Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Бурятский государственный университет имени Доржи Банзарова»

Институт математики и информатики

Кафедра прикладной математики и дифференциальных уравнений

«ДОПУСТИТЬ К ЗАЩИТЕ»

Зав.кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_

Цыренжапов Нима Булатович

к.ф-м.н., доцент

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Цыренов Доржи Галданович

Построение квадратурных формул с участием производных, связанных уравнением (m+1)=(G+1)( σ +1), при известных σ и m, и неизвестном G.

(Выпускная квалификационная работа)

Научный руководитель:

\_\_\_\_\_\_\_\_\_\_ к. ф.-м. н , доцент

Цыренжапов Нима Булатович

Дата защиты: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Улан-Удэ

2020

**ОГЛАВЛЕНИЕ**

**Введение** 3

1 **КВАДРАТУРНЫЕ ФОРМУЛЫ С УЧАСТИЕМ ПРОИЗВОДНЫХ** 4

1.1 Методы численного интегрирования. . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.2 Элементарные квадратурные формулы с участием производных . . . . . . . . . . . . . . . ... . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

1.3 Построение квадратурной формулы с участием производных на произвольном участке интегрирования. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

2 **ПРИЛОЖЕНИЕ «ОНЛАЙН-КАЛЬКУЛЯТОР»** 8

2.1 Оконное приложение «Онлайн-калькулятор» . . . . . . . . . . . . . . . . . . . . . 8

2.2 Вычислительные эксперименты. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**Заключение** 9

**Список литературы** 10

**Приложение**  11

**ВВЕДЕНИЕ**

Широкие возможности применения вычислительной техники в практике вычислений способствуют интенсивному развитию теории приближенного интегрирования. Интегрирование является одной из самых распространенных математических операций. В самых различных областях часто приходится вычислять определенные интегралы, для которых невозможно получить точное значение, поэтому задача о приближенном вычислении определенного интеграла является одной из **актуальных задач** вычислительной математики.

**Целью** дипломной работы является построение квадратурных формул с участием производных и разработка оконного приложения, вычисляющего приближено интеграл с помощью квадратурной формулы.

Для достижения поставленной цели были сформулированы следующие **задачи**:

1. Разобраться с квадратурными формулами;

2. Построить элементарные квадратурные формулы с участием производных, при заданных параметрах m и σ;

3. Осуществить построение квадратурных формул общего вида;

4. Провести сравнение вычислений с помощью квадратурных формул с участием производных и формул численного интегрирования;

5. Провести вычислительные эксперименты.

6. Разработать приложение с графическим интерфейсом

Данная работа состоит из двух глав. Первая глава посвящена обзору методов численного интегрирования и построению квадратурной формулы общего вида. Приведен пример решения интеграла по формуле Ньютона-Лейбница и полученной элементарной квадратурной формуле с участием производных.

Во второй главе рассматривается реализация калькулятора, вычисляющего приближенно интеграл и вычислительные эксперименты.

ГЛАВА 1

**КВАДРАТУРНЫЕ ФОРМУЛЫ С УЧАСТИЕМ ПРОИЗВОДНЫХ**

* 1. **Методы численного интегрирования**

Известно, что для большинства функций нельзя вычислить первообразные, поэтому приходится прибегать к методам численного интегрирования. При численном интегрировании по заданной подынтегральной функции строится сеточная функция, и эта функция заменяется интерполяционным многочленом.

Пусть на отрезке , дана непрерывная функция y=f(x), требуется на вычислить определенный интеграл

Заменим данную функцию f(x) на сеточную функцию и вместо точного значения интеграла будем искать его приближенное значение с помощью суммы:

В которой нужно определить коэффициенты и погрешность.

Наиболее простой является формула прямоугольников. Она основана на определении определенного интеграла как предела последовательности интегральных сумм.

Если в определении снять знак предела, то получим формулу прямоугольников численного интегрирования:

Рассмотрим погрешность формулы прямоугольников на всем отрезке численного интегрирования.

Таким образом, метод прямоугольников является методом первого порядка точности (главный член погрешности пропорционален шагу в первой степени).

Теперь будем вычислять определенный интеграл с помощью интерполяционного многочлена Лагранжа первой степени.

