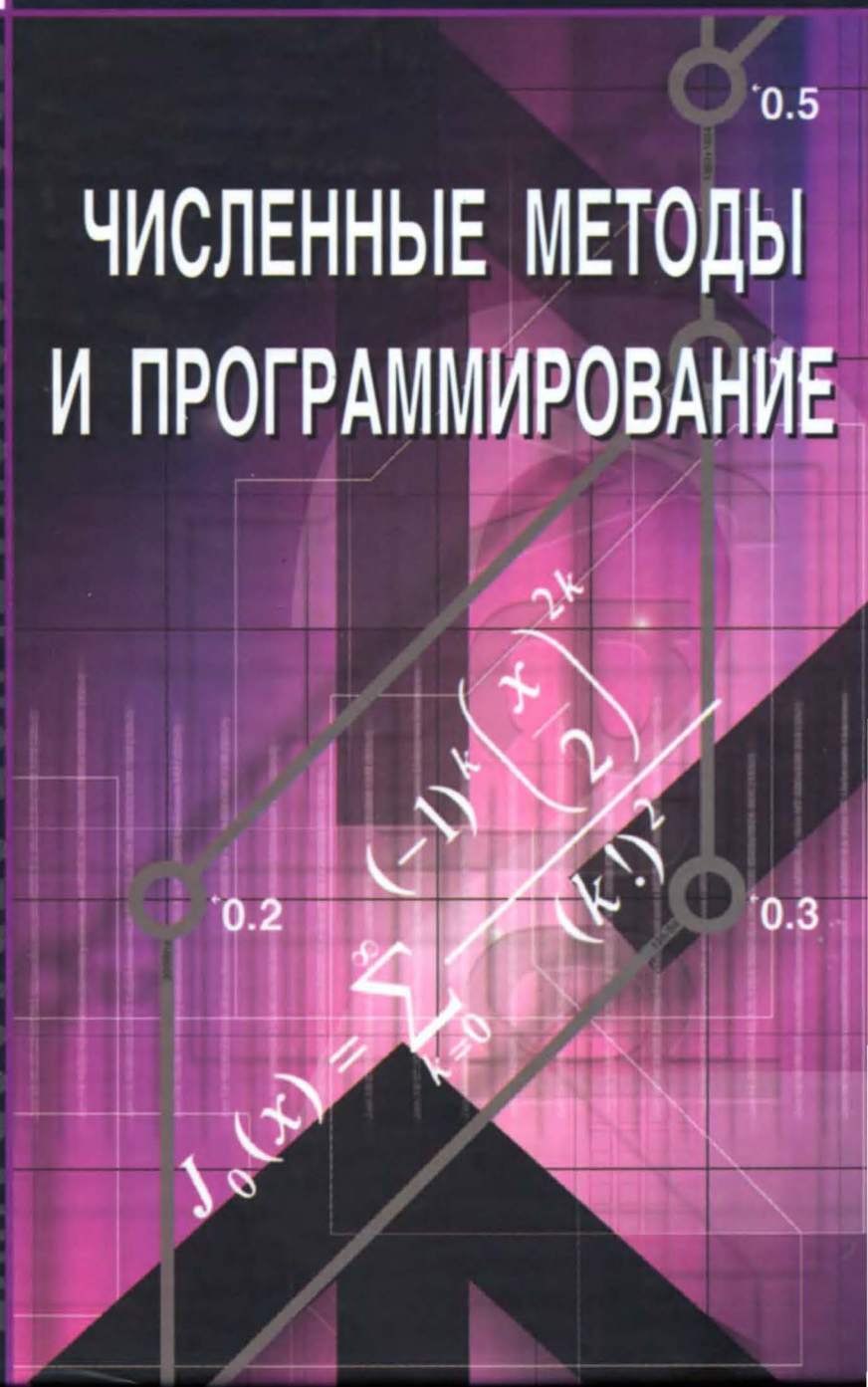
ИЗДАТЕЛЬСКИЙ ДОМ ФОРУМ

*В.Д. Колдаев*

# ЧИСЛЕННЫЕ МЕТОДЫ И ПРОГРАММИРОВАНИЕ

ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ



В. Д. Колдаев

# ЧИСЛЕННЫЕ МЕТОДЫ И ПРОГРАММИРОВАНИЕ

Под ред. проф. Л. Г. Гагариной

*Допущено Министерством образования Российской Федерации в качестве учебного пособия для студентов учреждений среднего профессионального образования, обучающихся по группе специальностей 2200 «Информатика и вычислительная техника»*

Москва
ИД «ФОРУМ» — ИНФРА-М
2009

УДК 004(075.32)
ББК 32.973-018я723
К60

*Рецензенты:*
доктор технических наук, профессор кафедры информатики
и программного обеспечения вычислительных
систем МГИЭТ (ТУ) *О. И. Лисов*;
кандидат технических наук, начальник отдела
ООО «Кедах Электроникс» *С. А. Костина*

**Колдаев В. Д.**

К60 Численные методы и программирование: учебное пособие / Под ред. проф. Л. Г. Гагариной. — М.: ИД «ФОРУМ»: ИНФРА-М, 2009. — 336 с.: ил. — (Профессиональное образование).

ISBN 978-5-8199-0333-9 (ИД «ФОРУМ»)
ISBN 978-5-16-003148-4 (ИНФРА-М)

Предложен широкий круг алгоритмов, сгруппированных по темам, для решения типичных задач, встречающихся в инженерных расчетах численными методами. Прикладная направленность отличает пособие от большинства учебников по численным методам, в которых, как правило, изложение ограничивается только теорией. Описание методов ориентировано на конкретную реализацию соответствующих алгоритмов на ПЭВМ. Пособие содержит большое количество заданий для самостоятельного решения. Даны рекомендации методологического плана по изучению тем в рамках курса математического моделирования.

Для студентов, обучающихся по направлению и специальностям программного обеспечения вычислительной техники и автоматизированных систем, прикладной математики и обработки информации, будет полезно широкому кругу специалистов по компьютерному моделированию.

УДК 004(075.32)
ББК 32.973-018я723

ISBN 978-5-8199-0333-9 (ИД «ФОРУМ»)
ISBN 978-5-16-003148-4 (ИНФРА-М)

# Введение

Главная особенность обучения основам численных методов, которая все отчетливее проявляется в последние годы, связана с интенсификацией процессов использования различных специализированных математических пакетов и систем программирования вычислительных методов как инструмента решения прикладных задач. Широкое внедрение математических методов в самые разнообразные сферы профессиональной деятельности человека требует создания и использования инструмента математического моделирования для решения вычислительных задач. Современные численные методы в совокупности с возможностью их автоматизации при использовании персональных компьютеров и превращаются в такой рабочий инструмент для решения задач научного, технического, экономического характера и др. Развитие алгоритмов и программных средств их реализации ставит задачу обучения эффективным навыкам использования численных методов для решения практических задач исследований.

Все известные издания посвящены либо изложению теоретических аспектов соответствующих разделов вычислительной математики, т. е. в них отсутствуют указания на конкретные приложения и практически не включены примеры конкретных алгоритмов и текстов программ, либо представляют собой сборники чисто алгоритмов и программ без какого бы то ни было введения в теорию используемых методов вычислений. Все это создает искусственный барьер между специалистами в области вычислительной математики и пользователями-нематематиками, которым по роду своей деятельности в той или иной сфере (естественные и экономические науки, техника, система образования и т. п.) приходится решать задачи с помощью соответствующих методов вычислений.

К сожалению, в литературе часто не уделяется внимание уже разработанным и признанным весьма эффективными алгоритмам, которые широко использовались для решения прикладных задач на персональных электронных вычислительных машинах (ПЭВМ).

Данное учебное пособие устраняет отмеченные недостатки путем систематизированного изложения алгоритмов с краткими предварительными сведениями по теории используемых методов по всем включенным в издание разделам вычислительной математики.

Наряду с наиболее эффективными алгоритмами, разработанными в последнее время, книга содержит оригинальные результаты, которые тщательно отбирались с учетом их эффективности и оптимальности.

Материал разбит на тематические части, соответствующие традиционно выделяемым областям. Каждая часть содержит краткое введение в теорию с постулированием основных положений и серию алгоритмов с их программной реализацией на языках Turbo Pascal и C++.

Книга построена таким образом, чтобы пользователь, работающий в конкретной предметной области и не являющийся специалистом непосредственно в вычислительной математике, мог без особого труда выбрать наиболее эффективный метод численного решения задачи. Методика подбора и изложения материала позволит использовать это издание в качестве настольной книги-справочника по эффективным алгоритмам и программам вычислительной математики.

В пособии не ставится задача фундаментальной подготовки в области профессионального программирования, так как в большинстве случаев для решения задач обработки эксперимента и математического моделирования процессов уже существуют готовые программные комплексы. Однако студенты должны иметь ясное представление об основных методах приближенных вычислений и границах их применимости. Это позволит, во-первых, выбирать подходящую для решения конкретной задачи программу, а во-вторых, правильно интерпретировать получаемые результаты.

Целью учебного пособия является ознакомление студентов с математическими основами численных методов решения задач и применение этих методов для решения проблем математического моделирования систем и процессов.

Рекомендуется учащимся лицеев, гимназий и школ, колледжей и техникумов, студентам младших курсов институтов и университетов, всем изучающим и преподающим программирование.

# Глава 1
# ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ПО ИНФОРМАТИКЕ

## 1.1. Основные понятия вычислительной математики

**Вычислительная математика** — раздел математики, включающий круг вопросов, связанных с использованием электронных вычислительных машин. В более узком понимании вычислительная математика — теория численных методов и алгоритмов решения типовых математических задач.

К задачам вычислительной математики относят:

- решение линейных уравнений;
- нахождение собственных значений и векторов матрицы;
- решение нелинейных алгебраических уравнений;
- решение систем нелинейных алгебраических уравнений;
- решение дифференциальных уравнений;
- решение систем дифференциальных уравнений;
- решение интегральных уравнений;
- задачи аппроксимации, интерполяции, экстраполяции.

В вычислительной математике можно выделить следующие разделы:

- анализ математических моделей, связанный с применением ПЭВМ в различных областях научной и практической деятельности;
- разработка методов и алгоритмов решения типовых математических задач, возникающих при исследованиях математических моделей;
- теория и практика программирования задач для ПЭВМ.

Анализ математических моделей включает в себя изучение постановки задачи, выбор модели, анализ и обработку входной информации, численное решение математических задач, возникающих в связи с исследованием модели, анализ результатов вы-

числений, и, наконец, вопросы, связанные с реализацией полученных результатов. В табл. 1.1 показана классификация математических моделей по различным признакам.

*Таблица 1.1.* **Классификация математических моделей**

| Признаки классификации | Виды математических моделей |
|---|---|
| Принадлежность к иерархическому уровню | Модели микроуровня<br>Модели макроуровня<br>Модели метауровня |
| Характер отображаемых свойств объекта | Структурные<br>Функциональные |
| Способ представления свойств объекта | Аналитические<br>Алгоритмические<br>Имитационные |
| Способ получения модели | Теоретические<br>Эмпирические |
| Особенности поведения объекта | Детерминированные<br>Вероятностные |

Рассматривая математический анализ явления как своего рода теоретический эксперимент, из общих и достаточно естественных соображений процесс *математического моделирования* разбивается на несколько этапов.

**1. Формулировка математической модели явления.** Математическая модель любого изучаемого явления по причине его чрезвычайной сложности должна охватывать важнейшие для рассматриваемой задачи стороны процесса, его существенные характеристики и формализованные связи, подлежащие учету. Как правило, *математическая модель* изучаемого физического явления формулируется в виде уравнений математической физики. На этой стадии анализа — это существенно нелинейные, многомерные системы уравнений, содержащие большое число неизвестных и параметров.

**2. Проведение математического исследования** полученной модели и получение соответствующего решения. На этом этапе моделирования в зависимости от сложности рассматриваемой модели применяют различные подходы к ее исследованию и различный смысл вкладывается в понятие решения задачи. Скажем, доказательство теорем *существования и единственности* в определенном смысле решает задачу, однако, являясь зачастую некон-

структивным, оно не позволяет решить проблему изучения этапов получения решения и оценки его количественных характеристик.

Для наиболее грубых и несложных моделей удается получить их *аналитическое решение*. Следует оговориться — использование средств символьных вычислений на ПЭВМ, таких как REDUCE, MAXIMA, MAPLE и «интеллектуальных калькуляторов» MATHEMATICA, MATHCAD, MATLAB, существенно революционизировало это поле деятельности.

Для наиболее точных и сложных моделей основными методами решения являются численные методы, требующие проведения большого объема необходимых вычислений на ПЭВМ. Эти методы позволяют добиться хорошего количественного и даже качественного результата в описании модели.

Приведенная на рис. 1.1 схема частично отражает взаимосвязи этапов математического моделирования. Каждый из этапов математического исследования модели связан с использованием численных методов и получением численного решения.

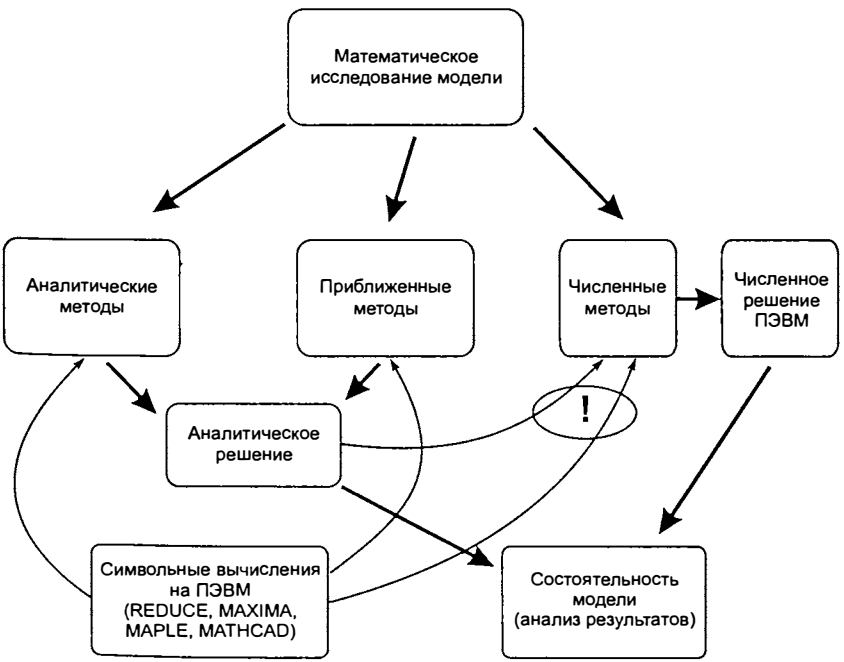



**Рис. 1.1.** Взаимосвязи этапов математического моделирования

**3. Анализ состоятельности предложенной модели**, т. е. осмысление результатов решения, сопоставление полученного решения с имеющимися данными физического эксперимента. На этом этапе решается вопрос о состоятельности математической модели и проведенного исследования. Хорошее согласование с экспериментом обычно свидетельствует о правильности выбора модели. В противном случае необходимы дополнительные уточнения, изменения и повторение предыдущих этапов исследования.

Использование ПЭВМ в процессе математического исследования модели требует специфических численных методов, т. е. такой интерпретации математической (*дискретной* или *вычислительной*) модели, которая может быть реализована на ПЭВМ. Поскольку ПЭВМ выполняет только арифметические и логические операции, то для реализации *вычислительной модели* требуется разработка соответствующего вычислительного алгоритма.

## 1.2. Модели объектов и процессов

Человек издавна использует моделирование для исследования объектов, явлений и процессов в различных областях. Моделирование помогает принимать обоснованные и продуманные решения, предвидеть последствия своей деятельности.

В 1870 г. английское Адмиралтейство спустило на воду новый броненосец «Кэптен». Корабль вышел в море и перевернулся, а вместе с ним погибли 523 человека. Это было совершенно неожиданно для всех, кроме кораблестроителя В. Рида, который предварительно провел исследования на модели броненосца и установил, что корабль опрокинется даже при небольшом волнении. Но ученому, проделывающему какие-то несерьезные опыты, не поверили лорды из Адмиралтейства и случилось непоправимое.

Модели и моделирование используются человечеством давно. С помощью моделей и модельных отношений развились разговорные языки, письменность, графика. Наскальные изображения наших предков, затем картины и книги — это модельные, информационные формы передачи знаний об окружающем мире.

Технология моделирования требует от исследователя умения ставить корректно проблемы и задачи, прогнозировать результаты исследования, проводить разумные оценки, выделять главные

и второстепенные факторы для построения моделей, выбирать аналогии и математические формулировки, решать задачи с использованием компьютерных систем, проводить анализ компьютерных экспериментов. Для успешной работы исследователю необходимо проявлять активный творческий поиск, любознательность и обладать максимумом терпения и трудолюбия.

Навыки моделирования очень важны человеку в жизни. Они помогут разумно планировать свой распорядок дня, учебу, труд, выбирать оптимальные варианты при наличии выбора, удачно разрешать жизненные ситуации.

Рассмотрим примеры, поясняющие понятие модели.

1. Архитектор готовится построить здание. Но прежде чем воздвигнуть его, он сооружает это здание из кубиков на столе, чтобы посмотреть, как оно будет выглядеть. Конечно, архитектор мог бы построить здание без предварительных экспериментов с кубиками, но он должен быть уверен, что здание будет выглядеть достаточно хорошо.

2. Для объяснения вопросов функционирования системы кровообращения лектор демонстрирует плакат, на котором стрелками изображены направления движения крови. Конечно, лектор мог бы для демонстрации воспользоваться подробным анатомическим атласом, но в данном случае ему это совершенно не нужно.

3. Перед тем как запустить в производство новый самолет, его помещают в аэродинамическую трубу и с помощью соответствующих датчиков определяют величины напряжений, возникающих в различных местах конструкции. Конечно, можно запустить самолет в производство и не зная, какие напряжения возникают, скажем, в крыльях. Но эти напряжения, если они окажутся достаточно большими, могут привести к разрушению самолета.

***Модель*** — упрощенное представление о реальном объекте, процессе или явлении. Модель — это такой материальный или мысленно представляемый объект, который в процессе изучения замещает объект-оригинал, сохраняя некоторые важные для данного исследования типичные его черты.

***Моделирование*** — построение моделей для исследования и изучения объектов, процессов или явлений.

Для чего создавать модель, а не исследовать сам оригинал?

Во-первых, можно моделировать оригинал (прототип), которого уже не существует или его нет в действительности.

Во-вторых, оригинал может иметь много свойств и взаимосвязей. Для изучения какого-либо свойства иногда полезно отказаться от менее существенных, вовсе не учитывая их.

## 1.3. Типы моделей

Рассмотрим наиболее распространенные признаки, по которым классифицируют модели:

- область использования;
- учет в модели временного фактора (динамики);
- отрасль знаний;
- способ представления моделей.

***Классификация по области использования***

По области использования модели можно классифицировать следующим образом:

- *учебные модели* — наглядные пособия, различные тренажеры, обучающие программы;
- *научно-технические модели*; создают для исследования процессов и явлений;
- *игровые модели* — это военные, экономические, спортивные, деловые игры;
- *имитационные модели* не просто отражают реальность, а имитируют ее. Эксперимент либо многократно повторяется, либо проводится одновременно со многими другими похожими объектами, но поставленными в разные условия.

***Классификация с учетом фактора времени***

Модели можно разделить на *статические* (это как бы одномоментный срез информации по объекту) и *динамические*. Динамическая модель позволяет увидеть изменения объекта во времени.

### 1.3.1. Классификация моделей

Рассмотрим схему классификации моделей по способу представления (рис. 1.2).

Модели делят на две большие группы: *материальные* и *информационные*.

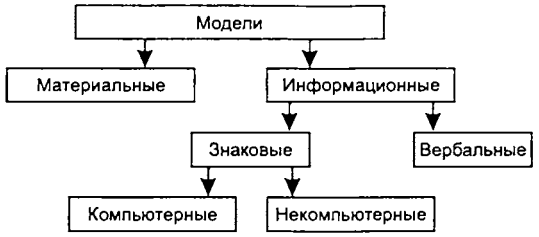



**Рис. 1.2.** Классификация моделей

*Материальные модели* иначе можно назвать предметными, физическими. Они воспроизводят геометрические и физические свойства оригинала и всегда имеют реальное воплощение.

В основе информационного метода моделирования лежит информационный подход к изучению окружающей действительности.

*Информационная модель* — совокупность информации, характеризующая свойства и состояние объекта, процесса, явления, а также взаимосвязь с внешним миром.

Информационные модели в свою очередь подразделяются на *знаковые* и *вербальные*.

*Вербальная модель* — информационная модель в мысленной или разговорной форме. (К таким моделям можно отнести и идею, возникшую у изобретателя, и музыкальную тему, и рифму, прозвучавшую пока еще в сознании автора.)

*Знаковая модель* — информационная модель, выраженная специальными знаками, т. е. средствами любого формального языка (это рисунки, тексты, графики, схемы). По форме представления можно выделить следующие виды информационных моделей:

- *геометрические* — графические формы и объемные конструкции;
- *словесные* — устные и письменные описания с использованием иллюстраций;
- *математические* — математические формулы, отображающие связь различных параметров объекта или процесса;
- *структурные* — схемы, графики, таблицы и т. п.;
- *логические* — модели, в которых представлены различные варианты выбора действий на основе умозаключений и анализа условий;
- *специальные* — ноты, химические формулы и т. п.;
- *компьютерные и некомпьютерные модели.*

## 1.4. Этапы моделирования

Моделирование является одним из ключевых видов деятельности человека. Оно всегда в той или иной форме предшествует любому делу. Моделирование занимает центральное место в исследовании объекта. Оно позволяет обоснованно принимать решение. Решение любой задачи разбивается на несколько этапов. Моделирование — творческий процесс и заключить его в формальные рамки очень трудно. В общем виде его можно представить поэтапно (рис. 1.3).

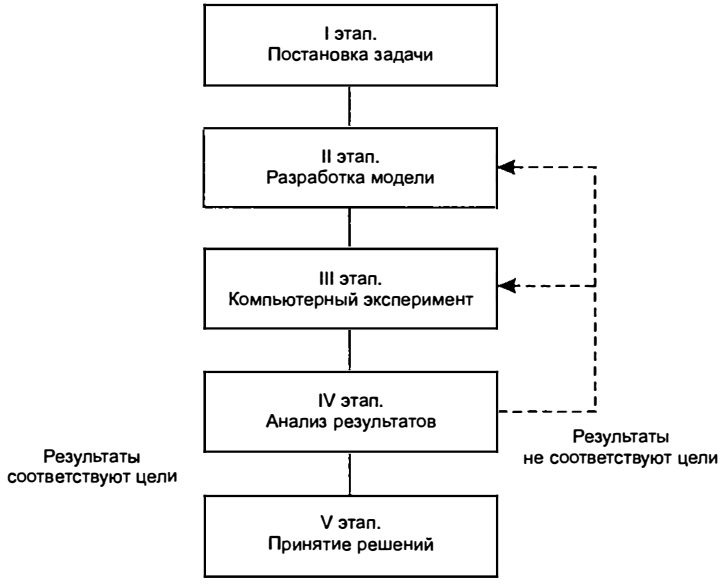



**Рис. 1.3.** Этапы моделирования

Все этапы определяются поставленной задачей и целями моделирования. Рассмотрим основные этапы моделирования подробнее.

**I этап.** Постановка задачи — описание задачи; определение цели моделирования; анализ объекта моделирования.

**II этап.** Разработка модели. На этом этапе выясняются свойства, состояния, действия и другие характеристики элементарных объектов. Формируется представление об элементарных объектах. Выбор наиболее существенной информации при соз-

дании информационной модели и ее сложность обусловлены целью моделирования.

**III этап.** Компьютерный эксперимент и тестирование, т. е. процесс проверки правильности модели.

**IV этап.** Анализ результатов моделирования. Конечная цель моделирования — принятие решения, которое должно быть выработано на основе всестороннего анализа полученных результатов.

## 1.5. Компьютерное моделирование

Основной задачей процесса моделирования является выбор наиболее адекватной к оригиналу модели и перенос результатов исследования на оригинал.

В настоящее время компьютерное моделирование в научных и практических исследованиях является одним из основных инструментов познания.

Компьютерное моделирование начинается как обычно с объекта изучения, в качестве которого могут выступать явления, процесс, предметная область, жизненные ситуации, задачи. После определения объекта изучения строится модель. При построении модели выделяют основные, доминирующие факторы, отбрасывая второстепенные. Выделенные факторы перекладывают на понятный машине язык, разрабатывают алгоритм и программу.

Когда программа готова, проводят компьютерный эксперимент и анализ полученных результатов моделирования при вариации модельных параметров. И уже в зависимости от этих выводов делают нужные коррекции на одном из этапов моделирования: либо уточняют модель, либо алгоритм, либо, точнее, более корректнее определяют объект изучения.

В методе компьютерного моделирования присутствуют все важные элементы развивающего обучения и познания: конструирование, описание, экспериментирование и т. д. В результате добываются знания об исследуемом объекте-оригинале.

Однако важно не путать компьютерную модель (моделирующую программу) с самим явлением. Модель полезна, когда она хорошо согласуется с реальностью. Но модели могут предсказывать и те события, которые не произойдут, тем не менее полез-

ность модели очевидна, так как она помогает понять, почему происходят те или иные явления.

*Современное компьютерное моделирование* выступает как средство общения людей (обмен информационными, компьютерными моделями и программами), осмысления и познания явлений окружающего мира (компьютерные модели Солнечной системы, атома и т. п.), обучения и тренировки (тренажеры), оптимизации (подбор параметров).

**Компьютерная модель** — это модель реального процесса или явления, реализованная компьютерными средствами. Компьютерные модели, как правило, являются знаковыми или информационными. К знаковым моделям в первую очередь относятся математические модели, демонстрационные и имитационные программы.

**Информационная модель** — набор величин, содержащий необходимую информацию об объекте, процессе, явлении.

При построении компьютерной модели используют системный подход, который заключается в следующем. Рассмотрим объект — Солнечную систему. Систему можно разбить на элементы — Солнце и планеты. Введем отношения между элементами, например, удаленность планет от Солнца. Теперь можно рассматривать независимо отношения между Солнцем и каждой из планет, затем обобщить эти отношения и составить общую картину Солнечной системы (принципы декомпозиции и синтеза).

Некоторые характеристики моделей являются неизменными, не меняют своих значений, а некоторые варьируются по определенным законам. Если состояние системы меняется во времени, то модели называют динамическими, в противном случае — статическими (рис. 1.4).

При построении моделей используют два принципа: дедуктивный (от общего к частному) и индуктивный (от частного к общему).

*При первом подходе* рассматривается частный случай общеизвестной, фундаментальной модели. Здесь при заданных предположениях известная модель приспосабливается к условиям моделируемого объекта. Например, можно построить модель свободно падающего тела на основе известного закона Ньютона и в качестве допустимого приближения принять модель равноускоренного движения для малого промежутка времени.

*Второй способ* предполагает выдвижение гипотез, декомпозицию сложного объекта, анализ, затем синтез. Здесь широко ис-

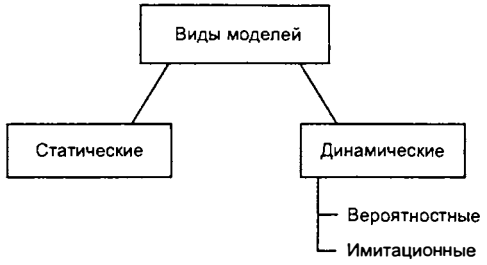



**Рис. 1.4.** Виды моделей

пользуется подобие, аналогичное моделирование, умозаключение с целью формирования каких-либо закономерностей в виде предположений о поведении системы. Например, подобным способом происходит моделирование строения атома. Вспомним модели Томсона, Резерфорда, Бора.

## 1.6. Имитационное моделирование

Теоретическая основа метода имитационного моделирования (метода Монте-Карло) была известна давно. Однако до появления ПЭВМ этот метод не мог найти сколько-нибудь широкого применения, так как моделировать случайные величины вручную — очень трудоемкая работа.

Процессы в системе могут протекать по-разному в зависимости от условий, в которых находится система. Следить за поведением реальной системы при различных условиях и исследовать всевозможные варианты бывает трудно, а иногда и невозможно. В таких случаях выручают модели. Построив модель, можно многократно возвращаться к начальному состоянию и наблюдать за поведением модели. Такой метод исследования систем называется *имитационным моделированием*. Имитационное моделирование применяют в тех случаях, когда необходимо учесть возможно большее разнообразие исходных данных, изучить протекание процессов в различных условиях.

Само название метода «Монте-Карло» происходит от города Монте-Карло в княжестве Монако, знаменитого своими игорными домами. Дело в том, что одним из механических приборов для получения случайных величин является рулетка, используемая в игорных домах.

**Пример 1.1.** Вычислить число π методом Монте-Карло.

***Решение.*** Число π — отношение длины окружности к ее диаметру. Для вычисления числа π с помощью метода Монте-Карло рассмотрим круг радиусом $r = 1$ с центром в точке (1, 1). Его площадь равна $\pi r^2 = \pi^2$. Круг вписан в квадрат, площадь которого равна 4. Выбираем внутри квадрата $N$ случайных точек. Обозначим через $N_{кр}$ число точек, попавших при этом внутрь круга. Геометрически очевидно (рис. 1.5), что

$$\frac{S_{кр}}{S_{кв}} = \frac{N_{кр}}{N}; \quad S_{кр} = 4\frac{N_{кр}}{N}; \quad \pi = 4\frac{N_{кр}}{N}.$$

Формула $\pi = 4\dfrac{N_{кр}}{N}$ дает оценку числа π. Чем больше $N$, тем больше точность этой оценки. Следует заметить, что данный метод вычисления площади будет справедлив только тогда, когда случайные точки будут не просто случайными, а еще и равномерно распределенными по всему квадрату. Для моделирования равномерно распределенных случайных чисел в языке программирования Turbo Pascal используется датчик случайных чисел — функция *Random*. Это специальная компьютерная программа, которая выдает последовательность случайных величин, равномерно распределенных от 0 до 1.

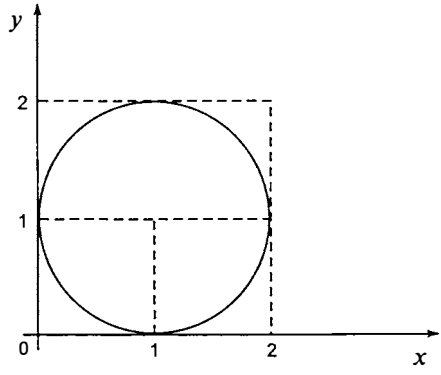



**Рис. 1.5.** Круг, вписанный в квадрат

Таким образом, суть компьютерного эксперимента заключается в обращении к функции *Random* для получения координат точки $x$ и $y$ $N$ раз. При этом определяется, попадет ли точка

с координатами $(x, y)$ в круг единичного радиуса. В случае попадания значение величины $N_{кр}$ увеличивается на единицу.

Ясно, что число точек, попавших внутрь фигуры, тем больше, чем больше фигура, а точность решения будет пропорциональна количеству точек в квадрате. Пара случайных чисел в этом методе может быть рассмотрена как координаты точки на плоскости.

**Программа**

```
Program  monte_karlo;
Uses Crt;
  var i,n,n1:Integer;
        q,x,y,pi:Real;
begin
  ClrScr;  randomize;
  write('Введите n=');  readln(n);
for i:=1 to n do
  begin
    x:=2*random; y:=2*random;
    if sqr(x-1)+sqr(y-1)<=1 then  n1:=n1+1;
  end;
  pi:=4*n1/n;
  writeln('pi=',pi);  Readkey;
end.
```

**Пример 1.2.** Построить математическую модель движения тела, брошенного под углом к горизонту. Выяснить зависимость расстояния и времени полета тела от угла броска и начальной скорости.

***Решение.*** Если тело брошено под углом μ с начальной скоростью $v_0$, то при отсутствии сопротивления воздуха оно опишет параболу. При этом координата положения его в некоторый момент времени $t$ после броска может быть определена из системы уравнений:

$$\begin{cases} x(t) = v_0 \cos at; \\ y(t) = v_0 \sin at - gt^2/2. \end{cases}$$

Эта математическая модель известна из курса физики и позволяет для заданных $v_0$ и $a$ проследить траекторию движения тела и отразить ее на графике.

Прежде всего определим, на каком интервале будет построен график. Начало отрезка — это точка, откуда брошен камень. Можно принять ее за начало отсчета, т. е. $A = 0$. Конец отрезка определяется точкой падения камня, в этой точке $y = 0$. Чтобы решить уравнение

$$v_0 \sin at - gt^2/2 = 0,$$

выразим $t$ через $x$

$$t = x/(v_0 \cos a).$$

Подставив это значение в уравнение, получим решение. Дальность полета

$$b = (v_0^2 \sin 2a)/g.$$

Минимальное значение $Y_{min} = 0$, а максимальное — это наибольшая высота подъема

$$Y_{max} = (v_0^2 \sin 2a)/2g.$$

Теперь, чтобы получить график, нужно изменять значение $t$ от 0 до $T_{max}$, где $T_{max}$ — момент падения, определяемый по формуле

$$T_{max} = (2v_0 \sin 2a)/g.$$

Для каждого значения $t$ получаем сразу обе координаты $(x_i, y_i)$ — в этом случае имеем дело с функцией, заданной параметрически.

Так как все значения, нужные для расчета коэффициентов, известны из формул, то нет необходимости повторного вычисления значений функции. Поэтому выбираем вариант без использования массива.

Оси разметим следующим образом: ось $0X$ — 11 значений от 0 до $v_0^2/g$, ось $0Y$ — 11 значений от 0 до $v_0^2/2g$.

## Программа

```
program stone;
uses graph;
const BoundX=70;  {отступы от края экрана слева и справа}
    BoundY=70;    {отступы от края экрана снизу и сверху}
    g=9.8;        {ускорение свободного падения}
    rad=3.1415/180;       {перевод в радианы}
```

```
var KX, KY,LX,LY:integer;      {коэффициенты}
HX, HY,DX,DY,HT:real;          {коэффициенты}
XM,YM:integer;                 {размер экрана}
x,y,x0,y0:integer;             {точки на графике}
i,gr,gm:integer;
v0,alfa:real;                  {начальная скорость и угол броска}
xi,yi:real;                    {абсциссы и ординаты точек}
Bmax,Hmax,Tmax:real;           {максимально возможные B, H, T}
t:real;                        {время}
Xmax,Ymax:real;                {наибольшие значения по осям}
R:real;
S:string[8];                   {строка для текста}
Procedure  FXY(T,V0,ALF:REAL; VAR X,Y:REAL);
  Begin
  X:=V0*COS(ALF*RAD)*T;
  Y:=V0*SIN(ALF*RAD)*T-G*T*T/2;
  End;
Begin
  Write('Введите начальную скорость '); readln(v0);
  Write('Введите угол броска '); readln(alfa);
  gr:=detect; initgraph(gr,gm, ' ');
{определение коэффициентов, зависящих от размера экрана}
XM:=GetMaxX; YM:=GetMaxY;
KX:=XM+1-2*BoundX; KY:=YM+1-2*BoundY;
{определяем остальные коэффициенты, нужные для построения}
  R:=sin(alfa*rad); Bmax:=v0*v0*sin(2*alfa*rad)/g;
  Xmax:=v0*v0/g; Hmax:=v0*v0*r*r/(2*g);
  Ymax:=v0*v0/(2*g); Tmax:=2*v0*r/g;
  hx:=Xmax/(kx-1); hy:=Ymax/(ky-1);
  lx:= kx div 10; ly:=ky div 10; dx:=Xmax/10; dy:=Ymax/10;
  ht:=Tmax/100;     {строим график по 100 точкам}
{определяем цвет экрана и цвет осей}
SetBkColor(Cyan); SetColor(Blue);
{рисуем оси}
  LINE(BoundX,BoundY,BoundX, YM-BoundY);
  x0:=BoundX; y0:=BoundY; x:=XM-BoundX; y:=y0;
  LINE(BoundX,YM-BoundY,XM-BoundX, YM-BoundY);
{размечаем оси}
x:=0; y:=ym-BoundY;
for i:=0 to 10 do
```

```
  begin
  STR ((dy*i):8:2,s);   OutTextXY(x,y,s);    y:=y-ly;
  end;
x:=BoundX div 2;   y:=ym-BoundY div 2;
for i:=0 to 10 do
  begin
  STR ((dx*i):8:2,s);   OutTextXY(x,y,s);    x:=x+lx;
  end;
STR(Bmax:8:2,s);
OutTextXY(BoundX+20,5, 'Дальность полета: '+s);
STR(Hmax:8:2,s);
OutTextXY(BoundX+20,15, 'Высота подьема: '+s);
{устанавливаем цвет графика функции}
SetColor(LightRed);
{рисуем график}
t:=0;  x0:=BoundX;  y0:=ym-boundY;
while (t<Tmax) do
  begin
  t:=t+ht;   FXY(t,v0,alfa,xi,yi);
  x:=BoundX+round(xi/hx);
  y:=ym-BoundY-round(yi/hy);
  line(x0,y0,x,y);   x0:=x;  y0:=y;
  end;
readln;
closegraph;
end.
```

## 1.7. Полное построение алгоритма

В последние годы большое распространение получила концепция структурного программирования. Понятие структурного программирования включает определенные принципы проектирования, кодирования, тестирования и документирования программ в соответствии с заранее определенной жесткой дисциплиной.

Полное построение алгоритма предусматривает выполнение идущих последовательно друг за другом следующих этапов:

1) постановка задачи;
2) построение модели;
3) разработка алгоритма;

4) проверка правильности алгоритма;
5) реализация, т. е. программирование алгоритма;
6) анализ алгоритма и его сложности;
7) проверка (отладка) программы;
8) составление документации.

Не все эти этапы четко различимы между собой, особенно эта различимость делается мало заметной при программировании простых задач (рис. 1.6). При программировании простых задач некоторые этапы могут вообще не выполняться — настолько очевидны их результаты.

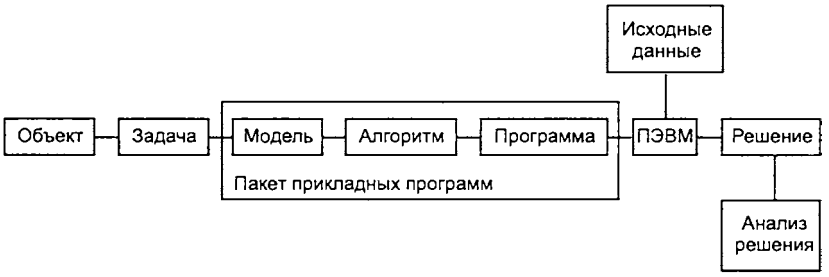



**Рис. 1.6.** Этапы решения задач на ПЭВМ

Наоборот, при программировании сложных, объемных задач некоторые из вышеперечисленных этапов приходится выполнять не в том порядке, как здесь указано, или выполнять их не один раз.

Рассмотрим более подробно каждый из этапов построения алгоритма.

**Постановка задачи.** Прежде чем понять задачу, ее нужно точно сформулировать. Это условие само по себе не является достаточным для понимания задачи, но оно абсолютно необходимо.

Обычно процесс точной формулировки задачи сводится к постановке правильных вопросов. Перечислим некоторые полезные вопросы для плохо сформулированных задач:

- Понятна ли терминология, используемая в предварительной формулировке?
- Что дано? Что нужно найти?
- Как определить решение?
- Каких данных не хватает, или, наоборот, все ли перечисленные в формулировке задачи данные используются?
- Какие сделаны допущения?

Возможны и другие вопросы, возникающие в зависимости от конкретной задачи.

**Построение модели.** Задача четко поставлена, теперь нужно сформулировать для нее математическую модель. Выбор модели существенно влияет на остальные этапы в процессе решения.

Выбор модели — в большей степени искусство, чем наука. Правда, если вы будете встречаться с типовой задачей, то опыт, приобретенный ранее, не потребует от вас особого творческого подхода, и в этом случае будет удобно и целесообразно воспользоваться ранее наработанными правилами. Поэтому изучение удачных моделей — это наилучший способ приобрести опыт в моделировании.

Приступая к разработке модели, следует задать, по крайней мере, два основных вопроса:

1. Какие математические структуры больше всего подходят для задачи?

2. Существуют ли решенные аналогичные задачи?

Второй вопрос, возможно, самый полезный во всей математике. В контексте моделирования он часто дает ответ на первый вопрос. Действительно, большинство решаемых в математике задач, как правило, являются модификациями ранее решенных.

Сначала нужно рассмотреть первый вопрос. Необходимо описать математически, что мы знаем и что хотим найти. На выбор соответствующей структуры будут оказывать влияние такие факторы, как:

- ограниченность наших знаний относительно небольшим количеством структур;
- удобство представления;
- простота вычислений;
- полезность различных операций, связанных с рассматриваемой структурой или структурами.

Сделав пробный выбор математической структуры, задачу следует переформулировать в соответствующих терминах математических объектов. Это будет одна из возможных моделей, если мы можем утвердительно ответить на такие вопросы:

- Вся ли важная информация задачи хорошо описана математическими объектами?
- Существует ли математическая величина, ассоциируемая с искомым результатом?

- Выявили ли мы какие-нибудь полезные отношения между объектами модели?
- Можем ли мы работать с моделью? Удобно ли с ней работать?

**Разработка алгоритма.** Выбор метода разработки алгоритма зачастую значительно зависит от выбора модели и может в большой степени повлиять на эффективность алгоритма решения. Два различных алгоритма могут быть правильными, но очень сильно различаться по эффективности.

**Правильность алгоритма.** Доказательство правильности алгоритма — это один из наиболее трудных, а иногда и особенно утомительных этапов создания алгоритма. Вероятно, наиболее распространенный прием доказательства правильности программы — это прогон ее на разных тестах. Если выданные программой ответы могут быть подтверждены известными или вычисленными вручную данными, возникает искушение сделать вывод, что программа «работает» правильно. Однако этот метод редко исключает все сомнения; может существовать случай, в котором программа не работает.

Рассмотрим следующую общую методику доказательства правильности алгоритма. Предположим, что алгоритм описан в виде последовательности шагов от 0 до $m$. Постараемся предложить некое обоснование правомерности для *каждого* шага. В частности, может потребоваться формулировка утверждения об условиях, действующих до и после пройденного шага. Затем постараемся предложить доказательство конечности алгоритма, при этом будут проверены все подходящие входные данные и получены все подходящие выходные данные. Другой метод доказательства правильности алгоритма, который не имеет специального названия, состоит в следующем. Для каждого цикла, который имеется в программе (алгоритме), вручную (например, на калькуляторе) подсчитываются две контрольные точки. Если контрольные точки совпадают со значениями, выданными программой, можно быть уверенным, что все циклы в программе работают правильно.

Почему речь идет о двух контрольных точках? Дело в том, что первое контрольное значение в программе может быть вычислено правильно, а затем в этом цикле будут произведены некоторые некорректные действия, которые приведут к искажению всех последующих результатов. Совпадение второго контрольного значения как раз и подтверждает, что в данном цикле некор-

ректности нет. Повторю только еще раз — два контрольных вычисления должны быть сделаны для каждого цикла программы.

Следует подчеркнуть и тот факт, что правильность алгоритма еще ничего не говорит об его эффективности. В этом смысле исчерпывающие алгоритмы, или, как их еще называют, — алгоритмы полного перебора — редко бывают хорошими во всех отношениях.

**Реализация алгоритма.** Как только алгоритм выражен, допустим, в виде последовательности шагов и мы убедились в его правильности, настает черед реализации алгоритма, т. е. написания программы для компьютера.

При этом возникают следующие проблемы:

- очень часто отдельно взятый шаг алгоритма может быть выражен в форме, которую трудно перевести непосредственно в конструкции языка программирования. Например, один из шагов алгоритма может быть записан в виде, требующем целой подпрограммы для своей реализации;
- реализация может оказаться трудным процессом потому, что перед тем, как написать программу, необходимо построить целую систему структур данных для представления важных аспектов используемой модели.

Чтобы сделать это, необходимо ответить, например, на такие вопросы:

- Каковы основные переменные?
- Каких типов они бывают?
- Сколько нужно массивов и какой размерности?
- Имеет ли смысл пользоваться связными списками?
- Какие нужны подпрограммы (возможно, уже записанные в памяти)?
- Каким языком программирования пользоваться?

Конкретная реализация может существенно влиять на требования к памяти и на скорость алгоритма.

Сделаем одно важное замечание. Одно дело — доказать правильность конкретного алгоритма, описанного в словесной форме. Другое дело — доказать, что данная машинная программа, предположительно являющаяся реализацией этого алгоритма, также правильна. Таким образом, необходимо тщательно следить, чтобы процесс преобразования правильного алгоритма (в словесной форме, или форме схемы алгоритма) в программу, написанную на алгоритмическом языке, «заслуживал доверия».

Исполняемая программа может быть описана некоторым множеством значений всех параметров, переменных, данных и результатов промежуточных вычислений. Это множество конечно и явным образом влияет на процесс выполнения программы. Все элементы данных множества можно идентифицировать, т. е. присвоить им индивидуальные имена, причем среди этих имен будут не только начальные данные, но и результаты промежуточных вычислений.

Для любой работающей программы каждый именованный элемент данных множества имеет либо определенное, либо неопределенное значение (например, элемент, которому еще не было присвоено значение).

Весь набор соответствий между именами элементов множества и их текущими значениями (определенными или неопределенными) называется **вектором состояния программы.**

С этой точки зрения оператор присваивания представляет собой описание преобразований вектора состояний. Выполнение программы начинается с первого вектора состояния. После выполнения очередного оператора программа переходит к другому состоянию, которое описывается другим вектором состояния. Последовательные преобразования вектора состояний полностью описывают результат исполнения программы.

На рис. 1.7 приняты следующие обозначения:

$V1$, $V2$, ..., $Vn$ — векторы состояний; $S1$, $S2$ — выполняемые операторы или блоки программы. Можно заметить, что, например, вектор $V2$ является одновременно выходным для $S1$ и входным для $S2$.

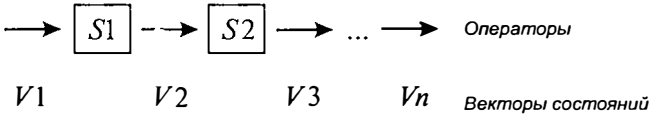



**Рис. 1.7.** Процесс исполнения программы

Изменение вектора состояний программы происходит по завершению выполнения очередного оператора, т. е. не плавно, а скачком. Вектор состояния во время выполнения очередного программного оператора неизменен и недоступен извне. Эта особенность вектора состояний программы важна при анализе выполнения повторяющихся операторов.

### 1.7.1. Эффективность программ

Основной задачей программирования является создание правильных, а не эффективных программ. Если программа работает неправильно, то вопрос об ее эффективности не имеет значения.

Однако правильность не является дополнительной характеристикой программы в отличие от эффективности. Неправильную, но эффективную программу редко можно сделать правильной, а правильную, но неэффективную можно оптимизировать и добиться эффективности. Поэтому оптимизация относится лишь ко второму этапу отладки программ.

Наиболее разумный подход в данном случае к программированию заключается в создании программы, которая наилучшим способом реализована, не уделяя особого внимание эффективности. Затем, если программа пригодна для решения задачи, ее можно модернизировать. Поэтому задача написания эффективных программ как бы разбивается на две подзадачи: нахождение того, что надо оптимизировать, и выбор способа оптимизации.

Здесь надо заметить, что в погоне за эффективностью программист может потратить огромное количество времени на оптимизацию редко выполняемых блоков и добиться небольших результатов по оптимизации. Экономии машинного времени и ресурсов можно достичь, только оптимизируя *многократно* выполняемые блоки программ.

Эффективность программы условно может быть определена двумя параметрами: необходимым для ее работы временем и памятью, которая требуется программе. Причем, фактор времени — более важный по сравнению с памятью.

Оптимизировать время выполнения можно за счет более экономного программирования часто выполняемых операций, удаление из программы перекрестных ссылок, использование сложных ссылочных структур и многое другое. С проблемой оптимизации памяти дело обстоит значительно сложнее.

Обычно программистов не заботит память, пока программа не превысит возможности машины и перестанет работать. Тогда программисту становится очевидным факт, что память машины не бесконечна. Идеальной будет ситуация, когда в соответствии с запросами по памяти программист может использовать более мощную ПЭВМ. Однако здесь существует еще одна опасность: любая программа стремится занять всю возможную память на ПЭВМ (это один из негласных принципов программирования) и

поэтому бесконечно наращивать ресурсы памяти машины нереально. Но, с другой стороны, экономное использование памяти почти всегда ведет к увеличению времени работы программы.

Возможно выход из данной ситуации лежит где-то в некоторой средней точке: наиболее малое время выполнения программы с той наиболее экономной моделью памяти, как это возможно.

Поэтому можно дать несколько советов для выполнения этих рекомендаций.

1. Создавать программы с помощью оптимизирующих трансляторов.

2. Максимально структурировать программы, широко использовать модульность.

3. Удалять ситуации, предполагающие исключительные действия программ, собирать эти действия в специальных модулях, что увеличит скорость работы с основными структурами программы.

4. Присваивать начальные значения элементам массивов данных и всем данным, если они используются в структурах типа «сумматор», непосредственно перед первым их использованием, а не в самом начале работы программы.

5. Широко использовать прямые доступы к данным и ссылочные переменные. Часто используемые данные располагать рядом в одной области памяти, предварительно сгруппировав их по описаниям или другому признаку.

6. Если некоторые процедуры выполняются только в одном месте программы, то лучше такую процедуру поместить непосредственно в программу.

7. Решить, что интересует в первую очередь: память или время, так как экономия на одном влияет на использование другого. И от этого принципа не отступать до окончания работы над всем проектом.

## 1.8. Главные принципы, лежащие в основе создания эффективных алгоритмов

Каждый, кто занимается разработкой алгоритмов, должен овладеть некоторыми основными методами и понятиями. Всем когда-то впервые пришлось столкнуться с трудной задачей, за-

дать вопрос: с чего начать? Один из возможных путей — просмотреть свой запас общих алгоритмических методов для того, чтобы проверить, нельзя ли с помощью одного из них сформулировать решение новой задачи.

Ну, а если такого запаса нет, то как все-таки разработать хороший алгоритм? С чего начать? У всех есть печальный опыт, когда смотришь на задачу и не знаешь, что делать. Рассмотрим три общих метода решения задач, полезных для разработки алгоритмов.

**Первый метод** связан со сведением трудной задачи к последовательности более простых задач. Конечно, мы надеемся на то, что более простые задачи легче поддаются обработке, чем первоначальная задача, а также на то, что решение первоначальной задачи может быть получено из решений этих более простых задач. Такая процедура называется *методом частных целей.*

Этот метод выглядит очень разумно. Но, как и большинство общих методов решения задач или разработки алгоритмов, его не всегда легко перенести на конкретную задачу. Осмысленный выбор более простых задач — скорее, искусство или интуиция, чем наука. Более того, не существует общего набора правил для определения класса задач, которые можно решать с помощью такого подхода. Размышление над любой конкретной задачей начинается с постановки вопросов. Частные цели могут быть установлены, когда получены ответы на следующие вопросы:

1. Можем ли мы решить часть задачи? Можно ли, игнорируя некоторые условия, решить оставшуюся часть задачи?

2. Можем ли мы решить задачу для частных случаев? Можно ли разработать алгоритм, который дает решение, удовлетворяющее всем условиям задачи, но входные данные которого ограничены некоторым подмножеством всех входных данных?

3. Есть ли что-то, относящееся к задаче, что мы недостаточно хорошо поняли? Если попытаться глубже вникнуть в некоторые особенности задачи, сможем ли мы что-то узнать, что поможет нам подойти к решению?

4. Встречались ли мы с похожей задачей, решение которой известно? Можно ли видоизменить ее решение для выполнения конкретной задачи? Возможно ли, что эта задача эквивалентна известной нерешенной задаче?

**Второй метод** разработки алгоритмов известен как *метод подъема.* Алгоритм подъема начинается с принятия начального

предположения или вычисления начального решения задачи. Затем начинается насколько возможно быстрое движение «вверх» от начального решения по направлению к лучшим решениям. Когда алгоритм достигнет такой точки, из которой больше невозможно двигаться вверх, алгоритм останавливается. К сожалению, мы не можем всегда гарантировать, что окончательное решение, полученное с помощью алгоритма подъема, будет оптимальным. Эта ситуация часто ограничивает применение метода подъема.

Вообще методы подъема являются «грубыми». Они запоминают некоторую цель и стараются сделать все, что могут и где могут, чтобы подойти ближе к цели. Это делает их несколько недальновидными. Недальновидность метода подъема хорошо иллюстрируется следующим примером.

Пусть требуется найти максимум функции $y = f(x)$, представленной графиком (рис. 1.8).

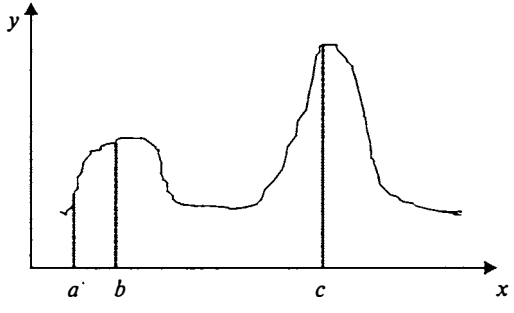



**Рис. 1.8.** Иллюстрация метода подъема

Если начальное значение аргумента $x$ равно $a$, то метод подъема дает устремление к ближайшей цели, т. е. к значению функции в точке $x = b$, тогда как подлинный максимум этой функции находится при $x = c$. То есть метод подъема в данном примере находит локальный максимум, но не глобальный. В этом и заключается «грубость» метода подъема.

**Третий метод** известен как отрабатывание назад, т. е. работа этого алгоритма начинается с цели или решения задачи и затем осуществляется движение к начальной постановке задачи. Затем, если эти действия обратимы, производится движение обратно — от постановки задачи к решению.

## 1.9. Источники и классификация погрешностей

Задача вычисления $y = A(x)$ называется корректно поставленной, если для любых входных данных из некоторого класса существует решение задачи, единственное и устойчивое по входным данным.

Источниками возникновения погрешности численного решения задачи являются следующие:

- неточность математического описания, в частности, неточность задания начальных данных;
- неточность численного метода решения задачи.

  Данная причина, возникает например, когда решение математической задачи требует неограниченного или неприемлемо большого числа арифметических операций, что приводит к необходимости ограничения их числа, т. е. использования приближенного решения;
- конечная точность машинной арифметики.

Существуют следующие виды погрешностей:

- неустранимая погрешность;
- погрешность метода;
- вычислительная погрешность.

**Неустранимая погрешность** состоит из двух частей:

а) погрешности, обусловленной неточностью задания числовых данных, входящих в математическое описание задачи;

б) погрешности, являющейся следствием несоответствия математического описания задачи реальной действительности (погрешность математической модели). Для вычислителя погрешность задачи следует считать неустранимой, хотя постановщик задачи иногда может ее изменить.

Результирующая погрешность определяется как сумма величин всех перечисленных выше погрешностей.

**Погрешность метода** связана со способом решения поставленной математической задачи. Она появляется в результате замены исходной математической модели другой и/или конечной последовательностью других более простых (например, линейных) моделей. При создании численных методов закладывается возможность отслеживания таких погрешностей и доведения их до сколь угодно малого уровня. Отсюда естественно отношение к погрешности метода как устранимой (или условной).

**Вычислительная погрешность** (погрешность округлений) обусловлена необходимостью выполнения арифметических операций над числами, усеченными до количества разрядов, зависящего от применяемой вычислительной техники.

### 1.9.1. Понятия о погрешности машинных вычислений

В основу запоминающего устройства ПЭВМ положены однотипные физические устройства, имеющие $r$ устойчивых состояний, причем каждому устройству ставится в соответствие одинаковое количество $k$ элементов. Упорядоченные элементы образуют разрядную сетку машинного слова: в каждом разряде может быть записано одно из базисных чисел 0, 1, ..., $r-1$ (одна из $r$ цифр $r$-й системы счисления) и в специальном разряде отображен знак «+» или «–». При записи чисел с фиксированной запятой, кроме упомянутых $r$ параметров (основания системы счисления) и $k$ (количества разрядов, отводимых под запись числа), указывается еще общее количество $l$ разрядов, выделяемых под дробную часть числа. Таким образом, положительное вещественное число $a$ представляет собой в $r$-й системе бесконечную, непериодическую дробь и отображается конечной последовательностью

$$a \approx \alpha_1 r^{k-l-1} + \alpha_2 r^{k-l-2} + \ldots + \alpha_{k-l} r^0 + \\ + \alpha_{k-l+1} r^{-1} + \ldots + \alpha_{k-1} r^{-(l-1)} + \alpha_k r^{-l},$$

где $\alpha_l \in \{0; 1; \ldots; r-1\}$.

**Диапазон** представляемых таким способом чисел определяется числами с наибольшими цифрами во всех разрядах, т. е. наименьшим числом $-(r-1)(r-1)\ldots(r-1)$ и наибольшим $(r-1)(r-1)\ldots(r-1)$, а абсолютная точность представления есть оценка, зависящая от способа округления.

Абсолютная точность представления вещественных чисел с фиксированной запятой одинакова в любой части диапазона. В то же время относительная точность может значительно различаться в зависимости от того, берется $a$ близким к нулю или к границе диапазона.

**Пример 1.3.** Рассмотрим запоминающее устройство (ЗУ) с фиксированной запятой, состоящее из $k = 7$ элементов и имеющее $r = 10$ ($r = 0, 1, ..., 9$). Будем считать, что общее количество разрядов, выделяемых под дробную часть, $l = 3$, под знак – один разряд, под целую часть – три разряда (рис. 1.9).

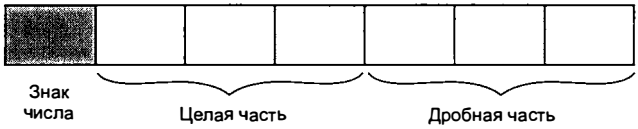



**Рис. 1.9.** Представление чисел

Тогда наибольшее число, которое можно сохранить в данном запоминающем устройстве, составляет 999,999, а наименьшее равно –999,999. У любого числа из указанного диапазона, являющегося бесконечной периодической дробью, вне зависимости от его величины после запятой сохраняется только три цифры. Поэтому абсолютная точность представления чисел $a_1 = 1{,}123456$ и $a_2 = 999{,}123456$ оказывается одинаковой

$$\Delta_1 = 1{,}123456 - 1{,}123 = 0{,}000456,$$

$$\Delta_2 = 999{,}123456 - 999{,}123 = 0{,}000456.$$

Относительные погрешности представления этих чисел в запоминающем устройстве будут различны:

$$\delta_1 = \frac{\Delta_1}{a_1} = 0{,}4\,\%, \quad \delta_2 = \frac{\Delta_2}{a_2} = 0{,}5 \cdot 10^{-4}\,\%.$$

В основе представления числа **с плавающей запятой** лежит экспоненциальная форма записи:

$$a = Mr^p,$$

где $r$ — основание; $p$ — порядок; $M$ — мантисса ($r^{-1} \le |M| \le 1$).

Структура машинного слова запоминающего устройства с плавающей запятой представлена на рис. 1.10.


**Рис. 1.10.** Структура машинного слова

Например, для записи числа в 48-разрядном машинном слове 40 двоичных разрядов выделялись под мантиссу, 6 — под порядок числа и 2 — под знаки мантиссы, т. е. $r = 2$, $l = 40$, $m = 6$. Следовательно, точность представления чисел с плавающей запятой не хуже $2^{-39}$ ($\approx 10^{-12}$), граница машинного нуля $2^{-64}$ ($\approx 10^{-19}$), машинной бесконечности $2^{63}$.

## 1.10. Абсолютная и относительная погрешности

**Абсолютной погрешностью измерения** называется величина, определяемая разницей между результатом измерения $x$ и истинным значением измеряемой величины $x_0$:

$$\Delta x = |x - x_0| .$$

Величина $\delta$, равная отношению абсолютной погрешности измерения к результату измерения, называется **относительной погрешностью**:

$$\delta = \frac{\Delta x}{|x|} \cdot 100\ \%.$$

**Пример 1.4.** Приближенным значением числа $\pi$ является 3,14. Тогда погрешность его равна 0,00159... . Абсолютную погрешность можно считать равной 0,0016, а относительную погрешность равной 0,0016/3,14 = 0,00051 = 0,051 %.

***Значащие цифры.*** Если абсолютная погрешность величины $a$ не превышает одной единицы разряда последней цифры числа $a$, то говорят, что у числа все знаки верные. Приближенные числа следует записывать, сохраняя только верные знаки. Если, например, абсолютная погрешность числа 52 400 равна 100, то это число должно быть записано, например, в виде $524 \cdot 10^2$ или $0{,}524 \cdot 10^5$. Оценить погрешность приближенного числа можно, указав, сколько верных значащих цифр оно содержит. При подсчете значащих цифр не считаются нули с левой стороны числа.

Например, число 0,0283 имеет три верных значащих цифры, а 2,5400 — пять верных значащих цифр.

***Правила округления чисел.*** Если приближенное число содержит лишние (или неверные) знаки, то его следует округлить.

При округлении возникает дополнительная погрешность, не превышающая половины единицы разряда последней значащей цифры ($d$) округленного числа. При округлении сохраняются только верные знаки; лишние знаки отбрасываются, причем если первая отбрасываемая цифра больше или равна $d/2$, то последняя сохраняемая цифра увеличивается на единицу.

Лишние цифры в целых числах заменяются нулями, а в десятичных дробях отбрасываются (как и лишние нули). Например, если погрешность измерения 0,001 мм, то результат 1,07005 округляется до 1,070. Если первая из изменяемых нулями и отбрасываемых цифр меньше 5, остающиеся цифры не изменяются. Например, число 148 935 с точностью измерения 50 имеет округление 148 900. Если первая из заменяемых нулями или отбрасываемых цифр равна 5, а за ней не следует никаких цифр или идут нули, то округление производится до ближайшего четного числа. Например, число 123,50 округляется до 124. Если первая из заменяемых нулями или отбрасываемых цифр больше 5 или равна 5, но за ней следует значащая цифра, то последняя остающаяся цифра увеличивается на единицу. Например, число 6783,6 округляется до 6784.

**Пример 1.5.** При округлении числа 1284 до 1300 абсолютная погрешность составляет $1300 - 1284 = 16$, а при округлении до 1280 абсолютная погрешность составляет $1280 - 1284 = 4$.

**Пример 1.6.** При округлении числа 197 до 200 абсолютная погрешность составляет $200 - 197 = 3$. Относительная погрешность равна $3/197 \approx 0{,}01523$ или приближенно $3/200 \approx 1{,}5$ %.

**Пример 1.7.** Продавец взвешивает арбуз на чашечных весах. В наборе гирь наименьшая — 50 г. Взвешивание дало 3600 г. Это число — приближенное. Точный вес арбуза неизвестен. Но абсолютная погрешность не превышает 50 г. Относительная погрешность не превышает $50/3600 \approx 1{,}4$ %.

## 1.11. Погрешности решения задачи на ПЭВМ

В качестве основных источников погрешности обычно рассматривают три вида ошибок. Это так называемые ошибки усечения, ошибки округления и ошибки распространения. Напри-

мер, при использовании итерационных методов поиска корней нелинейных уравнений результаты являются приближенными в отличие от прямых методов, дающих точное решение. С точки зрения точности результата использование прямых методов может показаться более предпочтительным. Однако на самом деле при решении задачи на компьютере ответ все равно будет содержать погрешность.

### 1.11.1. Ошибки усечения

Этот вид ошибок связан с погрешностью, заложенной в самой задаче. Он может быть обусловлен неточностью определения исходных данных. Например, если в условии задачи заданы какие-либо размеры, то на практике для реальных объектов эти размеры известны всегда с некоторой точностью. То же самое касается любых других физических параметров. Сюда же можно отнести неточность расчетных формул и входящих в них числовых коэффициентов.

Большое число расчетных формул являются эмпирическими и дают результат с некоторой погрешностью, содержат подгоночные коэффициенты, обеспечивающие приемлемую ошибку в ограниченном диапазоне входных параметров. Поэтому, как правило, если исходные данные известны с некоторой погрешностью, вряд ли стоит пытаться получить результат с меньшей погрешностью.

### 1.11.2. Ошибки распространения

Данный вид ошибок связан с применением того или иного способа решения задачи. В ходе вычислений неизбежно происходит накопление или, иначе говоря, распространение ошибки. Помимо того, что сами исходные данные не являются точными, новая погрешность возникает при их перемножении, сложении и т. п. Накопление ошибки зависит от характера и количества арифметических действий, используемых в расчете.

Обычно для решения одной и той же задачи может быть использован ряд различных методов решения. Например, систему линейных алгебраических уравнений можно решить методом Гаусса или через определители (методом Крамера). Теоретически

оба метода позволяют получить точное решение. Однако на практике при решении больших систем уравнений метод Гаусса обеспечивает меньшую погрешность, чем метод Крамера, так как использует меньший объем вычислений.

### 1.11.3. Ошибки округления

Это тип ошибок связан с тем, что истинное значение числа не всегда точно сохраняется компьютером. При сохранении вещественного числа в памяти компьютера оно записывается в виде мантиссы и порядка примерно так же, как отображается число на калькуляторе.

На рис. 1.11 используются следующие обозначения: $R_1$, $R_2$, $R_3$, ..., $R_n$ — разряды мантиссы; $D_1$, $D_2$, ..., $D_m$ — разряды порядка. На самом деле конечно, в отличие от дисплея калькулятора, мантисса и порядок числа, включая их знаки, в памяти компьютера хранятся в двоичном виде. Но для обсуждения природы ошибок округления это различие не столь принципиально.

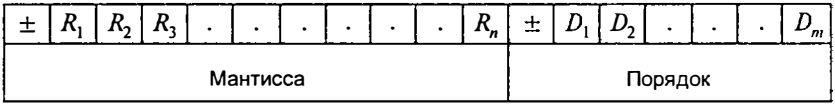



**Рис. 1.11.** Структура записи вещественного числа

Понятно, что иррациональные числа такие, как $\pi = 3{,}14159...$ и $e = 2{,}712...$, не могут быть представлены точно, так как количество их значащих цифр превышает число отведенных разрядов мантиссы. При этом цифра последнего сохраняемого в ПЭВМ разряда может быть записана с округлением или без него. Фактически при заданной структуре хранения числа компьютер может использовать не бесконечное, а конечное число рациональных чисел.

Поэтому любой входной параметр решаемой задачи, ее промежуточный результат и окончательной ответ всегда округляются до разрешенных в компьютере чисел.

Следующий важный вывод касается диапазона представления чисел в ПЭВМ. Если проводить рассуждения для десятичной системы счисления, то максимальное по модулю число, ко-

торое может быть представлено в соответствии со схемой на рис. 1.12, равно

$$\pm X_{\infty} = \pm 999 \ldots 9 \times 10^{+99 \ldots 9}.$$

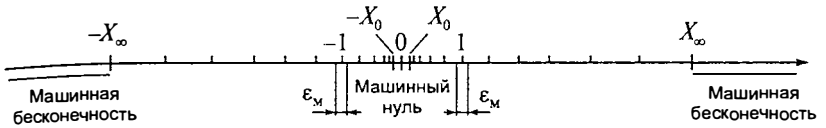



**Рис. 1.12.** Машинная числовая ось

Все числа, превышающие по модулю $X_{\infty}$, не представимы в ПЭВМ и рассматриваются как *машинная бесконечность*. Если в ходе расчетов будет получен результат, превышающий $X_{\infty}$, то произойдет аварийное завершение вычислений по *переполнению*.

Минимальное по модулю число, сохраняемое в памяти компьютера, равно

$$\pm X_0 = \pm 000 \ldots 1 \times 10^{-99 \ldots 9}.$$

Числа, модуль которых меньше $X_0$, воспринимаются ПЭВМ как нуль, точнее как *машинный нуль*. Если при выполнении расчетов будет получен результат меньше, чем $X_0$, то это будет воспринято как *потеря порядка*. Обычно в подобной ситуации результат полагается равным нулю, и вычисления продолжаются.

На рис. 1.12 показана «машинная» числовая ось, на которой отмечены $X_0$ и $X_{\infty}$. Числа располагаются на оси неравномерно. Их плотность возрастает по мере приближения к нулю.

Вблизи единицы отмечена небольшая область $\varepsilon_м$, которую называют *машинное эпсилон*. Параметр $\varepsilon_м$ весьма важен, так как он характеризует относительную точность представления чисел в компьютере. В зависимости от способа округления чисел в ПЭВМ величина $\varepsilon_м$ определяется первым отбрасываемым или последним сохраняемым разрядом мантиссы.

# Глава 2
# ЧИСЛЕННЫЕ МЕТОДЫ

## 2.1. Элементарные функции и их свойства

**Функцией** (*функциональной зависимостью*) называется закон, по которому каждому значению независимой переменной $x$ из некоторого множества чисел, называемого *областью определения функции*, ставится в соответствие одно вполне определенное значение величины $y$. Совокупность значений, которые принимает зависимая переменная $y$, называется *областью значений функции*.

**Графиком функции** называется множество всех точек координатной плоскости с координатами $(x; y)$, такими, что абсцисса $x$ принимает все значения из области определения, а ордината $y$ равна значению функции в точке $x$.

Функция $f(x)$ называется *четной*, если для любого $x$ из ее области определения $-x$ также принадлежит области определения, причем, $f(-x) = f(x)$.

Функция $f(x)$ *возрастает* на некотором интервале, если для любых значений $x_1$ и $x_2$, принадлежащих этому интервалу, таких, что $x_2 > x_1$, выполнено неравенство $f(x_2) > f(x_1)$. Функция $f(x)$ *убывает* на некотором интервале, если для любых значений $x_1$ и $x_2$, принадлежащих этому интервалу, таких, что $x_2 > x_1$, выполнено неравенство $f(x_2) < f(x_1)$.

Точка $x_0$ называется *точкой минимума* функции $f(x)$, если для всех значений $x$ из некоторой окрестности $x_0$ выполнено неравенство $b \in R$.

**Алгебраическая функция** — функция, которая в окрестности каждой точки области определения может быть задана неявно с помощью алгебраического уравнения.

Функция $F(x_1, x_2, \ldots, x_n)$ называется алгебраической в точке $A = (a_1, a_2, \ldots, a_n)$, если существует окрестность точки $A$, в которой верно тождество

$$P(F(x_1, x_2, \ldots, x_n), x_1, x_2, \ldots, x_n) = 0,$$

где $P$ — многочлен от $(n+1)$-й переменной.

Функция называется алгебраической, если она является алгебраической в каждой точке области определения.

*Все рациональные числа являются алгебраическими. Среди иррациональных чисел есть как алгебраические, так и трансцендентные.* Например, $\sqrt{2}$ — алгебраическое иррациональное число, а $\pi$ — трансцендентное иррациональное число.

### 2.1.1. Применение графиков в решении уравнений

**Уравнением** с одним неизвестным называется равенство $f(x) = g(x)$, в котором требуется найти неизвестную величину $x$.

Пусть задано квадратное уравнение $x^2 + px + q = 0$.

Перепишем его следующим образом: $x^2 = -px - q$ и построим графики зависимостей

$$y = x^2 \quad \text{и} \quad y = -px - q.$$

График первой зависимости известен — это парабола; вторая зависимость — линейная. В том случае, когда $x$ является решением уравнения, координаты точек обоих графиков равны между собой. Если прямая и парабола пересекаются, то абсциссы точек пересечения являются корнями квадратного уравнения.

**Пример 2.1.** Решить уравнение $4x^2 - 12x + 7 = 0$.

Представим уравнение в виде $x^2 = 3x - 7/4$.

Построим параболу $y = x^2$ и прямую $y = 3x - 7/4$ (рис. 2.1).

Парабола и прямая пересекаются в двух точках с абциссами $x_1 = 0{,}8$ и $x_2 = 2{,}2$.

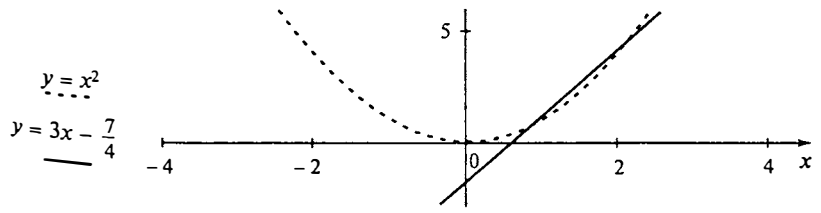



**Рис. 2.1.** Графическое решение уравнения

**Пример 2.2.** Решить уравнение $x^2 - x + 1 = 0$.

Запишем уравнение в виде $x^2 = x - 1$.

Построив параболу $y = x^2$ и прямую $y = x - 1$, увидим, что они не пересекаются (рис. 2.2) — значит, уравнение не имеет корней.

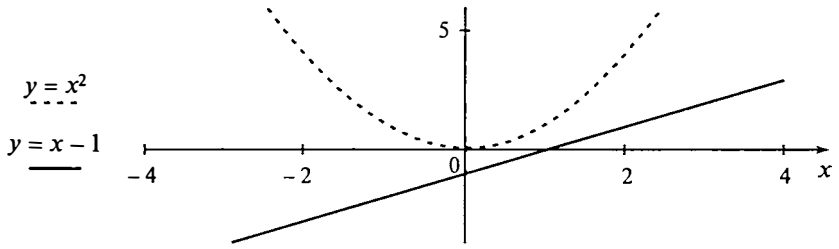



**Рис. 2.2.** Графическое решение уравнения

**Пример 2.3.** Решить систему $\begin{cases} x^2 + y^2 = 25; \\ y = -x^2 + 2x + 5. \end{cases}$

Построим в одной системе координат графики уравнений (рис. 2.3):

$$x^2 + y^2 = 25 \quad \text{и} \quad y = -x^2 + 2x + 5.$$

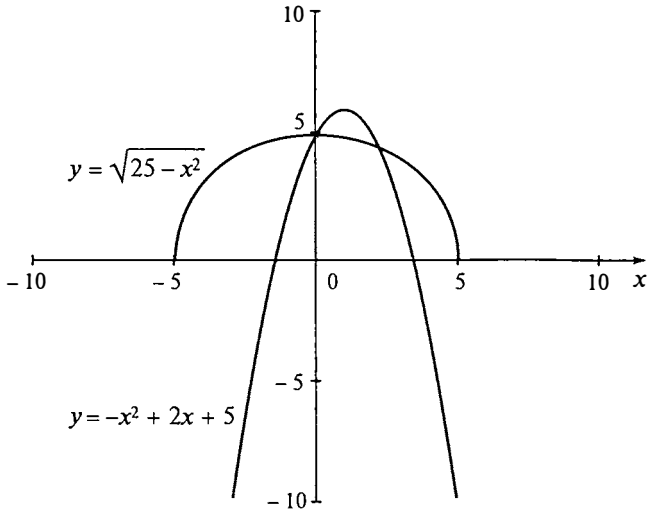



**Рис. 2.3.** Графическое решение системы уравнений

Координаты любой точки построенной окружности являются решением первого уравнения, а координаты любой точки параболы — решением второго уравнения.

Значит, координаты каждой из точек пересечения окружности и параболы удовлетворяют как первому уравнению системы, так и второму, т. е. являются решением рассматриваемой системы. Находим приближенные значения координат точек пересечения графиков: $A(-2,2; -4,5)$, $B(0; 5)$, $C(2,2; 4,5)$, $D(4; -3)$. Следовательно, система уравнений имеет четыре решения:

$$x_1 \approx -2,2, \quad y_1 \approx -4,5; \qquad x_2 \approx 0, \quad y_2 \approx 5;$$

$$x_3 \approx 2,2, \quad y_3 \approx 4,5; \qquad x_4 \approx 4, \quad y_4 \approx -3.$$

Подставив найденные значения в уравнения системы, можно убедиться, что второе и четвертое из этих решений являются точными, а первое и третье — приближенными.

**Пример 2.4**. Решить уравнение $\sin x + \cos x = 1$.

Тригонометрические уравнения решают как аналитически, так и графически. Рассмотрим графический способ решения на примере рис. 2.4.

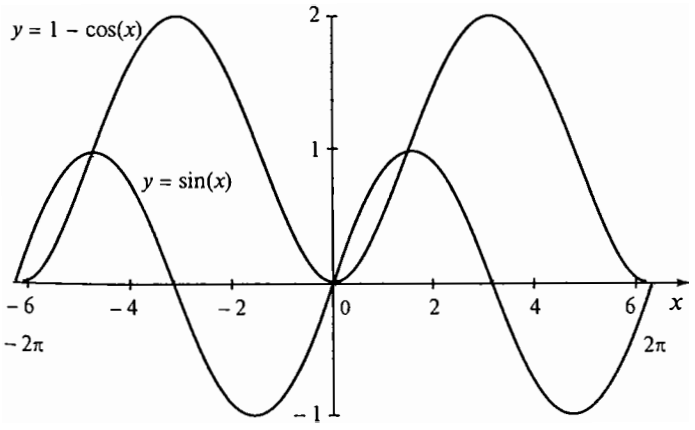



**Рис. 2.4.** Графическое решение уравнения

Построим графики функций $y = \sin x$ и $y = 1 - \cos x$. Из графика видно, что уравнение имеет два решения: $x = 2\pi n$, где $n \in Z$ и $x = \pi/2 + 2\pi k$, где $k \in Z$.

## 2.2. Матрицы

*Матрицей размером* $m \times n$ называется прямоугольная таблица чисел, содержащая $m$ строк и $n$ столбцов. Числа, составляющие матрицу, называются *элементами* матрицы. Обычно принято обозначать матрицы прописными (большими) буквами, а саму таблицу чисел заключать в круглые скобки. Матрицы можно обозначать как

$$\mathbf{A} = (a_{ij}) \ (i = 1, \ ..., \ m, \ j = 1, \ ..., \ n).$$

Например, $\mathbf{A} = \begin{pmatrix} 3 & 1 & 9 \\ 4 & -1 & 7 \end{pmatrix}$ — матрица размером $2 \times 3$;

$\mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ — матрица размером $3 \times 1$, или, другими словами, *матрица-столбец*;

$\mathbf{C} = (1 \quad -1 \quad 2 \quad 4)$ — матрица размером $1 \times 4$, или *матрица-строка*.

Иногда вместо круглых скобок в записи матрицы используют квадратные или двойные прямые линии

$$\begin{bmatrix} 3 & 2 \\ 1 & 4 \\ 1 & 0 \end{bmatrix} \quad \begin{Vmatrix} 1 & 2 \\ -\pi & e \end{Vmatrix}.$$

Для обозначения элементов матрицы используется та же буква, что и для обозначения матрицы, только не прописная, а строчная, и эта буква снабжается двумя индексами. Например, матрицу размером $m \times n$ можно записать в виде

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Если число строк матрицы равно числу столбцов, то матрица называется *квадратной*. Число строк или, что то же самое, число столбцов в ней называется *порядком* матрицы.

Совокупность элементов квадратной матрицы, расположенных на отрезке, соединяющем левый верхний угол с правым нижним, называется *главной диагональю* матрицы. Например, в матрице

$$\mathbf{B} = \begin{pmatrix} 1 & -2 & 1 \\ 0 & 3 & -2 \\ 1 & 0 & -4 \end{pmatrix}.$$

главную диагональ образуют числа {1; 3; –4}.

Квадратная матрица, у которой все элементы вне главной диагонали равны нулю, называется *диагональной*. Примеры диагональных матриц:

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0{,}7 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Квадратная матрица называется *верхней треугольной* (*нижней треугольной*), если все ее элементы, стоящие ниже (выше) главной диагонали, равны нулю. Например, верхние треугольные матрицы:

$$\begin{pmatrix} 1 & 1 & -2 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}, \quad \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}.$$

Нижние треугольные матрицы:

$$\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}.$$

Если $a_{mn} = a_{nm}$, то матрица называется **симметрической**, например $\begin{pmatrix} 2 & 1 & 5 \\ 1 & 3 & 6 \\ 5 & 6 & 4 \end{pmatrix}$.

*Единичной матрицей* называется диагональная матрица, у которой все элементы главной диагонали равны 1. Для обозначения единичной матрицы обычно используется буква **E**.

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Если все элементы матрицы равны нулю, то матрица называется *нулевой*. Матрица называется *обратной* к матрице **A** и обозначается $\mathbf{A}^{-1}$, если

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{E} \text{ (единичной матрице)}.$$

Две матрицы называются *равными*, если они имеют одинаковые размеры и элементы, стоящие на одинаковых местах, равны друг другу.

**Суммой (разностью)** матриц называется матрица, элементами которой являются соответственно сумма (разность) элементов исходных матриц

$$c_{ij} = a_{ij} \pm b_{ij}, \quad \mathbf{C} = \mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}.$$

Операция **умножения (деления)** матрицы любого размера на произвольное число сводится к умножению (делению) каждого элемента матрицы на это число

$$\alpha\mathbf{A} = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \dots & \alpha a_{2n} \\ \dots & \dots & & \dots \\ \alpha a_{m1} & \alpha a_{m2} & \dots & \alpha a_{mn} \end{pmatrix},$$

$$\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} \pm \alpha\mathbf{B},$$

$$\mathbf{A}(\alpha \pm \beta) = \alpha\mathbf{A} \pm \beta\mathbf{A}.$$

**Произведением** матриц называется матрица, элементы которой могут быть вычислены по следующим формулам:

$$\mathbf{A}\mathbf{B} = \mathbf{C};$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}.$$

Из приведенного определения видно, что операция умножения матриц определена только для матриц, число столбцов первой из которых равно числу строк второй.

Если для каких-либо матриц выполняется соотношение $\mathbf{AB} = \mathbf{BA}$, то такие матрицы называются **перестановочными**.

Матрицу $\mathbf{B}$ называют **транспонированной** матрицей к $\mathbf{A}$, а переход от $\mathbf{A}$ к $\mathbf{B}$ — транспонированием, если элементы каждой строки матрицы $\mathbf{A}$ записать в том же порядке в столбцы матрицы $\mathbf{B}$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \dots & \dots & & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}; \quad \mathbf{B} = \mathbf{A}^{\mathrm{T}} = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix},$$

другими словами, $b_{ji} = a_{ij}$. Квадратная матрица $\mathbf{U}$, для которой $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$, называется *ортогональной* матрицей.

**Определителем** квадратной матрицы $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$ называется число, которое может быть вычислено по элементам матрицы по формуле

$$\det \mathbf{A} = \sum_{k=1}^{n} (-1)^{k+1} a_{1k} M_{1k}, \qquad (2.1)$$

где $M_{1k}$ — детерминант матрицы, полученной из исходной вычеркиванием первой строки и $k$-го столбца. Следует обратить внимание на то, что определители имеют только квадратные матрицы, т. е. матрицы, у которых число строк равно числу столбцов. Формула (2.1) позволяет вычислить определитель матрицы по первой строке, также справедлива формула вычисления определителя по первому столбцу:

$$\det \mathbf{A} = \sum_{k=1}^{n} (-1)^{k+1} a_{k1} M_{k1}.$$

Вообще говоря, определитель может вычисляться по любой строке или столбцу матрицы, т. е. справедлива формула

$$\det \mathbf{A} = \sum_{k=1}^{n} (-1)^{k+i} a_{ik} M_{ik}, \quad i = 1, 2, \ldots, n.$$

Очевидно, что различные матрицы могут иметь одинаковые определители. Определитель единичной матрицы равен 1. Для указанной матрицы **A** число $M_{ik}$ называется **дополнительным минором** элемента матрицы $a_{ik}$. Таким образом, можно заключить, что каждый элемент матрицы имеет свой дополнительный минор. Дополнительные миноры существуют только в квадратных матрицах.

**Пример 2.5.** Вычислить сумму матриц:

$$\begin{pmatrix} -1 & 3 & 5 \\ -2 & 3 & 7 \\ 4 & 6 & 8 \end{pmatrix} \text{ и } \begin{pmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{pmatrix}.$$

***Решение***

$$\begin{pmatrix} -1 & 3 & 5 \\ -2 & 3 & 7 \\ 4 & 6 & 8 \end{pmatrix} + \begin{pmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 10 & 7 \\ 2 & -2 & 5 \\ 6 & 3 & 8 \end{pmatrix}.$$

$-1 + 2 = 1$; $3 + 7 = 10$; $5 + 2 = 7$;
$-2 + 4 = 2$; $3 + (-5) = -2$; $7 + (-2) = 5$;
$4 + 2 = 6$; $6 + (-3) = 3$; $8 + 0 = 8$.

**Пример 2.6.** Вычислить произведения матриц: а) **AB**; б) **CD**.

$$\mathbf{A} = \begin{pmatrix} -3 & 2 & 4 & 3 \\ 3 & -2 & 3 & 1 \\ 6 & -5 & 2 & 4 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 3 & -4 \\ -3 & 5 \\ 2 & 6 \\ -2 & 1 \end{pmatrix},$$

$$\mathbf{C} = \begin{pmatrix} 2 & 7 & 3 \\ -2 & 5 & 2 \\ 4 & 1 & 5 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 3 & -3 & 2 & -2 \\ 2 & -2 & 1 & 4 \\ 5 & 4 & -1 & 0 \end{pmatrix}.$$

***Решение***

а) $\begin{pmatrix} -3 & 2 & 4 & 3 \\ 3 & -2 & 3 & 1 \\ 6 & -5 & 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 & -4 \\ -3 & 5 \\ 2 & 6 \\ -2 & 1 \end{pmatrix} = \begin{pmatrix} -13 & 49 \\ 19 & -3 \\ 29 & -33 \end{pmatrix}$.

$-3 \cdot 3 + 2 \cdot (-3) + 4 \cdot 2 + 3 \cdot (-2) = -9 - 6 + 8 - 6 = -13$;
$-3 \cdot (-4) + 2 \cdot 5 + 4 \cdot 6 + 3 \cdot 1 = 12 + 10 + 24 + 3 = 49$;
$3 \cdot 3 + (-2) \cdot (-3) + 3 \cdot 2 + 1 \cdot (-2) = 9 + 6 + 6 - 2 = 19$;
$3 \cdot (-4) + (-2) \cdot 5 + 3 \cdot 6 + 1 \cdot 1 = -12 - 10 + 18 + 1 = -3$;
$6 \cdot 3 + (-5) \cdot (-3) + 2 \cdot 2 + 4 \cdot (-2) = 18 + 15 + 4 - 8 = 29$;
$6 \cdot (-4) + (-5) \cdot 5 + 2 \cdot 6 + 4 \cdot 1 = -20 - 25 + 12 + 4 = -33$.

б) $\begin{pmatrix} 2 & 7 & 3 \\ -2 & 5 & 2 \\ 4 & 1 & 5 \end{pmatrix} \cdot \begin{pmatrix} 3 & -3 & 2 & -2 \\ 2 & -2 & 1 & 4 \\ 5 & 4 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 35 & -8 & 8 & 24 \\ 14 & 4 & -1 & 24 \\ 39 & 6 & 4 & -4 \end{pmatrix}$.

$2 \cdot 3 + 7 \cdot 2 + 3 \cdot 5 = 35$;
$2 \cdot (-3) + 7 \cdot (-2) + 3 \cdot 4 = -8$;
$2 \cdot 2 + 7 \cdot 1 + 3 \cdot (-1) = 8$;
$2 \cdot (-2) + 7 \cdot 4 + 3 \cdot 0 = 24$;
$-2 \cdot 3 + 5 \cdot 2 + 2 \cdot 5 = 14$;
$-2 \cdot (-3) + 5 \cdot (-2) + 2 \cdot 4 = 4$;
$-2 \cdot 2 + 5 \cdot 1 + 2 \cdot (-1) = -1$;
$-2 \cdot (-2) + 5 \cdot 4 + 2 \cdot 0 = 24$;
$4 \cdot 3 + 1 \cdot 2 + 5 \cdot 5 = 39$;
$4 \cdot (-3) + 1 \cdot (-2) + 5 \cdot 4 = 6$;
$4 \cdot 2 + 1 \cdot 1 + 5 \cdot (-1) = 4$;
$4 \cdot (-2) + 1 \cdot 4 + 5 \cdot 0 = -4$.

**Пример 2.7**. Вычислить произведение матриц **AB**:

а) $\mathbf{A} = \begin{pmatrix} 5 & 4 & -2 & 3 \\ -2 & 5 & 6 & 1 \\ -4 & 2 & -3 & 5 \\ 3 & 1 & 8 & 0 \end{pmatrix}$, $\mathbf{B} = \begin{pmatrix} 2 \\ -4 \\ 6 \\ 5 \end{pmatrix}$;

б) $\mathbf{A} = (7 \quad -3 \quad 9 \quad 2 \quad 4)$, $\mathbf{B} = \begin{pmatrix} 4 & 3 & -2 \\ -2 & 9 & 4 \\ -3 & -5 & 7 \\ 1 & 8 & 3 \\ 5 & 1 & 0 \end{pmatrix}$.

***Решение***

а) $\begin{pmatrix} 5 & 4 & -2 & 3 \\ -2 & 5 & 6 & 1 \\ -4 & 2 & -3 & 5 \\ 3 & 1 & 8 & 0 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ -4 \\ 6 \\ 5 \end{pmatrix} = \begin{pmatrix} -3 \\ 17 \\ -9 \\ 50 \end{pmatrix}$.

б) $(7 \quad -3 \quad 9 \quad 2 \quad 4) \cdot \begin{pmatrix} 4 & 3 & -2 \\ -2 & 9 & 4 \\ -3 & -5 & 7 \\ 1 & 8 & 3 \\ 5 & 1 & 0 \end{pmatrix} = (9 \quad -31 \quad 43)$.

**Пример 2.8.** Вычислить произведение: а) $\mathbf{AB}^{\mathrm{T}}$; б) $\mathbf{B}^{\mathrm{T}}\mathbf{A}$. $\mathbf{A} = (9 \quad 6 \quad 3 \quad 1)$, $\mathbf{B} = (-2 \quad 3 \quad -5 \quad 7)$.

***Решение***

а) $(9 \quad 6 \quad 3 \quad 1) \cdot \begin{pmatrix} -2 \\ 3 \\ -5 \\ 7 \end{pmatrix} = -8$.

б) $\begin{pmatrix} -2 \\ 3 \\ -5 \\ 7 \end{pmatrix} \cdot (9 \quad 6 \quad 3 \quad 1) = \begin{pmatrix} -18 & -12 & -6 & -2 \\ 27 & 18 & 9 & 3 \\ -45 & -30 & -15 & -5 \\ 63 & 42 & 21 & 7 \end{pmatrix}$.

**Пример 2.9.** Даны матрицы $\mathbf{A} = \begin{pmatrix} 1 & 0 & 3 \\ 2 & 4 & 1 \\ 1 & -4 & 2 \end{pmatrix}$, $\mathbf{B} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$, $\mathbf{C} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}$ и число $\alpha = 2$. Найти $\mathbf{A}^{\mathrm{T}}\mathbf{B} + \alpha\mathbf{C}$.

***Решение***

$$\mathbf{A}^{\mathrm{T}} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 4 & -4 \\ 3 & 1 & 2 \end{pmatrix};$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{B} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 4 & -4 \\ 3 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 1\cdot 1 + 2\cdot 3 + 1\cdot 2 \\ 0\cdot 1 + 4\cdot 3 - 4\cdot 2 \\ 3\cdot 1 + 1\cdot 3 + 2\cdot 2 \end{pmatrix} = \begin{pmatrix} 9 \\ 4 \\ 10 \end{pmatrix};$$

$$\alpha\mathbf{C} = \begin{pmatrix} -2 \\ 4 \\ 2 \end{pmatrix}; \quad \mathbf{A}^{\mathrm{T}}\mathbf{B} + \alpha\mathbf{C} = \begin{pmatrix} 9 \\ 4 \\ 10 \end{pmatrix} + \begin{pmatrix} -2 \\ 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \\ 12 \end{pmatrix}.$$

**Пример 2.10.** Вычислить определитель матрицы $\mathbf{A} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 3 \\ 3 & 1 & 1 \end{pmatrix}$.

***Решение***

$$\begin{vmatrix} 1 & 2 & 1 \\ 0 & -2 & 3 \\ 3 & 1 & 1 \end{vmatrix} = 1 \cdot \begin{vmatrix} -2 & 3 \\ 1 & 1 \end{vmatrix} - 2 \cdot \begin{vmatrix} 0 & 3 \\ 3 & 1 \end{vmatrix} + 1 \cdot \begin{vmatrix} 0 & -2 \\ 3 & 1 \end{vmatrix} =$$

$$= (-2 \cdot 1 - 1 \cdot 3) - 2(0 \cdot 1 - 3 \cdot 3) + (0 \cdot 1 + 3 \cdot 2) = -5 + 18 + 6 = 19.$$

**Пример 2.11.** Даны матрицы $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $\mathbf{B} = \begin{pmatrix} 5 & 2 \\ 1 & 3 \end{pmatrix}$.

Найти det (**AB**).

***Решение***

*1-й способ:*

det **A** = 4 – 6 = –2; det **B** = 15 – 2 = 13;

det (**AB**) = det **A** · det **B** = –26.

*2-й способ:*

$$\mathbf{AB} = \begin{pmatrix} 1\cdot 5 + 2\cdot 1 & 1\cdot 5 + 2\cdot 3 \\ 3\cdot 5 + 4\cdot 1 & 3\cdot 2 + 4\cdot 3 \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 19 & 18 \end{pmatrix};$$

det (**AB**) = 7 · 18 – 8 · 19 = 126 – 152 = –26.

**Пример 2.12.** Вычислить определители:

а) $\begin{vmatrix} -3 & 5 \\ 2 & -2 \end{vmatrix}$;

б) по правилу треугольников $\begin{vmatrix} 2 & -2 & -1 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix}$;

в) разложением по первой строке $\begin{vmatrix} 2 & 3 & 5 \\ -7 & 10 & 0 \\ -2 & -1 & 15 \end{vmatrix}$;

г) разложением по первому столбцу $\begin{vmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{vmatrix}$;

д) свести матрицу к определителям меньшего порядка, упрощая с помощью линейной комбинации строк и столбцов

$$\begin{vmatrix} 5 & 4 & -2 & 3 \\ -2 & 5 & 6 & 1 \\ -4 & 2 & -3 & 5 \\ 3 & 1 & 8 & 0 \end{vmatrix}.$$

***Решение***

а) $\Delta = \begin{vmatrix} -3 & 5 \\ 2 & -2 \end{vmatrix} = -3 \cdot (-2) - 5 \cdot 2 = 6 - 10 = -4;$

б) $\Delta = \begin{vmatrix} 2 & -2 & -1 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix} = 2 \cdot 4 \cdot 2 + (-2) \cdot (-3) \cdot 1 + (-1) \cdot 3 \cdot 5 -$

$- (-1) \cdot 4 \cdot 1 - 2 \cdot (-3) \cdot 5 - (-2) \cdot 3 \cdot 2 = 53;$

в) $\Delta = \begin{vmatrix} 2 & 3 & 5 \\ -7 & 10 & 0 \\ -2 & -1 & 15 \end{vmatrix} = 2 \cdot (-1)^{1+1} \begin{vmatrix} 10 & 0 \\ -1 & 15 \end{vmatrix} + 3 \cdot (-1)^{1+2} \begin{vmatrix} -7 & 0 \\ -2 & 15 \end{vmatrix} +$

$+ 5 \cdot (-1)^{1+3} \begin{vmatrix} -7 & 10 \\ -2 & -1 \end{vmatrix} = 2 \cdot (10 \cdot 15 - (-1) \cdot 0) - 3 \cdot (-7 \cdot (-1) -$

$- (-2) \cdot 10) = 300 + 315 + 135 = 750;$

г) $\Delta = \begin{vmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{vmatrix} = 2 \cdot (-1)^{1+1} \begin{vmatrix} -5 & -2 \\ -3 & 0 \end{vmatrix} + 4 \cdot (-1)^{1+2} \begin{vmatrix} 7 & 2 \\ -3 & 0 \end{vmatrix} +$

$+ 2 \cdot (-1)^{1+3} \begin{vmatrix} 7 & 2 \\ -5 & -2 \end{vmatrix} = 2 \cdot (-5 \cdot 0 - (-3) \cdot (-2) - 4 \cdot (7 \cdot 0 -$

$- (-3) \cdot 2) + 2 \cdot (7 \cdot (-2) - (-5) \cdot 2) = -12 - 24 - 8 = -44;$

д) $\Delta = \begin{vmatrix} 5 & 4 & -2 & 3 \\ -2 & 5 & 6 & 1 \\ -4 & 2 & -3 & 5 \\ 3 & 1 & 8 & 0 \end{vmatrix} = \begin{vmatrix} 11 & -11 & -20 & 0 \\ -2 & 5 & 6 & 1 \\ 6 & -23 & -33 & 0 \\ 3 & 1 & 8 & 0 \end{vmatrix} =$

$= 1 \cdot (-1)^{2+4} \begin{vmatrix} 11 & -11 & -20 \\ 6 & -23 & -33 \\ 3 & 1 & 8 \end{vmatrix} = \begin{vmatrix} 44 & -11 & 68 \\ 75 & -23 & 151 \\ 0 & 1 & 0 \end{vmatrix} =$

$= 1 \cdot (-1)^{3+2} \begin{vmatrix} 44 & 68 \\ 75 & 151 \end{vmatrix} = -1 \cdot (44 \cdot 151 - 75 \cdot 68) = -1544.$

**Пример 2.13.**

а) Вычислить сумму двух определителей:

$\begin{vmatrix} 2 & 7 & 3 \\ -2 & 5 & 2 \\ 4 & 1 & 5 \end{vmatrix}$ и $\begin{vmatrix} 2 & 7 & 3 \\ 4 & -2 & -2 \\ 4 & 1 & 5 \end{vmatrix}$;

б) Представить определитель $\Delta = \begin{vmatrix} 2 & -2 & -1 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix}$ в виде суммы двух определителей.

***Решение***

а) $\Delta = \begin{vmatrix} 2 & 7 & 3 \\ -2 & 5 & 2 \\ 4 & 1 & 5 \end{vmatrix} + \begin{vmatrix} 2 & 7 & 3 \\ 4 & -2 & -2 \\ 4 & 1 & 5 \end{vmatrix} = \begin{vmatrix} 2 & 7 & 3 \\ 2 & 3 & 0 \\ 4 & 1 & 5 \end{vmatrix} =$

$= 2 \cdot 3 \cdot 5 + 7 \cdot 0 \cdot 4 + 3 \cdot 2 \cdot 1 - 3 \cdot 3 \cdot 4 - 2 \cdot 0 \cdot 1 - 7 \cdot 2 \cdot 5 = -70;$

б) $\Delta = \begin{vmatrix} 2 & -2 & -1 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix} = \begin{vmatrix} 1 & -1 & -1 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix} + \begin{vmatrix} 1 & -1 & 0 \\ 3 & 4 & -3 \\ 1 & 5 & 2 \end{vmatrix}$.

**Пример 2.14.** Представить определитель $\Delta = \begin{vmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{vmatrix}$ в виде произведения определителя на число.

***Решение***

$$\Delta = \begin{vmatrix} 2 & 7 & 2 \\ 4 & -5 & -2 \\ 2 & -3 & 0 \end{vmatrix} = 2 \begin{vmatrix} 1 & 7 & 2 \\ 2 & -5 & -2 \\ 1 & -3 & 0 \end{vmatrix}.$$

## 2.3. Алгебраические уравнения

**Уравнение** — аналитическая запись задачи о разыскании значений аргументов, при которых значения двух данных функций равны $f(x, y, ...) = g(x, y, ...)$. Аргументы, от которых зависят эти функции, называются обычно неизвестными, а значения неизвестных, при которых значения функций равны, — *решениями* или *корнями*. О таких значениях неизвестных говорят, что они удовлетворяют данному уравнению.

**Алгебраические уравнения** имеют следующий вид:

$$P(x_1, ..., x_n) = Q(x_1, ..., x_n),$$

где $P$ и $Q$ — многочлены с коэффициентами из поля рациональных чисел.

**Линейное уравнение** — это уравнение, обе части которого могут быть выражены многочленами (от неизвестных) первой степени.

Линейное уравнение можно привести к виду: $ax + b = 0$, где $a$ — ненулевой параметр, $b$ — произвольный параметр.

Линейное уравнение имеет единственное решение $x = -\frac{b}{a}$.

**Квадратное уравнение** — уравнение вида $ax^2 + bx + c = 0$, где $a \neq 0$.

В общем случае уравнение решается так:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Число $D = b^2 - 4ac$ называется дискриминантом многочлена $ax^2 + bx + c = 0$. Если $D > 0$, то уравнение имеет два различных вещественных корня. Если $D = 0$, то оба корня вещественны и равны. Если $D < 0$, то оба корня являются комплексными числами. Если коэффициент перед $x$ является четным числом, то уравнение можно записать в виде $ax^2 + 2kx + c = 0$. В этом случае рациональнее воспользоваться для решения следующей формулой:

$$D = k^2 - ac, \quad x_{1,2} = \frac{-k \pm \sqrt{D}}{a} \quad \text{при } D \geq 0.$$

Квадратное уравнение вида $x^2 + px + q = 0$, в котором ведущий коэффициент равен единице, называют *приведенным*.

В приведенном случае решение выглядит следующим образом:

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

**Теорема Виета.** Сумма корней приведенного квадратного уравнения $x^2 + px + q = 0$ равна коэффициенту $p$, взятому с обратным знаком, а произведение корней равно свободному члену $q$:

$$x_1 + x_2 = -p;$$

$$x_1 x_2 = q.$$

В случае неприведенного квадратного уравнения $ax^2 + bx + c = 0$:

$$x_1 + x_2 = -\frac{b}{a}; \quad x_1 x_2 = \frac{c}{a}.$$

**Кубическое уравнение** — уравнение вида $ax^3 + bx^2 + cx + d = 0$, где $a \neq 0$.

Заменяя в этом уравнении $x$ новым неизвестным $y$, связанным с $x$ равенством $x = y - \frac{b}{3a}$, уравнение можно привести к более простому (каноническому) виду:

$$y^3 + py + q = 0,$$

где

$$p = -\frac{b}{3a^2} + \frac{c}{a}, \quad q = \frac{2b}{27a^3} - \frac{bc}{3a^2} + \frac{d}{a}.$$

Решение этого уравнения можно получить с помощью формулы Кардано.

В табл. 2.1 приведены решения простейших уравнений.

*Таблица 2.1.* **Решения простейших уравнений**

| Уравнение | Условие | Решение |
|---|---|---|
| 1. $ax^2 + bx + c = 0, \quad a \neq 0$ | $D > 0$ | $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$ |
| | $D = 0$ | $x = -\frac{b}{2a}$ |
| | $D < 0$ | Корней нет |
| 2. $\lvert x \rvert = b$ | $b > 0$ | $x_1 = b, \ x_2 = -b$ |
| | $b = 0$ | $x = 0$ |
| | $b < 0$ | Корней нет |
| 3. $\sqrt{x} = b$ | $b \geq 0$ | $x = b^2$ |
| | $b < 0$ | Корней нет |
| 4. $x^{2m} = b, \quad m \in N$ | $b > 0$ | $x_1 = \sqrt[2m]{b}, \ x_2 = -\sqrt[2m]{b}$ |
| | $b = 0$ | $x = 0$ |
| | $b < 0$ | Корней нет |
| 5. $a^x = b, \quad a > 0, \quad a \neq 1$ | $b > 0$ | $x = \log_a b$ |
| | $b \leq 0$ | Корней нет |
| 6. $\log_a x = b, \quad a > 0, \quad a \neq 1$ | $b$ — любое | $x = a^b$ |

*Окончание табл. 2.1*

| Уравнение | Условие | Решение |
|---|---|---|
| 7. $\sin x = b$ | $\lvert b\rvert \le 1$ | $x = (-1)^n \arcsin b + \pi n,\ n \in Z$ |
| | $\lvert b\rvert > 1$ | Корней нет |
| 8. $\cos x = b$ | $\lvert b\rvert \le 1$ | $x = \pm \arccos b + 2\pi n,\ n \in Z$ |
| | $\lvert b\rvert > 1$ | Корней нет |
| 9. $\operatorname{tg} x = b$ | $b$ — любое | $x = \operatorname{arctg} b + \pi n,\ n \in Z$ |
| 10. $\arcsin x = b$ | $\lvert b\rvert \le \frac{\pi}{2}$ | $x = \sin b$ |
| | $\lvert b\rvert > \frac{\pi}{2}$ | Корней нет |

**Пример 2.15.** Решить уравнение $\sqrt{x^2 - 5} = 2$.

***Решение***

Возведем обе части этого уравнения в квадрат и получим $x^2 - 5 = 4$, откуда следует, что $x^2 = 9$, т. е. $x = \pm 3$.

Проверим, что полученные числа являются решениями уравнения. Действительно, при подстановке их в данное уравнение получаются верные равенства:

$$\sqrt{3^2 - 5} = 2 \quad \text{и} \quad \sqrt{(-3)^2 - 5} = 2.$$

Следовательно, $x = 3$ или $x = -3$ — решения данного уравнения.

**Пример 2.16.** Решить уравнение $\sqrt{x} = x - 2$.

***Решение***

Возведем в квадрат обе части уравнения, получим $x = x^2 - 4x + 4$. После преобразований приходим к квадратному уравнению $x^2 - 5x + 4 = 0$, корни которого $x = 1$ и $x = 4$.

Проверим, являются ли найденные числа решениями данного уравнения. При подстановке в него числа 4 получим верное равенство $\sqrt{4} = 4 - 2$, т. е. 4 — решение данного уравнения. При подстановке же числа 1 получаем в правой части $-1$, а в левой части число 1. Следовательно, 1 не является решением уравнения; говорят, что это посторонний корень, полученный в результате принятого способа решения.

*Ответ*: $x = 4$.

**Пример 2.17.** Решить уравнение $x - 1 = \sqrt[3]{x^2 - x - 1}$.

***Решение***

В отличие от рассмотренных ранее примеров, данное иррациональное уравнение содержит не квадратный корень, а корень третьей степени. Поэтому для того чтобы «избавиться от радикала», надо возвести обе части уравнения не в квадрат, а в куб:

$$(x - 1)^3 = x^2 - x - 1.$$

После преобразований получаем:

$$x^2 - 3x^2 + 3x - 1 = x^2 - x - 1;$$

$$x^3 - 4x^2 + 4x = 0;$$

$$x(x^2 - 4x + 4) = 0;$$

$$x(x - 2)^2 = 0.$$

Отсюда $x_1 = 0$, $x_2 = 2$.

**Диофантово уравнение**, или **уравнение в целых числах** — это уравнение с целыми коэффициентами и неизвестными, которые могут принимать только целые значения.

Общий вид линейного диофантова уравнения:

$$ax + by + \ldots + cz = d.$$

В литературе под диофантовыми уравнениями понимаются также уравнения более частного вида (с двумя неизвестными):

$$ax + by = c.$$

### 2.3.1. Уравнения с одним и двумя неизвестными

Рассмотрим уравнение первой степени с одним неизвестным

$$a_1 x + a_0 = 0. \qquad (2.2)$$

Пусть коэффициенты уравнения $a_1$ и $a_0$ — целые числа. Ясно, что решение этого уравнения

$$x = -\frac{a_0}{a_1}$$

будет целым числом только в том случае, когда $a_0$ нацело делится на $a_1$. Таким образом, уравнение (2.2) не всегда разрешимо в целых числах; так, например, из двух уравнений $3x - 27 = 0$ и $5x + 21 = 0$ первое имеет целое решение $x = 9$, а второе в целых числах неразрешимо.

С тем же обстоятельством мы встречаемся и в случае уравнений, степень которых выше первой: квадратное уравнение $x^2 + x - 2 = 0$ имеет целые решения $x_1 = 1$, $x_2 = -2$; уравнение $x^2 + x - 2 = 0$ в целых числах неразрешимо, так как его корни $x_{1,2} = 2 \pm \sqrt{2}$ иррациональны.

Вопрос о нахождении целых корней уравнения $n$-й степени с целыми коэффициентами

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0 \quad (n \geq 1) \tag{2.3}$$

решается легко. Действительно, пусть $x = a$ — целый корень этого уравнения. Тогда

$$a_n a^n + a_{n-1} a^{n-1} + \ldots + a_1 a + a_0 = 0;$$

$$a_0 = -a(a_n a^{n-1} + a_{n-1} a^{n-2} + \ldots + a_1).$$

Из последнего равенства видно, что $a_0$ делится на $a$ без остатка; следовательно, каждый целый корень уравнения (2.3) является делителем свободного члена уравнения. Для нахождения целых решений уравнения надо выбрать те из делителей $a_0$, которые при подстановке в уравнение обращают его в тождество. Так, например, из чисел 1, –1, 2 и –2, представляющих собой все делители свободного члена уравнения

$$x^{10} + x^7 + 2x^3 + 2 = 0,$$

только –1 является корнем. Следовательно это уравнение имеет единственный целый корень $x = -1$. Тем же методом легко показать, что уравнение

$$x^6 - x^5 + 3x^4 + x^2 - x + 3 = 0$$

в целых числах неразрешимо.

Значительно больший интерес представляет решение в целых числах уравнений с многими неизвестными.

Рассмотрим уравнение первой степени с двумя неизвестными

$$ax + by + c = 0, \tag{2.4}$$

где $a$ и $b$ — целые числа, отличные от нуля, а $c$ — произвольное целое. Будем считать, что коэффициенты $a$ и $b$ не имеют общих делителей, кроме единицы. Действительно, если общий наибольший делитель этих коэффициентов $d = (a, b)$ отличен от единицы, то справедливы равенства $a = a_1 d$, $b = b_1 d$; уравнение (2.4) принимает вид

$$(a_1 x + b_1 y)d + c = 0 \tag{2.5}$$

и может иметь целые решения только в том случае, когда $c$ делится на $d$. Таким образом, в случае $(a, b) = d \neq 1$ — все коэффициенты уравнения (2.5) должны делиться нацело на $d$, и, сокращая на $d$, придем к уравнению

$$a_1 x + b_1 y + c_1 = 0 \quad \left(c_1 = \frac{c}{d}\right), \tag{2.6}$$

коэффициенты которого $a_1$ и $b_1$ взаимно просты.

Рассмотрим сначала случай, когда $c = 0$. Уравнение (2.6) перепишется следующим образом:

$$ax + by = 0.$$

Решая это уравнение относительно $x$, получим

$$x = -\frac{b}{a} y.$$

Ясно, что $x$ будет принимать целые значения в том и только в том случае, когда $y$ делится на $a$ без остатка. Но всякое целое $y$, кратное $a$, можно записать в виде

$$y = at,$$

где $t$ принимает произвольные целые значения ($t = 0, \pm 1, \pm 2, \ldots$). Подставим это значение $y$ в предыдущее уравнение, тогда

$$x = -\frac{b}{a} at = -bt,$$

получим формулы, содержащие все целые решения уравнения:

$$x = -bt, \quad y = at \quad (t = 0, \pm 1, \pm 2, \ldots).$$

Перейдем теперь к случаю, когда $c \neq 0$.

Покажем, что для нахождения всех целых решений уравнения достаточно найти какое-нибудь одно его решение, т. е. найти такие целые числа $x_0$, $y_0$, для которых

$$ax_0 + by_0 + c = 0.$$

Введем обозначения:

$n$ — степень многочлена; $a$ — коэффициент при $x$; $c$ — свободный член уравнения; $d$ — делитель свободного члена; $w$ — вспомогательная переменная для возведения $d$ в степень аргумента; $x$ — сумма $w$, умноженных на $a$.

**Программа (уравнения с одним неизвестным)**

```
program matan1;
Uses Crt;
var    i,n,c,j,k,x,w,q,p:integer;
       a,d:array[1..100] of integer;
Begin
Clrscr;
writeln ('введите степень многочлена');  readln (n);
  for i:=1 to  n+1 do begin
  if i=n+1 then begin writeln ('введите свободный коэффициент');
  read (c); end;
  if i<>n+1 then begin Writeln ('введите коэффициент при x^', n-i+1);
  readln (a[i]); end; end;
w:=1;
for j:=1 to c do begin
  if c/j= (c div j) then begin d[j]:= -j;   k:=n;
     for i:=1 to n do begin
       for q:=1 to k do
       w:=w*d[j];  x:=x+w*a[i]; k:=k-1;w:=1;
       end;
  if  x+c=0  then begin  p:=p+1;
writeln('целый корень уравнения =',d[j]);end;
end;  x:=0;  end;
for j:=1 to c do begin
  if c/j= (c div j) then begin d[j]:=j;   k:=n;
     for i:=1 to n do begin
       for q:=1 to k do
       w:=w*d[j];  x:=x+w*a[i];  k:=k-1;w:=1;
       end;
```

```
  if x+c=0 then begin p:=p+1;
writeln('целый корень уравнения =',d[j]);end;
end;  x:=0;  end;
if  p=0 then writeln ('данное уравнение в целых числах неразрешимо');
readln;  readln;
End.
```

**Программа (уравнения первой степени с двумя неизвестными)**

```
program matan2;
Uses Crt;
var      p,q,t,n,i,k,x,y,w,r,s,d:integer;
         a,b,c:array[1..1000]of integer;
Begin
  Clrscr;
  writeln('вв. при x'); readln(p);
  writeln('вв. при y'); readln(q);
  writeln('вв. при c'); readln(t);
if p<0 then x:=-p else x:=p; if q<0 then y:=-q else y:=q;
n:=0; n:=0; k:=1;
for i:=1 to 10 do begin
  if k<>0 then begin n:=n+1;
  for i:=n to n do begin
    a[i]:=x; b[i]:=y;  c[i]:=x div y;  x:=x-c[i]*y;
    k:=k+1; n:=0; r:=r+1;
    if (x<y) and (x<>1) then begin w:=y; y:=x; x:=w; end else k:=0;
  end; end; end;
x:=p;  y:=q;
  for i:=1 to r do begin
  a[i]:=x; b[i]:=y; c[i]:=x div y; x:=x-c[i]*y; a[i]:=1; b[i]:=1;
  if (x<y) and (x<>1) then begin w:=y; y:=x; x:=w;end;
  end;
for i:=r downto 1 do begin
  b[r]:=0;  b[i]:=c[i]*b[i]+a[i];
  if i>1 then b[i-1]:=b[i];  if i>2 then a[i-2]:=b[i-1];
  end;
if (p*b[1]+q*a[1]+t)=0 then begin
writeln('корни уравнения x=',b[1],'y=',a[1]);
writeln ('все его решения будут содержаться в прогрессиях');
writeln('x=',b[1],'+',q,'*','t');  writeln('y=',a[1],'+',p,'*','t'); end;
readln;
End.
```

### 2.3.2. Численные методы решения уравнений

Во многих практически важных случаях, когда уравнение имеет сложный вид, аналитически его точное решение найти не удается. Отсутствуют методы решения в общем виде алгебраических уравнений высоких степеней. Для трансцендентных уравнений точное решение можно найти в немногих самых простых случаях.

Если решение нельзя найти в явном виде, то для отыскания корня используют другие методы. Например, приближенное решение можно получить методом последовательных приближений. Сравнительно легко корни уравнения определяются графически — достаточно лишь для уравнения $f(x) = 0$ построить график функции $y = f(x)$ и найти точки пересечения кривой с осью абсцисс, в которых эта функция равна нулю. Наконец, корень уравнения можно попытаться определить «методом подбора».

Однако ни один из перечисленных подходов нельзя считать достаточно *эффективным* при решении инженерных и научных задач на ПЭВМ. Более предпочтительны способы, обеспечивающие одновременно как оперативность получения результата, так и высокую точность.

Второе важное требование к методу — *универсальность*, т. е. способность находить решения для разных видов уравнений. В особенности эти требования должны соблюдаться в специальных пакетах программ, предназначенных для выполнения большого объема расчетов, например в системах автоматизированного проектирования (САПР).

В связи с этим для решения нелинейных уравнений на ПЭВМ широко применяются специальные методы, которые относятся к методам *вычислительной математики*. На их основе создано большое число алгоритмов, различающихся сложностью и эффективностью.

Когда говорят о методах решения нелинейных уравнений на ПЭВМ, то подразумевают в первую очередь *итерационные методы*. Главным признаком итерационного метода является многократное повторение одного и того же набора действий для получения результата.

В основе итерационного метода лежит итерационная, т. е. повторяемая процедура. Процедура эта строится таким образом,

что после каждого ее выполнения производится очередное приближение к корню. Можно сказать, что итерационный метод несколько напоминает отыскание корня подбором, однако этот подбор производится не наугад, а по вполне определенному алгоритму.

При решении практических задач обычно приходится проводить предварительное исследование уравнения до его решения. Дело в том, что если уравнение не удается решить аналитически, то заранее трудно определить, сколько оно имеет корней и какова их природа — сколько из них комплексных или вещественных, сколько отрицательных или положительных. Поиск корней наугад без предварительного исследования чреват тем, что правильный ответ так и не будет найден. Кроме того, зачастую некоторые корни не имеют физического смысла, и поэтому нет необходимости определять их точные значения.

Однако в ходе предварительного исследования уравнения можно, не вычисляя точных значений всех корней, сразу выбрать из них те, которые представляют наибольший интерес.

Примерное положение корней уравнения $f(x) = 0$ на числовой оси легко определить, построив график функции $y = f(x)$. Точки пересечения кривой $y = f(x)$ с осью абсцисс, где $y = 0$, и будут соответствовать искомым корням.

Построенный график позволяет провести отделение указанных корней, т. е. найти на оси $x$ границы отрезков, в каждом из которых располагается не более одного корня.

Решение многих практических задач сводится к решению уравнений

$$f(x) = 0, \tag{2.7}$$

где функция $f(x)$ определена и непрерывна на некотором интервале.

Если функция $f(x)$ представляет собой многочлен, то уравнение (2.7) называется **алгебраическим**. Если в функцию $f(x)$ входят трансцендентные (тригонометрические, логарифмические, показательные и т. д.) функции, то уравнение (2.7) называется **трансцендентным**.

Для решения алгебраических уравнений любой степени и трансцендентных уравнений разработаны численные методы.

**Пример 2.18.** Отделить корень уравнения $\cos x = 2x$.

***Решение***

Построим графики функций $y = \cos x$ и $y = 2x$.

Из рис. 2.5 видно, что уравнение имеет единственный корень, принадлежащий отрезку [0; 1]. Когда находится отрезок, внутри которого расположен корень, то этот этап решения называется *этапом отделения корня*.

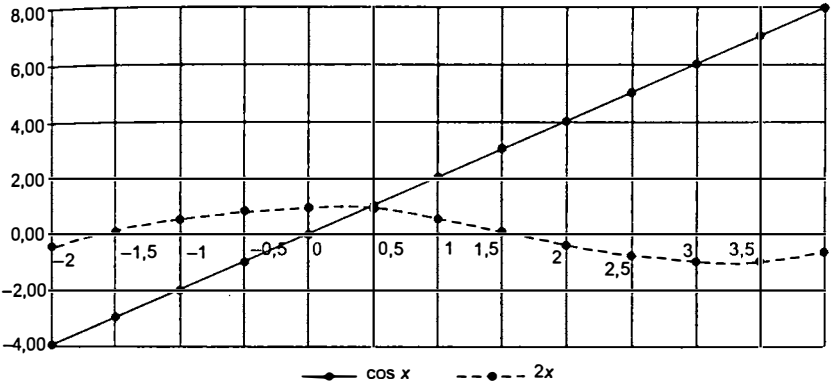



**Рис. 2.5.** Графики функций $y = \cos x$ и $y = 2x$

Если непрерывная функция $f(x)$ на отрезке $[a; b]$ строго монотонна и имеет на концах отрезка разные знаки, то на этом отрезке существует (и причем единственный) корень уравнения $f(x) = 0$.

Действительно, функция $f(x) = 2x - \cos x$ в точках $x = 0$ и $x = 1$ имеет разные знаки и возрастает на отрезке [0; 1]:

$$f(0) = (2 \cdot 0 - \cos 0) = 0 - 1 = -1 < 0;$$

$$f(1) = (2 \cdot 1 - \cos 1) \approx 2 - (0{,}5) \approx 1{,}5 > 0.$$

Действительно, если $f(a) < 0$, $f(b) > 0$ (или наоборот), то непрерывная функция $f(x)$ обязательно хотя бы один раз пересекает ось абсцисс, а иногда несколько раз (рис. 2.6).

Отделение корней осуществляют либо графически, либо на основании аналитических исследований, либо сочетают оба этих способа.

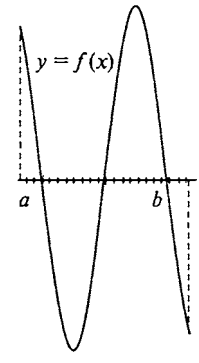



**Рис. 2.6.** График функции $y = f(x)$

**Пример 2.19.** Отделить корни уравнения $x^3 + 2x - 1 = 0$.

***Решение***

В данном случае $f(x) = x^3 + 2x - 1$, $f'(x) = 3x^2 + 2$. Поскольку $f'(x) > 0$ при всех $x$, то функция $f(x)$ возрастает в промежутке $(-\infty, +\infty)$. Корень считается отдельным, если указан конечный промежуток $(a; b)$, на котором он находится. Методом проб находим отрезок $[a; b]$, для которого $f(a)\,f(b) < 0$, т. е. на концах отрезка функция $f(x)$ принимает значения разных знаков. Для этого вычислим значения функции при некоторых значениях аргумента:

$$f(-1) = (-1)^3 + 2(-1) - 1 = -4 < 0,$$

$$f(0) = -1 < 0, \qquad f(1) = 1 + 2 - 1 = 2 > 0.$$

Так как $f(1)f(0) > 0$, то на отрезке $[-1; 0]$ корня нет; поскольку $f(-1)f(0) > 0$, то корень находится на отрезке $[0; 1]$.

*Замечание* 1. Можно указать отрезок меньшей длины, которому принадлежит корень. Взяв середину отрезка $[0; 1]$, т. е. положив $x = 0{,}5$, по формуле

$$f(0{,}5) = (0{,}5)^3 + 2 \cdot 0{,}5 - 1 > 0.$$

Так как $f(0)f(0{,}5) < 0$, то корень находится на отрезке $[0; 0{,}5]$. Этот процесс можно продолжать.

*Замечание* 2. Корень данного уравнения можно отделить и графически. Придадим уравнению вид $x^3 = -2x + 1$ и построим графики функций $y = x^3$, $y = -2x + 1$. Эти графики пересекаются в точке, абсцисса которой принадлежит интервалу $(0; 1)$.

Для уточнения корней используют несколько различных методов:

- деления отрезка пополам;
- хорд;
- касательных.

#### 2.3.2.1. Метод половинного деления

В методе половинного деления (дихотомии, бисекции) заданный отрезок $[a; b]$ разделим пополам (рис. 2.7) и положим $x_0 = (a + b)/2$. Из двух полученных отрезков $[a; x_0]$ и $[x_0; b]$ выбираем тот, на концах которого функция $f(x)$ имеет противоположные знаки. Полученный отрезок снова делим пополам и приводим те же рассуждения. Процесс продолжаем до тех пор, пока


**Рис. 2.7.** Метод половинного деления (дихотомии)

длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного ε, любую точку отрезка с точностью ε можно принять за корень уравнения $f(x) = 0$.

Таким образом, если $x_0$ и $x_1$ таковы, что $f(x_0)\,f(x_1) < 0$, то полагаем $x_2 = (x_0 + x_1)/2$ и вычисляем $f(x_2)$. Если $f(x_2) = 0$, то корень найден. В противном случае из отрезков $[x_0; x_2]$ и $[x_2; x_1]$ выбираем тот, на концах которого $f$ принимает значения разных знаков, и проделываем аналогичную операция. Процесс продолжаем до получения требуемой точности.

**Пример 2.20.** Составить программу для нахождения корней методом половинного деления для функции $F(x) = x^2 + 1{,}7x + 1{,}7$.

***Решение***

Схема алгоритма дихотомии показана на рис. 2.8.

**Программа**

```
program polovina;' {Половинное деление}
var a,b,e,x:real;
    function f(x:real):real;
    begin
      f:=x*x-1.7*x-1.7;
    end;
begin
  Write('введите a,b,e'); readln(a,b,e);
  repeat
  x:=(a+b)/2;
  if f(x)*f(b)<0 then a:=x else b:=x;
  until abs(a-b)<=e
  writeln('корень:',x:6:2); readln;
end.
```

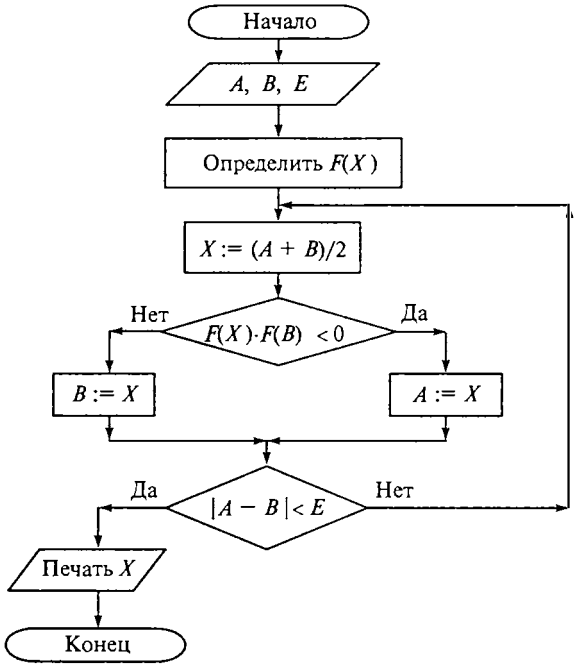



**Рис. 2.8.** Схема алгоритма дихотомии

**Пример 2.21.** Методом половинного деления найти корень уравнения $2e^{1-x} - 3{,}5\sin x = 0$ на отрезке [0,05; 1,5] с точностью $\varepsilon = 0{,}001$.

***Решение***

Схема алгоритма будет иметь вид, приведенный на рис. 2.9.

Вначале задаются значения границ отрезка $[a; b]$ и точность, с которой должен быть найден корень. Затем вычисляется значение функции в точке $a$

$$f(a) = 2e^{1-a} - 3{,}5\sin a.$$

В середине отрезка $[a; b]$ (точка $x = (a + b)/2$) вычисляется функция $f(x) = 2e^{1-x} - 3{,}5\sin x$. Проверка $f(a) * f(x) < 0$ определяет, имеют ли значения функции на границах отрезка $[a; x]$ разные знаки.

Если условие $f(a) * f(x) < 0$ верно, то корень находится на отрезке $[x; b]$.

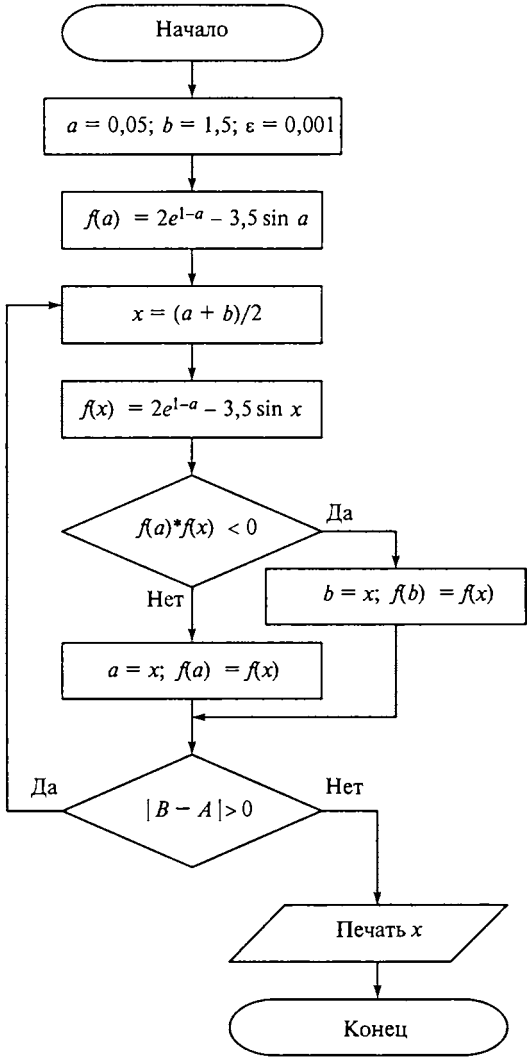



**Рис. 2.9.** Схема алгоритма дихотомии

Если условие $f(a) * f(x) < 0$ ложно, то корень находится на отрезке $[a; x]$.

При этом задаются новые значения для $a$ или $b$. Таким образом, новый отрезок $[a; b]$, на котором отыскивается корень, становится в 2 раза меньше предыдущего.

Если достигнута заданная точность, то выводят на печать найденное значение корня. В противном случае процесс деления интервала пополам продолжается.

**Пример 2.22.** Найти корень уравнения $x + \sqrt{x} + \sqrt[3]{x} - 2{,}5 = 0$ с точностью $\varepsilon = 10^{-4}$ на отрезке [0,4; 1], используя метод половинного деления.

***Решение***

Сначала выбираем начальное приближение, разделив отрезок пополам, т. е. $x_0 = (a + b)/2$. Если $F(x) = 0$, то $x_0$ является корнем уравнения. Если $F(x) \neq 0$, то выбираем тот из отрезков, на концах которого функция имеет противоположные знаки. Полученный отрезок снова делим пополам и выполняем действия сначала и т. д. Процесс деления отрезка продолжаем до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного числа ε.

**Программа**

```
program lab;
uses crt;
var x,a,b,e:real;
  iteraz:integer;
    function fun(x:real):real;
    begin
    fun:=x+sqrt(x)+exp((1/3)*(ln(x)))-2.5;
    end;
begin
  repeat
    clrscr;
    writeln('Корень уравнения находится на интервале [a,b]');
    write('Введите [a='); readln(a); write('Введите [b='); readln(b);
    write('Введите приближённое значение корня X='); readln(x);
    write('Введите точность e='); readln(e);
  until (b-a>e) or (x>a) or (x<b) or (a<>0);
  iteraz:=0; while (fun(x)<>0) and (abs(a-b)>e) do
    begin
    iteraz:=iteraz+1;
    if (fun(a)*fun(x))<0  then b:=x else a:=x;
    x:=((a+b)/2);
    end;
  writeln('Решение уравнения:');
```

```
  writeln('Вычисленное значение корня.. ',x:6:5);
  writeln('Число итераций.. ',iteraz);
  writeln('Программа закончена, нажмите Enter.');
  readln;
end.
```

**Пример 2.23.** Методом половинного деления найти корень уравнения $\sin(x) + \cos(x) = 0$ с точностью $\varepsilon$.

**Программа**

```
Program  Primer;
Uses Crt;
Var
a,b,e,x: real;
    Function f(x:real):real;
    Begin
    f:= sin(x)+cos(x);
    End;
Begin
Clrscr;
Write('Введите границы интервала: '); Readln(a,b);
if (f(a)*f(b)) > 0 then
  Begin  Sound(220); Delay(200); NoSound;
  Writeln('Ошибка! Функция не меняет знак на этом интервале!');
  Writeln('Нажмите любую клавишу'); Readkey; Halt;
  End
      else
  Begin  Write('Задайте точность вычислений: '); Readln(e);
  Writeln; Writeln('   X    F(X)    A    F(A)    B    F(B)');
  Writeln('-----------------------------------------------');
  While (Abs(b-a)>=2*e) do
    Begin
    x:=(b+a)/2;
    if f(x)*f(a)>0 then  a:=x  else  b:=x;
    Writeln(x:8:3,f(x):8:3,a:8:3,f(a):8:3,b:8:3,f(b):8:3);
    End;
End;
  Writeln; Writeln('Корень уравнения ',(B + A) / 2:4:2);
  Writeln('Нажмите любую клавишу');
Readkey;
End.
```

### 2.3.2.2. Метод итераций

Пусть задана функция $f(x)$, требуется найти корни уравнения

$$f(x) = 0. \tag{2.8}$$

Метод простых итераций (последовательных приближений) является наиболее общим, и многие другие методы можно представить как некоторую вариацию метода простых итераций.

Представим уравнение (2.8) в виде

$$x = \psi(x). \tag{2.9}$$

Это можно сделать, например, прибавив $x$ к обеим частям уравнения (2.9).

Рассмотрим последовательность чисел $x_i$, которая определяется следующим образом:

$$x_{k+1} = \psi(x_k), \quad x_0 \text{ принадлежит } [a; b].$$

Метод простых итераций имеет следующую наглядную геометрическую интерпретацию (рис. 2.10). Решением уравнения (2.9) будет абсцисса точки пересечения прямой $y = x$ с кривой $y = \psi(x)$. При выполнении итераций значение функции $\psi(x)$ в точке $x_i$ необходимо отложить по оси абсцисс. Это можно сделать, если провести горизонталь до пересечения с прямой $y = x$ и из точки их пересечения опустить перпендикуляр на ось абсцисс. На рис. 2.10 показаны разные ситуации: а) сходимость к корню односторонняя; б) сходимость с разных сторон.

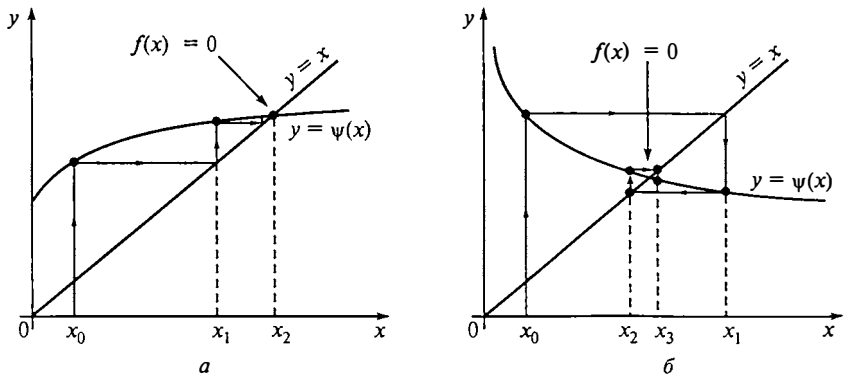



**Рис. 2.10.** Приближение к корню методом простой итерации

Сходимость процесса приближения к корню в значительной степени определяется видом зависимости $\psi(x)$. На рис. 2.11 показан расходящийся процесс, при котором метод простой итерации не находит решения уравнения.

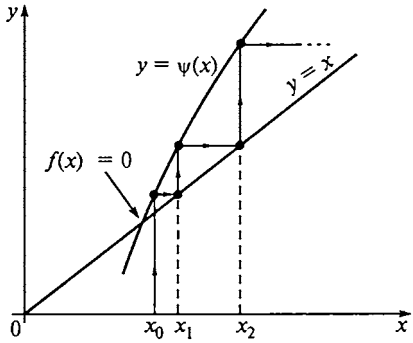



**Рис. 2.11.** Расходящийся процесс в методе простой итерации

На рис. 2.10 сходимость обеспечивается для медленно изменяющихся функций $\psi(x)$, для которых выполняется условие $|\psi'(x)| < 1$.

На рис. 2.11 расходящийся процесс наблюдается для более быстро меняющейся функции $|\psi'(x)| > 1$.

Можно сделать вывод, что для обеспечения сходимости метода простой итерации необходимо выполнить условие $|\psi'(x)| < 1$.

На практике в качестве рассматриваемой окрестности используют интервал $[a; b]$, а условие сходимости итерационного процесса имеет вид:

$$|\psi'(x)| < 1.$$

Для сходящегося итерационного процесса характерно следующее: при решении задачи переменная последовательно стремится к некоторому искомому пределу. Так как итерационный процесс представляет собой последовательность повторяющихся вычислительных процедур, то он, естественно, описывается циклическими алгоритмами. Особенность итерационного цикла заключается в том, что неизвестен закон изменения рекуррентной величины, выбранной в качестве параметра цикла, и неизвестно число повторений цикла. При этом значение, полученное на $n$-й итерации, является исходным для следующей $(n + 1)$-й итерации (рис. 2.12).

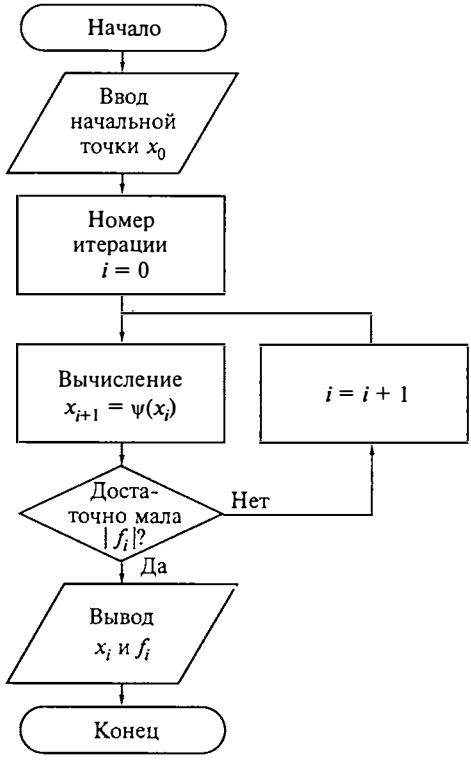



**Рис. 2.12.** Метод простой итерации

Процесс итераций продолжается до тех пор, пока для двух последовательных приближений $x_{n+1}$ и $x_n$ не будет обеспечено выполнение неравенства

$$|x_n - x_{n-1}| \le \varepsilon,$$

где $\varepsilon$ — точность вычислений.

**Пример 2.24.** Методом итераций уточнить с точностью до $10^{-4}$ корень уравнения $5x^3 - 20x + 3 = 0$, заключенный на отрезке $[0; 1]$.

***Решение***

Для отделения корней исследовалась производная уравнения $F'(x) = 15x^2 - 20$, корни которой легко определились аналитически: это $\pm 2/\sqrt{3}$. Определим знаки функции на интервалах

$$[-\infty; -2/\sqrt{3}]; \quad [-2/\sqrt{3}; 2/\sqrt{3}]; \quad [2/\sqrt{3}; +\infty].$$

| Параметр | Характеристики интервалов | | | | |
|---|---|---|---|---|---|
| Интервал | –3 | –2 | 0 | +1 | +2 |
| Знак ($F(x)$) | – | + | + | – | + |

Следовательно, корни расположены на отрезках $[-3; -2]$; $[0; 1]$ и $[1; 2]$.

Теперь уравнение $F(x) = 0$ следует привести к виду $x = \psi(x)$, что можно сделать разными способами, например:

1) $x = x + (5x^3 - 20x + 3)$, тогда $\psi_1(x) = 5x^3 - 19x + 3$;

2) $x = \sqrt[3]{\dfrac{20x - 3}{5}}$, тогда $\psi_2(x) = \sqrt[3]{\dfrac{20x - 3}{5}}$;

3) $x = \dfrac{5x^3 + 3}{20}$, тогда $\psi_3(x) = \dfrac{5x^3 + 3}{20}$.

Определим, какой из полученных функций $\psi(x)$ следует воспользоваться для вычисления последовательных приближений. Итерационный процесс сходится, если

$$|\psi'(x)| < 1.$$

Выберем на отрезке $[0; 1]$ произвольную точку $x_0$. Пусть $x_0 = 0{,}5$. Тогда

$$\psi_1'(x) = 15x^2 - 19;$$

$$\psi_2'(x) = \frac{4}{3}\left(\frac{20x - 3}{5}\right)^{-2/3};$$

$$\psi_3'(x) = \frac{3}{4}x^2.$$

Проверим условие сходимости итерационного процесса:

$\left.\begin{array}{l} |\psi_1'(x_0)| > 1 \\ |\psi_2'(x_0)| > 1 \end{array}\right\}$ — расходящийся итерационный процесс;

$|\psi_3'(x_0)| < 1$ — сходящийся итерационный процесс.

Следовательно, для вычисления последовательных приближений можно использовать только $\psi_3(x)$.

Тогда, выбирая $x_0 = 0{,}5$, определим $x_1 = \psi(x_0)$, т. е.

$$x_1 = \frac{5x_0^3 + 3}{20}.$$

Если $|x_2 - x_1| \le \varepsilon$, то $x_1$ — корень уравнения.
В противном случае вычисляем $x_2 = \psi(x_1)$, т. е.

$$x_2 = \frac{5x_1^3 + 3}{20}.$$

Затем снова проверка

$$|x_2 - x_1| \le \varepsilon.$$

Если условие выполняется, то $x_2$ — корень уравнения, в противном случае вычисляется величина $x_3 = \psi(x_2)$. Процесс продолжается до тех пор, пока не будет достигнута требуемая точность.

При составлении программы введены следующие обозначения:

X0 — предыдущее значение корня;
X1 — последующее значение корня;
Eps — точность вычислений.

**Программа**

```
Program Pr;
Var
   Eps, X0, X1,C : Real;
Begin
   Write (' Введите X0, Eps'); Readln (X0,Eps);
   Repeat
     X1:=(5*EXP(3*LN(X0))+3)/20;
     C:=ABS(X1-X0);
     X0:=X1;
   Until C <= Eps;
   Writeln(' Корень уравнения=',X0);
   Readln;
End.
```

В операторе Readln (X0,Eps) — операторе ввода начальных данных — переменным X0 и Eps задаются значения: X0: = 0,5; Eps: = 0,0001.

В операторе цикла Repeat...Until многократно вычисляется последующее $i$-е приближение (X1) через предыдущее $(i-1)$-е (X0). Оператор C: = ABS(X1-X0) производит вычисление модуля разности между последующим приближением и предыдущим. Оператор X0: = X1 пересылает значение последующего приближения (X1) в переменную X0, т. е. делается подготовка к следующей итерации.

Если |X1 – X0| > Eps, т. е. точность не достигнута, то цикл продолжается. Если корень найден с заданной точностью, осуществляется печать найденного корня.

**Пример 2.25.** Методом итераций уточнить с точностью до 0,0001 корень уравнения $2\cos(x+\pi/6)+x^2-3x+2=0$.

***Решение***

Первоначально корни уравнения определяем с точностью $\varepsilon = 0{,}1$ графическим методом, а затем найденное значение корня уточняем до 0,0001.

Перепишем уравнение в виде

$$2\cos(x+\pi/6) = -x^2+3x-2.$$

Если построить два графика: $y = 2\cos(x+\pi/6)$ и $y = -x^2 + 3x - 2$, то можно убедиться, что один корень равен ~1,1, а второй ~2,9. Поэтому первый интервал выбираем [0,9; 1,3], второй — [2,7; 3,1].

Если предположить, что все условия для реализации расчетов по методу простой итерации выполнены, то подпрограмма-функция будет иметь вид

**Функция**

```
FUNCTION ITER1 (X0: REAL; EPS : REAL;  KI : INTEGER) : REAL;
VAR
   X, Y : REAL;
     K : INTEGER;
   BEGIN
       K := 0;
       Y := X0;
       REPEAT    X := Y;    Y := FUNCI (X);    INC (K);
       UNTIL  (ABS(X-Y) < EPS)  OR  ( K > KI);
ITER1 := X;
END;
```

FUNCI (X) — подпрограмма-функция, которая вычисляет $\psi(x)$.

Если заранее неизвестно, выполняются условия или нет, то в подпрограмму-функцию следует включить дополнительную проверку:

**Функция**

```
FUNCTION  ITER2 (X0: REAL; EPS : REAL; KI : INTEGER) : REAL;
VAR       X, Y, EPS1, EPS2 : REAL;
                          K, L : INTEGER;
BEGIN
   K := 0;  Y := X0;   L := 0;  X := Y;   Y := FUNCI (X);   K := 1;
   EPS1 := ABS (X-Y);
   REPEAT
     X := Y;   Y := FUNCI (X);    INC (K);   EPS2 := ABS (X-Y);
     IF  EPS2 > EPS1  THEN  L := 1;  IF  K > KI  THEN   L := 2;
     EPS1 := EPS2;
   UNTIL  (EPS2 < EPS)  OR  ( L<>0 );
ITER2 := X;
END;
```

**Пример 2.26.** Найти методом итераций корень уравнения $x + \sqrt{x} + \sqrt[3]{x} - 2{,}5 = 0$ с точностью $\varepsilon = 10^{-4}$ на отрезке [0,4; 1].

***Решение***

Пусть уравнение $x + \sqrt{x} + \sqrt[3]{x} - 2{,}5 = 0$ имеет один корень на отрезке $[a; b]$. Функция $F(x)$ непрерывна на отрезке $[a; b]$.

Этот метод заключается в замене уравнения $x + \sqrt{x} + \sqrt[3]{x} - 2{,}5 = 0$ эквивалентным ему уравнением вида $x = 2{,}5 - \sqrt{x} - \sqrt[3]{x}$:

$$\psi(x) = 2{,}5 - \sqrt{x} - \sqrt[3]{x}.$$

После этого строится итерационный процесс: на заданном отрезке $[a; b]$ выберем точку $x_0$ (нулевое приближение) и вычислим $x_1 = \psi(x_0)$, после этого найдем $x_2 = \psi(x_1)$ и т. д.

Таким образом, процесс нахождения корня уравнения сводится к последовательному вычислению чисел:

$$x_n = \psi(x_{n-1}), \quad n = 1, 2, 3, \ldots .$$

Процесс итераций продолжается до тех пор, пока $|x_0 - x_1| \le \varepsilon$, где $\varepsilon$ — заданная абсолютная погрешность корня $x$.

**Программа**

```
program Pr;
uses crt;
  var      x0,x1,a,b,e: real;
               iteraz: integer;
  function fun(x:real): real;
    begin
    fun:=2.5-sqrt(x)-exp(1/3*ln(x));
    end;
begin
  clrscr;
  write('Введите приближённое значение x='); readln(x1);
  write('Введите точность e='); readln(e);
iteraz:=0;
  repeat
  iteraz:=iteraz+1;  x0:=x1;  x1:=fun(x0);
  until (abs(x1-x0)<=e);
    writeln('Решение уравнения:');
    writeln('Вычисленное значение корня...',x1:6:5);
    writeln('Число итераций... ',iteraz);
    writeln('Программа закончена, нажмите Enter.');
    readln;
end.
```

**Пример 2.27.** Найти методом итераций корень уравнения $\frac{\ln x}{2} = 0$ с точностью $\varepsilon = 10^{-5}$.

***Решение***

Исходное уравнение следует привести к виду $x = \psi(x)$, например

$$x = \ln(x)/2 + x.$$

**Подпрограмма**

```
procedure iter(a,b,eps,q:real; fn:func);  {Метод простой итерации}
  function Fi(x:real):real;
  begin
  Fi:=ln(x)/2+x;    {Эквивалентная функция}
  end;
```

```
var     n,i:word;
  x,xp:real;
    f:boolean;
begin
  clrscr;
  write('Введите кол-во итераций (>2) ');
    repeat
    {$I-}
    readln(n);
    {$I+}
    until (IOResult=0) and (n>1);
  x:=Fi((b-a)/2);   i:=1;   f:=true;
  while (i<n) and f do
    begin
    xp:=x;   x:=Fi(x);
    f:=abs((q*(x-xp))/(1-q))>=eps;  {f:=abs(x-xp)>=eps;}
    i:=i+1;
    end;
  if  f  then writeln('За ',n,' итераций нельзя достигнуть корня')
  else
    begin
    writeln('Корень вычислен с заданной точностью');
    writeln('Ответ: ', x:7:4);   writeln('F(x)=', fn(x):7:4);
    writeln('Кол-во итераций: ', i);
    end;
  readln;
end;
```

**Пример 2.28.** Найти методом итераций с точностью $\varepsilon = 10^{-3}$ корень уравнения, представленного в виде

$$x = F(x), \text{ где } F(x) = 5x^2 - e^{1-x} - 4.$$

***Решение***

Модернизируем метод итераций так, чтобы он быстрее сходился. Для этого функцию $F(x)$ разделим на коэффициент $M$, который является численным значением производной функции $F(x)$ в каждой точке итерационного приближения. Главной особенностью метода итерационных приближений является выбор *начального приближения*, т. е. самого первого значения $x$, с которого стартует метод итераций.

**Программа**

```
program ITERAT;
uses crt;
    const max_iter=100;  {максимальное количество итераций}
var
                i :integer;
    x,x0,eps,M :real;
function F(x:real): real;   {функция}
  begin
  F:=5*x*x-exp(1-x)-4;
  end;
begin  {основная программа}
  clrscr;
  write('Введите приближенное значение x='); readln(x);
  write('Введите точность вычислений eps='); readln(eps);
i:=0;
repeat
  {коэффициент для улучшения сходимости}
  M:=-(F(x+eps)-F(x-eps))/(2*eps);  x0:=x;
  x:=x0+F(x0)/M;   inc(i);   writeln('--- Итерация ', i:3,'  x=', x);
  writeln('F(x)=',F(x),'   точность=', abs(x-x0));
until (abs(x-x0)<=eps)or(i>max_iter);
if (abs(x-x0)<=eps) then writeln('Ответ: X=', x)
else writeln('Ответ не найден! За ', max_iter:0,' шагов ');
Readln;
end.
```

*Замечание.* Чтобы получить метод простых итераций, уберите строку:

```
{коэффициент для улучшения сходимости}
M: = -(F(x + eps) - F(x - eps))/(2*eps);
```

и вместо строки x: = x0 + F(x0)/M наберите x: = x0 + F(x0); .

### 2.3.2.3. Метод хорд

Если $x_0$, $x_1$ — приближенные значения корня уравнения $f(x) = 0$, а $f(x_0)\, f(x_1) < 0$, то последующие приближения находят по формуле

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(x_{n-1})}(x_n - x_{n-1}), \quad n = 1, 2, \ldots .$$

Методом хорд называют также метод, при котором один из концов отрезка $[a; b]$ закреплен (рис. 2.13), т. е. вычисление приближения корня уравнения $f(x) = 0$ производят по формулам

$$x_0 = b, \quad x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(a)}(x_n - a)$$

либо

$$x_0 = a, \quad x_{n+1} = x_n - \frac{f(x_n)}{f(x_n) - f(b)}(x_n - b).$$

При расчете предполагается, что корень уравнения находится на отрезке $[a; b]$, а $f''(x)$ сохраняет знак на $[a; b]$.

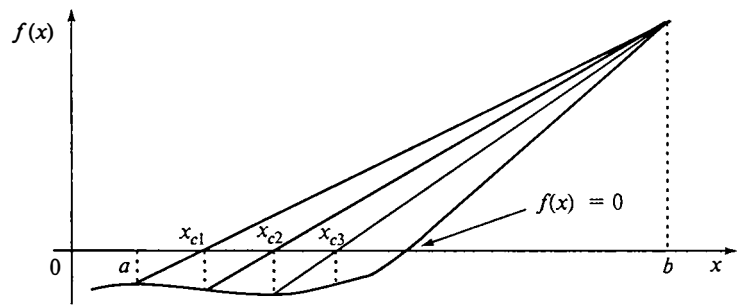



**Рис. 2.13.** Метод хорд

Из рис. 2.13 видно, что получаемые точки $x_c$ постепенно сходятся к корню уравнения. Поскольку в рассмотренном методе очередное приближение $x_c$ определяется с помощью интерполяции, учитывающей наклон кривой $f(x)$, он во многих случаях оказывается более эффективным, чем метод половинного деления.

**Пример 2.29**. Методом хорд найти корень уравнения $x^4 - 2x - 4 = 0$ с точностью до 0,01.

***Решение***

Положительный корень будет находиться на отрезке [1; 1,7], так как

$$f(1) = -5 < 0, \text{ а } f(1{,}7) = 0{,}952 > 0.$$

Найдем первое приближенное значение корня по формуле

$$x_1 = 1 - (1{,}7 - 1) \cdot f(1)/f(1{,}7) - f(1) = 1{,}588;$$

так как $f(1{,}588) = -0{,}817 < 0$, то, применяя вторично способ хорд к отрезку [1,588; 1,7], найдем второе приближенное значение корня:

$$x_2 = 1{,}588 - (1{,}7 - 1{,}588)f(1{,}588)/f(1{,}7) - f(1{,}588) = 1{,}639;$$

$$f(1{,}639) = -0{,}051 < 0.$$

Теперь найдем третье приближенное значение:

$$x_3 = 1{,}639 - (1{,}7 - 1{,}639)f(1{,}639)/f(1{,}7) - f(1{,}639) = 1{,}642;$$

$$f(1{,}642) = -0{,}016 < 0.$$

После этого найдем четвертое приближенное значение:

$$x_4 = 1{,}642 - (1{,}7 - 1{,}642)f(1{,}642)/f(1{,}7) - f(1{,}642) = 1{,}643;$$

$$f(1{,}643) = 0{,}004 > 0.$$

Следовательно, искомый корень с точностью до 0,01 равен 1,64.

**Пример 2.30.** Методом хорд найти корни уравнения $e^x - 10 = 0$ с заданной точностью ε.

***Решение***

Введем следующие обозначения:

a — начало отрезка; b — конец отрезка; eps — погрешность вычислений; x — искомое значение корня; min — модуль значения производной функции в начале отрезка; d — модуль значения производной функции в конце отрезка; x0 — точка, в которой отыскивается производная.

**Программа**

```
Program kurs;
uses crt;
  Var
    a,b,eps,x,min: real;
    {Вычисление заданной функции}
    Function fx(x:real): real;
      begin
      fx:=exp(x)-10*x;
      end;
{Функция вычисления производной и определение точности
вычислений}
```

```
{Вычисляем значение 2-й производной в точке (5*x0/4)}
Function proizv(x0,eps: real): real;
  Var     dx,dy,dy2: real;
  begin
    dx:=1;
    Repeat
      dx:=dx/2;  dy:=fx(x0+dx/2)-fx(x0-dx/2);
      dy2:=fx(5*x0/4+dx)-2*fx(5*x0/4);  dy2:=dy2+fx(5*x0/4-dx);
    Until abs(dy2/(2*dx))<eps;
    proizv:=dy/dx;
  end;
{Уточнение количества знаков после запятой}
Function utoch(eps:real): integer;
  Var     k: integer;
  begin
    k:=-1;
    Repeat
    eps:=eps*10;  k:=k+1;
    Until eps>1;
    utoch:=k;
  end;
{Процедура определения наименьшего значения производной
на заданном промежутке}
Procedure minimum(a,b,eps: real; var min: real);
  Var     d: real;
  begin
    a:=a-eps;  b:=b+eps;
    Repeat
      a:=a+eps;   b:=b-eps;   min:=abs(proizv(a,eps));
      d:=abs(proizv(b,eps));  If min>d Then min:=d
    Until  min <>0
  end;
{Процедура уточнения корня методом хорд}
Procedure chord(a,b,eps,min: real; var x:real);
Var     x1: real;
  begin
    x1:=a;
    Repeat
      x:=x1-((b-x1)*fx(x1))/(fx(b)-fx(x1));  x1:=x
    Until  abs(fx(x))/min<eps
  end;
```

```
{Основная программа}
Begin
  clrscr;
  Writeln ('Введите начало отрезка a, конец отрезка b');
  Readln (a,b);
  Writeln ('Введите погрешность измерений eps');
  Readln (eps);
  minimum(a,b,eps,min);
  chord(a,b,eps,min,x);
  Writeln ('Корень уравнения x= ',x:3:utoch(eps));
Readln;
End.
```

#### 2.3.2.4. Метод Ньютона (касательных)

Если $x_0$ — начальное приближение корня уравнения $f(x) = 0$, то последовательные приближения находят по формуле

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots .$$

Если $f'$ и $f''$ (первая и вторая производные) непрерывны и сохраняют определенные знаки на отрезке $[a; b]$, а $f(a)\,f(b) < 0$, то, исходя из начального приближения $x_0 \in [a; b]$, удовлетворяющего условию $f(x_0)f''(x_0) < 0$, можно вычислить с любой точностью единственный корень уравнения $f(x) = 0$.

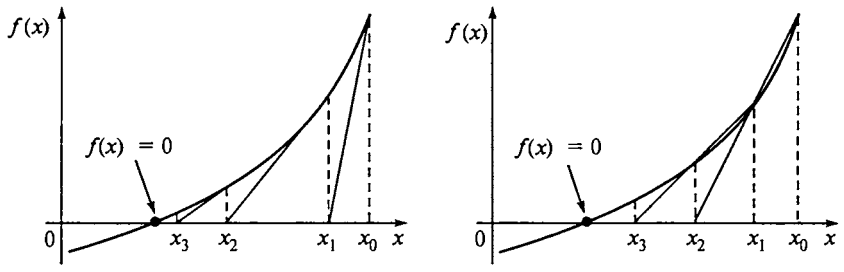
На практике часто используют модификации метода Ньютона, свободные от этого недостатка. Одно из упрощений сводится к тому, что производная вычисляется только один раз в начальной точке и затем это значение используется на всех последующих шагах. Данная модификация основывается на предположении о малом изменении производной вблизи корня.

Одной из наиболее известных модификаций является *метод секущих*. В этом методе производная заменяется ее приближенным значением:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \Rightarrow F'(x) = \frac{f(x_{i+1}) - f(x)}{x_{i+1} - x_i}.$$

В формуле для $F'(x)$ в отличие от $f'(x)$ приращение $\Delta x = x_{i+1} - x_i$ полагается малым, но $\Delta x \neq 0$. Геометрически это означает, что приближенным значением корня считается точка пересече-

ния секущей, проходящей через две точки функции $f(x_i)$ и $f(x_i + h)$, с осью абсцисс. Схема метода Ньютона показана на рис. 2.14.




*а* *б*

**Рис. 2.14.** Метод Ньютона (*а*) и метод секущих (*б*)

Выберем на отрезке $[a; b]$ произвольную точку $x_0$ — нулевое приближение. Затем найдем

$$x_1 = x_0 - \frac{F(x_0)}{F'(x_0)},$$

далее

$$x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}.$$

Таким образом, процесс нахождения корня уравнения сводится к вычислению чисел $x_n$ по формуле

$$x_n = x_{n-1} - \frac{F(x_{n-1})}{F'(x_{n-1})}, \quad n = 1, 2, 3, \ldots .$$

Процесс вычисления продолжается до тех пор, пока не будет выполнено условие

$$|x_n - x_{n-1}| < \varepsilon.$$

Схема итерационного процесса метода Ньютона приведена на рис. 2.15, из которого понятно, что каждое следующее приближение может быть определено по формуле

$$x^{(n+1)} - x^{(n)} = \frac{f(x^{(n)})}{\operatorname{tg}\alpha} = \frac{f(x^{(n)})}{f'(x^{(n)})}.$$

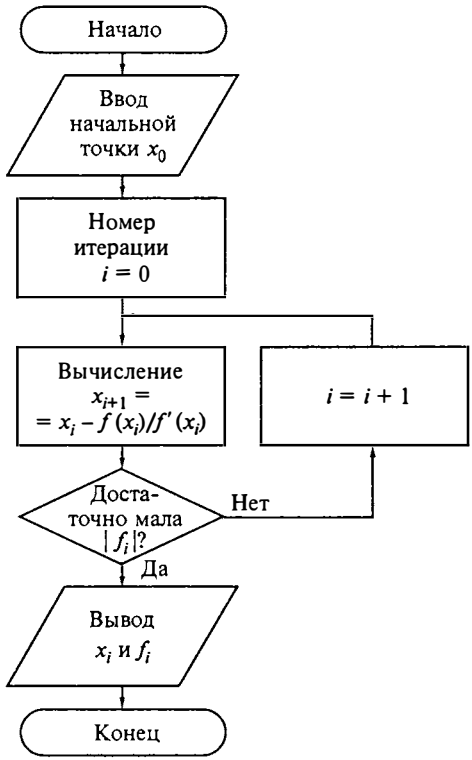



**Рис. 2.15.** Схема алгоритма Ньютона

**Пример 2.31.** Методом Ньютона (касательных) найти корень уравнения $x^4 - 2x - 4 = 0$ с точностью до 0,01.

***Решение***

В этом уравнении

$f(x) = x^4 - 2x - 4$, $f'(x) = 4x^3 - 2$, а $f''(x) = 12x^2$.

Так как $f(x)$ и $f''(x)$ при $x_0 = 1{,}7$ имеют один и тот же знак, а именно $f(1{,}7) = 0{,}952 > 0$ и $f''(1{,}7) > 0$, то применяем формулу

$$x_1 = x_0 - f(x_0)/f'(x_0),$$

где $f'(1{,}7) = 4 \cdot 1{,}7^3 - 2 = 17{,}652$.

Тогда

$$x_1 = 1{,}7 - 0{,}952/17{,}652 = 1{,}646.$$

Применяем второй раз способ касательных:

$$x_2 = x_1 - f(x_1)/f'(x_1),$$

где $f(x_1) = f(1{,}646) = 0{,}048$, $f'(1{,}646) = 15{,}838$;

$$x_2 = 1{,}646 - 0{,}048/15{,}838 = 1{,}643;$$

$$f(1{,}643) = 0{,}004, \quad f'(1{,}643) = 15{,}740;$$

$$x_3 = 1{,}643 - 0{,}004/15{,}740 = 1{,}6427.$$

Следовательно, искомый корень с точностью до 0,01 равен 1,64.

**Пример 2.32.** Методом касательных найти действительный корень уравнения $x^3 + x - 3 = 0$.

***Решение***

Записав данное уравнение в виде $x^3 = -x + 3$ и построив графики функций $f_1(x) = x^3$ и $f_2(x) = -x + 3$, найдем, что единственный корень уравнения принадлежит отрезку [1; 2].

Определим отрезок меньшей длины, на котором находится корень.

Так как $f(x) = x^3 + x - 3$, $f(1{,}2) = (1{,}2)^3 + 1{,}2 - 3 = -0{,}072 < 0$, $f(1{,}3) = (1{,}3)^3 + 1{,}3 - 3 = 0{,}497 > 0$, то корень лежит на отрезке [1,2; 1,3]. Серединой этого отрезка является точка $x = 1{,}25$. Поскольку $f(1{,}25) = (1{,}25)^3 + 1{,}25 - 3 = 0{,}203125 > 0$ и $f(1{,}2) < 0$, то искомый корень принадлежит отрезку [1,20; 1,25]. Данная функция $f(x) = x^3 + x - 3$ имеет производные $f'(x) = 3x^2 + 1$, $f''(x) = 6x$, принимающие положительные значения на отрезке [1,20; 1,25]. В качестве начального приближения выберем $x = 1{,}25$.

Результаты вычислений записываем в табл. 2.2, из которой видно, что искомый корень $x = 1{,}21341$.

*Таблица 2.2.* **Метод касательных**

| $n$ | $x_n$ | $f(x_n) = x_n^3 + x_n - 3$ | $f'(x_n) = 3x_{n+1}^2$ | $x_{n+1}$ |
|---|---|---|---|---|
| 0 | 1,25 | 0, 203125 | 5,6875 | 1,214286 |
| 1 | 1,214286 | 0,004738 | 5,42347 | 1,213412 |
| 2 | 1,213412 | 0,000002 | 5,417107 | 1,213412 |

**Пример 2.33**. Методом Ньютона найти корни уравнения $y = x^3 + 0,1x^2 + 0,4x - 1,2$ на отрезке [0; 1] с точностью ε.

***Решение***

Приведем уравнение к виду $x = \psi(x)$ так, чтобы $|\psi'(x)| < 1$ при $0 \le x \le +1$. Так как $\max|\psi'(x)| = \psi'(+1) = 3 + 0,1 + 0,4 = 3,5$, то можно выбрать $R = 2$. Вычислим $\psi(x) = x - (f(x)/R) = x - 0,5x^3 - 0,05x^2 - 0,2x + 0,6 = -0,5x^3 - 0,05x^2 + 0,8x + 0,6$.

**Программа**

```
program metod_kasatel;
  Uses Crt;
  Var   xn,xn1,a,b,c,mx,y0,x0 :real;
    function f1(x1:Real): Real; {Заданная функция}
      begin
      f1 := x1*x1*x1*(-0.5)-0.05*x1*x1+0.8*x1+0.6;
      end;
    function f2(x4:Real): Real;
      begin
      f2 := x4*x4*x4+0.5*x4*x4+0.1*x4*x4+0.4*x4-1.2;
      end;
begin {Основная программа}
  Clrscr;
  a:=0;  b:=1;  c:=0.00000001;
  Writeln(' От A=',a,' до B=',b);  Writeln(' Погрешность c=',c);
  Readln; { Ожидание нажатия клавиши Enter}
    xn:=b;    xn1:= f1(xn);    y0:=f2(b);
  while ABS(y0)>c do {Проверка по точности вычисления корня}
    begin
    xn:=xn1;   xn1:=f1(xn);    y0:= f2(xn1);
    {Печать промежуточного результата}
    Writeln('xn=',xn,' xn+1=',xn1,' f(xn+1)=',y0);
    Readln;
    end; {Конец тела цикла}
  Writeln('Конечные значения'); Writeln(' xn+1=',xn1,' f(xn+1)=',y0);
  Readln;
end.
```

**Пример 2.34**. Комбинированным методом хорд и касательных найти корни уравнения, заданного функцией FUNC(X) на отрезке $[a; b]$ с точностью ε.

***Решение***

Формальные параметры процедуры:

*входные:* a, b — заданный отрезок, на котором ищется решение; eps — точность решения (погрешность); it — наибольшее разрешенное число итераций.

*выходные:* x — решение, найденное с заданной точностью; k — целое число, равное 0, если процесс решения прошел удачно, и 1, если решение расходится.

**Подпрограмма**

```
Procedure HORDKAS (A,B,EPS:real; IT:integer; VAR X : real;
VAR K:integer);
Var X1,X2,X3,A1,B1 : real;
          K1 : integer;
Begin
   K := 1;  X1 := FUNC(A);  X2 := FUNC(B);
   A1 :=A;  B1 := B;
   X3 := B - X2*(B-A)/(X2-X1);
   IF SIGN(X2)=SIGN(FUNC(X3)) then K1:=1 else   K1:=2;
   repeat
     inc (K);  X := X3;
     Case K1 of
     1: X2:= X3-FUNC(X3)*(X3-A1)/ (FUNC(X3)-FUNC(A1));
     2: X2:= X3-FUNC(X3)*(B1-X3)/ (FUNC(B1)-FUNC(X3));
     end;
     X3 := X2;
     Case K1 of
     1:  begin
       A1 := A1 - FUNC(A1)/FUNC1(A1);
       X1 := A1;
       end;
     2: begin
       B1 := B1 - FUNC(B1)/FUNC1(B1);
       X1 := B1;
       end;
     end;
until (K>IT) or (ABS(X-X2)<EPS);
End;
```

## 2.4. Ряды

Ряды играют исключительную роль в математике как очень эффективное средство математического исследования и моделирования. Известные всем таблицы тригонометрических функций, таблицы логарифмов и т. п. составляются с помощью рядов для этих функций. Точное значение числа $\pi$ также получается с помощью ряда.

Понятие суммы конечного числа чисел и свойства суммы были известны уже в древнейшие времена. С частными примерами сумм бесконечных рядов, например, с суммой членов убывающей геометрической прогрессии, математики имели дело уже во времена Архимеда. Успешно пользовались рядами Ньютон, Лейбниц, Эйлер, Гаусс. Однако точная теория рядов, основанная на понятии предела последовательности и содержащая доказательства основных теорем, была построена в первой половине XIX в. в основном Коши. С тех пор ряды стали незаменимым рабочим средством для математики, появились разделы математики, например, теория аналитических функций, целиком основанные на теории рядов.

Сумма членов бесконечной числовой последовательности $u_1$, $u_2$, ..., $u_n$, ... называется **числовым рядом**.

$$u_1 + u_2 + \ldots + u_n + \ldots = \sum_{n=1}^{\infty} u_n,$$

при этом числа $u_1$, $u_2$, ... будем называть членами ряда, а $u_n$ — общим членом ряда.

Суммы $S_n = u_1 + u_2 + \ldots + u_n = \sum_{k=1}^{n} u_k$, $n = 1, 2, \ldots$ называются **частичными** (частными) **суммами** ряда

$$S_1 = u_1,$$

$$S_2 = u_1 + u_2,$$

$$S_3 = u_1 + u_2 + u_3,$$

$$S_n = u_1 + u_2 + u_3 + \ldots + u_n.$$

Таким образом, возможно рассматривать последовательности частичных сумм ряда $S_1$, $S_2$, ..., $S_n$, ... . При этом разность между

суммой $S$ и частичной суммой $S_n$ называется $n$-м остатком ряда $R_n = S - S_n$.

Так как $S$ есть предел последовательности $S_n$, то очевидно:

$$\lim_{n\to\infty} R_n = \lim_{n\to\infty}(S - S_n) = 0.$$

Поэтому, взяв достаточно большое число членов сходящегося ряда, сумму этого ряда можно вычислить с необходимой степенью точности.

Для сходящегося ряда его $n$-й член $u_n$ при неограниченном возрастании номера $n$ стремится к нулю, т. е. $u_n = S_n - S_{n-1}$; $\lim\limits_{n\to\infty} u_n = 0$.

Ряд $u_1 + u_2 + \ldots + u_n + \ldots = \sum\limits_{n=1}^{\infty} u_n$ называется **сходящимся**, если сходится последовательность его частных сумм. Сумма сходящегося ряда — предел последовательности его частных сумм

$$\lim S_n = S, \quad S = \sum_{n=1}^{\infty} u_n.$$

Если последовательность частных сумм ряда расходится, т. е. не имеет предела или имеет бесконечный предел, то ряд называется **расходящимся** и ему не ставят в соответствие никакой суммы.

Если ряд $\sum u_i$ сходится и его сумма равна $S$, то ряд $\sum Cu_i$ тоже сходится, и его сумма равна $CS$ $(C \neq 0)$.

**Суммой** или **разностью** этих рядов будет называться ряд $\sum(u_n \pm v_n)$, где элементы получены в результате сложения (вычитания) исходных элементов с одинаковыми номерами.

Если ряды $\sum u_n$ и $\sum v_n$ сходятся и их суммы равны соответственно $S$ и $\sigma$, то ряд $\sum(u_n \pm v_n)$ тоже сходится и его сумма равна $S + \sigma$:

$$\sum(u_n + v_n) = \sum u_n + \sum v_n = S + \sigma.$$

Разность двух сходящихся рядов также будет сходящимся рядом.

**Пример 2.35.** Исследовать ряд $\sum\limits_{n=1}^{\infty}\left(\dfrac{2n-1}{n+2}\right)^n$ на сходимость.

***Решение***

Находим

$$\lim_{n\to\infty}\sqrt[n]{U_n}=\lim_{n\to\infty}\sqrt[n]{\left(\frac{2n-1}{n+2}\right)^n}=\lim_{n\to\infty}\frac{2n-1}{n+2}=\left[\frac{\infty}{\infty}\right]=\lim_{n\to\infty}\frac{2n/n-1/n}{n/n+2/n}=$$

$$=\lim_{n\to\infty}\frac{2-1/n}{1+2/n}=2.$$

Так как $2>1$, то по признаку Коши ряд расходится.

*Ответ*: ряд расходится.

**Пример 2.36.** Исследовать ряд $\sum\limits_{n=1}^{\infty}\frac{(2n)!}{(n!)^2}$ на сходимость.

***Решение***

Так как в $U_n$ присутствуют факториалы, используем признак Даламбера.

Составим $U_{n+1}=\frac{(2(n+1))!}{((n+1)!)^2}$, тогда

$$\lim_{n\to\infty}\frac{U_{n+1}}{U_n}=\lim_{n\to\infty}\frac{(2n+2)!(n!)^2}{((n+1)!)^2(2n)!}=\left|\begin{matrix}(2n+2!=(2n)!(2n+1)(2n+2)\\(n+1)!=n!(n+1)\end{matrix}\right|=$$

$$=\lim_{n\to\infty}\frac{(2n)!(2n+1)(2n+2)(n!)^2}{(n!)^2(n+1)^2(2n)!}=\left[\frac{\infty}{\infty}\right]=$$

$$=\lim_{n\to\infty}\frac{\left(\frac{2n}{n}+\frac{1}{n}\right)\left(\frac{2n}{n}+\frac{2}{n}\right)}{\left(\frac{n+1}{2}\right)^2}=\lim_{n\to\infty}\frac{\left(2+\frac{1}{n}\right)\left(2+\frac{2}{n}\right)}{\left(1+\frac{1}{n}\right)^2}=4>1.$$

Следовательно, по признаку Даламбера ряд расходится.

*Ответ*: ряд расходится.

**Пример 2.37.** Исследовать ряд $\sum\limits_{n=1}^{\infty}\left(\frac{2n+1}{2n+3}\right)^n$ на сходимость.

***Решение***

Признак Коши в этом случае не работает, так как $\lim\limits_{n\to\infty}\sqrt[n]{U_n} = 1$. Здесь применим необходимый признак сходимости рядов. Находим

$$\lim_{n\to\infty} U_n = \lim_{n\to\infty}\left(\frac{2n+1}{2n+3}\right)^n = [1+0]^{\infty}.$$

Сводим ко второму замечательному пределу

$$\lim_{n\to\infty} U_n = \lim_{n\to\infty}\left(\frac{(2n+1)+2-2}{2n+3}\right)^n = \lim_{n\to\infty}\left(1+\frac{-2}{2n+3}\right) = e^{-1} \neq 0.$$

Следовательно, по необходимому признаку ряд расходится.

*Ответ:* ряд расходится.

**Пример 2.38.** Методом математической индукции доказать формулу

$$1^3 + 2^3 + 3^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^3,$$

где $n$ — натуральное число.

***Решение***

При $n = 1$ обе части равенства обращаются в единицу и, следовательно, первое условие принципа математической индукции выполнено.

Предположим, что формула верна при $n = k$, т. е.

$$1^3 + 2^3 + 3^3 + \ldots + k^3 = \left(\frac{k(k+1)}{2}\right)^2.$$

Прибавим к обеим частям этого равенства $(k+1)^3$ и преобразуем правую часть. Тогда получим

$$1^3 + 2^3 + \ldots + k^3 + (k+1)^3 = \left(\frac{k(k+1)}{2}\right)^2 + (k+1)^3 =$$

$$= (k+1)^2\left(\frac{k^2}{4} + k + 1\right) = \left(\frac{k+1}{2}\right)^2 (k^2 + 4k + 4) = \left(\frac{(k+1)(k+2)}{2}\right)^2.$$

Таким образом, из условия, что формула верна при $n = k$, следует, что она верна и при $n = k+1$. Это утверждение справедливо при любом натуральном значении $k$. Итак, второе условие

принципа математической индукции тоже выполнено. Формула доказана.

**Пример 2. 39.** Доказать, что сумма $n$ первых чисел натурального ряда равна $\frac{n(n+1)}{2}$.

***Решение***

Обозначим искомую сумму $S_n$, т. е. $S_n = 1 + 2 + \ldots + n$.

При $n = 1$ гипотеза верна.

Пусть $S_k = 1 + 2 + \ldots + k = \frac{k(k+1)}{2}$.

Покажем, что $S_{k+1} = \frac{(k+1)(k+2)}{2}$.

В самом деле,

$$S_{k+1} = S_k + (k+1) = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}.$$

Задача решена.

**Пример 2.40.** Вычислить сумму ряда

$$y = \frac{x}{2} - \frac{x^2}{2^2} + \frac{x^3}{2^3} - \frac{x^4}{2^4} + \ldots$$

с точностью $\varepsilon = 0{,}001$ при заданном значении $x = 0{,}235$.

***Решение***

Вычисление суммы ряда с заданной точностью $\varepsilon$ следует производить до тех пор, пока не выполнится условие

$$|u_n| \le \varepsilon.$$

В данном примере вычисление суммы ряда

$$y = \frac{x}{2} - \frac{x^2}{2^2} + \frac{x^3}{2^3} - \frac{x^4}{2^4} + \ldots$$

необходимо производить до тех пор, пока очередной член ряда по абсолютной величине не будет меньше $\varepsilon = 0{,}001$. Обозначим $n$-й член ряда через

$$U = (-1)^{n+1} \frac{x^n}{2^n}.$$

Тогда алгоритм нахождения суммы ряда будет иметь вид:

$$y = y + U.$$

**Программа**

```
Program PR;
Uses Crt;
Const  Eps=0.001;
Var
    U, X,Y : Real;
    N, R: Integer;
Begin
    Clrscr;
    X:=0.235;  Y:=0;  N:=0;  R:=1;
    REPEAT
    N:=N+1;
    U:=R*EXP(N*LN(X/2));
    Y:=Y+U;
    R:= -R;
    UNTIL  ABS(U) <=Eps;
    WRITELN('Y=', Y:8:3);
Readkey;
End.
```

В данной программе очередной член ряда вычисляется независимо от предыдущего. При таком подходе для вычисления $U$ необходимо использовать операции умножения и возведения в степень. Для повышения эффективности программы рассмотрим второй вариант.

Определим последующий член ряда через предыдущий:

$$U_{n+1} = -U_n \frac{x}{2}, \text{ при этом } U_0 = -1, U_1 = \frac{x}{2}.$$

Обозначим:

Y — значение суммы ряда;

U — $n$-й член ряда;

R — «мигающая» единица;

N — номер вычисляемого члена ряда.

Тогда схема алгоритма примет следующий вид (рис. 2.16).

**Программа**

```
Program PR;
Uses Crt;
Const  Eps=0.001;
```

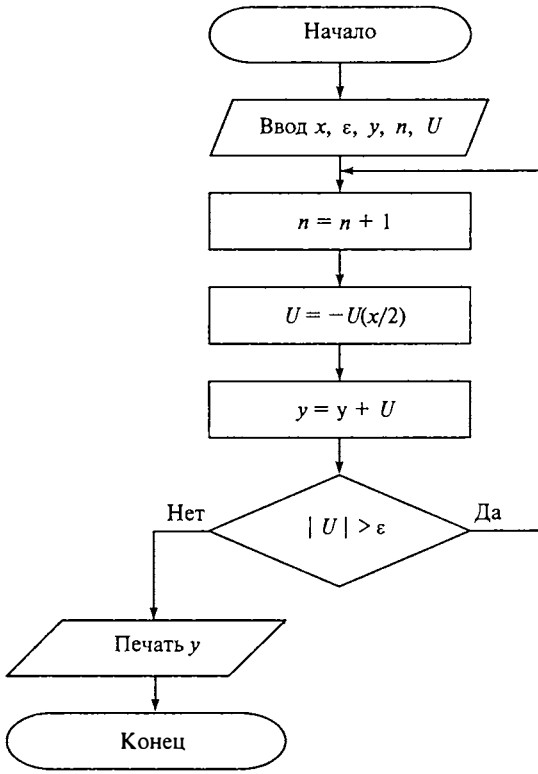



**Рис. 2.16.** Схема алгоритма вычисления суммы ряда

```
Var
    U, X,Y : Real;
    N: Integer;
Begin
    ClrScr;
    X:=0.235;  Y:=0;  N:=0; U:= -1;
    REPEAT
    N:=N+1;
    U:=  -U*X/2;
    Y:=Y+U;
    UNTIL  ABS(U) <=Eps;
    WRITELN('Y=', Y:9:4);
Readkey;
End.
```

**Пример 2.41.** Вычислить сумму ряда с заданной точностью $\varepsilon = 0{,}0001$:

$$y = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{4!} + \ldots \quad \text{при } x = 1{,}25.$$

***Решение***

Общий член данного ряда выражается формулой

$$U_n = \frac{x^n}{n!}.$$

**Программа**

```
Program PR;
Const Eps=0.0001;
Var
   U, X,Y : Real;
   N: Integer;
   IFAKT:LongInt;
Begin
   X:=0.235; Y:=1; N:=0; IFAKT:=1;
   REPEAT
   N:=N+1;
   IFAKT:=IFAKT*N;
   U:=EXP(N*LN(X))/ IFAKT;
   Y:=Y+U;
   UNTIL U <=Eps;
   WRITELN('Y=', Y:8:4);
Readln;
End.
```

**Пример 2.42.** Вычислить сумму ряда с точностью $\varepsilon = 10^{-4}$, общий член которого $a_n = \frac{n!}{n^n}$.

***Решение***

При определении суммы членов ряда следует использовать заданную рекуррентную формулу.

При составлении программы считать, что точность достигнута, если $a_n \le 10^{-4}$.

**Программа**

```
program Pr;
Uses CRT;
Var    sum,an:real;
          n:integer;
          nfakt: longint;
begin
clrscr;
sum:=0;  n:=1; nfakt:=1; an=1;
    while an>0.0001 do
    begin
      nfakt:=nfakt*n;
      an:=nfakt/(exp(n*ln(n));
      sum:=sum+an;
      n:=n+1;
    end;
writeln('Сумма ',n,' элементов равна =',sum:8:6);
readkey;
end.
```

**Пример 2.43.** Вычислить функцию Бесселя

$$J_0(x) = \sum_{k=0}^{\infty} \frac{(-1)^k \left(\frac{x}{2}\right)^{2k}}{(k!)^2}$$

с точностью $\varepsilon = 10^{-3}$.

***Решение***

Суммирование будет производится до тех пор, пока элемент ряда не станет меньше значения погрешности, т. е. $|a_n| < \varepsilon$. Напишем простейшую программу, подсчитывающую значение функции Бесселя.

**Функция 1**

```
function  Func(x, eps : Real):Real;
{ x - аргумент функции, eps  — погрешность округления.
Функция возвращает приближенное значение функции Бесселя }
var
    k: word;
    a, s: Real;
begin
  k:= 0;   a:=1.0;   s:= 1.0;
```

```
    while abs(a) > eps do
      begin
      k:= k + 1;
      a:= -a * sqr (x / 2.0) / sqr (k);
      s:= s + a;
      end;
  Func := s;
end;
```

При этом $k$ лучше было бы хранить как вещественное число.

Для увеличения скорости работы программы удобно затабулировать квадрат знаменателя. Запишем разность знаменателей на следующих один за другим шагах цикла. $Z_{k+1} = (k+1)^2 - k^2 = 2k + 1$, в свою очередь, табулируя полученную линейную функцию, получаем $U_{k+1} = Z_{k+1} - Z_k = 2$. Таким образом нахождение $k^2$ сводится к сложению, представленному следующей таблицей:

| $k$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $k^2$ | 0 | 1 | 4 | 9 | 16 |
| $z$ | 1 | 1 | 3 | 5 | 7 |
| $u$ | 2 | 2 | 2 | 2 | 2 |

Теперь можно написать другую подпрограмму.

**Функция 2**

```
function J0 (x, eps :Real) : Real;
{ x — значение аргумента, eps — погрешность округления.
Функция возвращает приближенное значение функции Бесселя }
  const
  u : = 2.0;
var
  s, a, z, z0, r : Real;
begin
  a:= 1.0;  s:= a;  z:= -1.0;  z0:= 0.0;
  r:= -sqr (x/2.0);
  while abs (a) > eps do
    begin
    z:= z + u;
    z0 := z0 + z;
```

```
      a:= a * r / z0;
      s:= s+a;
      end;
   J0:= s;
end;
```

Математические свойства данной программы будут такими же, так как используется тот же математический аппарат.

## 2.5. Системы уравнений

Многие задачи практики сводятся к необходимости решения системы линейных уравнений. При конструировании инженерных сооружений, обработке результатов измерений, решении задач планирования производственного процесса и ряда других задач техники, экономики, научного эксперимента приходится решать системы линейных уравнений.

Система $m$ линейных уравнений с $n$ неизвестными (или линейная система) в линейной алгебре — это система уравнений вида

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \end{cases}$$

Здесь $x_1$, $x_2$, ..., $x_n$ — неизвестные, которые надо определить; $a_{11}$, $a_{12}$, ..., $a_{mn}$ — коэффициенты системы; $b_1$, $b_2$, ... $b_m$ — свободные члены (предполагаются известными). Индексы коэффициентов ($a_{ij}$) системы обозначают номера уравнения ($i$) и неизвестного ($j$), при котором стоит этот коэффициент.

Система называется *однородной*, если все ее свободные члены равны нулю, т. е. $b_1 = b_2 = \ldots = b_m = 0$, иначе — *неоднородной*.

Система называется *квадратной*, если число $m$ уравнений равно числу $n$ неизвестных.

*Решение* системы — совокупность $n$ чисел $c_1$, $c_2$, ..., $c_n$, таких, что подстановка каждого $c_i$ вместо $x_i$ в систему обращает все ее уравнения в тождества.

Система называется *совместной*, если она имеет хотя бы одно решение, и *несовместной*, если у нее нет ни одного реше-

ния. Совместная система может иметь одно или более решений. Решения $c_1^{(1)}, c_2^{(1)}, \ldots, c_n^{(1)}$ и $c_1^{(2)}, c_2^{(2)}, \ldots, c_n^{(2)}$ совместной системы называются *различными*, если нарушается хотя бы одно из равенств

$$c_1^{(1)} = c_1^{(2)}, \quad c_2^{(1)} = c_2^{(2)}, \quad \ldots, \quad c_n^{(1)} = c_n^{(2)}.$$

Совместная система называется *определенной*, если она имеет единственное решение; если же у нее есть хотя бы два различных решения, то она называется *неопределенной*.

### 2.5.1. Матричный метод

**Матричный метод** решения систем линейных алгебраических уравнений с ненулевым определителем состоит в следующем.

Пусть дана система $n$ линейных уравнений с $n$ неизвестными:

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots; \\ a_{n1}x_1 + \ldots + a_{nn}x_n = b_n. \end{cases}$$

Тогда ее можно переписать в матричной форме:

$$\mathbf{AX} = \mathbf{B},$$

где $\mathbf{A}$ — основная матрица системы, $\mathbf{B}$ и $\mathbf{X}$ — столбцы свободных членов и решений системы соответственно:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Умножим это матричное уравнение слева на $\mathbf{A}^{-1}$ — матрицу, обратную матрице $\mathbf{A}$:

$$\mathbf{A}^{-1}(\mathbf{AX}) = \mathbf{A}^{-1}\mathbf{B}.$$

Так как $\mathbf{A}^{-1}\mathbf{A} = \mathbf{E}$ (учитывая ассоциативность матричного произведения), получаем $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$. Правая часть этого уравнения даст столбец решений исходной системы. Условием применимости данного метода (как и вообще существования решения неод-

нородной системы линейных уравнений с числом уравнений, равных числу неизвестных) является невырожденность матрицы **A**. Необходимым и достаточным условием этого является неравенство нулю определителя матрицы **A**:

$$\det \mathbf{A} \neq 0.$$

Для однородной системы линейных уравнений, т. е. когда вектор $\mathbf{B} = 0$, действительно обратное правило: система $\mathbf{AX} = 0$ имеет нетривиальное (т. е. ненулевое) решение только если $\det \mathbf{A} = 0$. Такая связь между решениями однородных и неоднородных систем линейных уравнений носит название альтернативы Фредгольма.

**Трехдиагональной** называют матрицу вида

$$\mathbf{A} = \begin{pmatrix} C_1 & B_1 & 0 & 0 & & 0 & 0 \\ A_2 & C_2 & B_2 & 0 & & 0 & 0 \\ 0 & A_3 & C_3 & B_3 & & 0 & 0 \\ \dots & \dots & \dots & \dots & & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & A_{n-1} & C_{n-1} \end{pmatrix}.$$

Системы с такими матрицами встречаются при решении многих задач математики и физики. Помимо матрицы в правой части в системе задаются краевые условия $x_0$ и $x_n$, которые берутся из контекста задачи.

**Пример 2.44.** Решить систему линейных уравнений, заданную таблично, методом простой итерации.

| $Y_1$ | $Y_2$ | $Y_3$ | $F$ |
|---|---|---|---|
| 10 | 4 | 0 | 13 |
| 2 | 10 | 5 | 14 |
| 0 | 2 | 10 | 15 |

***Решение***

Приводим систему к виду:

$$\begin{cases} X_1 = (F_1 - 4X_2)/10; \\ X_2 = (14 - 2X_1 - 5X_3)/10; \\ X_3 = (15 - 2X_2)/10. \end{cases}$$

Построим таблицу итераций (последовательных приближений). Во второй столбец вводим начальные значения аргументов, в третий — соответствующие формулы, исходя из системы:

$(X_1 = 1{,}3 - 0{,}4*B_4;\ \ X_2 = 1{,}4 - 0{,}2*B_3 - 0{,}5*B_5;\ \ X_3 = 1{,}5 - 0{,}2*B_4)$

| Номер итерации | 1 | 2 |
|---|---|---|
| $X_1$ | 1,3 | 0,74 |
| $X_2$ | 1,4 | 0,39 |
| $X_3$ | 1,5 | 1,22 |
| | Расхожд. $X_1$ | 0,56 |
| | Расхожд. $X_2$ | 1,01 |
| | Расхожд. $X_3$ | 0,28 |

Автозаполняем таблицу до тех пор, пока расхождение значений $X$ соседних итераций не снизится до приемлемой величины. Таким образом на десятой итерации получаем

| Номер итерации | 10 |
|---|---|
| $X_1$ | **1,109368** |
| $X_2$ | **0,47552** |
| $X_3$ | **1,404684** |
| Расхожд. $X_1$ | 0,000588 |
| Расхожд. $X_2$ | 0,00106 |
| Расхожд. $X_3$ | 0,000294 |

**Пример 2.45.** Методом итераций решить систему 4×4 линейных уравнений с точностью $\varepsilon = 0{,}0001$:

$$\begin{cases} X_1 = 0{,}08X_1 + 0{,}05X_2 + 0{,}11X_3 + 0{,}08X_4 + 2{,}15; \\ X_2 = 0{,}05X_1 + 0{,}13X_2 + 0{,}27X_3 + 0{,}28X_4 + 0{,}44; \\ X_3 = 0{,}11X_1 + 0{,}27X_2 + 0{,}28X_3 + 0{,}06X_4 + 0{,}83; \\ X_1 = 0{,}08X_1 + 0{,}18X_2 + 0{,}06X_3 + 0{,}12X_4 + 1{,}16. \end{cases}$$

***Решение***

Формальные параметры процедуры:

*входные:* A — матрица, составленная из коэффициентов при X преобразованного уравнения; B — матрица, составленная из

свободных членов; N — размерность массивов **A**($N \times N$) и **B**($N$); IK — предельно возможное количество итераций (введено для того, чтобы в случае расхождения процесса выйти из подпрограммы. Обычно решение достигается за 3—6 итераций); EPS — заданная погрешность решения;

*выходные:* X — массив, в котором находится решение системы.

**Подпрограмма**

```
Procedure  ITER (N, IK :integer; EPS : real;
                          A : mas1; B : mas; VAR X : mas);
VAR      X1 : mas;
          S : real;
          I, J, K : integer;
BEGIN
   X1 := B;    X := X1;  K := 0;
 REPEAT  S := 0.0;    Inc(K);
   for I := 1 to N do
   BEGIN
     for J := 1 to N do X[I] := A[I,J]*X1[J] + B[J];
     S := S + abs (X[I]-X1[I]);
   END;
   S := S / N;      X1 := X;
 UNTIL (S<EPS) AND (K>IK);
END;
```

Результаты вычислений представлены в таблице:

| $X[1]$ | $X[2]$ | $X[3]$ | $X[4]$ | Номер итерации |
|---|---|---|---|---|
| 2,150000 | 0,440000 | 0,830000 | 1,160000 | *ITER* = 0 |
| 1,252800 | 1,484800 | 1,229600 | 1,299200 | *ITER* = 1 |
| 1,263936 | 1,523776 | 1,237952 | 1,315904 | *ITER* = 2 |
| 1,265272 | 1,528453 | 1,238954 | 1,317908 | *ITER* = 3 |
| 1,265433 | 1,529014 | 1,239075 | 1,318149 | *ITER* = 4 |
| 1,265452 | 1,529082 | 1,239089 | 1,318178 | *ITER* = 5 |
| 1,265454 | 1,529090 | 1,239091 | 1,318181 | *ITER* = 6 |
| 1,265455 | 1,529091 | 1,239091 | 1,318182 | *ITER* = 7 |
| 1,265455 | 1,529091 | 1,239091 | 1,318182: | *РЕШЕНИЕ* |

**Пример 2.46.** Методом Ньютона решить систему двух уравнений

$$\begin{cases} e^{(x+y)} - x^2 + y = 2; \\ (x + 0{,}5)^2 + y^2 = 1. \end{cases}$$

***Решение***

Начальное приближение решения системы (рис. 2.17) найдено графически (0,4; 0,2).

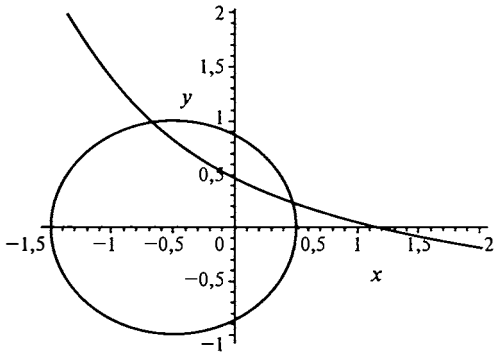



**Рис. 2.17.** Графическое решение системы

## Программа

```
function funcF(x,y:real):real;
begin
funcF:=exp(x+y)-x*x+y-2
end;
  function funcG(x,y:real):real;
  begin
  funcG:=(x+0.5)*(x+0.5)+y*y-1
  end;
{Процедура метода Ньютона (x0, y0 — начальное приближение,
eps — точность, max — максимальное число итераций, x, y —
точное решение, IP — код ошибки)}
{hx, hy — шаг по оси X (по Y),    Fx,Fy,Gx,Gy — производные
функций}
procedure NN(x0, y0, eps: real; max :integer; var x, y :real;
var IP:integer);
var max1 : integer;
  hx,hy, Fx0,Gx0, Fx,Fy,Gx,Gy, x1,y1 : real;
```

```
begin
max1:=0;  x1:=x0+0.00001;  y1:=y0+0.00001;
hx:=0.00001;   hy:=0.00001;
Fx0:=funcF(x0,y0);  Gx0:=funcG(x0,y0);
while (abs(x0-x1)>=eps) and (max1<=max) do
begin
  max1:=max1+1;   Fx0:=funcF(x0,y0);   Gx0:=funcG(x0,y0);
  Fx:=(funcF(x1,y0)-funcF(x0-hx,y0))/(2*hx); {производная функции
                                              F по X}
  Gx:=(funcG(x1,y0)-funcG(x0-hx,y0))/(2*hx); {производная
                                              функций G по X}
  Fy:=(funcF(x0,y1)-funcF(x0-hy,y0))/(2*hy); {производная функций
                                              F по Y}
  Gy:=(funcG(x0,y1)-funcG(x0-hy,y0))/(2*hy); { производная
                                              функций F по Y}
  x1:=x0-(Fx0*Gy-Fy*Gx0)/(Fx*Gy-Fy*Gx);
  y1:=y0-(Fx*Gx0-Fx0*Gx)/(Fx*Gy-Fy*Gx);
  x:=x1; y:=y1;
  writeln(' x=',x,' y=',y);  {промежуточные значения}
  hx:=x1-x0;    hy:=y1-y0;    x0:=x1;
  x1:=x0+hx;    y0:=y1;    y1:=y0+hy;
  end;
if max1>max then IP:=1 else IP:=0;
end;
var x1,y1 : real;
    IP    : integer;
begin
NN(0.2,0.4,0.0000001,1000,x1,y1,IP); {вызов процедуры}
writeln('x1 = ',x1,'; y1 = ',y1,'; IP = ',IP);
Readln;
end.
```

### 2.5.1.1. Метод Зейделя

Этот метод представляет собой некоторую модификацию метода простой итерации. Основная его идея заключается в том, что при вычислении $(k+1)$-го приближения неизвестной $x_i$ учитываются уже вычисленные ранее $(k+1)$-е приближения $(x_1, x_2, \ldots, x_{i-1})$.

Для того чтобы применить метод Зейделя к решению системы линейных алгебраических уравнений $\mathbf{A}x = b$ с квадратной невырожденной матрицей $\mathbf{A}$, необходимо предварительно преобразовать эту систему к виду $x = \mathbf{B}x + c$.

Здесь $\mathbf{B}$ — квадратная матрица с элементами $b_{ij}$ ($i, j = 1, 2, ..., n$); $c$ — вектор-столбец с элементами $c_{ij}$ ($i = 1, 2, ..., n$).

Самый простой способ приведения системы к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы выразим неизвестное $x_1$, из второго уравнения — неизвестное $x_2$ и т. д. В результате получим систему

$$\begin{cases} x_1 = b_{12}x_2 + b_{13}x_3 + \ldots + b_{1,n-1}x_{n-1} + b_{1n}x_n + c_1; \\ x_2 = b_{21}x_1 + b_{23}x_3 + \ldots + b_{2,n-1}x_{n-1} + b_{2n}x_n + c_2; \\ x_3 = b_{31}x_1 + b_{32}x_2 + \ldots + b_{3,n-1}x_{n-1} + b_{3n}x_n + c_3; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = b_{n1}x_1 + b_{n2}x_2 + b_{n3}x_3 + \ldots + b_{n,n-1}x_{n-1} + c_n, \end{cases}$$

в которой на главной диагонали матрицы $\mathbf{B}$ находятся нулевые элементы. Остальные элементы выражаются по формулам

$$b_{ij} = -a_{ij}/a_{ii}, \quad c_i = b_i/a_{ii} \quad (i, j = 1, 2, ..., n, j \neq i).$$

Для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы $\mathbf{A}$ были ненулевыми.

**Пример 2.47.** Решить систему линейных алгебраических уравнений с вещественными коэффициентами по методу Зейделя.

$$\begin{cases} 2x_1 + 3x_2 - 4x_3 + x_4 = 3; \\ x_1 - 2x_2 - 5x_3 + x_4 = 2; \\ 5x_1 - 3x_2 + x_3 - 4x_4 = 1; \\ 10x_1 + 2x_2 - x_3 + 2x_4 = 4. \end{cases}$$

Для того чтобы обеспечить достаточные условия сходимости итерационного процесса (преобладающие значения диагональных элементов), преобразуем исходную систему и приведем к удобному виду. Чтобы дальнейшие преобразования были понят-

ны, обозначим уравнения исходной системы буквами А, Б, В и Г соответственно:

$$x_1 = -0,2x_2 + 0,1x_3 - 0,2x_4 - 0,4; \quad (Г)$$
$$x_2 = -0,2x_1 - 0,2x_3 + 0,2; \quad (А - Б)$$
$$x_3 = 0,2x_1 - 0,4x_2 + 0,2x_4 - 0,4; \quad (Б)$$
$$x_4 = 0,333x_1 - 1,111. \quad (2А - Б + 2В - Г)$$

Преобразованную систему будем решать методом Зейделя, при этом получим:

$$x_4^{(k+1)} = 0,333x_1^{(k)} - 1,111;$$
$$x_2^{(k+1)} = -0,2x_1^{(k)} + 0,2x_3^{(k)} + 0,2;$$
$$x_1^{(k+1)} = -0,2x_2^{(k+1)} + 0,1x_3^{(k)} - 0,2x_4^{(k+1)} - 0,4;$$
$$x_3^{(k+1)} = 0,2x_1^{(k+1)} - 0,4x_2^{(k+1)} + 0,2x_4^{(k+1)} - 0,4.$$

В качестве нулевого приближения ($k = 0$) возьмем $x_i^{(0)} = \beta_i$. Зададим количество итераций $k = 2$.

| Итерация $k$ | Значения неизвестных | | | | Невязки | | | |
|---|---|---|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\varepsilon_1$ | $\varepsilon_2$ | $\varepsilon_3$ | $\varepsilon_4$ |
| 0 | –0,4 | 0,2 | –0,4 | –1,111 | –2,711 | –1,911 | 0,444 | –1,422 |
| 1 | –0,263 | 0,36 | –0,846 | –1,244 | –0,309 | 1,0 | 0,734 | 0,446 |
| 2 | –0,329. | 0,422 | –0,874 | –1,199 | 0,095 | –0,00 | 0,009 | 0,029 |

В приведенной таблице, кроме значений неизвестных, на каждом шаге оценивались *невязки*. Так как используется итерационный метод, значения неизвестных вычисляются приближенно, то, подставляя значения неизвестных в *исходную* систему, справа получим не ноль, а некоторые значения, называемые *невязкой* первого, второго, ... уравнений на $k$-м шаге.

***Решение***

В переменную *n* вводится порядок матрицы системы, в переменную *e* — максимальная абсолютная погрешность. С помощью вспомогательной процедуры *ReadSystem* в двухмерный массив *a* и одномерный массив *b* вводится с клавиатуры расширенная матрица системы. Начальное приближение предполагается равным

нулю. Оба массива и переменные *n* и *e* передаются функции *Seidel*, в которой исследуется сходимость системы. И в том случае, если система не сходится, выполнение функции прекращается с результатом *false*. В ходе каждой итерации вычисляется новое приближение и абсолютная погрешность. Когда полученная погрешность становится меньше заданной, выполнение функции прекращается. Полученное решение выводится на экран с помощью вспомогательной процедуры *WriteX*.

**Программа**

```
Uses CRT;
Const
    maxn = 10;
Type
  Data = Real;
  Matrix = Array[1..maxn, 1..maxn] of Data;
  Vector = Array[1..maxn] of Data;
{ Процедура ввода расширенной матрицы системы }
Procedure ReadSystem(n: Integer; var a: Matrix; var b: Vector);
Var
   i, j, r: Integer;
Begin
  r := WhereY;  GotoXY(2, r);  Write('A');
  For i := 1 to n do begin
    GotoXY(i * 6 + 2, r);  Write(i);
    GotoXY(1, r + i + 1); Write(i:2);
  end;
  GotoXY((n + 1) * 6 + 2, r);  Write('b');
  For i := 1 to n do begin
    For j := 1 to n do begin
      GotoXY(j * 6 + 2, r + i + 1);  Read(a[i, j]);
    end;
    GotoXY((n + 1) * 6 + 2, r + i + 1);  Read(b[i]);
  end;
End;
{ Процедура вывода результатов }
Procedure WriteX(n :Integer; x: Vector);
Var
   i: Integer;
Begin
  For i := 1 to n do  Writeln('x', i, ' = ', x[i]);
End;
```

```
{ Функция, реализующая метод Зейделя }
Function Seidel(n: Integer; a: Matrix; b: Vector; var x: Vector; e: Data)
:Boolean;
Var      i, j: Integer;
       s1, s2, s, v, m: Data;
Begin       { Исследуем сходимость }
  For i := 1 to n do begin
  s := 0;
  For j := 1 to n do  If j <> i then  s := s + Abs(a[i, j]);
    If s >= Abs(a[i, i]) then begin
      Seidel := false;
      Exit;
  end;
end;
Repeat
  m := 0;
  For i := 1 to n do begin
    { Вычисляем суммы }
    s1 := 0;   s2 := 0;
    For j := 1 to i - 1 do    s1 := s1 + a[i, j] * x[j];
      For j := i to n do    s2 := s2 + a[i, j] * x[j];
    { Вычисляем новое приближение и погрешность }
    v := x[i];   x[i] := x[i] - (1 / a[i, i]) * (s1 + s2 - b[i]);
    If Abs(v - x[i]) > m then    m := Abs(v - x[i]);
  end;
Until m < e;
Seidel := true;
End;
Var
  n, i: Integer;
  a: Matrix;
  b, x: Vector;
  e: Data;
Begin
  ClrScr;
  Writeln('Программа реш. систем лин. уравнений по методу
  Зейделя');
  Writeln;
  Writeln('Введите порядок матрицы системы (макс. 10)');
  Repeat
    Write('>');  Read(n);
  Until (n > 0) and (n <= maxn);
```

```
  Writeln;
  Writeln('Введите точность вычислений');
  Repeat
    Write('>');   Read(e);
  Until (e > 0) and (e < 1);
    Writeln;
    Writeln('Введите расширенную матрицу системы');
  ReadSystem(n, a, b);
  Writeln;
  { Предполагаем начальное приближение равным нулю }
  For i := 1 to n do x[i] := 0;
  If Seidel(n, a, b, x, e) then begin
    Writeln('Результат вычислений по методу Зейделя');
    WriteX(n, x);
  end
  else
    Writeln('Метод Зейделя не сходится для данной системы');
  Writeln;
End.
```

*Ответ*: $x_1 = 5{,}2$; $x_2 = -4{,}2$; $x_3 = 3$; $x_4 = -1{,}8$.

#### 2.5.1.2. Метод прогонки

Метод прогонки основан на предположении, что искомые неизвестные связаны рекуррентным соотношением:

$$x_i = \alpha_{i+1} x_{i+1} + \beta_{i+1}.$$

Используя это соотношение, выразим $x_{i-1}$ и $x_i$ через $x_{i+1}$ и подставим в $i$-е уравнение:

$$(A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i)x_i + A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0,$$

где $F_i$ — правая часть $i$-го уравнения. Это соотношение будет выполняться независимо от решения, если потребовать

$$A_i\alpha_i\alpha_{i+1} + C_i\alpha_{i+1} + B_i = 0;$$

$$A_i\alpha_i\beta_{i+1} + A_i\beta_i + C_i\beta_{i+1} - F_i = 0.$$

Отсюда следует:

$$\alpha_{i+1} = \frac{-B_i}{A_i\alpha_i + C_i}; \quad \beta_{i+1} = \frac{F_i - A_i\beta_i}{A_i\alpha_i + C_i}.$$

Из уравнения $x_0 = \alpha_1 x_1 + \beta_1$ получим $\alpha_1 = 0$; $\beta_1 = x_0$.

После нахождения прогоночных коэффициентов $\alpha$ и $\beta$ получим решение системы.

### 2.5.2. Метод Гаусса

Метод Гаусса — метод последовательного исключения неизвестных, имеет много разных вычислительных схем. Рассмотрим схему единственного деления, алгоритм которого состоит в следующем.

*Прямой ход*: путем элементарных преобразований строк (прибавлений к строке другой строки, умноженной на число и перестановок строк) матрица приводится к верхнетреугольному виду.

С этого момента начинается *обратный ход*. Из последнего ненулевого уравнения выражаем каждую из базисных переменных через небазисные и подставляем в предыдущие уравнения. Повторяя эту процедуру для всех базисных переменных, получаем фундаментальное решение.

**Например**, пусть дана расширенная матрица некоторой системы $m$ линейных уравнений с $n$ неизвестными:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array}\right).$$

Будем считать, что $a_{11} \neq 0$. (Если это не так, то достаточно переставить первую и некоторую другую строку расширенной матрицы местами.) Проведем следующие элементарные преобразования:

$$\begin{array}{l} C_2 - \dfrac{a_{21}}{a_{11}} C_1 \\ \dots \\ C_m - \dfrac{a_{m1}}{a_{11}} C_1 \end{array}, \quad \text{т. е. } C_i - \frac{a_{i1}}{a_{11}} C_1, \ i = 2, 3, \dots, m,$$

т. е. из каждой строки расширенной матрицы (кроме первой) вычитаем первую строку, умноженную на частное от деления первого элемента этой строки на диагональный элемент $a_{11}$.

В результате получим матрицу

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & & a_{1n} & b_1 \\ 0 & \alpha_{22} & \dots & \alpha_{2n} & \beta_2 \\ \dots & \dots & & \dots & \dots \\ 0 & \alpha_{m2} & \dots & \alpha_{mn} & \beta_m \end{array}\right),$$

т. е. первая строка осталась без изменений, а в столбце под $a_{11}$ на всех местах оказались нули. Обратите внимание, что преобразования коснулись всех элементов строк, начиная со второй, всей расширенной матрицы системы.

Теперь задача состоит в том, чтобы получить нули подо всеми диагональными элементами матрицы $A - a_{ij}$, где $i = j$.

Повторим элементарные преобразования, но уже для элемента $a_{22}$

$$\begin{array}{l} C_1 - \dfrac{a_{12}}{a_{22}} C_2 \\ \dots \\ C_m - \dfrac{a_{m2}}{a_{22}} C_2 \end{array}, \quad \text{т. е. } C_i - \frac{a_{i2}}{a_{22}} C_2, \ i = 3, \dots, m,$$

т. е. из каждой строки расширенной матрицы (теперь кроме первой и второй) вычитаем вторую строку, умноженную на частное от деления первого элемента этой (текущей) строки на диагональный элемент $a_{22}$.

Такие преобразования продолжаются до тех пор, пока матрица не приведется к верхнетреугольному виду. То есть под главной диагональю не окажутся все нули:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & & a_{1n} & b_1 \\ 0 & \alpha_{22} & \dots & \alpha_{2n} & \beta_2 \\ \dots & & & \dots & \dots \\ 0 & \alpha_{m2} & \dots & \gamma_{mn} & \delta_m \end{array}\right),$$

Вспомнив, что каждая строка представляет собой одно из уравнений линейной системы уравнений, легко заметить, что последнее $m$-е уравнение принимает вид

$$\gamma_{mn} x_n = \delta m.$$

Отсюда легко можно найти значение первого корня

$$x_n = \frac{\delta m}{\gamma_{mn}}.$$

Подставив это значение в предыдущее $(m-1)$-е уравнение, легко получим значение $x_{n-1}$-го корня.

Таким образом, поднимаясь до самого верха обратным ходом метода Гаусса, последовательно найдем все корни системы уравнений.

**Пример 2.48.** Решить по методу Гаусса систему из трех уравнений с тремя неизвестными

$$\begin{cases} 3x_1 - 2x_2 + 5x_3 = 7; \\ 7x_1 + 4x_2 - 8x_3 = 3; \\ 5x_1 - 3x_2 - 4x_3 = -12. \end{cases}$$

***Решение***

Запишем расширенную матрицу системы: $\left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 7 & 4 & -8 & 3 \\ 5 & -3 & -4 & -12 \end{array}\right)$.

Проведем элементарные преобразования для первого диагонального элемента:

$$\left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 7 & 4 & -8 & 3 \\ 5 & -3 & -4 & -12 \end{array}\right); \qquad \underline{C_i - \frac{a_{i1}}{a_{11}} C_1,\ i = 2, 3, \ldots, m}$$

$$\left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & 4 - \frac{7}{3}\cdot(-2) & -8 - \frac{7}{3}\cdot 5 & 3 - \frac{7}{3}\cdot 7 \\ 0 & -3 - \frac{7}{3}\cdot(-2) & -4 - \frac{7}{3}\cdot 5 & -12 - \frac{7}{3}\cdot 7 \end{array}\right) =$$

$$= \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & 4 + \frac{14}{3} & -8 - \frac{35}{3} & 3 - \frac{49}{3} \\ 0 & -3 + \frac{14}{3} & -4 - \frac{35}{3} & -12 - \frac{49}{3} \end{array}\right) = \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & \frac{5}{3} & -\frac{47}{3} & -\frac{85}{3} \end{array}\right).$$

Теперь проведем элементарные преобразования для второго диагонального элемента

$$\left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & \frac{5}{3} & -\frac{47}{3} & -\frac{85}{3} \end{array}\right); \qquad C_i - \frac{a_{i2}}{a_{22}} C_2,\ i = 3, \ldots, m$$

$$\left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & 0 & -\frac{47}{3} - \left(\frac{5}{3} \Big/ \frac{26}{3}\right) \cdot \frac{59}{3} & -\frac{85}{3} - \left(\frac{5}{3} \Big/ \frac{3}{26}\right) \cdot \left(-\frac{40}{3}\right) \end{array}\right) =$$

$$= \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & 0 & -\frac{47}{3} - \left(\frac{5}{3} \cdot \frac{3}{26}\right) \cdot \frac{59}{3} & -\frac{85}{3} - \left(\frac{5}{3} \cdot \frac{3}{26}\right) \cdot \left(-\frac{40}{3}\right) \end{array}\right) =$$

$$= \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & 0 & -\frac{47}{3} - \frac{5}{26} \cdot \frac{59}{3} & -\frac{85}{3} - \frac{5}{26} \cdot \left(-\frac{40}{3}\right) \end{array}\right) =$$

$$= \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & 0 & -\frac{1222 + 295}{78} & -\frac{2210 + 200}{78} \end{array}\right) = \left(\begin{array}{ccc|c} 3 & -2 & 5 & 7 \\ 0 & \frac{26}{3} & \frac{59}{3} & -\frac{40}{3} \\ 0 & 0 & -\frac{1517}{78} & -\frac{2410}{78} \end{array}\right).$$

Получили матрицу верхнетреугольного вида.

Теперь можем применить обратный ход метода Гаусса. Последнее уравнение принимает вид

$$-\frac{1517}{78}x_3 = -\frac{2410}{78}.$$

Отсюда

$$x_3 = -\frac{2410}{78} \Big/ \left(-\frac{1517}{78}\right) =$$

$$= -\frac{2410}{78} \cdot \left(-\frac{78}{1517}\right) = \frac{2410}{1517} = 1{,}588661832564 \approx 1{,}59.$$

Подставляем это значение в предыдущее уравнение, которое принимает вид

$$\frac{26}{3}x_2 + \frac{59}{3} \cdot 1{,}59 = -\frac{40}{3};$$

$$\frac{26}{3}x_2 + \frac{93{,}73}{3} = -\frac{40}{3}.$$

Отсюда

$$\frac{26}{3}x_2 = -\frac{40}{3} - \frac{93{,}73}{3};$$

$$\frac{26}{3}x_2 = -\frac{133{,}73}{3};$$

$$x_2 = -\frac{133{,}73}{3} \Big/ \frac{26}{3} = -\frac{133{,}73}{26} = -5{,}143461538462 \approx -5{,}14.$$

Подставляем это значение и предыдущий корень в первое уравнение, которое принимает вид:

$$3x_1 - 2(-5{,}14) + 5 \cdot 1{,}59 = 7;$$

$$3x_1 + 2(5{,}14) + 5 \cdot 1{,}59 = 7;$$

$$3x_1 = 7 - 2(5{,}14) - 5 \cdot 1{,}59;$$

$$3x_1 = 7 - 10{,}28 - 7{,}95 = 7 - 18{,}23 = -11{,}23;$$

$$x_1 = -\frac{11{,}23}{3} = -3{,}743333333333 \approx -3{,}74.$$

**Программа**

```
Program GAUS;     {Метод Гаусса для системы }
                  {уравнений с тремя неизвестными}
Uses Crt;
VAR    A:array[1..3,1..3] of real;
       d,b, x: array [1..3] of Real;
           m,s:real; i,j,k:integer;
begin
Clrscr;
for i:=1 to 3 do begin
for j:=1 to 3 do read(a[i,j]);   readln;  end;
for i:=1 to 3 do readln(b[i]);
{Исключение неизвестных}
for k:=1 to 2 do
for i:=k+1 to 3 do
    begin
    m:=a[i,k]/a[k,k];
    for j:=k+1 to 3 do  a[i,j]:=a[i,j]-m*a[k,j]; b[i]:=b[i]-m*b[k];
    end;
{Обратная подстановка}
x[3]:=b[3]/a[3,3];
for i:=2 downto 1 do
    begin
    s:=0;
    for j:=i+1 to 3 do s:=s+a[i,j]*x[j];
    x[i]:=(b[i]-s)/a[i,i];
    end;
for i:=1 to 3 do
writeln('x',i,'=',x[i]);
Readln;
end.
```

**Пример 2.49.** Решить систему линейных уравнений

$$\begin{cases} x_1 - x_2 + x_3 = 2; \\ 2x_1 - x_2 + x_3 = 3; \\ x_1 + x_2 - 2x_3 = -3 \end{cases}$$

методом Гаусса с точностью до 0,0001.

***Решение***

Определитель системы не равен нулю, поэтому система совместна и определена (решение единственно). Выполним преобразования.

Первое уравнение оставим без изменения. Для того чтобы избавиться от первого неизвестного во втором и третьем уравнениях, к ним прибавим первое, умноженное на –2 в первом случае и на –1 во втором

$$\begin{cases} x_1 - x_2 + x_3 = 2; \\ x_2 - x_3 = -1; \\ 2x_2 - 3x_3 = -5. \end{cases}$$

Теперь избавимся от второго неизвестного в третьем уравнении. Для этого второе уравнение умножим на –2 и прибавим к третьему. Получим эквивалентную заданной систему треугольного вида

$$\begin{cases} x_1 - x_2 + x_3 = 2; \\ x_2 - x_3 = -1; \\ -x_3 = -3. \end{cases}$$

Решаем систему снизу вверх. Из третьего уравнения имеем $x_3 = 3$ и, подставляя его во второе уравнение, находим $x_2 = 2$. Поставив найденные неизвестные в первое уравнение, получим $x_1 = 1$.

Таким образом, получим решение системы: $x_1 = 1$, $x_2 = 2$, $x_3 = 3$.

Проверка:

$$\begin{cases} 1 - 2 + 3 = 2; \\ 2 - 2 + 3 = 3; \\ 1 + 2 - 2 \cdot 3 = -3. \end{cases}$$

Получили три тождества.

**Подпрограмма 1**

Формальные параметры процедуры:

*входные:* n — порядок системы; A — массив коэффициентов системы размером $n \times n$; B — массив-строка свободных членов;

*выходные:* x — массив-строка, в который помещается решение системы.

```
Procedure GAUS (N:integer; A : mas; B : mas1;  VAR X : mas1);
TYPE          MST = ARRAY [1..N] OF REAL;
              MSS = ARRAY [1..N] OF MST;
VAR       A1 : MSS; B1 : MST;
              I, J, M, K : integer;
                  H : real;
```

```
BEGIN
   FOR  I := 1 to N do
     BEGIN
     B1[I] := B[I];
     FOR  J := 1 to N  DO   A1 [I,J] := A[I,J];
     END;
  FOR I := 1 TO N-1 DO
    FOR J := I+1 TO N DO
      BEGIN
      A1[J,I] :=- A1[J,I] / A1[I,I];
      FOR K := I+1 TO N DO    A1[J,K] := A1 [J,K] + A1[J,I]*A1[I,K];
        B1[J] := B1[J] + B1[I]*A1[J,I];
        END;
      X[N] := B1[N] / A1[N,N];
  FOR I := N-1 DOWNTO 1 DO
    BEGIN
      H := B1[I];
      FOR J := I+1 TO N DO    H := H - X[J]*A1[I,J];
      X[I] := H / A1[I,I];
    END;
END;
```

Если необходим контроль за невырожденностью матрицы **A**, то можно предложить воспользоваться другой процедурой GAUS1.

В отличие от первой процедуры в данном случае в выходных параметрах есть переменная *tol*, возвращающая 0 при нормальном завершении работы процедуры, или 1, если на главной диагонали один из элементов равен 0, или 2, если матрица **A** размерностью больше чем $50 \times 50$.

**Подпрограмма 2**

```
Procedure GAUS1 (N:integer; A : mas; B : mas1;
                      VAR X : mas1; VAR TOL : integer);
TYPE          MST = array [1..50] OF real;
              MSS = array [1..50] OF MST;
VAR   A1 : MSS; B1 : MST;
      I, J, M, K : integer;     H : real;
BEGIN
   FOR I := 1 TO N DO
     BEGIN
     B1[I] := B[I];
```

```
      FOR J := 1 TO N DO   A1 [I,J] := A[I,J];
      END;
  TOL := 0;
   IF N > 50 THEN     BEGIN  TOL := 2;   EXIT;   END;
   FOR I := 1 TO  N-1 DO
     FOR J := I+1 TO N DO
       BEGIN
       IF ABS(A1[I,I])< 1.0E-10 THEN
         BEGIN
         TOL := 1;  EXIT;
         END;
       A1[J,I] :=- A1[J,I] / A1[I,I];
       FOR K := I+1 TO N DO   A1[J,K] := A1 [J,K] + A1[J,I]*A1[I,K];
                   B1[J] := B1[J] + B1[I]*A1[J,I]
         END;
     IF ABS(A1[N,N]) < 1.0E-10 THEN
         BEGIN
         TOL := 1; EXIT;
         END;
   X[N] := B1[N] / A1[N,N];
   FOR I := N-1 DOWNTO 1 DO
     BEGIN
     IF ABS(A1[I,I]) < 1.0E-10 THEN
       BEGIN
       TOL := 1;  EXIT;
       END;
     H := B1[I];
     FOR J := I+1 TO N DO   H := H - X[J]*A1[I,J];
     X[I] := H / A1[I,I];
     END;
  END;
```

**Пример 2.50.** Решить систему линейных уравнений, заданную матрицей

| | *A* | *B* | *C* | *D* |
|---|---|---|---|---|
| 1 | 10 | 4 | 0 | 13 |
| 2 | 2 | 10 | 5 | 14 |
| 3 | 0 | 2 | 10 | 15 |

***Решение***

Приводим исходную матрицу к виду

| | *A* | *B* | *C* | *D* |
|---|---|---|---|---|
| 9 | 1 | 0,4 | 0 | 1,3 |
| 10 | 0 | 9,2 | 5 | 11,4 |
| 11 | 0 | 0 | 8,913 | 12,52174 |

Используем формулы:

$$X_1 = (D_1 - B_1B_{14})/A_1; \quad X_2 = (D_{10} - C_{10}B_{15})/B_{10}; \quad X_3 = D_{11}/C_{11}.$$

Находим аргументы:

| | |
|---|---|
| $X_1 =$ | 1,109758 |
| $X_2 =$ | 0,475606 |
| $X_3 =$ | 1,404885 |

**Программа**

```
unit gauss;
{----------------------------}
interface
const     ndim = 20;
type
  d_vector = array[1..ndim] of double;
  d_matrix = array[1..ndim,1..ndim] of double;
procedure leq( a : d_matrix; var b : d_vector; n : word);
implementation
{----------------------------}
procedure leq( a : d_matrix; var b : d_vector; n : word);
{Решение системы линейных уравнений методом Гаусса.
На входе — матрица системы a, число уравнений n и вектор-столбец b.
На выходе решение системы содержится в векторе b}
var
  i,j,k,l : word;
  s,t : double;
begin
  if (n > ndim) then n := ndim;
{ Каждое уравнение делится на наибольший элемент
соответствующей строки матрицы a}
for i := 1 to n do
```

```
  begin
    s := abs(a[i,1]);
    for k := 2 to n do
    begin
    t := abs(a[i,k]);    if ( t > s) then s := t;
    end;
    for k := 1 to n do a[i,k] := a[i,k]/s;   b[i] := b[i]/s;
    end;
{ Прямой ход }
for k := 1 to n do
  begin
{ Поиск j-го уравнения, имеющего наибольший a[j, j]}
  j := k;     s := abs(a[k,k]);
  for i := (k+1) to n do
  begin
  t := abs(a[i,k]);
  if ( t > s) then   begin  s := t;    j := i;  end
  end;
{ найденное уравнение меняется местами с текущим k-м}
  if ( j <> k) then
  begin
  for l := k to n do
    begin   s := a[j,l];   a[j,l] := a[k,l];  a[k,l] := s;  end;
  s := b[j];      b[j] := b[k];      b[k] := s;
    end;
{ делим k-е уравнение на a[k,k]}
  s := a[k,k];    a[k,k] := 1.0;
  for l := (k+1) to n do   a[k,l] := a[k,l]/s;    b[k] := b[k]/s;
{ в этом цикле — исключение столбца ниже диагонали}
  for i := (k+1) to n do
    begin
    s := a[i,k];      a[i,k] := 0.0;
    for l := (k+1) to n do  a[i,l] := a[i,l] - a[k,l]*s;    b[i] := b[i] - b[k]*s
    end;
  end;
{Обратный ход}
for i := (n-1) downto 1 do
  begin
  s := b[i];
  for k := (i+1) to n do s := s - a[i,k]*b[k];     b[i] := s
  end
end;
End.
```

**Пример 2.51.** Решить систему линейных уравнений методом Гаусса с точностью до 0,001:

$$\begin{cases} 3{,}2x_1 + 5{,}4x_2 + 4{,}2x_3 + 2{,}2x_4 = 2{,}6; \\ 2{,}1x_1 + 3{,}2x_2 + 3{,}1x_3 + 1{,}1x_4 = 4{,}8; \\ 1{,}2x_1 + 0{,}4x_2 - 0{,}8x_3 - 0{,}8x_4 = 3{,}6; \\ 4{,}7x_1 + 10{,}4x_2 + 9{,}7x_3 + 9{,}7x_4 = -8{,}4. \end{cases}$$

***Решение***

В данной программе реализован метод Гаусса со схемой частичного выбора.

В переменную *n* вводится порядок матрицы системы. С помощью вспомогательной процедуры *ReadSystem* в двухмерный массив *a* и одномерный массив *b* вводится с клавиатуры расширенная матрица системы, после этого оба массива и переменная *n* передаются функции *Gauss*. В фукции *Gauss* для каждого *k*-го шага вычислений выполняется поиск максимального элемента в *k*-м столбце матрицы, начиная с *k*-й строки. Номер строки, содержащей максимальный элемент, сохраняется в переменной *l*. В том случае если максимальный элемент находится не в *k*-й строке, строки с номерами *k* и *l* меняются местами. Если же все эти элементы равны нулю, то происходит прекращение выполнения функции *Gauss* с результатом *false*. После выбора строки выполняется преобразование матрицы по методу Гаусса. Далее вычисляется решение системы и помещается в массив *x*. Полученное решение выводится на экран с помощью вспомогательной процедуры *WriteX*.

**Программа**

```
Program MetGauss;
Uses CRT;
Const  maxn = 10;
Type
  Data = Real;
  Matrix = Array[1..maxn, 1..maxn] of Data;
  Vector = Array[1..maxn] of Data;
{ Процедура ввода расширенной матрицы системы }
Procedure ReadSystem(n: Integer; var a: Matrix; var b: Vector);
Var
  i, j, r: Integer;
```

```
Begin
  r := WhereY;  GotoXY(2, r);  Write('A');
  For i := 1 to n do begin
    GotoXY(i*6+2, r);  Write(i);  GotoXY(1, r+i+1);  Write(i:2);
    end;
  GotoXY((n+1)*6+2, r);  Write('b');
  For i := 1 to n do begin
  For j := 1 to n do begin
      GotoXY(j * 6 + 2, r + i + 1);  Read(a[i, j]);
      end;
    GotoXY((n + 1) * 6 + 2, r + i + 1); Read(b[i]);
  end;
End;
{ Процедура вывода результатов }
Procedure WriteX(n :Integer; x: Vector);
Var
  i: Integer;
    Begin
    For i := 1 to n·do  Writeln('x', i, ' = ', x[i]);
    End;
{ Функция, реализующая метод Гаусса }
Function Gauss(n: Integer; a: Matrix; b: Vector; var x:Vector): Boolean;
Var
  i, j, k, l: Integer;
  q, m, t: Data;
Begin
  For k := 1 to n - 1 do begin
    { Ищем строку l с максимальным элементом в k-м столбце}
    l := 0;   m := 0;·
    For i := k to n do
    If  Abs(a[i, k]) > m  then begin  m := Abs(a[i, k]);  l := i; end;
    { Если у всех строк от k до n элемент в k-м столбце нулевой,
        то система не имеет однозначного решения }
    If  l = 0  then  begin  Gauss := false;  Exit;  end;
    { Меняем местами  l-ю строку с k-й }
    If l <> k then begin
      For j := 1 to n do begin
        t := a[k, j];  a[k, j] := a[l, j];  a[l, j] := t;
        end;
      t := b[k];  b[k] := b[l];  b[l] := t;
      end;
      { Преобразуем матрицу }
    For i := k + 1 to n do begin   q := a[i, k] / a[k, k];
      For j := 1 to n do  If  j = k  then  a[i, j] := 0
```

```
        else  a[i, j] := a[i, j] - q * a[k, j];   b[i] := b[i] - q * b[k];
        end;
    end;
    { Вычисляем решение }
    x[n] := b[n] / a[n, n];
    For i := n - 1 downto 1 do begin  t := 0;
      For j := 1 to n-i do   t := t + a[i, i + j] * x[i + j];
      x[i] := (1 / a[i, i]) * (b[i] - t);
    end;
    Gauss := true;
  End;
  Var
    n, i: Integer;
    a: Matrix ;
    b, x: Vector;
  Begin
    ClrScr;
    Writeln('Реш. систем лин. уравнений по методу Гаусса');
    Writeln;
    Writeln('Введите порядок матрицы системы (макс. 10)');
      Repeat
      Write('>');  Read(n);
      Until (n > 0) and (n <= maxn);
    Writeln;
    Writeln('Введите расширенную матрицу системы');
    ReadSystem(n, a, b); Writeln;
    If Gauss(n, a, b, x) then begin
      Writeln('Результат вычислений  методом  Гаусса');
      WriteX(n, x);
    end
    else
    Writeln('Данную систему невозможно решить по методу Гаусса');
    Writeln;
  Readln;
  End.
```

*Ответ:* $x_1 = 5$, $x_2 = -4$, $x_3 = 3$, $x_4 = -2$.

### 2.5.3. Метод Жордана — Гаусса

**Метод Жордана — Гаусса** используется для решения систем линейных алгебраических уравнений, нахождения обратной матрицы, нахождения координат вектора в заданном базисе, оты-

скания ранга матрицы. Метод является модификацией метода Гаусса.

*Алгоритм*

1. Выбирается первая колонка слева, в которой есть хоть одно отличное от нуля значение.

2. Если самое верхнее число в этой колонке есть нуль, то меняется вся первая строка матрицы с другой ее строкой, где в этой колонке нет нуля.

3. Все элементы первой строки делятся на верхний элемент выбранной колонки.

4. Из оставшихся строк вычитается соотвественно многократно первая строка с целью получить первым элементом каждой строки (кроме первой) нуль.

5. Вычеркивается первая строка и первая колонка и на оставшейся матрице повторяем предыдущие шаги.

### 2.5.4. Метод Крамера

**Метод Крамера (правило Крамера)** — способ решения квадратных систем линейных алгебраических уравнений с ненулевым определителем основной матрицы (причем для таких уравнений решение существует и оно единственно).

**Описание метода.** Для системы $n$ линейных уравнений с $n$ неизвестными (над произвольным полем)

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + \ldots + a_{nn}x_n = b_n \end{cases}$$

с определителем матрицы системы $\Delta$, отличным от нуля, решение записывается в виде

$$x_i = \frac{\Delta_i}{\Delta},$$

где

$$\Delta_i = \begin{vmatrix} a_{11}\ldots a_{1,j-1}b_1 a_{1,j+1}\ldots a_{1n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}\ldots a_{n,j-1}b_n a_{n,j+1}\ldots a_{nn} \end{vmatrix}$$

($i$-й столбец матрицы системы заменяется столбцом свободных членов).

Пусть дана система линейных уравнений с двумя неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = c_1; \\ a_{21}x_1 + a_{22}x_2 = c_2. \end{cases}$$

Если главный определитель системы отличен от нуля, то система имеет решение, притом единственное. Решение системы определяется формулами

$$x_1 = \frac{\Delta x_1}{\Delta}; \quad x_2 = \frac{\Delta x_2}{\Delta},$$

где $x_1$, $x_2$ — корни системы уравнений; $\Delta$ — главный определитель системы; $\Delta x_1$, $\Delta x_2$ — вспомогательные определители.

Главный определитель системы вычисляется по схеме:

$$\Delta = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

Вспомогательные определители:

$$\Delta x_1 = \begin{vmatrix} c_1 & a_{12} \\ c_2 & a_{22} \end{vmatrix} = c_1a_{22} - a_{12}c_2.$$

$$\Delta x_2 = \begin{vmatrix} a_{11} & c_1 \\ a_{12} & c_2 \end{vmatrix} = a_{11}c_2 - c_1a_{12}.$$

**Пример 2.52.** Решить систему линейных уравнений

$$\begin{cases} 3x_1 + 4x_2 - 5x_3 = 6; \\ -2x_1 + 5x_2 - 3x_3 = 4; \\ 2x_1 - 3x_2 + x_3 = 2 \end{cases}$$

а) с помощью обратной матрицы;

б) с помощью формул Крамера.

***Решение***

а) $\begin{cases} 3x_1 + 4x_2 - 5x_3 = 6; \\ -2x_1 + 5x_2 - 3x_3 = 4; \\ 2x_1 - 3x_2 + x_3 = 2; \end{cases}$

$$\Delta = \begin{vmatrix} 3 & 4 & -5 \\ -2 & 5 & -3 \\ 2 & -3 & 1 \end{vmatrix} = 3 \cdot 5 \cdot 1 + 4 \cdot (-3) \cdot 2 + (-5) \cdot (-2) \cdot (-3) - (-5) \times$$

$$\times 5 \cdot 2 - 3 \cdot (-3) \cdot (-3) - 4 \cdot (-2) \cdot 1 = -8 \neq 0.$$

$$\widetilde{\mathbf{A}} = \begin{vmatrix} -4 & -4 & -4 \\ 11 & 13 & 17 \\ 13 & 19 & 23 \end{vmatrix}, \quad \widetilde{\mathbf{A}}^{\mathrm{T}} = \begin{vmatrix} -4 & 11 & 13 \\ -4 & 13 & 19 \\ -4 & 17 & 23 \end{vmatrix}.$$

$$\mathbf{A}^{-1} = -\frac{1}{8} \begin{vmatrix} -4 & 11 & 13 \\ -4 & 13 & 19 \\ -4 & 17 & 23 \end{vmatrix} = \begin{vmatrix} \frac{4}{8} & \frac{-11}{8} & \frac{-13}{8} \\ \frac{4}{8} & \frac{-13}{8} & \frac{-19}{8} \\ \frac{4}{8} & \frac{-17}{8} & \frac{-23}{8} \end{vmatrix},$$

$$\begin{vmatrix} \frac{4}{8} & \frac{-11}{8} & \frac{-13}{8} \\ \frac{4}{8} & \frac{-13}{8} & \frac{-19}{8} \\ \frac{4}{8} & \frac{-17}{8} & \frac{-23}{8} \end{vmatrix} \cdot \begin{vmatrix} 6 \\ 4 \\ 2 \end{vmatrix} = \begin{vmatrix} \frac{-46}{8} \\ \frac{-66}{8} \\ \frac{-90}{8} \end{vmatrix},$$

$$\frac{4}{8} \cdot 6 + \left(\frac{-11}{8}\right) \cdot 4 + \left(\frac{-13}{8}\right) \cdot 2 = \left(\frac{-46}{8}\right),$$

$$\frac{4}{8} \cdot 6 + \left(\frac{-13}{8}\right) \cdot 4 + \left(\frac{-19}{8}\right) \cdot 2 = \left(\frac{-66}{8}\right),$$

$$\frac{4}{8} \cdot 6 + \left(\frac{-17}{8}\right) \cdot 4 + \left(\frac{-23}{8}\right) \cdot 2 = \left(\frac{-90}{8}\right).$$

*Ответ:*

$$x_1 = \left(\frac{-46}{8}\right) = -5{,}75, \quad x_2 = \left(\frac{-66}{8}\right) = -8{,}25, \quad x_3 = \left(\frac{-90}{8}\right) = -11{,}25.$$

б) $\begin{cases} 3x_1 + 4x_2 - 5x_3 = 6; \\ -2x_1 + 5x_2 - 3x_3 = 4; \\ 2x_1 - 3x_2 + x_3 = 2; \end{cases}$

$$\Delta = \begin{vmatrix} 3 & 4 & -5 \\ -2 & 5 & -3 \\ 2 & -3 & 1 \end{vmatrix} = 3 \cdot 5 \cdot 1 + 4 \cdot (-3) \cdot 2 + (-5) \cdot (-2) \cdot (-3) - (-5) \times$$

$$\times 5 \cdot 2 - 3 \cdot (-3) \cdot (-3) - 4 \cdot (-2) \cdot 1 = -8 \neq 0.$$

$$x_1 = -\frac{1}{8}\begin{vmatrix} 6 & 4 & -5 \\ 4 & 5 & -3 \\ 2 & -3 & 1 \end{vmatrix} = -\frac{1}{8}(6\begin{vmatrix} 5 & -3 \\ -3 & 1 \end{vmatrix} - 4\begin{vmatrix} 4 & -3 \\ 2 & 1 \end{vmatrix} - 5\begin{vmatrix} 4 & 5 \\ 2 & -3 \end{vmatrix}) =$$

$$= -\frac{1}{8}(6 \cdot (5 - 9) - 4 \cdot (4 + 6) - 5 \cdot (-12 - 10)) = \left(-\frac{46}{8}\right) = -5{,}75.$$

$$x_2 = -\frac{1}{8}\begin{vmatrix} 3 & 6 & -5 \\ -2 & 4 & -3 \\ 2 & 2 & 1 \end{vmatrix} = -\frac{1}{8}(3\begin{vmatrix} 4 & -3 \\ 2 & 1 \end{vmatrix} - 6\begin{vmatrix} -2 & -3 \\ 2 & 1 \end{vmatrix} - 5\begin{vmatrix} -2 & 4 \\ 2 & 2 \end{vmatrix}) =$$

$$= -\frac{1}{8}(3 \cdot (4 + 6) - 6 \cdot (-2 + 6) - 5 \cdot (-4 - 8)) = \left(-\frac{66}{8}\right) = -8{,}25.$$

$$x_3 = -\frac{1}{8}\begin{vmatrix} 3 & 4 & 6 \\ -2 & 5 & 4 \\ 2 & -3 & 2 \end{vmatrix} = -\frac{1}{8}(3\begin{vmatrix} 5 & 4 \\ -3 & 2 \end{vmatrix} - 4\begin{vmatrix} -2 & 4 \\ 2 & 2 \end{vmatrix} + 6\begin{vmatrix} -2 & 5 \\ 2 & -3 \end{vmatrix}) =$$

$$= -\frac{1}{8}(3 \cdot (10 + 22) - 4 \cdot (-4 - 8) + 6 \cdot (6 - 10)) = \left(-\frac{90}{8}\right) = -11{,}25.$$

*Ответ*: $x_1 = \left(-\frac{46}{8}\right) = -5{,}75$, $x_2 = \left(-\frac{66}{8}\right) = -8{,}25$,

$x_3 = \left(-\frac{90}{8}\right) = -11{,}25.$

**Пример 2.53.** Определить, является ли система совместной и сколько решений она имеет

$$\begin{cases} -3x_1 + 4x_2 - 5x_3 = 6; \\ 3x_1 + 3x_2 - 5x_3 = 8; \\ 4x_1 - 5x_2 + x_3 = 2. \end{cases}$$

***Решение***

$$\mathbf{A} = \begin{pmatrix} -3 & 4 & -5 \\ 3 & 3 & -5 \\ 4 & -5 & 1 \end{pmatrix}, \quad \mathbf{B} = \left(\begin{array}{ccc|c} -3 & 4 & -5 & 6 \\ 3 & 3 & -5 & 8 \\ 4 & -5 & 1 & 2 \end{array}\right).$$

$$\Delta\mathbf{A} = \begin{vmatrix} -3 & 4 & -5 \\ 3 & 3 & -5 \\ 4 & -5 & 1 \end{vmatrix} = 109 \neq 0;$$

$\Delta\mathbf{A} = 109 \neq 0$, следовательно rang (**A**) = 3.

$$\Delta_1 = \begin{vmatrix} -3 & -5 & 6 \\ 3 & -5 & 8 \\ 4 & 1 & 2 \end{vmatrix} = -62 \neq 0;$$

$$\Delta_2 = \begin{vmatrix} 4 & -5 & 6 \\ 3 & -5 & 8 \\ -5 & 1 & 2 \end{vmatrix} = 26 \neq 0;$$

$$\Delta_3 = \begin{vmatrix} -3 & 4 & 6 \\ 3 & 3 & 8 \\ 4 & -5 & 2 \end{vmatrix} = -196 \neq 0.$$

Следовательно, rang (**B**) = 3.

Так как rang (**A**) = rang (**B**) = 3, то система совместна. Так как rang (**A**) = rang (**B**) = $n$, то система имеет единственное решение.

**Пример 2.54.** Решить систему уравнений $\begin{cases} 2x_1 - 3x_2 = -5; \\ 5x_1 + 7x_2 = 31 \end{cases}$ по формулам Крамера.

***Решение***

$$\Delta = \begin{vmatrix} 2 & -3 \\ 5 & 7 \end{vmatrix} = 2 \cdot 7 - (-3) \cdot 5 = 29;$$

$$\Delta x_1 = \begin{vmatrix} -5 & -3 \\ 31 & 7 \end{vmatrix} = (-5) \cdot 7 - (-3) \cdot 31 = -35 + 93 = 58;$$

$$\Delta x_2 = \begin{vmatrix} 2 & -5 \\ 5 & 31 \end{vmatrix} = 2 \cdot 31 - (-5) \cdot 5 = 62 + 25 = 87;$$

$$x_1 = \frac{\Delta x_1}{\Delta} = \frac{58}{29} = 2; \qquad x_2 = \frac{\Delta x_2}{\Delta} = \frac{87}{29} = 3.$$

*Ответ*: $x_1 = 2$; $x_2 = 3$.

**Программа**

```
program korni1;   {Метод Крамера для системы }
label Out;              {уравнений с двумя неизвестными}
var     a: array[1..2,1..3] of integer;
        x,y,dx,dy,d:real; i,j:integer;
begin
for i:=1 to 2 do
for j:=1 to 3 do readln(a[i,j]);
  d:=a[1,1]*a[2,2]-a[2,1]*a[1,2];
  if d=0 then begin
  writeln('Единственного решения нет'); goto Out; end;
  dx:=a[1,3]*a[2,2]-a[2,3]*a[1,2];
  dy:=a[1,1]*a[2,3]-a[2,1]*a[1,3];
  x:=dx/d; y:=dy/d;
writeln('x=',x); writeln('y=',y);
Readln;
Out: end.
```

### 2.5.4.1. Система линейных уравнений с тремя неизвестными

Если главный определитель системы

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1; \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2; \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3 \end{cases}$$

отличен от нуля, то система имеет решение, притом единственное. Решение системы определяется формулами

$$x_1 = \frac{\Delta x_1}{\Delta}; \quad x_2 = \frac{\Delta x_2}{\Delta}; \quad x_3 = \frac{\Delta x_3}{\Delta},$$

где $x_1$, $x_2$, $x_3$ — корни системы уравнений; $\Delta$ — главный определитель системы; $\Delta x_1$, $\Delta x_2$, $\Delta x_3$ — вспомогательные определители.

Главный определитель системы вычисляется по схеме:

$$\Delta = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} \end{vmatrix} =$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}.$$

Вспомогательные определители:

$$\Delta x_1 = \begin{vmatrix} c_1 & a_{12} & a_{13} & c_1 & a_{12} \\ c_2 & a_{22} & a_{23} & c_2 & a_{22} \\ c_3 & a_{32} & a_{33} & c_3 & a_{32} \end{vmatrix} =$$

$$= c_1a_{22}a_{33} + a_{12}a_{23}c_3 + a_{13}c_2a_{32} - a_{13}a_{22}c_3 - c_1a_{23}a_{32} - a_{12}c_2a_{33};$$

$$\Delta x_2 = \begin{vmatrix} a_{11} & c_1 & a_{13} & a_{11} & c_1 \\ a_{21} & c_2 & a_{23} & a_{21} & c_2 \\ a_{31} & c_3 & a_{33} & a_{31} & c_3 \end{vmatrix} =$$

$$= a_{11}c_2a_{33} + c_1a_{23}a_{31} + a_{13}a_{21}c_3 - a_{13}c_2a_{31} - a_{11}a_{23}c_3 - c_1a_{21}a_{33};$$

$$\Delta x_3 = \begin{vmatrix} a_{11} & a_{12} & c_1 & a_{11} & a_{12} \\ a_{21} & a_{22} & c_2 & a_{21} & a_{22} \\ a_{31} & a_{32} & c_3 & a_{31} & a_{32} \end{vmatrix} =$$

$$= a_{11}a_{22}c_3 + a_{12}c_2a_{31} + c_1a_{21}a_{32} - c_1a_{22}a_{31} - a_{11}c_2a_{32} - a_{12}a_{21}c_3.$$

**Пример 2.55.** Решить систему уравнений

$$\begin{cases} 2x + y + z = 7; \\ x + 2y + z = 8; \\ x + y + 2z = 9 \end{cases}$$

по формулам Крамера.

***Решение***

$$\Delta = \begin{vmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{vmatrix} \begin{matrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{matrix} =$$

$$= 2 \cdot 2 \cdot 2 + 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 1 - 1 \cdot 2 \cdot 1 - 2 \cdot 1 \cdot 1 - 1 \cdot 1 \cdot 2 = 4;$$

$$\Delta x = \begin{vmatrix} 7 & 1 & 1 \\ 8 & 2 & 1 \\ 9 & 1 & 2 \end{vmatrix} \begin{matrix} 7 & 1 \\ 8 & 2 \\ 9 & 1 \end{matrix} =$$

$$= 7 \cdot 2 \cdot 2 + 1 \cdot 1 \cdot 9 + 1 \cdot 8 \cdot 1 - 1 \cdot 2 \cdot 9 - 7 \cdot 1 \cdot 1 - 1 \cdot 8 \cdot 2 = 4;$$

$$\Delta y = \begin{vmatrix} 2 & 7 & 1 \\ 1 & 8 & 1 \\ 1 & 9 & 2 \end{vmatrix} \begin{matrix} 2 & 7 \\ 1 & 8 \\ 1 & 9 \end{matrix} =$$

$$= 2 \cdot 8 \cdot 2 + 7 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot 9 - 1 \cdot 8 \cdot 1 - 2 \cdot 1 \cdot 9 - 7 \cdot 1 \cdot 2 = 8;$$

$$\Delta z = \begin{vmatrix} 2 & 1 & 7 \\ 1 & 2 & 8 \\ 1 & 1 & 9 \end{vmatrix} \begin{matrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{matrix} =$$

$$= 2 \cdot 2 \cdot 9 + 1 \cdot 8 \cdot 1 + 7 \cdot 1 \cdot 1 - 7 \cdot 2 \cdot 1 - 2 \cdot 8 \cdot 1 - 1 \cdot 1 \cdot 9 = 12;$$

$$x = \frac{\Delta x}{\Delta} = \frac{4}{4} = 1, \quad y = \frac{\Delta y}{\Delta} = \frac{8}{4} = 2, \quad z = \frac{\Delta z}{\Delta} = \frac{12}{4} = 3.$$

*Ответ*: $x = 1$, $y = 2$, $z = 3$.

**Программа**

```
program korni2; {Метод Крамера для системы }
label Out;          {уравнений с тремя неизвестными}
var                 a:array[1..3,1..4] of integer;
  d1,d2,x,y,z,dx,dy,d,dz:real; i,j:integer;
begin
for i:=1 to 3 do begin
for j:=1 to 4 do read(a[i,j]); readln; end;
d1:=a[1,1]*a[2,2]*a[3,3]+a[1,2]*a[2,3]*a[3,1]+a[1,3]*a[2,1]*a[3,2];
d2:=a[3,1]*a[2,2]*a[1,3]+a[3,2]*a[2,3]*a[1,1]+a[3,3]*a[2,1]*a[1,2];
d:=d1-d2;
```

```
if d=0 then begin   writeln('Единственного решения нет');
  goto Out;
  end;
d1:=a[1,4]*a[2,2]*a[3,3]+a[1,2]*a[2,3]*a[3,4]+a[1,3]*a[2,4]*a[3,2];
d2:=a[3,4]*a[2,2]*a[1,3]+a[3,2]*a[2,3]*a[1,4]+a[3,3]*a[2,4]*a[1,2];
dx:=d1-d2;
d1:=a[1,1]*a[2,4]*a[3,3]+a[1,4]*a[2,3]*a[3,1]+a[1,3]*a[2,1]*a[3,4];
d2:=a[3,1]*a[2,4]*a[1,3]+a[3,4]*a[2,3]*a[1,1]+a[3,3]*a[2,1]*a[1,4];
dy:=d1-d2;
d1:=a[1,1]*a[2,2]*a[3,4]+a[1,2]*a[2,4]*a[3,1]+a[1,4]*a[2,1]*a[3,2];
d2:=a[3,1]*a[2,2]*a[1,4]+a[3,2]*a[2,4]*a[1,1]+a[3,4]*a[2,1]*a[1,2];
dz:=d1-d2; x:=dx/d; y:=dy/d; z:=dz/d;
writeln('x=',x); writeln('y=',y); writeln('z=',z);
Readln;
Out: end.
```

## 2.6. Дифференциальные уравнения

Теория дифференциальных уравнений — раздел математики, который занимается изучением дифференциальных уравнений и связанных с ними задач. Ее результаты применяются во многих естественных науках, особенно широко — в физике.

Неформально говоря, дифференциальное уравнение — это уравнение, в котором неизвестной величиной является некоторая функция. При этом в самом уравнении участвует не только неизвестная функция, но и различные производные от нее.

**Дифференциальным уравнением** называется уравнение, связывающее аргумент, функцию этого аргумента и производные этой функции до некоторого порядка включительно. Наивысший порядок производной, входящей в дифференциальное уравнение, называется порядком уравнения.

Различают обыкновенные дифференциальные уравнения (ОДУ) и дифференциальные уравнения в частных производных (ДУЧП).

Первоначально дифференциальные уравнения возникли из задач механики, в которых участвовали координаты тел, их скорости и ускорения, рассматриваемые как функции времени.

**Обыкновенные дифференциальные уравнения** — это уравнения вида $F(t, x, x', x'', \ldots, x^{(n)}) = 0$, где $x = x(t)$ — неизвестная функция (возможно, вектор-функция; в таком случае часто говорят о сис-

теме дифференциальных уравнений), зависящая от переменной времени $t$, штрих означает дифференцирование по $t$. Число $n$ называется порядком дифференциального уравнения.

**Дифференциальное уравнение в частных производных** — это уравнение, содержащее неизвестные функции от нескольких переменных и их частные производные.

Решение задач на нахождение функции по заданным свойствам сводится к решению уравнения, связывающего искомую функцию и величины, задающие ее свойства. Поскольку свойства функции выражаются через ее производные, то, решая указанную выше задачу, приходим к уравнению, связывающему искомую функцию и ее производные. Такие уравнения называются дифференциальными. Решая полученное дифференциальное уравнение, находят искомую функцию.

Решением дифференциального уравнения называют любую функцию, при подстановке которой в это уравнение получается тождество. График решения дифференциального уравнения называется интегральной кривой этого уравнения.

### 2.6.1. Численное решение дифференциального уравнения

Решить задачу Коши на примере уравнения первого порядка

$$\begin{cases} \dfrac{dy}{dx} = f(x, y), \quad a \le x \le b; \\ y(a) = y_a. \end{cases} \tag{2.10}$$

Уравнения высших порядков можно свести к системе уравнений первого порядка. Например, уравнение второго порядка

$$y'' = f(x, y, y')$$

можно переписать в следующем виде:

$$z' = f(x, y, z);$$

$$y' = z,$$

где $z$ — новая зависимая переменная, определяемая вторым уравнением. Теперь получается система уравнений относитель-

но $y$ и $z$. Решение этой системы дает функцию и ее производную.

Построение численных алгоритмов решения уравнения (2.10) опирается на дискретизацию задачи. Введем в области расчета $x \in [a, b]$ дискретный набор точек $x_i = a + hi$, $i = 0, 1, ..., N$, $h = (b - a)/N$, в которых будет вычисляться приближенное решение. Точки $x_i$ будем называть *узлами интегрирования* или *узлами сетки* (рис. 2.18), расстояние $h$ между узлами — *шагом интегрирования* или *шагом сетки*. Совокупность всех узлов $\{x_i, i = 0, 1, ..., N\}$ будем называть *сеточной областью* или просто *сеткой узлов*.

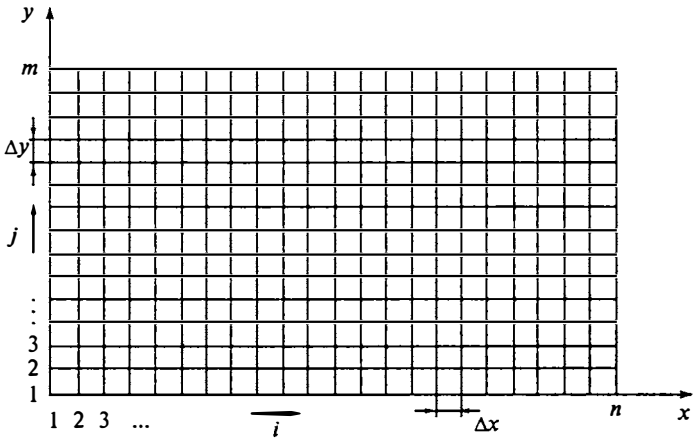



**Рис. 2.18.** Прямоугольная сетка

Также будем пользоваться другими обозначениями:

$\{y_i, i = 0, 1, ..., N\}$ — совокупность искомых приближенных значений решения задачи в узлах сетки;

$\{f_i = f(x_i, y_i), i = 0, 1, ..., N\}$ — совокупность значений правой части уравнения в узлах.

Различные совокупности величин, отнесенных к узлам сетки, называются *сеточными функциями*.

Для характеристики точности численных методов определим погрешность приближенного решения следующим образом:

$$\delta = \max_i |y_i - y(x_i)|,$$

где $y(x_i)$ — значение точного решения в узле сетки.

Метод, по которому получено численное решение, является методом $p$-го порядка точности, если выполняется неравенство

$$\delta \le \text{const} \cdot h^p.$$

Переходим к обсуждению конкретных методов получения приближенного решения задачи в узлах сетки.

Простейший способ их конструирования опирается на замену производной в левой части уравнения в окрестности каждого узла приближенным разностным отношением по формулам численного дифференцирования.

### 2.6.1.1. Метод Эйлера

Заменяя в (2.10) производную в окрестности каждого $i$-го узла сетки разностным отношением, приходим к методу Эйлера:

$$\begin{cases} \dfrac{y_{i+1} - y_i}{h} = f(x_i, y_i), \quad i = 0, 1, \ldots, N-1; \\ y_0 = y_a. \end{cases} \tag{2.11}$$

Алгебраические соотношения между компонентами сеточной функции, которыми заменяются исходные дифференциальные уравнения в окрестности каждого узла сетки, будем называть *разностными уравнениями* (*соотношениями*).

Замкнутую систему разностных уравнений вместе с дополнительными условиями (начальными или краевыми) называют *разностной схемой*. Таким образом, (2.11) — это разностная схема Эйлера.

Последовательные значения $y_i$ вычисляются по формуле

$$y_{i+1} = y_i + hf(x_i, y_i), \tag{2.12}$$

которая непосредственно следует из соотношения (2.11).

Метод Эйлера имеет очень простую геометрическую интерпретацию. Искомая интегральная кривая $y(x)$ на отрезке $[a; b]$ приближается к ломаной (рис. 2.19), наклон которой на каждом элементарном участке $[x_i, x_{i+1}]$ определяется наклоном интегральной кривой уравнения в точке $(x_i, y_i)$.

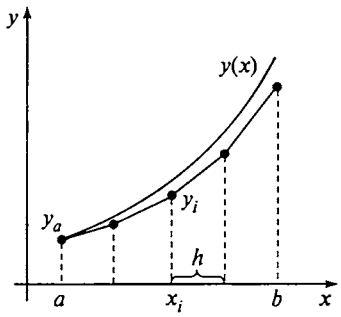



**Рис. 2.19.** Интегральная кривая

*Замечание.* К этому же методу можно придти, заменяя производную в уравнении (2.12) разностным отношением

$$\begin{cases} \dfrac{y_i - y_{i-1}}{h} = f(x_i,\ y_i), \quad i = 0,\ 1,\ \ldots,\ N-1; \\ y_0 = y_a. \end{cases}$$

Последовательные значения $y_i$ в этом случае вычисляются по формуле

$$y_i = y_{i-1} + hf(x_i,\ y_i).$$

Однако при этом возникают некоторые трудности, связанные с тем, что искомая величина $y_i$ входит в правую часть уравнения, причем, в общем случае, нелинейным образом. Эти трудности непринципиальны, достаточно вспомнить о методах решения нелинейных уравнений.

Например, можно предложить следующий итерационный процесс для вычисления приближенного решения в очередном $i$-м узле

$$y_i^{(k)} = y_{i-1} + hf(x_k,\ y_i^{(k-1)}), \quad y_i^{(0)} = y_{i-1}.$$

Такого рода методы, в которых для вычисления приближенного решения в очередном $i$-м узле необходимо дополнительно решать некоторые уравнения (линейные или нелинейные), называются *неявными методами*. В противоположность этому методы, в которых приближенное решение в очередном $i$-м узле явно выражается через предыдущие значения $y_{i-1}$, $y_{i-2}$, ..., называются *явными методами*. При этом, если для вычисления $y_i$ используется только одно предыдущее значение $y_{i-1}$, то метод называется

*одношаговым*, а если несколько предыдущих значений — *многошаговым*. Таким образом, метод Эйлера является явным одношаговым методом (рис. 2.20).

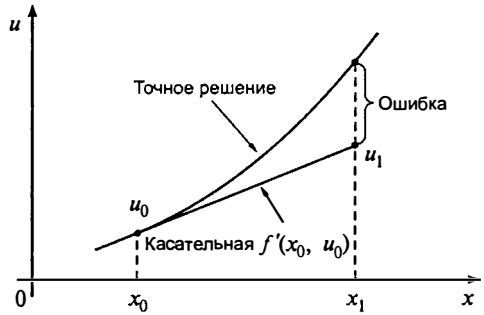



**Рис. 2.20.** Начальный шаг метода Эйлера

В основе метода Эйлера лежит идея *графического построения решения* дифференциального уравнения. Однако этот метод дает одновременно и способ нахождения искомой функции в табличной форме.

Пусть дано дифференциальное уравнение $y' = f(x, y)$. Найти приближенное численное решение этого дифференциального уравнения, т. е. составить таблицу приближенных значений функции $y = y(x)$, удовлетворяющей заданным начальным условиям

| $x$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | ... | $x_n$ |
|---|---|---|---|---|---|---|---|---|
| $y$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | ... | $y_n$ |

где $x_i = x_0 + ih$, $h = \dfrac{x - x_0}{n}$ — шаг таблицы.

Приближенно можно считать, что правая часть в $y' = f(x, y)$ остается постоянной на каждом из отрезков между точками деления. Метод Эйлера состоит в непосредственной замене производной разностными отношениями по приближенной формуле

$$\frac{\Delta y}{\Delta x} = f(x, y);$$

$$y - y_0 = f(x_0, y_0)(x - x_0), \qquad y = y_0 + f(x_0, y_0)(x - x_0);$$

если $x = x_1$, то

$$y_1 = y_0 + f(x_0, y_0)(x_1 - x_0), \quad y_1 = y_0 + hf(x_0, y_0), \quad \Delta y_0 = hf(x_0, y_0);$$

если $x = x_2$, то

$$y_2 = y_1 + f(x_1, y_1)(x_2 - x_1), \quad y_2 = y_1 + hf(x_1, y_1), \quad \Delta y_1 = hf(x_1, y_1),$$

если $x = x_{i+1}$, то

$$y_{i+1} = y_i + hf(x_i, y_i), \quad \Delta y_i = hf(x_i, y_i).$$

Таким образом, получение таблицы значений искомой функции $y(x)$ по методу Эйлера заключается в циклическом применении пары формул

$$\Delta y_k = hf(x_k, y_k), \quad y_{k+1} = y_k + \Delta y_k,$$

где $k = 0, 1, 2, \ldots, n$.

Геометрически эти формулы означают, что на отрезке $[x_i; x_{i+1}]$ интегральная кривая заменяется отрезком касательной к кривой (рис. 2.21, 2.22).

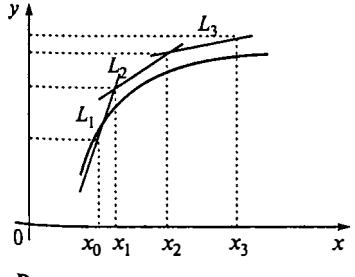



**Рис. 2.21.** Интегральная кривая

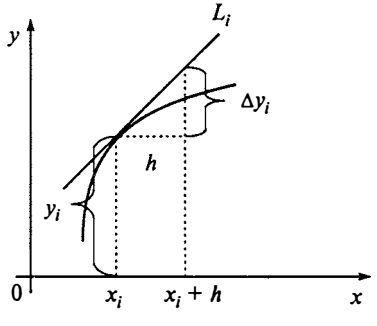



**Рис. 2.22.** Касательная к кривой

**Пример 2.56.** Проинтегрировать методом Эйлера дифференциальное уравнение

$$y' = y - x$$

с начальными условиями $x_0 = 0$; $y_0 = 1{,}5$ на отрезке $[0; 1{,}5]$ при $h = 0{,}25$.

***Решение***

| $i$ | $x_i$ | $y_i$ | $y'_i = y_i - x_i$ | $\Delta y_i = h_i y'_i$ |
|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** |
| 0 | 0 | 1,5 | 1,5 | 0,375 |
| 1 | 0,25 | 1,875 | 1,625 | 0,406 |
| 2 | 0,5 | 2,281 | 1,781 | 0,445 |
| 3 | 0,75 | 2,726 | 1,976 | 0,494 |
| 4 | 1 | 3,22 | 2,221 | 0,555 |
| 5 | 1,25 | 3,775 | 2,525 | 0,631 |
| 6 | 1,5 | 4,407 | | |

По начальным данным заполним первую строку в столбцах (2) и (3).

Из уравнения $y'_i = y_i - x_i$ вычисляем $y'_i$ ($i = 0,\ 1,\ 2,\ ...,\ 5$) в столбце (4).

К содержимому столбца (3) прибавляем содержимое столбца (5) этой же строки (вычисляем $y_{i+1} = y_i + \Delta y_i$), и результат записываем в столбец (3) следующей строки. Определяем $x_{i+1} = x_i + h$ и затем шаги повторяем до тех пор, пока не будет пройден весь отрезок.

**Пример 2.57.** Решить методом Эйлера дифференциальное уравнение $y' = \cos y + 3x$ с начальным условием $y(0) = 1{,}3$ на отрезке $[0; 1]$, приняв шаг $h = 0{,}2$.

***Решение***

Табличный способ

| $k$ | $x_k$ | $y_k$ | $\Delta y_k = 0{,}2(\cos y_k + 3x_k)$ |
|---|---|---|---|
| 0 | 0 | 1,3 | 0,05 |
| 1 | 0,2 | 1,35 | 0,16 |
| 2 | 0,4 | 1,52 | 0,25 |
| 3 | 0,6 | 1,77 | 0,32 |
| 4 | 0,8 | 2,09 | 0,38 |
| 5 | 1 | 2,47 | |

**Программа**

```
var x,y,a,b,h:real;     {Метод Эйлера}
    function f(x,y:real):real;
    begin f:= cos(y)+3*x;
    end;
begin
writeln('введите y, a, b, h');   readln(y,a,b,h);
x:=a;
repeat
  writeln(x:0:3,' ',y:0:3);
  y:=y+h*f(x,y);
  x:=x+h;
until x>b+0.1;
readln;
end.
```

Схема алгоритма показана на рис. 2.23.

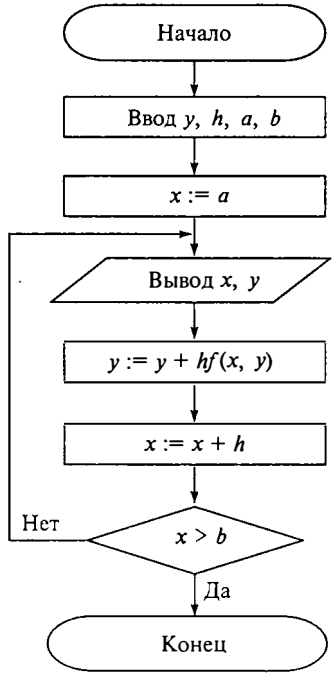



**Рис. 2.23.** Метод Эйлера

**Пример 2.58.** Методом Эйлера решить систему дифференциальных уравнений

$$\begin{cases} y' = (z - y)x; \\ z' = (z + y)x \end{cases}$$

с начальными уравнениями $y(0) = 1$; $z(0) = 1$ на отрезке $[0; 0{,}6]$, шаг интегрирования $h = 0{,}1$.

***Решение***

| $i$ | $x_i$ | $y_i$ | $y_i' = (z_i - y_i)x_i$ | $\Delta y_i = y_i' h$ | $z_i$ | $z_i' = (z_i + y_i)x_i$ | $\Delta z_i = z_i' h$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0,1 | 1 | 0 | 0 | 1 | 0,2 | 0,02 |
| 2 | 0,2 | 1 | 0,004 | 0,0004 | 1,02 | 0,404 | 0,0404 |
| 3 | 0,3 | 1,0004 | 0,018 | 0,0018 | 1,06 | 0,618 | 0,0618 |
| 4 | 0,4 | 1,0022 | 0,048 | 0,0048 | 1,12 | 0,849 | 0,085 |
| 5 | 0,5 | 1,007 | 0,1001 | 0,01 | 1,21 | 1,107 | 0,1107 |
| 6 | 0,6 | 1,017 | | | 1,317 | | |

### 2.6.1.2. Метод Рунге — Кутта второго порядка

Рассмотрим метод Рунге — Кутта второго порядка (метод Эйлера — Коши).

В этом методе величины $y_{i+1}$ вычисляются по следующим формулам (рис. 2.24):

$$y_{i+1} = y_i + \Delta y_i;\ \Delta y_i = \Delta y_{i1} + \Delta y_{i2},$$

$$\Delta y_{i1} = \frac{h}{2} f(x_i, y_i), \quad \Delta y_{i2} = \frac{h}{2} f(x_i + h, y_i + hf(x_i, y_i)).$$

Тогда

$$y_{i+1} = y_i + h\frac{f(x_i, y_i) + f(x_{i+1}, y_i + hf(x_i, y_i))}{2}.$$

Обозначим

$$y_{i+1}^* = y_1 + hf(x_i, y_i),$$

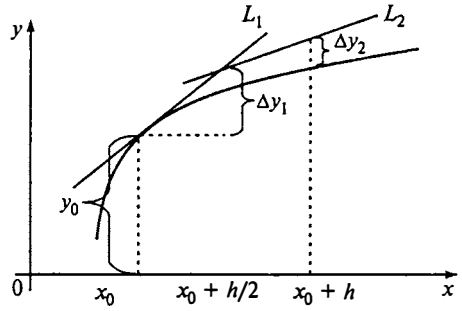



**Рис. 2.24.** Метод Эйлера — Коши

тогда

$$y_{i+1} = y_1 + h\frac{f(x_i, y_i) + f(x_{i+1}, y^*_{i+1}))}{2}.$$

Геометрически это означает, что определяется направление интегральной кривой в исходной точке $(x_i, y_i)$ и во вспомогательной точке $(x_{i+1}, y^*_{i+1})$, а в качестве окончательного выберем среднее из этих направлений.

**Пример 2.59.** Решить методом Эйлера дифференциальное уравнение $y' = \cos y + 3x$ методом Рунге — Кутта второго порядка (начальное условие $y(0) = 1{,}3$ на отрезке $[0; 1]$).

**Программа**

```
Program Rumge; {Метод Рунге — Кутта 2-го порядка}
Uses Crt;
var z,x,y,a,b,h:real;
      function f(x,y:real):real;
        begin
        f:= cos(y)+3*x;
        end;
begin
ClrScr;
writeln('Введите y, a, b, h');    readln(y,a,b,h);    x:=a;
repeat
  writeln(x:0:3,' ',y:0:3);
  z:=y+h*y*(1-x);
  y:=y+h*(f(x,y)+f(x+h,z))/2; x:=x+h;
until x>b+0.1;
readkey;
end.
```

### 2.6.1.3. Метод Рунге — Кутта 4-го порядка

В вычислительной практике наиболее часто используется метод Рунге — Кутта четвертого порядка. В этом методе величины $y_{i+1}$ вычисляются по следующим формулам:

$$y_{i+1} = y_i + \Delta y_i;$$

$$\Delta y_i = h(k_1 + 2k_2 + 2k_3 + k_4)/6, \text{ где } i = 0, 1, \ldots;$$

$$k_1 = f(x_i, y_i);$$

$$k_2 = f(x_i + h/2, y_i + hk_1/2);$$

$$k_3 = f(x_i + h/2, y_i + hk_2/2);$$

$$k_4 = f(x_i + h, y_i + hk_3).$$

**Пример 2.60.** Пусть дано дифференциальное уравнение $y' = y - x$ с начальным условием $y(0) = 1{,}5$.

Найти с точностью $\varepsilon = 0{,}01$ решение этого уравнения методом Рунге — Кутта при $x = 1{,}5$.

***Решение***

Пусть $h = 0{,}25$.

Весь отрезок интегрирования [0; 1,5] разобьем на шесть частей точками:

$x_0 = 0;$ $x_4 = 1;$

$x_1 = 0{,}25;$ $x_5 = 1{,}25;$

$x_2 = 0{,}5;$ $x_6 = 1{,}5.$

$x_3 = 0{,}75;$

Из начальных условий имеем $x_0 = 0;\ y_0 = 1{,}5$.

Найдем первое приближение

$$y_1 = y_0 + \Delta y_0, \text{ где } \Delta y_0 = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Используя формулы метода Рунге — Кутта, получаем

$$k_1 = (y_0 - x_0)h = 1{,}5 \cdot 0{,}25 = 0{,}375;$$

$$k_2 = [(y_0 + \frac{k_1}{2}) - (x_0 + \frac{h}{2})]h = [(1{,}5 + 0{,}187) - 0{,}125] \cdot 0{,}25 = 0{,}39;$$

$k_3 = [(y_0 + \frac{k_2}{2}) - (x_0 + \frac{h}{2})]h = 0{,}392;$

$k_4 = [(y_0 + k_3) - (x_0 + h)]h = 0{,}41.$

Следовательно $\Delta y_0 = \frac{1}{6}(0{,}375 + 2 \cdot 0{,}39 + 2 \cdot 0{,}392 + 0{,}41) = 0{,}392.$

Отсюда

$$y_1 = 1{,}5 + 0{,}392 = 1{,}892 \text{ и т. д.}$$

**Пример 2.61.** Проинтегрировать дифференциальное уравнение

$$\frac{dy}{dx} = x^2 + 2y^2 + 3e^4$$

методом Рунге — Кутта на отрезке [0; 2] и выдать на печать значение функции в каждой двадцатой точке. В качестве шага интегрирования можно взять $h = 0{,}005$, в качестве начальных условий — $y(0) = 1$.

**Программа**

**F(X,Y)** — подпрограмма-функция (правая часть уравнения)

```
Program  Dif;
Uses Crt;
Const    H= 0.005;
Var    X,Y:Real;
    F1,F2,F3,F4:real;
    I,N:Integer;
Begin
ClrScr;
    X:=0;   Y:=1;   N:=20;
    For I:=1 TO N DO
    Begin
      X:=X+H;
      F1:=H*F(X,Y);
      F2:=H*F(X+H/2.,Y+F1/2.);
      F3:=H*F(X+H/2.,Y+F2/2.);
      F4:=H*F(X+H,Y+F3);
      Y:=Y+(F1+2*F2+2*F3+F4)/6;
    End;
    Writeln('Y=', Y:8:3)
    Readkey;
    End.
```

*Замечание.* Метод Рунге—Кутта для решения дифференциального уравнения вида

$$\frac{dy}{dx} = F(x,\ y)$$

сводится к последовательному вычислению следующих равенств

$$x_{i+1} = x_i + h;$$

$$k_1 = hF(x_i,\ y_i);$$

$$k_2 = hF(x_i + \frac{h}{2},\ y_i + \frac{k_1}{2});$$

$$k_3 = hF(x_i + \frac{h}{2},\ y_i + \frac{k_2}{2});$$

$$k_4 = hF(x_i + h,\ y_i + k_3);$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),\ \text{где } i = 1,\ \ldots,\ n.$$

**Пример 2.62.** Решить дифференциальное уравнение

$$\frac{dy}{dx} = \cos y + 3x,$$

$y(0) = 1{,}3$ на отрезке $[0;\ 1]$ методом Рунге — Кутта 4-го порядка.

**Программа**

```
Program Runge;  {Метод Рунге - Кутта 4-го порядка}
var     k1,k2,k3,k4,x,y,a,b,h,d:real;
    function f(x,y:real):real;
    begin f:= cos(y)+3*x;
    end;
begin
writeln('Введите y, a, b, h');   readln(y,a,b,h);
x:=a;   .
repeat
    writeln(x:0:3,' ',y:0:3);
    k1:=f(x,y);
    k2:=f(x+h/2,y+h*k1/2);
    k3:=f(x+h/2,y+h*k2/2);
    k4:=f(x+h,y+h*k3);
    d:=(k1+2*k2+2*k3+k4)/6;
    y:=y+d; x:=x+h;
until x>b+0.1;
readln;
end.
```

## 2.7. Аппроксимация

**Аппроксимация**, или приближение, — замена одних математических объектов другими, в том или ином смысле близкими к исходным. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны). В теории чисел изучаются диофантовы приближения, в частности приближения иррациональных чисел рациональными. В геометрии рассматриваются аппроксимации кривых ломаными. Некоторые разделы математики в сущности целиком посвящены аппроксимации, например теория приближения функций, численные методы анализа.

***Аппроксимацией*** (приближением) функции $f(x)$ называется нахождение такой функции $g(x)$ (аппроксимирующей функции), которая была бы близка заданной. Критерии близости функций $f(x)$ и $g(x)$ могут быть различные.

В том случае, когда приближение строится на дискретном наборе точек, аппроксимацию называют точечной или ***дискретной***. В том случае, когда аппроксимация проводится на непрерывном множестве точек (отрезке), то аппроксимация называется непрерывной или ***интегральной***. Примером такой аппроксимации может служить разложение функции в ряд Тейлора, т. е. замена некоторой функции степенным многочленом.

Например:

- для приближенного вычисления интеграла используется формула прямоугольников или формула трапеций, или более сложная квадратурная формула. Фактически при этом происходит приближение подынтегральной функции ступенчатой функцией или вписанной ломаной;
- для вычисление значений сложных функций часто используется вычисление значения отрезка ряда, аппроксимирующего функцию (рис. 2.25).

**Программа**

```
program Kram;
uses CRT;
const  n=25;
type
```

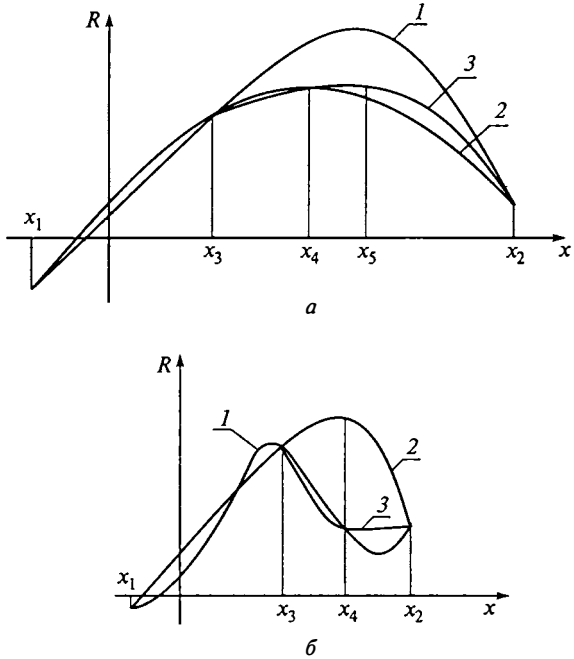



**Рис. 2.25.** Иллюстрация метода параболической аппроксимации: *а* — решение найти можно; *б* — решение найти нельзя; *1* — функция, экстремум которой ищется; *2* — аппроксимирующая парабола первого этапа, построенная по точкам $x_1$, $x_2$, $x_3$; *3* — аппроксимирующая парабола второго этапа, построенная по точкам $x_2$, $x_3$, $x_4$; $x_3$ — середина исходного интервала; $x_2$, $x_4$ — точки максимума первой параболы; $x_5$ — точка максимума второй параболы

```
TArrayXY = array[1..2,1..n] of real;
TArray = array[1..n] of real;
var
SumX,SumY,SumX2,SumXY,SumX3,SumX4,SumX2Y,SumLnY,
SumXLnY: real;
OPRlin,OPRkvadr,OPRa1,OPRa2,OPRa3:real;
a1lin,a2lin,a1kvadr,a2kvadr,a3kvadr,a1exp,a2exp,cexp:real;
Xsr,Ysr,S1,S2,S3,Slin,Skvadr,Sexp:real;
Kkor,KdetLin,KdetKvadr,KdetExp:real;
i:byte;
const
ArrayXY:TArrayXY=((12.85,12.32,11.43,10.59,10.21,9.65,9.63,9.22,8.44,
8.07, 7.74,7.32,7.08,6.87,5.23,5.02,4.65,4.53,3.24,2.55,1.86,1.76,1.11,
0.99,0.72),
```

```
(154.77,145.59,108.37,100.76,98.32,81.43,80.97,79.04,61.76,60.54,55.86,
47.63,48.03,36.85,25.65,24.98,22.87,20.32,9.06,6.23,3.91,3.22,1.22,1.10,
0.53));
begin
ClrScr;
SumX:=0.0; SumY:=0.0; SumXY:=0.0; SumX2:=0.0; SumX3:=0.0;
SumX4:=0.0; SumX2Y:=0.0; SumLnY:=0.0; SumXLnY:=0.0;
{ Вычисление сумм x, y, x*y, x^2, x^3, x^4, (x^2)*y, Ln(y), x*Ln(y) }
for i:=1 to n do
begin
  SumX:=SumX+ArrayXY[1,i];
  SumY:=SumY+ArrayXY[2,i];
  SumXY:=SumXY+ArrayXY[1,i]*ArrayXY[2,i];
  SumX2:=SumX2+sqr(ArrayXY[1,i]);
  SumX3:=SumX3+ArrayXY[1,i]*ArrayXY[1,i]*ArrayXY[1,i];
  SumX4:=SumX4+sqr(ArrayXY[1,i])*sqr(ArrayXY[1,i]);
  SumX2Y:=SumX2Y+sqr(ArrayXY[1,i])*ArrayXY[2,i];
  SumLnY:=SumLnY+ln(ArrayXY[2,i]);
  SumXLnY:=SumXLnY+ArrayXY[1,i]*ln(ArrayXY[2,i])
end;
{ Вычисление коэффициентов }
OPRlin:=0.0; a1lin:=0.0; a2lin:=0.0; a1kvadr:=0.0; a2exp:=0.0;
OPRkvadr:=0.0; a2kvadr:=0.0; a2kvadr:=0.0; a1exp:=0.0;
OPRlin:=n*SumX2-SumX*SumX;
a1lin:=(SumX2*SumY-SumX*SumXY)/OPRlin;
a2lin:=(n*SumXY-SumX*SumY)/OPRlin;
OPRkvadr:=n*SumX2*SumX4+SumX*SumX3*SumX2+SumX2*SumX*
SumX3-
SumX2*SumX2*SumX2-n*SumX3*SumX3-SumX*SumX*SumX4;
a1kvadr:=(SumY*SumX2*SumX4+SumX*SumX2Y*SumX3+SumX2*
SumXY*
SumX3-SumX2*SumX2*SumX2Y-SumY*SumX3*SumX3-
SumX*SumXY*SumX4)/OPRkvadr;
a2kvadr:=(n*SumXY*SumX4+SumY*SumX3*SumX2+SumX2*SumX*
SumX2Y-
SumX2*SumX2*SumXY-n*SumX3*SumX2Y-SumY*SumX*SumX4)/
OPRkvadr;
a3kvadr:=(n*SumX2*SumX2Y+SumX*SumXY*SumX2+SumY*SumX*
SumX3-
SumY*SumX2*SumX2-n*SumXY*SumX3-SumX*SumX*SumX2Y)/
OPrkvadr;
```

```
a2exp:=(n*SumXLnY-SumX*SumLnY)/OPRlin;
cexp:=(SumX2*SumLnY-SumX*SumXLnY)/OPRlin;
a1exp:=exp(cexp);
{ Вычисление средних арифметических x и y }
Xsr:=SumX/n;  Ysr:=SumY/n;
S1:=0.0;  S2:=0.0;  S3:=0.0;  Slin:=0.0;  Skvadr:=0.0;  Sexp:=0.0;
Kkor:=0.0;  KdetLin:=0.0;  KdetKvadr:=0.0;  KdetExp:=0.0;
for i:=1 to n do
begin
  S1:=S1+(ArrayXY[1,i]-Xsr)*(ArrayXY[2,i]-Ysr);
  S2:=S2+sqr(ArrayXY[1,i]-Xsr);
  S3:=S3+sqr(ArrayXY[2,i]-Ysr);
  Slin:=Slin+sqr(a1lin+a2lin*ArrayXY[1,i]-ArrayXY[2,i]);
  Skvadr:=Skvadr+sqr(a1kvadr+a2kvadr*ArrayXY[1,i]+a3kvadr*
  ArrayXY[1,i]*ArrayXY[1,i]-ArrayXY[2,i]);
  Sexp:=Sexp+sqr(a1exp*exp(a2exp*ArrayXY[1,i])-ArrayXY[2,i]);
end;
{Вычисление коэффициентов корреляции и детерминированности}
Kkor:=S1/sqrt(S2*S3);    KdetLin:=1-Slin/S3;
KdetKvadr:=1-Skvadr/S3;   KdetExp:=1-Sexp/S3;
{ Вывод результатов }
WriteLn('Линейная функция');  WriteLn('a1=',a1lin:8:5);
WriteLn('a2=',a2lin:8:5);
WriteLn('Квадратичная функция');  WriteLn('a1=',a1kvadr:8:5);
WriteLn('a2=',a2kvadr:8:5);  WriteLn('a3=',a3kvadr:8:5);
WriteLn('Экспоненциальная функция');
WriteLn('a1=',a1exp:8:5);  WriteLn('a2=',a2exp:8:5);
WriteLn('c=',cexp:8:5);  WriteLn('Xcp=',Xsr:8:5);
WriteLn('Ycp=',Ysr:8:5);
WriteLn('Коэффициент корреляции ',Kkor:8:5);
WriteLn('Коэф. детерминированности (линейная аппроксимация)',
  KdetLin:2:5);
WriteLn('Коэф. детерминированности (квадратическая
аппроксимация)',
  KdetKvadr:2:5);
WriteLn('Коэф. детерминированности (экспоненциальная
аппроксимация)',
  KdetExp:2:5);
Readln;
End.
```

### 2.7.1. Метод конечных элементов

Существуют различные формулировки метода конечных элементов, различающиеся как в основных, так и в менее значительных деталях. Рассмотрим основные этапы решения задачи этим методом.

Метод конечных элементов основывается на том, что любое непрерывное распределение физической переменной $u(x, y, z, t)$ в расчетной области, например деформацию или температурное поле, можно аппроксимировать набором кусочно-непрерывных функций, определенных на конечном числе подобластей (*конечных элементов*). Данные элементы имеют общие узловые точки и в совокупности аппроксимируют форму области.

В зависимости от геометрии и размерности задачи используют различные виды конечных элементов (рис. 2.26). Чаще всего применяются простейшие элементы — *симплексы*.

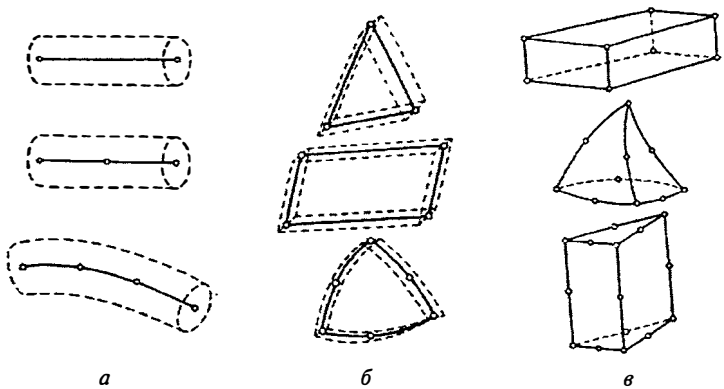
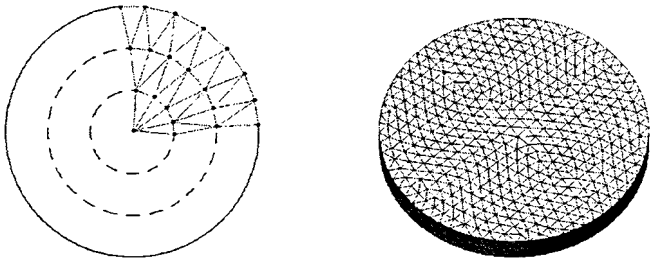



**Рис. 2.26.** Некоторые виды конечных элементов:
*а* — одномерные; *б* — двухмерные; *в* — трехмерные

Количество узлов в симплексе на единицу превышает размерность задачи. Для двухмерной задачи симплекс-элементом будет являться прямолинейный трехузловой треугольник, а для трехмерных — прямолинейный четырехузловой тетраэдр. Широкое применение симплексов обусловлено тем, что они позволяют заполнять расчетную область произвольной формы полностью без разрывов, а также на них удобно использовать линейные полиномы в качестве аппроксимирующих функций.

Обычно для разбиения расчетной области на элементы используется специальный *алгоритм покрытия*, обеспечивающий автоматическую генерацию сетки. Одна из таких процедур работает следующим образом (рис. 2.27, *а*). Вначале производится нанесение с некоторым шагом узлов на границу области. После этого внутри области строится вспомогательная кривая, эквидистантная границе. На кривую также наносятся узлы. Поочередное соединение узлов на первом и втором контурах дает симплексы. Далее все операции повторяются до заполнения симплексами всей области.



*а* *б*

**Рис. 2.27.** Построение сетки конечных элементов

Известны и другие алгоритмы формирования конечных элементов, например, «картографический», использующий наложение на расчетную область сетки, которая затем адаптируется к границам и неоднородностям геометрии, или методы, основанные на заполнении объекта набором фигур (тел) с использованием свойств симметрии или отражения.

Пример автоматически сгенерированной трехмерной сетки для круглого диска показан на рис. 2.27, *б*.

#### 2.7.1.1. Конечно-элементная аппроксимация

Рассмотрим построение аппроксимации на одномерном примере. Пусть требуется найти распределение некоторой непрерывной функции $u(x)$ вдоль стержня (рис. 2.28, *а*). На практике эта функция может описывать, например, распределение температуры или деформацию стержня.

Выберем и пронумеруем ряд точек вдоль оси $x$ — это узловые точки (рис. 2.28, *б*). В данном примере взято всего пять точек.

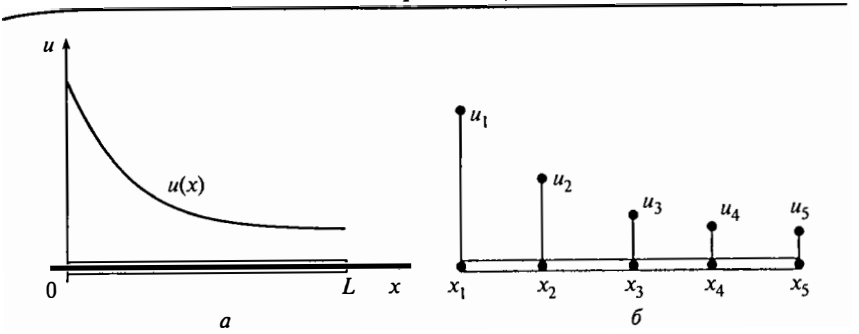



**Рис. 2.28.** Одномерное распределение

Вообще говоря, их может быть произвольное количество, и располагаться они могут не на равном расстоянии друг от друга. Предположим, что значения $u(x)$ в узловых точках известны. Они обозначены на рис. 2.28, *б* в соответствии с номерами узлов — $u_1$, $u_2$, $u_3$, $u_4$, $u_5$.

Разбиение расчетной области, т. е. стержня, на конечные элементы может быть проведено различными способами. Можно, например, выделить четыре элемента, включив в каждый из них по два соседних узла (рис. 2.29, *а*). А можно выделить в стержне всего два элемента, содержащие по три узла (рис. 2.29, *б*).

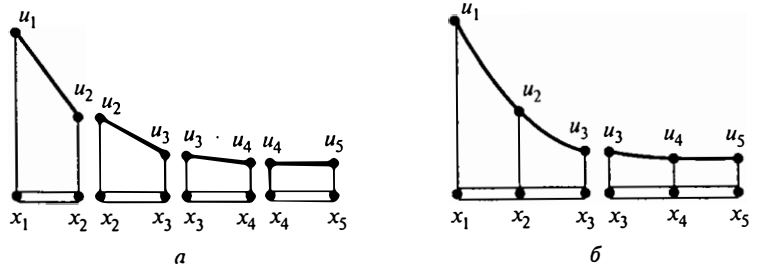



**Рис. 2.29.** Варианты разбиения стержня на элементы

При использовании четырех элементов, каждый из которых включает только два узла, аппроксимирующая функция в пределах элемента будет линейна по $x$, так как две точки однозначно определяют прямую линию. Общая аппроксимация зависимости $u(x)$ по всей длине стержня будет складываться из четырех отрезков прямых (рис. 2.29, *а*).

Зависимость $u(x)$ в пределах одного элемента, ограниченного двумя соседними узлами $x_i$ и $x_j$ ($j = i + 1$), можно представить ли-

нейным интерполяционным полиномом $u(x) \approx \alpha + \alpha_x x$. Определив параметры $\alpha$ и $\alpha_x$ по известным в точках $x_i$ и $x_j$ значениям функции $u_i$ и $u_j$, запишем интерполяционный полином, т. е. *функцию элемента* следующим образом:

$$u(x) \approx \frac{x_j - x}{x_j - x_i} u_i + \frac{x - x_i}{x_j - x_i} u_j = N_i u_i + N_j u_j = [N^{(e)}][u^{(e)}],$$

где $N_i$ и $N_j$ — так называемые *функции формы* конечного элемента; $u_i$ и $u_j$ — значения функции $u(x)$ в точках $x_i$ и $x_j$; $[N^{(e)}] = [N_i N_j]$ — матричная строка функций формы элемента; $[u^{(e)}] = \begin{bmatrix} u_i \\ u_j \end{bmatrix}$ — вектор-столбец.

Здесь следует отметить, что ряд терминов метода конечных элементов получил название из механики, где он впервые начал активно использоваться.

К достоинствам метода конечных элементов, благодаря которым он находит широкое применение, относят гибкость и разнообразие сеток, четко формализованные алгоритмы построения дискретных задач для произвольных областей, простоту учета естественных краевых условий. Кроме того, этот метод применим к широкому классу исходных задач, а оценки погрешностей приближенных решений, как правило, получаются при менее жестких ограничениях, чем в методе конечных разностей.

## 2.8. Интерполяция и экстраполяция

**Интерполяция** — способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

В научных и инженерных расчетах часто приходится оперировать наборами значений, полученных экспериментальным путем или методом случайной выборки. Как правило, на основании этих наборов требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения. Например, известны некоторые значения функции — фи-

зической величины, замеренные через 1 ч. Необходимо найти значения в промежутках через 30 мин.

Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит точно через имеющиеся точки данных.

Существует также близкая к интерполяции задача, которая заключается в аппроксимации какой-либо сложной функции другой, более простой функцией. Если некоторая функция слишком сложна для производительных вычислений, можно попытаться вычислить ее значение в нескольких точках, а по ним построить, т. е. интерполировать, более простую функцию. Разумеется, использование упрощенной функции не позволяет получить такие же точные результаты, какие давала бы первоначальная функция, но в некоторых классах задач достигнутый выигрыш в простоте и скорости вычислений может перевесить получаемую погрешность в результатах.

Наиболее часто встречающимся видом точечной аппроксимации является *интерполяция*. Пусть задан дискретный набор точек $x_i$ ($i = 0, 1, ..., n$), называемых *узлами интерполяции*, причем среди этих точек нет совпадающих, а также значения функции $y_i$ в этих точках. Требуется построить функцию $g(x)$, проходящую через все заданные узлы. Таким образом, критерием близости функции является $g(x_i) = y_i$. В качестве функции $g(x)$ обычно выбирается полином, который называют *интерполяционным полиномом*. В том случае, если полином един для всей области интерполяции, говорят, что интерполяция ***глобальная***.

В тех случаях, когда между различными узлами полиномы различны, говорят о ***кусочной*** или ***локальной интерполяции***. Найдя интерполяционный полином, можно вычислить значения функции $f(x)$ между узлами (провести *интерполяцию в узком смысле слова*), а также определить значение функции $f(x)$ даже за пределами заданного интервала (провести ***экстраполяцию***).

Пусть имеется $n$ значений $x_i$, каждому из которых соответствует свое значение $y_i$. Требуется найти такую функцию $F$, что:

$$F(x_i) = y_i, \quad i = 0, 1, ..., n.$$

При этом:

- $x_i$ называют *узлами интерполяции*;
- пары $(x_i, y_i)$ называют *точками данных*;

- разницу между соседними значениями $(x_i - x_{i-1})$ называют *шагом*;
- функцию $F(x)$ — *интерполирующей функцией* или *интерполянтом*.

Задача интерполирования состоит в том, чтобы по значениям функции в некоторых точках восстановить ее значения в остальных точках отрезка. Функция $F$ называется интерполирующей, точки $x_0, x_1, x_2, \ldots, x_n$ — узлами интерполяции.

Будем искать функцию $F$ в виде многочлена степени $n$:

$$F(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n.$$

Можно найти коэффициенты $a_i$, $i = 0, 1, \ldots, n$, при этом получим систему из $(n + 1)$ уравнения с $(n + 1)$ неизвестными

$$\begin{cases} a_0x_0^n + a_1x_0^{n-1} + \ldots + a_{n-1}x_0 + a_n = y_0; \\ a_0x_1^n + a_1x_1^{n-1} + \ldots + a_{n-1}x_1 + a_n = y_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0x_n^n + a_1x_n^{n-1} + \ldots + a_{n-1}x_n + a_n = y_n. \end{cases}$$

Эта система имеет единственное решение, так как по нашему предположению все $x_i$ различны. Решая эту систему относительно неизвестных $a_0, a_1, \ldots, a_n$, получим аналитическое выражение многочлена.

Описанный прием можно использовать при решении задач интерполирования, но на практике используют другие более удобные и менее трудоемкие методы.

### 2.8.1. Интерполяционный многочлен Лагранжа

Пусть функция $y = f(x)$ задана таблицей. Построим интерполяционный многочлен $L_n(x)$, степень которого не больше $n$ и выполняются условия: $L_n(x_i) = y_i$, $i = 0, 1, \ldots, n$. Будем искать $L_n(x)$ в виде

$$L_n(x) = p_0(x)y_0 + p_1(x)y_1 + \ldots + p_n(x)y_n = \sum_{i=1}^{n} p_i(x)y_i,$$

где $p_i(x)$ — многочлен степени $n$;

$p_i(x_j) = \begin{cases} 1, & i = j; \\ 0, & i \neq j, \end{cases}$ т. е. $p_i(x)$ только в одной точке отличен от нуля при $i = j$, а в остальных точках он обращается в нуль. Следовательно, все эти точки являются для него корнями:

$$p_i(x) = c(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n);$$

при $x = x_i$

$$p_i(x_i) = c(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n);$$

$$c = [(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)]^{-1};$$

подставим $c$ в формулу $p_i(x)$, получим:

$$p_i(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)},$$

отсюда

$$L_n(x) = \sum_{i=0}^{n} \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)} y_i .$$

Это и есть интерполяционный многочлен Лагранжа. По исходной таблице формула позволяет весьма просто составить внешний вид многочлена.

**Пример 2.63.** Построить интерполяционный многочлен Лагранжа для функции, заданной таблично

| $X$ | 1 | 2 | 3 | 5 |
|---|---|---|---|---|
| $Y$ | 1 | 5 | 14 | 81 |

***Решение***

Степень $L_n(x)$ не выше третьей, так как функция задается четырьмя значениями:

$$L_3(x) = 1 \cdot \frac{(x-2)(x-3)(x-5)}{(1-2)(1-3)(1-5)} + 5 \cdot \frac{(x-1)(x-3)(x-5)}{(2-1)(2-3)(2-5)} +$$

$$+ 14 \cdot \frac{(x-1)(x-2)(x-5)}{(3-1)(3-2)(3-5)} + 81 \cdot \frac{(x-1)(x-2)(x-3)}{(5-1)(5-2)(5-3)} =$$

$$= x^3 - 2x^2 + 3x - 1.$$

**Пример 2.64.** Для функции $y = \sin(\pi x)$ построить интерполяционный полином Лагранжа, выбрав узлы $x_0 = 0;\ x_1 = \frac{1}{6};\ x_2 = \frac{1}{2}$.

***Решение***

Вычислим соответствующие значения функции:

$$y_0 = 0; \quad y_1 = \sin\frac{\pi}{6}; \quad y_2 = \sin\frac{\pi}{2} = 1.$$

Применяя формулу, получаем

$$L_2(x) = \frac{\left(x-\frac{1}{6}\right)\left(x-\frac{1}{2}\right)\cdot 0}{\left(-\frac{1}{6}\right)\left(-\frac{1}{2}\right)} + \frac{1}{2}\cdot\frac{x\left(x-\frac{1}{2}\right)}{\frac{1}{6}\left(\frac{1}{6}-\frac{1}{2}\right)} +$$

$$+ 1\cdot\frac{x\left(x-\frac{1}{6}\right)}{\frac{1}{2}\left(\frac{1}{2}-\frac{1}{6}\right)} = \frac{7}{2}x - 3x^2.$$

**Пример 2.65.** Построить интерполяционный полином Лагранжа для двух узлов интерполяции:

| $x$ | $x_0$ | $x_1$ |
|---|---|---|
| $y$ | $y_0$ | $y_1$ |

| $x$ | 1 | 3 |
|---|---|---|
| $y$ | 1 | 9 |

***Решение***

$N = 0,\ 1$ (два узла интерполяции).

$$L_n(x) = \frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1$$

— уравнение прямой, проходящей через точки $(x_0, y_0)$, $(x_1, y_1)$;

$$L_1(x) = \frac{x-3}{1-3}\cdot 1 + \frac{x-1}{3-1}\cdot 9 = \frac{x-3}{-2} + \frac{x-1}{2}\cdot 9 = \frac{8x-6}{2} = 4x - 3.$$

**Пример 2.66.** Построить интерполяционный полином Лагранжа для трех узлов интерполяции:

| $x$ | $x_0$ | $x_1$ | $x_2$ |
|---|---|---|---|
| $y$ | $y_0$ | $y_1$ | $y_2$ |

| $x$ | 1 | 3 | 4 |
|---|---|---|---|
| $y$ | 12 | 4 | 6 |

***Решение***

$N = 0, 1, 2$ (три узла интерполяции).

$$L_n(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 +$$

$$+ \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2$$

— уравнение параболы, проходящей через точки $(x_0, y_0)$, $(x_1, y_1)$, $(x_2, y_2)$;

$$L_n(x) = \frac{(x - 3)(x - 4)}{(1 - 3)(1 - 4)} \cdot 12 + \frac{(x - 1)(x - 4)}{(3 - 1)(3 - 4)} \cdot 4 +$$

$$+ \frac{(x - 1)(x - 3)}{(4 - 1)(4 - 3)} \cdot 6 = 2x^2 - 12x + 22.$$

Построим график этой функции (рис. 2.30) и отметим на нем узловые точки $M_i(x_i, y_i)$.

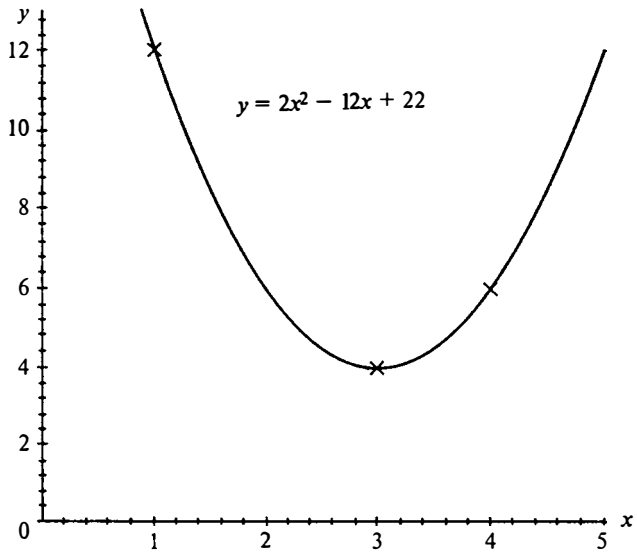



**Рис 2.30.** График полученной функции (к примеру 2.66)

**Пример 2.67.** Найти приближенное значение функции $f(\text{arg})$ при данном значении аргумента arg с помощью интерполяционного многочлена Лагранжа, если функция задана таблицей

| $x$ | $x_0$ | $x_1$ | ... | $x_n$ |
|---|---|---|---|---|
| $y$ | $y_0$ | $y_1$ | ... | $y_n$ |
| arg = ... | | | | |

**Программа**

```
program Lagrange;
uses Crt;
var    X, Y : array[1..100] of Real;
                  Arg, L, F : Real;
                  I, J, N :Integer;
begin
Write('Введите количество узлов интерполяции');
Readln(n);
WriteLn('Введите таблицу значений xi, yi');
for I:=0 to N do begin
    Write('X[',I,']='); ReadLn(X[I]);
    Write('Y[',I,']='); ReadLn(Y[I]);
    end;
Write('Введите аргумент '); ReadLn(Arg);
L:=0;
For I:=0 to N do begin
  F:=1;
  for J:=0 to N do
  if I<>J then F:=F*(Arg-X[J])/(X[I]-X[J]);
  F:=F*Y[I]; L:=L+F;
  end;
WriteLn('Значение многочлена Лагранжа в точке ',Arg:0:3);
WriteLn('равно ', L:0:3);
ReadLn;
end.
```

## 2.8.2. Использование электронных таблиц

Напомним формулу интерполяционного многочлена Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)} y_i.$$

Введем обозначения:

$$\Pi_{n+1}(x) = (x - x_0)(x - x_1)\ldots(x - x_n);$$

$$D_i = (x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n),$$

тогда

$$L_n(x) = \Pi_{n+1}(x) \sum_{i=1}^{n} \frac{y_i}{D_i}.$$

По этой формуле удобно вычислять многочлен Лагранжа $L_n(x)$ в таблице.

**Пример 2.68.** Пусть требуется вычислить значение многочлена Лагранжа $L_n(x)$ в точке $x$.

***Решение***

Составим следующую таблицу

| $i$ | $x_i$ | Разности | | | | | | $y_i$ | $D_i$ | $y_i/D_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | $x_0$ | $(x - x_0)$ | $(x_0 - x_1)$ | $(x_0 - x_2)$ | $(x_0 - x_3)$ | $(x_0 - x_4)$ | $(x_0 - x_5)$ | $y_0$ | | |
| 1 | $x_1$ | $(x_1 - x_0)$ | $(x - x_1)$ | $(x_1 - x_2)$ | $(x_1 - x_3)$ | $(x_1 - x_4)$ | $(x_1 - x_5)$ | $y_1$ | | |
| 2 | $x_2$ | $(x_2 - x_0)$ | $(x_2 - x_1)$ | $(x - x_2)$ | $(x_2 - x_3)$ | $(x_2 - x_4)$ | $(x_2 - x_5)$ | $y_2$ | | |
| 3 | $x_3$ | $(x_3 - x_0)$ | $(x_3 - x_1)$ | $(x_3 - x_2)$ | $(x - x_3)$ | $(x_3 - x_4)$ | $(x_3 - x_5)$ | $y_3$ | | |
| 4 | $x_4$ | $(x_4 - x_0)$ | $(x_4 - x_1)$ | $(x_4 - x_2)$ | $(x_4 - x_3)$ | $(x - x_4)$ | $(x_4 - x_5)$ | $y_4$ | | |
| 5 | $x_5$ | $(x_5 - x_0)$ | $(x_5 - x_1)$ | $(x_5 - x_2)$ | $(x_5 - x_3)$ | $(x_5 - x_4)$ | $(x - x_5)$ | $y_5$ | | |

Далее необходимо вычислить

$$\Pi_{5+1}(x) = (x - x_0)(x - x_1)(x - x_2)(x - x_3)(x - x_4)(x - x_5)$$

и сумму последнего столбца $\sum_{i=0}^{5} \frac{y_i}{D_i}$. Тогда получаем

$$L_5(x) = \Pi_{5+1}(x) \sum_{i=0}^{5} \frac{y_i}{D_i}.$$

**Пример 2.69.** Найти приближенное значение функции при данном значении аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана своей таблицей. Вычислить значение функции $F(x)$ при $x = 0{,}263$.

| $x$ | 0,05 | 0,10 | 0,17 | 0,25 | 0,30 | 0,36 |
|---|---|---|---|---|---|---|
| $y$ | 0,050042 | 0,100335 | 0,171657 | 0,255342 | 0,309336 | 0,376403 |

Воспользовавшись формулой интерполяционного многочлена Лагранжа, составим таблицу разностей, где запись $mE-b$ означает $m \cdot 10^{-b}$.

| $x = 0{,}263$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | $x_i$ | Разности | | | | | | $y_i$ | $D_i$ | $y_i/D_i$ |
| 0 | 0,05 | 0,213 | −0,05 | −0,12 | −0,2 | −0,25 | −0,31 | 0,050042 | −2E-05 | −2526,23 |
| 1 | 0,10 | 0,05 | 0,163 | −0,07 | −0,15 | −0,2 | −0,26 | 0,100335 | 4,45E-06 | 22547,7 |
| 2 | 0,17 | 0,12 | 0,07 | 0,093 | −0,08 | −0,13 | −0,19 | 0,171657 | −1,5E-06 | −111202 |
| 3 | 0,25 | 0,2 | 0,15 | 0,08 | 0,013 | −0,05 | −0,11 | 0,255342 | 1,72E-07 | 1488007 |
| 4 | 0,30 | 0,25 | 0,2 | 0,13 | 0,05 | −0,037 | −0,06 | 0,309336 | 7,21E-07 | 428740,1 |
| 5 | 0,36 | 0,31 | 0,26 | 0,19 | 0,11 | 0,06 | −0,097 | 0,376403 | −9,8E-06 | −38392,7 |

Вычисляем:

$$\Pi_{5+1}(0{,}263) = (0{,}263 - x_0)(0{,}263 - x_1)(0{,}263 - x_2)(0{,}263 - x_3) \times$$
$$\times (0{,}263 - x_4)(0{,}263 - x_5) = 0{,}1506492 \cdot 10^{-6},$$

сумма последнего столбца отсюда

$$F(0{,}263) = \Pi_{5+1}(0{,}263) \sum_{i=0}^{5} \frac{y_i}{D_i} =$$

$$= 0{,}1506492 \cdot 10^{-6} \cdot 1790173{,}8 = 0{,}269678.$$

Вычисления вручную довольно громоздки, но решение можно получить с помощью электронной таблицы.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,263 | | | | | | | | | | |
| 2 | $i$ | $x_i$ | Разности | | | | | | $y_i$ | $D_i$ | $y_i/D_i$ |
| 3 | 0 | 0,05 | =$A$1-$B3 | =B3-$B$4 | =B3-$B$5 | =B3-$B$6 | =B3-$B$7 | =B3-$B$8 | 0,050042 | =ПРОИЗВЕД(C3:H3) | =I3/J3 |
| 4 | 1 | 0,1 | =B4-$B$3 | | | | | | 0,100335 | | |
| 5 | 2 | 0,17 | | | | | | | 0,171657 | | |
| 6 | 3 | 0,25 | | | | | | | 0,255342 | | |
| 7 | 4 | 0,3 | | | | | | | 0,309336 | | |
| 8 | 5 | 0,36 | | | | | | | 0,376403 | | |
| 9 | | | | | | | | | =ПРОИЗВЕД(C3,D4,E5,F6,G7,H8) | | СУММ(K3:K8) |
| 10 | | | | | | | | | =I9*K9 | | |

Заполняем таблицу по образцу. Затем копируем ячейку C4 в C5:C8, ячейку D3 — в D5:D8, ячейку E3 — в E4, E6:E8, ячейку F3 — в F4, F5, F7,F8, ячейку G3 — в G4:G6, G8, ячейку H3 — в H4:H7, ячейку C3 — в D4, E5, F6, G7, H8, ячейку J3 — в J4:J8, ячейку K3 — в K4:K8.

В результате вычислений в ячейке I10 получаем значение многочлена Лагранжа.

В математике **экстраполяция** обозначает особый тип аппроксимации, при котором функция аппроксимируется не *между* заданными значениями, а *вне* заданного интервала.

## 2.9. Численное интегрирование

Многие инженерные задачи, задачи физики, геометрии и многих других областей человеческой деятельности приводят к необходимости вычислять определенный интеграл вида

$$\int_a^b f(x)dx, \qquad (2.13),$$

где $f(x)$ — подынтегральная функция, непрерывная на отрезке $[a; b]$.

Пусть задана подынтегральная функция $f(x)$, необходимо найти определенный интеграл, который вычисляется по формуле Ньютона — Лейбница

$$\int_a^b f(x)dx = F(b) - F(a). \qquad (2.14)$$

Если же интеграл от данной функции не может быть вычислен по формуле (2.14), или если функция $f(x)$ задана графически или таблицей, то для вычисления определенного интеграла применяют приближенные формулы. Для приближенного вычисления интеграла (2.14) существует много численных методов, таких как:

- метод прямоугольников;
- трапеций;
- Симпсона и др.

При вычислении интеграла следует помнить, каков геометрический смысл определенного интеграла.

Если $f(x) \geq 0$ на отрезке $[a; b]$, то $\int_a^b f(x)dx$ численно равен площади фигуры, ограниченной графиком функции $y = f(x)$, отрезком оси абсцисс, прямой $x = a$ и прямой $x = b$ (рис. 2.31). Таким образом, вычисление интеграла равносильно вычислению площади криволинейной трапеции.

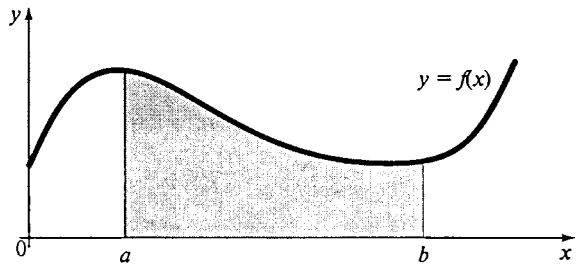



**Рис. 2.31.** Геометрический смысл интеграла

### 2.9.1. Метод прямоугольников

Разделим отрезок $[a; b]$ на $n$ равных частей, т. е. на $n$ элементарных отрезков. Длина каждого элементарного отрезка $h = \frac{b-a}{n}$. Точками деления будут: $x_0 = a$; $x_1 = a + h$; $x_2 = a + 2h$, ...,

$x_{n-1} = a + (n-1)h$; $x_n = b$. Эти числа будем называть узлами. Вычислим значения функции $f(x)$ в узлах, обозначим их $y_0$, $y_1$, $y_2$, ..., $y_n$. Стало быть, $y_0 = f(a)$, $y_1 = f(x_1)$, $y_2 = f(x_2)$, ..., $y_n = f(b)$. Числа $y_0$, $y_1$, $y_2$, ..., $y_n$ являются ординатами точек графика функции, соответствующих абсциссам $x_0$, $x_1$, $x_2$, ..., $x_n$ (рис. 2.31).

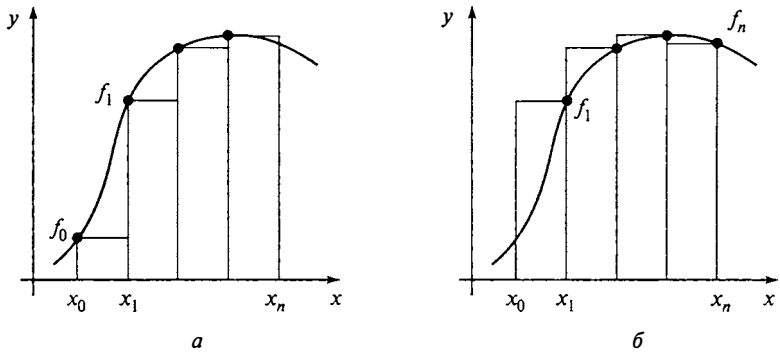



**Рис. 2.32.** Методы левых (*а*) и правых (*б*) прямоугольников

Из рис. 2.32 видно, что площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из $n$ прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы $n$ элементарных прямоугольников

$$S \approx \int_a^b f(x)dx \approx y_0 h + y_1 h + y_2 h + y_3 h + \ldots + y_{n-1} h \approx$$

$$\approx h(y_0 + y_1 + y_2 + y_3 + \ldots + y_{n-1}); \qquad (2.15)$$

$$S \approx \int_a^b f(x)dx \approx y_1 h + y_2 h + y_3 h + y_4 h + \ldots + y_n h \approx$$

$$\approx h(y_1 + y_2 + y_3 + y_4 + \ldots + y_n). \qquad (2.16)$$

Формула (2.15) называется формулой левых прямоугольников, (2.16) — правых прямоугольников, (2.17) — формулой средних прямоугольников (рис. 2.33).

$$S \approx h\sum_{i=0}^{n-1} y(x_i + \frac{h}{2}). \qquad (2.17)$$

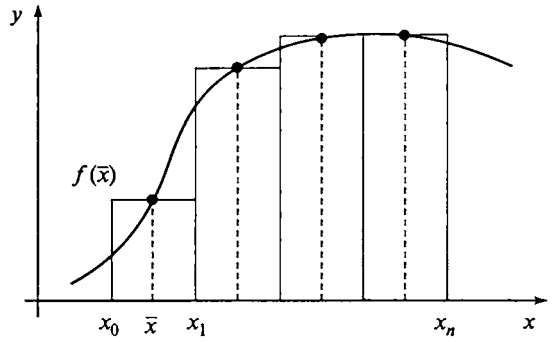



**Рис. 2.33.** Метод средних прямоугольников

Алгоритм вычисления интеграла по формуле левых прямоугольников показан на рис. 2.34.

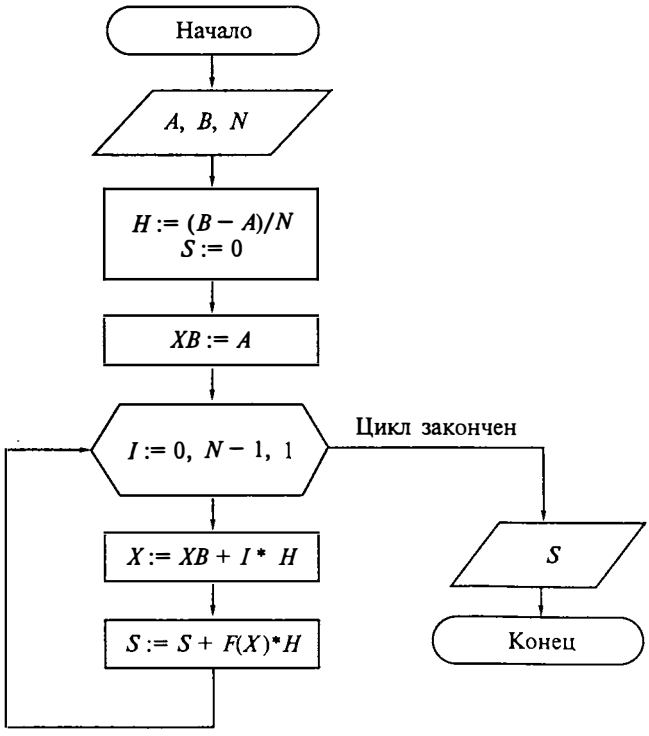



**Рис. 2.34.** Схема алгоритма вычисления интеграла

**Пример 2.70.** С помощью метода левых и правых прямоугольников вычислить определенный интеграл

$$\int_1^9 \frac{dx}{x+2}, \text{ полагая } n = 4.$$

***Решение***

Зная пределы интегрирования, $a = 1$ и $b = 9$, находим шаг

$$h = \frac{b-a}{n} = \frac{9-1}{4} = 2.$$

Тогда точками разбиения будут

$$x_0 = 1;\ x_1 = 3;\ x_2 = 5;\ x_3 = 7;\ x_4 = 9.$$

Значения подынтегральной функции $f(x) = \frac{1}{x+2}$ в этих точках таковы:

$$y_0 = f(x_0) = \frac{1}{3};\ y_1 = f(x_1) = \frac{1}{5};\ y_2 = \frac{1}{7};\ y_3 = \frac{1}{9};\ y_4 = \frac{1}{11}.$$

Найдем численное значение интеграла по формуле левых прямоугольников:

$$I_l = h(y_0 + y_1 + y_2 + y_3) = 2\left(\frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9}\right) \approx 1{,}6024.$$

По формуле правых прямоугольников:

$$I_n = h(y_1 + y_2 + y_3 + y_4) = 2\left(\frac{1}{5} + \frac{1}{7} + \frac{1}{9} + \frac{1}{11}\right) \approx 1{,}1053.$$

**Пример 2.71.** Вычислить по методу левых прямоугольников:

$$\int_1^2 \frac{1}{x}\sin\left(\pi\frac{x}{2}\right).$$

**Программа**

```
program integral1;{Метод левых прямоугольников}
uses crt;
var  i,n:integer;
       a,b,h,x,xb,s:real;
         function f(x:real):real;
         begin
         f:=(1/x)*sin(Pi*x/2);
         end;
```

```
begin
clrscr;  gotoxy(10,10);  textbackground(1); clrscr;
write('Введите нижний предел интегрирования ');
readln(a);  gotoxy(10,12);
write('Введите верхний предел интегрирования ');
readln(b);  gotoxy(10,14);
write('Введите количество отрезков ');  readln(n);
h:=(b-a)/n;  s:=0;  xb:=a;
for  i:=0  to n-1 do
    begin  x:=xb+i*h;  s:=s+f(x)*h;  end;
gotoxy(10,18);
writeln('Интеграл равен ',s:12:10);
readkey;
end.
```

**Пример 2.72.** Вычислить по методу правых прямоугольников:

$$\int_1^2 \frac{1}{x}\sin\left(\pi\frac{x}{2}\right).$$

**Программа**

```
program integral; {Метод правых прямоугольников}
uses crt;
var  i,n:integer;
        a,b,h,x,xb,s:real;
        function f(x:real):real;
        begin
        f:=(1/x)*sin(3.14*x/2);
        end;
begin
clrscr;  gotoxy(10,10);  textbackground(1); clrscr;
write('Введите нижний предел интегрирования ');
readln(a);  gotoxy(10,12);
write('Введите верхний предел интегрирования ');
readln(b);  gotoxy(10,14);
write('Введите количество отрезков '); readln(n);
h:=(b-a)/n;  s:=0;  xb:=a;
for i:=1 to n do
    begin  x:=xb+i*h;  s:=s+f(x)*h;  end;
gotoxy(10,18);
writeln('Интеграл равен ',s:12:10);
readkey;
end.
```

**Задача 2.73.** Вычислить по методу средних прямоугольников:

$$\int_1^2 \frac{1}{x}\sin\left(\pi\frac{x}{2}\right).$$

**Программа**

```
program integral; {Метод средних прямоугольников}
uses crt;
var  i, n: integer;
       a, b, dx, x, s, xb : real;
       function f(x : real):real;
       begin
       f:=(1/x)*sin(3.14*x/2);
       end;
begin
clrscr;  gotoxy(10,10);  textbackground(1); clrscr;
write('Введите нижний предел интегрирования ');
readln(a);  gotoxy(10,12);
write('Введите верхний предел интегрирования ');
readln(b);  gotoxy(10,14);
write('Введите количество отрезков '); readln(n);
dx:=(b-a)/n;  xb:=a+dx/2;  s:=0;
for i:=0 to n-1 do
    begin  x:=xb+i*dx;  s:=s+f(x)*dx;  end;
gotoxy(10,18);
write('Интеграл равен ',s:15:10);
readkey;
end.
```

### 2.9.2. Метод трапеций

Формула трапеций имеет следующий вид:

$$S \approx \int_a^b f(x)dx \approx h\left(\frac{y_0 + y_n}{2} + y_1 + y_2 + \ldots + y_{n-1}\right). \qquad (2.19)$$

Эта формула означает, что площадь криволинейной трапеции заменяется площадью многоугольника, составленного из $n$ трапеций (рис. 2.35), при этом кривая заменяется вписанной в нее ломаной.

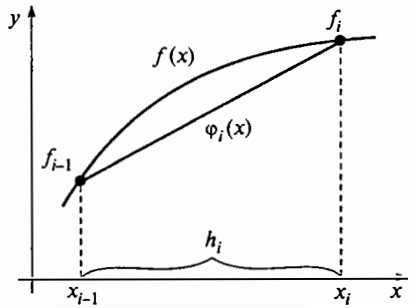



**Рис. 2.35.** Метод трапеций

**Пример 2.74.** Пользуясь формулой трапеций, вычислить определенный интеграл

$$\int_0^1 \frac{dx}{1+x^2} \text{ при } n = 4.$$

***Решение***

Подынтегральная функция на отрезке [0; 1] равна $F(x) = \dfrac{1}{1+x^2}$.

Находим шаг вычислений

$$h = \frac{b-a}{n} = 0{,}25.$$

Отсюда точки интегрирования:

$$x_0 = 0; \quad x_1 = 0{,}25; \quad x_2 = 0{,}5; \quad x_3 = 0{,}75; \quad x_4 = 1.$$

Тогда по формуле трапеций имеем

$$\int_0^1 \frac{dx}{1+x^2} = \frac{0{,}25}{2}\left(1 + \frac{2}{1+0{,}25^2} + \frac{2}{1+0{,}5^2} + \frac{2}{1+0{,}75^2} + \frac{1}{2}\right) \approx 0{,}764.$$

*Ответ*: $\displaystyle\int_0^1 \frac{dx}{1+x^2} = 0{,}764.$

**Пример 2.75.** Методом трапеций вычислить определенный интеграл

$$\int_a^b \frac{\sin(\pi x/2)}{x}\,dx.$$

**Программа**

```
program integral; {метод трапеций}
uses crt;
var   i,n:integer;
      a,b,h,x,s:real;
      function f(x:real):real;
      begin
      f:=(1/x)*sin(3.14*x/2);
end;
begin
clrscr; gotoxy(10,10); textbackground(1); clrscr;
write('Введите нижний предел интегрирования');
readln(a); gotoxy(10,12);
write('Введите верхний предел интегрирования ');
readln(b); gotoxy(10,14);
write('Введите количество отрезков '); readln(n);
h:=(b-a)/n;  s:=0;  x:=a;
for i:=1 to n-1 do
begin x:=x+h; s:=s+f(x); end;
s:=h*((f(a)+f(b))/2+s);  gotoxy(10,18);
writeln('Интеграл равен ',s:11:4);
readkey;
end.
```

**Пример 2.76.** Методами трапеций и средних прямоугольников вычислить определенный интеграл

$$\int_a^b (2x^2 + 3x)\,dx.$$

**Программа**

```
program Integral;
uses  Crt, Dos;
var
  dx,x1,x2,e,i:real;
    function Fx(x:real):real;
    begin
    Fx:=2x*x+3x; {подынтегральная функция}
    end;
procedure CountViaBar;
var
  xx1,xx2:real;
```

```
  c:longint;
begin
  writeln('-------------------------------------------------');
  writeln('-->Метод средних прямоугольников.');
  writeln('Всего итераций:',round(abs(x2-x1)/e));
  i:=0;
  for c:=1 to round(abs(x2-x1)/e) do begin
    write('Итерация ',c,chr(13));
    xx1:=Fx(x1+c*e);   xx2:=Fx(x1+c*e+e);
    i:=i+abs(xx1+xx2)/2*e;
    end;
  writeln('-------------------------------------------------');
  writeln('Интеграл=',i);
end;
procedure CountViaTrap;
var
  xx1,xx2,xx3:real;
  c:longint;
begin
  writeln('-------------------------------------------------');
  writeln('-->Метод трапеций.');
  writeln('Всего итераций:',round(abs(x2-x1)/e));
  i:=0;
  for c:=1 to round(abs(x2-x1)/e) do begin
    write('Итерация ',c,chr(13));
    xx1:=Fx(x1+c*e);   xx2:=Fx(x1+c*e+e);
    if xx2>xx1 then xx3:=xx1 else xx3:=xx2;
    i:=i+abs(xx2-xx1)*e+abs(xx3)*e;
  end;
  writeln('-------------------------------------------------');
  writeln('Интеграл=',i);
end;
begin
  writeln('-=Программа вычисления определенного интеграла=-');
  writeln('Введите исходные значения:');
  write('Начальное значение x (x1)='); Readln(x1);
  write('Конечное значение x (x2)='); Readln(x2);
  write('Точность вычисления (e)='); Readln(e);
  CountViaBar;   CountViaTrap;
writeln('-------------------------------------------------');
Readln;
end.
```

### 2.9.3. Метод Монте-Карло

Метод статистических испытаний, называемый также *методом Монте-Карло*, применяют для решения разнообразных задач вычислительной математики, в том числе для вычисления интегралов. Рассмотрим вначале один из простых вариантов метода Монте-Карло, который можно интерпретировать как статистический метод прямоугольников (рис. 2.36).

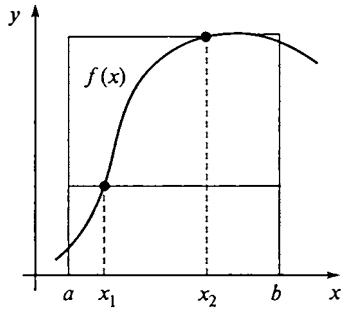



**Рис. 2.36.** Метод Монте-Карло

На отрезке интегрирования $[a; b]$ выберем $N$ случайных точек $x_1, x_2, \ldots, x_N$, являющихся значениями случайной величины $x$ с равномерным распределением на данном отрезке. Для каждой точки вычислим площадь прямоугольника, одна сторона которого равна $b - a$, а вторая — значению функции в данной точке $f(x_i)$: $S_i = (b - a) f(x_i)$.

Вследствие случайного выбора узла $x_i$ значение площадей $S_i$ также будет носить случайный характер. В качестве приближенного значения интеграла можно принять результат усреднения площадей $S_i$:

$$\int_a^b f(x)dx \approx \frac{S_1 + S_2 + \ldots + S_N}{N} = \frac{b - a}{N} \sum_{i=1}^{N} f(x_i). \qquad (2.18)$$

Погрешность вычисления интеграла будет уменьшаться с ростом числа испытаний $N$ по закону $\varepsilon \approx N^{-1/2}$.

Полученная формула формально совпадает с формулой правых прямоугольников, но различие состоит в том, что в формуле

(2.18) узлы интегрирования расположены регулярно, а в данном случае расположение узлов носит случайный характер.

Формула (2.18) непосредственно обобщается на кратные интегралы

$$\iint\limits_{G} \ldots \int f(x, y, \ldots, z)dv = \frac{v_G}{N} \sum_{i=1}^{N} f(x_i, y_i, \ldots, z_i),$$

где $v_G$ — объем области интегрирования. Например, для двукратного интеграла с прямоугольной областью интегрирования имеем

$$\int\limits_{a}^{b} \int\limits_{c}^{d} f(x, y)dxdy = \frac{(b-a)(d-c)}{N} \sum_{i=1}^{N} f(x_i, y_i).$$

**Пример 2.77.** Интегрирование методом Монте-Карло на области $D$, включенной в прямоугольник $[a_x, b_x] \cdot [a_y, b_y]$, с использованием $n$ опорных точек.

**Программа**

```
#include "ap.h"
/*-------------------------------------------------
This routines must be defined by the programmer:
bool d(double x, double y);
double f(double x, double y);
-------------------------------------------------*/
double montecarlointegration(const double& ax,
  const double& bx,
  const double& ay,
  const double& by,
  const int& n);
{
  double result;     int i;
  double x;     double y;
  i = 1;     result = 0;
  do
  {
    x = ax+(bx-ax)*ap::randomreal();
    y = ay+(by-ay)*ap::randomreal();
```

```
    if( d(x, y) )
    {
      result = result+f(x, y);
    }
    i = i+1;
  }
  while(i<=n);
  result = (bx-ax)*(by-ay)*result/n;
  return result;
}
```

### 2.9.4. Метод Симпсона

Геометрически иллюстрация формулы Симпсона состоит в том, что на каждом из сдвоенных частичных отрезков заменяем дугу данной кривой дугой графика квадратного трехчлена (рис. 2.37).

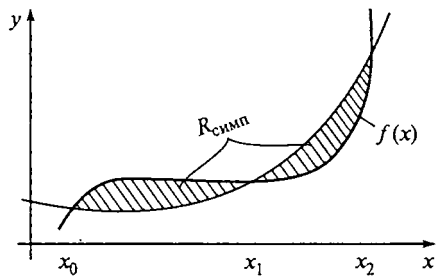



**Рис. 2.37.** Метод Симпсона

Разобьем отрезок интегрирования $[a; b]$ на $2n$ равных частей длиной $h = \dfrac{b-a}{2n}$. Обозначим точки разбиения $x_0 = a$; $x_1 = x_0 + h$, ..., $x_i = x_0 + ih$, ..., $x_{2n} = b$. Значения функции $f$ в точках $x_i$ обозначим $y_i$, т. е. $y_i = f(x_i)$. Тогда, согласно методу Симпсона

$$S \approx \int_a^b f(x)dx \approx \frac{b-a}{6n}(y_0 + 4y_1 + 2y_2 + \ldots + 4y_{2n-1} + y_{2n}) \approx$$

$$\approx \frac{b-a}{6n}(y_0 + y_{2n} + \sum_{i=0}^{2n-1}(3 + (-1)^{i-1})y_i).$$

**Пример 2.78.** Вычислить определенный интеграл по формуле Симпсона

$$\int_0^1 \frac{dx}{1+x^2} \text{ при } h = 0{,}25.$$

***Решение***

$$\int_0^1 \frac{dx}{1+x^2} = \frac{h}{3}[y_0 + 4y_1 + 2y_2 + 4y_3 + y_4].$$

Вычислим значения подынтегральной функции $f(x) = \dfrac{1}{1+x^2}$ в точках

$$x_0 = 0; \quad x_1 = 0{,}25; \quad x_2 = 0{,}5; \quad x_3 = 0{,}75; \quad x_4 = 1,$$

получим

$$\int_0^1 \frac{dx}{1+x^2} = \frac{0{,}25}{3}\left[1 + \frac{4}{1+0{,}25^2} + \frac{2}{1+0{,}5^2} + \frac{4}{1+0{,}75^2} + \frac{1}{2}\right] \approx 0{,}702.$$

*Ответ:* $\int_0^1 \dfrac{dx}{1+x^2} = 0{,}702.$

**Пример 2.79.** Вычислить интеграл $I = \int_0^1 \dfrac{dx}{1+x}$, $n = 10$ методом Симпсона, где $n$ — количество разбиений отрезка интегрирования.

***Решение***

Имеем $n = 10$, отсюда $h = \dfrac{1-0}{10} = 0{,}1$. Результаты вычислений приведены в таблице.

| $i$ | $x_i$ | $y_{2j-1}$ | $y_{2j}$ |
|---|---|---|---|
| 0 | 0 | | $y_0 = 1{,}00000$ |
| 1 | 0,1 | 0,90909 | |

*Окончание табл.*

| $i$ | $x_i$ | $y_{2j-1}$ | $y_{2j}$ |
|---|---|---|---|
| 2 | 0,2 | | 0,83333 |
| 3 | 0,3 | 0,76923 | |
| 4 | 0,4 | | 0,71429 |
| 5 | 0,5 | 0,66667 | |
| 6 | 0,6 | | 0,62500 |
| 7 | 0,7 | 0,58824 | |
| 8 | 0,8 | | 0,55556 |
| 9 | 0,9 | 0,52632, | |
| 10 | 1,0 | | 0,50000 = $y_n$ |
| $\Sigma$ | | 3,45955($\sigma$1) | 2,72818($\sigma$2) |

По формуле Симпсона получаем:

$$I \approx \frac{h(y_0 + y_n + 4\sigma_1 + 4\sigma_2)}{3} = 0{,}69315.$$

Подсчитаем погрешность полученного результата. Полная погрешность $R$ складывается из погрешностей действий $R_1$ и остаточного члена $R_2$. Очевидно

$$R_1 = \sum A(i)\,\varepsilon; \quad i = \overline{1, n},$$

где $A(i)$ — коэффициенты формулы Симпсона; $\varepsilon$ — максимальная ошибка округления значений подынтегральной функции.

Тогда

$$R_1 = n \cdot h \cdot \varepsilon.$$

**Пример 2.80**. Вычислить интеграл $\int_2^3 \frac{\cos}{x}\,dx$ методом Симпсона при $n = 10$.

***Решение***

| $x_i$ | $\cos x_i$ | $f(x_i)=\frac{\cos x_i}{x_i}$ | $n=10$ |
|---|---|---|---|
| 2 | –0,41613 | –0,208065 | 1 |
| 2,05 | –0,46107 | –0,224912 | |
| 2,1 | –0,59485 | –0,240405 | 4 |
| 2,15 | –0,54736 | –0,254586 | |
| 2,2 | –0,58850 | –0,267500 | 2 |
| 2,25 | –0,62817 | –0,279187 | |
| 2,3 | –0,66628 | –0,289687 | 4 |
| 2,35 | –0,70271 | –0,299026 | |
| 2,4 | –0,73739 | –0,307246 | 2 |
| 2,45 | –0,77023 | –0,314380 | |
| 2,5 | –0,80114 | –0,320465 | 4 |
| 2,55 | –0,83005 | –0,325510 | |
| 2,6 | –0,85689 | –0,329573 | 2 |
| 2,65 | –0,88158 | –0,332672 | |
| 2,7 | –0,90407 | –0,334841 | 4 |
| 2,75 | –0,92430 | –0,336109 | |
| 2,8 | –0,94222 | –0,336507 | 2 |
| ,85 | –0,95779 | –0,336067 | |
| 2,9 | –0,97096 | –0,334814 | 4 |
| 2,95 | –0,98170 | –0,332780 | |
| 3 | –0,98999 | –0,329997 | 1 |

Оценки для погрешности $R$ метода Симпсона:

$R \le 0{,}0000017$ для $h = 0{,}1$, $R \le 0{,}0000002$ для $h = 0{,}05$.

Чтобы погрешность округления не искажала столь точный результат для формулы Симпсона, все вычисления проводились с шестью знаками после запятой.

Окончательные результаты сведены в таблицу

| $h$ | $I$ | $R$ |
|---|---|---|
| 0,1 | –0,30335 | 0,0000017 |
| 0,05 | –0,30335 | 0,0000002 |

**Подпрограмма**

```
Procedure INTEGR (var S:real; K:integer); {Метод Симпсона}
begin
   S := 0.0;   X := A;
   IF K<5 THEN
     FOR I := 0  TO  N  DO
     begin
     IF K <>3   THEN  X := A+ H*I   ELSE X:=A+H/2+H*I;
     F := FUNC (X);
     IF K=4 THEN  F := F/2.0;
     IF (I=0) AND (K<>2)  THEN S := S + F;
     IF (I=N) AND ((K=2) OR (K=4))  THEN S := S + F;
     IF (I<>0) AND (I<>N)  THEN
         IF K<>4  THEN S := S+F
             ELSE S := S + F*2;
     end
   ELSE
 begin
     FOR I := 1  TO  N-1  DO
         S := 2*FUNC(X+H*I) + 4*FUNC(X+H*I-H/2) + S;
     S := S + 4*FUNC(B-H/2) + FUNC(A) + FUNC(B);
     S := S / 6;
 end;
 S := S * H;
end;
```

**Пример 2.81.** Вычислить интеграл методом Симпсона

$$\int_a^b \frac{\sin(\pi x/2)}{x}\,dx.$$

**Программа**

```
Program integral; {Метод Симпсона}
Uses crt;
var  i,n,c:integer;
     a,b,h,x,s:real;
     function f(x:real):real;
     begin
     f:=(1/x)*sin(Pi*x/2);
     end;
begin
clrscr;  gotoxy(10,10);  textbackground(1); clrscr;
write('Введите нижний предел интегрирования ');
readln(a);  gotoxy(10,12);
write('Введите верхний предел интегрирования ');
readln(b);  gotoxy(10,14);
write('Введите количество отрезков '); readln(n);
h:=(b-a)/n;  s:=0;  x:=a;  c:=1;
for i:=1 to n-1 do
begin  x:=x+h;   s:=s+(3+c)*f(x);   c:=-c;  end;
s:=h*(f(a)+f(b)+s)/3;  gotoxy(10,18);
writeln('Интеграл равен ',s:11:5);
readln;
end.
```

**Пример 2.82.** Вычислить интеграл $\int_a^b \frac{0{,}43429}{x \ln x}\,dx$ методами трапеций и Симпсона.

**Программа**

```
program tr_s;
uses crt, graph;
```

```
var
a,b:real;       { Границы отрезка }
r,r2:real;      { Предыдущее и текущее значения интеграла}
n:integer;      { Счетчик }
{ Интегрируемая функция }
    function f(x:real):real;
    begin
    f:=0.43429/(x*ln(x));
    end;
{ Метод трапеций }
function trap(a,b:real;n:integer):real;
var
s:real;         {Сумма }
h:real;         { Шаг }
m:integer;      { Счетчик }
begin
h:=(b-a)/(n-1);
s:=(f(a)+f(b))/2;       { Начальное значение суммы }
for m:=1 to n-2 do s:=s+f(a+m*h);
trap:=s*h;              {Значение интеграла }
end;
{ Метод Симпсона }
function simpson(a,b:real;n:integer):real;
var
s:real;          { Сумма }
h:real;          { Шаг }
m:integer;       { Счетчик }
mn:integer;      { Очередной множитель }
begin
h:=(b-a)/(n-1);    { Вычисление шаг }
s:=f(a)+f(b);      { Начальное значение шага }
mn:=4;             { Первый множитель }
{ Суммирование остальных элементов }
for m:=1 to n-2 do begin
s:=s+mn*f(a+h*m);
if (mn=4) then mn:=2 else mn:=4;
end;
simpson:=s*h/3; { Вычисленное значение }
```

```
end;
Begin
clrscr;
write('Введите нижний предел интегрирования ');
readln(a);
write('Введите верхний предел интегрирования ');
readln(b);
write('Введите количество отрезков '); readln(n);
writeln('Интеграл равен ', trap(a,b,n), simpson(a,b,n));
readln;
end.
```

## 2.10. Математическая статистика

**Математическая статистика** — раздел математики, посвященный математическим методам систематизации, обработки и использования статистических данных для практических и научных выводов. Такие понятия, как генеральная совокупность, выборка, вариационный ряд, характеристики вариационного ряда, станут для школьной математики столь же обычными и не менее важными понятиями, как медиана, апофема, вписанный угол, монотонность функций или производная.

### 2.10.1. Вычисление средних

В статистике очень часто применяется вычисление средней арифметической величины. Пусть величина $X$ имеет $n$ значений — $X_1$; $X_2$, ..., $X_n$, тогда

1) вычисление среднего арифметического осуществляется по формуле

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n};$$

2) среднее квадратичное этих же чисел:

$$\overline{X}_q = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n}};$$

3) среднее кубическое:

$$\overline{X}_k = \sqrt[3]{\frac{1}{n}\sum_{i=1}^{n} x_i^3} = \sqrt{\frac{x_1^3 + x_2^3 + \ldots + x_n^3}{n}};$$

4) среднее гармоническое:

$$\overline{X}_h = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}};$$

5) общая формула, пригодная для вычисления всех средних:

$$M = \left[ \frac{\sum_{i=1}^{n} x_i^k}{n} \right]^{\frac{1}{k}}.$$

Если $k = 1$, то производится вычисление среднего арифметического, если $k = 2$ — среднего квадратичного, $k = 3$ — среднего кубического, $k = -1$— среднего гармонического.

Таким образом, можно составить одну программу для вычисления всех средних. Подчеркнем, что средняя арифметическая используется, если размерность оцениваемой величины равна единице (средняя длина). Если необходимо найти среднее значение площади нескольких участков, то лучше в качестве средней взять среднюю квадратичную. Если же размерность оцениваемой величины равна трем (средний объем), то в качестве средней берется средняя кубическая.

### 2.10.2. Числовые характеристики случайных величин

Многие явления в природе, технике, экономике и в других областях носят случайный характер. В этом случае величина, принимающая свои значения в зависимости от исходов некоторого испытания (опыта), называется случайной величиной.

Пусть $X$ — дискретная случайная величина, возможными значениями которой являются числа $x_1, x_2, \ldots, x_n$.

Обозначим через $p_i = P(X = x_i)$ (где $i = 1, 2, ..., n$) вероятности этих значений, т. е. $p_i$ — вероятность события, состоящего в том, что $X$ принимает значение $x_i$.

Закон распределения полностью задает дискретную случайную величину. Однако часто закон распределения случайной величины неизвестен. В таких ситуациях ее описывают числовыми характеристиками.

Случайные величины характеризуются следующими числовыми параметрами.

- *Математическое ожидание* ($M(X)$) — это статистическое среднее случайной величины:

$$M(X) = \sum_{i=1}^{n} x_i p_i,$$

где $x_i$ — значение случайной величины; $p_i$ — вероятность появления этой величины, определяемая по формуле

$$\sum_{i=1}^{n} p_i = 1.$$

Для равновероятных событий:

$$M(X) = \frac{\sum_{i=1}^{n} x_i}{n}.$$

- *Дисперсия* — это математическое ожидание квадрата отклонения случайной величины от ее математического ожидания:

$$D(x) = \sum_{i=1}^{n} (x_i - M(X))^2 p_i.$$

- *Среднее квадратичное отклонение* случайной величины $\sigma$ — корень квадратный из дисперсии:

$$\sigma(x) = \sqrt{D(x)};$$

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - M(X))^2}.$$

- *Точность оценки* — отношение среднего квадратичного отклонения к корню квадратному из количества измерений $n$:

$$\lambda_M = \frac{\sigma}{\sqrt{n}}.$$

- *Показатель достоверности* математического ожидания — отношение математического ожидания к точности оценки:

$$T_M = \frac{M(X)}{\lambda_M}.$$

- *Коэффициент корреляции* определяет зависимость между двумя потоками случайных величин:

$$R(x, y) = \frac{M(x \cdot y) - M(x)M(y)}{\sigma(x)\sigma(y)}.$$

Коэффициент корреляции принимает значения на отрезке $-1 \le R(x, y) \le 1$.

Потоки случайных величин, для которых $R(x, y) = 0$, называются *некоррелированными* (*независимыми*).

Рассмотрим алгоритмы для детерминированной выборки случайных чисел.

### 2.10.3. Метод середины квадрата

Один из ранних генераторов случайных чисел, принадлежащий Джону фон Нейману, известен как *метод середины квадрата*.

Метод используется для генерации $k$-разрядных псевдослучайных чисел. Должно быть задано $k$-разрядное начальное число $x_0$ (для удобства предполагаем, что $k$ четно). Это число возводится в квадрат и получается число $y$. Число $y$ должно иметь $2k$ разрядов. Если число разрядов меньше $2k$, то число $y$ дополняется слева нулями. Затем в $y$ выделяют средние $k$ разрядов, которые и дают новое случайное число.

Алгоритм сводится к выполнению следующих шагов:

***Шаг 0.*** [Инициализация] $x := x_0$.

***Шаг 1.*** [Основной цикл] *FOR* $j := 1$ *TO* $m$ *DO шаг 2*; *Stop*.

*Шаг 2.* [Генерация нового случайного числа $x_j$]
$y := x^2$; $x_j :=$ (*средние k разрядов числа y*); $x := x_j$.

(Число $y$ будет иметь $2k$ разрядов, а следующее случайное число $x_j$ получается, если удалить по $k/2$ разрядов с каждого конца $y$. Десятичная точка помещается перед первым разрядом числа $x_j$ до поступления его на выход случайного генератора.)

**Пример 2.83.** Получить три случайных числа методом середины квадрата

$k = 4$; $x_0 = 3167$; $y = 10029889$;
$x_1 = 0298$; $y = 00088804$;
$x_2 = 0888$; $y = 00788544$.
$x_3 = 7885$;

**Пример 2.84.** Для $k = 4$ и $x_0 = 2134$ получить первые восемь псевдослучайных чисел, выработанные алгоритмом:

| | |
|---|---|
| $x_0^2 = 04\ 5539\ 56$ | $x_1 = 0{,}5539$ |
| $x_1^2 = 30\ 6805\ 21$ | $x_2 = 0{,}6805$ |
| $x_2^2 = 46\ 3080\ 25$ | $x_3 = 0{,}3080$ |
| $x_3^2 = 09\ 4864\ 00$ | $x_4 = 0{,}4864$ |
| $x_4^2 = 23\ 6584\ 96$ | $x_5 = 0{,}6584$ |
| $x_5^2 = 43\ 3490\ 56$ | $x_6 = 0{,}3490$ |
| $x_6^2 = 12\ 1801\ 00$ | $x_7 = 0{,}1801$ |
| $x_7^2 = 03\ 2436\ 01$ | $x_8 = 0{,}2436$ |

Несмотря на видимую случайность чисел, генерируемых алгоритмом, ему свойственны недостатки. В самом деле, если в последовательности когда-нибудь появится число **0.0000**, то все следующие за ним числа будут также равны **0.0000**.

Таким образом, многое зависит от начального выбора $k$ и $x_0$.

### 2.10.4. Линейный конгруэнтный метод

Метод используется для генерации последовательности $x_1$, $x_2$, ..., $x_m$ из $m$ псевдослучайных чисел. Должны быть заданы следующие входные значения:

- $b$ — целочисленный множитель, $b \geq 1$;
- $x_0$ — начальное случайное целое число, $x_0 \geq 1$;
- $k$ — шаг; $k \geq 0$ целое;
- $m$ — целочисленный модуль, $m \geq x_0$, $b$, $k$.

Случайные числа генерируются по рекуррентной формуле

$$x_i = (bx_{i-1} + k) \mod m.$$

Алгоритм сводится к выполнению следующих шагов:

***Шаг 1.*** [Основной цикл]

*FOR* $j := 1$ *TO* $m$ *DO шаг2, шаг3*; *Stop*.

***Шаг 2.*** [Генерация нового необработанного случайного числа]

$$x_i := (b * x_{i-1} + k) \mod m.$$

(Число $x_i$ должно лежать в интервале $0 \leq x_i < m$. По определению, если $a \mod m = x$ для целых $a$ и $m$, то $x$ есть остаток от целочисленного деления $a$ на $m$. Например, $5 \mod 3 = 2$; $7 \mod 4 = 3$; $9 \mod 3 = 0$.)

***Шаг 3.*** [Генерация следующего случайного числа]

$$y_i := x_i / m$$

($y_i$ будет лежать в интервале $0 \leq y_i < 1$ и будет обладать требуемым распределением).

**Пример 2.85.** Получить три случайных числа линейным конгруэнтным методом

$x_0 = 27$; $b = 5$; $k = 10$; $m = 40$;

$x_1 = (5*27 + 10) \mod 40 = 145 \mod 40 = 25$;

$x_2 = (5*25 + 10) \mod 40 = 15$;

$x_3 = (5*15 + 10) \mod 40 = 5$.

**Пример 2.86.** Для $k = 0$, $m = 2^{10}$, $b = 101$, $x_0 = 432$ получить первые восемь псевдослучайных чисел, выработанные алгоритмом:

| | |
|---|---|
| $x_1 = 624$ | $y_1 = 624/1024 = 0{,}610$ |
| $x_2 = 560$ | $y_2 = 560/1024 = 0{,}546$ |
| $x_3 = 240$ | $y_3 = 240/1024 = 0{,}234$ |
| $x_4 = 688$ | $y_4 = 688/1023 = 0{,}673$ |
| $x_5 = 880$ | $y_5 = 880/1024 = 0{,}859$ |
| $x_6 = 816$ | $y_6 = 816/1024 = 0{,}796$ |
| $x_7 = 496$ | $y_7 = 496/1024 = 0{,}486$ |
| $x_8 = 944$ | $y_8 = 944/1024 = 0{,}923$ |

Существует такой выбор параметров $k$, $b$, $m$, $x_0$, при котором алгоритм будет генерировать числа на отрезке [0; 1], представляющиеся непредсказуемыми и удовлетворяющие определенным статистическим критериям. Для всех практических целей эти числа оказываются последовательностью наблюдений равномерно распределенной случайной переменной.

### 2.10.5. Полярный метод

Метод используется для генерации двух независимых случайных чисел с нормальным распределением ($N(0, 1)$) из двух независимых равномерно распределенных случайных чисел. Распределение $N(0, 1)$ преобразуется в $N(\mu, \sigma^2)$ с использованием центральной предельной теоремы.

Алгоритм сводится к выполнению следующих шагов:

***Шаг 1.*** [Генерация двух равномерно распределенных случайных чисел.] Генерируются два независимых случайных числа $u_1$ и $u_2$ с распределением $U(0, 1)$

$$v_1 := 2u_1 - 1;\ v_2 := 2u_2 - 1$$

($v_1$ и $v_2$ имеют распределение $U(-1, +1)$.

*Шаг 2.* [Вычисление и проверка $s$]

$$s := v_1^2 + v_2^2; \quad IF\ s \geq 1\ THEN\ GOTO\ шаг1.$$

*Шаг 3.* [Вычисление $n_1$ и $n_2$]

$$n_1 := v_1\sqrt{-2\ln s/s}; \quad n_2 := v_2\sqrt{-2\ln s/s}$$

*Stop* ($n_1$ и $n_2$ распределены по нормальному закону).

Алгоритм имеет важное вычислительное преимущество: он обычно генерирует по одному числу с нормальным распределением на каждое число с равномерным распределением. Правда, при этом следует убедиться, что компенсируется дополнительное время, затрачиваемое на вычисление натуральных логарифмов и квадратных корней.

**Пример 2.87.** Составить программу для вычисления всех средних. Произвести расчеты на ПЭВМ, результаты поместить в таблицу

| № опыта | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| 1 | 2,5 | 3,1 | 4,2 | 1,7 | 1,9 |
| 2 | 0,4 | 1,4 | 3,3 | 4,7 | 6,3 |
| 3 | 1,1 | 1,3 | 2,4 | 5,6 | 7,9 |

**Программа**

```
Program statist;
Uses crt;
var      k, i, n: integer; sr, s: real;
         x: array[1..100] of real;
begin
ClrScr;
write('Введите k и n'); readln(k, n);
  for i:=1 to n do readln(x[i]);
     s:=0;
     for i:=1 to n do s:=s+exp(k*ln(x[i]));
  sr:=exp((1/k)*ln(s/n));
writeln(sr);
Readkey;
end.
```

**Пример 2.88.** На телефонной станции производились наблюдения за числом неправильных соединений в минуту. Наблюдения в течение часа дали следующие результаты:

3, 1, 3, 4, 2, 1, 1, 3, 2, 7, 2, 0,
2, 4, 0, 3, 0, 2, 0, 1, 3, 3, 1, 2,
2, 0, 2, 1, 4, 3, 4, 2, 0, 2, 3, 1,
3, 1, 4, 2, 2, 1, 2, 5, 1, 1, 0, 1,
1, 2, 1, 0, 3, 4, 1, 2, 2, 1, 1, 5.

Составить программу для нахождения математического ожидания и среднего квадратичного отклонения.

**Программа**

```
Program  telephon;
Uses crt;
  Const    a: array[1..60] of integer =(3,1,3,4,2,1,1,3,2,7,2,0,
  2,4,0,3,0,2,0,1,3,3,1,2,2,0,2,1,4,3,4,2,0,2,3,1,3,1,4,2,2, 1,2,5,1,1,
  0,1,1,2,1,0,3,4,1,2,2,1,1,5);
var                  i, n, k: integer;
         e, mx, dx, q, x, w: real;
Begin
  ClrScr;  mx:=0;  dx:=0;
    for i:=1 to 60 do  mx:=mx+a[i];
      mx:=mx/60;
      for i:=1 to 60 do dx:=dx+sqr(a[i]-mx);
      w:=sqrt(dx/59);   q:=w/sqrt(60);   e:=mx/q;
      writeln('среднее =',mx:5:2);
      writeln('отклонение =',w:5:2);
      writeln('точность оценки =',q:5:2);
      writeln('показатель достоверности =',e:5:2);
Readkey;
End.
```

**Пример 2.89.** Даны результаты измерения роста (в см) случайно отобранных 100 студентов:

| Рост | 154—158 | 158—162 | 162—166 | 166—170 | 170—174 | 174—178 | 178—182 |
|---|---|---|---|---|---|---|---|
| Число студентов | 10 | 14 | 26 | 28 | 12 | 8 | 2 |

Составить программу для нахождения выборочного среднего (математическое ожидание), среднего квадратичного отклонения, точности оценки и показателя достоверности.

*Указание.* Середины интервалов принять в качестве значений 156; 160; 164; 168; 172; 176; 180.

**Программа**

```
Program statist;
Uses crt;
Const
    x:array[1..7] of integer=(156,160,164,168,172,176,180);
    p:array [1..7] of integer=(10,14,26,28,12,8,2);
var     mx, dx, w:real;
          i: integer;
begin
  ClrScr;    mx:=0;  dx:=0;
  for i:=1 to 7 do mx:=mx+x[i]*p[i];
    mx:=mx/100;
    for i:=1 to 7 do dx:=dx+sqr(x[i]-mx);
    w:=sqrt(dx/6);
    writeln('среднее=', mx:5:2);
    writeln('среднее квадратичное отклонение=',w:8:2);
readkey;
end.
```

## 2.11. Линейное программирование

Использование информационных технологий для решения задач оптимизации может быть получено с помощью методов оптимального проектирования.

Методы оптимизации подразделяются на:

- линейное и динамическое программирование (задачи распределения ресурсов);

- теорию массового обслуживания (задачи со случайным характером поступления и обслуживания заявок в системе);
- имитационное моделирование (задачи, где реальный эксперимент заменяется имитационной моделью);
- статистическое моделирование (задачи, в которых результат находится методами математической статистики из большого числа расчетов с различными факторами);
- теорию управляемых марковских процессов (задачи со случайными неконтролируемыми факторами);
- теорию игр (состязательные задачи в условиях неопределенности);
- теорию расписаний (задачи календарного упорядочения работ);
- сетевое планирование и управление (задачи с неопределенной оценкой времени выполнения различных видов работ);
- векторную оптимизацию (многокритериальные задачи);
- теорию распознавания образов и др.

При математическом моделировании требуется получить быстрый ответ на поставленный вопрос и получить возможность широкого экспериментирования.

Для успешного математического моделирования необходимо:

- учитывать главные свойства моделируемого объекта;
- пренебрегать его второстепенными свойствами;
- уметь отделить главные свойства от второстепенных.

### 2.11.1. Общий случай задачи оптимизации

В общем случае задачу оптимизации можно записать так:

$$\begin{cases} \text{ЦФ} \quad F = f(x_j) \to \max(\min, \text{const}); \\ \text{ОГР} \quad \begin{cases} g_1(x_j) \le (=; \ge) b_j; \\ \dots \\ g_i(x_j) \le (=; \ge) b_i; \\ \dots \\ g_m(x_j) \le (=; \ge) b_m; \end{cases} \\ \text{ГРУ} \quad d_j \le x_j \le D_j; \ i = 1 \dots m; \ j = 1 \dots n. \end{cases} \tag{2.19}$$

Систему (2.19) принято записывать более компактно:

$$\begin{cases} F = f(x_j) \to \max; \\ g_i(x_j) \le b_i; \\ d_j \le x_j \le D_j; \\ i = 1...m; \quad j = 1...n. \end{cases} \tag{2.20}$$

Запись (2.20) является **общей формой записи задач оптимизации.**

В эту систему входят три составляющие.

1. Целевая функция **(ЦФ)**, или критерий оптимизации, показывает, в каком смысле решение должно быть оптимальным, т. е. наилучшим. При этом возможны три вида назначения целевой функции:

- максимизация;
- минимизация;
- назначение заданного значения.

2. **ОГР** — ограничения, устанавливают зависимости между переменными. Они могут быть односторонними ($g_i(x_j) \le b_i$,) или двухсторонними ($a_i \le g_i(x_j) \le b$). Причем любое двухстороннее ограничение можно записать в виде двух односторонних ($g_i(x_j) \ge a_i$), ($g_i(x_j) \le b_i$).

3. **ГРУ** — граничные условия, показывают, в каких пределах могут быть значения искомых переменных в оптимальном решении.

Решения задачи, удовлетворяющие всем ограничениям и граничным условиям, называются *допустимыми*. Важной характеристикой задачи оптимизации является ее размерность, определяемая числом переменных $n$ и числом ограничений $m$. Соотношение этих величин является определяющим при постановке задачи оптимизации.

В общем случае ОГР имеют вид: $g_i(x_j) \le b_i$, $i = 1...m$; $j = 1...n$; их можно записать в виде: $g_i(x_j) + y_i = b_i y_i \ge 0$; $i = 1...m$; $j = 1...n$.

В этом случае общее число переменных $x_j$ и $y_i$, равное $N$, будет: $N = n + m$, а число уравнений останется прежним, равным $m$. Очевидно, что $N = n + m > m$, и система имеет бесчисленное множество решений. Значит, если ограничениями являются не-

равенства, то система всегда имеет бесчисленное множество решений.

Таким образом, условие $n > m$ — это непременное требование задач оптимизации.

Оптимальное решение — это наилучшее решение, но наилучшего решения во всех смыслах быть не может. Может быть лучшим только в одном, строго установленном смысле.

Принимающий решение должен абсолютно точно представлять, в чем заключается оптимальность решения, т. е. по какому критерию принимаемое решение должно быть оптимальным.

Критерий называют *целевой функцией*. С помощью критерия можно оценивать качества как желательные (например, прибыль, производительность, надежность), так и нежелательные (затраты, расходы, простои и т. д.). Тогда в первом случае стремятся к максимизации критерия, во втором — к минимизации.

Итак, задача имеет оптимальное решение, если она удовлетворяет двум требованиям:

- есть реальная возможность иметь более одного решения, т. е. существуют допустимые решения;
- имеется критерий, показывающий, в каком смысле принимаемое решение должно быть оптимальным, т. е. наилучшим из допустимых.

### 2.11.2. Решение задачи линейного программирования

Задача линейного программирования является частным случаем задачи оптимизации.

Задача, в которой требуется минимизировать (или максимизировать) линейную форму

$$\sum_{j=1}^{n} c_i x_i \to \min(\max)$$

при условии, что $\sum_{j=1}^{n} a_{ij} x_i \le b_j$, $j = 1, \ldots, m$ или $\sum_{i=1}^{n} a_{ij} x_i = b_j$, $j = m+1, \ldots, p$ и $x_i > 0$, $i = 1, \ldots, n$, называется **задачей линейного программирования** в произвольной форме записи.

Задача в матричной форме вида

$$\left.\begin{array}{c} \bar{c}^{\mathrm{T}}\bar{x} \to \min(\max), \\ A\bar{x} \le \bar{b}, \\ \bar{x} \ge \bar{0} \end{array}\right\}$$

называется **симметричной** (канонической) **формой** записи задачи линейного программирования.

Любую задачу линейного программирования можно привести к канонической форме.

Если система ограничений задана в форме $A\bar{x} \le \bar{b}$, то можно, введя дополнительные переменные, привести ее к виду

$$A\bar{x} + E\bar{y} = \bar{b}, \quad x \ge 0, \quad y \ge 0, \quad \text{где} \quad \bar{y} = [x_{n+1}, \ldots, x_{n+m},]^{\mathrm{T}}.$$

Если же ограничения в задаче заданы в форме $A\bar{x} \ge \bar{b}$, то

$$A\bar{x} - E\bar{y} = \bar{b}, \quad \bar{x} \ge 0, \quad \bar{y} \ge 0.$$

Рассмотрим задачу с ограничениями $A\bar{x} \le \bar{b}$. Эту систему ограничений можно представить в виде системы

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n + x_{n+1} = b_1; \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n + x_{n+2} = b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n + x_{n+m} = b_m. \end{cases}$$

Введем следующие обозначения:

$$\bar{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}, \quad \bar{\mathbf{A}}_1 = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \ldots \\ a_{m,1} \end{bmatrix}, \quad \bar{\mathbf{A}}_n = \begin{bmatrix} a_{1,n} \\ a_{2,n} \\ \ldots \\ a_{m,n} \end{bmatrix}, \quad \bar{\mathbf{A}}_{n+1} = \begin{bmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{bmatrix},$$

$$\bar{\mathbf{A}}_{n+m} = \begin{bmatrix} 0 \\ 0 \\ \ldots \\ 1 \end{bmatrix}, \quad \bar{\mathbf{A}}_0 = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}.$$

Тогда задачу линейного программирования можно записать в виде:

$$\sum_{i=1}^{n} c_i x_i \to \min(\max);$$

$$x_1\bar{\mathbf{A}}_1 + x_2\bar{\mathbf{A}}_2 + \ldots + x_n\bar{\mathbf{A}}_n + x_{n+1}\bar{\mathbf{A}}_{n+1} + \ldots + x_{n+m}\bar{\mathbf{A}}_{n+m} = \mathbf{A}_0;$$

$$\bar{x} \geq 0.$$

Векторы $\bar{\mathbf{A}}_i$ называются *векторами условий*, а сама задача линейного программирования — *расширенной* по отношению к исходной.

Пусть $D$ и $D_1$ — допустимые множества решений исходной и расширенной задач соответственно.

Тогда любой точке множества $D_1$ соответствует единственная точка множества $D$, и наоборот. В общем случае допустимое множество $D$ исходной задачи есть проекция множества $D_1$ расширенной задачи на подпространство исходных переменных.

Набор чисел $\bar{x} = [x_1, x_2, \ldots, x_n]^T$, удовлетворяющий ограничениям задачи линейного программирования, называется ее *планом*.

**Решением задачи линейного программирования** называется ее план, минимизирующий (или максимизирующий) линейную форму.

Введем понятие базисного решения. Из матрицы расширенной задачи $\mathbf{A}_p = \left\lfloor \bar{A}_1, \bar{A}_2, \ldots, \bar{A}_{n+m} \right\rfloor$ выберем $m$ линейно независимых векторов-столбцов, которые обозначим как матрицу $\mathbf{B}_{m\times m}$, а через $\mathbf{D}_{m\times n}$ — матрицу из оставшихся столбцов. Тогда $\mathbf{A}_p = [B, D]$, и ограничения расширенной задачи линейного программирования можно записать в виде:

$$\bar{\mathbf{A}}_p\bar{x} = \mathbf{B}\,\bar{x}_B + \mathbf{D}\,\bar{x}_D = \bar{A}_0.$$

Очевидно, что столбцы матрицы $\mathbf{B}$ образуют базис $m$-мерного пространства. Поэтому вектор $\bar{A}_0$ и любой столбец матрицы $\mathbf{D}$ можно представить в виде линейной комбинации столбцов матрицы $\mathbf{B}$.

Тогда $\mathbf{B}^{-1} \cdot \mathbf{B} \cdot \bar{x}_B + \mathbf{B}^{-1} \cdot \mathbf{D} \cdot \bar{x}_D = \mathbf{B}^{-1} \cdot \bar{A}_0$.

Отсюда $\bar{x}_B = \mathbf{B}^{-1} \cdot \bar{\mathbf{A}}_0 - \mathbf{B}^{-1} \cdot \mathbf{D} \cdot \bar{x}_D$.

Придавая $\bar{x}_D$ различные значения, будем получать различные решения $\bar{x}_B$.

Если положить $\bar{x}_D = \bar{0}$, то $\bar{x}_B = \mathbf{B}^{-1} \cdot \bar{A}_0$.

Решение $\bar{x}_B = \mathbf{B}^{-1} \cdot \bar{A}_0$ называют **базисным решением** системы из $m$ уравнений с $m + n$ неизвестными.

Если полученное решение содержит только положительные компоненты, то оно называется *базисным допустимым*.

Особенность допустимых базисных решений состоит в том, что они являются крайними точками допустимого множества $D_1$ расширенной задачи (рис. 2.38).

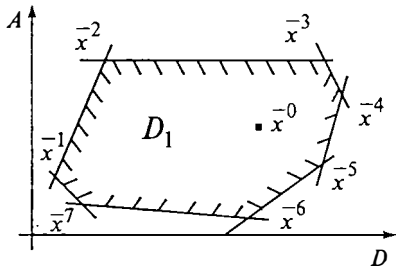



**Рис. 2.38.** Допустимые базисные решения

Если среди компонент $\bar{x}_B$ нет нулевых, то базисное допустимое решение называется **невырожденным**.

План $\bar{x}$ задачи линейного программирования будем называть **опорным**, если векторы условий $\bar{A}_i$ с положительными коэффициентами линейно независимы, т. е. опорный план — это базисное допустимое решение расширенной системы, угловая точка многогранника решений.

Опорное решение называется **невырожденным**, если оно содержит $m$ положительных компонентов (по числу ограничений). Невырожденный опорный план образуется пересечением $n$ гиперплоскостей из образующих допустимую область. В случае вырожденности в угловой точке многогранника решений пересекается более $n$ гиперплоскостей.

Зависимости между переменными (как целевые функции, так и ограничения) могут быть линейными и нелинейными. Линейными называются такие зависимости, в которые переменные входят в первой степени и с ними выполняются только действия сложения и вычитания. Иначе, если переменные входят не в первой степени или над ними выполняются другие (кроме сло-

жения и вычитания) операции, то зависимости называются нелинейными.

Задачи линейного программирования можно решить аналитическим путем и графическим методом. Второй метод — наглядный, но пригоден лишь для $n = 2$ (т. е. если число переменных равно 2).

**Пример 2.90.** Пусть уравнение прямой имеет вид $a_1x_1 + a_2x_2 = b$, т. е. задано уравнение $2x_1 + x_2 = 2$.

***Решение***

Перепишем исходное уравнение в виде $\frac{x_1}{1} + \frac{x_2}{2} = 1$.

При такой форме записи в знаменателе показаны отрезки, которые отсекает прямая на осях координат (рис. 2.39). Если от исходного уравнения перейти к неравенству $2x_1 + x_2 \le 2$, то графически решение этого неравенства показано на рис. 2.40, т. е. решением линейного неравенства с двумя переменными является полуплоскость. На рис. 2.40 часть плоскости, которая не удовлетворяет неравенству, расположена выше прямой и заштрихована. Координаты всех точек, принадлежащих к незаштрихованной части плоскости, имеют такие значения $x_1$ и $x_2$, которые удовлетворяют заданному неравенству. Эта полуплоскость является областью допустимых решений (ОДР).

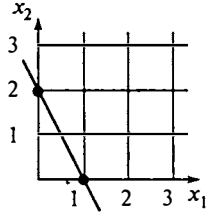



**Рис. 2.39.** Исходная прямая

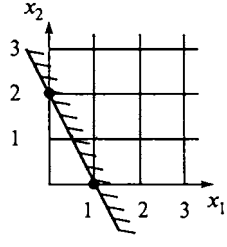



**Рис. 2.40.** Решение неравенства

**Пример 2.91**. Пусть дана система неравенств:

$$\begin{cases} x_1 + 4x_2 \le 14; \\ 3x_1 + 4x_2 \le 18; \\ 6x_1 + 2x_2 \le 27; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

***Решение***

Для удобства запишем ее в следующем виде:

$$\begin{cases} \dfrac{x_1}{14} + \dfrac{x_2}{7/2} \le 1; \\ \dfrac{x_1}{6} + \dfrac{x_2}{9/2} \le 1; \\ \dfrac{x_1}{9/2} + \dfrac{x_2}{27/2} \le 1; \\ x_1 \ge 0;\ x_2 \ge 0. \end{cases}$$

Графическое решение этой системы показано на рис. 2.41.

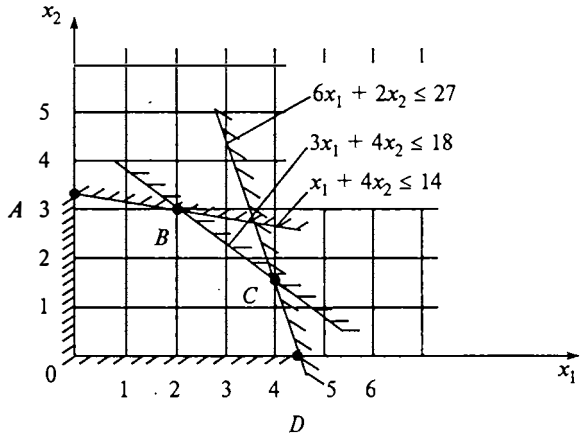



**Рис. 2.41.** Графическое решение системы

Решением системы являются координаты всех точек, принадлежащих ОДР, т. е. многоугольнику $ABCD0$. Так как в ОДР бесчисленное множество точек, значит, рассматриваемая система имеет бесчисленное множество допустимых решений.

Если требуется найти оптимальное решение, то нужно принять целевую функцию равной $F = x_1 + x_2 \to \max$. Эта зависимость на рис. 2.42 представлена в форме уравнения прямой с угловым коэффициентом $x_2 = F - x_1$, из которого видно, что $\operatorname{tg} a = -1$. При этом угол $a = 135°$, а величина $F$ равна отрезку, отсекаемому прямой на оси ординат. Если прямую перемещать

параллельно самой себе в направлении, указанном стрелками, то величина $F$ будет возрастать. Совместим теперь ОДР, изображенную на рис. 2.41, с линией целевой функции $F$, построенной на рис. 2.42, получим рис. 2.43.

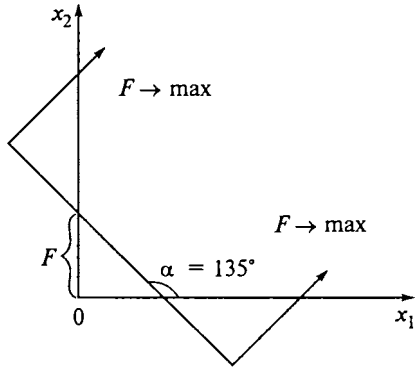



**Рис. 2.42.** Отсекаемый отрезок

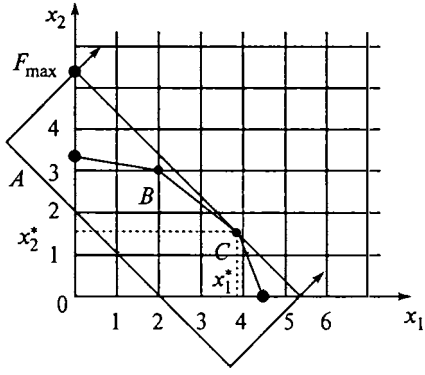



**Рис. 2.43.** График целевой функции

Поскольку требуется найти оптимальное решение, при котором целевая функция $F = x_1 + x_2 \to \max$, т. е. стремится к максимуму, будем перемещать график целевой функции в направлении увеличения $F$. Очевидно, что оптимальным решением будут координаты точки $C$, равные $x_1^*$ и $x_2^*$. При этом $F = F^*$.

На основании рассмотренного можно сделать вывод: *оптимальным решением* являются координаты вершин ОДР.

На этом базируется аналитический метод решения задач линейного программирования, который заключается в следующем:

- найти вершины ОДР, как точки пересечения ограничений;
- определить последовательно значения целевой функции в вершинах;
- вершина, в которой ЦФ приобретает оптимальное (максимальное или минимальное) значение, является оптимальной;
- координаты этой вершины и являются искомыми оптимальными значениями переменных.

### 2.11.3. Симплекс-метод

В геометрии используется такое понятие, как **«симплекс»**. Симплексом тел в $k$-мерном пространстве называется совокупность $(k + 1)$ его вершин.

Так, для плоскости при $k = 2$ симплексом будут три вершины треугольника. При $k = 3$ — четыре вершины четырехгранника и т. д. С учетом этого понятия аналитический метод решения задач линейного программирования называется симплекс-методом. Вычисления, обеспечивающие определение значения целевой функции и переменных в одной вершине, называются *итерацией*.

**Симплекс-метод** (метод последовательного улучшения плана) предназначен для решения общей задачи линейного программирования.

Пусть имеется задача:

$$Q(\bar{x}) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \min$$

с системой ограничений следующего вида:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m,1}x_1 + a_{m,2}x_2 + \ldots + a_{m,n}x_n = b_m. \end{cases}$$

Разрешим эту систему относительно переменных $x_1, \ldots, x_m$:

$$\begin{cases} x_1 = a'_{1,m+1}x_{m+1} + \ldots + a'_{1,n}x_n + b'_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_m = a'_{m,m+1}x_{m+1} + \ldots + a'_{m,n}x_n + b'_m. \end{cases}$$

Векторы условий, соответствующие $x_1, \ldots, x_m$, образуют базис. Переменные $x_1, \ldots, x_m$ назовем **базисными переменными**. Остальные переменные задачи — небазисные.

Целевую функцию можно выразить через небазисные переменные:

$$Q(\bar{x}) = c'_{m+1}x_{m+1} + c'_{m+2}x_{m+2} + \ldots + c'_n x_n + c'_0 \to \min.$$

Если приравнять небазисные переменные нулю: $x_{m+1} = 0$, $x_{m+2} = 0$, ..., $x_n = 0$, то соответствующие базисные переменные примут значения:

$$x_1 = b'_1; \quad x_2 = b'_2; \quad \ldots; \quad x_m = b'_m.$$

Вектор $\bar{x}$ с такими компонентами представляет собой угловую точку многогранника решений (допустимую) при условии, что $b'_i \geq 0$ (опорный план).

Теперь необходимо перейти к другой угловой точке с меньшим значением целевой функции. Для этого следует выбрать некоторую небазисную переменную и некоторую базисную так, чтобы после того, как «поменяем их местами», значение целевой функции уменьшилось. Такой направленный перебор в конце концов приведет к решению задачи.

Формализованный алгоритм симплекс-метода состоит из двух основных этапов:

1) построение опорного плана;

2) построение оптимального плана.

**Пример 2.92.** Симплекс-методом решить задачу линейного программирования.

Пусть $Q(\bar{x}) = x_4 - x_5 \to \min$. Система ограничений имеет вид:

$$\begin{cases} x_1 + x_4 - 2x_5 = 1; \\ x_2 - 2x_4 + x_5 = 2; \\ x_3 + 3x_4 + x_5 = 3. \end{cases}$$

Выберем в качестве базисных следующие переменные $\{x_1, x_2, x_3\}$ и разрешим систему относительно этих переменных. Система ограничений примет следующий вид:

$$\begin{cases} x_1 = 1 - x_4 + 2x_5; \\ x_2 = 2 + 2x_4 - x_5; \\ x_3 = 3 - 3x_4 - x_5. \end{cases}$$

***Решение***

Переменные $\{x_4, x_5\}$ являются небазисными. Если взять $x_4 = 0$ и $x_5 = 0$, то получим угловую точку (опорный план):

$$\bar{x}^1 = [1 \;\; 2 \;\; 3 \;\; 0 \;\; 0]^T,$$

которой соответствует $Q(\bar{x}^1) = 0$.

Значение целевой функции можно уменьшить за счет увеличения $x_5$. При увеличении $x_5$ величина $x_1$ также увеличивается, а $x_2$ и $x_3$ — уменьшаются. Причем величина $x_2$ раньше может стать отрицательной. Поэтому, вводя в базис переменную $x_5$, одновременно $x_2$ исключаем из базиса. В результате после очевидных преобразований получим следующие выражения для новой системы базисных переменных и целевой функции:

$$\begin{cases} x_5 = 2 - x_2 + 2x_4; \\ x_1 = 5 - 2x_2 + 3x_4; \\ x_3 = 1 + x_2 - 5x_4, \end{cases}$$

$$Q(\bar{x}) = -2 - x_4 + x_2 \to \min.$$

Соответствующий опорный план:

$$\bar{x}^2 = [5 \;\; 0 \;\; 1 \;\; 0 \;\; 2]^T \quad \text{и} \quad Q(\bar{x}^2) = -2.$$

Целевую функцию можно уменьшить за счет увеличения $x_4$. Увеличение $x_4$ приводит к уменьшению только $x_3$. Поэтому вводим в базис переменную $x_4$, а $x_3$ исключаем из базиса. В результате получим следующие выражения для новой системы базисных переменных и целевой функции:

$$\begin{cases} x_4 = \dfrac{1}{5} + \dfrac{1}{2}x_2 - \dfrac{1}{5}x_3; \\ x_1 = \dfrac{28}{5} - \dfrac{7}{5}x_2 - \dfrac{3}{5}x_3; \\ x_5 = \dfrac{12}{5} - \dfrac{3}{5}x_2 - \dfrac{2}{3}x_3, \end{cases}$$

$$Q(\bar{x}) = -\frac{11}{5} + \frac{4}{5}x_2 + \frac{1}{5}x_3 \to \min.$$

Тогда соответствующий опорный план равен

$$\bar{x}^3 = \left[\frac{28}{5} \;\; 0 \;\; 0 \;\; \frac{1}{5} \;\; \frac{12}{5}\right]^{\mathrm{T}},$$

а значение целевой функции

$$Q(\bar{x}^3) = -\frac{11}{5}.$$

Так как все коэффициенты при небазисных переменных в целевой функции неотрицательны, то нельзя уменьшить целевую функцию за счет увеличения $x_2$ или $x_3$, следовательно, полученный план $\bar{x}^3$ является оптимальным.

**Пример 2.93.** Пусть имеется задача: $Q(\bar{x}) = x_1 - x_2 \to \min$. Система ограничений имеет вид:

$$\begin{cases} x_3 = 1 + x_1 - x_2; \\ x_4 = 2 - x_1 + 2x_2; \\ \bar{x} \geq 0. \end{cases}$$

***Решение***

Переменные $\{x_3, x_4\}$ — базисные, а $\{x_1, x_2\}$ — небазисные переменные. Опорный план:

$$\bar{x}^0 = [0 \;\; 0 \;\; 1 \;\; 2], \quad Q(\bar{x}^0) = 0.$$

Теперь вводим в базис переменную $x_1$, а $x_4$ исключаем из базиса. В результате получаем следующие выражения для базисных переменных и целевой функции:

$$\begin{cases} x_1 = 2 + 2x_2 - x_4; \\ x_3 = 3 + x_2 - x_4, \end{cases}$$

$$Q(\bar{x}) = -2 - 3x_2 + x_4.$$

Опорный план равен $\bar{x}^1 = [2 \;\; 0 \;\; 3 \;\; 0]^{\mathrm{T}}$, значение целевой функции $Q(\bar{x}^1) = -2$. Можно заметить, что при увеличении $x_2$ значения переменных $x_1$ и $x_3$ также возрастают, т. е. при $x_2 \to -\infty$ в допустимой области $Q(\bar{x}) \to -\infty$ (задача не имеет решения).

*Замечание.* В процессе поиска допустимого плана может быть выявлена противоречивость системы ограничений.

**Пример 2.94.** С использованием таблиц решить задачу линейного программирования симплекс-методом.

$$Q(\bar{x}) = x_4 - x_5 \to \min.$$

Система ограничений имеет вид:

$$\begin{cases} x_1 + x_4 - 2x_5 = 1; \\ x_2 - 2x_4 + x_5 = 2; \\ x_3 + 3x_4 + x_5 = 3; \\ \bar{x} \geq 0. \end{cases}$$

***Решение***

Пусть необходимо решить задачу:

$$Q(\bar{x}) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \min(\max)$$

$$\begin{cases} a_{1,1}x_1 + \ldots\ldots + a_{1,n}x_n = b_1; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m,1}x_1 + \ldots\ldots + a_{m,n}x_n = b_m; \\ a_{m+1,1}x_1 + \ldots + a_{m+1,n}x_n \leq b_{m+1}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m+p,1}x_1 + \ldots + a_{m+p,n}x_n \leq b_{m+p}. \end{cases}$$

Введем дополнительные переменные, чтобы преобразовать ограничения-неравенства к равенствам. В ограничениях-равенствах дополнительные переменные должны быть нулевыми. Тогда система ограничений принимает вид:

$$\begin{cases} 0 = b_1 - a_{1,1}x_1 - \ldots\ldots\ldots - a_{1,n}x_n; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 = b_m - a_{m,1}x_1 - \ldots\ldots\ldots - a_{m,n}x_n; \\ x_{n+1} = b_{m+1} - a_{m+1,1}x_1 - \ldots - a_{m+1,n}x_n; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{n+p} = b_{m+p} - a_{m+p,1}x_1 - \ldots - a_{m+p,n}x_n, \end{cases}$$

где $x_{n+i} \geq 0$, $i = 1, \ldots, p$.

В качестве базисных переменных выберем систему дополнительно введенных переменных. Тогда симплексная таблица для преобразованной задачи будет иметь вид табл. 2.3.

*Таблица* 2.3. **Симплексная таблица для преобразованной задачи**

| | $-x_1$ | $-x_2$ | ... | $-x_s$ | ... | $-x_n$ | 1 |
|---|---|---|---|---|---|---|---|
| 0 | $a_{11}$ | $a_{12}$ | ... | $a_{1,s}$ | ... | $a_{1,n}$ | $b_1$ |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 0 | $a_{m,1}$ | $a_{m,2}$ | ... | $a_{m,s}$ | ... | $a_{m,n}$ | $b_m$ |
| $x_{m+1}$ | $a_{m+1,1}$ | $a_{m+1,2}$ | ... | $a_{m+1,s}$ | ... | $a_{m+1,n}$ | $b_{m+1}$ |
| ... | ... | ... | ... | ... | ... | ... | ... |
| $x_{m+p}$ | $a_{m+p,1}$ | $a_{m+p,2}$ | ... | $a_{m+p,s}$ | ... | $a_{m+p,n}$ | $b_{m+p}$ |
| $Q(\bar{x})$ | $-c_1$ | $-c_2$ | ... | $-c_s$ | ... | $-c_n$ | 0 |

В случае базисных переменных $\{x_1, x_2, x_3\}$ начальная симплексная таблица для данного примера будет выглядеть следующим образом (табл. 2.4).

*Таблица* 2.4. **Симплексная таблица**

| | $-x_4$ | $-x_5$ | 1 |
|---|---|---|---|
| $x_1 =$ | 1 | –2 | 1 |
| $x_2 =$ | –2 | 1 | 2 |
| $x_3 =$ | 3 | 1 | 3 |
| $Q(\bar{x})$ | –1 | 1 | 0 |

Она уже соответствует опорному плану $\bar{x}^1 = [1\ 2\ 3\ 0\ 0]^T$ (столбец свободных членов).

**Построение оптимального плана.** Для того чтобы опорный план был оптимален, при минимизации целевой функции необходимо, чтобы коэффициенты в строке целевой функции были неположительными (в случае максимизации — неотрицательными), т. е. при поиске минимума мы должны освободиться от положительных коэффициентов в строке $Q(\bar{x})$.

**Выбор разрешающего элемента.** Если при поиске минимума в строке целевой функции есть коэффициенты больше нуля, то выбираем столбец с положительным коэффициентом в строке

целевой функции в качестве разрешающего. Пусть это столбец с номером $l$.

Для выбора разрешающей строки (разрешающего элемента) среди положительных коэффициентов разрешающего столбца выбираем тот, для которого отношение коэффициента в столбце свободных членов к коэффициенту в разрешающем столбце минимально:

$$\frac{b_r}{a_{rl}} = \min\left\{\frac{b_i}{a_{il}} \middle| a_{il} \geq 0\right\},$$

где $a_{rl}$ — разрешающий (направляющий) элемент; строка $r$ — разрешающая.

Для перехода к следующей симплексной таблице (следующему опорному плану с меньшим значением целевой функции) делается шаг модифицированного жорданова исключения с разрешающим элементом $a_{rl}$.

Если в разрешающем столбце нет положительных коэффициентов, то целевая функция не ограничена снизу (при максимизации — не ограничена сверху).

**Шаг модифицированного исключения.** На месте разрешающего элемента ставится 1. Остальные элементы разрешающего столбца меняют знак на противоположный и делятся на разрешающий элемент.

Остальные элементы разрешающей строки делятся на разрешающий элемент.

Все остальные элементы симплексной таблицы (табл. 2.5) вычисляются по следующей формуле:

$$a_{ij} = \frac{a_{ij}a_{rl} - a_{rj}a_{il}}{a_{rl}} = a_{ij} - \frac{a_{rj}a_{il}}{a_{rl}}.$$

*Таблица 2.5.* **Вычисление элементов симплексной таблицы**

| | $-x_4$ | $-x_5$ | 1 |
|---|---|---|---|
| $x_1$ | 1 | −2 | 1 |
| $x_2$ | −2 | 1 | 2 |
| $x_3$ | 3 | 1 | 3 |
| $Q(\bar{x})$ | −1 | 1 | 0 |

| | $-x_4$ | $-x_2$ | 1 |
|---|---|---|---|
| $x_1$ | −3 | 2 | 5 |
| $x_5$ | −2 | 1 | 2 |
| $x_3$ | 5 | −1 | 1 |
| $Q(\bar{x})$ | 1 | −1 | −2 |

| | $-x_3$ | $-x_2$ | 1 |
|---|---|---|---|
| $x_1$ | 3/5 | 7/5 | 28/5 |
| $x_5$ | 2/5 | 3/5 | 12/5 |
| $x_4$ | 1/5 | −1/5 | 1/5 |
| $Q(\bar{x})$ | −1/5 | −4/5 | −11/5 |

**Пример 2.95.** Определить, в каком количестве надо выпускать продукцию четырех типов Прод1, Прод2, Прод3, Прод4, для изготовления которой требуются ресурсы трех видов: трудовые, сырье, финансы.

Количество ресурса каждого вида, необходимое для выпуска единицы продукции данного типа, называется нормой расхода.

Нормы расхода, наличие располагаемого ресурса, а также прибыль, получаемая от реализации единицы каждого типа продукции, приведены в табл. 2.6.

*Таблица 2.6.* **Исходная симплекс-таблица**

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Ресурс | Прод1 | Прод2 | Прод3 | Прод4 | Знак | Наличие |
| 2 | Прибыль | 60 | 70 | 120 | 130 | Макс | — |
| 3 | Трудовые | 1 | 1 | 1 | 1 | ≤ | 16 |
| 4 | Сырье | 6 | 5 | 4 | 3 | ≤ | 110 |
| 5 | Финансы | 4 | 6 | 10 | 13 | ≤ | 100 |

***Решение***

**1.** Составление *математической модели*. Для этого введем следующие обозначения:

$x_j$ — количество выпускаемой продукции $j$-го типа, $j = 1, 2, 3, 4$;

$b_i$ — количество располагаемого ресурса $i$-го вида, $i = 1, 2, 3$;

$a_{ij}$ — норма расхода $i$-го ресурса для выпуска единицы продукции $j$-го типа;

$c_j$ — прибыль, получаемая от реализации единицы продукции $j$-го типа.

Для выпуска единицы Прод1 требуется 6 единиц сырья, значит, для выпуска всей продукции Прод1 требуется $6x_1$ единиц сырья, где $x_1$ — количество выпускаемой продукции Прод1. С учетом того, что для других видов продукции зависимости аналогичны, ограничение по сырью будет иметь вид:

$$6x_1 + 5x_2 + 4x_3 + 3x_4 \le 110.$$

В этом ограничении левая часть равна величине требуемого ресурса, а правая показывает количество имеющегося ресурса.

Аналогично можно составить ограничения для остальных ресурсов и написать зависимость для целевой функции. Тогда математическая модель задачи будет иметь вид:

$$\begin{cases} F = 60x_1 + 70x_2 + 120x_3 + 130x_4 \to \max; \\ x_1 + x_2 + x_3 + x_4 \le 16; \\ 6x_1 + 5x_2 + 4x_3 + 3x_4 \le 110; \\ 4x_1 + 6x_2 + 10x_3 + 13x_4 \le 100; \\ x_j \ge 0;\ j = 1, 2, 3, 4. \end{cases} \qquad (2.24)$$

**2.** *Ввод исходных данных.* Ввести исходные данные в форму (табл. 2.7) согласно условию задачи.

*Таблица 2.7.* **Форма ввода данных**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | | | |
| 2 | Имя | Прод1 | Прод2 | Прод3 | Прод4 | | | |
| 3 | Значение | | | | | | | |
| 4 | Нижн. гр. | 0 | 0 | 0 | 0 | | | |
| 5 | Верх. гр. | | | | | ЦФ | Напр | |
| 6 | Коэф. в ЦФ | 60 | 70 | 120 | 130 | 0 | Макс | |
| 7 | *Ограничения* | | | | | | | |
| 8 | Вид ресурсов | | | | | Лев. часть | Знак | Прав. часть |
| 9 | Трудовые | 1 | 1 | 1 | 1 | 0 | ≤ | 16 |
| 10 | Сырье | 6 | 5 | 4 | 3 | 0 | ≤ | 110 |
| 11 | Финансы | 4 | 6 | 10 | 13 | 0 | ≤ | 100 |

**3.** *Ввод зависимостей из математической модели.* Ввести зависимости из математической модели (2.24). В режиме представления формул это будет выглядеть следующим образом (табл. 2.8).

*Таблица 2.8.* **Ввод зависимостей**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | | | |
| 2 | Имя | Прод1 | Прод2 | Прод3 | Прод4 | | | |
| 3 | Значение | | | | | | | |
| 4 | Нижн. гр. | 0 | 0 | 0 | 0 | | | |
| 5 | Верх. гр. | | | | | ЦФ | Напр | |
| 6 | Коэф. в ЦФ | 60 | 70 | 120 | 130 | =СУММПРОИЗВ (B$3:E$3;B6:E6) | Макс | |
| 7 | *Ограничения* | | | | | | | |
| 8 | Вид ресурсов | | | | | Лев. часть | Знак | Прав. часть |
| 9 | Трудовые | 1 | 1 | 1 | 1 | =СУММПРОИЗВ (B$3:E$3;B9:E9) | ≤ | 16 |
| 10 | Сырье | 6 | 5 | 4 | 3 | =СУММПРОИЗВ (B$3:E$3;B10:E10) | ≤ | 110 |
| 11 | Финансы | 4 | 6 | 10 | 13 | =СУММПРОИЗВ (B$3:E$3;B11:E11) | ≤ | 100 |

Назначить целевую функцию: $F$6. Ввести адреса искомых переменных: $B$3:$E$3.

Ввести граничные условия: $B$3:$E$3 >=$B$4:$E$4.

Ввести ограничения: $F$9:$F$11 <=$H$9:$H$11.

Параметры, используемые по умолчанию, подходят для решения большинства задач. Решение найдено и результат оптимального решения задачи приведен в табл. 2.9.

Из табл. 2.9 видно, что в оптимальном решении Прод1 = 10, Прод2 = 0, Прод3 = 6, Прод4 = 0. При этом максимальная прибыль будет составлять 1320, а количество использованных ресурсов: трудовых = 16, сырья = 84, финансов = 100.

Важным фактором, помогающим принять решение, является графическое представление полученного результата (рис. 2.44).

*Таблица 2.9.* **Оптимальное решение задачи**

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | | | |
| 2 | Имя | Прод1 | Прод2 | Прод3 | Прод4 | | | |
| 3 | Значение | 10 | 0 | 6 | 0 | | | |
| 4 | Нижн. гр. | | | | | | | |
| 5 | Верх. гр. | | | | | ЦФ | Направ-ление | |
| 6 | Коэф. в ЦФ | 60 | 70 | 120 | 130 | 1320 | Макс | |
| 7 | *Ограничения* | | | | | | | |
| 8 | Вид ресурсов | | | | | Лев. часть | Знак | Прав. часть |
| 9 | Трудовые | 1 | 1 | 1 | 1 | 16 | ≤ | 16 |
| 10 | Сырье | 6 | 5 | 4 | 3 | 84 | ≤ | 110 |
| 11 | Финансы | 4 | 6 | 10 | 13 | 100 | ≤ | 100 |

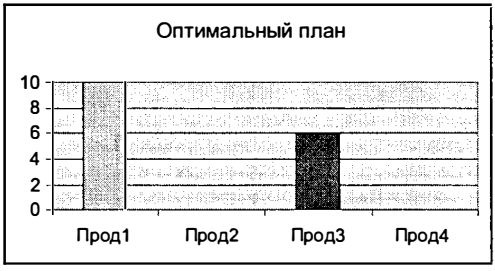



**Рис. 2.44.** Гистограмма оптимального решения

**Пример 2.96.** Компания производит два основных типа товара (Изделие-1, Изделие-2). Изделие-1 требует 2 ед. сырья *A* и 2 ед. сырья *B*, оно приносит прибыль компании 2 ден. единицы.

Изделие-2 требует 3 ед. сырья *A* и 5 ед. сырья *B*, оно приносит прибыль 4 ден. ед. Найдите оптимальный план производ-

ства, если доступно всего 1200 единиц сырья $A$ и 1600 единиц сырья $B$.

***Решение***

Пусть $X$ — количество производимого товара первого изделия, $Y$ — количество производимого товара второго изделия. Составим математическую модель.

$$\begin{cases} \text{ЦФ: } F(X, Y) = 2X + 4Y \to \max; \\ \quad\quad 2X + 2Y \to 1200; \\ \text{ОГР: } 2X + 5Y \to 1600; \\ \text{ГРУ: } X \geq 0;\ Y \geq 0. \end{cases}$$

Создадим форму для ввода условий задачи (табл. 2.10) и введем исходные данные.

*Таблица 2.10.* **Форма для ввода исходных данных**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | |
| 2 | Имя | Изделие-1 | Изделие-2 | | | |
| 3 | Значения | | | | | |
| 4 | Нижн. Гр. | 0 | 0 | | | |
| 5 | Верх. Гр. | | | ЦФ | | |
| 6 | Коэф. ЦФ | 2 | 4 | | | |
| 7 | *Ограничения* | | | | | |
| 8 | Вид | | | Левая часть | Знак | Правая часть |
| 9 | Сырье А | 2 | 3 | | ≤ | 1200 |
| 10 | Сырье Б | 2 | 5 | | ≤ | 1600 |

Введем зависимости из математической модели (табл. 2.11).

Назначить целевую функцию $D$6. Ввести адреса искомых переменных: $B$3:$C$3. Ввести ограничения и граничные условия: $D$9:$D$10 <= $F$9:$F$10, $B$3:$C$3 >= $B$4:$C$4.

Результат оптимального решения задачи приведен в табл. 2.12.

*Таблица 2.11.* **Ввод зависимостей**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | |
| 2 | Имя | Изделие-1 | Изделие-2 | | | |
| 3 | Значения | | | | | |
| 4 | Нижн. Гр. | 0 | 0 | | | |
| 5 | Верх. Гр. | | | ЦФ | | |
| 6 | Коэф. ЦФ | 2 | 4 | =СУММПРОИЗВ (B$3:C$3;B6:C6) | | |
| 7 | *Ограничения* | | | | | |
| 8 | Вид | | | Левая часть | Знак | Правая часть |
| 9 | Сырье А | 2 | 3 | =СУММПРОИЗВ (B$3:C$3;B9:C9) | ≤ | 1200 |
| 10 | Сырье Б | 2 | 5 | =СУММПРОИЗВ (B$3:C$3;B10:C10) | ≤ | 1600 |

*Таблица 2.12.* **Оптимальное решение**

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | *Переменные* | | | | | |
| 2 | Имя | Изделие-1 | Изделие-2 | | | |
| 3 | Значения | 300 | 200 | | | |
| 4 | Нижн. Гр. | 0 | 0 | | | |
| 5 | Верх. Гр. | | | ЦФ | | |
| 6 | Коэф. ЦФ | 2 | 4 | 1400 | | |
| 7 | *Ограничения* | | | | | |
| 8 | Вид | | | Левая часть | Знак | Правая часть |
| 9 | Сырье А | 2 | 3 | 1200 | ≤ | 1200 |
| 10 | Сырье Б | 2 | 5 | 1600 | ≤ | 1600 |

По полученным результатам построим гистограмму (рис. 2.45).

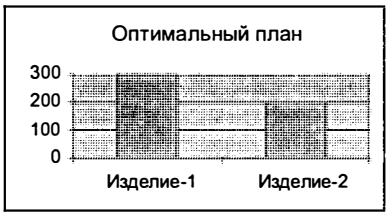



**Рис. 2.45.** Гистограмма оптимального решения

## Программа

```
Program Simpl;
Uses crt;
Const    n = 2;   m = 3;   Epsilon = 0.000001;
var
  VectorA : array [1..m, 0..m+n] of real;
  TargetVector : array [1..m+n] of real;
  SimplexVector : array [0..m+n] of real;
  DigitOfBasisVector : array [1..m] of real;
  BasisVector : array [1..m] of integer;
  IndexOfEnterVector : integer;
  IndexOfOutputString : integer;
  MinimumBuffer : real;
  key : char;   FileOfOutput : text;
{ Описание процедур }
procedure ReadDates; { считывание данных из файла }
  var  DateFile : text;
procedure ReadDatesTargetVector; { считывание данных целевого
вектора }
  var   i : integer;
  begin
  for i:=1 to n do Readln(DateFile, TargetVector[i]);
  end;
procedure ReadDatesVectorA; { считывание вектора A и заполнение
единицами  диагонали}
  var   i,j : integer;
begin
for j:=0 to n do
for i:=1 to m do Readln(DateFile, VectorA[i, j]);
```

```
  i:=1;
  for j:=n+1 to n+m do begin
  VectorA[i, j]:=1; inc(i)end;
end;
procedure ReadDatesBasisVector;
  var i : integer;
  begin
  for i:=1 to m do BasisVector[i]:=n+i;
  end;
begin
Assign(DateFile, 'kurs.dat'); Reset(DateFile);
ReadDatesTargetVector; ReadDatesVectorA;
ReadDatesBasisVector; Close(DateFile);
end;
procedure CountSimplexVector; { расчет симплекс-вектора }
var
  i,j : integer;  Summa : real;  Simplex : real;
begin SimplexVector[0]:=0;
for i:=1 to m do
SimplexVector[0]:=SimplexVector[0] + DigitOfBasisVector[i]*VectorA[i, 0];
for j:=1 to m+n do
begin  Summa:=0;
for i:=1 to m do Summa:=Summa + DigitOfBasisVector[i]*VectorA[i, j];
SimplexVector[j]:=Summa - TargetVector[j];
if abs(SimplexVector[j]) <= Epsilon then SimplexVector[j]:=0;
end;
end;
function GetEnterVector : integer; { поиск вводимого вектора }
var
  i : integer;  Min : real;
begin
GetEnterVector:=1;  Min:=SimplexVector[1];
for i:=2 to m+n do
if Min > SimplexVector[i]
then begin GetEnterVector:=i;  Min:=SimplexVector[i]; end;
end;
function GetOutputString : integer; { поиск выводимой строки }
var
  i : integer;  Temp : real;
begin
GetOutputString:=1;
if VectorA[1, IndexOfEnterVector] > 0 then
```

```
MinimumBuffer:=VectorA[1, 0] / VectorA[1, IndexOfEnterVector];
for i:=2 to m do
begin
Temp:=VectorA[i, 0] / VectorA[i, IndexOfEnterVector];
if Temp > 0 then  if MinimumBuffer >= Temp then
begin   MinimumBuffer:=Temp;   GetOutputString:=i;  end;
end;
end;
procedure ReCountOutputString; {пересчет коэф. выводимой строки }
  var  i,j : integer;   Buffer : real;
procedure ReCountDigitOfBasisVector;
begin
DigitOfBasisVector[IndexOfOutputString]:=
      TargetVector[IndexOfEnterVector];
end;
procedure ReCountBasisVector;
begin
BasisVector[IndexOfOutputString]:=IndexOfEnterVector;
end;
begin
ReCountDigitOfBasisVector;   ReCountBasisVector;
Buffer:=VectorA[IndexOfOutputString, IndexOfEnterVector];
for i:=0 to m+n do
begin
VectorA[IndexOfOutputString, i]:=VectorA[IndexOfOutputString, i] /
Buffer;
end;
end;
procedure ReCountVectorA;
  var    i,j : integer;
begin
  for j:=0 to m+n do
  begin
  for i:=1 to m do
  begin
  if i <> IndexOfOutputString then   if j <> IndexOfEnterVector
  then VectorA[i, j]:=VectorA[i, j] - VectorA
  [i,IndexOfEnterVector]*VectorA[IndexOfOutputString,j];
  end;  end;
for i:=1 to m do
if i <> IndexOfOutputString then VectorA[i, IndexOfEnterVector]:=0;
end;
function AllIsPositiv : boolean;
```

```
  var  i : integer;
  begin
  AllIsPositiv:=True;
  for i:=1 to m+n do
  if SimplexVector[i] < 0 then AllIsPositiv:=False;
  end;
function ToStr(const D : real) : string;
  var  S : string;
  begin
  str(D:6:2, S);   ToStr:=' ' + S + ' ';
  end;
procedure WriteMatrixs;
procedure WriteTargetMatrix;
  var  i : integer;
begin
  writeln(' +--------------------------------+');   write ('  Target ');
  for i:=1 to n+m do write(ToStr(TargetVector[i]),'');  writeln;
  end;
procedure WriteMatrixA;
  var  i,j : integer;
begin
writeln('
+---------------+--------+------+-------+-------+-------+-------+--------');
writeln('  Basis  D.Basis  A 0   A 1   A 2   A 3   A 4   A 5');
writeln('
+--------+--------+--------+--------+--------+--------+--------+--------');
for i:=1 to m do
begin
write('  A',BasisVector[i],' ',ToStr(DigitOfBasisVector[i]),'');
for j:=0 to m+n do write(ToStr(VectorA[i, j]),''); writeln;
if i = m then writeln('
+--------+--------+--------+--------+--------+--------+--------+--------')
else writeln('
+------+-------+--------+--------+-------+--------+--------+--------');
end;
end;
procedure WriteMatrixSimplex;
  var  i : integer;
begin
write('  Simplex');
for  i:=0 to  m+n do write(ToStr(SimplexVector[i]),'');  writeln;
```

```
writeln('
+-------------------------------------------------------------+');
end;
begin
clrscr;
WriteTargetMatrix;  WriteMatrixA;  WriteMatrixSimplex;
end;
procedure WriteMatrixsInFile;
procedure WriteTargetMatrix;
  var  i : integer;
begin
writeln(FileOfOutput, '
+-----------------------------------------------------+');
write (FileOfOutput, '  Target ');
for i:=1 to n+m do write(FileOfOutput, ToStr(TargetVector[i]),'');
writeln(FileOfOutput);
end;
procedure WriteMatrixA;
  var  i,j : integer;
begin
writeln(FileOfOutput, '
+----------------+--------+--------+--------+--------+--------+--------');
writeln(FileOfOutput, '  Basis  D.Basis  A 0  A 1  A 2  A 3  A 4  A 5
');
writeln(FileOfOutput, '
+--------+--------+--------+--------+--------+--------+--------+--------');
for i:=1 to m do
begin
write(FileOfOutput, '  A',BasisVector[i],'
',ToStr(DigitOfBasisVector[i]),'');
for j:=0 to m+n do write(FileOfOutput, ToStr(VectorA[i, j]),'');
writeln(FileOfOutput);
if i = m then writeln(FileOfOutput, '
+--------+--------+--------+--------+--------+--------+--------+--------')
else writeln(FileOfOutput, '
+--------+--------+--------+--------+--------+--------+--------+--------');
end;
end;
procedure WriteMatrixSimplex;
  var  i : integer;
begin
```

```
write(FileOfOutput, '  Simplex');
for i:=0 to m+n do write(FileOfOutput, ToStr(SimplexVector[i]),' ');
writeln(FileOfOutput);
writeln(FileOfOutput, '
+------------------------------------------------------------------+');
end;
begin
clrscr;
WriteTargetMatrix;  WriteMatrixA;  WriteMatrixSimplex;
end;
{ Основная программа }
BEGIN
ClrScr;  ReadDates;  Assign(FileOfOutput, 'kurs97.res');
Rewrite(FileOfOutput);  CountSimplexVector;  WriteMatrixs;
while not AllIsPositiv do
begin
IndexOfEnterVector:=GetEnterVector;
IndexOfOutputString:=GetOutputString;
ReCountOutputString;  ReCountVectorA;
CountSimplexVector;  WriteMatrixsInFile;  WriteMatrixs;
if key=#0  then  key:=readkey;  key:=#0;
end;
Close(FileOfOutput); ReadIn;
END.
```

В программе реализованны следующие процедуры:

1. ReadDates — считывание данных из файла.
2. ReadDatesTargetVector — считывание коэффициентов при неизвестных в целевой функции из файла.
3. ReadDatesVector — считывание входного файла матрицы A и заполнение диагональной матрицы.
4. CountSimplexVector — расчет симплекс-разностей.
5. GetEnterVector — поиск вводимого в базис столбца.
6. GetOutputString — поиск выводимой из базиса строки.
7. ReCountOutputString— пересчет выводимой строки.
8. ReCountVectorA — пересчет остальной матрицы ограничений.
9. WriteMatrixA, WriteTargetMatrix, WriteMatrixSimplex — печать результирующих таблиц и запись в файл.

## 2.12. Пакет Mathcad

В настоящее время научные и инженерные расчеты остаются одной из важнейших сфер приложения компьютеров. За многие годы накоплены обширные библиотеки научных подпрограмм и целый ряд различных математических пакетов, реализующих разнообразные численные методы, а также способных производить аналитические математические преобразования. Пожалуй, наиболее известными сегодня являются следующие пакеты: Mathematica (фирма Wolfram Research), Maple (фирма Waterloo Maple Inc), Matlab (фирма The MathWorks), Mathcad (фирма MathSoft Inc).

Пакет Mathematica является наиболее популярным в научных кругах, особенно среди теоретиков. Пакет предоставляет широкие возможности в проведении символических (аналитических) преобразований, однако требует значительных ресурсов компьютера.

Пакет Maple также весьма популярен в научных кругах. Кроме аналитических преобразований, пакет в состоянии решать задачи численно. Характерной особенностью пакета является то, что он позволяет конвертировать документы в формат LaTeX — стандартный формат подавляющего большинства научных издательств мирового класса. Кроме того, ряд других программных продуктов использует интегрированный символический процессор Maple.

Пакет Matlab фактически представляет собой своеобразный язык программирования высокого уровня, ориентированный на решение научных задач. Характерной особенностью пакета является то, что он позволяет сохранять документы в формате языка программирования C.

Пакет Mathcad более популярен в инженерной, чем в научной среде. Характерной особенностью пакета является использование привычных стандартных математических обозначений. Для использования пакета не требуется изучать какую-либо систему команд, как, например, в случае пакетов Mathematica или Maple. Пакет ориентирован в первую очередь на проведение численных расчетов, но имеет встроенный символический процессор Maple, что позволяет выполнять аналитические преобразования. В последних версиях предусмотрена возможность создавать связки документов Mathcad с документами Mathlab. В от-

личие от упомянутых выше пакетов, Mathcad является средой визуального программирования, т. е. не требует знания специфического набора команд.

### 2.12.1. Примеры выполнения расчетов в пакете Mathcad

**Работа с матрицами**

Простейшие операции матричной алгебры реализованы в Mathcad в виде операторов. Написание операторов по смыслу максимально приближено к их математическому действию. Каждый оператор выражается соответствующим символом. Векторы являются частным случаем матриц размерности $n \times 1$, поэтому для них справедливы все операции, что и для матриц, если ограничения особо не оговорены (например, некоторые операции применимы только к квадратным матрицам $n \times n$). Какие-то действия допустимы только для векторов (например, скалярное произведение), а какие-то, несмотря на одинаковое написание, по-разному действуют на векторы и матрицы. При работе с матрицами используется панель инструментов «Матрицы» (рис. 2.46).

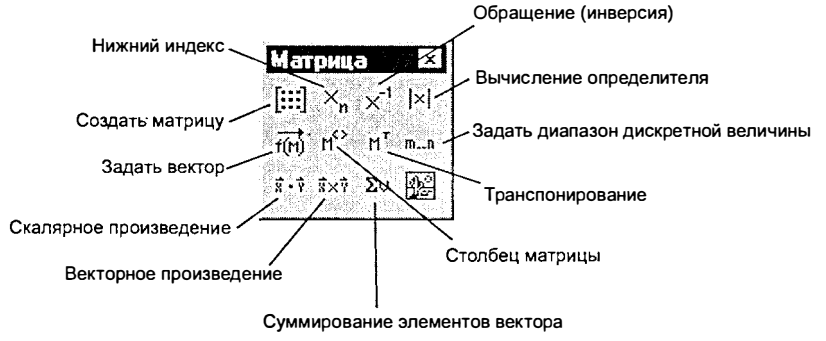



**Рис. 2.46.** Панель инструментов «Матрицы»

Для того чтобы выполнить какую-либо операцию с помощью панели инструментов, нужно:

- выделить матрицу и щелкнуть в панели по кнопке операции;
- щелкнуть по кнопке в панели и ввести в помеченной позиции имя матрицы.

Меню «Символы» содержит три операции — *транспонирование*, *инвертирование*, *определитель*. Например, чтобы вычислить определитель матрицы, можно выполнить команду *Символы — Матрицы — Определитель*.

### Нахождение нулей функций

Для определения значений, в которых функция обращается в нуль, используется встроенная функция **root($f(x)$, $x$)**. Первый аргумент — функция, нуль которой необходимо найти, второй — переменная, которую необходимо варьировать. Функция $f(x)$ может быть функцией многих переменных и необходимо указывать, по какой именно переменной отыскивается нуль функции. Кроме того, необходимо задать начальное приближение поиска. Точность вычислений задается встроенной переменной TOL. По умолчанию ее значение равно 0,001. Это значение можно изменить непосредственно в тексте документа: TOL := $10^{-9}$.

Например, зададим начальное приближение: x := –1.

После этого вычисляем корень: root(f(x), x) = –1.570796327.

Функция **r(x)** возвращает значение корня, ближайшее к $x$, т. е. к задаваемому, начальному приближению аргумента функции.

### Нахождение корней полиномов

Для нахождения корней полиномов имеется встроенная функция **polyroots($a$)**. Аргументом функции является вектор коэффициентов полинома

$$a_0 + a_1x + a_2x^2 + \ldots + a_nx^n.$$

Если в полиноме отсутствуют некоторые степени, то на соответствующих местах следует записать 0.

Например, пусть требуется найти корни полинома $p(x) = x^3 + 2x^2 - 1$.

$$a_1 := \begin{pmatrix} -1 \\ 0 \\ 2 \\ 1 \end{pmatrix}$$

i := 0..3

$$\text{polyroots(a)} = \begin{pmatrix} -1.618 \\ -1 \\ 0.618 \end{pmatrix}.$$

### Нахождение корней уравнений

Mathcad представляет ряд дополнительных возможностей для поиска корней уравнений. Функция **root(*f*(*var*1, *var*2, ...), *var*1, [*a*, *b*])** имеет два необязательных аргумента *a* и *b*, которые определяют границы интервала, на котором следует искать корень. На концах отрезка $[a; b]$ функция *f* должна менять знак ($f(a) f(b) < 0$). Задавать начальное приближение для корня не нужно. Местоположение корней определим графически.

Тогда функции для поиска корней уравнения будут иметь вид

$$\text{root}(f(x),x,-1,8) = 0; \qquad \text{root}(f(x),x,-10,-0.1) = -2.818.$$

### Решение систем уравнений и неравенств

Системы линейных и нелинейных уравнений и неравенств позволяет решать процедура ***given*** в сочетании с функцией ***Find***.

Система уравнений и/или неравенств должна быть записана после или правее слова ***given***. Перед словом ***given*** необходимо указывать начальные приближения для всех переменных.

Например, зададим начальные приближения и решим систему нелинейных уравнений

$$x := 1 \quad y := 1$$

$$\text{given } x^2 - y = 23 \quad x2 \cdot y = 50 \quad \text{Find}(x, y) = \begin{pmatrix} 5 \\ 2 \end{pmatrix}.$$

Решение системы линейных уравнений будет выглядеть следующим образом:

$$x := -2 \quad y := 1 \quad z := 2$$

$$\text{given } 2 \cdot x - 4 \cdot y + 3 \cdot z = 1 \quad x - 2 \cdot y + 4 \cdot z = 3 \quad 3 \cdot x - y + 5 \cdot z = 2$$

$$\text{Find}(x, y, z) = \begin{bmatrix} -1 \\ -3.904 \cdot 10^{-10} \\ 1 \end{bmatrix}.$$

### Интерполяция функций

Простейшим случаем локальной интерполяции является линейная интерполяция, когда в качестве интерполяционной функции выбирается полином первой степени.

Линейная интерполяция осуществляется с помощью встроенной функции **linterp**.

Например, требуется провести линейную интерполяцию функции sin(*x*) на интервале [0..6], используя пять узлов интерполяции, и вычислить значения функции в четырех точках $X_k$, тогда

$$k := 0..3$$

$X_k :=$

| |
|---|
| –0.5 |
| 1.111 |
| 2.333 |
| 4.574 |

Задаем интервал изменения *x* и число узловых точек

$$x_{min} := 0; \quad x_{max} := 6; \quad n := 5.$$

Определяем шаг изменения *x*:

$$h := \frac{(x_{max} - x_{min})}{n}.$$

Вычисляем координаты узлов и значения функции в них:

$$i := 0..n - 1; \quad x_i := x_{min} + ih; \quad y_i := \sin(x_i).$$

Проводим линейную интерполяцию: g(z) := linterp(x, y, z).

Вычисляем значение интерполяционной функции в заданных точках

| g ($X_k$) | sin ($X_k$) |
|---|---|
| –0.388 | –0.479 |
| 0.863 | 0.896 |
| 0,69 | 0.723 |
| –0.892 | –0,99 |

Как видно, результаты интерполяции отличаются от точных значений функции незначительно.

### Вычисление определенных интегралов

Для вычисления определенного интеграла необходимо выбрать знак интеграла из палитры или набрать его нажатием клавиши &. После этого следует задать пределы интегрирования, подынтегральную функцию и переменную интегрирования. Точность вычислений регулируется встроенной переменной TOL. По умолчанию ее значение установлено TOL := $10^{-3}$.

Зависимость результата от заданной точности вычислений представлена ниже:

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx = 3.14159265369356; \quad \int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx = 3.14159265358979.$$

### Статистические расчеты

Mathcad дает возможность обрабатывать статистическую информацию.

Создадим с помощью датчика случайных чисел **rnd** последовательность, подчиняющуюся равномерному закону распределения.

$$n := 999; \quad i := n; \quad xi := rnd(10).$$

Вычислим среднее значение

$$Xmean := mean(x); \quad Xmean = 4.966,$$

стандартное отклонение

$$\alpha := 0; \quad Xst = 2.885$$

и дисперсию

$$\sigma := var(x); \quad \sigma = 8.323.$$

## 2.13. Реализация численных методов на языке C++

**Пример 2.97. Метод дихотомии**

Дано уравнение $(0,2x)^3 = \cos x$.

Приближенно вычислить корни уравнения с точностью $\varepsilon = 10^{-5}$

***Решение***

Преобразуем уравнение $(0,2x)^3 - \cos x = 0$ следующим образом:

$$F(x) = (0,2x)^3 - \cos(x).$$

Для нахождения корней в численных методах используются первая и вторая производные.

Начало интервала $a = -100$. Конец интервала $b = +100$.

*Переменные и константы*

| Переменная | Идентификатор |
|---|---|
| Координата *X* для графика | x |
| Координата *Y* для графика | y |
| Начало отрезка | a |
| Конец отрезка | b |
| Значение функции в начале отрезка | Fa |
| Значение функции в конце отрезка | Fb |
| Счетчик количества корней | k |
| Счетчик количества итераций | f |
| Масштаб графика | Scale |

**Функции**

```
//Исходная функция.
double F (double x)
{
  return (0.2*x)*(0.2*x)*(0.2*x)-cos(x);
}
//-----------------------------------------------------------------
// Функция преобразованная
double Fi (double x)
{
  return acos((0.2*x)*(0.2*x)*(0.2*x));
}
//-----------------------------------------------------------------
```

```
//Производная функции.
double P (double x)
{
  return 0.4*(0.2*x)+sin(x);
}
//---------------------------------------------------------------------------
//Вторая производная функции.
double Pp (double x)
{
  return 0.08-cos(x);
}
//---------------------------------------------------------------------------
```

*Переменные и константы процедуры метода дихотомии*

| Переменная | Идентификатор |
|---|---|
| Длина отрезка | h |
| Середина отрезка | x0 |
| Значение функции в середине отрезка | Fx0 |
| Начало интервала | aa |
| Конец интервала | bb |
| Точность (по условию) = 0,00001 | e |
| Шаг отделения корней. По условию = 0,1 | hh |
| Переменная цикла | i |

**Программа**

```
//Решение методом дихотомии.
void __fastcall TForm1::NMetodDihotomClick(TObject *Sender)
{
  double h;     //Длина отрезка
  double x0;    //Середина отрезка
  double Fx0;   //Значение функции в середине отрезка
  double aa=Eda->Text.ToDouble(); //Начало интервала
  double bb=Edb->Text.ToDouble(); //Конец интервала
  double e =Ede->Text.ToDouble(); //Точность. По условию = 0.00001
  double hh=Edhh->Text.ToDouble();//Шаг отделения корней = 0.1
  Memo->Lines->Add("Метод дихотомии.");
```

```
//Отделение корней методом табулирования.
k=0;
for (long double i=aa; i<bb; i=i+hh){
  a=i;    b=i+hh;    Fa = F(a);    Fb = F(b);
  //Здесь находится  корень.
  if ((Fa<=0 & Fb>=0)^(Fa>=0 & Fb<=0)){
    k++;
    Memo->Lines->Add("Корень " +IntToStr(k));
    Memo->Lines->Add("a = " +FloatToStr(a));
    Memo->Lines->Add("b = " +FloatToStr(b));
    f=0;
    //Уточнение значения корня.
    do {
      x0=(a+b)/2;   Fa = F(a);   Fb = F(b);   Fx0 = F(x0);
      if ((Fa>0 & Fx0<=0)^(Fa<0 & Fx0>=0)){
        b=x0;
      }
      else if ((Fx0>=0 & Fb<0)^(Fx0<=0 & Fb>0)){
      a=x0;
      }
      else {
ShowMessage("На интервале ["+FloatToStr(a)+", "+FloatToStr(b)+"]
      решения нет.");
      break;
      }
      h=b-a;
      f++;
    } while (h>2*e);//Необходимая точность достигнута.
    // Ответ.
    Memo->Lines->Add("x = " +FloatToStr(x0));
    Memo->Lines->Add("Итераций = " + IntToStr(f));
    Memo->Lines->Add(" ");
    }
  }
  if (aa>bb) {
    Memo->Lines->Add("Точка ""a"" должна быть меньше ""b""!");
  }
  Memo->Lines->Add("Ok");
  Memo->Lines->Add(" ");
}//Решение методом дихотомии.
```

**Пример 2.98. Метод хорд**

Дано уравнение $(0,2x)^3 = \cos x$.

Приближенно вычислить корни уравнения с точностью $\varepsilon = 10^{-5}$.

***Решение***

*Переменные и константы процедуры метода хорд*

| Переменная | Идентификатор |
|---|---|
| Приближение к корню ($X_n$) | xn |
| Приближение к корню ($X_{n-1}$) | xn1 |
| Значение функции в приближении | Fxn1 |
| Значение первой производной a | Pa |
| Значение второй производной a | Ppa |
| Неподвижный конец | T |
| Значение функции в неподвижном конце | FT |
| Начало интервала | aa |
| Конец интервала | bb |
| Точность. По условию = 0,00001 | e |
| Шаг отделения корней. По условию = 0,1 | hh |
| Переменная цикла | i |
| Модуль xn1-xn0 | TestE |

**Программа**

```
//Решение методом хорд.
void __fastcall TForm1::NMetodHordClick(TObject *Sender)
{
  double xn;     //Приближение к корню (Xn).
  double xn1;    //Приближение к корню (Xn-1).
  double Fxn1;  //Значение функции в приближении.
  double Pa;     //Значение первой производной a.
  double Ppa;    //Значение второй производной a.
  double T;      //Неподвижный конец.
  double FT;     //Значение функции в неподвижном конце.
  double aa=Eda->Text.ToDouble(); //Начало интервала
  double bb=Edb->Text.ToDouble(); //Конец интервала
  double e =Ede->Text.ToDouble(); //Точность. По условию = 0.00001
```

```
double hh=Edhh->Text.ToDouble();//Шаг отделения корней = 0.1
double TestE; //Модуль xn1-xn0.
Memo->Lines->Add("Метод хорд.");
//Отделение корней методом табулирования.
k=0;
for (long double i=aa; i<bb; i=i+hh){
  a=i;     b=i+hh;     Fa = F(a);     Fb = F(b);
  // Здесь находится  корень.
  if ((Fa<=0 & Fb>=0)||(Fa>=0 & Fb<=0)){
    k++;
    Memo->Lines->Add("Корень " +IntToStr(k));
    Memo->Lines->Add("a = " +FloatToStr(a));
    Memo->Lines->Add("b = " +FloatToStr(b));
    f=0;
    if (a>b) {ShowMessage("Точка ""a"" должна быть меньше ""b""!");}
    //Определим неподвижный конец.
    Fa = F(a);     Pa = P(a);     Ppa = Pp(a);
    if ((Fa<0 & Pa>0 & Ppa>0)||(Fa>0 & Pa<0 & Ppa<0)){
      T=b;     xn1=a;
    }
    else if ((Fa>0 & Pa<0 & Ppa>0)||(Fa<0 & Pa>0 & Ppa<0)){
      T=a;     xn1=b;
    }
    else {
ShowMessage("На интервале ["+FloatToStr(a)+", "+FloatToStr(b)+"]
решения нет.");
    break;
    }
    FT = F(T);
    //Уточнение значения корня.
    do {
      if (f>0) {xn1=xn;}
      Fxn1 = F(xn1);
      xn=xn1-Fxn1*(T-xn1)/(FT-Fxn1);
      f++;
      TestE=xn-xn1;
      if (TestE<0) {TestE=TestE*(-1);} //Отбросим знак.
    } while (TestE>e || f<=1);//Необходимая точность достигнута.
  //Всё. Ответ.
  Memo->Lines->Add("x = " +FloatToStr(xn));
  Memo->Lines->Add("Итераций = " + IntToStr(f));
  Memo->Lines->Add(" ");
  }
}
```

```
  if (aa>bb) {Memo->Lines->Add("Точка ""a"" должна быть меньше
  ""b""!");}
  Memo->Lines->Add("Ok");
  Memo->Lines->Add(" ");
}//Решение методом хорд.
```

**Пример 2.99. Метод Ньютона**

Дано уравнение $(0,2x)^3 = \cos x$.

Приближенно вычислить корни уравнения с точностью $\varepsilon = 10^{-5}$.

***Решение***

*Переменные и константы процедуры метода Ньютона*

| Переменная | Идентификатор |
|---|---|
| Приближение к корню ($X_{n1}$) | xn1 |
| Приближение к корню ($X_{n0}$) | xn0 |
| Значение функции в приближении ($FX_{n0}$) | Fxn0 |
| Значение первой производной в приближении ($PX_{n0}$) | Pxn0 |
| Значение второй производной a | Ppa |
| Значение второй производной b | Ppb |
| Начало интервала | aa |
| Конец интервала | bb |
| Точность. По условию = 0,00001 | e |
| Шаг отделения корней. По условию = 0,1 | hh |
| Переменная цикла | i |
| Модуль xn1-xn0 | TestE |

**Программа**

```
//Решение методом Ньютона.
void __fastcall TForm1::NMetodNewtonClick(TObject *Sender)
{
  double xn1;   //Приближение к корню (Xn).
  double xn0;   //Приближение к корню (Xn0).
  double Fxn0;  //Значение функции в приближении (Xn0).
  double Pxn0;//Значение первой производной в приближении (Xn0).
```

```
double Ppa;  //Значение второй производной a.
double Ppb;  //Значение второй производной b.
double aa=Eda->Text.ToDouble();
double bb=Edb->Text.ToDouble();
double e=Ede->Text.ToDouble();
double hh=Edhh->Text.ToDouble(); //Шаг отделения корней = 0.1
double TestE; //Модуль xn1-xn0.
Memo->Lines->Add("Метод Ньютона.");
//Отделение корней методом табулирования.
k=0;
for (long double i=aa; i<bb; i=i+hh){
  a=i;    b=i+hh;    Fa = F(a);    Fb = F(b);
  // Здесь находится  корень.
  if ((Fa<=0 && Fb>=0)||(Fa>=0 && Fb<=0)){
    k++;
    Memo->Lines->Add("Корень " +IntToStr(k));
    Memo->Lines->Add("a = " +FloatToStr(a));
    Memo->Lines->Add("b = " +FloatToStr(b));
    f=0;
  if (a>b) {ShowMessage("Точка ""a"" должна быть меньше ""b""!");}
    //Уточнение значения корня.
    do {
    Fa = F(a);    Fb = F(b);    Ppa = Pp(a);    Ppb = Pp(b);
    if ((Fa * Ppa)>0){
      xn0=a;
    }
    else if ((Fb * Ppb)>0){
      xn0=b;
    }
    else {
 ShowMessage("На интервале ["+FloatToStr(a)+", "+FloatToStr(b)+"]
решения нет.");
      break;
    }
      Fxn0 = F(xn0);   Pxn0 = P(xn0;.  xn1=xn0-Fxn0/Pxn0;
      f++;
      if (xn0==a) {a=xn1;} else {b=xn1;}
      TestE=xn1-xn0;
      if (TestE<0) {TestE=TestE*(-1);} //Отбросим знак.
    } while (TestE>e || f<=1);//Необходимая точность достигнута.
```

```
    // Ответ.
    Memo->Lines->Add("x = " +FloatToStr(xn1));
  //    Memo->Lines->Add("x = " +FloatToStrF(xn1,ffGeneral,12,0));
    Memo->Lines->Add("Итераций = " + IntToStr(f));
    Memo->Lines->Add(" ");
    }
  }
  if (aa>bb) {
    Memo->Lines->Add("Точка ""a"" должна быть меньше ""b""!");
  }
  Memo->Lines->Add("Ok");
  Memo->Lines->Add(" ");
}//Решение методом Ньютона.
```

### Программа для решения кубических уравнений методом касательных (метод Ньютона)

Кубическое уравнение имеет вид: $a_1x^3 + a_2x^2 + a_3x + a_4 = 0$.

Исходное уравнение имеет вид: $x^3 + (-6)x^2 + (-9)x + (58) = 0$.

Первая производная имеет вид: $f'(x) = 3x^2 + (-12)x + (-9)$.

Вторая производная имеет вид: $f''(x) = 6x + (-12)$.

```
//Метод Ньютона для решения кубических уравнений
#include<math.h>
#include<iostream.h>
double a[4]={0}, b[3]={0}, c[2]={0}, prec=0.00000;
double minim=0, maxim=0;
void Hello(void); void Input();
void Derivative(); void Calculation();
double Calc_Fun(double);
double Calc_First(double);
double Calc_Second(double);
main(void)
{
    Hello(); Input(); Derivative(); Calculation();
    return 0;
}
void Hello(void)
{
  cout<<"Программа для решения кубических уравнений методом
  касательных (метод Ньютона).\n\n";
}
```

```
void Input()
{
    cout<<"Кубическое уравнение имеет вид"<<endl
      <<"a1*x^3+a2*x^2+a3*x+a4=0"<<endl<<endl;
    for (int i=0;i<4;i++)
    {
      cout<<"Введите значение коэффициента a["<<i+1<<"] : ";
      cin>>a[i];
    }
    cout<<endl<<"Необходимо указать интервал поиска
  решения."<<endl
      <<"Введите нижнюю границу поиска : ";
    cin>>minim;
    cout<<"Введите верхнюю границу поиска : ";
    cin>>maxim;
    while(minim==maxim||minim>maxim)
    {
      cout<<"\nНижняя граница должна быть меньше верхней и
          не может быть ей  равна."<<endl
        <<"Повторите ввод нижней границы : ";
        cin>>minim;
        cout<<"Повторите ввод верхней границы : ";
        cin>>maxim;
    }
    cout<<"Введите допустимую погрешность : ";
    cin>>prec;
}
void Derivative()
{
    b[0]=a[0]*3;  b[1]=a[1]*2;  b[2]=a[2];
    c[0]=b[0]*2;  c[1]=b[1];
    cout<<"\n\n\n"
    <<"Исходное уравнение имеет вид : \n\n"
  <<a[0]<<"x^3+("<<a[1]<<")x^2+("<<a[2]<<")x+("<<a[3]<<")=0\n\n"
    <<"Первая производная имеет вид : \n\n"
    <<"f'(x)="<<b[0]<<"x^2+("<<b[1]<<")x+("<<b[2]<<")\n\n"
    <<"Вторая производная имеет вид : \n\n"
    <<"f''(x)="<<c[0]<<"x+("<<c[1]<<")\n\n";
}
void Calculation()
{
double x=0, m=0;
    cout<<"-------------------------------------------------"<<endl
      <<"|    Xn     |    f(Xn)    |  |f(Xn)|/m   |"<<endl
```

```
    <<"-----------------------------------------------"<<endl;
    if (abs(Calc_Fun(minim))*abs(Calc_Second(minim))>0) x=minim;
    else x=maxim;
  if (Calc_First(minim)>Calc_First(maxim)) m=abs(Calc_First(maxim));
  else m=abs(Calc_First(minim));
  cout<<"|";
  cout.width(15);cout.precision(10);
  cout<<x;
  cout<<"|";
  cout.width(15);cout.precision(10);
  cout<<Calc_Fun(x);
  cout<<"|";
  cout.width(15);cout.precision(10);
  cout<<(fabs(Calc_Fun(x))/m);
  cout<<"|\n";
  while((fabs(Calc_Fun(x))/m)>prec)
  {
    x=(x-(Calc_Fun(x)/Calc_First(x)));
    cout<<"|";
    cout.width(15);cout.precision(10);
    cout<<x;
    cout<<"|";
    cout.width(15);cout.precision(10);
    cout<<Calc_Fun(x);
    cout<<"|";
    cout.width(15);cout.precision(10);
    cout<<fabs(Calc_Fun(x))/m;
    cout<<"|\n";
  }
  cout<<"-----------------------------------------------";
}
double Calc_Fun(double x)
{
    return (a[0]*x*x*x+a[1]*x*x+a[2]*x+a[3]);
}
double Calc_First(double x)
{
    return (b[0]*x*x+b[1]*x+b[2]);
}
double Calc_Second(double x)
{
    return (c[0]*x+c[1]);
}
```

**Пример 2.100. Метод смешанный**

Дано уравнение $(0,2x)^3 = \cos x$.

Приближенно вычислить корни уравнения с точностью $\varepsilon = 10^{-5}$.

***Решение***

*Переменные и константы процедуры смешанного метода*

| Переменная | Идентификатор |
|---|---|
| Приближение к корню ($X_{n0}$) | xn0 |
| Значение функции в приближении ($FX_{n0}$) | Fxn0 |
| Значение первой производной Pa | Pa |
| Значение первой производной Pb | Pb |
| Значение второй производной Ppa | Ppa |
| Начало интервала | aa |
| Конец интервала | bb |
| Точность. По условию = 0,00001 | e |
| Шаг отделения корней. По условию = 0,1 | hh |
| Переменная цикла | i |
| Модуль xn1-xn0 | TestE |

**Программа**

```
//Решение смешанным методом.
void __fastcall TForm1::NMetodMixClick(TObject *Sender)
{
  double xn0;   //Приближение к корню (Xn0).
  double Fxn0; //Значение функции в приближении (Xn0).
  double Pa;    //Значение первой производной a.
  double Pb;    //Значение первой производной b.
  double Ppa;   //Значение второй производной a.
  double aa=Eda->Text.ToDouble(); //Начало интервала
  double bb=Edb->Text.ToDouble(); //Конец интервала
  double e =Ede->Text.ToDouble(); //Точность.
                                  //  По условию = 0.00001
  double hh=Edhh->Text.ToDouble();//Шаг отделения корней.
                                  // По усл. = 0.1
  double TestE; //функция  mod.
```

```
  Memo->Lines->Add("Метод смешанный.");
  //Отделение корней методом табулирования.
  k=0;
  for (long double i=aa; i<bb; i=i+hh){
    a=i;    b=i+hh;    Fa = F(a);    Fb = F(b);
    //Тут есть корень.
    if ((Fa<=0 && Fb>=0)||(Fa>=0 && Fb<=0)){
      k++;
      Memo->Lines->Add("Корень " +IntToStr(k));
      Memo->Lines->Add("a = " +FloatToStr(a));
      Memo->Lines->Add("b = " +FloatToStr(b));
      f=0;
      do{
      Fa = F(a);  Fb = F(b);   Pa = P(a);   Pb = P(b);  Ppa = Pp(a);
      if (Fa * Ppa < 0){
        a = a - Fa * (b - a) / (Fb - Fa);    b = b - Fb / Pb;
      }
      else{
        b = b - Fb * (b - a) / (Fb - Fa);   a = a - Fa / Pa;
      }
      f++;
      //ShowMessage("Дальше?");
      TestE=b-a; //mod
      if (TestE<0) {TestE=TestE*(-1);} //Отбросим знак.
      } while (TestE>e);//Необходимая точность достигнута.
      xn0 = (a + b) / 2;
      //Ответ.
      Memo->Lines->Add("x = " +FloatToStr(xn0));
Memo->Lines->Add("f(x) = " +FloatToStrF(F(xn0),ffGeneral,11,12));
      Memo->Lines->Add("Итераций = " + IntToStr(f));
      Memo->Lines->Add(" ");
    }
  }
  Memo->Lines->Add("Ok");
  Memo->Lines->Add(" ");
}//Решение смешанным методом
```

**Пример 2.101. Метод итераций**

Дано уравнение $(0{,}2x)^3 = \cos x$.

Приближенно вычислить корни уравнения с точностью $\varepsilon = 10^{-5}$.

***Решение***

*Переменные и константы процедуры метода итераций*

| Переменная | Идентификатор |
|---|---|
| Приближение к корню ($X_{n0}$) | xn0 |
| Приближение к корню ($X_{n1}$) | xn1 |
| Начало интервала | aa |
| Конец интервала | bb |
| Точность. По условию = 0,00001 | e |
| Шаг отделения корней. По условию = 0,1 | hh |
| Переменная цикла | i |
| Знак функции фи | M |
| Модуль xn1-xn0 | TestE |

**Программа**

```
//Решение методом итераций.
void __fastcall TForm1::NMetodInteracClick(TObject *Sender)
{
  double xn0;   //Приближение к корню (Xn0).
  double xn1;    //Приближение к корню (Xn1).
  double aa=Eda->Text.ToDouble(); //Начало интервала
  double bb=Edb->Text.ToDouble(); //Конец интервала
  double e =Ede->Text.ToDouble(); //Точность. По условию = 0.00001
  double hh=Edhh->Text.ToDouble();//Шаг отделения корней.
                                  // По условию = 0.1
  double TestE; //
  double M = 1; //Знак функции.
  Memo->Lines->Add("Метод итераций.");
  //Отделение корней методом табулирования.
  k=0;
  for (long double i=aa; i<bb; i=i+hh){
    a=i;    b=i+hh;    Fa = F(a);    Fb = F(b);
    //Здесь находится  корень.
    if ((Fa<=0 && Fb>=0)||(Fa>=0 && Fb<=0)){
      k++;
      Memo->Lines->Add("Корень " +IntToStr(k));
      Memo->Lines->Add("a = " +FloatToStr(a));
```

```
      Memo->Lines->Add("b = " +FloatToStr(b));
      f=0;      xn0 = (b + a) / 2;      M = max (a,b);
      if (P(M)>0){M = 1;}
      else {M = -1;}
      do{
        xn1 = xn0;      xn0 = M * Fi(xn1);
        f++;
        //ShowMessage("Дальше?");
        TestE=xn0-xn1; //mod
        if (TestE<0) {TestE=TestE*(-1);} //Отбросим знак.
      } while (TestE>e);//Необходимая точность достигнута.
      //Ответ.
      Memo->Lines->Add("x = " +FloatToStr(xn0));
  Memo->Lines->Add("f(x) = " +FloatToStrF(F(xn0),ffGeneral,11,12));
      Memo->Lines->Add("Итераций = " + IntToStr(f));
      Memo->Lines->Add(" ");
    }
  }
  Memo->Lines->Add("Ok");
  Memo->Lines->Add(" ");
}//Решение методом итераций.
```

## Программа для решения кубических уравнений методом итераций

Кубическое уравнение имеет вид $a_1x^3 + a_2x^2 + a_3x + a_4 = 0$.

```
//Метод итераций для решения кубических уравнений
#include<math.h>
#include<iostream.h>
double a[4]={0}, b[3]={0},    prec=0.00000;
double minim=0, maxim=0;
void Hello(void);   void Input();
void Derivative();   void Calculation();
double Calc_Fun(double);   double Calc_First(double);
main(void)
{
  Hello();   Input();   Derivative();
  Calculation();
  return 0;
}
void Hello(void)
{
  cout<<"Программа для решения кубических уравнений методом
  итераций.\n\n";
}
```

```
void Input()
{
  cout<<"Кубическое уравнение имеет вид"<<endl
        <<"a1*x^3+a2*x^2+a3*x+a4=0"<<endl<<endl;
  for (int i=0;i<4;i++)
  {
    cout<<"Введите значение коэффициента a["<<i+1<<"] : ";
    cin>>a[i];
  }
  cout<<endl<<"Необходимо указать интервал поиска
    решения."<<endl
      <<"Введите нижнюю границу поиска : ";
  cin>>minim;
  cout<<"Введите верхнюю границу поиска : ";
  cin>>maxim;
  while(minim==maxim||minim>maxim)
  {
    cout<<"\nНижняя граница должна быть меньше верхней и не
      может быть ей  равна." <<endl
          <<"Повторите ввод нижней границы : ";
    cin>>minim;
    cout<<"Повторите ввод верхней границы : ";
    cin>>maxim;
  }
  cout<<"Введите допустимую погрешность : ";
  cin>>prec;
}
void Derivative()
{
  b[0]=a[0]*3;   b[1]=a[1]*2;   b[2]=a[2];
}
void Calculation()
{      .
  double x=0, x_old=0, m=0;
  cout<<"-------------------------------------------------"<<endl
      <<"|     Xn      |    f(Xn)    | X(n+1)-Xn   |"<<endl
  <<"-------------------------------------------------"<<endl;
  if(fabs(Calc_First(minim))>fabs(Calc_First(maxim)))
  m=x=x_old=minim;
  else m=x=x_old=maxim;
  m=fabs(1/Calc_First(m));
  cout<<"|";
  cout.width(15);cout.precision(10);
  cout<<x;
```

```
    cout<<"|";
    cout.width(15);cout.precision(10);
    cout<<Calc_Fun(x);
    cout<<"|              |\n";
    if(Calc_First(x)>0)
    {
      do
      {
        x_old=x;
        x=x_old-m*Calc_Fun(x_old);
      cout<<"|";
        cout.width(15);cout.precision(10);
      cout<<x;
        cout<<"|";
        cout.width(15);cout.precision(10);
        cout<<Calc_Fun(x);
      cout<<"|";
        cout.width(15);cout.precision(10);
        cout<<fabs( Calc_Fun(x) - Calc_Fun(x_old) );
      cout<<"|\n";
      }
      while(( fabs( Calc_Fun(x) - Calc_Fun(x_old) ) )>prec);
    }
    else
    {
      do
      {
        x_old=x;
        x=x_old+m*Calc_Fun(x_old);
      cout<<"|";
          cout.width(15);cout.precision(10);
      cout<<x;
          cout<<"|";
          cout.width(15);cout.precision(10);
          cout<<Calc_Fun(x);
      cout<<"|";
          cout.width(15);cout.precision(10);
          cout<<fabs( Calc_Fun(x) - Calc_Fun(x_old) );
      cout<<"|\n";
      }
      while(( fabs( Calc_Fun(x) - Calc_Fun(x_old) ) )>prec);
    }
    cout<<"-----------------------------------------------";
}
```

```
double Calc_Fun(double x)
{
    return (a[0]*x*x*x+a[1]*x*x+a[2]*x+a[3]);
}
double Calc_First(double x)
{
    return (b[0]*x*x+b[1]*x+b[2]);
}
```

**Пример 2.102. Метод Гаусса**

Решить систему линейных уравнений

$$\begin{cases} 1{,}00x_1 - 0{,}51x_2 + 0{,}12x_3 + 0{,}55x_4 = 0{,}12; \\ 0{,}12x_1 + 0{,}18x_2 - 0{,}22x_3 - 0{,}41x_4 = 0{,}13; \\ 0{,}22x_1 - 3{,}01x_2 + 0{,}31x_3 + 0{,}58x_4 = 1{,}00; \\ 1{,}00x_1 + 0{,}24x_2 - 3{,}05x_3 - 0{,}22x_4 = 3{,}41. \end{cases}$$

***Решение***

*Глобальные переменные и константы*

| Переменная | Идентификатор |
|---|---|
| Точность вычисления | e |
| Обрабатываемая матрица | a |
| Исходная матрица | b |
| Решение уравнения | X |
| Промежуточные данные | p |
| Переменные цикла | i, j, k |

**Программа**

```
//Решение методом Гаусса
void __fastcall TForm1::NGaussClick(TObject *Sender)
{
  double p=0;
  //Сделаем массивы одинаковыми.
  for (int i=0; i<4; ++i){
    for (int j=0; j<5; ++j){
      a[i][j] = b[i][j];
    }
  }
```

```
//Решение.
for (int i=0; i<=3; ++i){
  if (a[i][i]==0){  //0 на гл. диагонали меняем местами
    for (int k=i+1; k<=3; ++k){
    if (a[k][i]!=0) {
      for (int j=0; j<=4; ++j){
        p = a[i][j];
        a[i][j] = a[k][j];
        a[k][j] = p;
      }
    }
  }
}
for (int j=4; j>=i; --j){ //на гл. диагонали получаем 1
  a[i][j] = a[i][j] / a[i][i];
}
for (int k=i+1; k<=3; ++k){ //ниже гл. диагонали делаем нули
  for (int j=4; j>=i; --j){
    a[k][j]=a[k][j]-a[i][j]*a[k][i];
    }
  }
}
for (int i=3; i>=0; --i){ //корректируем ответы для получ.
                          //коэффициентов
  X[i] = a[i][4];
  for (int j=3; j>=i+1; --j){
    X[i]=X[i]-X[j]*a[i][j];
  }
}
//Вывод ответа.
Memo->Clear();
Memo->Lines->Add("Метод Гаусса.");
Memo->Lines->Add("Корни системы.");
for (int i=0; i<=3; i++){
  Memo->Lines->Add("X(" + IntToStr(i) + ")=" +
      FloatToStrF(X[i],ffFixed,12,12));
}
Memo->Lines->Add("Преобразованная матрица.");
for (int i=0; i<=3; i++){
  /*for (unsigned j=1; j<=5; j++){
  Memo->Lines->Add(a[i][j]);
} */
```

```
  Memo->Lines->Add(FloatToStrF(a[i][0],ffFixed,12,4)+"
  "+FloatToStrF(a[i][1],ffFixed,12,4)+"  "+
    FloatToStrF(a[i][2],ffFixed,12,4)+"
  "+FloatToStrF(a[i][3],ffFixed,12,4)+"
  "+FloatToStrF(a[i][4],ffFixed,12,4));
  }
}
```

**Пример 2.103. Метод Жордана**

Решить систему линейных уравнений

$$\begin{cases} 1{,}00x_1 - 0{,}51x_2 + 0{,}12x_3 + 0{,}55x_4 = 0{,}12; \\ 0{,}12x_1 + 0{,}18x_2 - 0{,}22x_3 - 0{,}41x_4 = 0{,}13; \\ 0{,}22x_1 - 3{,}01x_2 + 0{,}31x_3 + 0{,}58x_4 = 1{,}00; \\ 1{,}00x_1 + 0{,}24x_2 - 3{,}05x_3 - 0{,}22x_4 = 3{,}41. \end{cases}$$

**Программа**

```
//Решение методом Жордана
void __fastcall TForm1::NJordanClick(TObject *Sender)
{
  double p=0;
  //Сделаем массивы одинаковыми.
  for (int i=0; i<=3; ++i){
    for (int j=0; j<=4; ++j){
      a[i][j] = b[i][j];
    }
  }
  //Решение.
  for (int i=0; i<=3; ++i){
    if (a[i][i]==0){  //0 на гл. диагонали меняем местами
      for (int k=i+1; k<=3; ++k){
       if (a[k][i]!=0) {
         for (int j=0; j<=4; ++j){
           p = a[i][j];   a[i][j] = a[k][j];   a[k][j] = p;
         }
       }
    }
  }
  for (int j=4; j>=i; --j){ //на гл. диагонали получаем 1
    a[i][j] = a[i][j] / a[i][i];
  }
  for (int k=0; k<=3; ++k){ //над и под гл. диагональю  обнуляем
```

```
    if (i!=k) {
      for (int j=4; j>=i; --j){
        a[k][j]=a[k][j]-a[i][j]*a[k][i];
        }
      }
    }
  }
  for (int i=3; i>=0; --i){ //корректируем ответы для получ.
                            // коэффициентов
    X[i] = a[i][4];
    for (int j=3; j>=i+1; --j){
      X[i]=X[i]-X[j]*a[i][j];
    }
  }
  //Вывод ответа.
  Memo->Clear();
  Memo->Lines->Add("Метод Жордана.");
  Memo->Lines->Add("Корни системы.");
  for (unsigned i=0; i<4; i++){
    Memo->Lines->Add("X(" + IntToStr(i) + ")=" +
      FloatToStrF(X[i],ffFixed,12,12));
  }
  Memo->Lines->Add("Преобразованная матрица.");
  for (unsigned i=0; i<4; i++){
    /*for (unsigned j=1; j<=5; j++){
      Memo->Lines->Add(a[i][j]);
  } */
    Memo->Lines->Add(FloatToStrF(a[i][0],ffFixed,12,4)+"
"+FloatToStrF(a[i][1],ffFixed,12,4)+"  "+
    FloatToStrF(a[i][2],ffFixed,12,4)+"
  "+FloatToStrF(a[i][3],ffFixed,12,4)+"
  "+FloatToStrF(a[i][4],ffFixed,12,4));
  }
}
```

**Пример 2.104. Метод итераций**

Решить систему линейных уравнений

$$\begin{cases} 1{,}00x_1 - 0{,}51x_2 + 0{,}12x_3 + 0{,}55x_4 = 0{,}12; \\ 0{,}12x_1 + 0{,}18x_2 - 0{,}22x_3 - 0{,}41x_4 = 0{,}13; \\ 0{,}22x_1 - 3{,}01x_2 + 0{,}31x_3 + 0{,}58x_4 = 1{,}00; \\ 1{,}00x_1 + 0{,}24x_2 - 3{,}05x_3 - 0{,}22x_4 = 3{,}41. \end{cases}$$

*Решение*

*Переменные и константы процедуры*

| Переменная | Идентификатор |
|---|---|
| Транспонированная матрица | at |
| Промежуточные данные | p |
| Переменные цикла | i, j, k |
| Количество итераций | it |

**Программа**

```
//Решение методом итераций
void __fastcall TForm1::NIterClick(TObject *Sender)
{
  double xt[4];  double at[4][4];  double p=0;
  //Сделаем массивы одинаковыми.
  for (int i=0; i<=3; ++i){
    for (int j=0; j<=3; ++j){
      at[i][j] = b[j][i];//С транспонированием
    }
  }
  //Решение.
  for (int i=0; i<=3; ++i){ //произведение трансп. и исходной
    for (int k=0; k<=3; ++k){ //
      a[i][k]=0;
      for (int j=0; j<=3; ++j){
      a[i][k] = a[i][k] + at[i][j] * b[j][k];
    }
  }
  a[i][4]=0;
  for (int j=0; j<=3; ++j){
      a[i][4]=a[i][4]+at[i][j]*b[j][4];
    }
  }
  for (int i=0; i<=3; ++i){ //
    p=a[i][i];
    for (int j=0; j<=4; ++j){ //
      a[i][j]=a[i][j]/p;
    }
  }
```

```
  X[0]=1; X[1]=1; X[2]=1; X[3]=1;
  int it=0;
  do{
    for (int i=0; i<=3; ++i){ //
      xt[i]=StrSum(i);
    }
    for (int i=0; i<=3; ++i){ //
      X[i]=xt[i];
    }
    it++;
    }while (GetE()>e);
    //Вывод ответа.
    Memo->Clear();
    Memo->Lines->Add("Метод итераций.");
    Memo->Lines->Add("Количество итераций = "+IntToStr(it));
    Memo->Lines->Add("Корни системы.");
    for (unsigned i=0; i<4; i++){
    Memo->Lines->Add("X(" + IntToStr(i) + ")=" +
        FloatToStrF(X[i],ffFixed,12,12));
    }
    Memo->Lines->Add("Преобразованная матрица.");
    for (unsigned i=0; i<4; i++){
      /*for (unsigned j=1; j<=5; j++){
        Memo->Lines->Add(a[i][j]);
      } */
      Memo->Lines->Add(FloatToStrF(a[i][0],ffFixed,12,4)+"
  "+FloatToStrF(a[i][1],ffFixed,12,4)+"  "+
      FloatToStrF(a[i][2],ffFixed,12,4)+"
    "+FloatToStrF(a[i][3],ffFixed,12,4)+"
    "+FloatToStrF(a[i][4],ffFixed,12,4));
  }
}
```

**Пример 2.105. Метод Зейделя**

Решить систему линейных уравнений

$$\begin{cases} 1{,}00x_1 - 0{,}51x_2 + 0{,}12x_3 + 0{,}55x_4 = 0{,}12; \\ 0{,}12x_1 + 0{,}18x_2 - 0{,}22x_3 - 0{,}41x_4 = 0{,}13; \\ 0{,}22x_1 - 3{,}01x_2 + 0{,}31x_3 + 0{,}58x_4 = 1{,}00; \\ 1{,}00x_1 + 0{,}24x_2 - 3{,}05x_3 - 0{,}22x_4 = 3{,}41. \end{cases}$$

*Решение*

*Переменные и константы процедуры*

| Переменная | Идентификатор |
|---|---|
| Транспонированная матрица | at |
| Промежуточные данные | p |
| Переменные цикла | i, j, k |
| Количество итераций | it |
| Сумма погрешностей | s |
| Итоговая погрешность | ss |
| Переменные цикла | i, j |

**Программа**

```
//Функция вычисления максимальной погрешности
double GetE()
{
  double s;  double ss;
  for (int i=0; i<=3; ++i){
    s=0;
    for (int j=0; j<=3; ++j){
      s = s+X[j]* b[i][j];
    }
    //s=abs(s-b[i][4]);
    s=s-b[i][4];
    if (i==0){
      ss=s;
    }
    else if (s>ss){
      ss=s;
    }
  }
  return ss;
}
//Функция вычисления суммы
double StrSum(int i)
{
  double s=0;    s = a[i][4];
  for (int j=0; j<=3; ++j){
    if (i!=j){
      s=s-X[j]*a[i][j];
```

```
    }
  }
  return s;
}
//Решение методом Зейделя
void __fastcall TForm1::NZeidelClick(TObject *Sender)
{
  double at[4][4];   double p=0;
  //Сделаем массивы одинаковыми.
  for (int i=0; i<=3; ++i){
    for (int j=0; j<=3; ++j){
      at[i][j] = b[j][i]; //с транспонированием
    }
  }
  //Решение.
  for (int i=0; i<=3; ++i){ //
    for (int k=0; k<=3; ++k){ //
      a[i][k]=0;
      for (int j=0; j<=3; ++j){
        a[i][k] = a[i][k] + at[i][j] * b[j][k];
      }
    }
    a[i][4]=0;
    for (int j=0; j<=3; ++j){
      a[i][4]=a[i][4]+at[i][j]*b[j][4];
      }
    }
  for (int i=0; i<=3; ++i){ //
    p=a[i][i];
    for (int j=0; j<=4; ++j){ //
      a[i][j]=a[i][j]/p;
    }
  }
  X[0]=1;  X[1]=1;  X[2]=1;  X[3]=1;
  int it=0;
  do{
    for (int i=0; i<=3; ++i){ //
      X[i]=StrSum(i);
    }
    it++;
  }while (GetE()>e);
  //Вывод ответа.
  Memo->Clear();
  Memo->Lines->Add("Метод Зейделя.");
```

```
  Memo->Lines->Add("Количество итераций = "+IntToStr(it));
  Memo->Lines->Add("Корни системы.");
  for (unsigned i=0; i<4; i++){
    Memo->Lines->Add("X(" + IntToStr(i) + ")=" +
        FloatToStrF(X[i],ffFixed,12,12));
  }
  Memo->Lines->Add("Преобразованная матрица.");
  for (unsigned i=0; i<4; i++){
      Memo->Lines->Add(FloatToStrF(a[i][0],ffFixed,12,4)+"
  "+FloatToStrF(a[i][1],ffFixed,12,4)+"  "+
      FloatToStrF(a[i][2],ffFixed,12,4)+"
  "+FloatToStrF(a[i][3],ffFixed,12,4)+"
  "+FloatToStrF(a[i][4],ffFixed,12,4));
  }
}
```

***Глобальный модуль***

```
double const e = 0.00001;
double a[4][5];
double b[4][5] = {
    {1.15, 0.62,-0.83, 0.92, 2.15},
    {0.82,-0.54, 0.43,-0.25, 0.62},
    {0.24, 1.15,-0.33, 1.42,-0.62},
    {0.73,-0.81, 1.27,-0.67, 0.88}
  };
  double X[4];
#include "ap.h"
/*-----------------------------------------------
This routines must be defined by the programmer:

double det(ap::real_2d_array a, int n, double epsilon);
-----------------------------------------------*/
bool leskramersolve(ap::real_2d_array a,
    const int& n,
    ap::real_1d_array& x,
    const double& epsilon);

/*************************************************
Функция решения системы линейных уравнений вида:
A[1,1]*X[1] + .. + A[1,N]*X[N] = A[1,N+1]
      .....................
A[N,1]*X[1] + .. + A[N,N]*X[N] = A[N,N+1]

Функция возвращает True, если  det(A')<>0
```

(X — решение), и False, если det(A')=0, в таком случае X не хранит решение. A' — матрица из первых N столбцов матрицы A.
Epsilon определяет точность сравнения чисел. Если число по модулю меньше или равно Epsilon, то оно считается равным 0. Это число показывает, насколько именно переданная матрица должна быть близка к вырожденной, чтобы вызвать выход со значением True. Чтобы проводить вычисления независимо от того, насколько матрица близка к вырожденной, следует задать epsilon равным 0.
**********/

```
bool leskramersolve(ap::real_2d_array a,
    const int& n,
    ap::real_1d_array& x,
    const double& epsilon)
{
    bool result;
    int i;    int j;    double d;    double dt;
    ap::real_2d_array b;
    x.setbounds(1, n);   b.setbounds(1, n, 1, n);   result = false;
    for(i = 1; i <= n; i++)
    {
      for(j = 1; j <= n; j++)
      {
        b(i,j) = a(i,j);
    }
  }
  d = det(b, n, epsilon);
  if( fabs(d)>epsilon )
  {
    result = true;
    for(i = 1; i <= n; i++)
    {
      for(j = 1; j <= n; j++)
      {
        b(i,j) = a(i,j);
      }
    }
    for(i = 1; i <= n; i++)
      {
        b(i,1) = a(i,n+1);
      }
    for(i = 2; i <= n; i++)
    {
      dt = det(b, n, epsilon);    x(i-1) = dt/d;
```

```
        for(j = 1; j <= n; j++)
        {
          b(j,i-1) = a(j,i-1);     b(j,i) = a(j,n+1);
        }
      }
      dt = det(b, n, epsilon);    x(n) = dt/d;
    }
    return result;
}
```

**Пример 2.106. Интерполирование функций**

С помощью одной из интерполяционных формул Ньютона вычислить значение функции, заданной таблично.

***Решение***

*Переменные и константы*

| Переменная | Идентификатор |
|---|---|
| Таблица значений X | x0 |
| Таблица значений Y | y0 |
| Точка поиска | x |
| Таблица конечных разностей | y |
| Сумма многочлена | s |
| Значение произведений | p |
| Факториал | k |

**Программа**

```
//Решение методом Лагранжа
void __fastcall TForm1::NLagrClick(TObject *Sender)
{
  Memo->Lines->Add("Метод Лагранжа.");
  s = 0;
  for (int i=0; i<=5; ++i){
    p = 1;
    for (int j=0; j<=5; ++j){
      if (j!=i){
        p = p * (x - x0[j]) / (x0[i] - x0[j]);
      }
    }
    p = p * y0[i];    s = s + p;
  }
```

```
  Memo->Lines->Add("==========");
  Memo->Lines->Add("Y=" + FloatToStrF(s,ffFixed,12,9));
  Memo->Lines->Add("OK");
  Memo->Lines->Add("");
}
```

*Глобальный модуль*

```
double x0[6] = {2.15, 2.20, 2.25, 2.30, 2.35, 2.40};
double y0[6] = {1.263,1.288,1.315,1.343,1.373,1.405};
double x = 2.36;  double y[6];  double s;  double p;  int k;

//Решение методом Ньютона
void __fastcall TForm1::NNewtonClick(TObject *Sender)
{
  double h;  double t;  int k;
  Memo->Lines->Add("Метод Ньютона.");
  h = x0[2] - x0[1];
  for (int i=0; i<=5; ++i){
    y[i] = y0[i];
  }
  Memo->Lines->Add("Конечные разности.");
  for (int i=4; i>=0; --i){
    Memo->Lines->Add("--- d" + IntToStr(5-i) + "y ---");
    for (int j=0; j<=i; ++j){
      y[j] = y[j + 1] - y[j];
      Memo->Lines->Add("d(" + IntToStr(j) + ")=" +
        FloatToStrF(y[j],ffFixed,12,3));
    }
  }
  //Интерполяция назад
  t=(x-x0[5])/h;    k=0;    s=y[5];    p=1;
  for (int i=4; i>=0; --i){
    p = p * t++ / ++k;    s = s + p * y[i];
  }
  Memo->Lines->Add("==========");
  Memo->Lines->Add("Y=" + FloatToStrF(s,ffFixed,12,9));
  Memo->Lines->Add("OK");
  Memo->Lines->Add("");
}

#include "ap.h"
double lagrangeinterpolate(int n,
    const ap::real_1d_array& x,
    ap::real_1d_array f,
    double t);
```

```
/************************************
Полином Лагранжа по Эйтекену.
function LagrangeInterpolate(
    const  n  :  Integer;
    const  x  :  array of Real;
           F  :  array of Real;
           t  :  Real):Real;
Параметры:
    N      — число точек в массиве
    X, F   — массивы абсцисс и ординат с номерами от 0 до N-1.
    T      — параметр, положение точки на кривой.
Результат:
Значение интерполяционного полинома, проходящего через
точки (x[i], F[i]).
************************************/
double lagrangeinterpolate(int n,
    const ap::real_1d_array& x,
    ap::real_1d_array f,
    double t)
{
  double result;   int i;   int j;   n = n-1;
  for(j = 0; j <= n-1; j++)
  {
    for(i = j+1; i <= n; i++)
    {
      f(i) = ((t-x(j))*f(i)-(t-x(i))*f(j))/(x(i)-x(j));
    }
  }
  result = f(n);
  return result;
}
```

**Пример 2.107.** Решение обыкновенных дифференциальных уравнений.

Решение обыкновенных дифференциальных уравнений первого порядка методом Рунге — Кутта (при $h = 16$).

**Программа**

```
double f(double x, double y)
{
  return x*exp(y); //Here you can insert your function like dy/dx = f(x,y);
}
```

```
void rgkt(double *xm, double *ym, double a, double b, double y0,
double dy0, int n)
{
    double x = a, y = y0, h = (b - a)/n;
    double q1, q2, q3, q4; ym[0] = y0;
    for(int i = 0; i < n; i++)
    {
      q1 = f(x,y);    q2 = f(x + h/2,y + h*q1/2);
      q3 = f(x + h/2,y + h*q2/2);    q4 = f(x + h/2,y + h*q3);
      ym[i+1] = ym[i] + h*(q1 + 2*q2 + 2*q3 + q4)/6;
      xm[i] = x;     x += h;
    }
}
double max(double *y, int n)
{
    double max = -10*exp(10);
    for(int i=0; i<n; i++)
    {
      if(y[i] > max) max = y[i];
    }
    return max;
}
int main(int argc, char* argv[])
{
    int n = 10; //Number of steps
    double a = 0, b = 1; //Solving area
    double y0 = 0, dy0 = 3.56; //Initial values
    double *x = new double[n];
    double *y = new double[n];
    double *x1 = new double[n];
    double *y1 = new double[n];
    rgkt(x, y, a, b, y0,dy0,n); rgkt(x1, y1, a, b, y0,dy0,n*2);
    double delta = fabs(max(y, n) - max(y1,n*2))/15;
    printf("%10s %10s\n", "x   ", "f(x)  ");
    for(int i=0; i<n; i++)
    {
        printf("%10f %10f\n", x[i], y[i]);
    }
    printf("\n  Absolute inaccuracy is equal: %f\n\n", delta);
    return 0;
}
```

# Глава 3
# ПРАКТИКУМ ПО ЧИСЛЕННЫМ МЕТОДАМ

## Варианты заданий для самостоятельной работы

### *Задание 1. Составить таблицы значений функций*

| Номер варианта | Сформировать таблицу |
|---|---|
| **1** | Значений функции $y = 2\sin(3x - 0{,}8)$ для $x$ на отрезке $[-1{,}57; 1{,}57]$ с шагом 0,25 |
| **2** | Перевода английских фунтов (*lb*) в метрические килограммы для значений от 1 до 20 *lb* с шагом 0,5 *lb* (1 *lb* = 0,454 кг) |
| **3** | Значений функции $y = e^{2-3x-x^2}$ для $x$ на отрезке $[-1{,}43; 1{,}17]$ с шагом 0,25 |
| **4** | Перевода километров в американские мили (*mi*) для расстояний от 1 до 20 км с шагом 0,5 км (1 км = 0,62137 *mi*) |
| **5** | Значений функции $y = \ln(2x^2 + 3x - 5)$ для $x$ на отрезке $[1{,}05; 2{,}10]$ с шагом 0,05 |
| **6** | Перевода температуры по шкале Цельсия (С) в температуру по шкале Фаренгейта для температур от –5 до 30 °С с шагом 2 °С (1 °F = 1,8 °C + 32) |
| **7** | Значений функции $y = 2e^{-x}\cos(1{,}7x - 0{,}3)$ для значений $x$ на отрезке $[-1{,}5; 1{,}5]$ с шагом 0,1 |
| **8** | Соответствия между метрической массой в граммах и английским фунтом (*lb*) для *m* от 5 до 20 *lb* с шагом 2 *lb* (1 *lb* = 453,59 г) |
| **9** | Значений функции $y = -3x^2 + 7x - 2$ для $x$ на отрезке $[-0{,}12; 0{,}71]$ с шагом 0,05 |
| **10** | Перевода английских дюймов (*in*) в сантиметры для значений от 10 до 100 *in* с шагом 5 *in* (1 *in* = 2,54 см) |

### *Задание 2. Построить графики функций*

**1.** $y = x^2 + \sin x$.
**2.** $y = 2x^2 + 13$.
**3.** $y = x^2 \cos 2x$.
**4.** $y = 7x - x^2 - 10$.
**5.** $y = \sqrt{4 - 3x}$.
**6.** $y = \cos^2 x - \sin^2 x$.
**7.** $y = -e^x - 1$
**8.** $y = \dfrac{x^2}{x - 2}$.
**9.** $y = 4x - x^2$.
**10.** $y = \cos \pi x + 1$.

### *Задание 3. Построить графики двух функций*

**1.** $y = \ln(x + 6)$, $y = 3 \ln x$.
**2.** $y = 6x^2 - 5x + 1$, $y = \cos \pi x$.
**3.** $y = x - 2$, $y = x^2 - 2x$.
**4.** $y = \cos \dfrac{1}{x}$, $y = x^2 - 2x$.
**5.** $y = x^2 + 1$, $y = 2 \cos x$.
**6.** $y = \sqrt{x}$, $y = \sqrt{4 - 3x}$.
**7.** $y = 4x - x^2$, $y = x^2 - 4x + 2$.
**8.** $y = 21x + 4$, $y = 2 \sin x$.
**9.** $y = x^3$, $y = \dfrac{\sin x}{x}$.
**10.** $y = x^2 - 6$, $y = -e^x$.

### *Задание 4. Найти производную функции в произвольной точке*

**1.** $y = \ln(\sqrt{1 + x^2} + x)$.
**2.** $y = \ln \operatorname{tg} \dfrac{x}{2}$.
**3.** $y = x \lg x$.
**4.** $y = \dfrac{x^2}{\sqrt{1 + x^2}}$. .
**5.** $y = (1 + \sqrt[3]{x})^3$.
**6.** $y = \dfrac{2 \cos x}{\sqrt{\cos^2 x}}$.
**7.** $y = \cos 2x \lg x$.
**8.** $y = \left(\sqrt{x} + \dfrac{1}{\sqrt{x}}\right)^{10}$.
**9.** $y = \dfrac{2 \cos x}{\sqrt{\cos 2x}}$.
**10.** $y = e^x \sin x \cos^3 x$.

### *Задание 5. Построить график функции $y = f(x)$ и касательную к графику в точке с абсциссой $x = x_0$.*

*Замечание.* Уравнение касательной $Y = f(x_0)(x - x_0) + f(x_0)$.

**1.** $f(x) = \dfrac{1}{x^4} + 2$, $x_0 = 1$.
**2.** $f(x) = \sqrt{x^2 + 1}$, $x_0 = 2$.
**3.** $f(x) = x \ln x$, $x_0 = 2$.
**4.** $f(x) = x^2 + 1$, $x_0 = -1$.
**5.** $f(x) = -x^2 + 1$, $x_0 = 1$.
**6.** $f(x) = \dfrac{1}{2} \sin^2\left(4x - \dfrac{\pi}{3}\right)$, $x_0 = \pi/6$;
**7.** $f(x) = x^2 - 2x - 8$, $x_0 = -1$.
**8.** $f(x) = \cos x$, $x_0 = -\pi/2$.
**9.** $f(x) = x^2 - 3x + 2$, $x_0 = 3$.
**10.** $f(x) = e^{2x+3}$, $x_0 = -2$.

## Задание 6. Решить системы линейных уравнений вида $Ax = B$ и $Cx = D$.

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 1 | 1 | 0,47 | −0,11 | 0,55 | 1,33 | 1 | 2 | 3 | 13 |
| | 0,42 | 1 | 0,35 | 0,17 | 1,29 | 2 | 3 | 5 | 4 |
| | −0,25 | 0,67 | 1 | 0,36 | 2,11 | 3 | 5 | 9 | 17 |
| | 0,54 | −0,32 | −0,74 | 1 | 0,10 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 2 | 0,63 | 1 | 0,11 | 0,34 | 2,08 | 1 | 2 | 3 | 0,55 |
| | 0,17 | 1,18 | −0,45 | 0,11 | 0,17 | 1 | 4 | 9 | 1,35 |
| | 0,31 | −0,15 | 1,17 | −2,35 | 1,28 | 1 | 8 | 27 | 3,55 |
| | 0,58 | 0,21 | −3,45 | −1,18 | 0,05 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 3 | 0,77 | 0,04 | −0,21 | 0,18 | 1,24 | 0,42 | 1,43 | 0,27 | 1 |
| | −0,45 | 1,23 | −0,06 | 0 | −0,88 | 1,43 | −0,84 | 0,93 | 2 |
| | −0,26 | −0,34 | 1,11 | 0 | 0,62 | 0,27 | 0,93 | −0,48 | 3 |
| | −0,05 | 0,26 | −0,34 | 1,12 | −1,17 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 4 | 0,79 | −0,12 | 0,34 | 0,16 | −0,64 | 0,64 | 0,54 | −0,33 | 3 |
| | -0,34 | 1,18 | −0,17 | 0,18 | 1,42 | 0,54 | −0,92 | 0,24 | 2 |
| | −0,16 | −0,34 | 0,85 | 0,31 | −0,42 | −0,33 | 0,24 | 0,78 | 1 |
| | −0,12 | 0,26 | 0,08 | 0,75 | 0,83 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 5 | −0,68 | −0,18 | 0,02 | 0,21 | −1,83 | 0,5 | 1,77 | 0,39 | 1,5 |
| | 0,16 | −0,88 | −0,14 | 0,27 | 0,65 | 0,84 | 1,79 | 0,95 | 2,5 |
| | 0,37 | 0,27 | −1,02 | −0,24 | −2,23 | 0,24 | 1,03 | −0,41 | 3 |
| | 0,12 | 0,21 | −0,18 | −0,75 | 1,13 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 6 | –0,58 | –0,32 | 0,03 | 0 | –0,44 | 0,19 | 0,51 | 0,86 | 0,35 |
| | 0,11 | –1,26 | –0,36 | 0 | –1,42 | 0,51 | 0,32 | 0,95 | 0,42 |
| | 0,12 | 0,08 | –1,14 | –0,24 | 0,83 | 0,86 | 0,95 | –0,12 | 0,45 |
| | 0,15 | –0,35 | –0,18 | 0 | 1,42 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 7 | –0,83 | 0,31 | -0,18 | 0,22 | 1,71 | 0,64 | 1,54 | –0,33 | 0,3 |
| | –0,21 | –0,67 | 0 | 0,22 | –0,62 | 1,54 | –0,92 | 0,24 | 0,2 |
| | 0,32 | –0,18 | –0,95 | –0,19 | 0,89 | –0,33 | 0,24 | 0,78 | 0,1 |
| | 0,12 | 0,28 | –0,14 | –1 | –0,94 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 8 | –0,87 | 0,27 | –0,22 | –0,18 | –1,21 | 0,55 | 1,77 | 1,39 | 1,5 |
| | –0,21 | –1 | –0,45 | 0,18 | 0,33 | 0,84 | 1,79 | 0,95 | 2,5 |
| | 0,12 | 0,13 | –0,33 | 0,18 | 0,48 | 1,24 | 1,03 | –0,41 | 3 |
| | 0,33 | –0,41 | 0 | –1 | 1,21 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 9 | –0,81 | –0,07 | 0,38 | –0,21 | 0,81 | 0,59 | 1,77 | 1,39 | 1,5 |
| | –0,22 | –0,92 | 0,11 | 0,33 | 0,64 | 0,84 | 1,79 | 0,95 | 2,5 |
| | 0,51 | –0,07 | –0,81 | –0,11 | 1,71 | 1,24 | 1,03 | –0,41 | 3 |
| | 0,33 | –0,41 | 0 | –1 | 1,21 | | | | |

| | A | | | | B | C | | | D |
|---|---|---|---|---|---|---|---|---|---|
| Вариант 10 | –1 | 0,22 | –0,11 | 0,31 | –2,7 | 1,42 | 1,43 | 0,27 | 0,1 |
| | 0,38 | –1 | –0,12 | 0,22 | 1,5 | 1,43 | –0,84 | 0,93 | 0,2 |
| | 0,11 | 0,23 | 1 | –0,51 | 1,2 | 0,27 | 0,93 | –0,48 | 0,3 |
| | 0,17 | –0,21 | 0,31 | –1 | 0,17 | | | | |

### Задание 7. Вычислить выражения с заданной точностью

**1.** $y = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$ при $x = 0{,}321$ с точностью $\varepsilon = 2{,}5 \cdot 10^{-5}$.

**2.** $z = 1 + 2\sum_{k=1}^{\infty}(-1)^k e^{-2kx}$ при $x = 0{,}986$ с точностью $\varepsilon = 10^{-7}$.

**3.** $\sin x = x\prod_{k=1}^{\infty}\left(1 - \frac{x^2}{k^2\pi^2}\right)$ при $x = 0{,}837$ с точностью $\varepsilon = 10^{-6}$.

**4.** $y = \sum_{k=1}^{\infty}\ln\left(1 - \frac{x^2}{k^2\pi^2}\right)$ при $x = 1{,}298$ с точностью $\varepsilon = 10^{-5}$.

**5.** $y = \sum_{k=1}^{\infty}\frac{1}{(2k-1)^4}$ с заданной точностью $\varepsilon = 0{,}3 \cdot 10^{-5}$.

**6.** Вычислить число $\pi$ с точностью $\varepsilon = 5 \cdot 10^{-6}$, пользуясь произведением Валлиса $\frac{\pi}{2} = \frac{2}{4}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{7}\cdot\frac{6}{7}\cdot\ldots$.

**7.** $y = \sum_{n=0}^{\infty}2^n \sin\frac{x}{3^n}$ при заданном $x$ с точностью $\varepsilon = 0{,}3 \cdot 10^{-7}$.

**8.** $y = \sum_{n=1}^{\infty}\frac{(-1)^n}{x^n n!}$ при заданном $x$ с точностью $\varepsilon = 0{,}3 \cdot 10^{-4}$.

**9.** $y = \sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}$ при заданном $x$ с точностью $\varepsilon = 0{,}3 \cdot 10^{-7}$.

**10.** $y = \sum_{n=1}^{\infty}\frac{1}{(2n-1)2n}$ с заданной точностью $\varepsilon = 0{,}3 \cdot 10^{-7}$.

**11.** Вычислить приближенное значение функции:

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots .$$

*Замечание.* В сумме учитывать все слагаемые, большие $\varepsilon = 10^{-3}$.

**12.** Вычислить приближенное значение функций

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{(-1)^n x^{2n+1}}{(2n+1)!} + \ldots ;$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + \frac{(-1)^n x^{2n}}{(2n)!} + \ldots .$$

*Замечание.* В сумме учитывать все слагаемые, большие по модулю $\varepsilon = 10^{-4}$.

**13.** Вычислить приближенное значение функций

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + \frac{(-1)^n x^{n+1}}{n+1} + \ldots ;$$

$$\operatorname{arctg}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots + \frac{(-1)^n x^{2n+1}}{(2n+1)} + \ldots .$$

*Замечание.* В сумме учитывать все слагаемые, большие по модулю $\varepsilon = 10^{-5}$.

### *Задание 8. Найти корни уравнений*

*Решить уравнения методом итераций с точностью* $\varepsilon$

**1.** $2x - \ln(x+2) = 0, \quad x \in [0, 1], \quad \varepsilon = 2{,}17 \cdot 10^{-5}$.

**2.** $x - \sin(x+1) = 0, \quad x \in \left[0, \frac{\pi}{2}\right], \quad \varepsilon = 0{,}5 \cdot 10^{-4}$.

**3.** $x - \cos\sqrt{1+x^2} = 0, \quad x \in [0, 1], \quad \varepsilon = 5 \cdot 10^{-6}$.

**4.** $2x - \cos x = 0, \quad x \in [0, 1], \quad \varepsilon = 25 \cdot 10^{-7}$.

**5.** $\sin(x)e^{-2x} = 0$ для $x \in [-2; 2]$.

**6.** $x^3 - 2{,}56x^2 - 1{,}3251x + 4{,}395006 = 0$.

**7.** $x^3 - 2{,}92x^2 + 1{,}4355x + 0{,}791136 = 0$ для $x \in [-3; 3]$.

**8.** $x^3 - 2{,}84x^2 - 5{,}6064x - 1{,}476336 = 0$.

**9.** $x^3 + 1{,}41x^2 - 5{,}4724x - 7{,}380384 = 0$.

*Решить уравнения методом касательных с точностью* $\varepsilon = 10^{-3}$ *на отрезке* $[0; \pi/2]$

**10.** $\dfrac{2}{1+x} - 3\sin x = 0.$

**11.** $3\cos^2 \dfrac{x}{2} - 5x + 6 = 0.$

**12.** $\dfrac{4}{3-x} - 4\cos x = 0.$

**13.** $\sin x - (2-x)^{\frac{3}{2}} = 0.$

**14.** $x^2 + \sin x - 0{,}5 = 0.$

*Решить уравнения методом половинного деления (дихотомии) с точностью* $\varepsilon = 10^{-4}$

**15.** $\sin 2x + 2\sqrt{0{,}1 + x} - 1 = 0, \quad x \in \left[0, \dfrac{\pi}{4}\right].$

**16.** $x \operatorname{tg} x - \ln\left|\dfrac{\pi}{3} + 1 - x\right| = 0, \quad x \in \left[0, \dfrac{\pi}{3}\right].$

**17.** $2 \operatorname{tg} x - \sqrt{\dfrac{\pi}{3} - x} = 0, \quad x \in \left[0, \dfrac{\pi}{3}\right].$

**18.** $5\ln|1{,}9 - x| + 2\cos x, \quad x \in \left[0{,}9, \dfrac{\pi}{2}\right].$

**19.** $0{,}9x = \sin x + 1, \quad x \in [0; 1].$

**20.** $5\sin x = x^2, \quad x \in [0; 1].$

**21.** $\sin x = \cos x, \quad x \in [0; 1].$

**22.** $e^x = x^3 + 1, \quad x \in [0; 1].$

**23.** $x = \cos x - 2, \quad x \in [0; 1].$

**24.** $\sin x = (x + 4)/0{,}9, \quad x \in [0; 1]$.

**25.** $x^3 - 2 = \sin x, \quad x \in [0; 1]$.

**26.** $\cos x = x^3 - 5x, \quad x \in [0; 1]$.

**27.** *Найти корни уравнений методом дихотомии с точностью* 0,0001

| Номер варианта | Задание 1 | Задание 2 |
|---|---|---|
| **1** | $3x^4 + 4x^3 - 12x^2 - 5 = 0$ | $\ln(x) + (x + 1)^3 = 0$ |
| **2** | $2x^3 - 9x^2 - 60x + 1 = 0$ | $x2^x = 1$ |
| **3** | $x^4 - x - 1 = 0$ | $x + \cos(x) = 1$ |
| **4** | $2x^4 - x^2 - 10 = 0$ | $x + \lg(1 + x) = 1{,}5$ |
| **5** | $3x^4 + 8x^3 + 6x^2 - 10 = 0$ | $\lg(2 + x) + 2x = 3$ |
| **6** | $x^4 - 18x^2 + 5x - 8 = 0$ | $2^x + 5x - 3 = 0$ |
| **7** | $x^4 + 4x^3 - 12x^2 + 1 = 0$ | $5^x + 3x = 0$ |
| **8** | $x^4 - x^3 - 2x^2 + 3x - 3 = 0$ | $3e^x = 5x + 2$ |
| **9** | $3x^4 + 4x^3 - 12x^2 + 1 = 0$ | $5^x = 6x + 3$ |
| **10** | $3x^4 - 8x^3 - 18x^2 + 2 = 0$ | $2e^x + 5x - 6 = 0$ |

*Решить уравнения методом хорд с точностью* $\varepsilon = 10^{-4}$ *на отрезке* [0; 1]

**28.** $2^{x+1} + x - 3 = 0$.

**29.** $\operatorname{arctg} x - 5 \ln|2 - x| = 0$.

**30.** $4 \lg x - \dfrac{1}{x} = 0, \quad x \in [1, 3]$.

**31.** $8 \lg x - \dfrac{3}{x^2} = 0, \quad x \in [1, 3]$.

**32.** *Найти корни уравнения* $f(x) = 0$ *с точностью* $\varepsilon = 10^{-3}$

| Номер варианта | $f(x)$ | Номер варианта | $f(x)$ |
|---|---|---|---|
| **1** | $\ln x - \frac{1}{x^2}$ | **13** | $e^{-x} - \sqrt{x-1}$ |
| **2** | $2\ln x - \frac{x}{2} + 1$ | **14** | $2\sin(3x) - 1{,}5x$ |
| **3** | $\frac{1-x}{x} - 3\cos(4x)$ | **15** | $0{,}1e^{-x} - \frac{x}{2}$ |
| **4** | $\operatorname{ctg} x - x^2$ | **16** | $\ln(1{,}2x) - 1{,}5x + 2$ |
| **5** | $\operatorname{tg}\frac{x}{4} - x - 2$ | **17** | $\operatorname{tg}(2{,}5x) - 5x$ |
| **6** | $\sqrt{x} - 3\sin x$ | **18** | $\ln x - 2\cos x$ |
| **7** | $\sqrt{x} - \cos\frac{x}{2}$ | **19** | $\sqrt{2-x^2} - e^x$ |
| **8** | $2\ln x - \frac{1}{x}$ | **20** | $e^{-(x+1)} + x^2 + 2x - 1$ |
| **9** | $x - 3\cos^2 x$ | **21** | $e^{-x} - 2 + x^2$ |
| **10** | $\operatorname{tg}(7{,}5x) - 2(x+1)$ | **22** | $xe^x - x - 1$ |
| **11** | $\ln x - \frac{7}{2x+6}$ | **23** | $(\cos x)^2 - \frac{1}{2}\cos x + \frac{1}{4}$ |
| **12** | $e^{-x} - (x-1)^2$ | **24** | $\sin(x+2) - x^2 + 2x - 1$ |

**33.** *Отделить корни уравнений графически (или аналитически) и найти приближенные значения корней с точностью* $\varepsilon = 0{,}00001$

| Номер варианта | Функция | Номер варианта | Функция |
|---|---|---|---|
| **1** | $x - \sin x = 0{,}25$<br>$x^3 - 3x^2 + 9x - 8 = 0$ | **13** | $x^2\cos(2x+1) = -1$<br>$2x^4 - 3x^2 + 15 = 0$ |
| **2** | $\operatorname{tg}(0{,}2 + x) = x^3 + 3$<br>$x^3 + 2x^2 - 7x + 1 = 0$ | **14** | $0{,}5^x - 1 = (x+2)^2$<br>$3x^3 - 2x^2 + x + 1 = 0$ |
| **3** | $(x+1)^{1/3} - \cos(0{,}3 + 0{,}4x) = 2$<br>$x^3 + x^2 - 3x + 4 = 0$ | **15** | $\cos(x+2) - x + 2x + 1 = 0$<br>$x^4 + 2x^2 + 3x - 10 = 0$ |

*Окончание таблицы*

| Номер варианта | Функция | Номер варианта | Функция |
|---|---|---|---|
| **4** | $\operatorname{tg}(2{,}3 + 0{,}5x) = 3x + 2$<br>$2x^3 - 3x^2 + x + 1 = 0$ | **16** | $x + \ln(2x + 3) = 0{,}5$<br>$-2^{x-1} - x = 0$ |
| **5** | $2e^{x+1} + 3x + 1 = 0$<br>$3x^4 + 4x^3 - 12x - 1 = 0$ | **17** | $x^2 + 4\sin(x + 1) = 0$<br>$3x^4 + 4x^3 - 12x^2 - 5 = 0$ |
| **6** | $3x^2 + \cos(2x + 1) = 1$<br>$x^4 - 2x^3 - 10x^2 - 2 = 0$ | **18** | $e^{2x+1} + 5x - 1 = 0$<br>$7x^3 - 2x^2 + 3x - 10 = 0$ |
| **7** | $5x\sin(2x + 1) = 0{,}43$<br>$x^3 - 7x^2 + 2x - 1 = 0$ | **19** | $2e^{x+1} - 3x + 1 = 0$<br>$x\lg(x^2 + 2x - 1) = 1$ |
| **8** | $x\cos(x + 2) = x^2 - 3x + 1$<br>$x^3 + x^2 + x - 10 = 0$ | **20** | $x^2\cos(2x - 1) = 1$<br>$2x^4 - 3x^3 + 3x^2 - x + 1 = 0$ |
| **9** | $(x - 3)\cos(x + 2) = 1$<br>$2x^4 - 3x^3 + x - 1 = 0$ | **21** | $2x - \lg(x + 3) = 7$<br>$\operatorname{tg}^3(x) + x - 1 = 0$ |
| **10** | $\ln x + (x + 1)^3 = 0$<br>$x^2 - 2 + 0{,}5^x = 0$ | **22** | $(1 - x)e^{3x-1} = 0{,}5$<br>$3\sin^2(x + 1) - x^2 + x = 2$ |
| **11** | $x\lg(x + 1) = 1$<br>$2x^3 - 9x^2 - 60x = 0$ | **23** | $x = (\log(x + 2))^{1/2} - 1$<br>$x^4 - x^3 - 2x^2 + 3x - 3 = 0$ |
| **12** | $\operatorname{arctg} x - 1/(3x^3) = 0$<br>$2x^4 + x - 3 = 0$ | **24** | $5\cos(x + 3) = x - 0{,}5$<br>$3^x + 2 - x = 0$ |

## *Задание 9. Решить нелинейные уравнения $f_1(x) = f_2(x)$*

| Номер варианта | $f_1(x)$ | $f_2(x)$ | Отрезок |
|---|---|---|---|
| **1** | $\sqrt{2x - 1}$ | $2\cos(3x + 1) + 1$ | [1; 4] |
| **2** | $\dfrac{2}{x + 4}$ | $1 + \cos 2x$ | [–2; 3] |
| **3** | $x^3 - 5x^2 + 7$ | $\dfrac{5 - x}{\sqrt{x + 2}}$ | [–1; 5] |
| **4** | $2x + 4 - x^2$ | $7\sin(2x + 2)$ | [–3; 1] |

*Окончание таблицы*

| Номер варианта | $f_1(x)$ | $f_2(x)$ | Отрезок |
|---|---|---|---|
| 5 | $\cos(2{,}5x + 1)$ | $\frac{1 - x^2}{2}$ | [–2; 0,5] |
| 6 | $2\sin(1 - 3x)$ | $\frac{x^2 + 2}{3} - 1$ | $[0; \pi]$ |
| 7 | $\frac{\sqrt{1 - x}}{2}$ | $\sin(3x - 1)$ | [–2; 1] |
| 8 | $\frac{2}{x} - 1$ | $3\sin 2x$ | [1; 5] |
| 9 | $3\cos 2x$ | $\frac{7 - x}{x^2 + 3x + 5}$ | [–2; 3] |
| 10 | $\frac{\sqrt{x^2 + 1}}{3x}$ | $2\sin(2x + 1)$ | [0,1; 3,5] |

### *Задание 10. Решить систему уравнений*

**1.** $\begin{cases} 2x - y = 2; \\ 3x - y = 5. \end{cases}$

**2.** $\begin{cases} 2x + y = -1; \\ 0{,}4x - y = -5. \end{cases}$

**3.** $\begin{cases} x^2 + y^2 = 25; \\ xy = 12. \end{cases}$

**4.** $\begin{cases} 21x - y = -4; \\ 17x + y = -7. \end{cases}$

**5.** $\begin{cases} \frac{x}{3} - \frac{y}{2} = 1; \\ \frac{x}{6} + \frac{y}{8} = 2. \end{cases}$

**6.** $\begin{cases} x + y = 1; \\ xy = 84. \end{cases}$

**7.** $\begin{cases} \frac{1}{4}x + y = -5; \\ 7x - y = 3{,}5. \end{cases}$

**8.** $\begin{cases} 6x + y = -0{,}2; \\ 2x - y = -5. \end{cases}$

**9.** $\begin{cases} \frac{1}{5}x + y = 7{,}6; \\ \frac{1}{7}x - y = 4. \end{cases}$

**10.** $\begin{cases} 3x - y = 7; \\ x + y = -4. \end{cases}$

**11.** Получить результаты решения системы уравнений:

а) $\begin{cases} x - y = 2; \\ 2x + y = 1; \end{cases}$ б) $\begin{cases} 5x + y = 25; \\ 8x - 3y = 17; \end{cases}$ в) $\begin{cases} 5x - y = 22; \\ 2x + 2y = 16. \end{cases}$

**12.** Составить программу для решения системы уравнений с тремя неизвестными по формулам Крамера и получить результаты для решения системы уравнений:

а) $\begin{cases} 10x - 9z = 19; \\ 8x - y = 10; \\ y - 12z = 10; \end{cases}$ б) $\begin{cases} x + 2y + z = -7; \\ 2x + y - z = 1; \\ 3x - y - 2z = 2. \end{cases}$

**13.** Получить результаты решения системы уравнений:

а) $\begin{cases} 3x - 4y + 5z = 18; \\ 2x + 4y - 3z = 26; \\ x - 6y + 8z = 0, \end{cases}$ б) $\begin{cases} 2x + y + z = 7; \\ x + 2y + z = 8; \\ x + y + 2z = 9. \end{cases}$

**14.** Решить систему уравнений по формулам Крамера

$$\begin{cases} x + 2y + 3z = 8; \\ 3x + y + z = 6; \\ x + y + 2z = 6. \end{cases}$$

**15.** Решить систему уравнений по методу Гаусса

$$\begin{cases} 2x_1 + 4x_2 - 6x_3 = -4; \\ 2x_1 - x_2 + x_3 = 3; \\ x_1 + x_2 + x_3 = 2. \end{cases}$$

**16.** Решить систему уравнений методом Крамера

| Номер варианта | Система уравнений |
|---|---|
| 1 | $\begin{cases} 7{,}54x_1 + 5{,}29x_2 - 2{,}34x_3 = 2{,}21; \\ -9{,}49x_1 + 2{,}34x_2 + 8{,}58x_3 = 4{,}16; \\ 1{,}95x_1 + 3{,}77x_2 - 4{,}68x_3 = 8{,}32 \end{cases}$ |
| 2 | $\begin{cases} 6{,}89x_1 - 8{,}45x_2 + 4{,}16x_3 = 9{,}49; \\ 9{,}10x_1 + 7{,}67x_2 - 1{,}56x_3 = 5{,}85; \\ 1{,}56x_1 - 3{,}38x_2 + 7{,}02x_3 = 8{,}19 \end{cases}$ |
| 3 | $\begin{cases} -8{,}97x_1 + 1{,}82x_2 + 5{,}85x_3 = -5{,}07; \\ 2{,}73x_1 + 7{,}28x_2 + 5{,}07x_3 = -6{,}84; \\ 6{,}37x_1 + 6{,}11x_2 - 3{,}77x_3 = 4{,}13 \end{cases}$ |

*Окончание табл.*

| Номер варианта | Система уравнений |
|---|---|
| **4** | $\begin{cases} 6{,}63x_1 + 4{,}16x_2 - 7{,}67x_3 = 3{,}12; \\ 9{,}23x_1 - 1{,}69x_2 + 9{,}12x_3 = -4{,}55; \\ -2{,}61x_1 + 1{,}69x_2 + 7{,}67x_3 = -1{,}82 \end{cases}$ |
| **5** | $\begin{cases} 6{,}89x_1 + 4{,}16x_2 - 7{,}67x_3 = 7{,}54; \\ -1{,}95x_1 + 7{,}54x_2 - 7{,}15x_3 = 8{,}19; \\ 4{,}81x_1 - 4{,}68x_2 + 8{,}58x_3 = 6{,}24 \end{cases}$ |
| **6** | $\begin{cases} -4{,}68x_1 + 2{,}63x_2 + 2{,}47x_3 = 7{,}54; \\ 1{,}34x_1 + 7{,}16x_2 - 7{,}51x_3 = 8{,}97; \\ 4{,}18x_1 + 2{,}63x_2 + 3{,}12x_3 = -5{,}42 \end{cases}$ |
| **7** | $\begin{cases} 4{,}94x_1 + 6{,}89x_2 + 5{,}46x_3 = 2{,}99; \\ 2{,}61x_1 + 5{,}85x_2 - 5{,}07x_3 = 4{,}55; \\ -9{,}36x_1 - 3{,}77x_2 + 2{,}47x_3 = 7{,}28 \end{cases}$ |
| **8** | $\begin{cases} -2{,}08x_1 + 6{,}50x_2 + 9{,}36x_3 = -2{,}86; \\ -5{,}33x_1 - 4{,}68x_2 + 8{,}06x_3 = 6{,}86; \\ -8{,}97x_1 + 3{,}90x_2 + 6{,}52x_3 = 8{,}32 \end{cases}$ |
| **9** | $\begin{cases} 6{,}24x_1 - 9{,}23x_2 - 2{,}60x_3 = 3{,}64; \\ 4{,}55x_1 - 3{,}77x_2 + 1{,}69x_3 = -6{,}76; \\ -5{,}33x_1 + 4{,}94x_2 + 6{,}37x_3 = 8{,}71 \end{cases}$ |
| **10** | $\begin{cases} 7{,}93x_1 + 6{,}89x_2 + 4{,}68x_3 = 6{,}76; \\ -2{,}91x_1 - 4{,}29x_2 + 4{,}81x_3 = 8{,}32; \\ -1{,}56x_1 - 3{,}12x_2 - 3{,}51x_3 = 8{,}94 \end{cases}$ |

### *Задание 11. Найти площадь фигуры, ограниченную графиками функций*

**1.** $y = x^2 - 2x + 2, \quad y = 2 + 4x - x^2.$

**2.** $y = \sin x, \quad y = \cos x, \quad x = 0, \quad x = 2\pi.$

**3.** $y = x^3, \quad y = x^{\frac{1}{3}}, \quad x = 0, \quad x = 1.$

**4.** $y = \operatorname{tg} x, \quad y = \frac{2}{3}\cos x, \quad x \in \left[0, \frac{\pi}{2}\right), \; x = 0.$

**5.** $y = 6x^2 - 5x + 1, \quad y = \cos \pi x, \quad x = 0, \quad x = 1.$

**6.** $y = x^2, \quad y = \frac{x^2}{x-2}, \quad y = 0, \quad x = 4.$

**7.** $y = \sqrt{x}, \quad y = \sqrt{4 - 3x}, \quad y = 0.$

**8.** $y = -x^2, \quad y = 2e^x, \quad x = 0, \quad x = 1.$

**9.** $y = x - 2, \quad y = x^2 - 4x + 2.$

**10.** $y = \sin x, \quad y = 2\sin x, \quad x = \frac{5\pi}{4}, \quad x = 0.$

### *Задание 12. Вычислить определенный интеграл*

**1.** *Вычислить определенный интеграл с заданной точностью*

| Номер варианта | Интеграл | Точность | Метод |
|---|---|---|---|
| 1 | $\int_0^1 \sqrt{1+x}\,dx$ | $10^{-6}$ | Средних прямоугольников |
| 2 | $\int_0^1 x^2\sqrt{1-x^3}\,dx$ | $10^{-5}$ | Трапеций |
| 3 | $\int_0^1 (e^x - 1)^2 e^x\,dx$ | $10^{-8}$ | Симпсона |
| 4 | $\int_1^e \frac{dx}{x\sqrt{1-\ln(x)}}$ | $10^{-5}$ | Средних прямоугольников |
| 5 | $\int_2^5 \frac{\cos^2 x}{\ln x}\,dx$ | $10^{-8}$ | Трапеций |
| 6 | $\int_0^{\pi/2} x\cos x\,dx$ | $10^{-6}$ | Симпсона |

*Окончание табл.*

| Номер варианта | Интеграл | Точность | Метод |
|---|---|---|---|
| **7** | $\int_0^1 \cos(xe^{-3x})\,dx$ | $10^{-5}$ | Средних прямоугольников |
| **8** | $\int_1^2 x^{-1}\ln(1+x)\,dx$ | $10^{-8}$ | Трапеций |
| **9** | $\int_1^2 x^{-1}e^x\,dx$ | $10^{-8}$ | Симпсона |
| **10** | $\int_1^2 \mathrm{sh}(x^2)\,dx$ | $10^{-5}$ | Средних прямоугольников |
| **11** | $\int_0^1 \cos(x^2+x)\,dx$ | $10^{-6}$ | Трапеций |
| **12** | $\int_0^{\pi/3} x\sin(x^3)\,dx$ | $10^{-8}$ | Симпсона |
| **13** | $\int_0^{\pi} \sin(\cos(2x+1))\,dx$ | $10^{-6}$ | Средних прямоугольников |
| **14** | $\int_0^{\pi/4} \ln(1+\cos x)\,dx$ | $10^{-5}$ | Трапеций |
| **15** | $\int_0^{\pi/3} \ln(1+\sqrt{\sin x})\,dx$ | $10^{-8}$ | Симпсона |
| **16** | $\int_{0.1}^2 \frac{\sin x}{\sqrt{x}}\,dx$ | $10^{-5}$ | Средних прямоугольников |
| **17** | $\int_0^{\pi/4} x^3\cos(x^2)\,dx$ | $10^{-8}$ | Трапеций |
| **18** | $\int_0^1 \mathrm{ch}\,x^2\,dx$ | $10^{-6}$ | Симпсона |
| **19** | $\int_{\pi/2}^{\pi} \sqrt{x}e^{-x}\,dx$ | $10^{-5}$ | Средних прямоугольников |
| **20** | $\int_{0.1}^2 \frac{\cos(x^2)}{x+x^3}\,dx$ | $10^{-8}$ | Трапеций |

**2.** *Вычислить определенный интеграл методом трапеций* с точностью ε

| Номер варианта | Интеграл | Точность |
|---|---|---|
| **1** | $\int_0^{\sqrt{3}} \frac{x\,dx}{\sqrt{4-x^2}}$ | $10^{-6}$ |
| **2** | $\int_1^2 (x^2 + \frac{1}{x^4})\,dx$ | $10^{-5}$ |
| **3** | $\int_0^1 \frac{dx}{\sqrt{x^2+1}}$ | $10^{-8}$ |
| **4** | $\int_0^{\pi/4} \sin 4x\,dx$ | $10^{-5}$ |
| **5** | $\int_4^9 \frac{dx}{\sqrt{x}-1}$ | $10^{-8}$ |
| **6** | $\int_0^4 \frac{dx}{1+\sqrt{2x+1}}$ | $10^{-6}$ |
| **7** | $\int_0^1 \frac{x^2 dx}{\sqrt{4-x^2}}$ | $10^{-5}$ |
| **8** | $\int_0^1 \frac{dx}{e^x+1}$ | $10^{-8}$ |
| **9** | $\int_0^5 \sqrt{\frac{x}{10-x}}\,dx$ | $10^{-8}$ |
| **10** | $\int_0^{\pi/2} \sin x \cos^2 x\,dx$ | $10^{-5}$ |

**3.** *Вычислить определенный интеграл по формулам левых и правых прямоугольников* при $N = 20, 50, 100, 200$

| Номер варианта | Интеграл | Номер варианта | Интеграл |
|---|---|---|---|
| 1 | $\int_0^1 \sqrt{1+x}\,dx$ | 2 | $\int_0^1 x^2\sqrt{1-x^3}\,dx$ |
| 3 | $\int_0^1 (e^x-1)a^x\,dx$ | 4 | $\int_1^e \frac{dx}{x\sqrt{1-\ln(x)}}$ |
| 5 | $\int_0^{\pi/2} x\cos x\,dx$ | 6 | $\int_0^1 \cos(x+x^3)\,dx$ |
| 7 | $\int_0^1 \sin(x+x^3)\,dx$ | 8 | $\int_0^1 \cos(xe^{-3x})\,dx$ |
| 9 | $\int_0^1 \ln x\,(x+1)^{-1}\,dx$ | 10 | $\int_{\pi/2}^{\pi} \sqrt{x}e^{-x}\,dx$ |
| 11 | $\int_0^1 \cos(x^3)\,dx$ | 12 | $\int_1^2 x^{-1}\ln(1+x)\,dx$ |
| 13 | $\int_1^2 x^{-1}e^x\,dx$ | 14 | $\int_0^1 \cos(x^2+x)\,dx$ |
| 15 | $\int_0^{\pi/3} x\sin(x^3)\,dx$ | 16 | $\int_0^{\pi} \cos(\sin(x+1))\,dx$ |
| 17 | $\int_0^{\pi} \cos(2\sin(3x))\,dx$ | 18 | $\int_0^{\pi} \sin(\cos(2x+1))\,dx$ |
| 19 | $\int_0^{\pi/4} \ln(1+\cos x)\,dx$ | 20 | $\int_1^2 \sin(x^3)\,dx$ |
| 21 | $\int_0^{\pi/3} \ln(1+\sqrt{\cos x})\,dx$ | 22 | $\int_0^{\pi} \sin(3\cos x)\,dx$ |
| 23 | $\int_0^{\pi/4} (x+x^3)\cos(x^2)\,dx$ | 24 | $\int_0^{\pi/3} \ln(1+\sqrt{\sin x})\,dx$ |
| 25 | $\int_0^{\pi} x^2e^{-2x}\,dx$ | 26 | $\int_{0,1}^{2} \frac{\cos x}{\sqrt{x+1}}\,dx$ |

## *Задание 13. Построить графики первой и второй производных функции*

*Замечание.*

$$\frac{df}{dx} \approx \frac{f(x+h)-f(x)}{h}, \quad \frac{df}{dx} \approx \frac{f(x+h)-f(x-h)}{2h},$$

$$\frac{d^2f}{dx^2} \approx \frac{f(x-h)-2f(x)+f(x+h)}{h^2},$$

где $h$ — малое приращение аргумента (шаг).

| Номер варианта | Функция | Номер варианта | Функция |
|---|---|---|---|
| **1** | $\sin(x)e^{-0,5x}$, $x \in [0; 2\pi]$ | **9** | $\frac{x^2}{x^2+1}$, $x \in [0; 3]$ |
| **2** | $e^{-x^2/5}$, $x \in [-5; 5]$ | **10** | $x \sin x$, $x \in [0; 2\pi]$ |
| **3** | $\sin^2(0,5x)\sqrt{x}$, $x \in [0; 10]$ | **11** | $\frac{x}{5^x}$, $x \in [0; 5]$ |
| **4** | $\frac{x}{x^2+1}$, $x \in [0; 10]$ | **12** | $x + \sin x$, $x \in [0; 2\pi]$ |
| **5** | $\frac{x^3}{e^x}$, $x \in [0; 8]$ | **13** | $\frac{\cos x}{1+x^2}$, $x \in [-5; 5]$ |
| **6** | $\frac{\sin x}{x}$, $x \in [2; 10]$ | **14** | $\frac{1}{x^4+1}$, $x \in [0; 4]$ |
| **7** | $(x-5)^2 \operatorname{arctg} x$, $x \in [0; 6]$ | **15** | $\sqrt[3]{x} \sin x$, $x \in [0; 2\pi]$ |
| **8** | $\frac{\sin x}{1+x^2}$, $x \in [-5; 5]$ | **16** | $x - \arcsin x$, $x \in [-0,5; 0,5]$ |

## *Задание 14. Решить дифференциальное уравнение*

**1.** *Решить задачу Коши методами Эйлера и Рунге — Кутта* на отрезке $[0,1; 1,1]$ с шагом $h = 0,1$ при начальном условии $y(0, 1) = 0,25$ с точностью $10^{-5}$.

Четные варианты: $y' = a(x^2 + \sin(bx)) + cy$.

Нечетные варианты: $y' = a(x^2 + \cos(bx)) + cy$.

| Номер варианта | $a$ | $b$ | $c$ | Номер варианта | $a$ | $b$ | $c$ |
|---|---|---|---|---|---|---|---|
| **1** | 0,133 | 2 | 0,872 | **14** | 0,258 | 0,5 | 1,278 |
| **2** | 0,215 | 1,5 | 1,283 | **15** | 0,317 | 1,1 | 0,283 |
| **3** | 0,158 | 0,8 | 1,164 | **16** | 0,166 | 1,3 | 0,461 |
| **4** | 0,173 | 0,7 | 0,754 | **17** | 0,186 | 0,7 | 0,457 |
| **5** | 0,221 | 1,2 | 0,452 | **18** | 0,215 | 0,4 | 0,254 |
| **6** | 0,163 | 0,4 | 0,635 | **19** | 0,188 | 0,1 | 0,536 |
| **7** | 0,133 | 2 | 0,872 | **20** | 0,193 | 1,9 | 0,248 |
| **8** | 0,145 | 0,5 | 0,842 | **21** | 0,291 | 0,3 | 1,354 |
| **9** | 0,213 | 1,8 | 0,368 | **22** | 0,311 | 1,7 | 0,279 |
| **10** | 0,127 | 0,6 | 0,573 | **23** | 0,353 | 1,3 | 1,216 |
| **11** | 0,232 | 1,6 | 1,453 | **24** | 0,415 | 0,4 | 0,354 |
| **12** | 0,417 | 0,8 | 0,972 | **25** | 0,233 | 0,2 | 0,427 |
| **13** | 0,324 | 1,5 | 0,612 | **26** | 0,355 | 1,2 | 0,388 |

**2.** *Найти приближенные решения задачи Коши* $u' = f(x, u)$, $u(0) = 0$ на отрезке $[0; 1]$ с точностью $10^{-4}$, используя методы Эйлера и Рунге — Кутта

| Номер варианта | $f(x, u)$ | Номер варианта | $f(x, u)$ |
|---|---|---|---|
| **1** | $\cos(x+u) + \frac{3}{2}(x-u)$ | **14** | $(0{,}7 - u^2)\cos x + 0{,}3u$ |
| **2** | $0{,}6\sin x - 1{,}2u^2 + 1$ | **15** | $1 + \frac{1}{5}u\sin x - u^2$ |
| **3** | $1 - \sin(u+2x) + \frac{0{,}5u}{2+x}$ | **16** | $1 + (0{,}7 - x)\sin x - 1{,}2xu$ |
| **4** | $\cos(2x+u) + \frac{3}{2}(x-u)$ | **17** | $0{,}8e^{-(0{,}8+xu)} + 0{,}6x^2u$ |
| **5** | $\frac{\cos u}{1+x} - \frac{u^2}{2}$ | **18** | $\cos(1{,}75x+u) + 1{,}2(x-u)$ |
| **6** | $1 - \frac{0{,}1x}{u+2} - \sin(2x+u)$ | **19** | $1 - \sin(2x+u) + \frac{0{,}5u^2}{2+x}$ |
| **7** | $(0{,}8 - u^2)\cos x + 0{,}4u$ | **20** | $\frac{\cos u}{1{,}25+x} - 0{,}3u^2$ |

*Окончание табл.*

| Номер варианта | $f(x,\ u)$ | Номер варианта | $f(x,\ u)$ |
|---|---|---|---|
| **8** | $\dfrac{\cos u}{1{,}25 + x} - 0{,}1u^2$ | **21** | $1 + (1{,}2 - x)\sin u - u(1 + x)$ |
| **9** | $1 + 0{,}6\,u \sin x - 1{,}5\,u^2$ | **22** | $(0{,}9 - u^2)\cos x + 0{,}5\,u$ |
| **10** | $\cos(1{,}5x + u) + 1{,}5(x - u)$ | **23** | $1 + 0{,}8\,u \sin x - 1{,}75\,u^2$ |
| **11** | $0{,}2e^{-xu} + 0{,}3\,(x^2 + u)$ | **24** | $0{,}4\,e^{-(0{,}4+xu)} + 0{,}4\,(x + u)$ |
| **12** | $u\cos^2\left(u - \dfrac{x}{2}\right) + 0{,}1\,x^2$ | **25** | $(1 - x)\sin 2u - (0{,}6 + x)\,u$ |
| **13** | $\dfrac{\cos u}{1{,}5 + x} - 0{,}1\,u^2$ | **26** | $\cos(x + u + xu) - 1{,}3\,x$ |

## *Задание 15. Осуществить интерполирование функций*

**1.** *Осуществить интерполирование функций, заданных на отрезке*

| Номер варианта | Функция | Номер варианта | Функция |
|---|---|---|---|
| **1** | $\sqrt[3]{x}\sin x,\ \ x \in [0;\ 2\pi]$ | **9** | $\dfrac{x}{2}\sin^2 x,\ \ x \in [0;\ 2\pi]$ |
| **2** | $\sin\dfrac{1}{x},\ \ x \in [0{,}1;\ 0{,}5]$ | **10** | $\sin(\cos x),\ \ x \in [0;\ 2\pi]$ |
| **3** | $(x - 5)^2 \operatorname{arctg} x,\ \ x \in [2;\ 10]$ | **11** | $\left(\dfrac{1}{3}\right)^{\sin x},\ \ x \in [0;\ 2\pi]$ |
| **4** | $\dfrac{\sin x}{x},\ \ x \in [-5;\ 5]$ | **12** | $\dfrac{1}{x^4 + 1},\ \ x \in [0;\ 4]$ |
| **5** | $e^{-x^2/5},\ \ x \in [-5;\ 5]$ | **13** | $x \sin x,\ \ x \in [0;\ 2\pi]$ |
| **6** | $\dfrac{x^3}{e^x},\ \ x \in [0;\ 8]$ | **14** | $\dfrac{\cos x}{1 + x^2},\ \ x \in [-5;\ 5]$ |
| **7** | $\sin(x)e^{-0{,}5x},\ \ x \in [0;\ 2\pi]$ | **15** | $\dfrac{x^2}{x^2 + 1},\ \ x \in [0;\ 3]$ |
| **8** | $\dfrac{x}{x^2 + 1},\ \ x \in [0;\ 10]$ | **16** | $x + \sin x,\ \ x \in [0;\ 2\pi]$ |

**2.** *Осуществить интерполирование функций, заданных таблично*

| | | | | | | |
|---|---|---|---|---|---|---|
| **Вариант 1** | | | | | | |
| $x$ | –1 | 0 | 3 | | | |
| $y$ | –3 | 5 | 2 | | | |
| arg = 0,702 | | | | | | |
| **Вариант 2** | | | | | | |
| $x$ | 0,43 | 0,48 | 0,55 | 0,62 | 0,7 | 0,75 |
| $y$ | 1,63597 | 1,73234 | 1,87686 | 2,03345 | 2,22846 | 2,35973 |
| arg = 0,702 | | | | | | |
| **Вариант 3** | | | | | | |
| $x$ | 2 | 3 | 5 | | | |
| $y$ | 4 | 1 | 7 | | | |
| arg = 0,102 | | | | | | |
| **Вариант 4** | | | | | | |
| $x$ | 0,02 | 0,08 | 0,12 | 0,17 | 0,23 | 0,3 |
| $y$ | 1,02316 | 1,0959 | 1,14725 | 1,21483 | 1,3012 | 1,40976 |
| arg = 0,102 | | | | | | |
| **Вариант 5** | | | | | | |
| $x$ | 0 | 2 | 3 | | | |
| $y$ | –1 | –4 | 2 | | | |
| arg = 0,526 | | | | | | |
| **Вариант 6** | | | | | | |
| $x$ | 0,35 | 0,41 | 0,47 | 0,51 | 0,56 | 0,64 |
| $y$ | 2,73951 | 2,3008 | 1,96864 | 1,78776 | 1,59502 | 1,3431 |
| arg = 0,526 | | | | | | |
| **Вариант 7** | | | | | | |
| $x$ | 7 | 9 | 13 | | | |
| $y$ | 2 | –2 | 3 | | | |
| arg = 0,616 | | | | | | |
| **Вариант 8** | | | | | | |
| $x$ | 0,41 | 0,46 | 0,52 | 0,6 | 0,65 | 0,72 |
| $y$ | 2,57418 | 2,32513 | 2,09336 | 1,86203 | 1,74926 | 1,62098 |
| arg = 0,616 | | | | | | |

## *Задание 16. Осуществить аппроксимацию функций*

*Для исходных данных выполнить аппроксимацию алгебраическими многочленами различной степени и оценить их качество по отношению остаточного и исходного среднеквадратичного отклонений*

| **Вариант 1** | | **Вариант 2** | |
|---|---|---|---|
| $x$ | $y(x)$ | $x$ | $y(x)$ |
| 1,0 | 0 | 4,0 | 0,000196 |
| 1,1 | 0,324097 | 4,1 | −5,19505 |
| 1,2 | 0,643881 | 4,2 | −10,3689 |
| 1,3 | 0,922415 | 4,3 | −14,959 |
| 1,4 | 1,1253 | 4,4 | −18,4126 |
| 1,5 | 1,224745 | 4,5 | −20,25 |
| 1,6 | 1,20301 | 4,6 | −20,1243 |
| 1,7 | 1,054847 | 4,7 | −17,8711 |
| 1,8 | 0,788625 | 4,8 | −13,5425 |
| 1,9 | 0,425989 | 4,9 | −7,41942 |
| 2,0 | 0,0000462 | 5,0 | 0 |
| 2,1 | −0,44776 | 5,1 | 8,037451 |
| 2,2 | −0,87178 | 5,2 | 15,89357 |
| 2,3 | −1,2269 | 5,3 | 22,72513 |
| 2,4 | −1,47335 | 5,4 | 27,73269 |
| 2,5 | −1,58114 | 5,5 | 30,25 |
| 2,6 | −1,53356 | 5,6 | 29,82532 |
| 2,7 | −1,3294 | 5,7 | 26,2854 |
| 2,8 | −0,98363 | 5,8 | 19,77381 |
| 2,9 | −0,52634 | 5,9 | 10,75785 |
| 3,0 | −0,00011 | 6,0 | 0,001176 |
| 3,1 | 0,543966 | 6,1 | −11,4973 |
| 3,2 | 1,051358 | 6,2 | −22,5932 |
| 3,3 | 1,469572 | 6,3 | −32,1089 |
| 3,4 | 1,753617 | 6,4 | −38,9547 |
| 3,5 | 1,870829 | 6,5 | −42,25 |

### Задание 17. Решить системы линейных уравнений

*Найти решение системы линейных уравнений методом Зейделя.* Вычисления выполнять с точностью $\varepsilon = 10^{-5}$

| Номер варианта | $X_1$ | $X_2$ | $X_3$ | $B$ | Номер варианта | $X_1$ | $X_2$ | $X_3$ | $B$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,34 | 0,71 | 0,63 | 2,08 | 2 | 3,75 | −0,28 | 0,17 | 0,75 |
| | 0,71 | 0,65 | −0,18 | 0,17 | | 2,11 | −0,11 | −0,12 | 1,11 |
| | 1,17 | −2,35 | 0,75 | 1,28 | | 0,22 | −3,17 | 1,81 | 0,05 |
| 3 | 0,21 | −0,18 | 0,75 | 0,11 | 4 | 0,13 | −0,14 | 2,0 | 0,15 |
| | 0,13 | 0,75 | −0,11 | 2,0 | | 0,75 | 0,18 | −0,77 | 0,11 |
| | 3,01 | −0,33 | 0,11 | 0,13 | | 0,28 | 0,17 | 0,39 | 0,12 |
| 5 | 3,01 | −0,14 | −0,15 | 1,0 | 6 | 0,92 | −0,83 | 0,25 | 2,15 |
| | 1,11 | 0,13 | −0,75 | 0,13 | | 0,24 | −0,54 | 0,43 | 0,62 |
| | 0,17 | −2,11 | 0,71 | 0,17 | | 0,73 | −0,81 | −0,67 | 0,88 |
| 7 | 1,24 | −0,87 | 3,17 | 0,46 | 8 | 0,32 | −0,42 | 0,85 | 1,32 |
| | 2,11 | −0,45 | 1,44 | 1,5 | | 0,63 | −1,43 | −0,58 | −0,44 |
| | 0,48 | 1,25 | −0,63 | 0,35 | | 0,84 | −2,23 | −0,52 | 0,64 |

### Задание 18. Решить задачи линейного программирования

**Задача 1.** Привести задачу линейного программирования к канонической форме.

Целевая функция: $F = x_1 + x_2 - x_3 - x_4$.

Ограничения:

| | | | | |
|---|---|---|---|---|
| $x_1$ | $+3x_2$ | | $-x_4$ | $\le 2$ |
| $-x_1$ | $+x_2$ | | $+x_4$ | $\ge 1$ |
| | $x_2$ | $+x_3$ | | $\le 3$ |
| $x_1 \ge 0$ | $x_2 \ge 0$ | $x_3 \ge 0$ | $x_4 \ge 0$ | |

Характер экстремума: min.

*Замечание.*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $F=$ | $-x_1$ | $-x_2$ | $+x_3$ | $+x_4$ | | | | $=0$ |
| | $x_1$ | $+3x_2$ | | $-x_4$ | $+x_5$ | | | $=2$ |
| | $-x_1$ | $+x_2$ | | $+x_4$ | | $-x_6$ | $+y_1$ | $=1$ |
| | | $x_2$ | $+x_3$ | | | | $+x_7$ | $=3$ |
| | $x_1 \geq 0$ | $x_2 \geq 0$ | $x_3 \geq 0$ | $x_4 \geq 0$ | $x_5 \geq 0$ | $x_6 \geq 0$ | $x_7 \geq 0$ | |

$x_5$, $x_6$, $x_7$ — свободные переменные; $y_1$ — искусственная переменная.

**Задача 2.** Используя геометрические построения, найти решение следующей задачи линейного программирования.

Целевая функция: $F = 9/2\, x_1 + x_2$.

Ограничения:

| | | | |
|---|---|---|---|
| $x_1$ | $+4x_2$ | $\geq 7$ | (1) |
| $3x_1$ | $+x_2$ | $\geq 8$ | (2) |
| $3x_1$ | $+2x_2$ | $\geq 11$ | (3) |
| $x_1 \geq 0$ | $x_2 \geq 0$ | | |

Характер экстремума: min.

*Замечание.*

$x_1 + 4x_2 = 7$; если $x_1 = 0$, тогда $x_2 = 7/4 = 1{,}75$; если $x_2 = 0$, тогда $x_1 = 7$.

$3x_1 + x_2 = 8$; если $x_1 = 0$, тогда $x_2 = 8$; если $x_2 = 0$, тогда $x_1 = 8/3 \approx 2{,}667$.

$3x_1 + 2x_2 = 11$; если $x_1 = 0$, тогда $x_2 = 11/2 = 5{,}5$; если $x_2 = 0$, тогда $x_1 = 11/3 \approx$ $\approx 3{,}667$.

Оптимальными являются такие значения, при которых целевая функция минимальна и выполняются указанные ограничения.

$F = 9/2\, x_1 + x_2$.

$10 = 9/2x_1 + x_2$; если $x_1 = 0$, тогда $x_2 = 10$; если $x_2 = 0$, тогда $x_1 = 10/9 \cdot 2 \approx 2{,}222$.

$4{,}5 = 9/2\, x_1 + x_2$; если $x_1 = 0$, тогда $x_2 = 4{,}5$; если $x_2 = 0$, тогда $x_1 = 4{,}5/9 \cdot 2 = 1$.

Следовательно, оптимальное значение можно получить, решив совместно уравнения:

| | | |
|---|---|---|
| $3x_1$ | $+x_2$ | $=8$ |
| $x_1$ | | $=0$ |

**Задача 3**. Составить математическую модель задачи.

Для сохранения здоровья и работоспособности человек должен потреблять в сутки питательных веществ:

$B1$ — не менее 4 ед., $B2$ — не менее 6 ед.,

$B3$ — не менее 9 ед., $B4$ — не менее 6 ед.

Имеется два вида пищи — $P1$ и $P2$.

В 1 кг пищи $P1$ питательных веществ: $B1$ — 1, $B2$ — 0, $B3$ — 1, $B4$ — 3 ед.

В 1 кг пищи $P2$ питательных веществ: $B1$ — 1, $B2$ — 3, $B3$ — 3, $B4$ — 2 ед.

1 кг $P1$ стоит 3 ден. ед., 1 кг $P2$ — 2 ден. ед.

Требуется так организовать питание, чтобы стоимость его была наименьшей, а организм получал бы суточную норму питательных веществ.

*Замечание.*
Соответствующие показатели можно представить в виде таблицы:

| Вещество | Продукт $P1$ | Продукт $P2$ | Необходимый объем веществ |
|---|---|---|---|
| $B1$ | 1 | 1 | $B1 \geq 4$ |
| $B2$ | 0 | 3 | $B2 \geq 6$ |
| $B3$ | 1 | 3 | $B3 \geq 9$ |
| $B4$ | 3 | 2 | $B4 \geq 6$ |
| Цена | 3 | 2 | min |

Пусть $x_1$ — количество продукта $P1$, а $x_2$ — количество продукта $P2$, тогда математическая модель задачи запишется следующим образом:

найти минимум целевой функции $F = 3x_1 + 2x_2$ при ограничениях:

| | | | | |
|---|---|---|---|---|
| $1x_1$ | $+1x_2$ | $\geq 4$ | Для вещества $B1$ | (1) |
| $0x_1$ | $+3x_2$ | $\geq 6$ | Для вещества $B2$ | (2) |
| $1x_1$ | $+3x_2$ | $\geq 9$ | Для вещества $B3$ | (3) |
| $3x_1$ | $+2x_2$ | $\geq 6$ | Для вещества $B4$ | (4) |
| $x_1 \geq 0$ | $x_2 \geq 0$ | | | |

### *Задание 19. Решить задачи симплекс-методом с точностью* $\varepsilon$ = **0,01**

Для производства шкафов требуется три вида заготовок из фанеры. Каждый лист фанеры можно разрезать на заготовки разными способами. На день работы цеха требуется $X$ заготовок вида 1; $Y$ заготовок вида 2 и $Z$ заготовок вида 3. Определить количество фанерных листов, которое требуется доставить в цех, чтобы удовлетворить потребности производства, но при этом иметь минимальное количество отходов.

| Номер варианта | $X$ | $Y$ | $Z$ | Вид раскроя | Число заготовок | | | Отходы |
|---|---|---|---|---|---|---|---|---|
| | | | | | $X$ вид1 | $Y$ вид2 | $Z$ вид3 | |
| **1** | 10 | 15 | 6 | 1 | 4 | 4 | 0 | 1 |
| **2** | 6 | 22 | 13 | 2 | 3 | 3 | 2 | 5 |
| **3** | 19 | 50 | 40 | 3 | 1 | 6 | 1 | 3 |
| **4** | 10 | 13 | 27 | 4 | 1 | 1 | 10 | 0 |
| **5** | 13 | 25 | 30 | 1 | 3 | 2 | 4 | 4 |
| **6** | 6 | 71 | 10 | 2 | 3 | 3 | 2 | 1 |
| **7** | 10 | 30 | 50 | 3 | 0 | 5 | 5 | 3 |
| **8** | 19 | 29 | 10 | 4 | 4 | 3 | 1 | 1 |
| **9** | 9 | 13 | 64 | 1 | 10 | 3 | 5 | 4 |
| **10** | 11 | 9 | 16 | 2 | 3 | 3 | 30 | 2 |
| **11** | 53 | 17 | 40 | 3 | 20 | 0 | 1 | 5 |
| **12** | 111 | 10 | 10 | 4 | 4 | 10 | 0 | 1 |
| **13** | 10 | 10 | 50 | 1 | 5 | 5 | 3 | 4 |
| **14** | 13 | 13 | 13 | 2 | 3 | 8 | 3 | 2 |
| **15** | 50 | 40 | 30 | 3 | 4 | 4 | 10 | 0 |
| **16** | 6 | 17 | 25 | 4 | 0 | 10 | 5 | 1 |

## Задание 20. Найти абсолютную и относительную погрешности вычисления функции при заданных значениях аргумента

| Номер варианта | Функция | Аргументы |
|---|---|---|
| 1 | $S = 1/2\, a^2 b \sin c$ | $a \approx 17{,}25 \pm 0{,}01$<br>$b \approx 34{,}625 \pm 0{,}005$<br>$c \approx 18{,}3 \pm 0{,}1$ |
| 2 | $V = 1/6\, h^3 + 1/2\, \pi (r_1^2 + r_2^3) h$ | $h \approx 1{,}52 \pm 0{,}05$<br>$r_1 \approx 6{,}28 \pm 0{,}02$<br>$r_2 \approx 5{,}137 \pm 0{,}003$ |
| 3 | $y = A \exp(-\alpha x) \sin(\omega x + \varphi_0)$ | $A \approx 49{,}83 \pm 0{,}01$<br>$\alpha \approx 2{,}35 \pm 0{,}01$<br>$\omega \approx 11{,}7 \pm 0{,}1$<br>$\varphi_0 \approx 3{,}147 \pm 0{,}001$<br>$x \approx 1{,}78 \pm 0{,}01$ |
| 4 | $y = a \operatorname{tg} \varphi + l \sin \varphi$ | $a \approx 2{,}435 \pm 0{,}005$<br>$l \approx 1{,}27 \pm 0{,}01$<br>$\varphi \approx 1{,}372 \pm 0{,}01$ |
| 5 | $L = a\sqrt{1 + \lambda^2 - 2\lambda \cos t}$ | $a \approx 9{,}14 \pm 0{,}05$<br>$\lambda \approx 1{,}2 \pm 0{,}2$<br>$t \approx 0{,}56 \pm 0{,}01$ |
| 6 | $y = a \ln \dfrac{a + \sqrt{a^2 - x^2}}{x}$ | $a \approx 5{,}93 \pm 0{,}05$<br>$x \approx 3{,}1415 \pm 0{,}0001$ |
| 7 | $S = \dfrac{\pi a^2}{2} + \pi l^2$ | $a \approx 52{,}34 \pm 0{,}01$<br>$l \approx 2{,}043 \pm 0{,}001$ |
| 8 | $v = \sin(a^3) \sin \dfrac{b+1}{b}$ | $a \approx 0{,}235 \pm 0{,}003$<br>$b \approx 48{,}39 \pm 0{,}01$ |
| 9 | $A = F\, l \cos \alpha$ | $F \approx 12{,}23 \pm 0{,}02$<br>$l \approx 34{,}6 \pm 0{,}1$<br>$\alpha \approx 0{,}28 \pm 0{,}01$ |
| 10 | $\xi = \dfrac{A}{\sqrt{r}} \sin(\omega t - kr)$ | $A \approx 4{,}1 \pm 0{,}2$<br>$r \approx 2{,}13 \pm 0{,}01$<br>$\omega \approx 12{,}778 \pm 0{,}003$<br>$t \approx 0{,}13 \pm 0{,}02$<br>$k \approx 5{,}6 \pm 0{,}1$ |

### *Задание 21. Решить задачи оптимизации*

**Задача 1.** Завод выпускает обычные станки и станки с программным управлением, затрачивая на один обычный станок 200 кг стали и 200 кг цветного металла, а на один станок с программным управлением 700 кг стали и 100 кг цветного металла. Завод может израсходовать в месяц до 46 т стали и до 22 т цветного металла. Сколько станков каждого типа должен выпустить за месяц завод, чтобы объем реализации был максимальным, если один обычный станок стоит 2000 ден. ед., а станок с программным управлением 5000 ден. ед.

**Задача 2.** Для производства двух видов изделий $A$ и $B$ используется три типа технологического оборудования. На изготовление одного изделия $A$ оборудование первого типа используется в течение 5 ч, второго — в течение 3 ч и третьего — 2 ч. На производство одного изделия $B$ расходуется соответственно: 2, 3 и 3 ч. В плановом периоде оборудование первого типа может быть использовано в течение 505 ч, второго — 394 ч и третьего — 348 ч. Прибыль от реализации одного изделия $A$ составляет 7 ден. ед., $B$ — 4 ден. ед. Составить план производства, максимизирующий прибыль предприятия.

**Задача 3.** Для изготовления изделий $A$ и $B$ предприятие использует три вида сырья. На производство одного изделия $A$ требуется сырья первого вида 15 кг, второго — 11 кг, третьего — 9 кг, а на производство одного изделия $B$, соответственно, 4, 5 и 10 кг. Сырья первого вида имеется 1095 кг, второго — 865 кг, третьего — 1080 кг. Составить план производства, максимизирующий прибыль, если прибыль от реализации единицы изделия $A$ составляет 3 ден. ед., $B$ — 2 ден. ед.

**Задача 4.** Для производства изделий $A$ и $B$ используется три вида оборудования. При изготовлении одного изделия $A$ оборудование первого вида занято 7 ч, второго — 6 ч и третьего — 1 ч. При изготовлении одного изделия $B$ расходуется, соответственно, 3, 3 и 2 ч. В месяц оборудование первого вида может быть занято 1365 ч, второго — 1245 ч и третьего — 650 ч. Составить план производства, максимизирующий прибыль, если прибыль от реализации одного изделия $A$ составляет 6 ден. ед., изделия $B$ — 5 ден. ед.

**Задача 5.** Для изготовления изделий $A$ и $B$ используется три вида сырья. На изготовление одного изделия $A$ требуется 9 кг сырья первого вида, 6 кг сырья второго вида и 3 кг сырья третьего вида. На изготовление одного изделия $B$ требуется, соответственно, 4, 7 и 8 кг сырья. Производство обеспечено сырьем первого вида в количестве 801 кг, второго — 807 кг, третьего — 703 кг. Прибыль от продажи изделия $A$ составляет 3 ден. ед., изделия $B$ — 2 ден. ед. Составить план производства, максимизирующий прибыль.

**Задача 6.** Завод выпускает два вида редукторов. На изготовление одного редуктора первого вида расходуется 4 т чугуна и 1 т стали, а на изготовление одного редуктора второго вида — 2 т чугуна и 1 т стали. Завод располагает на месяц 160 т чугуна и 120 т стали. Составить месячный план производства редукторов, максимизирующий прибыль завода, если прибыль от продажи одного редуктора первого вида равна 400 ден. ед., а второго — 200 ден. ед.

**Задача 7.** Для производства изделий $A$ и $B$ используются три вида станков. На производство одного изделия $A$ требуется 6 ч работы станка первого вида, 4 ч работы станка второго вида и 3 ч работы станка третьего вида. На производство одного изделия $B$ требуется 2 ч работы станка первого вида, 3 ч работы станка второго вида и 4 ч работы станка третьего вида. Месячный ресурс работы всех станков первого вида, имеющихся на заводе равен 600 ч, всех станков второго вида — 520 ч и всех станков третьего вида — 600 ч. Прибыль от реализации одного изделия $A$ составляет 6 ден. ед., изделия $B$ — 3 ден. ед. Составить план производства на месяц, максимизирующий прибыль предприятия.

**Задача 8.** На ферме разводят нутрий и кроликов. В недельный рацион нутрий входят 17 кг белков, 11 кг углеводов и 5 кг жиров, а для кроликов эти нормы, соответственно, равны 13, 15 и 7 кг. Доход от реализации одного кролика составляет 20 ден. ед., а от реализации одной нутрии — 25 ден. ед. Найти план разведения животных, максимизирующий доход фермы, если ферма не может расходовать в неделю более 184 кг белков, 152 кг углеводов и 70 кг жиров.

**Задача 9.** Для изготовления изделий $A$ и $B$ предприятие использует три вида сырья. На производство одного изделия $A$ тре-

буется 12 кг сырья первого вида, 10 — второго и 3 — третьего, а на производство одного изделия $B$, соответственно, 3, 5 и 6 кг. Производство обеспечено сырьем первого вида в количестве 684 кг, второго — 690 кг и третьего 558 кг. Одно изделие $A$ дает предприятию 6 ден. ед. прибыли, изделие $B$ — 2 ден. ед. Составить план производства, максимизирующий прибыль предприятия.

**Задача 10.** Мастерская по покраске кузовов автомобилей рассчитана на покраску не более 160 кузовов в месяц. На покраску кузова автомобиля «Москвич» краски расходуется 4 кг, а кузова «Волги» — 7 кг. Мастерская располагает 820 кг краски на месяц. Составить месячный план покраски автомобилей, максимизирующий прибыль мастерской, если покраска одного «Москвича» дает 30 ден. ед. прибыли, а одной «Волги» — 40 ден. ед. прибыли.

**Задача 11.** В одном районе расположены четыре населенных пункта. По территории района проходит железная дорога. По просьбе жителей района планируется построить железнодорожную станцию и проложить дороги от нее до населенного пункта. Требуется определить наиболее удобное расположение железнодорожной станции. (Место для станции надо выбрать так, чтобы наибольшее из расстояний от нее до населенных пунктов было как можно меньше.)

**Задача 12.** На заданном расстоянии от пушки находится стена. Известны угол наклона пушки и начальная скорость снаряда. Попадет ли снаряд в стену?

**Задача 13.** Расположенный на берегу реки металлургический завод осуществил сброс сточных вод, в результате чего концентрация вредных веществ в реке резко увеличилась. С течением времени эта концентрация, естественно, уменьшается. Требуется сообщить, каков будет уровень загрязнения реки через сутки, двое суток и т. д. до тех пор, пока концентрация не станет меньше предельно допустимой.

**Задача 14.** Бетон, производимый на заводах $A$ и $B$, нужно развозить по трем стройплощадкам: $C1$, $C2$ и $C3$. Известны потребности стройплощадок в бетоне, запасы бетона на каждом заводе и затраты на перевозку 1 т бетона от каждого завода до ка-

ждой стройплощадки. Требуется составить такой план перевозок, который обеспечивал бы наименьшие затраты.

**Задача 15.** Для полива трех полей колхоз использует насосную станцию. На первое поле требуется подать не менее 200 кубометров воды в сутки, на второе — не менее 300, на третье не менее 350. В распоряжении колхоза 1200 кубометров воды в сутки. Стоимость подачи $q$ кубометров воды на первое поле $1570q$ руб., на второе поле $1720q$ руб., на третье $1930q$ руб. Сколько кубометров воды надо подать на каждое поле, чтобы затраты были наименьшими?

**Задача 16.** Для производства вакцины на заводе планируется выращивать культуру бактерий. Известно, что если масса бактерий — $x$ г, то через день она увеличится на $(a - bx)x$ г, где коэффициенты $a$ и $b$ зависят от вида бактерий. Как изменяется масса бактерий через 1, 2, 3, ..., 365 дней (до конца года)?

**Задача 17.** На острове живут зайцы и волки. Экологи установили такую закономерность: если в начале года количество зайцев равно $x$, а количество волков — $y$, то через год число зайцев будет равно $x + (4 - 0{,}001y - 0{,}0001x)x$, а число волков будет равно $y + (-0{,}03 + 0{,}003x)y$. Сколько зайцев и волков будет на острове через год, два, три и т. д.? При каких начальных значениях $x$ и $y$ на острове исчезнут волки и зайцы?

# Заключение

Информационные и коммуникационные технологии решительно вторгаются в научно-практическую и образовательную деятельность. Стремительно повышаются требования к уровню подготовки в этой сфере специалистов различных областей. В этой связи изменяется программа обучения, во все большей степени отражающая прикладной, практический подход к применению знаний. Возрастает роль дисциплины «Численные методы», которая развивает идеи численного решения задач, возникающих в процессе компьютерного математического моделирования реальных явлений в различных предметных сферах.

Главная особенность обучения основам численных методов, которая все отчетливее проявляется в последние годы, связана с интенсификацией процессов использования различных специализированных математических пакетов и систем программирования вычислительных методов как инструмента решения прикладных задач. В связи с этим явное включение в содержание дисциплины вопросов, раскрывающих применение современных информационных технологий в прикладной математике, является необходимым требованием времени.

Теория приближенного решения математических задач постоянно пополняется все более совершенными численными методами, появление которых стимулируется как особенностями машинной математики, так и расширением функциональных возможностей прикладных программных средств. Все это требует определенного уровня понимания, который необходимо обеспечить при обучении численным методам. Понятия о приближенных методах решения прикладных задач подготавливают студентов к разработке и применению с помощью ПЭВМ вычислительных алгоритмов решения математических задач, возникающих в процессе познания и использования в практической деятельности законов реального мира, посредством математического моделирования.

Учебное пособие способствует выработке практических навыков решения прикладных задач, поэтому включенные в него задания представлены как предметные задачи, описывающие реальный объект или процесс.

# Литература


1. *Еннер Р. А., Пустоваченко Н. Н., Фролова В. И.* Методы прикладной математики: методическое пособие для учителя информатики и математики. Мурманск, 2001.

2. *Левицкий А. А.* Информатика. Основы численных методов. Лабораторный практикум. Красноярск: ИПЦ КГТУ, 2005.

3. *Колдаев В. Д.* Основы алгоритмизации и программирования: учеб. пособие / Под ред. проф. Л. Г. Гагариной. М.: ИД «ФОРУМ»: ИНФРА-М, 2006.

4. *Поршев С. В.* Вычислительная математика. Курс лекций. СПб.: БХИ-Санкт-Петербург, 2003.

5. *Приклонский В. И.* Численные методы в физике. М.: МГУ, физфак, 2001.

6. *Шатрова Л. Н.* Вычислительные методы. Методические указания к лабораторным и практическим занятиям. Киров, 2001.

7. *Воробьева Г. Н., Данилова А. Н.* Практикум по вычислительной математике: учеб. пособие для техникумов. 2-е изд., перераб. М.: Высш. шк., 1990.

8. *Касаткин В. Н., Верлань А. Ф.* Основы информатики и вычислительной техники: проб. учеб. пособие для 10—11 кл. сред. шк. К.: Рад. шк., 1989.

9. *Бахвалов Н. С., Лапин А. В., Чижонков Е. В.* Численные методы в.задачах и упражнениях. М.: Высш. шк., 2000.

10. *Вержбицкий В. М.* Численные методы. Линейная алгебра и нелинейные уравнения. М.: Высш. шк., 2000.

11. *Чернова Н. М.* Лабораторный практикум по курсу «Численные методы» / Международный педагогический университет. Магадан: Изд. МПУ, 1996.

12. *Левицкий А. А.* MATLAB 3.05, MathCAD 2.5. Практическое руководство. Красноярск, 1997.

13. *Курицкий Б. Я.* Поиск оптимальных решений средствами Excel 7.0. СПб.: BHV-Санкт-Петербург, 1997.

14. *Заварыкин В. М.*, *Житомирский В. Г.*, *Лапчик М. П.* Численные методы: учеб. пособие для студентов физ.-мат. спец. пединститутов. М.: Просвещение, 1990.

15. *Бахвалов Н. С.*, *Жидков Н. П.*, *Кобельков Г. М.* Численные методы. М.: Бином. Лабораторные знания, 2003.

16. Численные методы: учебник для техникумов / И. И. Данилина и др. М.: Высш. шк., 1976.

17. *Колдаев В. Д.*, *Павлова Е. Ю.* Сборник задач и упражнений по информатике: учеб. пособие / Под ред. проф. Л. Г. Гагариной. М.: ИД «ФОРУМ»: ИНФРА-М, 2007.

# Справочная информация по математике

## *Арифметическая прогрессия*

Арифметической прогрессией называется такая последовательность $\{a_n\}$, у которой каждый ее член, начиная со второго, равен предшествующему члену, сложенному с одним и тем же (определенным для данной последовательности) числом $d$, называемым разностью прогрессии:

$$a_{n+1} = a_n + d, \quad n = 1, 2, \ldots .$$

Формула общего члена прогрессии будет иметь вид:

$$a_n = a_1 + (n-1)d, \quad n = 2, 3, \ldots ,$$

таким образом, арифметическая прогрессия полностью определяется своим первым членом $a_1$ и разностью $d$. Формула для суммы $S_n$ первых $n$ членов будет иметь вид:

$$S_n = \frac{a_1 + a_n}{2} n \quad \text{или} \quad S_n = \frac{2a_1 + (n-1)d}{2} n.$$

## *Геометрическая прогрессия*

Числовую последовательность, первый член которой отличен от нуля, а каждый член, начиная со второго, равен предшествующему, умноженному на одно и то же отличное от нуля число $q$, называют *геометрической прогрессией*. Это число $q$ называют знаменателем геометрической прогрессии:

$$b_n = b_{n-1} q.$$

Для геометрической прогрессии $\{b_n\}$ имеем:

$$b_n = b_1 q^{n-1}; \quad S_n = \frac{b_n q - b_1}{q-1} = \frac{b_1 (q^n - 1)}{q-1},$$

где $q \neq 1$ — знаменатель прогрессии; $S_n$ — сумма ее первых $n$ членов.

Чтобы задать геометрическую прогрессию, достаточно задать ее первый член и знаменатель прогрессии.

