МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт высокоточных систем им. В.П. Грязева

Кафедра систем автоматического управления

Утверждено на заседании кафедры
«Системы автоматического управления»
«13» января 2021 г., протокол № 8

Заведующий кафедрой

_____О.В.Горячев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
**по выполнению лабораторных работ
по дисциплине (модулю)**

«Системы аналитических вычислений»

**основной профессиональной образовательной программы
высшего образования – программы специалитета**

по специальности
24.05.06 Системы управления летательными аппаратами

со специализацией
Системы управления беспилотными летательными аппаратами

Форма обучения: очная

Идентификационный номер образовательной программы: 240506-01-21

Тула 2021 год

Разработчик(и) методических указаний

Макаров Н.Н., профессор, д.т.н., профессор
(ФИО, должность, ученая степень, ученое звание)

(подпись)

Оглавление

1 Вводное занятие. Правила техники безопасности при выполнении лабораторных работ в дисплейном классе. 4

2 Использование системы MathCad в качестве научного калькулятора. 4

3 Графика и анимация в пакете *MathCad*..... 9

4 Аналитические вычисления в системе *MathCad*..... 18

5 Приближение функций в пакете *MathCad*..... 27

6 Решение алгебраических уравнений и систем уравнений в пакете *MathCad*..... 34

7 Решение ОДУ в системе *MathCad*..... 43

8 Анализ линейной системы управления средствами системы *MathCad*..... 48

9 Синтез линейной системы управления средствами системы *MathCad*..... 52

10 Изучение интерфейса системы *Matlab*..... 56

11 Использование системы *Matlab* в качестве научного калькулятора. Работа с векторами и матрицами..... 62

12 Графические средства системы *Matlab*..... 76

13 Использование объектов и методов *Control System Toolbox* для анализа линейной системы управления..... 80

14 Использование визуальных средств синтеза линейной системы управления для расчёта корректирующего фильтра. *SISO design*. 93

15 Средства анализа и синтеза систем управления в пространстве состояний. 100

16 Визуальный инструмент синтеза системы Simulink..... 106

17 Список литиературы 114

1 Вводное занятие. Правила техники безопасности при выполнении лабораторных работ в дисплейном классе.

2 Использование системы MathCad в качестве научного калькулятора.

MathCAD работает с *документами*. С точки зрения пользователя, документ - это чистый лист бумаги, на котором можно размещать блоки трех основных типов: математические выражения, текстовые фрагменты и графические области.

Расположение нетекстовых блоков в документе имеет принципиальное значение – *слева направо и сверху вниз.*

Математические выражения

К основным элементам математических выражений MathCAD относятся *типы данных, операторы, функции* и *управляющие структуры.*

Операторы

Операторы - элементы MathCAD, с помощью которых можно создавать математические выражения. К ним, например, относятся символы арифметических операций, знаки вычисления сумм, произведений, производной и интеграла и т.д.

Оператор определяет:

1. действие, которое должно выполняться над операндами;
2. сколько, где и какие операнды должны быть введены в оператор.

Операнд – число или выражение, на которое действует оператор. Например, в выражении **5! + 3** число **3** и выражение **5!** – операнды оператора **+** (плюс), а число **5** операнд оператора факториал (**!**). После указания *операндов* операторы становятся исполняемыми по документу блоками.

Типы данных

К *типам данных* относятся числовые константы, обычные и системные переменные, массивы (векторы и матрицы) и данные файлового типа.

Константами называют поименованные объекты, хранящие некоторые значения, которые не могут быть изменены. ***Переменные*** являются поименованными объектами, имеющими некоторое значение, которое может изменяться по ходу выполнения программы. Тип переменной определяется ее значением; переменные могут быть числовыми, строковыми, символьными и т. д. Имена констант, переменных и иных объектов называют *идентификаторами*. Идентификаторы в MathCAD представляют собой набор латинских или греческих букв и цифр.

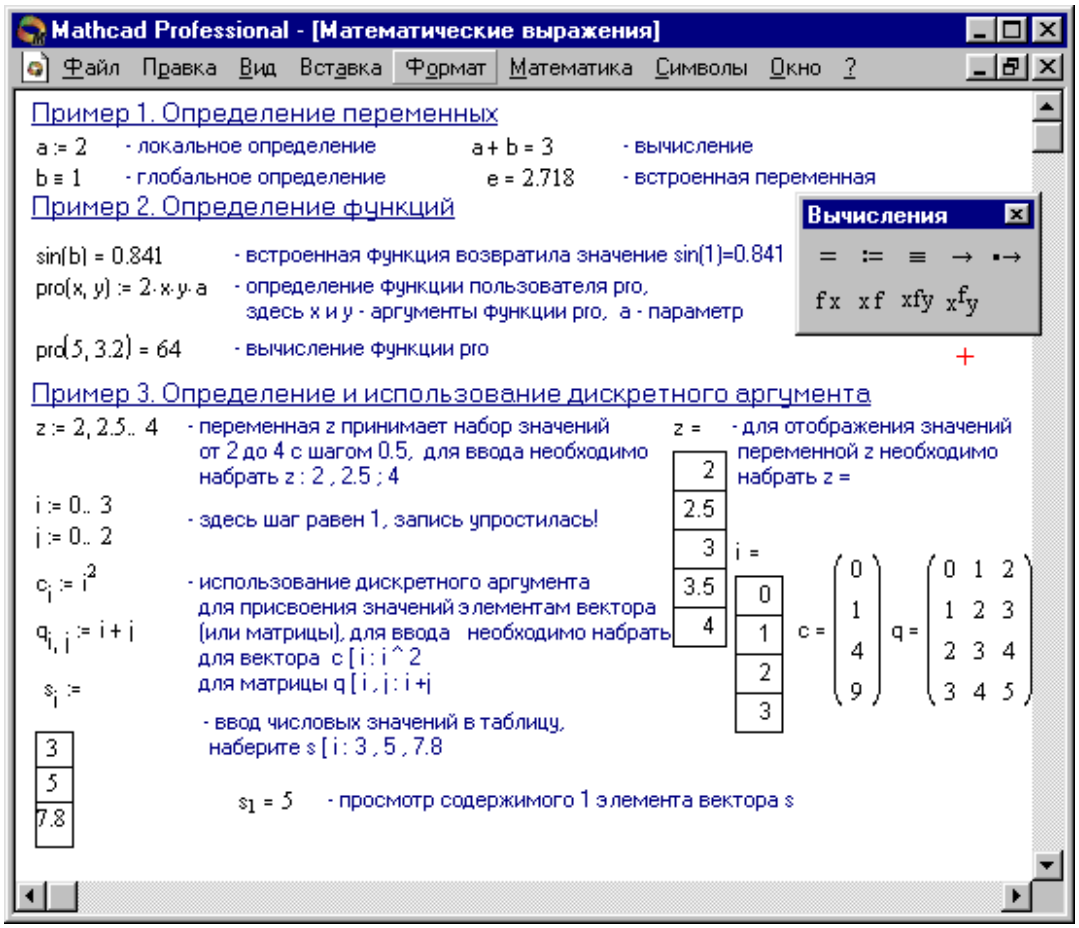


Рисунок 1. Математические выражения

В MathCAD содержится небольшая группа особых объектов, которые нельзя отнести ни к классу констант, ни к классу переменных, значения которых определены сразу после запуска программы. Их правильнее считать ***системными переменными***, имеющими предопределенные системой начальные значения (см. Приложение 1). Изменение значений системных переменных производят во вкладке **Встроенные переменные** диалогового окна **Math Options** команды **Математика ⇒ Опции.**

Обычные переменные отличаются от системных тем, что они должны быть предварительно *определены* пользователем, т. е. им необходимо хотя бы однажды *присвоить значение*. В качестве *оператора присваивания* используется знак **:=**, тогда как знак = отведен для *вывода значения* константы или переменной.

Если переменной присваивается начальное значение с помощью оператора :=, вызывается нажатием клавиши **:** (двоеточие) на клавиатуре, такое присваивание называется *локальным*. До этого присваивания переменная не определена и ее нельзя использовать. Однако с помощью знака ≡ (клавиша ~ на клавиатуре) можно обеспечить *глобальное* присваивание (см. Пример 1 Рисунка 1). MathCAD прочитывает весь документ дважды слева направо и сверху вниз. При первом проходе выполняются все действия, предписанные локальным оператором присваивания (≡), а при втором – производятся действия, предписанные

локальным оператором присваивания (:=), и отображаются все необходимые результаты вычислений (=).

Существуют также жирный знак равенства = (комбинация клавиш **Ctrl +** =), который используется, например, как оператор приближенного равенства при решении систем уравнений, и символьный знак равенства → (комбинация клавиш **Ctrl + .**).

Дискретные аргументы - особый класс переменных, который в пакете MathCAD зачастую заменяет ***управляющие структуры***, называемые циклами (однако полноценной такая замена не является). Эти переменные имеют ряд фиксированных значений, либо целочисленных (1 способ), либо в виде чисел с определенным шагом, меняющихся от начального значения до конечного (2 способ).

1. Name := Nbegin .. Nend,

где *Name* – имя переменной, *Nbegin* – ее начальное значение, *Nend* – конечное значение, **..** – символ, указывающий на изменение переменной в заданных пределах (вводится клавишей **;**). Если *Nbegin* < *Nend*, то шаг переменной будет равен +1, иначе –1.

2. Name := Nbegin, (Nbegin + Step) .. Nend

Здесь *Step* – заданный шаг изменения переменной (он должен быть положительным, если *Nbegin* < *Nend*, или отрицательным в обратном случае).

Дискретные аргументы значительно расширяют возможности MathCAD, позволяя выполнять многократные вычисления или циклы с повторяющимися вычислениями, формировать векторы и матрицы (Пример 3 Рисунка 1).

Массив - имеющая уникальное имя совокупность конечного числа числовых или символьных элементов, упорядоченных некоторым образом и имеющих определенные адреса. В пакете MathCAD используются массивы двух наиболее распространенных типов:

- одномерные (векторы);
- двумерные (матрицы).

Порядковый номер элемента, который является его адресом, называется *индексом*. Индексы могут иметь только целочисленные значения. Они могут начинаться с нуля или единицы, в соответствии со значением системной переменной **ORIGIN** (см. Приложение 1).

Векторы и матрицы можно задавать различными способами:

- с помощью команды **Вставка ⇒ Матрица**, или комбинации клавиш **Ctrl + M**, или щелчком на кнопке панели **Матрица,** заполнив массив пустых полей для не слишком больших массивов;
- с использованием дискретного аргумента, когда имеется некоторая явная зависимость для вычисления элементов через их индексы (Пример 3 Рисунка 1).

Функции

Функция – выражение, согласно которому проводятся некоторые вычисления с *аргументами* и определяется его числовое значение.

Следует особо отметить разницу между *аргументами* и *параметрами* функции. Переменные, указанные в скобках после имени функции, являются ее *аргументами* и заменяются при вычислении функции значениями из скобок. Переменные в правой части определения функции, не указанные скобках в левой части, являются *параметрами* и должны задаваться *до* определения функции (см. Пример 2 Рисунка 1).

Главным признаком функции является *возврат значения*, т.е. функция в ответ на обращение к ней по имени с указанием ее аргументов должна возвратить свое значение.

Функции в пакете MathCAD могут быть *встроенные* (см. Приложение 3), т. е. заблаговременно введенные разработчиками, и *определенные пользователем*.

Способы вставки встроенной функции:

1. Выбрать пункт меню **Вставка ⇒ Функция**.
2. Нажать комбинацию клавиш **Ctrl + E**.
3. Щелкнуть на кнопке .

Текстовые фрагменты

Текстовые фрагменты представляют собой куски текста, которые пользователь хотел бы видеть в своем документе. Существуют два вида текстовых фрагментов:

- *текстовая область* предназначена для небольших кусков текста - подписей, комментариев и т. п. Вставляется с помощью команды **Вставка ⇒ Текстовая регион** или комбинации клавиш **Shift + "** (двойная кавычка);
- *текстовый абзац* применяется в том случае, если необходимо работать с абзацами или страницами. Вставляется с помощью комбинации клавиш **Shift + Enter**.

Порядок выполнения лабораторной работы 1

Упражнение 1.

Вычислить:

$\sqrt{100} =$ $|-10| =$ $10! =$.

Это и все остальные задания снабдить комментариями, используя команду **Вставка ⇒ Текстовая область**.

Упражнение 2.

Определить переменные: $a := 3.4$, $b := 6.22$, $c \equiv 0.149$ (причем переменную c - глобально) и выражения:

$$Z := \frac{2ab + \sqrt[3]{c}}{\sqrt{\left(a^2 + b^{a+c}\right) \cdot c}} \qquad N := e^{\sin c} \cos\frac{a}{b}.$$

- Вычислить выражения.
- С помощью команды **Формат⇒Результат⇒Формат чисел⇒Число знаков** изменить точность отображения результатов вычисления *глобально*.

Упражнение 3.

Вывести на экран значение *системной константы* π и установить максимальный формат ее отображения *локально*.

Упражнение 4.

Выполнить следующие операции с комплексными числами:

Z := -3 + 2i |Z| = Re(Z) = Im(Z) = arg(Z) =

$\sqrt{Z}$ = $\sqrt{-5}$ = 2 · Z = Z1 := 1 + 2i Z2 := 3 + 4i

Z1 + Z2 = Z1 - Z2 = Z1· Z2 = Z1/Z2 =

Упражнение 5.

Выполнить следующие операции:

$i := 1 .. 10 \qquad \sum_i i = \qquad \prod_i (i+1) = \qquad \int_0^{0.4} x^2 \cdot \lg(x+2)dx =$

$\int_{0.8}^{1.2} \frac{\operatorname{ctg} 2x}{(\sin 2x)^2} dx = \qquad x := 2 \qquad \frac{d}{dx} x^5 = \qquad \frac{d}{dx}\sin(x) =$

Контрольные вопросы

1. С помощью какого оператора можно вычислить выражение?
2. Как вставить текстовую область в документ Mathcad?
3. Чем отличается глобальное и локальное определение переменных? С помощью каких операторов определяются?
4. Как изменить формат чисел для всего документа?
5. Как изменить формат чисел для отдельного выражения?
6. Какие системные (предопределенные) переменные Вам известны? Как узнать их значение? Как изменить их значение?
7. Какие виды функций в Mathcad Вам известны?
8. Как вставить встроенную функцию в документ Mathcad?
9. С помощью каких операторов можно вычислить интегралы, производные, суммы и произведения?
10. Как определить дискретные переменные с произвольным шагом? Какой шаг по умолчанию?
11. Как определить индексированную переменную?
12. Какие виды массивов в Mathcad Вам известны?
13. Какая системная переменная определяет нижнюю границу индексации элементов массива?
14. Опишите способы создания массивов в Mathcad.

15. Как просмотреть содержимое массива, определенного через дискретный аргумент?

3 Графика и анимация в пакете *MathCad.*

Графические области

Графические области делятся на три основных типа - двумерные графики, трехмерные графики и импортированные графические образы. Двумерные и трехмерные графики строятся самим MathCAD на основании обработанных данных.

Для создания декартового графика:

1. Установить визир в пустом месте рабочего документа.
2. Выбрать команду Вставка ⇒ График ⇒ Х-У график, или нажать комбинацию клавиш **Shift + @,** или щелкнуть кнопку панели **Графики**. Появится шаблон декартового графика.
3. Введите в плейсхолдер под осью *X* первую независимую переменную, через запятую – вторую и так до 10, например $x1$, $x2$, …
4. Введите в плейсхолдер слева от вертикальной оси *Y* первую независимую переменную, через запятую – вторую и т. д., например $y1(x1)$, $y2(x2)$, …, или соответствующие выражения.
5. Щелкните за пределами области графика, что бы начать его построение.

Трехмерные, или ***3D-графики***, отображают функции двух переменных вида $Z(X, Y)$. При построении трехмерных графиков в ранних версиях MathCAD поверхность нужно было определить математически (Рисунок 2, способ 2). Теперь применяют функцию MathCAD *CreateMesh.*

CreateMesh(F (или G, или $f1$, $f2$, $f3$), $x0$, $x1$, $y0$, $y1$, *xgrid*, *ygrid*, *fmap*)

Создает сетку на поверхности, определенной функцией F. $x0$, $x1$, $y0$, $y1$ – диапазон изменения переменных, *xgrid*, *ygrid* – размеры сетки переменных, *fmap* – функция отображения. Все параметры, за исключением F, - факультативные. Функция *CreateMesh* по умолчанию создает сетку на поверхности с диапазоном изменения переменных от –5 до 5 и с сеткой 20×20 точек.

Пример использования функции *CreateMesh* для построения 3D-графиков приведен на Рисунке 2, способ 1. На Рисунке 2 построена одна и та же поверхность разными способами, с разным форматированием, причем изображены поверхности и под ними те же поверхности в виде контурного графика. Такое построение способно придать рисунку большую наглядность.

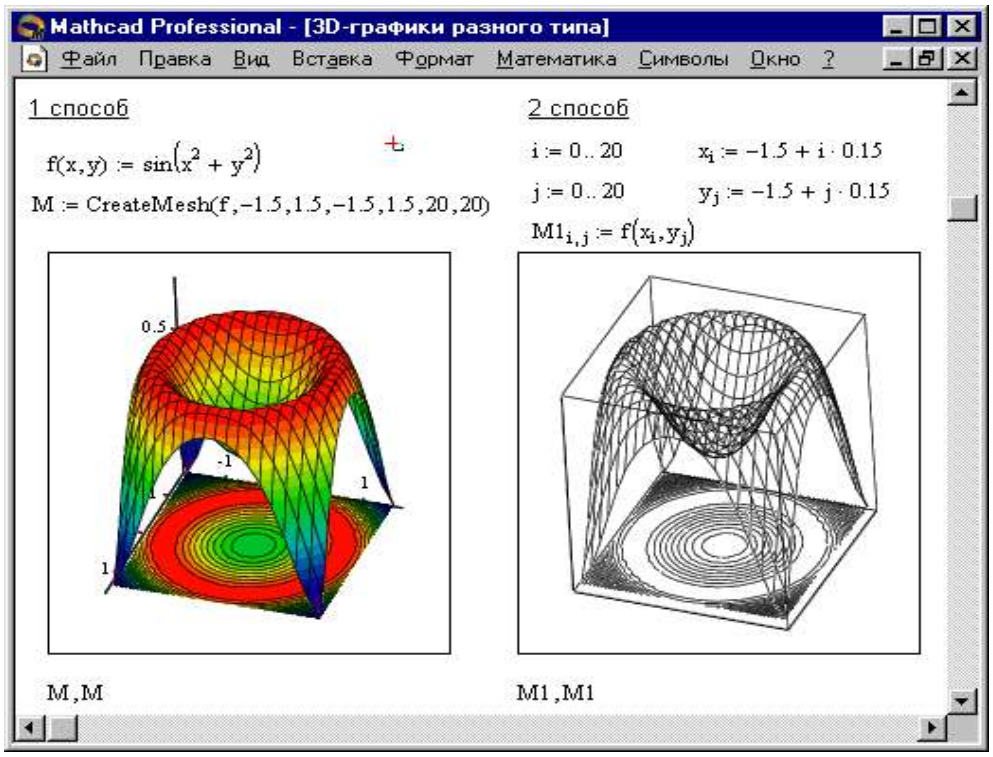


Рисунок 2. Пример построения на одном рисунке двух 3D-графиков разного типа

Нередко поверхности и пространственные кривые представляют в виде точек, кружочков или иных фигур. Такой график создается операцией **Вставка ⇒ График ⇒ 3D Точечный**, причем поверхность задается параметрически – с помощью трех матриц (*X*, *Y*, *Z*) (см. Рисунок 3, способ 2), а не одной как в примере на Рисунке 2. Для определения исходных данных для такого вида графиков используется функция *CreateSpace* (см. Рисунок 3, способ 1).

CreateSpace (*F* , *t*0, *t*1, *tgrid*, *fmap*)

Возвращает вложенный массив трех векторов, представляющих *x*-, *y*-, и *z*-координаты пространственной кривой, определенной функцией *F*. *t*0 и *t*1 – диапазон изменения переменной, *tgrid* – размер сетки переменной, *fmap* – функция отображения. Все параметры, за исключением *F*, - факультативные.

Построение пересекающихся фигур

Особый интерес представляет собой возможность построения на одном графике ряда разных фигур или поверхностей с автоматическим учетом их взаимного пересечения. Для этого надо раздельно задать матрицы соответствующих поверхностей и после вывода шаблона 3D-графика перечислить эти матрицы под ним с использованием в качестве разделителя запятой (Рисунок 4).

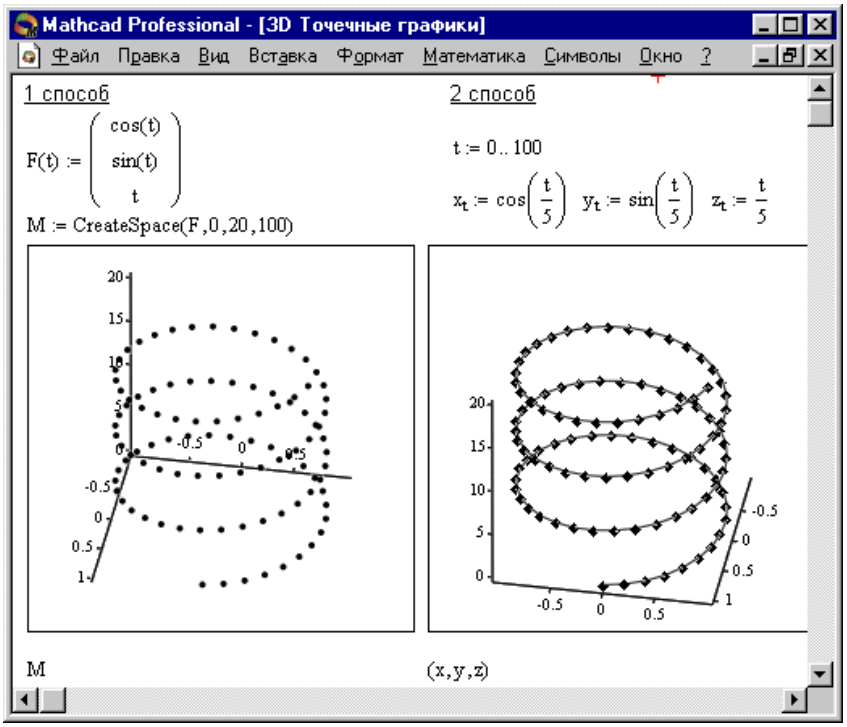


Рисунок 3.Построение 3D - точечных графиков

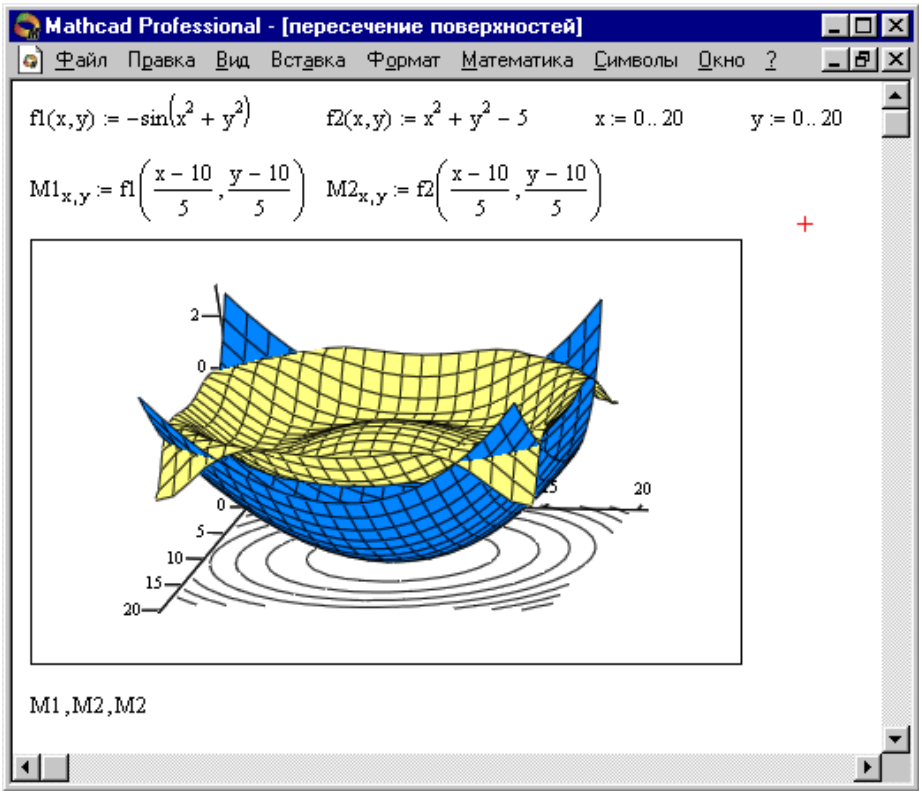


Рисунок 4.. Построение двух пересекающихся поверхностей и одного контурного графика одновременно.

Создание анимационного клипа

MathCAD имеет встроенную переменную FRAME, чье единственное назначение - управление *анимациями*:

- Создайте объект, чей вид зависит от FRAME.
- Убедитесь, что установлен режим автоматического расчета (**Математика ⇒ Автоматическое Вычисление**).
- Выберите **Вид ⇒ Анимация** для вызова одноименного диалогового окна.
- Заключите в выделяющий пунктирный прямоугольник часть рабочего документа, которую нужно анимировать.
- Установите нижние и верхние границы FRAME (поля **От:** и **До:**).
- В поле **Скорость** введите значение скорости воспроизведения (кадров/сек).
- Выберите **Анимация**. Сейчас анимация только создается.
- Сохраните анимацию как AVI файл (**Сохранить как**).
- Воспроизведите сохраненную анимацию **Вид ⇒Воспроизведение.**

Упражнение 1.

Определить векторы *d, S* и *R* через дискретный аргумент *i*. Отобразить графически таблично заданные функции $S_i(d_i)$ и $R_i(d_i)$, используя команду **Вставка⇒График⇒X-Y Зависимость**. Чтобы оформить график, необходимо выполнить следующие команды:

i	d_i	S_i	R_i
0	0.5	3.3	2
1	1	5.9	3.9
2	1.5	7	4.5
3	2	6.3	3.7
4	2.5	4.2	1.2

- Щелкнуть левой клавишей мыши на графике, чтобы выделить его. Затем щелкнуть правой клавишей мыши, при этом появится контекстное меню в котором необходимо выбрать команду **Формат** (появится диалоговое окно **“Formatting Currently Selected X-Y Plot”).**
- Нанести линии сетки на график (**Оси X-Y ⇒Вспом. линии**) и отобразить легенду (**След⇒Скрыть легенду**)
- Отформатировать график так, чтобы в каждой узловой точке графика функции $S_i(d_i)$ стоял знак вида □ (**След⇒Символ⇒box**), а график функции $R_i(d_i)$ отобразить в виде гистограммы (**След⇒Тип⇒bar**).

Упражнение 2.

Построить декартовы (**X-Y Зависимость**) и полярные (**Полярные Координаты**) графики следующих функций:

$X(\alpha) := \cos(\alpha) \cdot \sin(\alpha)$

$Y(\alpha) := 1.5\cos(\alpha)^2 - 1$

$P(\alpha) := \cos(\alpha).$

Для этого необходимо определить α как дискретный аргумент на интервале от 0 до 2·π с шагом π/30.

Определить по графику **X-Y Зависимость** координаты любой из точек пересечения графиков $Y(\alpha)$ и $P(\alpha)$, для этого необходимо:

- Выделить график и выбрать из контекстного меню **Масштаб** (появится диалоговое окно **“X-Y Zoom”**) для увеличения части графика в области точки пересечения.
- На чертеже выделить пунктирным прямоугольником окрестность точки пересечения графиков $Y(\alpha)$ и $P(\alpha)$, которую нужно увеличить.
- Нажать кнопку **Масштаб+**, чтобы перерисовать график.
- Чтобы сделать это изображение постоянным, выбрать ОК.
- Выбрать из контекстного меню Трассировка (появится диалоговое окно “X-Y Trace”).
- Внутри чертежа нажать кнопку мыши и переместить указатель мыши на точку, чьи координаты нужно увидеть.
- Выбрать **Copy X** (или **Copy Y**), на свободном поле документа набрать *Xper* := (или *Yper* :=) и выбрать пункт меню **Правка⇒Вставка.**

Вычислить значения функций $X(\alpha)$ и $Y(\alpha)$ при $\alpha := \pi/2$.

Упражнение 3.

Используя команду **Вставка⇒Матрица** создать матрицу *Q* размером 6×6, заполнить ее произвольно и отобразить графически с помощью команды **Вставка⇒График⇒Поверхности**.

Упражнение 4.

Построить график поверхности (**Поверхности**) и карту линий уровня (**Контурный**) для функции двух переменных

$X(t,\alpha) := t \cdot \cos(\alpha) \cdot \sin(\alpha)$, двумя способами:

1. С помощью функции *CreateMesh* (сетка размером 40 × 40, диапазон изменения *t* от –5 до 5, α - от 0 до 2·π).
2. Задав поверхность математически, для этого:
   - Определить функцию $X(t,\alpha)$
   - Задать на осях переменных *t* и α по 41 точке

   $i := 0..40 \quad j := 0..40$

   для переменной t_i со значениями, изменяющимися от -5 до 5 с шагом 0.25 $t_i := -5 + 0.25 \cdot i$, а для переменной α_j - от 0 до 2·π с шагом π/20 $\alpha_j := \pi/20 \cdot j$.
   - Определить матрицу $M_{i\ j} := X(t_i,\alpha_j)$ и отобразить ее графически.

   С помощью команды **Формат** контекстного меню вызвать диалоговое окно “**Формат 3-D графика”** и изменить:
   - характеристики просмотра (**Общее⇒Вид⇒Вращение, Наклон**),

- цвета и линии поверхности (Внешний Вид⇒Свойства линии, Свойства заливки),
- параметры осей (**Оси**),
- вид заголовка графика (**Название**).

Упражнение 5.

Отобразить графически пересечение поверхностей $f1(x, y) := \frac{(x+y)^2}{10}$ и $f2(x, y) := 5 \cdot \cos\left(\frac{x-y}{3}\right)$. Матрицы для построения поверхностей задать с помощью функции *CreateMesh*, значения факультативных параметров не указывать. Выполнить однотонную заливку для поверхностей, выбрав из контекстного меню команду **Формат**. Также из контекстного меню выбрать эффекты **Туман**, **Освещение, Перспектива**.

