

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

”Новосибирский национальный исследовательский государственный университет”  
(Новосибирский государственный университет)

Структурное подразделение Новосибирского государственного университета –  
Высший колледж информатики НГУ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

## Создания модуля для параллельного решения бигармонического уравнения методом Монте-Карло

Дипломный проект  
на квалификацию техник

Студент  
гр. 903а2

Семенов С.А.  
”\_\_\_”\_\_\_\_\_2013

Научный руководитель  
к.ф-м.н., н.с ИВМиМГ СО РАН

Лукинов В.Л.  
”\_\_\_”\_\_\_\_\_2013

Новосибирск 2013

## ОГЛАВЛЕНИЕ

<b>ВВЕДЕНИЕ</b> .....	<b>3</b>
<b>ГЛАВА 1 Постановка задачи</b>	<b>7</b>
1.1 Конкретизации требований и задачи . . . . .	7
1.2 Формулировка задачи . . . . .	8
1.3 Аналогии . . . . .	8
<b>ГЛАВА 2 Методы решения и алгоритмы</b>	<b>9</b>
2.1 Блуждание по сферам . . . . .	9
2.1.1 Оценки решения метагармонического уравнения $(\delta + c)^p u = g$	10
2.2 Алгоритмы распределения задачи . . . . .	10
<b>ГЛАВА 3 Итоги работы</b>	<b>12</b>
3.1 Численные результаты . . . . .	12
<b>ГЛАВА 4 Руководство пользователя</b>	<b>13</b>
4.1 Установка . . . . .	13
4.2 Запуск приложения . . . . .	13
<b>ЗАКЛЮЧЕНИЕ</b> .....	<b>14</b>
<b>СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ</b> .....	<b>15</b>
<b>ГЛАВА Приложение</b>	<b>16</b>
<b>ГЛАВА А Исходные коды и диаграммы программ</b>	<b>16</b>
А.1 Основной аналог - Biharmon2 измененная . . . . .	16
<b>ГЛАВА В Справка</b>	<b>17</b>
В.1 Исполняемая программа . . . . .	17
В.2 Библиотека . . . . .	17
В.3 Файл данных . . . . .	17

<b>ГЛАВА C Выходные данные</b>	<b>18</b>
C.1 tex,html . . . . .	18
<b>ГЛАВА D Тезаурус</b>	<b>19</b>

## ВВЕДЕНИЕ

Дипломная работа посвящена созданию эффективной библиотеки для численного решения первой краевой задачи для бигармонического уравнения методами Монте-Карло.

Официальной датой рождения метода Монте-Карло принято считать 1949 год, когда была опубликована статья С. Улама и Н. Метрополиса [1]. Сам термин был предложен еще во время Второй мировой войны выдающимися учеными XX века математиком Дж. фон Нейманом и физиком Энрико Ферми в Лос-Аламосе (США) в процессе работ по ядерной тематике. Хотя методы Монте-Карло были известны и до 40-х годов, интенсивное развитие статистическое моделирование получило несколько позже в связи с появлением компьютеров, что позволило проводить вычисления больших объемов. С другой стороны, более широкое распространение получает статистическое описание тех или иных сложных физических процессов в связи с чем методы Монте-Карло все более активно используются во многих научных областях (теория переноса, теория массового обслуживания, теория надежности, статистическая физика и др.).

Основными преимуществами данных методов являются:

- физическая наглядность и простота реализации,
- малая зависимость трудоемкости задачи от размерности,
- возможность решения задач со сложной геометрией,
- оценивание отдельных функционалов от решения без запоминания значений решения во всей области,
- вероятностные представления позволяют строить обобщенные решения уравнений,
- одновременное оценивание вероятностной погрешности оценки искомого функционала,
- простое распараллеливание методов.

Бигармонические уравнения используются при решении задач теории упругости. Например, уравнение изгиба тонких пластин имеет вид  $\Delta\Delta u = g$ , где  $u$  – нормальный прогиб пластины. Если пластина лежит на упругом основании, то  $u$  удовлетворяет уравнению  $\Delta\Delta u + cu = g$ .

