МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

САМАРКАНДСИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени Алишера Навои

МЕХАНИКО – МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ»

На правах рукописи

УДК

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

для получения академической степени магистра

по специальности

«5А480101 –Вычислительная математика»

КОСИМОВА ДИЛМУРОДА НАСИМОВИЧА

на тему:

СРАВНЕНИЕ МЕТОДОВ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Работа рассмотрена и допущено к защите:

Декан факультета:	**проф. Солеев А.**
Заведующий кафедрой:	**доц. Абдирашидов А.**
Научный руководитель:	**доц. Маматов Ш.С.**

Самарканд – 2012

РЕЦЕНЗИЯ

на магистерскую диссертацию Косимова Д. на тему

«Сравнение методов численного решения задач оптимизации»

Поиск экстремума функции одной переменной имеет самостоятельный интерес, так как является составной частью многих методов многомерной оптимизации. Существует много методов одномерной безусловной оптимизации так, например, метод пассивного поиска, метод блочного поиска, метод деления интервала пополам, метод дихотомии, метод золотого сечения, метод чисел Фибоначчи и т.д.

В данной магистерской диссертационной работе для конкретной целевой функции применяется 5 методов оптимизации и сравнивается полученные результаты. Выбирается самый эффективный метод для поиска экстремума. Далее разработан пакет прикладных программ для численного решения задач оптимизации.

Результаты работы являются новыми, и представляет научный интерес, могут, применяется в лабораторных и практических занятиях по численным методам оптимизации.

Диссертационная работа удовлетворяет всем требованиям, предъявляемым к магистерским диссертационным работам и заслуживает оценке 18 баллов.

Зав. кафедрой «Информационной
технологии» СамИЭС **доц. Аликулов А.И.**

ОТЗЫВ

на магистерскую диссертацию Косимова Д. на тему

«Сравнение методов численного решения задач оптимизации»

От правильной организации одномерного поиска существенно зависет успех решения всей задачи. Кроме того, одномерная оптимизация, будучи простой по формулировке задачей, позволяет легко войти в общую проблематику оптимизационных задач.

Диссертационная работа Косимова Д. посвящена этим проблематикам. В первой главе для целевой функции строятся алгоритмы поиска экстремума 5-ги методов. Проделано сравнительный анализ результатов, дано заключение об эффективности предложенных методов одномерной безусловной оптимизации. Во второй главе разработана пакет прикладных программ для решения задач одномерной безусловной оптимизации .

В период работы над диссертационной работой Косимов Д. проявлял способности к творческой деятельности. Результаты являются новыми и получены самостоятельно.

Работа удовлетворяет всем требованиям и её автор, в случае успешной защиты, заслуживает оценке 27 баллов.

Научный руководитель: **доц. Маматов Ш.С**

ANNOTATSIYA

1-bobda ekstremumni topisning algoritmlari quyidagi metodlar uchun keltirilgan:

- minimumni passiv qidiruv algoritmi;
- minimumni aktiv qidiruv algoritmi;
- tekis blokli qidiruv algoritmi;
- oraliqni teng 2ga bo’lish algoritmi;
- dixotomiya metodi algoritmi;
- oltin kesim metodi algoritmi;
- urinmalar metodi algoritmi;

2-bobda “Сравнение методов оптимизации” nomli umumiy dastur tuzilgan va dastur algoritmining ishlash qoidalari bayon qilingan. Dastur 4 ta metod uchun tuzilgan: teng 2 ga bo’lish, oltin kesim va parabolik approksimatsiyalah metodlari.

Dastur javobida kesmaning o’rtasi, optimallik mezoni va iteratsiyalar soni ko’rsatiladi.

Dastur Dellphi7 algoritmik tilida tuzildi.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методы оптимизации очень широко используются на практике [1,5,7,8,12].

Существует достаточно большое количество численных методов оптимизации, классифицирующиеся следующим образом:

1. По размерности решаемой задачи: одномерные и многомерные.

2. По способу формирования шага многомерные методы делятся на следующие виды:

2.1. Градиентные.

- по способу вычисления градиента: с парной пробой и центральной пробой;
- по алгоритму коррекции шага;
- по алгоритму вычисления новой точки: одношаговые и многошаговые.

2.2. Безградиентные: с поочередным изменением переменных и с одновременным изменением переменных.

2.3. Случайного поиска: с чисто случайной стратегией и со смешанной стратегией.

3. По наличию активных ограничений.

3.1. Без ограничений (безусловные).

3.2. С ограничениями (условные):

- с ограничениями типа равенств;
- с ограничениями типа неравенств;
- смешанные.

Методы одномерной оптимизации являются базой для некоторых «многомерных» методов [6, 9, 10, 13-16]. **В многомерной градиентной оптимизации** строится улучшающая последовательность в зависимости от скорости изменения критерия по различным направлениям. При этом под улучшающей последовательностью понимается такая последовательность x_0, x_1,... , x_i, ..., в каждой точке которой значение критерия оптимальности лучше, чем в предыдущей. **В**

безградиентных методах величина и направление шага к оптимуму при построении улучшающей последовательности формируется однозначно по определенным детерминированным функциям в зависимости от свойств критерия оптимальности в окрестности текущей точки без использования производных (т.е. градиента).

Случайные методы используются в задачах высокой размерности. Многомерная условная оптимизация учитывает активные ограничения, выраженные в виде равенств и неравенств. В каждом из рассмотренных направлений имеется большое число методов, обладающих своими достоинствами и недостатками, которые зависят прежде всего от свойств тех функций, экстремум которых ищется. Одним из сравнительных показателей качества метода является количество значений функции, которое нужно вычислить для решения задачи с заданной погрешностью. Чем это число меньше, тем при прочих равных условиях эффективнее метод.

Постановка задачи. При решении конкретной задачи оптимизации, прежде всего, выбирается математической метод, который приводил бы к конечным результатом с наименьшими затратами на вычисления или же давал возможность получить наибольший объём информации об искомом решении. Выбор того или иного метода в значительной степени определяется постановкой оптимальной задачи, а также используемой математической моделью объекта оптимизации.

В настоящее время для решения оптимальных задач применяют в основном следующие методы [5, 8, 12, 23, 25]:

- Методы исследования функций классического анализа;
- Методы, основанные на использовании неопределенных множителей Лагранжа;
- вариационное исчисление;
- динамическое программирование;
- принцип максимума;
- линейное программирование;

В последнее время разработаны и успешно применяется для решения определенного класса задач метод геометрического программирования.

Как правило, нельзя рекомендовать какой либо один метод, который можно использовать для решения всех без исключения задач, возникающих на практике. Одни методы в этом отношении являются более общими, другие менее общими. Наконец, целью группу методов (методы исследования функций классического анализа, метод множителей Лагранжа, методы нелинейного программирования) на определенных этапах решения оптимальной задачи можно применять в сочетании с другими, например динамическим программированием или принципом максимума [20-22, 26-29].

В данной магистерской диссертации рассматривается задачи сравнение методов одномерной оптимизации.

Актуалность. Поиск экстремума функции одной переменной имеет самостоятельный интерес, так как является составной частью многих методов многомерной оптимизации. От правильной организации одномерного поиска существенно зависит успех решения всей задачи. Кроме того, одномерная оптимизация, будучи простой по формулировке задачей, позволяет легко войти в общую проблематику оптимизационных задач.

Цель и задачи работы. Целью настоящей работы является знакомство с оптимизационными задачами, изучение различных методов одномерной оптимизации, сравнение различных методов одномерной оптимизации и сравнение эффективности их решения для целевой функции $R(x)=D\sin(Ax^B + C)$ на интервале от [-1; 2] с погрешностью в 0.05.

Создать программу, «Сравнение методов оптимизации», предназначенную для решения задач оптимизации численными методами. Всего в программе должно быть четыре метода:

1) метод сканирования,
2) метод деления пополам,
3) метод золотого сечения,
4) метод параболической аппроксимации.

В ответе должны выводиться: середина отрезка, критерий оптимальности и количество итераций.

Научная новизна. Сравнительный анализ методов одномерной безусловной оптимизации. Составление пакет прикладных программ, реализующй алгоритм решения задач методов оптимизации.

Степень разработанности задачи. Создан программа «Сравнение методов оптимизации», предназначенная для решения задач оптимизации численными методами: метод сканирования; метод деления пополам; метод золотого сечения; метод параболической аппроксимации.

Предмет исследования. Численные решения задачи методами безусловной оптимизации и их сравнение.

Объект исследования. Одномерные задачи безусловной оптимизации.

Методы исследования. Задача оптимизации решается следующими методами одномерной оптимизации:

1) метод сканирования,
2) метод деления пополам,
3) метод золотого сечения,
4) метод параболической аппроксимации.

Научное и практическое значение. Проделано сравнительный анализ 4-х методов одномерной безусловной оптимизации. Составлена программа, реализующий алгоритм решения задач методов оптимизации. Результатами работы могут пользоваться на практических и лабораторных занятиях по численным методам оптимизации.

Структура работа. Работа состоит из введения, двух глав, заключение и списка используемых литератур. Общий объём работы составляет 70 страниц.

ГЛАВА 1.
СРАВНЕНИЕ МЕТОДОВ ОДНОМЕРНОЙ БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ

Оптимизация 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. По этому 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. Допустим, 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 разместить 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.

Конечно, и во втором примере при выборе решения можно действовать интуитивно, опираясь на опыт и здравый смысл. Но решения окажутся гораздо более разумными, если они будут подкреплены количественными, математическими расчетами. Эти предварительные расчеты помогут избежать длительного и дорогостоящего поиска правильного решения "на ощупь".

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 мето-

дов, позволяющих заранее оценить последствия каждого решения, заранее отбросить недопустимые варианты и рекомендовать те, которые представляются наиболее удачными.

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 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.

1.1. Постановка задачи одномерной безусловной оптимизации

Поиск экстремума функций одной переменной имеет самостоятельный интерес, так как является составной частью многих методов многомерной оптимизмации. От правильной организации одномерного поиска существенно зависит успех решения всей задачи. Кроме того, одномерная оптимизация, будучи простой по формулировке задачей, позволяет легко войти в общую проблематику оптимизационных задач [14-16].

Далее, для конкретности, мы будем рассматривать задачи оптимизации на примеры задачи минимизации в силу эквивалентности двух типов оптимизационных задач (максимизации и минимизации). Задача поиска минимум целевой функции формулируется в виде:

$$x^*=argminf(x), x\in X.$$

где *X*-множество допустимых точек, среди которых ищется точка x^*.

Другая распространенная запись задачи минимизации.

$$minf(x), x\in X$$

Когда $X = R$, мы имеем дело с одномерной безусловной задачей минимизациц, т.е. когда целевая функция $f(x)$ имеет только один простой аргумент и область X есть вся вещественная ось чисел.

В методах одномерной оптимизации вместо $X = R$ рассматривается отрезок $X = [a, b]$ содержащей искомое решение x^*. Такой отрезок называется отрезком неопределенности, или отрезком локализации. Относительно целевой функции $f(x)$ часто предполагается, что она унимодальная.

Определение: Функция $f(x)$ называется унимодальной на $X = [a, b]$, если существует такая точка $x^* \in X$,что

$$f(x_1) > f(x_2), \text{если } x_1 < x_2 < x^*, x_1, x_2 \in X,$$

$$f(x_1) < f(x_2), \text{если } x^* < x_1 < x_2, x_1, x_2 \in X,$$

Если ограничиваться рассмотрением лишь неперывных функций $f(x)$, то свойство унимодальности функции попросту означает наличие у нее единственного локального минимум и этот минимума достигается в точке $x = x^*$.

В ряде методов относительно целевой функции $f(x)$ предполагается, что она выпуклая на X.

Определение: Функция $f(x)$ называется выпуклой на $X=(a,b)$ если

$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1- \alpha) f(x_2)$ при любых $x_1, x_2 \in X$, и всех α, $0 \leq \alpha < 1$.

Если при любых $x_1, x_2 \in X$ неравенство будет строгим-строго выпуклой . Непрерывная строго выпуклая функция является унимодальной. Однако не всякая унимодальная функция является выпуклой или непрерывной.

1.2. Алгоритм пассивного поиска минимума

Отрезок (a, b) исходный отрезок неопределенности. Пусть N-число точек, в которых необходимо провести вычисления целевой функции $f(x)$, т.е. N экспериментов. Точки, в которых необходимо провести эксперименты, определяются следующим образом:

- Если $N = 2к - 1$ - нечётное, то x_i=a+$\frac{b-a}{N+1} * i$; i=1,2,3,…, N
- Если $N = 2k$-чётное, то x_{2i}=a+$\frac{b-a}{k+1} * i$; x_{2i-1}=x_{2i}-δ; i=1,2,…,k

Среди вычисленных значений $\{f(x_i=)\}$ (i =1,2,..., N), ищется точка x_j,в которой достигаются минимум:

$$f(x_j) = \min_{1 \le i \le n} f(x_i)$$

Найденная точка принимается за приближенное решение задачи $\tilde{x}$=x_j. Исходный отрезок неопределенности [a,b] после экспериментов [x_{j-1}, x_{j+1}], в N точках сужается до [x_{j-1}, x_{j+1}], длина которого

$L_N = L_N(x_1, x_2, \ldots, x_n)$=

$$\max_{1 \le i \le n}(x_{i+1} - x_{i-1}) = x_{j+1} - x_{j-1} = \begin{cases} 2\frac{b-a}{N+1}, & \text{если } N = 2k-1 \\ \frac{b-a}{\frac{N}{2}+1} + \delta & \text{если } N = 2k \end{cases}$$

Точность найденного решения $\tilde{x}$равно половине отрезка неопределенности, т.е.$|x^* - \tilde{x}| \le \varepsilon$,где $\varepsilon = \frac{1}{2} L_N$ и x^* - точное решение.

1.3.Алгоритм активного поиска минимума

В алгоритмах активного поиска очередная точка, в которой производится эксперимент, выбирается с учётом информации, полученной в предыдущих опытах. Рассмотрение этих алгоритмов начнем с методов блочного поиска, которое сочетают в себе пассивные и последовательные стратегии поиска. При этом вычисления в точках объедини в блоке, в каждом из которых проводится одновременно n_i, экспериментов, общее число экспериментов, будет $\sum_{i=1}^{m} n_i$ =N), т.е. блок-это совокупность из нескольких экспериментов, которые проводятся одновременного (пассивный поиск).

Результаты, полученные в $(i-1)$-м блоке$\{x_{ij}(j=1,2,\ldots,n_i)\}$ (последовательный поиск). Если размере блоков равны единице, т.е.$n_i = 1,$ то мы имеем обычный последовательный алгоритм поиска.

1.3.1. Алгоритм равномерного блочного поиска.

Схема алгоритма.

Шаг 1.Задаются исходный отрезок неопределенности $[a,b], \varepsilon$ - точность приближенного решения $\tilde{x},$ число экспериментов

В блоке *n* - нечётное($n = 2к - 1$). Проводим эксперимент в средине отрезка [*a*,*b*], т.е. вычисляем $y_k = f(x_k),$ где $x_k = (a+b)/2.$

Шаг 2. Проводим эксперименты в остальных точках блока: $i = 1,2,\ldots,n, i \neq к.$ находим точку $x_i,$ в которой достигается минимум среди вычисленных значений:

$$f(x_i) = minf(x),$$

следовательно точное значение минимума x^* содержится на отрезке $[x_{j-1}, x_{j+1}]$.

Шаг 3. Полагаем $a = x_j - 1$, $b = x_j + 1$, $x_k = x_j$, $y_j = y_j$. Если $b\text{-}a \leq 2\varepsilon$, то $\tilde{x} = x_k$, $\tilde{y} = y_k$ и поиск заканчивается. Иначе перейти к шагу 2. Если заданная точностъ ε достигнута после *m* итераций, т.е. после экспериментов в *m* блоках, то длина отрезка неопределённости после всех *n* вычислений

$$(N = n + (n-1)(m-1)m + 1)$$

будет:

$$\text{Ln}=\left(2\frac{b-a}{N+1}\right)^m \text{ и } |x^* - \tilde{x}| \leq \frac{1}{2}\text{Ln}$$

1.3.2. Алгоритм деления интервал пополам.

Это вариант предыдущего алгоритма при $n = 3$.

Схема алгоритма.

Шаг 1. Задаются a,b,ε. Производим эксперимент в точке $x_2=(a+b)/2$, т.е. вычисляем $y_n = f(x_n)$.

