Воронежский институт высоких технологий

Факультет заочного и послевузовского обучения

Кафедра математики и информатики

Курсовая работа

По дисциплине: Методы оптимизации

На тему: Минимизация функций нескольких переменных. Метод спуска

Воронеж 2010

**Введение**

Метод оптимизации как раздел математики существует достаточно давно. Оптимизация - это выбор, т.е. то, чем постоянно приходится заниматься в повседневной жизни. Термином "оптимизация" в литературе обозначают процесс или последовательность операций, позволяющих получить уточненное решение. Хотя конечной целью оптимизации является отыскание наилучшего или "оптимального" решения, обычно приходится довольствоваться улучшением известных решений, а не доведением их до совершенства. Поэтому под оптимизацией понимают скорее стремление к совершенству, которое, возможно, и не будет достигнуто.

Необходимость принятия наилучших решений так же стара, как само человечество. Испокон веку люди, приступая к осуществлению своих мероприятий, раздумывали над их возможными последствиями и принимали решения, выбирая тем или другим образом зависящие от них параметры - способы организации мероприятий. Но до поры, до времени решения могли приниматься без специального математического анализа, просто на основе опыта и здравого смысла.

Возьмем пример: человек вышел утром из дому, чтобы ехать на работу. По ходу дела ему приходится принять целый ряд решений: брать ли с собой зонтик? В каком месте перейти улицу? Каким видом транспорта воспользоваться? И так далее. Разумеется, все эти решения человек принимает без специальных расчетов, просто опираясь на имеющийся у него опыт и на здравый смысл. Для обоснования таких решений никакая наука не нужна, да вряд ли понадобится и в дальнейшем.

Однако возьмем другой пример. Допусти, организуется работа городского транспорта. В нашем распоряжении имеется какое-то количество транспортных средств. Необходимо принять ряд решений, например: какое количество и каких транспортных средств направить по тому или другому маршруту? Как изменять частоту следования машин в зависимости от времени суток? Где поместить остановки? И так далее.

Эти решения являются гораздо более ответственными, чем решения предыдущего примера. В силу сложности явления последствия каждого из них не столь ясны; для того, чтобы представить себе эти последствия, нужно провести расчеты. А главное, от этих решений гораздо больше зависит. В первом примере неправильный выбор решения затронет интересы одного человека; во втором - может отразиться на деловой жизни целого города.

Наиболее сложно обстоит дело с принятием решений, когда речь идет о мероприятиях, опыта, в проведении которых еще не существует и, следовательно, здравому смыслу не на что опереться, а интуиция может обмануть. Пусть, например, составляется перспективный план развития вооружения на несколько лет вперед. Образцы вооружения, о которых может идти речь, еще не существуют, никакого опыта их применения нет. При планировании приходится опираться на большое количество данных, относящихся не столько к прошлому опыту, сколько к предвидимому будущему. Выбранное решение должно по возможности гарантировать нас от ошибок, связанных с неточным прогнозированием, и быть достаточно эффективным для широкого круга условий. Для обоснования такого решения приводится в действие сложная система математических расчетов.

Вообще, чем сложнее организуемое мероприятие, чем больше вкладывается в него материальных средств, чем шире спектр его возможных последствий, тем менее допустимы так называемые "волевые" решения, не опирающиеся на научный расчет, и тем большее значение получает совокупность научных методов, позволяющих заранее оценить последствия каждого решения, заранее отбросить недопустимые варианты и рекомендовать те, которые представляются наиболее удачными.

**Методы спуска**

## *Общая схема*

Все методы спуска решения задачи безусловной минимизации различаются либо выбором направления спуска, либо способом движения вдоль направления спуска. Это позволяет написать общую схему методов спуска.

Решается задача минимизации функции ϕ(x) на всём пространстве En. Методы спуска состоят в следующей процедуре построения последовательности {xk}. В качестве начального приближения выбирается любая точка x0∈En. Последовательные приближения x1, x2, … строятся по следующей схеме:

1. в точке xk выбирают направление спуска - Sk;
2. находят (k+1)-е приближение по формуле xk+1=xk-hkSk.

Направление Sk выбирают таким образом, чтобы обеспечить неравенство f(xk+1)<f(xk) по крайней мере для малых значений величины hk. На вопрос, какому из способов выбора направления спуска следует отдать предпочтение при решении конкретной задачи, однозначного ответа нет.