В настоящей работе использовались алгоритмы, основанные на двух принципиально разных подходах к решению краевых задач методом Монте-Карло.

Первый подход заключается в сведении исходной дифференциальной задачи к некоторому интегральному уравнению, что дает возможность использовать развитой аппарат методов Монте-Карло для решения интегральных уравнений второго рода. На этой основе строятся алгоритмы “блуждания по сферам”.

Во втором подходе дифференциальная задача заменяется соответствующей разностной, которую после приведения её к специальному виду возможно решить методом Монте-Карло. В рамках этого подхода получаются простые и универсальные алгоритмы “блуждания по решетке”

Несмотря на то, что рассматриваемые в дипломе алгоритмы хорошо изучены, новой и неисследованной является задача изучения данных алгоритмов при вычисления на кластерах. Основная задача дипломной работы состоит в построении масштабируемой вычислительной библиотеки для кластерных вычислений.

При создании требуемой библиотеки были использованы следующие программные средства и технологии.

Message Passing Interface (MPI, интерфейс передачи сообщений) – программный интерфейс (API)<sup>1</sup> для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу. Разработан Уильямом Гроуппом, Эвином Ласком и другими.

MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании. Существуют его реализации для большого числа компьютерных платформ. MPI используется при разработке программ для кластеров и суперкомпьютеров. Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. Стандартизацией MPI занимается MPI Forum. В стандарте MPI описан интерфейс передачи сообщений, который должен поддерживаться как на платформе, так и в приложениях пользователя. В настоящее время существует большое количество бесплатных и коммерческих реализаций MPI. Существуют реализации для языков Фортран 77/90, Java, Си и Си++.

---

<sup>1</sup> см. сокращение

В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP<sup>2</sup> ориентирован на системы с общей памятью (многоядерные с общим кэшем). Обе технологии могут использоваться совместно, дабы оптимально использовать в кластере многоядерные системы. Более подробно об этом [3].

При разработке прикладного кода использовалась распределённая система управления версиями файлов – Git

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день поддерживается Джунио Хамано.

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools). Программа является свободной и выпущена под лицензией GNU GPL 2.

Язык для написания приложения был выбран C++ как наиболее подходящий для разработки статической библиотеки.

Статическая библиотека в программировании — сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО) выполненных в виде исходного текста, подключаемого программистом к своей программе на этапе написания, либо в виде объектных файлов, присоединяемых (линкуемых) к исполняемой программе на этапе компиляции. В результате

---

<sup>2</sup><http://openmp.org/wp/>

программа включает в себя все необходимые функции, что делает её автономной, но увеличивает размер. Без статических библиотек объектных модулей (файлов) невозможно использование большинства современных компилирующих языков и систем программирования: Fortran, Pascal, C, C++ и других.

Статическая библиотека присоединяется во время компиляции программы в то время как присоединение динамической происходит во время выполнения.

# 1 Постановка задачи

## 1.1 Конкретизации требований и задачи

Входными условиями вычисления (пользовательскими функциями) является определение:

- функции  $\phi$ ;
- функций  $u, g$ ;
- границ области.

Функции  $\phi, u, g$  соответствуют функциям в уравнении:  $(\Delta + c)^{p+1}u = -g, (\Delta + c)^k u|_{\Gamma} = \phi_k$ . Функция границ области возвращает единицу если точка с некоторой погрешностью находится на границе. Входными данными является:

- количество путей;
- начальная точка.

Для задания пользовательских функций мы можем использовать программный код, прессинг функций или скрип. Первый наиболее скор в разработки, но заставляет компилировать программу каждый раз когда мы меняем вычисляемое уравнение. Для борьбы с этим недостатком сделаем вычисление в классе, который вынесем в отдельный модуль. Получаемый модуль параллельного вычисления скомпилируем как статическую библиотеку. Определение пользовательских функций проходит как задания функций обратного вызова. Так-же сделаем шаблон программы для облегчения определения пользователем своих функций. В комплект необходимо вести реализацию под конкретные условия.

