Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра «»

Лабораторная работа №7

по дисциплине «Вычислительная математика»

«Методы многомерной оптимизации. Методы первого порядка»

Вариант 15

Выполнил: студент гр. qwinmen.

Проверил:.

Тамбов, 20

**Задача:**

Составить блок-схему алгоритма и реализовать его в программе для вычислений на ЭВМ для следующих методов первого порядка:

1. Метод градиентного спуска с постоянным шагом.

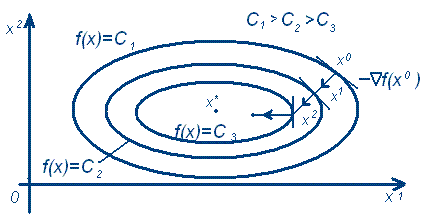
### Метод наискорейшего спуска.

### Метод Гаусса-Зейделя (наискорейшего покоординатного спуска).

### Метод градиентного спуска с постоянным шагом.

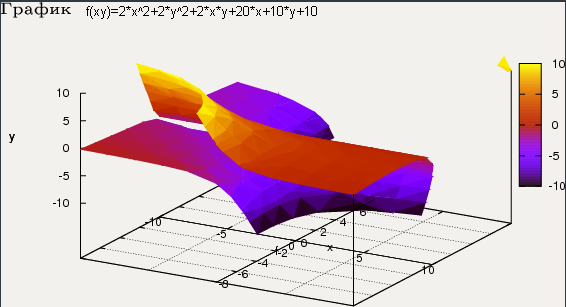
Стратегия поиска:

Стратегия решения задачи состоит в построении последовательности точек таких, что Точки последовательности вычисляются по правилу:

В качестве начала итераций выбирается произвольная точка Величина шага задается пользователем и остаётся постоянной до тех пор, пока функция убывает в точках последовательности, т.е. до тех пор, пока выполняется соотношение Если это условие не выполняется, то производится коррекция длины шага, например , и опять проверяется выполнение неравенства. Процесс завершается в точке для которой выполняется условие , где заданное малое число. Для случая функции двух переменных градиентный метод имеет простую геометрическую интерпретацию: на каждом шаге мы сдвигаемся по вектору антиградиента\*, уменьшенному в раз.

\*Антиградиент функции - вектор противоположный градиенту и направленый в сторону наискорейшего убывания функции.

Дано:



Решение: - задать

Положим Найдем градиент функции в произвольной точке :

- найти производную по икс:

- найти производную по игрек:

Градиент функции есть: ;

Итерация 0:

Вычислить :

;

Проверить выполнение критерия останова :

а) если критерий выполнен, расчет окончен,

б) если критерий не выполнен, то перейти к cледующей итерации.

Итерация 1:

Зададим величину шага ;

Вычислить :

Проверить выполнение условия :

Т.к. то шаг выбран удачно.

Итерация 2:

Т.к.

и т.д.

Искомая точка

Анализ точки:

Найдем матрицу \*Гессе функции

**1. Найдем частные производные**.

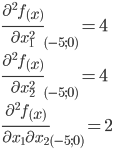
**2. Решим систему уравнений**.

Получим:  
а) Из первого уравнения выражаем x1 и подставляем во второе уравнение:

Данные значения x2 подставляем в выражение для x1. Получаем: x1 = -5  
Количество критических точек равно 1.  
M1(-5;0)

