Федеральное государственное автономное образовательное учреждение высшего профессионального образования
Национальный исследовательский ядерный университет "МИФИ"

НОВОУРАЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ

Кафедра информатики и программирования

РЕШЕНИЕ ЗАДАЧ В СИСТЕМЕ MATHCAD

Методическое пособие по курсам «Информатика»,
«Решение инженерных задач на ЭВМ»,
«Вычислительные методы в инженерных расчетах»
для преподавателей и студентов всех специальностей
очной, очно-заочной форм обучения

Новоуральск 2013

УДК 681.3.06

МиМ – 2.3.- - 13

Автор Тихонова Евгения Валерьевна

Рецензент Николаев Николай Александрович
зав. кафедрой ИиП

Решение задач в системе MathCad. Методическое пособие по курсам «Информатика», «Решение инженерных задач на ЭВМ», «Вычислительные методы в инженерных расчетах» для преподавателей и студентов всех специальностей очной, очно-заочной форм обучения.

Новоуральск, НТИ НИЯУ МИФИ, 2013. - 75 с.

Пособие представляет собой описание способов решения математических задач с использованием программного продукта MathCad. Приводится множество примеров из различных областей математики.
Содержит 56 рисунков, 3 таблицы, 21 библиографических названий.

Пособие может использоваться при самостоятельном изучении системы MathCad.

Методическое пособие рассмотрено на заседании кафедры
Протокол № 5 от 28 мая 2013 г.

Зав.кафедрой Николаев Н.А.

СОГЛАСОВАНО:

Председатель методического совета
НТИ НИЯУ МИФИ д.т.н., профессор Беляев А.Е.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Среди пакетов прикладных программ, предназначенных для выполнения математических расчетов на компьютере, одно из ведущих мест занимает система MathCad. Она включает разнообразные средства визуализации расчетных данных и обладает наглядностью записи математических формул, богатейшими функциональными возможностями, развитыми средствами графического представления данных, а также наличием собственного языка программирования.

Система MathCad чрезвычайно полезна преподавателям и студентам высших учебных заведений - для иллюстрации изучаемых математических методов, выполнения самых различных вычислений, и выбор литературы по описанию этой системы в настоящее время достаточно широк.

Выбор литературы по описанию системы MathCad в настоящее время достаточно широк. Здесь есть краткий справочник [1.3]; полное описание версии MathCad 11 [1.1], которое автор рекомендует всем желающим глубоко изучить пакет. Создание настоящего пособия обусловлено тем, что на компьютерах дисплейных классов НТИ НИЯУ МИФИ в настоящее время установлена версия пакета MathCad 11, которая имеет некоторые отличия от рассмотренных в работах [2.1, 2.6, 2.7] версий. Данная работа посвящена описанию математического пакета MathCad 11 и является модернизацией методического пособия по работе в MathCad 2000 [3.1] выпущенного в 2003 году.

В данном пособии более полно рассмотрены некоторые ранее существовавшие возможности обработки данных, внесены изменения, определенные десятилетним опытом использования системы MathCad, переработаны ряд разделов, переделаны практически все рисунки документа, изменены некоторые задания для выполнения на лабораторных работах, исправлены опечатки. Кроме того, MathCad 11 имеет ряд изменений в интерфейсе, появились новые средства и возможности обработки данных

Настоящее пособие является продолжением работы [3.2] и предназначено для студентов всех специальностей, изучающих курсы “Информатика”, “Вычислительные методы в инженерных расчетах”, “Решение инженерных задач на ЭВМ”, а также для студентов и преподавателей, которые в своей работе встречаются с необходимостью вести математические расчеты различной степени сложности.

Описание относится к версии MathCad 11 и может быть с некоторыми коррективами использоваться для других версий этого пакета.

В разделе 1 описано, как при помощи MathCad можно решать нелинейные уравнения и системы таких уравнений.

Раздел 2 посвящен способам нахождения экстремумов функций.

Линейная интерполяция и интерполяция кубическими сплайнами функции одной переменной описаны в разделе 3.

Раздел 4 посвящен вопросам аппроксимации функций различными видами зависимостей.

Решение задачи Коши для дифференциальных уравнений, решение краевых задач и некоторые возможности MathCad по решению дифференциальных уравнений в частных производных описаны в разделе 5.

В разделе 6 рассмотрены статистические возможности MathCad: вычисление статистических оценок совокупностей, работа со случайными величинами, имеющими различные законы распределения.

Раздел 7 посвящен символьным возможностям MathCad, когда результатом вычислений является не число, а другое аналитическое выражение.

В разделе 8 описан собственный язык программирования MathCad, позволяющий решать такие задачи, которые невозможно или очень трудно решить другим способом.

В разделе 9 приведены задания для студентов, выполняющих лабораторные и практические работы по курсам, включающим в себя изучение MathCad. Эта часть пособия может быть также использована при самостоятельном изучении интегрированного пакета. Для возможности самоконтроля некоторые задачи (например, по дифференциальным уравнениям) снабжены ответами.

1 РЕШЕНИЕ УРАВНЕНИЙ И СИСТЕМ

В данном разделе описывается, как при помощи MathCad можно решать уравнения и системы уравнений. Максимальное число уравнений и неизвестных в системе равно 50.

1.1 Решение одного уравнения с одним неизвестным

Для решения одного уравнения с одним неизвестным $f(z) = 0$ используется функция **root(*f(z)*, *z*)**, которая возвращает значение z , при котором выражение или функция *f*(z) обращается в ноль.

Первый аргумент есть функция или выражение скалярного типа.

Второй аргумент - имя переменной, которая используется в выражении. Это та переменная, варьируя которой MathCad будет пытаться обратить выражение в ноль. Этой переменной перед использованием функции ***root*** необходимо присвоить начальное числовое значение, которое используется как начальное приближение при поиске корня.

Задача решения уравнения в системе Mathcad разбивается на два основных этапа:

1. отделение корней и нахождение начального значения,
2. непосредственное решение уравнения с помощью функции root.

Рассмотрим на примере 1.1 выполнение этих этапов.

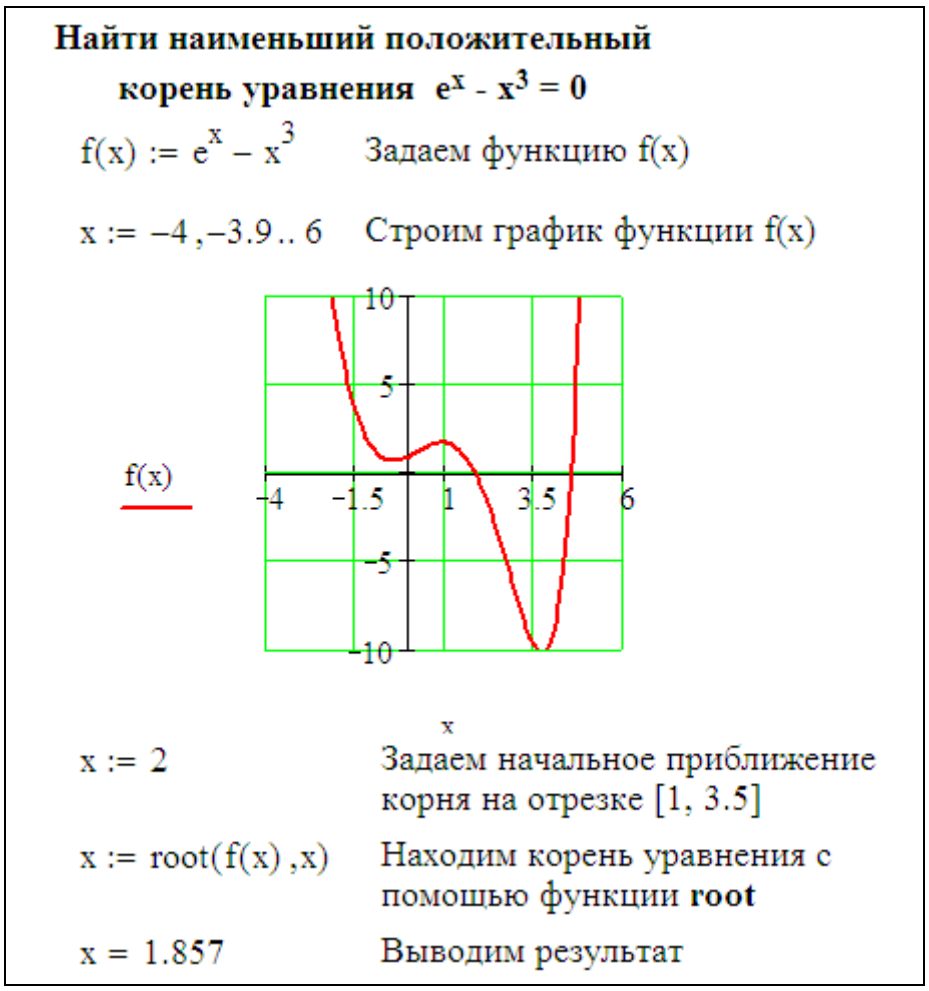


Рисунок 1.1 – Пример использования функции root

При использовании функции ***root*** необходимо иметь в виду следующее:

- переменной должно быть присвоено начальное значение до начала использования функции root;
- для выражения с несколькими корнями, начальное значение определяет корень, который будет найден MathCad. На рисунке 1.2(а) приведен пример, в котором функция root возвращает различные значения, каждое из которых зависит от начального приближения;

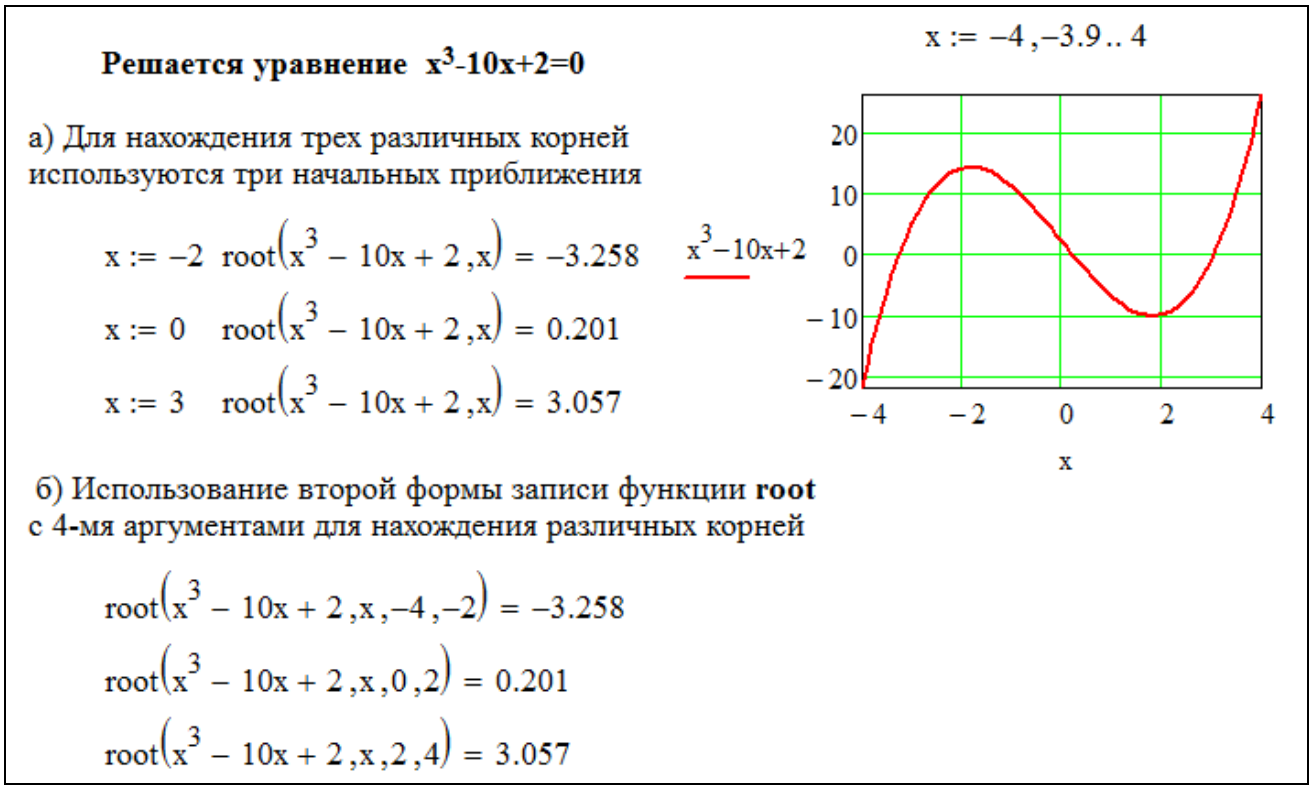


Рисунок 1.2 - Использование функции root для нахождения корней уравнения

- функция позволяет находить как вещественные, так и комплексные корни. Для поиска комплексного корня следует в качестве начального приближения взять комплексное число.
- задача решения уравнения вида *f(x) = g(x)* эквивалентна задаче поиска корня выражения *f(x) - g(x) = 0*. Для этого функция root может быть использована следующим образом: **root(f(x) - g(x), x)**.
- функция root предназначена только для решения одного уравнения с одним неизвестным.

MathCad при использовании функции *root* ищет корень *методом секущих* и начальное значение становится первым приближением к искомому корню. Когда значение выражения при очередной итерации становится меньше **TOL**, корень считается найденным, и функция *root* возвращает результат.

Если подходящее приближение не достигается, то появляется сообщение об ошибке, которое может быть вызвана следующими причинами:

- уравнение не имеет корней;
- корни уравнения расположены далеко от начального приближения;
- выражение имеет локальные экстремумы или разрывы между начальным приближением и корнями.

Чтобы установить причину ошибки, нужно исследовать график $f(x)$. Он поможет выяснить наличие корней уравнения $f(x) = 0$ и определить приблизительно их значения.

Для изменения точности нахождения корня можно изменить значение встроенной переменной **TOL** . При уменьшении **TOL** функция root будет сходиться медленнее, но ответ будет более точен.

В среде MathCad 11 появилась вторая форма функции root – с четырьмя аргументами – **root(*f(z),z,a,b*).** Первая форма (два аргумента) означает прежнее содержание этой функции – поиск корня уравнения с опорой на начальное приближение, а вторая форма (четыре аргумента) – поиск корня в заданном интервале $[a,b]$. Пример использования второй формы функции root показан на рис 1.2(б).

1.2 Нахождение корней полиномов

Для нахождения корней уравнения $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$ используется функция

polyroots(*a*),

аргументом которой является вектор длины $n + 1$, содержащий коэффициенты полинома $a_0, a_1, \ldots, a_n$. Коэффициенты могут быть как вещественными числами, так и комплексными.

В отличие от функции root, polyroots не требует начального приближения и возвращает сразу все корни, как вещественные, так и мнимые.

На рисунке 1.3 показан пример использования функции polyroots.

Решается уравнение $x^3-10x+2=0$ с помощью функции *polyroots*

Вектор коэффициентов, начинающийся с константы. Сюда должны войти все коэффициенты, в том числе равные нулю:

$$a := \begin{pmatrix} 2 \\ -10 \\ 0 \\ 1 \end{pmatrix}$$

Возвращаются все ответы:

$$\text{polyroots}(a) = \begin{pmatrix} -3.258 \\ 0.201 \\ 3.057 \end{pmatrix}$$

Рисунок 1.3 - Пример использования функции polyroots.

1.3 Решение систем нелинейных уравнений

Рассмотрим решение системы ***n*** нелинейных уравнений с ***n*** неизвестными:

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0, \\ \quad \ldots \\ f_n(x_1, \ldots, x_n) = 0, \end{cases}$$

Для решения систем в MathCad необходимо:

- задать *начальное приближение* для всех неизвестных, входящих в систему, на основе которых строится итерационная последовательность уточнений начального приближения, сходящаяся к искомому решению;
- напечатать ключевое слово **Given**, которое указывает MathCad, что далее следует система уравнений;
- ниже слова Given ввести уравнения и неравенства. Для ввода знака равенства между левыми и правыми частями уравнений необходимо использовать клавиатурную комбинацию "**Ctrl/=**". Между левыми и правыми частями неравенств могут также стоять знаки: $>, <, \leq, \geq$;
- ввести любое выражение, которое включает функцию **Find (z_1, z_2 , ...).** Число аргументов в этой функции должно быть равно числу неизвестных.

Функция Find возвращает найденное решение следующим образом:

- если она имеет только один аргумент, то возвращается скаляр, являющийся решением уравнения, расположенного между ключевым словом Given и функцией Find;
- если Find имеет больше одного аргумента, то она возвращает ответ в виде вектора, содержащего решение системы уравнений.

Ключевое слово Given, уравнения и неравенства, следующие за ним, и выражение, содержащее функцию Find, называются *блоком решения* уравнений.

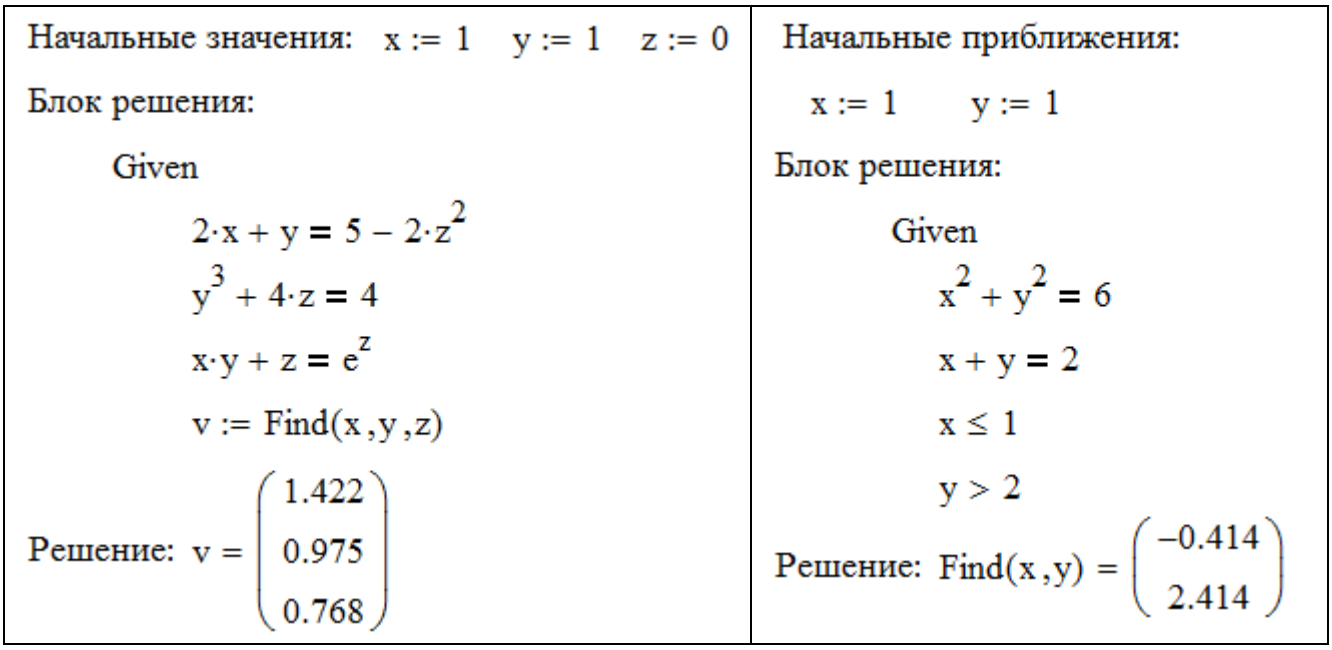
В блоке решений недопустимы операторы присваивания, дискретные аргументы, а также выражения, их содержащие, и неравенства вида $a < b < c$.

Блоки решений не могут быть вложены друг в друга. Каждый блок должен иметь только одно ключевое слово Given и функцию Find.

На рисунке 1.4(а) показан пример блока решения системы трех уравнений с тремя неизвестными. В результате функция Find содержит три аргумента, x , y и z, и возвращает ответ в виде вектора с тремя компонентами.

Функция Find, которая завершает блок решения уравнений, может быть использована аналогично любой другой функции и чаще всего с ней производится одно из следующих действий:

- можно вывести найденное решение, напечатав выражение вида Find(z_1, z_2,...)=. Пример того, как это делается для системы двух уравнений с двумя неизвестными, приведен на рисунке 1.4(б);
- можно определить переменную с использованием этой функции. Для этого в конце блока решения необходимо записать оператор присваивания, слева в котором стоит переменная, а справа функция Find. Пример, иллюстрирующий такую возможность, показан на рисунке 1.4(а);



а) б)

Рисунок 1.4 - Примеры решения систем нелинейных уравнений

- используя Find, можно определить другую функцию. Для этого необходимо закончить блок решения выражением вида

$$f(a, b, \ldots) := \text{Find}(z_1, z_2, \ldots).$$

Эта конструкция удобна при многократном решении системы уравнений для различных значений параметров *a, b, ...*, непосредственно входящих в систему уравнений. Пример такого использования показан на рисунке 1.5. Определенная в этом примере функция Find для каждого значения параметра R из заданного диапазона возвращает вектор, состоящий из двух элементов, поэтому выводить здесь необходимо раздельно элементы вектора $F(R)_0$ и $F(R)_1$.

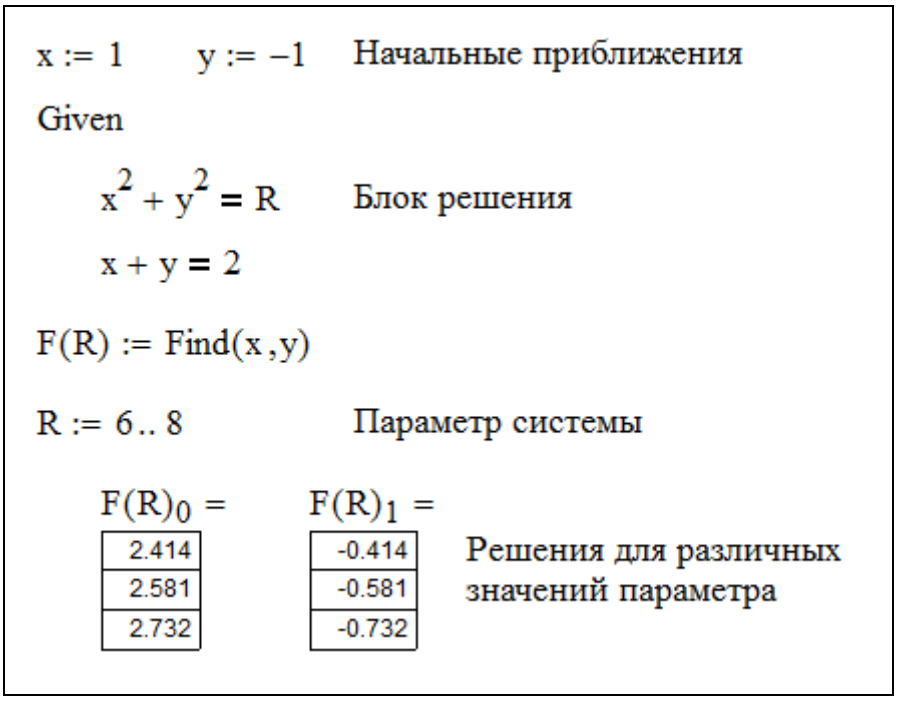


Рисунок 1.5 - Пример многократного решения системы

MathCad возвращает в блоке решения уравнений только одно решение системы. Однако, система может иметь несколько решений. Если одно решение найдено, то для поиска других решений можно использовать другие начальные приближения либо ввести дополнительные ограничения в виде неравенств, определяющих нужную область. На рисунке 1.6 показано, как различные начальные приближения дают различные решения задачи, а на рисунке 1.7 - как добавить ограничения в виде неравенств для поиска другого решения.

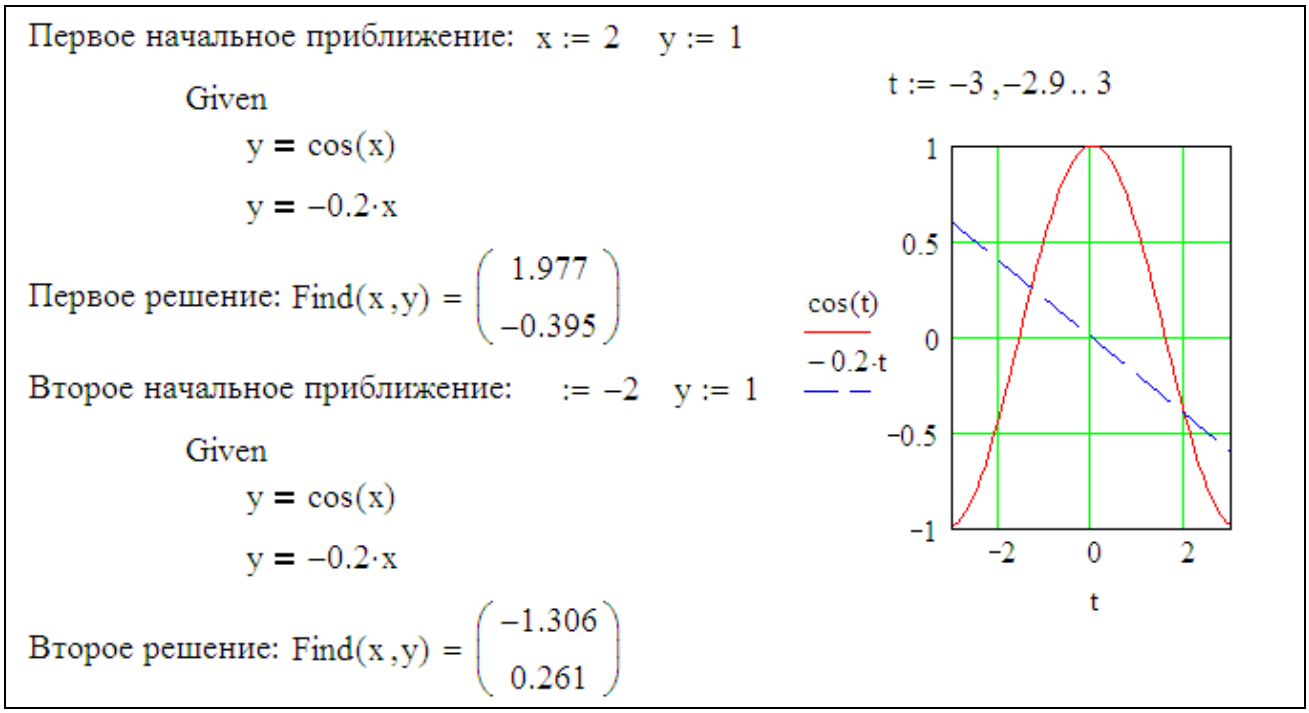


Рисунок 1.6 - Влияние начальных приближений на решение системы

Рисунок 1.7 - Ограничения в блоке решения уравнений

Если в результате итерационного процесса не может быть найдено более близкое приближение к искомому решению по сравнению с предыдущим шагом, то поиск решения прекращается, а функция Find помечается сообщением об ошибке.

Причиной появления этого сообщения может быть следующее:

- задача не имеет решения;
- для уравнения, которое не имеет вещественных корней, в качестве начального приближения взято вещественное число;
- в процессе решения получена точка, из которой применяемый MathCad метод минимизации не может определить дальнейшее направление движения. Метод преодоления этой проблемы - изменение начального приближения или добавление ограничений на переменные в виде неравенств, чтобы обойти критическую точку.