С учетом того, что конечный программный продукт будет запускается как с изменением предыдущих параметров так и для частного конкретного случая ввод данных следует сделать с помощью аргументов и(или) файлов данных.

Вывод осуществляется на экран.

Конкретизируем задачу:

- а) Создание статической библиотеки класса с функциями обратного вызова.
- б) Создание приложение под конкретные условия.
- в) Создание файла данных под программу созданную по предыдущим условиям.
- г) Создание справки.



Интерфейс программы смотреть приложение "Справка".

## **1.2 Формулировка задачи**

Создать библиотеку параллельного вычисления вышеоговоренной задачи с удобным интерфейсом. Снабдить библиотеку примером и сопутствующей документацией (описание интерфейса и методов запуска).

В связи с трудностью задания условий задачи скриптовыми методами и ориентировании на малый объем и неделимость готовой программы, использовать функции обратного вызова.

Для быстрого изменения работы программы без изменения ее структуры обеспечить выбор методов решения и алгоритмов распределения задач флагами.

Обеспечить отказоустойчивость и защищенность от дурака создание отдельного класса с необходимым интерфейсом.

## **1.3 Аналоги**

Для решения данного класса задач инженерами и математиками используются самописные программы последовательного и параллельного вычисления, а также скриптовые математические пакеты. Последние в силу своей структуры не позволяют решить задачу методами представленными в данной дипломной работе. Самописные программы же требуют изучения языка хотя бы высокого уровня. Что заставляет людей изучать в принципе не нужные им вещи на достаточном высоком уровне. Создание же библиотеки снижает этот уровень.

Главным аналогом на основе которого и разрабатывается приложение является программа Viharmon2. Все сравнительные тесты проводились именно с ней. Численные результаты полученные приложением являются эталонными. Алгоритм этой программы приведен в приложении. Недостатком данной реализации алгоритмов является:

- необходимость изменять алгоритм и функции основной программы (малая степень защиты от дурака);
- последовательность вычислений;
- при изменении алгоритма вычисления меняется и часть программы.

## 2 Методы решения и алгоритмы

В работе рассматриваются два метода решения (блуждание по сферам и блуждание по решетке) и два алгоритма распределения задачи (статический и динамический).

### 2.1 Блуждание по сферам

Рассмотрим задачу Дирихле для уравнения Гельмгольца

$$\Delta u + cu = g, u|_{\Gamma} = \phi \quad (2.1)$$

в области  $D \subset R^n$  с границей  $\Gamma$ , причем  $c < c^*$ , где  $c^*$  первое собственное число оператора Лапласа для области  $D$ ,  $r = (x_1; \dots; x_n) \in D$ . Предполагаются выполненными сформулированные условия регулярности функций  $g$ ,  $\phi$  и границы  $\Gamma$ , обеспечивающие существование и единственность решения задачи, а также его вероятностное представление и интегральное представление с помощью шаровой функции Грина.

Введем следующие обозначения:

- $\bar{D}$  – замыкание области  $D$ ;
- $d(P)$  – расстояние от точки  $P$  до границы  $\Gamma$ ;
- $\epsilon > 0$  – числовой параметр;
- $\Gamma_\epsilon$  –  $\epsilon$  – окрестность границы  $\Gamma$ , т. е.  $\Gamma_\epsilon = \{P \in \bar{D} : d(P) < \epsilon\}$ ;
- $S(P)$  максимальная из сфер (точнее - из гиперсфер) с центром в точке  $P$ , целиком лежащих в  $\bar{D}$ ,  $S(P) = \{Q \in \bar{D} : |Q - P| = d(P)\}$ .

