Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информационные технологии»

ПРИМЕНЕНИЕ СИСТЕМ MATCAD В ИНЖЕНЕРНЫХ РАСЧЕТАХ

ПОСОБИЕ
по выполнению лабораторных работ по курсу «Информатика» для студентов технических специальностей дневной формы обучения

Гомель 2008

УДК 004.451(075.8)
ББК 32.973-018.2я73
П76

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 7 от 10.03.2008 г.*)

Составители: *Т. А. Трохова, Т. Л. Романькова*

Рецензент: начальник сектора программных средств АСУ вычислительного центра ГГТУ им. П. О. Сухого *Н. С. Шестакова*

П76 **Применение** системы MatCad в инженерных расчетах : пособие по курсу по курсу «Информатика» для студентов техн. специальностей днев. формы обучения / сост.: Т. А. Трохова, Т. Л. Романькова. – Гомель : ГГТУ им. П. О. Сухого, 2008. – 52 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://gstu.local/lib. – Загл. с титул. экрана.

Описаны приемы решения задач для лабораторных работ по курсу «Информатика» с примерами решения инженерных задач с помощью системы MatCad.

Для студентов технических специальностей дневной формы обучения.

УДК 004.451(075.8)
ББК 32.973-018.2я73

1. Создание программных фрагментов в MathCad

1.1. Операторы палитры «Программирование»

Система MathCad позволяет создавать программные фрагменты для вычисления алгоритмов, которые нельзя реализовать базовым набором средств и методов Mathcad.

Программный фрагмент можно использовать в операторе «=» или в правой части оператора «**:=**». Слева в операторе «**:=**» может находиться либо переменная, либо пользовательская функция.

Программный фрагмент состоит из строк программы, каждая из которых может содержать операторы программы.

Для создания программного фрагмента используется панель программирования, показанная на рис. 1.1:

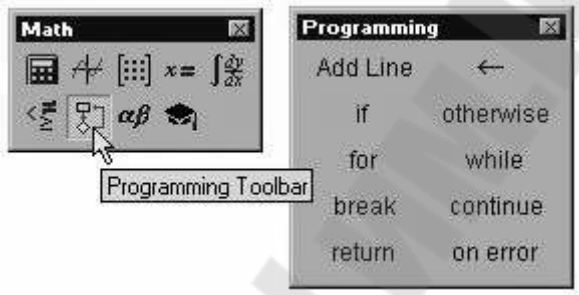


Рис. 1.1. Палитра «Программирование».

Кнопки этой палитры имеют следующее назначение:

Add Line	Создание и расширение программного фрагмента.
←	Оператор внутреннего локального присваивания.
if	Оператор условия.
for	Оператор цикла с заданным числом повторений.
while	Оператор цикла с предусловием.
otherwise	Оператор "*иначе*".
break	Оператор прерывания работы цикла (если этот оператор используется в теле цикла) или программы.
continue	Оператор возврата к началу цикла.
return	Выход из программы с возвратом значения выражения указанного после оператора ***return***

Результаты работы программного фрагмента передаются следующим образом: во внешнюю вычислительную область из программного фрагмента передается значение последнего выражения, вычисленного в программном фрагменте или значение переменной, имя которой записано в последней строке программы.

Программный фрагмент может заканчиваться оператором «=», а может быть присвоен переменной. Программный фрагмент может

участвовать при создании пользовательской функции, которая затем вычисляется при конкретном значении ее аргумента. Внутренние переменные программного фрагмента являются неопределенными вне его.

1.2. Программирование разветвляющихся алгоритмов

Для программирования разветвляющихся алгоритмов используются оператор условия и оператор ***otherwise***.

Общий вид оператора **if**:

выражение if условие

Порядок выполнения: если логическое выражение, стоящее в ***условии***, истинно, то вычисляется ***выражение***, стоящее слева от оператора if.

Оператор ***otherwise*** обычно используется совместно с ***if*** для выполнения действий в случае невыполнения условия.

Общий вид оператора **otherwise**:

выражение otherwise

Примером разветвляющегося алгоритма может служить алгоритм вычисления кусочно-непрерывной функции, когда значение функции вычисляется по разным аналитическим зависимостям при различных значениях аргумента.

Последовательность действий для создания программного фрагмента вычисления значения кусочно-непрерывной функции такова:

- открыть палитру программирования и палитру логических операторов;
- набрать имя пользовательской функции, например, Y(x);
- с помощью оператора «**:=**» и кнопки «**Add line**» палитры программирования сформировать шаблон для записи операторов вычисления значения функции для различных диапазонов значения аргумента, например:

  Y(x) :=
  ▪
  ▪
  ▪ ;

- в каждую строку программного фрагмента, кроме последней, вывести шаблон оператора условия, используя кнопку «**if**» палитры программирования, например:

  Y(x) :=
  ▪ **if** ▪
  ▪ **if** ▪
  ▪ ;

- в каждом операторе условия слева от ***if*** набрать выражение для вычисления значения функции, а справа, используя кнопки палитры логических операторов, набрать условие, определяющее диапазон значения аргумента, например:

$$\mathbf{Y(x)} := \begin{cases} \mathbf{x}^2 & \textbf{if } \mathbf{x} \ge 0 \\ \mathbf{sin(x)} & \textbf{if } -3 < \mathbf{x} \le -1 \\ \blacksquare & \end{cases};$$

- в последнюю строку программного фрагмента вывести шаблон оператора «иначе», используя кнопку «**otherwise**» палитры программирования;
- заполнить шаблон оператора «иначе» выражением для вычисления значения функции, например:

$$\mathbf{Y(x)} := \begin{cases} \mathbf{x}^2 & \textbf{if } \mathbf{x} \ge 0 \\ \mathbf{sin(x)} & \textbf{if } -3 < \mathbf{x} \le -1 \\ \mathbf{cos}\left(\mathbf{x}^2\right) & \textbf{otherwise} \end{cases};$$

- получить значение функции Y(x) для различных значений аргумента x.

Пример 1.1. Вычислить значение кусочно-непрерывной функции

$$Y(x) = \begin{cases} 3 \cdot x^2, & \textit{если} \quad x > 10 \\ 3 \cdot x, & \textit{если} \quad x \le 2 \\ 4 \cdot x, & \textit{в остальных случаях} \end{cases}$$

для значений аргумента $x = 15$, $x = 5$ и $x = -1$ с использованием программного фрагмента.

Реализация данного примера в MathCad приведена на рис. 1.2.

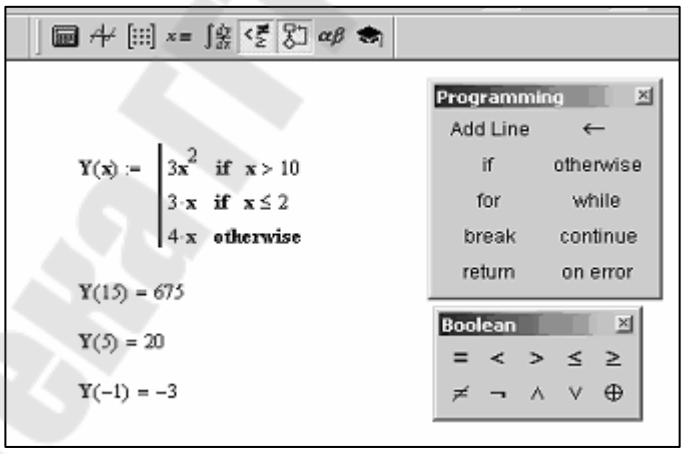


Рис. 1.2. Пример вычисления кусочно-непрерывной функции.

Пример 1.2. Вычислить значение скорости выходного звена кулачкового механизма для различных значений угла φ по известному закону движения:

$$S1 = \begin{cases} h\dfrac{\varphi}{\varphi_1\left(\varphi_y - \varphi_1\right)} & \textit{при} \quad 0 \le \varphi \le \varphi_1 \\ h\dfrac{1}{\varphi_y - \varphi_1} & \textit{при} \quad \varphi_1 < \varphi \le \left(\varphi_y - \varphi_1\right) \\ h\dfrac{\varphi_y - \varphi}{\varphi_1\left(\varphi_y - \varphi_1\right)} & \textit{при} \left(\varphi_y - \varphi_1\right) < \varphi \le \varphi_y \end{cases},$$

где S1 – скорость, h– максимальная скорость, φ- текущее значение угла, φ_1- промежуточное значение угла, φ_y- конечное значение угла ($\varphi_y = 2\pi$),

h=6,9 φ_1=1,57

Реализация данного примера в MathCad приведена на рис. 1.3.

$$h := 6.9 \qquad \phi_1 := 1.57 \qquad \phi_y := 2\cdot\pi$$

$$S1(\phi) := \begin{cases} h\cdot\dfrac{\phi}{\phi_1\cdot(\phi_y-\phi_1)} & \text{if } 0 \le \phi \le \phi_1 \\ h\cdot\dfrac{1}{(\phi_y-\phi_1)} & \text{if } \phi_1 < \phi \le (\phi_y-\phi_1) \\ h\cdot\dfrac{\phi_y-\phi}{\phi_1\cdot(\phi_y-\phi_1)} & \text{if } (\phi_y-\phi_1) < \phi \le \phi_y \end{cases}$$

$$S1(0.85) = 0.793 \qquad S1(2.59) = 1.464 \qquad S1(5.43) = 0.796$$

Рис. 1.3. Пример вычисления кусочно-непрерывной функции.

1.3. Программирование циклических алгоритмов

Для программирования циклических алгоритмов используются операторы цикла ***for*** и **while.**

Общий вид ***for*** :

for Var ∈ Nmin .. Nmax.

Nmin .. Nmax – диапазон изменения переменной цикла, организуется по правилам формирования дискретных переменных.

Порядок выполнения: переменную цикла ***Var*** изменяет свое значение в пределах от ***Nmin*** до ***Nmax*** с шагом 1 или -1, при каждом новом значении переменной цикла выполняется рабочая часть цикла.

Общий вид ***while***:

while условие

Рабочая часть цикла записывается на месте шаблона

Порядок выполнения: рабочая часть цикла выполняется до тех пор, пока логическое выражение, стоящее в условии, истинно.

Примерами циклического алгоритма могут служить алгоритмы вычисления суммы однотипных слагаемых и произведения однотипных множителей.

Последовательность действий для создания программного фрагмента, реализующего вычисление суммы (произведения) однотипных слагаемых (множителей), такова:

- в любой форме (например, в виде графической схемы) составить алгоритм решения задачи, например, как показано на рис. 1.4;

- открыть палитру программирования;

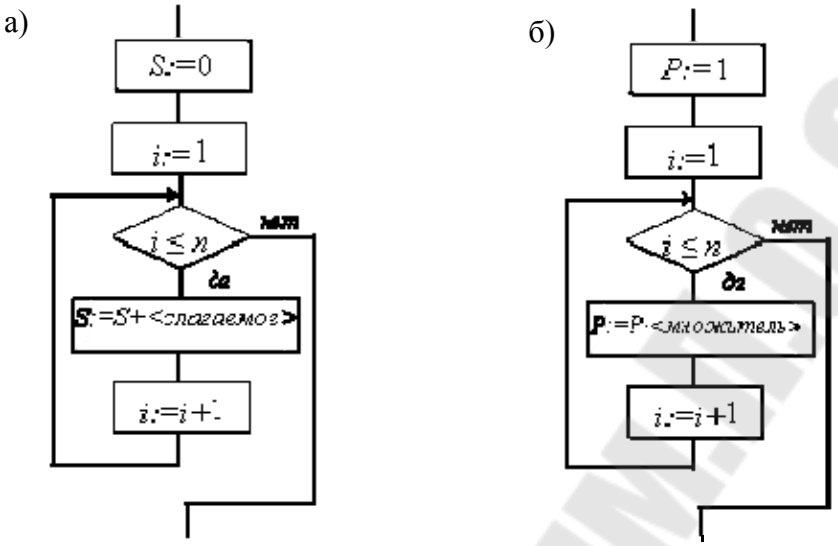


Рис. 1.4. Схемы алгоритмов вычисления

а) суммы вида $\sum_{i=1}^{n} < слагаемое >$; б) произведения вида $\prod_{i=1}^{n} < множитель >$.

- набрать имя переменной, в которую будет помещен результат вычислений, и знак «**:=**»;
- с помощью кнопки «**Add line**» палитры программирования создать программный фрагмент с необходимым количеством строк;
- в каждую строку программного фрагмента с помощью кнопок палитры программирования ввести шаблон оператора, реализующий ту или иную часть разработанного алгоритма, затем заполнить этот шаблон (например, для блока $S:=0$ нужно использовать оператор локального присваивания $S \leftarrow 0$, а для реализации цикла по переменной i - оператор цикла с заданным числом повторений

for $i \in 1..n$

$S \leftarrow S + слагаемое$,

где значение n и вид слагаемого зависят от условия задачи);
- в последней строке программного фрагмента набрать имя локальной переменной, используемой для накопления суммы (произведения);
- вывести значение переменной, которой присваивается результат выполнения программного фрагмента, с помощью оператора «=».

Пример 1.3. С использованием программного фрагмента вычислить сумму

$$\sum_{i=1}^{125} \frac{i}{\cos(i)+2}.$$

Схема алгоритма решения приведена на рис. 1.5.

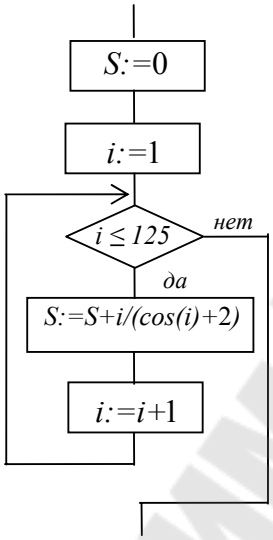


Рис. 1.5. Схема алгоритма вычисления $\sum_{i=1}^{125} \frac{i}{\cos(i)+2}$

На рис. 1.6 приведена реализация данного примера в MathCad.

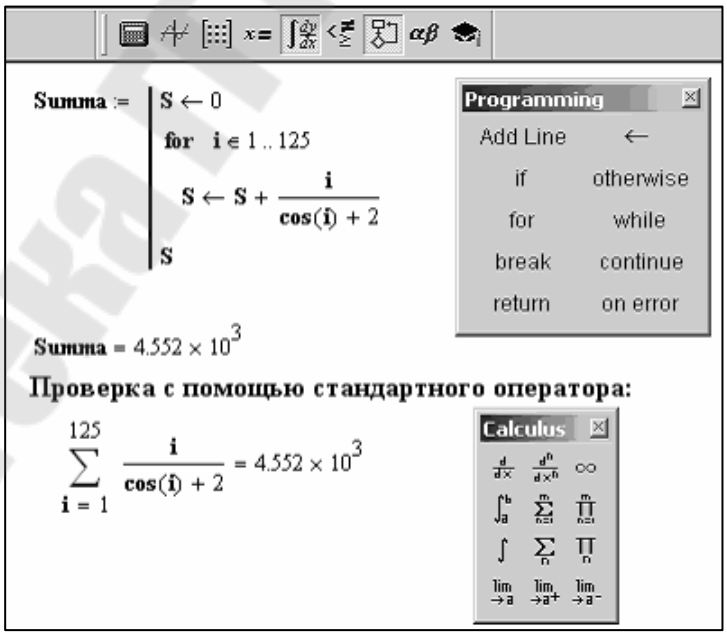


Рис. 1.6. Вычисление $\sum_{i=1}^{125} \frac{i}{\cos(i)+2}$.

Примером циклического алгоритма может также служить алгоритм табулирования функции.

1.4. Программирование алгоритмов работы с массивами

Последовательность действий для создания программного фрагмента, предназначенного для обработки массива, такова:

- в любой форме (например, в виде графической схемы) составить алгоритм решения задачи;
- открыть все необходимые палитры;
- задать исходный вектор чисел;
- определить функцию для решения поставленной задачи в виде программного фрагмента следующим образом:

  - набрать имя функции с двумя формальными параметрами: первый - размерность массива, второй – имя массива, а затем оператор «**:=**»;

  - с помощью кнопки «**Add line**» палитры программирования создать программный фрагмент с необходимым количеством строк;
- в каждую строку программного фрагмента с помощью кнопок палитры

  программирования ввести шаблон оператора, реализующий ту или иную часть разработанного алгоритма, затем заполнить этот шаблон (для реализации цикла по номеру элемента массива *i* нужно использовать оператор цикла с предусловием **while**, если шаг изменения номера *i* не равен 1);
- в последней строке программного фрагмента набрать имя локальной переменной, содержащей результат вычисления;
- вывести значение функции для заданных исходных данных, подставив в качестве фактических параметров количество элементов массива и имя определенного выше массива, с помощью оператора «=».

