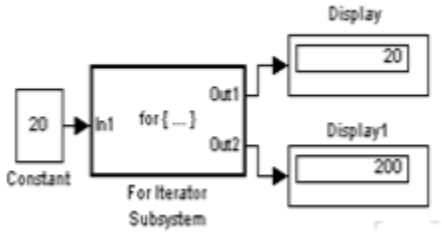
Антоанета Иванова Хинова

Някои практически решения в Matlab

Ръководство за лабораторни упражнения



Университетско издателство
„Васил Априлов“
Габрово 2021

Анотация

Настоящото ръководство е сборник от решени примери и задачи. Теоретичните обяснения са сведени до минимум. Целта е с помощта на преподавател или самостоятелно, обучаемият да придобие практически умения за работа в програмната среда на Matlab. Примерите са съобразени с програмата за обучението на професионален бакалавър по дисциплините „Автоматизация на проектирането” в специалността „Компютърни системи и технологии” и „Автоматизирано конструиране и проектиране“ на специалност „ Електротехника”.

Ръководството може да се ползва от всички, които желаят да се упражняват и да придобият увереност за работа в този вид програмна среда.

д-р инж. Антоанета Иванова Хинова, автор

Някои практически решения в MATLAB

Ръководство за лабораторни упражнения

Университетско издателство „Васил Априлов“ –Габрово, 2021

ISBN 978-954-683-657-1

ПРЕДГОВОР

Някои практически решения в MATLAB

В пособието се описва система от решения за научно-технически разчети в MATLAB. Описват се най-използваните команди, типове данни, научна графика, програмиране на функции, основни типове числени методи.

Ръководството е предназначено за студенти в технически специалности, може да се използва от докторанти, преподаватели и научни работници.

Историята на Matlab датира повече от три десетилетия. Името е съкращение от “Matrix Laboratory”. Класическият вариант на продукта е написан от Клиф Моулер в университета в Ню Мексико през 1977г. Той е представлявал интерактивна лаборатория, позволяваща да се ползват подпрограми от пакетите LINPACK и EISPACK. През 1984г. К. Моулер, С. Бангерт и Дж. Литл са основали фирмата MathWorks. Сега MATLAB представлява мощен математически пакет със свой програмен език, гъвкави графични възможности, средства за съчетаване с други програмни езици и няколко десетки прложни пакети.

В пособието се описват основните команди в MATLAB, научна графика, типовете данни, програмиране на функции, основни типове числени методи. Съдържанието е съобразено с най-новата версия на MATLAB, за която университетът има лиценз, но е съвместимо и с версиите MATLAB 8.0 (R2013b), MATLAB 7.0.

Ръководството не е самоучител, ползва се от обучаемите с преподавател. За по-нататъшно изучаване на MATLAB може да се използват книгите, посочени в библиографията.

Съдържание

1. Основни команди

1.1. Кратко въведение в Matlab.

След пускане на MATLAB на екрана се появява основния прозорец, съдържащ няколко подпрозореца. COMMAND window- това е командния прозорец, в който се набират команди, а MATLAB представя резултати. Графичните резултати се извеждат в един или в няколко графични прозореца. Командите на ползвателя се въвеждат след промпта:

```
>>
```

Например:

```
>> (76 + 21¡ 85)*3/4
```

Въвеждането завършва с клавиша ENTER, а MATLAB отговаря:

```
ans =
9
>> (1+sqrt(5))/2
```

Отговор:

```
ans =
1.6180
```

В този пример sqrt е функция за намиране на квадратен корен ; ans е специална променлива,в която винаги се записва резултата от последната команда. Тази променлива може да се използва в следващата команда. Например, ползвателят може да създаде променливата Z и да я използва в следващата команда.

```
>> e=2 + 1/2 + 1/6 + 1/24 + 1/120 + 1/720
e =
2.7181
```

Функцията exp изчислява командата e^x. Записът ¡2.2627e¡4 е

$$¡2.2627e¡4 = ¡2:2627 ¢ 10^{¡4} = ¡0:00022627.$$

Преди да се използва дадена променлива, тя трябва да се инициализира, т.е. да й се присвои някаква стойност, иначе ще се получи съобщение за грешка. Например, ако в следния израз p и q са недефинирани, се получава съобщение за грешка:

>> x = ¡p + sqrt(p^2 ¡ q)

Въвеждането на всяка команда завършва с ENTER. В резултат се получава exo (отклик). Ако се запише знак „ ; „(точка и запетая), отклик не се получава. Например е израза

>> e = 2 + 1/2 + 1/6 + 1/24 + 1/120 + 1/720;

На променливата *e* ще бъде присвоена стойността 2.7181 и веднага ще се появи нов промпт:

>>

По –нататък в примерите промт няма да се печата. На един ред могат да се записват няколко команди. Отделят се със „ ,“ (запетая), или с „ ; “ (точка и запетая).

1.2. Променливи.

Име на променлива може да бъде всяка последователност от букви, цифри и знаци за подчертаване, като се чете до 31-я знак. Регистърът е важен, например Err и err са различни променливи.

На машините, поддържащи IEEE-аритметика, максимално представимото реално число се пази в променливата realmax, а минималното положително нормализовано реално число - в променливата realmin. И така, в IEEE аритметиката:

$eps = 2:2204 ¢ 10^{i16};$

$realmax = 1:7977 ¢ 10^{308};$

$realmin = 2:2251 ¢ 10^{i308};$

„Нечисло“ се обозначава с „NaN”. Променливите esp, realmin, realmax, Inf, NaN, pi, i, j могат да се появят в дясната част на равенството. След това те губят своето първоначално значение. Например, променливите i, j, често се използват като броячи в цикли. Тогава 1+2*i, вече не е комплексното число 1+2i. Връщането към първоначалното значение на предопределените константи може да стане с командата clear.

Например, clear i може да върне значението на променливата i като имагинерна единица. Списъкът на всички вградени елементарни математически функции може да се получи чрез командата “ help elfun”. Ето най-често употребяваните:

sin, cos, tan - тригонометрични функции ;
acos, asin - обратни тригонометрически функции;
exp, log - експонента и натурален логаритъм;
sqrt - квадратен корен;

round - закръгление до най- близкото цяло число;
fix - закръгление с отхвърляне на дробната част;
abs - модул на реално или комплексно число;
real, imag - реална и имагинерна част на комплексно число;
conj - комплексно спрегнато число.

Променливите участват в основните функции за работа с матрици.

Основен обект в системата на MATLAB-това са матриците или масивите. Дори скаларните величини се разглеждат като матрица с размерност 1x1.

Вектор (едномерен масив) представлява матрица–ред с размерност 1 x n, или матрица-стълб с размерност m x 1. За да се зададе вектор-ред, е достатъчно да се напишат неговите елементи в квадратни скоби, разделени със запетая или интервал. Елементите на вектор-стълб се разделят със символ „ ; “ или с клавиша ENTER. Например, командата

p = [1 2 3 4], задава вектор-ред p = (1*;* 2*;* 3*;* 4), а команда

d = [1; 2; 3; 4], задава вектор-стълб: d = 1
2
3
4 .

Вектори, представляващи последователни членове на аритметична прогресия, лесно се задават с помощта на командата „ : “. Командата p:h:d определя вектора (p; p+h; p+2h; :::::; d). Командата p:d определя числова редица като част от аритметична прогресия със стъпка 1.

За да се зададе матрица (двумерен масив), е достатъчно да се зададат елементите й по редове, разделени с точка и запетая. Например командата

A = [1 2;3 4]

определя матрицата

A= 1 2
3 4.

Да се изпълнят следните команди

a= 123;456;457

b= 1 2 9;1 3 9;1 4 8

s= zeros(3,5)

a==b

W= 1 2 3;4 5 6;4 5 7

S=ones(2,4)

W=eve(3,5)

Матриците, с които се извършват операции, трябва да бъдат с еднаква размерност, в противен случай MATLAB извежда съобщение за грешка.

Специалните матрици се извеждат със следните функции:

zeros(m.n)-нулева матрица;

ones(m,n)-матрица, състояща се само от единици;

eye(m,n)-матрица с единици по главния диагонал и нули извън диагонала;

rand(m,n)-матрица със случайни елементи, намиращи се в интервала 0.1);

randn(m,n)- матрица с елементи, разпределени по нормален закон с математическо очакване, равно на 0 и средноквадравично отклонение, равно на 1.

В горните записи m и n определят размерността на матриците, като съответно m е броя на редовете, а n - броя на стълбовете. Всяка от тези функции може да се извика с един параметър. В този случай се генерира квадратна матрица от указания порядък със съответстващото свойство.

1.3. Основни действия с матрици

Към матриците са приложими всички стандартни математически функции: те се прилагат покомпонентно към всеки елемент. За разлика от тези функции, операциите, обозначени със символи +, /, се изпълняват като матрични операции.

1.3.1. Списък на основните математически операции

a+*b* събиране на скалари, вектори или матрици;

a ¡ *b* изваждане на скалари, вектори или матрици;

*a***b* умножение на скалари; матрично умножение;

a.**b* поелементно умножение на елементите на матриците;

a^*b* повдигане на скалар или матрица на степен;

a.^*b* повдигане на всеки елемент на матрицата в степен;

a/*b* деление на скалари; деление на матрици отдясно;

a./*b* поелементно деление на матрици;

*a*n*b* ляво деление на матрици;

a’ транспониране на матрици.

Ако размерите на операндите (матрици, участващи в операциите) не са съгласувани, програмната система формира съобщение за грешка. Номерацията на страниците и стълбовете в матриците започва с 1. За да се получи достъп до елементите на матрица a, от i-тия ред, j-тия стълб, е достатъчно да се въведе a (i, j).

За да се получи достъп до к-тия компонент на вектора a, се въвежда a(k). Командата a(к) работи и за матрици: Matlab търси к-тия елемент на матрицата a, предполагайки, че елементите се номерират по стълбове.
Какво ще се получи, ако в командата има препратка към несъществуващ елемент на матрица? Например, ако матрицата A има размери 2 £ 2, а е зададено обръщение към елемента A(3, 5). Всичко зависи от това, в коя част от знака за присвояване е разположена препратката към елемент A(3, 5). Ако A(3, 5) се намира в израза вдясно от знака за присвояване, ще се получи съобщение за грешка. Ако A(3, 5) стои отляво на равенството, то размерите на матриците автоматично се определят до 3x5, като на новите елементи на матриците се присвояват нулеви значения, а елемента A(3, 5) е с пресметната стойност.

За достъп до последния стълб или последния ред може да се използва командата end. Например, A(end,k)- това е елемент на матрицата A, стоящ в последния ред на к-тия стълб. За достъп до всички страници или стълове на матрицата се използва командата „:“(двоеточие). Например, A (:, k)-това е к-тия стълб, а A(k, :)-k-тия ред на матрицата A. Подобни изрази могат да се записват и в лявата част на равенството. Например командата

A(k, :)=3

присвоява на всички елементи на к-тия ред стойност 3.

Резултатът от операцията A(:) се явява дълъг стълб, съставен от стълбовете на матрицата A. Функциите max(a) и min (a) възвръщат максималния и съответно минималния елементи на вектора a. Ако освен най-голямата и най-малката стойност на b ни е нужен и индекса i, трябва да се извика функция с два входни параметъра:

[b,i]=max(a),
[b,i]=min(a).

Ако максималните (минималните) стойности са няколко, то се възвръща номера на първия от тях.

Ако A е матрица, то max(A), min(A) сформират вектор-стълб b, j-тият елемент на който е равен на максималния и съответно на минималния елемент в j-тия стълб на матрицата A.

Функциите

[b,i]=max(A),

[b,i]=min(A),

освен b възвръщат също ред i, в j-тата позиция на който е записан индекса на максималния (съответно на минималния) елемент в j-я стълб. За намиране на максимума сред всички елементи на матрицата A може да се използват max(max(A)) или max(A(:)).

Функцията size (A) възвръща двукомпонентен масив, съдържащ броя на редовете и на стълбовете на матрицата A. Функцията size (A,1) възвръща броя на редовете, а функцията size (A,2)-броя на стълбовете; lendth(A) връща максимума на броя на редовете и на стълбовете.

В системата MATLAB лесно се реализират блокови операции с матрици. Така, командите

C = [A B],
C = [A;B],

формират блокова матрица C=(A B), като размерите на матриците A и B трябва да бъдат съгласувани.

Нека c, p са вектори, в които са записани номерата на някои редове и стълбове, съответно на матрицата A. Тогава A (c,p) е матрица, състояща се от елементите на изходната матрица, стоящи на пресечната точка на редовете с номера c и стълбове с номера p.

Системата Matlab поддържа работа с празни матрици, т.е. с матрици в които броят на редовете или/и броят на стълбовете е равен на нула. Един от начините да се зададе такава матрица е чрез функцията zeros.
Например:

A=zeros(0,5).

Да се определи матрица с размер 0 x 0 е възможно с помощта на операцията [ ]. Операцията [ ] помага също при отделяне на редовете или стълбовете на матриците. Командата

A(i,:)=[ ]

отделя *i*-тия ред, а командата

A(:, j)=[ ]

отделя j-я стълб на матрицата A.

В системата MATLAB има удобни средства за работа с диагоналите на матриците. Нека d е вектор от n компонента. Командата

A=diag(d, k)

възвръща квадратната матрица A от ред n + jkj с елементи d на k-я диагонал; k = 0 съответства на главния диагонал на матрицата, k>0 –на k-я наддиагонал, k<0- на jkj-я поддиагонал. Ако аргументът k не е указан, то d се разполага на главния диагонал на матрицата *A*. Командата

d=diag(*A*,*k*)

връща вектор-стълб d, съдържащ елементите на k-я диагонал на матрицата A. Ако аргументът k не е указан, то се възвръща главния диагонал на матрицата A.

В системата MATLAB има следните (бинарни) логически операции: «по-малко» <, «по-голямо» >, «не по-голямо» <=, «не по-малко»>=, «равно» ==, «неравно» ~=. Те изпълняват поелементно сравнение на два масива с еднакви размери и връщат така наречения логически масив със същия размер. Неговите елементи са равни на 1 или 0 в зависимост от това, истинно ли е или не разглежданото отношение за съответстващата двойка елементи от два масива. Един от аргументите може да бъде скалар. В този случай, както и за аритметичните операции, вместо него ще бъде разгледана матрица, запълнена с този скалар.

Логическите операции - това са двоичните операции «и» &, «или» *j* и унарната операция «не» ~. Аргументите на тези операции излизат като логическите масиви, но могат да бъдат и обикновени числови масиви. В този случай нулевата стойност се интерпретира като лъжа, а единицата – като истина.

A & B е логически масив, елементите на който са равни на 1 на тези позиции, на които два съответстващи елемента в A и B имат ненулеви стойности, и равни на 0 на останалите позиции.

A j B (A "или" B*)* е логически масив, елементите на който са равни на 1 на тези позиции, на които в крайна сметка един от съответстващите елементи в A и B има ненулева стойност, ~A-това е логически масив, елементите на който са равни на 1 на тези позиции, където съответстващите елементи на масива A са ненулеви, и равни на 0 в противен случай.

Матриците A и B трябва да имат еднакви размери или един от аргументите да е скалар.

Операциите-отношения и логическите операции често се използват заедно с функциите find, any, all.

Функцията find връща индекси и стойности на ненулевите елементи. Функцията има следния синтаксис:

[i,j,v]=find(A)

където i, j , v са вектори, съдържащи съответно номера на реда, стълба и стойностите на ненулевите елементи на матрицата A.

Да разгледаме следния пример:

Нека f е вектор, съдържащ протабулирани стойности на някои функции в точки, намиращи се в x.Търсят се корените на функцията. Това може да се направи по следния начин:

```
n=length(x);
w=1:n ¡ 1;
x (find(f(w) .* f(w+1)< 0jf(w)==0))
```

Ако *a* е вектор, то функцията all(a) връща 1 , когато всички елементи на масива *a* не са равни на нула, и 0 в противен случай. Ако A е матрица, то all(A) изпълнява функцията all за всеки стълб на матрицата *A* и връща получените стойности във вектор-стълб.

Ако *a* е вектор, то функцията any(a) връша 1, когато между елементите на масива a има ненулев, и 0 в противен случай. Ако A е матрица, то any(A) изпълнява функцията any за всеки стълб на матрицата A и връща получените стойности във вектор-стълб. Нека например, ако

```
A =  0 0 1 1
     1 0 1 0
```

Тогава any(A) връща вектор (0; 1; 1; 1), а all(A) —вектор (0; 0; 0; 1).

1.3.2. Символни масиви.

MATLAB позволява да се работи с редовете на текста. За да се определи такъв ред, е достатъчно символният ред да се затвори в кавички. Например,

```
s= 'Isaac Newton'
```

Редовите значения се разглеждат от системата като масиви. В горния пример s е вектор-ред от 12 елемента-символи (никакъв завърващ нулев символ няма).