В процессе блуждания по сферам очередная точка  $P_{k+1}$  выбирается равномерно по поверхности сферы  $S(P_k)$ ; процесс обрывается, если точка попадает в  $\Gamma_\epsilon$ . Дадим точное определение процесса блуждания по сферам. Зададим цепь Маркова  $\{R_m\}_{m=1,2,\dots,N}$  следующими характеристиками:

- $\pi(r) = \delta(r - r_0)$  - плотность начального распределения (т.е. цепь выходит из точки  $r_0$ );
- $p(r, r') = \delta_r(r')$  плотность перехода из  $r$  в  $r'$ , представляющая собой обобщенную плотность равномерного распределения вероятностей на сфере  $S(r)$ ;

–  $p_0(r)$  вероятность обрыва цепи, определяемая выражением

$$p_0(r) = \begin{cases} 0, r \notin \Gamma_\epsilon \\ 1, r \in \Gamma_\epsilon \end{cases} \quad (2.2)$$

–  $N$  - номер последнего состояния.

Как уже указывалось, данная цепь называется процессом блуждания по сферам. Ее можно, очевидно, записать в виде  $r_m = r_{m-1} + \omega_m d(r_{m-1})$ ;  $m = 1; 2; \dots$ ;  $\omega_m$  – последовательность независимых изотропных векторов единичной длины.

### 2.1.1 Оценки решения метагармонического уравнения $(\delta + c)^p u = g$

Для случая  $L = \delta, \lambda = 0, c = const < c^*$  вероятностное представление задачи имеет вид

$$u(r_0) = E \int_0^\gamma e^{ct} g(\xi(t)) dt + E[e^{c\tau} \Phi(\xi(\tau))], \quad (2.3)$$

где  $\xi(t)$  начинающийся в точке  $r_0$  соответствующий оператору Лапласа диффузионный процесс,  $\tau$  момент первого выхода процесса из области  $D$ . На основе строго марковского свойства процесса отсюда имеем

$$u(r_0) = \sum_{i=0}^{\infty} E[e^{c\tau_i} \int_0^{\tau_{i+1}-\tau_i} e^{ct} g(\xi(t + \tau_i)) dt + E[\Phi(\xi(\tau)) \prod_{i=0}^{\infty} e^{c(\tau_{i+1}-\tau_i)}], \quad (2.4)$$

где  $\tau_i$  - момент первого выхода процесса  $\xi(t)$  на поверхность  $i$ -й сферы соответствующего блуждания по сферам.

## 2.2 Алгоритмы распределения задачи

В библиотеке используется два метода статический (равное распределение задачи при равных вычислительных узлах) и динамический (предполагает разные вычислительные мощности отдельных узлов).

Статический метод распределяет задачи в равных долях плюс процесс для не нулевого и вычит общего числа процессов на нулевом. Это связано с общим суммированием результата на нулевом процессе.

Динамический распаралеливает половину задачи в равных долях между не нулевыми процессами (в дальнейшем – считоводах). Нулевой процесс выполняет роль менеджера распределения задач (МРЗ). При окончании расчетов первым считоводом МПЗ получает коэффициент производительности (КП) вы-

раженный в секундах на проход. Этой велечене присваевается единичный статус и назад отправляется количество проходов равное первоначальным условиям, но от оставшегося числа. Если остается меньше необходимого минимума он отправляется полностью, а МРЗ переходит к сбору данных. При приходе последующих КП они сравниваются с единичным. Если он меньше, то текущей становится единичным. Назад отправляем

$$nWay = \frac{N}{2size \frac{Kp_i}{Kp_1}}, \quad (2.5)$$

где  $Kp$  - коэффициент производительности,  $size$  - общее количество прочесов. Если  $nWay$  меньше необходимого минимума отправляется  $N$  полностью, а МРЗ переходит к сбору данных.

### 3 Итоги работы

#### 3.1 Численные результаты

Рассмотрим численные результаты для следующей задачи Дихиле:

$$\begin{cases} (\Delta + 2)(\Delta + 8)u = 40e^x e^y \\ (\Delta + 8)u|_{\Gamma} = 10e^x e^y \\ u|_{\Gamma} = e^x e^y \end{cases} \quad (3.1)$$

в единичном квадрате. Решение данной задачи дано в [2].

