The fractional Fick's law for non-local transport processes // Physica A. — 2001. – 293 (1-2). – p. 130.
40. Povstenko Y.Z. Thermoelasticity which uses fractional heat conduction equation. // Мат. методи та фіз.-мех. поля. — 2008. – 51, №2. – С. 239-246.
41. Abdennadher A., Neel M. Fractional flux and non-normal diffusion. // Electronic Journal of differential equation. — 2007. – Conference, 16. – С. 1-13.
42. Zhang H., Liu F., Anh Vo.V. Numerical approximation of Levy-Feller diffusion equation and its probability interpretation. // Journal of Computational and Applied Mathematics. — 2006. – 206 (2). – p. 1098-1115.
43. Zhang H., Liu F. The fundamental solutions of the space, space-time Riesz fractional partial differential equations with periodic conditions. // Numer. Math. J. Chinese Univ. (English Ser.). — 2007. – issue 2, vol. 16. – p. 181-192.
44. Huang F., Liu F. The fundamental solution of the space-time fractional advection-dispersion equation. // J. Appl. Math. & Computing. — 2005. – vol. 18, No. 1-2, – p. 339-350.
45. Weitzner H., Zaslavsky G.M. Some applications of fractional equations. // Communications in nonlinear science and numerical simulation. — 2003. – vol. 8, num. 3. – p. 273-281.
46. Mainardi F. Applications of integral transforms in fractional diffusion processes. // Integral transforms and special functions. — 2004. – vol. 15, no. 6. – p. 477-484.
47. Momani Sh. General solutions for the space and time-fractional diffusion-wave equation. // Journal of physical sciences. — 2006. – vol. 10. – p. 30-43.
48. Золотухин И.В., Каинин Ю.Е., Логинова В.И. Твердотельные фрактальные структуры. // Альтернативная энергетика и экология. — 2005. – №9, 29. – С. 56-66.
49. Balankin A.S. Elastic behavior of materials with multifractal structures // Phys.Rev. B. — 1996. – vol. 53, no. 9. – p. 5438-5443.

50. Смирнов Б.М. Аэрогели. // УФН. — 1987. – т. 152, вып. 1. – С. 133-157.
51. Carpinteri A., Chiaia B., Cornetti P. A fractal theory for the mechanics of elastic materials. // Materials science and engineering A. — 2004. – 365. – C. 235-240.
52. Atanackovic T.M., Stankovic B. Generalized wave equation in nonlocal elasticity. // Acta Mechanica. — 2009. – 208. – p. 1-10.
53. Ostoja-Starzewski M. Towards thermoelasticity of fractal media. // Journal of thermal stresses. — 2007. – 30. – p. 889-896.
54. Povstenko Y.Z. Theory of thermoelasticity based on the space-time-fractional heat conduction equation. // Phys. Scr. — 2009. – T 136. – p. 014017-014023.
55. Povstenko Y.Z. Fundamental solutions to central symmetric problems for fractional heat conduction equation and associated thermal stresses. // Journal of thermal stresses. — 2008. – vol. 31, issue 2. – p. 127-148.
56. Jaemin Ahn, Sungkwon Kang, YongHoon Kwon. A flexible inverse Laplace transform algorithm and its application // Computing. – 2003. – 71, № 2. – P. 115–131.
57. Фильштинский Л. А. Периодические по времени однородные решения уравнения теплопроводности для анизотропного слоя в R^3 // Мат. методи та фіз.-мех. поля. – 2003. – 46, № 2. – С. 147–154.
58. Абрамовиц М., Стиган И. Справочник по специальным функциям с формулами, графиками и математическими таблицами. – Москва: Наука, 1979. – 832 с.
59. El-Sayed A.M., Gaber M. On the finite Caputo and finite Riesz derivatives. // EJTP. — 2006. – 3 No. 12. – p. 81-95.
60. Новацкий В. Теория упругости. — М.: Мир, 1975. – 872 с.
61. Бейтмен Г., Эрдейи А. Высшие трансцендентные функции. Эллиптические и автоморфные функции. Функции Ламе и Матье. (Серия: «Справочная математическая библитотека»). — М.: Наука, 1967. – 300 с.
62. Luchko Y. Algorithms for evaluation of the Wright function for the real arguments' values. // Fractional calculus and applied analysis. — 2008. – vol. 11, no. 1. – p. 57-75.

63. Даниловская В.И. Температурные напряжения в упругом полупространстве, возникающие вследствие внезапного нагревания границы // ПММ. — 1950. – 14. – С. 316-318.
64. Автореферат диссертации на соискание ученой степени кандидата физико-математических наук на тему: «Гибридная модель параллельного программирования DVM/OpenMP». http://www.keldysh.ru/council/1/bakhtin.htm
65. Свободная энциклопедия Википедия. http://ru.wikipedia.org/
66. Кнут Д. Э. Искусство программирования, том 3. Сортировка и поиск. Алгоритмы сортировки. Электронное издание. http://algolist.manual.ru/sort/
67. Богачев К. Ю. Основы параллельного программирования. М.: БИНОМ. Лаборатория знаний, 2003.
68. Корнеев В. Д. Параллельное программирование в MPI. Москва-Ижевск: Институт компьютерных исследований, 2003.
69. Документация по библиотеке MPI. http://www.mpi-forum.org/docs/docs.html
70. Grama A., Gupta A., Karypis G., Kumar V., Introduction to parallel computing. Addison-Wesley, 2003
71. В. П. Гергель. Теория и практика параллельных вычислений. Электронное издание.
72. MPI Forum. http://www.mpi-forum.org/
73. Параллельное программирование в интерфейсе MPI. Виртуальный курс. http://www.ccas.ru/mmes/educat/lab04k/01/basics.html
74. Паралельні обчислювальні системи. http://www.refine.org.ua/pageid-5072-5.html
75. OpenMP Architecture Review Board. OpenMP Application Program Interface. Version 3.0 May 2008. http://openmp.org
76. Ian Foster. Designing and Building Parallel Programs. - Argonne National Laboratory http://www.mcs.anl.gov/~itf/dbpp/text/node1.html
77. Blaise Barney. Introduction to Parallel Computing. - Lawrence Livermore National Laboratory https://computing.llnl.gov/tutorials/parallel_comp/
78. Blaise Barney. OpenMP. - Lawrence Livermore National Laboratory https://computing.llnl.gov/tutorials/openMP/*

2 ВВЕДЕНИЕ В ПРОБЛЕМАТИКУ РАСЧЕТОВ ФРАКТАЛЬНОЙ ТЕПЛОПРОВОДНОСТИ И ТЕРМОУПРУГОСТИ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

2.1 Актуальность параллельных вычислений

Современные задачи науки и техники, такие как моделирование климата, генная инженерия, проектирование интегральных схем, анализ загрязнения окружающей среды, создание лекарственных препаратов и др. требуют для своего решения ЭВМ с огромной вычислительной производительностью. Единичные рабочие станции уже не способны в короткие сроки решать вышеперечисленные задачи. Вот здесь и возникает необходимость использования параллельных вычислений, когда над общей задачей работают десятки, сотни и даже тысячи рабочих станций.