Символите могат да се обединят в двумерни масиви. Това позволява да се съхранява набор от редови стойности с еднаква дължина, Например, след въвеждане на командата

```
S = ['Isaac Newton' 'Blaise Pascal'] ,
```

MATLAB създава следния двумерен масив (2 x13):

```
S ='I ' ' s' ' a ' ' a ' ' c ' ' ' ' 'N' 'e' ' w ' ' t ' 'o' 'n' ' '
```

‘B’ ‘ l ’ ‘ a ’ ‘ i ‘ ‘ s ’ ‘ e ’ ‘ ‘ ‘P’ ‘ a ’ ‘ s ’ ‘ c’ ‘a’ ‘ l ‘

В дадения пример специално се добавя към името Нютон един спейс (празна позиция), за да се изравни броят на елементите във всеки стълб на масива. Ако това не се направи, MATLAB дава съобщение за грешка. Командата *S*(1*;* :) в дадения пример връща реда ‘Isaac Newton ’, а командата *S*(2*;* :) – реда ‘Blaise Pascal’. Въобще, към символните масиви са приложими повечето разгледани дотук команди. Достъпни са блоковите операции, командите на отделяне на подмасиви и др. В примера:

```
a='Matrix';
b='Laboratory';
c=[a(1:3)  b(1:3)]
```

на променливата с ще бъде присвоено символно значение ‘MatLab’.

Вътре в системата MATLAB символите са представени във формат UNICODE. За съхранение на един символ се използват 2 байта.

1.4. Форматиран изход.

MATLAB предоставя широки възможности за управление на формата на изхода на числовите стойности в командния прозорец. По подразбиране се използва формат short. В него се прилагат следните правила:

а)Ако всички *е*лементи на масива са цели числа с не повече от 9 цифри всяко число, то те се извеждат така, както са.

б)Ако всички елементи на масива по абсолютна стойност са по-малки от 1000 и не по-малки от 0*:*0001, то те се извеждат такива, каквито са.

В останалите случаи MATLAB извежда елемент на масива с използване на общ множител (изключение правят скаларните величини). Елементите се извеждат във формат §d.ddddesppp, където d са цифрите на мантисата, p са иифрите на показателя (порядъка). Ако числото не е нула, то старшата цифра пред десетичната точка не е равна на нула. Ако числото е отрицателно, то отпред се поставя знак ¡ . Например, след въвеждане на командата:

```
1/7
```

MATLAB печата:

```
0.1429.
```

Форматът на изхода може да се управлява или с помощта на пункта от меню FILE | PREFERENCES | COMMAND WINDOW | NUMERIC FORMAT или с помощта на командата format с параметри. Действието на тази команда се разпространява за всички последващи извеждания, докато не бъде въведена нова команда format с други параметри. Да приведем списъка на някои от възможните параметри в Таблица 1.

Таблица 1

Формат	Описание	Пример
format	Също като short; установен по подразбиране	3.1416
format short	Формат за представяне на числа с фиксирана запетая: 5 значещи цифри	3.1416
format long	Формат за представяне на числа с фиксиран запетая: 15 значещи цифри	3.1415926535 8979
format short e	Формат за представяне на числа с плаваща запетая: 5 значещи цифри	3.1416e+000
format long e	Формат за представяне на числа с плаваща запетая: 15 значещи цифри	3.1415926535 89793e+000
format rat	Апроксимация на числа на рационална дроб	355/113

Да подчертаем, че командата format влияе само на извода на данните, а не на това как се съхраняват те. Това се отнася и към формата rat:. Винаги, когато е нужно да се изведе матрицата X, MATLAB намира към всеки неин елемент x_{ij} най-доброто рационално приближение $p_{ij}=q_{ij}$, такова, че $p_{ij}=q_{ij}$ ¡ $x_{ij}j$; tol ¢ $jx_{ij}j$; където tol = 0^{i6} : i; jji$x_{ij}j$.

1.5. Справка и документация.

Системата MATLAB предоставя на ползвателя удобен справочен навигатор с развита система за намиране на необходимата информация. Навигаторът е достъпен чрез пункта от меню HELP|MATLAB HELP. Справката обхваща всички раздели, ядрото на MATLAB и разширителните пакети, включва програми-примери, препратки към демонстрационни приложения. Бърз начин да се открие нужния раздел за справка, касаещ н име на функцията якои функции, е да се набере в командния прозорец:

doc име на функцията

Тази информация, но непосредствено в командния ред, може да се получи чрез командата

Help име на функцията

Например, ако се набере

help inv

се получава справка по функции inv:

INV Matrix inverse. INV(X) is the inverse of the square matrix X. A warning message …
See also SLASH, PINV …
Overloaded methods …

За отбелязване е, че в справката на командния прозорец всички функции се набират на горен регистър, за да се разграничат от текста. Реално, в действителност имената на всички вградени функции в системата са в
нисък регистър.

1.6. Средата на Matlab

1.6.1. Работно пространство на командния прозорец.

Стойността на създаваните в командния прозорец променливи се съхранява в отделна област на паметта, която се нарича работно пространство на командния прозорец или основно работно пространство. Покрай основното работно пространство Matlab създава и в нужното време отделя работни пространства на извикваните функции. В тези пространства се разполагат вътрешните променливи на функциите.Имената на всички променливи, намиращи се в момента в основното работно пространство се изобразяват в подпрозореца WORKSPACE. Списъкът от имена на променливи може също да се получи чрез командата who, а списъкът на променливите с техните стойности се получава чрез командата whos. Командата

clear списък променливи

изключва (изтрива) указаните променливи от работното пространство.
При това техните стойности се губят. Командата clear all, изтрива всички променливи от работното пространство. Да разгледаме следния пример:

```
A=rand(100);
B=rand(100);
C=A*B;
who
```

MATLAB отпечатва:

```
Your variables are:
A   B   C
```

Набира се:

clear A B whos

Получава се

Nam e
C:
Size
100x100 Bytes Class
80000 double array
Grand total is 10000 elements using 80000 bytes

1.7. Съхраняване и въвеждане на променливи

След излизане от системата Matlab всички променливи се унищожават и не са достъпни в следващия работен сеанс. Да се запишат на диск всички променливи от работното пространство е възможно с командата:

save име на файла.

При това всички променливи се съхраняват в двоичен код в указания файл.

По подразбиране разширението на този файл е - mat, затова файлове с такъв формат се наричат mat-файлове. Не се препоръчва за mat-файлове да се използват други разширения. За да се запаметят не всички, а само част от променливите от работното пространство, се използва командата:

save име на файла списък променливи,

като имената на променливите се отделят едно от друго чрез празни позиции. Прочитането на mat-файл става с командата:

load име на файла.

която зарежда всички променливи от файла, или с помощта на файла

load име на файла списък променливи,

която зарежда от файла само указаните променливи.
Например:

x=[0;1;2;3;4];
W=[x.^0, x.^1, x.^2,x.^3]; save Wndrmd x W

clear all
load Wndrmd who

MATLAB печата:

W x

С помощта на командата save името на променливата ¡asci може да се съхрани в едноименен текстови файл. В дадения файл в текстов формат поредово ще се записват елементите на матрицата. Такива файлове могат да се създават във всеки текстови редактор. Независимо от това, как е бил създаден такъв файл, той може да се прочете с командата:

load име на файла ¡asci

Стойностите от файла ще бъдат присвоени на променлива, името на която съвпада с името на файла (без разширение).

1.7.1. Команди dir, type, delete, cd

На контролния панел в основния прозорец на MATLAB е разположено падащо меню CURRENT DIRECTORY, в което може да се избере текущата папка. Командата

dir

извежда на екран списък файлове от текущата папка. Командата

type име на файла,

извежда на екран разпечатка на указанния файл от текущата папка. Командата

delete име на файла,

изтрива указания файл. Командата

cd нов каталог,

изменя текущата папка.
Същото действие може да се извърщи, ако се използва падащото меню в подпрозореца CURRENT DIRECTORY или менюто от командния панел.

1.7.2. Дневник на работата

За да се запише във файл всичко, изобразено в командния прозорец, зададените команди и отговорите на системата, може да се използва командата diary. Командата

diary име на файла

открива дневник, т.е. указва на системата, че всичко, което се появява след тази команда на екрана до следващата команда diary ще бъде записано в упоменатия текстови файл. Прекъсва записа в дневника команда за откриване на нов дневник или командата

diary off.

1.7.3. Стартиране на външни програми

За да се стартира за изпълнение коя да е команда на операционната система или произволна външна програма, намираща се в текущата папка, може да се набере

! име на програмата

1.8. Сценарий

Сценарий, или просто скрипт, се нарича последователност от команди на системата MATLAB, записани в текстов файл. Файлът трябва да има да има разширение m. Командите се отделят една от друга, както и в командния прозорец, чрез символи «,», «;» или чрез символите на преминаване на нов ред. Коментарите започват със символа % и продължават плътно до края на реда. Група от три поредни точки «...», поставени в края на реда, означава, че текущата команда продължава на следващата страница.

За да се изпълнят командите, записанни в програмата–сценарий, е достатъчно името на файла (без разширение m) да се набере в командния прозорец. Сценарият може също да се извика от други програми–сценарии.

2. Двумерна графика

Целият графичен извод в системата MATLAB постъпва в един или в няколко графични прозореца.

2.1. Функцията plot

Ако *x* и *y* са два вектора с еднаква дължина, то функцията

plot(*x*, *y*)

в графичния прозорец строи крива по точки с абсциси, записани в x, и ординати, записани в y. Мащабът по двете оси се избира автоматично,

така, че кривата да се побира в графичния прозорец. Ако до изпълнение на тази команда ни един графичен прозорец не е отворен, то такъв се отваря автоматично. В противен случай изходът ще се появи в последния (текущ) графичен прозорец и по подразбиране ще се изтрие старото изображение. Пример:

```
x=0:pi/100:2*pi; %задава се интервала на изменение на аргумента
y=sin(x);% задава се функцията
plot(x,y) % команда за изчертаване
```

Функциите xlabel, ylabel добавят надписи съответно към абсцисната и ординатната оси, а title определя заглавието.Тези функции се прилагат в следния пример:

```
xlabel('x=0:2\pi') ylabel('Sine of x')
title('Plot of the Sine Function')
```

Резултатът е онагледен на фиг.1. Функцията plot, както повечето функции в MATLAB, може да се използва с различен брой аргументи:

plot(x, y1,'str1', 'param ', value of par, ...),

двойките или тройките аргументи могат да бъдат следвани от двойки опции (parameter/values parts), задаващи допълнителни свойства на линиите и маркерите:

'LineWidth',n- дебелина на линията, н-цяло число,

'MarkerFaceColor','symbol' –цвят на маркера. Стрингът 'symbol' е един от синмволите за задаване на цвят;

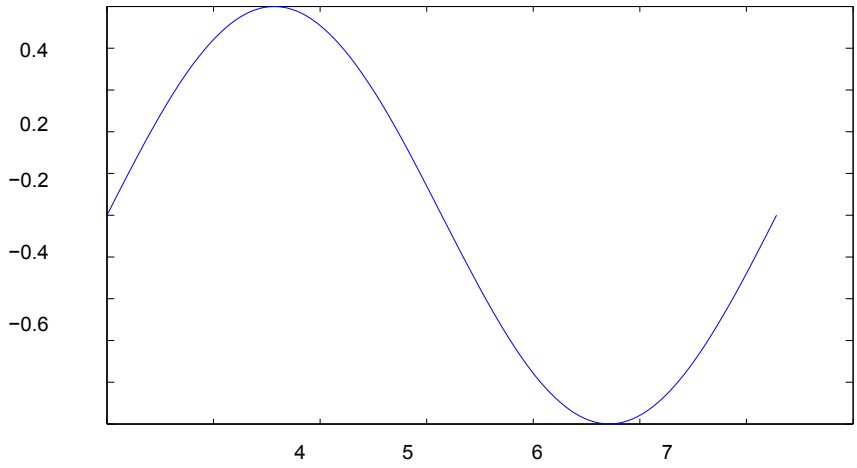
'MarkerColor', 'symbol' – цвят за запълване на маркера, ако е затворена фигура;

plot(x) или *plot(y)*-възможно е приложението на командата с един аргумент.

Може да се построи графика едновременно на три функции:

```
y1=sin(x);
y2=cos(x);
y3=sin(x)/x
plot(x,y1,x,y2,x,y3
```

При построяване на графиката на фиг.1 е избрана стъпка на аргумента x, равна на 0,1, за построяване на по-плавна графика. Функцията plot чертае само зададените точки като елементи на вектора x, а после те биват съединени с отсечки по метода на линейната интерполация. Следователно при по-малка стъпка, формата на графиката на функцията е по-плавна и по-точна.



Фиг.1. Графика на функцията sin x

2.1.1. Няколко криви на една графика

Всяка нова функция plot изтрива старото изображение. За начертаване на няколко графики се използва командата hold on («замразяване»), включваща режим на съхранение на предишния графичен резултат. Например, построяване на графиките на три функции в една координатна система е показано на фиг.2. Излизането от ражим hold on става с командата hold off.

Друг начин да се изведат няколко графики в един прозорец е чрез прилагане на функцията:

plot(*x1*, *y1*, … , *xn*, *yn*)

с няколко двойки параметри *x*, *y*. Нека построим тези графики:

plot(*x*, *y1*, *x*, *y2*, *x*, *y3*)

За разлика от предходния пример, графиките са изчертани с различни цветове. Редуването на цветовете е регламентирано с правила.

Командата legend добавя легенда - пояснение към графиката. При начертани три графики, броят на параметрите на функцията legend трябва да бъде също три:

legend('sin(x)', 'exp(-x^2)', '0.5 atan(x)')

Легендата ще се появи в десния горен ъгъл на координатната система. В дадения случай това не е удачно, защото се закрива част от графиката. За да се измени положението на легендата, тя може да се премести с мишката или да се използва функцията legend с още един параметър:

legend('sin(x)', 'exp(-x^2)', '0.5 atan(x)', 2)

Ще се получи изображението на фиг.2. Този параметър може да заема стойности от 0 до 5. Числата от 1 до 4 съответстват на номера на квадранта, в който ще бъде разположена легендата: 1 - десен горен, 2 - ляв горен, 3 - ляв долен, 4 - десен долен. Ако е указана 0, то MATLAB намира най-удачното място на графиката. Ако е указана 5, то MATLAB разполага легендата вън от координатните оси.

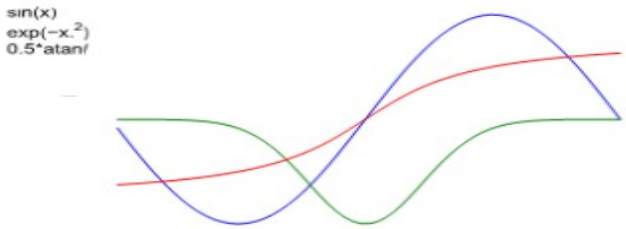
Ако x е вектор с дължина m, а Y - матрица с размери то командата:

plot(*x*,*Y*) е еквивалент на команда
plot(*x*,*Y*(:,1), *x*, *Y*(:,2), …, *x*, *Y*(:,*n*)).

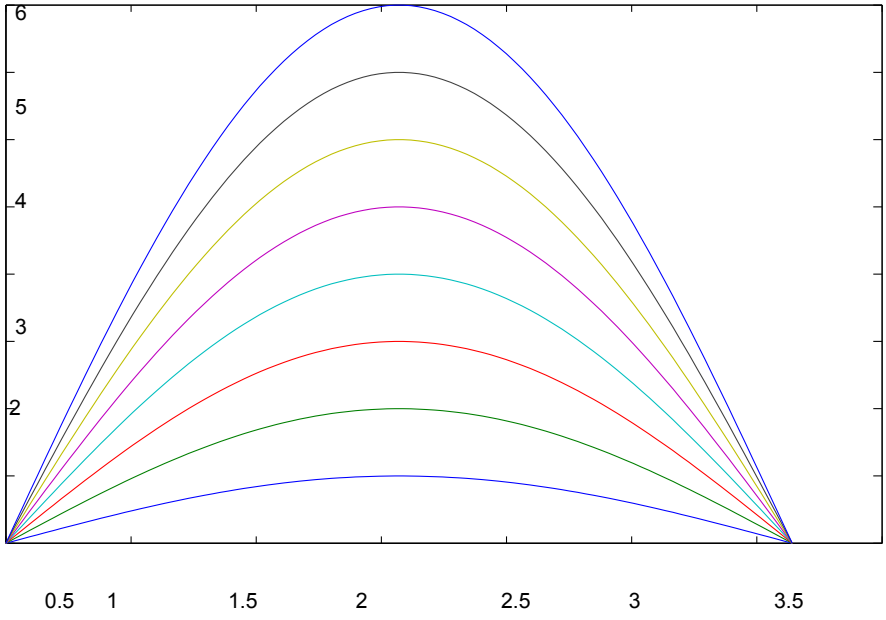
Ако няма указания за цвета на линиите и точките, той се избира автоматично от таблицата на цветовете, като белия цвят се изключва. Ако линиите са повече от 6, то изборът на цветовете се повтаря. За построяване на графики на функции с аргументи X и У, изменящи се в широки граници се използва логаритмичен мащаб. Синтаксисът на командата log.log(…) в този случай е аналогичен на този на командата plot(…). Логаритмичен мащаб се използва за координатните оси X и У. Пример за приложение на дадената команда:

```
x=logspace(-1,3);
loglog(x, exp(x)./x);
grid on.
```

Например, графиките от предходния пример могат да се получат с помощта на командата plot(x, [y1, y2, y3]). На фиг. 2 са дадени графиките на функциите sin(x),exp(-x^2) и 0,5 atan(x).



