

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВПО «Воронежский государственный технический
университет»

Кафедра «Ракетные двигатели»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для проведения лабораторных занятий по дисциплине
«Методы математического моделирования»
для студентов специальности 160700.65, 24.05.02
«Проектирование авиационных и
ракетных двигателей» очной формы обучения

Воронеж 2015

Составители: д-р техн. наук Ю.В. Демьяненко
канд. техн. наук А.А. Гуртовой
д-р техн. наук А.В. Кретинин
канд. физ.-мат. наук А.М. Сушков

УДК 629.13

Методические указания для проведения лабораторных занятий по дисциплине «Методы математического моделирования» для студентов специальности 160700.65, 24.05.02 «Проектирование авиационных и ракетных двигателей» очной формы обучения / ФГБОУ ВПО "Воронежский государственный технический университет"; Сост. А.А. Гуртовой, Ю.В. Демьяненко, А.В. Кретинин, А.М. Сушков. Воронеж, 2015. 47 с.

Методические указания содержат описание и руководство по работе с одной из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций – системой *MATLAB*.

Издание соответствует требованиям Федерального государственного образовательного стандарта высшего профессионального образования по направлению 160700.65, 24.05.02 «Проектирование авиационных и ракетных двигателей» дисциплине «Методы математического моделирования».

Рецензент д-р техн. наук, проф. Г.И. Скоморохов.

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. В.С. Рачук.

Издается по решению редакционно-издательского совета Воронежского государственного технического университета.

© ФГБОУ ВПО «Воронежский
государственный технический
университет», 2015

ВВЕДЕНИЕ

Современная компьютерная математика предлагает целый набор интегрированных программных систем и пакетов программ для автоматизации математических расчетов: *Eureka*, *Gauss*, *TK Solver*, *Derive*, *Mathcad*, *Mathematica*, *Maple* и др. *MATLAB* — одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций. Матрицы широко применяются в сложных математических расчетах, например при решении задач линейной алгебры и математического моделирования статических и динамических систем и объектов. Однако в настоящее время *MATLAB* далеко вышла за пределы специализированной матричной системы и стала одной из наиболее мощных универсальных интегрированных СКМ. Слово «интегрированная» указывает на то, что в этой системе объединены удобная оболочка, редактор выражений и текстовых комментариев, вычислитель и графический программный процессор. В целом *MATLAB* — это уникальная коллекция реализаций современных численных методов компьютерной математики, созданных за последние три десятка лет, сочетающихся с мощными средствами графической визуализации. Это делает *MATLAB* весьма удобным средством для расчетов практически в любой области науки и техники. Например, очень широко используется при математическом моделировании механических устройств и систем, в частности в динамике, гидродинамике, аэродинамике, акустике, энергетике и т. д. Этому способствует не только расширенный набор матричных и иных операций и функций, но и наличие пакета расширения (*toolbox*) *Simulink*, специально предназначенного для решения задач блочного моделирования динамических систем и устройств, а также десятков других пакетов расширений. Большинство команд и функций системы реализованы в виде

текстовых m-файлов (с расширением .m) и файлов на языке Си, причем все файлы доступны для модификации.

ЛАБОРАТОРНАЯ РАБОТА № 1

СРЕДА MATLAB. ЭЛЕМЕНТАРНЫЕ ВЫЧИСЛЕНИЯ

Цель данной работы заключается в получении первоначальных навыков работы с системой *MATLAB*.

В работе необходимо реализовать алгоритм расчета теплоемкости по данным калориметрического эксперимента с использованием системы *MATLAB*.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

По умолчанию после запуска *MATLAB* на экране появится комбинированное окно (рис. 1), включающее четыре панели:

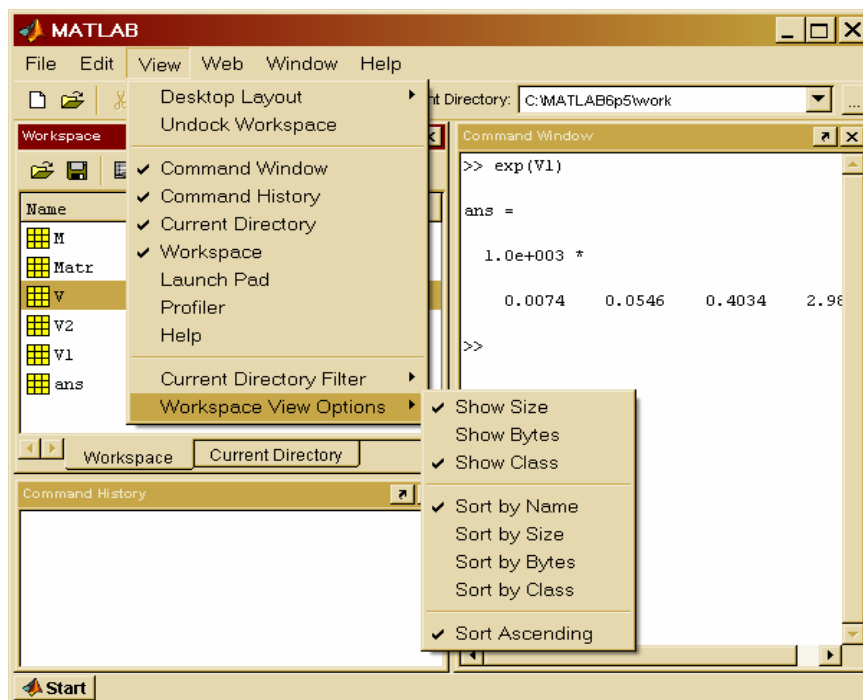


Рис.1. Главное окно MatLab

– *Command Window* (окно команд) – вводятся команды (после знака >>) подлежащие немедленному исполнению (Enter для исполнения).

– *Workspace* (рабочее пространство) – отображает текущий набор переменных, определенных пользователем в командном окне.

– *Command History* (история команд) – хранит команды которые исполнялись в командном окне.

– *Current Directory* (текущий каталог) – определяет каталог с которым работает система (т.е если при выполнении некоторой программы пользователь ссылается на файл с именем «любое», то по умолчанию система будет искать его в текущем каталоге), а так же используется как проводник для создания новых и выполнения уже существующих проектов.

Наличие того или иного окна может быть установлено с помощью вкладки *View* меню главного окна системы. Помимо вкладки *View* главное меню содержит пункты:

– *File* – выполняет обычные функции характерные данному пункту в любых других системах, а именно создания нового (*New*) и открытия (*Open*) уже созданного проекта, (*Save*) сохранения проекта, (*Print*) распечатки командного окна и др.

– *Edit* – выполняет вырезание (*Cut*), копирование (*Copy*), вставку (*Paste*), и удалению (*Delete*) выделенных фрагментов текстов или графических объектов.

– *Web* – подключение к сайту фирмы – производителя.

Вычисления выполняется в командном окне (*Command Window*). После знака “>>” символизирующего начало строки вводится любое выражение удовлетворяющее синтаксису *MATLAB*, далее для выполнения команды нажать “*Enter*”.

Пример 1 :

```
>> a = 4*2 +1          >> a/2
a =                    ans =
  9                    4.5000
```

Если в конце команды поставить “;”(>> a = 4*2 +1;),то вывод полученного значения не произойдет, но переменная *a* будет создана. В этом случае ее значение можно вывести с

помощью команды “>> a”. Альтернативным доступом к значению переменной является редактор переменных (*Array Editor*, рис. 2),

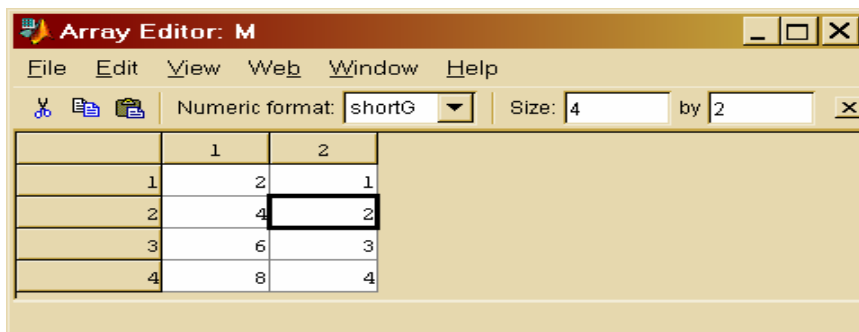


Рис.2. Редактор переменных

запуск которого осуществляется двойным щелчком по переменной в окне *Workspase*. Если введено выражение без использования идентификатора, то по умолчанию результат будет присвоен переменной с именем *ans*(см. предыдущий пример).

