САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА

ВЫЧИСЛЕНИЕ ИНТЕГРАЛОВ МЕТОДОМ МОНТЕ - КАРЛО

Выполнил:

Руководитель:

Саратов, 2009

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc137123909)

[1. МАТЕМАТИЧЕСКОЕ ОБОСНОВАНИЕ АЛГОРИТМА ВЫЧИСЛЕНИЯ ИНТЕГРАЛА](#_Toc137123910)

[1.1 Принцип работы метода Монте – Карло](#_Toc137123911)

[1.2 Применение метода Монте – Карло для вычисления n – мерного интеграла.](#_Toc137123912)

[1.3 Сплайн – интерполяция 8](#_Toc137123913)

[1.4 Алгоритм расчета интеграла](#_Toc137123914)

[2. ГЕНЕРАТОР ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ](#_Toc137123915)

[2.1 Генератор псевдослучайных чисел применительно к методу Монте – Карло.](#_Toc137123916)

[2.2 Алгоритм генератора псевдослучайных чисел](#_Toc137123917)

[2.3 Проверка равномерности распределения генератора псевдослучайных чисел.](#_Toc137123918)

[ЗАКЛЮЧЕНИЕ](#_Toc137123919)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc137123920)

# ВВЕДЕНИЕ

Целью данной работы является создание программного продукта для участия в конкурсе, проводимом группой компаний «Траст» по созданию программных разработок. Для реализации было выбрано следующее технической задание:

**Задание 12 Вычисление интегралов методом Монте – Карло.**

**Цель:**

1. Реализация генератора случайных чисел для метода Монте – Карло.
2. Сравнение равномерного распределения и специально разработанного.
3. Вычисление тестового многомерного интеграла в сложной области.

**Продукт:**

1. Программный код в виде функции на языке С++ или Fortran .
2. Тестовые примеры в виде программы, вызывающие реализованные функции.
3. Обзор использованной литературы.

Для реализации данного технического задания был выбран язык C++. Код реализован в интегрированной среде разработки приложений Borland C++ Builder Enterprises и математически обоснован соответствующий способ вычисления интеграла.

# 1. МАТЕМАТИЧЕСКОЕ ОБОСНОВАНИЕ АЛГОРИТМА ВЫЧИСЛЕНИЯ ИНТЕГРАЛА

##

## 1.1 Принцип работы метода Монте – Карло

Датой рождения метода Монте - Карло признано считать 1949 год, когда американские ученые Н. Метрополис и С. Услам опубликовали статью под названием «Метод Монте - Карло», в которой были изложены принципы этого метода. Название метода происходит от названия города Монте – Карло, славившегося своими игорными заведениями, непременным атрибутом которых являлась рулетка – одно из простейших средств получения случайных чисел с хорошим равномерным распределением, на использовании которых основан этот метод.

Метод Монте – Карло это статистический метод. Его используют при вычислении сложных интегралов, решении систем алгебраических уравнений высокого порядка, моделировании поведения элементарных частиц, в теориях передачи информации, при исследовании сложных экономических систем.

Сущность метода состоит в том, что в задачу вводят случайную величину , изменяющуюся по какому то правилу . Случайную величину выбирают таким образом, чтобы искомая в задаче величина  стала математическим ожидание от , то есть .

Таким образом, искомая величина  определяется лишь теоретически. Чтобы найти ее численно необходимо воспользоваться статистическими методами. То есть необходимо взять выборку случайных чисел  объемом . Затем необходимо вычислить выборочное среднее  варианта случайной величины  по формуле:

. (1)

Вычисленное выборочное среднее принимают за приближенное значение .

Для получения результата приемлемой точности необходимо большое количество статистических испытаний.

Теория метода Монте – Карло изучает способы выбора случайных величин  для решения различных задач, а также способы уменьшения дисперсии случайных величин.

