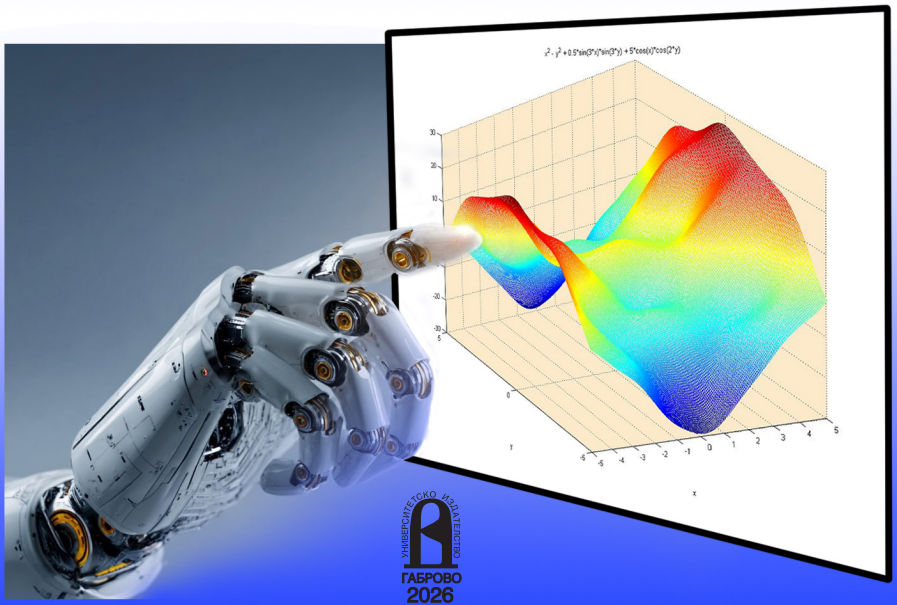
ЕЛЕНА МОНОВА

Симулиране и моделиране в MatLab



ГАБРОВО 2026

РЪКОВОДСТВО ЗА ЛАБОРАТОРНИ УПРАЖНЕНИЯ

Габрово 2026

Елена Монова

СИМУЛИРАНЕ И МОДЕЛИРАНЕ В *MATLAB*

ръководство за лабораторни упражнения

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
"ВАСИЛ АПРИЛОВ"
ГАБРОВО, 2026

СИМУЛИРАНЕ И МОДЕЛИРАНЕ В MATLAB -

Ръководство за лабораторни упражнени

Рецензент: доц.д-р инж. Станимир Йорданов Йорданов

ISBN: 978-954-683-742-4

Съдържание

Предговор

Настоящото ръководство за лабораторни упражнения по дисциплината „Симулиране и моделиране в *Matlab*“ е разработено като съпътстващо учебно пособие за студентите от специалност „Автоматика, роботика и компютърни управляващи системи“ при Технически университет – Габрово.

В него са включени лабораторни упражнения, насочени към усвояване на основните възможности на програмната среда *Matlab*. Разгледани са възможностите за генериране и обработка на вектори и матрици, извършване на числени и символни изчисления, графично представяне на данни и функции в двумерни координатни системи и в тримерното пространство, както и създаване и работа със script файлове и файл-функции.

В ръководството е включено и описание на възможностите на Simulink. Разгледани са библиотеките, основните блокове, изграждането на модели и са включени различни възможности за моделиране и симулиране на динамични системи в средата Simulink.

Ръководството включва още лабораторно упражнение, представящо вариант за разработване на потребителски графичен интерфейс в *Matlab* чрез *GUIDE*, както и лабораторно упражнение, описващо някои от възможностите на *Matlab* за числено и аналитично решаване на обикновени диференциални уравнения.

Лабораторно упражнение 1
Въведение в *Matlab*.
Числа, променливи и изрази

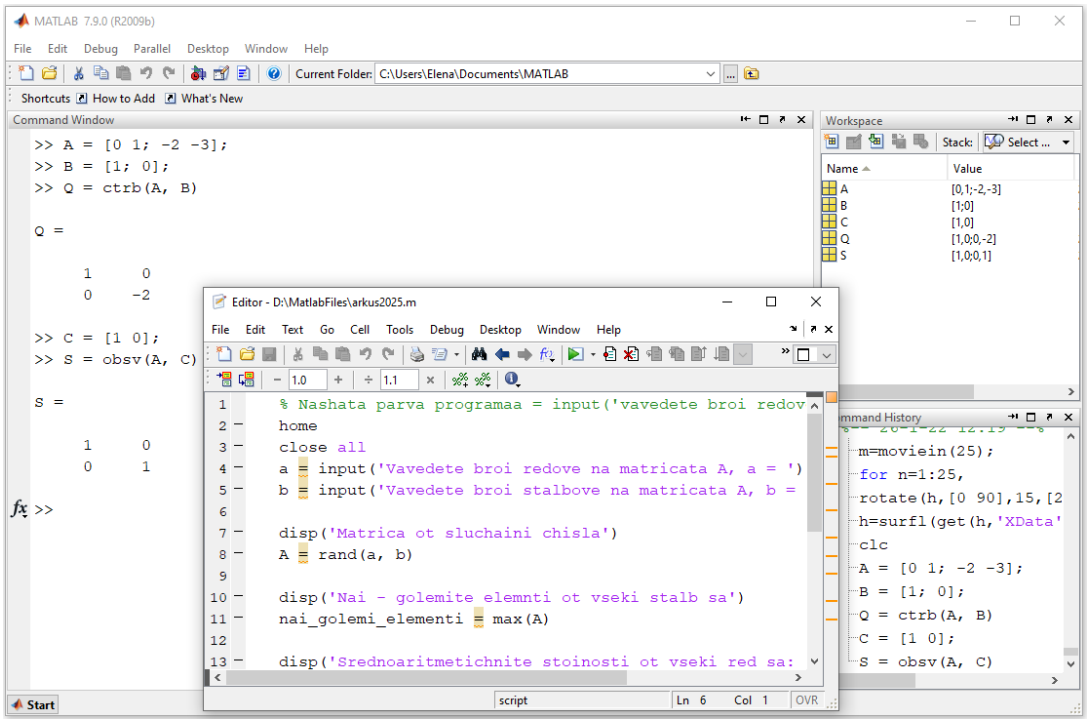
Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с програмния продукт *Matlab* – интерфейс, въвеждане на числа, променливи, изрази, действия с комплексни числа.

Въведение

Matlab (*MATrix LABoratory*) е програмен продукт от високо ниво, използван широко в инженерството, математиката, управлението на системи и анализа на данни. Продуктът обединява мощни средства за числени изчисления, визуализация, моделиране и симулация в единен и интуитивен интерфейс. Основните елементи на интерфейса включват командния прозорец, работното пространство, редактора на скриптове и функции, както и инструменти за визуализация на резултатите. *Matlab* позволява лесна работа с матрици и вектори, което го прави особено подходящ за решаване на линейни и нелинейни задачи. Има възможност за работа както с реални, така и с комплексни матрици, като могат да бъдат извършвани всички математически операции с матрици, вектори, масиви, скалари (събиране, изваждане, умножение, деление, степенуване, транспониране и т.н.). Чрез богат набор от вградени функции и допълнителни пакети (toolboxes), софтуерът може да се адаптира към разнообразни приложни области. Възможностите за двумерно и тримерно графично представяне на данни улесняват анализа и интерпретацията на резултатите. Поради своята гъвкавост и разширяемост *Matlab* е предпочитан инструмент както в образованието, така и в индустриалната практика.

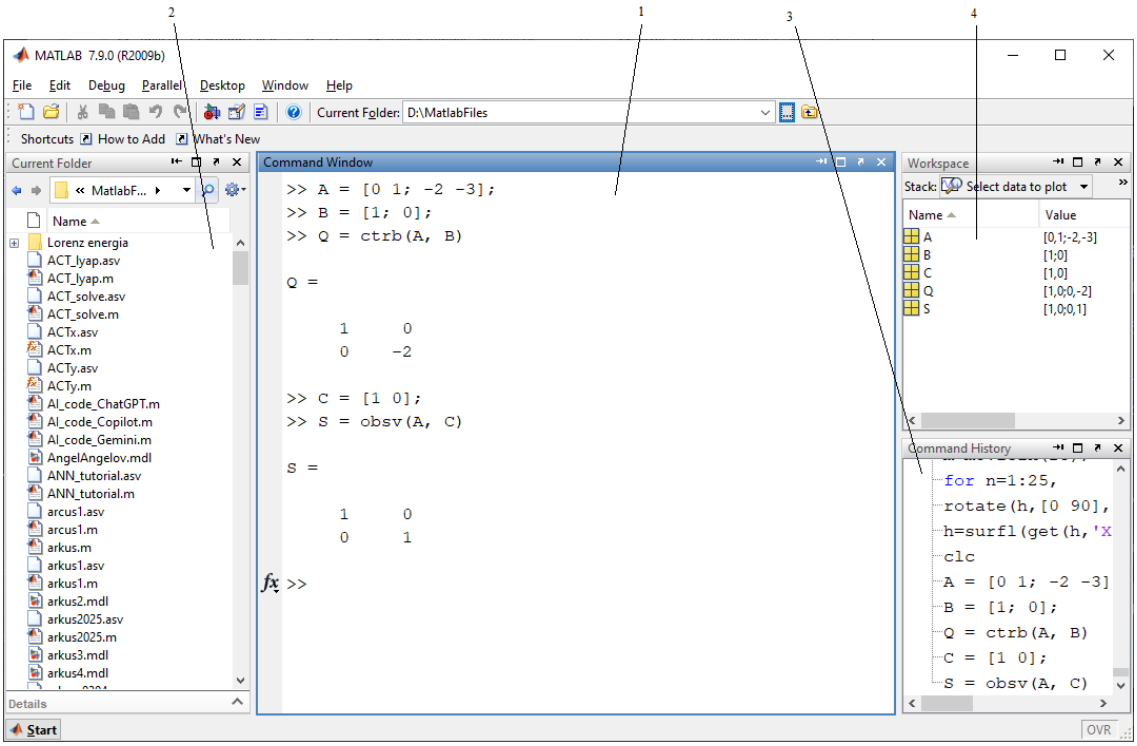
Matlab има вграден програмен език от високо ниво, позволяващ работа на системата не само в режим калкулатор, а и в програмен режим.



фиг. 1.1

Matlab работи в два режима – команден и програмен. Изпълнението на една *Matlab* програма се извършва чрез извикване с команда, представляваща нейното име.

Интерфейс на програмния продукт Matlab



фиг. 1.2

Интерфейсът на *Matlab* се състои от 4 прозореца (фиг.1.2):

- Команден прозорец (*Command Window*) (1);

- Текуща директория (*Current Folder*) (2).

- История на командите (*Command History*) (3);

- Работно пространство (*Workspace*) (4);

Най-често използваните прозорци са **Command Window** и **Command History**.

Командният прозорец (**Command Window**) е основния прозорец на *Matlab*, в който след символа „>>” се въвеждат желаните команди. Числените резултати се визуализират в същия прозорец след натискане на *„Enter”*, а графичните резултати се извеждат в отделен прозорец.

При въвеждането на командите са в сила следните правила:

- Когато след дадена команда бъде поставен символа „;”, резултатът от изпълнението й не се извежда.

- Когато след командата липсва символа „;”,полученият от изпълнението й резултат се извежда в командния прозорец и може да бъде видян.

- Резултатите от изпълнението на дадена команда са едни и същи, независимо дали в края на командния ред има или няма символ „;”,.

- Ако се пресмята някакъв израз, без резултатът да се присвоява на определена променлива, *Matlab* автоматично го присвоява на системната променлива “*ans*” (от *answer*).

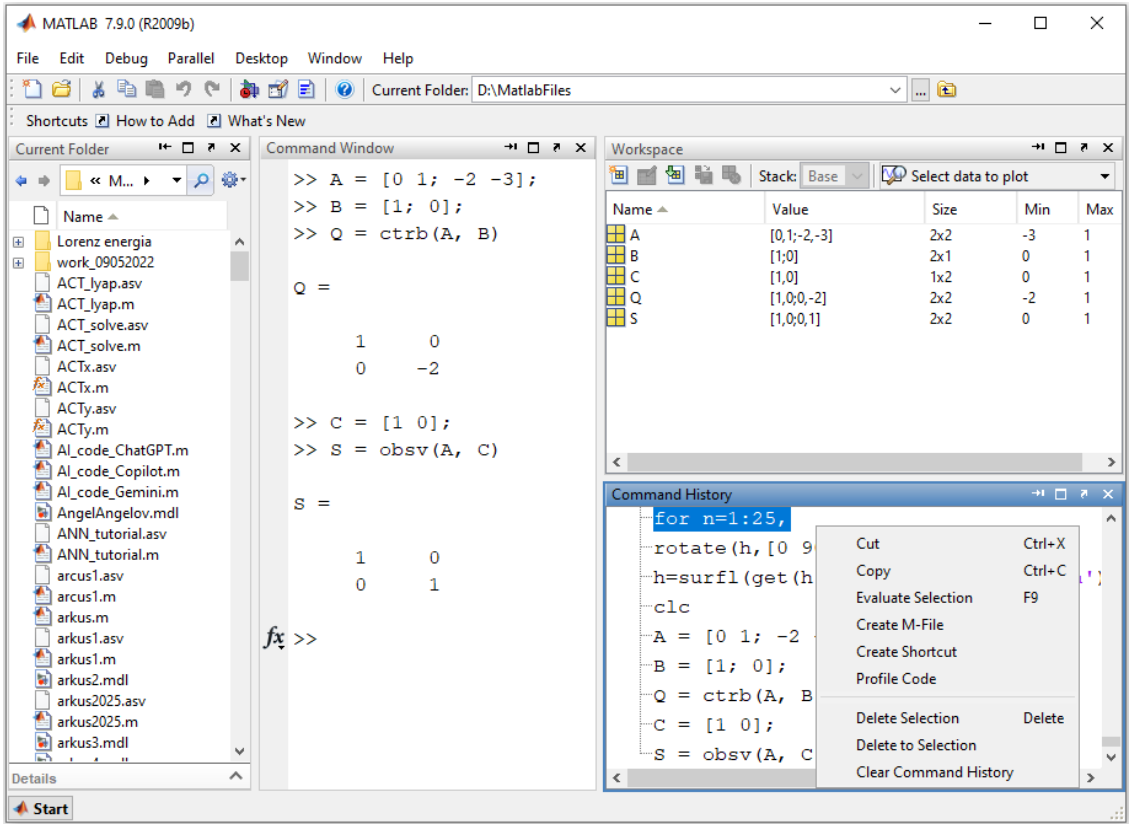
- Въведените в дадена работна сесия команди се запаметяват от системата и могат да се извикват на текущия команден ред посредством стрелка нагоре, след което могат да се правят промени и отново да се изпълняват, натискайки “*Enter”*.

- На един ред могат да бъдат въведени повече от една команди, като разделянето им може да бъде с един от символите “,” или “;”. В първия случай резултатите от изпълнението на командите се извеждат в командния прозорец, а във втория – командите се изпълняват, без да се визуализират получените резултати.

- За начало на коментар се използва символ „%”.

История на командите (**Command History**)

Matlab запаметява всички въведени команди както от текущата, така и от предишни сесии. Тези команди се извеждат и могат да бъдат прегледани в прозореца *Command History*. Върху тях могат да се извършват различни операции:



фиг. 1.3

- изпълнение на дадена команда от *Command History* чрез

  - двукратно щракане върху командата, при което тя се копира и изпълнява в командния прозорец (*Command Window*);
  - провлачване на командата до командния прозорец и натискане на „*Enter*”.

- при щракане с десен бутон върху команда (или последователност от команди) в *Command History*, се появява контекстно меню, в което има следните възможности за избор:

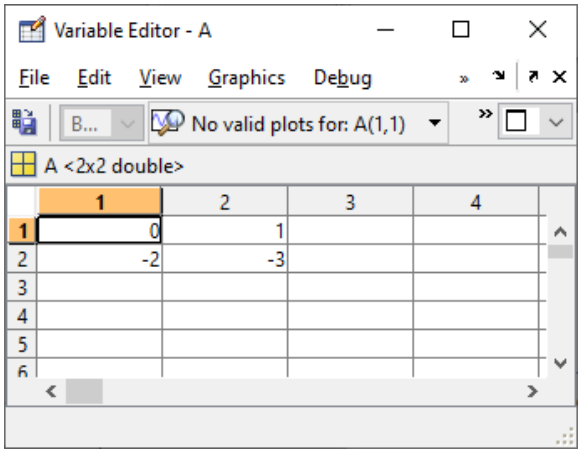
  - *Cut*
  - *Copy*
  - *Evaluate Selection* – копира избраните команди в командния прозорец и ги изпълнява;

- *Create M-File* – отваря с *Matlab* редактора нов m-файл и копира в него избраните команди;
- *Delete Selection* – изтрива избраните команди;
- *Delete to Selection* – изтрива всички команди преди първата маркирана;
- *Delete Entire History* – изтрива всички команди, включително и немаркираните.

Работно пространство (**Workspace**) - всички променливи, които са дефинирани в програмния продукт по време на текущата сесия, се съхраняват в така нареченото работно пространство (*Workspace*). В прозореца Workspace се извежда пълна информация за тези променливи, като: име, размерност, големина и клас.

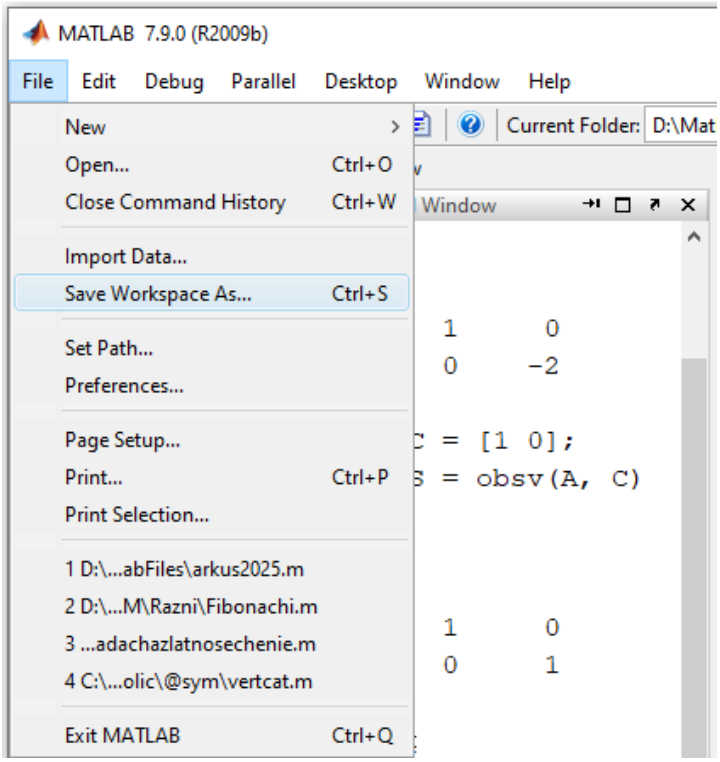
Особеното в *Matlab* е, че дори и скаларните величини се разглеждат като матрици, чиято размерност е 1x1 (1 ред и 1 стълб).

Чрез двукратно щракане върху името на масива се извиква *Variable Editor*, след което могат да се изтриват променливи или да се коригират стойности на елементите от дадения масив.



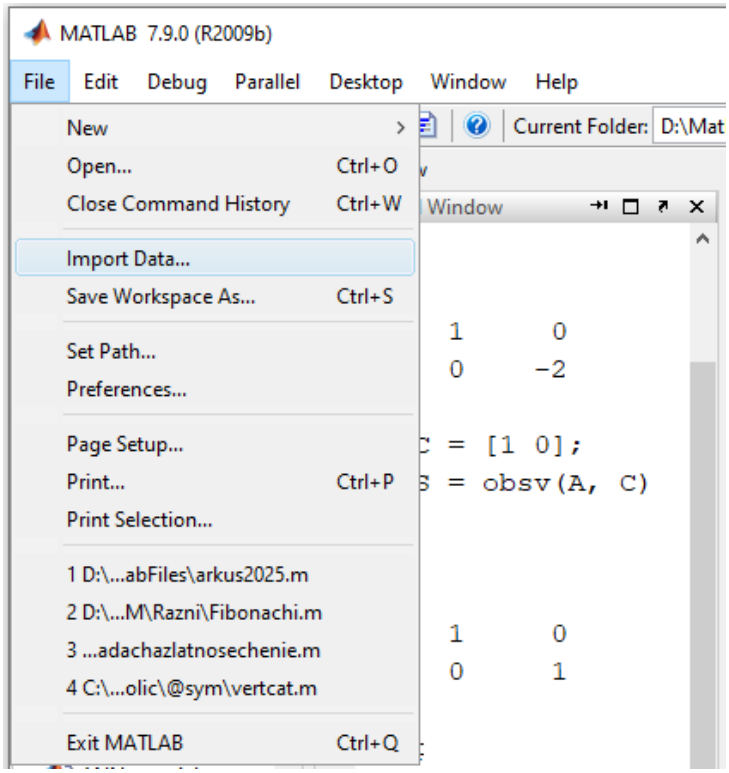
фиг. 1.4

Информацията от работното пространство (т.е. всички дефинирани в текущата сесия променливи) може да бъде записана във файл с разширение .mat. Това става като изберем от менюто *File–>Save Workspace As…*



фиг. 1.5

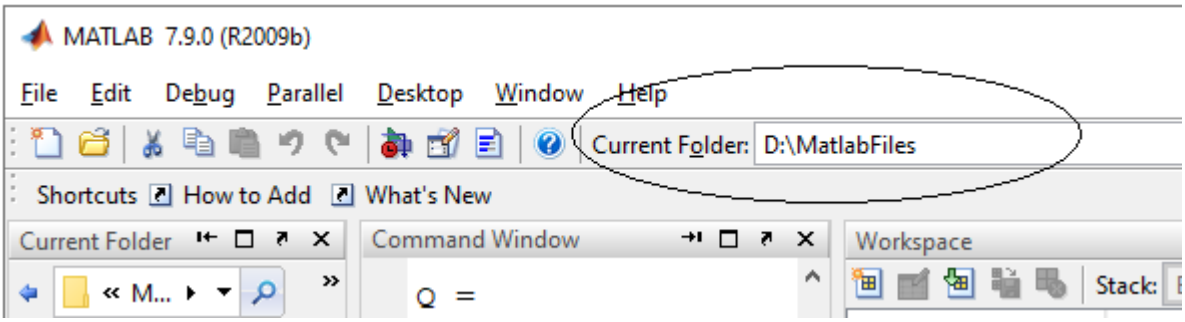
Така запазените данни могат да се използват в следваща сесия. Извикването на .mat файла и зареждането на данните от него става с помощта на командата *File–>Import Data* .



фиг. 1.6

Информация за работното пространство, коригиране на променливите и техните стойности, както и запазването на данните във файл, може да става и от командния прозорец.

Текуща директория (**Current Folder**) - в този прозорец се извеждат всички файлове от текущата директория. До тази директория системата има директен достъп, дори и ако не е обявена за достъпна с командата path. В нея обикновено се записват и създаваните от потребителите файлове. Коя директория е зададена като текуща може да бъде видяно в горния край на интерфейсния прозорец. (фиг.1.7)



фиг. 1.7

Синтаксис на командите в програмния продукт Matlab

В *Matlab* използваните формати на командите са два – функционален и команден.

- *Функционален формат (Function format)*

В този формат командата се състои от името на функцията, след което в кръгли скоби се задава *входящ аргумент*. Ако входящите аргументи са повече от един, разделянето им става чрез запетая.

Function_name (argument1, argument2, ...)

Пример за команда, записана в този формат е

sin(1),

която намира и извежда стойността на функцията *sinus* от 1.

Резултатът от изпълнението на функцията може да се присвои на една или няколко променливи, наречени *изходящи аргументи*. Те се разделят със запетаи, но се заграждат с квадратни скоби.

[out1, out2, ...] = Function_name (argument1, argument2, ...)

Пример за използването на изходящи аргументи е намирането на решението на $\sqrt{729}$ и присвояване на резултата на променлива *x*.

x = sqrt(729)

- *Команден формат (Command format)*

В този формат командата се състои от името на функцията, след което се задават *входящите аргументи, разделени чрез интервали*.

Function_name argument1 argument2 …

При този формат няма изходящи аргументи, т.е. резултатът от изпълнението на функцията не може да се присвоява на дадена променлива.

Пример за команда, записана в този формат е дефинирането на символни променливи в *Matlab*, което се записва като:

syms fun1 matricaA

Системни команди в Matlab

Най-често използваните системни команди в *Matlab* са:

- *clear* – изтрива променливите от работното пространство;
  - *clear var1 var2 …* - изтрива посочените променливи.
- *clc* – изчиства командния прозорец, като изтрива всички команди;
- *home* – изчиства видимата част от командния прозорец, чрез скролирането му. Ако командният прозорец не е бил запълнен не настъпват никакви промени;
- *format* – настройва формата на изходните данни;

Пресмятанията в *Matlab* се извършват с двойна точност (double precision), т.е. с около 15 значещи цифри. Резултатите обаче могат да се изобразяват в различен формат, който се определя от командата *format*.

- *format short* – фиксирана точка (десетична запетая) с 4 цифри след точката;
- *format long* - фиксирана точка (десетична запетая) с 15 цифри;
- *format short e* – плаваща точка с 5 цифри – експоненциална форма;
- *format long e* - плаваща точка с 15 цифри;

Получаване на Help информация

Matlab предлага три начина за получаване на помощна информация:

► от *m*-файлове – при въвеждане на команда *help file_name* се извеждат всички коментари, намиращи се в началото на посочения файл до първата команда или до първия празен ред;

► фирмената документация в *html* или *pdf* формат;

► WEB сайта на фирмата Math Works.

Достъпът до информацията може да се осъществи по няколко начина:

► От главното меню Help – *Help->Using The Command Window*;

► Чрез бутон „?";

► От командния прозорец с помощта на командите:

- *help* – извежда информация за съответната команда в командния прозорец;
- *helpwin* – информацията от m-файла се извежда в Help Browser;
- *doc* – информация от документацията се извежда в Help Browser;

Задача: Изведете информация за командата *cos* по два начина – в командния прозорец и в Help Browser.

Демонстрационни задачи - Demos

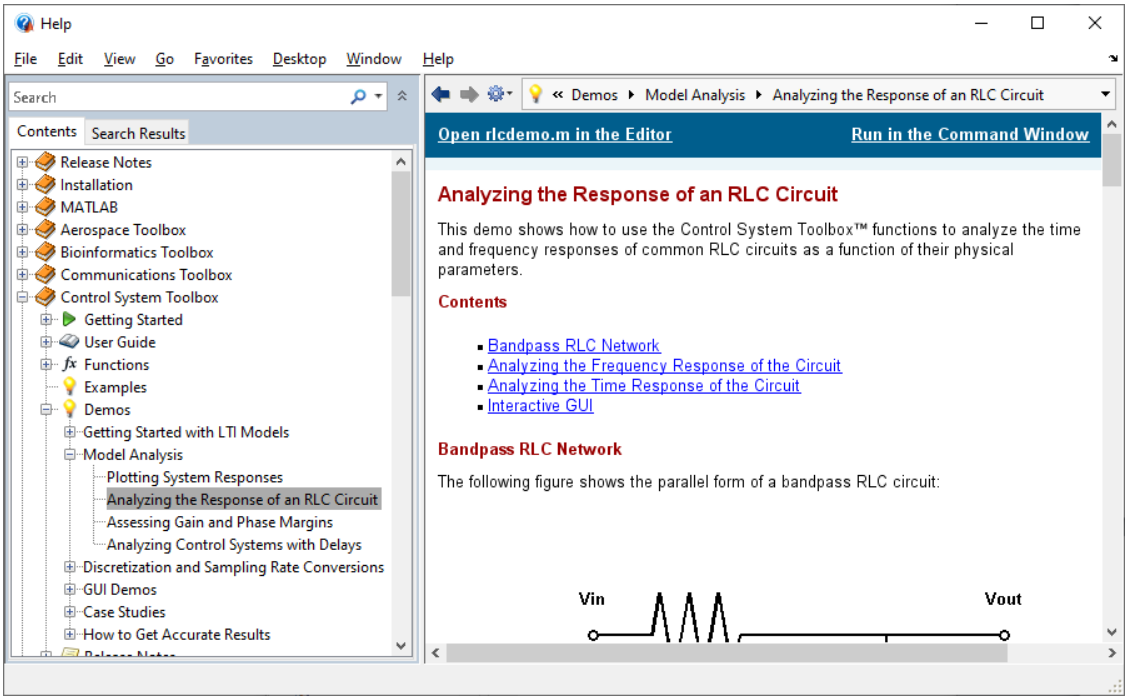
Matlab предлага голям набор от демонстрационни задачи, показващи възможностите както на ядрото на *Matlab*, така и на различните Toolbox-ове. Достъпът до тези задачи може да се осъществи по следните начини:

► От главното меню – *Help->Demos*;

► На командния ред в Command Window се въвежда командата *demo*;

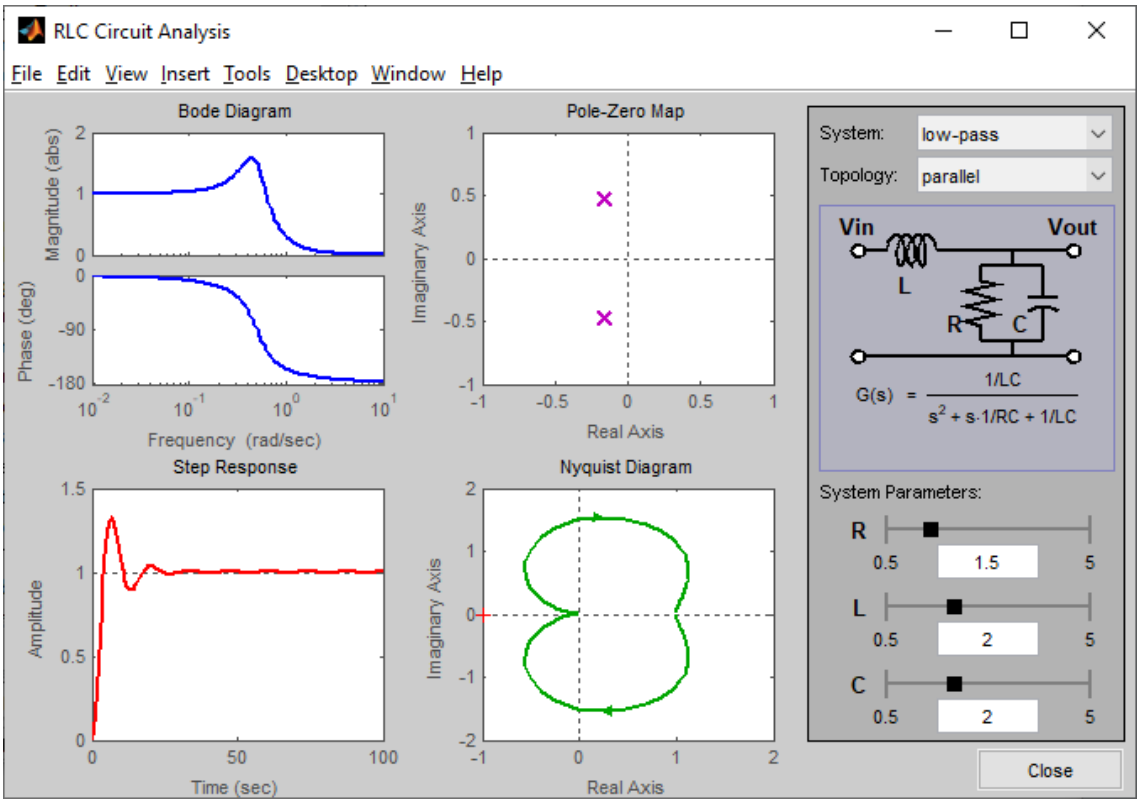
Появява се прозорец, от който се избира желаната тема от *Matlab* или съответния *Toolbox*, задачата и след това демонстрационната задача се стартира от *Run this demo*.

Задача: С помощта на един от посочените начини отворете прозореца с *demo* примерите. От списъка в ляво изберете *Control System Toolbox/ Demos/ Model Analysis/ Analyzing the Response of an RLC Circuit*.



фиг. 1.8

След отваряне на прозореца, стартирайте примера чрез *Run in the Command Window* (горе вдясно) или изберете *Interactive GUI*.



фиг. 1.9

Избраният пример се стартира, след което в полетата за параметрите *R*, *L* и *C* могат да бъдат променяни стойностите им, а в полетата System и

Topology могат да бъдат избирани различни варианти от падащите менюта (фиг.1.9).

Променливи, числа, изрази

При използване на променливи в *Matlab* не е необходимо предварително да се декларира техния тип и размерност, но има изисквания към имената им. Имената на променливите в *Matlab* могат да съдържат малки, големи букви, числа и знакът „_”. Правилата, на които също така трябва да отговарят, са няколко:

- да започват винаги с буква, след която могат да присъстват всички гореспоменати символи;

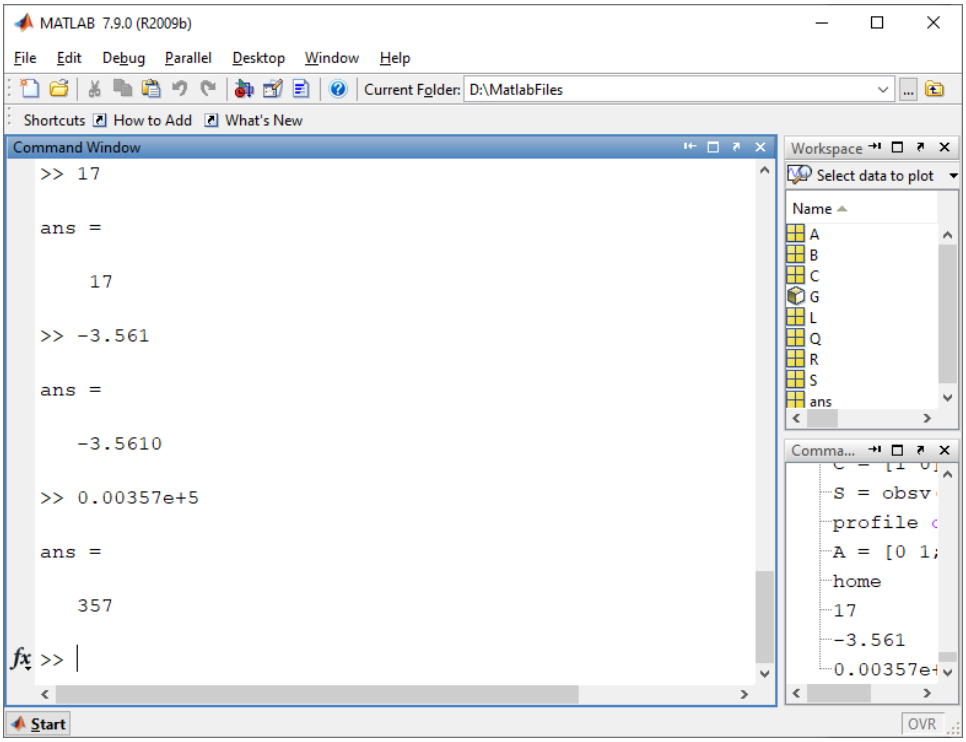
- няма ограничение за дължината на името, но системата различава само първите 31 символа;

- *Matlab* различава малки и големи букви, т.е. var1 и Var1 са имена на две различни променливи;

- имената на променливите не трябва да дублират имена на системни команди и функции в *Matlab*.

Въвеждането на числа в *Matlab* става по следния начин:

► цели числа – 3, 17, -63, -39, 175;



фиг. 1.10

► десетични числа – 31.5, 7.639, -1.33, 0.71. Последното десетично число може да бъде записано по още един начин, изпускайки нулата и записвайки само .71.

► числа в експоненциална форма – 3.675e+2, 10.6e-3, 0.0731e5.

Операторите, записвани в *Matlab*, имат вида

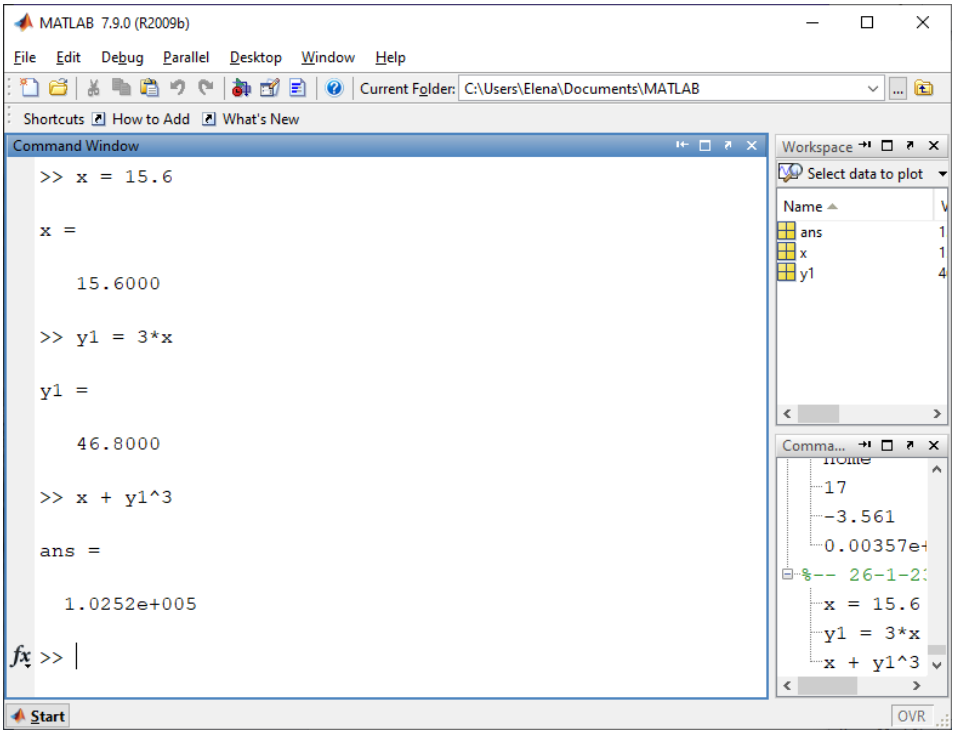
променлива = израз

или само

израз

Изразите се състоят от знаци за операции, имена на променливи, команди.

Въвеждането на изразите завършва с натискане на клавиш „ENTER”.



фиг. 1.11

Задача: Дефинирайте в *Matlab* следните променливи:

1) X1 = 3,75

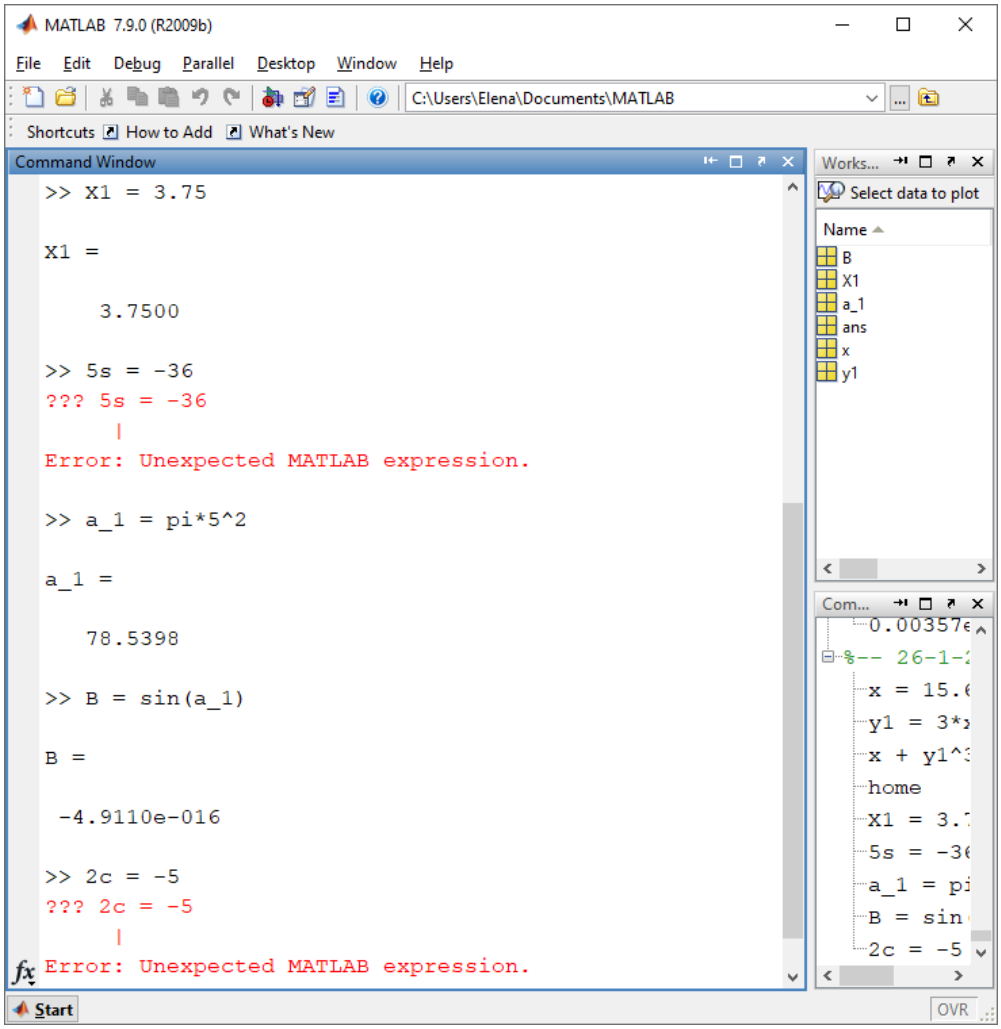
2) 5s = -36

3) a_1 = pi*5^2

4) B = sin(a_1)

5) 2c = -5

Какъв е резултатът от извършените действия?



фиг. 1.12

В *Matlab* се съхранява информация за няколко системни константи:

► *pi* – числото π, което се изчислява от *Matlab* по два начина начина – като 4*atan(1) и imag(log(-1));

► *i* или *j* – имагинерна единица;

► *realmin* – минималното реално число;

► *realmax* - максималното реално число;

► *Inf* – безкрайност (резултат от действия с деление на 0);

► *NaN (Not-a-Number)* – този резултат се връща от системата в случаите когато при пресмятането на даден израз се получи неопределеност от вида 0/0, ∞-∞.

Математически оператори

В изразите, записвани в *Matlab*, могат да се използват познатите аритметични оператори:

- събиране „+”;

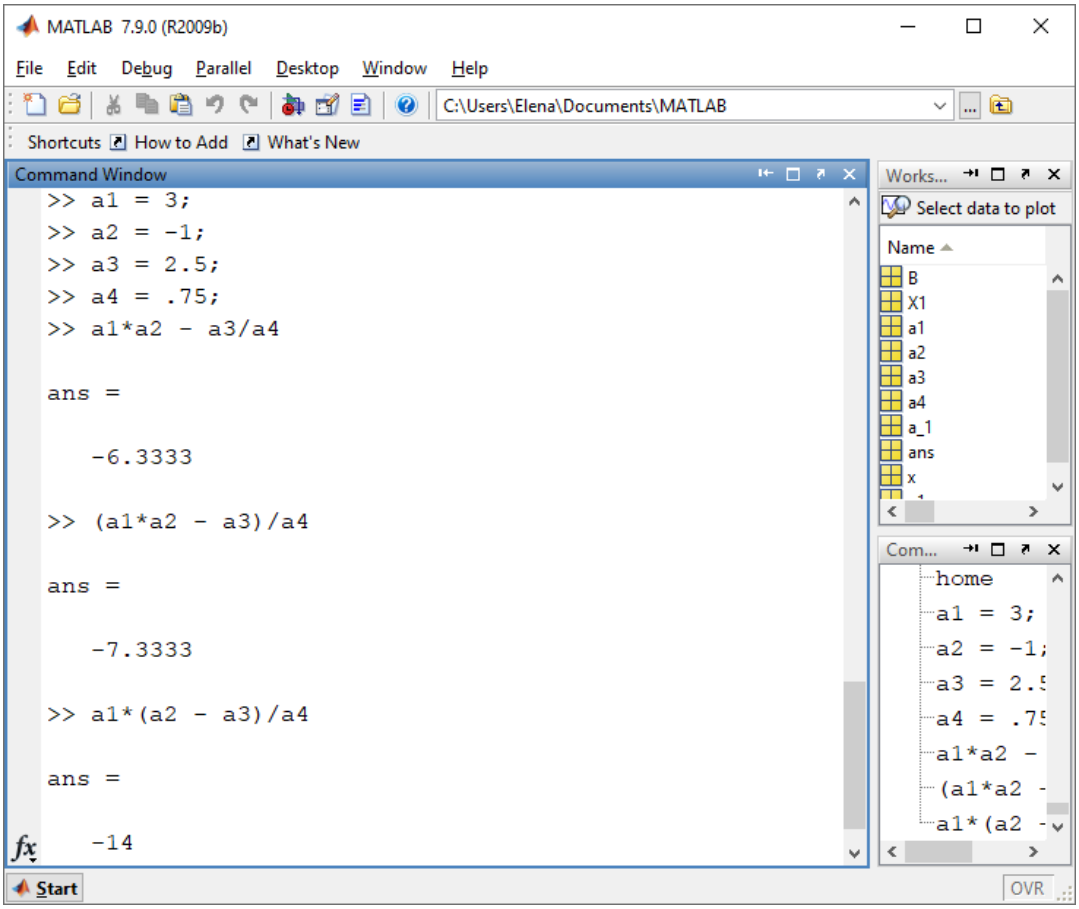
- изваждане „-”;

- умножение „*”;

- деление от дясно и деление от ляво „/” и „\”;

- степенуване „^”.

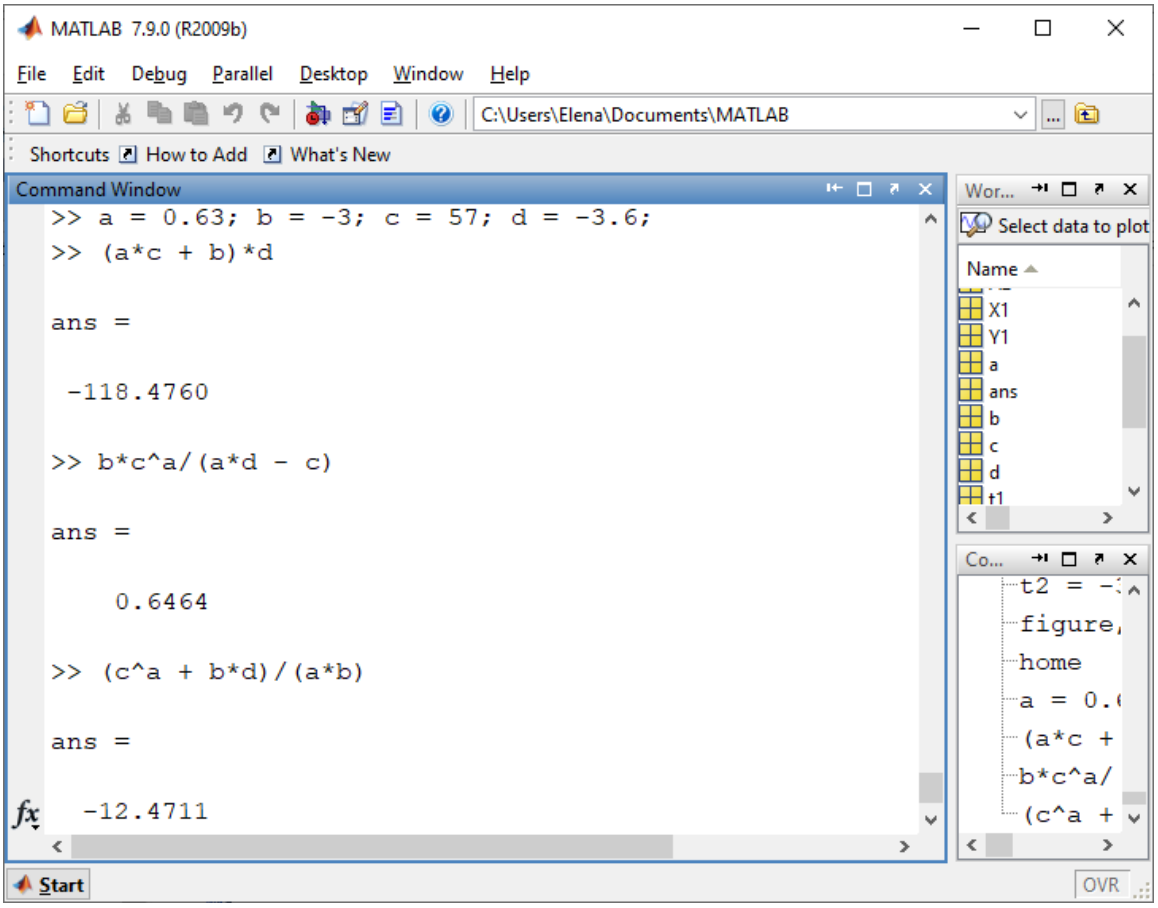
Последователността за извършване на математическите действия в даден израз се извършва по всички правила на математиката, като тази последователност може да се променя чрез кръглите скоби “( )”.



фиг. 1.13

Задача: Дефинирайте променливи a = 0.63, b = -3, c = 57 и d = -3.6 и извършете следните действия:

1) $(ac+b)d$; 2) $\frac{bc^a}{ad-c}$; 3) $\frac{c^a+bd}{ab}$



фиг. 1.14

Освен тези аритметични операции, в *Matlab* има възможност за извършване на много други математически действия, които могат да бъдат разгледани в библиотека *elfun*.

Комплексни числа

Имагинерната единица на дадено комплексно число в *Matlab* може да се зададе по два начина – с i и j.

Въвеждането на комплексни числа в *Matlab* може да бъде извършено по няколко начина:

- z = 5 + 6i или z = 5 + 6j
- z = 5 + 6*i или z = 5 + 6*j
- z = 5 + i*6 или z = 5 + j*6
- z = complex(5, 6)

Действията с комплексните числа се извършват с обикновените аритметични оператори : +, -, *, /, \, ^, а информация за вградените в *Matlab* функции за работа с комплексни числа вижте с команда

>> help elfun

Някои от вградените в *Matlab* функции, предназначени за работа с комплексни числа, са:

► *abs* – връща като резултат модула на комплексното число, зададено като аргумент на функцията;

► *angle* – аргумент на комплексното число;

► *conj* – комплексно спрегнато;

► *real* – реалната част на комплексното число;

► *imag* – имагинерната част на комплексното число;

Задача: Въведете в *Matlab* следните комплексни числа:

1) $z_1 = 3 - j11;$ 2) $z_2 = -11 + j25;$ 3) $z_3 = 6 + j7.$

Извършете следните действия с тях:

- намерете сбора и разликата на трите числа;

- намерете комплексно спрегнатото на z_1;

- намерете модула на z_2;

- намерете аргумента на z_3.

Въпроси и задачи за изпълнение:

1. С коя команда в *Matlab* може да се настрои форматът на данните?

2. С коя команда в *Matlab* може да се намери модулът на дадено комплексно число?

3. Въведете в *Matlab* произволни стойности за променливи *x*, *y*, *p* и *s* (може да използвате цели и дробни числа, положителни и отрицателни, реални и комплексни). Извършете следните действия:

- $f_1 = \cos\left(\frac{\pi}{2}\right) + \frac{xy}{s};$
- $f_2 = \frac{py}{\sin\left(\frac{\pi}{s}\right)};$
- $f_3 = \frac{p^3 + sx}{\sqrt{y + x}}.$

Лабораторно упражнение 2
Генериране и поелементно въвеждане на вектори и матрици в *Matlab*. Действия с елементите на матрици, матрични функции. Полиноми

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е придобиване на знания за начините за въвеждане на вектори и матрици в *Matlab*, запознаване с възможностите за генериране на различни видове вектори и матрици чрез вградените команди и извършване на различни действия, както с целите матрици, така и с отделните им елементи.

Вектори и матрици в Matlab

В *Matlab* всички данни се разглеждат като матрици. Това е така дори когато разглежданите данни са с размерност *px1* или *1xp* (вектори), или дори 1x1, т.е. скалари.

Въвеждане на вектори и матрици в Matlab

При поелементно въвеждане на вектори и матрици са в сила следните правила:

► въвеждането на елементите започва с отваряща квадратна скоба “[” и завършва със затваряща - ”]”;

► отделните елементи от даден ред на съответната матрица или на вектор – ред се разделят помежду си чрез интервал или запетая;

► отделните редове на дадена матрица или елементите на вектор – стълб се разделят помежду си чрез точка и запетая “;”;

► вместо точка и запетая, може всеки отделен ред на матрицата или всеки елемент на вектора-стълб да се записват на нов ред.

Задача: Въведете в командния прозорец на *Matlab* матрица A по три различни начина

$$A = \begin{bmatrix} 1 & 6{,}7 & -5 \\ 0{,}63 & 9 & 15 \\ -3 & 0 & 33 \end{bmatrix}$$

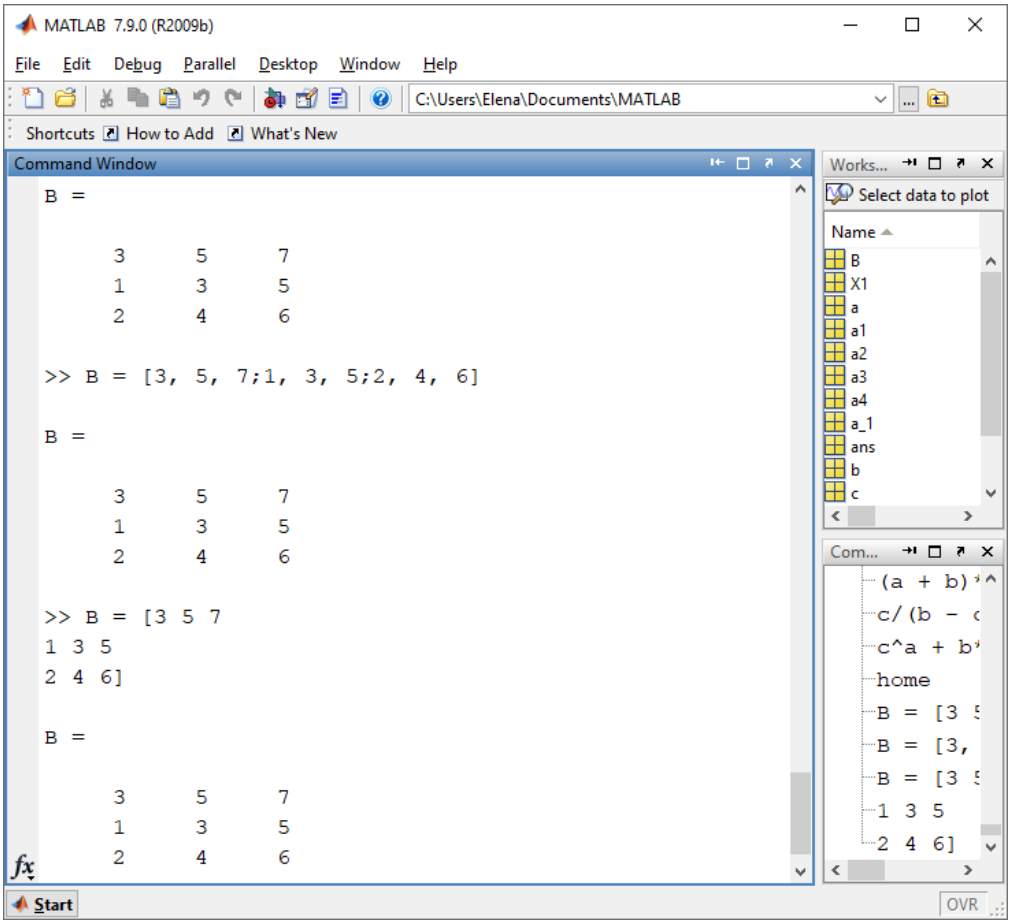
Упътване:

► въведете матрица A, като за разделители между елементите използвате интервали;

► въведете матрица A, като за разделители между елементите използвате запетаи;

► въведете матрица A, като всеки ред на матрицата записвате на нов ред в командния прозорец.

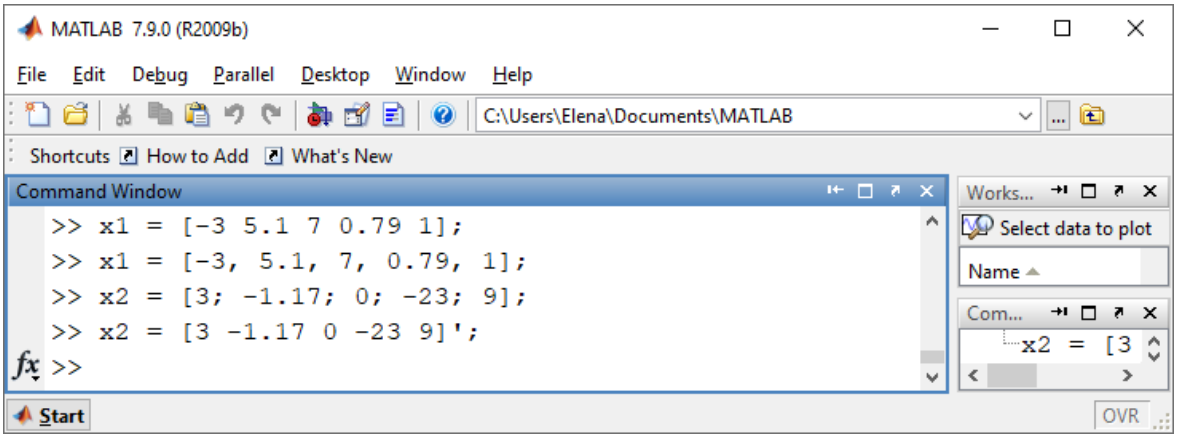
Какви са получените резултати от трите начина на въвеждане на матрицата?



фиг. 2.1

 Задача: Въведете в командния прозорец на *Matlab* векторите x_1 и x_2 по два различни начина.

$$x_1 = \begin{bmatrix} -3 & 5{,}1 & 7 & 0{,}79 & 1 \end{bmatrix} \text{ и } \quad x_2 = \begin{bmatrix} 3 \\ -1{,}17 \\ 0 \\ -23 \\ 9 \end{bmatrix}.$$



фиг. 2.2

Генериране на вектори и матрици

В *Matlab* има отделни функции за генериране на определен вид вектори и матрици. Най-често използваните от тях са:

► *zeros(n, m)* – с помощта на командата се генерира матрица от нули с размерност *n*x*m (n реда и m стълба)*;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* матрица от нули Z1 с размерност 3 реда и 5 стълба.

► *ones(n, m)* – използва се за генериране на матрица от единици с размерност *n*x*m*;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* матрица от единици E1 с размерност 4 реда и 5 стълба.

► *eye(n, m)* – генерира единична матрица (единици по главния диагонал и нулеви елементи извън него) с размерност *n*x*m*;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* единична матрица I1 с размерност 3 реда и 3 стълба.

► *rand(n, m)* – генерира матрица от случайни числа със стойности в интервала от 0 до 1 с размерност *n*x*m*;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* матрица от случайни числа R с размерност 4 реда и 3 стълба.

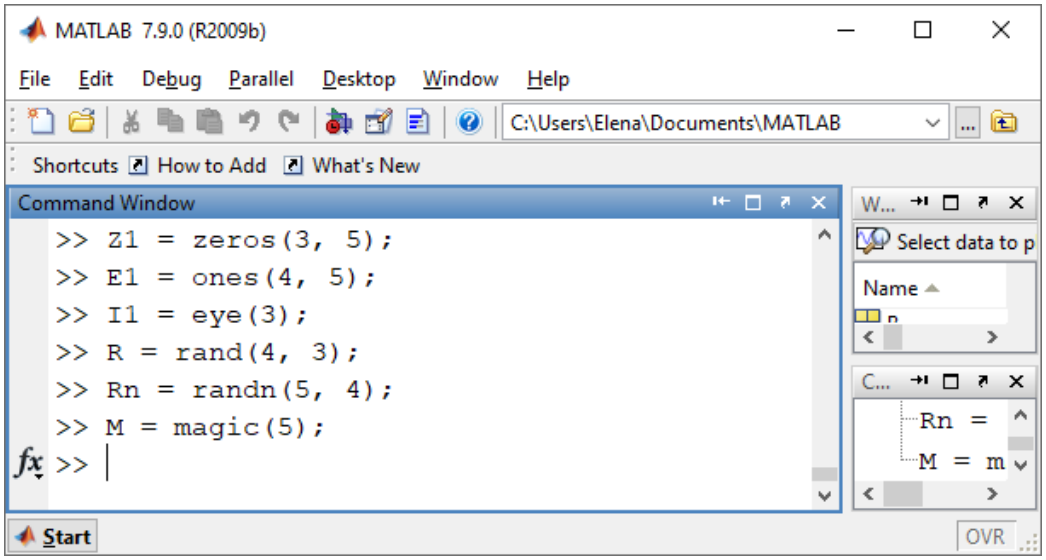
► *randn(n, m)* – генерира матрица от случайни числа, разпределени по нормален закон, с размерност *n*x*m*;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* матрица от случайни числа, разпределени по нормален закон, Rn с размерност 5 реда и 4 стълба.

► *magic(n)* – с помощта на командата се генерира квадратна „магическа“ матрица, за която сумата от елементите на всеки ред, стълб и диагонал е една и съща. Командата приема само един аргумент, т.к. генерираната матрица винаги е квадратна (еднакъв брой редове и стълбове).

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* магически квадрат M с размерност 5 реда и 5 стълба.

!!! В случай, че като аргумент на която и да е от изброените функции бъде подадено само едно число, то резултатът ще бъде квадратна матрица със зададената размерност.



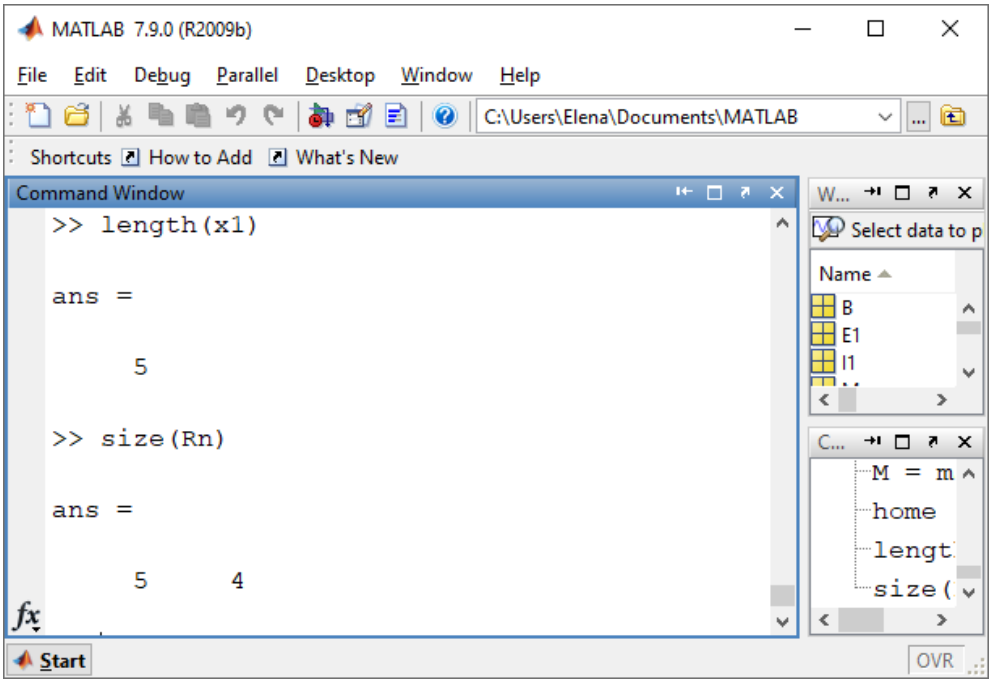
фиг. 2.3

В *Matlab* има команди, с които да бъде определена размерността на даден вектор или матрица. Това са командите:

► *length(x)* – връща като резултат дължината на вектор x;

► *size(A)* – връща като резултат размерността на матрица A;

🤔 ***Задача:*** Да се определи размерността на вектор x_1 и на матрица Rn.



фиг. 2.4

Операции с вектори и матрици

В *Matlab* има дълъг списък с команди, свързани с действия с вектори и матрици (част от които вече разгледахме). Всички те могат да се видят с командата *help elmat.*

>> help elmat

Преобразуване на вектори и матрици

Част от командите, които *Matlab* предлага за преобразуване на вектори и матрици, са:

► *fliplr(A)* – връща огледален образ на матрица *A* спрямо вертикална ос (fliplr – Left Right);

🤔 ***Задача:*** Намерете огледалния образ спрямо вертикалната ос на матрица I1.

► *flipud(A)* – връща огледален образ на матрица *A* спрямо хоризонтална ос (flipud – Up Down);

🤔 ***Задача:*** Намерете огледалния образ спрямо хоризонталната ос на матрица R.

► *rot90(A)* – завърта матрица *A* обратно на часовниковата стрелка на 90 градуса;

🤔 ***Задача:*** Намерете матрица M2, която представлява обърнатата на 180 градуса матрица M.

► tril(A) – връща долната триъгълна част на матрица A;

🤔 ***Задача:*** Изведете в командния прозорец на *Matlab* долната триъгълна част на матрица R.

► triu(A) – връща горната триъгълна част на матрица A;

🤔 ***Задача:*** Изведете в командния прозорец на *Matlab* горната триъгълна част на матрица Rn.

► diag(A) – създава вектор-стълб с елементите от главния диагонал на матрица A;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* вектор v_1, който да съдържа елементите от главния диагонал на матрица Rn.

► diag(A, n) – създава вектор-стълб с елементите от n-тия диагонал над главния диагонал на матрица A;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* вектор v_2, който да съдържа елементите от втория диагонал над главния на матрица R.

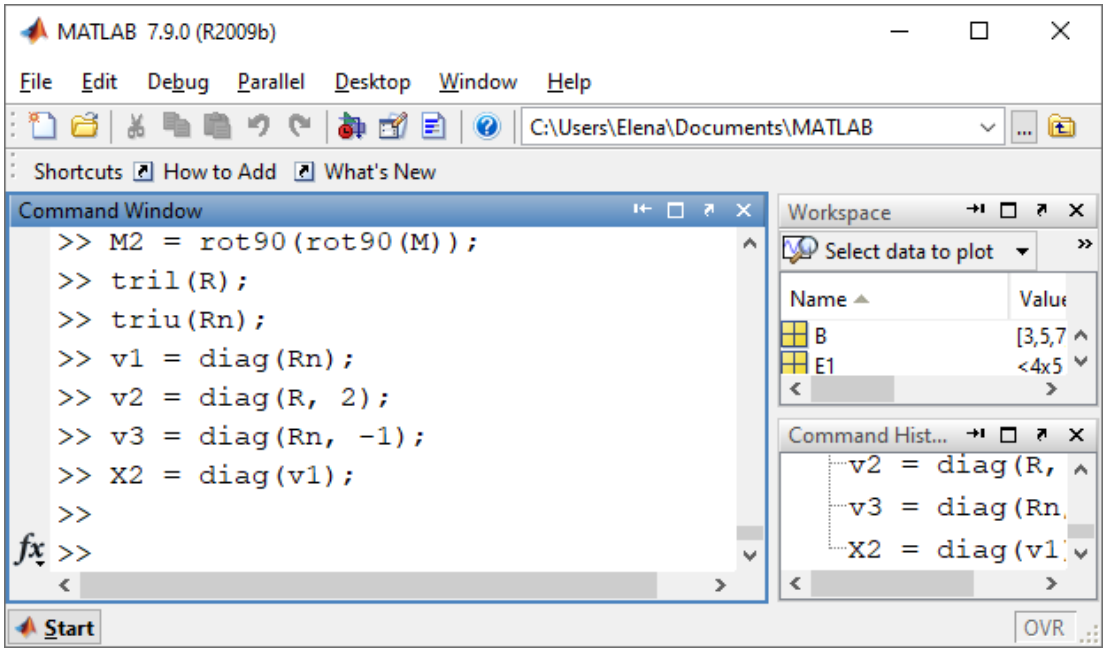
► diag(A, -n) – създава вектор-стълб с елементите от n-тия диагонал под главния диагонал на матрица A;

🤔 ***Задача:*** Създайте в командния прозорец на *Matlab* вектор v_3, който да съдържа елементите от първия диагонал под главния на матрица Rn.

!!! Ако на функцията diag като аргумент се подаде вектор, то резултатът ще бъде матрица, която по зададения от функцията диагонал ще съдържа елементите на вектора, а останалите елементи на матрицата ще са нули.

► diag(x) – създава матрица, главният диагонал на която съдържа елементите на вектор x;

Аналогично е действието и в останалите варианти на функцията, т.е. с диагонал, различен от главния (diag(x, n) и diag(x, -n)).



фиг. 2.5

Извличане, вмъкване и премахване на елементи (части) от матрица

В *Matlab* е възможно обръщане към всеки отделен елемент, ред, стълб или блок на дадена матрица. Възможностите са:

► A(i, j) – извлича елемента a_{ij} от матрица *A* (елементът, намиращ се на *i*- тия ред и *j*-тия стълб на матрицата A);

Задача: Изведете в командния прозорец на *Matlab* елемента от четвъртия ред, трети стълб на матрица Rn.

► A(i, j) = 3*pi - присвоява стойността 3*pi на елемента a_{ij} от матрица *A*;

Задача: Променете елемента от третия ред, първи стълб на матрица Rn, като му зададете стойност 15.

► x = A(i, :) – връща вектор-ред ***x***, съдържащ елементите от i-тия ред на матрица A;

!!! Символът „:” означава всички елементи.

Задача: Изведете в командния прозорец на *Matlab* третия ред на матрица R.

► y = A(:, j) – връща вектор-стълб ***y***, съдържащ елементите от j -тия стълб на матрица A;

Задача: Изведете в командния прозорец на *Matlab* втория стълб на матрица R.

► Извеждане на части от матрица

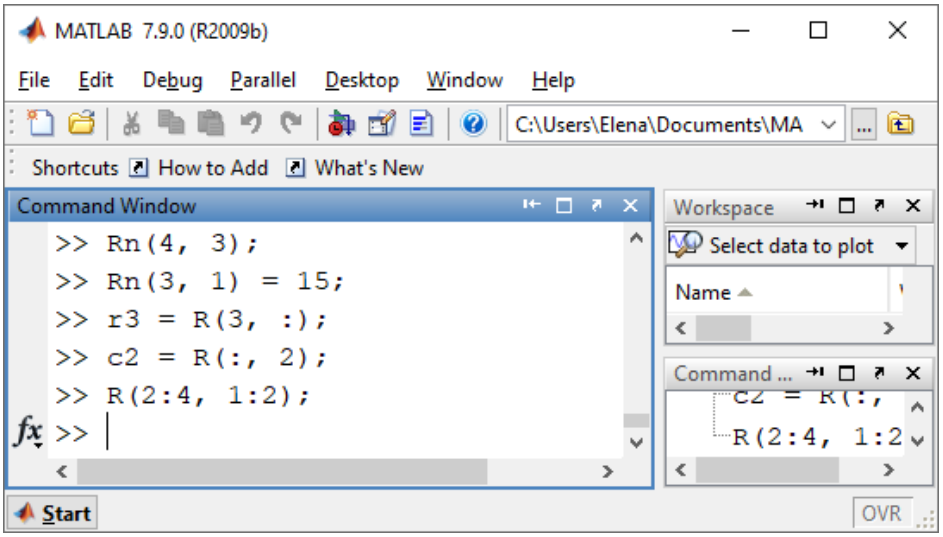
B = A(1:4, 3:5)

B = A(4:end, 3:end).

Задача: Изведете в командния прозорец на *Matlab* елементите от втори, трети, четвърти ред и първи и втори стълб на матрица R.