Следует отметить, что до сих пор применение параллельных вычислений не получило столь широкого распространения, как это ожидалось многими исследователями. Одной из возможных причин подобной ситуации являлась до недавнего времени высокая стоимость высокопроизводительных систем (приобрести супер-ЭВМ могли себе позволить только крупные компании и организации). Современная тенденция построения параллельных вычислительных комплексов из типовых конструктивных элементов (микропроцессоров, микросхем памяти, коммуникационных устройств), массовый выпуск которых освоен промышленностью, снизила влияние этого фактора, и в настоящий момент практически каждый потребитель может иметь в своем распоряжении многопроцессорные вычислительные системы достаточно высокой производительности. Наиболее кардинально ситуация изменилась в сторону параллельных вычислений с появлением многоядерных процессоров, которые уже в 2006 г. использовались более чем в 70% компьютерных систем. [72]

В течение нескольких десятилетий развитие ЭВМ сопровождалось удвоением их быстродействия каждые 1.5-2 года. Это обеспечивалось и повышением тактовой частоты и совершенствованием архитектуры (параллельное и конвейерное выполнение команд). Однако наступил период, когда дальнейшее повышение тактовой

частоты процессора стало практически невозможно из-за резкого повышения потребления и выделения энергии. Использовать эффективно постоянно возрастающее количество элементов микросхемы не удается из-за того, что практически достигнут предел логической сложности процессоров, а дальнейшее увеличение размеров кэш-памяти уже не приводит к ускорению выполнения программ. При достигнутом уровне тактовой частоты сигналы уже не успевают за один такт пройти требуемый путь внутри кристалла процессора. [64]

Несколько лет назад разработчики процессоров пришли к выводу, что дальнейшее повышение быстродействия надо обеспечивать путем реализации в кристалле многих параллельно работающих процессоров – ядер. Такие кристаллы стали называться многоядерными процессорами. Необходимо отметить, что ядра могут совместно использовать некоторые ресурсы, такие как кэш-память, буфера быстрой переадресации (TLB) и каналы доступа к оперативной памяти.

Появление и широкое распространение многоядерных процессоров приводит к революционным изменениям для программистов – очень скоро понятие «параллельное программирование» потеряет свой смысл, поскольку все программы будут параллельными. [64]

Другая и, пожалуй, теперь основная причина сдерживания массового распространения параллелизма состоит в том, что для проведения параллельных вычислений необходимо «параллельное» обобщение традиционной – последовательной – технологии решения задач на ЭВМ. Так, численные методы в случае многопроцессорных систем должны проектироваться как системы параллельных и взаимодействующих между собой процессов, допускающих исполнение на независимых процессорах. Применяемые алгоритмические языки и системное программное обеспечение должны обеспечивать создание параллельных программ, организовывать синхронизацию и взаимоисключение асинхронных процессов и т. п. [72]

Для оценки производительности современных ЭВМ был введен термин флопс (от FLOPS, акроним от англ. Floating point Operations Per Second) – величина, показывающая, сколько операций с плавающей запятой (точкой) в секунду выполняет данная вычислительная система. [65] Поскольку современные компьютеры облада-

ют высоким уровнем производительности, более распространены производные величины от FLOPS, образуемые путём использования стандартных приставок системы СИ. Соотношение между ними показано в таблице 2.1.

Таблица 2.1

флопс	10^0	Flops
килофлопс	10^3	KFlops
мегафлопс	10^6	MFlops
гигафлопс	10^9	GFlops
терафлопс	10^{12}	TFlops
петафлопс	10^{15}	PFlops

Однако под определением «операций с плавающей запятой» может скрываться масса разных понятий. Кроме того, величина флопс подвержена влиянию очень многих факторов, напрямую не связанных с производительностью вычислительного модуля, таких как: пропускная способность каналов связи с окружением процессора, производительность основной памяти и синхронность работы кэш-памяти разных уровней. [65]

Всё это, в конечном итоге, приводит к тому, что результаты, полученные на одном и том же компьютере при помощи разных программ, могут существенным образом отличаться, более того, с каждым новым испытанием разные результаты можно получить при использовании одного алгоритма. Эта проблема решается соглашением об использовании единообразных тестовых программ (таких как тест LINPACK) с усреднением результатов.

Современные многоядерные рабочие станции способны достигать производительности 60-80 GFlops. Однако проблемы современной науки и техники, такие как моделирование климата, генная инженерия, поиск внеземных цивилизаций и др. – требуют для своего анализа ЭВМ с производительностью в несколько тысяч миллиардов операций с плавающей запятой в секунду (несколько TFlops). Тогда на помощь ученым приходят кластеры.

Кластер – группа компьютеров, объединённых высокоскоростными каналами связи, представляющая с точки зрения пользователя единую машину. [65]

Один из первых архитекторов кластерной технологии Грегори Пфистер (Gregory F. Pfister) дал кластеру следующее определение: «Кластер – это разновидность параллельной или распределённой системы, которая:

1) состоит из нескольких связанных между собой компьютеров;
2) используется как единый, унифицированный компьютерный ресурс». [65]

Обычно различают следующие основные виды кластеров:

1) отказоустойчивые кластеры (High-availability clusters, HA);
2) кластеры с балансировкой нагрузки (Load balancing clusters);
3) вычислительные кластеры (Computing clusters);
4) grid-системы.

Computing clusters используются в вычислительных целях, в частности в научных исследованиях. Для вычислительных кластеров существенными показателями являются высокая производительность процессора (FLops) и низкая латентность объединяющей сети. Вычислительные кластеры позволяют уменьшить время расчетов, по сравнению с одиночным компьютером, разбивая задание на параллельно выполняющиеся ветки, которые обмениваются данными по связывающей сети. Одна из типичных конфигураций – набор компьютеров, собранных из общедоступных компонентов, с установленной на них операционной системой Linux, и связанных сетью Ethernet, InfiniBand или другими относительно недорогими сетями. Специально выделяют высокопроизводительные кластеры (обозначаются англ. аббревиатурой HPC Cluster – High-performance computing cluster). Следует отметить, что в качестве операционной системы может использоваться Microsoft Windows HPC Server 2008, 30 лицензий которой было закуплено нашим университетом для организации параллельных вычислений в вузе. Список самых мощных высокопроизводительных компьютеров (также может обозначаться англ. аббревиатурой HPC) попадает в мировой рейтинг TOP500.

Следует отметить, что существующие современные реализации стандартов для процесса параллельных вычислений позволяют организовать кластер на имею-

щемся оборудовании: так компьютерный класс в 12-15 ПК с современными многоядерными процессорами можно задействовать для решения задач параллельных вычислений.

Кластерная система относится к вычислительным системам с разделенной памятью, в отличие от систем с общей памятью (где «параллельные» задачи можно решать, например, с помощью модели OpenMP). Для реализации параллельных вычислений в такой системе хорошо подходит стандарт MPI.

2.2 Использование стандарта MPI для организации параллельных вычислений

MPI (Message Passing Interface, интерфейс передачи сообщений) – программный интерфейс для передачи информации, который позволяет обмениваться сообщениями между компьютерами, выполняющими одну задачу. [65] Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. MPI является на данный момент стандартом по умолчанию для систем с распределенной памятью и самой развитой переносимой библиотекой параллельного программирования с передачей сообщений. MPI не является формальным стандартом, подобным тем, что выпускают организации стандартизации, такие как ANSI или ISO. Стандартизацией MPI занимается MPI Forum [73]. «Стандарт» MPI был введен MPI-форумом в мае 1994 и обновлен в июне 1995. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков программирования Фортран 77/90, Си и Си++.

Большинство современных реализаций MPI поддерживают версию 1.1. Стандарт MPI версии 2.0 не изменяет предыдущий и поддерживается большинством современных реализаций, однако некоторые функции могут быть реализованы не до конца.

В MPI 1.1 (опубликован 12 июня 1995 года) поддерживаются следующие функции:

1) передача и получение сообщений между отдельными процессами;

2) коллективные взаимодействия процессов;
3) взаимодействия в группах процессов;
4) реализация топологий процессов. [65]

В MPI 2.0 (опубликован 18 июля 1997 года) дополнительно поддерживаются следующие функции:

1) управление динамическими процессами;
2) ассиметричные операции;
3) параллельный ввод/вывод (I/O);
4) привязка к Си++ и ФОРТРАН 90;
5) внешние интерфейсы;
6) расширенные коллективные коммуникации;
7) расширения реального времени и др. [74]

В настоящее время наиболее распространенными реализациями MPI являются:

1) MPICH – самая распространённая бесплатная реализация от Argonne National Lab и Mississippi State University, работающая на UNIX и Windows системах;
2) WMPI – бесплатная реализация MPI для Windows;
3) LAM/MPI – ещё одна бесплатная реализация MPI от Ohio Supercomputer Center;
4) Intel MPI – коммерческая реализация для Windows / GNU/Linux;
5) MS MPI – коммерческая реализация фирмы Microsoft, используется в Windows HPC Server;
6) Open MPI – бесплатная реализация MPI, наследник LAM/MPI.