Где погрешность, а интерполяционный многочлен Лагранжа первой степени, проведенный через два узла интерполяции.

Или

Где

Теперь интеграл можно представить в виде:

Это выражение называют формулой трапеций численного интегрирования на одном шаге интегрирования.

Для всего отрезка нужно просуммировать n раз:

Погрешность имеет вид:

Таким образом, метод трапеций является методом второго порядка точности относительно шага h.

Теперь заменим подынтегральную функцию интерполяционным многочленом

Где

Сделаем замену

Слагаемые примут вид

Тогда

Откуда

Это выражение называют формулой Симпсона численного интегрирования на паре шагов от до

На всем отрезке необходимо сложить m раз, поскольку имеется m пар отрезков длиной h, получим формулу Симпсона численного интегрирования определенного интеграла:

В формуле Симпсона на всем отрезке погрешность пропорциональна четвертой степени шага, и, следовательно, метод Симпсона является методом четвертого порядка точности (т.е. главный член погрешности пропорционален четвертой степени шага h).

**1.2. Элементарные квадратурные формулы с участием производных.**

Рассмотрим элементарную квадратурную формулу общего вида

 (1)

где G+1 – число точек, лежащих на оси OX, σ – порядок старшей производной.

Академик С.М. Никольский отмечает [2], что если нам известны не только значения функции, но и значения производных в узлах, то при правильном использовании всех этих данных мы можем ожидать более точный результат, чем в случае использования только значений функции.

При построении этой элементарной квадратурной формулы возникает переопределенная линейная система, с помощью которой мы должны найти коэффициенты квадратурной формулы.

Так как единственным образом определить коэффициенты в данном случае невозможно [3], то для нахождения приближенного значения определенного интеграла мы используем ньютоновскую систему узлов, тем самым уменьшим количество узлов. Пусть N – число узлов, M – число одночленов от n переменных, тогда  , где n – размерность, m – гладкость пространства. Так как взята ньютоновская система узлов, то M=N.

Пусть m – точность квадратурной формулы (1). m+1 – число всех одночленов, входящих в произвольный многочлен степени m, которое будет зависеть от выбора параметров G и σ. G+1 – число все узлов лежащих на оси x. σ+1 – число значений функции и ее производных в одной точке, то (G+1)(σ+1) – число всех коэффициентов формулы (1). Тогда точность m формулы (1) определяется из уравнения:

 (2)

Будем использовать полученное уравнение связанности (2) и разложение функции в ряд Маклорена в операторной форме. Теперь коэффициенты будут определяться единственным образом.

**1.1.1. Формула №1.**

Рассмотрим пример при неизвестном G. Пусть . Тогда элементарная квадратурная формула примет вид:

 (3)

Чтобы найти коэффициенты квадратурной формулы, разложим функцию f(x) в ряд Маклорена в операторной форме [1]:

,

где 

Найдем производные:

 (4)

 (5)

Найдем интеграл:

 (6)

Подставим в равенство (3) функции, ее производные и интеграл в виде ряда, отбрасывая слагаемые степень, которых больше 5, и сократим обе части равенства на f(0):

(7)

Приравнивая коэффициенты при соответствующих степенях d, получим СЛАУ с 6 неизвестными:



Решив СЛАУ, получим коэффициенты квадратурной формулы (3):



Квадратурная формула для будет иметь вид:

 (8)

Вычислим интеграл по полученной квадратурной формуле и формуле Ньютона-Лейбница.

Пусть







Точное решение совпадает с решением, вычисленным с помощью квадратурной формулы. Таким образом, была построена элементарная квадратурная формула с участием производных, которая точно интегрирует многочлен до шестой степени.