► A(i, :) = [ ] – премахва *i* – тия ред на матрица A;

► A(:, j) = [ ] – премахва *j* – тия стълб на матрица A;



фиг. 2.6

Аритметични действия с вектори и матрици

Основните действия с вектори и матрици, разглеждани в линейната алгебра, могат да бъдат извършвани и в *Matlab* с операторите за аритметични действия: +, -, *, ^. Освен тях, в *Matlab* има дефинирано и действието *деление* на матрици, което се извършва с операторите за дясно и ляво деление /, \.

Задача: Създайте в командния прозорец на *Matlab* две произволни матрици S и Q, които да са с еднакви размерности (например 5 реда и 5 стълба).

Упътване: Може да въведете матриците, като зададете стойностите на елементите една по една или да генерирате матриците с някои от разгледаните команди, например матрица от случайни числа и магически квадрат.

Използвайте създадените матрици и извършете с тях следните действия:

► събиране на двете матрици – A + B;

► изваждане на матриците – A - B;

► умножение на матриците – A*B;

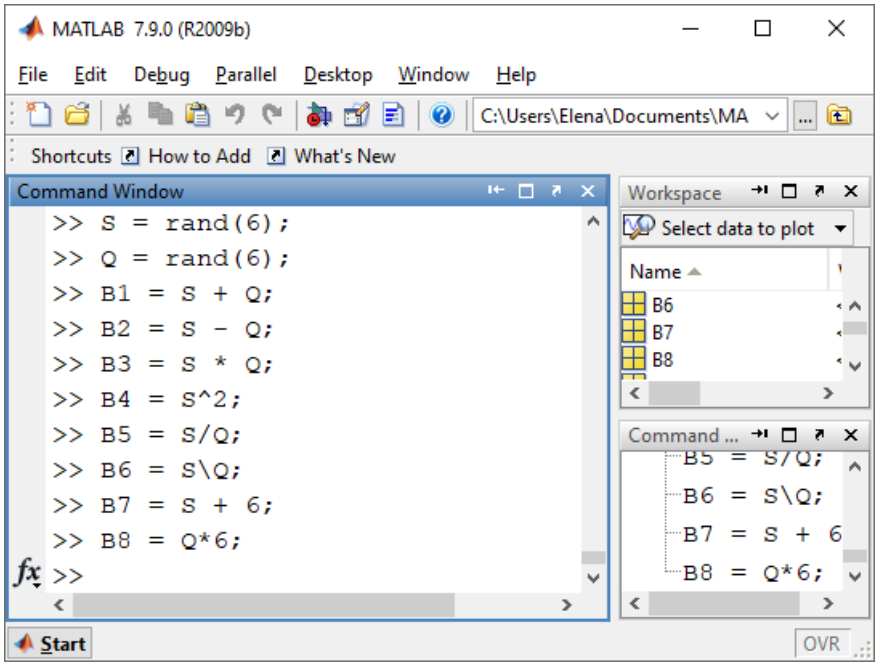
► степенуване на матрици – A^n (n е цяло число);

► *деление* на матрици - в *Matlab* има дефинирано и действие *деление* на матрици, което може да бъде извършено по два начина:

- деление отдясно – A/B;
- деление отляво – A\B.

!!! При действия събиране, изваждане и умножение, единият от аргументите може да бъде скалар (A+b, A-b, A*b)!

Задача: Към матрица S добавете произволно число, а матрица Q умножете със същото число.



фиг. 2.7

Поелементни операции с вектори и матрици

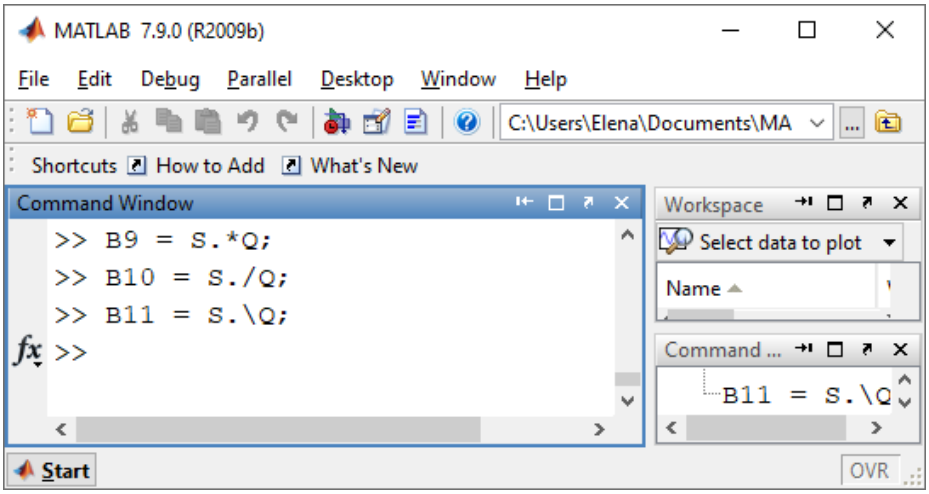
В *Matlab* са дефинирани т.нар. поелементни операции. Това са операции, които се изпълняват върху всеки отделен елемент на дадена матрица. Такива операции са:

► поелементно умножение – „ .* ”;

► поелементно деление – „ ./ ” и „ .\ ”;

► поелементно степенуване – „ .^ “.

🤔 ***Задача:*** Извършете посочените поелементни операции, като използвате матрици S и Q.



фиг. 2.8

Матрични функции

В *Matlab* има вградени следните матрични функции:

► детерминанта на матрица - *det(A)* ;

🤔 ***Задача:*** Намерете детерминантите на матрици S и Q.

► обратна матрица - *inv(A)* ;

🤔 ***Задача:*** Намерете обратната матрица на S и Q.

► транспониране на матрица – *A'*;

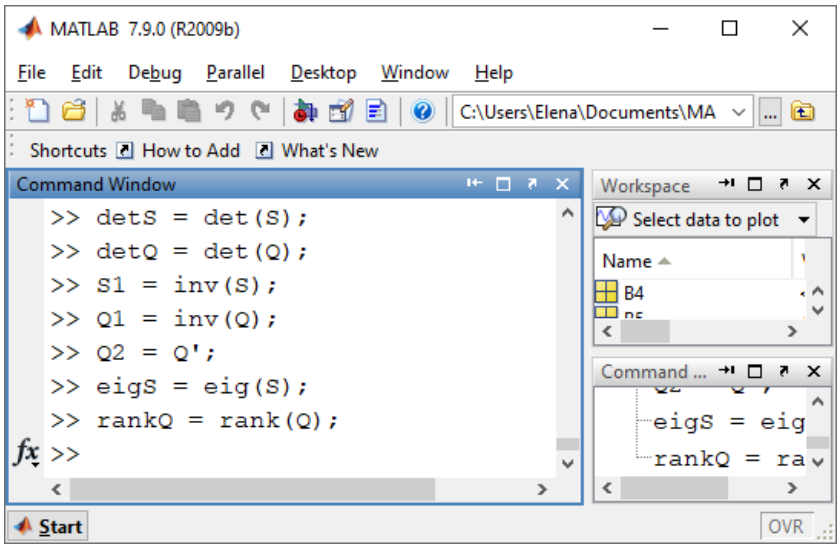
🤔 ***Задача:*** Намерете транспонираната матрица на S.

► собствени стойности на матрица - *eig(A)* ;

🤔 ***Задача:*** Намерете собствените стойности на матрици S и Q.

► ранг на матрица - *rank(A)* ;

🤔 ***Задача:*** Намерете ранга на матрици S и Q.



фиг. 2.9

Обработка на експериментални данни

Matlab предлага множество функции за обработка на експериментални данни, представени във вид на вектори или матрици. Списък с функциите може да се види с командата

>> help datafun

Някои от най-често използваните функции са тези за намиране на най-голям и най-малък елемент, средноаритметична стойност и др.

► *max(x)* – тази команда връща стойността на най-големия елемент на вектор x;

► *min(x)* – връща стойността на най-малкия елемент на вектор x;

► *mean(x)* – използва се за пресмятане на средноаритметичната стойност на елементите на вектор x;

► *sort(x)* – тази команда се използва за подреждане на елементите на вектор x във възходящ ред;

► *sum(x)* – пресмята сумата от всички елементи на вектор x;

► *prod(x)* – пресмята произведението от всички елементи на вектор x;

!!! В случай, че като аргумент на разгледаните функции бъде подадена матрица вместо вектор, то използваната функция ще се изпълни върху отделните стълбове на дадената матрица. Ако като аргумент на функцията бъде подадена транспонираната матрица, то

действието на функцията ще бъде изпълнено върху редовете на дадената матрица.

!!! Функциите за намиране на най-малък и най-голям елемент, min и max, могат да бъдат използвани с два изходящи аргумента:

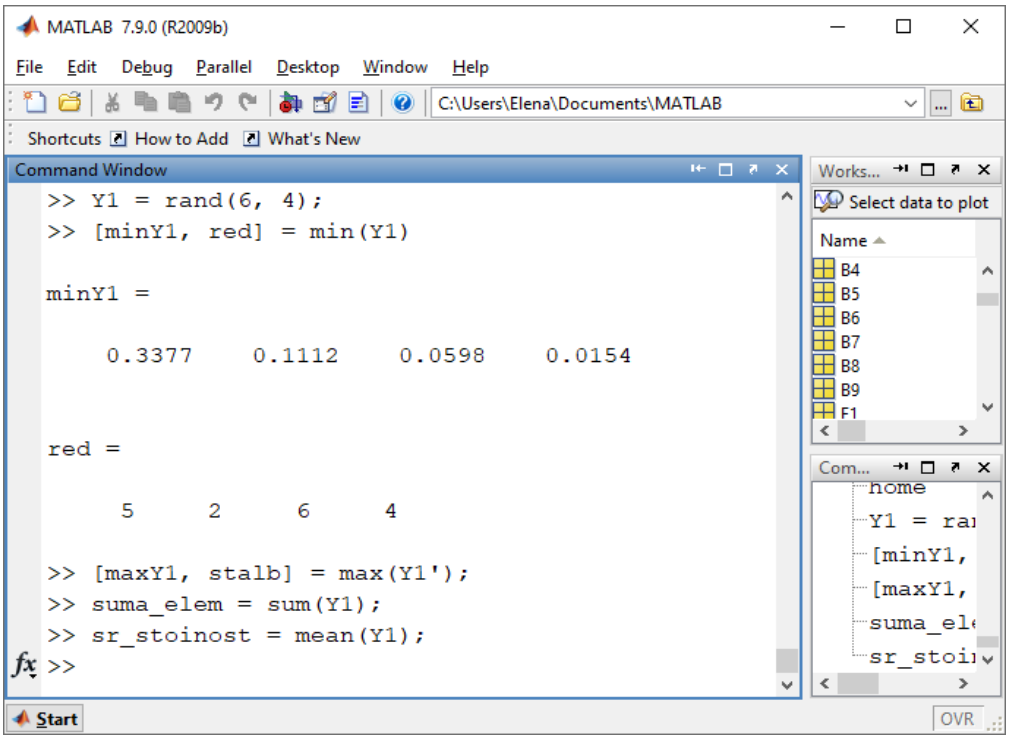
► [v, n] = max(x) или [v, n] = max(A);

► [v, n] = min(x) или [v, n] = min(A).

В този случай *v* ще приеме стойността на най-големия (или най-малкия) елемент от вектора (или матрицата), а *n* – неговия индекс.

Задача: Генерирайте в командния прозорец на *Matlab* матрица от случайни числа X_1 с размерност 6 реда и 4 стълба. За получената матрица намерете:

- стойността на най - малкия елемент от всеки стълб на X_1 и реда, на който се намира той;
- стойността на най - големия елемент от всеки ред на X_1 и номера на стълба, на който се намира;
- сумата от елементите на стълбовете на X_1;
- средноаритметичната стойност от елементите на стълбовете на X_1;



фиг. 2.10

Задаване на полиноми

Полиномът

$$P(x)=a_0x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n$$

в *Matlab* се задава като вектор ред, елементите на който са коефициентите на дадения полином. Векторът изглежда по следния начин:

$$P=[a_0 \quad a_1 \quad \ldots \quad a_{n-1} \quad a_n].$$

Операции с полиноми

Списък на командите за работа с полиноми може да се види с командата

>> help polyfun

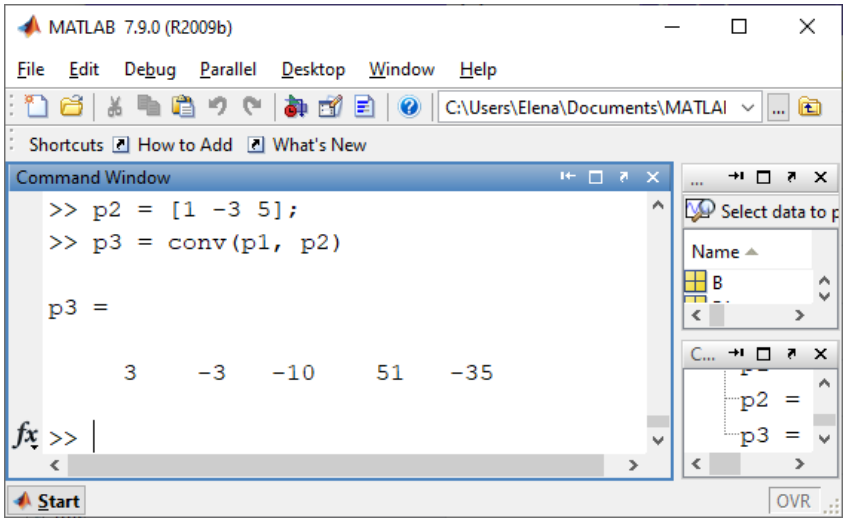
Най – често използваните команди, свързани с действия с полиноми, са:

► умножение на полиноми

p=conv(p1, p2);

***Задача**:* Да се намери произведението на полиноми p_1 и p_2

$$p_1=3x^2+6x-7 \text{ и } p_2=x^2-3x+5.$$



фиг. 2.11

След извършване на умножението, полученият полином има вида:

$$p_3=3x^4-3x^3-10x^2+51x-35$$

► пресмятане на корените на полином *p*

r = roots(p);

***Задача**:* Да се намерят корените на полинома $p_4=x^3+9x^2+23x+15$.

► намиране на коефициентите на полином *p* по зададени негови корени r

$$p = poly(r);$$

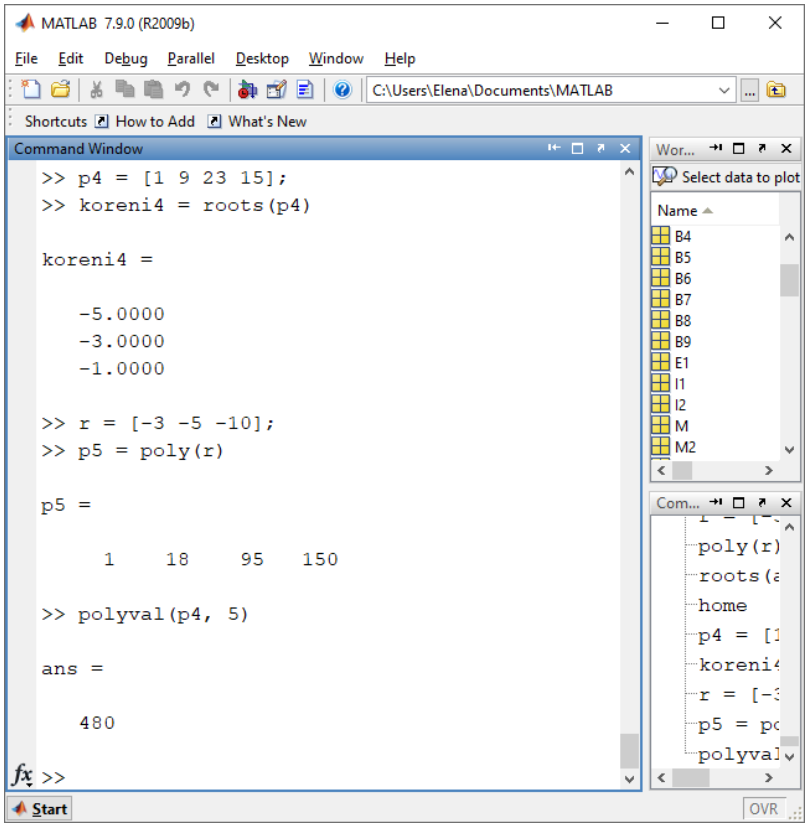
Задача: Да се намерят коефициентите на полином p_5, ако корените му са

$$r = [-3 \quad -5 \quad -10].$$

► пресмятане стойността на даден полином p за конкретна стойност на аргумента *x*

$$polyval(p, x).$$

Задача: Да се пресметне стойността на полинома p_4 за $x = 5$.



фиг. 2.12

Въпроси и задачи за изпълнение:

1. Какви действия с матрици могат да бъдат извършвани в средата на *Matlab*?

2. Каква е същността на поелементните действия и как се извършват те в средата на *Matlab*?

3. Как може да се извлече определен елемент от дадена матрица? А цял ред или стълб?

4. Как в *Matlab* се представят полиноми?

5. В командния прозорец на *Matlab* да се генерират две матрици - A_1, която да бъде магически квадрат с размерност 5 реда и 5 стълба, и A_2, която да е единична матрица (единици по главния диагонал и нули извън него) с размерността на A_1. С тези две матрици да се извършат следните действия:

- Матрица A_1 да се умножи с произволно число, а резултатът от извършеното действие да се запише като матрица A_3.
- да се умножат матрици A_1 и A_2 по правилата на линейната алгебра, а резултатът да се запише като матрица A_4;
- да се получи матрица A_5, която да е резултат от поелементно умножение на A_1 и A_2;
- да се създаде матрица A_6, която да съдържа елементите на матрица A_1, подредени във възходящ ред.

6. В командния прозорец на *Matlab* да се генерира матрица от случайни числа B с произволна размерност, за която да се намери:

- сумата от елементите на всеки стълб;
- средноаритметичната стойност от елементите на всеки стълб;
- произведението от елементите на всеки ред;
- да се изведат елементите от главния диагонал на матрица B и да се запишат във вектор b_1;
- да се намерят корените на полинома, коефициентите на който са елементите на вектор b_1.

7. В командния прозорец на *Matlab* да се генерира произволна матрица B_1 с размерност 6 реда и 6 стълба, за която да се пресметне:

- нейната детерминанта;
- транспонираната й матрица;
- обратната й матрица;
- собствените й стойности;
- стойността на най-малкия и най-големия елемент от всеки стълб на матрица B_1, като се определи и на кой ред се намират те;

- стойността на най-малкия и най-големия елемент от всеки ред на матрица B_1, като се определи и на кой стълб се намират те.

8. Да се намерят корените на полинома:

$$P_1 = x^3 + 27x^2 + 215x + 525.$$

9. Да се намерят коефициентите на полинома, който има 4 корена:

$$r_1 = -1, \quad r_2 = 3, \quad r_3 = 9, \quad r_4 = 17.$$

Лабораторно упражнение 3
Графично представяне на данни и функции в двумерни координатни системи в програмния продукт Matlab

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите на програмния продукт *Matlab* за изчертаване на различни двумерни графики, както и за извършване на определени допълнителни действия като мащабиране на осите, нанасяне на надписи и др.

Графики в Matlab

Matlab има голям брой възможности за графично представяне на данни и резултати от различни изчисления. Чрез вградени функции има възможност бързо и лесно да бъдат изчертавани двумерни, тримерни графики и анимация. Програмният продукт позволява гъвкаво управление на оси, мащаби, цветове, надписи и легенди, което улеснява анализа и интерпретацията на данните. Графиките могат да бъдат създавани както чрез команди, така и чрез интерактивни инструменти на интерфейса. Тези възможности правят *Matlab* подходящ инструмент за визуализация в научни и инженерни приложения.

В това упражнение са представени някои основни и най-често използвани графични команди и функции за изчертаване на двумерна графика, както и подходи за графично представяне на данни.

Двумерната графика в *Matlab* е разделена условно на две групи – обикновена и специална.

Пълен списък на всички графични команди и функции може да се изведе с командите:

>> *help graph2d* - обикновена двумерна графика

>> *help specgraph* - специална графика

Обикновена двумерна графика

В програмния продукт *Matlab* командите, свързани с графиките, могат да бъдат групирани в три направления:

► команди за изчертаване на графиките – *plot*, *loglog*, *semilogx*, *semilogy*, *polar*;

► команди за управление на осите – *axis*, *grid*, *hold*, *subplot*;

► команди за нанасяне на надписи върху графиките – *xlabel*, *ylabel*, *title*, *legend*.

Команди за изчертаване на графиките

Най-използваната функция за изчертаване на двумерни графики в *Matlab* е *plot*. Чрез нея се получава линейна графика в правоъгълна координатна система като броят на аргументите, с които се използва функцията, може да бъде различен.

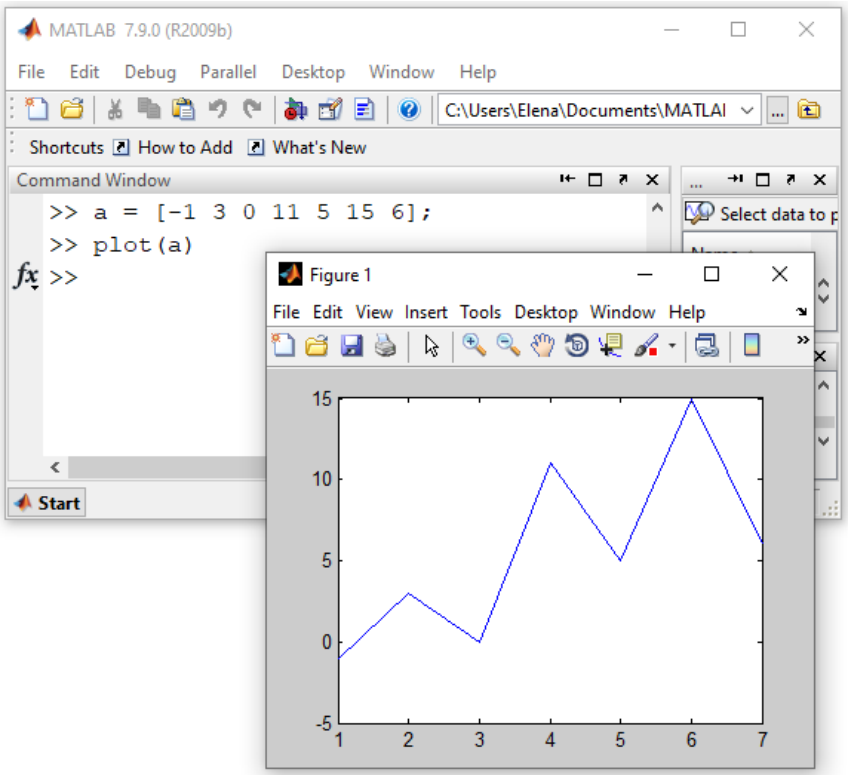
- Използване на команда *plot* с един аргумент, представляващ вектор със стойности.

plot (y)

Така използвана, командата изчертава графика на функция, чиито стойности са зададени чрез вектора y. По абсцисната ос в този случай стоят индексите на елементите на вектора y.

Задача: Да се изчертае графика, изобразяваща елементите на вектор

$$x = [-1 \quad 3 \quad 11 \quad 0 \quad 5 \quad 15].$$



фиг. 3.1

🤔 ***Задача***: Да се изчертае графика на функцията

$$y = \sin\left(\frac{2t}{3}\right),$$

където t се изменя в интервала от -5π до 5π със стъпка $\frac{\pi}{10}$.

!!! В *Matlab* интервал може да се дефинира по следния начин:

променлива = начална стойност на интервала : стъпка на изменение : крайна стойност

x = -1 : 0.5 : 25

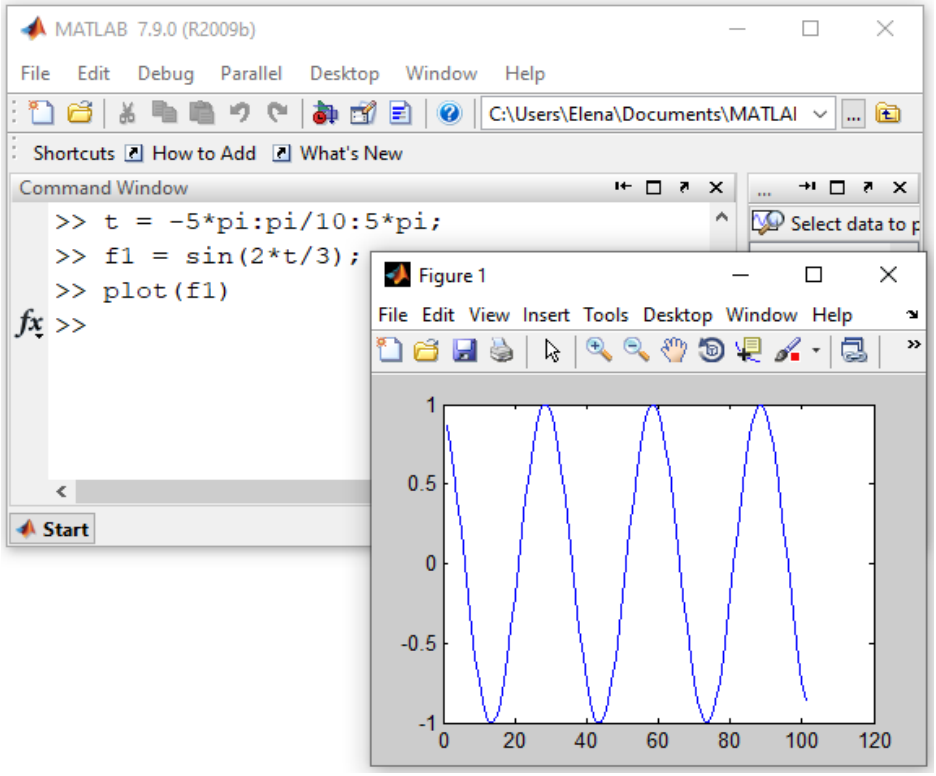
Упътване: За да се дефинира в *Matlab* интервала на изменение на *t*, е необходимо да се напише следното:

*>> t = -5*pi : pi/10 : 5*pi;* % Поставяйте символ **;** в края на реда, за да не се извеждат на екрана всички стойности на променливата t

Следващата стъпка е пресмятане на функцията y и изчертаването ѝ с команда plot:

*>> y = sin(2*t/3);* % Поставяйте отново символ **;** в края на реда, за да не се извеждат на екрана всички стойности на функцията y

>> plot(y)



фиг. 3.2

- Използване на команда *plot* с един аргумент, представляващ матрица със стойности.

plot (Y)

В този случай резултатът от изпълнението на командата е едновременно изчертаване на графики на няколко функции, зададени чрез елементите от всеки стълб на матрица Y. По абсцисната ос в този случай стоят първите индекси на елементите на матрицата;

Задача: Да се създаде матрица S, за която:

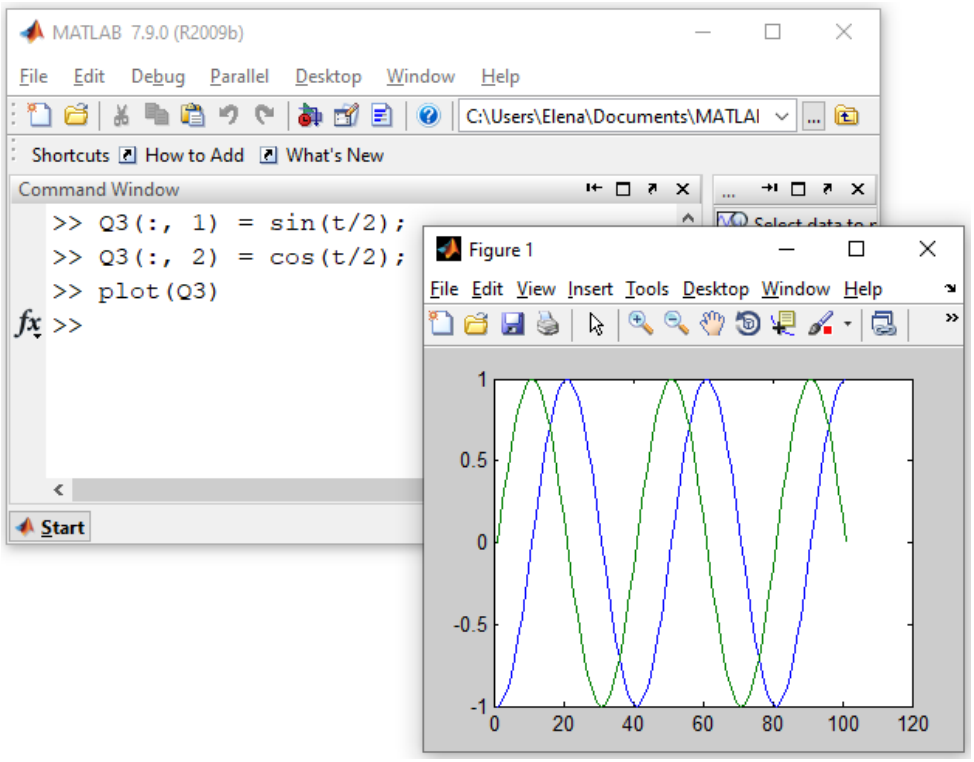
- елементите на първия й стълб да съдържат стойностите на функцията $\sin\frac{t}{2}$ (променливата t е вече създадена в предишна задача и представлява интервал [-5π: 5π]);

- елементите на втория стълб да са стойностите на функцията $\cos\frac{t}{2}$.

С команда plot да се изчертае получената матрица S.

Упътване: За да се дефинира в *Matlab* матрица S с така описаните стойности, е необходимо да се пресметнат стойностите на двете функции и да бъдат присвоени на двата стълба на матрицата. Това става по следния начин (лаб.упр.2):

```
>> S (:, 1) = sin(t/2);     % Поставяйте символ ; в края на реда, за да не
                            се извеждат на екрана всички стойности
                            присвоени на първия стълб на матрицата;
>> S (:, 2) = cos(t/2);     % Поставяйте символ ; в края на реда, за да не
                            се извеждат на екрана всички стойности
                            присвоени на втория стълб на матрицата;
>> plot(S)                  % резултатът е две графики в правоъгълна
                            координатна система, отразяващи стойностите
                            на двете функции - sin(t/2) и cos(t/2).
```



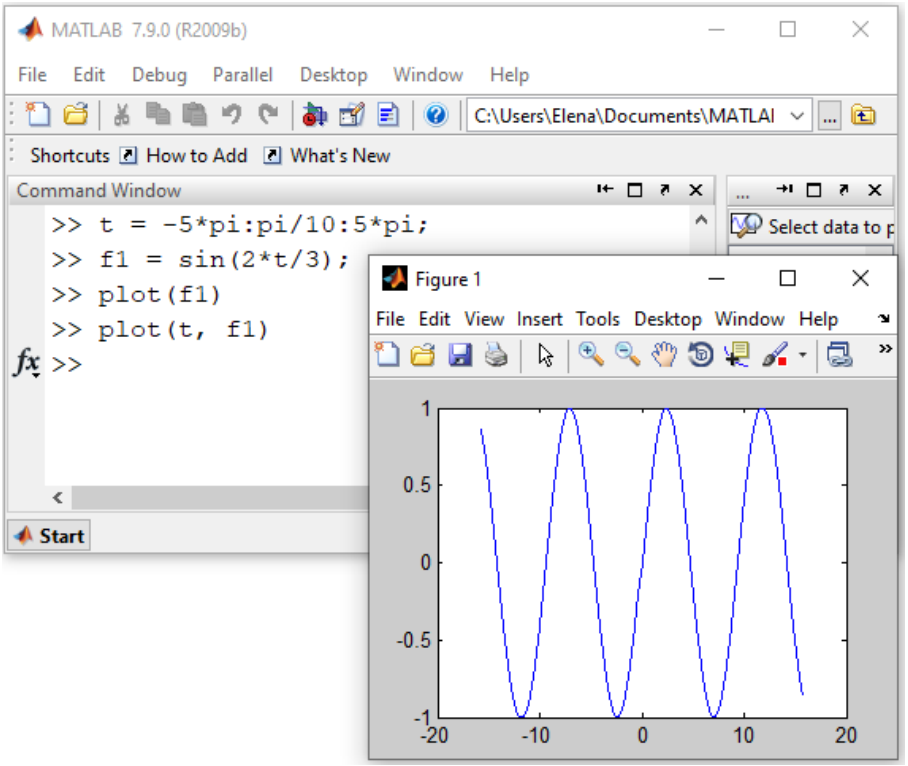
фиг. 3.3

- Използване на команда *plot* с два аргумента, представляващи два вектора със стойности

plot (x,y)

При този запис на командата в *Matlab* се изчертава графика на функцията y = f(x). В случая елементите на вектора *x*, зададен като първи аргумент на командата, задават стойностите на графиката по абсцисната ос, а тези на вектор *y* (вторият аргумент на командата) – стойностите по ординатната ос;

Задача: Да се изчертае отново графика на функцията $y = \sin\left(\frac{2t}{3}\right)$, като на команда *plot* бъдат зададени два аргумента – *t* и *y*.



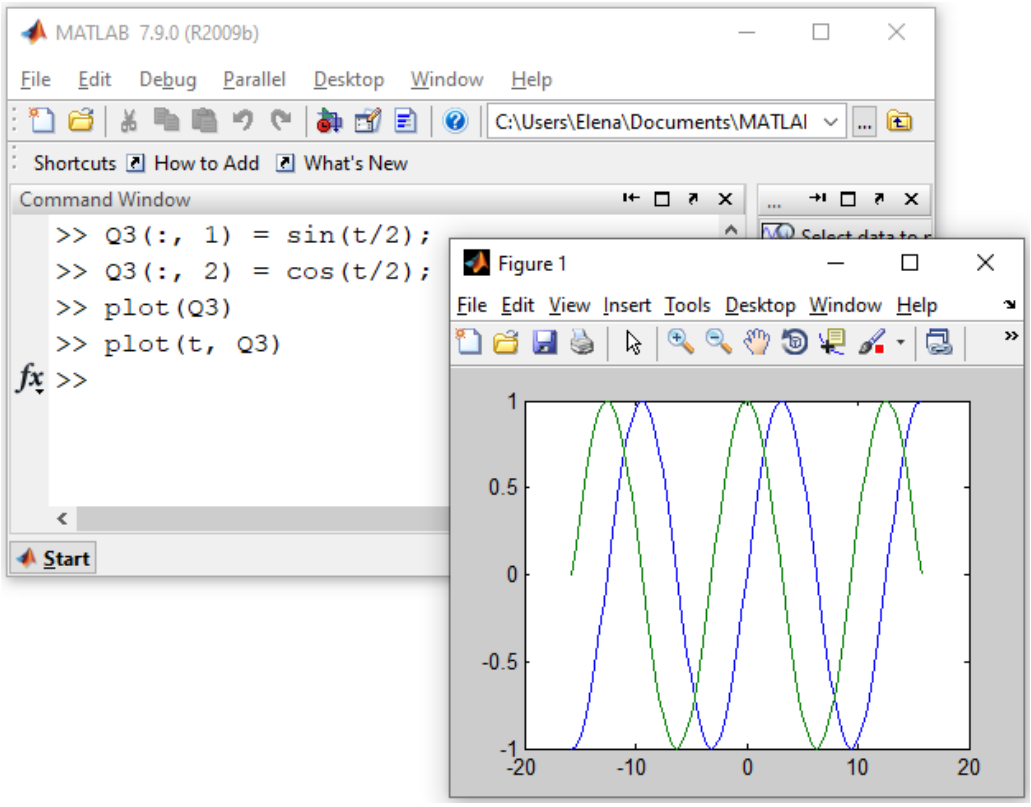
фиг. 3.4

- Използване на команда *plot* с два аргумента, представляващи вектор и матрица

plot (x,Y)

В този случай след изпълнението на командата се получава едновременно изчертаване на графики на няколко функции с един и същ аргумент – $y_1 = f_1(x)$, $y_2 = f_2(x)$, …, $y_n = f_n(x)$. Стойностите на отделните функции y_1, y_2, …, y_n са зададени като стълбове на матрица Y.

Задача: Да се изчертае още веднъж графика на матрица S като на команда *plot* бъдат зададени два аргумента – t и S.



фиг. 3.5

- Използване на команда *plot* с повече от два аргумента

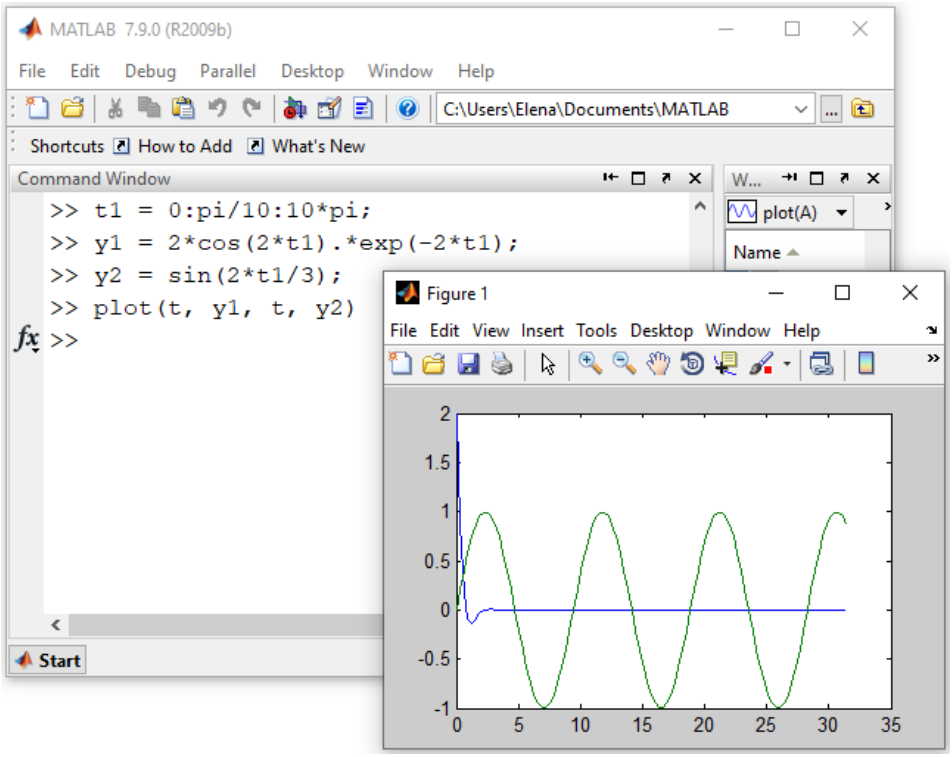
plot (x, y1, x, y2, ...)

Действието на командата, записана по този начи, е аналогично на *plot (x,Y)*, с тази разлика, че функциите, които се изчертават са зададени като вектори, а не като матрица. По този начин могат да бъдат изчертавани n на брой графики, в зависимост от броя на зададените като аргументи функции. Особеното при този запис на командата е, че всяка функция, чиято графика ще бъде изчертавана, трябва да бъде предхождана от вектора със стойности по абсцисната ос.

Задача: На една координатна система да се изчертаят графики на функциите

$$y_1 = 2e^{-2t}\cos 2t \text{ и } y_2 = \sin\frac{2t}{3},$$

където t се изменя в интервала от 0 до 10π със стъпка $\frac{\pi}{10}$.



фиг. 3.6

- Използване на команда *plot* с аргумент функция, вместо променлива

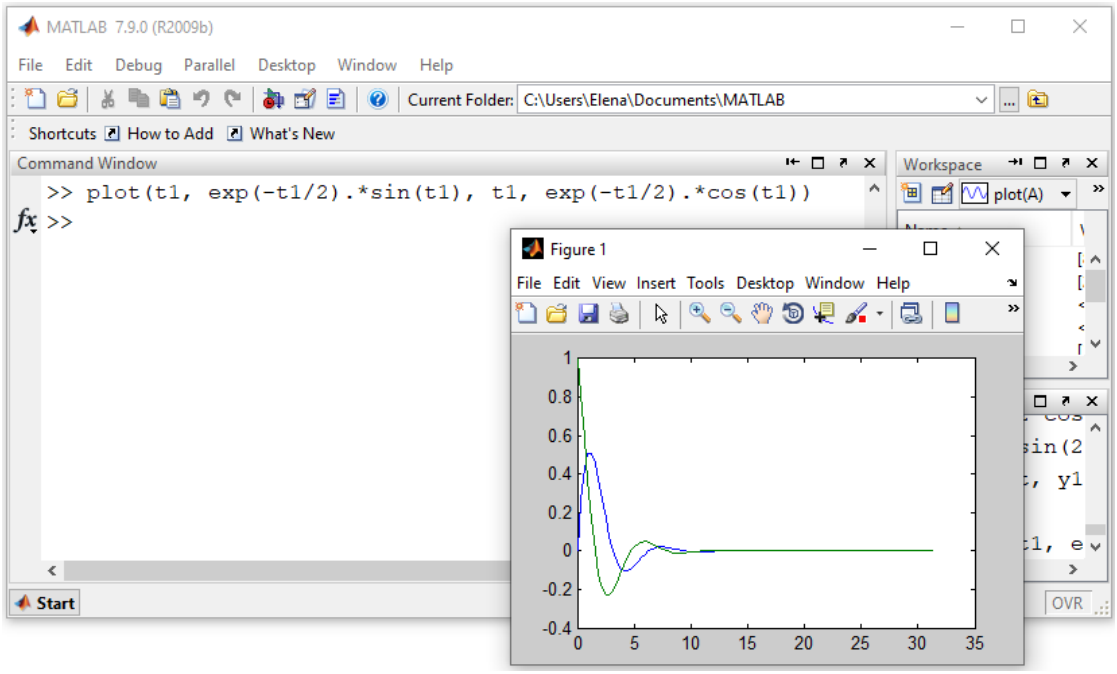
plot (x, f1(x), x, f2(x), ...)

В този случай действието на командата е както в предходния пример, с разликата, че функциите, функции, чиито графики се изчертават, се пресмятат директно в командата plot, а не предварително.

Задача: На една координатна система да се изчертаят графики на функциите

$$y_1 = e^{-\frac{t}{2}} \sin t \text{ и } y_2 = e^{-\frac{t}{2}} \cos t,$$

като пресмятането им да се извърши директно в командата за изчертаване. За t да се използват стойностите от предходната задача.



фиг. 3.7

- Използване на команда *plot* с допълнителен аргумент, свързан с описание за графиката – цвят, вид на линията и на маркера

plot (x, y1, 'str1', x, y2, ' str2', ...)

Допълнителният аргумент в случая е стринг, състоящ се от 1 до 3 символа, с помощта на които могат да се задават цвета и вида на линията и видът на маркера.

Подробна информация за вида на тези символи може да бъде получена с помощта на команда help:

>> help plot

Цветовете, които могат да се използват за линиите на графиките, са следните:

- r - red – червен;
- m – magenta – малинов;
- g – green – зелен;
- c – cyan – синьо-зелен;
- b – blue – син;
- k – black – черен;
- y – yellow – жълт;
- w – white – бял.

Вид на маркера:

- . точка
- d ромб

- \+ знак “плюс”
- * звезда
- o кръгче
- x кръстче
- s квадратче
- v триъгълник с върха надолу
- ^ триъгълник с върха нагоре
- > триъгълник с върха надясно
- < триъгълник с върха наляво

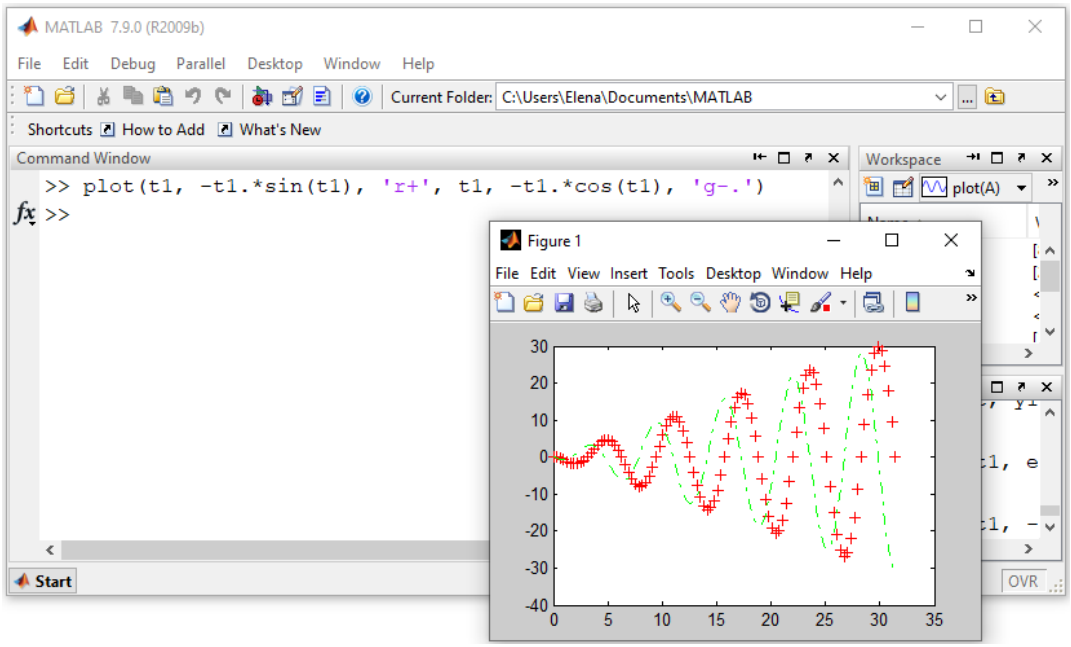
Видове линии за изчертаване на графиките

- \- непрекъсната линия;
- \- - прекъсната линия;
- : линията се визуализира с точки;
- -. тире и точка.

Задача: На една координатна система да се изчертаят графиките на две функции

$$y_3 = -t\sin t \text{ и } y_4 = -t\cos t,$$

като y_3 се изчертае с червени звездички, а y_4 - с тирета и точки в зелен цвят.



фиг. 3.8

- Използване на команда *plot* с аргумент, задаващ допълнителни свойства на линията и маркера

plot (x, y1, 'str1', 'parameter', 'parameter_value', x, y2, 'str2', 'parameter', 'parameter_value',...)–

Чрез допълнителните аргументи могат да се задават следните свойства на линиите и маркерите:

'LineWidth', n – дебелина на линията, като n е цяло число;

'MarkerEdgeColor', 'symbol' – цвят на маркера, ако е отворен, или цвят на рамката на маркера, ако е затворена фигура;

'MarkerFaceColor', 'symbol' - цвят на запълване на маркера, ако е затворена фигура;

'MarkerSize', n – размер на маркера.

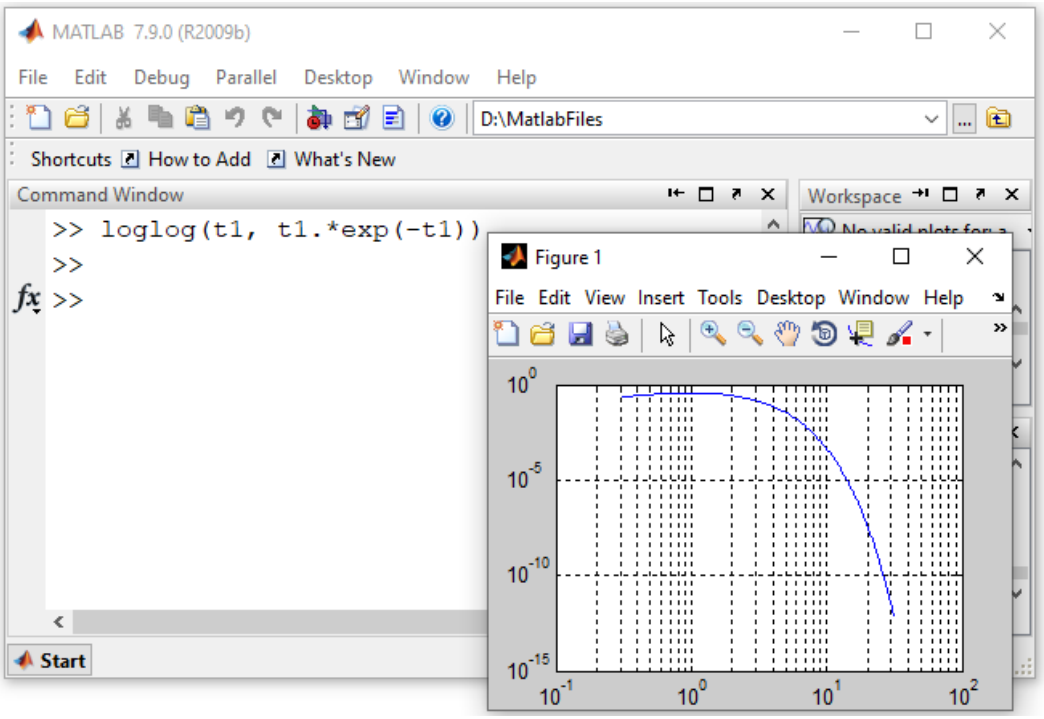
!!! Особености при изчертаване на графики в *Matlab*:

- **при изчертаване на няколко графики едновременно, без да е указан цвета на линиите, системата автоматично подбира различен цвят за всяка графика;**
- **при задаване цвета на линията, вида на линията и на маркера, символите могат да се подреждат произволно. Символите могат да са от 1 до 3, но не повече от 1 от всяка категория;**
- **ако не се укаже видът на линията, се изчертават само точките, без те да се свързват;**
- **ако при извикване на дадена графична команда има отворена фигура, новата графика се изчертава в нея, т.е. препокрива предишната изчертана графика. За да се изчертае новата графика в нов прозорец, е необходимо преди командата за изчертаването ѝ да се отвори нова фигура с командата *figure*.**

В програмния продукт *Matlab* има възможност за изчертаване на графики и в логаритмичен мащаб, който се използва при анализ на честотни характеристики на системите. За целта се използват командите *loglog*, *semilogx*, *semilogy*, които са аналогични на *plot*, с разлики в използвания мащаб по координатните оси:

► *loglog* – изчертава графика като използва логаритмичен мащаб по двете координатни оси;

🤔 ***Задача***: Да се начертае в логаритмичен мащаб графика на функцията $y_5 = te^{-t}$.

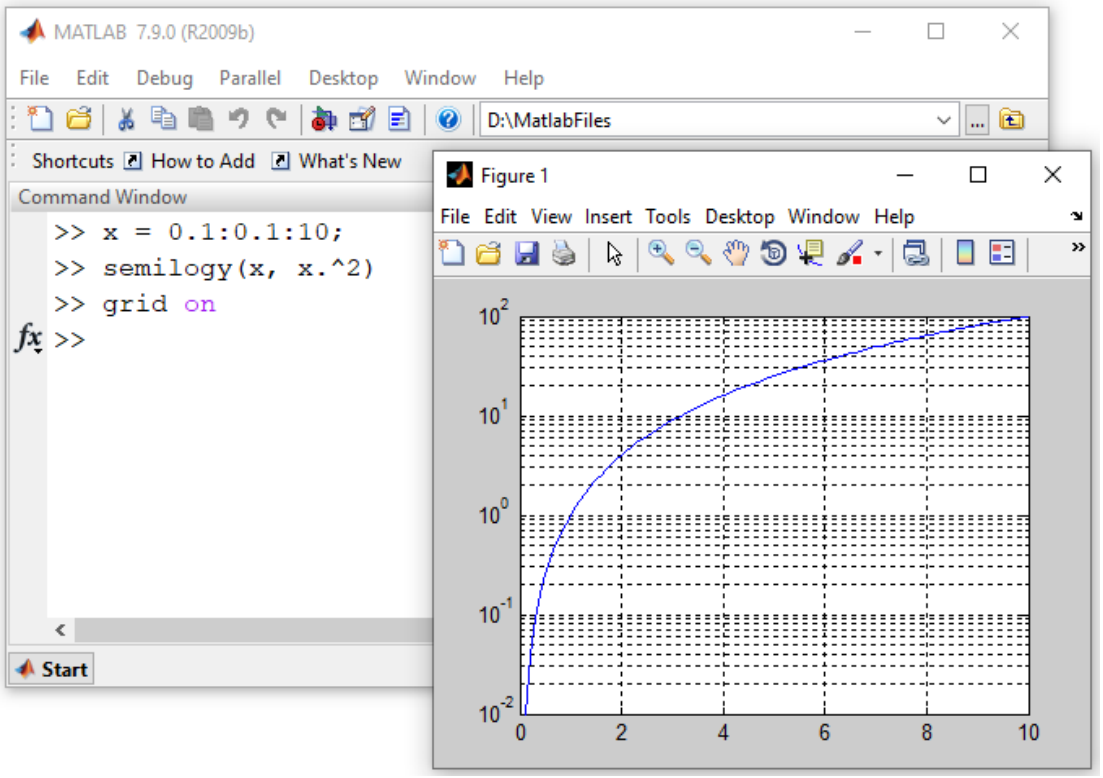


фиг. 3.9

► *semilogx* – при използване на тази команда, зададената функция се изчертава като се задава логаритмичен мащаб по оста *x*, а остта *y* остава линейна;

► *semilogy* - при използване на тази команда, логаритмичният мащаб е по оста *y*, а остта *x* остава линейна;

Задача: Да се начертае в полулогаритмичен мащаб (логаритмичен по остта y) графика на функцията $y_6 = x^2$ за стойности на x в интервала от 0.1 до 10 със стъпка на изменение 0.1.



фиг. 3.10

В програмния продукт *Matlab* също така има възможност за изчертаване на графики и в полярни координати.

► *polar* – с помощта на тази команда се изчертава графика на функцията $r = r(\varphi)$ в полярни координати. Записът на функцията е следният:

polar (phi, r, 'str'),

където:

phi – вектор със стойностите на полярния ъгъл в радиани;

r - вектор със стойностите на радиуса;

'str' – аргумент, задаващ видът на линията, както при предишните функции.

Задача: Да се изчертае графика в полярни координати на функцията $r_1 = 0.1\theta$, където θ приема 600 стойности в интервала от 0 до 6π. Да се зададе червен цвят на линията.

Упътване: За да се дефинира в *Matlab* вектор с определен брой стойности, разпределени в даден интервал, се използва командата *linspace()*. Възможните варианти за нейното приложение са:

✓ генериране на вектор, съдържащ 100 елемента, линейно разпределени в интервала *(a, b)*

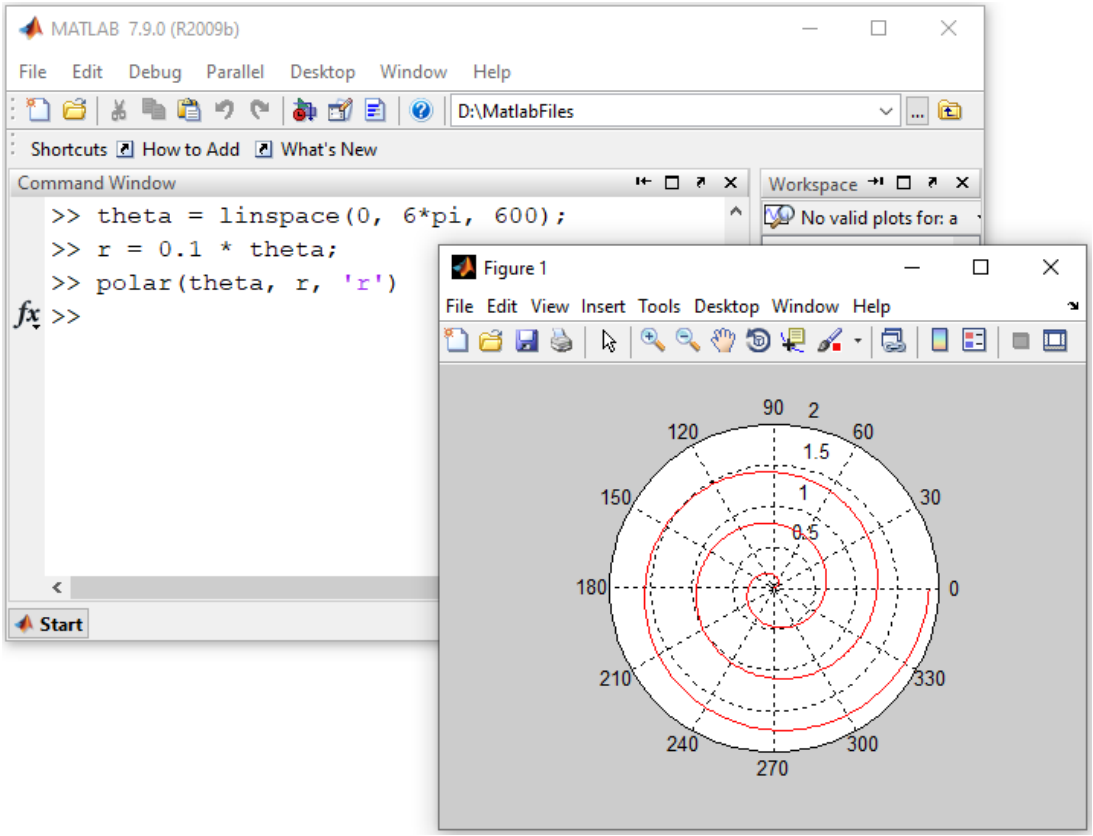
linspace(a, b)

✓ генериране на вектор, съдържащ *n* елемента, линейно разпределени в интервала *(a, b)*

linspace(a, b, n)

За изпълнение на поставената задача, е необходимо да се генерира вектор с 600 елемента, линейно разпределени в дадения интервал. Това може да бъде получено с командата:

*>> theta = linspace(0, 6*pi, 600);*



фиг. 3.11

Команди за управление на осите

Функцията *axis* управлява мащабирането на координатните оси на текущата графика. Някои от по-често използваните варианти на функцията са следните:

- *axis ([xmin xmax ymin ymax])* – задава интервалите на изменение на променливите по координатните оси;

- *axis auto* – връща стойностите, избрани от системата;

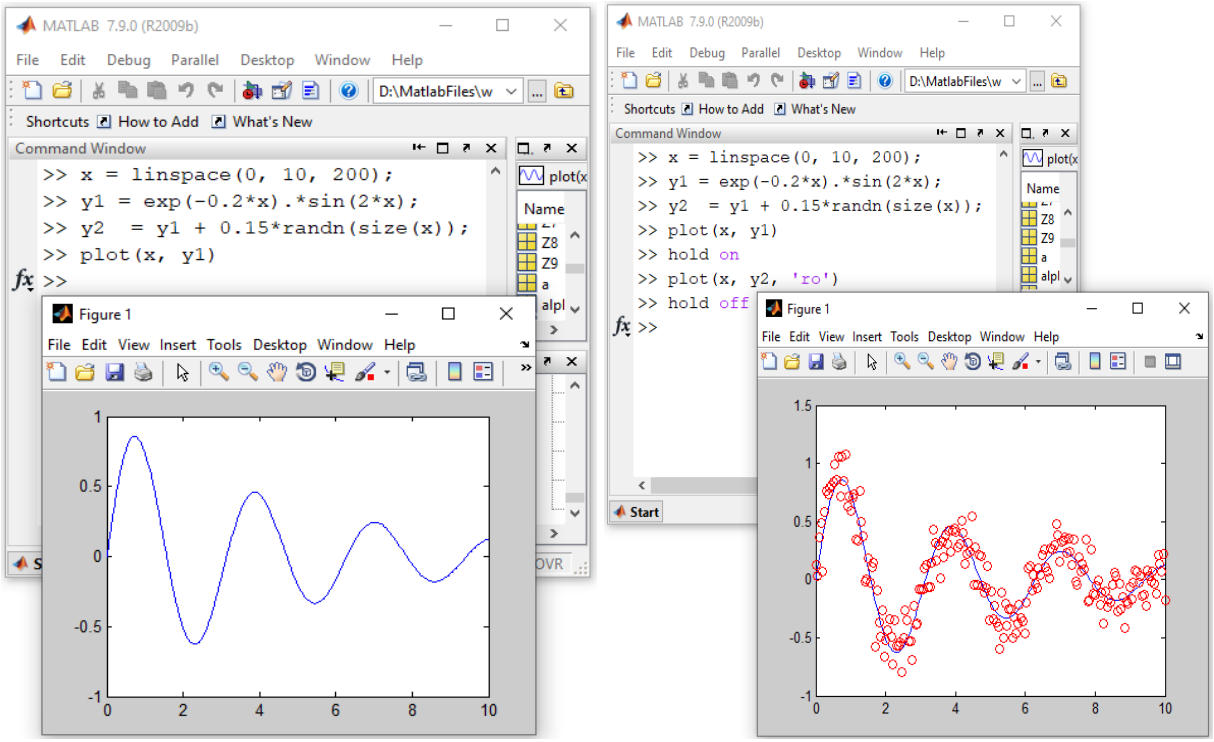
В *Matlab* има команда, с помощта на която може да се добави или да се премахне мрежата на текущата графика. Това са командите *grid on* - показва мрежа към текущата графика, и *grid off*, която премахва мрежата.

Командата *hold on* запомня съществуващата графика на екрана, а всички следващи команди, изчертаващи нови графики, добавят тези графики към вече съществуващите, като се използва зададеният към момента мащаб на осите. Команда *hold off* прекратява действието на команда *hold on*.

Задача: В командния прозорец на *Matlab* да се създаде променлива x, която да съдържа 200 стойности в интервала от 0 до 10. Чрез зададената променлива x да се пресметнат функциите

$y_1 = e^{-0.2x}\sin 2x$ и y_2, която е формирана от функцията y_1, към която са добавени случайни числа .

Да се изчертае графика на функция y_1, а използвайки команда hold on, към нея да се добави и графика на функцията y_2, която да се изчертае с червени кръгчета.



фиг. 3.12

В програмния продукт *Matlab* има възможност в един графичен прозорец да бъдат изчертани няколко графики, разположени в отделни координатни системи. За да стане това, е необходимо преди изчертаването им графичният прозорец да бъде разделен по начин, по който да могат да бъдат подредени желаните координатни системи. Това се извършва с помощта на командата *subplot*, на която трябва да бъдат зададени три аргумента:

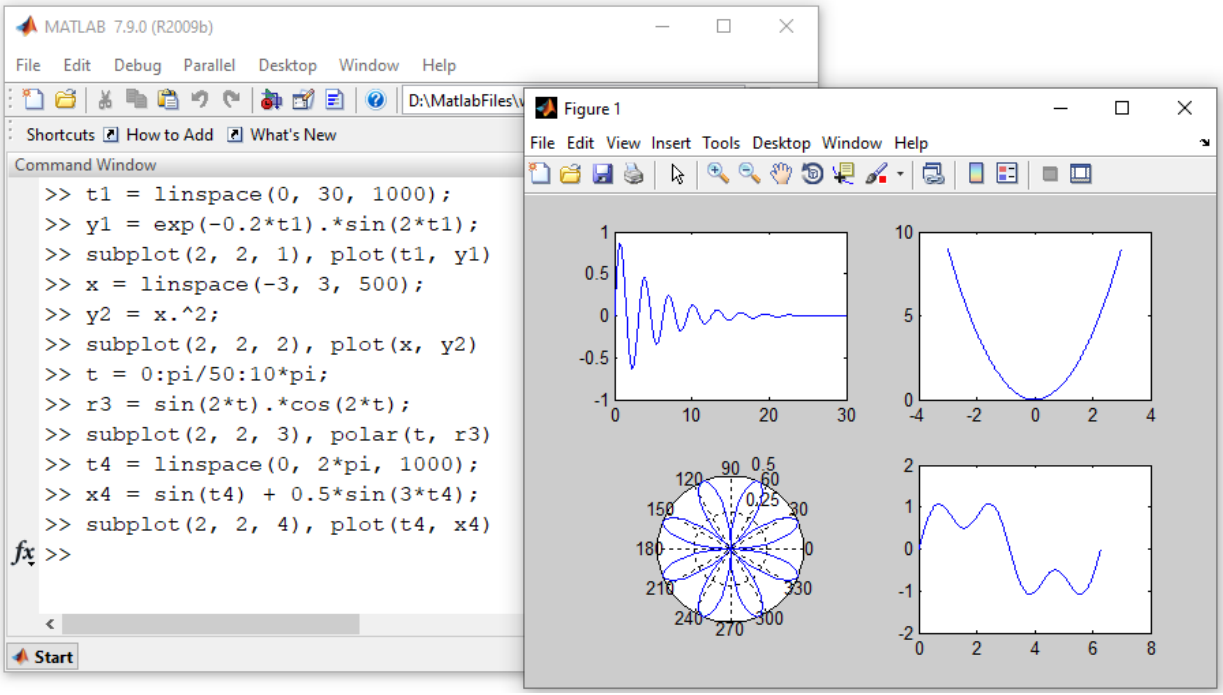
subplot(m, n, k)

Действието на командата се състои в разделяне на графичния прозорец на *m* реда и *n* стълба. Третият аргумент на командата, *k*, указва номера на координатната система, на която ще се изчертае графиката на посочената функция.

!!! Важна подробност при използване на командата *subplot* е да не се затваря графичния прозорец до изчертаването на всички графики.

Задача: В един графичен прозорец на четири отделни координатни системи да се изчертаят графики на функциите

- ✓ $y_1 = e^{-0.2t_1}\sin 2t_1$, за t_1 да се зададат 1000 стойности в интервала от 0 до 30.
- ✓ $y_2 = x^2$, за 500 стойности на *x* в интервала от -3 до 3.
- ✓ $r_3 = \sin 2t \cos 2t$, представена в полярна координатна система, като t се изменя в интервала от 0 до 10π със стъпка $\frac{\pi}{50}$.
- ✓ $x_4 = \sin t_4 + 0.5\sin 3t_4$, за 1000 стойности на t_4 в интервала от 0 до 2π.



фиг. 3.13

Команди за нанасяне на надписи върху графиките

Върху графиките могат да се поставят различни надписи. Функциите, които се използват за целта са: *xlabel*, *ylabel*, *title*, *legend*.

► *title*('text') – с използването на тази команда се изписва заглавие (зададено като аргумент на командата) в горната част на графиката;

► *xlabel* ('text') – изписва *text* по абсцисната ос;

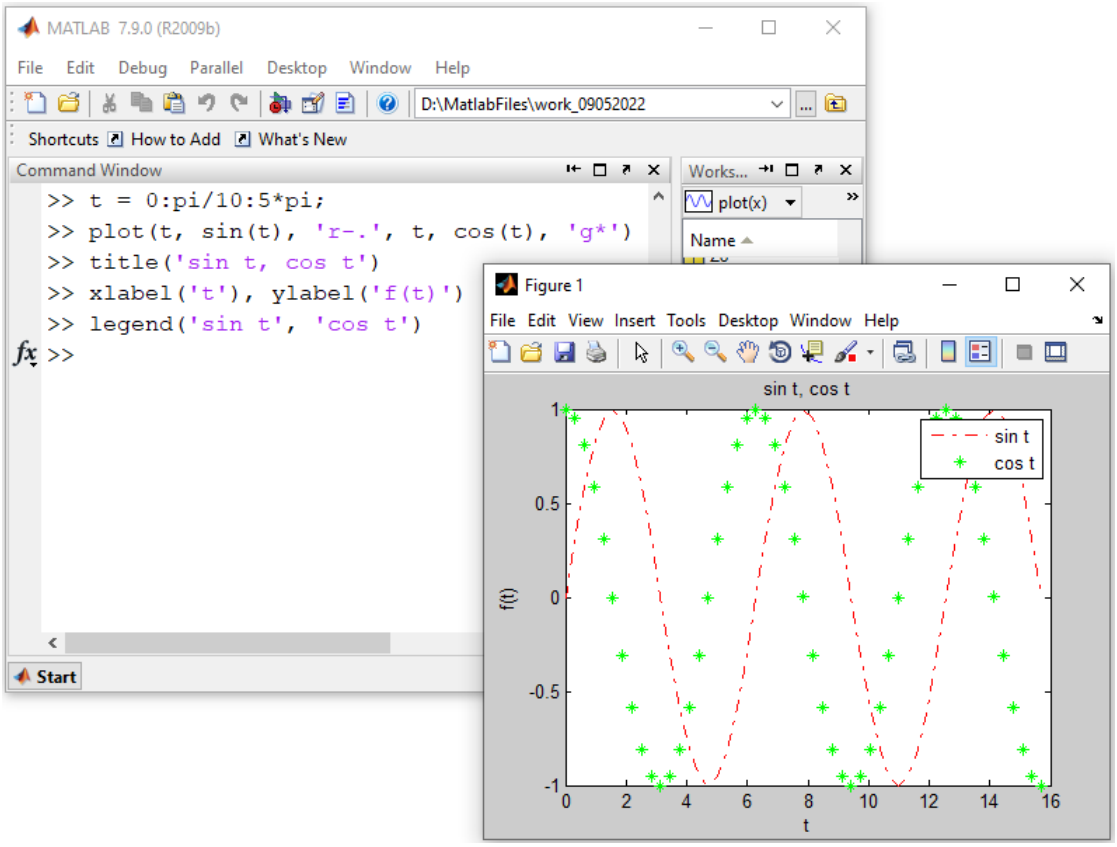
► *ylabel* ('text') – изписва *text* по ординатната ос;

► *legend* ('text1', 'text2', …) – изобразява легенда в горната дясна част на графичния прозорец. Удобно е да се използва когато в един прозорец са изобразени графики на няколко функции. За всяка графика в легендата е посочена линия с цвета и вида на линията ѝ, а след нея се изписва съответният *text* от аргументите на командата *legend*. Последователността на аргументите на функцията трябва да отговаря на последователността на построяване на графиките.

Задача: За визуализиране действието на описаните команди, да се изчератят две графики в една кооординатна система, за които да се зададат цвят, вид на линията и маркера. Например графики на функциите $\sin t$

(изчертана с тире и точка в червен цвят) и $\cos t$ (изчертана със звездички в зелен цвят).

Да се постави заглавен текст към графика, надписи по координатните оси и легенда.



фиг. 3.14

Специална двумерна графика

Най-често използваните функции, принадлежащи към тази група, са: *comet, bar, stem, stairs, pie, hist, fplot, ezplot, ezpolar*.

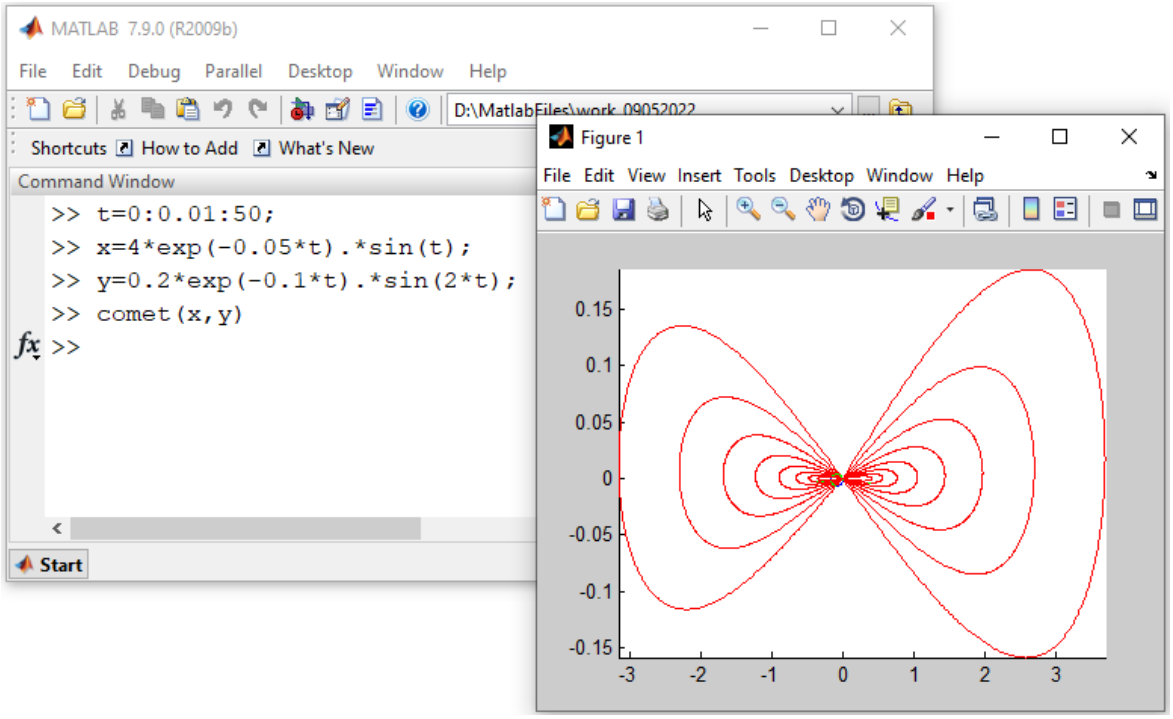
► *comet* – изчертава анимирано графика на функцията y = f(x) с помощта на движеща се опашата комета. Аргументите на функцията могат да бъдат един или два:

comet(y) или *comet(x, y)*

Задача: Да се пресметнат функциите

$$x = 4e^{-0,05t}\sin t \text{ и } y = 0,2e^{-0,1t}\sin 2t$$

за стойности на t от 0 до 50 и стъпка на изменение на t 0.01. С функция *comet* да се изчертае графиката на $y = f(x)$.

