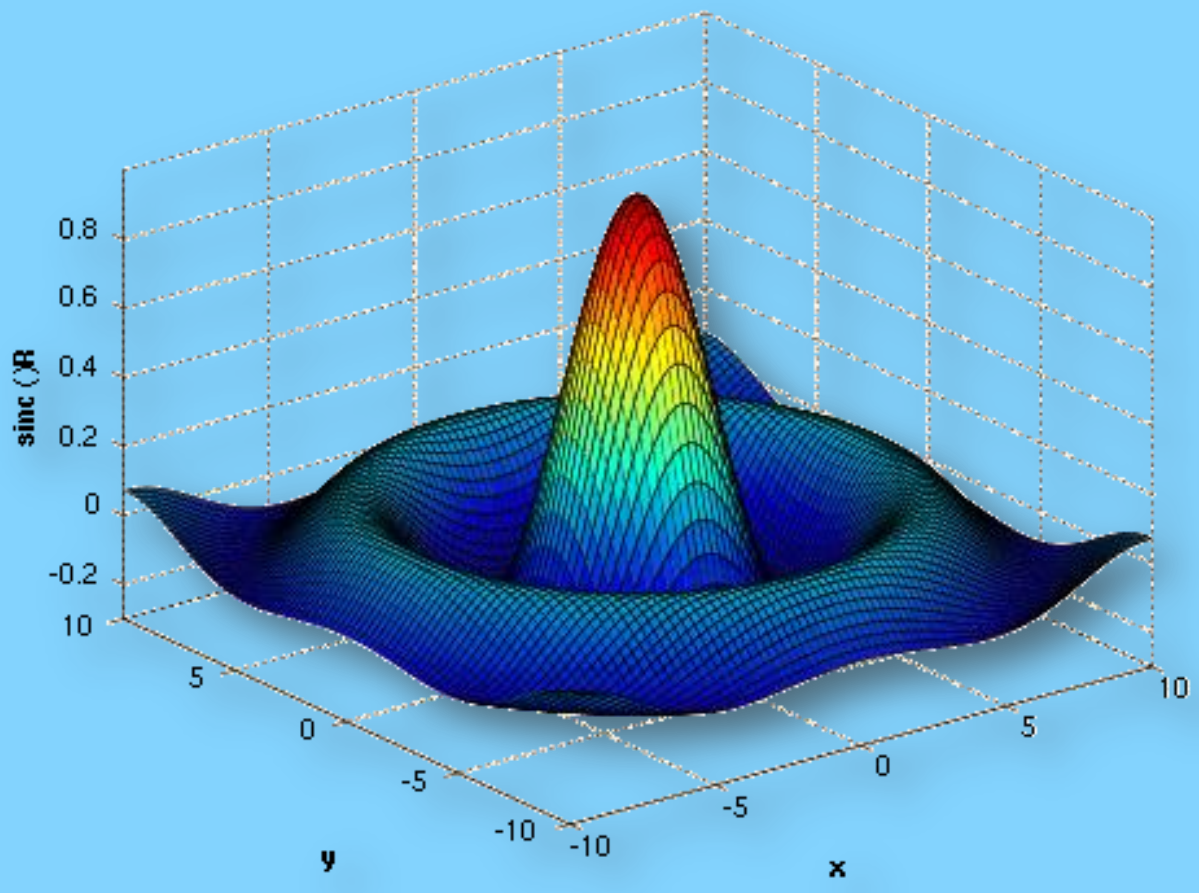
Namazov Manafəddin, Allahverdiyeva Nailə

KOMPÜTERLƏRIN MÜHƏNDISLIKDƏ TƏTBIQI

MATLAB



BAKI - 2013

B A K I A L İ N E F T M Ə K T Ə B İ

Namazov Manafəddin, Allahverdiyeva Nailə

KOMPÜTERLƏRİN MÜHƏNDİSLİKDƏ TƏTBİQİ

MATLAB

(Mühazirələr konspekti)

Bakı – 2013

Müəlliflər:

Namazov Manafəddin Bəşir oğlu – Bakı Ali Neft Məktəbinin "Kompüter və informasiya-kommunikasiya texnologiyaları" kafedrasının dosenti;

Allahverdiyeva Nailə Rizvan qızı - Bakı Ali Neft Məktəbinin "Kompüter və informasiya-kommunikasiya texnologiyaları" kafedrasının dosenti.

Namazov M.B., Allahverdiyeva N.R. MATLAB (dərs vəsaiti). – Bakı: ARDNŞ-ın mətbəəsi, 2013, 207 səh.

Dərs vəsaiti MATLAB proqram təminatı, MATLAB proqram təminatında proqramlaşdırma, funksiyaların qrafiki, SYMBOLIC MATHEMATICS TOOLBOX – simvollu riyazi alətlər paketi və SIMULINK proqramı bölmələrini əhatə edir.

Dərs vəsaiti Bakı Ali Neft Məktəbində "Kompüterlərin Mühəndislikdə Tətbiqi" fənnində deyilən mühazirələr əsasında tərtib edilmişdir. Dərs vəsaitindən ali texniki məktəblərin bakalavr və magistləri, müəllimlər, aspirantlar və elmi-tədqiqatçılar istifadə edə bilərlər.

MÜNDƏRICAT

GIRIŞ

Müasir kompüterlər yaddaş və imkanları ilə informasiyaların ötürülməsi, yeni texnologiyaların yaradılması, elm və texnikanın mürəkkəb məsələlərinin həll edilməsi prosesində alətə çevrilmişdir.

Obyektiv səbəblərə görə həll edilən məsələlərin mürəkkəbliyinin artması alqoritmlərin mürəkkəbləşməsinə və onların Ci, Pascal, Fortran və s. alqoritmik dillərdə reallaşdırılmasına gətirir. Bundan əlavə kodların hazırlanmasına çox vaxt sərf edilir. Bu səbəblər avtomatlaşdırılmış layihələndirmə sisteminin yaradılmasına gətirmişdir. Belə sistemlər kifayət qədər çoxdan meydana gəlmiş və dar ixtisaslaşdırılmışdır. Riyazi ALS arasında MathCAD (MathSoft Ins.), Mathematica (Wolfram Research, lns.), MATLAB (MathWorks lns.), Maple V (Wayerloo Maple Lns.) daha çox populyarlıq qazanmışlar.

Bu hazırlanmış material ən güclü və daha çox açıq MATLAB sisteminə və bu sistemdə işləyən hesablama proqramlarına həsr edilir. Bu sistem kifayət qədər mürəkkəb tətbiqi məsələlərin həlli üçün mexanizm rolunu oynaya bilər. Burada əsas problem sistemin məhsuldarlığının artırılması problemidir.

Bu kitabda sistemin məhsuldarlığının əhəmiyyətli dərəcədə artırılmasına imkan verən momentlər və hətta MATLAB çərçivəsindən "çıxmaq" izah edilir.

Dərslik kimi təklif olunan bu kitabda iki və üçölçülü qrafiklərin qurulması, tərtibatı və onların başqa sistemlərə ötürülməsi qaydaları, həmçinin SYMBOLIC MATHEMATICS TOOLBOX tətbiqi proqramlar paketinin bir hissəsi öz əksini tapmışdır.

Dərsliyin sonunda isə MATLAB V sisteminin indeks göstəriciləri verilmişdir. Bu isə sistemin əmrlərinin, funksiyalarının və operatorlarının başa düşülməsində əsas vasitədir.

MATLAB ədədi hesablamalar sistemidir, hər hansı "mexaniki" hesabatlar üçün simvollarla hesablama paketindən, məsələn, Maple V paketindən istifadə etmək imkanına malikdir.

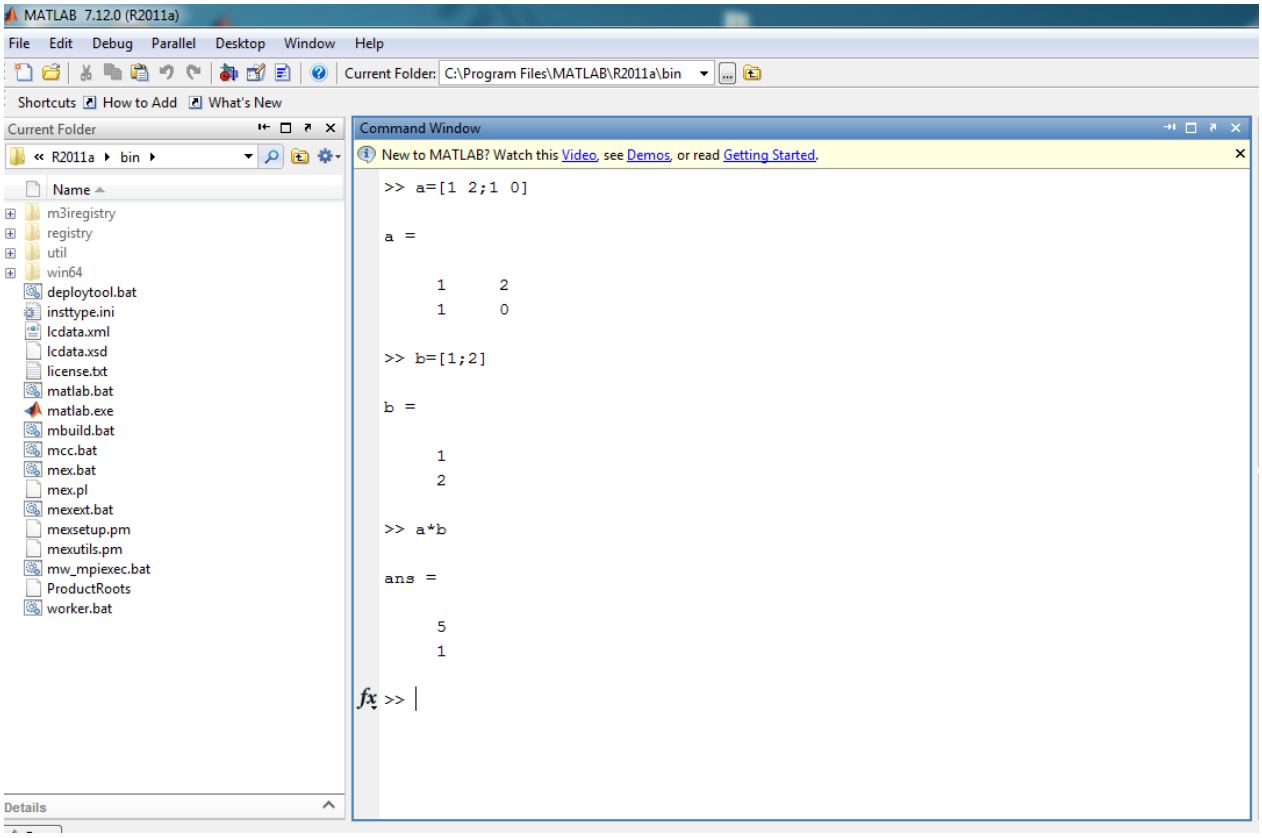
MATLAB-ın xüsusiyyəti Simulink üstqurumundan ibarətdir. Bu isə RAD rejimində bir çox məsələləri həll etməyə, yəni ayrı-ayrı bloklardan model yaratmağa və prosesi işə salmağa imkan verir.

Bu kitabın davamı olaraq çapa hazırlanmış materiallar ədədi üsulların MATLAB proqramlar paketi ilə reallaşdırılmasına həsr olunmuşdur.

1. MATLAB PROQRAMINA GİRİŞ

1.1 INTERFEYS

MATLAB əmrlər interfeysinə malikdir. Əgər hər hansı əmr daxil edilmişdirsə, onda siz onu ləğv (geri götürə) və ya dəyişə bilməzsiniz. Əgər siz əvvəllər Maple V-də işləmişsinizsə, onda bu başlanğıcda sizin xoşunuza gəlməyəcəkdir. Amma sistemin ideyasını yavaş-yavaş anladıqdan sonra Mathematics və ya Maple-də düsturun redaktə olunması imkanı gülməli görünəcəkdir. Sistem işə salındıqdan sonra işləməyə hazırdır və sizdən əmrin daxil olmasını gözləyir.



Növbəti əmrləri yığın (burada siz nə daxil etməlisinizsə, göy rənglə ayrılmışdır):

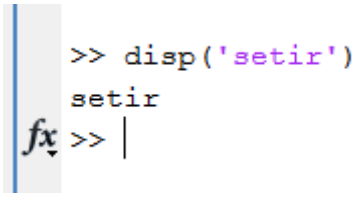
```
» a=[1 2;1 0]
a =
     1 2
     1 0

» b=[1;2]
b =
     1
     2

» a*b
ans =
     5
     1
```

Bu nümunədə siz başa düşdüyünüz kimi matris və vektor bir-birinə vurulmuşdur. Belə nümunənin seçilməsi təsadüfi olmamışdır. Iş bundadır ki, MATLAB matrislərlə və vektorlarla işi nəzərdə tutan sistemdir. Siz skalyar, verilənlər və ya səthlər massivi verə bilərsiniz (baxmayaraq ki, MATLAB-ın daxili təsəvvürü onu uyğun ölçülü massiv kimi

təyin (müəyyən) edir):

» s=10;
» d(1,1,1)=2.5
d = 2.5 0 0

» disp('sətir')
sətir



Əgər əmrin sonunda nöqtəli vergül qoysaydıq, siz əməliyyatın nəticəsinin çıxmayacağını necə seçə bilərdiniz.

MATLAB-ın hər hansı əmrinə (operatoruna) funksiyasına görə tarixçəsini almaq üçün alətlər panelində "**?**" piktoqramı seçmək olar və ya help əmrin_adı yığmaq lazımdır:

» help dir

DIR List directory.

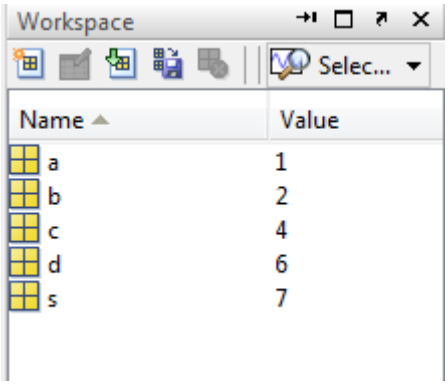
DIR directory_name lists...

Ekranın işlək oblastını təmizləmək üçün aşağıdakı əmri daxil edin:

» clc

1.2 Yaddaşın idarə edilməsi

Necə bilək ki, sistemə hansı dəyişənlər məlumdur? Bunun üçün "işçi oblastının kataloqu" -  piktoqramı seçin:



Sizə bu və ya digər massivi ehtiyatda saxlamaq üçün yaddaşın neçə özəyi ayırmaq lazım gəldiyini sistemə göstərməyə ehtiyac yoxdur. MATLAB dinamiki olaraq massivin hər bir təzə elementi üçün onu özü ayırır və bu üçüncü hissədə danışacağımız əsas xüsusiyyətlərdən biridir. Matlabda verilənlərin saxlanması, pozulması və ya yaradılması sadədir.

Verilənlərin diskdə saxlanılması üçün aşağıdakı əmri yazmaq lazımdır:

» save data_a a

Belə əmr ilə siz cari qovluqda **data-a.mat** faylında ***a*** dəyişənini saxladınız. Əgər bütün dəyişənləri saxlamaq istəyirsinizsə, onda yazmaq lazımdır:

» save data_all

Dəyişəni yaddaşdan pozmaq (silmək) üçün aşağıdakı əmr yazılmalıdır:

» clear a

Bütün dəyişənləri yaddaşdan pozmaq üçün əmr aşağıdakı kimi olmalıdır:

» clear

Diskdə əvvəlcədən saxlanılan dəyişənləri yükləmək üçün **load** əmrindən istifadə olunur:

» load data_a

Indi isə siz yenidən **a** matrisinin qiymətlərindən istifadə edə bilərsiniz.

1.3 Həqiqi ədədlər

MATLAB mühitində hesablama aparılan verilənlərin əsas tipləri, verilən dəqiqliklə ixtiyari həqiqi ədədə yaxınlaşan sonlu onluq kəsrlərdir. Sonuncu, ümumi halda ancaq sonsuz onluq kəsr şəklində göstərilə bilər. Demək olar ki, MATLAB həqiqi ədədlərlə təqribi olaraq işləyir. MATLAB-da həqiqi ədədlər mantissa və tərtib göstəricisi ilə verilir:

2.851038547e+12; -456.38456978; 0.0045692e0; 0.93185e-1; 4.5; -123

və s.

Tam ədədlərdə kəsr hissə yoxdur, amma onlar da MATLAB sistemi ilə maşın səviyyəsində kəsr ədədlər kimi həmin formada göstərilirlər. Verilənlərin belə əsas tipi **double** adlanır.

Məhz verilənlərin belə tipi, ixtiyari dəyişən üçün "var sayilma" fərz edilir. Mantissa və tərtib göstəricilərinə (maşın səviyyəsində ədədlərin ikili sistemdə yazılışından istifadə olunur) yaddaşın 8 baytı ayrılır. Nəticədə, onluq ədədin 15 əhəmiyyətli rəqəm tərtibli göstərilmə dəqiqliyinə nail olunur. Bu zaman MATLAB sistemində göstərilə bilən və mütləq qiymətə görə maksimal həqiqi ədəd

1.797693134862316e+308,

mütləq qiymətinə görə minimum ədəd isə belədir:

2.225073858507202e-308

Bu ədədlər üçün hətta xüsusi adlar verilmişdir: **realmax** və **realmin**.

MATLAB mühiti işə salındıqdan sonra onun əmrlər pəncərəsində >> dəvət işarəsi üzə çıxır.

```
>> res=5.345*2.868/3.14-99.455+1.274

res =

  -93.2990
```

Az müddətdən sonra klaviaturadan ədəd, dəyişənlərin adlarını, əməliyyat işarələrini (xüsusi halda =-lik işarəsi mənimsəmək əməliyyatına uyğundur) daxil etmək olar ki, bunlar da birlikdə hər hansı ifadəni təşkil edir. **Dəyişənin adı** hərf ilə başlayır və hərflərdən, rəqəmlərdən və işarələrin yığımından ibarətdir. MATLAB 31 simvola qədər dəyişənlərin adlarını tanıyır (qalanlara əhəmiyyət vermir) və simvollar registrlərini fərqləndirir. **Enter** düyməsinin basılması MATLAB sistemini məcbur edir ki, ifadənin qiymətini hesablasın və şəkildə görünən kimi nəticəni göstərsin.

MATLAB həqiqi ədədi ekrana çıxarmaq üçün öz əmrlər pəncərəsini gərəksiz təfsilatlarla yükləməməkdən ötrü gizlincə **short** formatından istifadə edir və bu zaman vergüldən sonra ancaq dörd onluq rəqəm görünür. Əgər tam təsvir tələb olunursa, onda klaviaturadan

format long

əmrini daxil etmək, bundan sonra hesablamanın nəticəsini görmək üçün **res** dəyişənin adını yazmaq lazımdır.

ENTER düyməsini basaraq daha ətraflı məlumat alırıq:

res = -93.29900636942675

Indi MATLAB-ın cari seansdakı iş müddətində hesablamanın bütün nəticələri yüksək dəqiqliklə göstəriləcəkdir. Əgər işin bu seansının qurtarmasına qədər MATLAB-ın əmrlər pəncərəsindən həqiqi ədədin əvvəlki dəqiqliklə vizual görünüşünə qayıtmaq tələb olunarsa,

format short

əmrini daxil etmək və ENTER düyməsini basmaqla yerinə yetirmək lazımdır:

Başqa maraqlı format həqiqi ədədin adi kəsr şəklində göstərilməsidir ki, bunun üçün

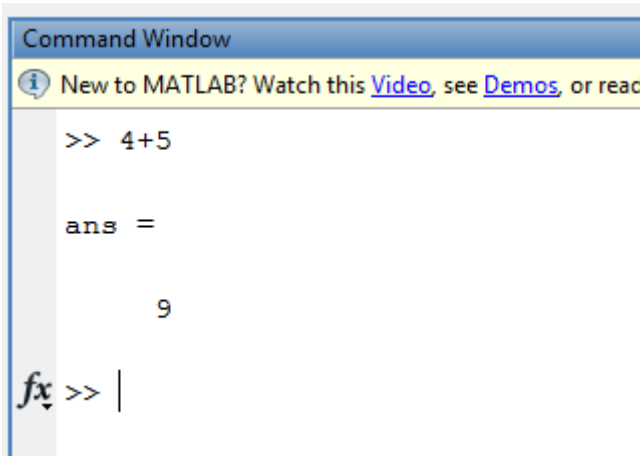
format rat

əmri daxil edilir.

Əvvəl hesablanılan **res** dəyişəni aşağıdakı şəkildə göstəriləcəkdir:

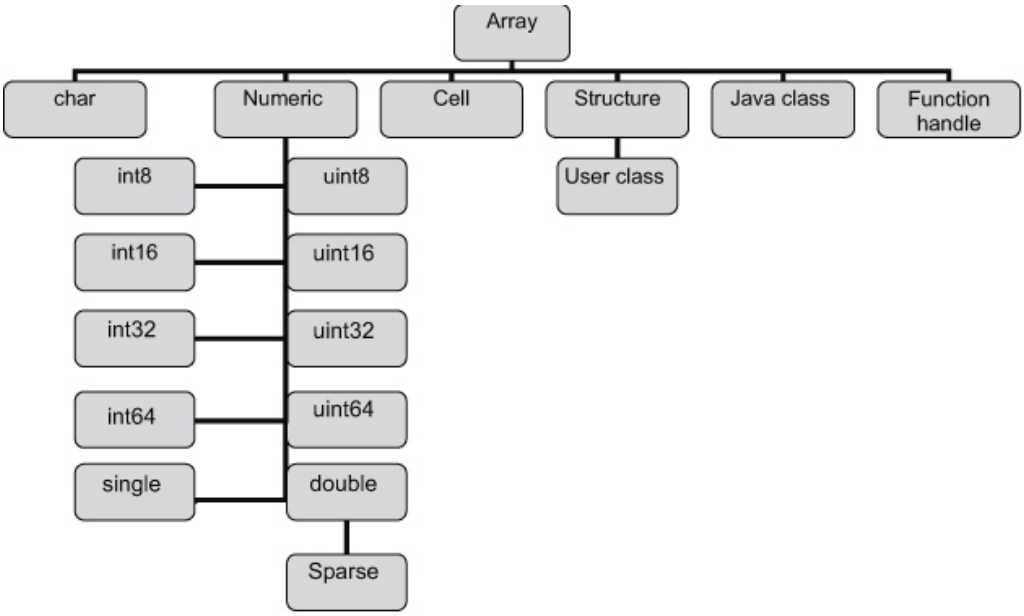
res = -9050/97

Nəhayət ki, əgər operandlar və hesablama nəticələri tamdırlarsa da onlar maşının yaddaşında kəsr ədəd kimi təsəvvür edilir. MATLAB-ın əmrlər pəncərəsində onlar vizual tam ədəd şəklində görünürlər. Bu aşağıdakı şəkildə izah edilir. Əgər hesablanan ifadə başqa bir adlı hər hansı dəyişənə mənimsədilməyibsə, nəticəyə xüsusi **ans** adı verilir (bu standart işarələmədir).



1.4 Matlabda verilənlərin tipi

Matlabda 14 tip əsas verilənlərin vardır. Massivlərin formalaşdırlmasında hər bu tipdən istifadə etmək olar. Aşağıdakı cədvəldə massiv və onda istifadə olunan tiplər göstərilmişdir.



1.5 Həqiqi ədədlər üzərində əməllər

Həqiqi ədədlər üzərində toplama, çıxma, vurma və bölmə hesab əməlləri aparılır və bunlar üçün **+, –, * və /** işarələrindən istifadə edilir.

Arithmetic operation	Symbol	Example
Addition	+	6 + 3 = 9
Subtraction	–	6 – 3 = 3
Multiplication	*	6 * 3 = 18
Right division	/	6/3 = 2
Left division	\	6\3 = 3/6 = 1/2
Exponentiation	^	6 ^ 3 = 6^3 = 216

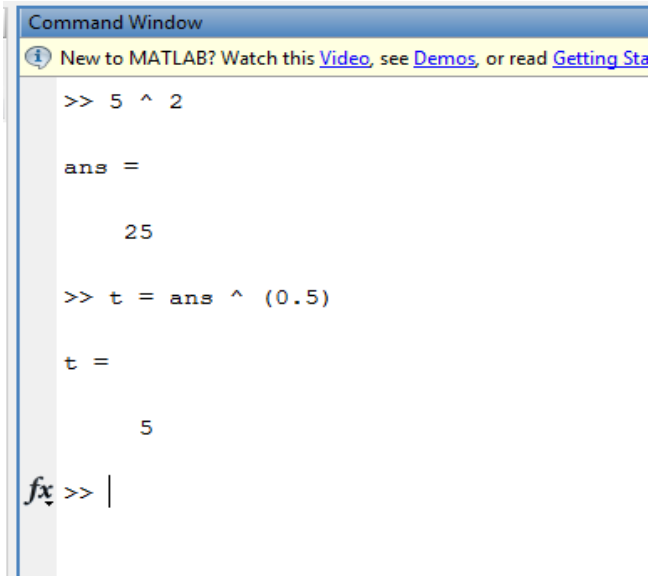
Bundan başqa ^ işarəsi ilə işarələnmiş qüvvətə yüksəltmə əməli var. Bu əməllərin tətbiqinin nəticələri aşağıda göstərilmişdir:

5 ^ 2

ans = 25

t=5

t = ans ^ (0.5)



Hesab əməlləri yerinə yetirilərkən birincilik adidir: əvvəlcə - qüvvətə yüksəltmə, sonra - toplama və çıxma. Birinciliyi eyni olan əməllər növbə ilə soldan sağa yerinə yetirilir, amma dairəvi mötərizələr bu növbəni dəyişə bilər. Hesab əməllərinin yerinə yetrilməsində üstünlük dərəcəsi aşağıdakı cədvəldə verilmişdir,

Precedence	Operation
1 (highest)	Parentheses (If nested parentheses exist, inner have precedence).
2	Exponentiation.
3	Logical NOT (~).
4	Multiplication, Division.
5	Addition, Subtraction.
6	Relational operators (>, <, >=, <=, = =, ~=).
7	Logical AND (&).
8 (lowest)	Logical OR (|).

Hesab əməllərindən asılı olmayaraq başqa münasibət əməli və məntiqi əməllərdən istifadə edilir. Münasibət əməli iki operandı kəmiyyətə görə öz aralarında müqayisə edir. Bu əməllər işarələr və ya işarələr kombinasiyası ilə aşağıdakı kimi yazılır

<	Kiçikdir
<=	Kiçikdir və ya bərabərdir
>	Böyükdür
>=	Böyükdür və ya bərabərdir
==	Bərabərdir
~=	Bərabər deyil

```
>> res=(a<b)+(c~=b)+(b==a)

res =

     2
```

Burada **a** ifadəsi dəyişən kəmiyyətin həqiqətən **b** dəyişən kəmiyyətlərdən (həqiqətən) kiçik olduğu üçün vahid hasil edir.

c ~ = b ifadəsi həqiqidir, belə ki, doğrudan da **c** ifadəsi **3**-ə bərabərdir, lakin ikiyə bərabər olan **b**-yə bərabər deyil. Son olaraq o, **1** qiymətini hasil edir. Sonuncu ifadədə, **b == a**, həqiqi deyildir və **0** hasil edir. Nəticədə **res** dəyişəni, bu üç ifadənin qiymətləri cəminə bərabər olaraq ikiyə bərabər edilir.

Münasibət əməlləri hesab əməllərinə nisbətən daha aşağı birinciliyə malikdir. Buna görə də yuxarıda baxılan **res** dəyişəni ancaq üç münasibət əməlinin qiymətləri cəminə bərabərdir, ona görə ki, biz bu əməlləri dairəvi mövtərizələrə almışıq. Bunu həmişə yadda saxlamaq çox vacibdir, belə ki, dairəvi mötərizələrin olmaması nəticənin dəyişməsinə gətirə bilər. Məsələn,əgər **a=1**, **b=1**, **c=3**, olarsa, onda

c + (b == a)

ifadəsi **4**-ə bərabərdir, bu zaman dairəvi mötərizəsiz olan

c + b == a

ifadəsi **0**-a bərabərdir.

Indi MATLAB sistemində nöqtəli vergülün **(;)** roluna diqqət verək. Nöqtəli vergül müxtəlif məqsədlər üçün istifadə edilə bilər. Biz klaviatura ilə hər hansı ifadəni daxil

etdikdə (0>> - dəvət işarəsindən sonra yerləşir) və **ENTER** düyməsini basdıqda MATLAB bu ifadənin hazırlanmasını həyata keçirir və nəticəni özünün əmrlər pəncərəsinə çıxarır. Əgər biz həmin an hesablamaların nəticəsini görmək istəmiriksə (məsələn, bu aralıq nəticələr üçün xarakterikdir), onda daxil edilən ifadənin sonunda nöqtəli vergül qoymaq və ancaq bundan sonra **ENTER**-i basmaq lazımdır.

Bundan başqa, əgər biz istəsək ki, bir dəfəyə, daha doğrusu **ENTER** düyməsini bir dəfə basmaqla bir neçə müxtəlif ifadəni hesablayaq, onların qiymətlərini müxtəlif dəyişənlərə mənimsədək, onda bu ifadələri bir-birindən nöqtəli vergüllə ayırmaq lazımdır. Bu isə qabaqdakı şəkildə **MATLAB** sisteminin əmrlər pəncərəsində göstərilmişdir. Sonuncu qrup əməllər növbəti cədvəldə verilən məntiqi əməllərdir:

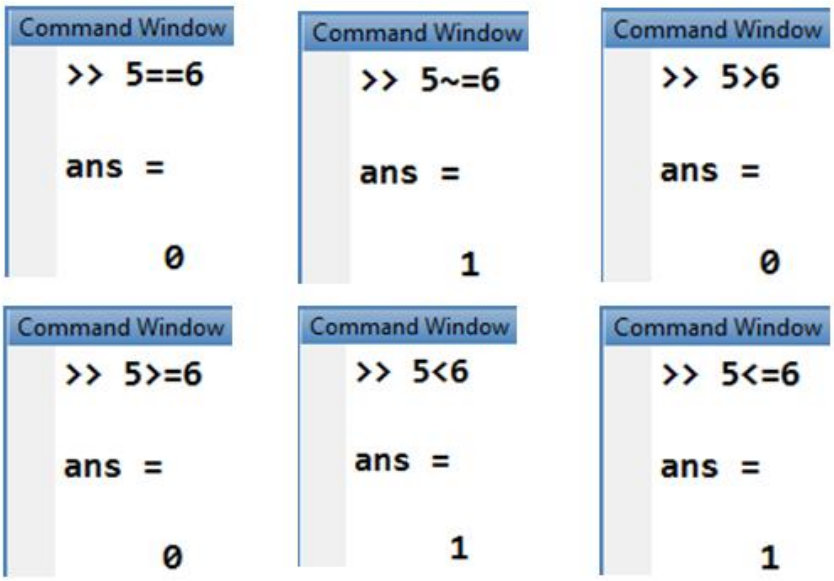
&	**VƏ (AND)**
|	**VƏ YA (OR)**
~	**Yox (NOT)**

Məntiqi əməllər öz operandlarını "*həqiqi*" (sıfıra bərabər olmayan) və ya "*yalan*" (sıfıra bərabər) kimi şərh edirlər. Əgər "və" əməlinin hər iki operandı həqiqidirsə (sıfra bərabər deyilsə), onda bu əməlin nəticəsi "həqiqi"-yə bərabərdir; qalan bütün hallarda "və" əməli (yalan) qiyməti hasil edir. "Və ya" əməli operandların hər ikisi də yalan (sıfıra bərabər) olduğu halda, ancaq (yalan) hasil edir. Nəhayət, "yox" əməli "Yalan"ı "həqiqi"yə və tərsinə çevirir. Daha doğrusu, əgər onun operandı sıfır olmayan ədəddirsə, onda bu əməl **1** hasil edir, amma əgər operanda sıfırdırsa, bu zaman "yox" əməlinin tətbiqinin nəticəsi vahid olacaqdır. Məntiqi əməllər çox aşağı üstünlük dərəcəsinə malikdirlər.

Eyni bir ifadədə bütün sadalanan əməllərdən istifadə etmək olar: hesab, məntiqi və müqayisə əməlləri. Əməllərin yerinə yetirilmə ardıcıllığı onların ifadənin daxilində yerləşməsi, onların üsütünlük dərcəsi və dairəvi mötərizələrin nəzərə alınması ilə müəyyənləşdirilir. Məntiqi əməllərin yerinə yetirilməsinə aid misllara baxalım:

```
5==6   →   0
5~=6   →   1
5>6    →   0
5>=6   →   0
5<6    →   1
5<=6   →   1
```

Bunu Matlabın əmrlər sətrində belə yerinə yetirmək olar:



1.6 MATLAB sisteminin işçi fəzası və onun əmrlər pəncərəsi

Nə vaxt ki, siz MATLAB-ı işə salırsınız və hesablama əməllərini yerinə yetirməyə başlayırsınız, əmrlər pəncərəsində klaviatura ilə daxil etdiyiniz ədəd, dəyişən (onların adı ilə), hesablamaların nəticələri görünür. Adətən, hesablama yenidən və yenidən təkrarlanır: klaviatura ilə yeni ədədi verilənlər və yeni simvollu ifadələr daxil edilir. Nəticədə, MATLAB-ın pəncərəsində sərbəst yer çatışmır və ("skrollirovanie") ("protəcka", "prokrutka") əmələ gəlir - bütün sətirlər bir səviyyə yuxarı yerini dəyişir. Bu zaman ən yuxarıda yerləşən sətir görünən oblastı tərk edir, amma pəncərənin ən aşağısında yeni verilənləri daxil etmək üçün sərbəst sətir görünür. Əlbəttə, bu sətir >> - dəvət işarəsini saxlayır.

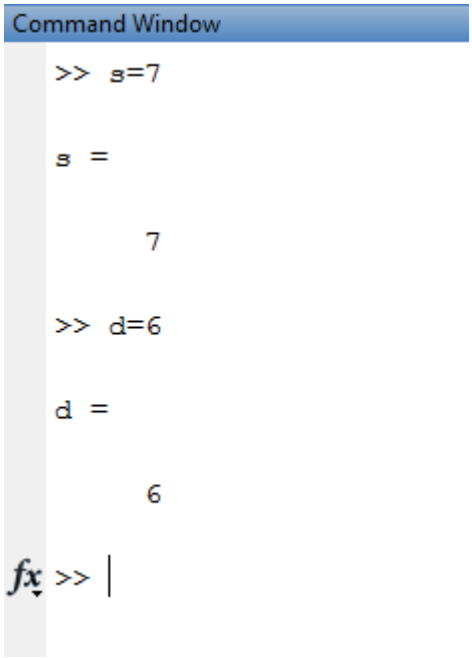
Pəncərənin görünən hissəsini tərk edən məlumatlar heç yana yox olmur. Ona, həmişə, standart qrafiki idarəetmə aləti - fırlatma zolağı (ingiliscə - **Scrollbar**) ilə pəncərənin içərisindəkiləri fırlatmaqla həyata keçirməklə yenidən baxmaq olar. Bunun üçün siçan ilə bu zolağın üstünə sıqqıldatmaq lazımdır və ya siçanın köməyi ilə fırlatma zolağının sürüngəcini lazımi istiqamətdə (yuxarı və ya aşağı) sürümək (dartmaq) lazımdır.

MATLAB sisteminin əmrlər pəncərəsinin, həmçinin içərisindəkilərin fırladılmasını klaviaturanın növbəti düymələrinin köməyi ilə həyata keçirmək olar: **Page Up, Page Down, Ctrl+Home** (**Ctrl** və **Home** düymələrini eyni zamanda basmaqla) və **Ctrl+End**.

"*Yuxarı ox*" və "*Aşağı ox*" düymələri ixtiyari mətn redaktorunda kursorun aşağı-yuxarı yerdəyişməsi və pəncərənin içindəkiləri çevirmək MATLAB sistemdə başqa cür işləyir. Bu düymələr klaviatura ilə əvvəl daxil edilmiş əmrləri daxiletmə sətrinə qaytarmağa və başqa giriş məlumatı daxil etməyə imkan verir. Daha doğrusu bütün bu məlumatlar yaddaşın xüsusi oblastında yadda saxlanılır. Yaddaşın bu oblastını *əmrlər qamçısı* adlandırırlar. Belə ki, son daxil edilən məlumat "Yuxarı ox" düyməsi ilə onu çevirən zaman birinci olaraq üzə çıxır. Sonra sonuncudan əvvəlki əmr üzə çıxır və belə davam edir. "Aşağı ox" düyməsi

əmrləri əks istiqamətdə çevrilməsini həyata keçirir.

Ümumiləşdirərək demək olar ki, MATLAB sisteminin pəncərəsində görünən bütün məlumat prinsipal fərqlənən iki müxtəlif zonada yerləşir: *baxılma zonası* və *redaktə edilmə zonası*.



Gözdən keçirmə zonasında kursoru onun üstünə qoysan da artıq heç nəyi düzəltmə olmaz, amma klaviaturadan daxil etməyə reaksiya, kursorun redaktə zonasında yerləşən giriş sətrinə (daha doğrusu giriş nöqtəyə) avtomatik yerdəyişməsi olacaqdır. Siçanın köməyi ilə baxılma zonasından ixtiyari məlumatı ayırmaq və onu Windows əməliyyatlar sisteminin dəyişmə buferinə (daha doğrusu **Clipboard**-a) köçürmək, sonra isə onu ya redaktə mətin sənədinə (məsələn, Word redaktəsinə), ya da yenə də giriş sətrinə qoymaq olar.

Redaktə zonası adətən **MATLAB** sisteminin əmrlər pəncərəsində >> - dəvət işarəsi göstərilən bir (sonuncu) sətri tutur. Onu giriş sətri adlandırırlar. Amma, lazım olduqda bu məntiqi "sətri" **MATLAB**-ın əmrlər pəncərəsində bir neçə fiziki sətrə paylamaq olar. Bunun üçün sadəcə olaraq **ENTER** düyməsini basmaq olmaz, belə ki, bu zaman məlumatın daxil edilməsi qurtaracaqdır və **MATLAB** hesablamağa və nəticələri göstərməyə başlayacaqdır. Buna görə də görünərək daxil edilməni davam etdirmək üçün

```
>> a=4+5+5

a =

    14
```

şəklində göstərilən kimi daxil edilən məlumatdan növbəti fiziki sətrə keçmək üçün üç və daha çox nöqtə qoyulduqdan sonra **ENTER** düyməsini basmaq lazımdır.

Amma bu halda da redaktə zonası, ancaq ən sonuncu sətirdə paylanır (bu halda o, >> - dəvət işarəsini saxlamır), amma əvvəlki məntiqi giriş sətirlərinin fiziki sətirlərində heç nəyi dəyişmək olmaz. Məntiqi giriş sətri 256 simvoldan çox saxlaya bilmir.

MATLAB sisteminin cari seansındakı işi müddətində hesablanan bütün dəyişənlərin

qiymətləri kompüterin yaddaşında ***işçi fəza*** adlanan xüsusi oblastda saxlanılır (ingiliscə adı - **Workspace**).

clc

əmri ilə MATLAB sistemi pəncərəsi içərisindəki bütün görünənləri silmək olar, amma bu işçi fəzanın içindəkilərə toxunmur. Həqiqətən də, əgər bundan sonra əvvəl yığılan **a** dəyişəninin adını yığsaq, onda ENTER düyməsi basıldıqdan sonra biz yenidən onun qiymətini görərik:

```
>>a
a = 20
```

MATLAB bütün əvvəlki nəticələri (həmçinin əmrləri) avtomatik olaraq yadda saxlaması çox əlverişlidir. Amma burada əgər yadda saxlanılan məlumatların həcmi olduqca böyük olarsa, onda xoşagəlməzlik də aşkar ola bilər. (Sonralar biz göstərəcəyik ki, MATLAB nəhəng ölçülü verilənlərlə işləyə bilir). Əgər sizə verilən iş seansında bəzi dəyişənləri saxlamaq daha tələb edilmirsə, onları **ad1** və **ad2** adlı dəyişənləri işçi fəzadan silən

clear ad1 ad2 …

əmri ilə maşının yaddaşından silmək olar. Bütün dəyişənləri bir dəfəyə silməkdən ötrü

clear

əmrindən istifadə etmək lazımdır.

Əgər siz işçi fəzada hansı dəyişənlərin qaldığını bilmirsinizsə və ya şübhələnirsinizsə (tərəddüd edirsinizsə), siz həmişə

who

əmrini yerinə yetirə bilərsiniz. Belə ki, bu əmr MATLAB sisteminin işçi fəzasına daxil olan bütün dəyişənlərin siyahısını həmin an üzə çıxaracaqdır (növbəti şəklə bax).

```
>> who

Your variables are:

a    ans  b    c    d    res  s    t

fx >>
```

Bu Işçi fəzadan ixtiyari dəyişənin qiymətinə baxmaq üçün onun adını yığmaq və ENTER düyməsini basmaq kifayətdir:

```
>> a

a =

     1

>> b

b =

     2

>>
```

MATLAB-la iş seansını bağladıqda cari seans zamanı hesablanan bütün dəyişənlər itir. Bu dəyişənləri digər növbəti seanslarda istifadə etmək üçün saxlamaq olar, bunun üçün Işçi fəzanın içindəkiləri menyu komandası vasitəsilə fayl şəklində diskdə yadda saxlamaq lazımdır:

File | Save Workspace As…

Bundan sonra diskdə qovluğun və faylın adının seçilməsi üçün Windows əməliyyat sisteminin standart dialoq pəncərəsi üzə çıxır. Faylın genişlənmə adı hökmən **.mat** olmalıdır. Belə faylları MAT-fayllar adlandıracayıq. Menyu əmri əvəzinə MATLAB-ın əmrlər pəncərəsində bilavasitə

save fayla_yol\MAT-faylın_adı

əmrini yığmaq olar.

MATLAB sisteminin yeni seansında menyu əmri ilə diskin Işçi fəzasında əvvəlcədən saxlanılanları bərpa edə bilərsiniz:

File | Load Workspace…

Bundan sonra standart dialoq pəncərəsində lazım olan MAT-faylı göstərilməlidir.

Bundan əlavə, bu əmri bir neçə dəfə müxtəlif fayllarla yerinə yetirib MATLAB sisteminin bu işçi fəzasına bir neçə əvvəlki iş seanslarının içindəkilərini birləşdirə bilərik. Amma, əgər müxtəlif olan seanslardan dəyişənlərin adları üst-üstə düşərsə, onda bu işçi fəzada ancaq sonuncu açılan MAT-fayldan dəyişənlər veriləcəkdir. Menyu əmri əvəzinə MATLAB-ın əmrlər pəncərəsində bilavasitə

load MAT-faylın_adı

əmrini yığmaq olar.

Həmçinin MAT-fayl diskində yazılanlardan İşçi fəzada müxtəlif dəyişənlərin qiymətlərini hesablamaq olar. Bunun üçün

load MAT-faylın_adı ad1, ad2, ...

Nəticədə göstərilən MAT-fayldan **ad1, ad2** və s. adlı dəyişənlər yaddaşa yüklənəcəkdir. Bu zaman əgər MAT-faylın yolu tam göstərilməyibsə, onda o, mütləq *MATLAB sisteminin cari qovluğundadır*. Cari qovluğu həmişə

cd

əmrinin köməyi ilə bilmək olar.

Bu qovluğu

cd yeni_kataloqa_yol

əmri ilə dəyişmək olar.

Bu və başqa əmrlər haqqında qısa məlumat aşağıdakı cədvəllərdə verilmişdir.

Dəyişənlərin idarəedilməsi üzrə əmrlər

Command	Description
clear	Removes all variables from the memory.
clear x, y, z	Clears/removes only variables **x, y** and **z** from the memory.
who	Lists the variables currently in the workspace.
whos	Displays a list of the variables currently in the memory and their size together with information about their bytes and class.

On-line yardım üçün istifadə edilən əmrlər

Function	Description
help	Lists topics on which help is available.
helpwin	Opens the interactive help window.
helpdesk	Opens the web browser based help facility.
help *topic*	Provides help on *topic*.
lookfor *string*	Lists help topics containing *string*.
demo	Runs the demo program.

Workspace-dən alınan informasiya

Function	Description
who	Lists variables currently in the workspace.
whos	Lists variables currently in the workspace with their size.
what	Lists m-, mat- and mex-files on the disk.
clear	Clears the workspace, all variables are removed.
clear *x y z*	Clears only variables *x*, *y*, and *z*.
clear all	Clears all variables and functions from workspace.
mlock *fun*	Locks function *fun* so that **clear** cannot remove it.
munlock *fun*	Unlocks function *fun* so that **clear** can remove it.
clc	Clears command window, command history is lost.
home	Same as **clc**.
clf	Clears figure window.

Qovluqlar haqqında məlumat

Function	Description
pwd	Shows the current working directory.
cd	Changes the current working directory.
dir	Lists contents of the current directory.
ls	Lists contents of the current directory, same as **dir**.
path	Gets or sets MATLAB search path.
editpath	Modifies MATLAB search path.
copyfile	Copies a file.
mkdir	Creates a directory.

Ümumi məlumat almaq üçün əmrlər

Function	Description
computer	Tells you the computer type you are using.
clock	Gives you wall clock time and date as a vector.
date	Tells you the date as a string.
more	Controls the paged output according to the screen size.
ver	Gives the license and the version information about MATLAB installed on your computer.
bench	Benchmarks your computer on running MATLAB compared to other computers.

Proqramı dayandıran əmrlər

Function	Description
c (Control-c)	Local abort, kills the current command execution.
quit	Quits MATLAB.
exit	Same as **quit**.

Çalışmalar:

1. Ifadəni hesablayın:

```
1 + 2 * 3
4 / 2 * 2
1+2 / 4
1 + 2\4
2*2 ^ 3
2 * 3 \ 3
2 ^ (1 + 2)/3
1/2e-1
```

2.

(a) $\frac{1}{2 \times 3}$ (0.1667)

(b) $2^{2\times3}$ (64)

(c) $1.5 \times 10^{-4} + 2.5 \times 10^{-2}$

1.6.1 Öncədən təyin olunmuş dəyişənlər

Matlab proqram paketində öncədən təyin olunmuş dəyişənlər var ki, oları bir çox hesablamalarda istifadə etmək olar. Bu dəyişənlər aşağıdakı cədvəldə verilmişdir.

Predefined variable in MATLAB	Description
ans	Represents a value computed by an expression but not stored in variable name.
pi	Represents the number π.
eps	Represents the floating-point precision for the computer being used. This is the smallest difference between two numbers.
inf	Represents infinity which for instance occurs as a result of a division by zero. A warning message will be displayed or the value will be printed as ∞.
i	Defined as $\sqrt{-1}$, which is: $0 + 1.0000i$.
j	Same as i.
NaN	Stands for Not a Number. Typically occurs as a result of an expression being undefined, as in the case of division of zero by zero.
clock	Represents the current time in a six-element row vector containing year, month, day, hour, minute, and seconds.
date	Represents the current date in a character string format.

1.6.2 Ədədlərin ekranda əks edilməsi üçün istifadə olunan formatlar

Matlabda ədədləri ekranda təsvir etmək üçün müxtəlif fortmatlar vardir ki, onların hamısını haqqında daha geniş məlumatı **help format** əmri ilə ala bilərsiniz. Aşağıdakı cədvəldə bu formatlar haqqında qısa məlumat verilmişdir.

Formatlar

Command	Description	Example
format short	Fixed-point with 4 decimal digits	>> 351/7 ans = 50.1429
format long	Fixed-point with 14 decimal digits	>> 351/7 ans = 50.14285714285715
format short e	Scientific notation with 4 decimal digits	>> 351/7 ans = 5.0143e + 001
format long e	Scientific notation with 15 decimal digits	>> 351/7 ans = 5.014285714285715e001
format short g	Best of 5 digit fixed or floating point	>> 351/7 ans = 50.143

format long g	Best of 15 digit fixed or floating point	>> 351/7 ans = 50.1428571428571
format bank	Two decimal digits	>> 351/7 ans = 50.14
format compact	Eliminates empty lines to allow more lines with information displayed on the screen	
format loose	Adds empty lines (opposite of compact)	

1.7 Elementar riyazi funksiyalar

MATLAB sistemində bütün əsas elementar funksiyalar iştirak edir: qüvvət, üstlü, triqonometrik və bunların tərsi. Ixtiyari funksiya öz adı ilə, giriş arqumentlərin siyahısı (funksiyanın adından sonra dairəvi mötərizələrin içində durur və vergül ilə sadalanır) və hesablanan (çıxış) qiymətlə xarakterizə olunur.

Function	Description
abs(x)	Computes the absolute value of **x**.
sqrt(x)	Computes the square root of **x**.
round(x)	Rounds **x** to the nearest integer.
fix(x)	Rounds (or truncates) **x** to the nearest integer toward 0.
floor(x)	Rounds **x** to the nearest integer toward $-\infty$.
ceil(x)	Rounds **x** to the nearest integer toward ∞.
sign(x)	Returns a value of −1 if **x** is less than 0, a value of 0 if **x** equals 0, and a value of 1 otherwise.
rem(x,y)	Returns the remainder of **x/y**. for example, **rem(25, 4)** is 1, and **rem(100, 21)** is 16. This function is also called a **modulus** function.
exp(x)	Computes $e^{\mathbf{x}}$, where e is the base for natural logarithms, or approximately 2.718282.
log(x)	Computes ln **x**, the natural logarithm of **x** to the base e.
log10(x)	Computes $\log_{10}$ **x**, the common logarithm of **x** to the base 10.

Ilkin olaraq əvvəl baxdığımız məntiqi əməllər yığımını tamamlayan məntiqi **xor** funksiyasına baxaq. Bu funksiya iki giriş arqumentinə malikdir və onlar üzərində "VƏ YA çıxarmaq" əməlini hesablayır, hansı ki, vahid hasil edir (həqiqi) ancaq o halda ki, ədədi arqumentlərdən biri həqiqidir (sıfıra bərabər deyil), o birisi isə yalandır (sıfıra bərabərdir). Məsələn,

```
>>a=1; b=0;
>>xor(a, b)
ans =    1
```

lakin əgər hər iki arqument "həqiqi" və ya hər ikisi "yalan"-dırsa, onda bu funksiya sıfır hasil edir:

```
>>a=1; b=1;
>>xor(a, b)
```

ans = 0

Matlabda istifadə olunan bu və başqa məntiqi operator rolunu daşıyan məntiqi funksiyalar aşağıda verilmişdir.

and (A, B)	Equivalent to *A* & *B*
or (A, B)	Equivalent to *A* | *B*
not (A)	Equivalent to ~*A*

Məntiqi operatorların doğruluq cədvəli

INPUT		OUTPUT				
A	*B*	AND *A&B*	OR *A|B*	XOR (*A*,*B*)	NOT ~*A*	NOT ~*B*
false	false	false	false	false	true	true
false	true	false	true	true	true	false
true	false	false	true	true	false	true
true	true	true	true	false	false	false

Əlavə məntiqi funksiyalar

Function	Description	Example
xor(a, b)	Exclusive or. Returns true (1) if one operand is true and the other is false.	>>xor(8, –1) ans = 0 >>xor(8, 0) ans = 1
all(A)	Returns 1 (true) if all elements in a vector *A* are true (non-zero). Returns 0 (false) if one or more elements are false (zero). If *A* is a matrix, treats columns of *A* as vectors, returns a vector with 1's and 0's.	>>*A* = [5 3 11 7 8 15] >>all(*A*) ans = 1 >>*B* = [3 6 11 4 0 13] >>all(*B*) ans = 0
any(A)	Returns 1 (true) if any element in a vector *A* is true (non-zero). Returns 0 (false) if all elements are false (zero). If *A* is a matrix, treats columns of *A* as vectors, returns a vector with 1's and 0's.	>>*A* = [5 0 14 0 0 13] >>any(*A*) ans = 1 >>*B* = [0 0 0 0 0 0] >>any(*B*) ans = 0
find(A) **find(A>d)**	If *A* is a vector, returns the indices of the non-zero elements. If *A* is a vector, returns the address of the elements that are larger than *d* (any relational operator can be used).	>>*A* = [0 7 4 2 8 0 0 3 9] >>find(*A*) ans = 2 3 4 5 8 9 >>find(*A* > 4) ans = 4 5 6

MATLAB sistemində ixtiyari elementar funksiya haqqında tez bir zamanda sorğu məlumatı

almaq üçün aşağıdakı əmri yerinə yetirmək lazımdır:

```
help funksiyanın_adı
```

Triqonometrik funksiyalar tam əhatəli göstərilmişdir: **sin**, **cos**, **tan** (tangels), **cot** (kotangels), **asin** (arksinus), **acos** (arkkosinus), **atan** (arktangels), **acot** (arkkotangels). Həmçinin az işlədilən funksiyalar tipindən olan sekans, kosekans və s., hiperbolik funksiyalar (eksopentanın kombinasiyası) var.

Function	Description
sin(x)	Computes the sine of **x**, where **x** is in radians.
cos(x)	Computes the cosine of **x**, where **x** is in radians.
tan(x)	Computes the tangent of **x**, where **x** is in radians.
asin(x)	Computes the arcsine or inverse sine of **x**, where **x** must be between –1 and 1. The function returns an angle in radians between $-\pi/2$ and $\pi/2$.
acos(x)	Computes the arccosine or inverse cosine of **x**, where **x** must be between –1 and 1. The function returns an angle in radians between 0 and π.
atan(x)	Computes the arctangent or inverse tangent of **x**. The function returns an angle in radians between $-\pi/2$ and $\pi/2$.
atan2(y,x)	Computes the arctangent or inverse tangent of the value y/x. The function returns an angle in radians that will be between $-\pi$ and π, depending on the signs of **x** and **y**.
sinh(x)	Computes the hyperbolic sine of **x**, which is equal to $\frac{e^{x}-e^{-x}}{2}$.
cosh(x)	Computes the hyperbolic cosine of **x**, which is equal to $\frac{e^{x}+e^{-x}}{2}$.
tanh(x)	Computes the hyperbolic tangent of **x**, which is equal to $\frac{\sinh x}{\cosh x}$.
asinh(x)	Computes the inverse hyperbolic sine of **x**, which is equal to $\ln\left(x+\sqrt{x^2+1}\right)$.
acosh(x)	Computes the inverse hyperbolic cosine of **x**, which is equal to $\ln\left(x+\sqrt{x^2-1}\right)$.
atanh(x)	Computes the inverse hyperbolic tangent of **x**, which is equal to $\ln\sqrt{\frac{1+x}{1-x}}$ for $\lvert x\rvert \le 1$.

Nümunə üçün **asin** funksiyasının hesablanması daxil olan **2*asin(1)** ifadəsini hesablasaq aşağıdakı

```
ans =    3.1416
```

nəticəsini alarıq ki, o, π ədədinə uyğundur. MATLAB sistemində π ədədi üçün xüsusi **pi**

işarələməsi var.

Tamqiymətli hesabla əlaqədə olan başqa funksiyaları yada salaq. Məsələn, *yuvarlaqlaşdırma funksiyası*: **round** (yaxın tam ədədə qədər yuvarlaqlaşdırma), **fix** (ədədin kəsr hissəsinin kəsiyi), **floor** (ən kiçik tama qədər yuvarlaqlaşdırma), **ceil** (ən böyük tama qədər yuvaqlaqlaşdırma).

Function	Description	Example
round(x)	Round to the nearest integer	>> round(20/6) ans = 3
fix(x)	Round towards zero	>> fix(13/6) ans = 2
ceil(x)	Round towards infinity	>> ceil(13/5) ans = 3
floor(x)	Round towards minus infinity	>> floor(–10/4) ans = –3
rem(x,y)	Returns the remainder after **x** is divided by **y**	>> rem(14,3) ans = 2
sign(x,y)	Signum function. Returns 1 if **x** > 0, –1 if **x** < 0, and 0 if **x** = 0.	>> sign(7) ans = 1

Bundan əlavə başqa funksiyalar var: **mod** (işarəni nəzərə almaqla bölmədən qalıq), **rem** (modul hesabı mənasında qalıq), **sign** (ədədin işarəsi), **factor** (ədədin sadə vuruqlara ayrılması), **isprime** (həqiqi, əgər ədəd sadədirsə), **primes** (sadə ədədlərin siyahısını yaratmaq), **rat** (ədədin rasional kəsr şəklində yaxınlaşması), **lcm** (ən kiçik ortaq bölünən), **gcd** (ən böyük ortaq bölən).

mod və **rem** fuksiyaları müsbət arqumentlər üçün eyni nəticəni verir. Xüsusi halda

mod(7,2) == rem(7,2) == 1

Lakin müxtəlif işarəli arqumentlərlə əməllər üçün onlar müxtəlif qiymətlər hasil edir:

mod(-7,2) = 1; rem(-7,2) = -1

Ümumi halda bu funksiyalar yuvarlaqlaşdırma funksiyaları ilə aşağıdakı şəkildə bağlıdır:

rem(x,y) == x—y*fix(x/y)

mod(x,y) == x—y*floor(x/y)

Və nəhayət, *kombinatorikadan* bəzi standart nəticələri hesablayan **perms** funksiyası yerdəyişmələrin sayını, **nchoosek** funksiyası birləşmələrin sayını hesablayır. Məsələn, 10-dan 3-ə görə birləşmələrin sayı aşağıdakı funksiyalarının çağrılması ilə çox asan tapılır:

```
Command Window
>> nchoosek(10,3)

ans =

   120
```

Sadalanan funksiyalardan çoxu bütün **R** həqiqi ədədlər çoxluğundan fərqli təyin oblastına malikdirlər. Funksiya üçün arqumentin mümkün olmayan qiyməti verildiyi zaman və ya mümkün olmayan əməli yerinə yetirmək cəhd edildiyi halda biz xəbərdarlıq məlumatı alırıq, məsələn, sıfıra bölməyə cəhd edildikdə

Warning: Divide by zero.

xəbərdarlıq sətri görünür, amma nəticə kimi

ans = Inf

məlumatı çıxarılır, burada **Inf** sonsuzluğu simvollaşdırır. Həmin bu nəticə sıfırın loqarifmini hesablamaq cəhdi zamanı da alınır.

Hərçənd ki, çox zaman həqiqi dəyişənli funksiyanın təyin oblastından çıxan arqumenti verilən zaman (məsələn, kvadrat kök üçün mənfi arqument) MATLAB sistemi avtomatik olaraq kompleks ədədlər oblastına çıxır və kompleks funksiyaya analoji qiyməti hesablayır.

1.8 MATLAB sistemində kompleks hesablamalar

sqrt(-1) funksiyası hesablanan zaman kompleks ədəd alınır:

```
>> sqrt(-1)

ans =

        0 + 1.0000i
```

Şəkildən görünür ki, MATLAB sistemində xəyali vahid riyazi ədəbiyyatlar üçün ənənəvi olan **i** hərfi ilə işarə olunur. Bununla yanaşı xəyali vahid üçün MATLAB-da **j** hərfi də ehtiyatda saxlanılır. Yaxşı olar ki, özünüzü çaşdırmamaq məqsədilə başqa məqsədlər üçün bu hərflərdən istifadə etməyəsiniz.

Dəyişən olan halda sadəcə olaraq **x+iy** yazmaq olmaz, mütləq vurma işarəsindən istifadə etmək lazımdır, yəni **x+i*y.**

Kompleks ədələr funksiyaları

Function	Description
conj(x)	Computes the complex **conjugate** of the complex number **x**. Thus, if **x** is equal to $a + ib$, then **conj(x)** will be equal to $a - ib$.
angle(x)	Computes the real portion of the complex number **x**.
real(x)	Computes the imaginary portion of the complex number **x**.
imag(x)	Computes the absolute value of **magnitude** of the complex number **x**.
abs(x)	Computes the angle using the value of **atan2(imag(x), real(x))**; thus, the angle value is between $-\pi$ and π.

Kompleks ədədlər üzərində hesab əməllər

Operation	Result
$c_1 + c_2$	$(a_1 + a_2) + i(b_1 + b_2)$
$c_1 + c_2$	$(a_1 - a_2) + i(b_1 - b_2)$
$c_1 \bullet c_2$	$(a_1a_2 - b_1b_2) + i(a_1b_2 - a_2b_1)$
$\frac{c_1}{c_2}$	$\frac{a_1a_2}{a_2^2}\ \frac{b_1b_2}{b_2^2}\quad i\ \frac{a_2b_1}{a_2^2}\ \frac{b_2a_1}{b_2^2}$
$\lvert c_1\rvert$	$\sqrt{a_1^2\ \ b_1^2}$ (magnitude or absolute value of c_1)
c_1^*	$a_1 - ib_1$ (conjugate of c_1)
(Assume that $c_1 = a_1 + ib_1$ and $c_2 = a_2 + ib_2$.)	

Demək olar ki, bütün elementar funksiyalar kompleks arqumentlə hesablamalar aparmağa imkan verir. Məsələn:

res = sin(2+3i)*atan(4i)/(1 — 6i)

res= -1.8009 — 1.9190i

Indi biz məşhur *Eyler düsturunu* yoxlaya bilərik:

exp(i*x) = cos(x)+i*sin(x)

x həqiqi dəyişəninə müxtəlif qiymətlər verərək bu düsturun sağ və sol tərəfindəki ifadələri hesablayaq. Məsələn **x=1** olduqda həm sağ və həm də sol tərəf üçün eyni qiymətləri alırıq: **0.5403+0.8415i**. **x=2** üçün bu bərabərliyin hər iki tərəfi yenidən eyni nəticəni verir: **-0.4161+0.9093i** və belə davam edir.

Amma bəzi hallarda kompleks arqumentlər haqqında danışmaq belə mümkün deyil:

```
>> mod(7,2i)
??? Error using ==> mod
Arguments must be real.
```

Kompleks ədədlərlə işləmək üçün xüsusi olaraq aşağıdakı funksiyalar nəzərdə tutulmuşdur: **abs** (kompleks ədədin mütləq qiyməti), **conj** (kompleks qoşma ədəd), **imag** (kompleks ədədin xəyali hissəsi), **real** (kompleks ədədin həqiqi hissəsi), **angle** (kompleks ədədin arqumenti), **isreal** (həqiqi, əgər ədəd həqiqidirsə).

Qeyd edək ki, kompleks işarəli dəyişən və kompleks funksiyaların adları haqqında heç bir xüsusi razılaşma (mövcud deyil) yoxdur. Dəyişənlər əvvəlcədən heç bir yazılış tələb etmirlər. Bütün hesablamalar həqiqi oblastdan kompleks oblasta avtomatik olaraq ötürülür. Bu kompleks operandlar verilən zaman (kompleks vahid üçün ehtiyatda saxlanılan **i** və ya **j** adlarından istifadə olunur) və ya ancaq həqiqi hesablamalarla kifayətlənmək mümkün olmadığı halda (**sqrt(-1)** funksiyası hesablanılan haldakı kimi) baş verir.