**3. Найдем частные производные второго порядка**.

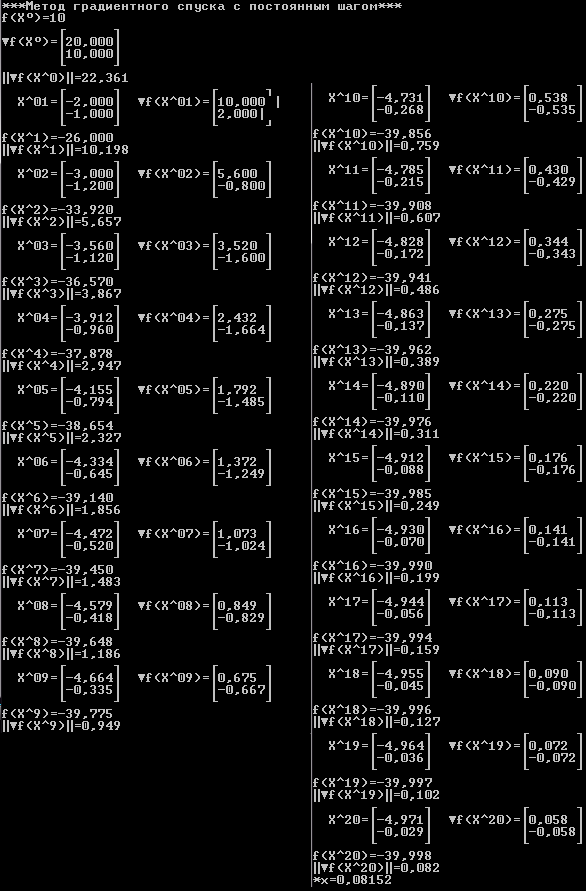
**4. Вычислим значение этих частных производных второго порядка в критических точках M(x0;y0)**.  
Вычисляем значения для точки M1(-5;0)

 и строим матрицу Гессе:

Так как матрица Гессе является положительно определенной, то функция строго выпукла и, следовательно, в стационарной точке достигается глобальный минимум.

\* Матрица Гессе - квадратная матрица размерности (n\*n), составленная из вторых частных производных функции f(x) по всем переменным.

Результат работы программы:



Код программы:

class ГрадиентныйСпуск

{

static double F(double x, double y)

{ return (2 \* (x \* x) + 2 \* (y \* y) + 2 \* x \* y + 20 \* x + 10 \* y + 10); }

static double Sistema\_X(double x, double y)

{ return 4 \* x + 2 \* y + 20; }

static double Sistema\_Y(double x, double y)

{ return 4 \* y + 2 \* x + 10; }

static double Kriterii(double x, double y)

{ return Math.Sqrt((x \* x) + (y \* y)); }

public static void GradF(double x0, double y0, double epsilon)

{//epsilon=0.1//X0=[0;0]

if (epsilon < 0) { Console.WriteLine("Значение epsilon меньше нуля!"); return; }

//Найти градиент функции в точке [x0;y0]=X0

//Итерация 0

double[] x\_X = new double[50], x\_Y = new double[50];

int k = 0;

x\_X[k] = x0; x\_Y[k] = y0;

double fX0 = F(x0, y0);//f(X^0)=10

Console.Write("f(X°)={0}\n", fX0);

Console.WriteLine(@" ┌ ┐

▼f(X°)=│{0:f3}│

│{1:f3}│

└ ┘", Sistema\_X(x0, y0), Sistema\_Y(x0, y0));

double[] kritPoiska = new double[50];

double sistVerxn\_X, sistNizn\_Y;

//Шаг 1

step1:

sistVerxn\_X = Sistema\_X(x\_X[k], x\_Y[k]);//20

sistNizn\_Y = Sistema\_Y(x\_X[k], x\_Y[k]);//10

//kritPoiska[k] это delta f(X^k)

kritPoiska[k] = Kriterii(sistVerxn\_X, sistNizn\_Y);//22.3607

Console.WriteLine("║▼f(X^{1})║={0:f3}", kritPoiska[k],k);

//Шаг 2;

if (kritPoiska[k] < epsilon)

{ Console.WriteLine("\*x={0:f5}", kritPoiska[k]); }

else

{ //Шаг 3

double t0 = 0.1;//"Введите шаг поиска t"

//Шаг 4

step4:

//X^1=[-2;-1]

x\_X[k + 1] = x\_X[k] - t0 \* sistVerxn\_X;//-2

x\_Y[k + 1] = x\_Y[k] - t0 \* sistNizn\_Y;//-1

Console.WriteLine(@" ┌ ┐ ┌ ┐

X^{2:d2}=│{0:f3}│ ▼f(X^{2:d2})=│{3:f3}│

│{1:f3}│ │{4:f3}│

└ ┘ └ ┘", x\_X[k + 1], x\_Y[k + 1], k + 1, Sistema\_X(x\_X[k + 1], x\_Y[k + 1]), Sistema\_Y(x\_X[k + 1], x\_Y[k + 1]));

//Шаг 5+

//f(X^1)= -26

double fX1 = F(x\_X[k + 1], x\_Y[k + 1]);//f(X^1)=-26

Console.WriteLine("f(X^{1})={0:f3}", fX1,k+1);