Фиг. 2. Три графики в една координатна система



Фиг.3. Синусоиди с различни амплитуди

Да начертаем 8 синусоиди с различни амплитуди:

```
a=1:8;
x=linspace(0,pi,100);
y=sin(x)';
plot(x, y*a);
```

и да разгледаме редуването на цветовете. Редът на цветовете е следния:

син—зелен—червен—небесен—лилав—жълт—черен

blue—green—red—cyan—magenta—yellow—black

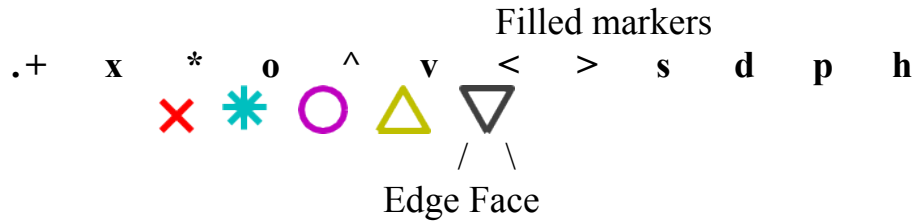
2.1.2. Стил и цвят на линията.

Да разгледаме още един вариант на извикване на функцията plot:

plot(*x*, *y*, стил)

Тук стилът е редът, състоящ се от 1 до 4 символа, обозначаващи цвета и стила на линията и типа на маркера:

Цвят: *c*, *m*, *y*, *r*, *g*, *b*, *w*, and *k*



Фиг 4. Типове маркери

Стил на линията: ¡ , ¡ ¡ , :, ¡ .

Тип маркера: +, o, *, x, s, d, ^, v, >, <, p, h

Цветът на линията и маркера се определят с една буква:

Символ	Цвят
’b’	син
’g’	зелен
’r’	червен
’c’	небесен
’m’	лилав
’y’	жълт
’k’	черен

Да разгледаме възможните стилове на линиите:

Таблица 2

Символ	Стил на линията
’-’	непрекъсната линия
’–’	штрих-линия
’:’	пунктирана линия

’-.’	штрих-пунктииана линия
’none’	няма линия

Може да се построи и зададе стила едновременно на няколко графики:

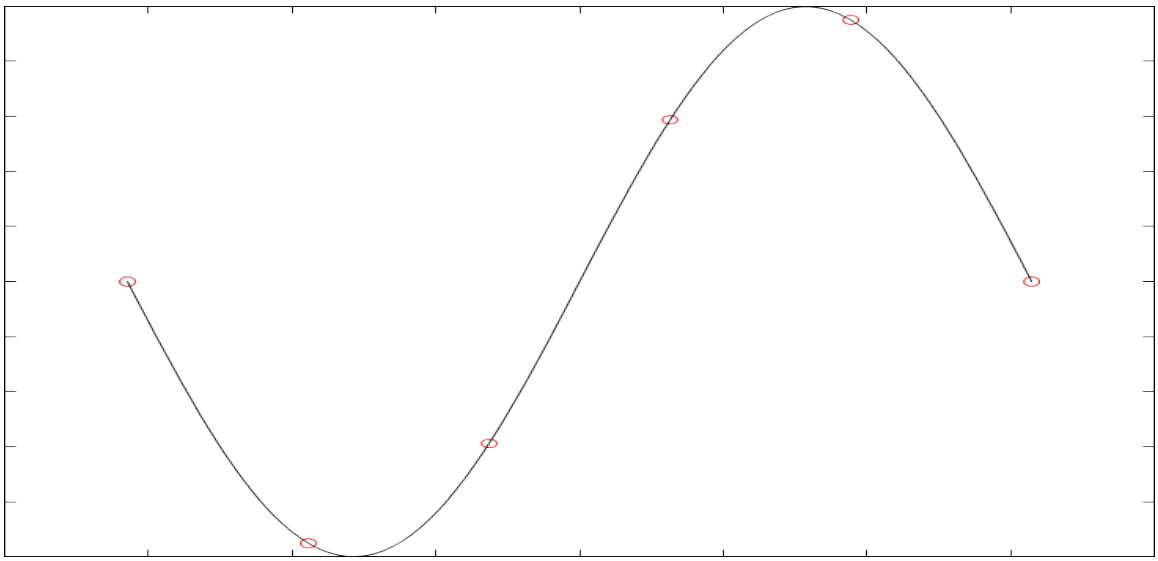
plot(x1, y1, стил1, x2, y2, стил2 …).

Средствата за създаване на графики в новите версии на MATLAB са съществено допълнени. Менюто на новата позиция Grapfics съдържа три команди:

New figure-открива празен прозорец за графика;
Plot tools-открива прозорец на нов мощен графичен редактор;
More plots… - открива прозорец на достъп към различни видове графика.

За решаване на графични задачи за построение се използва дескрипторна графика “Handle grapfics”. При нея на всеки графичен обект се поставя описание-дескриптор, към който са възможни препратки при използване на съответния обект. Дескрипторната графика позволява да се осъществи визуално програмиране на обекти чрез потребителския интерфейс-управляващи бутони, текстови панели и т.н. Графиките в MATLAB се строят в отделни мащабируеми и премествеми прозорци. Строят се, като се прилага линейна интерполация на функция в интервалите между отделни точки. Ако зададем интервал на изменение на аргумента x със стъпка 0,1 на функцията y=sin(x). За построение на графиката е достатъчно в началото да се зададе вектор x=0:0.1:15 и след това да се използва командата за построение на графика. И така, за да се построи графиката на синосуидата е необходимо да се изпълнят следните команди:

```
x=0:0.1:15;
y=sin(x);
plot(x,y)
grid off  % отменя  извеждането на работна  мрежа
```



Фиг. 5. Задаване на стила на няколко графики

Командата grid on добавя координатна мрежа към графиката. Наличието на мрежа облекчава количествената оценка на координатните точки към графиката. В експерименталнитео случаи се налага построението на много наложени една върху друга графики в един и същи прозорец. За това служи командата за продължение на графическото построение hold.

Пример:

```
x=linspace(¡pi,pi,6); y=sin(x);
xx=linspace(¡pi,pi,100);
yy=sin(xx);
plot(x,y,'or',xx,yy,'-k')
```

Резултатът е видим на фиг 5.

2.1.3. Командите axis и grid

Функцията

xlim([xmin,xmax]

задава диапазон на изменение на графиката по координати *x*; а

ylim([ymin,ymax]

установява диапазон на изменение на графиката по координати y.

Може веднага да се зададат границите на изменение и за двете коорди нати:

axis([*xmin*, *xmax*, *ymin*, *ymax*])

Командата axis auto може да възстанови режим, в който границите се изчисляват автоматично.Командата axis square установява еднакви граници по всички оси. Командата axis equal устанавява еднакъв мащаб по всички оси. Команда axis off премахва отметките по координатните оси и белия правоъгълник на графичния извод. При това всички графики остават. Командата axis on възстановява отметките на координатните оси.

Команда grid on включва, а команда grid off изключва режима на показване на растерната работна мрежа. За пример може да се построи графиката на функцията, описваща колебателни затихвания на различни линейни осцилатори:

```
t=[0:.1:10]';
y1=1.03*exp(¡0.25*t).*sin(0.97*t);
y2=1.07*exp(¡0.35*t).*sin(0.94*t);
y3=1.15*exp(¡0.5*t) .*sin(0.87*t);
y4=0.45*(exp(¡0.38*t)exp(¡2.62*t));
y5=0.22*(exp(¡0.21*t)exp(¡4.80*t));
Y=[y1, y2, y3, y4, y5];
plot(t, Y) ylim([¡0.4,.8]) grid on
```

2.1.4. Графики на многозначните функции

В MATLAB не е трудно да се получи графика на обратната функция, достатъчно е във функцията plot да се разменят местата на аргументите. В следващия пример ще се проследи построяването на графиката на функцията $y = \sin x$ и $x = \cos x$.

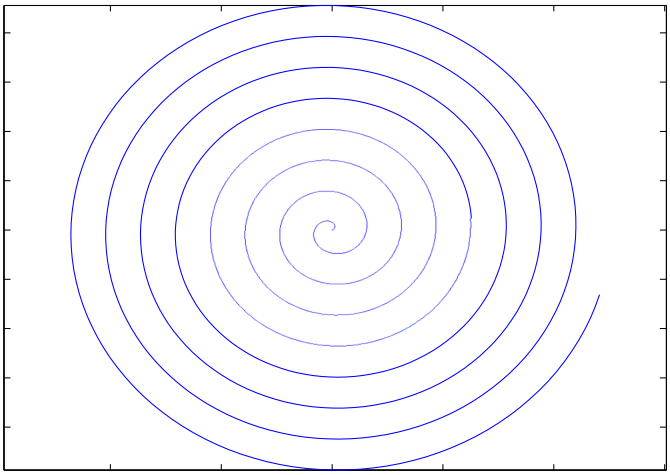
```
x = ¡3*pi:0.1:3*pi;
y = cos(x);
plot(x, y, y, x);
legend('y = cos(x)', 'x = cos(y)');
```

2.1.5. Криви, зададени параметрически

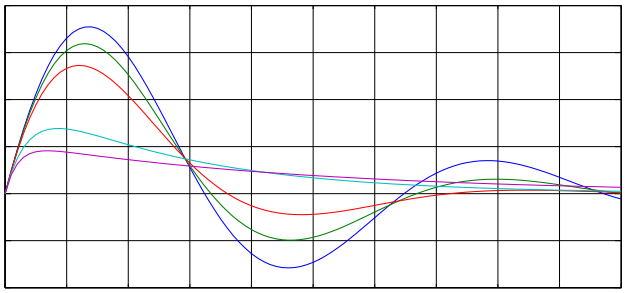
За изобразяване на криви, зададени параметрически, се използва функцията. Например, за се начертае архимедова спирала

```
t=[0:0.1:10];
y1=1.03*exp(¡0.25*t).*sin(0.97*t);
y2=1.07*exp(¡0.35*t).*sin(0.94*t);
y3=1.15*exp(¡0.5*t) .*sin(0.87*t);
y4=0.45*(exp(¡0.38*t)¡ exp(¡2.62*t));
y5=0.22*(exp(¡0.21*t)¡ exp(¡4.80*t));
```

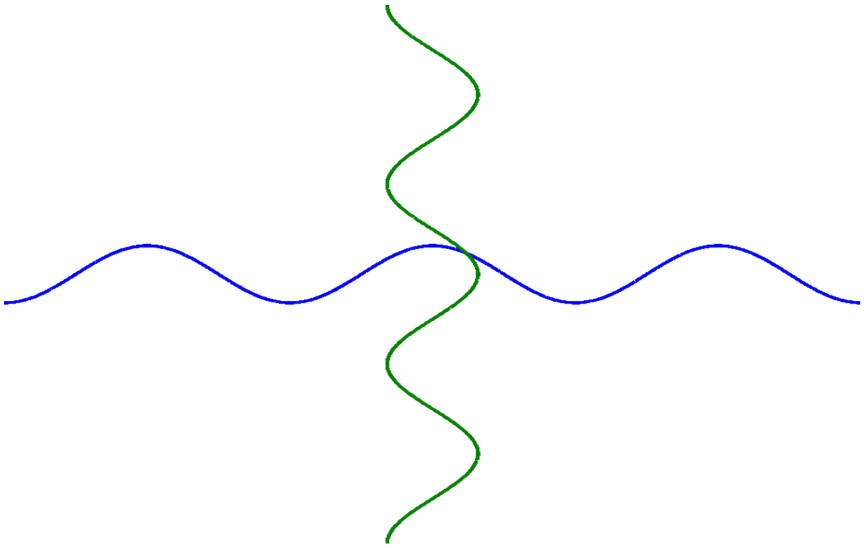
```
Y=[y1, y2, y3, y4, y5];
plot(t, Y) ylim([¡0.4, 0.8])
grid on
```



Фиг.6. Архимедова спирала



фиг.7. Линеен осцилатор



Фиг.8. Графики на многозначите функции

$y = \sin x$ и $x = \cos x$

```
x = ¡3*pi:.01:3*pi; :y = cos(x);
plot(x, y, y, x);
```

legend(’y = cos(x)’, ’x = cos(y)’);

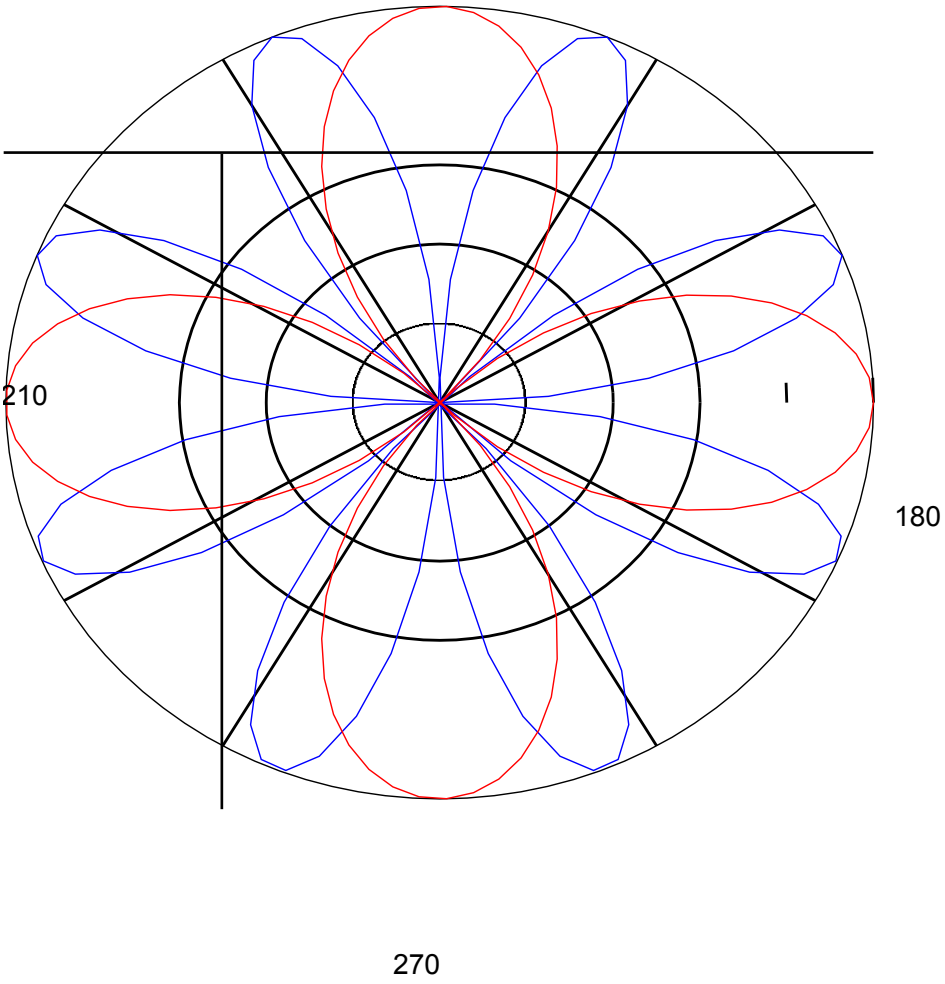
2.1.6. Графики в полярни координати

Функцията

polar(*phi*, *r*)

построява графиката на функция, зададена в полярни координати. Векторите phi и r трябва да имат еднаква дължина и да съдържат стойности на полярния ъгъл и на радиуса съответно. Например:

```
phi = linspace(0, 2*pi, 200);
polar(phi, sin(4*ph));
hold on;
polar(phi,
cos(2*phi), ’r’);
hold off;
phi = linspace(0, 2*pi, 180);
polar(phi, sin(4*ph));
hold on;
polar(phi,
cos(2*phi), ’r’);
hold off;
```



Фиг.9. Графика в полярни координати

2.2. Тримерна графика

2.2.1. Пространствени криви

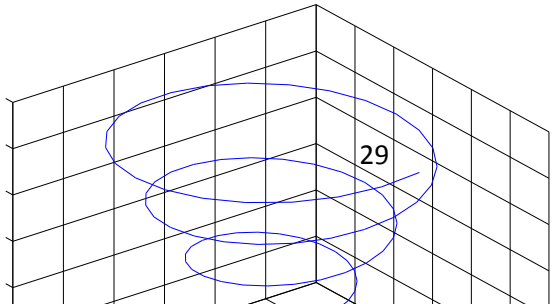
Функцията

plot3(*x, y, z, стил)*

е предназначена за построяване на пространствени криви има аналогичен синтаксис на функцията plot. За пример може да се начертае пространствена спирала.

x=t cos(*t*)*;y=t sin(t) 1<t<30:*
t=1:.25:30;
x=*t*.**cos*(*t*);
y=*t*.**sin*(*t*);
z=*t*;
plot3(*x, y, z*)
grid

Резултатът е видим на фиг.10.