Beləliklə, MATLAB sistemi istifadəçi üçün şəffaf şəkildə həqiqi ədədlərlə hesablamalardan kompleks ədədlərlə hesablamalara keçə bilir. Bundan başqa, MATLAB qarışıq həqiqi (və ya kompleks) ədədlərlə hesablamaları ayrı-ayrı ədədlərdə olduğu kimi apara bilər. MATLAB sisteminin ən xarakteristik xüsusiyyətlərindən biri bundan ibarətdir.

1.9 MATLAB sistemində ədədi massivlər

Proqramlaşdırmada qarışıq ədədləri ***massiv*** adlandırmaq qəbul edilmişdir. Bütöv massivə bir ad verilir, lakin massivin ayrı-ayrı elementlərinə müraciət tam ədədli indekslə, daha doğrusu, massiv elementinin nömrəsi ilə həyata keçirilir.

Massivlər bir indeksdən (nömrə) istifadə olunduqda ***birölçülü*** və digər hallarda ***çoxölçülü*** (xüsusi halda, ***ikiölçülü***) olurlar.

Əvvəlcə birölçülü massivlərə baxaq. Bu xətti qarışıq ədədlər (elementlər) içərisində hər bir elementin mövqeyi vahid ədədlə - onun nömrəsi ilə verilir. Massivin birinci elementi, ikinci elementi və s. demək olar.

Bir neçə ədədlərdən ibarət (həqiqi və ya kompleks) birölçülü massivin verilməsi üçün kvadrat mötərizələrin **[ ]** köməyi ilə işarə olunan ***konkatenasiya əməlindən*** istifadə olunur. Məsələn, növbəti ifadə

```
>> a1 = [ 1 2 3 ]
```

üç elementdən (həqiqi ədədlər) birölçülü **a1** massiv adlanan dəyişəni formalaşdırır. Massivdə birləşdirilən elementlər bir-birindən ya boşluqla, ya da vergüllə ayrılırlar. Buna görə də,

```
>> a1 = [ 1, 2, 3 ]
```

ifadəsi əvvəlki ifadə ilə eynidir.

Birölçülü massivin fərdi elementinə müraciət etmək üçün onun adından sonra dairəvi mötərizədə bu elementin indeksini (nömrəsini) göstərmək lazımdır. Məsələn, **a1** massivinin üçüncü elementi **a1(3)** kimi işarə olunur, birinci element - **a1(1)** kimi, ikinci element - **a1(2)** kimi.

Əgər yuxarıda konkatenasiya əməli ilə formalaşdırılmış **a1** massivinin üçüncü elementini dəyişmək tələb olunursa, onda mənimsəmə əməlini tətbiq etmək olar:

>>a1(3) = 789

Tutaq ki, nümunədə **a1** massivinin ikinci elementi birinci və üçüncü elementlərinin ədədi ortasına bərabər olmalıdır. Bunun üçün növbəti əməli yerinə yetirək:

a1(2) = (a1(1)+a1(3))/2

Birölçülü massivdəki elementlərin sayını (miqdarını) **length** funksiyası vasitəsilə təyin etmək olar:

>>length(a1)

ans = 3

Mövcud olmayan elementin oxunması cəhdi zamanı (məsələn, **a1** massivinin dördüncü elementi) MATLAB-ın əmrlər pəncərəsində səhv haqqında məlumat üzə çıxır:

```
>> a(4)
??? Undefined function or method 'a' for input arguments of
type 'double'.

>> dd(4)
??? Undefined function or method 'dd' for input arguments of
type 'double'.
```

Bu məlumatda təsdiq edilir ki, indeks massivin ölçüsündən kənara çıxır. Həmçinin bu zaman mövcud olmayan elementin yazılmasının tam mümkünlüyü – o, təzə elementin artıq mövcud olan massivə əlavə olunduğunu göstərir:

>> a1(4) = 7

a1 massivinə **length** funksiyasını tətbiq edərək bu massivdəki elementlərinin sayının (miqdarının) dördə qədər artdığını alırıq:

>> length(a1)

ans = 4

Eyni əməli, yəni "**a1** massivinin uzadılması" əməlini konkatenasiya əməlinin köməyi ilə də yerinə yetirmək olar:

>> a1 = [a1 7]

Burada konkatenasiya əməlləri operandı üç elementdən təşkil olunmuş və ona **7**-yə bərabər dördüncü element əlavə olunmuş **a1** massividir.

Indi digər birölçülü **a2** massivi yaradaq, belə ki, onun yaradılması üçün konkatenasiya əməlindən istifadə etməyəcəyik (biz yuxarıda etdiyimiz kimi). Bunun əvəzinə yaradılan massivin hər bir elementini ayrıca yazacayıq:

>>a2(1) = 67

>>a2(2) = 7.8

>>a2(3) = 0.017

Mövcud olan iki massivdən — dördelementli **a1** massivi və üç elementli **a2** massivi ilə bir (qrup şəklində) konkatenasiya əməli ilə yeddi elementdən ibarət birölçülü **b** massivi yaratmaq olar:

>>b = [a1, a2]

Massivlər yalnız həqiqi ədədlərdən ibarət olmaya da bilər. Məsələn,

>> d = [1+2i, 2+3i, 3-7i]

ifadəsi kompleks ədədli birölçülü **d** massivini formalaşdırır. Formalaşan birölçülü massivin elementlərini bir-birindən *ayıran* ya boşluq, ya da vergül ola bilər. Ifadələrdə kompleks ədədlərdən istifadəsi zamanı vergüldən istifadə edilməsi daha üstün hesab edilir.

Indi isə *ikiölçülü massivlərə* baxaq. Ikiölçülü massivləri *düzbucaqlı cədvəl* şəklində düzülmüş qarışıq ədədlər kimi şərh etmək olar. Bu halda ixtiyari elementə girmək üçün iki indeksdən - *sətrin* nömrəsi və *sütunun* nömrəsindən (onların kəsişməsində seçilmiş element durur) istifadə olunur.

Ikiölçülü massiv sətirlərin və sütunların sayı (miqdarı) ilə xarakterizə olunur. Iki sütun və üç sətirdən ibarət **a3** massivi tərtib edək:

```
>> a3=[1,2;3,4;5,6]

a3 =

     1     2
     3     4
     5     6
```

Bu şəkildən yaxşı görünür ki, konkatenasiya əməlinin köməyi ilə formalaşan ikiölçülü massivdə *sətir ayırıcıları* kimi *nöqtəli vergül* işlədilmişdir.

Birölçülü massivlərdə olduğu kimi ikiölçülü massivi onun elementlərini təkbaşına yazmaqla yaratmaq olar:

>>a3(1,1) = 1
>>a3(1,2) = 2
>>a3(2,1) = 3
>>a3(2,2) = 4
>>a3(3,1) = 5
>>a3(3,2) = 6

Ikiölçülü massivin ayrı-ayrı elementlərinə müraciət etmək üçün dairəvi mötərizəli ifadədən istifadə olunur və onun indeksləri vergüllə sadalanır. Birinci indeks sətrin nömrəsini, ikinci indeks isə sütunun nömrəsini göstərir.

MATLAB sistemi böyük ölçülü massivlə işləyə bilir. Onlara sonra növbəti fəsildə baxılacaqdır.

Yenidən riyaziyyatda ***matrislər*** adlandırılması qəbul edilmiş ikiölçülü massivlərə qayıdaq. Massivin ixtiyari sətri ***birölçülü massiv*** və matrisin ixtiyari sütunu da, həmçinin, ***birölçülü massiv*** adlanır. Amma matris nöqteyi-nəzərindən onların elementlərinin düzülüşündə bəzi fərqlər var: birinci birölçülü massivin elementləri massivin sətri boyunca (üfüqi), ikincinin elementləri isə - sütunlar boyu (şaquli) düzülmüşdür.

Əgər birölçülü massiv anlayışında bu fərqi aşkar şəkildə nəzərə alsaq, onda birinci tip massivi ***sətir-vektor***, ikinci tip massivi isə ***sütun-vektor*** adlandırırlar. Bu halda, həmçinin, hesab etmək olar ki, sətir-vektor sətirlərinin sayı vahidə bərabər matrisin xüsusi halı, sütun vektor isə sütunlarının sayı vahidə bərabər matrisin xüsusi halıdır.

MATLAB sistemində bütün birölçülü massivlər ya sətir-vektor, ya da sütun-vektor kimi şərh edilir. Bu vaxta qədər biz ancaq sətir-vektoru daxil etmişik. Konkatenasiya əməlindən istifadə edən növbəti ifadə sütun-vektor verir:

>> a4=[1; 2; 3]

Bu sütun-vektor üç sətirdən ibarətdir, belə ki, konkatenasiya əməlində nöqtəli vergül təzə sətirə keçidi göstərir.

a4 massivi üçün **length(a4)** funksiyası **3** ədədini qaytarır, belə ki, həqiqətən bu massiv üç elementdən ibarətdir. **length** funksiyası sətir-vektoru və sütun-vektoru fərqləndirmir.

Əgər MATLAB sistemindən **a4** dəyişəninin qiymətini göstərməyi tələb etsək, onda biz növbəti şəkli görərik:

```
>> a4=[ 1; 2; 3]

a4 =

     1
     2
     3

fx >>
```

Yəni MATLAB bu birölçülü massivin "həndəsəsini" müəyyən edir və öz pəncərəsində elementi şaquli yerləşdirərək onu əyani surətdə əks etdirir.

Sətir-vektorlar və sütun-vektorların düzgün həndəsi əks olunması və həmçinin hər iki istiqamətdə ikiölçülü massivin ölçülərini bilmək üçün **size** funksiyasından istifadə olunur. Ikiölçülü **a3** massivi üçün növbəti nəticə alınır:

>>size(a3)

ans = 3 2

Burada birinci parametr sətirlərin sayı, ikinci isə sütunların sayıdır. Bu funksiyanı birölçülü massivlərə tətbiq edək. **a2** sətir-vektoru üçün bir sətir və üç sütundan ibarət nəticə alınır.

>>size(a2)

ans = 1 3

Üç sətir və bir sütundan ibarət **a4** sütun-vektoru üçün isə **size** funksiyasının tətbiqi ilə

>>size(a4)

ans = 3 1

nəticəsini alırıq. Nəhayət, bu funksiyanı bir ədədi qiymətdən ibarət dəyişənə, yəni skalyara tətbiq edək:

>>var1 = 5

>>size(var1)

ans = 1 1

Buradan görünür ki, MATLAB sistemi hətta mahiyyətcə skalyar olan kəmiyyətləri də **1x1** ölçülü massiv kimi şərh edir. Bu isə öz növbəsində istifadəçi üçün heç nəyi dəyişmir, və

o, bütün bunları nəzərə almaya bilər. MATLAB, istifadəçidən əlavə əməllər tələb etmədən skalyardan şəffaf şəkildə massivlərə keçidi yerinə yetirir.

Beləliklə, MATLAB-ın işlədiyi bütün nə varsa, müxtəlif ölçülü massivlərdir. **whos** əmrinin köməyi ilə cari iş seansının (cari Işçi fəzada - Workspace) bütün massivlərinə struktur nöqteyi-nəzərindən baxmaq olar:

```
>> whos
  Name        Size              Bytes  Class     Attributes

  a3          3x2                  48  double
  a4          3x1                  24  double
  ans         1x1                  16  double    complex

fx >>
```

1.10. Massivlərlə aparılan əməliyyatların sintaksisi

Massivlər MATLAB sistemində diqqəti cəlb edən vacib obyektlər olduğundan massivlərlə aparılan əməliyyatların sintaksisinə daha geniş baxaq.

Indiyə kimi məhz massivlərlə iş üçün nəzərdə tutulmuş iki əməliyyata baxılmışdı: ***konkatenasiya*** əməli (kvadrat mötərizələrdən istifadə olunur) və ***indeksləşdirmə*** əməli (dairəvi mötərizələrdən istifadə olunur).

Qeyd edək ki, konkatenasiya əməlində elementləri ayırmaq üçün vergüllərdən (və ya boşluqlardan) və nöqtəli vergüllərdən istifadə olunur. Birinci halda "üfüqi" və ya sətir boyunca konkatenasiyadan, ikinci halda isə "şaquli" və ya sütun boyunca konkatenasiyadan danışılır. Məsələn, sətirlərin sayı bərabər nəzərdə tutulmuş iki **A1** və **A2** matrislərdən ibarət

>>C = [A1 , A2]

əməli sütunlarının sayı toplanan matrislərin sütunlarının cəminə bərabər olan daha "geniş" **C** matrisi əmələ gətirir,

>>C = [A1 ; A2]

əməli, isə **A1** və **A2** matrislərini şaquli birləşdirərək, **C** matrisini əmələ gətirir, bu matrisin sətirlərinin sayı **A1** və **A2** matrislərin sətirlərinin sayının cəminə bərabərdir.

```
>> A1=[1 2; 3 4]

A1 =

     1     2
     3     4

>> A2=[4 6; 8 9]

A2 =

     4     6
     8     9

>> C=[A1 A2]

C =

     1     2     4     6
     3     4     8     9
```

A1 və **A2** matrislərinin sütunlarının sayı müxtəlif olduqda səhv haqqında aşağıdakı məlumat göstəriləcəkdir.

```
>> A1

A1 =

     1     2
     3     4

>> A2

A2 =

     4     6
     8     9
     9     6

>> C=[A1 A2]
??? Error using ==> horzcat
CAT arguments dimensions are not consistent.
```

Konkatenasiya əməli iri həcmli həqiqi əmələri kompakt şəkildə ifadə etməyə imkan verən güclü qrup əməlidir. Konkatenasiyasız massivin ayrı-ayrı elementlərinə qiymətləri mənimsətmək üçün çox böyük iş görmək lazım gələrdi. Konkatenasiya əməli üçün operand rolunu ədədlər, vektor-sətirlər, vektor-sütunlar, matrislər, həmçinin daha böyük ölçülü

massivlər oynaya bilər. Bu, konkatenasiya əməlini insanın qavraması üçün kifayət qədər mürəkkəbləşdirir, ona görə biz buna daha mürəkkəb misallara baxmaq üçün yenidən qayıdacağıq.

Indeksləşdirmə əməlində isə yeganə ayırıcıdan - vergüldən istifadə olunur. O, massiv elementlərinin qruplaşmalarının müxtəlif istiqamətlərinə uyğun (iki ölçülü massivlərdə - sətir boyunca və sütun boyunca) indeksləri bir-birindən ayırır.

Massivlərə tətbiq edilən daha bir güclü əməllər qrupu mövcuddur. Bu, *ikinöqtə* ilə işarə olunan ***qiymətlər diapazonunu təşkil etmə*** əməlidir. **3.7**-dən (daxil olmaqla) başlayaraq **0.3** addımı ilə (artımı ilə) verilmiş və **8.947**-dən böyük olmayan bütün həqiqi ədədlərdən ibarət birölçülü **diap1** massivini yaradaq. Bunun üçün iki nöqtə ilə işarə olunan əməldən istifadə edirik:

```
>>diap1 = 3.7 : 0.3 : 8.947;
```

Sonuncu vergüllü nöqtədən MATLAB sisteminin əmrlər pəncərəsində əməllərin nəticələrinin, yəni **diap1** massivinin elementlərinin dərhal görünməsini dəf etmək üçün istifadə olunub. Ədədlərin sayı çox olduqda bütün elementlərin göstərilməsi pəncərənin tərkibinin böyük sürətlə keçməsilə müşayət olunacaq, bu isə göz üçün kifayət qədər yorucudur.

Verilmiş diapazona düşən, yəni **diap1** massivinə uyğun elementlərinin ümumi sayını həmin anda fikirdə hesablamaq çox çətindir. Buna görə də **length** funksiyasını aşağıdakı kimi çağırırıq:

```
>>length(diap1)
ans =
   18
```

Bu əməli, adətən, tam ədədlərin diapazonunu təşkil etmək üçün tətbiq edirlər:

```
>>diap2 = 4 : 2 : 26
```

Əgər artım (addım) birə bərabərdirsə, onda onu yazmamaq olar:

```
>>diap3 = 2:45
```

Massivlərlə iş üçün spesifik olan sonuncu əməli də daxil etdikdən sonra onların birgə tətbiqinin bəzi incəliklərini araşdıraq.

Vektor-sətirin formalaşdirilması üçün iki nöqtədən istifadə edilir.

```
>> 1+1:5

ans =

     2     3     4     5
```

Vektor-sütunun formalaşdırılması üçün ***transpose*** əməliyyətindən istifadə edilir. Bu əməl sətir və sütunun yerlərini dəyişir.

```
>> [1:5]'

ans =

     1
     2
     3
     4
     5
```

Verilən addım ilə formalaşdırılan sətir-vektor **linspace** əmri ilə yerinə yetirilir.

```
>> x=linspace(2,10,5)

x =

     2     4     6     8    10
```

"Diapazon" əməlini indeksləşdirmə əməli kontekstində tətbiq edək:

```
>> a = [ 1 2 ; 3 4 ; 5 6 ];
  b = a(2,:)

b =

     3     4
```

Burada "diapazon" əməli ikinci indeksin bütün mümkün qiymətlərini tapmağa imkan verir. Nəticədə **b** dəyişəni ikiölçülü massivin ikinci sətrinin elementlərindən təşkil olunmuş birölçülü massivə bərabər olur. Beləliklə, **a(2,:)** kompakt ifadəsi **a** matrisinin ikinci sətrini "götürməyə" imkan verir.

Əgər ***boş massiv*** mənasını verən və **[ ]** şəklində işarə olunan anlayışdan istifadə etsək, onda

>>a(2,:) = [ ]

ifadəsi sətri (və ya sütunu) yox etməyə imkan verir.

Başqa bir misala baxaq. Tutaq

>>a = [1 6 3 9 2];

birölçülü massiv verilib. Bu massivin ikincidən dördüncüyə qədər elementlərini ayırmağa çalışaq. Bunun üçün

>>b = a(2:4)

indeks ifadəsini yazaq. Göründüyü kimi burada indeksin bir qiyməti yox, indekslərin **2**-dən **4**-ə qədər ardıcıl artan tam qiymətlər diapazonu verilmişdir. Bu isə o deməkdir ki, biz **a** massivinin ardıcıl olaraq ikinci, üçüncü və dördüncü elementlərini **b** dəyişəninə yerləşdirməklə qrup əməliyyatı yerinə yetiririk. Beləliklə, birölçülü **b** massivi yaranır:

```
b =
   6 3 9
```

Yuxarıdakı izahdan **a** massivindən ixtiyari ardıcıllıqla, məsələn, üç elementin seçilməsi üsulunu almaq olar, yəni birinci elementi **a** massivinin sonuncu elementinə, ikinci elementi **a**-nın ikinci elementinə, üçüncü elementi isə **a**-nın birinci elementinə bərabər birölçülü **b** massivini təşkil etmək olar: Bu məsələni həll edən ifadə

>>b = a([5 3 1])

şəklindədir, indeksləşdirmədə birincisi **5**-ə, ikincisi **3**-ə (izləmə ardıcıllığına görə), üçüncüsü isə **1**-ə bərabər olan indekslər yığını iştirak edir. Bu da **b** massivinin təşkil olunma ardıcıllığını izah edir.

```
>> a=[1 2 3; 4 5 6; 7 8 9]

a =

     1     2     3
     4     5     6
     7     8     9
```

a matrisində sütunların yerlərini dəyişək. Biz sonuncu sütunun birinci, ikinci sütunun sonuncu və birinci sütunun ikinci olmasını istəyirik. Məsələnin həlli aşağıdakı kimidir:

>>a = a(:,[3 1 2])

Bu şəkildə sətirlərin və sütunların yerlərini dəyişməkdən əlavə, onların bir hissəsini yox edib (yuxarıda bu məqsədlə boş massivdən istifadə olunmuşdu), digər sətirləri (sütunları) dublləşdirmək olar. Məsələn, əvvəlki əməliyyatdan alınan və

```
a =

     3     1     2
     6     4     5
     9     7     8
```

şəklində olan 3x3 ölçülü **a** matrisindən ölçüsü **4x5** olan **c** matrisini almaq olar:

```
>>    c = a([ 2 3 3 2 ] , [ 2 1 3 2 1 ])

c =

     4     6     5     4     6
     7     9     8     7     9
     7     9     8     7     9
     4     6     5     4     6
```

a matrisindən **c** matrisi alınan zaman onun birinci sətri yox edilib (istifadə olunmayıb), ikinci və üçüncü sətirləri təkrar edilib (iki dəfə götürülüb). Bu halda bu sətirlərdən hər biri onların birinci və ikinci elementlərinin təkrar edilməsi hesabına “uzadılıb” (elementlərin yerlərini dəyişməklə).

Misallar.

```
>> x=[1 2 3 4], y=[5,6]

x =
     1     2     3     4
y =

   5     6

>> whos
  Name          Size                Bytes  Class     Attributes
  x             1x4                    32  double
  y             1x2                    16  double
>> z=[3;5;7;9]
z =
     3
     5
     7
     9
>> whos z
  Name          Size                Bytes  Class     Attributes
  z             4x1                    32  double
```

Xüsusi matrislərə aid misallar

```
>> A=zeros(2,3)
          A =
     0     0     0
     0     0     0

>> B=ones(3,4)
          B =
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> C=eye(3)
          C =
     1     0     0
     0     1     0
     0     0     1
>> B=A(:,3:5)
```

```
B =

     8     1     2
     4     5    -3
    -6     4     3

>> C = A(1:3,4:5)

C =

     1     2
     5    -3
     4     3
```

```
>> x

x =

     5    -1     2     1     8     3

>> u = x([5 2 end])

u =

     8    -1     3
```

1.11 Vektor və matrislərin qarşılıqlı çevrilmələri

Vektor-sütunlardan və vektor-sətirlərdən matrislərin alınmasına və tərs misallara baxaq. Konkatenasiya əməliyyatı ilə vektor-sütunu yaradaq:

>>v = [1 ; 2]

Qeyd edək ki, bu nəticəni indeksləşdirmə əməliyyatı ilə də almaq olar və bunun üçün bir yox iki indeksdən istifadə etmək lazımdır:

>>v(1,1) = 1;

>>v(2,1) = 2;

Çünki,

v(1) = 1; v(2) = 2

yazdıqda vektor-sətir yaranmış olur.

Indi isə **v** dəyişənini vektor-sütunundan **2x2** ölçülü matrisə çevirək:

```
>> v(1,1) = 1;
      v(2,1) = 2;
>> v = v(:,[ 1 1 ])

v =

     1     1
     2     2
```

Yerinə yetirilən əməliyyatda matrisi almaq üçün verilən vektor-sütun edilib (iki dəfə artırılıb). Bəs əgər sütunu çox dəfə artırmaq lazımdırsa, bəs onda nə etməli? Konkatenasiyanı bir neçə kilometr uzunluğunda yazmayacağıq ki? Və ya əgər vektor-sütunu **N** dəfə dublə etmək lazım gələrsə, burada **N** — dəyişəndir. Bu halda bütün elementləri birə bərabər olan istənilən ölçülü massiv yarada bilən **ones** funksiyası lazım olacaq.

Xüsusi halda, birölçülü massivin istehsalı üçün bu funksiya massivin ölçüsünə bərabər (elementlərinin sayı) olan bir arqument qəbul edir. Aşağıdakı

U = ones(N)

ifadəsi **N** sayda elementlərdən ibarət birölçülü **U** matrisini yaradır. Nəticədə isə

v = v(:,ones(N))

ifadəsi **v** vektor-sütununu **N** dəfə artırmağa, yəni **2xN** matrisini almağa imkan verir.

Sonda çox mürəkkəb bir ifadə göstərək. Tutaq ki, **A** — **3x2** matrisdir:

A = [4 2 ; 1 6 ; 5 3],

a=[1 7 2 4 5 1] — vektor-sütundur. Biz bilirik ki, **A(2,1)** ifadəsi **A** matrisinin konkret elementini (ikinci sətirdə və birinci sütunda yerləşmiş) verir; **A(:,2)** ifadəsi isə **A** matrisinin ikinci sütununu (ikinci indeks **2**-dir, birinci indeks isə artma sırası ilə bütün mümkün qiymətləri alır) verir.

Bəs onda **A(:)** ifadəsi nə deməkdir? Sən demə bu elə **A(:,:)** ifadəsinin eynisidir. Sonuncu elə həmin **A** matrisidir, **A(:)** isə **A** matrisinin sütunlar üzrə xətti nizamlanmış bütün elementlərin yığınıdır (yəni vektor-sütundur): əvvəlcə birinci sütunun bütün elementləri, sonra ikinci sütunun bütün elementləri və s. ardıcıl düzülür (qeyd edək ki, kompüterin yaddaşında bütün matrislər bu şəkildə saxlanılır). Indi aydındır ki,

a(A(:)) == [4 1 5 7 1 2]

bərabərliyi doğrudur. Burada **A** matrisinin sütunlar üzrə nizamlanmış elementləri **a** massivinə giriş indeksləridir və bu massivdən nəticə-massivin təşkili üçün elementlər götürülür.

Bəs **a(A)** kimi ifadə hansı mənanı kəsb edir? Bu barədə MATLAB sisteminin özündən soruşaq:

```
>> a=[ 1 7 2 4 5 1 ]

a =

     1     7     2     4     5     1

>> A = [ 4 2 ; 1 6 ; 5 3 ]

A =

     4     2
     1     6
     5     3

>> a(A(:)) == [ 4 1 5 7 1 2 ]

ans =

     1     1     1     1     1     1
```

Şəkildən görünür ki, birölçülü massivdən **3x2** ölçülü matris alınır. MATLAB bu ifadəni birölçülü **a** massivindən qiymətlərin seçilməsi üçün indekslər verən **A** matrisi ilə eyni strukturlu ikiölçülü massivin yaradılması ilə izah edir.

Beləliklə, indekslər üçün yalnız xətti diapazondan deyil, həmçinin həndəsi strukturlaşdırılmış ədədlər yığınından da istifadə etmək olar!

1.12 MATLAB sisteminin çoxölçülü massivləri

Ölçüləri ikidən çox olan massivləri çoxölçülü massivlər adlandıracağıq. Belə massivlərin elementlərinin indeksləşdirilməsi üçün seçilən elementin bir neçə nizamlanma istiqaməti boyunca vəziyyətini göstərən üç və ya daha çox nömrələrdən istifadə olunacaq.

Üçölçülü massiv nümunəsində çoxölçülü matrislərin əyani təsvirlərini verək. Tutaq ki, bir ay müddətində hər gün eyni kəmiyyətlər üzərində ölçü aparılır və alınan qiymətləri gün ərzində düzbucaqlı cədvəldə qeyd edirlər. Ayın sonunda otuz sayda ikiölçülü cədvəl alınır. Təcrubi verilən çoxluğu necə nizamlamalı? Bunun üçün bu matrisləri hər hansı istiqamət üzrə düzərək onları nömrələmək olar. Bu, verilənlərin üçüncü nizamlanma istiqaməti olacaq, belə ki, birinci istiqamət üzrə (şərti olaraq - şaquli) sətirlər, ikinci istiqamət üzrə (üfüqi) sütunlar nizamlanmışdır. Nəticədə ayrılıqda hər bir elementə müraciət üçün göstərilən üçüncü indeks "*zaman istiqaməti*" boyunca cədvəlin nömrəsinə bərabər olacaq.

Əgər nizamlanmış verilənlərdən ibarət üçölçülü **A** massivi yaradırıqsa, onda hər bir verilənə üç indeksin köməyilə müraciət etmək olar. **A(1,2,2)** elementi birinci sətirdə, ikinci sütunda və ikinci matrisdə yerləşir. Şəkildən görünür ki, bu element **3**-ə bərabərdir.

Indi isə belə massivi yaratmaq üçün lazım olan əməliyyatlar haqda danışaq. Bunlardan birincisi, massivin bütün elementlərinin xətti yazılışıdır:

A(1,1,1)=3; A(2,1,1)=5; … A(1,2,1)=4;
A(2,2,1)=8; … A(1,1,2)=2;
A(2,1,2)=3; … A(1,2,2)=3;
… A(1,1,30)=7; A(2,1,30)=4;

Belə üçölçülü massivin bütün elementlərinin verilməsi prosesi qruplaşdırma imkanı verir. Əvvəlcə, növbə ilə birinci **B1** matrisini (ixtiyari üsulla), sonra ikinci **B2** matrisini və s. təşkil etmək olar. Bütün matrislər təşkil olunduqdan sonra üçölçülü massivi yaratmaq üçün aşağıdakı otuz qrup mənimsəmə əməliyyatını (bir mənimsəməyə bir matris) aparmaq olar:

A(:,:,1) = B1; A(:,:,2) = B2; …A(:,:,30) = B30;

Bu üsula daha əvvəl birölçülü və ikiölçülü massivlərin yaradılması zamanı baxmışdıq. Bundan başqa, belə matrislər üçün konkatenasiya əməliyyatı da tətbiq olunurdu. Lakin bu əməliyyat bilavasitə yüksək ölçülü massivlərin yaradılması üçün tətbiq edilə bilməz, çünki ayırıcılar (vergül və nöqtəli vergül) bu əməliyyat zamanı yalnız sütun və sətirlərin ayrılması üçün işlənir.

Və nəhayət, çoxölçülü massivlərin demək olar ki, bütün elementlərinin vahidə bərabər olduğu halda **ones** funksiyasından istifadə etmək və bundan sonra azsaylı qeyri-vahid elementlərə fərqi qiymətlər vermək məqsədə uyğundur.

Misal üçün ölçüsü **2x3x2** olan üçölçülü **D** massivi yaradaq:

>>D=ones(2,3,2);

```
>> D=ones(2,3,2)

D(:,:,1) =

     1     1     1
     1     1     1


D(:,:,2) =

     1     1     1
     1     1     1
```

sonra iki elementin qiymətini dəyişək:

>>D(2,2,1)=7; D(1,2,2)=5;

Bundan sonra MATLABdan bu massivi göstərməyi xahiş edək:

```
D(:,:,2) =

     1     1     1
     1     1     1

>>
    D(2,2,1)=7; D(1,2,2)=5;
>> D(:,:,1)

ans =

     1     1     1
     1     7     1

>> D(:,:,2)

ans =

     1     5     1
     1     1     1
```

Əvvəl matrislərlə bağlı (ölçüsü iki olan massivlər) demişdik ki, MATLAB massivin elementlərini yaddaşda bir böyük sütun şəklində yadda saxlayır. Bu, çoxölçülü massivlər üçün də doğrudur. Misal üçün, bundan əvvəl şəkildə göstərilən üçölçülü **D** massivi üçün əvvəlcə birinci matrisin (”səhifənin”) birinci sütun elementləri, sonra birinci matrisin ikinci sütun elementləri, daha sonra ikinci matrisin birinci sütun elementləri, və nəhayət, sonuncu olaraq ikinci matrisin ikinci sütun elementləri düzülür. Bu ardıcıllıqla yerləşdirilmiş elementlərə üçölçülü **D** massivi üçün tək bir indeks göstərməklə müraciət etmək olar:

D(3)=1; … D(4)=7; … D(7)=1; … D(9)=5; …

4 və daha yüksək ölçülü massivlərlə iş pinsipi 3 ölçülü massivlərlə iş prinsipinə analojidir.

1.13 Massivlərlə işləmək üçün funksiyalar

MATLAB sistemində məhz massivlərlə işləmək üçün nəzərdə tutulan bir neçə funksiya var. Belə funksiyalardan biri olan **ones** funksiyasını əvvəlki bölmədə araşdırmışdıq. Bu funksiya elementləri vahidə bərabər olan istənilən ölçülü massivlərin yaradılması üçün təyin edilmişdir.

Bu qrupdan olan **zeros** funksiyası sıfırlardan ibarət massiv, **rand** funksiyası isə **0**-dan

1-ə qədər intervalda bərabər paylanmış təsadüfi elementlərdən massiv yaradır.

Məsələn, bu funksiyanın iştirakı ilə verilən

>>A = rand(3)

ifadəsi elementləri 0-dan 1-ə qədər intervalda bərabər paylanmış **3x3** ölçülü massiv verir.

```
>> A = rand(3)

A =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
```

Əgər bu funksiyanı iki arqumentlə çağırsaq:

>>B = rand(2,3)

onda ölçüsü **2x3** olan **B** matrisi alınar, onun da elementləri təsadüfi ədədlərdir. Bu funksiyanı çoxsaylı skalyar arqumentlərlə də çağırmaq olar - bu halda təsadüfi ədədlərdən təşkil olunmuş çoxölçülü massiv yaratmaq mümkündür.

Ümumiyyətlə, MATLAB sistemində ***bir adla, lakin müxtəlif arqumentlərlə funksiyanı*** çağırmaq yalnız bu sistem üçün xarakterikdir!!! Istənilən funksiyanın çağırışının bütün mümkün variantları

>>help ???_???????

əmrinin köməyilə sadalanır.

rand funksiyası haqqında belə arayış aldıqdan sonra, məlum olur ki, **rand(2,3)** funksiyasının çağırılması **rand([2,3])** funksiyasının çağırılmasına tamamilə ekvivalentdir, yəni iki skalyar arqumentin yerinə iki elementli bir vektor-funksiya vermək olar.

size funksiyası **C = [1 2 ; 4 5 ; 7 8]** matrisi üçün **v** vektor-sətrini qaytardığından,

```
>> v = size(C)
v =
   3 2
```

Burada **C** matrisinin ölçüləri onun elementləridir,

>>D = rand(3,2); D = rand([3,2])

və

>>D = rand(size(C))

çağırışları ekvivalentdirlər. Sonuncu ifadənin yerinə aşağıdakı iki ardıcıl

>> [m,n] = size(C); D = rand(m,n);

ifadələrindən istifadə etmək olar, burada biz **C** matrisinin ölçüləri üçün **size** funksiyasının bir çağırılışı ilə təyin edilən iki **m** və **n** skalyar dəyişən daxil etmişik.

Çoxölçülü massivləri daha əvvəl araşdıran zaman biz demişdik ki, matrislərin ölçülərini dəyişdirməklə yeni matrisin yaradılmasında konkatenasiya əməliyyatını kvadrat mötərizələrin köməyilə tətbiq etmək olmaz (1 ölçülüdən 2 ölçülüyə keçid istisna olmaqla).

Bunun üçün xüsusi olaraq **cat** funksiyası tətbiq edilir. Tutaq ki, iki sayda ikiölçülü **A1 = [1 2 3 ; 4 5 6]** və **A2 = [7 8 9 ; 3 2 1]** massivləri (matrisləri) verilib. Onda onların müxtəlif istiqamətlər boyunca konkatenasiyası aşağıdakı nəticələri verir:

>> cat(1,A1,A2)

əmri **[1 2 3 ; 4 5 6 ; 7 8 9 ; 3 2 1]** massivini yaradır, yəni **[A1 ; A2]** nəticəsi alınır.

```
A1 =

     1     2     3
     4     5     6

>> A2 = [ 7 8 9 ; 3 2 1 ]

A2 =

     7     8     9
     3     2     1

>> cat(1,A1,A2)

ans =

     1     2     3
     4     5     6
     7     8     9
     3     2     1
```

>>cat(2,A1,A2)

əmri **[1 2 3 7 8 9 ; 4 5 6 3 2 1]** massivini yaradır, yəni **[A1 , A2]** nəticəsi alınır.

```
>> A1 = [ 1 2 3 ; 4 5 6 ]

A1 =

     1     2     3
     4     5     6

>> A2 = [ 7 8 9 ; 3 2 1 ]

A2 =

     7     8     9
     3     2     1

>> cat(2,A1,A2)

ans =

     1     2     3     7     8     9
     4     5     6     3     2     1
```

>>A3 = cat(3,A1,A2)

əmri **A1** və **A2** ikiölçülü massivlərdən **A3** üçölçülü massivini yaradır və bu əmrlər pəncərəsində aşağıdakı kimi göstərilir:

```
>> A3 = cat(3,A1,A2)

A3(:,:,1) =

     1     2     3
     4     5     6


A3(:,:,2) =

     7     8     9
     3     2     1
```

Massivlərin elementləri ilə sadə hesablamaları aparmaq üçün **sum**, **prod**, **max**, **min** və **sort** funksiyaları təyin olunmuşdur. **sum** funksiyasını, adətən, birölçülü massivlərin bütün elementlərinin toplanması üçün tətbiq edirlər. Matrislər üçün (ikiölçülü massivlər) bu funksiya sütunlardakı elementləri toplayır, belə ki, əgər **B = [1 2 3 ; 4 5 6 ; 7 8 9]** olarsa, onda

```
>> B = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]

B =

     1     2     3
     4     5     6
     7     8     9

>> sum(B)

ans =

    12    15    18
```

Çoxölçülü massivlər üçün bu funksiya elementlərin toplanma istiqamətini seçməyə imkan verir. **prod** funksiyası indicə baxdığımız **sum** funksiyasına analojidir, lakin toplama yerinə vurma yerinə yetirilir.

```
>> B = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]

B =

     1     2     3
     4     5     6
     7     8     9

>> prod(B)

ans =

    28    80   162
```

max və **min** funksiyaları massivlərdə uyğun olaraq maksimal və minimal elementləri axtarır, belə ki, birölçülü halda ekstremal element təkdir, lakin matrislər üçün hər sütun üzrə seçilərək ekstremal elementlər yığını alınır. Beləliklə,

```
>> B = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]

B =

     1     2     3
     4     5     6
     7     8     9

>> max(B)

ans =

     7     8     9
```

Bu funksiyalar misalında MATLAB sistemində *funksiyaların bir unikal xüsusiyyətini* - bunu bacara bilən funksiyalardan *qayıdan qiymətlərin müxtəlif sayda olmasını* tələb etmək imkanını - açıqlayaq!!!

Tutaq ki, **v=[5 2 6 8 3];** vektor sətri verilib**.** Bu massiv üçün **min** vektor-funksiyasını çağırmaqla onun elementləri arasından ən kiçik olanını tapırıq:

m = min(v);

Bu 2-dir. Əgər biz ən kiçik elementin **N** nömrəsi (indeksi) ilə də maraqlanırıqsa, onda **min** funksiyasını aşağıdakı kimi çağırmalıyıq:

```
>> v=[ 5 2 6 8 3];
>> m = min(v)

m =

     2

>> [ m , N ] = min(v)

m =

     2


N =

     2
```

Buradan görünür ki, **v** vektorunda ən kiçik element ikinci yerdə durur.

sort funksiyası birölçülü massivlərin elementlərini, matrislərdə sütunların elementlərini (hər sütunu ayrılıqda) və s. artan sıra ilə nizamlayır.

Sonda çox vacib olan **all**, **any** və **find** funksiyalarına baxaq. **all** funksiyası vektora tətbiq edilərkən vektorun bütün elementləri sıfırdan fərqli olan halda **1** ("*doğru*"), və əks halda (yəni, əgər elementlərdən heç olmasa biri sıfır olan halda) **0** qaytarır. **any** funksiyası əks tərzdə işləyir. O, vektorun bütün elementləri sıfır olan halda sıfır, heç olmasa bir element sıfırdan fərqli olan halda isə vahid qaytarır. Hər iki funksiyanı matrislərə tətbiq etdikdə, onlar matrisin sütunlarını vektorlar kimi qəbul edir və onların hər biri üzrə yuxarıda göstərilən şəkildə nəticə verir. Misal üçün, əgər **F = [1 2 0 ; 0 3 0; 7 4 0]** olarsa, onda

all(F) == [0,1,0]; any(F) == [1,1,0];

find funksiyası arqument kimi hər hansı bir şərti qəbul edir və massivin elementlərinin sıfıra bərabər olmayan nömrələrinin (indekslərinin) yığınını qaytarır. Birölçülü **v** massivi üçün bu funksiyanın

>>u = find(v)

çağırılışı sıfırdan fərqli elementlərin indekslərindən ibarət **u** vektorunu qaytarır. Xüsusi halda, **v=[1 0 3]** vektoru üçün

find(v) == [1 3]

bərabərliyi doğrudur, çünki nömrələri (indeksləri) **1** və **3** olan elementlər sıfırdan fərqlidir.

Matrislər üçün **find** funksiyası bir indekslər vektoru deyil, indekslər yığını qaytarır. Bu yığının hər bir vektoru matrisin ayrı-ayrılıqda götürülmüş sütunlarının işlənməsi nəticəsində alınır. Məsələn, **A = [1 0 3; 0 4 5; 6 7 8]** matrisi üçün bu funksiyanın

>> [u1 , u2] = find(A)

çağırışı **u1** və **u2** vektor-sütunlarını qaytarır, bunlardan birincisi **A** matrisinin sıfırdan fərqli elementlərinin sətirlərinin nömrələrindən təşkil olunub, burada matrisin sütunlarına soldan-sağa istiqamətində baxılır (əvvəlcə birinci sütun, sonra ikinci sütun və s.). **u2** sütun vektoru sıfırdan fərqli elementlərin sütunlarının nömrələrindən təşkil olunub. Yəni aşağıdakı

u1 == [1; 3; 2; 3; 1; 2; 3];
u2 == [1; 1; 2; 2; 2; 3; 3; 3]

bərabərliyi doğrudur.

1.14 Massivlərlə hesablama

Massivlər MATLAB sistemində ən geniş yayılan və ən dəyərli verilənlər tipi olduğundan, onlarla aparılan hesablamaların bütün incəliklərini daha geniş şəkildə araşdıraq. Əvvəlki bölmələrdə biz hesabi, məntiqi əməliyyatların və xüsusi halda skalyarların (**1x1** ölçülü massiv) müqayisəsi əməliyyatlarının işləmə prinsipinə baxmışdıq. Indi bütün bunları massivlər halı üçün təkrar edək.

Hesab əməliyyatlarından başlayaq. Matrislərin toplanma və çıxma əməliyyatları elementlər üzrə aparılır və ənənəvi işarələrdən istifadə olunur.

Əgər,

A = [1 1 1; 2 2 2; 3 3 3]; **B = [0 0 0; 7 7 7; 1 2 3]** olarsa, onda **A + B == [1 1 1 ; 9 9 9 ; 4 5 6];** bərabərliyi doğrudur.

```
>> A = [ 1 1 1; 2 2 2; 3 3 3 ]

A =

     1     1     1
     2     2     2
     3     3     3

>> B = [ 0 0 0; 7 7 7; 1 2 3 ]

B =

     0     0     0
     7     7     7
     1     2     3

>> A + B

ans =

     1     1     1
     9     9     9
     4     5     6
```

Toplanan (çıxılan) matrislər eyni ölçülü olmalıdırlar. Toplananlardan birinin skalyar olduğu hal istisna edilir. Bu halda verilən matrisin özünə uyğun ölçülü matris yaradılır ki, sonuncunun elementlərinin hamısı eyni olub verilmiş skalyara bərabər olsun. Məsələn,

```
>> A + 5

ans =

     6     6     6
     7     7     7
     8     8     8
```

Yəni elementlər üzrə toplama aparmadan öncə verilən skalyardan **[5 5 5; 5 5 5; 5 5 5]** adsız matrisini alırlar və onu A matrisi ilə toplayırlar.

Indi isə toplamaya və çıxmaya (həmçinin qüvvətə yüksəltməyə) baxaq. Burada *ənənəvi işarələr xətti cəbrin əməlləri üçün nəzərdə tutulub*: ***** işarəsi matrislərin vurulmasını (həmçinin vektor-sətir və vektor-sütunların) məlum "*sətir-sütuna*" qaydası üzrə yerinə yetirir. Ənənəvi / bölmə işarəsi (həmçinin \ işarəsi) "*xətti tənliklər sisteminin həllinin*

tapılması əməliyyatı” üçün nəzərdə tutulmuşdur. Bütün bu xətti cəbrin əməliyyatlarına (trasponirə etmə əməliyyatı da daxil olmaqla) MATLAB sisteminin köməyi ilə riyazi məsələlərin həllinə həsr olunmuş başlıqda geniş şəkildə baxılacaq.

Elementlər üzrə yerinə yetirilən əməliyyatlar üçün ənənəvi işarələr özlərindən əvvəl qoyulan nöqtə ilə müşayət olunurlar. Yəni massivlərin elementlər üzrə vurulması **.*** ilə işarə olunur və buna görə aşağıdakı bərabərlik aydındır:

```
>> A .* B

ans =

     0     0     0
    14    14    14
     3     6     9
```

Uyğun olaraq **A.^2** ifadəsi **A** matrisinin hər bir elementini kvadrata yüksəltməyə imkan verir. Skalyara elementlər üzrə vurmanı yerinə yetirdikdə “*skalyarın matrisə qədər genişləndirilməsi*” aparılır, başqa sözlə, matrisin hər bir elementi bu skalyara vurulur:

```
>> A .* 4

ans =

     4     4     4
     8     8     8
    12    12    12
```

Elementlər üzrə bölmə aparmaq üçün **./** və **.** işarələrinin kombinasiyasından istifadə edilir və uyğun olaraq “*sağdan və soldan*” elementlər üzrə bölmə adlandırılır. **A./B** ifadəsi elementləri **A(k,m)/B(k,m)** olan matris, **A.\B** ifadəsi isə elementləri **B(k,m)/A(k,m)** olan matrisi verir. Hər iki matris eyni ölçüyə malik olmalıdır, skalyar halında skalyarın operand-matrisin ölçüsünə qədər genişləndirilməsi aparılır. Məsələn,

```
>> 6 ./ A

ans =

     6     6     6
     3     3     3
     2     2     2
```

Münasibət və məntiqi əməllər üçün matris halında eyni işarələrdən istifadə olunur və əməllər elementlər üzrə aparılır. Skalyar halında “*genişləndirilmə*” aparılır. “*Kiçikdir və ya bərabərdir”* əməlindən istifadə etməklə **A** və **B** matrislərinin müqayisəsini aparaq:

```
>> A<=B

ans =

     0     0     0
     1     1     1
     0     0     1
```

Burada matrisdaxili yerləşmə üçün hər bir sıfır "*yalan*", hər bir vahid "*doğru*" mənasını verir. Alınmış matris (vahidlərlə) **A** matrisinin hansı elementlərinin uyğun olaraq **B** matrisinin elementlərindən kiçik və ya bərabər olduğunu göstərir.

Massivlərin müqayisəsi əməliyyatı adətən **find** funksiyasının çağırışı ilə kombinə edilir. Həqiqətən, fərz edək ki, **v = [1 3 6 7 0]** massivininin vahiddən böyük bütün elementlərini **9**-a bərabərləşdirmək lazımdır. Bu məsələ

>>v(find(v > 1)) = 9;

ifadəsinin köməyi ilə həll olunur. Bu ifadənin iş prinsipini aydınlaşdıraq. **find** funksiyası **v** vektorunun **v>1** bərabərliyini ödəyən elementlərinin nömrələri yığınını (massivini) qaytarır. Aydındır ki, bu **[2 3 4]** nömrələridir. Sonra bu nömrələr **v** massivin indeksləşdirmə əməliyyatına daxil olur və bu **v**-nin ikinci, üçüncü və dördüncü elementlərinə doqquzun mənimsədilməsini təmin edir:

```
>> v = [ 1 3 6 7 0 ]

v =

     1     3     6     7     0

>> v(find(v > 1)) = 9

v =

     1     9     9     9     0
```

Beləliklə, vahiddən böyük bütün elementlər doqquza bərabər oldu.

Məntiqi əməliyyatların işini **v** vektorunun yeni aldığımız qiymətləri üzərində ona "YOX" əməliyyatını tətbiq etməklə nümayiş etdirək:

~v == [0 0 0 0 1]

Əvvəlki bölmələrdə baxdığımız riyazi funksiyalardan yalnız skalyar arqumentər üçün deyil, həmçinin massiv-arqumentlər üçün də istifadə edilir. Nəticədə, çox kompakt yazılışların köməyilə böyük həcmli hesablamalar aparmaq mümkün olur. Məsələn,

>> x = 0 : 0.01 : pi/2; y = sin(x);

ifadəsi **sin** funksiyasına **158** nöqtədə eyni vaxtda hesablama aparmağa imkan verir. Nəticədə, biz **x** arqumentlər massivni (cəmi 158 qiymət) və **y** funksiyanın qiymətlər massivini (cəmi 158 qiymət) alırıq. Bu iki matris, **sin** funksiyasının qrafikini ətraflı şəkildə qurmağa kifayət edir. MATLAB sistemində bu iki massivi funksiyaların qrafikini avtomatik quran xüsusi funksiyaya vermək olar. Bu texnikaya növbəti fəsildə ayrılıqda baxılacaq. Indi isə

sin(x) = 0.315

tənliyinin təqribi həllini tapaq. Bunun üçün bizə yalnız bir ifadə yazmaq lazım gələcək:

```
>>res = (max(find(y < 0.315))— 1)*0.01
```

Hesablama nəticəsində təqribi kök alınır: **res = 0.3200**. Hesablama üçün 0.01 addımı verildiyindən, alınan təqribi kökün dəqiqliyi 0.01-dən çox deyil. Tapılmış kökün dəqiqliyini yoxlamaq üçün

```
>>asin(0.315)
ans =
0.3205
```

hesablamasını aparaq. Buradan görünür ki, bizim tapılmış kökün dəqiqliyi haqda irəli sürdüyümüz fikir özünü doğruldur.

I FƏSIL ÜZRƏ MISALLAR

Misal 1. Tutaq ki, iki matris verilmişdir.

$$\mathbf{A}=\begin{bmatrix}1 & 0 & 1\\ 2 & 3 & 4\\ -1 & 6 & 7\end{bmatrix} \quad \text{and} \quad \mathbf{B}=\begin{bmatrix}7 & 4 & 2\\ 3 & 5 & 6\\ -1 & 2 & 1\end{bmatrix}$$

Matlabdan istifadə edərək aşağıdakı əməlləri yerinə yetirin:

(*a*) $\mathbf{A}+\mathbf{B}$

(*b*) $\mathbf{AB}$

(*c*) $\mathbf{A}^2$

(*d*) $\mathbf{A}^T$

(*e*) $\mathbf{B}^{-1}$

(*f*) $\mathbf{B}^T\mathbf{A}^T$

(*g*) $\mathbf{A}^2+\mathbf{B}^2-\mathbf{AB}$

(h) det (A), det(B), det(AB)

Həll:

```
A =

     1     0     1
     2     3     4
    -1     6     7

>> B = [7 4 2; 3 5 6; -1 2

B =

     7     4     2
     3     5     6
    -1     2     1

>> C = A + B

C =

     8     4     3
     5     8    10
    -2     8     8
```

b)

```
>> D = A*B

D =

     6     6     3
    19    31    26
     4    40    41
```

c)

```
>> E=A^2

E =

     0     6     8
     4    33    42
     4    60    72
```

d)

```
>> % Let F= transpose of A
>> F=A'

F =

     1     2    -1
     0     3     6
     1     4     7
```

e)

```
>> H = inv (B)

H =

    0.1111    0.0000   -0.2222
    0.1429   -0.1429    0.5714
   -0.1746    0.2857   -0.3651
```

f)

```
>> J = B'*A'

J =

     6    19     4
     6    31    40
     3    26    41
```

g)

```
>> K = A^2 + B^2 - A*B

K =

    53    52    45
    15    51    58
    -2    28    42
```

h

```
>

ans =

    12

>> det (B)

ans =

  -63.0000

>> det(A*B)

ans =

 -756.0000
```

Misal 2. Aşağıdakı verilmiş matrislərin məxsusi ədələrini və vektorunu tapın.

$$\mathbf{A}=\begin{bmatrix} 4 & 2 & -3 \\ -1 & 1 & 3 \\ 2 & 5 & 7 \end{bmatrix} \quad \text{and} \quad \mathbf{B}=\begin{bmatrix} 1 & 2 & 3 \\ 8 & 7 & 6 \\ 5 & 3 & 1 \end{bmatrix}$$

Həlli:

```
>> % Determine the eigenvalues and eigenvectors
>> A = [4 2 -3 ; -1 1 3 ; 2 5 7]

A =

     4     2    -3
    -1     1     3
     2     5     7

>> eig(A)

ans =

    0.5949
    3.0000
    8.4051

>> lamda = eig(A)

lamda =

    0.5949
    3.0000
    8.4051
```

```
>> [V, D] = eig (A)

V =

   -0.6713    0.9163   -0.3905
    0.6713   -0.3984    0.3905
   -0.3144    0.0398    0.8337


D =

    0.5949         0         0
         0    3.0000         0
         0         0    8.4051
```

Misal 3. Aşağıdakı xətti tənliklər sistemini həll edərək, x_1, x_2, x_3 dəyişənlərini tapnınız.

$$x_2 - 3x_3 = -5$$
$$2x_1 + 3x_2 - x_3 = 7$$
$$4x_1 + 5x_2 - 2x_3 = 10$$

Indi isə bu xətti tənliklər sistemini matris şəklində yazaq,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & -3 \\ 2 & 3 & -1 \\ 4 & 5 & -2 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 5 \\ 7 \\ 10 \end{bmatrix} \text{ and } \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\mathbf{AX} = \mathbf{B}$$
$$\mathbf{A^{-1}AX} = \mathbf{A^{-1}B}$$
$$\mathbf{IX} = \mathbf{A^{-1}B}$$
$$\mathbf{X} = \mathbf{A^{-1}B}$$

Həllı

```
>> A = [0 1 -3; 2 3 -1; 4 5 -2];
>> B = [-5; 7; 10]
>> x = inv (A)*B
   x =
      -1.0000
       4.0000
       3.0000
>> check = A*x
check =
      -5
       7
      10
```

```
% Alternative method
>> x = A\B
   x =
      -1
       4
       3
```

2. MATLAB SISTEMINDƏ PROQRAMLAŞDIRMA

m- faylların yerinə yetirilməsində istifadəçiyə aşağıdakı ehtiyacları ola bilər:

- Ekrana məlumatları klaviaturadan daxil etmək üçün ;
- Düymənin basılmazından əvvəl ara vermək;
- Istifadəçinin qrafiki interfeysindən istifadə etmək.

2.1 Klaviaturadan verilənlərin daxil edilməsi üçün sorğunun formalaşması

Bu əmrin sintaksisi Input funksiyası ekrana sorğu göndərir və istifadəçidən cavab verir. aşağıdakı şəkildədir:

```
>>n = input('sorgu')
```

klaviaturadan daxil edilən verilənləri geri qaytarır. Əgər istifadəçi riyazı ifadə daxil edirsə, onda funksiya ifadəni hesablayır və uyğun qiyməti geri qaytarır. Funksiya dialoq tələb edən tətbiqi proqramlar üçün çox faydalıdırlar. Input funksiyası sadəcə ədədi yox həm də sətir tipli ifadələri də geri qaytara bilər. Simvol tipli ifadələri daxil etmək üçün funksiya parametrlərinin siyahısına 's' əlavə edilir:

Misal

```
>>name = input('Adresi daxil edinz:','s');
```

Aralığın verilməsi. Alqoritmin ayrı-ayrı addımları arasında müəyyən ara verilməsi ehtiyacı duyulur. Məsələn, qrafiklərin ekrana çıxarılması zamanı **pause** əmrindən istifadə edilir. Bu əmr əlavə parametrlərdən istifadə etmir. Bu əmr proqramın yerinə yetirməsini hər hansı klavişin basılmasına kimi duruxdurur, yəni ara verir. Müəyyən zaman ara verilməsini təmin etmək üçün mötərizədə saniyələrlə ara müddətini verirsən. İstifadə olunma şəkli aşağıdakı kimdir:

```
>>pause(n)
```

2.2 if şərti operatoru

Proqramda hər hansı məntiqi realizə etmək üçün məntiqi operatorlardan istifadə edilir. Bu operatorlar verilən şərtə görə proqramın yerinə yetirilmə istiqamətini dəyişir. Bu operatorun sintaksisi aşağıdakı şəkildədir:

```
if <ifadə>
<operatorlari>
end
```

Əgər «ifadə» parametri «doğrudursa», onda operator yerinə yetirilir əks halda proqram tərəfindən onun yerinə yetirilməsi aparılmır. Onu da qeyd edək ki, «ifadə» şərti ifadə olub hər hansı şərtin yerinə yetirilməsini yoxlayır. Aşağıdakı cədvəldə if operatorunun sadə variantları verilmişdir.

Cədvəl 2.1.

Sadə məntiqi ifadələr

if a < b	əgər ***a*** dəyişəni ***b*** dəyişənindən kiçikdirsə doğru. Tərsi olursa isə yanlışdır
if a > b	Əgər ***a*** dəyişəni ***b*** dəyişənindən böyükdürsə doğru. Tərsi olursa isə yanlışdır
if a == b	Əgər ***a*** dəyişəni ***b*** dəyişəninə bərabərdirsə doğru. Tərsi olursa isə yanlışdır
if a <= b	Əgər ***a*** dəyişəni ***b*** dəyişənindən kiçik və ya bərabərdirsə doğru. Tərsi olursa isə yanlışdır
if a >= b	Əgər ***a*** dəyişəni ***b*** dəyişənindən böyük və ya bərabərdirsə doğru. Tərsi olursa isə yanlışdır
if a ~= b	Əgər ***a*** dəyişəni ***b*** dəyişəninə bərabər deyilsə doğru. Tərsi olursa isə yanlışdır

Aşağıda sign() funksiyasının realizə olunma misalı göstərilmişdir. Əgər ədəd sıfırdan böyük isə +1 qiymətini geri qaytarır, əgər ədəd sıfırdan kiçik isə -1 qiymətini geri qaytarır.

```
function my_sign
x = 5;
if x > 0
disp(1);
end
if x < 0
disp(-1);
end
if x == 0
disp(0);
end
```

yuxarıda göstərilən misalın analizi onu göstərir ki, üç şərt bir birini inkar edən şərtdir, yəni bu şərtlərdən biri çalışdığı zaman o birisi şərtləri yoxlamaq lazım deyil. Belə hallarda aşağıdakı konstruksiyadan istifadə etmək lazımdır.

```
if <ifadə>
<operatorlar1>      % əgər şərt doğru isə yerinə yetirilir
else
<operatorlar2>      % əgər şərt yanlış isə yerinə yetirilir
end
```

Buna uyğun olaraq misalı aşağıdakı şəkildə yaz bilərik:

```
function my_sign
x = 5;
if x > 0
disp(1);
else
if x < 0
disp(-1);
else
disp(0);
end
end
```

Bu proqram ilə ilkin olaraq *x* dəyişənin müsbət olmasını yoxlayır. Əgər o müsbət isə ekrana 1 və başqa şərtlər diqqətə alınmır. Əgər birinci şərt yanlış isə proqram else (əks halda) ilə ikinci şərtə keçir. Ikinci şərtdə isə x dəyişəninin mənfi olması yoxlanılır, əgər şərt doğru isə onda ekrana -1 qiyməti çıxarılır. Əgər hər iki şərt yanlış isə o zaman ekrana 0 çıxarılır. Qeyd olunan problemi daha bir konstruksiya ilə həll edilir. Bu konstruksiya üçün **elseif** operatordan istifadə edilir:

```
if <ifadə1>
<operatorlar1>      % ifadə1 doğru isə yerinə yetirilir
elseif <ifadə2>
<operatorlar2>      % ifadə2 doğru isə yerinə yetirilir
...
elseif <ifadəN>
<operatorlarN>      % ifadəN doğru isə yerinə yetirilir
end
```

daha konkret misal üçün aşağıdakı kimi yazılır:

```
function my_sign
x = 5;
if x > 0
    disp(1);      %   əgər x > 0 isə yerinə yetirilir
elseif x < 0
    disp(-1);     % əgər   x < 0 isə yerinə yetirilir
else
    disp(0);      % Əgər   x = 0 isə yerinə yetirilir
end
```

Əsas misal olaraq kvadrat tənliyin həllində bu operatorun istifadəsinə baxaq,

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Əgər $b^2 - 4ac > 0$, onda tənliyin iki real kökü, Əgər $b^2 - 4ac < 0$ isə onda iki kompleks kökə sahib olacaqdir.

```
D = b^2-4*a*c;

if D > 0
    disp('This equation has two distinct real roots');
elseif D == 0
    disp('This equation has two identical real roots');
else
    disp('This equation has two complex roots.');
end
```

Tələbələrin aldiqları ballara uyğun qiymətləndirmə yerinə yetiriləndə bu funksiyadan da istifadə edə bilərik:

$$\begin{array}{rcccl} 95 & < & \text{grade} & & \text{A} \\ 86 & < & \text{grade} & \leq & 95 \quad \text{B} \\ 76 & < & \text{grade} & \leq & 86 \quad \text{C} \\ 66 & < & \text{grade} & \leq & 76 \quad \text{D} \\ 0 & < & \text{grade} & \leq & 66 \quad \text{F} \end{array}$$

```
grade = 80; % example
if grade > 95
 disp('The grade is A');
elseif grade <= 95 & grade > 86
 disp('The grade is B');
elseif grade <= 86 & grade > 76
 disp('The grade is C');
elseif grade <= 76 & grade > 66
 disp('The grade is D');
else
 disp('The grade is F');
end
```

bir başqa yanaşma da belə etmək olar:

```
grade = 80;
if grade > 95
    disp('The grade is A');
else
    if grade > 86
        disp('The grade is B');
    else
        if grade > 76
```

```
            disp('The grade is C');
        else
            if grade > 66
                disp('The grade is D');
            else
                disp('The grade is F');
            end
        end
    end
end
```

İf operatorundan mürəkkəb şərtləri yoxlamaq üçün məntiqi operatorlardan istifadə etmək lazımdır. Bunun üçün Matlab -da məntiqi operatorlardan istifadə etmək lazimdir:

& - məntiqi AND
| - məntiqi OR
~ - məntiqi NOT

İndi isə mürəkkəb şərtlərin istifadəsinə aid misala baxaq. Fərz edək ki, x dəyişənin 0 ilə 2 diapazonuna düşməsi yoxlanılır. Bu proqram aşağıdakı kimi yazılır:

```
function my_if
x = 1;
if x >= 0 & x <= 2
   disp('x - 0-dan   2-kimi diapazondadir');
else
   disp('x -   0-kimi   2 –kimi diapazonunda   deyil');
end
```

ikinci misalda x dəyişəninin 0- 2 diapazonuna aid olmamasına dair yoxlama aparaq. Bu aşağıdakı iki şərtdən birinin yerinə yetirilməsi hesabına başa gəlir:

x < 0 və ya x > 2:

```
function my_if
x = 1;
if x < 0 | x > 2
   disp('x dəyişəni 0 – 2 diapazonuna aid deyil');
else
   disp('x   dəyişəni   0-2 diapazonuna aid edyil');
end
```

Bununla bərabər ANA, OR, NOT məntiqi operatorların köməyi ilə etməklə müxtəlif mürəkkəb operatorlar yaratmaq olar. Məsələn, x dəyişənin [5 -5] diapazonuna düşməsini və [0 1] düşməməsini yoxlamaq lazım olur. Bunu aşağıdakı olduğu kimi realizə etmək olar:

```
function my_if
x = 1;
if (x >= -5 & x <= 5) & (x < 0 | x > 1)
   disp('x dəyişəni [-5, 5] diapazonuna aiddir, amma [0, 1] intervalına aid deyil');
else
   disp('x dəyişəni [-5, 5] və [0, 1] diapazonuna aid deyil ');
end
```

Diqqətə almaq lazımdır ki, mürəkkəb şərtlərin formalaşdırılmasında fiqurlu mötərizədən istifadə edilmişdi. AND əməliyyatının üstünlük dərəcəsi OR əməliyyatının üstünlük dərəcəsindən yüksəkdir. Fiqurlu mötərizə olmadığı halda məntiqi ifadənin nəticəsi tamam fərqli nəticə verəcəkdir. Fiqurlu mötərizə proqramlaşmada operatorların üstünlük dərəcəsini (prioritetini) dəyişmək üçün istifadə edilir. Cəbri operatorlarında olduğu kimi məntiqi operatorlar da proqramçının istəyinə görə dəyişdirilə bilinər. İlk olaraq mötərizə daxili operatorlar daha sonra isə mötərizənin xaricində yerinə yetirilir. Buna görə də yuxarıda göstərilən misalda lazımı nəticəni almaq üçün mötərizə istifadə olunmuşdu.