Пример 1.4.

Дан вектор чисел произвольной длины. Используя программный фрагмент, вычислить произведение отрицательных элементов с четными номерами.

Схема алгоритма приведена на рис. 1.7.

Здесь x – исходный вектор, n – количество элементов этого вектора, i – номер текущего элемента вектора, P - произведение отрицательных элементов с четными номерами.

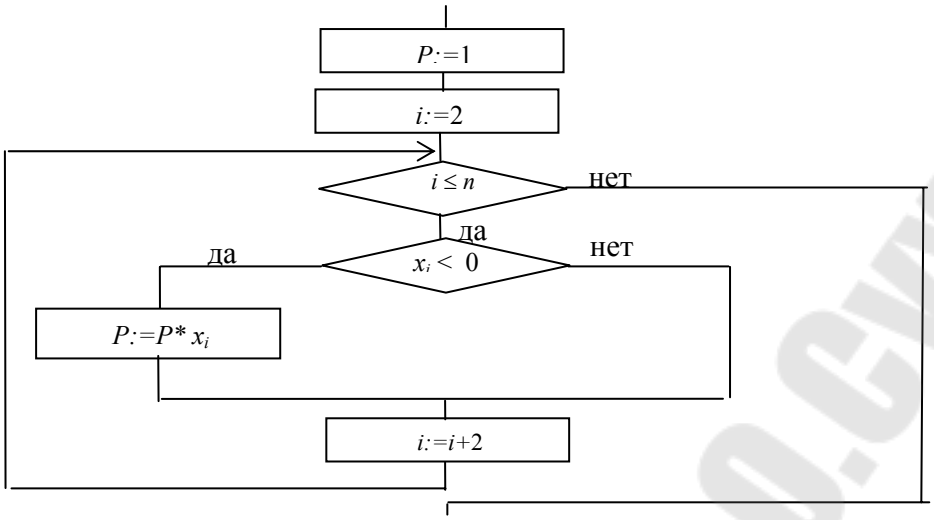


Рис. 1.7. Схема алгоритма вычисления произведения отрицательных элементов массива с четными номерами.

На рис. 1.8 приведена реализация этого алгоритма с помощью программного фрагмента.

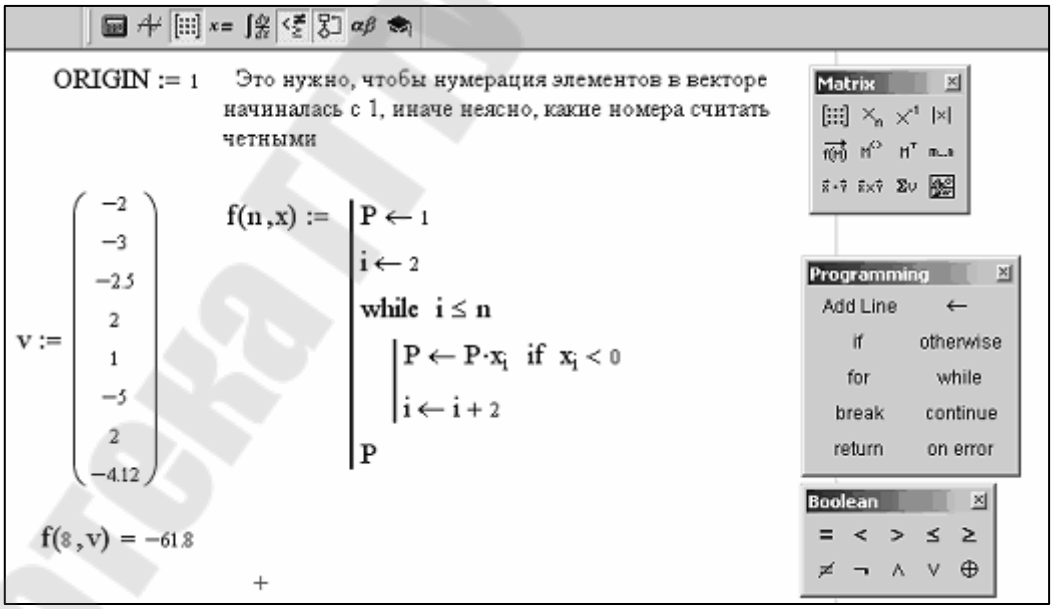


Рис. 1.8 – Вычисление произведения отрицательных элементов массива с четными номерами в MathCad.

Пример 1.5. Дана таблица значений нормативного коэффициента запаса прочности приводных роликовых цепей (табл. 1). Определить по заданному шагу цепи среднее табличное значение нормативного коэффициента запаса прочности, учитывая следующие рекомендации:

- заданное значение шага цепи может быть не равно табличному, тогда выбирается наиболее близкое из двух соседних значений,

например, заданное значение t=14, следовательно t принимается равным 15, если заданное значение t=20, для поиска t берется равным 19.

- если заданное значение t меньше или больше граничных значений в таблице, то программа должна выдать сообщение «Шага t нет в таблице».

Таблица 1.

Таблица значений нормативного коэффициента запаса прочности приводных роликовых цепей.

Угловая скорость ω, рад/с	Шаг цепи t, мм					
	12	15	19	25	31	38
5	7,1	7,2	7,2	7,3	7,4	7,5
10	7,3	7,4	7,5	7,6	7,8	8,0
31	7,9	8,2	8,4	8,9	9,4	9,8
52	8,5	8,9	9,4	10,2	11,8	12,5
78	9,3	10,0	10,7	12,0	13,0	14,0

На рис. 1.9 показано, что исходные данные можно задать в виде двух векторов и матрицы.

Массив значений шага цепи: **Массив значений угловой скорости:**

$$\mathbf{t} := \begin{pmatrix} 12 \\ 15 \\ 19 \\ 25 \\ 31 \\ 38 \end{pmatrix} \qquad \boldsymbol{\omega} := \begin{pmatrix} 5 \\ 10 \\ 31 \\ 52 \\ 78 \end{pmatrix}$$

Матрица значений нормативного коэффициента запаса прочности приводных роликовых цепей:

$$\underset{\sim}{\mathbf{K}} := \begin{pmatrix} 7.1 & 7.2 & 7.2 & 7.3 & 7.4 & 7.5 \\ 7.3 & 7.4 & 7.5 & 7.6 & 7.8 & 8 \\ 7.2 & 7.4 & 8.2 & 8.9 & 9.4 & 9.8 \\ 8.5 & 8.9 & 9.4 & 10.2 & 11.8 & 12.5 \\ 9.3 & 10 & 10.7 & 12 & 13 & 14 \end{pmatrix}$$

Рис. 1.9. Исходные данные.

Программный фрагмент для решения задачи приведен на рис. 1.10.

$$K\_z(tt, t, \omega, k) := \left| \begin{array}{l} \text{return "Шага t нет в таблице"} \quad \text{if} \quad tt < t_0 \vee tt > t_{last(t)} \\ \text{for} \quad i \in 0 .. last(t) - 1 \\ \quad \left| \begin{array}{l} n \leftarrow i \quad \text{if} \quad t_i \le tt < t_{i+1} \wedge (tt - t_i) < (t_{i+1} - tt) \\ n \leftarrow i + 1 \quad \text{if} \quad (t_i < tt) \le t_{i+1} \wedge tt - t_i > t_{i+1} - tt \end{array} \right. \\ S \leftarrow 0 \\ \text{for} \quad j \in 0 .. last(\omega) \\ \quad S \leftarrow S + K_{j,n} \\ \dfrac{S}{length(\omega)} \end{array} \right.$$

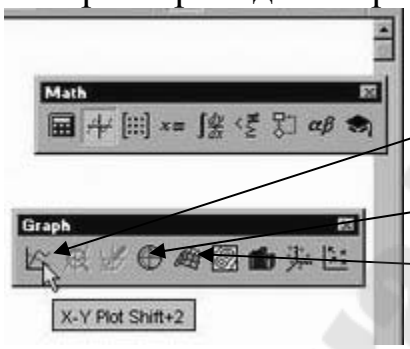
$$K\_z(33, t, \omega, K) = 9.88 \qquad K\_z(45, t, \omega, K) = \text{"Шага t нет в таблице"}$$

Рис.1.10. Определение среднего коэффициента запаса для заданного шага.

2. Построение графиков

2.1. Построение двумерных графиков

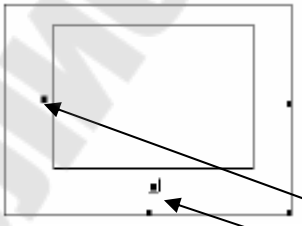
Для построения графиков используется палитра графиков, вид которой приведен на рис. 2.1.



- ***X-Y Plot*** – график в декартовой системе координат;
- ***Polar Plot***– график в полярной системе координат;
- ***Surface Plot*** – трехмерный график (график поверхности);

Рис. 2.1. Палитра графиков.

При построении двумерных графиков после нажатия соответствующей кнопки на панели графических инструментов появляется шаблон вида:



В шаблоне графика по вертикали задаются через запятую функции, а по горизонтали – аргументы. График строится по точкам

соединяющихся между собой разнообразными линиями (сплошной, пунктирной и т. д.). Исходные (узловые) точки могут быть показаны в виде маркеров (квадратов, ромбов, окружностей и т. д.). Крайние шаблоны данных служат для указания предельных значений абсцисс и ординат, т. е. они задают масштабы графика. Если оставить эти шаблоны незаполненными, то масштабы по осям графика будут устанавливаться автоматически.

Аргумент функции может быть задан в виде дискретной переменной. Тогда функция определяется в виде аналитической зависимости. Аргумент и функция могут быть заданы как одномерные массивы одинаковой длины.

После построения график может быть отформатирован по следующим направлениям: форматирование осей графика, форматирование линий графика, форматирование надписей на графике.

Для вызова окна форматирования графика нужно либо дважды щелкнуть левой кнопкой мыши на поле графика, либо, выделив график, выбрать из основного меню команду ***Format – Graph.*** Вид окна форматирования осей графика приведен на рис. 2.2.

Log Scale (Лог. масштаб) — установка логарифмического масштаба;
Crid Lines (Линии сетки) — установка линий масштабной сетки;
Numbered (Пронумеровать) — установка цифровых данных по осям;
Autoscale (Автомасштаб) — автоматическое масштабирование графика;
Show Markers (Нанести риски) — установка фоновых линий по осям;
Auto Grid (Автосетка) — автоматическая установка масштабных линий;
Nunber of Grids (Число интервалов) — установка заданного числа масштабных линий.
Boxed (Рамка) — оси в виде прямоугольника;
Crossed (Репер) — оси в виде креста;
None (Ничего) — отсутствие осей;

Рис. 2.2. Страница X-Y Axes окна редактирования графика.

При форматировании линий графика выбирается закладка **Traces** в окне форматирования, вид окна форматирования линий приведен на рис. 2.3.

Legend Label (Имя кривой) — указание названий линий в легенде графика;
Symbol (Маркер) — установка символа отметки базовых точек графика;
Line (Линия) — установка типа линий;
Color (Цвет) — установка цвета линии и базовых точек;
Type (Тип) — тип графиков;
Weight (Толщина) — толщина линий.
Hide Argument — скрываются обозначения математических выражений по осям графика;
Hide Legend — скрываются обозначения имен кривых графика.

Рис. 2.3. Страница Traces окна редактирования графика.

Последовательность действий для построения графиков функции и ее первой и второй производных, а также графического определения точек экстремума, такова:

- описать исходную функцию, например,
  $\mathbf{f(x) := \sin(3 \cdot x)}$;
- описать функции первой и второй производных с использованием соответствующих кнопок палитры математического анализа, например,
  $\mathbf{f1(x) := \frac{d}{dx} f(x)} \qquad \mathbf{f2(x) := \frac{d^2}{dx^2} f(x)}$;
- определить аргумент в виде дискретной переменной, например,
  $\mathbf{x := 0, \frac{\pi}{50} .. \pi}$;
- открыть палитру графиков;
- нажать кнопку ***X-Y Plot*** палитры для построения графика в декартовой системе координат, после чего в рабочий документ по местоположению курсора будет вставлен шаблон графика;
- в поле ввода ■ по оси абсцисс поместить имя аргумента, а по оси ординат ввести имена функций через запятую (например, **f(x), f1(x), f2(x)**);

- двойным щелчком мыши по области графика открыть окно форматирования;
- произвести форматирование осей графика, рекомендуется установить оси в виде креста;
- для вывода легенды в окне форматирования графика выбрать закладку **Traces** и задать названия для каждой функции в поле ***Legend Label***:
  - выбрать в списке соответствующую линию, например, **trace1**;
  - в поле ввода под списком набрать название линии,

  например,

  для **f(x)** вместо **trace1** набрать **функция f(x)**,

  для **f1(x)** вместо **trace2** набрать **первая производная функции f(x)**;
- щелчком мыши убрать флажок переключателя ***Hide Legend***;
- для нанесения фоновых линий выбрать закладку **X-Y Axes** окна форматирования и щелчком мыши установить флажок переключателя ***Show Markers*** для оси абсцисс (**X-Axis**);
- закрыть окно форматирования графика нажатием кнопки **ОК**, при этом под осью абсцисс появится легенда и два пустых поля ■ для определения точек нанесения фоновых линий;
- для определения точек экстремума функции выполнить команду меню **Format – Graph – Trace…**, после чего будет открыто окно **X-Y Trace**

(*Замечание*: в точках экстремума первая производная функции равна нулю);
- щелчком мыши убрать флажок переключателя ***Track Data Points***;
- при нажатой левой кнопке мыши перемещать указатель вдоль линии графика первой производной до точки пересечения ее с осью абсцисс, после чего отпустить кнопку мыши, при этом в поле ***X-Value*** окна **X-Y Trace** появится значение аргумента, при котором первая производная равна нулю;
- заполнить одно из пустых полей ■ под осью абсцисс, набрав значение из поля ***X-Value***;
- для визуализации графика щелкнуть левой кнопкой мыши или нажать клавишу **Enter**;
- закрыть окно **X-Y Trace.**

Пример 2.1 Построить график функции $f(x) = \sin(3 \cdot x)$ и графики ее первой и второй производной. Нанести фоновые линии в точках экстремума функции $\left(\frac{d}{dx} f(x) = 0\right)$. Функция должна рассчитываться не менее чем в n=50 точках, при $x_{нач} = 0$ и $x_{кон} = \pi$.

На рис. 2.4 приведена реализация данного примера в MathCad.

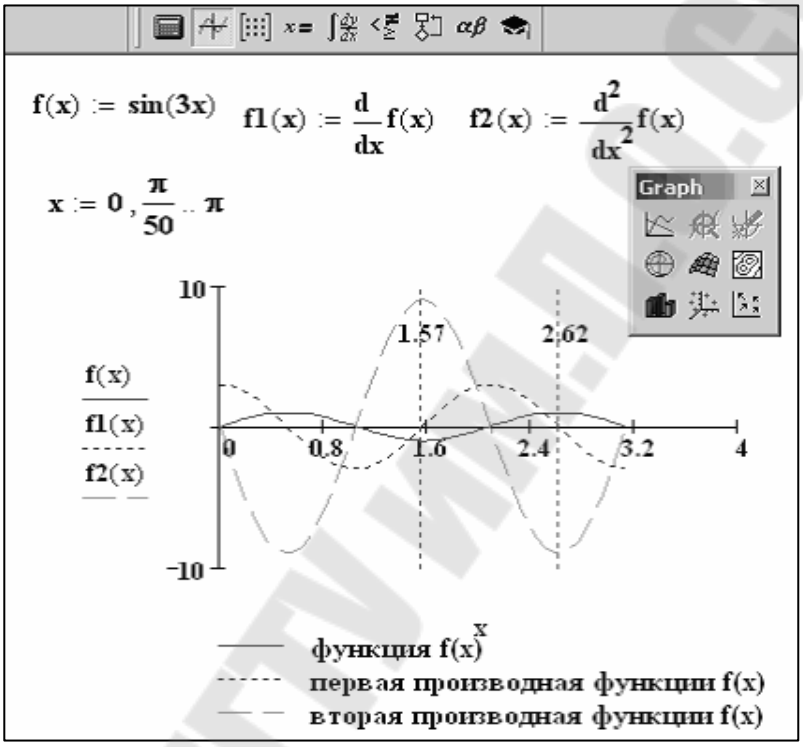


Рис. 2.4. Построение графика функции, ее первой и второй производных.