##

## 1.2 Применение метода Монте – Карло для вычисления n – мерного интеграла.

Рассмотрим *n –* мерный интеграл

 для . (2)

Будем считать, что область интегрирования , и что  ограниченное множество в . Следовательно, каждая точка *х* множества  имеет *n* координат: .

Функцию  возьмем такую, что она ограничена сверху и снизу на множестве : .

Воспользуемся ограниченностью множества  и впишем его в некоторый *n* – мерный параллелепипед , следующим образом:

,

где - минимумы и максимумы, соответственно,  - ой координаты всех точек множества : .

Доопределяем подынтегральную функцию  таким образом, чтобы она обращалась в ноль в точках параллелепипеда , которые не принадлежат :

 (3)

Таким образом, уравнение (2) можно записать в виде

. (4)

Область интегрирования представляет собой *n –* мерный параллелепипед  со сторонами параллельными осям координат. Данный параллелепипед можно однозначно задать двумя вершинами , которые имеют самые младшие и самые старшие координаты всех точек параллелепипеда.

Обозначим через  *n*-мерный вектор, имеющий равномерное распределение в параллелепипеде : , где .

Тогда ее плотность вероятностей  будет определена следующим образом

 (5)

Значение подынтегральной функции  от случайного вектора  будет случайной величиной , математическое ожидание  которой является средним значением функции на множестве :

. (6)

Среднее значение функции на множестве  равняется отношению значения искомого интеграла к объему параллелепипеда :

 (7)

Обозначим  объем параллелепипеда .

Таким образом, значение искомого интеграла можно выразить как произведение математического ожидания функции и объема *n*- мерного параллелепипеда :

 (8)

Следовательно, необходимо найти значение математического ожидания . Его приближенное значение можно найти произведя *n* испытаний, получив, таким образом, выборку  случайных векторов, имеющих равномерное распределение на . Обозначим  и . Для оценки математического ожидания воспользуемся результатом

, (9)

где ,

,

 - квантиль нормального распределения, соответствующей доверительной вероятности .

Умножив двойное неравенство из (9) на  получим интервал для *I*:

. (10)

Обозначим  точечную оценку . Получаем оценку (с надежностью ):

. (11)

Аналогично можно найти выражение для относительной погрешности :

. (12)

Если задана целевая абсолютная погрешность , из (11) можно определить объем выборки, обеспечивающий заданную точность и надежность:

. (13)

Если задана целевая относительная погрешность, из (12) получаем аналогичное выражение для объема выборки:

. (14)

##

## 1.3 Сплайн – интерполяция.

В данном программном продукте реализована возможность задавать дополнительные ограничения области интегрирования двумя двумерными сплайн – поверхностями (для подынтегральной функции размерности 3). Для задания этих поверхностей используются двумерные сплайны типа гибкой пластинки \4\.

Под сплайном (от англ. spline - планка, рейка) обычно понимают агрегатную функцию, совпадающую с функциями более простой природы на каждом элементе разбиения своей области определения. Сплайн – функция имеет следующий вид:

. (15)

Исходные данные представляют собой  троек точек .

Коэффициенты  и  определяются из системы:

, (16)

где ,



.

##

## 1.4 Алгоритм расчета интеграла

Реализованный алгоритм включает следующие шаги:

1. выбирается начальное значение , разыгрываются случайные векторы из  и определяются  и ;
2. в зависимости от вида погрешности (абсолютная, относительная) определяется достигнутая погрешность; если она меньше целевой, вычисление прерывается;
3. по формулам (13) или (14) вычисляется новый объем выборки;
4. объем выборки увеличивается на 20%
5. переход к шагу 1;
6. конец.

# 2. ГЕНЕРАТОР ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

##

## 2.1 Генератор псевдослучайных чисел применительно к методу Монте – Карло.

В любом алгоритме использующем метод Монте – Карло генератор псевдослучайных чисел играет очень важную роль. Степень соответствия псевдослучайных чисел заданному распределению является важным фактором проведения качественных статистических испытаний.