В любом случае полезно вывести графики, связанные с системой, которые облегчат поиск области, где находится решение и выбрать подходящее начальное приближение.

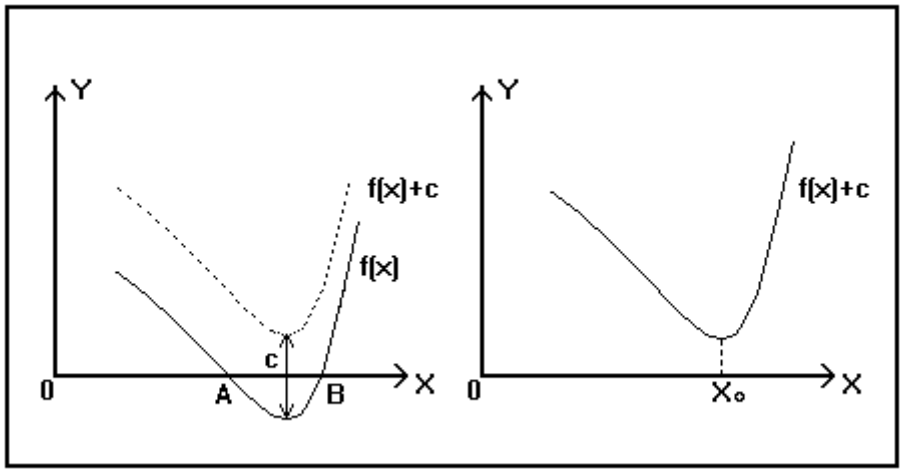
Точность решения систем уравнений определяется встроенной константой TOL, изменяя которую можно увеличивать или уменьшать погрешность вычисления корней.

2 НАХОЖДЕНИЕ ЭКСТРЕМУМОВ ФУНКЦИЙ

2.1 Функция Minerr

В системе MathCad до версии 8 отсутствовали специально предназначенные функции для поиска минимумов (максимумов), но для этой цели с успехом использовалась встроенная функция **Minerr (*z1, z2,...*)**, которая очень похожа на функцию Find и использует тот же самый алгоритм поиска. Различие между ними состоит в следующем: если в результате поиска решения не может быть получено дальнейшее уточнение текущего приближения к решению, Minerr возвращает это приближение, а функция Find, в отличие от функции Minerr, возвращает в этом случае сообщение об ошибке. Правила использования функции Minerr точно такие же, как и функции Find.

Идея нахождения минимума с помощью функции Minerr состоит в следующем. Если применить эту встроенную функцию для решения уравнения $f(x) = 0$ (рисунок 2.1 а), то будут найдены корни этого уравнения (точки A или B - в зависимости от начального приближения). Если же исследуемую функцию $f(x)$ сдвинуть в положительном направлении оси 0Y таким образом, чтобы она вся находилась выше оси абсцисс (рисунок 2.1 б), то решение уравнения с помощью функции Minerr даст координату X_0, в которой функция $f(x) + c$ наиболее приближена к оси 0X.



а) б)

Рисунок 2.1 - Преобразование функции для нахождения минимума

а) исходная функция $f(x)$. A, B - корни уравнения $f(x) = 0$;
б) преобразованная функция $f(x) + c$. X_0 - точка минимума.

Для нахождения минимума функции $f(x)$ необходимо:

- построить график $f(x)$ и из этого графика оценить величину сдвига C, на который нужно “приподнять” функцию, чтобы она целиком располагалась выше оси абсцисс;
- с помощью функции Minerr решить уравнение $f(x) + c = 0$, находя тем самым величину X_0.

Максимум функции находится аналогично либо сдвигом функции $f(x)$ в отрицательном направлении оси 0Y так, чтобы результирующая функции целиком располагалась ниже оси абсцисс, либо нахождением минимума функции $-f(x)$.

На рисунке 2.2 показан пример нахождения минимума функции одной переменной, на рисунке 2.3 - двух переменных.

Рисунок 2.2 - Пример нахождения минимума функции одной переменной

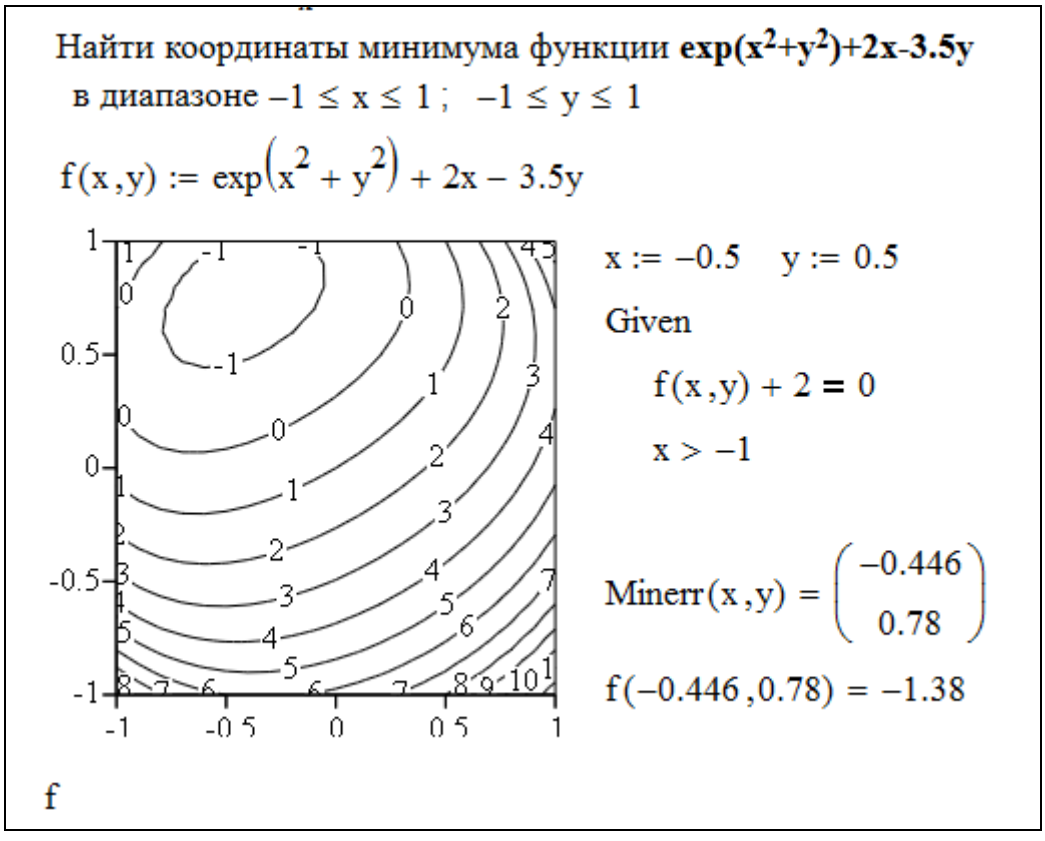


Рисунок 2.3 - Пример нахождения минимума функции двух переменных

2.2 Функции Minimize, Maximize

В среду MathCad версии 8 и выше введены две новые функции minimize и maximize, позволяющие решать оптимизационные задачи без вышеописанных в разделе 2.1 ухищрений. Эти функции имеют вид:

Minimize(*f*, *var1*, *var2*, ...);

Maximize(*f*, *var1*, *var2*, ...).

Функции вычисляют величины *var1, var2, ...*, которые удовлетворяя условиям, заданным в блоке решения делают значения функции *f* минимальными либо максимальными.

Аргументы:

var1, var2, ... – скалярные переменные, используемые в блоке решения. Их начальные значения должны быть заданы перед блоком решения.

f – функция, которая должна быть определена перед блоком решения.

Для нахождения минимума или максимума функции необходимо:

- определить функцию, минимум или максимум нужно найти;
- задать начальное приближение для всех неизвестных, которые нужно найти;
- напечатать ключевое слово Given, начинающее блок решения;
- ниже слова Given ввести уравнения и неравенства, определяющие область решения;
- ввести любое выражение, которое включает функцию minimize или maximize.

Эти функции возвращают скаляр, если используется только одна переменная. Иначе результат есть вектор, первый элемент которого есть *var1*, второй - *var2*, и т. д.

Если в блоке решения нет условий, то слово Given можно не вводить.

Функции minimize и maximize нечувствительны к регистру.

Пример использования функций для нахождения экстремумов показан на рисунках 2.4 и 2.5.

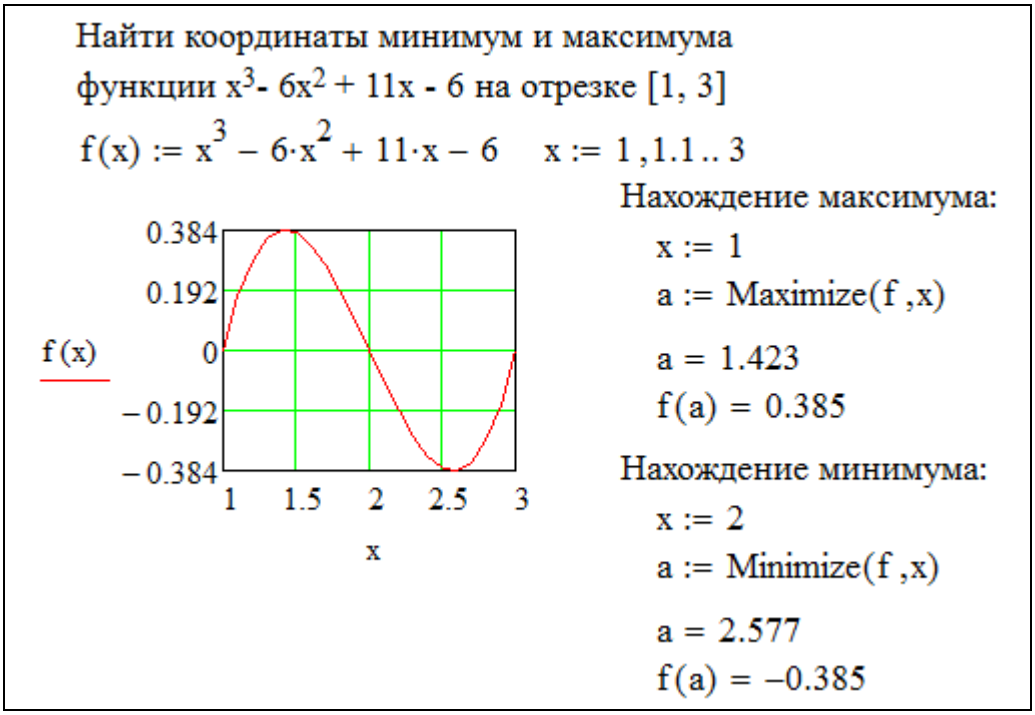
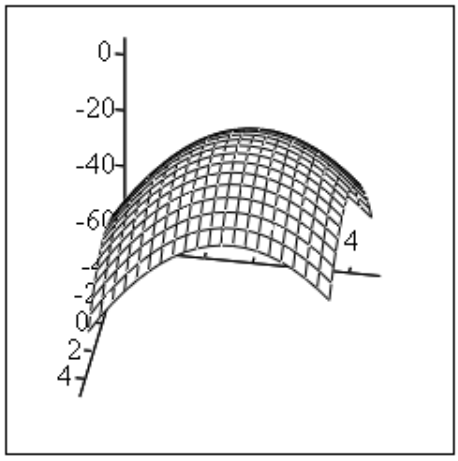


Рисунок 2.4 - Пример нахождения минимума и максимума одномерной функции

Наити максимум и минимум функции $f(x,y) := 2 + 2\cdot x + 2\cdot y - x^2 - y^2$ в области, ограниченной треугольником с координатами вершин (0,0), (9,0) и (0,9)

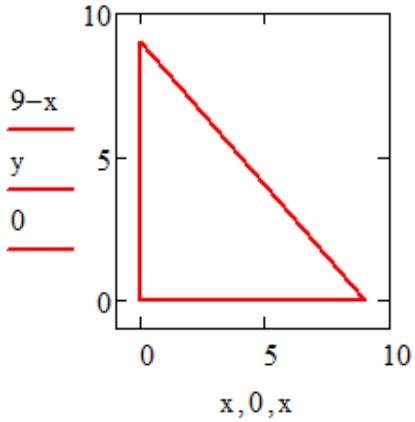
График функции



f

График области поиска
(xy -плоскость)

$x := 0..9 \quad y := 0..9$



1. Нахождение первого минимума

$x := 4 \quad y := 5$

Given

$x \geq 0$

$0 \leq y \leq 9 - x$

$P := \text{Minimize}(f, x, y) \quad P = \begin{pmatrix} 0 \\ 9 \end{pmatrix} \quad f(P_0, P_1) = -61$

2. Нахождение второго минимума

$x := 5 \quad y := 4$

Given

$x \geq 0$

$0 \leq y \leq 9 - x$

$Q := \text{Minimize}(f, x, y) \quad Q = \begin{pmatrix} 9 \\ 0 \end{pmatrix} \quad f(Q_0, Q_1) = -61$

3. Нахождение максимума

$x := 5 \quad y := 4$

Given

$x \geq 0$

$0 \leq y \leq 9 - x$

$R := \text{Maximize}(f, x, y) \quad R = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad f(R_0, R_1) = 4$

Рисунок 2.5. Пример нахождения минимума и максимума двумерной функции.

3 ИНТЕРПОЛЯЦИЯ

Интерполяция использует значения таблично заданной функции в ряде точек (узлах интерполяции), чтобы вычислить значения этой функции в точках, лежащих между узлами. В MathCad можно соединять точки данных прямыми линиями (линейная интерполяция), фрагментами кубического полинома (кубическая сплайн-интерполяция) или полиномами. Функции интерполяции всегда определяют кривую, точно проходящую через заданные точки.

3.1 Линейная интерполяция

При линейной интерполяции MathCad соединяет существующие точки данных прямыми линиями. Это выполняется функцией

linterp (*vx, vy, x*),

где: **vx**, **vy** - векторы данных, соответствующих *x* и *y*, которые должны быть одинаковой длины. Вектор **vx** должен содержать вещественные значения, расположенные в порядке возрастания;
x - точка, для которой нужно найти значение функции.

Эта функция соединяет точки данных отрезками прямых, создавая таким образом *ломаную линию*. Интерполируемое значение для заданного *x* есть ордината *y* соответствующей точки ломаной.

Для значений *x*, расположенных перед первой точкой в **vx**, MathCad продолжает ломаную прямой линией, проходящей через первые две точки данных. Для значений *x*, расположенных за последней точкой в **vx**, MathCad продолжает ломаную прямой линией, проходящей через последние две точки данных.

На рисунке 3.1 показаны некоторые примеры линейной интерполяции.

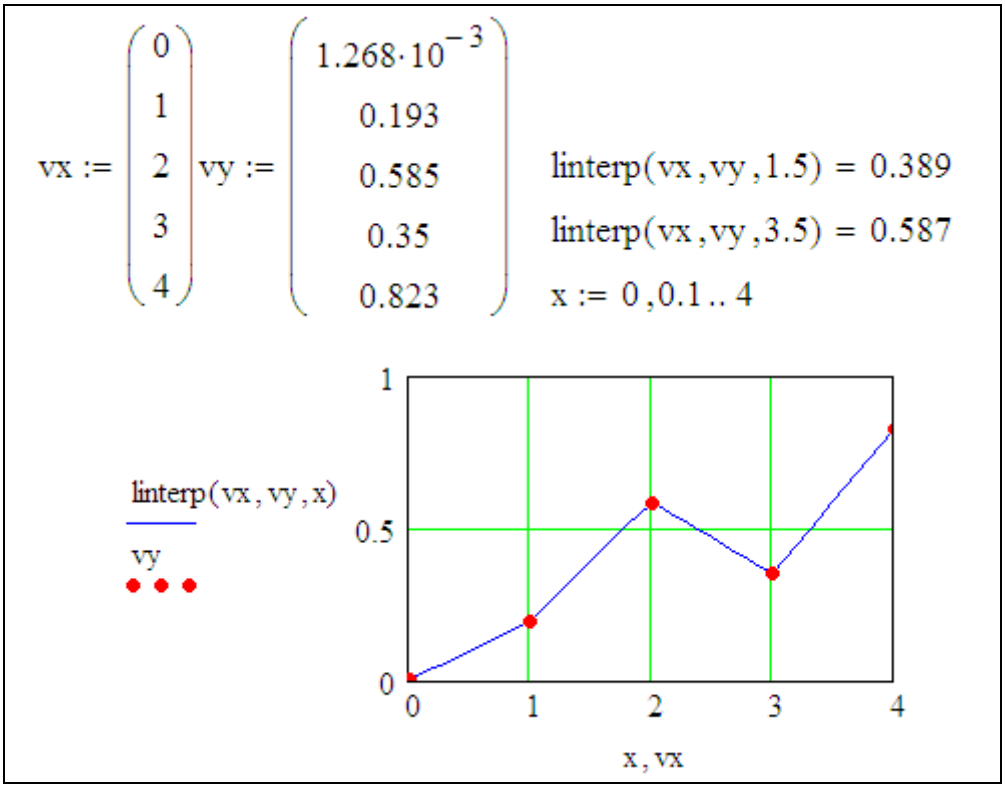


Рисунок 3.1 - Примеры линейной интерполяции

3.2 Кубическая сплайн-интерполяция

Кубическая сплайн-интерполяция позволяет провести кривую через набор точек таким образом, что первые и вторые производные кривой непрерывны в каждой точке. Эта кривая образуется путем создания ряда кубических полиномов, проходящих через наборы из трех смежных точек. Кубические полиномы затем состыковываются друг с другом, чтобы образовать гладкую кривую.

Чтобы провести кубический сплайн через набор точек (x_i, y_i) нужно:

- создать векторы **vx** и **vy**, содержащие координаты x и y, через которые нужно провести кубический сплайн. Элементы **vx** должны быть расположены в порядке возрастания;
- вычислить вектор **vs**:= cspline (**vx**, **vy**). Вектор **vs** будет содержать вторые производные интерполяционной кривой в рассматриваемых точках;
- вычислить интерполируемое значение в произвольной точке x_0 с помощью функции interp (**vs**, **vx**, **vy**, x0), где **vs**, **vx**, **vy** - векторы, описанные выше.

То же самое можно сделать, вычисляя:

interp (cspline (*vx, vy, vx, vy, x0*)

На рисунке 3.2 показан пример кубической сплайн-интерполяции.

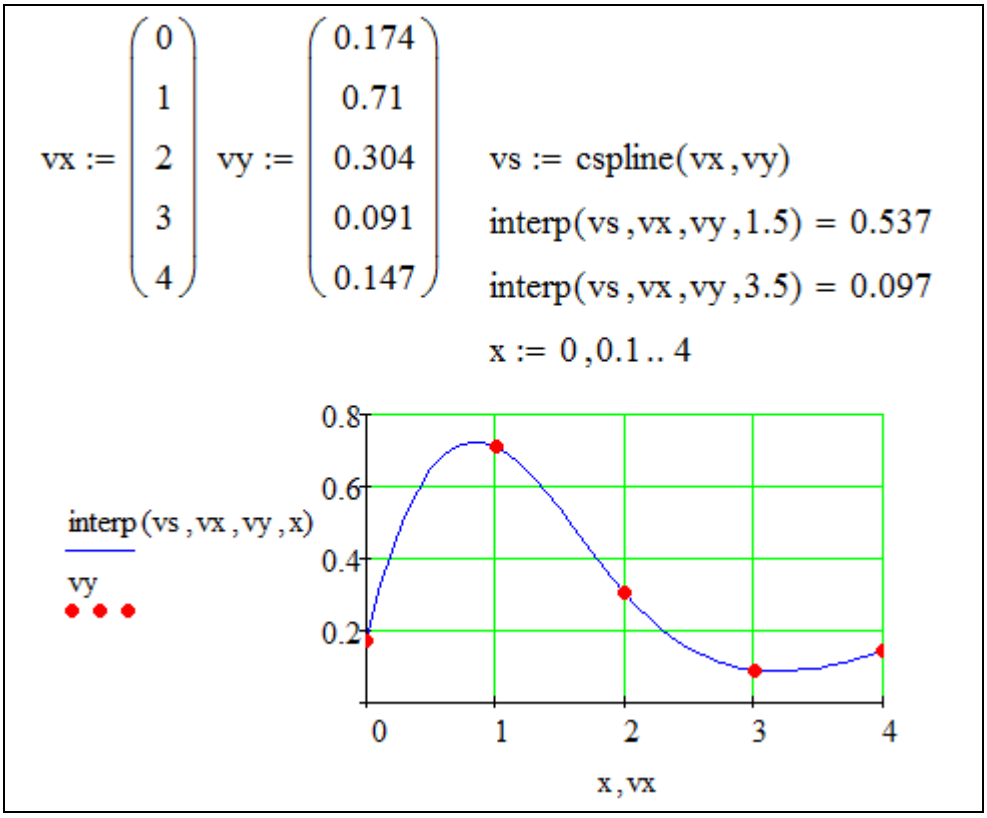


Рисунок 3.2 - Пример кубической сплайн-интерполяции

Обратите внимание, что массив **vs** должен вычисляться только один раз даже для нескольких интерполяций. Так как вычисление **vs** требует много времени, лучше сохранять промежуточные результаты в виде вектора, чем повторно вычислять их по мере необходимости.

Интерполируемое значение для конкретного x есть ордината y соответствующей точки сплайна. Для значений x, расположенных перед первой точкой в **vx**, MathCad продолжает сплайн первой из составляющих его кубических парабол. Для

значений *x*, расположенных за последней точкой в **vx**, MathCad продолжает сплайн последней из составляющих его кубических парабол.

В дополнение к **cspline** существуют еще две другие кубические сплайн-функции:

- ✓ функция **lspline**(**vx**, **vy**) генерирует кривую сплайна, которая приближается к прямой линии в граничных точках;
- ✓ функция **pspline**(**vx**, **vy**) генерирует кривую сплайна, которая приближается к параболе в граничных точках;

3.3 Полиномиальная сплайн-интерполяция

Кубический сплайн является эффективным средством построения интерполяционной кривой в подавляющем большинстве случаев. Но в некоторых случаях использование кубического сплайна может привести к нежелательным результатам. Чаще всего это происходит в тех случаях, когда данные очень неравномерно распределены вдоль оси *x*. В некоторых подобных случаях получить лучшую интерполяционную кривую помогает использование другого вида интерполяции – B-сплайна. Основное отличие B-сплайна от всех описанных выше методов – сшивка отрезков кривых происходит не в экспериментальных точках, а между ними, в специально заданных точках.

В MathCAD для реализации интерполяции B-сплайном служит функция

bspline(*vx, vy, u, n*),

где **vx** и **vy** – уже знакомые нам векторы, содержащие координаты экспериментальных точек,

u – вектор, содержащий координаты точек сшивки,

n - порядок интерполирующих полиномов (1, 2 или 3).

Результатом функции *bspline* является вектор, который далее следует использовать как аргумент функции *interp*, хотя содержимое этого вектора сильно отличается от результатов функций кубических сплайнов.

Количество точек сшивки не является произвольной величиной и должно быть всегда на *n-1* меньше, чем количество экспериментальных точек. Также на координаты точек сшивки накладывается следующие условие: первый элемент вектора *u* и должен быть меньше или равен первому элементу вектора *vx*, а последний элемент — больше или равен последнему элементу *vx*. Остальные точки сшивки могут произвольным образом располагаться внутри отрезка, конечно, при условии, что через них возможно провести интерполяционную кривую

Пример использования B-сплайна на рисунке 3.3 иллюстрирует, как с помощью B-сплайна можно иногда получить лучший результат, чем с помощью кубического сплайна.

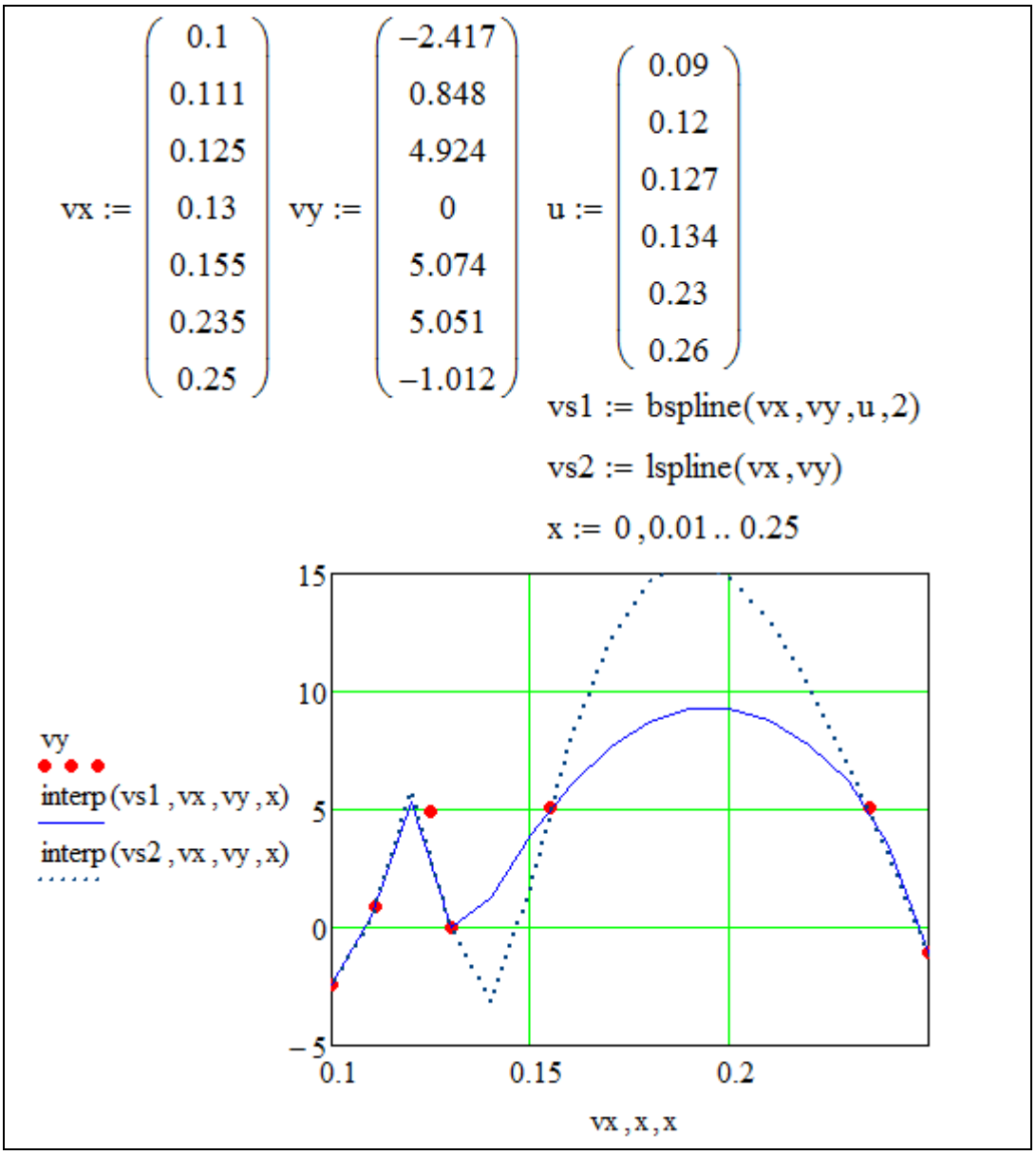


Рисунок 3.3 – Сравнение эффективности кубического сплайна и B- сплайна

3.4 Двумерная сплайн-интерполяция

С помощью кубических сплайнов в MathCad можно также построить интерполяционную поверхность для экспериментальных зависимостей, редставленных в виде функции двух переменных. Для этого служат уже знакомые функции *lspline*, *pspline*, *cspline* и *interp*.