Число hk определяет расстояние от точки xk до точки хk+1. Это число называется длиной шага или просто шагом. Основная задача при выборе величины hk - это обеспечить выполнение неравенства ϕ(xk+1)<ϕ(xk).

Величина шага сильно влияет на эффективность метода. Большей эффективностью обладает вариант метода, когда шаг по каждой переменной определяется направляющими косинусами градиента(в градиентных методах).

xk+1=xk-hk cos

где - cos=

В этом случаи величина рабочего шага не зависит от величины модуля градиента, и ею легче управлять изменением h . В районе оптимума может возникать значительное «рыскание», поэтому используют различные алгоритмы коррекции h.

Наибольшее распространение получили следующие алгоритмы:

1. (без коррекции);

2.  если ;  если 

3. , если  ; , если; ,если ,

где –угол между градиентами на предыдущем и текущем шаге;

и – заданные пороговые значения выбираются субъективно

(например,  ).

Вдали от оптимума направление градиента меняется мало, поэтому шаг можно увеличить (второе выражение), вблизи от оптимума направление резко меняется (угол между градиентами R(x) большой), поэтому h сокращается (третье выражение).

***Метод покоординатного спуска***

Пусть нужно найти наименьшее значение целевой функции u=f(M)=f(x, x, . . . ,xn). Здесь через М обозначена точка n-мерного пространства с координатами x, x, . . . ,xn: M=(x, x, . . . ,xn). Выберем какую-нибудь начальную точку М=(x, x, . . . ,xn0) и рассмотрим функцию f при фиксированных значениях всех переменных, кроме первой: f(x, x,x, . . . ,xn0 ). Тогда она превратится в функцию одной переменной x . Изменяя эту переменную, будем двигаться от начальной точки x=x в сторону убывания функции, пока не дойдем до ее минимума при x=x, после которого она начинает возрастать. Точку с координатами ( x, x,x, . . . ,xn0) обозначим через М, при этом f(M0)  f(M).

Фиксируем теперь переменные: x=x, x= x, . . . ,xn=xn0 и рассмотрим функцию f как функцию одной переменной x: f(x, x, x . . . ,xn0). Изменяя x , будем опять двигаться от начального значения x2=x20 в сторону убывания функции, пока не дойдем до минимума при x2=x21 .Точку с координатами {x, x, x . . . xn0} обозначим через М, при этом f(M1) f(M).

Проведем такую же минимизацию целевой функции по переменным x, x, . . . ,xn. Дойдя до переменной xn, снова вернемся к x и продолжим процесс. Эта процедура вполне оправдывает название метода. С ее помощью мы построим последовательность точек МММ. . . , которой соответствует монотонная последовательность значений функции f(M0) f (M)f(M)Обрывая ее на некотором шаге k можно приближенно принять значение функции f(Mk) за ее наименьшее значение в рассматриваемой области.

Проведем такую же минимизацию целевой функции по переменным x, x, . . . ,xn. Дойдя до переменной xn, снова вернемся к x и продолжим процесс. Эта процедура вполне оправдывает название метода. С ее помощью мы построим последовательность точек М,М,М, . . . , которой соответствует монотонная последовательность значений функции

f(M0)f(M)f(M) Обрывая ее на некотором шаге k можно приближенно принять значение функции f(Mk) за ее наименьшее значение в рассматриваемой области. Отметим , что данный метод сводит задачу поиска наименьшего значения функции нескольких переменных к многократному решению одномерных задач оптимизации. Если целевая функция f(x, x, ... ,xn) задана явной формулой и является дифференцируемой, то мы можем вычислить ее частные производные и использовать их для определения направления убывания функции по каждой переменной и поиска соответствующих одномерных минимумов. В противном случае, когда явной формулы для целевой функции нет, одномерные задачи следует решать с помощью одномерных методов

На рис.изображены линии уровня некоторой функции двух переменных u= f (х, у). Вдоль этих линий функция сохраняет постоянные значения, равные 1, 3, 5, 7, 9. Показана траектория поиска ее наименьшего значения, которое достигается в точке О, с помощью метода покоординатного спуска. При этом нужно ясно понимать, что рисунок служит только для иллюстрации метода.