##

## 2.2 Алгоритм генератора псевдослучайных чисел

В программе реализован конгруэнтный метод генерации псевдослучайных чисел \3\:

, (17)

где =8192,

=67101323.

Авторский код, реализующий защиту от переполнения был, реализован на С++. Перед использование первые три числа последовательности удаляются. Для получении чисел из интервала (0,1) все числа делятся на .

##

## 2.3 Проверка равномерности распределения генератора псевдослучайных чисел.

Проверка равномерности распределения псевдослучайных чисел проводилась с помощью стандартного критерия χ2 \2\.

Были использованы 3 последовательности псевдослучайных чисел, определяемых стартовыми значениями 1, 1001, 1000000 длиной 300000.

Интервал (0,1) подразделялся на 50 равных интервалов и программно подсчитывались абсолютные частоты (рис. 1).

Рис. 1

Результаты проверки приведены в Таблице 1.

Таблица 1

|  |  |
| --- | --- |
|  | стартовое значение ГСЧ |
|  | 1 | 1001 | 1000000 |
| хи-квадрат | 44.0533333333333 | 45.007 | 48.618 |
| df | 50 | 50 | 50 |
| p-значение | 0.709735881642893 | 0.673522612551685 | 0.528941919633451 |

Следовательно, равномерность распределения не отвергается на уровне 5%.

# ЗАКЛЮЧЕНИЕ

В заключение можно сказать, что поставленная задача была полностью выполнена. То есть на языке С++ были разработаны генератор псевдослучайных чисел, функция рассчитывающая интеграл методом Монте – Карло (Приложение 1); был проведен расчет тестовых многомерных интегралов (Приложение 2); в интегрированной среде разработки приложений Borland C++ Builder Enterprises 7.0 был создан программный продукт «CarloS», реализующий описанные выше алгоритмы (Приложение 3).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бережная Е. В., Бережной В. И. Математические методы моделирования экономических систем. – М.: Финансы и статистика, 2001. – 368 с.
2. Мюллер П., Нойман П., Шторм Р. Таблицы по математической статистике. – М.: Финансы и статистика, 1982. – 278 с.
3. Теннант-Смит Дж. Бейсик для статистиков. – М.: Мир, 1988. – 208 с.
4. Baranger J. Analyse numérique. Hermann, 1991.
5. Маделунг Э. Математический аппарат физики. Справочное руководство. М.: Наука, 1968., с.287.
6. В.Е. Гмурман Теория вероятностей и математическая статистика – М.: Высшая школа, 2003

ПРИЛОЖЕНИЕ 1

**ЛИСТИНГИ ОСНОВНЫХ ФУНКЦИЙ**

**Листинг 1 Функция расчета интеграла**

**void** integral ()

**{**

 // *вычисление интеграла методом Монте – Карло*

 // *размерность области интегрирования*

 **unsigned** d\_int=fun\_dim;

 *//----- 3 d график --------------------------------------------------------*

 *// максимальное число троек*

 **unsigned** plot\_dim\_max=10000;

 *// матрица троек*

 pmatd xyz,xyz\_tmp;

 if (d\_int==3) xyz=new matd(plot\_dim\_max,3);

 *//-------------------------------------------------------------------------*

 *// индикатор относительной погрешности*

 mcres.relok=Read1double("error\_type.txt");

 *// целевая погрешность*

 mcres.dlt\_int=Read1double("error\_value.txt");

 *// номер стандартного значения доверительной вероятности (начиная с 0)*

 **int** nome\_int=Read1double("error\_omega.txt");

 *// ГСЧ*

 **unsigned long** b=m\_rng\*m\_rng-d\_rng,c,r,i,PSChunk;

 *// "росток" ГСЧ*

 mcres.rng\_seed=Read1double("rng\_seed.txt");

 pmatd fun\_b, fun\_A, con\_b, con\_A, con\_U, con\_v, \

 a\_int, b\_int, ba\_int, x\_int, xyz\_top, xyz\_bottom;

