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 Семенов С.А.
гр. 903а2 «___»_____2013

Научный руководитель Лукинов В.Л.
к.ф-м.н., н.с ИВМиМГ СО РАН «___»_____2013

Новосибирск 2013

Оглавление

Введение

Данный дипломный проект рассматривает приложения распределенного вычисления отклонения пластины под воздействием статичных сил. Вычисление проводились методом Монте-Карло, а выполнен он с помощью технологии MPI (Message Passing Interface). Одной из целей работ является сравнение эффективности приложений распределенного и последовательного вычисления, а также общая оценка эффективности алгоритма.

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.).

Одна из схем решения краевых задач методом Монте-Карло заключается в сведении исходной дифференциальной задачи к некоторым конечно-разностным уравнениям. Для решение данных уравнений используется ”Алгоритм блуждания по решетке” и ”Алгоритм блуждания по сферам”. Основным недостатком данных алгоритмов является большой объем независимых друг от друга случайных вычислений. Данная независимость вычислений позволяет распараллелить их. Для распараллеливания данного действия и применяется MPI.

Message Passing Interface (MPI, интерфейс передачи сообщений) — программный интерфейс (API)[1] 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++.

В первую очередь MPI ориентирован на системы с распределенной памятью, то есть когда затраты на передачу данных велики, в то время как OpenMP[2] 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]

[1] см. сокращение

[2] http://openmp.org/wp/

Глава 1

Постановка задачи

1.1. Конкретизации требований и задачи

Входными условиями вычисления (пользовательскими функциями) является определение:

- функции ϕ;
- функций u, g;
- границ области.

Функции ϕ, 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 в tex, html файлы и на экран.

Конкретизируем задачу:

1. Создание статической библиотеки класса с функциями обратного вызова .
2. Создание приложение под конкретные условия.
3. Создание файла данных под программу созданную по предыдущим условиям.
4. Создание справки.

Интерфейс программы смотреть приложение ”Справка”.

1.2. Формулировка задачи

Создание MPI приложения вычисление отклонения пластины под воздействием статичных сил.

1.3. Аналоги

Главным аналогом на основе которого и разрабатывается приложение является программа Biharmon2, чей код 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.

Литература

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. http://www.mpich.org/

Приложение А

Исходные коды и диаграммы программ

А.1. Основной аналог - Biharmon2 измененная

```
#include <iostream>
#include <stdlib.h>
#include <cmath>                          // for trigonometry functions

using namespace std;

 const double epsilon=0.001e-00;
 const double xfirst=0.5e-00;
 const double yfirst=0.5e-00;
 const double PI=3.141592e-00;
 double x;
 double y;



 double u(double x, double y)
 {
         return double(exp(x)*exp(y));
         //return double(sin(x)*sin(y)*sin(z));
 }

 double g(double x, double y)
 {
         return double(-4.0*exp(x)*exp(y));
        // return double(-4.0*sin(x)*sin(y)*sin(z));
 }
 double phi_0(double x, double y)
 {
     return double(exp(x)*exp(y));
        // return double(sin(x)*sin(y)*sin(z));
 }
 double phi_1(double x, double y)
 {
         return double(2.0*exp(x)*exp(y));
        // return double(-2.0*sin(x)*sin(y)*sin(z));
 }
```

```
int boundary(double x1, double y1) // являетсялиточкаграницей
{
        if (x1<epsilon) {x=0.0; return 0;};
        if (x1>(1-epsilon)) {x=1.0; return 0;};
        if (y1<epsilon) {y=0.0; return 0;};
        if (y1> (1-epsilon)) {y=1.0; return 0;};
        return 1;

}

double min(double a, double b)
{
        if ( b>=a)  return a;
           else return b;

}
double diam(double x, double y) // диаметрматрицы  ?
{
        if (x> fabs(1-x)) x= fabs(1-x); // fabs – абсолютноезначениедля

//аргумента сплавоющейточкой
        if (y>fabs(1-y)) y=fabs(1-y); // пишемвглобальныепеременные
        return min(x,y); // возвращаемминимальное  R ?

}
double stat2(double a, double b, double c)
{
     if ( c>=b) if (b>=a) return b;
              else if (a>=c) return c;
                                  else return a;
    else if (c>=a) return c;
                else if (b>=a)return a;
                  else return b;

}



int main()

{
  double N; // количествопутей

  cout << "\nInput a number of ways\n";
  cin >> N;
  double U=0;
  double Disp=0;
  for (int k= 0; k<N; k++ )
  {


                        x=xfirst;
```

```
                y=yfirst;

                int SN=0;
                double S=0;
                double S1=0;

                while (boundary(x,y))// поканеграница
            {


                    double d=diam(x,y);
                    double alpha= double (rand())/ double(RAND_MAX
?
                    double omega1=cos(2*PI*alpha);
                    double omega2=sin(2*PI*alpha);
                    alpha= double (rand())/ double(RAND_MAX);
                    double om1=cos(2*PI*alpha);
                    double om2=sin(2*PI*alpha);
                    int indeks=1;
                    double alpha1=0;
                    double alpha2=0;
                    while(indeks)
                    {
                            alpha1=double (rand())/ double(RAND_MA
                            alpha2=double (rand())/ double(RAND_MA
                            alpha2=4*alpha2/exp(1);
                            if (alpha2<(-4*alpha1*log(alpha1))) in

                    }
                        //cout<< "\n  "<<alpha1 <<" "<< alpha2 <<

                    double nu=alpha1*d;
                    S=S+(S1-((d*d-nu*nu-nu*nu*log(d/nu))/log(d/nu)
                    SN=SN+1;
                    S1=S1-d*d;
                    x=x+omega1*d;
                    y=y+omega2*d;
                    //cout << "\n  "<<x <<" "<< y<<  ;
            };
            //cout << "\n boundary  "<<x <<" "<< y ;
            S=S+(S1/4)*phi_1(x,y)+phi_0(x,y);
            //S=S+phi_0(x,y,z);
            cout <<"\n␣␣␣␣"<< SN;
            U=U+S/N;
            Disp=Disp+(S*S)/N;


    };
    Disp=Disp-U*U;
    Disp=sqrt(fabs(Disp)/N);

    cout << "\nPresise␣solution␣␣␣"<< u(xfirst,yfirst) ;
```

```
    cout << "\nNumerical␣solution␣␣␣␣"<< U ;
    cout << "\nDelta␣␣␣␣"<< fabs(u(xfirst,yfirst)-U) <<"\n" ;
    cout << "\nDisp␣␣␣␣"<< Disp <<"\n" ;

        return 0;

}
```