Ввод векторов и матриц осуществляется следующим образом:

```
>> V1 = [2 4 6 8]      >> Matr = [1 2 3 4;5 6 7 8]
V1 =
    2    4    6    8
Matr =
    1    2    3    4
    5    6    7    8
```

Матрицу можно получить путем группирования векторов и других матриц одинаковой размерности, например

```
>> V2 = [1 2 3 4]      >> M = [V1;V2]
V2 =
    1    2    3    4
M =
    2    4    6    8
    1    2    3    4

>> M = [V1.' V2.']      >> M.'
M =
    2    1
    4    2
ans =
    2    4    6    8
    1    2    3    4
```

```
6 3
8 4
```

Как можно заметить оператор “ ! ” выполняет транспонирование матрицы. Вектора и матрицы могут быть введены с использованием оператора “ : ”, например :

```
>> V = [1:2:8]           >> V=[1:4:20]
V =                      V =
 1  3  5  7              1  5  9 13 17
```

Оператор “ : ” может быть использован следующим образом :

```
>> M(:,2)      >> M(4,:)      >> M(2:4,:)
ans =          ans =          ans =
 1             8  4           4  2
 2             8  4           6  3
 3             8  4           8  4
 4
```

Представления любого числа в матричной форме определяют специфичную особенность системы, которая может быть продемонстрирована на следующем примере :

```
>> V1 = [2 4 6 8]      >> V2=[1 2 3 4]
V1 =                   V2 =
 2 4 6 8                1 2 3 4
```

```
>> V1/V2             >> V1./V2
ans =                 ans =
 2                     2 2 2 2
```

Как видно из примера арифметические операторы в MATLAB означают матричные отношения. Для поэлементных действий над матрицами перед оператором добавляется точка.

Для удаления созданных переменных используется функция *clear* :

```
>> clear a    >> clear
```


При этом в первом случае будет удалена только переменная a , а во втором все переменные отраженные в окне *Workspace*. *MATLAB* вычисляет элементарные математические функции, например:

```
>> exp(V1)          {вычисление экспоненты}
ans =
  1.0e+003 *
  0.0074  0.0546  0.4034  2.9810
```

Для получения полного списка элементарных функций вычисляемых системой необходимо выполнить команду:

```
>> help elfun
```

Рассмотренный выше пошаговый режим работы используется для разовых вычислений по сути представляет работу системы в режиме калькулятора, и реализация определенного алгоритма представляется весьма обременительной. Для программной реализации предусмотрена возможность последовательного выполнения команд, записанных в текстовом файле. Данный подход составляет основу программирования в *MATLAB*. Для этого в системе имеется свой встроенный текстовый редактор (рис. 3), который помимо ввода текста программы позволяет проводить ее отладку. Доступ к встроенному текстовому редактору при организации нового проекта осуществляется через главное меню *File* → *New* → *M – file* (или *Ctrl – N*).

Операторы в редакторе разделяются знаком “ ; ”. Комментарии выделяются символами “ % ”. Меню окна текстового редактора содержит уже знакомые вкладки, которые выполняют те же функции. С помощью пунктов *Debug* и *Breakpoints* проводится отладка программы. Запуск программы осуществляется из меню окна редактора (*Debug*→*Run*) или использованием клавиши *F5*.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Для закрепления первоначальных навыков работы с системой предлагается выполнить расчет теплоемкости воздуха по полученным экспериментальным данным (табл. 1). При исследованиях оставались постоянными следующие величины:

Площадь проходного сечения шайбы – $F = 7.85 \cdot 10^{-4} \text{ м}^2$

Сила тока в нагревателе – $I = 2 \text{ А}$

Напряжение на нагревателе – $U = 18 \text{ В}$

Плотность воздуха – $\rho = 1.29 \text{ кг/м}^3$

Для начала рекомендуем убрать окно *Command History*. Это можно сделать сняв метку на вкладке *View* → *Command History*.

Таблица 1

№ Вар.	Подогрев воздуха Δt	Давление на входе в шайбу, Па P_1	Давление на выходе из шайбы, Па P_2	Коэффициент расхода шайбы μ
1	5.458056	121935.9	120000.0	0.1257982
2	4.717562	120179.1	120000.0	0.4852207
3	3.911148	121619.1	120000.0	0.1948820
4	3.574879	120796.0	120000.0	0.3003445
5	3.558563	121753.3	120000.0	0.2090244

Далее последовательность действий следующая :

1. Исходные данные вводим используя *Array Editor*. Для этого создаем переменную *Tab* – матрица размером 5×4 .

>> `Tab(5,4)=0 ;`

Двойным щелчком по созданной переменной в окне *Workspase* открываем окно *Array Editor* и вводим таблицу исходных данных.

2. Определяем переменные Δt , P_1 , P_2 , μ , F , I , U , ρ (плотность):

```

>> dt=Tab(:,1); ( $\Delta t$ )
>> P1=Tab(:,2);
>> P2=Tab(:,3);
>> Mu=Tab(:,4); ( $\mu$ )
>> F=7.85e - 4;
>> I=2;
>> U=18;
>> Ro=1.29;

```

3. Расход воздуха определяем по формуле :

```

>> m = Mu. * F. * sqrt(2 * Ro * (P1 - P2)) ;

```

4. Определяем количество тепла >> $Q = I * U$;

5. Определяем изобарную теплоемкость >> $C_p = Q. / (m. * \Delta t)$;

6. В качестве результата сформируем матрицу: >> $Result = [Tab$
 $C_p]$

7. Среднемассовое значение теплоемкости определяется по формуле

$$C_p^{cp} = \frac{\int_{m_1}^{m_5} C_p dm}{(m_5 - m_1)}$$

Интеграл в этой формуле вычислим по методу трапеций с использованием стандартной функции $trapz(x,y)$:

```

>> Cpsr=trapz(m,Cp)/(m(5)-m(1));

```

Вышеизложенный алгоритм (начиная с п. 2) удобно выполнять не по действиям в главном окне, а используя текстовый редактор. С помощью вкладки *File* меню главного окна *MATLAB* открываем текстовый редактор (*File*→*New*→*M-*

File). Набираем в нем приведенный алгоритм. Запускаем (F5 или *Debug*→*Run*) (рис. 4).

ОТЧЕТ ПО РАБОТЕ

Отчет по работе должен содержать информацию об основах вычислений в системе MATLAB. По практической части работы в отчет необходимо включить алгоритм решения и полученный результат, а также представить преподавателю рабочий алгоритм расчета в текстовом редакторе системы.

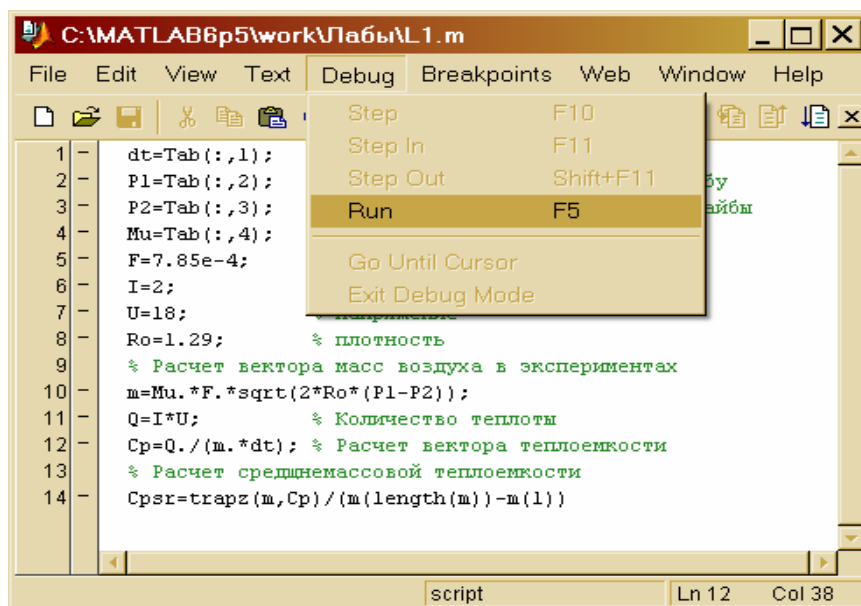


Рис.5. Расчет среднemasсовой теплоемкости воздуха

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем состоят особенности системы MATLAB?
2. Сформировать матрицу M размером 5×8 ?
3. Как сформировать вектор V являющийся строкой (столбцом) матрицы?
4. Как задается вектор с использованием знака ':'?
5. Сформировать матрицу из трех векторов?

ЛАБОРАТОРНАЯ РАБОТА № 2 ВИЗУАЛИЗАЦИЯ В MATLAB

Цель работы: получение навыков визуализации решения задач в системе MATLAB.

В практической части лабораторной работы необходимо построить графическую интерпретацию решения задачи об определении электрического поля порождаемого двумя электрическими зарядами.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Одно из достоинств системы *MATLAB* — обилие средств графики, начиная от команд построения простых графиков функций одной переменной в декартовой системе координат и кончая комбинированными и презентационными графиками с элементами анимации, а также средствами проектирования графического пользовательского интерфейса (*GUI*). Особое внимание в системе уделено трехмерной графике с функциональной окраской отображаемых фигур и имитацией различных световых эффектов. Рассмотрим примеры наиболее часто используемых средств визуализации системы.