Пусть требуется решить задачу (2):

f(x)  min, х Rn. (2)

В двумерном пространстве R2. Решение задачи (2) методом покоординатного спуска, иначе называемого методом Гаусса - Зейделя, производят по следующей общей схеме.

Выбирают произвольно начальную точку х(0) из области определения функции f(х). Приближения х(k) определяются соотношениями

(3): x(k+1)=x(k)+t(k)S(k) (k=0,1,2, ...),

где вектор направления спуска s(k)- это единичный вектор, совпадающий с каким-либо координатным направлением (например, если S(k) параллелен х1, то S(k)= {1,0,0,...,0}, если он параллелен x2, то S(k)={0, 1, 0, . . . ,0} и т.д.) ; величина t(k) является решением задачи одномерной минимизации: f(x(k)+ts(k))  min, t R1, (k=0,1,2, ...), и может определяться, в частности, методом сканирования. Детальная реализация общей схемы в двумерном случае R2 дает траекторий приближения к точке х\* методом покоординатного спуска, состоящую из звеньев ломаной, соединяющих точки х(k), x1(k) x(k+1) (k=0, 1, 2,) . При k=0, исходя из начальной точки х(0)=(x1(0),x2(0)), находят точку х(0)= (x1(0),x2(0)), минимума функции одной переменной f(x1,x2(0)); при этом f(x(0))f(x(0)).Затем находят точку минимума x(1) функции f (x1(0),x2) по второй координате. Далее делают следующий шаг вычислений при k=1. Полагают, что исходной точкой расчета является х(1). Фиксируя вторую координату точки х(1), находят точку минимума х(1)= (x1(1),x2(1)), функции f(x1,x2(1)) одной переменной x(1); при этом f(x(1))f(x(1))f(x(0)). Точку х(2) получают, минимизируя целевую функцию f(x1(1),x2), вновь по коорданате х2, фиксируя координату x1(1) ,точки x(1) , и т.д. Условием прекращения вычислительной процедуры при достижении заданной точности  может служить неравенство x(k+1) - x(k) <

Блок-схема поиска минимума функции двух переменных методом покоординатного спуска.

дискретный оптимизация спуск функция



***Метод градиентного спуска***

Рассмотрим функцию f, считая для определенности, что она зависит от трех переменных x,y,z. Вычислим ее частные производные дf/дх, дf/ду, дf/дz и образуем с их помощью вектор, который называют градиентом функции:

grad f(x, у, z) = дf (х, у,z) /дх\*i+дf( x, у, z)/ду\*j+дf(x, y,z)/дг\*k.

Здесь i, j, k - единичные векторы, параллельные координатным осям. Частные производные характеризуют изменение функции f по каждой независимой переменной в отдельности. Образованный с их помощью вектор градиента дает общее представление о поведении функции в окрестности точки (х, у,z). Направление этого вектора является направлением наиболее быстрого возрастания функции в данной точке. Противоположное ему направление, которое часто называют антиградиентным, представляет собой направление наиболее быстрого убывания функции. Модуль градиента grad (х, у,z)дf/дх (х, у,z))2 +(дf/ду( x, у, z))2+(дf/дг(x, y,z))2. определяет скорость возрастания и убывания функции в направлении градиента и антиградиента. Для всех остальных направлений скорость изменения функции в точке (х, у, z) меньше модуля градиента. При переходе от одной точки к другой как направление градиента, так и его модуль, вообще говоря, меняются. Понятие градиента естественным образом переносится на функции любого числа переменных.

Перейдем к описанию метода градиентного спуска. Основная его идея состоит в том, чтобы двигаться к минимуму в направлении наиболее быстрого убывания функции, которое определяется антиградиентом. Эта идея реализуется следующим образом.

Выберем каким-либо способом начальную точку, вычислим в ней градиент рассматриваемой функции и сделаем небольшой шаг в обратном, антиградиентном направлении. В результате мы придем в точку, в которой значение функции будет меньше первоначального. В новой точке повторим процедуру: снова вычислим градиент функции и сделаем шаг в обратном направлении. Продолжая этот процесс, мы будем двигаться в сторону убывания функции. Специальный выбор направления движения на каждом шаге позволяет надеяться на то, что в данном случае приближение к наименьшему значению функции будет более быстрым, чем в методе покоординатного спуска.