Следует учитывать, что в практике термин MPI носит «двойственное» значение. С одной стороны под ним подразумевают стандарт, которому должны удовлетворять средства организации передачи сообщений, с другой – это программные средства, которые обеспечивают возможность передачи сообщений и при этом соответствуют всем требованиям стандарта MPI (библиотеки MPI). Правильное значение выбирается из контекста.

Среди основных достоинств использования MPI следует выделить:

1) программы разрабатываются на широко распространенных языках программирования Си и Фортран;
2) разработанная с использованием библиотеки MPI программа, как правило, будет работать на разных вычислительных платформах;
3) практически для каждого типа вычислительных систем существуют реализации библиотек MPI, в максимальной степени учитывающие возможности компьютерного оборудования;

MPI облегчает написание программ, поскольку много основных операций передачи данных предусматривается стандартом MPI, а с другой стороны, уже имеется большое количество библиотек параллельных методов, созданных с использованием MPI.

2.3 Применение реализации MPICH2 стандарта MPI для решения задачи сортировки данных

Пожалуй, никакая другая проблема не породила такого количества разнообразнейших решений, как задача сортировки. Существует ли некий «универсальный», наилучший алгоритм? Строго говоря, нет. Однако, имея приблизительные характеристики входных данных, можно подобрать метод, работающий оптимальным образом.

Сортировка является одной из типовых проблем обработки данных и обычно понимается как задача размещения элементов неупорядоченного набора значений $a_1, a_2, a_3, \ldots a_n$ в порядке монотонного возрастания или убывания $a_1', a_2', a_3', \ldots a_n'$, где $a_1' \le a_2' \le a_3' \le \ldots a_n'$ (в случае возрастания).

Возможные способы решения этой задачи широко обсуждаются в литературе; один из наиболее полных обзоров алгоритмов сортировки содержится в работе [66].

Вычислительная трудоемкость процедуры упорядочивания является достаточно высокой. Так, для ряда известных простых методов (пузырьковая сортировка, сортировка вставками и др.) количество необходимых операций определяется квадратичной зависимостью от числа упорядочиваемых данных $T \sim n^2$.

Для более эффективных алгоритмов (сортировка слиянием, сортировка Шелла, быстрая сортировка) трудоемкость определяется величиной $\log_2 n$. Ускорение сортировки может быть получено при использовании нескольких процессоров (ядер). Исходный набор данных в этом случае разделяется между процессорами; в ходе сортировки данные пересылаются между процессорами и сравниваются между собой. Результирующий (упорядоченный) набор, как правило, также разделен между процессорами; при этом для систематизации такого разделения для процессоров вводится та или иная система последовательной нумерации и обычно требуется, чтобы при завершении сортировки значения, располагаемые на процессорах с меньшими номерами (рангами), не превышали значений процессоров с большими номерами (случай возрастания). В конечном итоге, результат получается сборкой всех частей отсортированных данных на одном узле вычислительной системы.

При внимательном рассмотрении способов упорядочивания данных, применяемых в алгоритмах сортировки, можно заметить, что многие методы основаны на применении одной и той же базовой операции «сравнить и переставить» (compare-exchange), состоящей в сравнении той или иной пары значений из сортируемого набора данных и перестановке этих значений, если их порядок не соответствует условиям сортировки.

Целенаправленное применение данной операции позволяет упорядочить данные; в способах выбора пар значений для сравнения. В этом и проявляется различие алгоритмов сортировки.

Последовательный алгоритм пузырьковой сортировки (the bubble sort) сравнивает и обменивает пары соседних элементов в последовательности, которую нужно отсортировать. Для последовательности $(a_1, a_2, a_3, \ldots, a_n)$ алгоритм сначала выполняет $n-1$ базовых операций «сравнения-обмена» для последовательных пар элементов $(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)$.

В результате после первой итерации алгоритма самый больший элемент перемещается («всплывает») в конец последовательности (в случае сортировки по возрастанию). Далее последний элемент в преобразованной последовательности может

быть исключен из рассмотрения, и описанная выше процедура применяется к оставшейся части последовательности $a_1', a_2', a_3', ..., a_{n-1}'$.

Как можно увидеть, последовательность будет отсортирована после $n-1$ итерации. Эффективность пузырьковой сортировки может быть улучшена несколькими способами:

1) в случае отсутствия каких-либо изменений сортируемой последовательности данных в ходе какой-либо итерации сортировки выполнение алгоритма можно прекратить – последовательность уже и так отсортирована;
2) если запоминать индекс последнего обмена и сортировать до этого индекса, то количество сравнений в итерации можно уменьшить; [67]
3) если использовать так называемый алгоритм «шейкер-сортировки», когда на первой итерации мы сортируем в одном направлении, а на второй – в обратном и т. д. Это улучшение полезно в случае, когда массив изначально почти отсортирован.

Последовательная реализация метода пузырьковой сортировки может, к примеру, иметь следующий вид:

```
void bubblesort(int *mass, int N)
{
    int temp, i, j;
    for( i = 1; i < N - 1; i++ )
        for( j = 0; j < N - i; j++ )
            if( mass[j] > =mass[j + 1] )
            {
                temp = mass[j + 1];
                mass[j + 1] = mass[j];
                mass[j] = temp;
            }
```

Здесь $*mass$ – сортируемые данные (числа типа int), N – их количество.

Алгоритм пузырьковой сортировки достаточно сложен для распараллеливания – сравнение пар значений упорядочиваемого набора данных происходит строго последовательно. В связи с этим для параллельного применения используется не сам этот алгоритм, а его модификация, известная в литературе как метод чет-нечетной перестановки (the odd-even transposition method) – см., например, [71]. Суть модификации состоит в том, что в алгоритм сортировки вводятся два разных правила выполнения итераций метода: в зависимости от четности или нечетности номера итерации сортировки для обработки выбираются элементы с четными или нечетными

индексами соответственно, сравнение выделяемых значений всегда осуществляется с их правыми соседними элементами. Таким образом, на всех нечетных итерациях сравниваются пары $(a_1, a_2), (a_3, a_4), ..., (a_{n-1}, a_n)$ (при четном n), а на четных итерациях обрабатываются элементы $(a_2, a_3), (a_4, a_5), ..., (a_{n-2}, a_{n-1})$.

После n-кратного повторения итераций сортировки исходный набор данных оказывается упорядоченным. [72]

Получение параллельного варианта для метода чет-нечетной перестановки уже не представляет каких-либо затруднений – сравнения пар значений на итерациях сортировки являются независимыми и могут быть выполнены параллельно. В случае, когда количество процессоров (ядер процессора) меньше числа упорядочиваемых значений, процессоры содержат блоки данных размера n/p, где n – количество элементов в наборе данных, p – количество процессоров (ядер процессора).

Для пояснения работы параллельного алгоритма чет-нечетной перестановки можно привести пример:

Таблица 2.2

№ и тип итерации		Процессоры			
		1	2	3	4
Исходные данные		16 52 54 76	13 73 43 55	5 19 29 75	8 17 72 44
1 нечет (1, 2), (3, 4)	до	16 52 54 76	13 73 43 55	5 19 29 75	8 17 72 44
	после	13 16 43 52	54 55 73 76	5 8 17 19	29 44 72 75
2 чет (2, 3)	до	13 16 43 52	54 55 73 76	5 8 17 19	29 44 72 75
	после	13 16 43 52	5 8 17 19	54 55 73 76	29 44 72 75
3 нечет (1, 2), (3, 4)	до	13 16 43 52	5 8 17 19	54 55 73 76	29 44 72 75
	после	5 8 13 16	17 19 43 52	29 44 54 55	72 73 75 76
4 чет (2, 3)	до	5 8 13 16	17 19 43 52	29 44 54 55	72 73 75 76
	после	5 8 13 16	17 19 29 43	44 52 54 55	72 73 75 76

В приведенном примере, количество исходных данных $n = 16$, задействовано 4 процессора ($p = 4$).

Алгоритм пузырьковой сортировки работает довольно медленно в связи с большим количеством операций «сравнить и переставить» поэтому его можно эффективно использовать для загрузки современных процессоров.