Məntiqi əməliyyatların üstünlük dərəcəsi:

NOT (~) – ən yüksək üstünlük dərəcəsi;

AND (&) – orta üstünlük dərəcəsi;

OR(|) – ən aşağı üstünlük dərəcəsi.

2.3 switch şərti operatoru

Bu operator sabitlər çoxluğu ilə qiymətlərin müqayisəsini həyata keçirir. Məsələn kiçik hərflərin böyük hərflər ilə əvəzedilməsində, cari simvolu bütün hərflər ilə müqayisə etmək lazım olur. Belə məqsədlər üçün switch operatorundan istifadə etmək lazımdır. Bu operatorun istifadə olunma sintaksisi aşağıdakı kimidir:

```
 switch expr
       case case_expr,
         <operatorlar1>
       case {case_expr1, case_expr2, case_expr3,...}
         <operatorlar2>
       ...
       otherwise,
         <operatorlar>
end
```

```
witch SWITCH_EXPRESSION
case CASE_EXPR1
        code 1
        code 2
        ...
case CASE_EXPR2
        code 1
        code 2
        ...
otherwise
        code 1
        code 2
        ...
end
```

burada, expr – dəyişənin qiymətinin bu və digər konstantalar bərabər olub olmamağı yoxlanılır; case_expr – dəyişənin qiymətləri yoxlanılan konstanta; otherwise –kəlmə olub bütün şərtlərin yanlış olduğu halda operatorları yerinə yetirir. Aşağıdakı misalda əlifbanın kiçik hərflərini böyük hərflər ilə əvəz edilir.

```
function upper_symbol
ch='c';
switch ch
    case 'a', ch='A';
    case 'b', ch='B';
    case 'c', ch='C';
    case 'd', ch='D';
    case 'e', ch='E';
   ...
    case 'z', ch='Z';
end

disp(ch);
```

Bu proqramda ch verilən simvol tipli dəyişəndir. Onun qiyməti ***c*** bərabərdir. Daha sonra isə switch operatorunun köməyi ilə əlifbanın a-dan z-ə kimi bütünü kiçik hərfləri ilə müqayisə edilir. Bu şərtlərdən biri ödəndikdən sonra switch operatoru işini bitirir və idarəetmə **disp()** operatoruna keçir. O isə öz növbəsində *ch* dəyişənin qiymətini ekrana çıxarır.

Misal. Temperaturun müxtəlif ölçü vahidinə çevrilməsinə aid proqram:

```
nit1 = input('Enter the current units (K,C,F): ','s');
T1 = input('Enter the value of the temperature to be converted: ');
Unit2 = input('Enter the new unit (K,C,F): ','s');

switch Unit1
 case 'C'
      Tin = T1;
 case 'F'
      Tin = (T1-32)*(5/9);
 case 'K'
      Tin = T1-273;
end

switch Unit2
 case 'C'
      Tout = Tin;
 case 'F'
      Tout = (5/9)*Tin+32;
 case 'K'
      Tout = Tin + 273;
end
disp(['The temperature in the new unit is ' num2str(Tout) ' ' Unit2 ]);
```

Misal: Klaviaturadan 1 ilə 5 arasında bir tam ədəd girin və bunu ekranda çap edin. Bu araladığın xaricində ədəd daxil edildiyin də xəbərdarlıq verən mesaj verməsini təmin edən bir proqram fraqmentini yazınız.

program yazalım.
Çözüm:

```
s=input('Sayınızı girin :');
switch s
case 1;fprintf('Bir...' );
case 2 ;fprintf('Đki...' );
case 3 ;fprintf('Üç...' );
case 4 ;fprintf('Dört...' );
case 5 ;fprintf('Bes...' );
otherwise fprintf('Zehmet olmasa, 1- 5 arasında bir tam eded daxil edin'); end;
```

2.4 while dövrü operatoru

MatLab Proqramlaşdırma mühitində iki ədəd dövr operatoru vardır: while və for. Onların köməyi ilə rekurent alqoritmlərin proqramlaşdırılması yerinə yetirilir, məsələn, sıraların hesabında, variantların bir-bir seçilməsi kimi əməliyyatlarda istifadə edilə bilinir. Ən sadə hallarda while operatorunun sintaksisi aşağıdakı kimidir:

```
while <şərt>
    <opertorlar>
end
```

burada, **<şərt>** şərti ifadədir. Bu şərt doğru isə operatorlar yerinə yetirilir, əks halda isə end sözündən sonrakı operator çalışacaqdır.

```
x = 1;
while x <= 10
    x = 2*x;
    fprintf('x = %d \n',x)
end
```

Misal. While operatorundan istifadə etməklə 1-dən 20 kimi ədədi sıranın cəmini tapmalı.

$$S = \sum_{i=1}^{20} i$$

```
function sum_i
S = 0;              % cəmin başlanğıc qiyməti
i=1;                % cəmin sayğacı
while i <= 20       % dövr (i <= 20 olan kim çalışır)
    S=S+i;          % cəm hesablanır
    i=i+1;          % sayğacın qiyməti bir vahid artır
end                 % dövr sonu
disp(S);            % nəticəni yəni 210 ekranda çap edir
```

İndi isə məsələni daha da mürəkkəbləşdirək. 1-dən 20 kimi ədədi siranın cəmini cəmin qiyməti 20 olana kimi hesablayın yəni $S \le 20$. Burada operatorda iki şərt vardır: ya cəmin sayğacı 20 çatanda durur, ya da S qiyməti 20 böyük(yəni kiçik bərabər) olsun. Bunu while-dən istifadə etməklə bu məntiqi realizə etmək olar:

```
function sum_i
S = 0;              % cəmin başlanğıc qiyməti
i=1;                % cəmin sayğacı
while i <= 20 & S <= 20 %i<=10 və S<=20 olduğu_müddət_ərzində çalışır
    S=S+i;          % cəmi hesablayaq
    i=i+1;          % sayğac qiyməti 1 vahid artır
end                 % dövrün sonu
disp(S);            % şərti ödəyən cəmin çapı- 21
```

Dövr operatorunun çalışmasını zorumla olaraq bitirmə üçün xüsusi operatordan istifadə edilir (break):

```
function sum_i
S = 0;              %cəmin başlanğıc qiyməti
i=1;                % cəmin sayğacı
while i <= 20       % dövr (i<=10 olan kimi
    S=S+i;          % cəm hesablanır
    i=i+1;          % sayğac 1 vahid artır
   if S > 20        % əgər S > 20,
      break;        % dövr bitir
    end
end                 % dövrün sonu
disp(S);            % nəticənin ekranda çap edilməsi
```

İkinci şərt o zaman bitir ki, S<20 olsun. Break operatoru vasitəsi ilə dövrdən çıxır və disp() operatoru ilə qiyməti çap edilir. Dövrün ikinci operatoru **continue operatorudur.**

Bu operatordan sonra yerləşən proqram fraqmentinin yerinə yetirilməsi təxirə salınır. Məsələn, massiv elementlərinin cəmini hesablayın:

```
a = [1 2 3 4 5 6 7 8 9];
```

Bu zaman 5 nömrəli indeksi bu cəmə daxil etməmək. Bunu aşağıdakı proqramda realizə edilir:

```
function sum_array
S = 0;              % cəmin başlanğıc qiyməti
a = [1 2 3 4 5 6 7 8 9]; % massiv
i=0;                % massiv indekslərinin sayğacı
while i < length(a)     % dövr ( i –nan qiymətləri
                % a massivinin uzunluğundan kiçik olana kimi çalışır)
    i=i+1;          %   indeks sayğacı bir vahid artır
```

```
    if i == 5              % Əgər indeks 5 bərabərdir isə
        continue;       % onu hesaba daxil edilmir
    end
    S=S+a(i);          % massiv elementlərinin cəmi
end                  % dövrün sonu
disp(S);              % 40 bərabər nəticəsini çap edin
```

2.5 for dövrü operatoru

Dövr operatorları müəyyən əməliyyatların bir neçə dəfə təkrar yerinə yetirilməsinə imkan verir. Bu operatorlardan proqramlaşdırmada çox geniş istifadə olunur.

<sayğac və dövr parametri> - dövrün parametri olub İNTEGER tipli dəyişəndir (eyni zamanda ixtiyari nizami tip ola bilər),

<başlanğıc qiymət> - dövr parametrinin tipində olan parametrin başlanğıc qiymətidir,

<addım>- dövr parametrinin dəyişmə addımıdır,

<son qiymət> - son qiymət dövr parametrinin tipində olan parametrin son qiymətidir,

Operator belə yerinə yetirilir: Əvvəlcə dövr parametrinə yəni sayğaca başlanğıc qiymət mənimsədilir. əgər o son qiymətdən kiçikdirsə onda idarəetmə operatora verilir, əks halda idarəetmə növbəti sətrə verilir. Növbəti mərhələdə başlanğıc qiymətin üzərinə vahid əlavə edilir və proses analoji olaraq yerinə yetirilir. Qeyd edək ki, burada ,

<başlanğıc qiymət> ≤ **<son qiymət>** şərti ödənilməlidir

For operatorunun istifadə olunma şəkli aşağıdakı kimidir:

```
for <sayğac> = <başlanğıc>:<addım>:<son qiymət>
    <dörv operatorları>
end
```

Dövr parametrini qiymətini kvadrat yüksəldən proqram fraqmentinə baxaq:

```
x = zeros(1,10);
for ii = 1:10
        x(ii) = ii^2;
end
```

Nəticə:

```
> x
```

```
x =
     1     4     9    16    25    36    49    64    81   100
```

Vektorda olan maksimal elementin tapan alqoritmin realizə edilməsində istifadə olunan for operatoruna baxaq:

```
function search_max
a = [3 6 5 3 6 9 5 3 1 0];
m = a(1);                  % cari maksimal qiymət
for i=1:length(a)   % massivin 1-ci elementindən başlayaraq addım =1
                          % olmaqla addımın var sayılan qiyməti=1
    if m < a(i)               % əgər a(i) > m,
        m = a(i);          % onda m = a(i)
    end
end                        % for   dövrün sonu
disp(m);
```

Yuxarıda verilmiş misalda for i sayğacının qiyməti 1-dən 10 kimi 1 addımı ilə dəyişir. Əgər addımın qiyməti göstərilmirsə, onda addimin qiyməti 1-ə bərabər qəbul edilir. Aşağıdakı misalda isə vektor elementlərinin qiymətlərini sağa sürüşdürən, yəni sondan bir element əvvəl duran element sona doğru sürüşdürülür. Bu alqoritmi realizə edən proqram mətni göstərilmişdir:

```
 function queue
a = [3 6 5 3 6 9 5 3 1 0];
disp(a);
for i=length(a):-1:2    % 10-dan 2 kimi 1 addımı ilə, dövr
    a(i)=a(i-1);          % a vektorunun elementlərini sürüşdürür
end                       % for dövr operatorunun   sonu
disp(a);
```

Proqramın çalışma sonucu

```
3 6 5 3 6 9 5 3 1 0
3 3 6 5 3 6 9 5 3 1
```

Yuxarida qeyd olunan misaldan görünür ki, dövr sayğacı böyükdən kiçiyə doğru istifadə edilərsə, addımın qiymətini açıq şəklində göstərmək lazımdır, bu halda addımın qiyməti-1 bərabərdir. Əgər bu edilməsə proqram doğru çalışmaz.

3. MATLAB SISTEMINDƏ FUNKSIYALARIN QRAFIKI

3.1 Funksiyaların ikiölçülü qrafiklərinin qurulması

MATLAB sistemində hesablama nəticəsində, adətən, çox böyük verilənlər massivi alınır və onu əyani vizuallaşdırmadan araşdırmaq çox çətindir. Buna görə də MATLAB-da yaradılmış vizuallaşdırma sistemi bu paketin xüsusi praktiki əhəmiyyətini artırır.

MATLAB sisteminin qrafiki imkanları güclü və müxtəlifdir. Birinci növbədə istifadəsi daha asan olan imkanları öyrənmək məqsədəuyğundur. Onları, adətən, yüksək səviyyəli qrafik adlandırırlar. Bu ad belə bir xoş faktı özündə əks etdirir: istifadəçiyə qrafiklərlə işləmənin incə və dərin gizlədilən detallarına varmağa heç bir ehtiyac yoxdur.

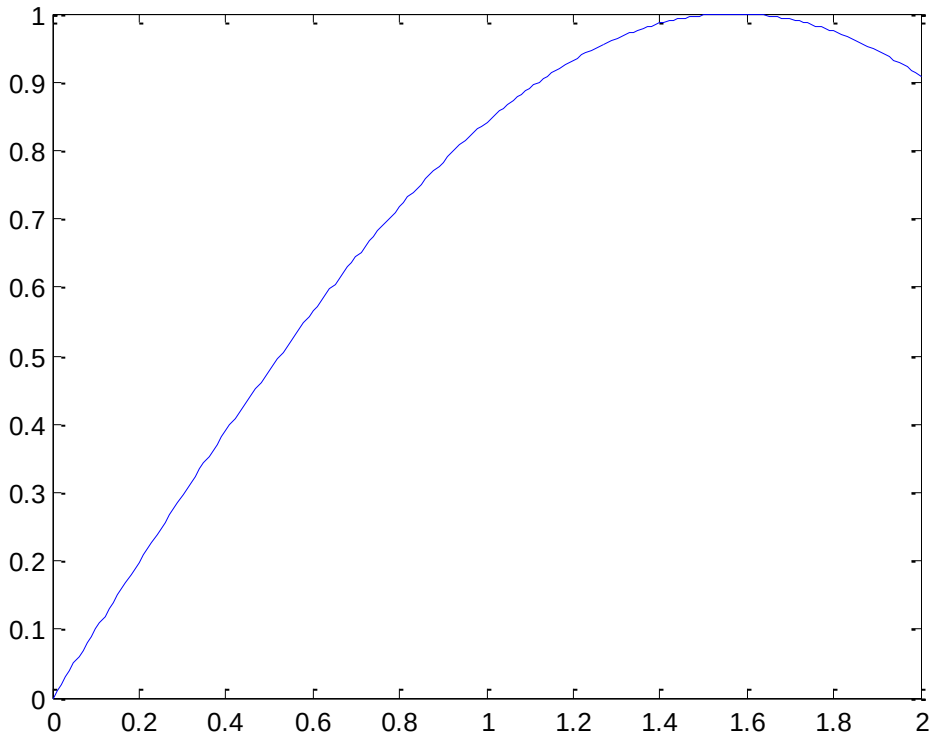
Məsələn, bir həqiqi dəyişənli funksiyanın qrafikini qurmaqdan asan heç nə yoxdur.

```
>>x = 0 : 0.01 : 2;
>>y = sin( x );
```

komandaları verilmiş arqumentlər yığını üçün sin funksiyasının qiymətlərinin y massivini hesablayır. Bundan sonra

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

yeganə komandası ilə funksiyanın kifayət qədər keyfiyyətli görünüşlu qrafikini qurmaq olur:



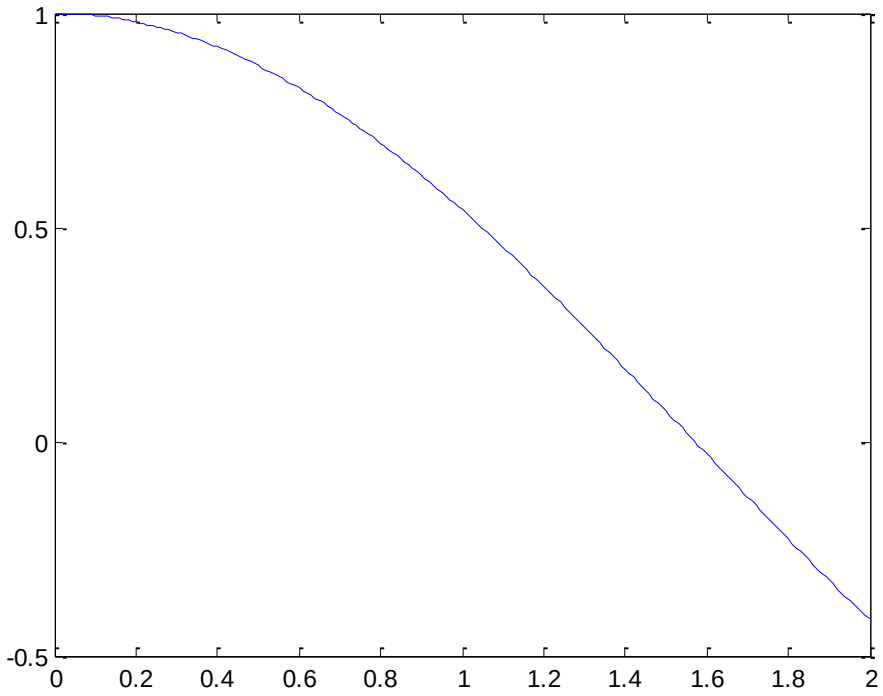
MATLAB qrafiki obyektləri, başlığında Figure (təsvir, xarici görünüş, fiqur) yazılmış xüsusi qrafiki pəncərələrdə ekrana çıxarır.

Qrafiklərin qurulması zamanı belə bir fakt üzə çıxır ki, işin çox böyük hissəsini MATLAB öz üzərinə götürür. Biz, əmrlər sətrində yalnız bir əmr yazdıq, amma sistem özü qrafiki pəncərə yaratdı, koordinat oxları qurdu, x və y dəyişənlərinin dəyişmə diapazonunu hesabladı; koordinat oxlarında bölgü nöqtələrini və onlara uyğun ədədi qiymətləri düzdü, susmaya görə hər hansı rənglə dayaq nöqtələrindən funksiyanın qrafikini keçirdi; qrafiki pəncərənin başlığında cari (hazırki, indiki) iş seansındakı qrafikin nömrəsini yazdı.

Əgər biz ekrandan birinci qrafiki kənarlaşdırmadan daha bir əmrlər yığınını daxil etsək və yerinə yetirsək,

```
x = 0 : 0.01 : 2;
z = cos( x );
plot( x , z )
```

onda həmin qrafiki pəncərədə funksiyanın yeni qrafikini almış olarıq (bu zaman pəncərədə köhnə koordinat oxları və qrafik yox olur - buna clf əmri vasitəsilə də nail olmaq olar, cla əmri isə koordinat oxlarını 0-dan 1-ə qədər standart diapazona gətirməklə yalnız qrafiki pozur):



Əgər ikinci qrafiki "birinci qrafikin üstündən" keçirmək lazımdırsa, onda ikinci **plot** qrafiki əmrini yerinə yetirməzdən əvvəl

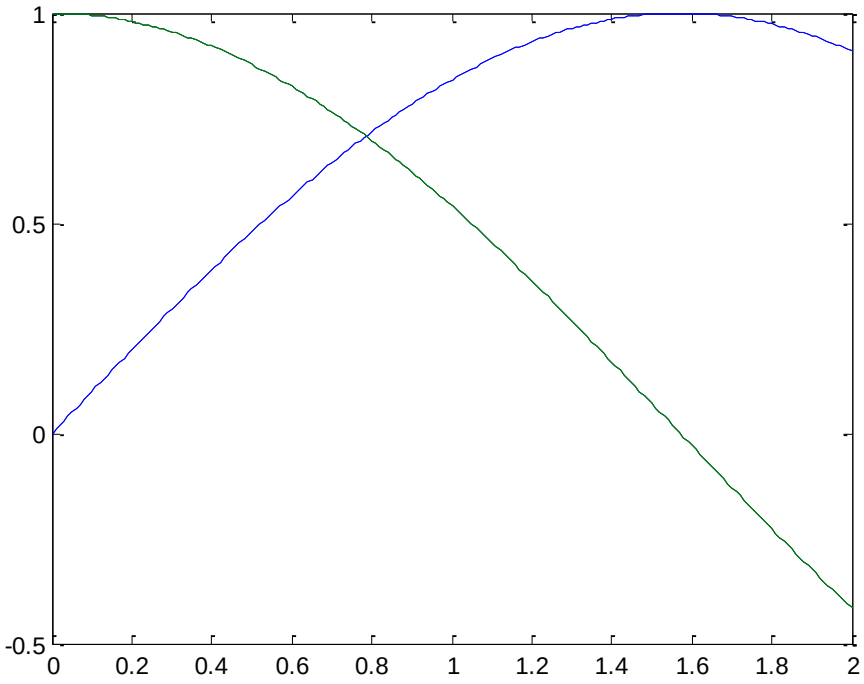
```
>>hold on
```

əmrini yerinə yetirmək lazımdır, bu əmr cari qrafiki pəncərəni saxlamaq üçün nəzərdə tutulmuşdur. Nəticədə, aşağıdakı təsvir alınır:

Buna nail olmaq üçün, həmçinin, **plot** funksiyasından eyni koordinat oxlarında bir neçə qrafiki eyni zamanda qurmağı tələb etmək lazımdır:

```
x = 0 : 0.01 : 2;
y = sin( x ); z = cos( x );
```

plot(x , y , x , z)



Bu üsulun daha bir üstünlüyü (**hold on** əmrinə qənaətdən başqa) ondan ibarətdir ki, müxtəlif qrafiklər avtomatik olaraq müxtəlif rənglərlə qurulur.

Bir neçə qrafiklərin eyni koordinat oxlarında qurulması üsullarının çatışmamazlığı ondan ibarətdir ki, iki funksiyanın müqayisə olunmayan qiymətləri üçün koordinatların dəyişmə diapazonları eyni götürülür və bu qrafiklərdən hər hansı birinin təsvirini pisləşdirir.

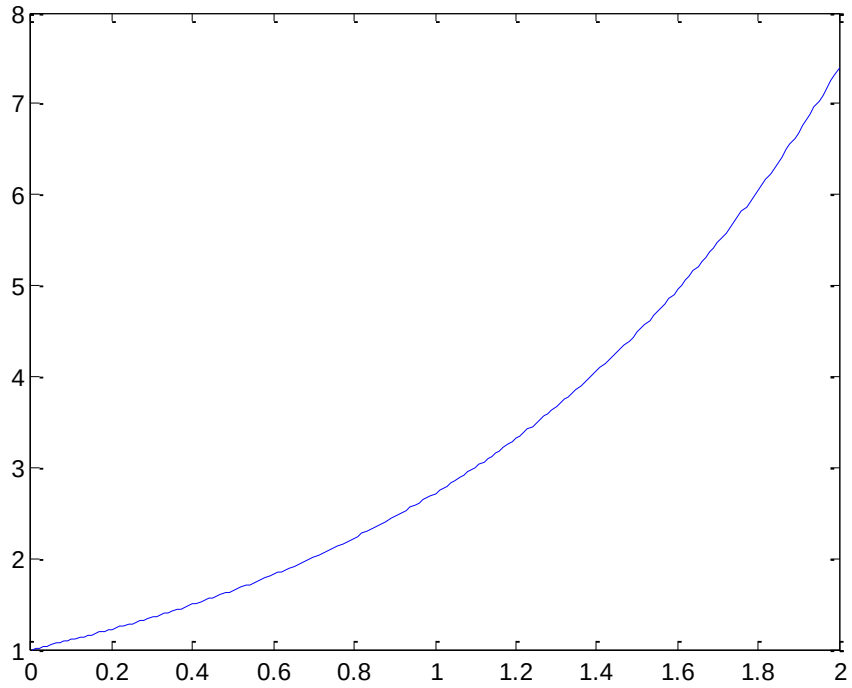
Lakin əgər yenə də bir neçə qrafiki bir-birlərinə mane olmadan eyni zamanda vizuallaşdırmaq lazım olarsa, onda bunu iki üsulla həyata keçirmək olar. Birincisi, onları müxtəlif qrafiki pəncərələrdə qurmaq olar. Məsələn, sin və cos funksiyalarının qrafiklərini eyni bir qrafiki pəncərədə quraraq (yuxarıda göstərilib) exp funksiyası üçün qiymətləri hesablayırıq:

>>w = exp(x);

Bundan sonra

>>figure; plot(x , w)

əmrini yerinə yetiririk. Bu isə exp funksiyasının qrafikini yeni qrafiki pəncərədə qurur, çünki **figure** əmri yeni (əlavə) qrafiki pəncərə yaradır və bundan sonrakı bütün qrafik qurma əmrləri onları yeni pəncərəyə çıxarır:

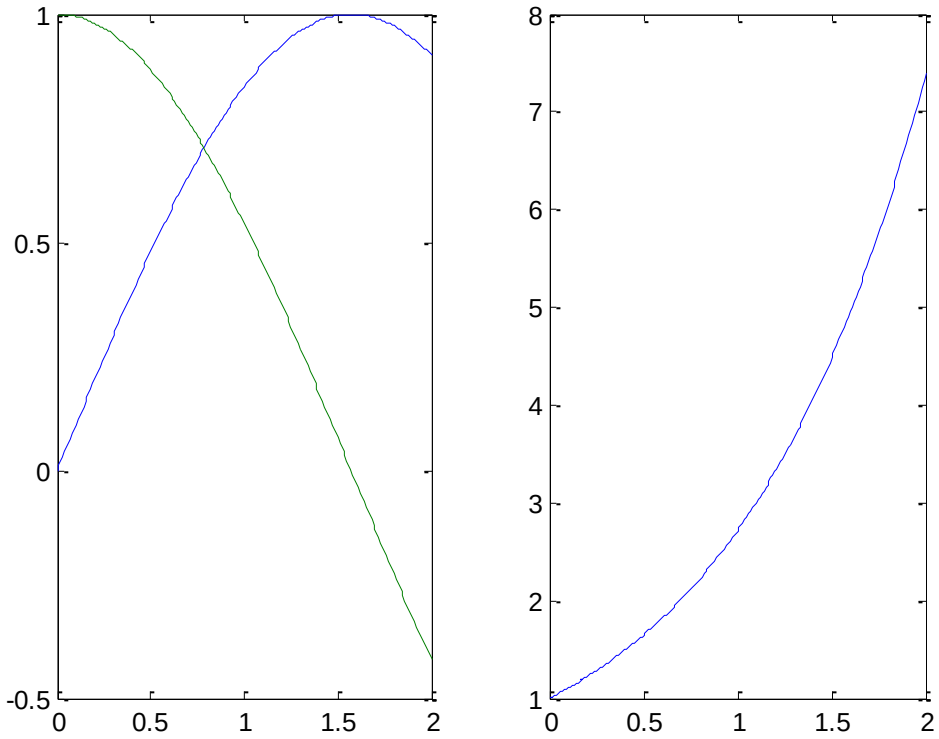


Koordinat oxlarının dəyişmə diapazonlarının münaqişəsi olmadan bir neçə qrafikin eyni vaxtda göstərilməsi məsələsinin ikinci həlli **subplot** funksiyasından istifadədir. Bu funksiya qrafiki məlumatların çıxarılması oblastını bir neçə alt oblastlara bölməyə imkan verir ki, onların hər birində müxtəlif funksiyaların qrafikini çıxarmaq olar.

Məsələn, sin, cos və exp funksiyalarının daha əvvəl hesablanmış qiymətləri üçün eyni qrafiki pəncərədə birinci iki funksiyanın qrafikini birinci alt oblastda, üçüncü funksiyanın qrafikini isə ikinci oblastda qurmaq olar:

```
>>subplot(1,2,1); plot(x,y,x,z)
>>subplot(1,2,2); plot(x,w)
```

və nəticədə aşağıdakı şəkildə qrafiki pəncərə alırıq:



Bu alt oblastların koordinat oxları dəyişənlərinin dəyişmə diapazonları bir-birlərindən asılı deyildir.

subplot funksiyası üç ədədi arqumentə malikdir, bunlardan birincisi alt oblast sıralarının sayına, ikincisi alt oblast sütunlarının sayına, üçüncüsü isə altoblastın nömrəsinə (nömrə yeni sıraya keçid boyunca sonadək hesablanır) bərabərdir.

Əgər qrafik üçün bir və ya hər iki koordinat oxları üzrə dəyişənlərin dəyişmə diapazonu çox böyükdürsə, onda funksiyaların qrafiklərini qurmaq üçün loqarifmik miqyasdan istifadə etmək olar. Bunun üçün **semilogx, semilogy** və **loglog** funksiyaları verilmişdir. Bu funksiyaların istifadəsi üzrə geniş məlumatı

>>help ad_ funksiyanın adı

əmrinin köməyi ilə almaq olar. Əmr klaviaturanın köməyi ilə yazılır və MATLAB sisteminin əmrlər pəncərəsində yerinə yetirilir.

Beləliklə, baxılan misallardan görünür ki, MATLAB-ın yüksək səviyyəli qrafiki altsistemi istifadəçidən çox iş tələb etmədən qrafik qurmanın müxtəlif hallarının öhdəsindən asanlıqla gəlir. Daha bir misala - polyar koordinatlarında qrafiklərin qurulmasına baxaq. Məsələn, t=sin(3ϕ) funksiyasının qrafikini polyar koordinatlarında qurmaq üçün

```
>>phi = 0 : 0.01 : 2*pi; r = sin( 3* phi );
>>polar( phi , r )
```

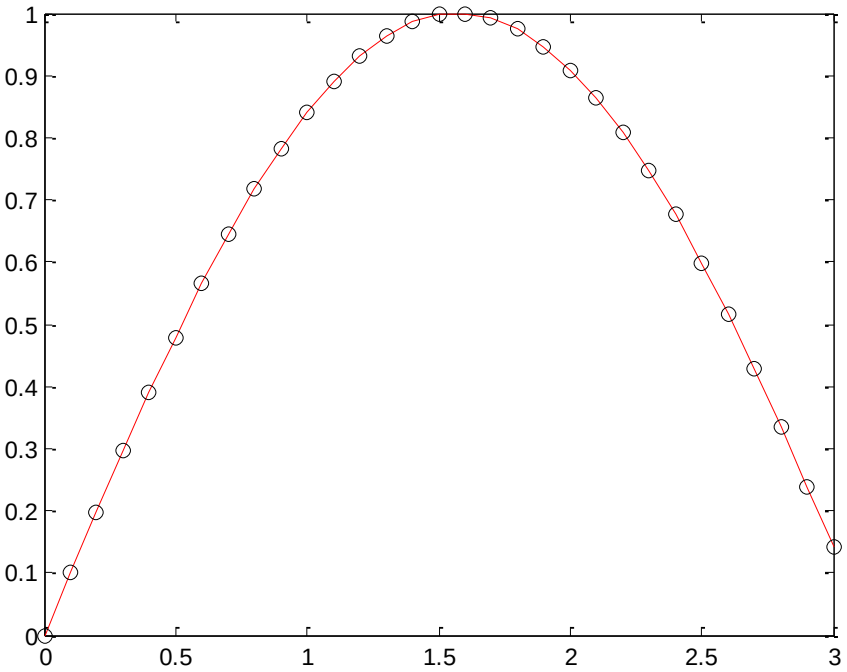
əmrlərini yerinə yetirmək lazımdır.

3.2 Funksiyaların qrafiklərinin tərtibatı

Indi isə funksiyaların qrafiklərinin xarici görünüşləri ilə - qrafiki əyrilərin rəngi və stili, həmçinin qrafiki pəncərə daxilində müxtəlif yazılışlarla bağlı bir sıra məsələlərə baxaq. Məsələn,

```
>>x = 0 : 0.1 : 3; y = sin( x );
>>plot( x, y, 'r-', x, y, 'ko' )
```

əmrləri vasitəsilə qrafikə bütöv qırmızı əyri görkəmi vermək olar, Əyri üzərində hesablanmış diskret nöqtələrə isə qara çevrələr qoyulur:



Burada **plot** funksiyası eyni funksiyanın iki müxtəlif stildə qrafikini qurur. Bu stillərdən birincisi 'r-' kimi işarə olunub, əyrinin qırmızı olması r hərfi, bütöv olması isə ştrix ilə verilmişdir. Ikinci **stil** isə 'ko' kimi işarə olunub, hesablanmış nöqtələrdə qara rənglə (k hərfi) çevrələrin (o hərfi) qoyulmasını göstərir.

Ümumi halda

```
>>plot( x1, y1, s1, x2, y2, s2, … )
```

funksiyası bir neçə y1(x1), y2(x2),… funksiyalarının qrafiklərini s1, s2, … stilləri vasitəsilə birləşdirməyə imkan verir.

```
>>plot( x1, y1, s1, x1, y1, s2 )
```

şəklində funksiya isə yeganə y1(x1) funksiyasının qrafikinin əyrisini bir rənglə, üzərindəki nöqtələri (hesablanmış nöqtələri) isə başqa rənglə qeyd edir.

s1, s2,… stilləri tək dırnaq arasında üç marker simvolları yığını şəklində verilir. Bu markerlərdən birincisi (ardıcıllıq mütləq deyil) xəttin növünü müəyyənləşdirir:

Marker	Əyrinin növü
-	kəsilməz
--	ştrixli
:	punktirli
-.	ştrix-punktirli

Ikinci marker rəngi müəyyənləşdirir:

Marker	Əyrinin rəngi
c	Mavi
m	Bənövşəyi
y	Sarı
r	Qırmızı
g	Yaşıl
b	Göy
w	Ağ
k	Qara

Sonuncu marker, qoyulmuş ”nöqtələrin” tipini müəyyənləşdirir:

Marker	Nöqtənin növü
.	Nöqtə
+	Plyus
*	Ulduz
o	Çevrə
x	Xaç

Hər üç markeri göstərməmək də olar. Bu zaman lazım olduqda “susmaya görə” qurulmuş markerlərdən istifadə olunur. Markerlərin verilmə ardıcıllığı əhəmiyyətli deyil, belə ki, 'r+-' və '-+r' eyni nəticə verir.

Əgər stil sətrində nöqtənin növü üçün marker qoysaq, əyrinin növü üçün isə marker qoymasaq, onda hesablanmış nöqtələr əks olunacaq, lakin onlar kəsilməz əyrilərlə birləşdirilməyəcək.

Funksiyaların qrafiklərinin tərtibatının (və başqa qrafiki işlərin yerinə yetirilməsinin) daha geniş üsulu **deskriptor** üsuldur. Bu üsulun tam öyrənilməsi MATLAB sisteminin aşağısəviyyəli qrafikasına aiddir və bu vəsaitin çərçivəsindən kənara çıxır. Lakin biz indi (və daha sonra) bəzi sadə misallara baxacağıq.

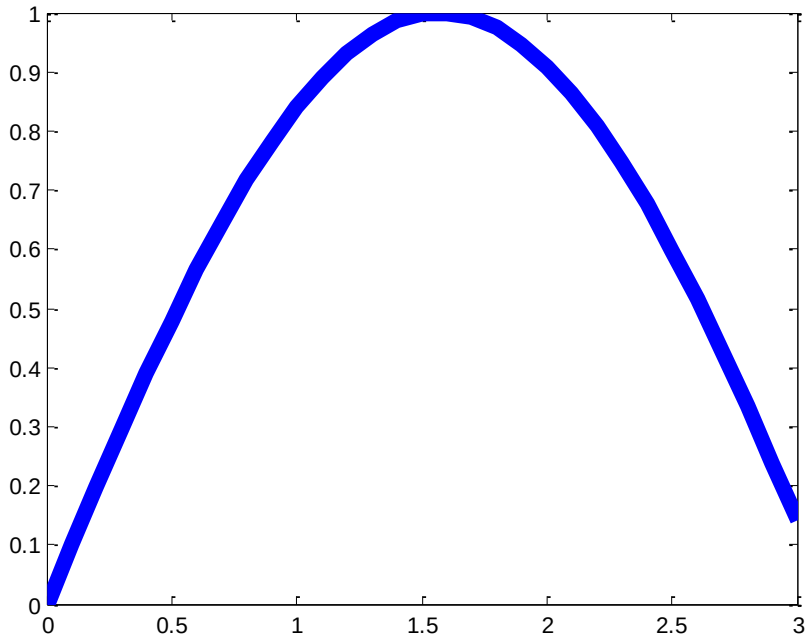
Yuxarıda biz sin funksiyasının qrafikini kəsilməz qırmızı əyrinin və qara dairəciklərin köməyi ilə tərtib etdik. Indi isə onu yalnız kəsilməz, lakin çox qalın əyri vasitəsilə tərtib etməyə cəhd edək. Bunu necə etmək olar? Deskriptor qrafikası bazasında sadə həllə baxaq:

```
>>x = 0 : 0.1 : 3; y = sin( x );
```

```
>>hPlot = plot( x, y );
>>set( hPlot, 'LineWidth', 7 );
```

plot funksiyası koordinatları x, y olan dayaq (hesablanmış) nöqtələrindən düz xətt parçaları keçirir. MATLAB sistemində düz xətlər **Line** növlü qrafiki obyekt kimi təsvir olunur. Bu obyektlər dəyişdirilə bilən çoxsaylı xassə və xüsusiyyətlərə malikdir. Bu obyektlərə giriş onların təsvirlərilə (deskriptorlarla, handles) mümkündür.

Bizim qrafikin qurulması üçün istifadə olunan **Line** obyektinin təsviri **plot** funksiyası vasitəsilə qayıdır. Biz onu daha sonra **hPlot** dəyişənində istifadə etmək üçün yadda saxlayırıq. Sonra bu təsvir konkret qrafiki obyekti tanımaq üçün **set** funksiyasına təklif edilir. Məhz belə tanınmış obyektin **set** funksiyasını çağırdıqda başqa arqumentlərdə göstərilən xüsusiyyətlərini **set** funksiyası dəyişir. Bizim misalda **'LineWidth'** (əyrinin qalınlığı) xassəsi göstərilib və onun üçün yeni 7 qiyməti verilib (susmaya görə isə 0.5). Nəticədə, aşağıdakı şəkil alınır:



Qrafiki obyektin istənilən parametrinin (atributunun, xüsusiyyətinin) cari qiymətini **get** funksiyasının köməyi ilə bilmək olar. Məsələn, əgər şəkildə göstərilən qrafiki aldıqdan sonra

```
>>width = get( hPlot, 'LineWidth' )
```

əmrini daxil etsək və yerinə yetirsək, onda **width** dəyişəni üçün 7 qiyməti alınar.

Indi isə əyrinin bilavasitə tərtibatından koordinat sistemi oxlarının, oxların adlarının yazılışının və s. tərtibatına keçək. MATLAB üfüqi ox üzrə sərhədləri asılı olmayan dəyişənin verilmiş sərhəd qiymətlərinə bərabər qəbul edir. Şaquli ox üzrə asılı dəyişən üçün MATLAB funksiyanın dəyişmə diapazonunu hesablayır. Sonra bu hesablanmış diapazon koordinat sisteminin şaquli oxunda qeyd olunur və sanki funksiyanın qrafiki düzbucaqlı daxilinə çəkilmiş şəkildə olur.

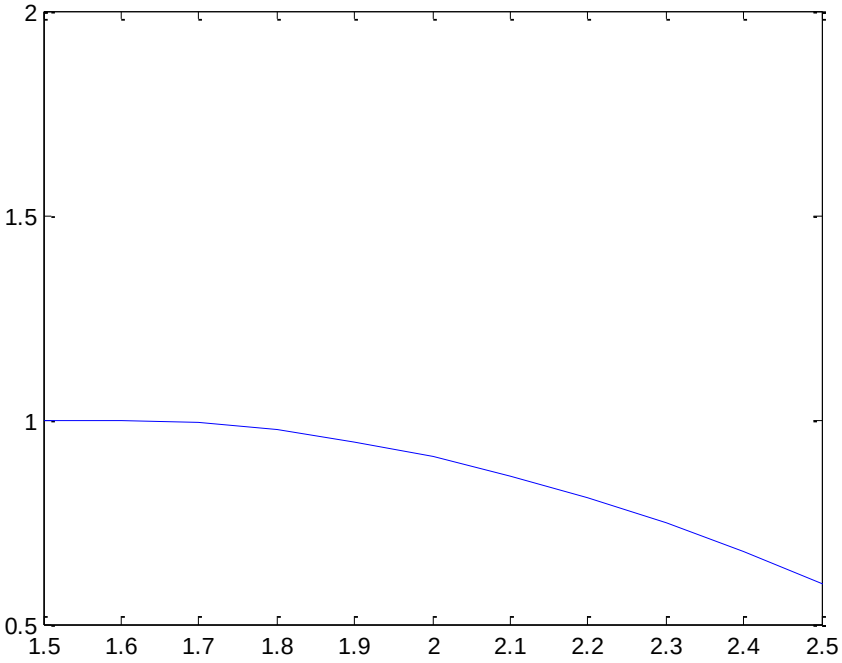
Əgər MATLAB sistemində qrafiklərin qurulması zamanı miqyaslaşdırmanın bu xüsusiyyətindən imtina etmək istəyiriksə, onda biz mütləq koordinat oxları üzrə dəyişənlərin dəyişmə sərhədlərini aşkar şəkildə verməliyik. Bu

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

funksiyasının köməyi ilə verilir. Qrafik qurulduqdan sonra təsvirə baxaraq onun vizual görünüşünü yaxşılaşdırmaq məqsədilə bu əmri istənilən dəfə yerinə yetirmək olar. Əmr klaviatura vasitəsilə daxil edilir. Belə miqyaslaşdırma konkret araşdırma zamanı qrafikin daha çox maraq doğuran hissələrini daha aydın təsvir etməyə imkan verir. Məsələn, sin funksiyasının daha əvvəl alınmış qrafikində sinusoidin təpələrini daha aydın görmək üçün koordinat oxları üzrə sərhədləri

>>axis([1.5, 2.5, 0.5, 2])

əmri vasitəsilə daraltmaq lazımdır:



Təsvirin miqyasının böyüdülməsi üsulundan, adətən, tənliklərin qrafiki həlli zaman tənliyin təqribi kökünü yüksək dəqiqliklə almaq üçün istifadə olunur.

Indi isə koordinat oxları üzərindəki bölgülərin sayını artıraq. Bəzən onlar kifayət qədər olmur (sonuncu şəklin üfüqi oxunda bu bölgülərin sayı üçdür - 1.5, 2 və 2,5 qiymətləridə).

Koordinat oxlarında bölgülərin sayını **set** funksiyası vasitəsilə dəyişmək olar. Bu funksiya **Axes** qrafiki obyektini emal edir. Bu obyekt koordinat oxlarından və daxilində funksiyanın qrafiki yerləşəcək ağ düzbucaqlıdan ibarətdir. Belə obyektin yazılışını almaq üçün parametrlərsiz çağrılan **gca** funksiyası tətbiq edilir.

Nəticədə, qrafik qurulandan sonra yerinə yetirilən

>>set(hAxes, 'xtick', [1.5, 1.75, 2.0, 2.25, 2.5])

kod fraqmenti üfüqi koordinat oxu üzərində yeni bölgülər yaradır (beş ədəd).

Alınmış şəkil üzərində müxtəlif yazıları yerləşdirmək üçün **xlabel, ylabel, title** və **text**

funksiyalarından istifadə edilir. **xlabel** funksiyası üfüqi oxun, **ylabel** funksiyası isə şaquli oxun adını yazmaq üçündür (bu yazılar koordinat oxu boyunca istiqamətlənir).

Əgər yazını rəsmin ixtiyari bir yerində yerləşdirmək nəzərdə tutulubsa, onda **text** funksiyasını daxil edirik:

```
>>text( x, y, 'some text')
```

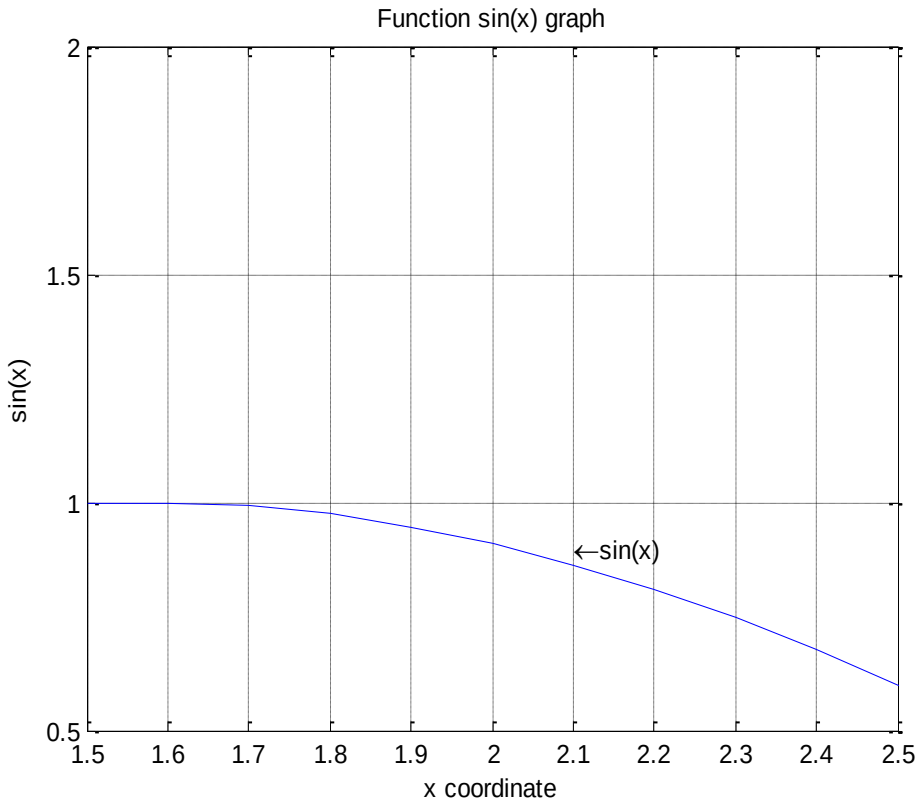
Qrafikin ümumi başlığı **title** funksiyası ilə verilir. Bundan başqa,

```
>>grid on
```

əmrindən istifadə etsək qrafikin qurulma oblastını ölçü toru ilə örtmək olar. Bütün bu vasitələri tətbiq edərək

```
>>title( 'Function sin(x) graph' );
>>xlabel( 'x coordinate' ); ylabel( 'sin(x)' );
>>text( 2.1, 0.9, '\leftarrowsin(x)' ); grid on;
```

qrafiki aşağıdakı şəklə salmaq olar:



text funksiyası başlığı koordinatları ilk iki arqumentlə verilmiş nöqtədən başlayaraq yerləşdirir. Xüsusi simvollar mətnə \ simvolundan (“tərs çəp xətt”) sonra daxil edilir. Nümunədə “sola ox” xüsusi simvolunu bu şəkildə daxil etmişik. Xüsusi simvolların işarələri elmi mətnlərin hazırlandığı **TeX** sistemindəki uyğun simvollarla üst-üstə düşür.

3.3 Üçölçülü qrafik

Ikidəyişənli funksiyaların qrafikləri funksiyaların təyin oblastları üzərində asılmış

səthlərin hissələri kimi əks olunur. Buradan aydın olur ki, ikidəyişənli funksiyaların qrafiklərinin təsviri "üçölçülü qrafikin" kompüterin müstəvi ekranında realizə olunmasını tələb edir.

MATLAB-ın yüksək səviyyəli qrafiki altsistemi üçölçülü qrafikini istifadəçidən xüsusi güc sərfi tələb etmədən avtomatik realizə edir.Tutaq ki, koordinatları x1, y1 olan nöqtədə **z=f(x,y)** funksiyasının qiyməti hesablanmış və z1-ə bərabər olumuşdur. Başqa bir x2, y2 nöqtədəsində (yəni arqumentlərin başqa qiymətlərində) funksiyanın z2 qiyməti hesablanır. Bu prosesi davam etdirərək üçölçülü fəzada yerləşmiş N sayda **(x1, y1, z1), (x2, y2, z2), ..., (xN, yN, zN)** nöqtələr massivi (yığını) alınır. MATLAB sisteminin xüsusi funksiyaları bu nöqtələrdən hamar səthlər keçirir və onların proyeksiyasını kompüterin müstəvi ekranına əks etdirir.

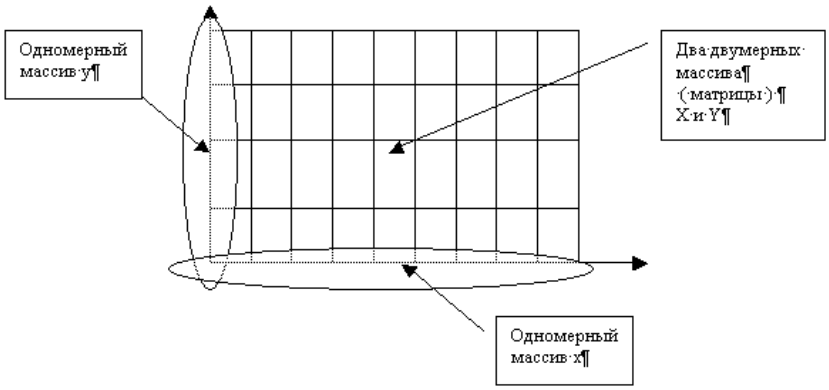
Adətən, arqumentlərin nöqtələri funksiyanın təyin oblastında müntəzəm şəkildə düzbucaqlı tor (yəni matris) kimi yerləşir. Nöqtələrin bu şəkildə verilmiş toru eyni strukturlu iki matris yaradır: birinci matrisdə bu nöqtələrin birinci koordinatlarının (x - koordinatının) qiymətləri, ikinci matrisdə isə ikinci koordinatlarının (y - koordinatının) qiymətləri yerləşir. Birinci matrisi X, ikincini isə Y işarə edək. Bundan əlavə, üçüncü matris - bu arqumentlərdə z=f(x,y) funksiyasının qiymətləri matrisi də vardır. Bu matrisi Z hərfi ilə işarə edək.

MATLAB sistemində ikidəyişənli funksiyanın qurulması üçün ən sadə funksiya

>>plot3(X , Y , Z)

funksiyasıdır, burada **X, Y** və **Z** – eyni ölçülü matrislərdir, onların mənasını izah etmişik.

MATLAB sistemində birölçülü x, y massivləri üzrə ikiölçülü X və Y matrislərini almaq üçün xüsusi funksiya vardır.



Tutaq ki, x oxu üzrə qiymətlər diapazonu

>>u = -2 : 0.1 : 2

vektoru şəklində, y oxu üzrə qiymətlər diapazonu isə

>>v = -2 : 0.1 : 1

vektoru şəklində verilmişdir.

Alınan düzbucaqlı torun nöqtələrinin birinci və ikinci koordinatlarını təsvir edən X və Y matrislərini almaq üçün MATLAB sistemində

>> [X , Y] = meshgrid(u, v)

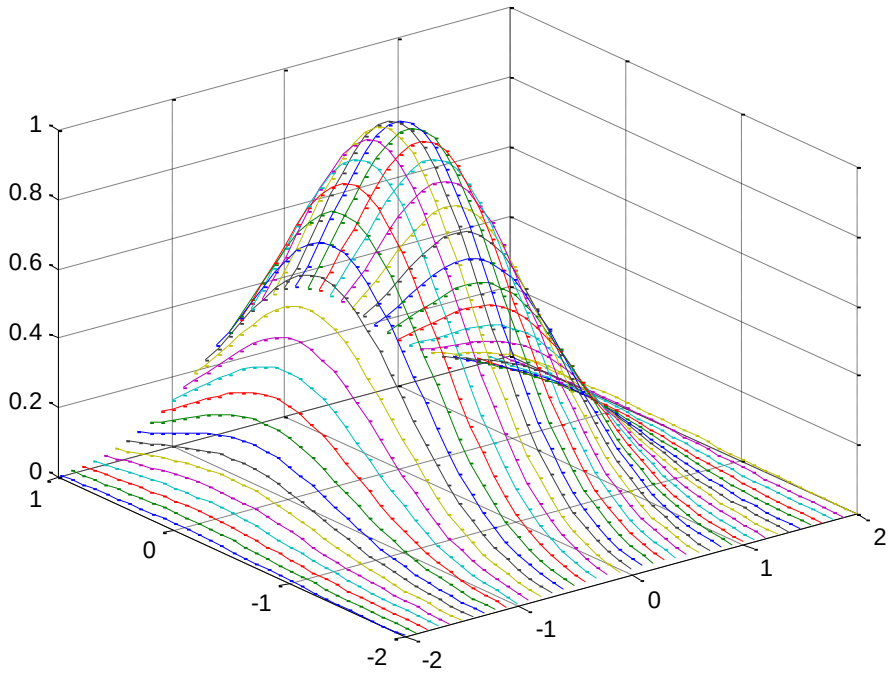
xüsusi funksiyasından istifadə edilir.

Gördüyümüz kimi, bu funksiya girişdə koordinat oxları üzərində nöqtələr massivi şəklində iki birölçülü massiv (vektor) alır, və tez iki axtarılan ikiölçülü massivi qaytarır, məsələn, exp funksiyası:

>>Z = exp(- X.^2 - Y.^2)

Nəhayət, yuxarıda verilən **plot3** funksiyasını tətbiq edərək bu funksiyanın üçölçülü qrafiki təsvirini alırıq:

```
>> u = -2 : 0.1 : 2;
>> v = -2 : 0.1 : 1 ;
>> [ X , Y ] = meshgrid( u, v );
>> Z = exp( - X.^2 - Y.^2 );
>> plot3( x, y, z );grid on
??? Undefined function or variable 'x'.

>> plot3( X, Y, Z );grid on
>>
```



Bu şəkildən görünür ki, **plot3** funksiyası qrafiki fəzada əyrilər yığını şəklində qurur. Bu əyrilərdən hər biri üçölçülü səthin yOz müstəvisinə paralel müstəvilərlə kəsişməsindən alınır. Başqa sözlə belə demək olar ki, hər bir əyri koordinatları X, Y və Z matrislərinin eyni sütunlarından götürülmüş nöqtələr yığınını birləşdirən düz xətt parçalarından alınır. Yəni, birinci əyri X, Y, Z matrislərinin birinci sütunlarına, ikinci əyri ikinci sütunlarına və i.a. uyğun gəlir.

Parametrik şəkildə verilmiş üçölçülü əyriləri qurmaq üçün **plot3** funksiyasının digər çağırış formasından istifadə edilir:

>>plot3(x, y, z)

burada x, y və z düz xətt parçaları ilə birləşdiriləcək nöqtələrin koordinatlarının birölçülü

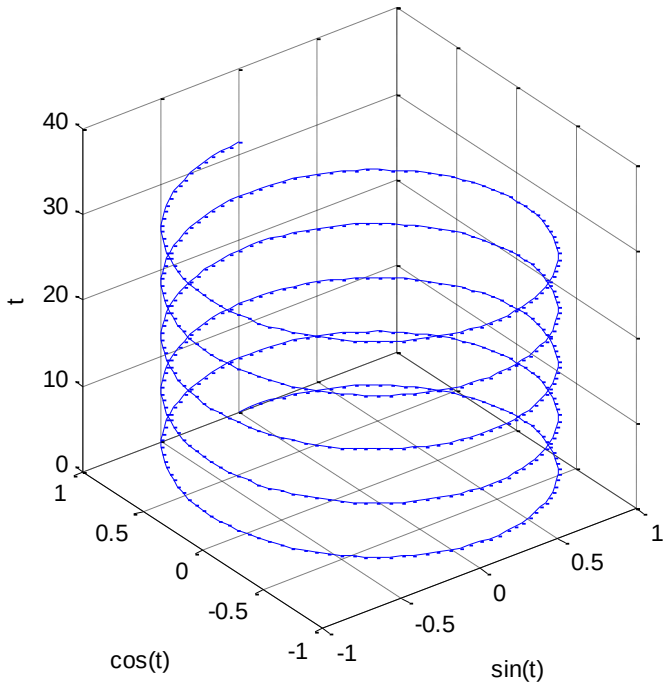
massivləridir. Məsələn, müstəvi qrafiklərdən məlum olan və funksiyanın qrafikinin qurulma oblastında koordinat qiymətləri torununun qurulması üçün tətbiq edilən (həmçinin daha əvvəl "müstəvi" halda qrafiklərin tərtibatında baxılan əmr və funksiyalardan da istifadə etmək olar)

```
>>grid on
```

əmrinin iştirakı ilə verilmiş

```
>>t = 0:pi/50:10*pi;
>>plot3(sin(t),cos(t),t)
>>xlabel('sin(t)')
>>ylabel('cos(t)')
>>zlabel('t')
>>grid on
>>axis square
```

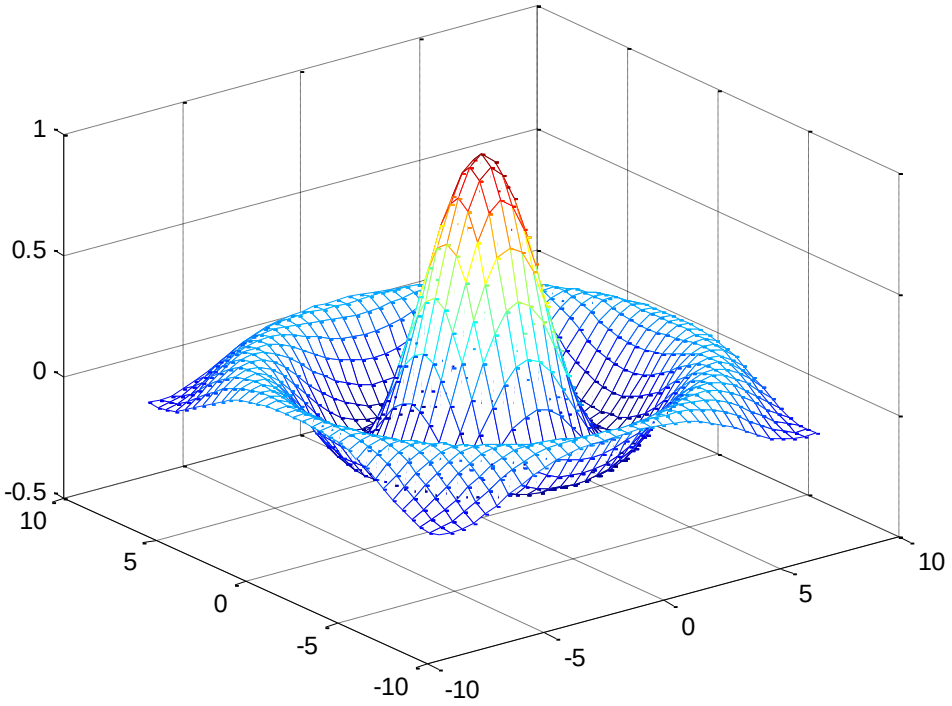
kod fraqmenti təsviri aşağıdakı şəkildə verilmiş vintvari əyrini qurmağa imkan verir.



Bundan başqa MATLAB sistemində üçölçülü qrafiklərin daha real təsvirinə nail olmağa imkan verən bir sıra funksiyalar vardır. Bu **mesh, surf** və **surfl** funksiyalarıdır.