Метод градиентного спуска требует вычисления градиента целевой функции на каждом шаге. Если она задана аналитически, то это, как правило, не проблема: для частных производных, определяющих градиент, можно получить явные формулы. В противном случае частные производные в нужных точках приходится вычислять приближенно.

Отметим, что при таких расчетах gi ,нельзя брать слишком малым, а значения функции нужно вычислять с достаточно высокой степенью точности, иначе при вычислении разности

f(x1, ...,xi+ gi, ..., xn) - f(x1, ..., xi, ..., xn)

f(x1, ...,xi+ gi, ..., xn) - f(x1, ..., xi- gi,..., xn)

будет допущена большая ошибка.

Первый алгоритм требует меньших затрат по сравнению со вторым (обычно затраты выражаются количеством вычислений критерия оптимальности), но позволяет получить решение менее точно, чем второй, эта погрешность зависит от величины пробного шага

***Метод наискорейшего спуска***

Суть метода наискорейшего спуска состоит в следующем. Как и прежде, в начальной точке определяется антиградиент минимизируемой функции. Однако теперь в направлении антиградиента делается ни один шаг, а движутся в данном направлении до тех пор, пока целевая функция убывает, достигает в некоторой точке минимума. В этой точке опять определяют антиградиент и ищут новую точку минимума целевой функции и так далее. В данном методе спуск имеет более целеустремлённый характер, производится более крупными шагами и градиент функции вычисляется в меньшем числе точек.

Описание программы:

Программа предназначена для нахождения точек минимума функций нескольких переменных – другими словами для минимизации этих функций.

В программе реализован один из методов спуска – Градиентный метод спуска с выбором шага. Начальный шаг задается.

Изменение шага осуществляется по схеме

 если ;  если 

Вычисление градиента происходит по методу с парными пробами, это улучшает поиск за счёт более точного вычисления градиента.

Метод наискорейшего спуска по сравнению с обычным градиентным методом дает некоторое ускорение , метод хорошо "работает" при минимизации гладких функций и если начальное приближение выбрано достаточно далеко от оптимума. Если же очередная точка окажется в окрестности оптимума, то уменьшение целевой функции будет очень медленным. Это происходит из-за того, что для получения оптимума с высокой точностью необходимо выполнить большое число мелких шагов.

Метод наискорейшего спуска хотя не дает особенного ускорения сходимости он свободен от параметров и на практике может дать некоторый выигрыш, особенно на начальных итерациях.

В связи с этим в программе был реализован более точный метод градиентного спуска.

В качестве условия окончания поиска задаётся требуемая малость модуля градиента функции, т.е. должно выполнятся условие



(В области оптимума градиент равен 0, но достичь этого значения практически не возможно, поэтому задаётся требуемая малость близкая к 0).

Так же в программе можно задавать номер итерации выхода из цикла,

Другими словами при достижении какого количества точек прерывать цикл, если он не прервется сам раньше.

###### Исследование функции

###### U=1\*x1^3+2\*x2^2-3\*x1-4\*x2

###### (изменением шага).

h=0,1; x1 =-0,5; x2=-1 ; x1нач=-2, x1кон=2, x2нач=-2, x2кон=2

|  |  |  |  |
| --- | --- | --- | --- |
| № | x1 | x2 | R |
| 0 | -0,5 | -1 | 7,375 |
| 1 | -0,2750 | -0,1999 | 1,6842 |
| 2 | 0,0023 | 0,2800 | -0,9701 |
| 3 | 0,3023 | 0,5680 | -2,5059 |
| 4 | 0,5749 | 0,7408 | -3,4002 |
| 5 | 0,7757 | 0,8445 | -3,8120 |
| 6 | 0,8952 | 0,9067 | -3,9508 |
| 7 | 0,9548 | 0,9440 | -3,9877 |
| 8 | 0,9813 | 0,9664 | -3,9967 |
| 9 | 0,9924 | 0,9798 | -3,9990 |
| 10 | 0,9969 | 0,9879 | -3,9997 |
| 11 | 0,9988 | 0,9927 | -3,9999 |
| 12 | 0,9995 | 0,9956 | -4,0000 |
| 13 | 0,9998 | 0,9974 | -4,0000 |
| 14 | 1,0000 | 0,9984 | -4,0000 |