Упражнение 6.

Используя переменную **FRAME** и команду **Вид ⇒ Анимация**, создать анимационные клипы с помощью данных, приведенных в Таблице 1.

Таблица 1

Варианты упражнения 5

ариантa	Переменные и функции	FRAME	Тип графика
1	$x := 0, 0.1 .. 30$ $f(x) := x +$ FRAME	от 0 до 20	График Полярные Координаты
2	$i := 0 ..$ FRAME $+ 1$ $g_i := 0.5 \cdot i \cdot \cos(i)$ $h_i := i \cdot \sin(i)$ $k_i := 2 \cdot i$	от 0 до 50	3D точечный график границы на осях Min Max x - 50 50 y - 50 50 z 0 50 В плейсхолдере для ввода матрицы укажите (g, h, k)
3	$i := 0 .. 20$ $j := 0 .. 20$ $f(x,y) := \sin(x^2 + y^2 +$ FRAME$)$ $x_i := -1.5 + 0.15 \cdot i$ $y_j := -1.5 + 0.15 \cdot j$ $M_{i,j} := f(x_i, y_j)$	от 0 до 50	График Поверхности В плейсхолдере для ввода матрицы укажите M

ариантa	Переменные и функции	FRAME	Тип графика
4	r := FRAME R := 6 n := 0 .. 20 m := 0 .. 20 $v_n := \frac{2\cdot\pi\cdot n}{r+1}$ $w_m := \frac{2\cdot\pi\cdot m}{r+1}$ $x_{m\,n} := (R + r\cdot\cos(v_n))\cdot\cos(w_m)$ $y_{m\,n} := (R + r\cdot\cos(v_n))\cdot\sin(w_m)$ $z_{m\,n} := r\cdot\sin(v_n)$	от 0 до 20	График Поверхности (границы на всех осях установить от -11 до 11) В плейсхолдере для ввода матрицы укажите (x, y, z)

Контрольные вопросы

1. Как построить графики: поверхности; полярный; декартовый?
2. Как построить несколько графиков в одной системе координат?
3. Как изменить масштаб графика?
4. Как определить координату точки на графике?
5. Как построить гистограмму?
6. Какие функции используются для построения трехмерных графиков?
7. Как создать анимацию в Mathcad?
8. Какое расширение имеют сохраненные файлы анимаций?

4 Аналитические вычисления в системе *MathCad*.

Системы компьютерной алгебры снабжаются специальным процессором для выполнения аналитических (символьных) вычислений. Его основой является ядро, хранящее всю совокупность формул и формульных преобразований, с помощью которых производятся аналитические вычисления. Чем больше этих формул в ядре, тем надежней работа символьного процессора и тем вероятнее, что поставленная задача будет решена, если такое решение существует в принципе (что бывает далеко не всегда).

Ядро символьного процессора системы MathCAD — несколько упрощенный вариант ядра известной системы символьной математики Maple V фирмы Waterloo Maple Software, у которой фирма MathSoft (разработчик MathCAD) приобрела лицензию на его применение, благодаря чему MathCAD стал (начиная с версии 3. 0) системой символьной математики. Символьные вычисления выполняются столь же просто (для пользователя), как вычисление квадрата x.

Символьные операции можно выполнять двумя способами:

- Непосредственно в командном режиме (используя операции меню **Symbolics**);
- С помощью операторов символьного преобразования (используя панель инструментов **Symbolic Keyword Toolbar**).

Рассмотрим первый способ.

Выделение выражений для символьных вычислений

Чтобы символьные операции выполнялись, процессору необходимо указать, над каким выражением эти операции должны производиться, т. е. надо выделить выражение. Для ряда операций следует не только указать выражение, к которому они относятся, но и наметить переменную, относительно которой выполняется та или иная символьная операция. Само выражение в таком случае не выделяется.

Таким образом, для выполнения операций с символьным процессором нужно выделить объект (целое выражение или его часть) синими сплошными линиями.

Символьные операции разбиты на пять характерных разделов. Первыми идут наиболее часто используемые операции. Они могут выполняться с выражениями, содержащими комплексные числа или имеющими решения в комплексном виде.

Символьные операции

Операции с выделенными выражениями

Если в документе есть выделенное выражение, то с ним можно выполнять различные операции, представленные ниже:

Evaluate — преобразовать выражение с выбором вида преобразований из подменю;

Symbolicaly [Shift] F9 – выполнить символьное преобразование выделенного выражения;

Floating Point… – вычислить выделенное выражение в вещественных числах;

Complex – выполнить вычисления в комплексном виде;

Simplify — упростить выделенное выражение с выполнением таких операций, как сокращение подобных слагаемых, приведение к общему знаменателю, использование основных тригонометрических тождеств и т д.;

Expand — раскрыть выражение [например, для $(X + Y)(X - Y)$ получаем $X^2 - Y^2$];

Factor — разложить число или выражение на множители [например, $X^2 - Y^2$ даст $(X + Y)(X - Y)$];

Collect — собрать слагаемые, подобные выделенному выражению, которое может быть отдельной переменной или функцией со своим аргументом (результатом будет выражение, полиномиальное относительно выбранного выражения);

Polynomial Coefficients — по заданной переменной найти коэффициенты полинома, аппроксимирующего выражение, в котором эта переменная использована.

Операции с выделенными переменными

Для ряда операций надо знать, относительно какой переменной они выполняются. В этом случае необходимо выделить переменную, установив на ней маркер ввода. После этого становятся доступными следующие операции подменю **Variable**:

Solve — найти значения выделенной переменной, при которых содержащее ее выражение становится равным нулю;

Substitute — заменить указанную переменную содержимым буфера обмена;

Differentiate — дифференцировать выражение, содержащее выделенную переменную, по этой переменной (остальные переменные рассматриваются как константы);

Integrate — интегрировать все выражение, содержащее переменную, по этой переменной;

Expand To Series... — найти несколько членов разложения выражения в ряд Тейлора относительно выделенной переменной;

Convert to Partial Fraction — разложить на элементарные дроби выражение, которое рассматривается как рациональная дробь относительно выделенной переменной.

Операции с выделенными матрицами

Операции с выделенными матрицами представлены позицией подменю **Matrix**, которая имеет свое подменю со следующими операциями:

Transpose — получить транспонированную матрицу;

Invert — создать обратную матрицу;

Determinant — вычислить детерминант (определитель) матрицы.

Результаты символьных операций с матрицами часто оказываются чрезмерно громоздкими и поэтому плохо обозримы.

Интегральные преобразования

В позиции **Transform** содержится раздел операций преобразования, создающий подменю со следующими возможностями:

Fourier — выполнить прямое преобразование Фурье относительно выделенной переменной;

Inverse Fourier — выполнить обратное преобразование Фурье относительно выделенной переменной;

Laplace — выполнить прямое преобразование Лапласа относительно выделенной переменной (результат — функция переменной s);

Inverse Laplace — выполнить обратное преобразование Лапласа относительно выделенной переменной (результат — функция переменной *t*);

Z — выполнить прямое Z-преобразование выражения относительно выделенной переменной (результат — функция переменной *z*);

Inverse Z — выполнить обратное Z-преобразование относительно выделенной переменной (результат — функция переменной *n*) .

Стиль представления результатов вычислений

На наглядность вычислений влияет стиль представления их результатов. Следующая команда позволяет задать тот или иной стиль:

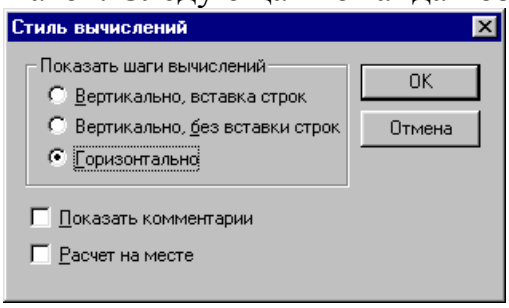


Рисунок 3. Стиль Вычислений

Evaluation Style... — задать вывод результата символьной операции под основным выражением, рядом с ним или вместо него (Рисунок 1).

Примеры символьных операций в командном режиме

Большинство символьных операций легко выполняются, так что ниже мы остановимся лишь на некоторых примерах. Символьная операция **Evaluate** обеспечивает работу с математическими выражениями, содержащими встроенные в систему функции и представленными в различном виде: полиномиальном, дробно-рациональном, в виде сумм и произведений, производных и интегралов и т. д. (Рисунок 2). Операция стремится произвести все возможные численные вычисления и представить выражение в наиболее простом виде. Она возможна над матрицами с

символьными элементами. Производные и определенные интегралы, символьные значения которых вычисляются, должны быть представлены в своей естественной форме.

Особо следует отметить возможность выполнения численных вычислений с повышенной точностью — 20 знаков после запятой. Для перехода в такой режим вычислений нужно числовые константы в вычисляемых объектах задавать с обязательным указанием десятичной точки, например 10.0 или 3.0, а не 10 или 3. Этот признак является указанием на проведение вычислений такого типа.

На Рисунке 2 показаны типовые примеры действия операции **Evaluate.**

Здесь слева показаны исходные выражения, подвергаемые символьным преобразованиям, а справа — результат этих преобразований.

Операция **Evaluate** одна из самых мощных. Как видно из Рисунка 2, она позволяет в символьном виде вычислять суммы (и произведения) рядов, производные и неопределенные интегралы, выполнять символьные и численные операции с матрицами.

Эта операция содержит подменю. Команда **Symbolicaly** тут наиболее важная. Назначение других команд очевидно: они нужны, если результат требуется получить в форме комплексного или действительного числа. К примеру, если вы хотите вместо числа π получить 3.141..., используйте команду **Floating Point….** В режиме символьных вычислений результат может превосходить машинную бесконечность системы — см. пример на вычисление exp(1000.0) на Рисунке 2. При этом число точных значащих цифр результата практически не ограничено (или, точнее говоря, зависит от емкости ОЗУ).

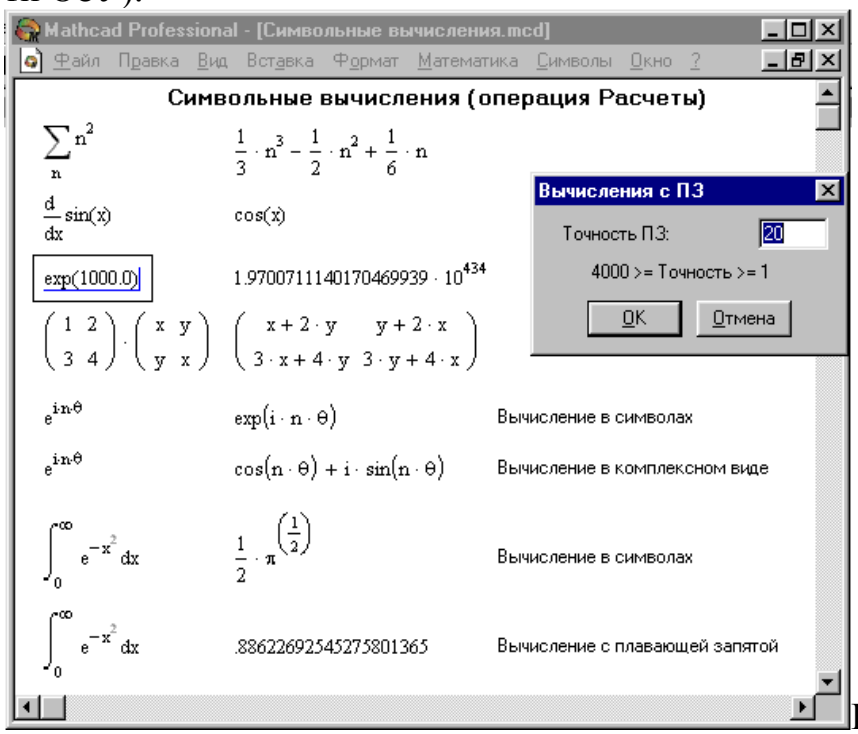


Рисунок 4. Символьные вычисления

Операция **Polynomial Coefficients...** возвращает разложение в ряд Тейлора выражения относительно выделенной переменной с заданным по запросу числом членов ряда *n* (число определяется по степеням ряда). По умолчанию задано *n* = 6. В разложении указывается остаточная погрешность разложения. На Рисунке 3 представлено применение этой операции для разложения функции $\frac{\sin(x)}{x}$. Минимальная погрешность получается при малых *x* (см. графическое представление функции и ее ряда).

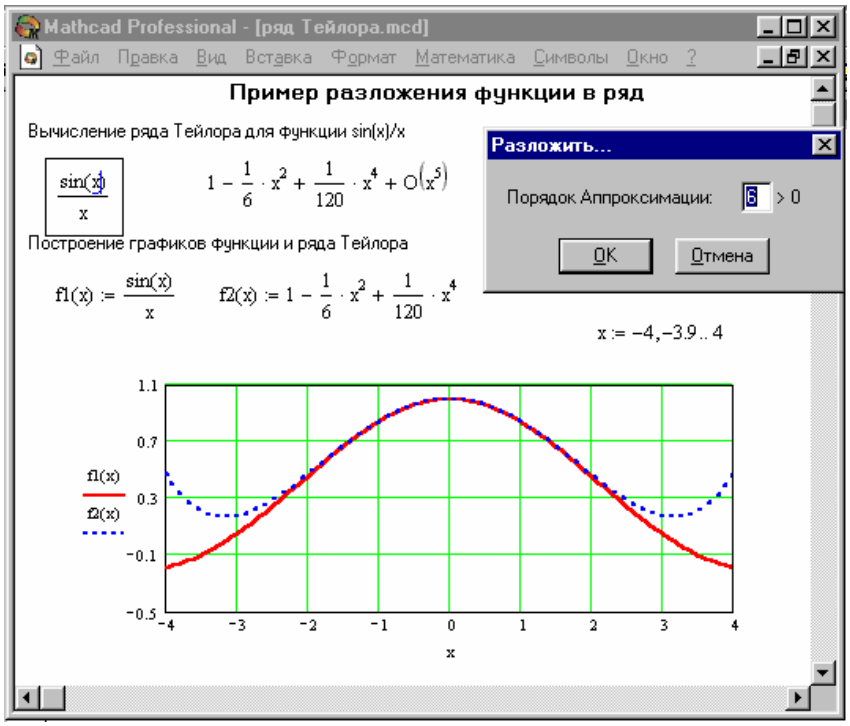


Рисунок 3.. Разложение функции в ряд Тейлора

Операторы вычисления пределов функций

Для вычисления пределов функций в систему введен символьный оператор limit. Помимо ввода с наборной панели **Calculus,** его в трех формах можно ввести нажатием следующих комбинаций клавиш:

[Ctrl] **L** — ввод шаблона оператора вычисления предела функции при *x,* стремящемся к заданному значению,

[Ctrl] **A** — ввод шаблона вычисления предела функции слева от заданной точки,

[Ctrl] **B** — ввод шаблона вычисления предела функции справа от заданной точки.

На Рисунке 4 показаны примеры вычисления пределов. При вычислении пределов нужно заполнить шаблоны, входящие в главный шаблон для вычисления пределов, а затем ввести функцию, имя переменной, по которой ищется предел, и значение переменной — аргумента функции.

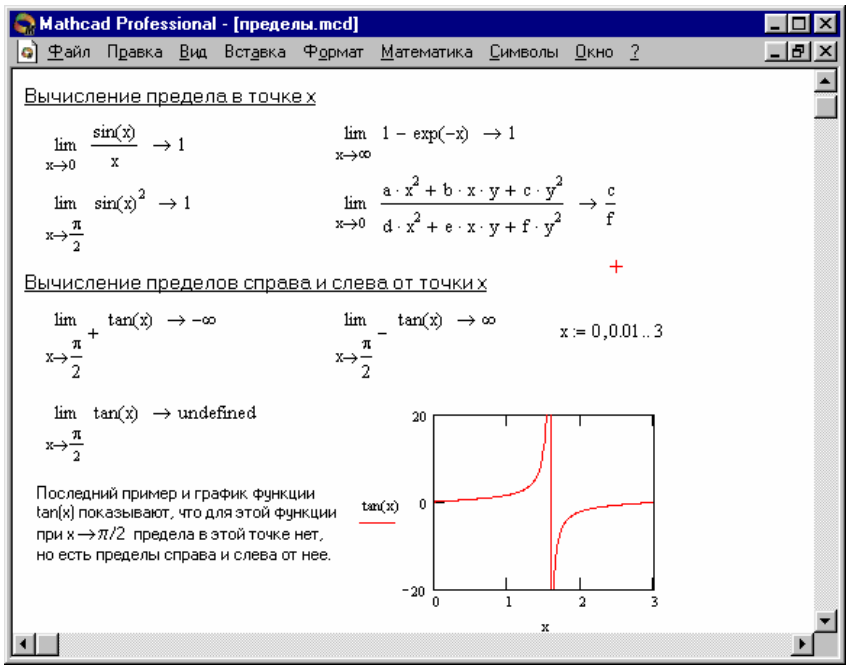


Рисунок 4. Вычисление пределов

Для получения результата установите после блока вычисления предела стрелку с острием, направленным вправо. Предел (если он существует) будет вычислен и появится в шаблоне у острия стрелки. Если функция не имеет предела, вместо результата появится надпись сообщение (hint) *No symbolic result was found*.

Порядок выполнения лабораторной работы

Внимание! Все задания Вы должны выполнить в двух вариантах – с использованием системного меню **Symbolics** и с использованием панели инструментов **Symbolic Keyword Toolbar.**

Упражнение 1. Используя операцию вычисления с плавающей точкой, представьте:

1) число π в 7 позициях;
2) число 12,345667 в 3 позициях.

Упражнение 2. Вычислите символьно комплексные значения выражений:

1) $\sqrt{-7}$;
2) tg (a $\sqrt{-3}$);
3) $e^{1+\frac{\pi}{4}i}$;
4) для выражения 3) выполните упрощение результата**.**

Упражнение 3. Для полинома $g(x)$ (см. Таблица 1) выполнить следующие действия:

1) разложить на множители, используя операцию **Factor;**
2) подставьте выражение $x = y + z$ в $g(x)$, используя операцию **Substitute.** (Для выполнения подстановки через системное меню предварительно надо скопировать подставляемое выражение в буфер обмена, выделив его и нажав комбинацию клавиш **Ctrl + C** или **F2**);

3) используя операцию **Expand,** разложите по степеням выражение, полученное в 2)**;**
4) используя операцию **Collect,** сверните выражение, полученное в 3), по переменной *z***.**

Таблица 1

Варианты упражнения 3

ариант а	g(x)	ариа-анта	$g(x)$
1	$x^4 - 2x^3 + x^2 - 12x + 20$	**9**	$x^4 + x^3 - 17x^2 - 45x - 100$
2	$x^4 + 6x^3 + x^2 - 4x - 60$	**10**	$x^4 - 5x^3 + x^2 - 15x + 50$
3	$x^4 - 14x^2 - 40x - 75$	**11**	$x^4 - 4x^3 - 2x^2 - 20x + 25$
4	$x^4 - x^3 + x^2 - 11x + 10$	**12**	$x^4 + 5x^3 + 7x^2 + 7x - 20$
5	$x^4 - x^3 - 29x^2 - 71x - 140$	**13**	$x^4 - 7x^3 + 7x^2 - 5x + 100$
6	$x^4 + 7x^3 + 9x^2 + 13x - 30$	**14**	$x^4 + 10x^3 + 36x^2 + 70x + 75$
7	$x^4 + 3x^3 - 23x^2 - 55x - 150$	**15**	$x^4 + 9x^3 + 31x^2 + 59x + 60$
8	$x^4 - 6x^3 + 4x^2 + 10x + 75$		

Упражнение 4. Разложите выражения на элементарные дроби

1) $\frac{6x^2 - x + 1}{x^3 - x}$; 2) $\frac{3x^2 - 2}{(x^2 + x + 1)(x + 1)}$;

3) $\frac{x + 1}{x(x - 1)^3}$; 4) $\frac{5x^2 - 4x + 16}{(x^2 - x + 1)^2 (x - 3)}$.

Упражнение 5. Разложите выражения в ряд Тейлора с заданным порядком**:**

1) $\ln(1 + x)$, $x_0 = 0$, порядок разложения 6;
2) $\sin(x)^2$, $x_0 = 0$, порядок разложения 6.

Упражнение 6. Найдите первообразную аналитически заданной функции $f(x)$ (Таблица 4).

Упражнение 7. Определите символьное значение первой и второй производных $f(x)$ (Таблица 4).

Таблица 4

Варианты упражнений 6 и 7

вари-анта	$f(x)$	ари-анта	$f(x)$	вари-анта	$f(x)$
1	$1/(\mathrm{tg}2x$	**6**	$x^2 \cdot$	**1**	$(2x + 3)$ $\sin x$
2		**7**		**2**	$\cos 3x/(1-\mathrm{co}$ 2
3	$1/(x$ $\sqrt{x^3+4})$	**8**	2	**3**	$1/(1 + x$ $+ x^2)$
4		**9**	$(x + 1)$ $\sin x$	**4**	
5	$x^2 \cdot$	**0**	$5x + x$ $\lg x$	**5**	

1) Транспонируйте символьно матрицу M

$$\begin{pmatrix} 1 & a & b \\ x & 2 & c \\ x^2 & 3 & d \end{pmatrix}$$

2) Инвертируйте (вычислите обратную матрицу) символьно матрицу

$$\begin{pmatrix} 1 & y \\ x & 2 \end{pmatrix}$$

3) Вычислите символьно определитель матрицы M

$$\begin{pmatrix} 1 & a & b \\ x & 2 & c \\ x^2 & 3 & d \end{pmatrix}$$

Упражнение 8. Вычислите символьно пределы:

1) $\lim\limits_{x\to 1} \dfrac{x^2 + 2\cdot x + 5}{x^2 + 1}$

2) $\lim\limits_{x\to \frac{\pi}{2}} (2\cdot \sin(x) - \cos(x) + \mathrm{ctg}(x))$

3) $\lim\limits_{h\to 0} \dfrac{(x+h)^3 - x^3}{h}$

4) $\lim\limits_{x\to\infty} \dfrac{\sqrt{x^2-3}}{\sqrt[3]{x^3}+1} \to 1$

5) $\lim\limits_{x\to+\infty} x\cdot\left(\sqrt{x^2+1} - x\right)$

6) $\lim\limits_{x \to -\infty} x \cdot \left(\sqrt{x^2 + 1} - x\right)$

7) $\lim\limits_{x \to 0^+} (1 + x)^{\frac{1}{x}}$

8) $\lim\limits_{n \to \infty} - \left(1 + \frac{1}{n}\right)^n$

Контрольные вопросы

1. Назовите способы выполнения символьных операций в MathCAD.
2. Что необходимо сделать с выражением перед применением символьных преобразований в командном режиме?
3. Перечислите символьные операции с выделенными выражениями.
4. Перечислите символьные операции с выделенными переменными.
5. Перечислите символьные операции с выделенными матрицами.
6. Перечислите символьные операции преобразования.
7. Какие параметры определяет стиль представления результатов вычислений и где он задается?
8. В каких случаях результат символьных преобразований помещается в буфер обмена?
9. Каким образом можно вычислить предел в MathCAD?

5 Приближение функций в пакете *MathCad.*

Интерполяция функций

Пусть имеется таблично заданная функция (таблица данных), представленная двумя векторами

$$x := \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}^T \quad y := \begin{pmatrix} 0 & 2 & -1 \end{pmatrix}^T$$

Линейную интерполяцию этой функции осуществляет функция *linterp.* Ей требуется три аргумента: два вектора, *x* и *y*, содержащие интерполируемую таблицу, и скаляр – значение аргумента, для которого требуется вычислить значение интерполированной функции. Например, вычислим интерполированное значение для точки *x* = 1.25

$$linterp(x, y, 1.25) = 0.5$$

Несколько сложнее выполняется интерполяция кубическим сплайном. Она состоит из двух последовательных этапов: сначала по таблице данных вычисляются коэффициенты сплайна, для чего служит любая из трёх функций *cspline, lspline, pspline.* Они различаются лишь дополнительными условиями, дополняющими собственно уравнения интерполяции и непрерывности первых двух производных.Аргументами этих функций служат вектора *x* и *y* то есть таблица данных.

$$c_c := cspline(x, y) \qquad c_l := lspline(x, y) \quad c_p := pspline(x, y)$$

$$c_c = \begin{pmatrix} 0 \\ 3 \\ 2 \\ -10 \\ -5 \\ 0 \end{pmatrix} \qquad c_l = \begin{pmatrix} 0 \\ 3 \\ 0 \\ 0 \\ -7.5 \\ 0 \end{pmatrix} \qquad c_p = \begin{pmatrix} 0 \\ 3 \\ 1 \\ -5 \\ -5 \\ -5 \end{pmatrix}$$

Для вычисления интерполированных значений в любом варианте сплайна используется функция *interp* которой надо передать четыре аргумента – вычисленный вектор коэффициентов, вектор значений аргумента и вектор значений функции, которую интерполируют, и, наконец, то значение, для которого вычисляется интерполированное значение. Например в точке 1,25 значение интерполированной функции вычисляется так:

$$interp(c_c, x, y, 1.25) = 1.242$$

Исходная таблица содержит 3 узла, следовательно сплайн включает 2 кубических многочлена и имеет 8 коэффициентов, однако вычисляется только 6. Последние n (в данном случае - 3) из вычисленных коэффициентов

- это значения 2-й производной сплайна в узловых точках. Первые 3 числа используются функцией interp для её целей.

К сожалению, классическая ньютонова интерполяция полиномами в *Mathcad* не реализована. Но её нетрудно осуществить самостоятельно, воспользовавшись ньютоновой формулой

$$\varphi(x)=\sum_{i=1}^{n} y_i \cdot \prod_{\substack{j=1\\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}.$$

Аппроксимация функций

Сама по себе идея интерполяции, то есть требования, чтобы приближённая функция в точности проходила через все узлы имеющейся таблицы, не всегда оправдана. Во-первых, при длинных таблицах она приводит к сложным приближениям. Во-вторых, она не кажется разумной в тех случаях, когда таблица заведомо не точна, что всегда имеет место при использовании экспериментальных данных. Часто более полезным оказывается потребовать, чтобы приближённая функция проходила близко к узлам таблицы. Дадим этой близости количественный смысл:

$$\varepsilon\left(\{x_i, y_i\}_n, \varphi(x,\alpha)\right)=\sum_{i=1}^{n}\left(y_i-\varphi(x_i,\alpha)\right)^2$$

Функционал качества приближения носит название *среднеквадратичного отклонения* (СКО). Он равен сумме квадратов "промахов" приближённой функции по всем узлам таблицы. Теперь осталось лишь подобрать значение параметра α таким, чтобы СКО оказалось наименьшим из всех возможных:

$$\sum_{i=1}^{n}\left(y_i-\varphi(x_i,\alpha)\right)^2 \xrightarrow[\alpha]{} \min$$

В точке минимума функции, как известно, все её частные производные обращаются в ноль, что приводит нас к следующей системе уравнений:

$$\frac{\partial}{\partial \alpha_k}\varepsilon\left(\{x_i, y_i\}_n, \varphi(x,\alpha)\right)=0, \quad k=1,2\ldots m$$

Для задачи эта система уравнений приобретает вид

$$\sum_{i=1}^{n}\left[\varphi(x_i,\alpha)\frac{\partial}{\partial \alpha_k}\varphi(x_i,\alpha)\right]=\sum_{i=1}^{n}\left[y_i\frac{\partial}{\partial \alpha_k}\varphi(x_i,\alpha)\right],$$

$$k=1,2\ldots m.$$

Задача минимизации СКО особенно просто решается, если в качестве приближённой функции использовать *обобщённый многочлен*.

Выберем произвольно m функций $\varphi_1(x),\varphi_2(x),\ldots\varphi_n(x)$ и будем в качестве приближённой использовать их взвешенную сумму:

$$\varphi(x)=\sum_{i=1}^{m}\alpha_i\varphi_i(x).$$

Именно такую взвешенную сумму и называют *обобщённым многочленом.* Входящие в него фиксированные функции $\varphi_i(x)$ называются *базисными*, а их совокупность – *базисом*. Подставляя, получаем окончательную формулировку задачи приближения:

$$\sum_{i=1}^{n}\left(y_i-\sum_{k=1}^{m}\alpha_k\varphi_k(x_i)\right)^2\xrightarrow[\alpha]{}\min$$

Легко убедиться, что эта задача приводит к системе линейных алгебраических уравнений относительно коэффициентов обобщённого многочлена:

$$A\alpha=b,$$

где A – квадратная матрица размерности m.
b – вектор той же размерности,
определяемые выражениями

$$A_{i,j}=\sum_{k=1}^{n}\varphi_i(x_k)\varphi_j(x_k),$$

$$b_i=\sum_{k=1}^{n}y_k\varphi_i(x_k),\quad i,j=1,2\text{K}\ m.$$

Эта система всегда имеет единственное решение, если только матрица A – невырожденная. Такой способ получения приближённой функции получил название *метода наименьших квадратов* (МНК).

Описанный подход к получению приближённой функции встречается в литературе под разными названиями. В русскоязычных публикациях наиболее распространены термины *аппроксимация* и *регрессия*. В англоязычных публикациях кроме последнего термина, *Regression*, часто используется выражение *подгонка* (*Curve Fitting*).

В Маткаде достаточно большая группа функций нацелена на среднеквадратичную аппроксимацию (*Curve Fitting and Regression*). Из них следует выделить две наиболее универсальные – *genfit* и *linfit*. Кроме того часто полезно использовать функцию полиномиальной аппроксимации *regress*.

***linfit*(*vx,vy,F*)** возвращает вектор коэффициентов обобщённого многочлена, аппроксимирующего функцию, заданную таблично. Поскольку задача сводится к решению СЛАУ, начальное приближение не требуется.

- ***vx*** и ***vy*** – вектора одинаковой размерности, содержащие иксы и игреки таблично заданной функции,

- ***F*(*x*)** – векторная функция скалярного аргумента, координаты этой функции образуют базис аппроксимации, при вызове используется без аргументов.

Размерность векторов данных не может быть ниже, чем размерность базиса аппроксимации. *В случае совпадения размерностей мы получим интерполирующую функцию!*

***genfit*(*vx,vy,vg,F*)** возвращает вектор параметров аппроксимации.

- ***vx*** и ***vy*** - как и выше, вектора, содержащие исходную функцию,
- ***vg*** – вектор начальных приближений вычисляемых параметов,
- ***F*(*x,u*)** – векторная функция, в которой первая координата содержит собственно аппроксимирующую функцию, описанную как функцию собственно независимого аргумента и вектора параметров или списка параметров. Задача как раз и заключается в определении значений этих параметров. Остальные координаты – частные производные по параметрам. Все координаты должны иметь идентичный список аргументов.