Функции одной переменной $y(x)$ находят широкое применение в практике математических и других расчетов, а также в технике компьютерного математического моделирования. Для отображения таких функций используются графики в декартовой (прямоугольной) системе координат. При этом обычно строятся две оси — горизонтальная X и вертикальная Y , и задаются координаты x и y , определяющие узловые точки функции $y(x)$. Эти точки соединяются друг с другом отрезками прямых, т. е. при построении графика осуществляется линейная интерполяция для промежуточных точек. Поскольку *MATLAB* — матричная система, совокупность точек $y(x)$ задается векторами X и Y

одинакового размера. Для построения таких графиков используется процедура *fplot*, например:

```
>>x=[0:0.1:6.28];  
>>y=sin(x).*0.5;  
>>plot(x,y);
```

Результатом выполнения будет график функции $y(x) = 0.5 \sin(x)$

Если обратиться к процедуре *plot* повторно, то она создаст новое окно с заголовком Figure No.2 и разместит в нем следующий график. Совместить оба графика в одном окне можно двумя разными способами. В первом случае перед вызовом процедуры *plot* мы должны построить таблицы обеих функций, например, $(x1,y1)$ и $(x2,y2)$, и обратиться к процедуре следующим образом (результат на рис. 1):

```
>>x1=[0:0.1:6.28];  
>>y1= sin(x1).*0.5;  
>>y2=cos(x1).*0.5;  
>>plot(x1,y1,x1,y2);
```

Второй способ заключается в блокировании режима создания нового графического окна с помощью процедуры *hold on*. Так, например, предыдущий результат можно получить так:

```
>>plot(x1,y1);  
>>hold on;  
>>plot(x1,y2);
```

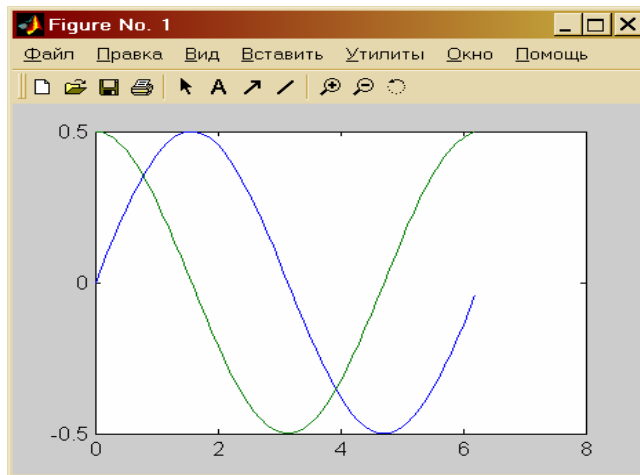


Рис. 1

Заметим, что график второй функции отображен другим цветом. Повлиять на выбор цвета графика может пользователь, указав в процедуре дополнительный третий параметр:

```
>>plot(x,y,'g')
```

В данном случае выводимый график будет зеленого цвета (синий по умолчанию). Дополнительно предусмотрена возможность задания типа линии и маркера, которые указываются вместе с цветом (порядок роли не играет см. рис. 3). Дополнительно можно снабдить график заголовком (процедура *title*), подписать оси (процедуры *xlabel,ylabel*), нанести координатную сетку (процедура *grid on*) и разместить легенду (процедура *legend*). Используем все эти возможности для рис. 1 (рис. 2) :

```
>>x1=[0:0.2:6.28];
>>y1= sin(x1).*0.5;
>>y2=cos(x1).*0.5;
>>plot(x1,y1,'k-s',x1,y2,'b--o'); % k – черный, '-' сплошная, 's' –
маркер квадратный.
```

```

>>legend('sin', 'cos',4);
>>ylabel('y');
>>xlabel('x');
>>grid on;
>>title('Function sin & cos');

```

В процедуре `legend` третий параметр указывает место расположения легенды :

- -1 – легенда помещается вне поля графика, вверху справа.
- 0 – система выбирает лучшее место в поле графика не перекрываемое данными.
- 1,2,3,4 – по углам в поле графика.

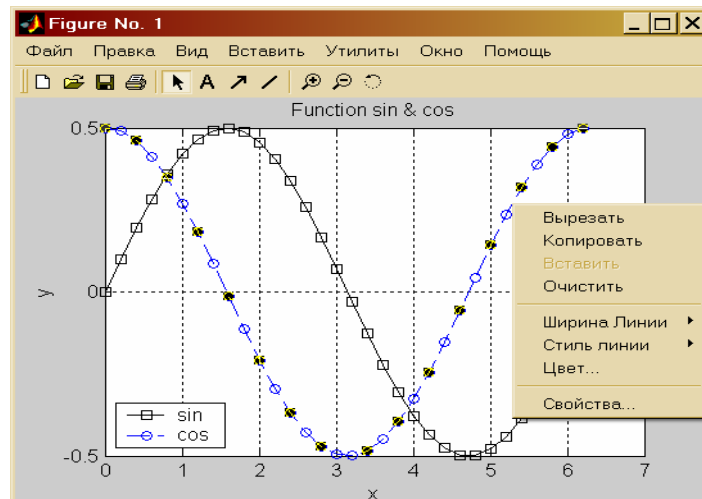


Рис. 2

Редактирование графика можно производить непосредственно из окна `Figure`. Для этого следует выбрать в строке меню окна стрелку, затем выбрав необходимый для редактирования график, с помощью правой кнопки мыши открыть меню (рис. 3), используя которое можно установить нужный цвет, маркер, стиль линии и т.д. Пункт “вставить” (*Insert*) главного меню окна можно использовать

для создания легенды, заголовка, поясняющих надписей в поле графика и т.д. Для корректировки свойств осей (масштаба и т.д.) необходимо использовать пункт “правка”→”свойства осей”(Edit→*Axes properties*).

На практике большое распространение получили функции двух переменных, графическим изображением которых является трехмерная поверхность. В *MATLAB* такие поверхности должны быть заданы узлами сетки размерности $m \times n$, заполняющей на плоскости Oxy прямоугольную область, и массивом той же размерности, представляющим значения функции в узлах этой сетки.

Для формирования равномерной прямоугольной сетки удобно воспользоваться функцией *meshgrid* :

```
>> x=[0 1 2 3 4 5];
>> y=[0 2 4];
>> [X,Y]=meshgrid(x,y)
X =           Y =
    0     1     2     3         0     0     0     0
    0     1     2     3         2     2     2     2
    0     1     2     3         4     4     4     4
```

Для отображения поверхности используются две основные функции – *mesh* и *surf*. Первая строит проволочный каркас поверхности, вторая – закрашенную поверхность. Приведем пример построения поверхности заданной равномерной сеткой с шагом 0.25 и функцией $Z(x, y) = x \cdot \exp(-x^2 - y^2)$ (рис. 4):

```
>> [X,Y]=meshgrid([-2:0.25:2]);
>> Z=X .* exp(- X .^2 - Y.^2);
>> mesh(X, Y, Z);
>> xlabel('X'), ylabel('Y'), zlabel('Z');
```

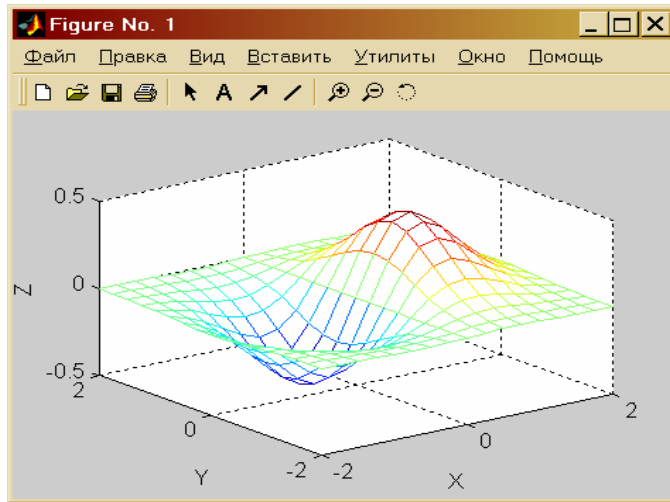


Рис. 3

Закрашивание каждой чешуйки поверхности осуществляется функциями *surf* и *surfz*. Используя предыдущую сетку и функцию обратимся к функции *surf* (рис. 4):

```
>> surf(x,y,Z);
>> xlabel('X'),ylabel('Y'),zlabel('Z');
```

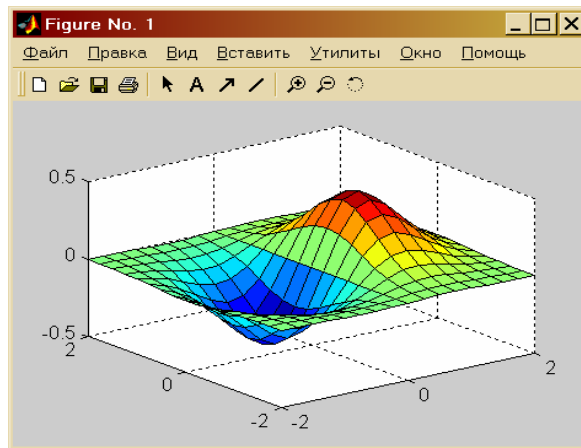


Рис. 4

ПРАКТИЧЕСКАЯ ЧАСТЬ

Рассмотрим электрическое поле создаваемое двумя одинаковыми зарядами ($q = 1 \cdot 10^{-9}$ Кл) находящимися в вакууме на достаточно большом расстоянии друг от друга. Поставим задачу графической визуализации рассматриваемого поля.