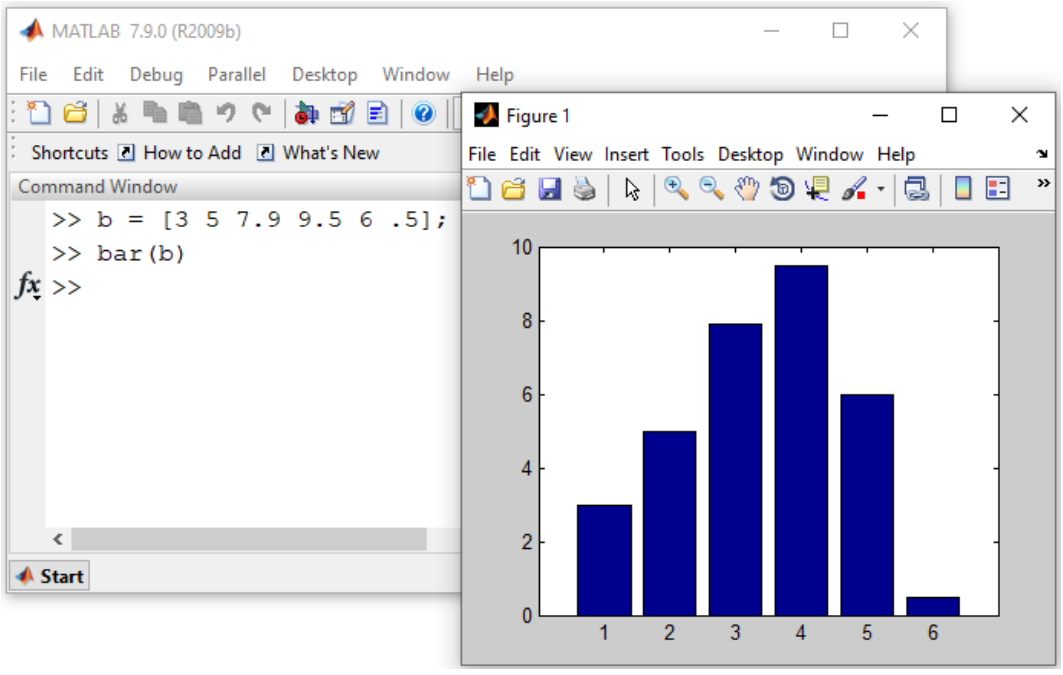


фиг. 3.15

► *bar* – чрез тази функция се чертае лентова графика, като аргументите на командата могат да бъдат записани по някой от следните начини:

bar(y), bar(x,y)

Задача: Да се изчертае лентова графика на вектор $b_1 = [3 \quad 5 \quad 7{,}9 \quad 9{,}5 \quad 6 \quad 0{,}5]$.



фиг. 3.16

► *stem* – тази функция обикновено се използва за визуализация на дискретни функции. Чрез нея се изобразява графика на дадената функция само с кръгчета, свързани с абцисната ос. Параметрите, с които може да бъде използвана командата, са от един до три.

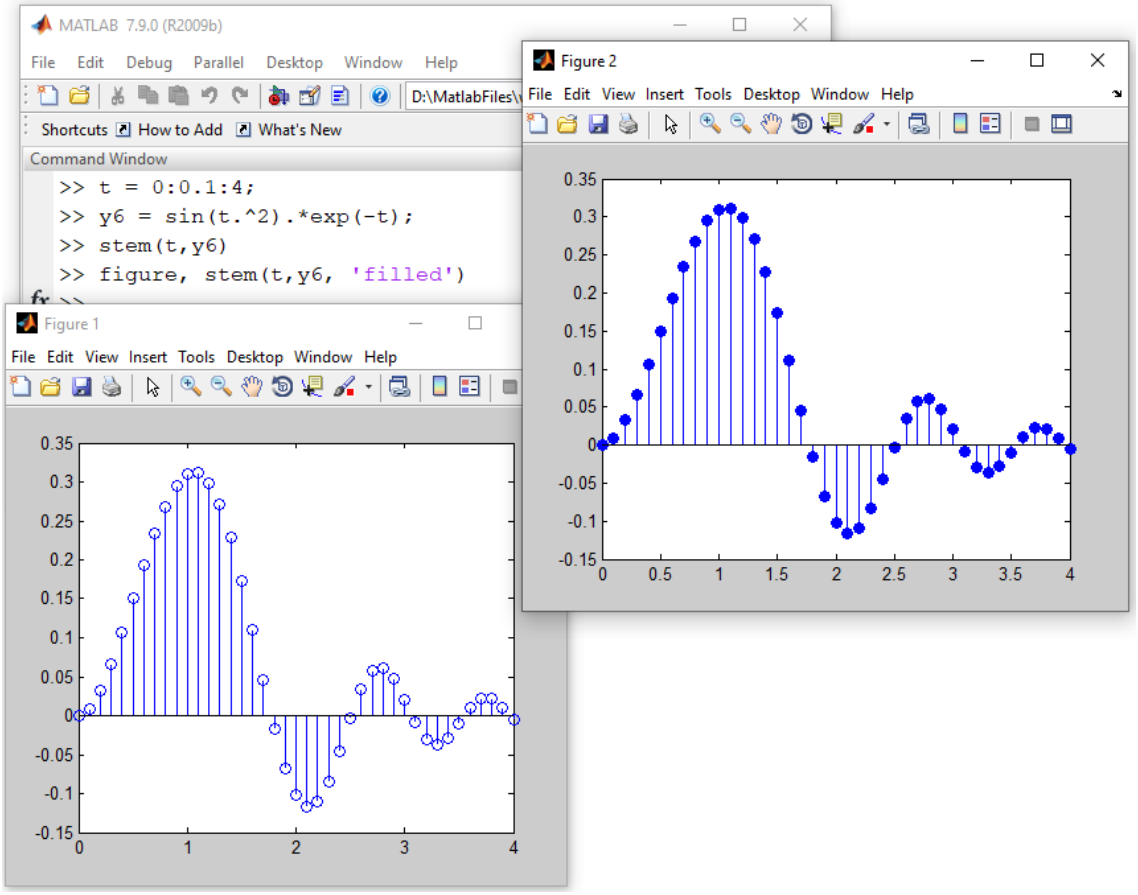
stem(y), stem(x,y)

stem(x,y, 'filled') - аргументът *'filled'* се използва за запълване на кръгчетата.

Задача: Да се изчертае графика на функцията

$$y_6 = e^{-t}\sin t^2$$

за стойности на t от 0 до 4 със стъпка 0.1.

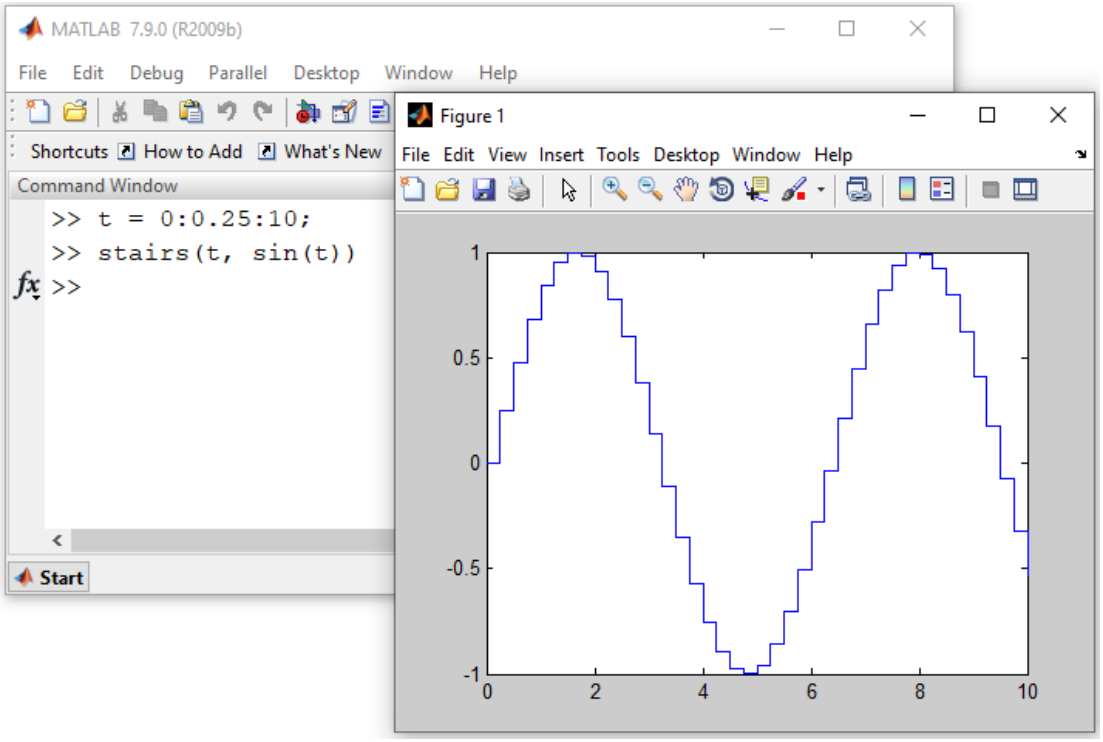


фиг. 3.17

► *stairs* – тази функция се използва за изчертаване на стъпаловидна графика

stairs(y), stairs(x, y)

Задача: Да се изчертае стъпаловидна графика на функцията *sint*, за стойности на t от 0 до 10 със стъпка 0.25.



фиг. 3.18

► *pie*- служи за построяване на кръгови диаграми;

pie(y);

Командата може да има и допълнителен аргумент:

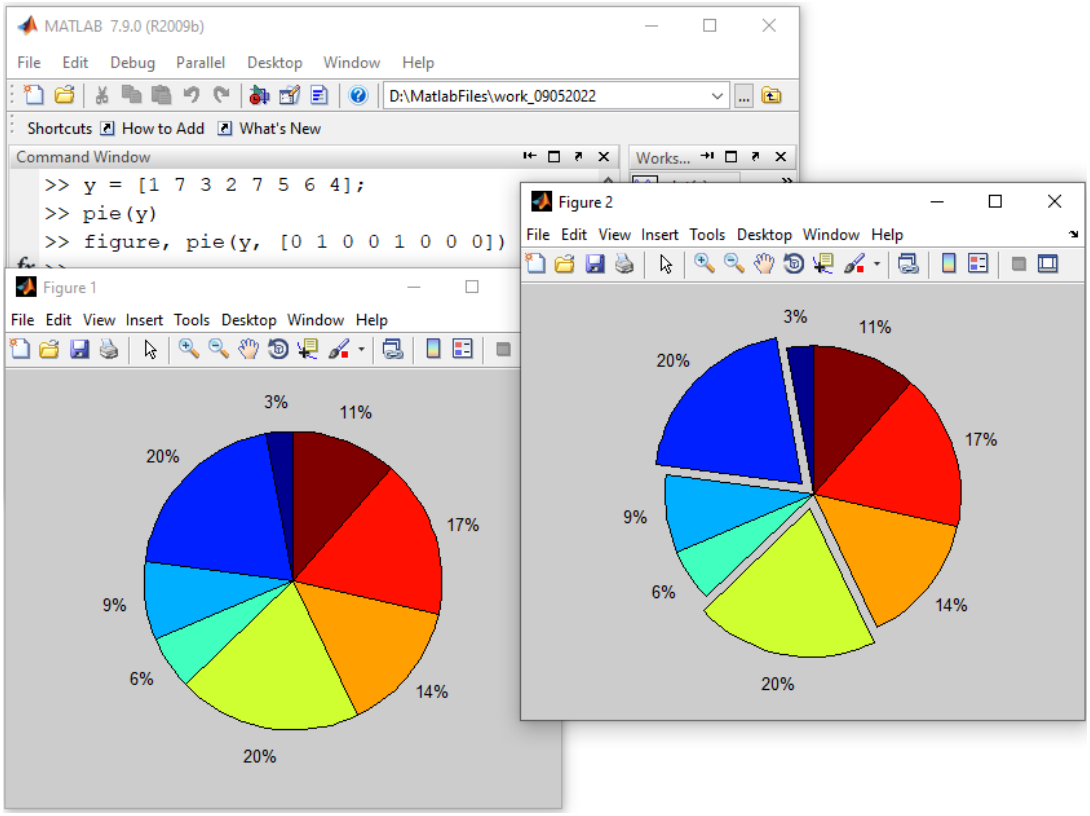
pie(y, explode),

където *explode* е вектор с дължината на y, който се състои само от нули и единици. При изчертаване на кръговата диаграма, секторите, съответстващи на елементите от *explode* със стойности 1, се изобразяват изместени навън от центъра.

Задача: Да се изчертае кръгова диаграма на вектор

$$y_7 = [1 \quad 7 \quad 3 \quad 2 \quad 7 \quad 5 \quad 6 \quad 4].$$

Елементите на y_7, които имат стойност 7, да се изобразят на графиката изместени от центъра навън.



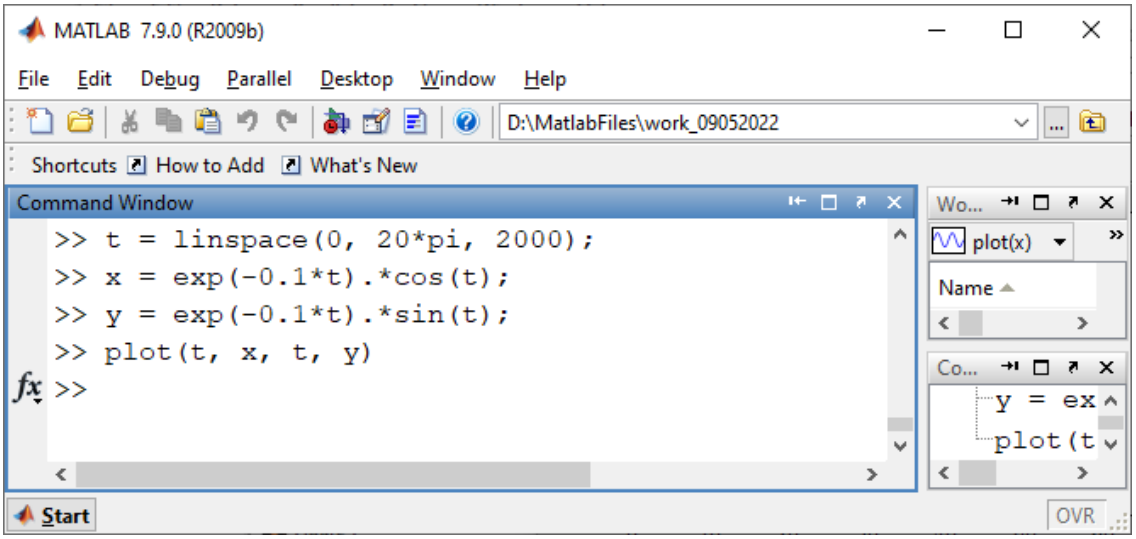
фиг. 3.19

В програмния продукт *Matlab* има възможност за допълнителна обработка на вече изчертаните графики. Това може да се направи от меню *Edit->Figure Properties* на графичния прозорец.

Задача: В един графичен прозорец, на една координатна система за 2000 стойности на t в интервала от 0 до 20π да се изчертаят графики на функциите:

$$x = e^{-0.1t}\cos t$$

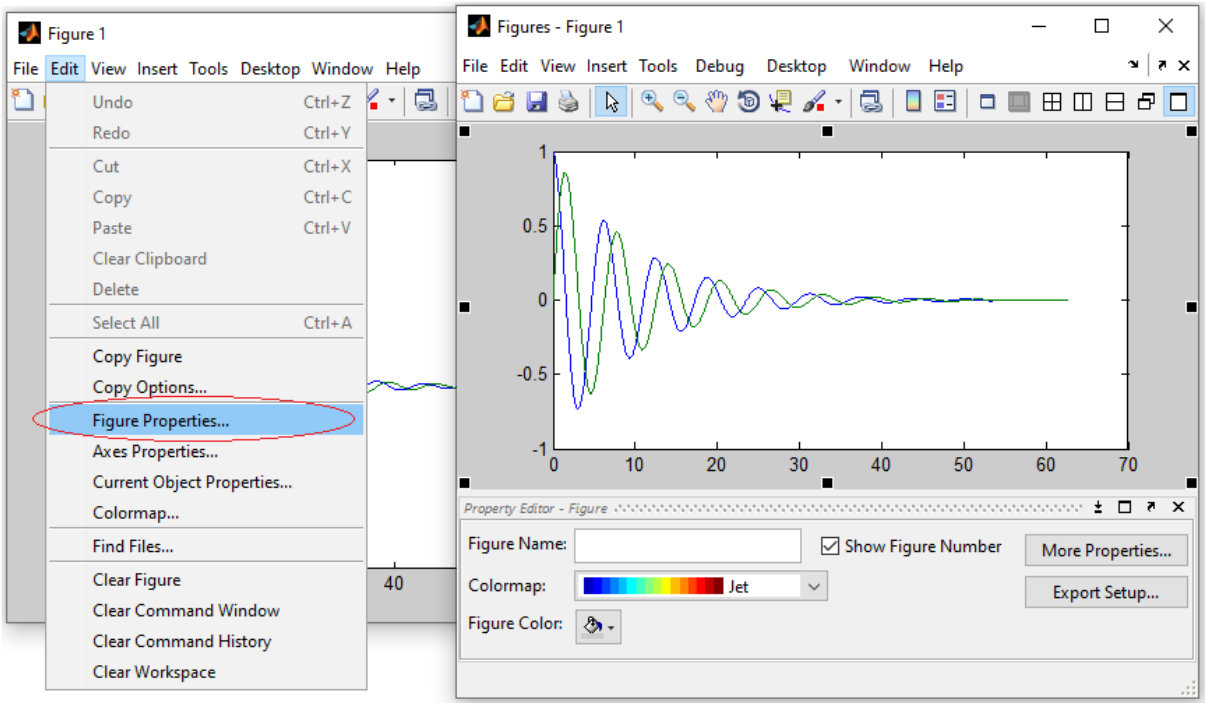
$$y = e^{-0.1t}\sin t$$



фиг. 3.20

За получените графики да се променят видът и цветът на линиите, както и видът на маркера. За тази цел от графичния прозорец се отваря меню *Edit->Figure Properties*.

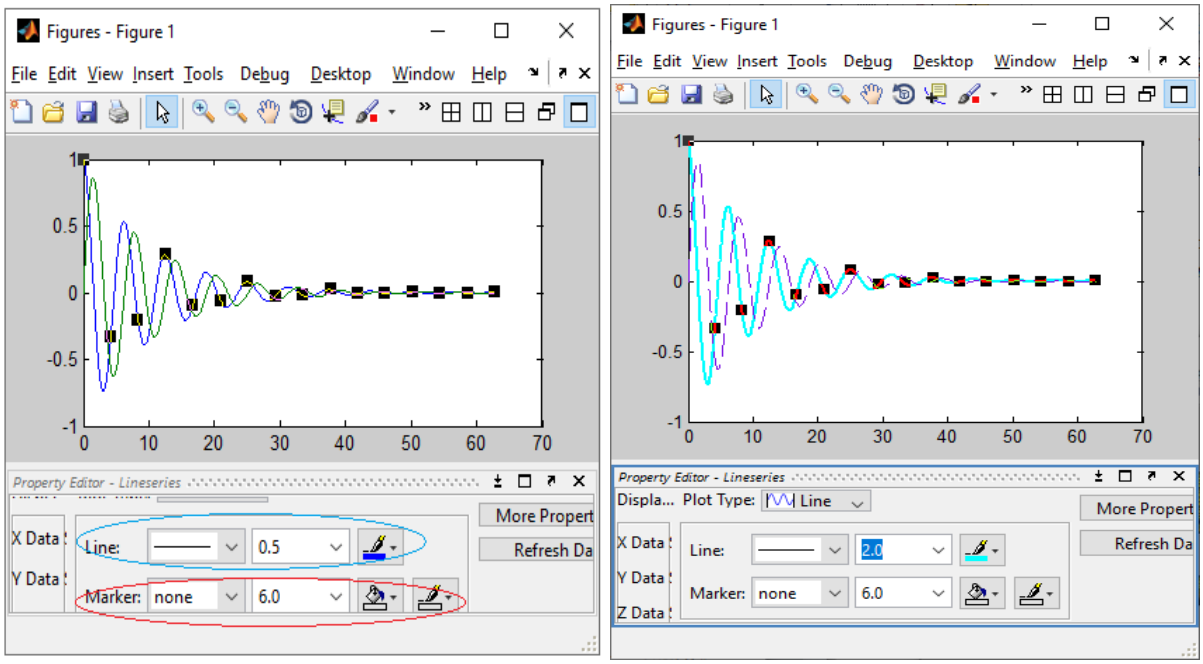
Това води до появата на прозореца с настройки за дадената фигура.



фиг. 3.21

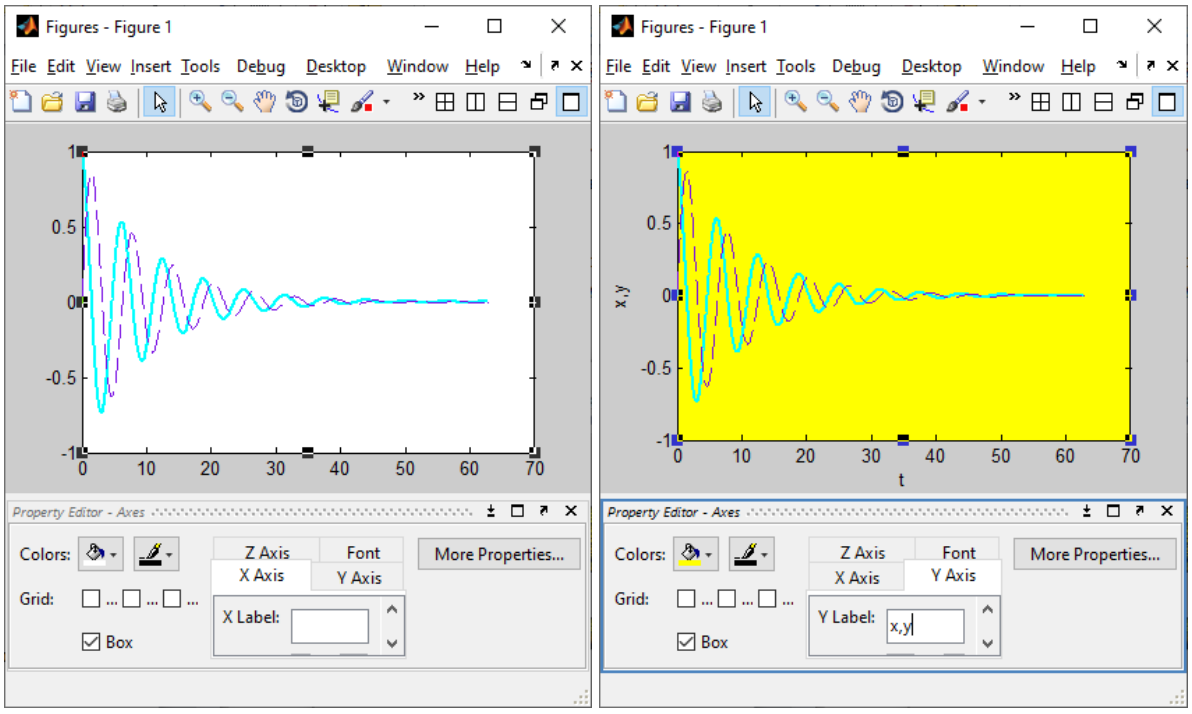
В този прозореца могат да бъдат променяни различни настройки. За да стане това е необходимо първо да бъде маркарана тази част от фигурата, за която се отнасят те. При маркиране на дадена графика могат да бъдат

прменяни вид, цвят и дебелина на линията, както и вид, цвят и дебелина на маркера. (фиг.3.22)



фиг. 3.22

По същият начин могат да бъдат променяни някои параметри на самата координатна система като цвят на фона или кооординатните оси, както и да бъдат поставяни различни надписи по координатните оси. (фиг.3.23)



фиг. 3.23

Въпроси и задачи за изпълнение:

1. Какъв ще бъде резултатът от изпълнението на команда plot когато като аргумент бъде зададена матрица с размерност 5 реда и 7 стълба?

2. С коя команда може да се изчертае графика в полярна координатна система?

3. Кои са командите за нанасяне на надписи върху графиките?

4. С кои команди може да се променя мащабирането на осите?

5. С коя команда може да се раздели графичният прозорец на определен брой подпрозорци? Какви са нейните параметри?

6. Да се намерят корените на полинома

$$p_6 = x^5 - 52x^4 + 912x^3 - 7022x^2 + 24191x - 30030$$

и получените стойности да се изчертаят със звездички в малинов цвят.

7. За стойности на *t* от 0 до 10π със стъпка $\frac{\pi}{10}$, да се изчертае графика на функцията $x_3 = 3\sin\frac{2t}{3}\cos\frac{2t}{3}$ в полярни координати. Графиката да бъде със зелена прекъсната линия.

8. На една координатна система да се изчертаят графики на функциите $x_1 = \sin t \cos t$ и $x_2 = \sin t \cos 3t$. За *t* да се зададат стойности от 0 до 2π със стъпка $\frac{\pi}{300}$. Функция x_1 да се изчертае с тирета и точки в червен цвят, а x_2 - със синьо-зелена прекъсната линия. Да се сложат надписи на графиката и легенда за така изобразените графики.

9. За стойности на t от 0 до 6π, стъпка 0.01, да се изчертае двумерна графика, за която като първи аргумент на командата да се зададе функцията $t\sin t$, а като втори - $t\cos t$. Графиката да се изчертае със зелени плюсчета.

10. Да се създаде матрица X, за която:

- елементите от първия стълб са решения на $e^{0,2x}\sin 3x$, за 1000 стойности на *x* в интервала от 0 до 10;
- елементите от втория стълб са решенията на $5e^{0,2x}\sin 2x$, за същите стойности на *x*.

На една координатна система да се изчертаят графиките на двете функции, зададени като елементи на матрица X.

11. В един графичен прозорец на три отделни координатни системи една под друга да се начертаят следните графики:

- за t = 0 до 6π, стъпка 0.01, да се изчертае графика на функцията $y = f(x)$, където $x = \sin t$, а $y = \cos t$ (по абсцисната ос се задават стойностите на *x*, а по ординатната – тези на *y*). Графиката да бъде изобразена в малинов цвят;
- $y_1 = e^{-t} arctgt$, като изобразяването да бъде с червена прекъсната линия;
- за t = 0 до 6π, стъпка 0.1, да се изчертае стъпаловидна графика в зелен цвят на функцията $y_2(t) = e^{-\frac{t}{3}} \sin 2t$.

12. Да се изчертае графика на дискретната функция *sint*, за стойности на t от 0 до 2π със стъпка $\frac{\pi}{10}$. Задайте на функцията параметъра, с който кръгчетата на изчертаната графика да са запълнени.

13. Да се визуализира траекторията на затихващ осцилатор, представена с уравненията:

$$x = e^{-0.05t} \cos t$$

$$y = e^{-0.05t} \sin t$$

За t да се зададат 1000 стойности, равномерно разпределени в интервала от 0 до 20π. За визуализация на графиката да се използва команда *comet(x, y)*.

Лабораторно упражнение 4
Визуализация на функции в тримерното пространство в програмния продукт *Matlab*.

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите на програмния продукт *Matlab* за изчертаване на различни тримерни графики.

Тримерни графики в Matlab

Matlab разполага с голям брой разнообразни средства за визуализация на данни и функции в тримерно пространство. Чрез различни команди потребителят може да изчертава повърхнини, пространствени криви и мрежови графики, както и да управлява изгледа, осветлението и цветовия изглед на графиките. Тримерните графики позволяват по-добро разбиране на сложни зависимости между няколко променливи. Те намират широко приложение при моделиране, анализ на динамични системи и интерпретация на резултати от числени изчисления.

В това упражнение са представени някои основни и най-често използвани графични команди за изчертаване на тримерна графика – пространствени криви, повърхнини и др.

Тримерната графика в *Matlab* е разделена на две групи – обикновена и специална.

Пълен списък на всички графични команди и функции се извежда с командите:

help graph3d - обикновена тримерна графика

help specgraph - специална графика

При тримерните графики могат да се използват почти всички функции за управление на осите и за нанасяне на надписи, които се използват при двумерните графики.

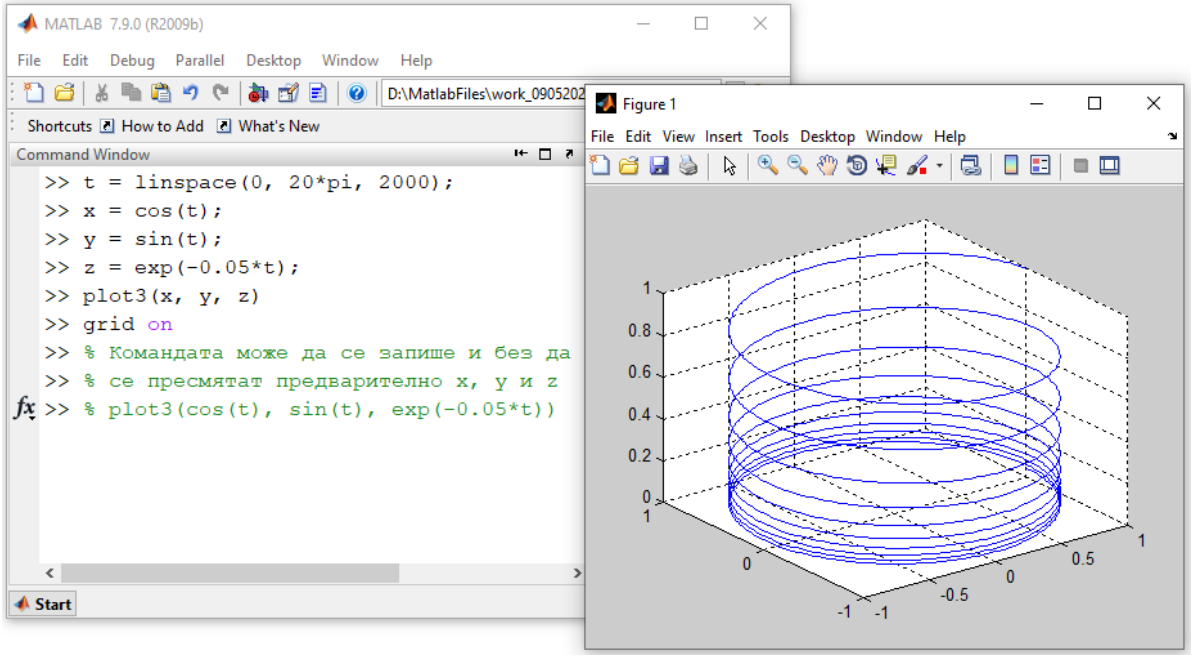
Командите в *Matlab* за тримерни графики са свързани с изчертаване на криви в тримерното пространство или с изчертаване на повърхнини, описвани с дадена функция.

Аналог на командата за изчертаване на двумерна графика *plot* е функцията *plot3*

plot3(x,y,z),

чрез която се изчертава графика на пространствена крива, зададена в параметричен вид;

Задача: За 2000 стойности на t в изтервала от 0 до 20π, да се изчертае пространствена крива, описвана с $x=\cos(t)$, $y=\sin(t)$, и $z=e^{-0.05t}$. На графиката да се добави и мрежа.



фиг. 4.1

За изобразяване на повърхнини, описвани с функцията z = f (x, y), могат да се използват командите:

- *mesh* – изобразява повърхнината чрез мрежа;
- *surf* – изобразява графиката чрез непрекъсната повърхнина;
- *surfl* – изчертава се графиката с допълнително осветяване.

Построяването на графиката става в следната последователност:

- ✓ С помощта на функцията *meshgrid* се пресмятат матриците X и Y от координатите x и y на възлите на мрежата:

[X, Y] = meshgrid(x, y) – преобразува равнината, описана от векторите x и y в масиви X и Y. Редовете на изходния масив X са копия на вектор x, а стълбовете на изходния масив Y са копия на вектор y;

✓ Пресмята се матрицата Z със стойностите на функцията във всички възли:

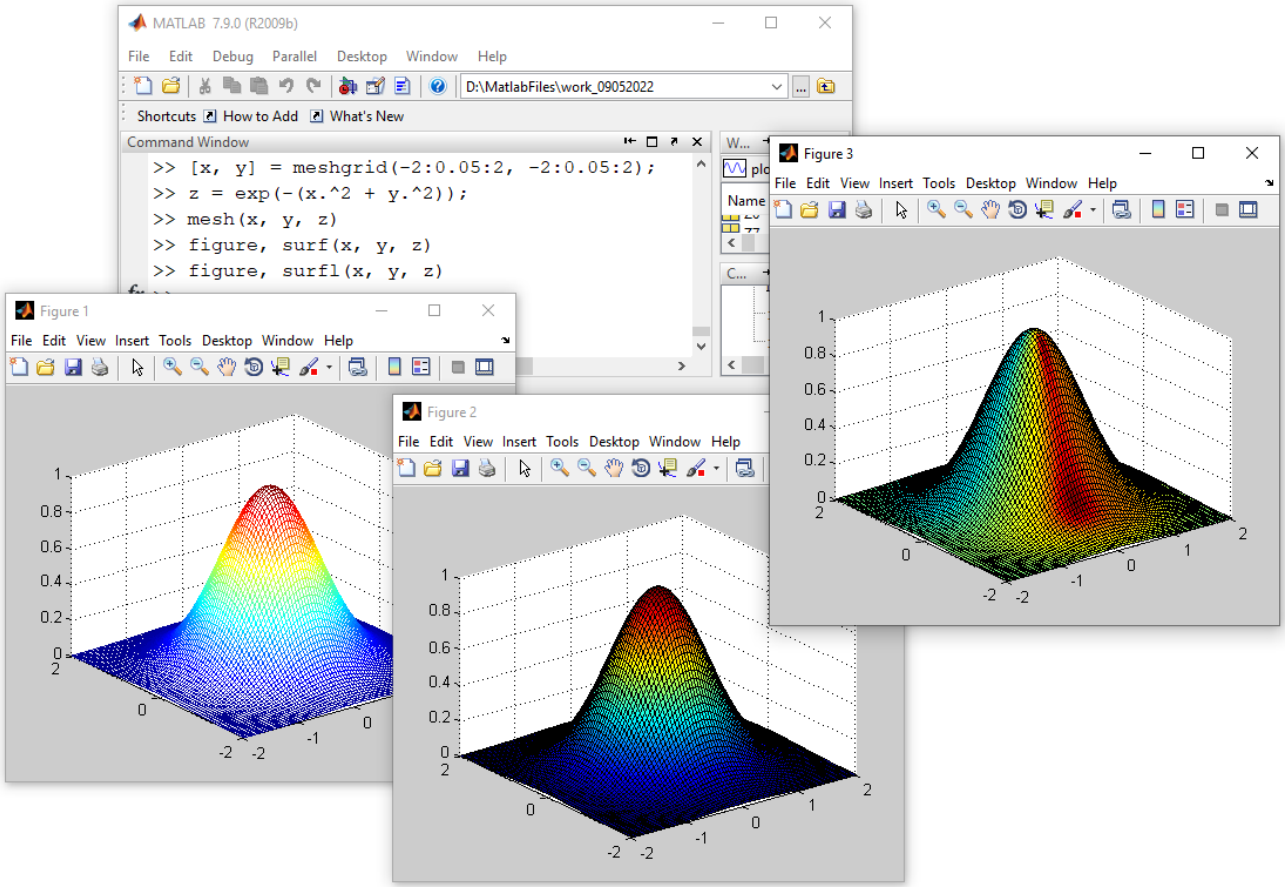
Z = F(X, Y), където F е конкретна функция, въведена с използване на поелементните оператори .*, ./, .^ ;

✓ Извиква се една от трите функции *mesh*, *surf* или *surfl* за построяване на графиката:

mesh(X, Y, Z); surf(X, Y, Z); surfl(X, Y, Z).

Задача: Да се изчертае повърхнината, описвана с функцията $z = e^{-x^2-y^2}$, за стойности на *x* и *y* в интервала от -2 до 2 и стъпка на изменение 0.05.

За изчертаване на повърхнината да се използват трите посочени по-горе функции, като трите варианта се представят в три отделни графични прозореца (команда *figure*, лаб.упр. №3).



фиг. 4.2

Допълнителни функции за оформяне на графиката

Допълнителните функции за нанасяне на надписи към графиката (заглавие, надписи по координатните оси) са аналогични на тези, използвани при двумерната графика:

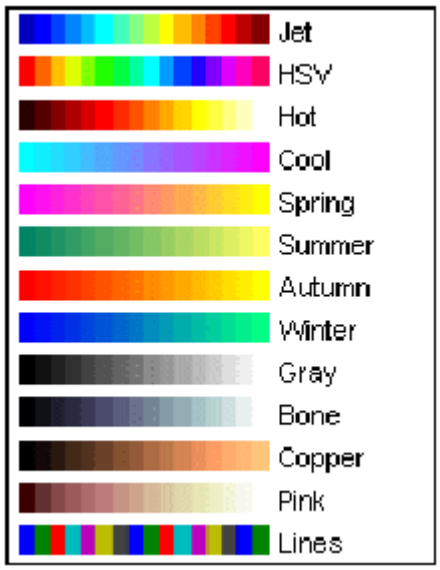
- *title*('text') – изписва заглавие в горната част на графиката;
- *xlabel* ('text') – изписва *text* по оста x;
- *ylabel* ('text') – изписва *text* по оста y;
- *zlabel* ('text') – изписва *text* по оста z;

Задача: На фигурите от последната задача поставете надписи по трите координатни оси, съответно - x, y, z .

При изчертаване на повърхнини в програмния продукт *Matlab*, може да се променя цветовата гама, в която дадената повърхнина да бъде визуализирана. Това става с помощта на команда

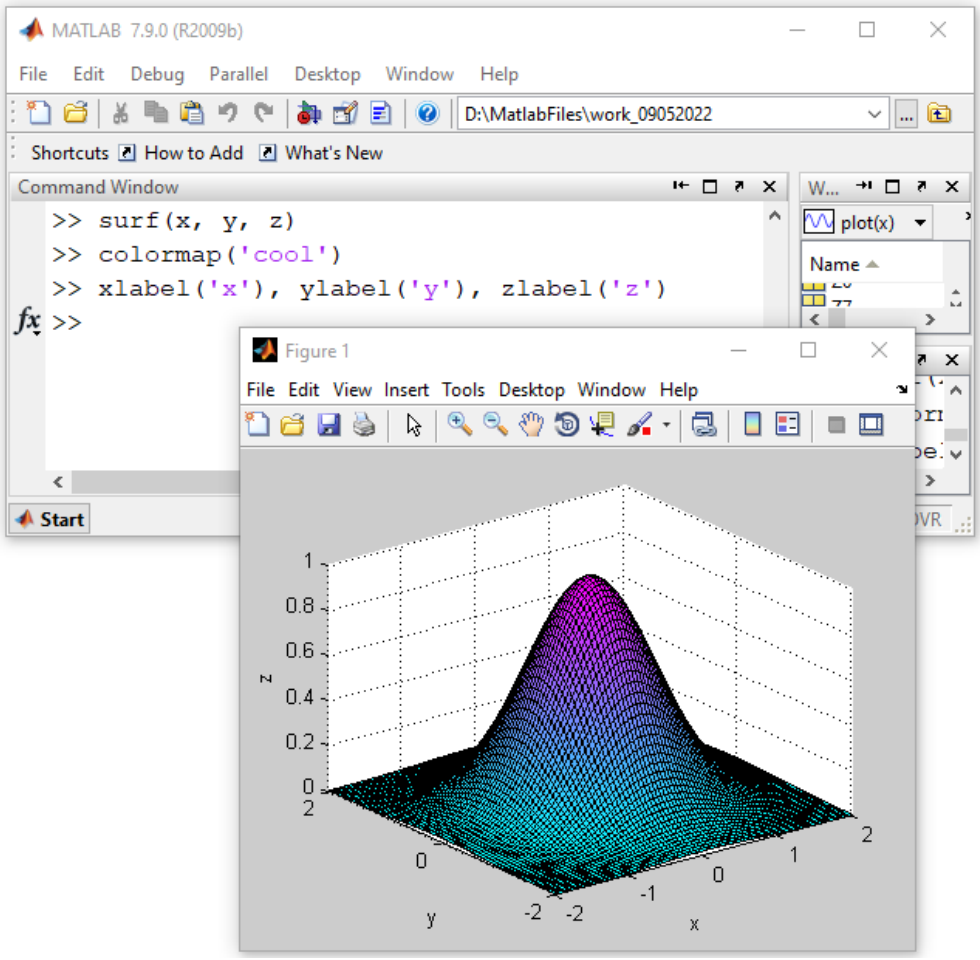
colormap ('colmap')

Чрез аргумента *'colmap'* може да се зададе цветова палитра, определяща оцветяването на изображението. Стойността на аргумента може да бъде *grey*, *copper*, *pink*, *spring*, *summer* и т.н. (фиг.3.).



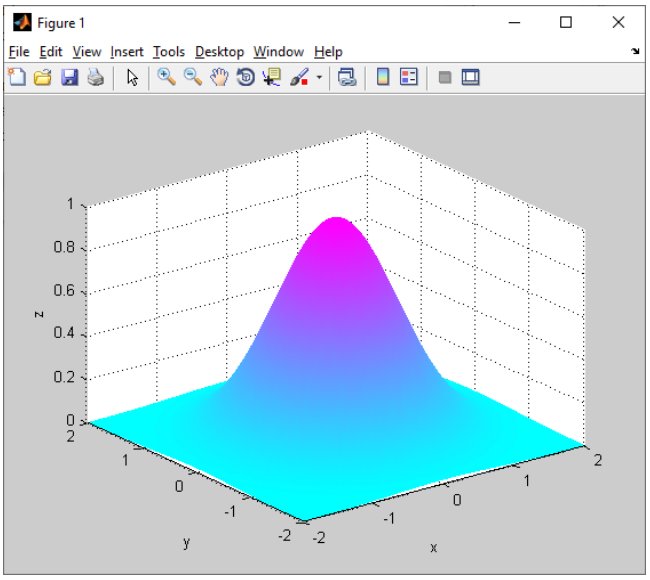
фиг. 4.3

Задача: Да се промени по избор цветовата палитра (напр. *cool*) на повърхнината от предходната задача ($z = e^{-x^2-y^2}$) и да се сложат надписи по трите координатни оси.



фиг. 4.4

В Matlab има команда, чрез която изчертаваната повърхнина може да се визуализира с плавен преход между цветовете, без рязка промяна на цветовете между съседните върхове на мрежата. Това е командата *shading interp* и се използва без аргументи.

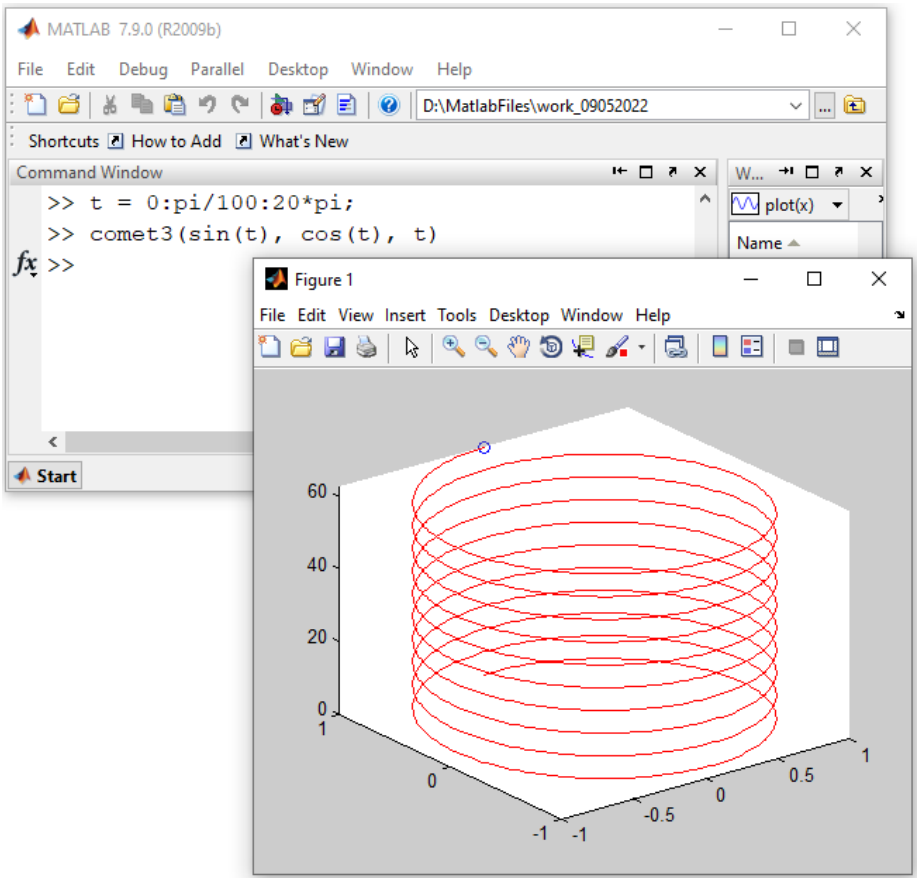


фиг. 4.5

В програмния продукт *Matlab* има възможност и за анимирано изчертаване на крива в тримерното пространство. За тази цел се използва команда

comet3(x, y, z)

Задача: За стойности на t от 0 до 20π със стъпка на изменение $\frac{\pi}{100}$, да се изчертае анимирано пространствената крива, описвана с $x=\cos(t)$, $y=\sin(t)$ и $z=t$.



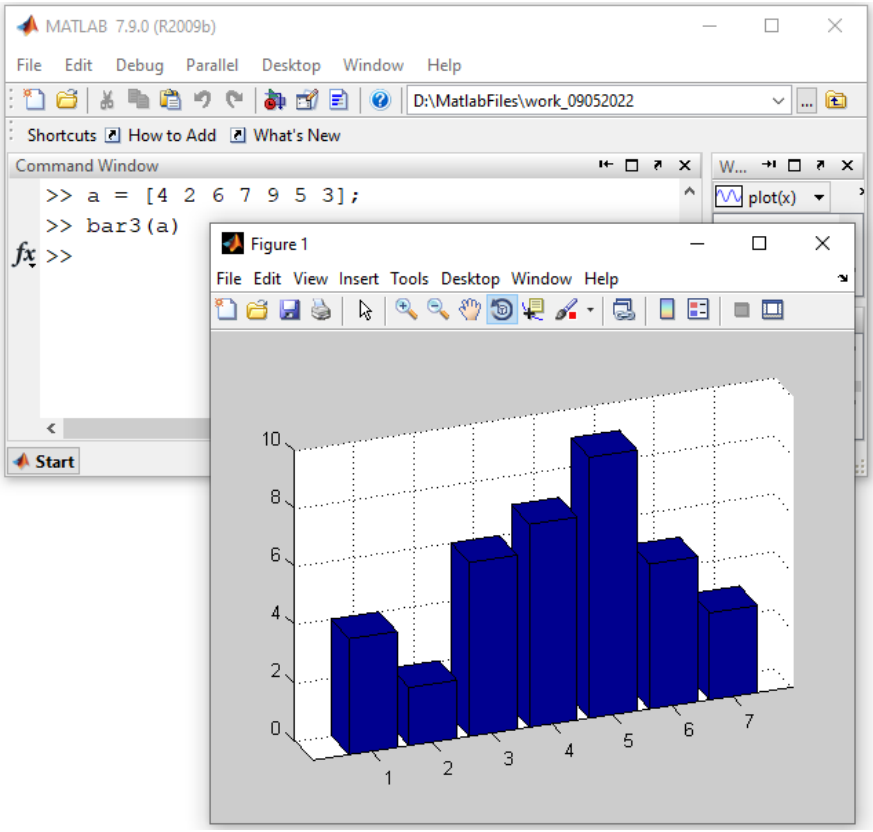
фиг. 4.6

Аналог на командата за изчертаване на лентова графика (bar), е командата

bar3(y)

С използването ѝ, графиката се изчертава с блокчета вместо с ленти.

Задача: Да се изчертае тримерна лентова графика, изобразяваща елементите на вектор $q=[4 \quad 2 \quad 6 \quad 7 \quad 9 \quad 5 \quad 3]$.

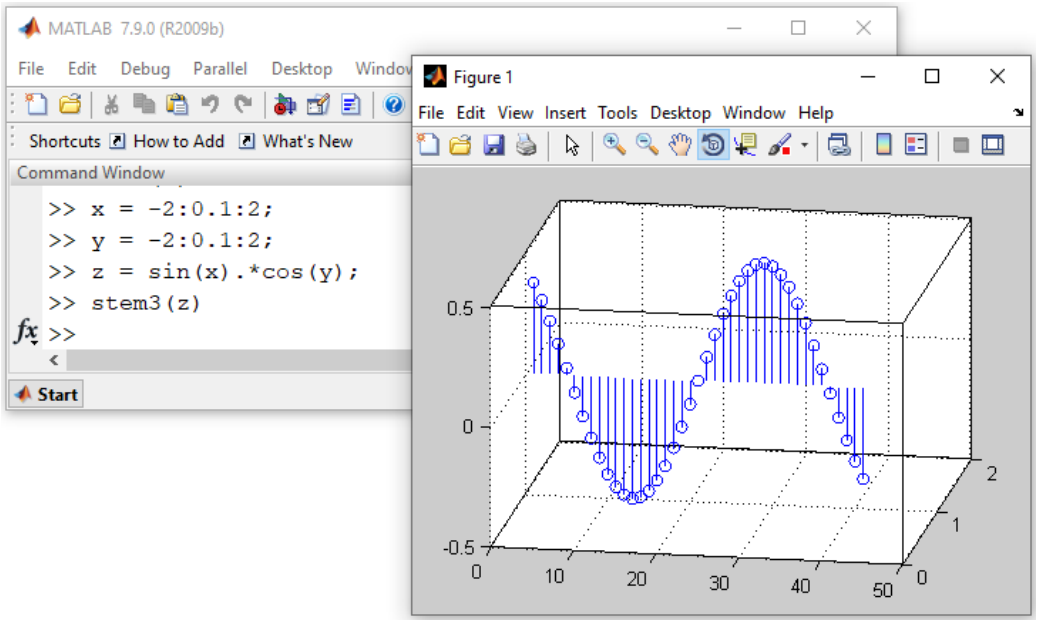


фиг. 4.7

За изобразяване на дискретна функция *z=f(x,y)* в тримерен вид, се използва команда

stem3(z)

Задача: Да се изчертае графика на дискретната функция $f = \sin x \cos y$, за стойности на *x* и *y* от -2 до 2 и стъпка на изменение 0.1.



фиг. 4.8

В програмния продукт *Matlab* има възможност и за изчертаване на тримерна кръгова диаграма. Командата, с помощта на която може да бъде извършено това, е:

pie3(x)

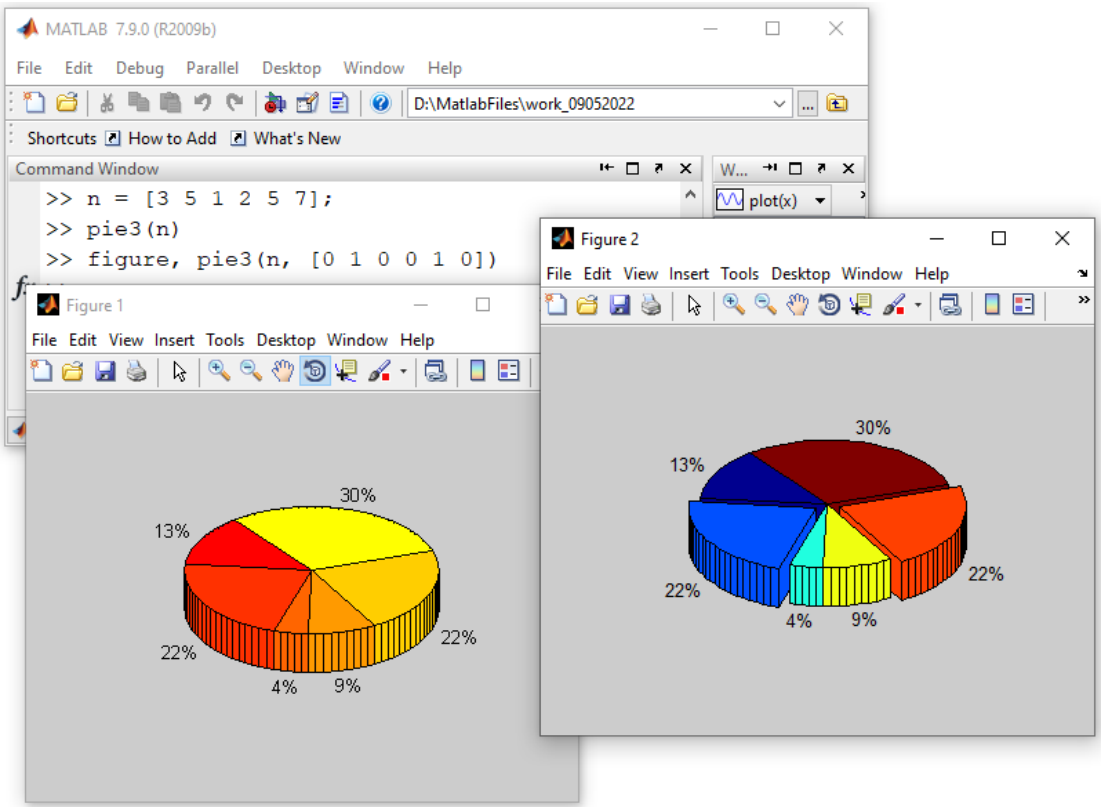
В този случай се изчертава тримерна кръгова диаграма в съответствие на стойностите на вектор *x*.

И тук може да бъде добавен втори аргумент на командата:

pie3(x, explode)

при което добавеният арумент *explode* е вектор с дължината на *x*, който се състои само от нули и единици. При изчертаване на кръговата диаграма, секторите, съответстващи на елементите от *explode* със стойности 1, се изобразяват изместени от центъра навън.

Задача: Да се изчертае тримерна кръгова диаграма на вектор $m=[3 \quad 5 \quad 1 \quad 2 \quad 5 \quad 7]$. В нова фигура да се изчертае същата диаграма, но секторите, съответстващи на елементите със стойност 5, да се изчертаят изместени навън от центъра на диаграмата.

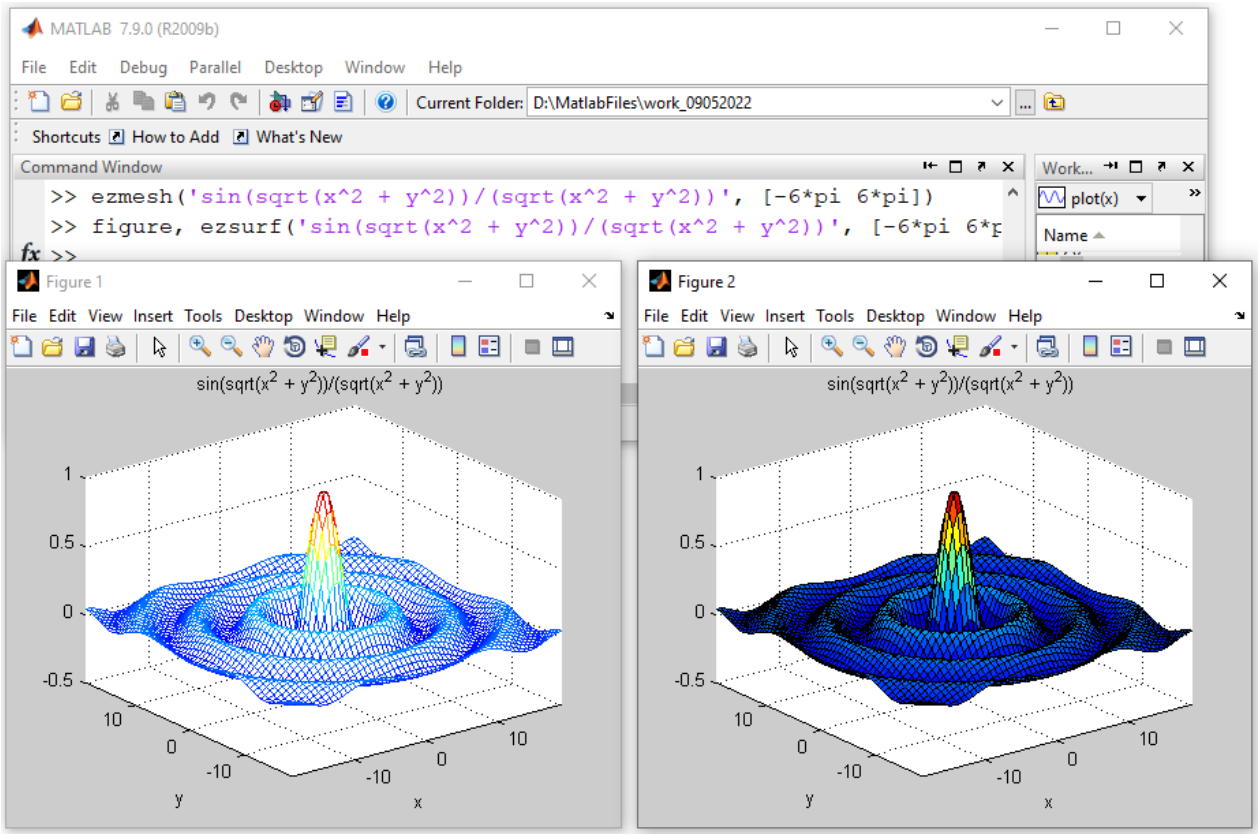


фиг. 4.9

В *Matlab* има и една група команди, които се използват за „облекчено“ изчертаване на повърхнини, зададени с аналитичните си изрази. Това са командите *ezmesh*, *ezsurf* и *ezgraph3*. Синтаксисът на първите две е:

- *ezmesh('f(x, y)')* – изчертава чрез *mesh* повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала $(-2\pi, 2\pi)$;
- *ezmesh('f(x, y)', [a b])* – изчертава чрез *mesh* повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала (a, b);
- *ezmesh('f(x, y)', [xmin xmax ymin ymax])* - изчертава чрез *mesh* повърхнината $f(x,y)$ за стойности на *x* в интервала *(xmin, xmax)* и на *y* в интервала *(ymin, ymax)*;
  - *ezsurf('f(x, y)')* – изчертава чрез *surf* повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала $(-2\pi, 2\pi)$;
  - *ezsurf('f(x, y)', [a b])])* – изчертава чрез *surf* повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала *(a, b)*;
  - *ezsurf('f(x, y)', [xmin xmax ymin ymax])* - изчертава чрез *surf* повърхнината $f(x,y)$ за стойности на x в интервала *(xmin, xmax)* и на y в интервала *(ymin, ymax)*.

Задача: Да се изчертае с командите *ezsurf* или *ezmesh* повърхнината, описвана с уравнението $z = \dfrac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}$ за интервала $(-6\pi, 6\pi)$.



фиг. 4.10

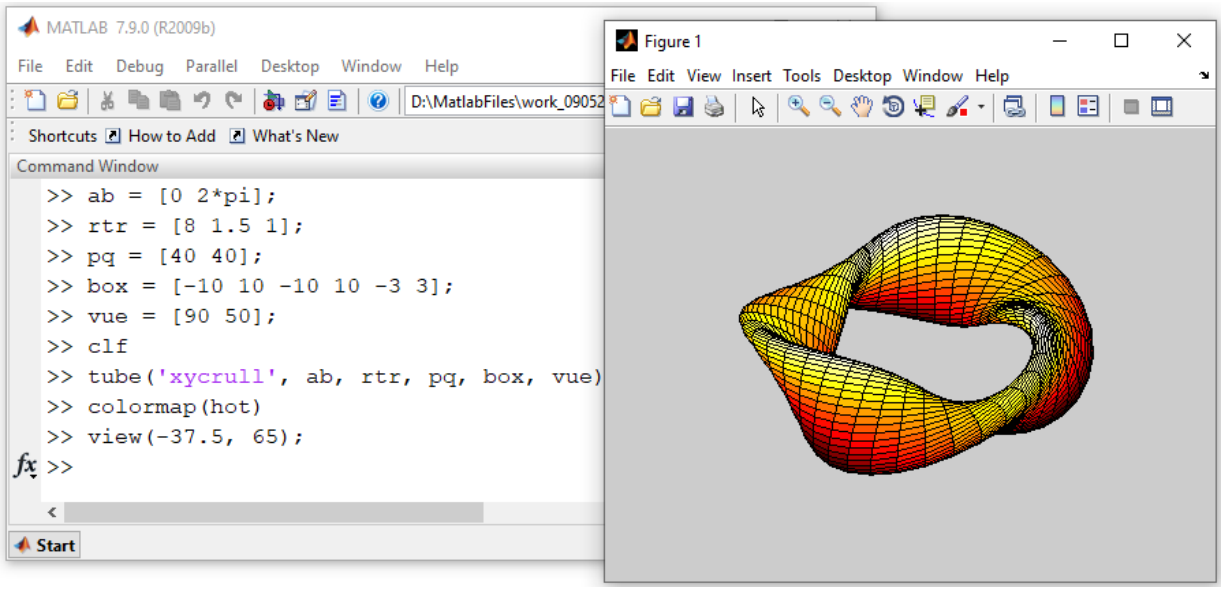
Третата команда включва в себе си първите две. Тя се използва за „облекчено“ изчертаване на повърхнини, като се използва една от функциите – mesh, surf или surfl, което се задава като първи аргумент на командата (*plotfun*).

- *ezgraph3('plotfun', 'f(x, y)')* – изчертава чрез '*plotfun*' повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала $(-2\pi, 2\pi)$;

- *ezgraph3('plotfun', 'f(x, y)', [a b])* – изчертава чрез '*plotfun*' повърхнината $f(x,y)$ за стойности на *x* и *y* в интервала *(a, b)*;

- *ezgraph3('plotfun', 'f(x, y)', [xmin xmax ymin ymax])* - изчертава чрез '*plotfun*' повърхнината $f(x,y)$ за стойности на *x* в интервала *(xmin, xmax)* и на *y* в интервала *(ymin, ymax)*.

В *Matlab* има възможност за изчертаване и на различни специални графики, като например параметрична крива на Edward, за което се използва функцията *tube*.

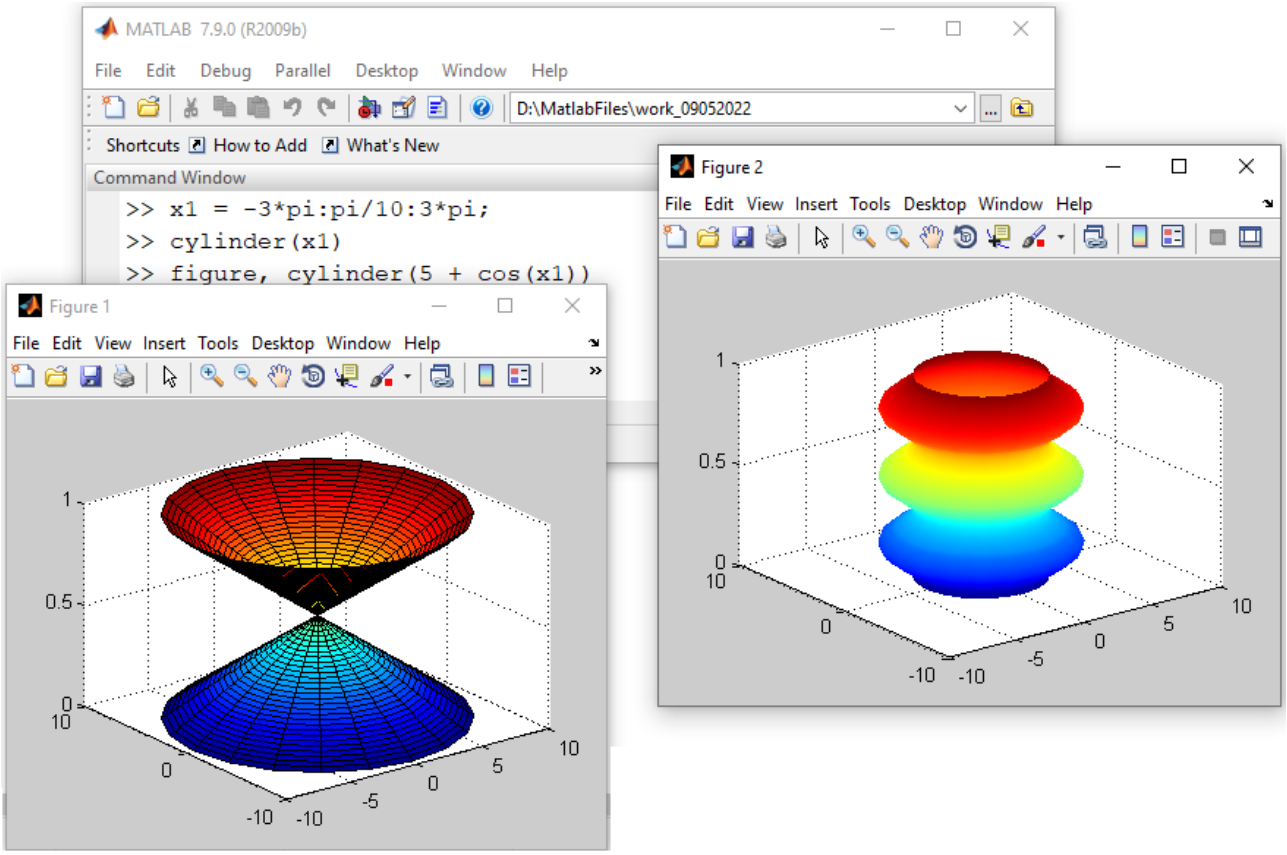


фиг. 4.11

В програмния продукт *Matlab* има вградени и команди, с помощта на които могат да бъдат изчертавани тримерни графики на сфера, елипсоид, цилиндър и др.

Например, с командата *cylinder* се изчертава цилиндър с радиус $r = 1$. При задаване на аргумент на командата, могат да бъдат изчертани разновидности на цилиндър в тримерното пространство.

Задача: За стойности на x в интервала $(-3\pi, 3\pi)$ и стъпка на изменение интервала $\frac{\pi}{10}$, с помощта на команда *cylinder(x)* да се изчертае графика на повърхнината. За същите стойности на x да се изчертае графика като на командата се подаде аргумент $5+\cos(x)$.



фиг. 4.12

Въпроси и задачи за изпълнение:

1. Каква е последователността за изчертаване на повърхнини в Matlab?

2. Каква е командата за анимирано изчертаване на крива в тримерното пространство?

3. Коя е командата, с която може да се смени цветовата гама на изчертаваната повърхнина?

4. Да се изчертае тримерна графика на единична матрица с размерност *n = 15*.

5. Да се изчертае тримерна графика на магически квадрат с размерност *n = 15*.

6. Да се изчертае тримерна повърхнина на функцията $z = \sin(y)\cos(x)$, за стойности на *x* и *y* в интервала от *-6 до 6*, стъпка *0.1*.

7. Да се изчертае повърхнината, описвана с уравнение $f = \frac{-6.7x_3 - 6.7x_2^2 + x_3x_2^2 + x_3^2}{4x_2}$

- ✓ за стойности на x_2 и x_3 интервала $(-2\pi, 2\pi)$. Цветовете на повърхнината да са тип *autumn*.
- ✓ за стойности на x_2 и x_3 интервала $(-100, 100)$. Цветовете на повърхнината да са тип *spring*, а графиката да се представи в нов прозорец.

8. Да се изчертае повърхнината, описвана с уравнението $x_3 = 10(x_1^2 - 2.67x_2)$, за x_1 в интервала (-20, 20) и x_2 в интервала (-10, 30).

9. За стойности на *x* в интервала от -50 до 50 със стъпка на изменение 0.01, да се изчертае пространствена крива, описвана с $x = t$, $y = \sin(t)$, $z = t^2 + y^2$. На графиката да се добави и мрежа.

Лабораторно упражнение 5
Програмиране в средата на *Matlab*.
Създаване на script файлове и файл-функции.
Управляващи оператори

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с различните видове програмни файлове, които могат да бъдат създавани в програмния продукт *Matlab*, с управляващите оператори и техния синтаксис.

Програмиране в Matlab

Matlab разполага с пълен език за програмиране, който позволява на потребителя да пише програми. Програмата представлява последователност от оператори на *Matlab*, записани във файл. Тези оператори се изпълняват като в командния прозорец се напише името на файла и се натисне <Enter>– т.е. с една команда.

Програмното решаване на задачи в *Matlab* се извършва чрез последователно интерпретиране на команди (оператори) от езика на *Matlab*. В *Matlab* основният начин за съхраняване и организиране на програмен код са файловете с разширение *.m*. Те позволяват автоматизиране на изчисления, повторно използване на код и структуриране на по-сложни задачи.

Името на файла е от вида - *filename.m*, т.е. има разширение *m* и затова се нарича *m-файл*. Редакторът за създаване на нов програмен файл се стартира след избиране на командата *File/New/Blank M-File*.

M-файловете могат да бъдат скриптове (*scripts*) или функции (*function*).

В зависимост от предназначението си *m*-файловете се разделят основно на скриптове (съдържат последователност от команди, които се изпълняват в текущото работно пространство) и функции (реализират самостоятелни програмни блокове с входни и изходни аргументи).

Скриптовете са най-простата разновидност на *m-файл*. Те се използват за автоматизиране на последователността от команди на *Matlab*, като изчисления, които се повтарят при изпълнението им от командния ред. Всички променливи, които скриптът създава, остават в работното

пространство след като завърши изпълнението на скрипта и потребителят може да използва променливите за следващи изчисления. Script файловете не използват входни и изходни параметри (аргументи), оперират с данни от работната област (Workspace) и генерират резултати (данни), които се съхраняват и могат да се ползват отново в работната област.

Файл-функциите съдържат входни и изходни параметри. Това са променливи, с които се предават данни за изчисление на стойност (стойности) на функцията – входни параметри и променливи (или само една променлива), с които се извежда резултат от изчислението на функцията – изходни параметри (или само един изходен параметър). Вътрешните им променливи се явяват локални по отношение на функцията и не могат да се използват в работното пространство.

Стъпките при създаването и използването на *m-файлове* са:

1. Създаване на файла в текстов редактор и записване под определено име и разширение *.m*

2. Стартиране (извикване) на файла от командния ред в *Command Window* или в друг М-файл.