Пример 2.2. Построить график зависимости числа витков заданной электрической катушки с определенным набором исходных данных от значения индуктивности

$$W = \sqrt{\frac{L_0}{2\pi\left(D - \sqrt{D^2 - D_1^2}\right)}}, \text{ где}$$

D=3 см и D_1=1 см – диаметры, а индуктивность L_0 принимает значения в пределах от L_0нач =100000нГн до L_0кон =110000нГн с шагом L_0шаг=1000нГн.

На рис. 2.5 приведена реализация данного примера в MathCad.

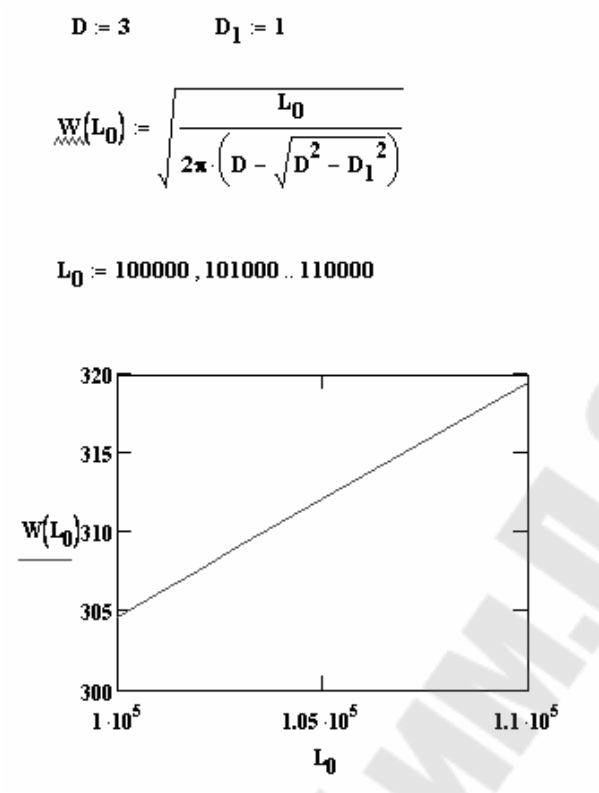


Рис. 2.5. Построение графика зависимости числа витков заданной электрической катушки от значения индуктивности.

2.2. Построение графиков кусочно-непрерывных функций

Последовательность действий для построения графика кусочно-непрерывной функции такова:

- с помощью программного фрагмента задать кусочно-непрерывную функцию, например,

$$y(x) := \begin{cases} 3\sin(x) & \text{if } x > 25 \\ \frac{x^2}{15} & \text{if } x < 12 \\ (x - 20) & \text{otherwise} \end{cases};$$

- определить аргумент в виде дискретной переменной таким образом, чтобы функция вычислялась по каждой из формул, например $x := 1,1.5 .. 35$;
- построить график заданной функции;
- двойным щелчком мыши по области графика открыть окно форматирования;
- для нанесения вспомогательных линий щелчком мыши установить флажок переключателя ***Crid Lines*** для каждой оси;

- для ввода названий осей и самого графика в окне форматирования выбрать закладку **Labels**, ввести заголовок графика в поле ***Title*** и установить флажок переключателя ***Show Title***;
- ввести названия осей в полях ввода панели ***Axis Labels***;
- для изменения толщины линии выбрать закладку **Traces** и задать толщину линии графика, используя список поля ***Weight*** :

;

- закрыть окно форматирования графика нажатием кнопки **ОК**.

Пример 2.3. Построить график кусочно-непрерывной функции

$$y=\begin{cases}3\sin(x), & если \quad x>25\\ \dfrac{x^2}{15}, & если \quad x<12\\ x-20, & в \quad остальных \quad случаях\end{cases}.$$

Пределы изменения аргумента подобрать так, чтобы перекрывались все три диапазона. При задании вида функции использовать программный фрагмент.

На рис. 2.6 приведена реализация данного примера в MathCad.

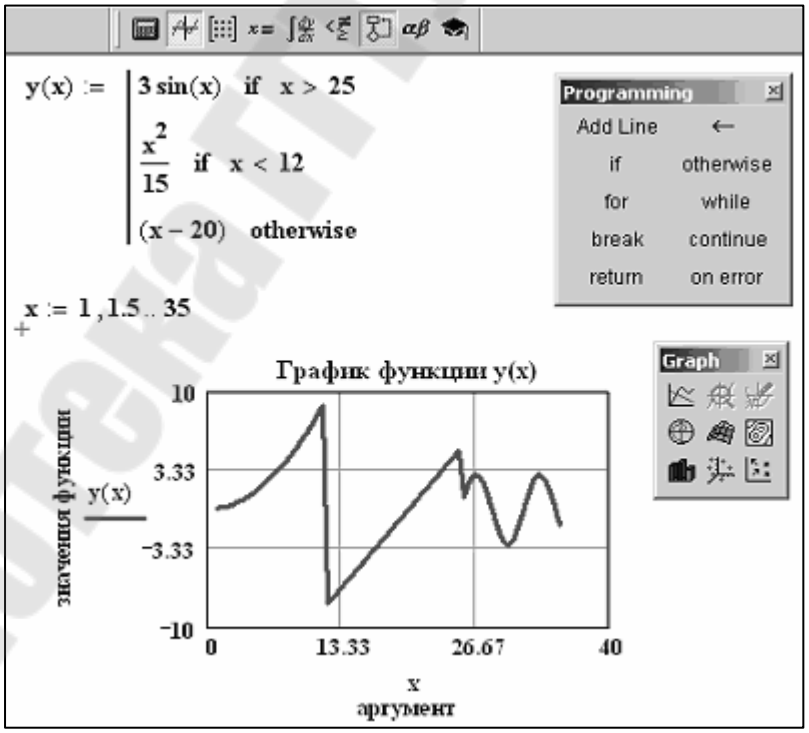


Рис. 2.6. Построение графика кусочно-непрерывной функции.

Пример 2.4. Построить график функции аналога ускорения толкателя кулачкового механизма, если угол φ изменяется от 0 до 2π.

φ_1=1,75; φ_2=3,18; φ_3=5,22; a_1=8,2; a_2=6.

$$S(\varphi)=\begin{cases} a_1 \sin\dfrac{2\pi}{\varphi_1}\varphi & при \quad 0 \le \varphi < \varphi_1 \\ 0 & при \quad \varphi_1 \le \varphi < \varphi_2 \\ -a_2 \sin\dfrac{2\pi}{\varphi_3-\varphi_2}\varphi & при \quad \varphi_2 \le \varphi < \varphi_3 \\ 0 & при \quad \varphi_3 \le \varphi < 2\pi \end{cases}$$

На рис. 2.7 приведена реализация данного примера в MathCad.

$\phi_1 := 1.75$ $\phi_2 := 3.18$ $\phi_3 := 5.22$ $a_1 := 8.2$ $a_2 := 6$

$\phi := 0, 0.01 .. 2\cdot\pi$

$$S(\phi) := \begin{cases} a_1\cdot\sin\left(\dfrac{2\cdot\pi\cdot\phi}{\phi_1}\right) & \text{if } 0 \le \phi < \phi_1 \\ 0 & \text{if } \phi_1 \le \phi < \phi_2 \\ \left(-a_2\cdot\sin\left(\dfrac{2\cdot\pi\cdot\phi}{\phi_3-\phi_2}\right)\right) & \text{if } \phi_2 \le \phi < \phi_3 \\ 0 & \text{if } \phi_3 \le \phi < 2\cdot\pi \end{cases}$$

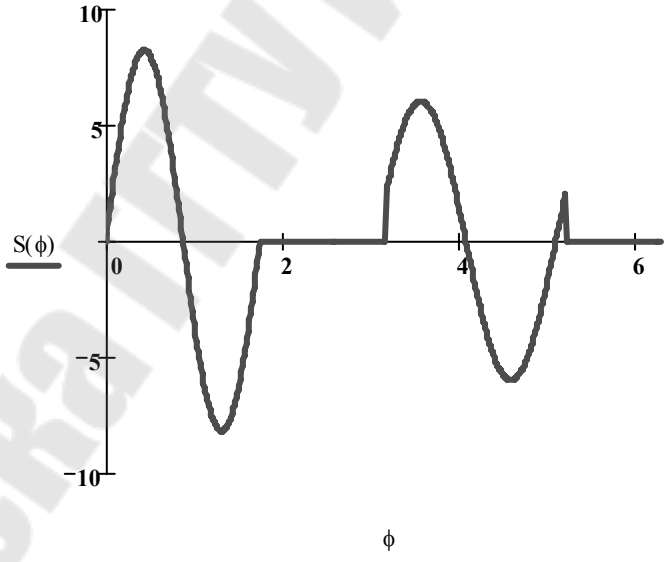


Рис. 2.7. Построение графика функции аналога ускорения толкателя кулачкового механизма.

2.3. Построение графиков поверхностей

График поверхности представляет собой графическое изображение матрицы. Поэтому необходимо сначала задать эту матрицу. Это можно сделать следующим образом:

- Определить функцию двух переменных, описывающую поверхность. Например, $f(x,y) := \sin(x^2 + y^2)$.
- Задать номера строк и столбцов в виде дискретных переменных. Например, $i := 0..20 \qquad j := 0..20$
- Поэлементно сформировать матрицу, элементами которой будут являться значения заданной функции. Например,

$$M_{i,j} := f(\frac{i-10}{5}, \frac{j-10}{5}).$$

Можно также сначала сформировать векторы аргументов функции, например:

$i := 0..20 \qquad j := 0..20 \qquad x_i := -5 + i \cdot 0.25 \qquad y_j := -1 + j \cdot 0.2$.

А затем сформировать матрицу, используя в качестве аргументов заданной функции элементы созданных векторов:

$$M_{i,j} := f(x_i, y_j).$$

Матрица также может быть сформирована вручную или получена в результате выполнения каких либо вычислений. Например, в результате численного решения уравнения теплопроводности с заданными начальными и краевыми условиями получается матрица распределения температуры. Это температурное поле можно отразить на графике.

Последовательность действий для построения графика поверхности такова:

- подготовить матрицу значений одним из описанных выше способов;
- открыть палитру графиков;
- нажать кнопку ***Surface Plot*** палитры для построения трехмерного графика, после чего в рабочий документ по местоположению курсора будет вставлен шаблон графика;
- в поле ввода ■ поместить имя матрицы;
- двойным щелчком мыши по области графика открыть окно форматирования и установить необходимые параметры, используя следующие вкладки:
  - **General** (Общие) – общие параметры изображения;
  - **Axes** (Оси) – параметры координатных осей (тип, толщина и цвет линий осей, число отметок, нумерация, масштаб и др.);
  - **Appearance** (Вид) – параметры отображения графика (цвет линий и тип точек, используемых для построения фигур и поверхностей);
  - **Backplanes** (Грани) – параметры граней;
  - **Lighting** (Освещение) – параметры условий и схемы освещения;

- **Special** (Специальные) – специальные параметры (контурные линии, столбцы, интерполяция по цвету и др.);
- **Advanced** (Дополнительно) – дополнительные параметры (перспектива, световые эффекты, качество печати и др.);
- **Quick Plot Data** (Быстрое построение графика по данным) – параметры быстрого построения графиков.

Пример 2.5. Построить гиперболический параболоид, каноническое уравнение которого имеет вид:

$$\frac{x^2}{p} - \frac{y^2}{q} = 2z \ (p > 0, q > 0).$$

Реализация данного примера приведена на рис. 2.8.

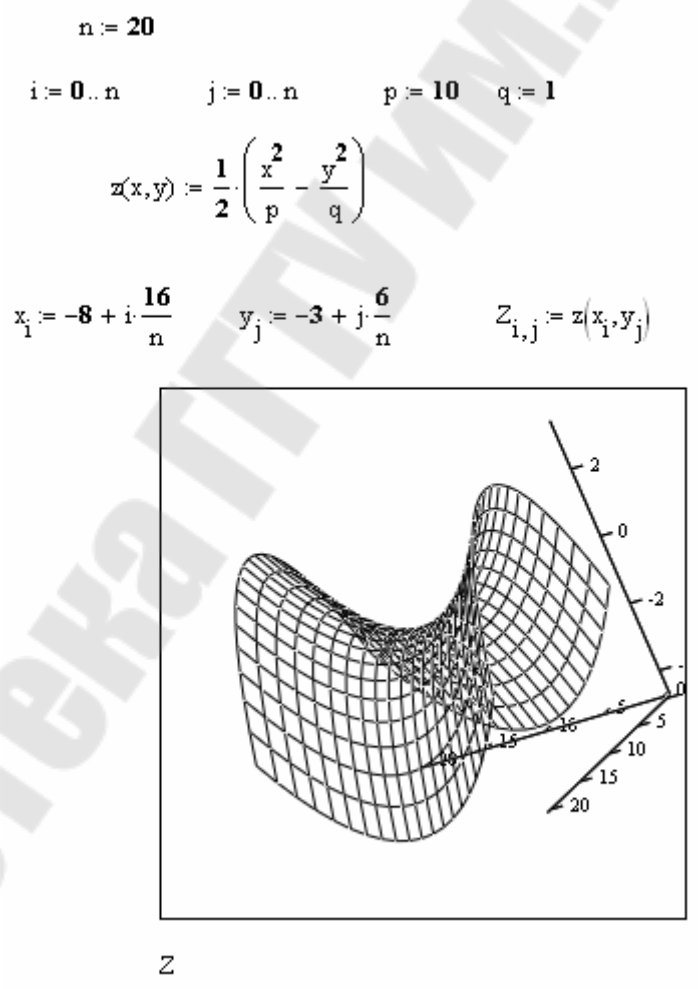


Рис. 2.8. Построение гиперболического параболоида.

3. Решение уравнений и систем
3.1. Поиск корней уравнения, графическая интерпретация

Решение алгебраических уравнений вида ***f(x)=0*** в MathCad осуществляется с помощью стандартной функции ***root.***

Функция имеет следующий общий вид:

root(f(x), x), где

f(x) – функция, описывающая левую часть выражения вида ***f(x)=0,***

x – имя переменной, относительно которой решается уравнение.

Функция ***root*** реализует алгоритм поиска корня численным методом и требует предварительного задания начального приближения искомой переменной ***x***. Поиск корня будет производиться вблизи этого числа. Таким образом, присвоение начального значения требует предварительной информации о примерной локализации корня.

Функция позволяет найти как вещественные корни, так и комплексные. В случае комплексного корня начальное приближение нужно задать в виде комплексного числа.

Если после многих итераций Mathcad не находит подходящего приближения, то появится сообщение «отсутствует сходимость».

Эта ошибка может быть вызвана следующими причинами:

- уравнение не имеет корней;
- корни уравнения расположены далеко от начального приближения;
- выражение ***f(x)*** имеет разрывы между начальным приближением и корнем;
- выражение имеет комплексный корень, но начальное приближение было вещественным и наоборот.