В нашем примере алгоритм, описанный в таблице 2.2, был несколько доработан: исходные данные разбивались между доступными процессорами не по схеме n/p, а по схеме $n/2p$. Таким образом, количество итераций возросло в два раза. Это было сделано по двум причинам:

1) Как показывает практика, зависимость между числом сортируемых данных и временем имеет экспоненциальный характер (см. рис. 2.1). Поэтому логично предположить, что сортировка и передача меньших порций каналу Ethernet даст больший прирост производительности.
2) Как следует из табл. 2.2, в момент сравнения данных между процессорами, например, (1, 2) и (3, 4) в первой итерации вычислительная нагрузка накладывается только на один процессор из пары (1 или 2 для первой пары, 3 или 4 для второй). Разбив же данные дополнительно по предложенному методу, вычислениями будут загружены практически все процессоры.

Исходя из всего вышеперечисленного, оптимизированный пример сортировки данных параллельным методом чет-нечетной перестановки представим в табл. 2.3.

Таблица 2.3

№ и тип итерации	Процессоры							
	1		2		3		4	
Исходные данные	16 52	54 76	13 73	43 55	5 19	29 75	8 17	72 44
(1а, 1б), (2а, 2б), (3а, 3б), (4а, 4б)	16 52	54 76	13 43	55 73	5 19	29 75	8 17	44 72
(1б, 2а), (2б, 3а), (3б, 4а)	16 52	13 43	54 76	5 19	55 73	8 17	29 75	44 72
(1а, 1б), (2а, 2б), (3а, 3б), (4а, 4б)	13 16	43 52	5 19	54 76	8 17	55 73	29 44	72 75
(1б, 2а), (2б, 3а), (3б, 4а)	13 16	5 19	43 52	8 17	54 76	29 44	55 73	72 75
(1а, 1б), (2а, 2б), (3а, 3б), (4а, 4б)	5 13	16 19	8 17	43 52	29 44	54 76	55 72	73 75
(1б, 2а), (2б, 3а), (3б, 4а)	5 13	8 16	17 19	29 43	44 52	54 55	72 76	73 75
(1а, 1б), (2а, 2б), (3а, 3б), (4а, 4б)	5 8	13 16	17 19	29 43	44 52	54 55	72 73	75 76
(1б, 2а), (2б, 3а), (3б, 4а)	5 8	13 16	17 19	29 43	44 52	54 55	72 73	75 76

Из табл. 2.3 видно, что в задаче в конкретный момент времени задействованы либо все процессоры либо $p-1$.

Следует отметить, что в качестве исходных данных могут выступать, например, как числа типа int, *float*, *double* так и строковые величины. Будем рассматривать задачу, в которой алгоритм работает с натуральными числами. В ходе программной реализации были написаны следующие программы:

1) программа для генерации файла с исходными данными (натуральные числа), в которую в качестве параметров командой строки передаются количество элементов и имя файла. Следует отметить, что генерация данных производится действительно случайным способом, т. е. сгенерированные данной программой несколько файлов, например, по 10000 строк будут содержать различные значения элементов. Первая строчка файла содержит количество элементов, затем перечисляются все элементы по одному в каждой строке;
2) программа, реализующая последовательный алгоритм пузырьковой сортировки;
3) программа, выполняющая пузырьковую сортировку с использованием метода чет-нечетной перестановки;
4) параллельная программа, выполняющая пузырьковую сортировку с использованием метода чет-нечетной перестановки.

Программы были разработаны в среде Microsoft Visual Studio 2008, на языке программирования Си, с использованием реализации стандарта MPI 2 в бесплатной среде MPICH2.

Запуск тестовых программ проводился в компьютерном классе Г-807 на компьютерах класса Celeron 2 GHz, канал связи – сеть Ethernet 100 Mbit/sec. Результаты вычислений приведены на рис. 2.1.

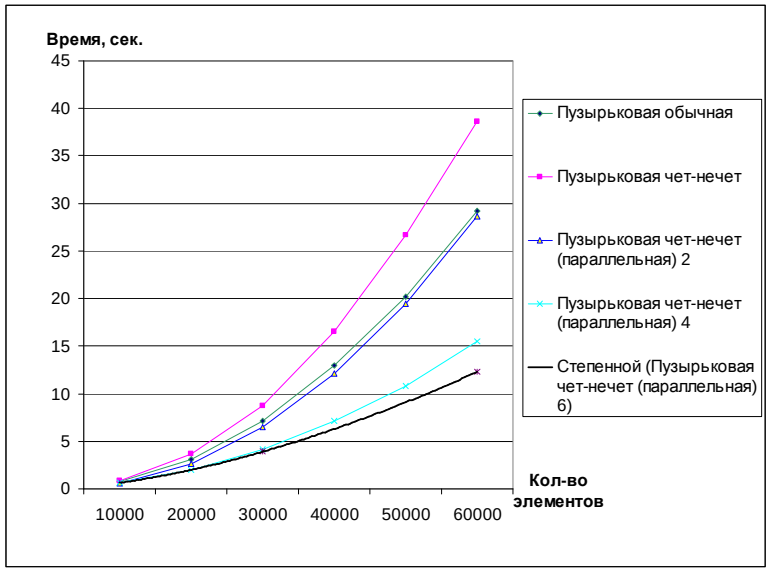

Рисунок 2.1 – Результаты вычислительных экспериментов для параллельного алгоритма пузырьковой сортировки.

Как и следует из теории, последовательный алгоритм чет-нечетной сортировки метода пузырька показывает наихудший результат, потому что использует наибольшее количество итераций. В сравнении с последовательным методом пузырьковой сортировки его параллельная реализация показывает полное преимущество от использования нескольких процессоров. Результаты вычислений можно видеть в табл. 2.4, где время вычислений указано в секундах.

Таблица 2.4.

Кол-во эл.	Пузырьковая сортировка		Параллельная пузырьковая сортировка (процессоры)		
	обычная	чет-нечет	2	4	6
10000	0,73	0,84	0,6	0,7	–
20000	3,1	3,7	2,6	2	–
30000	7,1	8,7	6,5	4,1	3,9
40000	13	16,5	12,1	7,1	–
50000	20,2	26,7	19,4	10,8	–
60000	29,2	38,6	28,7	15,5	12,3

Анализируя результаты таблицы 2.4, делаем вывод, что с увеличением числа доступных для задачи процессоров время выполнение параллельного алгоритма существенно уменьшается при увеличении количества сортируемых элементов. Так,

при $n = 30000$ время выполнения алгоритма в 1,73 раза на четырех и в 1,82 раза на шести процессорах меньше чем у последовательной реализации. При увеличении n до 60000 время выполнения в 1,88 раза и 2,37 раза соответственно меньше от исходного. Т. е. ускорение в вычислениях достигает 2,4 раза.

Следует отметить, что в [72] автором решалась аналогичная задача, однако вместо прироста производительности при выполнении параллельного алгоритма пузырьковой сортировки он получил замедление процесса выполнения задачи. Может быть, программная реализация использовала интенсивный обмен данными между процессорами по каналу связи, что свело на нет все преимущества параллельных вычислений.

После выполнения параллельных вычислений на кластере возникла идея проверить работу параллельного алгоритма на компьютере с четырехядерным процессором Core 2 Quad 8200. Результаты вычислений показаны на рис. 2.2.

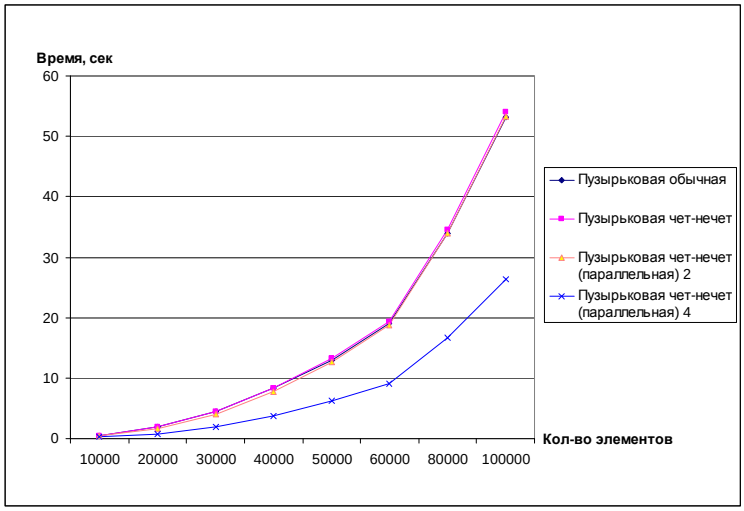

Рисунок 2.2 – Результаты вычислительных экспериментов для параллельного алгоритма пузырьковой сортировки (локальный кластер).