Конечно, в отличие от одномерной интерполяции, в данном случае экспериментальные значения не могут быть заданы в произвольных точках на плоскости. Интерполяционная поверхность может быть построена только в том случае, когда значения заданы в узлах прямоугольной сетки $n \times n$. Для построения такой поверхности нужно выполнить следующую последовательность действий:

- задать экспериментальные значения в виде квадратной матрицы **Z** размерности n;
- задать матрицу **M** для описания сетки. Эта матрица должна состоять из двух столбцов и n строк. Каждый столбец матрицы M задает положение линий сетки по одной из координат;

- для вычисления двумерного сплайна следует воспользоваться функцией **cspline(*M, Z*)** (также можно *lspline* или *pspline*). Результатом данной функции будет вектор, который далее нужно использовать как аргумент функции *interp.*

На рисунке 3.4 показан пример двумерной кубической сплайн-интерполяции.

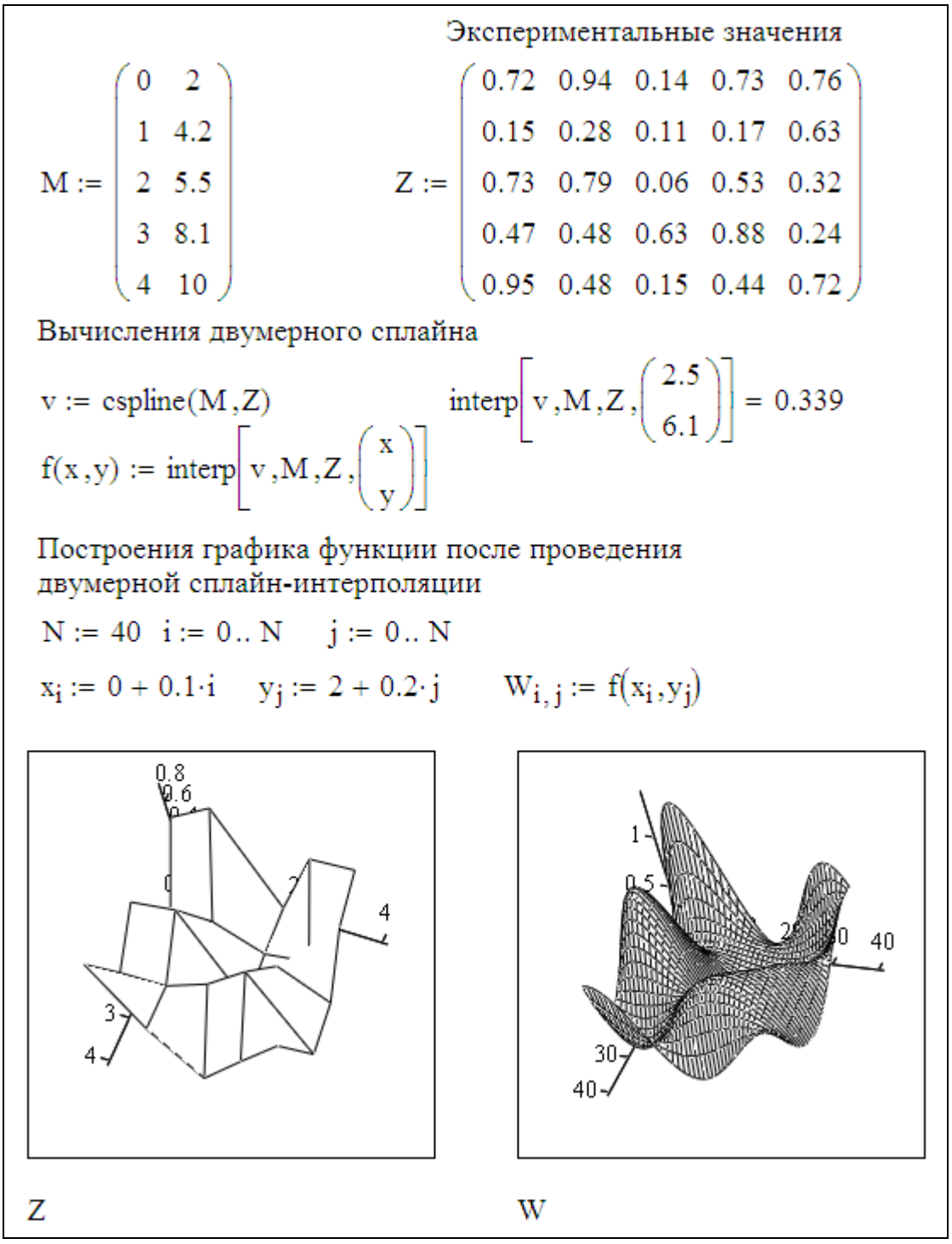


Рисунок 3.4 - Пример двумерной интерполяции кубическим сплайном

3.5 Линейное предсказание

Функции интерполяции, описанные в этом разделе до сих пор, позволяют по заданным значениям некоторой функции в ряде точек оценить ее значение в промежуточных точках. Иногда бывает необходимо оценить значения функции в точках, расположенных вне области расположения сетки, на которой заданы значения функ-

ции. В MathCad есть функция predict, которая позволяет это сделать. Эта функция использует линейный алгоритм предсказания, который уверенно работает, когда экстраполируемая функция является гладкой и осциллирующей, хотя и не обязательно периодической.

Линейное предсказание можно рассматривать как разновидность экстраполяции, но оно не имеет ничего общего с линейной или полиномиальной экстраполяцией. Функция линейного предсказания имеет вид:

predict (*v, m, n*),

где **v** - вектор, содержащий значения, подлежащие обработке с целью предсказания дальнейших значений. Элементы этого вектора должны представлять собой значений, взятые для равноотстоящих узлов;

m - число последних точек в векторе **v**, используемых для предсказания (m + 1)- -ой точки;

n - число предсказанных значений, возвращаемых функцией predict.

Пример использования функции predict показан на рисунке 3.5.

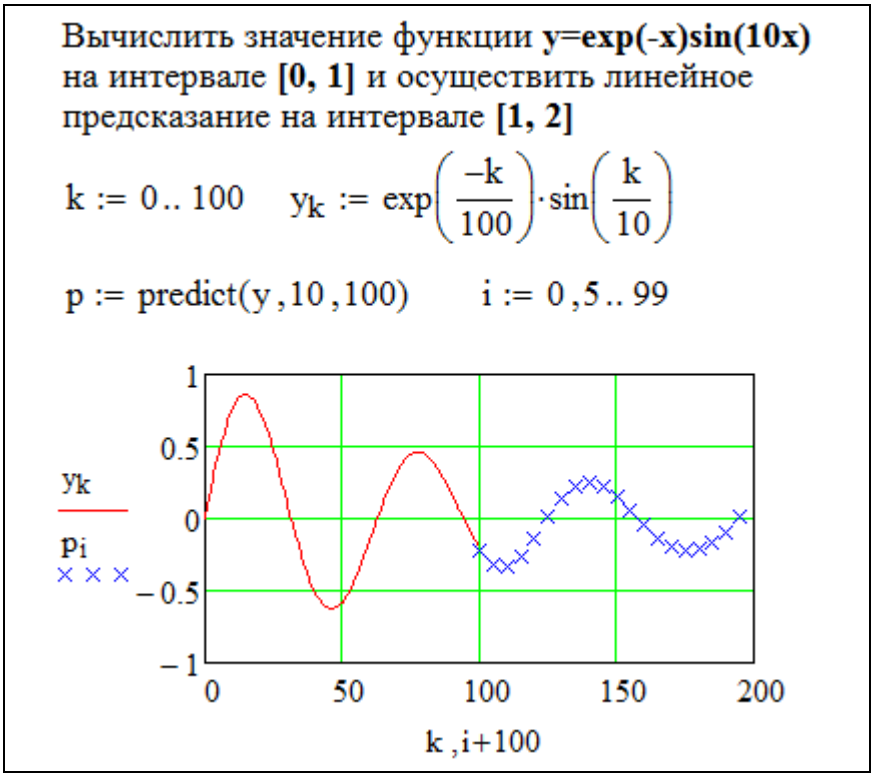


Рисунок 3.5 - Использование функции предсказания для экстраполяции данных

4 АППРОКСИМАЦИЯ

MathCad включает несколько функций для аппроксимации экспериментальных данных теоретической кривой, параметры которой определяются из условия минимума суммы квадратов отклонений.

В отличие от функций интерполяции, обсужденных в предыдущем разделе, эти функции не требуют, чтобы аппроксимирующая кривая или поверхность проходили в точности через точки данных, поэтому они гораздо менее чувствительны к ошибкам данных, чем функции интерполяции.

Конечный результат аппроксимации - вычисление параметров аппроксимирующей функции, с помощью которой можно, например, оценить значения в промежутках между заданными точками.

4.1 Линейная зависимость

Если известно, что зависимость между величинами x и y определяется выражением $y = a\,x + b$, то угловой коэффициент a и свободный член b могут быть найдены с помощью функций MathCad:

$$a := \mathbf{slope}(x, y);$$
$$b := \mathbf{intercept}(x, y),$$

где $\boldsymbol{x}$ - вектор значений независимой переменной;

$\boldsymbol{y}$ - вектор значений зависимой переменной.

На рисунке 4.1 показан пример аппроксимации данных линейной зависимостью.

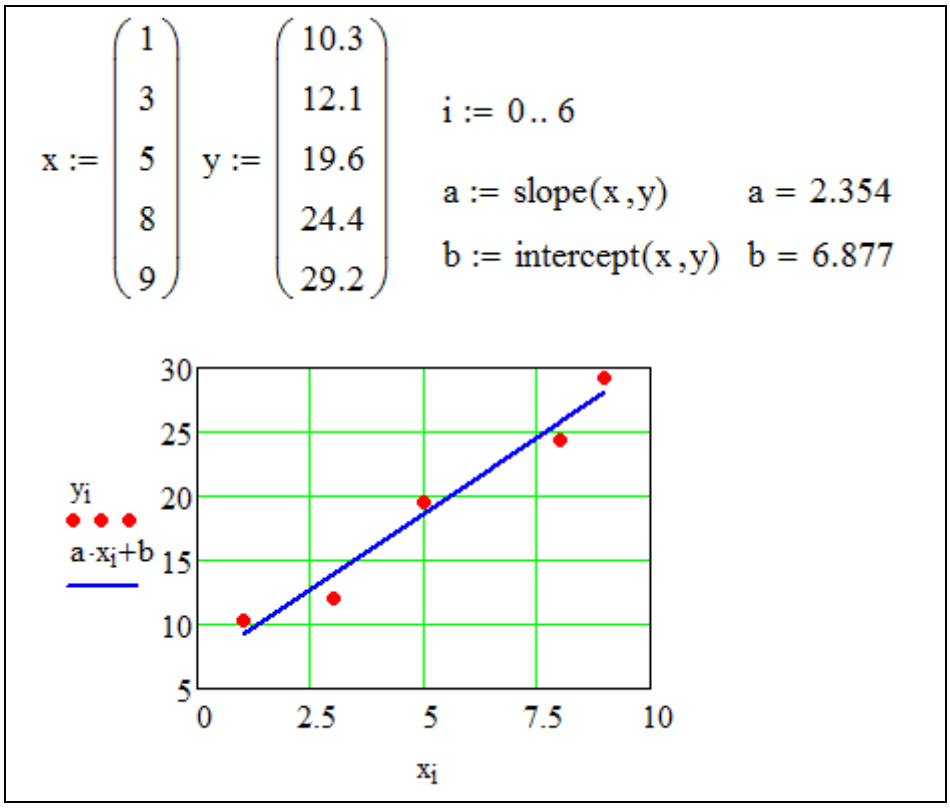


Рисунок 4.1 - Пример аппроксимации данных линейной зависимостью

4.2 Зависимость, сводящаяся к линейной

Часто нелинейная зависимость путем элементарных математических преобразований может быть сведена к линейной, в этом случае для ее аппроксимации могут быть использованы функции, описанные в разделе 4.1.

Пример 1. Если *x* и *y* связаны соотношениями вида

$$y = b \cdot exp(-a \cdot x^2)$$

1. Прологарифмируем обе части выражения: $ln(y) = ln(b) - a \cdot x^2$.

2. Выполним замену переменных $Y = ln(y)$ и $X = -x^2$, $B = ln(b)$, сводим зависимость к линейной: $Y = B + a \cdot X$

3. Найдем коэффициенты *a*, *b*.

a : = *slope (X, Y)* ;

b : = *exp (intercept (X, Y))* .

На рисунке 4.2 показана аппроксимация данных зависимостью из примера 1.

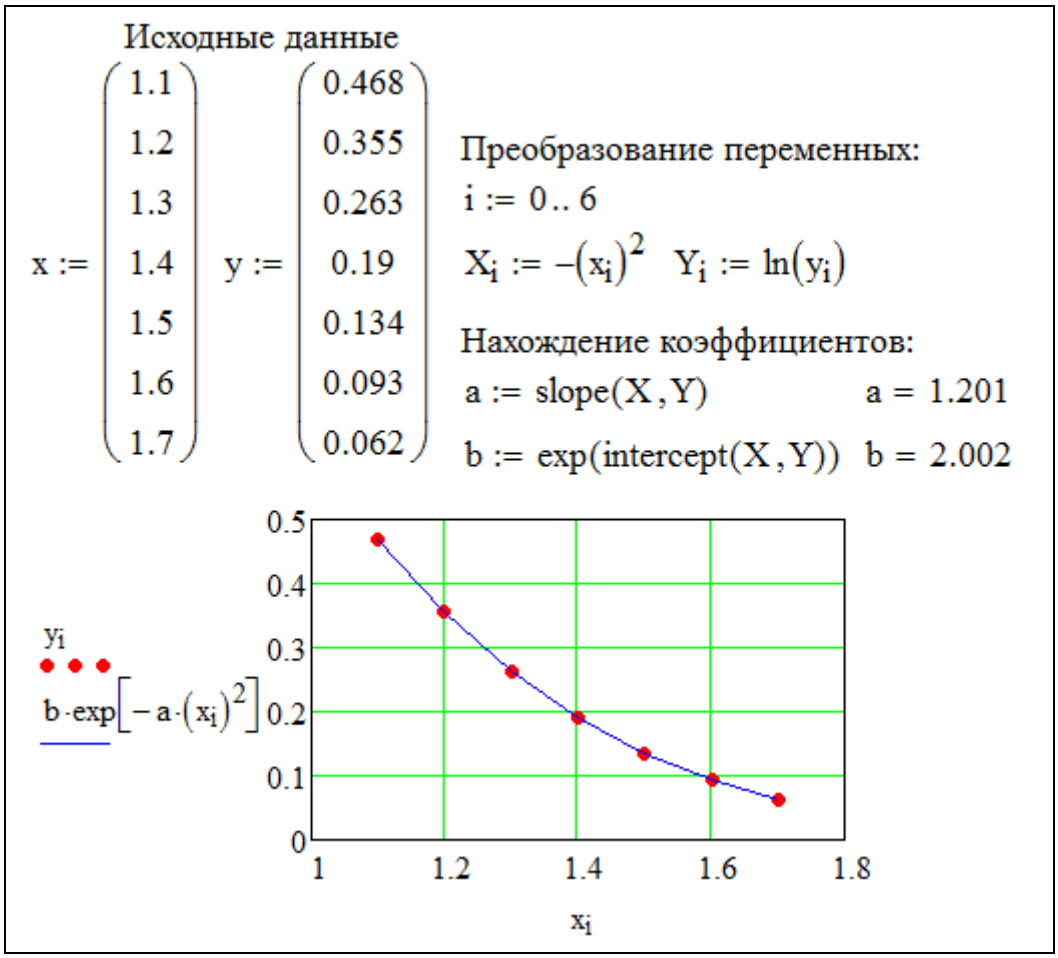


Рисунок.4.2 - Аппроксимация данных зависимостью $y = b \cdot exp(-a \cdot x^2)$

Пример 2. Если *x* и *y* связаны соотношениями вида

$$y = x/(a \cdot x + b)$$

1. «Перевернём» обе части выражения: $1/y = a + b/x$.

2. Выполним замену переменных $Y = 1/y$ *и* $X = 1/x$, сводим зависимость к линейной: $Y = a + b \cdot X$

3. Найдем коэффициенты *a*, *b*.

b : = slope (X, Y) ;

a : = intercept (X, Y) .

На рисунке 4.3 показана аппроксимация данных зависимостью из примера 2.

Рисунок.4.3 - Аппроксимация данных зависимостью y = x/(a·x+b)

4.3 Полиномиальная зависимость

Для аппроксимации данных полиномом $y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$, используется функция MathCad

regress (*x*, *y*, *n*),

где ***x*** - вектор значений независимой переменной,

y - вектор значений зависимой переменной,

n - степень аппроксимирующего полинома.

Функция regress допускает использование полинома любого порядка, но на практике использование полинома выше $n = 4 \div 5$ обычно лишено смысла.

Функция возвращает вектор, содержащий $n + 3$ компоненты, причем первые три из них - вспомогательные, а остальные соответствуют коэффициентам $a_0, a_1, \ldots, a_n$.

Для вычисления значений аппроксимирующей функции в произвольной точке x_0 можно использовать выражение:

$$\sum_{i=3}^{n} regress(\boldsymbol{x}, \boldsymbol{y}, n)_i \cdot x_0^{\,i-3},$$

либо использовать встроенную функцию MathCad

interp (*vs, x, y, x₀*),

где ***vs*** - вектор, вычисленный функцией regress.

Пример полиномиальной регрессии показан на рисунке 4.4.

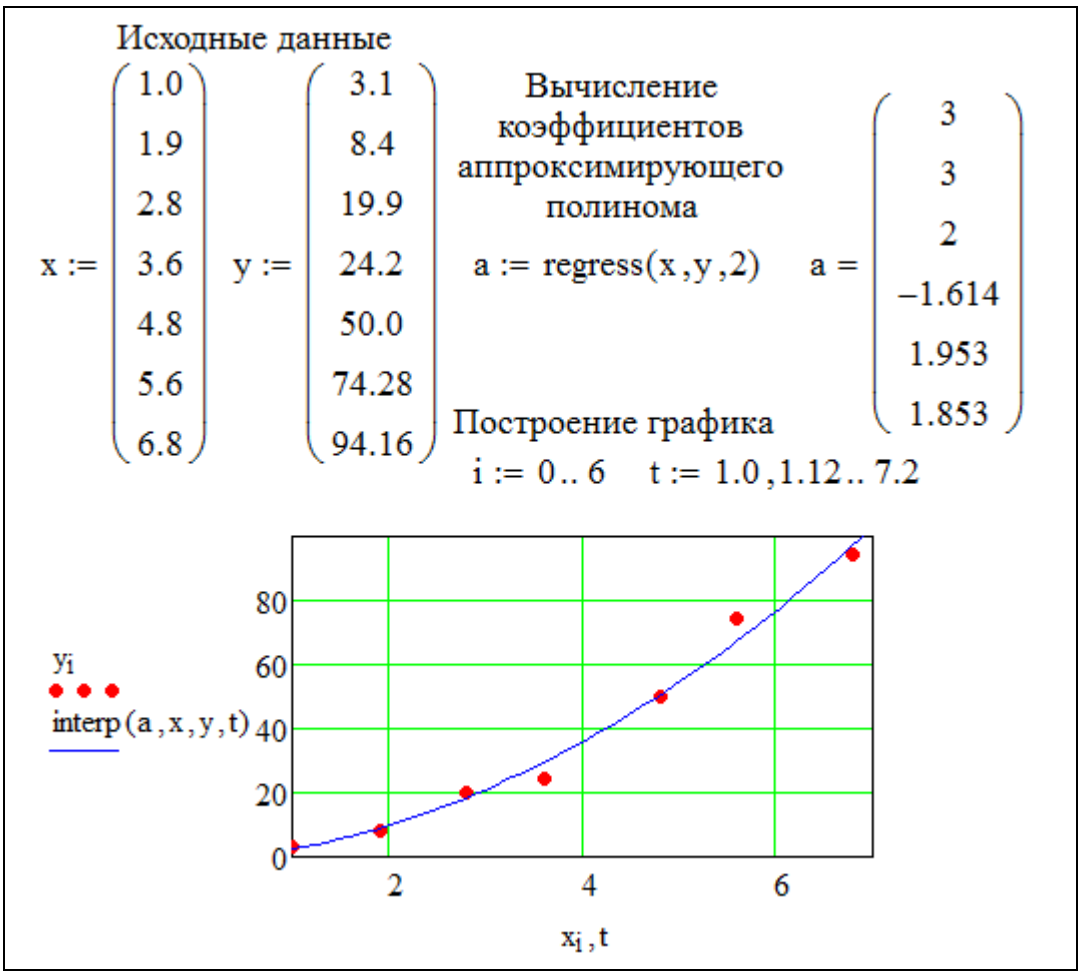


Рисунок 4.4 - Аппроксимация данных полиномом второго порядка

4.4 Линейная комбинация функций

Рассмотренные зависимости далеко не во всех случаях подходят для описания данных. Бывает, что нужно искать эту зависимость в виде линейных комбинаций известных функций. Например, в рядах Фурье следует аппроксимировать данные, используя комбинацию комплексных компонент.

Функция MathCad **linfit(*x*, *y*, F)**, где ***x***, ***y*** - векторы данных, **F** - функция, содержащая вектор функций $f_i(x)$, которые нужно объединить в виде линейной комбинации, предназначена для решения этой задачи. Функция возвращает коэффициенты a_0, a_1, … , a_n аппроксимирующего выражения

$$y=a_0f_0(x)+a_1f_1(x)+\cdots+a_nf_n(x)$$

На рисунке 4.5 показан пример аппроксимации данных выражением

$$a_0x+a_1x^2+a_2(x+1)^{-1}$$

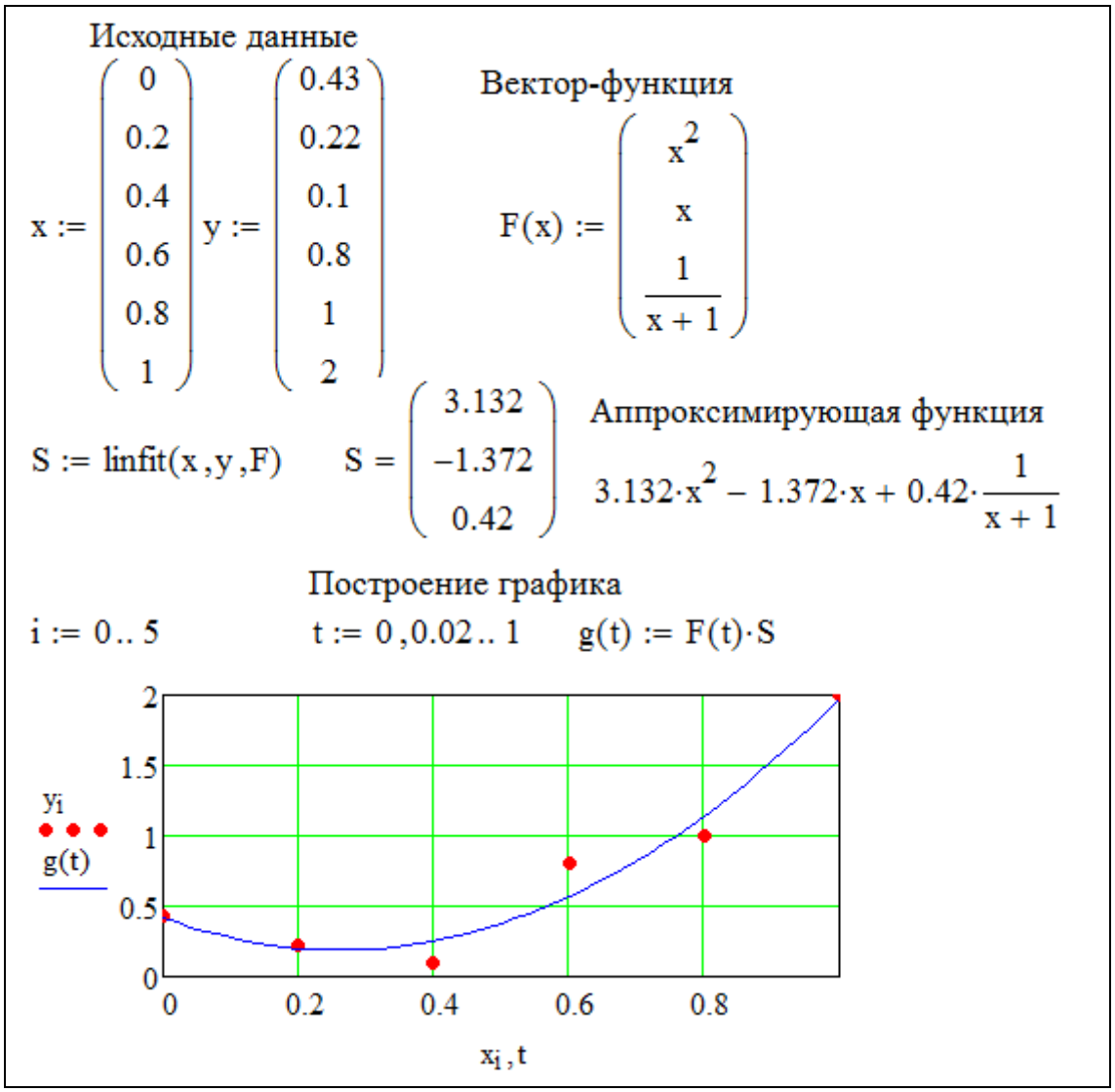


Рисунок 4.5 - Пример использования функции linfit

4.5 Другие виды аппроксимации

В MathCad существует несколько функций, позволяющих выполнять аппроксимацию с использованием зависимостей, наиболее часто встречающихся на практике.

- ✓ **expfit (*vx, vy, vg*)** – аппроксимация экспоненциальной функцией $y = a \cdot e^{bx} + c$;
- ✓ **logfit (*vx, vy, vg*)** – аппроксимация логарифмической функцией $y = a \cdot ln(x + b) + c$;
- ✓ **pwrfit (*vx, vy, vg*)** – аппроксимация степенной функцией $y = a \cdot x^b + c$;
- ✓ **sinfit (*vx, vy, vg*)** – аппроксимация синусоидой функции $y = a \cdot sin(x + b) + c$;
- ✓ **lgsfit (*vx, vy, vg*)** – аппроксимация логистической функцией $y = a/(1 + b \cdot e^{-cx})$;

В данных функциях введен дополнительный аргумент **vg** – это трехкомпонентный вектор, содержащий приблизительные значения параметров *a*, *b* и *c*, входящих в аппроксимирующую функцию. Конечно, в большинстве случаев нет необходимости в точной локализации значений параметров, но в некоторых случаях неправильный выбор элементов вектора **vg** может привести к неудовлетворительному результату аппроксимации. На рисунке 4.6 приведен пример проведения экспоненциальной аппроксимации.