Особености на Script файловете

- *Script* файловете са независимо изпълняеми блокове от оператори и команди;
- В *Script* файловете променливите образуват т.нар. работна област. Стойностите на променливите в този случай се запазват в оперативната памет и при изпълнение на следващия Script файл те са валидни и могат да се използват. В някои случаи това е неудобство, защото програмистът трябва да се съобразява с имената на използваните до момента променливи.
- Липсва заглавие на Script файловете;
- Коментарните редове започват с < % >;
- Коментарните редове преди първия изпълним оператор се възприемат като описатели. Ако е стартирана командата *help име_на_файла*, тогава съдържанието на тези описателни редове се извежда в командния прозорец;
- В *Matlab* няма оператор за край на програмата;

- Обръщението към Script файла не изисква никакви имена на променливи (входни или изходни).

Задача: Да се състави Script файл, с който да се пресметне функцията $y = \dfrac{a\sin^2(\pi) + \sqrt{\sin^3\left(\dfrac{\pi}{2}\right) - \cos^3\left(\dfrac{\pi}{2}\right)}}{b}$ за стойност на $a = 0.7$ и $b = 2$.

Стъпки за изпълнение на задачата:

1. Отваряне на нов файл - *File/New/Blank M-File*.

2. Задаване стойности за *a* и *b*.

3. Дефиниране на функцията *y*.

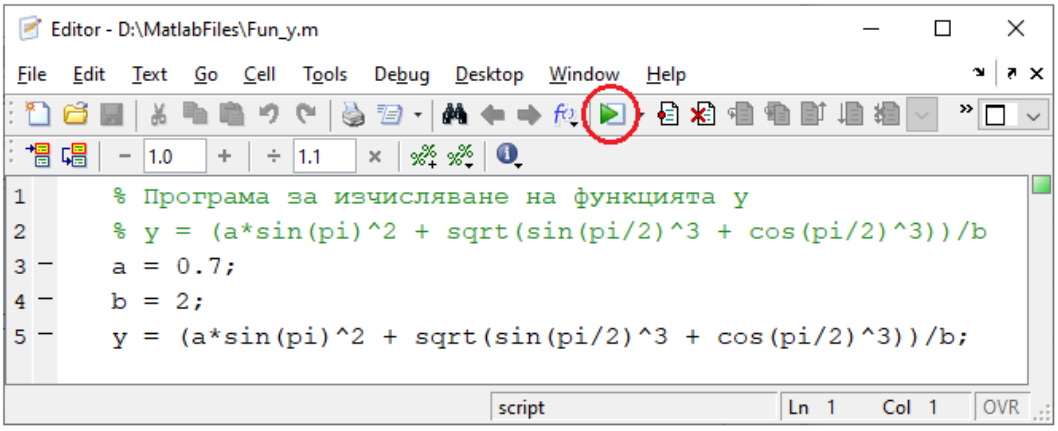
4. Записване на файла с име *Fun_y* - *File/Save As* или с натискане на иконата *Save*.

5. Стартиране на файла.

Стартирането на файла може да стане по два начина:

✓ В *Command Window* на командния ред (>>) се въвежда името на Script файла (без разширение) - *Fun_y* и след това се натиска клавиша Enter;

✓ Натиска се бутона Run от лентата на редактора.



фиг. 5.1

И в двата случая след изпълнението на програмата стойностите на променливите *a* и *y* се появяват в работното пространство (Workspace) и в Command Window (ако няма ; в края на редовете).

Особености на файл-функциите

Разликата между двата вида файлове е в заглавния ред на файл-функцията, който изглежда по следния начин:

function [списък на изх.променливи] = име на функцията (списък на вх. променливи)

Името на файла се състои от името на функцията с разширение .m!

Всички имена на входно-изходните променливи са локални! Т.е. техните стойности са валидни само до завършване изпълнението на функцията.

Създаване на файл- функции

Ако списъкът на изходните променливи съдържа само една променлива, то заглавният ред е от вида:

function име на изх. променлива = име на функцията (списък на вх. променливи)

Ако резултатът трябва да се формира в повече от една изходни променливи, то заглавният ред трябва да бъде от вида:

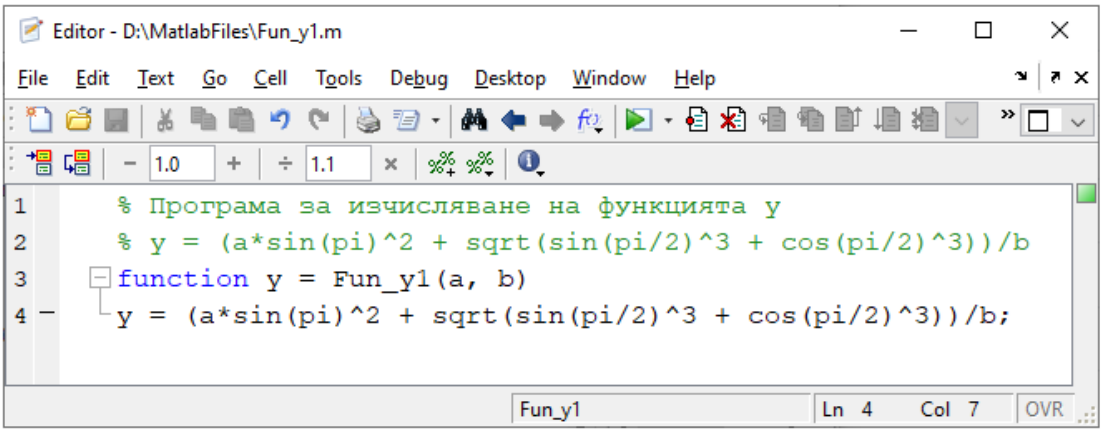
function [y1, y2, ..., yn] = име на функцията (списък на вх. променливи)

т.е. списъкът от имената на изходните променливи трябва да се представи като вектор, като всеки от елементите му може да бъде матрица.

Задача: Да се състави файл-функция за изчисляване на функцията:

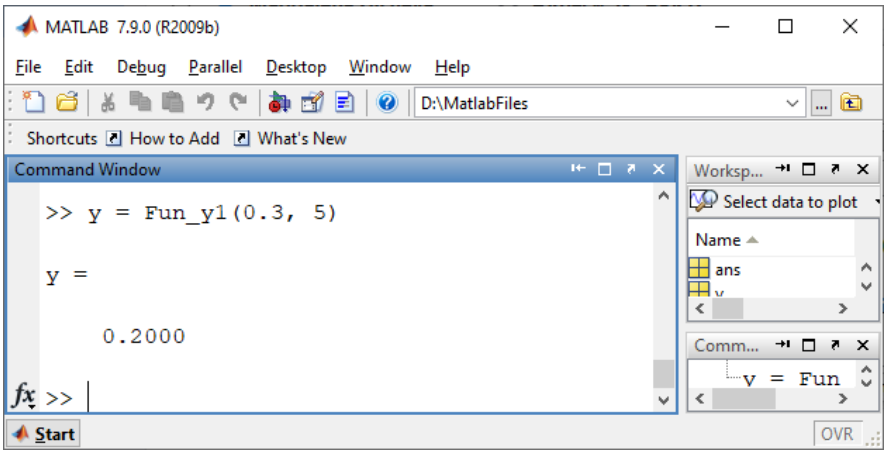
$$y = \frac{a\sin^2(\pi) + \sqrt{\sin^3\left(\frac{\pi}{2}\right) - \cos^3\left(\frac{\pi}{2}\right)}}{b}.$$

Стойностите за a и b да се задават като аргументи на функцията.



фиг. 5.2

Името на файла задължително трябва да бъде *Fun_y1.m*, а извикването на функцията трябва да стане със задаване на числени стойности на параметри *a* и *b* като нейни аргументи.



фиг. 5.3

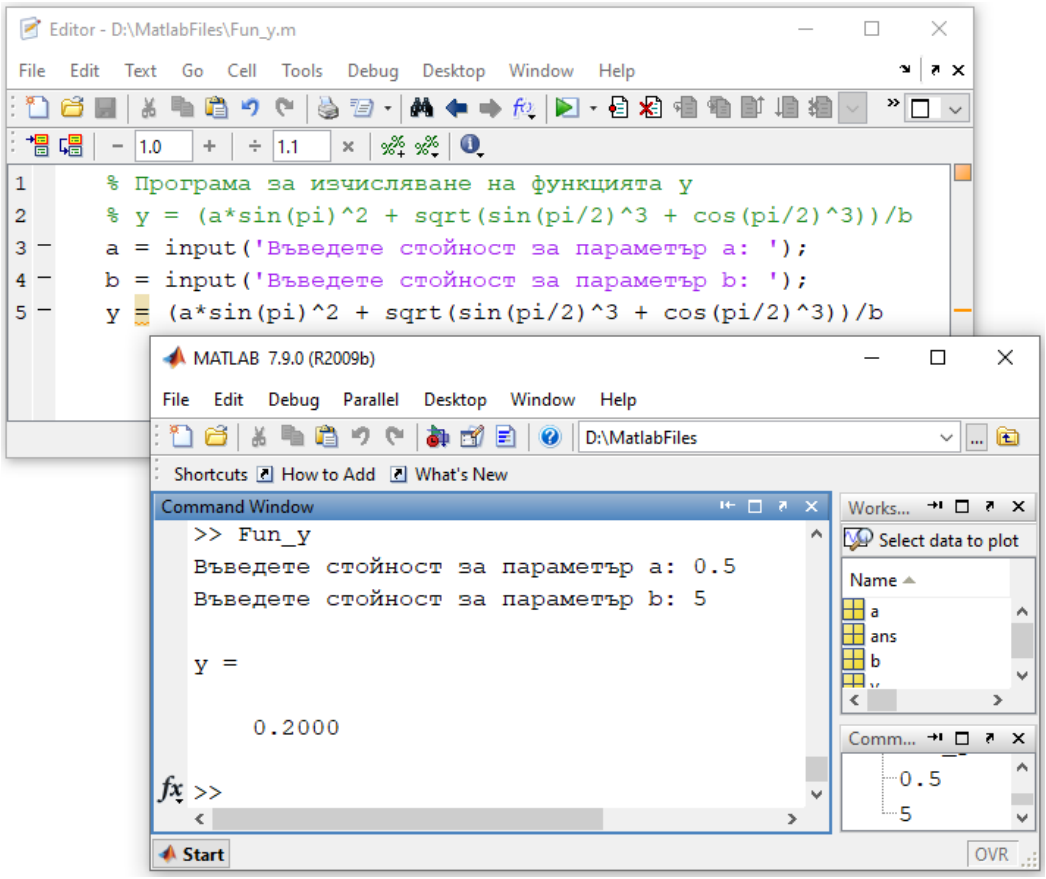
Въвеждане на информация в диалогов режим

Въвеждането на данни в диалогов режим може да се извърши чрез команда input.

променлива = input('Въведете стойност на променлива : ')

Задача: Да се промени файл *Fun_y* така, че стойностите за *a* и *b* да се задават от клавиатурата в диалогов режим.

Да се стартира файлът няколко пъти, като се задават различни стойности за двете променливи.



фиг. 5.4

Извеждане на информация в командния прозорец

За да бъде изведена информация в командния прозорец могат да бъдат използвани две различни команди – *disp* и *fprintf*. Разликата между тях е в задаването на аргументите.

✓ disp – извежда в командния прозорец текста или името на променливата, зададени като аргумент на функцията.

disp('текст')

Ако трябва да се визуализира числена стойност на дадена променлива, то тя трябва да бъде преобразувана в текст.

disp(['текст', num2str(променлива)])

% disp(['Стойността на x е ', num2str(x)])

✓ *fprintf*('x = %g',x) – обединява текст ('x =') с числова стойност на променлива (x) и ги извежда в командния прозорец на *Matlab*.

За разлика от *disp*, тук не е необходимо преобразуване на числените стойности в текстов вид, но пък трябва да бъде зададен видът, в който да се

изведат те – като цяло число, десетично, в експоненциален вид и т.н. Вариантите за избор са:

%d – десетично число;

%i – цяло число;

%e – число в експоненциална форма;

%f – число с плаваща запетая;

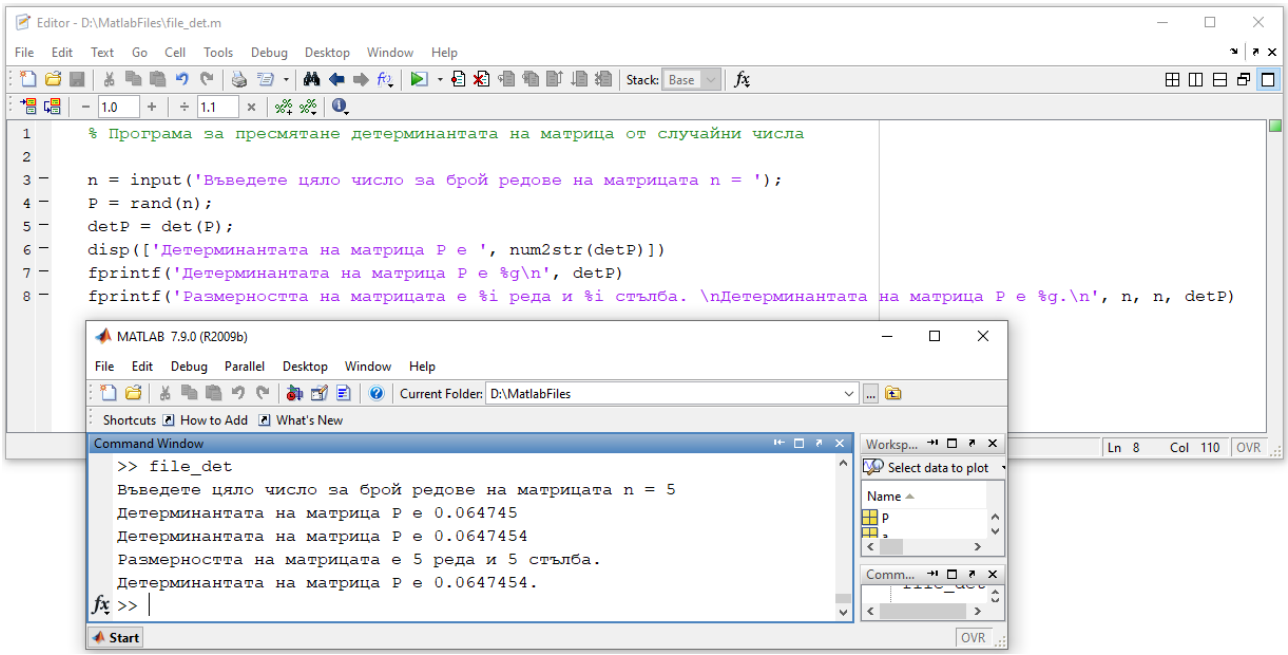
%g – по-компактния вариант между %e и %f;

%s – стринг;

Друга особеност при използването на *fprintf* е, че задължително в края на записа трябва да бъде указано преминаването на нов ред. Ако това не бъде направено, следващата информация ще бъде визуализирана непосредствено след текста, зададен като аргумент на командата.

\n – преминаване на нов ред.

🤔 ***Задача***: Да се създаде m-файл, в който да се генерира матрица *P* от случайни числа с размерност *n* реда и *n* стълба и да се намери детерминантата на P, стойността на която, заедно с пояснителен текст, да се изведе в командния прозорец. Размерността на матрицата n да се въвежда в диалогов режим.



фиг. 5.5

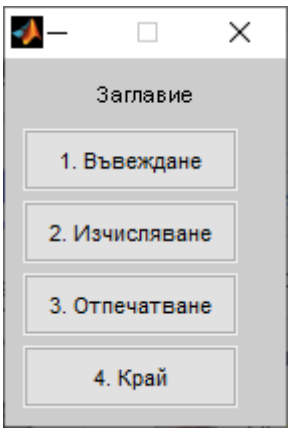
Използвайки команда *fprintf* може да се изведе комбинация от текстове, напр.

fprintf('Размерността на матрицата е %i реда и %i стълба. \nДетерминантата на матрица P е %g.\n', n, n, detP),

при което в командния прозорец ще се изведе резултатът, показан на фиг. 5.5.

✓ команда *menu* – създава меню по подобие на интерфейса на Windows. В менюто може да има до 32 позиции.

k = menu('Заглавие', '1. Въвеждане', '2. Изчисляване', '3. Отпечатване', '4. Край')



фиг. 5.6

✓ *pause* – прекратява временно изпълнението на програмата. Ако командата е въведена без аргумент, паузата продължава до натискане на произволен клавиш от клавиатурата. При въвеждане на командата с аргумент

pause(n),

паузата продължава *n* секунди, след което продължава изпълнението на програмата.

Оператори за сравнение

Операторите за сравнение са:

>	по-голям от	< =	по-малък или равен на
> =	по-голям или равен на	= =	равни на
<	по-малък от	~ =	не равни на

Операторите за сравнение могат да действат върху скалари, вектори или матрици. Всички сравнения се извършват елемент по елемент и се връщат стойности: *1* (логично вярно) или *0* (логично невярно).

Логически оператори

Логическите изразявания могат да бъдат комбинирани с три логически оператора

&	AND
|	OR
~	NOT

При използването им трябва да се знае, че ~ *(NOT)* има по-високо предимство, отколкото *& (AND)* и | *(OR).* Правилата за използването им са същите, както при операторите за сравнение. Ако се използват две матрици, размерностите им трябва да бъдат еднакви.

Логически функции в Matlab

- ✓ *all (x)*- истина, ако всички елементи са различни от 0;

all(логически израз(x)) - истина, ако всички елементи на вектор x отговарят на условието (логически израз).

- ✓ *any(x)* – истина, ако има поне един елемент различен от 0;

any(логически израз(x)) - истина, ако поне един от елементите на вектор x отговаря на условието (логически израз).

✓ *find* – намира ненулевите елементи във вектор (или тези, отговарящи на зададено условие) и връща вектор с индексите на тези елементи.

find(y), find(y<10)

Управляващи оператори

Управляващите оператори дават възможност за организиране на циклични пресмятания, изпълняване на различни разклонения на програмата, както и прекратяване на програмата при определени условия.

В *Matlab* са вградени следните управляващи оператори:

- *if* – оператор за условен преход;
- *switch* – превключващ оператор;

- *for* – аритметичен оператор за цикъл;
- *while* – условен оператор за цикъл;

✓ *Оператор за условен преход if*

Възможни са три варианта за използването му:

Вариант 1

```
if логически израз
        действие
end
```

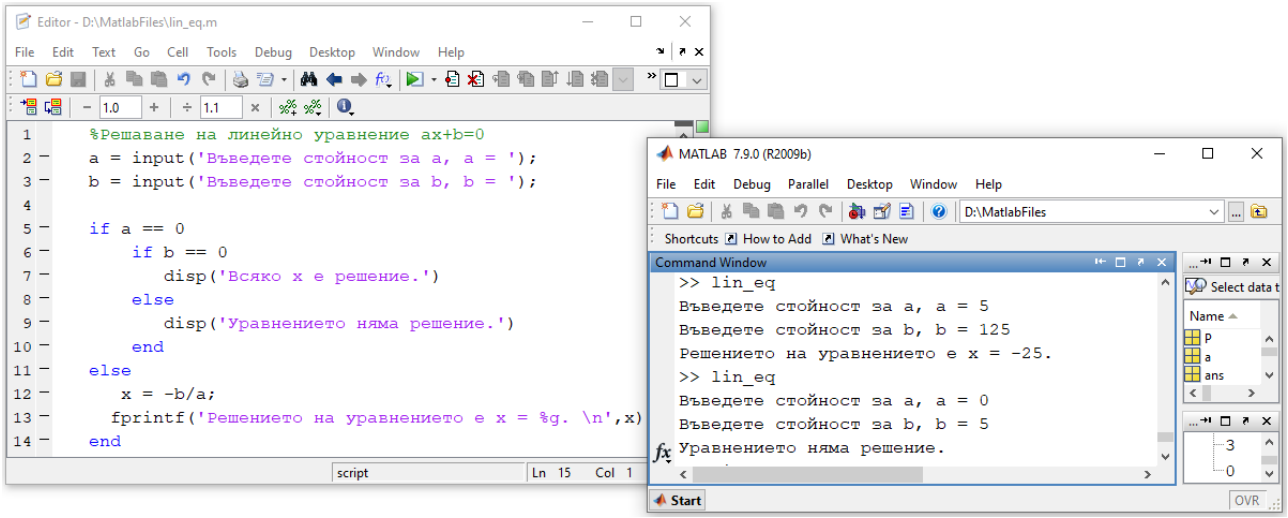
Вариант 2

```
if логически израз
        действие1
else
        действие2;
end
```

Вариант 3

```
if логически израз1
        действие1;
elseif логически израз2
        действие2;
elseif логически израз3
        действие3;
……
else
        действие;
end
```

Задача: Да се създаде m-файл, в който с помощта на оператор за условен преход *if* да се решава линейното уравнение $ax+b=0$. Стойностите за a и b да се въвеждат от клавиатурата в диалогов режим. Да се изведе в командния прозорец решението на уравнението с пояснителен текст.



фиг. 5.7

✓ *Аритметичен оператор за цикъл for*

Използва се за циклично изпълнение на определена част от програма. Броят на циклите е фиксиран. Общият вид на цикъл for е следния:

for index=нач.ст-ст:стъпка:крайна ст-ст

действие;

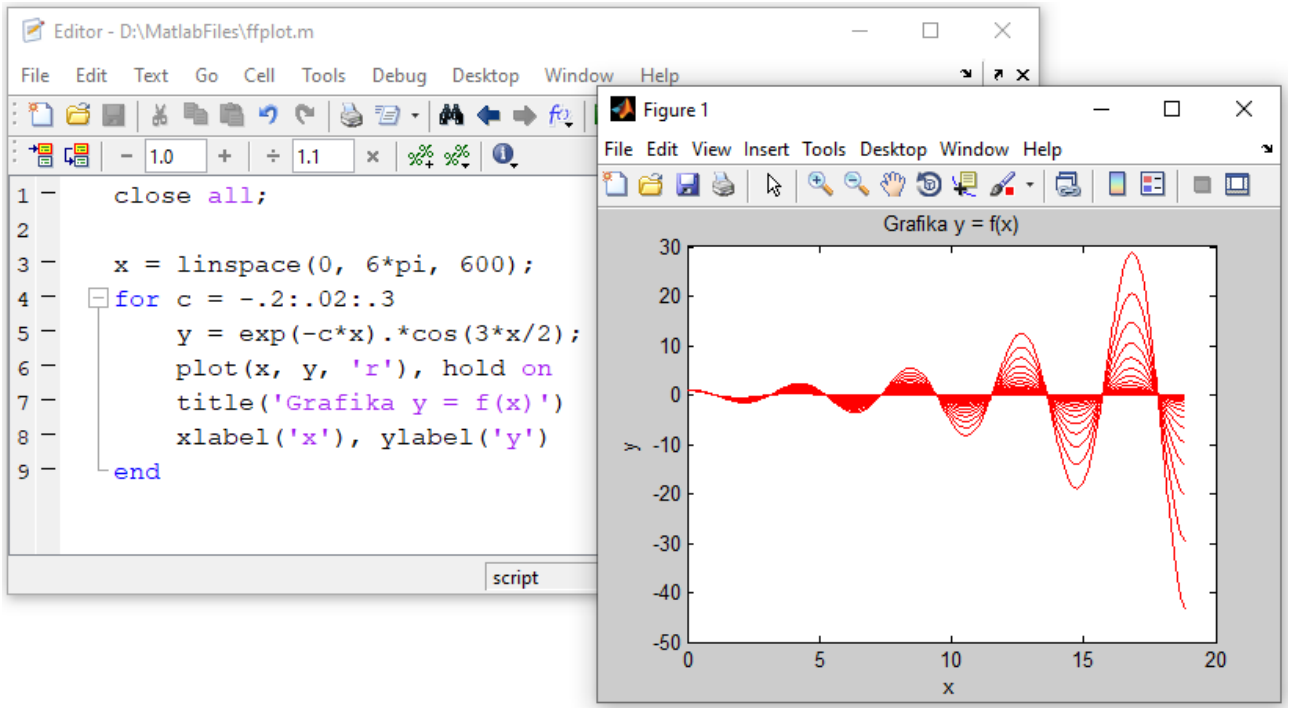
end

index – управляващата променлива на цикъла. Може да приема както цели, така и дробни стойности, а изменението ѝ в зависимост от стъпката, може да е във възходящ или низходящ ред.

Задача: Да се създаде m-файл, чрез който с помощта на *for*-цикъл да се изчертаят графиките на семейство криви, от вида:

$$y = e^{-bt}\cos t$$

за различни стойности на параметър *b* в интервала от -0.2 до 0.3, със стъпка на изменение 0.02, а за t да бъдат използвани 600 стойности в интервала от 0 до 6π.



фиг. 5.8

✓ *Условен оператор за цикъл while*

Използва се за организиране на цикли, които се изпълняват докато е изпълнено дадено условие. За разлика от for, тук броят на циклите не е фиксиран.

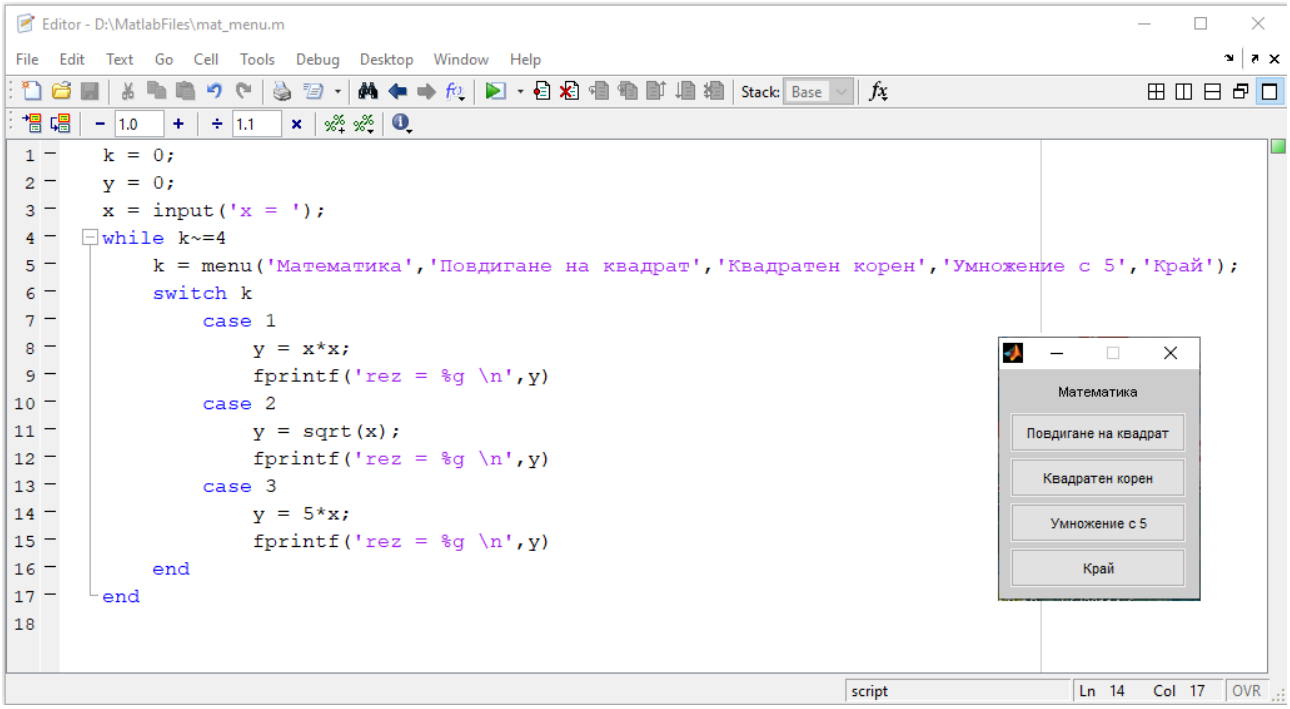
```
while логически израз
  действие;
end
```

✓ *Превключващ оператор switch*

Прилага се когато в зависимост от стойността на даден израз, трябва да се изпълняват различни клонове на програмата. В общия случай конструкцията е следната:

```
switch израз
  case стойност1
        действие1;
  case {стойност2, стойност3}
        действие2;
  otherwise
        действие;
end
```

🤔 ***Задача***: Да се създаде меню, което да има четири варианта за избор – повдигане на дадено число на квадрат, квадратен корен на даденото число, умножение на числото с 5 и край. Числото, с което да се извършват действията, да се въвежда в диалогов режим при стартиране на файла.



фиг. 5.9

Оператори break, continue

Операторите *break* и *continue* най-често се използват заедно с оператор *if* в тялото на *for* или *while* цикъл.

Оператор *break* прекратява изпълнението на цикъла и предава управлението на операторите, разположени след оператор *end*.

Оператор *continue* води до прескачане на операторите, разположени между него и оператор *end*.

Въпроси и задачи за изпълнение:

1. Кое отличава script файловете от файл-функциите в *Matlab*?

2. С коя команда в *Matlab* може да се въвежда информация в диалогов режим?

3. Да се създаде m-файл, в който да се генерират две матрици *A* и *B* от случайни числа с размерност n реда и m стълба (n и m – цели числа, въведени в диалогов режим след стартиране на файла).

➢ Да се намери матрица C, която да е сума от елементите на матрици *A* и *B*;

- Да се намери матрица A_1, която е произведение на матрица A с нейната транспонирана матрица;
- Да се пресметне детерминантата на матрица A_1;
- Да се намерят най-малките елементи от всеки стълб на двете матрици и да се запишат във вектори съответно *minA* и *minB*;
- В една фигура, на две отделни координатни системи, разположени една под друга да се изчертаят графики на *minA* (с червени плюсчета) и *minB* (със зелени звездички) и да се сложи заглавен текст над двете координатни системи.

4. Да се състави m-файл за изчисляване на функция y:

$$y = \begin{cases} \sqrt{a^2 + b^2}, & ако \quad a > b, \\ a^2, & ако \quad a = b, \\ -a * b, & ако \quad a < b. \end{cases},$$

за стойности на *a* и *b*, въведени в диалогов режим след стартиране на файла. Резултатите да се извеждат с пояснителен текст.

5. Да се създаде m-файл, в който с помощта на оператор за условен преход *if*, да се решава квадратното уравнение $ax^2 + bx + c = 0$. Стойностите за *a*, *b* и *c* да се въвеждат от клавиатурата в диалогов режим. Да се изведе в командния прозорец решението на уравнението с пояснителен текст.

Лабораторно упражнение 6
Въведение и работа със Simulink:
библиотеки, блокове и изграждане на модели

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите на *Simulink*, основните библиотеки, които включва, създаването на модели, настройка на параметрите на симулацията.

Въведение

Simulink е графична среда в програмния продукт *Matlab* и е неделима част от него, т.е. не може да бъде стартирана самостоятелно.

Възможностите, които *Simulink* предлага, могат да бъдат използвани за моделиране, симулация и анализ на различни динамични системи. Тя позволява изграждане на модели чрез свързване на функционални блокове, без необходимост от писане на програмен код. *Simulink* е особено подходящ за представяне на непрекъснати и дискретни системи, системи за управление и обработка на сигнали. Средата предлага разнообразни средства за визуализация на резултатите и изследване на поведението на системите във времето. При решаване на задачите е възможно както самостоятелно симулиране и анализ на дадената система изцяло в средата на *Simulink*, така и експортиране на данни в средата на *Matlab* за допълнителна обработка с помощта на функции на *Matlab*.

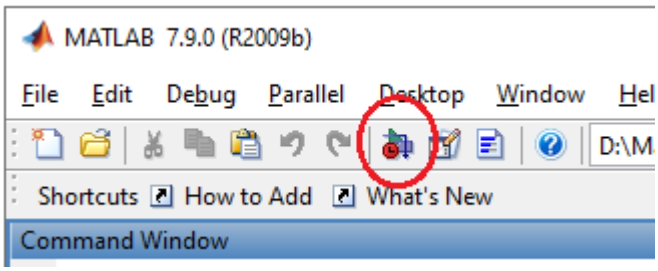
Стартиране на Simulink

След стартиране на програмата *Matlab*, графичната среда *Simulink* може да бъде стартирана по три начина:

- от командния ред на *Matlab* с ключовата дума *simulink*:

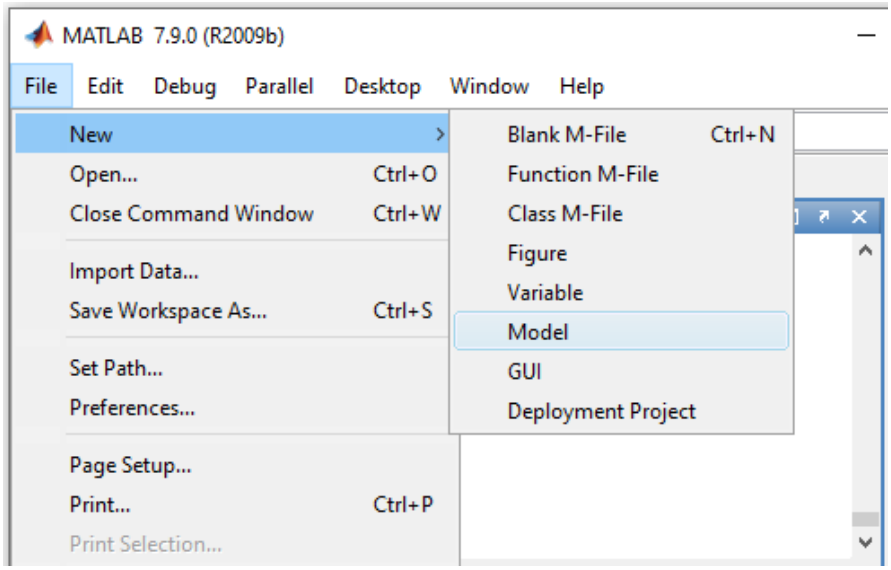
```
>> simulink
```

- от лентата с бутоните от прозореца на *Matlab*:



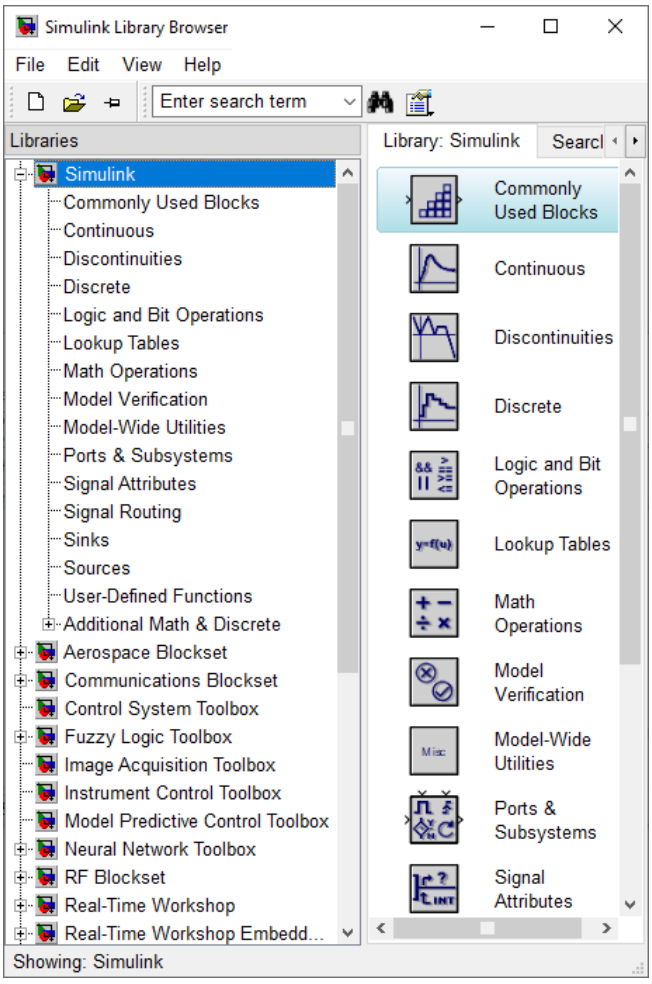
фиг. 6.1

- от менюто *File* на прозореца на *Matlab* - избира се *File* → *New* → *Model* .



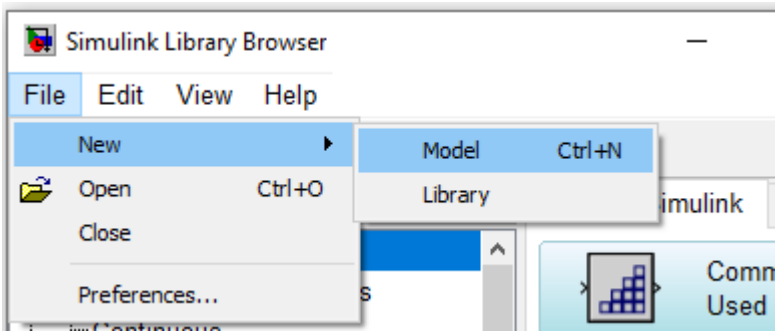
фиг. 6.2

При стартиране по първия или по втория начин се зарежда основния прозорец с библиотеките на *Simulink* (фиг.6.3).



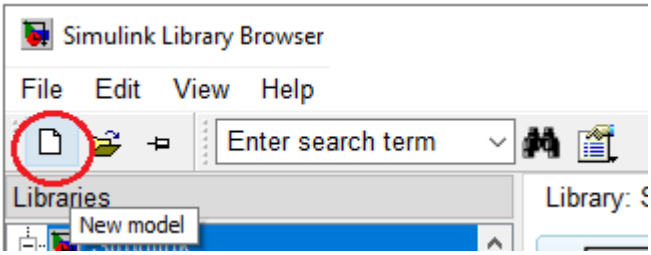
фиг. 6.3

Блок-схемата на дадената система се изгражда в друг прозорец, който се отваря от меню *File/New/Model*



фиг. 6.4

или с натискане на бутона *New Model* .



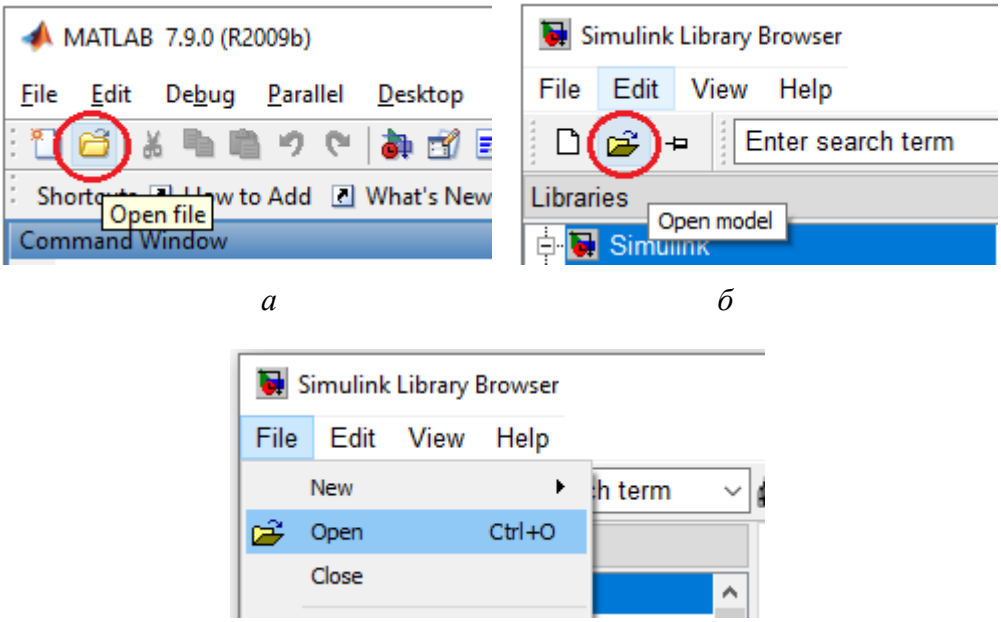
фиг. 6.5

В резултат се отваря празен прозорец, в който последователно се копират необходимите блокове от съответните библиотеки, свързват се и се настройват параметрите на симулацията.

При стартиране на *Simulink* по третия начин (от меню *File*) се отваря нов файл, в който се изгражда блок схемата, а прозорецът с библиотеките трябва да се отвори допълнително.

Отваряне на съществуващ файл в Simulink

За отваряне на вече създаден *Simulink* файл има три начина (фиг.6) – с бутон *Open file* от прозореца на *Matlab* (а), с бутон *Open model* от прозореца на *Simulink* (б) или от меню *File/ Open* от прозореца на *Simulink* (в).



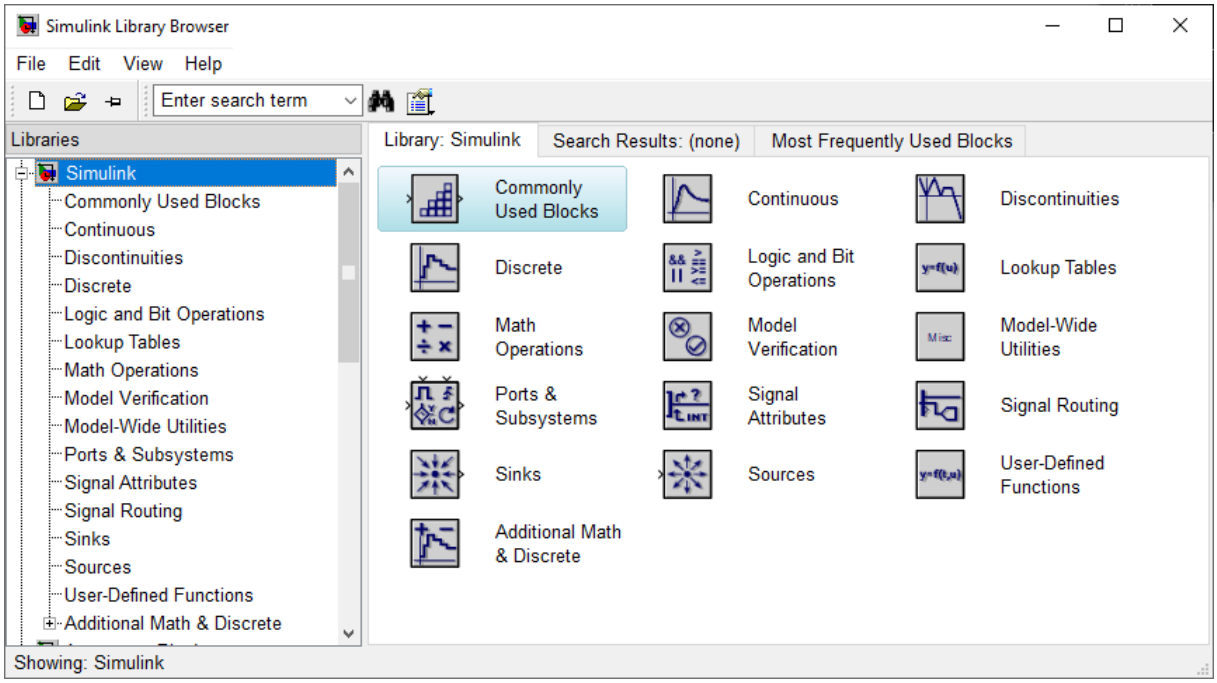
а *б*

в

фиг. 6.6

Създаването на модели в Simulink се основава на графично представяне на динамичните системи чрез функционални блокове, групирани в отделни библиотеки, и връзки между тях. Всеки блок описва определена математическа операция или елемент на системата, а линиите показват потока на сигналите. Този подход позволява интуитивно изграждане и модифициране на модели както на базата на математическо описание на дадена система, така и без директно формулиране на уравнения. Чрез симулация потребителят може да изследва поведението на системата при различни входни условия и параметри.

Групирането на функционалните блокове в библиотеките на *Simulink* изглежда по следния начин:



фиг. 6.7

- *Commonly Used Blocks* – често използвани блокове;
- *Continuous* – тази библиотека включва блокове за моделиране на непрекъснати динамични системи, като интегратори, диференциатори и непрекъснати предавателни функции. Тя се използва основно при описание на системи с непрекъснато време.;
- *Discontinuities* – библиотеката съдържа блокове, описващи нелинейни и прекъснати зависимости, като релета, насищане, зони на нечувствителност и превключватели. Тези блокове се използват при моделиране на реални системи с ограничения, превключвания или нелинейно поведение;
- *Discrete* - съдържа блокове за дискретни системи и алгоритми, работещи с дискретно време, като дискретни интегратори, закъснения и дискретни предавателни функции;
- *Logic and Bit Operations* – тази библиотека предоставя блокове за логически операции и побитова обработка на сигнали. Използва се при моделиране на логически схеми и цифрови системи;
- *Lookup Tables* - библиотеката позволява реализиране на нелинейни зависимости чрез таблично зададени стойности. Тези блокове са особено полезни при моделиране на емпирични или експериментално получени характеристики.

- *Math Operations* – в библиотеката се намират блокове за различни аритметични и логически операции, включително събиране, умножение, деление, степенуване и елементарни математически функции. Те позволяват реализиране на математически зависимости между сигналите;
- *Model Verification* – предоставя блокове за проверка и валидиране на Simulink модели. Блоковете от тази библиотека позволяват откриване на грешки, проверка на диапазони, логически условия и коректност на сигналите по време на симулация;
- *Model-Wide Utilities* - съдържа помощни блокове, които влияят върху поведението на целия модел или подпомагат неговата организация. Те се използват за управление на параметри, конфигуриране на симулацията и улесняване на работата с по-големи и сложни модели;
- *Ports&Subsystems* – тази библиотека съдържа блокове за изграждане на подсистеми и дефиниране на входно-изходни портове. Те подпомагат структурираното изграждане на сложни модели и повторното използване на части от тях;
- *Signal Attributes* – включва блокове за задаване и управление на параметри на сигналите, като тип на данните, размерност, единици и граници. Тези блокове са важни за осигуряване на съвместимост между различните части на модела и за предотвратяване на грешки при симулация;
- *Signal Routing* - библиотека с помощни за изграждането на моделите блокове. Тя съдържа блокове за разклоняване, обединяване и насочване на сигнали в модела. С тяхна помощ се организира потокът на данните между различните части на системата;
- *Sinks* – блоковете от тази библиотека служат за визуализация и запис на резултатите от симулацията. Тук са включени блокове за графично и цифрово визуализиране на данни, както и за запис на информация във файлове и към работното пространство на *Matlab*;
- *Sourses* – съдържа блокове за генериране на входни сигнали, като константи, стъпаловидни функции, синусоиди, импулси и шум. Те

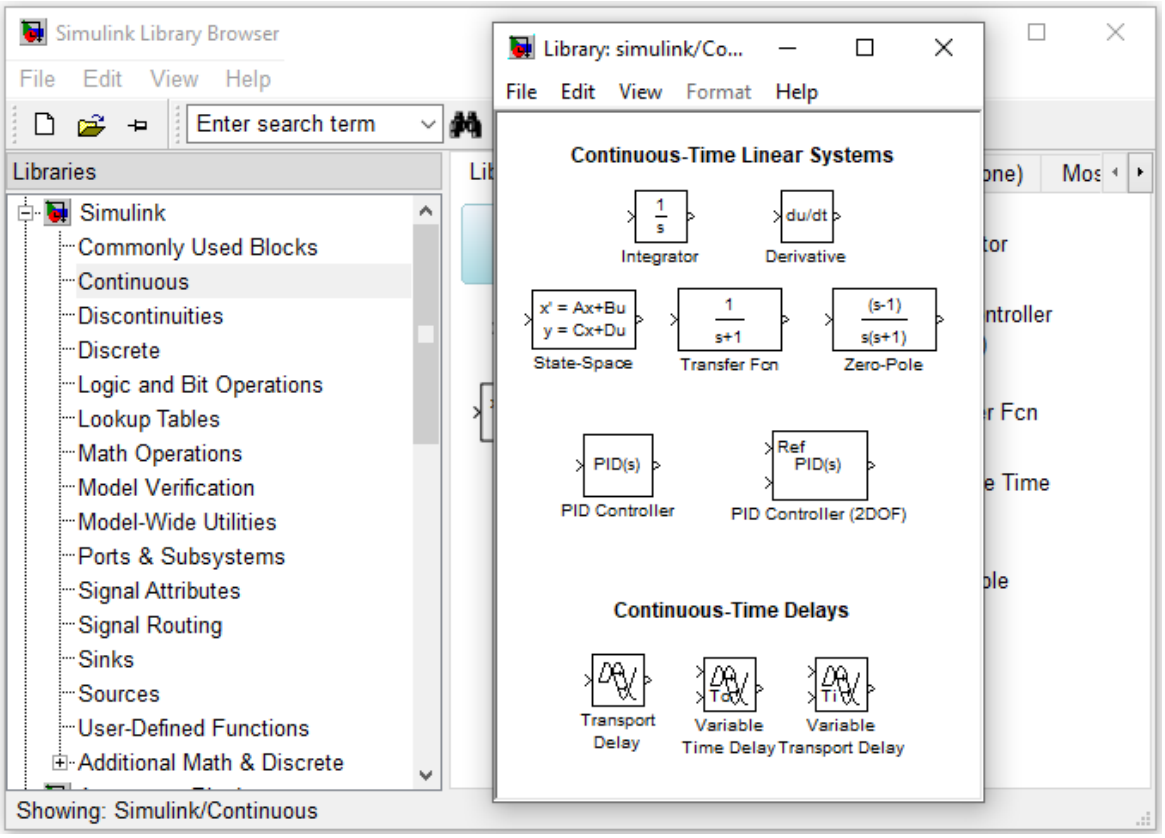
се използват за изследване на реакцията на системите при задаване на различни входни сигнали;

- *User-Defined Functions* – съдържа обобщени потребителски функции;
- *Additional Math & Discrete* – допълнителни математични и дискретни функции.

От посочените библиотеки ще бъдат разгледани най-често използваните блокове. Всяка от групите блокове може да бъде отворена както в същия прозорец (чрез еднократно щракане върху името й), така и в отделен (с десен бутон върху името на групата и избиране на *"Open 'xxx' library"*.

Група *Continuous* - фиг.6.8

В тази група са включени блокове, свързани с описание на непрекъснати системи.



фиг. 6.8

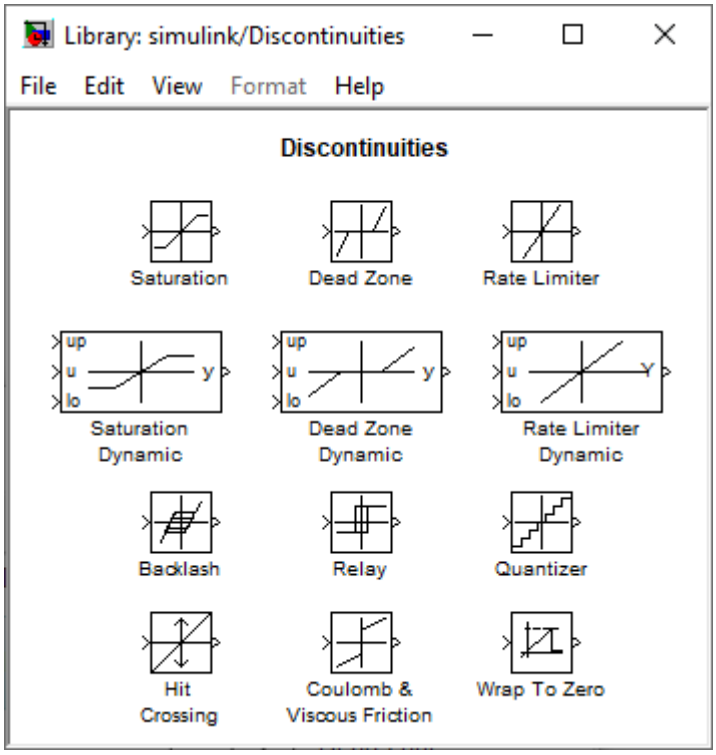
- *Integrator* - блок за реализиране на математическо интегриране на входния сигнал. Той се използва за моделиране на състояния на динамични системи и е основен градивен елемент при реализация на диференциални уравнения в Simulink;

- *Derivative* - блок за изчисляване на производната на входния сигнал. Поради чувствителността си към шум, той много често се използва в комбинация с филтри;
- *State-Space* - представя система с матрици A, B, C и D в пространството на състоянието. Този блок е особено подходящ за моделиране на многовходови и многоизходни системи;
- *Transfer Fcn* - представя система с нейната предавателна функция в операторната област. Той е удобен за бързо изграждане и анализ на линейни динамични системи без изрично формулиране на диференциални уравнения;
- *Zero-Pole* - представя линейна система чрез нейните нули, полюси и коефициент на усилване. Той е удобен при анализ на устойчивост и честотни характеристики;
- *Transport Delay*, *Variable Transport Delay* - блокове за времезакъснение.

Група *Discontinuities* - фиг.6.9

Тази група включва блокове, свързани с описание и моделиране на нелинейни, прекъснати и превключващи елементи, характерни за реални системи.

- *Saturation* - блок за усилвател с насищане, ограничава изходния сигнал между зададени минимална и максимална стойност. Използва се за моделиране на физически ограничения на системи;
- *Dead Zone* - блок за усилвател със зона на нечувствителност около нулата, при която изходът остава нулев;
- *Relay* – блок за реализация на реле с хистерезис;
- *Quantizer* – дискретизира входния сигнал с избрания от потребителя такт.

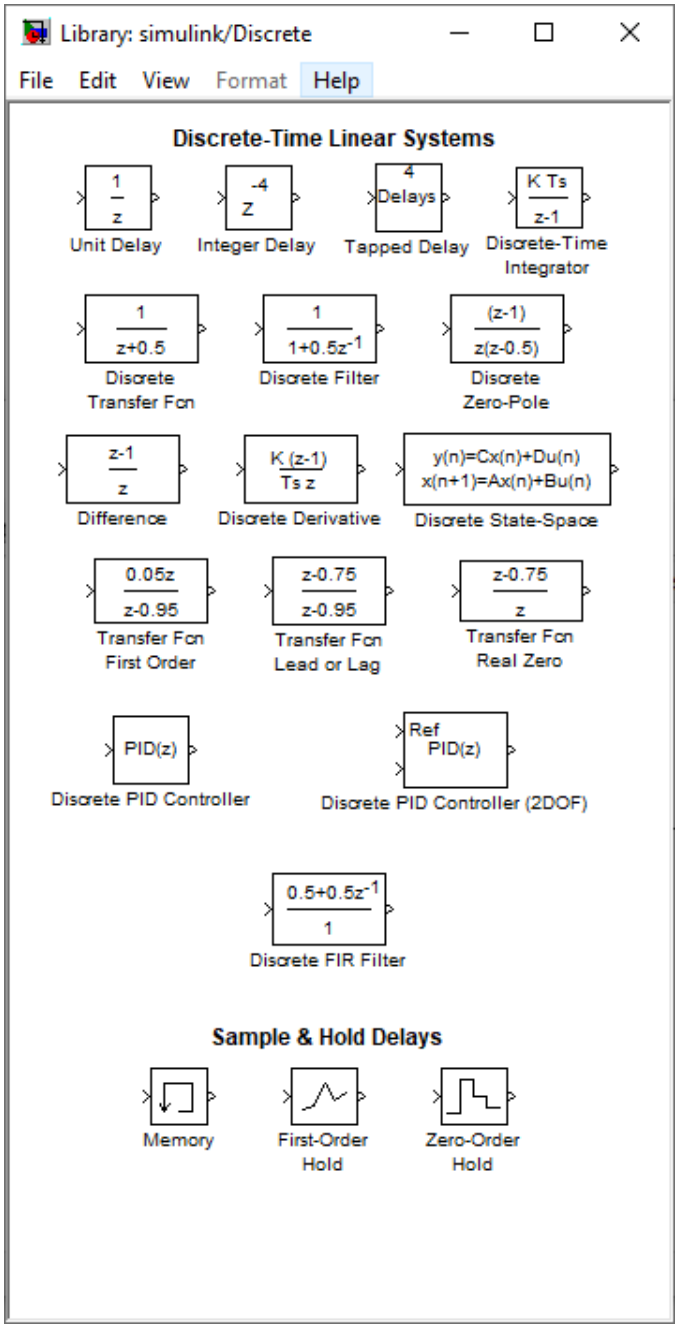


фиг. 6.9

Група *Discrete* – фиг.6.10

В тази група са включени блокове, свързани със симулиране на дискретни системи, като повечето са аналогични на тези от група *Continuous*.

- *Unit Delay* - реализира закъснение от една дискретна стъпка;
- *Discrete – Time Integrator* – реализира числено интегриране във времето и представлява дискретен аналог на непрекъснатия интегратор;
- *Discrete Transfer Fcn* – представя системата с нейната дискретна предавателна функция;
- *Discrete Filter* – дискретен филтър;
- *Discrete Zero Pole* – представя системата с нейната дискретна предавателна функция с нули и полюси;
- *Memory* – блок за забавяне с един такт като запазва последната изчислена стойност на сигнала.

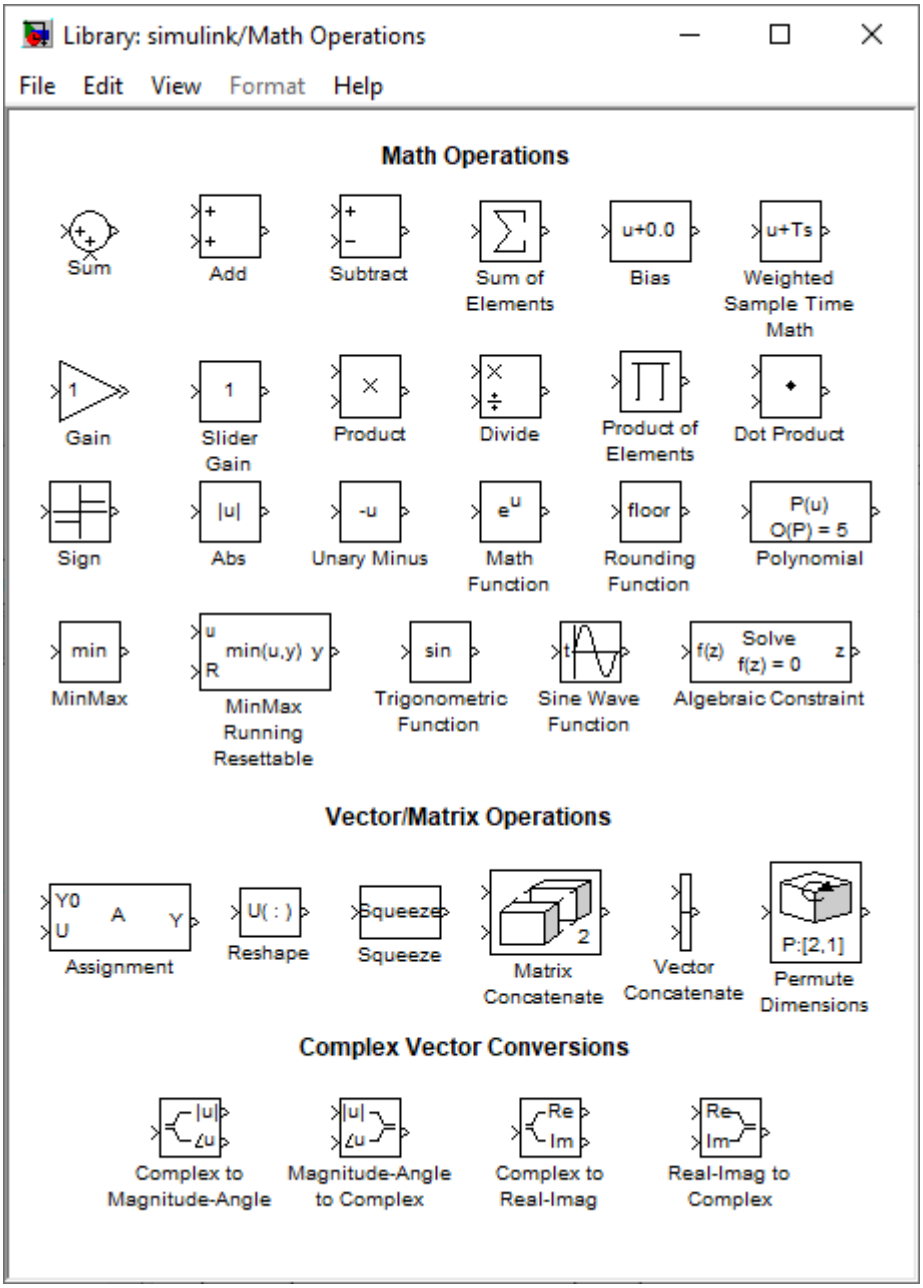


фиг. 6.10

Група *Math Operations* – фиг.6.11

Тази група съдържа блокове, които се използват за извършване на основни алгебрични и математически операции върху сигналите.

- *Sum* - извършва събиране и изваждане на входните сигнали. Той е основен елемент при изграждане на обратни връзки;
- *Gain* - умножава входния сигнал със зададен коефициент. Често се използва за настройка на усилвания и параметри;



фиг. 6.11

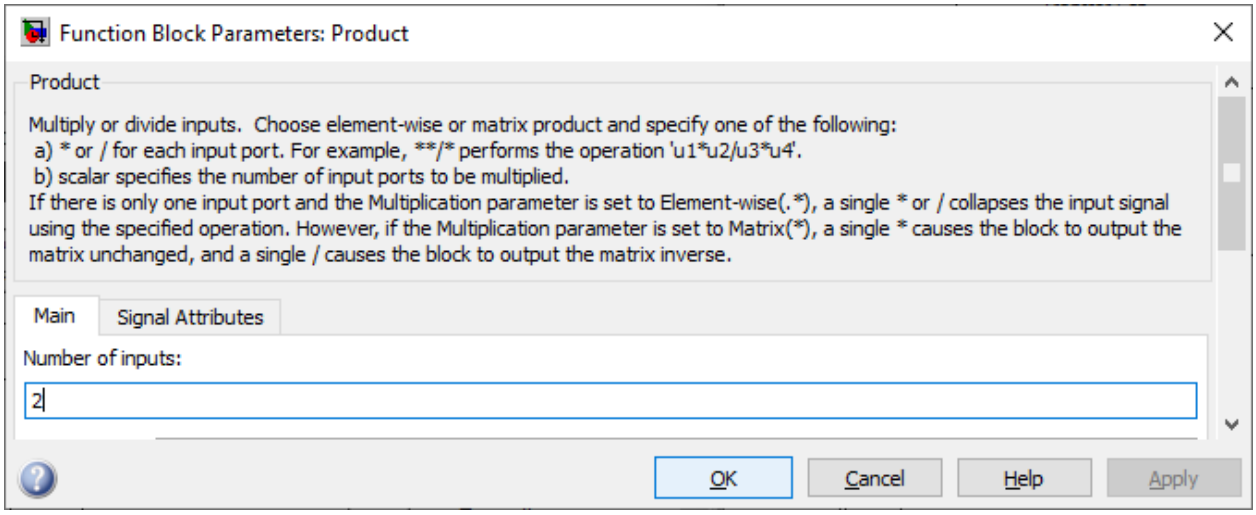
- *Product* – на изхода на блока се получава произведението от входните сигнали;
- *Divide* - деление на входните сигнали;
- *Sign* - сигнум функция (идеално реле) - изходът на блока е 1 или -1 в зависимост от знака на входния сигнал;
- *Abs* - изходът на блока е абсолютната стойност на входния сигнал;
- *Math Function* - прилага избрана от потребителя основна математическа функция (логаритъм, експонента, степенуване, корен и др.);

- *MinMax* – в зависимост от избраната функция на изхода на блока се извежда най-малкия или най-големия елемент на входния сигнал;
- *Trigonometric Function* - прилага избрана от потребителя тригонометрична функция;

Някои от горните блокове имат променлив брой входове, избираем от потребителя.

Такива са например блоковете *Sum* и *Product*.

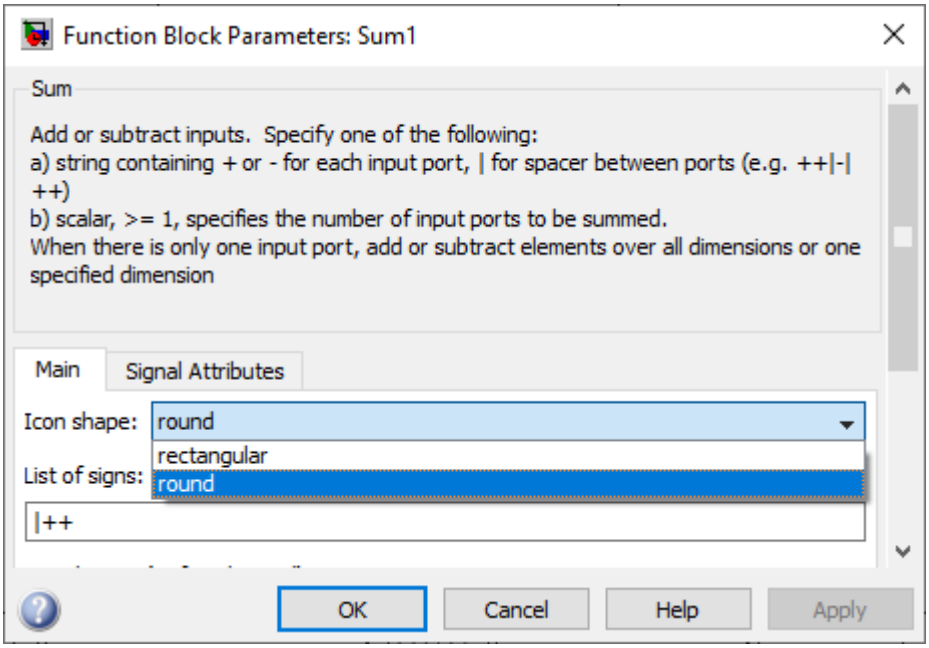
Броят на входовете за блок *Product* се задава в полето *Number of inputs* от прозореца с параметрите (фиг.6.12). По подразбиране броят им е 2.



фиг. 6.12

За блок *Sum* броят на входовете се задава в полето *List of signs*, но не чрез цифра, а чрез знаците „+” или „-”. Броят на записаните знаци определя броят на входовете за блок *Sum*, а в зависимост от вида на поставения знак, за сигналът на съответния вход ще бъде извършено действие сумиране (знак „+”) или изваждане (знак „-”).

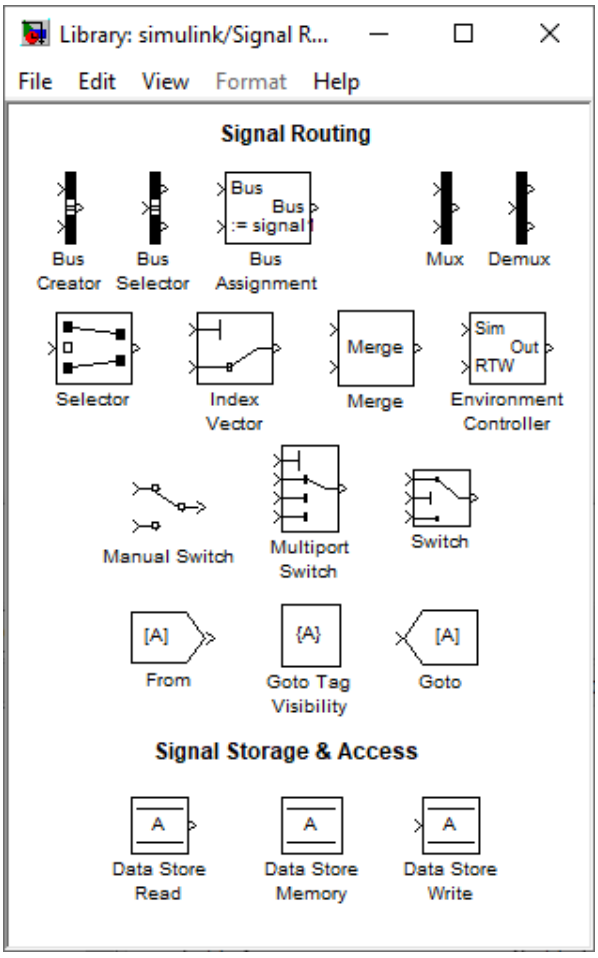
За блок *Sum* има възможност да се променя и формата му, която по подразбиране е кръгла. Промяната става от полето *Icon shape*, а възможностите са – кръгла или правоъгълна форма.



фиг. 6.13

Група *Signal Routing* – фиг.6.14

Тази библиотека съдържа блокове, които играят помощна роля при изграждането на различни модели. Най-често използваните блокове са:



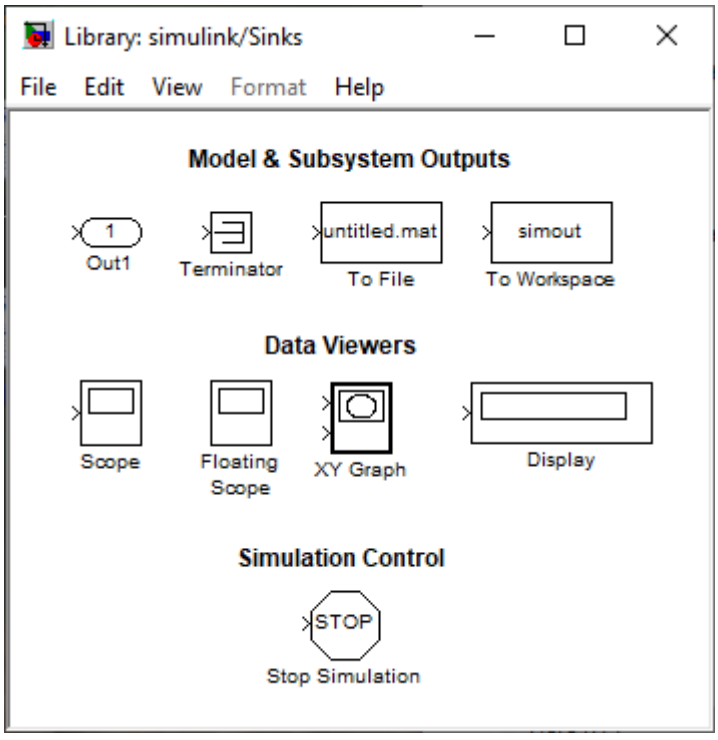
фиг. 6.14

- *Mux* – блок, който комбинира подадените на входа скаларни или векторни сигнали в една обща структура;
- *Demux* – блок, който разделя подадения на входа векторен сигнал на няколко изходни сигнала;
- *Switch* – блок, който превключва между два входни сигнала в зависимост от стойността на трети сигнал.

Група *Sinks* – фиг.6.15

Блоковете, включени в тази група, са свързани с визуализация или запис на информация от другите блокове в модела. Те имат само входове, без изходи. Най-често използваните блокове са:

- *To File* – записва данните, подадени на входа, във файл (*.mat);
- *To Workspace* – експортира данните, подадени на входа, в средата на *Matlab*;
- *Scope* – визуализира графически подадения на входа сигнал;



фиг. 6.15

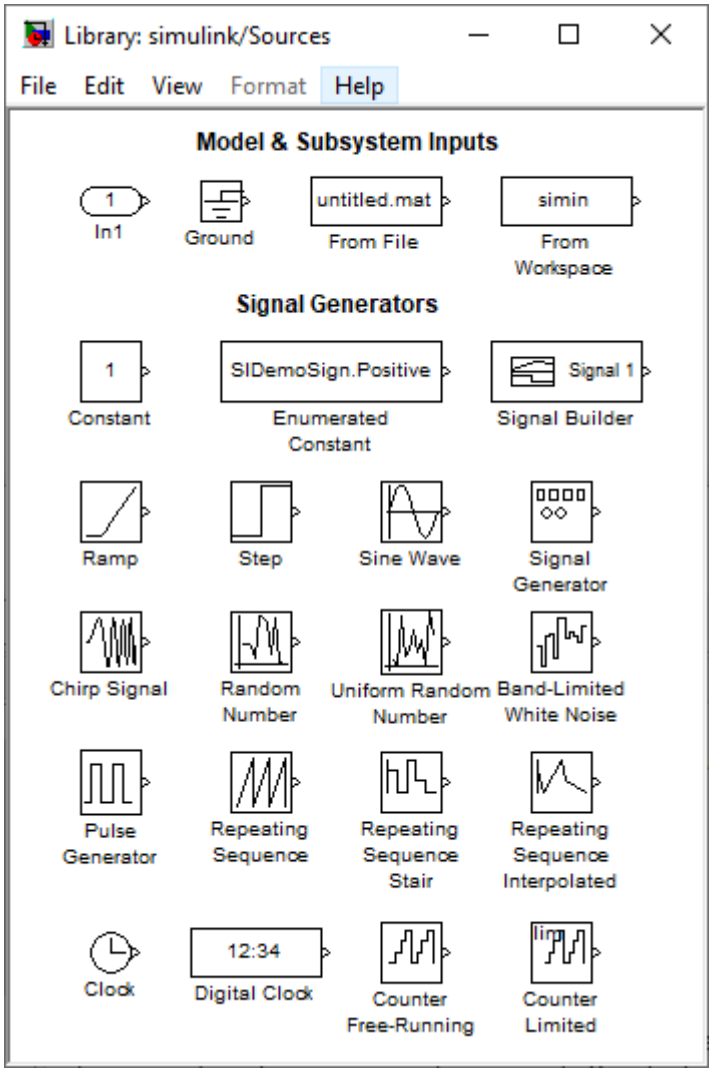
- *XY Graph* – блокът има два входа и показва графика, която представлява зависимостта на сигнала от втория вход във функция от сигнала, подаден на първия вход;

- *Display* – показва числената стойност на входния сигнал със зададена от потребителя точност;
- *Stop Simulation* – блок, използван за спиране на симулацията при изпълнение на дадено условие за входния сигнал.