Как видно по рис. 2.2 существенный прирост наблюдается лишь при использовании 4-х ядер процессора. При вычислении на двух ядрах прирост незначительный (3-6%).

Сортировка Шелла (the Shell sort) является довольно интересной модификацией алгоритма сортировки простыми вставками. Общая идея сортировки состоит в сравнении на начальных стадиях сортировки пар значений, располагаемых доста-

точно далеко друг от друга в упорядочиваемом наборе данных. [72] Существует несколько вариантов этой сортировки.

Общую схему можно представить следующим образом: на первом шаге весь массив данных делится на $n/2$ пар и упорядочиваются элементы, стоящие на i и $n/2+i$ местах $(i=1,2,...,n/2)$. На втором шаге упорядочиваем $n/4$ пар элементов, стоящих на местах i, $n/4+i$, $n/2+i$ и $3n/4+i$ местах. На третьем шаге работаем с $n/8$ парами элементов и т. д. На последнем шаге упорядочиваются все элементы данных. Для упорядочивания элементов в группах применяется метод сортировки вставками.

Такая модификация метода сортировки позволяет быстро переставлять далекие неупорядоченные пары значений (сортировка таких пар обычно требует большого количества перестановок, если используется сравнение только соседних элементов).

Можно заметить, что общее количество итераций алгоритма Шелла равно $\log_2 n$.

Использованный в примере набор 1, 2, 4, 8, … пар – неплохой выбор, особенно, когда количество элементов – степень двойки. Однако гораздо лучший вариант предложил Р. Седжвик. Его последовательность имеет вид:

$$inc[s] = \begin{cases} 9*2^s - 9*2^{s/2} + 1, \text{если s четно} \\ 8*2^s - 6*2^{(s+1)/2} + 1, \text{если s нечет.} \end{cases}$$

При использовании таких приращений среднее количество итераций равно $n^{7/6}$, в худшем случае – порядка $n^{4/3}$ [67].

На языке программирования Си последовательный алгоритм сортировки Шелла можно представить следующим образом:

```
void shell( int *mass, int N ) //используем четное число элементов.
{
    int temp, p, k, i, j;

    for( p = 2;(k = N / p) > 1; p *= 2)
        for (i=1;i<p;i++)
            if ( mass[k * i] < mass[k * (i - 1)] )
            {
                j = k * i;
                do
                {
                    temp = mass[j];
                    mass[j] = mass[j - k];
                    mass[j - k] = temp;
                    j -= k;
```

```
                }while( j >= k && mass[j] < mass[j - k]);
            }

    for( i = 1; i < N; i++ ) //последний проход для всего массива данных.
        if( mass[i] < mass[i - 1] )
        {
            j = i;
            do
            {
                temp = mass[j];
                mass[j] = mass[j - 1];
                mass[j - 1] = temp;
                j--;
            }while( j > 0 && mass[j] < mass[j - 1] );
        }
}
```

В работе [72] предложен вариант реализации для частного случая, когда топология коммуникационной сети представлена в виде N-мерного гиперкуба. Однако, нашем случае можно воспользоваться подходом, изложенным выше для алгоритма пузырьковой сортировки, т. е. применить метод чет-нечетной перестановки.

В процессе проведения численного эксперимента были написаны следующие программы:

1) программа, реализующая последовательный алгоритм сортировки Шелла (для натуральных чисел);
2) программа, выполняющая реализацию сортировки Шелла с использованием метода чет-нечетной перестановки для натуральных чисел;
3) «параллельная» программа, выполняющая сортировку Шелла с использованием метода чет-нечетной перестановки (с натуральными числами);
4) программа для генерации файла исходных данных строкового типа, в которой в качестве параметров командой строки передаются количество элементов и имя файла. В первой строчке файла находится количество элементов, затем перечисляются все элементы по одному в каждой строке;
5) параллельная программа, выполняющая сортировку Шелла с использованием метода чет-нечетной перестановки (со строковыми данными).

На рис. 2.3 показана эффективность алгоритма при использовании параллельных вычислений.

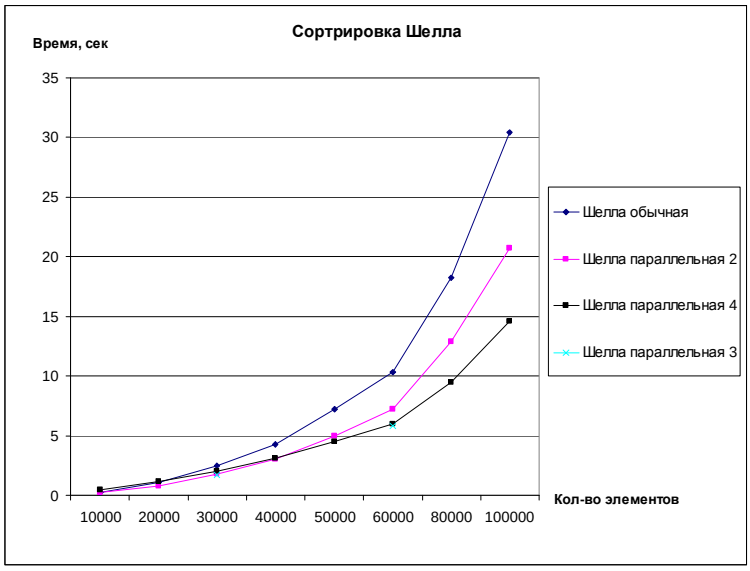

Рисунок 2.3 – Результаты работы последовательного и параллельного алгоритмов сортировки Шелла.

Выполнение сортировки осуществлялось над данными из исходного примера пузырьковой сортировки (использовались элементы, представляющие натуральные числа). Как видно по рис. 2.3, с увеличением узлов кластера время сортировки параллельного алгоритма существенно уменьшается при увеличении количества сортируемых элементов. Так, при $n = 60000$ элементов время выполнения алгоритма в 1,43 раза на двух и в 1,72 раза на четырех процессорах меньше чем у последовательной реализации. Прирост от использования параллельной реализации алгоритма оказывается меньшим, чем в методе пузырька, где ускорение достигало 2,4 раза. Однако быстрота самого алгоритма сортировки Шелла и прирост в 1,72 раза по сравнению с последовательной реализацией поднимают решение подобных задач на новый уровень.

В работе [72] автор проводил реализацию сортировки Шелла, однако по его методу прироста производительности не наблюдается, а, наоборот, при выполнении реализации алгоритма он получил замедление процесса выполнения задачи. Однако следует отметить, что между нашими как программной, так и аппаратной составляющими экспериментов имеются существенные отличия. Так, автор [72] использовал процессоры класса Intel Xeon 4 EM64T, 3000 МГц и операционную систему (ОС) Microsoft Windows Server 2003 Standard x64 Edition и систему управления кла-

стером Microsoft Compute Cluster Server. Наши вычисления проводились в ОС Microsoft Windows XP SP3 x86 с использованием MPICH2.

2.4 Общие сведенья об OpenMP

Как следует из классификации Флинна, существует два подхода к распараллеливанию вычислений: параллелизм данных и параллелизм задач. В основе каждого подхода лежит распределение вычислительной работы между Доступными пользователю процессорами параллельного компьютера. При этом приходится решать разнообразные проблемы. Прежде всего, это равномерная загрузка процессоров, т. к. если основная вычислительная работа будет ложиться только на часть из них, уменьшится и выигрыш от распараллеливания. Другая не менее важная проблема - эффективная организация обмена информацией между задачами. Если вычисления выполняются на высокопроизводительных процессорах, загрузка которых достаточно равномерна, но скорость обмена данными при этом низка, большая часть времени будет тратиться впустую на ожидание информации, необходимой для дальнейшей работы задачи. Это может существенно снизить скорость работы программы.