**1.1.2. Формула №2**

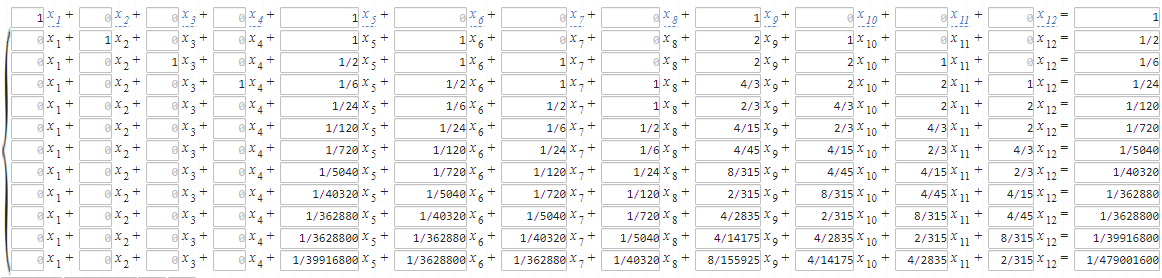
Пусть . Тогда элементарная квадратурная формула примет вид:

 (9)

Найдем третью производную:



И подставим ее и производные (4), (5) и интеграл (6) в нашу квадратурную формулу как мы делали в формуле №1, тогда получим СЛАУ с 12 неизвестными:



Где 





Квадратурная формула для будет иметь вид:



Вычислим интеграл по полученной квадратурной формуле и формуле Ньютона-Лейбница.

Пусть







Точное решение совпадает с решением, вычисленным с помощью квадратурной формулы. Таким образом, была построена элементарная квадратурная формула с участием производных, которая точно интегрирует многочлен до одиннадцатой степени.

**1.3.** **Построение квадратурной формулы с участием производных на произвольном участке интегрирования.**

Построим квадратурную формулу для интеграла , при тех же известных параметрах , что и в *формуле №1*. По свойству аддитивности, определенный интеграл можно разбить на сумму N интегралов:





Каждый полученный интеграл распишем по элементарной квадратурной формуле (8):



Далее произведем преобразования относительно коэффициентов и каждую сумму перед коэффициентами согласно виду усложненной квадратурной формулы, для точности вычисления умножим на уточняющее число, где  – порядок старшей производной у функции, стоящей после коэффициента[4]:

(10)

Таким образом, была получена квадратурная формула общего вида, позволяющая точно интегрировать многочлены до шестой степени на произвольном отрезке интегрирования.

Аналогично, можем получить квадратурную формулу общего вида для второго случая:

 (11)

Которая будет точно интегрировать многочлены до 11 степени на произвольном отрезке интегрирования.

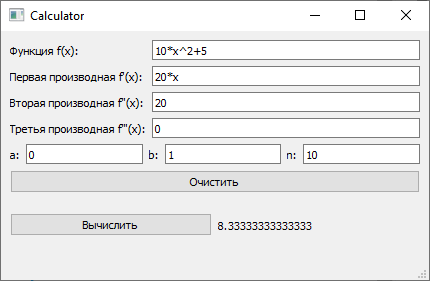
ГЛАВА 2

**ПРИЛОЖЕНИЕ «ОНЛАЙН-КАЛЬКУЛЯТОР»**

**2.1.** **Оконное приложение «Онлайн-калькулятор»**

Код программы был реализован на языке C++. Был написан класс Parser с помощью которого вводятся функции и реализован визуальный интерфейс с помощью фреймворка Qt.

Чтобы вычислить определенный интеграл с помощью квадратурной формулы на произвольном участке интегрирования, пользователь должен ввести функцию и ее производные, пределы интегрирования и количество разбиений.



При нажатии на кнопку «вычислить» программы выводит приближенное значение определенного интеграла. Кнопка «очистить» удаляет все введенные данные.

При вводе функции синтаксический анализатор анализирует исходное математическое выражение и записывает его в vector <string> в удобном для компьютера представлении. Для этого были использованы обратная польская нотация и алгоритм сортировочной станции.

Алгоритм сортировочной станции – способ разбора математического выражения, написанного в инфиксной нотации в постфиксную.

Обратная польская запись – это форма записи математического выражения, в котором операнды записывается перед знаками операций и функций.

При вызове функции в коде, она вычисляется с помощью стека.

Далее программа вычисляет значение интеграла с помощью квадратурной формулы.

Для ввода функции используются операции: +, -, \*, /, унарный минус и функции: sin, cos, tg, ctg, sqrt, log, arcsin, arccos, arctg, arcctg без пробелов.

И константы: e, pi. Переменная: x

**2.2. Вычислительные эксперименты.**

Пример.

Вычислим и сравним значения интеграла функции f(x), полученные по формуле Симпсона и квадратурной формуле на отрезке от 0 до 10.



Найдем значение интеграла по формуле Ньютона-Лейбница:



Вычислим по квадратурной формуле для произвольного участка интегрирования (10):





N=100



Подсчитаем по формуле Симпсона:





Вычислим по квадратурной формуле для произвольного участка интегрирования (11):





Оценим погрешность вычисления сначала для нашей квадратурной формулы(для этого от точного значения вычисления вычтем наш полученный результат), а затем для формулы Симпсона.







Явно видно, что погрешность вычисления квадратурной формулы заметно меньше, чем погрешность вычисления формулы Симпсона.

*Вычисление интегралов на участке от 0 до 10.*

1)  - точное значение

|  |  |  |  |
| --- | --- | --- | --- |
| N | 100 | 500 | 1000 |
| Квадратурная формула №1 | 6.10691309 | 0.03861375866 | 0.038851915 |
| Формула Симпсона | -3.256715 | 0.12223525 | 0.0399380099 |
| Квадратурная формула №2 | -9.3174203751 | 0.03885525235 | 0.038855037385730 |

2) 

|  |  |  |  |
| --- | --- | --- | --- |
| N | 100 | 500 | 1000 |
| Квадратурная формула | -293.928605 | -293.92932622 | -293.929326268 |
| Формула Симпсона | -295.205355 | -293.93129242 | -293.929449007 |
| Квадратурная формула №2 | -293.9293262658 | -293.929326268 | -293.929326268 |

3) 

|  |  |  |  |
| --- | --- | --- | --- |
| N | 100 | 500 | 1000 |
| Квадратурная формула | 999.949363952 | 999.9493634635 | 999.949363435 |
| Формула Симпсона | 999.94904471 | 999.949362983 | 999.949363407 |
| Квадратурная формула №2 | 999.94936343588744 | 999.94936343587 | 999.94936343587141891 |

**ЗАКЛЮЧЕНИЕ**

В ходе работы были изучены квадратурные формулы и найдены коэффициенты для элементарных квадратурных формул с участием производных при m=11, и m=5, . Также были построены формулы для произвольного участка интегрирования.

Было разработано приложение «Онлайн-калькулятор», вычисляющий определенный интеграл с помощью квадратурной формулы (m=11, ). И проведен сравнительный анализ для квадратурных формул, формулы Симпсона и формулы Ньютона-Лейбница.

Таким образом, задача численного интегрирования является одной из актуальных задач вычислительной математики, и используя при интегрировании значения не только функции, но и ее производных мы можем получить более точное значение определенного интеграла.

**СПИСОК ЛИТЕРАТУРЫ**

[1] Березин И.С. Методы вычислений: в 2-х т. /И.С. Березин ,Н.П. Жидков – Москва: Гос. изд. физ.-мат. лит-ры, 1959. – Т.1 – 464 с.

[2] Никольский С.М. Квадратурные формулы. / С.М.Никольский. - Москва: Наука, 1974.- 224 с.

[3] Урбаханов А.В. Оценка погрешности кубатурных формул общего вида с узлами на ньютоновской решетке в пространстве Соболева 𝑊𝑝𝑚(𝐸𝑛): дис.на соиск. учен. степ. канд. физ.-мат. наук (01.01.07)/ Урбаханов Александр Валерьевич. – Улан-Удэ, 2005. – 97 с.

[4] Цыренжапов Н.Б. Оценка погрешности кубатурных формул общего вида с с пограничным слоем и узлами на решетке в пространстве Соболева Lpm(En): дис. На соиск. канд. физ.-мат. наук (01.01.07) /Цыренжапов Нима Булатович. – Улан-Удэ, 2004. – 102 с.

[5] Формалев В. Ф., Ревизников Д. Л. Численные методы. – М.: ФИЗМАЛИТ, 2004. – 400 с.

*ПРИЛОЖЕНИЕ*

**Код для вычисления интеграла по формуле Симпсона**

double simpson(double a, double b, double n)

{

double res=0;

double h, sum2=0, sum4=0, sum=0;

h=(b-a)/n;

for (int i=1; i<=n-1; i+=2)

{

sum4+=f(a+h\*i);

sum2+=f(a+h\*(i+1));

}

sum=f(a)+4\*sum4+2\*sum2-f(b);

return res=(h/3)\*sum;

}

**Код для вычисления интеграла по квадратурной формуле**

double quad\_form(double a, double b, int n)

{

double h=(b-a)/n;

double x0=a, x1=a+h, x2=a+2\*h;

double sum1=0, sum2=0, sum3=0, sum4=0, sum5=0, sum6=0,

sum7=0, sum8=0, sum9=0, sum10=0, sum11=0, sum12=0;

for (int i=0; i<n; i++)

{

sum1+=f(x0);

sum2+=f1(x0);

sum3+=f2(x0);

sum4+=f3(x0);

sum5+=f(x1);

sum6+=f1(x1);

sum7+=f2(x1);

sum8+=f3(x1);

sum9+=f(x2);

sum10+=f1(x2);

sum11+=f2(x2);

sum12+=f3(x2);

x0+=h;

x1+=h;

x2+=h;

}

Double res=24965./59136\*h\*sum1+4357./59136\*h\*h\*sum2+2819./443520\*h\*h\*h\*sum3+41./177408\*h\*h\*h\*h\*sum4+128./231\*h\*sum5-1./6\*h\*h\*sum6+64./3465\*h\*h\*h\*sum7-1./360\*h\*h\*h\*h\*sum8+1403./59136\*h\*sum9-169./19712\*h\*h\*sum10+509./443520\*h\*h\*h\*sum11-17./295680\*h\*h\*h\*h\*sum12;

return res;

}

**Класс Parser**

void Parser :: **toPost**()

{

stack <string> a;

int unary=1;

for (int i=0; i<inf.size(); i++)

{

if (inf[i]>='0' && inf[i]<='9')

{

string n="";

n.push\_back(inf[i]);

while((inf[i+1]>='0' && inf[i+1]<='9' || inf[i+1]=='.') && (i+1<inf.size()))

{

i++;

n.push\_back(inf[i]);

}

post.push\_back(n);

unary=0;

}

else if (inf[i]=='x')

{

post.push\_back("x");

unary=0;

}

else if (inf[i]=='p')

{

post.push\_back("pi");

unary=0;

}

else if (inf[i]=='e')

{

post.push\_back("e");

unary=0;

}

else if (inf[i]==',')

{

while (a.top()!="(")

{

post.push\_back(a.top());

a.pop();

}

unary=1;

}

else if (inf[i]=='(')

{

a.push("(");

unary=1;

}

else if (inf[i]==')')

{

while (a.top()!="(")

{

post.push\_back(a.top());

a.pop();

}

a.pop();

unary=0;

}

else if (inf[i]=='+' || inf[i]=='-')

{

if (unary==1)

{

while (!a.empty())

{

if (a.top()=="log" || a.top()=="cos" || a.top()=="sin" || a.top()=="sqrt" || a.top()=="ctg" || a.top()=="tg" || a.top()=="arccos" || a.top()=="arcsin" || a.top()=="arctg" || a.top()=="arcctg")

{

post.push\_back(a.top());

a.pop();

}

else

break;

}

a.push("~");

}

else

{

while (!a.empty())

{

if (a.top()=="log" || a.top()=="~" || a.top()=="+" || a.top()=="-" || a.top()=="\*" || a.top()=="/" || a.top()=="^" || a.top()=="cos" || a.top()=="sin" || a.top()=="sqrt" || a.top()=="ctg" || a.top()=="tg" || a.top()=="arccos" || a.top()=="arcsin" || a.top()=="arctg" || a.top()=="arcctg")

{

post.push\_back(a.top());

a.pop();

}

else

break;

}

if (inf[i]=='+')

a.push("+");

else

a.push("-");

}

unary=1;

}

else if (inf[i]=='\*' || inf[i]=='/')

{

while (!a.empty())

{

if (a.top()=="log" || a.top()=="~" || a.top()=="\*" || a.top()=="/" || a.top()=="^" || a.top()=="cos" || a.top()=="sin" || a.top()=="sqrt" || a.top()=="ctg" || a.top()=="tg" || a.top()=="arccos" || a.top()=="arcsin" || a.top()=="arctg" || a.top()=="arcctg")

{

post.push\_back(a.top());

a.pop();

}

else

break;

}

if (inf[i]=='\*')

a.push("\*");

else

a.push("/");

unary=1;

}

else if (inf[i]=='^')

{

while (!a.empty())

{

if (a.top()=="log" || a.top()=="~" || a.top()=="cos" || a.top()=="sin" || a.top()=="sqrt" || a.top()=="ctg" || a.top()=="tg" || a.top()=="arccos" || a.top()=="arcsin" || a.top()=="arctg" || a.top()=="arcctg")

{

post.push\_back(a.top());

a.pop();

}

else

break;

}

a.push("^");

unary=1;

}

else if (inf[i]=='s' || inf[i]=='c' || inf[i]=='t' || inf[i]=='a' || inf[i]=='l')

{

if (inf[i]=='s')

{

if (inf[i+1]=='q')

{

a.push("sqrt");

i+=3;

}

else if (inf[i+1]=='i')

{

a.push("sin");

i+=2;

}

}

else if (inf[i]=='c')

{

if (inf[i+1]=='o')

{

a.push("cos");

i+=2;

}

else if (inf[i+1]=='t')

{

a.push("ctg");

i+=2;

}

}

else if (inf[i]=='t')

{

a.push("tg");

i++;

}

else if (inf[i]=='a')

{

if (inf[i+4]=='o')

{

a.push("arccos");

i+=5;

}

else if (inf[i+4]=='t')

{

a.push("arcctg");

i+=5;

}

else if (inf[i+4]=='i')

{

a.push("arcsin");

i+=5;

}

else if (inf[i+4]=='g')

{

a.push("arctg");

i+=4;

}

}

else if (inf[i]=='l')

{

a.push("log");

i+=2;

}

unary=0;

}

}

while (!a.empty())

{

post.push\_back(a.top());

a.pop();

}

}

double Parser :: **calculate**(double x)

{

stack <double> b;

double op1, op2;

for (int i=0; i<post.size(); i++)

{

if(post[i]=="~")

{

op1=b.top();

b.pop();

b.push(-op1);

}

else if (post[i]=="-")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(op1-op2);

}

else if (post[i]=="+")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(op1+op2);

}

else if (post[i]=="\*")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(op1\*op2);

}

else if (post[i]=="/")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(op1/op2);

}

else if (post[i]=="^")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(pow(op1, op2));

}

else if (post[i]=="log")

{

op2=b.top();

b.pop();

op1=b.top();

b.pop();

b.push(log(op2)/log(op1));

}

else if (post[i]=="sin")

{

op1=b.top();

b.pop();

b.push(sin(op1));

}

else if (post[i]=="cos")

{

op1=b.top();

b.pop();

b.push(cos(op1));

}

else if (post[i]=="tg")

{

op1=b.top();

b.pop();

b.push(tan(op1));

}

else if (post[i]=="ctg")

{

op1=b.top();

b.pop();

b.push(cos(op1)/sin(op1));

}

else if (post[i]=="sqrt")

{

op1=b.top();

b.pop();

b.push(sqrt(op1));

}

else if (post[i]=="sin")

{

op1=b.top();

b.pop();

b.push(sin(op1));

}

else if (post[i]=="arcsin")

{

op1=b.top();

b.pop();

b.push(asin(op1));

}

else if (post[i]=="arccos")

{

op1=b.top();

b.pop();

b.push(acos(op1));

}

else if (post[i]=="arctg")

{

op1=b.top();

b.pop();

b.push(atan(op1));

}

else if (post[i]=="arcctg")

{

op1=b.top();

b.pop();

b.push(acos(-1.0)/2-atan(x));

}

else if (post[i]=="x")

{

b.push(x);

}

else if (post[i]=="e")

{

b.push(M\_E);

}

else if (post[i]=="pi")

{

b.push(M\_PI);

}

else

{

b.push(atof(post[i].c\_str()));

}

}

return b.top();

}