Задачу будем решать на равномерной сетке $x = -0.7, 0.7; y = -0.7, 0.7$ с шагом 0.1. Центр системы координат Оху расположим по середине между зарядами, а ось у направим перпендикулярно линии их соединяющей. Тогда координаты зарядов будут $(x_1, 0)$ и $(x_2, 0)$ (примем $x_1 = -5$, $x_2 = 5$), а проекции напряженности, создаваемой в точке поля с координатой (x, y) i – м зарядом, на оси координат определятся по формуле :

$$\begin{aligned} E_x^{(i)} &= C \cdot \frac{q \cdot (x - x_i)}{\sqrt{((x - x_i)^2 + y^2)^3}} \\ E_y^{(i)} &= C \cdot \frac{q \cdot y}{\sqrt{((x - x_i)^2 + y^2)^3}} \end{aligned} \quad (1)$$

Где $C = \frac{1}{4 \cdot \pi \cdot \epsilon_0}$ ($\epsilon_0 = 8.85 \cdot 10^{-12}$ электрическая постоянная)

Суммарную напряженность в точке определим согласно закону:

$$\begin{aligned} E_x &= E_x^{(1)} + E_x^{(2)} \\ E_y &= E_y^{(1)} + E_y^{(2)} \\ E(x, y) &= E_x^2 + E_y^2 \end{aligned}$$

Потенциал в точке поля определим согласно выражению:

$$P_i(x, y) = \frac{q}{\sqrt{((x - x_i)^2 + y^2)}} \quad (2)$$

$$P(x, y) = P_1(x, y) + P_2(x, y)$$

Реализация приведенного алгоритма в *MATLAB* будет выглядеть следующим образом :

1. Ввод исходных данных :

```
>> x1=-5 ;
>> x2=5;
>> q=1e-9;
>> E0=8.85e-12;
>> C=1/(4*pi*E0);
```

2. Формирование сетки

```
>> x0=[-0.7:0.1:0.7];
>> y0=[-0.7:0.1:0.7];
>> [x,y]=meshgrid(x0,y0);
```

3. Вычисление напряженности в узлах сетки :

```
>> E1x=C*q.*(x-x1)./(sqrt(((x-x1).^2+y.^2).^3));
>> E1y=C*q.*y./(sqrt(((x-x1).^2+y.^2).^3));
>> E2x=C*q.*(x-x2)./(sqrt(((x-x2).^2+y.^2).^3));
>> E2y=C*q.*y./(sqrt(((x-x2).^2+y.^2).^3));
>> Ex=E1x+E2x;
>> Ey=E1y+E2y;
>> E=Ex.^2+Ey.^2;
```

4. Вычисление потенциала :

```
>> R1=sqrt((x-x1).^2+y.^2);
>> R2=sqrt((x-x2).^2+y.^2);
>> P1=q./R1;
>> P2=q./R2;
>> P=P1+P2;
```

5. Для построения графиков разобьем графическую область окна Figure на 4 подобласти с помощью функции subplot :

```
>> subplot(2,2,1)
```

Первое число указывает число окон по вертикали, второе – по горизонтали, третье – номер окна в которое будет выведен текущий график (нумерация окон слева – направо).

6. Строим график потенциала:

```
>> surf(x, y, P);
```

7. Строим график величины напряженности в следующем окне:

```
>> subplot(2,2,2);
```

```
>> surf(x, y, E);
```

8. Для построения векторного поля напряженности воспользуемся функцией quiver :

```
>> subplot(2,2,3);
```

```
>> quiver(x, y, Ex, Ey);
```

Данная функция строит векторное поле по значениям проекций вектора в данной точке. При этом необходимо следить, чтобы размерность x , y , E_x , E_y была одинаковой.

9. Построим графики сечений поверхности напряженности в направлении оси x и в направлении y следующим образом:

```
>> subplot(2,2,4);
```

```
>> plot(y0,E(:,10),'k.--',x0,E(10,:),'-o');
```

Приведенный алгоритм рекомендуется выполнить с помощью текстового редактора *MATLAB* (см. работу №1).

Полученный графический результат необходимо дополнить соответствующими пояснениями как показано на рис. 9., с использованием меню окна Figure.

ОТЧЕТ ПО РАБОТЕ

Отчет по работе должен содержать все необходимые сведения для построения графиков функций одной переменной и поверхностей, а так же дополнительных возможностей для пояснений к графикам. Результатом выполнения практической части должен быть рисунок, приведенный на рис. 9 (сохранить в формате jpg).

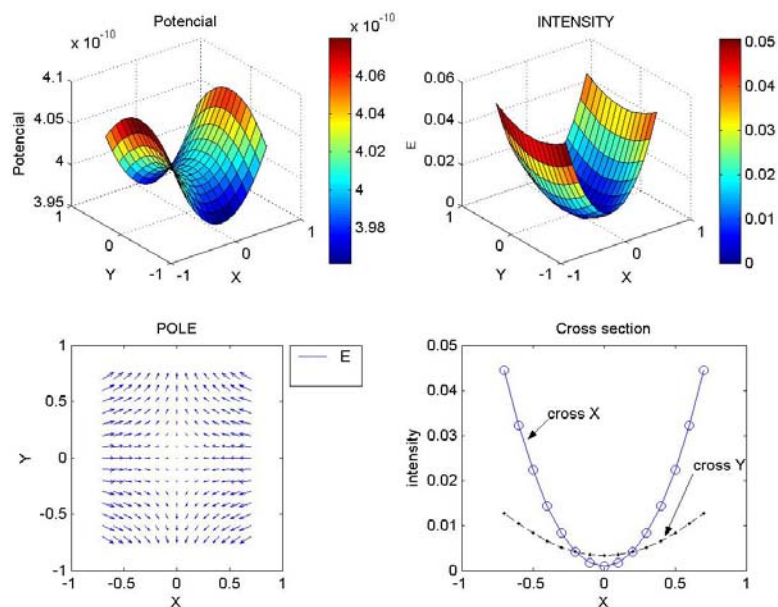


Рис. 9

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Графические возможности системы MATLAB?
2. Системные процедуры, используемые для построения графика функции одной переменной?
3. Как задается равномерная сетка в плоскости Oxy ?

4. Системные процедуры, используемые для построения поверхностей?
5. Какие возможности предусмотрены в системе для пояснения графиков и их редактирования?
6. Какую функцию необходимо использовать для разбиения графического окна на несколько областей?
7. С помощью какой функции строится векторное поле?

ЛАБОРАТОРНАЯ РАБОТА № 3 ОСНОВЫ ПРОГРАММИРОВАНИЯ В MATLAB

Цель работы : изучение особенностей программирования в системе MATLAB.

В практической части работы предлагается написать программу определения показателя политропы.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Практически невозможно предусмотреть в одной, даже самой большой и мощной, математической системе возможность решения всех задач, которые могут интересовать пользователя. Программирование в системе *MATLAB* является эффективным средством ее расширения и адаптации к решению специфических проблем. Оно реализуется с помощью языка программирования системы. Большинство объектов этого языка, в частности все команды, операторы и функции, одновременно являются объектами входного языка общения с системой в командном режиме работы. Программы на языке программирования *MATLAB* сохраняются в виде текстовых *m*-файлов.

Приведенные выше примеры программ (см. работы №№ 1, 2) часто называют скриптами (*script* – файлы). Более удобной разновидностью *m* – файлов являются функции, первой строкой которых является заголовок, использующий оператор

function. Использование функций является удобным средством разделения программы на отдельные блоки алгоритмов, обратиться к которым можно из любой точки программы.

Функции получают исходные данные в виде списка входных параметров и возвращают результаты своей работы также в виде списка выходных параметров. Одной из самых важных особенностей функций является аппарат локальных переменных. Все переменные, появляющиеся в теле функции, за исключением глобальных переменных, входных и выходных параметров, считаются локальными. Они образуют локальное рабочее пространство и доступны только в теле породившей их функции, и никакие скрипты или другие функции воспользоваться ими не могут. При написании программ – функций требуется, чтобы имя *m* – файла, в котором запоминается программа обязательно совпадало с именем функции.

Пример универсальной программы вычисления $n!$ может быть написан следующим образом и должен быть сохранен под именем *fact.m* :

```
function y=fact(n)
k=1
for i=1:n
    k=k*I ;
end
y=k ;
```

Для вычисления $5!$ Достаточно набрать в текущей строке :

```
>> p=fact(5)
p=
    120
>>
```

Отметим, что при обращении к функции должно выполняться соответствие между числом входных и выходных параметров. Так например обращения типа:

```
>> [k1,k2]=fact(5)
```


??? Ошибка использования ==> fact
Слишком много выходных аргументов.

```
>> k1=fact(5,6)
```

??? Ошибка использования ==> fact
Слишком много входных аргументов.

```
>> [k1,k2]=fact(5,6)
```

??? Ошибка использования ==> fact
Слишком много входных аргументов.

вызовут ошибку, и работа программы будет прекращена.