Для изменения точности, с которой функция ***root*** ищет корень, нужно изменить значение системной переменной ***TOL***. Например, просле задания в документе оператора ***TOL:=0.00001*** точность вычисления корня станет равной 0.00001. На рис. 3.1 приведены примеры вычисления корней уравнений с помощью функции ***root.***

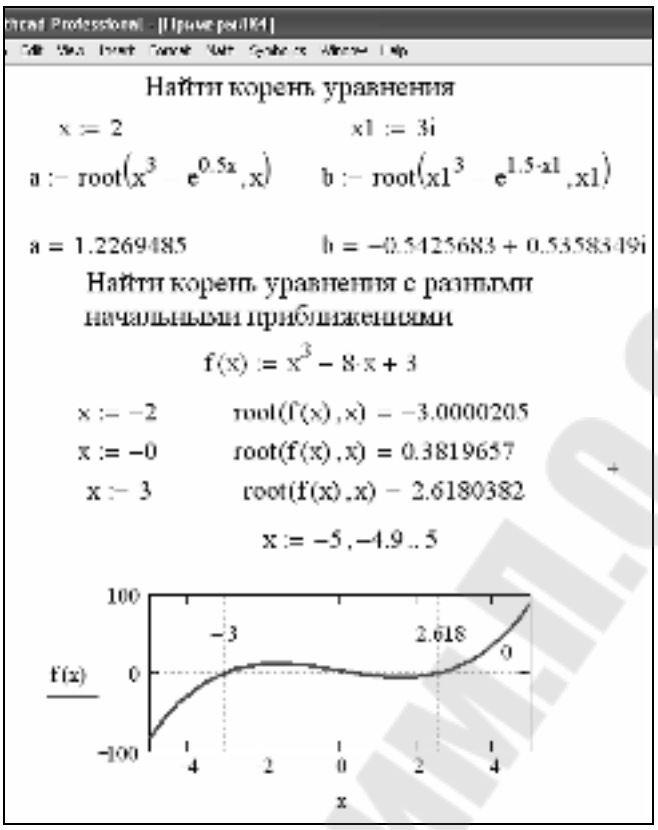


Рис. 3.1. Примеры решения уравнений

Последовательность действий для нахождения корня уравнения с применением функции **root** такова:

- привести уравнение к виду ***f(x)=0***, если это необходимо, например, уравнение $\sin(2x)+\cos(2x)=\sqrt{x}\sin(3x)$ преобразуется в $\sin(2x)+\cos(2x)-\sqrt{x}\sin(3x)=0$;
- установить курсор в свободное место рабочего окна документа MathCad;
- определить функцию, содержащуюся в левой части уравнения вида ***f(x)=0***, например, $f(x):=\sin(2x)+\cos(2x)-\sqrt{x}\sin(3x)$;
- задать начальное приближение искомой переменной ***x***;
- подставляя в качестве параметров функции ***root*** имя функции левой части уравнения и переменную, содержащую начальное приближение, вывести значение корня уравнения при заданном начальном приближении с помощью оператора «=»;
- задать аргумент функции ***f(x)*** в виде дискретной переменной, при этом диапазон изменения аргумента должен включать найденный корень уравнения;
- построить график функции ***f(x)***;
- установить оси графика в виде креста;
- нанести фоновую линию в точке, совпадающей с найденным корнем уравнения.

Пример 3.1. Найти корень уравнения $\sin(2x)+\cos(2x)=\sqrt{x}\sin(3x)$ при заданном начальном значении $x \approx 5$. Выполнить графическую интерпретацию результата.

Реализация данного примера в MathCad приведена на рис. 3.2.

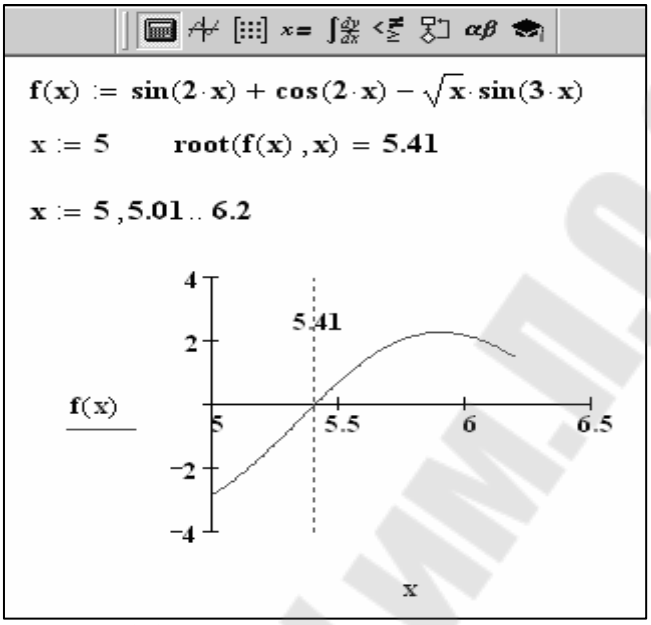


Рис. 3.2. Примеры решения уравнения

Пример 3.2. Для того, чтобы найти время подъема захвата манипулятора (см. рис. 3.3) на высоту H, необходимо найти численно корень уравнения t, если уравнение имеет вид:

YC(t) = H, где

YC(t) – функция координаты Y захвата манипулятора в зависимости от времени. Предполагается, что аналитический вид функции задан или получен в результате предварительных вычислений.

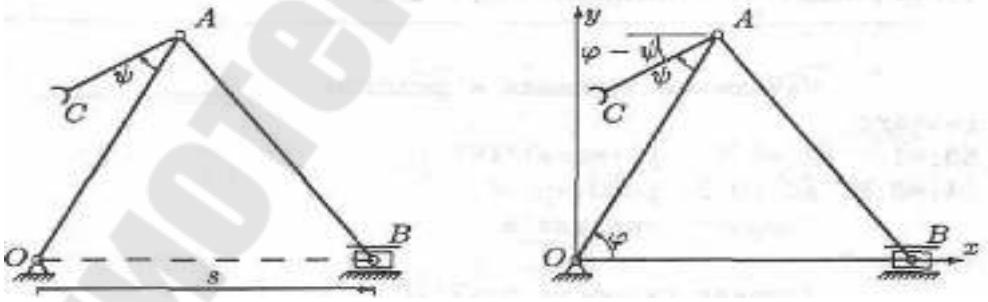


Рис. 3.3. Схема манипулятора

Фрагмент документа решения этой задачи приведен на рис. 3.4. Здесь задается вектор высот подъема захвата манипулятора – H и начальное приближение корня – значение времени t0. Задается функция, описывающая левую часть нелинейного уравнения

YC(t) - H= 0,

у которой два параметра: время подъема t и высота подъема H. При циклическом обращении к функции root происходит формирование вектора T, каждый элемент которого является корнем уравнения при заданном значении высоты. Графическое отображение результатов показывает, что корни найдены верно.

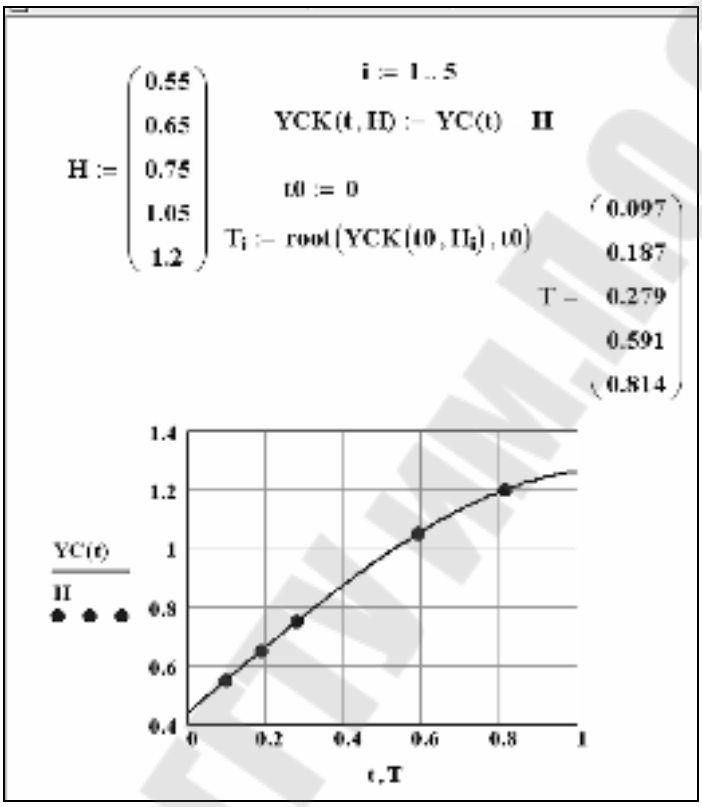


Рис. 3.4. Фрагмент документа решения примера 3.2

3.2. Поиск корней полиномиального уравнения, графическая интерпретация

Для нахождения корней полиномиального уравнения вида

$$v_n \cdot x^n + v_{n-1} \cdot x^{n-1} + \ldots + v_1 \cdot x + v_0 = 0$$

используется функция ***polyroots.***

В отличие от функции ***root, polyroots*** не требует начального приближения и вычисляет сразу все корни, как вещественные, так и комплексные.

Общий вид функции следующий:

polyroots(v),

где ***v*** – вектор коэффициентов полинома длины ***n+1,*** ***n*** – степень полинома. Вектор ***v*** формируется следующим образом: в первый его элемент заносится значение коэффициента полинома при x^0, т.е. v_0, во второй элемент - значение коэффициента полинома при x^1, т.е. v_1 и т.д.

Таким образом, вектор заполняется коэффициентами перед степенями полинома справа налево.

Функция вычисляет вектор длины ***n***, состоящий из корней полинома. На рис. 3.4 приведены примеры вычисления корней уравнений с помощью функции ***polyroots*.**

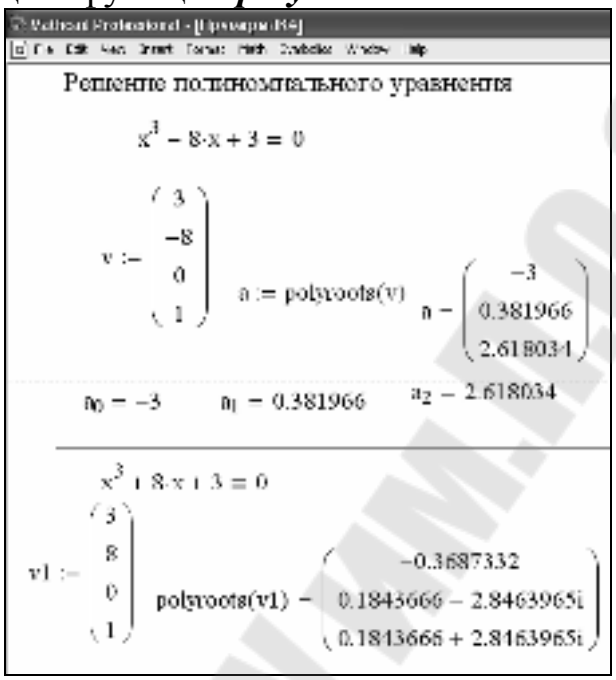


Рис. 3.4. Примеры решения уравнений

Последовательность действий для нахождения корней полиномиального уравнения вида $v_n \cdot x^n + v_{n-1} \cdot x^{n-1} + ... + v_1 \cdot x + v_0 = 0$ с использованием функции ***polyroots*** такова:

- создать вектор длиной (***n*+1**), состоящий из коэффициентов полинома, расположенного в левой части уравнения (первый элемент - значение коэффициента полинома при x^0, второй элемент - значение коэффициента полинома при x^1 и т.д. и присвоить его какой-либо переменной;
- используя имя вектора в качестве аргумента функции ***polyroots***, получить числовые значения корней полиномиального уравнения с помощью оператора «=».

Пример 3.3. Вычислить множество корней полиномиального уравнения $1.2755x^3 - 3.601 \cdot x^2 - 1.37 \cdot x + 6.76 = 0$ с использованием функции ***polyroots***.

Реализация данного примера в MathCad приведена на рис. 3.5.

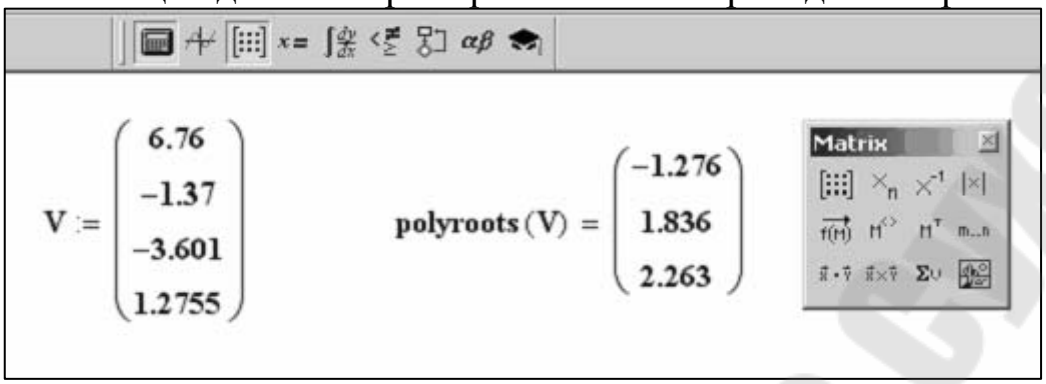


Рис. 3.5. Реализация примера 3.3 в MathCad

3.3. Решение системы линейных уравнений

MathCAD дает возможность решать системы уравнений и неравенств.

Наиболее распространенным методом решения уравнений в Mathcad является блочный метод. Для решения системы этим методом необходимо выполнить следующее:

а) задать начальное приближение для всех неизвестных, входящих в систему уравнений;

б) задать ключевое слово ***Given***, которое указывает, что далее следует система уравнений;

в) ввести уравнения и неравенства в любом порядке (использовать кнопку логического равенства на панели знаков логических операций Boole = для набора знака «=» в уравнении);

г) ввести любое выражение, которое включает функцию ***Find***.

Решающим блоком называется часть документа, расположенная между ключевыми словами ***Given*** и ***Find.***

После набора решающего блока Mathcad возвращает точное решение уравнения или системы уравнений.

Обратиться к функции ***Find*** можно несколькими способами:

Find(x1, x2,…) = - корень или корни уравнения вычисляются и выводятся в окно документа.

x := Find(x1, x2,…) – формируется переменная или вектор, содержащий вычисленные значения корней.

Сообщение об ошибке «*Решение не найдено»* появляется тогда, когда система не имеет решения или для уравнения, которое не имеет вещественных корней, в качестве начального приближения взято вещественное число и наоборот.

Приближенное решение уравнения или системы можно получить с помощью функции ***Minerr*.**

Если в результате поиска не может быть получено дальнейшее уточнение текущего приближения к решению, ***Minerr*** возвращает это приближение. Функция ***Find*** в этом случае возвращает сообщение об ошибке. Правила использования функции ***Minerr*** такие же, как и для функции ***Find***. Часть документа, расположенная между ключевыми словами ***Given*** и ***Minerr*** так же носит название решающего блока.

Примеры решения систем уравнений с помощью решающего блока приведены на рис. 3.6.

Для решения систем линейных уравнений можно использовать общепринятые математические методы: метод Крамера, матричный метод и т.д.

Матричный метод решения системы линейных уравнений реализован в функции ***lsolve.*** Общий вид функции:

lsolve(a, b)

где ***a*** – матрица коэффициентов перед неизвестными, ***b*** – вектор свободных членов.

Матричный метод можно реализовать и с помощью обратной матрицы. Примеры решения систем линейных уравнений с помощью матричного метода приведены на рис. 3.6.

Из рис. 3.6 видно, что при решении системы уравнений блочным методом можно получить численные значения корней системы уравнений, без присваивания и с присваиванием их в переменные x1 и y1. При решении системы уравнений матричным методом продемонстрированы два варианта: с использованием стандартной функции ***lsolve*** и с использованием обратной матрицы.