Група *Sources* – фиг.6.16

Блоковете, включени в тази група, са източници на различни сигнали, т.е. те имат само изход. Чрез блоковете от тази библиотека се задава входния сигнал на системата. Най-често използваните блокове са:

- *From File* – блок за четене на данни от даден файл;
- *From Workspace* – блок за четене на данни от работното пространство на *Matlab*;



фиг. 6.16

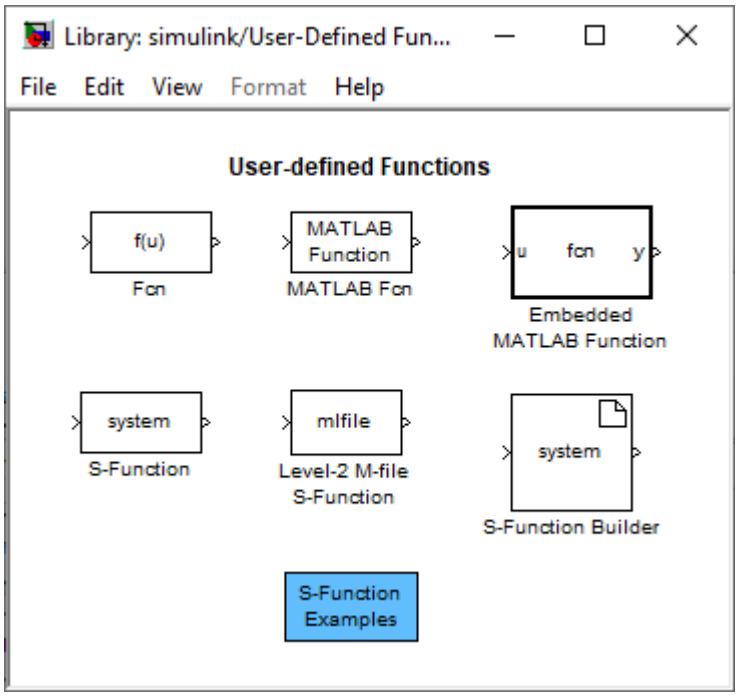
- *Constant* – блок за задаване на постоянен сигнал (константа);

- *Step* – блок, подаващ на системата стъпаловидна функция;
- *Sine Wave* – блок за генериране на синусоидален сигнал;
- *Random Number* – блок, генериращ случаен сигнал;
- *Signal Generator, Pulse Generator, Repeating Sequence* – блокове, позволяващи генериране на широк набор от непрекъснати и дискретни сигнали;
- *Clock* – този блок дава симулационното време като самостоятелен сигнал.

Група *User-defined Functions* - фиг.6.17

Тази група съдържа блокове, описващи различни обобщени функции. Основните блокове в нея са:

- *Fcn* – блок функция, при която зададен от потребителя израз се прилага върху входния сигнал на блока;
- *Matlab Fcn* – функция на *Matlab*, с този блок може да се извиква конкретна функция на *Matlab* и да се приложи върху входния сигнал.



фиг. 6.17

Изграждане на модел в Simulink

След отваряне на основния прозорец на *Simulink* по избран от потребителя начин, може да започне изграждането на конкретна блок-схема. За да се улесни работата при избиране на даден блок от библиотеките е добре основният прозорец и прозорецът с библиотеките да бъдат така позиционирани на работното поле на Windows, че да бъдат едновременно изцяло видими.

За да бъде използван даден блок, е необходимо той да бъде избран от дадената библиотека и чрез влаченето му с мишката да бъде прехвърлен в отворения файл на *Simulink*. След като се прехвърлят дадените блокове, е необходимо те да бъдат свързани помежду си.

Свързването на два блока може да стане по един от следните начини:

- С натискане на левия бутон на мишката върху изхода на единия блок и влачене до входа на другия блок;
- Щракане с левия бутон на мишката върху блока, чийто изход ще свързваме, натискане на клавиш *Ctrl* и след това щракане върху блока, чийто вход трябва да бъде свързан.

🤔 ***Задача***: Да се изгради модел в *Simulink*, визуализиращ графиката на синусоидален входен сигнал.

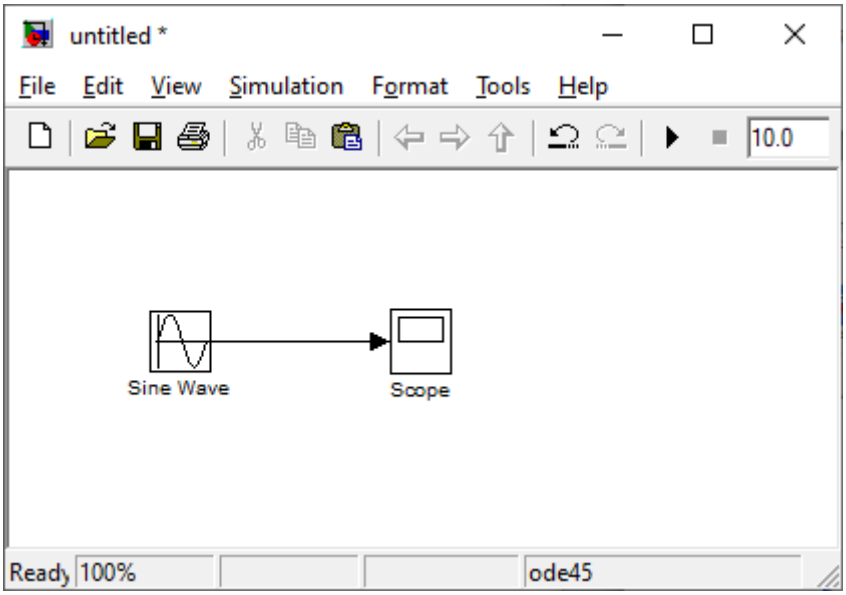
Стъпките за изпълнение на задачата са следните:

1. Отваря се нов *Simulink* файл;

2. От прозорецът с библиотеките от библиотека *Sources* се избира блок *Sine Wave* и влачейки се прехвърля в отворения файл.

3. По същия начин от библиотека *Sinks* се прехвърля блок *Scope*.

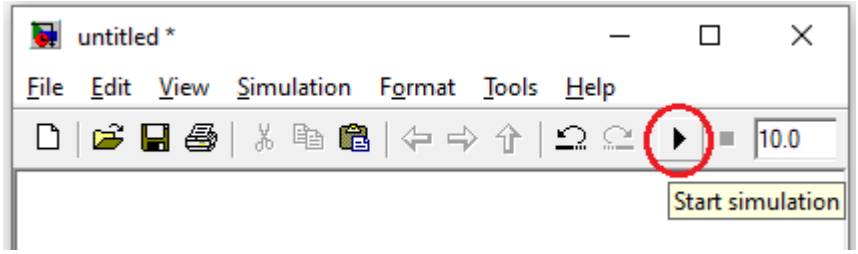
4. По един от посочените по-горе начини се свързват двата блока.



фиг. 6.18

След като схемата е създадена, може да бъде стартирана симулацията. Това става по един от следните начини:

- От меню *Simulation/Start*
- От лентата с бутони (фиг.6.19)



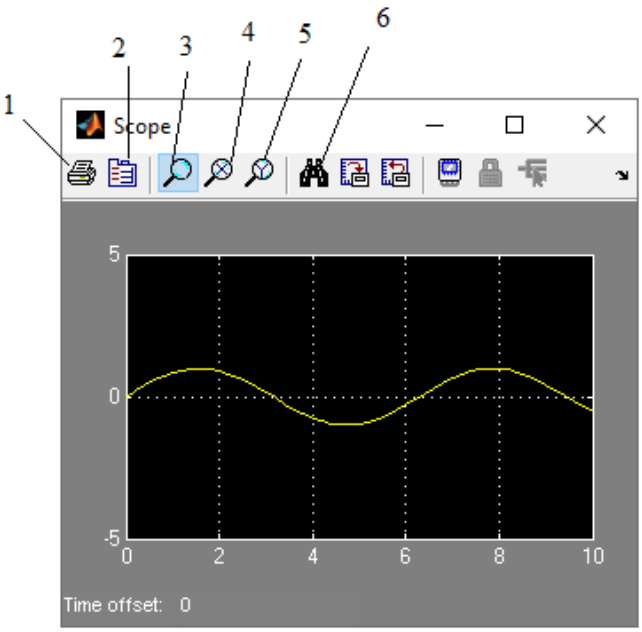
фиг. 6.19

- С клавишната комбинация Ctrl+T.

След като симулацията е стартирана, получената графика може да бъде видяна чрез двукратно кликване върху блок Scope.

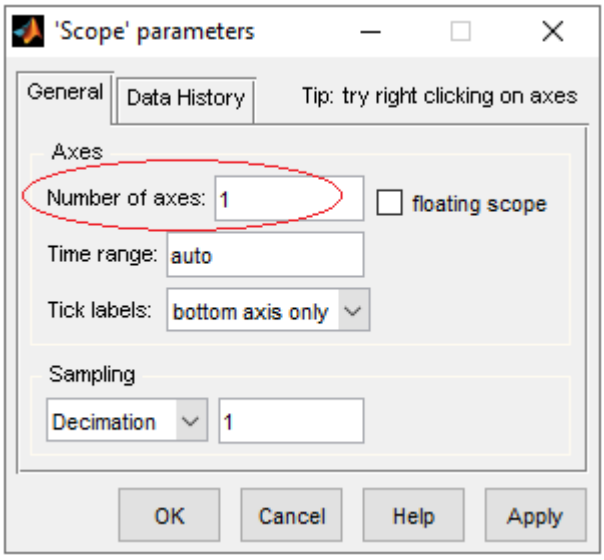
От лентата с бутоните на отворения графичен прозорец могат да бъдат извършвани действия като:

- разпечатване на графиката (1),



фиг. 6.20

- Настройка на различни параметри (2), като например промяна на броя координатни оси в дадения графичен прозорец (фиг.6.21),



фиг. 6.21

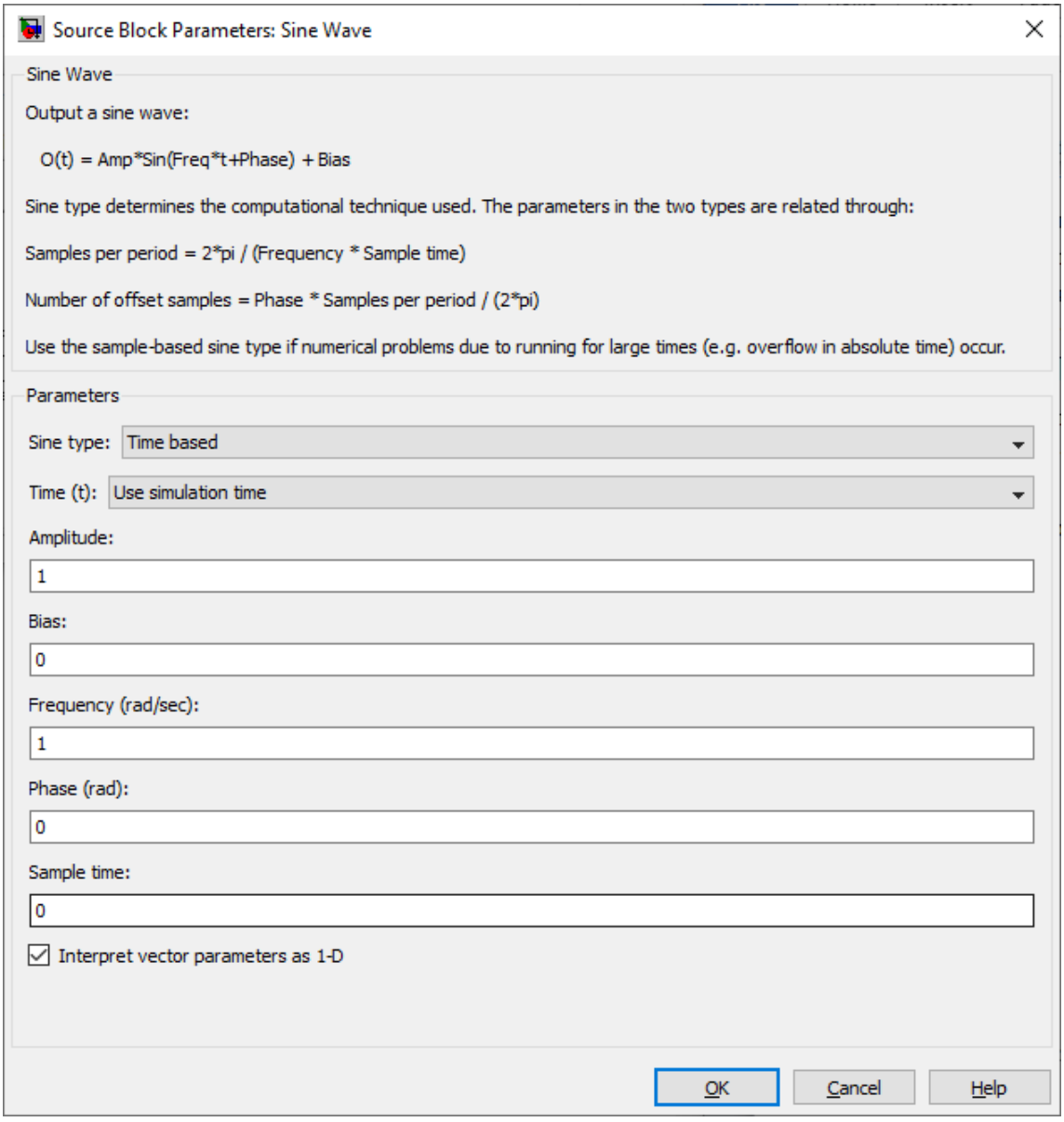
- промяна на мащаба по дадена ос (3, 4, 5),
- автоматично мащабиране на осите, като *Simulink* избира най-подходящия мащаб (6).

Задаване на параметри на блоковете

При по-голяма част от блоковете в *Simulink* има възможност за задаване на параметри от потребителя. Тези параметри се намират в прозореца с параметри, който се отваря чрез двукратно кликване върху дадения блок. За

всички параметри има предварително зададени стойности по подразбиране, които се използват от Simulink в случай, че потребителят не е задал други.

За изполвания в задачата блок *Sine Wave* параметрите на синусоидалния сигнал по подразбиране са: амплитуда (*Amplitude*) 1, честота (*Frequency*) 1. За да бъдат променени тези параметри се кликва два пъти върху блока (*Sine Wave*). Отваря се диалогов прозорец, съдържащ параметрите на входния сигнал (фиг.6.22):



фиг. 6.22

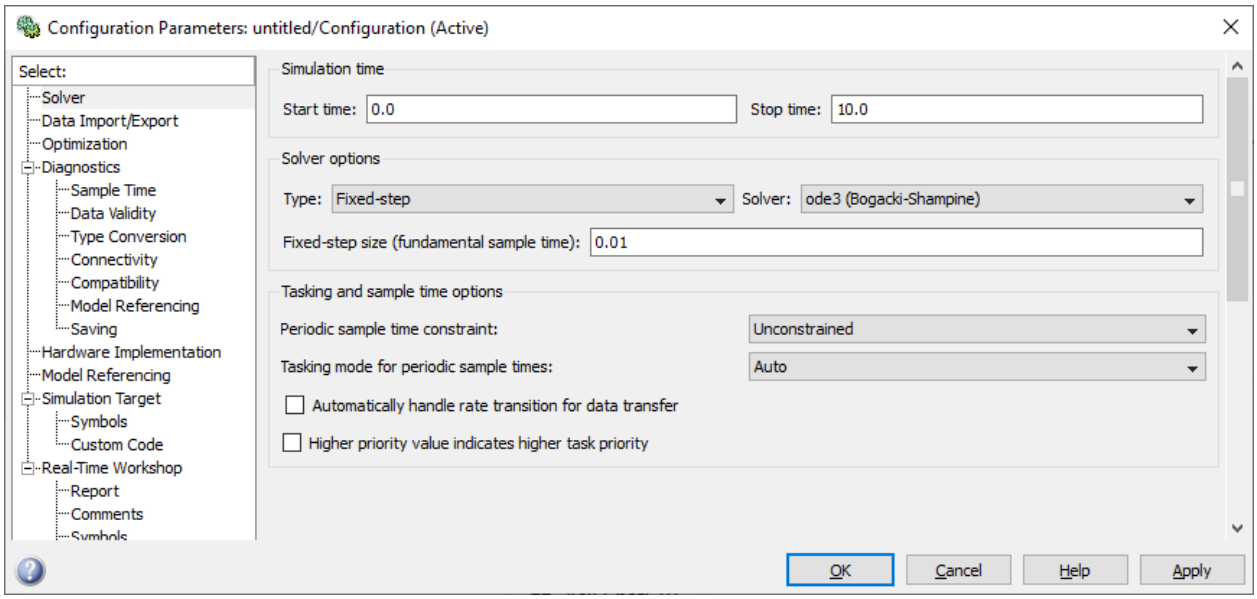
- *Amplitude* – амплитуда на входния сигнал;
- *Bias* - постоянна съставяща, която може да се добавя към сигнала, което води до вертикално отместване на синусоидата спрямо нулевата линия;

- *Frequency* – честота на входния сигнал, в rad/s;
- *Phase* – начална фаза на синусоидата (в радиани);
- *Sample time* - период на дискретизация; зададената по подразбиране стойност „0“ означава непрекъснат сигнал.

Задача: Да се наблюдава сигнала на графиката при задаване на различни стойности на параметрите на синусоидалния входен сигнал – амплитуда (напр. 5, 15), честота (10, 15 rad/s) и фаза (напр. pi/5, pi/2).

Настройка на параметрите на симулация

По подразбиране времето за симулация е 10 симулационни секунди. Продължителността може да бъде променена от меню *Simulation/Configuration Parameters* (или с клавишна комбинация *Ctrl+E*).



фиг. 6.23

Параметрите, които могат да бъдат настройвани, са:

- ✓ време за начало и край на симулацията (*Start* и *Stop Time*);
- ✓ видът на стъпката на симулацията (постоянна, *Fixed Step*, или променлива, *Variable Step*), както и стойността ѝ (*Step size*);
- ✓ методът на интегриране (*Solver*).

Запис на файл и отваряне на съществуващ файл

Записът на даден *Simulink* файл става от меню *File -> Save As*, а отварянето на съществуващ *Simulink* файл се извършва от меню *File -> Open*.

Задача: Да се наблюдава реакцията на система, зададена чрез предавателната си функция

$$W = \frac{1}{p^2 + 1.5p + 10},$$

при подаден единичен стъпаловиден сигнал на входа й.

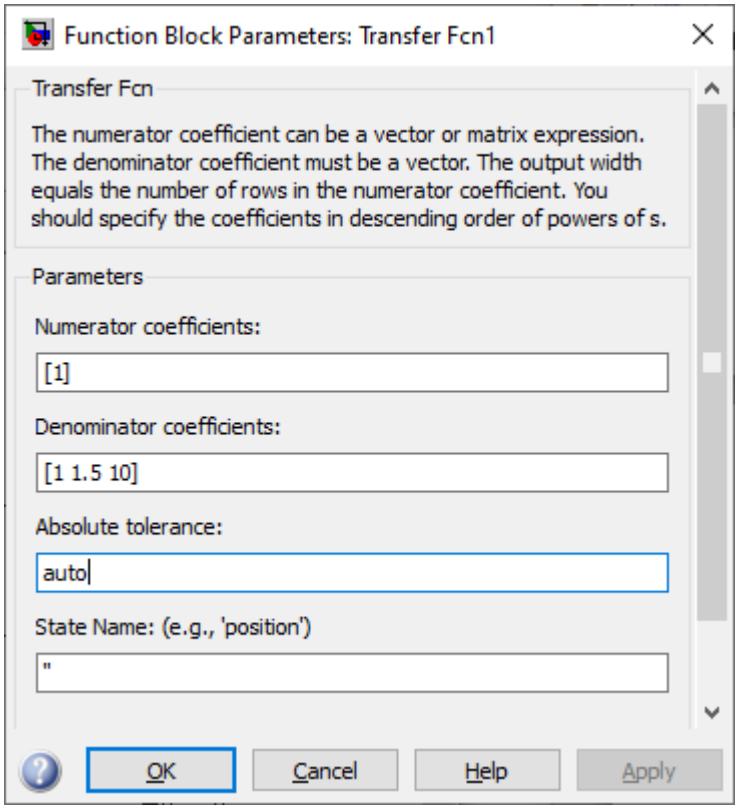
Получената блокова схема да се запише във файл с име *lab6_1.mdl*.

Стъпките за изпълнение на задачата са следните:

1. За генериране на стъпаловиден входен сигнал, от прозорецът с библиотеките, от библиотека *Sources* се избира блок *Step* и се добавя в отворения *Simulink* файл;

2. От библиотека *Continuous* се избира блок *Transfer Fcn*, който ще бъде използван за задаване на математичния модел на дадената система. На входа му се подава сигнал от изхода на блок *Step*;

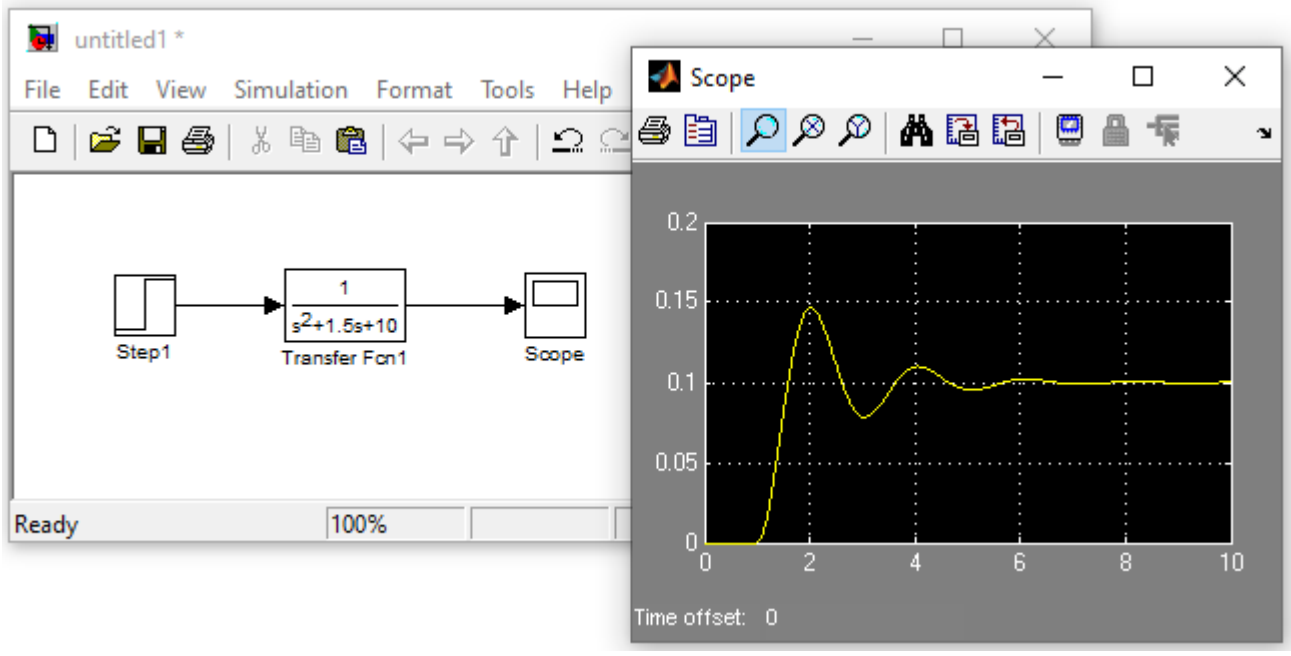
3. За да бъдат зададени параметрите от предавателната функция на системата, се отваря прозорецът с параметри на блок *Transfer Fcn*. В този прозорец, в полето *Numerator coefficients*, се задава вектор, съдържащ коефициентите от полинома в числителя на предавателната функция, а в полето *Denominator coefficients* - вектор, съдържащ коефициентите от полинома в знаменателя на предавателната функция.



фиг. 6.24

4. От библиотека *Sinks* се избира блок *Scope*, който ще бъде използван за визуализация и на входа му се подава сигнал от изхода на блок *Transfer Fcn*.

5. След като блоковата схема е готова се стартира симулацията по познатия начин и се наблюдава получения изходен сигнал.



фиг. 6.25

🤔 ***Задача***: Да се създаде блокова схема за визуализиране на графики на: синусоидален входен сигнал, на входния сигнал, умножен с константа 5, на входния сигнал, след преминаването му през блок за насищане (Saturation) и на първата производна на входния сигнал. Времето за симулация да е 30 секунди, а стъпката да бъде фиксирана със стойност 0.01.

Стъпките за изпълнение на задачата са следните:

1. От прозорецът с библиотеките, от библиотека *Math Operations* се избира блок *Gain*, добавя се към отвореният *Simulink* файл и в прозореца за параметрите се задава стойност 5;

2. От библиотека *Discontinuous* се избира блок *Saturation* и с помощта на мишката се провлачва в отворения файл;

3. От библиотека *Continuous* се избира блок *Derivative* и също се добавя към файла;

4. Избраните блокове се свързват, като сигналът от блок *Sine Wave* се подава на трите блока– *Gain, Saturation* и *Derivative*;

5. От менюто с параметри на блок *Scope* (фиг. 6.20), се променят броя на координатните оси на 4;

6. На всеки от създадените входове на блок *Scope* се подава по един сигнал (от изхода на блок *Sine Wave*, *Gain, Saturation* и *Derivative*);

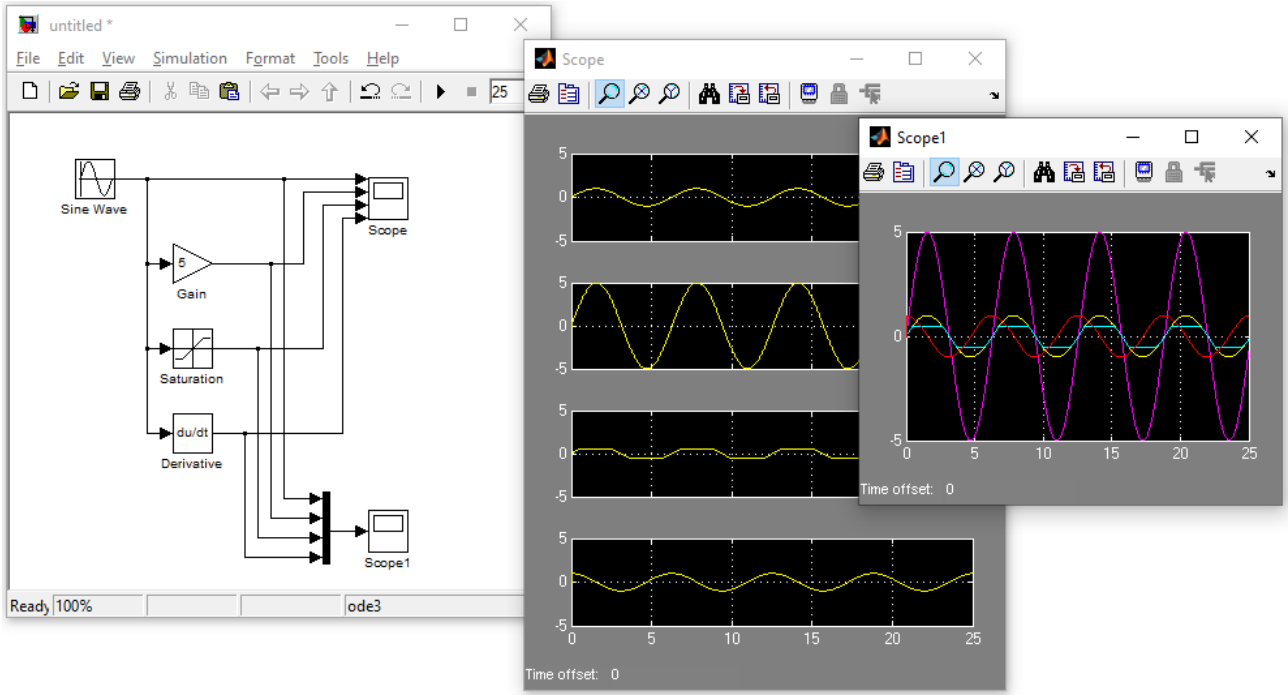
Четирите наблюдавани графики могат да бъдат визуализирани и на една координатна ситема. За целта е необходимо преди да бъдат подадени към блок *Scope*, сигналите да бъдат обединени в един масив чрез блок *Mux*.

7. От библиотека *Signal Routing* се избира блок *Mux*, добавя се към модела и в блока с параметрите се променя броят на входовете му (трябва да бъдат 4), като на тях се подават сигналите от изходите на блоковете *Sine Wave, Gain, Saturation* и *Derivative*;

8. Добавя се още един блок *Scope* (от библиотека *Sinks*), на входа на който се подава сигнала от изхода на блок *Mux*;

9. В прозореца за настройка на параметрите на симулацията се задава времето, видът и стойността на стъпката, след което се стартира симулацията и се отварят графичните прозорци, за да бъдат наблюдавани получените графики.

10. Създаденият модел да се запише във файл с име lab6_2.mdl.



фиг. 6.26

С този пример се вижда как визуализирането на няколко графики може да стане по два начина – всяка графика да бъде представена на отделна координатна система в един графичен прозорец или всички графики да бъдат изчертани на една координатна система.

Копиране на блокове

Копирането на блок или група от блокове става след маркирането им и натискане на клавишните комбинации *Ctrl + C*, *Ctrl + V*. Друг вариант е копиране от менюто *Edit/Copy*, *Edit/Paste* или с десен бутон на мишката и избор на съответните команди от падащото меню.

Изтриване на блокове

Изтриването на блок или група от блокове става след маркирането им и натискане на клавиш *Delete* или *Backspace*. Друг начин за изтриване е използването на командите *Cut* и *Clear* от меню *Edit*.

Променяне ориентацията на блок

По подразбиране сигналният поток през един блок тече от ляво на дясно, т.е. входовете на блока са от ляво, а изходите – отдясно. Ориентацията на блока може да се смени като се избере една от следните команди от менюто *Format*:

- ✓ *Flip Block* завърта блока на 180°;
- ✓ *Rotate Block* завърта блока на 90° по часовниковата стрелка;

Промяна на размера на блок

За да бъде променен размерът на даден блок е необходимо този блок да бъде маркиран, след което с натиснат бутон на мишката се изтегля едно от квадратчета, появили се в ъглите на избрания блок. Правоъгълникът с прекъснати линии показва новия размер на блока и след освобождаването на бутона на мишката блокът е с новият си размер.

Изчертаване на разклоняваща линия

Разклоняваща е линията, която започва от съществуваща вече линия и пренасяща нейния сигнал до входа на даден блок. И двете линии пренасят един и същ сигнал. Използването на разклоняваща линия позволява сигналът да бъде пренасян до повече от един вход.

За изчертаване на такава връзка се прави следното:

- ✓ Курсорът се позиционира върху линията, от която трябва да започне разклоняващата линия.
- ✓ Натиска се и се задържа клавиша *Ctrl*, след което се натиска и се задържа бутона на мишката.
- ✓ Курсорът се премества върху входа на желания блок и бутона на мишката и клавиша *Ctrl* се освобождават.

Въпроси и задачи за изпълнение:

1. По какви начини може да бъде стартиран *Simulink*?

2. Какви са параметрите на симулация в *Simulink* и как става тяхната настройка?

3. Да се състави блокова схема за визуализиране на графиките на различни входни сигнали, изчертавайки ги на отделни кооординатни системи в един графичен прозорец. Входните сигнали да се зададат с

блокове *Sine Wave*, *Pulse Generator* и *Repeating Sequence*. Да се направят експерименти с различни стойности на параметрите за трите входни сигнала.

4. За съставената блокова схема от предходната задача да се направят експерименти, променяйки стъпката на интегриране – да се промени стъпката като се зададат последователно стойности 0.3, 0.1, 0.01.

5. Да се направят експерименти върху последния модел, променяйки метода на интегриране.

Лабораторно упражнение 7
Моделиране и симулиране на системи в Simulink, зададени чрез математичните им модели

Цел на лабораторното упражнение

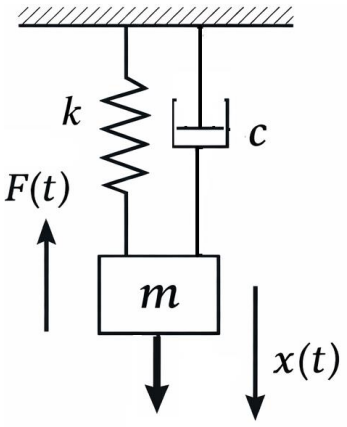
Целта на настоящото лабораторно упражнение е придобиване на знания за симулиране в средата на *Simulink* на системи, зададени посредством математичните им модели.

Въведение

За да бъде симулирана една система е необходимо да се знае нейният математичен модел, който най-често се описва с едно или няколко диференциални уравнения от n-ти ред.

Симулиране в Simulink на система, описвана с едно диференциално уравнение

За да бъде показана възможността за симулиране на система, представена чрез математичния й модел, ще бъде разгледана динамична система с една степен на свобода.



фиг. 7.1

Движението на масата m се описва с едно линейно диференциално уравнение от втори ред с постоянни коефициенти:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \qquad (1)$$

където:

m е масата на тялото (kg);

c – коефициент на демпфериране ($N.s/m$);

k – коравината на пружината (N/m);

F(t) – приложената върху тялото сила (*N*);

x(t) – преместването на тялото (*m*).

За да може да бъде създадена блоковата схема в *Simulink*, е необходимо уравнението да бъде преобразувано като бъде изразена най-високата производна на променливата *x(t)* чрез останалите параметри на системата. В случая най-високата производна е втора, т.е. уравнението ще изглежда по следния начин:

$$\ddot{x}(t) = \frac{1}{m}\left[F(t) - c\dot{x}(t) - kx(t)\right]. \qquad (2)$$

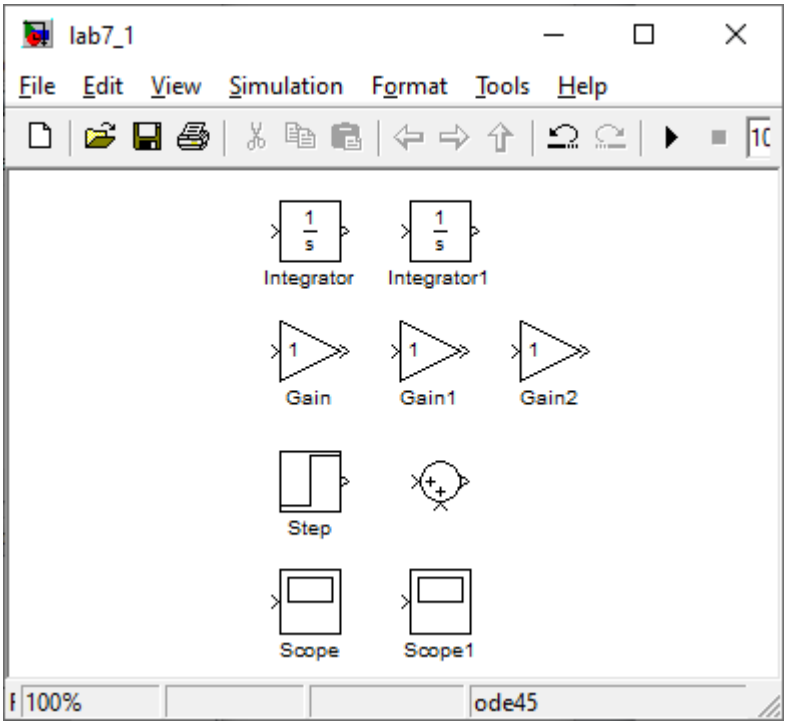
Задача: Да се симулира системата в *Simulink* за стойности на параметрите

$$m = 15kg,\ c = 1.5\ N.s/m,\ k = 0.7\ N/m$$

и големина на приложената външна сила $F(t) = 1\ N$.

За да бъде създадена блоковата схема на системата в *Simulink* са необходими следните блокове:

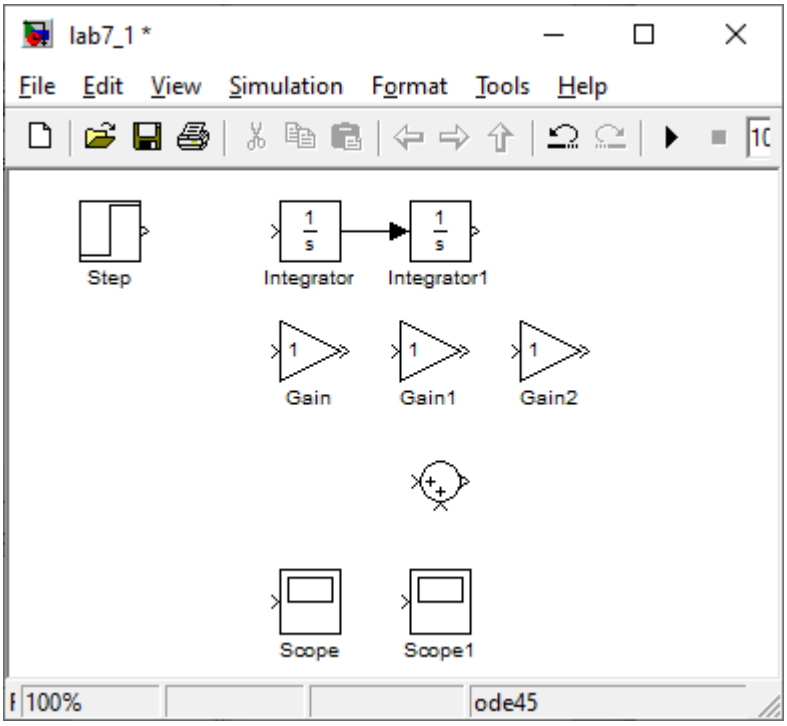
- блок за интегриране (*Int*) – 2 броя
- блок за задаване на входния сигнал (*Step*) – 1 брой
- блок за умножение с константа (*Gain*) – 3 броя (за произведенията $c\dot{x}$, kx и за константата $\frac{1}{m}$)
- блок за сумиране (*Sum*) – 1 брой
- блокове за визуализация (*Scope*)



фиг. 7.2

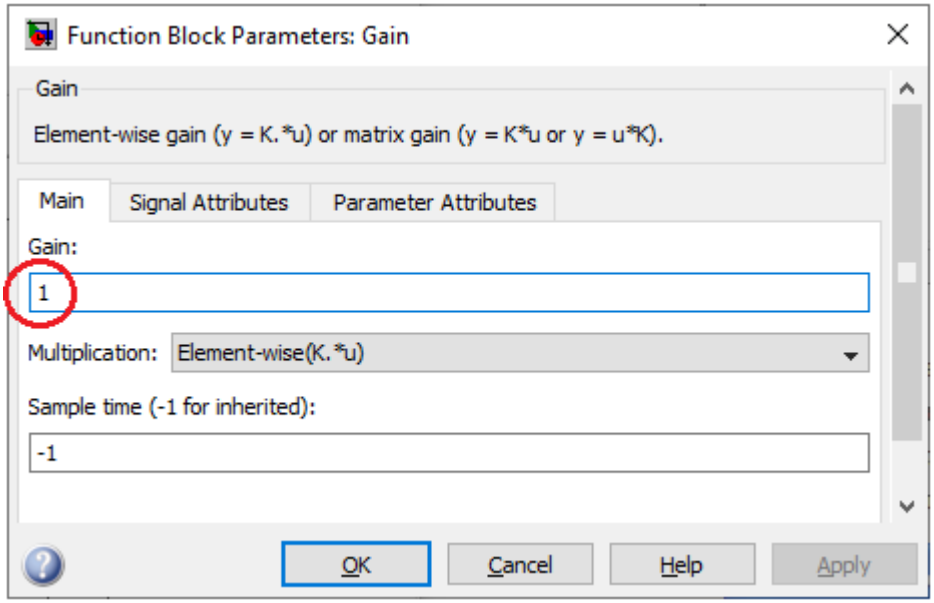
След като необходимите блокове са поставени в новия файл те трябва да бъдат свързани помежду си.

За да бъде получена променливата x е необходимо двукратно интегриране на втората ѝ производна, което се реализира чрез последователно свързване на двата блока *Int*.



фиг. 7.3

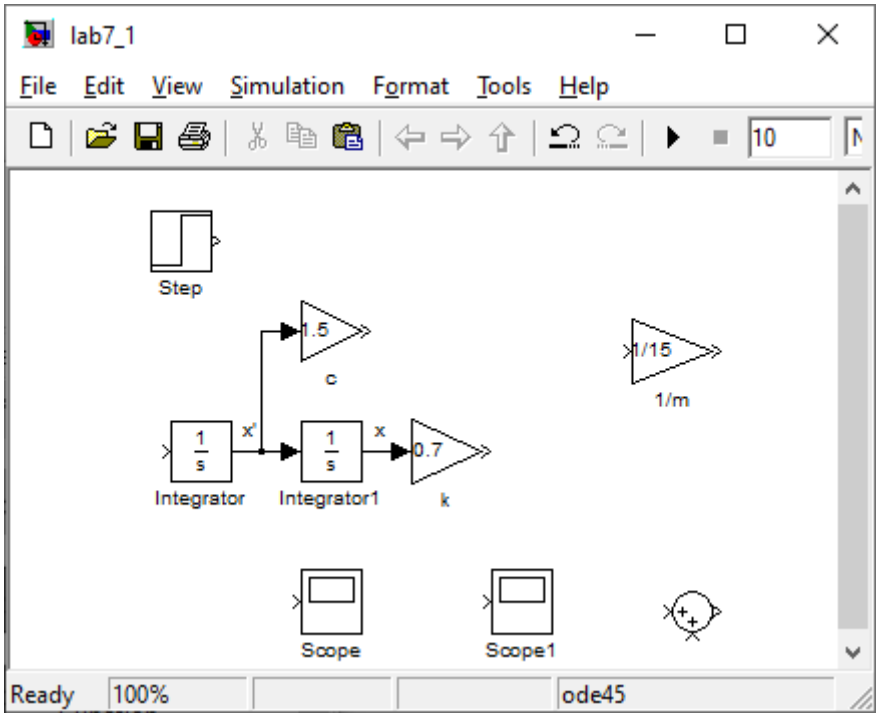
Следващата стъпка е задаване стойности на параметрите *c* и *k* и свързване на двата блока *Gain*. Числовите стойности на параметрите се задават след двукратно щракване с мишката върху съответния блок *Gain*. Отваря се прозорецът с параметрите и в полето *Gain* се записва стойността за съответния параметър.



фиг. 7.4

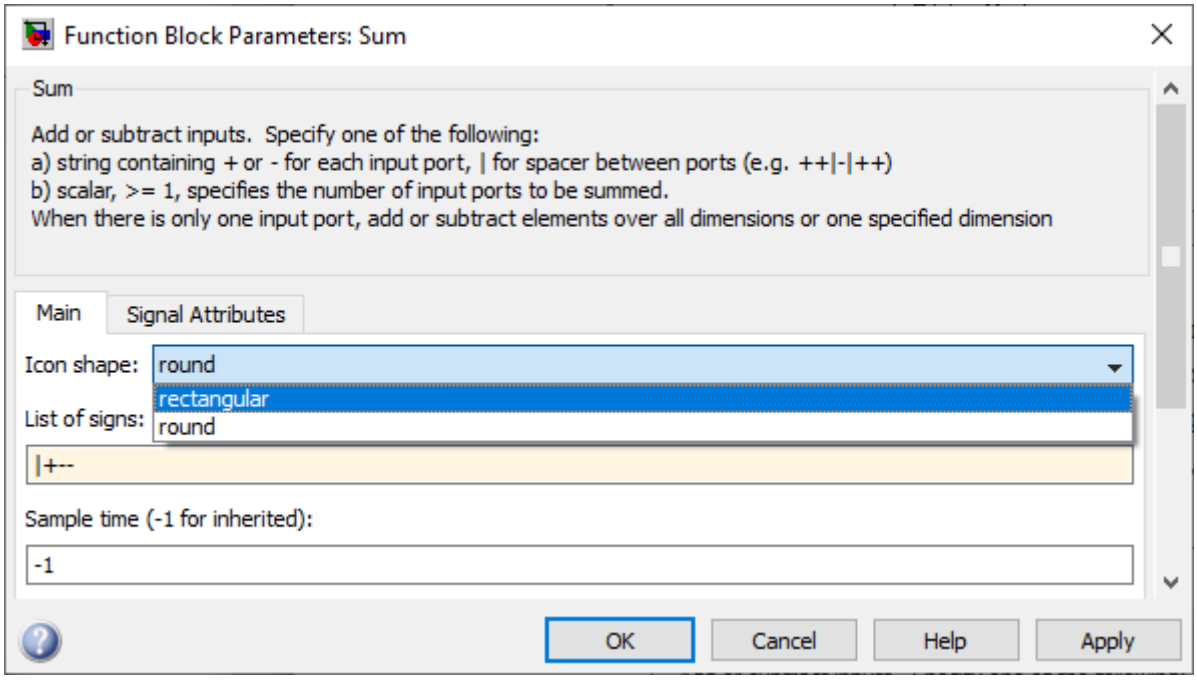
На блока с параметър *c* се подава сигнал от изхода на първия блок *Int* ($\dot{x}$), а входният сигнал на блока с параметър *k* е от изхода на втория блок *Int* (x).

На третия блок *Gain* също трябва да бъде зададена числова стойност. Чрез него ще бъде извършвано умножението с $\frac{1}{m}$, т.е. като стойност на параметъра на този блок се записва 1/15.



фиг. 7.5

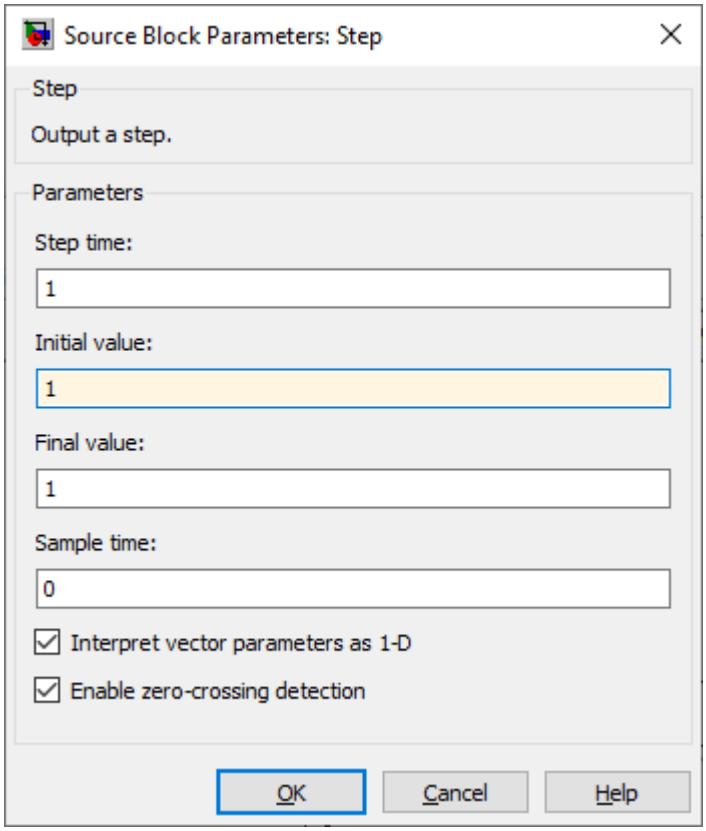
За да бъде продължена схемата, е необходимо промяна на броя входове на блок *Sum*. За целта се щраква два пъти с мишката върху него и в отворения диалогов прозорец се записват знаците на трите събираеми от разглежданото уравнение, които са „+--”. За по-голямо удобство в същия диалогов прозорец може да бъде променена и формата на блока – от кръгла на правоъгълна.



фиг. 7.6

На входовете със знак „-” се подават сигналите от изходите на двата блока *Gain*, а сигналът на входа със знак „+” е от блок *Step*.

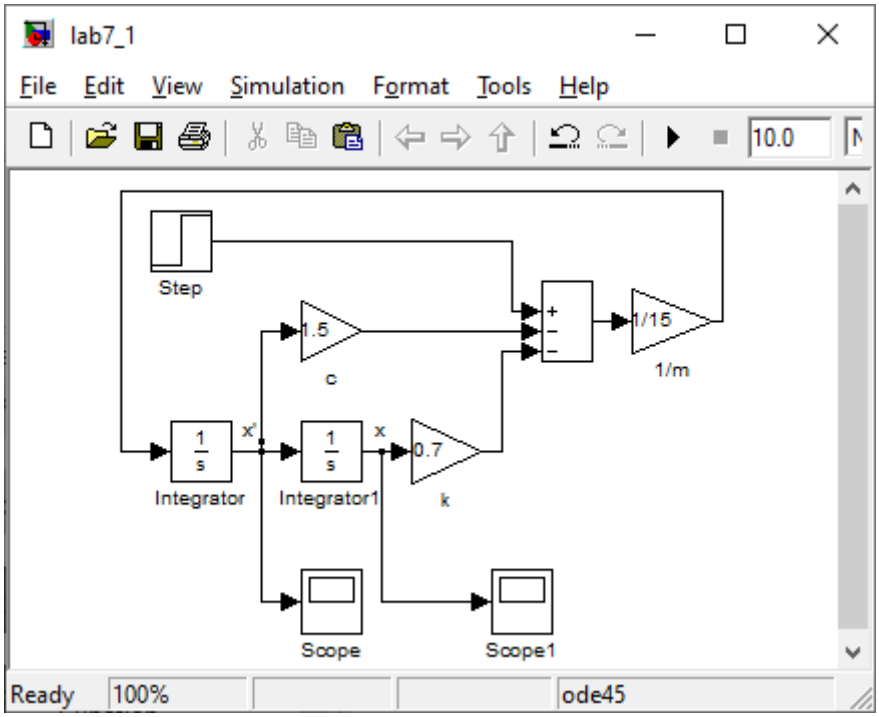
На блок *Step* се задават параметри, описващи желания входен сигнал (постоянно входно въздействие от 1N).



фиг. 7.7

За да бъде завършено описанието на системата трябва да се подаде сигнал на третия блок *Gain* от изхода на блок *Sum*.

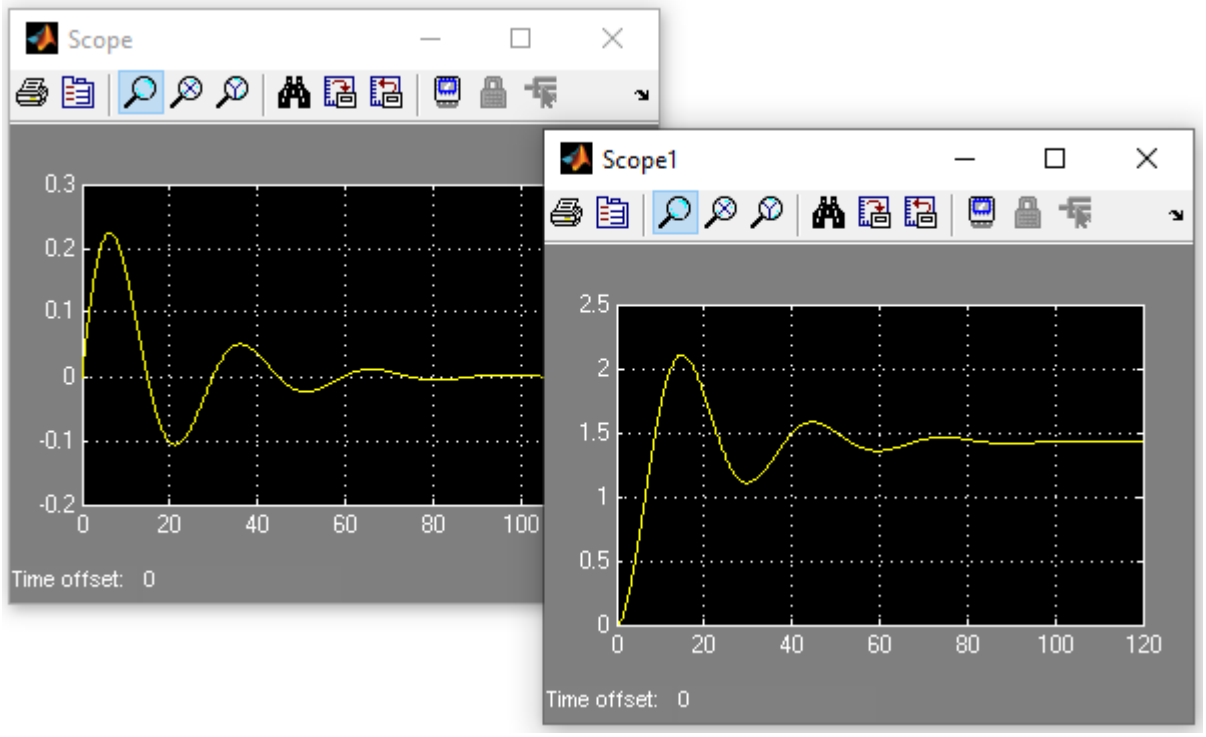
Чрез включването на блоковете за визуализация могат да бъдат наблюдавани променливите x (движението на тялото) и $\dot{x}$ (скоростта на тялото). За тази цел един блок *Scope* се свързва към изхода на втория блок *Int*, а втори блок *Scope* се свързва към изхода на първия блок *Int*.



фиг. 7.8

След като схемата е реализирана е необходимо да бъдат настроени и параметрите на симулацията. Т.к. преходният процес затихва бавно е необходимо да се увеличи времето на симулацията, напр. на 120s.

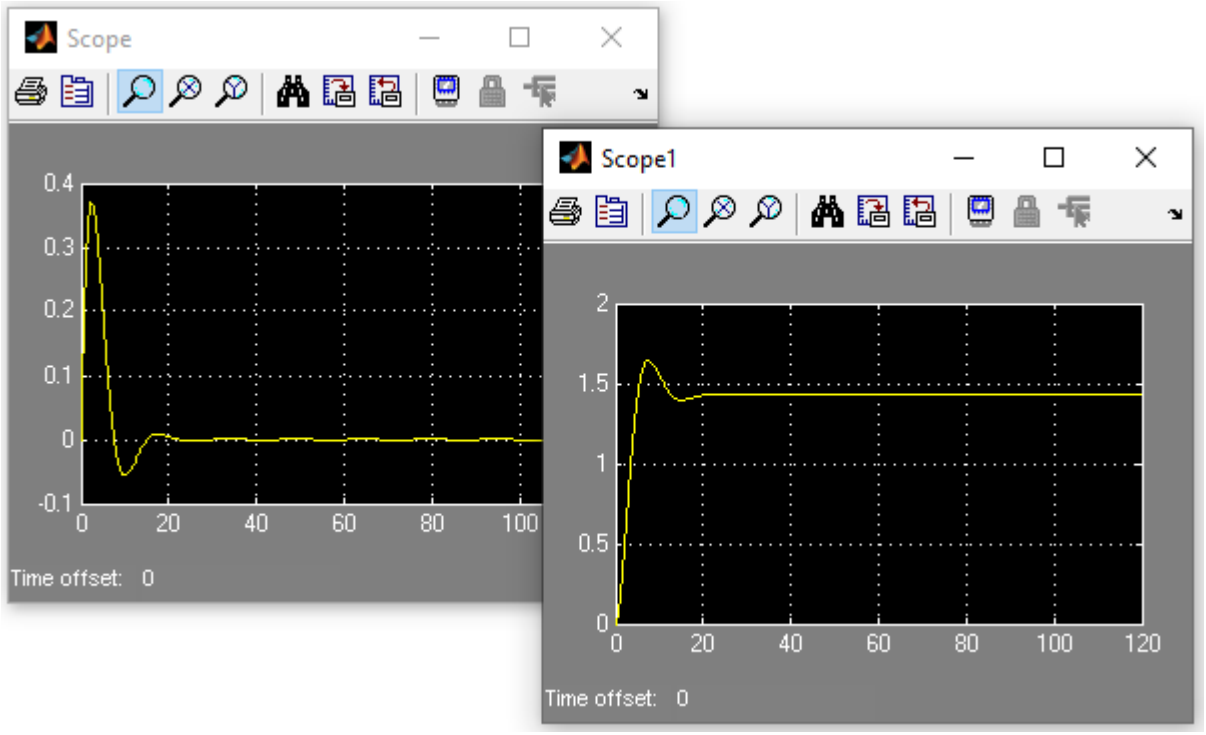
За да бъдат показани графиките, след изтичане на симулационното време, трябва да се щракне да пъти върху блоковете *Scope*.



фиг. 7.9

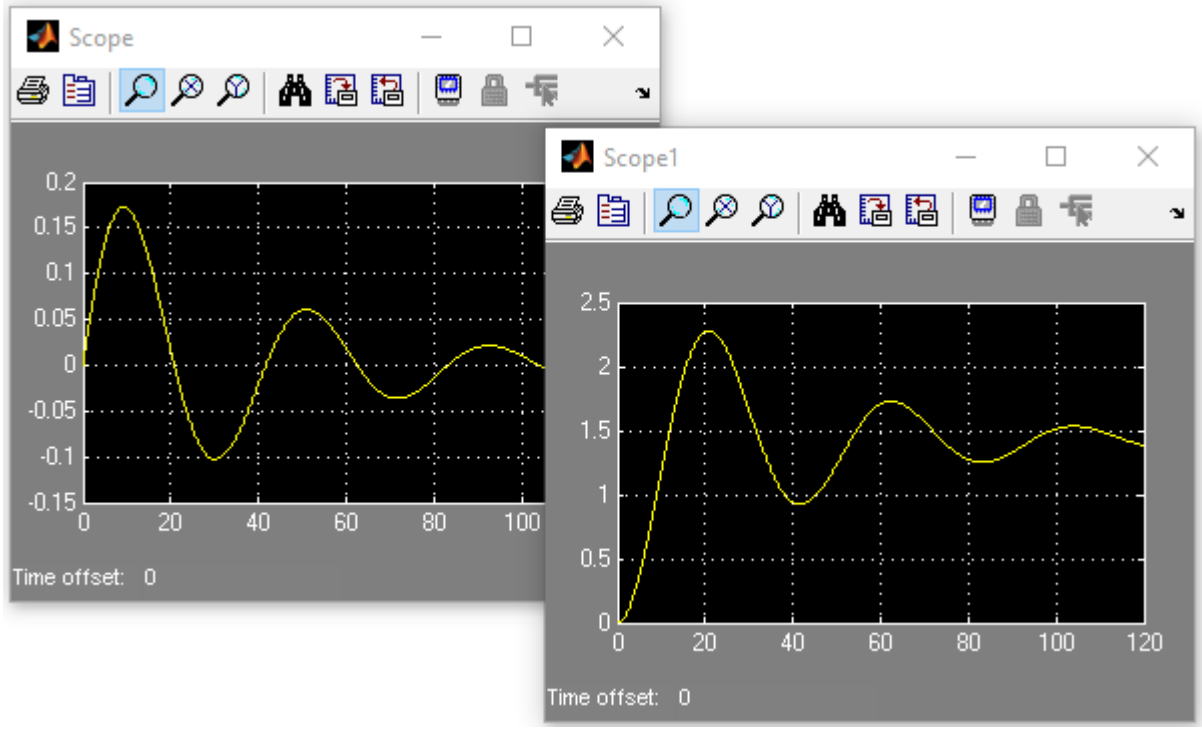
🤔 ***Задача***: Да се промени масата на тялото и да се наблюдава влиянието й върху движението на тялото.

m = 3kg



фиг. 7.10

m= 30kg



фиг. 7.11

Задачи за изпълнение:

1. За изследваната система да се направят експерименти с различни стойности на параметъра *c*.

2. За изследваната система да се направят експерименти с различни стойности на параметъра *k*.

3. За изследваната система да се промени видът на входното въздействие *f(t)* – напр. синусоидален сигнал с амплитуда (*Amplitude*) 3 и честота (*Frequency*) 10 rad/s.

Лабораторно упражнение 8
Моделиране и симулиране на системи от трети ред в Simulink по зададени математични модели

Цел на лабораторното упражнение

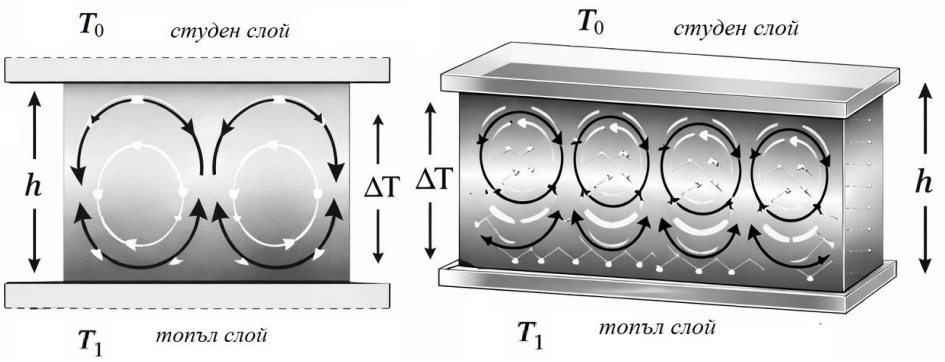
Целта на настоящото лабораторно упражнение е придобиване на знания за симулиране на системи от по-висок ред в средата на *Simulink*. Разглежданите системи са зададени посредством математичните им модели.

Въведение

За да бъде симулирана една система е необходимо да се знае нейният математичен модел, който най-често се описва с едно или няколко диференциални уравнения от n-ти ред.

Симулиране в Simulink на система, описвана с математичен модел, състоящ се от три диференциални уравнения от първи ред

За да бъдат представени възможностите за симулиране на динамични системи в *Simulink* ще бъде разгледан опростен математичен модел на динамиката на нагряван слой течност с дебелина h при постоянна разлика на температурата между горния и долния й граничен слой (фиг. 8.1).



фиг. 8.1

Като опростен модел на топлинна конвекция, моделът се описва със следната система уравнения:

$$\begin{aligned} &-\dot{x}_1 - ax_1 + ax_2 = 0, \\ &-x_1x_3 + rx_1 - x_2 - \dot{x}_2 = 0, \\ &x_1x_2 - bx_3 - \dot{x}_3 = 0, \end{aligned} \qquad (1)$$

със стойности на параметрите $a = 10, b = 2.67$ и $r = 28$.

Физическата същност на променливите при модела на нагрявания слой течност е:

- x_1 - скорост на конвективна циркулация на флуида (за $x_1 > 0$ циркулацията е по посока на часовниковата стрелка, при $x_1 < 0$ - в обратна посока);

- x_2 - температурна разлика между издигащия и падащия поток;

- x_3 - температурен градиент.

Преди да започне изграждането на блоковата схема е необходимо уравненията на системата да бъдат преобразувани така, че най-високата производна на променливите (в случая първа производна) да бъде от едната страна на уравнението, всичко останало – от другата. След преобразуването се получава следната система:

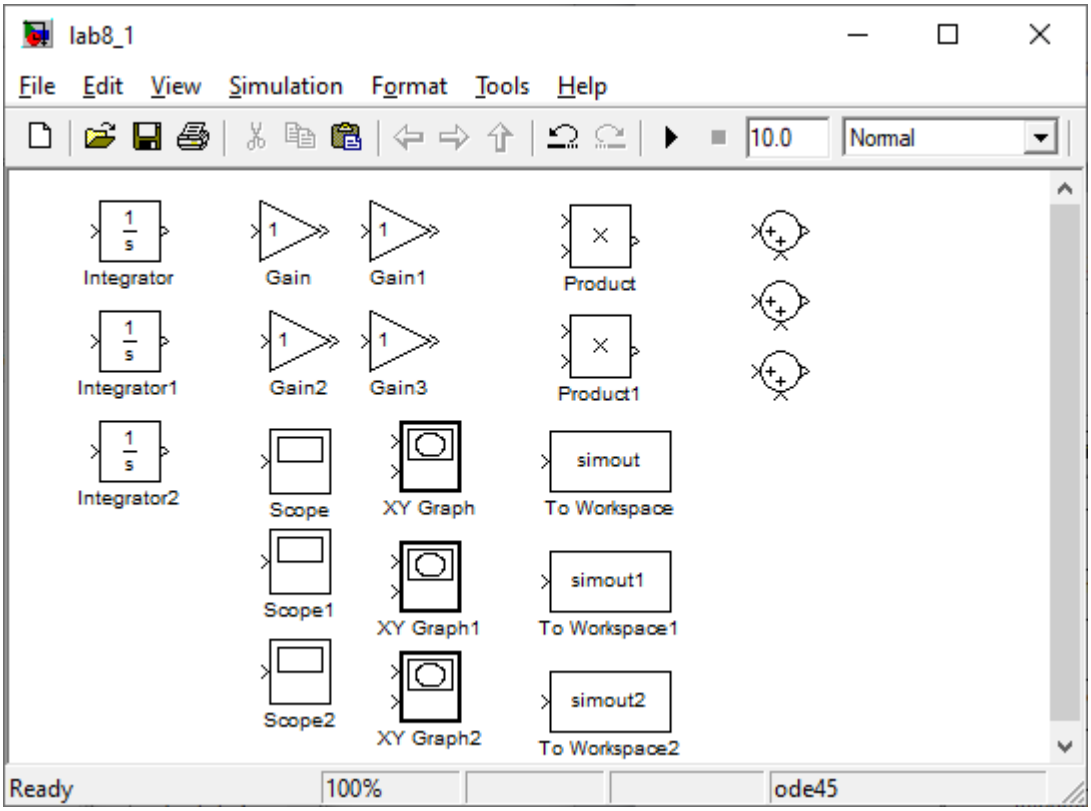
$$\begin{aligned} \dot{x}_1 &= -ax_1 + ax_2, \\ \dot{x}_2 &= -x_1x_3 + rx_1 - x_2, \\ \dot{x}_3 &= x_1x_2 - bx_3, \end{aligned} \qquad (2)$$

След преобразуване на модела може да се започне изграждането на схемата в *Simulink*.

За симулиране на дадената система ще бъдат представени два варианта на блоковата схема в *Simulink* – чрез използване на отделни функционални блокове за всяко математично действие от уравненията в модела, и като втори вариант – чрез използване на специален функционален блок, в който по определен начин може да бъде записано всяко от зададените уравнения.

Вариант 1

За изграждане на блоковата схема за разглежданата система са необходими следните блокове, с помощта на които ще бъдат описани всички математични действия: (фиг.8.2)



фиг. 8.2

- блок за интегриране (*Int*) – 3 броя (по един за трите променливи - $x_1,\ x_2,\ x_3$)
- блок за умножение с константа (*Gain*) – 4 броя (за произведенията $ax_1,\ ax_2,\ rx_1,\ bx_3$)
- блок за умножение (*Product*) – 2 броя (x_1x_3, x_1x_2)
- блок за сумиране (*Sum*) – 3 броя (за трите уравнения)
- блокове за визуализация (*Scope, XY Graph*)
- блок за експортиране на данни в средата на *Matlab* (*To Workspace*)

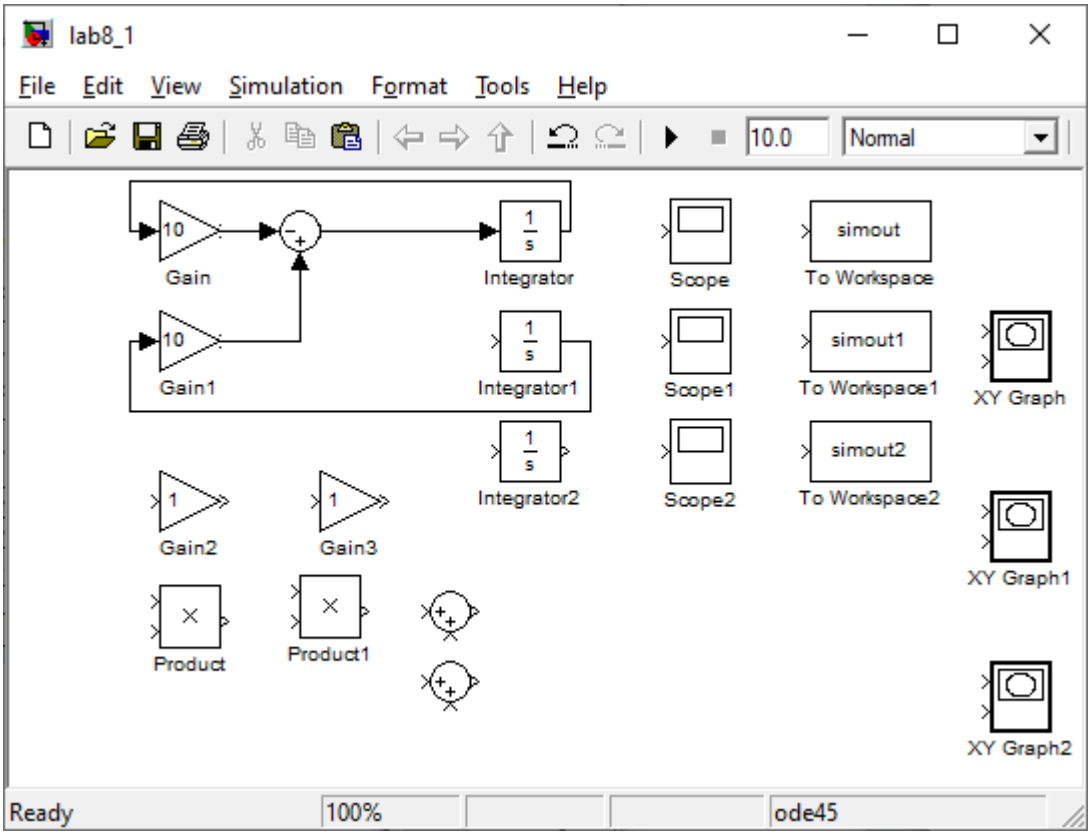
След като необходимите блокове са поставени в новия файл, е необходимо те да бъдат свързани съгласно уравненията от математичния модел.

За реализиране на действията от първото уравнение на системата са необходими блокове *Gain, Sum, Int*. На входа на първия интегратор е необходимо да се подаде сигналът $\dot{x}_1$, който се описва с първото уравнение. На изхода на същия интегратор се получава сигнал x_1.

Умножението на константа a с x_1 става чрез блок *Gain*, на входа на който се подава сигнал от първия интегратор, а като параметър на блока се

задава константата 10. Аналогично се извършва свързването на втория блок *Gain*, на входа на който се подава сигнал от втория интегратор (т.е. сигнал x_2).