Фиг.10. Развъртаща се спирала

2.2.2. Команда meshgrid

Този е функцията, незаменима при изобразяване на повърхнини. Нека *x* и *y* са вектори с дължина *n* и *m* съответно, тогава командата

[*X,Y*]=*meshgrid*(*x*, *y*)

Генерира две матрици X и Y с размери *m £ n*.
Членовете на тези матрици са от вида; x(1), x(2), …..x(n) и y(1), y(2)….y(n).
Повтарят се m реда.

```
[X,Y]=meshgrid(-10:0.25:10,-10:0.25:10);
F=sinc(sqrt((X/pi).^2l*+(Y/pi).^2));
Mesh*(X,Y,f);
axis ( [-10,10,-10,10,-0.3,1]
xlabel (‘{\bfx}’)
ylabel (‘{\bfy}’)
zlabel (‘{\bfsinc} ({\bfR})’)
hidden off
```

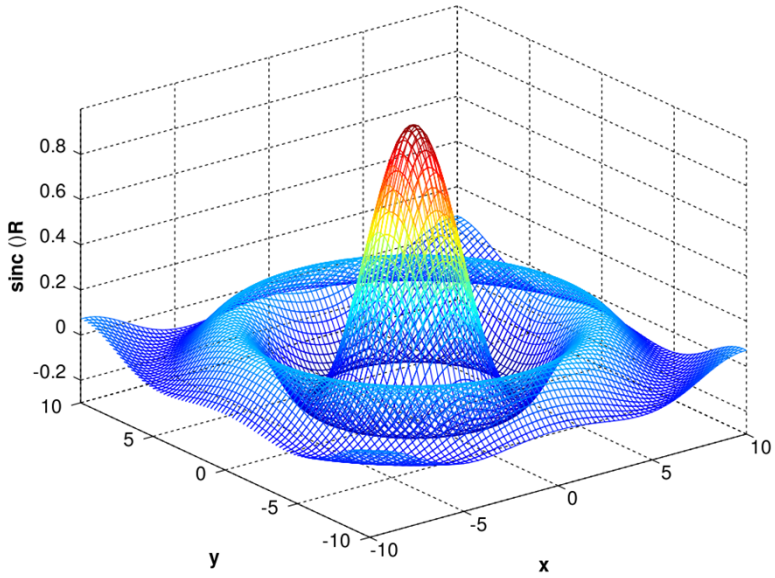
Командата meshgrid е най-често използваната при проектиране на триизмерни фигури. Разгъваща се в пространството повърхнина и развъртаща се спирала са показани съответно на фиг.10 и фиг.11. Командата grid on добавя координатна мрежа към графиката. Наличието на мрежа облекчава количествената оценка на координатните точки към графиката. В много случаи се налага построението на много наложени една върху друга графики в един и същи прозорец. За това служи командата за продължение на графическото построение hold. Съществуват допълнителните команди, свързани с управление на цветовите ефекти:

hidden-управлява показването на невидимите линии;

colstyle-отделя цвета и стила *на графиката от зададения масив;*
rgbplot-изобразява палитрата;
hsf2rgb-преобразува палитрата HSV в палитра RGB;
rgb2hsf- преобразува палитрата RGB в палитра HSV.

Създаването на координатни оси и управлението им, се осъществява с една група команди:

axes- създаване на координатни оси;
box- поставяне на правоъгълник около графика;
cla- премахване на построените координатни оси;
gsa- поставяне на правоъгълник около графика;
hold- запазване на построените координатни оси;
ishold-проверка на статуса hold (1-ако е запазен, 0-ако не е). Поставяне на легенди и скали в прозореца на графиката се извършва с менюто на бутона:
Inset.



Фиг.11. Резултатът от изпълнението на програмата с командата meshgrid

2.2.3. Команди mesh, surf, surfl

За построяване на графиките на функциите на две променливи в системата MATLAB има няколко функции. Ето някои от тях:

mesh(*X*, *Y*, *Z*) *surf*(*X*, *Y*, *Z*) *surfl*(*X*, *Y*, *Z*)

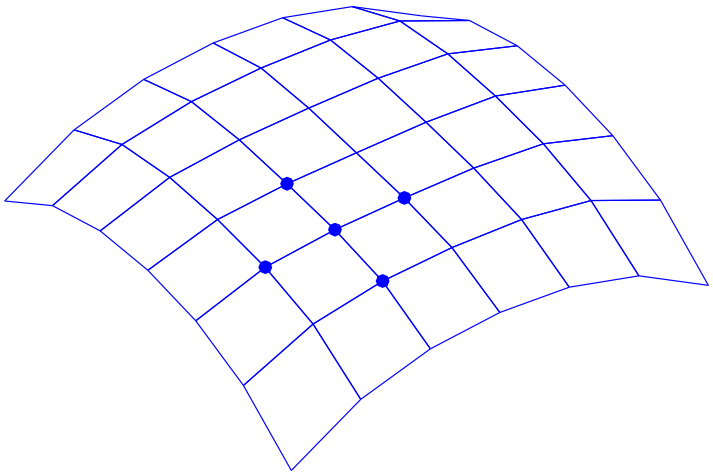
Параметрите на трите приведени функции имат еднакъв смисъл: X, Y, Z- това са масиви, определящи координатите x, y, z на повърхността на фигурата.

Функцията mesh начертава мрежеста (нишковидна) повърхност, съединявайки правите отрези на «съседни» точки, т. е. точки,

координатите на които са разположени или в съседни редове, или в съседни стълбове на матрицата X, Y, Z (фиг.12). На фиг.12 са дадени координатите на точките графично, а под нея са описани аналитично.

Пример за приложение на командата surf може да се наблюдава на графика от типа на фиг.13. Необжодимо е да се отбележи, че построената тримерна графика може да се върти с мишката и да се наблюдава под различен ъгъл.

```
[X,Y]=meshgrid(-5:0.1:5);% построяване на повърхност и нейните проекции
Z=X.*sin(X+Y);
meshc(X,Y,Z)
```



Фиг.12. Параметрите X, Y, Z на функцията mesh и графиката

X(i+1,j), Y(i+1,j), Z(i+1,j) X(i+1,j),

Yi+1,j), Z(i+1,j) X(i+1,j), Y(i+1,j),

Z(i+1,j) X(i,j), Y (i,j), Z(i,j) X(i,j 1), Y (i,j 1), Z(i,j -1)

```
[X, Y]=meshgrid(¡3:.25:3, ¡3:.25:3);
Z=sin(X).*sin(Y);
mesh(X, Y, Z);
```

По подразбиране невидимите линии не се изобразяват. Командата

```
hidden off
```

отключва режим на отделяне на невидимите линии.

Непрекъсната (червената) повърхност се чертае с функцията:

```
surf(X, Y, Z)
```

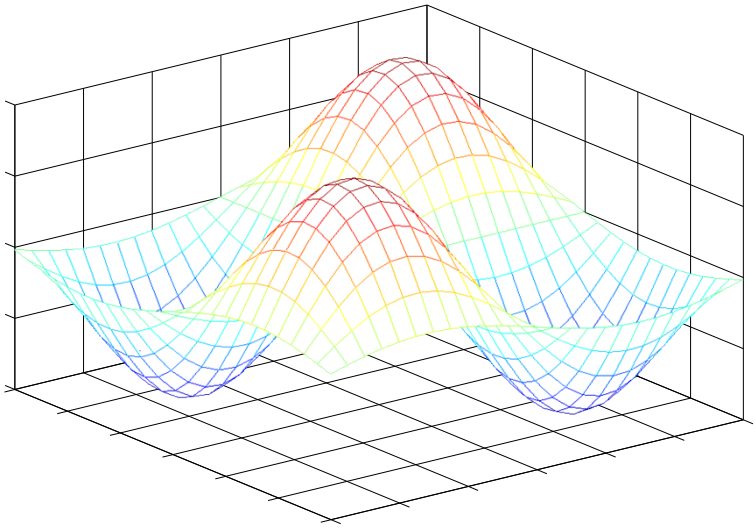
Командата

colormap палитра

позволява да се изберат за оцветяване на повърхността определен набор цветове, т.е. палитра. Достъпни са следните палитри: winter, spring, summer, autumn, bone, copper, hot, cool, gray, pink и др.

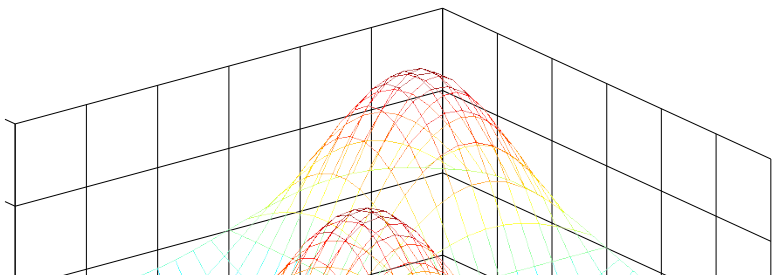
Разгледаните дотук примери са построение на графика са типични и несложни, но от тях може да се направи извод, че за решаване на конкретна задача трябва да се познават съответните команди и функции. За целта може да се използва и справочната система на MATLAB.

За въртене на графиката е достатъчно да се активизира последния бутон отдясно на инструменталното меню, с изображение на пунктирана окръжност със стрелка. След това, въвеждайки курсора на мишката в областта на графиката и натискайки левия бутон на мишката, може с кръгови движения да се застави графиката да се върти. В новите версии на MATLAB могат да се завъртат и двумерните графики и да се наблюдава обръщането на равнината, в която са построени.



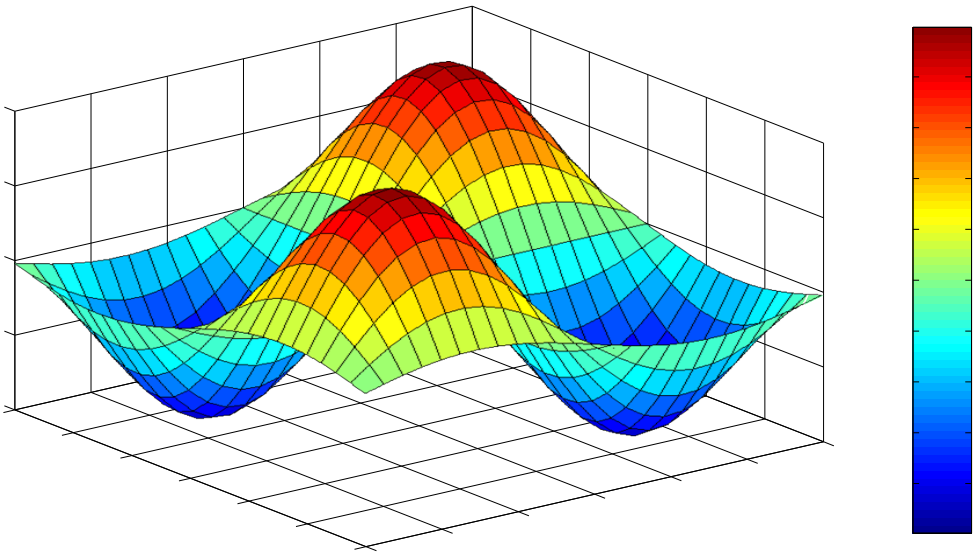
Фиг..13.Нишковиден модел на повърхността

За изучаване на свойствата на графиката е предвиден инспектор на графиката. Той се активира от прозореца на графичния редактор чрез бутона Inspector. С помощта на този подпрозорец могат да се установят различни свойства на графичния обект, като цвят на конкретната линия, нейния тип, дебелина и много други свойства, които са нужни за инженерен анализ на обекта.



Фиг.14. Модел, в който невидимите части са отстранени

За превключване в режим на редактиране на графика се избира бутона Edit Plot (редактиране на графика), с изображение курсор-стрелка. В този режим графиката може да се управлява с помощта на контексното меню, активирано с десния бутон на мишката. С помощта на миш-ката също може да се отдели графика.С натискане на левия бутон във вид на правоъгълник се въвеждат набор точки в областта на графиката, отваря се правоъгълник, в който могат да се нанасят стрелки или надписи. Панелът на инструментите в MATLAB е опростен и съдържа типичните бутони отляво надясно.



Фиг.15. Палитрата " Summer"

Може да се зададе метод за оцветяване на повърхностите може с помощта на следващите команди: За превключване в режим на редактиране на графика се избира бутона Edit Plot (редактиране на

графика), с изображение курсор-стрелка. В този режим графиката може да се управлява с помощта на контексното меню, активирано с десния бутон на мишката. С помощта на мишката също може да се отдели графика. С натискане на левия бутон във вид на правоъгълник се въвеждат набор точки в областта на графика-та, отваря се правоъгълник, в който могат да се нанасят стрелки или надписи. Панелът на инструментите в MATLAB е опростен и съдържа типичните бутони отляво надясно.

shading faceted shading flat shading interp
shading faceted

украсява всеки четириъгълник, който изгражда повърхността с един цвят. Ребрата са в черен цвят. Този режим действа по подразбиране. Действието shading faceted е аналогично на действието на предишната команда, но при това ребрата са невидими. Командата shading interp оцветява всеки връх неравномерно, използвайки линейната интерполация на цвета.

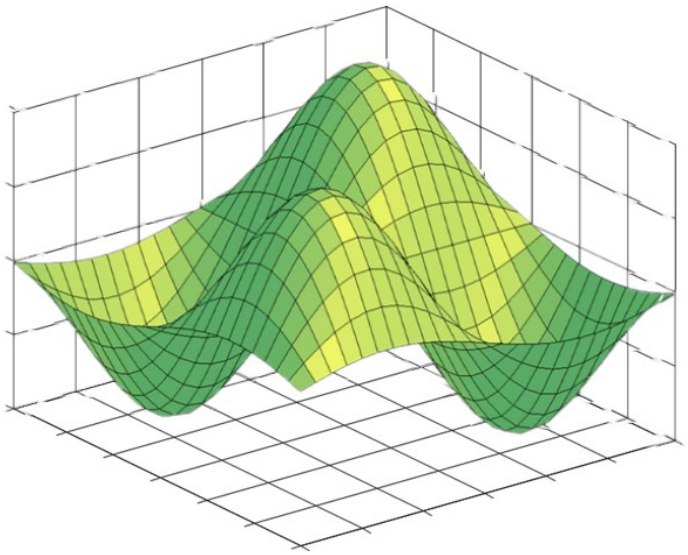
Функцията surfl(X, Y, Z) е аналогична на функцията surf(X, Y, Z), но в нея за контраст се използват цвят, имитиращ осветление. Функцията surf p се използва заедно с палитрите gray, bone, copper, pink.

Функцията view(az,el) определя точката на камерата (наблюдателя). Азимутът az е ъгълът на завъртане на камерата (в градуси) край оста Oz, измерван от отрицателното направлени на оста Oy. Положителните стойности на азимута съответстват на посока, обратна на часовниковата стрелка. Възвишението el е ъгъл (в градуси), който съставлява вектор, идващ от на-чалото на кооpлинатите към камерата с равнина Oxy. Положителните стойности на възвишението съответстват на точките над равнината Oxy, отрицателните стойности на точките под равнината Oxy.

Командата abce наблюдава завъртането й под различни ъгли. За целта може да се използва справочната система на MATLAB, от менюто „Help”.

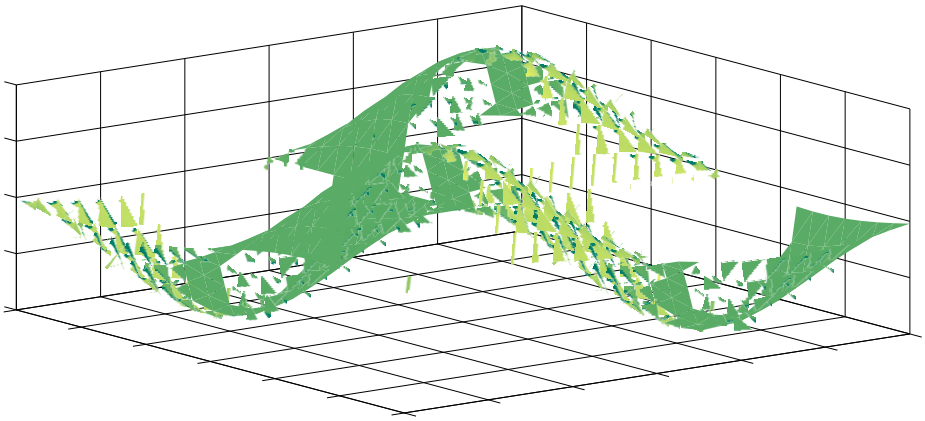
В графичния прозорец от командата Insert може да се активира функцията Ins за поставяне на надписи по дължина на осите, титулни надписи, надписи в графиката, стрелки, легенди и цветови скали, както е показано на фиг.15. По-големи възможности за форматиране на графиката открива позицията Tools в графичния прозорец. Тук може да се задават опции за редактиране, изменение на размерите, въртене, извод на графичния курсор за определяне на координатите на произволна точка от графиката и други опции и команди. Особено внимание може да се обърне на обработката на графиката в нейния прозорец Basic Fitting и опцията за представяне на статистически данни на графиката Data Statistics.