Шаг 2. Производим эксперименты в остальных точках блока:

$$x_1 = (a + x_n)/2, y = f(x_1), \qquad x_3 = (x_n + b), y_3 = f(x_3).$$

Находим x_j такую, что

$$f(x_i)=\min_{1<i\leq 3}\{f(x_i)\}.$$

Тогда точное решение x^* содержится на отрезке $[x_{j-1}, x_{j+1}]$.

Предполагается $x_0 = a$, $x_4 = b$.

Шаг 3. Полагаем $a=x_{j-1}$, $b= x_{j+1}$, $y_2=y_j$. Если

$b - a \leq \varepsilon$, то $\tilde{x} = x_2$, $\tilde{y} = y_2$ и поиск заканчивается. Иначе перейти к шагу 2.

Поиск к итераций общее число проведенных экспериментов равно $N = 2k + 1$, а длина получившегося отрезка неопределенности будет

$$L_N=\frac{b-a}{2^k}=\frac{b-a}{2^{[N/2]}},$$

где [z] - целая часть числа z .

Следовательно, достигнутая точность будет

$$|x^*-\tilde{x}|\leq \varepsilon, \qquad \varepsilon=\frac{1}{2}.$$

1.3.3 Метод дихотомии.

Это алгоритм блочного поиска для $n_i = n = 2$ т.е. когда в блоке два эксперимента. Так как пассивная составляющая алгоритма, т.е. блок, содержит четное число экспериментов, то оптимальный выбор точек X_{ij} в которых необходимо провести эксперименты, будет неравномерным, в отличие от предыду-

щих алгоритмов, где число экспериментов в блоке было нечетным и. соответственно, расположение точек равномерным. Если блок содержит два эксперимента, то оптимальное расположение точек, в которых будут проводиться эксперименты, это как можно ближе к середине отрезка. Такое расположение точек позволяет получить наименьший отрезок неопределенностей после экспериментов в блоке.

Схема алгоритма.

Шаг 1. Задаются a,b,ε и δ малое положительное число, значительно меньшее чем ε.

Шаг 2. Определяется середина отрезка $x=(a+b)/2$. Производятся эксперименты в двух точках, близких к середине:

$$y_1=f(x-\delta), y_2=f(x+\delta),$$

Шаг 3. Определяется следующей отрезок локализации, т.е. определяется какой из отрезков $[a,x+\delta]$ и $[x-\delta,b]$ содержит точное решение x^*. Если $y_1 \le y_2$, то это отрезок $[c,x+\delta]$ и $a=x-\delta$, т.е выбранный отрезок локализации мы снова обозначили как $[a,b]$.

Шаг 4. Если $b-a<2\varepsilon$, то $x=(a+b)/2$, $y=f(\tilde{x})$ и поиск заканчивается. Иначе перейти к шагу 2.

Поиск к итераций общее число экспериментов будет $N=2k$, а длина получившегося отрезка неопределенности

$$L_N<\frac{b-a}{2^{n/2}}+\varepsilon$$

Следовательно,

$$|x^*-\tilde{x}|<\tfrac{1}{2}L$$

А теперь перейдём к рассмотрению чисто последовательных методов поиска.

1.3.4. Метод золотого сечения.

Для того чтобы уменьшить отрезок неопределённости $[a,b]$ нам необходимо вычислить значение целевой функции *f*(*x*) по крайней мере, в двух точках на отрезке $[a,b]$.

В результате этих двух экспериментов отрезок неопределённости сузится до отрезка $[a,x_2]$ или $[x_1,b]$. Так как у нас нет никаких оснований предпочесть один из этих вариантов, то точки x_1 и x_2 должны быть симметричны относительно середины отрезка $[a,b]$. В этом случае длины отрезков $[a,x_2]$ и $[x_1,b]$.] будут равны. Таким образом, остаётся вопрос как выбрать точку x_1(см. рис.1).

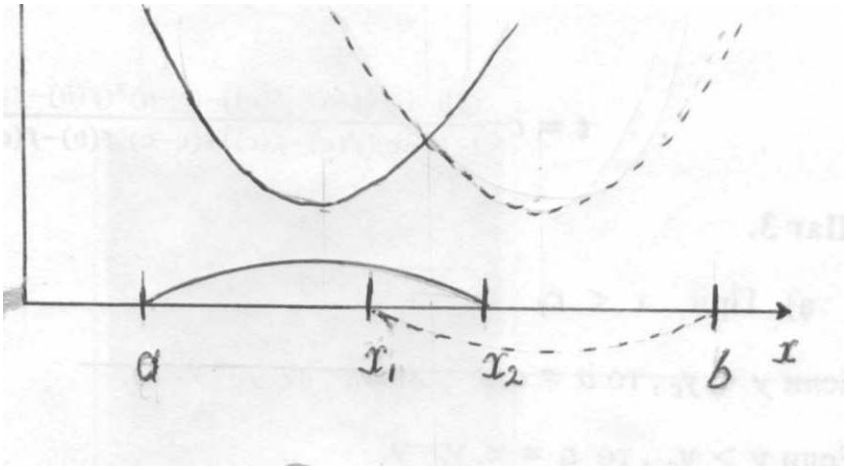


Рис.1.

В методе золотого сечения точка x_1 выбирается из соображения, что должно выполняется соотношение:

$$\frac{\text{длина } [a,b]}{\text{длина } [x_1,b]} = \frac{\text{длина } [x_1,b]}{\text{длина } [a,x_1]} = \lambda = 1{,}68033989\ldots$$

т.е. x_1 делит отрезок $[a,b]$ по правилу и “золотого сечения ”, где λ –есть “золотое отношение”. Точка x_2 определяется как точка симметричная к x_1 относительно середины отрезка.

В результате экспериментов у нас получается отрезок неопределённости $[a_1, x_2]$, содержащий точку x_1, или отрезок неопределённости $[x_1, b]$, содержащий точку x_2.

1.4. Методы поиска, основанные на аппроксимации целевой функции

Суть этих методов заключается в том, что по полученной в ходе вычислений информации строится аппроксимирующая функция и её минимум принимается за точку очередного вычисления. Такие методы дают хорошие результаты при минимизации достаточно гладких унимодальных функций.

1.4.1. Метод касательных.

Пусть функция $f(x)$ выпукла и дифференцируема на $[a,b]$. Идея метода состоит в следующем. Пусть $[a,b]$ – отрезок неопределённости и $f(a), f'(a), f(b), f'(b)$ – результаты вычислений в точках a и b. По этой информации строится аппроксимирующая функция, представляющую из себя кусочно-линейную функцию, состоящую из касательной

$$L_a(x) = f(a) + f'(a)(x-a)$$

в точке a и касательной $L_b(x) = f(b)+f'(b))(x-b)$ к $f(x)$ в точке b.

Полученная аппроксимирующая функция есть ломаная, состоящая из прямой $L_a(x)$ на $[a,c]$ и $L_b(x)$ на $[c,b]$, где c – точка пересечения касательных (см. рис 2). Легко заметить что при $f(a) < 0$ и $f(b) > 0$ минимум аппроксимирующей функции достигается в точке c. Значение точки пересечения с можно определить по формуле

$$c = \frac{\left(bf'(b) - af'(a)\right) - (f(b) - f(a))}{f'(b) - f'(a)},$$

В точке производятся вычисления $f(c)$ и $f'(c)$. Если $f'(c) = 0$, то решением задачи будет $x^* = c$. Если же $f'(c) > 0$, то в качестве следующего отрезка

неопределённости будет [a,c]. Если же $f'(c) < 0$, то – отрезок $[c, b]$. Процесс повторяется до тех пор пока $f'(c) = 0$, или отрезок не определенности не достигнет заданной точности.

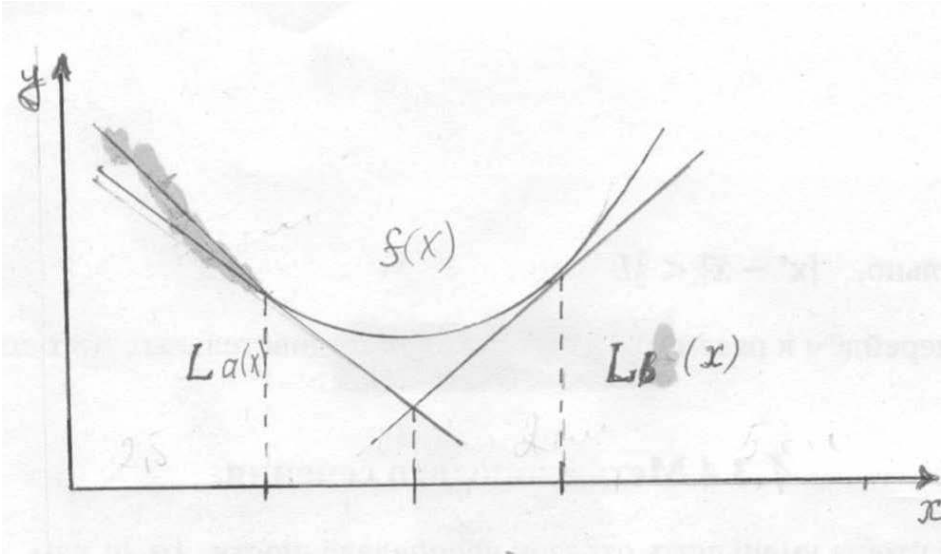


Рис.2

Шаг 1. Заданы *a*,*b*,ε вычислить $y_1 = f(a)$, $y_2 = f(b)$, $z_1 = f'(a)$, $z_2 = f'(b)$.

Шаг 2. Заданы $b - a < 2\varepsilon$, то получаем $\tilde{x} = (a + b)/2$, $\tilde{y} = f(\tilde{x})$. Поиск окончен.

Если $b - a < 2\varepsilon$, то вычислить $c = \frac{(bz_2 - az_1) - (y_2 - y_1)}{z_2 - z_1}$, $y = f(c)$, $z = f'(c)$.

Если $z = 0$, то полагаем $\tilde{x} = c$, $\tilde{y} = y$ и поиск окончен.

Если $z < 0$, то $a = c$, $y_1 = y, z_1 = z$.

Если $z > 0$, то $b = c, y_2 = y$, $z_1 = z$.

Повторить шаг 2.

1.4.2. Метод парабол.

Рассмотрим алгоритм квадратичной интерполяции или метод парабол, т.е. в качестве аппроксимирующей функции используется парабола. Для однозначного задания параболы необходимы три точки. Пусть имеются три точки, для которых выполняется $a < c < b, f(c) \le f(a), f(c) \le f(b)$. Так как $[c, b]$отре-

зок неопределенности и $f(x)$– унимодальная функция, то найти такую точку c нетрудно. Парабола, проходящая через три точки $(a,f(a)),(c,f(c)),(b,f(b))$ имеет вид

$$P(x)=\left(\frac{f(b)-f(c)}{(b-c)}+\frac{f(a)-f(c)}{(a-c)}\right)\frac{(x-c)-(x-b)}{(b-a)}+\frac{f(b)-f(c)}{(b-c)}\ (x-c)\ +f(c)$$

Поскольку f – унимодальная функция, то выполняется хотя бы одно из неравенств $f(c)\le f(a),\ \ f(c)\le f(b).$ строго и, следовательно, коэффициент при старшем члене $P(x)$ положителен. Тогда $P(x)$ достигает минимума в точке

$$t=c+\frac{1}{2}\frac{(b-c)^2\left(f(a)-f(c)\right)-(c-a)^2(f(b)-f(c))}{(b-c)\left(f(a)-f(c)\right)+(c-a)(f(b)-f(c))}$$

Причем $(a+c)(2\le t\le(c-b))/2.$ Эта точка выбирается в качестве точки очередного вычисления значения функции (см. рис. 3 и 4).

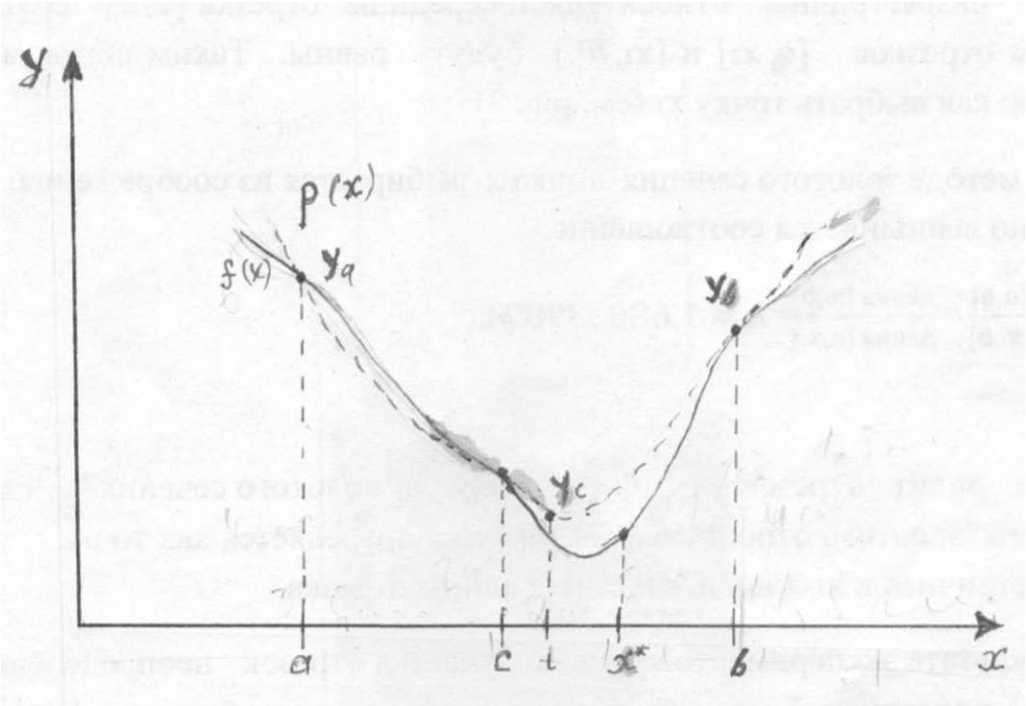


Рис.3

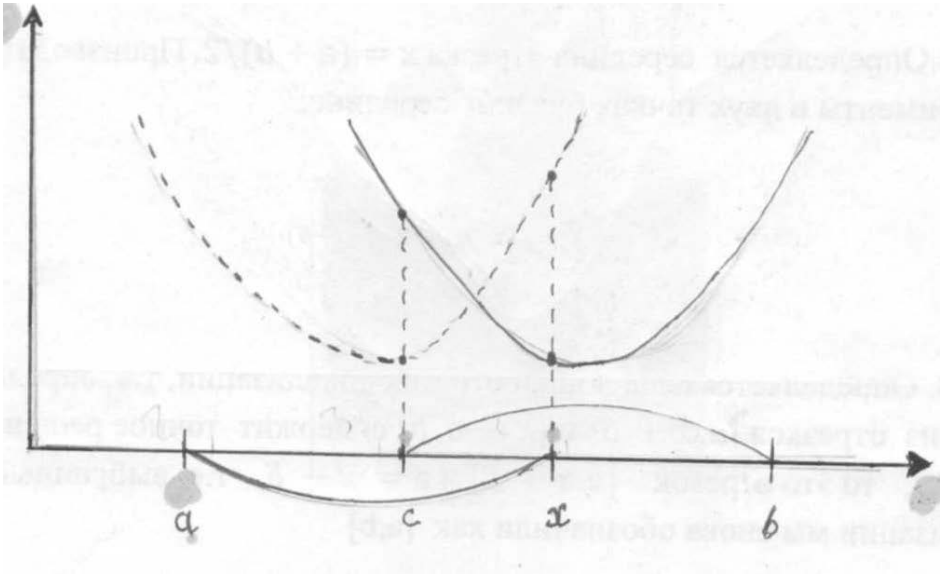


Рис.4

Если оказалось, что $t = c$ условимся в качестве точке очередного вычисления выбирать точку $(a + c)/2$. Итак, следующее вычисление проводится в точке

$$x=\begin{cases} t, & \text{если } t \neq c \\ (a+c)/2, & \text{если } t \neq c \end{cases}$$

Определим новый отрезок неопределенности служащей внутри него точкой, для которой выполняются условия, аналогичные условиям, которым удовлетворяло точка *c*.