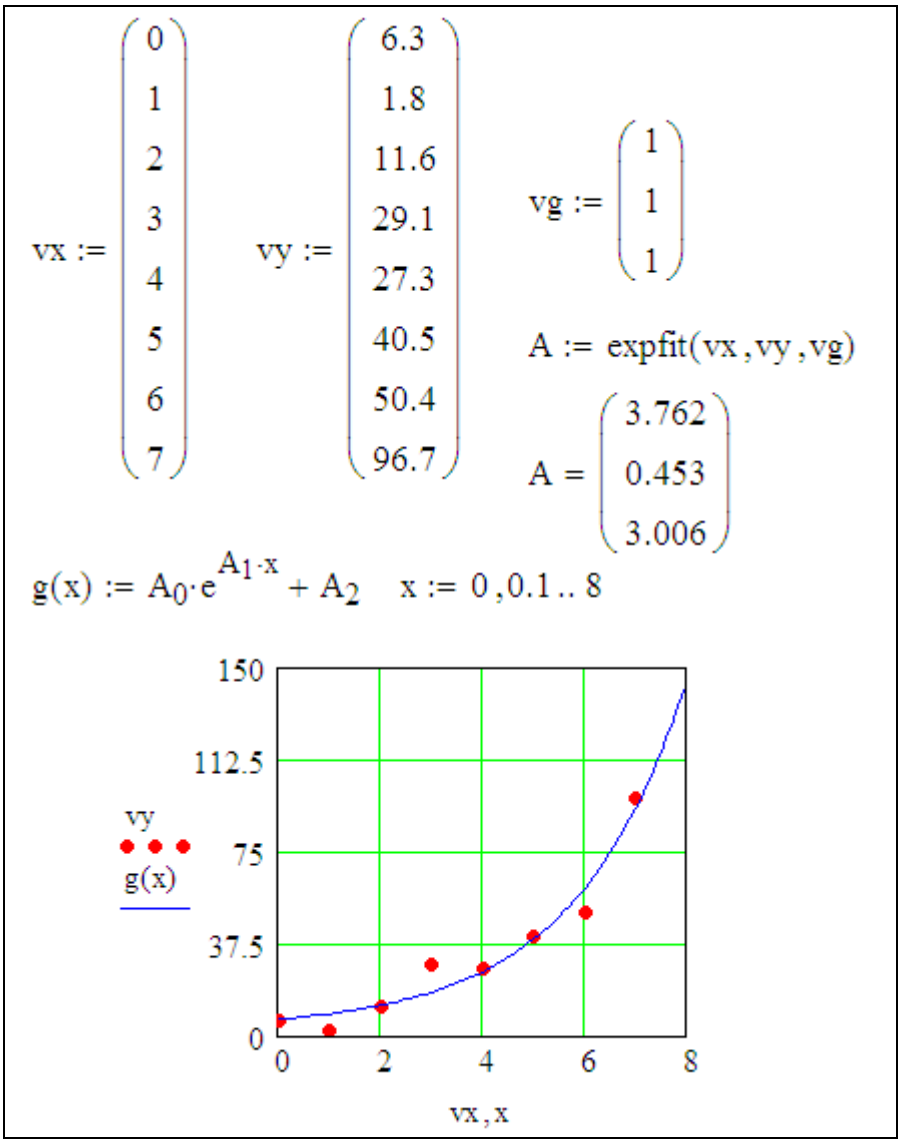


Рисунок 4.6 - Пример использования функции expfit

4.6 Произвольная зависимость

В том случае, если аппроксимирующая зависимость не может быть сведена к линейной, полиномиальной или линейной комбинации других функций, для нахождения ее параметров можно минимизировать сумму квадратов отклонений данных от искомой зависимости с помощью функции Minerr (сумма квадратов отклонений всегда положительна) либо Minimize.

На рисунке 4.7 показан пример, в котором функция Minerr используется, чтобы определить неизвестные параметры в распределении Вейбулла $y=a \cdot b \cdot x^{(b-1)} \cdot exp(-a \cdot x^b)$

$F(x,a,b) := a \cdot b \cdot x^{(b-1)} \cdot \exp\left(-a \cdot x^{b}\right)$ Аппроксимирующая функция

$i := 0..11 \quad x_i := \frac{i}{5} \quad y_i := F(x_i, 0.1, 3)$ Моделирование исходных данных

$SSE(a,b) := \sum_i \left(y_i - F(x_i,a,b)\right)^2$ Сумма квадратов отклонений (минимизируемый функционал)

$a := 0.8 \quad b := 1.5$ Задание начальных значений

Given

$SSE(a,b) = 0$ Блок решения

$\begin{pmatrix} A \\ B \end{pmatrix} := Minerr(a,b) \quad \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0.1 \\ 3 \end{pmatrix}$ Решение задачи

$t := 0, 0.1..2.3$

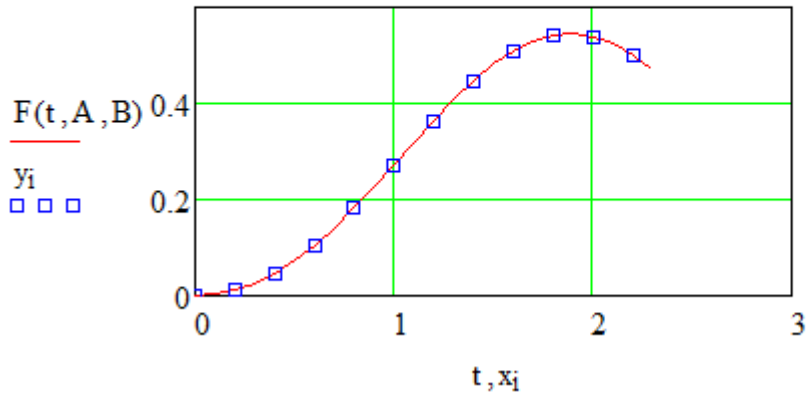


Рисунок 4.7 - Использование функции Minerr для решения задачи аппроксимации

4.7 Функции сглаживания данных

Данные большинства экспериментов имеют случайные составляющие погрешности. Поэтому часто возникает необходимость статистического сглаживания данных. Ряд функций MathCad предназначен для выполнения операций сглаживания данных различными методами.

- ✓ **medsmooth(*vy, n*)** – для вектора с *m* действительными числами возвращает *m*-мерный вектор сглаженных данных по методу скользящей медианы, параметр *n* задает ширину окна сглаживания (***n*** должно быть нечетным числом, меньшим *m*);
- ✓ **ksmooth(*vx, vy, b*)** – возвращает *n*-мерный вектор сглаженных *vy*, вычисленных на основе распределения Гаусса. ***vx*** и ***vy*** – *n*-мерные векторы действительных чисел. Параметр ***b*** (полоса пропускания) задает ширину окна сглаживания (***b*** должно в несколько раз превышать интервал между точками по оси *x*);
- ✓ **supsmooth(*vx, vy*)** – возвращает *n*-мерный вектор сглаженных ***vy***, вычисленных на основе использования процедуры линейного сглаживания методом наименьших квадратов по правилу k-ближайших соседей с адаптивным выбором k. ***vx*** и ***vy*** – *n*-мерные векторы действительных чисел. Элементы вектора ***vx*** должны идти в порядке возрастания.

На рисунке 4.8 показан пример использования функций сглаживания для 50 случайных чисел из отрезка [0, 1].

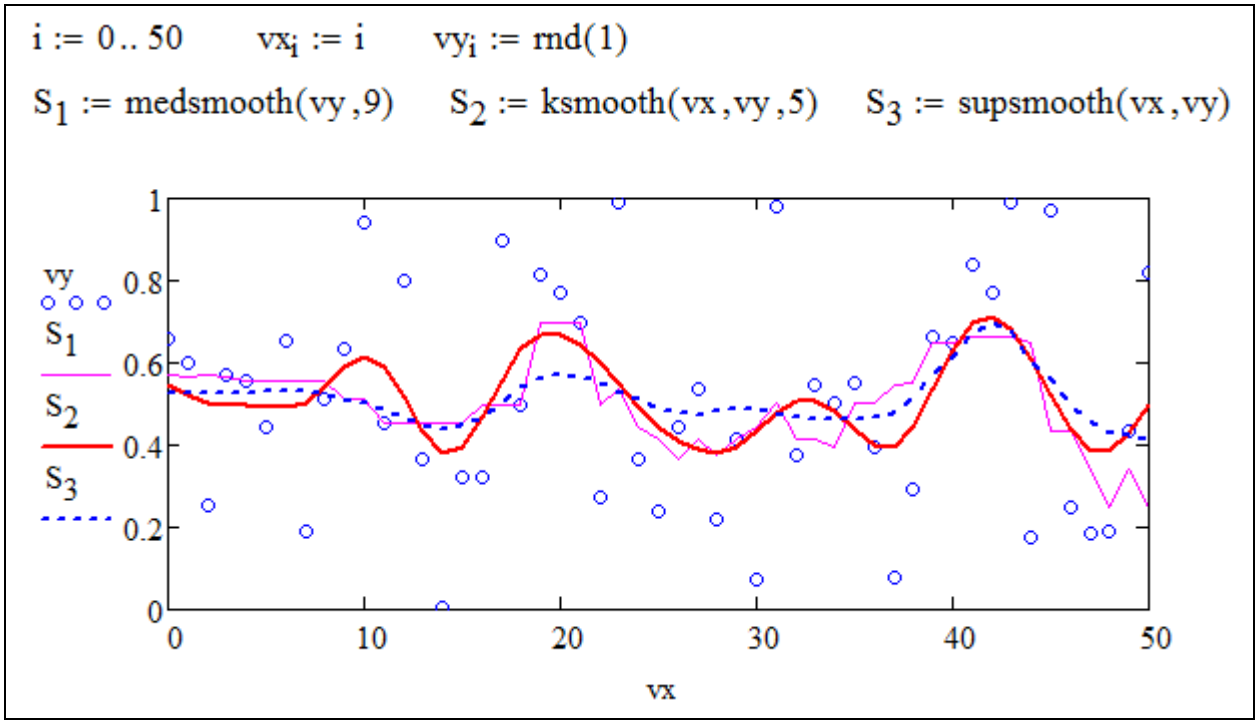


Рисунок 4.8 - Использование функций сглаживания данных

5 РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

5.1 Решение задачи Коши для дифференциальных уравнений

5.1.1 Решение дифференциальных уравнений без использования встроенных функций

Может возникнуть вопрос: а нужно ли создавать свои документы для реализации таких методов? Ответ на него не однозначен. Если ваша цель — решение конкретной задачи, то проще воспользоваться готовыми функциями, которые будут описаны ниже. Однако реализация известных численных методов в системе MathCAD легка и наглядна. Более того, она позволяет вмешиваться в алгоритмическую реализацию методов решения, что способствует созданию новых или улучшенных методов решения дифференциальных уравнений, ориентированных на решение интересующих пользователя задач. По существу приведенные уравнения повторяют известные формулы часто встречающиеся в учебной литературе по численным методам решения дифференциальных уравнений. Кроме этого, «ручная» реализация процесса решения задачи очень полезна студентам как с точки зрения закрепления знаний по вычислительной математике, так и для более глубокого освоения системы MathCAD. На рисунке 5.1 показан пример решения дифференциального уравнения 1-го порядка методами Эйлера и Рунге-Кутта.

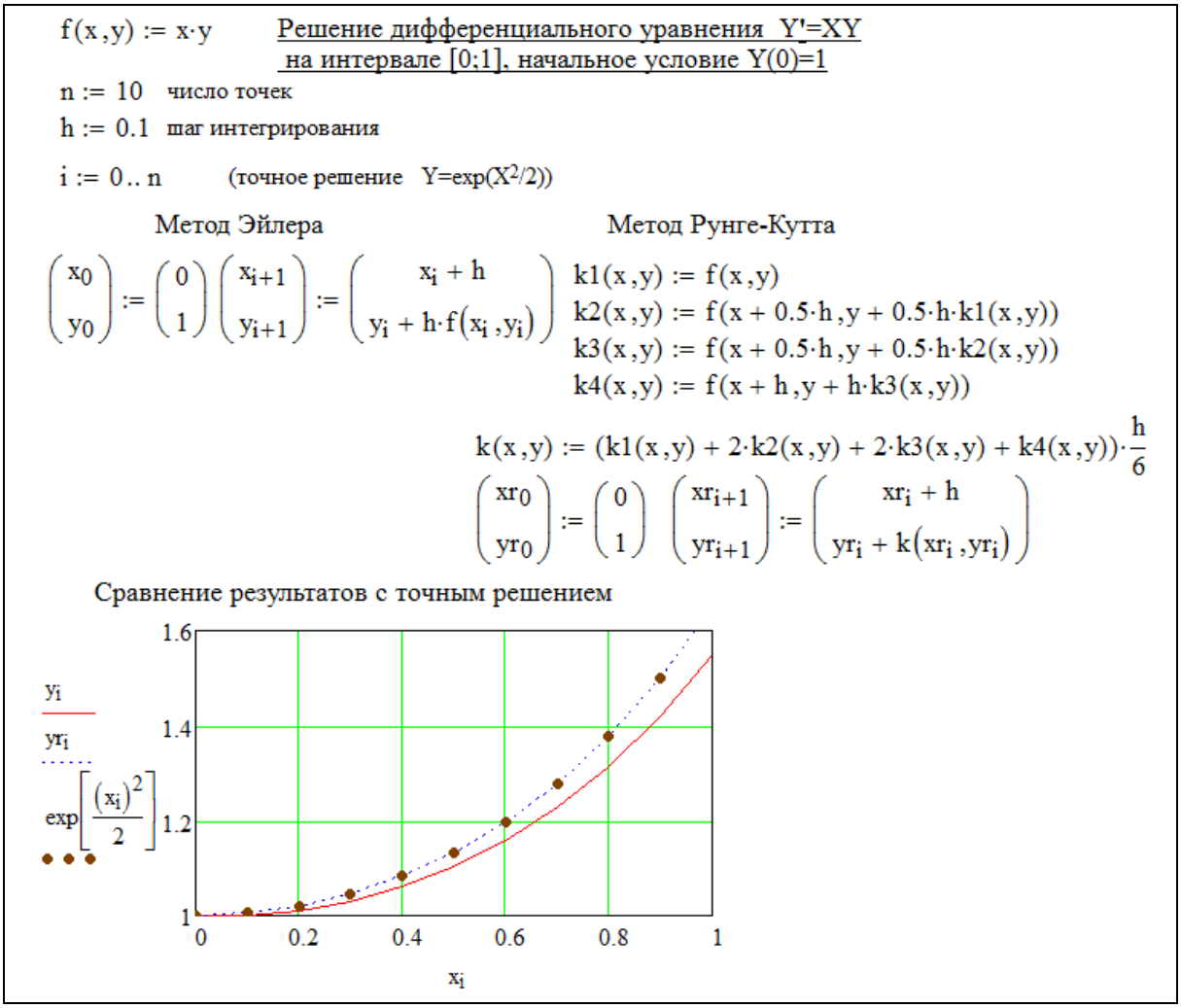


Рисунок 5.1 - Решение дифференциального уравнения методами Эйлера и Рунге-Кутта

*5.1.2 Функция **rkfixed***

Наиболее часто для нахождения решений дифференциальных уравнений используется встроенная функция MathCad rkfixed.

Для решения уравнения с помощью этой функции необходимо ввести в MathCad:

- дифференциальное уравнение, записанное в виде:

$$y' = f(x, y) \qquad (5.1)$$

или систему n уравнений, представленную в форме:

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, ..., y_n) \\ y_2' &= f_2(x, y_1, y_2, ..., y_n) \\ &\cdots\cdots\cdots\cdots\cdots \\ y_n' &= f_n(x, y_1, y_2, ..., y_n) \end{aligned} \qquad (5.2)$$

- начальные условия;
- набор точек, в которых нужно найти решение.

Общий вид функции:

rkfixed (*y, x1, xk, k , D*),

где **y** - вектор начальных условий размера n, где n - число уравнений в системе. Для дифференциального уравнения первого порядка этот вектор вырождается в одну точку y_0 (эта точка должна быть записана именно как нулевой элемент вектора y');

x1 , xk - граничные точки интервала, на котором ищется решение. Начальные условия, заданные в векторе **y**, - это значение решения в точке x1;

k - число точек (не считая начальной), в которых ищется решение. При помощи этого аргумента определяется число строк (k + 1) в матрице, возвращаемой функцией rkfixed;

D (*x, y*) - функция-вектор, представляющая собой вектор из n элементов, содержащих первые производные неизвестных функций (правые части системы (5.2) или уравнения (5.1)).

Функция rkfixed использует для решения метод Рунге-Кутта четвертого порядка с фиксированным шагом. В результате решения получается матрица, имеющая n+1 столбцов:

- первый столбец содержит точки, в которых ищется решение дифференциального уравнения (значения x);
- второй столбец содержит $y_1(x)$, третий - $y_2(x)$ и т.д.

На рисунке 5.2 показаны примеры решения уравнения первого порядка, на рисунке 5.3 - системы двух уравнений.

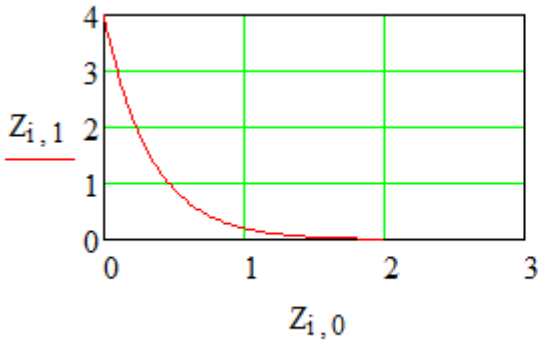
Решение дифференциального уравнения $y' = -3y$
на отрезке [0, 2] начальное условие $y(0) = 4$

$y_0 := 4$ Задание начального значения

$D(x, y) := -3 \cdot y_0$ Определение правой части уравнения

$Z := rkfixed(y, 0, 2, 100, D)$ Решение уравнения в 100 точках

$i := 0 .. 100$



$Z =$

	0	1
0	0	4
1	0.02	3.767
2	0.04	3.548
3	0.06	3.341
4	0.08	3.147
5	0.1	2.963
6	0.12	2.791
7	0.14	2.628
8	0.16	2.475
9	0.18	2.331
10	0.2	...

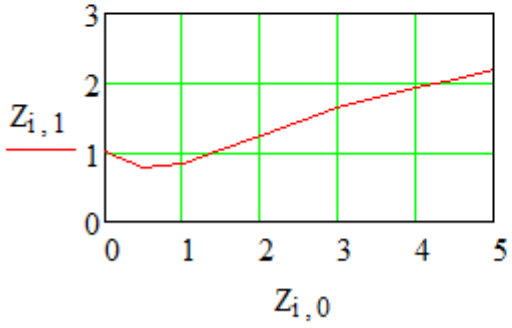
Решение дифференциального уравнения $y' = -y^2 + x$
на отрезке [0, 5] начальное условие $y(0) = 1$

$y_0 := 1$ Задание начального значения.

$D(x, y) := -(y_0)^2 + x$ Определение правой части уравнения

$Z := rkfixed(y, 0, 5, 10, D)$ Решение уравнения в 10 точках

$i := 0 .. 10$



$Z =$

	0	1
0	0	1
1	0.5	0.765
2	1	0.834
3	1.5	1.03
4	2	1.252
5	2.5	1.455
6	3	1.632
7	3.5	1.788
8	4	1.93
9	4.5	2.06
10	5	2.181

Рисунок 5.2 - Решение дифференциального уравнения первого порядка

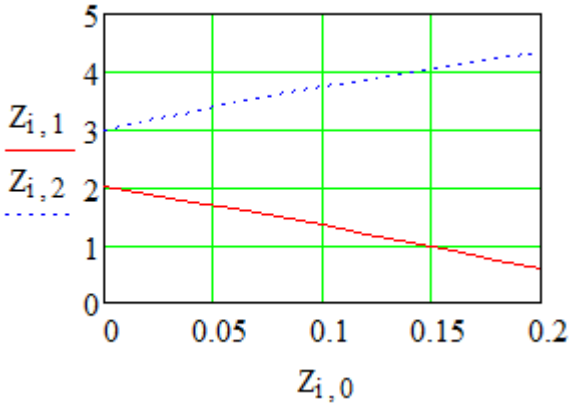
$y := \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ Задание вектора начальных значений

$D(x,y) := \begin{pmatrix} 3\cdot x - 2\cdot y_1 \\ 2\cdot y_0 + 4 \end{pmatrix}$ Определение функции-вектора, содержащей правые части системы

$Z := rkfixed(y,0,0.2,10,D)$ Решение системы в 10 точках

$i := 0..10$

Построение графиков $y_1 = f(x)$ и $y_2 = f(x)$



$Z =$

	0	1	2
0	0	2	3
1	0.02	1.877	3.158
2	0.04	1.75	3.31
3	0.06	1.617	3.457
4	0.08	1.481	3.599
5	0.1	1.339	3.736
6	0.12	1.194	3.867
7	0.14	1.044	3.991
8	0.16	0.891	4.11
9	0.18	0.735	4.223
10	0.2	0.575	4.329

Решить систему диф-ых уравнений $y_1 = \mu\cdot y_1 - y_2 - \left[(y_1)^2 + (y_2)^2\right]\cdot y_1$ $y_1(0)=0$

$y_2 = \mu\cdot y_2 + y_1 - \left[(y_1)^2 + (y_2)^2\right]\cdot y_2$ $y_2(0)=1$

и построить зависимость $y_2 = f(y_1)$ $\mu := -0.2$

$y := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $D(x,y) := \begin{bmatrix} \mu\cdot y_0 - y_1 - \left[(y_0)^2 + (y_1)^2\right]\cdot y_0 \\ \mu\cdot y_1 + y_0 - \left[(y_0)^2 + (y_1)^2\right]\cdot y_1 \end{bmatrix}$

$Z := rkfixed(y,0,20,100,D)$ $i := 0..100$

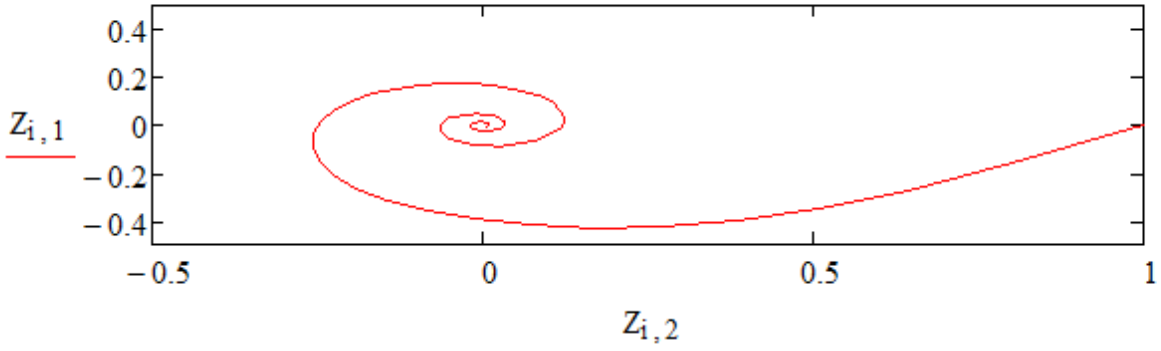


Рисунок 5.3 - Решение системы двух дифференциальных уравнений первого порядка

Для решения или систем уравнений более высоких порядков, их нужно сначала преобразовать к виду (5.2). Заметим, что любое уравнение вида

$$y^{(n)} = f(x, y^{(n-1)}, y^{(n-2)}, \ldots, y)$$

посредством замены

$$y_0(x) = y(x)$$
$$y_2(x) = y'(x)$$
$$\cdots\cdots\cdots\ \cdots\cdots$$
$$y_{n-1}(x) = y^{(n-1)}(x)$$

может быть сведено к виду (5.2):

$$y_0' = y_1(x)$$
$$y_1' = y_2(x)$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$y_{n-1}' = f(x, y_1, y_2, \ldots, y_n)$$

На рисунке 5.4 показан пример решения уравнения четвертого порядка, на рисунке 5.5 - системы двух дифференциальных уравнений второго порядка, сведенных к виду (5.2), по рассмотренной выше методике.

Пример 1. Решить ДУ на отрезке [0, 2]:

$y'' - 2y'' + 4y = 0$

$y(0) = 0 \quad y'(0) = 1 \quad y''(0) = 2 \quad y'''(0) = 3$

Показать решение на графике.

1. Введем обозначения: $y \to y_0, \ y' \to y_1, \ y'' \to y_2, \ y''' \to y_3$
2. Посредством замены, сводим ДУ к системе:

$$\begin{cases} y_0' = y_1 & y_0(0) = 0 \\ y_1' = y_2 & y_1(0) = 1 \\ y_2' = y_3 & y_2(0) = 2 \\ y_3' = 2y_2 - 4y_0 & y_3(0) = 3 \end{cases}$$

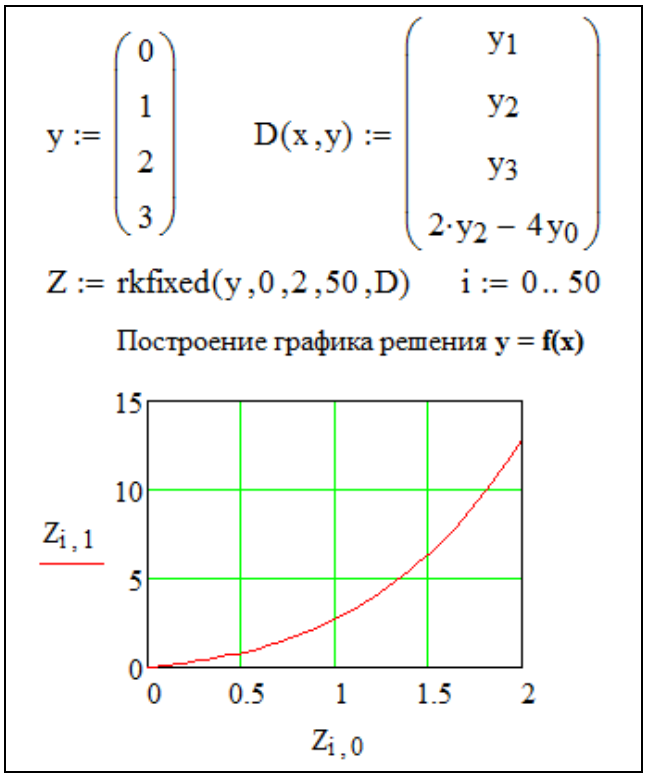


Рисунок 5.4 - Решение дифференциального уравнения четвертого порядка

Пример 2. Решить систему ДУ на отрезке [0, 0.4]:

$$\begin{cases} u'' = 2v & u(0) = 1.5 \;\; u'(0) = 1.5 \\ v'' = 4v - 2u & v(0) = 1 \quad\; v'(0) = 1 \end{cases}$$

Построить зависимости u(x), v(x).