Надписи допълнително могат да се нанасят и от инсрументалния панел,чрез бутона а надпис „A“.



Фиг.16. С командата Shading faseted

Движението на точка по зададена траектория, както в двумерното, така и в тримерното пространство, е най-опростения пример за анимация.



Фиг.17. С командата shading flat

Областта на паметта на компютъра, в която се съхраняват отделните имена на променливите, се нарича работна област- Workspase Browser.

2.3. Примери

2.3.1. Тороид

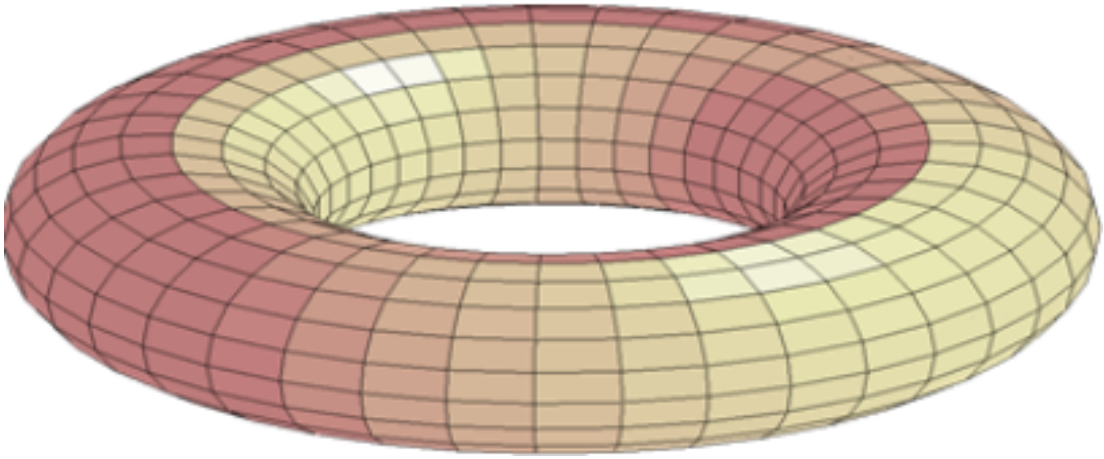
Тороидът може да се зададе параметрически:

$x = (R + r\cos v)\cos u;$

$y = (R + r\cos v);$

$z = r\sin v,$ където

r е „малкия“, а R е „големия“ радиус на тороида (фиг.18).

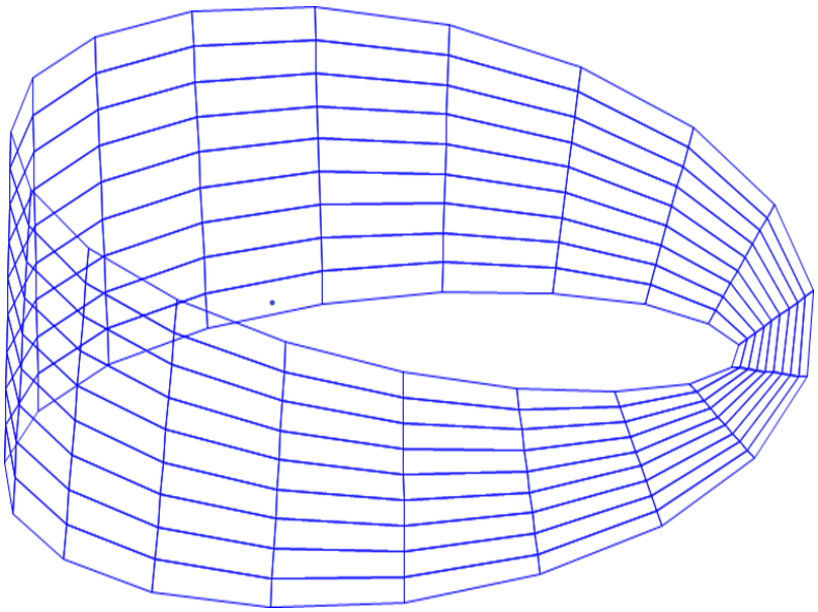


Фиг.18. Тороид

```
[U, V]=meshgrid(linspace(0, 2*pi, n), linspace(0, 2*pi, m));
X =(R + r.*cos(V)).*cos(U);
Y=(R + r.*cos(V)).*sin(U);
Z = r.*sin(V);
surfl(X, Y, Z)
axis([¡5 5 ¡5 5 ¡5 5]) axis off
colormap pink
[U, V] = meshgrid(linspace(0, 2*pi, n), linspace(0, 2*pi, m));
X = (R + r.*cos(V)).*cos(U);
Y = (R + r.*cos(V)).*sin(U);
Z = r.*sin(V);
surfl(X, Y, Z)
axis([¡15 15 ¡15 15 ¡15 15]) axis off
colormap pink
```

Чрез командите who и whos може да се получи списък на въведените променливи и на тяхната дължина:

```
>> who  Your variables are:
MVx >> whos Name Size Bytes Class
M 4x4 128 double array
V 1x5 40 double array
X 1x1 8 double array
Grand total is 22 elements using 176 bytes.
```



Фиг.19. Мьобиусиана

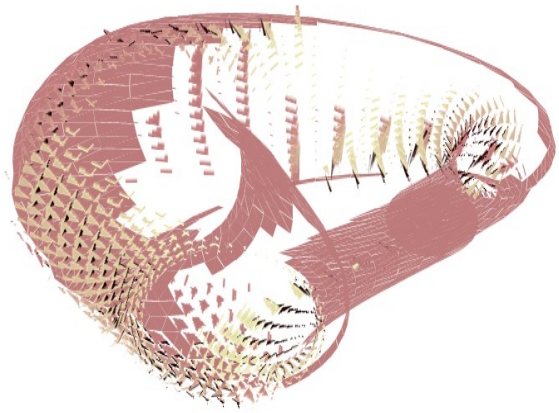
```
R=5; r=2; n=40;
m=20;
[U,V]=meshgrid(linspace(0, 2*pi, n), linspace(0, 2*pi, m));
X=(R + r.*cos(V)).*cos(U);
Y=(R + r.*cos(V)).*sin(U);
Z=r.*sin(V);
surfl(X,Y,Z)
```

2.3.2. Крива на Мьобиус

Кривата на Мьобиус може да се зададе параметрически, например:

```
x=cos u + v cos u=2:/cos u;
y=sin u+ v cos u=2:/sin u;
z=v sin u=2
[u,v]=meshgrid(linspace(0, 2*pi, 20),
linspace(¡.1, .1, 10)); mesh( ...
cos(u)+v.*cos(u/2).*cos(u), ...
sin(u)+v.*cos(u/2).*sin(u), ... v.*sin(u/2), ...
'Edgecolor', 'blue'); view(15, 56);
axis off;
hidden off;
```

Мьобиусианата е изобразена на фиг.19.



Фиг.20. Бутилката на Клайн

$R = 5;\ r = 2;\ n = 40;$

2.3.3. Бутилка на Клайн

Бутилката на Клайн може да се получи от „залепването“ на две повърхности, всяка от които може да се зададе параметрически:

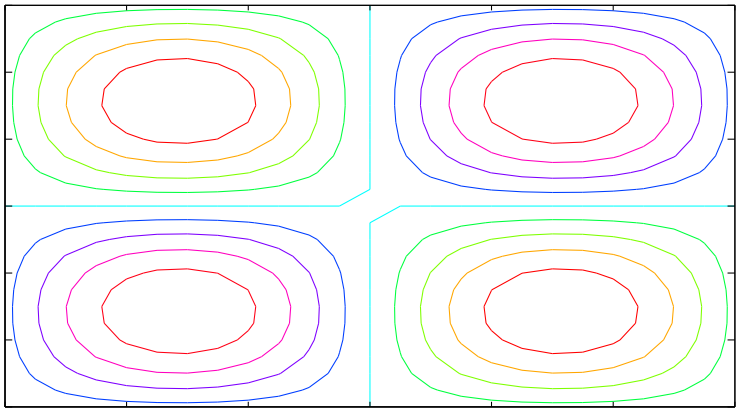
$x=a\cos u(1+\sin u)+r\cos u\cos v; b\sin u+r\sin u\cos v;$
$z=r\sin v$
$x=a\cos u(1+\sin u)+r\cos(v+...)$
$y=b\sin u;$
$(0<u<...; 0<v<2...)$; *където r=1jcos(u/2)*

```
[u, v]=meshgrid(linspace(0, pi, 25), linspace(0, 2*pi, 50))
r1= 4*(1¡cos(u)/2);
x1=6*cos(u).*(1+sin(u))+r1.*cos(u).*cos(v);
y1=16*sin(u)+r1.*sin(u).*cos(v);
z1=r1.*sin(v);
[u, v]=meshgrid(linspace(0, pi, 15), linspace(0, 2*pi, 25))
r1=4*(1¡cos(u)/2);
x1=6*cos(u).*(1+sin(u))+r1.*cos(u).*cos(v);
y1=26*sin(u)+r1.*sin(u).*cos(v);
z1=r1.*sin(v);
[u,v]=meshgrid(linspace(pi, 2*pi, 25), linspace(0, 2*pi, 50))
r2=4*(1¡cos(u)/2);
x2=6*cos(u).*(1+sin(u))+r2.*cos(v+pi); y2 = 16*sin(u);
z2=r2.*sin(v);
surfl(x1, y1, z1); hold on; surfl(x2, y2, z2); view(9, 21) ...
colormap pink;
shading interp;
hold off;
axis equal;
axis off;
```

2.4. Линии на нивото.

Функцията contour(X, Y, Z) чертае линиите на повърхнината за функцията z = f(x;y), стойностите на която се пазят в матриците X, Y и Z. Например:

```
[X, Y] = meshgrid(¡3:.25:3, ¡3:.25:3);
Z = sin(X).*sin(Y);
contour(X, Y, Z);
```



Фиг.21. Линии на нивото

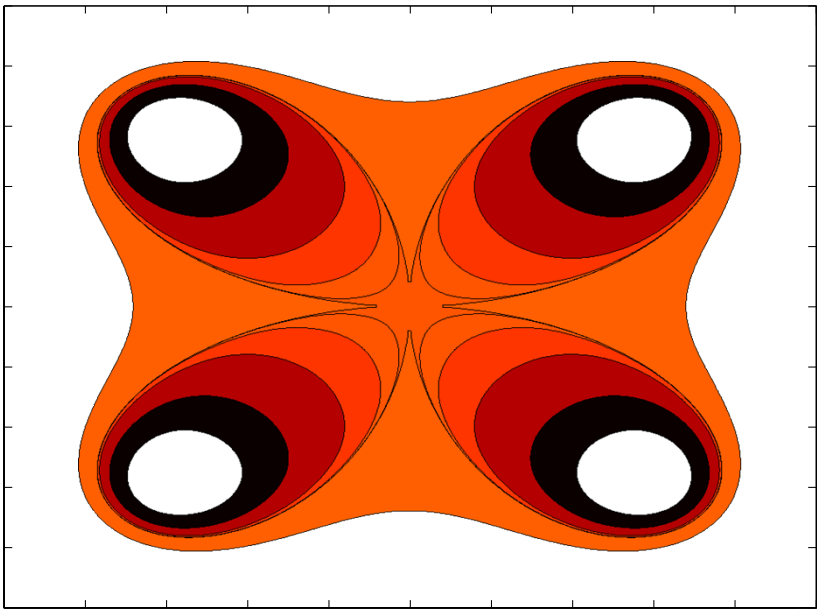
Количеството и големината на стойностите на функцията, за които ще бъдат начертани линиите на нивото, се определят автоматично. Да се зададе нужното количество k на линиите на нивото може, като се определи стойността на четвъртия аргумент:

```
contour(X, Y, Z, k);
```

Ако четвъртият аргумент е вектор, то той се интерпретира като набор от стойности на функциите, за които трябва да бъдат построени линии на нивото. Функциите рисуват линии на нивото и оцветяват промеждутъка между тях в цветове от текущата палитра. Например функцията

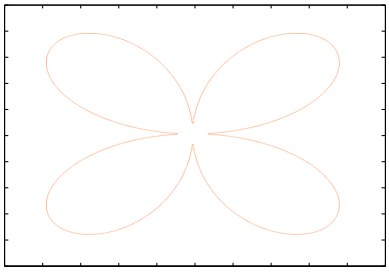
```
z=(x2 + y2)3 ¡4x2y2:
[X, Y]=meshgrid(linspace(¡1, 1, 300));
Z=(X.^2 + Y.^2).^3 ¡ 4*X.^2.*Y.^2;
contourf(X, Y, Z, [¡0.1 ¡0.05 ¡0.01 ¡0.001 ¡0.00005 0 0.1])
colormap hot
```

рисува линии на нивото с оцветени промеждутъци.



фиг..22. Линии на нивото с оцветени промеждутъци

С помощта на функцията contour може да се построят криви, зададени с уравнения. За предишния пример contour(X, Y, Z, [0 0]); рисува четирилистна роза (x2 + y2)3¡4x2y2 = 0. Дублира се 0 във вектора, явяващ се последен аргумент в функцията, тъй като contour(X, Y, Z, 0); би било проинтерпретирано, като задание да се построи линия на ниво k = 0.



Фиг.23. Четирилистна детелина. Кривата е зададена чрез уравнението $(x^2 + y^2)^3 ¡ 4x^2y^2 = 0$

2.5. Избягване на табулацията на функцията.

Ако е известно аналитичното задание на функцията във вид на обикновено или параметрическо уравнение, то за удобство понякога се използва ez-вариант на съответстващата графична команда. Към такива ez-функции се отнасят ezplot, ezpolar, ezplot3, ezmesh, ezsurf, ezcontour. Използвайки ги, ползвателят не е длъжен за мисли за предварителната табулация на функцията. Изборът на възлите и самите изчисления на стойностите на функцията в тези възли се осъществява автоматически.

Примери за използване на ez-функции
Построяване на графиката на функцията $y = \cos(\mathrm{tg}\, x)$, ¡ $3 < x < 3$:

ezplot('cos(tan(x))',¡ 3 , 3)

Чертае се окръжност: $x^2 + y^2 = 1$:

ezplot('x^2 + y^2 - 1')

Изобразява се кривата $x = \sin y$:

ezplot('x - sin(y)')
x=t.sint, като t<0<10
*ezplot('sin(t)/t', 'cos(t)/t', [1, 10*pi])*

пространствена крива, зададена параметрически,

$x = t \sin t;$ $0 < t < 6$
y=tcost
z=t
*ezplot3('t*sin(t)', 't*cos(t)', 't', [0,6*pi])*
*ezplot3('t*sin(t)', 't*cos(t)', 't', [0,6*pi])*.

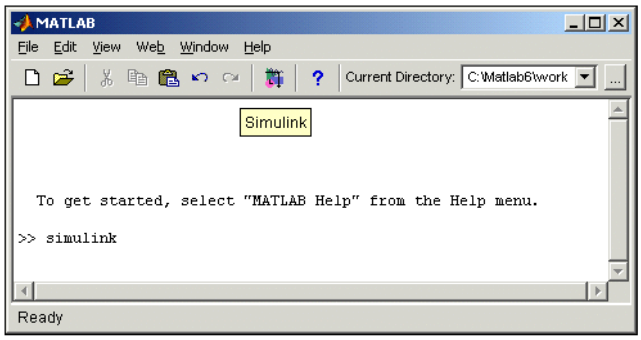
Построяване на графиката и линиите на нивото за функцията

$z = \sin(x^2 + y^2) + e^{¡x^2}$; ¡ 2 $< x <$ 2; ¡ 2 $< y <$ 2 :
f='sin(x^2 + y^2) + exp(-x^2)'; *d*=[¡ 2, 2, ¡ 2, 2];
ezmesh(*f*,*d*)
ezsurf(*f*,*d*);
ezcontour(*f*,*d*).

3. Общи сведения за работа със Simulink

Програмата Simulink е разширение на програмния пакет MATLAB. При моделиране с използване на Simulink се реализира принципа на визуалното програмиране, в съответствие с който ползвателят на екрана създава модел на устройство от библиотеки със стандартни блокове и осъществява симулации.