h=0,2; x1 =-0,5; x2=-1 ; x1нач=-2, x1кон=2, x2нач=-2, x2кон=2

|  |  |  |  |
| --- | --- | --- | --- |
| № | x1 | x2 | R |
| 0 | -0,5 | -1 | 7,375 |
| 1 | 0,0500 | 0,6000 | -1,5301 |
| 2 | 0,5485 | 0,9200 | -3,4676 |
| 3 | 0,9680 | 0,9840 | -3,9964 |
| 4 | 1,0058 | 0,9968 | -3,9999 |
| 5 | 0,9988 | 0,9994 | -4,0000 |

h=0,3; x1 =-0,5; x2=-1 ; x1нач=-2, x1кон=2, x2нач=-2, x2кон=2

|  |  |  |  |
| --- | --- | --- | --- |
| № | x1 | x2 | R |
| 0 | -0,5 | -1 | 7,375 |
| 1 | 0,1750 | 1,4 | -2,1996 |
| 2 | 1,0473 | 0,9200 | -3,9804 |
| 3 | 0,9600 | 1,016 | -3,9948 |
| 4 | 1,0305 | 0,9968 | -3,9972 |
| 5 | 0,9747 | 0,0006 | -3,9981 |
| 6 | 1,0196 | 0,9999 | -3,9988 |
| 7 | 0,9839 | 1,0000 | -3,9992 |
| 8 | 1,0126 | 1,0000 | -3,9995 |
| 9 | 0,9898 | 1,0000 | -3,9997 |
| 10 | 1,0081 | 1,0000 | -39998 |
| 11 | 0,9935 | 1,0000 | -3,9999 |
| 12 | 1,0052 | 1,0000 | -3,9999 |
| 13 | 0,9958 | 1.0000 | -3,9999 |
| 14 | 1,0033 | 1,0000 | -4,0000 |
| 15 | 0,9973 | 1,0000 | -4,0000 |
| 16 | 1,0021 | 1,0000 | -4,0000 |
| 17 | 0,9983 | 1,0000 | -4,0000 |
| 18 | 1,0013 | 1,0000 | -4,0000 |

h=1; x1 =-0,5; x2=-1 ; x1нач=-2, x1кон=2, x2нач=-2, x2кон=2

|  |  |  |  |
| --- | --- | --- | --- |
| № | x1 | x2 | R |
| 0 | -0,5 | -1 | 7,375 |
| 1 | 0,6250 | 3 | 4,3692 |
| 2 | 1,5391 | -1,0000 | 5,0283 |
| 3 | 0,5125 | 1 | -3,4029 |
| 4 | 1,0655 | 1 | -3,9869 |
| 5 | 0,9640 | 1 | -3,9961 |
| 6 | 1,0170 | 1 | -3,9991 |
| 7 | 0,9913 | 1,0000 | -3,9998 |
| 8 | 1,0043 | 1 | -3,9999 |
| 9 | 0,9978 | 1 | -4,0000 |
| 10 | 1,0011 | 1 | -4,0000 |

# Заключение

Практика порождает все новые и новые задачи оптимизации, причем их сложность растет. Требуются новые математические модели и методы, которые учитывают наличие многих критериев, проводят глобальный поиск оптимума. Другими словами, жизнь заставляет развивать математический аппарат оптимизации.

Реальные прикладные задачи дискретной оптимизации очень сложны. Современные методы оптимизации далеко не всегда справляются с решением реальных задач без помощи человека. Нет, пока такой теории, которая учла бы любые особенности функций, описывающих постановку задачи. Следует отдавать предпочтение таким методам, которыми проще управлять в процессе решения задачи.

**Список используемой литературы**

1. Корнеенко В. П. Методы оптимизации: Учебник / В. П. Корнеенко. – М.: Высш. шк., 2007.
2. Пантелеев А. В. Методы оптимизации в примерах и задачах: Учеб. пособие / А. В. Пантелеев, Т. А. Летова. – М.: Высш. шк., 2005.
3. Батищев Д. И. Оптимизация в САПР: Учебник / Д. И. Батищев, Я. Е. Львович, В. Н. Фролов. – Воронеж: Изд-во ВГУ, 1997.
4. Банди Б. Методы оптимизации. Вводный курс / Б. Банди. – М.: Радио и связь, 1988.
5. Реклейтис Г. Оптимизация в технике: в 2 кн. / Г. Реклейтис, А. Рейвиндран. – М.: Мир, 1986.