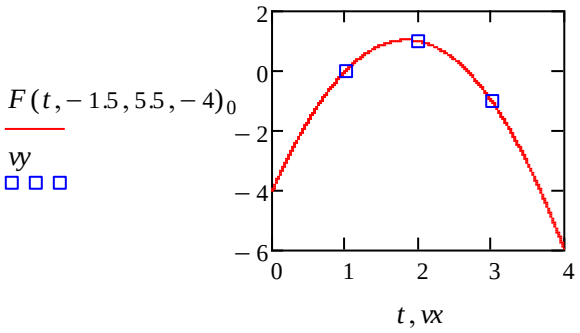
Функция, *genfit*, использует итерационные численные методы и поэтому нуждается в начальном приближении и не гарантирует результата, *linfit* и *regress* всегда дают единственный ответ и не нуждаются в начальном приближении.

Пример использования функции *genfit*:

$$F(x,a,b,c) := \begin{pmatrix} a\cdot x^2 + b\cdot x + c \\ x^2 \\ x \\ 1 \end{pmatrix}$$

$$vx := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \qquad vy := \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \qquad vg := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

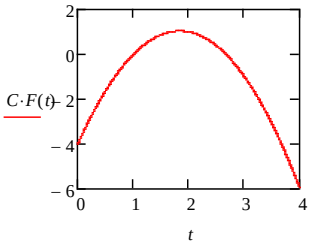
$$genfit(vx,vy,vg,F) = \begin{pmatrix} -1.5 \\ 5.5 \\ -4 \end{pmatrix}$$



Пример несколько примитивен, поскольку параметры *a,b,c* входят в аппроксимирующую функцию линейно, и логичнее было бы использовать функцию *linfit*, а не *genfit*. Как в следующем примере.

Пример использования функции *linfit*:

$$F(x) := \begin{matrix} x^2 \\ x \\ 1 \end{matrix} \qquad C := linfit(vx, vy, F) = \begin{pmatrix} -1.5 \\ 5.5 \\ -4 \end{pmatrix}$$



Порядок выполнения лабораторной работы

Для упражнений 1-3 требуется:

а) Табулировать заданную функцию в интервале (0; 3), то есть составить таблицу значений *x* и *y* в виде двух векторов размерности 5.

б) Построить графики исходной функции, табулированной функции и приближённой функции.

в) Оценить погрешность приближения и сделать вывод о качестве приближения.

г) Подобрать рациональный шаг таблицы (возможно, не постоянный).

Упражнение 1.

Выполнить линейную интерполяцию заданной функции.

Упражнение 2.

Выполнить интерполяцию заданной функции по формуле Ньютона.

Упражнение 3.

Выполнить интерполяцию заданной функции сплайном с различными доопределениями вторых производных.

Упражнение 4.

Выполнить аппроксимацию таблично заданной функции с использованием функции *linfit*. Построить графики исходной и аппроксимированной функций.

Упражнение 5.

Выполнить аппроксимацию таблично заданной функции с использованием функции *genfit*. Построить графики исходной и аппроксимированной функций.

Варианты заданий для упражнений 1-3

	$f1(t) := e^{-2\cdot t}\cdot sin(3\cdot t)$		$f11(t) := ln(t+1)\cdot sin(3\cdot t)$
	$f2(t) := sin(2.5t)\cdot cos(t)$		$f12(t) := sin(2.5t)\cdot cos\left(\frac{1.3}{t+1}\right)$
	$f3(t) := e^{-t}\cdot cos(3\cdot t)$		$f13(t) := ln(t+1)\cdot cos(3\cdot t)$
	$f4(t) := e^{-t}\cdot sin(4\cdot t)$	0	$f14(t) := sin\left(4\cdot t\cdot e^{-t}\right)$
	$f5(t) := sin(2\cdot t)\cdot cos(1.3t)$	1	$f15(t) := sin(2\cdot t)\cdot cos(ln(3\cdot t+.1))$
	$f6(t) := e^{-1.3\cdot t}\cdot cos(2\cdot t)$	2	$f16(t) := e^{-1.3\cdot t}\cdot cos\left(2\cdot t^2\right)$

Варианты заданий для упражнений 4-5

Значения аппроксимируемой функции по вариантам

		.3	.6	.9	.2	.5	.8	.1	.4	.7	
1	0.022	.396	.27	.023	0.124	0.046	0.027	.028	.116	.045	.05
2	.043	.697	.857	.431	.055	0.078	.257	.425	.174	0.443	0.954
3	.028	.448	0.12	0.304	0.305	0.047	.16	.167	0.09	0.124	0.035
4	0.031	.631	.376	0.141	0.285	0.062	.093	.084	0.05	0.073	.024
5	.009	.548	.7	.379	.021	0.044	.329	.775	.961	.873	.189
6	.936	.597	.146	0.131	0.207	0.142	0.085	0.019	.028	.031	0.015
7	0.081	.276	.506	.217	0.342	0.811	0.812	0.083	.901	.203	.545
8	.003	.339	.669	.63	.128	0.476	0.932	0.84	0.314	.402	.823
9	0.142	.227	0.067	0.573	0.757	0.239	.61	.145	.724	0.254	1.332
10	.019	.798	.056	.962	.02	.051	.837	.922	.712	.687	.51
11	0.062	.586	.651	.495	.196	.08	0.002	.226	.427	.402	.176
12	.026	.707	.391	.03	0.19	0.065	.118	0.061	0.021	.093	0.048

Базисы аппроксимации по вариантам.

F_0 – для аппроксимации с помощью функции *genfit.*

F_1 – F_4 – для аппроксимации с помощью функции *linfit*

	F_0	F_1	F_2	F_3	F_4
	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1	x	x^2	x^3
	$a\cdot sin(b\cdot t)\cdot cos(c\cdot$	1	x	x^2	x^3
3	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	1	x	x^2	x^3
4	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1	x	x^2	x^3
5	$a\cdot sin(b\cdot t)\cdot cos(c\cdot$	1	x	x^2	x^3
	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	1	x	x^2	x^3

6					
7	$a\cdot ln(t+b)\cdot sin(c$	1	x	x^2	x^3
8	$sin(a\cdot t)\cdot cos\left(\frac{b}{t+}\right.$	1	x	x^2	x^3
9	$a\cdot ln(t+b)\cdot cos(c$	1	x	x^2	x^3
10	$a\cdot sin\left(b\cdot t\cdot e^{-c\cdot t}\right)$	1	x	x^2	x^3
11	$sin(a\cdot t)\cdot cos(ln(b$	1	x	x^2	x^3
12	$a\cdot e^{-b\cdot t}\cdot cos\left(c\cdot t^2\right)$	1	x	x^2	x^3

Контрольные вопросы

В чём состоит интерполяция функции с математической точки зрения?

Какие аргументы требуются для функции *interp*?

Что вычисляет функция *interp*?

Что вычисляет и какие аргументы использует функция *linterp*?

Какие аргументы используют функции *cspline, pspline, lspline?*

Что вычисляют функции cspline, pspline, lspline?

Что такое сплайн

В чём заключается метод наименьших квадратов?

Что вычисляет и какие аргументы использует функция *linfit*?

Что вычисляет и какие аргументы использует функция *genfit*?

6 Решение алгебраических уравнений и систем уравнений в пакете *MathCad*.

Рассмотрим простейшую задачу решения уравнений. Пусть задана некоторая действительная функция $f(x)$. Требуется найти такое значение аргумента, при котором значение функции равно нулю.

$$f(x)=0$$

Уравнение может иметь единственное решение, иметь множество решений, вовсе не иметь решений – этот вопрос нас в настоящее время не интересует. Мы верим, что решение имеется, и хотим его отыскать.

Эта задача может быть усложнена, например, расширением пространства аргументов и значений. Предполагая аргумент и значение

функции комплексными числами, мы получаем комплексное уравнение, равносильное системе двух действительных уравнений:

$$\begin{cases} f_1(x+iy)=0, \\ f_2(x+iy)=0. \end{cases}$$

где f_1 и f_2 представляют собой соответственно действительную и мнимую части значения. Ещё более общий вид системы уравнений представляет случай, когда и аргумент (неизвестное) и значение функции – вектора одинаковой размерности:

$$\begin{cases} f_1(x_1, x_2, \ldots x_n)=0, \\ f_2(x_1, x_2, \ldots x_n)=0, \\ \ldots \\ f_n(x_1, x_2, \ldots x_n)=0. \end{cases}$$

Формально все эти случаи описываются уравнением с соответствующим уточнением природы *x* и *f*.

Задача "решить уравнение" тоже может иметь различные интерпретации. Например, можно попытаться найти хотя бы один корень, не важно, какой, любой. Можно искать корень, удовлетворяющий определённым условиям, например, лежащий в заданном интервале или принадлежащий заданному множеству. Можно, наконец, задаться целью отыскать все корни уравнения.

Все эти три задачи нашли отражение в средствах пакета Маткад. Это численная функция (фактически – алгоритм) отыскания корня *root*, функция вычисления всех корней полинома *polyroots*, решающий блок *given – find*. Имеются ещё специальные средства символьного решения уравнений – функция *solve*.

Численное решение скалярного уравнения

Простейшим способом нахождения одного корня уравнения является использование функции *root*(*f*(*var*), *var*, *a*, *b*). Она возвращает значение *var*, лежащее между *a* и *b*, обращающее выражение *f*(*var*), зависящее от этой переменной, в нуль.

$$root\left(x^2-4, x, 0, 3\right) = 2$$

Здесь решается уравнение $x^2-4=0$, причём разыскивается один действительный корень, лежащий в интервале (0; 3). Второй аргумент, *x*, посто указывает, по какой переменной ищется корень, ведь в уравнении могут присутствовать и другие переменные. Решение ищется численно, методом дихотомии, поэтому на концах заданного интервала левая часть уравнения должна принимать значения разного знака. Для того, чтобы подобрать подходящий интервал, полезно построить график левой части уравнения. Это позволит дополнительно оценить количество и характер

корней и их локализацию. Эта функция может быть вызвана и другим способом, без указания интервала, в котором надо искать корень. Тогда будет использован метод простых итераций, для которого должно быть задано начальное приближение, то есть вторая переменная должна быть определена к моменту вызова.

$$x := -1$$

$$\mathrm{root}\left(x^2 - 6.25, x\right) = -2.5$$

Функция *root* может находить и комплексный корень, только при этом нельзя указывать интервал, в котором этот корень должен лежать.

$$x := 2 \qquad \mathrm{root}\left(x^2 + 4, x\right) = 2i$$

$$x := -1 \qquad \mathrm{root}\left(x^2 + 4, x\right) = -2i$$

Если корней несколько, то функция *root* найдёт какой-то один из них, какой именно, зависит от начального приближения.

Отдельная функция *Mathcad* предназначена для отыскания **всех корней многочлена**. Это функция *polyroots*. Ей требуется только один аргумент – вектор коэффициентов многочлена, причём номер элемента совпадает со степенью переменной, при которой он стоит. То есть многочлену

$$3 + 2x - 4x^2 + 5x^3$$

соответствует вектор коэффициентов

$$\begin{pmatrix} 3 \\ 2 \\ -4 \\ 5 \end{pmatrix}.$$

Пример использования этой функции:

$$C := (3 \;\; 2 \;\; -4 \;\; 5)^T \qquad \mathrm{polyroots}(C) = \begin{pmatrix} -0.537 \\ 0.668 + 0.819i \\ 0.668 - 0.819i \end{pmatrix}$$

Этой функции не требуется начального приближения.

Аналитическое решение скалярного уравнения

Аналитическое решение одного уравнения в системе *Mathcad* может быть выполнено тремя способами:

с использованием меню *Symbolics*,

вызвав специальную символьную функцию *solve*,

применив численную функцию *root* в символьном варианте.

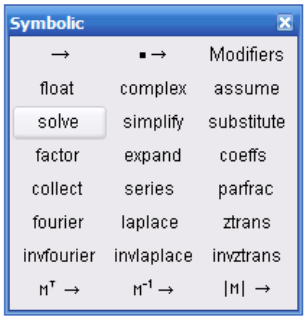
При использовании меню Symbolics в записанном уравнении или выражении надо выделить "синей галочкой" ту переменную, по которой решается уравнение,

$\sin(x + 2) - \frac{1}{2}$ или $\sin(x + 2) = \frac{1}{2}$,

после чего вызывается пункт меню ***Symbolics – Variable – Solve.*** В обоих случаях получается один и тот же результат:

$$-2 + \frac{1}{6}\cdot\pi$$

Другой вариант – использование панели инструментов *Symbolic*



Набираем функцию, чей корень хотим найти, и присоединяем к ней символьную функцию *solve*, нажав соответствующую кнопку на панели инструментов. Вот так:

$$x^2 + 6.25\,\mathrm{solve} \rightarrow \begin{pmatrix} -2.5i \\ 2.5i \end{pmatrix}$$

Символьный процессор распознал полином и нашёл все его корни, хотя такая удача и не всегда бывает. Вот ещё пример удачного решения.

$$\sin(x)\cdot\cos(2\cdot x)=\frac{\pi}{6}\ \text{solve}\ \rightarrow \left[\begin{array}{l}
-\operatorname{asin}\left[\dfrac{1}{\left(\pi-\frac{1}{3}\cdot\sqrt{9\cdot\pi^2-24}\right)^{\frac{1}{3}}}\cdot\left[\frac{1}{6}\cdot\left(3\cdot\pi-\sqrt{9\cdot\pi^2-24}\right)^{\frac{2}{3}}+\frac{1}{3}\cdot 3^{\frac{1}{3}}\right]\right] \\
\pi+\operatorname{asin}\left[\dfrac{\dfrac{\left(18\cdot\pi^2-6\cdot\pi\cdot\sqrt{9\cdot\pi^2-24}-24\right)^{\frac{1}{3}}}{6}+\dfrac{3^{\frac{1}{3}}}{3}}{\left(\pi-\dfrac{\sqrt{9\cdot\pi^2-24}}{3}\right)^{\frac{1}{3}}}\right] \\
\operatorname{asin}\left[\dfrac{3^{\frac{2}{3}}\cdot\left[2^{\frac{1}{3}}\cdot\left(3\cdot\pi^2-\pi\cdot\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}-4\right)^{\frac{1}{3}}+2-2\cdot i\cdot\sqrt{3}+2^{\frac{1}{3}}\cdot\sqrt{3}\cdot\left(3\cdot\pi^2-\pi\cdot\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}-4\right)^{\frac{1}{3}}\cdot i\right]}{12\cdot\left(3\cdot\pi-\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}\right)^{\frac{1}{3}}}\right] \\
\operatorname{asin}\left[\dfrac{3^{\frac{2}{3}}\cdot\left[2^{\frac{1}{3}}\cdot\left(3\cdot\pi^2-\pi\cdot\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}-4\right)^{\frac{1}{3}}+2-2^{\frac{1}{3}}\cdot\sqrt{3}\cdot\left(3\cdot\pi^2-\pi\cdot\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}-4\right)^{\frac{1}{3}}\cdot i+2\cdot i\cdot\sqrt{3}\right]}{12\cdot\left(3\cdot\pi-\sqrt{3}\cdot\sqrt{3\cdot\pi^2-8}\right)^{\frac{1}{3}}}\right] \\
\pi-\operatorname{asin}\left[\dfrac{1}{\left(\pi-\frac{1}{3}\cdot\sqrt{9\cdot\pi^2-24}\right)^{\frac{1}{3}}}\cdot\left[\frac{1}{12}\cdot\left(18\cdot\pi^2-6\cdot\pi\cdot\sqrt{9\cdot\pi^2-24}-24\right)^{\frac{1}{3}}+\frac{1}{6}\cdot 3^{\frac{1}{3}}-\frac{243^{\frac{1}{6}}\cdot\left(6\cdot\pi^2-2\cdot\pi\cdot\sqrt{9\cdot\pi^2-24}-8\right)^{\frac{1}{3}}}{12}\cdot i+\frac{243^{\frac{1}{6}}}{6}\cdot i\right]\right] \\
\pi-\operatorname{asin}\left[\dfrac{1}{\left(\pi-\frac{1}{3}\cdot\sqrt{9\cdot\pi^2-24}\right)^{\frac{1}{3}}}\cdot\left[\frac{1}{12}\cdot\left(18\cdot\pi^2-6\cdot\pi\cdot\sqrt{9\cdot\pi^2-24}-24\right)^{\frac{1}{3}}+\frac{1}{6}\cdot 3^{\frac{1}{3}}+\frac{243^{\frac{1}{6}}\cdot\left(6\cdot\pi^2-2\cdot\pi\cdot\sqrt{9\cdot\pi^2-24}-8\right)^{\frac{1}{3}}}{12}\cdot i-\frac{243^{\frac{1}{6}}}{6}\cdot i\right]\right]
\end{array}\right]$$

Как видно из последнего примера, символьное решение даёт не число, а выражение, как и должно быть по смыслу. Однако можно, конечно, и вычислить это выражение, например, символьной функцией *float*.

Наконец, можно находить корни символьно и использованием численной функции *root*, выполняя её символьно, через **<ctrl+".">**.

$$\operatorname{root}\left(\sin(x)\cdot\cos(2\cdot x)-\frac{\pi}{6},x,0,10\right)\rightarrow\left[\begin{array}{l}
\pi+\operatorname{asin}\left[\dfrac{\dfrac{\left[9\cdot\left(\dfrac{\sqrt{9\cdot\pi^2-24}}{3}-\pi\right)^2\right]^{\frac{1}{3}}}{6}+\dfrac{3^{\frac{1}{3}}}{3}}{\left(\pi-\dfrac{\sqrt{9\cdot\pi^2-24}}{3}\right)^{\frac{1}{3}}}\right] \\
2\cdot\pi-\operatorname{asin}\left[\frac{1}{6}\cdot\left(9\cdot\pi-3\cdot\sqrt{9\cdot\pi^2-24}\right)^{\frac{1}{3}}+\dfrac{\frac{1}{3}}{\left(\frac{1}{3}\cdot\pi-\frac{1}{9}\cdot\sqrt{9\cdot\pi^2-24}\right)^{\frac{1}{3}}}\right]
\end{array}\right]$$

Обратите внимание на отличие результатов символьного решения одного и того же уравнения двумя разными способами.

Решение системы линейных алгебраических уравнений (СЛАУ)

Численное решение СЛАУ возможно тремя способами: используя обратную матрицу, специальную функцию *lsolve*, или с помощью решающего блока *given – find*.

Все эти пути используют разные вычислительные методы и приводят порой к разным результатам. В особенности заметна разница результатов решения плохо обусловленных или вырожденных систем.

СЛАУ обычно записывают в матрично-векторной форме:

$$Ax = b,$$

где: A – матрица левых частей системы,

b – вектор правых частей,

x – неизвестный вектор.

Домножив обе части уравнения на обратную матрицу A^{-1}, получаем сразу в явном виде решение

$$x = A^{-1}b.$$

Например:

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9.1 \end{pmatrix} \qquad b := \begin{pmatrix} 2 \\ 5 \\ 2 \end{pmatrix} \qquad x := A^{-1} \cdot b \qquad x = \begin{pmatrix} -60 \\ 121 \\ -60 \end{pmatrix}$$

Тот же результат даст использование функции *lsolve*, которой требуется два аргумента – матрица левой части и вектор правой части:

$$lsolve(A, b) = \begin{pmatrix} -60 \\ 121 \\ -60 \end{pmatrix}$$

Наконец, можно воспользоваться решающим блоком, причём сделать это в двух вариантах: в матрично-векторной форме,

$$Given \;\; A \cdot x = b \qquad Find(x) = \begin{pmatrix} -60 \\ 121 \\ -60 \end{pmatrix}$$

или покоординатно:

$given$

$$a11\cdot x1 + a12\cdot x2 + a13\cdot x3 = b1$$

$$a21\cdot x1 + a22\cdot x2 + a23\cdot x3 = b2$$

$$a31\cdot x1 + a32\cdot x2 + a33\cdot x3 = b1$$

$$find(x1, x2, x3) = \begin{pmatrix} -60 \\ 121 \\ -60 \end{pmatrix}$$

Порядок выполнения лабораторной работы

Упражнение 1.

Построить график заданной функции и локализовать четыре её корня. Уточнить корни с помощью функции *root* и с помощью решающего блока. Проверить найденные корни подстановкой в заданную функцию. Сравнить решения, полученные разными способами. Оценить точность решения.

Варианты исследуемой функции.

	F	a	b	c
	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1 .088	0 .97	1 .038
	$a\cdot sin(b\cdot t)\cdot cos(c\cdot t)$	1 .764	1 .514	4 .104
3	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	1 .916	1 .238	0 .862
4	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1 .125	1 .497	1 .687
5	$a\cdot sin(b\cdot t)\cdot cos(c\cdot t)$	0 .978	1 .188	1 .47
6	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	1 .738	0 .388	1 .443
7	$a\cdot ln(t+b)\cdot sin(c\cdot t)$	1 .581	0 .908	1 .637
8	$sin(a\cdot t)\cdot cos\left(\frac{b}{t+c}\right.$	1 .994	1 .083	0 .703
9	$a\cdot ln(t+b)\cdot cos(c\cdot t$	1 .262	1 .528	4 .19
10	$a\cdot sin\left(b\cdot t\cdot e^{-c\cdot t}\right)$	0 .606	0 .421	2 .633
11	$sin(a\cdot t)\cdot cos(ln(b\cdot t$	1 .696	1 .035	2 .188
12	$a\cdot e^{-b\cdot t}\cdot cos\left(c\cdot t^2\right)$	0 .814	1 .43	4 .752
	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1	0	2

3		.387	.38	.793
4	$a\cdot sin(b\cdot t)\cdot cos(c\cdot t)$	0 .117	0 .369	2 .411
5	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	0 .624	1 .416	4 .25
6	$a\cdot e^{-b\cdot t}\cdot sin(c\cdot t)$	1 .217	0 .91	2 .335
7	$a\cdot sin(b\cdot t)\cdot cos(c\cdot t)$	1 .691	1 .937	4 .916
8	$a\cdot e^{-b\cdot t}\cdot cos(c\cdot t)$	1 .021	0 .391	3 .722
9	$a\cdot ln(t+b)\cdot sin(c\cdot t)$	1 .513	1 .661	1 .06

Упражнение 2.

Найти все корни заданного полинома. Проверить подстановкой.

Варианты коэффициентов многочлена

	a_0	a_1	a_2	a_3	a_4	a_5
	3.394	-3.862	1.281	-1.85	1.538	4.517
	9.122e-3	2.522	-0.484	-1.942	-3.952	-4.722
3	-4.725	0.434	0.978	-3.914	-2.729	-4.441
4	0.726	-0.633	3.548	3.512	4.2	-3.684
5	0.313	1.962	1.248	-3.451	1.628	3.643
6	3.43	-0.633	0.657	-4.207	-0.075	2.216
7	1.576	0.779	-3.157	1.41	-0.032	-4.853
8	3.421	1.287	0.551	0.451	0.093	2.079
9	-3.901	0.041	-2.571	-0.91	1.882	-2.824
10	-1.859	1.958	1.047	-0.344	1.063	-3.311
11	-2.139	-3.1	0.846	-3.473	-4.941	-1.59
12	-3.597	-3.216	-0.056	2.38	-3.994	-1.324

3	3.346	-0.425	2.407	3.267	3.632	3.024
4	1.002	-4.025	1.204	3.733	2.473	0.265
5	-2.473	-4.056	3.045	-1.999	-1.199	2.982
6	-4.984	4.315	0.76	-3.728	0.528	-3.552
7	3.062	3.946	4.115	2.85	4.557	-0.976
8	-2.894	-2.727	2.277	1.096	-3.234	-3.634
9	0.532	-0.893	1.678	-4.277	-3.684	-4.338

Упражнение 3.

Тремя способами решить систему линейных алгебраических уравнений. Проверить подстановкой. Сравнить решения, полученные разными методами.

Варианты коэффициентов СЛАУ

	11	12	13	21	22	23	31	32	33	1	2	3
3												
4												
5												
6												
7												
8												
9												
10												
11												

12												
3												
4												
5												
6												
7												
8												
9												

Упражнение 4.

Вычислить с использованием решающего блока обратную матрицу от матрицы левых частей из упражнения 3. Сравнить с результатом, полученным штатным инструментом *Mathcad*. Проверить оба результата подстановкой.

Контрольные вопросы

1. Какие три средства имеет Mathcad для решения систем линейных алгебраических уравнений?
2. Перечислите отличия в использовании решающего блока и функции *lsolve* при решении систем линейных алгебраических уравнений.
3. Можно ли и как вычислить обратную матрицу с помощью функции *lsolve*.
4. Какие аргументы требуются функции *lsolve*?

7 Решение ОДУ в системе *MathCad*.

Использование решающего блока

Mathcad содержит несколько различных инструментов для решения обыкновенных дифференциальных уравнений (ОДУ) или систем ОДУ. Функция ***Odesolve*** используется в составе решающего блока, в котором описана решаемая система в произвольной форме, использующей обычную математическую нотацию. Кроме того имеется ещё несколько функций, которые могут использоваться самостоятельно, без решающего блока и требуют специальной записи решаемой системы.

Дифференциальное уравнение записывается внутри решающего блока в традиционной форме. Рассмотрим колебательное звено с параметрами

$T = 0.1$, $\xi = 0.2$, $K = 1$, $f(t)=1(t)$,

и входным сигналом $f(t) = sin(t)$. Выходной сигнал такого звена описывается ОДУ

$$T^2 \cdot x''(t) + 2 \cdot \xi \cdot T \cdot x'(t) + x(t) = K \cdot f(t)$$

Решение этого уравнения можно получить с помощью такого решающего блока:

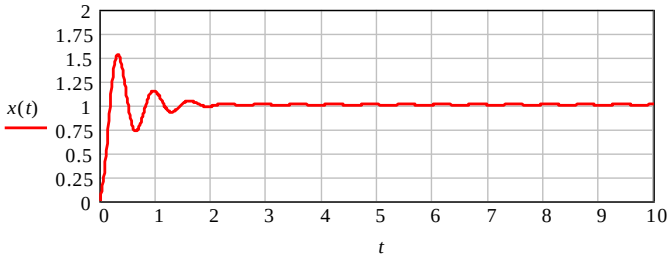
$given$

$T^2 \cdot x''(t) + 2 \cdot \xi \cdot T \cdot x'(t) + x(t) = K \cdot f(t)$

$x(0) = 1 \qquad x'(0) = 0$

$x := odesolve(t, 10)$

Внутри блока должно использоваться булевское равенство, ***ctrl***+"=". Для того, чтобы изобразить символ производной, штрих, надо нажать ***ctrl***+F7. Аргументами функции *Odesolve* в нашем случае являются имя свободной переменной (у нас это время *t*), и правая граница интервала, на котором решается уравнение (10). Вычисленным результатом, *x*, будет функция одной переменной, являющаяся решением содержащейся внутри блока задачи Коши (дифференциальное уравнение с начальными условиями).



Эта функция не является аналитическим выражением, для вычисления её значения *Mathcad* формирует численную процедуру, использующую численный алгоритм интегрирования дифференциального уравнения. Этим алгоритмом до некоторой степени можно управлять. Выполнив над словом *odesolve* **rightclick** (щелчок правой кнопкой мыши), можно выбрать из выпавшего меню один из 3-х вариантов: *Fixed*, *Adaptive*, *Stiff*.

Fixed – интегрирование методом Рунге-Кутты с постоянным (фиксированным) шагом,

Adaptive – интегрирование методом Рунге-Кутты с переменным (настраиваемым, адаптивным) шагом,

Stiff – интегрирование специальным методом, ориентированным на "жёсткие" системы, при этом игнорируются быстрые движения.

Обычно оптимальным является выбор настраиваемого шага интегрирования (*Adaptive*). Другие методы целесообразны в некоторых особых случаях.

Интервал, на котором определено найденное решение, задаётся, с одной стороны, начальными условиями (в нашем случае – 0), а с другой – аргументом функции *odesolve* (в нашем случае – 10). Вне этого интервала функция не определена, и попытка вычислить её значение приведёт к ошибке.

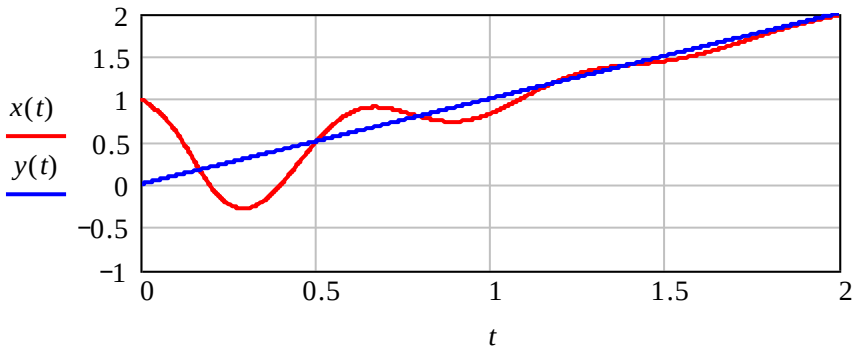
Дифференциальное уравнение может быть записано и в виде системы, причём не обязательно в 1-й нормальной форме Коши. Пусть перед колебательным звеном стоит интегратор с выходом *y*.

Given

$$T^2 \cdot x''(t) + 2 \cdot \xi \cdot T \cdot x'(t) + x(t) = y(t) \qquad x'(0) = 0 \qquad y(0) = 0$$

$$y'(t) = f(t) \qquad x(0) = 1$$

$$\begin{pmatrix} x \\ y \end{pmatrix} := Odesolve\left[\begin{pmatrix} x \\ y \end{pmatrix}, t, 2\right]$$



При этом несколько изменяется вызов функции *Odesolve*. Первым аргументом должен быть вектор, содержащий имена вычисляемых функций. Их должно быть столько же, сколько дифференциальных уравнений содержит блок. Соответственно и результатом будет вектор, состоящий из имён функций скалярного аргумента.

Функция *odesolve* может принять и ещё один аргумент, количество промежуточных точек решения. Задание слишком маленького значения этого аргумента всегда приводит к ошибкам в решении.