 **unsigned** j,ii,jj,con\_ok;

 **struct** date dat;

 **struct** time tim;

 pspl2d sp\_top,sp\_bottom;

 *// квантили нормального распределения*

 **double** omegas\_int[6]=**{**0.9,0.95,0.99,0.999,0.9999,0.99999**}**;

 **double** zs\_int[6]=**{**1.64485362695147,1.95996398454005,2.5758293035489, \

 3.29052673149191, 3.89059188641317, 4.4171734134667**}**;

 mcres.omega\_int=omegas\_int[nome\_int];

 mcres.z\_int=zs\_int[nome\_int];

 **double** fun\_cd,con\_wd,fu\_int,con\_sum,sum1\_int,sum2\_int;

 *// вид интегрируемой функции*

 *// 0 - постоянная*

 *// 1 - линейная*

 *// 2 - квадратичная*

 mcres.fun\_type=Read1double("fun\_kind.txt");

 *// вид системы ограничений*

 *// 0 – отсутствуют (весь параллелепипед)*

 *// 1 - линейные*

 *// 2 - квадратичное*

 *// 3 – сплайн - поверхности*

 mcres.con\_type=Read1double("con\_type.txt");

 *// загрузка параметров интегрируемой функции*

 **switch** (mcres.fun\_type)

 **{**

 **case** 2: fun\_A=**new** matd("fun\_A.txt");

 **case** 1: fun\_b=**new** matd("fun\_b.txt");

 **case** 0: fun\_cd=Read1double("fun\_c.txt");

 **}**

 *// загрузка параметров ограничений*

 **switch** (mcres.con\_type)

 **{**

 **case** 3: *// сплайн - поверхности*

 *// верхняя*

 xyz\_top=**new** matd("xyz\_top.txt");

 *// нижняя*

 xyz\_bottom=**new** matd("xyz\_bottom.txt");

 *// двумерная интерполяция*

 sp\_top=**new** spl2d(xyz\_top);

 sp\_bottom=**new** spl2d(xyz\_bottom);

 **break;**

 **case** 2: *// квадратичная функция ограничений*

 con\_U=**new** matd("con\_U.txt");

 con\_v=**new** matd("con\_v.txt");

 con\_wd=Read1double("con\_w.txt");

 **break;**

 **case** 1: *// линейные ограничения*

 con\_b=**new** matd("con\_b.txt"); con\_A=**new** matd("con\_A.txt");

 **}**

 *// объемлющий параллелепипед*

 a\_int=**new** matd("con\_xmin.txt");

 b\_int=**new** matd("con\_xmax.txt");

 *// разность границ параллелепипеда*

 ba\_int=**new** matd;

 ba\_int=&(\*b\_int - (\*a\_int));

 *// аргумент интегрируемой функции*

 x\_int=**new** matd(d\_int,1);

 *//объем объемлющего параллелепипеда*

 mcres.V0\_int=1;

 **for** (j=1; j <= d\_int; j++)

 **{**

 **if** (\_p(ba\_int,j,1) <= 0)

 **{**

 DbBox("Нижняя граница объемлющего параллелепипеда выше верхней для \

координаты ",j);

 **goto** clean\_exit;

 **}**

 mcres.V0\_int=mcres.V0\_int\*\_p(ba\_int,j,1);

 **}**

 *// начальный объем выборки*

 mcres.n1\_int=10000;

 *// основной цикл для достижения заданной точности*

 *// число итераций, потребовавшихся для достижения заданной точности*

 mcres.n\_ite=0;

 getdate(&dat); gettime(&tim); mcres.t\_start=dostounix(&dat,&tim);

 WaitForm->Show();

 **while** (1)

 **{**

 mcres.n\_ite++;

 WaitForm->Edit1->Text=mcres.n\_ite;

 WaitForm->Edit2->Text=mcres.n1\_int;

 WaitForm->ProgressBar1->Position=0;

 WaitForm->Refresh();

 *// генерация случайных точек и накопление суммы*

 sum1\_int=0; sum2\_int=0;

 mcres.in\_G\_int=0;