Таблица 3.1 — Время выполнения и расхождения

Metot	N	$t_{cp}$	$\Delta N$
Par1	100003	1 sec	+/- 200
Par1	1000030	10 sec	+/- 250
Par2	100003	1 sec	+/- 10
Par2	1000030	12 sec	+/- 10
Pos	100003	2 sec	—
Pos	1000030	25 sec	—

В таблице представлены результаты производительности, оценки точности и численные результаты совпадают для всех методов. Используются следующие сокращения для таблицы: Pos - результаты получены в аналоге, Par1 - статичный метод расчета, Par2 - динамичный метод расчета.

## 4 Руководство пользователя

### 4.1 Установка

Для сборки приложения под Windows необходимо MPICH2, набор утилит для компиляции: компилятор GNU GCC и GNU Make данные утилиты представлены в пакете MinGW. Для Linux GNU GCC не обязателен, компиляция происходит силами пакета MPICH2.

- а) Скачайте необходимую версию библиотеки с тестовым примером.
- б) Запустите консоль в папке проекта или перейдите в нее с помощью команды `cd`.
  - нажмите Пуск -> Выполнить -> `cmd` - Это откроет консоль Windows;
  - в консоли наберите имя диска на котором располагается проект с двоиточием на конце (C:);
  - там же напечатайте `cd <путь к проекту> (cd C: project)`.
- в) Запустите `make`.

Результатом станет скомпилированный тестовый пример и статическая библиотека находящиеся в папке с проектом.

### 4.2 Запуск приложения

Для запуска приложения набрать в консоли:

linux

`mpirun -n <Кол. процессов> <Имя программы> [опции]`

windows

`mpiexec -n <Кол. процессов> <Имя программы> [опции]`

Опции

`-n<Кол.Путей>` - Задаёт количество путей

`-f` - Печать в файл

Пример:

`mpirun -n 100 ./main -n3000 -f`

## **ЗАКЛЮЧЕНИЕ**

Сформируем основные результаты работы:

- а) Создана статической библиотеки класса с функциями обратного вызова .
- б) Создана приложение под конкретные условия.
- в) Оценино оптимизация для двух алгоритмов и двух методов решения.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. S. Ulam and N. Metropolis. The Monte Carlo method, Journal of American Statistical Association, 44, 335, 1949.
2. Лукинов Виталий Леонидович *Скалярные Алгоритмы метода Монте-Карло для решения мета-гармонических уравнений* 2005
3. Официальный сайт MPICH. "Документация MPICH" [Электронный ресурс] Электрон. ст. М. 2013 – URL:<http://www.mpich.org/> (дата обращения: 10.06.2013).