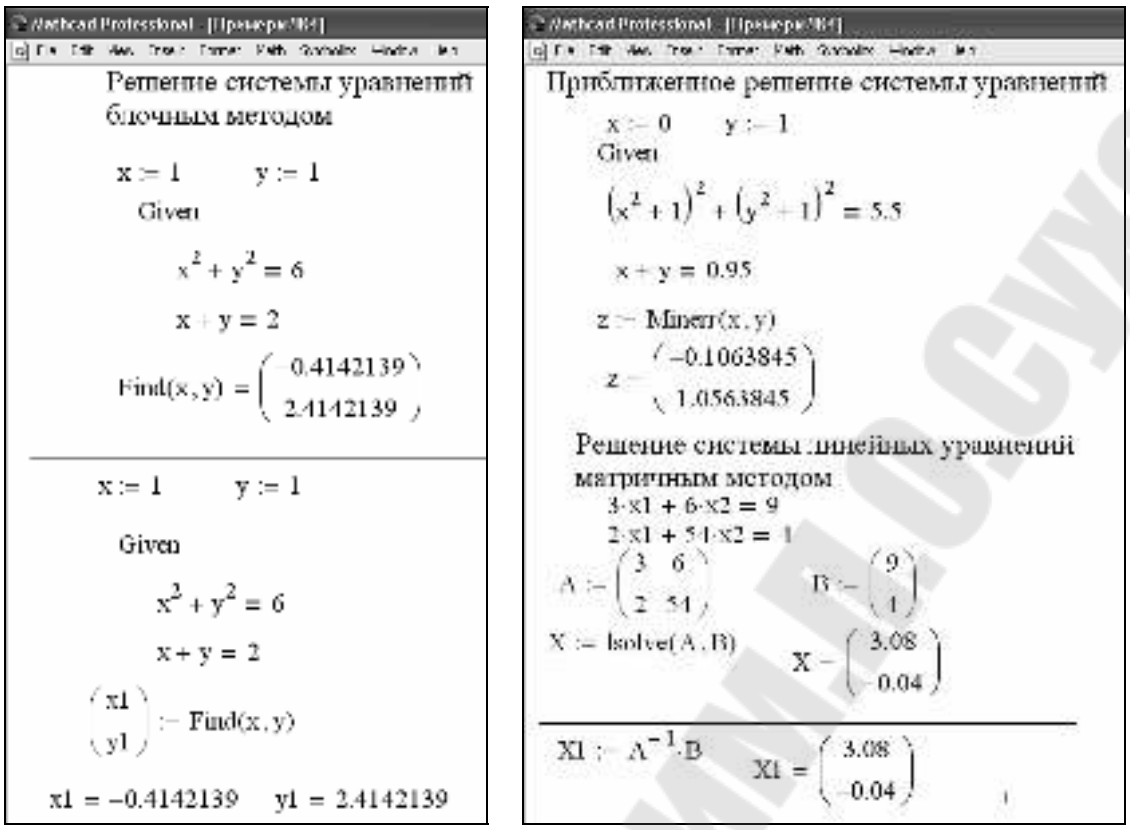


Рис. 3.6. Примеры решения систем уравнений

Ниже рассмотрена пошаговая последовательность действий при решении системы линейных уравнений различными методами.

А) Последовательность действий для решения системы линейных уравнений методом Крамера такова:

- создать матрицу коэффициентов системы линейных уравнений, например, $A := \begin{pmatrix} 2.48 & 1.75 & -3.24 \\ 4.78 & 1.2 & 2.2 \\ -3.24 & 2.19 & -1.86 \end{pmatrix}$;
- создать вектор свободных членов, например $B := \begin{pmatrix} 1.23 \\ 3.55 \\ -0.15 \end{pmatrix}$;
- с помощью оператора «**:=**» создать матрицу, равную матрице коэффициентов, например, $A1 := A$;
- заменить в созданной матрице первый столбец вектором свободных членов, используя операцию выделения столбца матрицы, например, $A1^{<1>} := B$ или $A1^{<0>} := B$ (в зависимости от значения переменной ***ORIGIN***);
- аналогично из матрицы коэффициентов создать матрицу, в которой второй столбец заменен вектором свободных членов, затем матрицу, в которой третий столбец заменен вектором

свободных членов, и т.д. (количество таких матриц определяется количеством неизвестных в системе уравнений);

- найти первый корень, разделив определитель матрицы с замененным первым столбцом на определитель матрицы коэффициентов, например: $\frac{|A1|}{|A|} =$;
- найти остальные корни системы уравнений аналогично.

Б) Последовательность действий для решения системы линейных уравнений матричным методом такова:

- создать матрицу коэффициентов системы линейных уравнений, например, A;
- создать вектор свободных членов системы линейных уравнений, например, B;
- получить решение системы с помощью функции ***lsolve***, параметрами которой являются матрица коэффициентов и вектор свободных членов, например: $X := lsolve(A, B)$

  (решение также можно получить, умножив матрицу, обратную к матрице коэффициентов, на вектор свободных членов: $X := A^{-1} \cdot B$);
- вывести полученный вектор, содержащий корни системы, с помощью оператора «=».

В) Последовательность действий для решения системы линейных уравнений блочным методом такова:

- задать начальные приближения для всех неизвестных, входящих в систему уравнений;
- набрать ключевое слово ***Given***;
- ниже слова ***Given*** набрать уравнения, отделяя правую и левую части символом логического равенства «=»;
- набрать функцию ***Find***, подставляя в качестве аргументов имена неизвестных системы;
- вывести вектор, содержащий вычисленные значения корней, с помощью оператора «=», например ***Find(x1,x2,x3)=***.

Замечание. Корни системы уравнений, полученные разными способами, должны совпасть.

Пример 3.4. Решить систему линейных уравнений

$$\begin{cases} 2.48 \cdot x_1 + 1.75 \cdot x_2 - 3.24 \cdot x_3 = 1.23 \\ 4.78 \cdot x_1 - 1.2 \cdot x_2 + 2.2 \cdot x_3 = 3.55 \\ -3.24 \cdot x_1 + 2.19 \cdot x_2 - 1.86 \cdot x_3 = -0.15 \end{cases}$$

методом Крамера, матричным и блочным методами. Сравнить полученные результаты. Начальные значения корней при использовании блочного метода принять равными 1.

Реализация данного примера в MathCad приведена на рис. 3.7:

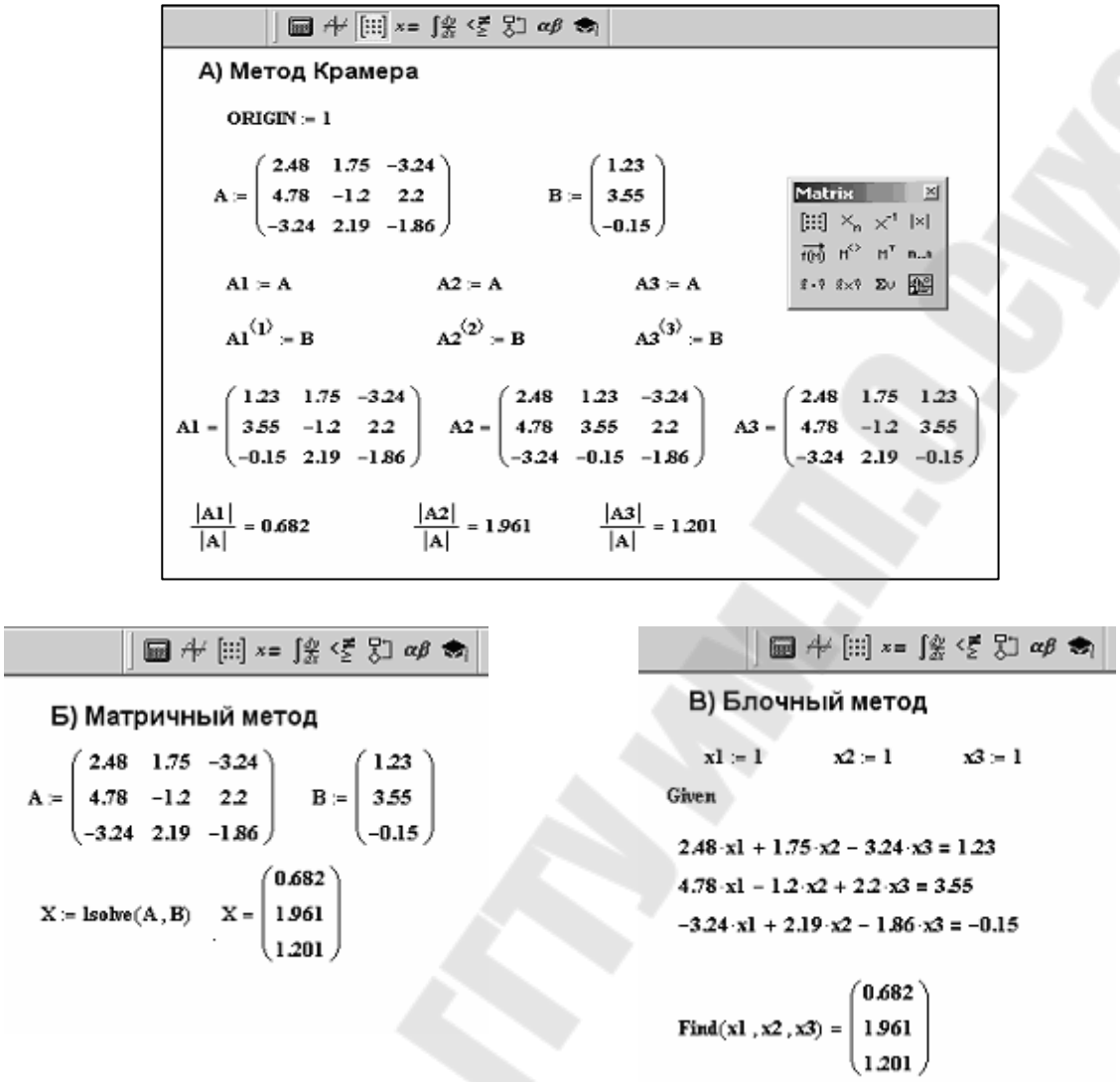


Рис. 3.7. Реализация примера 3.4 в Mathcad

Пример 3.5. Дан кривошипно-ползунный механизм (рис. 3.8), исходными данными для проектирования которого служит функциональная зависимость перемещения ползуна S от угла поворота кривошипа φ. Необходимо определить длины звеньев a_1, a_2 и значение параметра a_3. Таблица значений φ_i и S_i может содержать по три значения, т.е. задаются три положения механизма (i=1,2,3).

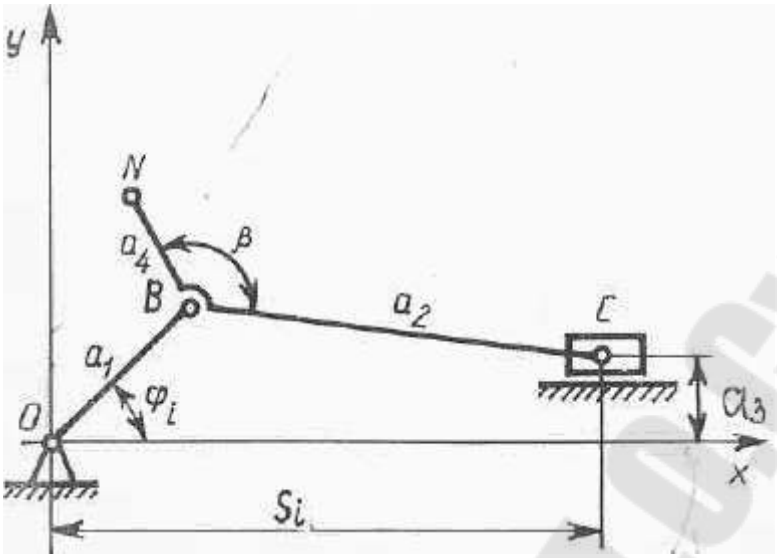


Рис. 3.8. Кинематическая схема кривошипно-ползунного механизма

При i=3 механизм описывается системой уравнений вида:

$$K_1 S_1 \cos\varphi_1 + K_2 \sin\varphi_1 - K_3 = S_1^2$$
$$K_1 S_2 \cos\varphi_2 + K_2 \sin\varphi_2 - K_3 = S_2^2$$
$$K_1 S_3 \cos\varphi_3 + K_2 \sin\varphi_3 - K_3 = S_3^2$$

Длины звеньев вычисляются по формулам:

$$a_2 = \sqrt{a_1^2 + a_3^2 - K_3} \qquad a_1 = \frac{K_1}{2} \qquad a_3 = \frac{K_2}{2a_1}$$

На рис. 3.9 приведен фрагмент документа, реализующий вычисление длин звеньев механизма. Здесь задаются исходные данные, характеризующие три положения механизма – вектора S и φ, причем вектор φ задается в градусах, но переводится в радианы умножением на системную переменную deg. Решение системы линейных уравнений выполняется блочным методом, следовательно, корни системы – переменные k_1, k_2, k_3 – должны получить начальные числовые значения до открытия решающего блока. После описания системы формируется вектор k, в который помещаются вычисленные с помощью стандартной функции Find корни системы. С использование полученных корней вычисляются длины звеньев механизма.

При решении данной задачи в Mathcad можно было использовать матричный метод для поиска корней системы линейных кравнений.

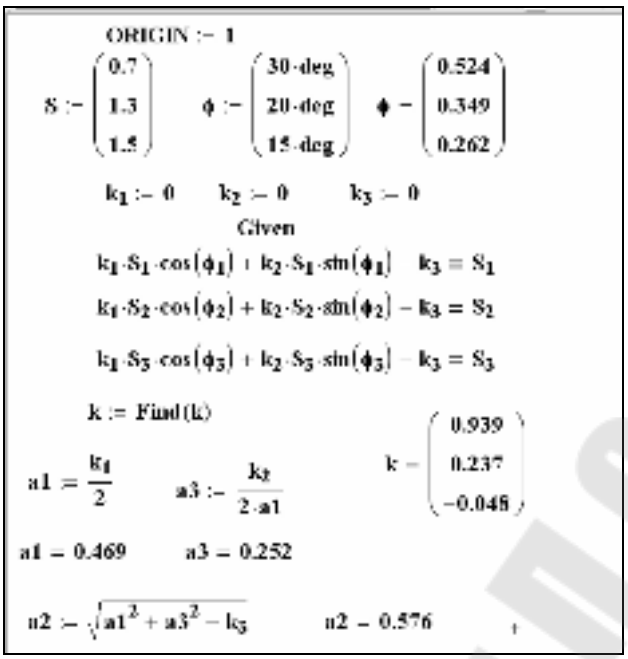


Рис. 3.9. Решение системы уравнений при синезе параметров механизма

3.4. Решение систем нелинейных уравнений

Последовательность действий для решения системы нелинейных уравнений блочным методом такая же, как и для решения систем линейных уравнений, изложенная в пункте 3.3.

Пример 3.6. Решить систему нелинейных уравнений

$$\begin{cases} 2 \cdot x + y + z = 6 \\ x^2 + y^2 + z^2 = 14 \\ x^3 - y^3 + z^3 = 36 \end{cases}$$

блочным методом. Начальные значения корней принять равными 1. Реализация данного примера в MathCad приведена на рис. 3.10.

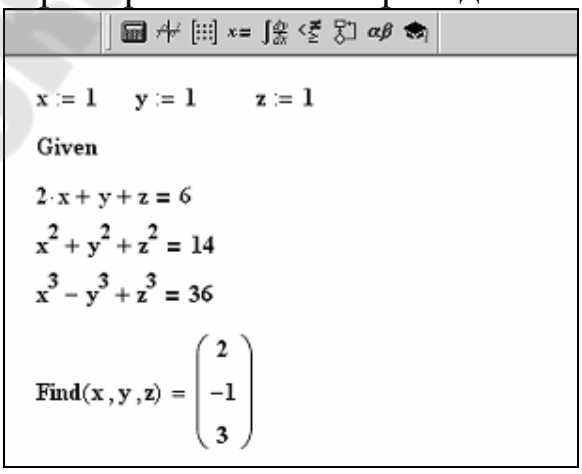


Рис. 3.10. Решение системы нелинейных уравнений

Пример 3.6. Точка движется по плоской кривой ***y(x)*** с постоянной скоростью ***v*** (рис. 3.11). Для нахождения проекций скорости

движения точки на оси X и Y (***vx, vy***) нужно при заданном значении *x* решить систему уравнений вида:

$vy = y1(x) \cdot vx$

$vx^2 + vy^2 = v^2$, где

$y1(x) = \frac{d}{dx} y(x)$ - первая производная по координате *x*, y(x) – заданный закон движения точки.