Simulink е достатъчно самостоятелен инсрумент на MATLAB и не зависи от останалите приложения. От друга страна достъпът до функциите на MATLAB и другите негови инструменти остава открит и те могат да се използват в Simulink.Тази програма може да се стартира по три начина. След отваряне на основния прозорец може да се пусне програмата по един от тях:



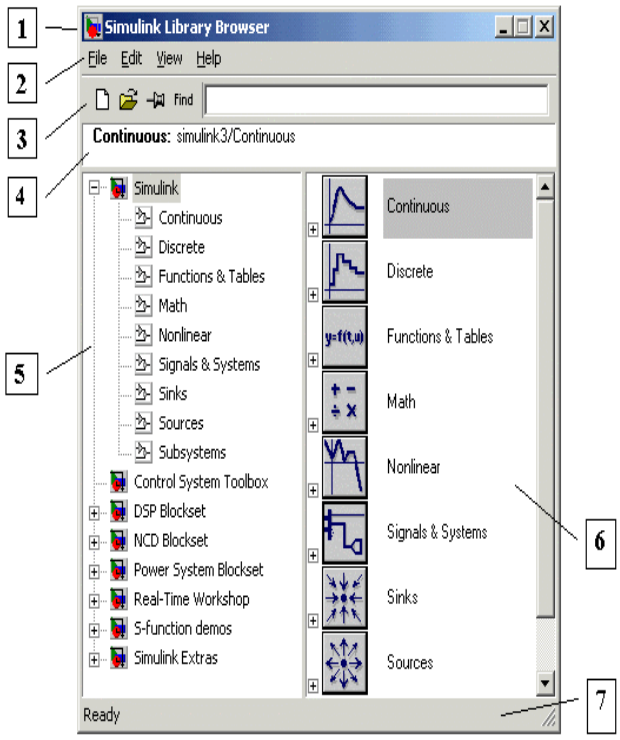
Фиг. 24. Основен прозорец на програмата

1. Да се избере иконата от командния панел.
2. В командния ред на MATLAB да се напише Simulink.
3. Да се натисне ENTER на клавиатурата.
3. Да се изпълни командата Open в менюто File.
4. Да се създаде файл на модела от типа mdl-файл.

3.1. Панел на библиотеките в Simulink.

От панела на библиотеките в това приложение започва моделирането. С избирането на съответната библиотеката се отваря списък на подбиблиотечното съдържание.

На фиг. 25 могат да се видят основните библиотеки.



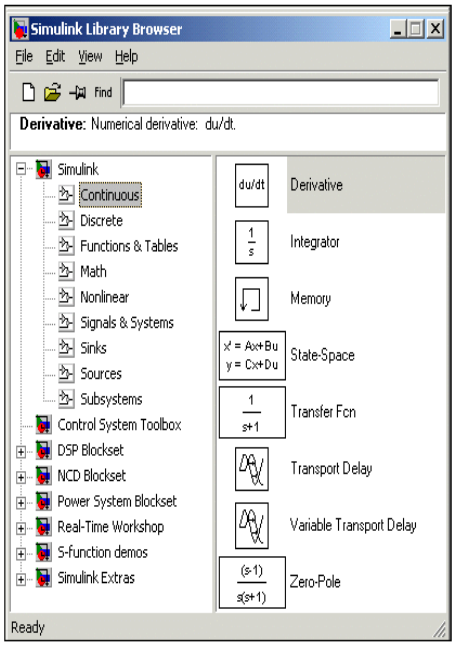
Фиг.25 Панел на каталога на библиотеките в Simulink

Панелът на каталога на библиотеките съдържа следните елементи:

1. Название на прозореца– Simulink Library Browser;
2. Меню, с командите File, Edit, View, Help;
3. Панел на инсрументите;
4. Прозорец с коментари за извеждане на пояснящи съобщения за избрания блок;
5. Списък на разделите на библиотеките, реализиран във вид на дърво;
6. Прозорец на съдържанието на разделите на библиотеката;
7. Страница на състоянието, съдържаща подсказка по изпълняваното действие.

На фиг.25 е дадена основната библиотека Simulink (в лявата част на прозореца) и са показани нейните раздели (в дясната част на прозореца). Библиотеката на Simulink съдържа следните основни раздели:
1. Continuous – линейни блокове;
2. Discrete – дискретни блокове;
3. Functions & Tables – функции и таблици;
4. Math – блокове за математически операции;
5. Nonlinear – нелинейни блокове;
6. Signals & Systems – сигнали и системи;
7. Sinks - регистриращи устройства;
8. Sources — източници на сигнали и въздействия;
9. Subsystems – блокове на подсистеми.

Списъкът на разделите на библиотеката Simulink е представен във вид на дърво, и правилата за работа с него са общи за списъци от този вид. Пиктограмата на затворения възел на дървовидната структура съдържа символ "+", а пиктограмата на отворения възел съдържа символ "-".



Фиг.26 Панел със списъка на блоковете на раздела на библиотеката

За да се отвори или затвори възел на дървовидната структура, достатъчно е да се щракне на неговата пиктограма с левия клавиш на мишката.
При избора на библиотека, в дясната част на менюта на панела се изобразява нейното съдържание.

За отваряне на работния прозорец се използват командите в менюто, което съдържа следните пунктове:

1. (Файл) - Работа с файовете на библиотеката.
2. Edit (Редактиране) - Добавяне на блокове и тяхното търсене (по название).
3.View (Вид) - Управление на поазването на елементите на интерфейса.
4. Help (Справка) - Извеждане на справка по каталога на библиотеките.

За работа с каталога може да се използват и бутоните на лентата на инструментите (фиг.27).

иг.27. Лента на инструментите

Бутоните на инструменталния панел имат следното предназначение:
1. Да се създаде нов S-модел (да се открие нов прозорец на модела).
2. Да се открие един от съществуващите S-модели.
3. Да се изменят свойствата на прозореца на каталога. Даден бутон позволява да се установи режим на изобразяване на прозорците на всички икони. Повторното му избиране отменя този режим.
4. Търсене на блок по наименование (по първия символ на наименованието). След като блокът бъде намерен, се отваря съответстващия библиотечен раздел, а блокът се отделя. Ако няма блок с такова название, то в прозореца на коментарите излиза съобщение:
„Not found“ <име на блока> (Блокът не е открит).

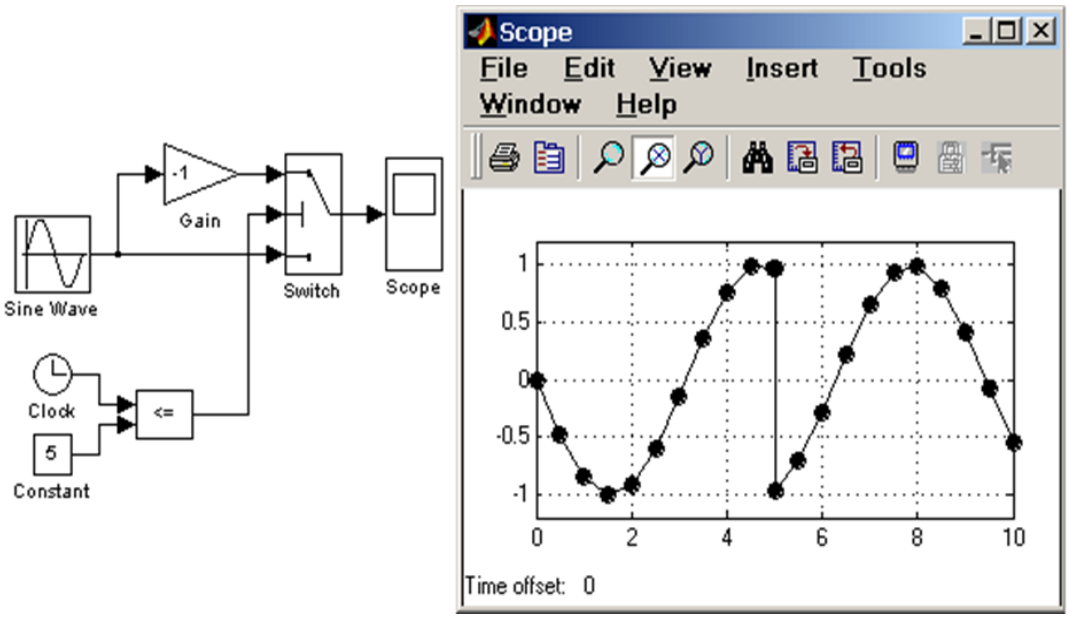
3.2. Панел за задаване на точки на прекъсване по условието: “Break on conditions”

Често при извършване на симулационни разчети се получават прекъсвания , които се отчитат и във получените времедиаграми. Справка за отделните блокове при правилно избран входен сигнал може да се получи от Help, а от Blocks се получава справка по отделния блок.Възникналите прекъсвания могат да бъдат по различни причини. Програмата отчита тези причини чрез панела “Break on conditions”, чиито опции са дадени на фиг.28.

Фиг.28. Списък на условията за прекъсване

Панелът на фиг.26 съдържа списък от условия, при изпълнението на които симулацията трябва да спре. Случая „Zero crossing" се получава при преминаване на сигнала през нулевото ниво при неговото скокообразно изменение. На фиг.26 е показан пример на модел при такава ситуация. От графиката се вижда, че в момента t=5 сигналът се изменя скокообразно и пресича нулевото ниво. Други случаи, при които се преустановява симулационния разчет:

„Step size limited by state"-състояние, ограничаващо стъпката на разчета. Тази опция е полезна при модели, изискващи твърде много разчетни стъпки. Опцията преустановява моделирането, когато моделът използва блок с променлива стъпка и блокът се сблъсква със състояние, изискващо ограничение на стъпката на разчет.



Фиг.29. Симулационен модел със скокообразно изменение на сигнала

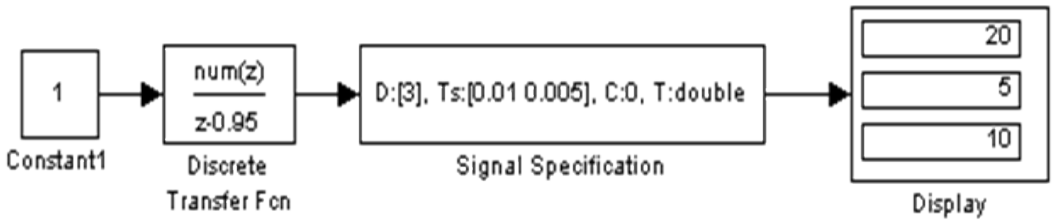
„Minor time steps"- режим с използване на вътрешни малки стъпки. При изпълнение на разчетите Simulink може да намалява зададената стъпка за достигане на нужната точност. За да се видят и тези малки вътрешни стъпки, е необходимо да се установи тази опция.

„NaN values“-нечислова стойност. Разчетът ще бъде прекъснат, когато изчислената стойност е безкрайнаили лежи вън от заложения диапазон стойности, които могат да бъдат представени при моделирането (твърде малки или твърде големи).

Блокът за проверка на сигнала е „Signal Specification“ и изпълнява проверка за съответствие на зададения за сигнала параметър. Параметрите са:

1. „Dimension“ – Размерност на сигнала. Задава се чрез скалар, ако входния сигнал е вектор или чрез матрица от вида [m x n], (m – количество редове, n – количество стълбове), ако входният сигнал е матрица. Ако стойността на параметъра е зададена като „ –1“ (минус 1), то проверка не се възпроизвежда.
2. „Sample time“ – Стъпка на моделното време. Задава чрез вектор от вида[period offset], където period е стойност на стъпката на моделното време, а offset е смущение. Ако стойността на параметъра е зададена като „–1” (минус 1), то проверка не възниква. Може също да се зададе стойност „ –1“ (минус 1) и отделно за параметрите period или offset. В този случай няма да се проверяват именно тези параметри. Важно е да се избере прецизно моделното време.
3. „Data Type“ - Тип на данните. Избира се от списъка: auto (проверка не се генерира), double, single, int8, uint8, int16, uint16, int32, uint32 или boolean.
4. „Signal type“ – Тип на сигнала. Избира се от списъка: auto (проверка не се генерира), real или complex. На пиктограмата на блока са изобразени проверяемите параметри на сигнала и техните значения.

Пример за приложение на блока Signal Specification е показан на фиг.30.



Фиг.30. Пример за използване на блок Signal Specification

Командата“ Simulink Model Differencing“ сравнява два Simulink-модела и генерира графично изображение на различията между тях. На даденото изображение на фиг.31 се разполагат еднакви моделни блокове с различни атрибути, оцветени в червено. Блоковете със син цвят присъстват само в един от двата модела. Ползвателят може да настрои изображението така, че да проследи само блоковете с графични различия, само блоковете с неграфични различия или блокове, различаващи се по други параметри и

характеристики. За изпълнение на сравнението на моделите е необходимо да се изпълни командата:

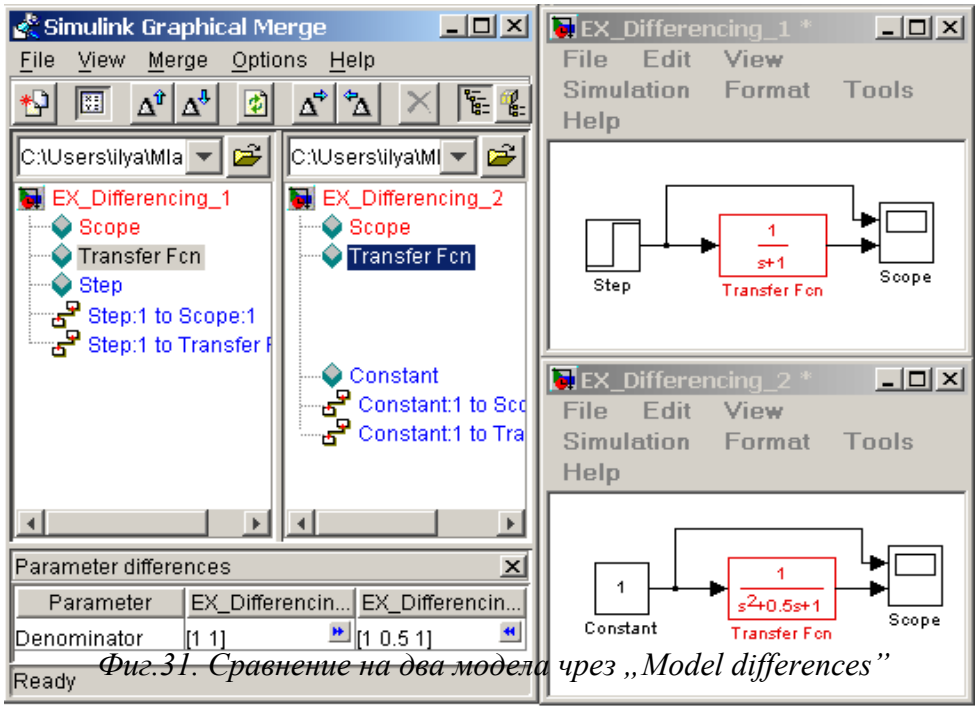
Model differences/Merge/Compare two models

от менюто Tools на прозореца на първия модел и в процеса на диалога да се избере файла на втория модел. Възможно е също така да се изпълни сравнение на текущото състояние на модела и неговият последен запис.На фиг. 31 е показан пример за сравнение на два модела.

Пакетът с разширения „Instrument Control Toolbox" предоставя програмни функции за стиковка с осцилографа на MATLAB. За детайлно запознаване с всяка функция на пакета е достатъчно в командния прозорец да се зададе командата:

>> insthelp name,

където name е име на съответната функция от този пакет.



Фиг.31. Сравнение на два модела чрез „Model differences"

Други примери за упражнение са дадени в „Допълнителни примери".

Допълнителни примери

Накратко за МАТЛАБ: Наименованието MATLAB произлиза от MATrix LABoratory. MATLAB е продукт, създаден за работа със структури от данни, основани на матрици, широк спектър от функции, интегрирана среда на разработка, обектно-ориентирани възможности и интерфейси към програми, написани на други програмни езици.

Приложение на продукта:

– математически изчисления;
– разработка на алгоритми;
– моделиране, симулиране и създаване на прототипи;
– представяне, визуализиране и анализ на данни;
– графика;
– разработка на приложения с графичен потребителски интерфейс.

Включва специални приложения (toolboxes) за обработка на сигнали, системи за управление, невронни мрежи, симулация и др.

Характеристики на системата MATLAB:

1. MATLAB език – език от високо ниво за обработка на матрици;
включва: оператори, функции, структури данни, вход/изход,
обектно-ориентирано програмиране.

2. Работна среда на MATLAB – средства за управление на променли-вите в работната среда, внасяне и изнасяне на данни;

- разработка и управление на M-файлове, MATLAB приложения.

3. Управление на графика – двумерна и тримерна визуализация, обработка на изображения, анимация и представяне на графики;

- изграждане на графичен потребителски интерфейс.

4. Библиотека с математически функции – сума, тригонометрични - функции,
комплексна аритметика, обръщане на матрица, Беселови функции, бързи трансформации на Фурие и др.

5. Интерфейс за приложни програми – динамично свързване на модули, написани на C, C++, Java или Fortran с модули от MATLAB.

7. MATLAB като мощен калкулатор:

» указва въвеждане на команда
; блокира извеждане на резултата
ans променлива със стойността на резултата
= присвояване
fun() вградена функция
... пренасяне на израз на нов ред

» 2+3 ans =

```
5
» x=sin(0.5)
x =
0.4794
» V=[1 2 3 4];
» sin(V)
ans =
0.8415    0.9093    0.1411   -0.7568
```

вградена функция

```
» exp(V)
ans =
2.7183    7.3891   20.0855   54.5982
» magic(3)
ans =
8  1  6
3  5  7
4  9  2
```

MAGIC(N) е NxN матрица от целите числа 1xN2 с еднакви суми във всеки ред, стълб и диагонали, където N>0 без N=2.

Типове данни

Дефинирани са 15 типа данни (или класове). Всеки тип данни е във формата на масив.

Двумерният масив се нарича матрица. Едномерният-вектор.