В силу унимодальности функции f и b зависимости от выполнения или невыполнение условий $x < c$, $f(a) \leq f(x)$, $f(x) \leq f(c)$.

Это будут отрезки с точкой внутри

: $[a,c]$ и x; $[x,b]$ и c; $[x,c]$ и $(x + c)/2$; $[a,]$ и c; $[c,b]$ и c; $[c,x]$ и c; $[c,x]$и $(x + c)/2$.

(смотри рис.4). Далее строится парабола, определяется ее минимум, и т.д, до тех пор пока длина отрезка неопределенности не удовлетворяет заданной точности. (см. рис.5)

Схема алгоритма.

Шаг 1. Задаются a, c, b и ε. Вычислить $y_a=f(a), y_c=f(c), y_b=f(b)$.

Шаг 2. Вычислить

$$x = \begin{cases} t, \text{если } \_t \neq c \\ \dfrac{a+c}{2}, \text{если } \_t \neq c \end{cases}, y = f(x), \text{ где}$$

$$t = c + \frac{1}{2}\frac{(b-c)^2\left(f(a)-f(c)\right)-(c-a)^2(f(b)-f(c))}{(b-c)\left(f(a)-f(c)\right)+(c-a)(f(b)-f(c))},$$

Шаг 3.

a) При $x < c.$

- Если $y < y_c$, то $b = c, c = x, y_a = y_c\ y_c = y$.
- Если $y > y_c$, то $a = x$, $y_c = y$.
- Если $y = y_c$, то $a = x,\ b = c, c = \frac{x+c}{2}, y_a = y\ , y_b = y_c, y_c = f(c)$

b) При$x > c$

- Если $y < y_c$, то $a = c, c = x,\ y_a = y_c\ y_c = y$.
- Если $y > y_c$, то $b = x, y_b = y$.
- Если $y = y_c$, то a=c, $b = x$, $c = \frac{x+c}{2}, y_b = y\ , y_a = y_c, y_c = f(c)$

Шаг 4. Если $b - a < \varepsilon$, то закончить поиск, положив $\tilde{x} = x$, $\tilde{y} = y$ иначе перейти к шагу 2.

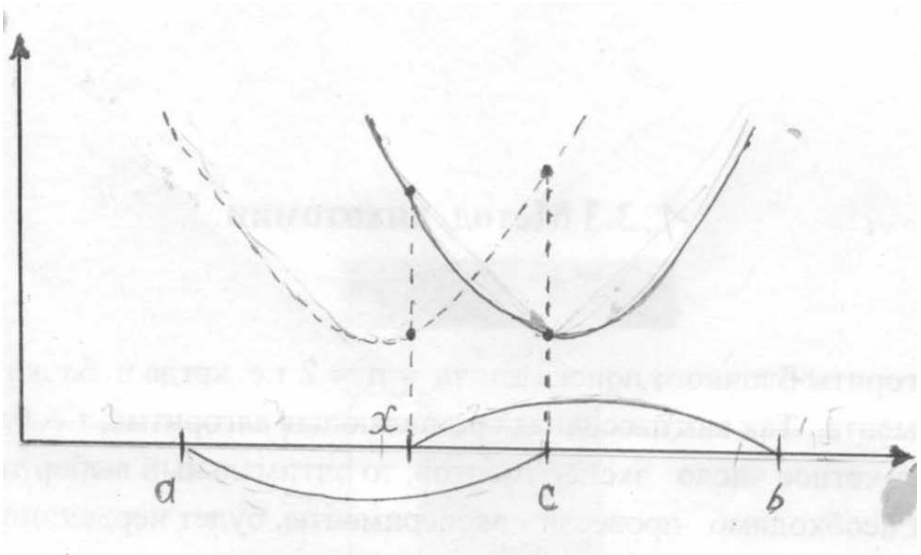


Рис.5.

Все эти алгоритмы реализованы нижеприведенными программами [2-4].

ПРОГРАММЫ РЕАЛИЗАЦИИ РАСЧЕТОВ

1. Программа для метода золотого сечения

```
//Для n=2
#inclyde<iostream.>
 #inclyde<math.h>
#inclyde<windows>
Void main()
{
Float  a.b.t//пункт№1
Int h.N
Float E,xp,yp,x1,y1,x2,y2
    a=0;
    b=1
cout<<”N=”
    cin>>N;
t=(1+sqrt(5))/2;
h=0
x1=a+(b-a)/pow(t,2);
y1=x1*x1*x1*x1-1.5*atan(x2);
h=h+2
do if(y1<y2){b=x2;x2=x1;y2=y1;x1=a+b-x2;
yl=x1*x1*x1*x1-1.5*atan(x1);h=h+1;
        }
else{a=x1;x1=x2;y1=y2:
       x2=a+b-x1;y2=x2*x2*x2*-1.5*atan(x2);h=h+1;
              }
```

```
}
Wlhile(h<=N);// Пункт№7
{if(y1<y2)b=x2;
  else  a=x1
xp=(a+b)/2;
yp=xp*xp*xp*xp*-1.5*atan(xp);
E=(b-a)/2;
cout << “xp=”<<xp<<endl;
cout << “xp=”<<yp<<endl;
cout<<”E=”<<E<< endl;
system (“pause”);
```

2. Программа для метода Фибоначчи

```
#inclyde<iostream.>
 #inclyde<math.h>
#inclyde<windows>
Void main()
{
float  a,b,d,t*F,x1,x2,y1,y2 xp,yp;// пункт№1
int N,I;
float E;
a=0;
b=1;
cout<<”N=”
cin>>N;
```

```
t=(1+sqrt(5))/2;
F[0]=1;
F[1] =1;
For(i=2,i<=N;i++) F[i]=F[i-2];
D=(b-a)*(F[N-2]/F[N]);
y1=x1*x1*x1*x1-1.5*atan(x1);
x2=a+(b-a)*(F[N-1]/F[N];
y2=x2*x2*x2*x2*-1.5*atan(x2);
```

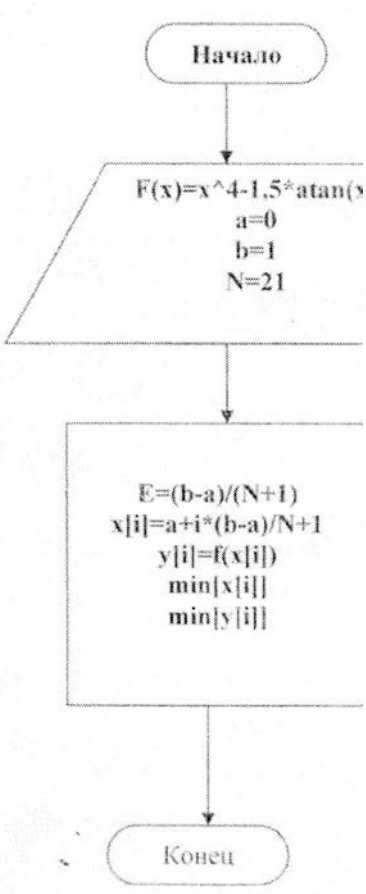
3. Программа для метода деления интервала пополам

```
#1)Для  n=3
#include<iostream.h>
# include<math.h>
# include<windows.h>
Void main()
|        Float a,b;//пупкт №1
        Int  I,h,N,n,f;
Float min .E,xp,yp.*x.*y;
   a=0;
   b=1:
cout<<”N=”>;
  cin>>N;
n=3;
x=new float[n+2]
y= new float[n+2]
```

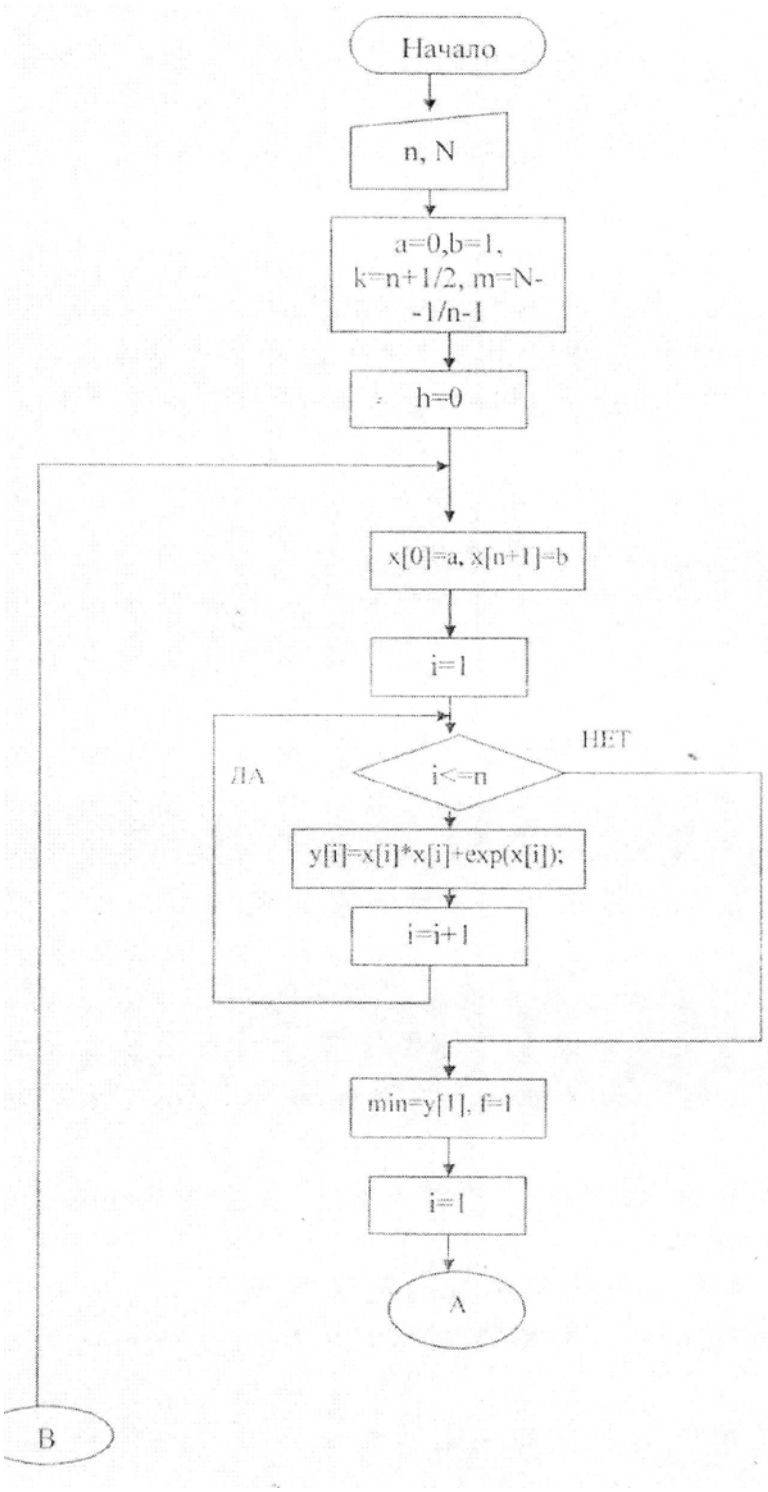
```
      x[2]=[a+b]/2;
y[2]=x[2]*x[2] *x[2] *x[2]-1.5*atan(x[2]);
h=l;
do
|X[1]=(a+x[2])/2;
Y[1]=x[1]*x[1] *x[1] *x[1]-1.5*atan(x[1]);
X[3]=(x[2]+b)/2;
Y[3]=x[3]*x[3] *x[3] *x[3]-1.5*atan(x[3]);
    H=h+2
    X[0]=a;
X[n+1]=b
Min=y[1];
F=1;
For(i=1;i<=n;i++)
If(y[i]<min){min=y[i]; f=I;}
A=x[f-1];
Yp=xp *xp* xp* xp-1.5*atan(xp);
E=(b-a)/2
Cout<<”xp=”<<xp<<endl;
Cout<<”yp=”<<yp<<endl;
Cout<<”E=”<<E<<endl;
System(“pause”);
```

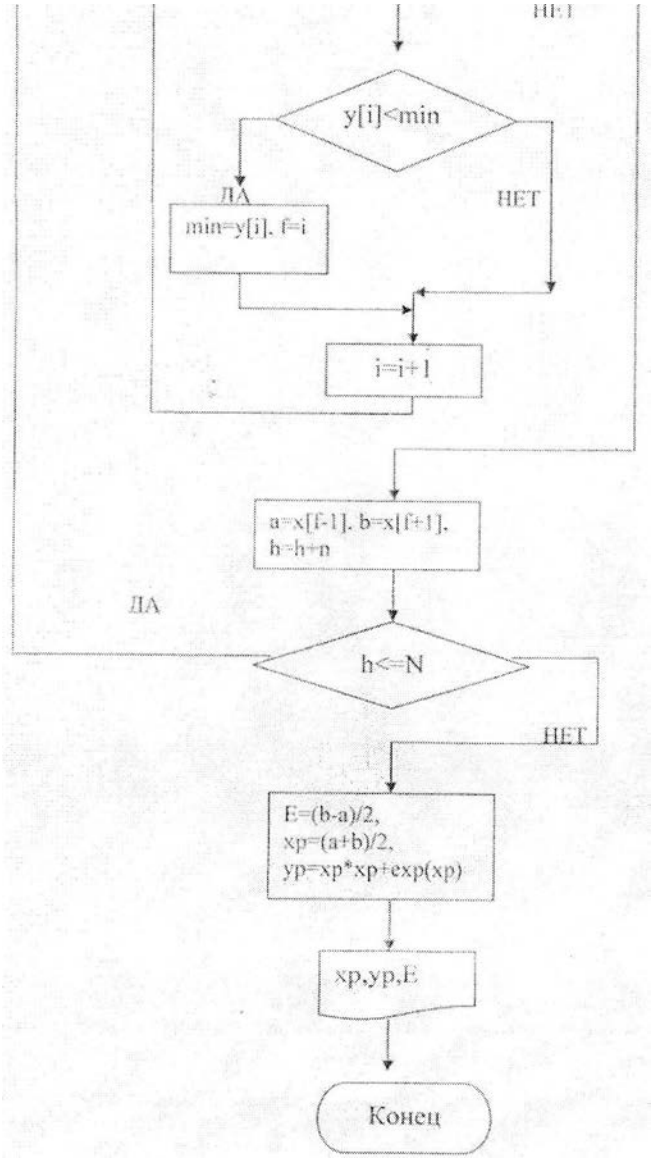
БЛОК СХЕМЫ

1. Пассивный метод

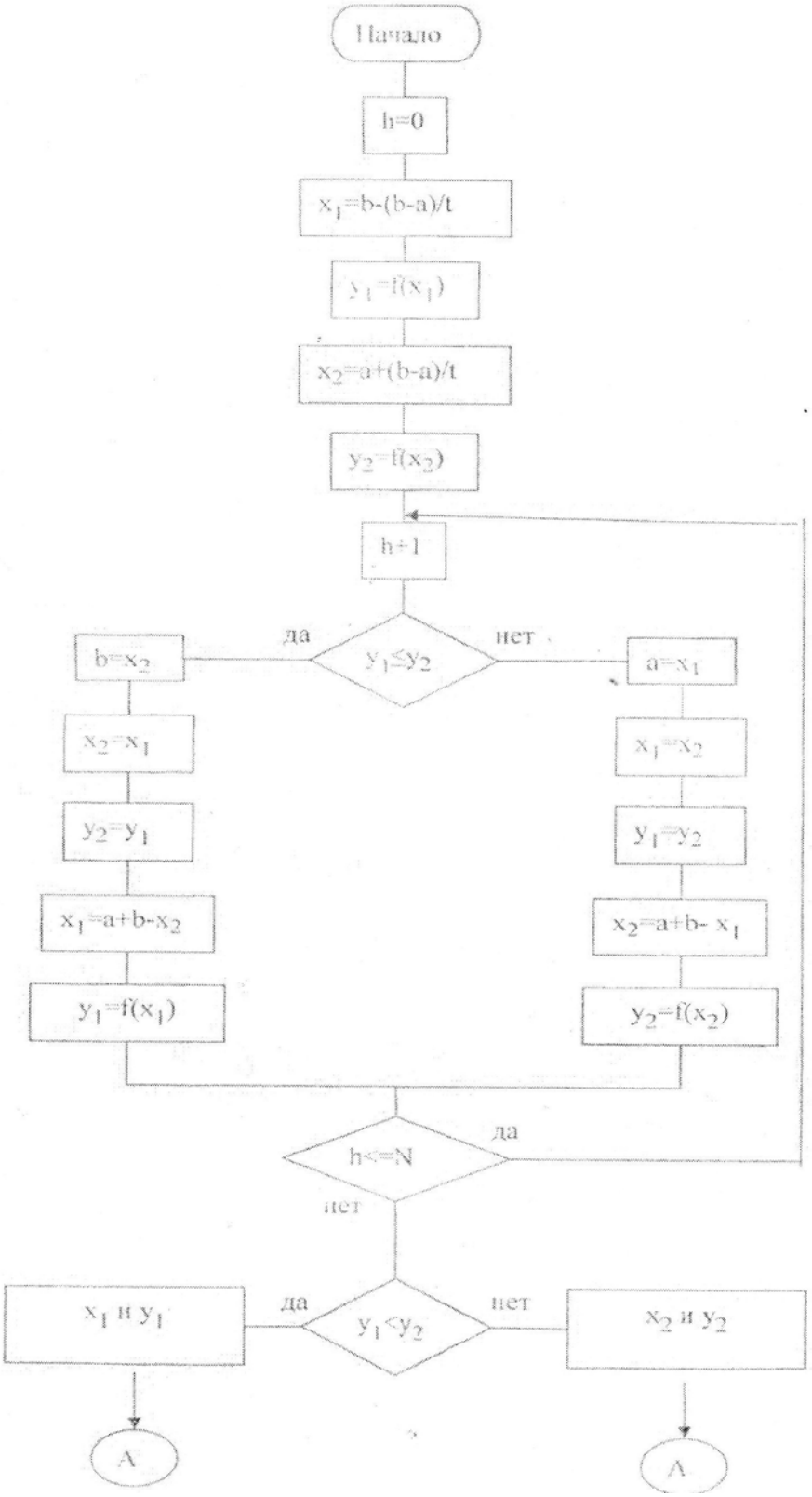


2. Алгоритм для блочного метода.

