Институт транспорта и связи

Факультет наук управления, экономики и транспорта

ПРИКЛАДНЫЕ ПАКЕТЫ ДЛЯ НАУЧНЫХ ИССЛЕДОВАНИЙ

ЧАСТЬ I
Система инженерных расчетов и моделирования MATLAB

Учебное пособие и методические указания

по выполнению

практических, контрольных и зачетных
работ для студентов дневного, вечернего
и заочного отделения

Составитель: **А. В. Граковский**

Рига 2007

004.
П 759

Transporta un sakaru institūts
Институт транспорта и связи

П 759 Прикладные пакеты для научных исследований. Часть I. Система инженерных расчетов и моделирования MATLAB: Учебное пособие и методические указания / Сост. А. В. Граковский. Рига: Институт транспорта и связи, 2007. 31 с.

Учебное пособие "Прикладные пакеты для научных исследований. Часть I. Система инженерных расчетов и моделирования MATLAB" предназначено для студентов магистерской программы обучения следующих направлений:

- Экономика,
- Управление

для всех форм обучения (дневной, вечерней, заочной).

Разделы пособия соответствуют учебной программе магистерской степени обучения по вышеуказанным программам. Курс "Прикладные пакеты для научных исследований" содержит инструментарий и технологии, необходимые в дальнейшем для проведения самостоятельных научных разработок и освоения таких дисциплин, как "Методология научных исследований", "Эконометрика", "Имитационное моделирование". Курс состоит из двух частей: "Система инженерных расчетов и моделирования MATLAB" и "Статистическая обработка данных с помощью пакета STATISTICA". Настоящее учебное пособие предназначено для использования в первой части этого курса.

В каждом разделе даны основные теоретические положения, области применения, методические указания и примеры решения типовых заданий. Некоторые примеры из области матричных операций и методов оптимизации были любезно предоставлены профессором В.Еремеевым и доцентом В.Трухачевым, за что автор выражает им глубокую признательность.

Вторая часть пособия - "Зачетные задания" содержит варианты заданий на зачетную (контрольную) работу В конце пособия представлен список литературы, содержащей более глубокую теоретическую информацию о рассматриваемых в нем вопросах.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задачи научного исследования *объектов, систем или процессов*, возникающие при анализе современных проблем в области экономики и управления, также как и в других областях знаний, в большинстве случаев определяются классической триадой совершенствования знаний: *«наблюдать – понимать – управлять»*.

Как правило, *исследование* любого процесса начинается с его систематических наблюдений и *феноменологического* (качественного) описания. Параллельно этому идет процесс накопления и систематизации числовых данных, которые отражают поведение процесса в динамике и при разных внешних условиях. Накопленные данные обрабатываются различными, например, статистическими методами, для определения их устойчивости, полноты, взаимосвязанности и т.д.

Когда накоплен необходимый объем первоначальных данных, начинается этап *понимания*, т.е. непротиворечивого (математического) описания процесса на основе каких-либо теоретических положений. Это есть *задача моделирования*. В зависимости от выбранного теоретического базиса модель может быть *детерминированной* (определенной), *стохастической* (случайной) или совмещать в себе элементы того и другого. Если задача может быть представлена в таком виде, то для нее существует общепринятая типовая последовательность действий, не зависящая, строго говоря, от прикладной области знаний, в которой она должна быть решена. Эта последовательность состоит из следующих этапов:

Постановка задачи. Считается, что от правильной постановки задачи зависит 80-95% успеха в ее решении. Постановка задачи включает в себя четкое и недвусмысленное определение, что дано и с какими погрешностями, насколько эти данные представительны для исследуемого процесса; что требуется найти и с какими ограничениями (допустимый уровень ошибки, время, требуемые ресурсы и т.д.).

Математическая модель. Математическая модель – это перевод описания задачи на язык математики (формулы, выражения, уравнения, неравенства), основанный на той или иной теоретической базе. От корректного выбора и применения теории (адекватности модели) напрямую зависит качество и достоверность получаемых результатов.

Выбор (разработка) метода. Как правило, все исследовательские задачи, которые можно решить аналитически (вручную), уже были решены в период с XVII по первую половину XX века, поэтому для решения современных задач основным инструментом исследования является численный метод с большим объемом вычислений, посильных только для компьютера.

Разработка алгоритма и программы. Этот этап предусматривает разбиение решения задачи на последовательность отдельных шагов и запись их на выбранном языке программирования.

Отладка и тестирование. Данный шаг предусматривает устранение ошибок и проверку работоспособности программы для задачи (теста), близкой к решаемой, но, в отличие от последней, аналитическое (точное) решение которой нам заранее известно. Сравнение полученных результатов с истинным решением позволяет судить о качестве работы и точности нашего метода и программы.

Валидация модели. Это проверка непротиворечивости модели выполняется путем сравнения результатов расчетов с реальными данными при известных реальных внешних и внутренних условиях (параметрах модели).

После того, как мы убедились в работоспособности модели, можно переходить к ее практическому использованию для *прогноза развития ситуации и обоснованного управления*. Как правило, такой путь наиболее характерен для детерминистических задач, которые и будут главным объектом рассмотрения в первой части этого курса.

Проведение расчетов (численный эксперимент). Это проведение расчетов на множестве исходных данных, при разных условиях, т.е. по заданной программе исследования. Это помогает определить и классифицировать возможные варианты решений, параметры модели, наиболее сильно влияющие на результат (управляющие параметры), границы устойчивости и предсказуемости процесса.

Анализ результатов, прогноз, рекомендации к управлению. Этот этап и есть тот основной шаг, ради которого были выполнены все предыдущие. Анализ текущего состояния, его классификация и прогноз поведения объекта (системы, процесса) в будущем – вот основная цель практически любой исследовательской задачи.

К сожалению, выполнение такой последовательности действий требует навыков программирования, временных затрат (от 1 недели до 6 месяцев) и, возможно, привлечения к задаче квалифицированных программистов (финансовые затраты). И все это еще не гарантирует положительного результата – здесь многое зависит от правильности построения математической модели. Требуется более простой и гибкий инструмент исследования, позволяющий быстро реализовать любую модель и провести численный эксперимент при минимуме временных и финансовых затрат – так называемый инструмент проверки идей, гипотез и моделей или «научный калькулятор». На роль такого инструмента, достаточно простого в освоении и в управлении, обладающего гигантской базой готовых программ-функций и их теоретического описания из различных прикладных областей науки, может претендовать система прикладных инженерных и научных расчетов и моделирования MATLAB.

1. КРАТКАЯ ХАРАКТЕРИСТИКА ПАКЕТА MATLAB

Первая версия пакета MATLAB 1.0 была представлена научной общественности в начале 80-х годов (разработчик Cleve Moler, компания The MathWorks Inc., Natick, Massachusetts, USA). Это была DOS-ориентированная версия, она написана на языке программирования FORTRAN и содержала богатейшую коллекцию разработанных ранее пакетов прикладных программ (ППП). В первую очередь она предназначалась для выполнения операций матричной алгебры (MATrix LABoratory). Современные версии – это С-программы, работающие в среде Windows (версии 4.0 и 4.1 - для Win 3.11, версии 5.0, 5.2, 5.3, 6.0 и выше - для Win 95, 98, NT, 2000, ME, XP). Кроме того, есть версии, разработанные для других операционных систем - UNIX, VAX, LINUX и др. В целом, современная система MATLAB – это многократно улучшенный вариант первых версий, пополненный множеством дополнительных специальных методов, функций и возможностей, включая графический интерфейс и систему моделирования.

Система MATLAB широко используется в более чем 70 лучших университетов и институтов мира, включая Стэнфорд, Кембридж, Массачусетский технологический институт, Калифорнийский университет, МГУ, МИФИ, МФТУ им. Баумана, Киото (Япония), Эйндховен (Голландия), Хельсинки, NASA, национальные лаборатории Argonna, Los-Alamos и Livermore, корпорации AeroSpace, Boeng, General Dynamics, IBM, Lockheed, Siemens AG и многие другие.

Система MATLAB дает возможность очень быстро создать программу расчетов, проверить и отладить ее, используя встроенный редактор-отладчик, произвести необходимые вычисления, представить результаты в числовой и (или) графической форме, сохранить текст программы, данные и результаты расчетов на диске. Пакет инсталлируется на жесткий диск компьютера и подключается к операционной системе, требуемое дисковое пространство зависит от выбранной пользователем конфигурации пакета и не превышает для версии MATLAB 7.0 объема 1.7 - 4 ГБ. Минимальные требования к персональному компьютеру для устойчивой и надежной работы– не менее 128 МБ RAM, операционная система – не ниже Windows 2000.

2. СТРУКТУРА ПАКЕТА MATLAB

Современный пакет MATLAB – это интерактивная система (возможны ее пополнение изменение и улучшение самим пользователем), ориентированная на работу с массивами данных. Она использует математический сопроцессор и также может работать с FORTRAN и C++ программами, обеспечивая международный IEEE стандарт арифметики. Он содержит также мощную систему информационной поддержки, включая MATLAB-Internet-конференции, сетевой поиск и банки данных, графический интерфейс пользователя (GUI) для проектирования диалоговых режимов работы программ. Все операторы и команды системы MATLAB имеют простейшую BASIC-образную форму.

Структурно пакет MATLAB состоит из двух частей – постоянной (базовые продукты) и пополняемой (структурированные по областям знаний пакеты прикладных программ – ToolBoxes (наборы инструментов)). Основная часть содержит программы и файлы, обеспечивающие функционирование собственно самой системы MATLAB, а пополняемая часть (ToolBoxes) содержит готовые наборы функций и прикладных программ для решения различных прикладных задач. Аналогично структурирована и система помощи и подсказки HELP, включающая в себя постоянную часть (вместе с Internet Help Desk) и описание каждого комплекта инструментов в виде .PDF (Acrobat Reader) и .HTML (Internet Explorer) файлов, где содержится полное описание теоретических основ методов, правил применения предлагаемых программ и примеров.

2.1. Постоянная часть MATLAB

Базовые продукты MATLAB	Описание
MATLAB-base	Система управление пакетом
MATLAB Editor-Debuger	Редактор-отладчик для M-программ
Math Library C, C++	Библиотеки математических функций
MATLAB Help	Система помощи и подсказок
Compiler C	Компилятор языка C
SimuLink	Система визуального и имитационного Моделирования
Excel Link	Инструмент связи с MS Excel (от в. 5.0)
GUI (Graphic User Interface)	Система создания экранных форм

2.2. Переменная (пополняемая) часть MATLAB

Инструменты MATLAB (Toolbox)	Описание
NAG Foundation Toolbox	Элементарные и специальные функции
Spline Toolbox	Интерполяция и аппроксимация
Optimization Toolbox	Методы оптимизации
Symbolic Math Toolbox	Методы символьной математики
Financial Toolbox	Финансовый
Statistics Toolbox	Статистика
Control Systems Toolbox	Системы автоматического управления (САУ)
System Identification Toolbox	Идентификация систем
Partial Differential Equations Toolbox	Дифф. уравнения в частных производных
Real Time Toolbox	Системы реального времени
Data Signal Processing Toolbox	Обработка данных
Signal Processing Toolbox	Обработка сигналов
Image Processing Toolbox	Обработка изображений

High Order Spectral Analysis Toolbox	Спектральный анализ высоких порядков
Communication Toolbox	Системы связи
Fuzzy Logics Toolbox	Нечеткие множества
Newral Networks Toolbox	Нейронные сети
Wavelet Toolbox	Волновая импульсная декомпозиция
Power Systems Toolbox	Энергетические системы
.....	.....
Mapping Toolbox	Дискретные отображения

2.3. Рабочие окна MATLAB и варианты работы в системе

Система запускается тремя способами:

- через иконку “Matlab” на рабочем столе DeskTop;
- с помощью последовательности действий Start -> Programs -> Matlab;
- непосредственным стартом программы управления системой matlab.exe (обычно располагается в каталоге C:\MATLAB\bin).