Матрица1x1 се нарича скалар. .

Примери:

```
A=[7 2;4 3];
» B=[1 4;6;7];
» A+B
ans =
8     6
10    10
» A*B
ans =
19    42
22    37
```

```
% поелемеентно
» A.*B
ans =
7       8
24      21
» A.^B
ans =
7       16
4096  2187
 % ляво деление
 » A\B
 ans =
 -0.6923      -0.1538
 2.9231       2.5385
 » inv(A)*B
 ans =
 -0.6923      -0.1538
 2.9231       2.5385
 % дясно деление
 » B/A
 ans =
 -1.0000       2.0000
   -0.7692       2.8462
   » B*inv(A)
   ans =
   -1.0000    2.0000
   -0.7692    2.8462
```

inv(A) – обръща матрицата A.

1. Променливи

– автоматично се създава променлива за всяко име и се отделя необходимата памет;

– името на променливата се състои от букви, цифри, знак за подчертване (_), като MATLAB използва първите 31 символа;

– различава малки и главни букви.

» A=10 % създава се 1x1 матрица с име A и един елемент със стойност 10
A =10

2. Числа – съхраняват се с дълга плаваща запетая (16 значещи цифри,
област 10-308 до 10+308)

3. Начини за представяне:

– десетично представяне с незадължителна десетична точка и водещ знак плюс;
– научно представяне чрез буквата e, определяща 10 на степен;
– имагинерните числа използват като суфикс i
или j.

10 -43 0.001

3.4632987 1.63e-201.43i

» z=3+j*4;

% Конвертира z в полярни координати

» A=abs(z) A =

5

» theta=angle(z) theta =

0.9273

% Обратно конвертира до z=x+jy

» z=A*exp(j*theta) z =

3.0000 + 4.0000i

4. Символи и коментари

Символна константа е последователност от символи, заградени в апос трофи ′.Коментар започва със символа %.

» A=[′София′;′Пловдив′] % матрица 2x7

A = София Пловдив

5. Оператори

– аритметични оператори;
– оператори за сравнение;
– логически оператори.

5.1. Оператор:

a:b – резултатът е вектор-ред, съдържащ числата от a до b със стъпка 1.

a:c:b – резултатът е вектор-ред, съдържащ числата

от a до b със стъпка c.

```
» 1:10          % стъпка 1
ans =
1     2     3     4     5   6     7     8     9     10
» 100:-10:50          % стъпка -10
ans =
100     90     80     70     60     50
```

6. Функции

abs абсолютна стойност

sqrt корен квадратен

exp е на степен

sin синус

clear изчиства променливите и функциите от паметта

clear all изчиства всички променливи, глобални

променливи, функции и връзки

close затваря фигура

close all затваря всички прозорци с фигури

help получаване на помощ

format установява изходния формат

Константи:

pi 3.1459265

i имагинерна единица -1

j имагинерна единица -1

» help elfun % елементарни математически функции

» help specfun % специални математически функции

» help elmat % матрични функции

» format short % мащабиран формат с фиксирана точка с 5 цифри

» format long % мащабиран формат с фиксирана точка с 15 цифри

» format hex % шестнадесетичен формат

» format rat % апроксимация чрез отношение от малки цели числа

Примери:

» s=(1+sqrt(10))/2 s =

2.0811

» n=1:5 % оператор : -> вектор-ред

n =

1 2 3 4 5

» n=n' % оператор ' (транспониране) -> вектор-стълб

n =

1

2

3

4

5

» x=n.^2 % оператор .^ (поелементно повдигане на степен)

x =

1

4

9

16

25

Изчертаване на графики

plot(y) Създава начупена линейна графика на вектора y спрямо индекса на елементите на y.
plot(x,y) Създава линейна графика на вектора y спрямо вектора x.
plot(x1,y1,x2,y2,…) Създава много графики на
векторите yi спрямо векторите xi.
plot(x,y,′цвят_маркер_линия′) Изчертава графика с определен цвят, тип маркер и тип линия; цвят_ маркер_ линия е 1-, 2- или 3- символен низ; Ако се дефинира цвят маркер без линия, изчертава се само маркерът.
loglog(x,y) Създава логаритмична графика log10(y) спрямо log10(x).
semilogx(x,y) Създава полулогаритмична графика на вектора y спрямо log10(x)
semilogy(x,y) Създава полулогаритмична графика на log10(y) спрямо вектора x.

Цвят	Линия		Маркер
r червен	-	плътна	+
b син	--	тирета	o
g зелен	-.	тирета и точки	*
w бял	.	от точки	.
k черен	:	двуеточия	x
c синьозелен	none	без линия	
m пурпурен			
y жълт			

Пример: Изчертава синусоида в интервала

» t=0:pi/100:2*pi;
» y=sin(t);
» plot(t,y)

В полярни координати:

» polar(t,y) xlabel(′текст′) Добавя текст до x–оста.
ylabel(′текст′) Добавя текст до y-оста.
title(′текст′) Добавя текст като заглавие.
text(x,y,′текст′) Добавя текст, който започва от т. (x,y) н а графичния екран.
axis([xmin xmax ymin ymax])Управлява мащабирането на осите.
Grid Променя състоянието на мрежата.
grid on / grid off Добавя/премахва координатната мрежа.
Hold Променя състоянието на графиката.
hold on / hold off Задържа текущата графика, като следващите графични команди се добавят към графиката / връща подразбира-щия се режим.
figure(H) Създава нова фигура с манипулоатор H или прави текущата фигура да бъде H.

subplot(m,n,p) или subplot(mnp) Изчертава няколко графики в един прозорец, като разделя прозореца на mxn матрица от коорди-натни системи и избира p-тата система.

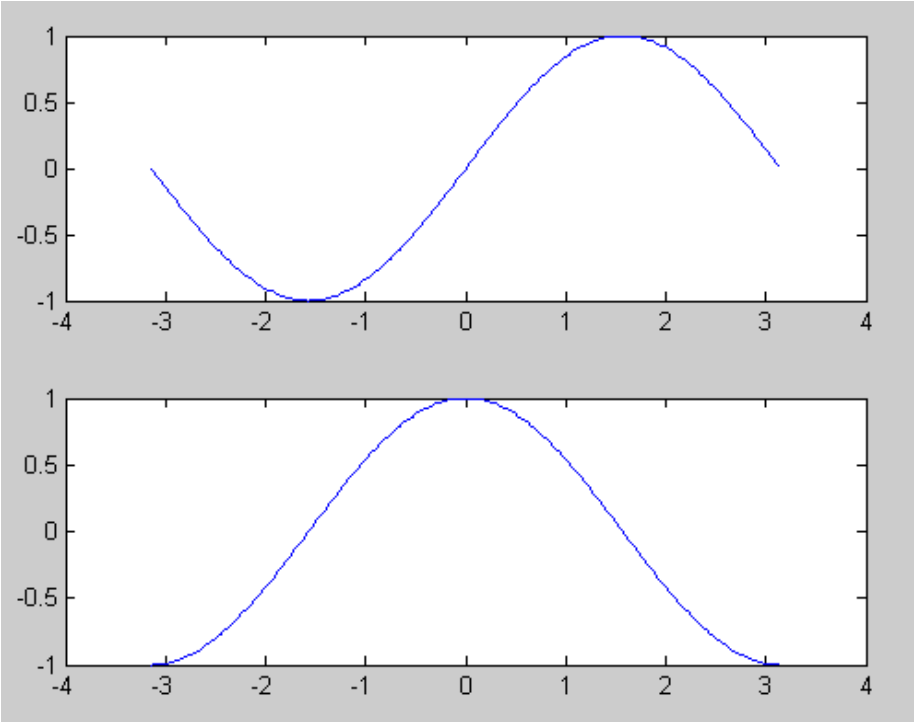
subplot(111) Връща към пълен екран за изчертаване.

```
» y1=sin(t);
» y2=cos(t);
» subplot(211),plot(t,y1)
» subplot(212),plot(t,y2)
» subplot(111)
```

% Програма за построяване на четири графики в различни подпрозорци:

```
x=-5:0.1:5;
subplot(2,2,1),plot(x,sin(x))
subplot(2,2,2),plot(sin(5*x),cos(2*x+0.2))
subplot(2,2,3),contour(peaks)
subplot(2,2,4),surf(peaks)
```

При пускане на програмата се строят 4 графики в различни подпрозорци на един прозорец, разделен на четири части.

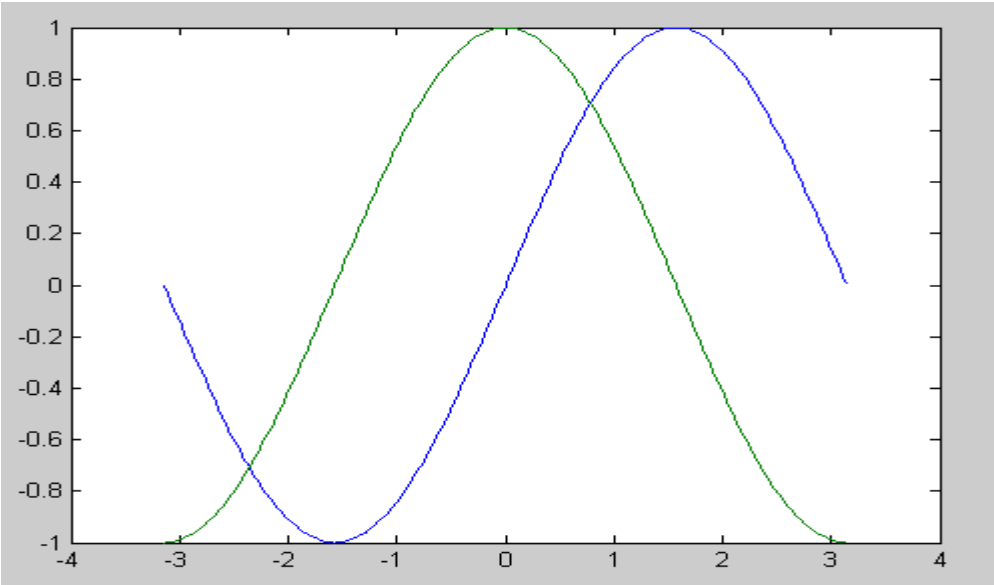


Фиг.32.

plot(x1,y1,x2,y2,…) Изчертава няколко графики в един прозорец върху една координатна система. Видове масиви в MATlAB:
double-3*10^300; 5+6i Масив от реални числа с двойна точност.

Най-често използваният тип в MATLAB.
cell{15 'Java'} Масив от елементи с различни размери, които съдържат други масиви.
structure stud.fn=123456;
stud.name='Иван';Масиви от структури с елементи, които съдържат други масиви.
@име_функция fhandle = @sin; feval(fhandle,0.5) Манипулатор на функция, който обикновено се предава като аргумент на функция и се оценява чрез функцията feval.
потребителски клас inline('sin(x)')
Създава се чрез използване на функциите на MATLAB като наследници на класа structure:
Java клас javax.swing.JFrame.
Използват се Java класове.

plot(t,y1,t,y2)



Фиг.33. Две графики в един прозорец

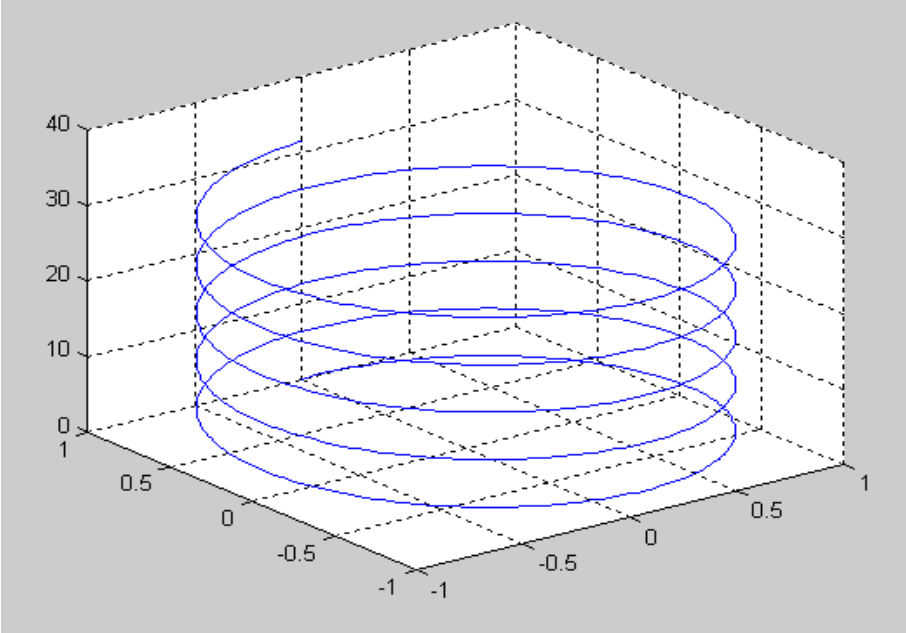
plot3(x,y,z)-3-D аналог на plot. Генерира 3-D линия с координати елементите на трите вектора x, y и z (векторите трябва да бъдат с еднаква дължина) и създава 2-D проекция на линията върху екрана.

```
» t = 0:pi/100:10*pi;
» y1 = sin(t);
» y2 = cos(t);
» plot3(y1, y2, t);
» grid on
```

Резултатът е показан на фиг.34.

» x=1.234; % преглеждане на променливите в работната област

» V=[1 2 3 4 5];
»M=magic(4);
» who
Your variables are:
MVx
» whos
Name Size Bytes Class
M 4x4 128 double array
V 1x5 40 double array
X 1x1 8 double array
Grand total is 22 elements using 176 bytes.

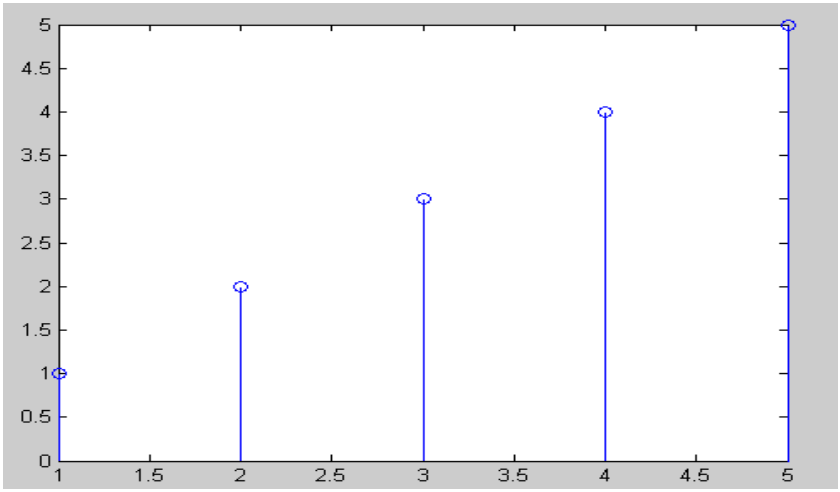


Фиг.34.

stem(y) За дискретни сигнали изчертава данните във вектора y спрямо оста x като всяка точка е малка окръжност и права линия.
stem(x,y) Изчертава данните във вектора y спрямо вектора x.
stem(x,y,′filled′) Използва запълнени маркери.
stem(x,y,′тип_линия′) Използва тип_линия при изчертаването.

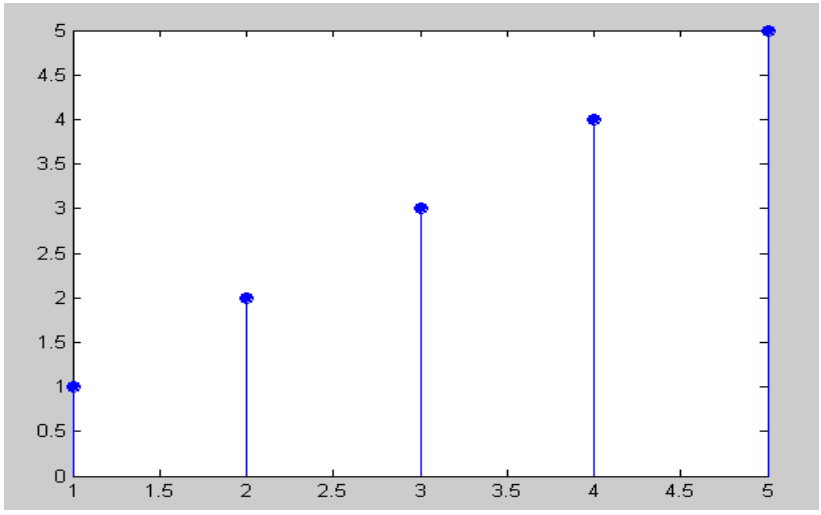
Пример: Изчертава дискретен сигнал y[x].

» x=1:5;
» y=[1 2 3 4 5];
» stem(x,y)



Фиг.35. Дискретен сигнал

» stem(x,y,'filled')-фиг.36



Фиг.36.Дискретен сигнал със запълнени маркери

Действия с матрици

M1и M2 са матрици. Значение на отделните аритметични оператори: A1 и A2 са вектори. При поелементното деление отляво надясно за всеки еле- мент на матриците се извършва операцията: M1(I,j)./M2(I,j), а при поелементното деление отдясно наляво: M2(I,j).\M1(I,j).