Прямой вызов решателей

Другой способ решения ОДУ в *Mathcad* – прямой вызов решателей:

Rkfixed(x_0, t_1, t_2, *npoints*, *D*) – интегрирование методом Рунге-Кутты с постоянным шагом,

Rkadapt(x_0, t_1, t_2, *npoints*, *D*) – интегрирование методом Рунге-Кутты с настраиваемым (адаптивным) шагом,

Radau(x_0, t_1, t_2, *npoints*, *D*) – интегрирование "жёстких" систем.

Во всех этих функциях набор параметров одинаков:

x_0 – вектор начальных условий, соответствующих моменту времени t_1,

t_1, t_2 – концы интервала свободной переменной (у нас это время), на котором вычисляется решение,

npoints – количество узлов решения на заданном интервале,

D – имя вектор-функции правых частей системы ОДУ, записанной в первой нормальной форме Коши. Эта функция должна, в свою очередь, иметь аргументами свободную переменную (время) и вектор состояния.

Например, для последней рассмотренной системы, имеющей 3-й порядок, потребуется такая запись:

$$F(t,x) := \begin{pmatrix} x_1 \\ \dfrac{x_2}{T^2} - \dfrac{2\cdot\xi}{T}\cdot x_1 - \dfrac{1}{T^2}\cdot x_0 \\ f(t) \end{pmatrix} \qquad x0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$X := Rkadapt(x0, 0, 2, 100, F)$$

$X =$

	0	1	2	3
0	0	1	0	0
1	0.02	0.981	-1.89	0.02
2	0.04	0.926	-3.524	0.04
3	0.06	0.842	-4.859	0.06
4	0.08	0.734	-5.867	0.08
5	0.1	0.609	-6.534	0.1
6	0.12	0.475	-6.861	0.12
7	0.14	0.337	-6.862	0.14
8	0.16	0.202	-6.562	0.16

Результатом является матрица, содержащая *npoints* строк, в каждой строке на первом месте стоит значение свободной переменной (время), а следом – соответствующие этому моменту времени значения переменных состояния, в том порядке, в каком они записаны в определении функции *F*. Естественно, что результат будет тем же, что и в предыдущем примере.

Задание на работу

Задана передаточная функция системы (см. Варианты задания). Начальные условия принимаются нулевыми. На вход поступает 2 варианта сигнала – ступенчатый и гармонический. Время интегрирования подберите исходя из длительности переходных процессов. Требуется:

1. Записать одно дифференциальное уравнение и решить его для обоих вариантов входного сигнала с использованием решающего блока. Выбрать метод решения. Построить графики.
2. Представить систему в виде последовательного соединения элементарных звеньев, составить соответствующую систему дифференциальных уравнений и решить с использованием решающего блока. Построить графики.
3. Представить систему в виде параллельного соединения элементарных звеньев, составить соответствующую систему дифференциальных уравнений и решить с использованием решающего блока. Построить графики.
4. Сформировать системы дифференциальных уравнений в 1-й нормальной форме Коши, соответствующих одному дифференциальному уравнению, последовательному и параллельному соединению элементарных звеньев и решить их с использованием прямого вызова решателей. Выбрать метод решения. Построить графики.
5. Сравнить 4 описанных выше варианта решения по трудоёмкости и удобству применения. Сделать выводы о том, когда какой метод предпочтительней использовать.

Варианты задания.

Передаточная функция имеет вид:

$$W(p) = \frac{b_1 p + b_0}{a_3 p + a_2 p + a_1 p + a_0}$$

Входной сигнал:

$$y(t) = \sin(\omega \cdot t + \varphi)$$

№ варианта	a_0	a_1	a_2	a_3	b_0	b_1	[illegible]	[illegible]
1.	0.002	0.091	0.06	0.021	.23	.302	.015	.842
2.	0.009	0.092	0.095	0.068	.892	.363	.055	.429
3.	0.009	0.016	0.04	0.037	.768	.128	.047	.731
4.	0.001	0.057	0.019	0.012	.98	.007	.031	.123
5.	0.002	0.094	0.058	0.021	.591	.653	.035	.941
6.	0.008	0.058	0.063	0.066	.697	.257	.061	.518
7.	0.	0.	0	0.	[illegible]	[illegible]	[illegible]	[illegible]

	002	026	.006	069	.1	.854	.098	.367
8.	0	0.036	0.027	0.009	.356	.604	.016	.372
9.	0.001	0.08	0.09	0.004	.485	.354	.071	.822
10.	0.003	0.008	0.073	0.031	.93	.157	.078	.298
11.	0	0.059	0.04	0.015	.927	.606	.063	.104
12.	0.002	0.021	0.09	0.008	.884	.95	.069	.541
13.	0.002	0.065	0.039	0.073	.092	.97	.057	.134
14.	0.009	0.061	0.036	0.07	.607	.503	.055	.642
15.	0.002	0.045	0.075	0.066	.037	.629	.025	.22
16.	0.006	0.04	0.074	0.011	.575	.746	.05	.825

Контрольные вопросы

1. В чём отличие функций Rkfixed, Rkadapt, Radau, Odesolve?
2. Как вставить символ дифференцирования в решающем блоке?
3. Что означает термин «жёсткая система»?
4. Как представить передаточную функцию в виде суммы элементарных дробей?
5. Как представить передаточную функцию в виде произведения элементарных дробей?
6. Как вычислить характеристические числа передаточной функции?
7. Какие численные методы интегрирования ОДУ может реализовать функция odesolve и как сожно управлять выбором конкретного метода?
8. Что оказывается результатом решения, число, массив, фунукция или иная структура?

8 Анализ линейной системы управления средствами системы *MathCad*.

ВНИМАНИЕ! В этой работе Вы будете использовать систему из Вашей курсовой работы по теории автоматического управления! Требуется построить частотные характеристики разомкнутой системы, временные

характеристики замкнутой системы без коррекции, определить для неё критерии точности и качества процесса регулирования.

Ниже в качестве примера используется некоторая абстрактная система.

Пусть передаточная функция разомкнутой системы имеет вид

$$W(s) := \frac{K}{s\cdot\left(T^2\cdot s^2 + 2\cdot\xi\cdot T\cdot s + 1\right)}$$

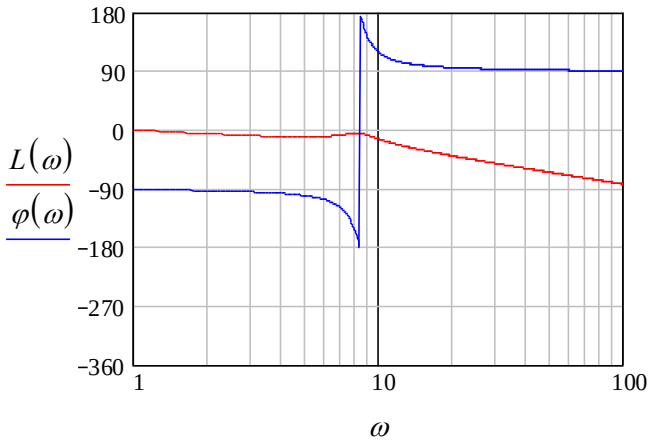
Построим частотные характеристики этой системы, для чего подставим в передаточную функцию jω вместо s. Амплитудно-частотная характеристика (АЧХ) равна модулю комплексной частотной характеристики, а фазочастотная (ФЧХ) - её аргументу, выраженному в градусах.

$$A(\omega) := \left|W(j\cdot\omega)\right|, \qquad \varphi(\omega) := arg\left(W(j\cdot\omega)\right)\cdot\frac{180}{\pi}$$

Обычно пользуются логарифмической амплитудной характеристикой (ЛАЧХ), измеряемой в децибелах (дБ), которая вычисляется по формуле

$$L(\omega) := 20\cdot log\left(A(\omega)\right)$$

Графики частотных характеристик принято строить в логарифмическом масштабе по оси частот, тогда они называются логарифмическими амплитудно-фазовыми частотными характеристиками (ЛАФЧХ). Логарифмический масштаб по оси абсцисс можно задать в настройках графика (*Logscale*). При построении графика получаем следующую картину:



Фазовая характеристика отображена неправильно. Это связано с неоднозначностью математической функции «аргумент» и, соответственно, ограниченностью области значений маткадовской функции *arg*.

Чтобы избежать этой неприятности, изменим способ вычисления фазы. Зададим массив частот длиной *n*, для которых будет вычисляться фаза. Частоту в массиве будем изменять не линейно, а логарифмически, чтобы точки равномерно располагались на логарифмической шкале.

$n := 100 \qquad k := 1..n \qquad \omega_{min} := 0.1 \qquad \omega_{max} := 100 \qquad \omega_0 := \omega_{min}$

$$h := \sqrt[n]{\frac{\omega_{max}}{\omega_{min}}} \qquad \omega_k := \omega_{k-1} \cdot h$$

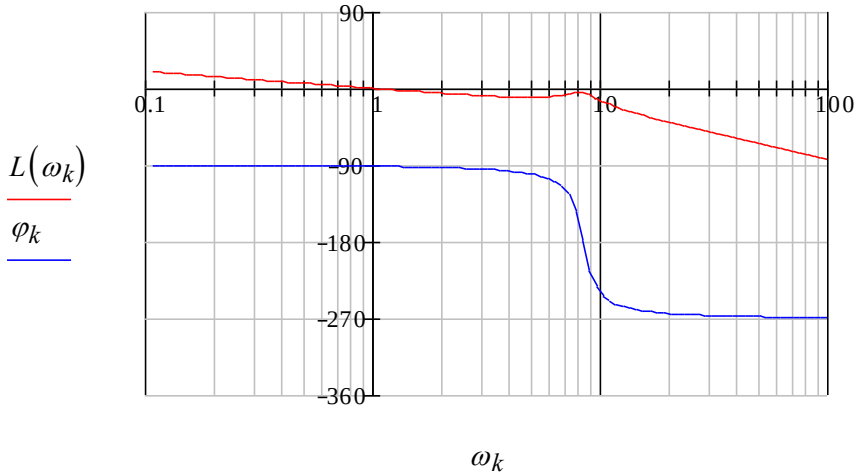
Будем вычислять фазу не непосредственно, а через приращения, воспользовавшись тем, что аргумент отношения равен разности аргументов:

$$\arg\left(\frac{z_1}{z_2}\right) = \arg(z_1) - \arg(z_2),$$

откуда получаем

$$\varphi_0 = \arg(W(j\omega_0)), \qquad \varphi_k = \varphi_{k-1} + \arg\left(\frac{W(j\omega_k)}{W(j\omega_{k-1})}\right) \cdot \frac{180}{\pi}, \qquad k = 1, 2 \ldots n.$$

Теперь график принимает правильный вид:



Для построения временных характеристик (весовая функция и переходная характеристика) надо получить обратное преобразование Лапласа от передаточной функции замкнутой системы.

$$W_3(s) := \frac{W(s)}{1 + W(s)}$$

$$W_3(s) \ invlaplace, s \ \rightarrow 1.00444530639369067 \cdot exp(-1.00964449474404898755250 \cdot t) \ldots$$

Такая точность замечательна, но не слишком удобна, поэтому ограничим её четырьмя знаками

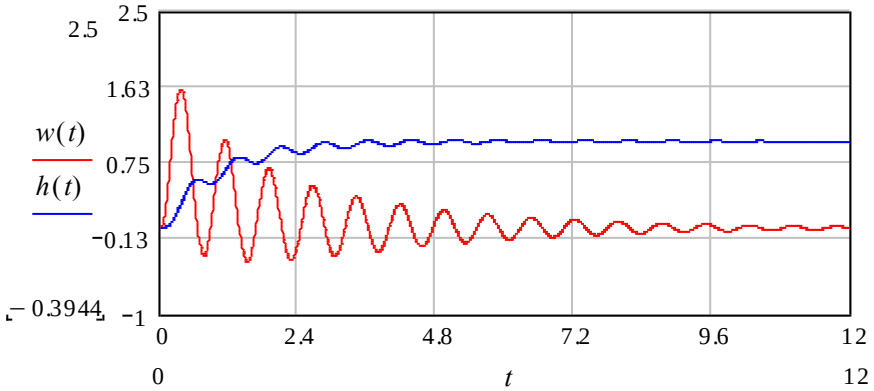
$$w(t) := W_3(s) \left| \begin{array}{l} invlaplace, s \\ simplify \\ float, 4 \end{array} \right. \rightarrow 1.004 \cdot exp(-1.010 \cdot t) - 1.004 \cdot exp(-.3285 \cdot t) \cdot cos(8.287 \cdot t) + 8.256 \cdot 10^{-2} \cdot exp(-.3285 \cdot t) \cdot sin(8.287 \cdot t)$$

Переходная характеристика равна первообразной от весовой функции. Вычислим интеграл также символьно с четырьмя значащими цифрами.

$$h(t) := \int_0^t w(\tau)\, d\tau \;\Big|\; \begin{matrix} simplify \\ float, 2 \end{matrix} \rightarrow -.99 \cdot exp(-1.0 \cdot t) - 5.2 \cdot 10^{-3} \cdot exp(-.33 \cdot t) \cdot cos(8.3 \cdot t) - .12 \cdot exp(-.33 \cdot t) \cdot sin($$

....

Построим графики полученных временных характеристик.



Используя передаточную функцию замкнутой системы:

$$W_3(s)\ simplify \rightarrow \frac{625.}{9.\cdot s^3 + 15.\cdot s^2 + 625.\cdot s + 625.}$$

напишем дифференциальные уравнения замкнутой системы:

$$Given$$

$$9 \cdot x'''(t) + 15 \cdot x''(t) + 625 \cdot x'(t) + 625 \cdot x(t) = 625$$

$$x(0) = 0 \qquad x'(0) = 0 \qquad x''(0) = 0$$

$$x := Odesolve(t, 10)$$

Чтобы поставить штрих над x-ом, надо нажать **<ctrl> + <F7>**

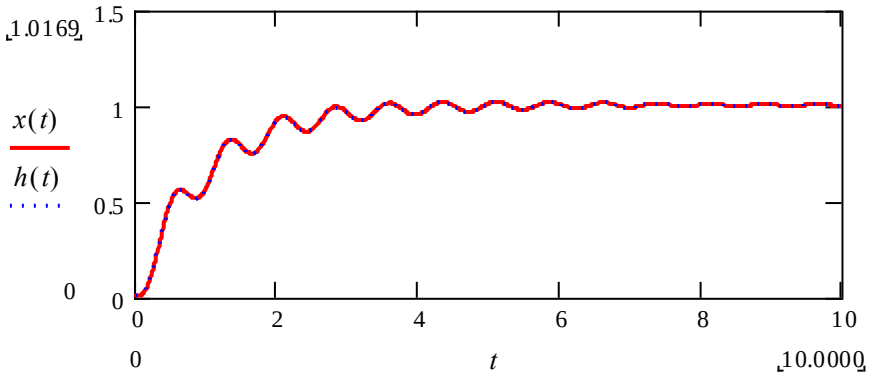


Рисунок показывает, что переходная характеристика, вычисленная как численно, так и символьно, имеет один и тот же (практически) вид.

Задание

1. Постройте ЛАФЧХ и годограф Найквиста неизменяемой части своей системы из задания на курсовую работу по ТАУ.

2. Определите частоту среза и запасы устойчивости, используя инструменты трассировки (*Trace*) графиков и аналитически.

3. Замкните систему и постройте частотные и временные характеристики замкнутой системы. Вычислите корни характеристического уравнения. Сделай те вывод об устойчивости системы.

4. Постройте временные характеристики замкнутой системы, как с помощью символьных преобразований, так и численно, решив дифференциальные уравнения, описывающие эту систему.

5. Сделайте вывод о необходимости коррекции системы.

Контрольные вопросы

1. Как обеспечить логарифмический масштаб по оси частот?
2. Как получить коэффициенты характеристического многочлена замкнутой системы?
3. Перечислите символьные операции, позволяющие получить временные характеристики системы по её передаточной функции.
4. Как поставить в формуле символ дифференцирования - штрих?
5. Какие численные методы интегрирования ОДУ может реализовать функция odesolve и как сожно управлять выбором конкретного метода?
6. Что оказывается результатом решения, число, массив, фунукция или иная структура?

9 Синтез линейной системы управления средствами системы *MathCad*.

Динамические свойства системы практически полностью определяются расположением корней характеристического многочлена на комплексной плоскости. Степень устойчивости η определяет быстродействие системы (время регулирования). Колебательность μ определяет качество переходного процесса - перерегулирование и колебательность. Наконец, статическую точность определяет коэффициент передачи разомкнутой системы k.

Напомним основные определения и зависимости.

Степенью устойчивости η устойчивой системы называется расстояние от мнимой оси до ближайшего к ней корня. Иначе её можно определить как наибольшую действительную часть характеристических чисел, взятую с противоположным знаком. У неустойчивой системы эта характеристика отсутствует или, в зависимости от определения, отрицательна. Степень устойчивости связана с самым "медленным" корнем. Именно время затухания соответствующей ему составляющей свободного движения и определяет общее время затухания переходного процесса, то есть время регулирования. Это время приближённо оценивается как

$$t_p = \frac{3}{\eta}$$

Поэтому степень устойчивости обычно стремятся по возможности увеличить.

Колебательностью μ называют наибольшее отношение мнимой части характеристического числа к его действительной части. Обычно переходный процесс затухает приблизительно за μ колебаний. Поэтому μ обычно стремятся уменьшить.

Если известны все характеристические числа λ_i (замкнутой) системы, то степень устойчивости и колебательность можно вычислить вот так:

$$\eta = max\left(Re\left(\lambda_i\right)\right) \qquad \mu = max\left(\frac{Im\left(\lambda_i\right)}{Re\left(\lambda_i\right)}\right)$$

При проектировании САУ обычно формулируются требования именно по этим трём параметрам - статической точности, времени регулирования и колебательности. Можно также поставить задачу оптимизации, рассматривая одну из характеристик как критерий оптимизации, а остальные - как ограничения. Например, потребовать наилучшей точности при выполнении условий по быстродействию и колебательности.

Одним из средств коррекции САУ, то есть обеспечения устойчивости и требуемого качества регулирования, является использование ПД-регулятора, то есть регулятора с передаточной функцией

$$C\left(p, k, k_n\right) := \left(k + k_n \cdot p\right).$$

Это передаточная функция последовательного корректирующего фильтра. **В Вашей системе она может иметь и другой вид**!

Пусть передаточная функция неизменяемой части разомкнутой системы равна

$$W(p) = \frac{Q(p)}{R(p)}$$

Тогда передаточная функция замкнутой системы будет иметь вид

$$W_3(p) := \frac{C\left(p, k, k_n\right) \cdot Q(p)}{C\left(p, k, k_n\right) \cdot Q(p) + R(p)}$$

Возьмём для примера в качестве неизменяемой части систему

$$T \equiv 0.05 \qquad \xi \equiv 0.3 \qquad Q(p) \equiv 15 \qquad R(p) \equiv \left(T^2 \cdot p^2 + 2 \cdot \xi \cdot T \cdot p + 1\right) \cdot (p \cdot 0.001 + 1) \cdot p.$$

Получим характеристический многочлен замкнутой системы, выделив знаменатель передаточной функции замкнутой системы.

$$D\left(p, k, k_n\right) := denom\left(W_3(p)\right) \left| \begin{matrix} collect, p \\ float, 4 \end{matrix} \right. \rightarrow 2.5 \cdot 10^{-6} \cdot p^4 + 2.530 \cdot 10^{-3} \cdot p^3 + 3.1 \cdot 10^{-2} \cdot p^2 + \left(15. \cdot k_n + 1.\right) \cdot p + 15. \cdot k$$

Вычислим коэффициенты v характеристического многочлена D нескорректированной системы

$$v(k,k_n) := D(p,k,k_n) \left| \begin{matrix} coeffs, p \\ float, 6 \end{matrix} \right. \rightarrow \begin{pmatrix} 15.\cdot k \\ 15.\cdot k_n + 1. \\ 3.10000\cdot 10^{-2} \\ 2.53000\cdot 10^{-3} \\ 2.50000\cdot 10^{-6} \end{pmatrix}$$

и определим его корни для нескорректированной системы, то есть при k=1 k_n =0.

$$polyroots(v(1,0))^T = (-999.994 \quad -14.143 \quad 1.069 - 20.569i \quad 1.069 + 20.569i)$$

Нескорректированная система неустойчива.

Определим теперь функции, вычисляющие степень устойчивости и колебательность:

$$\eta(k,k_n) := -max(Re(polyroots(v(k,k_n)))), \qquad \eta(1,0) = -1.069 .$$

$$\mu(k,k_n) := max\left(\overrightarrow{\left(\frac{Im(polyroots(v(k,k_n)))}{Re(polyroots(v(k,k_n)))}\right)}\right), \qquad \mu(1,0) = 19.248 .$$

При вычислении колебательности деление должно выполняться не над векторами, а поэлементно, на что и указывает символ "векторизации" - стрелка над дробью. Простейший способ её поставить - выделить соответствующее выражение и нажать **<ctrl>** + < – >.

Теперь оформим задачу оптимизации. Целью будет максимизация коэффициента передачи k, то есть повышение точности системы в установившемся режиме. Ограничения зададим такие:

$$\eta > 3 \qquad \mu < 2$$

Это означает затухание переходного процесса менее чем за секунду и затухание колебаний за 2 периода.

Определим функцию качества $f(k,k_n)$ и начальные значения искомых (оптимизируемых) параметров:

$$f(k,k_n) := k \qquad k := 1, \qquad k_n := 0.$$

Теперь сформируем решающий блок:

given

$\eta(k, k_n) > 3 \qquad \mu(k, k_n) < 2$

$$\begin{pmatrix} k \\ k_n \end{pmatrix} := Maximize(f, k, k_n)$$

После завершения оптимизации получаются такие значения:

$k = 0.057 \qquad k_n = -0.047 \qquad \eta(k, k_n) = 4.074$

$\mu(k, k_n) = 2$

Надо заметить, что в жизни решение не всегда выглядит столь гладко. Для того, чтобы получить какой-то результат, надо без устали подбирать начальные приближения, менять, быть может, ограничения, экспериментировать с методом оптимизации. И если даже решения найти не удалось, это ещё не значит, что его действительно не существует. Просто его не удалось отыскать.

Проанализируем полученные результаты.

Задачу решить удалось, но один из коэффициентов регулятора оказался отрицательным. Поскольку это коэффициент при производной, никакой беды в этом нет. Положительная обратная связь по производной придаёт некоторые форсирующие свойства системе, в то время как отрицательная обратная связь - демпфирует.

Значение критерия получилось довольно маленьким. Однако в системе имеется астатизм 1-го порядка, так что статическая точность так или иначе будет обеспечена.

Колебательность получилась на границе допустимого. Значит, именно она и является лимитирующим фактором оптимизации. Расширив границы допустимой колебательности можно надеяться получить более высокий коэффициент передачи. Проверим это.

Given

$\eta(k, k_n) > 3$

$\mu(k, k_n) < 5$

$$\begin{pmatrix} k \\ k_n \end{pmatrix} := Maximize(f, k, k_n) \qquad k = 0.275 \qquad k_n = 3.891 \times 10^{-3}$$

$\eta(k, k_n) = 3.867 \qquad \mu(k, k_n) = 5$

Можно попробовать другую постановку - минимизировать колебательность при ограничениях на точность и быстродействие.

Given

$\eta(k, k_n) > 3$

$k > 1.3$

$\begin{pmatrix} k \\ k_n \end{pmatrix} := Minimize(\mu, k, k_n)$ $\quad k = 1.3 \quad k_n = 0.218$

$\eta(k, k_n) = 3.01 \quad \mu(k, k_n) = 13.532$

Исследователю самому придётся решить, какой вариант предпочтительней.

Задание:

1. Оптимизируйте свою систему из задания на курсовую работу по ТАУ. За варьируемые параметры возьмите параметры последовательного корректирующего фильтра, построенного методом ЖЛАХ в курсовой по ТАУ.

4. Постройте временные характеристики замкнутой системы как с помощью символьных преобразований, так и численно, решив дифференциальные уравнения, описывающие эту систему.

5.

Контрольные вопросы

1. Как определить степень устойчивости? Какие функции Маткада при этом используются?
2. Как обеспечить поэлементное деление векторов? Какой горячей клавише это соответствует?
3. Какие функции решения ОДУ кроме *odesolve* имеются в Маткаде и как можно управлять точностью решения численного метода?
4. Зачем используется функция *collect*? Что будет, если не использовать функцию *float* при получении характеристического многочлена и его корней? Зачем она использована в примере?

10 Изучение интерфейса системы *Matlab*.

Теоретические сведения

После запуска MATLAB на экране появляется основное окно системы MATLAB, показанное на рис. 10.1 Обычно это окно раскрыто не полностью и занимает часть рабочего стола. Вы можете раскрыть окно полностью, щелкнув на средней из трех кнопок, расположенных в конце титульной (верхней) строки окна. Левая кнопка сворачивает окно в кнопку с именем приложения, помещаемую в панель задач Windows, а правая закрывает окно и прекращает работу с MATLAB. Система готова к проведению вычислений в *командном режиме.*

Для уточнения версии системы следует вывести окно с информацией о системе (команда About MATLAB (О MATLAB) в меню Help (Помощь). Это окно представлено на рис. 10.2. Из него видно, что запущена версия 6.0 (R12) от 22 сентября 2000 г. Поскольку номер лицензии имеет конфиденциальный характер, вместо него на рисунке показан 0.

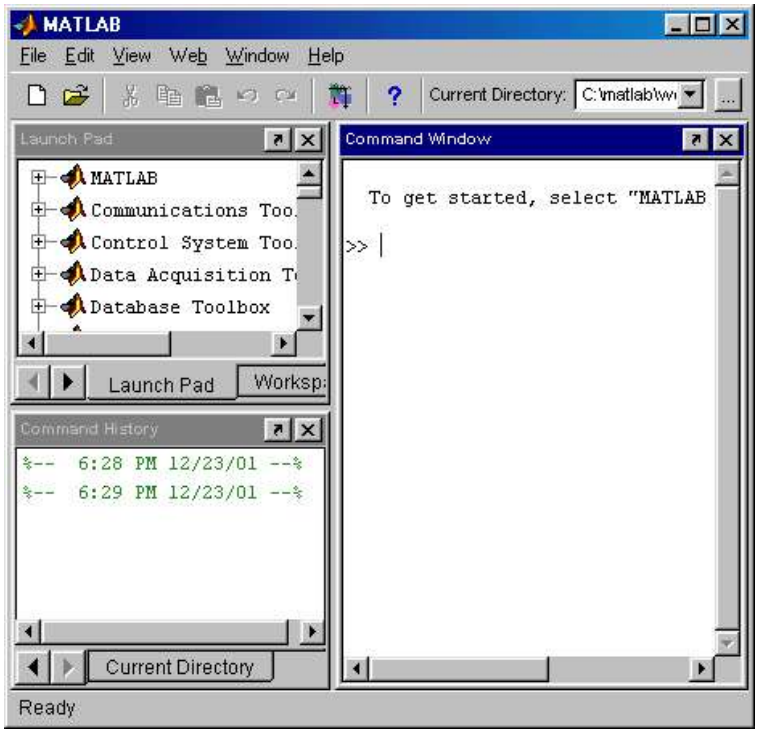


Рис..10.1. Окно системы MATLAB после запуска и выполнения простых вычислений

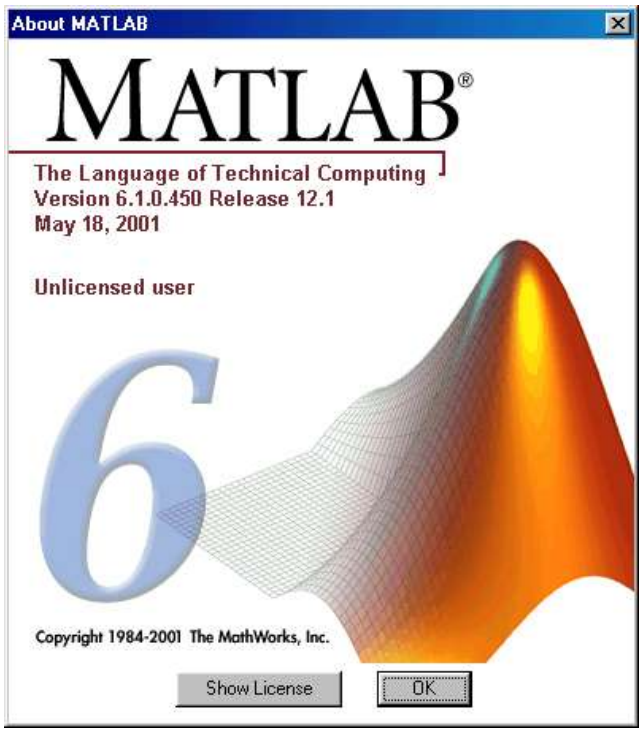


Рис. 10.2. Окно с логотипом системы MATLAB 6.1

Полезно знать, что в начале запуска автоматически выполняется команда matlabrc, которая исполняет загрузочный файл matlabrc.m и файл startup.m, если таковой существует. Эти файлы выполняют начальную настройку терминала системы и задают ряд ее параметров. В частности, могут быть заданы пути доступа к другим файлам, необходимым для корректной работы системы MATLAB. Таким образом, опытные пользователи могут выполнить настройку системы под свои запросы. Однако в большинстве случаев особой необходимости в этом нет. Поскольку указанные файлы имеют текстовый формат, их легко просмотреть с помощью какого-либо текстового редактора или с помощью команды type в командном режиме работы MATLAB.

Сеанс работы с MATLAB принято именовать *сессией* (session). Сессия, в сущности, является текущим документом (протоколом), отражающим работу пользователя с системой MATLAB. В ней имеются строки ввода, вывода и сообщений об ошибках. Входящие в сессию определения переменных и функций, расположенные в рабочей области памяти, но не саму сессию, можно записать на диск (файлы формата .mat), используя команду save (Сохранить). Команда load (Загрузить) позволяет считать с диска данные рабочей области. Фрагменты сессии можно оформить в виде дневника с помощью команды diary (Дневник).