В любом m – файле можно описать несколько функций (в скриптах не допускается описание функций). Самая первая из них обладает тем преимуществом, что может быть вызвана извне. Все остальные функции считаются внутренними и доступны только в рамках данного m – файла. Функция может не содержать входных и выходных параметров. Если такая функция является внешней, то она выполняет роль скрипта. Однако в отличие от скрипта в ней могут присутствовать другие подфункции, отпадает необходимость в использовании функций как отдельных m – файлов, и текст программы можно целиком поместить в одном файле, а внешние подключать как дополнительные библиотеки (например файлы данных). Поэтому рекомендуем для написания программы отказаться от скриптов и оформлять программу сразу в виде функции без входных и выходных параметров.

Входной язык Matlab насчитывает всего 9 операторов, использующих 14 служебных слов. Соответствующие синтаксические конструкции приведены ниже (табл. 1):

Таблица 1

№	Формат оператора	Пояснение
1	var =expr	Оператор присваивания, вычисляет значение выражения expr и заносит результаты вычислений в переменную var
2	if условие 1 операторы_1 elseif условие_2 операторы_2 elseif условие_3 операторы_3 else операторы end	Условный оператор. Если справедливо условие_1, то выполняется группа операторы_1, если справедливо условие_2, то выполняется группа операторов_2,... Если все указанные условия не справедливы, то выполняются операторы, расположенные между else и end
3	switch expr case var_1 операторы_1 case var_2 операторы_2 Otherwise операторы end	Переключатель по значению выражения expr. Если оно совпадает с величиной var_1, то выполняются операторы_1, ... Если expr не совпадает ни с одной из перечисленных величин, то выполняются операторы расположенные между otherwise и end
4	for var=e1:e2:e3 операторы end	Цикл типа арифметической прогрессии, в котором переменная var при каждом повторении тела цикла изменяется от первоначального значения e1 с шагом e2 до конечного значения e3.

5	<code>while условие операторы end</code>	Цикл с предусловием, повторяющийся до тех пор пока истинно указанное условие.
6	<code>try операторы_1 catch операторы_2 end</code>	Попытка выполнить группу операторы_1. При условии, что в результате их выполнения произошла ошибка, то управление передается группе операторы_2. Если ошибка не произошла выполняются операторы 1.
7	<code>break</code>	Досрочный выход из конструкций типа <code>for</code> , <code>while</code> , <code>switch</code> , <code>try-catch</code>
8	<code>function [y1, ...]=fun(x1, ...)</code>	Заголовок функции. <code>y1, y2, ...</code> - список выходных параметров. <code>x1, x2, ...</code> - список входных параметров.
9	<code>return</code>	Досрочный выход из тела функции.

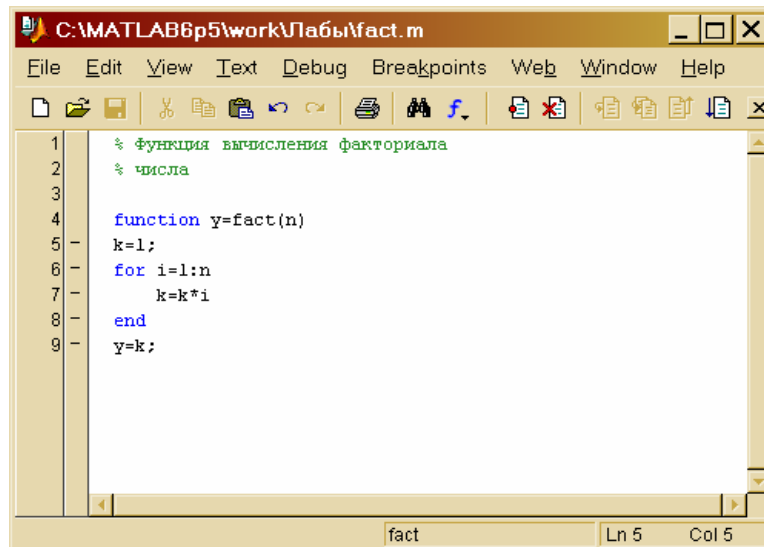
Комментарии начинаются с символа `%` и располагаются с начала строки либо правее любого последнего оператора строки. Начальные комментарии выполняют особую роль. Первая группа строк с подряд идущими комментариями до пустой строки образует текст, выдаваемый в командном окне по команде `help` с именем `m` – файла. Добавим, например, к приведенной выше программе вычисления `5!` Следующий комментарий (рис. 1):

Выполним команду:

```
>> help fact
```

Функция вычисления факториала числа

>>



```
C:\MATLAB6p5\work\Лабы\fact.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons]
1 % функция вычисления факториала
2 % числа
3
4 function y=fact(n)
5 k=1;
6 for i=1:n
7 k=k*i
8 end
9 y=k;
```

Рис. 1

Важным при написании программ является взаимодействие пользователя с программой. Простейший способ ввода числовой и символьной информации, набираемой с клавиатуры, использует функцию `input`. Эта функция допускает два формата обращения :

```
>> x=input('Введите x= ');
```

Введите x=

В данном случае программа ожидает ввода (с клавиатуры) значения, которое будет присвоено переменной `x`. Вместо конкретного значения пользователь может ввести выражение, величина которого будет подсчитана с учетом текущего состояния переменных рабочего пространства.

Второй формат обращения к функции `input` имеет вид :

```
x=input('Введите x','s');
```

В данном случае текст, вводимый пользователем, будет рассматриваться как строка символов.

Для того чтобы вывести на экран значение некоторой переменной можно исключить знак ';' в конце строки, например :

```
>> x= 5;
>> x
x=
    5
```

```
>>
```

Более гибким вариантом вывода является использование функции disp :

```
>> disp(x)
    5
>> disp('x^2+3')
x^2+3
>>
```

Как видно данная функция не отображает имя выводимой переменной и дает возможность дополнить вывод определенными пояснениями, например :

```
>> disp(strcat('Значение переменной x= ',
'num2str(x)'));
Значение переменной x=5
>>
```

В данном примере текст выводимый функцией disp был составлен из двух строк : ' Значение переменной x= ', '5' (функция num2str(x,n) преобразует значение переменной x в строковый тип, n – количество цифр дробной части). Функция strcat('s1','s2') выполняет операцию горизонтального объединения двух строк s1 и s2. Для вертикального объединения можно воспользоваться функцией strvcat :

```
>> disp(strvcat('Значение переменной x= ',
'int2str(x)));
Значение переменной x=
5
```

>>

Отладка программы производится с помощью пунктов Debug и Breakpoint меню окна текстового редактора системы, путем установки точек останова программы. Выполнение программы приостанавливается при достижении строки помеченной точкой как показано на рис. 2. При этом можно проверить значения переменных, выполнить определенные действия по определению корректного значения переменных и т.д. Дальнейшее выполнение программы можно организовать пооператорно (для продолжения нажать F10), что облегчает поиск ошибки, а можно и в обычном режиме (до конца или до следующей точки останова) используя клавишу F5. Как видно из рис.2 с помощью пункта Breakpoints можно регулировать условия останова выполнения программы в случае появления ошибки (stop if error), недопустимого значения переменных (stop if NaN Or Inf) или появлении предупреждения (stop if warning).

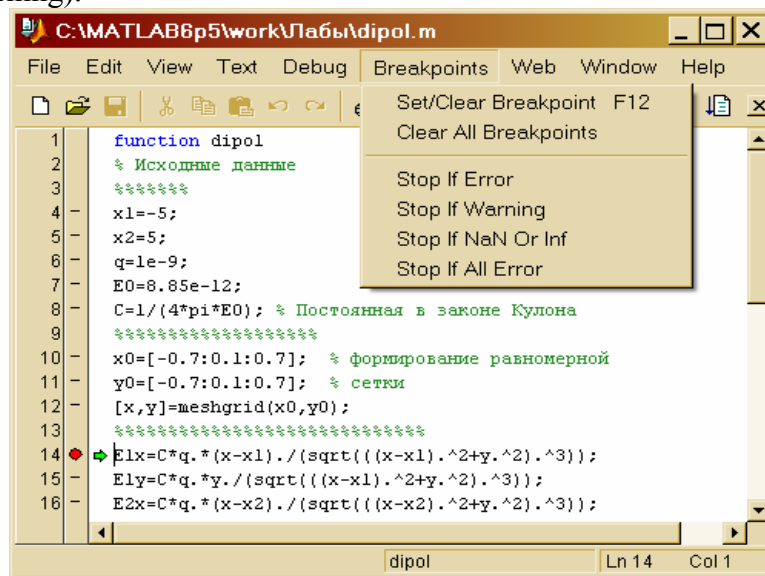


Рис. 2

ПРАКТИЧЕСКАЯ ЧАСТЬ

В процессе сжатия газа (воздуха) определялось значение его начального и конечного объема и давления. Требуется определить показатель политропы процесса сжатия.

Программу для решения поставленной задачи удобно разделить на три части:

1. Ввод исходных данных: начального и конечного объема, давления.
2. вычисление показателя политропы.
3. Ответ.