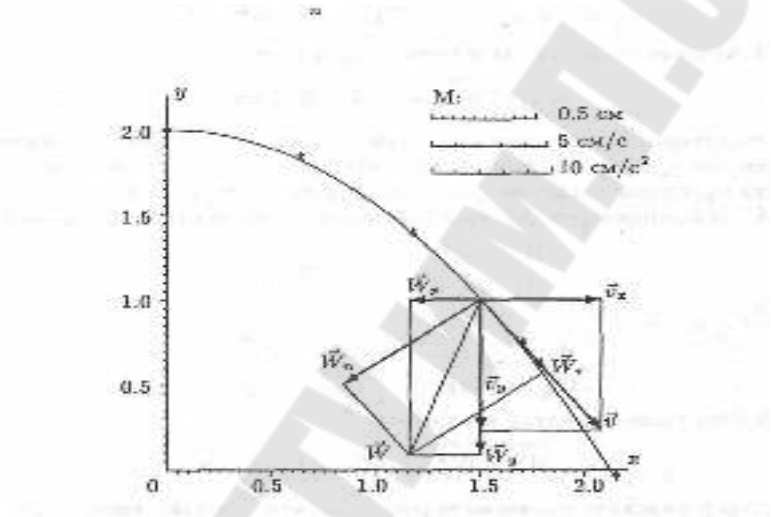


Рис. 3.11. График движения материальной точки

На рис. 3.12 приведен фрагмент документа Mathcad, реализующего решение данной задачи. Предполагается, что аналитический вид функции y(x) задан или получен в результате предварительных вычислений. В документе формируется новая функция y1(x) как производная от функции y(x). Задаются исходные данные: значения скорости и координата x точки.

Для решения системы нелинейных уравнений используется метод решающего блока: переменным vx, vy присваиваются начальные числовые значения, организуется решающий блок – Given, описывается вид системы нелинейных уравнений, формируется вектор корней системы, которые принимают численные значения после применения функции Find.

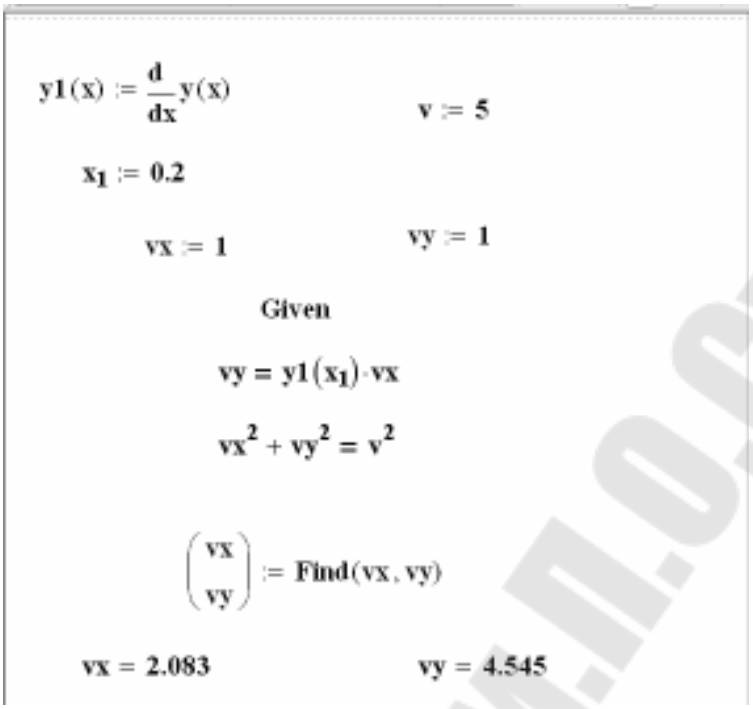


Рис. 3.12. Решение системы уравнений при нахождении проекций скорости материальной точки

4. Решение дифференциальных уравнений и систем в MathCad

4.1. Решение дифференциальных уравнений первого порядка

Для решения дифференциальных уравнений с начальными условиями система ***Mathcad*** имеет ряд встроенных функций:

rkfixed – функция для решения ОДУ и систем ОДУ методом Рунге–Кутта четвертого порядка с постоянным шагом;

Rkadapt – функция решения ОДУ и систем ОДУ методом Рунге–Кутта с переменным шагом;

Odesolve – функция, решающая ОДУ блочным методом.

Ниже приведено описание стандартной функции ***rkfixed*** с указанием параметров функции.

rkfixed(y, x1, x2, p, D)

Аргументы функции:

y – вектор начальных условий из ***k*** элементов (k – количество уравнений в системе);

x1 и ***x2*** – левая и правая границы интервала, на котором ищется решение ОДУ или системы ОДУ;

p – число точек внутри интервала ***(x1, x2)***, в которых ищется решение;

D – вектор, состоящий из ***k-***элементов, который содержит первую производную искомой функции или первые производные искомых функций, если речь идет о решении системы.

Результатом работы функции является матрица из p+1 строк, первый столбец которой содержит точки, в которых получено решение, а остальные столбцы – сами решения.

Стандартная функция ***Odesolve*** имеет следующий общий вид:

Odesolve(x, n) или

Odesolve(v, x, n)

Аргументы функции следующие:

- n – правая граница интервала, на котором ищется решение дифференциального уравнения;
- x – переменная, относительно которой ищется решение ОДУ;
- v – вектор имен функций при решении систем дифференциальных уравнений.

Левая граница интервала определяется из начальных условий.

В результате работы Odesolve формируется функция, являющаяся решением дифференциального уравнения относительно аргумента x. При решении ОДУ и систем ОДУ с помощью функции Odesolve необходимо организовать решающий блок по аналогии с решающим блоком для решения систем алгебраических уравнений.

Последовательность решения такова:

- задается ключевое слово Given (начало решающего блока);
- задается вид дифференциального уравнения, причем знак производной можно задать с помощью клавиш Ctrl-F7, а знак равенства – с помощью кнопки логического равенства на палитре знаков логических операций;
- задаются начальные условия, для чего используется знак логического равенства;
- задается функция Odesolve с параметрами, формируется результирующая функция.

Все вычислительные блоки, расположенные между зарезервированными словами Given и Odesolve, называются решающим блоком для решения дифференциальных уравнений и систем.

На рис. 4.1 приведен конкретный пример решения дифференциального уравнения первого порядка в MathCAD с помощью различных стандартных функций. Ниже приведено краткое описание этого примера.

При решении дифференциального уравнения первого порядка нужно создать вектор начальных условий из одного элемента Y_1, который затем используется при формировании вектора-функции правой части дифференциального уравнения. При обращении к функции ***rkfixed*** указывается имя вектора ***Y,*** границы интервала, на котором ищется решение уравнения, например, (0 ; 2), количество

точек, в которых ищется решение – 100, вектор-функция, описывающая правую часть дифференциального уравнения – ***D***.

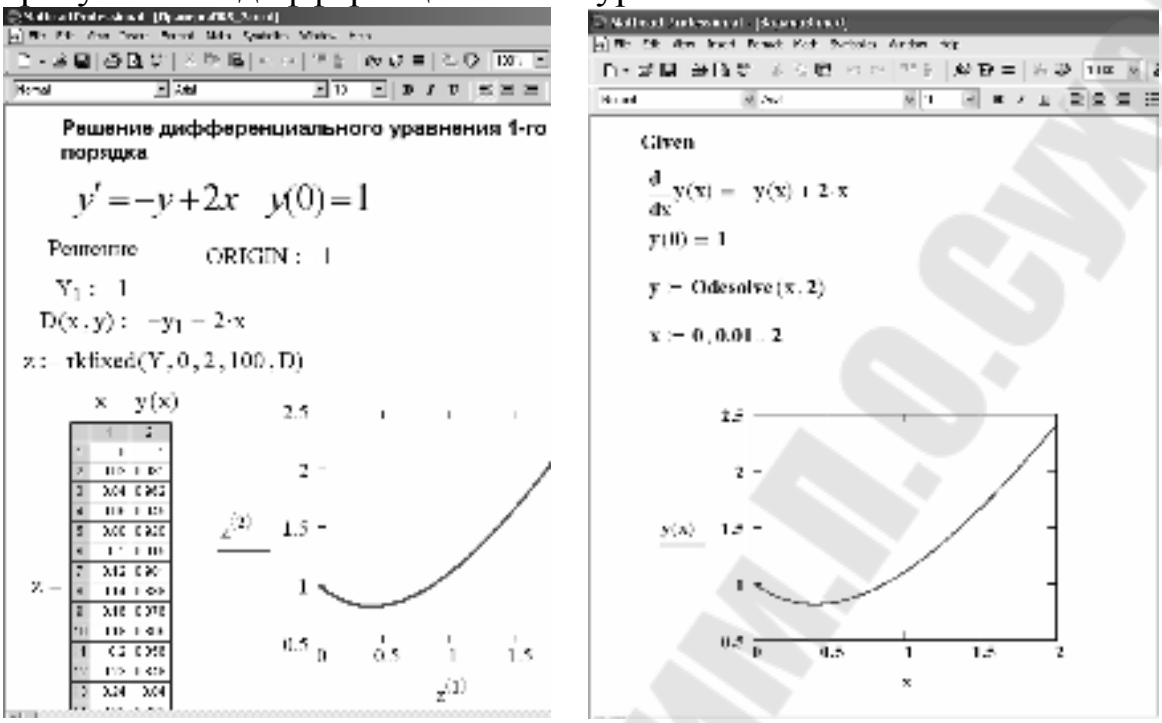


Рис. 4.1. Пример решения дифференциального уравнения

В результате получается матрица ***z,*** в первом столбце которой содержатся значения аргумента искомой функции, во втором – значения самой результирующей функции. При построении графика функции первый столбец полученной матрицы указывается как аргумент, второй столбец – как функция.

При решении с помощью функции Odesolve создается решающий блок, в котором описывается само уравнение и начальное условие для его решения в естественном математическом виде. Затем формируется результирующая функция y (она задается только своим именем) путем обращения к функции Odesolve с параметрами x – переменная, относительно которой ищется решение ОДУ, 2 – конечное значение интервала изменения этой переменной. Результирующее значение функции y(x) представлено в виде графика, причем x необходимо описать как дискретную переменную.

Последовательность действий для решения дифференциального уравнения первого порядка такова:

- сформировать вектор начальных условий из одного элемента, присвоив начальное значение искомой функции переменной с индексом, например: $Y_1 :=$ или $Y_0 :=$ (в зависимости от значения переменной ***ORIGIN***);
- определить вектор-функцию из одного элемента, которая содержит первую производную неизвестной функции:

- набрать имя функции с двумя параметрами: первый параметр – аргумент искомой функции (независимая переменная), второй – имя вектора, содержащего искомую функцию (можно использовать имя вектора начальных условий), например, $D(x,Y)$;
- набрать оператор «**:=**» и выражение для первой производной (выразить из дифференциального уравнения), в котором вместо имени искомой функции подставлен первый элемент вектора-параметра, например, для уравнения $y'(3 \cdot x - y) = y$ вектор-функция будет определятся следующим образом: $D(x,Y) := \frac{Y_1}{3 \cdot x - Y_1}$ (если ***ORIGIN*=0**, подставлять Y_0);

- присвоить некоторой переменной значение функции ***rkfixed***, указав в скобках следующие параметры:
  - первый – имя вектора начальных условий,
  - второй – левая граница интервала, на котором ищется решение, в виде числовой константы,
  - третий – правая граница интервала, на котором ищется решение, в виде числовой константы,
  - четвертый – количество точек, в которых ищется решение,
  - пятый – имя вектора-функции, описывающего первую производную, без параметров;

  например: $Z := rkfixed(Y,0.2,5,1000,D)$,

  (в результате получится матрица Z, в первом столбце которой содержатся значения аргумента искомой функции, во втором – значения самой функции);
- вывести матрицу, содержащую решение ДУ с помощь оператора «=», например: $Z =$;
- построить график найденной функции, указав в качестве аргумента по оси абсцисс столбец $Z^{<1>}$, а в качестве значения функции по оси ординат – столбец $Z^{<2>}$ (если ***ORIGIN*=0**, набирать соответственно $Z^{<0>}$ и $Z^{<1>}$).

Пример 4.1 Найти численное решение дифференциального уравнения первого порядка $y'(3 \cdot x - y) = y$ на интервале от 0.2 до 5 в 1000 точках, при начальном условии y(0)=0.1. Выполнить графическую интерпретацию результатов.

Решение данного примера в MathCad приведено на рис. 4.2.

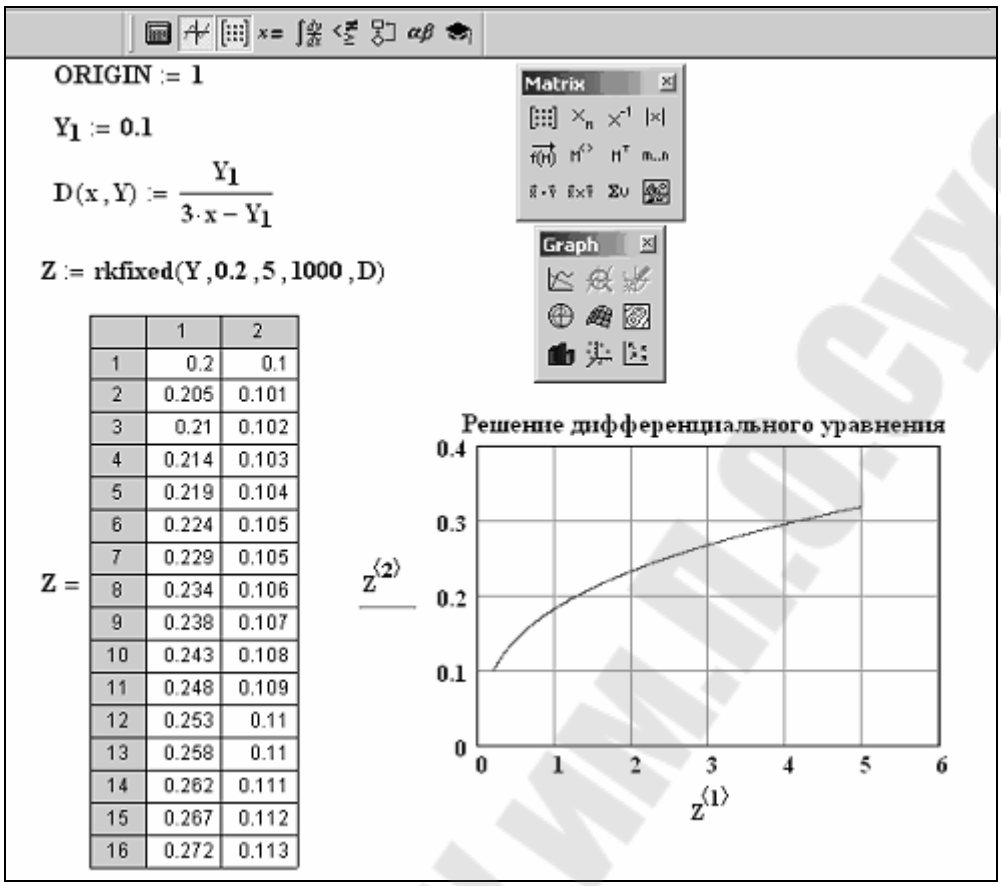


Рис. 4.2. Решение дифференциального уравнения первого порядка

Пример 4.2 Электрическая цепь, представляющая собой параллельно соединенные источник тока, переменное сопротивление и емкость (рис. 4.3), описывается дифференциальным уравнением вида:

$$C\frac{du_c}{dt} + \frac{u_c}{R(t)} = J$$

Переменное сопротивление задано функцией вида:

$$\mathrm{R(t)} = \begin{cases} 2 & \textit{если} \quad 0 \le \mathrm{t} < \tau \\ 1 & \text{если} \quad t \ge \tau \end{cases}$$

Необходимо найти значения напряжения на конденсаторе в заданном временном интервале 2 ***τ.***

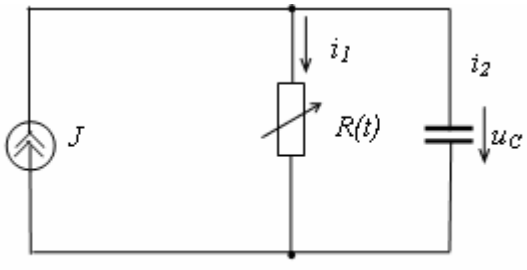


Рис. 4.3. Вид электрической цепи для примера 4.2

Исходными данными для решения задачи являются следующие:

J=10 – сила тока в источнике;
C=1 – исходная емкость конденсатора;
$\boldsymbol{uc}_1$ =11.87– начальное значение напряжения на конденсаторе;
τ=2 - время, равное полупериоду изменения ***R(t).***

На рис. 4.4 приведен документ Mathcad для решения этой задачи. Здесь задаются исходные данные: числовые значения для J, C и ***τ***, затем создается программный фрагмент, описывающий функцию переменного сопротивления R(t).