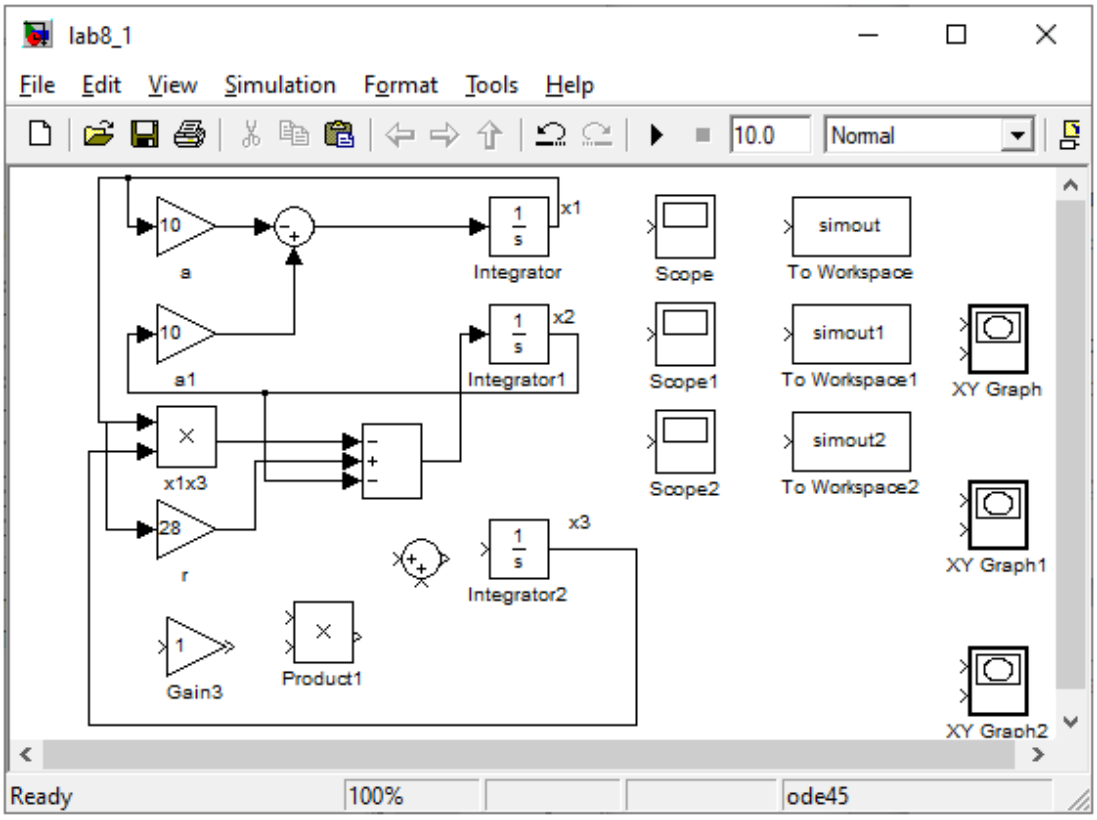
За да бъдат продължени действията в първото уравнение, е необходимо да бъдат променени входовете на блок *Sum* – трябва да бъдат зададени знаци „–” и „+”. На входа със знак „-” трябва да се подаде сигналът от изхода на първия блок *Gain* (ax_1), на входа със знак „+” – изхода на втория блок *Gain* (ax_2). Сигналът, получен на изхода на блок *Sum*, представлява променливата $\dot{x}_1$, която трябва да се подаде на входа на първия интегратор.



фиг. 8.3

По гореописания начин трябва да бъдат свързани и останалите блокове, описващи действията от второто и третото уравнение от математичния модел на разглежданата система.

За второто уравнение е необходимо да бъде променен броят на входовете на блок *Sum*, което става чрез двукратно щракване с мишката върху него и записване последователно на знаците -+-.

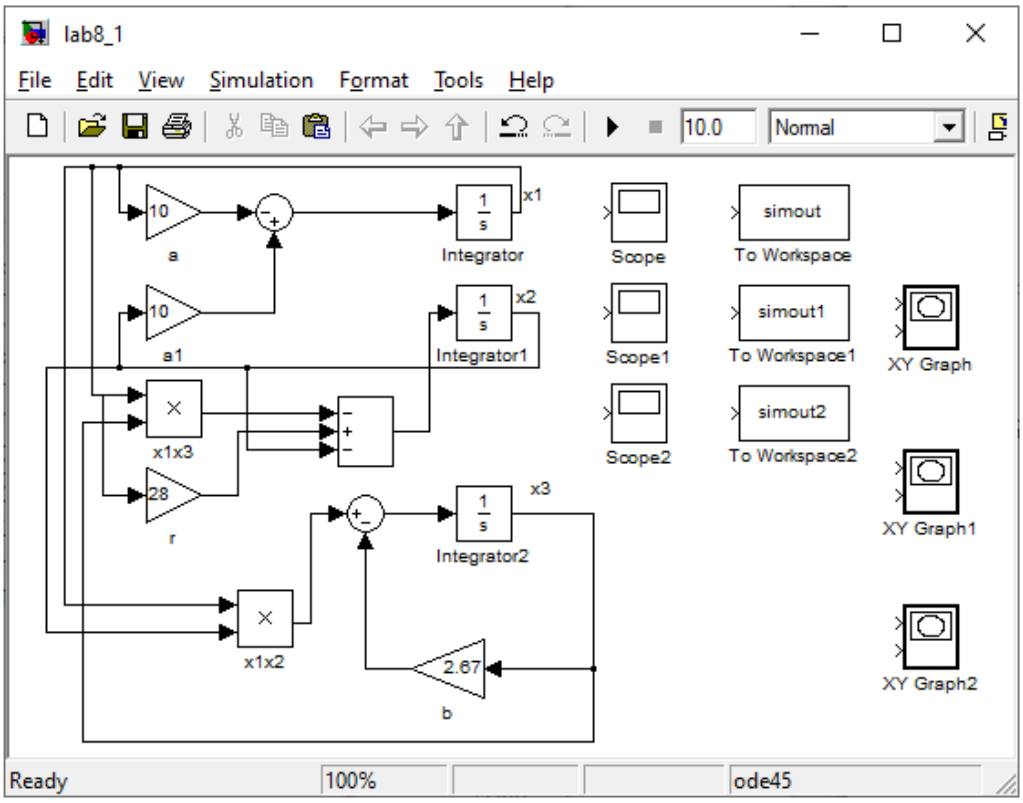


фиг. 8.4

За удобство при разглеждане на блок-схемата, могат да бъдат променяни имената на блоковете (чрез щракване с мишката върху името на блока и записване на желания текст), както и да бъдат поставяни коментари над свързващите линии (чрез двукратно щракване с мишката на произволно място над линията и записване на желания текст в появилото се текстово поле).

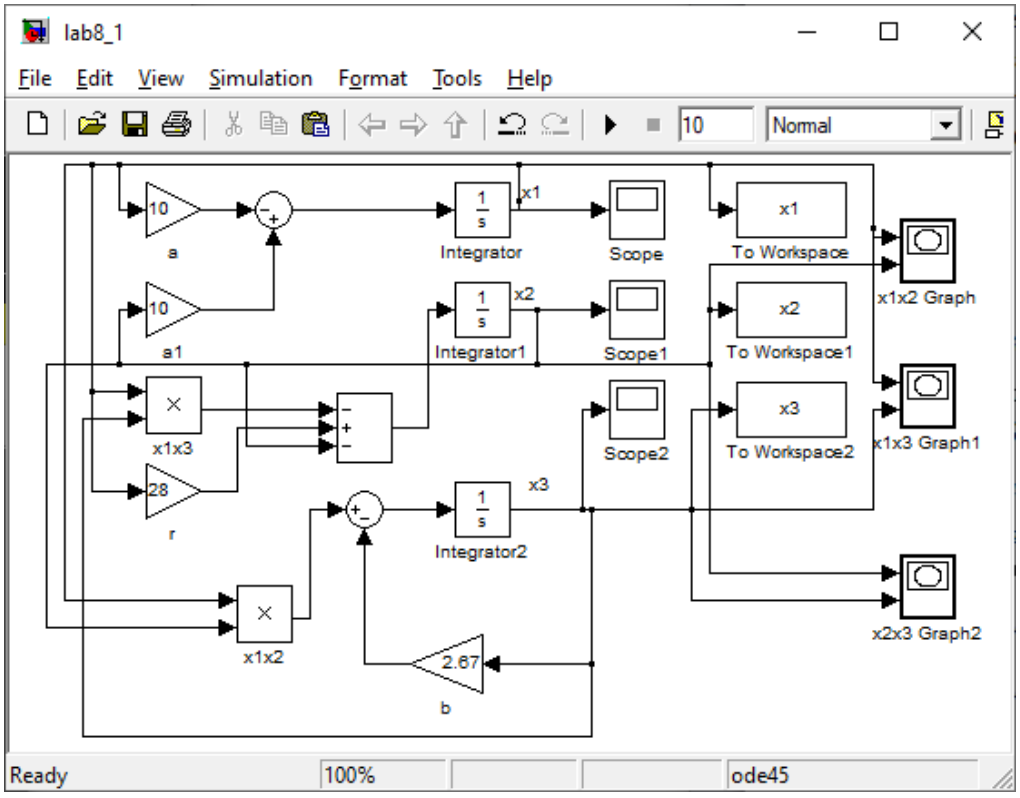
След като математичните действия от модела на системата са реализирани в *Simulink*, в блоковата схема остава да се свържат блоковете за визуализация.

За този вариант на задачата са представени графики във времето на трите променливи x_1, x_2, x_3, всяка от които се визуализира в отделен графичен прозорец. Това се реализира чрез три отделни блока *Scope*, като на входа на всеки от тях се подава сигналът от изхода на съответния интегратор. (фиг. 8.5)



фиг. 8.5

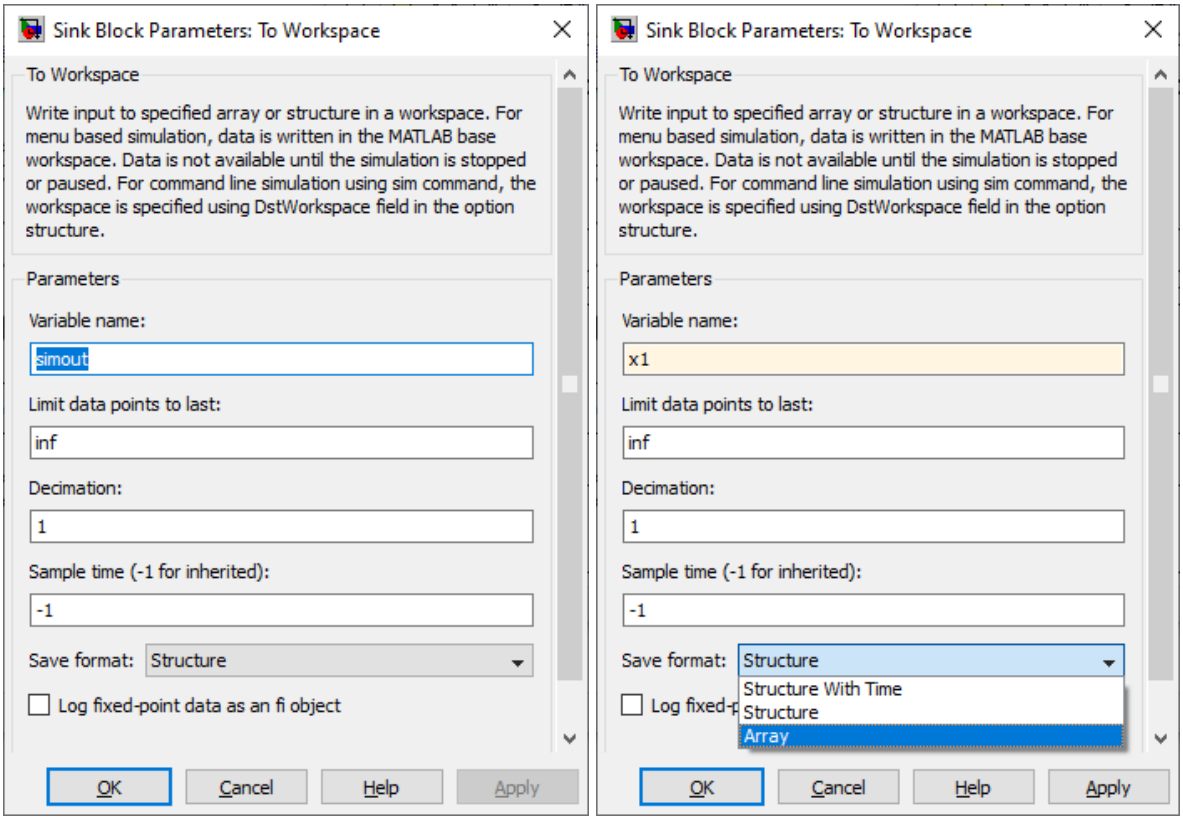
За получаване на двумерни графики от типа x_1x_2, x_1x_3, x_2x_3, се използват блоковете *XY Graph*, на входовете на които се подават два сигнала, за съответните променливи - x_1, x_2, x_3. (фиг. 8.6)



фиг. 8.6

За изчертаване на тримерна графика е необходимо експортиране на стойностите за трите променливи в средата на *Matlab* с помощта на блокове *To Workspace*. За целта са използвани три такива блока, по един за трите променливи.

Блок *To Workspace* има подразбиращи се параметри, показани на фиг.8.7а.



а).....б)

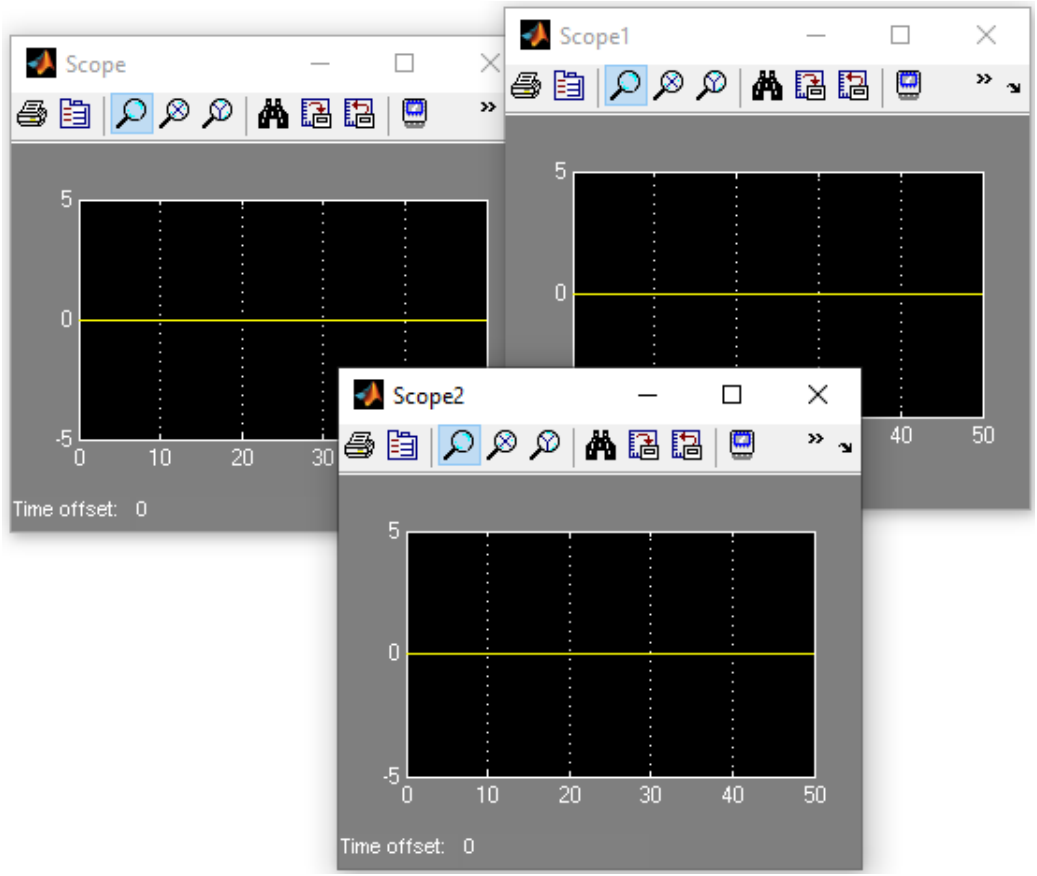
фиг. 8.7

Това, което е необходимо да се промени, е форматът за записване на данните от „*Structure*” на „*Array*” (фиг. 8.7б).

По желание могат да се променят имената, с които се записват променливите в работното пространство на Matlab (в полето *Variable name*). По подразбиране името е *simout*, като то може да бъде променено например на “*x1*” (фиг. 8.7б).

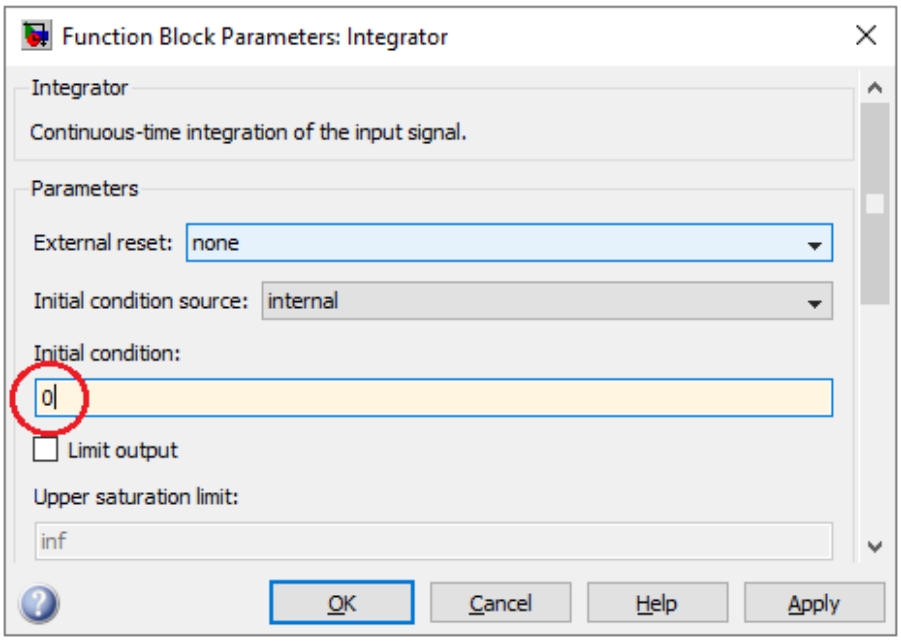
След като са свързани всички блокове, остава да бъдат настроени параметрите на симулацията (*Simulation -> Configuration Parameters*). По подразбиране времето на симулация е 10s, което може да бъде увеличено (напр. на 50s), а за по-прецизна графика може да бъдат променени типа и размера на стъпката – *Type – Fixed-step, Size* – 0.01.

🤔 ***Задача***: Да се стартира симулацията и да се наблюдават графиките за трите променливи x_1, x_2, x_3.



фиг. 8.8

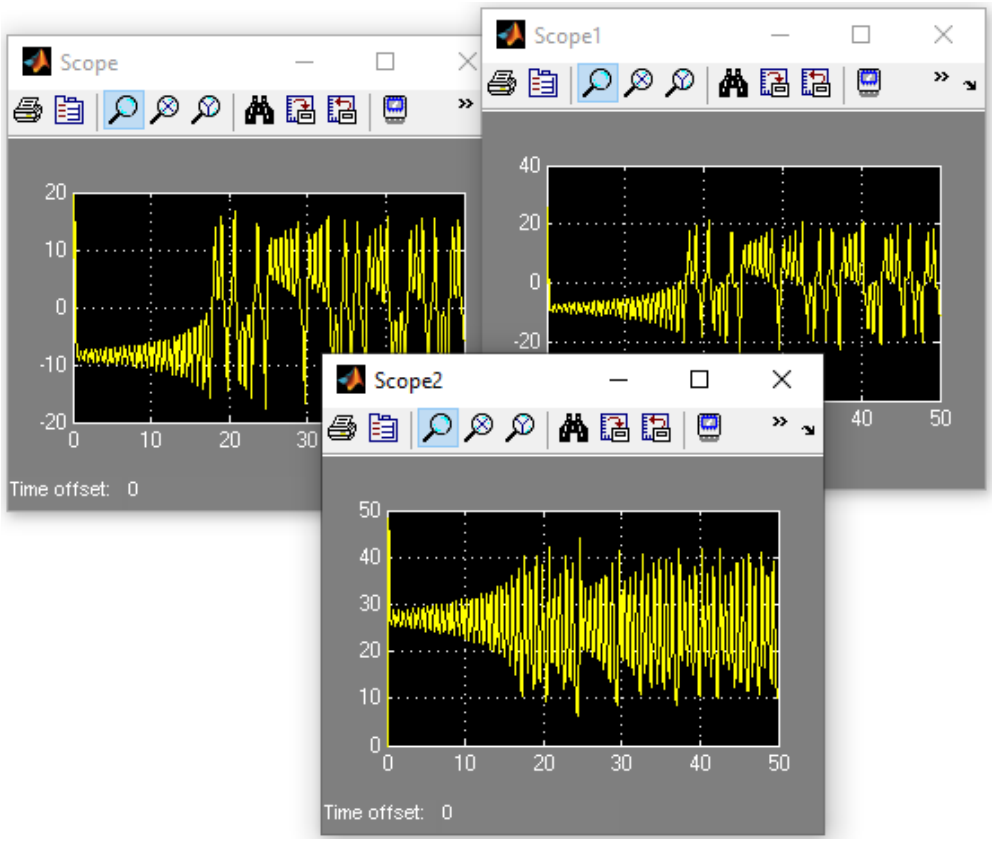
Стартирайки създадената схема при така зададените параметри в системата няма да настъпят изменения в състоянието (фиг. 8). Причината за това е, че в системата няма външно въздействие, поради което тя остава в равновесно състояние. От това състояние, освен чрез външен сигнал, тя може да бъде изведена чрез ненулеви начални условия, които се задават в блока с параметри на интеграторите, параметър „*Initial condition*” (фиг.8.9).



фиг. 8.9

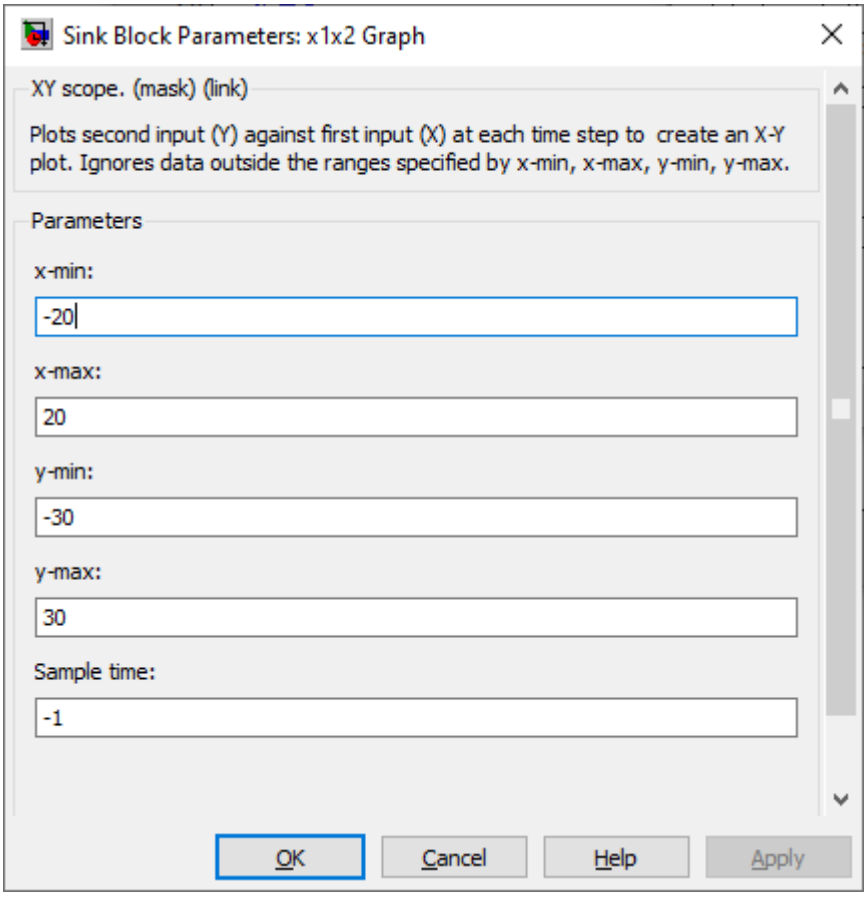
След като бъдат променени стойностите на началните условия на интеграторите, може да бъде стартирана симулацията на системата.

Графиките, които се получават за променливите x_1, x_2, x_3, могат да бъдат отворени чрез двукратно щракване с мишката върху трите блока *Scope*.



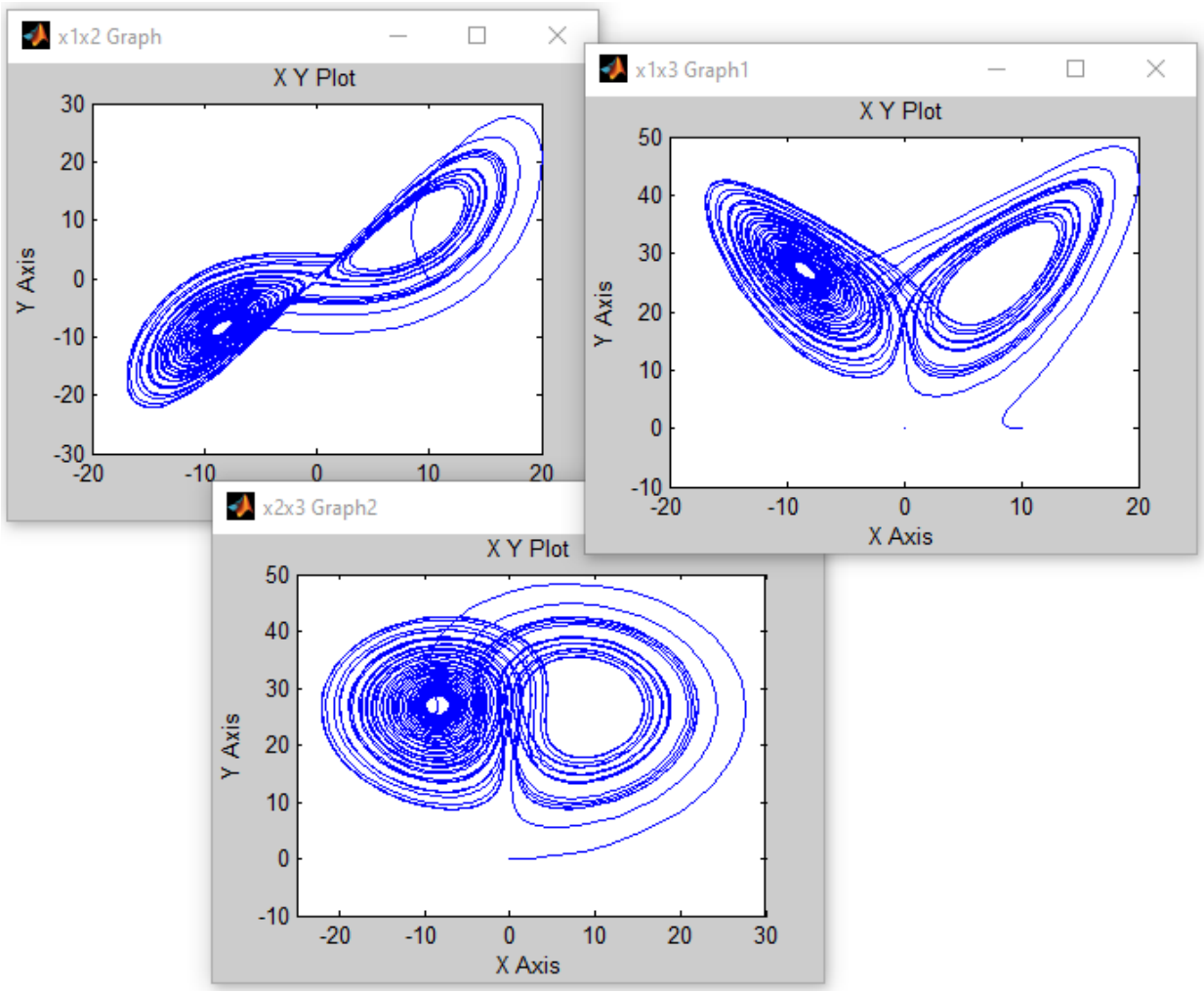
фиг. 8.10

Чрез блокове *XY Graph* се визуализират графиките x_1x_2, x_1x_3, x_2x_3 (фиг.8.12). В този случай, за да бъдат визуализирани графиките коректно, е необходимо да се извърши ръчно мащабиране на кооординатните оси. За тази цел се отваря диалоговия прозорец с параметрите чрез двукратно кликване с мишката върху съответния блок *XY Graph*. Видът на прозореца е показан на фиг.8.11. В него трябва да бъдат зададени диапазоните за изчертаване на графиката по абсцисната (*x-min, x-max*) и по ординатната ос (*y-min, y-max*). това трябва да бъде направено поотделно за всеки от трите блока *XY Graph*.



фиг. 8.11

След като са мащабирани координатните оси, могат да се наблюдават получените графики, като те се визуализират автоматично на екрана при стартиране на симулацията.



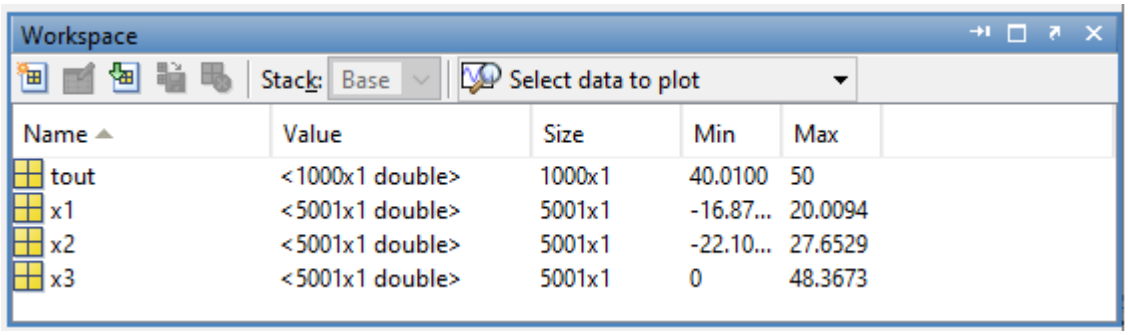
фиг. 8.12

За да бъде изчертана тримерна графика, е необходимо в прозореца *Command Window* на *Matlab* да се запише командата

>> plot3(x1, x2, x3)

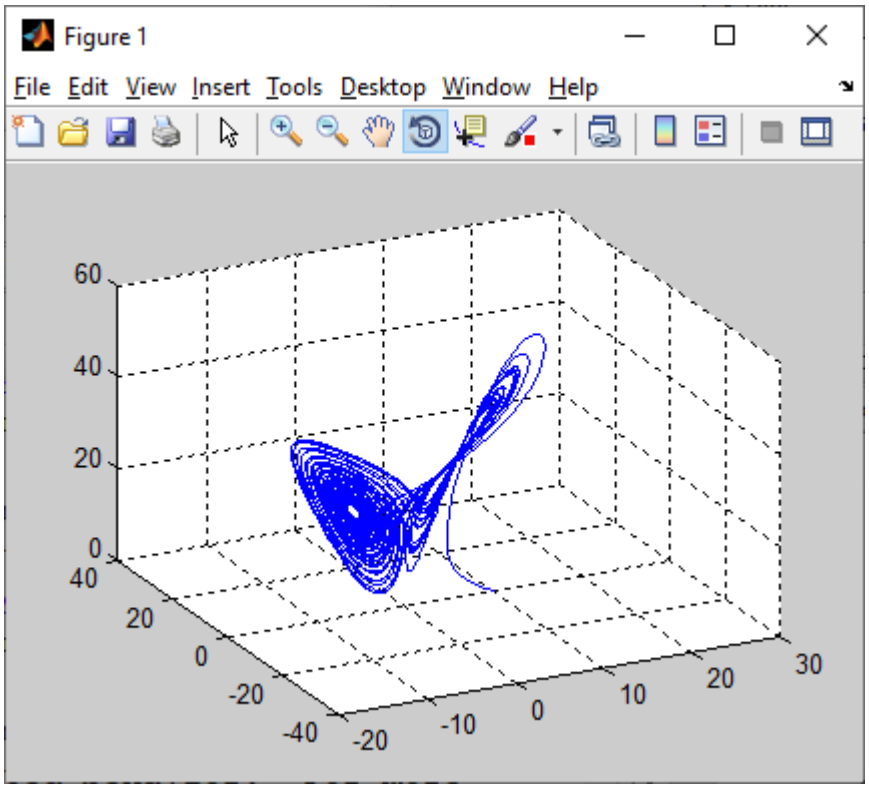
където x1, x2, x3 са имената на променливите, зададени в блокове *To Workspace*.

Това е възможно ако данните за трите променливи са експортирани правилно в средата на *Matlab*. (фиг.8.13)



фиг. 8.13

В този случай получената тримерна графика е показана на фиг.8.14.



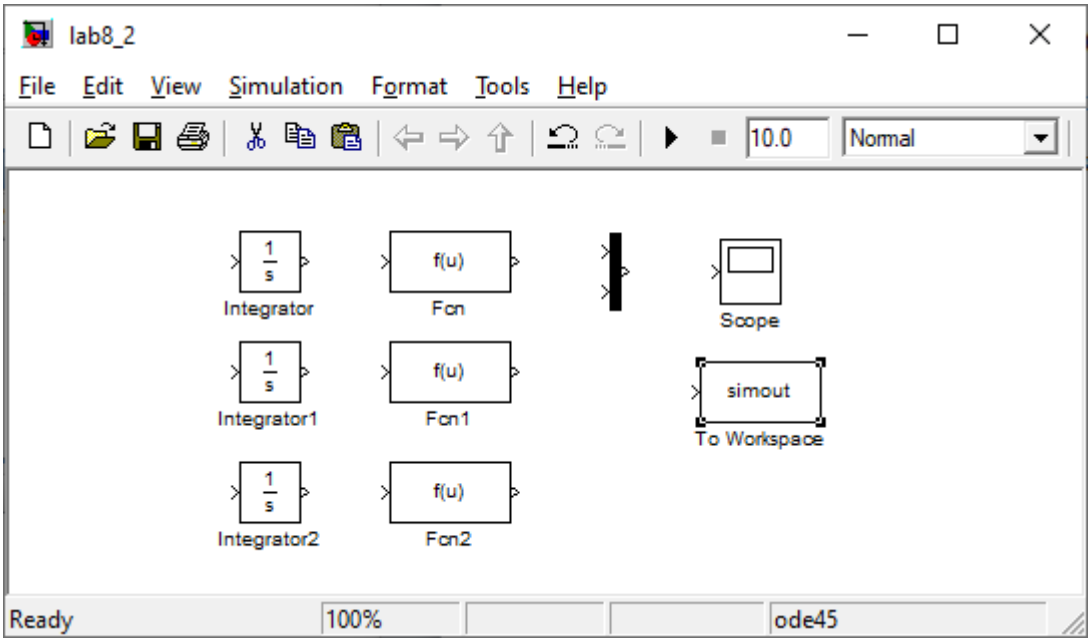
фиг. 8.14

Вариант 2

За изграждане на втори вариант на блоковата схема за разглежданата система са необходими следните блокове:

- блок за интегриране (*Int*) – 3 броя (по един за трите променливи - x_1, x_2, x_3)
- блок за описание на функциите (*Fcn*) – 3 броя (за трите уравнения)
- блок за комбиниране на трите променливи x_1, x_2, x_3 в една обща структура (*Mux*)
- блоко за визуализация (*Scope*)
- блок за експортиране на данни в средата на *Matlab* (*To Workspace*)

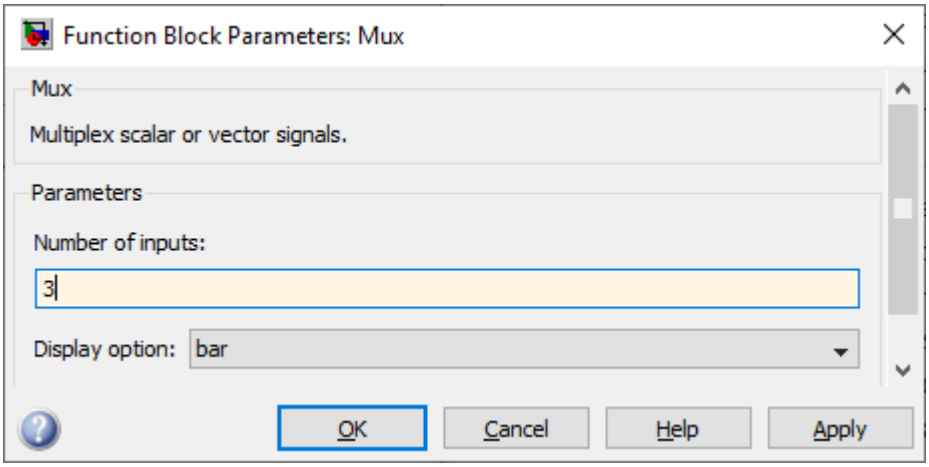
На фиг. 8.15 са показани всички блокове, необходими за изграждане на блоковата схема.



фиг. 8.15

Следващата стъпка за реализиране на системата в *Simulink* е свързването на отделните блокове. В случая отделните уравнения от математичния модел се описват не чрез отделни блокове за всяко математично действие (събиране, изваждане, умножение с константа и т.н.), а чрез блокове, описващи цялото уравнение (*Fcn*). Блок *Fcn* има един вход и един изход. В случая, който разглеждаме, променливите в системата са три (x_1, x_2, x_3) и те трябва да бъдат подадени на този блок като входни параметри. За да се осъществи това, трите променливи могат да бъдат обединени в обща структура и след това да бъдат подадени на входа на блок *Fcn*. За тази цел се използва блок *Mux*, чието действие е обединяване на параметрите, подадени на входовете му, в един общ масив.

По подразбиране блок *Mux* има два входа, но броят им може да бъде променен. При двукратно щракване с мишката върху блока се отваря прозорецът с параметри, показан на фиг. 8.16, в който в полето *Number of inputs* се задава броя на входовете. За разглеждания случай броят на входните параметри е 3.



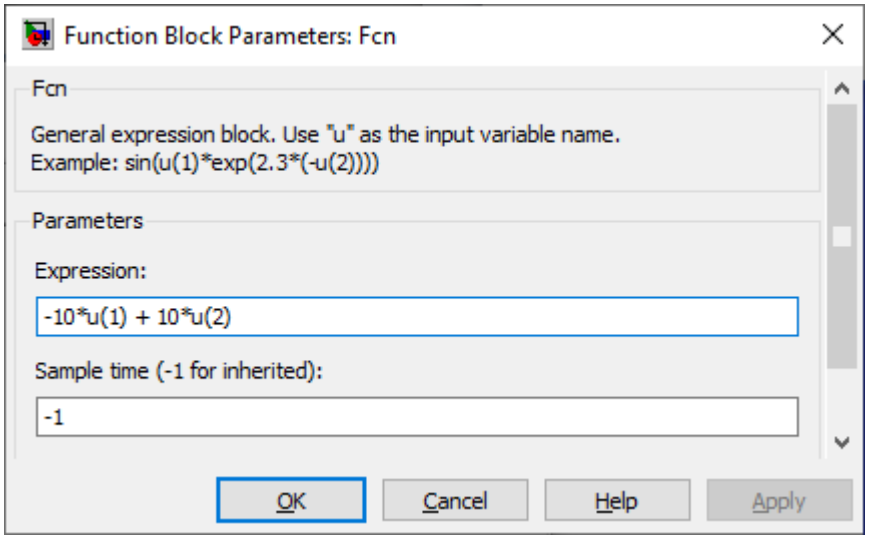
фиг. 8.16

Сигналите за тези входове се получават от изходите на трите интегратора (т.е. трябва да се свържат последователно изходите на интеграторите с входовете на блок *Mux*).

Следващата стъпка е описание на уравненията от математичния модел. За тази цел трябва да се щракне два пъти върху първия от блоковете *Fcn*, при което се отваря прозорецът с параметри (фиг. 8.17). В полето *Expression* се описват действията на входните променливи.

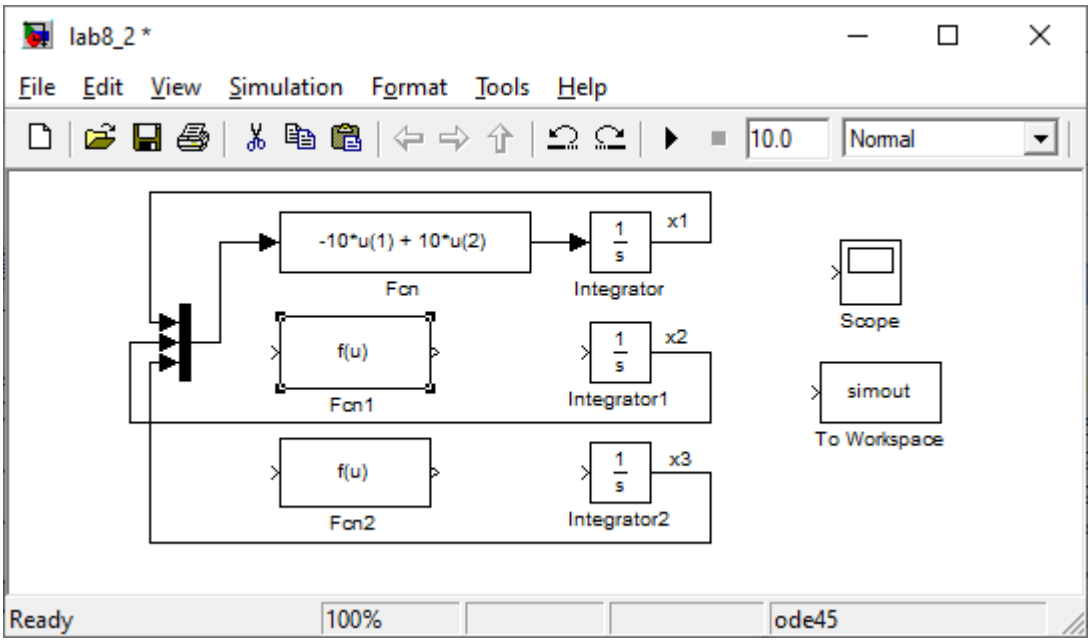
За задаване на променливите се използва записът *u(1)*, *u(2)*, *u(3)* и т.н., като цифрата в скобите съответства на номера на входа на блок *Mux*. Използвайки тези означения и отчитайки факта, че на първия вход на блок *Mux* е подаден сигнал x_1, на втория вход – сигнал x_2 и на третия вход – сигнал x_3, за първото уравнение от разглежданата система в полето *Expression* трябва да се запише израза

-10*u(1) + 10*u(2)



фиг. 8.17

След като е записана информацията в поле *Expression*, остава да се свържат блоковете – изходът на блок *Mux* с входа на *Fcn* и изхода на *Fcn* с входа на *Int*.



фиг. 8.18

По аналогичен начин е необходимо да се опишат и останалите две уравнения. В поле *Expression* на втория блок *Fcn* трябва да се запишат действията от второто уравнение на системата по следния начин:

$$-u(1)*u(3) + 28*u(1) - u(2)$$

а за описване на третото уравнение, в третия блок *Fcn* се записва

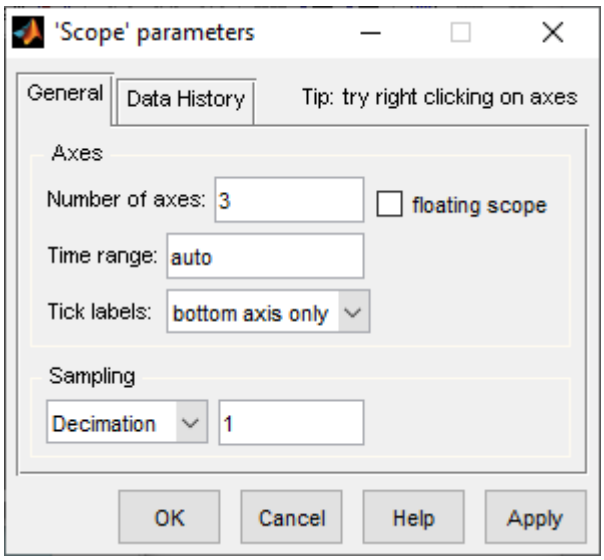
$$u(1)*u(2) - 2.67*u(3)$$

Сигналите на входовете на тези два блока също се получават от изхода на блок *Mux*. Свързването става от входа на дадения блок (*Fcn*) към вече съществуващата линия от изхода на блок *Mux* (щрака се с мишката върху входа на блок *Fcn* и се влачи до споменатата линия) или чрез разклоняване на съществуващата линия (лаб.упр. № 6). С тези действия блоковете, описващи системата, са свързани.

За да бъде завършена блоковата схема, е необходимо да бъдат включени блоковете, свързани с визуализацията на променливите и експортирането им в средата на *Matlab – Scope* и *To Workspace*.

В първия вариант на задачата беше показано визуализирането на трите променливи с три отделни блока *Scope*, при което всяка една от тях се изчертава в отделен прозорец. Има възможност трите променливи да бъдат

изчертани на три отделни координатни системи, но в един прозорец. Това се реализира чрез промяна на един от параметрите на блок *Scope*. Кликвайки с мишката два пъти върху блока, се отваря прозореца с параметрите. В полето *General* има параметър *Number of axes*, на който е необходимо да се зададе стойност 3, т.к. толкова са променливите, които ще визуализираме (фиг. 8.19).

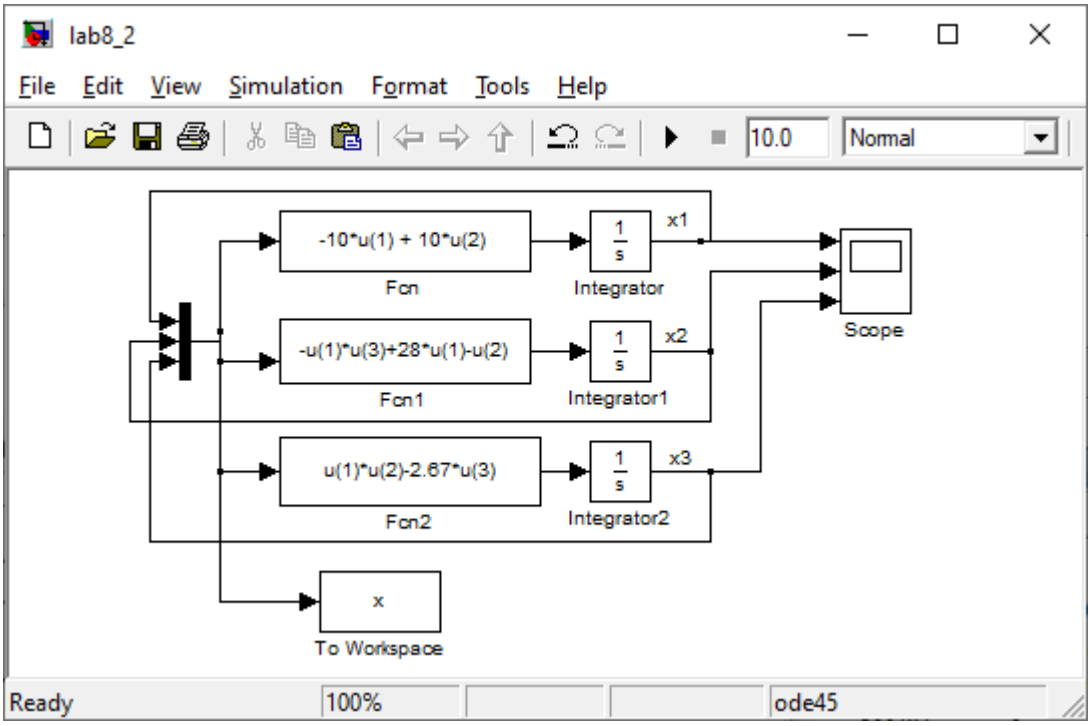


фиг. 8.19

След натискане на OK може да се види, че блок *Scope* вече има не един, а три входа. Към тези три входа е необходимо да бъдат подадени сигналите от трите блока *Int*, които представляват променливите x_1, x_2, x_3. Изчертаването на променливите става в последователността, в която са подадени на трите входа, т.е. променливата, подадена на първия вход се визуализира на първата координатна система и т.н.

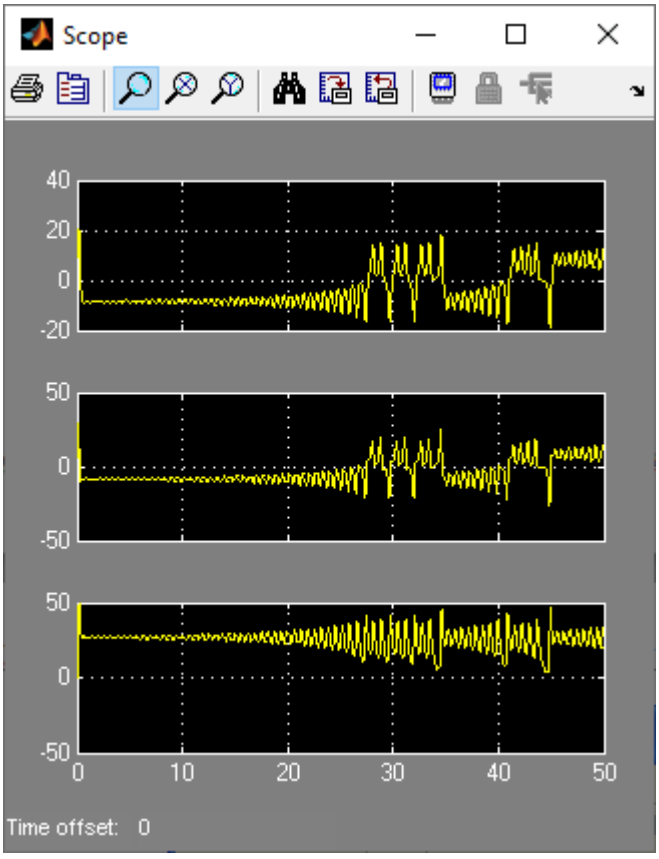
За експортиране на данните в работното поле на *Matlab* се използва блок *To Workspace*.

В първия вариант на задачата реализацията беше чрез три еднакви блока - по един за трите променливи x_1, x_2, x_3. Във втория вариант ще бъде представен друг начин за използване на блока като експортирането на трите променливи ще бъде осъществено чрез един блок *To Workspace*. Т.к. този блок има един вход, а в случая е необходимо да бъдат подадени данни за три променливи, то е необходимо използването на масива, в който те са обединени (т.е. изходът на блок *Mux*). Отново е необходимо промяна на формата от *Structure* на *Array*, а името на масива може да бъде променено по желание.



фиг. 8.20

Последна стъпка при симулиране на системата, е настройка на параметрите на симулацията и задаване на начални условия на блоковете за интегриране.



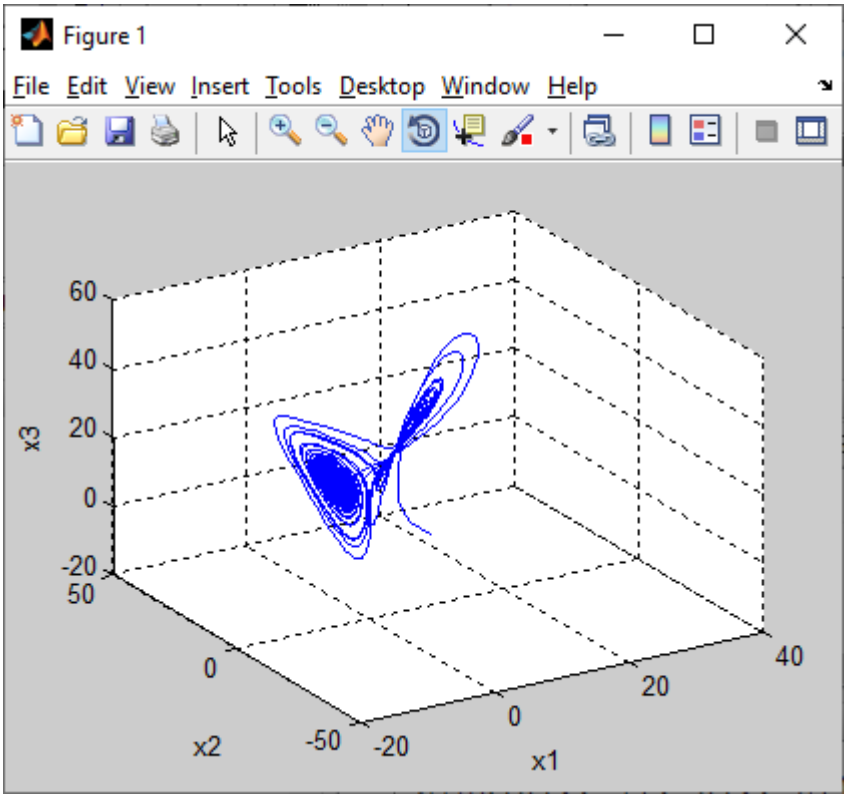
фиг. 8.21

С това, блоковата схема на разглежданата система е готова и може да бъде стартирана симулацията.

На фиг. 8.21 са показани времевите графики, които се изчертават с помощта на блок Scope.

За да бъде изчертана тримерната графика (фиг.8.22), се използва отново команда *plot3*, но с различно зададени аргументи (данните от първи, втори и трети стълб на масив *x*):

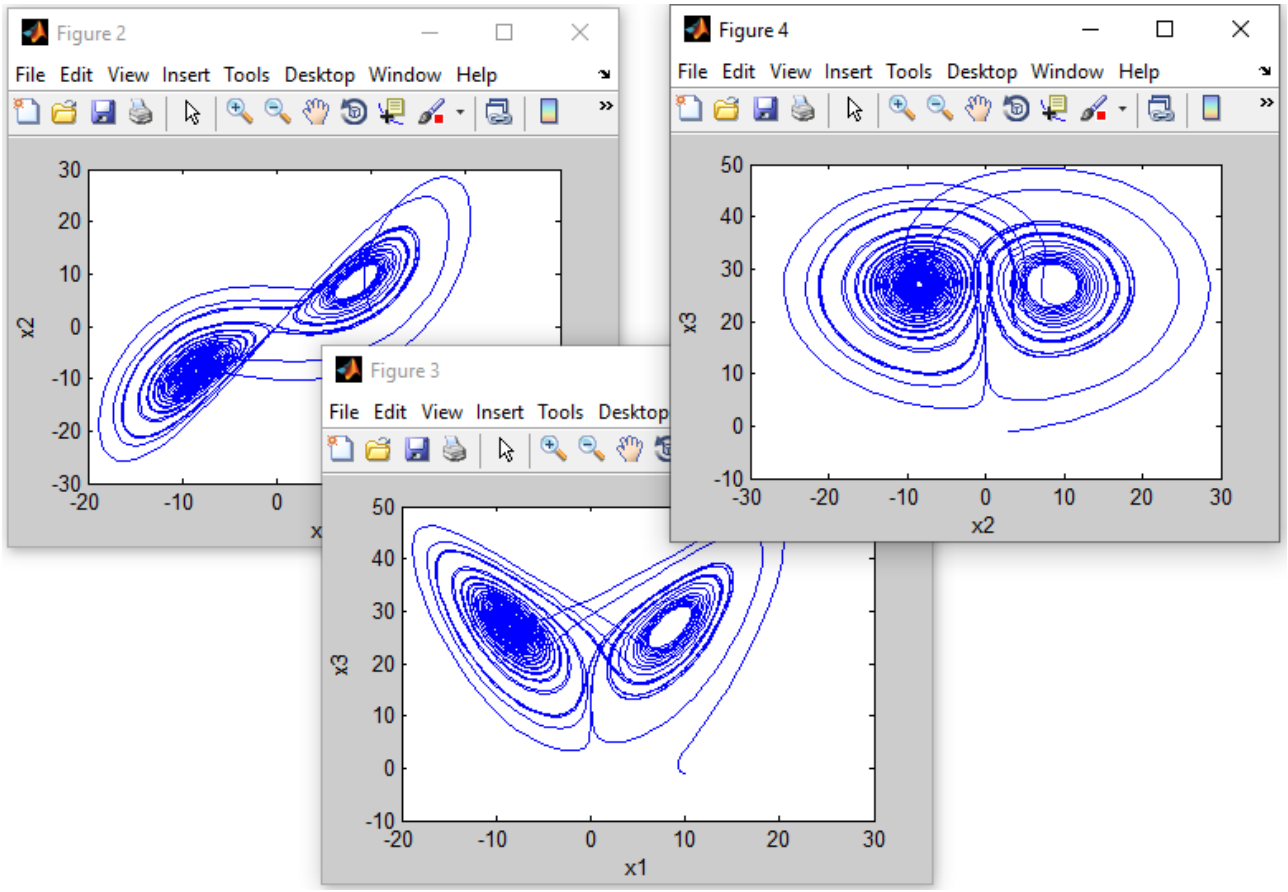
```
>> plot3(x(:, 1), x(:, 2), x(:, 3))
```



фиг. 8.22

Използвайки данните от същия масив, могат да бъдат изчертани и двумерните графики x_1x_2, x_1x_3, x_2x_3, представляващи двумерни проекции на получената тримерна графика (фиг.8.23). Командата, която е необходимо да бъде използвана, е *plot*, а синтаксисът:

```
>> figure, plot(x(:, 1), x(:, 2))
>> figure, plot(x(:, 1), x(:, 3))
>> figure, plot(x(:, 2), x(:, 3))
```



фиг. 8.23

По подобие на представеният начин за симулиране на система, зададена с математичен модел от три уравнения, може да се изгради симулационна схема и за система от по-висок ред.

Задачи за изпълнение:

1. За разглежданата система от три уравнения да се направят експерименти и да се наблюдават графиките при промяна на параметрите ѝ *a*, *b* и *r*.

2. За разглежданата система от три уравнения да се направят експерименти и да се наблюдават графиките при промяна на началните условия на интеграторите.

3. При изчертаване на графиките в *Matlab* да бъдат поставяни надписи по координатните оси.

Лабораторно упражнение 9
Символни изчисления и преобразувания в *Matlab*

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите за символни преобразувания в програмния продукт *Matlab*.

Symbolic Math Toolbox

Matlab предоставя възможности за различни символни пресмятания чрез *Symbolic Math Toolbox*, който позволява работа с аналитични изрази, уравнения и трансформации. За разлика от числените изчисления, символните операции се извършват точно, без апроксимации. Потребителят може да извършва действия като диференциране, интегриране, решаване на алгебрични и диференциални уравнения в аналитичен вид, намиране на граници на функция и др.

Symbolic Math Toolbox е предназначен за извършване на аналитични преобразувания в средата на *Matlab*. Възможностите, които има са:

- ✓ аритметика с променлива точност (vpa – variable precision arithmetic);
- ✓ преобразуване и опростяване на изрази;
- ✓ линейна алгебра – пресмятане на обратни матрици, детерминанти, собствени стойности и т.н.;
- ✓ математичен анализ – диференциране, интегриране, граници, суми, редове на Тейлор;
- ✓ аналитично и числено решаване на алгебрични уравнения;
- ✓ аналитично решаване на диференциални уравнения;
- ✓ интегрални преобразувания на Лаплас, Фурие, Z – преобразувание и обратните им.

Списък с всички функции, включени в *Symbolic Math Toolbox* може да бъде получен чрез командата

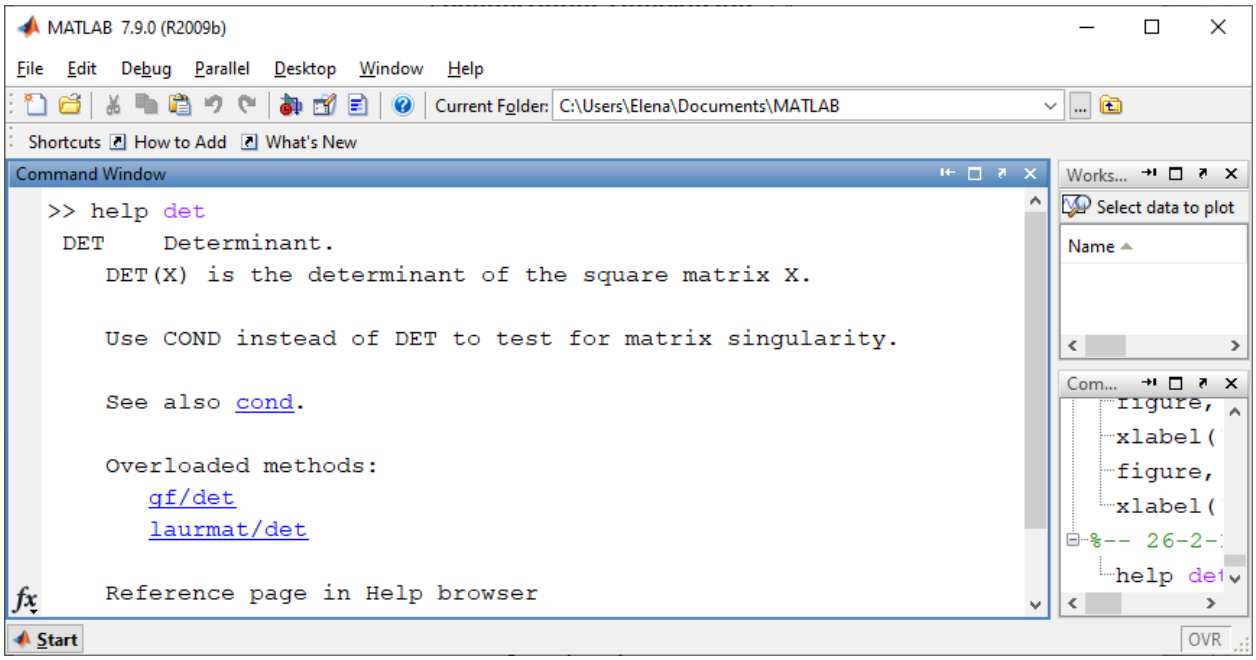
>> help symbolic

За извеждане на информация за дадена команда от Matlab трябва да се има предвид следното:

✓ ако в командния прозорец бъде записано

>> help det

то информацията, която ще се изведе, е за числения вариант на функцията *det* (фиг.9.1).

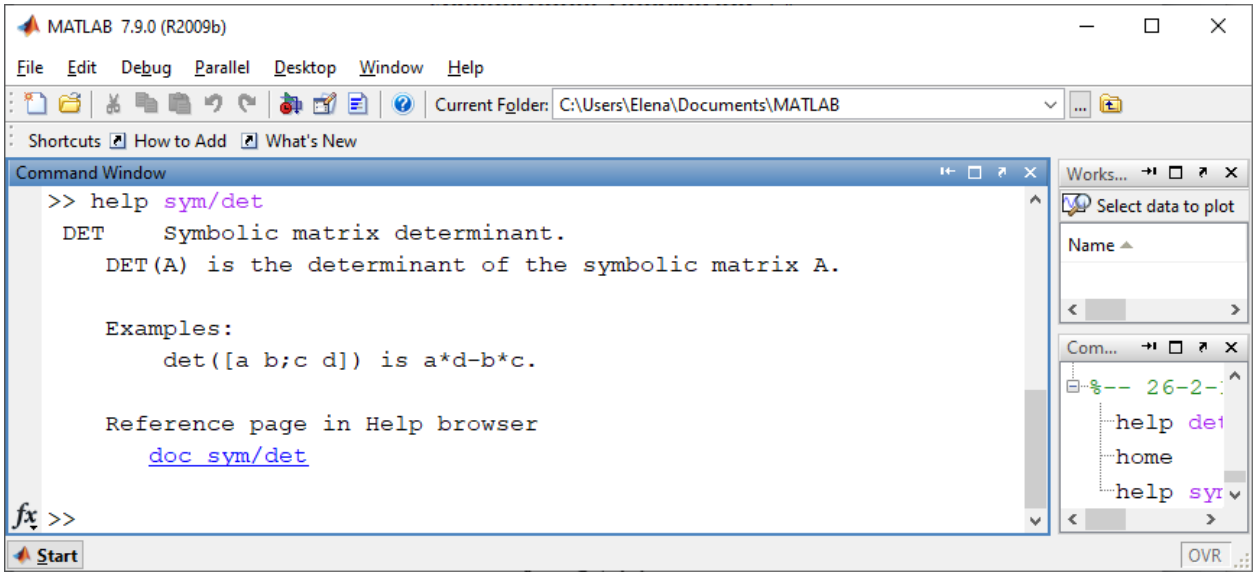


фиг. 9.1

✓ ако в командния прозорец бъде записано

>> help sym/det

то информацията, която ще се получи, е свързана със символния вариант на функцията *det* (фиг.9.2).



фиг. 9.2

Дефиниране на символни променливи и изрази

За да бъде използвана една променлива като символна, то тази променлива трябва да бъде дефинирана като такава. Дефинирането на символни променливи може да стане по два начина:

- ✓ дефиниране на символни променливи чрез команда *sym* – в този случай символната променлива се задава като аргумент на функцията (т.е. в кръгли скоби) и се огражда с апострофи:

$$x = sym('x')$$

Използвайки функцията *sym* могат да бъдат дефинирани не само символни променливи, а и цели изрази, например:

$$p = sym('a*x^2 + b*x + c')$$

- ✓ дефиниране на символни променливи чрез команда *syms* – при използването на тази функция, дефинирането на символните променливи става чрез записването им след командата, без да е необходимо да се поставят в скоби

$$syms\ y$$

При дефиниране на символни променливи чрез командата *syms* могат на един ред да бъдат дефинирани повече от една променливи, като разделянето им една от друга става чрез интервал.

$$syms\ a\ b\ c\ d\ x$$

По подразбиране, в разгледаните случаи дефинираните символни променливи са от тип *real* (реални числа). Ако използваните променливи не са реални числа, а комплексни, то тогава типът им трябва да се укаже явно:

$$z1 = sym('z1',\ 'unreal'),\ z2 = sym('z2',\ 'unreal')$$

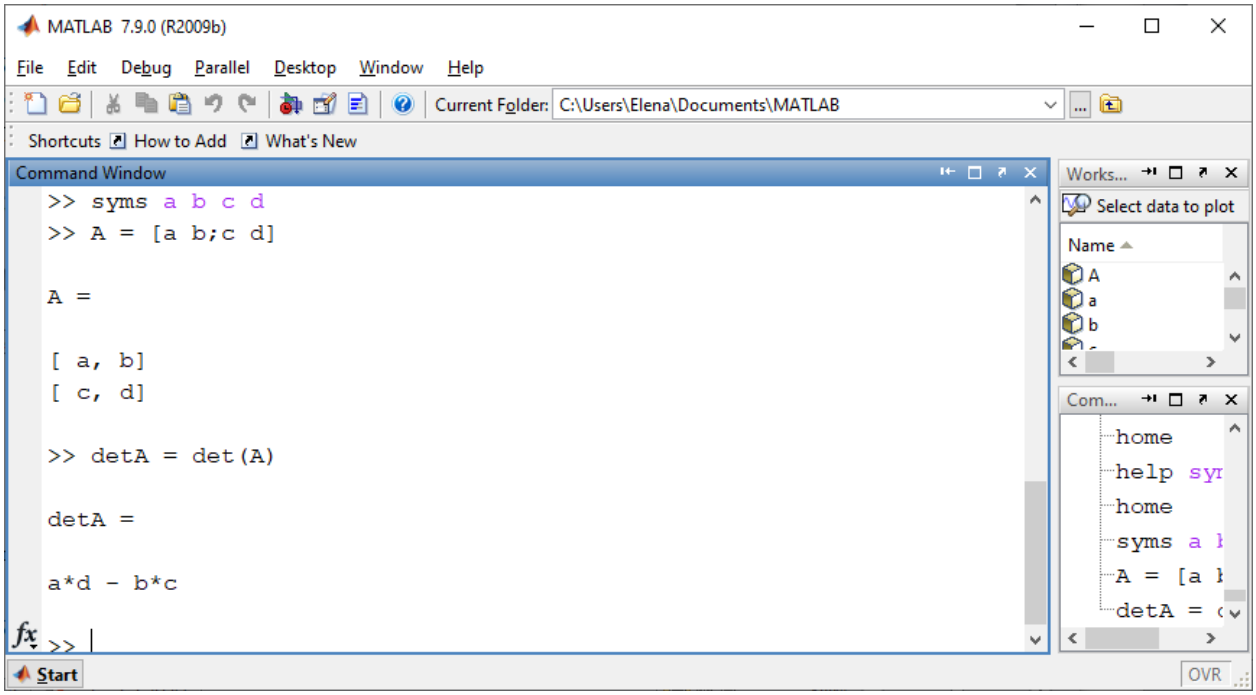
или

$$syms\ z1\ z2\ unreal.$$

🤔 ***Задача***: Да се дефинира матрица A

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

и да се намери нейната детерминанта.



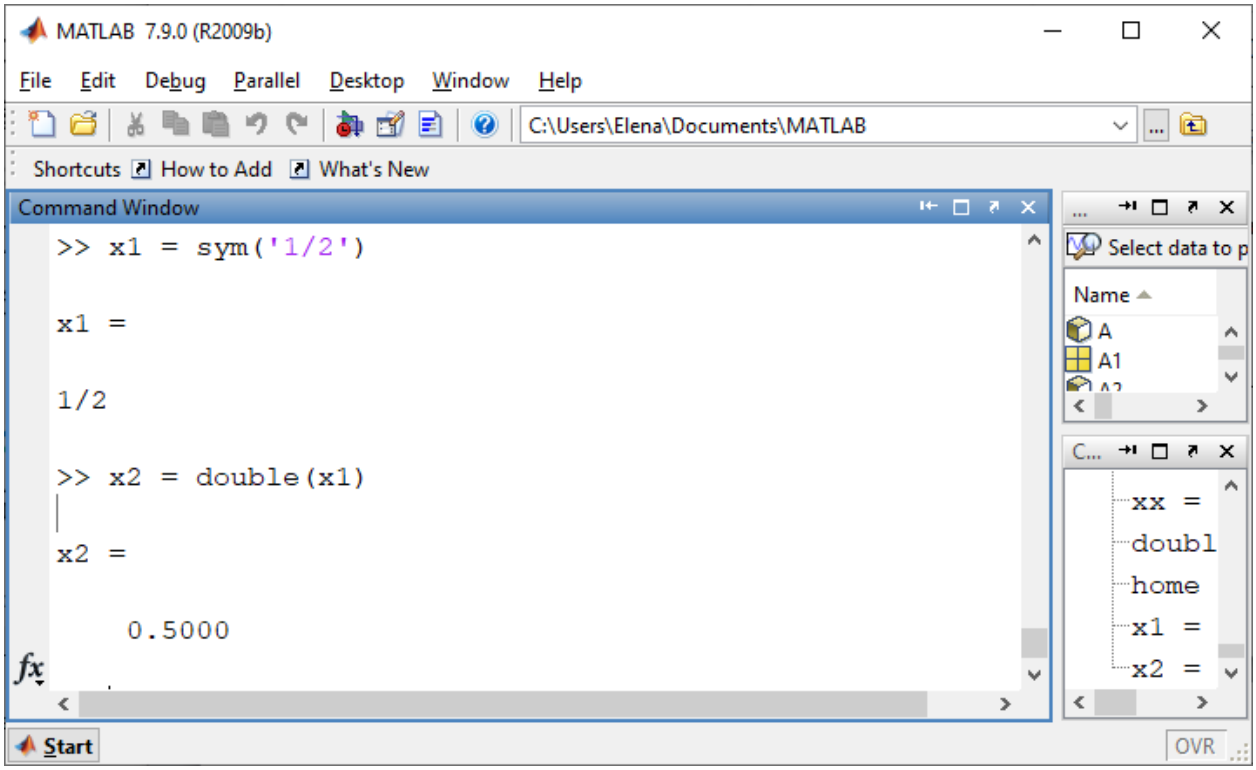
фиг. 9.3

Подразбираща се символна променлива

При използване на дадена функция от *Symbolic Math Toolbox*, е необходимо да се укаже променливата, по отношение на която ще се извършва даденото действие (например интегриране, диференциране и т.н.). В *Matlab* са възможни два варианта за това:

- ✓ явно указване на променливата чрез допълнителен аргумент към дадената функция;
- ✓ променливата не се задава явно, при което системата сама я определя като се спазва правилото, че подразбираща се символна променлива в даден символен израз е „*x*” или най-близката до „*x*” в английската азбука. Ако има две променливи, еднакво отдалечени от „*x*”, за подразбираща се променлива се приема тази, която е след „*x*”, т.е. по-назад в азбуката.