1. Введем обозначения: $u \to y_0,\;\; u' \to y_1,\;\; v \to y_2,\;\; v' \to y_3$
2. Посредством замены, сводим данную систему ДУ к системе:

$$\begin{cases} y_0' = y_1 & y_0(0) = 1.5 \\ y_1' = 2y_2 & y_1(0) = 1.5 \\ y_2' = y_3 & y_2(0) = 1 \\ y_3' = 4y_2 - 2y_0 & y_3(0) = 1 \end{cases}$$

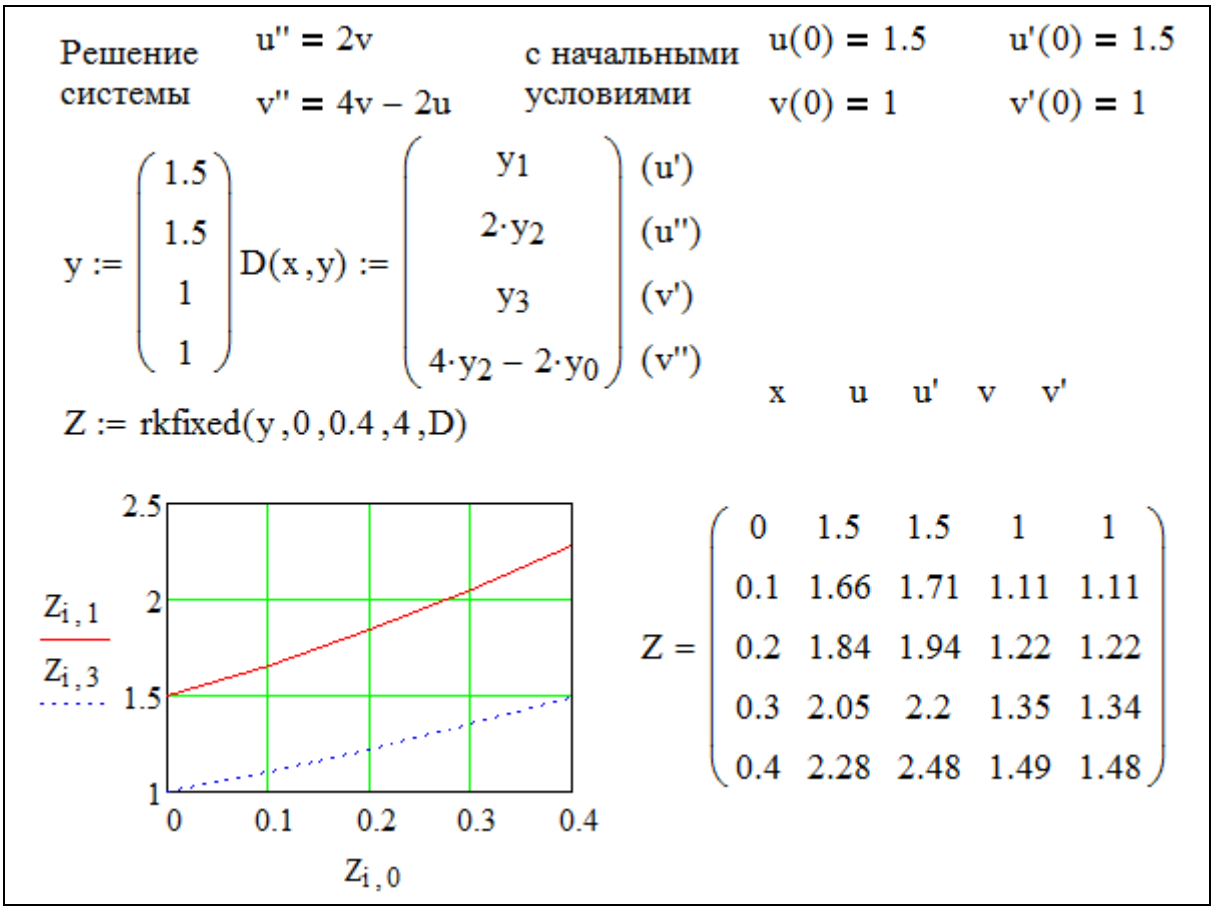


Рисунок 5.5 - Решение системы дифференциальных уравнений второго порядка

*5.1.3 Функция **odesolve***

В системе MathCad 2000 появилась новая, чрезвычайно удобная функция для решения дифференциальных уравнений, которая используется совместно с ключевым словом **Given** и имеет вид:

odesolve(*x,b,[step]*)

Аргументы функции:

x – переменная интегрирования;

b – конечная точка отрезка интегрирования;

step - необязательный параметр, число шагов интегрирования.

Теперь запись решения задачи Коши вполне естественна: пишется ключевое слово **Given,** затем дифференциальное уравнение и начальные условия. Функция odesolve, завершающая эту конструкцию, возвращает функцию – решение задачи. У

этой функции нельзя просмотреть ее аналитический вид, но тем не менее это полноценная MathCad-функция пользователя, которую можно протабулировать, построить график, найти у нее особые точки – корни, минимум, максимум. На рисунке 5.6 показан пример использования данной функции для решения дифференциального уравнения первого порядка.

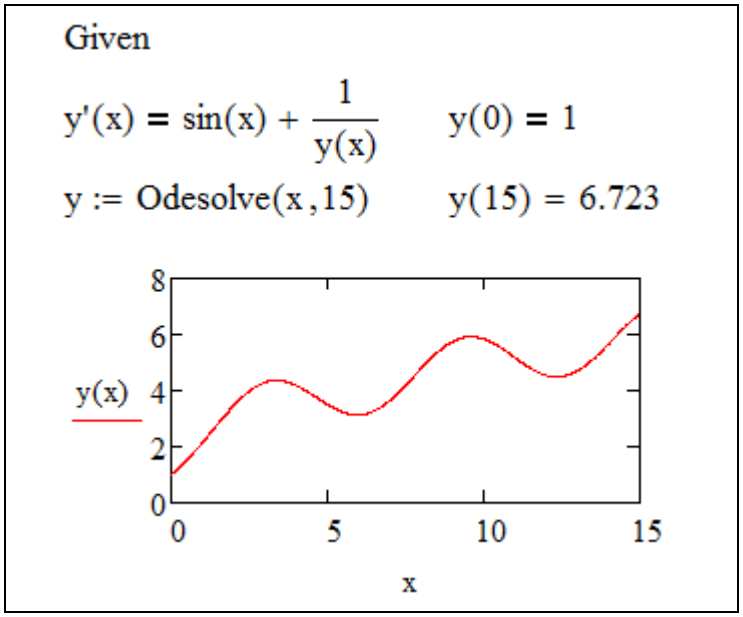


Рисунок 5.6 - Решение дифференциального уравнения первого порядка с помощью odesolve

Дифференциальное выражение может быть записано с использованием операторов дифференцирования **d/dx**, $\mathbf{d^2/dx^2}$ или с помощью знака производной **y′(x)**, который вводится с помощью клавиатурной комбинации **Ctrl/F7**.

Условия, которые задаются в блоке решения, могут быть вида $y(a)=b$ или $y'(a)=b$, но нельзя задавать ограничения вида $y'(a)+y(a)=b$.

Функция odesolve предназначена для решения уравнений, но не систем, поэтому при необходимости решить систему с помощью этой функции, ее сначала нужно преобразовать к одному уравнению по изложенной выше методике.

На рисунке 5.7 показан пример решения уравнения 3-го порядка с помощью функции odesolve.

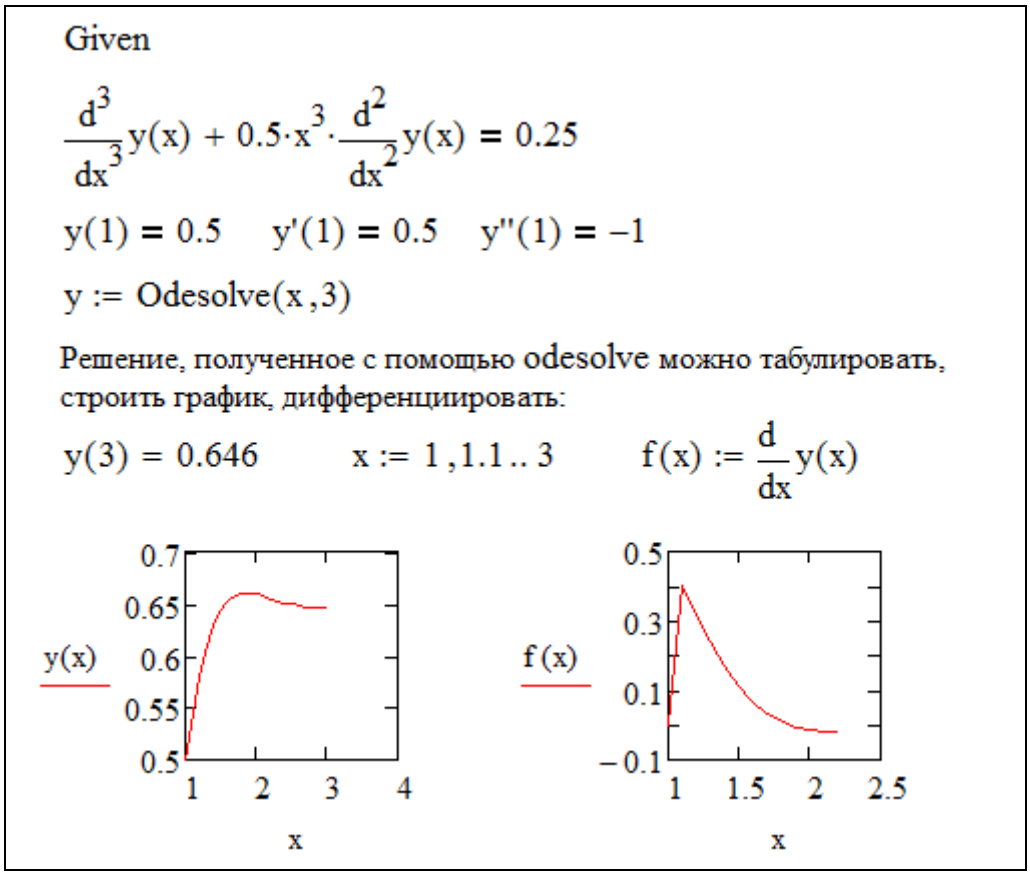


Рисунок 5.7 - Решение дифференциального уравнения третьего порядка с помощью odesolve

*5.1.4 Функции **Bulstoer**, **Rkadapt***

Когда известно, что решение системы является гладкой функцией, то для решения дифференциальных уравнений лучше использовать функцию

Bulstoer (*y, x1, xk, k, D*),

куда заложен метод Булириш-Штера, а не Рунге-Кутта, используемый функцией rkfixed. В этом случае решение будет точнее.

Решение дифференциального уравнения удобнее использовать, если вычислять его достаточно часто на интервале, где искомая функция меняется быстро, и не очень часто - где функция меняется медленно. Для этого можно использовать функцию

Rkadapt (*y, x1, xk, k, D*).

В отличие от функции rkfixed, которая ищет решение с постоянным шагом, функция Rkadapt проверяет, как быстро изменяется приближенное решение, и, соответственно, адаптирует размер шага. Этот адаптивный контроль дает возможность данной функции вычислять решение на более мелкой сетке в тех областях, где оно меняется быстро, и на более крупной - в тех областях, где оно меняется медленно. Это позволяет повысить точность и сократить время, требуемое для решения уравнения.

Обратите внимание, что, хотя функция Rkadapt при решении использует во внутренних расчетах переменный шаг, возвращает решение она на равномерной сетке.

Список аргументов, матрица решения и порядок использования этих функции те же самые, что и для rkfixed.

5.1.5. Функции ***bulstoer, rkadapt***

Функции, описанные выше, искали решение *y(x)* в равноотстоящих точках на отрезке [x1, xk]. Однако, иногда возникает задача, когда необходимо найти приближенное решение только в одной конечной точке интервала. Хотя функции, описанные выше, позволяют вычислить *y*(xk), но при этом они будут делать дополнительную работу, возвращая промежуточные значения *y(x*). Если необходимо вычислить только значение *y*(xk), лучше использовать функции, перечисленные ниже. Каждая из них соответствует одной из функций, описанной в предыдущих разделах и обладает аналогичными свойствами.

Функции имеют вид:

bulstoer (*y, x1, xk, e, D, kmax, save*)
rkadapt (*y, x1, xk, e, D, kmax, save*) ,

где ***y*** - вектор начальных условий;
x1, xk - граничные точки интервала, на котором ищется решение;
D - функция-вектор, описывающая исходную систему;
e - параметр, контролирующий точность решения. Малое значение *e* определяет меньшие шаги вдоль траектории, что увеличивает точность решения. Значение *e*, равное 10^{-3} - 10^{-5} обычно дает хороший результат;
kmax - максимальное число промежуточных точек, в которых ищется промежуточное решение. Значение kmax содержит ограничение сверху на число строк матрицы, возвращаемой этими функциями;
save - минимально допустимый интервал между точками, в которых ищется решение. Он определяет нижнюю границу различия между любыми двумя числами в первом столбце матрицы, возвращаемой функцией.

На рисунке 5.8 показан пример решения уравнения в одной точке.

Решение дифференциального уравнения
$y'' = x \cdot \exp(x)$ в одной точке x=1.2
начальное условие $y(0) = 1$ $y'(0) = 0$

$$y := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad D(x,y) := \begin{pmatrix} y_1 \\ x \cdot e^x \end{pmatrix}$$

$Z := \text{rkadapt}(y, 0, 1.2, 0.001, D, 100, 1.2)$

$$Z = \begin{pmatrix} 0 & 1 & 0 \\ 1.2 & 1.544 & 1.664 \end{pmatrix} \qquad Z_{1,1} = 1.544$$

Рисунок 5.8 - Решение дифференциального уравнения с помощью rkadapt

5.2 Двухточечные краевые задачи

Достаточно часто встречаются задачи, в которых значения решения известны в граничных точках интервала, например, к такой задаче сводится описание натянутой струны, закрепленной на концах. Задачи такого вида называются краевыми.

Предположим, что заданы не все начальные условия в начальной точке интервала, как в задаче Коши, но зато известны дополнительно значения решения и/или некоторых его производных в другой точке интервала. В частности, если:

- задано дифференциальное уравнение *n*-го порядка;
- в начальной точке x1 интервала задана только часть информации о значениях решения и некоторых его производных;
- в конце интервала xk известны некоторые (не все) значения решения и некоторых его производных;
- общее количество условий, заданных в точках x1 и x2, равно *n*.

В этом случае необходимо использовать функцию sbval, чтобы найти недостающие начальные условия в точке x1. После того, как эти недостающие начальные условия найдены, можно будет решать обычную задачу Коши любым методом, рассмотренным выше.

Пример, приведенный на рисунке 5.9 показывает, как использовать функцию sbval. Обратите внимание, что она не возвращает решение дифференциального уравнения, а только вычисляет недостающие начальные условия. При этом они вычисляются соответствующими тем значениям, которые были заданы в точке xk. Далее нужно использовать эти найденные начальные условия, и решать обычную задачу Коши.

Решение дифференциального уравнения $y'''' + y = 0$

удовлетворяющее заданным краевым условиям

$y(0) = 0 \quad y'(0) = 7 \quad y(1) = 1 \quad y'(1) = 10 \quad y''(1) = 5$

$$v := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} y''(0) \\ y'''(0) \\ y''''(0) \end{matrix}$$

Начальные значения для поиска величин

$$load(x1, v) := \begin{pmatrix} 0 \\ 7 \\ v_0 \\ v_1 \\ v_2 \end{pmatrix}$$

$$D(x, y) := \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ -y_0 \end{pmatrix} \qquad score(xk, y) := \begin{pmatrix} y_0 - 1 \\ y_1 - 10 \\ y_2 - 5 \end{pmatrix}$$

Недостающие начальные условия, найденные для функции rkfixed

$$S := sbval(v, 0, 1, D, load, score) \qquad S = \begin{pmatrix} -85.014 \\ 348.107 \\ -516.257 \end{pmatrix} \begin{matrix} y''(0) \\ y'''(0) \\ y''''(0) \end{matrix}$$

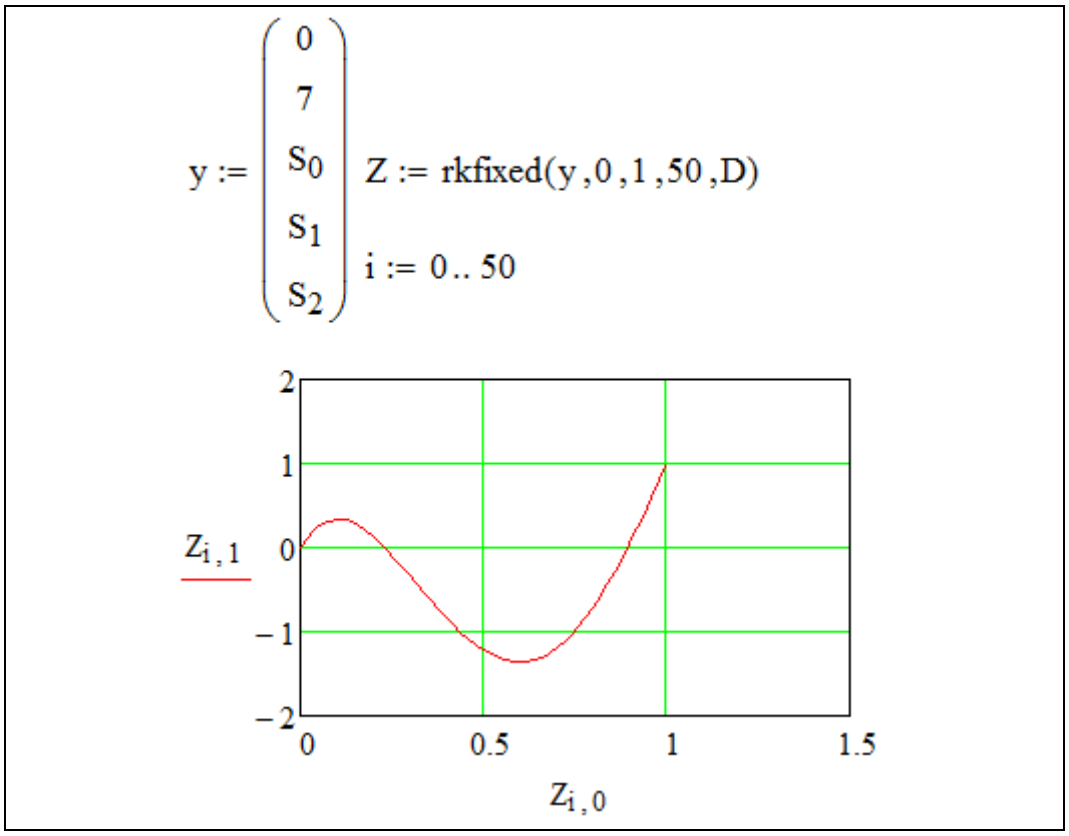


Рисунок 5.9 - Пример использования функции sbval для подготовки краевой задачи к решению

Функция sbval возвращает вектор, содержащий недостающие начальные условия в точке x1 и имеет вид:

sbval (*v, x1, xk, D, load, score*) ,

где **v** - вектор начальных приближений для искомых недостающих начальных значений в точке x1;

x1, xk - граничные точки интервала, на котором ищется решение;

D(*x, y*) - функция-вектор, описывающая решаемое уравнение или систему, как это описано в разделе 5.1.1.;

load(x1, **v**) - функция-вектор, возвращающая значения начальных условий в точке x1. Вектор состоит из *n* элементов, некоторые из которых будут константами, определяемыми заданными начальными условиями. Другие элементы будут неизвестными и будут найдены функцией sbval. Если значение начального условия неизвестно, необходимо использовать вектор, соответствующий вектору начальных приближений из **v** ;

score(xk, *y*) - функция-вектор, количество элементов которого равно числу элементов вектора **v**. Каждый элемент вектора содержит разность между начальным условием, заданным в точке xk, и значением искомого решения в этой точке. Компоненты этого вектора показывают, насколько значения найденного в точке xk решения близки к значениям, заданным в xk.

Как видно из приведенного примера, решение краевой задачи с помощью функции sbval представляет достаточные трудности. Эта задача сама по себе довольно сложная, и вдобавок механизм реализации решения в среде MathCad довольно запутанный.

Для устранения этого недостатка, разработчики не только вставили в MathCad функцию odesolve, но и возложили на нее возможность использования для решения краевой задачи. Порядок применения этой функции практически тот же самый, что и для решения задачи Коши. Пример решения краевой задачи, приведенной на рисунке 5.9, с помощью функции odesolve показан на рисунке 5.10.

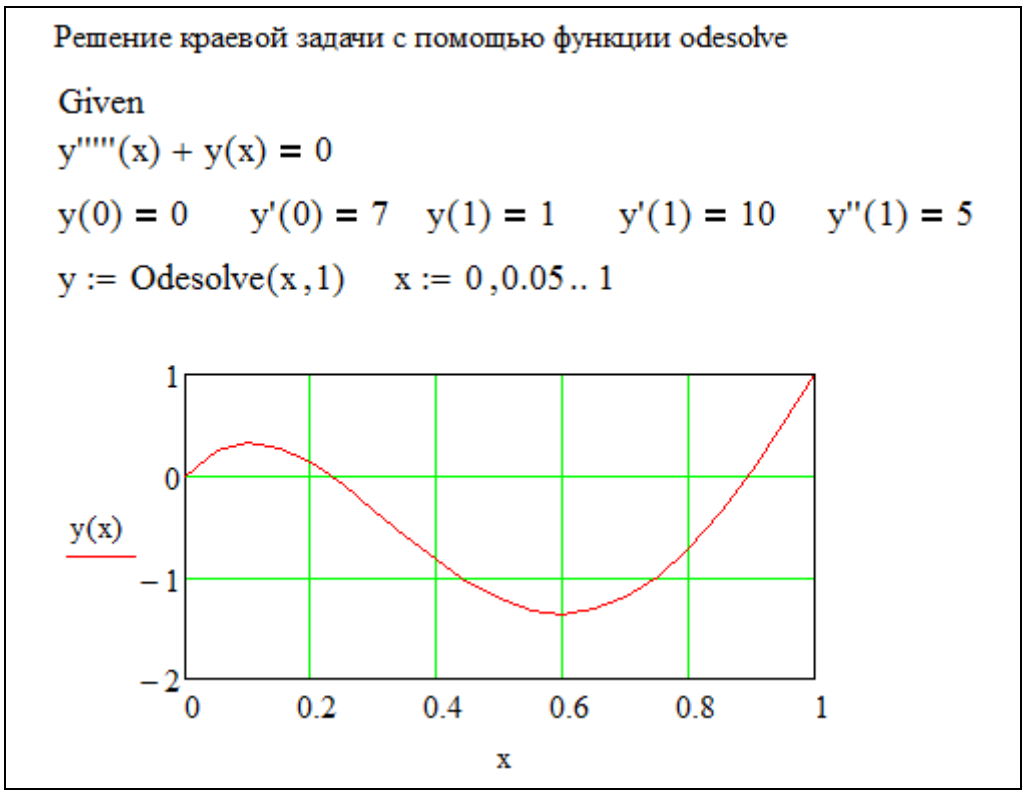


Рисунок 5.10 - Пример решения краевой задачи с помощью функции odesolve

К сожалению, некоторые краевые задачи, которые успешно решаются в MathCad с помощью комбинации функций sbval и rkfixed, приводят к сообщению об ошибке, если используется функция odesolve.

В этом случае рекомендуется поэкспериментировать со значениями параметра step (см. раздел 5.1.3), чтобы повысить точность решения. Можно также попытаться изменить применяемый при работе odesolve метод интегрирования: с фиксированным шагом (fixed step) или с адаптирующимся шагом (adaptive step size) , дважды щелкнув мышкой на слове odesolve и отмечая Fixed или Adaptive в контекстном меню.

6 СТАТИСТИЧЕСКИЕ ФУНКЦИИ

6.1 Статистические оценки совокупностей

MathCad содержит несколько функций, предназначенных для вычисления статистических оценок случайных совокупностей. Все функции могут быть применены к векторам либо матрицам. В последующих описаниях *m* и *n* представляют число строк и столбцов рассматриваемых массивов. Во всех используемых формулах переменная ORIGIN считается равной нулю.

Таблица 6.1 - Функции вычисления статистических оценок

Функция	Описание
mean (**A**)	среднее значение элементов массива **A** размера *m* x *n* по формуле $$mean(\mathbf{A}) = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_{i,j}$$
median (**A**)	медиана элементов массива **A**, т.е. величина, выше и ниже которой в вариационном ряду находится одинаковое количество членов. Если число элементов массива четное, то медиана определяется как среднее арифметическое двух центральных величин.
var (**A**)	дисперсия элементов массива **A** $$var(\mathbf{A}) = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left(A_{i,j} - mean(\mathbf{A})\right)^2$$
stdev (**A**)	среднеквадратичное отклонение (квадратный корень из дисперсии элементов массива **A** $$stdev(\mathbf{A}) = \sqrt{var(\mathbf{A})}$$
corr (**A**, **B**)	коэффициент корреляции для двух массивов **A**, **B .**

6.2 Законы распределения случайных величин

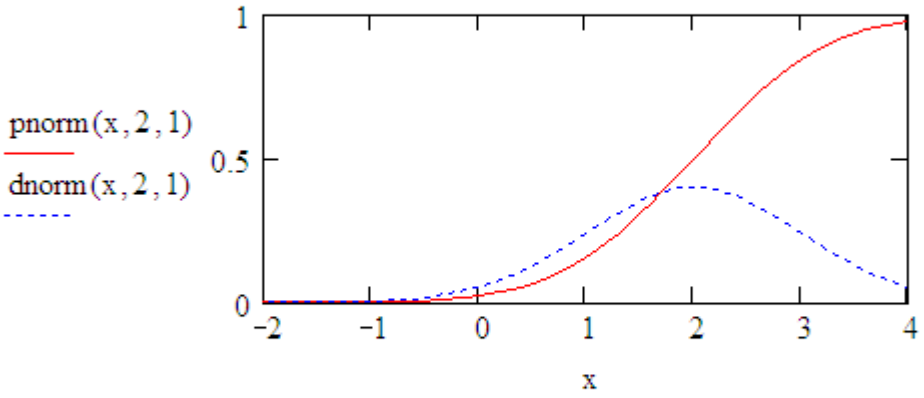
MathCad использует несколько функций для работы с распространенными законами распределения случайных величин. Эти функции распадаются на три класса:

- плотность распределения вероятности *f(x)*. Эта функция показывают отношение вероятности того, что случайная величина попадает в малый диапазон значений *dx*, к величине этого интервала;
  Плотность распределения вероятности - производная от соответствующей функции распределения
  $$f(x) = F'(x) ;$$
- функция распределения F(*x*) определяет вероятность того, что случайная величина X будет принимать значение, меньшее фиксированного числа *x*, т.е.
  $$F(x) = P(X < x)$$
- обращение функций распределения позволяет по заданной вероятности p вычислить такое значение *x*, что P (X < *x*) = p.

В таблице 6.2 приведены эти три класса функций для наиболее часто используемых законов распределения случайных величин. На рисунке 6.1 изображена связь между плотностью вероятности и функцией распределения случайной величины.

$x := -10, -9.9 .. 10$

Построить плотность распределения и функцию распределения для нормального распределения с m=2, s=1



Определить x, чтобы площадь под пунктирной кривой, лежащая слева от x, была равна 0.5 $qnorm(0.5, 2, 1) = 2$

Показать, что pnorm и qnorm - взаимно обратные функции $pnorm(2, 2, 1) = 0.5$

Рисунок 6.1 - Пример использования функций для работы с законами распределения случайных величин

Таблица 6.2 - Функции для работы с различными законами распределения случайных величин.