Для решения дифференциального уравнения его нужно привести к форме Коши, когда первая производная находится в левой части уравнения.

$$\frac{du_c}{dt} = \frac{J - \dfrac{u_c}{R(t)}}{C}$$

Вектор начальных условий $\boldsymbol{uc}_1$ состоит из одного элемента - значения начального напряжения на конденсаторе. Вектор-функция z(t,uc) описывает правую часть решаемого дифференциального уравнения. После обращения к функции rkfixed формируется результирующая функция зависимости напряжения на конденсаторе от времени, представленная в табличном виде (матрица s). Первый ее столбец содержит дискретные значения времени, а второй – значения напряжения на конденсаторе в эти моменты времени.

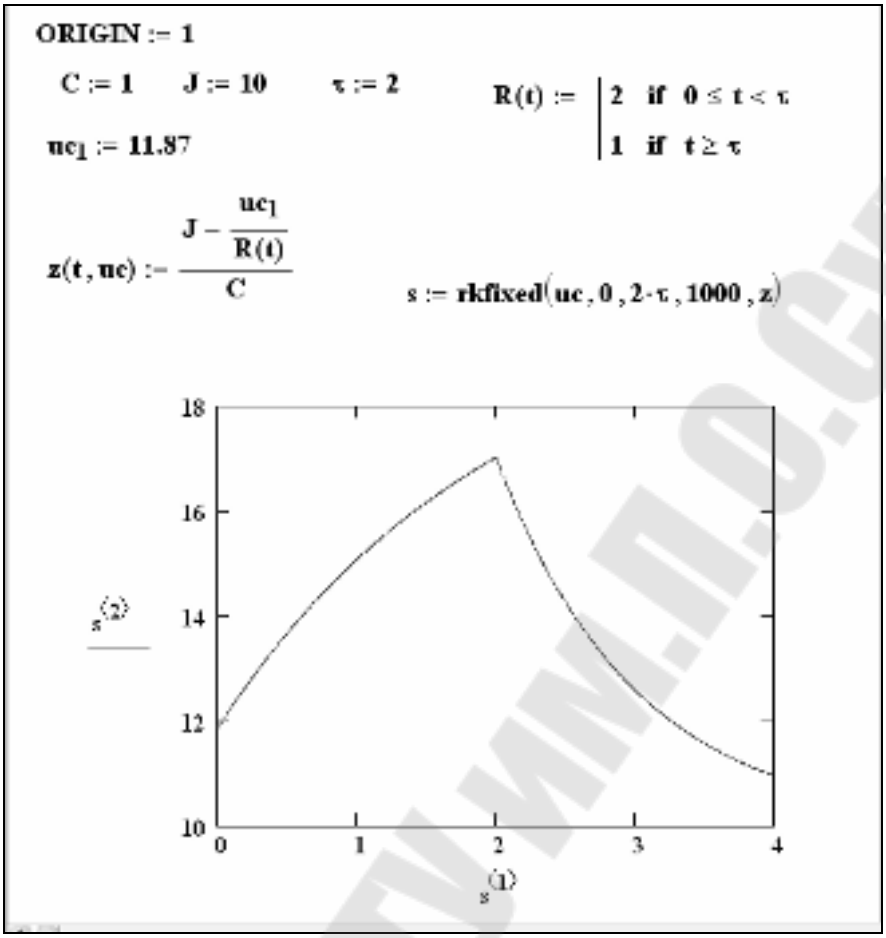


Рис. 4.4. Расчет электрической цепи

4.2 Решение систем дифференциальных уравнений

Системы дифференциальных уравнений можно решать как с помощью функции ***rkfixed*** так и с помощью ***Odesolve.*** При решении системы дифференциальных уравнений с помощью функции ***rkfixed*** нужно создать вектор начальных условий из двух элементов, например, вектор ***v***, который затем используется при формировании вектора-функции правой части дифференциального уравнения. При обращении к функции ***rkfixed*** указывается имя вектора ***v,*** и границы интервала, на котором ищется решение уравнения, например, (0 ; 5), количество точек, в которых ищется решение – 100, вектор-функция, описывающая правую часть дифференциального уравнения – ***D***. В результате получается матрица ***s,*** в первом столбце которой содержатся значения аргумента искомых функций, во втором и третьем столбцах – значения самих функций при соответствующем значении аргумента. При построении графика можно воспользоваться первым столбцом

полученной матрицы как аргументом, а вторым и третьим столбцами – как функциями (рис. 4.5) .

При решении с помощью функции ***Odesolve*** нужно создать решающий блок, в котором в естественном математическом виде описать систему дифференциальных уравнений (при вводе знака производной использовать клавиши Ctrl-F7) и начальные условия. Далее нужно организовать вектор из двух элементов, в который занести имена результирующих функций – решений системы. В качестве первого аргумента функции ***Odesolve*** также используется вектор, в котором содержатся имена тех функций, которые использовались в решающем блоке. Результатами расчетов являются функции x(t) и y(t). После задания дискретной переменной t можно получить графики этих функций (рис. 4.5).

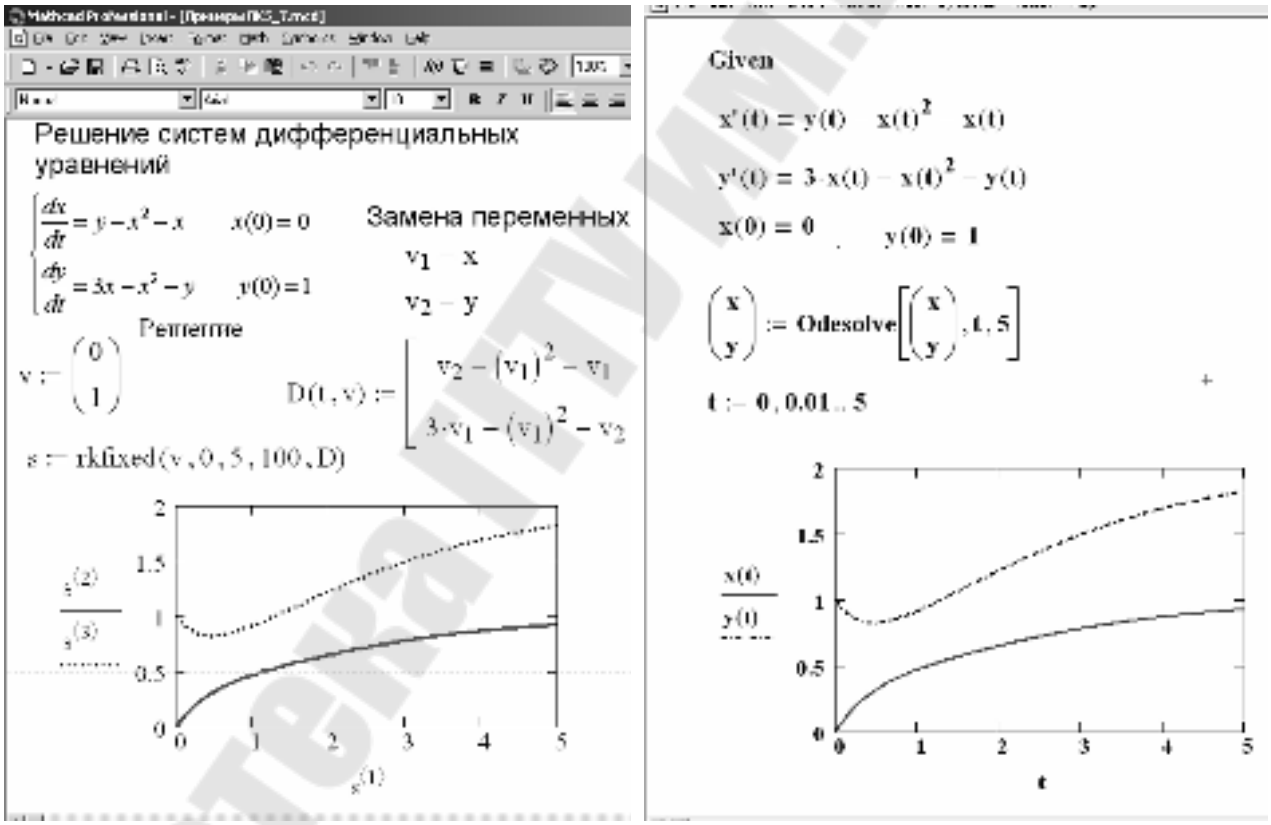


Рис. 4.5 – Пример решения системы дифференциальных уравнений

Последовательность действий для решения системы дифференциальных уравнений первого порядка такова (описана для значения ***ORIGIN*=0**):

- перейти в исходной системе уравнений к однотипным обозначениям функций и выразить первые производные,

например, систему $\begin{cases} x' + 2 \cdot y = 3 \cdot x \\ y't = 2 \cdot x + 8 \cdot y \end{cases}$ можно преобразовать в

$$\begin{cases} V_1' = 3 \cdot V_1 - 2 \cdot V_2 \\ V_2' = \dfrac{2 \cdot V_1 + 8 \cdot V_2}{t} \end{cases};$$

- в документе MathCad сформировать вектор начальных условий, количество элементов которого равно количеству уравнений системы, присвоив его некоторой переменной;

  например, $V := \begin{pmatrix} 0.1 \\ 1 \end{pmatrix}$;

- определить вектор-функцию, которая содержит первые производные искомых функций:
  - набрать имя функции с двумя параметрами: первый параметр – аргумент искомых функций (независимая переменная), второй – имя вектора, содержащего искомые функции (можно использовать имя вектора начальных условий), например, $D(t,V)$;

  (*Замечание*: е*сли независимая переменная явно не присутствует в системе, то в качестве ее имени можно выбрать любую переменную*)
  - набрать оператор «**:=**» и вставить шаблон вектора, количество элементов которого равно количеству уравнений системы ;
  - набрать в качестве элементов вектора правые части системы уравнений, в которых искомые функции представлены соответствующими элементами вектора-параметра, например,

    $$D(t,V) := \begin{pmatrix} 3 \cdot V_1 - 2 \cdot V_2 \\ \dfrac{2 \cdot V_1 + 8 \cdot V_2}{t} \end{pmatrix};$$

- присвоить некоторой переменной значение функции ***rkfixed***, указав в скобках следующие параметры:
  - первый – имя вектора начальных условий,
  - второй – левая граница интервала, на котором ищется решение, в виде числовой константы,
  - третий – правая граница интервала, на котором ищется решение, в виде числовой константы,
  - четвертый – количество точек, в которых ищется решение,
  - пятый – имя вектора-функции, описывающего первые производные, без параметров;

например: $Z := rkfixed(V,0,0.5,1000,D)$,

(в результате получится матрица Z, в первом столбце которой содержатся значения аргумента искомых функций, во втором – значения первой функции, в третьем – значения второй функции и т.д.);

- вывести матрицу, содержащую решение системы ДУ с помощь оператора «=», например: $Z =$;
- построить графики найденных функций, указав в качестве аргумента по оси абсцисс первый столбец матрицы решений, например, $Z^{<1>}$, а в качестве значений функций по оси ординат – остальные столбцы матрицы через запятую, например, $Z^{<2>}$, $Z^{<3>}$ и т.д.

Пример 4.3 Найти решение системы дифференциальных уравнений

$$\begin{cases} x' = 3 \cdot x - 2 \cdot y \\ y' = 2 \cdot x + 8 \cdot y \end{cases}$$

на интервале от 0 до 0.5 в 1000 точках, при следующих начальных условиях: x(0)=0.1 и y(0)=1. Выполнить графическую интерпретацию результатов.

Реализация примера в MathCad приведена на рис. 4.6.

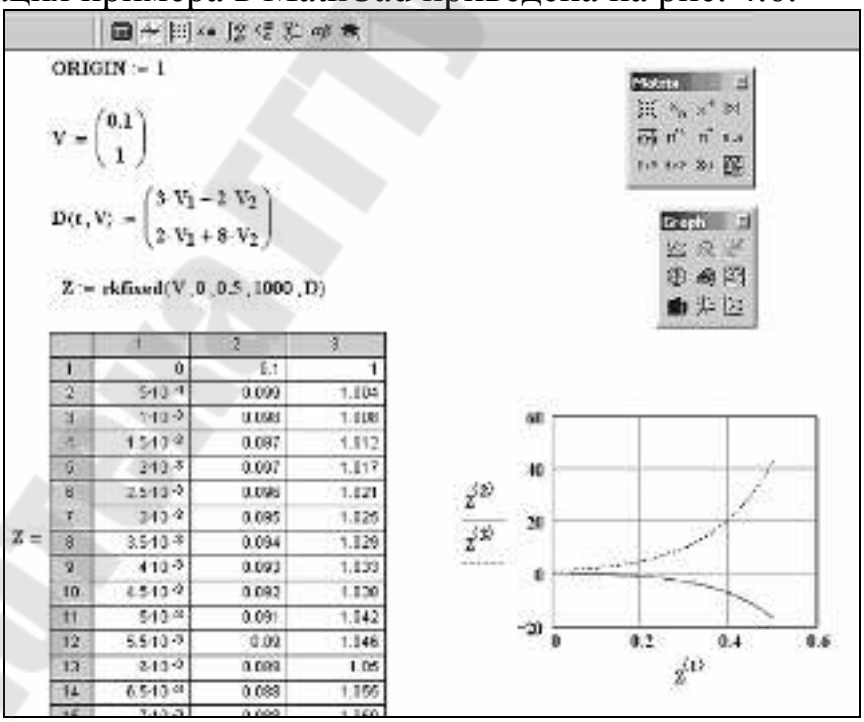


Рис. 4.6. Решение дифференциального уравнения первого порядка

Пример 4.4 Электрическая цепь, приведенная на рис. 4.7, описывается системой дифференциальных уравнений вида:

$$\frac{di}{dt} = \frac{E - i \cdot R - u}{L}$$

$$\frac{du}{dt} = \frac{i - I(u)}{C}$$

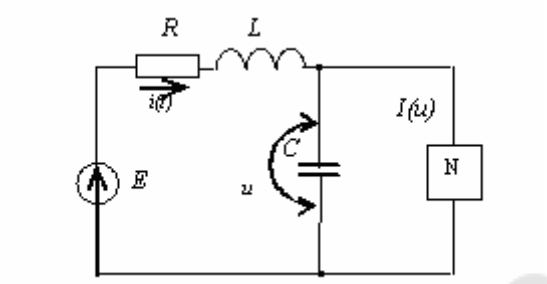


Рис. 4.7 – Вид электрической цепи с туннельным диодом

Вольт-амперная характеристика туннельного диода имеет вид:

$$I(u) = A \cdot u \cdot e^{-\alpha \cdot u} + D \cdot (e^{\beta \cdot u} - 1)$$

Параметры характеристики вычисляются по формулам:

$$A = e \cdot \frac{Ip}{U1} \qquad \alpha = \frac{1}{U1}$$

Исходными данными для решения задачи являются следующие:
Ip,U1, D, β – параметры вольт-амперной характеристики диода;
E – ЕДС;
R – исходное сопротивление;
C – исходная емкость;
L – исходная индуктивность;
u_0 – начальное значение напряжения;
i_0 – начальное значение тока;
T – время исследования;

При решении системы дифференциальных уравнений принять i_0 и u_0 (начальные значения тока и напряжения) равными 0.

Ниже приведен алгоритм выполнения расчетов в системе Mathcad

.

1. Задать исходные данные для решения задачи.
2. Вычислить значения A и α.
3. Создать вектор ***y*** начальных условий и вектор-функцию ***z(t,y),*** описывающую правые части системы дифференциальных уравнений.
4. Решить систему дифференциальных уравнений с использованием функции ***rkfixed*** системы Mathcad. Получить матрицу ***q,*** первый столбец которой, содержит дискретные значения времени расчета, второй столбец – значения тока в эти моменты времени, третий столбец – значения напряжения.

5. Построить график функции тока в цепи в зависимости от времени.

На рис. 4.8. приведен документ Mathcad, реализующий этот алгоритм.