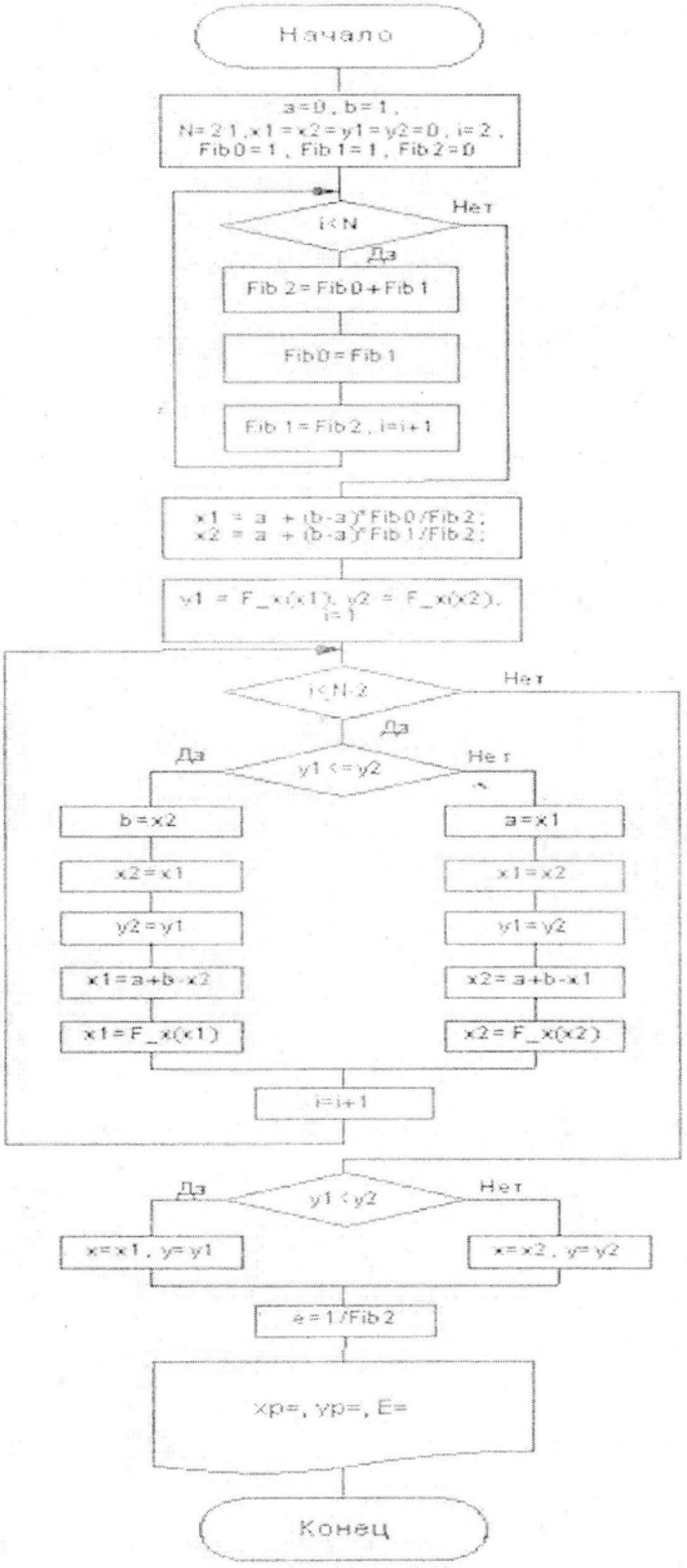



НЕТ
y[i]<min
ДА
НЕТ
min=y[i], f=i
i=i+1
a=x[f-1], b=x[f+1],
h=h+n
ДА
h<=N
НЕТ
E=(b-a)/2,
xp=(a+b)/2,
yp=xp*xp+exp(xp)
xp,yp,E
Конец

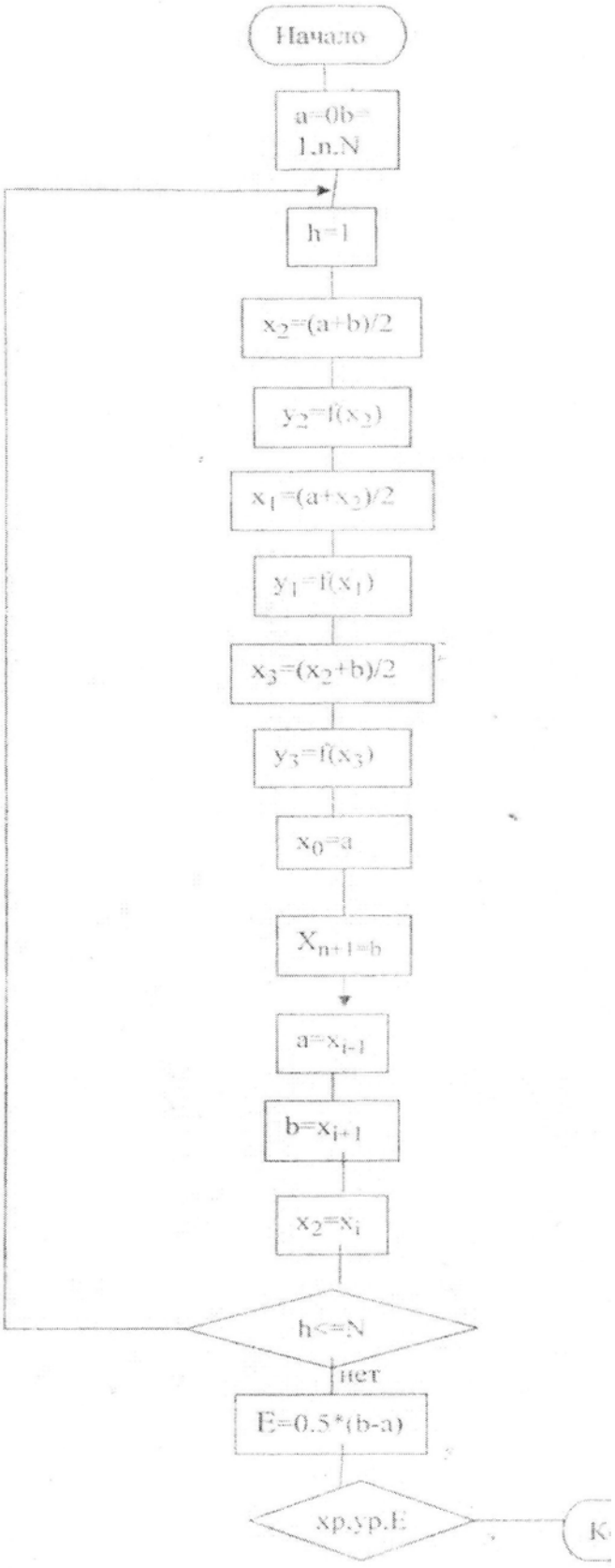
3. Метод золотого сечения



4. Метод Фибоначчи



5. Метод деления интервала пополам



ГЛАВА 2.
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СРАВНЕНИЯ МЕТОДОВ ОПТИМИЗАЦИИ

2.1 Назначение и анализ использования разработки

В практических задачах методы оптимизации предназначены для нахождения критериев оптимальности, это оптимальное проектирование - выбор наилучших номинальных технологических режимов, элементов конструкций, структуры технологических цепочек, условий экономической деятельности, повышение доходности и т.д., оптимальное управление, построение нелинейных математических моделей объектов управления (минимизации невязок различной структуры модели и реального объекта) и многие другие аспекты решения экономических и социальных проблем (например, управление запасами, трудовыми ресурсами, транспортными потоками и т.д. и т.п.).

В данной программе решается уравнение вида $R(x)=D\ \sin(Ax^B + C)$ на интервале от [-1; 2] с погрешностью в 0.05

2.2. Постановка задачи. Входная и выходная информация

Рассмотрим методы решения **одномерных задач оптимизации** вида $R(x) \rightarrow \max$, где $a \le x \le b$, где x — скаляр, a *и* b — соответственно минимальное и максимальное возможные значения переменной x.

Будем рассматривать алгоритмы, связанные с построением улучшающей последовательности. Решением задачи называется x^*, при котором $R(x^*) \ge R(x)$ для любого значения $a \le x \le b$. При решении задач не будем различать два значения x_i , и x_{i+1}, если $|x_i - x_{i+1}| \le \varepsilon$, где ε — задаваемая погрешность решения.

Метод сканирования. Заключается в последовательном переборе всех значений $a \le x \le b$ с шагом ε (погрешность решения) с вычислением критерия

оптимальности R в каждой точке. Путем выбора наибольшего из всех вычисленных значений R и находится решение задачи x^*.

Достоинство метода в том, что можно найти глобальный максимум критерия, если $R(x)$ — многоэкстремальная функция. К недостаткам данного метода относится значительное число повторных вычислений $R(x)$, что в случае сложной функции $R(x)$ требует существенных затрат времени.

На практике можно реализовать одну из основных модификаций метода — **последовательное уточнение решения**, или **сканирование с переменным шагом** (см. рис. 1).

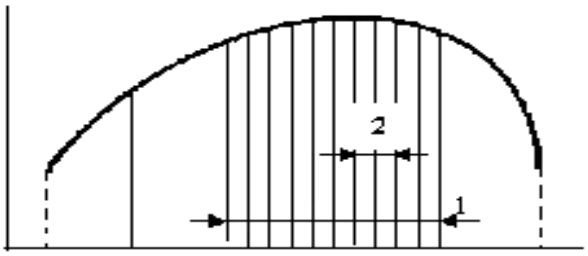


Рис. 1. Иллюстрация модифицированного метода сканирования: 1 — интервал, включающий в себя искомый максимум функции после первого этапа сканирования (исходный участок разбит на 5 участков); 2 — то же, после второго этапа.

На первом этапе сканирование осуществляют с крупным шагом, затем отрезок, внутри которого получено наибольшее значение R(x), разбивается на более мелкие отрезки, ищется, новый отрезок, внутри которого находится уточненное значение максимума.

Новый отрезок опять делится на более мелкие и т.д., до тех пор, пока величина отрезка, содержащего максимальное значение $R(x)$, не будет меньше заданной погрешности. Главный недостаток этого варианта метода — возможность пропуска «острого» глобального максимума $R(x)$.

Метод деления пополам. Метод деления пополам основан на делении текущего отрезка $[a, b]$, где содержится искомый экстремум, на две равные части с последующим выбором одной из половин, в которой локализуется максимум в качестве следующего текущего отрезка. Экстремум локализуется путем срав-

нения двух значений критерия оптимальности в точках, отстоящих от середины отрезка на $\varepsilon/2$, где ε — погрешность решения задачи оптимизации.

Если $R(x + \varepsilon/2) > R(x - \varepsilon/2)$, то максимум располагается на правой половине текущего отрезка $[a, b]$, в противном случае — на левой.

Процесс поиска завершается при достижении отрезком $[a, b]$ величины заданной погрешности ε.

К недостаткам метода относится его работоспособность только для одноэкстремальных функций $R(x)$ (т.е. таких, которые содержат один экстремум того типа, который мы ищем в задаче), так как в других случаях при сравнении двух критериев в соседних точках невозможно правильно выбрать следующий интервал, где находится максимум.

На рис. 2 приведены три этапа метода половинного деления. Сплошными вертикальными линиями отмечены середины отрезков, а пунктирными — вычисляемые значения критерия оптимальности слева и справа на $\varepsilon/2$ от середин.

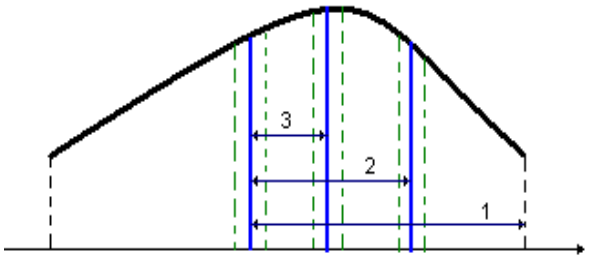


Рис. 2. Иллюстрация метода половинного деления: 1 — интервал, включающий в себя искомый максимум функции после первого этапа (первого деления пополам); 2,3 – то же соответственно после второго.

Существует и другой вариант алгоритма, заключающийся в следующем.

После нахождения середины отрезка (например, точка c_1) в одной из половинок (допустим, в левой) находят среднюю точку (точка c_2) и, сравнивая значения функции в этих точках, определяют, в какой из половинок находится экстремум. Если $R(c_1) < R(c_2)$, то в качестве следующего отрезка выбираем отрезок $[a, c_1)]$, если же $R(c_1) > R(c_2)$, то берут новую точку в середине правой половины

(точка c_3) и в ней вычисляют функцию. В зависимости от сравнения значений функции в точках c_3 и c_1) выбирают новый отрезок $[c_1, b]$ или $[c_2, c_3]$ и т.д.

Второй вариант метода не имеет с точки зрения эффективности принципиального отличия от первого, так как эффективность принято оценивать по наихудшему варианту (т.е. по двум вычислениям *f(x)* на каждом шаге). В первом варианте метода есть одна особенность, которая его делает очень эффективным при экспериментальном отыскании экстремума (например, при автоматической настройке технических систем или при практическом поиске наилучших условий деятельности экономического объекта). Малые отклонения от текущей точки обеспечивают в процессе поиска отсутствие «шараханий», сопровождающихся резкими отклонениями состояния системы.

Метод золотого сечения. Рассмотрим метод золотого сечения:

Золотое сечение определяется по правилу: *отношение всего отрезка к большей его части равно отношению большей части отрезка к меньшей.* Ему удовлетворяют две точки *c* и *d*, расположенные симметрично относительно середины отрезка (рис 3).

$$ab/cb=cb/ac; \quad ab/ad=ad/db;$$

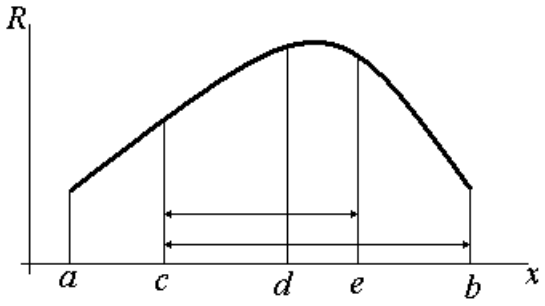


Рис. 3. Иллюстрация метода – золотого сечения: 1 — интервал, включающий в себя искомый максимум функции после , первого (первого золотого сечения в точках *c* и *d*); 2 – то же, после второго этапа (новая точка *e* и старая точка *d*)

Путем сравнения $R(c)$ и $R(d)$ определяют следующий отрезок, где содержится максимум. Если $R(d)>R(c)$, то в качестве следующего отрезка выбирается отрезок $[c, b]$, в противном случае — отрезок $[a, d]$.