Вид распределения	Плотность распределения	Функция распределения	Обращение функции распределения
Биноминальное	**dbinom (*k*, *n*, *p*)** $\frac{n!}{k!(n-k)!}p^k(1-p)^{n-k}$	**pbinom (*k*, *n*, *p*)**	**qbinom (*p*, *n*, *r*)** r - вероятность успеха в одиночном испытании
ХИ - квадрат	**dchisq (*x*, *d*)** $\frac{e^{-x/2}}{2\cdot e\cdot(d/2)}\left(\frac{x}{2}\right)^{\frac{d}{2}-1}$	**pchisq (*x*, *d*)**	**qchisq (*p*, *d*)**
Нормальное	**dnorm (*x*, *μ*, *σ*)** $\frac{1}{\sqrt{2\cdot\pi}\cdot\sigma}exp\left(-\frac{(x-\mu)^2}{2\cdot\sigma^2}\right)$ $\sigma > 0$	**pnorm (*x*, *μ*, *σ*)**	**qnorm (*p*, *μ*, *σ*)**
Пуассона	**dpois (*k*, *λ*)** $\frac{\lambda^k}{k!}e^{-\lambda}$ $\lambda > 0$	**ppois (*k*,*λ*)**	**qpois (*p*,*λ*)**

Вид распределения	Плотность распределения	Функция распределения	Обращение функции распределения
t - распределение Стьюдента	**dt (*x, d*)** $\frac{e\cdot\left(\frac{d+1}{2}\right)}{e\cdot\left(\frac{d}{2}\right)\cdot\sqrt{\pi\cdot d}}\left(1+\frac{x^2}{d}\right)^{-\frac{d+1}{2}}$ d > 0	**pt (*x, d*)**	**qt (*p, d*)**
Равномерное	**dunif (*x, a, b*)** $\frac{1}{b-a}\quad a\le x\le b$	**punif (*x, a, b*)**	**qunif (*p, a, b*)**

6.3 Функция hist

Для вычисления частотного распределения, применяемого при построении гистограмм в MathCad существует функция

hist (*int, A*),

которая возвращает вектор, представляющий частоты, с которыми величины, содержащиеся в векторе **A**, попадает в интервалы, определяемые вектором **int**. Элементы в **A** и **int** должны быть вещественными. Элементы вектора **int** должны быть расположены в порядке возрастания. Возвращаемый результат - вектор, содержащий на один элемент меньше, чем **int**.

MathCad интерпретирует **int** как набор точек, определяющих последовательность интервалов в гистограмме. В возвращаемом функцией hist векторе ***f***, f_i - есть число значений val из **A**, удовлетворяющих условию:

$$int_i \le val < int_{i+1}$$

Данные, меньшие чем первое значение в **int** и большие, чем последнее значение - игнорируются. На рисунке 6.2 показан пример использования гистограммы в MathCad.

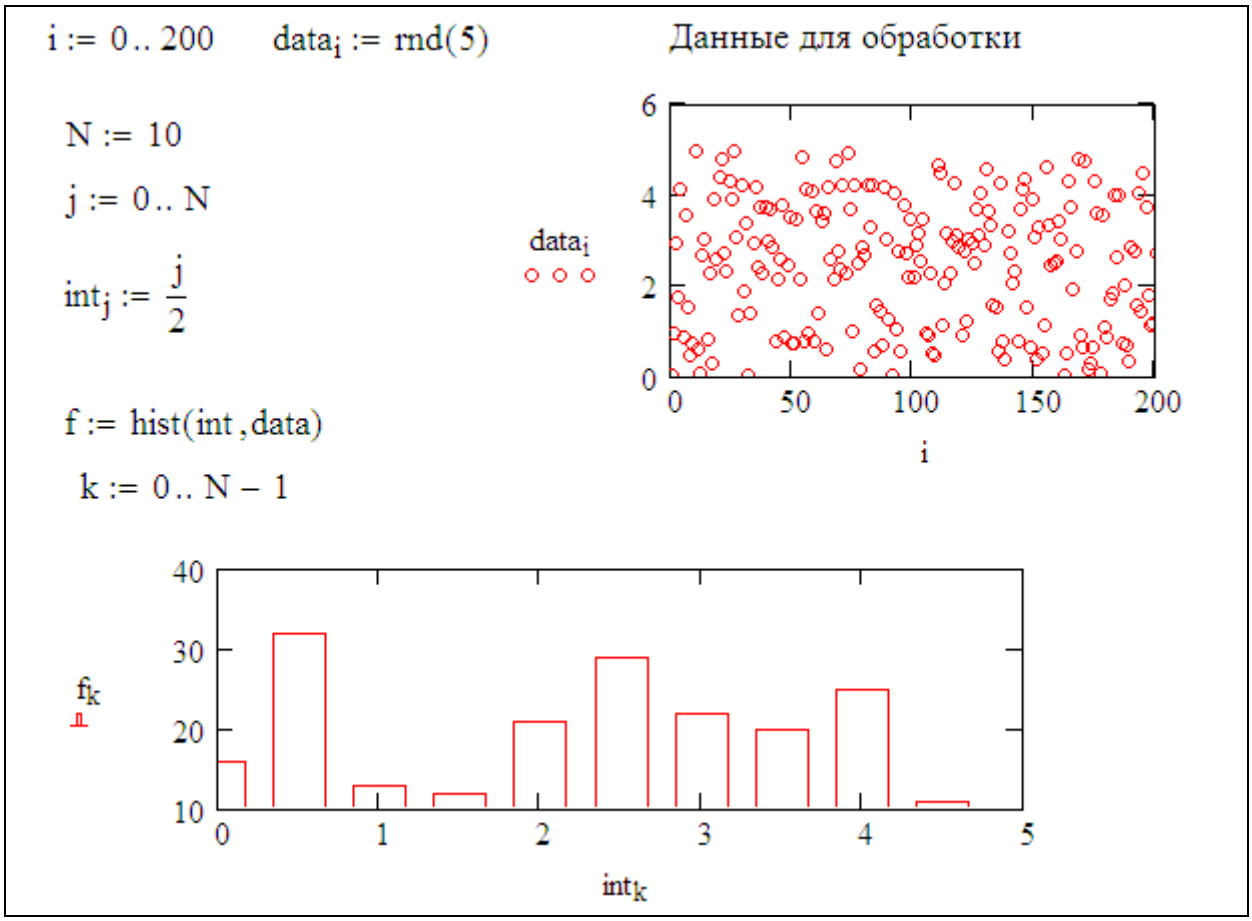


Рисунок 6.2 - Пример использования гистограмм в MathCad

6.4 Случайные числа

В MathCad существует набор функций для генерирования случайных чисел, имеющих разнообразные распределения вероятностей. Во всех этих функциях, представленных в таблице 6.3, первый аргумент *m* есть число случайных точек, содержащихся в векторе, который возвращает соответствующая функция, остальные аргументы - параметры распределения.

Таблица 6.3 - Функции для генерирования случайных чисел

Вид распределения	Функция
Биноминальное	rbinom (m, n, p)
ХИ - квадрат	rchisq (m, d)
Нормальное	rnorm (m, μ, σ)
Пуассона	rpois (m, λ)
Стьюдента	rt (m, d)
Равномерное	runif (m, *a*, b) rnd (x) эквивалентно runif (0, 1, x)

Каждый раз, когда повторно вычисляется выражение, содержащее одну из этих функций, MathCad генерирует новые случайные числа. Чтобы заставить MathCad получить новые случайные числа в уже использованном выражении, нужно щелкнуть мышью на этом выражении и нажать клавишу [**F9**].

Любая последовательность случайных чисел всегда связана с некоторым начальным значением. Каждое нажатие [**F9**] заставляет функцию выдать новое значение из этой последовательности. Одно и то же начальное значение производит одинаковые последовательности чисел. Изменение начального значения приводит к смене последовательности случайных чисел, выдаваемых функцией.

Чтобы изменить это значение, нужно выбрать команду **Математика/Параметры/Переменные** и изменить начальное значение в диалоговом окне.

Чтобы перезапустить генератор случайных чисел MathCad, не изменяя стартового значения, нужно выбрать команду **Математика/Параметры/Переменные** и нажать “ОК”, чтобы принять текущее значение. Затем необходимо выделить щелчком мыши выражение с функцией, генерирующей случайное число и нажать [**F9**].

Так как генератор случайных чисел был сброшен, MathCad будет производить те же самые случайные числа, которые генерировались бы после перезапуска MathCad.

Если нужно несколько раз использовать одну и ту же последовательность случайных чисел, необходимо сбрасывать генератор случайных чисел между вычислениями, как это было описано выше.

На рисунке 6.3 показан пример использования генератора случайных чисел, на рисунке 6.4 - как создать вектор случайных чисел, имеющих заданное распределение.

Полярный график, на котором точки имеют случайные координаты

$i := 0 .. 350$ $\theta_i := rnd(2 \cdot \pi)$ $r_i := rnd(1)$

i =	r_i =	θ_i =
0	0.818	0.338
1	0.136	3.297
2	0.02	0.595
3	0.748	5.602
4	0.902	0.921
5	0.768	5.853
6	0.631	0.296
7	0.682	2.114
8	0.5	2.504
9	0.266	5.945
10	0.647	3.357
11	0.264	4.358
12	0.46	1.628

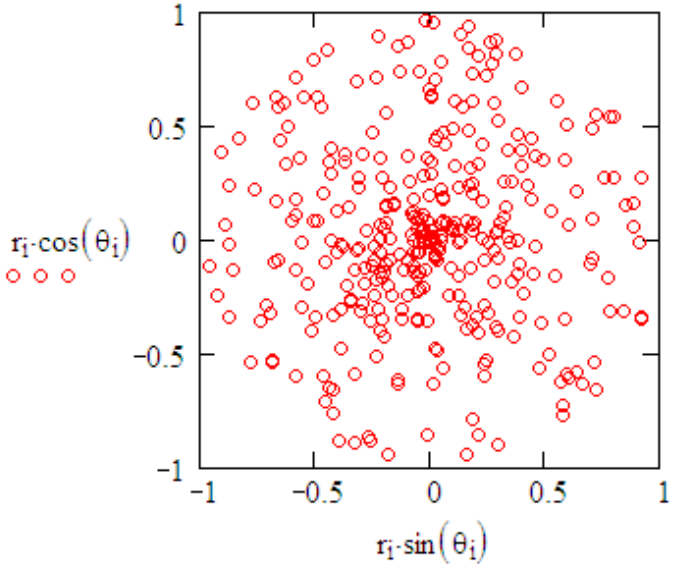


Рисунок. 6.3 - Равномерно распределенные случайные числа

Создание вектора нормально распределенных случайных чисел

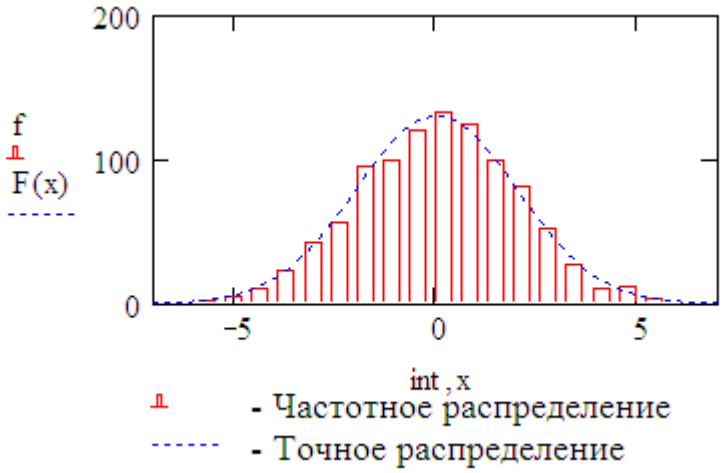
$n := 1000 \quad \mu := 0 \quad \sigma := 2 \quad bin := 20$ число столбцов гистограммы

$N := rnorm(n, \mu, \sigma)$

$lower := floor(min(N)) \quad upper := ceil(max(N)) \quad h := \frac{upper - lower}{bin}$

$j := 0 .. bin \quad int_j := lower + h \cdot j \quad k := 0 .. bin - 1$

$f := hist(int, N) \quad int := int + 0.5 \cdot h \quad F(x) := n \cdot h \cdot dnorm(x, \mu, \sigma)$



- Частотное распределение

- Точное распределение

Рисунок 6.4 - Вектор случайных чисел, распределенных по нормальному закону

7 СИМВОЛЬНЫЕ ВЫЧИСЛЕНИЯ

Кроме численного решения задач, MathCad может производить некоторые вычисления в аналитическом виде. Выражения можно аналитически разлагать на множители, производить дифференцирование и интегрирование, разлагать функции в ряд и т.д.

Есть два основных способа выполнить символьное преобразование выражений:

- использование символьного знака равенства;
- выполнение преобразований с помощью меню **Symbolic** (Символы).

Имеется весьма существенное различие между этими двумя способами преобразований. При использовании символьного знака равенства полученные результаты вычисляются заново всякий раз при внесении любых изменений в рабочий документ, точно так же как и для численных выражений. Результат, полученный с использованием меню, автоматически модифицироваться не будет. Чтобы получить новый ответ в этом случае, нужно снова выделить исходное выражение, которое только что изменили, и снова выбрать соответствующий пункт из меню **Symbolic** (Символы).

Не следует ожидать очень многого от работы с аналитическими выражениями, т.к. при работе с символьными вычислениями обнаружится, что, во-первых, многие вычисления могут быть выполнены только численно, а во-вторых еще больше вычислений возвращают такие длинные ответы, что оказывается удобнее выполнять их численно (попробуйте, например, аналитически решить уравнение $x^3 + a\,x + 1 = 0$).

7.1 Преобразования с использованием символьного знака равенства

Для выполнения символьных преобразований этим способом нужно:

- ввести выражение, которое нужно упростить;
- нажать комбинацию клавиш “**Ctrl/·**“ (клавиша “**Ctrl**” вместе с точкой). MathCad отобразит символьный знак равенства - стрелку “→”;
- щелкнуть мышью вне выражения. Справа от стрелки MathCad отобразит упрощенную версию первоначального выражения. Если выражение не может быть упрощено, то оно будет просто повторено справа от стрелки.

Символьный знак равенства является оператором, подобным любому оператору MathCad. Когда делаются изменения где либо выше или левее от него, MathCad модифицирует результат.

На рисунке 7.1 показаны некоторые примеры использования этого оператора. Обратите внимание, что оператор “→” применяется только ко всему выражению, нельзя применять этот оператор ни к части выражения, ни к результату предыдущего действия.

Следует обратить внимание на то, что понятие *упростить* является весьма относительным. Поэтому это понятие можно уточнить, помещая одно из ключевых слов MathCad, перед выражением, содержащим “→”. В MathCad 6 таких слов было 7, в среде MathCad 11 их количество существенно увеличилось до 28 и, в связи с этим, появилась новая палитра символьной математики, показанная на рисунке 7.2. Символьный знак равенства с уточнением операции имеет вид $\blacksquare\rightarrow$ Черный квадратик перед стрелкой предназначен для ввода в него соответствующего ключевого слова. Ввод этого оператора осуществляется либо с помощью комбинации клавиш “**Ctrl/Shift/·**“ , либо с помощью палитры символьной математики.

$$x + 2\cdot x \rightarrow 3\cdot x \qquad x\cdot \mathrm{acos}(0) \rightarrow \frac{1}{2}\cdot x\cdot \pi \qquad \exp(2\cdot \ln(a)) \rightarrow a^2$$

$$\int_a^b x^3\,dx \rightarrow \frac{1}{4}\cdot b^4 - \frac{1}{4}\cdot a^4 \qquad \int \frac{1}{\sin(\alpha\cdot x)\cdot\cos(\alpha\cdot x)}\,dx \rightarrow \frac{\ln(\tan(\alpha\cdot x))}{\alpha}$$

$$\frac{d}{dx}x^n \rightarrow x^n\cdot\frac{n}{x} \qquad \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \rightarrow \exp(1) \qquad \sum_{n=1}^{\infty}\frac{1}{(2n-1)^2} \rightarrow \frac{1}{8}\cdot\pi^2$$

Рисунок 7.1 - Использование символьного знака равенства

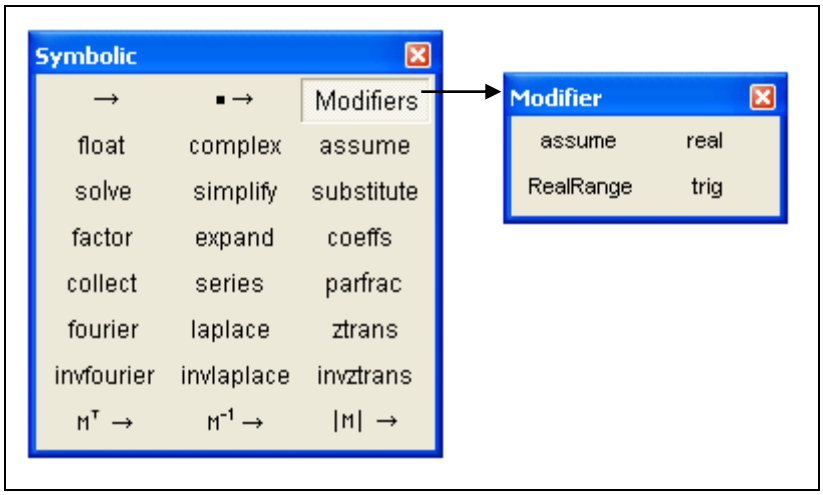


Рисунок 7.2 - Палитра символьной математики MathCad 2000

Краткое описание некоторых ключевых слов:

float — преобразование в формат чисел с плавающей точкой;

complex — преобразования в комплексной форме;

assume — присваивание переменным неопределенного значения;

solve — решение уравнений и систем;

simplify — упрощение выражений;

factor — разложение выражения на простые дроби;

expand — разложение выражения по степеням;

coeffs — возвращает коэффициенты полинома;

collect — приведение подобных членов;

series — разложение в ряд по заданным переменным;

parfac — разложение на элементарные дроби;

fourier — прямое преобразование Фурье;

laplace — прямое преобразование Лапласа;

ztrans — прямое Z-преобразование;

invfourier — обратное преобразование Фурье;

invlaplace — обратное преобразование Лапласа;

invztrans — обратное Z-преобразование;

M^T, M^{-1}, $|M|$ — транспонирование, инвертирование и вычисление определителя матрицы;

Modifiers — модифицированные команды.

На рисунке 7.3 приведены примеры использования этих ключевых слов. Заметьте, что ключевое слово действует только до следующего символа “→”.

$$\frac{x+1}{x^2-1}\cdot\left(\sin(x)^2+\cos(x)^2\right)\ \text{simplify}\ \rightarrow \frac{1}{x-1}$$

$$(x+y)^3\ \text{expand}\ \rightarrow x^3+3\cdot x^2\cdot y+3\cdot x\cdot y^2+y^3$$

$$x^3+3\cdot x^2\cdot y+3\cdot x\cdot y^2+y^3\ \text{factor}\ \rightarrow (x+y)^3$$

$$(1-a\cdot y)\cdot x^2+\left(2\cdot y^2-1\right)\cdot x\ \text{coeffs},x\ \rightarrow \begin{pmatrix}0\\ 2\cdot y^2-1\\ 1-a\cdot y\end{pmatrix}$$

$$x^2-a\cdot y\cdot x^2+2\cdot y^2\cdot x-x\ \text{collect},x\ \rightarrow (1-a\cdot y)\cdot x^2+\left(2\cdot y^2-1\right)\cdot x$$

Разложение на простые дроби

$$\frac{2\cdot x^2-3x+1}{x^3+2\cdot x^2-9x-18}\ \text{convert},\text{parfrac},x\ \rightarrow \frac{1}{3\cdot(x-3)}-\frac{3}{x+2}+\frac{14}{3\cdot(x+3)}$$

Заменить одну переменную на другую

$$\frac{x+3}{x^2}\ \text{substitute},x=(y+1)^2\ \rightarrow \frac{y^2+2\cdot y+4}{(y+1)^4}$$

Преобразование выражений в комплексной форме к виду a+bi

$$e^{i\cdot n\cdot\theta}\ \text{complex}\ \rightarrow \cos(\theta\cdot n)+\sin(\theta\cdot n)\cdot i$$

Вывод результата в символьной форме

$$2\cdot\text{acos}(0)\ \text{float},15\ \rightarrow 3.14159265358979$$

Решение уравнений с помощью символьного знака равенства

$$\sin(x)=-1\ \text{solve},x\ \rightarrow \frac{3\cdot\pi}{2}$$

$$\frac{1}{2}\cdot x^2+x+2\ \text{solve},x\ \rightarrow \begin{pmatrix}-1-\sqrt{3}\cdot i\\ -1+\sqrt{3}\cdot i\end{pmatrix}$$

$$\left(x^3-5\cdot x^2-4\cdot x\right)+20>0\ \text{solve},x\ \rightarrow 5<x\vee -2<x<2$$

Рисунок 7.3 - Использование ключевых слов с символьным знаком равенства

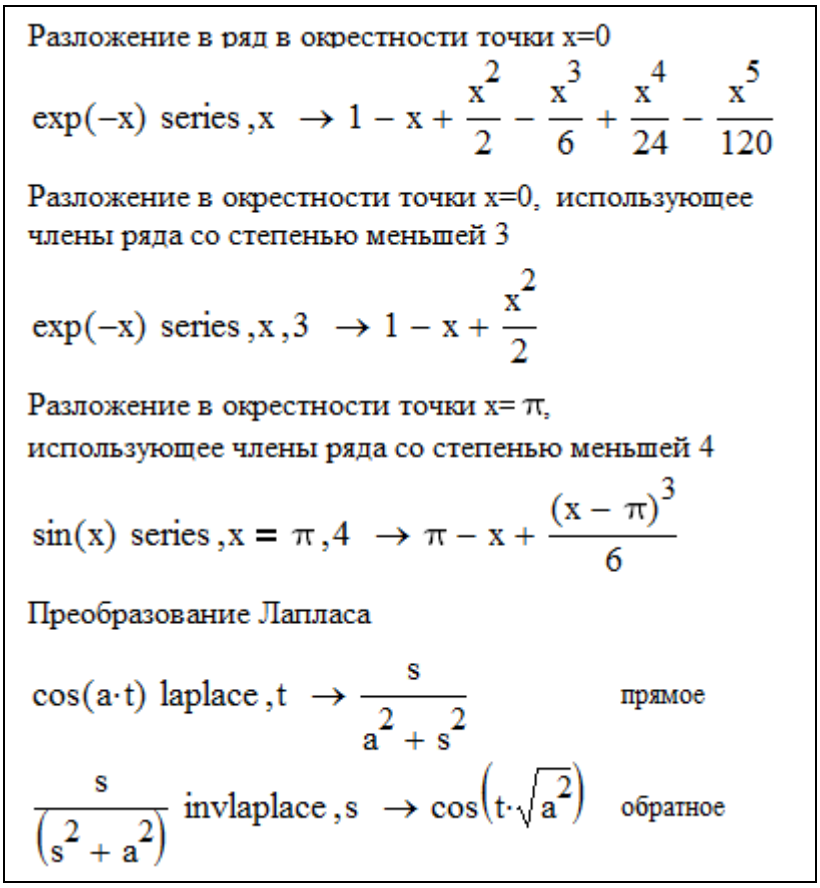


Рисунок 7.3 - Использование ключевых слов с символьным знаком равенства

7.2 Преобразования с использование меню Symbolic

Команды из меню **Symbolic** (Символы) обеспечивают несколько более широкое управление видами преобразований, чем использование символьного знака равенства.

Основные шаги для использования меню **Symbolic** (Символы) - те же самые, что и для всех команд меню:

- заключите все, что требуется преобразовать в выделяющую рамку;
- выберите команду из меню **Symbolic** (Символы);
- MathCad поместит преобразованное выражение в рабочий документ.

Таким способом преобразовать можно не только целиком выражение, как в предыдущем случае, но и отдельные его части, заключив их в выделяющую рамку.

Ниже в этом разделе будут рассмотрены основные команды меню **Символы**.

Evaluate/Symbolically (Расчеты/Символические) - вычислить символьно.

Данная команда по своему действию эквивалентна символьному знаку равенства, использованному без ключевых слов.

Simplify (Упростить) - упростить выражение.

Результат этой команды - то же самое действие, которое производит символьный знак равенства совместно с ключевым словом Simplify, т.е. выполнение арифметических преобразований, сокращение общих множителей, использование основных тригонометрических тождеств, упрощение квадратных корней и степеней.

На рисунке 7.4 показаны примеры использования этого пункта меню.

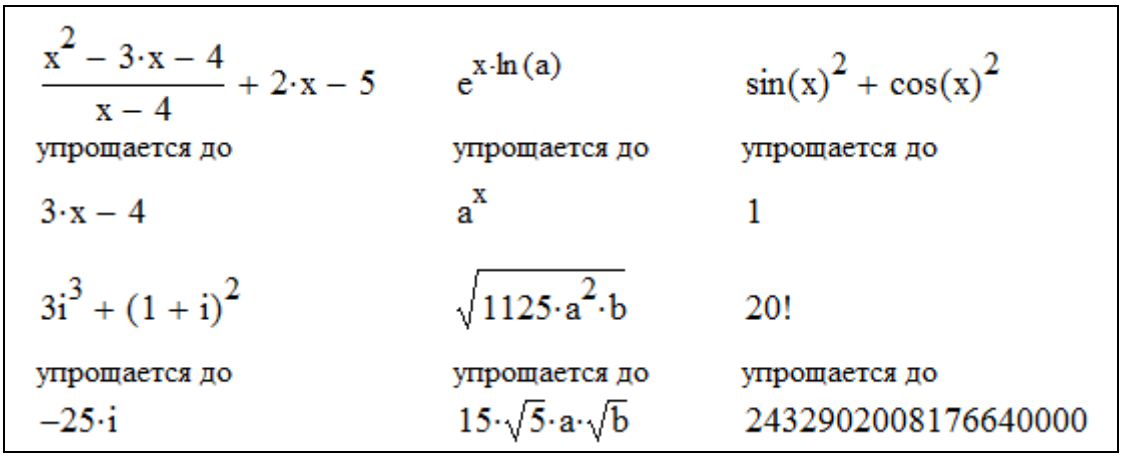


Рисунок 7.4 - Примеры использования пункта меню Simplify.

Expand (Расширить) - разложить выражение по степеням.

Данная команда аналогична по действию символьному знаку равенства, использованному совместно с ключевым словом Expand и разлагает все степени и произведения сумм в выделенном выражении. Если выражение - дробь, числитель будет разложен и выражение будет представлено как сумма дробей. Синусы, косинусы и тангенсы сумм и произведений будут разложены по мере возможности в выражения, включающие только синусы и косинусы одиночных переменных.

Так, например, *cos(5 x)* после разложения по степеням дает

$$16\cos(x)^5 - 20\cos(x)^3 + 5\cos(x).$$

Factor (Фактор) **-** разложить выражение на множители.

Если выражение представляет целое число, MathCad разложит его на множители по степеням простых чисел, в остальных случаях будет сделана попытка преобразовать выражение в произведение. Так, в результате применения этого действия к числу 3546738466 будет получен ответ $2^2 \cdot 3^2 \cdot 7 \cdot 13 \cdot 509 \cdot 709$, разложение на множители выражения

$$(-5\cdot x\cdot z\cdot y + 2\cdot x\cdot z^2 - x^2\cdot y - 2\cdot x^2\cdot z + 3\cdot y^2\cdot z + 6\cdot y\cdot z^2 - 3\cdot x\cdot y)$$

дает

$$(x + 3\cdot y)\cdot(2\cdot z + y)\cdot(z - x).$$

При выборе этой команды нужно стараться выделить такую часть выражения, разложение на множители которой дает заметный результат. Так, если выделить все выражение $a \cdot b + a \cdot c + x$, то MathCad вернет это выражение неизмененным, но если выбрать только первые два члена, то будет возвращено $a \cdot (b + c) + x$.