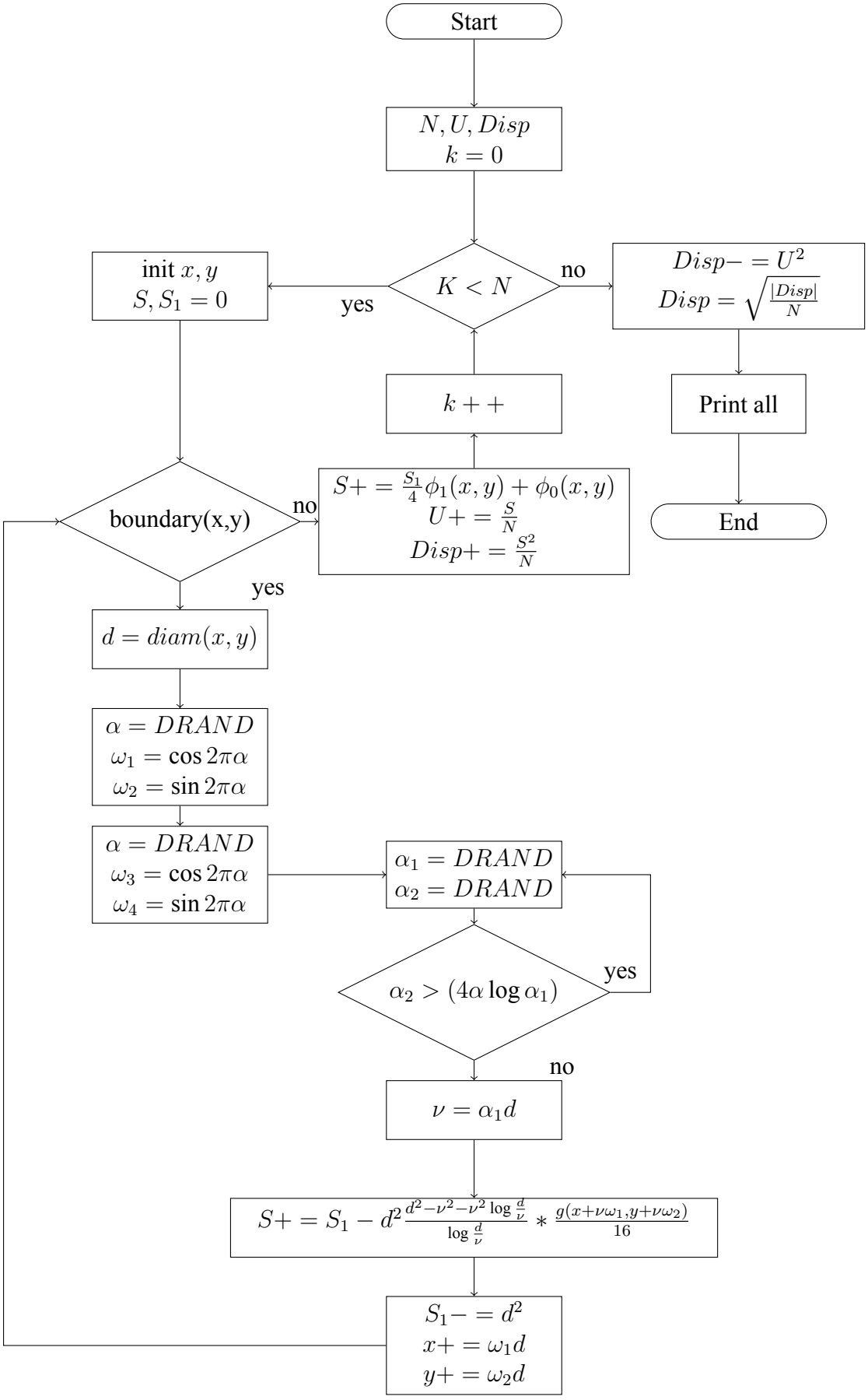

Рис. А.1: Принцип действия программы

Приложение B

Справка

B.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

B.2. Библиотека

Открытые методы класса
init(int argc,char *args[])
setFunPhi();

B.3. Файл данных

Построчный.
X Y [SectionName]

Приложение C

Выходные данные

C.1. tex,html

Отчет
N=10^4

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

Приложение 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.