 PSChunk=**long**(mcres.n1\_int/50.0);

 *// запуск ГСЧ*

 r=mcres.rng\_seed;

 **for** (i=1; i < 3; i++)

 **{**

 c=**int**(r/m\_rng);

 r=b\*c+m\_rng\*(r-m\_rng\*c);

 **if** (r > d\_rng) r=r-d\_rng;

 **}**

 **for** (i=1; i <= mcres.n1\_int; i++)

 **{**

 *// случайный вектор*

 **for** (j=1; j <= d\_int; j++)

 **{**

 *// случайное число*

 c=**int**(r/m\_rng);

 r=b\*c+m\_rng\*(r-m\_rng\*c);

 **if** (r > d\_rng) r=r-d\_rng;

 \_p(x\_int,j,1)=\_p(a\_int,j,1)+\_p(ba\_int,j,1)\*double(r)/d\_rng;

 **}**

 *// прогресс*

 **if** (!(i % PSChunk))

 **{**

 WaitForm->ProgressBar1->Position=100.0\*(i-1)/(mcres.n1\_int-1);

 WaitForm->Refresh();

 **}**

 *// проверка ограничения*

 con\_ok=1;

 **switch** (mcres.con\_type)

 {

 **case** 3: *// сплайн – поверхности*

 **if** ((\_p(x\_int,3,1) < sp\_bottom->f(\_p(x\_int,1,1), \

\_p(x\_int,2,1)))||(\_p(x\_int,3,1) > sp\_top->f(\_p(x\_int,1,1),\_p(x\_int,2,1)))) con\_ok=0;

 **break**;

 **case** 2: *// квадратичная функция ограничений*

 con\_sum=0;

 **for** (ii=1; ii <= d\_int; ii++)

 **for** (jj=1; jj <= d\_int; jj++)

 **if** (\_p(con\_U,ii,jj) != 0)

 con\_sum += \_p(x\_int,ii,1)\*\_p(con\_U,ii,jj)\*\_p(x\_int,jj,1);

 **for** (ii=1; ii <= d\_int; ii++)

 **if** (\_p(con\_v,ii,1) != 0)

 con\_sum += \_p(con\_v,ii,1)\*\_p(x\_int,ii,1);

 **if** (con\_sum > con\_wd) con\_ok=0;

 **break**;

 **case** 1: *// линейная функция ограничений*

 **for** (ii=1; ii <= con\_A->nl; ii++)

 **{**

 con\_sum=0;

 **for** (jj=1; jj <= d\_int; jj++)

 con\_sum += \_p(con\_A,ii,jj)\*\_p(x\_int,jj,1);

 **if** (con\_sum > \_p(con\_b,ii,1)) **{** con\_ok=0; **break**; **}**

 **}**

 **}**

 fu\_int=0;

 **if** (con\_ok != 0)

 **{**

 mcres.in\_G\_int++;

 *// точки 3d графика*

 **if** (d\_int==3)

 **if** (mcres.in\_G\_int <= plot\_dim\_max)

 **{**

 \_p(xyz,mcres.in\_G\_int,1)=\_p(x\_int,1,1);

 \_p(xyz,mcres.in\_G\_int,2)=\_p(x\_int,2,1);

 \_p(xyz,mcres.in\_G\_int,3)=\_p(x\_int,3,1);

 **}**

 *// значение интегрируемой функции*

 **switch** (mcres.fun\_type)

 **{**

 **case** 2: *// квадратичный член*

 **for** (ii=1; ii <= d\_int; ii++)

 **for** (jj=1; jj <= d\_int; jj++)

 **if** (\_p(fun\_A,ii,jj) != 0)

 fu\_int += \_p(x\_int,ii,1)\*\_p(fun\_A,ii,jj)\*\_p(x\_int,jj,1);