# А Исходные коды и диаграммы программ

## А.1 Основной аналог - Biharmon2 измененная

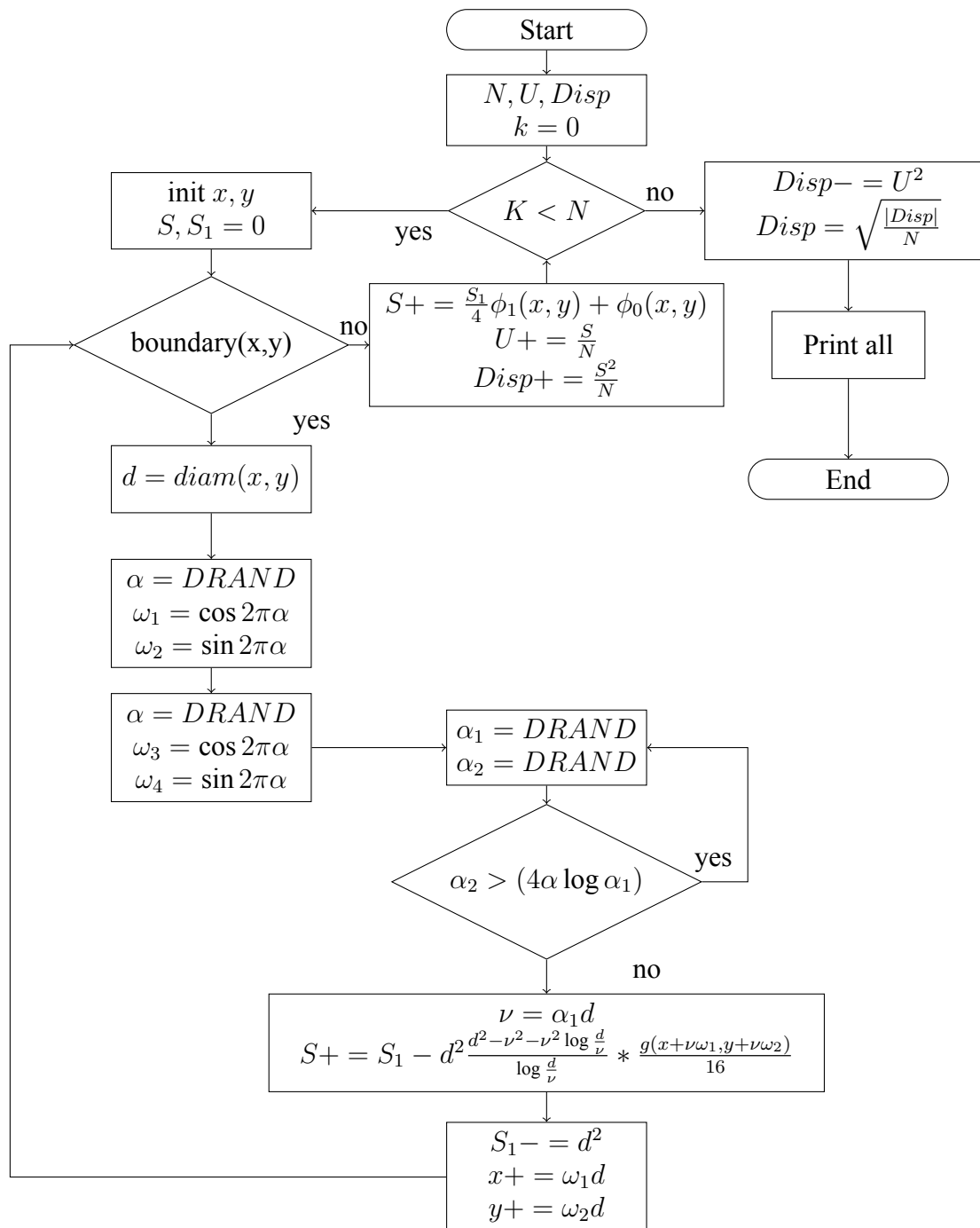


Рисунок А.1 — Принцип действия программы

## В Справка

### В.1 Исполняемая программа

biharmon -l [[ -d ]] -h

biharmon -l [[ -d ]] -h [filename]

- l Latex-file - происходит создание файла report.tex

- d Display - вывод данных на дисплей

- h html-file - происходит создание файла report.html

Если файл существует к имени добавляется индекс.

filename - файл данных;

Приоритет -l, -h, -d

### В.2 Библиотека

Открытые методы класса

init(int argc, char \*args[])

setFunPhi();

### В.3 Файл данных

Построчный.

X Y [SectionName]

## С Выходные данные

### С.1 tex,html

Отчет

$N=10^4$

$x, y$	$h$	$u(r)$	$u(r)$	$ u(r) $
0.5, 0.5	0.1	0.22	0.22	0.0004

## D Тезаурус

API - Интерфейс программирования приложений (иногда интерфейс прикладного программирования) (англ. application programming interface) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

MPI - Message Passing Interface (интерфейс передачи сообщений) — API для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.