Основная идея подхода, основанного на параллелизме данных - применение одной операции сразу к нескольким элементам массива данных. Различные фрагменты такого массива обрабатываются на векторном процессоре или на разных процессорах параллельной машины. Векторизация или распараллеливание в этом случае чаще всего выполняются уже во время трансляции - перевода исходного текста программы в машинные команды. Роль программиста в этом случае обычно сводится к заданию опций векторной или параллельной оптимизации транслятору, директив параллельной компиляции, использованию специализированных языков параллельных вычислений.

Основные особенности рассматриваемого подхода:

1) обработкой данных управляет одна программа;

2) пространство имен является глобальным, т. е. для программиста существует одна единственная память, а детали структуры данных, доступа к памяти и межпроцессорного обмена данными от него скрыты;
3) слабая синхронизация вычислений на параллельных процессорах, т. е. выполнение команд на разных процессорах происходит, как правило, независимо и только иногда производится согласование выполнения циклов или других программных конструкций - их синхронизация. Каждый процессор выполняет один и тот же фрагмент программы, но нет гарантии, что в заданный момент времени на всех процессорах выполняется одна и та же машинная команда;
4) параллельные операции над элементами массива выполняются одновременно на всех доступных данной программе процессорах.

Таким образом, в рамках данного подхода от программиста не требуется больших усилий по векторизации или распараллеливанию вычислений. Даже при программировании сложных вычислительных алгоритмов можно использовать библиотеки подпрограмм, специально разработанных с учетом конкретной архитектуры компьютера и оптимизированных для этой архитектуры.

Реализация модели параллелизма данных требует поддержки параллелизма на уровне транслятора. Такую поддержку могут обеспечивать:

1) препроцессоры, использующие существующие последовательные трансляторы и специализированные библиотеки, с реализациями параллельных алгоритмических конструкций;
2) предтрансляторы, которые выполняют предварительный анализ логической структуры программы, проверку зависимостей и ограниченную параллельную оптимизацию;
3) распараллеливающие трансляторы, которые выявляют параллелизм в исходном коде программы и выполняют его преобразование в параллельные конструкции. Для того чтобы упростить преобразование, в исходный текст программы могут добавляться специальные директивы трансляции.

OpenMP - это набор спецификаций по параллелизации программ в среде с общей памятью. OpenMP предоставляет набор прагм, процедур и переменных среды,

которые программисты могут использовать для определения параллелизма в системах с общей памятью в программах на Фортране, C и C++.

Когда прагмы OpenMP используются в программе, они дают указание компилятору, поддерживающему OpenMP, создать исполнимый модуль, который будет выполняться параллельно с использованием нескольких потоков. При этом в исходный код необходимо внести небольшие изменения (отличные от доводки для достижения максимальной производительности). Прагмы OpenMP позволяют использовать красивый, единообразный и переносимый интерфейс для параллелизации программ на различных архитектурах и системах. Спецификация OpenMP принята многими и поддерживается такими поставщиками, как Sun, Intel, IBM и SGI. (Ссылка на веб-сайт OpenMP, где имеется документ с последней версией спецификации OpenMP, находится ниже в разделе литературы.)

Модель OpenMP позволяет параллельному программированию подняться на следующий уровень, создавая для программиста потоки и управляя ими. Все, что необходимо, это вставить соответствующие прагмы в исходный код программы и затем скомпилировать программу компилятором, поддерживающим OpenMP, с соответствующим ключом. Компилятор интерпретирует прагмы и параллелизует код. При использовании компиляторов, не поддерживающих OpenMP, прагмы OpenMP игнорируются без дополнительных сообщений.

2.5 Реализация решения задачи вычисления определенного интеграла с использованием OpenMP

Для оценки выгод использования возможностей интерфейса OpenMP рассмотрим небольшой пример распараллеливания некой вычислительной процедуры. Покажем на ее примере некоторые моменты, которые удобно учитывать при написании программы для дальнейшего распараллеливания. Приведенный ниже код иллюстрирует вычисление определенного интеграла на интервале $[a;b]$ от некоторой функции одной переменной методом средних прямоугольников:

```
#include <stdio.h>
#include <math.h>

double f( double x );

int main()
{
    int intl_count; // количество интервалов
    int iter_count; //количество выполненных итераций
    int i; // вспомогательная переменная
    double a; // начальная точка интервала
    double b; // конечная точка интервала
    double h; // шаг интервала
    double eps; // точность приближения
    double sum; // значение суммы
    double pre_sum; // предыдущее значение суммы

    intl_count = 10;
    a = 0.0;
    b = 1.0;
    eps = 0.1e-6;
    sum = 0.0;
    iter_count = 0;

    do
    {
        pre_sum = sum;
        h = abs(a - b) / (double)intl_count;
        sum = 0.0;

        for ( i = 0; i < intl_count; i++ )
        {
            sum += f(a + h * ((double)i + 0.5)) * h;
        }

        intl_count++;
        iter_count++;

    } while ( abs(sum - pre_sum) > eps );

    printf("Calculated value is %.16f, %d iterations\n", sum, iter_count);

    return 0;
}

double f( double x )
{
    return x * sin(x);
}
```

Программа задает некоторое начальное количество интервалов разбиения области интегрирования и итеративно вычисляет приближенные значения интеграла, увеличивая количество интервалов разбиения после каждой итерации. Выполнение завершается, когда разница между вычисленными приближенными значениями интеграла на двух последних итерациях становится меньше заданной точности. Для осуществления такой проверки должно быть выполнено минимум две итерации. На каждой итерации осуществляется вычисление шага текущего разбиения и цикл суммирования площадей прямоугольников. В теле цикла осуществляется вычисление

середины интервала и значение интегрируемой функции в ней, после чего вычисленное значение площади прямоугольника добавляется к текущей сумме.

В приведенном примере верхняя граница внешнего цикла не известна заранее, поэтому, несмотря на отсутствие зависимостей между итерациями, его нельзя распараллелить автоматически. Распараллеливанию подлежит внутренний цикл, в котором осуществляется суммирование. При объявлении этого цикла параллельным нам потребуется атомарный доступ к внешней по отношению к циклу переменной sum, в которой хранится текущее значение суммы, либо использование параметра reduction. Помимо этого, в теле цикла используются дополнительные переменные. Поскольку они являются внешними по отношению к циклу и видимыми в момент объявления цикла параллельным, по умолчанию они являются общими. Однако для корректной работы параллельного цикла они должны быть сделаны частными. В результате получаем параллельный цикл следующего вида:

```
#pragma omp parallel for private(x, fx) reduction(+: sum)
        for ( i = 0; i < intl_count; i++ )
        {
            x = a + h * ((double)i + 0.5);
            fx = f(x);
            sum += fx * h;
        }
```

Мы видим, что даже в такой достаточно простой вычислительной процедуре потребовалось явное объявление частных переменных. Если бы по каким-либо причинам эти переменные были сделаны статическими или глобальными, ситуация потребовала бы использования дополнительных директив. При реализации более сложных вычислительных алгоритмов задача явного указания частных и общих переменных может оказаться гораздо сложнее, в результате чего станет легко ошибиться в деталях.

Данный пример был протестирован на компьютерах с 1 – 2 – 4хядерной архитектурами. Покажем график зависимости скорости вычислений от количества используемых ядер.