**Приложение**

# Листинг программы:

#include <vcl.h>

#pragma hdrstop

#include "Math.hpp"

#include "math.h"

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

int ii=0,n=0,s=0;

AnsiString Formula[3]={"U=A\*x1^3+B\*x2^2-C\*x1-D\*x2","U=x1^2+x1\*x2+x2^2","U=X1^2+X2^2"};

int KolPer[3]={2,2,2};// массив в котором хранится кол-во перемен. для каждой ф-ии

bool DD=true,Diapozon=true; // если true то точка входит в диапозон иначе нет

double PeremenN[5]={0};//double \*PeremenN =new double[n]; //нул.приб

double InterN[5]={0};//double \*InterN =new double[n]; //нач

double InterK[5]={0};//double \*InterK =new double[n]; //кон

double Param[4]={0}; //параметры

double T1[5]={0};//double \*T1 =new double[n]; //tochka i -я

double T2[5]={0};//double \*T2 =new double[n]; //tochka i+1 -я

double TempT[5]={0};//double \*TempT =new double[n]; // временная tochka i+1 -я

double BB[5]={0};//double \*BB= new double [n]; // BB - массив с измененой i-ой точкой X[i]+g

double B[5]={0};//double \*B= new double [n]; //B - массив с измененой i-ой точкой X[i]-g

int g=0;

double ModG =0; //модуль градиента

int ss=0,ind=0;

double \*\*Tochki; // указатель на массив с точками приближения

//---------------------------------------------------------------------------

double TForm1::F1( double T[]) //Formula1 U=A\*x1^3+B\*x2^2-C\*x1-D\*x2

{ double U = 0;

U=IntPower(T[0],3)+2\*IntPower(T[1],2)-3\*T[0]-4\*T[1];

return U; }

//---------------------------------------------------------------------------

double TForm1::F2( double T[]) //Formula2 U=x1^2+x1\*x2+x2^2

{ double U = 0;

U = IntPower(T[0],2)+T[0]\*T[1]+IntPower(T[1],2);

return U; }

//---------------------------------------------------------------------------

double TForm1::F3( double T[]) //Formula3 U=X1^2+X2^2

{ double U = 0;

U =T[0]\*T[0]+T[1]\*T[1]+1;

return U; }

//---------------------------------------------------------------------------

void TForm1::Tochka(double shag) // функция считает координаты следующей точки

{ // n - количество переменных

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

{

TempT[i]=T2[i]-shag\*Gr(i);

//точка X[j+1]=точка X[j]- h козф шага \*градиет grad R(X[j])

}

}

//---------------------------------------------------------------------------

double TForm1::Gr( int i) //gradient i-номер переменной

{

double dR=0; // dR - градиент по i

for (int j=0;j<n;j++) //BB,B==T1;

{

BB[j]=T2[j];

B[j]=T2[j];

}

BB[i]=T2[i]+ Param[1] ; // добавляем и отнимаем пробный шаг

B[i]=T2[i]- Param[1] ; // к i-ой переменной

switch (UD->Position) {

case 0: dR = (F1(BB)- F1(B))/(2\*Param[1]) ; break;

case 1: dR = (F2(BB)-F2(B))/(2\*Param[1]); break;

case 2: dR = (F3(BB)-F3(B))/(2\*Param[1]); break;

}

return dR;

}

//--------------------------------------------------------------------------

void TForm1::Min()

{ // массив в котором

//double Tochki[1][5]; //хранится первое приближение

//double \*\*Tochki; //создаем массив Temp[ss][n]

Tochki = new double\*[100];

for (int j = 0; j < 100; j++)

Tochki[j] = new double[3];

bool Minimum=false,Pogresh=false,shag=false;

double sh=Param[0];

for (int j=0;j<n;j++) //T1,T2,TempT=PeremenN;

{

T1[j]=PeremenN[j];

T2[j]=PeremenN[j];

TempT[j]=PeremenN[j];

Tochki[0][j]=PeremenN[j];

}

while ( Minimum == false ) // после выхода из цикла

{ // минимум в точке T2

shag=false;

// начало блока 2

while (shag == false)

{

double R=0;