//Шаг 5. Проверить выполнение условия f(X^1)-f(X^0) < 0

if (fX1 - fX0 < 0)

{ k = k + 1; goto step1; }

else

{

t0 = t0 / 2;

goto step4;

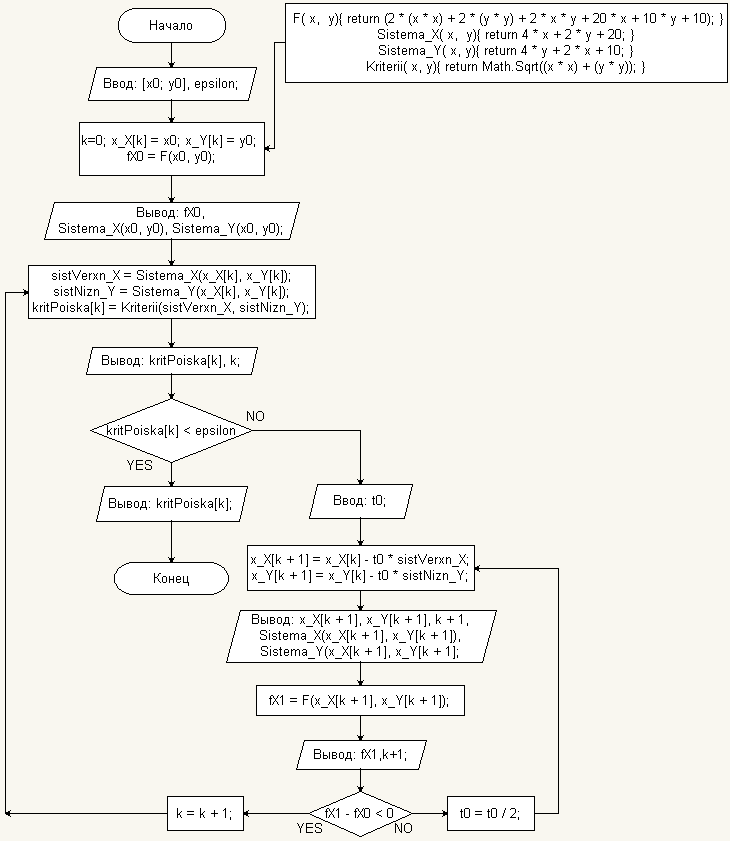
}

}

}

}

Блок схема программы:



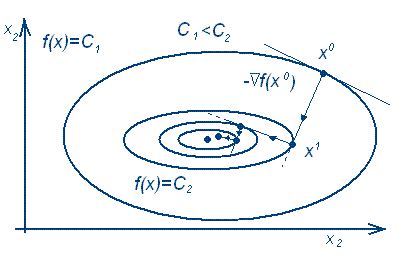
**Метод наискорейшего спуска.**

Стратегия поиска:

Стратегия решения задачи состоит в построении последовательности точек

таких, что Точки последовательности вычисляются по правилу:

В качестве начала итераций выбирается произвольная точка Основное отличие этого метода от градиентного состоит в способе выбора шага При использовании метода наискорейшего спуска на каждой итерации величина шага выбирается из условия минимума функции в направлении спуска, т. е. . Это условие означает, что движение вдоль антиградиента происходит до тех пор, пока значение функции убывает. С математической точки зрения на каждой итерации необходимо решать задачу одномерной минимизации по t функции



В случае функции двух переменных данный метод имеет следующую геометрическую интерпретацию: направление движения из точки касается линии уровня в точке Траектория спуска зигзагообразная, причем соседние звенья зигзага ортогональны друг другу. Действительно, шаг Необходимое условие минимума функции Отсюда получим условие ортогональности векторов направлений спуска в соседних точках:

Дано:

Решение: - задать

Положим Найдем градиент функции в произвольной точке :

- найти производную по икс:

- найти производную по игрек:

Градиент функции есть: ;

Итерация 0:

Вычислить :

;

Проверить выполнение критерия останова :

а) если критерий выполнен, расчет окончен,

б) если критерий не выполнен, то перейти к cледующей итерации.