Collect (Подобные) **-** приведение подобных членов.

Данная команда объединяет члены, содержащие одинаковые степени выделенного подвыражения. Выбираемое подвыражение должно быть простой переменной либо встроенной функцией вместе с аргументом. Результатом является полином от подвыражения.

Так, если выделить переменную *x* в любом месте выражения

$$x^2 - a\cdot y\cdot x^2 + 2\cdot y^2\cdot x - x$$

и выбрать команду **Collect**, то в результате получим

$$(1 - a\cdot y)\cdot x^2 + (2\cdot y^2 - 1)\cdot x.$$

Polinomial Coefficients (Коэффициенты полинома) **-** коэффициенты полинома.

Многие выражения могут быть представлены в виде полиномов от выделенной переменной. Данная команда позволяет выделить коэффициенты этого полинома. Так, например, выделив переменную x в выражении

$$3 \cdot b \cdot x^3 - \pi \cdot x + 2/3$$

и выполнив данную команду, получим вектор

$$\begin{bmatrix} \frac{2}{3} \\ -\pi \\ 0 \\ 3 \cdot b \end{bmatrix}$$

Следующая группа команд, объединенная общим заголовком **Variable** (Переменные) требует указания (выделения) переменной, по которой выполняется соответствующее действие.

Variable/Solve (Переменные/Вычислить) **–** решить относительно переменной.

Чтобы решить уравнение относительно переменной, нужно:

- напечатать уравнение или выражение. Если MathCad не находит знака равенства (введено выражение), то предполагается, что выражение требуется приравнять нулю. Не забудьте, что знак равенства вводится комбинацией "**Ctrl/=**";
- выделить переменную, относительно которой нужно решить уравнение;
- выбрать команду **Variable/Solve**.

MathCad решит уравнение относительно выделенной переменной и вставит результат в рабочий документ. Если решений получается несколько, то они представляются в виде вектора.

Можно также решить неравенство, использующее знаки >, ≥, <, ≤. Решения для неравенств отображаются в виде булевских выражений. Несколько решений также помещаются в вектор.

На рисунке 7.5 показаны примеры решения уравнений и неравенств.

$e^x = -10$ имеет решение $\ln(-10)$

$\frac{1}{2} \cdot x^2 + x + 2$ имеет решение $\begin{pmatrix} -1 + i \cdot \sqrt{3} \\ -1 - i \cdot \sqrt{3} \end{pmatrix}$

$\sin(x) - \frac{1}{3} \cdot \tan(x)$ имеет решение $\begin{pmatrix} 0 \\ \operatorname{atan}(2 \cdot \sqrt{2}) \\ -\operatorname{atan}(2 \cdot \sqrt{2}) \end{pmatrix}$

$\left(x^3 - 5 \cdot x^2 - 4 \cdot x\right) + 20 > 0$ имеет решение $\begin{bmatrix} (-2 < x) \cdot (x < 2) \\ 5 < x \end{bmatrix}$

Рисунок 7.5 - Примеры решения уравнений и неравенств

Variable/Substitute (Переменные/Замена) – замена переменных.

Эта команда заменяет выделенным выражением заданную переменную.

Чтобы использовать эту команду:

- выделите выражение, которое будет заменять переменную;
- скопируйте его в буфер обмена;
- выделите переменную, которую нужно заменить, и выполните рассматриваемую команду.

Так, например, если в буфер обмена помещено выражение $x + 3 \cdot a$, а в выражении $z^2 + 2/z$ выделена переменная z, то в результате действия команды получится: $(x + 3 \cdot a)^2 + \frac{2}{(x + 3 \cdot a)}$.

Variable/ Differentiate (Переменные/Дифференциалы) – дифференцировать по переменной.

Данная команда позволяет найти производную по выделенной переменной. Например, чтобы продифференцировать $2 \cdot x^2$ по x, нужно выделить x и выбрать из меню рассматриваемую команду, после чего MathCad отобразит результат $4 \cdot x$.

Если выделить переменную y вместо x, то результат будет 1, MathCad рассматривает все переменные, кроме выделенной, как константы.

Если не выделить ни x, ни y, команда меню будет недоступна для выбора.

Variable/Integrate (Переменные/Интеграция) – интегрировать по переменной.

Ранее был рассмотрен способ аналитического вычисления неопределенных и определенных интегралов с использование знака интеграла и символьного знака равенства, как это было показано на рисунке 7.1;

Пункт меню Variable/Integrate позволяет использовать еще один способ нахождения неопределенного интеграла, при этом знак интеграла не используется, а необходимо только записать подинтегральное выражение, выделить в нем переменную, по которой нужно выполнить интегрирование, и выполнить рассматриваемую команду.

На рисунке 7.6 показан пример вычисления интегралов различными способами.

Если MathCad не может найти неопределенный интеграл, он возвращает выражение неизмененным.

При вычислении неопределенного интеграла следует помнить, что если *f(x)* - интеграл данной функции, то для любой константы С интегралом будет также *f(x)* + С.

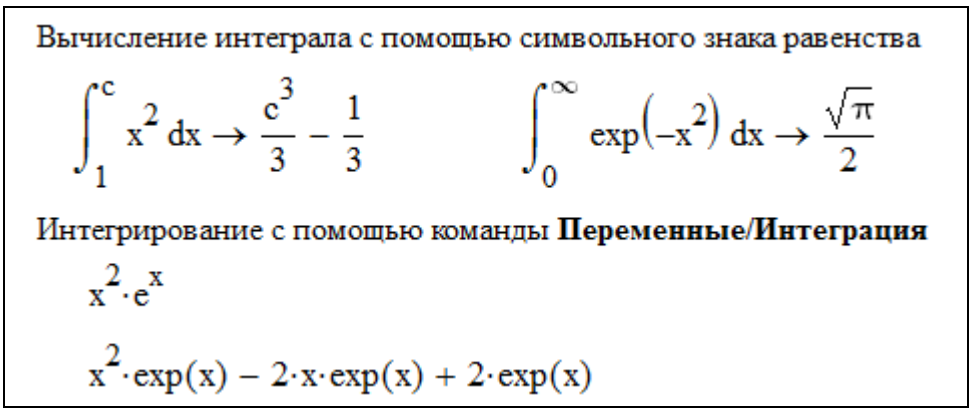


Рисунок 7.6 - Символьное вычисление интегралов

Variable/Expand to Series (Переменные/Разложить на составляющие) - разложить в ряд

По назначению эта команда аналогична действию символьного знака равенства совместно с ключевым словом Expand, но имеются следующие отличия:

- для задания степени разложения используется диалоговое окно, которое появляется при выборе команды;
- команда применяется для разложения в ряд только по одной переменной;
- разложение в ряд производится только в окрестности точки 0.

Convert to Partial Fraction (Переменные/Преобразовать в частичные доли) - разложить на элементарные дроби

Чтобы преобразовать выражение в сумму элементарных дробей, нужно выделить переменную в знаменателе выражения и выбрать команду из меню. Все константы в выражении должны быть целыми числами или дробями, но не константами с десятичной точкой.

Например, выделив переменную *x* в знаменателе выражения:

$$\frac{2\cdot x^2 - 3\cdot x + 1}{x^3 + 2\cdot x^2 - 9\cdot x - 18}$$

и выполнив команду , получим ответ

$$\frac{1}{3\cdot(x-3)} + \frac{14}{3\cdot(x+3)} - \frac{3}{x+2} \ .$$

Matrix (Матрицы) - матричные операции

В эту группу команд входят три подкоманды, позволяющие выполнять символьным образом транспонирование, обращение и вычисление детерминанта матрицы.

Перед использованием этих команд необходимо заключить в выделяющую рамку всю матрицу целиком.

Примеры выполнения матричных операций приведен на рисунке 7.7

Транспонирование матрицы с помощью команды **Matrix/Transpose**

$$\begin{pmatrix} A & B \\ C & D \\ E & F \end{pmatrix} \qquad \begin{pmatrix} A & C & E \\ B & D & F \end{pmatrix}$$

Обращение матрицы с помощью команды **Matrix/Invert**

$$\begin{pmatrix} A-1 & B-2 \\ C-3 & D-4 \end{pmatrix}$$

$$\frac{1}{(A\cdot D - 4\cdot A - D - 2 - B\cdot C + 3\cdot B + 2\cdot C)} \cdot \begin{pmatrix} D-4 & -B+2 \\ -C+3 & A-1 \end{pmatrix}$$

Вычисление определителя с помощью команды **Matrix/Determinant**

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad A\cdot D - B\cdot C$$

Рисунок 7.7 - Примеры выполнения матричных операций

8 ПРОГРАММИРОВАНИЕ

MathCad позволяет писать программы на своем собственном встроенном языке. Хотя язык программирования MathCad весьма значительно отличается от таких языков, как Бейсик, Паскаль и т.п., он содержит конструкции, во многом подобные конструкциям других языков программирования: условные операторы, операторы цикла, подпрограммы.

Написание программ в MathCad позволяет решить такие задачи, которые трудно решить другим способом.

8.1 Создание программ

Программа есть частный случай выражения MathCad. Главным отличием программы от выражения является способ задания вычислений. При использовании выражения алгоритм нахождения ответа задается одним оператором, в программе их может быть любое количество.

Алгоритмические конструкции в MathCad вводятся не традиционным набором на клавиатуре ключевых слов if, then, else, while и др., а нажатием одной из кнопок панели программирования, показанной на рисунке 8.1.

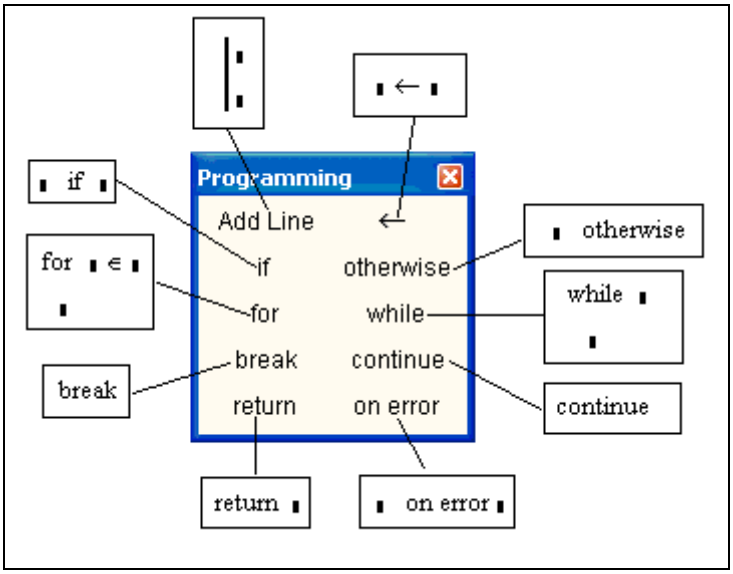


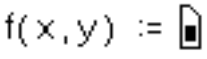
Рисунок 8.1 - Панель программирования MathCad

Щелчок по одной из этих кнопок создает на экране заготовку соответствующей программной конструкции.

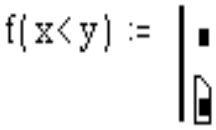
Следующий пример показывает, как написать программу вычисления функции $f(x,y)=log\frac{x}{y}$.

Конечно, эту задачу можно решить и без использования программы, но на этом примере будет показано, как создаются программы в MathCad.

- введите левую часть определения функции и знак присваивания;



- откройте панель программирования и нажмите на ней кнопку “**Add Line**”. Появится вертикальный столбец с двумя полями ввода для занесения операторов;



- перейдите в верхнее поле, щелкнув по нему мышью или нажав клавишу “**Tab**”. Напечатайте Z и нажмите кнопку “←“ на панели программирования;

$f(x,y) := \left| Z \leftarrow \blacksquare \right.$

- в поле ввода справа от “←“ введите *x/y* ;

$f(x,y) := \left| Z \leftarrow \frac{x}{y} \right.$

- нижнее поле ввода предназначено для задания возвращаемого функцией значения log(Z);

$f(x,y) := \left| Z \leftarrow \frac{x}{y} \right. \log(Z)$

Символ “←“ в MathCad соответствует оператору присваивания, а выражение **Z ← x/y** означает, что переменной Z нужно присвоить значение, равное *x/y*.

Теперь рассмотренную программу можно использовать точно так же, как и любую другую функцию. На рисунке 8.2 показано два примера записи программы MathCad. Обратите внимание, что переменная Z является локальной и определена только внутри программы.

Программа может содержать любое числа операторов (второй пример рисунке 8.2). Для добавления оператора служит кнопка “**Add Line**” на панели программирования. Чтобы удалить позицию ввода, ее нужно выделить и нажать клавишу “**Del**”.

$f(x,y) := \left| z \leftarrow \frac{x}{y} \right. \log(z)$

$f(100,10) = 1$

z= Переменная не определена вне программы

Вычисление выражения $r(a,b,c) = \frac{-b + \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$

$r(a,b,c) := \left| D \leftarrow b^2 - 4 \cdot a \cdot c \right. t \leftarrow -b + \sqrt{D} \; z \leftarrow 2 \cdot a \; \frac{t}{z}$

$r(1,2,1) = -1$

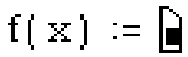
Рисунок 8.2 - Примеры простейших программ MathCad

Возвращаемым программой значением является значение последнего выражения программы. Возвращаться может число, как на рисунке 8.2, или массив чисел, как это будет показано ниже.

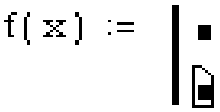
8.2 Условный оператор

Ниже приводится пример определения функции, задаваемой различными аналитическими выражениями на разных участках области определения:

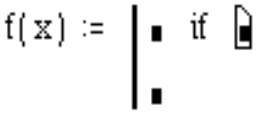
- введите левую часть определения функции и знак присваивания;



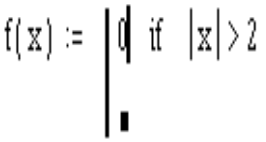
- нажмите кнопку “**Add Line**” на панели программирования;



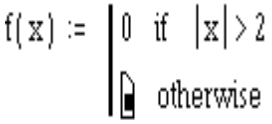
- перейдите в верхнее поле ввода и щелкните по кнопке “ **if** “ на панели программирования;



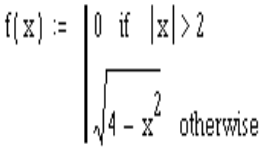
- сейчас активировано поле для ввода логического выражения. Левое поле предназначено для значения, которое будет иметь выражение, если логическое выражение справа истинно;



- перейдите в нижнее поле ввода и нажмите кнопку “**otherwise**” на панели программирования;



- введите значение, которое программа должна возвратить, если логическое выражение ложно.



На рисунке 8.3 показан график этой функции, на рисунке 8.4 - более сложная программа, использующая условный оператор, в сравнении с использованием встроенной функции if.

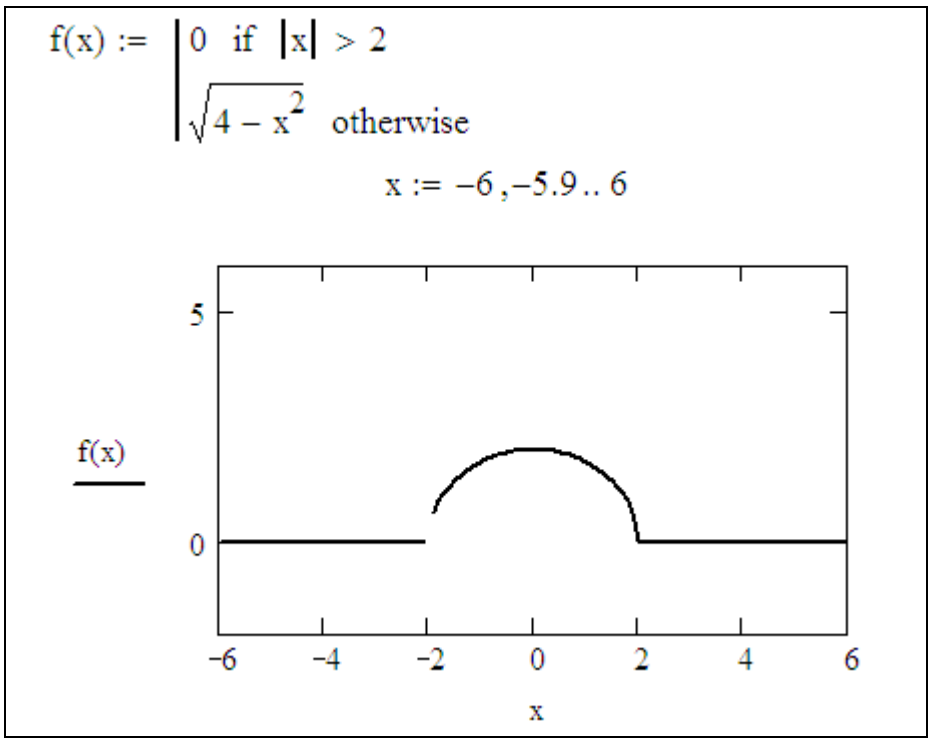
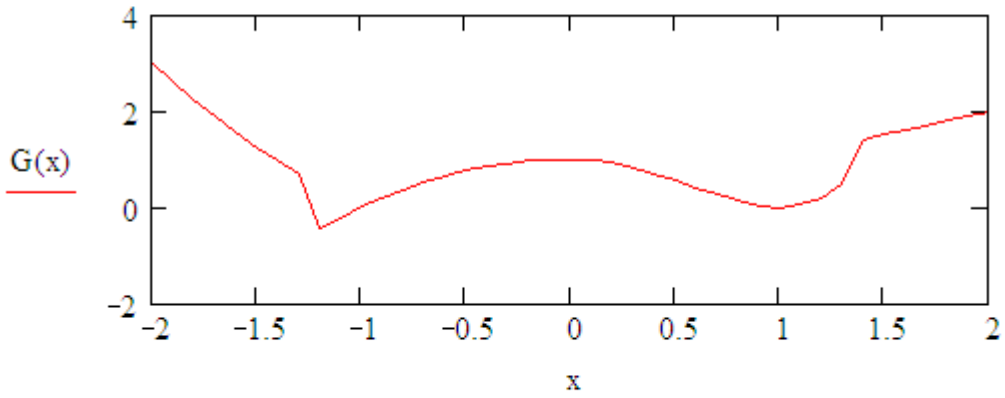


Рисунок 8.3 - Пример программы с использованием условного оператора

$x := -2, -1.9 .. 2$ $\quad F(x) := x^2 - 1$ $\quad G(x) := \begin{cases} F(x) & \text{if } -2 \le x \le -1.3 \\ -F(x) & \text{if } -1.3 < x < 0 \\ F(x)^2 & \text{if } 0 \le x < 1.4 \\ x & \text{otherwise} \end{cases}$



Тоже с использованием функции if

$g(x) := \text{if}\left(-2 \le x \le 1.3, F(x), \text{if}\left(-1.3 < x < 0, -F(x), \text{if}\left(0 \le x < 1.4, F(x)^2, x\right)\right)\right)$

Рисунок 8.4 - Сравнение программного оператора "if" и встроенной функции "if"

8.3 Оператор цикла "while"

Структура оператора while в MathCad аналогична структуре такого же оператора Паскаля. Методика ввода операторов достаточно подробно рассмотрена в предыдущих разделах. Если программа зациклилась, то ее можно остановить нажатием клавиши "**Esc**".

На рисунке 8.5 показан пример использования оператора "while".

Поиск первого элемента вектора, большего заданного значения

$i := 0 .. 100$ $\quad v_i := 1 + \sin(i)$ Создание вектора

$y(v, t) := \begin{cases} k \leftarrow 0 \\ \text{while } v_k \le t \\ \quad k \leftarrow k + 1 \\ k \end{cases}$

$y(v, 1.98) = 8$ Величина 1.98 впервые превосходит восьмой элемент вектора

$v =$

	0
0	1
1	1.841
2	1.909
3	1.141
4	0.243
5	0.041
6	0.721
7	1.657
8	1.989
9	1.412
10	0.456

Рисунок 8.5 - Пример использования оператора "while"

8.4 Операторы “break”, “return”

Часто бывает необходимо выйти из цикла при выполнении некоторого условия. Для этого, можно использовать оператор “break”, который, так же как соответствующий оператор в Паскале, обеспечивает немедленный выход из любого оператора цикла.

Иногда требуется немедленный выход не только из цикла, но и из всей программы. Для этого служит оператор “return”, который прерывает выполнение программы-функции и возвращает значение операнда, стоящего за ним. Следующий пример поясняет работу этого оператора. Например, программа на рисунке 8.5 будет зациклена, если каждый элемент вектора **v** меньше, чем t. В этом случае, записанное после “while” условие никогда не станет ложным и поиск выйдет за границы вектора, что приведет к сообщению о выходе индекса за допустимые пределы.

Программа на рисунке 8.6 возвратит ноль в том случае, если все элементы вектора **v** меньше заданного t. В противном случае она возвращает индекс и значение первого элемента, превосходящего t.

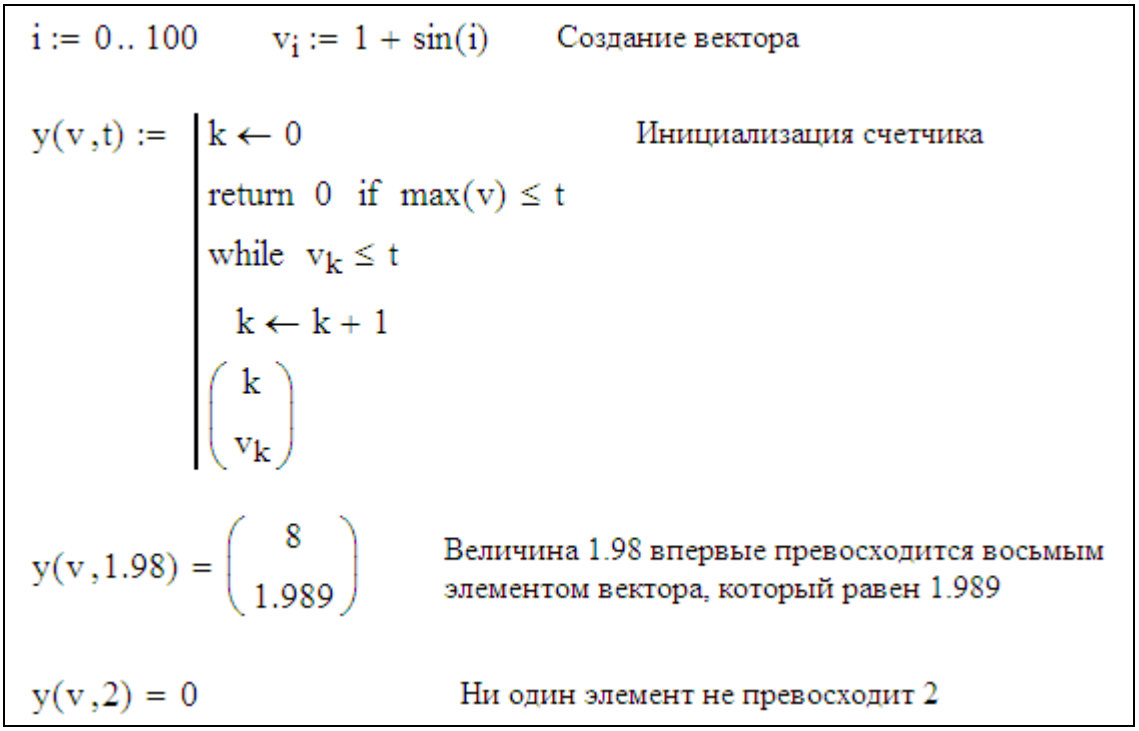
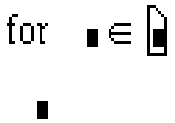


Рисунок 8.6 - Пример использования оператора “return”

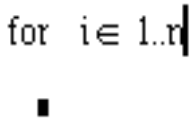
8.5 Оператор цикла “for”

Оператор цикла “for”, как и в Паскале, задает заранее определенное число повторений. Для создания цикла for:

- щелкните по кнопке “**for**” на панели программирования;



- в поле слева от знака ∈ введите имя переменной цикла, а в поле справа - диапазон значений, в котором должна изменяться переменная цикла. Диапазон может вводиться точно так же, как и дискретный аргумент;
- в поле внизу введите оператор или группу операторов, представляющих тело цикла.



В верхней части рисунка 8.7 показан цикл “for”, используемый для вычисления факториала числа N.

В нижней части рисунка приведен пример, в котором переменная цикла определена не с помощью диапазона, а через элементы вектора. Необходимо отметить, что хотя выражение, справа от знака ∈ обычно является диапазоном, оно может быть также вектором или списком скаляров, диапазонов и векторов, разделенных запятыми.

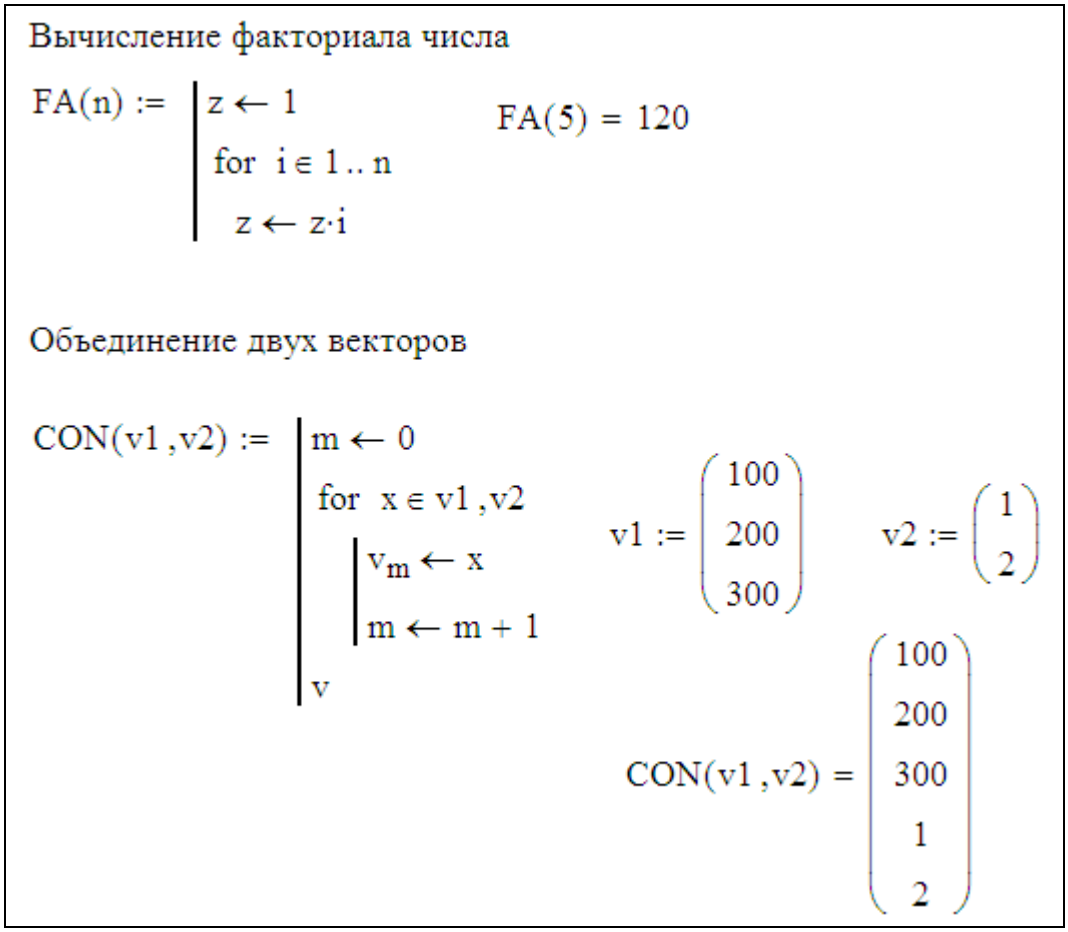


Рисунок 8.7 - Примеры использования оператора “for”

8.6 Примеры программ

Одной из характеристик, определяющих гибкость методов программирования, является возможность использовать одни программные структуры внутри других. В MathCad это можно сделать следующими путями:

- один из операторов можно сделать, в свою очередь, программой;
- можно определить функцию рекурсивным методом.