Tochka(sh);

switch (UD->Position) {

case 0: R=F1(TempT)-F1(T1) ; break;

case 1: R=F2(TempT)-F2(T1); break;

case 2: R=F3(TempT)-F3(T1); break; }

if (R > 0) // шаг большой то

{ // уменьшаем его в 2 раза

sh= sh/2;

}

else

{ shag =true; }

}

// конец блока 2

// Проверяем входит ли точка в указанный диапозон

// если нет то считаем предыдущую точку минимальной

if (DD==true )

{

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

{

if ( InterN[i] > TempT[i])

{

Diapozon=false;

Minimum = true;

}

if (InterK[i] < TempT[i])

{

Diapozon=false;

Minimum = true;

}

}

}

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

{

T1[j]=T2[j]; //T1=T2

T2[j]=TempT[j]; //T2=TempT

}

// начало блока 3

ModG=0; //- модуль градиента

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

{

ModG+= Gr(i)\*Gr(i);

}

ModG=sqrt(ModG);

if ( ModG < Param[2]) // /gradient/ < e где e-погрешность

{ Minimum=true; } // /gradient/ - модуль градиента

// конец блока 3

ss++;

if (Param[3] != -1 )

if (ss == Param[3])

Minimum=true;

// начало блока 4

if ( ss > 99 )

{ MessageDlg("Предел превышен ...точек более 100 ..измените шаг",mtWarning,

TMsgDlgButtons() << mbOK , 0);break;}

if(Diapozon==true)

{

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

Tochki[ss][j] = T2[j];

}

}

for (int j = 0; j <= ss; j++)

{

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

TempT[i]= Tochki[j][i];

switch (UD->Position) {

case 0: Tochki[j][2] = F1(TempT) ; break;

case 1: Tochki[j][2] = F2(TempT); break;

case 2: Tochki[j][2] = F3(TempT); break; }

}

//

/\* double \*\*Temp; //создаем массив Temp[ss][n]

Temp = new double\*[ss];

for (int j = 0; j < ss; j++)

Temp[j] = new double[n];

//

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

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

Temp[i][j]=Tochki[i][j];

//

//for (int i = 0; i < ss; i++) //удаляем массив Tochki[ss][n]

//delete[] Tochki[i];

//delete Tochki;

//

int mm=ss+1;

double \*\*Tochki; //создаем массив Tochki[ss+1][n]

Tochki = new double\*[mm];

for (int j = 0; j < mm; j++)

Tochki[j] = new double[n];

//

for (int i = 0; i < mm-1; i++)

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

Tochki[i][j] = Temp[i][j];

//

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

Tochki[ss][j] = T2[j];

//

//for (int i = 0; i < ss; i++) //удаляем массив Temp[ss][n]

//delete[] Temp[i];

//delete [] Temp;

}

// конец блока 4 \*/

}

//--------------------------------------------------------------------------

void \_\_fastcall TForm1::UDClick(TObject \*Sender, TUDBtnType Button)

{

Edit2->Text=Formula[UD->Position];

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::StartClick(TObject \*Sender)

{

Panel1->Visible=false;

Edit2->Text=Formula[UD->Position];

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Sh1NextClick(TObject \*Sender)

{

ii++;

PageControl1->ActivePageIndex=ii;

g=1;

switch (UD->Position) {

case 0: Kol->Caption=KolPer[0];break;

case 1: Kol->Caption=KolPer[1];break;

case 2: Kol->Caption=KolPer[2];break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Sh2NextClick(TObject \*Sender)

{

ii++;

PageControl1->ActivePageIndex=ii;

g=3;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Sh3BackClick(TObject \*Sender)

{

ii--;

PageControl1->ActivePageIndex=ii;

Panel5->Visible=false;

Sh2Next->Visible=false;

g=2;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Sh2BackClick(TObject \*Sender)

{

if (g ==1 )

{

ii--;

PageControl1->ActivePageIndex=ii;

Panel2->Visible=true;

Panel3->Visible=false;

Panel4->Visible=false;

Panel5->Visible=false;

}

if (g == 2)

{

Panel3->Visible=true;

Panel4->Visible=false;

Panel5->Visible=false;

n=KolPer[UD->Position];

Per->Caption="X1 =";

s=0;

g=1;

}

if (g == 3)

{

Panel5->Visible=false;

Sh2Next->Visible=false;

g=2;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Panel3->Visible=true;

n=KolPer[UD->Position];