Для каждой из рассмотренных частей напишем свою функцию:

1. Функция ввода исходных данных:

Ввод исходных данных будем осуществлять с помощью стандартной функции системы меню (рис. 3):

```
>> k=menu('Заголовок', 'Кнопка1', 'Кнопка2', 'Кнопка3',...);
```

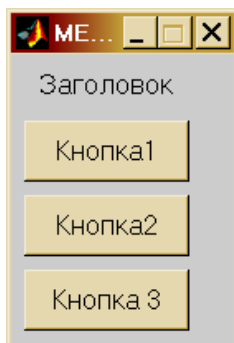


Рис. 3

При выборе одной из кнопок переменная k будет принимать значения 1,2,3,... в зависимости от номера выбранной кнопки в списке функции `menu`.

```
function [P0,V0,P1,V1]=Vvod
```

```

P0=100000;V0=0.1;P1=100000;V1=0.1
n=0;
while n<5
    n=menu('Исходные данные',...
        strcat('P0= ',num2str(P0)),...
        strcat('V0= ',num2str(V0)),...
        strcat('P1= ',num2str(P1)),...
        strcat('V1= ',num2str(V1)),...
        'Расчет');
    switch n
        case 1
            P0=input('P0= ');
        case 2
            V0=input('V0= ');
        case 3
            P1=input('P1= ');
        case 4
            V1=input('V1= ');
    end
end
end

```

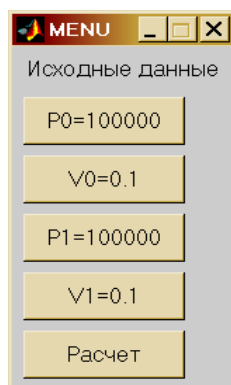


Рис. 4
Троеточие используется для переноса строки.

Работа функции осуществляется следующим образом : при выборе одной из кнопок (например первой), переменной n будет присвоено значение равное 1, которое сигнализирует программе о необходимости изменить параметр соответствующий кнопке 1. Выбор и изменение параметра происходит посредством оператора switch. Так как условие оператора while выполнено (n=1<5), то цикл повторится и появится меню с новым значением измененного параметра. Выход из цикла осуществится только при выборе кнопки 'Расчет', так как значение n=5, и условие цикла нарушается.

В результате выполнения данной функции будет создано меню (рис. 4), которое будет позволять пользователю изменять значения исходных данных.

Функция Vvod возвращает значения исходных параметров заданных пользователем.

2. Рассчитываем политропы сжатия:

$$n = \frac{\ln\left(\frac{P_0}{P_1}\right)}{\ln\left(\frac{V_1}{V_0}\right)}$$

3. Сформируем ответ в виде функции:

```
function Otvet(P0, P1, V0, V1, n)
clc; % выполняет очистку главного окна системы
disp(' Параметры процесса ');
disp(strcat(' Начальное давление P0: ',num2str(P0)));
disp(strcat(' Конечное давление P1: ',num2str(P1)));
disp(strcat(' Начальный объем V0: ',num2str(P0)));
disp(strcat(' Конечный объем V1: ',num2str(P0)));
disp(strcat(' Показатель расширения n : ',num2str(L)));
```

```

4. Общую функцию назовем Pokazatel:
function Pokazatel
clc;
[P0,V0,P1,V1]=Vvod;    % осуществляем ввод данных
% Определяем n
n=log(P0/P1)/log(V1/V0);
%Выводим результат расчета
Otvet(P0,V0,P1,V1,n);

```

Функция Pokazatel должна располагаться самой первой в тексте программы.

На рис. 5. приведен фрагмент описанной программы в текстовом редакторе системы.

ОТЧЕТ ПО РАБОТЕ.

Отчет по работе должен содержать сведения о использовании подпрограмм в языке программирования системы *MATLAB*, а так же синтаксис используемых операторов. В отчет необходимо внести функции осуществляющие ввод вывод. С помощью использования программы необходимо произвести расчет для различных исходных данных (Табл.2):

Таблица 2

№ Вар.	Начальное давление газа, Па p_0	Начальный объем газа, m^3 v_0	Конечное давление газа, Па p_1	Конечный объем, m^3 v_1
1	2	3	4	5
1	100000.0	0.10	110006.7	9.1052033E-02
2	100000.0	0.10	290405.8	4.5984197E-02
3	100000.0	0.10	400386.7	2.7131518E-02
4	100000.0	0.10	264492.8	4.3083698E-02
5	100000.0	0.10	188871.8	6.1478741E-02
6	100000.0	0.10	136864.3	7.7425711E-02
7	100000.0	0.10	284668.9	4.5706544E-02

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Варианты формата оператора вызова функции. Особенности использования внутренних параметров функции?
2. Какие операторы предусмотрены в системе для организации циклов ? Условный оператор?
3. Функции для организации ввода вывода?
4. как осуществляется отладка программы?
5. Как работает функция menu?
6. С помощью каких функций осуществляется операция конкатенации (объединения) строк и преобразования числовых данных в строковый тип?
7. В чем состоит недостаток использования script – файлов?

ЛАБОРАТОРНАЯ РАБОТА № 4

ИНТЕРПОЛЯЦИЯ. ЛИНЕЙНЫЕ, НЕЛИНЕЙНЫЕ УРАВНЕНИЯ И СИСТЕМЫ. МИНИМИЗАЦИЯ ФУНКЦИИ

Цель работы : рассмотреть основные функции предоставляемые системой MATLAB для интерполяции данных, а так же решения систем линейных и нелинейных уравнений, поиска точек экстремумов функций.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Рассмотрим некоторые функции предоставляемые системой MATLAB для решения задач весьма часто встречающихся на практике.

Интерполяция

Предположим, что в результате эксперимента или реализации численного метода была получена зависимость

между величинами x и y в табличной форме. При этом часто для анализа полученных результатов необходимо иметь аналитическую зависимость $y=f(x)$. Задача интерполяции состоит в построении аналитической зависимости в некотором смысле приближающей полученные табличные данные.

Для аппроксимации может быть выбран полином степени $n - 1$ (n – количество наблюдений) :

$$y(x) = \sum_{m=1}^n p_m x^{n-m}$$

Коэффициенты p_m полинома (1), проходящего через заданные табличные узлы $[x_i, y_i]$, могут быть найдены с помощью функции `vander`, которая создает матрицу Вандермонда. (см. пример 1).

Пример 1:

```
>> x=[0.05; 0.1; 0.17; 0.25; 0.3; 0.36];
>> y=[0.05; 0.1003; 0.1717; 0.2553; 0.3093; 0.3764];
>> P = vander(x)\y
```

```
P=
- 4.5785
  5.0346
- 1.6381
  0.3434
  0.9746
  0.0006
```

Вектор P содержит значения коэффициентов p_m полинома $y(x)$.

Можно выбрать полином, который дает наименьшее среднеквадратическое отклонение в узлах с помощью функции `polyfit`:

```
P=polyfit(x,y,N); % N – степень полинома.
```

Для вычисления значения полинома в любой точке отрезка $[x_1, x_n]$, можно использовать функцию `polyval`: $y=polyval(P,x)$.

На практике широкое распространение получила аппроксимация сплайнами. Сплайн интерполяция

осуществляется с помощью функций `spline` и `pchip`, обращение к которым может иметь вид :

```
ys = spline(x, y, xs)   ys = spline(x, y)
ys = pchip(x, y, xs)   ys = pchip(x, y)
```

В первом случае в переменную `ys` будет возвращено значение в точке `xs` полученное путем сплайн интерполяции на содержащем данную точку отрезке. Во втором случае будет возвращена кусочно – полиномиальная форма – таблица содержащая каждый интервал и значения коэффициентов кубического полинома использованного для интерполяции на данном отрезке (рис. 1).

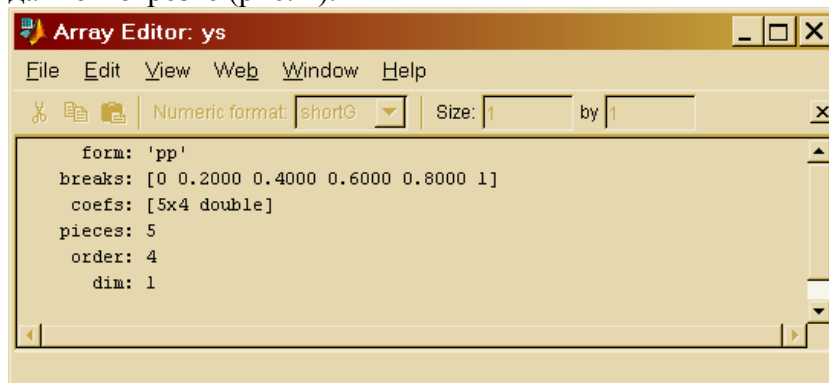


рис. 1

Доступ к значению коэффициентов можно получить следующим образом :

```
>> x=-3:3;
>> y=[-1 -1 -1 0 1 1];
>> ys=spline(x, y);
>> koef=ys.coefs
koef =
    0.2500  -0.7500   0.5000  -1.0000
    0.2500   0.0000  -0.2500  -1.0000
   -0.2500   0.7500   0.5000  -1.0000
   -0.2500   0.0000   1.2500   0.0000
    0.2500  -0.7500   0.5000   1.0000
    0.2500   0.0000  -0.2500   1.0000
```

В переменную `koef` будет помещена матрица коэффициентов (первый столбец соответствует 3 – ей степени и т.д.) строки которой соответствуют заданным интервалам. Получить значение функции в точке по полученной кусочно – полиномиальной форме можно с помощью функции `ppval` : $y = \text{ppval}(x_s, y_s)$, где y_s – кусочно – полиномиальная форма, а x_s – точка в которой необходимо вычислить функцию.