В данном разделе рассмотрены три примера более сложных программ, которые трудно было бы решить без возможностей программирования.

На рисунке 8.8 показан пример программы, реализующей метод Ньютона решения нелинейного уравнения, операторы которой сами являются программой, на рисунке 8.9 - программы с вложенными циклами.

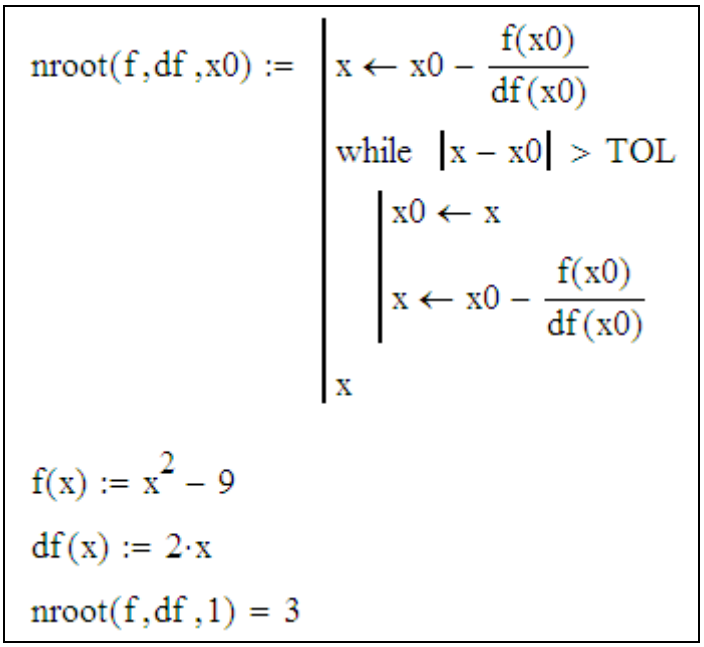


Рисунок 8.8 - Программа решения нелинейного уравнения методом Ньютона

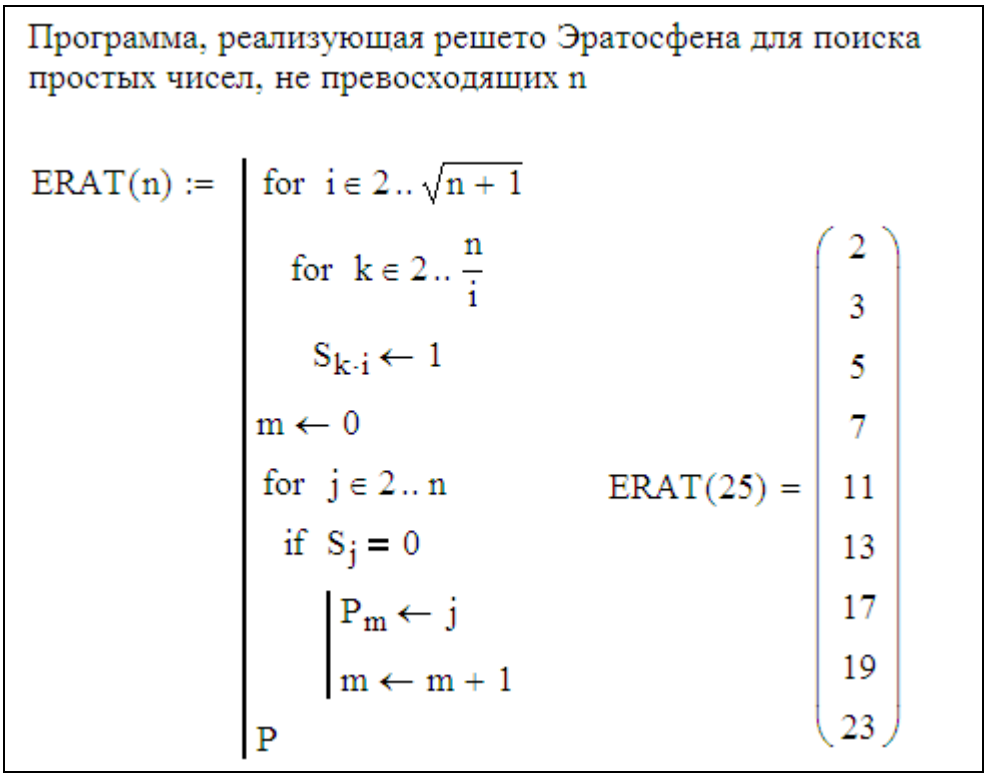


Рисунок 8.9 - Программа нахождения простых чисел

Рекурсия является одним из мощных методов программирования и заключается в определении функции через саму себя, как это показано на рисунке 8.10.

Рекурсивные определения функции всегда состоят по меньшей мере из двух частей:

- начального определения, предотвращающего бесконечную рекурсию;
- определения функции в терминах предыдущего значения функции. Основная идея подобна идее математической функции: если можно получить значение $f(n+1)$ из $f(n)$ и известно $f(0)$, то известна и вся функция f.

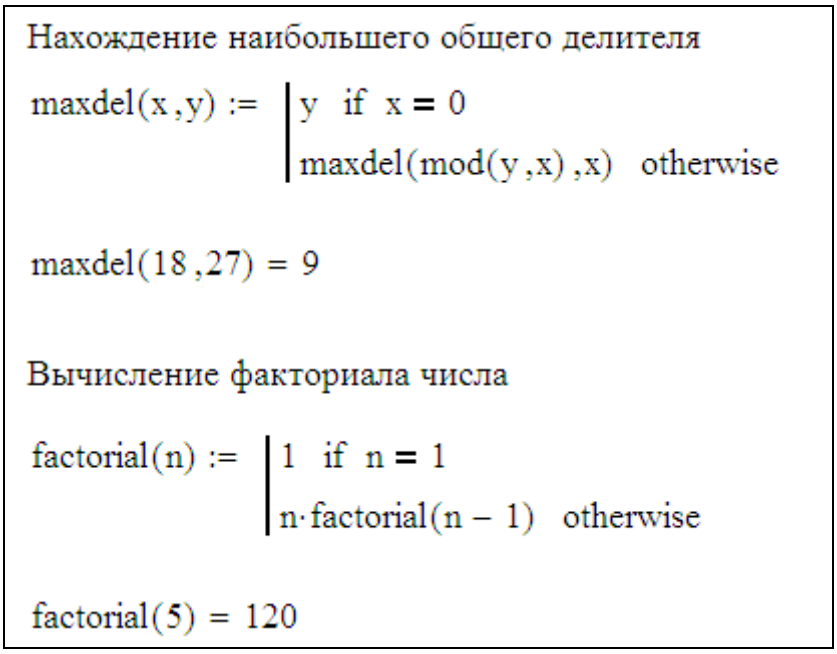


Рисунок 8.10 - Примеры рекурсивных определений функции

9 ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ НА ЛАБОРАТОРНЫХ РАБОТАХ

9.1 Решение уравнений и систем

В задачах 9.1.1. - 9.1.5. все решения должны быть проиллюстрированы соответствующими графиками.

9.1.1 Найти наименьший положительный корень уравнения:

$$tg(0.9464 \cdot x) - 1.3825 \cdot x = 0$$

9.1.2 Найти больший корень уравнения:

$$ln(0.6098 \cdot x) - 0.6872 \cdot x + 1.5 = 0$$

9.1.3 Найти все корни уравнения:

$$x^3 + e^{-x} - 3 \cdot e^{-(3x-3)^2} = 1$$

9.1.4 Найти все корни уравнения:

$$x^4 - 10 \cdot x^3 + 35 \cdot x^2 - 50 \cdot x + 24 = 0$$

9.1.5 Найти корень уравнения с точностью 10^{-6}:

$$x + lg(x) = 0.5$$

9.1.6 Найти все корни системы нелинейных уравнений:

$$\begin{cases} x^2 + y^2 = 2 \\ y = sin(x) \end{cases}$$

Для построения окружности воспользоваться ее параметрическим представлением $x = R \cdot cos(t); \quad y = R \cdot sin(t); \quad 0 \le t \le 2\pi$.

9.1.7 Решить систему нелинейных уравнений:

$$\begin{cases} 2 \cdot x + y = 5 - 2 \cdot z^2 \\ y^3 + 4 \cdot z = 4 \\ x \cdot y + z = e^z \end{cases}$$

9.2 Нахождение экстремумов функций

В задачах 9.2.1. - 9.2.7. все решения должны быть показаны на соответствующих графиках.

9.2.1 Найти минимум функции *f(x)* на интервале [*- 4; 0*]:

$$f(x) = x \cdot exp(-x^2 / 2)$$

9.2.2 Найти минимум функции *f(x)* на интервале [*0; 4*]:

$$f(x) = sin(x + x \cdot sin(x))$$

9.2.3 Найти максимум функции *f(x)* на интервале [*-1; 1*]:

$$f(x) = x \cdot \sqrt{1 - x^2}$$

9.2.4 Найти максимум функции *f(x)* на интервале [*-2; 4*]:

$$f(x) = sin(x + x \cdot sin(x))$$

9.2.5 Найти максимум и минимум функции

$$f(x)=e^{-x^2}\int_0^x e^{t^2}\cdot dt$$

9.2.6 Найти минимум функции *f(x, y)* в области $0.4 \le x \le 2.6,\ -0.2 \le y \le 1$

$$f(x,y)=0.5\cdot x^3+4\cdot y^2-4\cdot x-4\cdot y+2$$

9.2.7 Найти максимум функции *f(x, y)* в области $-0.2 \le x \le 0.4,\ 0 \le y \le 0.5$

$$f(x,y)=x(2\cdot x-1)^2+4\cdot x\cdot y(2\cdot x\cdot y-x)^2$$

9.3 Интерполяция данных

9.3.1 Исходные данные имеют вид:

X	0.5	1.2	2.6	4.3	4.9	6.8	8.7
Y	4.1	2.4	3	4.3	3.6	5.2	6

С помощью линейной интерполяции вычислить значение *y*, если $x = 3.45$. Построить график, отображающий линейную интерполяцию для $-1 \le x \le 10$, отложив по оси ординат величину *interp(X, Y, x)*. Показать на графике исходные данные.

9.3.2 С помощью сплайн-интерполяции вычислить значение y, если $x = 3.45$, для данных из задачи 9.3.1.

Построить график, отображающий сплайн-интерполяцию для $-1 \le x \le 10$. Показать на графике исходные данные.

9.3.3 С помощью полиномиальной интерполяции вычислить значение y, если $x = 3.45$, для данных из задачи 9.3.1.

Построить график, отображающий сплайн-интерполяцию для $0.5 \le x \le 9$. Показать на графике исходные данные.

9.3.4 Функция *f (x,y)* представлены в виде таблицы:

y / *x*	1.2	1.4	1.6	1.8	2.0	2.2
0.3	0.738	1.574	2.746	4.302	6.29	8.758
0.6	- 0.072	0.584	1.576	2.952	4.76	7.048
0.9	- 0.702	- 0.226	0.586	1.782	3.41	5.518
1.2	- 1.152	- 0.856	- 0.224	0.792	2.24	4.168
1.5	- 1.422	- 1.306	- 0.854	- 0.018	1.25	2.998
1.8	- 1.512	- 1.576	- 1.304	- 0.648	0.44	2.008

Интерполировать кубическим сплайном функцию *f (x,y)* в точке (*1, 1.5*). Построить интерполяционную поверхность.

9.3.5 Вычислить значение функции $f(x)=2\cdot exp\left(\frac{x}{2}-0.3\cdot x^2\right)$ на интервале [*-2; 1*] и осуществить линейное предсказание ее поведения на интервале [*1; 2*]. Результат показать на графике.

9.4 Аппроксимация данных

В задачах 9.4.1 - 9.4.7 нужно аппроксимировать данные указанными зависимостями (вычислить их коэффициенты) и построить графики, на которых следует отобразить как сами данные, так и аппроксимирующие кривые.

9.4.1 $y = A \cdot x + B$

X	1	3	6	7	9
Y	2.8	7.4	12.7	15.5	18.5

9.4.2 $y = A_0 + A_1 \cdot x + A_2 \cdot x^2 + A_3 \cdot x^3$

X	0	1	2	3
Y	1	10	49	142

В задачах 9.4.3 - 9.4.5 указанные зависимости необходимо сначала путем преобразований и замены переменных свести к линейному виду.

9.4.3 $y = A \cdot e^{\frac{B}{x}}$

X	3	4	5
Y	67.54	39.62	28.77

9.4.4 $y = \dfrac{x}{A \cdot x + B}$

X	1	2	3
Y	0.1333	0.1600	0.1714

9.4.5 $y = exp\left(A \cdot x^2 + \dfrac{B}{100} \right)$

X	1	3	5	7
Y	1.05	1.32	2.06	4.04

9.4.6 $y = A \cdot x + B \cdot e^x + C \cdot \dfrac{1}{x}$

X	1	2	3	4	5
Y	1.575	9.211	22.268	52.529	130.131

9.4.7 $y = A \cdot sin(x + B) + C$

X	2	2.6	3.8	4.2	5.5
Y	4.2	5.4	10.3	12.2	13.9

9.4.8 Решить задачу 9.4.3, минимизируя сумму квадратов отклонений аппроксимирующей функции $A \cdot exp\ (B/x)$ от исходных данных с помощью встроенной функции *minerr*.

9.5 Решение дифференциальных уравнений

9.5.1 Решить дифференциальное уравнение на отрезке [*0; 1*] .

$$y' = 2 \cdot y + e^x - x \qquad y(0) = 0.25$$

Построить график решения, на котором дополнительно точками отобразить функцию $f(x) = e^{2x} - e^x + 0.5 \cdot x + 0.3$.

9.5.2 Решить систему дифференциальных уравнений на отрезке [*1; 2*] .

$$\begin{cases} y_1' = -y_2 & y_1(1) = 1 \\ y_2' = \dfrac{y_2^2}{y_1} & y_2(1) = -0.5 \end{cases}$$

Построить графики решения $y_1(x)$, $y_2(x)$, на этом же рисунке дополнительно отобразить точками график функции: $f_1(x) = \sqrt{x}$, $f_2(x) = -\dfrac{1}{2\sqrt{x}}$.

9.5.3 Решить дифференциальное уравнение на отрезке [*1; 2*] .

$$y''' = \frac{ln(x)}{x^2}$$

$$y(1) = 0; \qquad y'(1) = 1; \qquad y''(1) = 2$$

Показать решение на графике, Отобразить точками на этом же графике функцию $f(x) = -\frac{x}{2} \cdot ln^2 x + \frac{3}{2}x^2 - 2 \cdot x + \frac{1}{2}$.

9.5.4 Решить систему дифференциальных уравнений:

$$\begin{cases} x''(t) = \dfrac{-a^2 \cdot x(t)}{R(t)} & x(0) = 1 - l \qquad x'(0) = 0 \\ y''(t) = \dfrac{-a^2 \cdot y(t)}{R(t)} & y(0) = 0 \qquad y'(0) = a \cdot \sqrt{\dfrac{1+l}{1-l}} \end{cases}$$

где $R(t) = \left[x^2(t) + y^2(t)\right]^{3/2}$;

$l = 0.25$;

$a = \pi/4$;

$0 \le t \le 12$

Построить зависимости *x(t); y (t); y (x)*.

9.5.5 Найти решение дифференциального уравнения только в одной точке *x =1.2*

$$y'' = x \cdot exp(x)$$

$$y(0) = 1; \quad y'(0) = 0 .$$

9.5.6 Найти решение дифференциального уравнения, удовлетворяющее заданным краевым условиям:

$$y'' - y = 0; \qquad y(0) = 0; \qquad y(2 \cdot \pi) = 1.$$

Показать решение графически. На этом же графике показать точками функцию

$$y = \frac{sh(x)}{sh(2 \cdot \pi)} .$$

9.5.7 Найти решение дифференциального уравнения, удовлетворяющее заданным краевым условиям, в одной единственной точке x = 0.6

$$y'' + y = 0; \qquad y'(0) = 0; \qquad y'(1) = 1.$$

9.6 Статистические задачи

9.6.1 В файле D:\WORK\STAT.PRN записан массив чисел.

а) считать этот массив в MathCad и определить:

- число элементов в массиве;
- минимальный Xmin и максимальный Xmax элементы массива;
- среднее, медиану, дисперсию и среднеквадратичное отклонение чисел массива;

б) показать элементы массива на графике в виде точек;

в) разбить диапазон (Xmax - Xmin) на 10 равных интервалов и построить гистограмму распределения частот попаданий считанных величин в соответствующие интервалы;

г) в предположении, что считанные величины подчиняются нормальному закону распределения, построить на том же графике теоретическую кривую распределения с параметрами, найденными в пункте а).

9.6.2 Вычислить вероятность того, что случайная величина, имеющая стандартное нормальное распределение ($\mu = 0;\ \sigma = 1$), превосходит 1.0 .

9.6.3 Определить плотность распределения вероятности в точке 5.5 случайной величины, имеющей распределение ХИ-квадрат с числом степеней свободы 11.

9.6.4 Создать вектор 1000 случайных величин, имеющих равномерное распределение на отрезке [0; 2] . Построить гистограмму распределения полученных значений, состоящую из 20 столбцов. На этом же графике показать линию, соответствующую теоретическому распределению.

9.7 Символьные вычисления

9.7.1 При помощи символьного знака равенства аналитически вычислить:

а) $\int_a^b sin(x)dx$ б) $\left(sin(x)\cdot cos(x)\right)'$

в) $\left(\frac{sin(x)}{cos(x)}\right)''$ г) $\sum_{k=0}^{3} \frac{3!}{k!\cdot(3-k)!}\cdot x^k \cdot 2^{3-k}$

9.7.2 При помощи символьного знака равенства и соответствующего ключевого слова упростить:

а) $\frac{x^{3/2} + x - 2\cdot\sqrt{x} - 2}{x - 2}$ б) $\frac{a^6 + 64}{a^4 - 4\cdot a^2 + 16}$

в) $\left(\sqrt[3]{7}\right)^{\frac{3}{lg\,7}}$ г) $\frac{(sin\alpha + cos\alpha)^2 - sin\,2\alpha}{cos\,2\alpha + 2\,sin^2\,\alpha}$

9.7.3 При помощи символьного знака равенства и ключевого слова *expand* раскрыть скобки в выражении, а затем, с помощью ключевого слова *factor* вернуться к исходному выражению

а) $(x-1)^2(a^2 - x^2)$ б) $\frac{(x-1)^4}{(a+1)^2}$

9.7.4 Разложить функцию *f(x)* в ряд Тейлора в окрестности точки *x0*, используя члены со степенью, меньше *n*:

а) *f(x) = cos (x)* $x_0 = 0;$ $n = 6$

б) *f(x) = ln (1 + x)* $x_0 = 0;$ $n = 4$

в) *f(x) = sh(x)* $x_0 = 1;$ $n = 6$

г) *f(x)= sin (x)* $x_0 = \pi/2;$ $n = 4$

9.7.5 Выполнить задание 9.7.2. с использованием меню **Символы**.

9.7.6 Выполнить задание 9.7.3. с использованием меню **Символы**.

9.7.7 Привести подобные члены в выражении, используя меню **Символы**

$$3\cdot x^3 - 2\cdot a\cdot x^2 + a\cdot x + 4\cdot x^2 - 3\cdot x + x^2 - x$$

9.7.8 Разложить на элементарные дроби выражение

$$\frac{5x^2 - 10x - 27}{x^3 - x^2 - 9x + 9}$$

9.7.9 С помощью меню **Символы** с найти производные *f '(x)* и упростить полученные выражения:

а) $f(x) = \frac{3x^2 - x + 7}{2x + 5}$ б) $f(x) = ln\sqrt{\frac{x^2 + 1}{x^2 - 1}}$

9.7.10 С помощью меню **Символы** вычислить неопределенные интегралы и упростить полученные выражения;

а) $\int \frac{x^3 + 1}{x(x-1)^2} dx$

б) $\int \frac{dx}{x^2\sqrt{a^2 - x^2}}$

в) $\int x \cdot cos(ax) \cdot dx$

9.7.11 Решить уравнения:

а) $ln\left(\frac{2x + 1}{a \cdot x}\right) - 3 = b$

б) $\frac{a \cdot x + 1}{x - b} = e^{-a}$

в) $sin(x) = cos(2 \cdot x)$

9.7.12 Решить неравенства:

а) $\frac{x^2 + x}{x - 3} < 0$ б) $ln(2x + 1) < ln(5 + x)$

9.7.13 Заменить в выражении $\frac{2 \cdot t}{t^2 - 4}$ переменную t на x + 3 и полученное выражение разложить на элементарные дроби.

9.8 Составление программ

9.8.1 Написать программу вычисления количества отрицательных, положительных и нулевых элементов вектора.

9.8.2 Составить программу вычисления произведения четных положительных чисел, меньших заданного *n*.

9.8.3 Для заданных a, b, d получить вектор, содержащий числа *a, a + d, d +2d, ..., b*. Например, если *a = 1, b = 2, d = 0.2*, то искомый вектор должен содержать числа *1, 1.2, 1.4, 1.6, 1.8, 2*.

9.8.4 Получить вектор из n наименьших целых положительных чисел, делящихся хотя бы на одно из заданных натуральных *a, b*.

9.8.5 Написать программу, результатом работы которой должна быть единичная матрица порядка n.

9.8.6 Преобразовать натуральное число в вектор, компонентами которого являются десятичные цифры числа.

9.8.7 Выполнить преобразование, обратное описанному в задаче 9.8.6.

9.8.8 Перемешать случайным образом компоненты вектора.

9.8.9 Написать программу для нахождения пересечения двух множеств чисел, содержащихся в векторах V и W.

9.8.10 С помощью рекурсивного определения функции написать программу вычисления сложных процентов: какова будет сумма вклада через n лет, если начальный вклад VH, и годовой процент - PROC.

9.8.11 Написать программу вычисления корня уравнения $f(x) = 0$ с точностью ε методом половинного деления.

9.8.12 Составить программу нахождения корня уравнения $f(x) = 0$ с точностью ε методом касательных.

ЛИТЕРАТУРА

1 Литература основная:

1.1 Дьяконов В.П. Книга Энциклопедия Mathcad 2001i и Mathcad 11. "Солон-Пресс", 2004. - 832 с.

1.2 Кирьянова Д. Mathcad 11.Самоучитель С-Пб: БХВ-Петербург, 2003. - 538 с.

1.3 Кудрявцев Е.М. Справочник по Mathcad 11. Издательство: ДМК Пресс, 2005. -184 с.

1.4 Плис А. И. Mathcad: Математический практикум для инженеров и экономистов: учеб. пособие / А. И. Плис, Н. А. Сливина. 2 е изд., перераб. и доп. М. : Финансы и статистика, 2003. - 656 с. : ил.

2 Литература дополнительная

2.1 MathCAD 6.0 Plus. Финансовые, инженерные и научные расчеты в среде Windows 95./Перевод с англ. - М.: Информационно издательский дом “Филинъ”, 1996. – 712 с.: ил.

2.2 Дьяконов В. Mathcad 2000 : учеб. курс / В. Дьяконов. СПб. : Питер, 2001. - 592 с. : ил.

2.3 Дьяконов В.П., Абраменкова И.В. MathCad Pro в математике, физике и Internet, М.: Нолидж: 2000, 512 с.: ил.

2.4 Дьяконов В.П. Справочник по MathCad 7.0 Pro.-М.:СК-ПРЕСС.-1998.-352 с.

2.5 Дьяконов В.П. Компьютерная математика. Теория и практика. М.: Нолидж, 2001. – 1296 с., ил.

2.6 Очков В.Ф. MathCad 7 Pro для студентов и инженеров.-М.: Компьютер ПРЕСС.- 1998.- 384 с.: ил.

2.7 Очков В.Ф. MathCad 8 Pro для студентов и инженеров.-М.:Компьютер ПРЕСС.-1999.- 523 с.:ил.

2.8 Поршнев С. В. Компьютерное моделирование физических процессов с использованием пакета MathCAD : учеб. пособие / С. В. Поршнев. - М. : Горячая линия : Телеком, 2002. - 252 с.

2.9 Поршнев С. В. Вычислительная математика. Курс лекций. СПб.: БХВ-Петербург, 2004. - 320 с.: ил.

2.10 Поршнев С.В., Беленкова И.В. Численные методы на базе MATHCAD. – СПб.: БХВ-Петербург, 2005. – 464 с.: ил.

2.11 Плис А. И. Mathcad 2000 : Математический практикум для экономистов и инженеров : учеб. пособие / А. И. Плис, Н. А. Сливина. М. : Финансы и экономика, 2000. - 656 с. : ил.

2.12 Плис А.И., Сливина Н.А. Mathcad 2000. –М.: Финансы и статистика, 2000.- 656 с.

3 Методические пособия

3.1 Николаев Н.А. Решение задач в системе MathCAD 2000. Методическое пособие. - Новоуральск, НПИ МИФИ, 2003. - 68 с. :ил.

3.2 Тихонова Е.В. Введение в MathCad. Методическое пособие. Новоуральск, НТИ НИЯУ МИФИ, 2012, - 80 с.

4 Обучающие системы и электронная документация (каталог EDUCATION сервера кафедры)

4.1 Mathcad. User's Guide. Mathcad 2000 Professional. Техническая документации, поставляемая с пакетом (англ.).
Файл Z:\EDUCATION\MathCad\Mathcad.pdf.

4.2 Электронный учебник по работе в MathCAD 7 PRO.
Файл Z:\EDUCATION\MathCad\MathCAD 7 PRO - электронный учебник.chm

4.3 Электронный учебник по работе в MathCAD 12.
Файл Z:\EDUCATION\MathCad\MathCAD 12 - электронный учебник.chm

УДК 681.3.06

Автор Тихонова Евгения Валерьевна

РЕШЕНИЕ ЗАДАЧ В СИСТЕМЕ MATHCAD

Методическое пособие по курсам «Информатика»,
«Решение инженерных задач на ЭВМ»,
«Вычислительные методы в инженерных расчетах»
для преподавателей и студентов всех специальностей
очной, очно-заочной форм обучения.

Новоуральск, НТИ НИЯУ МИФИ, 2013. - 75 с.

Сдано в печать Формат А5 Бумага писчая
Печать плоская Уч.-изд.л. 2.4 Тираж 50 экз.
Заказ Издательство НТИ Лицензия ИД №00751 г. Новоуральск, Ленина, 85