Запуск программы отображается на экране компьютера командным окном программы Matlab Command Window (рис.1). Главное командное окно сделано так, чтобы максимально повторять работу компьютера в режиме DOS. Возврат к предыдущей (последующей) команде – стрелка вверх (вниз), выполнение команды – *Enter*, помощь – *help ? (help <имя>)*. Если после команды стоит символ «;», то результаты выполнения команды на экран не выводятся. Регистр символов роли не играет.

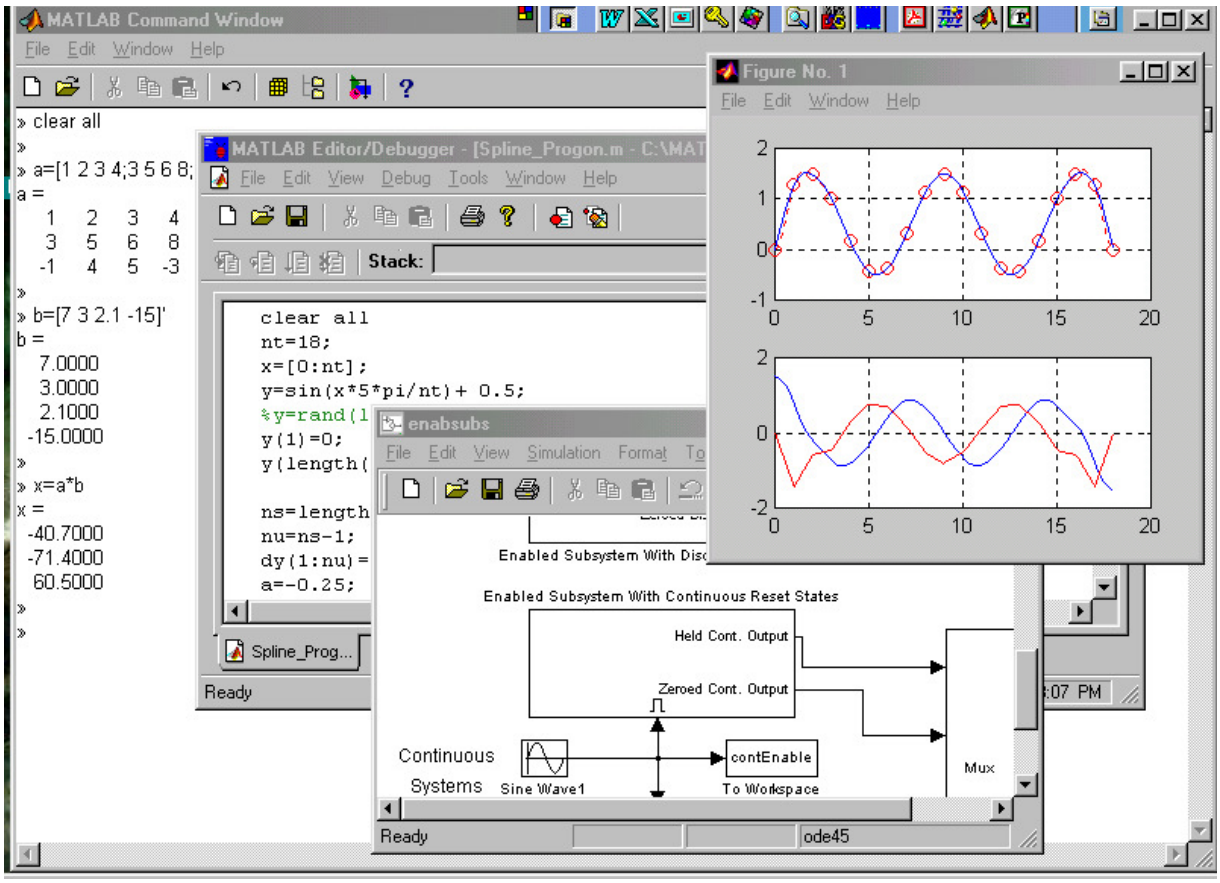


Рис.1. Рабочие окна системы MATLAB

В пакете MATLAB имеется возможность работать в двух основных режимах – *интерпретации* - когда все необходимые команды вводятся непосредственно из командной строки, и *компиляции* – когда все элементы программы записываются

сначала в М-файлы (работа в окне редактора-отладчика MATLAB Editor-Debugger), а потом вся программа целиком выполняется по команде *Run*.

Кроме того, возможна работа в окнах системы моделирования Simulink и графических окнах, где обычно помещаются графики и диаграммы (см. рис. 1). Все они являются независимыми окнами Windows, и правила переключения, минимизации, копирования содержимого окон и т.д. такие же, как обычно.

2.4. Файловая система MATLAB

При работе в пакете MATLAB главным образом приходится иметь дело с М-файлами. Это обычные текстовые файлы, имеющие расширение **.m*, в них содержатся команды, составленные из операторов и функций языка Basic Matlab. М-файлы бывают двух типов - исполняемые Script-файлы, содержащие последовательность команд, которые можно было бы вводить и последовательно в командной строке. Это аналогия головных программ. Другой вид М-файлов – файлы-функции, которые сами не исполняются, а могут быть только вызваны из командной строки, Script-файла или другого файла-функции. Файл-функция отличается от Script-файла тем, что обязательно должен начинаться с ключевого слова *function,* например, файл *my_prog.m*:

```
function y = my_prog(x)
```

и имя этого файла желательно задавать совпадающим с названием функции. Все стандартные прикладные программы, содержащиеся в инструментах системы (ядро и Toolboxes) – есть М-файлы, но пользователь может также создать свои. Если созданные пользователем М-файлы сохраняются в других папках и каталогах, не входящих в стандартный набор системы, то MATLAB «не видит» их и, соответственно, не может их использовать. Для обеспечения их видимости нужно указать пути к этим папкам («прописать») в редакторе путей Path Browser (File -> Set Path) главного окна MATLAB (рис.2).

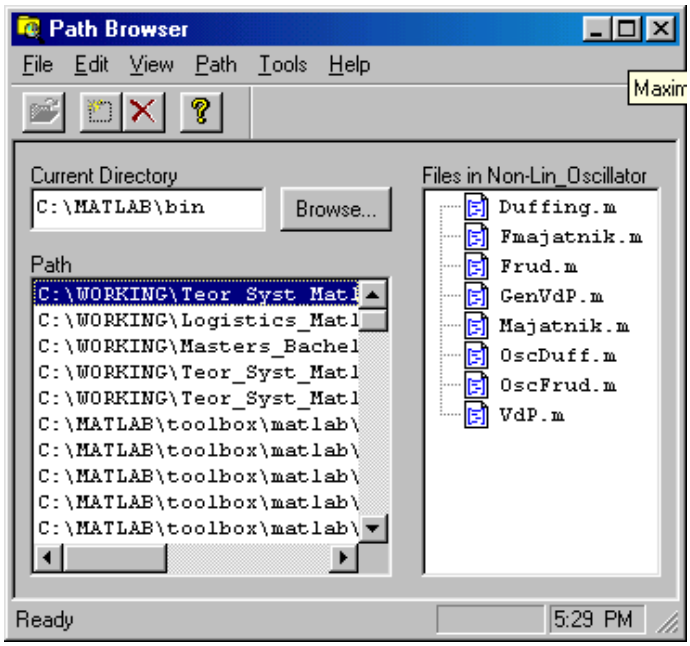


Рис.2. Окно редактора путей MATLAB (Path Browser)

После добавления новых путей информация о них сохранится в списке путей (файл C:\MATLAB\toolbox\local\pathdef.m).

Кроме вышеперечисленных, при работе с пакетом MATLAB используются файлы следующих видов:

– **.mat* – двоичный файл рабочей среды, где содержатся все переменные и массивы, использованные во время текущего сеанса работы с пакетом;

– **.mdl* – файлы моделей, созданные в среде SimuLink;
– **.mex* – объектные файлы, преобразованные из программ, написанных на языках C, C++ и Fortran.

2.5. Демонстрационный пакет MATLAB Demo

Рекомендуется начинать знакомство с пакетом MATLAB с его демо-версии. Это во-первых, поможет определить круг и форму представления задач, которые могут быть решены или уже решены (и оформлены в виде готовых к использованию прикладных программ) средствами пакета, а во-вторых, даст возможность ознакомиться с основными командами и элементами программирования в среде MATLAB. Для запуска демонстрационной программы надо ввести в командной строке главного окна системы команду *demo* (Рис. 3).

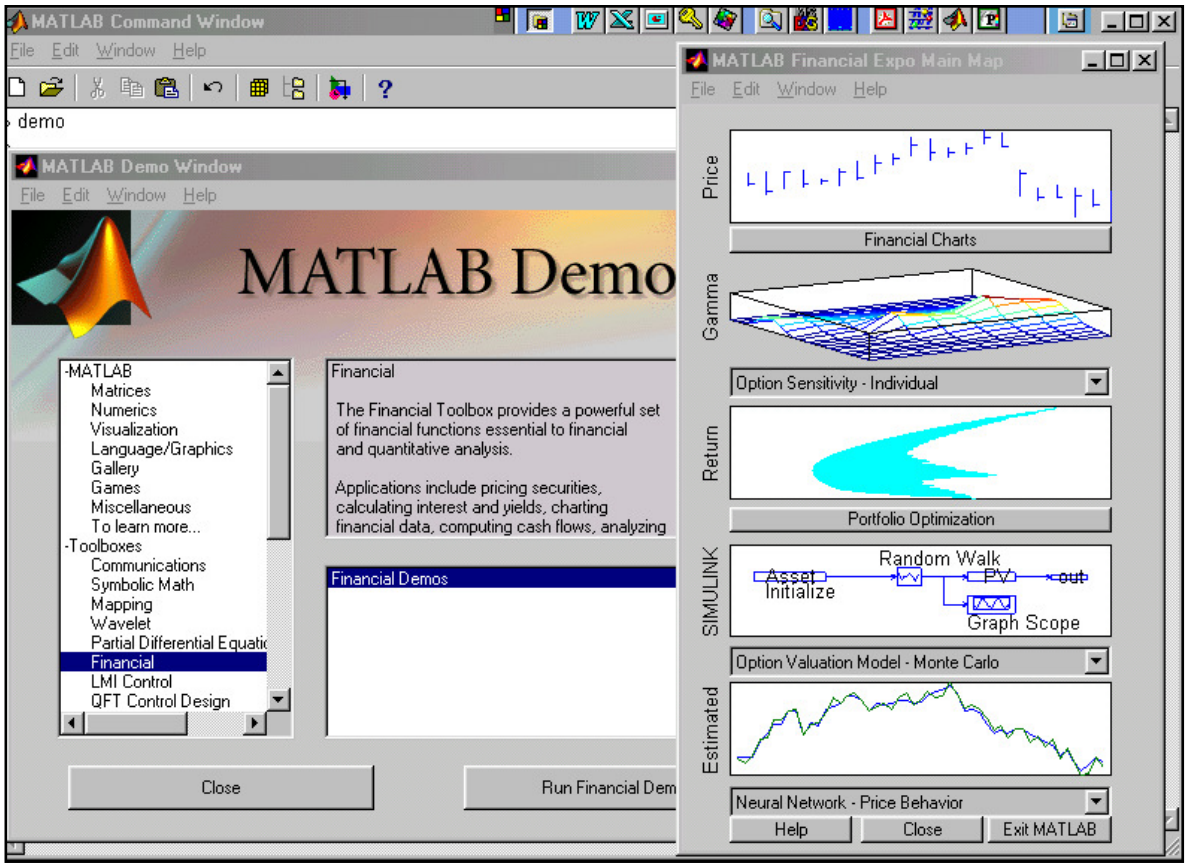


Рис.3. Окно демонстрационной версии MATLAB и пример запуска демо-версии “Финансы”

Выбор соответствующей группы демо-программ и запуск одной конкретной программы из окна MATLAB Demo, отражает не только результаты работы прикладной программы, но и ее части программного кода, что позволяет достаточно быстро и легко освоить принципы и приемы работы в системе.