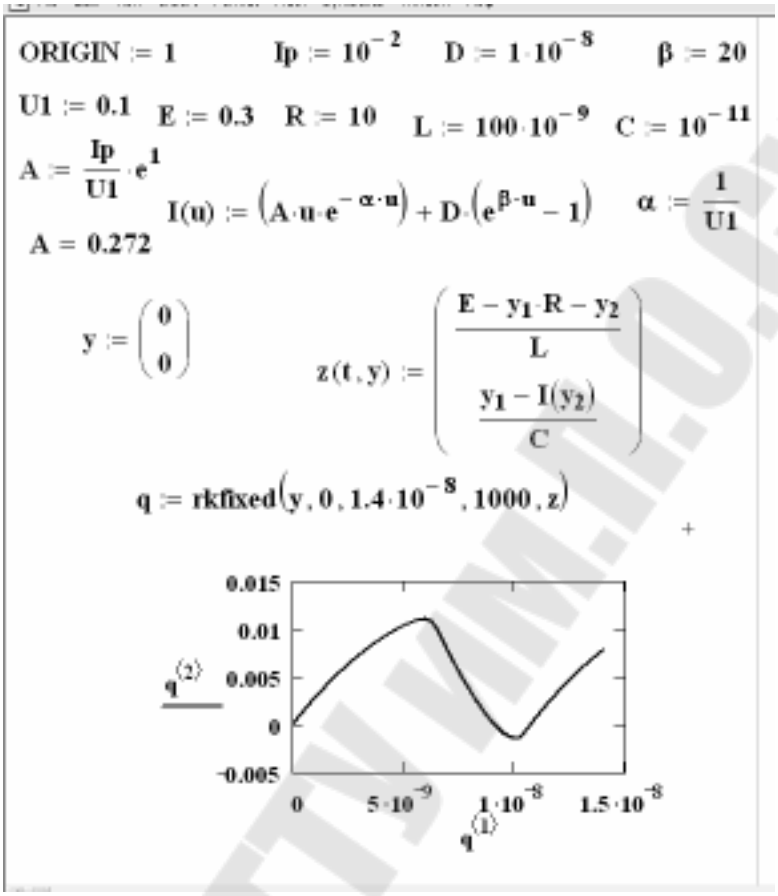


Рис. 4.8. Расчет цепи с туннельным диодом

4.3. Решение дифференциальных уравнений второго порядка

Для решения дифференциальных уравнений второго порядка проиеняются те же стандартные функции и алгоритмы их реализации, что и для решения ОДУ и систем ОДУ. Например, необходимо решить дифференциальное уравнение второго порядка с заданными начальными условиями вида:

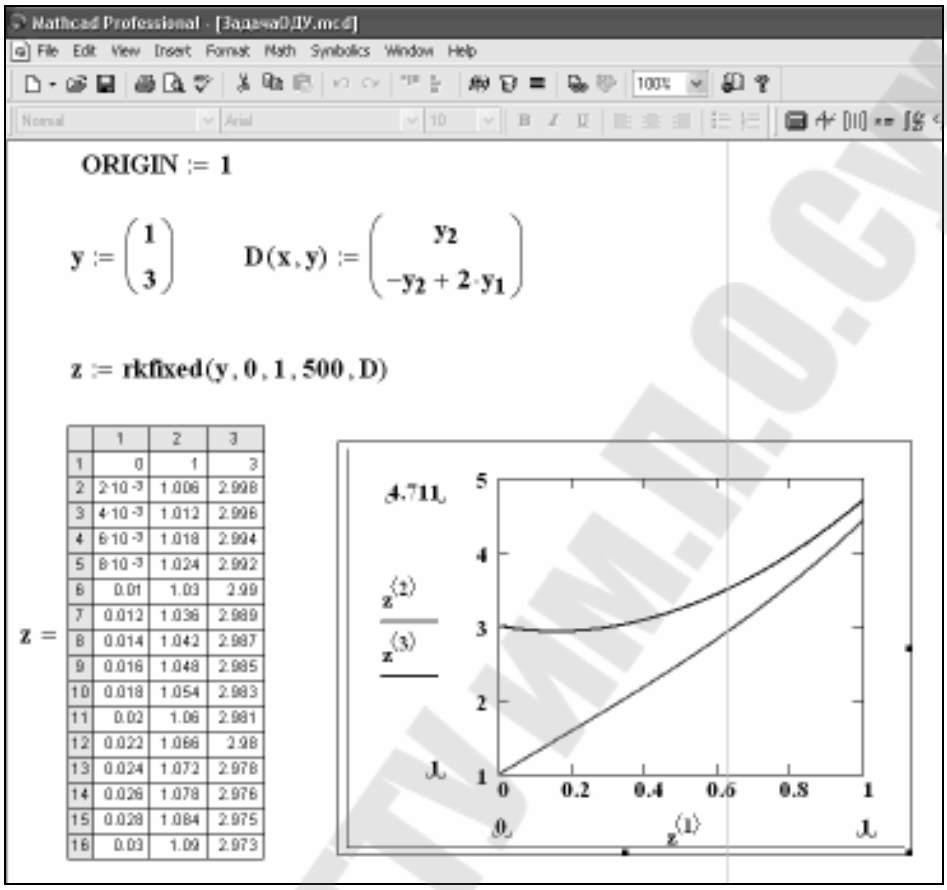
$$y'' = -y' + 2y \quad y(0) = 1 \quad y'(0) = 3$$

Для решения уравнения с помощью функции ***rkfixed*** нужно выполнить замену переменных и привести дифференциальное уравнение второго порядка к двум дифференциальным уравнениям первого порядка. Вид этих уравнений приведен ниже.

$$y - y_1 \qquad \frac{d}{dx} y_1 = y_2$$

$$\frac{d}{dx} y_2 = -y_2 + 2 \cdot y_1$$

На рис. 4.9 приведен пример решения дифференциального уравнения второго порядка с использованием функции ***rkfixed***.



	1	2	3
1	0	1	3
2	$2\cdot10^{-3}$	1.006	2.998
3	$4\cdot10^{-3}$	1.012	2.996
4	$6\cdot10^{-3}$	1.018	2.994
5	$8\cdot10^{-3}$	1.024	2.992
6	0.01	1.03	2.99
7	0.012	1.036	2.989
8	0.014	1.042	2.987
9	0.016	1.048	2.985
10	0.018	1.054	2.983
11	0.02	1.06	2.981
12	0.022	1.066	2.98
13	0.024	1.072	2.978
14	0.026	1.078	2.976
15	0.028	1.084	2.975
16	0.03	1.09	2.973

Рис. 4.9. Пример решения дифференциальных уравнений второго порядка с помощью ***rkfixed***

Документ формируется точно так же, как и при решении системы ОДУ.

На рис. 4.10 показана возможность решения дифференциального уравнения с помощью стандартной функции Odesolve методом решающего блока.

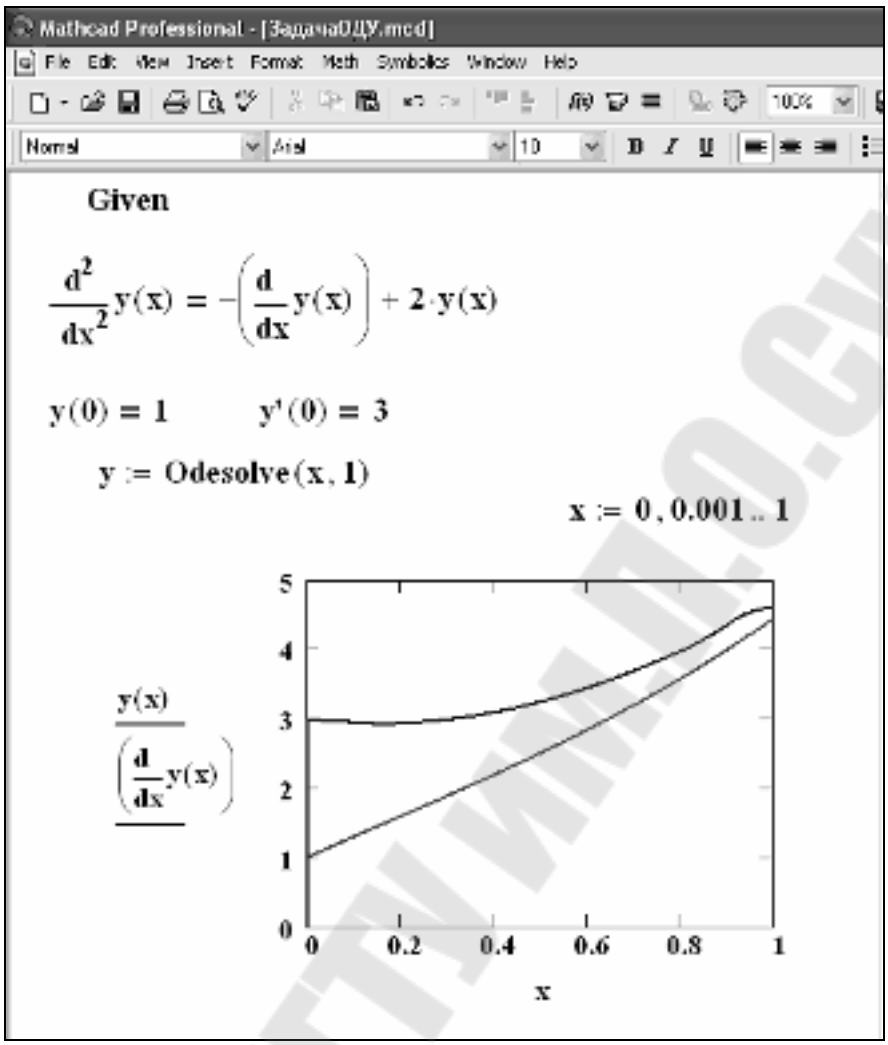


Рис. 4.10. Решение ОДУ второго порядка методом решающего блока

Пример 4.5 Дан гидравлический демпфер, представляющий собой поршень массой m, движущийся в жидкости (рис. 4.11).

Дифференциальное уравнение движения поршня имеет вид:

$$\ddot{y} + 2n\dot{y} + py = 0 ,$$

где n - приведенный коэффициент вязкого сопротивления вычисляется по формуле:

$$n = \left[4\pi\mu H / (mZ)\right] \cdot (D/d)^4 ,$$

p – частота собственных колебаний системы вычисляется по формуле:

$$p = c / m .$$

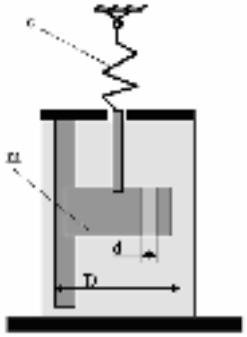


Рис. 4.11. Гидравлический демпфер

Необходимо найти функции перемещения и скорости демпфера в зависимости от времени.

Исходными данными для решения задачи являются следующие:

y0 –отклонение поршня демпфера от положения равновесия;

m – масса поршня;

c – жесткость пружин;

μ – динамический коэффициент вязкости жидкости;

D – диаметр цилиндра;

d – диаметр отверстия;

H – высота поршня;

z –число отверстий.

Для решения дифференциального уравнения в Mathcad с помощью функции rkfixed, его нужно привести к системе из двух дифференциальных уравнений вида:

$$y_1' = y_2$$

$$y_2' = -2ny_2 - py_1$$

где y_1 – перемещение демпфера; y_2 – скорость демпфера.

Ниже приведен алгоритм выполнения расчетов в системе Mathcad

.

6. Задать исходные данные для решения задачи.
7. Вычислить значения ***n*** и ***p***.
8. Создать вектор ***y*** начальных условий и вектор-функцию ***S(t,y),*** описывающую правые части системы дифференциальных уравнений.
9. Решить систему дифференциальных уравнений с использованием функции ***rkfixed*** системы Mathcad. Получить матрицу R, первый столбец которой, содержит дискретные значения времени расчета, второй столбец – значения перемещения, третий столбец – значения скорости.
10. Построить график функции перемещения системы в зависимости от времени.
11. Построить график функции скорости системы в зависимости от времени.

На рис. 4.12 приведены фрагменты документа на MathCAD, реализующего данный алгоритм.

Моделирование
гидравлического демпфера

1. Исходные данные

$m := 2.73 \quad H := 0.05 \quad c := 3 \cdot 10^5 \quad D := 0.1$

$d := 0.01 \quad z := 25 \quad \mu := 6 \cdot 10^{-2}$

2. Расчет приведенного коэффициента вязкого сопротивления и частоты собственных колебаний демпфера

$$n := 4 \cdot \pi \cdot \mu \cdot \frac{H}{m \cdot z} \left(\frac{D}{d} \right)^4 \qquad n = 5.524$$

$$p := \sqrt{\frac{c}{m}} \qquad p = 33.15$$

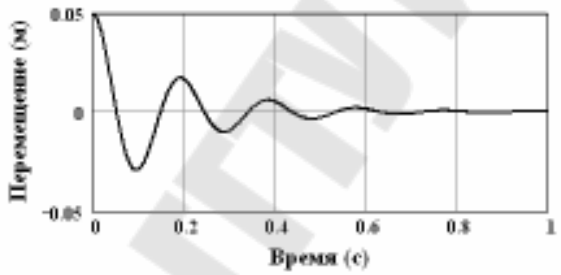
3. Вектор начальных условий и вектор-функция правых частей системы дифференциальных уравнений

$$y := \begin{pmatrix} 0.05 \\ 0 \end{pmatrix} \qquad S(t, y) := \begin{pmatrix} y_1 \\ -2 \cdot n \cdot y_1 - p^2 \cdot y_0 \end{pmatrix}$$

4. Решение системы ОДУ методом Рунге-Кутта

$R := \mathbf{rkfixed}(y, 0, 1, 1000, S)$

5. Графики функций перемещения и скорости демпфера



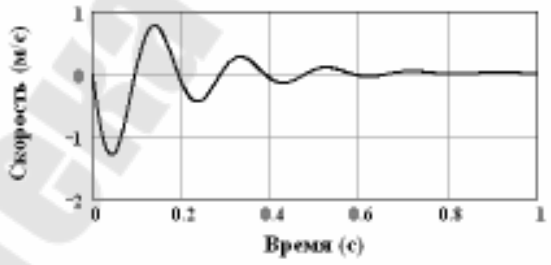


Рис. 4.12. Решение ОДУ второго порядка при расчете гидравлического демпфера

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Трохова Т.А. Практическое пособие по теме «Основные приемы работы в системе MathCad, версии 6.0» курса «ВТ и программирование» для студентов всех специальностей дневного и заочного отделений. – Гомель: ГГТУ, 1998. – 42с.
2. Новиков А.А. Практическое пособие к лабораторным и контрольным работам по теме «Решение инженерно-экономических задач в среде MathCad for Windows» курса «Информатика» для студентов заочного отделения. – Гомель: ГГТУ им.П.О.Сухого, 2000. – 46с.
3. Грудецкий Г.А., Мурашко И.А. Графические средства пакета MathCad: Практическое пособие для студентов всех специальностей дневного и заочного отделений. – Гомель: ГГТУ им.П.О.Сухого, 2001. – 36с.
4. Математический пакет MathCad: Практикум по курсу «Информатика» к лабораторным работам для студентов всех специальностей заочного отделения / Грудецкий Г.А., Коробейникова Е.В., Самовендюк Н.В., Трохова Т.А., Токочаков В.И. – Гомель: ГГТУ им П.О.Сухого,2003. – 49с.
5. Токочаков В.И. Практическое пособие по теме «Решение систем алгебраических и дифференциальных уравнений в среде MathCAD Windows» для студентов всех специальностей дневного и заочного отделений. – Гомель: ГГТУ им П.О.Сухого,2000. – 26с.
6. Дьяконов В.П. Справочник по MathCad Plus 7.0 PRO. – М.: СК Пресс, 1997. – 352с.

СОДЕРЖАНИЕ

ПРИМЕНЕНИЕ СИСТЕМ MATCAD В ИНЖЕНЕРНЫХ РАСЧЕТАХ

Пособие по выполнению лабораторных работ по курсу «Информатика» для студентов технических специальностей дневной формы обучения

Составители: **Трохова** Татьяна Анатольевна
Романькова Татьяна Леонидовна

Подписано к размещению в электронную библиотеку
ГГТУ им. П О. Сухого в качестве электронного документа
учебно-методических материалов 28.07.08.

Рег. № 1Э.
E-mail: ic@gstu.gomel.by
http://www.gstu.gomel.by