 **case** 1: *// линейный член*

 **for** (ii=1; ii <= d\_int; ii++)

 **if** (\_p(fun\_b,ii,1) != 0)

 fu\_int += \_p(fun\_b,ii,1)\*\_p(x\_int,ii,1);

 **case** 0: *// постоянная*

 fu\_int += fun\_cd;

 **}**

 **}**

 sum1\_int+=fu\_int; sum2\_int+=fu\_int\*fu\_int;

 **}**

 *// оценка мат. ожидания и дисперсии*

 mcres.f1\_int=sum1\_int/mcres.n1\_int;

 mcres.vari\_int=(sum2\_int-sum1\_int\*sum1\_int/mcres.n1\_int)/(mcres.n1\_int-1);

 *// расчет погрешности*

 **if** (mcres.relok==0)

 **{**

 *// абсолютная погрешность*

 mcres.deltar=mcres.V0\_int\*mcres.z\_int\*sqrt(mcres.vari\_int/mcres.n1\_int);

 **}**

 **else**

 **{**

 *// относительная погрешность*

 **if** (mcres.f1\_int!=0)

 **{**

 mcres.deltar=mcres.z\_int/fabs(mcres.f1\_int)\*sqrt(mcres.vari\_int/mcres.n1\_int);

  **}**

 **else**

 **{**

 *// форма результатов*

 mcres.inte\_int=0;

 mcres.deltar=0;

 getdate(&dat); gettime(&tim); mcres.t\_end=dostounix(&dat,&tim);

 mcres.t\_calc=mcres.t\_end-mcres.t\_start;

 InfoBox("*Оценка интеграла = 0 (выбрана относ. погрешность), вычисление \*

*прервано*.");

 ResultForm->Show();

 WaitForm->Close();

 **goto** clean\_exit;

 **}**

 **}**

 WaitForm->Edit3->Text=mcres.deltar;

 WaitForm->Refresh();

 **if** (mcres.deltar < mcres.dlt\_int)

 **{**

 *// точность достаточна*

 mcres.inte\_int=mcres.V0\_int\*mcres.f1\_int;

 getdate(&dat); gettime(&tim); mcres.t\_end=dostounix(&dat,&tim);

 mcres.t\_calc=mcres.t\_end-mcres.t\_start;

 ResultForm->Show();

 **break**;

 **}**

 *// вычисление нового объема выборки*

 **if** (mcres.relok==0)

 **{**

 *// абс. погрешность*

 mcres.n1\_int=ceil(mcres.vari\_int\*pow(mcres.V0\_int\*mcres.z\_int/mcres.dlt\_int,2));

 **}**

 **else**

 **{**

 *// отн.погрешность*

 mcres.n1\_int=ceil(mcres.vari\_int\*pow(mcres.z\_int/mcres.dlt\_int/mcres.f1\_int,2));

 **}**

 *// корректировка объема выборки в большую сторону*

 *//для сокращения числа итераций*

 mcres.n1\_int=1.2\*mcres.n1\_int;

 *// минимальный объем выборки*

 **if** (mcres.n1\_int < 1000) mcres.n1\_int=1000;

 **}** *// конец основного цикла*

 WaitForm->Close();

 *// 3d график*

 **if** (d\_int==3)

 **{**

 **if** (mcres.in\_G\_int==0)

 **{**

 *// множество точек пусто*

 Zero\_File("xyz.txt");

 **}**

 **else**

 **if** (mcres.in\_G\_int < xyz->nl)

 **{**

 *// точек не набралось, чтобы заполнить матрицу*

 xyz\_tmp=new matd(mcres.in\_G\_int,3);

 **for** (i=1; i <= mcres.in\_G\_int; i++)

 **{**

 \_p(xyz\_tmp,i,1)=\_p(xyz,i,1);

 \_p(xyz\_tmp,i,2)=\_p(xyz,i,2);

 \_p(xyz\_tmp,i,3)=\_p(xyz,i,3);

 **}**

 xyz\_tmp->txprint("xyz.txt");