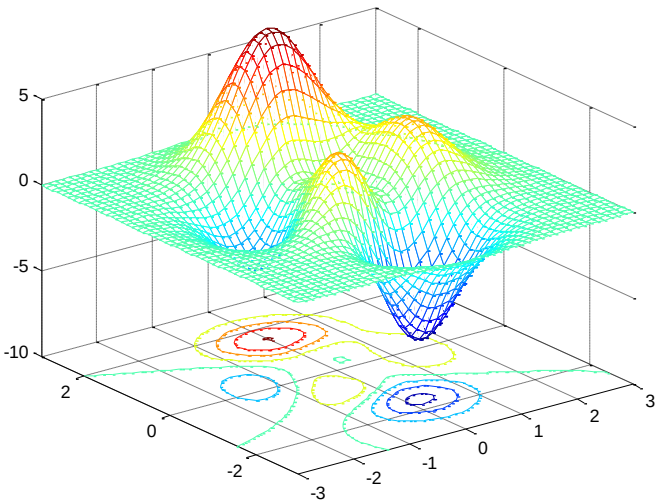
mesh funksiyası qrafiki səthin hesablanmış qonşu nöqtələrini düz xətt parçaları ilə birləşdirir və MATLAB sisteminin qrafiki pəncərəsində bu "karkas-qabırğalı" (ingiliscə **wireframe mesh**) həcmli cismin müstəvi proyeksiyasını göstərir.

```
>>figure
>> [X,Y] = meshgrid(-8:.5:8);
>>R = sqrt(X.^2 + Y.^2) + eps;
>>Z = sin(R)./R;
>>mesh(X,Y,Z)
```



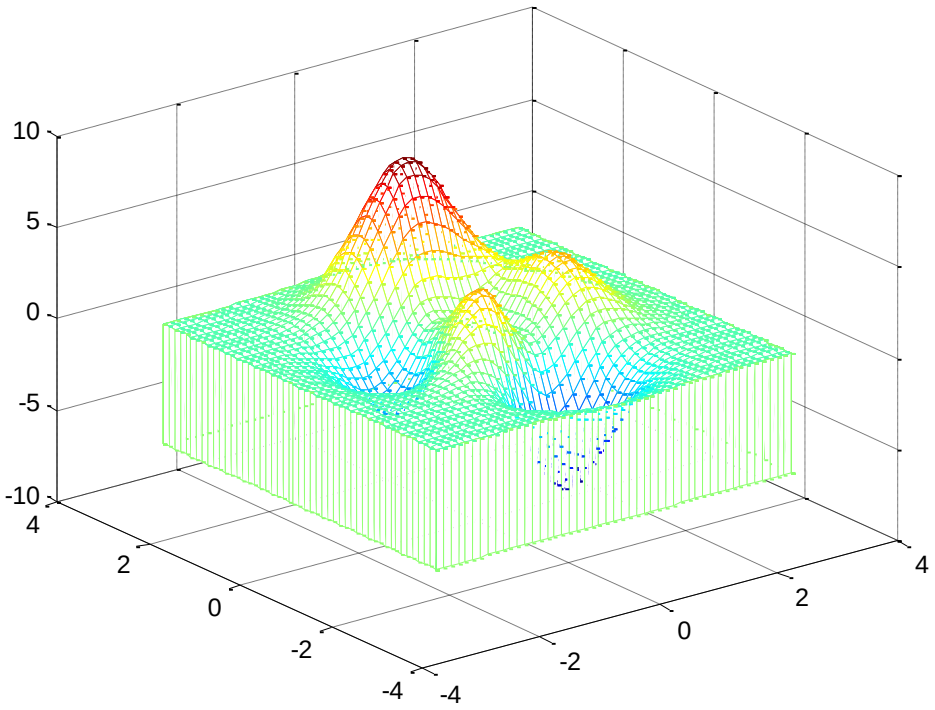
Aşağıdakı misalda **mesh** və **contour** əmrlərinin birgə istifadəsi göstərilmişdir.

```
>> [X,Y] = meshgrid(-3:.125:3);
>>Z = peaks(X,Y);
>>meshc(X,Y,Z);
>>axis([-3 3 -3 3 -10 5])
```

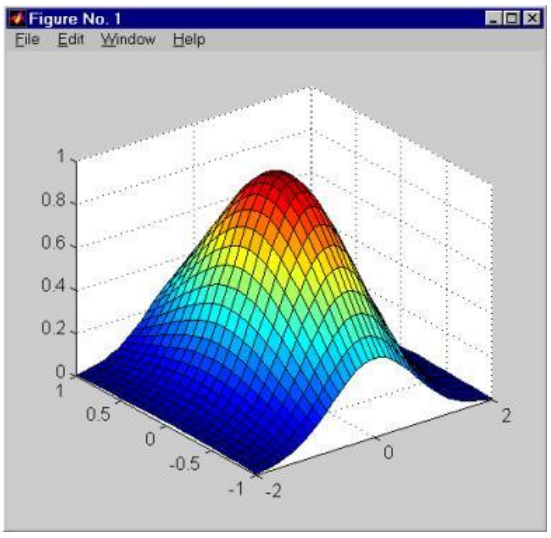


"waterfall", yəni şəlalə effektin yaratmaq üçün **meshz** funksiyasından istifadə edilmişdir:

```
>> [X,Y] = meshgrid(-3:.125:3);
>>Z = peaks(X,Y);
>>meshz(X,Y,Z)
```



"Həcmliliyi" daha yaxşı görmək üçün təsvirdə müxtəlif qabırğalar avtomatik olaraq müxtəlif rənglərlə boyanır. Bundan əlavə (plot3 funksiyasından fərqli olaraq) görünməz xətləri yox etmək mümkündür. Əgər siz hesab edirsinizsə ki, təsvir olunan qabırğalı cisim şəffafdır və arxa xətlər gizlənməməlidir, onda **hidden off** əmrini daxil edin və bundan sonra bu xətlər təsvirdə görünəcəklər. Səthin daha sıx təsvirini **mesh** funksiyası əvəzinə **surf(X,Y,Z)** funksiyasını tətbiq etməklə əldə etmək olar. Nəticədə sıx (qeyri-şəffaf) torvari səthi əks etdirən aşağıdakı şəkli alırıq, bu torvari səthin ayrı-ayrı xanaları (oyuq, yuva) (üzləri) (müstəvi dördbucaqlılar) avtomatik olaraq müxtəlif rənglərə boyanır.



surf funksiyasının köməyi ilə süni rənglənmiş olsalar da, çox aydın görünüşlü təsvirlər alınır. Əgər biz səthlərin daha təbii və obyektiv rənglənmə üsullarını əldə etmək istəyiriksə,

onda **surf** funksiyasından istifadə etməliyik.

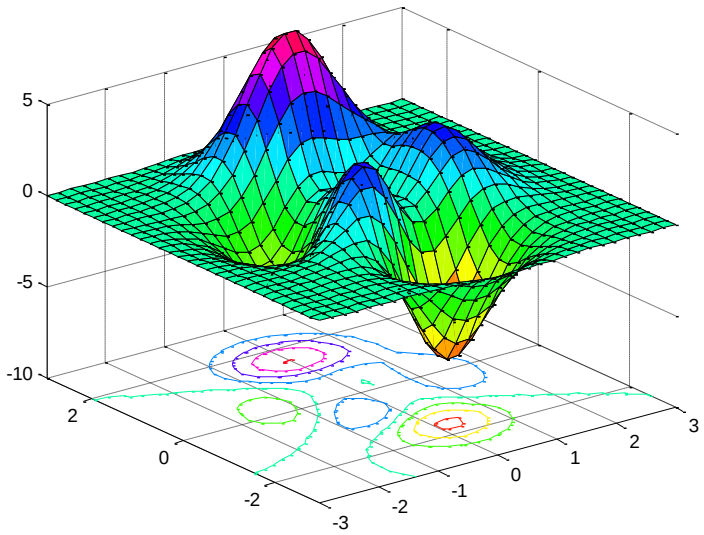
surf funksiyası səthin qrafikini müəyyən fiziki işığı qaytarma xassələrinə malik maddi səth kimi izah edir. Susmaya görə belə maddi səthi işıqlandıran hər hansı işıq mənbəyi verilir, bundan sonra şərti kameranın obyektinə düşən və əks olunan şüaların trayektoriyaları hesablanır. Məhz belə kameradakı təsvir MATLAB sisteminin qrafiki pəncərəsində göstərilir.

Müxtəlif materiallar düşən şüaları müxtəlif cür əks etdirdiklərindən daha yaxşı təsvir üçün (istifadəçinin nöqteyi-nəzərincə) lazım olan hər hansı material seçmək olar. Xüsusi halda **colormap(copper)** funksiyasından istifadə edərək qrafikin təsviri üçün misin (mis inciliscə - copper) səthindən əks olunan işıq üçün xarakterik olan rənglər yığınını (ingiliscə - colormap) seçmək olar. Bundan sonra *surf(X,Y,Z)* funksiyasının yerinə tətbiq olunan

>>surfl(X, Y, Z)

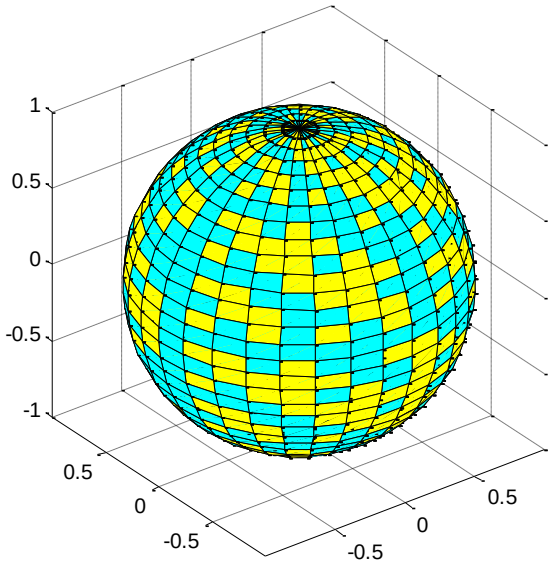
funksiyası daha təbii və aydın görünüşlü olan qrafikin alınmasına imkan verir.

```
>> [X,Y,Z] = peaks(30);
>>surfc(X,Y,Z)
>>colormap hsv
>>axis([-3 3 -3 3 -10 5])
```



Color a sphere with the pattern of +1s and -1s in a Hadamard matrix.

```
>>k = 5;
>>n = 2^k-1;
>> [x,y,z] = sphere(n);
>>c = hadamard(2^k);
>>surf(x,y,z,c);
>>colormap([1   1   0; 0   1   1])
>>axis equal
```



Bu cür qrafikdən qabırğaları əks etdirən qara xətləri yox etmək olar, bundan başqa əgər **shading interp** əmrini yerinə yetirsək, səthin işıqlanmasının daha yumşaq keçidinə nail ola bilərik. Bu əmr bildirir ki, indi rəng (işıqlanma) hətta ayrı-ayrı üzlərin (xanaların) daxilində də dəyişəcək. Nəticədə, artıq hər hansı həcmli fiqurun daha real təsviri alınacaq. Bunun ikidəyişənli funksiyaların qrafiklərinin təsviri məsələsi üçün yaxşı yoxsa pis olduğunu yalnız istifadəçi deyə bilər.

3.4 Kameranın vəziyyəti və üçölçülü qrafiklərin fırlanması

Üçölçülü qrafiklərin tərtibatının bir çox priyomları (üsulları) birdəyişənli funksiyaların müstəvi qrafiklərini öyrənərkən baxdığımız priyomlarla üst-üstə düşür. Xüsusi halda miqyaslaşdırma üçün **axis** funksiyasından istifadə etmək daha əlverişlidir, üçölçülü halda bu funksiya üç cüt skalyar arqument qəbul edir:

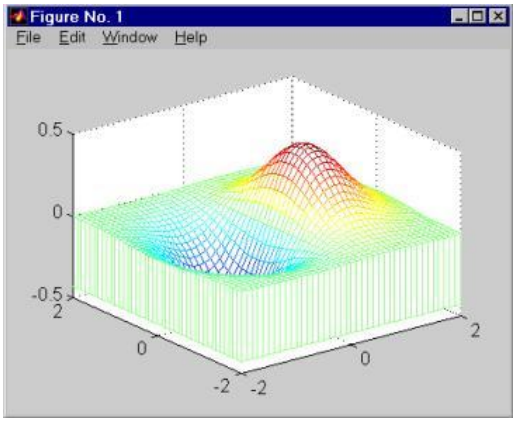
>>axis([xmin, xmax, ymin, ymax, zmin, zmax])

Əvvəlki kimi **text, xlabel, ylabel, zlabel, title** funksiyalarından və koordinat oxlarında bölgülər qoymaq üçün **set** funksiyasından istifadə etmək olar. Həmçinin **subplot** funksiyasının köməyi ilə eyni bir qrafiki pəncərədə bir neçə üçölçülü qrafiki yerləşdirmək olar.

Üçölçülü qrafiklərin tərtibatının yeni üsullarına **mesh** funksiyasının z və c suffiksləri ilə (**meshz** və **meshc**) çağırışını, **surf** funksiyasının isə ***c*** suffiksi ilə çağırışını əlavə etmək olar. ***z*** suffiksindən istifadə "qrafikin ***pyedestalla***" qurulmasına imkan verir. Məsələn,

```
>> [X,Y] = meshgrid( -2 : 0.1 : 2 );
>>Z = X .* exp( - X.^2 - Y.^2 );
>>meshz( X, Y, Z )
```

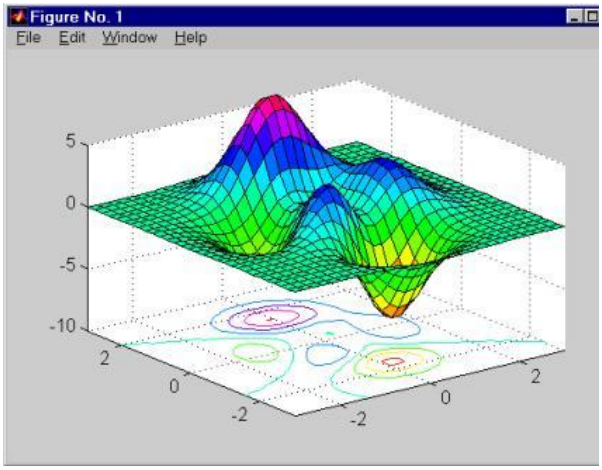
kod fraqmenti aşağıdakı qrafiki qurur:



c suffiksli funksiyalar isə məxsusi üçölçülü qrafikdən əlavə səviyyə xətlərini də qururlar. Məsələn,

```
>> [X,Y,Z] = peaks(30); surfc(X,Y,Z);
>>colormap( hsv ); axis([-3 3 -3 3 -10 5]);
```

fraqmenti aşağıdakı təsviri verir:



peaks adlı funksiya (standart qauss funksiyalarının müəyyən miqyaslaşdırılmış kombinasiyasıdır) adətən, MATLAB-ın məlumat (sorğu) sistemində qrafiki funksiyaların əyani təsviri üçün tətbiq olunur.

Nəhayət, üçölçülü qrafiklər üçün qrafikin baxış nöqtəsini, yəni şərti kameranın vəziyyətini dəyişmək imkanı mövcuddur. Kameranın vəziyyəti azimut bucağı (adətən **az** işarə edilir) və qaldırılma bucağı (adətən **el** işarə edilir) ilə təyin olunur. Birinci bucağın dəyişdirilməsi xOy müstəvisinin Oz oxu ətrafında saat əqrəbinin əksi istiqamətində fırlanmasını göstərir. Qaldırılma (yüksəltmə) bucağı kameraya istiqamət ilə xOy müstəvisi arasında qalan bucaqdır.

mesh və **surf** funksiyaları yerinə yetirildikdə, susmaya görə, az=-37.5°, el=30° qəbul edilir. Bu qiymətlərin istənilən zaman anında

```
>>view( [ az , el ] )
```

funksiyası vasitəsilə dəyişmək olar, burada arqumentlərin adları özləri haqda məlumat verir.

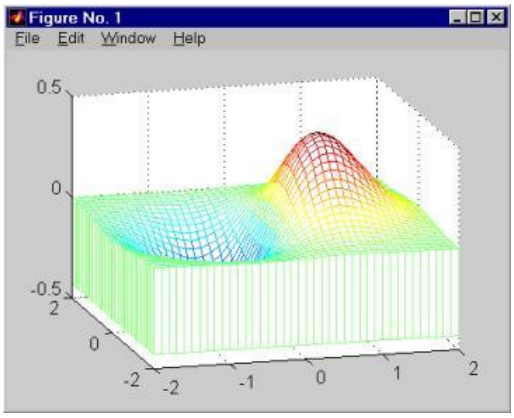
Xüsusi halda, əgər yuxarıda verilmiş

X .* exp(- X.^2 - Y.^2)

funksiyanın qrafikini qurduqdan sonra

>>view([-15 , 20])

əmrini yerinə yetirsək, onda qrafik öz şəklni dəyişər:



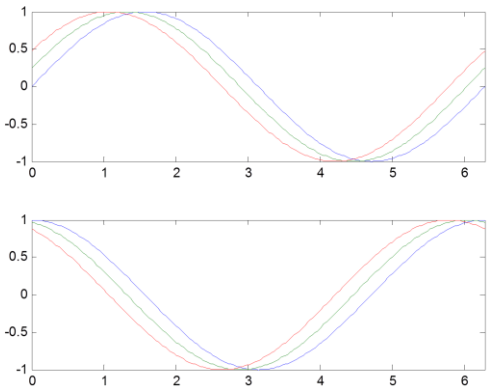
MATLAB sistemiin müxtəlif funksiyalarının çağırışlarını kombinə etsək və rəngləmə və baxış bucağının müxtəlif variantlarını seçsək, üçölçülü qrafiklərin optimal şəklini almaq olar.

3.4.1 Qrafiki istrumental vasitə

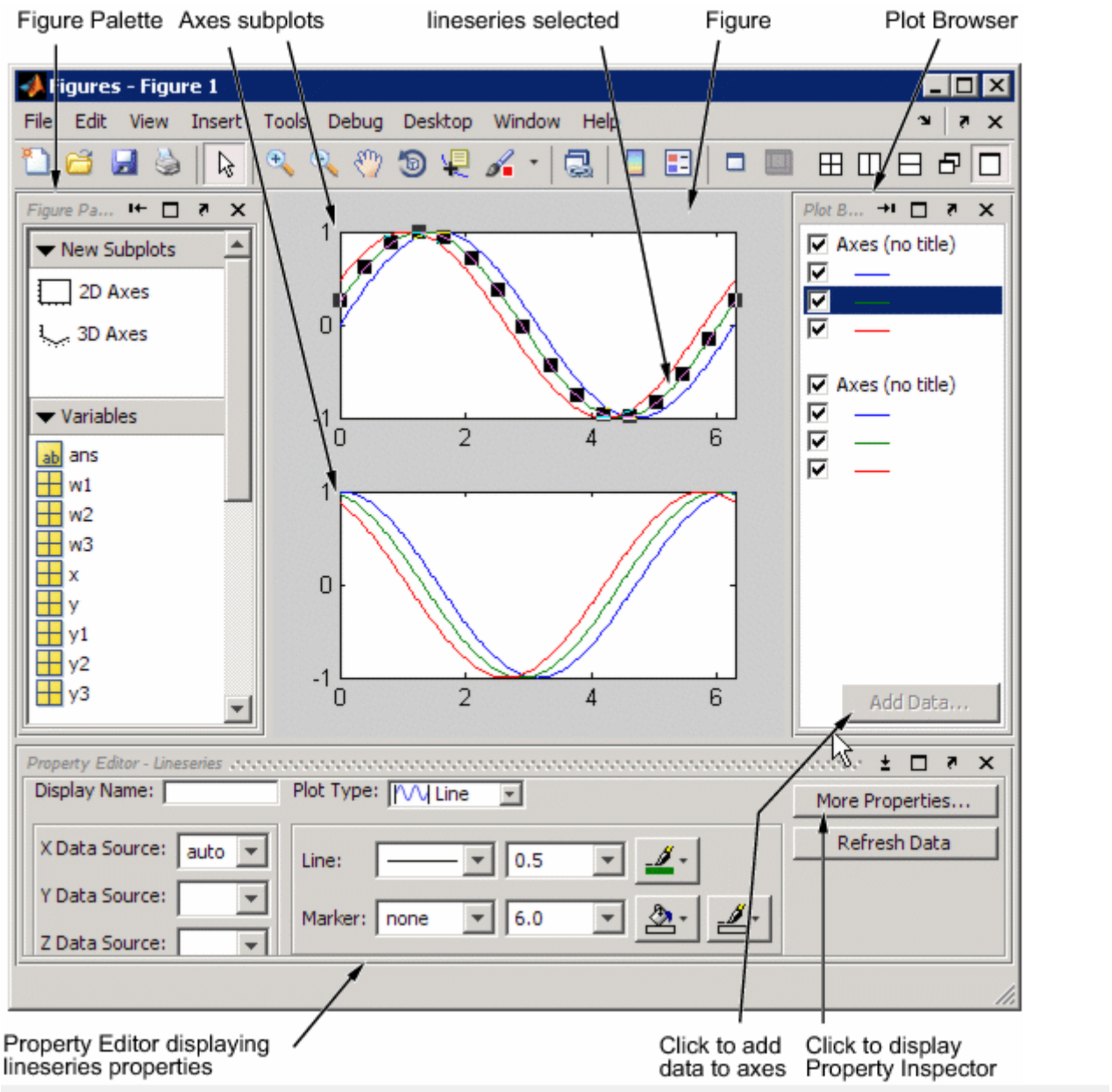
Matlabda müxtəlif məqsədli çoxlu sayda istrumental vasitələrdən vardır ki, onların köməyi ilə proqramlaşdırma vərdişləri olmayan istifadəçilər üçün nəzərdə tutlur və bir çox əməliyyatları birlikəd yerinə yetirmək imkanı vardır. Onlardan bir də **Plottools** instrumental vasitəsidir**.**

Bu istumental vasitəsni daha yaxşı anlamaq üçün aşaıdakl misala mürəciət edək,

```
>> x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x+.25);
y3 = sin(x+.5);
subplot(2,1,1);
plot(x,y1,x,y2,x,y3);
axis tight;
w1 = cos(x);
w2 = cos(x+.25);
w3 = cos(x+.5);
subplot(2,1,2);
plot(x,w1,x,w2,x,w3);
axis tight;
>>
```



Bu və bir çox işləri yerinə yetirə bilcək bir istrumental vasitə ilə bunu yerinə yetirək:



3.5 Fayllarda saxlama və MATLAB-ın qrafiki təsvirlərinin başqa proqramlara ötürülməsi

Tutaq ki, uğurlu təsviri aldıqdan sonra onu gələcəkdə MATLAB sistemində və həmçinin, başqa əlavələrdə bir daha baxış üçün yaddaşda saxlamaq istəyirik.

Qrafiki təsvirlərin yaddaşda saxlanılmsının ən sadə üsulu **Edit | Copy Figure** menyu əmrindən istifadə olunmasıdır. Bu zaman təsvir (bu MATLAB-ın qrafiki pəncərəsinin xidmət gözləyən hissəsidir) **Windows** əməliyyat sisteminin mübadilə buferind **(Clipboard)** saxlanılacaq. Bundan sonra bu təsviri **Paste** menyu əmrinin sonuncusu ilə **Word** redaktorunun sənədində yerləşdirə bilərsiniz. Bütün sənədlərlə birgə MATLAB sistemində alınmış bu təsviri də printerdə çap etmək mümkün olacaq.

Qrafiki pəncərənin menyu komandalarının əvəzinə MATLAB sisteminin əmrlər pəncərəsindən

print -dbitmap və ya **print -dmeta**

əmrindən istifadə etmək olar, ikinci əmr təsviri mübadilə buferində **Bitmap** formatı ilə deyil, **Windows Metafile** formatında saxlayacaq.

Analoji olaraq MATLAB sistemində yaradılan təsviri **Paste** əmrinin köməyilə **Windows** əməliyyat sisteminin **Paint** sadə qrafiki redaktorunda yerləşdirmək olar. Nəticədə MATLAB sistemində alınmış təsviri **Paint** redaktorunun digər təsvirləri ilə quraşdırmaq olar.

MATLAB sistemindəki təsviri **Paste** əmri vasitəsilə məşhur qarışıq rəngli qrafika proqramına yerləşdirsək, daha güclü redaktə və quraşdırma işləri aparmaq mümkündür. Bundan sonra, praktiki olaraq, təsvirin sonrakı redaktələri üçün çox geniş imkanlar yaranacaq və onları müxtəlif formatı fayllarda saxlamaq və müxtəlif printerlərdə (həmçinin profiessional tipoqrafik) çap etmək mümkün olacaq.

Adətən, MATLAB-da alınmış təsviri geniş yayılmış qrafiki formatlı faylda saxlamaq əlverişlidir. Bunu

print -options FileName

əmri ilə asanlıqla yerinə yetirmək olar, burada **option** yerinə konkret vəziyyət üçün identifikatoru yazmaq lazımdır. Məsələn, əgər biz məşhur **Illustrator** vektor qrafikası paketi formatnda qrafiki fayl yaratmaq istəyiriksə, onda **option** yerinə **dill** yazmalıyıq:

print -dill FileName

Bu əmrin yerinə yetirilməsi nəticəsində disketə **FileName.ai** adlı fayl yazılacaq, burada **ai** genişlənməsi **Illustrator** paketi üçün xarakterikdir. Sonradan bu faylı **Illustrator** paketində açmaq və onun bu güclü vektor qrafikası paketi çərçivəsində sonrakı redaktəsini həyata keçirmək olar.

capture əmrini və **imwrite** funksiyasını tətbiq etməklə bir çox başqa məşhur qrafiki formatlı fayllar da yaratmaq olar. Məsələn,

```
>> [X,map]=capture(1);
>>imwrite(X,map,'myfile1.jpg')
```

kodu **myfile1.jpg** faylı yaradır ki, bu faylı **Internet Explorer** brauzerində baxış üçün **Internet**-səhifəyə daxil etmək yaxşıdır.

capture funksiyası təsvirin nöqtələrinə uyğun X matrisini və təsvirdə istifadə olunan rənglərin map matrisini (RGB formatında üç sütun) qaytarır. X matrisinin hər elementi map matrisinin sətirlərindən birinin nömrəsinə bərabərdir. MATLAB sistemində

```
>>colormap( map );
>>image( X );
```

əmrlərini istifadə edərək X və map matrisləri üzrə qrafiki təsviri bərpa etmək olar. Lakin bu, artıq MATLAB sisteminin image obyektlərinə aid mövzudur və bu mövzuya ayrıca bölmə həsr edəcəyik.

3.6 Təsvirlərdə rənglərin qarışdırılması

Kompüterin ekranında ixtiyari təsvir hər biri öz rəngi ilə xarakterizə olunan piksellər massivindən ibarətdir. Pikselin rəngi üç tərkiblə təyin olunur: qırmızı, yaşıl və göy (**Red, Green, Blue - RGB**). Pikselin hər bir rəng tərkibinin kəmiyyətini vermək üçün bir bayt (8 bit) yaddaş lazımdır ki, burada sıfırdan 255-ə qədər (cəmi 256) tam ədədlər yazmaq olar. Beləliklə, ekranın hər pikselinə 0-dan 255 diapazonu arasında dəyişən üç tam ədəd uyğun gəlməlidir. MATLAB sistemində belə tam ədədlərə **uint8** ilə işarə olunan verilənlər tipi uyğundur. **double tipli** adi həqiqi (kəsr) ədədlər üçün ayrılan 8 bayt əvəzinə belə verilənlər tipi üçün bir bayt yaddaş ayrılır. Susmaya görə MATLAB sistemində dəyişənlərə mənimsədilən ədədi qiymətlərdən asılı olmayaraq istənilən dəyişənə **double** tipi uyğun qoyulur. Məsələn, aşağıdakı

iVar1 = 128

sətri kod nəticəsində **iVar1** adlı və **double** tipli dəyişən yaradılır və ona 128 qiyməti mənimsədilir. Belə qiyməti yadda saxlamaq üçün bir bayt kifayət olduğu halda, **double** tipli **iVar1** dəyişəninə 8 bayt yaddaş ayrılır.

Belə sərflərdən uzaqlaşmaq üçün dəyişəni **uint8** modifikatorundan istifadə edərək tam elan etmək lazımdır:

iVar2 = uint8(128)

Beləliklə, yaradılan **iVar2** dəyişəni tam dəyişən (həqiqi deyil) hesab olunur və ona bir bayt yaddaş ayrılır. MATLAB sistemində (yaddaşa qənaət etmək məqsədilə) belə dəyişənlər xüsusi olaraq 0-dan 255-ə qədər tam ədədləri yadda saxlamaq üçün nəzərdə tutulmuşdur. Aşağıdakı

iVar2 = iVar2 + 1

fraqmenti üçün nəticə aşağıdakı kimi alınır:

```
>> iVar1 = 128

iVar1 =

   128

>> iVar2 = uint8( 128

iVar2 =

  128

>> iVar2 = iVar2 + 1

iVar2 =

  129
```

MATLAB sisteminin işçi fəzasından olan bu və ya digər dəyişənin hansı tipə malik olduğunu bilmək istəyirsinizsə (əgər yaddan çıxarmışsınızsa),

>>whos

əmrini daxil edin və yerinə yetirin, nəticədə MATLAB-ın əmrlər pəncərəsində aşağıdakı məlumat görünəcək:

```
>> whos
  Name        Size              Bytes  Class     Attributes

  iVar1       1x1                   8  double
  iVar2       1x1                   1  uint8
```

Buradan məlum olur ki, **iVar1, double** tipli 1x1 ölçülü massivdir (yəni fiziki skalyardır) və yaddaşda 8 baytlıq yer tutur, **iVar2** isə **uint8** tipinə malik olub yaddaşda yalnız 1 bayt (8 dəfə az) yer tutur. Hər iki halda dəyişənlər eyni qiymətə malikdirlər.

Palitra və **colormap** adlandırılan hər hansı rənglər yığınını (m sayda) **double** tipli mx3 ölçülü matris şəklində göstərmək olar. Məsələn, map1 matrisi

map1(1,1)=0.12; map1(1,2)=0.123; map1(1,3)=0.987;
map1(2,1)=0.456; map1(2,2)=0.7; map1(2,3)=0.22;
map1(3,1)=0.88; map1(3,2)=0.19; map1(3,3)=0.611;
map1(4,1)=0.255; map1(4,2)=0.298; map1(4,3)=0.128;
map1(5,1)=0.01; map1(5,2)=0.78; map1(5,3)=0.60

beş rəng yığınını verir. Hər sətir bir rəngə uyğun gəlir. Sətrin elementləri (soldan-sağa) qırmızı, yaşıl və göy rəngləri verir.

Indi **uint8** tipli **kxL** matrisini tərtib edək, bu matrisin hər elementi **map1** rənglər cədvəlinin sətirlərinin nömrələrindən (minus bir) birinə bərabərdir. Bu matrislə rənglər matrisi birlikdə kompüterin ekranında piksellər massivini, yəni ixtiyari təsviri göstərmək üçün kifayətdir. Məsələn,

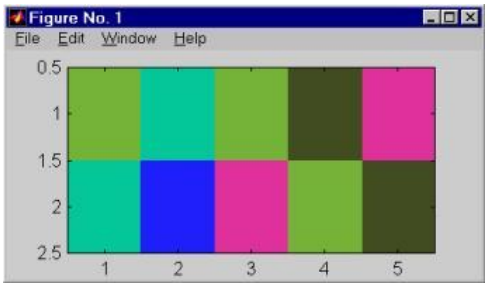
>>X1=uint8([1 4 1 3 2; 4 0 2 1 3])

X1 matrisi **uint8** tipli 2x5 piksel ölçülü massivi verir. Bu massiv **double** tipli massivdən fərqli olaraq yaddaşda 80 bayt deyil, 10 bayt yer tutur. Birinci sıradakı birinci piksel **map1** matrisinin ikinci sətri ilə, bu sıradakı ikinci piksel isə **map1** matrisinin beşinci sətrilə və s. verilən rəngə malikdir.

MATLAB sistemində ixtiyari piksellər şəklini real əks etdirmək istəyiriksə, **image** funksiyasını çağırmalıyıq. Məsələn,

>>image(X1); colormap(map1);

funksiyası MATLAB sisteminin **Image** qrafiki obyektini yaradır və bu obyektə MATLAB-ın qrafiki pəncərəsində real təsvir uyğun gəlir.



MATLAB sisteminin qrafiki pəncərəsinin ölçülərini dəyişdirə bilmirik, buna görə də pəncərə ekranda susmaya görə verilən müəyyən ölçülərlə görünür. Bizim təsvir hər birində beş piksel olmaqla iki sıradan ibarət olduğundan, bu isə çox kiçik təsvirdir (ekran pikselinin fiziki ölçüsü təxminən 0,2 mm-dir), MATLAB bu təsvirin aydın görünməsi üçün susmaya görə onu miqyaslaşdırır (böyüdür). Əgər belə miqyaslaşdırmadan imtina etmək tələb olunursa, onda lazımi ölçüləri aşkar şəkildə vermək lazımdır:

```
>> [ m , n ] = size( X1 );
>>figure( 'Units', 'pixels', 'Position', [100 100 n m] );
>>image( X1 ); colormap( map1 )
```

Burada X1 təsvirinin *m* və *n* ölçüləri MATLAB sisteminin qrafiki pəncərəsindəki şəklin fiziki ölçüləri kimi verilir. Çox kiçik şəkillər üçün bu yaxşı nəticə vermir.

Əgər biz matrislər vasitəsilə yeni təsvirlər yaratmırıqsa, yalnız fayllarda yazılmış hazır şəkilləri MATLAT sisteminin qrafiki pəncərəsində əks etdirmək istəyiriksə, onda bu faylları imread funksiya vasitəsilə oxumaq lazımdır. Xüsusi halda, əvvəl biz üçölçülü təsvirləri imwrite funksiyası vasitəsilə fayllara yazırdıq. Indi isə onları

```
>> [ X2, map2 ] = imread( 'myfile1.jpg' )
```

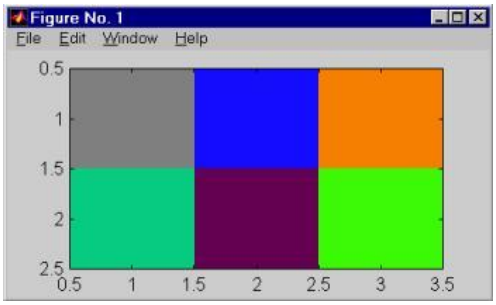
vasitəsilə oxumaq və qrafiki pəncərədə əks etdirmək olar. Qeyd edək ki, fayl MATLAB sisteminin cari kataloqunda yerləşməlidir, əks halda faylın adı ilə ona yolu tam göstərmək lazımdır.

Yuxarıda baxdığımız Image obyekti üçün iki matrisdən ibarət verilənlərin yaradılması (matrislərdən biri sıra ilə rəngləri təyin edir, ikincisi isə öz elementləri ilə rənglər cədvəlinə (matrisinə) girişi təmin edir) Indexed Image (indeksləşdirilmiş təsvir) kimi adlandırılır. Image obyektinin *Truecolor Image* (həddən artıq çox sayda rənglərdən ibarət şəkillər - 16 milyona qədər) adlandırılan başqa bir tipi də var. Image obyektinin ikinci tipi başqa cür təşkil olunub. *Truecolor Image* obyekti üçün rənglər cədvəli tələb olunmur, çünki belə obyektlərin verilənlər massivi rəngləri bilavasitə təyin edir. Bu massivlər mxnx3 ölçüyə malikdirlər (3 ölçülü massivlərdir). m və n kəmiyyətləri şəklin ekrandakı ölçülərini (mxn piksel) təyin edir, üçüncü istiqamət boyunca isə hər bir pikselin RGB tərkib rəngləri yerləşir. Misal üçün TrueColor təsviri üçün aşağıdakı massivi verək:

```
xTrue(1,1,1)=uint8(127);   xTrue(1,1,2)=uint8(127);   xTrue(1,1,3)=uint8(127);
xTrue(1,2,1)=uint8(19);    xTrue(1,2,2)=uint8(12);    xTrue(1,2,3)=uint8(255);
xTrue(1,3,1)=uint8(245);   xTrue(1,3,2)=uint8(127);   xTrue(1,3,3)=uint8(1);
xTrue(2,1,1)=uint8(6);     xTrue(2,1,2)=uint8(203);   xTrue(2,1,3)=uint8(128);
```

xTrue(2,2,1)=uint8(100); xTrue(2,2,2)=uint8(1); xTrue(2,2,3)=uint8(80);
xTrue(2,3,1)=uint8(60); xTrue(2,3,2)=uint8(249); xTrue(2,3,3)=uint8(5);

xTrue massivi tək bir **image(xTrue)** funksiyasının köməyi ilə 2x3 piksel təsvirini yaradır:



Əgər təsvir fayldadırsa və siz qabaqcadan onun hansı tipə (indeks tipinə, yəni rənglər palitrası ilə və ya TrueColor tipinə) malik olduğunu bilmirsinizsə, onda onu aşağıdakı şəkildə oxumaq olar:

>> [X, map] = imread('name.xxx')

Burada **TrueColor** təsviri üçün X matrisi mxnx3 ölçüsünü alır, map palitra matrisi isə boş olur:

size(map) = 0 0

Sonra **image** funksiyası avtomatik olaraq X matrisinin ölçüsünə görə təsvirin tipini tanıyır və hər iki halda lazımi şəkildə işləyir, **colormap** funksiyasına isə map massivinin boş halında heç nə etmir, odur ki, hər iki hal eyni cür işlənə bilər.

Lakin, əgər biz əvvəlcədən faylda **TrueColor** tipli təsvirin olduğunu biliriksə, onda onun oxunması üçün daha qısa olan

>>X = imread('name.xxx')

kodunu yerinə yetiririk, təsviri göstərmək üçün isə yalnız bir **image(X)** funksiyasını çağırırıq. Faylda olan təsvirin tipini öncədən öyrənmək üçün

>>imfinfo('name.xxx')

funksiyasını çağırmaq lazımdır.

Misal.

```
I = imread('pout.tif');
imshow(I)
```

Xüsusi halda vəsaitin əvvəlki bölməsində yaradılan **'pout.tif'** faylı üçün **imfinfo**

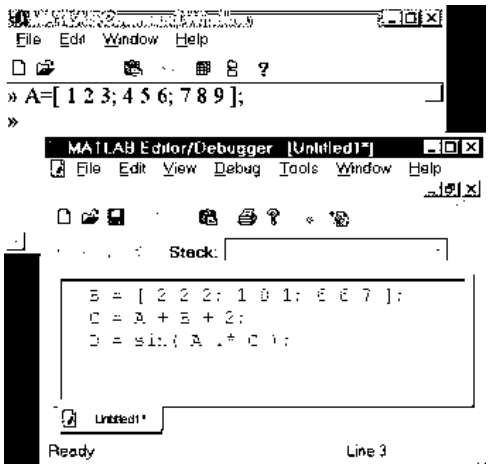
funksiyası aşağıdakı məlumatı verəcək:

```
ans =

                    Filename: [1x62 char]
                 FileModDate: '04-дек-2000 13:57:5(
                    FileSize: 69004
                      Format: 'tif'
                FormatVersion: []
                       Width: 240
                      Height: 291
                    BitDepth: 8
                   ColorType: 'grayscale'
             FormatSignature: [73 73 42 0]
                   ByteOrder: 'little-endian'
              NewSubFileType: 0
               BitsPerSample: 8
                 Compression: 'PackBits'
   PhotometricInterpretation: 'BlackIsZero'
                 StripOffsets: [9x1 double]
            SamplesPerPixel: 1
               RowsPerStrip: 34
            StripByteCounts: [9x1 double]
                XResolution: 72
                YResolution: 72
             ResolutionUnit: 'None'
                   Colormap: []
         PlanarConfiguration: 'Chunky'
                  TileWidth: []
                 TileLength: []
                TileOffsets: []
             TileByteCounts: []
                Orientation: 1
                  FillOrder: 1
```

3.7 Funksiya və ssenari anlayışları

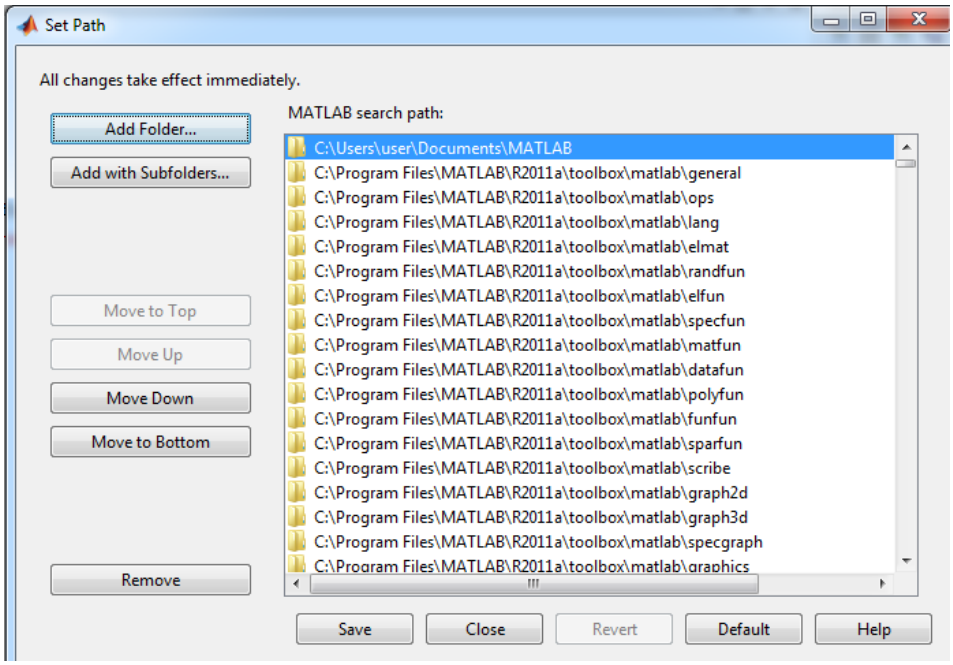
Interaktiv reʁimdə işləyərkən bütün əmrləri klaviatur vasitəsilə daxil etmək lazım gəlir. Sonra, MATLAB sistemi ilə növbəti iş seanslarında **load** əmri vasitəsilə daha əvvəl üzərində hesablamalar aparılan dəyişənlər haqda **MAT**-faylda saxlanılan məlumatı almaq olar. Lakin, bu məlumatın işlənməsi üçün tələb olunan əmrləri yenə də klaviatur vasitəsilə daxil etmək lazım gələcək. Bu o zaman əlverişli olar ki, məlumatın işlənmə ardıcıllığı qabaqcadan məlum olmasın və ya bu cür işlənmələrin həcmi çox böyük olmayıb və az təkrarlanan olsun. Əgər məlumatın işlənmə ardıcıllığı qabaqcadan məlumdursa və çoxsayda təkrarlanarsa, onda əmrlər ardıcıllığını ssenari şəklində tərtib etmək daha məqsədəuyğun olardı.

Ssenari - daxilində ardıcıl şəkildə yerinə yetiriləcək olan lazımi sıralama ilə yazılmış əmrlərdən ibarət mətn faylıdır. Belə faylı istənilən mətn redaktoru vasitəsilə yaratmaq olar, lakin MATLAB sisteminin bu məqsəd üçün özünəməxsus mətn redaktoru vardır və bu redaktor əlavə rahatlıqlarla təmin olunmuşdur. Nəticədə, məhz bu redaktordan istifadə etmək çox yaxşı olardı. O, **File | New | M-file** menyu əmri ilə çağrılır və öz pəncərəsində işləyir:



Ssenari, bilavasitə onun daxilində yaradılan dəyişənlərlə və həmçinin MATLAB sisteminin işçi fəzasında daha əvvəl təyin olunmuş dəyişənlərlə hesablama aparır. Beləliklə, onların dəyişənlər fəzası ümumidir. Ssenari yaratdıqdan sonra onu diskdə faylda saxlamaq lazımdır. Bu fayl ixtiyari ad ala bilər (yaxşı olardı ki, faylın adı MATLAB sisteminin standart *postavkalarına* daxil olan faylların adları ilə üst-üstə düşməsin), adın genişlənməsi isə tək bir m hərfindən ibarət olmalıdır (məsələn, **myScript1.m** və s.) Bu səbəbdən də belə bir deyiliş qəbul olunur: ssenari M-fayllarda saxlanılır. Onları daha öncə öyrənilən və daxilində MATLAB sisteminin işçi fəzasından olan dəyişənlər saxlanılan MAT-fayllarla qarışıq salmaq olmaz

Diskdə saxlanılan M-faylların kataloqu ixtiyari ola bilər, lakin bu kataloqa aparan yol mütləq MATLABa məlum olmalıdır. MATLAB bütün bu cür kataloqlar haqqında məlumatlar saxlayır. Yeni kataloq üçün **File | Set Path…**menyu (əmrlər pəncərəsinin) əmrini yerinə yetirmək lazımdır, bu əmrin köməyi ilə Path Browser (fayllara giriş yollarına baxış) adlı dialoq pəncərəsi çağırılır:



Bu pəncərədə qeydiyyata alınmış giriş yolları siyahısı göstərilir. Yeni kataloqu əlavə etmək üçün bu pəncərənin

Path | Add to path

menyu əmrindən istifadə etmək lazımdır. Yeni kataloqu əlavə etdikdən sonra buraya daxilində redaktorda yaradılan ssenari olan faylı saxlamaq olar. Bu şəkildə saxlanılan ssenarini əmrlər pəncərəsində ssenari yazılmış faylın adını (genişlənməsiz) yazmaqla və **Enter** düyməsini basmaqla istənilən anda tam yerinə yetirmək olar (yəni bir dəfəyə bu ssenaridə olan bütün əmrləri yerinə yetirmək olar). Bir daha xatırladaq ki, ssenari özünün və həmçinin MATLAB sisteminin əmrlər pəncərəsində ssenari çağırılana qədər təyin olunmuş və MATLAB-ın işçi fəzasında saxlanılan dəyişənləri işlədir. Ssenarini çağrılan an işçi fəzada olmayan əlavə məlumat üçün işlənməyə parametr şəklində göndərmək olmaz. Giriş parametrlərinin sonrakı işlənmələrini və həmçinin işçi fəzaların bölünməsi üçün ssenarinin ötürülməsini təmin etmək məqsədilə ssenari yerinə funksiyanı yazmaq lazımdır. Funksiyalar giriş məlumatlarının müəyyən işlənmə alqoritmini realizə edirlər və ümumilikdə böyük bir problemin həllinə imkan verən müxtəlif məsələlərin həlli üçün ideal uyğunlaşdırılıblar.

Biz öz diqqətimizi əsasən ssenarilərin deyil, funksiyaların işlənməsinə və yazılmasına yönəldəcəyik. Buna görə də ssenarilər haqda heç nə deməyəcəyik.

Funksiyalar da ssenarilər kimi əmrlərdən ibarətdirlər və onları da m genişlənməsi ilə mətn faylları şəklində yazır, M-funksiyalar adlandırırlar.

4. SYMBOLIC MATHEMATICS TOOLBOX

MATLAB sistemində **Symbolic Mathematics Toolbox** simvollar riyaziyyatı tətbiqi proqramlar paketi təkcə simvolik hesablamaları deyil, həm də dəyişmə dəqiqliyi ilə hesablamaları həyata keçirir. Paketin tərkibində **Waterloo Maple Software** (Kanada) firması tərəfindən işlənmiş **MAPLE V** simvolik hesablamalar sisteminin nüvəsi daxil edilmişdir. MATLAB sisteminin istifadəçiləri **Symbolic Mathematics Toolbox** paketini tətbiq edərək vahid hesablamalar mühitində ədədi və simvolik hesablamaları hesablamaların dəqiqliyinə və sürətinə heç bir zərər vermədən asan birləşdirirlər. Paket M–fayllar şəklində reallaşdırılan 100-dən çox simvolik və ədədi əməliyyatlara yardım edir.

4.1. Əsas əməliyyatlar

Simvol dəyişəni və ya obyekt yaratmaq: **SYM**
Sintaksis:

```
>>S=sym(A)
>>x=sym('x')
>>x=sym('x', 'real')
>>x=sym('x', 'unreal')
>>S=sym(A,flag)
```

Təsviri: S=sym(A) funksiyası A giriş arqumenti üçün 'sym' sinifli obyekt yaradır. Dəyişən S simvolunu təsvir edən A, əgər simvollar sətridirsə, onda nəticə simvollar ardıcıllığı olacaqdır; əgər A - rəqəmlər sətridirsə, onda nəticə ədəd olacaqdır. Əgər A - ədədlər massividirsə, onda nəticə simvollarla bu obyektə ekvivalent olacaqdır.

>>x=sym('x')

funksiyası x simvol dəyişənini yaradır.

>>x=sym('x', 'real')

funksiyası ancaq həqiqi qiymətlər alan və bütün əməliyyatlarda dəyişən kimi qəbul edilən x simvol dəyişənini yaradır. Bu o vaxta qədər doğru olacaqdır ki, ona hələ, məsələn, kompleks ədədi qiyməti verilməmişdir. Bundan sonra o, həqiqi simvol dəyişəni xassəsini itirir.

>>x=sym('x', 'unreal')

funksiyası əlavə məhdudiyyətsiz formal dəyişən adlanan x simvol dəyişəni yaradır.

Qeyd etmək lazımdır ki, delta=sym('1/10') və pi=sym('pi') şəklində operatorlar, hesabda ədədlərin üzən nöqtə ilə göstərilməsindən qaçaraq 1/10 və pi ədədlərinin simvolla göstərilməsini yaradır.

```
>>delta=sym('1/10')
>>delta=1/10
```

Ehtiyatda saxlanılmaqla yaradılan pi simvol dəyişəni qurulan funksiyanı həmin anda dəyişir.

```
>>pi=sym('pi')
>>pi = pi
>>vpa(pi,12)
>>ans = 3.14159265359
```

S=sym(A,flag) funksiyası A ədədlər massivini S simvol dəyişəninin tipini göstərmək üçün ikinci flag arqumentindən istifadə edərək simvollar massivi formasında dəyişdirir. Tapşırıqların aşağıdakı tipləri mümkündür:

flag-in qiymətləri	Simvol dəyişənin tipləri
‘f’	Üzən nöqtəvi ədəd; ‘±1.F’*2^(e), burada F, e - tam ədəddir.
‘r’	Rasional ədəddir; format p/q və ya p*2^q, burada p və q - tam ədədlərdir.
‘e’	Sükutla keçməyə görə tətbiq edilir. Xətası qiymətləndirilərək göstərilən rasional ədəd; format p*2^q–r*eps/s, burada p, q, r və s - tam ədədlərdir.
‘d’	Onluq ədəddir; format p.q, burada p - tam hissə, q - mantissadır; tam ədədlər və ya mantissada işarələrin sayı vpa və digits funksiyaları ilə müəyyən edilir.

Nümunə: Üzən nöqtəli ədəd şəklində göstəriliş:

```
>>sym(1/10,'f')
ans = '1.999999999999a'*2^(-4)
```

Hesabda 1/10 ədədini üzən nöqtə ilə dəqiq göstərmək olmaz, buna görə mantissanın 13-cü simvolu olan ‘a’(approximately) göstərir ki, o təqribi verilmişdir.

Rasional ədəd şəklində göstəriliş:

```
>>format p/q
>>sym(4/3,’r’)
      ans=4/3
>>format p*2^q
>>sym(1+sqrt(5),’r’)
      ans=7286977268806824*2^(–51)
```

Qayda kimi p/q formatından istifadə edilir, amma, əgər sadə rasional approksimasiyanı qurmaq mümkün deyilsə, p*2^q formatı tətbiq edilir, burada p və q çoxlu miqdarda işarəli tam ədədlərdir. Xətası qiymətləndirilən rasional ədəd şəklində göstəriliş:

```
>>sym(3*pi/4,'e')
ans = 3*pi/4-103*eps/249
```

Onluq ədəd şəklində göstəriliş:

```
>>digits(10),
>>sym(4/3,'d')
ans = 1.333333333
```

```
>>digits(20),
>>sym(4/3,'d')
ans = 1.3333333333333332593
```

Mantissası 16 rəqəmdən az olan onluq ədəd (nümunədə 10 rəqəm) 16 rəqəmdən çox olana (nümunədə 20 rəqəm) nisbətən son dərəcədə zəmanət verilmiş dəqiqliyə malik olmayıb az dəqiqliyə malikdir.

Müşayət edən funksiyalar: DIGITS, DOUBLE, SYMS, EPS.

Simvol obyektlər qrupu yaratmaq: SYMS

Sintaksis:

```
sym arg1 arg2 ...
sym arg1 arg2 ... real
sym arg1 arg2 ... unreal
```

***Təsviri*:** Sym arg1 arg2... funksiyası – bu, aşağıdakı sym funksiyalar ardıcıllığının qısa yazılışıdır: arg1=sym(‘arg1‘); arg2=sym(‘arg2‘); ...

Sym arg1 arg2 ... real funksiyası – bu, aşağıdakı sym funksiyalar ardıcıllığının qısa yazılışıdır: arg1=sym(‘arg1‘,‘real’); arg2=sym(‘arg2‘,’real’); ...

Sym arg1 arg2 ... unreal funksiyası – bu, aşağıdakı sym funksiyalar ardıcıllığının qısa yazılışıdır: arg1=sym(‘arg1‘,‘unreal’); arg2=sym(‘arg2‘,’unreal’); ...

Bu zaman hər bir giriş arqumentinin adı hökmən hərflə başlamalıdır və ancaq hərf və rəqəm daxil edilməlidir.

Müşayət edən funksiya: SYM.

Simvol dəyişənlər siyahısı tərtib etmək: FINDSYM

***Sintaksis*:**

```
>>r=findsym(S)
>>r=findsym(S,n)
```

***Təsviri*:** Findsym(S) funksiyası S ifadəsindəki bütün simvol dəyişənləri əlifba sırasına qaytarır və onları vergüllə bir-birindən ayırır. Əgər ifadədə heç bir simvol dəyişəni yoxdursa, onda baş sətir qayıdır.

Findsym(S,n) funksiyası əlifba sırası ilə nizamlanma mənasında x-ə yaxın n dəyişəni qaytarır.

Nümunə:

```
>>a=1;
>>syms x y w t
>>findsym(x+i*y-i*w)

ans = w, x, y
```

```
>>findsym(x+i*y-i*w,2)

ans = x,y
>>findsym(sin(pi*t),1)
ans = t

>>findsym(a+y,2)
ans = y
```

Müşayət edən funksiyalar: COMPOSE, DIFF, INT, LIMIT,

Simvol dəyişənin həqiqi və xəyali hissəsi- REAL, IMAG

***Sintaksis*:**

```
real(Z)
imag(Z)
```

***Təsviri*:** Kompleks simvol Z ədədləri massivi üçün real(Z) funksiyası həqiqi hissələr massivini, imag(Z) funksiyası isə xəyali hissələr massivini qaytarır.

Müşayət edən funksiya: CONJ.

Misal.

```
>>imag(2+3i)

ans =

    3
```

Simvol massivi elementlərinin kompleks qoşması - CONJ

***Sintaksis*:**

```
conj(X)
```

***Təsviri*: conj(X)** funksiyası X simvol massivini kompleks-qoşma göstərilişinə qaytarır, daha doğrusu X=real(X)+i∗imag(X) funksiyası üçün conj(X)=real(X)−i∗imag(X).

```
>>syms X
>>real(X)
ans = 1/2*X+1/2*conj(X)

>>imag(X)
ans = -1/2*i*(X-conj(X))
```

Müşayət edən funksiyalar: REAL, IMAG.

Simvol ifadələrin ekrana çıxarılması- PRETTY

***Sintaksis*:**

pretty(S)

pretty(S,n)

***Təsviri*: pretty(S)** funksiyası S simvol ifadəsini riyazi ifadənin çapına yaxın formatda ekrana çıxarır və bu zaman sətrin uzunluğunun 79 simvol olmasından istifadə olunur (susmaqla).

pretty(S,n) funksiyası n simvolların sətirləri uzunluğunu nəzərə almaqla S simvol ifadəsini ekrana çıxarır.

Nümunə:

```
>>A=sym(pascal(2));
>>pretty(A)
                    [1    1]
                    [       ]
                    [1    2]
>>B=eig(A);
>>pretty(B)
                  [                1/2]
                  [3/2 + 1/2 5     ]
                  [                   ]
                  [                1/2]
                  [3/2 - 1/2 5     ]
```

Müşayət edən funksiyalar: **SUBEXPR, LATEX, CCODE**.

Latex redaktoru kodlarında simvol ifadənin çevrilməsi - LATEX

Sintaksis:

latex(S)

***Təsviri*:** Latex(S) funksiyası S simvol ifadəsini mətn Latex redaktoru kodlarına qaytarır.

Nümunə:

```
>>syms x
>>f=taylor(log(1+x));
>>latex(f)
ans =
x-1/2\,{x}^{2}+1/3\,{x}^{3}-1/4\,{x}^{4}+1/5\,{x}^{5}

>>H=sym(hilb(3));
```

```
>>latex(H)
ans =\left [\begin {array}{ccc}
1&1/2&1/3\\\noalign{\medskip}1/2&1/3&1/4\\\noalign{\medskip}1/3&1/4&1/5\end
{array}\right ]

>>syms alpha t
>>A=[alpha t alpha*t];
>>latex(A)
ans =
\left [\begin {array}{ccc} \alpha&t&\alpha\,t\end {array}\right ]
```

Müşayət edən funksiyalar: **PRETTY, CCODE, FORTRAN.**

Simvol ifadənin C dilində yazılışı - **CCODE**

Sintaksis:
ccode(S)
Təsviri:
ccode(S) əmri *Symbolic Math Toolbox*-un köməyi ilə simvol dəyişənlərindən istifadə edilərək yazılan ifadə üçün C dilində yazılışı qaytarır.
Nümunə:
Aşağıdakı operatorları C dilinin kodu şəklində yazın:

Fraqment 1:

```
>>syms x
>>f=taylor(log(1+x));
>>ccode(f)
ans =
        t0 = x-x*x/2.0+x*x*x/3.0-x*x*x*x/4.0+x*x*x*x*x/5.0;
```

Fraqment 2:

```
>>H=sym(hilb(3));
>>ccode(H)
ans =
            H[0][0] = 1.0;      H[0][1] = 1.0/2.0;      H[0][2] = 1.0/3.0;
    H[1][0] = 1.0/2.0;      H[1][1] = 1.0/3.0;      H[1][2] = 1.0/4.0;      H[2][0] =
1.0/3.0;      H[2][1] = 1.0/4.0;      H[2][2] = 1.0/5.0;
```

Simvol ifadələrin Fortran dilində yazılışı - FORTRAN

Sintaksis:
fortran(S)
Təsviri: Fortran(S) funksiyası *Symbolic Math Toolbox*-un köməyi ilə simvol dəyişənlərindən istifadə olunaraq yazılmış ifadə üçün Fortran dilində yazılışı qaytarır.
Nümunə:
Aşağıdakı operatorları Fortran dilinin kodu şəklində yazın:

Fraqment 1:

```
>>syms x
>>f=taylor(log(1+x));
>>fortran(f)
ans = t0 = x-x**2/2+x**3/3-x**4/4+x**5/5
```

Fraqment 2:

```
>>H=sym(hilb(3));
>>fortran(H)
ans =
H(1,1) = 1 H(1,2) = 1.E0/2.E0 H(1,3) = 1.E0/3.E0 H(2,1) = 1.E0/2.E0
H(2,2) = 1.E0/3.E0 H(2,3) = 1.E0/4.E0 H(3,1) = 1.E0/3.E0 H(3,2) =
1.E0/4.E0 H(3,3) = 1.E0/5.E0
```

4.2. Obyektlərin çevrilməsi - DOUBLE

Simvol obyekti ədədi obyektə çevirmək
***Sintaksis*:**

```
R=double(S)
```

***Təsviri*:** R=double(S) funksiyası S simvol obyektini ədədi obyektə çevirir. Əgər S - simvol ədədi ifadədirsə, onda bu ifadənin qiyməti hesablamada dəqiqliyi ikiqat artırılmış sürüşən nöqtə ilə qayıdır. Əgər S - simvol ədədi massivdirsə, onda hesablamada dəqiqliyi ikiqat artırılmış sürüşən nöqtəli ədədlər massivinə qayıdır.
Nümunə:

```
>>double(sym('(1+sqrt(5))/2'))
ans = 1.6180
>>a=sym(2*sqrt(2));
>>b=sym((1-sqrt(3))^2);
>>T=[a,b]
T = [ sqrt(8), 4826943532748117*2^(-53)]

>>double(T)
```

```
ans =    2.8284        0.5359
```

Müşayət edən funksiyalar: SYM, VPA.

Simvol obyektini sətir obyektinə çevirmək - CHAR

Sintaksis:
char(S)
***Təsviri*:** char(S) funksiyası S simvol obyektini sətrə çevirir. Əgər S - simvol massividirsə, onda nəticə 'array([[...]])'sətir formasında qayıdır.
Nümunə:

```
>>char(sym('(1+sqrt(5))/2'))
ans = (1+sqrt(5))/2

>>a=sym(2*sqrt(2));
>>b=sym((1-sqrt(3))^2);
>>T=[a,b];
>>char(T)
ans = array([[sqrt(8),4826943532748117*2^(-53)]])
```

Müşayət edən funksiya: DOUBLE.

Vektor əmsalları simvol çoxhədliyə çevirmək- POLY2SYM
***Sintaksis*:**

```
r=poly2sym(C)
r=poly2sym(C,'v')
r=poly2sym(C,sym('v'))
```

***Təsviri*: r=poly2sym(C)** funksiyası əmsalları C vektoru ilə yazılmış çoxhədlini simvollarla göstərilişinə qaytarır. Susmaya görə belə çoxhədlinin dəyişəni x-dir.

r=poly2sym(C,'v') və **r=poly2sym(C,sym('v'))** funksiyaları dəyişən çoxhədlinin verilmiş ikinci arqumentindən V simvol dəyişəni kimi istifadə edir.

Çevirmə funksiyaları çoxhədlinin əmsallarını sym funksiyası istifadə olunan zaman susmaya görə qəbul edilmiş rasional ədədlər şəklində göstərilməsində istifadə edilir. Adətən p/q formatından istifadə olunur, amma, əgər sadə rasional approksimasiyalar qurmaq mümkün olmursa, p*2^q formatı tətbiq edilir, burada p və q çoxlu sayda işarələri olan tam ədədlərdir.

Əgər x simvol dəyişəninə ədədi qiymət verilmişdirsə və çoxhədlinin əmsalları sym funksiyası ilə dəqiq çevrilmişdirsə, onda eval(poly2sym(C)) və polyval(C,x) funksiyaları üst-üstə düşən nəticələr verir.
Nümunə:

```
>>poly2sym([1 1+sqrt(pi) 2])
ans = x^2+6243011064919989/2251799813685248*x+2

>>poly2sym([1    0    1    -1    2],'v')
ans =v^4+v^2-v+2

>>c=[1 3 2];x=1/2;
>>eval(poly2sym(c))
ans =3.7500

>>polyval(c,x)
ans =3.7500
```

Müşayət edən funksiyalar: SYM, SYM2POLY, POLYVAL.

Simvol çoxhədlisini vektor əmsallarına çevirmək- **SYM2POLY**

***Sintaksis*:**

```
C=sym2poly(r)
```

***Təsviri*: C=sym2poly(r)** funksiyası simvol çoxhədlinin əmsallarından ibarət C - vektor-sətrinə qaytarır. Bu zaman əmsallar asılı olmayan dəyişənin tərtibinin azalması ardıcıllığı ilə düzülür:

Nümunə:

```
>>syms x u v;
>>sym2poly(x^3-2*x-5)
ans = 1         0       -2       -5
>>sym2poly(u^4-3+5*u^2)
ans = 1         0         5         0       -3
>>sym2poly(sin(pi/6)*v+exp(1)*v^2)
ans = 2.7183       0.5000                 0
```

Müşayət edən funksiya: POLY2SYM.

4.3. Dəyişmə dəqiqliyi ilə hesab

Ədədi kəmiyyətlərlə əməliyyatların yerinə yetirilməsi zamanı mütləq yuvarlaqlaşdırma səhvi baş verir, çünki hesablamanın dəqiqliyi istifadə olunan hər bir əməliyyatda rəqəmlərin miqdarı ilə məhdudlaşır. Buna görə əməliyyatların çoxsaylı təkrarı zamanı səhvlər üst-üstə yığılır.

Simvol kəmiyyətlər üzərində əməliyyatlar dəqiq həyata keçirilə bilər, çünki ədədlərlə hesablamalar aparılmır və yuvarlaqlaşdırma səhvi yoxdur.

MATLAB sistemi özü ancaq üzən nöqtəli hesabda hesablamaları yerinə yetirir. Tez və istifadə olunan ən böyük yaddaş həcminin üzən nöqtəli əməliyyatlarda boşalması məhduddur ki, bu da yuvarlaqlaşdırma səhvləri və nəticələrdə xətalar əmələ gətirir. Hesabi əməliyyatlarda nisbi dəqiqlik 16-ya yaxın onluq rəqəm təşkil edir. Simvol əməliyyatlar öz növbəsində ixtiyari mənası olan rəqəmli məhdud ədədlə reallaşdırıla bilər. Ədəddəki qərəmlərin sayının artırılması ilə hesablama müddəti və tələb olunan yaddaş artır. Susmaya görə Symbolic Math Toolbox paketində nəticənin dəqiq göstərilişi 32 onluq rəqəm təşkil edir.