Използвайки символни променливи в даден момент е възможно да има необходимост те да бъдат превърнати в числени. За превръщането на дадена величина от символна в числена (число с плаваща запетая (точка)), се използва функцията *double*.

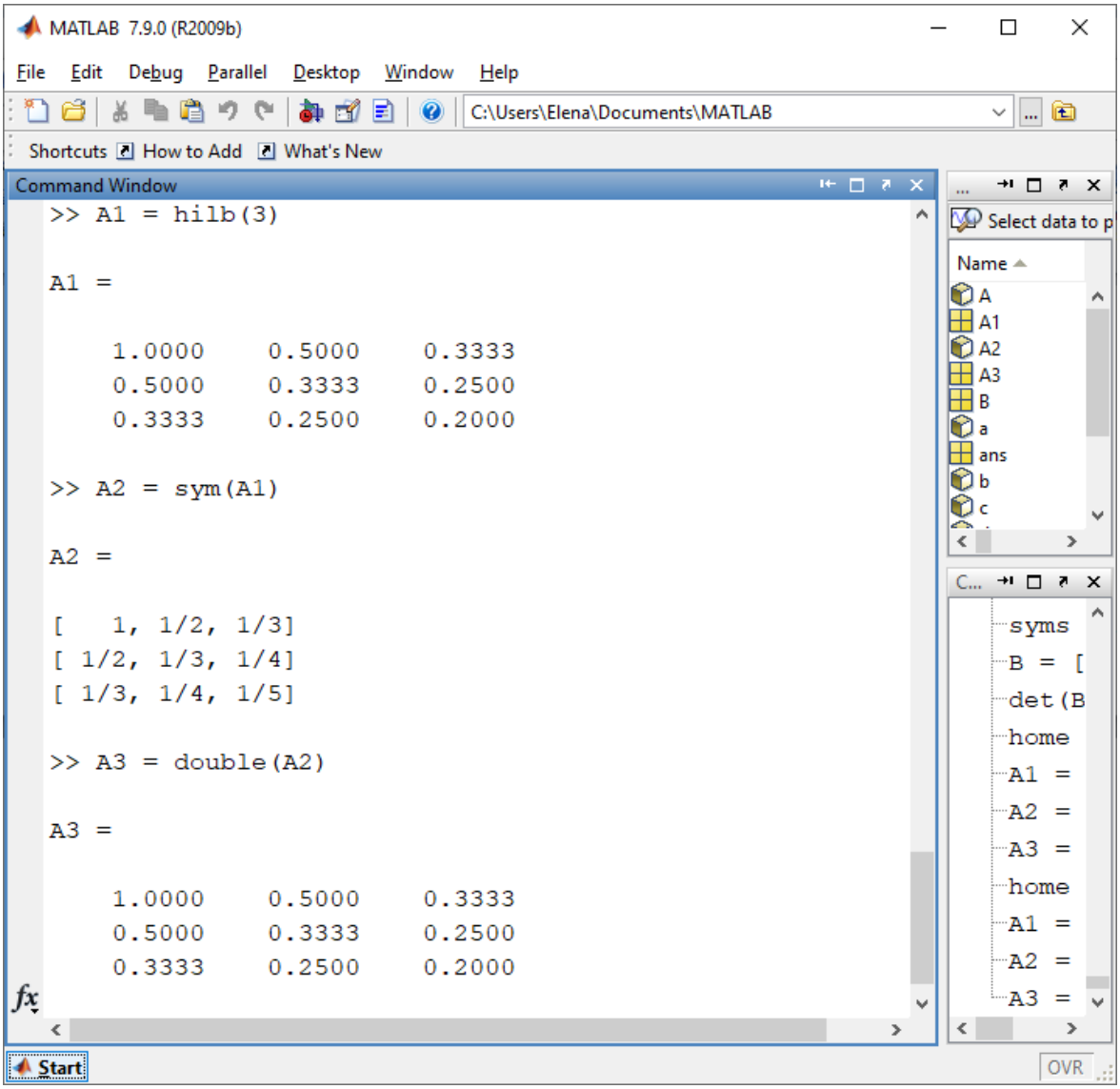


фиг. 9.4

Задача: Да се създаде матрица на Hilbert H_1, която да е с размерност 5 реда и 5 стълба. (Матрицата на Hilbert представлява квадратна матрица *nxn*, чиито елементи се получават по формулата $\frac{1}{i+j-1}$. Функцията, която се използва в *Matlab* е *hilb(n)*.).

Да се получи матрица H_2, която е символният вариант на матрица H_1.

Да се намери матрица H_3, която е преобразуваната в числов вид матрица H_2.



фиг. 9.5

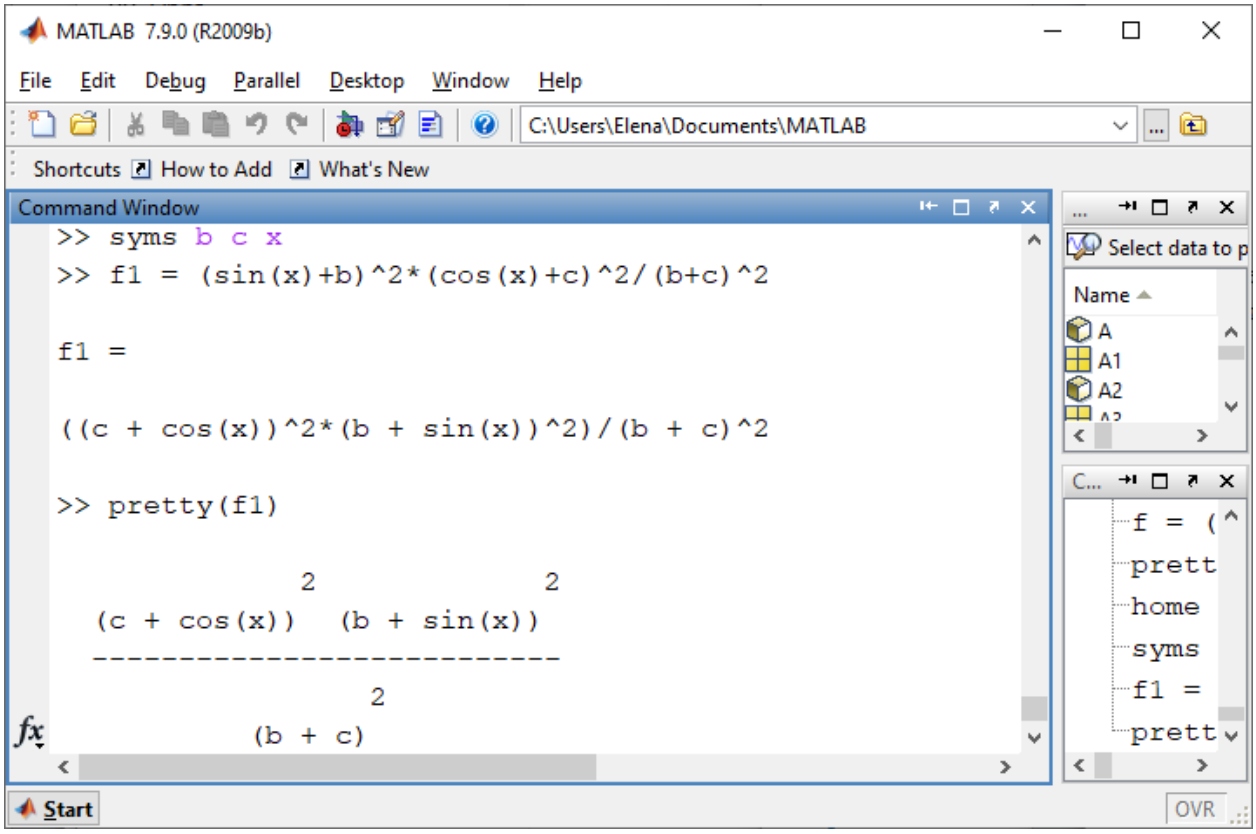
При използване на различните команди за символни преобразувания в *Matlab*, получените резултати се изписват на един ред. Много често това е дълъг израз, при което трудно може да се определят отделните му елементи. За по-четлив вид на изразите може да се използва команда *pretty*. Записът й е следният:

pretty(f)

🤔 ***Задача***: Да се запише символният израз

$$f = \frac{(\sin(x)+b)^2(\cos(x)+c)^2}{(b+c)^2}$$

и да се представи в четлив вид.



фиг. 9.6

Опростяване и преобразуване на символни изрази

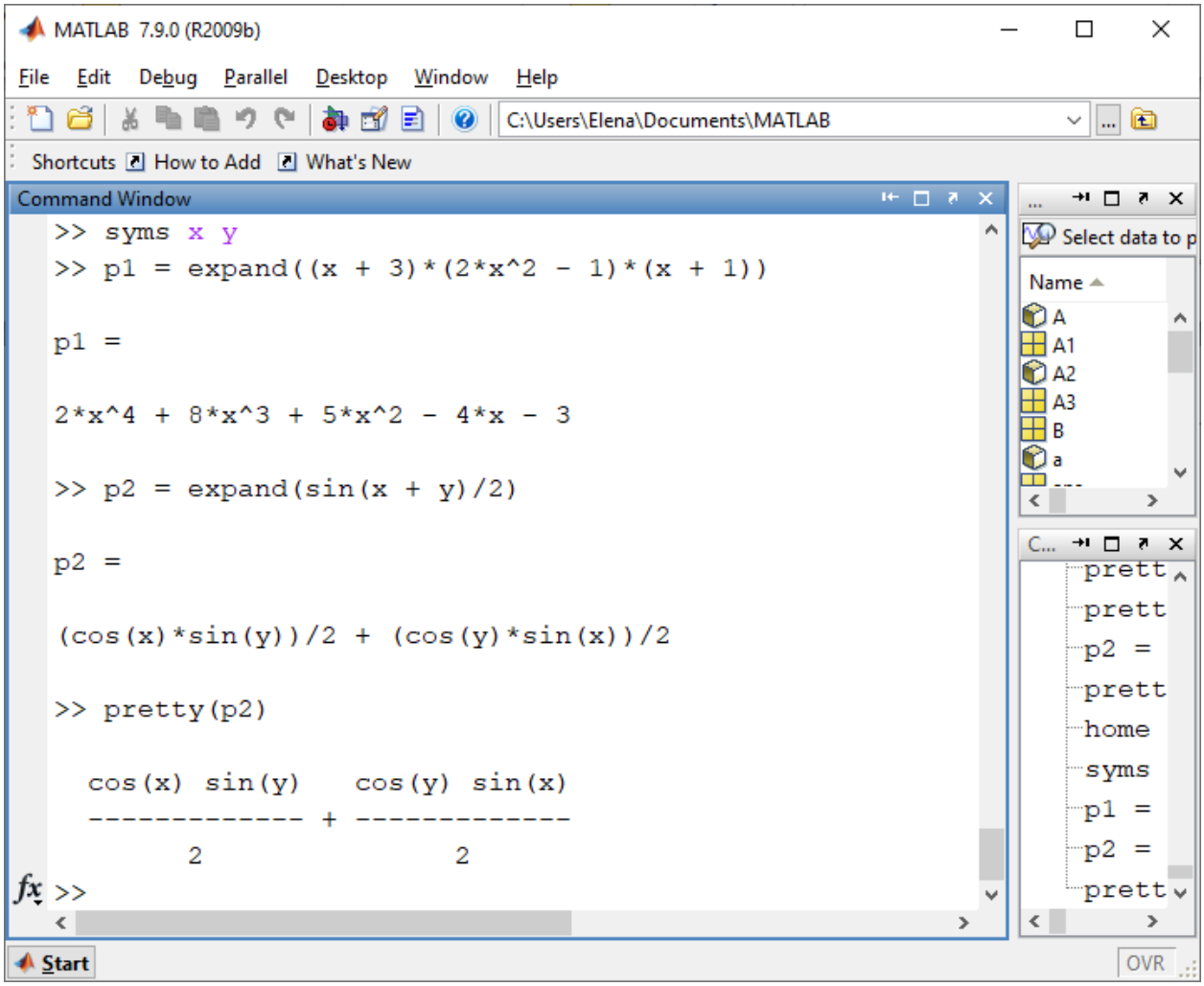
В *Matlab* са включени набор от функции, които дават възможност за опростяване и преобразуване на символни изрази, които позволяват представянето им в по-компактна и удобна за анализ форма. Чрез различни команди могат да бъдат извършвани различни алгебрични действия като разкриване на скоби, опростяване на символни изрази чрез извършване на всички възможни действия в тях, групиране на събираеми по степените на дадена променлива, разлагане на израз във вид на множители и др.

Едно от често извършваните в математиката действия, е разкриването на скоби в даден израз. В *Matlab* за тази цел се използва командата *expand*.

- *expand* – с използването на командата, в дадения израз се разкриват скобите и се извършват математичните действия

Задача: Да се разкрият скобите в изразите:

- $(x+3)(2x^2-1)(x+1)$, резултатът да се запише като променлива p_1;
- $\frac{\sin(x+y)}{2}$, резултатът да се запише като променлива p_2.



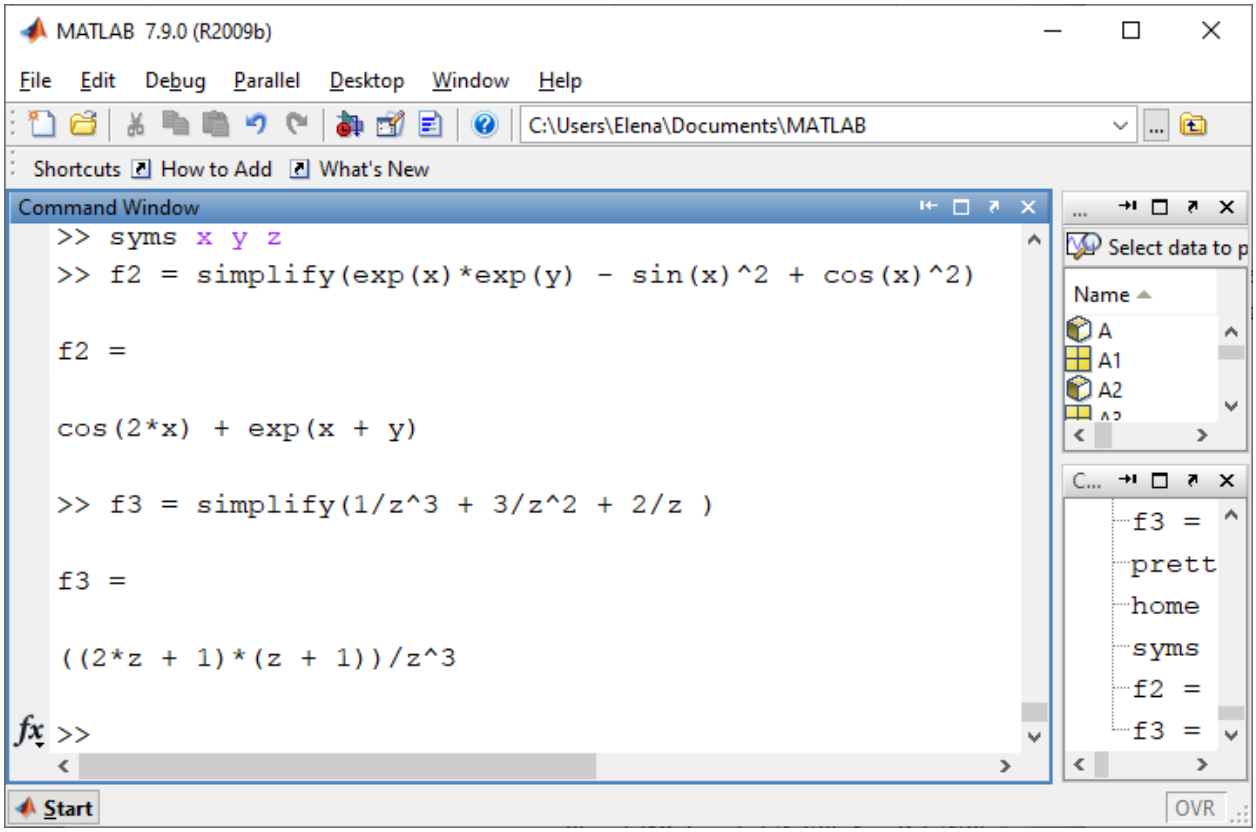
фиг. 9.7

За опростяването на даден израз чрез извършване на различните действия в него, могат да се използват две команди – *simplify* и *simple*.

- *simplify* – опростява символни изрази, в които са включени суми, степени, корени, тригонометрични, експоненциални функции и др.
- *simple* – опростява даденият израз като изпробва всички възможни варианти и връща като резултат решението с най-малко символи.

Задача: Да се извършат действията в изразите:

- $e^x e^y - \sin^2 x + \cos^2 x$, а резултатът да се запише като променлива p_3;
- $\frac{1}{z^3} + \frac{3}{z^2} + \frac{2}{z}$, като резултатът се запише като променлива p_4.



фиг. 9.8

За да видите разликата между командите *simplify* и *simple*, извършете посочените действия без да присвоявате резултатът на променлива. Например:

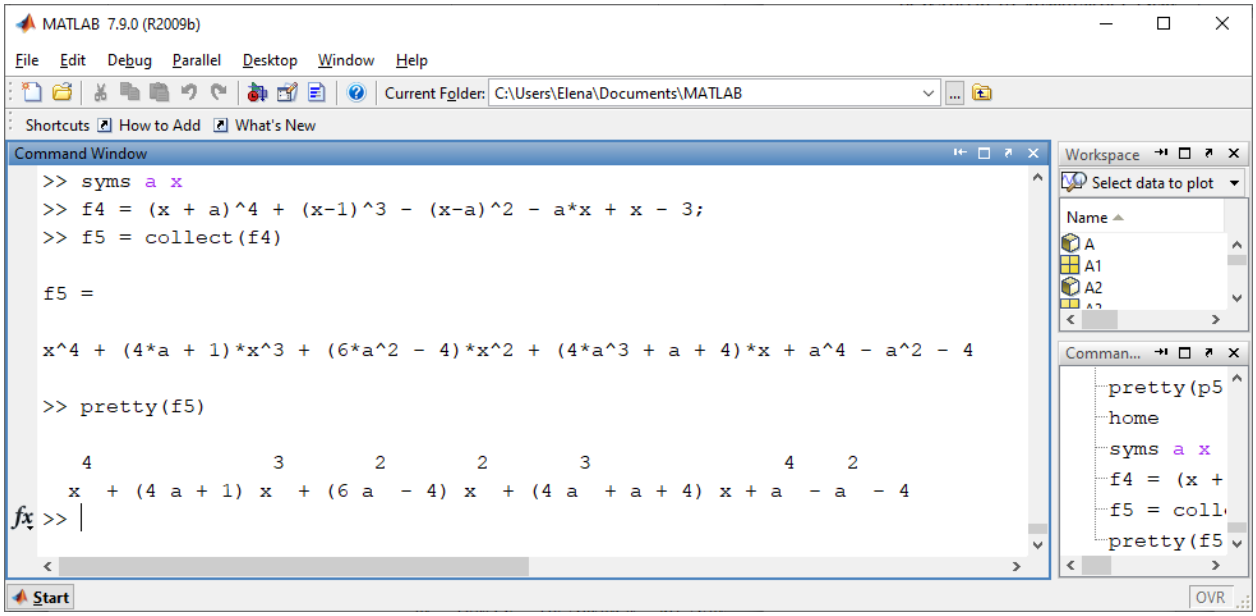
*>> simple(exp(x)*exp(y)-sin(x)^2+cos(x)^2)*

С помощта на команда от *Symbolic Math Toolbox*, в даден символен израз могат да бъдат извършени определените математически действия, след което в резултата събираемите да се представят чрез групиране по степените да дадена символна променлива. Това е възможно чрез използване на командата *collect()*.

- *collect* – групира събираемите по степените на дадена символнапроменлива

Задача: Да се групират по степените на x събираемите в израза:

$$p_5 = (x+a)^4 + (x-1)^3 - (x-a)^2 - ax + x - 3$$



фиг. 9.9

Субституции

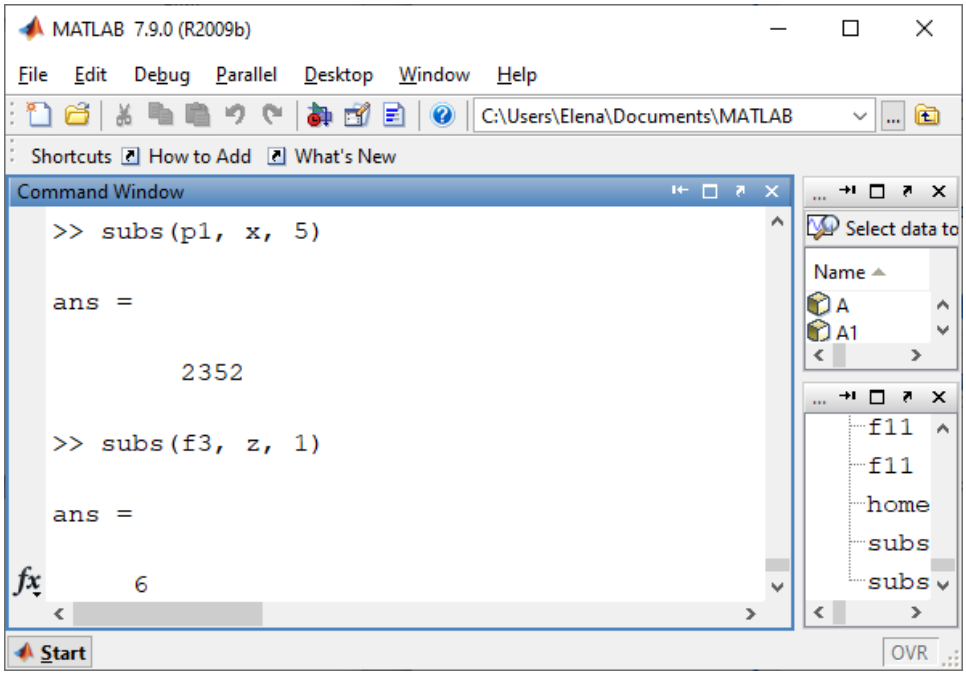
Много често след извършване на определени символни преобразувания е необходимо дадена променлива да бъде заместена с числова стойност. За тази цел се използва командата *subs*.

❖ *subs* – заместване на променливи в даден символен израз

subs(S, old, new),

където *S* е изразът, в който ще се извършва заместването, *old* е променливата, която ще се замества, а *new* е новата стойност на променливата.

Задача: В изразите p_1 и p_4 да се заместят променливите *x* и *z* с произволни стойности.

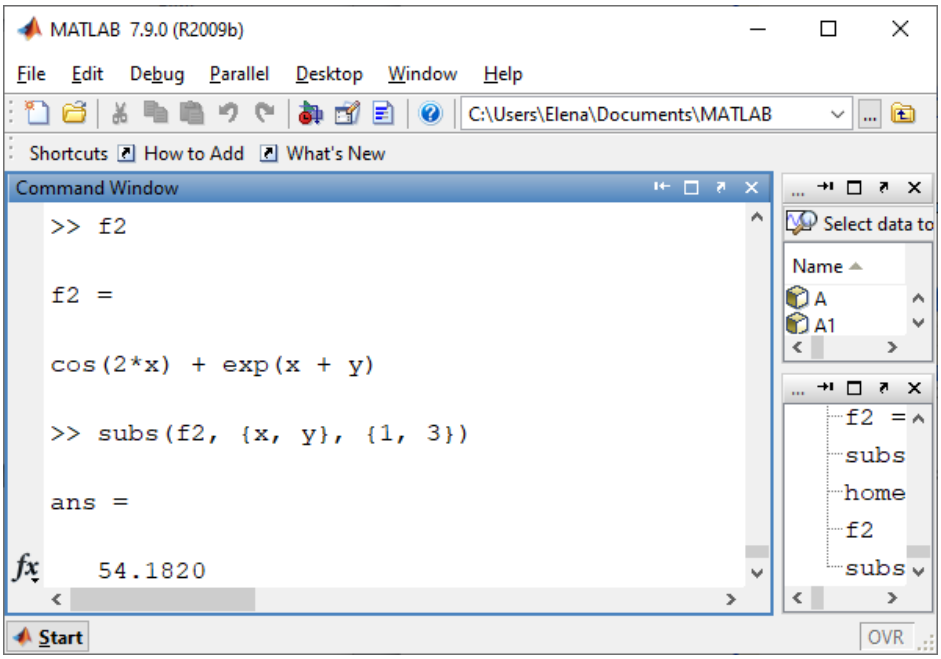


фиг. 9.10

Използвайки функцията subs могат едновременно да бъдат извършени повече от едно замествания.

subs(S, {x1, x2}, {5, 0.1})

Задача: В израза p_3 да се заместят променливите x и y с произволни стойности.

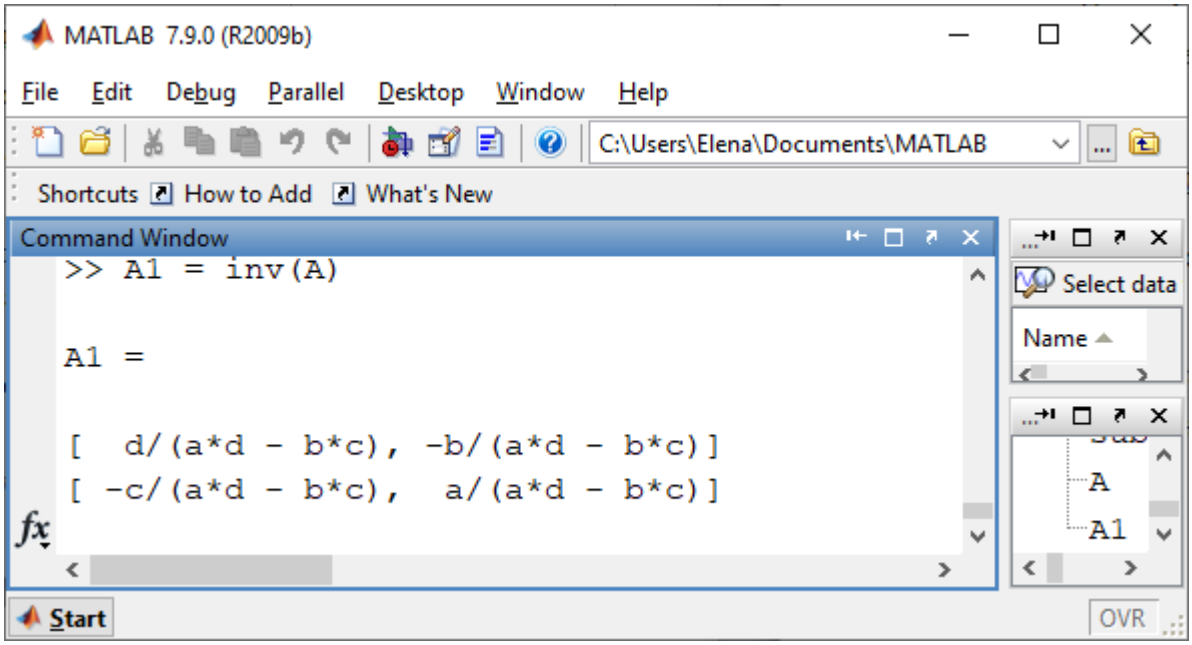


фиг. 9.11

Символни пресмятания от линейната алгебра

- *det(A)* - намира детерминантата на символна матрица A
- *inv(A)* – намира обратната матрица на символна матрица A

Задача: За дефинираната символна матрица $A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ да се намери обратната ѝ матрица.



фиг. 9.12

Математичен анализ

В *Matlab* могат да бъдат решавани и задачи, свързани с намиране на граница на функция, пресмятане на производни, на различни интеграли и др.

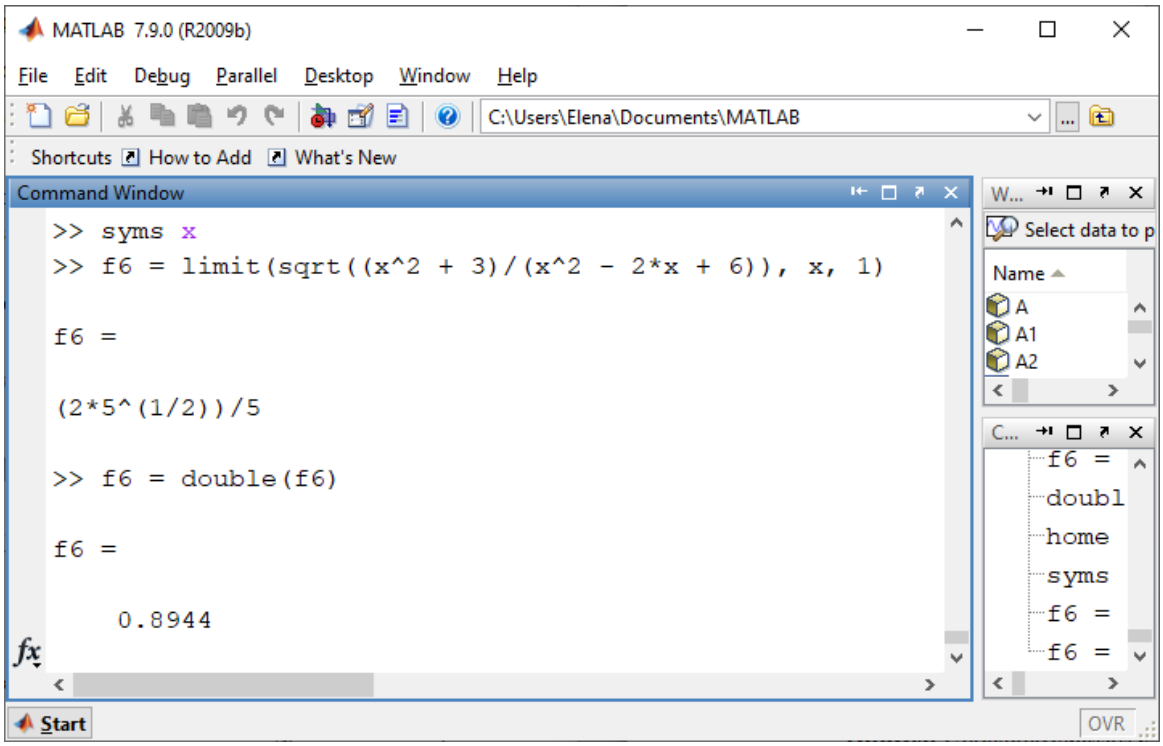
- *limit* – с тази команда се пресмята границата на функция на една променлива. Вариантите при използване на командата, в зависимост от зададените ѝ аргументи са следните:

limit(f) – пресмята граница на функцията *f* при подразбираща се символна променлива, клоняща към *нула*;

limit(f, p) – пресмята граница на функцията *f* при подразбираща се символна променлива, клоняща към *p*;

limit(f, x, p) – пресмята граница на функцията *f* при явно зададена символна променлива *x*, клоняща към *p*;

Задача: Да се намери граница на функцията $f = \sqrt{\frac{x^2+3}{x^2-2x+6}}$, за x клонящо към 1. Резултатът да се представи в числов вид.



фиг. 9.13

- *diff* – с използването на тази команда се пресмята производна на функция спрямо определена променлива. В зависимост от зададените й аргументи, вариантите за използването на функцията са следните:

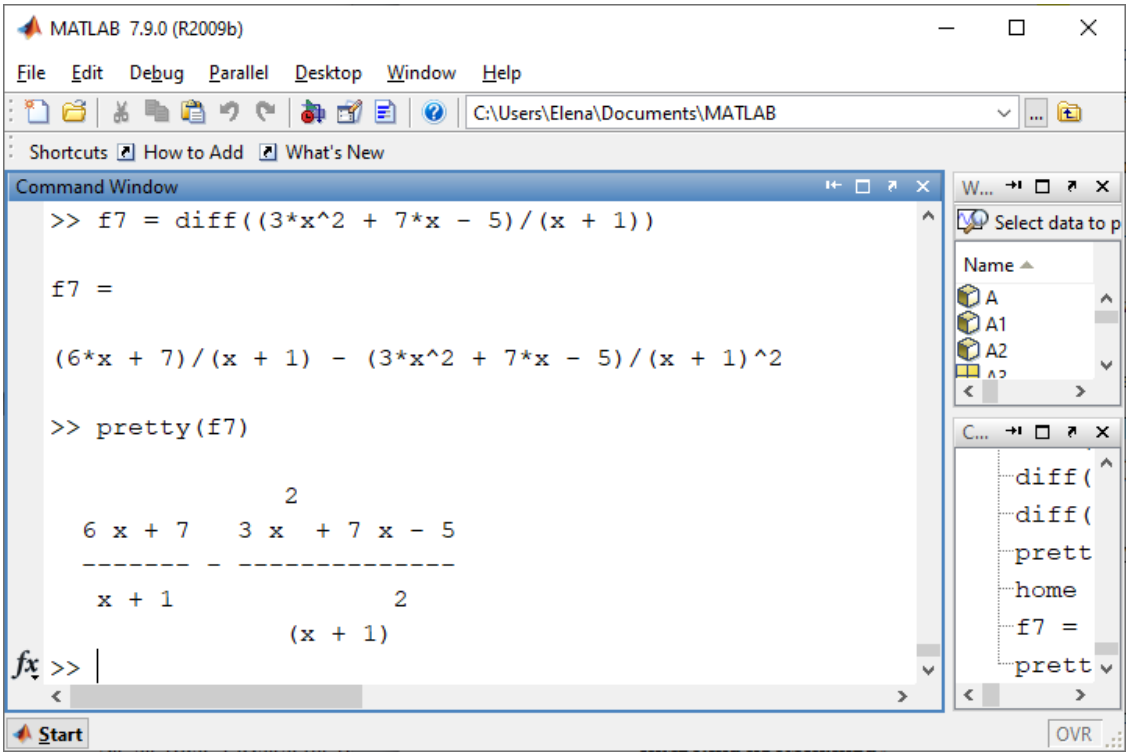
diff(f) – пресмята първа производна на функцията *f* спрямо подразбираща се символна променлива;

diff(f, x) – пресмята първа производна на функцията *f* спрямо явно зададената символна променлива *x*;

diff(f, n), където *n* е цяло число – пресмята *n*-та производна на функцията *f* спрямо подразбираща се символна променлива;

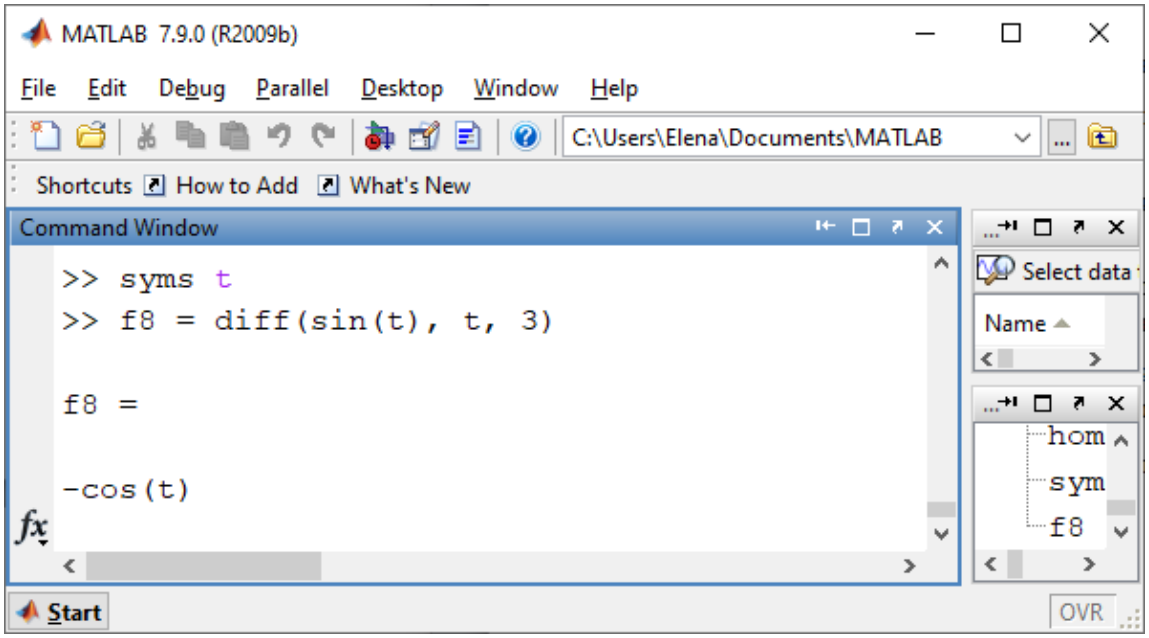
diff(f, x, n), където *n* е цяло число – пресмята *n*-та производна на функцията *f* спрямо явно зададена символна променлива *x*;

Задача: Да се намери първата производна на функцията $f_2 = \frac{3y^2+7y-5}{y+1}$ и резултатът да се представи в четлив вид (функция *pretty*).



фиг. 9.14

Задача: Да се намери третата производна на функцията $y = \sin x$.



фиг. 9.15

❖ *int* – с използването на тази команда се пресмята определен/неопределен интеграл от зададената функция по дадена символна променлива. В зависимост от зададените аргументи, вариантите за използване на функцията са следните:

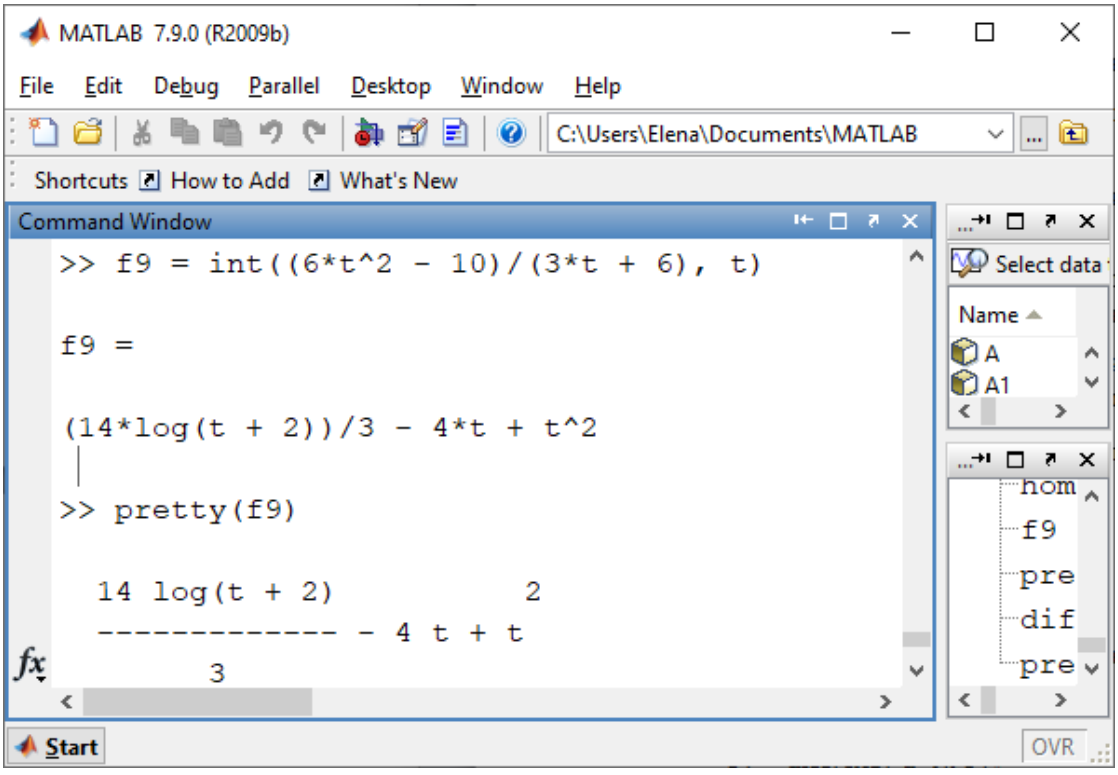
int(f) – пресмята неопределен интеграл от функцията *f* по подразбираща се символна променлива;

int(f, x) – пресмята неопределен интеграл от функцията *f* по явно зададената символна променлива *x*;

int(f, a, b) – пресмята определен интеграл от функцията *f* по подразбираща се символна променлива в границата от *a* до *b*;

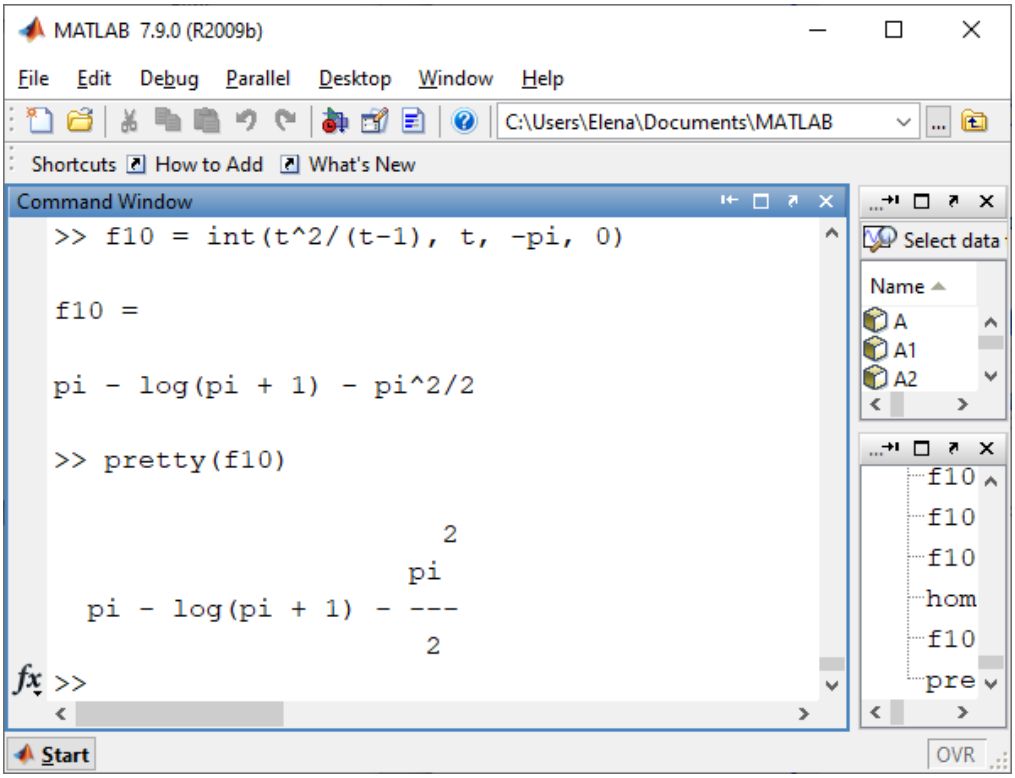
int(f, x, a, b) – пресмята определен интеграл от функцията *f* по явно зададената символна променлива *x* в границата от *a* до *b*;

Задача: Да се пресметне неопределения интеграл $f_3 = \int \frac{6x^2 - 10}{3x + 6} dx$.



фиг. 9.16

Задача: Да се пресметне определения интеграл $f_4 = \int_{-\pi}^{0} \frac{x^2}{x - 1} dx$, а резултатът да се представи в четлив вид.



фиг. 9.17

Решаване на алгебрични уравнения

С помощта на команда от *Symbolic Math Toolbox* има възможност да бъдат намирани решенията на уравнения, както и на системи уравнения.

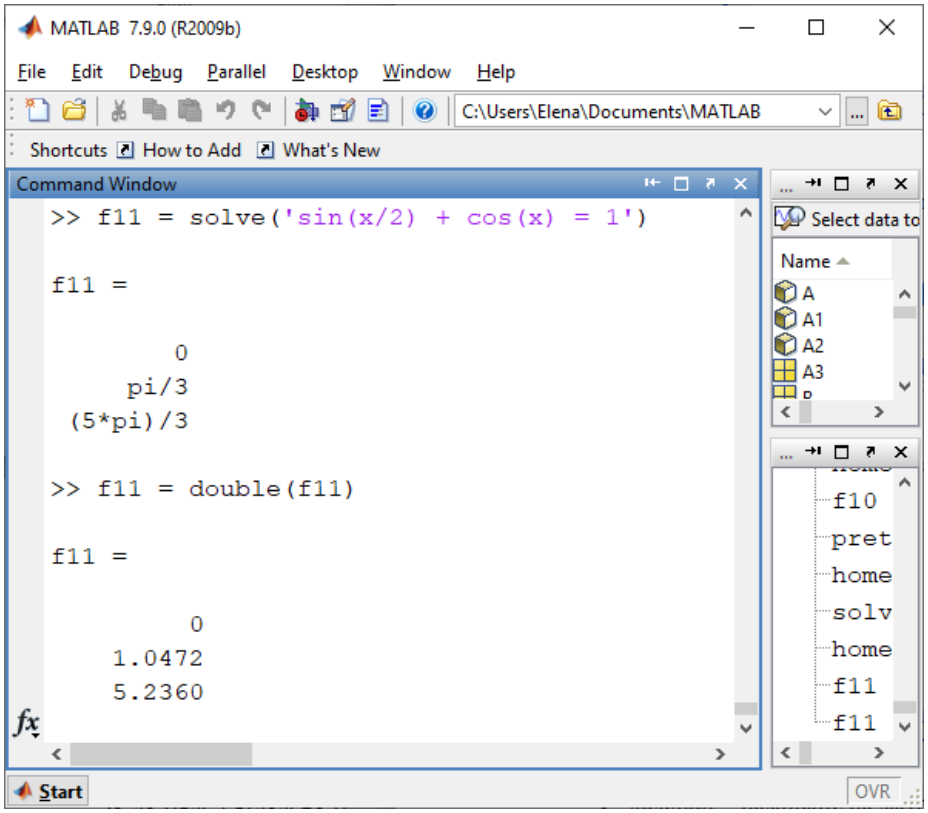
- *solve(f)* – решава уравнението *f = 0* спрямо подразбираща се символна променлива;
- *solve(f, x)* – решава уравнението *f = 0* спрямо явно зададената символна променлива *x*.

!!! Обикновено уравнението *f = 0* се представя само с лявата си част *f*. Ако уравнението е от вида $f(x)=q(x)$, то или трябва да се преобразува във вида $f(x)-q(x)=0$, или се загражда в апострофи solve('f(x)=q(x)').

Задача: Да се реши уравнението:

$$\sin\frac{x}{2}+\cos x=1.$$

Резултатът да се представи в числов вид.



фиг. 9.18

Решаване на система уравнения

Чрез функцията *solve* може да се решава система от две и повече уравнения.

- *solve(f1, f2)* – решава системата уравнения f1 = 0 и f2 = 0 спрямо подразбиращи се символни променливи;
- *solve(f1, f2, x, y)* – решава системата уравнения f1 = 0 и f2 = 0 спрямо явно указаните символни променливи x и y.

Изходящият аргумент е запис от полета, в които се съхраняват съответните корени. Имената на полетата съвпадат с имената на неизвестните:

K = solve(f1, f2, x, y)

Решенията за *x* се съхраняват в полето *K.x*, а тези за *y* в полето *K.y*. За извеждане на резултатите е необходимо да се напишат командите

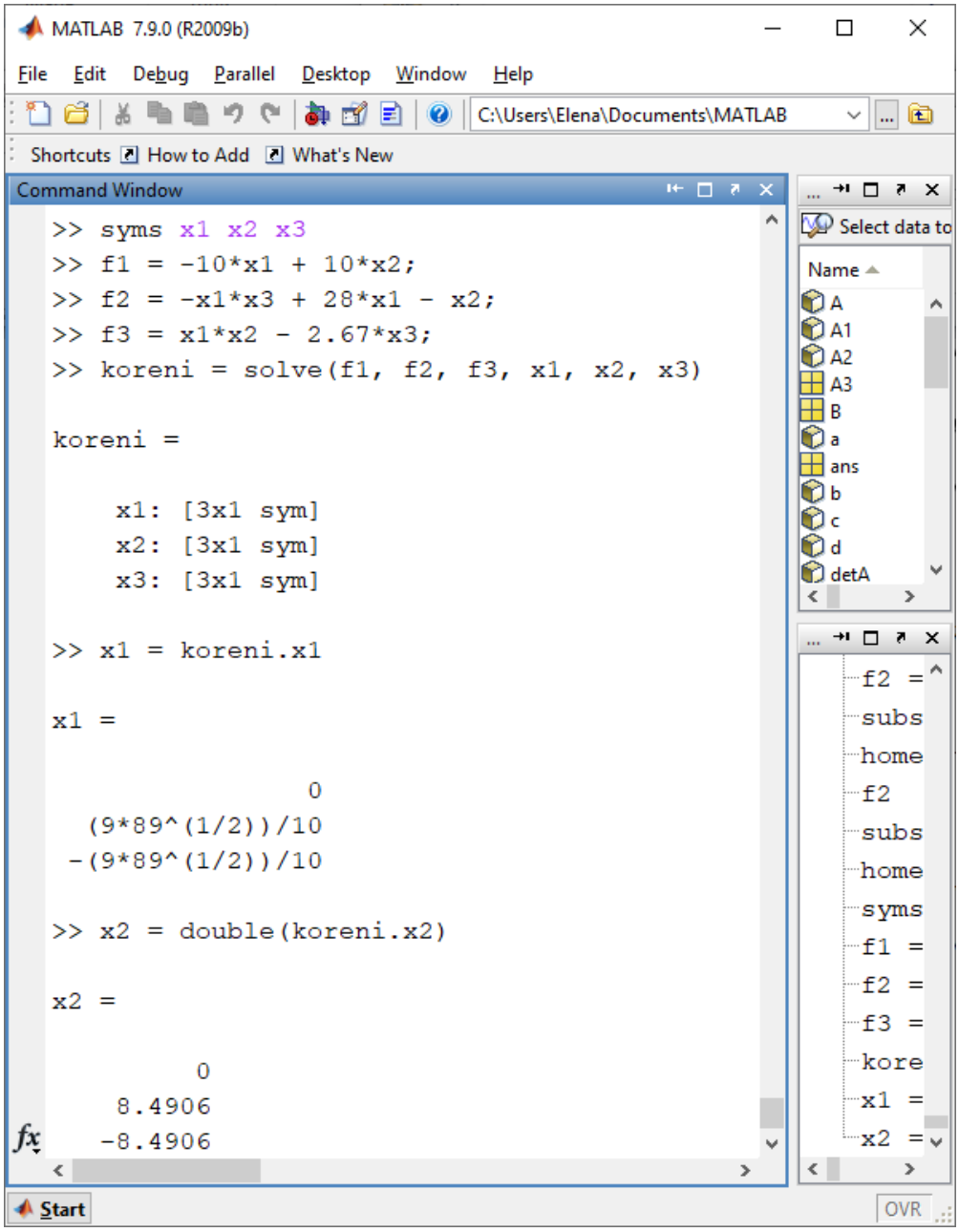
x = K.x и y = K.y

При нелинейни системи може да се получат няколко решения. За да извлечем дадено решение за системата (например първото) е необходимо да се напише командата:

Reshenie1 = [x(1) y(1)] или Reshenie1 = [K.x(1) K.y(1)]

Задача: Да се намери решението на системата и да се представи в числов вид.

$$\begin{aligned} -10x_1 + 10x_2 &= 0 \\ -x_1x_3 + 28x_1 - x_2 &= 0. \\ x_1x_2 - 2.67x_3 &= 0 \end{aligned}$$



фиг. 9.19

Въпроси и задачи за изпълнение:

1. Какви са вариантите за дефиниране на символни променливи в *Matlab*?

2. Да се разкрият скобите на функцията $\cos(x^2+y^2)$, а резултатът да се присвои на променлива $zad2$. В получения резултат да се заместят променливите *x* и *y* със стойности. $x=\pi$, $y=2\pi$.

3. Да се намери граница на функцията $f_2=\dfrac{(\cos(x+a)-\cos(x))}{a}$, при x клонящо към 5.

4. Да се намери първа и втора производна на функцията $f_4=5x^4-3x^3+x^2-3x+10$.

5. Да се пресметне неопределения интеграл $\int(t^2-2t)e^t dt$ и да се направи проверка на резултата с действие диференциране. Резултатът да се представи в четлив вид.

6. Да се пресметне двойния интеграл $\int_0^1\int_0^\pi \dfrac{t^2+2x}{t-1}dxdt$.

7. Да се реши системата уравнения:

$$5x^2-2y=1$$
$$3x+2y=7$$

и решенията да се представят в числов вид.

Лабораторно упражнение 10
Създаване на графичен потребителски интерфейс в средата на *Matlab*

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите за създаване графичен потребителски интерфейс в средата на *Matlab* и разработване на графичен потребителски интерфейс за визуализиране на двумерни и тримерни графики.

Основни сведения за графичен потребителски интерфейс

GUI (Graphical user interface или графичен потребителски интерфейс) позволява интерактивно взаимодействие на потребителя с операционна система или електронно устройство чрез визуална среда, вместо текстови команди.

Елементите на интерфейса (менюта, бутони, списъци и др.), които са предоставени на потребителя за управление, са изпълнени във вид на графични изображения. Най-често елементите на графичния интерфейс се реализират като икони, които подсказват тяхното предназначение и свойства, което помага на потребителите да разберат и усвоят програмите.

Графичният потребителски интерфейс (GUI) дава възможност на потребителите за визуално взаимодействие с програмата. Той е стандарт за изграждане на различни видове съвременни софтуерни приложения, като за целта се използва общ набор от компоненти на потребителския интерфейс. Този общ набор от компоненти позволява на потребителите, след като веднъж вече са използвали графично базисно приложение по - лесно да усвоят и използват всяко следващо. Основен компонент на графичното приложение е стандартният прозорец на Windows. В прозореца се разполагат контроли на GUI, които понякога се наричат интерфейс. Контролите са обекти, които дават възможност за визуализиране на информацията на екрана и за взаимодеиствието с приложението чрез мишка, клавиатура и други средства за въвеждане на данни.

Създаване на GUI с помощта на графичното приложение guide

Освен по чисто програмен път, графичният интерфейс на дадено приложение може да бъде създаден и с помощта на графичната среда *guide*. Тази среда облекчава и ускорява първия етап – създаването на графичния прозорец и разполагането на различни обекти върху него. *Guide* генерира автоматично основната функция, част от подфункциите и заглавните редове на *Callback*-функциите, обработващи отделните събития. За разлика от програмния подход, тук обменът на информация между *Callback*-функциите се осъществява чрез входните им аргументи.

Описание на диалоговата среда guide

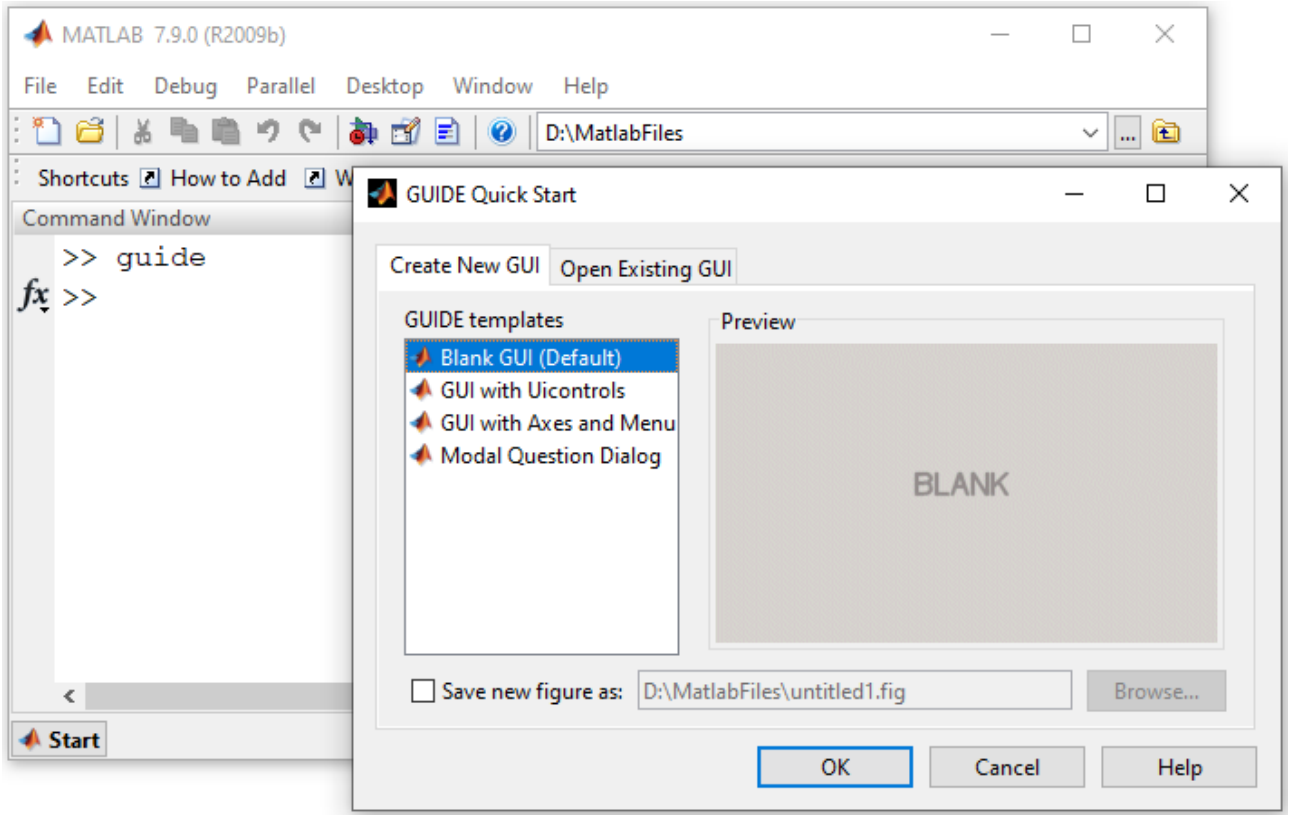
Приложението *guide* се стартира чрез въвеждане на името му в командния ред на *Matlab*:

>> guide

В резултат на това се появава прозорецът, представен на (фиг.10.1). Той съдържа две страници:

- *Create New GUI* – създаване на нов графичен потребителски интерфейс. Служи за избор на шаблон на интерфейса;

- *Open Existing GUI* – отваряне на съществуващ GUI.



фиг. 10.1

При създаване на нов GUI е препоръчително да се използва първия от предлаганите четири стандартни шаблона (*Blank GUI (Default)*). След натискане на бутона OK се появява основния прозорец, който служи като работна среда или графичен редактор на guide (фиг.2).

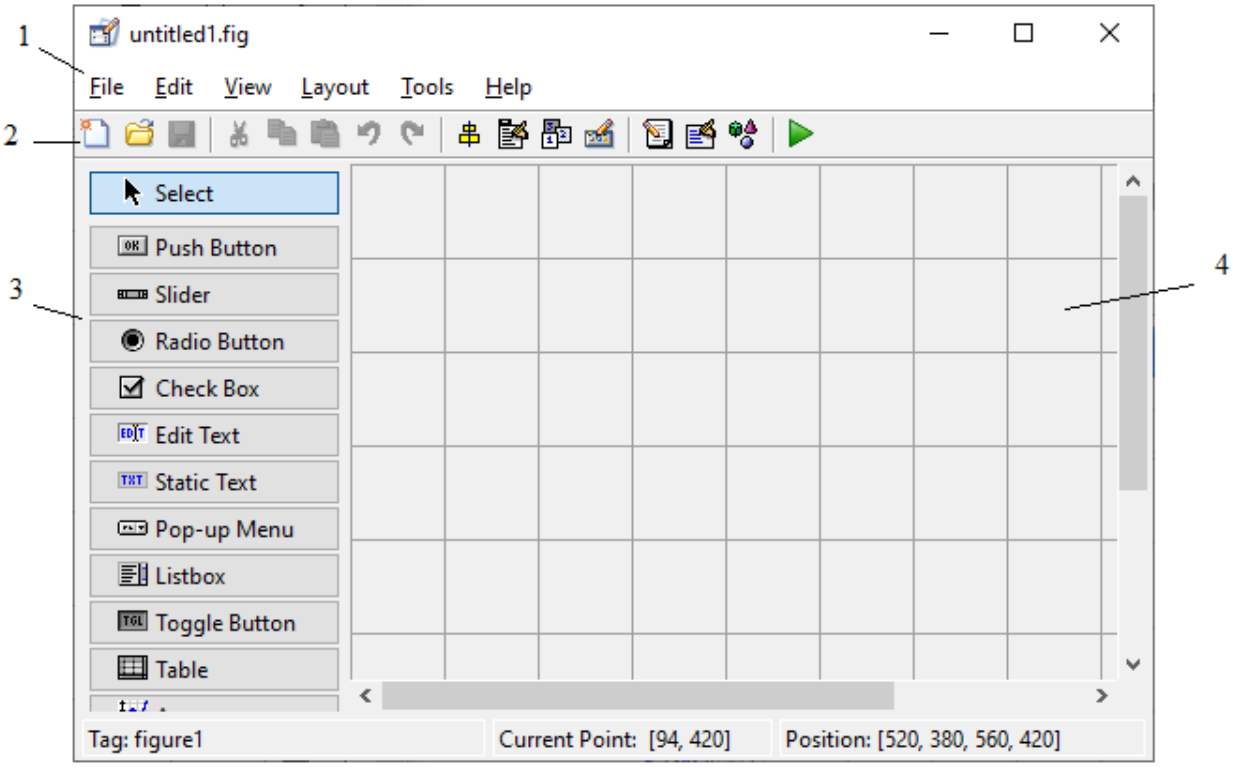
В прозореца на графичния редактор (фиг.10.2) са разположени :

- Стандартна лента с менюта (1);

- Хоризонтална лента с управляващи бутони (2);

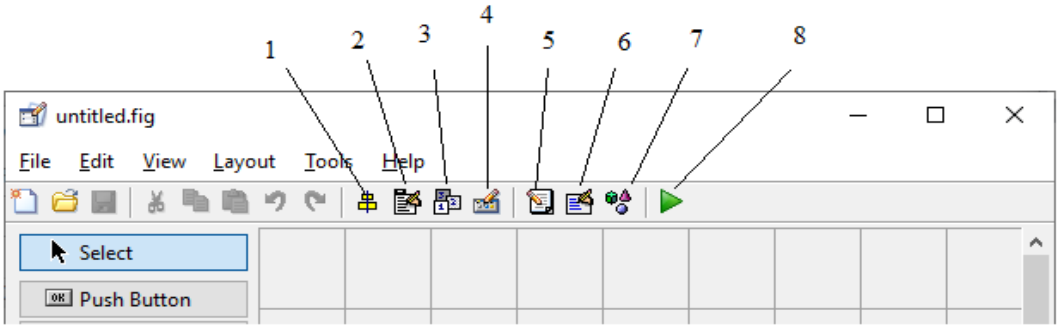
- Панел с основните елементи на графичния интерфейс (3);

- Празен графичен прозорец на бъдещия интерфейс (4);



фиг. 10.2

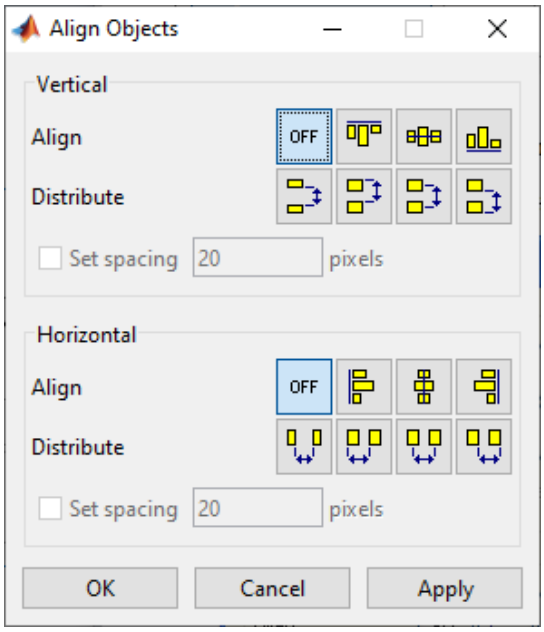
В лентата с управляващите бутони освен често срещаните, са включени и бутони, които са специфични за приложението *guide* (фиг.10.3):



фиг. 10.3

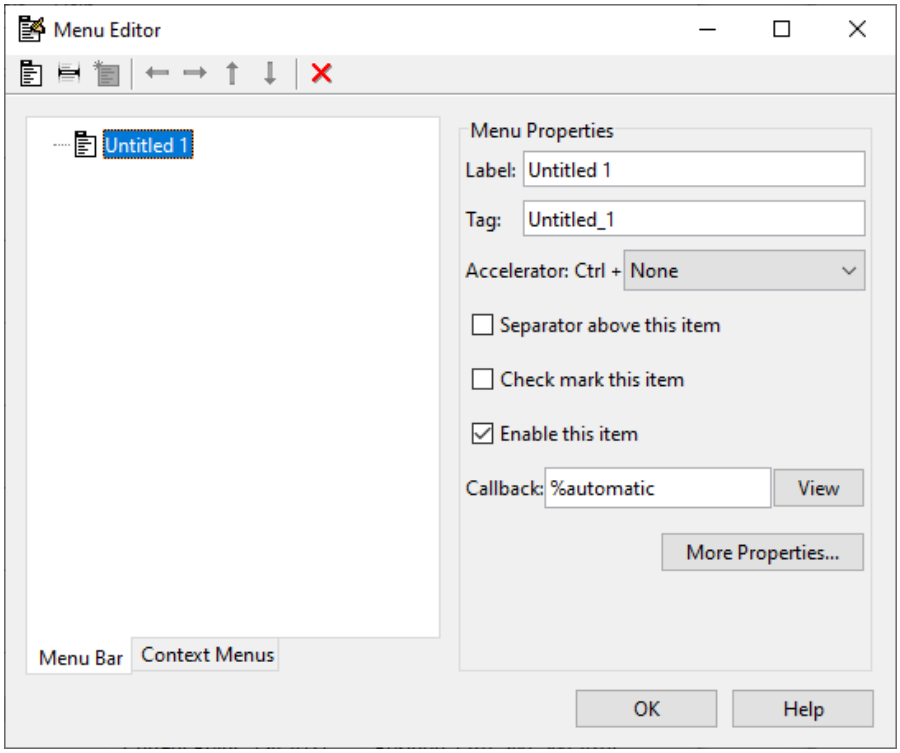
Значението на специфичните бутони е следното:

1- подравняване на графичните обекти (фиг.10.4);



фиг. 104

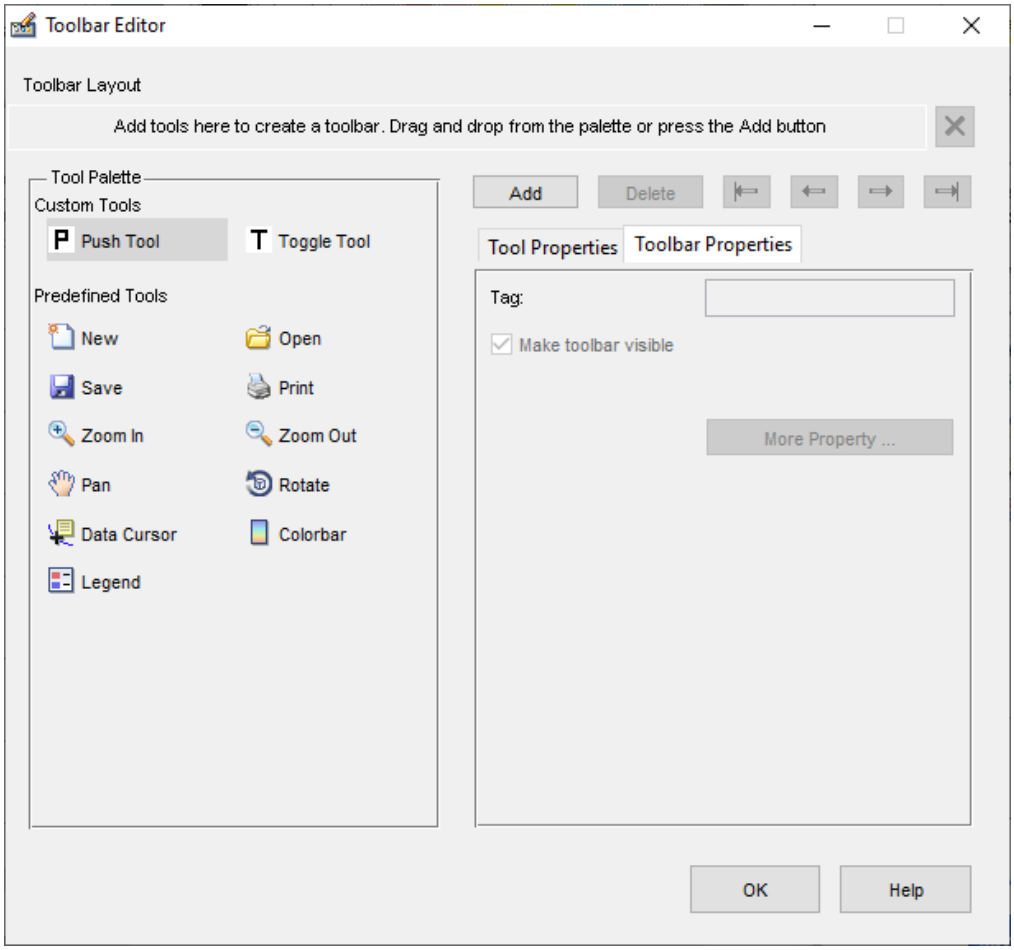
2- извикване редактора на менютата (фиг.10.5);



фиг. 10.5

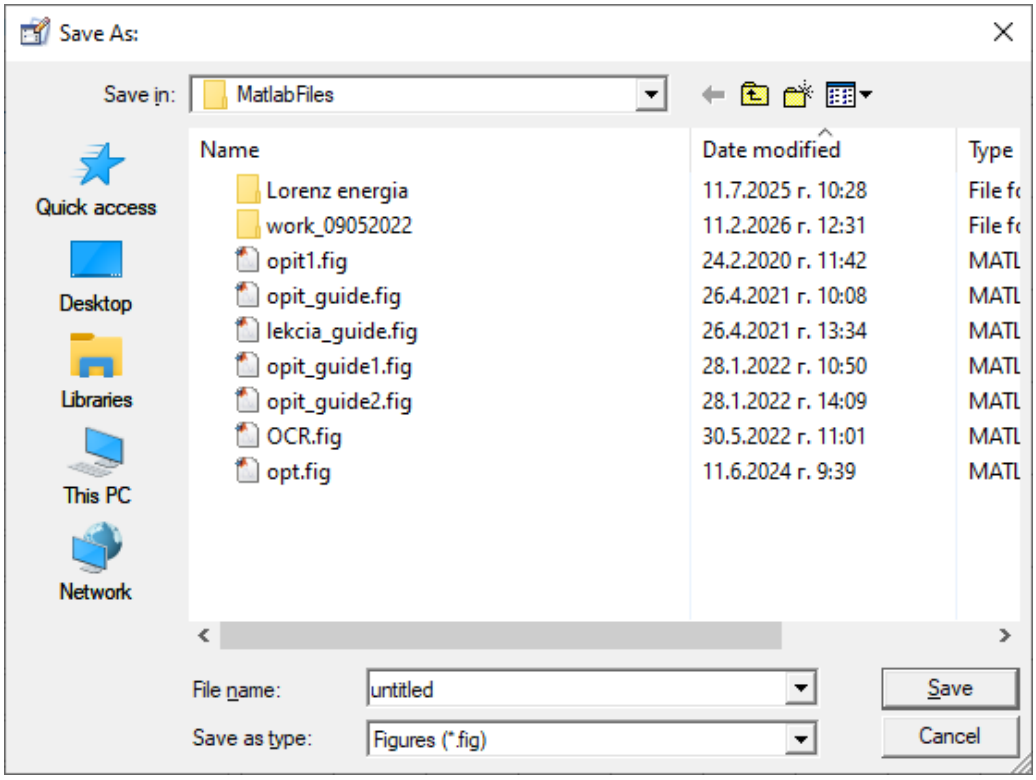
3- редактор за определяне на реда на обхождане на графичните обекти с клавиша TAB;

4 – отваряне на прозорец за Toolbar редактора (фиг.10.6);



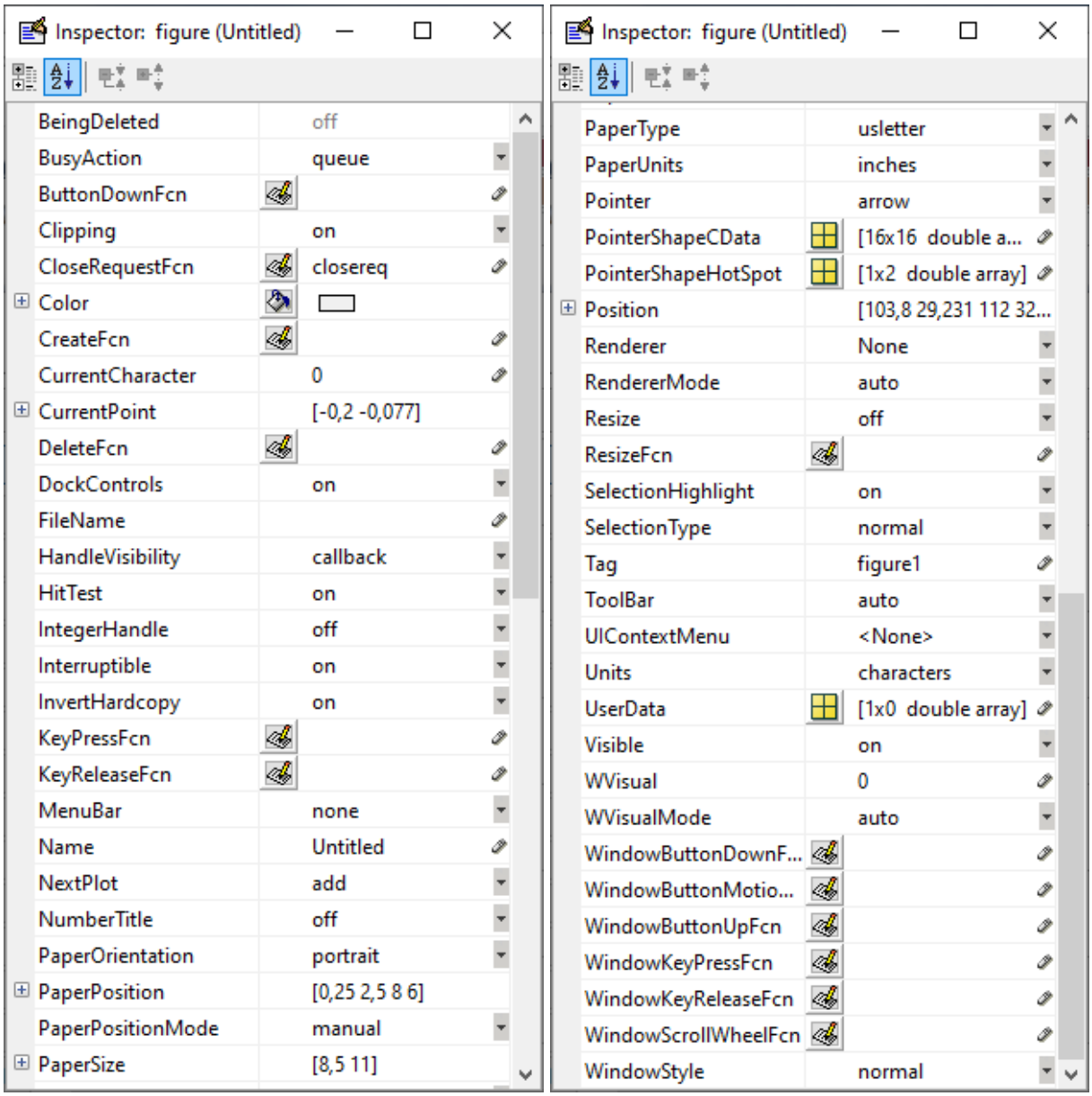
фиг. 10.6

5- отваряне редактора на m-файлове (фиг.10.7);



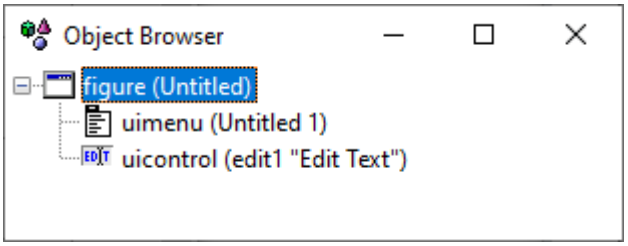
фиг. 10.7

6- извикване на инспектора на свойства (фиг.10.8);



фиг. 10.8

7- обхождане на обектите (фиг.10.9);



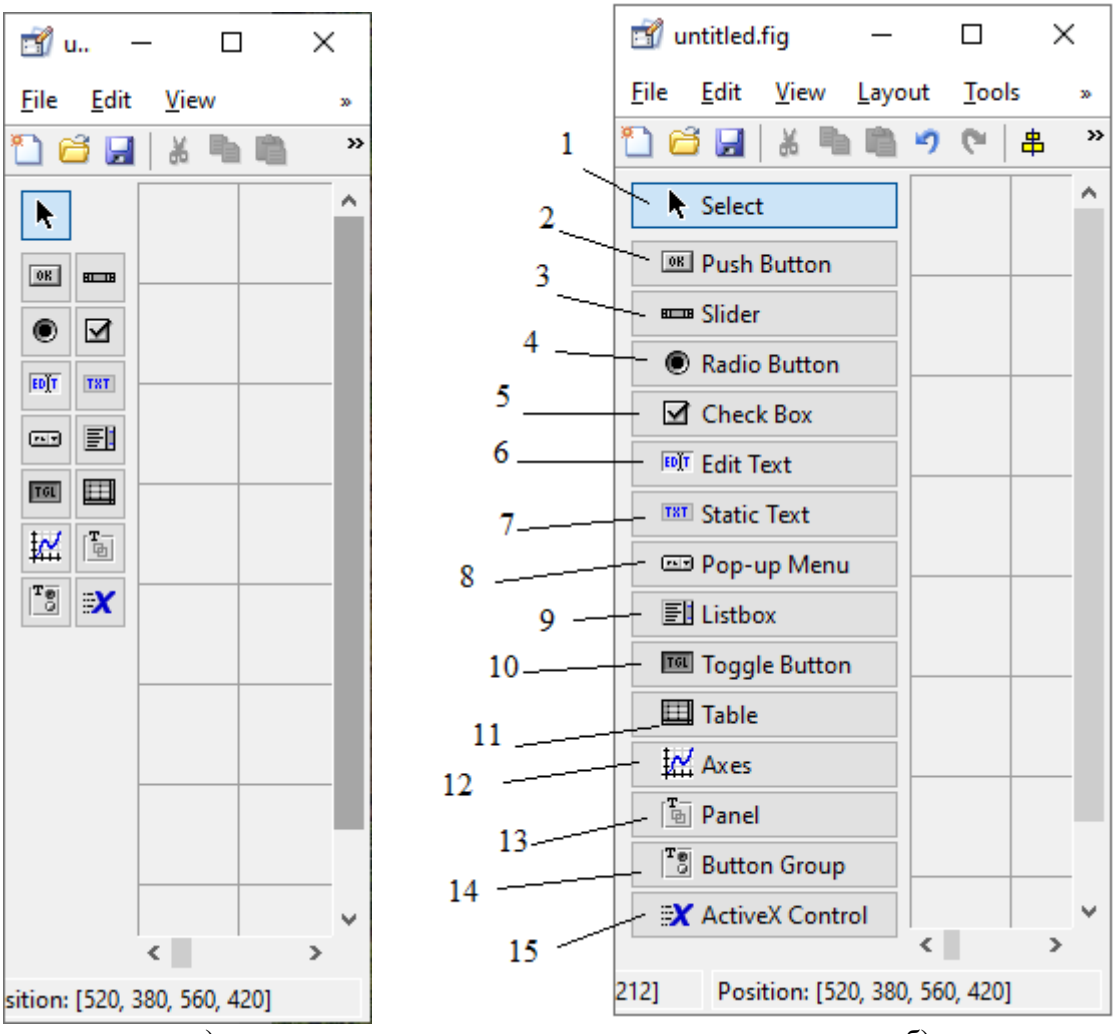
фиг. 10.9

8- стартиране на създавания графичен интерфейс след поредните корекции.