Следующим шагом в изучении системы MATLAB должно стать выполнение практических заданий, предложенных далее, связанных с наиболее общими видами задач, встречающихся в исследовательской практике. Каждое задание включает в себя сначала выполнение типовой задачи до получения заданных результатов, а затем, самостоятельное решение аналогичных задач по заданию преподавателя.

ПРИМЕЧАНИЕ. Система MATLAB изначально предназначена для работы с массивами и матрицами, поэтому результаты действий * - умножить, / - делить, ^ - возвести в степень и некоторых других, будут существенно отличаться друг от друга, либо вообще выполняться с ошибкой, в зависимости от того, чем являются объекты этих действий – переменными (отдельными числами) или массивами (матрицами).

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

«ВВЕДЕНИЕ В MATLAB. МАССИВЫ, МАТРИЦЫ И ОПЕРАЦИИ НАД НИМИ»

Поскольку MATLAB изначально создавался как система, облегчающая работу с матрицами и системами линейных уравнений, то и его ядро составляют операции с матрицами и массивами (одномерными или двумерными). Причем, определение того, является ли **A** именем одиночной переменной, одномерного или двумерного массива данных, зависит от ее первого объявления и использования (инициации). Например:

Примеры команд и функций, выполняемых в командной строке главного окна MATLAB	**Комментарии, пояснения, расшифровка команд**

Упражнение 1

```
A = 53.458 , a =144          %  Здесь A – обычная переменная, где символ
                             %  десятичного разделения – точка (.) или другой
ck =23/79                    %  пример, где a –целое (верхний и нижний
Dfm = a ^ 3 + sin(A)         %  регистры означают разные имена !),ck – простая
                             %  дробь (число), а Dfm – результат вычисления.
                             %  Символ (^) – возведение в степень a^3 = a*a*a.
                             %  Количество пробелов между отдельными
                             %  символами внутри выражения несущественно.
                             %  Проценты (%) – символ комментария

A = [ 1 2 3; 4 5 6; 7 8 10 ] %  Сейчас A - это уже двумерный массив A (3,3)
                             %  размерности 3, доступ к его элементам A(2, 3)=8
                             %  символ (;) здесь означает конец текущей строки.

B =  inv(A)                  %  Инверсия – вычисление обратной матрицы A^-1
C =  B * A                   %  Проверка правильности ( A*A^-1 = E )
     A * B – C               %  Еще одна проверка

clc, home                    %  Очистка экрана. Курсор – в левый верхний угол
A,   D = A( 1 : 2 , : )      %  Вырезка первых строк матрицы A
      inv( D ),              %  Не получается !
 R =  pinv( D ),             %  Псевдоинверсия - для неквадратных
                             %  матриц
      D*R                    %  Проверка правильности
      R*D                    %  расположения матриц при
                             %  псевдоинверсии
                             %  Можно ли в матрице A заменить 10 на 9
                             %  и искать далее инверсную ?
```

Упражнение 2

```
clc, home
log ( sqr ( atan2(3,4) ) )   %  Не получается !
                             %  Стрелкой вызовите вновь эту
                             %  строчку и исправьте sqr на sqrt
log ( sqrt ( atan2(3,4) ) )  %  Через File -> PREFERENCES получите ответ
                             %  с большей точностью, набрав команду
   ans                       %  Вывод результата на экран в неявной форме
                             %  через рабочую переменную ans .
```

Упражнение 3

```
home,  A,  A',  A(:),                  % Транспонирование и превращение в столбец
home,  A,  A( 1 : 2 , : ),  A( 1 : 2 , 2 : 3)      % Вырезки из матрицы
home,  A,  A( : , 1),   home,  A,  A( 2 , :)       % Выделение столбцов и строк
home,  A,    C = fliplr(A),   F = flipud(A)        % Повороты матрицы
             G = A + j * A'                        % Комплексная матрица
home,  G ,    G1 = G',          G2 = G .'          % Запомните разницу
home,  H = [ 1  1  1 ] ; A = [ H ; A ]             % Добавление строки
home,  T = [ sin (imag(log(-1)))   sin (4*atan(1))      % Новый способ расчета
             cos (imag(log(-1)))   cos (4*atan(1)) ]    % и введения матриц.
                                                        % Почему T так проста ?

who, whos                   % Отображение переменных и массивов из рабочей среды
home, clear, clear all      % Очистка экрана и рабочей среды
eps                         % Вывод на экран машинной точности
```

Упражнение 4

```
      a =  5 ,      b = 8 ,              %
      c =  a / b ,  d =  a \ b ,         % Правое и левое деление

% ……................... Форматирование вывода ……..............................
      x =  [ -4/3   1.2345e-6 ] ,
           format  compact,  x                  % Эти же установки можно
           format  short e, x                   % выставить с помощью
           format  long,  x                     % команды из главного меню:
           format  long e,  x                   % File -> Preferences
           format  +,  x
           format  rational,  x
           format  , x
home,  A  =  magic(4)                           % Магический квадрат

      S   = sum (A),          S1  =  sum(A')    % Его свойства
      S2  = sum (diag(A)),  S3  =  sum (diag(fliplr(A))),     % Его свойства

           MA  =  max (A)          % Выделение максимальных элементов
           MI  =  min (A)          % Выделение минимальных элементов
           SO  =  sort (A)         % Сортировка элементов в столбцах
           SU  =  sort (A(:)')     % Строка - сортировка всех элементов
home,    A,  SQ  =  sqrt (A)       % Извлечение корня из матрицы
```

Упражнение 5

```
home,  clear,   format compact
x=sym( 'x' ), n=sym( 'n' )          % Объявление символьных переменных
      diff(  x^n  )                 % Символьное дифференцирование
      diff( cos(x)  )               % Символьная первая производная
      diff( cos(x) , 2)             % Символьная вторая производная
      int( 1/(1+ x^2) )             % Символьное интегрирование
      int(  x^2  + 3*x  + 5 )       % Символьное интегрирование
      int(  exp(-x^2) )             % Символьное интегрирование
      int(   log(x)*sqrt(x) ,  0, 1)    % Определенный интеграл функции
                                        % в пределах от 0 до 1
```

```
A = sym('[a  b; c  d]')                 % Задание символьной матрицы
    det ( A )                           % Вычисление детерминанта
    vpa  pi  50                         % Число π с 50-знаками
    vpa  '(1+sqrt(5))/2'  30            % "Золотое" число с 30-знаками
home,  ezplot  tan(sin(x))              % Символьный график
    x=sym( 'x' )                        % Объявление символьной переменной
    f  =  1/(5+4*cos(x));               % Символьная переменная - функция
clf,   ezplot(f)                        % Символьный график функции
clf,   ezplot (diff(f,2))               % и ее второй производной,
clf,   fplot('10/(1+exp(x)) ',[0 10]),  % Построение графика функции, заданной
                                        % символьным выражением в пределах
                                        % изменения аргумента от 0 до 10
    grid on, zoom on                    % рисование сетки графика и увеличение
    factor(1111111)                     % Разложение на простые сомножители
```

Упражнение 6

```
home,  clear,   format compact      % Округление чисел и массивов, размеры
a = [2  6  7  -3  -10  36  20]/pi   % Создание массива с вещественными числами
      fix(a)                        % Усечение дробной части числа
      ceil(a)                       % Округление до большего целого
      floor(a)                      % Округление до меньшего целого
      round(a)                      % Округление до ближайшего целого
[m,n] = size(a), nt = length(a)     % Определение размеров массивов и матриц

home,  b=magic(9)                   % Условные операции, поиск, сортировка
R = ( rem( b , 3 ) = = 0)           % Элементы, делимые на 3 (mod 3) без остатка
N = find( abs( a ) >= 5)            % Поиск элементов вектора, по модулю >= 5
[s, i] = sort (b)                   % Сортировка массива b, i - номера элементов
```

Упражнение 7

```
A = [15 18; 21 24], D=[1 2;3 4]     % Матрицы, решение систем линейных уравнений
A *  D,          A / D              % Умножение и деление матриц
A .* D,          A . / D            % Поэлементное умножение и деление массивов
A=[1 2 3;4 5 6;7 8 10], B=[9; -6 ;3] % Задание системы линейных уравнений AX=B
X=A \ B,  BN =A*X                   % Решение системы линейных уравнений, проверка
delta = A*X - B                     % Вычисление невязок уравнений (тоже проверка)
[s, i] = sort (b)                   % Сортировка массива b, i - номера элементов
z = det(A), g = cond(A)             % Расчет определителя и числа обусловленности
```

Упражнение 8

```
                                    % Полиномы, решение алгебраических уравнений
ktp = [1 -1  5  0  -8  2]           % Задание полинома 5-го порядка коэффициентами
r  = roots(ktp)                     % Решение уравнения x^5 - x^4 + 5x^3 + 0 x^2 – 8 x + 2=0
Koef = poly(r)                      % Вычисление к-тов полинома по его корням
x= 0:0.05:1; P = polyval( ktp , x );% Расчет значений полинома при 0 ≤ x ≤ 1, шаг 0.05
plot( x, P, 'b.- ' ), grid on, pause % Построение графика P(x) с координатной сеткой
plot( x, P, 'b.- ', x, exp(P), 'r.- ') % Вывод двух графиков на одном поле
help plot                           % Описание всех опций графической команды PLOT
```

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2

«МЕТОДЫ ПРИБЛИЖЕНИЯ УПОРЯДОЧЕННЫХ (ТАБЛИЧНЫХ) ДАННЫХ»

Обработка упорядоченных данных, например, временных рядов, отражающих динамику экономической системы, занимает важное место в научных исследованиях. Возможны два варианта задания данных:

- непосредственный ввод с клавиатуры в массивы, например *x = [1 2.3 3.0 4.95 ...]:*
- использование функций MATLAB, таких как *sin, rand, sawtooth, square, sinc* и других.

Кроме того, возможно импортирование данных, сформированных вне MATLAB. В зависимости от формата могут быть следующие пути:

- загрузка из ASCII или MAT – файлов командой *load*;
- чтение данных с использованием низкоуровневых функций ввода-вывода, таких как *fopen, fread, fscanf, fwrite, fclose* (смотри *help < имя функции ввода-вывода >*);
- применение MEX – файлов.

Задача предварительной обработки данных чаще всего представляется в двух вариантах, называемых *методами приближения функции*: задачах *аппроксимации* и *интерполяции*, соответственно. В исследовательской практике наиболее распространен случай, когда для конечного множества значений аргумента $x_0,\ldots,x_n$ известны полученные экспериментально (как результат наблюдений или измерений) соответствующие значения функции $f(x_0),\ldots,f(x_n)$. В этом случае аналитическое выражение функции неизвестно. При необходимости найти значения функции в промежуточных точках $x \neq x_0,\ldots,x_n$ строят приближенную (аппроксимирующую) функцию $\varphi_k(x)$, расчеты по которой либо совпадают (*задача интерполяции*), либо в определенном смысле приближаются к экспериментально полученным величинам (*задача аппроксимации*).

Аппроксимация применяется также в случае, когда вид функции известен, но очень сложен. Замена исходной функции ***f(x)*** более простой приближенной $\varphi_k(x)$ позволяет упростить вычисления или сгладить графическую форму кривой ***f(x)***.

Широкое распространение получила алгебраическая *аппроксимация* (*интерполяция*), т.е. приближенная замена на заданном интервале $x \in [x_0, x_n]$ табличной функции многочленом $P_k(x)$ степени k так, чтобы в точках x_i их значения были по возможности близки (*аппроксимация*), либо полностью совпадали (*интерполяция*): $P_k(x_i) \cong f(x_i), \quad i = 0,1,2,\ldots,n$. Точки x_i называют узлами интерполяции. Можно показать, что интерполяционный многочлен *n*-го порядка ($k=n$), проходящий через $(n+1)$ узел интерполяции, единственен.

1. Аппроксимация данных полиномом n-го порядка (функция POLYFIT)

Функция *p = polyfit(t,x,n)* находит коэффициенты полинома *p(t)* степени *n*, который аппроксимирует функцию *x(t)* по критерию минимума суммы квадратов ошибок (МНК). Выходом является вектор-строка *p* длины *n+1*, содержащая коэффициенты аппроксимирующего полинома (см. *help polyfit*).