Nəticədə mənası olan rəqəmlərin sayını müəyyən etmək - DIGITS

***Sintaksis*:**

d=digits
digits(d)
d=digits

***Təsviri*:** d=digits funksiyası dəyişmə dəqiqliyi hesabında istifadə olunan mənası olan rəqəmi cari görünüşünə qaytarır. Susmaya görə o, 32-yə bərabərdir. digits(d) funksiyası dəyişmə dəqiqliyi hesabında istifadə olunan və d-yə bərabər, mənası olan rəqəmlərin cari miqdarını müəyyən edir.

Nümunə:

```
>>z=1.0e-16;
>>x=1.0e+2;
```

Mənası olan 14 rəqəmli hesablama yerinə yetirə

```
>>digits(14)
>>y=vpa(x*z+1)

    y = 1.0000000000000
```

Mənası olan 15 rəqəmli hesablama yerinə yetirək:

```
>>digits(15)
>>y=vpa(x*z+1)

    y =1.00000000000001
```

Müşayət edən funksiyalar: DOUBLE, VPA.

Hesabda dəyişmə dəqiqliyi ilə hesablamaq- VPA

***Sintaksis*:**

R=vpa(A)
R=vpa(A,d)
vpa Ad

***Təsviri*: R=vpa(A)** funksiyası A massivinin elementlərini mənası olan rəqəmlərin cari miqdarı ilə hesablayır.

R=vpa(A,d) və **vpa Ad** funksiyaları A massivi elementlərini d-yə bərabər miqdarda mənası olan rəqəmlərlə hesablayır.

Nümunə:

```
>>digits(25)
>>vpa(sym(sin(pi/6)))
ans = .50000000000000000000000000

>>p=vpa(pi)
>>p = 3.14159265358979323846264338327950288419716939937510582

>>vpa pi 55

ans = 3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170679
>>w=vpa('(1+sqrt(5))/2')
>>w = 1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911375

>>vpa('exp(pi*sqrt(163))',22)
ans = 262537412640768744.0000

>>A=vpa(hilb(2),25);
>>pretty(A)

    [                                                                 1.
.500000000000000000000000000]
    [
]
    [.500000000000000000000000000       .333333333333333333333333333]
```

Müşayət edən funksiyalar: DIGITS, DOUBLE.

4.4. Riyazi ifadələrin sadələşdirilməsi

Simvol ifadələri sadələşdirmək - SIMPLIFY

***Sintaksis*:**

R=simplify(S)

***Təsviri*:** R=simplify(S) funksiyası S simvol massivinin hər bir elementini sadələşdirir.

Nümunə:

```
>>syms a b c x
>>simplify(sin(x)^2+cos(x)^2)
ans = 1

>>simplify(exp(c*log(sqrt(a+b))))
ans = (a+b)^(1/2*c)
>>S=[(x^2+5*x+6)/(x+2),sqrt(16)];
>>R=simplify(S)
R = [ x+3,      4]
```

Müşayət edən funksiyalar: COLLECT, EXPAND, FACTOR, HORNER, SIMPLE.

Simvol ifadəni açmaq- EXPAND

***Sintaksis*:**

R=expand(S)

***Təsviri*:** R=expand(S) funksiyası S simvol massivinin hər bir elementini açmağa imkan verir. Bu əməliyyat isə tez-tez yalnız çoxhədlilərə, həmçinin triqonometrik, eksponensial və loqarifmik funksiyalara tətbiq edilir.

Nümunə:

```
>>expand((x-2)*(x-4))
ans = x^2-6*x+8

>>expand(log(a*b/sqrt(c)))
ans = log(a*b/c^(1/2))

>>expand(exp((a+b)^2))
ans = exp(a^2)*exp(a*b)^2*exp(b^2)

>>expand([sin(2*t),cos(2*t)])
ans = [ 2*sin(t)*cos(t),        2*cos(t)^2-1]
```

Müşayət edən funksiyalar: COLLECT, FACTOR, HORNER, SIMPLE, SIMPLIFY, SYMS.

Simvol ifadəni sadə vuruqlara ayırmaq - FACTOR

***Sintaksis*:**

factor(N)

factor(S)

***Təsviri*:** factor(N) funksiyası ədədi və ya massiv elementlərini sadə vuruqların hasili şəklində kanonik ayrılışa qaytarır. Burada N - müsbət tam ədəddir.

factor(S) funksiyası massivin hər bir elementinin sadə vuruqlara ayrılışına qaytarır. Burada S - çoxhədlilər matrisidir.

Nümunə:

```
>>factor(sym('123456789012345678901'))
ans = (2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)

>>syms a b
>>factor([a^2-b^2,a^3+b^3])
ans = [              (a-b)*(a+b), (a+b)*(a^2-a*b+b^2)]
```

Müşayət edən funksiyalar: COLLECT, EXPAND, HORNER, SIMPLIFY, SIMPLE.

Bircins hədlərə gətirmə - COLLECT

***Sintaksis*:**

R=collect(S)

R=collect(S,v)

***Təsviri*: R=collect(S)** funksiyası bircins hədlərin x dəyişəninin dərəcəsinə görə toplanmasını həyata keçirir. Burada S - simvol çoxhədlilər massividir.

R=collect(S,v) funksiyası, belə ki, ancaq asılı olmayan dəyişənə, göstərilən ikinci arqumentə (verilən halda v) nəzərən yuxarıda qeyd olunan həmin funksiyanı yerinə yetirir.

Nümunə:

```
>>syms x y;
>>R1=collect((exp(x)+x)*(x+2))
R1 = x^2+(exp(x)+2)*x+2*exp(x)
>>R2=collect((x+y)*(x^2+y^2+1),y)
R2 = y^3+x*y^2+(x^2+1)*y+(x^2+1)*x
>>R3=collect([(x+1)*(y+1),x+y])
R3 =    [ (y+1)*x+y+1,              x+y]
```

Müşayət edən funksiyalar: EXPAND, FACTOR, SIMPLE, SIMPLIFY, SYMS.

Simvol ifadəni sadələşdirmək- SIMPLE

***Sintaksis*:**

simple(S)
r=simple(S)
[r,how]=simple(S)

***Təsviri*: simple(S)** funksiyası S - simvol ifadəsi üzərində müxtəlif cəbri çevirmələri yerinə yetirir, ifadənin qısaldılmış variantlarını ekrana çıxarır və son olaraq ən qısasını qaytarır.

r=simple(S) funksiyası həmin əməliyyatları yerinə yetirir, amma ekrana aralıq nəticələri çıxarmır.

[r,how]=simple(S) funksiyası əsas nəticəyə əlavə olaraq ikinci arqument kimi how sətrini çıxarır ki, bu da yerinə yetirilən çevirməni göstərir.

Nümunələr:

```
>> [r,how]=simple(cos(x)^2+sin(x)^2)
r = 1
how = combine
>> [r,how]=simple(2*cos(x)^2-sin(x)^2)
r = 3*cos(x)^2-1
how = simplify
>> [r,how]=simple(cos(x)+(-sin(x)^2)^(1/2))
r = cos(x)+i*sin(x)
how = radsimp
>> [r,how]=simple(cos(x)+i*sin(x))
r = exp(i*x)
how = convert(exp)
>> [r,how]=simple((x+1)*x*(x-1))
r = -x+x^3
how = collect(x)
>> [r,how]=simple(x^3+3*x^2+3*x+1)
r = (1+x)^3
how = factor
>> [r,how]=simple(cos(3*acos(x)))
r =    4*x^3-3*x
how =    expand
```

Müşayət edən funksiyalar: COLLECT, EXPAND, FACTOR, HORNER, SIMPLIFY.

Simvol çoxhədlilərin rasional şəklə gətirilməsi - NUMDEN

***Sintaksis*:**

[N,D]=numden(A)

***Təsviri*: [N,D]=numden(A)** funksiyası A simvol massivnin hər bir elementini tam qiymətli

əmsallarla sadələşməyən iki çoxhədlinin nisbəti şəklində çevirir. N və D - massiv elementlərinin uyğun olaraq surət və məxrəclərinin simvol massivləridir.

Nümunə:

```
>>syms x y a b
>> [n,d]=numden(x/y+y/x)
n = x^2+y^2
d = y*x
>>A=[a,1/b];
>> [n,d]=numden(A)
n = [ a, 1]
d = [ 1, b]
```

Çoxhədlinin Hörner sxemi şəklində göstərilməsi - HORNER

***Sintaksis*:**

R=horner(P)

***Təsviri*: R=horner(P)** funksiyası P simvol çoxhədli massivinin hər bir elementini Horner sxeminə uyğun çevirir.

Nümunə:

```
>>horner(x^3-6*x^2+11*x-6)
ans = -6+(11+(-6+x)*x)*x
>>horner([x^2+x;y^3-2*y])
ans =
  [       (1+x)*x]
[ (-2+y^2)*y]
```

Müşayət edən funksiyalar: EXPAND, FACTOR, SIMPLE, SIMPLIFY, SYMS.

Əvəzləmədən istifadə olunmaqla simvol ifadəni yazmaq - SUBEXPR

***Sintaksis*:**

[Y,SIGMA]=subexpr(X,SIGMA)

[Y,SIGMA]=subexpr(X,'SIGMA')

***Təsviri*:** [Y,SIGMA]=subexpr(X,SIGMA) və [Y,SIGMA]=subexpr(X, 'SIGMA') funksiyaları X simvol ifadəsini SIGMA əvəzləməsi ilə ilkin ifadəni sadələşdirməyə imkan verən formaya çevirir.

Nümunələr:

```
>>t=solve('a*x^3+b*x^2+c*x+d=0');
>> [r,s]=subexpr(t,'s');
>> [r,s]=subexpr(t,'s')
r =
[                 1/6/a*s^(1/3)-2/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a]
[ -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-
```

```
1/3*b/a+1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]
[ -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a-
1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]

s =

36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-
18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a
```

Müşayət edən funksiyalar: SIMPLIFY, SUBS.

Simvol dəyişənlərin qiymətlərinin əvəz edilməsi - SUBS

***Sintaksis*:**

subs(S)
subs(S,NEW)
subs(S,OLD,NEW)
subs(S,OLD,NEW,O)

***Təsviri*: subs(S)** funksiyası sərbəst simvol dəyişənləri onların ədədi qiymətləri ilə əvəz edir. Bu qiymətlər ya çağrılan funksiyadan, ya da MATLAB sisteminin işçi oblastından götürülür.

subs(S,NEW) funksiyası sərbəst simvol dəyişənləri NEW siyahısındakı ədədi qiymətlərlə əvəz edir.

subs(S,OLD,NEW) funksiyası sərbəst simvol dəyişənlərini OLD yeni simvol dəyişənləri ilə və ya NEW siyahısındakı ədədi qiymətlərlə əvəz edir. Əgər OLD və NEW eyni ölçülü özək massivlərdirsə, onda OLD massivinin hər bir elementi NEW massivinin uyğun elementi ilə əvəz olunur. Əgər S simvol ifadəsi və OLD siyahısı skalyardırsa, amma NEW ədədlər massivi və ya özəklər massividirsə, onda skalyarlar massivinin ölçüsünə qədər genişlənir. Əgər subs(S,OLD,NEW) əvəzləməsi S simvol ifadəsini dəyişmirsə, onda subs(S,NEW,OLD) əvəzləməsi yerinə yetirilir. Tərs əvəzləmə təşəbbüsünün qarşısını almaqdan ötrü subs(S,OLD,NEW,O) çevirməsindən istifadə etmək lazımdır.

Nümunələr:

MATLAB sisteminin işlək oblastından qiymətlərin Dy=–a∗y diferensial tənliyinin həllində əvəz edilməsi:

```
>>a=980;C1=3;
>>syms t
>>y=dsolve('Dy=-a*y');
>>subs(y)
```

```
ans =    3*exp(-980*t)
```

Birkomponentli əvəzləmə:

```
>>syms a b
>>subs(a+b,a,4)
ans = 4+b
```

Çoxkomponentli əvəzləmə:

```
>>subs(cos(a)+sin(b),{a,b},{sym('alpha'),pi/2})
ans = cos(alpha)+1
```

Skalyarın yerinə matrisin qoyulması:

```
>>subs(exp(a*t),'a',-magic(2))
ans =
[      exp(-t), exp(-3*t)]
[ exp(-4*t), exp(-2*t)]
```

Müşayət edən funksiyalar: SUBEXPR.

4.5. Riyazi analiz

Birdəyişənli funksiyanın limiti - LIMIT

***Sintaksis*:**

limit(F,x,a)
limit(F,a)
limit(F)
limit(F,x,a,'right')
limit(F,x,a,'left')

***Təsviri*:**

Limit(F,x,a) funksiyası x→a olduqda F(x) funksiyasının limitini müəyyən edir.

Limit(F,a) funksiyası avtomatik olaraq asılı olmayan dəyişəni, məsələn t-ni findsym(F) funksiyasının köməyi ilə müəyyən edir və sonra t→a olduqda F(t) funksiyasının limiti hesablanır.

Limit(F) funksiyası susmaya görə a=0-ı limit nöqtəsi kimi fərz edir.

Limit(F,x,a,'right') və limit(F,x,a,'left') funksiyaları uyğun olaraq sağ və sol limitləri hesablayır.

Nümunələr:

```
>>syms x a t h;
>>limit(sin(x)/x)
ans = 1

>>limit((x-2)/(x^2-4),2)
ans = 1/4

>>limit((1+2*t/x)^(3*x),x,inf)
ans = exp(6*t)

>>limit(1/x,x,0,'right')
ans = inf
>>limit(1/x,x,0,'left')
ans = -inf

>>limit((sin(x+h)-sin(x))/h,h,0)
ans = cos(x)

>>v=[(1+a/x)^x,exp(-x)];
>>limit(v,x,inf,'left')
ans = [ exp(a),          0]
```

Müşayət edən funksiyalar: PRETTY, CCODE, FORTRAN.

Birdəyişənli funksiyanın diferensiallanması - DIFF

***Sintaksis*:**

diff(S)
diff(S,'v')
diff(S,sym('v'))
diff(S,n)
diff(S,'v',n)
diff(S,n,'v')

***Təsviri*:**

diff(S) funksiyası avtomatik olaraq findsym(S) funksiyasının köməyi ilə asılı olmayan dəyişəni müəyyən edir və sonra uyğun diferensiallanmanı yerinə yetirir.

diff(S,'v') və **diff(S,sym('v'))** funksiyaları S simvol ifadəsini 'v' dəyişəninə görə diferensiallayır.

diff(S,n), diff(S,'v',n), diff(S,n,'v') funksiyaları S simvol ifadəsini 'v' dəyişəninə görə n dəfə diferensiallayır.

Nümunələr:

```
>>syms x t
>>diff(sin(x^2))
ans =    2*cos(x^2)*x
>>diff(t^6,6)
ans = 720
```

Aşağıdakı matrisləri iki dəfə diferensiallayaq:

```
>>syms a b c d
>>F=[a*x b*x^2;c*x^3 d*c]

F =
[      a*x, b*x^2]
[ c*x^3,      d*c]

>>diff(F,2)
ans =
[         0,      2*b]
[ 6*c*x,          0]
```

Müşayət edən funksiyalar: INT, JACOBIAN, FINDSYM.

Birdəyişənli funksiyanın inteqrallanması - INT

***Sintaksis*:**

R=int(S)
R=int(S,v)
R=int(S,a,b)
R=int(S,v,a,b)

***Təsviri*:**

R=int(S) funksiyası findsym(S) funksiyasının köməyi ilə asılı olmayan dəyişəni avtomatik müəyyən edir və sonra qeyri-müəyyən inteqralı hesablayır. Əgər S konstandırsa, onda inteqral x dəyişəninə görə götürülür.

R=int(S,v) funksiyası qeyri-müəyyən inteqralı v dəyişəninə görə hesablayır.

R=int(S,a,b) funksiyası asılı olmayan dəyişənə görə a-dan b-yə qədər müəyyən inteqralı hesablayır. Burada inteqrallama sərhədləri ya ədədi, ya da simvollu kəmiyyətlər ola bilər.

R=int(S,v,a,b) funksiyası v asılı olmayan dəyişəninə görə müəyyən inteqralı a-dan b-yə qədər hesablayır.

Nümunələr:

```
>>syms x x1 alpha u t
>>A=[cos(x*t),sin(x*t);-sin(x*t),cos(x*t)];
>>int(1/(1-x^2))
ans = atanh(x)

>>int(sin(alpha*u),alpha)
ans = -1/u*cos(alpha*u)

>>int(besselj(1,x),x)
ans = -besselj(0,x)

  %
>>int(x1*log(1+x1),0,1)
ans = 1/4

%
>>int(4*x*t,x,2,sin(t))
ans = 2*t*(sin(t)^2-4)

>>int([exp(t),exp(alpha*t)])
ans = [                    exp(t), 1/alpha*exp(alpha*t)]

>>int(A,t)
ans =
[ 1/x*sin(x*t),    -cos(x*t)/x]
[      cos(x*t)/x, 1/x*sin(x*t)]
```

Müşayət edən funksiyalar: DIFF, SYMSUM.

Sıraların cəmlənməsi - SYMSUM

***Sintaksis*:**

r=symsum(S)
r=symsum(S,n)
r=symsum(S,a,b)
r=symsum(S,n,a,b)

***Təsviri*:**

r=symsum(S) funksiyası ümumi həddi (k) olan sıranın indeksə görə cəmlənməsini yerinə yetirir. Belə ki, o, **findsym(S)** funksiyasının köməyi ilə 0-dan k–1-ə qədər avomatik müəyyən edilir.

r=symsum(S,n) funksiyası sıranın k indeksinə görə 0-dan n–1-ə qədər cəmlənməsini

yerinə yetirir.

r=symsum(S,a,b) və **r=symsum(S,n,a,b)** funksiyaları verilmiş a-dan b-yə qədər cəmləməni yerinə yetirir.

Nümunələr:

```
>>syms k n x
>>r=simple(symsum(k))
r = 1/2*k*(k-1)

>>r=simple(symsum(k,0,n-1))
r = 1/2*n*(n-1)
```

Hesabla: $\sum_{k=0}^{10} k^2$	Hesabla: $\sum_{k=11}^{10} k^2$	Hesabla: $\sum_{k=1}^{\infty} 1/k^2$
symsum(k^2,0,10) ans=385	symsum(k^2,11,10) ans=0	symsum(1/k^2,1,inf) ans=0

$\sum_{k=0}^{n} \sin(\pi k)/k$ hesablayın:

```
>>symsum(sin(k*pi)/k,0,n)
ans = -1/2*sin(k*(n+1))/k+1/2*sin(k)/k/(cos(k)-1)*cos(k*(n+1))-1/2*sin(k)/k/(cos(k)-1)

%
>>symsum(x^k/sym('k!'),k,0,inf)
ans =    exp(x)
```

Sonuncu nümunədə k! simvol ifadəsini yaratmaqdan ötrü istifadə edilmişdir. Bu onunla əlaqədardır ki, MATLAB sisteminin sintaksis analizatoru ! simvolunu faktorial işarəsi kimi tanımayacaqdır.

Müşayət edən funksiyalar: FINDSYM, INT, SYMS.

Funksiyaların Teylor və Makloren sıralarına ayrılışı - TAYLOR

***Sintaksis*:**

r=taylor(f,n,v)

r=taylor(f,n,v,a)

r=taylor(f)

***Təsviri*:**

f(x) analitik funksiyasının x=a nöqtəsində Teylor sırasına ayrılışı aşağıdakı şəkildə verilir:

$$f(x)=\sum_{n=0}^{\infty}(x-a)^n\frac{f^{(n)}(a)}{n!}$$.

a=0 olduqda aşağıdakı Teylor sırası Makloren ayrılışı adlanır:

$$f(x)=\sum_{n=0}^{\infty}x^n\frac{f^{(n)}(0)}{n!}$$.

r=taylor(f,n,v) funksiyası f - funksiyasını v dəyişəninin qüvvətinə görə n–1-ci tərtibə qədər n–1-ci tərtib də daxil olmaqla Makloren sırasına ayrılışı yerinə yetirir.

r=taylor(f,n,v,a) funksiyası f funksiyasını a nöqtəsində v dəyişəninin qüvvətinə görə n–1-ci tərtibə qədər (n–1-ci tərtib də daxil olmaqla) Teylor sırasına ayrılışı yerinə yetirir.

r=taylor(f) funksiyası n, v, a arqumentlərindən istifadə edir, belə ki, susmaya görə qəbul edilmişdir: n=6; əgər simvolun indeksi v göstərilməyibsə, onda findsym funksiyası tətbiq edilir; a=0.

Ümumiyyətlə, **r=taylor(f,n,v)** funksiyası istifadə olunan zaman giriş arqumentlərinin ixtiyari kombinasiyası mümkündür.

Nümunələr:

```
>>syms x
>>taylor(exp(-x))
ans = 1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
>>n=6;a=1;
>>taylor(log(x),n,a)
ans = x-1-1/2*(x-1)^2+1/3*(x-1)^3-1/4*(x-1)^4+1/5*(x-1)^5

>>a=pi/2;
>>taylor(sin(x),a,n)
ans =    1-1/2*(x-1/2*pi)^2+1/24*(x-1/2*pi)^4

>>syms t
>>a=3;
>>taylor(x^t,a,t)
ans = 1+log(x)*t+1/2*log(x)^2*t^2
```

Aşağıda verilmiş cədvəl müxtəlif formada verilmiş sıralar üçün **taylor** funksiyasından istifadə olunmasını təsvir edir.

Riyazi yazılışı	*MATLAB sistemində göstərilişi*
$\sum_{n=0}^{5}x^n\frac{f^{(n)}(0)}{n!}$	syms x taylor(f) Susmaya görə n=6, a=0.
$\sum_{n=0}^{N}x^n\frac{f^{(n)}(0)}{n!}$	syms x taylor(f,N)

	N - müsbət tam ədəddir. Susmaya görə a=0.
$\sum_{n=0}^{5}(x-a)^n\frac{f^{(n)}(0)}{n!}$	syms x taylor(f,a) Susmaya görə n=6.
$\sum_{n=0}^{N}(x-a)^n\frac{f^{(n)}(0)}{n!}$	syms x taylor(f,N,a) N - müsbət tam ədəddir.

Əgər f funksiyası iki və ya çoxdəyişənli funksiyadırsa, f=f(x,y,...), onda Teylor və ya Makloren sırasına hər hansı bir dəyişənə nəzərən ayrılışı bu dəyişənin aşkar göstərilməsi ilə verilir. Aşağıdakı cədvəldə bu ikidəyişənli funksiyalar üçün göstərilmişdir:

Riyazi yazılışı	*MATLAB sistemində göstərilişi*
$\sum_{n=0}^{5}\frac{y^n}{n!}\frac{\partial^{(n)}}{\partial y^n}f(x,y=0)$	syms x y taylor(f,y) Susmaya görə n=6, a=0.
$\sum_{n=0}^{N}\frac{y^n}{n!}\frac{\partial^{(n)}}{\partial y^n}f(x,y=0)$	syms x y taylor(f,y,N) və ya taylor(f,N,y) N - müsbət tam ədəddir. Susmaya görə a=0.
$\sum_{n=0}^{5}\frac{(y-a)^n}{n!}\frac{\partial^{(n)}}{\partial y^n}f(x,y=a)$	syms x y taylor(f,y,a) və ya taylor(f,a,y) Susmaya görə n=6.
$\sum_{n=0}^{N}\frac{(y-a)^n}{n!}\frac{\partial^{(n)}}{\partial y^n}f(x,y=a)$	syms x taylor(f,N,y,a) N - müsbət tam ədəddir.

Müşayət edən funksiyalar: FINDSYM, SYMSUM.

Funksiyalar yakobianının hesablanması - JACOBIAN

***Sintaksis*:**

R=jacobian(F,v)

***Təsviri*:**

Çoxölçülü F(x1,x2,...,xn) vektor funksiyası üçün Yakobi matrisi aşağıdakı şəkildə müəyyən edilir:

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}.$$

R=jacobian(F,v) funksiyası vektor və ya skalyar F funksiyası üçün v dəyişənli vektora görə Yakobi matrisini hesablayır. Yakobi matrisinin J(i,j) elementi $\partial F_i / \partial v_i$ -yə bərabərdir.

Nümunələr:

```
>>syms x y z
>>F=[x*y*z;y;x+z];
>>v=[x,y,z];
>>j=jacobian(F,v)
j =
[ y*z, x*z, x*y]
[     0,     1,     0]
[     1,     0,     1]

>>v=[x,z];
>>jxz=jacobian(F,v)
jxz =
[ y*z, x*y]
[     0,     0]
[     1,     1]

>>b=jacobian(x+y,v)
b = [ 1, 0]
```

Müşayət edən funksiya: DIFF.

4.6. Funksiyaların superpozisiyası, çevrilməsi və tənliklərin həlli

Funksiyaların superpozisiyası - COMPOSE

***Sintaksis*:**

compose(f,g)
compose(f,g,z)
compose(f,g,x,z)
compose(f,g,x,y,z)

***Təsviri*: compose(f,g)** funksiyası y dəyişənli funksiya kimi f(g(y)) funksiyasını qaytarır, burada f=f(x) və g=g(y) x və y asılı olmayan dəyişənləri findsym funksiyasının köməyi ilə müəyyən edilir.

compose(f,g,z) funksiyası z dəyişənli funksiya kimi f=f(x) və g=g(y) olmaqla f(g(y)) funksiyasına qaytarır.

compose(f,g,x,z) funksiyası f(g(z)) funksiyasına qaytarır və bu zaman x-ə f funksiyasının asılı olmayan dəyişəni kimi baxılır. Bu, funksiya iki və ya çox dəyişəndən asılı olduqda əhəmiyyətlidir. Məsələn, tutaq ki, f=cos(x/t), onda **compose(f,g,x,z)** funksiyası cos(g(z)/t)-yə, amma compose(f,g,t,z) funksiyası cos(x/g(z))-ə qaytarır.

compose(f,g,x,y,z) funksiyası f(g(z)) funksiyasına qaytarır və bu zaman x-ə f funksiyasının asılı olmayan dəyişəni kimi, y-ə isə g funksiyasının asılı olamayan dəyişəni kimi baxılır. Məsələn, tutaq ki, f=cos(x/t) və g=sin(y/u), onda **compose(f,g,x,y,z)** funksiyası cos(sin(z/u)/t)-yə, **composef(f,g,x,u,z)** funksiyası isə cos(sin(y/z)/t)-yə qaytarır.

Nümunələr:

```
>>syms x y z t u;
>>f=1/(1+x^2); g=sin(y); h=x^t; p=exp(-y/u);
>>compose(f,g)
ans = 1/(1+sin(y)^2)

>>compose(f,g,t)
ans = 1/(1+sin(t)^2)

>>compose(h,g,x,z)
ans =    sin(z)^t

>>compose(h,g,t,z)
ans = x^sin(z)

>>compose(h,p,x,y,z)
ans =    exp(-z/u)^t

>>compose(h,p,t,u,z)
ans =    x^exp(-y/z)
```

Müşayət edən funksiyalar: FINVERSE, FINDSYM, SUBS, SYMS.

Tərs funksiyaların hesablanması - FINVERSE

***Sintaksis*:**

g=finverse(f)

g=finverse(f,u)

***Təsviri*:**

g=finverse(f) funksiyası f-in tərs funksiyasını hesablayır. Fərz edilir ki, f funksiyası birdəyişənlidir, məsələn x-dən asılıdır, onda onun g tərs funksiyası g(g(x))=x şərtini ödəyir.

g=finverse(f,u) funksiyası hansı dəyişənə nəzərən tərs funksiyanın qurulmasının zəruriliyini göstərməyə imkan verir. Bu o vaxt zəruridir ki, f funksiyası bir neçə dəyişəndən asılıdır.

Nümunələr:

```
>>syms x
>>f=1/tan(x);
>>g=finverse(1/tan(x))
g = atan(1/x)

>>compose(g,t)
ans = atan(1/t)

>>compose(g,f)
ans = atan(tan(x))

>>syms y
>>f=x^2+y;
>>g=finverse(f,y)
g = -x^2+y

>>finverse(f)
Warning: finverse(x^2+y) is not unique.
> In C:\MATLABR12\toolbox\symbolic\@sym\finverse.m at line 43
ans = (-y+x)^(1/2)
>>syms u v
>>finverse(exp(u-2*v),u)
ans =    2*v+log(u)
```

Müşayət edən funksiyalar: COMPOSE, SYMS.

Tənliklərin və sistem tənliklərin simvollarla həlli - SOLVE

***Sintaksis*:**

g=solve(eq)
g=solve(eq,var)
g=solve(eq1,eq2,...,eqn)
g=solve(eq1,eq2,...,var1,var2,...,varn)

***Təsviri*:** Bəzi tənliklərin həlli.

g=solve(eq) funksiyası ya simvollar şəklində, ya da bərabərlik işarəsi göstərilən və ya

göstərilməyən sətir ifadələrlə verilmiş tənlikləri həll edir. Əgər bərabərlik işarəsi göstərilməmişdirsə, onda fərz edilir ki, tənlik eq=0 şəklindədir. Həllin hansı dəyişənə nəzərən axtarıldığı aşkar şəkildə göstərilməyibsə, həmin dəyişən findsym funksiyasının köməyi ilə avtomatik müəyyən edilir.

g=solve(eq,var)

funksiyası var dəyişəninə nəzərən tənliyi həll edir.

Tənliklər sisteminin həlli

g=solve(eq1,eq2,...,eqn) və

g=solve(eq1,eq2,...,var1,var2,...,varn)

funksiyası n dənə dəyişənə nəzərən tənliklər sistemini həll edir. Əgər dəyişənlər aşkar şəkildə göstərilməyibsə, onlar findsym funksiyasının köməyi ilə avtomatik müəyyən edilir. Əgər analitik həll tapılmazsa və tənliklərin sayı dəyişənlərin sayına bərabərdirsə, onda mümkün ədədi həllərdən ancaq biri axtarılır.

Hesablamaların nəticələri üç müxtəlif üsulla göstərilə bilər:

- bir tənlik və bir çıxış arqumenti üçün həll birölçülü və ya çoxölçülü özəklər massivi şəklində qaytarılır;
- çıxış arqumentlərinin saynın dəyişənlərin sayına bərabər tənliklər sistemi üçün həll əlifba sırası ilə dəyişənlərin adları ardıcıl qaytarılır;
- bir çıxış arqumentli tənliklər üçün həll qeydlər massivi şəklində qaytarılır.

Nümunələr:

Bəzi tənliklərin həlli

```
>>syms x p r
>>solve('p*sin(x)=r')
ans = asin(r/p)

>>syms a b c x
>>x=solve(a*x^2+b*x+c)
x =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]

>>syms x
>>solve('a*x^2+b*x=-c')
ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

```
>>syms x
>>b=solve(a*x^2+b*x+c,b)
b = -(a*x^2+c)/x

>>syms t
>>t=solve(tan(t)-sin(2*t))
t =
[             0]
[            pi]
[    1/4*pi]
[ -3/4*pi]
>>syms x
>>x=solve(cos(x)-sin(x^2))
x =    .84936886239267305068327920512917
```

Tənliklər sisteminin həlli

```
>>syms x y
>> [x,y]=solve('x^2+x*y+y=3','x^2-4*x+3=0')
x =                                          y =
[ 1]                                           [       1]
[ 3]                                           [ -3/2]

>>syms x y
>>S=solve('x^2+x*y+y=3','x^2-4*x+3=0')
S =

       x: [2x1 sym]
       y: [2x1 sym]

S.x                                          S.y

ans =                                        ans =
         [ 1]                                                [       1]
         [ 3]                                                [ -3/2]

>>syms a u v
>> [u,v]=solve('a*u^2+v^2=0','u-v=1')

u =
```

```
  [ 1/2/(a+1)*(-2*a+2*(-a)^(1/2))+1]
[ 1/2/(a+1)*(-2*a-2*(-a)^(1/2))+1]

v =

[ 1/2/(a+1)*(-2*a+2*(-a)^(1/2))]
[ 1/2/(a+1)*(-2*a-2*(-a)^(1/2))]

>>syms a u v
>>S=solve('a*u^2+v^2','u-v=1',a,u)
S =

    a: [1x1 sym]
    u: [1x1 sym]

S.a                                                    S.u

            ans = -v^2/(v^2+2*v+1)                     ans = v+1

>>syms a u v
>> [a,u,v]=solve('a*u^2+v^2','u-v=1','a^2-5*a+6')

a =                                          u =

   [ 2]                                        [ 1/3+1/3*i*2^(1/2)]
   [ 2]                                        [ 1/3-1/3*i*2^(1/2)]
   [ 3]                                        [ 1/4+1/4*i*3^(1/2)]
   [ 3]                                        [ 1/4-1/4*i*3^(1/2)]

v =

  [ -2/3+1/3*i*2^(1/2)]
[ -2/3-1/3*i*2^(1/2)]
[ -3/4+1/4*i*3^(1/2)]
[ -3/4-1/4*i*3^(1/2)]

>>syms x y
>>S=solve('sin(x+y)-exp(x)*y=0','x^2-y=2')

S =
```

x: [1x1 sym]
y: [1x1 sym]

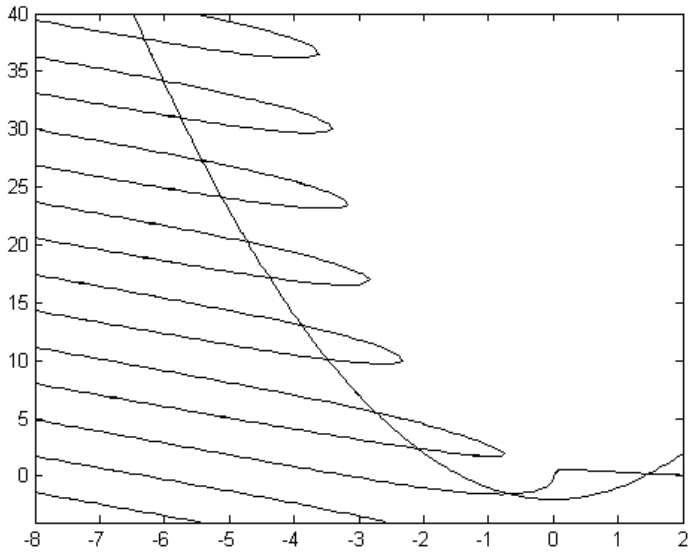
S.x

ans = -6.017327250059306564109729711790 5

S.y

ans = 34.208227234306296508646214438330

Burada həllər çoxluğundan ancaq biri tapılmışdır. Mümkün həlləri görmək üçün (x,y) müstəvisində bu funksiyaların kəsişmə nöqtələrini quraq (şəkil 4.1).



Şəkil 4.1

```
>> [X,Y]=meshgrid([-8:0.1:2],[-4:0.5:40]);
>>Z1=sin(X+Y)-exp(X).*Y;
>>Z2=X.^2-Y-2;
>>contour(X,Y,Z1,[0.001 0 0.001]),hold on,contour(X,Y,Z2,[0.001 0 0.001]);
>>plot(-6.017,34.208,'or')
```

Burada isə avtomatik həll edənin tapdığı həllər qeyd edilmişdir.
Üç məchullu üç cəbri tənliklər sistemini həll edək:

```
>> syms a u v
A=solve('a*u^2+v^2','u-v=1','a^2-5*a+6')

A =

    a: [4x1 sym]
    u: [4x1 sym]
    v: [4x1 sym]
```

```
>> A.u

ans =

 1/3 + (2^(1/2)*i)/3
 1/4 + (3^(1/2)*i)/4
 1/4 - (3^(1/2)*i)/4
 1/3 - (2^(1/2)*i)/3

>> A.v

ans =

 - 2/3 + (2^(1/2)*i)/3
 - 3/4 + (3^(1/2)*i)/4
 - 3/4 - (3^(1/2)*i)/4
 - 2/3 - (2^(1/2)*i)/3
```

Müşayət edən əmrlər və funksiyalar: DSOLVE, FINDSYM.

Tənliklərin və ADT sisteminin simvollarla həlli- DSOLVE

***Sintaksis*:**

r=dsolve(‘eq1’, ‘eq2’,...,‘cond11’,‘cond2’,...,’v’)

r=dsolve(‘eq1, eq2,...’,cond11,cond2,...’,’v’)

***Təsviri*:**

r=dsolve(‘eq1’, ‘eq2’,...,‘cond11’,‘cond2’,...,’v’)

funksiyası başlanğıc və ya sərhəd şərtləri cond1, cond2, ... dəyişənləri ilə müəyyən olunan eq1, eq2, ... tənlikləri ilə verilmiş adi diferensial tənliklərin (ADT) v asılı olmayan dəyişəninə nəzərən analitik həllini hesablayır. Susmaya görə asılı olmayan dəyişən kimi t - zaman qəbul edilir.

D simvolu asılı olmayan dəyişənə nəzərən diferensiallamanı göstərir, daha doğrusu d/dt.

D simvolundan sonra gələn rəqəm təkrar diferensiallamanı göstərir, yəni D2 d2/dt2-yə uyğundur. Diferensiallama operatorundan sonra gələn simvol asılı dəyişəndir, məsələn D3y göstərir ki, d3y/dt3. Qeyd edək ki, asılı dəyişənin adı D hərfi ilə başlamamalıdır.

Başlanğıc və sərhəd şərtləri y(a)=b, Dy(a)=b şəklində tənliklərlə verilir, burada y - asılı dəyişən, a və b - sabitlərdir. Əgər şərtlərin sayı dəyişənlərin sayından azdırsa, onda həllə c1, c2, ... ixtiyari sabitləri daxil ediləcəkdir.

Hər bir tənliyi və ya şərti ayrı arqument şəklində yazmaq olar; dsolve funksiyası 12-yə qədər giriş arqumentlərinə imkan verir.

Əgər çıxış arqumentləri göstərilməyibsə, dsolve həllər siyahısını qaytarır.

Əgər analitik həll tapılmayıbsa, dsolve xəbərdarlıq qaytarır. Bu halda, ədədi həllin axtarılması üçün ode23 və ya ode45 funksiyalarından istifadə etmək lazımdır.

r=dsolve(‘eq1, eq2,...’,cond11,cond2,...’,’v’)

şəklində funksiya bir neçə tənliyi və ya şərti bir arqumentdə birləşdirməyə imkan verir.

Hesablamaların nəticələri üç müxtəlif üsulla göstərilə bilər:

- bir tənlik və bir çıxış arqumenti üçün həll birölçülü və ya çoxölçülü özəklər massivi şəklində qaytarılır;
- çıxış arqumentlərinin sayının dəyişənlərin sayına bərabər tənliklər sistemi üçün həll əlifba sırası ilə dəyişənlərin adları ardıcıl qaytarılır;
- bir çıxış arqumentli tənliklər üçün həll qeydlər massivi şəklində qaytarılır.

Nümunələr:

```
>>dsolve('Dx=a*x')
ans = C1*exp(a*t)
>>x=dsolve('Dx=-a*x','x(0)=1','s')
x = exp(-a*s)

>>y=dsolve('(Dy)^2+y^2=1','y(0)=0')
y =
[    sin(t)]
[ -sin(t)]
>>S=dsolve('Df=f+g','Dg=-f+g','f(0)=1','g(0)=2')
S =
        f: [1x1 sym]
        g: [1x1 sym]

S.f
ans = exp(t)*(cos(t)+2*sin(t))
```

```
S.g
ans = -exp(t)*(sin(t)-2*cos(t))

>>dsolve('Df=f+sin(t)','f(pi/2)=0')
ans = -1/2*cos(t)-1/2*sin(t)+…
+1/2*exp(t)/(cosh(1/2*pi)+sinh(1/2*pi))
```

Sərhəd şərtləri olan ADT-in həlli:

```
>>dsolve('D2y=-a^2*y','y(0)=1,Dy(pi/a)=0')
ans = cos(a*t)
```

ADT sisteminin həlli:

```
>>S=dsolve('Dx=y','Dy=-x','x(0)=0','y(0)=1')
S =
    x: [1x1 sym]
    y: [1x1 sym]

S.x                                          S.y
ans = sin(t)                                 ans = cos(t)

>>S=dsolve('Du=v,Dv=w,Dw=u','u(0)=0,v(0)=0,w(0)=1')
S =
    u: [1x1 sym]
    v: [1x1 sym]
    w: [1x1 sym]

>>w=dsolve('D3w=-w','w(0)=1,Dw(0)=0,D2w(0)=0')
w = 1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*t*3^(1/2))
>>f=dsolve('Df=f+sin(t)')
f = -1/2*cos(t)-1/2*sin(t)+exp(t)*C1

>>y=dsolve('Dy=a*y','y(0)=b')
y = b*exp(a*t)

>>[x,y]=dsolve('Dx=y','Dy=-x')
x = cos(t)*C1+sin(t)*C2
y = -sin(t)*C1+cos(t)*C2
```

Diaqnostik məlumat:

Əgər dsolve funksiyası analitik həlli tapmırsa, məlumat daxil edilir:

warning: explicit solution could not be found and
return an empty sym object.

xəbərdarlıq: simvol həll tapılmamışdır, çıxış - boş simvol dəyişənləri - [empty sym].

Müşayət edən funksiyalar: SOLVE, SUBS.

4.7. Simvollu matrislər və massivlərlə iş

Simvollu matrislərdə və massivlərdə əməliyyatlar + – */ \ ^ ′

***Sintaksis*:**

Matrislər üzərində əməllər	Massivlər üzərində əməllər
A+B	A.*B
A–B	A./B
A*B	A.\B
A/B	A.^B
A\B	A.′
A^B	
A′	

Təsviri:

Symbolic Toolbox TPP-də (tətbiqi proqramlar paketi) iki tip simvol əməliyyatlar həyata keçirilir: massivlər və matrislər üzərində.

Massivlər üzərində əməliyyatlar hər bir element olmaqla yerinə yetirilir, amma matrislər üzərində əməliyyatlar xətti cəbrin qaydalarına uyğun olaraq müəyyən edilmişdir. Bu əməliyyatları fərqləndirmək üçün massivlər üzərindəki əməliyyataın sonunda nöqtə qoyulur. Massivlər və matrislər üzərində toplama və çıxma əməliyyatları eyni nəticələri verirlər.

+	Toplama A+B	Toplananlardan biri skalyar olduğu haldan başqa bütün hallarda toplananların ölçüləri eyni olmalıdır; skalyar başqa operandın bütün elementlərinə əlavə olunur.
–	Çıxma A–B	Azalan və çıxılan eyni ölçüdə olmalıdır, onlardan biri skalyar olduğu haldan başqa hallarda; skalyar başqa operandın bütün elementlərindən çıxılır.
*	Matrislərin vurulması A*B	Matrislərin və ya matrisin vektorlara vurulması zamanı birinci vuruğun sütunlarının sayı ikincinin sətirlərinin sayına bərabər olmalıdır. Vuruğun bütün elementləri skalyara vurulur.
.*	Massivlərin vurulması	Eyni ölçülü iki massivin elementlərinin bir-birinə

	A.*B	vurulması. Massivin bütün elementləri skalyara vurulur.
\	Xətti tənliklər sisteminin həlli AX=B	Əgər A - n×n ölçülü kvadrat matris və B - n×k ölçülü matrisdirsə. Düzbucaqlı A matrisi mümkündür, amma tənlik ümumi olmalıdır. Həll ən kiçik kvadratlar üsulu ilə hesablanmalıdır.
.\	Massivlərin soldan bölünməsi A.\B	Nəticə B(i,j)/A(i,j) elementlər massividir; massivlərdən biri skalyar olduğu haldan başqa bütün hallarda massivlərin ölçüləri eyni olmalıdır.
/	Xətti tənliklər sisteminin həlli XA=B	A/B əməliyyatı (A•\B•)• ilə eynigüclüdür. Düzbucaqlı A matrisi mümkündür, amma tənlik ümumi olmalıdır. Həll ən kiçik kvadrat üsulu ilə hesablanmışdır.
./	Massivlərin sağdan bölünməsi A./B	Nəticə A(i,j)/B(i,j) elementlər massividir; massivlərdən biri skalyar olduğu haldan başqa bütün hallarda massivlərin ölçüləri eyni olmalıdır.
^	Matrislərin qüvvəti A^P	Əgər P - tam müsbət ədəddirsə, onda matrisin qüvvəti matrisin özü-özünə vurulması yolu ilə hesablanır; əgər P - tam mənfi ədəddirsə, onda həmin qayda tərs matrislərin vurulması kimidir. P-nin başqa qiymətləri üçün məxsusi qiymətlər və vektorlar hesablanır. Belə ki, əgər [R,D]=eig(A), onda A^P=R*D.^P/R.
.^	Massivlərin qüvvəti A.^B	Nəticə A(i,j)^B(i,j) elementlər massividir; massivlərdən biri skalyar olduğu haldan başqa bütün hallarda massivlərin ölçüləri eyni olmalıdır.
′	Matrisin transponirəsi A′	Həqiqi matrislər üçün nəticə trapsponirə olunmuş matrisdir; kompleks matrislər üçün transponirə kompleks qoşmalarla tamamlanır.
.′	Massivin transponirəsi A.′	Həqiqi və kompleks massivlər üçün sətirlər sadəcə olaraq sütunlarla əvəz olunur; kompleks qoşma əməlyyatı yerinə yetirilmir.

Nümunə:

Üzərində yuxarıda yazılan əməliyyatlar aparılan nümunə kimi birölçülü iki massivə (vektora) baxaq; çap üçün format çat formatından istifadə olunur.

Matrislər üzərində əməllər	*Massivlər üzərində əməllər*
x = [a] [b] [c]	y = [d] [f] [g]
x′ [conj(a), conj(b), conj(c)]	y′ [conj(d), conj(f), conj(g)]

x+y	[a+d] [b+f] [c+g]	x+y	[a–d] [b–f] [c–g]
x+2	[a+2] [b+2] [c+2]	y+2	[d+2] [f+2] [g+2]
x*y	Error	x.*y	[a*d] [b*f] [c*g]
x'*y conj(a)*d+conj(b)*f+conj(c)*g		x'.*y	Error
x*y' [a*conj(d), a*conj(f), a*conj(g)] [b*conj(d), b*conj(f), b*conj(g)] [c*conj(d), c*conj(f), c*conj(g)]		x.*y'	Error
x*2	[2*a] [2*b] [2*c]	y.*2	[2*d] [2*f] [2*g]
x\y Warning: System is inconsistent. Solution does not exist.		x.\y	[d/a] [f/b] [g/c]
2\x	[1/2*a] [1/2*b] [1/2*c]	2.\y	[1/2*d] [1/2*f] [1/2*g]
x/y	[a/d, 0, 0] [b/d, 0, 0] [c/d, 0, 0]	x./y	[a/d] [b/f] [c/g]
x\2	[1/2*a] [1/2*b] [1/2*c]	y.\2	[1/2*d] [1/2*f] [1/2*g]
x^y	Error	x.^y	[a^d] [b^f] [c^g]
x^2	Error	x.^2	[a^2] [b^2] [c^2]
2^x	Error	2.^x	[2^a] [2^b]

	[2^c]
(x+i*y)'	[conj(a+i*d),conj(b+i*j),conj(c+i*g)]
(x+i*y).'	[a+i*d, b+i*f, c+i*g]

```
>>syms a b c d;
>>A=[a b; c d]

A =
[ a, b]
[ c, d]

A*A/A                                   A*A\A

ans =                                   ans =
[ a, b]                                 [    d/(-b*c+a*d), -b/(-b*c+a*d)]
[ c, d]                                 [ -c/(-b*c+a*d),    a/(-b*c+a*d)]


>>syms a11 a12 a21 a22 b1 b2;
>>A=[a11 a12; a21 a22];
>>B=[b1 b2];
>>X=B/A;

>>x1=X(1)
x1 = -(-a21*b2+b1*a22)/(-a11*a22+a12*a21)

>>x2=X(2)
x2 = (-a11*b2+a12*b1)/(-a11*a22+a12*a21)
```

Müşayət edən funksiyalar: NULL, SOLVE.

4.8. Xətti cəbr

Matrisin determinantı - DET

***Sintaksis*:**

r=det(A)

***Təsviri*: r=det(A)** funksiyası ədədi və ya simvol matrislərin determinantını hesablayır. Əgər A matrisi simvollar şəklində verilmişdirsə, onda nəticə simvol ifadədir, əgər A matrisi ədədlərlə verilmişdirsə, onda nəticə ədəddir.

Nümunə:

```
>>syms a b c d;
```

```
>>det([a,b;c,d])
ans = -b*c+a*d

>>A=sym([2/3 1/3;1 1])
A =
[ 2/3, 1/3]
[   1,   1]
>>r=det(A)
r =  1/3
```

Müşayət edən funksiya: COND.

Matrisin xarakteristik çoxhədlisi - POLY

***Sintaksis*:**

P=poly(A)

P=poly(A,v)

***Təsviri*: P=poly(A)** funksiyası A matrisinin xarakteristik çoxhədlisinin əmsallarını hesablayır. Əgər A matrisi simvol şəklində formalaşırsa, onda polinomda simvol şəklində formalaşır və bu zaman x dəyişənindən susmaya görə istifadə olunur.

P=poly(A,v) funksiyası çoxhədlinin asılı olmayan dəyişənini göstərməyə imkan verir.

Qeyd: Əgər A - ədədi matrisdirsə, onda poly(sym(A)) çoxhədlisi yuvarlaqlaşdırma səhvi hüdudunda ancaq **poly2sym(poly(A))** çoxhədlisinə təqribi uyğundur.

Nümunə:

```
>>A=gallery(3)
A =
   -149     -50    -154
    537     180     546
    -27      -9     -25

>>p=poly(A)
p = 1.0000    -6.0000    11.0000    -6.0000

>>q=poly(sym(A))
q = x^3-6*x^2+11*x-6

>>s=poly(sym(A),sym('z'))
s =  z^3-6*z^2+11*z-6
```

Müşayət edən funksiyalar: POLY2SYM, JORDAN, EIG, SOLVE.

Matrisin diaqonallarının çıxarılması və ya formalaşdırılması- DIAG

***Sintaksis*:**

X=diag(v) **v=diag(X)**

X=diag(v,k) **v=diag(X,k)**

***Təsviri*:**

X=diag(v) funksiyası baş diaqonalı v vektoru olan X kvadrat matrisini formalaşdırır.

X=diag(v,k) funksiyası k nömrəli diaqonalı v – vektoru olan length(v)+abs(k) tərtibli X kvadrat matrisini formalaşdırır.

v=diag(X) funksiyası X matrisindən baş diaqonalı çıxarır.

v=diag(X,k) funksiyası X matrisindən k nömrəli diaqonalı çıxarır; k>0 olduqda bu k nömrəli yuxarı diaqonaldır, k<0 olduqda bu k nömrəli aşağı diaqonaldır.

Nümunə:

```
>>syms a b c
>>v=[a b c];
>>diag(v)
>>diag(v,-2)

ans =                                   ans =
[ a, 0, 0]                              [ 0, 0, 0, 0, 0]
[ 0, b, 0]                              [ 0, 0, 0, 0, 0]
[ 0, 0, c]                              [ a, 0, 0, 0, 0]
                                        [ 0, b, 0, 0, 0]
                                        [ 0, 0, c, 0, 0]

>>syms
Columns 1 through 9
    'B'    'X'    'a'    'a11'    'a12'    'a21'    'a22'    'ans'    'b'
Columns 10 through 19
    'b1'    'b2'    'c'    'd'    'q'    'r'    's'    'v'    'x1'    'x2'
>>syms x y z
>>diag(A)
>>diag(A,1)

>>A=[a,b,c;1,2,3;x,y,z]
          ans =                 ans =
A =                                 [ a]                [ b]
[ a, b, c]                          [ 2]                [ 3]
[ 1, 2, 3]                          [ z]
[ x, y, z]
```

Müşayət edən funksiyalar: TRIL, TRIU.

Aşağı üçbucaqlı matrisin (massivin) formalaşdırılması - TRIL

Sintaksis:

L=tril(X)

L=tril(X,k)

Təsviri:

L=tril(X) funksiyası X matrisinin aşağı üçbucaqlı hissəsini saxlayır.

L=tril(X,k) funksiyası X matrisinin k nömrəli diaqonalından başlayaraq aşağı üçbucaqlı hissəsini saxlayır. k>0 olduqda bu, k nömrəli yuxarı diaqonaldır, k<0 olduqda bu, k nömrəli aşağı diaqonaldır.

Nümunə:

```
>>syms a b c
>>A=[a,b,c;1,2,3;a+1,b+2,c+3]
A =
[     a,     b,     c]
[     1,     2,     3]
[ a+1, b+2, c+3]

>>tril(A)
ans =
[     a,     0,     0]
[     1,     2,     0]
[ a+1, b+2, c+3]

>>tril(A,1)
ans =
[     a,     b,     0]
[     1,     2,     3]
[ a+1, b+2, c+3]

>>tril(A,-1)
ans =
[     0,     0,     0]
[     1,     0,     0]
[ a+1, b+2,     0]
```

Müşayət edən funksiyalar: DIAG, TRIU.

Yuxarı üçbucaqlı matrisin (massivin) formalaşdırılması - TRIU

Sintaksis:

U=triu(X)

U=triu(X,k)

Təsviri:

U=triu(X) funksiyası X matrisinin (massivinin) yuxarı üçbucaqlı hissəsini saxlayır.

U=triu(X,k) funksiyası X matrisinin (massivinin) k nömrəli diaqonalından başlayaraq yuxarı üçbucaqlı hissəsini saxlayır. k>0 olduqda bu, k nömrəli yuxarı diaqonaldır, k<0 olduqda bu, k nömrəli aşağı diaqonaldır.

Nümunə:

```
>>syms a b c
>>A=[a,b,c;1,2,3;a+1,b+2,c+3]
>>triu(A)
A =                                    ans =
[     a,     b,     c]                 [     a,     b,     c]
[     1,     2,     3]                 [     0,     2,     3]
[ a+1, b+2, c+3]                       [     0,     0, c+3]

>>triu(A,1)                           >>triu(A,-1)
ans =                                  ans =
[ 0, b, c]                             [     a,     b,     c]
[ 0, 0, 3]                             [     1,     2,     3]
[ 0, 0, 0]                             [     0, b+2, c+3]
```

Müşayət edən funksiyalar: DIAG, TRIL.

Tam ədədli matrislərin yuxarı uçbucaqlı formaya gətirilməsi - RREF

Sintaksis:

rref(A)

Təsviri: **rref(A)** funksiyası simvol matrisin yuxarı üçbucaq formaya gətirilməsini yerinə yetirir.

Nümunə:

```
>>rref(sym(magic(4)))
ans =
[    1,    0,    0,    1]
[    0,    1,    0,    3]
[    0,    0,    1, -3]
[    0,    0,    0,    0]
```

Bu ranqı 3 olan 4×4 ölçülü matrisdir.

Müşayət edən funksiyalar: RANK, NULL, SVD.

Tam ədədli matrislərin ranqı - RANK

Sintaksis:

rank(A)

Təsviri: **rank(A)** funksiyası A simvol matrisinin ranqını hesablayır.

Nümunə:

```
>>syms a b c d
>>rank([a b;c d])
 ans =    2
```

```
>>rank(sym(magic(4)))
ans =    3
```

Müşayət edən funksiyalar: PREF, NULL, SVD.

Tam ədədli matris sütunlarının bazis fəzası - COLSPACE

***Sintaksis*:**

B=colspace(A)

***Təsviri*: B=colspace(A)** funksiyası matris formalaşdırır. Formalaşan matrisin sütunları tam ədədli A matrisi sütunlarının bazis fəzalarından ibarətdir. B matrisinin sütunlarının sayı A matrisinin ranqına bərabərdir.

Nümunə:

```
>>B=colspace(sym(magic(4)))
B =
[    0,    0,    1]
[    0,    1,    0]
[    1,    0,    0]
[ -3,    3,    1]
```

B matrisi 3 sütuna malikdir və aydındır ki, magic(4) matrisinin ranqı 3-ə bərabərdir.

Müşayət edən funksiya: NULL.

Tam ədədli matrislər üçün sıfır-fəzalar- NULL

***Sintaksis*:**

Z=null(A)

***Təsviri*: Z=null(A)** funksiyası matris formalaşdırır. Formalaşan matrisin sütunları A tam ədədli matrisin sıfır-fəza bazisindən ibarətdir. Z matrisin sütunlarının sayı sıfır-fəzanın ölçüsünü müəyyən edir. A*Z=0. Əgər A matrisi tam ranqa malikdirsə, onda Z - boş matrisdir.

Nümunə:

```
>>A=[1 2 3;1 2 3;1 2 3]
A =
     1     2     3
     1     2     3
     1     2     3
```

```
>>null(sym(A))
ans =
[ -2, -3]
[    1,    0]
[    0,    1]
```

```
>>A=sym(magic(4));
>>Z=null(A)
Z =
[ -1]
```

```
>>A*Z
ans =
 [ 0]
```

```
[ -3]                                                    [ 0]
[  3]                                                    [ 0]
[  1]                                                    [ 0]
```

Müşayət edən funksiyalar: COLSPACE, RANK, RREF, SVD.

Simvollu və ya tam ədədli matrislərin çevrilməsi - INV

***Sintaksis*:**

R=inv(A)

Təsviri: **R=inv(A)** funksiyası A matrisinin tərsi olan matrisi formalaşdırır.

Nümunə:

Aşağıdakı tam ədədli matrisin çevrilməsini yerinə yetirək:

```
>>A=sym([2,-1,0;-1,2,-1;0,-1,2]);
>>inv(A)
ans =
[ 3/4, 1/2, 1/4]
[ 1/2,   1, 1/2]
[ 1/4, 1/2, 3/4]
```

Aşağıdakı simvol matrisin çevrilməsini yerinə yetirək:

```
>>syms a b c d
>>A=[a b;c d]                          >>inv(A)
A =                                       ans =
[ a, b]                                [   d/(d*a-b*c), -b/(d*a-b*c)]
[ c, d]                                [ -c/(d*a-b*c),   a/(d*a-b*c)]
```

Fərz edək ki, N×N ölçülü Hilbert simvol matrisini formalaşdıran aşağıdakı M - fayl mövcuddur.