Последният бутон се използва най-често.

Върху вертикалния панел (фиг.10.10) са разположени бутоните на всички графични обекти, които с помощта на мишката можем да разполагаме на произволно място в графичния прозорец.

Обикновено бутоните са визуализирани по начина, показан на фиг. фиг.10.10а. При необходимост е възможно върху бутоните на отделните графични обекти да се изписват техните имена (фиг.10.10б), което може да бъде направено като от менюто *File* се избира позицията *Preferences* и се поставя отметка пред съответната опция (*Guide Preferences -> Show names in component palette*).



а) б)

фиг. 10.10

1. Избор на режим за маркиране на обекти (*Select*);
2. Обикновен бутон (*Push Button*);
3. Лента с плъзгач (*Slider*);
4. Радио бутон (*Radio Button*);
5. Флаг (*Check Box*);

6. Поле за въвеждане на данни (*Edit Text*);
7. Поле за въвеждане на статичен текст (*Static Text*);
8. Падащо меню (*Pop-up Menu*);
9. Поле за въвеждане на текст (*Listbox*)
10. Бутон-превключвател (*Toggle Button*);
11. Таблица (*Table*);
12. Координатна ос (*Axes*);
13. Панел (*Panel*);
14. Панел за група бутони (*Button Group*);
15. ActiveX Control;

Разполагането на графичните обекти върху прозореца се извършва с помощта на мишката по следните два начина:

1. Чрез техника „влачи и пусни” (drag and drop) – натиска се левия бутон на мишката върху иконата на обекта и се влачи до избраното място върху графичния прозорец. Този метод има един недостатък – почти винаги след създаването на графичния елемент се налага да се променят неговите размери с мишката.
2. Чрез начертаване на контурите на обекта върху избраното място. За целта избираме бутона на обекта като кликваме с мишката върху него, след което позиционираме курсора (последният приема формата на кръст) на подходящо място, и при натиснат бутон на мишката начертаваме правоъгълния контур на обект. Този метод е по-бърз, защото едновременно със създаването на обекта се определя неговото място и размери.

Размерите на обектите могат да бъдат променяни с мишката по всяко време чрез разтегляне или свиване на контурите им. Изтриването на един обект или група обекти става след маркирането им и натискане на бутона *Delete*.

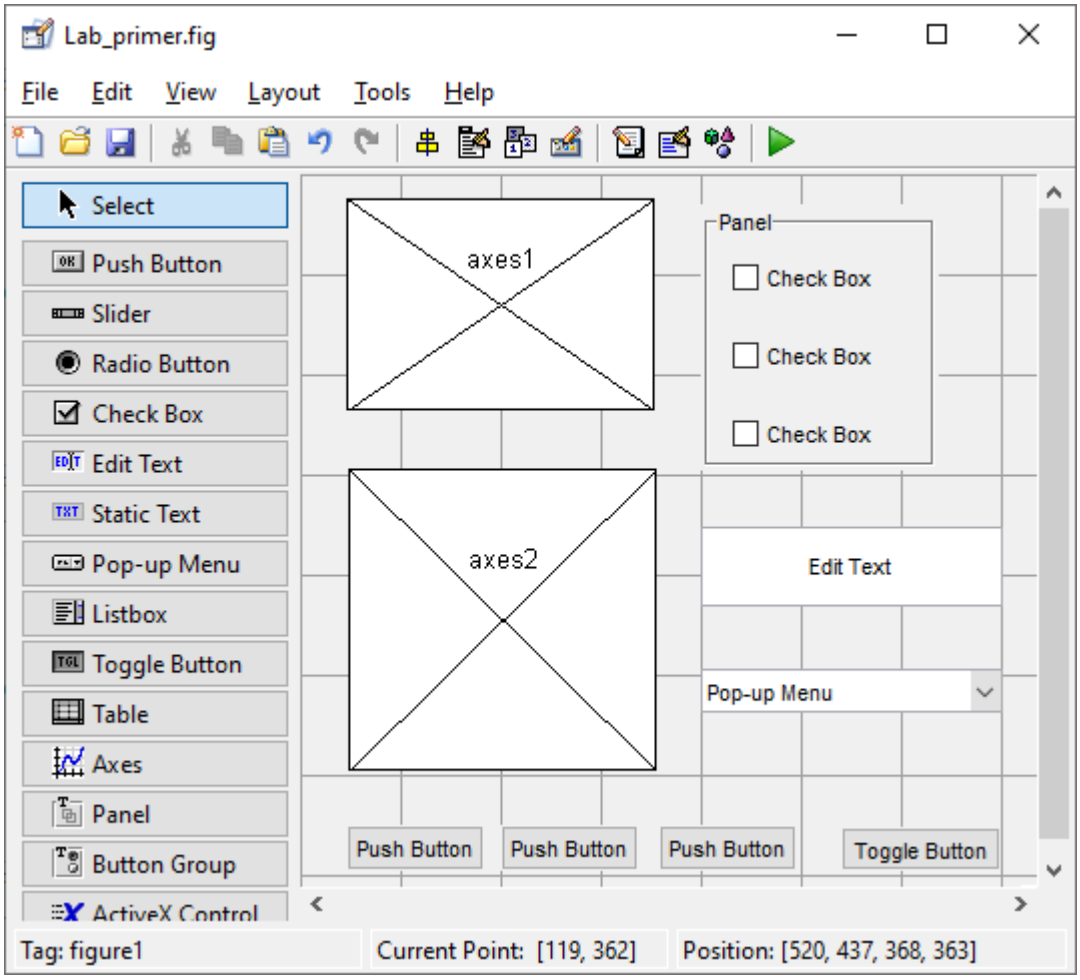
Създаваният чрез влачене графичен елемент „плъзгач” винаги се получава вертикален. За да се създаде хоризонтален плъзгач трябва да се деформират контурите му така, че широчината да стане по-голяма от височината.

От естетическа гледна точка е добре обектите да бъдат подравнявани. Това може да стане по два начина:

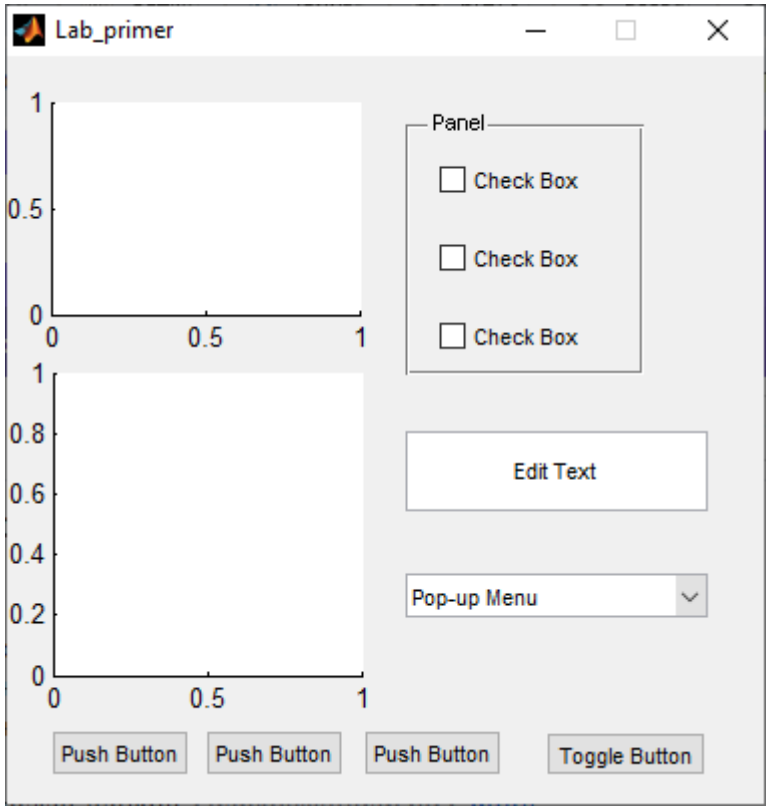
1. Подравняване на ръка чрез преместване на обектите с помощта на клавишите със стрелки;
2. Използване на специален инструмент за точно подравняване (фиг.4). Той може да се извика с натискане на бутона 1 (фиг.3) или чрез менюто – *Tools /Align Object*.

Естествено, всички подравнявани обекти трябва да бъдат маркирани предварително с мишката при натиснат бутон *Ctrl*. След избора на желаното подравняване и разстояние между обектите се натиска бутон *OK*. Обърнете внимание, че при хоризонталното подравняване се задава вертикално разстояние между обектите и обратно.

На фиг.10.11 е показано примерно разположение на представители на всички графични обекти върху прозореца на редактора. За да се стартира този графичен интерфейс, се натиска бутон *Run*, при което се получава работния графичен прозорец (фиг.10.12).



фиг. 10.11



фиг. 10.12

Т.к. обаче зад тези обекти няма описание на функции, при това стартиране на графичния интерфейс няма да бъдат изпълнени никакви действия.

Ако се опитате да въздействате с мишката върху някои от обектите (бутони, флагове, плъзгачи), ще видите че те реагират, но без видими резултати. Това е така, защото още не са съставени *Callback* – функциите, обработващи тези въздействия. Това е по-трудната и трудоемка част от създаването на GUI с помощта на guide.

При първоначално стартиране на разработвания интерфейс с бутона *Run*, по подразбиране се създават автоматично два файла. Единият от тях е с разширение **.fig* и съхранява информация за видимата графична част на интерфейса. Вторият е обикновен *m - файл*, който служи за програмиране на обработката на събитията, свързани с отделните управляващи елементи. Той се отваря автоматично от редактора на *Matlab* при всяко стартиране на приложението от средата на графичния редактор.

M - файлът се състои от основна функция и нейни подфункции. Основната функция и всички подфункции, които не съдържат в името си думата Callback, не е необходимо да бъдат променяни от програмиста. Сред подфункциите има такива, които се състоят само от заглавен ред и няколко автоматично генерирани указания под формата на коментари. Това са

Callback функциите, които се разпознават по структурата на имената им – етикета (tag) на съответния графичен елемент, долно тире „_” и думичката “Callback”. Тези функции програмистът постепенно запълва с програмен код до окончателно завършване на GUI.

Както беше вече казано в началото обемът на информацията между отделните Callback функции се осъществява чрез техните входни аргументи:

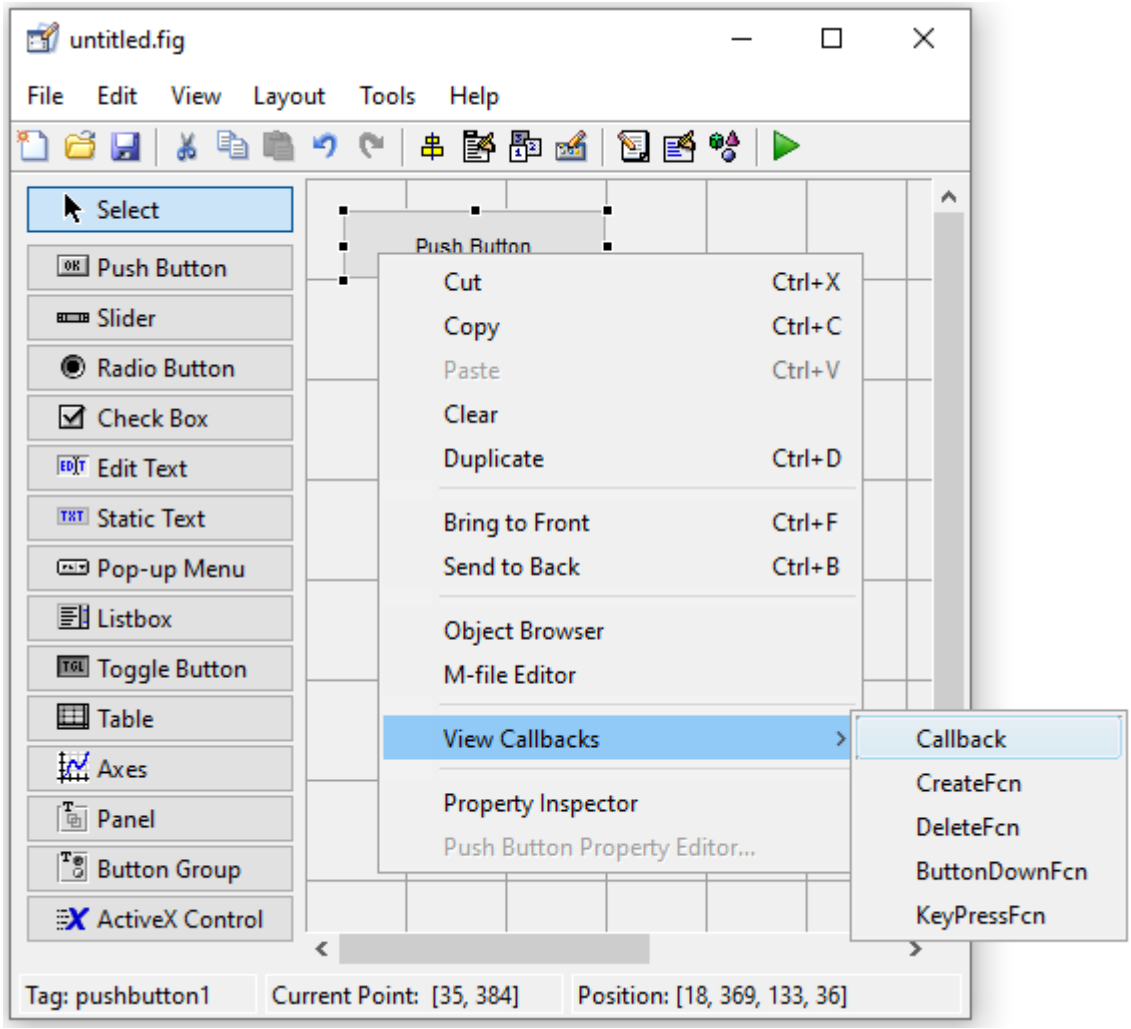
- *hObject* – указател на обекта, свързан с текущата функция;
- *eventdata* – аргумент, запазен за бъдещи версии;
- *handless* – запис с полета, в които се съхраняват указателите на всички графични обекти. Полетата имат имена, съвпадащи със свойството „Tag” на съответните обекти.

Отваряне на m – файла с автоматично позициониране на заглавния ред на дадена *Callback*–функция може да се извърши от контекстното меню, което се появява при натискане на десен бутон върху съответния обект (фиг.10.13).

От това контекстно меню се избира позицията *View Callbacks*/*Callback*. Същото меню може да се използва и за други действия, например за отваряне на често използвания *Property Inspector*.

Свойствата на даден обект могат да бъдат постоянни по време на работа на приложението или променливи. В зависимост от това присвояването на стойности на свойствата на обектите става по два начина:

1. С помощта на „Инспектора на свойства” (*Property Inspector*) – за постоянните свойства;
2. Програмно в Callback – функциите – за променливите свойства;



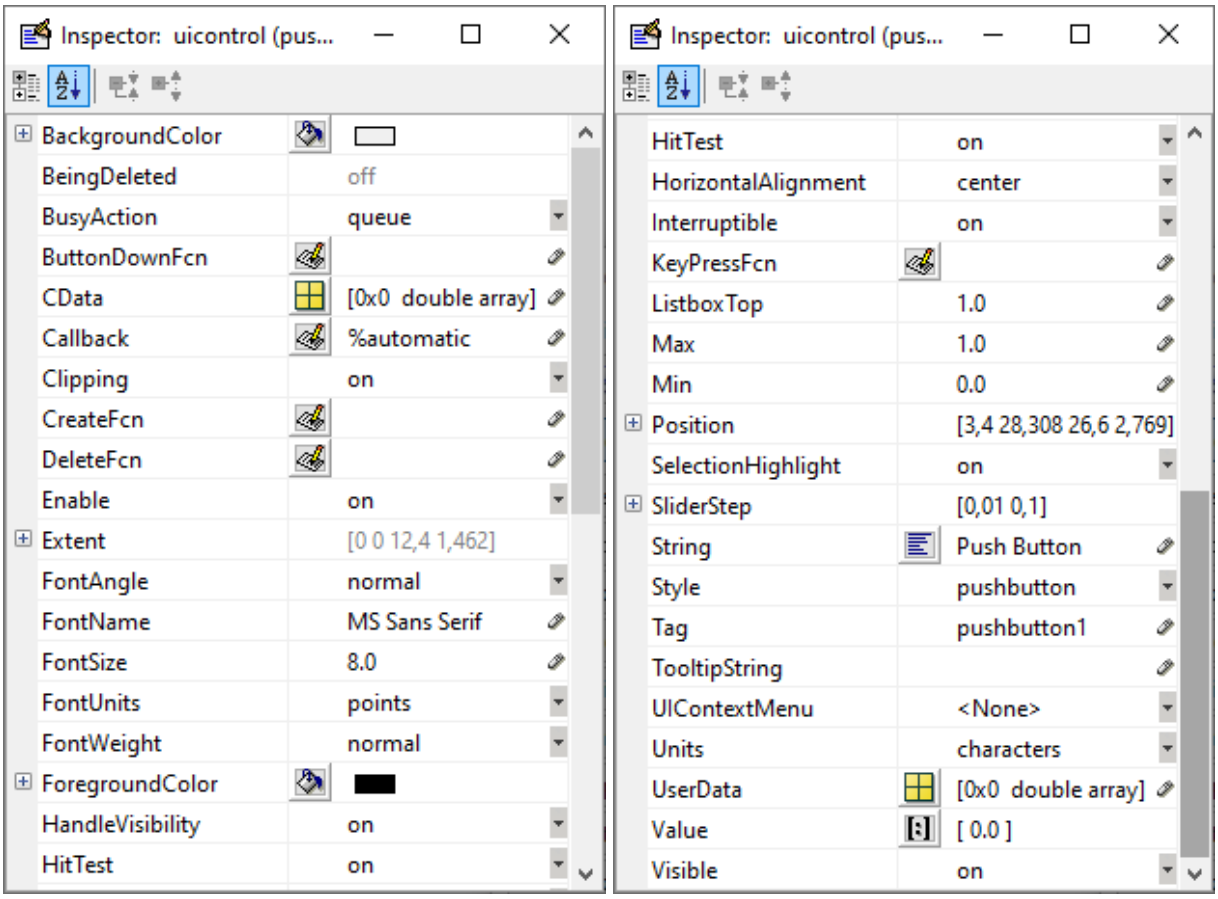
фиг. 10.13

Извикването на прозореца със свойствата на даден обект става най-бързо с двойно кликване на мишката върху този обект. На фиг.10.14 е представен прозорецът със свойствата на обекта *Pushbutton* (обикновен бутон). Част от свойствата имат подразбиращи се стойности, които могат да бъдат променяни при необходимост.

При работа с guide, най-важното свойство на един обект е свойството му „*Tag*” - етикет или име. Това свойство служи за еднозначното дефиниране на обекта и се поставя автоматично в началото на името на неговата Callback – функция. Стойността „*Pushbutton*” на свойството „*Style*” тук е въведен автоматично, тъй като обектът *pushbutton*, чийто *Property Inspector* е отворен, вече е бил създаден. Същото се отнася и за свойството „*Position*”. Чрез задаване на стойност на свойството „*String*”, се задава надпис върху повърхнината на обекта (в случая върху бутона).

След като се избере с мишката реда с дадено свойство, се задава или избира от падащия списък неговата стойност и се натиска *Enter* или малкото

бутонче в края на реда. Стойността на свойството „*String*” се въвежда в отделен прозорец, който се отваря след натискане на бутона, намиращ се в средата на реда.



фиг. 10.14

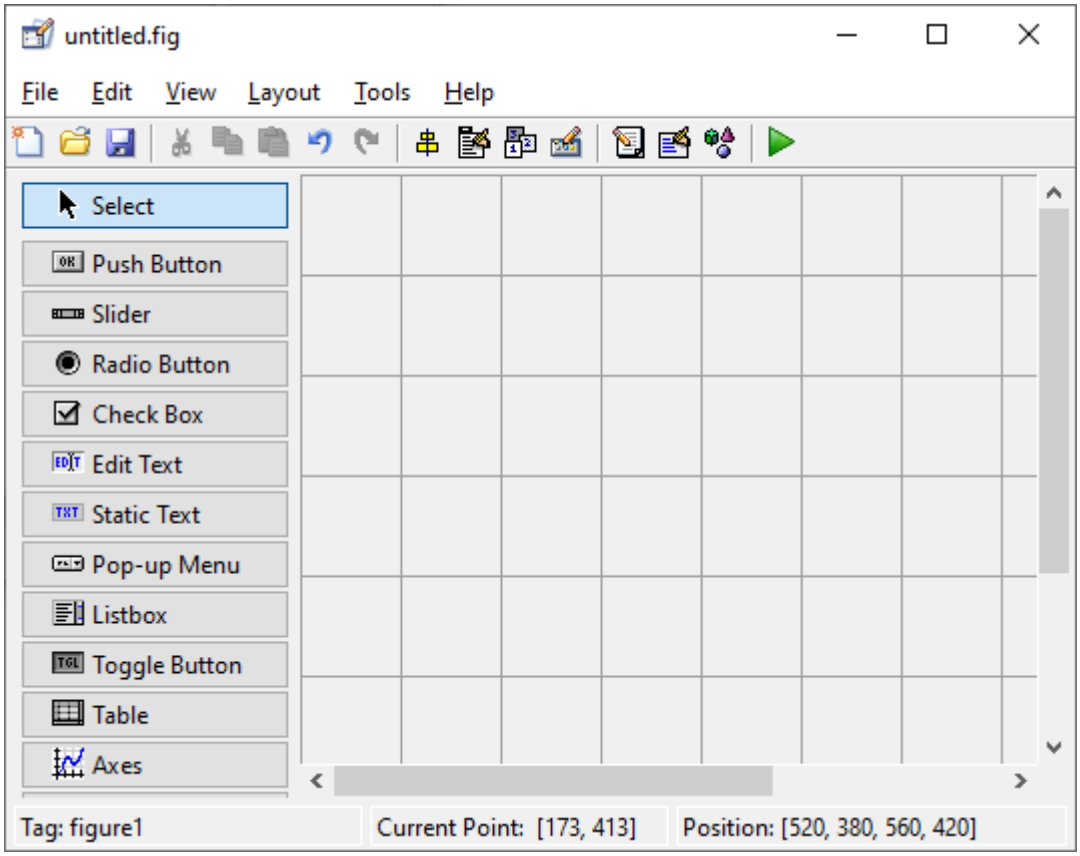
Създаване на графичен потребителски интерфейс за изчертаване на двумерна графика

За представяне графичните възможности на *Matlab* последователно ще бъде представено създаването на графичен потребителски интерфейс, визуализиращ изчертаването на двумерна графика.

Първата стъпка от създаване на приложението е стартиране на guide чрез въвеждане на името му в командния прозорец

>> guide

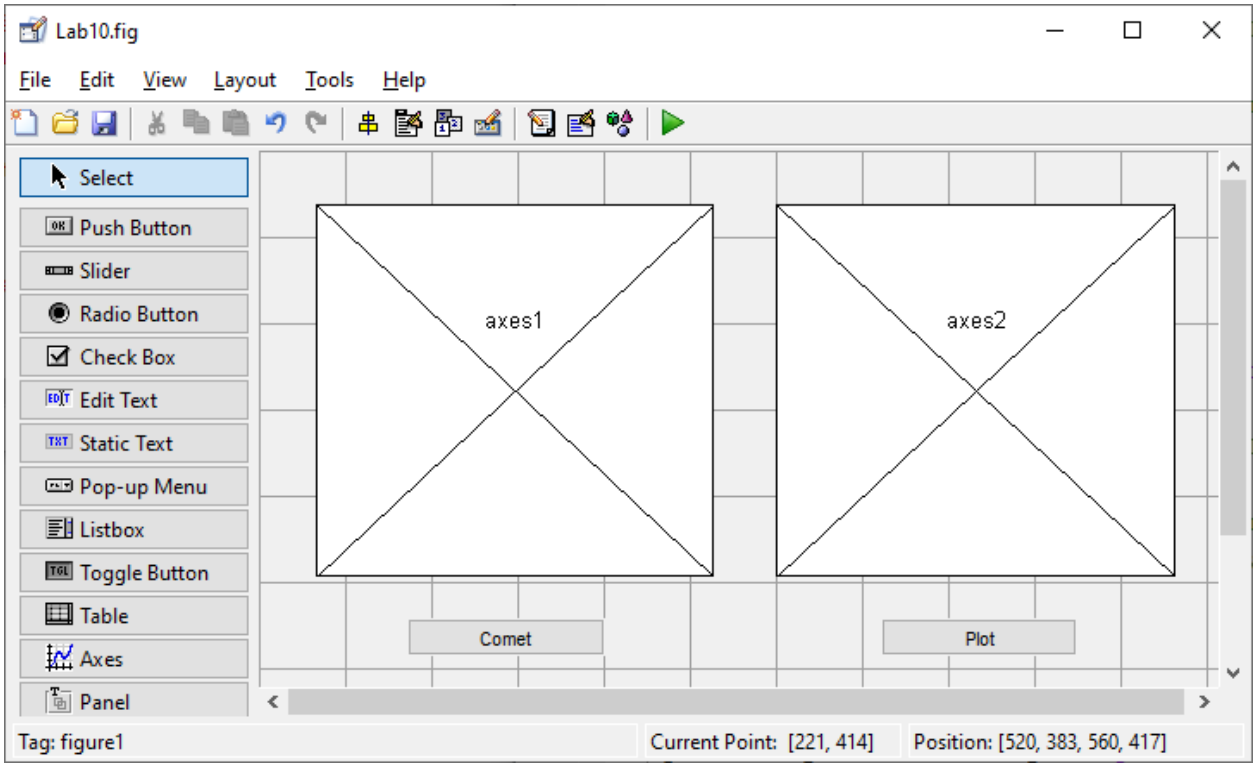
и отваряне на нов графичен прозорец (виж фиг.10.1 в началото на лабораторното упражнение).



фиг. 10.15

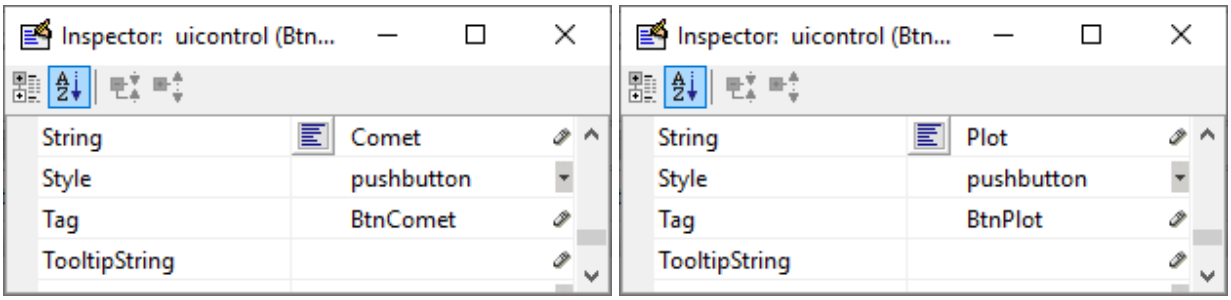
Върху вертикалния панел (фиг.10.15) са разположени бутоните на всички графични обекти, които могат да бъдат разполагани с мишката на произволно място върху графичния прозорец.

На (фиг.10.16) са представени първите стъпки при изграждането на потребителския интерфейс, а именно разполагане в полето на два обекта от тип *Axes* и два обекта от тип *Push Button*.



фиг. 10.16

За всеки от бутоните в *Property Inspector* се задават стойности на параметрите *Tag* и *String* (фиг.10.17).



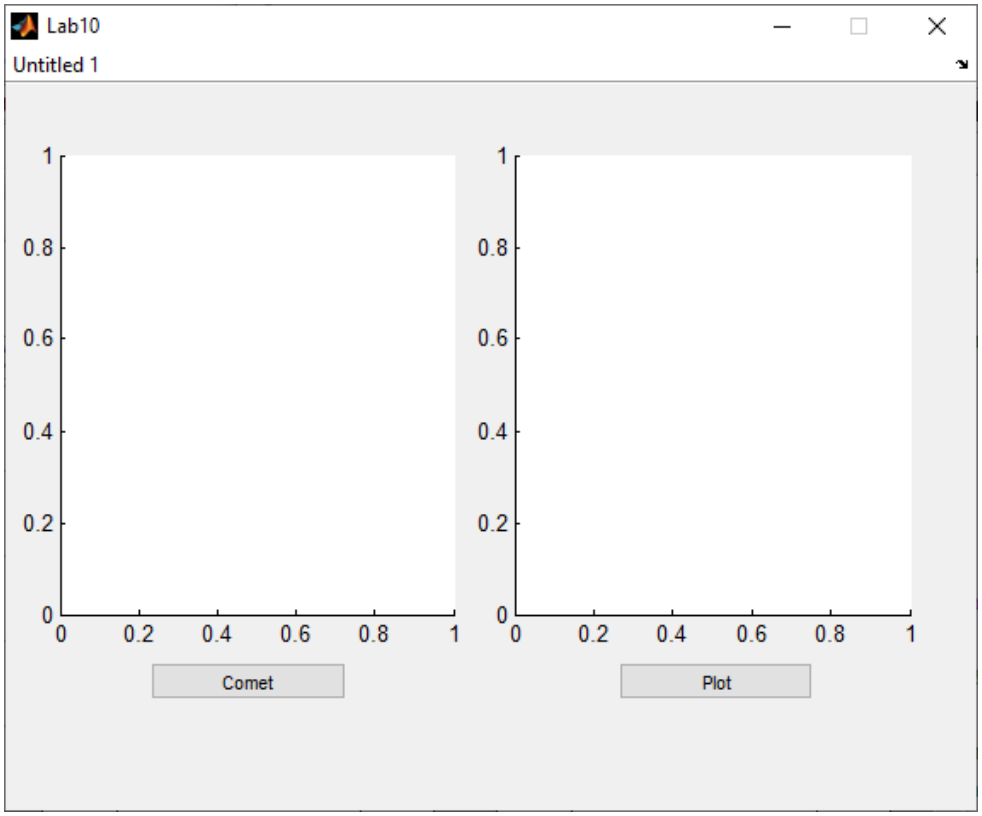
фиг. 10.17

За да бъдат изчертавани графики при натискане на единия от двата бутона, е необходимо записване на съответните команди в генерирания m-файл.

Сорс кодът на графичния потребителски интерфейс се намира в края на лабораторното упражнение.

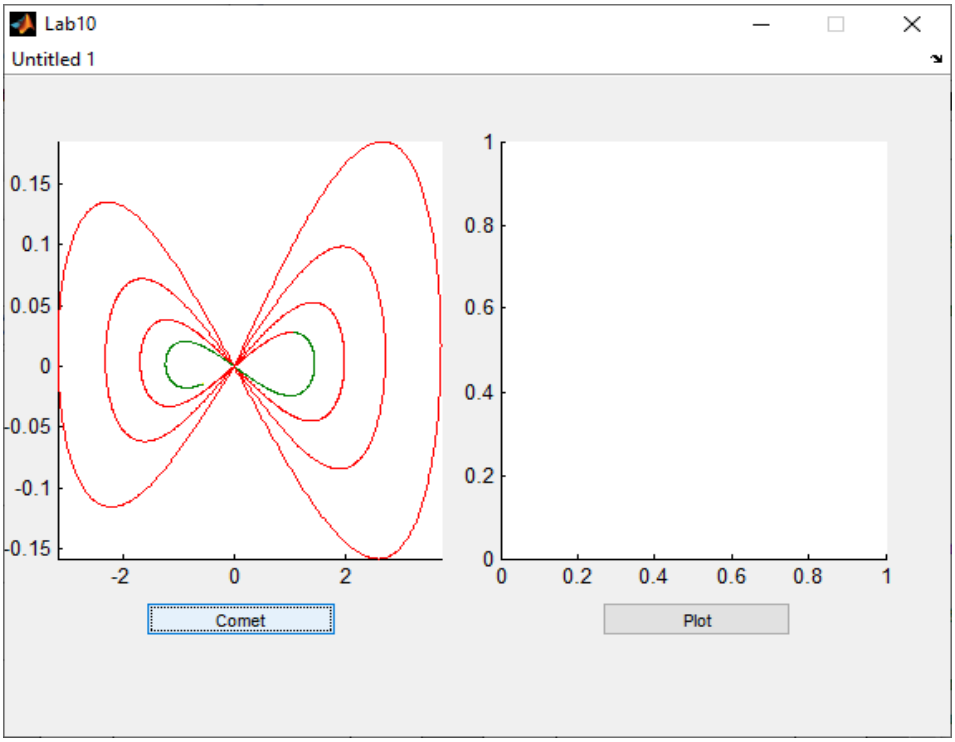
Стартиране на създадения графичен потребителски интерфейс

След записване на желаните команди в m-файла, графичният потребителски интерфейс (GUI) може да бъде стартиран чрез *Run Figure*, при което се появява прозорецът на създаденото приложение (фиг.10.18).



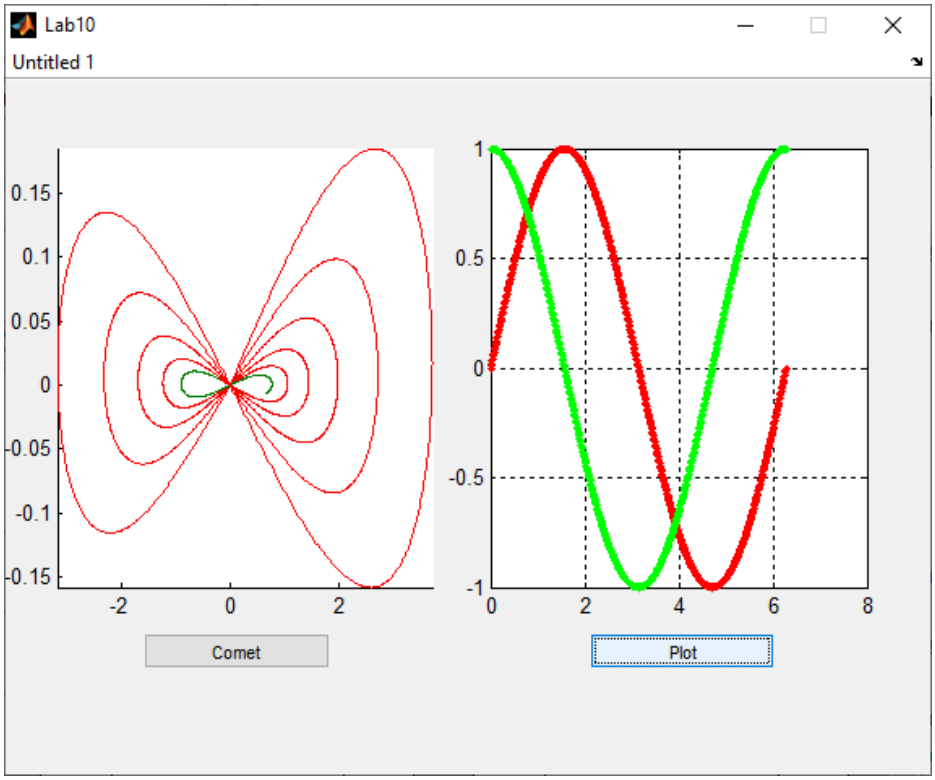
фиг. 10.18

За да бъдат визуализирани графиките е необходимо натискане на бутоните за тяхното изчертаване. При натискане на бутон Comet се наблюдава анимирано изчертаване на графиката, резултатът от което е показан на фиг.10.19.



фиг. 10.19

При натискане на бутон *Plot* се изчертава двумерна графика, която може да бъде видяна на фиг.10.20.



фиг. 10.20

Когато е приключена работата с приложението, то може да бъде затворено от знака „x” в горния десен ъгъл.

Задачи за изпълнение:

1. Да се създаде графичен потребителски интерфейс, с който да се визуализира изчертаването на тримерни графики в средата на *Matlab*.

Сорс код на графичен потребителски интерфейс за изчертаване на двумерна графика чрез функции *comet* и *plot*.

```
function varargout = Lab10(varargin)
% LAB10 M-file for Lab10.fig
% LAB10, by itself, creates a new LAB10 or raises the existing
% singleton*.
% H = LAB10 returns the handle to a new LAB10 or the handle to
% the existing singleton*.
%
% LAB10('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in LAB10.M with the given input arguments.
%
% LAB10('Property','Value',...) creates a new LAB10 or raises the
% existing singleton*.  Starting from the left, property value pairs
are
% applied to the GUI before Lab10_OpeningFcn gets called.  An
%  unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Lab10_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Lab10

% Last Modified by GUIDE v2.5 16-Feb-2026 12:23:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Lab10_OpeningFcn, ...
                   'gui_OutputFcn',  @Lab10_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Lab10 is made visible.
function Lab10_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Lab10 (see VARARGIN)
% Choose default command line output for Lab10
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Lab10 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Lab10_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in BtnComet.
function BtnComet_Callback(hObject, eventdata, handles)
%  Подфункция за обработка на събитието callback на левия бутон
%BtnComet
% Правим текуща лявата координатна система
axes(handles.axes1)
t=0:0.01:50;
x=4*exp(-0.05*t).*sin(t);
y=0.2*exp(-0.1*t).*sin(2*t);
comet(x,y);
plot(x, y, 'LineWidth', 5), grid on

% --- Executes on button press in BtnPlot.
function BtnPlot_Callback(hObject, eventdata, handles)
%  Подфункция за обработка на събитието callback на десния бутон
%BtnPlot
% Правим текуща дясната координатна система
axes(handles.axes2)
x = 0 : pi/300 : 2*pi;
y = sin(x);
y1=cos(x);
plot(x, y, 'r.-', x, y1, 'g.:')
grid on
```

Лабораторно упражнение 11
Решаване на обикновени диференциални уравнения в средата на *Matlab*

Цел на лабораторното упражнение

Целта на настоящото лабораторно упражнение е запознаване с възможностите за числено и символно решаване на обикновени диференциални уравнения (ОДУ) в *Matlab* и работа с функциите *ode45* и *dsolve*.

Функции (solver-и) за числено решаване на диференциални уравнения в Matlab

В *Matlab* има няколко функции, с които могат да бъдат решавани числено обикновени диференциални уравнения.

- *ode45* – най-често използваната функция, с която може да се извърши числено решаване на обикновени диференциални уравнения със зададени начални условия. Едностъпков алгоритъм, който използва адаптивен метод на Рунге–Кута от 4-ти и 5-ти ред, който автоматично избира подходяща стъпка на интегриране с цел постигане на добра точност и ефективност.
- *ode23* – тази функция се базира на метода на Рунге–Кута от 2-ри и 3-ти ред. Представлява едностъпков алгоритъм и се използва за бързо решаване на диференциални уравнения с не особено висока точност.
- *ode113* – тази функция използва метода на Адамс-Бешфорт-Мултон. Представлява многостъпков алгоритъм, при който за пресмятане на текущото решение са необходими решенията от няколко предишни точки.

Обикновени диференциални уравнения

Диференциалните уравнения описват динамиката на инженерни системи – механични, електрически, термични и др.

Общ вид на ОДУ от първи ред:

$$\frac{dy}{dt} = f(t, y) \tag{1}$$

с начално условие:

$$y(t_0) = y_0 \tag{2}$$

Решението представлява функция $y(t)$, която удовлетворява както уравнението, така и началното условие.

Същност на функцията ode45

Функцията работи само със системи от първи ред и се прилага за уравнения от вида

$\dot{y} = f(t, y)$, с начално условие $y(t_0) = y_0$.

Синтаксисът ѝ в *Matlab* е :

[t, y] = ode45(odefun, tspan, y0)

Аргументите на *ode45* представляват:

- *odefun* – функция, описваща дясната страна на диференциалното уравнение;
- *tspan* – вектор ([t0, tf]), задаващ интервала на интегриране;
- *y0* – начално условие.

Потребителят дефинира дясната страна на диференциалното уравнение под формата на функция *odefun,* задава интервала на интегриране *tspan* и началните условия *y0.*

Изходящите аргументи са:

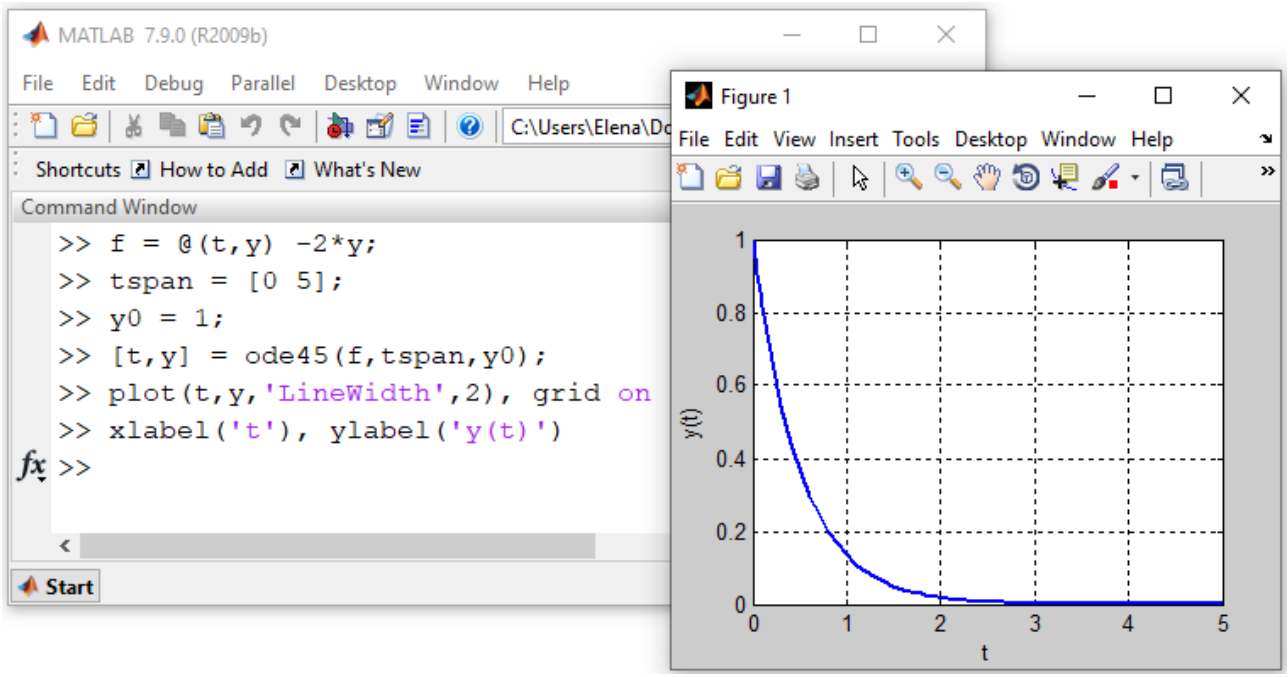
- *t* – вектор, стойностите на който представляват моментите на времето;
- *y* – числено решение. Представлява матрица, стълбовете на която съдържат отделните решения на системата.

Получените резултати могат да бъдат анализирани графично или използвани за по-нататъшни изчисления.

🤔 ***Задача***: Да се реши диференциалното уравнение от първи ред:

$$\frac{dy}{dt} = -2y$$

с начално условие $y(0) = 1$. Полученото решение да се представи в графичен вид.



фиг. 11.1

При необходимост да се реши диференциално уравнение от втори и по-висок ред, то първо трябва да бъде преобразувано и представено като система диференциални уравнения от първи ред, след което да бъде приложена функцията *ode45*.

Задача: Да се реши диференциалното уравнение от втори ред:

$$\ddot{y} + 2\dot{y} + 5y = 0$$

с начални условия $y(0) = 1$ и $\dot{y}(0) = 0$.

Упътване: За да се реши уравнението от втори ред, е необходимо да бъде представено като система от първи ред. Това може да стане по следния начин – прави се полагане

$$y_1 = y$$

$$y_2 = \dot{y}$$

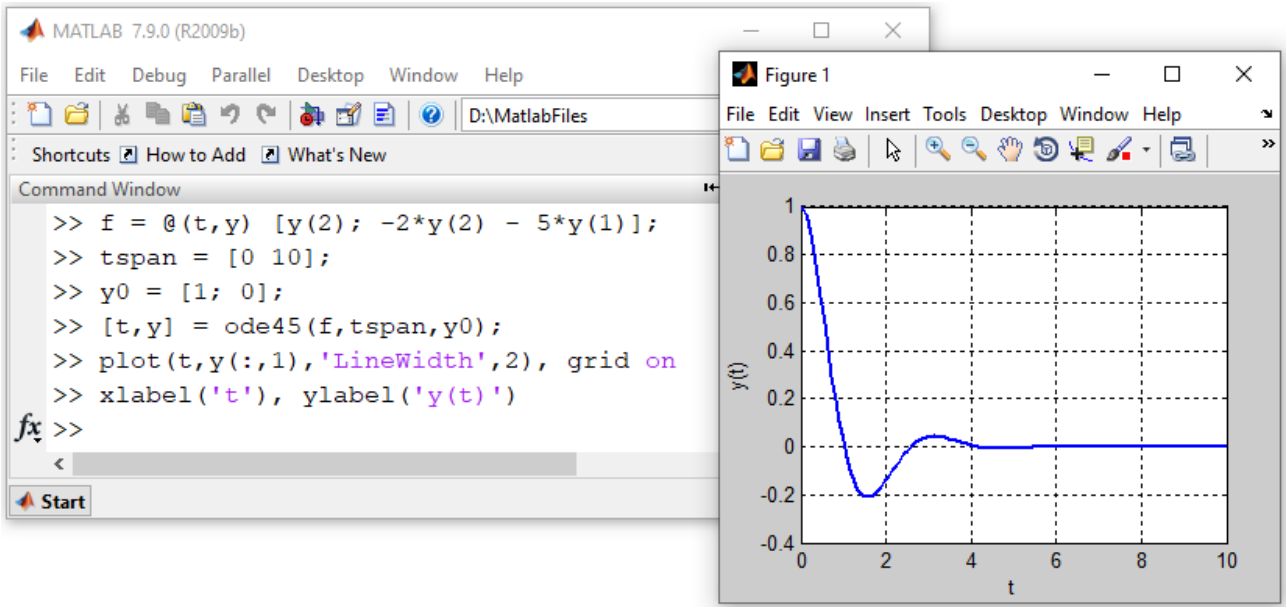
при което се получава системата:

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -2y_2 - 5y_1 \end{cases}$$

Получената система съдържа диференциални уравнения от първи ред, поради което вече може да бъде приложена функцията за числено решаване на диференциални уравнения *ode45* по описания начин.

Особеното в случая е, че задаването на функцията f става във вид на вектор. Първият му елемент е дясната страна на първото уравнение на системата, а вторият елемент съдържа дясната страна на второто уравнение.

Началните условия в случая също са две и също се записват като вектор.



фиг. 11.2

Задача: Да се реши диференциалното уравнение от първи ред:

$$\frac{du}{dt} = \frac{1}{RC}(U - u),$$

описващо процес на зареждане на кондензатор в RC верига. В представения математичен модел

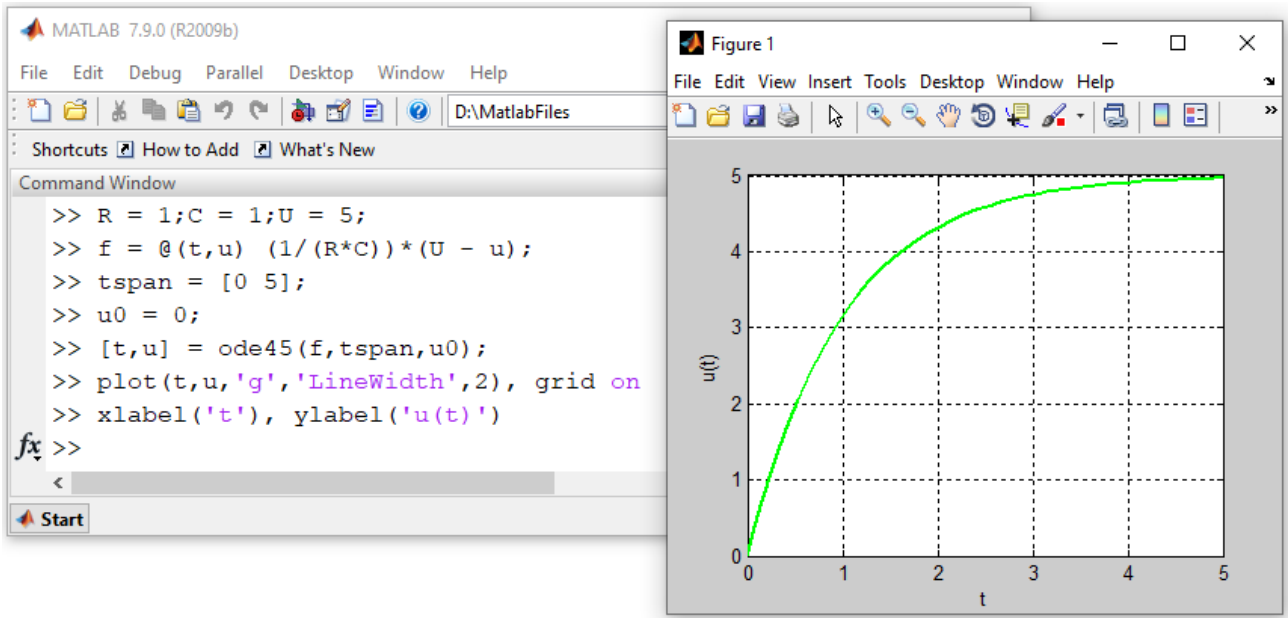
$u(t)$ – е напрежението върху кондензатора;

U – входното напрежение;

R, C – параметри във веригата.

Да се намери решението при стойности

$$R = 1\Omega,\ C = 1F,\ U = 5V \text{ и начално условие } u(0) = 0.$$



фиг. 11.3

Задача: Да се реши диференциалното уравнение от втори ред:

$$\ddot{y} + 0.5\dot{y} + 4y = 0$$

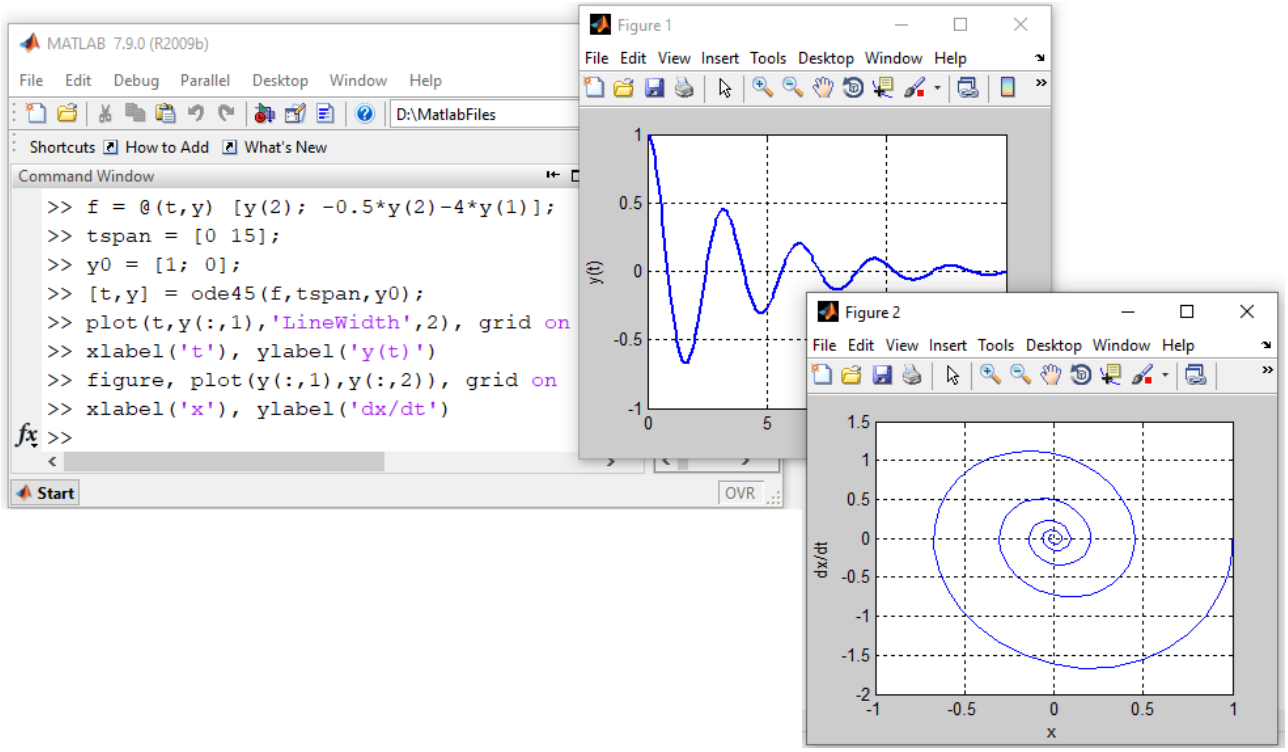
с начални условия $y(0) = 1$ и $\dot{y}(0) = 0$.

Да се изчертаят две графики - $y(t)$ и фазовия портрет $\dot{y} = f(y)$

За да се реши задачата, отново трябва да се преобразува диференциалното уравнение в система от две уравнения от първи ред. Получената система има вида:

$$\begin{cases} \dot{y}_1 = y_2 \\ \dot{y}_2 = -0.5y_2 - 4y_1 \end{cases}$$

и за решаването ѝ се използва функцията *ode45*.



фиг. 11.4

Аналитично решаване на диференциални уравнения в Matlab

В *Matlab* има има възможност и за символно решаване на обикновени диференциални уравнения. За тази цел се използва командата *dsolve*. Синтаксисът на командата има вида:

dsolve('диференциално_уравнение', 'начално условие')

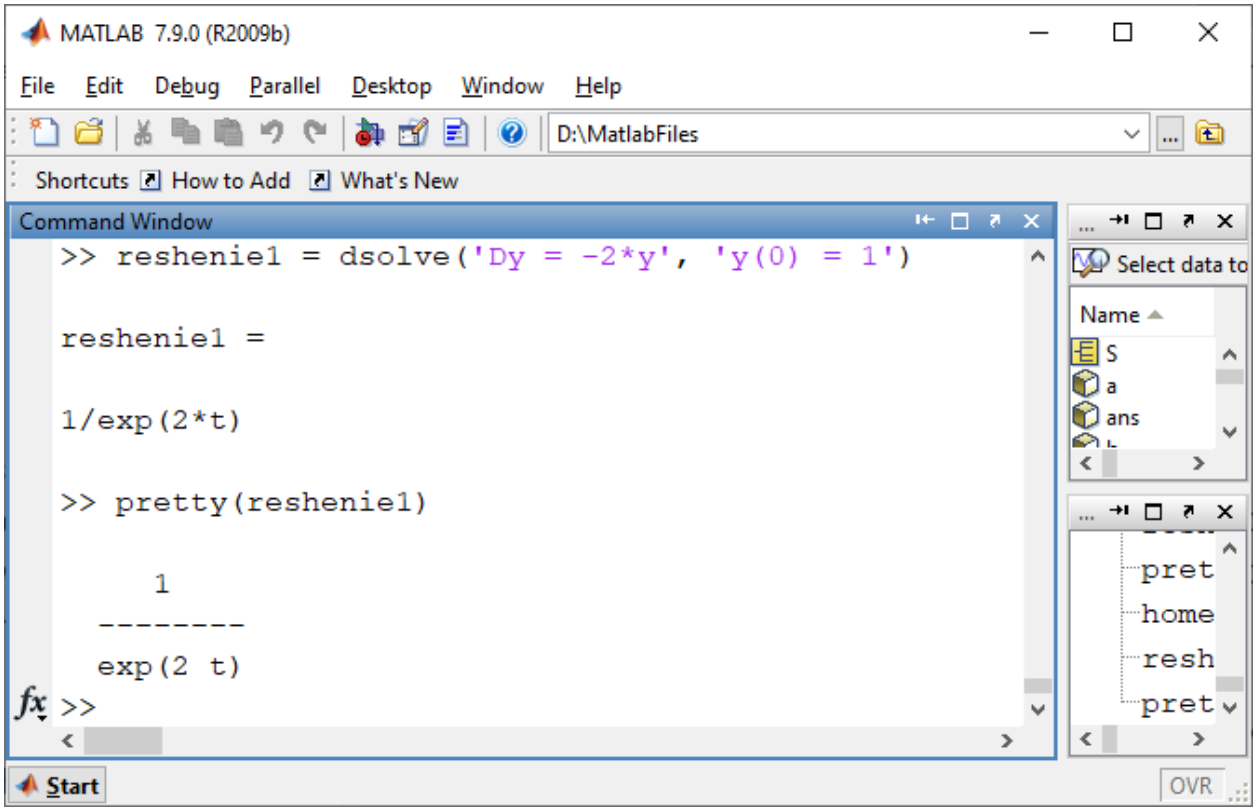
Диференциалното уравнение се дефинира като диференцирането спрямо независимата променлива ($\frac{d}{dx}$, $\frac{d}{dy}$ и т.н.) се записва чрез *Dx*, *Dy* и т.н.

Началните условия се задават, записвайки *y(0) = a*. Ако не бъдат зададени начални условия или те са по-малко от дадените променливи, то в решението ще се съдържат константи *C1*, *C2* и т.н.

Задача: Да се намери аналитичното решение на диференциалното уравнение от първи ред:

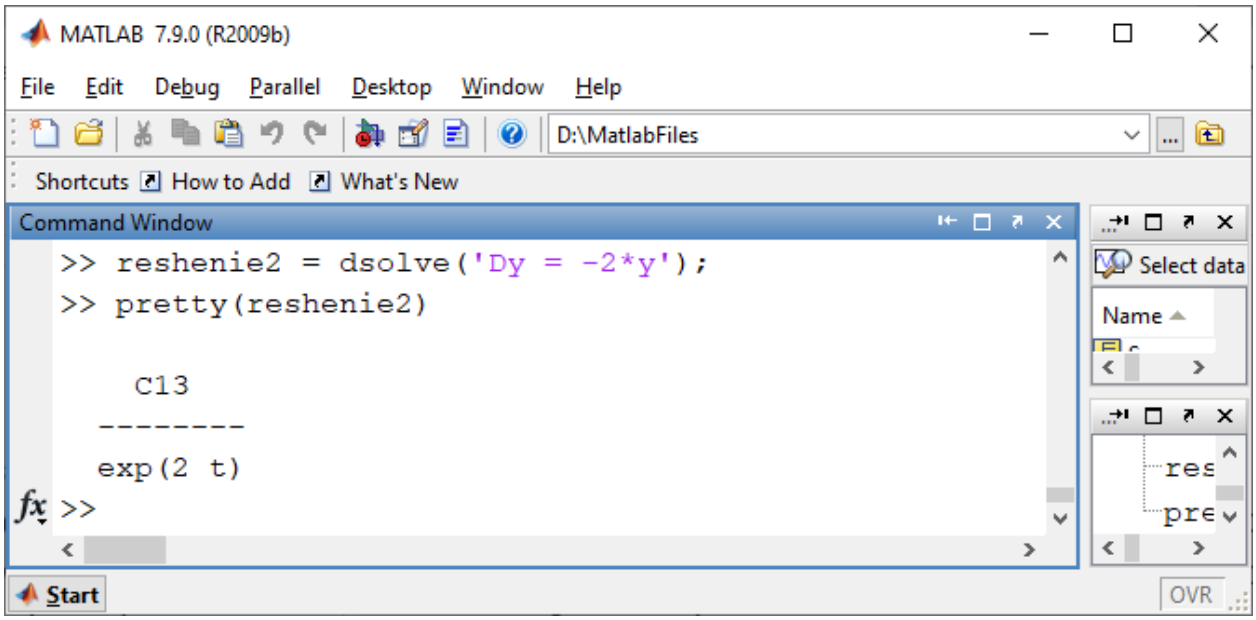
$$\frac{dy}{dt} = -2y$$

с начално условие $y(0) = 1$.



фиг. 11.5

На фиг.11.6 може да се види аналитичното решение на даденото ДУ ако като аргумент на командата не се зададе начално условие.



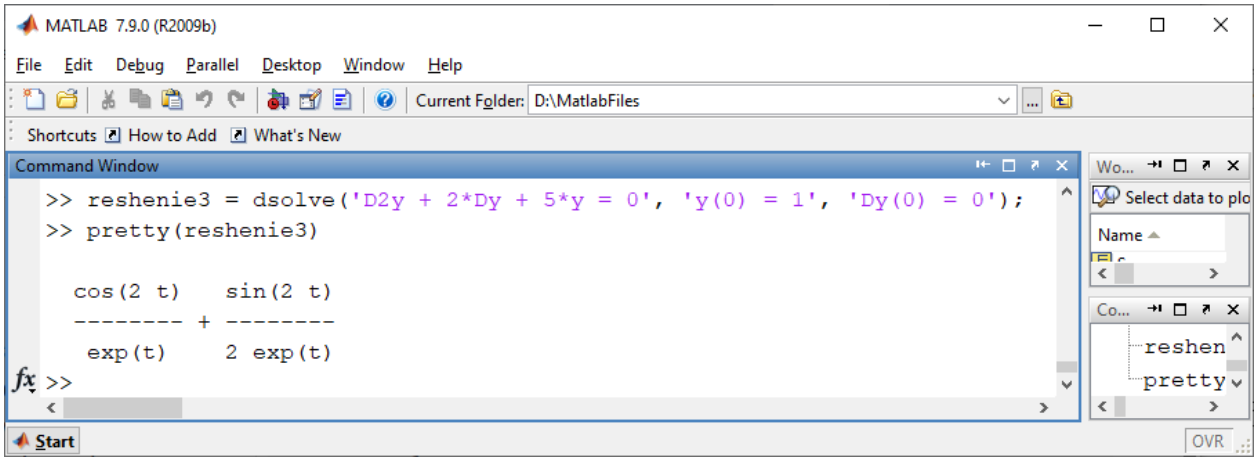
фиг. 11.6

В случай, че диференциалното уравнение е от по-висок ред, то производните се задават като *D2t* (втора производна на *t*), *D3x* – трета производна на *x* и т.н.

Задача: Да се намери аналитичното решение диференциалното уравнение от втори ред:

$$\ddot{y} + 2\dot{y} + 5y = 0$$

с начални условия $y(0) = 1$ и $\dot{y}(0) = 0$.



фиг. 11.77

Задачи за изпълнение:

1. Да се реши диференциалното уравнение от втори ред:

$$\ddot{y} + 0.5\dot{y} + 4y = 2\cos(t)$$

с начални условия $y(0) = 0$ и $\dot{y}(0) = 0$. Интервалът на интегриране да бъде от 0 до 20.

2. Да се намери числено и аналитично решение на диференциалното уравнение от първи ред:

$$\frac{dy}{dt} = y - y^2$$

с начално условие $y(0) = 0.1$ и интервал на интегриране от 0 до 10.