 **delete** xyz\_tmp;

 **}**

 **else**

 **{**

 *// вся матрица заполнена*

 xyz->txprint("xyz.txt");

 **}**

 **}** *// конец d\_int==3*

 clean\_exit:

 *// очистка памяти*

 **if** (d\_int==3) **delete** xyz;

 **switch** (mcres.fun\_type)

 **{**

 **case** 2: **delete** fun\_A;

 **case** 1: **delete** fun\_b;

 **}**

 **switch** (mcres.con\_type)

 **{**

 **case** 3: **delete** xyz\_top,xyz\_bottom,sp\_top,sp\_bottom; **break**;

 **case** 2: **delete** con\_U,con\_v; **break**;

 **case** 1: **delete** con\_b,con\_A;

 **}**

 **delete** a\_int,b\_int,ba\_int,x\_int;

**}** *//integral*

Листинг 2 структура для хранения результатов расчета интеграла

**struct** mcres\_struct

 **{**

 *// индикатор относительной погрешности*

 **int** relok;

 *// целевая погрешность*

 **double** dlt\_int;

 *// достигнутая погрешность*

 **double** deltar;

 *// доверительная вероятность*

 **double** omega\_int;

 *// квантиль норм. распределения*

 **double** z\_int;

 *// "росток" ГСЧ*

 **unsigned** **long** rng\_seed;

 *// ÷число итераций, потребовавшихся для достижения заданной точности*

 **unsigned** n\_ite;

 *// объем выборки на последней итерации*

 **unsigned** **long** n1\_int;

 *// число точек попавших в область интегрирования*

 **unsigned** in\_G\_int;

 *// интеграл*

 **double** inte\_int;

 *// объем объемлющего параллелепипеда*

 **double** V0\_int;

 *// выборочное среднее*

 **double** f1\_int;

 *// выборочная дисперсия*

 **double** vari\_int;

 *// время начала счета*

 time\_t t\_start;

 *// время окончания счета*

 time\_t t\_end;

 *// продолжительность вычисления интеграла*

 time\_t t\_calc;

 *// вид интегрируемой функции*

 **int** fun\_type;

 *// вид системы огрничений*

 **int** con\_type;

 **}**; *// mcres\_struct*

ПРИЛОЖЕНИЕ 2

**ТЕСТОВЫЕ ПРИМЕРЫ**

Пример 1 Интеграл от квадратичной функции по 3-мерному симплексу.

Точное значение интеграла:





Приближенное значение  найдено для целевой абсолютной погрешности 0.00001.

Погрешность: 0.000034416630896 или 0.014749984670 %.

Примеры 2-10 Объемы многомерных шаров

Точные и приближенные объемы многомерных шаров  приведены в следующей таблице.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Объем точный[[1]](#footnote-1) | Объем приближенный[[2]](#footnote-2) | Оценка CarloS[[3]](#footnote-3) | Относительная погрешность, % |
| 2 |  | 3.1415926535897932385 | 3.1504 | 0.280346543342 |
| 3 |  | 4.1887902047863909846 | 4.2032 | 0.344008520578 |
| 4 |  | 4.9348022005446793096 | 4.98099547511312 | .936071451118 |
| 5 |  | 5.2637890139143245968 | 5.18913116403891 | -1.4183290720439 |
| 6 |  | 5.1677127800499700296 | 5.16153372226575 | -.1195704569352 |
| 7 |  | 4.7247659703314011698 | 4.70163814726423 | -.4895019819476 |
| 8 |  | 4.0587121264167682184 | 3.98117943332154 | -1.9102782035357 |
| 9 |  | 3.2985089027387068695 | 3.30542485033746 | .209668908064 |
| 10 |  | 2.5501640398773454440 | 2.55096385956571 | .31363460384e-1 |

1. Источник [5], с. 287. [↑](#footnote-ref-1)
2. Вычислено в Maple (20 значащих цифр). [↑](#footnote-ref-2)
3. Расчет с целевой относительной погрешностью 2% [↑](#footnote-ref-3)