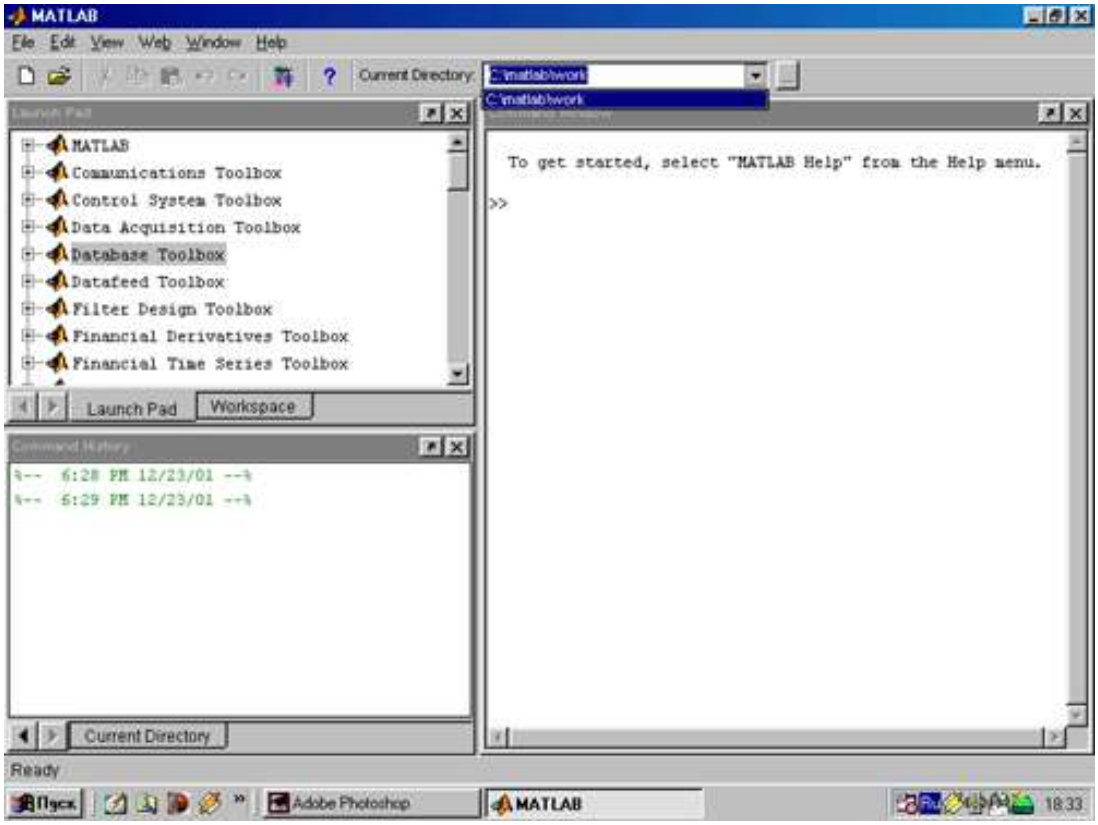


Рис. 9.3. Упрощенный интерфейс системы MATLAB 6.0

Управлять видом рабочего окна можно командой меню View> Desktop Layout (Интерфейс). Там же имеются и другие возможности модификации вида интерфейса системы MATLAB 6.0.

Основная работа происходит в командном окне (Command Window) в командном режиме, в котором действует простейший строчный редактор. Его команды перечислены в табл. 10.1.

Таблица 10.1. Команды строчного редактора MATLAB

Комбинация клавиш	Назначение
Ctrl+b	Перемещение курсора вправо на один символ
Ctrl+f	Перемещение курсора влево на один символ
Ctrl +r	Перемещение курсора вправо на одно слово
Ctrl +1	Перемещение курсора влево на одно слово
Home или Ctrl+a	Перемещение курсора в начало строки
End или Ctrl +e	Перемещение курсора в конец строки
Ctrl+pu Ctrl+n	Перелистывание предыдущих команд вверх или вниз для подстановки в строку ввода
Del или Ctrl+d	Стирание символа справа от курсора
Ctrl+h	Стирание символа слева от курсора
Ctrl+k	Стирание до конца строки
Esc	Очистка строки ввода
Ins	Включение/выключение режима вставки
PgUp	Перелистывание страниц сессии вверх
PgDn	Перелистывание страниц сессии вниз

Эти возможности кажутся примитивными, но позволяют пользователю быстро работать в стиле первых версий MATLAB для MS-DOS. Они обеспечивают важное свойство новых версий систем — их совместимость со старыми версиями в части преемственности навыков работы. Позже вы увидите, что в новых версиях есть вполне современный редактор со средствами отладки создаваемых документов — m-файлов.

Обратите особое внимание на применение клавиш вверх и вниз. Они используются для подстановки после маркера строки ввода » ранее введенных строк, например для их исправления, дублирования или дополнения. При этом указанные клавиши обеспечивают перелистывание ранее введенных строк снизу вверх или сверху вниз. Такая возможность существует благодаря организации специального стека, хранящего строки с исполненными ранее командами.

Команды управления окном

Полезно сразу усвоить некоторые команды управления окном командного режима:

- clc — очищает экран и размещает курсор в левом верхнем углу пустого экрана.
- home — возвращает курсор в левый верхний угол окна.
- echo <file_name> on — включает режим вывода на экран текста Script-файла (файла-сценария).
- echo <file_name> off — выключает режим вывода на экран текста Script-файла.
- echo <fi1e_name> — меняет режим вывода на противоположный.
- echo on all — включает режим вывода на экран текста всех m-файлов.
- echo off all — отключает режим вывода на экран текста всех m-файлов.
- more on — включает режим постраничного вывода (полезен при просмотре больших m-файлов).
- more off — отключает режим постраничного вывода (в этом случае для просмотра больших фалов надо пользоваться линейкой прокрутки).

В версии MATLAB 6.0 обе команды clc и home действуют аналогично — очищают экран и помещают курсор в левый верхний угол окна командного режима работы. Команды echo позволяют включать или выключать отображение текстов m-файлов при каждом обращении к ним. Как правило, отображение текста файлов сильно загромождает экран и часто не является необходимым. При больших размерах файлов начало их текста (листинга) убегает далеко за пределы области просмотра (текущего окна командного режима). Поэтому для просмотра длинных листингов файлов полезно включить постраничный вывод командой more on. Различие между m-файлами сценариев и функций мы обсудим позже.

Система MATLAB создана таким образом, что любые (подчас весьма сложные) вычисления можно было выполнять в режиме *прямых вычислений,* то есть без подготовки программы. Это превращает MATLAB в необычайно мощный калькулятор, который способен производить не только обычные для калькуляторов вычисления (например, выполнять арифметические операции и вычислять элементарные функции), но и операции с векторами и матрицами, комплексными числами, рядами и полиномами. Можно почти мгновенно задать и вывести графики различных функций — от простой синусоиды до сложной трехмерной фигуры.

Работа с системой в режиме прямых вычислений носит диалоговый характер и происходит по правилу «задал вопрос, получил ответ». Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) в командной строке и завершает ввод нажатием клавиши ENTER.

При этом полезно знать несколько несложных правил:

- для указания ввода исходных данных используется символ »;
- данные вводятся с помощью простейшего строчного редактора;
- для блокировки вывода результата вычислений некоторого выражения после него надо установить знак ; (точка с запятой);
- если не указана переменная для значения результата вычислений, то MATLAB назначает такую переменную с именем ans;
- знаком присваивания является привычный математикам знак равенства =, а не комбинированный знак :=, как во многих других языках программирования и математических системах;
- результат вычислений выводится в строках вывода (без знака »);
- встроенные функции (например, sin) записываются строчными буквами, и их аргументы указываются в *круглых скобках;*
- диалог происходит в стиле «задал вопрос — получил ответ».

Порядок выполнения лабораторной работы

Упражнение 1.

Вычислить:

$\sqrt{100} =$ |-10| = 10! = .

Упражнение 2.

Определить переменные: $a = 3.4$, $b = 6.22$, $c = 0.149$ и выражения:

$$Z = \frac{2ab + \sqrt[3]{c}}{\sqrt{\left(a^2 + b^{a+c}\right) \cdot c}} \qquad N = e^{\sin c} \cos\frac{a}{b}.$$

- Вычислить выражения.
- Изменить точность отображения результатов вычисления *глобально*.

Упражнение 3.

Вывести на экран значение *системной константы* π и установить максимальный формат ее отображения.

Упражнение 4.

Выполнить следующие операции с комплексными числами:

Z = -3 + 2i |Z| = Re(Z) = Im(Z) = arg(Z) =

$\sqrt{Z} =$ $\sqrt{-5} =$ 2 · Z = Z1 = 1 + 2i Z2 = 3 + 4i

Z1 + Z2 = Z1 - Z2 = Z1· Z2 = Z1/Z2 =

Контрольные вопросы

1. С помощью какого оператора можно вычислить выражение?
2. Что такое поэлементное выполнение операций и как его задать?
3. Как изменить формат представления чисел?
4. Как определить индексированную переменную?
5. Что значит двоеточие в разных вариантах применения?

6. Какие виды массивов Вам известны?
7. Как устроена система помощи Матлаб?

11 Использование системы *Matlab* в качестве научного калькулятора. Работа с векторами и матрицами.

Теоретические сведения

Operators and special characters.

Арифметические операторы.

plus	- Сложение	+
uplus	- Унарный плюс	+
minus	- Вычитание	-
uminus	- Унарный минус	-
mtimes	- Матричное умножение	*
times	- Поэлементное умножение	.*
mpower	- Степень матрицы	^
power	- Поэлементное возведение в степень	.^
mldivide	- "Левое" деление матриц	\
mrdivide	- "Правое" деление матриц	/
ldivide	- "Левое" поэлементное деление	.\
rdivide	- "Правое" поэлементное деление	./
kron	- Кронекерово произведение	kron

Отношения.

eq	- Равно	==
ne	- Не равно	~=
lt	- Меньше чем	<
gt	- Больше чем	>
le	- Меньше или равно	<=
ge	- Больше или равно	>=

Логические операторы.

	"Короткое" логическое "И"	&&
	"Короткое" логическое "ИЛИ"	||
and	- Логическое "И"	&
or	- Логическое "ИЛИ"	|
not	- Отрицание	~
xor	- Исключающее или	
any	- "Хоть один ненуль"	
all	- "Все ненули"	

Специальные символы.

colon - Двоеточие :
paren - Скобки и индексация ( )
paren - Создание массива [ ]
paren - Индексация массива ячеек { }
punct - Указатель на функцию @
punct - Десятичная точка .
punct - Доступ к полю структуры .
punct - Папка верхнего уровня ..
punct - Продолжение следует ...
punct - Разделитель ,
punct - Точка с запятой ;
punct - Комментарий %
punct - Команда операционной системы !
punct - Присвоить =
punct - Кавычка '
transpose - Транспонированная матрица .'
ctranspose - Сопряжённая матрица '
horzcat - Горизонтальное соединение [,]
vertcat - Вертикальное соединение [;]
subsasgn - Индексация элемента массива ( ),{ },.

Побитные операторы.
bitand - Побитное логическое "И".
bitcmp - Побитное сравнение.
bitor - Побитное логическое "ИЛИ".
bitmax - Наибольшее целое.
bitxor - Побитное логическое исключающее "ИЛИ".
bitset - Установить конкретный бит.
bitget - Получить конкретный бит.
bitshift - Побитный сдвиг.

Операторы над множествами (строки и вектора).
union - Объединение (для строк или векторов).
unique - Исключение повторов (для строк или векторов).
intersect - Пересечение.
setdiff - Разность.
setxor - Симметрическая разность.
ismember - Принадлежность.

Следует сделать некоторые замечания к этому тексту.

Во-первых, видно, что все арифметические (и прочие) операции выполняются обращением к некоторым функциям. Например, операцию сложения реализует функция plus.

```
>> x=plus(2,3)
```

```
x =
    5
>>
```

Для большинства операторов есть эквивалентная инфиксная форма, но для некоторых она отсутствует. В частности, для всех операций с множествами (Set operators).

Во-вторых, многие операции, например умножение, могут пониматься (и выполняться) как операции над числами, над матрицами или над битами (двоичные разряды числа). Например, умножение. Эти операции, хотя и изображаются одним и тем же символом, осуществляются разными функциями. Операции над битами осуществляют функции, начинающиеся на bit (bitand), функции матричных операций начинаются на *m* (mtimes). Если требуется подчеркнуть, что операция выполняется не по матричным правилам, а над элементами матрицы как над числами (element-wise), то перед символом операции ставится точка. Этот трюк в Маткаде назывался векторизацией оператора.

Arithmetic operators (арифметические операторы).

Для того чтобы задать значение матрицы, надо значения элементов поместить в квадратные скобки, отделяя элементы строки запятыми или пробелами, а строки друг от друга – точкой с запятой.

Пример 1:

```
>> x=[1 2 3; 4 5 6]
x =
    1    2    3
    4    5    6
>> x=[x;7 8 9]
x =
    1    2    3
    4    5    6
    7    8    9
>> y=[0; 1; 0 ]
y =
    0
    1
    0
>> x=[x,y]
x =
    1    2    3    0
    4    5    6    1
    7    8    9    0
>>
```

Разумеется, пользователь сам должен следить за соблюдением размерностей, например, за одинаковостью длин строк и столбцов.

Поскольку в Матлаб любые данные по умолчанию считаются матрицами, все арифметические операции понимаются как операции над матрицами. Для части операций (сложение, умножение на число) это не вызывает проблем, для других же результаты могут показаться неожиданными.

Пример 2

```
>> x=[1 2 3; 4 5 6; 7 8 9]
x =
    1    2    3
    4    5    6
    7    8    9
>> y=eye(3)
y =
    1    0    0
    0    1    0
    0    0    1
>> y\x
ans =
    1    2    3
    4    5    6
    7    8    9
>>
```

В частности, выше показано правое деление матриц "/", а заодно продемонстрирована функция создания единичной матрицы заданной размерности eye.

Ещё более удивительным выглядит деление неквадратных матриц.

```
>> x=[x(1,:); x(2,:)]
x =
    1    2    3
    4    5    6
>> y=[y(1,:); y(2,:)]
y =
    1    0    0
    0    1    0
>> x/y
ans =
    1    2
    4    5
>>
```

Обратите внимание, что это отнюдь не поэлементное, а матричное деление.

Relational и Logical operators (отношения и логические оперции)

Результатом операций сравнения будет логическое значение, true или false. Однако, как и в Маткаде, оно может использоваться в арифметических операциях (true =1, false = 0) и, напротив, числа могут рассматриваться как логические значения (0 = false, не 0 = true).

Пример 3:

```
>> 2==2
ans =
    1
>> 2 | 3.5
ans =
    1
>> 0 & 2
ans =
    0
>>
```

Внимания заслуживают "короткие" формы логических операторов, известные ещё из Паскаля, только там установка "короткой" формы логических операторов задаётся опцией компилятора. Две логические функции раньше никогда не встречались нам в других местах all и any. Аргументом этих функций является массив, рассматриваемый как массив логических значений. Первая из них возвращает "1", только когда все элементы массива – аргумента ненулевые, а вторая возвращает "0" только если все элементы массива – аргумента нули.

Special characters (специальные символы)

Из специальных символов стоит перечислить круглые скобки (индексация массивов, обычные скобки, аргументы функций), квадратные скобки (задание вектора или матрицы), точка с запятой (подавление печати результата в конце команды, разделение строк при задании матрицы) и двоеточие, о котором стоит поговорить отдельно.

Двоеточие позволяет формировать последовательность с постоянным шагом, причём эта последовательность оказывается индексированной, то есть вектором, если присваивается какой-то переменной:

```
>> x=1:0.2:2
x =
   1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
>> x(3)
```

```
ans =
   1.4000
>>
```

Вектор x формируется из последовательности значений начиная с 1 с шагом 0.2 и кончая значением 2.

Несколько иной смысл имеет двоеточие при индексации массивов. Просто двоеточие означает все значения индекса, двоеточие между двумя значениями задаёт диапазон значений индекса. Например:

```
>> x=[1 2 3 1 2 3;4 5 6 4 5 6;7 8 9 7 8 9]
x =
    1    2    3    1    2    3
    4    5    6    4    5    6
    7    8    9    7    8    9
>> x(:,2:5)
ans =
    2    3    1    2
    5    6    4    5
    8    9    7    8
>>
```

Запись x(:,2:5) означает все строки столбцов со 2-го по 5-й.

Программирование и m- файлы.

Структурные типы данных

Кроме простейшей для Матлаб структуры – матрицы, предопределены ещё структуры (struct) – аналог записей (record) языка Паскаль. Они также включают поля различных (неодинаковых) типов, которые идентифицируются не по индексу, как элементы массива, а по имени. Синтаксис вполне традиционный.

Пример 7

```
>> Colzveno=struct('T',0.01,'ksi',0.2,'K',12.34)
Colzveno =
      T: 0.0100
    ksi: 0.2000
      K: 12.3400
>> Colzveno.T
ans =
   0.0100
>>
```

Обратите внимание на первую стоку примера. В ней вызывается конструктор типа "структура", то есть специальная функция, создающая переменную типа "структура" с заданными полями. Аргументами этого конструктора служат пары "имя поля", "значение поля". Тип поля определяется по типу значения, по умолчанию – матрица. Создаётся,

строго говоря, также не одна переменная, а массив структур. Добавив к имени структуры индексы, можно обратиться к другим элементам этого массива.

Пример 8

```
>> Colzveno(1)       %проверяем первый элемент массива структур
ans =
      T: 0.0100
    ksi: 0.2000
      K: 12.3400
>> Colzveno(2).ksi=0.1    %определяем одно поле второго элемента
Colzveno =
1x2 struct array with fields:     % посмотрите, как изменился вывод
    T
    ksi
    K
>> Colzveno(2)     %проверяем второй элемент массива
ans =
      T: []
    ksi: 0.1000
      K: []
>>
```

Другая структура, используемая в Матлаб – массив ячеек (cell array). Он похож на вложенные массивы Маткада и состоит из элементов различного типа, в отличие от классических массивов, у которых все элементы однородны (одного типа). На первый взгляд, отличие в синтаксисе состоит в замене круглых скобок фигурными.

Пример 9

```
>> X{1}=123.567
X =
    [123.5670]
>> X{2}='John was here'
X =
    [123.5670]    'John was here'
>> X{2,1}=Colzveno(1)
X =
    [  123.5670]    'John was here'
    [1x1 struct]                 []
>>
```

При создании массива ячеек или изменении его элементов так оно и есть.

Пример 10

```
>> X{2}
ans =
      T: 0.0100
```

```
    ksi: 0.2000
      K: 12.3400
>> X(2)
ans =
    [1x1 struct]
>>
```

Обратите внимание, что круглые скобки тоже можно использовать, но результат оказывается иным. Используя фигурные скобки, мы можем "заглянуть внутрь" индексируемой ячейки, а используя круглые – только посмотреть на неё "снаружи", то есть увидеть, какого она типа.

Кроме того фигурные скобки являются конструктором массива ячеек

Пример 11

```
>> Y={'Jack was here too' Colzveno(2);[1 2 3;4 5 6] [7;8;9]}
Y =
    'Jack was here too'    [1x1 struct]
          [2x3 double]    [3x1 double]
>> Y(1,2)
ans =
    [1x1 struct]
>> Y{1,2}
ans =
      T: []
    ksi: 0.1000
      K: []
>> Y{1,2}.ksi
ans =
    0.1000
>> Y(1,2).ksi
??? Improper index matrix reference.
```

Как видите, используя круглые индексные скобки, не удаётся обратиться к внутренней структуре ячейки.

Управляющие операторы

Их немного.

Ветвление: IF и SWITCH

```
IF expression1
        statements
      ELSEIF expression2
        statements
      ELSEIF expression3
        statements
……
……
```

```
ELSE
  statements
END

SWITCH switch_expr
   CASE value1,
     statements1
   CASE {value2, value3,...}
     statements2
  ...
   OTHERWISE,
     statements3
 END
```

Если одним и тем же действиям может соответствовать несколько значений переключателя, в этом случае такие значения объединяются в фигурных скобках (образуют массив ячеек).

Циклы FOR и WHILE

Оператор цикла с параметром никаких проблем не должен вызвать.

```
>> for k=1:5,
      A(k)=0; xx=0;
      for j=0:0.3:k,
         A(k)=A(k)+k*j;
         xx=xx+j./k;
      end,
   end
>> A, xx
A =
   1.8000   12.6000   49.5000  109.2000  204.0000
xx =
   8.1600
>>
```

Цикл с предусловием тоже не содержит никаких необычных моментов.

```
>> x=2;
>> while abs(sin(x)-x)>0.0001,
        x=sin(x);
    end
>>x
x =
   0.0843
```

Команды ввода – вывода

Матлаб содержит богатые возможности сохранения данных на диске и вывода их на экран. Рассмотрим только некоторые из них: save – сохранение на диске переменных (рабочего пространства), fprintf – вывод форматированных данных на диск, sprintf – вывод форматированных данных в командное окно.

Команда save вызывается нетипично, не как функция, а именно как команда:

save filename var1 var2 … -key

filename – имя файла для сохранения (по умолчанию matlab.mat),

var1 var2 … – имена сохраняемых переменных (по умолчанию – все),

-key – ключи -ascii (текстовой файл) –mat (двоичный файл), по умолчанию -mat.

fprintf – функция, синтаксис которой точно повторяет синтаксис одноименной функции языка С:

fprintf(FileID, Format, Data1, Data2, …).

FileID – целый идентификатор файла, открытого функцией fopen,
Format – форматная строка, управляющая видом вывода,
Data – данные.

Пример 12

```
>>x = 0:.5:2;
>>fid = fopen('exp.txt', 'w')
        ans =
              3
>>fprintf(3,'%6.2f  %12.8f\n', [x; exp(x)])
        ans =
              100
>>fclose(3)
        ans =
               0
>>[x; exp(x)]
ans =
        0    0.5000    1.0000    1.5000    2.0000
   1.0000    1.6487    2.7183    4.4817    7.3891
```

Важно помнить, что эта функция разворачивает матрицу не по строкам, а по столбцам, поэтому в файле оказывается следующий текст:

```
0.00   1.00000000
0.50   1.64872127
1.00   2.71828183
1.50   4.48168907
2.00   7.38905610
```

Назначение функций fopen и fclose едва ли нуждается в пояснениях, а вот форматные строки встречаются нам впервые. Каждая форматная группа начинается символом "%" и содержит длину выводимой порции данных и "точность" – количество десятичных знаков, а завершается одним из символов "d", "i", "o", "u", "x", "X", "f", "e", "E", "g", "G", "c", или "s", кроме того могут встречаться ещё команды: "\n", "\r", "\t" – конец строки, возврат каретки, табуляция и некоторые другие. Кроме форматных групп и команд в форматной строке могут содержаться любые символы, которые будут присутствовать в выводе. Когда форматная строка кончилась, а данные – нет, форматную строку начинают снова сначала.

Совершенно аналогично работает функция строкового вывода sprintf, только ей, естественно, не требуется идентификатор файла, поскольку она осуществляет вывод в командное окно.

Пример 13

```
>> sprintf( ' x=%6.2f       |      y=%12.8f  |\n',[x; exp(x)])
ans =
 x=  0.00       |      y=  1.00000000  |
 x=  0.50       |      y=  1.64872127  |
 x=  1.00       |      y=  2.71828183  |
 x=  1.50       |      y=  4.48168907  |
 x=  2.00       |      y=  7.38905610  |

>>
```

Обратите внимание, что в конце вывода вставлена пустая строка, как это предписывает форматная команда.

Для ввода данных предназначены две симметричные рассмотренным функции fscanf и sscanf – форматированного ввода из файла и с клавиатуры.

Интерактивное взаимодействие

Рассмотрим только две функции диалогового взаимодействия, имеющиеся в Матлаб, input, inputdlg и menu. Первая напоминает одноимённый оператор языка Бэйсик:

```
>> x=input('Ввод данных \n Введите x  ')
Ввод данных
 Введите x  25.14
x =
      25.14
```

Аргументом этой функции служит строка – подсказка (запрос), а возвращает она то значение, которое будет введено с клавиатуры. Запрос

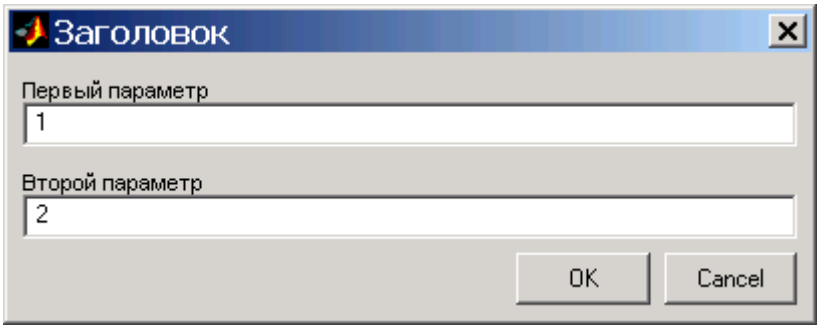
выводится в командном окне. Внутри запроса, кроме обычных символов, могут присутствовать некоторые команды, например, "\n" – перевод строки, так что запрос может состоять из нескольких строк.

Вторым, необязательным, аргументом может служить символ, управляющий преобразованием вводимых данных. Так 's' означает, что введённые данные должны рассматриваться не как число, а как строка.

```
>> >> x=input('Ввод данных \n Введите x  ', 's')
Ввод данных
 Введите x  25.14
x =
25.14
>> x/2
ans =
        25        26.5        23        24.5        26
>>
```

Введённое число интерпретируется как строка символов, а поделив x пополам, мы получаем матрицу-строку, состоящую из поделённых на два кодов цифр.

Вторая функция осуществляет ввод данных через отдельное диалоговое окно примерно такого вида:



которое выводится в ответ на команду

```
>> a=inputdlg({'Первый параметр';'Второй параметр'
},'Заголовок',1,{'1';'2'})
```

Аргументами этой функции являются:

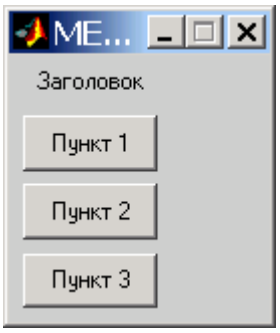
- подсказки – массив ячеек, состоит из строк по количеству вводимых параметров,
- Заголовок окна – строка,
- количество строк ввода (общее для всех параметров) – целое,
- значения по умолчанию – массив ячеек, состоит из строк.

Возвращает эта функция массив ячеек, состоящий из строк. Никаких преобразований в числовой формат она не производит, об этом надо позаботиться программисту, используя, например, функцию str2num.

```
a =
    '1'
    '2'
>> STR2NUM(a{1})
ans =
     1
>> STR2NUM(a{2})
ans =
     2
>>
```

Наконец, функция menu выводит в отдельном окне определяемое пользователем меню, а возвращает номер выбранного пользователем пункта (integer)

```
>> menu('Заголовок',{'Пункт 1','Пункт 2','Пункт 3'})
```




```
ans =
     3
```

Пользовательские функции и сценарии

Управляющие операторы, диалоги, меню неудобно использовать, находясь в командном окне, они логичны в отдельном программном модуле. Такие модули в Матлабе создаются пользователем в виде отдельных текстовых файлов с расширением m, m-файлах. Они бывают двух сортов – сценарии и функции. Эти два сорта файлов отличаются двумя основными моментами – наличием заголовка и пространством имён.

Сценарий не имеет специального заголовка, зато имеет общее с командным окном пространство имён, то есть все переменные, определённые в командном окне, доступны сценарию, и наоборот. Набрав в командном окне имя сценария (то есть файла, в котором он сохранён), мы запустим его выполнение, и все его команды будут выполнены. Никакого специального значения возвращено при этом не будет.

Например, в файле script.m содержится следующий текст:

```
k=menu('Заголовок',{'Пункт 1','Пункт 2','Пункт 3'});
```

```
switch k
   case 1, disp('Выбран пункт 1')
   case 2, disp('Выбран пункт 2')
   case 3, disp('Выбран пункт 3')
   otherwise, disp('Ничего не выбрано')
end
```

Естественно, никаких приглашений ">>" в нём быть не должно.

Вызов этого сценария из командного окна приводит к следующему результату:

```
>> script
Выбран пункт 2
>> k
k =
    2
>>
```

Определённая в сценарии переменная k доступна и в командном окне.

Функция, заданная в m-файле (m-функция), должна иметь заголовок специальной формы:

function [x y …] = filename(a,b,c, …)

где x, y, … – выходные параметры,

a,b,c, …. – входные параметры,

filename – имя функции, совпадающее с именем файла.

После строки заголовка могут содержаться комментарии к этой функции, каждая строка этих комментариев начинается с символа "%". Эти комментарии играют специальную роль. Набрав в командном окне команду help filename, мы увидим эти комментарии.

Порядок выполнения лабораторной работы

Составить сценарий, который запрашивает у пользователя параметры колебательного звена К, Т, Кси, и, по выбору пользователя через меню, либо строит ЛАЧХ, либо вычисляет частоту среза, либо вычисляет перерегулирование.

Контрольные вопросы

1. Чем отличается сценарий от m-функции?
2. В чём отличие функций input и inputdlg?
3. С каким оператором в паре может использоваться функция menu?
4. В чём отличие функций sprintf и fprintf?

12 Графические средства системы *Matlab*.

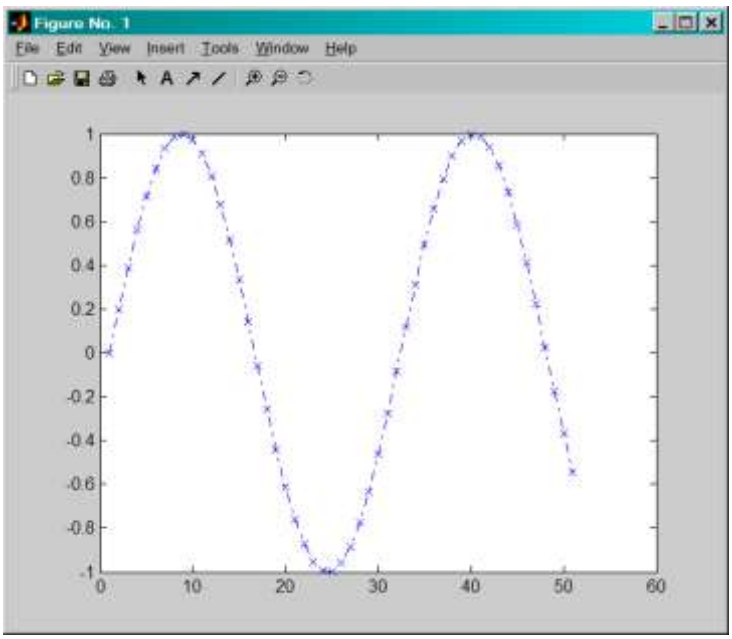
Теоретические сведения

Для построения двумерных графиков используется функция plot, которой требуется, по крайней мере, один векторный аргумент. Тогда индекс будет интерпретироваться как значение по оси абсцисс, а сам элемент вектора – по оси ординат. График будет изображён в отдельном окне, которое получит имя "Figure No. 1"

Пример 4:

```
>> x=0:0.2:10;
>> plot(sin(x),'b-.x')
>>
```

Создан массив иксов со значениями от нуля до 10-ти с шагом 0.2. Функции plot передан массив sin(x) (выполняется поэлементно) и параметры графика – цвет линии, стиль линии и символ, отмечающий каждую точку массива. Результат – на следующем рисунке.