Приведем пример работы функций `spline` и `pchip` (рис. 2):

```
function Prog
x=-3:3;
y=[-1 -1 -1 0 1 1 1];
t=[-3.5:0.05:3.5];
ps = spline(x, y);
pp= pchip(x, y);
plot(x, y, 'ko', t, ppval(ps, t), 'k-', t, ppval(pp, t), 'k-');
legend('(x, y)', 'spline', 'pchip', 2);
title('Spline Pchip');
xlabel('x'); ylabel('y');
```

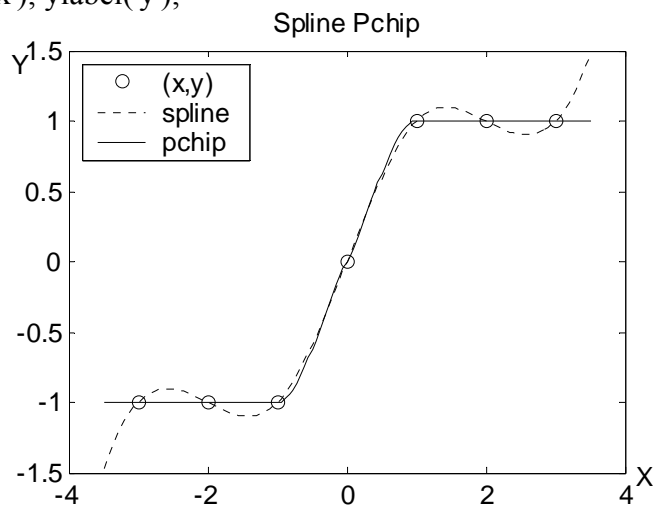


Рис. 2

Для интерполяции отрезками прямых можно воспользоваться функцией `interp1` :

```
ys=interp1(x, y, xs);
```

x , y – узлы интерполяции, xs – абсциссы контрольных точек в которых вычисляется значение табличной функции.

Нелинейные уравнения

Для решения нелинейных уравнений вида $f(x)=0$ используется функция `fzero`. Алгоритм реализованный ею представляет собой комбинацию хорошо известного метода бисекции, метода секущих и метода обратной квадратичной интерполяции. Обращение к функции в простейшем случае выглядит следующим образом:

```
x=fzero(fun,x0)
```

Аргумент `fun` представляет собой функцию $f(x)$, которая может быть задана как:

1. Формула с неизвестным x , заключенная в одинарные кавычки.
2. Имя m – файла в одинарных кавычках и без расширения m .
3. Указатель на функцию.

Указатель на функцию создается с помощью значка `@`, и представляет в системе отдельный тип данных (`function handle`). Так ,например, если имя функции `funcs` (или имеется отдельный файл `funcs.m`), то указатель на эту функцию создается следующим образом:

```
>> h=@funcs.
```

И обращение к функции `fzero` будет выглядеть так :

```
>> x=fzero(h,x0);           % возможна прямая запись  
x=fzero(@funcs,x0);
```

Аргумент x_0 может быть задан одним из двух способов:

1. Как вектор $[a \ b]$ представляющий интервал $(a<b)$, на котором функция $f(x)$ меняет знак.

2. Как скалярное значение, окрестности которого предполагается нахождение корня.

Чтобы облегчить работу по выбору начального приближения, разумнее всего построить график функции $f(x)$.

Приведем пример решения уравнения нелинейного уравнения

$$x \cdot e^{-x} + \sin(x):$$

```
>> x=0:0.1:2*pi;  
>> plot(x, x .*exp(-x)+sin(x),'k ');  
>> grid on  
>> title('y=x*exp(-x)+sin(x)');  
>> xlabel(' x ');  
>> ylabel(' y ');
```

Из графика (рис. 3) видно, что один корень находится на интервале $[3, 4]$. И этой информацией естественно воспользоваться при обращении к функции `fzero` :

```
>> x = fzero(' x .*exp(-x)+sin(x)',[3 4])  
x = 3.2665
```

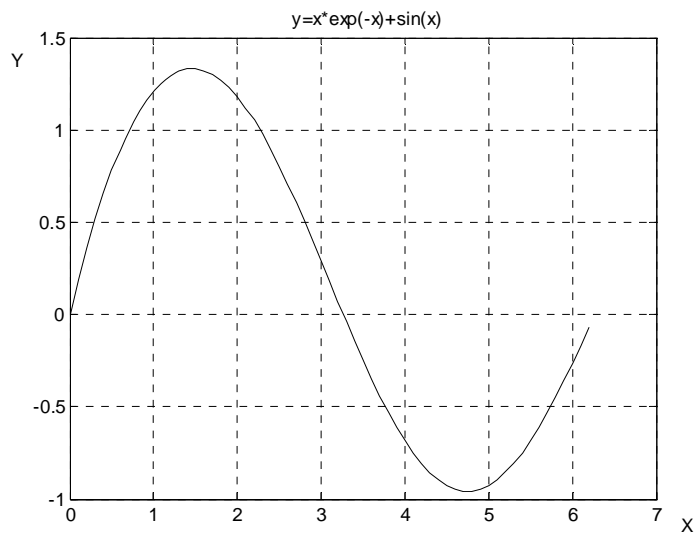


Рис. 3

Если бы уравнение было сохранено в виде m – файла с именем F1.m,
function y=F1(x)
y= x*exp(-x)+sin(x);
то обращение к функции fzero могло бы иметь вид
>> x= fzero(@F1,x0)
x =
3.2665.

Решение систем линейных и нелинейных уравнений

В силу специфики системы MATLAB решение совместной системы линейных уравнений может быть осуществлено с помощью левого матричного деления:

```
>> y=A\b;
```

где x – вектор неизвестных, b – вектор свободных членов системы, A – матрица коэффициентов системы.

Для решения системы можно воспользоваться функцией возвращающей обратную матрицу – $\text{inv}(A)$, тогда решение может быть представлено следующим образом :

```
>> x=inv(A)*b;
```

Для определения совместности имеющейся системы можно вычислить определитель матрицы A , воспользовавшись функцией $\text{det}(A)$.

Если система является несовместной , то можно воспользоваться псевдорешением. Тогда в качестве решения будет получен вектор x , который при подстановке в исходную систему дает вектор невязки ($R=A*x - b$) минимальный по евклидовой норме (метод наименьших квадратов). Очевидно, что для совместной системы псевдорешение является обычным решением. Псевдорешение ищется следующим образом :

```
>> x=pinv(A)*b;
```

Пример:

```
>> a=[3 9;4 12]
```

```

a =
    3    9
    4   12
>> b=[1;2];
>> det(a)      % Как видно главный определитель системы
равен 0
ans =
    0

```

Получим псевдорешение системы :

```

>> x=pinv(a)*b
x =
    0.0440
    0.1320

```

В общем виде систему нелинейных уравнений можно записать в виде $F(x)=0$, где x – вектор неизвестных системы. Для решения систем нелинейных уравнений предназначена функция `fsolve`. Алгоритм ее работы использует начальное приближение x_0 и базируется на минимизации суммы квадратов компонент функции F методами Гаусса – Ньютона и Левенберга – Марквардта. В простейшем случае обращение к функции `fsolve` имеет вид :

```
x=fsolve(F, x0);
```

Функция F должна быть оформлена в виде отдельной подпрограммы. Например, для системы :

$$\begin{cases} x_1 + x_2 - \sin(\pi x_1) = 0 \\ x_1 - x_2 - \cos(\pi x_2) = 0 \end{cases}$$

Подпрограмма должна быть оформлена следующим образом:

```

function y=Sistema(x)
y=[x(1)+x(2) - sin(pi*x(1)); x(1) - x(2) - cos(pi*x(2))];
Обращение к функции fsolve будет выглядеть следующим
образом:
>> x0=[1 2];
>> x=fsolve(@Sistema,x0)

```

x =
0.3915 0.5510

Поиск минимума функции одной переменной

Для поиска минимума функции $f(x)$ на отрезке $[a, b]$ можно воспользоваться функцией `fminbnd` :

>> `x=fminbnd(F, a, b)`;

Если минимизируемая функция на заданном отрезке многоэкстремальна, то `fminbnd` найдет какой либо один минимум и не выдаст никакой информации о других. Поэтому рекомендуется перед использованием `fminbnd` построить график.

В заключение теоретического введения отметим, что все рассмотренные выше функции предлагаемые системой MATLAB допускают и другие формы обращения к ним (нами приведены простейшие формы), позволяющие получить более полную информацию о процессе вычислений. Для получения сведений о возможных формах обращения к функциям системы можно получить используя команду `help` :

>> `help Fun`

`Fun` – имя интересующей функции (например `help fsolve`).