Новый отрезок снова делится на неравные части по правилу золотого сечения. Следует отметить, что точка d является и точкой золотого сечения отрезка $[a, b]$, т.е.

$$db/cd = cd/cb$$

Поэтому на каждой следующей итерации (кроме «запуска» метода на исходном отрезке) нужно вычислять только одно значение критерия оптимальности.

Существуют аналитические формулы для расчета новой точки на отрезке, где находится максимальное значение $R(x)$:

$$c = a + (b-a)\frac{\sqrt{5}-1}{2} \qquad d = b - (b-a)\frac{\sqrt{5}-1}{2}$$

Условие окончания поиска — величина отрезка, содержащего максимум, меньше заданной погрешности.

Метод обеспечивает более быструю сходимость к решению, чем многие другие методы, и применим, только для одно-экстремальных функций,

На рис. приведены два этапа поиска максимума функции методом золотого сечения.

Под одно-экстремальной функцией понимают функцию, содержащую один экстремум того типа, который ищется в задаче.

Метод параболической аппроксимации заключается в замене нелинейной функции $R(x)$ квадратичной параболой $R_2(x)$, построенной по трем точкам, принадлежащим $R(x)$, с последующим нахождением max параболической функции, используя аналитические условия оптимальности:

$$dR/dx=0.$$

На первом этапе в качестве исходных трех точек используются $x_1=a$, $x_2=b$ и $x_3=(a+b)/2$. В этих точках вычисляется $R(x)$ и по полученным точкам $R(x_1)$,

$R(x_2)$, $R(x_3)$ строится парабола $R_2=C_2x^2+C_1x+C_0$, коэффициенты которой находятся из решения соответствующей системы уравнений:

$$R_2(x_1)=R(x_1),\ R_2(x_2)=R(x_2),\ R_2(x_3)=R(x_3).$$

Условие оптимальности приводит к уравнению $x_4 = -C_1/(2C_2)$, где x_4 — точка максимума параболы $R_2(x)$. Далее выбирается новый отрезок, внутри которого находится точка x_4, и, используя x_3, x_4, строится новая парабола, по которой уточняется положение максимума $R(x)$ и т.д. до тех пор, пока величина отрезка, внутри которого находится максимум, не будет меньше заданной погрешности ε. Таким образом, метод имеет итерационный характер. Можно строить параболу на каждом шаге и по трем последним точкам, но только в том случае, если точно известно, что функция гладкая и одно-экстремальная. В противном случае первый вариант даст лучший результат.

К достоинству метода относится высокая скорость сходимости к оптимуму, хотя метод может не всегда сходиться к нему.

На рис.4 приведены два случая применения метода параболической аппроксимации: а) рассмотрена ситуация, когда метод параболической аппроксимации сходится к решению, уже на третьем этапе парабола, построенная по точкам x_3, x_4, x_5, практически совпадает с исходной функцией; б) парабола не имеет максимума уже на втором этапе.

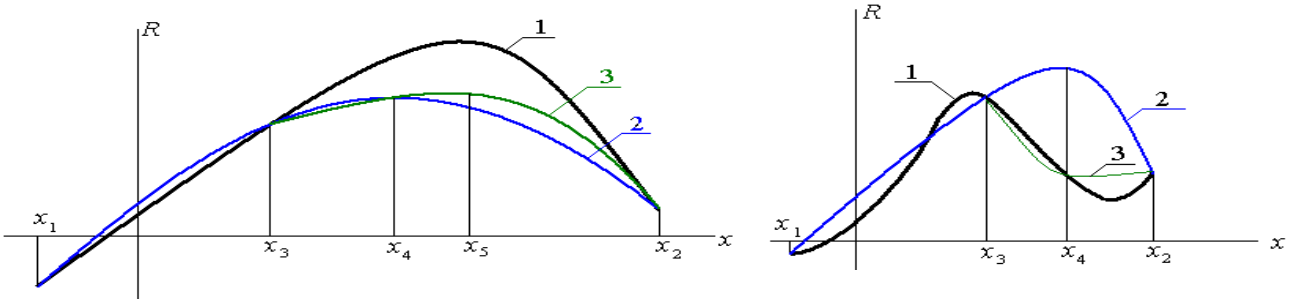


Рис. 4. Иллюстрация метода параболической аппроксимации: *а* — решение найти можно; *б* — решение найти нельзя; 1 — функция, экстремум которой ищется; 2 — аппроксимирующая парабола первого этапа, построенная по точкам x_1, x_2, x_3; 3 — аппроксимирующая парабола второго этапа, построенная по точкам x_2, x_3, x_4; x_3;— середина исходного интервала; x_2, x_4 — точка максимума первой параболы; x_5 — точка максимума второй параболы.

2.3. Описание алгоритма

Интерфейс созданной программы (рис. 5)– «Сравнение методов оптимизации» - состоит :

1) из семи полей ввода (вводятся значения A, B, C, D, погрешность и интервал (min и max));
2) из четырех переключателей (* Метод сканирования, * Метод деления пополам, * Метод золотого сечения, * Метод параболической аппроксимации);
3) кнопки Вычислить;
4) поля, в котором будет выводиться ответ;
5) кнопки Выход, а так же главного меню;

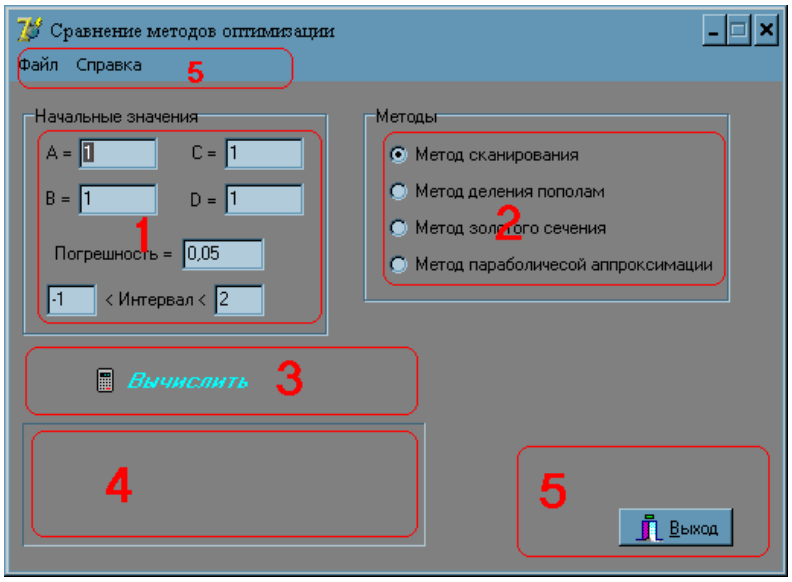


Рис 5. Интерфейс программы «Сравнение методов оптимизации»

Чтобы получить ответ необходимо выбрать метод решения и нажать на кнопку Вычислить. На этой кнопке прописан основной текст программы. Условно его можно разделить на пять частей. Первая часть содержит переменные, которые переводятся из строки в число, а так же условия их вода, вторая часть содержит текст программы метода сканирования, третья – деления пополам,

четвертая – золотого сечения, а пятая – параболической аппроксимации. Так же в программе прописана функция – одна для всех методов, на основе решения которой мы будем получать приближенные значения.

Описание алгоритма при выборе переключателя «Метод сканирования»:

1) первым делом обнулим счетчик, который считает количество итераций (it:=0);
2) оформим цикл типа **repeat - until** (до тех пор пока):
   - потом разбиваем отрезок на четыре части (Int:=(max-min)/4) и присваиваем некоторой переменной значение минимума (PER:=min);
   - оформляем цикл для нахождения массива чисел ($x[i]$) на заданном интервале;
   - на каждом интервале находим значение функции;
   - находим максимальное значение;
   - находим индекс максимального значения;
   - некоторой новой переменной присваиваем значение $x[k]$;
   - вычисляем новые значения минимума и максимума;
   - считаем количество итераций (it:=it+1);
3) на экран выводится ответ (рис. 6).

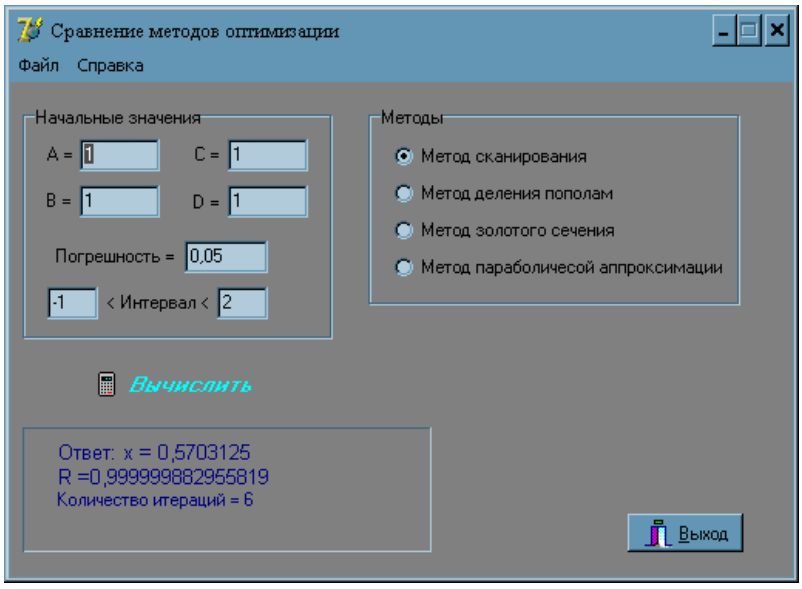


Рис 6. Приведены вычисления контрольной задачи «Методом сканирования.

Приведем пример: дана функция $R(x)=D\sin(Ax^B+C)$, где коэффициенты имеют следующие значения: A=1,0, B=1,0, C=1,0, D=1,0. Найти максимум на интервале: [-1, 2]. Ошибка задается по x: ε=0,05.

Результаты расчетов. Разобьем весь интервал на четыре подинтервала (крупный шаг), координаты x будут следующими:

$$x_1=-0{,}25,\ x_2=0{,}5,\ x_3=1{,}25.$$

Соответственно значения критерия равны:

$$R_1=0{,}68163,\ R_2=0{,}9974,\ R_3=0{,}77807.$$

Следовательно, в качестве нового отрезка выбираем отрезок [–0,25; 1,25], так как внутри него находится максимальное значение:

$$x_0=0{,}5,\quad R_0=0{,}99749499,$$

0 — номер итерации (после первого этапа). Разбиваем его снова на четыре части, имеем значения:

$$x_1=0{,}125,\quad x_2=0{,}5,\quad x_3=0{,}875.$$

Вычислив $R(x)$ в этих точках, получим, что новый интервал, внутри которого лежит экстремум, равен [0,125ж 0,875].

Далее в табл. №1 приводятся только координаты середин отрезков, при которых критерий имеет наибольшее значение, номер итерации и значение критерия.

Табл. №1 (Расчеты данных по этапам методом сканирования)

№	X	R
1	0,50000000	0,99749499
2	0,50000000	0,99749499
3	0,59375000	0,99973658
4	0,57031250	0,99999988
6	0,59375000	0,99973658

Всего проведено $6 \cdot 3 = 18$ вычислений критерия оптимальности.

Описание алгоритма при выборе переключателя «Метод деления пополам»:

1) обнулим счетчик который считает количество итераций (it:=0);
2) оформим цикл типа **repeat - until** (до тех пор пока):
   - вычислим середину отрезка (ser) и точки стоящие от середины отрезка на ep/2 (G и H);
   - далее вычисляем критерий оптимальности в этих точках (ser, H, G);
   - проверяем условие, если критерии оптимальности от точки G больше критерия оптимальности чем от точки H (R1>R2), то минимальное значение равно середине отрезка (min:=ser);
   - проверяем второе условие, если критерии оптимальности от точки G меньше критерия оптимальности чем от точки H (R1<R2), то максимальное значение равно середине отрезка (max:=ser);
   - считаем количество итераций (it:=it+1);
   - проверяем условие окончания вычислений – разница между максимумом и минимумом должна быть меньше заданной погрешности (until (max-min)<ep);
3) на экран выводится ответ (рис. 7).

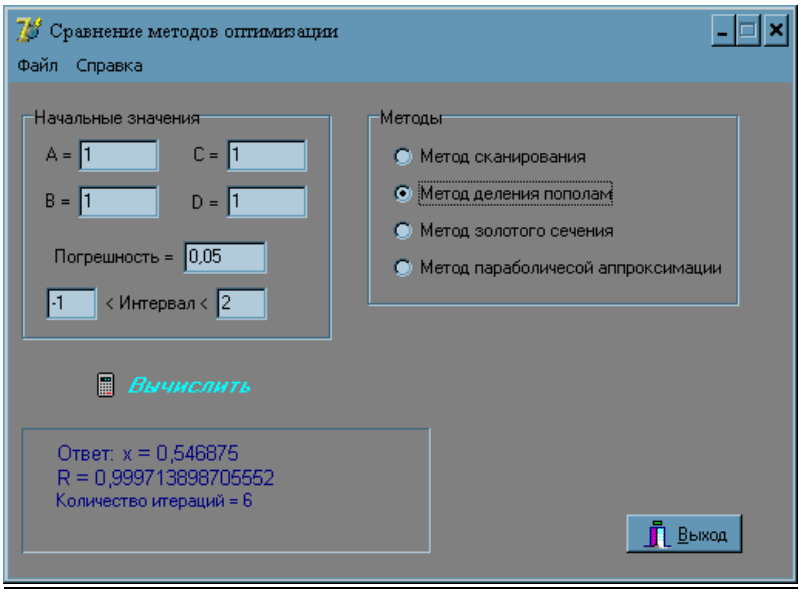


Рис 7. Приведены вычисления контрольной задачи

«Методом деления пополам»

Приведем пример: дана функция $R(x)=D\ sin(Ax^B+C)$, где коэффициенты имеют следующие значения: A=1,0, B=1,0, C=1,0, D=1,0. Найти максимум на интервале: [-1, 2]. Ошибка задается по x: ε=0,05.

Результаты расчетов. Середина отрезка x_0 =0,5000, значение критерия R_0=0,9975, значение

$R(0,5 - \varepsilon/2)$ =R(0,475)=0,97922273, значение R(0,5+ε/2)=R(0,525)=0,9989513. Следовательно, искомый максимум лежит в правой половине отрезка, т.е. теперь отрезком является [0,5; 2].

Далее приводятся только координаты середин отрезков с номером итерации, значения критерия в них и указывается новый отрезок (правый или левый).

$x1$=1,25000000 R1=0,77807320 левый

$x2$=0,87500000 R2=0,95408578 левый

$x3$=0,68750000 R3=0,99319785 левый

$x4$=0,59375000 R4=0,99973658 левый

$x5$=0,54687500 R5=0,99971390

$|x4 - x5| < \varepsilon$, поэтому в качестве решения можно принять любое из этих значений или середину между ними.

Всего восемь раз (4 · 2 = 8) вычислялся критерий оптимальности (не считая вычислений непосредственно в середине отрезка, которые не используются в алгоритме метода).

Описание алгоритма при выборе переключателя «Метод золотого сечения»:

1) обнулим счетчик который считает количество итераций (it:=0);
2) оформим цикл типа **repeat - until** (до тех пор пока):
   - некоторому числу m приравнивается минимум, а n – максимум;
   - делим отрезок на неравные части по правилу золотого сечения (находим точки Lev и Prav);

- вычисляем критерий оптимальности в точках Lev и Prav;
- проверяем условие, при котором минимум отрезка принимает значение Prav – если критерий оптимальности от точки Lev больше критерия оптимальности от точки Prav;
- выводим на экран получившийся критерий оптимальности;
- проверяем второе условие, при котором максимум отрезка принимает значение Lev – если критерий оптимальности от точки Lev меньше критерия оптимальности от точки Prav;
- выводим на экран получившийся критерий оптимальности;
- считаем количество итераций (it:=it+1);
- проверяем условие окончания вычислений – разница между максимумом и минимумом должна быть меньше заданной погрешности (until (max-min)<ep);

4) на экран выводится ответ (рис. 8).