Для примера рассмотрим аппроксимацию произвольного временного ряда с равномерным шагом по времени (расстояниями между соседними точками по оси времени *t*) полиномом третьего порядка:

```
t=(-3:1:4)';
x=[0.5 3.1 2 7 6.5 11.2 16.8 18]';
p = polyfit(t,x,3)
```

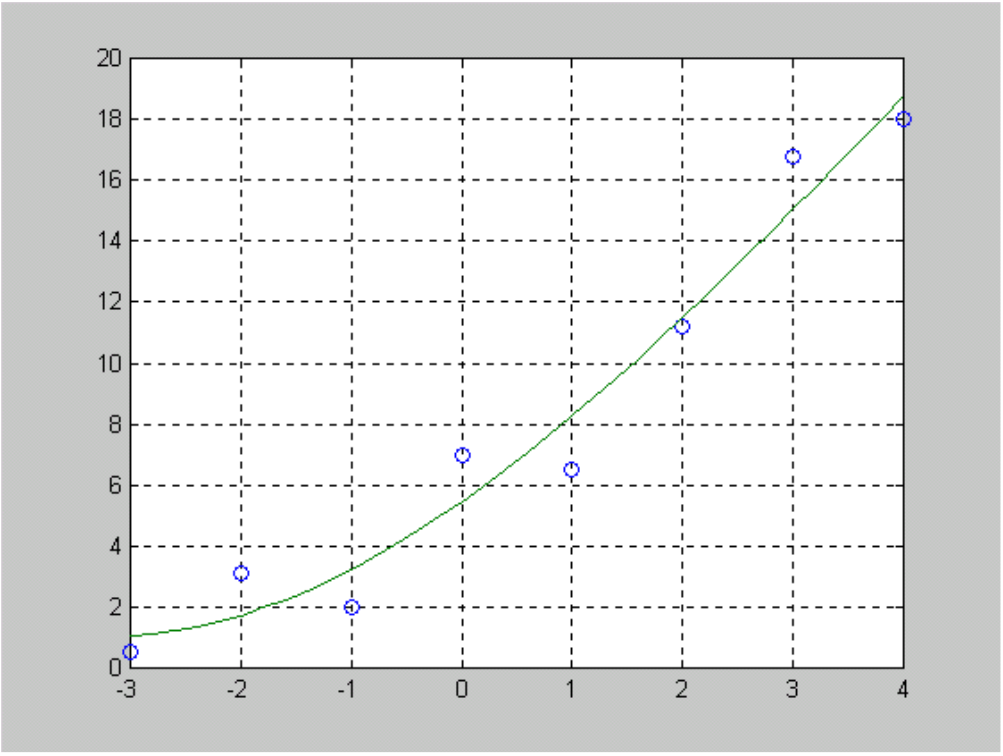
```
p =   -0.0225    0.2855    2.5357    5.4792
```

Это означает, что наши данные приближенно заменяются функцией:

$$p(t) = -0.0225 \cdot t^3 + 0.2855 \cdot t^2 + 2.5357 \cdot t + 5.4792$$

Для оценки качества приближения построим графики исходной функции и аппроксимирующего полинома на отрезке [-3 ; 4].

```
t0=(-3:0.2:4)'; x0=polyval(p,t0);
plot(t,x,'o',t0,x0),  grid on
```



Задание: Проверить, как изменится характер аппроксимирующей кривой, если порядок аппроксимации менять от *n=1* до *n=7* (при *n=1* мы наблюдаем линейную зависимость, часто называемую *линейным трендом* процесса). Выполнить аппроксимацию полиномом четвертой степени следующих данных:

1. x = (-3: 0.25 :3) ' ; y = cos(x) ;
2. x = (-2: 0.25: 3) ' ; y = 2*cos(2*x) ;
3. x= [0 0.1 0.3 0.5 1 3 7 20]; y=[0 0.05 0.2 1 5 10 20 50];
4. x=(-1:0.2:1) ' ; y = abs(x) ;

Оценить качество приближения графически, используя при выводе графиков в пять раз меньший шаг (расстояние между точками по *x*). Сделать выводы.

2. Интерполяция последовательности табличных данных произвольной формы

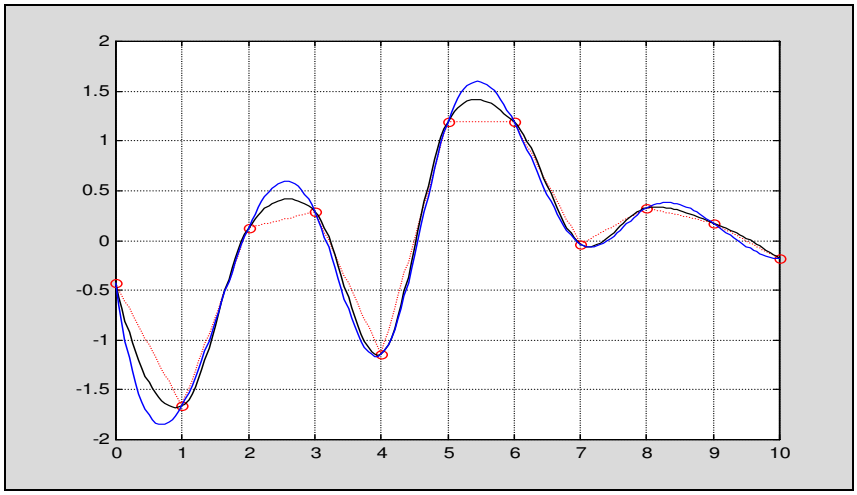
yi=interp1(x,y,xi,' <метод >'),

где соответственно *<метод> ='line',* (кусочно-линейная интерполяция),
<метод>='cubic', (локально-кубическая интерполяция),
<метод > = 'spline', (кубическая сплайн- интерполяция),

входные данные: *x,y* – исходные одномерные массивы одинаковой длины, а *xi* – новая более мелкая сетка.

Пример: интерполяция всеми тремя методами заданного случайного ряда с числом точек 10 (*n=10*). Для создания такой последовательности данных используется генератор псевдослучайных чисел с нормальным законом распределения ***randn(x)***.

```
x=0:10;  xi=0:0.05:10;
y=randn(size(x)); yl=interp1(x,y,xi,'line');
yc=interp1(x,y,xi,'cubic');
ys=interp1(x,y,xi,'spline');
plot(x,y,'or',xi,yl,'r:',xi,yc,'k',xi,ys,'b'),grid on
```



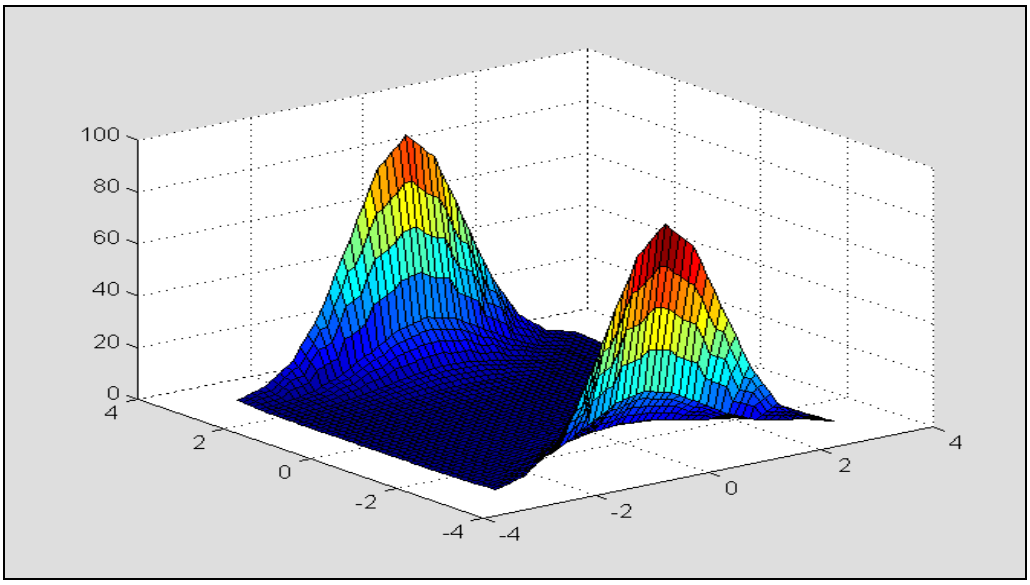
Задание: выполнить интерполяцию указанными способами отдельно, используя функции CUBIC и SPLINE, сравнить с результатами аппроксимации по методу наименьших квадратов POLYFIT порядка n=10.

3. Двумерная интерполяция на неравномерной сетке (функция GRIDDATA)

Описание: *[XI,YI, ZI]=griddata(x, y, z, xi, yi)*, где *xi, yi* – новая более плотная сетка - точечная (или дискретная) область изменения аргументов функции *x* и *y*.

Пример: Функция $z(x,y) = e^{-0.5\cdot(x^2-y^2)}$ задана на сетке [-3 ; 3] с крупным шагом **0.5**. Интерполировать эту функцию на том же интервале с шагом **0.125**.

```
[x,y]=meshgrid(-3:0.5:3);          [xi,yi]=meshgrid(-3:0.125:3);
 z=exp(-0.5*(x.^2-y.^2));          [xi,yi,zi]=griddata(x,y,z,xi,yi);
 surf(xi,yi,zi)   % Построение двумерной функции по заданным массивам xi, yi, zi
```



Задание: Попробуйте выполнить аналогичные действия для любой другой функции *F(x.y),* применяя для вывода графиков команды **plot3(…), mesh(…),** например следующей: *F(x,y) = sin(x)*exp(-x*x – y*y).*

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

«МЕТОДЫ ПОИСКА ОПТИМАЛЬНЫХ РЕШЕНИЙ»

Задача поиска наилучших управленческих, хозяйственных, технических решений (или задача *оптимизации*) может быть успешно решена, особенно, если найдено математическое описание зависимости оптимизируемой величины как функции различных параметров, влияющих на нее. Чаще всего эта задача сводится к минимизации издержек (потерь, транспортных расходов, сырья и т.д.) или к максимизации доходов (прибыли, производительности и т.д.).

И в том, и в другом случае для формализации задачи требуется знание целевой функции $F(x_1, x_2, \ldots, x_n)$, или в векторной форме ***F(X)***, зависящей от конечного числа управляющих параметров $x_1, x_2, \ldots, x_n$. Тогда задача оптимизации сводится к определению таких значений управляющих параметров, при которых $F(x_1, x_2, \ldots, x_n) = min$ (минимизация), или $-F(x_1, x_2, \ldots, x_n) = min$ (что равносильно $F(x_1, x_2, \ldots, x_n) = max$, т.е. задачи максимизации). Такие задачи решаются либо сведением ее к решению системы нелинейных уравнений $dF/dX = 0$, либо методами поиска минимума, в том числе и методами линейного (симплекс-метод) и нелинейного программирования. В некоторых случаях задача осложняется наличием ограничений, накладываемых на диапазон изменения управляющих параметров, что сужает область поиска оптимальных управляющих параметров.

1. Минимизация функций при отсутствии ограничений (FMINBND, FMINUNC)

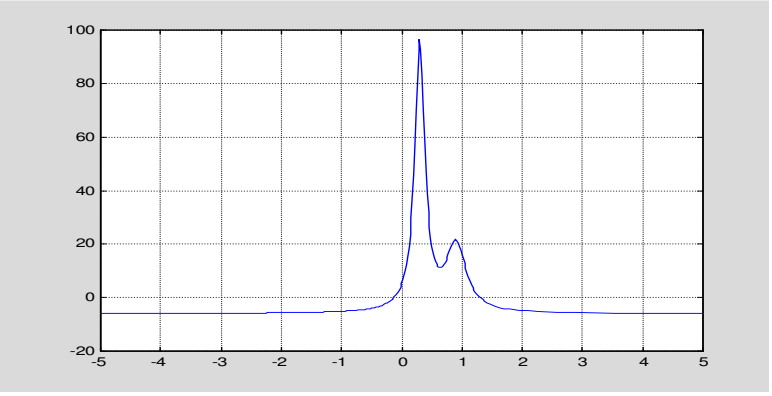
$$f(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6,$$

Как правило, математические функции в MATLAB определяются в М-файлах. Например, для функции пользователя, заданной в отдельном М-файле *targ.m (*)*

```
function  y = targ(x)
y = 1./((x - 0.3).^2 + 0.01) + 1./((x - 0.9).^2 + 0.04) - 6;
```

(*) Здесь и далее выбор имени функции произволен, но должен соответствовать названию М-файла. Можно вывести график этой функции на экран дисплея:

```
fplot('targ', [-5  5]),   grid on
```



Найдем минимум функции *targ* в интервале [0.3 ; 1] , используя функцию *fminbnd* (справочную информацию по ее использованию можно получить командой *help fminbnd*):

```
x = fminbnd('targ',0.3, 1)
```

, которая возвращает x = 0.6370 .