Ниже приведены расшифровки параметров графика

цвет		символ		стиль	
b	синий	.	точка	-	сплошная
g	зеленый	o	кружок	:	пунктир
r	красный	x	косой крест	-.	штрих-пунктир
c	голубой	+	прямой крест	--	штриховая
m	пурпурный	*	звёздочка		

y	жёлтый	s	квадрат
k	чёрный	d	ромб
		v	треугольник вершиной вниз
		^	треугольник вершиной вверх
		<	треугольник вершиной влево
		>	треугольник вершиной вправо
		p	пентаграмма
		h	гексаграмма

В графическом окне доступна масса настроек через систему меню, как главного, так и контекстных. Особенно надо отметить функцию копирования рисунка (**Edit** => **Copy Figure**). В буфер обмена будет скопирован график без обрамления окна, параметры копирования можно настроить через меню (**Edit**=> **Copy Options**). После настройки и копирования график может быть вставлен в документ, например, в таком виде:

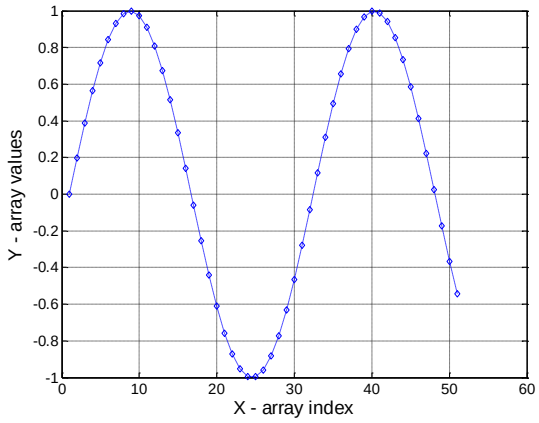


График можно сохранить на диске либо в собственном формате Матлаба с расширением fig, либо экспортировать в один из множества графических форматов, как растровых, так и векторных.

Полный вид обращения к функции plot таков:

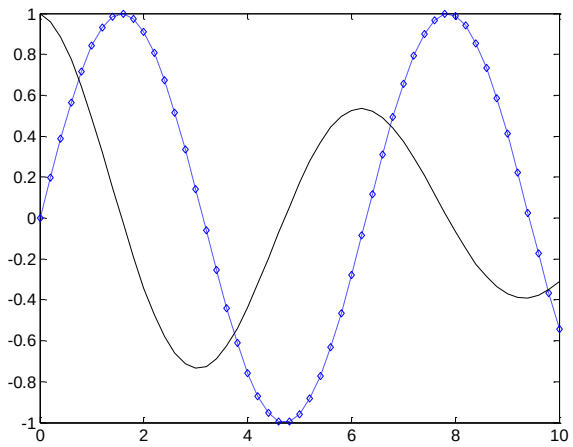
plot(X1,Y1,S1, X2,Y2,S2…),

где X1, X2, … – массивы X-ов, Y1, Y2, … – массивы Y – ков, S1, S2, … – массивы строк с параметрами графиков.

Например:

```
>> plot(x, sin(x), 'b-.d', x, exp(-x/10).*cos(x), 'k-')
```

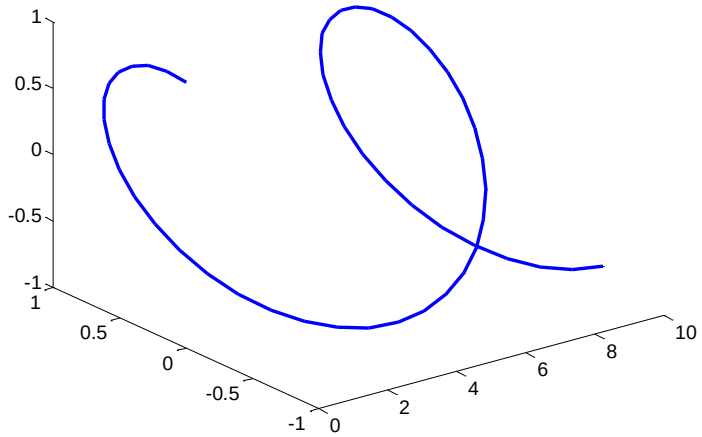
График построен для того же вектора x, что и в предыдущем примере и никакой дополнительной настройке не подвергался.



Трёхмерные линии строятся с помощью функции plot3, которой надо передавать тройки массивов, Xk, Yk, Zk, в которых содержатся соответствующие координаты точек в пространстве. Массивы должны в каждой тройке быть одинаковой размерности. Если массивы одномерные (вектора), будет построена одна линия для каждой триады Xk, Yk, Zk, если массивы двумерные (матрицы), будет построена линия для каждого столбца этих матриц. Как и в двумерных графиках, можно указывать дополнительные параметры линий в том же формате.

Пример 5

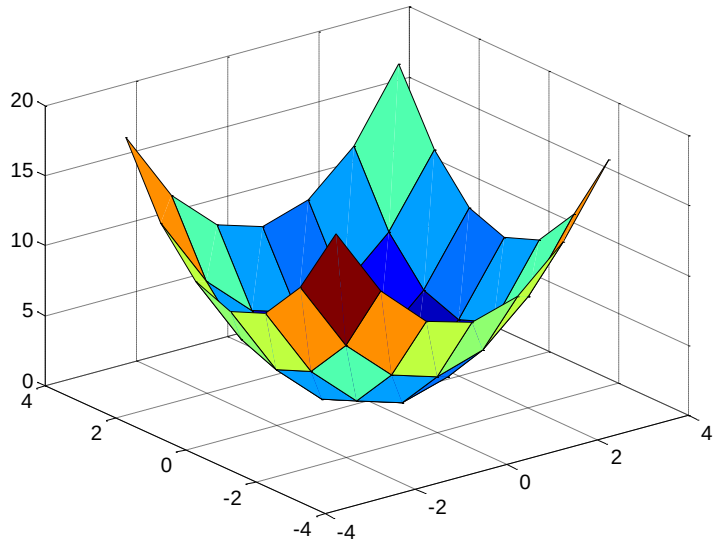
```
>>plot3(x, sin(x), cos(x))
```



Для построения поверхностей используется функция surf, для которой требуется по меньшей мере одна матрица, тогда её индексы будут рассматриваться как иксы и игреки, а элементы матрицы – как координата z. Если дать её в качестве аргументов три матрицы, то будет построена поверхность из точек, первые координаты которых, иксы, находятся в первой матрице, игреки – во второй, а зеты – в третьей. Третий вариант обращения – использование двух векторов и одной матрицы. Тогда первый вектор содержит все значения первой координаты, икса, второй вектор – игрека, а матрица – значения координаты z в точках $\left(x_i, y_j\right)$.

Пример 6:

```
>>  x=[-3 -2 -1 0 1 2 3];
>> y=x;
>> for i=1:7, for j=1:7, Z(i,j)=x(i)^2+y(j)^2; end, end;
>> surf(x, y, Z)
```



Порядок выполнения

Упражнение 1.

Определить векторы *d*, *S* и *R*. Отобразить графически таблично заданные функции $S_i(d_i)$ и $R_i(d_i)$.

i	d_i	S_i	R_i
0	0.5	3.3	2
1	1	5.9	3.9
2	1.5	7	4.5
3	2	6.3	3.7
4	2.5	4.2	1.2

- Нанести линии сетки на график и отобразить легенду
- Отформатировать график так, чтобы в каждой узловой точке графика функции $S_i(d_i)$ стоял знак вида □, а график функции $R_i(d_i)$ отобразить в виде гистограммы.

Упражнение2.

Построить декартовы графики следующих функций:

$$X(\alpha) = \cos(\alpha) \cdot \sin(\alpha)$$

$$Y(\alpha) = 1.5\cos(\alpha)^2 - 1$$

$$P(\alpha) = \cos(\alpha).$$

Упражнение 3.

Создать матрицу Q размером 6×6, заполнить ее произвольно и отобразить графически как трёхмерную поверхность.

Отобразить графически пересечение поверхностей $f1(x,y)=\frac{(x+y)^2}{10}$ и $f2(x,y)=5\cdot\cos\left(\frac{x-y}{3}\right)$.

Контрольные вопросы

1. Как построить график поверхности, пространственной линии?
2. Как построить несколько графиков в одной системе координат?
3. Как изменить масштаб графика?
4. Как определить координату точки на графике?
5. Какие функции используются для построения трехмерных графиков?

13 Использование объектов и методов *Control System Toolbox* для анализа линейной системы управления.

Теоретические сведения

MATLAB® содержит богатый набор функций, полезных для инженера или теоретика по системам автоматического управления. Комплексная арифметика, собственные числа, нахождение корней, обращение матриц, быстрое преобразование Фурье – вот только некоторые численные инструменты, имеющиеся в ядре MATLAB. Но кроме ядра Matlab содержит большое количество пакетов расширения (Toolbox), ориентированных на решение прикладных задач из разных областей.

Control System Toolbox содержит функции, реализующие основные задачи анализа и синтеза линейных стационарных (Linear Time Invariant – LTI) систем управления, как непрерывных, так и дискретных по времени. Удобный графический интерфейс (Graphic User Interface – GUI) упрощает решение многих типовых инженерных задач.

Основу Control System Toolbox составляет специальный класс объектов – LTI object, то есть обобщённая линейная стационарная динамическая система (ЛСС). Это абстрактный класс, который сам по себе не может иметь конкретных экземпляров, непосредственная работа выполняется с объектами-наследниками этого класса, которые реализуют три формы представления ЛСС:

tf – передаточной функцией, заданной коэффициентами числителя и знаменателя (Transfer Function);

zpk – передаточной функцией, заданной нулями (корни числителя), полюсами (корни знаменателя) и коэффициентом усиления (Zeros-Poles-GainK);

ss– в пространстве состояний (State Space) в виде четырёх матриц, A, B, C, D, стандартного описания ЛСС:

$$\dot{x} = Ax + Bu,$$

$$y = Cx + Du,$$

где x – вектор состояния ЛСС, u – вектор входных сигналов, y – вектор выходных сигналов.

Свойства ЛСС-объекта.

Все перечисленные объекты кроме общих свойств, унаследованных от класса LTI, имеют свои специфические свойства. Посмотреть список свойств любого класса можно с помощью функции get, единственным аргументом которой должно быть имя этого класса.

```
>> get (LTI)
          Ts: 0
     ioDelay: []
  InputDelay: [0x1 double]
 OutputDelay: [0x1 double]
   InputName: {0x1 cell}
  OutputName: {0x1 cell}
  InputGroup: {0x2 cell}
 OutputGroup: {0x2 cell}
       Notes: {}
    UserData: []
```

Эти свойства имеют следующий смысл:

- Ts: Time sample – интервал дискретизации, для непрерывных систем равен нулю.
- ioDelay: матрица запаздываний от i- го входа к j-тому выходу.
- InputDelay: вектор запаздываний на входах.
- OutputDelay: вектор запаздываний на выходах.
- InputName: имена входных сигналов (массив ячеек – строк).
- OutputName: имена выходных сигналов (массив ячеек – строк).
- InputGroup: группировка входных сигналов (массив ячеек – векторов, каждый вектор содержит номера сигналов одной группы).
- OutputGroup: группировка выходных сигналов (массив ячеек – векторов).
- Notes: произвольный текст, комментарии (массив ячеек – строк).
- UserData: дополнительные пользовательские данные (вектор).

В зависимости от количества входов и выходов ЛСС делят на одномерные (SISO – single-input, single-output) и многомерные (MIMO – multiple-input, multiple-output).

ЛСС, заданная **передаточной функцией**, tf, имеет дополнительные свойства:

- num: числители передаточных функций (двумерный массив ячеек – векторов, содержащих коэффициенты числителей передаточных функций от i- го входа к j-тому выходу в порядке убывания степени).
- den: знаменатели передаточных функций (двумерный массив ячеек – векторов, содержащих коэффициенты знаменателей передаточных функций от i- го входа к j-тому выходу в порядке убывания степени).
- Variable: переменная – аргумент передаточной функции (возможны только два варианта – 's' или 'p' для непрерывных систем, 'z' или 'p' для дискретных). Используется только для отображения передаточной функции.

ЛСС, заданная **нулями и полюсами**, zpk, имеет дополнительные свойства:

- z: нули (двумерный массив ячеек - векторов, содержащих корни числителей передаточных функций от i- го входа к j-тому выходу).
- p: полюса (двумерный массив ячеек - векторов, содержащих корни знаменателей передаточных функций от i- го входа к j-тому выходу).
- k: матрица коэффициентов усиления от i- го входа к j-тому выходу.
- Variable: переменная – аргумент передаточной функции (возможны только два варианта – 's' или 'p' для непрерывных систем, 'z' для дискретных). Используется только для отображения.

ЛСС, заданная **в пространстве состояний**, ss, имеет дополнительные свойства:

- a: матрица A.
- b: матрица B.
- c: матрица C.
- d: матрица D.
- e: матрица E.
- StateName: имена переменных состояния (массив ячеек - строк).

Матрица E, как правило, бывает пустой и используется только в "неявных" (дескрипторных) системах вида

$$E\dot{x} = Ax + Bu,$$

$$y = Cx + Du.$$

Такие системы в дальнейшем не рассматриваются. Для их создания служит специальная функция dss.

Создание и преобразования ЛСС

Имена классов – ss, tf, zpk, одновременно служат и именами конструкторов этих классов. Рассмотрим пример создания SISO.

Пример 1

Создадим одномерную систему Sys.

```
>> Sys=tf([3],[0.01 0.2 1])

Transfer function:
        3
--------------------
0.01 s^2 + 0.2 s + 1
```

Есть ещё одна интересная возможность создать tf-объект. Надо создать передаточную функцию s и обычным для математики образом записать дробно-рациональное выражение от этой переменной

```
>> s=tf('s')

Transfer function:
s
```

Теперь переменная s обозначает передаточную функцию "чистое дифференцирование", "s". Запишем теперь передаточную функцию

```
>> W=7/(0.001*s^2+0.02*s+1)
```

```
Transfer function:
        7
---------------------
0.001 s^2 + 0.02 s + 1
```

Внимательно присмотритесь к этому примеру. Последняя формула содержит арифметическое выражение, в которое входят обычные числовые константы и объект – передаточная функция. Для таких объектов, как и для большинства объектов Matlab, определены методы, реализующие арифметические операции, которые имеют, разумеется, свой особенный смысл для каждого класса объектов. Для ЛСС в частности сложение означает параллельное соединение, умножение – последовательное соединение, деление – последовательное соединение с обратной передаточной функцией. То есть обычные операции (структурные преобразования) над передаточными функциями. Для многомерных систем такие операции местами напоминают матричную арифметику. Но мы не будем заниматься многомерными системами.

Для манипулирования (структурных преобразований) ЛСС не хватает ещё одной незаменимой операции – замыкания обратной связи. Она реализуется функцией feedback с двумя обязательными аргументами – передаточными функциями прямой цепи и обратной связи. Если третьим необязательным аргументом будет "+1", будет замкнута положительная обратная связь, если не задавать третий аргумент, обратная связь будет отрицательной.

Пример 2

```
>>  W1=tf ( 1, [1 0] )

Transfer function:
1
-
s

>> W2=tf(1)

Transfer function:
1

>> feedback(W1,W2)

Transfer function:
  1
-----
s + 1

>> feedback(W1,W2,+1)
```

```
Transfer function:
  1
-----
s - 1

>>
```

Аналогичным путём можно создавать и другие разновидности (формы) ЛСС, что проще показать на примере.

Пример 3

```
>> z=[];                          %Вектор нулей (числителя) - пустой
>> p=[0  -1+2*j  -1-2*j  -2];     %Вектор полюсов (нулей знаменателя)
>> K=pi;                           %Коэффициент усиления
>> zpk(z,p,K)

Zero/pole/gain:
       3.1416                     %Передаточная функция
-----------------------           %с явно выделенными нулями
s (s+2) (s^2  + 2s + 5)            %и полюсами

>> A=[-1  1; -2 0];    B=[0; 1];   C=[1 0];   D=0;
>> ss(A,B,C,D)

a =
      x1  x2
  x1  -1   1
  x2  -2   0


b =
      u1
  x1   0
  x2   1


c =
      x1  x2
  y1   1   0


d =
      u1
  y1   0
```

```
Continuous-time model.
>>
```

Три перечисленные выше класса ЛСС взаимозаменяемы почти всюду, их можно смешивать в арифметических выражениях (структурных преобразованиях). Более того, их можно преобразовывать один в другой, используя те же конструкторы, что и для создания.

Последняя ss – система сохранена в переменной ans. Преобразуем её в передаточную функцию:

```
>> tf(ans)

Transfer function:
    1
-----------
s^2 + s + 2

>>
```

Выше была определена передаточная функция (tf-объект) W. Преобразуем её в другие формы:

```
>> zpk(W)

Zero/pole/gain:
      7000
-------------------
(s^2  + 20s + 1000)

>> ss(W)

a =
         x1      x2
  x1     -20  -15.63
  x2      64       0


b =
      u1
  x1   8
  x2   0


c =
        x1     x2
  y1     0  13.67
```

```
d =
      u1
  y1   0

Continuous-time model.
>>
```

Ещё одна неординарная возможность при работе с ЛСС – извлечение данных о модели. Для передаточной функции это числитель (вектор коэффициентов) и знаменатель (вектор коэффициентов). Для zpk – нули, полюса и коэффициент усиления, для ss – соответствующие матрицы.

Для всех типов возвращается ещё один параметр – квант времени, если система дискретная, и ноль, если непрерывная.

Пример 4

```
>> [a b c]=tfdata(W)
a =
    [1x3 double]
b =
    [1x3 double]
c =
     0
>> a{1}
ans =
     0     0     7
>> b{1}
ans =
    0.0010    0.0200    1.0000

>> [a b c]=zpkdata(W)
a =
    {[]}
b =
    [2x1 double]
c =
        7000
>> b{1}
ans =
 -10.0000 +30.0000i
 -10.0000 -30.0000i

>> [a b c d]=ssdata(W)
a =
  -20.0000  -15.6250
```

```
   64.0000         0
b =
    8
    0
c =
         0   13.6719
d =
    0
>>
```

Методы Control System Toolbox

Ниже перечислены все методы пакета, сгруппированные по назначению (назначение некоторых из них не будет понятно студенту 3-го курса, поскольку соответствующая теория проходится много позднее).

ЛСС – модели (LTI Models Function)

drss Создаёт случайную дискретную ss-модель

rss Создаёт случайную непрерывную ss-модель

dss Создаёт дескрипторную ss-модель (с матрицей E)

ss Создаёт ss-модель

tf Создаёт передаточную функцию

zpk Создаёт zpk-модель

frd Создаёт ЛСС по частотным характеристикам (FRD - model), несовместимую с остальными формами ЛСС)

frdata Извлекает частотную характеристику из FRD - модели

ssdata, dssdata Извлекает данные (матрицы) из непрерывных и дискретных ss-моделей

tfdata Извлекает данные из передаточной функции

zpkdata Извлекает данные из zpk-модели

get Запрашивает свойства ЛСС

set Устанавливает свойства ЛСС

totaldelay выдаёт полное запаздывание между входами и выходами

filt Создаёт передаточную функцию в форме, принятой для DSP Toolbox

Характеристики моделей (Model Characteristics)

class Показывает тип модели ('tf', 'zpk', 'ss', или 'frd')

hasdelay Показывает, есть ли в модели запаздывание (0 или 1)

isa Показывает, принадлежит ли модель к указанному типу (0 или 1)

isct Показывает, является ли модель непрерывной (0 или 1)

isdt Показывает, является ли модель дискретной (0 или 1)

isempty Проверяет, не пустая ли модель (0 или 1)

isproper Проверяет корректность ЛСС -модели (0 или 1)

issiso Проверяет, является ли модель одномерной (0 или 1)

ndims Показывает количество размерностей модели или массива

size Показывает размерности входов-выходов модели или число строк-столбцов массива

Преобразование моделей (Model Conversions Function)

c2d Преобразует непрерывную модель в дискретную

d2c Преобразует дискретную модель в непрерывную

d2d Изменяет период дискретизации в дискретной модели

chgunits Преобразует размерности параметров FRD - моделей

delay2z Преобразует в дискретных моделях запаздывание в полюса и нули или в FRD-модели – в фазовый сдвиг

pade Вычисляет аппроксимацию запаздывания дробно-рациональной передаточной функцией в форме Паде

frd Преобразует модель в форму FRD

ss Преобразует ЛСС (кроме FRD-модели) в ss-модель

tf Преобразует ЛСС (кроме FRD-модели) в tf-модель

zpk Преобразует ЛСС (кроме FRD-модели) в zpk-модель

reshape преобразует размерности моделей или массивов

residue Преобразует дробно-рациональную функцию, заданную векторами коэффициентов числителя и знаменателя, в сумму элементарных дробей и обратно

Упрощение моделей (Model Order Reduction)

balreal Балансирует модель за счёт изменения масштабов по каналам

minreal Вычисляет минимальную реализацию модели за счёт сокращения близких пар нулей-полюсов

modred Снижает порядок сбалансированной модели за счёт удаления малых постоянных времени

sminreal Снижает порядок структурированной (имеющей сгруппированные входы-выходы) модели

Преобразование ss-моделей (State-Space Realizations)

canon Приведение к заданной канонической форме ss-модели

sctrb Матрица управляемости

ctrbf Матрица управляемости со сгруппированными неуправляемыми переменными

obsv Матрица наблюдаемости

obsvf Матрица наблюдаемости со сгруппированными ненаблюдаемыми переменными

gram Вычисляет граммианы управляемости и наблюдаемости

ss2ss Преобразование координат в ss- модели заданным оператором

ssbal Преобразование ss-модели к сбалансированной форме

Динамические характеристики моделей (Model Dynamics)

bandwidth Вычисляет полосу пропускания одномерной модели

damp Вычисляет собственные частоты (частоты сопряжения) и соответствующие им коэффициенты демпфирования (затухания)

dcgain Вычисляет коэффициент усиления (DC) системы по постоянной составляющей

covar Вычисляет дисперсию выходного сигнала при белом шуме заданной интенсивности на входе

dsort Сортирует полюса дискретной системы по модулю

esort Сортирует полюса непрерывной системы по действительной части

iopzmap Строит графически размещение нулей и полюсов ЛСС (для каждой пары вход-выход в многомерных системах)

norm Вычисляет норму ЛСС (H_2 и L_∞)

pole, eig Вычисляет полюса ЛСС

pzmap Строит графически размещение всех нулей и полюсов ЛСС

rlocus Строит корневой годограф замкнутой системы по разомкнутой и коэффициенту обратной связи

roots Вычисляет корни полинома (по вектору коэффициентов)

sgrid,ngrid,zgrid Отображают специального вида сетку на графиках

zero Вычисляет нули ЛСС

Соединение моделей (Model Interconnections)

append Вертикальная конкатенация ЛСС

augstate Расширяет вектор выхода за счёт вектора состояния

connect Формирует модель соединением звеньев по заданной схеме

feedback Замыкание обратной связи

lft Формирование из двух ЛСС структуры "Звезда"

ord2 Формирует ЛСС второго порядка по собственной частоте и коэффициенту демпфирования

parallel Параллельное соединение 2-х ЛСС

series Последовательное соединение 2-х ЛСС

stack Объединяет несколько ЛСС в массив

Отклики во временной области (Time Responses Function)

gensig Формирует периодический входной сигнал

impulse Вычисляет и рисует весовую функцию

initial Вычисляет и рисует свободное движение ss-модели из заданных начальных условий

lsim Вычисляет и рисует отклик ЛСС на произвольный входной сигнал

ltiview Открывает LTI Viewer

step Вычисляет и рисует переходную характеристику

Частотные характеристики (Frequency Response)

allmargin вычисляет для разомкнутой ЛСС все запасы устойчивости и соответствующие частоты, определяет устойчивость

bode Вычисляет и строит ЛАФЧХ системы

bodemag Вычисляет и строи ЛАЧХ

evalfr Вычисляет $W(j\omega)$ на одной комплексной частоте

freqresp Вычисляет $W(j\omega)$ для заданного массива частот

interp Интерполирует frd-модель между узлами частоты

linspace Создаёт вектор значений с равномерным шагом

logspace Создаёт вектор значений с логарифмическим шагом

margin Вычисляет запасы по амплитуде и фазе

nichols Вычисляет и строит диаграмму Николса (амплитудно-фазовую частотную характеристику в координатах амплитуда-фаза)

nyquist Строит годограф Найквиста

ngrid Выводит специального вида сетку на диаграммах Николса и Найквиста

sigma Строит график сингулярных чисел от частоты

Порядок выполнения

Упражнение 1.

Создайте ЛСС (разомкнутую) по своему заданию на КР по ТАУ (система должна быть не ниже 3-го порядка). Преобразуйте её во все известные Вам формы представления. Сохраните созданную модель на диске для дальнейшего использования.

Упражнение 2.

Постройте для созданной системы корневой годограф, разберитесь, какую информацию несёт сетка на этой диаграмме. Попробуйте определить по корневому годографу, при каких значениях коэффициента усиления система будет устойчивой, при каких – не иметь перерегулирования.

Упражнение 3.

Постройте годограф Найквиста. Разберитесь, какую информацию несёт сетка на этом годографе. Попробуйте определить, какой максимальный подъём и на какой частоте будет иметь амплитудная характеристика разомкнутой системы, замкнутой системы.

Упражнение 4.

Постройте ЛАФЧХ своей системы, определите по ней запасы устойчивости.

Упражнение 5.

Постройте диаграмму Николса. Разберитесь, какую информацию несёт сетка на этой диаграмме. Определите запасы устойчивости, максимальный подъём амплитудной характеристики разомкнутой и замкнутой системы.

Упражнение 6.

Постройте переходную характеристику и весовую функцию замкнутой системы. Определите время регулирования и перерегулирование. Сравните с результатами из упражнения 2. Объясните несовпадение, если таковое обнаружится.

Упражнение 7.

Добавьте в свою систему запаздывание и посмотрите, какие изменения при этом произойдут в характеристиках системы (запасы устойчивости, максимальные подъёмы амплитуды, время регулирования, перерегулирование, колебательность).

Контрольные вопросы

1. Какие диаграммы можно строить в Matlab для линейных систем?
2. Какую информацию содержит диаграмма Николса?
3. Как построить весовую функцию замкнутой системы, имея разомкнутую?
4. Как определить степень устойчивости и колебательность с помощью Matlab?
5. Как вычислить запасы устойчивости без рисования графиков?
6. Как определить полосу пропускания системы без рисования графиков?
7. Как построить график амплитудно-частотной характеристики по ошибке?
8. Как включить-выключить сетку на графиках, отобразить-спрятать некоторые характерные точки на графиках?
9. Как построить несколько графиков в одной системе координат?

14 Использование визуальных средств синтеза линейной системы управления для расчёта корректирующего фильтра. *SISO design.*

Теоретические сведения

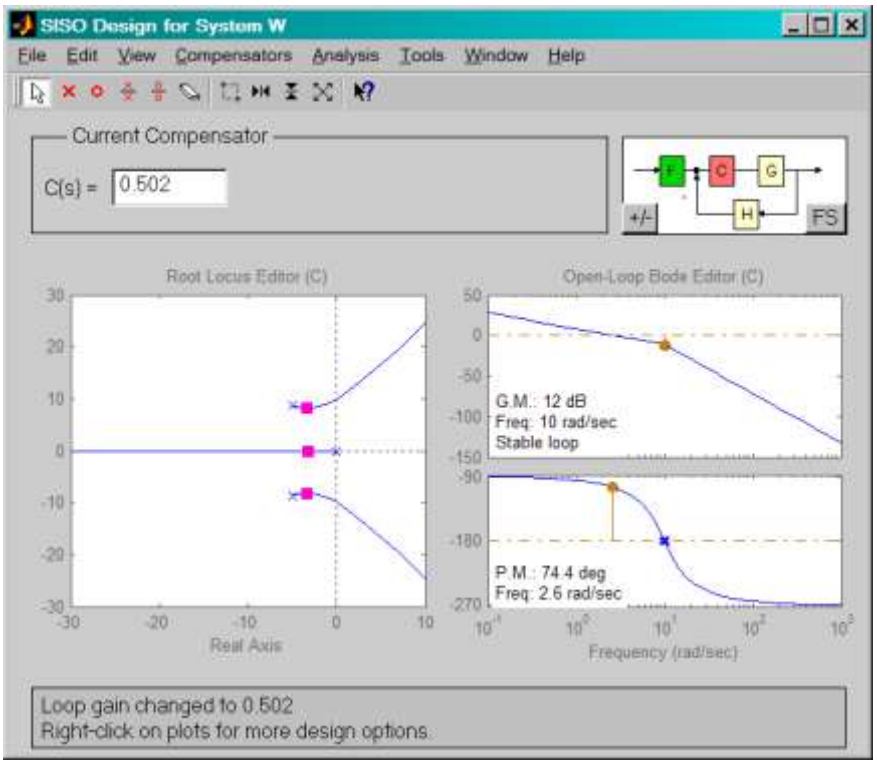
Визуальный синтез в частотной области

В Matlab в составе Control System Toolbox имеется визуальный инструмент SISO Design Tool, позволяющий выбирать параметры регулятора, наблюдая за изменениями в частотных характеристиках и расположением нулей и полюсов замкнутой системы, вызываемые изменением этих параметров. Вызвать этот инструмент можно командой "sisotool", которой в качестве параметра можно передать имя ЛСС – объекта управления (неизменяемой части) проектируемой системы.