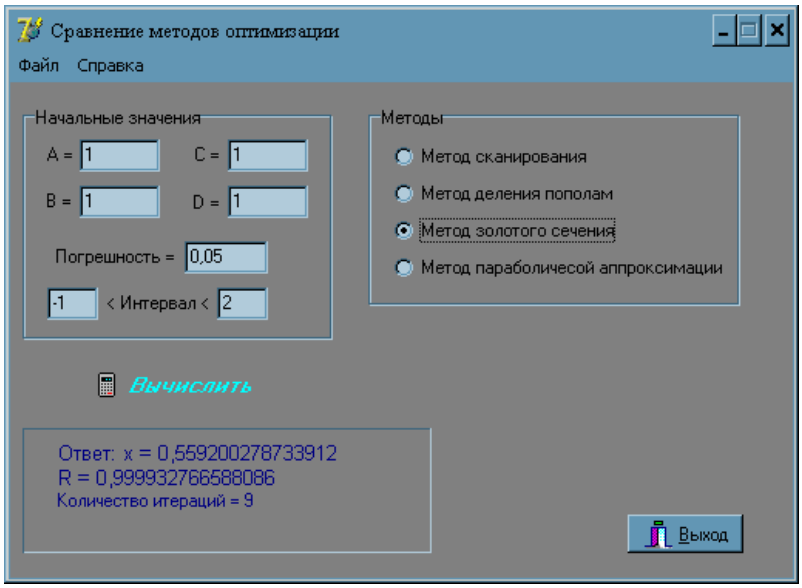


Рис 8. Приведены вычисления контрольной задачи «Методом золотого сечения»

Приведем пример: дана функция $R(x)=D\ sin(Ax^B+C)$, где коэффициенты имеют следующие значения: A=1,0, B=1,0, C=1,0, D=1,0. Найти максимум на интервале: [-1, 2]. Ошибка задается по x: ε=0,05.

Результаты расчетов. Для «запуска» метода найдем две симметричные точки золотого сечения для отрезка [-1, 2]:

$x1$ =0,145898, $x2$ =0,85410197.

Значения критериев в этих точках соответственно $R(x1)$=0,911080, $R(x2)$=0,960136. Следовательно, новым отрезком является [0,145898;2], внутри которого находится максимальное из найденных значений R. Точка золотого сечения для нового отрезка будет $x3$=0,58359214, а $R(x3)$=0,99991813.

Далее приведены только координаты лучших точек при очередном шаге, номер шага и значения критерия в этих точках.

x3=0,58359214 R3=0,99991813

x4=0,58359214 R4=0,99991813

x5=0,58359214 R5=0,99991813

x6=0,58359214 R6=0,99991813

x7=0,58359214 R7=0,99991813

x8=0,55920028 R8=0,99993277

x9=0,55920028 R9=0,99993277

Всего было проведено 10 вычислений критерия оптимальности.

Описание алгоритма при выборе переключателя «Метод параболической аппроксимации»:

1) обнулим счетчик который считает количество итераций (it:=0);
2) обнулим счетчик который считает сколько раз вычисляются коэффициенты параболы (i:=0);
3) рассчитываем середину отрезка (ser:=(max-min)/2+min);
4) оформим цикл типа **repeat - until** (до тех пор пока):

- вычисляем критерий оптимальности в точке min, max и ser (минимум, максимум и середина);
- находим общий делитель (dtl) – необходим чтобы использовать метод Крамера;
- методом Крамера находим коэффициенты X, Y, Z;
- находим коэффициенты параболы C0, C1, C2;
- считаем i:=i+1;
- приравниваем значению минимума значение середины (min:=ser);
- находим значение при котором парабола имеет максимум (Parab[i]:=-C1/(2*C2));
- считаем количество итераций (it:=it+1);
- проверяем условие окончания вычислений (until abs(Parab[i]-Parab[i-1])<ep) – разница между последним и предпоследним значениями должна быть меньше заданной погрешности;

5) выводится ответ (рис. 9).

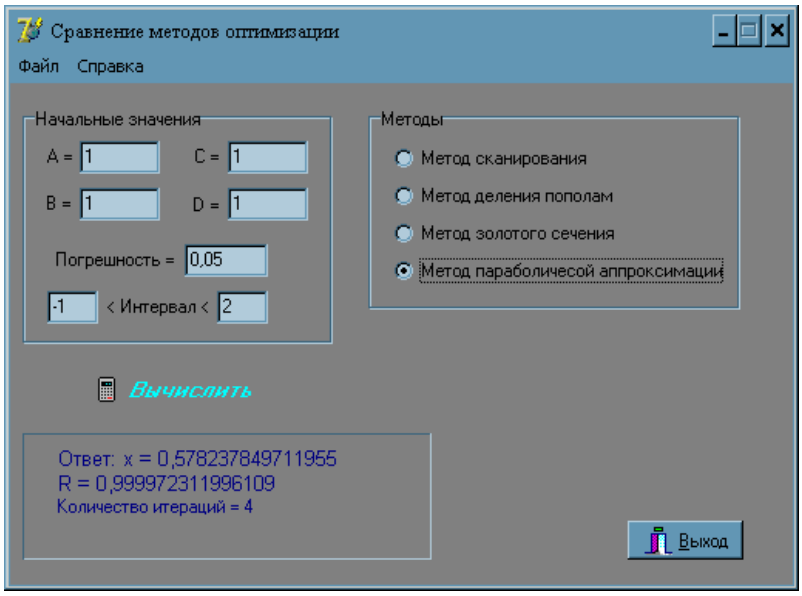


Рис 9. Приведены вычисления контрольной задачи методом «параболической аппроксимации»

Приведем пример: дана функция $R(x)=D\ sin(Ax^B+C)$, где коэффициенты имеют следующие значения: A=1,0, B=1,0, C=1,0, D=1,0. Найти максимум на интервале: [-1, 2]. Ошибка задается по x: ε=0,05.

Результаты расчетов. Первая аппроксимирующая парабола строится по точкам:

$x1$ =-1, $R(-1)$=0; $x2$=0,5 $R(0,5)$=0,9975; $x3$=2,0, $R(2,0)$=0,141120.

Запишем систему уравнений для нахождения коэффициентов параболы:

1*C2+(-1)C1+Co=0,

0,25C2 + 0,5C1 + C0 = 0,9975,

4C2+2C1+C0=0,14112.

Решением этой системы является

C2=-0,41197, C1=0,459012, C0=0,87089.

Находим x, при котором парабола имеет максимум:

$$x=-C1/2C2=-(-0,41197)/(2*0,459012)=0,55709139,$$

при этом R=0,99990609. По этой точке, а также по второй и третьей исходным точкам, лежащим по обе стороны от точки максимума параболы, аналогично строится вторая парабола, максимум которой оказывается в точке

x =0,57823785, а R =0,99997231. Разница между двумя точками максимума менее заданной погрешности, следовательно, можно заканчивать поиск.

В этом методе всего четыре раза вычислялся критерий оптимальности.

Как уже сказано выше в программе присутствует главное меню – в котором всего два пункта: Файл и Справка.

Пункт меню Файл содержит следующие подпункты (рис. 10):

1. Метод сканирования
2. Метод деления пополам
3. Метод золотого сечения
4. Метод параболической аппроксимации
5. Выход

Выбирая один из первых четырех пунктов пользователь выбирает один из методов – после чего он может нажать кнопку Вычислить для получения отве-

та. Если пользователь выберет пятый пункт – Выход – то произойдет выход из программы.

Пункт меню Справка содержит один подпункт – Вызов справки, при выборе этого пункта появится диалоговое окно которое состоит из двух вкладок – Разработчик и О программе, на рис. 11 приведены эти вкладки.

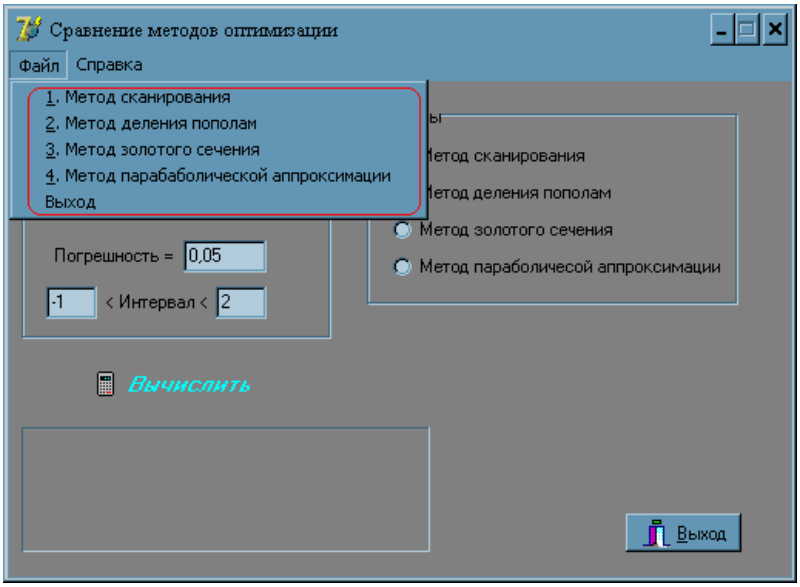


Рис 10. Пункт меню «Файл»

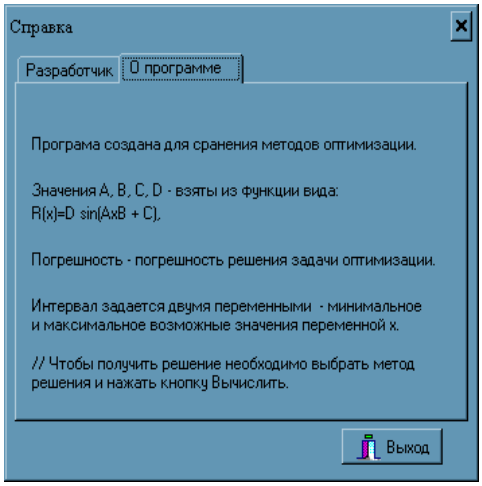


Рис 11. Справка в программе (две вкладки – Разработчик и О программе)

Текст программы с описанием

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, Menus;
type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label5: TLabel;
    Edit5: TEdit;
    Edit6: TEdit;
    Label6: TLabel;
    Edit7: TEdit;
    SpeedButton1: TSpeedButton;
    Label7: TLabel;
    GroupBox2: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    BitBtn1: TBitBtn;
    RadioButton3: TRadioButton;
```

```
    RadioButton4: TRadioButton;
    Label8: TLabel;
    Bevel1: TBevel;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N11: TMenuItem;
    N21: TMenuItem;
    N31: TMenuItem;
    N41: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    procedure SpeedButton1Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure N21Click(Sender: TObject);
    procedure N31Click(Sender: TObject);
    procedure N41Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
  max, min: real;          // Переменные используемые в программе
  A,B,C,D,ep,int,per,maxmn,newP,R: real;
  R0,R1,R2,ser,dlt,dltX,dltY,dltZ,C0,C1,C2: real;
  x,mass: array [1..25] of real;
```

```
  parab: array [0..99] of real;
  G,H,J,K: real;
  R_Lev, R_Prav: real;

implementation

uses Unit4;

{$R *.dfm}

function Scan(A,B,C,D,x: real): real;  // Вычисляется функция вида
                                       //R(x)=D sin(AxB + C)
var PRM: real;
    i: integer;
begin
  PRM:=1;
  for i:=1 to trunc(B) do PRM:=PRM*x;
  Scan:=D*sin(A*PRM+C);
end;

procedure TForm1.SpeedButton1Click(Sender: TObject); // Действие при нажатии
                                                     // кнопки Вычислить

var i,k,it: integer;            // Переменные используемые в программе
     m,n,Lev,Prav: real;
begin
try;
  ep:=strtofloat(Edit7.Text);      // ep (погрешность) из строки переводится в число
    if ep<0 then exit;
      A:=strtofloat(Edit1.Text);    // A из строки переводится в число
```

```
    B:=strtofloat(Edit2.Text);    // B из строки переводится в число
    C:=strtofloat(Edit3.Text);    // C из строки переводится в число
    D:=strtofloat(Edit4.Text);    // D из строки переводится в число
    min:=strtofloat(Edit5.Text);  // min из строки переводится в число
    max:=strtofloat(Edit6.Text);  // max из строки переводится в число
  if min>max then exit;
except
ShowMessage('Некорректное значение!');
exit;
end;

If RadioButton1.Checked then     // Вычисляется "Метод Сканирования"
 begin
  it:=0;     // Количество итераций равно 0
    repeat
      Int:=(max-min)/4;   // Отрезок разбивается на 4 части
      PER:=min;           // Некоторой переменной присваивается значение мини-
мума



      for i:=1 to 3 do    // Находим массив чисел x[i]
        begin
          PER:=PER+Int;
          x[i]:=PER;
        end;

for i:=1 to 3 do MASS[i]:=(Scan(A,B,C,D,x[i]));  // На каждом интервале находим
                                                 // значение функции

      maxmn:=mass[1];
```

```
      k:=1;
     for i:=2 to 3 do
     if maxmn<mass[i] then      // Находим максимальное значение
     begin
      maxmn:=mass[i];
      k:=i;                     // Индекс максимального значения
     end;
     newP:=x[k];
     min:=newP-int;             // Находим минимум
     max:=newP+int;             // Находим максимум

    it:=it+1;                   // Считается количество итераций
    until (max-min)<ep;         // Условие окончания вычислений

  Label7.Caption:='Ответ: x = '+  floattostr(newP) + #10#13 +
  'R =' + floattostr(mass[k]);          // Выводится ответ
  Label8.Caption:= 'Количество итераций = '+inttostr(it);    // Выводится
                                                        // количество итераций
end;

If RadioButton3.Checked then      //  Вычисляется "Метод деления пополам"
  begin
    it:=0;                      // Количество итераций равно 0
    repeat
      ser:=(max+min)/2;         // Вычисляется середина отрезка (ser) и точки
                                // стоящие от середины отрезка на ep/2 (G и H)
      G:=(ser+ep/2);
      H:=(ser-ep/2);

      R0:=scan(A,B,C,D,ser);      // вычисляется критерий оптимальности в точке
```

```
                                    // ser
      R1:=scan(A,B,C,D,G);        // вычисляется критерий оптимальности в точке
                                    // G
      R2:=scan(A,B,C,D,H);        // вычисляется критерий оптимальности в точке
                                    // H

      if R1>R2 then begin         // Условие при котором  середина равняется
                                  // минимуму
      min:=ser;
      end;
      if R1<R2 then begin         // Условие при котором  середина равняется
                                  // максимуму
      max:=ser;
      end;

      it:=it+1;         // считается количество итераций
    until (max-min)<ep;   // условие окончания вычислений

  Label7.Caption:='Ответ: x = '+floattostr(ser)+#10#13+
  'R = '+Floattostr(R0);              // На экран выводится ответ (ser, R0)
  Label8.Caption:= 'Количество итераций = '+inttostr(it);     // Выводится
                                                              // количество итераций
end;

If RadioButton4.Checked then     // Вычисляется метод "Золотого сечения"
  begin
    it:=2;    // Количество итераций равно 2 (т.к. две точки даны)
  repeat
     m:=min;    // m приравнивается к минимуму
     n:=max;    // n приравнивается к максимуму
```

```
    Lev:=m+(n-m)*((sqrt(5)-1)/2);    // Отрезок делится на неравные части по
    Prav:=n-(n-m)*((sqrt(5)-1)/2);     // Правилу золотого сечения (точки Lev и
                                       // Prav)

    R_Lev:=scan(A,B,C,D,Lev);          // Вычисляется критерий оптимальности в
                                        // точке Lev
    R_Prav:=scan(A,B,C,D,Prav);        // вычисляется критерий оптимальности в
                                       //точке Prav

    if R_Lev>R_Prav then               // Условие при котором  минимум отрезка
                                       // принимает значение Prav
    begin
      min:=Prav;
      Label7.Caption:='Ответ: x = '+floattostr(Lev)+#10#13+
      'R = '+Floattostr(R_Lev);        // На экран выводится ответ (Lev, R_Lev)
    end;

    if R_Lev<R_Prav then               // Условие при котором  максимум отрезка
                                       // принимает значение Lev
    begin
      max:=Lev;
      Label7.Caption:='Ответ: x = '+floattostr(Prav)+#10#13+
      'R = '+Floattostr(R_Prav);       // На экран выводится ответ (Prav, R_Prav)
    end;

   it:=it+1;          // Считается количество итераций
until (max-min)<ep;   // Условие окончания вычислений
Label8.Caption:= 'Количество итераций = '+inttostr(it);     // Выводится
                                                            // количество итераций
```

```
end;

if RadioButton2.Checked then     // Вычисляется метод "Параболической
                                 // аппроксимации"
begin
  it:=0;                    // Количество итераций равно 0
  i:=0;
  ser:=(max-min)/2+min;        // Рассчитывается середина отрезка
  repeat
    R0:=scan(A,B,C,D,min);     // Вычисляется критерий оптимальности в точке
min
    R1:=scan(A,B,C,D,ser);       // Вычисляется критерий оптимальности в точке
ser
    R2:=scan(A,B,C,D,max);     // Вычисляется критерий оптимальности в точке
max

    dlt:=(sqr(min)*(ser-max)-min*(sqr(ser)-sqr(max))+(sqr(ser)*max-ser*sqr(max)));
    // Находим общий делитель

    dltX:=R0*(ser-max)-min*(R1-R2)+(R1*max-R2*ser);
    dltY:=sqr(min)*(R1-R2)-R0*(sqr(ser)-sqr(max))+(sqr(ser)*R2-R1*sqr(max));
    dltZ:=sqr(min)*(ser*R2-max*R1)-min*(sqr(ser)*R2-
R1*sqr(max))+R0*(sqr(ser)*max-ser*sqr(max));
    // Методом Крамера находим X, Y, Z.