+	M1+M2 Събиране	* M1*M2	Умножение на матрици;
-	M1-M2 Изваждане	/ M1/M2	Дясно деление на матрици; Отляво надясно
.*	A1*A2 Умножение на масиви	\ M1\M2	Ляво деление на матрици Отдясно наляво

./ A1./A2 Дясно деление на масиви
отляво надясно, поелементно

^ M1^x Повдигане на степен на матрици

.\ A1.\ A2 Ляво деление на масиви

отдясно наляво, поелементно

+	+M	Унарен плюс
-	-M	Унарен минус
:		Двуеточие
.^	A1.^x	Поелементно повдигане на масив на степен x
.'	M .’	Транспониране
'	M’	Комплексно спрегнато транспониране

stem(y) За дискретни сигнали изчертава данните във вектора y спрямо оста x като всяка точка е малка окръжност и права линия.
stem(x,y) Изчертава данните във вектора y спрямо вектора x.
stem(x,y,′filled′) Използва запълнени маркери.
stem(x,y,′тип_линия′) Използва тип линия при изчертаването.

1. Въвеждане на матрици чрез явен списък от елементи:

» A=[16 3 2 13;5 10 11 8;9 6 7 12;4 15 14 1]

2. Сума от елементите във всеки стълб – резултатът е вектор-ред и се запазва в променливата ans.

» sum(A) ans =

34 34 34 34

3. Сума от елементите на всеки ред – резултатът е вектор-стълб

Алгоритъм:

1) транспониране на матрицата (операция ′);

2) изчисляване на сумите по стълбове;

3) транспониране на резултата

» A′
ans =

16	5	9	4
3	10	6	15
2	11	7	14
13	8	12	1

» sum(A′)′ ans=

34
34
34
34
34

4. Сума от елементите от главния диагонал

```
» diag(A)
ans =
16
10
7

» sum(diag(A))
ans =
34
```

5. Сума от елементите от второстепенния диагонал

```
» sum(diag(fliplr(A)))
ans =
34
```

Функцията fliplr обръща матрицата отляво надясно.

6. Индекси – елементът в ред i и стълб j се означава чрез A(i,j).
а) достъп до елемент на матрицата

```
» A(4,2)
ans = 15
```

б) сума от елементите в 4-тия стълб

```
» A(1,4)+A(2,4)+A(3,4)+A(4,4)
ans = 34
```

в) достъп до елемент на масив чрез един индекс –масивът се съхранява по стълбове като един дълъг вектор-стълб; напр. достъп до елемент A(4,2):

```
» A(8)
ans =15
```

Използване на оператор:

```
» sum(A(1:4,4))        % сума от 4-тия стълб
ans =
34

» sum(A(:,end))        % сума от последния стълб
```

Обратна матрица – функция inv

```
» A=pascal(3)
ans =
A =
1     1     1

1     2     3

1     3     6

» inv(A)
ans =
 3     3     1

3     5     2

 1     2     1
```

Специални матрици

zeros всички елементи са 0;
ones всички елементи са 1;
rand правилно разпределени случайни числа;
randn нормално разпределени случайни числа;
magic магически квадрат от степен N е NxN; т.е. матрица с елементи естествени числа от 1 до N2 така, че сумата от числата във всеки хоризонтал, вертикал или главен диагонал е една и съща, равна на N(N2+1)/2;
pascal матрица на Паскал от степен N; т.е. симетрична положително дефинирана матрица с целочислени стойности, получени от триъгълника на Паскал (първото и последното число във всеки ред е 1, а другите се получават като сума от двете числа над него).

```
» o=ones(2,4)
o =
```

1	1	1	1
1	1	1	1

» A=magic(3)
A =

8 1 6

3 5 7

4 9 2

» A=pascal(3)
A =

1 1 1

1 2 3

1 3 6

Слепване на матрици

Оператор за слепване

» A=magic(3)

A =

8 1 6

3 5 7

4 9 2

» B=[A A+5;A+10 A+20]
B =

8	1	6	13	6	11
3	5	7	8	10	12
4	9	2	9	14	7
18	11	16	28	21	26
13	15	17	23	25	27
14	19	12	24	29	22

Изтриване на редове и стълбове
[] двойка квадратни скоби

» A=magic(3) A=

8 1 6

3 5 7

4 9 2

» A(2,:)=[] % изтрива втори ред от A

A =

8	1	6
4	9	2

A(:,2)=[] % изтрива втори стълб от A

A =

8	6
4	2

Изграждане на таблици

» n=(0:9)'; % вектор-стълб
» pows = [n n.^2] % таблица от квадратите на 2
pows =
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81

Вход/изход на информация

Въвеждане на информация в диалогов режим – чрез функция input;

X=input('низ') Като подсказка се извежда низ, потребителят въвежда MATLAB израз, който се изчислява чрез променливите от работното пространство и резултатът се запазва в X.

X=input ('низ','s') Като подсказка се извежда низ, потребителят въвежда символен низ, който се запазва в X като MATLAB низ.

```
» A=[1 2 3 4 5];
» A(6)=input('Enter A[6]=')
Enter A[6]=6
A =
1  2       3       4       5       6
» name=input('Ime=','s')
Ime=Ivan name = Ivan
```

Извеждане на информация – чрез функция disp;
disp(X) Извежда масива X, без да отпечатва неговото име.
Ако X е низ, извежда се текста.

```
» disp(A(6))
6
» disp(name) Ivan
```

Файлова система MATLAB

Видове файлове:

1. .mat - двоичен файл, в който се записва работната сесия на системата. Запазване съдържанието на работната сесия във файл име.mat

```
» save име
```

Зареждане на работната сесия от файл име.mat

```
» load име
```

2. m - текстов файл, който съдържа функции и скриптове.

- файл-скрипт – съдържа команди на MATLAB:
- обработва съществуващи данни в работното пространство или създава нови данни;
-създадените променливи остават в работното пространство;
-файл-функция – има входни аргументи и връща изходни резултати:
-името на m-файла и името на функцията съвпадат;
- обработва променливи в тяхното собствено работно пространство, различно от основното.

Пример:

Файл-скрипт my.m

File =>New =>M-file

```
A=[…
16.0 3.0 2.0 13.0
5.0 10.0 11.0 8.0
9.0 6.0 7.0 12.0
4.0 15.0 14.0 1.0];
```

File =>Save As… =>my

Скриптът създава в работното пространство на MATLAB променлива A, която съдържа матрица.

Изпълнение на .m файл

– в работното пространство

```
» my
» A
A =
16      3      2     13
5      10     11      8
```

16	3	2	13
5	10	11	8

Операторът my чете от файла
my.m и използва създадената променлива A, която съдържа матрицата.

Пример:

Файл-функция my.m

File =>New =>M-file

```
function y=my() y=[...
 16.0   3.0    2.0   13.0
 5.0    10.0  11.0   8.0
 9.0    6.0   7.0    12.0
 4.0    15.0  14.0  1.0];
```

File =>Save As… =>my
File=> Compile

Изпълнение на .m файл

– в работното пространство

```
» A=my
A =
   16    3     2    13
    5   10    11     8
    9    6     7    12
    4   15    14     1
```

Извиква се функцията my и върнатият резултат се присвоява на променливата A, която получава матрицата.

Глобални променливи

global X

X – глобална променлива;

- ако X се дефинира в няколко функции и в работното пространство като global, тогава те споделят единствено копие на X;
- инициализира се с празна матрица.

Управляващи структури

1. Условен оператор if

if израз оператори elseif израз
оператори
else израз
оператори
end

2. Условен оператор switch

switch switch_израз case case_израз,
оператор,…,оператор
case {case_израз,...,case_израз} оператор,…,оператор...
otherwise оператор,…,оператор
end

3. Оператор за цикъл for

for променлива=старт:нарастване:край,
оператори
end

4. Оператор за цикъл while

while израз
оператори end

5. Оператор за прекъсване на цикъл break –
прекъсва изпълнението на while или for.

```
» for k=1:10, x(k)=cos(k); end
» x
```

```
x =
0.5403    -0.4161    -0.9900    -0.6536    0.2837    0.9602    0.7539-
0.1455    -0.9111    -0.8391
```
или

```
» k=1:10;
» x=cos(k);
» x
» a=2;

» b=5;
» if(a>b) c=a; else c=b; end
» c
c =
5
```

Пример: 3-битов A/D конвертор. Преобразува аналоговия сигнал x в цифров сигнал y.
Файл-функция AD3.m: function y=AD3(x)

```
if x<-2.5 y=0; elseif x<-1.5 y=1; elseif x<-0.5 y=2;
» y1=AD3(-1.25)
y1 =
2
» y2=AD3(2.57)
y2 =
6
» y3=AD3(6.0)
 y3 =
 7
```

Нули на функция. Минимум на функция.
1. Числено решаване на уравнение f(x)=0

```
X=fzero(FUN,X0)
```
Търси корените на функцията FUN около X0.

```
» X=fzero('sin(x)',3)% намира корена на sin(x) около X0=3
X =
3.1416
» X=fzero('sin(x)',[-1,1])
X =
0

» X=fzero('sin(x)',[-4,-3])
X =
```

-3.1416

2. Минимум на функция f(x)

X=fminbnd(FUN,X1,X2)

Търси минимума на функцията FUN в интервала [X1,X2].

» min=fminbnd('sin(x)',-2,-1) % минимум на sin(x) в интервала -2<x<-1

min =

-1.5708

Няма функция за намиране максимума на f(x) –

max(f(x))=min(-f(x))

» max=fminbnd('-sin(x)',1,2) % максимум на sin(x) в интервала 1<x<2

max =

1.5708

3. Диференциране

» syms x y; % дефинира реални символни променливи

» diff(x^y) % първа производна спрямо x (по подразбиране)

ans = x^y*y/x

» diff(x^y,x) % първа производна спрямо x

ans =

x^y*y/x

» diff(sin(y*x),x,3) % трета производна спрямо x

ans =

-cos(y*x)*y^3

4.Интегриране

» syms x;

» int(x^2,x) % неопределен интеграл

ans =

1/3*x^3

» int((x^2-2)/(x^3-1),x,2,5)% определен интеграл

ans =

-2/3*log(2)+2/3*log(31)+2/3*3^(1/2)*atan(11/3*3^(1/2))-2/3*log(7)- 2/3*3^(1/2)*atan(5/3*3^(1/2))

Нелинейни числени методи

I.Числено решаване на обикновени диференциални уравнения

Всяко обикновено диференциално уравнение от n- ти ред може да се запише като система от n уравнения от първи ред така, че от лявата страна да се намират само производните, а от дясната страна производните да не участват.
ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb – всички функции решават системата от обикновени диференциални уравнения y'=f(t,y);
ode15s, ode23s, ode23t и ode23tb решават системи от вида my'=f(t,y), като от последните с изключение на ode23s решават уравненията m(t)y'=f(t,y).
[t,y]=ode_function('f',tspan,yo) [t,y]=ode_function('f',tspan,yo,options)
[t,y]=ode_function('f',tspan,yo,options,p1,p2,...)
f е името на .m файла, който съдържа дясната страна на уравненията – функцията f(t,y) и връща вектор-стълб;
има вида:
function dydt = f(t,y)
където t p1,p2,... са незадължителни параметри, които се предават на f;
[t,y] е матрица на решението, където t е скалар, dydt и y са вектор-стълбове.
tspan е вектор, определящ интервала [t0 tfinal]; yo е вектор, определя началните условия;
options е незадължителен параметър, който се създава чрез функцията odeset;
p1,p2,... са незадължителни параметри, които се предават на f;
[t,y] е матрица на решението.

Полиноми

1. Нули на полином

r=roots(p)

p – вектор-ред, съдържа коефициентите на полинома в намаляващ ред

r – вектор-стълб, съдържа корените на полинома p

Пример:

```
» p=[1 -6 -72 -27];
» r=roots(p)
r = 12.1229
-5.7345
-0.3884
```

2. Коефициенти на полином

p=poly(r)

r – вектор-стълб, съдържа корените на полином

p – вектор-ред, съдържа коефициентите на полинома в намаляващ ред

Пример: Корените на полином са 12.1229, -5.7345 и -0.3884

```
» r=[12.1229 -5.7345 -0.3884]';
» p=poly(r)
p =
```

```
    1.0000   -6.0000        -72.0000      -27.0011
```

3. Стойност на полином

y=polyval(p,s)

p – вектор-ред с дължина n+1, съдържа коефициентите на полинома в намаляващ ред

s – вектор-ред, съдържа стойности, за които полиномът ще се изчислява

y –вектор-ред, съдържа изчислените стойности на полинома

Пример: Стойности на полинома при s = 1, 2 и 3

```
» p=[1 -6 -72 -27];
» s=[1 2 3];
» y=polyval(p,s)
 y =
-104    -187   -270
```

4. Апроксимиране с полином

p=polyfit(x,y,n)

x,y – данните, които се апроксимират

n – степен на полинома

p – вектор-ред с дължина n+1, съдържа коефициентите на полинома в намаляващ ред, който апроксимира p(x(i)) към y(i).

Пример: Апроксимиране с полином от степен 6 на данни от експеримент

```
» t=0:0.1:2.5;
» i=[0 0.1125 0.2227 0.3286 0.4284 0.5205 0.6039 ...
0.6778 0.7421 0.7969 0.8427 0.8802 0.9103 0.9340 ...
0.9523 0.9661 0.9763 0.9838 0.9891 0.9928 0.9953 ...
0.9970 0.9981 0.9989        0.9993 0.9996];
» p=polyfit(t,i,6)
» plot(t,i,'ro',t,polyval(p,t))
```

Резултати:

```
 p =
0.0084     -0.0985       0.4222        -0.7440       0.1475        1.1064
    0.0005
```

[r,p,k]=residue(num,den)

num – вектор-ред, съдържа коефициентите на полинома-числител в намаляващ ред

den – вектор-ред, съдържа коефициентите на полинома-знаменател в намаляващ ред

r – вектор-стълб, съдържа остатъците от частното на num и den

p – вектор-стълб, съдържа полюсите

k – вектор-ред, съдържа директните изрази

Пример за елементарна анимация

```
% Програма за въртене на тримерна фигура в
MATLAB
if ~exist('MovieGUIFlag'), figNumber=0; end; load logo
h=surfl(L,source);
colormap(M);
ax=[7 52 7 52 -.5 .8];
axis(ax);
axis on;
shading interp;
m=moviein(25);
for n=1:25,
rotate(h,[0 90],15,[21 21 0]);

h=surfl(get(h,'XData'),get(h,'YData'),get(h,'ZData'),(source);
axis(ax);
axis on;
shading interp;
m(:,n)=mvframe(figNumber,24);
end;
mvstore(figNumber,m);
```

Тази програма има два блока: в първия се задават изходната функция и нейния образ, а във втория, чрез цикъла, се създават кадрите и тяхното последователно възпроизвеждане, създаващо ефект на анимация.

Интерактивна система за помощ, която се реализира с помощта на следните команди:

```
» help
HELP topics:
matlab\general – General purpose commands.
Matlab\ops – Operators and special characters.
Matlab\lang – Programming language constructs.
Matlab\elmat – Elementary matrices and matrix
Matlab\elfun – Elementary math functions.
Matlab\specfun – Specialized math functions.
```

Справка по групи обекти в MATLAB

```
» help timefun
Time and dates
Current date and time
```

Now – Current date and time as date number
Date – Current date as date string
Clock – Current date and time as date vector
Basic functions
Datenum – Serial date number
Datestr – String representation of date
Datevec – Date components
Date functions
Calendar – Calendar
Weekday – Day of week
Eomday – End of month
Datetick – Date formatted tick labels
Timing functions
Cputime – CPU time in seconds
Tic – Start stopwatch timer
Toc – Stop stopwatch timer
Etime – Elapsed time
Pause – Wait in seconds

След уточняване на състава на определена група може да се получи детайлна справка за всеки избран обект. Приведените по-горе временни функции се използват за оценка на скоростта за изпълнение на отделни команди и изчисления.

Литература

1. Тончев, Й., Matlab 7. част1. Русенски университет. Русе. 2006. ISSN 957-712-226-6.
2. Тренчев, И., П. Миланов. Въведение в Matlab. Югозападен университет „Неофит Рилски“. Благоевград. 2007
3. Pietruszka,W.D.,Glocker Michael. Matlab und Simnulink in der Ingenieurpraxis. Springer Germany. 2021.ISBN 978-3-658-29740-4
4. Hoffman, Josef., Urban Brunner. Matlab and Tools. Addison Wesley. German. 2002. ISBN-13:978-3827318954
5. Eshkabilov, Sulaymon. Beginning Matlab and Simulink4. Hoffman. Springer Germany. ISBN 978-1-4842-5661-7, 2019
6. Moler, C.B. Numerical Computing with MATLAB. N. J.: Prentice Hall. 2002
7. Van Loan, C.F. Introduction to scientific computing: a matrix-vector approach using MATLAB. N. J. Prentice Hall. 2007
8. www.mathworks.com/products/matlab-online.html