1. Найти координату максимума функции *targ* на интервале [0 ; 0.5].
2. Найти координату максимума функции *targ* на интервале [0.7 ; 1.5].
3. Найти координату максимума функции *(x-sinx*cosx)/(1-cosx)* на интервале [0 ; π].
4. Найти координату минимума функции (x-sinx*cosx)/(1-cosx) на интервале [π ; 2π].

Рассмотрим минимизацию функции нескольких переменных, используя функцию *fminunc* (справочную информацию по ней можно получить командой *help fminunc*). Минимизируемую функцию двух переменных занесем в М-файл, например, *fun1.m*

```
function  f = fun1(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

Из командного окна MATLAB вызовем оптимизационную программу, предварительно сформировав вектор начального приближения x_0 (точку старта поиска):

```
x0 = [-1  1];  x = fminunc( 'fun1' , x0 )
```

Решение будет найдено после 36 вычислений функции → x = 0.5000 -1.0000
Значение функции при этом равно `fun1(x)` → ans = 1.3029 e-10.

Задание на самостоятельную работу:

Минимизируйте функцию

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

Минимизируйте функцию

$$f(x_1, x_2) = x_1^4 + x_2^4 + 2 * x_1^2 * x_2^2 - 4 * x_1 + 3$$

Обратите внимание на возможность изменения параметров минимизации с помощью вектора *options*.

2. Оптимизация при наличии ограничений (FMINCON)

Ограничениями будем называть внешние условия, накладываемые на точку минимума (максимума) целевой функции ***F(X)*** дополнительно. Математически ограничения записываются в виде систем линейных или нелинейных уравнений или неравенств:

- ограничения вида линейных неравенств $\mathbf{AX \le B}$;
- ограничения вида линейных равенств $\mathbf{A_e X = B_e}$;
- область допустимых значений $\mathbf{VLB \le X \le VUB}$;
- ограничения вида нелинейных неравенств $\mathbf{G(X) \le 0}$;
- ограничения вида нелинейных равенств $\mathbf{C_e(X) = 0}$.

Оптимизацию при наличие ограничений в MATLAB 7.0 выполняет функция:

fmincon('F', x0, A, B, Ae, Be, VLB, VUB, 'nonlincon')

Здесь ***F*** – имя М-файла, содержащего целевую функцию $F(x_1, x_2, \ldots, x_n)$ или сокращенно, ***F(X)***, ***x0*** - вектор начальных условий (его длина должна быть равна числу неизвестных в целевой функции $F(x_1, x_2, \ldots, x_n)$, и, наконец, ***nonlincon*** - имя М-файла, содержащего нелинейные ограничения вида нелинейных неравенств $G(X) \le 0$ и нелинейных равенств $C_e(X) \le 0$.

Минимизируем, например, функцию *fun1* из предыдущего примера при наличии ограничений

$$x_1 * x_2 - x_1 - x_2 \le -1.5,$$
$$x_1 * x_2 \ge -10 .$$

При наличии ограничений, во-первых , их необходимо включить в М-файл, названный, например, *mycon.m,* во-вторых, представить ограничения в виде $G(X) \le 0$ и $C_e(X) \le 0$ т.е. переписать их в форме

$$x_1 * x_2 - x_1 - x_2 + 1.5 \le 0,$$
$$-x_1 * x_2 - 10 \le 0.$$

Таким образом М-файл *mycon.m* принимает вид:

```
function  [g,ce] = mycon(x)
g(1) = 1.5 + x(1)*x(2) - x(1) - x(2) ;       % Constraints
g(2) = -x(1)*x(2)-10 ;
ce=[];
```

Последняя запись означает, что нелинейные ограничения типа равенства в задаче отсутствуют ([ ] означает «пусто»). Если в задаче отсутствую какие-то виды ограничений, то на соответствующих позициях в вызваемой функции *fmincon* также ставится значок [ ] – пусто.

Вызовем оптимизационную программу:

```
x0 = [-1  1];  x = fmincon( 'fun1' , x0, [],[],[],[],[],[], 'mycon' )
```

Решением задачи будет x = -9.5474 1.0474 ;
f = 0.0236 ;
g = 1.0e-15* (-0.8882 0.0000)' .

Если аргумент *x* должен лежать в определенном интервале, т.е. *vlb* < = *x* < = *vub*, где *vlb* и *vub* – нижняя и верхняя границы, обращение к программе оптимизации имеет вид:

fmincon('fun1' , x0, [ ], [ ], [ ], [ ], vlb, vub, 'mycon') ;

Например, при $\boldsymbol{x}_1 \ge 0, \boldsymbol{x}_2 \ge 0$ по умолчанию используются команды

```
x0 = [ -1 1 ];   vlb = [ 0, 0] ;   vub = [ ] ;
x = fmincon( 'fun1' , x0, [],[],[],[], vlb, vub, 'mycon' )
```

Результатом минимизации будут x = 0 1.5000 ;
f = 8.5000;
g = 0 -10 .

Если имеются ограничения типа равенств, в систему $C_e(x) \le 0$ они включаются дополнительно и добавляются в М-файл *mycon.m*.

Например, ограничение типа $x_1 + x_2 = 1$ приведет к

```
function  [g,ce] = mycon(x)
g(1) = 1.5 + x(1)*x(2) - x(1) - x(2) ;       % Constraints
g(2) = -x(1)*x(2)-10 ;                       % Constraints
ce = x(1) + x(2) - 1;                        % Equality constraints
```

а затем, в командной строке окна **MATLAB Command Window** надо ввести

```
x0 = [-1 1];
x = fmincon( 'fun1' , x0, [],[],[],[],[],[], 'mycon' )
```

Результаты: x = -2.7016 3.7016 ; f = 1.6775 ; g = -0.0000 -9.5000 0.0000 .

Задание на самостоятельную работу:

1. Минимизируйте функции из задач 1. и 2. предыдущего раздела при
   - наличии двух вышеприведенных ограничениях типа неравенств;
   - наличии вышеприведенных равенства и двух неравенств;
   - используя оптимизационные процедуры CONSTR, LEASTSQ .
2. Проделать п.1 для алгоритма MINIMAX (*в случае успешного выполнения всей работы !!!*).

3. Определение нулей нелинейных уравнений (FZERO)

Нахождение нулей функции одной переменной

fzero('F', x0, <tol>, <trace>) , (см. help !!!).

Примеры.

```
  x = fzero( 'sin' , 3 )
```

```
x = 3.1416
```

```
x = fzero( 'cos', [1 2])
```

```
x = 1.5708
```

Найдем нуль функции $f(x) = x^3 - 2x - 5$, создав предварительно М-файл *funz.m*

```
function y = funz(x)
y = x.^3 - 2*x - 5;
```

Из окна MATLAB найдем нуль вблизи $x = 2$:

```
z = fzero('funz',2)
```

z = 2.0946.

Проверим результат, воспользовавшись функцией

```
 roots( [1 0 -2 -5] )
```

4. Решение системы нелинейных уравнений многих переменных

fsolve('F', x0, <options>,<'grad'>)

Примеры:

1. Найти решение системы уравнений $\begin{aligned} 2x_1 - x_2 - e^{-x_1} &= 0, \\ -x_1 + 2x_2 - e^{-x_2} &= 0. \end{aligned}$

Создадим М-файл *fun4.m*

```
function F = fun4(x)
F = [2*x(1) - x(2) - exp(-x(1));  -x(1) + 2*x(2) - exp(-x(2)) ];
```

Из командного окна MATLAB набираем

```
x0 = -5*ones(2,1); options = foptions; options(1) = 1; % To display output
```

```
x = fsolve('fun4', x0, options)
```

Результаты : x = [0.5671; 0.5671], (F = 1.0e-07* [0.2642; 0.2642])

2. Найти матрицу X, удовлетворяющую уравнению

$X * X * X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, начиная с точки X = [1 1; 1 1].

Создадим M –файл :

```
function F = fun5(x)
F = x*x*x - [1 2;3 4];
```

Вызовем оптимизационную программу:

```
x0 = ones(2,2);
x = fsolve( 'fun5', x0)
```

x = -0.1291 0.8602
1.2903 1.1621

Задание на самостоятельную работу:

1. Решить систему нелинейных уравнений $\begin{cases} x_1^2 + x_2 - 11 = 0; \\ x_1 + x_2^2 - 7 = 0; \end{cases}$
2. Определить нули функции *targ (targ.m)*, начиная с точек $x = 0$ и $x = 1$.

5. Линейное программирование x =linprog(F,A,B) с ограничениями вида AX ≤ B

Методы линейного программирования как часть научного направления, называемого «исследование операций», появились в конце второй мировой войны как средство анализа и планирования операций. Линейное программирование используется, когда целевая функция есть линейная сумма, составленная из произведений неизвестных параметров управления, подлежащих определению, на известные весовые коэффициенты:

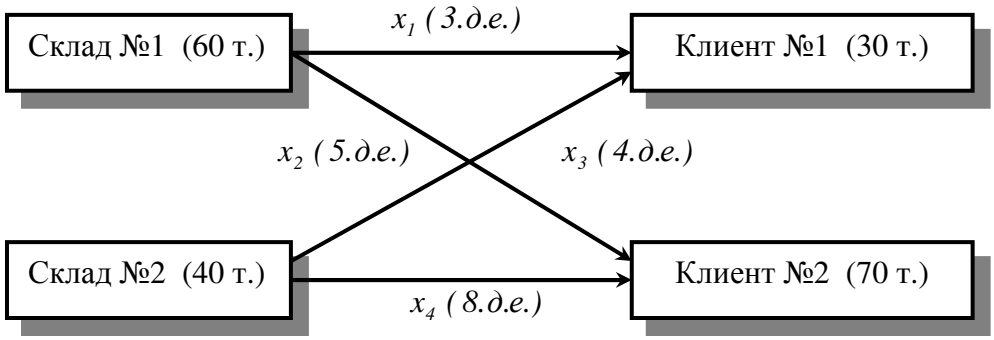
$$F(x_1, x_2, \ldots, x_n) = k_1 \cdot x_1 + k_2 \cdot x_2 + \ldots + k_n \cdot x_n,$$

а на искомые параметры $X = [x_1, x_2, \cdots, x_n]$ накладываются ограничения, которые можно записать в виде системы линейных неравенств $\boldsymbol{AX} \le \boldsymbol{B}$.

Рассмотрим в качестве примера использование метода линейного программирования для решения транспортной задачи.

Пример: Пусть на два склада поставщика поступило 100 тонн товара (60 и 40 соответственно), которые должны быть поставлены двум клиентам (30 и 70 тонн). Географическое положение складов и поставщиков таково, что стоимость доставки 1 тонны товара от каждого склада к каждому клиенту различны и составляют 3, 5, 4 и 8 денежных единиц соответственно. Требуется минимизировать транспортные расходы.

Представим для наглядности эту задачу в виде схемы:



Тогда целевая функция этой задачи (транспортные расходы), принимает вид:

$$F(x_1, x_2, x_3, x_4) = 3 \cdot x_1 + 5 \cdot x_2 + 4 \cdot x_3 + 8 \cdot x_4,$$

где x_{1-4} - часть груза, которую надо перевезти из одного склада к одному клиенту. Условие выполнения заказа на поставку и конечное число товара на складах накладывает на задачу ряд ограничений, которые можно записать в виде неравенств:

$$\begin{cases} x_1 + x_2 = 60, \\ x_3 + x_4 = 40, \\ x_1 + x_3 = 30, \\ x_2 + x_4 = 70, \\ x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0,\ x_4 \ge 0. \end{cases}$$

Приведем эти ограничения к требуемому виду **AX ≤ B**:

$$\begin{cases} x_1 + x_2 \le 60, \quad -x_1 - x_2 \le -60, \\ x_3 + x_4 \le 40, \quad -x_3 - x_4 \le -40, \\ x_1 + x_3 \le 30, \quad -x_1 - x_3 \le -30, \\ x_2 + x_4 \le 70, \quad -x_2 - x_4 \le -70, \\ -x_1 \le 0,\ -x_2 \le 0,\ -x_3 \le 0,\ -x_4 \le 0. \end{cases}$$

Для решения этой задачи задаем в командном окне MATLAB целевую функцию **F** в виде строки стоимостей, матрицу ограничений **A** и вектор правой части **B**:

```
F = [3. 5. 4. 8.]
A = [1 1 0 0 ; -1 -1 0 0 ; 0 0 1 1 ; 0 0 -1 -1 ; 1 0 1 0 ; -1 0 -1 0 ; →
    →  0 1 0 1 ; 0 -1 0 -1 ; -1 0 0 0 ; 0 -1 0 0 ; 0 0 -1 0 ; 0 0 0 -1]
B = [60 ; -60 ; 40 ; -40 ; 30 ; -30 ; 70 ; -70 ; 0 ; 0 ; 0 ; 0]
```

После этого в командной строке окна MATLAB можно ввести

```
x = linprog(F, A, B)
```

Полученный результат – и есть оптимальное распределение грузов:

```
x =
     0          (тонн)
  60.0000       (тонн)
  30.0000       (тонн)
  10.0000       (тонн)
```

Для получения величины минимально возможных транспортны расходов **TRR** перемножим **F** и **X**, что соответствует вычислению $F_{min} = 3 \cdot x_1 + 5 \cdot x_2 + 4 \cdot x_3 + 8 \cdot x_4$:

```
TRR = F * x
```

Получен ответ: TRR = 500.0000 (денежных единиц).

Заметим, что при любой другой комбинации распределения грузов транспортные расходы стали бы больше или же были бы нарушены условия поставки (ограничения).

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4

«ИССЛЕДОВАНИЕ ДИНАМИКИ ЭКОНОМИЧЕСКИХ СИСТЕМ. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ И ИХ СИСТЕМ»

Научные и практические задачи, связанные с моделированием и прогнозом поведения каких-то систем, т.е. с динамикой (изменением наблюдаемых показателей во времени), так или иначе сводятся к математическому описанию задачи в виде одного или системы нескольких обыкновенных дифференциальных уравнений (ОДУ), содержащих производную по времени (скорость изменения) от наблюдаемой величины

$$\frac{dx}{dt} = f(t,x), \quad \text{или} \quad \begin{cases} \dfrac{dx_1}{dt} = f_1(t,x_1,x_2), \\ \dfrac{dx_2}{dt} = f_3(t,x_1,x_2) \end{cases} \Rightarrow \dot{X} = F(t,X) \text{ , или } \begin{cases} \dfrac{dx}{dt} = f_1(t,x,y,z), \\ \dfrac{dy}{dt} = f_3(t,x,y,z), \\ \dfrac{dz}{dt} = f_3(t,x,y,z), \end{cases}$$

с заранее известными, начальными условиями *x(0), y(0), z(0)*, необходимыми для нахождения единственного решения *x(t), y(t), z(t)*, как функций от времени (задача Коши). Часто для упрощения записи производную по времени заменяют условным обозначением – точкой над переменной или штрихом : $dx/dt = \dot{x} = x'$, а уравнения старших порядков могут принимать вид: $x''' = f(t,x,x',x'')$. Тогда это уравнение нужно заменой переменных свести к системе ОДУ первого порядка $\dot{X} = F(t,X)$.

В системе MATLAB такие задачи можно решить с помощью встроенных функций *ODE23* и *ODE45*, для которых надо лишь правильно задать входные параметры и описать в отдельном М-файле функцию правой части уравнения : $f(t,X)$:

[t,x]=ODE23('<имя функции>', tspace, x0, tol, trace)
[t,x]=ODE45('<имя функции>', tspace, x0, tol, trace)

Функция *ODE23 (ODE45)* решает систему обыкновенных дифференциальных уравнений методом Рунге-Кутта 2-3 (4-5) порядка соответственно. Это методы адаптивного интегрирования с автоматическим выбором шага, дающие численное решение с заданной точностью 10^{-3} (10^{-6} для *ODE45*). Поскольку шаг по времени непостоянен, то дискретное время выводится в отдельный выходной массив *t*, содержащий столько же элементов, сколько массив решений *x(t)*.

tspace - временной интервал определения $[t_0 \quad t_{final}]$,

x0 - вектор-столбец начальных условий $[x_1(0), x_2(0), \ldots, x_n(0)]$,

tol - заданная точность решения задачи (по умолчанию *tol=1. e-3* для *ODE23* и *tol=1. e-6* для *ODE45*) – необязательный параметр,

trace – флаг выдачи промежуточных результатов (по умолчанию *trace=0*),

<имя функции> - имя М-файла, содержащего правые части системы уравнений, приведенной к стандартной форме:

$$\begin{cases} \dot{x}_1(t) = f_1(t, x_1(t), x_2(t), \ldots, x_n(t)), \\ \dot{x}_2(t) = f_2(t, x_1(t), x_2(t), \ldots, x_n(t)), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ \dot{x}_n(t) = f_n(t, x_1(t), x_2(t), \ldots, x_n(t)). \end{cases}$$

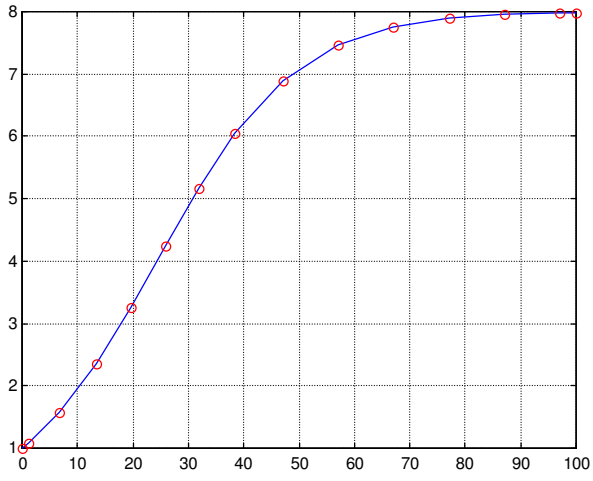
Пример 1. Рассмотрим в качестве примера динамику процесса реализации нового товара на ограниченном рынке, которая характеризуется двумя встречными эффектами: экспоненциальным ростом спроса на новый товар (вследствие рекламы и первичного ажиотажа) с коэффициентом роста a и снижением спроса, обусловленным ограниченной покупательной способностью населения и насыщением рынка с коэффициентом насыщения b. Такой процесс, как и многие, ему подобные, описывается нелинейным ОДУ первого порядка, называемым еще "логистическим" **[5]** уравнением: $\dot{x} = ax - bx^2$. Пусть начальная партия, выброшенная на рынок, составила $x(0) = 1$ тысяча единиц товара, а коэффициенты роста и насыщения рынка составляют $0{,}08$ и $0{,}01$ соответственно.

Для решения этой задачи сначала создадим М-файл с именем *logist.m:*

```
function y = logist(t,x)
% Function for calculation of the "logistic" equation right side
y=0.08*x-0.01*(x.^2);
```

Независимый параметр t время явно не входит в правую часть уравнения, а точка перед возведением в квадрат означает, что действие производится поэлементно. Затем уже в командной строке окна MATLAB вводим:

```
x0=1.;
t0f=[0 50]
[t,x]=ode23('logist',t0f,x0)
plot(t,x,t,x,'ro'),grid on,zoom on
```



Результаты вычисления выводим в графическое окно с помощью функции *PLOT*.

Как мы наблюдаем из графика, объемы реализации этого товара сначала стремительно нарастают, а потом темпы продаж замедляются и через время порядка 100 единиц безразмерного времени реализация выходит на стабильный уровень ≈ 8 тысяч единиц товара, соответствующий теоретическому пределу $x(\infty) = a/b$.

Пример 2. Решить линейное ОДУ второго порядка: $y'' + 5y' + 4y = 2$ с начальными условиями $y(0)=1,\ y'(0)=2$. Такое уравнение может служить простейшей моделью переходного процесса линейной динамической системы из одного состояния в другое. Здесь коэффициент 4 может интерпретироваться как частота (время) обращения, а 5 – как параметр противодействия переходу в новое устойчивое состояние. Для сравнения, нам известно точное аналитическое решение - $y(t)=0.5 + (4/3)e^{-t} - (5/6)e^{-4t}$.

Преобразуем исходное уравнение 2-го порядка в систему двух ОДУ 1-го порядка:

$$y' = z, \qquad z' = 2 - 5z - 4y.$$

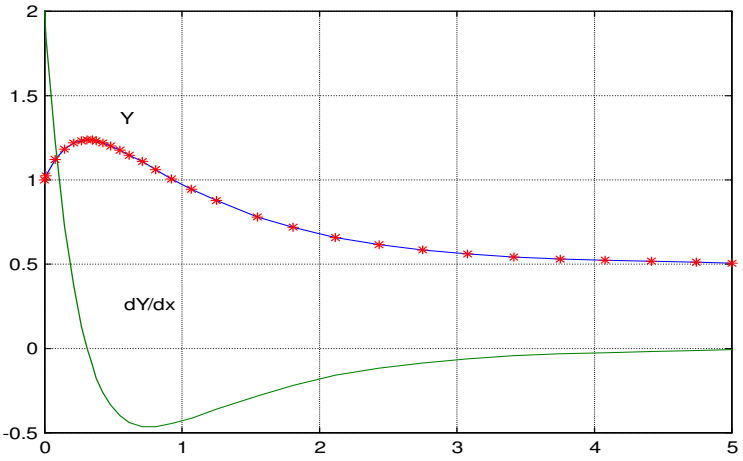
Создадим М-файл с именем *pdu2.m* для записи в него правых частей теперь уже системы двух дифференциальных уравнений с обозначениями $y \to x_1,\ \ z \to x_2$:

```
function y = pdu2(t,x)
% Program for calculation of the ODE right side
y=[x(2);2-5.*x(2)-4.*x(1)];
```

Решаем дифференциальное уравнение в командной строке окна MATLAB, задавая интервал наблюдения решения $0 < t < 5$ и вектор начальных условий $x_0 = [1 \quad 2]$:

```
[t,x]=ode23('pdu2',[0 5],[1 2]);
plot(t,x,t,0.5+4/3.*exp(-1.*t)-5/6.*exp(-4.*t),'r*'), grid, hold on
gtext('Y'),gtext('dY/dt')
```

Здесь мы выводим на график с помощью функции *PLOT* не только найденные решения *y(t)* и *z(t)* – на самом деле это колонки *x(1)* и *x(2)* двумерного массива *X*, но и отмеченное красными звездочками (`*r*`) аналитическое точное решение задачи. Оператор *HOLD ON* (держать поле графика открытым) требуется здесь для того, чтобы на этом же графике сделать пояснительные пометки `*Y*` и `*dY/dt*` с помощью команды *GTEXT*.



Пример 3. Прогноз результатов пассивной конкуренции двух фирм, действующих в одном секторе рынка. Математическая модель, отражающая такую конкуренцию, описывается системой нелинейных дифференциальных уравнений (другое название – “модель гонки вооружений” **[4]**):

$$\begin{cases} \dot{x}_1 = a_1 x_1 - a_{11} x_1^2 - a_{12} x_1 \cdot x_2, \\ \dot{x}_2 = a_2 x_2 - a_{22} x_2^2 - a_{21} x_1 \cdot x_2, \end{cases}$$

где – a_1, a_{11}, a_2, a_{22} коэффициенты «рождаемости» и «смертности», учитывающие собственные возможности конкурентов, а a_{12}, a_{21} – коэффициенты перекрестного влияния (*синергетики*) их друг на друга. От соотношения этих коэффициентов и зависит исход конкурентной борьбы. Зададим начальные условия $x_1(0) = 0.1$ и $x_2(0) = 0.8$ - преимущество второго конкурента на старте очевидно, а интервал времени наблюдения – *[0 ... 100]*. Пусть коэффициенты уравнения равны $a_{12} = 0.11,\ a_{21} = 0.25$, $a_1 = 0.3,\ a_{11} = 0.15, a_2 = 0.45, a_{22} = 0.22$.

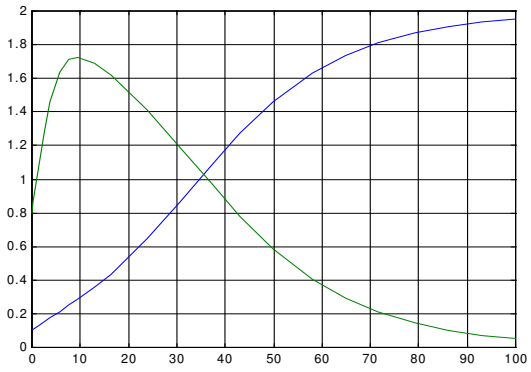
Создадим М-файл *Konkur.m* для записи в него правых частей системы двух дифференциальных уравнений с обозначениями $x = [x_1, x_2]$:

```
function y = Konkur(t,x)
% Program for calculation of the ODE right side
y=[0.3*x(1)-0.15*(x(1).^2)-0.11*(x(1).*x(2));
   0.45*x(2)-0.22*(x(2).^2)-0.25*(x(1).*x(2))];
```

Решаем дифференциальное уравнение в командной строке окна MATLAB, задавая интервал наблюдения решения $0 < t < 100$:

```
[t,x]=ode23('Konkur',[0 100],[0.1 0.8]);
 plot(t,x), grid on, zoom on          % Вывод x1(t),x2(t) на график
```

Анализ полученных кривых показывает, что, несмотря на более выгодные стартовые условия и большие производственные мощности ($a_1 < a_2$), вторая фирма проигрывает в конкурентной борьбе из-за более высоких потерь и коэффициента взаимного подавления. Следует отметить, что изменение коэффициентов уравнения может привести к прямо противоположным результатам. Определение таких коэффициентов по реальным данным может позволить осуществлять



постоянный мониторинг ситуации и контроль правильности собственных управленческих решений.

Пример 4. Анализ динамики производства товаров в модели бартерного обмена товарами x_1 и x_2 между производителями **[6]**:

$$\begin{cases} \dot{x}_1 = k_1 - j_1 \cdot x_1 \cdot x_2 - d_1 \cdot x_1, \\ \dot{x}_1 = k_2 - j_2 \cdot x_1 \cdot x_2 - d_2 \cdot x_2, \end{cases}$$

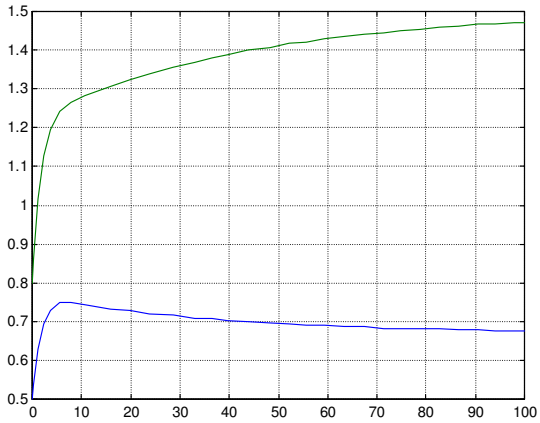
где k_1, k_2 - производительности 1-го и 2-го товаров в единицу времени, d_1, d_2 - долговечности (качество товаров), а j_1, j_2 - так называемая обменная эффективность обоих товаров, отражающая потребительский спрос на них ($c = j_1 / j_2$ - меновая стоимость обмена). Пусть эти коэффициенты составляют: $k_1 = 0.3, k_2 = 0.45, d_1 = 0.15,$ $d_2 = 0.15, j_1 = 0.2, j_2 = 0.35, x_1(0) = 0.5, x_2(0) = 0.8$.Для прогноза развития производства обоих товаров со временем создадим М-файл с *barter.m* для записи в него правых частей системы двух дифференциальных уравнений с обозначениями $x = [x_1, x_2]$:

```
function y = barter(t,x)
% Program for calculation of the ODE right side
y=[0.3-0.15*x(1)-0.2*(x(1).*x(2)); 0.45-0.15*x(1)-0.35*(x(1).*x(2))];
```

Решаем дифференциальное уравнение в командной строке окна MATLAB, задавая интервал наблюдения решения $0 < t < 100$:

```
[t,x]=ode45('barter',[0 100],[0.5 0.8]);
 plot(t,x), grid on,            % Вывод x1(t),x2(t)
```

Как видно из приведенных графиков, при заданных параметрах модели такая система имеет тенденцию к стабилизации объемов производства обоих товаров на уровнях, соответствующих их обменной стоимости. И, также как и в предыдущем случае, определение коэффициентов такой модели по реальным данным – временным рядам может позволить скорректировать производственную и бартерную политику предприятия для стабилизации производства.



Пример 5. Решение системы ОДУ третьего порядка (конвективная модель Лоренца):

$$x' = P(y - x),$$
$$y' = -xz + R\,x - y,$$
$$z' = xy - B\,z,$$

с начальными условиями $x(0) = -0.7,\ y(0) = -0.7,\ z(0) = 0.8$ на интервале времени $t = [0, 50]$, с параметрами модели $P=10, B=8/3, R=30.$

Такая модель, кроме своих физического и технического приложений (образование грозовых облаков, турбулентность и т.д.), нашла и свое применение в экономике, как модель развития мелкого и среднего бизнеса в рамках крупных городских агломераций **[7]**. Её характерная особенность – генерация решений в виде псевдослучайных временных рядов. Т.е. ее решения – хаотические колебания, внешне случайные, но

подчиняющиеся строго заданному закону управления. Этот закон, или упорядоченность, не видимая во временной области, становится заметной в трехмерном пространстве, называемом *фазовым пространством* системы.

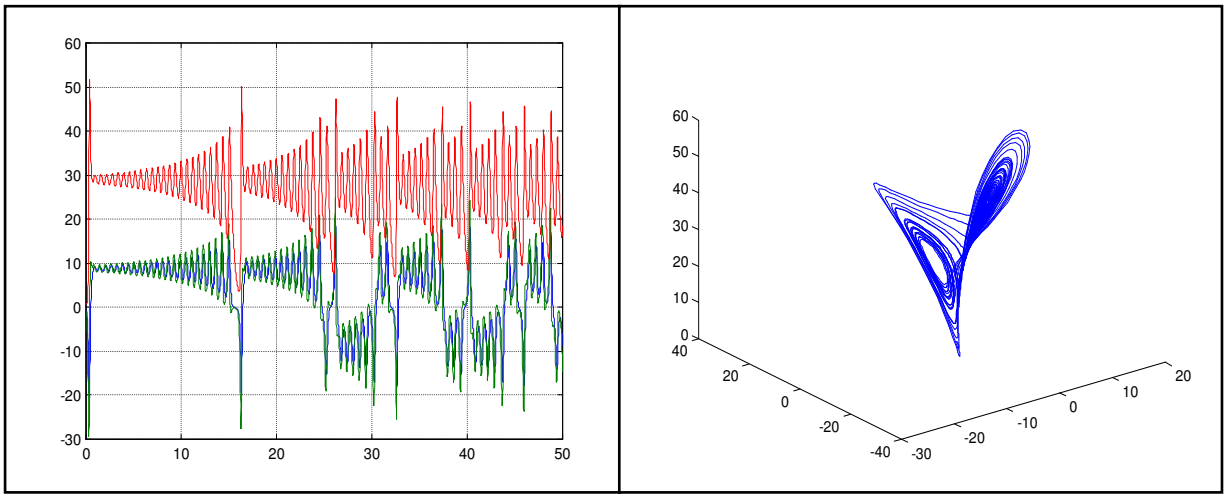
М-файл для решения этой задачи *lorenc.m* принимает вид:

```
function y = lorenc(t,x)
y=[10.*(x(2)-x(1));-x(1).*x(3)+30.*x(1)-x(2);x(1).*x(2)-8/3.*x(3)];
```

и тогда в командной строке окна MATLAB Command Window:

```
x0=[-0.7 -0.7 0.8];ts=[0 50];[t,x]=ode45('lorenc',ts,x0);
figure (1), plot(t,x),grid on                % Зависимость от времени
figure (2), comet3(x(:,1),x(:,2),x(:,3))     % Фазовая траектория
```

В этой программе мы используем новую графическую функцию *COMET3* для построения непрерывной кривой в трехмерном пространстве и демонстрации движения точки по фазовой траектории. Выражение *x(:,1)* означает, что берется первый столбец массива решений *X*.



Другой особенностью данного примера является использование функции *FIGURE(N)*, применяющейся для вывода графических изображений в разные графические окна с названиями *Figure 1, 2, ..., N* соответственно.

ЛИТЕРАТУРА

1. Anderson, Patrick L., Business economics and finance with MATLAB, GIS and simulation models. Chapman & Hall/CRC Press LLC, 2005.
2. В.Г.Потемкин. MATLAB. Справочное пособие. -М.: ДИАЛОГ МИФИ, 1997.
3. Дьяконов В.П. Справочник по применению системы PC MATLAB. М.: Наука, 1993.
4. С.П.Капица, С.П.Курдюмов, Г.Г.Малинецкий. Синергетика и прогнозы будущего. 2-е изд.- М: Эдиториал УРСС, 2001.
5. Э.Петерс. Хаос и порядок на рынках капитала.- М.: Мир, 2000.
6. В.П.Милованов. Неравновесные социально-экономические системы: синергетика и самоорганизация.- М: Эдиториал УРСС, 2001.
7. В.-Б.Занг. Синергетическая экономика. Время и перемены в нелинейной экономической теории. М.: Мир, 1999.

ЗАДАНИЯ ДЛЯ ИНДИВИДУАЛЬНОЙ РАБОТЫ

Выполнить задания в среде MATLAB, дать анализ и интерпретацию полученных результатов, оценить влияние изменения исходных данных на результаты расчетов, выполнить проверку найденного минимума путем сравнения с рядом стоящими точками (для задач оптимизации)

I. Интерполяция и аппроксимация

1. Выполнить интерполяцию функции *y(x)*, x=[-5 –4 –1 3 7], y=[3 1 –4 0 3] используя функцию INTERP1 и методы CUBIC и SPLINE, сравнить результаты, построить графики с шагом 0.1.
2. Выполнить аппроксимацию функции *y(x)*, x=[-5 –4 –1 3 7], y=[3 -1 4 0 3] по методу наименьших квадратов полиномом 3-го порядка, построить графики полинома, заданных точек и локальных ошибок (функция ERRORBAR) вместе на одном поле.
3. Выполнить интерполяцию функции *x(t)*, x=sin(0.67*t)*cos(0.2*t), t=0,1,2,…,20 с помощью а) кусочно-линейной интерполяции, б) кубическими сплайнами. Построить графики *x(t), $x_i(t_i)$*. Шаг изменения времени для графика интерполирующей функции выбрать 0.05.

II. Решение обыкновенных дифференциальных уравнений

1. Решение системы ОДУ (модель Лоттки-Вольтерра «хищник-жертва»):

$$\begin{cases} x' = x \cdot (1 - y), \\ y' = a \cdot y \cdot (x - 1). \end{cases}$$

при *a=0.1, a=0.6, a=1.2* и начальных условиях *x(0)=0.7* и *y(0)=0,01*. Построить для каждого значения *a* графики *x(t), y(t)* и фазовый потрет системы на временном интервале *[0 ; 50]*.

2. Система ОДУ (согласование спроса и предложения при фиксированных ценах):

$$\begin{cases} x' = a_1 \cdot y - c_1 \cdot x \cdot y - b_1 x^2, \\ y' = a_2 \cdot x - c_2 \cdot x \cdot y \end{cases}$$

при $a_1=0.3$, $a_2=0.6$, $b_1=0.12$, $c_1=0.08$, $c_2=0.1$ и начальных условиях *x(0)=0.1* и *y(0)=0.3*. Построить графики *x(t), y(t)* и фазовый потрет системы на временном интервале *[0 ; 50]*.

3. Решение системы ОДУ (стабилизация цены в денежном выражении):

$$\begin{cases} x' = a_1 - a_2 \cdot x \cdot y, \\ y' = b_1 - b_2 \cdot y - b_3 \cdot x \cdot y \end{cases}$$

при $a_1=0.3$, $a_2=0.1$, $b_1=0.52$, $b_2=0.18$, $b_3=0.1$ и начальных условиях *x(0)=1* и *y(0)=0.5*. Построить графики *x(t)* - деньги, *y(t)* - товар и фазовый потрет системы на временном интервале *[0 ; 75]*.

4. Решение системы ОДУ (модель макроэкономической системы):

$$\begin{cases} x' = x \cdot y - a_1 \cdot x, \\ y' = b_1 \cdot x \cdot y - b_2 \cdot y^2 - b_3 \cdot y - b_4 \cdot (x - z), \\ z' = c_1 - c_2 \cdot x. \end{cases}$$

при $a_1=0.21,\ b_1=0.15,\ b_2=0.06,\ b_3=0.08,\ b_4=0.1,\ c_1=0.01,\ c_2=0.03$ и начальных условиях $x(0)=1$ и $y(0)=0.1,\ z(0)=0.75$. Построить графики $x(t)$ – валовой продукт, $y(t)$ – темп роста валового продукта, $z(t)$ - платежеспособный спрос и фазовый потрет системы на временном интервале $[0\ ;\ 100]$.

5. Решение системы ОДУ (модель классификации экономической системы):

$$\begin{cases} x' = a_1 \cdot x \cdot y - a_2 \cdot y^2 - a_3 \cdot y - a_4 \cdot x, \\ y' = b_1 \cdot x \cdot y - b_2 \cdot y. \end{cases}$$

при $a_1=0.3,\ a_2=0.04,\ a_3=0.06,\ a_4=0.08,\ b_1=0.1,\ b_2=0.05$ и начальных условиях $x(0)=1$ и $y(0)=0.1$. Построить графики $x(t)$ – валовой продукт, $y(t)$ – трудовой ресурс и фазовый портрет системы на временном интервале $[0\ ;\ 50]$.

6. Решение одного ОДУ второго порядка: $y'' - (y')^2 + 3y = 2\sin(0.8t)$ с начальными условиями $y(0)=5,\ y'(0)=2$ при $0 < t < 7$. Построить графики функции, производной от времени и друг от друга (фазовый портрет).

7. Решение одного ОДУ третьего порядка: $y''' + 4y'' + 6y' + 2y = 2.5e^{-0.2t}$ с начальными условиями $y(0)=5,\ y'(0)=2,\ y''(0)=1$ при $0 < t < 5$. Построить графики функции, 1-й и 2-й производной от времени и друг от друга.

8. Решение системы ОДУ(модель Весслера):

$$x' = -y - z,$$
$$y' = x + Ay,$$
$$z' = B + z(x - C),$$

с начальными условиями $x(0)= -0.7,\ y(0)= -0.7,\ z(0)=1$, на интервале времени $t = [0\ ;\ 50]$, с параметрами модели $A=0.2,\ B=0.2,\ C=5.7$. Построить для каждого значения $A=0.1,\ 0.2,\ 0.35$ графики $x(t),\ y(t),\ z(t)$ и фазовый потрет системы.

9. Решение нелинейного ОДУ третьего порядка: $y''' + 4y''y + 6y' + 2y = (1 - t^{-2})$ с начальными условиями $y(0)=4,\ y'(0)=2,\ y''(0)=1$ при $1 < t < 5$. Построить графики функции, 1-ой и 2-ой производной от времени и друг от друга.

10. Решение нелинейного ОДУ второго порядка: $y'' + (y^2 - 1)y' + y = 0$ с начальными условиями $y(0)=0.8,\ y'(0)=0.1$, при $0 < t < 20$. Построить графики функции и 1-ой производной от времени и друг от друга.

III. Решить систему нелинейных уравнений

1. Начальные условия $x0 = [1\ 1\ 1]$.

$$2x_1 - x_2 + x_3^3 - e^{-x_3} = 0;$$
$$-x_1 * x_3 + 2x_2 - e^{-x_2} = 0;$$
$$3x_2 - x_3 * x_1 - 2e^{-x_1} = 0.$$

2. Начальные условия $x0 = [1\ 1\ 1]$.

$$x_1 * x_2 - x_1 - x_2 * x_3 - 1 = 0;$$
$$(x_1 - x_2)^4 - x_3 - 10 = 0;$$
$$2x_1 + 3x_2 - x_3^2 * x_1 + 2 = 0.$$

3. Начальные условия *x0* = [1 1 1].

$$-5\ sin(\ x_1\ ) + 2\ x_1 \cdot x_2 + x_3 = 3;$$
$$2\ x_3 \cdot cos(\ x_1\ ) - 0.5\ x_2^3 = -7;$$
$$4\ x_3 - 6\, x_1 \cdot x_2 = 10.$$

IV. Минимизировать функции при наличии ограничений

1. Начальные условия *x0* = [1 1 1].

$$f(x_1, x_2, x_3) = (x_1^2 + x_2 + x_3^2 - 12)^2 + (x_1 + x_2^2 - x_3 - 7)^2;$$
$$2 * x_1 - x_2 * x_3 + 3 \le 0;$$
$$x_3 \ge 0;$$
$$x_1 + x_2 = 5.$$

2. Начальные условия *x0* = [1 1 1].

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 + 4e^{-x_1^2 - x_2^2 - x_3^2} - 6x_1 * x_2 * x_3;$$
$$x_1 + 3 * x_2 * x_3 - 8 \le 0;$$
$$x_1 + x_2 = 4;$$
$$x_3 \ge 0.$$

3. Начальные условия *x0* = [1 1].

$$f\ (\ x_1\ ,\ x_2\ ) = e^{\ x_1^2 + 5\ x_2^2} + x_1^2 - 80\ \ x_2^2;$$
$$x_1 + 2\ x_2^2 \le 1;$$
$$x_1^2 + x_2^2 - 4\ x_1 \le 0;$$
$$x_1^2 + x_2^2 - x_1 - x_2 \le 0.$$

V. Минимизировать функции

Для всех заданий начальные условия *x0* = [1 2 1].

1. $f(x_1, x_2, x_3) = x_1^4 + x_2^4 + x_3^4 + 2 * x_1^2 * x_2^2 - 4 * x_1 * x_3 + 7;$
2. $f(x_1, x_2, x_3) = (x_1^2 + x_2 + x_3^2 - 12)^2 + (x_1 + x_2^2 - x_3 - 7)^2;$
3. $f(\ x_1, x_2, x_3\ ) = x_1^2 + x_2^2 + x_3^2 + 4e^{-x_1^2 - x_2^2 - x_3^2} + 6x_1 * x_2 * x_3;$

VI. Методом линейного программирования решить транспортную задачу

1. Фирма взяла кредит в 600000$ на приобретение грузовиков трех типов стоимостью 10000$, 20000$ и 23000$ каждый, соответственно. Сколько грузовиков каждого типа необходимо приобрести фирме, чтобы получить наибольшую производительность труда в тонно-километрах в день, если известно, что для грузовика 1-го типа требуется один водитель в смену, максимальное число смен три и производительность работы 2100 тонно-километров в смену, для грузовика 2-го типа требуется два водителя в смену, максимальное число смен три и производительность работы 3600 тонно-километров в смену и для грузовика 3-го

типа требуется два водителя в смену, максимальное число смен три и производительность работы 3780 тонно-километров в смену. Кроме того, число грузовиков не должно превышать 30, а число водителей 145.

Целевая функция:

$F = 2100*(a1+2*a2+3*a3) + 3600*(b1+2*b2+3*b3) + 3780*(c1+2*c2+3*c3) \rightarrow max$

Ограничения:

$10000*(a1+a1+a3) + 20000*(b1+b2+b3) + 23000*(c1+c2+c3) <= 600000;$
$a1 + a2 + a3 + b1 + b2 + b3 + c1 + c2 + c3 <= 30;$
$a1 + 2*a2 + 3*a3 + 2*b1 + 4*b2 + 6*b3 +2*c1 +4*c2 + 6*c3 <= 145;$
$a1>=0; a2>=0; a3>=0; b1>=0; b2>=0; b3>=0; c1>=0; c2>=0; c3>=0;$

2. Со складов С1 (50 т), С2 (90 т) и С3 (80 т) выполнить заявки клиентов К1 (120 т) и К2 (100 т) с минимальными затратами, если транспортные издержки по направлениям следующие: С1->К1 = 8 $/т, С1->К2 = 6 $/т, С2->К1 = 11 $/т,С2-> К2 = 15 $/т, С3->К1 = 14 $/т, С3->К2 = 7 $/т,

   Целевая функция:

   $F = 8*X_1+6*X_2 +11*X_3 +15*X_4+14*X_5 +7*X_6 \rightarrow min,$

   Ограничения:

   $X_1+ X_2=50; \quad X_3+ X_4=90; \quad X_5+ X_6=80;$
   $X_1+ X_3+ X_5=120; X_2+ X_4+ X_6=100;$

VII. Методом линейного программирования минимизировать функцию:

Найдите минимум целевой функции $F(x,y) = ax + by$ при указанных ограничениях ($x>=0, y>=0$)

N вар.	***F(x,y)***	*Ограничения*		
1	*3x+2y*	*x+y<=4*	*x+2y>=5*	*2x+y>=6*
2	*3x+2y*	*x+y<=6*	*x+2y>=7*	*2x+y>=8*
3	*3x+2y*	*x+y<=8*	*x+2y>=8*	*2x+y>=8*
4	*3x+2y*	*x+y<=9*	*x+2y>=7*	*2x+y>=6*
5	*3x+2y*	*x+y<=6*	*x+2y>=5*	*2x+y>=3*
6	*3x+2y*	*x+y<=8*	*x+2y>=9*	*2x+y>=7*
7	*3x+2y*	*x+y<=7*	*x+2y>=7*	*2x+y>=7*
8	*3x+2y*	*x+y<=11*	*x+2y>=11*	*2x+y>=11*
9	*3x+2y*	*x+y<=9*	*x+2y>=9*	*2x+y>=9*
10	*3x+2y*	*x+y<=10*	*x+2y>=10*	*2x+y>=10*
11	*2x+5y*	*x+y>=1*	*x+2y<=10*	*2x+y<=10*
12	*2x+5y*	*x+y>=4*	*x+2y<=12*	*2x+y<=12*
13	*2x+5y*	*x+y>=4*	*x+2y<=13*	*2x+y<=13*
14	*2x+5y*	*x+y>=5*	*x+2y<=14*	*2x+y<=14*
15	*2x+5y*	*x+y>=5*	*x+2y<=15*	*2x+y<=15*
16	*2x+5y*	*x+y>=6*	*x+2y<=11*	*2x+y<=8*
17	*2x+5y*	*x+y>=7*	*x+2y<=13*	*2x+y<=9*
18	*2x+5y*	*x+y>=8*	*x+2y<=15*	*2x+y<=10*
19	*2x+5y*	*x+y>=9*	*x+2y<=17*	*2x+y<=11*
20	*2x+5y*	*x+y>=3*	*x+2y<=3*	*2x+y<=4*

ПОЛЕ ДЛЯ ЗАМЕТОК

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....