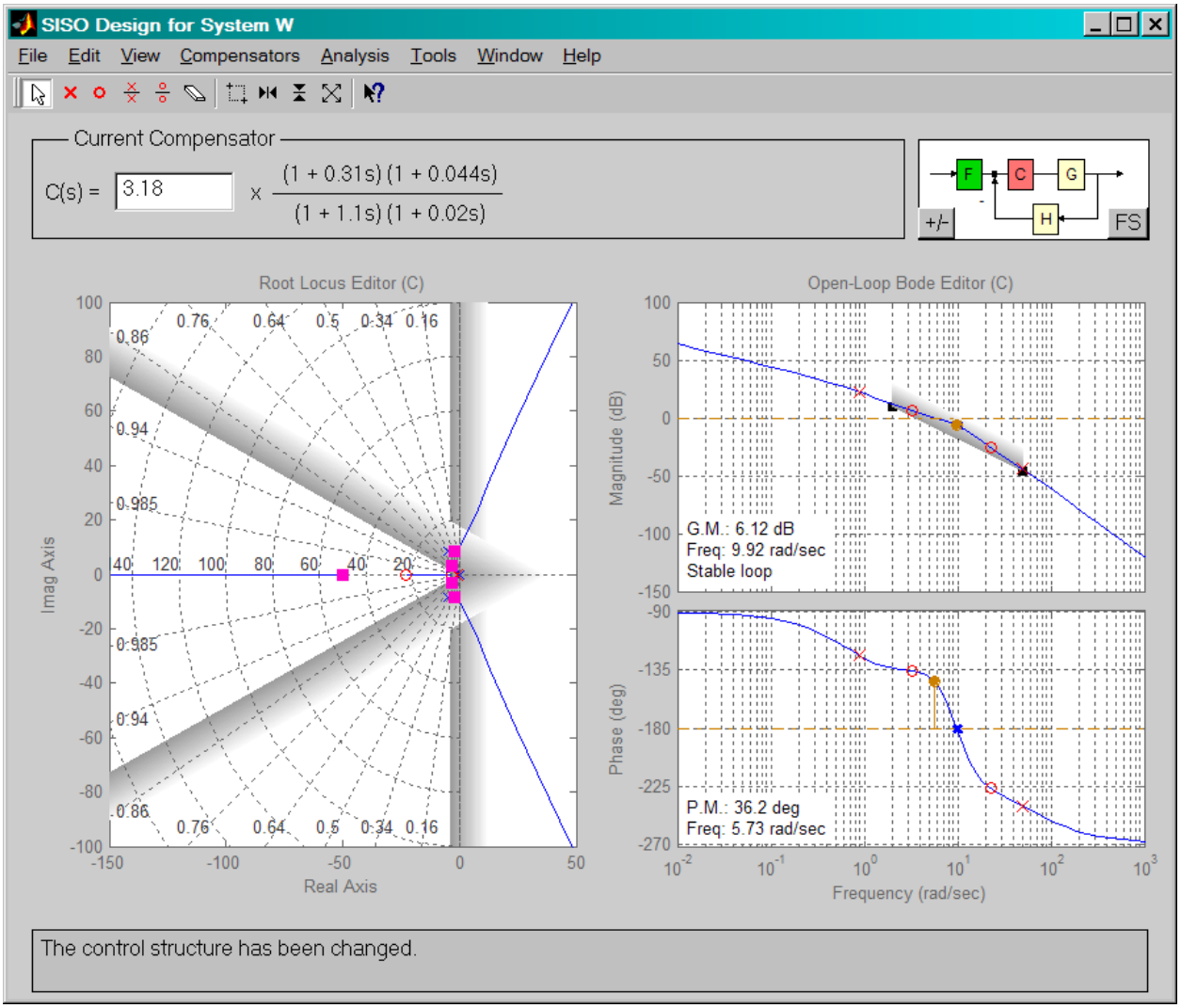
Пример 3

```
>> sisotool(W)
```

Появляется вот такая картинка:



После настройки и ряда манипуляций с регулятором картинка приобретает такой вид:



Разберёмся, что имеется (или может иметься) на этой картинке. Прежде всего, в правом верхнем углу имеется схема, с которой производится работа. В ней присутствуют четыре блока:

G – неизменяемая часть, объект управления (Plant Model);

C – корректирующий фильтр, то, что и требуется синтезировать (Compensator);

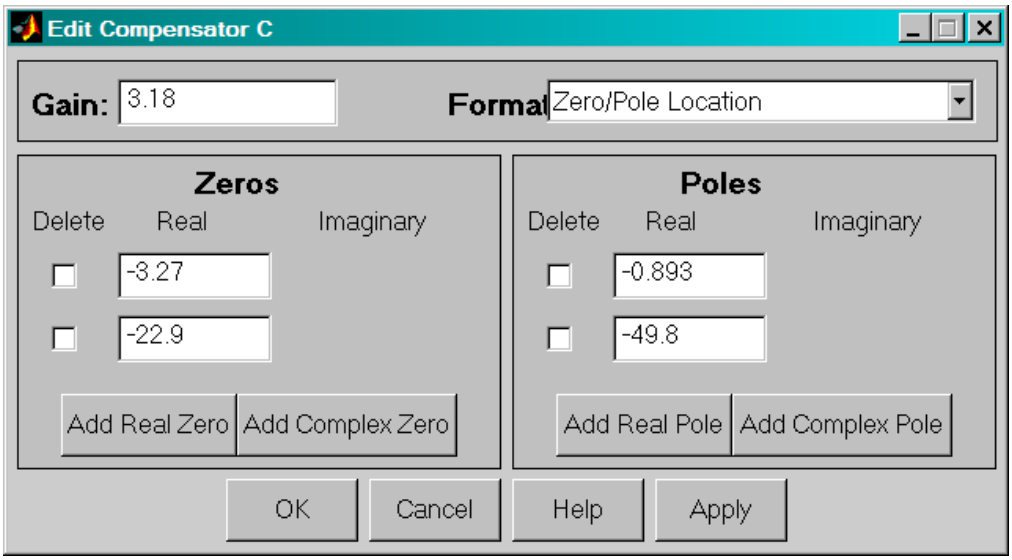
H – чувствительный элемент (Sensor), который тоже относится к неизменяемой части;

F – предварительный преобразователь (Prefiltr) входного сигнала, который реализует комбинированное управление в нашей терминологии. Тоже можно настраивать.

Нажимая кнопку FS, можно менять структуру системы – вместо последовательной коррекции применять параллельную, менять точку подключения префильтра. Кнопка +/- меняет знак обратной связи.

Щёлкнув мышкой по любому элементу схемы, можно открыть диалоговое окно, которое показывает (раскрывает) модель этого блока, а для компенсаторов ещё и позволяет изменять их, меняя коэффициент

усиления, положение полюсов и нулей, добавляя новые полюса и нули или удаляя старые.



Слева от структуры проектируемой системы отображается передаточная функция корректирующего фильтра, в которой можно задавать коэффициент усиления.

Кроме схемы на панели SISO Design Tool размещено несколько диаграмм, состав и количество которых можно настраивать. Это всё уже знакомые по предыдущей работе диаграммы Боде, Найквиста, Николса, переходная характеристика, корневой годограф. Только здесь они снабжены дополнительной функциональностью. На диаграмме Боде показываются нули и полюса регулятора, которые можно перемещать мышкой, и эти изменения тут же отображаются на остальных диаграммах. Кроме того на них можно размещать границы допустимых (запретных) областей (Constraints), в которых должны размещаться диаграммы или их характерные части. Эти границы, впрочем, чисто визуальные, обеспечить их соблюдение можно только вручную. При настройке границ кроме уже перечисленных в предыдущей работе терминов (см. Глоссарий) используются следующие:

Slope – наклон (дБ/дек)

Damping Ratio – отношение действительной части корня к мнимой (обратная величина по отношению к принятой у нас "колебательности" μ).

Upper Gain Limit – верхняя граница частотного коэффициента передачи.

Lower Gain Limit – нижняя граница частотного коэффициента передачи.

Constraint – ограничение.

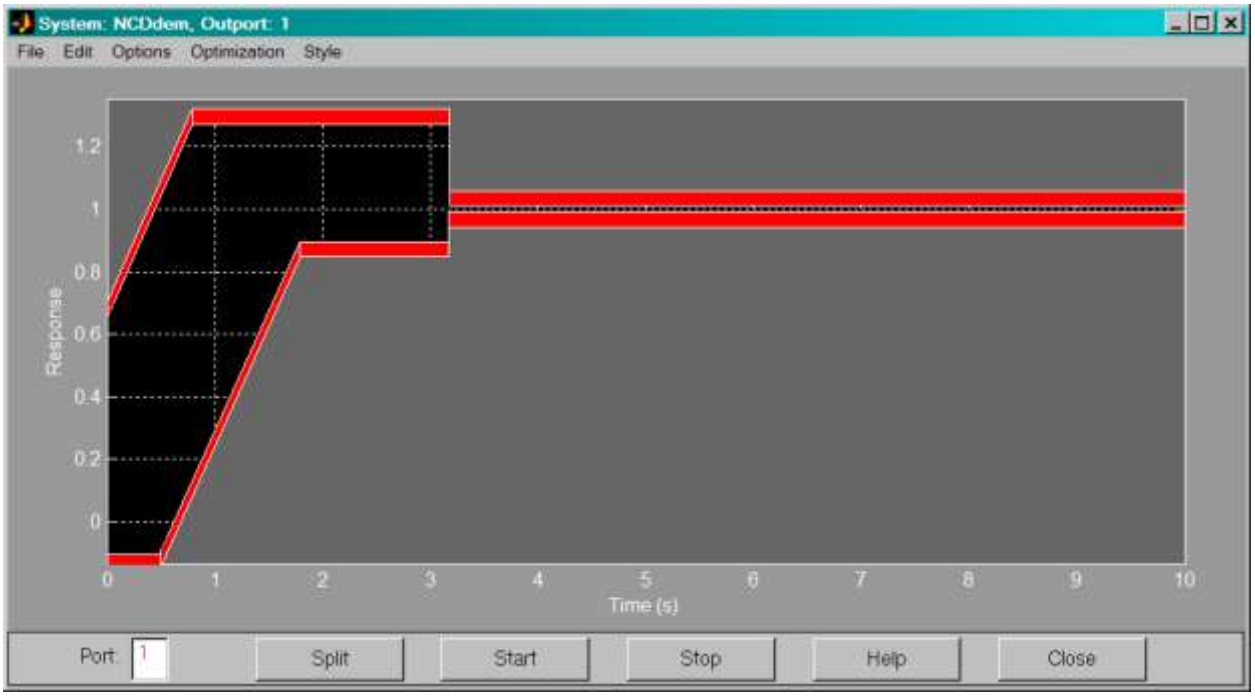
Инструмент позволяет импортировать ЛСС из рабочего пространства и из диаграмм *Simulink*. Кроме того возможен экспорт из

этого инструмента как отдельных блоков в рабочее пространство *Matlab*, так и всей системы – в диаграмму *Simulink*, с тем, чтобы проверить там результаты синтеза разносторонним моделированием.

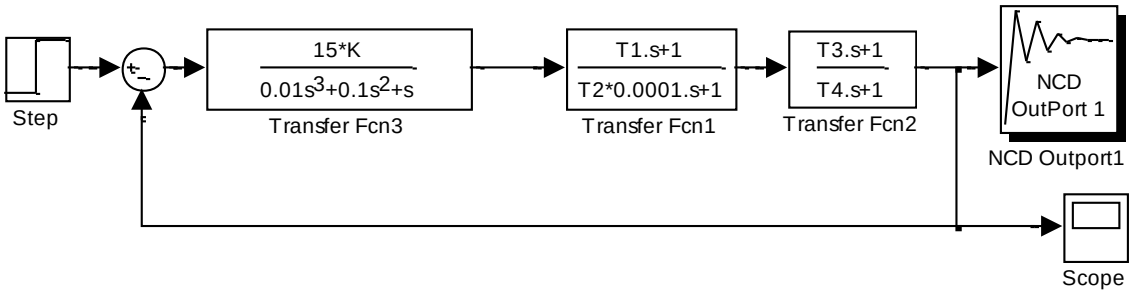
Синтез регулятора во временной области

Эта возможность реализуется средствами Simulink. Среди наборов (BlockSet) блоков этой системы в версиях Matlab от 5.3 до 6.5 имелась группа NCD (Nonlinear Control Design), содержащая, впрочем, фактически один блок с тем же именем, NCD. В более поздних версиях этот блок исчез, но остались заготовки для него, с помощью которых можно вручную повторить его функции. Причина этого – невысокая надёжность этого блока при высокой сложности.

Рассмотрим задачу, которую призван решать этот блок. Пусть у нас есть некоторые требования к системе, заданные на языке временных показателей качества. С помощью этих требований можно построить на плоскости "время-выход" допустимую область, в которой должна располагаться переходная характеристика. Например, вот так.



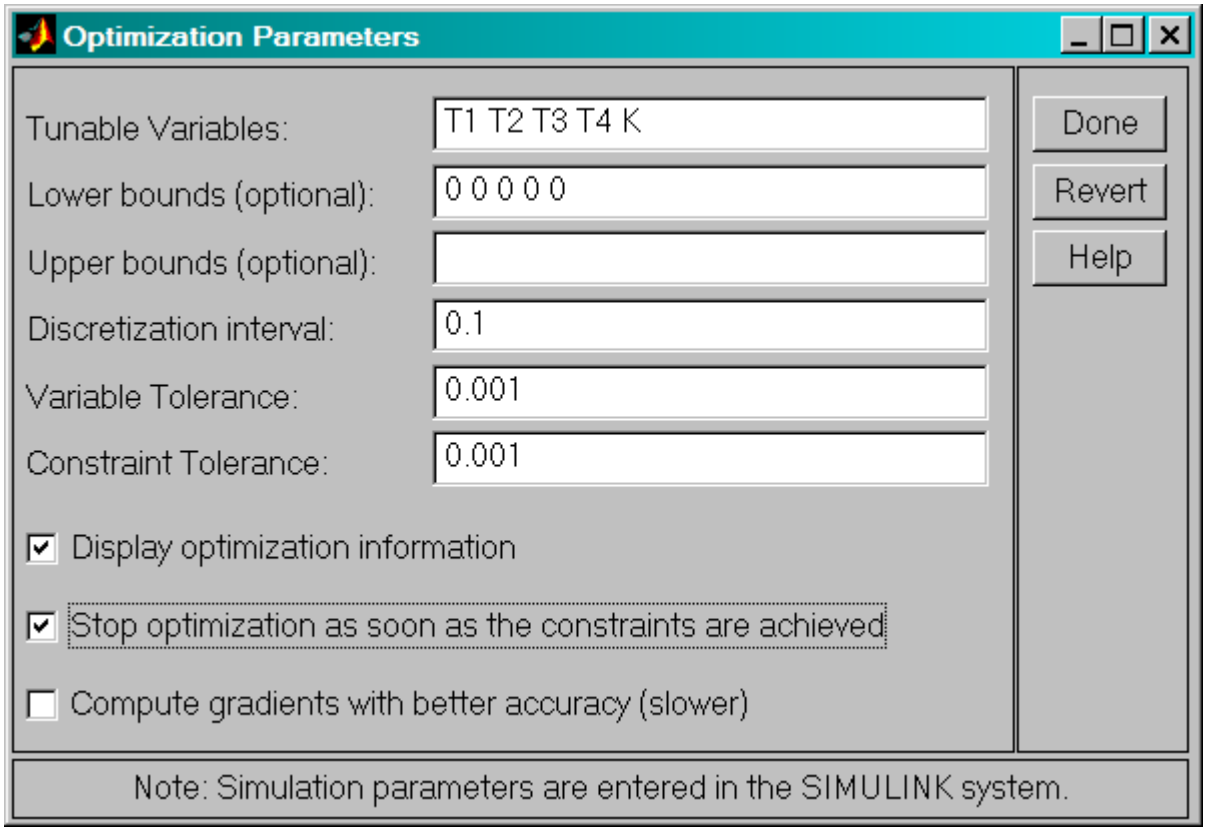
Теперь составим схему моделирования для Simulink, в которой на выход подключим NCD-блок. Вот так.



Здесь первое звено 3-го порядка представляет собой неизменяемую часть, остальные – корректирующие фильтры, подлежащие настройке. Другими словами, надо подобрать такие значения параметров T1, T2, T3, T4 и K, чтобы переходная характеристика (вход блока NCD) располагалась в разрешённой области, ограниченной красными линиями. Начальные значения этих переменных зададим в командном окне:

```
>>T1=0.4;T2=0.01;T3=0.3;T4=0.2;
>>
```

Это задача оптимизационная, сводящаяся к минимизации "выходов за границу" переходной характеристики. Настроим параметры оптимизации, вызвав пункт меню **Optimization=>Parameters.**



Tunable variables – перечень настраиваемых переменных (должны быть определены в рабочем пространстве и участвовать в Simulink-модели).

Lower bounds (optional) – нижние границы допустимых значений настраиваемых переменных (необязательны).

Upper bounds (optional) – верхние границы (необязательны).

Discretization interval – интервал времени, с которым проверяются ограничения.

Variable Tolerance – точность вычисления настраиваемых переменных.

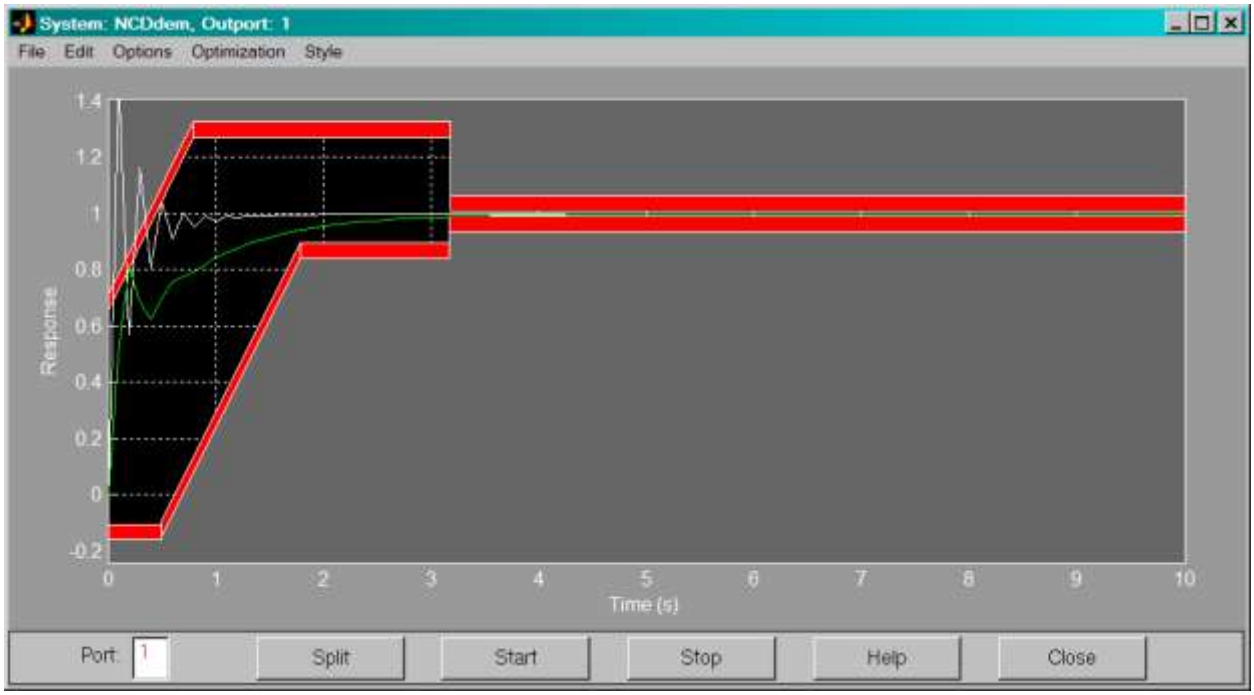
Constraint Tolerance – точность, с которой проверяются ограничения.

Display optimization information – отображение промежуточных результатов.

Stop optimization as soon as the constraints are achieved – остановить оптимизацию, как только ограничения будут выполнены.

Compute gradients with better accuracy (slower) – более точное вычисление градиента (работает медленнее).

Запустив оптимизацию кнопкой Start и подождав некоторое время (может быть довольно значительным), можно при определённом везении получить результат, во-первых, графический (зелёным показан процесс в синтезированной системе):



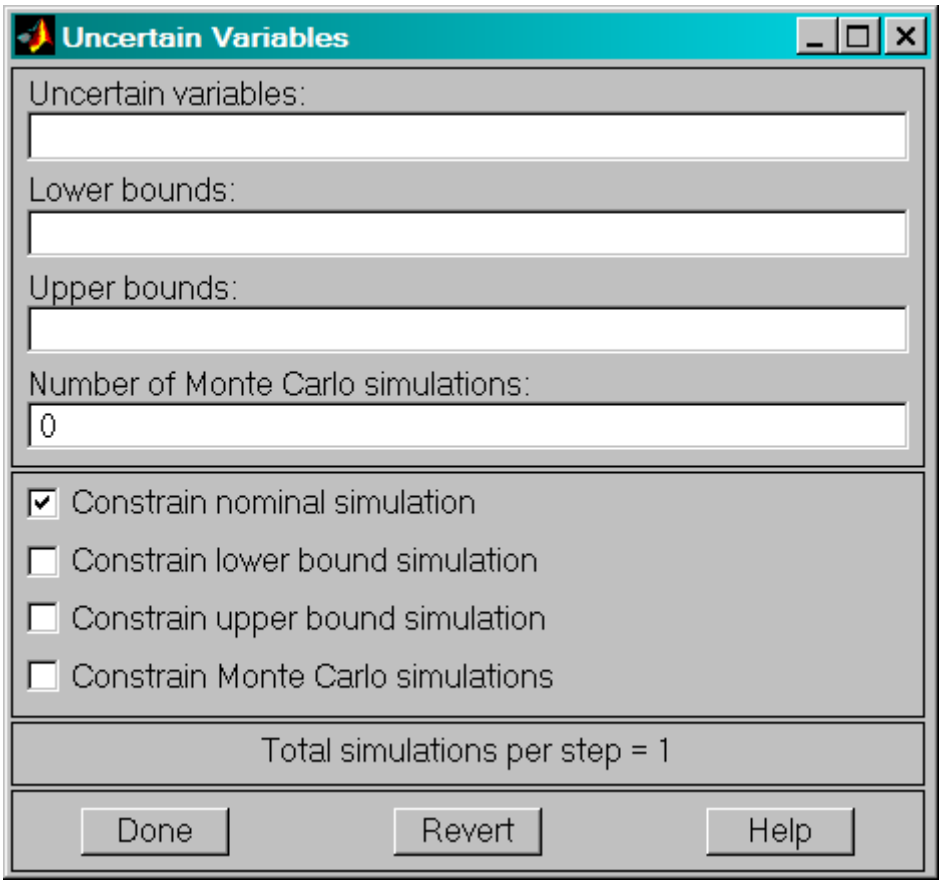
Во-вторых, числовой, настраиваемые переменные получили новые, оптимальные значения:

```
>> T1,T2,T3,T4,K
T1 =
   0.2378
T2 =
   0.0363
T3 =
   0.3103
```

```
T4 =
   0.0917
K =
   0.1528
>>
```

Обратите внимание на то, как используется переменная T2 в модели. Этот приём позволяет сделать сравнимыми масштабы по всем настраиваемым переменным, что облегчает численную оптимизацию.

Блок NCD позволяет не только настраивать номинальные значения параметров регулятора, но и учитывать возможные неопределённости (разбросы, допуски) в параметрах модели. Открыв меню **Optimization=>Uncertainty…** можно настроить эти неопределенности в диалоговом окне:



где можно перечислить нестабильные переменные, задать границы их изменения и указать режимы моделирования.

Порядок выполнения.

Упражнение 1.

Выполните с помощью инструмента *SISO Design Tool* синтез своей системы из задания на КП по ТАУ в соответствии со своим заданием.

Экспортируйте синтезированную систему в Simulink и проведите моделирование, подавая на вход сигналы различной формы (синус, ступеньку, пилу).

Сохраните на диске результаты синтеза и схему моделирования.

Упражнение 2.

Скорректируйте свою систему из КП по ТАУ с помощью блока *NCD*, задав требования из задания и добавив к ним разброс параметров неизменяемой части в пределах ± 10% от номинала. Получите результаты моделирования синтезированной системы при ступенчатом и гармоническом сигналах на входе. Результаты сохраните в doc-файле.

Контрольные вопросы

1. Какие методы синтеза САУ реализованы в Matlab?
2. Как получить модель скорректированной системы, имея модель неизменяемой части системы и параметры регулятора?
3. Как определить степень устойчивости и колебательность с помощью Matlab? Как добиться желаемого значения этих характеристик?
4. Можно ли с помощью Siso design tool рассчитать параллельное корректирующее устройство? Есть ли при этом какие-то ограничения? А можно ли там решать задачу модального управления?
5. Как определить полосу пропускания системы без рисования графиков?
6. Как построить график амплитудно-частотной характеристики по ошибке в рамках Siso design tool?
7. Можно ли синтезировать с помощью NCD-блока регулятор, обеспечивающий заданную точность воспроизведения гармонического сигнала?

15 Средства анализа и синтеза систем управления в пространстве состояний.

Теоретические сведения

MATLAB® содержит несколько инструментов синтеза (расчёта) регуляторов, из которых мы рассмотрим только четыре. Два из них не имеют графического интерфейса и реализуются в обычном интерактивном режиме через командное окно Matlab. Эти средства реализуют метод АКОР (аналитическое конструирование регуляторов, функция lqr, lqry) и метод модального управления (acker, place). Два других имеют графический интерфейс и реализуют методы, не имеющие установившегося названия. Один из них использует частотные характеристики (SISO Design Tool), другой – временную область (NCD).

АКОР

Пусть ЛСС задана в пространстве состояний уравнениями

$$\dot{x} = Ax + Bu,$$

$$y = Cx + Du,$$

Здесь x, y и u – вектора состояния, выхода и входа, а A, B, C и D – матрицы соответствующих размерностей. Это модель разомкнутой системы. Замкнём её линейным регулятором нулевого порядка (статическими обратными связями) в форме

$$u = Kx.$$

Регулятор представляет собой вектор обратных связей K. Этот вектор будем искать таким образом, чтобы возникающие в этой системе процессы минимизировали квадратичный функционал качества

$$J = \int_0^\infty \left(x^{\mathrm{T}} Q x + u^{\mathrm{T}} R u + 2 x^{\mathrm{T}} N u \right) dt,$$

где Q и R – неотрицательно определённые матрицы соответствующей размерности. Матрица N чаще всего выбирается нулевой. Известно, что оптимальный регулятор для линейной управляемой системы всегда существует и может быть найден с помощью решения так называемого матричного уравнения Риккати

$$S \cdot A + A^{\mathrm{T}} \cdot S - \left(S \cdot B + N \right) \cdot R^{-1} \cdot \left(B^{\mathrm{T}} S + N \right) + Q = 0$$

Расчёт оптимального регулятора осуществляет функция lqr, которой надо передать ЛСС в форме пространства состояний и матрицы Q, R и N, определяющие критерий качества. Эта функция возвращает кроме вектора обратных связей K, ещё решение уравнения Риккати S и вектор характеристических чисел замкнутой системы E. Матрицу N можно не задавать, тогда она будет считаться нулевой. Рассмотрим пример подобного синтеза.

Пример 1

Задаём передаточную функцию разомкнутой системы

```
W=tf(5,[0.01 0.1 1 0])

Transfer function:
        5
----------------------
0.01 s^3 + 0.1 s^2 + s
```

Функции синтеза регулятора требуют, чтобы модель объекта управления была задана в пространстве состояний, поэтому преобразуем её в ss-форму.

```
>> [A B C D]=ssdata(W)
A =
                -10              -3.125              0
                 32                 0                 0
                  0                16                 0
B =
    1
    0
    0
C =
                  0                 0            0.9765625
D =
    0
>>Wss=ss(A, B, C, D);
```

Это преобразование можно было бы сделать и короче, но нам ещё пригодятся явно заданные матрицы модели в пространстве состояний.

Зададим матрицы для вычисления функционала качества

```
>> Q=[1 0 0; 0 1 0; 0 0 1]
Q =
    1    0    0
    0    1    0
    0    0    1
>> R=2
R =
    2
>>
```

Матрица Q должна быть Эрмитовой (симметричной) и положительно полуопределённой[1]. Очевидно, единичная матрица такими свойствами обладает. Впрочем, нужными свойствами обладает любая диагональная матрица с неотрицательными элементами.