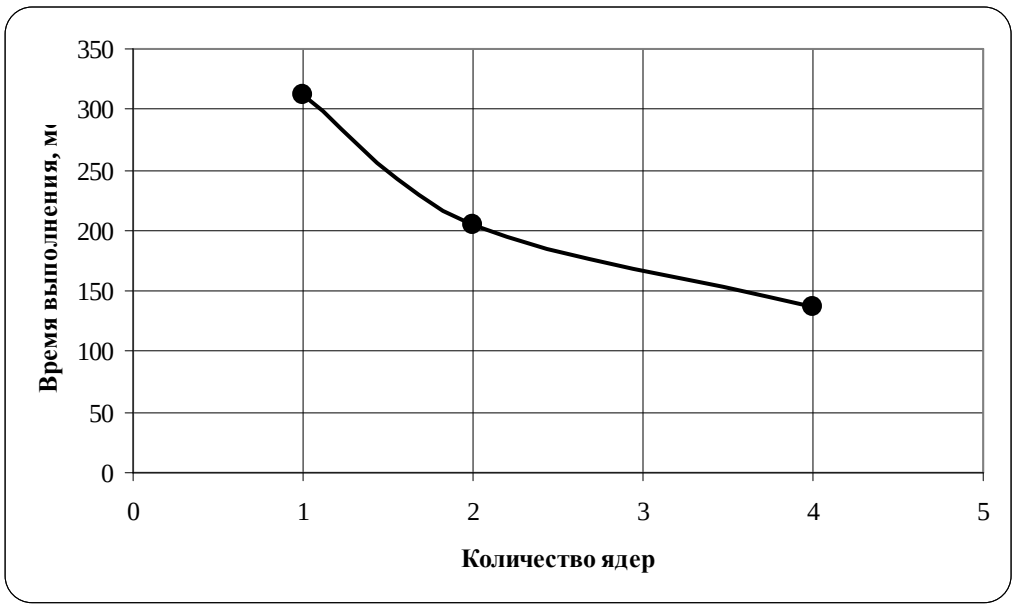

Рисунок 2.4 – Приращение вычислительной скорости при постоянной точности вычисления

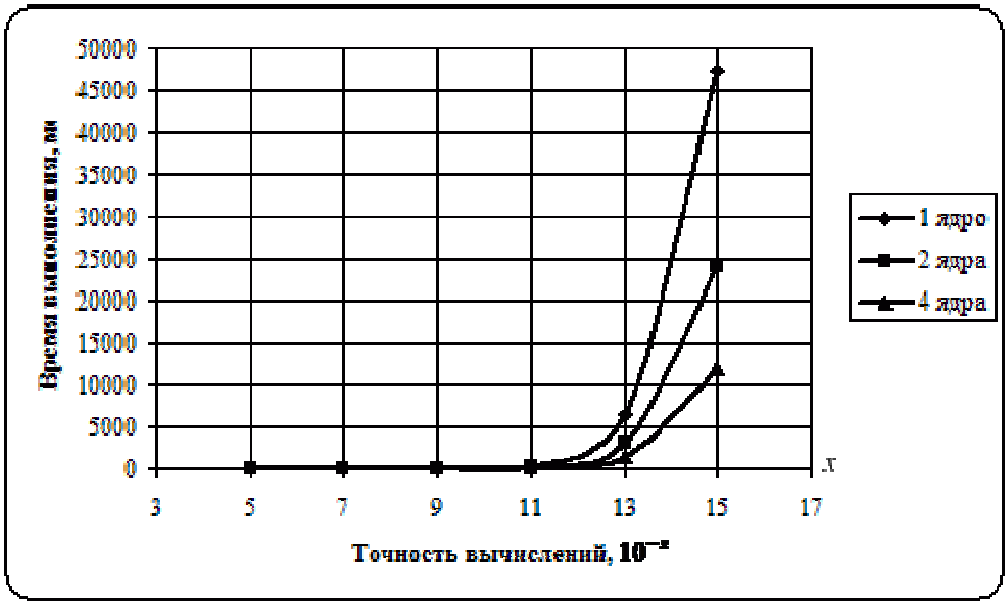

Рисунок 2.5 – Приращение вычислительной скорости при переменной точности вычисления

2.6 Реализация решения задачи параллельного умножения матриц с использованием OpenMP

На примере простой программы умножения матриц можно увидеть, как использовать OpenMP для параллелизации программы. Рассмотрим следующий небольшой фрагмент кода умножения двух матриц. Это очень простой пример, и если действительно необходимо написать хорошую подпрограмму для умножения матричных структур, следует принять во внимание эффекты кэширования или исполь-

зовать более эффективные алгоритмы (Штрассена, или Копперсмита и Винограда, либо другой).

Покажем простейший программный код последовательного умножения матриц:

```
#include <stdio.h>
#include <malloc.h>
#include <windows.h>
#include <omp.h>

int calc( int matrix_size );

int main()
{
    DWORD time;
    int i;

    for ( i = 100; i < 1000; i = i + 100 )
    {
        printf( "Start calculation for %d size\n", i );
        time = GetTickCount();
        calc( i );
        printf( "Calculation time is %d ms\n", GetTickCount() - time );
    }

    return 0;
}

int calc( int matrix_size )
{
    int i, j, k;
    double *a, *b, *c;

    a = (double *)calloc( matrix_size * matrix_size, sizeof( double ) );
    b = (double *)calloc( matrix_size * matrix_size, sizeof( double ) );
    c = (double *)calloc( matrix_size * matrix_size, sizeof( double ) );

    for ( i = 0; i < matrix_size; i++ )
    {
        for ( j = 0; j < matrix_size; j++ )
        {
            for ( k = 0; k < matrix_size; k++ )
            {
                a[i * matrix_size + j] += b[i * matrix_size + k] * c[k * matrix_size + j];
            }
        }
    }

    free( a );
    free( b );
    free( c );

    return 0;
}
```

Можно заметить, что на каждом уровне вложенности циклов блоки циклов могут выполняться независимо друг от друга. Так образом мы имеем право параллелизовать приведенный выше код простым путем: можно вставить прагму #pragma omp parallel for перед самым внешним циклом (циклом по переменной i). Лучше всего вставить прагму перед самым внешним циклом, так как это даст наибольший

выигрыш в производительности. В параллелизованном цикле переменные a, b, c и matrix_size являются общими для потоков, а переменные i, j и k являются частными для каждого потока. Приведенный выше код теперь приобретет следующий вид:

```
#pragma omp parallel for shared(a, b, c, matrix_size) private(i, j, k)
    for ( i = 0; i < matrix_size; i++ )
    {
        for ( j = 0; j < matrix_size; j++ )
        {
            for ( k = 0; k < matrix_size; k++ )
            {
                a[i * matrix_size + j] = b[i * matrix_size + k] * c[k * matrix_size + j];
            }
        }
    }
```

Необходимо заметить, что данный пример всего лишь отражает суть применения методов распараллеливания на основе OpenMP, но не является единственным.

Данный пример был протестирован на компьютерах с 1 – 2 – 4хядерной архитектурами. Покажем график зависимости скорости вычислений от количества используемых ядер.

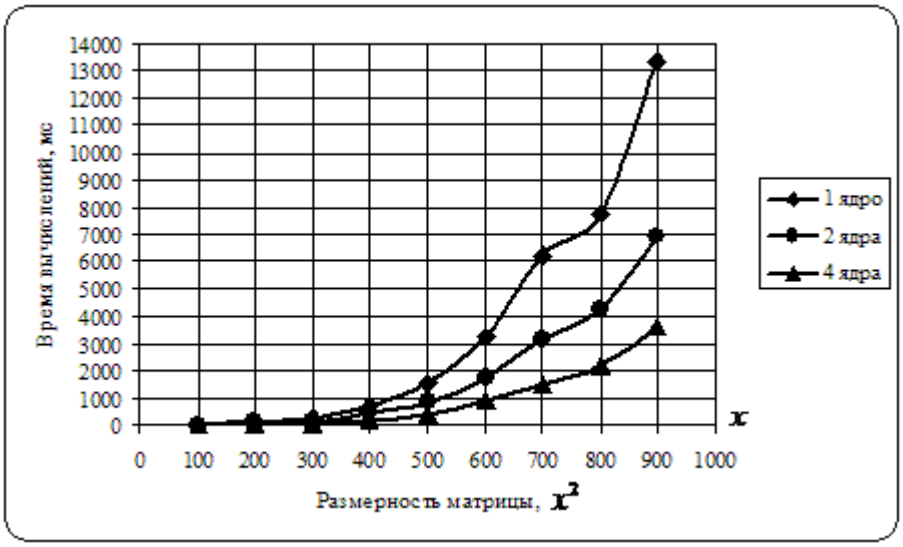

Рисунок 2.6 – Приращение вычислительной скорости при переменной размерности матриц