    C2:=dltX/dlt;    // Находим коэффициенты параболы C0, C1, C2
    C1:=dltY/dlt;
    C0:=dltZ/dlt;

    i:=i+1;
```

```
    min:=ser;
    Parab[i]:=-C1/(2*C2);   // Значение при котором парабола имеет максимум
    ser:=Parab[i];
    if i=1 then Parab[i-1]:=-Parab[i];

    it:=it+1;         // Считается количество итераций
  until abs(Parab[i]-Parab[i-1])<ep;     // Условие окончания вычислений

R:=Scan(A,B,C,D,Parab[i]);
Label7.Caption:='Ответ: x = '+floattostr(Parab[i])+#10#13+'R = ' + floattostr(R);
Label8.Caption:= 'Количество итераций = '+inttostr(it);    // Выводится количество
итераций
end;

end;

procedure TForm1.N3Click(Sender: TObject);
begin
  Close;    // Выход
end;

procedure TForm1.N4Click(Sender: TObject);
begin
  PagesDlg.Show;     // Вызов справки
end;

procedure TForm1.N11Click(Sender: TObject);
begin
RadioButton1.Checked:=true;        // Выбирается "Метод сканирования"
end;
```

```
procedure TForm1.N21Click(Sender: TObject);
begin
RadioButton3.Checked:=true;        // Выбирается "Метод деления пополам"
end;

procedure TForm1.N31Click(Sender: TObject);
begin
RadioButton4.Checked:=true;        // Выбирается "Метод золотого сечения"
end;

procedure TForm1.N41Click(Sender: TObject);
begin
RadioButton2.Checked:=true;        // Выбирается "Метод параболической ап-
проксимации"
end;
end.
```

2.4 Описание процесса отладки программы

Отладка программы представляет собой нахождение ошибок используя контрольные примеры. В данной программе ошибки могут быть если введены неточные (некорректные) значения в полях ввода, чтобы этого не произошло используется система *try…except…end*.

```
begin
try;
        ep:=strtofloat(Edit7.Text);      // ep (погрешность) из строки переводится в
      число
if  ep<0 then  exit;
```