Итерация 1:

Вычислить :

*Вычислим* величину шага из условия

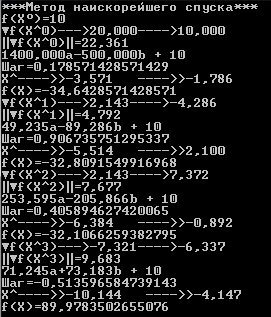
и т.д.

**Сходимость метода:**

Теорема 2. Пусть функция f(x) дифференцируема и ограничена снизу на а ее градиент удовлетворяет условию Липшица:

Тогда при произвольной начальной точке для метода наискорейшего градиентного спуска с постоянным шагом имеем ; Отметим, что теорема гарантирует сходимость последовательности к стационарной точке где Поэтому найденная точка нуждается в дополнительном исследовании для ее классификации.   
Оценка скорости сходимости получена только для сильно выпуклых функций, в этом случае последовательность сходится к точке минимума функции f(x) со скоростью геометрической прогрессии (линейная сходимость) где M и m – оценки наибольшего и наименьшего собственных значений матрицы Гессе H(x) функции f(x).

Результат работы программы:



Код программы:

class НаискорейшийСпуск

{

static double F(double x, double y)

{ return (2 \* (x \* x) + 2 \* (y \* y) + 2 \* x \* y + 20 \* x + 10 \* y + 10); }

static double Sistema\_X(double x, double y)

{ return 4 \* x + 2 \* y + 20; }

static double Sistema\_Y(double x, double y)

{ return 4 \* y + 2 \* x + 10; }

static double Kriterii(double x, double y)

{ return Math.Sqrt((x \* x) + y \* y); }

static double \_f\_a(double x, double y)

{ return 2.0 \* x \* x + 2.0 \* y \* y + 2.0 \* x \* y; }//1400a

static double \_f\_b(double x, double y)

{ return 20.0 \* x + 10.0 \* y; }//-500b

public static void N\_Spusk(double x0, double y0, double epsilon)

{//epsilon=0.1//X0=[0;0]

if (epsilon < 0)

{

Console.WriteLine("Значение epsilon меньше нуля!");

return;

}

double[] x\_X = new double[100], x\_Y = new double[100];

int k = 0;

double[] t = new double[100];

x\_X[k] = x0;

x\_Y[k] = y0;

double fX0 = F(x0, y0); //f(X^0)=10

Console.Write("f(X°)={0}\n", fX0);

double[] kritPoiska = new double[100];

double sistVerxn\_X, sistNizn\_Y;

//Шаг 1

step1:

sistVerxn\_X = Sistema\_X(x\_X[k], x\_Y[k]); //20//2.14//2.1428

sistNizn\_Y = Sistema\_Y(x\_X[k], x\_Y[k]); //10//-4.28//7.3723

Console.WriteLine("▼f(X^{2})--->{0:f3}---->{1:f3}", Sistema\_X(x\_X[k], x\_Y[k]), Sistema\_Y(x\_X[k], x\_Y[k]),k);

kritPoiska[k] = Kriterii(sistVerxn\_X, sistNizn\_Y); //22.3607//4.7916//7.67

Console.WriteLine("║▼f(X^{1})║={0:f3}", kritPoiska[k], k);

//Шаг 2

if (kritPoiska[k] < epsilon)

{

Console.WriteLine("\*x={0:f5}", kritPoiska[k]);

}

else

{

//step4:

//X^1=[-20;-10]//!!!ШАГ ЕЩЕ НЕОПРЕДЕЛЕН!!!поэтому 1.0

x\_X[k + 1] = x\_X[k] - 1.0\*sistVerxn\_X; //-20\*t0

x\_Y[k + 1] = x\_Y[k] - 1.0\*sistNizn\_Y; //-10\*t0

//нашли [-20t0;-10t0], ищем минимум, т.е. шаг

Console.WriteLine("{0:f3}a" + (\_f\_b(x\_X[k + 1], x\_Y[k + 1]) >= 0 ? "+{1:f3}b" : "{1:f3}b") + " + 10", \_f\_a(x\_X[k + 1], x\_Y[k + 1]), \_f\_b(x\_X[k + 1], x\_Y[k + 1]));//упращеное выражение f(x)

t[k] = -(\_f\_b(x\_X[k + 1], x\_Y[k + 1]))/(2\*\_f\_a(x\_X[k + 1], x\_Y[k + 1]));