2.7 Использование смешанного типа программирования с использованием стандартов OpenMP и MPI

Параллелизм, основанный на пересылке сообщениями, имеет высокий потенциал для разработки высокоэффективных и хорошо масштабируемых приложений, потому, что коммуникация данных и синхронизация процессов находятся полно-

стью под контролем пользователей. Модель передачи сообщений, поддерживаемая MPI, обеспечивает портируемость для создания параллельных приложений на распределенных архитектурах так же хорошо, как и на архитектурах с общей памятью. Объединение MPI и стандарта OpenMP может предоставить возможность использовать иерархический параллелизм, реализованный в приложениях или включенный в возможности аппаратного обеспечения. Подобного типа смешанный стиль программирования предоставляет более эффективные методы параллелизма, чем MPI, для использования которого необходимо явно указывать схему пересылки сообщений и использовать планировщик распределения задач. Параллелизм, основанный на стандарте OpenMP, предоставляет единое адресное пространство для каждого процесса, что дает возможность повысить уровень параллелизма.

У смешанного программирования MPI/OpenMP есть потенциал для увеличения масштабируемости приложений, но есть недостатки.

Использование смешанного типа программирования может быть выгодным в следующих ситуациях:

1) Такой программный подход хорошо применим к SMPs кластерам при использовании MPI для передачи данных через узлы и стандарт OpenMP в пределах узлов. Используя общую память, мы можем увеличить производительность приложений на основе стандарта OpenMP, без значимого увеличения объемов памяти для необходимых конфигураций.
2) Некоторые приложения предоставляют два уровня параллелизма: «плохой» параллелизм, осуществляемый при использовании MPI, где большое количество вычислений может быть выполнено независимо; и "хороший" параллелизм, который может быть эффективен на уровне циклов.
3) Системные требования приложений или системные ограничения могут ограничивать число одновременных MPI задач. В этом случае используя стандарт OpenMP в дополнение к MPI можно существенно увеличить уровень параллелизма.
4) Некоторые приложения проявляют несбалансированную рабочую нагрузку на уровне MPI. В этом случае, стандарт OpenMP обеспечивает удобный путь из-

бегания подобных ситуаций, используя дополнительный параллелизм на уровне общего адресного пространства и назначая различного числа потоков различным процессам MPI, в зависимости от рабочей нагрузки.

Использование смешанного типа программирования может быть невыгодным, когда:

1) Включение стандарта OpenMP в существующий код MPI влечет за собой включение недостатков стандарта OpenMP, таких как:
   - ограниченные возможности по управлению распределением задач и их синхронизациями;
   - дополнительные действия по созданию потока и его синхронизации;
   - зависимость от качества компилятора и средства динамической поддержки стандарта OpenMP;
   - зависимость от пригодности разделяемой памяти и связанных проблем, например размещение данных.
2) Взаимодействие MPI и библиотек поддерживающих стандарт OpenMP может оказывать негативный эффект на производительность программы, в зависимости от аппаратного обеспечения.
3) Некоторые приложения естественно выставляют только один уровень параллелизма, и тем самым не могут быть адаптированными под выше описанный способ представления иерархического параллелизма.

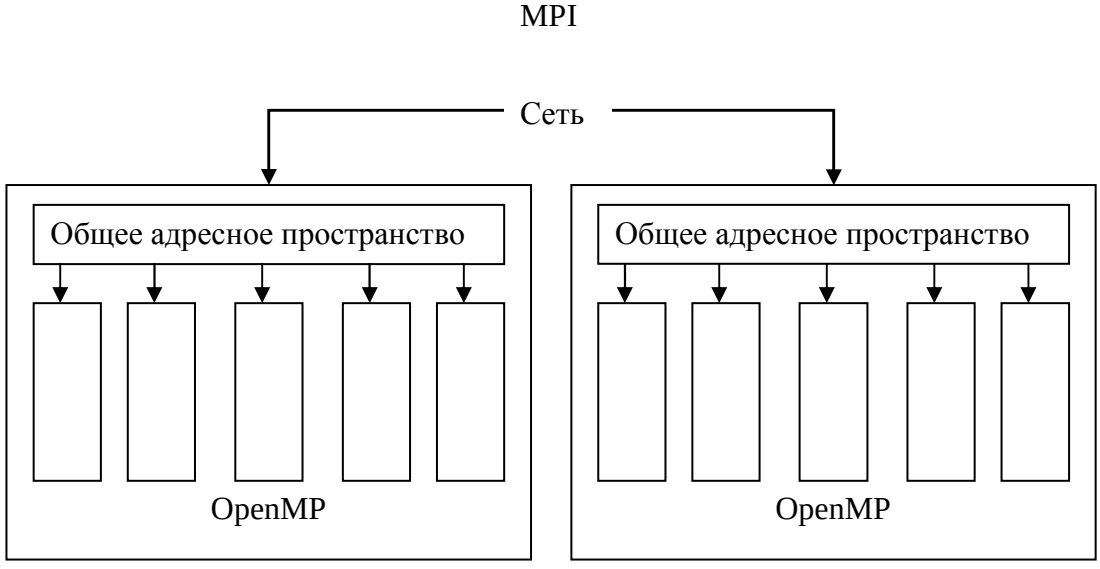

Рисунок 2.7 – Смешанная схема параллелизма OpenMP/MPI

ВЫВОДЫ

Получено решение одномерной начально-краевой задачи теплопроводности для фрактальной среды с порядками производных по времени $0<\alpha\le 2$ и по пространственной переменной $1<\beta\le 2$. Из решения и численного эксперимента вытекает следующие качественные особенности процесса.

При $\alpha<1$ $(\alpha=0{,}5)$ и $\beta=1{,}9$ наблюдается медленный процесс теплопроводности – температура достигает равновесного значения при $t\approx 12$ с, в то время как согласно классическому закону $(\alpha=1)$ – при $t\approx 2$ с. При $\alpha>1$ $(\alpha=1{,}9)$ процесс теплопроводности приобретает волновой характер, при $t=1{,}5$ с происходит отражение от границы. Температура изменяется по разному при различных значениях β (при постоянном $\alpha=1$). Однако термодинамическое равновесие наступает в один и тот же момент времени $t\approx 2$ с. При $(\alpha=0{,}5)$ и $\beta=1.9$ процесс близок к диффузионному, а при $\alpha=1.5;\ \beta=1.9$ процесс теплопроводности близок к волновому.

2. Рассмотрена новая трехмерная граничная задача фрактальной теплопроводности для слоя с туннельной полостью, на поверхности которой действует прямоугольный импульс. Получено точное и асимптотическое решение для температурной функции, приведены кривые эволюции температуры для различных дробных значений α.

Из результатов численных экспериментов можно сделать вывод, что распространение тепла (нагрев) в слое с полостью для сред с показателем $\alpha<1$ осуществляется медленнее в отличие от сред, для которых $\alpha=1$. При $1<\alpha\le 2$ процесс, происходящий во фрактальной среде, можно охарактеризовать как процесс перехода от классической теплопроводности к волновому поведению.

3. Рассмотрена квазистатическая задача термоупругости для фрактальной среды (полупространство) с производными Рисса и Капуто. В литературе указывается, что при $0<\alpha<1$ система ведет себя как термоизолятор, а при $1<\alpha<2$ приходим к аномальной теплопроводности с коэффициентом проводимости, стремящемся к

бесконечности в термодинамическом пределе. Приведенные в отчете результаты на рис. 1.10 - 1.14 подтверждают этот вывод.

4. Рассмотрена задача о внезапном нагреве границы упругого полупространства (фрактальной среды), являющаяся обобщением известной задачи Даниловской. Из полученных результатов следует, что при $\alpha < 0{,}75$ разрыв напряжения фактически исчезает. При $\alpha = 1$, в соответствии с решением Даниловской, имеет место разрыв напряжений с переменой знака при $t = x_1 / c_1$.