```
    A:=strtofloat(Edit1.Text);    // A из строки переводится в число
    B:=strtofloat(Edit2.Text);    // B из строки переводится в число
    C:=strtofloat(Edit3.Text);    // C из строки переводится в число
    D:=strtofloat(Edit4.Text);    // D из строки переводится в число
    min:=strtofloat(Edit5.Text);  // min из строки переводится в число
    max:=strtofloat(Edit6.Text);  // max из строки переводится в число
   if min>max then exit;
except
ShowMessage('Некорректное значение!');
exit;
end;
```

Наибольшее затруднение в написании программы было в методе золотого сечения – пришлось долго думать как вывести ответ – проблема в том, что если критерий оптимальности от левой точки больше критерия оптимальности чем от правой точки, то минимумом должна стать правая точка, а если критерий оптимальности от левой точки меньше критерия оптимальности чем от правой точки, то максимумом становится левая точка – и что выводить правую точку (минимум) или левую (максимум), решение в конце концов стало простым – выводить значение обеих точек на одну метку (мы увидим самое последние значение, т.е. правильное – а это уже не важно левая это точка или правая).

В процессе отладки возникали следующие ошибки:

- Undeclared identifier – используется переменная, не объявленная в разделе var;

- Missing operator or semicolon – после инструкции не поставлена точка с запятой;

- ”)” expected - забыл закрыть скобку;

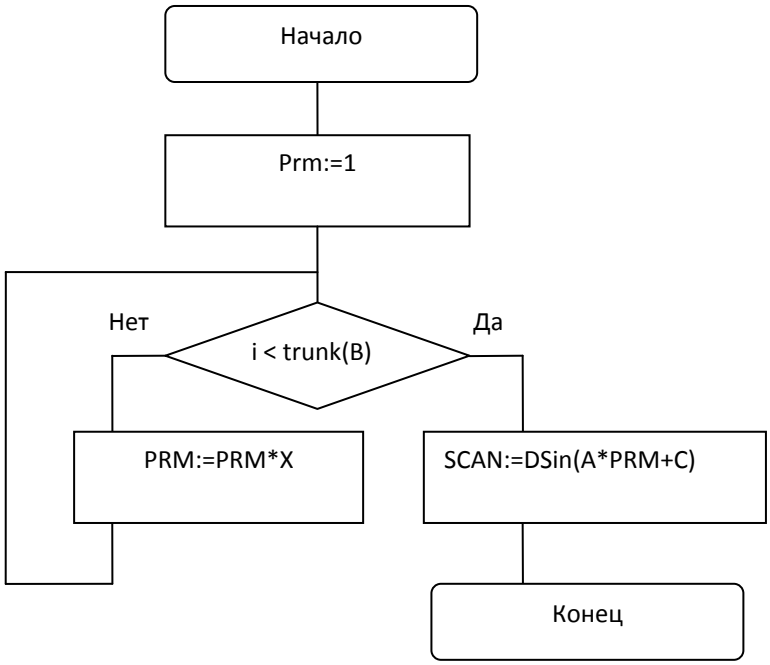
- Incompatible types – несоответствие типов и т.д.

БЛОК-СХЕМЫ

1. Блок-схема function Scan

function Scan(A,B,C,D,x: real): real – Вычисляется функция вида

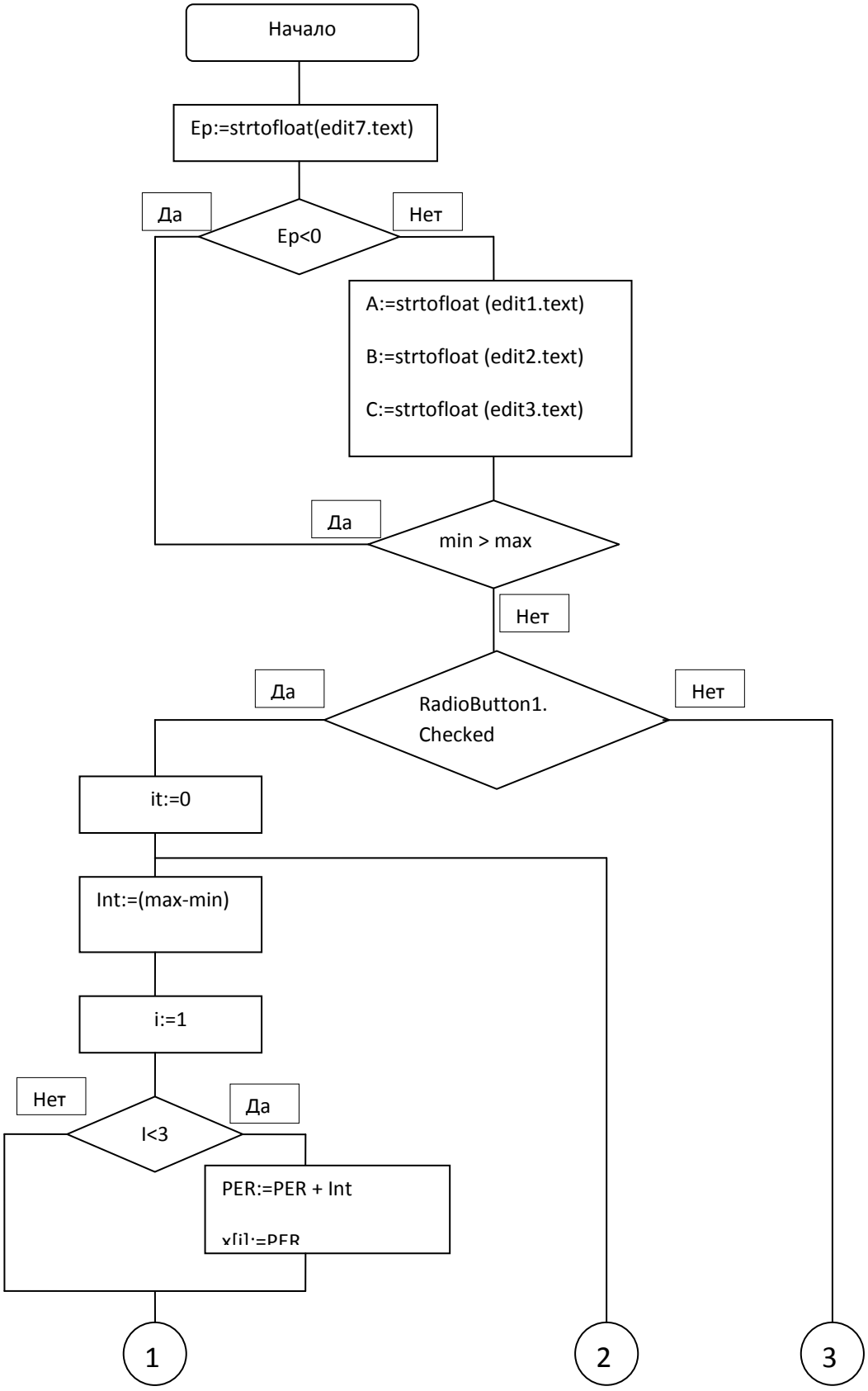
R(x)=D sin(AxB + C)

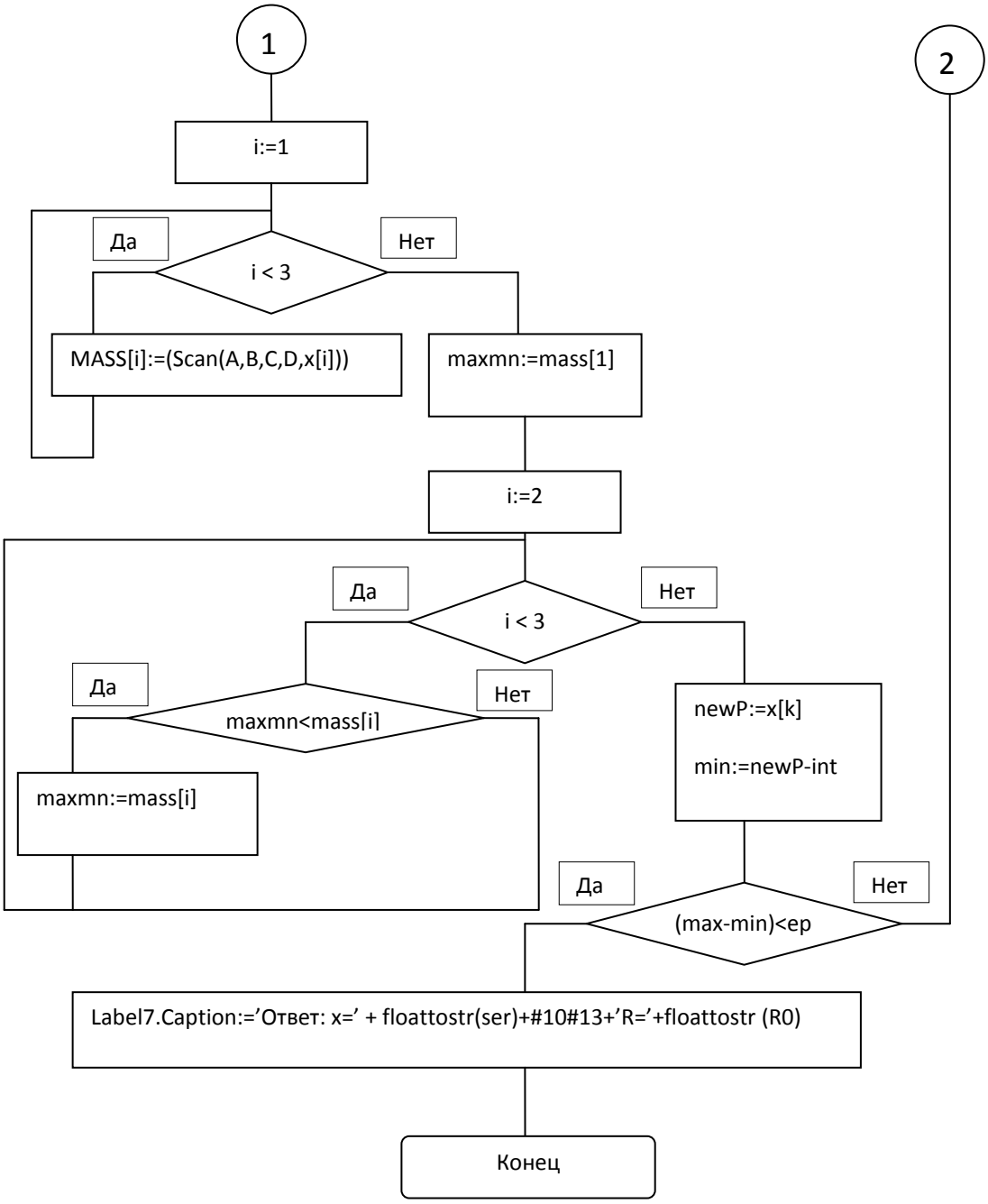


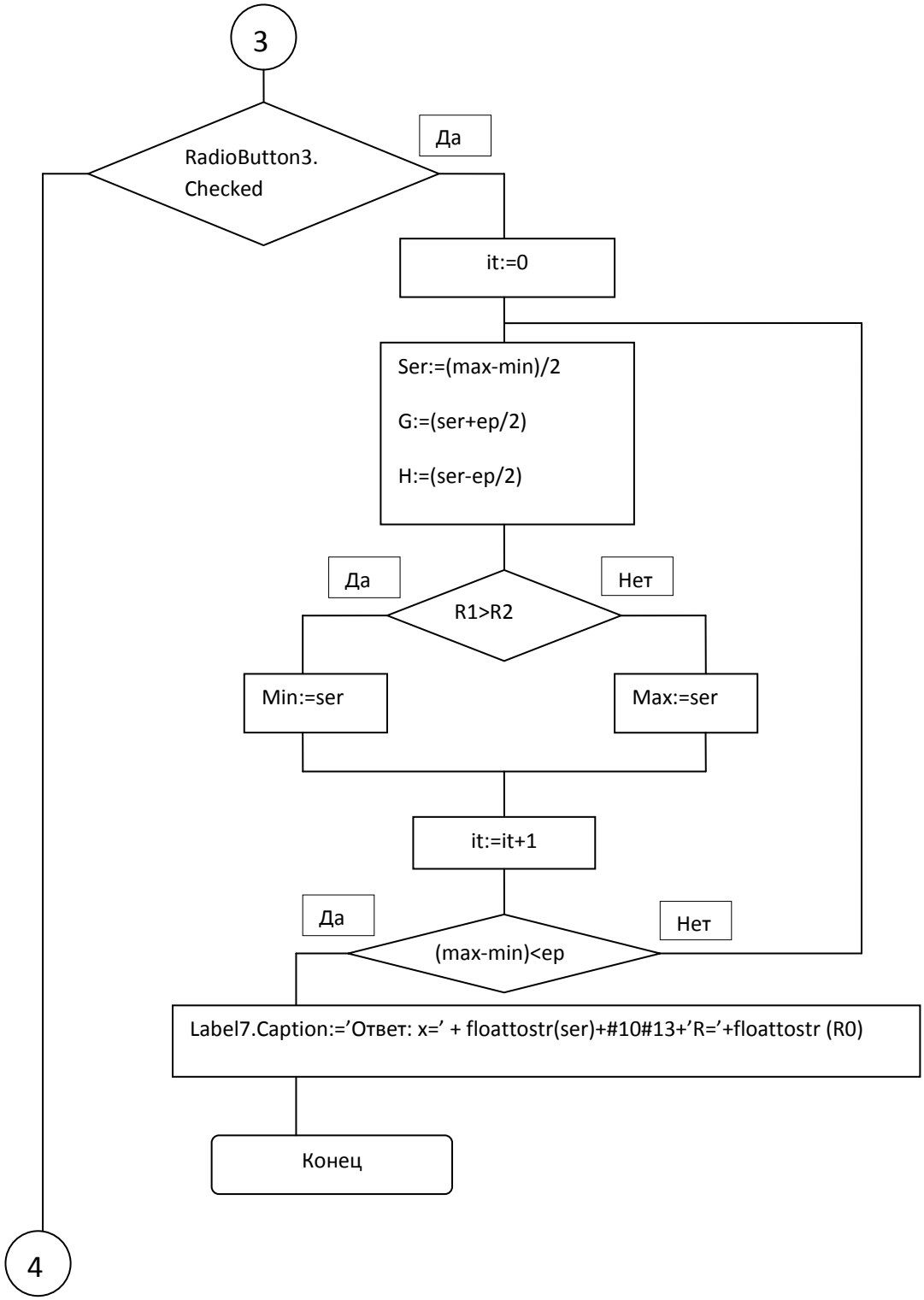
2. Блок-схема procedure TForm1.SpeedButton1Click

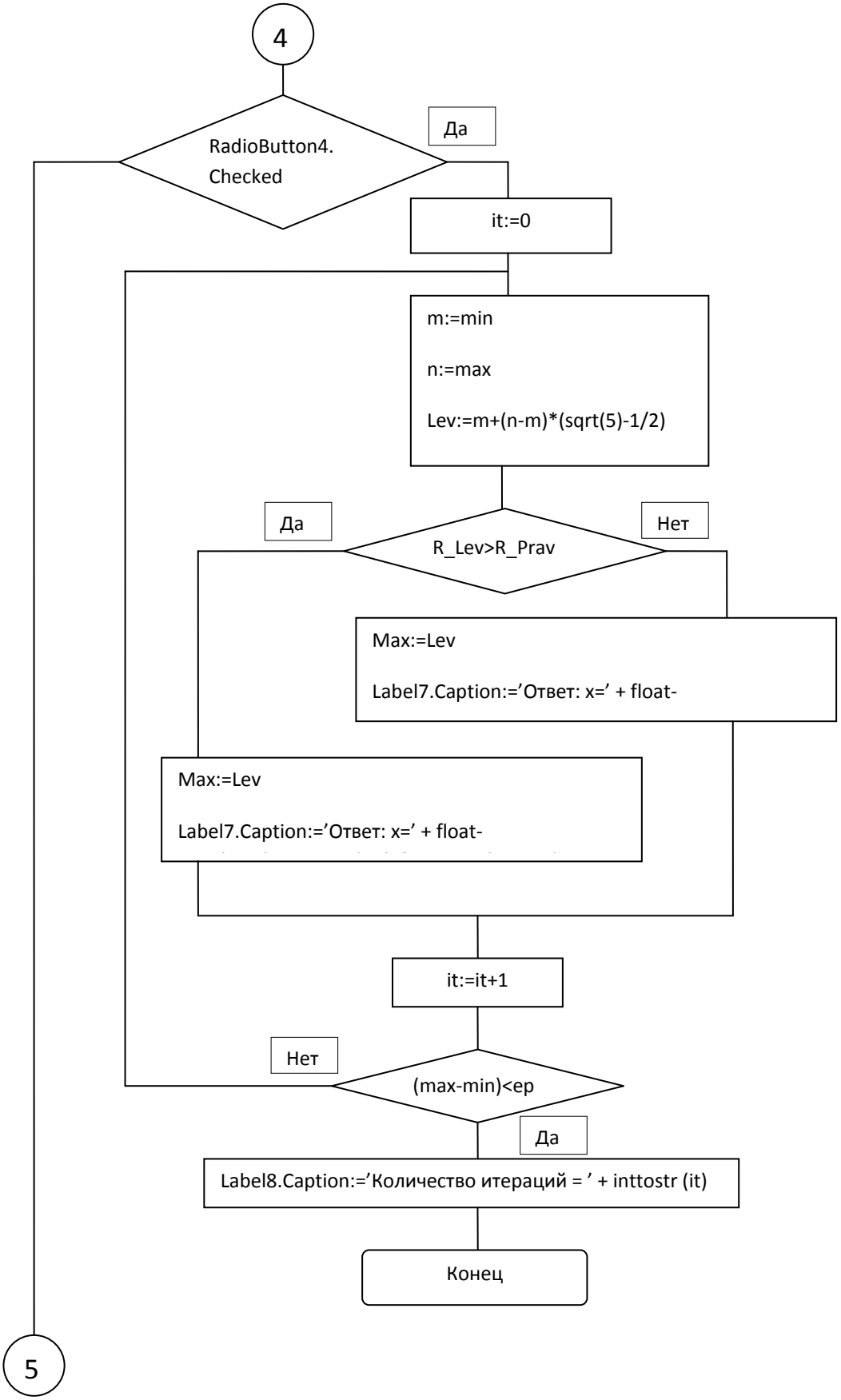
procedure TForm1.SpeedButton1Click(Sender: TObject);

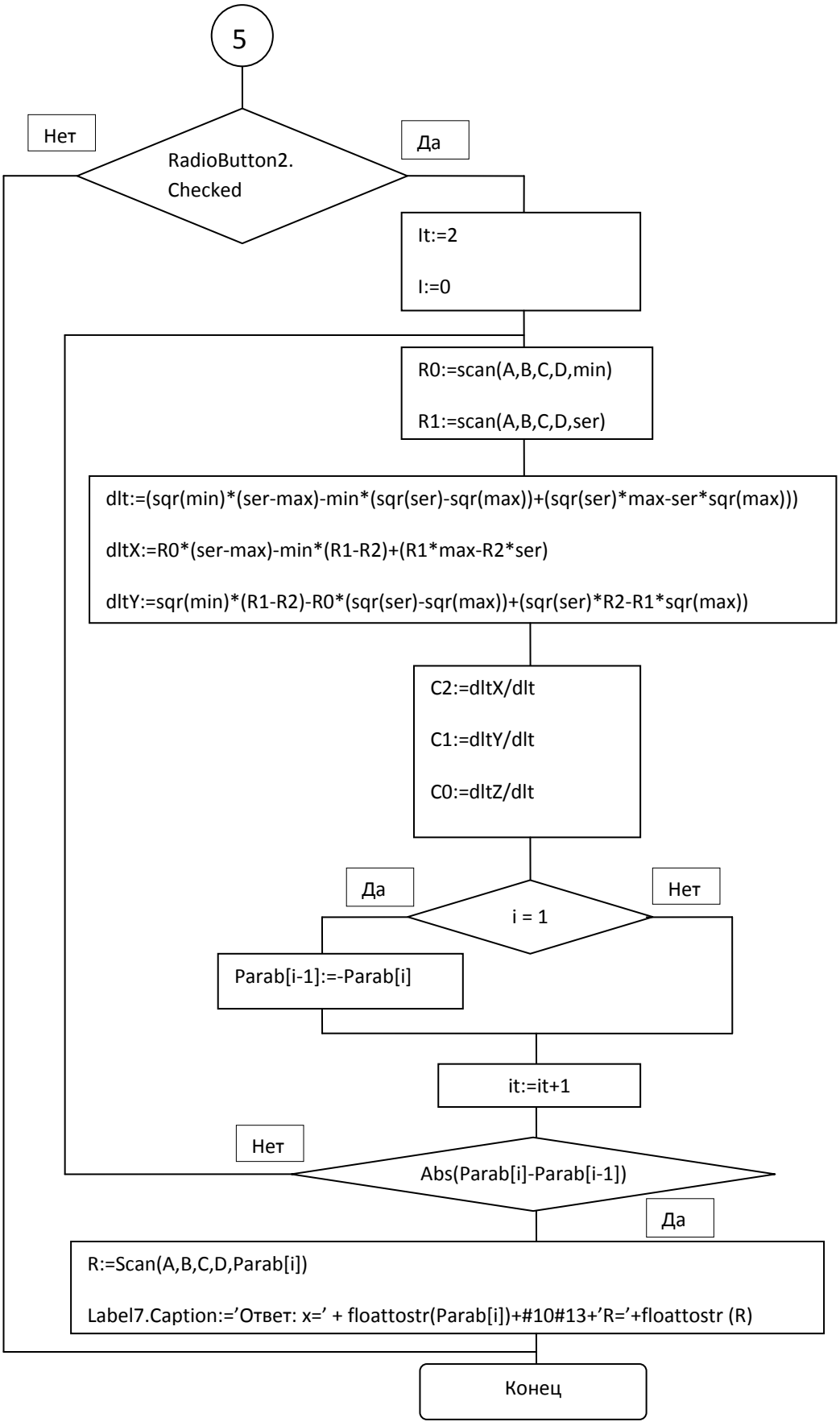
//Действие при нажатии кнопки "Вычислить"




1
2
i:=1
Да
i < 3
Нет
MASS[i]:=(Scan(A,B,C,D,x[i]))
maxmn:=mass[1]
i:=2
Да
i < 3
Нет
Да
maxmn<mass[i]
Нет
maxmn:=mass[i]
newP:=x[k]
min:=newP-int
Да
(max-min)<ep
Нет
Label7.Caption:=’Ответ: x=’ + floattostr(ser)+#10#13+’R=’+floattostr (R0)
Конец


3
RadioButton3.
Checked
Да
it:=0
Ser:=(max-min)/2
G:=(ser+ep/2)
H:=(ser-ep/2)
Да
Нет
R1>R2
Min:=ser
Max:=ser
it:=it+1
Да
Нет
(max-min)<ep
Label7.Caption:=’Ответ: x=’ + floattostr(ser)+#10#13+’R=’+floattostr (R0)
Конец
4


4
RadioButton4.
Checked
Да
it:=0
m:=min
n:=max
Lev:=m+(n-m)*(sqrt(5)-1/2)
Да
R_Lev>R_Prav
Нет
Max:=Lev
Label7.Caption:=’Ответ: x=’ + float-
Max:=Lev
Label7.Caption:=’Ответ: x=’ + float-
it:=it+1
Нет
(max-min)<ep
Да
Label8.Caption:=’Количество итераций = ’ + inttostr (it)
Конец
5


5
Нет
RadioButton2.
Checked
Да
It:=2
I:=0
R0:=scan(A,B,C,D,min)
R1:=scan(A,B,C,D,ser)
dlt:=(sqr(min)*(ser-max)-min*(sqr(ser)-sqr(max))+(sqr(ser)*max-ser*sqr(max)))
dltX:=R0*(ser-max)-min*(R1-R2)+(R1*max-R2*ser)
dltY:=sqr(min)*(R1-R2)-R0*(sqr(ser)-sqr(max))+(sqr(ser)*R2-R1*sqr(max))
C2:=dltX/dlt
C1:=dltY/dlt
C0:=dltZ/dlt
Да
i = 1
Нет
Parab[i-1]:=-Parab[i]
it:=it+1
Нет
Abs(Parab[i]-Parab[i-1])
Да
R:=Scan(A,B,C,D,Parab[i])
Label7.Caption:=’Ответ: x=’ + floattostr(Parab[i])+#10#13+’R=’+floattostr (R)
Конец

ЗАКЛЮЧЕНИЕ

Цель данной диссертационной работы была создать программу «Сравнение методов оптимизации» предназначенную для решения задач оптимизации численными методами. Всего в программе представлено четыре метода:

1. метод сканирования,
2. метод деления пополам,
3. метод золотого сечения,
4. метод параболической аппроксимации.

В ответе выводятся: середина отрезка, критерий оптимальности и количество итераций.

Для реализации этой программы был использован язык Delphi 7, который комбинирует в себе несколько важнейших технологий: высокопроизводительный компилятор в машинный код, объектно-ориентированная модель компонент и визуальное построение приложений из программных прототипов.

ЛИТЕРАТУРА

1. Ашманов С.А., Тимохов А.В. Теория оптимизации в задачах упражнениях. - М.: Наука,1991.
2. Никита Культин – «Основы программирования в Delphi 7», С.- П.: «БХВ Санкт-Петербург», 2003 г.
3. Шауцукова Л.З. – «Информатика», М.: Просвещение, 2000.
4. Васильков Ю. В., Василькова Н. Н. «Компьютерные технологии вычислений», М.: Финансы и статистика, 1999 г.
5. Бережная Е. В., Бережной В. И. «Математические методы моделирования в экономических системах»
6. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. – М.: Изд-во Бином. Лаборатория знаний, 2011. – 640 с.
7. Бахвалов Н. С., Корнев А. А., Чижонков Е. В. Численные методы. Решения задач и упражнения. – М.: Изд-во Дрофа, 2009. – 400 с.
8. Бахвалов Н. С., Лапин А. В., Чижонков Е. В. Численные методы в задачах и упражнениях. – М.: Изд-во Бином. Лаборатория знаний, 2010. – 240 с.
9. Вержбицкий В. М. Основы численных методов. – М.: Высшая школа, 2009. – 848 с.
10. Волков Е. А. Численные методы. – М.: Изд-во Лань, 2004. – 256 с.
11. Жидков В.Н. Вычислительная математика. – М, Академия, 2010. – 208 с.
12. Зализняк В.Е. Основы научных вычислений. Введение в численные методы для физиков. – Изд-во Едиториал УРСС, 2002. – 296 с.
13. Калиткин Н.Н. Численные методы. – С.Пб.: Изд-во БХВ-Петербург, 2011. – 592 с.
14. Лапчик М. П., Рагулина М. И., Хеннер Е. К. Численные методы. – М.: Академия, 2009. – 384 с.
15. Марчук Г.И. Методы вычислительной математики. – М.: Изд-во Лань, 2010. – 608 с.
16. Панокова Т.А. Численные методы. – М.: Изд-во Либроком, 2010. – 226 с.

17. Пантина И. В., Синчуков А. В. Вычислительная математика. – М, 2010. – 176 с.
18. Петров И. Б., Лобанов А. И. Лекции по вычислительной математике. – М.: Изд-во Бином. Лаборатория знаний, Интернет-университет ин., 2009. – 528 с.
19. Протасов И. Д. Лекции по вычислительной математике. – М.: Изд-во Гелиос АРВ, 2004. – 184 с.
20. Рено Н.Н. Численные методы. – М.: Изд-во КДУ, 2007. – 100 с.,
21. Рябенький В.С. Введение в вычислительную математик. – М,: Изд-во ФИЗМАТЛИТ, 2008. – 286 с.
22. Самарский А.А. Введение в численные методы. – М.: Изд-во Лань, 2009. - 288 с.
23. Самарский А. А., Вабищевич П. Н., Самарская Е. А. Задачи и упражнения по численным методам. – М.: Изд-во Либроком, 2009. – 208 с.
24. Срочко В.А. Численные методы. Курс лекции. – М.: Изд-во Лань, 2010. – 208 с.
25. Турчак Л. И., Плотников П. В. Основы численных методов. – М.: ФИЗМАТЛИТ, 2002. – 304 с.
26. Ульяницкий И.В. Введение в численные методы. – М., 2005.
27. Фаддеев М. А., Марков К. А. Основные методы вычислительной математики. – М.: Изд-во Лань, 2008. – 160 с.
28. Численные методы. Сборник задач. Под редакцией У. Г. Пирумова. – М.: Изд-во Дрофа, 2007. – 144 с.
29. Шахов Ю. Н., Деза Е. И. Численные методы. – М.: Изд-во Либроком, 2010. – 248 с.
30. www.ziyonet.uz
31. www.samdu.uz
32. www.edu.uz
33. www.edu.ru