Per->Caption="X1 =";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

PeremenN[s]=StrToFloat(Edit4->Text); //нул.приб

InterN[s]=StrToFloat(Edit3->Text); //нач

InterK[s]=StrToFloat(Edit5->Text); //кон

s++;

Per->Caption="X"+ IntToStr(s+1)+"=";

g=2;

if (s == n)

{Panel4->Visible=true;g=2;}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

Param[0]=StrToFloat(Edit6->Text); //коэ.шага

Param[1]=StrToFloat(Edit7->Text); // проб.шаг

Param[2]=StrToFloat(Edit8->Text); // погр.

if(CB1->Checked == true )

{Param[3]=StrToFloat(NT->Text); }

else

{Param[3]=-1;}

Sh2Next->Visible=true;

Panel5->Visible=true;

g=3;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::PuskClick(TObject \*Sender)

{

ss=0; //количество точек которых получилось

Diapozon=true;

Min();

if (Diapozon==false)

ss=ss-1;

Sh3Back->Visible=true;

Panel6->Visible=true;

Series1->Clear();

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

{

Series1->AddXY(i,Tochki[i][2],"",clBlue);

Nomer->Items->Add(i);

}

Nomer->Items->Add(ss);

//Nomer->Items->St

//ListT->Items->Add(123);

//if ( Diapozon=true )

//{ Itog->Caption="Точка минимума в указанном диапозоне "; }

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CB1Click(TObject \*Sender)

{

if(CB1->Checked == true )

NT->Visible=true;

if(CB1->Checked == false )

NT->Visible=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button8Click(TObject \*Sender)

{

Panel6->Visible=false;

ListT->Items->Clear();

Nomer->Items->Clear();

Nomer->ItemIndex=-1;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::NomerChange(TObject \*Sender)

{

int ind=Nomer->ItemIndex;

ListT->Items->Clear();

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

ListT->Items->Add(Tochki[ind][i]);

ListT->Items->Add(Tochki[ind][2]);

if (ind == ss)

if( Diapozon==true)

{ ListT->Items->Add(" Минимум");}

else

{

ListT->Items->Add(" Минимум");

ListT->Items->Add("Следующая точка в");

ListT->Items->Add("диапозон не входит");

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Pr1Click(TObject \*Sender)

{

if(Pr1->Checked == true )

DD=true;

if(Pr1->Checked == false )

{

DD=false;

MessageDlg("Вы отключили проверку диапозона точки,"

"убедитесь в этом",mtWarning,

TMsgDlgButtons() << mbOK , 0);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CB2Click(TObject \*Sender)

{

if(CB2->Checked == true )

{

Panel7->Visible=true;

Series1->Active=false;

Series2->Clear();

Perem->Text="Xi";

Perem->Items->Clear();

CB3->ItemIndex=-1;

CB3->Items->Clear();

CB4->ItemIndex=-1;

CB4->Items->Clear();

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

Perem->Items->Add(i+1);

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

{

CB3->Items->Add(i);

}

}

if(CB2->Checked == false )

{

Series2->Clear();

Series2->Active=false;

Series1->Active=true;

Panel7->Visible=false;

CB4->Enabled=false;

CB3->Enabled=false;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::PeremChange(TObject \*Sender)

{

int ind=Nomer->ItemIndex;

CB3->Enabled=true;

CB3->ItemIndex=0;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CB3Change(TObject \*Sender)

{

CB4->Items->Clear();

CB4->ItemIndex=-1;

int in=CB3->ItemIndex;

CB4->Enabled=true;

for(int i = in; i <=ss ; i++)

CB4->Items->Add(i);

CB4->ItemIndex=0;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::CB4Change(TObject \*Sender)

{

Bild->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::BildClick(TObject \*Sender)

{

Series2->Clear();

ListP->Items->Clear();

int nh=CB3->ItemIndex;

int nk=CB4->ItemIndex;

Series2->Active=true;

for(int i = nh; i <=nk+nh; i++)

{

Series2->AddXY(i,Tochki[i][ind],"",clBlue);

ListP->Items->Add(Tochki[i][ind]);

}

Bild->Visible=false;

CB4->Enabled=false;

CB4->Items->Clear();

CB4->ItemIndex=-1;

}

//---------------------------------------------------------------------------