//X^1=[x\_X[k + 1];x\_Y[k + 1]]

Console.WriteLine("Шаг={0}", t[k]);

//ШАГ 4

//X^1=[-20;-10]//!!!ШАГ ОПРЕДЕЛЕН!!!поэтому t[k]

x\_X[k+1] = x\_X[k] - t[k]\*sistVerxn\_X; //-3.75

Console.Write("X^---->>{0:f3}", x\_X[k + 1]);

x\_Y[k+1] = x\_Y[k] - t[k]\*sistNizn\_Y; //-1.78

Console.WriteLine(" ---->>{0:f3}", x\_Y[k + 1]);

fX0 = F(x\_X[k+1], x\_Y[k+1]); //f(X^0)=-34.64

Console.WriteLine("f(X)={0}", fX0);

k = k + 1;

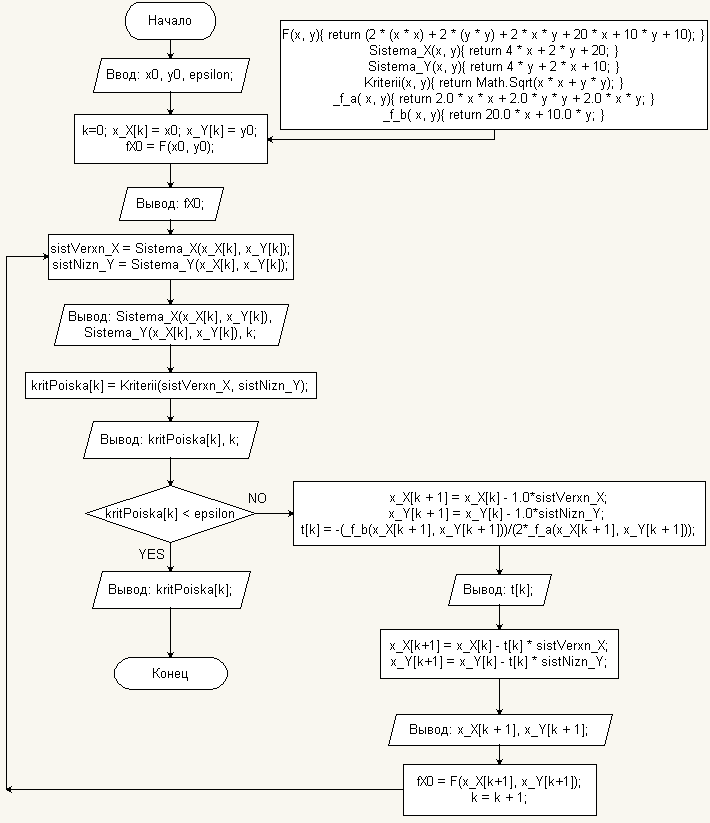
goto step1;

}

}//metod END

}

Блок схема программы:



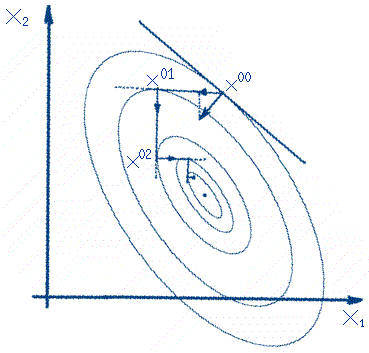
**Метод Гаусса-Зейделя (наискорейшего покоординатного спуска).**

Стратегия поиска.

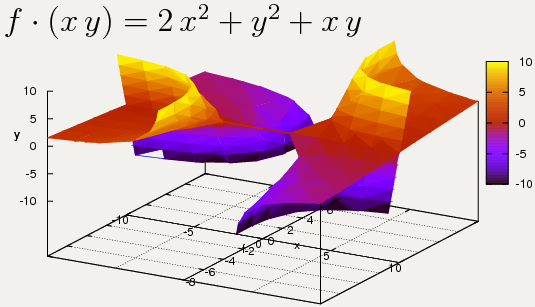
Суть метода заключается в эквивалентной замене общей многопараметрической задачи поиска экстремума критерия последовательностью однопараметрических задач поиска частных экстремумов.