```
function A=genhilb(N)
% N×N ölçülü Hilbert simvol matrisinin formalaşdırılması:
syms t
for i=1:n
        for j=1:N
                        A(i,j)=1/(i+j-t)
                end
end
>>H=genhilb(2)
H=
        [1/(2-t), 1/(3-t)]
        [1/(3-t), 1/(4-t)]
>>inv(H)
```

```
ans=
[    -(-3+t)^2*(-2+t), (-3+t)*(-2+t)*(-4+t)]
[(-3+t)*(-2+t)*(-4+t),          -(-3+t)^2*(-4+t)]
```

Müşayət edən funksiya: VPA.

Matrislərin sinqulyar və spektral ayrılışı

Simvollu və ya tam ədədli matrislərin sinqulyar ayrılışları - SVD

***Sintaksis*:**

sigma=svd(A)
sigma=svd(vpa(A))
[U,S,V]=svd(A)
[U,S,V]= svd(vpa(A))

***Təsviri*:**

sigma=svd(A) funksiyası simvol A matrisinin sinqulyar ədədlər vektorunu formalaşdırır.

sigma=svd(vpa(A)) funksiyası verilmiş dəqiqliklə hesablamadan istifadə edərək A matrisinin sinqulyar ədədlər vektorunun ədədi qiymətini hesablayır.

[U,S,V]=svd(A) və **[U,S,V]= svd(vpa(A))** funksiyaları elə U və V unitar matrislərini, həmçinin S diaqonal matrislərini hesablayır ki, onlar sinqulyar ədədlərdən ibarətdir və A=U*S*V′ ayrılışı ödənilir. Belə hesablamalar ancaq ədədlərlə yerinə yetirilir.

Nümunə:

```
>>digits(4)
>>A=sym(magic(4))
A =
[ 16,    2,    3, 13]
[    5, 11, 10,    8]
[    9,    7,    6, 12]
[    4, 14, 15,    1]
```

```
>>svd(A)
ans =
[                0]
[               34]
[  2*5^(1/2)]
[  8*5^(1/2)]
```

```
>>svd(vpa(A))
ans =
[ .3108e-6*i]
[           4.472]
[           17.89]
[           34.00]
```

```
>> [U,S,V]=svd(A)

U =

  [ -.5000,    .6708,    .5000, -.2236]
[ -.5000, -.2236, -.5000, -.6708]
[ -.5000,    .2236, -.5000,    .6708]
[ -.5000, -.6708,    .5000,    .2236]
S =
```

```
[      34.00,          0,          0,          0]
[          0,      17.89,          0,          0]
[          0,          0,      4.472,          0]
[          0,          0,          0, .8346e-15]

V =

[ -.5000,    .5000,    .6708, -.2236]
[ -.5000, -.5000, -.2236, -.6708]
[ -.5000, -.5000,    .2236,    .6708]
[ -.5000,    .5000, -.6708,    .2236]
```

Müşayət edən funksiyalar: DIGITS, EIG, VPA.

Simvol matrislərin məxsusi vektorları və məxsusi qiymətləri - EIG

Sintaksis:

lambda=eig(A)
[V,D]=eig(A)
[V,D,P]=eig(A)
lambda=eig(vpa(A))
[V,D]=eig(vpa(A))

***Təsviri*:**

lambda=eig(A) funksiyası A simvol matrisinin məxsusi qiymətlərinin lambda vektorunu formalaşdırır.

[V,D]=eig(A) funksiyası V sağ məxsusi vektorlar matrisini və məxsusi qiymətlərin diaqonal matrisini qaytarır. Əgər V matrisinin ölçüsü A matrisinin ölçüsü ilə üst-üstə düşərsə, onda A xətti asılı olmayan məxsusi vektorların tam sisteminə malikdir və A*V=V*D spektorial ayrılışı ödənilir.

[V,D,P]=eig(A) funksiyası həmçinin P indeksli vektoru qaytarır. Bu vektorun uzunluğu xətti asılı olmayan vektorların sayına bərabərdir və A*V=V*D(P,P) şərti ödənilir.

lambda=eig(vpa(A)) və **[V,D]=eig(vpa(A))** funksiyaları məxsusi qiymətlərin və məxsusi vektorların ədədi qiymətlərini hesablayır (dəyişən dəqiqlikli hesablama ilə). Əgər A matrisi məxsusi vektorların tam sisteminə malik deyilsə, onda V matrisinin sütunları xətti asılı olacaqdır.

Nümunə:

Resser matrisinin məxsusi qiymətləri üçün analitik həlli müəyyən edək:

```
>>R=sym(gallery('rosser'));
>>eig(R)
ans =
```

```
[                        0]
[                     1020]
[ 510+100*26^(1/2)]
[ 510-100*26^(1/2)]
[     10*10405^(1/2)]
[    -10*10405^(1/2)]
[                     1000]
[                     1000]
```

Bu qiymətləri hesabda dəyişən dəqiqliklə hesablayaq:

```
>>eig(vpa(R))
ans =
[       -1020.]
[ .1031e-12]
[    .9805e-1]
[         1000.]
[         1000.]
[         1020.]
[         1020.]
[         1020.]
>>Ch=sym(gallery('chebspec',4));
>> [V,D,p]=eig(Ch^2)
V =                                                    D =
[    0,    1]                                          [ 0, 0, 0, 0]
[    1,    0]                                          [ 0, 0, 0, 0]
[    3, -2]                                       [ 0, 0, 0, 0]
[    4, -3]                                            [ 0, 0, 0, 0]

p =         1         2
```

Burada təkrar məxsusi qiymətlərə uyğun məxsusi vektorların iki alt fəzasını əmələ gətirən iki baş məxsusi vektor var. Bu göstərir ki, Ъordan xanası mövcuddur.
Müşayət edən funksiyalar: JORDAN, POLY, SVD, VPA.

Simvol matrisin Jordan kanonik şəkli - JORDAN

***Sintaksis*:**

J=jordan(A)

[V,J]=jordan(A)

***Təsviri*:**

J=jordan(A) funksiyası simvol və ya ədədi A matrisi üçün Ъordan kanonik şəklini

hesablayır. A matrisi mütləq dəqiq verilməlidir, daha doğrusu onun elementləri tam və ya rasional ədədlər olmalıdır. Başlanğıc verilənlərdə ixtiyari xəta Ъordan xanasının strukturunu tamamilə dəyişə bilər.

[V,J]=jordan(A) funksiyası həm Ъordanın kanonik şəklini, həm də sütunları məxsusi vektorların ümumiləşməsi olan oxşar çevrilmiş matrisi hesablayır, V\A*V=J münasibəti ödənilir.

Nümunə:

EIG funksiyası üçün nümunədəki həmin Çebışev matrisinə baxaq və Ъordan xanasının dəqiq strukturunu və ümumiləşdirilmiş məxsusi vektorlar sistemini hesablayaq:

```
>>Ch=sym(gallery('chebspec',4));
>> [V,J]=jordan(Ch^2)
V =
[    116/15,        1/5,       12/5,       -4/5]
[       10/3,          0,          0,          0]
[    -82/15,          1,     -24/5,          1]
[ -148/15,          0,     -36/5,          0]
J =
[ 0, 1, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 1]
[ 0, 0, 0, 0]
```

Üzən vergüllü ikiqat dəqiqliklə hesabda ayrılışın yoxlanmasını yerinə yetirək:

```
>>V=double(V)
V =
      7.7333       0.2000       2.4000      -0.8000
      3.3333            0            0            0
     -5.4667       1.0000      -4.8000       1.0000
     -9.8667            0      -7.2000            0
>>J=V\(Ch^2)*V
J =
[ 0, 1, 0, 0]
[ 0, 0, 0, 0]
[ 0, 0, 0, 1]
[ 0, 0, 0, 0]
```

Müşayət edən funksiyalar: EIG, POLY.

Matrisin eksponentası - EXPM

Sintaksis:

expm(A)

Təsviri: **expm(A)** funksiyası üçün nümunədəki həmin Çebışev matrisinə baxaq və T-nin ixtiyari qiyməti üçün expm(Ch*T) funksiyasını hesablayaq:

```
>>syms T
>>Ch=sym(gallery('chebspec',4));
>>expm(Ch*T)
ans =
[ 1+19/6*T+8/3*T^2+2/3*T^3,       -4/3*T^3-14/3*T^2-4*T,
4/3*T^3+10/3*T^2+4/3*T,     -2/3*T^3-4/3*T^2-1/2*T]
[ T+5/3*T^2+2/3*T^3,   1-4/3*T^3-8/3*T^2-1/3*T,          4/3*T^3+4/3*T^2-T,     -
2/3*T^3-1/3*T^2+1/3*T]
[     -1/3*T-1/3*T^2+2/3*T^3,          -4/3*T^3+4/3*T^2+T,   1+4/3*T^3-
8/3*T^2+1/3*T,          -2/3*T^3+5/3*T^2-T]
[      1/2*T-4/3*T^2+2/3*T^3,   -4/3*T^3+10/3*T^2-4/3*T,       4/3*T^3-
14/3*T^2+4*T, 1-2/3*T^3+8/3*T^2-19/6*T]

>>digits(4)
>>expm(vpa(Ch*1/10))
ans =
[        1.344,       -.4480,        .1680, -.6400e-1]
[        .1173,        .9387, -.8534e-1,    .2933e-1]
[ -.3600e-1,        .1120,        1.008, -.8400e-1]
[    .3734e-1,       -.1013,        .3547,        .7093]
```

Müşayət edən funksiyalar: EIG, VPA.

4.9. Xüsusi funksiyalar

Inteqral kosinus və sinus funksiyaları - COSINT, SININT

***Sintaksis*:**

y=cosint(Z)

y=sinint(Z)

***Təsviri*:** **y=cosint(Z)** funksiyası simvol və ya ədədi A matrisindən inteqral kosinusu hesablayır, belə ki, A matrisi kompleks də ola bilər. Riyazi funksiya olan inteqral kosinus aşağıdakı şəkildə müəyyən edilir:

$$\mathrm{Ci}(z) = \gamma + \ln(z) + \int_0^z \frac{\cos(t)-1}{t}\,dt, \quad |\arg(z)| < \pi$$

.

y=sinint(Z) funksiyası simvol və ya ədədi A matrisindən inteqral sinusu hesablayır, A matrisi kompleks də ola bilər. Riyazi funksiya olan inteqral sinus aşağıdakı şəkildə müəyyən edilir:

$$\mathrm{Si}(z) = \int_0^z \frac{\sin(t)}{t}\,dt$$

.

Nümunələr:

Inteqral kosinus və inteqral sinus funksiyalarının z=π nöqtəsində qiymətlərini hesablayaq:

```
>>cosint(pi)
ans =       0.0737

>>sinint(pi)
ans =       1.8519
```

Inteqral kosinus funksiyasının [0,1] intervalında 0,1 addımla qiymətlərini hesablayaq:

```
>>cosint([0:0.1:1])
ans =
Columns 1 through 7
Inf     -1.7279     -1.0422     -0.6492     -0.3788     -0.1778     -0.0223
Columns 8 through 11
      0.1005      0.1983      0.2761      0.3374
```

Inteqral sinus funksiyasının [0,1] intervalında 0,1 addımla qiymətlərini hesablayaq:

```
>>sinint([0:0.1:1])
ans =
   Columns 1 through 7
   0       0.0999       0.1996       0.2985       0.3965       0.4931       0.5881
   Columns 8 through 11
   0.6812       0.7721       0.8605       0.9461
```

Simvol şəklində inteqral kosinus və sinus funksiyalarının törəməsini hesablayaq:

```
>>syms x;
>>f=cosint(x);
>>diff(f)
ans = cos(x)/x
```

```
>>syms x;
>>f=sinint(x);
>> diff(f)
ans = sin(x)/x
```

W - Lambert funksiyası - LAMBERTW

***Sintaksis*:**

W=lambertw(X)

W=lambertw(k,X)

***Təsviri*:**

W=lambertw(X) funksiyası W*exp(W)=X şəklində transandent tənliyinin W-Lambert funksiyası adlanan həllini tapır.

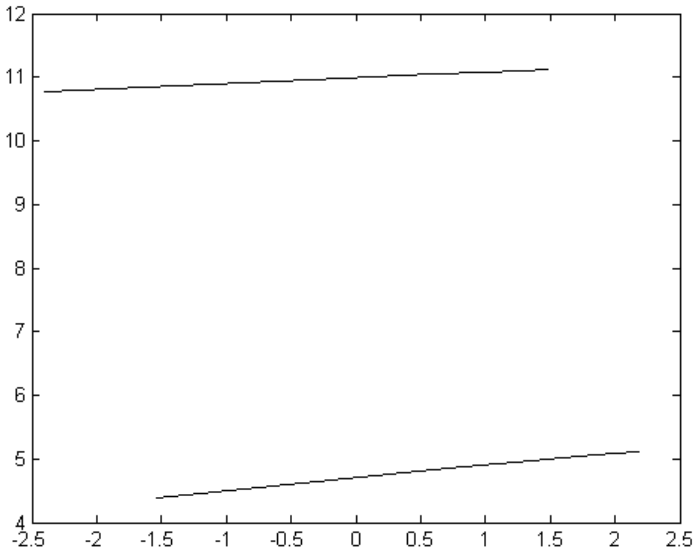
[1-3].

W=lambertw(k,X) funksiyası çoxqiymətli W(X) funksiyasının k nömrəli kompleks budağını tapır.

Nümunə:

Qrafikdə (şək. 4.2) x-in həqiqi qiymətləri üçün W-Lambert funksiyasının iki (k=1,2) kompleks-qiymətli budaqları qurulmuşdur:

```
>>x=[1,10:10:50];
>>y1=lambertw(1,x);
>>y2=lambertw(2,x);
>>plot(y1),hold on, plot(y2)
```



Şəkil 4.2.

Rimanın Zeta funksiyası - ZETA

Sintaksis:

y=zeta(X)

y=zeta(n,X)

Təsviri:

y=zeta(X) funksiyası ədədi və ya simvol massivindən istifadə edərək Rimanın ζ-funksiyasını hesablayır.

y=zeta(n,X) funksiyası ζ-funksiyası aşağıdakı kimi müəyyən edilir:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s}, \quad \mathrm{Re}(s) > 0$$

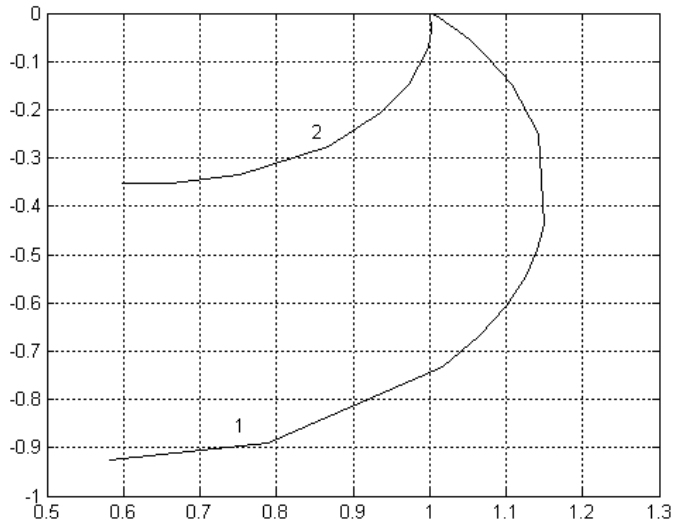
.

Nümunə:

Arqumentin s=i+a1, 2*i+a2 kompleks qiymətləri üçün Rimanın ζ-funksiyasını

hesablayın və qrafikini qurun (şək. 4.3):

```
>>a1=[1 1.2 1.5 1.6 1.7 1.8 1.9 2.0 2.5 3.0 3.5 4.0 4.5 5:10];
>>yz=zeta(i+a1);
>>a2=[1.0 1.2 1.5 2.0 2.5 3.0 4.0 5.0 10.0];
>>yz2=zeta(2*i+a2);
>>plot(real(yz),imag(yz)),grid,hold on,...
>>plot(real(yz2),imag(yz2))
```



Şəkil 4.3

5. SIMULINK PROQRAMI HAQQINDA ÜMUMI MƏLUMAT

Simulink proqramı MATLAB paketinə əlavədir. Simulinkdən istifadə edərək modelləşdirdikdə vizual proqramlaşdırma prinsipi realizə edilir, buna uyğun, istifadəçi standart bloklar kitabxanasından monitora qurğunun modelini yaradır və hesabatı həyata keçirir. Bu zaman klassik modelləşmə xüsusiyyətindən fərqli olaraq istifadəçiyə proqramlaşdırma dilini və riyazi ədədi metodları ətraflı öyrənmək lazım gəlir, ancaq kompüterdə işləmək üçün tələb olunan ümumi biliklər və təbii ki, işlədiyi fənn üzrə biliklər kifayət edir.

Simulink MATLAB-ın kifayət qədər sərbəst alətidir və onunla işlədikdə heç də **MATLAB** və onun digər əlavələrini tam bilmək tələb olunmur. Digər tərəfdən **MATLAB** funksiyalarına və onun digər alətlərinə giriş acıqdır və onlardan **Simulink**-də istifadə etmək olar.

Paketə daxil olan təşkiledicilərin bir hissəsi **Simulink**-də qoyulmuş alətlərdən (məs., **LTI - Viewer. - Control System Toolbox**-a əlavə, idarəetmə sistemini işləmək üçün paket) ibarətdir. Həmçinin müxtəlif sahələrdə tətbiq edilmək üçün əlavə kitabxana blokları (məs., **Power System Blockset** elektromexaniki qurğuların modelləşdirilməsi, **Digital Siqnal Prosessing Blockset** – Siqnalların rəqəmli emalı üçün bloklar dəsti) vardır.

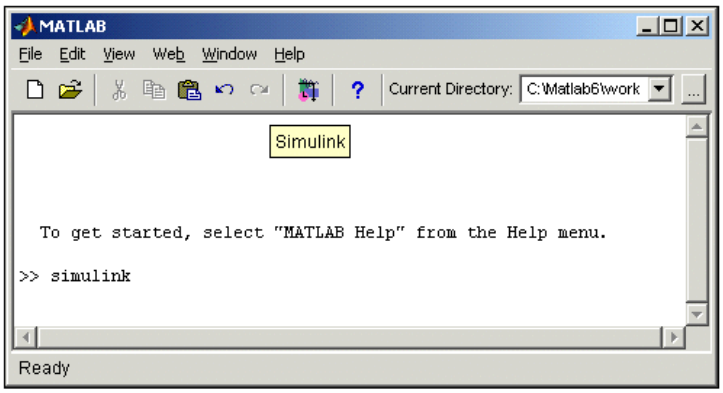
Simulink-lə işlədikdə istifadəçi kitabxana bloklarını modernizə etmək, özü üçün yenisini yaratmaq, həmçinin yeni kitabxana blokları tərtib etmək imkanına malikdir.

Modelləşdirmə zamanı istifadəçi differensial tənlikləri həll metodlarını, həmçinin model müddətinin dəyişmə xüsusiyyətini (qeyd olunmuş və dəyişən addımlı) seçə bilər. Modelləşdirmə gedişində sistemdə gedən prosesləri izləmək imkanı vardır. Bunun üçün **Simulink**-in kitabxanasına daxil olan xüsusi müşahidə qurğusundan istifadə olunur. Modelləşdirmənin nəticələri qrafiklər və ya cədvəllər şəklində təqdim etmək olar.

Simulink-in üstünlüyü ondan ibarətdir ki, kitabxanalar blokunu MATLAB, C++, Fortran və Ada dillərində yazılmış altproqramlarla doldurmağa imkan verir.

5.1. Simulinkin işə salınması

Proqramı işə salmaq üçün əvvəlcədən MATLAB paketini buraxmaq lazımdır. MATLAB paketinin əsas pəncərəsi şəkil 5.1-də göstərilmişdir.

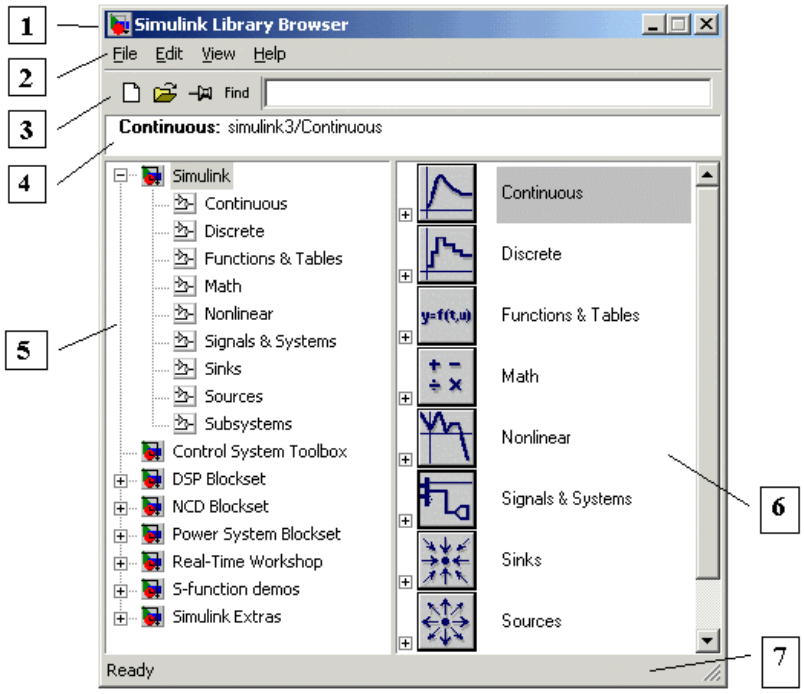


Şəkil 5.1. MATLAB paketinin əsas pəncərəsi

MATLAB proqramının əsas pəncərəsini açdıqdan sonra Simulink proqramını işə buraxmaq lazımdır. Bunu aşağıdakı 3 üsuldan biri ilə yerinə yetirmək olar:

1. MATLAB-ın əmrlər pəncərəsindəki alətlər panelində  (Simulink) düyməsini basmaqla.
2. MATLAB-ın baş pəncərəsinin əmrlər sətrində Simulink çap etmək və Enter düyməsini basmaq.
3. File menyusunda **Open...** əmrini yerinə yetirmək və fayl modelini (**mdl- fayl**) açmaq.

Axırıncı variantdan istifadə etmək o vaxt əlverişlidir ki, artıq hazır və sazlanmış model işə buraxılsın, bu zaman ancaq hesablamalar tələb olunur və modelə yeni bloklar əlavə etmək tələb olunur. Birinci və ikinci variantlardan istifadə icmalçı pəncərədə Simulink kitabxanalar bölməsinin açılmasına gətirib çıxarır (şəkil 5.2).



Şək. 5.2.

5.2. Simulink kitabxanalar bölməsinin icmalı

Blokların kitabxanalar icmalı pəncərəsi aşağıdakı elementləri özündə birləşdirir (şəkil 5.2).

1. *Simulink Library Browser* - adlı pəncərə başlığı;
2. *File, Edit, View*, Help əmrləri menyusu;
3. Daha tez-tez istifadə olunan qısayol əmrlərin alətlər paneli;
4. Saçılmış blokda şərh edilən mətnin çıxarılması üçün şərh pəncərəsi;
5. Ağacşəkilli kitabxanalar bölməsinin siyahısı;
6. Kitabxanalar bölməsini birləşdirən pəncərə;
7. Hərəkətin yerinə yetirilməsi haqqında məlumat verən hal sətirləri.

Şəkil 5.2-də Simulinkin əsas kitabxanası (pəncərənin sol hissəsi) ayrılmış və onun bölmələri (pəncərənin sağ hissəsi) göstərilmişdir.

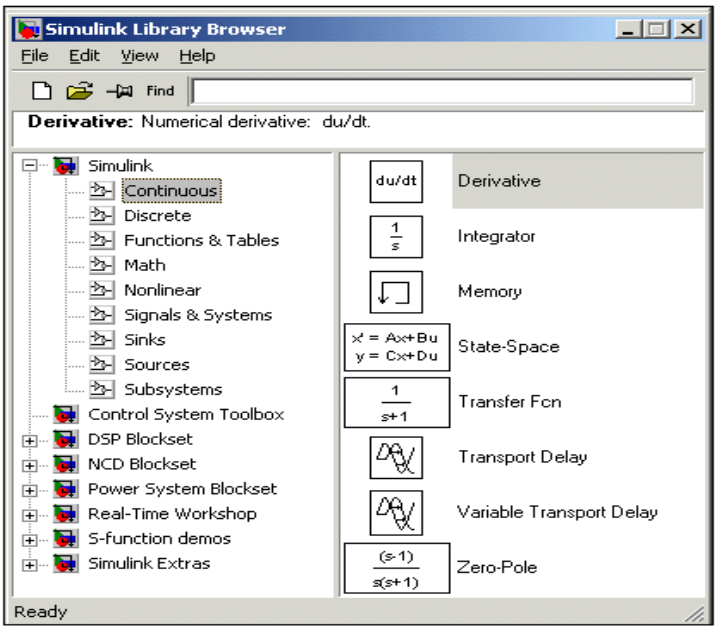
Simulink kitabxanası aşağıdakı əsas bölmələrdən ibarətdir:

1. Contious - xətti bloklar;
2. Discrete - diskret bloklar;
3. Functions & Tables - finksiyalar və cədvəllər;
4. Math - riyazi əməliyyatlar bloku;
5. Nonlinear - qeyri-xətti bloklar;
6. Signals & Systems - siqnallar və sistemlər;
7. Sinks - qeydedici qurğu;
8. Sources - siqnallar və təsirlər mənbəyi;
9. Subsystems - altsistem blokları.

Simulink kitabxanalar bölməsinin siyahısı Ağac şəklində təqdim edilmişdi və onunla işləmə qaydası aşağıdakı növ siyahılar üçün ümumidir:

- Ağac qovşağının yığılmış piktoqramı «+» simvolu ilə, açılmış piktoqramı isə «-» simvolu ilə verilir.
- Ağac qovşağını açmaq və yığmaq üçün onun piktoqramında mouse-un sol düyməsini basmaq lazımdır.

Kitabxanası uyğun bölməsini seçdikdə pəncərənin sağ hissəsində onun məzmunu əks olunur (şəkil 5.3).



Şəkil 5.3. Kitabxana bölməsinin bloklar dəstinin icmal pəncərəsi

Pəncərə ilə işləmək üçün menyuda yığılmış əmrlərdən istifadə olunur.
Kitabxana icmalı menyusu aşağıdakı punktlardan ibarətdir:

- File (Fayl) - kitabxana faylları ilə iş;
- Edit (Redaktəetmə) - blokların əlavə edilməsi və onların adlarına görə axtarışı;
- View (Növ) - interfeys elementlərinin göstərişinin idarə olunması;
- Help (Arayış) - kitabxana icmalının arayışının pəncərəyə çıxarılması.

Kitabxana icmalı menyusunun tam əmrlər siyahısı Əlavə 1-də verilmişdi.
Icmal ilə işləmək üçün, həmçinin, alətlər panelindəki düymələrdən istifadə etmək olar (şəkil 5.4).

Şəkil 5.4. Kitabxanalar bölməsi icmalının alətlər paneli

Alətlər paneli düymələrinin aşağıdakı təyinatları vardır:

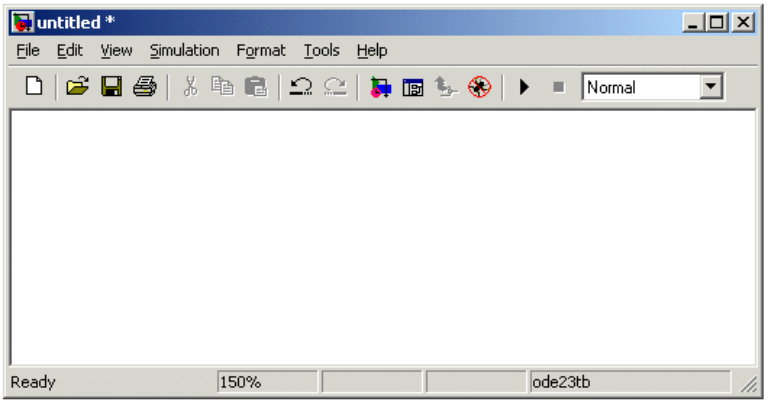
1. Yeni S – modelin yaradılması (yeni model pəncərəsi açmaq);
2. Mövcud S – modellərdən birini açmaq;
3. Icmal pəncərəsinin xüsusiyyətini dəyişmək. Verilmiş düymə «bütün pəncərələrin üstündə» icmal pəncərəsinin əksolunma rejimini qoymağa imkan verir. Düymənin təkrar sıxılması belə rejimi təxirə salır.
4. Blokların adlara görə (adın birinci simvoluna görə) axtarışı. Blok tapıldıqdan sonra, icmal pəncərəsində kitabxananın uyğun bölməsi açılır, blok isə ayrılır. Əgər

belə ad ilə blok mövcud deyilsə, onda şərh pəncərəsində Not found «blokun adı» (Blok tapılmır) məlumatı verəcəkdir.

5.3. Modellərin yaradılması

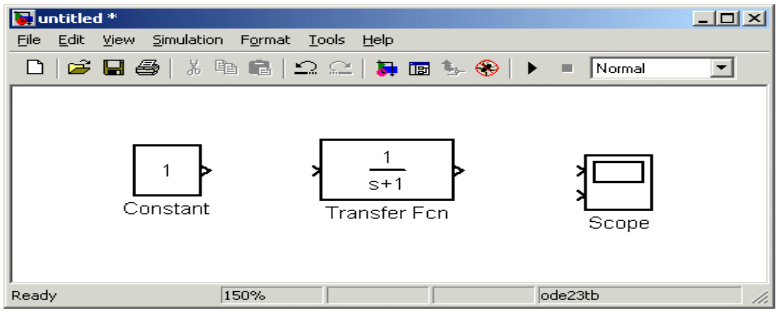
SIMULINK mühitində modelləri yaratmaq üçün aşağıdakı hərəkətləri ardıcıl yerinə yetirmək lazımdır:

Modellərin yeni fayllarını File/New/Model əmrlərinin köməyi ilə və yaxud alətlər panelindəki  düymədən istifadə etməklə (burada və sonra «1» simvolunun köməyi ilə göstərilən hərəkətin yerinə yetirilməsi üçün ardıcıl olar seçməklə proqramlar menyusunun punktları göstərilir) yaradılır. Yeni yaradılmış modellər pəncərəsi şəkil 5.5-də göstərilmişdir.



Şəkil 5.5. Modelin boş pəncərəsi

Model pəncərəsində blokların yerləşdirilməsi. Bunun üçün kitabxananın uyğun bölməsini (məs., Sources - Mənbələr) açmaq lazımdır. Sonra tələb olunan bloku kursorda göstərmək və mouse-un sol klavişini basmaqla bloku yaradılmış pəncərəyə «tutub dartmaq» lazımdır, bu zaman Mouse-un klavişi sıxılmış vəziyyətdə saxlanılmalıdır. Şəkil 5.6-da bloklar daxil olan modellər pəncərəsi göstərilmişdir.

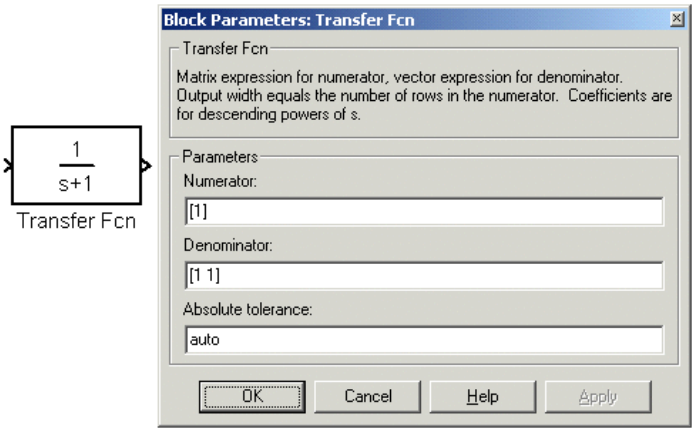


Şəkil 5.6. Bloklar daxil olan modellər pəncərəsi

Bloku yox etmək üçün blok seçilməlidir (onun təsvirini kursorla göstərmək və sol düyməni basmaq), sonra isə klaviaturada **Delete** klavişi basilir. Blokların ölçülərini

dəyişmək üçün bloku seçmək tələb olunur, kursoru blokun künclərindən birinə qoyub, sol klavişi basmaqla yerinə yetirmək olar (kursor bu zaman ikitərəfli oxlu olur).

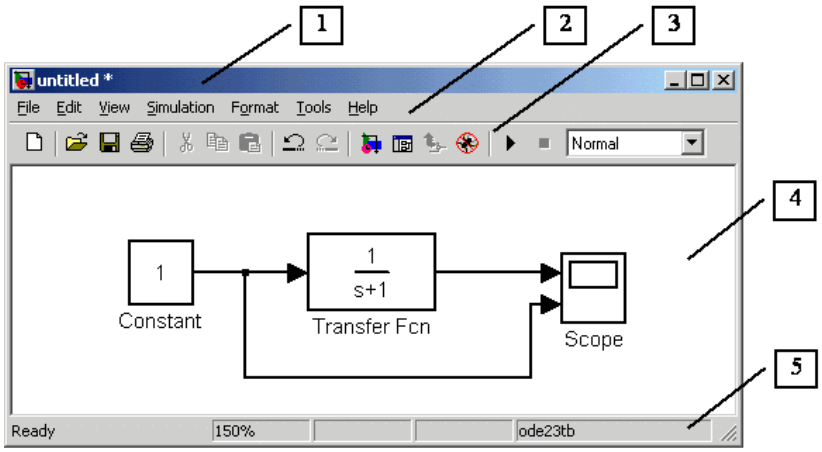
Əgər «var sayılan» (default) proqramı ilə yerləşdirilmiş blokun parametrlərini dəyişmək lazımdırsa, sol klavişi iki dəfə basmaq lazımdır, bu zaman blokun təsvirini kursorla göstərmək lazımdır. Onda verilmiş blokun parametrlərini redaktəetmə pəncərəsi açılır. Ədədi qiymətləri tapşırmaq üçün nəzərə lazımdır ki, onluq kəsrlərlə verilən parametrləri daxil etmək üçün vergüldən yox, nöqtədən istifadə etmək lazımdır. Dəyişikliklər daxil etdikdən sonra **OK** düyməsi ilə pəncərəni bağlamaq lazımdır. Şəkil 5.7-də misal kimi ötürmə funksiyasını modelləşdirilən və onun parametrlərinin redaktəetmə pəncərəsi olan blok göstərilmişdir.



Şəkil 5.7. Ötürmə funksiyası modelləşdirən blok və onun parametrlərinin redaktəetmə pəncərəsi

Tələb olunan kitabxanadan bütün bloklar sxemdə yığıldıqdan sonra sxemin elementlərini birləşdirmək lazımdır. Blokları birləşdirmək üçün kursorla blokların «çıxışını» göstərmək lazımdır, sonra isə mouse-un sol klavişini basıb buraxmadan o biri blokun girişinə xətt çəkmək lazımdır, bundan sonra klavişi buraxmaq lazımdır. Birləşmə düzgün olduqda blokun girişində oxun təsvirinin rəngi dəyişir. Birləşdirmə xətlərində budaqlanma nöqtəsi yaratmaq üçün kursoru qovşağın nəzərdə tutulan yerinə gətirmək və sağ klavişi basmaqla xətt çəkmək lazımdır. Xətti yox etmək (silmək) üçün xətti seçmək lazımdır (bloklar üçün yerinə yetirildiyi kimi), sonra isə klaviaturada **Delete** düyməsini basmaq lazımdır.

Bloklar arasında birləşmənin yerinə yetirildiyi sxemin modeli şəkil 5.8-də göstərilmişdir.



Şəkil 5.8. Modelin sxemi

Hesab işlərini yerinə yetirən sxemləri tərtib etdikdən sonra onu diskdə fayl şəklində saxlamaq lazımdır, bunun üçün File menyusundan *Save As...* bəndini seçib qovluğu və faylın adını göstərmək lazımdır. Nəzərə almaq lazımdır ki, faylın adı 32 simvoldan çox olmamalıdır, hərflə başlamalıdır, kiril simvolları və xüsusi simvollardan istifadə etmək olmaz. Bu tələbatlar fayllar saxlanılan qovluqlara da aiddir. Sxemlərin sonrakı redaktəsi üçün ***File/Save*** menyu bəndindən istifadə edilir.

SIMULINK proqramının təkrar buraxılması üçün sxemin yüklənməsi ***File/Open*** menyusunun köməyi ilə və ya MATLAB-ın əsas pəncərəsindən yerinə yetirilir.

5.4. Modellər pəncərəsi

Modellər pəncərəsi aşağıdakı elementlərdən ibrətdir (şəkil 5.9):

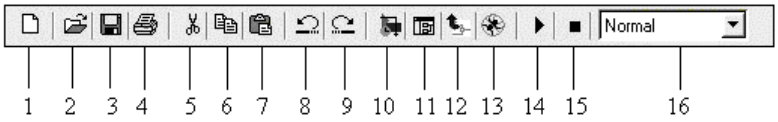
1. Pəncərənin adı olan başlıq (sərlövhə). Yeni yaradılan pəncərə uyğun nömrə ilə *Untitled* adı ilə adlandırılır.
2. File, Edit, View və s. əmrləri menyu.
3. Alətlər paneli.
4. Modellər sxemi yaratmaq üçün pəncərə.
5. Modellərin cari vəziyyətləri haqqında informasiya daşıyan hal sətirləri.

Pəncərə menyusu modelləri redaktə etmək üçün əmrləri, onların sazlanmasını və hesablama proseslərinin idarə olunmasını, fayllarla işləməni və s. özündə birləşdirir:

- File (Fayl) - Modelin faylları ilə işləmək.
- Edit (Redaktəetmə) - Modellərin dəyişdirilməsi və blokların axtarılması.
- View (Növ) - Interfeys elementlərini göstərişinin idarə olunması.
- Simulation (Modelləşmə) - Modelləşmə üçün sazlanmasını tapşırılması və hesabat proseslərinin idarə olunması.
- Format (Formatlaşdırma) - Blokların və bütövlükdə modellərin xarici görünüşünün dəyişdirilməsi.

- Tools (Alət vasitələri) - Modellə işləmək üçün xüsusi vasitələrin (düzənnmsəi, xətti təhlil və s.) tətbiqi.
- Help (Arayış) - Arayış sistemlərinin pəncərəyə çıxarılması.

Modellər pəncərəsi menyusunun əmrlərinin tam siyahısı Əlavə 2-də verilmişdi. Modellə işləmək üçün, həmçinin, alətlər panelindəki düymələrdən də istifadə etmək olar (şəkil 5.9).



Şəkil 5.9. Modellər pəncərəsinin alətlər paneli

Alətlər panelinin düymələrinin təyinatları aşağıdakılardır:

1. *New Model* - Yeni model pəncərəsi (boş) açmaq;
2. *Open Model* - Mövcud mdL-faylını açmaq;
3. *Save Model* - mdL-faylını diskdə saxlamaq;
4. *Print Model* - Modelin blok-diaqramını çap çıxarmaq;
5. *Cut* - Aralıq saxlamaq buferində modelin ayrılmış hissəsini kəsmək;
6. *Copy* - Aralıq saxlama buferində modelin ayrılmış hissəsini köçürmək;
7. *Paste* - Aralıq saxlama buferində olan model pəncərəsinə qoymaq;
8. *Undo* - Əvvəlki redaktəetmə əməliyyatının təxirə salınması;
9. *Redo* - Təxirə salınmış redaktəetmə əməliyyatının nəticələrinin bərpası;
10. *Library Browser* - Kitabxana icmalı pəncərəsinin açılması;
11. *Toggle Model Browser* - Modellər icmalı pəncərəsinin açılması;
12. *Goto parent system* - Alt sistemdən ierarxiyanın yarır səviyyə sisteminə keçid («Valideyn sistemi»). Əmrlər yalnız o vaxt mümkündür ki, altsistem acıq olsun;
13. *Debug* – Model düzənləyicinin yüklənməsi;
14. *Start/Pause/Continue Simulation* - Modellərin icra edilməsi (Start əmri);

    Model işə buraxıldıqdan sonra düymənin təsvirində [II] simvolu çıxır və ona artıq Rause əmri uyğun olur (modelləşmənin durdulması), modelləşməni yeniləşdirmək üçün elə həmin düyməni basmaq lazımdır, belə ki, fasilə rejimində ona *Continue* (Davametmə) əmri uyğundur;
15. *Stop* - Modelləşməni qurtarmaq;

    Düymə modelləşməni başladıqdan sonra, həmçinin *Pause* əmri yerinə yetirildikdən sonra müdaxilə etmək olur.
16. *Normal/Accelerator* - Adi/Cəld hesabat rejimi. Alət o vaxt müdaxilə edilə bilən olur ki, *Simulink Performance Tool* əlavəsi qoyulmuş olsun.

Model pəncərəsinin aşağı hissəsində hal sətirləri yerləşir ki, burda alətlər panelinin düymələrinə, həmçinin, mouse-un göstəricisi interfeysin uyğun elementi üzərində olduğu halda menyu punktun qısa şərh əks olunur.

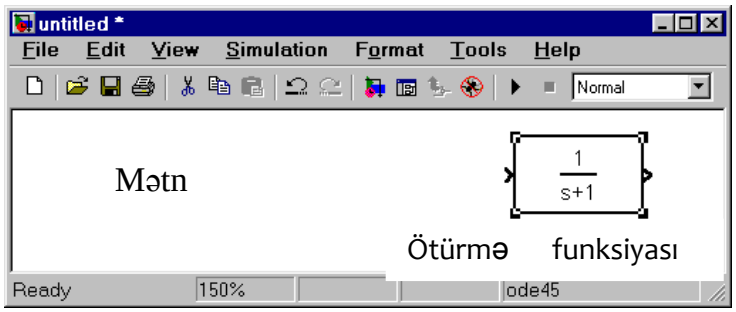
Bu mətn sahəsi həm də Simulink: *Ready* (Hazır) və ya *Running* (Yerinə yetirmə) halını indikasiya etmək üçün istifadə edilir. Hal sətrində, həmçinin, aşağıdakılar əks olunur:

- blok - diaqram təsvirinin miqyası (faizlə, ilkin verilənlər 100%-ə bərbərdir).
- modelləşməni seansının başaçatma dərəcəsi indikatoru (modeli işə saldıqdan sonra (əmələ gəlir) görünür),
- model vaxtının cari qiyməti (bu da model işə buraxıldıqdan sonra çıxarılır),
- model halını hesablama alqoritmindən istifadə (həll metodu).

5.5. Modellərin hazırlanması və redaktə edilməsinin əsas nümunələri

5.5.1. Mətn yazılarının əlavə edilməsi

Modellərin əyaniliyini artırmaq üçün mətn yazılarından istifadə etmək daha əlverişlidir. Yazını yazmaq üçün yazılacaq yer göstərməlidir və mausun sol düyməsi iki dəfə basılmalıdır. Bundan sonra daxiletmə kursorlu düzbucaqlı çərçivə əmələ gəlir. Analoji olaraq model blokları üzərindəki yazıları da dəyişmək olar. Şəkil 5.10-da ötürmə funksiyası blokunda mətn yazıları və onların dəyişdirilməsi göstərilmişdir. Nəzərə almaq lazımdır ki, proqramın bu versiyası (Simulink 4) kiril hərflərindən istifadəyə adaptasiya olunmamışdır və onlardan istifadə müxtəlif fəsadlar verə bilər: yazılar oxunmaz şəklə düşər; yazıların kəsilməsi, səhv haqqında məlumat, həmçinin, saxlanıldıqdan sonra modelin açıla bilməməsi. Ona görə də bu simvollardan istifadə arzuolunmazdır.



Şəkil 5.10. Mətn yazısı və ötürmə funksiyasında yazının dəyişdirilməsi

5.5.2. Obyektlərin seçilməsi

Modelin elementləri üzərində hər hansı bir əməliyyatı yerinə yetirmək üçün (blok, birləşdirici xətlər, üstyazı) bu elementi əvvəlcə seçmək lazımdır.

Obyektlərin seçilməsi mausla daha asandır. Bunun üçün kursoru lazımi obyektin üzərinə qoymaq və sol düyməni basmaq lazımdır. Bu zaman obyektin seçilməsi baş verir. Bu halda obyektin küncündəki marker şəklini dəyişir. Həmçinin bir neçə obyekti

seçmək olar. Bunun üçün kursoru obyektlər qrupunun yaxınlığında yerləşdirmək, sol düyməni basmaq və onu buraxmadan mausu hərəkət etdirmək lazımdır. Mouse-u hərəkət etdikcə ölçüləri dəyişən punktirli (qırıq-qırıq) çərçivə yaranır, Bu çərçivə ilə əhatə olunmuş bütün obyektlər ayrılmış olur. Bütün obyektləri həm də *Edit/Select All* əmrindən istifadə etməklə seçmək olar. Obyekti ayırdıqdan sonra onu köçürmək və ya aralıq saxlama buferinə yerdəyişdirmək, buferdən gətirmək, həmçinin, silmək olar.

5.5.3. Aralıq saxlama buferində Obyektlərin köçürülməsi və yerdəyişdirilməsi

Obyekti buferə köçürmək üçün əvvəlcə onu seçmək lazımdır, sonra isə *Edit/Copy* əmri yerinə yetirilir və ya alətlər panelində  - alətindən istifadə edilir.

Obyekti buferdən kəsmək üçün əvvəlcə onu seçmək lazımdır, sonra isə *Edit/Cut* əmri yerinə yetirilir və ya alətlər panelindən  - alətindən istifadə edilir. Verilmiş əməliyyatı yerinə yetirmək üçün nəzərə almaq lazımdır ki, obyektlər MATLAB-ın məxsusi buferində hərəkət edirlər və başqa əlavələr müdaxilə edilməzdir. *Edit/Copy* modeli clipboard əmrlərində istifadə modelin qrafiki təsvirini Windows buferində yerləşdirməyə imkan verir və uyğun olaraq onu digər proqramlar üçün əldə ediləbilən edir.

Köçürməni, həmçinin, belə formada da yerinə yetirmək olar: mouse-un sağ düyməsini sıxmaq və onu buraxmadan obyekti hərəkət etdirmək. Bu zaman obyektin kopyası yaranacaqdır ki, bunu da lazımi yerə yerləşdirmək (sürüşdürmək) olar.

5.5.4. Aralıq saxlama buferindən obyektin qoyulması

Buferdən obyekti qoymaq üçün əvvəlcə onun qoyulacaq yerini göstərmək lazımdır. Bunun üçün Mouse-un sol düyməsini qoyulması nəzərdə tutulan yerdə sıxmaq lazımdır, sonra isə *Edit/Paste* əmri yerinə yetirilir və ya alətlər panelindəki- alətdən istifadə edilir.

5.5.5. Obyektlərin silinməsi

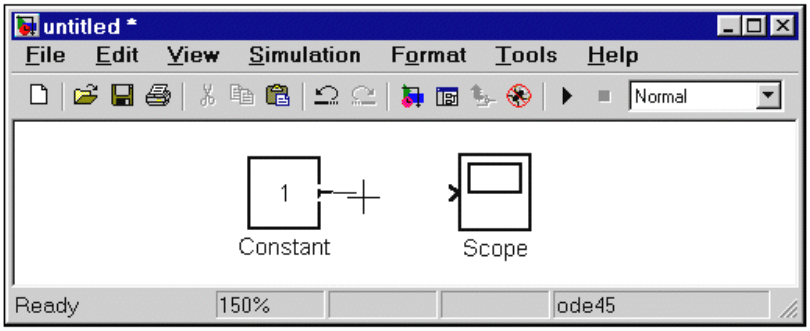
Obyektləri silmək üçün əvvəlcə onları seçmək lazımdır, sonra isə *Edit/Clear* əmri yerinə yetirilir və ya klaviaturanın *Delete* düyməsindən istifadə edilir. Nəzərə almaq lazımdır ki, *Clear* əmri bloku mübadilə buferinə yerləşdirmədən onu silir (yox edir). Ancaq bu əməliyyatı *File/Undo* menyusunun əmri ilə təxirə salmaq olar.

5.5.6. Blokların birləşdirməsi (əlaqələndirilməsi)

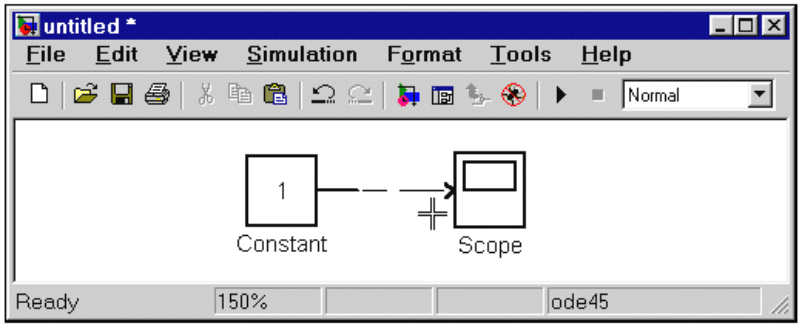
Blokları birləşdirmək üçün əvvəlcə kursoru bloklardan birinin çıxış portunda yerləşdirmək lazımdır. Bu zaman kursor nazik xətti böyük xaça çevrilir (şəkil 5.11). Mouse-un sol düyməsini basılı saxlayaraq kursoru lazımi blokun giriş portuna hərəkət etdirmək lazımdır. Mouse-un kursoru bu zaman nazik iki xətli xaç şəklini alır (şəkil 5.12).

Xətti yaratdıqdan sonra mouse-un sol düyməsini buraxmaq lazımdır. Birləşmənin yaradılmasını blokun giriş portunda qalın xətti oxun olması təsdiq edir. Xətlərin ayrılması

eyni ilə blokların ayrılması kimidir, yəni sol düymə bir dəfə basılır. Birləşdirmə xətərindəki qara markerlərin yerləşməsi xəttin ayrılması göstərir.

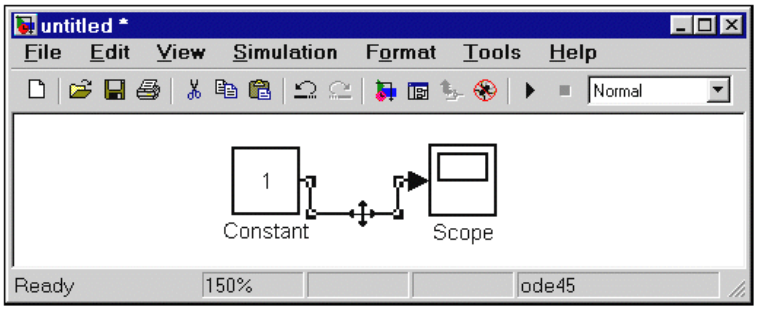


Şəkil 5.11. Birləşmənin yaradılmasının başlanğıcı



Şəkil 5.12. Birləşmənin yaradılmasının sonu

Birləşmə xətləri ilgəyinin yaradılması blokların yerdəyişmələri kimi yerinə yetirilir. Birləşmə xətti ayrılır, sonra isə xəttin lazımi hissəsi hərəkət etdirilir (yerdəyişmə edilir). Şəkil 5.13 bu prosesi aydınlaşdırır.

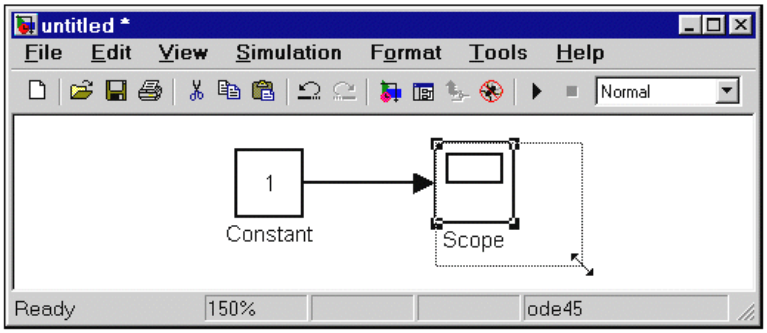


Şəkil 5.13. Birləşdirmə xətlərində ilgəyin yaradılması

Əlaqələrin götürülməsi (silinməsi) istənilən digər obyektlərdə olduğu kimidir.

5.5.7. Blokların ölçülərinin dəyişdirilməsi

Obyektin ölçülərini dəyişmək üçün o əvvəlcə seçilir, bundan sonra kursoru blokun küncündəki markerlərdən birində yerləşdirmək lazımdır. Kursor ikitərəfli oxa çevrildikdən sonra mouse-un sol düyməsini sıxmaq lazımdır və blokun təsvirini dartmaq (və ya sıxmaq) tələb olunur. Şəkil 5.14-də bu proses göstərilmişdir. Bu zaman blokların üstyazılarının ölçüləri dəyişmir.



Şəkil 5.14. Blokların ölçülərinin dəyişməsi

5.5.8. Blokların hərəkət etdirilməsi

Modelin istənilən blokunu sol düyməsi basılı saxlamaqla hərəkət etdirmək, onu seçmək və yerini dəyişdirmək olar. Əgər blokların giriş və çıxışlarına əlaqə xətləri çəkilmişdirsə, onda onlar qırılmır, ancaq onların uzunluqları azalır və ya artır. Birləşməyə, həmçinin, bir giriş və bir çıxışı olan blok da qoymaq (əlavə etmək) olar. Bunun üçün onu birləşmə xəttinin tələb olunan yerinə qoymaq lazımdır.

5.5.9. Undo və Redo əmrlərindən istifadə

Proqramla işlədikdə istifadəçi bəzi əməliyyatları (məs. təsadüfən modelin bir hissəsini silmək, köçürmək və s.) yerinə yetirə bilər. Bu halda *Undo* - sonuncu əməliyyatın təxirə salınması əmrindən istifadə edilir. Bu əmri modellər pəncərəsindəki alətlər panelindəki - düymənin köməyi ilə və ya *Edit* menyusundan çağırmaq olar. Təxirə salınmış əməliyyatı bərpa etmək üçün *Redo* əmrindən istifadə olunur.

5.5.10. Obyektlərin formatlaşdırılması

Format menyusunda (həmçinin, obyektdə sağ klavişi basmaqla) blokların formatlaşdırma əmrlərinin yığımı yerləşir. Formatlaşdırma əmrləri bir neçə qrupa bölünür:

1. Üst yazıların təsvirlərinin dəyişdirilməsi.
   - *Font* - üstyazı şriftlərinin və mətn bloklarının formalaşdırılması.
   - *Text aliqument* - üstyazı mətinlərində yazının bərabərləşdirilməsi.
   - *Flip name* - blokun altyazısının yerdəyişdirilməsi (hərəkəti).
   - *Show/Hind name* - blokun altyazısının təsvir edilməsi və ya gizlədilməsi.
2. Blokların təsvirlərinin rənginin dəyişdirilməsi.

- *Foreqround color* - blokları seçmək üçün xəttin rənginin seçilməsi.
- *Backqround color* - seçilmiş blokların fonunun rənginin seçilməsi.
- *Screen color* - modelin bütün pəncərəsi üçün fon rənginin seçilməsi.

3. Blokun vəziyyətinin və onun şəklinin dəyişdirilməsi.

- *Flip block* - vertikal simmetriya oxuna nəzərən güzgü əksi.
- *Rotate block* - saat əqrəbi istiqamətində blokun 90^o dönməsi.
- *Show drop shadow* - blokdan kölgənin göstərilməsi.
- *Show port labels* - portların işarələnməsinin göstərilməsi.

5.6. Sources – siqnallar mənbəyi

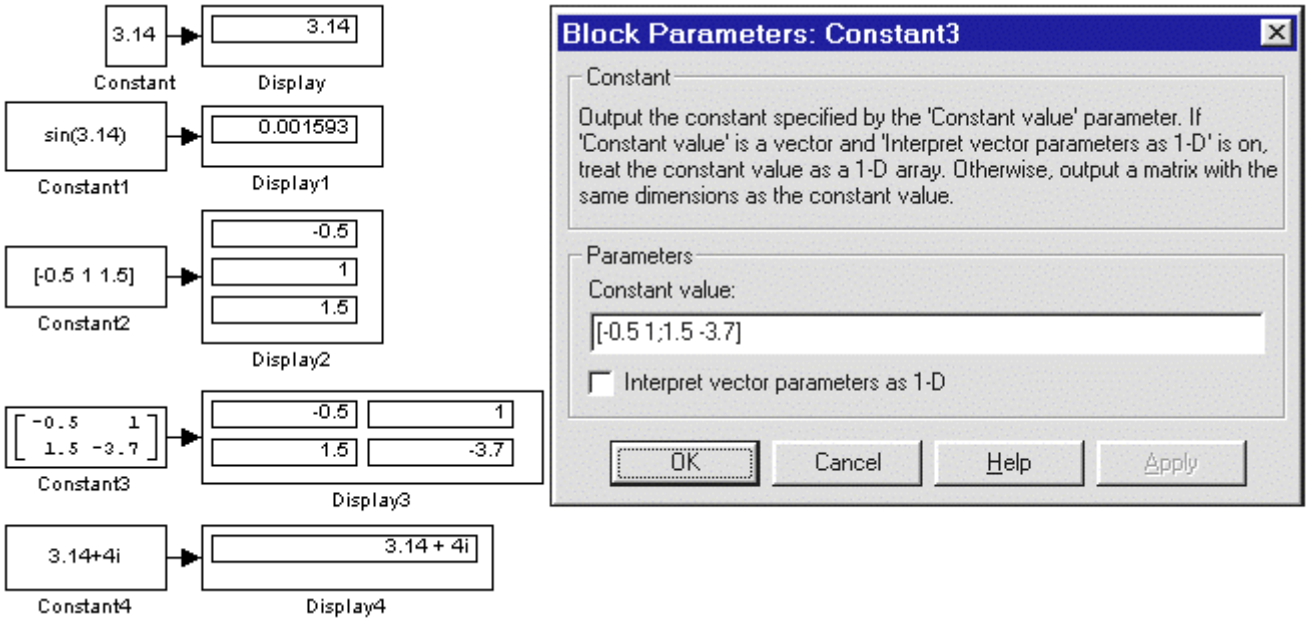
5.6.1. Sabit siqnallar mənbəyi - Constant

***Təyinatı*:** Səviyyəyə görə sabit siqnal verir.

Parametrlər:

1. **Constant value** – sabit kəmiyyət.

2. **Interpret vector parameters as 1-D** – parametrlər vektorunu birölçülü interpretasiya edilsin (bayraq qoyulsun). Bu parametr **Simulink** kitbxanasının bloklarını çoxunda rast gəlinir. Sabit kəmiyyətin qiyməti həqiqi, kompleks, riyazı ifadə, vektor və matris ola bilər. Aşağıdakı şəkil 5.15-də bu mənbənin istifadə olunma nümunəsi və onun çıxış sıqnalının **Display (**rəqəmsal indikator) ilə ölçülməsi göstərilmişdir.



Şəkil 5.15 Sabit kəmiyyət mənbəsi- **Constant**

5.6.2. Sinusoidal siqnal mənbəyi - Sine Wave

***Təyinatı*:** Verilmiş tezliyə, amplitudaya, fazaya, sürüşməyə malik olan sinusodial siqnalı

formalaşdırır.

Çıxış siqnalını formalaşdırmaq üçün iki alqoritmdən istifadə edilir. Alqoritmin tipi **Sine Type** sətrində daxil edilmiş parametr təyin edir (siqnalın formalaşdırma üsulu):

- **Time-based** – cari zamana görə.
- **Sample-based** – model zamanının qiymətinə görə.

5.6.3. Çıxış siqnalının zamanın cari qiymətinə görə formalaşdırılması (kəsilməz sistemlər üçün)

Bu rejimdə formalaşan siqnal aşağıdakı ifadəyə görə generasiya edilir:

y = Amplitude sin(frequency* time + phase) + bias.*

Parametrlər:

1. **Amplitude** – Ampiltuda,
2. **Bias** – siqnalın sabit təşkiledicisi (biləşəni),
3. **Frequency (rads/sec)** - Tezlik (rad/s).
4. **Phase (rads)** – başlanğıc faza (rad).
5. **Sample time** – model zamanının addımı. Mənbə ilə modelin başqa komponetlərinin zamana görə razılaşdırlması üçün istifadə edilir. Paramet aşağıdakı qiymətlərdən birini ala bilər:

   0 (default rejimdə) – kəsilməz sistemlərn modelləşdirilməsi üçün istifadə edilir.

   > 0 (müsbət qiymət) – diskret sistemlərin modelləşdirilməsi üçün istfadə edilir. Bu halda model zaman addımını çıxış sıqnalının kvant addımı kimi interpretasiya etmək lazımdır.

   -1 – model zaman addımı əvvəlki blokda olduğu kimi daxil edilir (yəni haradan siqnal verilmiş bloka gəlir).

Bu parametr **Simulink** kitabxanasında olan əksər bloklar üçün istifadə edilə bilər. Gələcəkdə bu parametrə baxılmayacaqdır. Hesablamalarda zamanın daha böyük qiymətlərindən istfadə edildiyində çıxış sıqnalların hesablama dəqiqliyi düşür. Bunun əsas səbəblərindən biri də çoxlu sayda yuvarlıqlaşmanın olmasıdır.

5.6.4. Diskret sistemlər üçün zamanın cari qiymətinə çıxış sıqnalının formalaşdırılması

Çıxış sıqnalının qiymətinin hesablanması üçün o, matris şəklində verilmiş ifadə ilə təyin edilir (matris şəklində):

$$\begin{bmatrix}\sin(t+\Delta t)\\ \cos(t+\Delta t)\end{bmatrix} = \begin{bmatrix}\cos(\Delta t) & \sin(\Delta t)\\ -\sin(\Delta t) & \cos(\Delta t)\end{bmatrix}\begin{bmatrix}\sin(t)\\ \cos(t)\end{bmatrix}$$

Harada, Δt – Sample time qiymətinə bərabər olan zaman sabitidir.

Bu alqoritmdə yuvarlaqlaşdırma zamanı yaranan xətalar hesablama dəqiqliyi azaldır.

5.6.5. Çıxış siqnalının model zaman qiyməti və bir periodda istifadə edilən hesablama addımlaırnın sayı əsasında formalaşdırılması

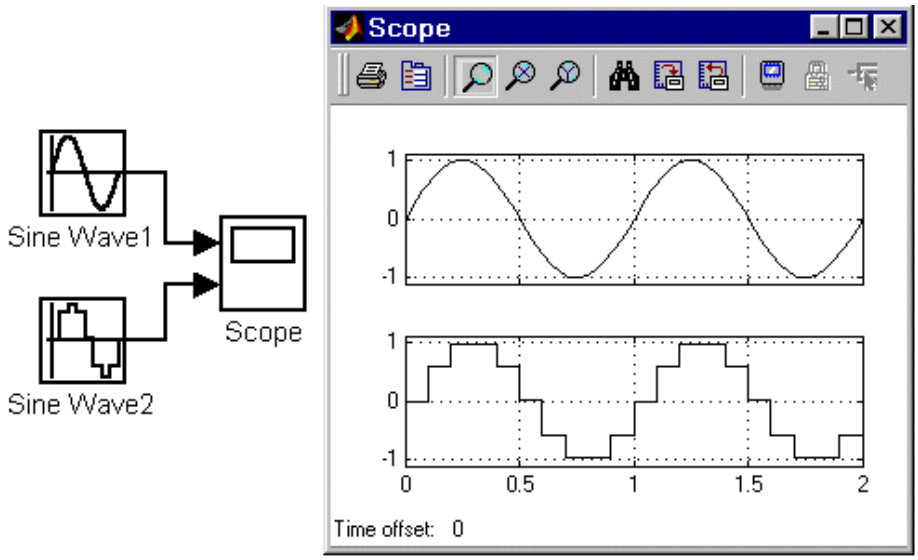
Çıxış sıqanlı bu rejimdə aşağıdakı ifadəyə uyğundur:

*y= Amplitude** sin[(*k + Number of offset samples)* / *Samples per* period] + *bias* ,
harada, ***k*** – hesablamanın cari nömrəsidir.
Parametrlər:

1. **Amplitude** – Amplituda.
2. **Bias** – siqnalın sabit təşkiledicidir.
3. **Samples per period** – sinusoidal siqnalın bir addımında istifadə edilən hesablama addımlaırnın sayı:
   Samples per period = 2*p* / (*frequency** *Sample time)*
4. **Number of offset samples** – siqnalın başlanğıc fazası. Model zamanın addım sayları ilə verilir:
   *Number of offset samples = Phase** *Samples per period*/ (2*p*).
5. **Sample time** – model zamanının addımı.

Bu rejimdə yuvarlaqlaşdırma xətası üst-üstə yığılmır. Bunun səbəbi, **Simulink** –in hər period üçün cari addımı sayarkən sıfırdan başlamasıdır.

Aşağıdakı şəkil 5.16-da bu blokun model zamanının addımının müxtəlif qiymətləri üçün siqnal qiymətləri göstərilmişdir (**Sample time** = 0 birinci siqnal üçün (**Sine Wave 1) və Sample time** = 0.1 ikinci siqnal üçün (**Sine Wave 2)**). Çıxış siqnallarını göstərmək üçün virtual ossiloqrafdan (**Scope**) istifadə edilmişdir.



Şəkil 5.16. **Sine Wave bloku**

5.6.6. Xətti dəyişən mənbə siqnalı - Ramp

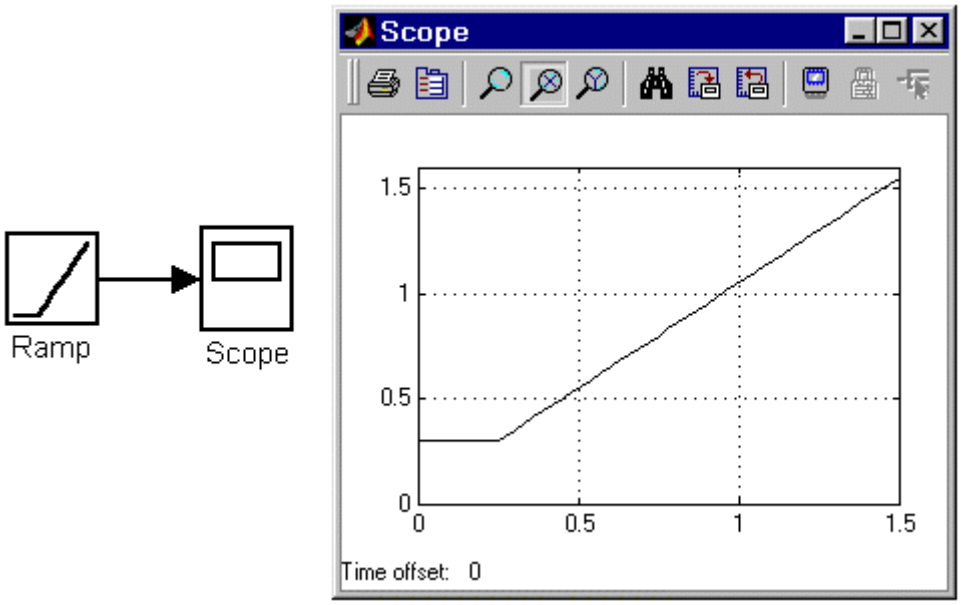
Təyinatı:
aşağıdakı ifadəyə uyğun olaraq xətti siqanlı formallaşdıran bir blokdur

y = *Slope time + Initial value***

Parametrlər:

1. **Slope** — çıxış siqnalın dəyişmə sürəti.
2. **Start time** — sıqnalın formalaşdırılmasında istifadə olunan başlanğıc zaman.
3. **Initial value** — blokun çıxışında istifadə olunan başlanğıc səviyyə.

Aşağıdakı şəkil 5.17-də blokdan istifadə olunma nümunəsi göstərilmişdir.



Şəkil 5.17. **Ramp bloku**

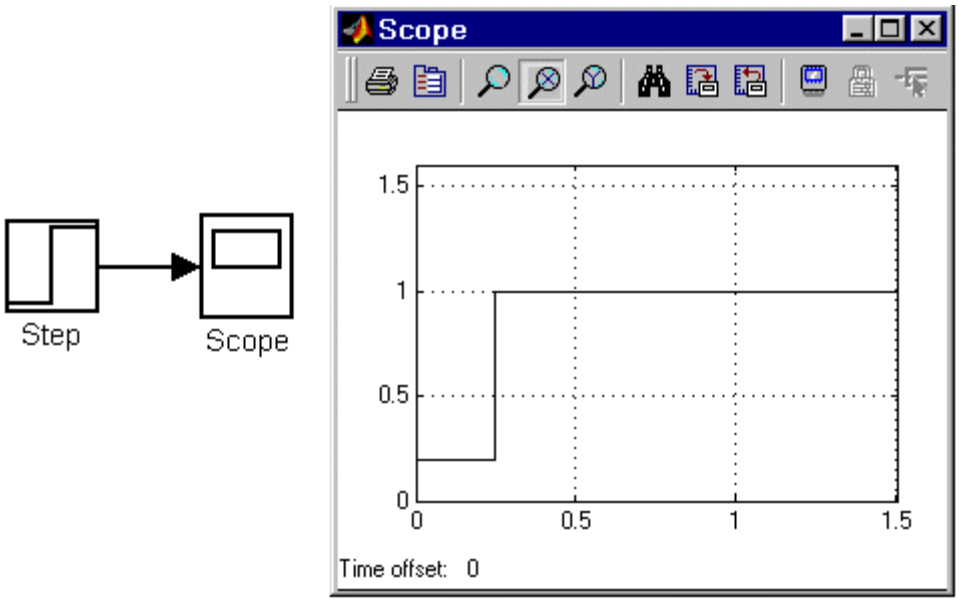
5.6.7. Addım sıqnalını generasiya edən siqnal

***Təyinatı*:** Addım tipli siqnalı formalaşdırır.

***Parametrlər*:**

1. **Step time** – siqnalın sıçrayışla dəyişmə zamanı (san).
2. **Initial value** – siqanlın başlanğıc qiyməti.
3. **Final value** – siqnalın son qiyməti.

Sıçrayış həm artan, həm də azalan tərəfə ola bilər (sonuncu qiymət ilkin qiymətdən böyükdür). Başlanğıc və son qiymətlər həm müsbət, həm də mənfi ola bilər (məsələn, siqnal səviyyəsi –5–dən –3-ə kimi). Şəkil 5.18-də addım sıqnalını generaisya edən blokun istifadəsinə aid nümunə aşağıda göstərilmişdir.



Şəkil 5.18. **Step bloku**

5.6.8. Siqnallar generatoru - Signal Generator

***Təyinatı*:** Aşağıdakı periodik sıqnalları formalaşdırır:

1. **sine** — Sinusoidal sıqnal,
2. **square** — düzbucaqlı sıqnal,
3. **sawtooth** — mişarvari sıqnal,
4. **random** — təsadüfi siqnal.

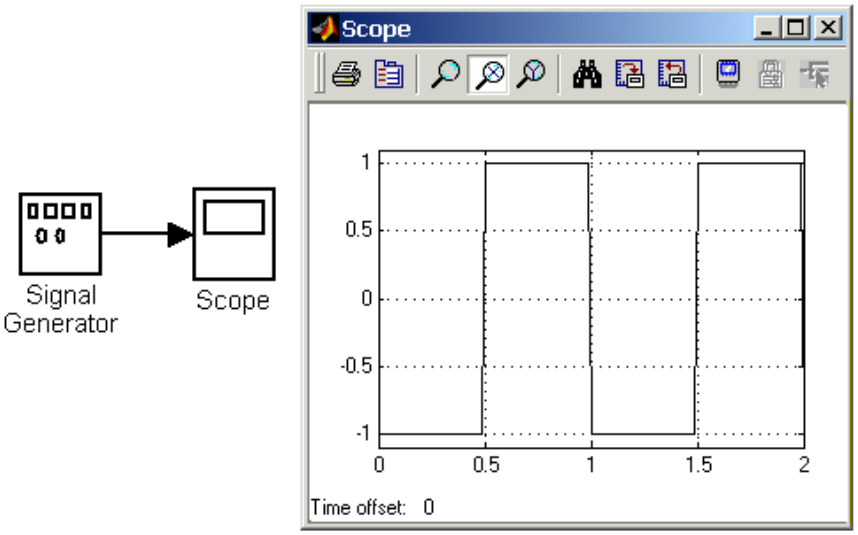
Parametrlər:

1. **Wave form** – Siqnalın tipi və ya görünüşü,
2. **Amplitude** – siqnalın amplıtudu,
3. **Frequency** - tezlik (rad/san).
4. **Units** – tezliyin ölçmə vahidi. İki qiymət ala bilər:

- Hertz - Hs.

- rad/sec – rad/san.

Şəkil 5.19-da *Signal Generator* mənbə blokundan istifadə edərək düzbucaqlı siqnalın necə generasiya edilməsi nümunəsi göstərilmişdir.



Şəkil 5.19. Siqnalları generasiya edən blok

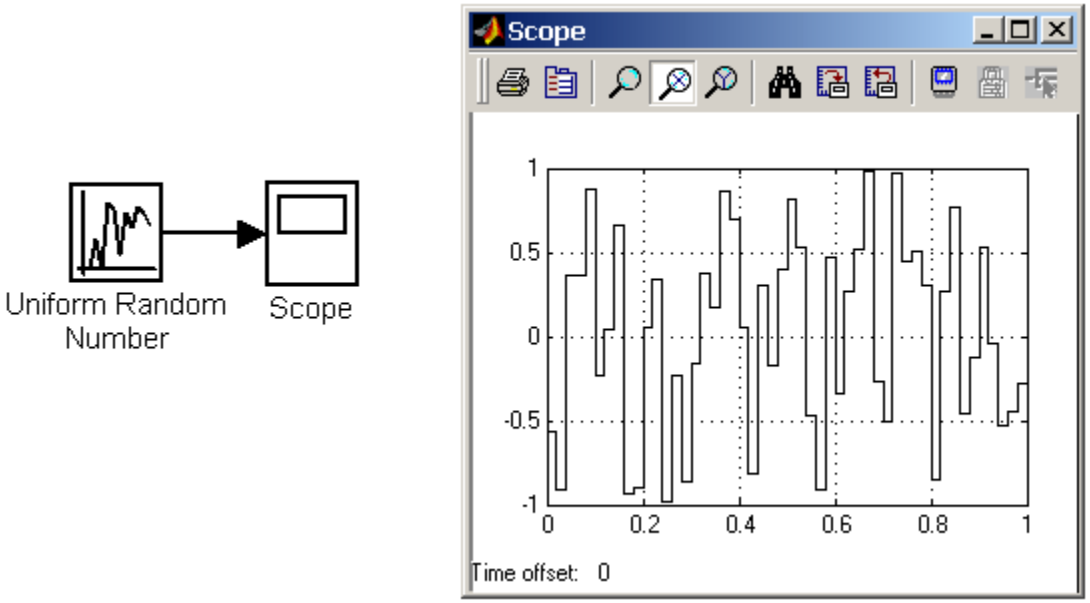
5.6.9. Bərabər paylanma qanununa malik olan təsadüfi ədədləri generasiya edən blok - Uniform Random Number

Təyinatı: Bərabər paylanma qanununa malik olan təsadüfi ədədləri formalaşdırır.

Parametrlər :

1. **Minimum** – siqnalın minimal səviyyəsi,
2. **Maximum** – siqnalın maksimal səviyyəsi,
3. **Initial seed** – başlanğıc qiymətlər.

Bu blokun çıxışında alınan siqnalın qrafiki aşağıdakı şəkil 5.20-də göstərilmişdir.



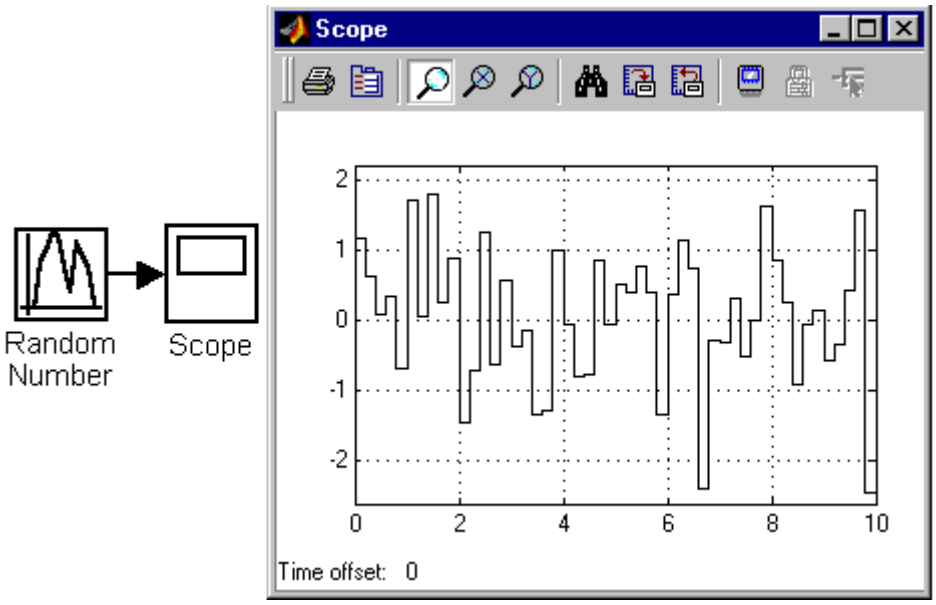
Şəkil 5.20. Bərabər paylanma qanununa malik olan təsadüfi ədədləri generasiya edən mənbə blokunun qrafiki

5.6.10. Normal paylanma qanununa malik olan təsadüfi ədədləri generasiya edən blok - Random Number

Təyinatı: Normal paylanma qanununa malik olan təsadüfi ədədləri formalaşdırır.
Parametrlər:

1. **Mean** – siqnalın orta qiyməti
2. **Variance**- dispersiya
3. **Initial seed** – başlanğıc qiymətlər.

Bu blokun çıxışında alınan siqnalın qrafiki aşağıdakı şəkil 5.21-də göstərilmişdir.



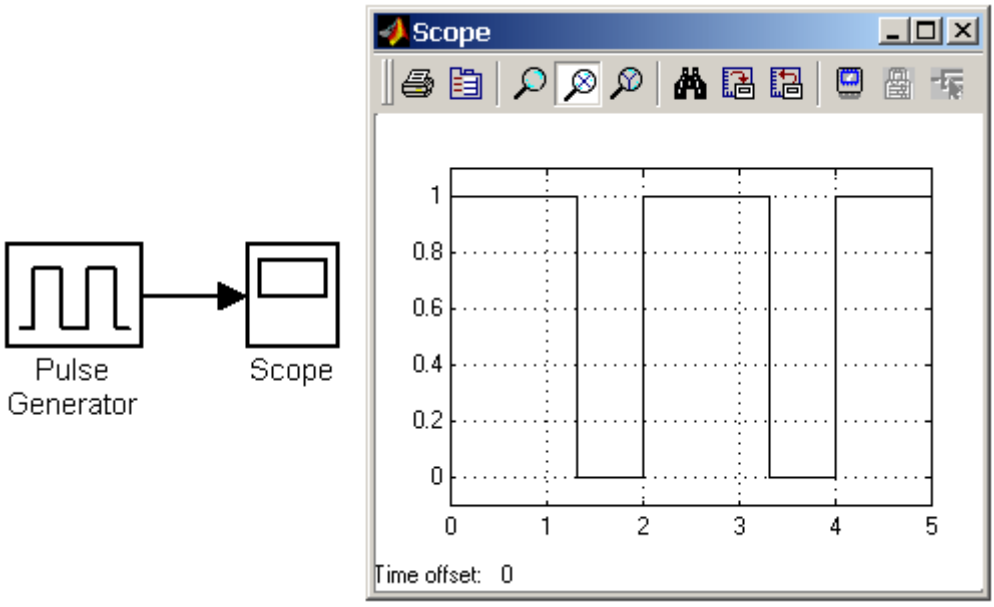
Şəkil 5.21. Normal paylanma qanununa malik olan təsadüfi ədədləri generasiya edən blokun çıxış siqnalının qrafiki

5.6.11. İmpuls tipli siqnal mənbəyi - Pulse Generator

Təyinatı: Düzbucaqlı impulsların formalaşdırılması.
Parametrləri:

1. **Pulse Type** – siqnalın formalaşdırma üsulu. İki qiymət ala bilər:
   - **Time-based** – cari zamana görə,
   - **Sample-based** – model zamanın qiyməti və hesablama addimların sayına görə;
2. **Amplitude** — Amplituda,
3. **Period** — Period. İki şəkildə verilir: **Time-based Pulse Type**- bu tip üçün saniyələr ilə verilir və ya model zaman addımı ilə (**Sample-based Pulse Type)**
4. **Pulse width** — impulsların eni. Periodun faiz nisbəti şəklində verilir (**Time-based Pulse Type** üçün**)** və ya model zamanın addımları ilə (**Sample-based Pulse Type** üçün**)**.
5. **Phase delay** — Faza gecikməsi. **Time-based Pulse Type** üçün saniyələr ilə verilir və ya model zamanın addımı ilə (**Sample-based Pulse Type).**
6. **Sample time** — model zamanının addımı. **Sample-based Pulse Type** ilə verilir.

Pulse Generator blokunun istifadə nümunəsi aşağıdakı şəkil 5.22-də verilmişdir.



Şəkil 5.22. Düzbucaqlı impulsları generasiya edən blokun çıxışı

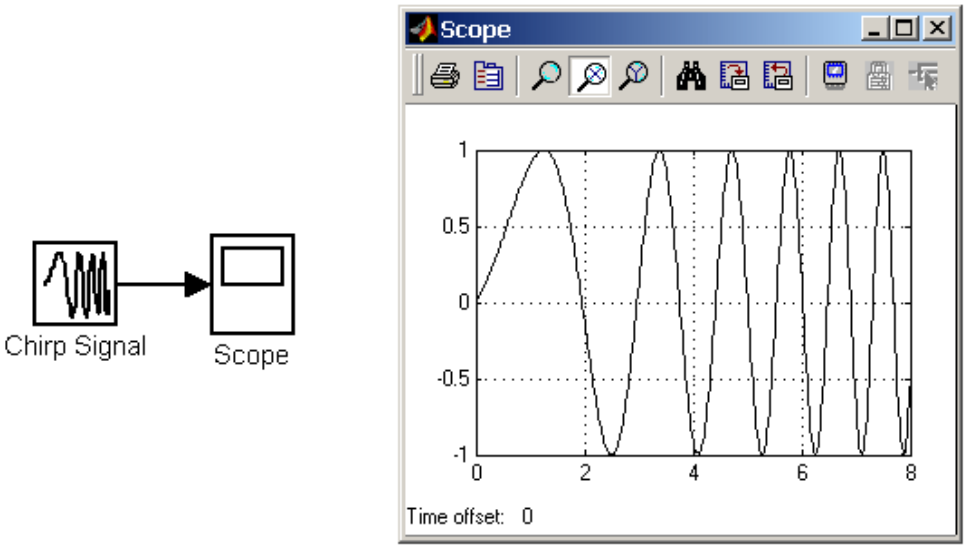
5.6.12. Tezliyi xətti dəyişən generator - Chirp Generator

Təyinatı: tezliyi xətti dəyişən siqnalları formalaşdıran blok.

Parametrləri:

1. **Initial frequence** — Başlanğıc tezlik (Hz);
2. **Target time** — tezliyin dəyişmə zamanı (san);
3. **Frequence at target time** — tezliyin son qiyməti (Hz).

Bu blokdan istifadə nümunəsini aşağıdakı şəkil 5.23-də göstərilmişdir.



Şəkil 5.23. Tezliyi xətti dəyişən generatorun çıxışı

5.6.13. Ağ küy generatoru (buraxma zolağı məhdud olan)

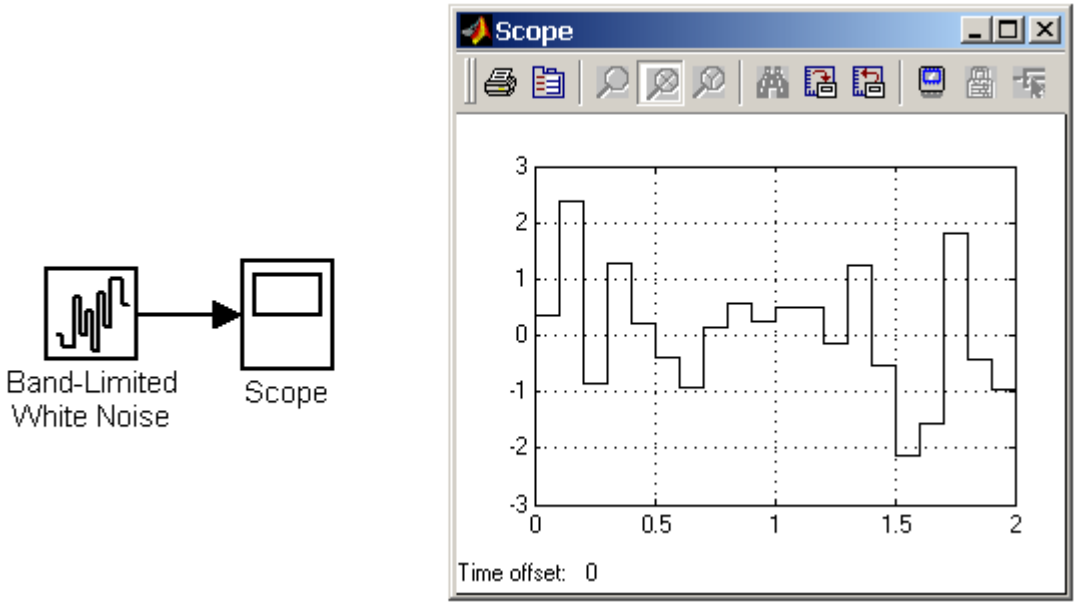
Təyinatı:

Verilmiş gücə malik olan və tərkibində bütün tezlikləri özündə saxlayan siqnal yaradır.

Parametrləri:

1. **Noise Power** – küyün gücü,
2. **Sample Time** – model zamanı,
3. **Seed** – təsadüfi ədədləri işə salmaq üçün başlanğıc ədəd.

Aşağıdakı şəkil 5.24-də bu qeneratorun işi göstərilmişdir.



Şəkil 5.24. Ağ küy generatoru

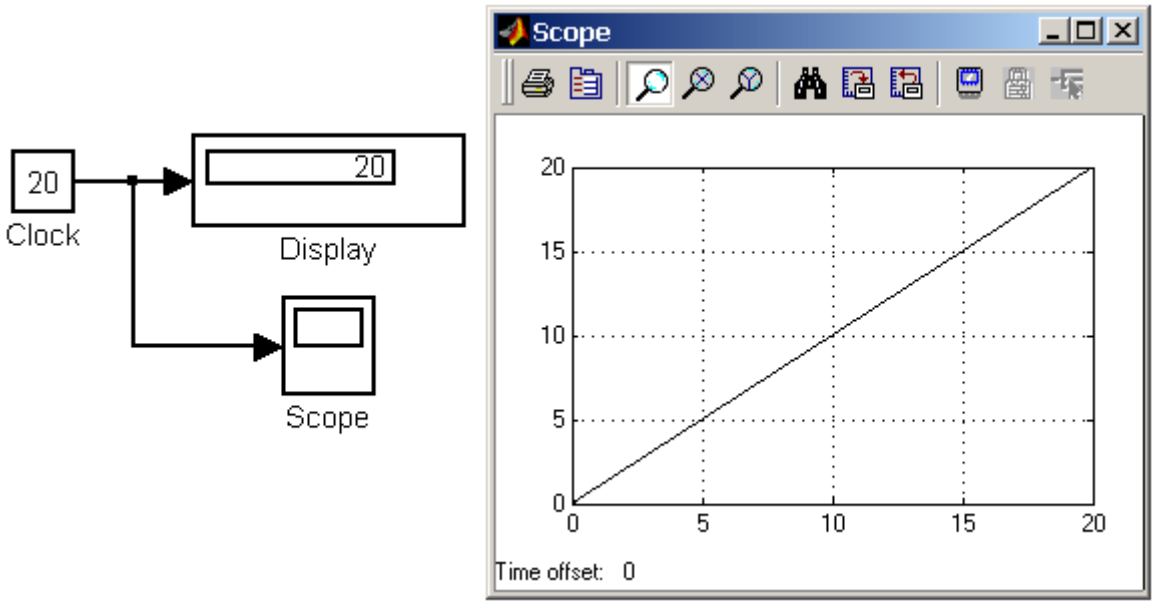
5.6.14. Zaman sıqnalı mənbəyi - Clock

Təyinatı: Elə bir sıqnal formalaşdırır ki, onun qiyməti hər addımda cari modelləşdirmə zamanına bərabərdir.

Parametrləri:

1. **Decimation** – elə addım qiymətidir ki, bü müddət ərzində mənbənin göstərişi dəyişir (əgər bu halda **Display time** parametrində bayraq qoymuş isə). Bu parametr hesablama addımının sayı kimi verilir. Məsələn, əgər modelin hesablama zamanı **Simulation parameters** dialoq pəncərəsində **0.01** san-dirsə, **Clock** blokunun **Decimation** parametri 1000-ə bərabərdirsə, onda zaman göstərişi 10 san-dən bir yeniləşəcəkdir.
2. **Display time** – mənbə blokunda zamanın təsvir olunması.

Aşağıdakı şəkil 5.25-də verilmiş mənbənin istifadə nümunəsi göstərilmişdir.



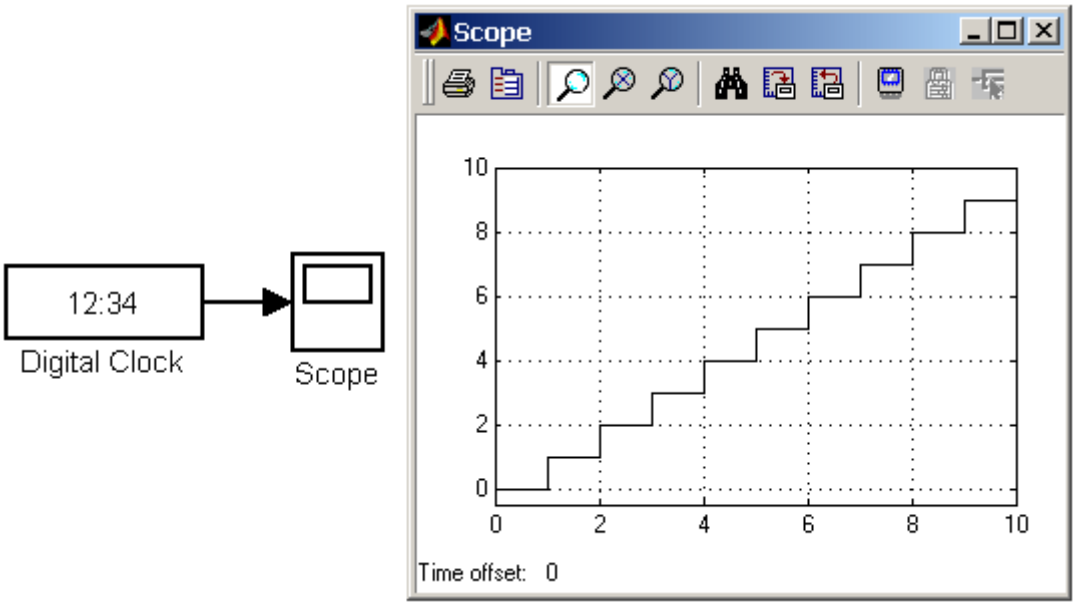
Şəkil 5.25. Zaman siqnalı mənbəyi

5.6.15. Rəqəmsal zaman mənbəyi - Digital Clock

Təyinatı: Diskret zaman siqnalını formalaşdırır.

Parametrləri: **Sample time** – modelin zaman addımı (san).

Aşağıdakı şəkil 5.26-da rəqəmsal zaman mənbəyi çıxış siqnalı göstərilmişdir- **Digital Clock**.



Şəkil 5.26. Rəqəmsal zaman mənbəyi

5.6.16. Fayldan verilənləri oxuyan blok - From File

Təyinatı: xarici fayldan verilənlərin alınması

Parametrləri:

1. **File Name** – verilənlər içində olan faylın adı,

2. **Sample time** – blokun çıxış sıqanalının dəyişmə addimi.

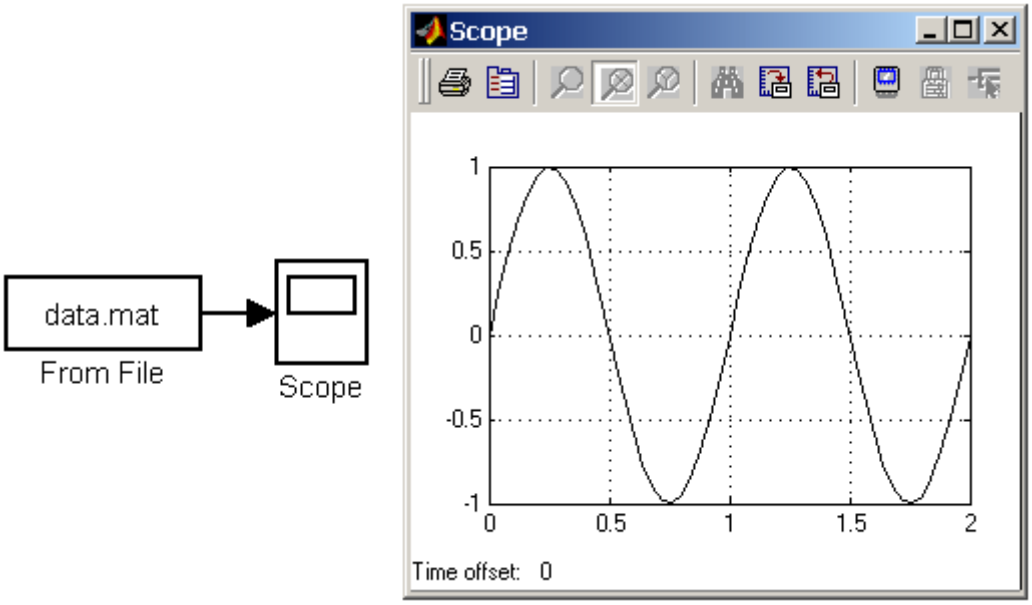
Verilənlər faylda matris şəklində təsvir olunmalıdır:

$$\begin{bmatrix} t_1 & t_2 & \dots t_{final} \\ u1_1 & u1_2 & \dots u1_{final} \\ \dots & & \\ um_1 & um_2 & \dots um_{final} \end{bmatrix}.$$

Matrisa heç olmasa minimum iki sətirdən ibarət olmalıdır. Zaman qiymətləri matrisanın birinci sətrinə yazılır, yerdə qalan sətirlərdə isə bu zamanın bu qiymətinə uyğun gələn siqnalların qiymətləri yerləşdirilir. Zaman qiymətləri artan sıra ilə yazılır. Blokun çıxış siqnalı yalnız siqnal qiymətlərindən ibarətdir, zaman qiymətləri isə orada yoxdur. Əgər cari modelin hesablama addimı verilənlər faylında olan zaman qiymətləri ilə üst-üstə düşmürsə, o zaman **Simulink** verilənlərin xətti interpolyasiyasını yerinə yetirir.

Qiymətlərin oxunduğu verilənlər faylı (**mat**-fayl) mətn tipli deyil. Faylın strukturu **MATLAB**-ın yardımçı sistemində daha geniş açıqlanmışdır.

Simulink istifadəçiləri üçün daha asan varinat **mat**-fayl yaratmaqdır. Bunun üçün **Sinks** kitabxanasının **To File** blokundan istifadə etmək lazımdır. Aşağıdakı şəkil 5.27-də bu blokun istifadə olunma nümunəsi göstərilmişdir. Burada **data.mat** faylından sinusoidal sıqnal oxunur.



Şəkil 5.27. **From File** bloku

5.6.17. İşçı oblastından verilənləri oxuyan blok - From Workspace

Təyinatı:

MATLAB-ın işçı oblastından verilənlərin oxunması.

Parametrləri:

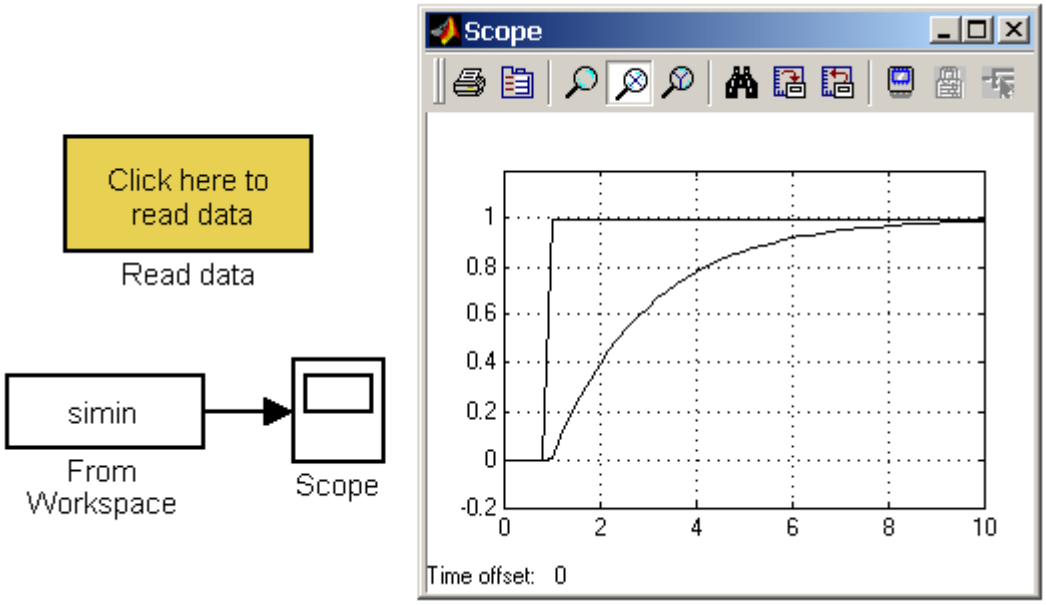
1. **Data** – verilənlərdən ibarət olan dəyişənin adı (matris və struktura) .
2. **Sample time** – blokun çıxış siqnalının dəyişmə və ya artım addımı.
3. **Interpolate data** — Verilmiş dəyişənləri ilə üst-üstə düşməyən model zaman

qiymətləri üçün verilənlərin interpolyasıyası .

4. **From output after final data value by – Data** dəyişənlərində olan verilənlər bitdikdən sonra çıxış siqnalının şəkli:

- **Extrapolate –** siqnalların xətti ekstrapolyasıyası.
- **SettingToZero –** siqnalın sıfır qiymətlərlə əvəzlənməsi.
- **HoldingFinalValue –** siqnalın çıxış qiymətləri sonuncu qiymətlərə bərabərdir.
- **CyclicRepetition –** siqnal qiymətlərinin periodik olaraq təkrar olunması. Bu variant o zaman istifadə oluna bilər ki, **Data** dəyişəni **Structure without time** formatına malikdir.

Aşağıdakı şəkil 5.28-də bu blokdan istifadə olunma nümunəsi göstərilmişdir.



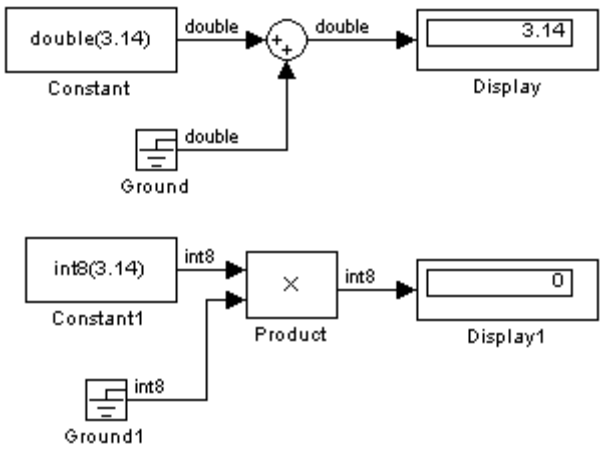
Şəkil 5.28. From Workspace blokunun istifadəsi

5.6.18. Sıfır səviyyəli siqnal bloku - Ground

Təyinatı: sıfır səviyyəli siqnalı formalaşdırır.
Parametrləri: Yoxdur.

Əgər blokun hər hansı girişi bağlanmamışdırsa, o zaman Matlab-ın əmrlər sətrində xəbərdarlayıcı mesaj ekrana çıxarılacaqdır. Bu nöqsanı aradan qaldırmaq üçün açıq qalan girişə **Ground** blokunu bağlamaq lazımdır.

Aşağıdakı şəkil 5.29-da bu blokun istifadə olunma nümunəsi göstərilmişdir. Birinci misalda **Ground** blokundan gələn siqnal cəmləyici blokunun girişlərindən birinə ikinci girişinə isə vurma blokundan daxil olur. **Display** blokunun göstərişləri **Ground** blokunun sıfır səviyyəli siqnal hasil etdiyini sübuta yetirir.



Şəkil 5.29. **Ground** blokunun tətbiq olunması

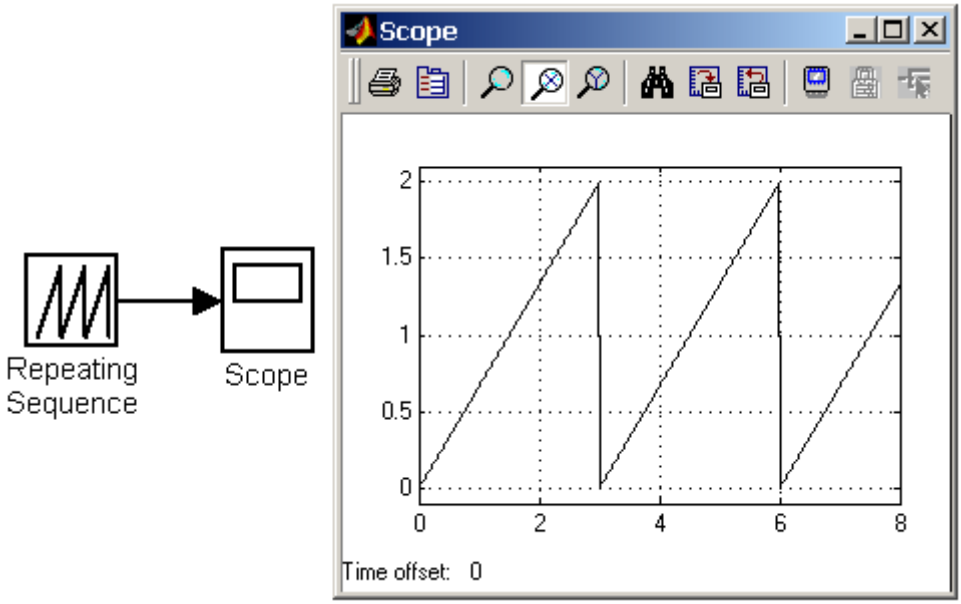
5.6.19. Periodik siqnal bloku -Repeating Sequence

Təyinatı: Periodik siqnalı formalaşdırır.

Parametrləri:

1. **Time values** – model zaman qiymətlərindən ibarət olan vektor.
2. **Output values – Time values** vektorunda verilmiş zamanın qiymətlərinə uyğun siqnal qiymətlərinin vektoru.

Bu blok, **Time values** vektorunda verilmiş zaman qiymətləri ilə uyğun üst-üstə düşməyən zaman qiymətləri üçün çıxış siqnalın xətti interpolyasiyasını yerinə yetirir. Şəkil 5.30–da mişarvari siqnalın istifadə olunma nümunəsi göstərilmişdir. Model zaman qiymətləri **[0 3]** vektoru ilə, çıxış siqnalın qiymətləri isə **[0 2]** aralığında verilmişdir.



Şəkil 5.30. Blokun istifadə olunması **Repeating Sequence**

5.6.20. Giriş port bloku - Inport

Təyinatı: Bu blok alt sistemlər və iyerarxiyanın yuxarı səviyyə modelləri üçün giriş port yaradır.

Parametrləri:

- **Port number –** portun nömrəsi.
- **Port dimensions –** giriş portunun ölçüsü. Əgər bu parametr –1 bərabərdirsə, onda giriş siqnalının ölçüsü avtomatik olaraq təyin olunacaqdır;
- **Sample time –**model zamanının addımı;
- **Data type –** giriş siqnal verilənlərinin tipləri: **auto, double, single, int8, uint8, int16, uint16, int32, uint32** və ya **boolean**.
- **Signal type –** giriş siqnal tipləri:

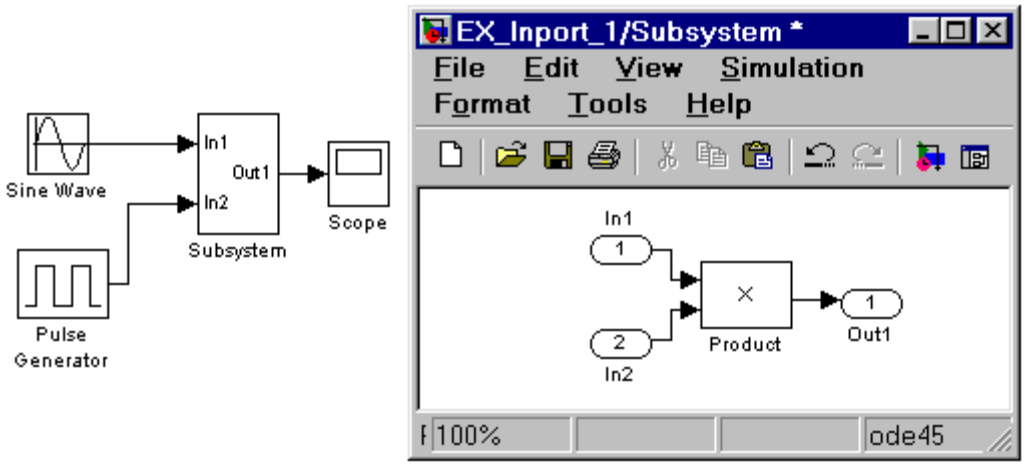
1. **auto –** tiplərin avtomatik təyini,
2. **real –** həqiqi tipli siqnal,
3. **complex-** kompleks tipli siqnal.

- **Interpolate data** (bayraq) – giriş sıqnalının interpolyasiyası. Əgər işçi fəzadan oxunan giriş siqnalının zamanlama şkalalanması model zamanlaması ilə üsüt-üstə düşmürsə, o zaman bu blok giriş siqnalının interpolyasiyasını yerinə yetirir. **Inport** blokunun istfadə edilən altsistemlərdə bu parametr istifadə edilmir.

Inport blokunun altsistemlərdə istifadə olunması

Inport bloku alt sistemin girişidir. **Inport** bloku vasitəsi ilə verilən siqnal altsistemin daxilinə ötürülür. Giriş portun adı altsistem pəncərəsində portun metkası kimi göstərilir. **Inport** blokundan istifadə edərək **Simulink-də** altsistem yaratmaq üçün aşağıdakı qaydlardan istifadə edilir:

1. **Edit/Create subsystem** əmrinin köməyi ilə altsistem yaradılarkən giriş portlar və nömrələnməsi avtomatik olraraq yerinə yetirilir. Bu nömərlənmə 1-dən başlayaraq yerinə yetirilir.
2. Əgr altsitemə yeni **Inport** blok daxil edilirsə, o zaman ona sıraya görə ona növbəti nömrə verilir.
3. Əgər **Inport** bloklarından biri ləgv edilirsə, o zaman yerdə qalan portlar yenidən adlandırlır və ya nömrələnir. Nəticədə nömrələnmə kəsilməz şəkildə olmalıdır.
4. Əgər portların nömrələnmə ardıcılığında kəsilməz deyilsə, o zaman **Simulink** proqramının yerinə yetirilməsind ə xəta əmələ gələcəkdir və duracaqdır. Bu zaman portları əl ilə yenidən adlandırtmaq lazımdır ki, nömrələrin ardıclığı bozulmasın. Aşağıdakı şəkildə bu blokdan istifadə edərək altsistem yaraılmışdır.

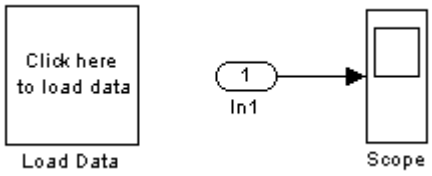


Şəkil 5.31. **Inport** Blokuunn istifadə olunması ilə altsistemin yaradılması

5.6.20.2. Inport Blokunun istifadə olunması ilə yuxarı səviyyə modellərinin yaradılması

Yuxarı səviyyə sistemində giriş portundan Matlab-ən içşi fəzadan olan siqnalı modelə öterlməsi üçün istifadə edilir. Siqnalı Matlab-ın işçi fəzasından ötrülməsi üçün modeldə nəinki giriş port bağlamaq lazımdır, **Simulation parameters…**

Для передачи сигнала из рабочего пространство **MATLAB** требуется не только установить в модели входной порт, но и выполнить установку параметров ввода на вкладке **Workspace I/O** окна диалога **Simulation parameters…**(должен быть установлен флажок для параметра **Input** и задано имя переменной, которая содержит входные данные). Тип вводимых данных: **Array** (массив), **Structure** (структура) или **Structure with time** (структура с полем "*время*") задается на этой же вкладке.
На Şəkil 5.32. показана модель, считывающая входной сигнал из рабочего пространства **MATLAB**. Подсистема **Load Data** обеспечивает ввод данных из файла в рабочую область **MATLAB**.



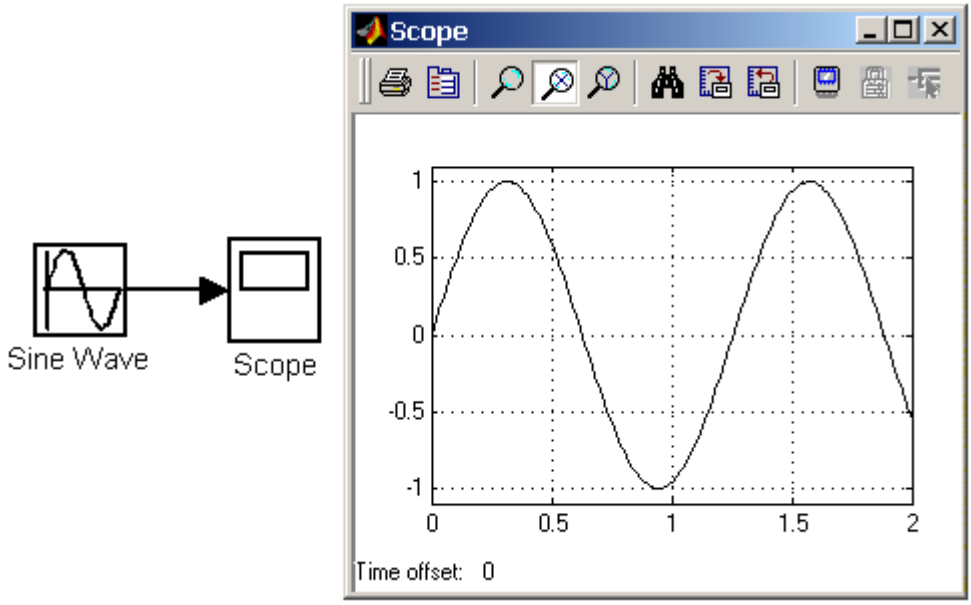
Şəkil 5.32. Модель, считывающая входной сигнал из рабочего пространства **MATLAB** с помощью блока **Input.**

5.7. Sinks – siqnal qəbul ediciləri

5.7.1. Osilloqraf- Scope

Təyinatı: Tətqiq olunan olunan siqnalların zamna görə qrafikini qurmaq üçün istifadə edilir. Modelləşdirmə müddəti ərzində sıqnalara müşaiyət etməyə imkan verir. Blok və

nəticə pəncərsinin təsviri aşağıdakı şəkildə verilmişdir.

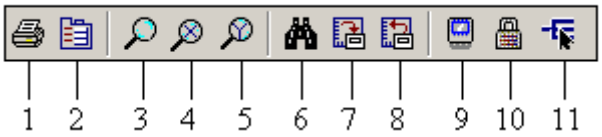


Şəkil 5.33. **Scope** osilloqrafı

Scope-da siqnalların qrafikini görmək üçün mouse-ilə blokun üzrəində iki dəfə tiklmaq lazımdır. Scope pəncərəsində yaranan qara fonu Word proqrmında istifadə edilməsi rahat olmadiğı üçün aşağıdakı **script** ilə onu figure pəncərəsinə dönüşdürə bilərsiniz.