Выполняем вычисление оптимального регулятора.

```
>> [K S E]=lqr (Wss,Q,R)

K =

      3.64118        1.33721        0.70710

S =

      7.28237        2.67443        1.41421
      2.67443        1.14414        0.60286
```

[1] Это требование означает, что при любом векторе *x* квадратичная форма $x^{T}Qx$ неотрицательна.

```
        1.41421          0.60286          0.39440

E =

        -5.150403 +      9.04741i
        -5.150403 -      9.04741i
       -3.340382

>>
```

Теперь надо собрать замкнутую систему, которая будет описываться уравнениями

$$\dot{x} = Ax + B(-Kx + u),$$
$$y = Cx + D(-Kx + u),$$

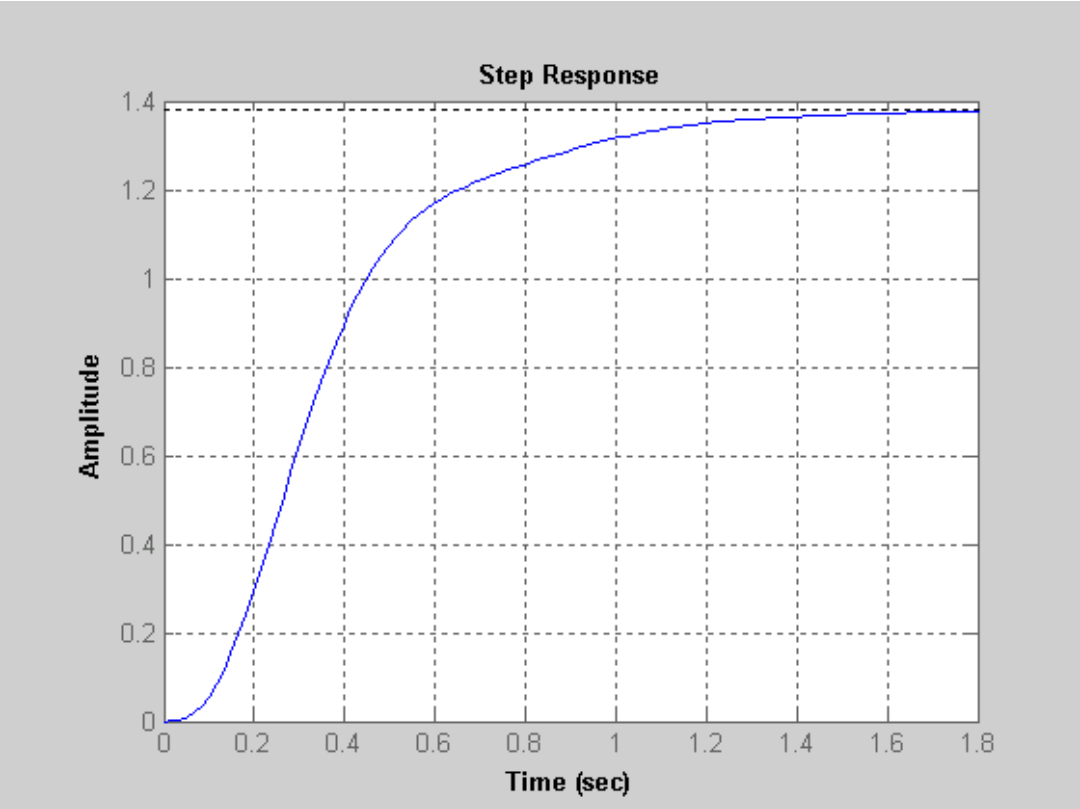
которую можно переписать также

$$\dot{x} = (A - BK)x + Bu,$$
$$y = (C - DK)x + Du.$$

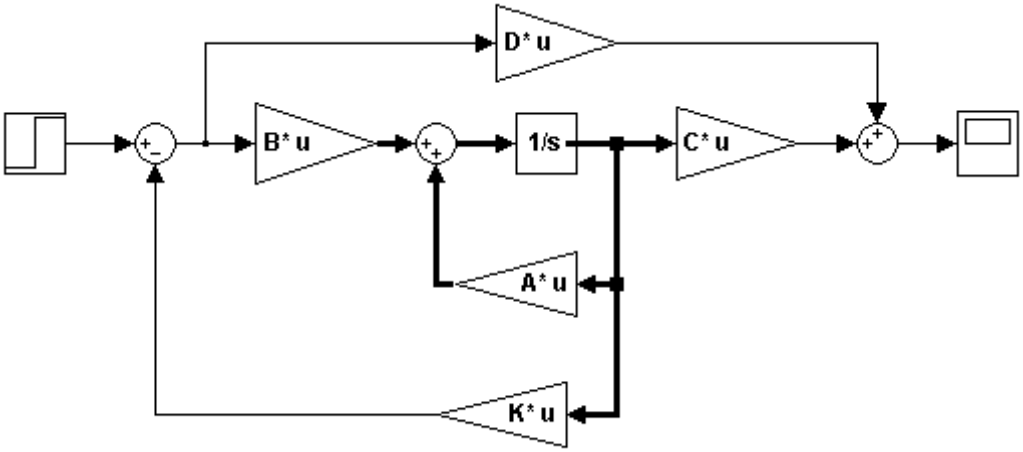
```
>> Wssz=ss(A-B*K, B, C-D*K, D)

a =
          x1       x2       x3
  x1   -13.64   -4.462  -0.7071
  x2       32        0        0
  x3        0       16        0

b =
      u1
  x1   1
  x2   0
  x3   0

c =
          x1       x2       x3
  y1       0        0   0.9766

d =
      u1
  y1   0

Continuous-time model.
>>
```

Теперь можно посмотреть, например, переходную характеристику замкнутой системы.

```
>>step(Wssz)
```

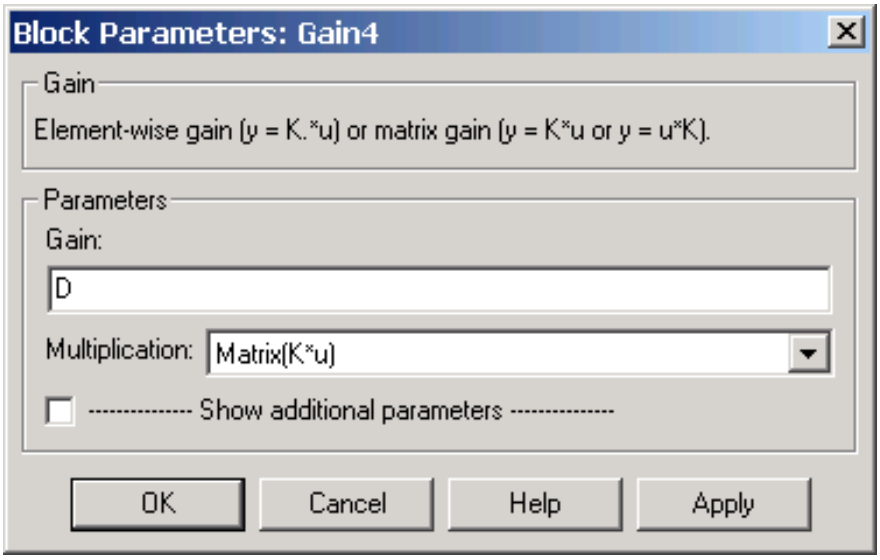


Более детальное исследование замкнутой системы можно выполнить с помощью пакета Simulink. Наберём приведённую ниже схему. В неё жирными линиями выделены нескалярные (векторные) линии связи.



Для того, чтобы схема работала, надо все усилители настроить на матричное умножение, то есть в окне настройки выбрать

Multiplication Matrix (K*u).



В усилителях в поле "Gain" надо установить имена тех матриц, которые определены нами ранее в командном окне.

Модальное управление

Оно заключается в таком выборе коэффициентов обратных связей, которые обеспечивают заданное расположение полюсов замкнутой системы. Разомкнутая система и регулятор – такие же, как и в методе АКОР. Только регулятор вычисляется функцией acker для одномерных систем невысокого (до десятого) порядка или функцией place для многомерных систем или систем высокого порядка.

Пример 2

Используем определённую в предыдущем примере ss-систему Wss. Для вызова функции acker нужны только две матрицы, *A* и *B*. В качестве требуемых полюсов используем полученные в предыдущем примере полюса замкнутой системы, то есть вектор *E*. Очевидно, должен получиться тот же регулятор, что и в предыдущем примере.

```
>> acker(Wss.a, Wss.b, E)
ans =
        3.64118        1.33721        0.70710
>>
```

Действительно, получился тот же самый регулятор.

Порядок выполнения

Упражнение 1.

Создайте ЛСС (разомкнутую) по своему заданию на КР по ТАУ (система должна быть не ниже 3-го порядка). Получите методом АКОР оптимальный регулятор для этой системы и промоделируйте её в Simulink

при разных входных сигналах. Оцените точность и показатели качества регулирования. Установите, как зависят эти показатели от соотношения весов при управлении и координатах состояния в функционале качества (то есть матрицы Q и R).

Упражнение 2.

Получите для своей системы регулятор, соответствующий следующим наборам полюсов:

-5 -3 -2i -3 +2i
-5 -6 -4i -6 +4i
-2.5 -3 -2i -3 +2i

Исследуйте динамику синтезированной системы и сделайте выводы о том, как влияет расположение полюсов на показатели качества системы и её точность.

Контрольные вопросы

1. Какие методы синтеза САУ реализованы в Matlab?
2. Для чего требуется преобразовывать систему в форму State Space при решении задачи АКОР?
3. Как получить модель скорректированной системы, имея модель неизменяемой части системы и параметры регулятора?
4. Возможно ли моделировать систему, синтезированную методом АКОР, в другом виде по сравнению с приведённым в этом пособии?
5. Как задать требуемое положение полюсов замкнутой системы в задаче модального управления? Как определить, удачно ли выбрано это расположение?
6. Как определить степень устойчивости и колебательность с помощью Matlab? Как добиться желаемого значения этих характеристик?
7. Как определить полосу пропускания системы без рисования графиков?

16 Визуальный инструмент синтеза системы Simulink.

ВВЕДЕНИЕ

Для синтеза нелинейных систем управления трудно переоценить значение алгоритма поиска значений параметров, обеспечивающих требуемое поведение систем. Такой алгоритм поддерживается модулем Response Optimization. входящем в состав пакета Simulink Design Optimization среды MatLab. Модуль оптимизирует линейные и нелинейные системы управления автоматической подстройкой заданных параметров модели по ее реакции. Автоматическая оптимизация повышает точность и

быстродействие систем, сокращает время проектирования систем управления.

В этой работе на примере оптимизации модели показаны основные этапы работы с пакетом Simulink Response Optimization.

ОБЩИЕ СВЕДЕНИЯ

Simulink Response Optimization работает с линейными и нелинейными, дискретными и непрерывными моделями. Пакет выполняет оптимизацию модели в циклической последовательности. На каждом шаге он запускает модель, сравнивает переходный процесс выбранной переменной состояния модели с заданными ограничениями, показывает промежуточные результаты оптимизации, подстраивает параметры модели в соответствии с выбранным алгоритмом численной оптимизации и вновь запускает модель с уточненными параметрами и так до достижения заданной формы выходного процесса (response.

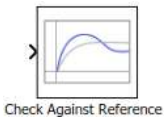
Перед началом оптимизации пользователь выбирает параметры модели, которые необходимо подстраивать в процессе оптимизации, задает диапазон изменения параметров, выбирает метод оптимизации (градиентный метод, Монте Карло), назначает критерии оптимизации (например, величину перерегулирования или время переходного процесса) и их величины в виде коридора ограничений переходного процесса (например, см. Рис. 6).

Для оптимизации во временной области можно расставить на графике отклика уровни – границы коридора в который должен попадать оптимизируемый сигнал. В библиотеке Simulink > Signal Design Optimization > Signal Constraints имеются три блока задания коридоров:

В блоке Check Custom Bounds

верхние (upper) и нижние (lower) границы (bounds) задаются матрицами амплитуд горизонтальных линий (например, [1.3 1.3; 1.12 1.12]) и соответствующих временных (в

секундах) точек начала и концов линий (например, [0 8.5; 8.5 11.4]) По умолчанию задание нижних границ выключено. Для того, чтобы их задать необходимо поставить галочку возле Include lower bound in assertion (Рис.).

В блоке Check Against Reference 

коридор задается интервалом времени (например, linspace(0,10)), функцией эталонного переходного процесса (например, 1-exp(-linspace(0,10)/2)), абсолютным и относительным отклонениями соответственно от оси и от графика функции.

В блоке Check Step Response Characteristics

коридор рассчитывается после входа в закладке Bounds основных показателей качества переходного процесса при типовом (ступенчатом) воздействии (Рис. 1.)

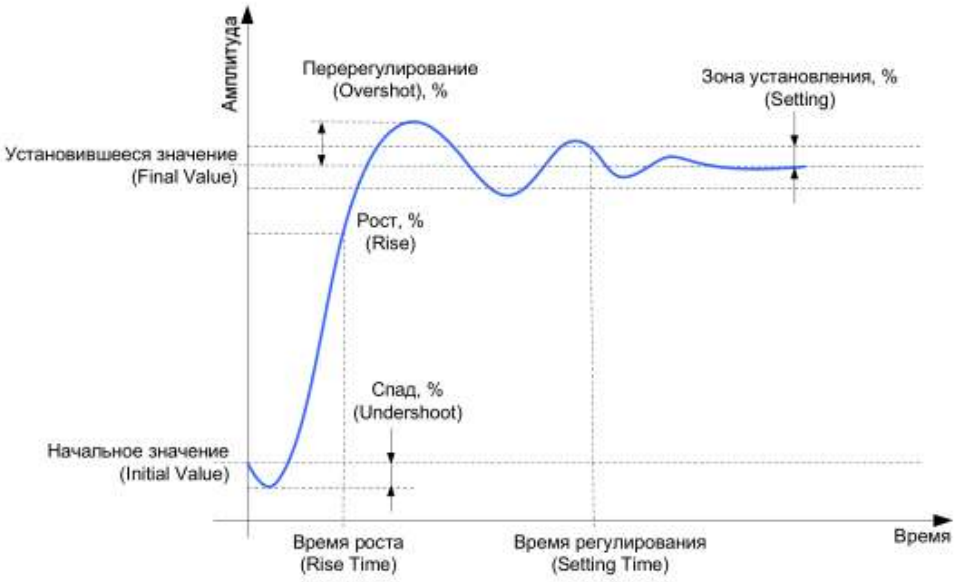


Рис. 1. Основные показатели качества переходного процесса при типовом (ступенчатом) воздействии задаваемые в блоке Check Step Response Characteristics для расчета коридора отклика системы.

Во всех блоках Signal Constraints кнопкой Show Plot окна Bound (вызывается двойным кликом по блоку) открывается график с заданными границами коридора отклика модели (рисунок 2).

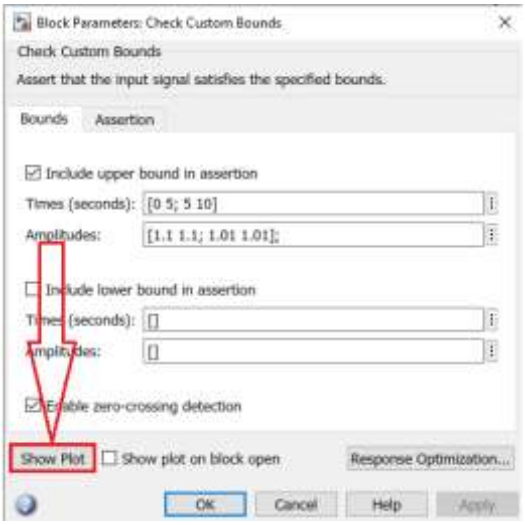


Рис 2. Интерфейс блока Check Custom Bounds

С помощью меню, мыши или горячих клавиш границы коридора можно перемещать, разворачивать, разбивать, укорачивать и удлинять до требуемого размера. После установки приемлемых границ, кнопкой Update block необходимо зафиксировать новые границы в статусе исходных границ для поиска оптимального переходного процесса. При этом, ранее введенные значения границ на закладке Bound будут пересчитаны (Рисунок 3).

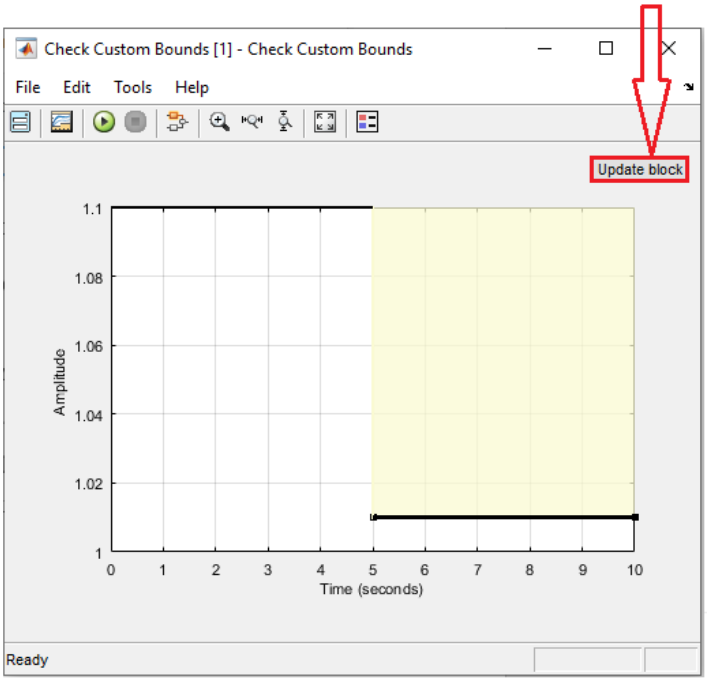


Рис. 3. Интерфейс Show Plot блока Check Custom Bounds

ПОСЛЕДОВАТЕЛЬНОСТЬ ДЕЙСТВИЙ ОПТИМИЗАЦИИ ПАРАМЕТРОВ МОДЕЛИ

Для поиска оптимальных параметров модели при помощи модуля Response Optimization пакета Simulink Design Optimization следует.

1. В рабочей области (workspace) MATLAB задать начальные, наилучшие, по предварительной оценке, значения параметров модели, которые необходимо оптимизировать;

2. Заменить соответствующие значения оптимизируемых параметров модели названиями переменных workspace;

3. Подсоединить блок(и) Constraint ко всем откликам (переменным состояния) модели, формы которых должны соответствовать заданным ограничениям;

4. Задать ограничения - коридоры откликов;

5. Ввести в окно Response Optimization > Design Variables Set параметры модели, которые необходимо оптимизировать и указать диапазон поиска параметров;

7. Подать на вход модели воздействие (ступенчатое, импульсное, др.), для которого требуется получить желаемый переходной процесс в точках с Constraint блоками.

8. Командой Evaluate Requirements

построить переходные процессы для начальных значений параметров относительно коридора ограничений.

8. Нажать кнопку

для запуска процесса оптимизации.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Построить модель контура стабилизации в Simulink

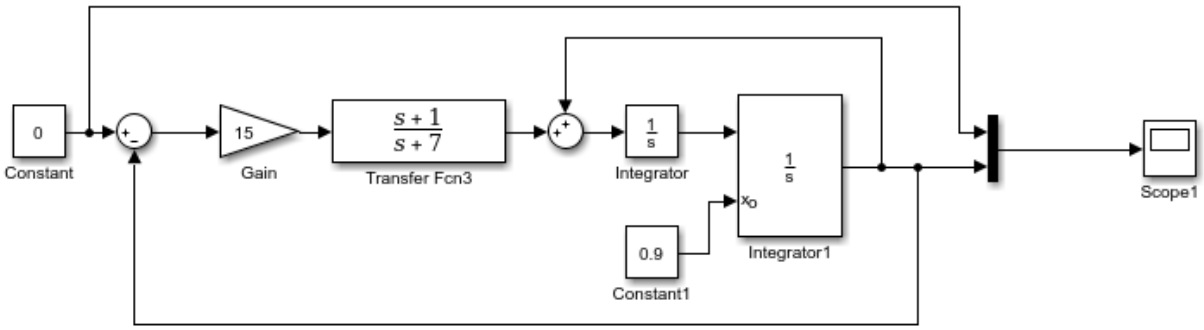


Рис. 4. Модель стабилизации маятника в верхнем положении

2. Построить график переходного процесса

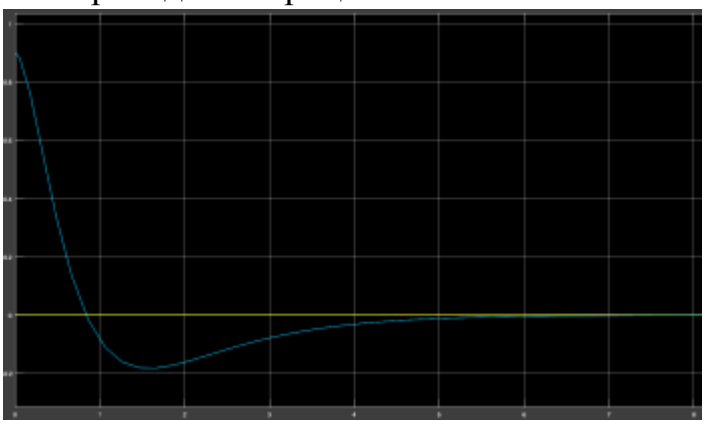


Рис. 5. Модель стабилизации маятника в верхнем положении

3. Замените числовые значения блоков параметров регулятора модели на переменные
4. Объявите переменные в Workspace с начальными значениями

Рис. 6. Модель стабилизации маятника в верхнем положении

5. Добавьте в модель блок, например, Check Custom Bounds из библиотеки Simulink > Signal Design Optimization > Signal Constraints и подключите его к выходу системы.

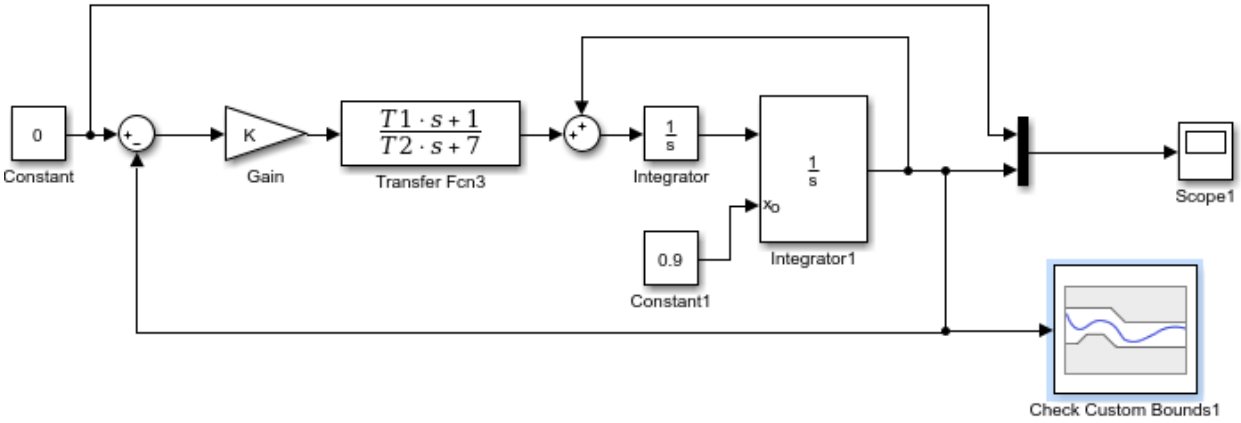
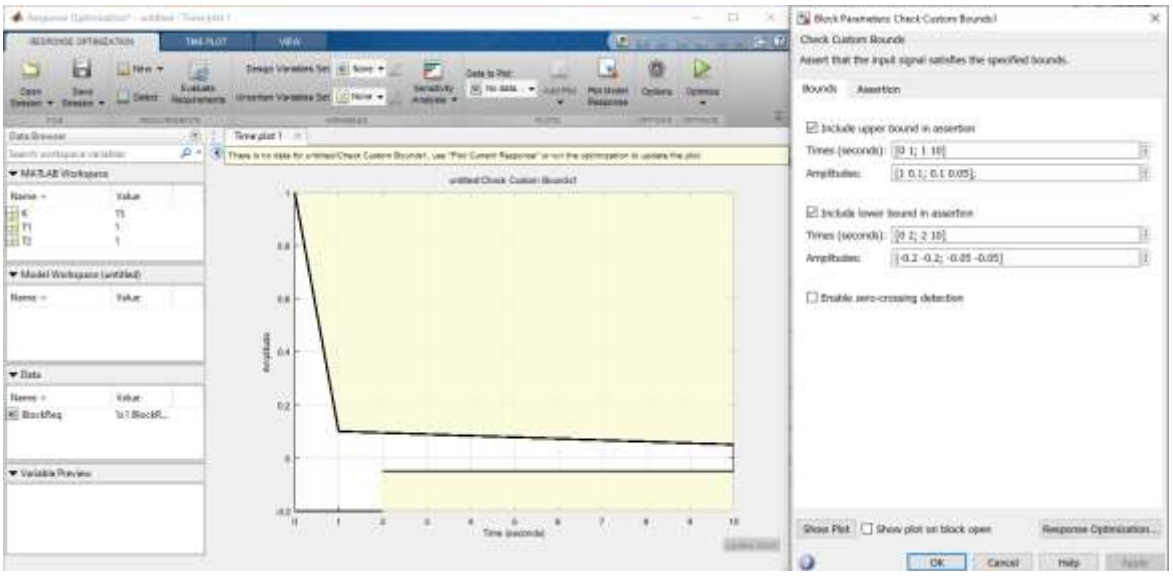
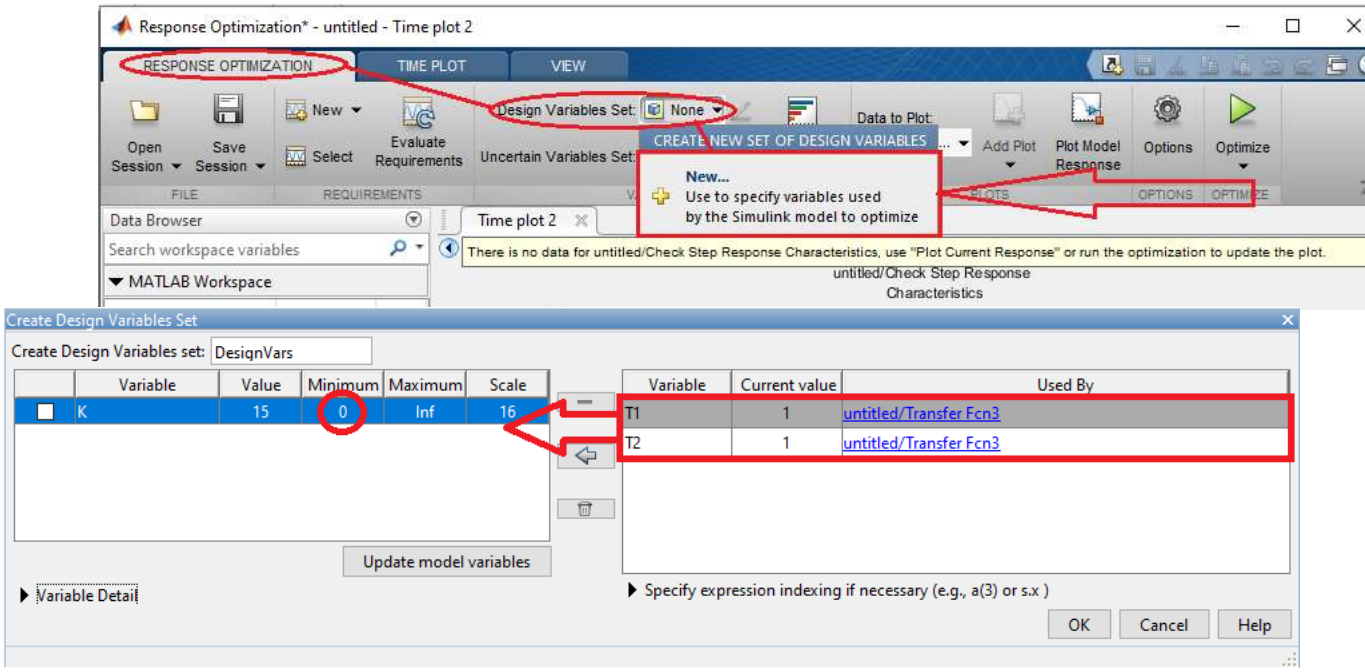


Рис. 7. Модель стабилизации маятника в верхнем положении

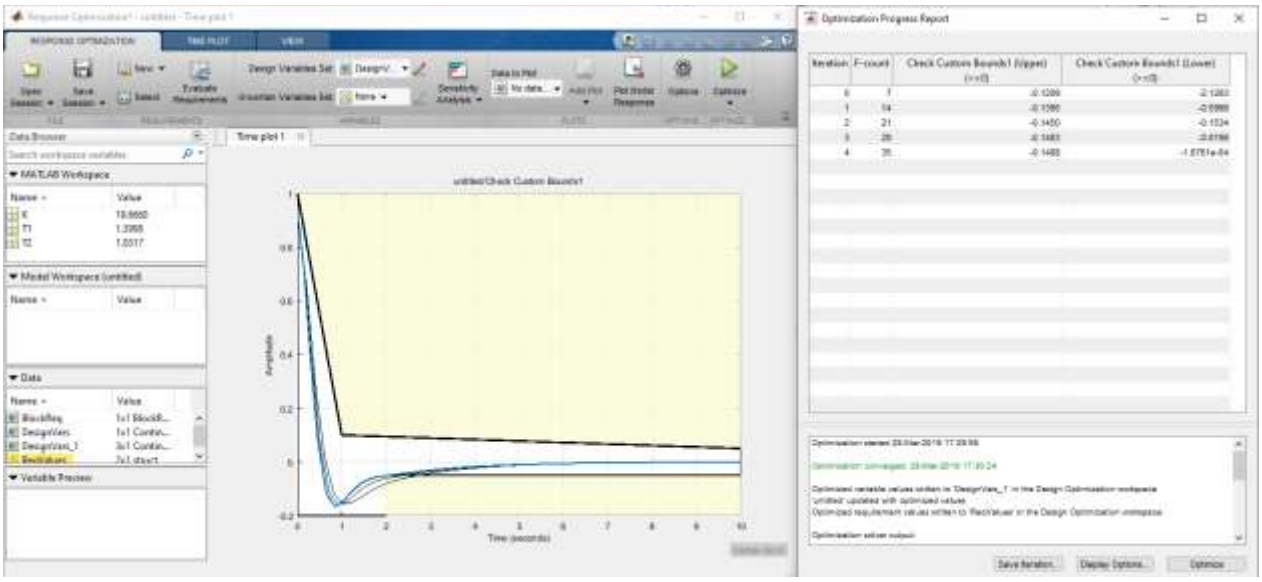
6. Используя исходные данные задайте допустимые границы коридора движения переменной. Постройте границы коридора введя команду Show Plot закладки Bounds.



7. Нажмите клавишу Response Optimization закладки Bounds.
8. Добавьте переменные регулятора в список Design Variable Set и задайте зону поиска переменных.



9. Запустите процесс оптимизации параметров модели по реакции системы.



ЗАДАНИЕ

Скорректируйте свою систему из КП по ТАУ с помощью блока, задав требования из задания и добавив к ним разброс параметров неизменяемой части в пределах ± 10% от номинала. Получите результаты моделирования синтезированной системы при ступенчатом и гармоническом сигналах на входе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие преимущества можно получить при использовании Simulink Response Optimization в разработке систем автоматического управления?
2. Как задается целевая функция в пакете Response Optimization?

3. Почему следует рассматривать поведение модели с оптимизированными параметрами при разных входных воздействиях?

17 Список литиературы

Основная литература

1. Макаров, Н.Н. Применение пакета MATHCAD в анализе и синтезе систем автоматического управления: учеб. пособие / Н.Н. Макаров, С.В. Феофилов. – Тула: Изд-во ТулГУ, 2007. –169 с. (16 экз.)

2. Макаров Н.Н. Анализ и синтез систем автоматического управления с использованием системы MATLAB / Н.Н. Макаров, С.В. Феофилов.- Тула: Изд-во ТулГУ, 2010. – 68 с. (10 экз)

Дополнительная литература

1. Очков, В.Ф. Mathcad 12 для студентов и инженеров / В.Ф.Очков .— СПб. : БХВ-Петербург, 2005 .— 464с. (11 экз)

2. Потемкин, В.Г. Вычисления в среде MATLAB / В.Г.Потемкин .— М. : Диалог-МИФИ, 2004 .— 720с. (3 экз)

3. Поршнев, С.В. MATLAB 7:основы работы и программирования : учеб.пособие для вузов / С.В.Поршнев .— М. : Бином, 2006 .— 320с. (4 экз)

4. Дьяконов, В.П. Mathcad 11/12/13 в математике : справочник / В.П.Дьяконов .— М. : Горячая линия-Телеком, 2007 .— 958с. (9 экз)