Стратегия решения задачи состоит в построении последовательности точек таких, что Точки последовательности вычисляются по правилу: где j – номер цикла вычислений, j =0, 1, 2, ....; k – номер итерации внутри цикла, k=0, 1, 2, ....,n-1; задается пользователем. Величина шага выбирается из условия:

В данном методе при фиксированном j за одну итерацию с номером k изменяется только одна координата вектора с номером k+1, а за весь цикл с номером j, начиная с k=0 и кончая k=n-1, изменяются все координаты вектора После этого точке присваивается номер и она берется за начальную точку для вычислений в (j+1)-м цикле. Расчет заканчивается в точке для которой выполняется условие где заданное малое число. В случае двух переменных данный метод имеет следующую геометрическую интерпретацию.



Дано:



Решение:

Предположим

Градиент функции есть:

Берем j=0.

Шаг Положим k=0

Шаг Проверим выполнение условия

Шаг Вычислим

Шаг Проверим выполнение критерия остановки

, критерий не выполнен.

Шаг Найдем величину шага из условия

Имеем:

Подставив полученные значения в имеем Имеем

Так как то найденное значение шага обеспечивает минимум функции

Шаг Найдем

Шаг Предположим k=1 и переходим к Шагу 2.

Шаг Проверим выполнение условия

Шаг Вычислим

Шаг Проверим выполнение критерия остановки

Критерий не выполнен.

Шаг Найдем величину шага из условия

Имеем:

Подставив полученные значения в имеем Имеем

Так как то найденное значение шага обеспечивает минимум функции

Шаг Найдем

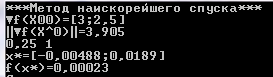
Шаг Предположим j=1 и переходим к Шагу 1 и т.д.

Искомая точка

Анализ точки:

Так как матрица Гессе является положительно определенной, то функция f(x) строго выпукла и, следовательно, в стационарной точке достигает глобальный минимум .

Результат работы программы:



Код программы:

class НаискорейшегоПокоординатногоСпуска

{

static double F(double x, double y)

{ return 2 \* x \* x + y \* y + x \* y; }

static double Grad\_Числит(double x, double y)

{ return 4 \* x + y; }

static double Grad\_Знаменат(double x, double y)

{ return x + 2 \* y; }

static double Kriterii(double x, double y)

{ return Math.Sqrt((x \* x) + (y \* y)); }

public static void NPokSpusk(double x0, double y0, double epsilon)

{//Задать [x;y], e>0

if(epsilon<0)return 0;

double[] x\_X = new double[50];

double[] x\_Y = new double[50];

x\_X[0] = x0;

x\_Y[0] = y0;

double t0;

double[] kritPoiska = new double[50];

//Найти градиент функции F в [x0;y0] Grad\_

Console.WriteLine("▼f(X00)=[{0};{1}]", Grad\_Числит(x\_X[0], x\_Y[0]), Grad\_Знаменат(x\_X[0], x\_Y[0]));

int j = 0;

//Шаг 1

step1:

int k = 0;

if (k<=j-1)//n-1

{

goto step3;

}

if (k==j)

{

j = j + 1;

goto step1;

}

step3:

double grad\_fX = Grad\_Числит(x\_X[0], x\_Y[0]);//3

double grad\_fY = Grad\_Знаменат(x\_X[0], x\_Y[0]);//2.5

Console.WriteLine("▼f(X00)=[{0};{1}]", grad\_fX, grad\_fY);

//Проверим выполнение критерия окончания

kritPoiska[k] = Kriterii(grad\_fX, grad\_fY);//3.9

Console.WriteLine("║▼f(X^{1})║={0:f3}", kritPoiska[k], k);

if (kritPoiska[k] < epsilon)

{ Console.WriteLine("\*x={0:f5}", kritPoiska[k]); }

else

{

//Найдем величину шага

double e1 = 1.0, e2 = 0.0;//единичный вектор//const

t0 = t0+k;

if (t0<0)

{

t0 = t0 + x\_X[k];

Grad\_Числит(x\_X[k], x\_Y[k]);

t0 = t0 - x\_Y[k];

Grad\_Знаменат(x\_X[0], x\_Y[0]);

}

double dx= x\_X[k] - t0\*e1;

double dy = x\_Y[k] - t0\*e2;

Console.WriteLine(dx+" "+dy);

k = k + 1;

goto step3;

}

}

}

Блок схема программы:

