ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Физики»

Учебно-методическое пособие

по междисциплинарной дисциплине

«Применение пакета компьютерной математики *SMATH STUDIO* при изучении дисциплин естественнонаучного и общетехнического циклов»

Автор

Бажин И.В.

Ростов-на-Дону, 2023

Аннотация

Представлены основные методы аналитического и численного решения задач, наиболее часто встречающихся при изучении «Математики», «Физики», «Электротехники и электроники» и других дисциплин с использованием отечественной системы компьютерной математики SMath Studio. Приведен краткий теоретический материал, необходимый для начала работы с данной программой. Рассмотренные в пособии примеры, направлены на формирование у студентов необходимой теоретической и практической базы для решения сложных прикладных задач.

Для бакалавров, магистрантов, аспирантов инженерных специальностей по применению пакета SMath Studio в образовании и профессиональной деятельности.

Автор

к.ф.-м.н., доцент каф. «Физика»
Бажин И.В.

Оглавление

Предисловие **..... 4**
1. **Пакет компьютерной математики SMath Studio и его возможности** 5
1.1. Методические материалы 19
2. **Математика и SMath Studio .....** 21
2.1. Векторы. Матрицы 21
2.2. Решение СЛАУ 34
2.3. Функции. Графики 45
2.4. Приведение кривых второго порядка к каноническому виду 56
2.5. Дифференцирование функции. Примеры использования 56
2.6. Определенный интеграл. Численное интегрирование 58
2.7. Методические материалы 74
3. **Физика, электротехника, электроника и SMath Studio .....** 80
3.1. Обработка результатов измерений 80
3.2. Движение тела брошенного под углом к горизонту 86
3.3. Колебания и волны. Сложение колебаний 95
3.4. Правила Кирхгофа 99
3.5. Фазо-импульсное управление 108
Перечень использованных информационных источников ... 112

ПРЕДИСЛОВИЕ

Данное учебное пособие следует рассматривать как введение в математические и численные методы решения прикладных инженерных задачи с использованием специализированного пакета компьютерной математики SMath Studio.

Пособие состоит из трех глав необходимых как для ознакомления с пакетом SMath Studio, так и изучения основных математических методов решения задач, возникающих при подготовке бакалавров, специалистов, магистрантов и аспирантов в техническом вузе в соответствии с требованиями федерального государственного образовательного стандарта высшего образования.

В первой главе представлены краткие сведения, необходимые для начала работы с пакетом компьютерной математики SMath Studio.

Во второй главе рассмотрены примеры по применению пакета SMath Studio для решения наиболее общих математических задач и углубленному изучению понятий о векторах, матрицах, функциях и операциях над ними.

В третьей главе рассмотрено применение пакета SMath Studio для решения задач по физике, электротехнике и электронике.

Методическое пособие основано на материалах наработанных автором при многолетнем чтении курсов «Специальные главы физики», «Методы математической физики», «Физические основы получения информации» и «Математические методы в электронике и светотехнике». Читателю предлагается разобрать ряд практических задач с использованием пакета SMath Studio, для более эффективного закрепления теоретического материала.

1. ПАКЕТ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ SMATH STUDIO И ЕГО ВОЗМОЖНОСТИ

Проведение компьютерного моделирования сложных (технических) систем требует знаний различных математических методов и проведение большого объема численных расчетов. Существенно упростить и повысить эффективность процесса моделирования позволяют системы компьютерной математики, самыми известными коммерческими пакетами являются *MATLAB*, *Maple*, *Mathematica*, *MathCAD* и некоторые другие.

Настоящее пособие посвящено изучению возможностей бесплатного отечественного программного продукта пакета *SMath Studio*[1] и его применения для изучения основных методов высшей и вычислительной математики для решения практически важных задач математического моделирования.

SMath Studio – система для аналитической и численной обработка выражений, включая дифференцирование, интегрирование, решение различных уравнений и систем уравнений, работу со списками, векторами, матрицами [1]. *SMath Studio* позволяет проводить численные расчеты, используя дроби, целые числа и числа с плавающей точкой произвольной точности. Результаты расчетов можно представить в виде двух и трехмерных графиков или анимационных клипов в интерактивном режиме. *SMath Studio* среди свободно распространяемых программ компьютерной математики обладает самыми широкими возможностями проведения численных вычислений, аналитических преобразований и графического представления результатов. Еще одним достоинством пакета *SMath Studio* является его открытость, в отличие от коммерческих пакетов, пользователи могут расширять возможности программы путем установки различных дополнений.

Из четырех вышеперечисленных систем компьютерной математики *Smath Studio* по внешнему виду рабочего окна и способу ввода математических выражений очень похожа на *MathCAD*. По меткому замечанию профессора НИУ «МЭИ» В.Ф. Очкова, автора многочисленных книг и статей по использованию пакета *MathCAD* в образовании и научно-технической деятельности, одного из первых российских специалистов, который начал активно применять *MathCAD* в своей профессиональной

[1]https://ru.smath.com, в этом руководстве рассматривается версия SMath Studio 1.00.8348 от 9.11.2022г.

деятельности, что *Smath Studio* – это «русский» *MathCAD* [2]. Это объясняется тем, что в данных программных продуктах реализован принцип ***WYSIWYG*** (*What You See Is What You Get* — «что видишь, то и получаешь»). Формульный (графический) редактор *Smath Studio* позволяет быстро, эффективно и интуитивно понятно вводить и редактировать математические формулы. Наличие простого в использовании графического интерфейса делает пакет *Smath Studio* доступным не только для студентов, но и для школьников. Близость встроенных языков *Smath Studio* и *MathCad* позволяет использовать наработки из многочисленных публикаций по *MathCad* и в среде *Smath Studio* [3 – 7].

Графическая среда *Smath Studio* предоставляет пользователю удобный и понятный (как на тетрадном листе) интерфейс на русском языке и русскоязычную справочную информацию, что облегчает изучение и использование пакета в образовательной и профессиональной деятельности.

Программу разрабатывает петербургский программист Андрей Ивашов. В развитии пакета принимают активное участие пользователи, ведущие обсуждение вопросов по использованию пакета на форуме проекта. К сожалению, русскоязычный форум в настоящее время приостановлен из-за того, что пользователи программы предпочитают общаться на английском языке. Но это, в свою очередь, говорит о международном признании данного программного проекта. Интерфейс и справочные материалы пакета *SMath Studio* доступны на 40 языках. Имеются монографии по использованию *SMath Studio* в различных областях науки и техники [8-9].

Пакет *SMath Studio* является мультиплатформенным, поддерживаются: Windows (стационарная и портативная версии), Linux, iOS, Android и др.

Для работы на устройствах с другими операционными системами или на компьютерах, где пакет не установлен, но есть браузер и доступ к интернету, есть возможность пользоваться сетевой версией пакета по адресу https://ru.smath.com/cloud/. Веб-сервис позволяет загружать и сохранять документы, получать ссылку на документ для публикации.

Среди других важных возможностей пакета *SMath Studio* является полная поддержка размерностей/единиц измерения, возможность создания анимированных графиков, встроенный отладчик позволяющий вы-

полнять пошаговый анализ сложных вычислений. И это далеко не полный перечень всех возможностей данной программы.

Существенно расширяют функционал базовой версии программы, многочисленные дополнения, установив которые, можно превратить *SMath Studio* в пакет компьютерной математики не сильно уступающим по возможностям лучшим коммерческим пакетам. Можно сказать, что *SMath Studio* может стать незаменимым помощником при решении любых профессиональных задач.

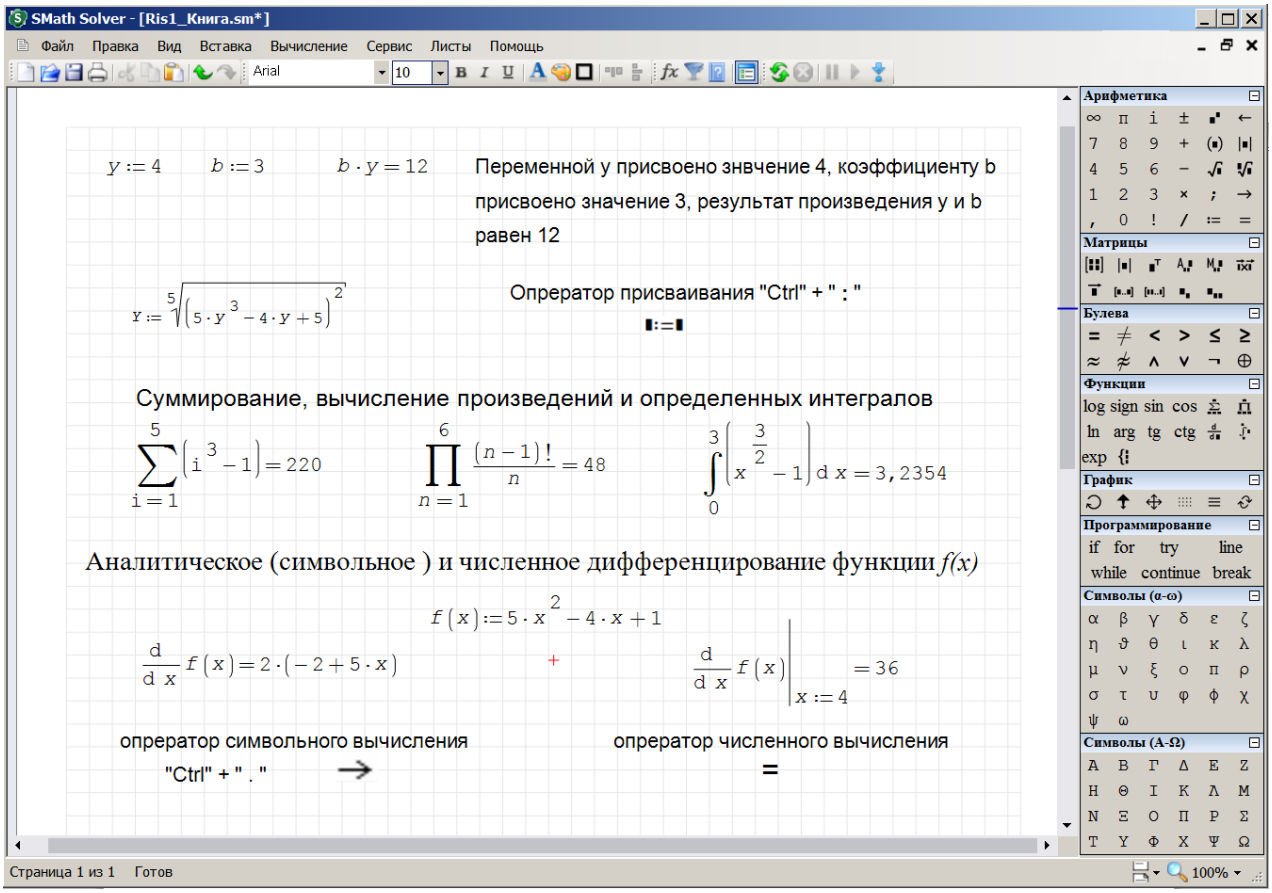


Рис. 1.1. Рабочее окно программы *SMath Studio*

Из рисунка 1.1 видно, что окно программы содержит заголовок, главное меню, панель инструментов, правая боковая панель и рабочее поле.

Рабочее поле программы *SMath Studio* выглядит как тетрадный лист в клеточку (можно легко отключить), на который с помощью «мыши», в место отмеченное красным крестиком, можно помещать математические символы и операторы из правой боковой панели или с клавиатуры с использованием «горячих клавиш» (что гораздо быстрее). Список горячих клавиш приведен в Приложении 1. Например, для ввода оператора присваивания «:=» можно воспользоваться разделом «Арифметика» с боко-

вой панели и выбрать оператор «:=«, или необходимо одновременно нажать клавиши «*Ctrl*» и «:» на клавиатуре. После того, как математическое выражение введено, можно нажать символ «=» и программа выдаст результат расчета. Для придания документу законченного и удобного для использования вида формулы и выражения могут сопровождаться комментариями.

Программа *SMath Studio* позволяет рассчитывать суммы, произведения, определенные интегралы и проводить численное и аналитическое вычисление производных («Функции» на правой боковой панели).

Следует отметить, что количество вынесенных на правую панель функций крайне ограничено, полный набор встроенных функций *SMath Studio*, содержится в подразделе «Функция» из раздела «Вставка» главного меню, которое выводится на экран сочетанием клавиш «*Ctrl*» и «E».

SMath Studio имеет все необходимые возможности работы с текстом, такие как работа с колонтитулами, поддержка фоновых изображений, форматирование текста, настраиваемый формат бумаги, автоматические поля, метаданные документа и многие другие функции создания и редактирования документов.

Графический редактор *SMath Studio* позволяет легко создавать и редактировать любые математические выражения. Кратко опишем нетривиальные возможности графического редактора.

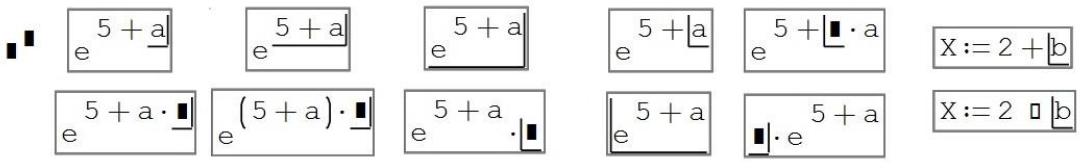


Рис. 1.2. Этапы редактирования математического выражения

На рис. 1.2 приведены основные операции необходимые при вводе и редактировании математических выражений. Рассмотрим пример с созданием выражения содержащего экспоненциальный множитель. После ввода оператора возведение в степень, воспользовавшись или правой панелью «Арифметика» с выбором оператора «■■« или набрав на клавиатуре «^», вводим математическое выражение. После ввода выражения его можно редактировать. Для этого необходимо подвести и щелкнуть левой клавишей мыши справа или слева символа, относительно которого будет осуществляться редактирование. На экране это отображается правым или левым «уголком». При последовательном нажатии клавиши «Пробел»,

можно отмечать различные части выражения для их редактирования. Показаны основные этапы редактирования как отдельного символа, части выражения или выражения целиком. В правой части рис. 1.2 приведен пример редактирования оператора сложения в набранном выражении. При наведении и клика указателем мыши на левую грань символа «***b***» и нажимая клавишу «Backspace» получаем возможность редактировать оператор.

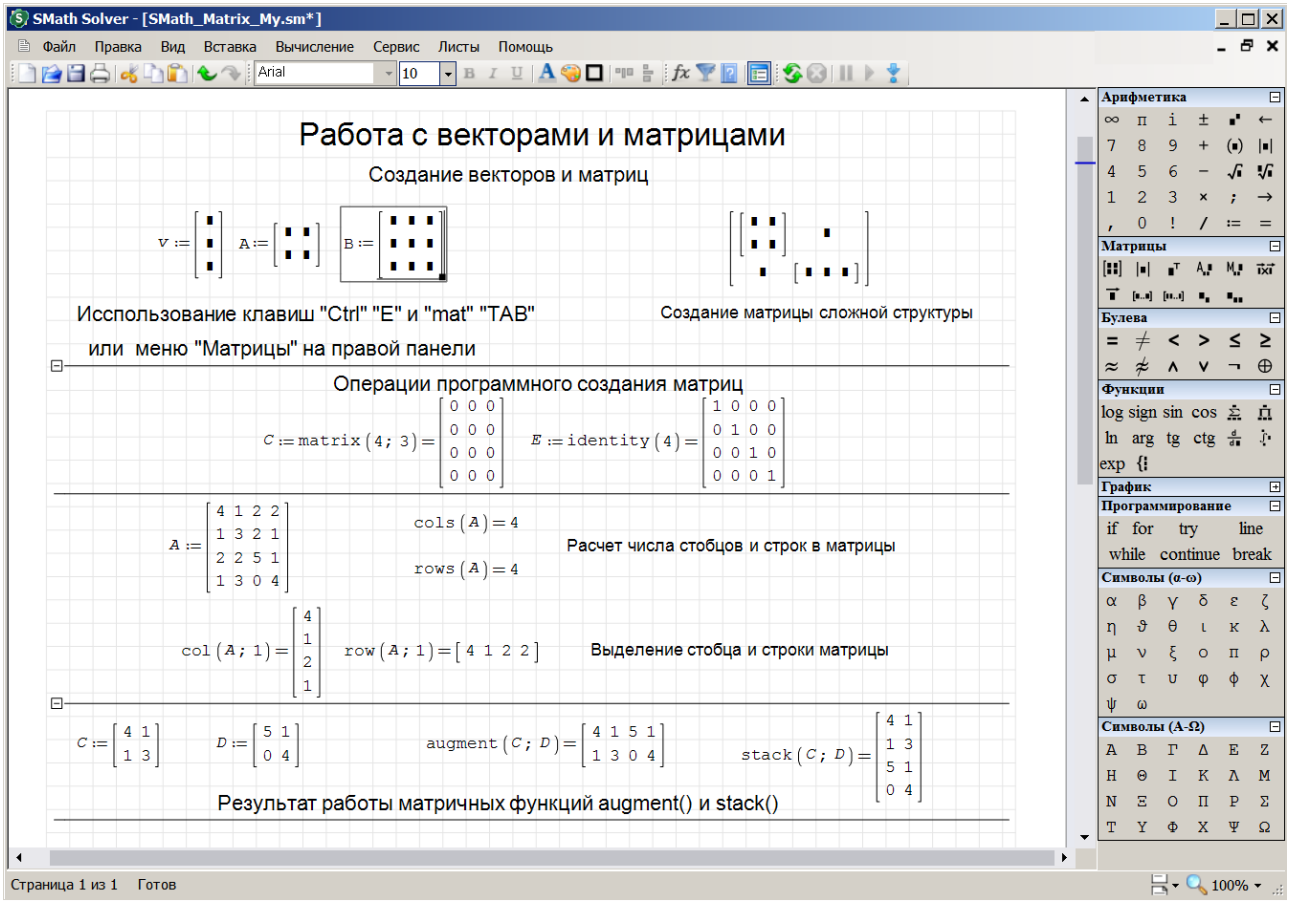


Рис. 1.3. Работа с векторами и матрицами в пакете *SMath Studio*

Для работы с большими данными *SMath Studio* предлагает полный набор матричных и векторных операций. Некоторые из них приведены на рис. 1.3. Для создания матриц и вызова основных матричных функций можно использовать правую панель «Матрицы», подраздел «Функции» из меню «Вставка» или использовать горячие клавиши (mat «TAB» или «Ctrl» «M»).

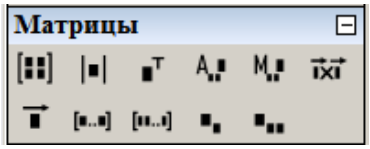
Для структурирования информации на Рабочем поле удобно пользоваться инструментами «Область» и «Разделитель» из меню «Вставка» (пример создания областей в документе рис. 1.3).

Для изменения размерности матрицы необходимо выделить «угол-

ком» всю матрицу, в правом нижнем углу появится чёрный квадратик. Матрица «B» в верхней части рис. 3. Растягивая его мышью, можно установить требуемый такой размер вектора или матрицы.

При программном создании матрицы можно использовать функции «***matrix****(N; M)*» для создания матрицы состоящей из *N* строк и *M* столбцов заполненных нулевыми элементами и «***identity****(N)*» для формирования единичной матрицы размерностью *N*x*N*.

Часто встречающиеся матричные операции вынесены на правую панель.



Это функции создания матрицы («Ctrl»«M»), вычисления определителя квадратной матрицы ***det****(A)*, транспонирование матрицы A^T, выделение алгебраического дополнения и минора матрицы, операция векторного произведения $\vec{i}\times\vec{i}$, операция векторизации, создание области изменения индекса с единичным и произвольным шагом, а так же создание и работа с индексами векторов $\blacksquare_{\blacksquare}$ и матриц $\blacksquare_{\blacksquare,\blacksquare}$.

Очень часто приходится создавать индексы для векторов и матриц. Вместо использования правой панели, намного быстрее и удобнее использовать горячую клавишу «**[**«, для создания одинарного индекса, если нажать клавишу «**;**», то можно получить второй индекс $\blacksquare_{\blacksquare,\blacksquare}$.

Теперь можно использовать элемент вектора (матрицы), для этого необходимо указать имя вектора или матрицы и индекс(ы) требуемого элемента. Необходимо запомнить, что нумерация всегда начинается с «1», а разделителем индексов служит «;». Другой способ обращения к индексированной переменной заключается в использовании матричной функций ***el***(имя; строка) для векторов или ***el***(имя; строка; столбец) для матриц. Получить строку или столбец можно с помощью функций: ***col****(A; m)* - *m*-тая строка матрицы *A*, ***row****(A; n)* - *n*-ный столбец матрицы *A*. Для расчета количества строк и столбцов матрицы используются функции ***rows****(A)* и ***cols****(A)*.

Для объединения двух матриц используются функции ***augment****()* и ***stack****()*. Функция ***augment****(A; B...; Z)* создает объединенную матрицу, в которой элементы A...Z следуют друг за другом слева-направо (число строк матриц A...Z должно быть одинаковым). Функция ***stack****(A; B...; Z)* создает матрицу путём размещения элементов A...Z друг под другом (число столбцов должно быть одинаковым).

Функция ***submatrux***$(A; m_1; m_2; n_1; n_2)$ формирует из матрицы A подматрицу путём выделения строк с m_1 по m_2 и столбцов с n_1 по n_2.

Среди других матричный функций следует отметить – ***invert***$(A) \equiv |A|^{-1}$ – расчет обратной матрицы; ***max***(A), ***min***(A) – вычисление максимального и минимального элемента; ***sort***(A), ***csort***$(A; n)$, ***rsort***$(A; n)$ – сортировка элементов вектора и матриц в порядке возрастания; ***rank***(A) – вычисление ранга матрицы A; ***norme***(A), ***norm1***(A), ***normi***(A) – вычисление норм квадратной матрицы A.

Данные матричные функции более подробно будут рассмотрены во второй части учебного пособия.

Пакет *SMath Studio* предоставляет богатые возможности по решению различных типов уравнений и систем уравнений.

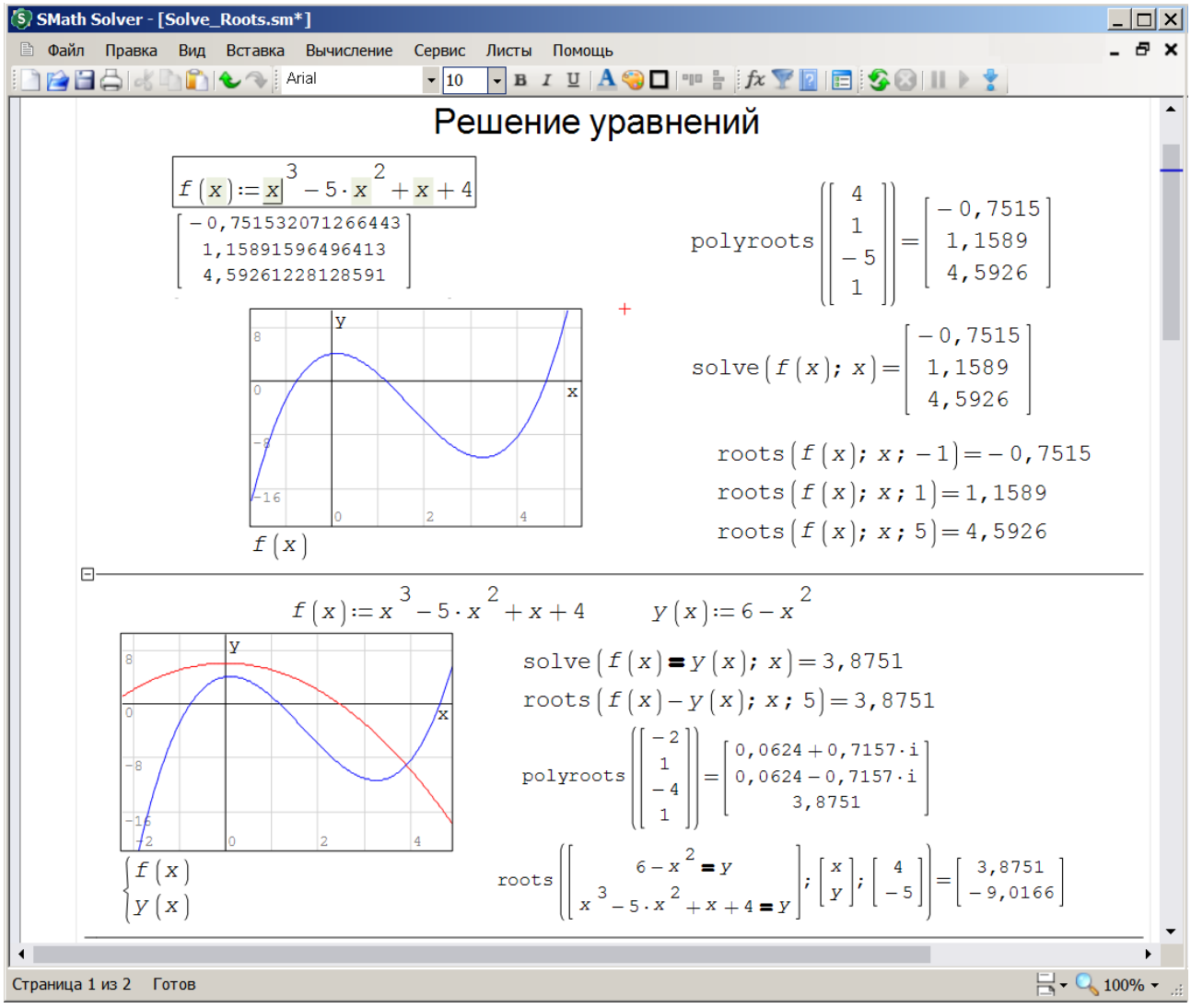


Рис. 1.4. Решение нелинейных уравнений с помощью различных встроенных функций

На рис. 1.4 приведены основные способы решения уравнений, представленные в базовой версии программы *SMath Studio*. В верхней части рисунка показано решение нелинейного уравнения, в качестве которого взят полином третьей степени. Показано пять способов решения уравнений с помощью *SMath Studio*.

Первый способ решения состоит в том, чтобы найти корни полинома, т.е. точки пересечения функции *f(x)* с осью абсцисс. Для этого выделим мышью, переменную x, относительно которой решается уравнении, заходим в пункт «Вычисление» главного меню, и выбираем пункт «найти корни». Под выражением, для которого ищутся корни, появляется матрица с решениями. Как видно из рисунка, результат встроенной в *SMath Studio* процедуры решения уравнений выдается с максимальной точностью, которая по умолчанию составляет 15 значащих цифр.

Для проверки правильности полученных решений построен график функции *f(x)*, по которому, используя операцию масштабирования можно приближенно получить те же самые корни. Это второй – графический способ решения уравнений.

Третий способ решения с помощью функции ***polyroots()***. В качестве аргументов этой функции используются коэффициенты полинома стоящие перед аргументами в соответствующей степени. Коэффициенты задаются вектором, верхний элемент которого – это свободный коэффициент, потом коэффициент перед первой степенью x, а последний, перед членом, содержащим аргумент в максимальной степени, в нашем случае это x^3. После вставки «**=**« получаем корни полинома.

Следующий способ, решение уравнения с помощью встроенной функции ***solve***().

Необходимо отметить, что в пакете *SMath Studio* встроенные функции различаются не только названием (которое, к тому же зависит от регистра), но и числа аргументов, которое может иметь выбранная функция. Так функция ***solve***() может иметь два или четыре аргумента. В примере на рис. 4, используется функция ***solve***(*f(x)*; *x*) от двух аргументов, первый из которых эта функция задающая уравнение, а второй параметр задает аргумент, по которому ищется корень уравнения. Результатом работы функции ***solve***(*f(x)*; *x*) является вектор, содержащий решения уравнения. Как видно из рисунка, выбранный полином имеет три корня.

Функцию ***solve***(*f(x)*; *x*) от двух аргументов удается использовать да-

леко не всегда, часто попытка ее использования завершается ошибкой, а само выражение, помещается в красную рамку, с надписью – «действительных корней нет». В этом случае можно попытаться использовать функцию ***solve***(*f(x)*; *x*; *a*; *b*) с четырьмя аргументами, где аргументы *a* и *b* задают область по оси *X*, в которой необходимо искать возможное решение. Недостатком функции ***solve()***, является то, что данная функция ищет только действительные корни уравнения.

Следующим, пятым способом решения уравнений в *SMath Studio* является встроенная функция ***roots()***. Данная функция также может иметь разное количество аргументов – два или три.

Воспользуемся функцией ***roots***(*f(x)*; *x*; *c*), где первый параметр это функция, второй – аргумент, третий – начальное приближение. Т.к. полином третьей степени имеет три корня применение функции ***roots***(*f(x)*; *x*) с двумя аргументами выдает только один первый корень. Приходится применять функцию ***roots***(*f(x)*; *x*; *c*) три раза, с тремя приближенными значениями корней. В этом заключается главный недостаток функции ***roots()***.

Теперь немного усложним нашу задачу, будем искать точку пересечения ранее рассмотренного полинома с параболой (нижняя часть рис. 4). На графике видно, что имеется один действительный корень. Из рисунка видно, что функции ***solve()*** и ***roots()*** находят искомый корень. Можно заметить, что разность двух полиномов, также является полиномом $f(x) - y(x) = x^3 - 4x^2 + x - 2$. Результирующий полином третьей степени, следовательно, у него три корня, два комплексных и один действительный. Соответственно, задача пересечения двух полиномов сводится к поиску корней одного полинома, которые выдает функция ***polyroots()***. Достоинством встроенной функции ***polyroots()*** является возможность поиска комплексных корней.

Рассмотрим нашу задачу по-другому, найдем абсциссу и ординату точки пересечения двух функций. Из рисунка 1.4 видно, что задача сводится к системе двух нелинейных уравнений. Для решения используется встроенная функция ***roots()***, где в качестве аргументов используются функции, аргументы и начальные приближения, задаваемые в виде векторов.

Завершая рассказ о решении уравнений, рассмотрим возможность решения систем линейных алгебраических уравнений (СЛАУ) «штатными»

средствами из пакета *SMath Studio*. На рис. 1.5 приведена СЛАУ 3х3, которую можно решить методом обратной матрицы и с использованием встроенной функции ***roots()***. Из рисунка видно, что систему уравнений можно задать или в «естественном» виде или в виде матрицы коэффициентов перед неизвестными величинами и вектора правой части СЛАУ. Метод обратной матрицы выдает самый компактный результат решения. Проведем проверку, подставляя решение вместо неизвестных величин, в результате получаем правую часть системы.

Туже систему подставляем в функцию ***roots()*** и получаем те же самые результаты. Т.е. система уравнений решена корректно. Как и ранее, аргументами функции ***roots()*** являются вектор, состоящий из уравнений системы и вектор неизвестных величин (*x*; *y*; *z*).

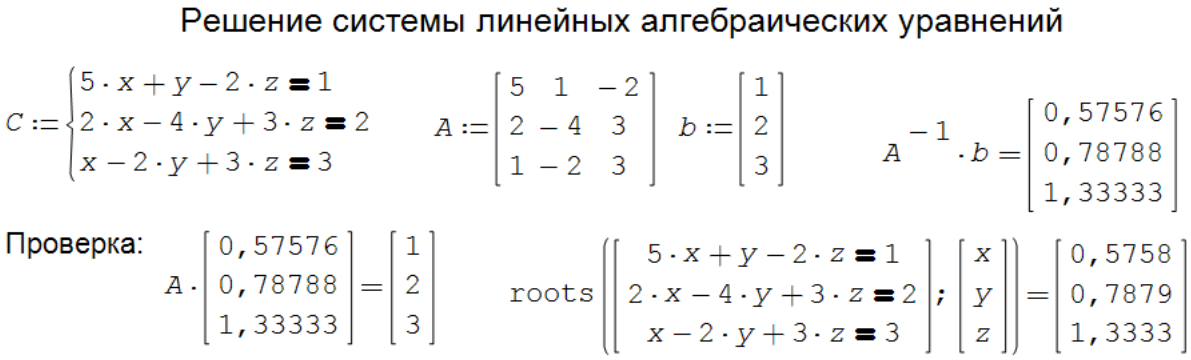


Рис. 1.5. Решение системы линейных алгебраических уравнений

Как и в других системах компьютерной математики, *SMath Studio* имеет возможность для графического представления результатов расчета. Ранее было показано, что построение графиков изучаемых функций существенно упрощает решения многих задач, в частности, нахождение числа коней уравнений и их значения.

Для построения графиков в *SMath Studio* имеются встроенные функции построения 2D и 3D графиков. Для вызова данных функций следует в пункте меню «Вставка» выбрать подменю график и выбрать «Двумерный 2D» или «Трехмерный 3D» график.

Рассмотрим работу с функцией «Двумерный 2D» график. После нажатия «горячих» клавишей «Shift @» на экране появляется пустое поле нового графика (рис. 6).

На место черного квадратика необходимо поместить аргумент функции «Двумерный 2D» график. Таким аргументом может выступать мате-

матическая функция, заданная в явном виде или двумерный массив (матрица) координат точек или система содержащая несколько объектов.

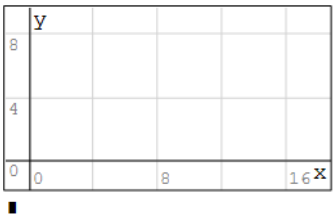


Рис. 1.6. Заготовка двумерного графика

Самым простым графиком, является график, построенный по точкам. Такие графики строятся при обработке экспериментальных данных. Пусть заданы шесть точек, в которых определено значение функции. На рис. 1.7 показаны значения аргументов и функций, заданных в виде векторов *X* и *Y*.

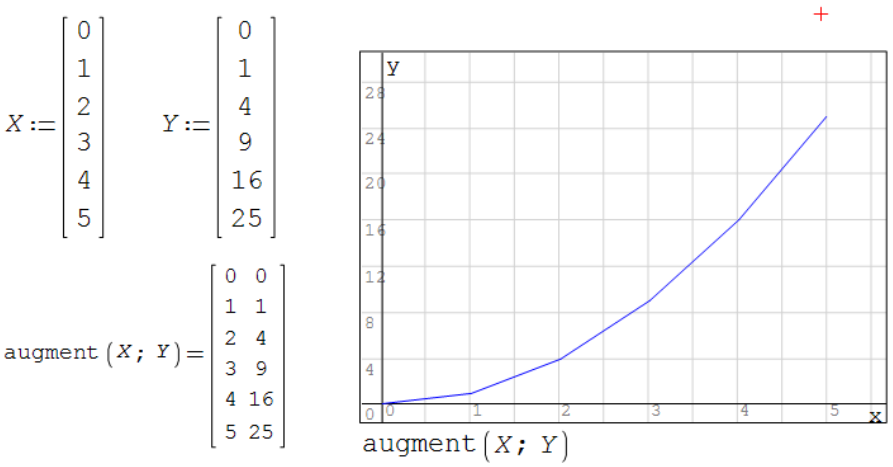
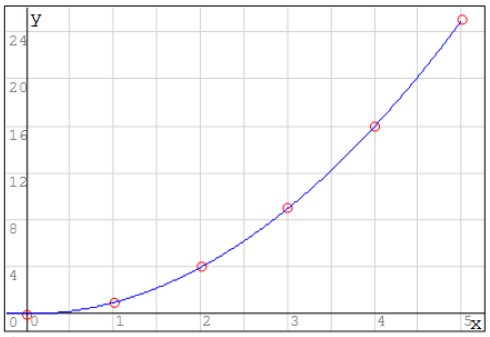


Рис. 1.7. Построение графика функции заданной по точкам

Т.к. значения заданы в виде двух векторов *X* и *Y*, а аргументом функции «Двумерный 2D» график является матрица, состоящая из двух столбцов (абсциссы и ординаты функции), сформируем данную матрицу с помощью матричной функции ***augment***(). В результате получилась ломаная линия, т.к. *SMath Studio* соединила их отрезками прямом линии, т.е. произвела линейную интерполяцию результатов.



$$XY := \text{augment}(X; Y; \text{"o"}; 10; \text{"red"})$$

$$XY = \begin{bmatrix} 0 & 0 & \text{"o"} & 10 & \text{"red"} \\ 1 & 1 & \text{"o"} & 10 & \text{"red"} \\ 2 & 4 & \text{"o"} & 10 & \text{"red"} \\ 3 & 9 & \text{"o"} & 10 & \text{"red"} \\ 4 & 16 & \text{"o"} & 10 & \text{"red"} \\ 5 & 25 & \text{"o"} & 10 & \text{"red"} \end{bmatrix}$$

$$\begin{cases} \text{cinterp}(X; Y; x) & x < 5 \\ XY \end{cases}$$

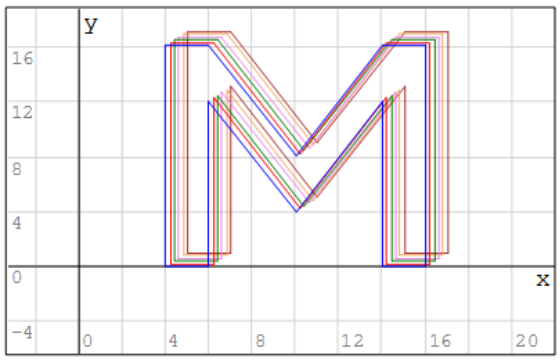
Рис. 1.8. «Сглаженный» график

Можем улучшить внешний вид графика. Для этого проведем кубическую интерполяцию функции заданной по точкам, воспользовавшись встроенной функцией ***cinterp***() и изобразим экспериментальные точки в виде красных кружков диаметром 10 пикселей. Результат показан на рис. 1.8.

В этом примере используются следующие возможности *SMath Studio*:

- Сформирована матрица XY, содержащая координаты заданной по точкам функции. Аргументами ***augment***(X; Y; «o»; 10; «red») выступают вектора абсцисс и ординат, символы на графике, размеры и цвет символов;
- Для сглаживания графика используется функция ***cinterp***(X; Y; x), отвечающая за кубическую интерполяцию заданной по точкам функции. Для ограничения кривой на графике использовали конструкцию в виде деления на $x < 5$;
- Для отображения на графике двух функций сплошной линии и шести точек, была использована конструкция – «система» $\begin{cases} \\ \end{cases}$, которая позволяет выводить на один график множество объектов. Для вызова «системы» можно воспользоваться правой панелью или комбинацией клавиш «sys» «TAB».

Функция «Двумерный 2D» график позволяет рисовать на экране различные объекты. На рис. 1.9 изображена заглавная буква «М», построенная по точкам.



$$M := \begin{bmatrix} 4 & 0 \\ 4 & 16 \\ 6 & 16 \\ 10 & 8 \\ 14 & 16 \\ 16 & 16 \\ 16 & 0 \\ 14 & 0 \\ 14 & 12 \\ 10 & 4 \\ 6 & 12 \\ 6 & 0 \\ 4 & 0 \end{bmatrix}$$

$$\begin{cases} M \\ M + 0,2 \\ M + 0,4 \\ M + 0,6 \\ M + 0,8 \\ M + 1 \end{cases}$$

Рис. 1.9. Изображение графических объектов с использованием функции «Двумерный 2D» график

Были рассмотрены наиболее простые, но часто используемые задачи, на данных примерах продемонстрированы базовые возможности пакета компьютерной математики *SMath Studio*. Этих базовых сведений должно хватить для начала самостоятельной работы с программой.

1.1. Методические материалы

Для закрепления теоретического материала предлагается самостоятельное выполнение Лабораторной работы № 1 «Изучение базовых возможностей программы *SMath Studio*».

Контрольные вопросы

1. Почему система *SMath Sdudio* считается универсальной математической системой?
2. Какие формы принимает курсор мыши в документе *SMath Sdudio*?
3. Назовите приемы управления формой курсора?
4. Из чего состоит алфавит входного языка системы *SMath Sdudio*?
5. Как вводятся вещественные числовые константы?
6. Как водятся размерные константы и для чего они используются?
7. Что называется переменной в *SMath Sdudio*?
8. Как задать (определить) переменную в программе? Какие здесь возможны ошибки.

9. Как получить числовое значение переменной?
10. Как изменить формат числа выводимого на экран?
11. Пояснить различие между глобальной и локальной переменной.
12. Как пользоваться встроенными функциями системы?
13. Как задать пользовательскую функцию?
14. Для чего предназначены ранжированные переменные в *SMath Sdudio*?
15. Как задается ранжированная переменная?
16. Что такое массив и как создается массив в системе *SMath Sdudio*?
17. Какие символьные вычисления доступны в системе *SMath Sdudio*?
18. Как создается график в *SMath Sdudio*?
19. Как в *SMath Sdudio* решить систему линейных уравнений?
20. Как в *SMath Sdudio* решить систему нелинейных уравнений?
21. Как убрать правую панель инструментов с экрана.
22. Сочетания клавиш для вызова графика, вставки матрицы, выполнения вычислений.
23. Как вывести несколько графиков в одной графической области?
24. Какое матричное уравнение необходимо применять для решения системы линейных уравнений?
25. Можно ли решить систему нелинейных уравнений с помощью матричного способа?
26. Как описать переменную в виде вектора, матрицы?
27. С какого номера начинается счет элементов вектора?
28. Сколько нужно индексных переменных для обращения к элементу матрицы *A*(3,2)?
29. Как найти сумму элементов первого столбца матрицы *A*(3,2)?
30. Какие функции применяются для поиска минимального и максимального значений?

Лабораторная работа № 1

«Изучение базовых возможностей программы *SMath Studio*»

Цель работы – приобретение навыков по использованию пакета *SMath Studio* для ввода данных, проведения математических расчетов и визуализации результатов вычислений.

Постановка задачи (задания)

1. Построить график функции $f(x)$ (согласно варианту).
2. Определить значение функции в точке $x_0 = 0.5$.
3. Используя аналитические возможности пакета *SMath Studio* определить значение первой производной $f'(x_0)$ в точке $x_0 = 0.5$.
4. Рассчитать значение определенного интеграла функции $f(x)$ на интервале $[a; b]$ (согласно варианту) с помощью математического пакета *SMath Studio*.
5. Определить корни уравнения $f(x) = 0$ используя встроенные функции *solve()* и *roots()*. Определить, какой вариант встроенных функций (зависящий от числа аргументов) наилучшим образом подходит для варианта задачи. Определить погрешность вычисления корней путем подстановки рассчитанного корня в исходное уравнение (при максимальной точности вычислений).
6. Дополнить график функции $f(x)$ изображением касательной и нормали к ней, в точке $x_0 = 0.5$.
7. Дополнить график функции $f(x)$ графиком функции $g(x)$. Определить точку пересечения двух кривых, решая соответствующее нелинейное уравнение (или систему уравнений) с использованием встроенных функций *solve()* и *roots()*. Провести проверку решения.
8. В соответствии с вариантом, создать матрицу A и вектор свободных членов b системы линейных алгебраических уравнений.
9. Определить, используя матричные функции, $|A|$, A^T, A^{-1}, $2A - A^{-1}$, $I - A^T/10$, I – единичная матрица.
10. Решить СЛАУ используя встроенную функцию *roots()* и метод обратной матрицы $x = A^{-1}b$. Рассчитать погрешность вычисления как $\varepsilon = Ax - b$.

Отчет должен содержать

1. Листинг (файл *.sm*), содержащий выполненные задания 1 – 10.

Варианты заданий (1 – 7)

0	$f(x)=e^{0.75x^2}-x^2-9tg(x^2)-2$ $g(x)=\sqrt{x^3}$	5	$f(x)=2x^3-2x^2-2\sin(x^2)+0.5$ $g(x)=3\cos(x^2)-2x^2$
1	$f(x)=\frac{x^4}{2}+3x^3-5\sin(x)$ $g(x)=3\sin(x^2)-2x+2$	6	$f(x)=\frac{x^3}{2}+3x^2-5\cos(x)$ $g(x)=0.5x^2+15$
2	$f(x)=2x^3-2x^2-2\sin(x^2)+0.5$ $g(x)=3\cos(x^2)-2x^2$	7	$f(x)=\sqrt[5]{x3-3\lvert x\rvert+7\sin(\lvert x\rvert)}$ $g(x)=3-\sqrt{x}$
3	$f(x)=e^{\frac{x^2}{5}}-x^3-2\cos(x^2)+0.75$ $g(x)=e^{-x}+x-1$	8	$f(x)=e^{\frac{x^2}{2}}-x^5-2\sin(x^2)-0.5$ $g(x)=e^{-(x+1)}+x-1$
4	$f(x)=5x-2\sin((x-3)^2)$ $g(x)=x^2+3$	9	$f(x)=2x^3-x^2-2\cos(x^2)+0.5$ $g(x)=3\cos(x^2)-2x^2$

Варианты заданий (8 – 10)

0	$\begin{cases}4x_1-x_2+x_3=8;\\2x_1+7x_2-2x_3=5;\\x_1-2x_2-9x_3=-4,\end{cases}$	5	$\begin{cases}11x_1+3x_2-4x_3=7;\\2x_1+8x_2+3x_3=5;\\x_1-x_2-x_3=-5,\end{cases}$
1	$\begin{cases}7x_1-3x_2+2x_3=6;\\2x_1+5x_2-1.1x_3=4;\\x_1-1.2x_2-8.8x_3=-9,\end{cases}$	6	$\begin{cases}9x_1-2x_2+2x_3=4;\\2x_1+8x_2-x_3=6;\\x_1-2x_2-5x_3=3,\end{cases}$
2	$\begin{cases}8x_1+3x_2-2x_3=6;\\2x_1+6x_2+x_3=4;\\2x_1-2x_2-9x_3=-7,\end{cases}$	7	$\begin{cases}4x_1+x_2-2x_3=4;\\2x_1+6x_2+x_3=3;\\2x_1-2x_2-5x_3=2,\end{cases}$
3	$\begin{cases}4x_1+2x_2-x_3=3;\\2x_1+4x_2+x_3=7;\\x_1-2x_2-7x_3=4,\end{cases}$	8	$\begin{cases}9x_1+3x_2=3;\\2x_1+x_3=9;\\x_1-2x_2-7x_3=2,\end{cases}$
4	$\begin{cases}6x_1+x_2=1;\\x_1+4x_3=1;\\x_1+x_2-7x_3=1,\end{cases}$	9	$\begin{cases}9x_1+2x_2-3x_3=6;\\2x_1+7x_2+2x_3=5;\\3x_1-3x_2-9x_3=5,\end{cases}$

2. МАТЕМАТИКА И SMATH STUDIO

2.1. Векторы. Матрицы

Фундаментальными понятиями в математике и во всех других, без исключения, областях знаний, являются понятия о векторах и матрицах. Без формирования в сознании устойчивых и адекватных образов данных понятий невозможно усвоение любых дисциплин, где эти понятия используются.

В настоящее время, сложилось, три направления использования понятия вектор (матрица), разберем их более подробно.

1) В программировании, информатике, особенно в их современных разделах *Data Science* и *Big Data*, под вектором понимают некий объект (одномерный массив) зависящий только от одного индекса, в противоположность матрице, которая может иметь несколько индексов (*n* – мерный массив). Притом вектором является строка или столбец двумерной матрицы (вектор-строка и вектор-столбец). Структура и состав данного объекта не регламентируется. Программа *SMath Studio*, в данном плане, не отстает от современных тенденций. На рис. 2.1 приведен пример задания вектора. Для ввода векторов, матриц и индексов можно воспользоваться Правой боковой панелью, вкладка «Матрицы», или горячими клавишами.

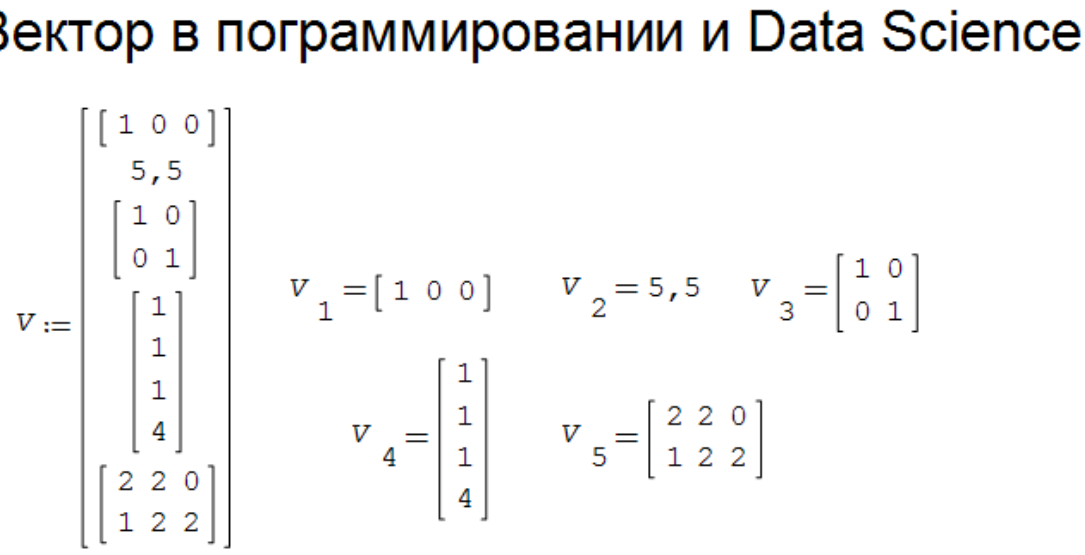


Рис. 2.1. Задание вектора в программе *SMath Studio*

Видно, что вектор *V* содержит пять разнородных элементов: вектор-строку, скаляр (матрицу размером 1x1), матрицу размером 2x2, вектор-столбец и матрицу размером 2x3. Такой подход позволят эффективно

хранить в памяти компьютера и обрабатывать большие массивы разнородной информации.

2) Примерно аналогичный подход к описанию векторов и матриц используется в некоторых разделах техники, например, в теории управления. Только во втором случае состав и структура элементов строго регламентирована: матрица управления, вектор состояния, вектор цели, вектор управления.

3) Наиболее строго понятие вектора (матрицы) введено в математике, физике и других точных науках. Как уже упоминалось выше, все математические величины могут быть представлены в виде матриц (тензоров) различного размера (ранга): матрицей размера 1x1 (или тензором нулевого ранга) записывается скалярная величина, значение которой не меняется при переходе к другим системам координат; далее следует вектор (тензор первого ранга) который можно представить в виде вектора-столбца или вектора-строки и который преобразуется по некоторому правилу (правилу направляющих косинусов) при повороте системы координат; далее следуют тензоры второго, третьего и т.д. рангов, которые в отличие от скаляра и вектора не имеют собственных названий.

Поэтому, когда мы встречаемся с данными понятиями, необходимо учитывать контекст и предметную область, где данные понятия применяются.

Рассмотрим понятие о векторах и векторных величинах (в математическом понимании). Необходимо отметить, что данное понятие сформировалось в научном обиходе только во второй половине XIX века, хотя представление о векторе как направленном отрезке берет свое начало с первых, дошедших до нас, математических трактатах: «Началах» Евклида (ок. 300 лет д.н.э) и древнекитайских «Девять глав математического искусства» X-II век д.н.э. Достаточно близко к пониманию векторов, в современном смысле данного понятия, подошли Декарт, Ферма, Лейбниц и Гаусс. Сам термин «вектор» был введен А.Гамильтоном в 1847 г. в ходе описания свойств гиперкомплексных чисел – кватернионов. Современную форму векторный анализ принял в работах Д.Гиббса и О.Хевисайда. Описание многих физических явлений в векторной форме получается очень компактным. Так Дж.Максвелл вывел свои знаменитые уравнения в виде двенадцати проекций векторного и скалярного потенциалов на декартовы оси. Оливер Хевисайд переписал уравнения Максвелла через векторы

напряженности электрического и индукции магнитного полей, получив в итоге четыре уравнения в современной форме. В начале XX века изначальная интерпретация векторов (до сих пор используемая в элементарной математике) как «направленных отрезков» сменилось на аксиоматику векторного пространства с двумя операциями: сложением векторов и умножение вектора на числа (на современном математическом языке, на элементы поля).

Для придания понятию вектора математической строгости потребуем, чтобы вектор удовлетворял следующим аксиомам (правилам):

$$
\begin{aligned}
&1)\ \vec{a}+\vec{b}=\vec{b}+\vec{a} \quad (\textit{коммуникативность})\\
&2)\ \vec{a}+(\vec{b}+\vec{c})=(\vec{a}+\vec{b})+\vec{c} \quad (\textit{ассоциативность})\\
&3)\ \vec{a}+0=0+\vec{a}=\vec{a} \quad (\textit{существование нулевого вектора})\\
&4)\ \vec{a}+(-\vec{a})=\vec{a}-\vec{a}=0 \quad (\textit{существование противоположного ввектор})\\
&5)\ \beta\cdot\vec{a}=\vec{a}\cdot\beta;\ \beta\cdot(\vec{a}+\vec{b})=\beta\cdot\vec{a}+\beta\cdot\vec{b} \quad (\textit{дистрибутивность})\\
&6)\ 1\cdot\vec{a}=\vec{a} \quad (\textit{унтарность})
\end{aligned}
\qquad (1)
$$

Для лучшего понимания понятия вектор, изобразим вектор в виде отрезка со стрелкой, как было принято в средней школе. Для этого построим 2D график в программе *SMath Studio* по точкам. На рис. 2.1. показано два варианта построения отрезков, единичной длины, направленных вдоль оси *OX*, со стрелками разной формы. Первый столбец матриц *arrow* и *Arrow* координаты точек по оси *OX*, второй столбец по оси *OY*.

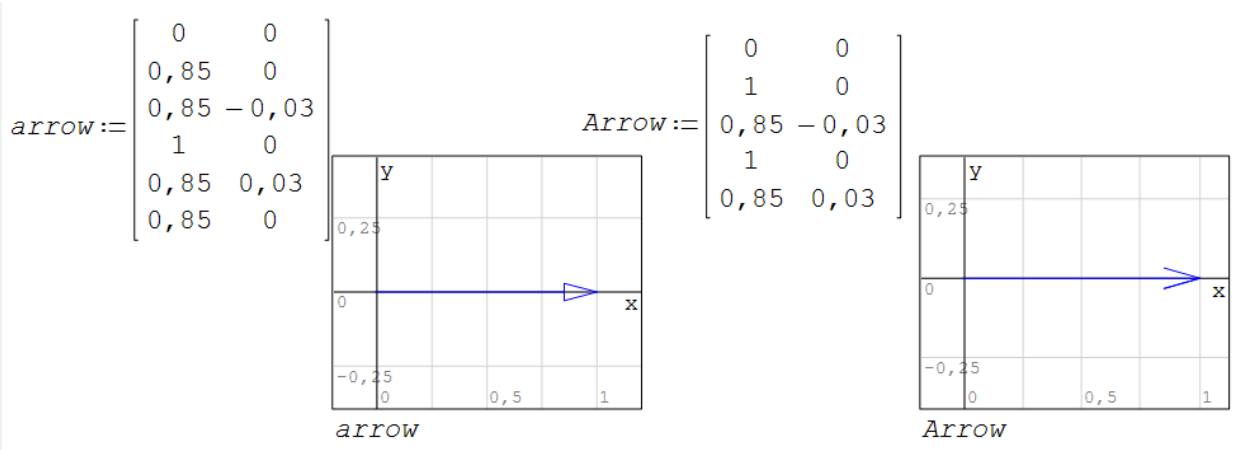


Рис. 2.1. 2D изображение вектора в виде стрелки

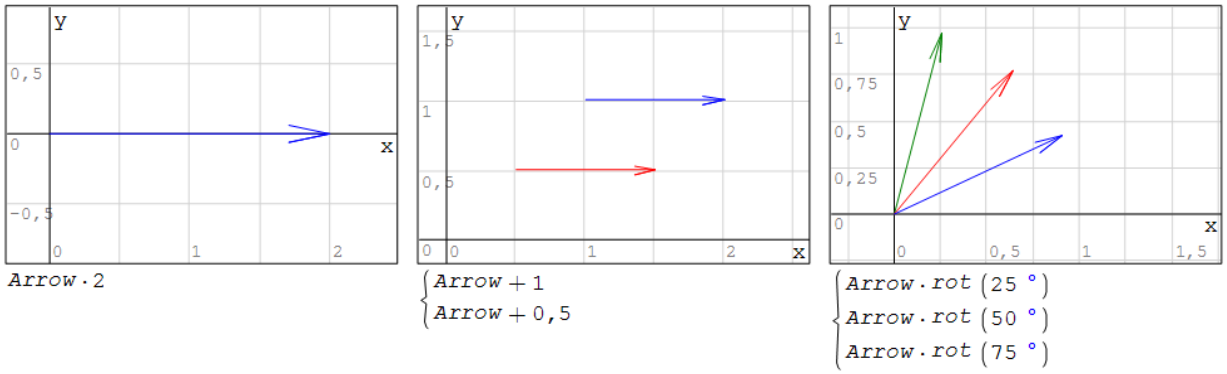


Рис. 2.2. Масштабирование, трансляция и вращение графического объекта

На рис. 2.2. приведены основные операции с графическими объектами. Операция масштабирования заключается в поэлементном умножении компонент матрицы содержащей графический объект на некоторый масштабный коэффициент, в данном случае мы растягиваем стрелку в два раза. Трансляция или смещение осуществляется путем добавления ко всем координатам объекта какого-либо числа, например, единицы и 0.5. Для вращения объекта относительно начала координат (или относительно оси *OZ*, направленной на нас), необходимо матрицу с координатами объекта умножить на матрицу вращений (приведена на следующем рисунке).

Научимся манипулировать данными векторами, т.к. в дальнейшем нам придется работать с различными графическими объектами, напишем подпрограмму (пользовательскую функцию), которая будет перемещать изображение на экране, вращать изображение и менять его масштаб. Эта функция ***Affine****(x; y; Angle; Scale; Body)*, где *x*, *y* – трансляции вдоль осей, на которые перемещается графический объект *Body*, *Angle* – угол на которой разворачивается объект, *Scale* – масштабный коэффициент, указывающий во сколько раз изменяется размер объекта. Название функции происходит из аффинного преобразования, которое не меняет углы объектов при трансляции, повороте и масштабировании. Объект *Body* задается по точкам с координатами (x_i; y_i) в виде двумерного массива. Текст пользовательской функции ***Affine()*** изображен на рис. 2.3. Функция представляет программный блок (вводится клавишей «]»), в первой строке формируется пустая (нулевая) вспомогательная матрица, состоящая из оного столбца, в которой число строк совпадает с числом строк в матрице *Body*.

```
Affine(x; y; Angle; Scale; Body) :=
    A := matrix(rows(Body); 1)
    "Координаты точки, в которую переносится"
    "начало координат"
    Trans := augment(A + x; A + y)
    "Матрица поворота тела на заданный угол"
    Ω := eval([[ cos(Angle)  sin(Angle) ]
               [ -sin(Angle) cos(Angle) ]])
    Trans + Scale · Body · Ω
```

Рис. 2.3. Пользовательская функция манипулирования графическими объектами на 2D графике

Трансляция объекта осуществляется объединением двух матриц содержащих первоначальные абсциссы и ординаты объекта *Body* к которым прибавляются трансляции вдоль осей *X* и *Y*. Далее формируется матрица поворота ·, с помощью которой производится поворот объекта на угол *Angle*. Функция ***Affine***() возвращает перемещенный объект, который масштабируется в *Scale* раз и поворачивается на угол *Angle* против часовой стрелки (такое направление в математике считается положительным).

Теперь появилась возможность манипулировать векторами, например, для визуализации операций по сложению и вычитанию векторов.

$$\begin{cases} Affine(0; 5; 0; 5; arrow) \\ Affine\left(0; 0; \frac{\pi}{2}; 5; arrow\right) \\ Affine\left(0; 0; \frac{\pi}{4}; 5 \cdot \sqrt{2}; arrow\right) \end{cases}$$

Рис. 2.4. Сложение векторов на 2D графике

В простейшем случае, проведем сложение векторов одинаковой длины, в данном случае равной 5 (Scale = 5), первый вектор направлен по оси *OX* и смещен по оси *OY* на 5 единиц, второй вектор повернут на 90^0 = п/2 относительно оси *OX*. В результате векторного сложения получает-

ся третий вектор длинной 5·√2 (по теореме Пифагора), направленный под углом 45^0 к оси *OX*. Видно, что сложение векторов проведено по правилу треугольника.

Можно рассмотреть произвольный случай сложения векторов различной длины и направления. На рис. 2.5 представлен пример графического сложения двух векторов произвольной длинны и направлений. Для сложения векторов используется обычная операция сложения (обведена рамкой). Правее показана операция покоординатного сложения элементов векторов. Для определения угла наклона результирующего вектора используется свойство скалярного произведения вектора $\vec{c}$ и оси *OX* = (1 0 0)^T.

Рис. 2.5. Графическое представление векторного сложения в *SMath Studio*

Используя функцию ***affine()*** на 2D графике проведем визуализацию операции сложения двух векторов по правилам треугольника (слева) и

параллелограмма (справа). На рис. 2.5 заданы два вектора один a_1 длинной 7 единиц и направленный под углом 75^0 к оси *OX*, второй b_1 имеет длину равную 10 и направление 10^0 к оси *OX*. В отличие от рис. 2.4 (где векторы заданы на плоскости или наукообразно в двумерном пространстве) в данном примере используется запись двумерных векторов в трехмерном пространстве, координата по оси *OZ* равна 0. Для записи вектора используются обозначения:

$$a_x = a_1,\ \ a_y = a_2,\ \ a_z = a_3,\ \ \vec{i} = \vec{e}_x,\ \ \vec{j} = \vec{e}_y,\ \ \vec{k} = \vec{e}_z,$$

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = (a_1\ \ a_2\ \ a_3)^T;\quad \vec{a} = \vec{i}\,a_1 + \vec{j}a_2 + \vec{k}a_3, \qquad (1)$$

где проекции вдоль координатных осей *OX*, *OY* и *OZ* нумеруются числами 1, 2 и 3, орты или единичные вектора вдоль данных осей обозначены как *i*, *j*, *k* (базис), а сам вектор a записывается или в виде вектора-столбца или транспонированного вектора-строки, разложение вектора *a* по базису (*i*, *j*, *k*) через компоненты (проекции на координатные оси) a_1, a_2, a_3.

Важнейшим свойством вектора является его длина. *SMath Studio* предоставляет несколько способов вычисления длины вектора. На рис. 2.6 рассчитана длина вектора с из предыдущего примера.

$|X| := \sqrt{\sum_{i=1}^{\mathrm{rows}(X)} X_i^2}$ $\quad l(u) := \sqrt{u \cdot u}$ — пользовательские функции

$\sqrt{c_1^2 + c_2^2} = 14,428$ — по теореме Пифагора

$\mathrm{norme}(c) = 14,428$ — встроенная функция расчета эрмитовой нормы

$\sqrt{c \cdot c} = 14,428$
$\sqrt{c^2} = 14,428$
$l(c) = 14,428$
— использование формулы скалярного произведения

$|c| = 14,428$ — использование пользовательской функции $|X|$, переопределяющей функцию определения модуля числа

Рис. 2.6. Вычисление длины вектора в *SMath Studio*

Кроме операций сложения и умножения на число, для векторных величин определяются операции произведения векторов: скалярное, векторное, смешанное, псевдоскаляр- ное, двойное векторное.

Скалярное произведение двух векторов – это число равное произведению модулей этих векторов на косинус угла между ними, т.е. $\vec{a}\cdot\vec{b}=|\vec{a}|\cdot|\vec{b}|\cdot\cos(\angle ab)$.

Свойства скалярного произведения:

$$\begin{aligned}&1)\ \vec{a}\cdot\vec{b}=\vec{b}\cdot\vec{a}\\&2)\ \vec{a}\cdot\vec{a}=\vec{a}^2=|\vec{a}|^2\\&3)\ \lambda\cdot(\vec{a}\cdot\vec{b})=\lambda\cdot\vec{a}\cdot\vec{b}\\&4)\ \vec{a}\cdot(\vec{b}+\vec{c})=\vec{a}\cdot\vec{b}+\vec{a}\cdot\vec{c}\end{aligned}\qquad(2)$$

Условие перпендикулярности двух векторов:

$$5)\ \vec{a}\cdot\vec{b}=0$$

Использование скалярного произведения позволяет определять длину вектора, угол между векторами, проекцию одного вектора на другой, определение ортогональности и коллинеарности между векторами.

При преобразовании (повороте) системы координат компоненты вектора должны меняться по строго определенному закону – правилу направляющих косинусов:

$$\begin{cases}a'_1=c_{11}a_1+c_{12}a_2+c_{13}a_3\\a'_2=c_{21}a_1+c_{22}a_2+c_{23}a_3,\\a'_3=c_{31}a_1+c_{32}a_2+c_{23}a_3\end{cases}$$

$$a'_i=\sum c_{ij}a_j,\quad a'_i=c_{ij}a_j,\quad i,j=1..3,$$

$$c_{ij}=\cos(\angle x'_i,x_j),\qquad(3)$$

где (a'_1, a'_2, a'_3) – координаты вектора в новой системе координат, (a_1, a_2, a_3) – координаты вектора в старой (исходной) системе координат, c_{ij} – косинусы углов между новыми и «старыми» осями. В предпоследней строчке выражения (2) приведена сокращенная форма исходной системы преобразований координат, позволяющая существенно сократить запись векторных преобразований. Это сокращение предложено А.Эйнштейном и заключается в том, что знак суммирования можно не писать, если число одинаковых индексов в правой части равенства равно двум (в нашем

примере индекс *j* встречается два раза).

Для однозначности задания всех векторных операций необходимо задать систему координат. Такой системой, для трехмерного пространства, является правая декартова система координат. Упорядоченная тройка некомпланарных векторов *X*, *Y*, *Z* называется правой тройкой, если после откладывания от одной точки *O* векторов *X*, *Y*, *Z* кратчайший поворот *OX* до совмещения с *OY* наблюдается из точки Z совершающимся против часовой стрелке. В противном случае тройка векторов называется левой. Для лучшего запоминания расположения векторов используется правило правой руки (рис. 2.7).

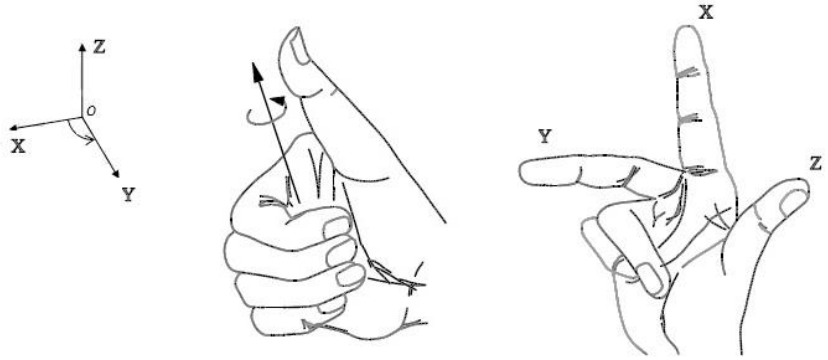


Рис. 2.7. Правая тройка векторов

Если векторы $\vec{a}, \vec{b}$ и $\vec{c}$ не коллинеарные, то векторным произведением векторов $\vec{a}$ и $\vec{b}$ называется вектор $\vec{c}$ удовлетворяющий условиям:

1) $|\vec{c}| = |\vec{a}|\,|\vec{b}|\sin(\angle\vec{a}\vec{b})$,

2) $\vec{c} \perp \vec{a}$ и $\vec{b} \perp \vec{c}$,

3) $\vec{a}, \vec{b}$ и $\vec{c}$ образуют правую тройку векторов.

4) Если $\vec{a} \parallel \vec{b}$, то $\vec{c} = 0$ (для коллинеарных $\vec{a}$ и $\vec{b}$).

Векторное произведение векторов $\vec{a}$ и $\vec{b}$ обозначается символом $\vec{a} \times \vec{b}$ или $[\vec{a}\,\vec{b}]$. Зная координаты векторов можно найти их векторное произведение. Пусть в правом ортонормированном базисе $(\vec{i}, \vec{j}, \vec{k})$ векторы $\vec{a}(a_1, a_2, a_3)$ и $\vec{b}(b_1, b_2, b_3)$, тогда векторное произведение векторов вычисляется по координатной формулой векторного произведения:

$$\vec{a}\times\vec{b}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ x_1 & y_1 & z_1\\ x_2 & y_2 & z_2\end{vmatrix}, \qquad (4)$$

определитель может быть раскрыт или правилу треугольника или разложением по элементам первой строки (столбца) или по правилу Саррюса (рис. 2.8). В последнем случае, надо дописать два первые столбца справа (или две первые строки ниже) данного определителя, и рассчитать определитель, вычеркивая диагональные элементы, как показано ниже (линии направленные слева – направо соответствуют положительным слагаемым, а направленные справа-налево – отрицательным). Векторное произведение находится в виде суммы компонент вдоль координатных осей (положительные и отрицательные слагаемые сгруппированы в виде определителей второго порядка):

$$\vec{a}\times\vec{b}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\end{vmatrix}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k} & \vec{i} & \vec{j}\\ a_1 & a_2 & a_3 & a_1 & a_2\\ b_1 & b_2 & b_3 & b_1 & b_2\end{vmatrix}=\vec{i}\begin{vmatrix}a_2 & a_3\\ b_2 & b_3\end{vmatrix}+\vec{j}\begin{vmatrix}a_3 & a_1\\ b_3 & b_1\end{vmatrix}+\vec{k}\begin{vmatrix}a_1 & a_2\\ b1 & b_2\end{vmatrix},$$

$$\vec{a}\times\vec{b}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\end{vmatrix}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\\ \vec{i} & \vec{j} & \vec{k}\\ a_1 & a_2 & a_3\end{vmatrix}=\vec{i}\begin{vmatrix}a_2 & a_3\\ b_2 & b_3\end{vmatrix}+\vec{j}\begin{vmatrix}a_3 & a_1\\ b_3 & b_1\end{vmatrix}+\vec{k}\begin{vmatrix}a_1 & a_2\\ b1 & b_2\end{vmatrix}.$$

Рис. 2.8. Правило Саррюса для расчета определителя третьего порядка

Векторное произведение можно использовать при нахождении площади параллелограмма и треугольника. Если векторы $\vec{a}$ и $\vec{b}$ неколлинеарны, то модуль их векторного произведения равен площади параллелограмма (а половина модуля площади треугольника), построенного на этих векторах (рис. 2.9):

$$S_{OACB}=\left|\vec{a}\times\vec{b}\right|=\left|\vec{a}\right|\left|\vec{b}\right|\sin(\angle\vec{a}\vec{b}),\quad S_{OAB}=\frac{1}{2}\left|\vec{a}\times\vec{b}\right|=\frac{1}{2}\left|\vec{a}\right|\left|\vec{b}\right|\sin(\angle\vec{a}\vec{b}) \qquad (5)$$

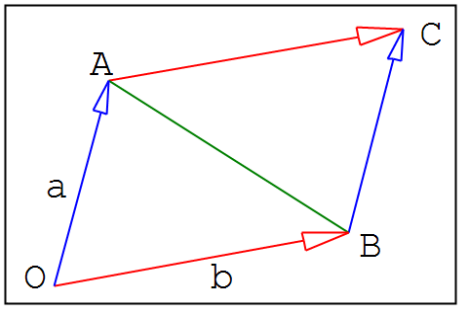


Рис. 2.9. Определение площади парраллелепипеда и треугольника построенного на векторах *a* и *b*

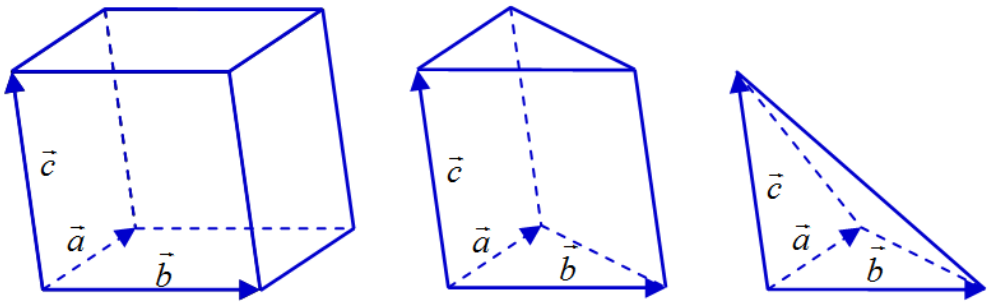
Произведение двух «истинных» или полярных векторов по формуле (4) приводит к т.н. аксиальному вектору или псевдовектору (пример, угловая скорость), который не подчиняется формуле (3), но используя правило правой руки, может использоваться также как полярный вектор.

Смешанным произведением трех векторов $\vec{a}, \vec{b}$ и $\vec{c}$ называется число, которое равно скалярному произведению векторного произведения векторов $\vec{a}$ и $\vec{b}$ на вектор $\vec{c}$.

Смешанное произведение векторов $\vec{a}, \vec{b}$ и $\vec{c}$ обозначается как $(\vec{a} \times \vec{b}) \cdot \vec{c}$ или $(\vec{a}\ \vec{b}\ \vec{c})$ и равно определителю

$$(\vec{a} \times \vec{b} \cdot \vec{c}) = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}.$$

В геометрическом смысле, смешанное произведение численно равно объему параллелепипеда (или призмы или пирамиды) построенному на данных векторах (рис. 2.10).



$$V_{par} = \left|(\vec{a}\vec{b}\vec{c})\right|, \qquad V_{pri} = \frac{1}{2}\left|(\vec{a}\vec{b}\vec{c})\right|, \qquad V_{pyr} = \frac{1}{6}\left|(\vec{a}\vec{b}\vec{c})\right|$$

Рис. 2.10. Определение объема параллелепипеда, призмы и пирамиды построенной на векторах $\vec{a}, \vec{b}$ и $\vec{c}$

Если смешанное произведение равно нулю, то вектора $\vec{a}, \vec{b}$ и $\vec{c}$ компланарны, т.е. лежат в одной плоскости. Смешанное произведение удобно использовать для контроля выбранной системы координат (правой или левой), если $(\vec{a}\ \vec{b}\ \vec{c}) > 0$, то вектора $\vec{a}, \vec{b}$ и $\vec{c}$ образуют правую тройку векторов, если $(\vec{a}\ \vec{b}\ \vec{c}) < 0$ – левую тройку векторов.

На рис 2.11. приведены примеры расчета скалярного, векторного и смешанного произведения в программе *SMath Studio*. Для проведения вычислений векторные величины необходимо представить в виде вектор-столбцов. Скалярное произведение вводится как обычное произведение через клавишу «*», для векторного произведения можно воспользоваться правой панелью (вкладка «Векторы») или горячей клавишей «Ctrl»«8», смешанное произведение предполагает последовательное применение данных операций.

Пример скалярного, векторного и смешанного произведения векторов

$$a := 10 \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad b := 5 \cdot \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \\ 0 \end{bmatrix} \qquad \alpha := \frac{\pi}{4}$$

Скалярное произведение $c = (\vec{a} \cdot \vec{b})$ $\quad c = \sum_{i=1}^{3} a_i \cdot b_i$ $\quad c = |a| \cdot |b| \cdot \cos(\alpha)$

$$c := a \cdot b = 35,3553 \qquad c := \sqrt{a^2} \cdot \sqrt{b^2} \cdot \cos(45\ °) = 35,3553$$

Векторное произведение $\vec{c} = \vec{a} \times \vec{b}$ $\quad \vec{a} \times \vec{b} = \begin{bmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$ $\quad |\vec{c}| = |a| \cdot |b| \cdot \sin(\alpha)$

$$a = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix} \quad b = \begin{bmatrix} 3,535534 \\ 3,535534 \\ 0 \end{bmatrix} \quad a \times b = \begin{bmatrix} 0 \\ 0 \\ 35,355339 \end{bmatrix} \qquad |\vec{c}| := \sqrt{a^2} \cdot \sqrt{b^2} \cdot \sin(\alpha) = 35,3553$$

Смешанное произведение $d = \vec{a} \times \vec{b} \cdot \vec{c}$ $\quad \vec{a} \times \vec{b} \cdot \vec{c} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$

$$c := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad d := a \times b \cdot c = 35,3553$$

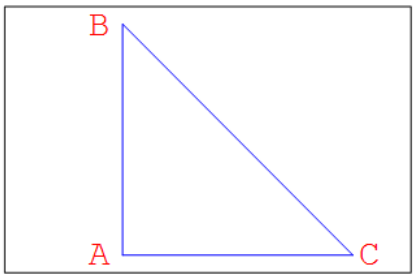
Рис. 2.11. Расчет скалярного, векторного и смешанного произведения

Для закрепления материала по основным векторным операциям, рассмотрим задачу на аналитического нахождения площади треугольника, заданного координатами его вершин. Пример приведен на рис. 2.12.

Запишем радиус векторы каждой из вершин в виде радиус-векторов OA, OB и OC и рассчитаем вектора AB, AC и BC (где вторая буква соответствует концу вектора, а первая его началу). Из определения скалярного произведения определим углы между векторами. Проведем проверку вычислений, сумма углов треугольника равна 180^0. Расчет площади треугольника проведем, используя свойство векторного произведения. Для проверки вспомним формулу Герона для расчета площади треугольника через известные длины сторон (модули векторов) и полупериметр треугольника.

Аналитическмя геометрия. Пример

$A := [2\ -1\ 3]$ $B := [2\ -1\ 3]$ $C := [2\ -1\ 3]$



$$OA := \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} \quad OB := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad OC := \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

$$AB := OB - OA = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix} \quad AC := OC - OA = \begin{bmatrix} -2 \\ 1 \\ 2 \end{bmatrix} \quad BC := OC - OB = \begin{bmatrix} -1 \\ -1 \\ 4 \end{bmatrix}$$

$$\alpha := \arccos\left(\frac{AB \cdot AC}{\sqrt{AC \cdot AC} \cdot \sqrt{AB \cdot AB}}\right) = 90\ ^\circ \qquad \beta := \arccos\left(\frac{(-AB) \cdot BC}{\sqrt{AB \cdot AB} \cdot \sqrt{BC \cdot BC}}\right) = 45\ ^\circ$$

$$\gamma := \arccos\left(\frac{(-AC) \cdot (-BC)}{\sqrt{BC \cdot BC} \cdot \sqrt{AC \cdot AC}}\right) = 45\ ^\circ$$

$\alpha + \beta + \gamma = 180\ ^\circ$ Проверка: сумма углов треугольника равна 180°

$S := \frac{1}{2} \cdot \sqrt{(AB \times AC)^2} = 4,5$ Площадь треугольника

Проверка:

$\sqrt{AB \cdot AB} = 3$ $\sqrt{AC \cdot AC} = 3$ $\sqrt{BC \cdot BC} = 4,2426$ Длины векторов

$P := \frac{\sqrt{AB^2} + \sqrt{AC^2} + \sqrt{BC^2}}{2} = 5,1213$ Полупериметр

Площадь треугольника по формуле Герона

$$S_1 := \sqrt{P \cdot \left(P - \sqrt{AB^2}\right) \cdot \left(P - \sqrt{AC^2}\right) \cdot \left(P - \sqrt{BC^2}\right)} = 4,5$$

Рис. 2.12. Пример по аналитической геометрии

2.2. Решение СЛАУ

Решение систем линейных алгебраических уравнений (СЛАУ) является важнейшей вычислительной задачей. Большинство математических моделей физики, электротехники, экономики, статистики либо сразу строятся как СЛАУ, либо сводятся к таковым посредством дискретизации или линеаризации. Методов решения СЛАУ очень много поэтому очень важно уметь выбирать наиболее оптимальные методы для каждой кон-

кретной задачи.

Обычно СЛАУ *n*-го порядка записывается в виде

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}, \qquad \sum_{j=1}^{n} a_{ij}x_j = b_i, \quad Ax = b, \tag{6}$$

где *i*, *j* – номера строк и столбцов матрицы *A*,

$$A = \begin{bmatrix} a_{11}\ a_{12} \cdots a_{1n} \\ a_{21}\ a_{22} \cdots a_{2n} \\ \cdots \\ a_{n1}\ a_{n2} \cdots a_{nn} \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

В соотношениях (6): *A* – матрица коэффициентов системы уравнений с n^2 элементами; *x* – вектор-столбец неизвестных; *b* – вектор-столбец свободных членов.

Необходимым условием разрешимости СЛАУ является равенство числа уравнений числу неизвестных.

Методы решения СЛАУ делятся на две группы:

– прямые (точные) методы;

– итерационные (приближенные) методы.

К ***прямым*** методам относятся такие методы, которые, в предположении, что вычисления ведутся без округлений, позволяют получить точные значения неизвестных. Они просты, универсальны и используются для широкого класса систем. Однако они не применимы к системам больших порядков ($n > 100$) и к плохо обусловленным системам из-за возникновения больших погрешностей. К ним можно отнести: ***правило Крамера***, метод ***обратной матрицы***, методы ***Гаусса***, ***прогонки*** и др.

К ***приближенным*** относятся методы, которые даже в предположении, что вычисления ведутся без округлений, позволяют получить решение системы лишь с заданной точностью. Это итерационные методы, т.е. методы последовательных приближений. К

ним относятся методы ***простой итерации***, ***Зейделя*** и др. Кратко рассмотрим данные методы.

Самым простым (по форме записи и реализации в *SMath Studio*) методом решения СЛАУ является метод ***обратной матрицы***.

Если задана система уравнений ***Ax = b***. То умножая левую и правую части этого выражения на A^{-1}, учитывая, что по определению $A^{-1} A = 1$ (или единичной матрице *E*), получаем решение системы:

$$A^{-1} A\, x = A^{-1} b; \qquad x = A^{-1} b.$$

Условием применимости метода обратной матрицы, является определенность матрицы *A* (не равенство нулю определителя матрица).

Обратная матрица, в пакете *SMath Studio*, вычисляется достаточно быстро. Это позволяет проводить оценку вычислительной эффективности изучаемых алгоритмов решения СЛАУ на основе их сравнения с методом обратной матрицы.

Правило (метод) Крамера

Рассмотрим систему (6). Как отмечалось выше, если определитель этой системы не равен нулю, то система будет иметь место единственное решение. Это необходимое и достаточное условие. Тогда по правилу Крамера

$$x_i = \frac{|A_i|}{|A|}, \quad i = 1, n, \qquad (7)$$

где $|A_i|$ – определитель матрицы, которая получается из матрицы *A* путем замены элементов *i*-го столбца свободными членами *b*.

Метод Крамера требует вычисления $n + 1$ определителей размерности $n \times n$. Следует отметить, что вычисление определителей высоких порядков представляет существенную вычислительную проблему. Полное число операций при использовании метода Крамера составляет $n!$ (*n* факториал). На рис. 2.13. представлен пример решения СЛАУ методом Крамера а программе *SMath Studio*. Для системы уравнений 4×4, число операций составляет 4! = 24, а для системы 10×10, число операций составит уже 10! = 3 628 800, что делает применение данного метода неэффективным.

Кроме исторической и дидактической роли, метод Крамера использу-

ется, когда необходимо находить только отдельные корни уравнений, или для проведения аналитических преобразований систем малой размерности.

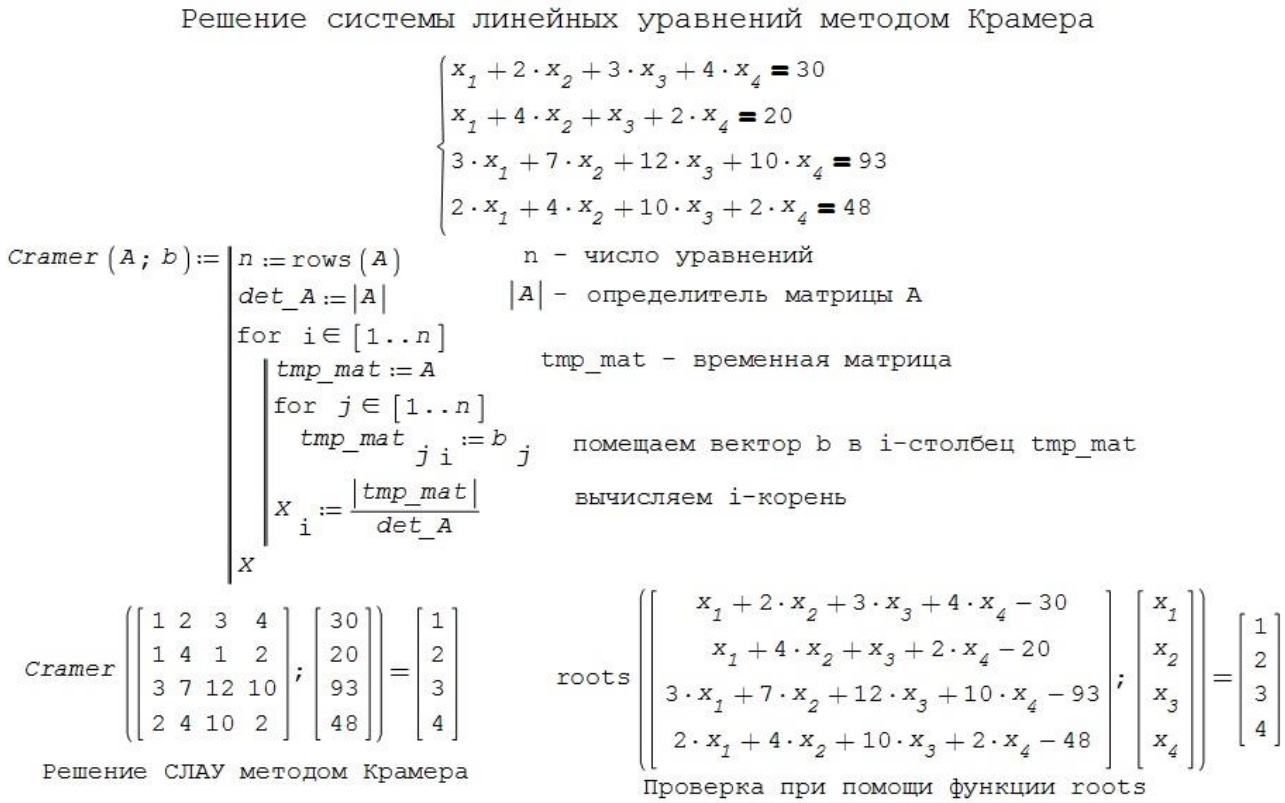


Рис. 2.13. Решение СЛАУ с использованием метода Крамера

Метод Гаусса

Этот метод является наиболее распространенным методом решения СЛАУ. Метод Гаусса еще называется методом последовательного исключения неизвестных, при котором исходная система уравнений приводится к треугольному виду, в которой коэффициенты лежащие ниже главной диагонали равны нулю.

Необходимо отметить, что первое упоминание о данном методе встречается в восьмой главе древнекитайского трактата «Девять глав математического искусства» датированного вторым веком до нашей эры. О данном трактате в Европе, до середины 18 века известно не было, поэтому название метода связывается с именем Гаусса, который подробно описал данный метод.

Существуют несколько вычислительных алгоритмов, реализующих данный метод. Рассмотрим наиболее универсальный метод Гаусса с выбором главного элемента по столбцу. Реализация и пример использова-

ния метода приведены на рис. 2.14.

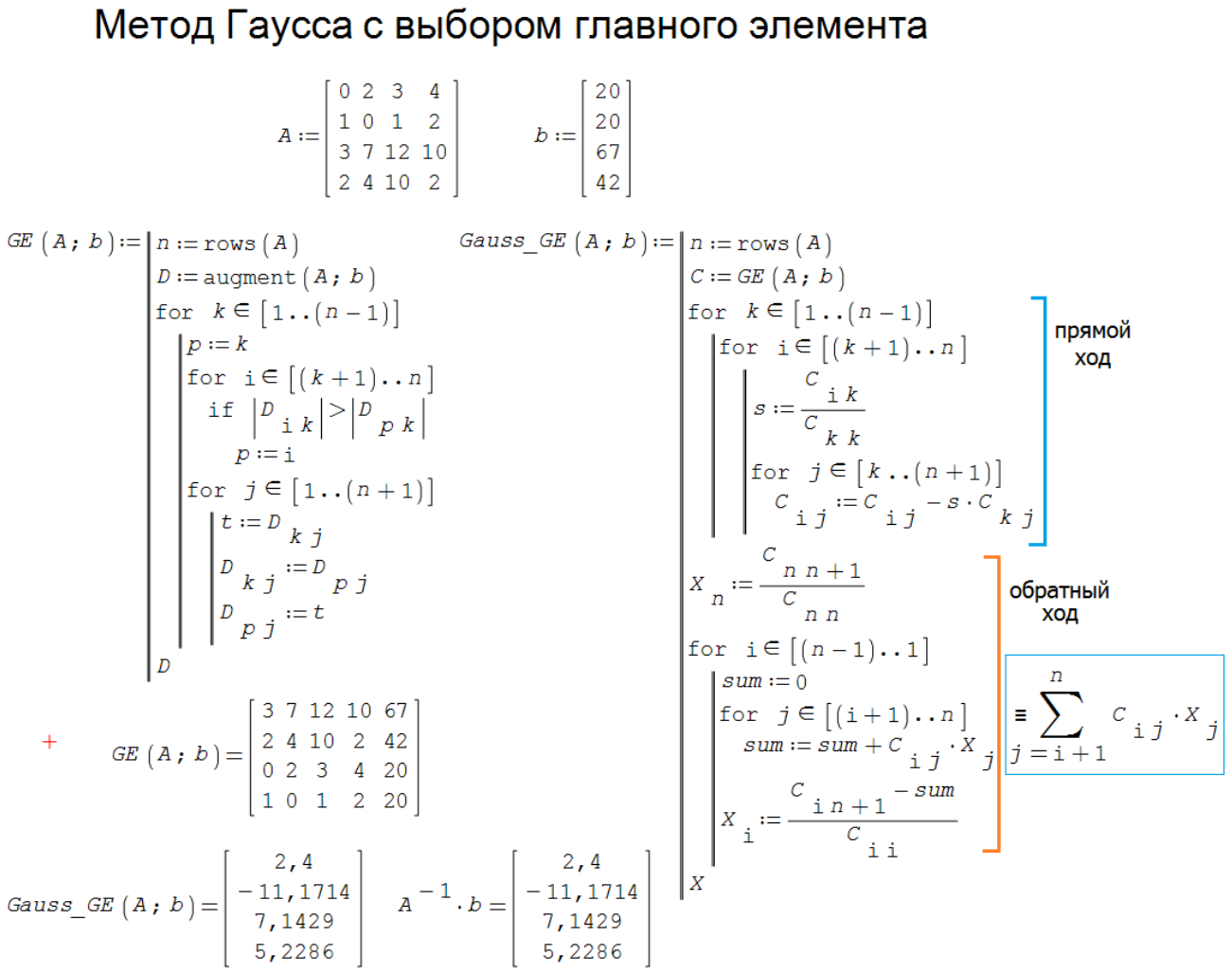


Рис. 2.14. Метод Гаусса с выбором главного элемента по столбцу

В качестве примера, выбрана матрица, у которой на главной диагонали находятся нулевые элементы. Простой метод Гаусса в данном случае выдал бы ошибку. Чтобы этого избежать напишем пользовательскую функцию ***GE****(A; b)* которая будет переставлять строки в исходной системе уравнений, чтобы на главной диагонали находились максимальным по модулю элементы. В ходе работы функции ***GE****(A; b)* формируется вспомогательная расширенная матрица *D*, путем объединения матрицы *A* и вектора *b*. Далее находится максимальный элемент в первом столбце, после чего происходит замена первой строки матрицы D на строку, содержащую максимальный элемент в первом столбце. Это происходит в цикле по номеру строки *k*. Результат работы функции ***GE****(A; b)* приведен на рис. 2.15. Видно, что теперь все диагональные элементы не нулевые.

Переходим к основной функции ***Gauss_GE****(A; b)*. Из рисунка видно,

что функции ***Gauss_GE***(*A; b)* состоит из двух частей, так называемых – «прямого хода» и «обратного хода» (выделены скобками). В первой части алгоритма последовательного исключения происходит умножение первого уравнения системы (5) на коэффициент a_{21}, а второго уравнения, на коэффициент a_{11} и вычитание первого уравнения из второго, на месте первого элемента во второй строке получается ноль. Повторяя данную операцию для последующих уравнений, исключаем коэффициенты перед неизвестными в первом столбце, кроме коэффициента a_{11}. Аналогично используя «новую» вторую строку, исключаем коэффициенты во втором столбце стоящие под коэффициентом a'_{22} (штрих означает, что коэффициент преобразован на предыдущем шаге). Исключение неизвестных повторяется до тех пор, пока в последнем уравнении не останется только один ненулевой коэффициент a_{nn}. В результате мы получаем треугольную матрицу A'.

Из последней строки можно найти неизвестный корень $X_n = b'_n / A'_{nn}$, далее находятся остальные корни СЛАУ по формуле

$$X_i = \frac{b'_i - \sum_{j=i+1}^{n} A'_{ij} \cdot X_{i-1}}{A'_{ii}}, \qquad (8)$$

где A' и b' – коэффициенты матриц пересчитанных во время выполнения «прямого хода» метода Гаусса.

На рис. 2.15 в тексте программы используется суммирование с использованием цикла ***for***, в рамке показан альтернативный способ суммирования с помощью встроенной функции. В нижней части рисунка приведен результат работы функции ***Gauss_GE***(*A; b)* в сравнении корнями СЛАУ, найденными с помощью обратной матрицы.

Основными достоинствами метода Гаусса являются:

а) менее трудоёмкий по сравнению с методами обратной матрицы и Крамера;

б) позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение;

в) позволяет найти максимальное число линейно независимых уравнений – ранг матрицы системы.

Один из основных недостатков метода Гаусса заключается в накап-

ливании вычислительной погрешности. Число умножений и делений в методе Гаусса составляет $n^3/3$, т.е. нарастает в кубической степени от размера матрицы.

Метод прогонки (метод Томаса)

Часто, при решении различных вычислительных задач, приходится иметь дело с матрицами специального вида. Использование особенностей таких матриц позволяет существенно сократить расчетное время. Наиболее наглядно это проявляется при работе с трех диагональными матрицами, которые возникают, в частности, при численной решении краевых задач математической физики. В трех диагональных матрицах отличны от нуля только элементы, лежащие на главной диагонали, а так же на и под ней.

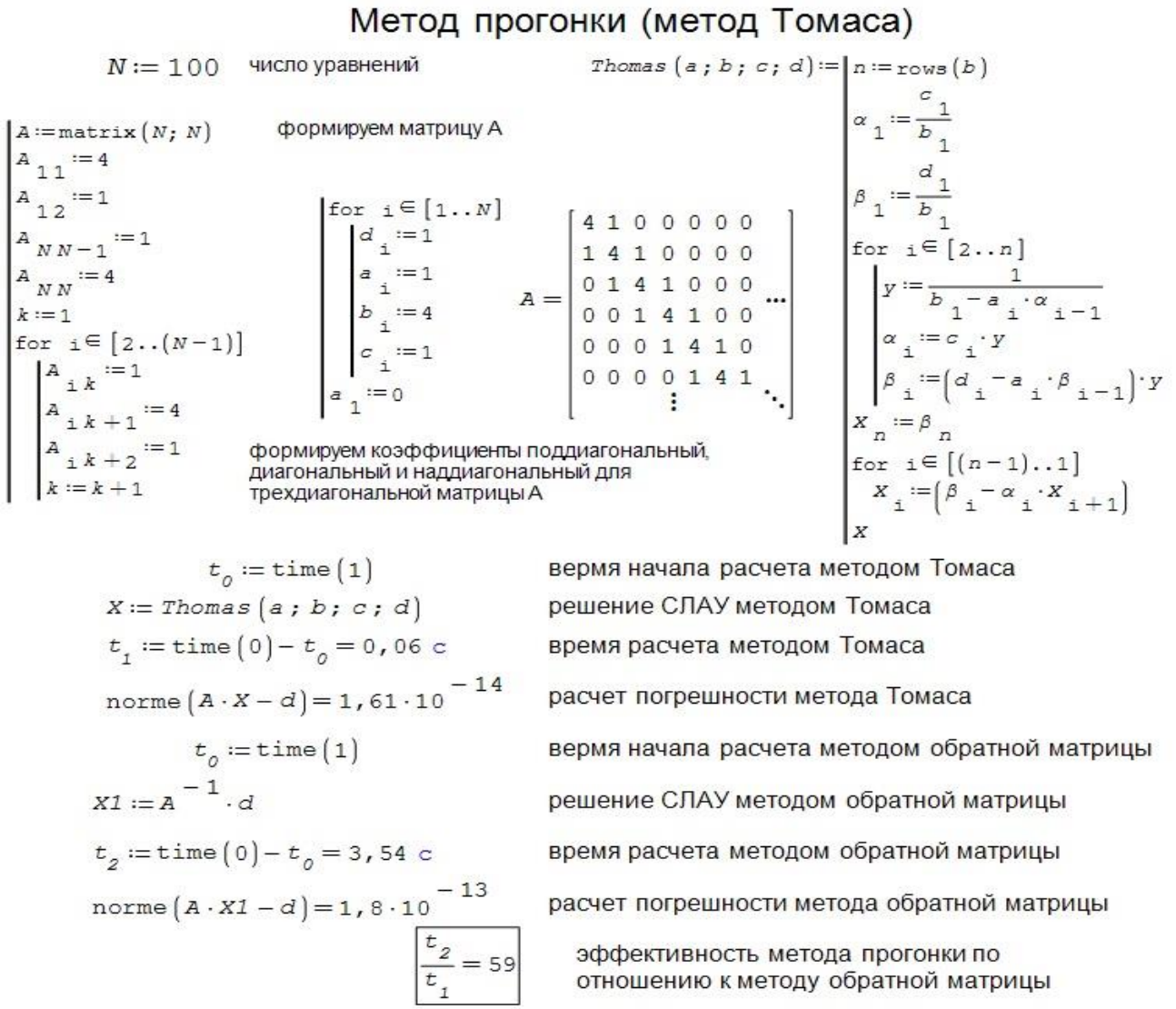


Рис. 2.15. Метод прогонки (Томаса) в сравнении с методом обратной матрицы

Данный метод также является модификацией метода Гаусса для частного случая *разреженных* систем – систем с матрицей трехдиагональноо вида.

Далее посредством прогоночных коэффициентов, последовательно вычисляем x_{n-1}, x_{n-2}, ..., x_1.

Алгоритм прогонки требует примерно $6N$ операций умножений и делений, что во много раз меньше, чем алгоритм Гаусса. Впервые метод прогонки был предложен Томасом во время работы над атомным проектом во время второй мировой войны. В настоящее время данный метод широко применяется при численном решении краевых задач, возникающих при решении дифференциальных уравнений в частных производных. На рис. 2.15 приведен пример по созданию модельной трехдиагональной матрицы, использованию функции ***Thomas****(a; b; c; d)* и сравнение результатов расчета СЛАУ методами прогонки и обратной матрицы. Видно, что для трехдиагональной матрицы размером 100×100 метод прогонки в 60 раз быстрее метода обратной матрицы.

Итерационные методы решения СЛАУ

Эффективными методами решения СЛАУ являются различные итерационные методы решения. К достоинствам итерационных методов относятся их применимость к плохо обусловленным системам и системам высоких порядков, а также простота программной реализации. Итерационные методы для начала вычисления требуют задания какого-либо начального приближения к искомому решению.

Для применения метода итераций исходную систему уравнений ***Ax = b*** необходимо привести к виду

$$x^{(k+1)} = A' x^{(k)} + b', \quad k = 0...n, \tag{9}$$

где A', b' – преобразованные матрицы, k – номер итерации.

Пример пошагового решения СЛАУ методом простой итерации приведен в левой части рис. 2.16. Сначала преобразуем матрицы A и b путем деления каждой строки на диагональный элемент a_{ii} и переносом слагаемых, не содержащих x_i, i – номер строки, в правую часть системы. Выбираем начальное приближение, в данном примере, в качестве начального приближения используется вектор b. После преобразования системы уравнений слева остается вектор неизвестных x, а справа прямоугольная

матрица с нулевыми диагональными элементами.

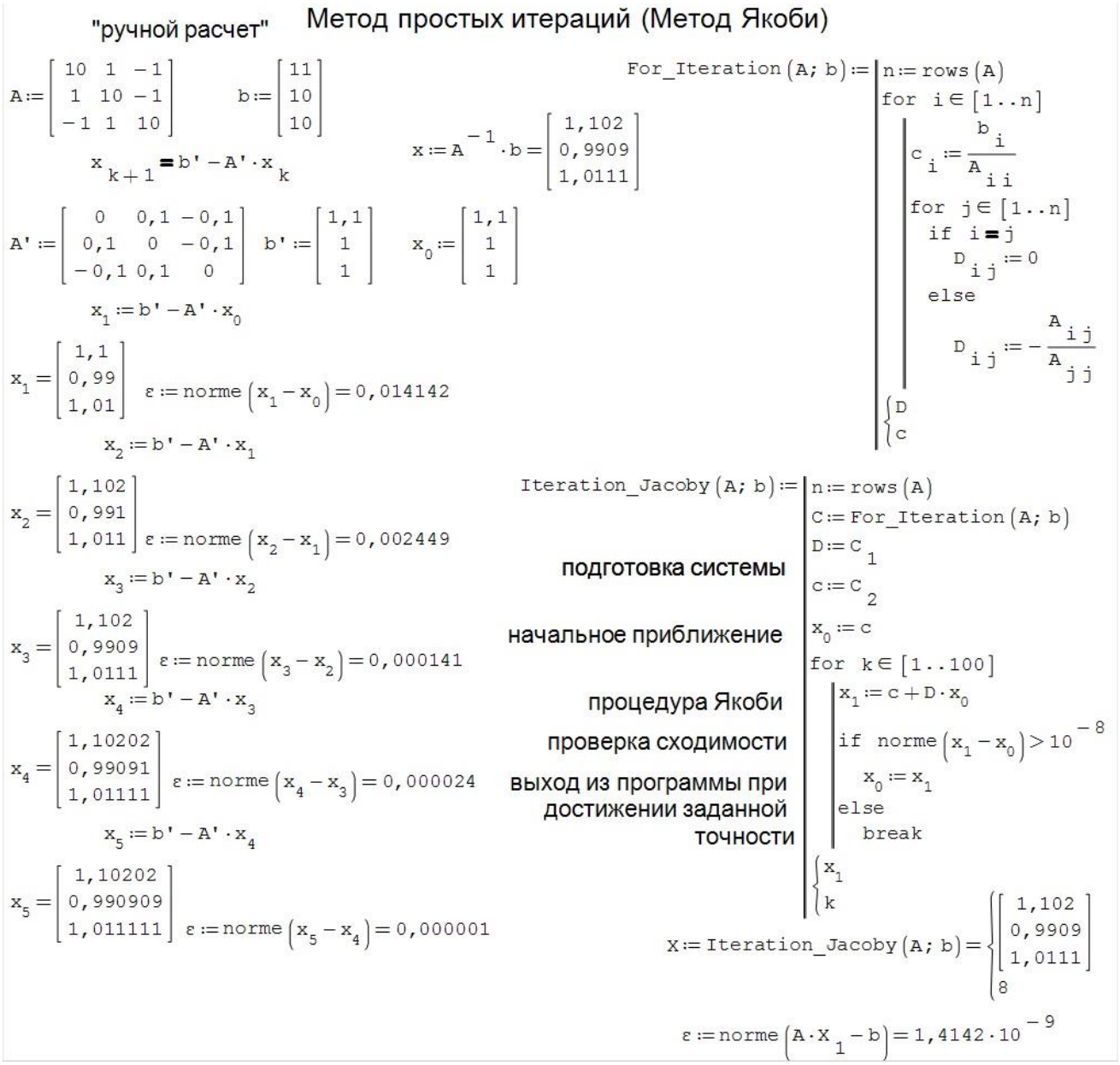


Рис. 2.16. Пример решения СЛАУ методом простой итерации (методом Якоби)

Итерационный процесс повторяем пять раз, каждый раз используя полученный результат в качестве приближения для последующего шага.

В качестве контроле сходимости итерационного процесса используется евклидова норма ***norme****(x(k+1) - x(k))*, т.е. «расстояние» между текущим и предыдущим решениями, численно равная вычислительной погрешности. Уже на пятом шаге погрешность составляет $\varepsilon = 10^{-6}$, это означает, что итерационный процесс сошелся и получено решение с абсолютной погрешностью не превышает ε.

Данный процесс легко автоматизировать. Справа на рис. 2.16 показано решение СЛАУ методом простых итераций с использованием пользовательских функций ***For_Iteration()*** и ***Iteration_Jacoby()***. Метод простых итераций был предложен Якоби в начале 18-го века и носит его имя.

Функций ***For_Iteration****(A; b)* подготавливает матрицу *A* и столбец свободных членов *b* к форме пригодной для начала итерационного процесса, путем построчного деления исходной системы уравнений на элементы стоящие на главной диагонали A_{ii}, и переносу остальных элементов в правую часть системы уравнений. Метод Якоби реализован функцией ***Iteration_Jacoby****(A; b)*, в которой задаются предельное число итераций *k* равное 100 и погрешность $\varepsilon = 10^{-8}$. В нижней части рисунка показан расчет корней СЛАУ с использованием метода простых итераций Якоби и проведен расчет невязки.

Следует заметить, что условия и скорость сходимости итерационного процесса существенно зависят от свойств матрицы *A* системы и от выбора начальных приближений. Необходимым условием сходимости итерационной процедуры является условие диагонального преобладания, которое предполагает, что элементы лежащие на главной диагонали матрицы *A* имеют значение по модулю большее, чем сумма других коэффициентов в данной строке:

$$|a_{ii}| > \sum_{i,j=1;i\neq j}^{n} |a_{ij}|, \qquad (10)$$

Если данное условие не выполняется, то итерационный процесс будет продолжаться бесконечно долго, для этого в итерационных процедурах необходимо ограничивать максимальное число итераций.

В ряде случаев, если диагональное преобладание не выполняется можно путем эквивалентных преобразований исходной матрица *A* (и вектора *b*, соответственно!) попытаться привести систему к требуемому виду.

Для уменьшения числа итераций Гаусс предложил идею, а Зейдель модифицировал метод простых итераций. Рассмотрим итерационный процесс Зейделя-Гаусса.

Перепишем формулу (9) для метода Якоби в следующем виде (алгебраическом и матричном):

$$x_i^{(k+1)} = \frac{b_i}{a_{ii}} - \sum_{j=1(j\neq i)}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)},$$
$$x^{(k+1)} = D^{-1}b - D^{-1}(E+F)x^{(k)}, \quad (10)$$

где D – диагональная матрица, состоящая из элементов a_{ii} исходной матрицы A, E и F – нижняя и верхняя части матрицы A с нулевыми диагональными элементами т.е. $A = E + D + F$.

Матричная форма метода Зейделя принимает вид

$$x^{(k+1)} = D^{-1}b - D^{-1}Ex^{(k+1)} + D^{-1}Fx^{(k)}, \quad (11)$$

т.е. приближенное значение корня, найденное на предыдущем шаге, начинает использоваться уже на текущем. За счет этого часто удается сократить общее число итераций.

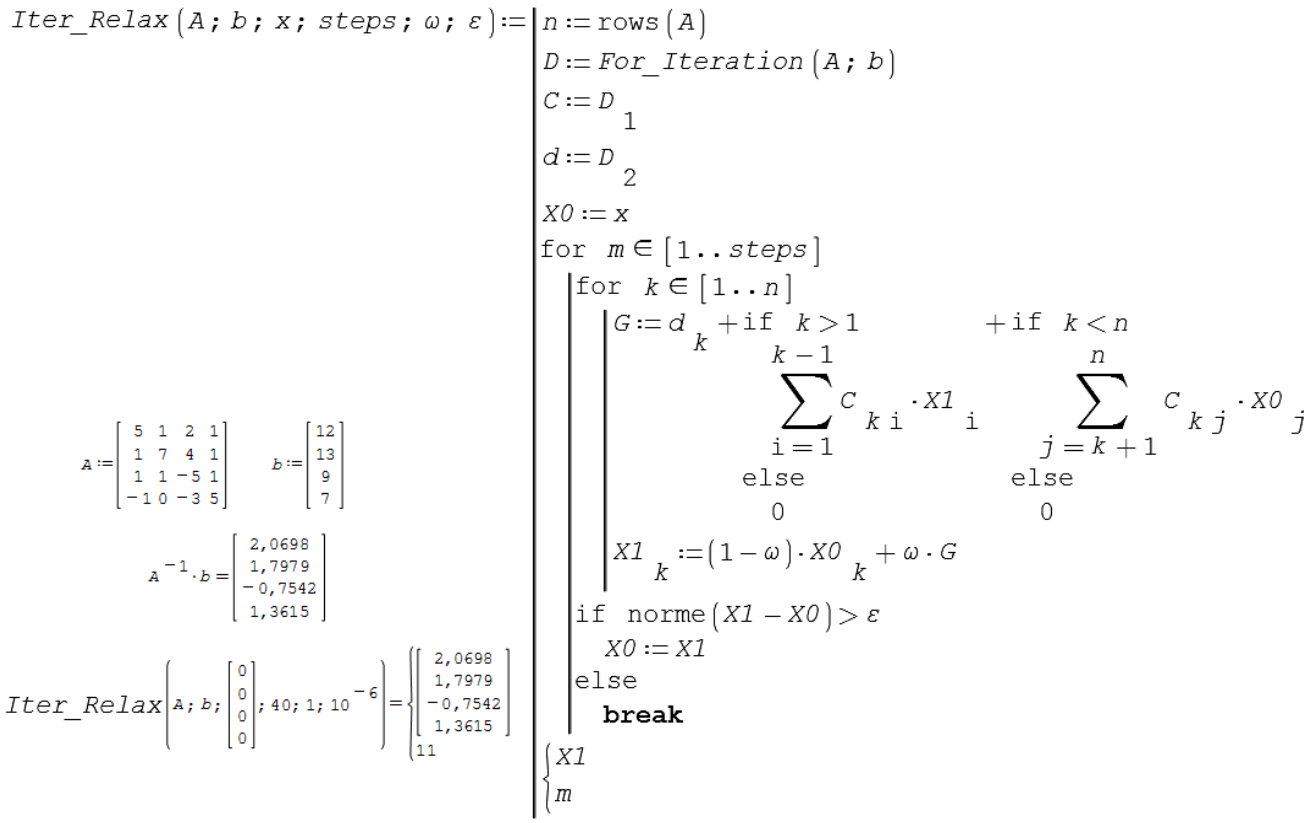


Рис. 2.18. Универсальная функция для изучения итерационных методов

Для изучения различных итерационных методов разработана универсальная функция ***Iter_Relax*** *(A; b; x; steps; ω; ε)* (рис. 2.18), которая реализует как методы Якоби и Зейделя, так и метод верхней релаксации. Входными параметрами ***Iter_Relax()***, является исходная матрица *A*, вектор свободных членов *b*, вектор начального приближения *x*, максималь-

ное количество итераций *steps*, параметр релаксации ω и заданную точность ε.

При значении параметра релаксации $\omega = 1$ программа реализует метод Зейделя (в соответствии с уравнением (11)). Если в данной программе в восьмой строке заменить $X1_i$ на $X0_i$ превращается в метод простой итерации (Якоби) (в соответствии с уравнением (10)).

2.3. Функции. Графики

Термин «функции» сегодня встречается не только в математика, поэтому, как и в случае с «вектором», необходимо учитывать контекст, в котором встречается данный термин. Например в фразе «Фирме требуются сотрудники с функциями и обязанностями менеджера, продавца, программиста», под понятием функция понимается возможность производить определенные действия. В программировании (пример, рис. 2.18) функция – это программный фрагмент, выполняющий действия по определенным правилам (алгоритмам).

В математике – функция – это соответствие между двумя множествами, при котором каждому элементу одного множества соответствует единственный элемент другого множества. Чаще всего функция одного аргумента обозначают как $y = f(x)$, где каждому числу x (аргументу функции) из из определенного множества значений D поставлено в соответствие число y из множества значений E. Множество значений D, которые может принимать x, называется областью определения функции, множество значений E, которые может принимать y, называется областью значений функции.

Функции могут быть элементарными, которые можно выразить через конечное число арифметических выражений, и неэлементарными (составными), если это не так. Пример,

$y = x^2 -$ элементарная функция, $y = |x| = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$ – неэлементарная,

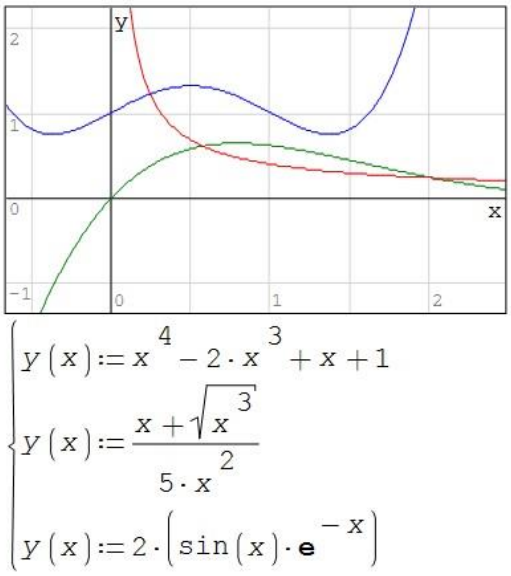
т.к. операции сравнения не являются арифметическими.

Элементарные функции делят на два класса: алгебраические (если ее значение можно получить из аргумента из действительных чисел с помощью операций сложения, вычитания, умножения, деления и возведения в

степень с рациональным показателем и трансцендентные (выражаемые через тригонометрические, экспоненциальные и др. подобные функции). Алгебраические функции делят на рациональные, если среди действий, которые производятся над независимой переменной, отсутствует извлечение корня и иррациональные (если присутствуют функции извлечения корней или степени с рациональными показателями). На рис. 2.19 приведены графики различных функций одного аргумента.

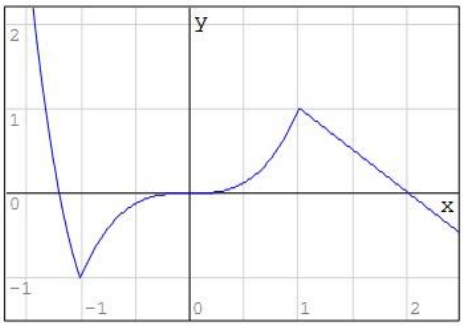
Функцию можно задать несколькими способами:

– аналитическим (с помощью формулы),
– графическим,
– табличным,
– описанием с помощью словесной формулировки.



$$\begin{cases} y(x) := x^4 - 2\cdot x^3 + x + 1 \\ y(x) := \dfrac{x + \sqrt{x^3}}{5\cdot x^2} \\ y(x) := 2\cdot\left(\sin(x)\cdot e^{-x}\right) \end{cases}$$

а)



$$y(x) := \begin{cases} x^4 - 2 & \text{if} \quad x < -1 \\ x^3 & \text{if} \quad -1 \le x \le 1 \\ -x + 2 & \text{otherwise} \end{cases}$$

б)

Рис. 2.19. Примеры функция: а) алгебраическая рациональная (синяя линия); алгебраическая иррациональная (красная); трансцендентная (зеленая); б) неэлементарная

Аналитический и графический способ задания функции демонстрирует рис. 2.19, табличное задание функции было показано ранее на рис. 1.7.

Построение графика функции позволяет наглядно представить функцию и наблюдать особенности ее поведения. Программа *SMath Studio* содержит инструменты, которые позволяют достаточно просто строить графики функций и визуализировать результаты расчетов.

Для построения графиков заданных в явном виде вполне подходит

стандартный инструмент «двумерный 2D» из «Вставка»/ «График» на верхней панели инструментов. Но часто приходится работать с функциями, которые заданы в неявном, параметрическом виде или строить графики в полярных координатах. Рассмотрим построение графиков в этих случаях.

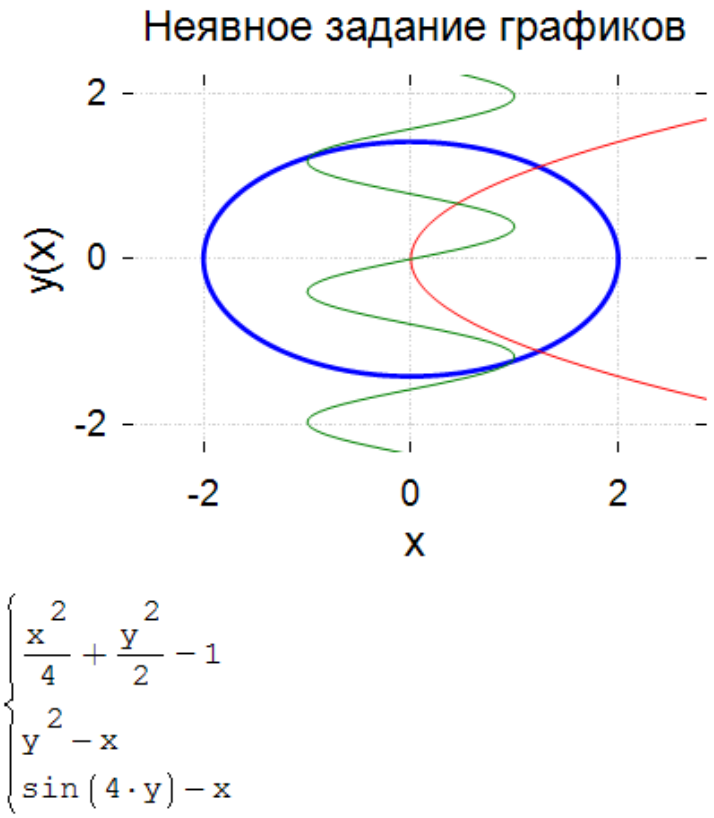


Рис. 2.20. Построение графиков заданных неявным образом

Для построения графика функции заданной в неявном виде проще всего воспользоваться дополнением «X-Y График», для этого в верхнем меню «Помощь/Примеры/Дополнения» установить «X-Y Plot Region», после этого в пункте меню «Вставка/График» появится новый пункт «X-Y График». Построение графиков функций и плоских кривых, заданных неявно, приведено на рис. 2.20.

Достоинством «X-Y График» является возможность произвольно менять цвет и стиль линий, создавать подписи по осям координат, создавать названия графика, независимо менять масштаб и сетку по координатным осям, т.е. практически все возможности, которыми обладают специализированные профессиональные программы. А возможность интерактивного масштабирования с помощью мыши облегчает работу с графикой (как и в встроенном инструменте «Двумер- ный 2D» график). Здесь можно от-

метить, что существует альтернативное дополнения к программе *SMath Studio* – «ZedGraph Region», в котором реализованы все возможности работы с «научной графикой», но в нем отсутствует возможность интерактивного управления графиками, к тому же данное дополнение намного сложнее.

На рис. 2.21 приведены примеры создания графиков плоских кривых заданных в параметрическом виде. Построим кривую второго порядка – окружность или эллипс, с помощью встроенной функции «Двумерный 2D» график. Можно увидеть, что замкнутые кривые второго порядка разбиваются на две ветви (верхнюю и нижнюю), при их задании в явном виде, а так как каждая кривая в штатном графопостроителе имеет свой цвет, результат выглядит не очень выразительным. Рядом приведен пример задания тех же кривых, заданных в параметрическом виде. Для этого создана пользовательская функция ***Par***$(\Delta\theta;\ N)$, где $\Delta\theta$ – шаг, N – число точек, которая формирует матрицу (таблицу), где первый столбец содержит абсциссы точек на графике, а второй их ординаты. По умолчанию ближайшие точки соединяются линиями.

Рис. 2.21. Построение графиков заданных в параметрическом виде

В нижней части рисунка 2.21 приведены еще две параметрически заданные кривые – кардиоида и астроида. Для выбора шага и числа точек достаточно просто представить, что полный оборот окружности это 360^0, следовательно, если шаг выбрать равным $2^0 = \pi/2 = 0.035$, то число точек будет равно 180.

Рис. 2.22. Построение графиков в полярных координатах

Далее, научимся строить кривые в полярных координатах. Полярная система координат – двумерная криволинейная система координат, в которой каждая точка на плоскости определяется двумя числами – полярным углом θ и полярным радиусом (азимутом) r. Полярная система координат особенно полезна в случаях, когда отношения между точками проще выразить в виде радиусов и углов, в то время как в более распространённой декартовой системе координат, такие отношения можно установить только с использованием тригонометрических уравнений.

На рис. 2.22 приведены кривые, построенные в полярных координатах с помощью пользовательской функции ***PolarG****(S; F; N)*, *S* – где начальный угол, *F* – конечный угол, *N* – число точек. На рис. 2.22 показаны формулы для перехода от декартовых координат к полярным. Обратный переход осуществляется по теореме Пифагора $r^2 = x^2 + y^2$ и $\mathrm{tg}(\theta) = y/x$.

Далее, на рис. 2.22 приведено уравнение эллипса, гиперболы и параболы которые задаются в полярных координатах одним уравнением, в котором меняются только параметры. Параметр $p = a - \varepsilon \cdot c$, $\varepsilon = c/a$ – эксцентриситет (для эллипса $\varepsilon \le 1$, для параболы $\varepsilon = 1$, для гиперболы $\varepsilon > 1$) $c = \sqrt{a^2 - b^2}$ – координата фокуса, *a* и *b* – большая и малые полуоси (для эллипса). Эллипс фигура замкнутая, т.к. $1 - \varepsilon \cdot \cos(\theta)$, при любых углах отличен от нуля (при $\varepsilon = 1$, эллипс превращается в окружность). Для гиперболы и параболы знаменатель обращается в ноль, и кривые уходят на бесконечность. На рис. 2.22 показаны примеры кривые эллипса, параболы и гиперболы, заданные в полярных координатах. В нижней части рисунка 2.22, приведены спираль Архимеда и кардиоида.

Отличительной чертой программы *SMath Studio*, является возможность создания анимированных графиков. Данная возможность была реализована в ранних версиях *MathCad* (до 15 версии включительно), но в современных версиях *MathCad Prime* больше не поддерживается.

Поэтому *SMath Studio* единственная простая, доступная и бесплатная система, в которой возможно создание анимации. Анимированные изображения позволяют наглядно рассматривать различные динамические процессы, тем самым облегчая их понимание.

Возможности по созданию анимированных графиков проиллюстрируем на примере графика циклоиды. Циклоида это трансцендентная кривая, не имеющая аналитического выражения в явном виде, возникающая как траектория точки расположенной на ободе колеса, равномерно катящегося без проскальзывания по горизонтальной поверхности.

Данная кривая изучалась многими математиками, на образе которой оттачивали свое мастерство Декарт, Ферма, Бернули, Гюйгенс, Лейбниц, Ньютон и многие другие великие ученые начиная с середины 16 века и до наших дней. Но компьютера в те времена не было, поэтому на изучение такой сложной кривой уходили годы. Использование пакета *SMath*

Studio позволяет изучить форму и свойства циклоиды и родственных кривых буквально за несколько минут, заодно познакомится с возможностями по созданию анимированных изображений (рис. 2.23).

Зададимся окружностью с радиусом равным 2 единицам. Шаг анимации примем равным 3^0, число точек примем равным 360, что соответствует трем периодам циклоиды. Параметр · (в дальнейшем понадобится для модификации циклоиды) положим равным 1. Определим уравнение движения центра колеса. Т.к. колесо равномерно катится по горизонтальной прямой, ордината колеса постоянна и равна R, а абсцисса линейно смещается вдоль оси *OX* по закону $R\Delta\theta t$. Также нам понадобится радиус колеса, отрезок соединяющий центр окружности с точкой на ободе, в начале координат, в момент времени $t = 0$, радиус направлен вертикально вниз. Далее запишем пользовательскую функцию ***DiskLine****(t)*, которая описывает движение диска с течением времени (задается центр диска с координатами *xD* и *yD*). Воспользуемся ранее введенной функцией ***Affine****()*, для построения радиуса диска в каждый момент времени. Далее формируем черную (черный цвет, если место для выбора цвета " " пустое) точку (размером в 10 пикселей), соответствующую центру диска. Формируем изображение диска с теми же координатами его центра. Диск (окружность) в данном примере формируется букой "O" зеленого цвета. К сожалению, диаметр в пикселях приходится подбирать экспериментально, в примере, радиус диска 53*R*, чтобы диск касался по горизонтальной линии, и красная точка лежала на его радиусе. Далее формируем красную точку размером 15 пикселей, которая лежит на ободе колеса и описывает циклоиду при движении. Уравнение циклоиды в параметрическом виде приводится во многих учебниках по математике и легко находится в сети Internet:

$$\begin{cases} x(t) = R(\Delta\vartheta t - \lambda\sin(t)), \\ y(t) = R(1 - \lambda\cos(t)). \end{cases} \qquad (12)$$

Затем формируем график циклоиды, заданной по точкам с использование функции $L(\lambda)$.

Создание анимации в *SMath Studio* заключается в последовательном формировании определенного числа кадров, в нашем примере, формируется строкой *step*:=[1..N]. В пакете MathCad для формирования анимированного изображения использовалась фиксированная переменная Frame,

в *SMath Studio* переменная анимации может задаваться произвольной переменной.

На графике (рис. 2.23) отображаются 5 объектов. Для того чтобы добавить объекты на 2D графике используется конструкция – система, которую можно ввести с правой панели *SMath Studio* (последний значок во вкладке «функции»). Для изменения количества элементов в системе можно использовать мышку (выделяя систему и изменяя ее размер двигая мышью за нижний правый прямоугольник) или используя клавишу «;» в конце (или начале) любой строки системы.

Рис. 2.23. Построение циклоиды в *SMath Studio*

Цвета на графиках *SMath Studio* (по умолчанию) циклически повторяются в следующем порядке – *blue*, *red*, *green*, *magenta*, *orange* and *brown*. Для пропуска очень светлых цветов *magenta* и *orange* в системе использовались пустые строки "". Ноль который стоит в предпоследней строчке системы вывода означает прямую линию $y(x) = 0$.

Далее, запуская расчет, мы увидим картинку (рис. 2.24) с выводом сообщения об ошибке, что переменная *t* не определена.

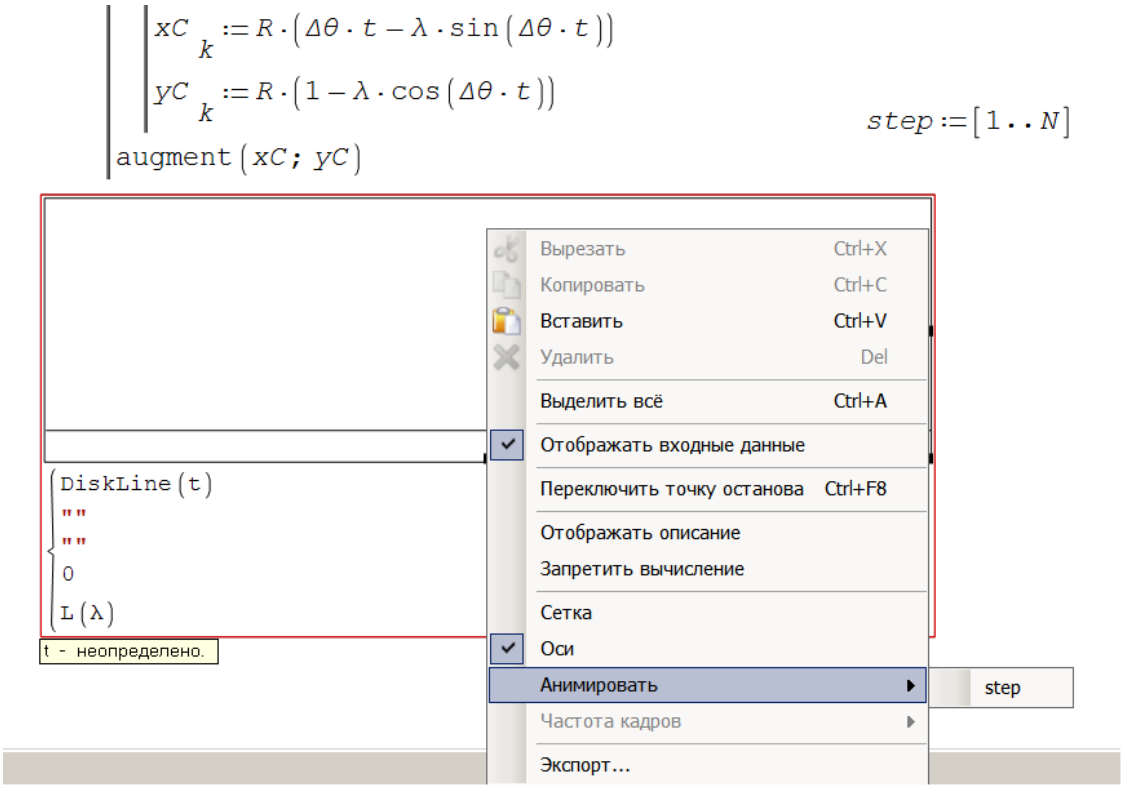


Рис. 2.24. Сообщение об ошибке при запуске анимации

Теперь необходимо войти в свойства 2D графика, с помощью правой клавиши мыши, и связать неопределенную переменную с переменной анимации, в нашем случае, с переменной *step*. Повторяя расчет, получаем анимированную картинку катящегося колеса по горизонтальной поверхности, в которой красная точка, лежащая на ободе колеса, описывает циклоиду (рис. 2.23). Из рис. 2.24 видно, что данную анимацию можно экспортировать в виде *gif*-файла, который можно использовать в различных документах, например, в презентациях.

В листинге программы (рис. 2.23), был зарезервирован параметр λ. Если изменять λ можно получить или укороченную циклоиду (когда точка лежит на расстоянии $r < R$ ($\lambda < 1$) меньше радиуса диска) или удли-

нённую циклоиду, если $r > R$ ($\lambda > 1$).

По аналогии с построением циклоиды, когда диск катится по плоской поверхности, возможно создание других кривых – эпициклоиды и гипоциклоиды (названия кривых предложил Галилей от эпи-цикл-оид и гипоцикл-оид – кривой подобной катящейся над(под) кругом). Эпициклоида соответствует кривой, которую рисует точка, лежащая на ободе диска радиуса *r*, который сам катится по внешней поверхности другого неподвижного диска (радиусом *R*). В случае, если диск *r* катится по внутренней поверхности цилиндра *R*, точка на поверхности диска r будет описывать гипоциклоиду. Пример гипоциклоиды с $R = 4R$ (астроиды), был приведен на рис. 2.21.

2.4. Приведение кривых второго порядка к каноническому виду

В данном примере, рассмотрим использование программы *SMath Studio* для решения задач по приведению уравнений кривых второго порядка к каноническому виду. При решении такого рода задач очень помогают графические возможности программы. Для построения графика в неявном виде использовалось дополнение X-Y Plot Region. Из рис. 2.9 видно, что кривая второго порядка, в данном случае это эллипс, расположена несимметрично относительно начала координат. Визуально приведение к каноническому виду означает переход к новой системе декартовых координат, в которой кривая будет иметь симметричный вид относительно новых координатных осей. Из рисунка видно, что нам потребуется сместить начало координат и осуществить поворот координатных осей на некоторый угол, чтобы придать кривой канонический вид. Существует несколько способов решения данной задачи.

Самым простым формальным способом решения такого типа задач является метод инвариантов, предложенный Якоби, но данный метод не дает ответа на вопросы о повороте и смещении начала координатных осей. Рассмотрим способ, который раскрывает геометрический смысл приведения кривых к каноническому виду.

Уравнение кривой второго порядка запишем в произвольном и «каноническом» виде:

$$A\cdot x^2 + 2B\cdot x\cdot y + C\cdot y^2 + 2D\cdot x + 2E\cdot y + F = 0; \quad A'\cdot x^2 + C'\cdot y^2 + F' = 0\,. \quad (13)$$

Видно, что приведение заклю- чается в эквивалентном преобразо-

вании исходного уравнения к «каноническому», в котором коэффициенты B', D' и E' будут равны нулю.

Сначала избавимся от коэффициентов D' и E' путем смещения начала отсчета канонической системы координат. Для этого решим систему линейных алгебраических уравнений приведенной на рис. 2.25.

Приведение кривой второго порядка к каноническому виду

$$A \cdot x^2 + 2 \cdot B \cdot x \cdot y + C \cdot y^2 + 2 \cdot D \cdot x + 2 \cdot E \cdot y + F = 0$$

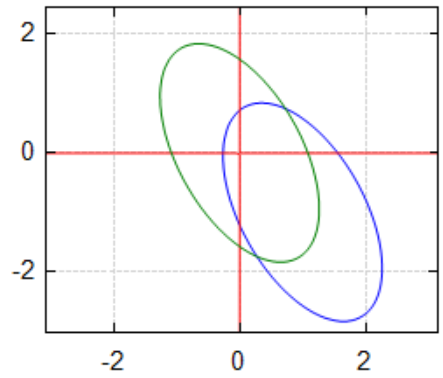
$$17 \cdot x^2 + 12 \cdot x \cdot y + 8 \cdot y^2 - 22 \cdot x + 4 \cdot y - 7 = 0$$

$$\begin{cases} A \cdot x + B \cdot y + D = 0 \\ B \cdot x + C \cdot y + E = 0 \end{cases}$$

Система уравнений для нахождения начала "новой" системы координат

$$\text{roots}\left(\begin{bmatrix} 17 \cdot x + 6 \cdot y - 11 \\ 6 \cdot x + 8 \cdot y + 2 \end{bmatrix}; \begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Координаты начала "новой" системы координат



При переходе к "новой" системе координат, центр эллипса совпал с началом отсчета

$$F1(x; y) := 4 \cdot \left(-5 + 3 \cdot x \cdot y + 2 \cdot y^2\right) + 17 \cdot x^2$$

аналитическая формула "смещенного" эллипса в которой коэффициентв D и E равны нулю

$$\begin{cases} 17 \cdot x^2 + 12 \cdot x \cdot y + 8 \cdot y^2 - 22 \cdot x + 4 \cdot y - 7 \\ -\frac{x}{y} \\ F1(x; y) \end{cases}$$

$$\text{tg}(2 \cdot \alpha) = \frac{2 \cdot B}{A - C}$$

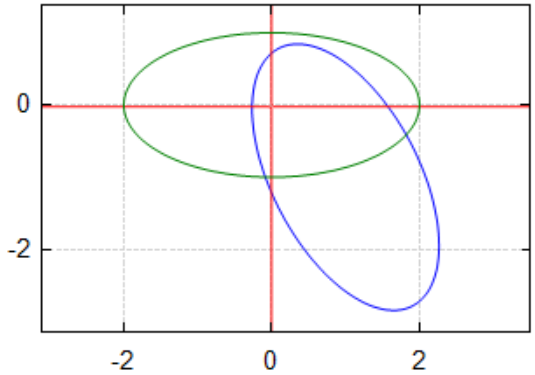
угол поворота "новой" канонической системы координат, если A = C, то α = ±π/4

$$\alpha := \frac{1}{2} \cdot \text{arctg}\left(\frac{2 \cdot 6}{17 - 8}\right) = 26,5651\ ^\circ$$

Ортогональное преобразование минора Δ для приведения его к диагональному виду (B≡0)

$$\Delta = \left(U^T \cdot \Delta \cdot U\right) \qquad U := \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

матрица поворота



$$U^T \cdot \begin{bmatrix} 17 & 6 \\ 6 & 8 \end{bmatrix} \cdot U = \begin{bmatrix} 20 & 4 \cdot 10^{-15} \\ 4 \cdot 10^{-15} & 5 \end{bmatrix}$$

Рис. 2.25. Приведения кривой второго порядка к каноническому виду

С использованием встроенной функции *roots()* находим начало координат «новой» системы. Получаем координаты (1; -1), производим замену переменных $x' = x + 1$ и $y' = y - 1$. Получаем новую формулу для кривой второго порядка *F1*(x; y) в которой коэффициенты *D*' и *E*' равны нулю. Рисуем график кривой и убеждаемся, что новое начало координат лежит в центре эллипса. Сразу замечаем, что свободный коэффициент *F* ' = – 20.

Далее, необходимо найти угол поворота, чтобы эллипс располагался горизонтально. В данном примере можно воспользоваться формулой $\mathrm{tg}(2\alpha) = 2B/(A - C)$. Проводим ортогональное вращение кривой на данный угол, для этого строим матрицу поворота и выполняем преобразование $U^T \cdot \begin{pmatrix} A & B \\ B & C \end{pmatrix} \cdot U$. В результате получаем новые коэффициенты *A*' и *C*', при этом коэффициент *B*' становится равным нулю. Получившееся уравнение $20x^2 + 5y^2 = 20$ делим на 20, следовательно, $x^2 + y^2/4 = 1$. Стоим график данной кривой. Получаем эллипс расположенный вертикально. Это нас не устраивает, по определению фокусы эллипса должны лежать на оси *OX*. Разворачиваем эллипс на 90^0, для этого меняем местами оси координат, переходя к уравнению $y^2 + x^2/4 = 1$. Выводим данную кривую на график и получаем каноническое уравнение эллипса.

2.5. Дифференцирование функции. Примеры использования

Производная функции в точке это основное понятие дифференциального исчисления, которая характеризует скорость изменения функции в указанной точке. Производная широко используется при решении целого ряда задач математики, физики и других наук, при изучении скорости протекания различных процессов. Процесс вычисления производной называется дифференцированием функции.

Рассмотрим различные случаи практического применения производных. Программа *SMath Studio* позволяет определять производные как аналитически, так и численно, а графические возможность программы позволяют визуализировать многие аспекты практического использования. На рис. 2.26 приведены примеры использования производной от функции при решении различных математических задач. Достаточно часто приходится строить прямые проходящие по касательной или

образующие нормаль в заданной точке к некоторой функции. Для этого необходимо вспомнить геометрический смысл производной как тангенс угла наклона к функции в данной точке. Для расчета производной исходную функции можно продифференцировать, используя встроенные возможности *SMath Studio*, вызывая операцию дифференцирования, вкладка «Функции» на боковой панели (или набирая на клавиатуре *diff* «TAB»). Запишем уравнение касательной в виде прямой $g(x)$, в котором необходимо определить два коэффициента a и b. Коэффициент a это значение производной в точке x_0 $a = y(x_0)$. Точка находится из условия (является решением уравнения) $g(x_0) = f(x_0)$ или $b = f(x_0) - f'(x_0) \cdot x$.

Уравнение нормали имеет угол 90^0 с уравнением касательной. Учитывая свойства тангенса, записываем уравнение нормали к кривой в точке x_0. Нормаль $g(x)$ и касательную $g1(x)$ откладываем на графике вместе с черным кружочком с координатами $(x_0; f(x_0))$. Первая кривая – график функции $f(x)$ имеет синий цвет, вторая – касательная – красный, нормаль к $f(x)$ имеет зеленый цвет.

Далее, изучаем особенности функции $f(x)$, находим точки минимума, максимума и точку перегиба, которым соответствуют экстремумы $f(x)$, когда первая производная $f'(x_0)$. Для этого решаем нелинейные уравнения с использованием встроенной функции ***solve()*** с четырьмя аргументами. Находим абсциссу X_{min}, соответствующую минимуму функции $f(x)$. Визуально, по графику определяем примерное положение минимума, записываем ***solve***$(f'(x); x; 0; 1)$. Рассчитываем положение минимума $f(x)$. Аналогичным способом определяем положение максимума $f(x)$. Положение минимума и максимума откладываем на графике в виде красных точек с координатами $(X_{min}; f(X_{min}))$ и $(X_{max}; f(X_{max}))$.

Определим положение точки перегиба, для этого решим уравнение $\frac{d^2 f(x)}{dx^2} = 0$, для определения обращения в ноль второй производной от функции $f(x)$. Решая нелинейное уравнение, получаем координату точки перегиба по оси OX, нанесем эту точку на график в виде красного креста.

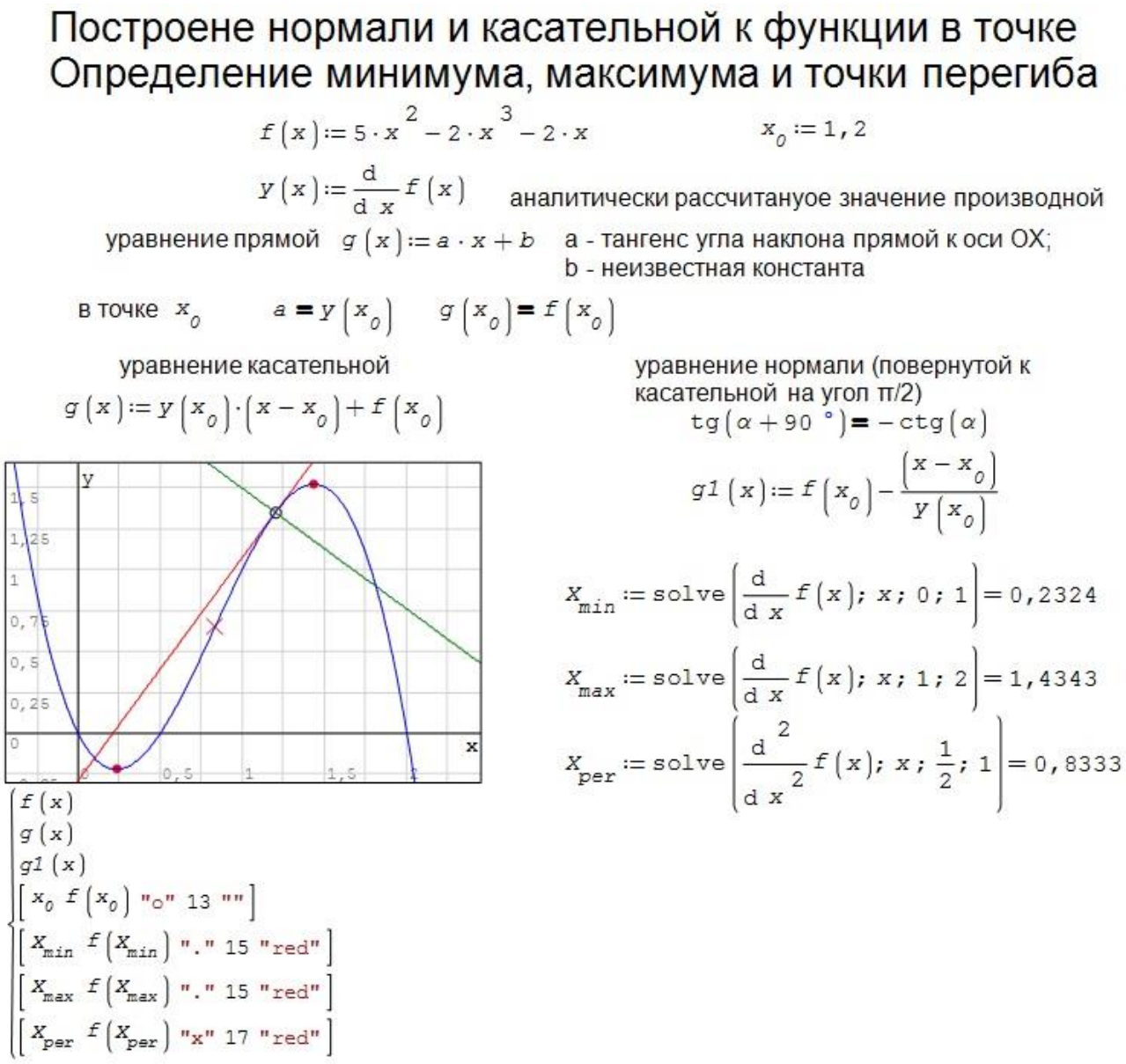


Рис. 2.26. Пример построения нормали и касательной, нахождение минимума, максимума и точки перегиба функции *f(x)* в точке x_0.

2.6. Определенный интеграл. Численное интегрирование

Еще одним важнейшим понятием в математике является определенный интеграл.

Основные свойства определенного интеграла:

$$1.\ \int_a^a f(x)dx = 0; \qquad 5.\ \int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx,\ a \le c \le b;$$

$$2.\ \int_a^b f(x)dx = -\int_b^a f(x)dx; \qquad 6.\ \left|\int_a^b f(x)dx\right| \le \int_a^b |f(x)|dx;$$

$$3.\ \int_a^b k \cdot f(x)dx = k\int_a^b f(x)dx; \qquad 7.\ \int_a^b f(x)dx = f(\mu)(b-a),\ a \le \mu \le b; \qquad (14)$$

$$4.\ \int_a^b (f(x) \pm g(x))dx = \int_a^b f(x)dx \pm \int_a^b g(x)dx;$$

Для расчета определенного интеграла часто используют формулу Ньютона-Лейбница, предполагая, что определенный интеграл численно равен площади криволинейной трапеции на интервале [*a*, *b*] (рис. 2.27).

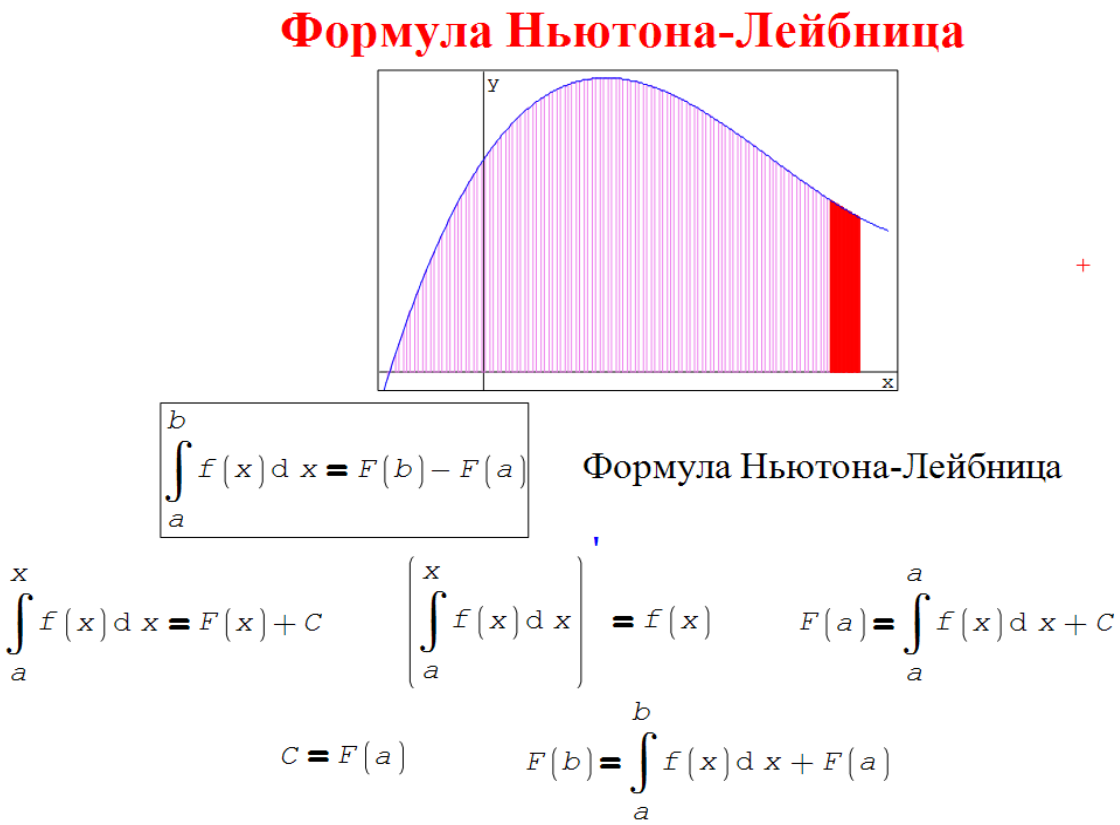


Рис. 2.27. Вывод формулы Ньютона-Лейбница для вычисления определенного интеграла

Определяем первообразную как интеграл с переменным верхнем пределом от функции *f(x)*, проверяем, что производная от первообразной равна самой функции, рассчитываем значение первообразной в точке *a*, т.к. определенный интеграл на интервале нулевой длинны, равен нулю,

получаем значение константы интегрирования. И наконец, вычисляя значение первообразной в точке *b*, получаем формулу Ньютона-Лейбница.

Однако данной формулой воспользоваться удается далеко не всегда, а только в случаях, когда первообразная функции известна в явном виде. Если интегрируемая функция задана в виде табличных данных, как это часто бывает при экспериментальном определении функции, формула Ньютона-Лейбница не применима. В этом случае необходимо воспользоваться некоторой квадратурной формулой (для определения определенного интеграла функции одной переменной) или кубатурной формулой, если функция зависит от большего числа переменных.

Наша задача найти квадратурные формулы, которые имеют простой вид и позволяют рассчитать определенный интеграл с заданной погрешностью с минимальными вычислительными затратами.

Метод прямоугольников (левых, правых, средних)

Самым простым численным методом расчета определенного интеграла является метод прямоугольников. На рис. 2.28 приведены результаты расчета определенного интеграла методами прямоугольников. Показана подынтегральная функция, аналитический расчет интеграла встроенными в *SMath Studio* средствами. Заданы пределы интегрирования, число интервалов интегрирования $N = 8$, рассчитан шаг интегрирования $h = (b - a)/N$. Далее проведена дискретизация подынтегральной функции для левых, правых и средних (центральных) прямоугольников. Высота прямоугольников вычисляется по левому, правому или среднему значению функции на каждом интервале. Значение определенного интеграла рассчитывается как сумма площадей всех прямоугольников.

Из нижней части рисунка видно, что при разбиении области интегрирования на ограниченное число интервалов (в нашем случае на 8), погрешность вычисления составляет порядка 4% для левого и правого прямоугольников, а для средних прямоугольников погрешность намного меньше. Для объяснения этого факта необходимо научиться рассчитывать погрешности численных методов интегрирования.

Различают априорную погрешность, которая может быть найдена из анализа задачи и используемых методов до начала вычислений, и апостериорная погрешность, которая рассчитывается в ходе решения задачи.

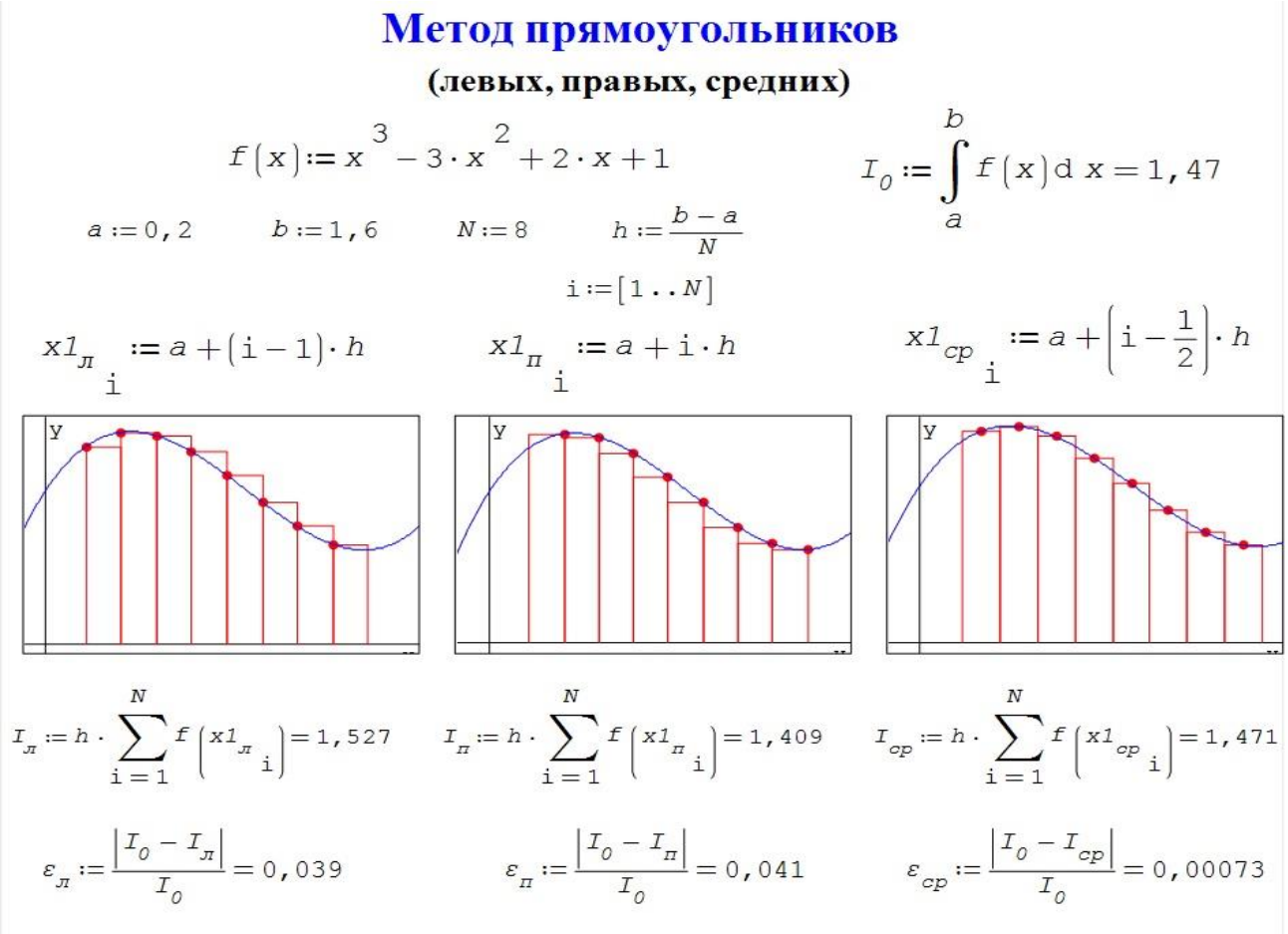


Рис. 2.28. Расчет определенного интеграла методом прямоугольников

Рассмотрим процесс нахождения априорной погрешности для методов левого, правого и среднего прямоугольников. Обозначим через · погрешность (абсолютную) численного метода интегрирования на всем интервале *[a; b]*.

$$\varepsilon = \int_a^b f(x)dx - \sum_{k=1}^{n} \int_0^h f(x_k)dx; \quad \varepsilon_k = \int_0^h f(x_k)dx - S, \tag{15}$$

где ε – абсолютная погрешность интегрирования, *n* – число шагов, *h* – шаг интегрирования, *S* – квадратурная формула, ε_k – погрешность на *k* интервале. Расчет проводится по методу прямоугольников (левом, среднем и правом), соответственно, $S_л = h \cdot f(0)$, $S_п = h \cdot f(h)$, $S_{ср} = h \cdot f(h/2)$. Для этого разложим подынтегральную функцию в ряд Тейлора.

$$f(x)=f(a)+\sum_{k=1}^{\infty}\frac{f^{(k)}(a)}{k!}(x-a)^k=f(a)+\sum_{k=1}^{n}\frac{f^{(k)}(a)}{k!}(x-a)^k+O((x-a)^n);$$
$$f(x)=f(a)+\frac{f'(x)}{1!}(x-a)+\frac{f''(x)}{2!}(x-a)^2+\frac{f'''(x)}{3!}(x-a)^3+\dots;$$
$$f(0)=f(0)+0\cdot f'(0);\quad f(h)=f(0)+h\cdot f'(0);\quad f(\frac{h}{2})=f(0)+\frac{h}{2}\cdot f'(0)+\left(\frac{h}{2}\right)^2\cdot\frac{f''(0)}{2};$$
$$f(x)=f(0)+x\cdot f'(0)+x^2\cdot\frac{f''(0)}{2},\qquad (16)$$

Для метода левых и правых прямоугольников в этом ряде достаточно ограничится только первыми двумя слагаемыми, т.к. последующие слагаемые будут намного меньше по абсолютному значению. Для метода средних прямоугольников количество слагаемых должно быть на одно больше. Отброшенные нами слагаемые в математике называются остаточными членами. Подставим результаты разложения в ряд Тейлора, как самой функции, так и ее значений в точках 0 и *h* в формулу (15). Аналитически рассчитываем интеграл, приводим подобные члены и получаем значения погрешности для метода левого, правого и среднего прямоугольников.

$$\varepsilon_{\text{л}}=\int_0^h(f(0)+x\cdot f'(0))dx-h\cdot(f(0)+0\cdot f'(0))=h\cdot f(0)+\frac{h^2}{2}f'(0)-h\cdot f(0)=\frac{h^2}{2}f'(0),$$
$$\varepsilon_{\text{п}}=\int_0^h(f(0)+x\cdot f'(0))dx-h(f(0)+h\cdot f'(0))=h\cdot f(0)+\frac{h^2}{2}f'(0)-h\cdot f(0)-h\cdot f'(0)=-\frac{h^2}{2}f'(0),$$
$$\varepsilon_{\text{ср}}=\int_{-h/2}^{h/2}(f(0)+x\cdot f'(0)+x^2f''(0))dx-h\cdot(f(0)+\frac{h}{2}\cdot f'(0)+\left(\frac{h}{2}\right)^2\cdot\frac{f''(0)}{2})=$$
$$=h\cdot f(0)+\frac{h^3}{12}\frac{f''(0)}{2}-h\cdot f(0)=\frac{h^3}{24}f''(0)\qquad (17)$$

Погрешность метода левых и правых прямоугольников, имеет второй порядок малости (степень *h*) и нулевой порядок алгебраической точности (*n* – 1 – порядок производной). Т.е. метод левых (правых) прямоугольников квадратично стремится к истинному значению при уменьшении шага *h*. Погрешность метода средних прямоугольников существенно выше. Для него порядок малости равен трем, а алгебраическая точность имеет первый порядок, т.е. формула абсолютно точна для всех линейных подынтегральных функций. Это происходит из-за того, что интервал интегрирования для метода средних прямоугольников симметричен, следовательно, интегралы от нечетных степеней равны нулю и нам пришлось в ряде Фурье оставить дополнительное слагаемое.

Метод трапеций и метод Симпсона

Для повышения точности численного интегрирования можно попытаться заменить подынтегральное выражение, на функцию, которую легко проинтегрировать аналитически.

На каждом интервале мы имеем трапецию, которая получается путем аппроксимации подынтегральной функции линейной, путем соединения начала и конца каждого интервала отрезком. На рис. 2.29. проведен расчет ранее проинтегрированной функции (на том же интервале и с тем же шагом) методом трапеций.

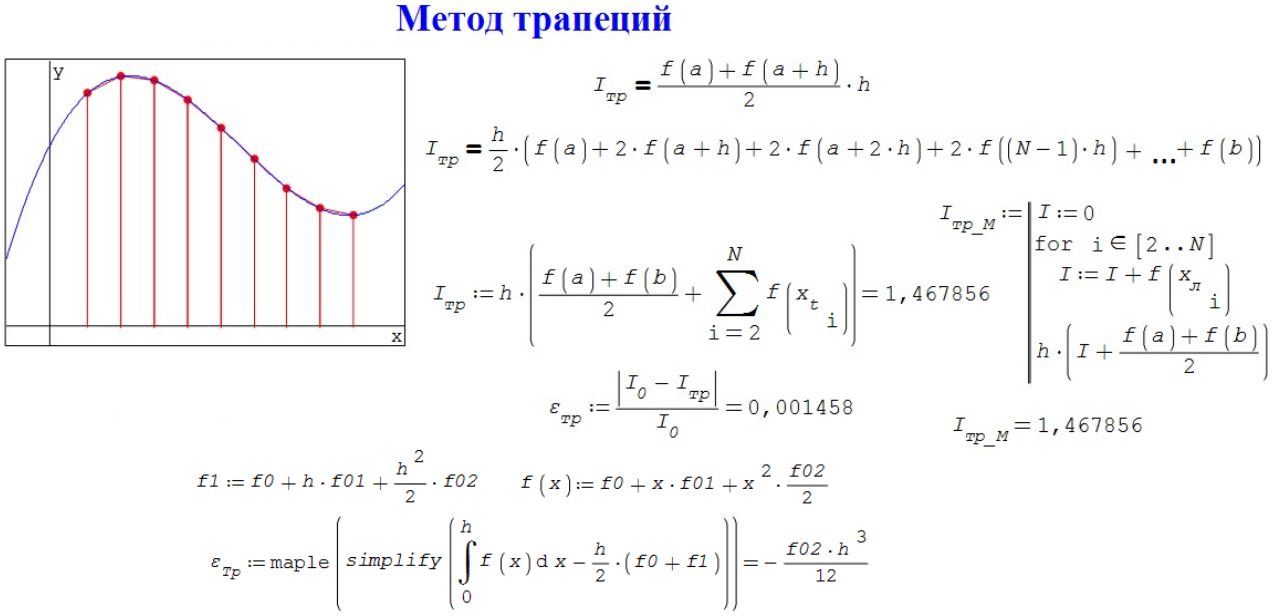


Рис. 2.29. Реализация метода трапеций в пакете SMath Studio

Видно, что линейная аппроксимация позволяет получить лучшее согласие с исходной функцией, чем метод правых или левых прямоугольников, но уступает по точности методу центральных прямоугольников.

Площадь одной трапеции равна произведению шага интегрирования на среднюю высоту трапеции. Если данную формулу расписать для всей суммы, можно получить расширенную формулу метода трапеций. На рис. 2.29 сокращенная и расширенная формулы приведены рядом с графиком. Видно, что первое и последнее слагаемое суммы имеют коэффициент ½, а остальные коэффициенты перед функциями равны единице. На рисунке показана реализация метода трапеций с использованием встроенной функции суммирования и с использованием программного блока.

Также видно, что относительная погрешность метода трапеций при шаге $N = 8$ лучше, чем для метода левых и правых прямоугольников, но все еще хуже, чем для метода средних прямоугольников. Внизу рисунка приведен расчет априорной погрешности с использованием аналитических возможностей дополнения Maple.

Дальнейшим развитием методов численного интегрирования является метод Симпсона, в котором подынтегральная функция аппроксимируется линией второго порядка – параболой.

На рис. 2.30 показано сравнение параболической и линейной аппроксимации рассматриваемой функции $f(x) = x^3 - 3x^2 + 2x + 1$.

Процесс линейной аппроксимации достаточно простой, достаточно соединить отрезками соседние точки на графике. Таким образом, аппроксимируется подынтегральная функция в методе трапеций. В методе Симпсона используется параболическая аппроксимация. На рис. 2.30 дано детальное описание данного процесса. Для рассматриваемой функции выведены на экран значения аргумента и самой функции для восьми интервалов, на которые мы разбили процесс интегрирования. Проведем параболу через три первые точки. Для задания параболы необходимо рассчитать коэффициенты A, B и C. Составим СЛАУ из трех уравнений, для трех первых точек, абсциссы и ординаты которых заданы на рис. Решение СЛАУ проводится с помощью встроенной функции ***roots***(). Присваиваем полученные параметры коэффициентам A, B и C и выводим информацию в виде графика. На нем изображены четыре объекта: подынтегральная функция $f(x)$, парабола $y(x)$ (проходящая через три первые точки), линейная аппроксимация исходной функции (по методу трапеций) и сами точки разбиения. Для большей наглядности на правом графике показан участок для первых трех точек в увеличенном масштабе. Видно, что параболическая аппроксимация намного лучше соответствует функции $f(x)$, чем линейное приближение по методу трапеций.

Вывод данной формулы Симпсона можно провести различными способами. На рисунке 2.30 приведен вывод формулы Симпсона с использованием определителя Вандермонда и интерполяционного полинома Лагранжа.

В первом случае записываем подынтегральную функцию, записываем ее первообразную, которая легко находится, и значение определенного интеграла по формуле Ньютона-Лейбница $I_{Simp} = F(x_2) - F(x_0)$.

Параболическая аппроксимация фунуции

$f(x) := x^3 - 3 \cdot x^2 + 2 \cdot x + 1$

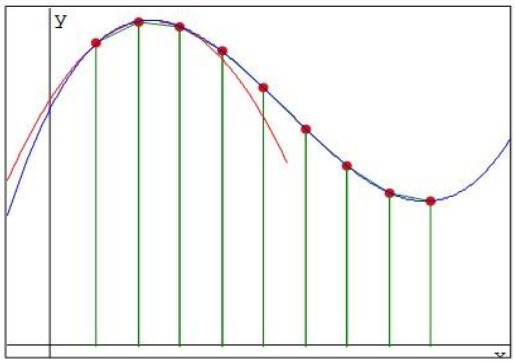
$a := 0,2 \qquad b := 1,6 \qquad N := 8 \qquad h := \frac{b-a}{N}$

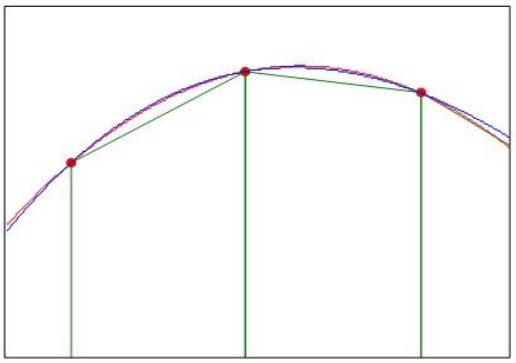
$y(x) := \left| A \cdot x^2 + B \cdot x + C \right.$

$$x_л = \begin{bmatrix} 0,2 \\ 0,375 \\ 0,55 \\ 0,725 \\ 0,9 \\ 1,075 \\ 1,25 \\ 1,425 \\ 1,6 \end{bmatrix} \qquad \overrightarrow{f(x_л)} = \begin{bmatrix} 1,288 \\ 1,380859 \\ 1,358875 \\ 1,254203 \\ 1,099 \\ 0,925422 \\ 0,765625 \\ 0,651766 \\ 0,616 \end{bmatrix}$$

$$P := \text{roots}\left(\begin{bmatrix} 0,2^2 \cdot A + 0,2 \cdot B + C - 1,288 \\ 0,375^2 \cdot A + 0,375 \cdot B + C - 1,380859 \\ 0,55^2 \cdot A + 0,55 \cdot B + C - 1,358875 \end{bmatrix}; \begin{bmatrix} A \\ B \\ C \end{bmatrix}\right)$$

$A := P_1 \qquad B := P_2 \qquad C := P_3$





Вывод формулы Симпсона

метод определителей

$x1 := x0 + \frac{h}{2} \qquad x2 := x0 + h$

$y(x) := A \cdot x^2 + B \cdot x + C \qquad F(x) := \frac{A \cdot x^3}{3} + \frac{B \cdot x^2}{2} + C \cdot x$

$I_{Simp} := F(x2) - F(x0)$

определитель Вандермонда

$$\Delta := \text{maple}\left(\left|\begin{bmatrix} x0^2 & x0 & 1 \\ x1^2 & x1 & 1 \\ x2^2 & x2 & 1 \end{bmatrix}\right|\right) \quad \Delta_1 := \text{maple}\left(\left|\begin{bmatrix} y0 & x0 & 1 \\ y1 & x1 & 1 \\ y2 & x2 & 1 \end{bmatrix}\right|\right) \quad \Delta_2 := \text{maple}\left(\left|\begin{bmatrix} x0^2 & y0 & 1 \\ x1^2 & y1 & 1 \\ x2^2 & y2 & 1 \end{bmatrix}\right|\right) \quad \Delta_3 := \text{maple}\left(\left|\begin{bmatrix} x0^2 & x0 & y0 \\ x1^2 & x1 & y1 \\ x2^2 & x2 & y2 \end{bmatrix}\right|\right)$$

$A := \frac{\Delta_1}{\Delta} \qquad B := \frac{\Delta_2}{\Delta} \qquad C := \frac{\Delta_3}{\Delta}$

$$I_{Simp} := \text{maple}\left(\text{expand}\left(I_{Simp}\right)\right) = \frac{h \cdot (y0 + y2 + 4 \cdot y1)}{6}$$

метод интерполяционных полиномов Лагранжа $L_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{j=0; i \neq j}^{n} \frac{x - x_j}{x_i - x_j}$

$$I_{Simp} := \text{maple}\left(simplify\left(\int_{x0}^{x2} \frac{y0 \cdot (x - x1) \cdot (x - x2)}{(x0 - x1) \cdot (x0 - x2)} + \frac{y1 \cdot (x - x0) \cdot (x - x2)}{(x1 - x0) \cdot (x1 - x2)} + \frac{y2 \cdot (x - x0) \cdot (x - x1)}{(x2 - x0) \cdot (x2 - x1)} \, \mathrm{d}\, x\right)\right)$$

$$I_{Simp} = \frac{h \cdot (y0 + y2 + 4 \cdot y1)}{6}$$

расчет погрешности

$$f2 := f0 + h \cdot f^1 + \frac{h^2}{2} \cdot f^2 + \frac{h^3}{6} \cdot f^3 + \frac{h^4}{24} \cdot f^4 \qquad f1 := f0 - h \cdot f^1 + \frac{h^2}{2} \cdot f^2 - \frac{h^3}{6} \cdot f^3 + \frac{h^4}{24} \cdot f^4$$

$$f(x) := f0 + x \cdot f^1 + x^2 \cdot \frac{f^2}{2} + x^3 \cdot \frac{f^3}{6} + x^4 \cdot \frac{f^4}{24}$$

$$Ii := \text{maple}\left(simplify\left(\int_{-h}^{h} f(x) \, \mathrm{d}\, x - \frac{h}{3} \cdot (4 \cdot f0 + f1 + f2)\right)\right) = -\frac{f^4 \cdot h^5}{90}$$

Рис. 2.30. Параболическая аппроксимация функции. Формула Симпсона

Систему линейных уравнений, которые получаются для вычисления коэффициентов *A*, *B*, *C*, решаем методом Крамера (методом определителей). Главный определитель системы – это определитель Вандермонда, для параболической интерполяции данный определитель третьего ранга.

Далее решается система уравнений для трех точек x_0, x_1 и x_2, и соответствующих им значениям функции y_0, y_1 и y_2. Формула Симпсона получается путем аналитического упрощения I_{Simp}. Ниже приведен вывод той же формулы с использованием полинома Лагранжа второго порядка, которым аппроксимируется исходная функция. Аналитическое вычисление интеграла и его упрощение приводит к той же самой формуле. Из формулы видно, что для применения формулы Симпсона, каждый интервал разбивается пополам, т.е. необходимо вычислять функцию в трех точках. Таким образом, общее число точек разбиения должно нечетным. Т.к. интервал интегрирования получается симметричным относительно своего центра, погрешность метода Симпсона намного меньше всех ранее рассмотренных методов интегрирования.

Этот вариант оформлен в виде пользовательской функции ***Simpson()***. Из рисунка видно, что относительная погрешность метода равна 0 для полинома третьей степени (в *SMath Studio* по умолчанию точность вычислений 10^{-15}).

В нижней части рис. 2.30 приведен аналитический расчет погрешности метода парабол (Симпсона). Воспользуемся аналитическими возможностями *Maple* для интегрирования первых пяти членов разложения получим выражение для погрешности метода. Метод Симпсона имеет пятый порядок малости и третий порядок алгебраической точности. Метод парабол абсолютно точно аппроксимирует полиномы до третьей степени включительно (рис.2.31).

Расширенная формула Симпсона и пример применения показаны на рис. 2.31. Приведены два варианта расширенных формул Симпсона, в первой из них выделены значения функции на концах интервала интегрирования, а во второй реализуется общая сокращенная формула. Второй вариант является предпочтительным, т.к. приходится вычислять только одну сумму.

Пример вычисления интеграла методом Симпсона

$$f(x) := x^3 - 3\cdot x^2 + 2\cdot x + 1 \qquad I_0 := \int_a^b f(x)\,dx = 1,47$$

$$a := 0,2 \qquad b := 1,6 \qquad N := 8 \qquad h := \frac{b-a}{N}$$

$$I_{Simp} := \frac{h}{3}\cdot\left(\frac{f(a)+f(b)}{2} + \sum_{k=1}^{N-1} f(a+k\cdot h) + 2\cdot\sum_{k=1}^{N} f\left(a-\frac{h}{2}+k\cdot h\right)\right) = 1,47$$

$$Il_{Simp} := \frac{h}{6}\cdot\left(\sum_{k=1}^{N}\left(f(a+(k-1)\cdot h) + 4\cdot f\left(a+k\cdot h-\frac{h}{2}\right) + f(a+k\cdot h)\right)\right) = 1,47$$

$$\varepsilon_{Simp} := \frac{|I_0 - I_{Simp}|}{I_0} = 2,34\cdot 10^{-16}$$

$Simpson(a; b; N) :=$

$h := \frac{b-a}{N}$

$I := 0$

for $i \in [1..N]$

$sa := f(a+(i-1)\cdot h)$

$sb := f(a+i\cdot h)$

$sh := f\left(a+k\cdot h-\frac{h}{2}\right)$

$I := I + sa + 4\cdot sh + sb$

$I := h\cdot\frac{I}{6}$

I

$Simpson(a; b; N) = 1,47$

Рис. 2.31. Вычисление определенного интеграла по методу Симпсона

Апостериорные методы оценки погрешностей численного интегрирования. Метод Ромберга

Из вышеизложенного материала видно, что для получения более высокой точности численного интегрирования приходится увеличивать число шагов, что приводит к росту времени счета. Также следует учитывать нелинейную зависимость погрешности расчета и шага интегрирования.

Использование априорной оценки погрешности имеет ряд недостатков, в частности, необходимо оценивать максимальную величину производных высоких порядков на интервале интегрирования. Поэтому кроме априорной оценки погрешности (известной до начала расчета) используется апостериорная погрешность (получаемая в ходе расчета). Априорные оценки погрешности записываются в виде

$$R_0 = A_{hp}, \qquad (18)$$

где A – коэффициент, зависящий от метода интегрирования и вида подынтегральной функции; h – шаг интегрирования, $p = n - 1$ – порядок аппроксимации.

Проведем вычисление определенного интеграла два раза. Сначала используя шаг интегрирования h, а затем пересчитаем результат с другим шагом kh, где константа k может быть как больше, так и меньше единицы. Результатами численного расчета будут величины I_h и I_{kh} (с точностью до пренебрежимо малых слагаемых)

$$I_0 = I_h + Ah^p, \quad I_0 = I_{kh} + A(kh)^p, \tag{19}$$

где I_0 – точное значение интеграла.

Отсюда, получим $R_0 = \dfrac{I_h - I_{kh}}{k^p - 1}$. (20)

Эта формула, выражающая апостериорную оценку главного члена погрешности величины I_0 путем двойного просчета с разным шагом, носит название первой формулы Рунге.

Так как модуль и знак апостериорной погрешности из формулы (20) известны, можно уточнить искомое значение $I_{corr} = I_h + R_0$. Это вторая формула Рунге.

Формула Рунге используется в методе Ромберга для последовательного уточнения значения интеграла при кратном увеличении числа разбиений, что позволяет вычислять определенные интегралы с заданной точностью, когда число разбиений интервала интегрирования осуществляется автоматически. На рис. 2.32 приведена функция ***Romberg****(a; b; ε)*, реализующая вычисление определенного интеграла с заданной точностью. В левой части рисунка приведено два примера. В первом интеграле используется кубическая парабола, для которого уже первое уточнение переводит формулу трапеций в формулу Симпсона, которая дает «истинное» значение для полиномов третьей степени. Во втором примере трансцендентной подынтегральной функции, приходится проводить намного большее большое число разбиений. Высокая эффективность метода Ромберга объясняется тем, что в методе используется минимальное количество шагов необходимое для достижения заданной точности.

Вычисление определенного интеграла методом Ромберга

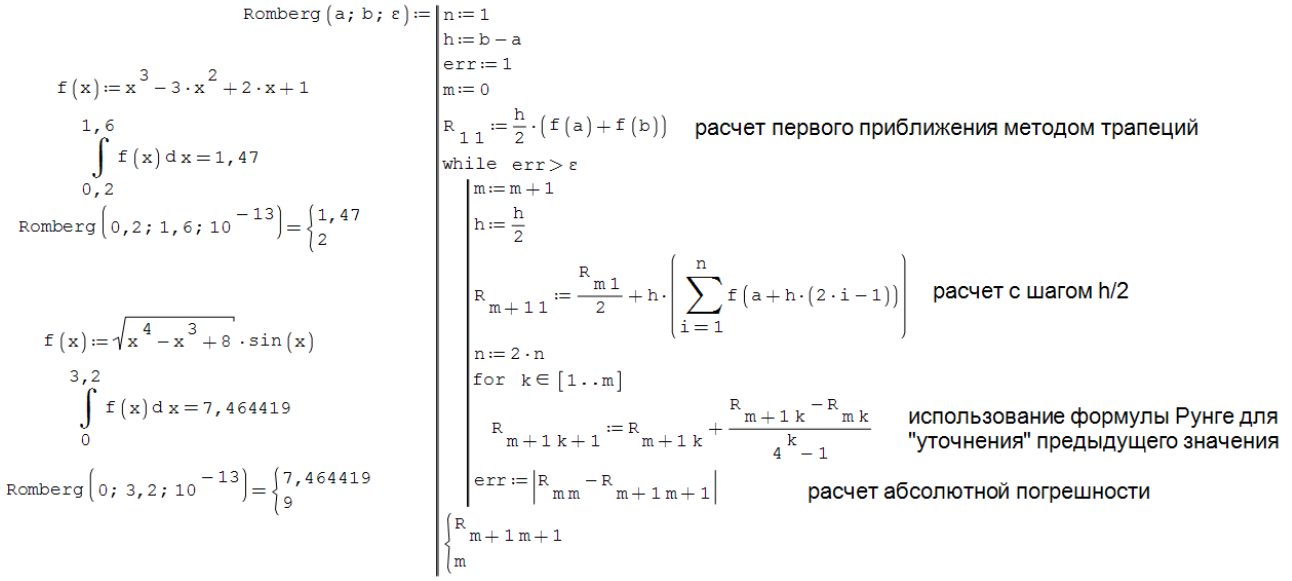


Рис. 2.32. Примеры вычисления определенных интегралов методом Ромберга

Численное интегрирование методом Монте-Карло

Все вышеизложенные методы численного расчета определенных интегралов основаны на полиноминальной аппроксимации подынтегральной функции. Существуют и друге способы аппроксимации, но они не имеют принципиальных преимуществ перед описанными ранее. Все данные методы относятся к детерминированным, т.е. лишенным элемента случайности, методам. Другим типом вычислительных методов являются стохастические (случайные, вероятностные) методы, также известные как методы Монте-Карло. Идея стохастических методов принадлежит Бюффону, который предложил метод расчета числа · путем вычисления вероятности многократного бросания швейной иглы на разлинованную поверхность. Однако, использоваться метод Монте-Карло (название, связанное с местоположением самых известных казино, предложил Н.Метрополис, работая в группе Джона фон Неймана и Станислава Улама над атомным проектом), начал только в середине 20 веке, после появления вычислительных машин.

В некоторых случаях из-за особенности подынтегральной функции (например, из-за ее большой сложности или неявного способа задания), детерминированные методы нельзя или нецелесообразно использовать. В этом случае применяется *метод Монте-Карло*.

Вычислим определенный интеграл, используя определение данное Риманом,

$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i), \quad \text{при } N \to \infty; \tag{21}$$

т.к. при увеличении числа интервалов разбиения, предел не зависит от типа разбиения. В качестве аргумента функции x_i может использоваться случайная величина из интервала [*a*; *b*]. Это один из возможных способов применения Метода Монте-Карло, который, однако, не раскрывает суть стохастического метода. На рис. 2.33 приведен пример расчета интеграла по формуле (21), где X_i – случайные величины на интервале[*a*; *b*].

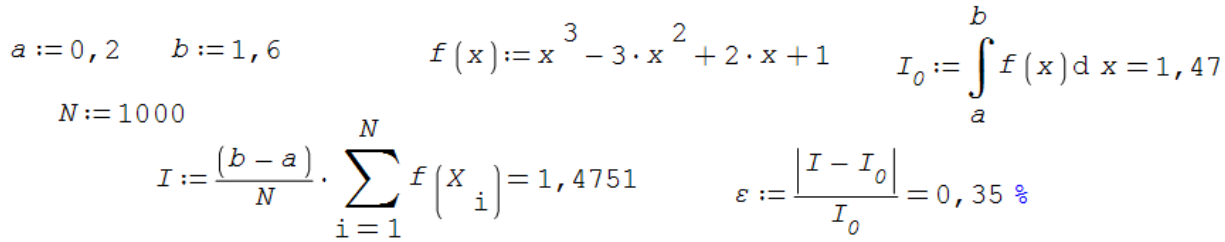


Рис. 2.33. Расчет определенного интеграла методом Монте-Карла по формуле (21)

Используя теорему о среднем, можем записать

$$\int_a^b f(x)dx = (b-a)\cdot f(x_{ab}) = (b-a)\cdot M(f(x)), \qquad a \le x_{ab} \le b.$$

где $f(x_{ab})$ – значение функции в некоторой точке, принадлежащей интервалу интегрирования, а M($f(x)$) – математическое ожидание функции на интервале [*a*; *b*]. Математическое ожидание является аналогом среднего арифметического значения дискретной величины для величины непрерывной. Поэтому, переписывая последнюю формулу имеем

$$\int_a^b f(x)dx = \frac{b-a}{N}\cdot\sum_{i=1}^{N} f(x_i), \tag{22}$$

Т.е. необходимо найти значение функции в большом числе точек (выбранных случайным образом) и взять их среднее арифметическое значение. Как видно, формулы (21) и (22) совпадают.

Математическое ожидание функции имеет и вероятностное определение

$$M(x) = \sum x_i \cdot p_i,$$

$$\int_a^b f(x)dx = (b-a) \cdot M(f(x)) \approx (b-a) \cdot P(f(x)) = (b-a)\frac{S}{N}, \qquad (23)$$

где x_i – значение величины, а p_i – вероятность ее проявления, $P(f(x))$ – вероятность попадания в подынтегральную область, S – число попаданий, N – число испытаний. Предполагая, что все величины равновеликие, то математическое ожидание равно вероятности их появления. Для этого проводят больше число опытов, суммарная вероятность попадания $f(x_i)$ в подынтегральную область, и по формуле (23) находят значение интеграла. Наглядная интерпретация метода Монте-Карло – число попаданий пропорционально площади подынтегральной кривой, т.е. значению определенного интеграла.

Главной проблемой, которая возникает при использовании метода статистических испытаний, является определение случайных величин. В математическом пакете *SMath Sudio* имеется встроенная функция для определения одной случайной величины *random(n)*, где *n* – верхний предел интервала [0; *n*] в котором генерируется целочисленное значение. Для формирования массива случайных чисел в заданном интервале, например, в интервале интегрирования [*a*; *b*] была создана пользовательская функция *RND(N; L; H)*, где *N* – размер массива случайных величин, *L*, *H* – нижний и верхний пределы. Для этого формируется массив *R* в интервале [0; *N*], а затем проводится его масштабирование в интервал [*L; H*]. Полученные таким образом величины достаточно хорошо воспроизводят последовательности случайных величин (рис. 2.34).

$$RND(N; L; H) := \left| \begin{array}{l} \text{for } i \in [1..N] \\ \quad R_i := \text{eval}(\text{random}(100)) \\ R := (H-L) \cdot \left(\text{eval}\left(\frac{R - \min(R)}{\max(R) - \min(R)} \right) \right) + L \end{array} \right.$$

Рис. 2.34. Функция формирования массива случайных чисел в интервале [*L*; *H*]

На рис 2.35 показан пример реализации метода Монте-Карло и его графическая интерпретация.

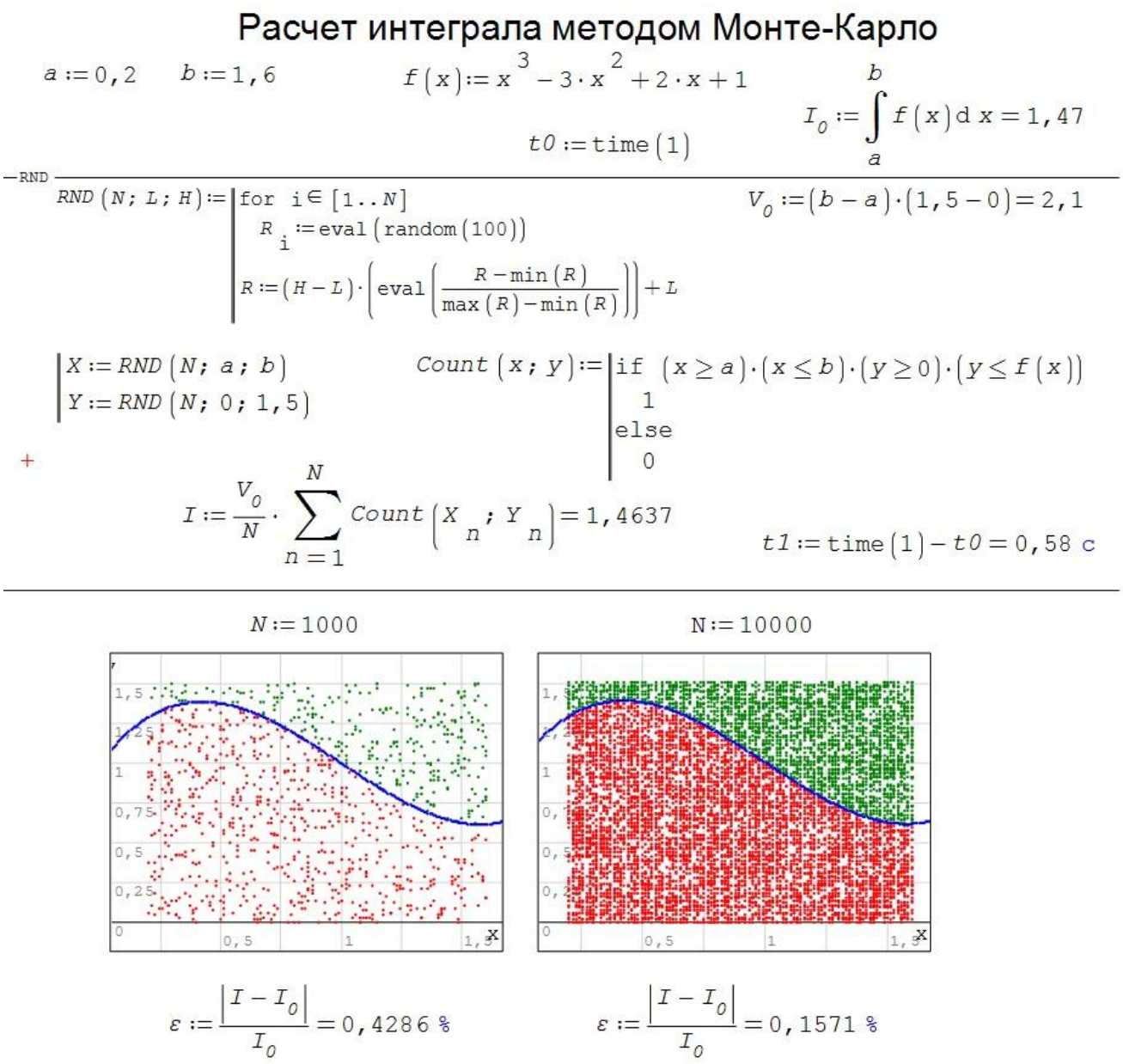


Рис. 2.35. Вычисление определенного интеграла методом Монте-Карло по формуле (23)

Рассмотрим этапы проведения вычисления определенного интеграла методом Монте-Карло. С помощью функции ***RND()*** формируются массивы случайных чисел X_i и Y_i, которые являются координатами точек, заполняющих область V_0. Функция ***Count****(x; y)* формирует массив счетчика числа попаданий в область подынтегральной функции (1 – точка (X_i, Y_i) лежит под *f(x)*, 0 – точка выше *f(x)*). Значение интеграла определяется как произведение размера области вычислений $V_0 = (b - a)(1.5 - 0)$ умноженное на вероятность того, что случайное значение точки (X_i, Y_i) попало в подынтегральную область – *S/N* (из формулы (23)). В нижней части ри-

сунка приведены результаты при $N = 1000$ и $N = 10000$ статистических испытаний. Видно, что увеличение числа точек приводит к более плотному заполнению области V_0 и, соответственно, к уменьшению относительной погрешности расчета.

Однако, если нажать клавишу ***F9***, в верхней части клавиатуры, чтобы *SMath Studio* пересчитала задачу, результат будет совершенно другим. В этом состоит еще один недостаток метода Монте-Карло. Стохастические методы подобны экспериментальному определению различных физических величин, когда измерения проводятся несколько раз, а затем проводится статистическая обработка результатов. Так и в случае использования метода Монте-Карло, расчеты необходимо повторить несколько раз, отбросить самый большой и самый маленький результат (как грубую погрешность или промах), а от оставшихся значений найти среднее арифметическое, которое и будет результатом численного расчета.

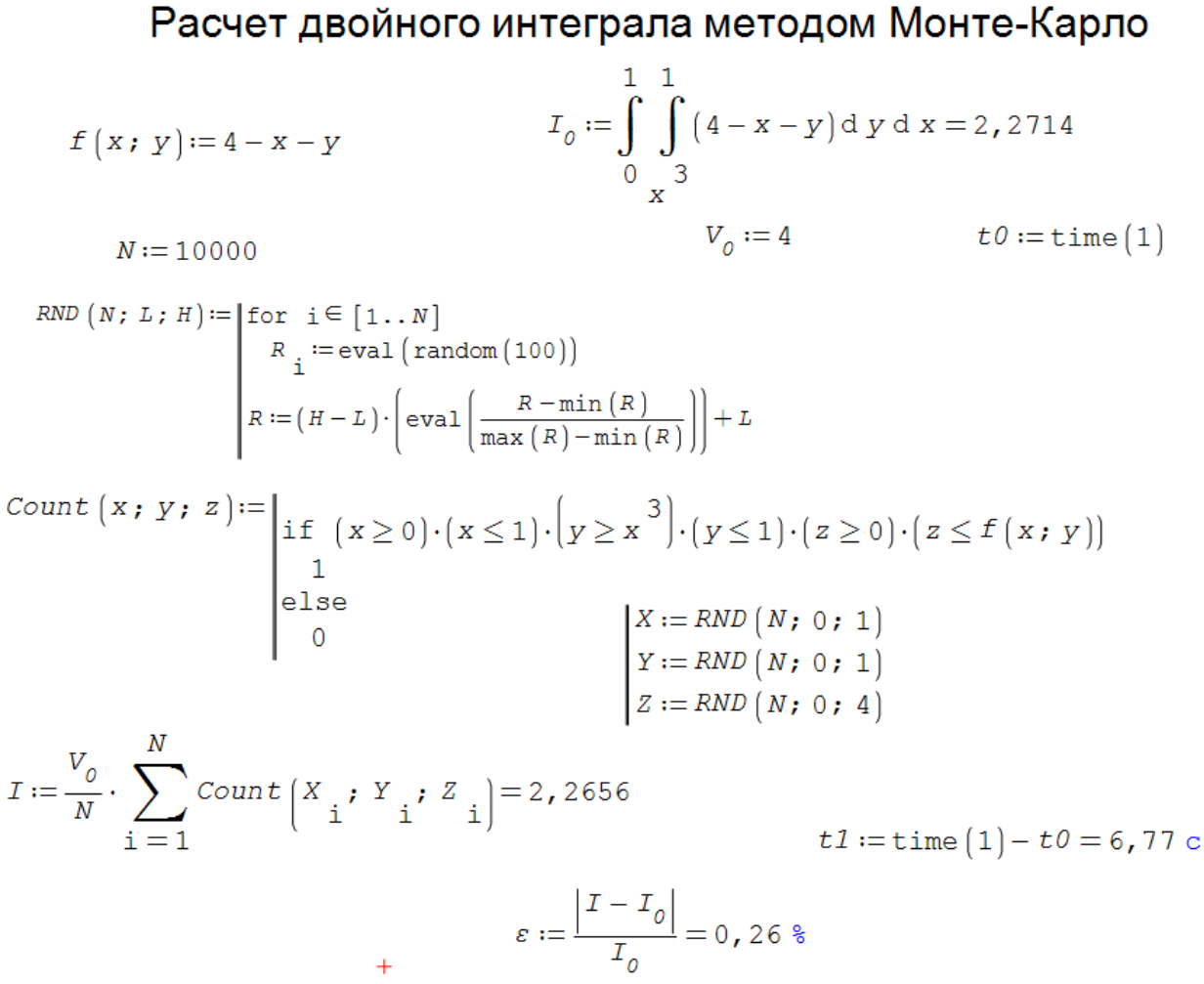


Рис. 2.36. Расчет двойного интеграла методом Монте-Карло

Наибольшая эффективность метода Монте-Карло проявляется при вычислении многомерных и кратных интегралов. Рассмотрим пример использования метода Монте-Карло для вычисления двойного интеграла от заданной функции (рис. 2.36).

Геометрический смысл двойного интеграла это объем, заключенный между областью интегрирования, заданной в плоскости (*XY*) и подынтегральной функцией изменяющейся по оси *Z*. Как видно из рисунка отличие от предыдущего случая заключается в появлении третьей координаты, и соответственно, третьего массива случайных чисел по оси *Z*. Теперь область интегрирования заключена внутрь объема V_0, а функция ***Count****(x; y; z)* задает подынтегральную область.

Метод Монте-Карло позволяет моделировать любой процесс, на протекание которого влияют случайные факторы. Таким образом, можно говорить о методе Монте-Карло как об универсальном методе решения стохастических задач в различных областях математики, естествознания и экономики.

В заключение отметим, что при приближенном вычислении определенных одномерных интегралов метод Монте-Карло значительно уступает квадратурным формулам. В то же время, он легко обобщается на случай многомерных интегралов, и при вычислении интегралов высокой кратности (больше трех) метод Монте-Карло оказывается более эффективным, нежели квадратурные формулы.

2.7. Методические материалы

Контрольные вопросы

1. Поясните геометрический смысл определенного интеграла.
2. Какие принципы лежат в основе численных методов интегрирования?
3. Как ставится задача численного интегрирования? Что такое квадратурные формулы?
4. Что такое квадратурная формула? Что такое узлы и веса?
5. Какие квадратурные формулы Вы знаете? Каковы их общее названия?
6. Каковы общие принципы выбора весов и узлов?
7. Какова общая схема построения квадратурных формул?
8. Каков порядок точности указанных методов?
9. В чем заключается априорный и апостериорный способ оценки по-

грешности численного метода?

10. Как получаются квадратурные формулы Ньютона-Котеса?
11. Получите формулы прямоугольников, трапеций, Симпсона (простые и обобщенные). Каков их геометрический смысл?
12. Запишите оценки погрешности и порядки точности обобщенных формул прямоугольников, трапеций и Симпсона.
13. Запишите оценку погрешности простейших квадратурных формул.
14. Каковы преимущества формулы парабол по сравнению с формулой трапеций и следствием чего являются эти преимущества?
15. В каких случаях квадратурные формулы оказываются точными?
16. Как влияет на точность численного интегрирования величина шага?
17. Из каких составляющих складывается погрешность вычисления интеграла?
18. Можно ли приближенными методами получить результат с той же точностью, что и при помощи аналитического выражения?
19. Какими двумя методами получить расчетные формулы метода прямоугольников и метода трапеций?
20. Как оценить точность вычислений в квадратурных формулах?
21. Что такое метод двойного счета? Как используется эта информация для вычисления интеграла с заданной точностью?
22. В чем состоит алгоритм интегрирования Ромберга?
23. Получите квадратурную формулу Гаусса для $n = 2$.
24. В чем заключаются смысл стохастических методов расчета.
25. Поясните суть метода Монте-Карло для нахождения значения определенного интеграла.

Лабораторная работа № 2

Вычисление определенных интегралов с использованием квадратурных формул

Цель работы – научиться численному интегрированию и оценке погрешности вычислений. Найти приближенное значение определенного интеграла

$$\int_a^b f(x)dx = \frac{b-a}{N} \cdot \sum_{i=1}^{N} c_i f(x_i),$$

где *f(x)* – непрерывная функция на отрезке [*a*, *b*], *N* – число интервалов

разбиения, с помощью квадратурных формул Ньютона-Котеса и Гаусса-Кристоффеля. Оценить априорную и апостериорную погрешности полученных результатов.

Постановка задачи

1. Изучить теоретические сведения. Научиться выводить основные квадратурные формулы в сокращенной и расширенной форме.
2. Выполнить расчет приближенного значения определенного интеграла, используя квадратурные формулы Ньютона-Котеса и Гаусса-Кристоффеля. Принять $n = 12$, где n – количество интервалов разбиения отрезка интегрирования $[a, b]$.
3. Найти необходимое количество интервалов разбиения n, для вычисления интеграла с точностью 10^{-6}.
4. Провести с помощью математического пакета *SMath Studio* приближенное значение определенного интеграла функции $f(x)$ на интервале $[a, b]$ для произвольного числа интервалов разбиения n. Оформить все вычисления в одном программном блоке или функции.
5. Сравнить количество шагов различных квадратурных формул, необходимых для достижения заданной погрешности.
6. Оценить априорную погрешность вычисления приближенного интеграла для каждой из квадратурных формул. Оформить все вычисления в одном программном блоке.
7. Рассчитать значение интеграла с двойным шагом для определения апостериорной погрешности. Оценить апостериорную погрешность вычисления приближенного интеграла для квадратурных формул Ньютона-Котеса. Оформить все вычисления в одном программном блоке (или функции).
8. Выполнить расчет заданного варианта. Проверить полученные результаты с помощью встроенных функций пакетов. Построить график функции $f(x)$.

Варианты заданий

1.	$\int_0^{\pi} \frac{\cos x}{\sqrt{1+x^2}}dx$	2.	$\int_0^{\pi} \frac{\sin x}{1+x^2}dx$	3.	$\int_1^2 e^{\left(x^{-2}-x^2\right)}dx$	4.	$\int_0^{\pi} \frac{\cos x}{1+x}dx$
5.	$\int_{0.1}^{\pi/2} e^{-\frac{x}{\sin x}}dx$	6.	$\int_1^2 x^{-2}e^{-2x}dx$	7.	$\int_0^{\pi} \frac{\cos x}{1+x^2}dx$	8.	$\int_{0.1}^{\pi/2} e^{-\frac{x}{\sin x}}dx$
9.	$\int_0^{\pi} \frac{x\cos x}{1+x^2}dx$	10.	$\int_0^{\pi} \frac{x\sin x}{1+x}dx$	11.	$\int_0^{\pi} \frac{x\sin x}{\sqrt{1+x^2}}dx$	12.	$\int_0^{\pi} \frac{\sin x}{1+x}dx$

Примечание. Для четных вариантов использовать методы левых, средних прямоугольников, Симпсона, Буля, Ромберга и Гаусса с $n = 3$; для нечетных вариантов используются методы правых и средних прямоугольников, трапеций, Симпсона 3/8, Ромберга и Гаусса с $n = 2$.

Контрольные вопросы:

1. Поясните геометрический смысл определенного интеграла.
2. Какие принципы лежат в основе численных методов интегрирования?
3. Как ставится задача численного интегрирования? Что такое квадратурные формулы?
4. Что такое квадратурная формула? Что такое узлы и веса?
5. Какие квадратурные формулы Вы знаете? Каковы их общее названия?
6. Каковы общие принципы выбора весов и узлов?
7. Какова общая схема построения квадратурных формул?
8. Каков порядок точности указанных методов?

Отчет должен содержать

1. Краткое описание изученных методов.
2. Листинг (файлы *.sm*) выполненных заданий 1 – 8.
3. Сравнение результатов расчета различными методами .
4. Заключение о применимости методов. Выбор оптимального метода.
5. Ответить на контрольные вопросы.

Лабораторная работа № 3

Вычисление определенных одинарных и двойных интегралов методом Монте-Карло

Цель работы – научиться численному интегрированию и оценке погрешности вычислений с использованием метода статистических испытаний (метод Монте-Карло).

Постановка задачи

1. Изучить теоретические сведения о методе Монте-Карло и способы использования для расчета определенных интегралов.
2. Выполнить расчет приближенного значения одномерного определенного интеграла (варианты использовать из предыдущей лабораторной работы), используя метод Монте-Карло. Число испытаний принять равным 1000, 5000 и 10000.
3. Для расчетов использовать формулы (12) и (14). Проверить полученные результаты с помощью встроенных функций пакета *SMath Studio*. Определить погрешность результатов статистических испытаний, используя методику, описанную в пункте 3.7. Оформить все вычисления в одном программном блоке или функции.
4. Рассчитать значение двойного интеграла с помощью математического пакета *SMath Studio*. Оценить погрешность вычисления приближенного интеграла для метода Монте-Карло. Оформить все вычисления в одном программном блоке (или функции).

Контрольные вопросы:

1. Охарактеризовать методику проведения статистического эксперимента
2. Расписать метод статистических испытаний (метод Монте-Карло).
3. Как зависит точность эксперимента от количества прогонов модели.
4. В каких случаях целесообразно применять имитационное моделирование.
5. Преимущества и недостатки использования имитационного моделирования.

Отчет должен содержать

1. Краткое описание метода
2. Листинг выполненных заданий 2 и 4.
3. Сравнение результатов расчета по методу Монте-Карло с аналитическим результатом.
4. Заключение о применимости метода, зависимости времени расчета от числа статистических испытаний, зависимости погрешности расчета от числа испытаний.
5. Ответить на контрольные вопросы.

3. ФИЗИКА, ЭЛЕКТРОТЕХНИКА, ЭЛЕКТРОНИКА И SMATH STUDIO

3.1. Обработка результатов измерений

Одна из первых задач, которая возникает у студентов при изучения курса физики, является задача обработки результатов физических экспериментов. Рассмотрим возможность реализации данной задачи в пакете *SMath Studio*.

Как известно, погрешности измерений можно разделить на грубые погрешности (или промахи), систематические (в основном, приборные) и случайные. Также следует различать погрешности прямых и косвенных измерений.

Абсолютной погрешностью измерения называется разность между измеренным значением и его «истинным» значением $\Delta X = X - X_0$. Относительная погрешность измерения, равная отношению абсолютной погрешности к истинному значению и вычисляется по формуле

$$\varepsilon_X = \frac{\Delta X}{X_0}.$$

Т.к. истинное значение измеряемой величины неизвестно, то на практике можно лишь приближенно оценить погрешность измерения.

За истинное значение физической величины принято принимать ее среднее арифметическое (или действительное) значение, следовательно, чем больше опытов будет проведено, тем ближе мы будем приближаться к «истинному» значению. Однако бесконечно большое количество разнообразных факторов влияющих на результаты отдельных опытов приводят к тому, что абсолютная погрешность отлична от нуля. Для того чтобы уменьшить погрешность измерений приходится проводить математическую обработку результатов измерений, включающую в себя введение поправок на методические погрешности, уменьшение систематических погрешностей и статистический учет случайных погрешностей.

Рассмотрим пример обработки измерений в программе *SMath Studio*, использование данной программы может существенно облегчить обработку экспериментальных данных. Зададим измеряемую величину в виде вектора-столбца X (число измерений выберем равным $N = 7$). Предположим, что среди экспериментальных данных присутствуют грубые погреш-

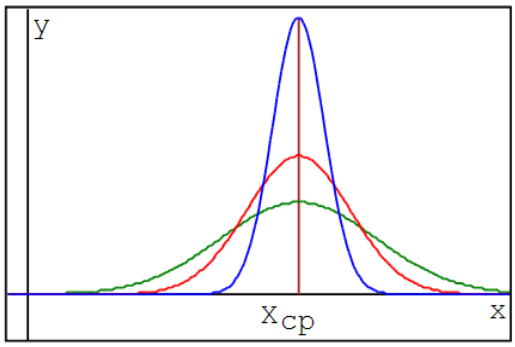
ности (промахи). Для этого отсортируем вектор *X* по возрастанию величин, используя встроенную функцию ***sort()***. За промахи примем самое малое и самое большое значение *X*, которые отбросим (на данном этапе). Используем оставшиеся пять значений для расчета среднего арифметического – действительного значения величины *X* (рис. 3.1).

$$X_э := \begin{bmatrix} 1,844 \\ 1,631 \\ 1,564 \\ 1,673 \\ 1,779 \\ 1,701 \\ 1,558 \end{bmatrix} \quad X := \mathrm{sort}(X_э) = \begin{bmatrix} 1,558 \\ 1,564 \\ 1,631 \\ 1,673 \\ 1,701 \\ 1,779 \\ 1,844 \end{bmatrix} \quad X := \mathrm{submatrix}(X;\ 2;\ 6;\ 1;\ 1) = \begin{bmatrix} 1,564 \\ 1,631 \\ 1,673 \\ 1,701 \\ 1,779 \end{bmatrix}$$

$$X_{ср} := \frac{X_1 + X_2 + X_3 + X_4 + X_5}{5} = 1,6696 \quad X_{ср} := \frac{\sum_{i=1}^{\mathrm{rows}(X)} X_i}{\mathrm{rows}(X)} = 1,6696 \quad \frac{\sum X}{\mathrm{rows}(X)} = 1,6696$$

Рис. 3.1. Исключение «промахов» и расчет среднего арифметического значения

После сортировки элементов вектора $X_э$, формируется вектор *X*, содержащий со второго по шестой элемент первого столбца $X_э$, для этого используется встроенная функция ***submatrix()***. Далее рассчитывается среднее арифметическое тремя способами (прямой расчет, использование встроенной функции из правой панели и использование функции сокращенного суммирования, вызывается с клавиатуры комбинацией «sum» «TAB»). Для придания фрагменту большей универсальности, возможности использования любого числа данных в *X*, используется встроенная функция подсчета числа элементов в столбце ***rows()***.



$$f(x) := \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot e^{-\frac{(x - X_{ср})^2}{2 \cdot \sigma^2}}$$

Рис. 3.2. Нормальное распределение

Если на результаты проведенных опытов влияют случайные факторы и количество опытов стремится к бесконечности, то мы будем наблюдать так называемое нормальное распределение Гаусса (рис. 3.2), где $X_{ср}$ – среднее арифметическое значение измеряемой величины, σ^2 – дисперсия (разброс) значений измеряемой величины.

Из рисунка видно, что случайная величина имеет максимум, соответствующий среднему арифметическому $X_{ср}$, а дисперсия определяет размытие результатов (чем больше σ, тем нормальное распределение шире и ниже), т.е. отклонение измеренного значения X сильнее отличается от $X_{ср}$. Распределение Гаусса нормировано,

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-X_{cp})^2}{2\sigma^2}} dx = 1.$$

Нормальное распределение можно рассматривать как плотность вероятности появления измеренного значения x. А интеграл

$$\int_{-\alpha}^{\alpha} \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}} dx = \frac{2}{\sigma\sqrt{2\pi}} \int_{0}^{\alpha} e^{-\frac{x^2}{2\sigma^2}} dx = P(\alpha),$$

(здесь для упрощения дальнейших расчетов заменили (x - $X_{ср}$) на x). Данный интеграл определяет вероятность попадания измеренного значения в некоторый интервал, который называется доверительным интервалом.

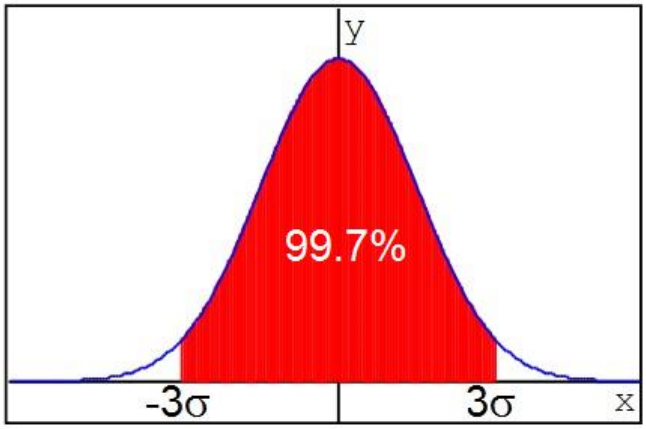


Рис. 3.3. Правило 3σ, вероятность попадания случайной величины с нормальным распределением в интервал [-3σ, 3σ] равно 99.7%

Для оценки случайной погрешности измерений задаются доверительным интервалом, который определяет максимальный разброс экспериментальных данный вокруг $X_{ср}$ с заданной доверительной вероятностью α (доверительная вероятность обычно, выбирается из ряда 98%, 99%, 99.5%, 99.9%).

Однако, все последние рассуждения справедливы для большого числа измерений. Ирландский математик Уильям Госсет, работая в пивоваренной компании Гиннес, в 1907 году разработал метод оценки случайных погрешностей для случая малого числа измерений. Сотрудникам компании было запрещено публиковать результаты своих работ и У.Госсет опубликовал свои результаты под псевдонимом Стьюдент. Метод оказался очень простым и эффективным, поэтому стал основным методом анализа случайных величин. Программа *SMath Studio* позволяет наглядно изучить свойства распределения Студента. На рис. 3.4 приведено сравнение распределений Гаусса и Стьюдента. Видно, что при малом числе измерений, распределение Стьюдента шире, чем Гаусса, а при числе измерений более $N > 30$, оба распределения, практически, совпадают.

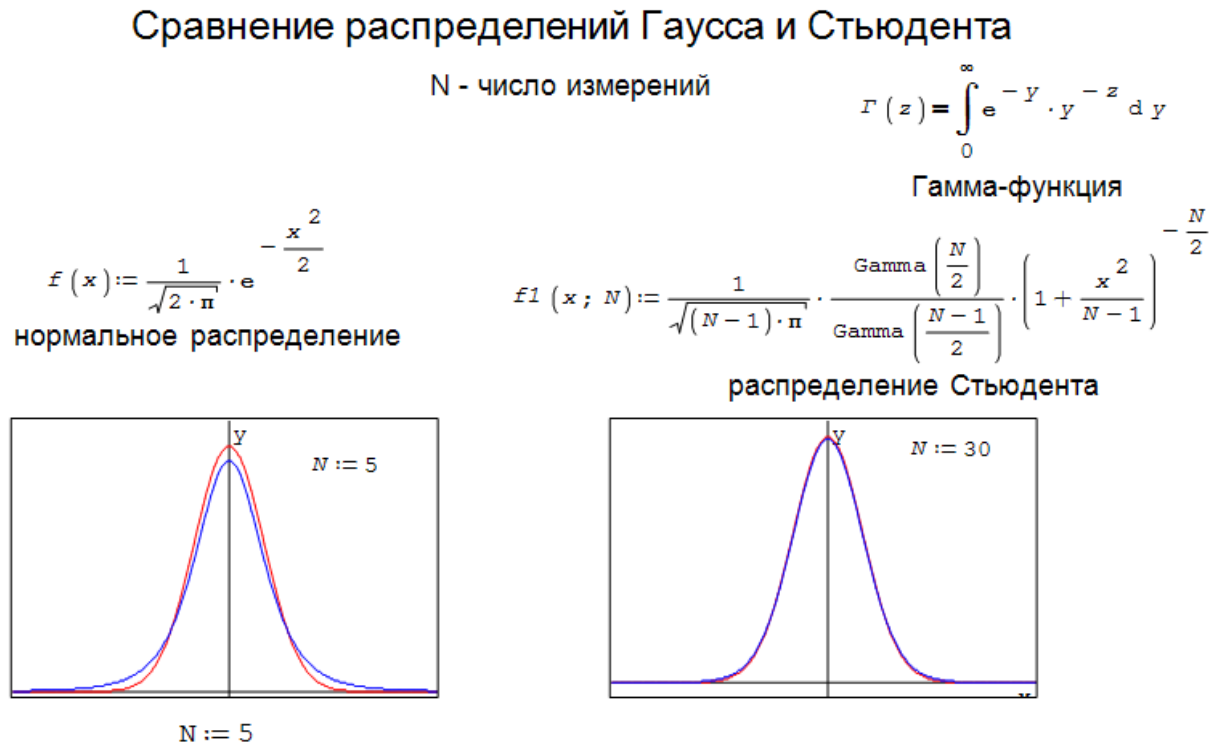


Рис. 3.4. Сравнение распределений Гаусса и Стьюдента для $N = 5$

Ниже приведен расчет коэффициентов Стьюдента для $N = 5$ для различных доверительных вероятностей в программе *SMath Studio*, для этого находятся корни нелинейного уравнения (функция ***solve()***), где граница доверительного интервала стоит справа от знака равно, верхним пределом в интеграле выступает искомый коэффициент Стьюдента. Для проверки расчетов приведена таблица коэффициентов Стьюдента, взятая из литературных источников.

$$\text{solve}\left(2\cdot\int_0^x f(t)\,dt = 0{,}9;\ x\right) = 2{,}13$$

$$\text{solve}\left(2\cdot\int_0^x f(t)\,dt = 0{,}95;\ x\right) = 2{,}78$$

$$\text{solve}\left(2\cdot\int_0^x f(t)\,dt = 0{,}98;\ x\right) = 3{,}75$$

$$\text{solve}\left(2\cdot\int_0^x f(t)\,dt = 0{,}99;\ x\right) = 4{,}6$$

$$\text{solve}\left(2\cdot\int_0^x f(t)\,dt = 0{,}999;\ x\right) = 8{,}61$$

Коэффициенты Стьюдента

n	Доверительная вероятность $P_д$				
	0,90	0,95	0,98	0,99	0,999
2	6,31	12,71	31,82	63,68	636,62
3	2,92	4,30	6,97	9,93	31,60
4	2,35	3,18	4,54	5,84	12,92
5	2,13	2,78	3,75	4,60	8,61
6	2,02	2,57	3,37	4,06	6,87
7	1,94	2,45	3,14	3,71	5,96
8	1,90	2,37	3,00	3,50	5,41
9	1,86	2,31	2,90	3,36	5,04
10	1,83	2,26	2,82	3,25	4,78
12	1,80	2,20	2,72	3,11	4,44
14	1,77	2,16	2,65	3,01	4,22
16	1,75	2,13	2,60	2,95	4,07
18	1,74	2,11	2,57	2,90	3,97
20	1,73	2,09	2,54	2,86	3,88
∞	1,65	1,96	2,33	2,58	3,29

$$\Delta X := X - X_{cp} = \begin{bmatrix} -0{,}1056 \\ -0{,}0386 \\ 0{,}0034 \\ 0{,}0314 \\ 0{,}1094 \end{bmatrix}$$

абсолютные погрешности

$$S_n := \sqrt{\frac{\sum_{i=1}^{N}\left(\Delta X_i\right)^2}{N\cdot(N-1)}} = 0{,}0358$$

среднеквадратичная погрешность

$t_{\alpha N} := 2{,}78$

коэффициент Стьюдента для N = 5, α = 0.95

$\Delta X_{дов} := t_{\alpha N}\cdot S_n = 0{,}099$

доверительный интервал

$\Delta X_{пр} := 0{,}01$

приборная погрешность

$$\Delta X := \sqrt{\Delta X_{дов}^{\ 2} + \Delta X_{пр}^{\ 2}} = 0{,}1$$

суммарная абсолютная погрешность

$$\varepsilon_X := \frac{\Delta X}{X_{cp}} = 5{,}99\ \%$$

относительная погрешность

Рис. 3.5. Расчет среднеквадратичной погрешности, доверительного интервала, полной абсолютной погрешности и относительной погрешности (продолжение рис. 3.1)

Далее (рис. 3.5) рассчитывается абсолютные погрешности для каждого измерения, среднеквадратичная погрешность S_n, используя рассчитанный или взятый из таблиц коэффициент Стьюдента $t_{\alpha N}$, рассчитывается доверительный интервал (т.е. определяется случайная погрешность экспериментальных данных). Далее, используя значение приборной погрешности, рассчитываются полная абсолютная и относительная погрешности результатов прямых измерений.

Возвращаемся к рис. 3.1, при обработке экспериментальных данных, мы отбросили самое малое и самое большое экспериментальные значения, приняв их за промахи. Используя, например, критерий Шовине, определим так ли это. Для 4 – 6 измерений критерий Шовине записывается как

$$\left|X_{cp} \pm X_{\substack{\min \\ \max}}\right| \le 1.6 S_n \sqrt{5}; \quad 1.6 S_n \sqrt{5} = 0.128; \quad \left|X_{cp} - \min(X_Э)\right| = 0.1116; \quad \left|X_{cp} - \min(X_Э)\right| = 0.1744.$$

Видно, что по критерию Шовине, минимальное значение не является промахом, а максимальное – является. Поэтому расчет по математической обработке результатов прямого измерения необходимо повторить, включив в расчет минимальное измеренное значение, которое не учитывали ранее.

Ответ экспериментально определяемой величины записывается в виде

$$X = (X_{cp} \pm \Delta X)[ед.изм.], \qquad \varepsilon_X = \Delta X / X_{cp},$$

где $X_{ср}$ – среднеарифметическое (или действительное) значение физической величины, ΔX – абсолютная погрешность, ε_X – относительная погрешность величины X. Подавляющее большинство физических величин имеют размерность, которую необходимо записывать в квадратных скобках. Часто студенты забывают указывать размерности, что является грубой ошибкой!

Объедения текст программ, приведенных на рис 3.1 и 3.5, получаем полную программу обработки экспериментальных данных, которая может облегчить выполнение лабораторных работ по различным дисциплинам.

3.2. Движение тела брошенного под углом к горизонту

С использование программы *SMath Studio*, решим школьную задачу о камне, брошенном под углом к горизонту с некоторой начальной скоростью (рис. 3.6). Вначале, пренебрежем силой сопротивления воздуха, и найдем основные параметры данного криволинейного движения. В данном случае, в программе их легко изменить, тело имеет начальную скорость 20 м/с, направленную под углом 45^0 к горизонту, ускорение свободного падения равно 9.8 м/с^2 (для введения единиц измерения используются горячие клавиши «Ctrl W» или значок в верхнем меню).

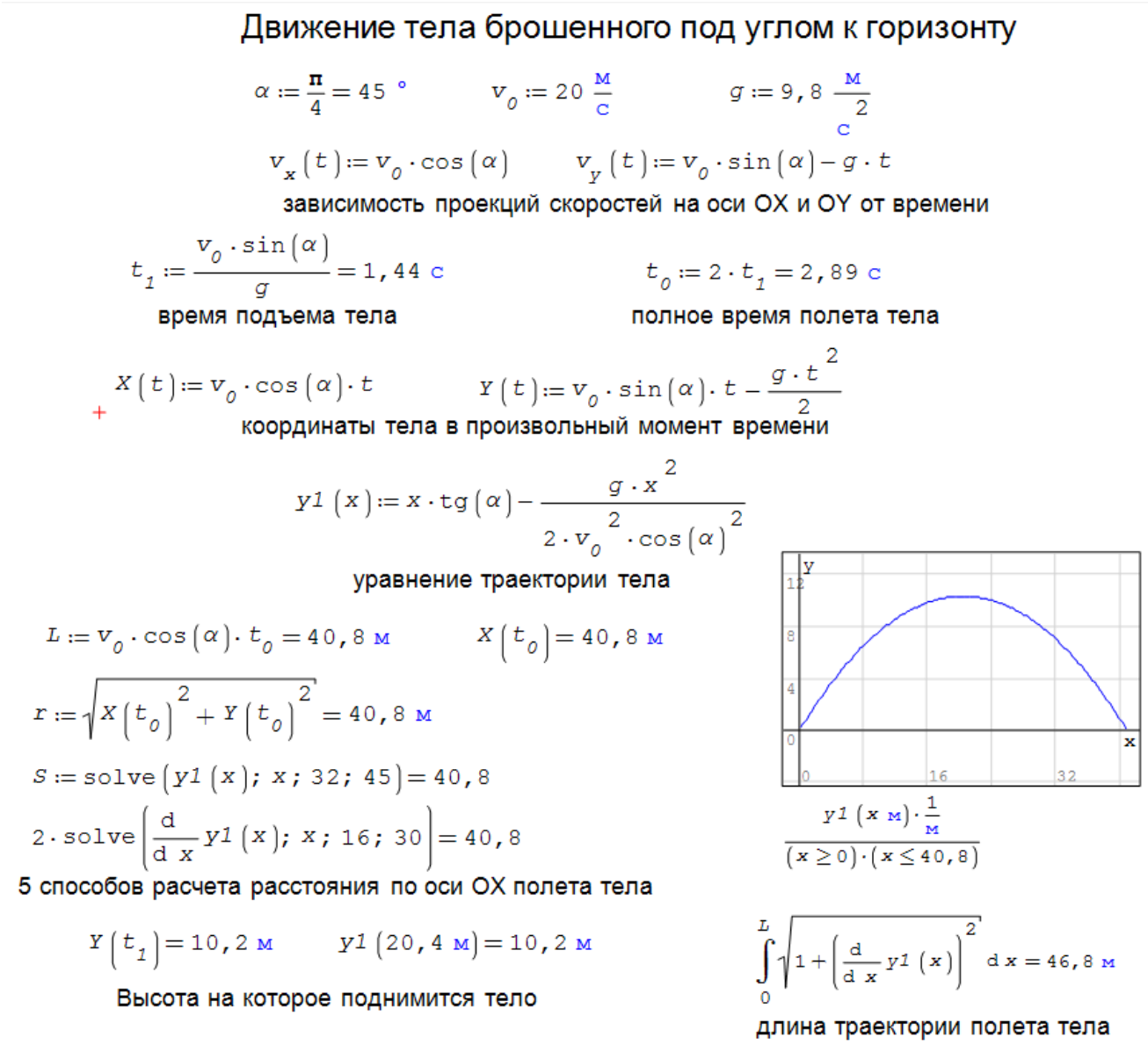


Рис. 3.6. Движение тела брошенного под углом к горизонту

Как известно, сложное криволинейное движение, по принципу суперпозиции сводится к двум более простым одномерным движениям тела вдоль декартовых осей координат *OX* и *OY*. Запишем проекции скоростей на данные оси. Скорость $v_x(t)$ вдоль оси *OX*, есть величина постоянная (тело движется равномерно). Движение по оси *OY* является равнозамедленным на первой половине пути и равноускоренным на второй. Скорость $v_y(t)$ находится по соответствующей формуле (см. рис. 3.6).

Из уравнения скорости вдоль оси *OY*, легко найти время подъема тела на максимальную высоту, это время t_1. Т.к. сопротивлением воздуха пренебрегаем, полное время полета тела равно t_0.

Далее, записываем уравнения движения вдоль координатных осей *OX* и *OY*. Если из уравнения *X(t)* выразить время *t* и подставить его в уравнения для *Y(t)*, получим уравнение траектории полета камня (тела) $y_1(t)$. Изобразим данную траекторию на 2D графике. Здесь, проявляются особенности работы программы *SMath Studio* с единицами измерения. Системная переменная x, по которой строится график, величина безразмерная, нам приходится дописывать выбранную размерность – расстояние в метрах, соответственно, перемещение по оси OY, наоборот, переводим в безразмерный вид (чтобы не трогать знаменатель). Знаменатель, в выражении под графиком, нужен для ограничения параболы точками начала движения (моментом броска камня) и моментом падения камня на землю.

Используя уже выведенные выражения, найдем путь, пройденный телом, т.е. расстояние, на которое улетит камень вдоль оси OX. Проведем расчет различными способами:

1) Путь равен произведению скорости $v_x(t)$ на время t_0;

2) Путь равен координате по оси OX *X(t)* в момент падения камня на землю t_0;

3) Путь как суперпозиция перемещений по осям *OX* и *OY*, полное перемещение по оси *OY* равно нулю (высота подъема камня со знаком плюс, падение камня со знаком минус);

4) Путь, как нахождение второго корня квадратного уравнения параболы (первый корень – это точка бросания с координатой $x = 0$). Здесь используется встроенная функция ***solve()*** с четырьмя параметрами, два последних – это интервал, в пределах которого ищется корень уравнения;

5) Путь как удвоенное расстояние до точки максимального подъема

камня. Максимальная точка траектории это точка, в которой производная $dy1(t)/dt = 0$.

В двух последних пунктах необходимо в свойствах выражения содержащего функцию ***solve()***, (вызывается правой клавишей мыши), выбрать пункт «Без учета единиц измерений», т.к. внутренняя переменная x – величина безразмерная (или руками вставлять размерности в выражение).

В конце рассчитаем длину траектории полета камня как криволинейный интеграл.

Задача о брошенном камне является хорошим примером (простым и наглядным), на котором можно изучать различные кинематические характеристики. Построим анимированное изображение полета камня (рис. 3.7).

Рис. 3.7. Анимация полета камня брошенного под углом к горизонту

В данном фрагменте будем использовать безразмерные переменные. Все формулы взяты из предыдущего рисунка. Для построения анимации разобьем полет камня на 30 точек по времени [1..31]. Расчет оформлен в виде программного блока. В цикле ***for*** проводится дискретизация задачи, непрерывная задача сводится к вычислению параметров в заданные моменты времени. Т.е. проводится «аналого-цифровое» преобразование. Проверяем, что начальное время равно нулю. Вычисляем координаты x, y, v_x, v_y в выбранные моменты времени. Затем проводится обратная операция – «цифро-аналоговое» преобразование, т.к. на графике мы хотим получить плавное движение тела.

Строим анимированный график по переменной ***step***. На график выводим пять объектов: траекторию движения (ограниченную точками бросания и падения камня); вектора горизонтальной v_x и вертикальной v_y скоростей тела, красную точку – соответствующую «мгновенному» положению тела и вектор начальной скорости v_0.

На графике видно, что горизонтальная скорость постоянна, вертикальная скорость, сначала, уменьшается до нуля в верней точке траектории, а затем начинает равномерно расти до момента падения камня на землю.

Продолжаем изучать кинематические характеристики криволинейного движения тела брошенного под углом к горизонту. Для этого постоим анимированный график с рассчитанными ускорениями – нормальным и тангенциальным (рис. 3.8).

В предыдущих фрагментах криволинейное движение рассматривалось как суперпозиция движений тела вдоль декартовых координатных осей. Рассмотрим движение тела как суперпозицию движений вдоль траектории и в направлении нармальной (перпендикулярной) к ней. В данном случае мы можем получить информацию о тангенциальном и нормальном ускорении, которые описывают особенности криволинейного движения. На рис. 3.8 показана программа расчета тангенциального и нормального ускорений и радиуса кривизны траектории движения тела.

$\alpha := \frac{\pi}{4}$ $v_0 := 20$ $g := 9,8$

$y1(x) := x \cdot tg(\alpha) - \frac{g \cdot x^2}{2 \cdot v_0^2 \cdot \cos(\alpha)^2}$

$t1 := \frac{v_0 \cdot \sin(\alpha)}{g} = 1,4431$

$S := 2 \cdot v_0 \cdot \cos(\alpha) \cdot t1 = 40,8163$

$N := 30$

for $i \in [1..(N+1)]$

$t_i := (i-1) \cdot 2 \cdot \frac{t1}{N}$

$x_i := v_0 \cdot \cos(\alpha) \cdot t_i$

$y_i := v_0 \cdot \sin(\alpha) \cdot t_i - g \cdot \frac{(t_i)^2}{2}$

$vx_i := v_0 \cdot \cos(\alpha)$

$vy_i := v_0 \cdot \sin(\alpha) - g \cdot t_i$

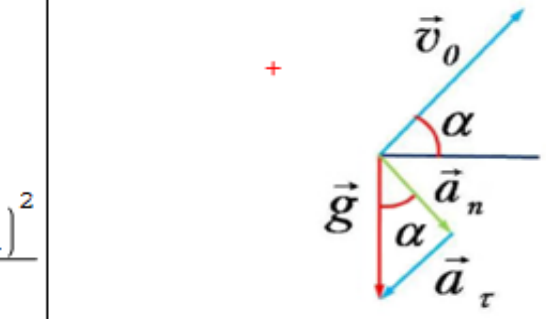
$x(t) := x_t$

$y(t) := y_t$

$vx(t) := vx_t$

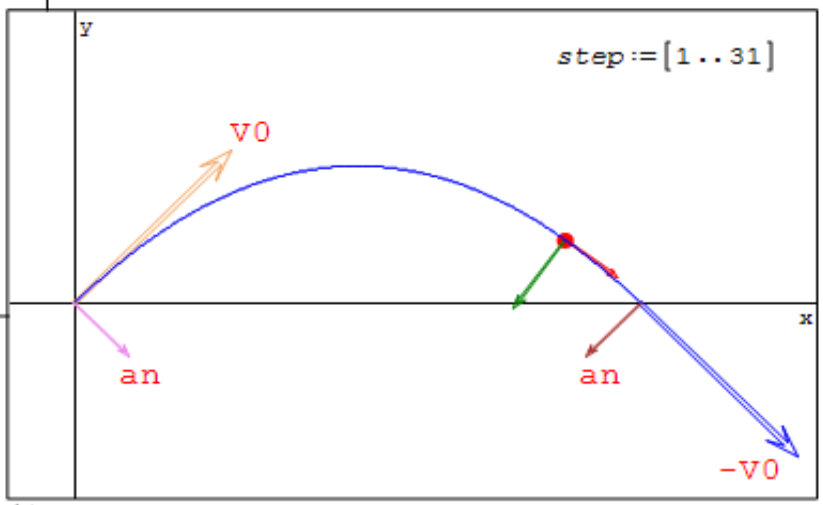
$vy(t) := vy_t$

$\varphi(t) := \text{arctg}\left(\frac{vy_t}{vx_t}\right)$



$v(t) := \sqrt{(vx(t))^2 + (vy(t))^2}$

step := [1..31]



$R_0 := \frac{v_0^2 \cdot \cos(\alpha)^2}{g} = 20,41$

$R_0 := \frac{v_0^2}{g \cdot \cos(\alpha)} = 57,72$

$\begin{cases} y1(x) & \text{if } (x \geq 0) \wedge (x \leq S) \\ "" & \text{otherwise} \end{cases}$

Affine $\left(x(t);\ y(t);\ \varphi(t);\ \frac{(-g) \cdot vy(t)}{v(t)};\ \text{arrow1}\right)$

Affine $\left(x(t);\ y(t);\ \varphi(t) - \frac{\pi}{2};\ \frac{g \cdot vx(t)}{v(t)};\ \text{arrow1}\right)$

Affine $(0;\ 0;\ -\alpha;\ g \cdot \cos(\alpha);\ \text{arrow1})$

Affine $(0;\ 0;\ \alpha;\ v_0;\ \text{arrow1})$

Affine $(S;\ 0;\ \alpha;\ (-g) \cdot \cos(\alpha);\ \text{arrow1})$

Affine $(S;\ 0;\ -\alpha;\ v_0;\ \text{arrow1})$

augment $(x(t);\ y(t);\ ".";\ 25;\ "red")$

augment $(11;\ 14;\ "V0";\ 15;\ "red")$

augment $(46;\ -11;\ "-V0";\ 15;\ "red")$

augment $(3;\ -4;\ "an";\ 15;\ "red")$

augment $(36;\ -4;\ "an";\ 15;\ "red")$

Рис. 3.8. Кинематические характеристики криволинейного движения

В дополнении к рассчитанным параметрам нам понадобится угол между направлениями скоростей v_y и v_x. Вспоминая, что нормальное и тангенциальное ускорение находится по формулам

$$\vec{a}_n = \frac{\vec{v}^2}{R}\vec{n}, \quad \vec{a}_\tau = \frac{d|\vec{v}|}{dt}\vec{\tau}, \tag{3.1}$$

где v – мгновенная (линейная) скорость тела, R – радиус кривизны траектории, τ и n – единичные вектора, направленные вдоль траектории (тангенциальное направление) и перпендикулярно к ней (направление нормали). В нашей задаче полным ускорением является ускорение свободного падения

$$\sqrt{\vec{a}_n{}^2 + \vec{a}_\tau{}^2} = g.$$

На рис. 3.8 показано разложение полного ускорения на нормальное и тангенциальное ускорение, в начальный момент времени. Следовательно, зная полную (линейную, мгновенную) скорость $\sqrt{v_x{}^2 + v_y{}^2} = v,$ и угол $\varphi(t)$ наклона касательной к оси OX, можно определить тангенциальное и нормальное ускорение

$$a_\tau(t) = -g\sin(\varphi(t)) = -g\frac{v_y}{v}, \; a_n(t) = g\cos(\varphi(t)) = g\frac{v_x}{v}.$$

Направление мгновенной (полной) скорости и тангенциального ускорения совпадают с углом $\varphi(t)$, а нормальное ускорение направлено перпендикулярно, т.е. повернуто на 90^0 против часовой стрелки.

Из рис. 3.8 видно, что в отсутствие силы сопротивления, начальные и конечные скорость и нормальное ускорение равны по модулю, а угол между ними составляет 90^0. Тангенциальное ускорение в первой половине пути направлено в противоположную сторону от направления полной скорости, т.е. движение равнозамедленное, во второй части пути, тангенциальное ускорение совпадает по направлению с вектором скорости и движение – равноускоренное.

Из формулы (3.1) можно рассчитать радиус кривизны траектории в верхней точке и в начале пути. Видно, что радиус кривизны принимает максимальное значение в начале (и в конце) движения тела, а минимальное значение в точке максимума, когда $a_n = g$.

Примеры, приведенные на рис. 3.6 – 3.8, показывают, что задача о движении тела брошенного под углом к горизонту, позволяет изучить весь раздел кинематики движения материальной точки, а программа *SMath Studio,* позволяет процесс изучения сделать более наглядным и доступным для восприятия.

Для завершения изучения процесса движения тела брошенного под углом к горизонту, рассмотрим задачу с учетом силы сопротивления воздуха (рис. 3.9).

Движение тела брошенного под углом к горизонту
с учетом сопротивления воздуха

$g := 9,8 \qquad \alpha := \frac{\pi}{4} \qquad v_0 := 20 \qquad V_{x0} := v_0 \cdot \cos(\alpha) \qquad V_{y0} := v_0 \cdot \sin(\alpha)$

$m := 1 \qquad k := 0,1 \qquad v_0 := \sqrt{V_{x0}^{\ 2} + V_{y0}^{\ 2}} = 20$

$F \equiv m \cdot g + F_{сопр} \qquad a_x \equiv -\frac{k}{m} \cdot v_x \qquad \begin{cases} \frac{dr}{dt} \equiv v \\ \frac{dv}{dt} \equiv \frac{F}{m} \end{cases}$

$F_{сопр} \equiv -k \cdot v \qquad a_y := -g - \frac{k}{m} \cdot v_y$

$$Z := \begin{bmatrix} 0 \\ 0 \\ V_{x0} \\ V_{y0} \end{bmatrix} \qquad D(t; S) := \begin{bmatrix} S_3 \\ S_4 \\ -\frac{k \cdot S_3}{m} \\ -g - \frac{k \cdot S_4}{m} \end{bmatrix} \qquad D1(t; S) := \begin{bmatrix} S_3 \\ S_4 \\ 0 \\ -g \end{bmatrix}$$

$U := \text{rkfixed}(Z; 0; 3; 10; D(t; S)) \qquad U0 := \text{rkfixed}(Z; 0; 3; 10; D1(t; S))$

$$U = \begin{bmatrix} 0 & 0 & 0 & 14,1421 & 14,1421 \\ 0,3 & 4,1796 & 3,743 & 13,7242 & 10,8278 \\ 0,6 & 8,2357 & 6,5065 & 13,3186 & 7,6115 \\ 0,9 & 12,172 & 8,3194 & 12,9249 & 4,4902 \\ 1,2 & 15,9919 & 9,2098 & 12,5429 & 1,4612 \\ 1,5 & 19,6989 & 9,205 & 12,1722 & -1,4784 \\ 1,8 & 23,2963 & 8,3315 & 11,8125 & -4,331 \\ 2,1 & 26,7874 & 6,6149 & 11,4634 & -7,0994 \\ 2,4 & 30,1754 & 4,0801 & 11,1246 & -9,7859 \\ 2,7 & 33,4632 & 0,7513 & 10,7958 & -12,393 \\ 3 & 36,6538 & -3,348 & 10,4768 & -14,9231 \end{bmatrix} \qquad U0 = \begin{bmatrix} 0 & 0 & 0 & 14,1421 & 14,1421 \\ 0,3 & 4,2426 & 3,8016 & 14,1421 & 11,2021 \\ 0,6 & 8,4853 & 6,7213 & 14,1421 & 8,2621 \\ 0,9 & 12,7279 & 8,7589 & 14,1421 & 5,3221 \\ 1,2 & 16,9706 & 9,9146 & 14,1421 & 2,3821 \\ 1,5 & 21,2132 & 10,1882 & 14,1421 & -0,5579 \\ 1,8 & 25,4558 & 9,5798 & 14,1421 & -3,4979 \\ 2,1 & 29,6985 & 8,0895 & 14,1421 & -6,4379 \\ 2,4 & 33,9411 & 5,7171 & 14,1421 & -9,3779 \\ 2,7 & 38,1838 & 2,4628 & 14,1421 & -12,3179 \\ 3 & 42,4264 & -1,6736 & 14,1421 & -15,2579 \end{bmatrix}$$

Рис. 3.9. Движение тела брошенного под углом к горизонту с учетом сопротивления воздуха

Как и в предыдущих примерах, камень брошен под углом 45^0 к горизонту со скоростью 20 м/с (будим решать задачу в безразмерных величинах). Находим проекции скорости на координатные оси в начальный мо-

мент времени и проверяем результат.

Пусть камень весит 1 *кг*, а коэффициент сопротивления воздуха k = 0.1. Как видим, кинематическая задача превратилась в динамическую. Запишем второй закон Ньютона с учетом силы сопротивления. Сила сопротивления пропорциональна скорости движения тела со знаком минус (сила сопротивления приводит к уменьшению скорости движения). Решать задачу будем в прямоугольных декартовых координатах. Распишем проекции ускорений на оси *OX* и *OY*. Видно, что учет сопротивления воздуха приводит к тому, что движение по оси *OX* будет равнозамедленным. Второй закон Ньютона (или уравнение движения) является дифференциальным уравнением второго порядка $F = ma = m\frac{dv}{dt} = m\frac{d^2r}{dt^2}$, относительно радиус-вектора r (или пространственных координат x и y). В математике, для решения линейных дифференциальных уравнений высоких порядков, существует процедура понижения порядка дифуравнений, в результате получаем систему дифференциальных уравнений первого порядка (см. рис. 3.9). В результате решения этой системы дифференциальных уравнений мы получим необходимые нам координаты $x(t)$, $y(t)$ и скорости $v_x(t)$ и $v_y(t)$. Дифференциальные уравнения имеют бесконечное число решений, для выбора нужного решения необходимо задать начальные условия. В нашем случае назальными координатами является начало координат x = 0, y = 0, а скорости имеют начальные значения $v_{x0}(t)$ и $v_{y0}(t)$. В программе начальные значения задаются в виде вектора-столбца *Z*.

Для численного решения системы дифференциальных уравнений используется встроенная функция ***rkfixed()***, которая реализует метод Рунге-Кутта четвертого порядка с фиксированным шагом. Это сравнительно простой, быстрый и универсальный численный метод решения дифференциальных уравнений. Используется функция ***rkfixed(X0; t0; tmax; N; D(t; S))*** с пятью аргументами: *X0* – начальные условия, *t0* – начальное время, *tmax* – конечное время, *N* – число точек по времени, *D(t; S)* – правая часть решаемой системы уравнений. Составим матрицу *D(t; S),* в нашем случае это скорости $v_x(t)$ и $v_y(t)$, которые будут стоять в 4 и 5 столбцах матрицы *S* и ускорения $a_x(t)$ и $a_y(t)$ (несоответствие индексов возникло из-за заимствования функция ***rkfixed()*** из пакета MathCad, в

котором индексы, по умолчанию, начинаются с нуля).

Результаты расчетов приведены на рис. 3.10. Для проверки правильности работы программы приведены расчеты полета камня без учета сил трения (входная матрица *D1(t; S)* и результат расчета *U0*). Разберем структуру матрицы *U (U0)* с результатами решения системы дифференциальных уравнений. Выходная матрица имеет размер 10x5, 10 – точек по времени, а в пяти колонках размещены: точки по времени; координаты $x(t)$, $y(t)$ и скорости $v_x(t)$, $v_y(t)$. Как и следовало ожидать, скорость $v_x(t)$ в матрице *U0* не меняется (т.к. сопротивление отсутствует).

Для более наглядного представления результатов расчета построим график траектории полета камня в случае наличия и отсутствия сопротивления воздуха (рис. 3.10).

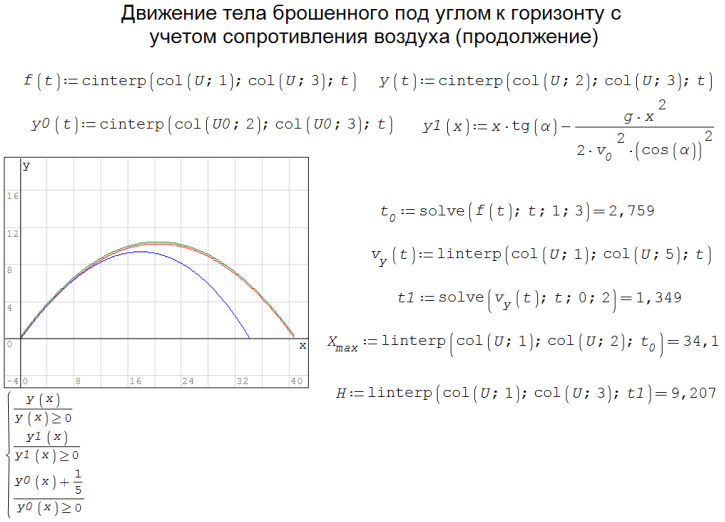


Рис. 3.10. Движение тела с учетом сопротивления воздуха (окончание)

При задании параметров было выбрано сравнительно небольшое число точек по времени – всего 10. Для того чтобы график был достаточно гладким проведем интерполяцию полученных результатов, т.е. по результатам заданным в 10 точкам проведем гладкую линию. Эта очень часто встречающаяся в науке и техника процедура называется интерполяцией. В данном примере используем встроенные функции ***cinterp()*** и ***linterp()***, которые реализуют кубическую и линейную интерполяции функций, заданных в табличном виде. Функция ***cinterp()*** соединяет соседние точки полиномом третьего порядка, а ***linterp()*** соединяет соседние точки полиномом первого порядка (отрезком прямой). Для построе-

ния графика траектории полета камня сформируем функцию *y(t)* = ***cinterp(col**(U; 2); **col**(U; 3); t)* проводя кривую полинома третьего порядка через рассчитанные абсциссы и ординаты, которые хранятся во 2 и 3 столбцах матицы *U*. Аналогично, строим график $y_0(x)$ – траектория полета камня без учета сопротивления среды и функцию $y_1(x)$ из предыдущего примера. Графики $y_0(x)$ и $y_1(x)$ полностью совпадают (что еще раз доказывает адекватность нашего расчета), поэтому к графику $y_0(x)$ прибавлено 0.2, чтобы красный и зеленый графики можно было различить.

Долее, рассчитываем время падения камня при учете сопротивления воздуха, как второй корень функции скорости по оси *OY* от времени *f(t)* = ***cinterp(col**(U; 1); **col**(U;3); t)* и t_0 = ***solve**(f(t); x*; 1; 3). Сравнивая время полета камня со значением из рис. 3.6 t_0 = 2.89 *c* (без учета сопротивления воздуха) и t_0 = 2.76 (с учетом сопротивления), замечаем, что во втором случае тело быстрее падает на землю.

Следующим расчетом является определение времени поднятия камня на максимальную высоту. Для этого с помощью линейной интерполяции строится функция $v_y(t)$= ***cinterp(col**(U; 1); **col**(U;5); t)*, а с помощью функции ***solve()*** рассчитывается время подъема камня на максимальную высоту t_1 = ***solve**(f(t); x*; 0; 2). Также определяется максимальное расстояние, на которое улетит камень и максимальная высота подъема камня. Можно заметить, что учет сопротивления воздуха приводит к уменьшению как X_{max}, так и H_{max}.

Данный пример показывает, что программа *SMath Studio* позволяет проводить моделирование достаточно сложных физических процессов, а результат моделирования представлять в наглядном виде.

Читателю рекомендуется провести моделирование движения тела, с различными углами ·, скоростями v_0 и различными коэффициентами сопротивления *k*.

3.3. Колебания и волны. Собственные колебания. Сложение колебаний

В окружающем нас мире многие процессы являются или периодически повторяющимися или повторяющимися с некоторыми изменениями. Яркими примерами повторяющихся (или их еще называют колебательных) процессов являются волны на море, колебание веток дерева, коле-

бания проводов, колебания маятника часов и т.п.. Колебательные процессы можно разделить на периодические и непериодические. Периодические процессы это такие процессы, которые повторяются через постоянный промежуток времени *T*, который называемый периодом колебаний. Самыми простыми по форме, но одновременно, самыми востребованными в науке и технике являются гармонические колебания, т.к. сложные колебательные процессы можно свести к сумме определенного числа гармонических колебаний, которые отличаются периодом, амплитудой и начальной фазой. Гармонические колебания это такие колебания, период которых не зависит от амплитуды колебаний и происходит по гармоническому закону (закону синуса, косинуса или экспоненты с мнимой степенью): $y(t) = A\sin(\omega \cdot t + \varphi), y(t) = A\cos(\omega \cdot t + \varphi), (t) = Ae^{-i(\omega \cdot t + \varphi)}$, где *A* – амплитуда колебаний, ω – круговая частота, φ – фаза колебаний.

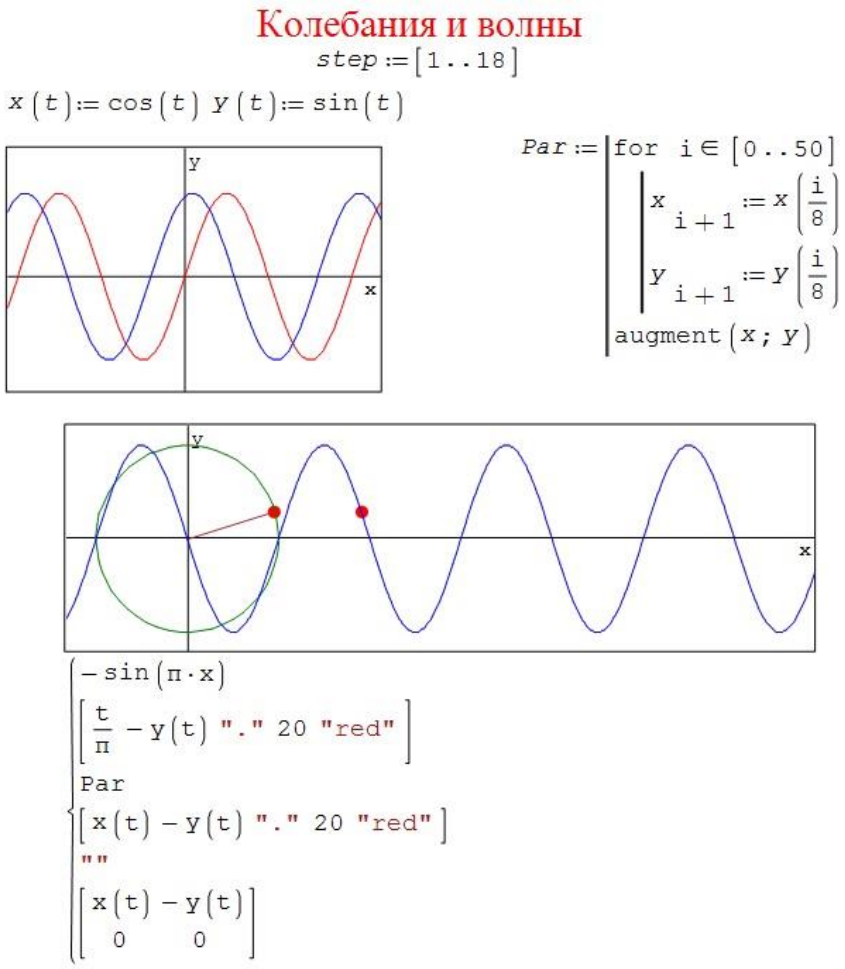


Рис. 3.12. Графическое представление колебательного и волнового процессов ($\omega = 1, \varphi_0 = 0$)

Колебания можно разделить на свободные и вынужденные, затухающие, автоколебания, параметрические колебания и т.д. Колебательный процесс, распространяющийся в пространстве, называется волновым процессом или волной.

Гармонические колебания проще всего рассмотреть на примере равномерного вращения точки по окружности с радиусом равным амплитуде колебаний, тогда точка на окружности определяет фазу ($\omega t + \varphi$), т.е. положение точки на синусоиде, которая движется с угловой скоростью ω, как по окружности, так и вдоль синусоиды (рис. 3.12). Меняя значение круговой частоты и фазы можно наблюдать различные колебательные (волновые) процессы.

Сложение колебаний

Пусть даны два одинокого направленных колебания заданных по законам:

$$Y_1(t) = A_1 \cos(\omega_1 + \varphi_1),$$
$$Y_2(t) = A_2 \cos(\omega_1 + \varphi_2),$$

где A_1 и A_2 – амплитуды колебаний, ω_1 – частота колебаний (одинакова для обоих колебаний), φ_1 и φ_2 – фазы колебаний.

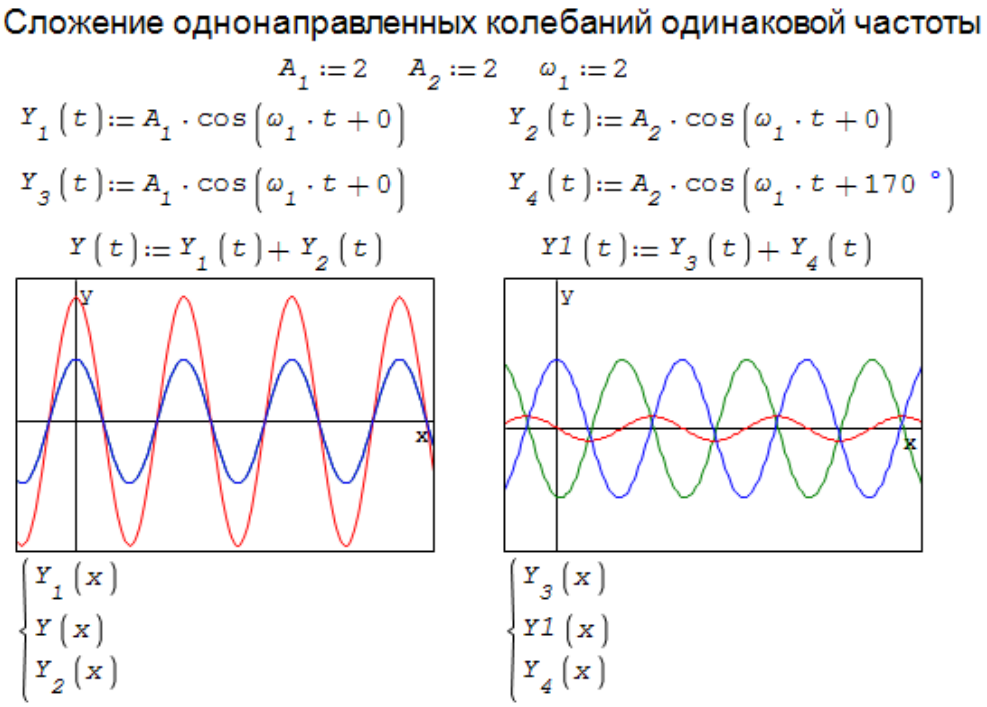


Рис. 3.13. Сложение однонаправленных колебаний

На рис. 3.13 показан результат сложения колебаний, когда разность фаз составляет $\varphi_1 - \varphi_2 = 0^0$ и когда $\varphi_1 - \varphi_2 = 170^0$.

Меняя фазу для каждого колебания можно заметить, что когда фазовый сдвиг между однонаправленными колебаниями с равной частотой равен нулю, происходит наложение колебаний с возрастанием амплитуды, если разность фаз приближается к 180^0 (или π радиан), колебания происходят в противоположные стороны, и суммарная амплитуда уменьшается до нуля, т.е. колебания компенсируют друг друга.

Если частота обоих колебаний немного отличается одна от другой, происходит модуляция колебаний или начинает проявляться явление возникновения биений колебаний. Явление биений можно наблюдать с помощью программы, приведенной на рис. 3.14.

Биения

$A_1 := 2 \quad A_2 := 2 \quad \omega_1 := 2 \quad \omega_2 := 2,1 \quad \varphi_1 := 0 \quad \varphi_2 := 0$

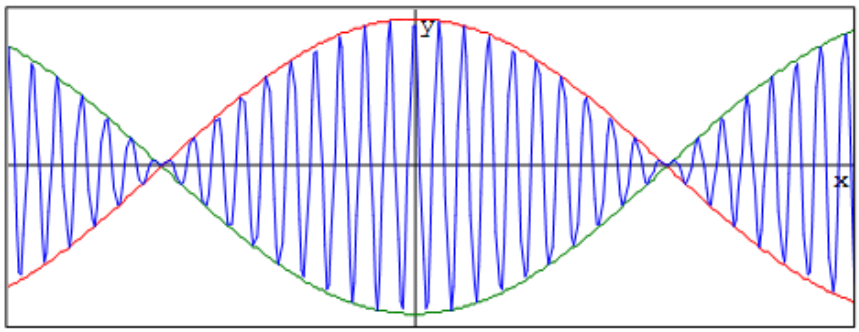
$Y_1(t) := A_1 \cdot \cos(\omega_1 \cdot t + \varphi_1) \qquad Y_2(t) := A_2 \cdot \cos(\omega_2 \cdot t + \varphi_2)$

$\varphi := \frac{(\varphi_2 + \varphi_1)}{2} \qquad A := A_1 + A_2 \qquad \omega := \frac{(\omega_2 + \omega_1)}{2}$

$\Delta\varphi := \frac{\varphi_2 - \varphi_1}{2} \qquad \Delta\omega := \frac{\omega_2 - \omega_1}{2}$

$YB1(t) := A \cdot \cos(\Delta\omega \cdot t + \Delta\varphi) \cdot \cos(\omega \cdot t + \varphi)$

$YB(t) := A \cdot \cos(\Delta\omega \cdot t + \Delta\varphi)$



$\begin{cases} Y_1(x) + Y_2(x) \\ \pm YB(x) \end{cases}$

Рис. 3.14. Сложение однонаправленных колебаний с близкими частотами (биения)

При сложение перпендикулярно направленных колебаний с частотами отличающимися в кратное число раз, получаем так называемые фигуры Лиссажу, которые легко визуализировать в пакете *SMath Studio,* строя график суммы функций x(t) – колебания относительно оси OX и y(t) – колебания относительно оси OY, в параметрическом виде (рис. 3.15). Из рисунка видно, что задавая частоты колебаний и начальные фазы, можно построить график, состоящий из 91 точки, которые содержатся в двумерном массиве Par.

Читателю предлагается самостоятельно построить фигуры Лиссажу при других начальных условиях.

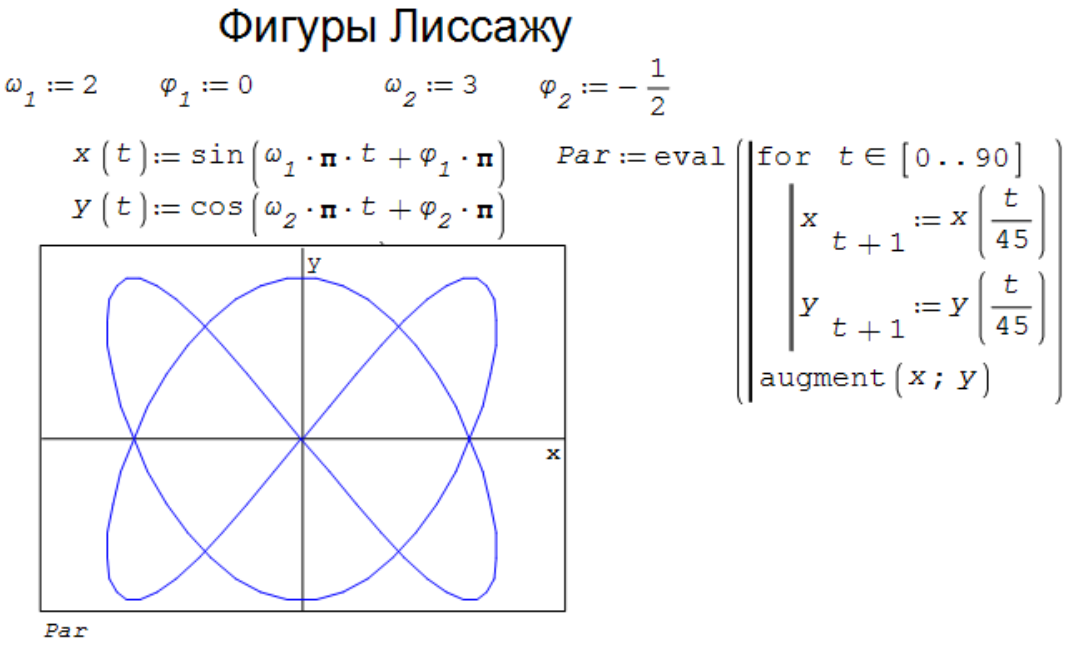


Рис. 3.15. Сложение перпендикулярных колебаний (фигуры Лиссажу)

3.4. Правила Кирхгофа

Законы (или правила Кирхгофа) занимают важное место при изучении «Физики» и «Электротехники». Покажем, как пакет *SMath Studio* позволяет упростить усвоение материала и решение задач на данную тему. Чтобы рассчитать неизвестные токи, протекающие в сложной цепи необходимо использовать правила Кирхгофа. Порядок действий при решении задач такой:

1. Нарисовать схему цепи. При необходимости провести ее упрощение, путем учета параллельного и последовательного соединения

элементов схемы. На схеме выбрать и показать направления токов на всех участках цепи (при этом проследить, чтобы во всех узлах выполнялось первое правила Кирхгофа, т.е. токи должны как входить в узел, так и выходить из него).

2. Используя первый закон Кирхгофа можно написать (k – 1) уравнений, где k - число узлов в рассматриваемой цепи. Для того чтобы система линейных алгебраических уравнений была разрешима, число уравнений должно совпадать с числом неизвестных. Если задаче необходимо определить n – неизвестных, дополним систему уравнений, используют второе правило Кирхгофа.

3. Выбрать замкнутые контуры обхода для применения второго закона Кирхгофа. Показать на рисунке направление обхода по контуру. Необходимо выбрать n – (k – 1) независимых контуров обхода. Контуры необходимо выбирать так, чтобы в каждый новый контур входил хотя бы один участок цепи, которого бы не было ни в одном из ранее рассмотренных контуров.

4. Воспользоваться вторым законом Кирхгофа, при этом надо учитывать следующее правило знаков: падение напряжения на каждом участке записывается со знаком «+», если направление обхода по этому участку совпадает с направлением тока на нем. И наоборот, если обход совершается противоположно направлению тока, то ставится знак «–». ЭДС записывается со знаком «+» в том случае, когда направление обхода совпадает с направлением поля сторонних сил в источнике тока и наоборот. Поле сторонних сил внутри источника всегда направлено от отрицательного полюса к положительному.

5. Решить полученную систему уравнений и найти искомые величины. В результате решения полученной системы уравнений, определяемые величины могут получаться отрицательными. Отрицательное значение тока указывает на то, что фактическое направление тока на данном участке цепи обратно тому, которое мы выбрали.

6. Провести проверку решения задачи, используя правило баланса мощностей. Необходимо проверить. чтобы мощности выделяющиеся на каждом активном элементе цепи были равны мощностям, вырабатываемым источниками питания.

Пример решения задачи на применение законов Кирхгофа приведен на рис. 3.16. Дана схема электрической цепи, в которой содержится семь

резисторов и четыре элемента питания (источника ЭДС), необходимо найти все токи, протекающие по элементам цепи. Записываем условие задачи, в котором заданы все сопротивления и ЭДС. Необходимо найти семь неизвестных токов.

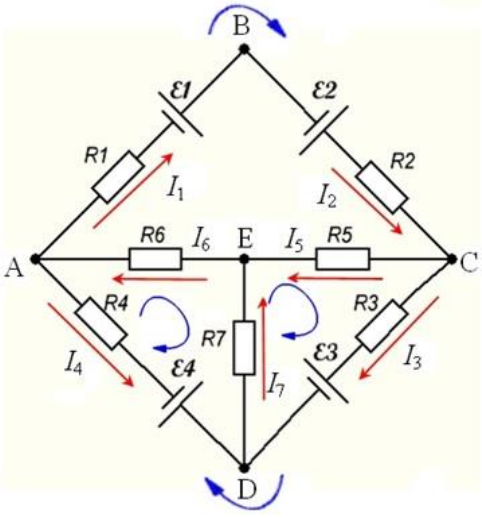
Для решения задачи используем встроенную функцию ***roots()***, с двумя параметрами, первый из которых содержит семь уравнений записанных в явном виде, а второй вектор, состоящий из семи решений, которые необходимо найти.

В соответствие со схемой, на которой расставлены направления токов (пока произвольным образом) и правила обхода, составляем систему из семи уравнений. Первые четыре (5 – 1) уравнения для четырех узлов (A, B, C и D) строятся первому правилу Кирхгофа, по которому алгебраическая сумма токов в узле равна нулю, или другими словами, сумма токов входящих в узел равна сумме токов выходящих из него.

Применение правил Кирхгофа

$R_1 := 100$
$R_2 := 100$
$R_3 := 200$
$R_4 := 200$
$R_5 := 100$
$R_6 := 100$
$R_7 := 200$

$E_1 := 6$
$E_2 := 6$
$E_3 := 9$
$E_4 := 8$



$$I = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \end{bmatrix}$$

$$I := \text{roots}\left(\begin{bmatrix} -I_1 - I_4 + I_6 = 0 \\ I_1 - I_2 = 0 \\ I_2 - I_3 - I_5 = 0 \\ I_5 - I_6 + I_7 = 0 \\ I_1 \cdot R_1 + I_2 \cdot R_2 + R_3 \cdot I_3 - R_4 \cdot I_4 = E_1 + E_2 - E_3 - E_4 \\ -R_4 \cdot I_4 - R_6 \cdot I_6 - R_7 \cdot I_7 = -E_4 \\ R_3 \cdot I_3 - I_5 \cdot R_5 + R_7 \cdot I_7 = -E_3 \end{bmatrix}; \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \end{bmatrix}\right) \text{A} \qquad I = \begin{bmatrix} 0,019\ \text{A} \\ 0,019\ \text{A} \\ -0,023\ \text{A} \\ 0,021\ \text{A} \\ 0,042\ \text{A} \\ 0,04\ \text{A} \\ -0,001\ \text{A} \end{bmatrix}$$

Рис. 3.16. Решение электротехнической задачи с использованием правил Кирхгофа

Токи, входящие в узел, берутся с положительным знаком, а выходящие из узла с отрицательным.

Таким образом, получены четыре уравнения, для дополнения системы до семи, необходимо воспользоваться вторым правилом Кирхгофа, которое гласит, что сумма падений напряжения на элементах контура (или произведение силы тока на напряжение на резисторе) равно сумме ЭДС в контуре (с учетом направления обхода и полярностью источника питания). Нами выбран самый большой контур, расположенный по периметру схемы и два нижних контура. За положительное направление выбрано направление по часовой стрелки. При нажатии клавиши «=«, получает ответ нашей задачи. Недостатком решения задачи по электротехнике с использованием встроенной функции ***roots()***, невозможность использовать единицы изменений. Частично, преодолеть данный недостаток можно с помощью приема показанного внизу рис. 3.16. Присваивая корни уравнения вектору I и назначая размерность в Амперах, получаем токи в привычной размерности.

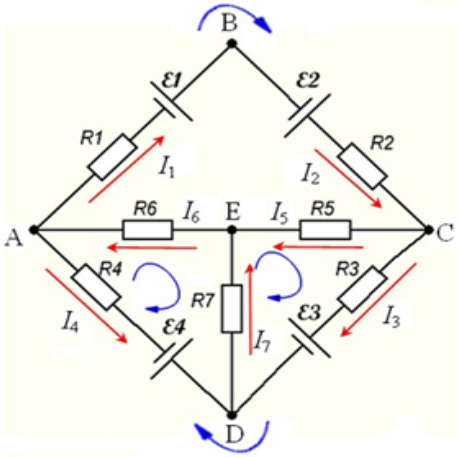
Долее следует провести проверку решения задачи методом баланса мощностей вырабатываемых источниками питания и потребляемыми (выделяемыми) резисторами. Данный расчет приведен на рис. 3.17.

Присваивая единицы измерений сопротивлению и ЭДС, рассчитываем мощности выделяемые на резисторах, по закону Джоуля-Ленца $P = I^2R$, получаем определенной значение. Затем вычисляем мощность, выделяемую источниками питания по формуле $P = EI$, учетом направления тока I в контуре и направления тока вырабатываемого источником питания (от плюса к минусу). Если значения мощностей совпадают, значит, задача решена правильно.

Дальнейшая формализация решение задач электротехники в виде матричного уравнения приведены в нижней части рис. 3.17.

Применение правил Кирхгофа (продолжение)

$$R := \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix} \text{ Ом} \qquad E := \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{bmatrix} \text{ В}$$



$$P = I \cdot U \qquad P = \frac{U^2}{R} \qquad P = I^2 \cdot R$$

$$I_1^2 \cdot R_1 + I_2^2 \cdot R_2 + I_3^2 \cdot R_3 + I_4^2 \cdot R_4 + I_5^2 \cdot R_5 + I_6^2 \cdot R_6 + I_7^2 \cdot R_7 = 0,603 \text{ Вт}$$

$$+$$

$$I_1 \cdot E_1 + I_2 \cdot E_2 - I_3 \cdot E_3 + \left(-I_4\right) \cdot \left(-E_4\right) = 0,603 \text{ Вт}$$

Матричный метод применения правил Кирхгофа

$$R \cdot I = U$$

$$R := \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \\ R_1 & R_2 & R_3 & -R_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -R_4 & 0 & -R_6 & -R_7 \\ 0 & 0 & R_3 & 0 & -R_5 & 0 & R_7 \end{bmatrix} \text{ Ом} \qquad U := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ E_1 + E_2 - E_3 - E_4 \\ -E_4 \\ -E_3 \end{bmatrix} \text{ В}$$

$$A \cdot x = b \qquad A^{-1} \cdot A \cdot x = A^{-1} \cdot b$$

$$A^{-1} \cdot A = 1 \qquad x = A^{-1} \cdot b$$

$$I := R^{-1} \cdot U = \begin{bmatrix} 0,019 \text{ А} \\ 0,019 \text{ А} \\ -0,023 \text{ А} \\ 0,021 \text{ А} \\ 0,042 \text{ А} \\ 0,04 \text{ А} \\ -0,001 \text{ А} \end{bmatrix}$$

Рис. 3.17. Метод баланса мощностей. Матричное представление метода Кирхгофа

В данном примере формируется матрица коэффициентов перед неизвестными токами, обозначим ее буквой *R*, а правые части исходной системы уравнений обозначим буквой *U*. Формально это можно сделать, умножая левые и правые части первых четырех уравнений на 1*Ом*. В результате получается формула закона Ома в матричной форме. Для решения этого матричного уравнения можно использовать метод обратной матрицы, прекрасно реализованный в программе *SMath Studio*. В резуль-

тате получаем искомые значения силы тока.

Сказанное, наглядно показывает, что и использование законов Ома и правил Кирхгофа позволяет решать электротехнические задачи любого уровня сложности.

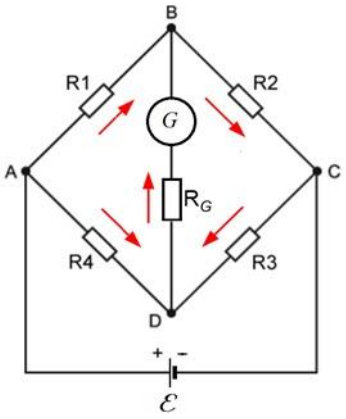
Следующий пример посвящен изучению мостовой схемы, которая находим широкое практическое использования благодаря своей простоте, дешевизне и эффективности. Так в измерительной технике, в электронике, в системах управления на основе мостовых строятся наиболее эффективные технические устройства.

На рис. 3.18 приведен расчет токов протекающих в мостовой схеме, использую правила Кирхгофа. Как и в предыдущем примере, записываем пять линейных уравнений (по числу резисторов в цепи). Первые два на основе первого правила Кирхгофа для точек B и D, а три последующие, по второму правилу Кирхгофа, используя верхнюю часть схемы (R1, R2 и источник питания E), нижнюю (R4, R3 и E) и правую часть (R1, R4 и RG), соответственно. Правила обхода принимаем по часовой стрелке. В результате решения системы уравнений получаем токи во всех ветвях мостовой схемы. В примере специально выбрано сопротивление R1 = 102 Ом, которое незначительно отличается от R2 = 100 Ом. В данном случае ток, протекающий через гальванометр (G) составляет 13.8 мкА. Если через гальванометр (в мостовой схеме выполняющий роль индикатора нуля) протекает ток, мостовая схема называется разбалансированной. Хотя, формально, мы решили электротехническую задачу, принцип действия, расчетные формулы и характеристики мостовой схемы и, более широко, мостового измерительного метода остаются не выясненными.

Мостовая схема

$R_1 := 102$
$R_2 := 100$
$R_3 := 200$
$R_4 := 200$
$R_G := 2000$
$E := 6$

IG – ?



$$I := \text{roots}\left(\begin{bmatrix} I_1 - I_2 + I_G = 0 \\ I_3 + I_4 - I_G = 0 \\ R_1 \cdot I_1 + R_2 \cdot I_2 = E \\ -R_3 \cdot I_3 + R_4 \cdot I_4 = E \\ R_1 \cdot I_1 - R_G \cdot I_G - R_4 \cdot I_4 = 0 \end{bmatrix}; \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_G \end{bmatrix}\right) = \begin{bmatrix} 0{,}02969613 \\ 0{,}02970994 \\ -0{,}01499309 \\ 0{,}01500691 \\ 0{,}00001381 \end{bmatrix}$$

$$I_G := I_2 - I_1 = 1{,}3812 \cdot 10^{-5}$$

(аналитический расчет)

$$A := \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & -1 \\ R1 & R2 & 0 & 0 & 0 \\ 0 & 0 & -R3 & R4 & 0 \\ R1 & 0 & 0 & -R4 & -RG \end{bmatrix} \qquad B := \begin{bmatrix} 0 \\ 0 \\ E \\ E \\ 0 \end{bmatrix}$$

$$\Delta := \text{maple}(|A|)$$

$$A5 := \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ R1 & R2 & 0 & 0 & E \\ 0 & 0 & -R3 & R4 & E \\ R1 & 0 & 0 & -R4 & 0 \end{bmatrix}$$

$$\Delta 5 := \text{maple}(|A5|) = E \cdot (-R2 \cdot R4 + R1 \cdot R3)$$

$$I_G := \text{maple}\left(simplify\left(\frac{\Delta 5}{\Delta}\right)\right)$$

$$I_G = -\frac{E \cdot (R2 \cdot R4 - R1 \cdot R3)}{R2 \cdot (RG \cdot (R4 + R3) + R4 \cdot (R3 + R1) + R1 \cdot R3) + R1 \cdot (RG \cdot (R4 + R3) + R3 \cdot R4)}$$

Рис. 3.18. Работа мостовой схемы численный расчет и аналитический вывод рабочей формулы

Для того, чтобы разобраться с принципом действия мостовой схемы необходимо вывести формулу зависимости тока протекающего через гальванометр от сопротивлений R1, R2, R3, R4 и ЭДС источника питания E, в явной аналитической форме. Как видно из рис. 3.18 система состоит из пяти линейных уравнений. Для решения данного уравнения в общем виде можно использовать метод Крамера или метод определителей. По данному методу корни СЛАУ вычисляются по формуле $x_i = \frac{\Delta_i}{\Delta}$, где Δ – основной определитель матрицы *A*, а вспомогательные определители Δ_i – строятся путем замены столбца i на столбец *B* правой части СЛАУ (см. рис. 3.18). Метод Крамера очень трудоемкий, немного спасает то, что нам необходим только один корень, а именно, ток I_G (последний пятый корень).

Для нахождения тока через гальванометр воспользуемся аналитическими возможностями, которые обеспечивает дополнение Maple к программе *SMath Studio*. Вывод аналитического выражения для тока, протекающего через гальванометр мостовой схемы, приведен на рис. 3.19.

Формируем основную матрицу *A*, интересующей СЛАУ, вспомогательную матрицу A5, в которой заменяем пятый столбец матрицы *A* на столбец свободных членов B (правую часть СЛАУ). Рассчитываем определители пятого порядка Δ и $\Delta 5$. «Ручной» расчет определителей порядка выше трех крайне затруднителен, а в программе *SMath Studio* эти аналитические преобразования проводятся достаточно просто. Долее выражаем ток через гальванометр как упрощенное выражение $\Delta/\Delta 5$. Аналитическое упрощение алгебраического выражения осуществляется с помощью процедуры ***simplify()***.

В результате получаем окончательное выражение для I_G. Выражение сложное, но основные особенности уже легко заметить. Первое что необходимо запомнить это условие балансировки моста

$$R_1 R_3 - R_2 R_4 = 0 \quad \Rightarrow \quad R_1 = \frac{R_2 R_4}{R_3}. \tag{3.2}$$

Ток через гальванометр измерительного моста не течет когда выполняется данное выражение. Запомнить это условие просто – произведение сопротивлений находящихся на противоположных диагоналях моста должно быть равно между собой. Из рис. 3.18 видно, что это выражение появляется при нахождении определителя $\Delta 5$.

Данное условие определяем простоту и относительную дешевизну мостового метода измерений. У нас имеется четыре резистора, один неизвестный (пусть будет R1), один эталонный (R2) и два произвольных вспомогательных (R3 и R4), а также источник питания и гальванометр. В качестве эталонного резистора обычно используется магазин сопротивлений, с помощью которого добиваются полной балансировки моста и по формуле (3.2) рассчитывают сопротивление неизвестного резистора R1.

Следует отметить, что мостовой метод позволяет определять не только сопротивления резисторов. Если в качестве питания использовать источник переменного напряжения, то мостовая схема позволяет определять реактивные параметры цепи – емкости и индуктивности.

Использование возможностей Maple и *SMath Studio* это хорошо, но мы

можем получить данную формулу аналитически методом эквивалентного генератора, который широко применяется для анализа сложных электротехнических и радиоэлектронных схем.

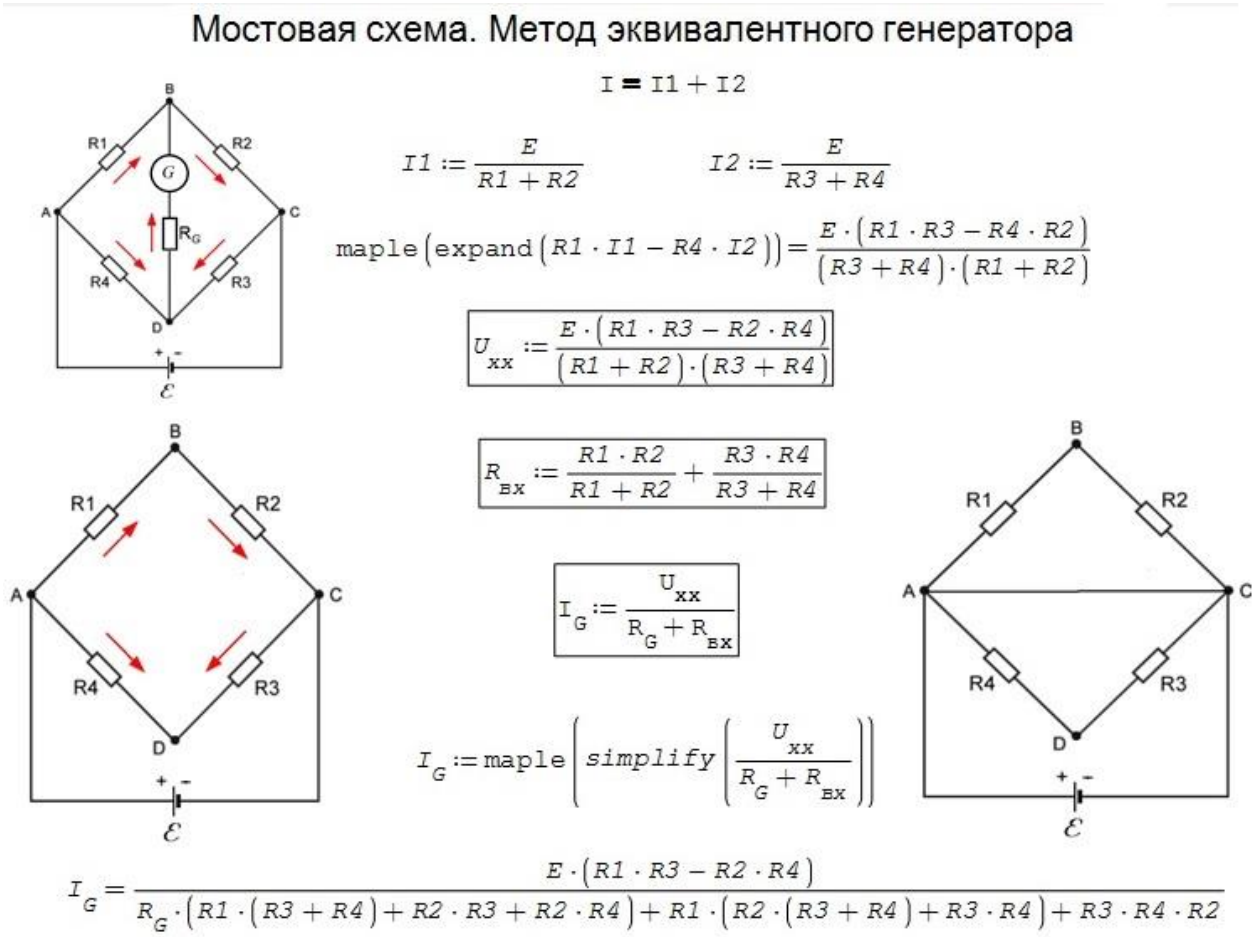


Рис. 3.19. Анализ работы мостовой схемы методом эквивалентного генератора

На рис. 3.19 приведен вывод уравнения мостовой схемы с использованием метода эквивалентного генератора. Для этого заменим реальную схему на две вспомогательные. В первом случае мы исключаем из схемы гальванометр. В этом случае, мы можем рассчитать так называемое напряжение «холостого хода» – разность потенциалов между точками B и D. Для этого записываем ток источника питания как сумму токов I1 и I2. Выражаем данные токи через сопротивления верхней и нижней части моста. И аналитически получаем потенциал между точками B и D как разность потенциалов в данных точках.

На следующем шаге, мысленно замыкая точки A и C, рассчитываем сопротивление между точками B и D. Теперь сопротивление точками B и

D равно сумме сопротивлений резисторов (R1 и R2), (R3 и R4), соединенных параллельно. Это так называемое «входное сопротивление» мостовой схемы. Теперь используя закон Ома для полной цепи, можем определить ток, протекающий через гальванометр. Проверяем выражение, получаем результат, совпадающий с формулой для IG на рис. 3.18 (необходимо перегруппировать слагаемые в знаменателе).

3.5. Фазо-импульсное управление

В заключении данного пособия рассмотрим задачу из электроники и теории управления мощностью электроустановок с использованием самого простого и популярного фазо-импульсного метода управления. Метод основан на выделении из синусоидального напряжения промышленной частоты участков определенной длительности. На рис. 3.20 показан принцип работы фазо-импульсного регулятора мощности. Промышленные источники имеют напряжение 220В и частоту синусоидального напряжения 50 Гц. Для того, чтобы передать нагрузке только половину мощности, необходимо включать нагрузку только в определенный момент времени. Для регулятора настроенного на включение нагрузки в момент, когда синусоида принимает максимальное значение, мощность, передаваемая в нагрузку составит 50%. Меняя момент включения можно плавно регулировать мощность электроустановок. Когда нагрузка включается в момент перехода синусоиды через ноль, мощность в нагрузке составит 100% и т.д.

Используя программу *SMath Studio* проведем пример моделирование процесса включения нагрузки с помощью простейшего симисторного регулятора мощности (рис. 3.21). Предположим, что мы хотим включить нагрузку на 50% мощности (k = 0.5). Для этого мы должны подать импульс на включение с задержкой равной ¼ периода колебаний переменного тока в сети.

В примере (рис. 3.21) показаны уравнение синусоиды, линейная частота 50 *Гц*, период колебаний напряжения в сети 20 *мс*, круговая частота 314.16 $с^{-1}$, и рассчитанный импульс на открытие равный 20/4 = *5 мс*. Исходная синусоида (красная линия), результирующая обрезанная синусоида (синяя линия) и открывающий импульс показаны на рисунке. Там же приведена электрическая схема простейшего симмисторного регуля-

тора мощности. Эта схема часто используется в быту для регулирования яркости свечения ламп накала, поэтому схема еще называется симисторным диммером.

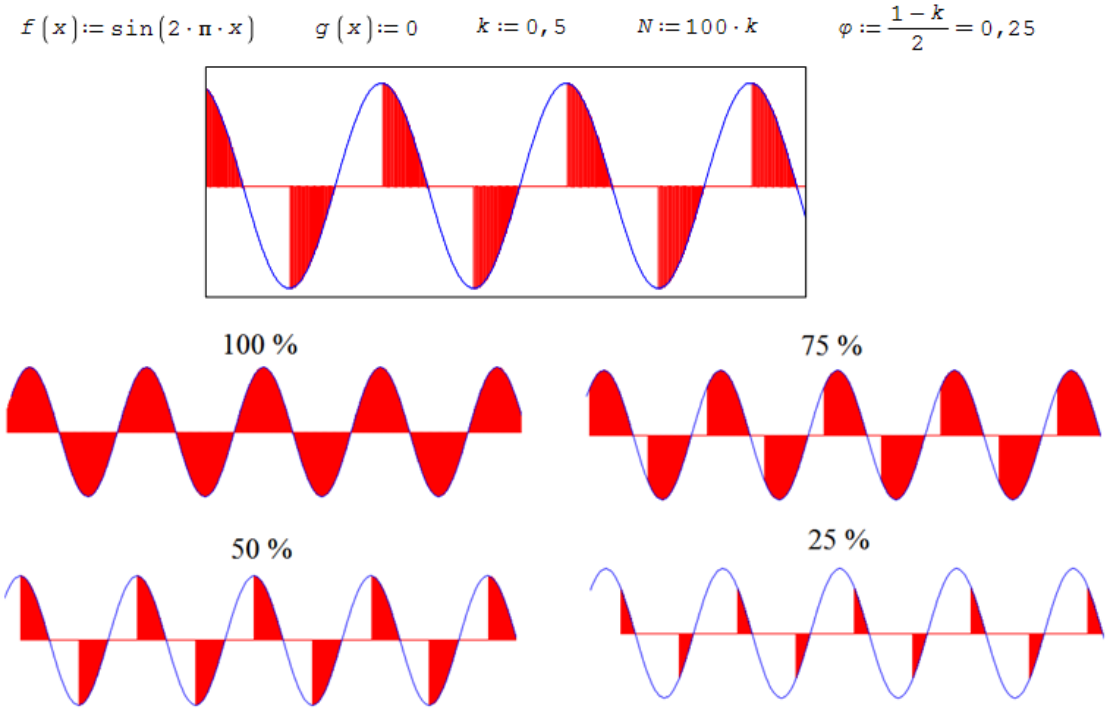


Рис. 3.20. Основы фазо-импульсного управления

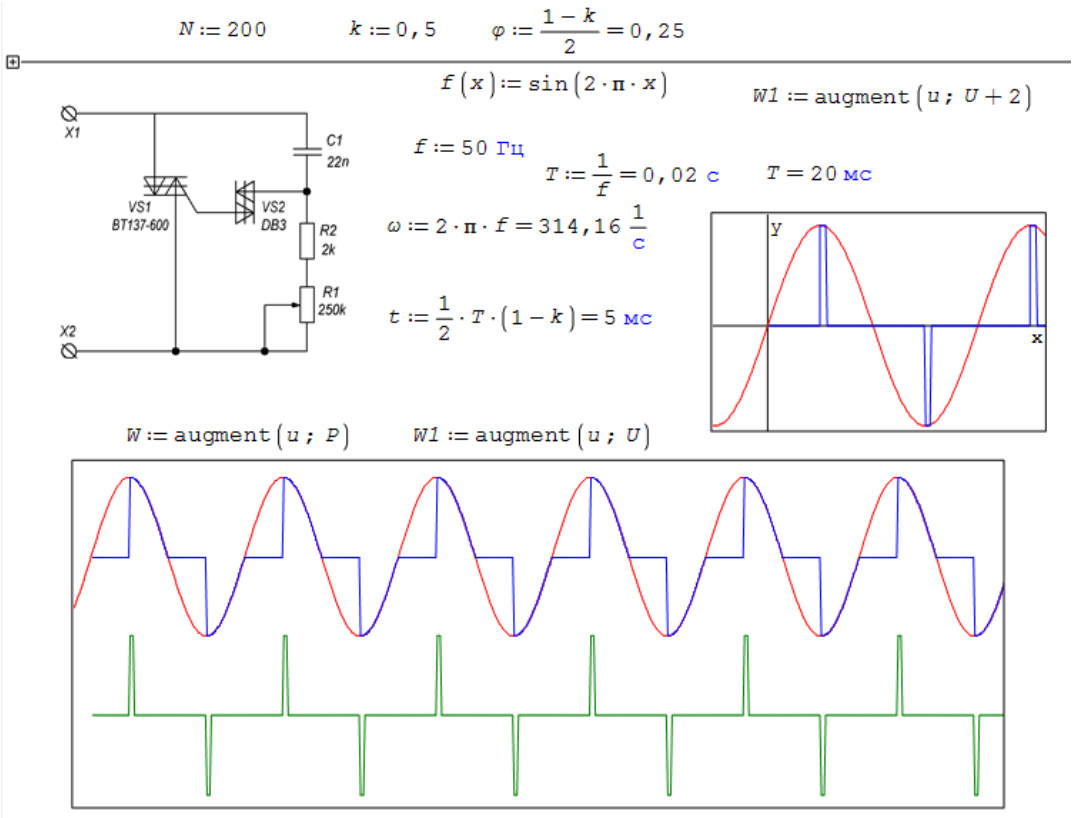


Рис. 3.21. Симисторная схема фазо-импульсного управление

Работает данная схема следующим образом, конденсатор *C*1 и резистор (*R*1+*R*2) образуют *RC*-цепочку с постоянной времени $\tau = RC$, через это время, конденсатор заряжается до напряжения при котором диак *VS*2 (диодный тиристор DB3) открывается, вызывая открытие мощного симистора *VS*1 (симметричный тиристор BT137-600) и напряжение подается на нагрузку. При очередном прохождении синусоиды через ноль симистор возвращается в закрытое состояние. Т.е. для регулировки мощности электроустановки необходимо регулировать время открытия (постоянную времени). Для мощности равной 50% и 100% это рассчитать легко. А если мы хотим получить произвольную мощность на нагрузке, приходится решать сложное нелинейное уравнение. И тут нам на помощь приходит *SMath Studio*.

На рис. 3.22 показан расчет постоянной времени, для рассмотренного уровня мощности 50%, это значение легко исправить на любое другое.

$$\int_0^{0,5} \sin(2\cdot\pi\cdot x)\,dx = 0,31831$$

$$A := \left| \text{solve}\left(\int_0^{a} \sin(2\cdot\pi\cdot y)\,dy = 0,31831\cdot 50\,\%;\ a;\ 0;\ 0,5 \right) \right|$$

$$A = 0,25$$

$$2\cdot\int_0^{A} \sin(2\cdot\pi\cdot x)\,dx = 0,31831$$

Рис. 3.22. Расчет времени открытия симистора

В первой строчке указана площадь кривой под верхней частью одного периода синусоиды. Нелинейное уравнение, которое необходимо решить записано во второй строчке. Используется функция ***solve()*** с пятью параметрами, два последних задают интервал поиска корней (т.к. синус функция периодическая). Уровень мощности определяется как часть площади под синусоидой, фазу включения на данном рисунке обозначено буквой A. Получаем для 50% мощности фазу A = 0.25. Проверка показы-

вает, что для достижения 50% мощности от синусоиды необходимо отрезать ровно половину.

Таким образом, мы показали, что использование программы *SMath Studio* позволяет решать задачи из самых разных областей знаний, простота, доступность программы должны помочь студентам при изучении различных учебных дисциплин.

Для успешного закрепления пройденного материала читателям предлагается с использованием бесплатной отечественной системы компьютерной математики SMath Studio провести моделирование, визуализацию и анимирование физико-технических процессов из нижеприведенного списка.

1. Анимация основных численных методов решения нелинейных уравнений.
2. Анимация основных численных методов решения линейных уравнений.
3. Анимация основных численных методов решения систем уравнений.
4. Моделирование эффекта Магнуса.
5. Запуск искусственных спутников Земли и расчет баллистических кривых.
6. Колебания тела на пружине в среде воздух–жидкость.
7. Колебания маятника с колеблющейся точкой подвеса.
8. Маятники Галилея и Гюйгенса.
9. Движения заряженной частицы в переменном электромагнитном поле.
10. Моделирование и анимация расчета параметров сложной электрической цепи.
11. Моделирование и анимирование решения уравнений математической физики.
12. Поведение плазмы в переменном магнитном поле.
13. Распространение света в толстой линзе. Оптические аберрации.
14. Распространение света в цилиндрической линзе.
15. Кольца Ньютона в отраженном и проходящем свете.
16. Дифракция света на щели и дифракционной решетке.
17. Дисперсия на призме.
18. Моделирование процесса теплового расширения твердых тел.
19. Модель деформации кристаллической решетки твердого тела.
20. Отыскание формы траектории с минимальным временем спуска.

Перечень использованных информационных источников

1. Официальный сайт SMath Studio URL: https://ru.smath.com/обзор/SMathStudio (дата обращения: 20.11.2022).

2. Теплотехнические расчеты на компьютере: учебное пособие / под редакцией Н.Д. Рогалева / М.: МЭИ, 2019.

3. Очков В.Ф., Богомолова Е.П., Иванов Д.А., Физико-математические этюды с Mathcad и Интернет: Учебное пособие. 2-е изд., испр. и доп. / СПб.: «Лань», 2018.

4. Очков В. Ф. MathCAD 14 для студентов и инженеров / В. Ф. Очков. – СПб. : БХВ-Петербург, 2009.

5. Кирьянов Д. В. Mathcad 15/Mathcad Prime 1.0. / СПб.: БХВ-Петербург, 2012.

6. Черняк А.А., Черняк Ж.А., Математические расчеты в среде Mathcad 3-е изд., испр. и доп. / М: Юрайт, 2021.

7. Черняк А.А., Черняк Ж.А., Доманова Ю.А., Высшая математика на базе Mathcad. Общий курс. / СПб.: БХВ-Петербург, 2004.

8. Bernard V Liengme, SMath for Physics: A primer. / Morgan & Claypool Publishers, 2015.

9. Thomas Geike, Rechenbuch Fluidtechnik mit Mathcad, SMath Studio und Python. / Books on Demand Verlag, 2020.