### *Соотношения между функциями одного аргумента*

$$\arcsin a = -\arcsin(-1) = \frac{\pi}{2} - \arccos a = \operatorname{arctg} \frac{a}{\sqrt{1-a^2}};$$

$$\arccos a = \pi - \arccos(-a) = \frac{\pi}{2} - \arcsin a = \operatorname{arcctg} \frac{a}{\sqrt{1-a^2}};$$

$$\operatorname{arctg} a = -\operatorname{arctg}(-a) = \frac{\pi}{2} - \operatorname{arcctg} a = \arcsin \frac{a}{\sqrt{1+a^2}};$$

$$\operatorname{arcctg} a = \pi - \operatorname{arcctg}(-a) = \frac{\pi}{2} - \operatorname{arctg} a = \arccos \frac{a}{\sqrt{1+a^2}};$$

$$\arcsin a = \arccos \sqrt{1-a^2};$$

$$\arccos a = \arcsin \sqrt{1-a^2};$$

$$\operatorname{arctg} a = \operatorname{arcctg} \frac{1}{a} = \arcsin \frac{a}{\sqrt{1+a^2}} = \arccos \frac{1}{\sqrt{1+a^2}};$$

$$\arcsin a + \arccos a = \frac{\pi}{2};$$

$$\operatorname{arctg} a + \operatorname{arcctg} a = \frac{\pi}{2}.$$

### *Модуль действительного числа. Свойства модулей*

$$|a| = \begin{cases} a, & \text{если } a \geq 0; \\ -a, & \text{если } a < 0; \end{cases} \qquad |a| \geq 0; \qquad |a| = |-a|;$$

$$|ab| = |a| \cdot |b|; \qquad |a|^2 = a^2 \qquad b \neq 0; \qquad |a+b| \leq |a| + |b|;$$

$$\left|\frac{a}{b}\right| = \frac{|a|}{|a|}; \qquad |a-b| \geq |a| - |b|.$$

## Некоторые постоянные

| | |
|---|---|
| $\pi = 3{,}1416$ | $\pi^2 = 9{,}8696$ |
| $2\pi = 6{,}2832$ | $\sqrt{\pi} = 1{,}7725$ |
| $\pi/2 = 1{,}5708$ | $\pi/180 = 0{,}0175$ |
| $\pi/3 = 1{,}0472$ | $(\pi/180)^2 = 0{,}0003$ |
| $1/\pi = 0{,}3183$ | $\sqrt{e} = 1{,}6487$ |
| $e = 2{,}7183$ | $1/e = 0{,}3679$ |
| $e^2 = 7{,}3891$ | $1/M = \ln 10 = 2{,}3026$ |
| $M = \lg e = 0{,}4343$ | $\ln 6 = 1{,}7918$ |
| $\ln 2 = 0{,}6931$ | $\ln 7 = 1{,}9459$ |
| $\ln 3 = 1{,}0986$ | $\ln 8 = 2{,}0794$ |
| $\ln 4 = 1{,}3863$ | $\ln 9 = 2{,}1972$ |
| $\ln 5 = 1{,}6094$ | $\sqrt{5} = 2{,}2361$ |
| $\sqrt{2} = 1{,}4142$ | $\sqrt{6} = 2{,}4495$ |
| $\sqrt{3} = 1{,}7321$ | $\frac{1}{7} - 0{,}1429$ |
| $\frac{1}{3} = 0{,}3333$ | $\frac{1}{3!} = \frac{1}{6} = 0{,}1667$ |

## Степени и корни

Степень с целым показателем:

$$a^n = a \cdot a \ \ldots \ a,$$

$$a^1 = a, \ \ a^0 = 1 \ \ (a \neq 0), \ \ a^{-n} = 1/a^n \ \ (a \neq 0),$$

$$a^m a^n = a^{m+n}, \ \ a^m/a^n = a^{m-n}, \ \ (a^m)^n = a^{mn},$$

$$(ab)^n = a^n b^n, \ \ (a/b)^n = a^n/b^n.$$

Корень $n$-й степени:

$$(\sqrt[n]{a})^n = a, \quad \sqrt[n]{ab} = \sqrt[n]{a}\sqrt[n]{b},$$

$$\sqrt[n]{a/b} = \sqrt[n]{a}/\sqrt[n]{b} \quad (b > 0), \quad \sqrt[n]{\sqrt[k]{a}} = \sqrt[nk]{a}.$$

$$(\sqrt{a})^2 = a, \quad \sqrt{a^2} = |a|.$$

Степень с дробным (рациональным) показателем:

$$a^{m/n} = \sqrt[n]{a^m}, \quad m \in Z, \quad n \in N, \quad n \geq 2, \quad a > 0.$$

Свойства степени с действительным показателем:

$$(a > 0, \quad b > 0, \quad x \in R, \quad y \in R)$$

$$a^x a^y = a^{x+y}, \quad (a^x)^y = a^{xy}, \quad (ab)^x = a^x b^x,$$

$$(a/b)^n = a^n/b^n, \quad a^x = b^{x \log_b a},$$

$$a^x = e^{x \ln a} = \exp(x \ln a), \quad a^x = 10^{x \lg a}.$$

### *Логарифмы*

$\log_a b$ ($a > 0$, $a \neq 1$, $b > 0$) — логарифм числа $b$ по основанию $a$.
Основное логарифмическое тождество: $a^{\log_a b} = b$.
$\lg b$ — десятичный логарифм (логарифм по основанию 10).
$\ln b$ — натуральный логарифм (логарифм по основанию $e$).
Переход от одного основания к другому:

$$\log_a b = \frac{\log_c b}{\log_c a}.$$

$$\log_a b = \frac{1}{\log_b a} = \frac{\lg b}{\lg a} = \frac{\ln b}{\ln a}, \quad \ln b = \frac{\lg b}{\lg a}.$$

Свойства логарифмов ($u$, $v > 0$):

$$\log_a a = 1, \quad \log_a 1 = 0, \quad \log_a(uv) = \log_a u + \log_a v,$$

$$\log_a \frac{1}{v} = -\log_a v, \quad \log_a \frac{u}{v} = \log_a u - \log_a v,$$

$$\log_a u^\alpha = \alpha \log_a u, \quad \log_a \sqrt[n]{u} = \frac{1}{n} \log_a u, \quad n \in N, \quad n \neq 1.$$

## Некоторые тождества

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2},$$

$$1^2 + 2^2 + 3^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6},$$

$$1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^2(n+1)^2}{4},$$

$$1 + 3 + 5 + \ldots + (2n-1) = n^2,$$

$$1^2 + 3^2 + 5^2 + \ldots + (2n-1)^2 = \frac{n(4n^2-1)}{3},$$

$$1^3 + 3^3 + 5^3 + \ldots + (2n-1)^3 = n^2(2n^2-1),$$

$$1 \cdot 2 + 2 \cdot 3 + \ldots + n(n+1) = \frac{n(n+1)(n+2)}{3},$$

$$1 \cdot 4 + 2 \cdot 7 + \ldots + n(3n+1) = n(n+1)^2,$$

$$1 \cdot 2 \cdot 3 + 2 \cdot 3 \cdot 4 + \ldots + n(n+1)(n+2) = \frac{1}{4}n(n+1)(n+2)(n+3),$$

$$1^2 - 2^2 + 3^2 - 4^2 + \ldots + (-1)^{n-1}n^2 = (-1)^{n-1}\frac{n(n+1)}{2},$$

$$\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \ldots + \frac{1}{n(n+1)} = \frac{n}{n+1},$$

$$\frac{1}{1 \cdot 3} + \frac{1}{3 \cdot 5} + \ldots + \frac{1}{(2n-1)(2n+1)} = \frac{n}{2n+1}.$$

## Формулы сокращенного умножения

$$a^2 - b^2 = (a-b)(a+b),$$

$$a^3 - b^3 = (a-b)(a^2 + ab + b^2),$$

$$a^3 + b^3 = (a+b)(a^2 - ab + b^2),$$

$$a^4 - b^4 = (a-b)(a^3 + a^2b + ab^2 + b^3) = (a-b)(a+b)(a^2 + b^2),$$

$$a^5 - b^5 = (a-b)(a^4 + a^3b + a^2b^2 + ab^3 + b^4),$$

$$a^5 + b^5 = (a+b)(a^4 - a^3b + a^2b^2 - ab^3 + b^4),$$

$$a^n - b^n = (a-b)(a^{n-1} + a^{n-2}b + \ldots + ab^{n-2} + b^{n-1}),$$

$$(a+b)^2 = a^2 + 2ab + b^2, \quad (a-b)^2 = a^2 - 2ab + b^2,$$
$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3,$$
$$(a-b)^3 = a^3 - 3a^2b + 3ab^2 - b^3.$$

### *Свойства числовых неравенств*

1. Если $a < b$, то при любом $c$ $a + c < b + c$.
2. Если $a < b$ и $c > 0$, то $ac < bc$.
3. Если $a < b$ и $c < 0$, то $ac > bc$.
4. Если $a < b$, $a$ и $b$ одного знака, то $1/a > 1/b$.
5. Если $a < b$ и $c < d$, то $a + c < b + d$, $a - d < b - c$.
6. Если $a < b$, $c < d$, $a > 0$, $b > 0$, $c > 0$, $d > 0$, то $ac < bd$.
7. Если $a < b$, $a > 0$, $b > 0$, то $a^2 < b^2$, $a^n < b^n$ $(n \in N)$.
8. Если $|a| < |b|$, то $a^2 < b^2$.

### *Некоторые неравенства*

1. Сравнение среднего геометрического и среднего арифметического неотрицательных чисел:

$$\sqrt{ab} \le (a+b)/2$$

(равенство лишь при $a = b$).

$$\sqrt{a_1 a_2 \ldots a_n} \le \frac{1}{n}(a_1 + a_2 + \ldots + a_n)$$

(равенство лишь при $a_1 = a_2 = \ldots = a_n$).

2. $\dfrac{2ab}{a^2 + b^2} \le 1$, $a, b > 0$ (равенство лишь при $a = b$).

3. Неравенство Буняковского:

$$(a_1b_1 + a_2b_2 + \ldots + a_nb_n)^2 \le (a_1^2 + a_2^2 + \ldots + a_n^2)(b_1^2 + b_2^2 + \ldots + b_n^2).$$

4. Неравенства с модулем:

$$|a+b| \le |a| + |b|, \quad |a-b| \le |a| + |b|,$$
$$|a-b| \ge |a| - |b|, \quad |a| \le b \Leftrightarrow -b \le a \le b.$$

## Основные формулы для гиперболических функций

*Определение гиперболических функций*

Гиперболический синус: $\operatorname{sh} x = \dfrac{e^x - e^{-x}}{2}$.

Гиперболический косинус: $\operatorname{ch} x = \dfrac{e^x + e^{-x}}{2}$.

Гиперболический тангенс: $\operatorname{th} x = \dfrac{\operatorname{sh} x}{\operatorname{ch} x} = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$.

Гиперболический котангенс: $\operatorname{cth} x = \dfrac{\operatorname{ch} x}{\operatorname{sh} x} = \dfrac{e^x + e^{-x}}{e^x - e^{-x}}$, $x \neq 0$.

*Основные тождества*

$$\operatorname{ch}^2 x - \operatorname{sh}^2 x = 1; \quad \operatorname{th} x \cdot \operatorname{cth} x = 1, \quad x \neq 0;$$

$$1 - \operatorname{th}^2 x = \frac{1}{\operatorname{ch}^2 x}; \quad \operatorname{cth}^2 x - 1 = \frac{1}{\operatorname{sh}^2 x}, \quad x \neq 0.$$

*Выражение гиперболических функций через одну из них*

Через $\operatorname{sh} x$:

$$\operatorname{ch} x = \sqrt{1 + \operatorname{sh}^2 x};$$

$$\operatorname{th} x = \frac{\operatorname{sh} x}{\sqrt{1 + \operatorname{sh}^2 x}}; \quad \operatorname{cth} x = \frac{\sqrt{1 + \operatorname{sh}^2 x}}{\operatorname{sh} x}, \quad x \neq 0.$$

Через $\operatorname{th} x$:

$$\operatorname{ch} x = \frac{\operatorname{th} x}{\sqrt{1 - \operatorname{th}^2 x}}; \quad \operatorname{ch} x = \frac{1}{\sqrt{1 - \operatorname{th}^2 x}};$$

$$\operatorname{cth} x = \frac{1}{\operatorname{th} x}, \quad x \neq 0.$$

## Тригонометрические формулы

$$\operatorname{tg}\alpha = \sin\alpha/\cos\alpha, \quad \alpha \neq \pi/2 + \pi n, \quad n \in Z,$$

$$\operatorname{ctg}\alpha = \cos\alpha/\sin\alpha, \quad \alpha \neq \pi n, \quad n \in Z,$$

$$\sec\alpha = 1/\cos\alpha, \quad \alpha \neq \pi/2 + \pi n, \quad n \in Z,$$

$$\operatorname{cosec}\alpha = 1/\sin\alpha, \quad \alpha \neq \pi n, \quad n \in Z.$$

$$\cos\alpha = \pm\sqrt{1 - \sin^2\alpha},$$

$$\operatorname{tg}\alpha = \frac{\sin\alpha}{\pm\sqrt{1 - \sin^2\alpha}}, \quad \operatorname{ctg}\alpha = \frac{\pm\sqrt{1 - \sin^2\alpha}}{\sin\alpha}.$$

$$\sin\alpha = \pm\sqrt{1 - \cos^2\alpha},$$

$$\operatorname{tg}\alpha = \frac{\pm\sqrt{1 - \cos^2\alpha}}{\cos\alpha}, \quad \operatorname{ctg}\alpha = \frac{\cos\alpha}{\pm\sqrt{1 - \cos^2\alpha}}.$$

## Планиметрия

*Некоторые обозначения*

$[AB]$ — отрезок с концами $A$ и $B$,

$|AB|$, $AB$ — длина отрезка $[AB]$,

$1^\circ$ — один градус, $1/180$ часть развернутого угла,

$1'$ — одна минута, $1' = (1/60)^\circ$,

$1''$ — одна секунда, $1'' = (1/60)'$,

1 рад — один радиан, 1 рад $= (180/\pi)^\circ \approx 57^\circ 17' 45''$,

$1^g$ — один град, $1/100$ прямого угла, $1^g = 0{,}9^\circ$.

*Связь между различными мерами угла*

$$\alpha^\circ = \alpha\,\frac{\pi}{180}\ \text{рад} = \left(\frac{10}{9}\alpha\right)^g.$$

*Треугольник*

Сумма внутренних углов:

$$\alpha + \beta + \gamma = \pi.$$

Теорема косинусов:

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha,$$

$$b^2 = a^2 + c^2 - 2ac \cdot \cos\beta,$$

$$c^2 = b^2 + a^2 - 2ab \cdot \cos\gamma.$$

Теорема синусов:

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} = 2R$$ ($R$ — радиус описанной окружности).

## *Основные формулы стереометрии*

**1. Произвольная призма** ($S$ — площадь основания; $H$ — высота; $V$ — объем):

$$V = SH.$$

**2. Прямая призма** ($P$ — периметр основания; $l$ — боковое ребро; $S_b$ — боковая поверхность):

$$S_b = Pl.$$

**3. Прямоугольный параллелепипед** ($a$, $b$, $c$ — его измерения; $d$ — диагональ):

$$V = abc; \quad d^2 = a^2 + b^2 + c^2.$$

**4. Куб** ($a$ — ребро):

$$V = a^3; \quad d = a\sqrt{3}.$$

**5. Произвольная пирамида** ($S$ — площадь основания; $H$ — высота; $V$ — объем):

$$V = \frac{1}{3}SH.$$

**6. Правильная пирамида** ($P$ — периметр основания; $l$ — апофема; $S_b$ — площадь боковой поверхности):

$$S_b = \frac{1}{2}Pl; \quad V = \frac{1}{3}SH.$$

**7. Произвольная усеченная пирамида** ($S_1$ и $S_2$ — площади оснований; $h$ — высота; $V$ — объем):

$$V = \frac{1}{3}h(S_1 + S_2 + \sqrt{S_1 S_2}).$$

**8. Цилиндр** ($R$ — радиус основания; $H$ — высота; $S_b$ — площадь боковой поверхности; $V$ — объем):

$$S_b = 2\pi RH; \quad V = \pi R^2 H.$$

**9. Конус** ($R$ — радиус основания; $H$ — высота; $l$ — образующая; $S_b$ — площадь боковой поверхности; $V$ — объем):

$$S_b = \pi Rl; \quad V = \frac{1}{3}\pi R^2 H.$$

**10. Шар, сфера** ($S$ — радиус шара; $S$ — площадь сферической поверхности; $V$ — объем):

$$S = 4\pi R^2; \quad V = \frac{4}{3}\pi R^3.$$

**11. Шаровой сегмент** ($R$ — радиус шара; $h$ — высота сегмента; $S$ — площадь сферической поверхности сегмента; $V$ — объем):

$$S = 2\pi Rh; \quad V = \pi h^2\left(R - \frac{1}{3}h\right).$$

**12. Шаровой сектор** ($R$ — радиус шара; $h$ — высота сегмента; $V$ — объем):

$$V = \frac{2}{3}\pi R^2 h.$$

## Таблица возникновения основных математических знаков

| Знак | Его значение | Кто ввел | Дата введения, год |
|---|---|---|---|
| + | Сложение | Я. Видман | Конец XV в. |
| — | Вычитание | Я. Видман | Конец XV в. |
| * | Умножение | У. Оутред | 1631 |
| . | Умножение | Г. Лейбниц | 1698 |
| : | Деление | Г. Лейбниц | 1684 |
| $a^2$, $a^3$, ..., $a^n$ | Степени | Р. Декарт | 1637 |
| $\sqrt{}$ | Корень | Х. Рудольф,<br>А. Жирор | 1525<br>1629 |
| Log, log | Логарифм | И. Кеплер | 1624 |
| sin | Синус | Б. Кавальери | 1632 |
| cos | Косинус | А. Эйлер | 1748 |
| tg | Тангенс | А. Эйлер | 1753 |
| arcsin, arctg | Арксинус, арктангенс | Ж. Лагранж | 1772 |
| $dx$, ..., $d^2x$ | Дифференциал | Г. Лейбниц | 1675 |
| $\int y\,dx$ | Интеграл | Г. Лейбниц | 1675 |
| $\frac{dx}{dy}$ | Производная | Г. Лейбниц | 1675 |
| $\int$ | Определенный интеграл | Ж. Фурье | 1819 |
| $\sum$ | Сумма | Л. Эйлер | 1755 |
| $k$ | Факториал | Х. Крамп | 1803 |
| Lim | Предел | У. Гамильтон | 1853 |
| $\lim\limits_{n\to\infty}$, $\lim\limits_{n\to\infty}$ | Предел | Многие математики | Начало XX в. |
| $f(x)$ | Функция | И. Бернулли, Л. Эйлер | 1718<br>1734 |
| $\infty$ | Бесконечность | Дж. Валлис | 1655 |
| $\pi$ | Отношение длины окружности к диаметру | У. Джонс,<br>Л. Эйлер | 1706<br>1736 |

*Окончание табл.*

| Знак | Его значение | Кто ввел | Дата введения, год |
|---|---|---|---|
| $x, y, z$ | Неизвестные величины | Р. Декарт | 1637 |
| $\rightarrow$ | Вектор | О. Коши | 1853 |
| $=$ | Равенство | Р. Рекорд | 1557 |
| $><$ | Больше, меньше | Т. Гарриот | 1631 |
| $\equiv$ | Сравнимость | К. Гаусс | 1801 |
| $\parallel$ | Параллельность | У. Оутред | 1677 |
| $\perp$ | Перпендикулярность | П. Эригон | 1634 |
| $\leq \geq$ | Нестрогие неравенства | П. Буге | 1734 |
| [ ] | Квадратные скобки | Р. Бомбелли | 1550 |
| ( ) | Круглые скобки | Н. Тарталья | 1556 |
| { } | Фигурные скобки | Ф. Виет | 1593 |
| $e$ | Основание натуральных логарифмов | Л. Эйлер | 1736 |
| $\equiv$ | Знак тождества | Б. Риман | 1857 |
| $\cap$ | Пересечение | Дж. Пеано | 1895 |

## *Приставки для обозначения кратных единиц*

| Множитель | Приставка | Обозначение приставки | |
|---|---|---|---|
| | | русское | международное |
| $10^{18}$ | экса | Э | E |
| $10^{15}$ | пета | П | P |
| $10^{12}$ | тера | Т | T |
| $10^{9}$ | гига | Г | G |
| $10^{6}$ | мега | М | M |
| $10^{3}$ | кило | к | k |
| $10^{2}$ | гекто | г | h |
| $10^{1}$ | дека | да | da |

*Окончание табл.*

| Множитель | Приставка | Обозначение приставки | |
|---|---|---|---|
| | | русское | международное |
| $10^{-1}$ | деци | д | d |
| $10^{-2}$ | санти | с | c |
| $10^{-3}$ | милли | м | m |
| $10^{-6}$ | микро | мк | μ |
| $10^{-9}$ | нано | н | n |
| $10^{-12}$ | пико | п | p |
| $10^{-15}$ | фемто | ф | f |
| $10^{-18}$ | атто | а | a |

## Старинные русские меры длины, массы и объема

| Единицы длины | Единицы массы | Единицы объема |
|---|---|---|
| 1 точка = 0,254 мм | 1 доля = 44,434940 мг | 1 чарка = 1/100 ведра = = 0,122994 дм$^3$ |
| 1 линия = 2,54 мм | 1 золотник = 4,265542 г | 1 бутылка водочная = 1/20 ведра = 0,61497 дм$^3$ |
| 1 дюйм = 2,54 см | 1 лот = 12,797262 г | 1 бутылка винная = 1/16 ведра = 0,768712 дм$^3$ |
| 1 вершок = 4,445 см | 1 фунт = 0,40951741 кг | 1 штоф = 1/10 ведра = 1,22994 дм$^3$ |
| 1 фут = 30,48 см | 1 пуд = 16,380496 кг | 1 ведро = 12,22994 дм$^3$ |
| 1 аршин = 0,7112 м | | 1 четверть = 0,262387 м$^3$ (для сыпучих материалов) |
| 1 сажень = 2,1336 м | | |
| 1 верста = 1066,8 м | | |
| 1 сотка = 2,1336 см | | |

# Решение математических задач в среде Excel

## *П2.1. Численное дифференцирование*

Известно, что численными (приближенными) методами производная функции в заданной точке может быть вычислена с использованием конечных разностей. Выражение, записанное в конечных разностях, для вычисления производной функции одного переменного имеет вид:

$$F(x) = \frac{\Delta F}{\Delta x} = \frac{F(x_{k+1}) - F(x_k)}{x_{k+1} - x_k}.$$

Рассмотрим методику вычисления производной.

Пусть требуется найти производную функции $Y = 2x^3 + x^2$ в точке $x = 3$. Производная, вычисленная аналитическим методом, равна 60.

Для вычисления производной в Excel выполните следующие действия:

1. Протабулируйте заданную функцию в окрестности точки $x = 3$ с достаточно малым шагом, например 0,001 (рис. П2.1)

2. В ячейку C2 введите формулу вычисления производной. Здесь ячейка B2 содержит значение $x_{k+1}$, ячейка A2 — $x_k$.

3. Скопируйте формулу до строки 7, получите значения производных в точках табуляции аргумента.

Для значения $x = 3$ производная функции равна значению 60,019, что близко к значению, вычисленному аналитически.

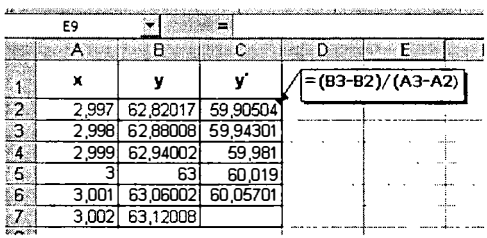



**Рис. П2.1.** Вычисления в Excel

## П2.2. Численное вычисление определенных интегралов

Для численного вычисления определенного интеграла методом трапеций используется формула

$$\int_k^n F(x) = \sum_{i=k}^{n} F(x_{i+1})(x_{i+1} - x_i) + \frac{(F(x_{i+1}) - F(x_i))(x_{i+1} - x_i)}{2}.$$

Методику вычисления определенного интеграла в Excel с использованием приведенной формулы рассмотрим на примере.

Пусть требуется вычислить определенный интеграл $\int_0^3 2x\,dx$.

Величина интеграла, вычисленная аналитически, равна 9. Для вычисления интеграла с использованием приведенной формулы выполните следующие действия:

- протабулируйте подынтегральную функцию в диапазоне изменения значений аргумента от 0 до 3 (рис. П2.2);
- в ячейку C3 введите формулу

  =(A3 – A2)*B2 + (A3 – A2)*(B3 – B2)/2 + C2,

  которая реализует подынтегральную функцию;

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | x | y | Частные суммы | | |
| 2 | 0 | 0 | | =(A3-A2)*B2+(A3-A2)*(B3-B2)/2+C2 | |
| 3 | 0,2 | 0,4 | 0,04 | | |
| 4 | 0,4 | 0,8 | 0,16 | | |
| 5 | 0,6 | 1,2 | 0,36 | | |
| 6 | 0,8 | 1,6 | 0,64 | | |
| 7 | 1 | 2 | 1 | | |
| 8 | 1,2 | 2,4 | 1,44 | | |
| 9 | 1,4 | 2,8 | 1,96 | | |
| 10 | 1,6 | 3,2 | 2,56 | | |
| 11 | 1,8 | 3,6 | 3,24 | | |
| 12 | 2 | 4 | 4 | | |
| 13 | 2,2 | 4,4 | 4,84 | | |
| 14 | 2,4 | 4,8 | 5,76 | | |
| 15 | 2,6 | 5,2 | 6,76 | | |
| 16 | 2,8 | 5,6 | 7,84 | | |
| 17 | 3 | 6 | 9 | | |

**Рис. П2.2.** Вычисления в Excel

- скопируйте формулу, записанную в ячейке C3 до значения аргумента $x = 3$. Вычисленное значение в ячейке C17 и будет величиной заданного интеграла — 9.

## П2.3. Нахождение экстремумов функций

Если функция $F(x)$ непрерывна на отрезке $[a; b]$ и имеет внутри этого отрезка локальный экстремум, то его можно найти, используя надстройку Excel **Поиск решения**.

Рассмотрим последовательность нахождения экстремума функции на следующем примере.

Пусть задана неразрывная функция $Y = X^2 + X + 2$. Требуется найти ее экстремум (минимальное значение).

Для решения задачи выполните действия:

- в ячейку A2 рабочего листа введите любое число, принадлежащее области определения функции, в этой ячейке будет находиться значение $X$;
- в ячейку B2 введите формулу, определяющую заданную функцию. Вместо переменной $X$ в этой формуле должна быть ссылка на ячейку A2:

$$= A2\text{^}2 + A2 + 2;$$

- выполните команду меню **Сервис/Поиск решения**;
- настройте параметры инструмента **Поиск решения**: число итераций — 1000, относительная погрешность 0,00001;
- в поле *Установить целевую ячейку* укажите адрес ячейки, содержащей формулу (A2), установите переключатель *Минимальному значению*, в поле *Изменяя ячейки* введите адрес ячейки, содержащей $X$ (A2);
- щелкните на кнопке Выполнить. В ячейке A2 будет помещено значение $X$ функции, при котором она имеет минимальное значение, а в ячейке $B2$ — минимальное значение функции.

Обратите внимание, что в окне Поиск решения можно устанавливать ограничения. Их целесообразно использовать, если функция многоэкстремальна, а нужно найти экстремум в заданном диапазоне изменения аргумента.

## П2.4. Решение систем линейных уравнений

Известно, что система линейных уравнений в матричном представлении записывается в виде

$$\mathbf{AX} = \mathbf{B}.$$

Решение такой системы записывается в виде

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B},$$

где $\mathbf{A}^{-1}$ — матрица, обратная по отношению к **A**.

В библиотеке Excel в разделе математических функций есть функции для выполнения операций над матрицами (табл. П2.1).

*Таблица П2.1.* **Операции над матрицами**

| Русифицированное имя функции | Англоязычное имя функции | Выполняемое действие |
|---|---|---|
| МОБР (параметр) | MINVERSE (parametr) | Обращение матрицы |
| МОПР (параметр) | MDETERM (parametr) | Вычисление определителя матрицы |
| МУМНОЖ (список параметров) | MMULT (parametrlist) | Умножение матриц |

Параметрами функций, приведенных в таблице, могут быть адресные ссылки на массивы, содержащие значения матриц, или имена диапазонов и выражения, например МОБР (A1: B2) или МОПР (матрица_1).

Например, пусть система уравнений задана матрицами:

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 4 & 5 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}.$$

Для решения задачи выполните действия:

- выделите диапазон размерностью $2 \times 2$ и присвойте ему имя A;
- выделите диапазон размерностью $1 \times 2$ и присвойте ему имя B;
- выделите диапазон размерностью $1 \times 2$ и присвойте ему имя X;
- используя список имен, выделите диапазон *A* и введите в него значения элементов матрицы A;
- используя список имен выделите диапазон *B* и введите в него значения элементов вектора B;
- используя список имен выделите диапазон *X* для помещения результата решения системы;
- в выделенный диапазон *X* введите формулу

= МУМНОЖ(МОБР(A);B);

- для выполнения операции над массивами нажмите комбинацию клавиш <Ctrl>+<Shift>+<Enter>, в ячейках диапазона $X$ будет получен результат: $x_1 = 2{,}16667$, $x_2 = -1{,}33333$.

Чтобы выполнить проверку полученных результатов, достаточно перемножить исходную матрицу на вектор результата, итогом этой операции является вектор свободных членов.

## *П2.5. Корни уравнения*

Используя возможности Excel можно находить корни нелинейного уравнения в допустимой области определения переменной. Последовательность операций нахождения корней следующая.

1. Уравнение представляется в виде функции одной переменной.

2. Производится табулирование функции в диапазоне вероятного существования корней.

3. По таблице фиксируются ближайшие приближения к значениям корней.

4. Используя средство Excel **Подбор параметра**, вычисляются корни уравнения с заданной точностью.

Пусть требуется найти все корни уравнения $X^3 - 0{,}01X^2 - 0{,}7044X + 0{,}139104 = 0$ на отрезке [–1; 1]. Правая часть уравнения представлена полиномом третьей степени, следовательно, уравнение может иметь не более трех корней.

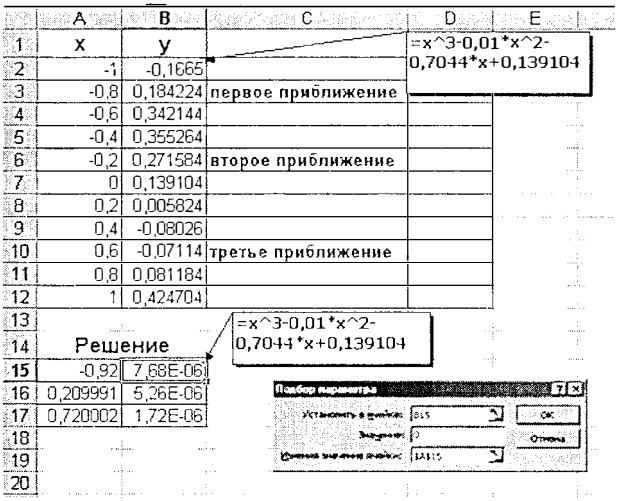
1. Представим уравнение следующим образом:

$$Y = X^3 - 0{,}01X^2 - 0{,}7044X + 0{,}139104.$$

2. Для локализации начальных приближений необходимо определить интервалы значений $X$, внутри которых значение функции пересекает ось абсцисс, т. е. функция меняет знак. С этой целью протабулируем функцию на отрезке [–1; +1] с шагом 0,2. Из полученной таблицы находим, что функция трижды пересекает ось $X$, следовательно, исходное уравнение имеет на заданном отрезке три корня.

3. Анализ таблицы показывает, что функция меняет знак в следующих интервалах значений аргумента $X$: [–1; –0,8], [–0,2; 0,4] и [0,6; 0,8], поэтому в качестве начальных приближений возьмем значения $X$: –0,8; –0,2 и 0,6.

4. На свободном участке рабочего листа, как показано на рис. П2.3, в ячейки A15: A17 введите начальные приближения, а в соответствующие ячейки столбца B скопируйте формулу.




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | x | y | | | |
| 2 | -1 | -0,1665 | | | |
| 3 | -0,8 | 0,184224 | первое приближение | | |
| 4 | -0,6 | 0,342144 | | | |
| 5 | -0,4 | 0,355264 | | | |
| 6 | -0,2 | 0,271584 | второе приближение | | |
| 7 | 0 | 0,139104 | | | |
| 8 | 0,2 | 0,005824 | | | |
| 9 | 0,4 | -0,08026 | | | |
| 10 | 0,6 | -0,07114 | третье приближение | | |
| 11 | 0,8 | 0,081184 | | | |
| 12 | 1 | 0,424704 | | | |
| 13 | | | | | |
| 14 | Решение | | | | |
| 15 | -0,92 | 7,68E-06 | | | |
| 16 | 0,209991 | 5,26E-06 | | | |
| 17 | 0,720002 | 1,72E-06 | | | |
| 18 | | | | | |
| 19 | | | | | |
| 20 | | | | | |

**Рис. П2.3.** Вычисления в Excel

5. Выполните команду меню **Сервис/Параметры**, во вкладке **Вычисления** установите относительную погрешность вычислений $E = 0{,}00001$, а число итераций $N = 1000$, установите флажок **Итерации**.

6. Выполните команду меню **Сервис/Подбор параметра**. В диалоговом окне заполните следующие поля:

***Установить в ячейке:*** в поле указывается адрес ячейки, в которой записана формула правой части функции;

***Значение:*** в поле указывается значение, которое должен получить полином в результате вычислений, т. е. правая часть уравнения (в нашем случае 0);

***Изменяя значение:*** в поле указывается адрес ячейки (где записано начальное приближение), в которой будет вычисляться корень уравнения и на которую ссылается формула.

После нажатия «ОК» получим значение первого корня: –0,92.

Выполнив последовательно операции, аналогичные предыдущим, вычислим значения остальных корней: –0,209991 и 0,720002.

## П2.6. Решение систем нелинейных уравнений

Применяя надстройку Excel **Поиск решения**, можно решать системы нелинейных уравнений. Предварительно система уравнений должна быть приведена к одному уравнению.

Пусть дана система двух уравнений:

$$\begin{cases} x^2 + y^2 = 3; \\ 2x + 3y = 1. \end{cases}$$

Требуется найти все корни приведенного уравнения для диапазона значений $x$ и $y$ [–3; 3].

***Шаг 1.*** Приведем систему к одному уравнению. Пара $(x, y)$ является решением системы тогда и только тогда, когда она является решением следующего уравнения с двумя неизвестными:

$$(x^2 + y^2 - 3)^2 + (2x + 3y - 1)^2 = 0.$$

***Шаг 2*** Для решения последнего уравнения необходимо найти начальные приближения, для этого протабулируем выражение, стоящее в левой части как функцию по двум переменным $x$ и $y$. Для табуляции функции выполните следующие действия:

- в столбец A введите последовательность значений $X$ с шагом 0,5, а в строку 3 — последовательность значений $Y$ также с шагом 0,5;
- присвойте диапазонам значений $X$ и $Y$ имена X и Y, соответственно;
- выделите диапазон ячеек, в котором будут вычисляться значения функции (B4:N16);
- в выделенный диапазон введите формулу

$$= (X\text{^}2 + Y\text{^}2 - 3)\text{^}2 + (2*X + 3*Y - 1)\text{^}2;$$

- нажав комбинацию клавиш [Ctrl]+[Shift]+[Enter], выполните операцию над выделенным массивом. В выделенном диапазоне появятся вычисленные значения функции.

***Шаг 3.*** Найдем начальные приближения. Поскольку табулируемая функция задает поверхность, то начальные приближения следует искать во впадинах, т. е. в точках, где функция принимает наименьшие значения. На рис. П2.6 эти точки затемнены.

Начальными приближениями являются пары (–1; 1) и (1,5; –0,5).

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | Решение системы нелинейных уравнений | | | | | | | | | | | | |
| 2 | | Y | | | | | | | | | | | | |
| 3 | X | -3 | -2,5 | -2 | -1,5 | -1 | -0,5 | 0 | 0,5 | 1 | 1,5 | 2 | 2,5 | 3 |
| 4 | -3 | 481,00 | 360,31 | 269,00 | 200,31 | 149,00 | 111,31 | 85,00 | 69,31 | 65,00 | 74,31 | 101,00 | 150,31 | 229,00 |
| 5 | -2,5 | 375,06 | 272,50 | 196,56 | 140,50 | 99,06 | 68,50 | 46,56 | 32,50 | 27,06 | 32,50 | 52,56 | 92,50 | 159,06 |
| 6 | -2 | 296,00 | 208,81 | 146,00 | 100,81 | 68,00 | 43,81 | 26,00 | 13,81 | 8,00 | 10,81 | 26,00 | 58,81 | 116,00 |
| 7 | -1,5 | 237,06 | 162,50 | 110,56 | 74,50 | 49,06 | 30,50 | 16,56 | 6,50 | 1,06 | 2,50 | 14,56 | 42,50 | 93,06 |
| 8 | -1 | 193,00 | 128,31 | 85,00 | 56,31 | 37,00 | 23,31 | 13,00 | 5,31 | 1,00 | 2,31 | 13,00 | 38,31 | 85,00 |
| 9 | -0,5 | 160,06 | 102,50 | 65,56 | 42,50 | 28,06 | 18,50 | 11,56 | 6,50 | 4,06 | 6,50 | 17,56 | 42,50 | 88,06 |
| 10 | 0 | 136,00 | 82,81 | 50,00 | 30,81 | 20,00 | 13,81 | 10,00 | 7,81 | 8,00 | 12,81 | 26,00 | 52,81 | 100,00 |
| 11 | 0,5 | 120,06 | 68,50 | 37,56 | 20,50 | 12,06 | 8,50 | 7,56 | 8,50 | 12,06 | 20,50 | 37,56 | 68,50 | 120,06 |
| 12 | 1 | 113,00 | 60,31 | 29,00 | 12,31 | 5,00 | 3,31 | 5,00 | 9,31 | 17,00 | 30,31 | 53,00 | 90,31 | 149,00 |
| 13 | 1,5 | 117,06 | 60,50 | 26,56 | 8,50 | 1,06 | 0,50 | 4,56 | 12,50 | 25,06 | 44,50 | 74,56 | 120,50 | 189,06 |
| 14 | 2 | 136,00 | 72,81 | 34,00 | 12,81 | 4,00 | 3,81 | 10,00 | 21,81 | 40,00 | 66,81 | 106,00 | 162,81 | 244,00 |
| 15 | 2,5 | 175,06 | 102,50 | 56,56 | 30,50 | 19,06 | 18,50 | 26,56 | 42,50 | 67,06 | 102,50 | 152,56 | 222,50 | 319,06 |
| 16 | 3 | 241,00 | 156,31 | 101,00 | 68,31 | 53,00 | 51,31 | 61,00 | 81,31 | 113,00 | 158,31 | 221,00 | 306,31 | 421,00 |
| 17 | | | | | | | | | | | | | | |
| 18 | Начальные приближени | | | | | | | | | | | | | |
| 19 | Первая | XX | YY | | Формула | | | | | | | | | |
| 20 | пара | -1 | 1 | | 1,00 | =(XX^2+YY^2-3)^2+(2*XX+3*YY-1)^2 | | | | | | | | |
| 21 | Вторая пара | 1,5 | -0,5 | | 0,50 | | | | | | | | | |
| 22 | | | | | | | | | | | | | | |

**Рис. П2.6.** Вычисления в Excel

Введите значения найденных приближений в смежные ячейки рабочего листа (рис. П2.6). Над столбцами сделайте надписи XX и YY, которые будут выполнять в формулах роль меток. Обратите внимание, что уже использованы имена X и Y, поэтому имена новых меток должны отличаться.

***Шаг 4.*** В ячейку строки, в которой записана первая пара X и Y, введите формулу, вычисляющую значение функции:

$$= (XX\text{^}2 + YY\text{^}2 - 3)\text{^}2 + (2*XX + 3*YY - 1)\text{^}2$$

и скопируйте ее в следующую строку.

***Шаг 5.*** Установите курсор на ячейку, в которой записана формула, и выполните команду меню **Сервис/Поиск решения**. Выполните настройку параметров инструмента Поиск решения: предельное число итераций — 1000, относительная погрешность 0,000001.

В окне **Поиск решения** в качестве целевой ячейки установите адрес ячейки, содержащей формулу, взведите переключатель *Минимальному значению*, в поле *Изменяя ячейки* укажите адрес диапазона, содержащего начальные приближения и щелкните на OK. В ячейках, где хранились начальные приближения будет получена первая пара корней. Повторите такие же операции для второй пары приближений.

Решением системы являются пары (–1,269; 1,1791) и (1,5764; –0,718).

# Вычислительные алгоритмы

## П3.1. Комбинаторные алгоритмы

### Перестановки

Перестановкой из $n$ элементов назовем упорядоченный набор из $n$ различных натуральных чисел, принадлежащих отрезку от 1 до $n$.

Например, пусть $n = 3$.

Перечислим все перестановки из трех элементов: (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1). Количество перестановок из $n$ элементов (обозначается как $P(n)$) равно $n! = 1 \cdot 2 \cdot \ldots \cdot n$ (факториалу числа $n$). Действительно, есть $n$ способов для выбора элемента на первое место, $(n - 1)$ способ для выбора элемента на второе место и т. д. Общее количество способов — это произведение $n \cdot (n - 1) \cdot (n - 2) \cdot \ldots \cdot 2 \cdot 1$.

### Размещения с повторениями

Пусть имеется множество элементов, относящихся к $n$ различным видам. Из них составляются всевозможные наборы по $m$ элементов в каждом. При этом в наборы могут входить и элементы одного вида. Два набора считаются различными, если они отличаются друг от друга или входящими в них элементами, или их порядком.

Например, пусть $n = 3$, $m = 2$. Размещения с повторениями: (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3). Число таких наборов $R(n, m) = n^m$.

### Размещения без повторений

Размещением без повторений из $n$ элементов по $m$ называется упорядоченный набор из $m$ различных чисел, принадлежащих отрезку от 1 до $n$. Два набора считаются различными, если они отличаются составом или порядком элементов.

Например, пусть $n = 4$, $m = 2$. Перечислим все размещения без повторений: (1, 2), (2, 1), (1, 3), (3, 1), (1, 4), (4, 1), (2, 3), (3, 2), (2, 4), (4, 2), (3, 4), (4, 3). Количество размещений $A(n, m)$ вычисляется по формуле $A(n, m) = n(n - 1) \cdot \ldots \cdot (n - m + 1)$ или

$$A(n, m) = n!/(n - m)!$$

### Сочетания

Сочетанием из $n$ элементов по $m$ называется набор из $m$ различных чисел, выбранных из диапазона от 1 до $n$. Наборы различаются лишь составом элементов, порядок элементов роли не играет.

Например, пусть $n = 5$, $m = 3$. Перечислим все сочетания: (1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5), (3, 4, 5). Каждому сочетанию соответствует $m!$ размещений. Обозначим число сочетаний $C(n, m)$. Таким образом

$$A(n, m) = C(n, m) \cdot m! \text{ или}$$

$$C(n, m) = A(n, m)/\mathrm{m}! = n!/\mathrm{m}!(n - m)!.$$

**Пример П3.1.** Составить программу подсчета числа сочетаний — $C(n, m)$.

***Решение***

Написание основной программы сводится к программированию формулы

$$C(n, m) = n!/m!(n - m)!$$

### Программа

```
Program Pr1;
  Var   n,m: Integer;    c: Longint;
    Function Fact(n:Integer):Longint;
    Var   i: Integer;    rez: Longint;
    Begin
      rez:=1;
      For i:=1 To n Do rez:=rez*i;
      Fact:=rez
    End;
```

```
Begin
  WriteLn('Введите n и m :');     ReadLn(n,m);
  c:=Fact(n) / (Fact(m) * Fact(n-m));
  WriteLn(c);
  Readln
End.
```

*Усовершенственная программа*

```
Program  Prlm;
  Var n,m:Integer;
    Function S(n,m:Integer):LongInt;
    Var  i:Integer;     rez,cht:LongInt;
    Begin
      rez:=l;    cht:=l;
      For i:=l To m Do Begin
              rez:=rez*i;
              cht:=cht*(n-i+l)
              end;
      S:=cht Div rez
    End;
    Begin
      WriteLn('Введите два числа: ');     ReadLn(n,m);
      WriteLn(S(n,m));
      ReadLn
    End.
```

**Пример П3.2.** Определить, является ли введенное число палиндромом, т. е. читается одинаково слева направо и справа налево.

**Программа**

```
{$R+}
Program  My25;
  Uses Crt;
  Const MaxN=1000;
  Type MyArray=Array[0..MaxN+1] Of Integer;
  {Числа храним в массиве}
  Var A,Z:MyArray;
      cnt,n:Integer;
  Function Eq:Boolean;
  {Является ли число палиндромом?}
  Var i:Integer;
```

```
  Begin
    i:=1;
    While (i<=A[0] div 2) And (A[i]=A[A[0]-i+1]) Do Inc(i);
    Eq:=(i>A[0] div 2)
  End;
  Procedure Swap(Var a,b:Integer);
    Var c:Integer;     Begin  c:=a; a:=b; b:=c  End;
  Procedure Rev;{ "Перевертываем" число.}
    Var i:Integer;
    Begin
      For i:=1 To A[0] Div 2 Do Swap(A[i],A[A[0]-i+1])
    End;
  Procedure Change(n:Integer);
  {Записываем число в массив.}
    Var i:Integer;
    Begin
      i:=1;
      While n<>0 Do Begin
        A[i]:=n Mod 10; n:=n Div 10;
        Inc(A[0]);Inc(i)
      End
    End;
Procedure Add;
{Складываем два больших числа.}
  Var i,r,w:Integer;
  Begin
    i:=1;r:=0;
    While (i<=A[0]) Or (r<>0) Do Begin
      w:=A[i]+Z[i]+r; A[i]:=w Mod 10;
      r:= w Div 10;
      If A[A[0]+1]<>0 Then Inc(A[0]);
      Inc(i)
    End
  End;
Begin
    ClrScr; n:=100;
    While (n<1000) Do Begin
    {Исследуем диапазон от 100 до 999.}
      FillChar(A,SizeOf(A),0);
      Change(n);{Преобразуем число.}
      cnt:=0;
      {Счетчик количества шагов — преобразований числа.}
      While (Not Eq) And (A[0]<=MaxN) Do Begin
```

```
    {Пока не палиндром и нет выхода за пределы массива A, считаем.}
      Z:=A;Inc(cnt);Rev;Add
    End;
    WriteLn(n,' ',cnt);
    {Вывод числа и количества шагов.}
    If cnt>=MaxN Then ReadLn;
    {Фиксируем факт выхода за пределы массива A.}
    Inc(n)
  End
ReadIn;
End.
```

## П3.2. Рекурсивные алгоритмы

*Рекурсией* называется ситуация, когда процедура или функция обращается к самой себе.

**Пример П3.3.** Проиллюстрировать последовательность рекурсивных вызовов на примере функции вычисления факториала числа.

**Функция**

```
Function Factorial(n: Integer): LongInt;
Begin
    If n=1 Then Factorial:=1
    Else Factorial:=n*Factorial(n-1)
End;
```

**Пример П3.4.** Определить, является ли заданное натуральное число простым.

Пусть требуется определить, верно ли, что заданное натуральное число $n$ не делится ни на одно число, большее или равное $m$, но меньшее $n$. При этом значения $m$ находятся в промежутке от 2 до $n$.

***Решение***

Утверждение истинно в двух случаях:

- если $m = n$;
- если $n$ не делится на $m$ и истинно утверждение для чисел от $(m + 1)$ до $(n - 1)$.

**Функция**

```
Function Simple(m,n: Integer): Boolean;
Begin
   If m=n Then Simple:=True
Else Simple:=(n Mod m<>0) And Simple(m+1,n)
   End;
```

*Замечание.* Первый вызов функции имеет вид Simple(2, *n*), где *n* — проверяемое число.

**Пример П3.5.** Написать рекурсивную функцию для вычисления наибольшего общего делителя двух чисел.

**Функция**

```
Function Nod(a,b:Integer):Integer;
  Begin
    If (a=0) or (b=0) Then Nod:=a+b
    Else If a>b Then Nod:=Nod(a-b,b)
      Else Nod:=Nod(a,b-a)
  End;
```

**Пример П3.6.** Написать функцию для возведения целого числа *a* в целую неотрицательную степень *n*.

**Функция**

```
Function Pow(a,n:Integer):Integer;
  Begin
    If n=0 Then Pow:=1
    Else If n Mod 2=0 Then Pow:=Pow(a*2,n Div 2)
      Else Pow:=Pow(a,n-1)*a
  End;
```

**Пример П3.7.** Составить функцию для получения чисел Фибоначчи.

***Решение***

Каждое число Фибоначчи, начиная с третьего, равно сумме двух предыдущих чисел, а первые два равны 1 (1, 1, 2, 3, 5, 8, 13, 21, ...). В общем виде *n*-е число Фибоначчи можно определить так:

$$\Phi(n) = \begin{cases} 1, & \text{если } n = 1 \text{ или } n = 2; \\ \Phi(n-1) + \Phi(n-2), & \text{если } n > 2. \end{cases}$$

**Функция**

```
Function Fib(n:Integer):Integer;
  Begin
    If n<=2 Then Fib:=1
      Else Fib:=Fib(n-1)+Fib(n-2)
  End;
```

**Пример П3.8**. Найти сумму первых $n$ членов арифметической (геометрической) прогрессии.

***Решение***

Арифметическая прогрессия определяется тремя параметрами: первым элементом $a$, разностью между соседними элементами $d$ и количеством членов прогрессии $n$. Очередной элемент прогрессии вычисляется по предыдущему прибавлением разности

$$a_{\text{новое}} = a_{\text{старое}} + d.$$

**Функция**

```
Function Sa(n,a:Integer):Integer;
  Begin
    If n>0 Then Sa:=a+Sa(n-1,a+d)
    Else Sa:=0
  End;
```

**Пример П3.9**. Написать рекурсивную процедуру генерации перестановок чисел от 1 до $n$.

***Решение***

Фиксируем на первом месте очередное число и генерируем все перестановки этого вида. При этом поступаем точно по такой же схеме. Фиксируем число на втором месте и генерируем все перестановки уже с фиксированными элементами на первом и втором местах.

Например, при $n = 3$ перестановки должны генерироваться в следующей последовательности:

1 2 3, 1 3 2, 2 1 3, 2 3 1, 3 2 1, 3 1 2.

**Процедура**

```
Procedure Swap(Var a,b:Integer);
Var c:Integer;
```

```
Begin c:=a; a:=b; b:=c  End;
Procedure Solve(t:Integer);
  Var i:Integer;
  Begin
    If t>=n Then Print
    Else For i:=t+1 To n Do Begin
         Swap(A[t+1],A[i]);
         Solve(t+1);
         Swap(A[t+1],A[i])
         End
  End;
```

**Пример П3.10.** Дано натуральное число $n$. Необходимо получить все разбиения числа $n$ на сумму слагаемых $n = a_1 + a_2 + ... + a_k$, где $k$, $a_1$, $a_2$, ..., $a_k > 0$.

***Решение***

Будем считать разбиения эквивалентными, если суммы отличаются только порядком слагаемых. Множество эквивалентных сумм представляем последовательностью $a_1$, $a_2$, ..., $a_k$, где $a_1 \geq a_2 \geq ... \geq a_k$.

При $n = 4$ разбиения $1 + 1 + 1 + 1$, $2 + 1 + 1$, $2 + 2$, $3 + 1$, $4$ перечислены в лексикографическом порядке. Для каждой пары соседних разбиений находится номер позиции, в которой число из второго разбиения больше числа из первого разбиения, а до этой позиции последовательности совпадают. Будем генерировать разбиение в порядке, обратном лексикографическому:

$$4,\ \ 3 + 1,\ \ 2 + 2,\ \ 2 + 1 + 1,\ \ 1 + 1 + 1 + 1.$$

Рекурсивная схема реализации просматривается достаточно легко.

Пусть в позиции $t$ записано число $a_t$, сумма чисел $a_1 + a_2 + ... + a_t = s$.

С позиции $(t + 1)$ генерируем все разбиения числа $(n - s)$, причем для элемента $A[t + 1]$ (и остальных) должно выполняться неравенство $A[t + 1] \leq A[t]$. Переход к следующему разбиению в позиции $t$ сводится, если это возможно, к вычитанию 1.

Разбиение хранится в глобальном массиве $A$. Количество элементов в разбиении различно. Значение переменной $t$ определяет текущий размер выводимой части массива $A$.

**Процедура**

```
Procedure Print(t:Integer);
  Var i:Integer;
  Begin
    For i:=1 To t Do Write(A[i]:3);
    WriteLn
  End;
```

Рекурсивная процедура Solve имеет два параметра: число $n$ и позицию $t$, начиная с которой необходимо получить все разбиения числа $n$. Первым разбиением является разбиение, соответствующее записи числа $n$ в позицию $t$. Последовательно записываем в позицию $t$ числа $(n-1)$, $(n-2)$, ..., и так до 1, а с позиции $t+1$ генерируем все разбиения чисел 1, 2, ..., $(n-1)$ соответственно.

**Процедура**

```
Procedure Solve(n,t:Integer);
  Var i,q:Integer;
  Begin
    If n=1 Then Begin A[t]:=1;Print(t) End
      Else Begin
      If n<=A[t-1] Then Begin
        A[t]:=n; Print(t) End;
        q:=Min(n-1,A[t-1]);
        For i:=q DownTo 1 Do Begin
          A[t]:=i;
          Solve(n-i,t+1)
          End
        End
  End;
```

При программировании потребуется функция Min:

```
Function Min(a,b:Integer):Integer;
Begin
    If a<b Then Min:=a Else Min:=b
End;
```

**Пример П3.11.** Составить программу решения задачи «Ханойская башня».

Задача «Ханойская башня» своим происхождением обязана легенде, в которой рассказывается, что в храме Бенареса на

бронзовой плите установлены три алмазных стержня, на один из которых Бог нанизал в виде пирамиды 64 золотых диска разных диаметров. С тех пор день и ночь монахи, сменяя друг друга, перекладывают по одному диску с одного стержня на другой согласно правилу: на диск большего диаметра нельзя класть диск меньшего диаметра. По преданию конец света наступит тогда, когда пирамида из 64 дисков будет перемещена на другой стержень.

Приведем программу, которая печатает последовательность переноса дисков посредством рекурсивной процедуры.

**Программа**

```
Program Pr1;
  Var n: Integer;
  Procedure MoveTown(High,FromI,ToI,WorkI: Integer);
    Begin
      If High>0 Then Begin
        MoveTown(High-1,FromI,WorkI,ToI);
        Writeln('Перенести кольцо №',High,' со стержня ',FromI,'
        на стержень ',ToI);
        MoveTown(High-1,WorkI,ToI,FromI)
      End
    End;
Begin
  Write('Количество колец = ');
  ReadLn(n);
  MoveTown(n,1,3,2)
End.
```

## *П3.3. Поиск простых чисел*

Простое число — это число, которое не делится ни на какое другое, кроме 1 и на само себя.

Существуют различные способы поиска простых чисел. Можно даже построить специальное просеивающее устройство, подобное промывным желобам, которые старатели применяют при поиске самородков, но так или иначе их приходится искать, потому что никто не знает, где они могут встретиться (рис. П3.1).

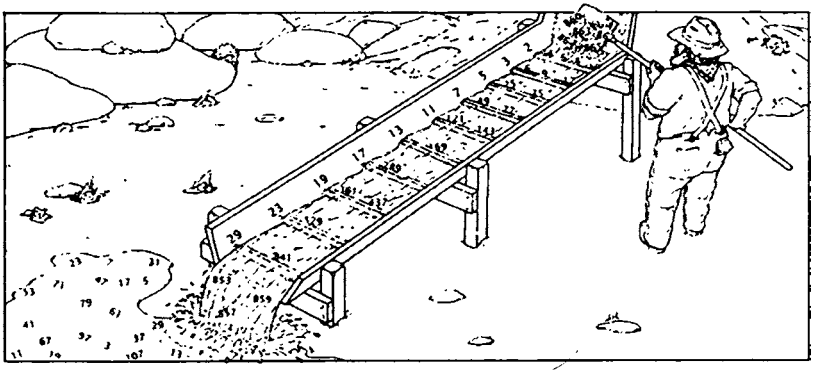


**Рис. П3.1.** Просеивающее устройство

Есть, правда, кое-какие геологические приметы, по которым можно искать их залежи.

В общем количество простых чисел до $n$ включительно приблизительно равно $n/\log n$. Посмотрим, как действует это правило, подставив в формулу несколько пробных значений $n$. Например, сколько простых чисел содержится в интервале от 1 до 100 и в интервале от 1 до 1000? В первом случае формула дает что-то около 22, во втором — около 145.

Станислав Улам открыл одну из таких закономерностей, случайно нарисовав решетку из горизонтальных и вертикальных линий. В одной из полученных таким образом клеток он поставил 1 и стал нумеровать остальные клетки по спирали, расходящейся от первой клетки:

**5 4 3**
**6 1 2**
**7 8 9**

Когда спираль совершила уже несколько оборотов, Улам начал обводить кружками простые числа, не преследуя никакой определенной цели. Однако вскоре заметил, как на его глазах возникает довольно любопытная закономерность. Откуда ни возьмись, стали появляться прямые линии. Улам, конечно, сразу понял, что такие линии говорят о закономерности, которую можно облечь в формулу для простых чисел. На рис. П3.2 составные числа представлены маленькими белыми квадратиками, а простые — черными.

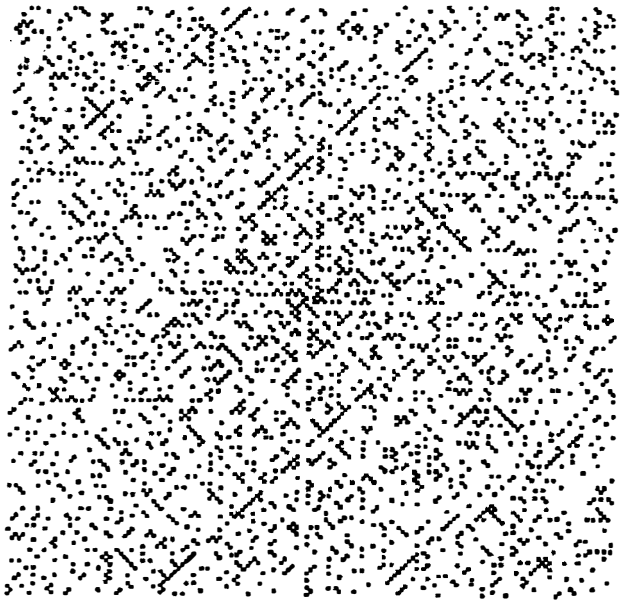


**Рис. П3.2.** Изображение множества простых чисел

Выделяющиеся на рис. П3.2 темные линии — это залежи простых чисел. Оказывается, эту последовательность можно описать квадратичной функцией

$$4x^2 + 4x - 1.$$

Алгоритм модели промывного желоба может послужить основой простейшей программы для определения простых чисел.

Можно предложить алгоритм, в котором все числа перебираются по очереди, причем для каждого из них производится проверка, является ли оно простым.

Еще в Древней Греции был известен более эффектный алгоритм, носящий название «решето Эратосфена».

Согласно этому алгоритму сначала рассматривается множество всех натуральных чисел от 2 до $N$ в порядке возрастания, а затем из него последовательно удаляются элементы, не являющиеся простыми числами. Для этого на каждом шаге алгоритма выбирается наименьший элемент множества и вычеркиваются все кратные ему элементы. Сам этот элемент будет простым числом, и в дальнейшем не рассматривается.

Первый алгоритм позволяет проверить на «простоту» каждое число из отрезка [2; *N*]. По определению оно должно делиться только на 1 и себя. Подсчитаем сколько раз оно делится нацело и получим ответ.

```
For I:=1 to N do
begin
K:=0;
For J:=1 to I do If (I/J)  trunc(I/J) then K:=K+1;
If K=2 then Writeln(I); {в случае только двух делителей}
end;
```

Если необходимо исследовать остаток от деление целого числа на целое, то следует использовать операции целочисленного деления div и mod. При этом на Turbo Pascal по некоторым причинам результат компиляции Inc(K) работает быстрее, чем K:=K+1. Нет смысла делить i на все числа, большие, чем половина i.

Кроме того, подсчет количества делений сам по себе не нужен: любое удачное деление нацело на число, не равное 1 и самому себе, свидетельствует о том, что число не является простым.

```
For I:= 1 to N do
begin
  J:=2; K:=I div 2;
  While (I mod J  0) and (J K) Do inc(J);
  If J=K then Writeln (I, ' простое число');
end;
```

Дальнейшее улучшение алгоритма вычисления множества простых чисел возможно лишь за счет дополнительных затрат памяти. С другой стороны, любое подобного рода вычисление должно иметь своей целью построение таблицы простых чисел, так что излишество затрат памяти весьма сомнительно. Суть алгоритма заключается в следующем: нет необходимости производить деления на составные числа, так как каждое составное число есть произведение простых его составляющих.

**Программа**

```
uses crt;
const n=1000; {Максимальное простое число}
```

```
var a:array[1..n] of boolean;
    procedure init; {Заполняем массив значениями true}
    var i:word;
    begin
        for i:=1 to n do a[i]:=true;
    end;
  procedure delete(i:word);
    {Удаляем (присваиваем) значение false
    элементам массива с номером кратным i}
  var j:word;
    begin
    j:=2*i;
    while j n do begin a[j]:=false; j:=j+i end;
    end;
var i:word;
  begin
    clrscr; init;
{Удаляем все непростые элементы массива}
    for i:=2 to n do if a[i] then delete(i);
{выводим все простые числа}
for i:=1 to n do if a[i] then write (i:5);
end.
```

Продолжая тему простых чисел в применении к квадратным матрицам, рассмотрим задачу, придуманную Г. Э. Дьюдни, известным английским специалистом по головоломкам. Наверное, многим знакомы так называемые магические квадраты — квадратные матрицы чисел, у которых суммирование элементов по любой строке, любому столбцу и двум главным диагоналям дает одно и то же число. Существуют ли магические квадраты, состоящие только из простых чисел? Магический квадрат размером 3 × 3, приведенный на рис. П3.3, имеет сумму 111.

| | | |
|---|---|---|
| **67** | **1** | **43** |
| **13** | **37** | **61** |
| **31** | **73** | **7** |

**Рис. П3.3.** Магический квадрат Дьюдни

## *П3.4. Вычисление квадратного корня из целого числа*

Задача вычисления квадратного корня при построении программ достаточна тривиальная. Функция для ее решения (sqrt) присутствует практически в любом из современных языков программирования. Однако практика использования функции sqrt показала, что данная функция ведет себя совершенно различным способом для целочисленных и действительных аргументов.

**Программа 1**

```
#include <stdio.h>
#include <math.h>
void main( )
{
  int i = 169, j = 168;
  printf(
    <sqrt(%d)=%d, sqrt(%d)=%d>,
    i, (int)sqrt(i),
    j, (int)sqrt(j)
  );
}
```

Результат выполнения кода, приведенного в программе 1, выглядит так:

```
sqrt(169) = 13, sqrt(168) = 12.
```

В действительности, значение квадратного корня для числа 168 соответствует числу 12,96, что по общепринятым правилам округления ближе к целому числу 13. В данной программе мы видим классический машинный случай округления с отбрасыванием дробной части.

Рассматривая задачу вычисления квадратного корня с точки зрения уменьшения величины вычислительных затрат, более привлекателен целочисленный алгоритм, реализующий формулу Ньютона.

**Программа 2**

```
unsigned sqrt_cpu_newton(long l)
{
  unsigned rslt = (unsigned)l;
```

```
  long div = l;
  if (l <= 0)
    return 0;
  while (1)
  {
    div =  (l / div + div) / 2;
    if (rslt > div)
      rslt = (unsigned)div;
    else
      return rslt;
  }
}
```

Результат работы алгоритма из программы 2 опять тот же — округление до целого числа отбрасыванием дробной части.

В заключение следует отметить о существовании еще одной модификации алгоритма. На этот раз модификация преследует только задачу повышения производительности алгоритма. Повысить производительность итерационных алгоритмов возможно только одним способом — уменьшить количество итераций.

**Программа 3**

```
unsigned sqrt_newton(long l)
{
  long temp , div;
  unsigned  rslt = (unsigned)l;
  if (l <=  0)
    return 0;
  else if (l & 0xFFFF0000L)
    if  (l & 0xFF000000L)
      div  = 0x3FFF;
    else
      div = 0x3FF;
  else
    if  (l & 0x0FF00L)
      div  = 0x3F;
    else
      div  = (l > 4) ? 0x7 : l;
  while (1)
  {
    temp = l  / div + div;
    div =  temp >>  1;
```

```
    div += temp  & 1;
    if  (rslt > div)
      rslt = (unsigned)div;
    else
    {
      if (l / rslt == rslt - 1 && l % rslt == 0)
        rslt-;
      return rslt;
    }
  }
}
```

Последняя модификация алгоритма вычисляет квадратный корень из числа без ошибок округления на диапазоне [0...10 000] в среднем за три итерационных цикла.

# Глоссарий

**Абстрагирование** — метод решения задач, при котором объекты разного рода объединяются общим понятием (концепцией). Сгруппированные сущности рассматриваются как элементы единой категории.

**Абсцисса** — одна из декартовых координат точки, обычно первая, обозначаемая буквой $x$.

**Аддитивность** — свойство величин, состоящее в том, что значение величины, соответствующее целому объекту, равно сумме значений величин, соответствующих его частям при любом разбиении объекта на части.

**Аксиома** — основное положение, самоочевидный принцип. Впервые термин встречается у Аристотеля. Использовался в книгах Евклида «Начала». Большую роль сыграли работы древнегреческого ученого Архимеда, который сформулировал аксиомы, относящиеся к измерению величин.

**Аксонометрия** — один из способов изображения пространственных фигур на плоскости.

**Алгебра** — часть математики, развивающаяся в связи с задачей о решении алгебраических уравнений.

**Алгебра логики (булева алгебра)** — математический аппарат, с помощью которого записывают (кодируют), упрощают, вычисляют и преобразовывают логические высказывания.

**Алгоритм** — понятное и точное предписание (указание) исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов.

**Аналогия** — умозаключение по сходству частных свойств, имеющихся у двух математических понятий.

**Аппликата** — одна из декартовых координат точки в пространстве, обычно трётья, обозначаемая буквой $z$.

**Аппроксимация** — замена одних математических объектов другими, в том или ином смысле близкими к исходным.

**Аргумент функции** — независимая переменная величина, по значениям которой определяют значения функции.

**Арифметика** — наука, изучающая действия над числами. Арифметика возникла в странах Др. Востока, Вавилоне, Китае, Индии, Египте. Особый вклад внесли: Анаксагор и Зенон, Евклид, Эратосфен, Диофант, Пифагор, Л. Пизанский и др.

**Асимметрия** — отсутствие или нарушение симметрии.

**Асимптота** — прямая, к которой неограниченно приближаются точки некоторой кривой по мере того, как эти точки удаляются в бесконечность.

**Астроида** — алгебраическая кривая.

**Биллион** — тысяча миллионов, число изображаемое единицей с девятью нулями. В некоторых странах биллионом называют число, равное $10^{12}$.

**Бином** — сумма или разность двух чисел или алгебраических выражений, называемых членами бинома.

**Вектор** — направленный отрезок прямой, у которой один конец называют началом вектора, другой конец — концом вектора.

**Геометрия** — часть математики, изучающая пространственные отношения и формы.

**Гипербола** — незамкнутая кривая из двух неограниченно простирающихся ветвей.

**Гипотенуза** — сторона прямоугольного треугольника, лежащая против прямого угла.

**Гомотетия** — расположение подобных между собой фигур, при котором прямые, соединяющие соответствующие друг другу точки фигур, пересекаются в одной и той же точке, называемой центром гомотетии.

**Градус** — единица измерения плоского угла, равная 1/90 части прямого угла. Измерение углов в градусах появилось более 3 веков назад в Вавилоне.

**График** — кривая на плоскости, изображаемая зависимость функции от аргумента.

**Данные** — информация, представленная в определенной форме, закодированная для того, чтобы упростить ее обработку, запись или передачу.

**Дедукция** — форма мышления, посредством которой утверждение выводится чисто логически (по правилам логики) из некоторых данных утверждений (посылок).

**Деференты** — окружность, по которой вращаются эпициклоиды каждой планеты. У Птолемея планеты вращаются по окружностям — эпициклам, а центры эпициклов каждой планеты — вокруг Земли по большим окружностям — деферентам.

**Дидактические подходы** — принципы и способы обучения, преподавания.

**Дистанционное обучение** — способ организации учебного процесса с использованием образовательной среды, основанной на современных информационных и телекоммуникационных технологиях, позволяющих осуществлять обучение на расстоянии без непосредственного контакта между преподавателем и учащимся.

**Диагональ** — отрезок прямой, соединяющий две вершины многоугольника, не лежащие на одной стороне.

**Дискретность** — прерывность; противопоставляется непрерывности.

**Дискриминант** — выражение, составленное из величин, определяющих функцию, обращением которого в нуль характеризует то или иное отклонение функции от нормы.

**Дихотомия** — разделение надвое (способ классификации).

**Доказательство** — цепь правильных умозаключений, ведущих от истинных посылок к доказываемым тезисам.

**Иерархия** — структура, упорядоченная по подчиненности в соответствии с некоторым набором правил.

**Инвариантность** — неизменность какой-либо величины по отношению к преобразованиям координат.

**Индукция** — один из методов доказательства математических утверждений.

**Индекс** — числовой или буквенный указатель, которым снабжаются математические выражения для того, чтобы отличать их друг от друга.

**Интеграл** — одно из основных понятий математического анализа, возникшее в связи с потребностью измерять площади, объемы, отыскивать функции по их производным. Знак $\int$ — стилизованная буква $S$ от лат. слова *summa* — «сумма».

**Интервал** — множество действительных чисел, удовлетворяющее неравенству $a < x < b$.

**Интерактивный** — качество оборудования, программ или условий эксплуатации, которое позволяет действовать в форме, приближающейся к диалогу с пользователями или в реальном времени с компьютерами.

**Интерфейс** — сопряжение двух устройств, обменивающихся информацией. Место соединения оборудования, программного обеспечения (компьютера), с одной стороны, и человека — с другой.

**Информация** — сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления, передаваемые и хранимые с помощью условных сигналов (знаков).

**Итерация** — повторение. Результат повторного применения какой-либо математической операции.

**Квинтиллион** — число, изображаемое единицей с 18 нулями.

**Коллинеарность** — расположенность на одной линии (прямой).

**Комбинаторика** — раздел математики, в котором изучаются различные соединения и размещения, связанные с подсчетом комбинаций из элементов данного конечного множества.

**Компланарность** — расположение в одной плоскости.

**Конгруэнтность** — соразмерный. Термин, употребляемый для обозначения равенства отрезков, углов, треугольников и др.

**Константа** — постоянная величина при рассмотрении математических и др. процессов.

**Контент** — содержание курса. Все учебные материалы, пособия, документы, задания, тесты и контрольные мероприятия курса.

**Конфигурация** — расположение фигур.

**Корреляция** — связь переменных, при которой одному значению одного признака соответствует несколько значений другого признака, отклоняющегося в ту или иную сторону от своего среднего значения.

**Коэффициент** — множитель, обычно выражаемый цифрами.

**Лемма** — вспомогательное предложение, употребляемое при доказательствах других утверждений. Термин введен древнегреческими геометрами, особенно часто встречается у Архимеда.

**Логарифм** — показатель степени $m$, в которую необходимо возвести $a$, чтобы получить $n$.

**Максимум** — наибольшее значение функции на множестве определения функции.

**Мантисса** — дробная часть десятичного логарифма. Термин был предложен российским математиком Л. Эйлером (1748).

**Матрица** — прямоугольная таблица, образованная из некоторого множества и состоящая из строк и столбцов.

**Метрика** — правило определения расстояния между любыми двумя точками данного пространства.

**Метод Монте-Карло** — численный метод, основанный на получении большого числа реализаций случайного процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи.

**Минимум** — наименьшее значение функции на множестве определения функции.

**Минус** — математический знак в виде горизонтальной черты, употребляемый для обозначения отрицательных чисел и действия вычитания.

**Модуль** — абсолютная величина действительного числа.

**Мультипликативность** — умножение (свойство функции Эйлера).

**Он-лайн обучение** — способ организации процесса самостоятельного изучения учебных материалов и получения сертификатов с использованием образовательной среды, основанной на Интернет-технологиях.

**Ордината** — одна из декартовых координат точки, обычно вторая, обозначаемая буквой $y$.

**Орт** — единичный вектор, длина которого принята равной единице.

**Ортогональность** — обобщение понятия перпендикулярности.

**Парадигма** — базовая модель конкретного способа организации информации. Объектно-ориентированная парадигма делает упор на поведении и обязанностях.

**Параметр** — вспомогательная переменная, входящая в формулы и выражения.

**Планиметрия** — часть элементарной геометрии, в которой изучаются свойства фигур, лежащих в плоскости.

**Полином** — многочлен, т. е. сумма некоторого числа одночленов.

**Предел** — одно из основных понятий математики, означающее, что некоторая переменная величина в рассматриваемом процессе ее изменения неограниченно приближается к определенному постоянному значению.

**Пропорция** — равенство между двумя отношениями четырех величин.

**Пост-тест** — тест на оценку, проверяет знания студентов по пройденным темам.

**Пре ПК** (презентация с помощью компьютера) — использование мультимедийных носителей и техники для достижения лучшего понимания доклада, привлечения внимания собравшихся к предлагаемой информации.

**Пре-тест** — тест, за который не ставится оценка. Он определяет, насколько студент знаком с новой темой, какие вопросы предыдущей требуют пояснения преподавателя или дополнительной практики.

**Программное обеспечение** — совокупность программ, выполняемых компьютером, а также вся область деятельности по проектированию и разработке программ.

**Радикал** — знак $\sqrt{}$ состоит из двух частей: модифицированной буквы *r* и черты, заменявшей ранее скобки.

**Рекуррентный** — возвращаться назад; возвратное движение в математике.

**Сайт** — «место» в Интернете. Речь идет об информации в сервере, доступной для подключенных к глобальной сети пользователей. Сайт иногда заменяют словом «страница», хотя эти понятия не всегда совпадают.

**Семантика** — система правил истолкования отдельных языковых конструкций. Устанавливает, какие последовательности действий описываются теми или иными фразами языка и какой алгоритм определен данным текстом на алгоритмическом языке.

**Сертификат** — информационное сообщение (запись), подтверждающее успешное завершение изучения курса.

**Сигнатура аргументов** — внутреннее представление списка типов данных аргументов функции. Используется для ликвидации двусмысленности при вызове перегруженных функций. Из числа претендентов выбирается та функция, которая наиболее близко соответствует сигнатуре вызываемой функции.

**Секстиллион** — число, изображаемое с 21 нулем, т. е. число $10^{21}$.

**Сектор** — часть круга, ограниченная дугой его граничной окружности и двумя ее радиусами, соединяющими концы дуги с центром круга.

**Симметрия** — свойство формы или расположения фигур симметрично.

**Скаляр** — величина, каждое значение которой выражается одним числом.

**Стереометрия** — часть элементарной геометрии, в которой изучаются пространственные фигуры.

**Теорема** — математическое утверждение, истинность которого установлена путем доказательства.

**Топология** — ветвь геометрии, изучающая свойства геометрических фигур, связанных с их взаимным расположением.

**Транспозиция** — перестановка элементов данной совокупности, при которой меняются местами два элемента.

**Форум** — инструмент для общения на сайте. Сообщения в форуме чем-то похожи на почтовые — каждое из них имеет автора, тему и собственно содержание.

**Уникурсальный** — маршрут обхода всех ребер построенного графа, при котором ни одно ребро не проходит дважды.

**Формула** — комбинация математических знаков, выражающая какое-либо предложение.

**Функция** — одно из основных понятий математики, выражающее зависимость одних переменных величин от других.

**Число** $\pi$ — отношение длины окружности к ее диаметру ($\pi = 3{,}14159265358979323...$).

**Шкала** — последовательность чисел, служащая для количественной оценки каких-либо величин.

**Экстраполирование** — продолжение функции, принадлежащей заданному классу, за пределы ее области определения.

**Экстремум** — общее название максимума и минимума функции.

**Эссе** — письменная работа студента (группы), выражающая (кратко и в свободной форме) индивидуальные впечатления и соображения по конкретному поводу или вопросу и заведомо не претендующая на определяющую или исчерпывающую трактовку предмета.

**Язык** — система символов и правил, предназначенная определять задачи, которые компьютер должен решать.

# Оглавление

**Колдаев Виктор Дмитриевич**

# Численные методы и программирование

Учебное пособие

Редактор *А. В. Волковицкая*
Корректор *А. В. Алешина*
Компьютерная верстка *И. В. Кондратьевой*
Оформление серии *К. В. Пономарева*

Подписано в печать 25.06.2007. Формат 60 × 90/16.
Печать офсетная. Гарнитура «Таймс». Усл. печ. л. 21,0. Уч.-изд. л. 21,5.
Бумага офсетная. Доп. тираж 2000 экз. Заказ № 1335.

ЛР № 071629 от 20.04.98
Издательский Дом «ФОРУМ»
101000, Москва — Центр, Колпачный пер., д. 9а
Тел./факс: (495) 625-39-27
E-mail: forum-books@mail.ru

ЛР № 070824 от 21.01.93
Издательский Дом «ИНФРА-М»
127282, Москва, Полярная ул., д. 31в
Тел.: (495) 380-05-40
Факс: (495) 363-92-12
E-mail: books@infra-m.ru
Http://www.infra-m.ru

***По вопросам приобретения книг обращайтесь:***

*Отдел продаж «ИНФРА-М»*
127282, Москва, ул. Полярная, д. 31в
Тел.: (495) 363-42-60
Факс: (495) 363-92-12
E-mail: books@infra-m.ru

*Центр комплектования библиотек*
119019, Москва, ул. Моховая, д. 16
(Российская государственная библиотека, кор. К)
Тел.: (495) 202-93-15

*Магазин «Библиосфера» (розничная продажа)*
109147, Москва, ул. Марксистская, д. 9
Тел.: (495) 670-52-18, (495) 670-52-19

Отпечатано с готовых диапозитивов в ОАО ордена «Знак Почета»
«Смоленская областная типография им. В. И. Смирнова».
214000, г. Смоленск, проспект им. Ю. Гагарина, 2.

## *КОЛДАЕВ ВИКТОР ДМИТРИЕВИЧ*

Доцент кафедры «Информатика и программное обеспечение вычислительных систем» Московского Государственного института электронной техники – технического университета. Имеет свыше 50 научных работ, в том числе методические и учебные пособия по дисциплинам «Вычислительная техника и программирование», «ЭВМ и программное обеспечение», «Информатика», «Компьютерная практика», «Теория алгоритмов и вычислительные методы», «Алгоритмы и структуры данных». Колдаев В.Д. занял первое место в конкурсах «Лучший преподаватель МИЭТ», лучший учитель г. Зеленограда, финалист конкурса «Учитель года Москвы 2003», лауреат конкурсов «Грант Москвы» 2003 и 2005 гг., «Грант МИЭТ 2006». Ему присвоено почетное звание «Заслуженный работник МИЭТ».

ISBN 978-5-8199-0333-9

9 785819 903339