ПРАКТИЧЕСКАЯ ЧАСТЬ

1. При заданном значении давления зависимость плотности вещества от температуры задана таблично (табл. 1):

Таблица 1

T, К	20	40	70	90	100	150	200	250
ρ , кг/м ³	80	63,4	35,3	26	23	15,2	11,4	9,2

2. Определить значения плотности при температуре $T_0 = 30 \ 50 \ 60 \ 80 \ 93 \ 135 \ 178 \ 223 \text{ К}$.

Получить плотность при указанных температурах можно воспользовавшись интерполяцией исходной табличной функции $\rho(T)$.

Интерполяцию осуществим разными методами – кубическими сплайнами `spline` и линейно (Linear):

```
>> T=[20 40 70 90 100 150 200 250];  
>> T0=[30 50 60 80 93 135 178 223];  
>> Ro=[80 63.4 35.3 26 23 15.2 11.4 9.2];
```

Определим значения плотности при температуре T_0 с помощью интерполяции сплайнами :

```
>> RoSpline=spline(T, Ro);  
>> RoTs = ppval(T0, RoSpline);    % значения плотности при  
темп. T0
```

Построим график `RoSpline(T)` (рис.4) :

```
>> t=20:0.5:250;  
>> plot(t, ppval(t, RoSpline), 'k');
```

Значения плотности с помощью линейной интерполяции получим следующим образом :

```
>> RoTL=interp1(T, Ro, T0);
```

Построим график (рис. 4) :

```
>> hold on;  
>> plot(t, interp1(T, Ro, t), 'k--');  
>> title(' Spline & Linear ');  
>> legend(' Spline ', ' Linear ');  
>> xlabel(' T ');  
>> ylabel(' Ro ');
```

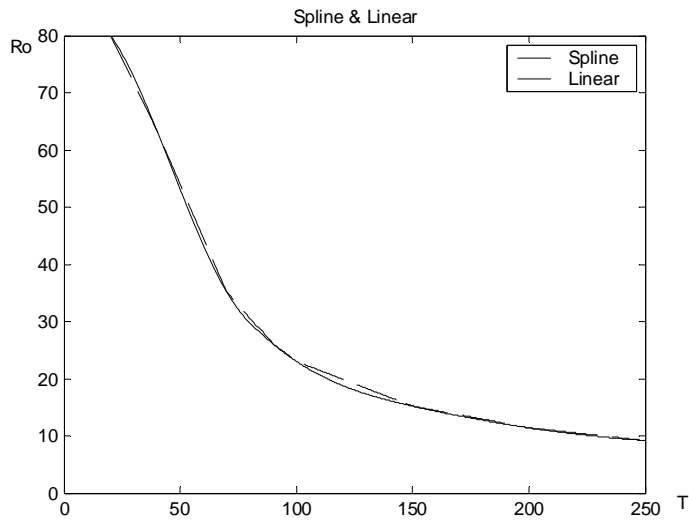


Рис. 4

Как видно разный способ интерполяции дает разный результат вне табличных узлов. Оценим среднеквадратические ошибки использованных методов интерполяции если предположить, что были проведены дополнительные эксперименты из которых определено (Табл. 2):

Таблица 2

T, К	30	50	60	80
ρ , кг/м ³	72,7	52,8	42,9	30

```
>> Td = [30 50 60 80];
```

```
>> Rod = [72,7 52.8 42.9 30].
```

Определим значение плотности при температуре Td :

```
>> RoTds = ppval(Td, RoSpline); % кубическими  
сплайнами
```

```
>> RoTdL= interp1(T, Ro, Td); % линейная  
интерполяция
```

Оценим погрешность работы примененных интерполяционных методов :

```
>> ZSp = 100*sqrt((RoTds- Rod).^2)/Rod;  
>> ZLin= 100*sqrt((RoTdL- Rod).^2)/Rod;
```

1. Определить минимум функции

$$y(x) = \cos\left(\pi \cdot \frac{x}{5}\right) - \frac{\sin(\pi x)}{x}$$

на отрезке $x = [1, 5]$, и получить решение уравнения $y(x)=0$.

Построим график функции $y(x)$ на заданном отрезке (рис.5) :

```
>> x=1:0.1:5  
>> plot(x, cos(pi*x./5)-sin(pi*x)./x);  
>> title(' y=cos(pi*x/5)-sin(pi*x)/x ');  
>> xlabel(' x ');  
>> ylabel(' y ');
```

Как видно из рис. 5 функция $y(x)$ имеет на заданном отрезке два локальных минимума : первый на отрезке $[2,3]$, второй – $[4,5]$. Для их определения воспользуемся функцией `fminbnd` :

```
>> x(1)=fminbnd(' cos(pi*x./5)-sin(pi*x)./x ',2,3);  
>> x(2)=fminbnd(' cos(pi*x./5)-sin(pi*x)./x ',4,5);
```

Для решения уравнения $y(x)=0$ воспользуемся функцией `fzero` :

Как видно из графика, решение находится в окрестности точки $x=2$

```
>> x=fzero(' cos(pi*x./5)-sin(pi*x)./x ', 2);
```

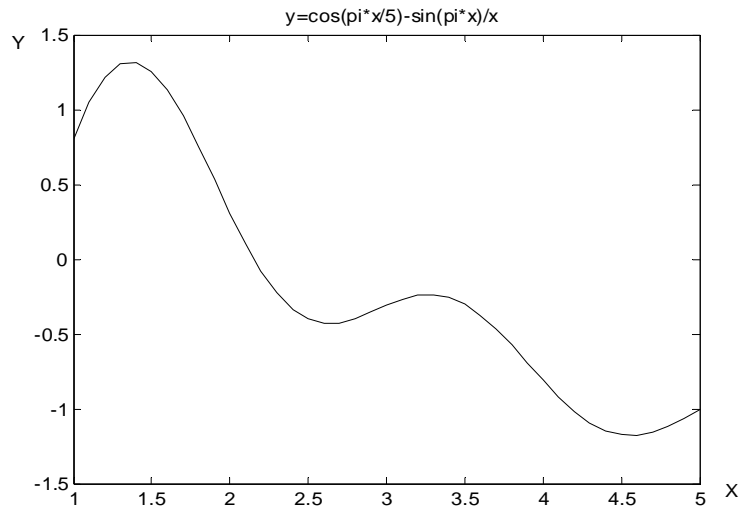


Рис. 5

ОТЧЕТ ПО РАБОТЕ

Отчет по работе должен содержать сведения о функциях предоставляемых системой MATLAB для решения задач интерполяции, решения нелинейных уравнений и систем, минимизации функции одной переменной.

В результате выполнения практической части необходимо получить:

1. График рис. 4., значения плотностей при температуре T_0 и величины погрешностей (Z_{Sp} и Z_{Lin}) реализации разных методов интерполяции.
2. График рис.5, значения минимумов ($x(1)$ и $x(2)$) и корень уравнения $y(x)=0$.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какими способами система MATLAB позволяет проводить интерполяцию данных?

2. Каким образом можно определить значение функции в любой точке интервала (x_1, x_2) если она задана на этом интервале дискретно (таблицей)?
3. Как вычисляется значение функции по имеющейся кусочно – полиномиальной форме?
4. Решение системы линейных уравнений можно получить следующим образом:
 $\gg x=A\backslash b$ или $\gg x=\text{pinv}(A)*b$
В чем состоит особенность использования данных методов?
5. С помощью какой функции осуществляется поиск корней нелинейных уравнений?
6. Какая функция предназначена для решения систем нелинейных уравнений?
7. Что такое указатель на функцию? Для чего он нужен?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кетков Ю.Л. Matlab: программирование численных методов / Ю.Л. Кетков, АЮ. Кетков, Н.Н. Шульц. СПб.: БХВ – Петербург, 2004.- 672 с.

СОДЕРЖАНИЕ

Введение	3
1. Лабораторная работа № 1	
СРЕДА MATLAB. ЭЛЕМЕНТАРНЫЕ ВЫЧИСЛЕНИЯ	3
2. Лабораторная работа № 2	
ВИЗУАЛИЗАЦИЯ В MATLAB	12
3. Лабораторная работа № 3	
ОСНОВЫ ПРОГРАММИРОВАНИЯ В MATLAB	24
4. Лабораторная работа № 4	
ИНТЕРПОЛЯЦИЯ. ЛИНЕЙНЫЕ, НЕЛИНЕЙНЫЕ УРАВНЕНИЯ И СИСТЕМЫ. МИНИМИЗАЦИЯ ФУНКЦИИ	36
Библиографический список	48

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для проведения лабораторных занятий по дисциплине
«Методы математического моделирования»
для студентов специальности 160700.65,
24.05.02 «Проектирование авиационных и
ракетных двигателей» очной формы обучения

Составители: Гуртовой Андрей Александрович
Демьяненко Юрий Васильевич
Кретинин Александр Валентинович
Сушков Алексей Михайлович

В авторской редакции

ФГБОУ ВПО «Воронежский государственный технический
университет»
394026 Воронеж, Московский пр., 14