```
%%Begin Code%%%
set(0,'ShowHiddenHandles','On')
set(gcf,'menubar','figure')
```

Scope-nın ayarlanması instrumentlər panelı vasitəsi ilə yerinə yetrilir.



Şəkil 5.34. **Scope blokunun instrumentlər paneli**

İnstrumentlər panelinin aşağıdakı düymələri vardır:

1. **Print** – Scope pəncərəsini çap etmək üçün istifadə edilir,
2. **Parameters** – parametrlərin ayarlanma pəncərəsi,
3. **Zoom** – kooordinat sisteminin hər iki oxu üzər miqyasların artırılması,
4. **Zoom X-axis** – X oxu üzrə miqyasın artırılması,
5. **Zoom Y-axis** – Y oxu üzər miqyasın artırılması,
6. **Autoscale** – hər iki ox üzrə miqyasların avtomatik olaraq ayarlanması,
7. **Save current axes settings** – pəncərnin carı ayarlanının yadda saxlanması,
8. **Restore saved axes settings** – daha öncə yadda saxlanamış pəncərə

ayarlarının daxil edilməsi,

9. **Floating scope** – scope-nın "sərbəst*" rejimə dönüşdürülməsi*.

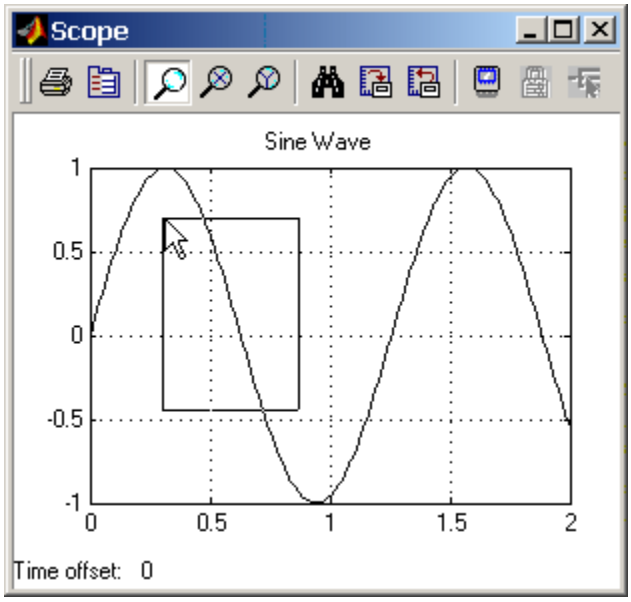
10. **Lock/Unlock axes selection** – təsvir oluanan siqnalın və pəncərinin cari koordinat sistemi arasında əlaqəni saxlamaq/qırmaq kimi işləri yerinə yetirmək olar. **Floating scope rejimi** çalışdığı müddət ərzində yerinə bunu yetrimək olar.

11. **Signal selection** – təsvir olunacaq siqnalın seçimi. **Floating** scope rejimi çalışırsa o zaman təsvir etmək üçün siqnalı seçmək imkanı olur.

Təsvir oluanan qrafiklərin miqyasını dəyişmək üçün bir neçə yolla yerinə yetirilir:

1. İnstumentlər panelinin (, və ya) uyğun düymələrin mouse-nın sol düyməsi ilə tiklamakla miqyasın 2,5 dəfəlik armasını təmmin edirlər.

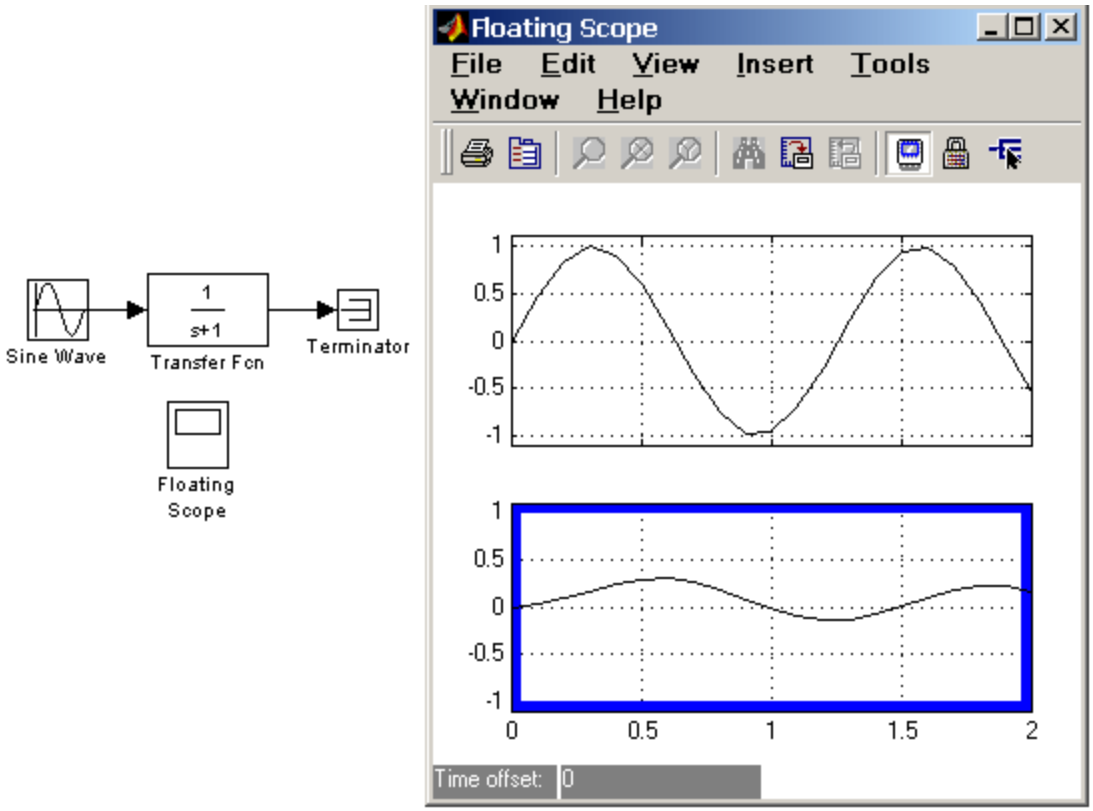
2. İnstumentlər panelinin (, və ya) uyğun düymələrin mouse-nın sol düyməsi ilə dinamiki çərçivə vasitəsi ilə sadəcə lazım olan oblastın miqyasını artırmaq olar. Bunu aşağıdakı şəkil daha aydın göstərir.



Şəkil 5.35. Qrafikin miqyasının artırılması

5.7.2. Floating Scope tipli osilloqraf

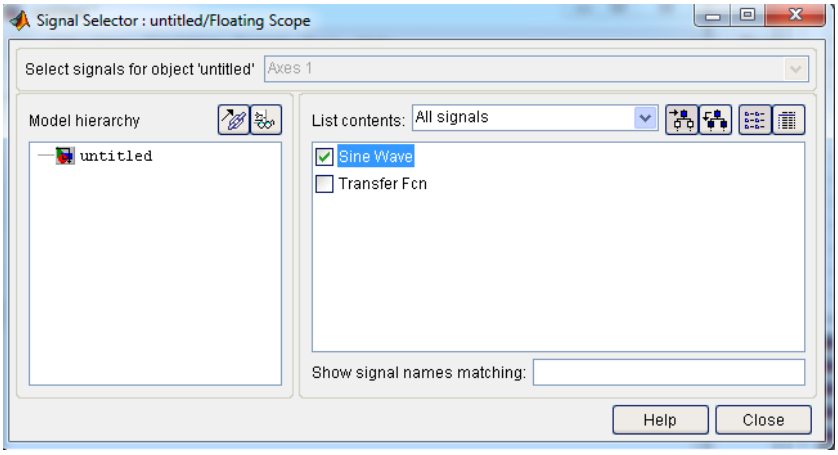
Floating Scope tipli osilloqraf sadə tipli osiloqrafa çox bəznzəyir, bu osiloqraf "sərbəst*"* rejimdə çalışir. Bu blokun istufadə olunmasına aid nümunəni aşağıdakı şəkildə görə bilərsiniz.



Şəkil 5.36. **Floating Scope** tipli osilloqrafın istifadəsinə aid nümunə

Bu rejimdə osiloqraf girişə malik deyil. Siqnalın seceilməsi instrumentlər panelində olan (**Signal selection**) istrumenti ilə yerinə yetirilir. Siqnallları seçmək üçün aşağıdakı əməliyyatları yerinə yetrimək lazımdır:

1. Qrafikin təsvir olunacağı koordinat sistemini seçmək lazımdır. Bunun üçün mouse-un sol klavişi ilə lazım olan sistemin daxilində seçmək lazimdır. Semilmiş korrdinat sisteminin perimetri mavi rənglə boyanacaqdır.
2. istrumenti ilə **Signal Selector** dialoq pəncərəsini açmalı,
3. Bayraq ilə siqnalları təsvir ediləcək blokların adlarını seçmək lazımdır. Bundan sonra **Floating Scope** blokunda seçilmiş sıqnallar təsvir olunacaqdır.



Şəkil 5.37. **Signal Selector** dialoq pəncərəsi

5.7.3. Qrafikqurucu - XY Graph

Təyinatı: hər hansı bir siqnalın başqa sıqnaldan asılığını qurur (**Y(X)-tipli**).

Parametrləri:

x-min – **X** –oxu uzrə siqnalın minimal qiyməti,

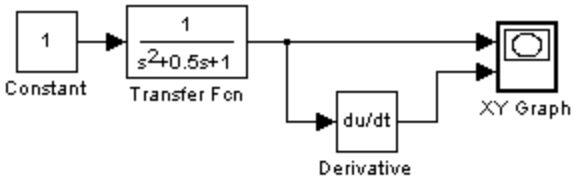
x-max – **X** oxu uzrə siqnalın maksimal qiymət,

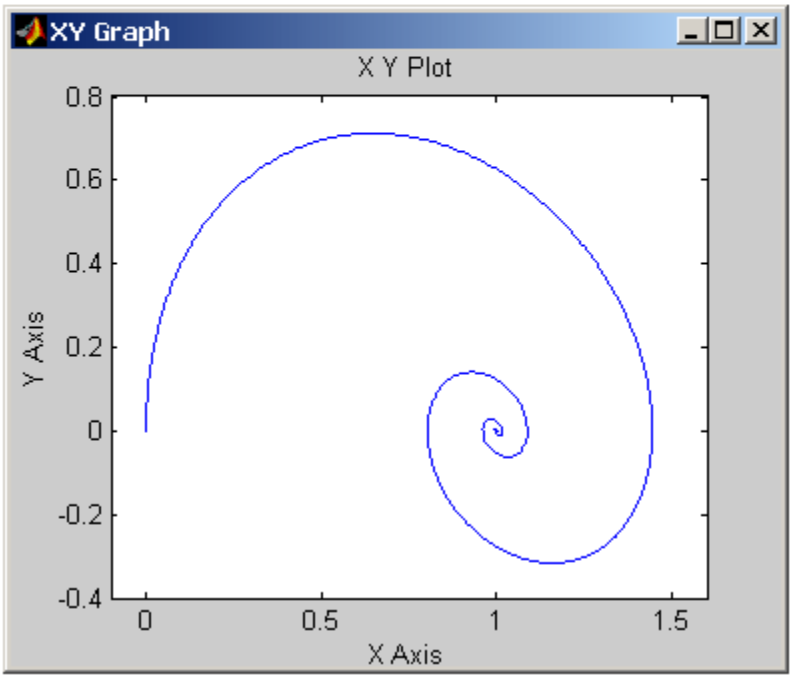
y-min – **Y** –oxu uzrə siqnalın minimal qiyməti,

y-max – **Y** –oxu uzrə siqnalın maksimal qiyməti,

Sample time – model zaman addımı,

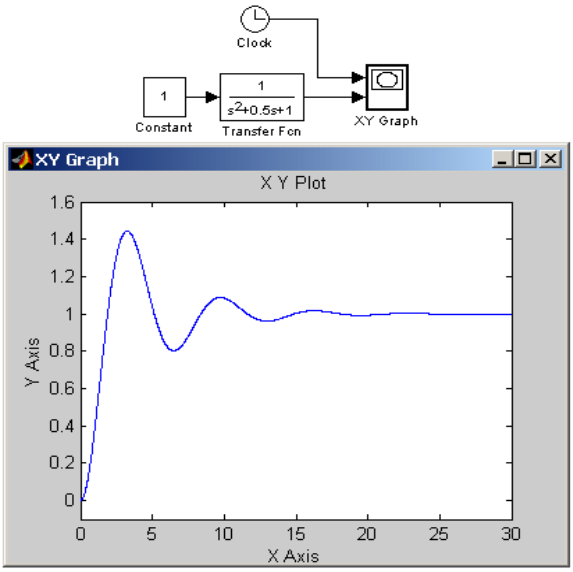
Bu blokun iki girişi vardır. Blokun yuxarı girişi **X-arqumenti**, aşağı girişinə isə **Y**-funksiya qiymətlərini verilməsi üçün stifadə edilir. Aşağıdakı şəkildə bu blokun istufadə olunmasına aid nümunə verilmişdir.





Şəkil 5.38. Qrafikqurucunun(XY Graph) istifadə olunmasına aid nümunə

Qrafikqurucusu zamandan asılı qrafiklərin qurulması üçün də istifadə oluna bilər. Bunun onun birinci girişinə zamandan asılı siqmalıən yəni **Clock** blokunun çıxışını bağlamaq lazmdır. Qrafikqurucunun bu cür istifadə olunmasına aid nümunə aşağıdakı şəkildə verilmişdir.



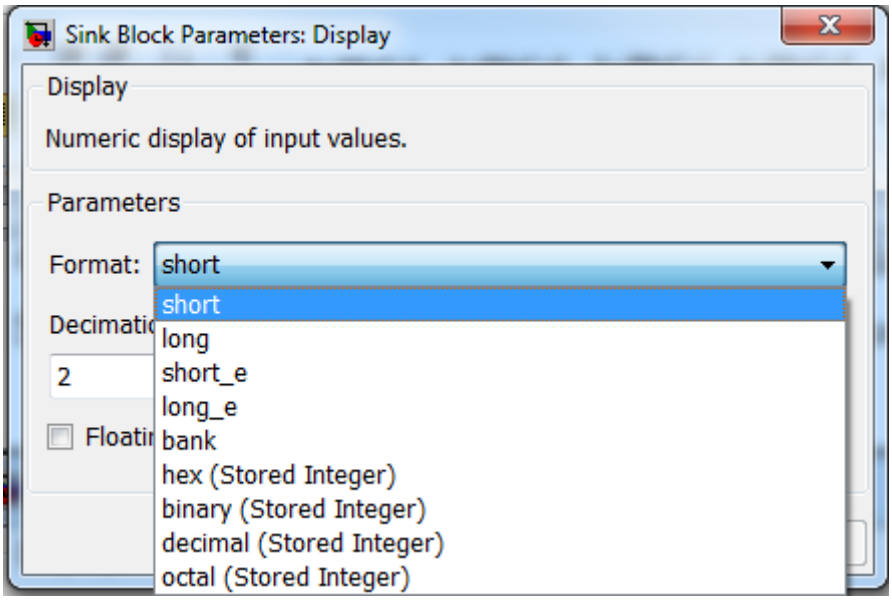
Şəkil. 5.39. **XY Graph** blokunun zamandan asılı olaraq dəyişən sqnalların təsviri üçün istifadəsi

5.7.4. Rəqəmsal indikator - Display

Təyinatı: siqnal qiymətinin rəqmsal çəkildə təsvir edilməsi.

Parametrləri:

- **Format** – verilənləri təsvir etmək üçün istifadə olaunan formatdır. **Format** parametri aşağıdakı qiymətlərdən birni ala bilər:
  1. **short** – 5 ədəd əhəmiyyətli onluq rəqəmlər.
  2. **long** – 15 ədəd əhəmiyyətli onluq rəqəmlər
  3. **short_e** –5 ədəd əhəmiyyətli onluq rəqəmlər və 3 simvol isə onluq qüvvətində
  4. **long_e** – 15 ədəd əhəmiyyətli onluq rəqəmlər və 3 isə onluq qüvvətində
  5. **bank** – "pul" formatı. Fiksə edilmiş nöqtə tipli format və ədədin kəsir hissəsində isə iki onluq qərəm.

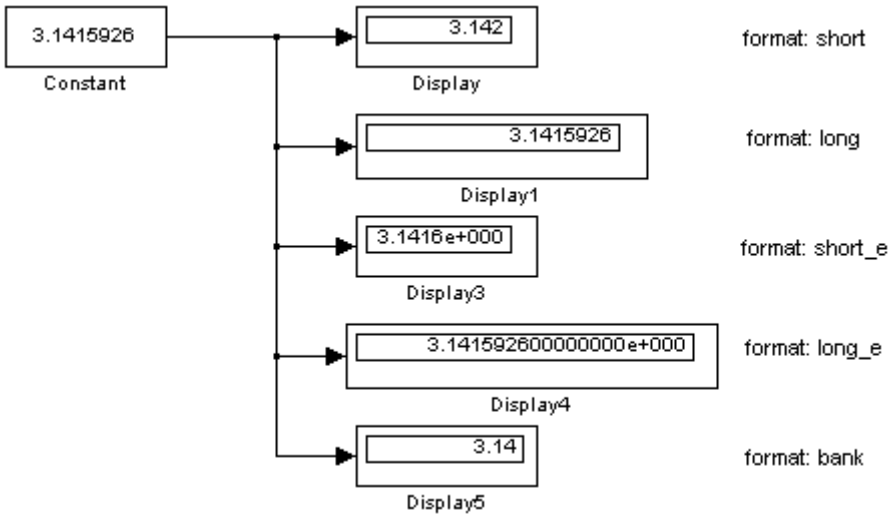


- **Decimation** – giriş siqnalnın tərtib dərcəsi.

- **Decimation = 1** olduqda giriş sıqnalən hər bir qiyməti göstərilir,
- **Decimation = 2** olduqda hər iki qiymətdən biri göstərilir,
- **Decimation = 3** – olduqda hər üç qiymətdən biri göstərilir,

- **Sample time** – model zaman addımı. Təsvir olunan verilənlərin dikskret sayını təyin edilir.
- **Floating display** (bayraq)– bloku "sərbəst" rejimə dönüşdürür. Bu rejimdə blokun giriş portu yoxdur. Siqnalın seçilməsi üçün mouse-un sol düyməsini uyğun bağlantı xəttinin üzərinə basmaqla yerinə yetrilir. Bu rejimdə parametrin hesabı üçün **Signal storage reuse**-də **off** qiyməti qouyulur (**Simulation parameters…** dialoq pəncərəsində).

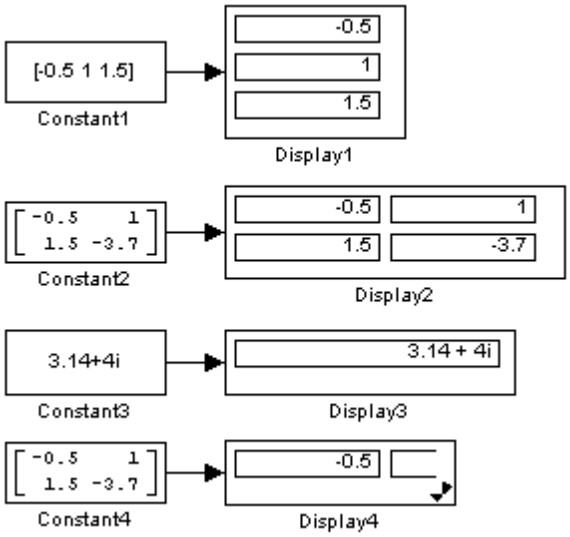
Display blokunda müxtəlif format parametrlərinin tətbiqinə aid nümunə göstərilmişdir.



Şəkil. 5.40. **Display** blokunda müxtəlif format parametrlərinin tətbiqi

Display blokundan nəinki skalyar siqnaları, həm də vektor və matris tipli siqnalların təsvir

edilməsi üçün istifadə edilir. Əgər təsvir olunan qiymətlər bloka yerləşmirsə, o zaman blokun sağ küncündə simvolu əmələ gəlir. Bu onu görsədir ki, blokun ölçüsünü artırmaq lazımdır.



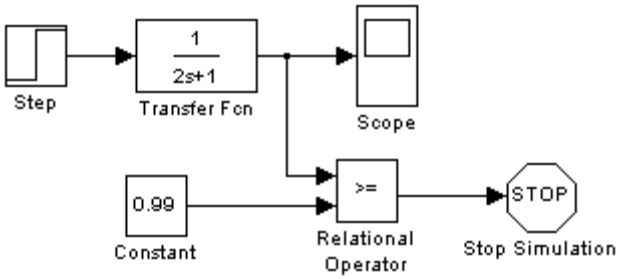
Şəkil 5.41. **Display** blokunun tətbiqinə aid misallar

5.7.5. Modelləşdirməni dayandıran blok - Stop Simulation

Təyinatı: əgər giriş siqnalı sıfıra bərabər deyilsə, o zaman hesabatın durdurulmasını təmin edilir.

Parametrləri: Yoxdur.

Simulink –in sıfır olmayan blok girişinə siqnal daxil edildikdə o cari hesbati yerinə yetirir və sonra modelləşdirməni durdurur. Əgər stop blokunun girişinə vektor tipli siqnal verilmişsə, o zaman hesabln durması üçün vektorun hech olmmasa bir elementi sifir olmalıdır. Aşağıdakı şəkildə bu blokun istifadəsinə aid misal göstərilmişdir. Bu musalda əgər çıxış siqnalının dəyəri 0.99-dən büyük və ya bərabərdirsə hesab əməli durur.



Şəkil 5.41. **Stop Simulation** blokunun istfadəsinə aid misal

5.7.6. Verilənlərin faylda yadda saxlanılması - To File

Təyinatı: blok giriçinə daxil olan verilənləri fay a yazır

Parametrləri:

- **Filename – faylın adı**. Var sayılan rejimdə faylın adı **untitled.mat** qəbul edilir. Gər fayl tam yolü göstərilməyibsə, onda fayıl carı kataloqda yadda saxlanacaqdır.

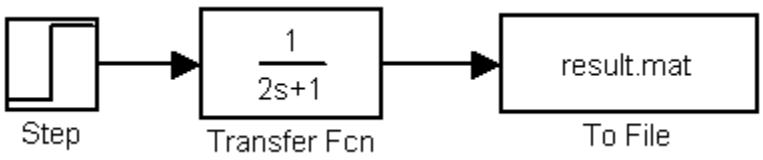
- **Variable name –** yadda saxlanılan verilənləri özündə əks etdirən dəyişənin adı.
- **Decimation** – fayla yazılan giriş sıqnalının dikkretlərinin sayı. Əgər **Decimation = 1** isə o zaman giriş siqnalının bütün qiymətləri yəni bütün diskretlərdə olan qiymətləri yazılır,
- **Əgər Decimation = 2** hər iki qiymətdən biri yazılır,
- **Əgər Decimation = 3** – hər üç qiymətdən biri,
- **Sample time** – moidel zaman addımı. Bu parametr verilənlərin yazılma diskterliyini təyin edir.

Verilənlər fayılda matris şəklində yadda saxlanırlar:

$$\begin{bmatrix} t_1 & t_2 & \dots t_{final} \\ u1_1 & u1_2 & \dots u1_{final} \\ \dots & & \\ um_1 & um_2 & \dots um_{final} \end{bmatrix}.$$

Zaman qiymətləri matrisin birinci sətrində yazılır vı yerdı qalan sətirlərdə isə bu zaman qiymətlərinə uyğun gələn siqnalın qiymətləridir.(**mat**-faylı) tipli verilənlər faylı mətin tipli deyillər. Bu faylın strukturu **MATLAB** yardımçı sistemində daha ətraflı verilmişdir. Пользователям **Simulink** istifadəçilərinə удобнее всего считывать данные из **mat**-faylından məlumatların oxunması daha asandır. Bunun üçün **From File** blokundan (**Sources kitabxanası**) istifadə etmək lazmdır.

Aşağıdakı şəkildə bu blokun istifadə olunma nümunəsi göstərilmişdir. Hesablama nəticələri **result.mat faylında** göstərilmişdir.



Şəkil 5.42. **To File** blokunun tətbiq olunması

5.7.7. Verilənlərin işçı oblastda və ya fəzada yadda saxlayan blok - To Workspace

Təyinatı: bu blok onun girişinə daxil olan verilənləri Matlab-ın içşi fəzasında yadda saxlayır.

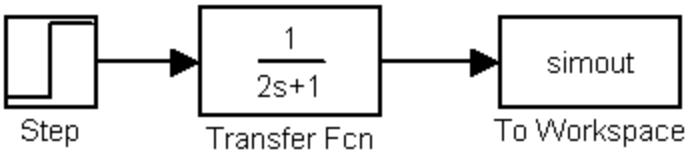
Parametrləri:

- **Variable name –** yadda saxlanan dəyişənləri özündə əks etdirən dəyişənənin adı
- **Limit data points to last** – zamana hesablanan maksimal nöqtələrin sayı (hesablama müddəti molleləmənin bitmə zamanından hesablanır). Əgər **Limit data points to last** parametrinin qiyməti inf kimi verilmiş isə o zaman işçı oblastada bütün qiymətlər yadda saxlanacaqdır.
- **Decimation** – işçı oblasta yazılna verilənlərin diskret sayı
- **Sample time** – model zaman addımı. Fayla yazılana verilənlərin dikretliyini təyin edir
- **Save format** – verilənlərin yadda saxlanma formatı. Aşağıdakı qiymətləri ala

bilər:

1. **Matrix** – matris. Verilənlər massiv kimi yadda saxlanır, sətirlərin sayı zamana görə hesablanan nöqtələri ilə təyin edilir, sütünlar isə blokun girişinə daxil olan siqnalın ölmüsü ilə təyi edilir. Əgər girtişə skalyar siqnal verilirsə, o zaman matris yalnız bir sütündan ibarət olacaqdır.
2. **Structure** – struktura. Verilənlər struktura şəklində yadda saxlanırlar, onun üç sahəsi vardır: **time – zaman**, **signals** – yadda saxlanılan siqnalın qiymətləri, **blockName** – modlein adı və **To Workspace bloku adı**. verilmiş formata görə time sahəsi doldurulmur.
3. **Structure with Time** – əlavə sahələrə sahib olan struktura. Verilmiş formata görə **time** sahəsi zaman qiymətləri ilə doldurulmuş olur.

Aşağıdakı şəkildə bu blokun istifadə olunmasına aid nümunə göstərilmişdir. Hesablama nəticələri **simout** dəyişənində yadda saxlanır. İşçi fəzasından verilənləri oxumaq üçün **From Workspace** bolokundan (Sources kitabxana).



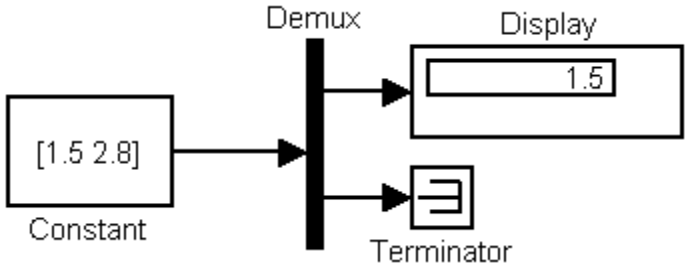
Şəkil 5.43. **To Workspace** blokunun istifadəsinə aid nümunə

5.7.8. Sonlandırıcı qəbuledici- Terminator

Təyinatı: bir başqa blokun istifadə olunmayan çıxışlarına siqnalın verilməsi üçün istifadə edilir.

Parametrləri: *yoxdur.*

Əgər hər hansı bir səbəbə görə blokun çıxışı başqa blokun giirşinə bağlı deyilsə, o zaman **Simulink MATLAB-ın** əmrlər sətrində xəta içərən mesaj vermeyə başlar. Bu xətaları önləmək üçün **Terminator** blokundan istifadə etmək lazımdır. Aşağıdakı şəkildə belə sonlandırcı qəbuledici blokunun ustfadəsinə aid nümunə göstərilmişdir. **Demux bloku** vasitəsi ilə **Constant** blokundan əldə edilən ikinci element hech bir yerdə istifadə olunmadığı üçün**,** **Terminator** blokunun girişinə verilir.



Şəkil 5.44. **Terminator** blokunun tətbiqinə aid misal

5.7.9. Çıxış port bloku - Outport

Təyinatı: ieraxiyanin yuxarı səviyyə modelləri və ya alt sistemlər üçün çıxış portu yaradır.

Parametrləri:

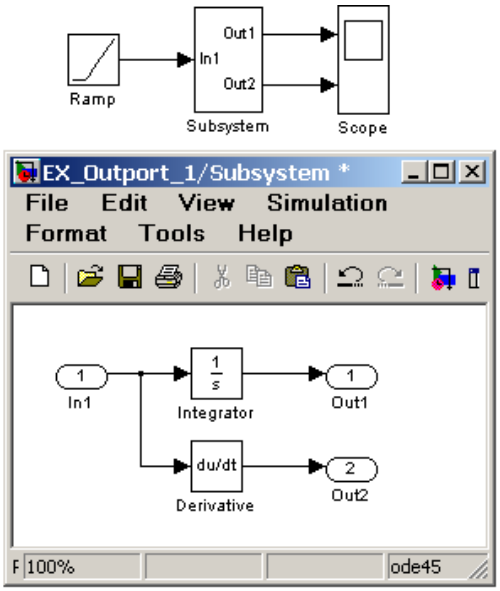
- **Port number** – portun nömərsi.
- **Output when disabled** – əgər altsistem bağlı isə o zaman altsistemin çıxıçında alınan siqnalın şəkli. Bu aşağıdakı qiymətləri ala bilər:
  1. **held** – altsistemin çıxış siqnalı axırıncı hesablanmış qiymətə bərabər götürülür.
  2. **reset** – – altsistemin çıxış siqnalı Initial output vasitəsi ilə verilən qiymətə bərabər götürülür.
- **Initial output** - əgər altsistem bağlı isə o zaman altsistemin çıxış siqnalı onun çalışmadan əvvəl qiymətlərə bərabər götürülür.
-

5.7.9.1. Outport Blokunun istifadə olunmasına aid nümunələr

Altsistemlərin **Outport** blokları onun çıxışları hesab edilir. **A**ltsisteminin içində yerləşən **Outport** bloku ona verilən siqnalı modelə və ya daha yuxarı səviyyədə yerləşən altsistemə ötürür. Çıxış portunun adı altsistemin qrafiki təsvirində metka və ya nişanlanmış işarə kimi göstəriləckdir.

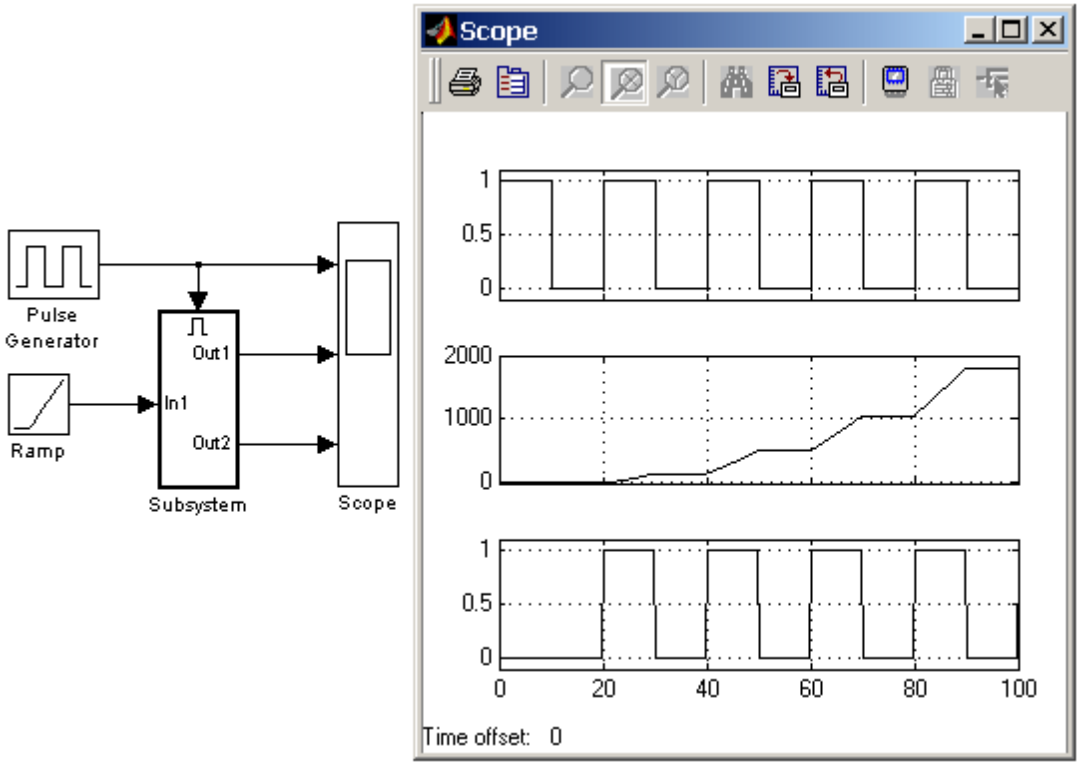
Altsistemlərin yaradılması və ora **Outport** blokunun yaraılması üçün Simulink-də aşağıdakı qaydalardan istifadə edilir:

1. Edit/Create subsystem əmrindən itifadə olunaraq altsistem yaradılarkən 1-dən başlayaraq nömrələnir.
2. Əgər altsistemə yeni bir Outport bloku daxil edilirsə, onda ona avtomatik olraq birsonrakı nömrə daxil edilir.
3. Əgər **Outport bloku** ləğv edilirsə o zaman portlar eləə nömrələnir ki, portun nömrələrinin ardıcılığı pozulmur.
4. Əgər portlrın nömrələnməsi kəsintisiz deyillərsə o zaman **Simulink** –də modelləşdirmə zamanı xəta içərən mesajlar ekrana çıxacaqdır. Bu kimi hadislərin meydana gəlməməsi üçün zəruri olan hallarda nömrələnməni əl ilə yerinə yetirmək lazmdır. Aşağıdakı şəkildə bu blokun istifadəsinə aid nümunə göstərilmişdir



Şəkil 5.45

Əgər altsistem idarəolunandırsa, altsistem blok edildiyi halda zaman intervalları üçün çıxış portu üçün çıxış sıqnalının şəklini verir. Aşağıdakı şəkildə **Outport Blokunun** idarə olunan altsistemdə istifadəsi göstərilmişdir. Altsistemin birinci çıxış portu üçün Output when disabled parametri held kimi verilib, ikinci halda reset kimi(başlanğıc qiymət olaraq sıfır verilmişdir),

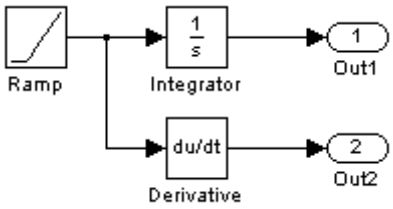


Şəkil 5.46.

Yuxarı səviyyəli modeldə Outport blokunun istifadə olunması

Yüksək səviyyə sistemlərində istifadə olunan çıxış port iki halda istifadə oluna bilərlər:

1. Matlab-in işçi fəzasına sıqnaları ötürçək üçün.
2. Modelləri analiz etmək üçün onun çıxışları ilə modelləri analzi etmə funksiyları ilə əlaqə yaradır.



Şəkil 5.47

5.8. Math – riyazi əməliyyatlar bloku

5.8.1. Cəmi hesablayan blok - Sum

Təyinatı: siqnalların qiymətlərini cəmləyən blok

Parametrlər

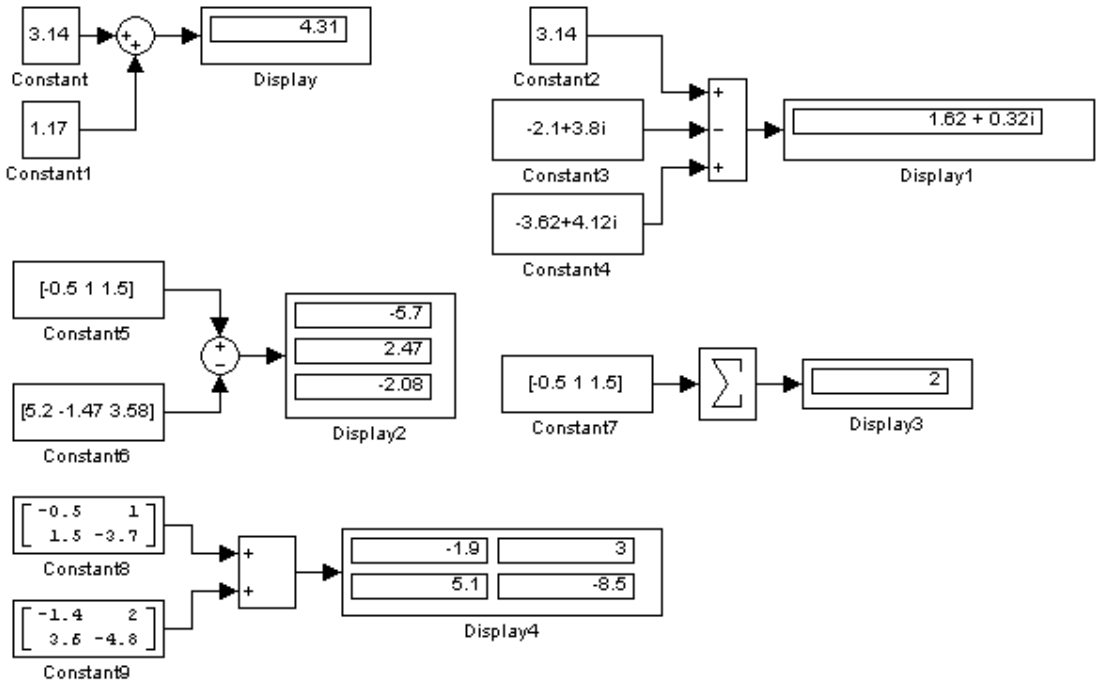
1. **Icon shape – blokun forması**. Siyahıdan seçilir.
   - **round –** çevrə,
   - **rectangular –** düzbucaqlı.
2. **List of sign** – işarə ardıcıllığı. Ardıcıllıqda aşağıdakı işarələri istifadə etmək olar:

+ (toplama), - (çıxma) və | (işarələri ayrılması).

Saturate on integer overflow (bayraq) – tam tipli siqnalın dolamasına qarşı önləm alınması. Bu bayraq qoyulduğu zaman tam tipli siqnalın məhdudlaşdırılması xətasız aparılır.

3. Girişlərin sayı və əməliyyatın tipi (toplama və ya çıxma) işarələrdən ibarət olan ardıcıllığın köməyi ilə təyin edilir (**List of sign).** Bu zaman blokun girişində qoyulan işarələrə uyğun dəyişir. **List of sign** parametrində eyni zamanda blokun giriş saylarını göstərmək olar. Blokun bütün girişləri cəmləyici də ola bilər.

Əgər blokun girişlərin sayı üçdən çoxdursa, o zaman **Sum** blokunu düzbucaqlı şəklində də istifadə edə bilərsiniz. Bu blokdan skalyar, vektor və ya matris tipli siqnalları cəmləmək üçün istifadə etmək olar. Cəmlənən siqnalların tipləri üst-üstə düşməlidir. Eyni bir cəmləyici bloka tam və həqiqi tipə malik olan siqnalı vermək olmaz. Əgər blokun giriş sayları birdən çoxdursa, onda blok vektor və ya matris siqnallar üzərində element-element əməliyyatları yerinə yetirəcəkdir. Eyni zamanda onu da qeyd etmək lazimdır ki, vektor və matrisin elementləri üst-üstə düşməlidirlər. Əgər işarələr ardıcıllığında **1** ədədini göstərsək, onda bloku vektor elementlərinin cəmini tapmaq üçün istifadə etmək olar. Bu blokun istifadə olunmasına aid misallar aşağıda şəkil 5.48-də verişmişdir.



Şəkil 5.48. **Sum** blokunun istifadəsinə aid nümunələr

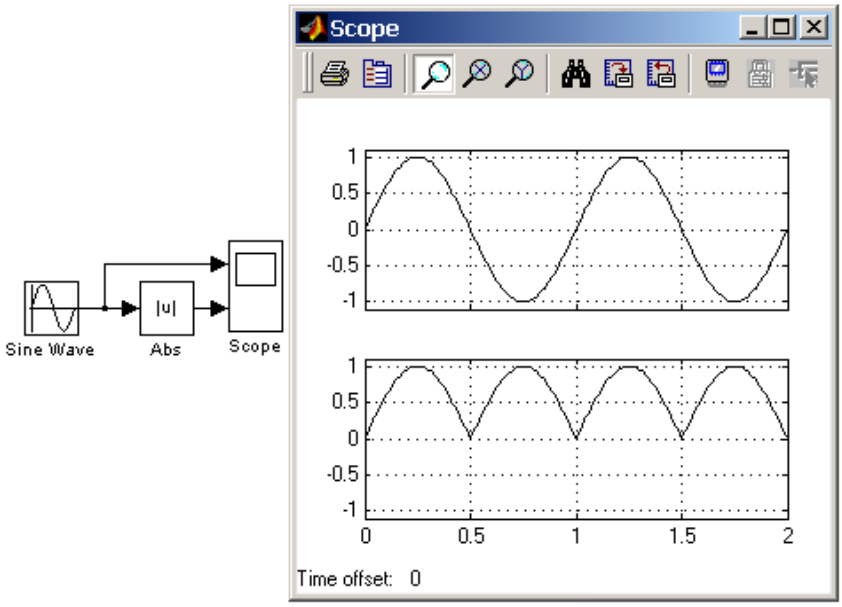
5.8.2. Mütləq qiymətinin hesablama bloku - Abs

Təyinatı: Siqnal qiymətlərinin mutləq qiymətini hesablayır.

Parametrlər:

- **Saturate on integer overflow** (bayraq) – tam ədədlərin dolmasını aradan qaldırmaq. Bayraq qoyulduqda tam ədədlərin dolmasına qoyulmuş məhdudiyyət yerinə yetirilir.

Aşağıdakı şəkil 5.49-da **Abs** blokunun istifadə olunma nümunəsi göstərilmişdir.

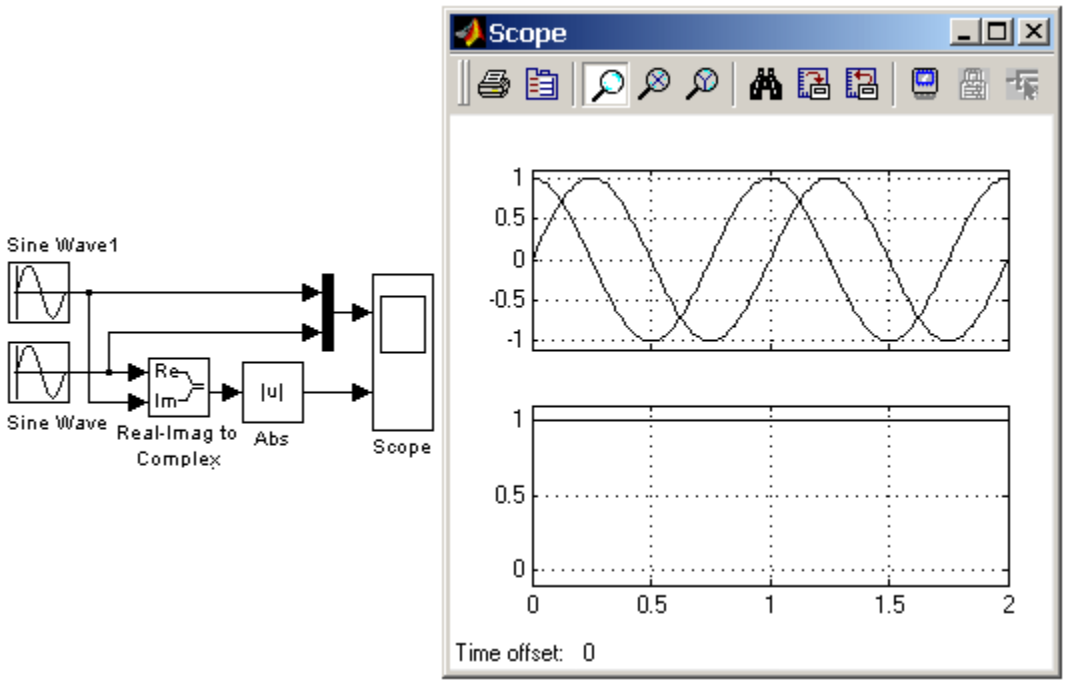


Şəkil 5.49. **Abs** blokunun istifadə olunma nümunəsi

Abs blokundan kompleks tipli siqnalın modulunu hesablamaq üçün istifadə etmək olar. Bu məqsəd ilə şəkil 5.50–də belə bir nümunə göstərilmişdir:

$$u = \cos(\omega t) + i \cdot \sin(\omega t)$$

Belə bir siqnalın modulu istənilən zamanın anı üçün **1** -ə bərabərdir.



Şəkil 5.50. Kompleks tipli siqnalın mütləq qiymətinin hesablanması(**Abs)**

5.8.3. Vurma bloku - Product

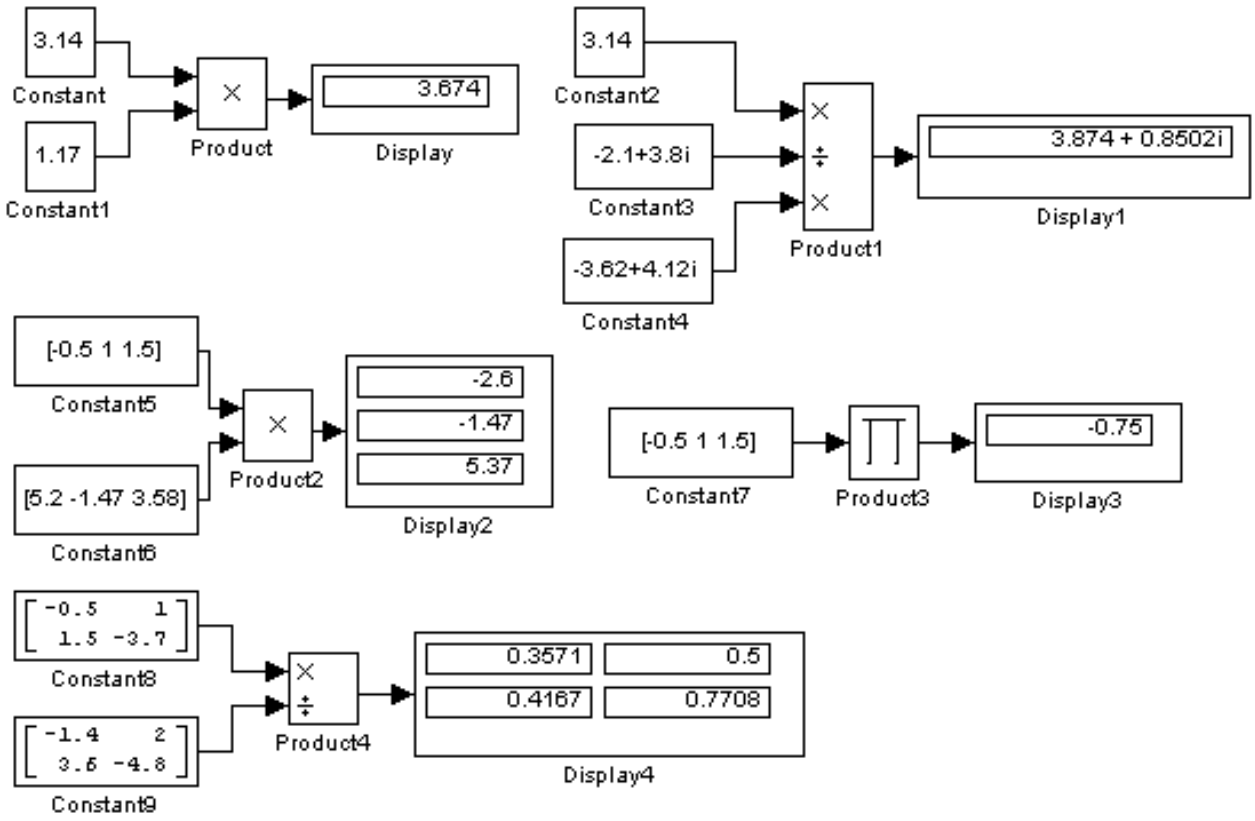
Təyinatı: siqnalların cari qiymətlərini bir-birinə vurur

Parametrlər

1. **Number of inputs** – girişlərin sayı. Ədəd kimi və ya işarələr ardıcıllığı kimi verilə bilər. Bu ardıcıllıqda *** (vurma)** və / (bölmə) işarələri ola bilər.
2. **Multiplication** – Əməliyyatın yerinə yetirmə üsulları. Aşağıdakı qiymətlərdən birini ala bilər:
   - **Element-wise** – element-element
   - **Matrix** – matris tipli
3. **Saturate on integer overflow** (bayraq) – tam tipli siqnalın dolamasına qarşı önləm alınması. Bu bayraq qoyulduğu zaman tam tipli siqnalın məhdudlaşdırılması xətasız aparılır.

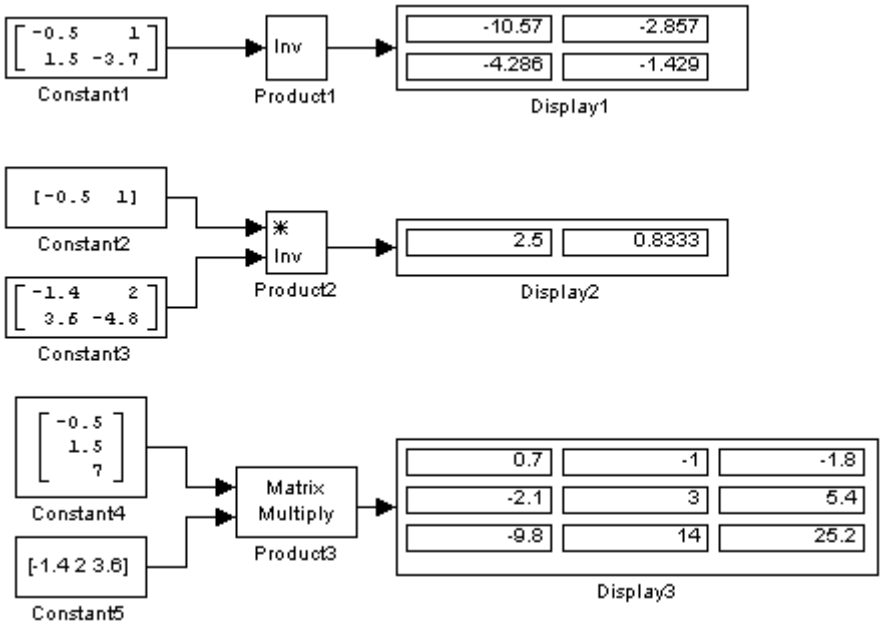
Əgər **Number of inputs** parametri siyahı şəklində verilmişsə, onda uyğun riyazi əməlin simvolik işarəsi blokun girişində göstəriləcəkdir. Blok skalyar, vektor və matris tipli siqnalların vurulması kimi əməliyyatları yerinə yetirir. Blokun giriş siqnalları üst-üstə düşməlidir. Əgər giriş saylarını **1** olaraq kimi göstərsəniz, o zaman blok vektor

elementlərinin vurulmasını yerinə yetirə bilər. Bu blokun istifadə olunmasına aid nümunə aşağıdakı şəkil 5.51-də göstərilmişdir.



Şəkil 5.51. **Product** blokunun skalyar və element-element əməllərinin yerinə yetirilməsinə aid nümunələr

Matris tipli əməliyyatların yerinə yetirilməsində müəyyən qaydaların yerinə yetirilməsi lazımdır. Məsələn, iki matrisin bir-birinə vurulması üçün birinci matrisin sətirlərinin sayı ikinci matrisin sütunlarının sayına bərabər olmalıdır. **Product** blokunun istifadəsinə aid nümunələr aşağıdakı şəkil 5.52-də göstərilmişdir. Aşağıdakı sxemlər tərs matrisin yaradılması, matrisin bölünməsi və matrislərin vurulması kimi əməliyyatları yerinə yetirirlər.



Şəkil 5.52. **Product** blokunun matrislər üzərində əməliyyatları yerinə yetirilməsinə aid nümunələr

5.8.4. Siqnalın işarəsini təyin edən blok - Sign

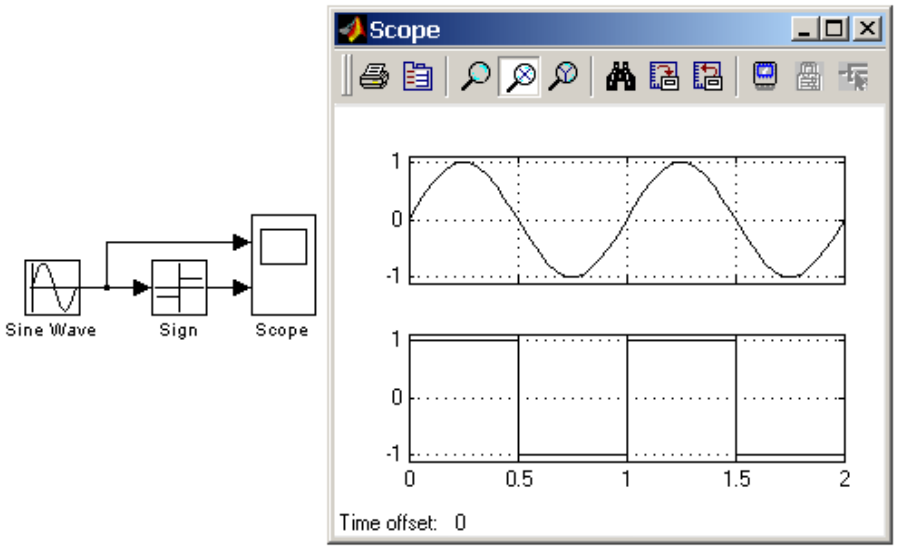
Təyinatı: giriş siqnalının işarəsini təyin edir

Parametrlər: yoxdur.

Blok aşağıdakı alqoritm əsasında işləyir:

- Əgər giriş siqnal müsbətdirsə, onda çıxış sıqnal 1-ə bərabər olacaqdır
- Əgər giriş siqnal mənfidirsə, onda çıxış sıqnal **-1**-ə bərabər olacaqdır
- Əgər giriş siqnal sıfira bərabərdirsə, onda çıxış sıqnal **0**-a bərabər olacaqdır

Aşağıdakı şəkil 5.53-də **Sign** blokunun istifadə olunmasına aid misal göstərilmişdir.



Şəkil 5.53. **Sign** blokunun istifadə olunma nümunəsi

5.8.5. Gücləndirmə bloku (Gain və Matrix Gain)

Təyinatı: Giriş siqnalının sabit əmsalına vurulmasını yerinə yetirir.

Parametrlər

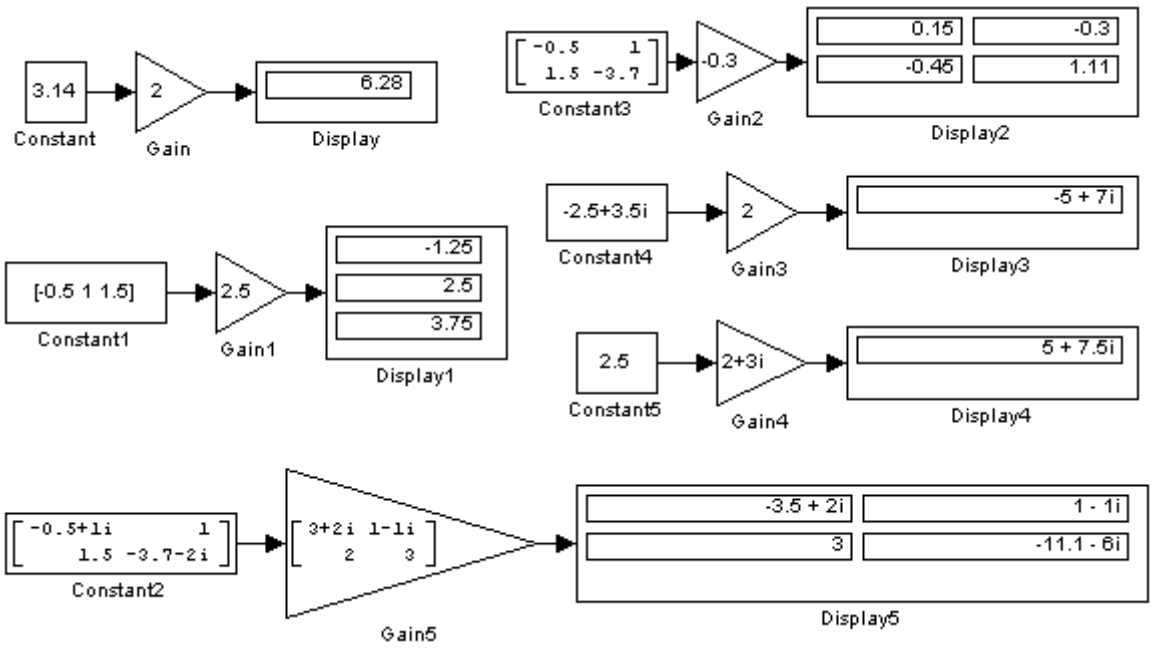
1. **Gain** – Gücləndirmə əmsalı
2. **Multiplication** – əməliyyatın yerinə yetirmə üsulu. Aşağıdakı siyahıdan qiymətləri ala bilər:
   - **Element-wise K*u**– element-element vurma
   - **Matrix K*u** – matris tipli vurma. Gücləndirmə əmsalı sol tərəfli operanddır.
   - **Matrix u*K** – Gücləndirmə əmsalı sağ tərəfli operanddır.
3. **Saturate on integer overflow** (bayraq) – tam tipli siqnalın dolmasının qabağını alır. Bu bayraq qoyulduğu zaman tam tipli siqnalların məhdudlaşması və ya dolması daha korrekt və ya xətasız yerinə yetirilir.

Gain və **Matrix Gain** gücləndirmə blokları, demək olar ki, eyni bloklardır. Fərq **Multiplication** vurma parametrindən asılıdır. **Gain** blokunun parametri 1-dən kiçik və böyük, mənfi və ya müsbət ola bilər. Gücləndirmə əmsalını skalyar, matris, vektor və riyazı ifadə şəklində verilə bilər.

Əgər **Multiplication** parametri **Element-wise K*u** kimi verilmişsə, o zaman verilmiş gücləndirmə əmsalını skalyar siqnala və ya vektor siqnalının hər bir elementinə vurma əməliyyatı kimi yerinə yetirilir. Əks halda bu blok gücləndirmə əmsallarından ibarət olan matrisi, sıqnala matris kimi vurur. Var sayılan rejimdə (default) Gücləndirmə əmsalı **double** tipli həqiqi ədəd kimi götürülür. Element-element gücləndirmə əməliyyatını yerinə yetirmək üçün giriş siqnal skalyar, vektor və matris tipli ola bilər (məntiq tipli ola bilməz). Vektorun elementləri eyni tipli siqnal olamalıdırlar. Çıxış siqnalı giriş sıqnalı ilə eyni tipdə olmalıdır. **Gain** blokunun parametri skalyar, vektor və ya matris ola bilər. Məntiqi tipdən başqa istənilən tip ola bilər. Çıxış sıqnalını hesablayan zaman **Gain** bloku aşağıdakı qaydalara əməl edir:

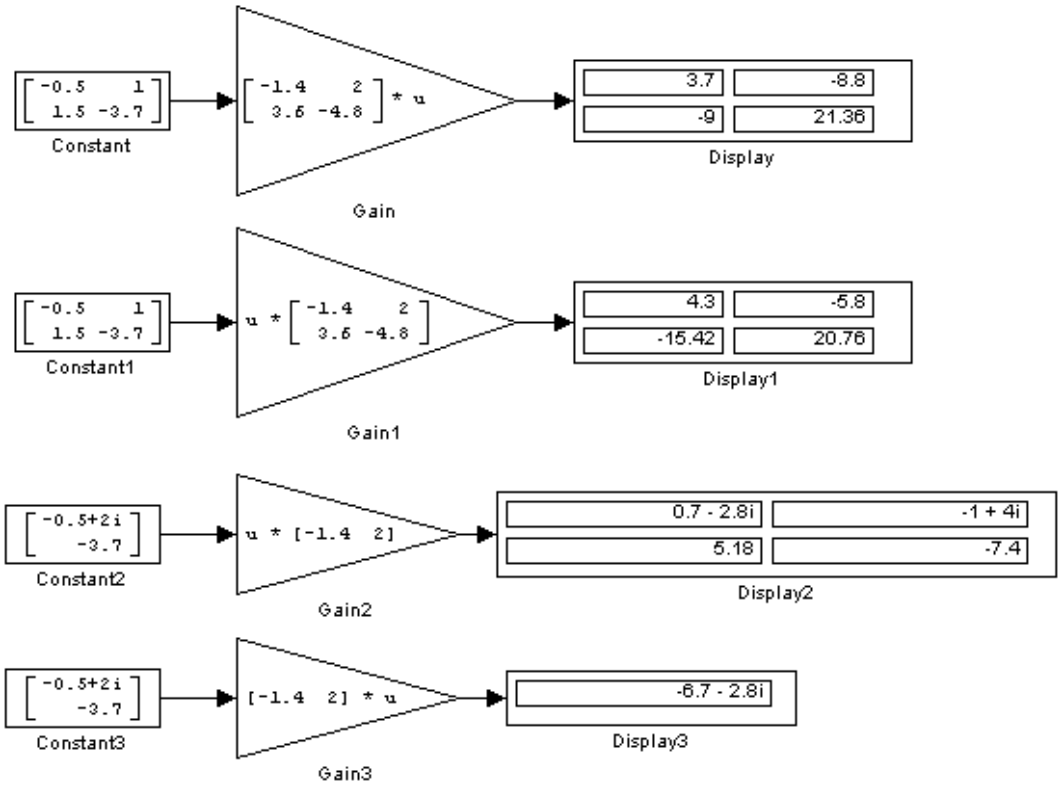
- Əgər giriş siqnalı həqiqi tipdədirsə, o zaman gücləndirmə əmsalı kompleks tipdədirsə, onda çıxış sıqnalı da kompleks tipli olacaqdır;
- Əgər giriş siqnalının tipi gücləndirmə əmsalının tipindən fərqlənirsə, o zaman **Simulink** gücləndirmə əmsalının tipini giriş siqnalının tipi ilə eyniləşdirmək istəyir. Bu eyniləşdirmə mümkün deyilsə, onda xəta alındığı haqda mesaj verilir. Bu cür vəziyyət giriş siqnalın tipi işarəsiz tam (**uint8**) ədəd kimi göstərilmişsə və **Gain** parametri də mənfi ədəd kimi verildiyi zaman ortaya çıxa bilər.

Gain blokun istifadəsinə aid nümunələr aşağıdakı şəkil 5.54-də göstərilmişdir.



Şəkil 5.54. **Gain** blokunun istifadəsinə aid nümunələr

Matris tipli gücləndirmə üçün (giriş siqnalının verilmiş əmsala matris hasili) giriş siqnalı və gücləndirmə əmsalı skalyar, vektor və ya matris şəklində olan bilər. Bu kəmiyyətlərin isə kompleks və həqiqi ədəd şəklində (tipləri isə **single** və ya **double)** olan bilər. Bu blokun istifadəsinə aid nümunə aşağıdakı şəkil 5.55-də göstərilmişdir.



Şəkil 5.55. **Matrix Gain** blokunun istifadəsinə aid nümunələr

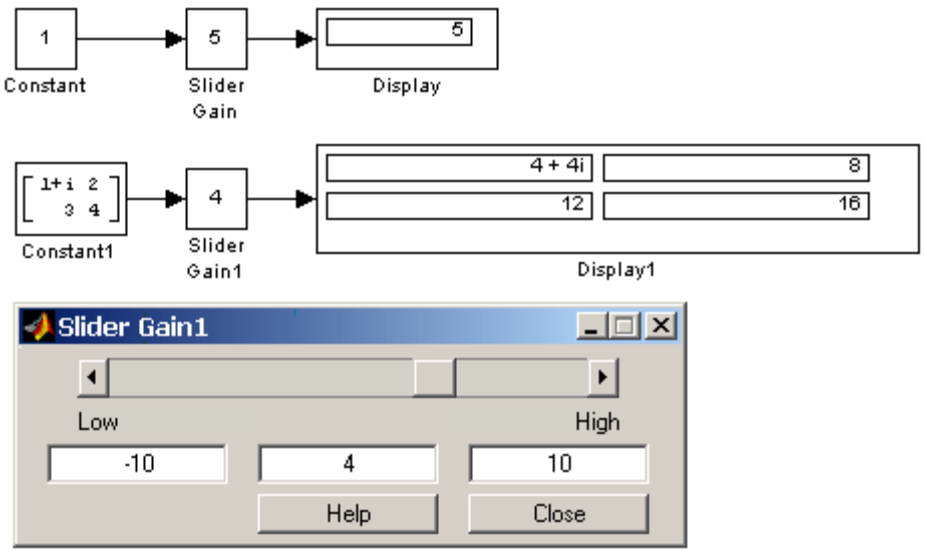
5.8.6. Sürüşkən düyməli tənzimləyici - Slider Gain

***Təyinatı*:** simulyasiya və hesablama müddəti ərzində gücləndirmə əmsalını dəyişməyə imkan verir.

Parametrlər

1. **Low** – Gücləndirmə əmsalının aşağı həddi
2. **High** – Gücləndirmə əmsalının yuxarı həddi.

Gücləndirmə əmsalını dəyişmək üçün **Slider Gain** blokunda yerləşən sürüşkən düyməni (slider) yerini dəyişdirmək lazımdır. Sağa sürüşdürməklə Gücləndirmə əmsalını artırmaq, sola sürüşdürməklə isə azaltmaq olar. Bu dəyişim Low və High diapazonu arasında olacaqdır. Bu blok vektor və matris tipli siqnallar üçündə yerinə yetirə bilər. Giriş siqnalı eyni zamanda kompleks tipli də ola bilər. **Slider Gain** blokunun istifadəsinə aid nümunələr aşağıda şəkil 5.56-da göstərilmişdir.



Şəkil 5.56. **Slider Gain** blokunun istifadəsinə aid nümunələr

5.8.7. Skalyar hasili hesablayan blok - Dot Product

Təyinatı: iki vektorun skalyar hasilini hesablayir.

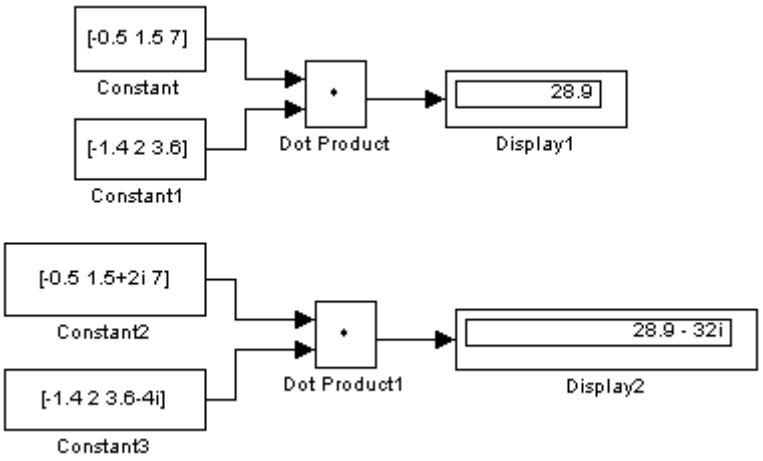
Parametrlər- *yoxdur.*

Bu blok çıxış siqnalını aşağıdakı ifadəyə görə hesablayır:

$$y = sum(conj(u1).* u2)\ ,$$

burada, **u1** və **u2** – giriş vektorlarıdır; **conj** – kompleks qoşa ədədin hesablanma əməliyyatı; **sum** – cəmi hesablayan əməliyyat.

Əgər hər iki giriş vektor həqiqi tipdədirsə, onda çıxış siqnal da həqiqi tipli olacaqdır. Əgər giriş vektorlarında biri kompleks vektor olarsa, o zaman çıxış vektoru da kompleks tipli olacaqdır. **Dot Product** blokunun istifadəsinə aid misal aşağıdakı şəkil 5.57-də göstərilmişdir.



Şəkil 5.57. **Dot Product** blokunun istifadəsinə aid nümunələr

5.8.8. Riyazi funksiyaları hesablayan blok - Math Function

Təyinatı: Riyazi funksiyaları hesablayır

Parametrlər

1. **Function –** hesablanan riyazi funksiya (siyahıdan seçilir):

exp – eksponensial funksiya;
log – natural loqarifm funksiyası;
10^u – 10 qüvvətinin hesablanması;
log10 – loqarifm funksiyası;
magnitude^2 – giriş siqnalının modul kvadratının hesablanması;
square – giriş siqnalının kvadratının hesablanması;
sqrt – kvadrat kökaltı;
pow – qüvvətə yüksəltmə;
conj – kompleks qoşma ədədin hesablanması;
reciprocal – 1-ci giriş siqnalının 2-ci siqnala bölünməsində alınan qismətin hesablanması;
hypot – giriş siqnallarının kvadratları cəminin kvadrat kökünü hesablayır;
rem – 1-ci giriş siqnalının 2-ci siqnala bölünməsində alınan qalıq;
mod – işarəsini diqqətə alan bölmədən alınan qalığını hesablayan funksiya
transpose – matrislərin transpor edilməsi;
hermitian – ermit matrisinin hesablanması

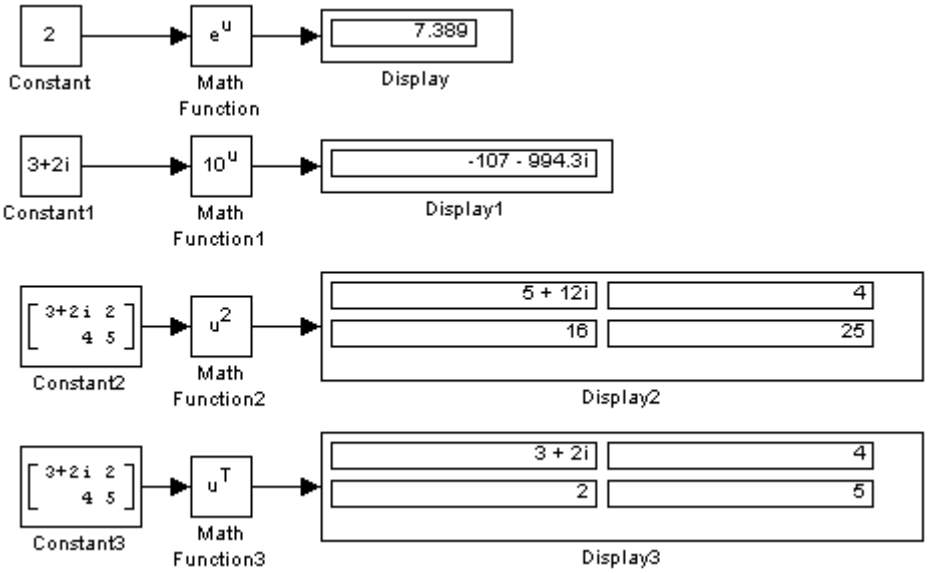
2. **Output signal type – çıxış siqnalının tipi** (siyahıdan seçilir):

auto – tipin avtomatik təyin edilməsi
real – həqiqi siqnal
complex- kompleks siqnal

Hesablanan funksiyanın və **Output signal type** blokunun parametrindən asılı olaraq çıxış sıqnalının giriş sıqnalının tipində asılılığı cədvəldə göstərilmişdir.

Funksiya	Giriş siqnal	Çıxış siqnalı		
		Avto	Real	Complex
Exp, log, 10^u, log10, square, sqrt, pow, reciprocal, conjugate, transpose, hermitian	real complex	real complex	real error	complex complex
magnitude squared	real complex	real real	real real	complex complex
Hypot, rem, mod	real complex	real error	real error	complex error

Math Function blokunun istifadə olunmasına aid misallar aşağıdakı şəkil 5.58-də göstərilmişdir.



Şəkil 5.58. **Math Function** blokunun istifadə olunmasına aid nümunələr

5.8.9. Triqonometrik funksiyaları hesablayan blok - Trigonometric Function

Təyinatı: Triqonometrik funksiyaları hesablayır.

Parametrlər

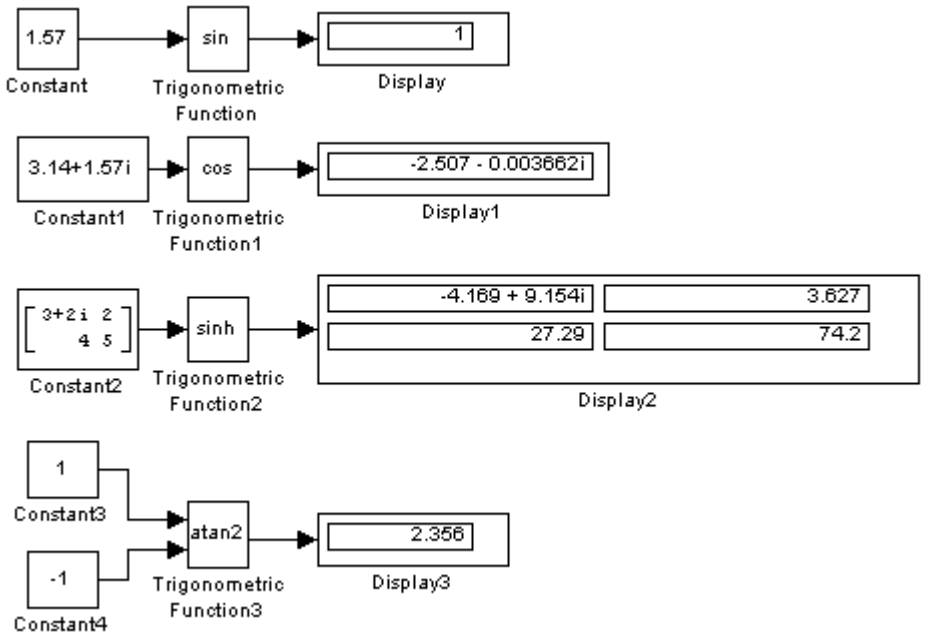
1. **Function –** hesablanan funksiyalar bunlardır: **sin, cos, tan, asin, acos, atan, atan2, sinh, cosh** və **tanh.**
2. **Output signal type** – Çıxış siqnalının tipi (siyahıdan seçilir):

- auto – siqnal tipinin avtomatik təyin edilməsi

- real – həqiqi siqnal

- complex- kompleks siqnal

Giriş siqnallar vektor və matris tipli olduğunda bu blok verilmiş funksiyanı element-element yerinə yetirir. **Trigonometric Function** blokunun istifadəsinə aid nümunələr aşağıdakı şəkil 5.59-da göstərilmişdir.



Şəkil 5.59. **Trigonometric Function** blokunun istifadəsinə aid nümunələr

5.8.10. Kompleks ədədin həqiqi və xəyali hissələri ayıran blok - Complex to Real-Imag

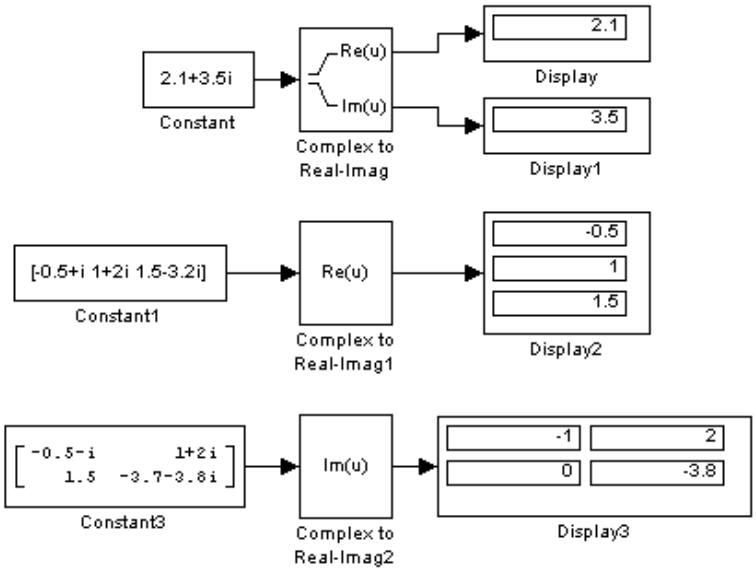
Təyinatı: Kompleks ədədi həqiqi və xəyali hissələrə ayırır.

Parametrlər:

Output – Çıxış siqnal (siyahıdan seçilir):

- **Real –** həqiqi hissə
- **Image –** xəyali hissə
- **RealAndImage –** həqiqi və xəyali hissə

Blokun giriş siqnalı skalyar, vektor və matris tipli ola bilər. **Complex to Real-Imag** blokunun istifadə olunma nümunələri aşağıdakı şəkil 5.60-da göstərilmişdir.



Şəkil 5.60. **Complex to Real-Imag** blokunun istifadə olunmasına aid nümunələr

5.8.11. Kompleks ədədin modul və arqumentinin hesablanması bloku - Complex to Magnitude-Angle

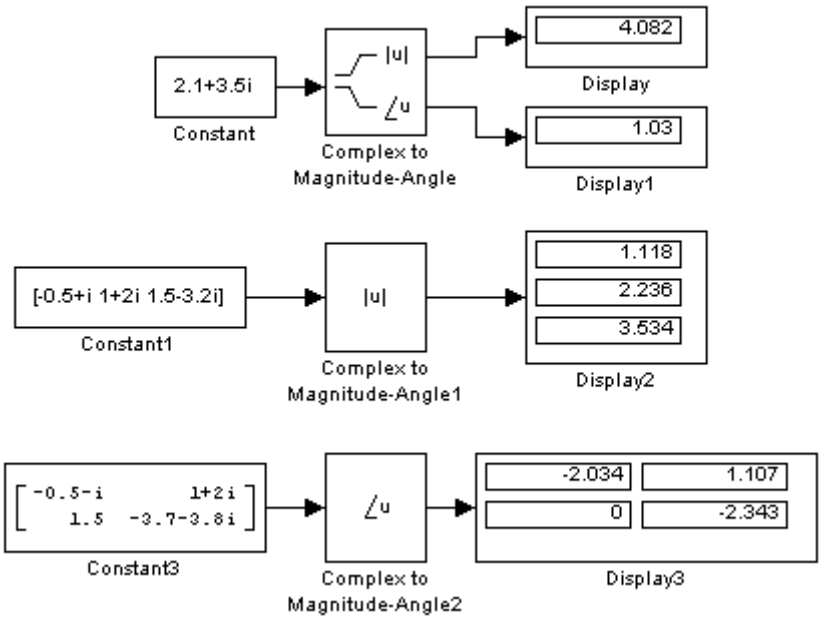
Təyinatı: Kompleks ədədin modulunu və ya arqumentini hesablayır

Parametrlər:

Output – çıxış siqnalı (sıyahıdan seçilir):

- **Magnitude** – modul
- **Angle** – arqument
- **MagnitudeAndAngle** – modul və arqument

Blokun giriş siqnalı skalyar, vektor və matris tipli ola bilər. **Complex to Magnitude-Angle** blokunun istifadə olunma nümunəsi şəkil 5.61-də göstərilmişdir.



Şəkil 5.61. **Complex to Magnitude-Angle** blokunun istifadəsinə aid nümunələr

5.8.12. Həqiqi və xəyali hissələrə görə kompleks ədədin formalaşdırılması- Real-Imag to Complex

Təyinatı: Həqiqi və xəyali hissələri əsasında kompleks ədədi hesablayır.

Parametrlər

1. **Input –** Giriş siqnalı (siyahıdan seçilir)

   Real – Həqiqi hissə

   Image – Xəyali hissə

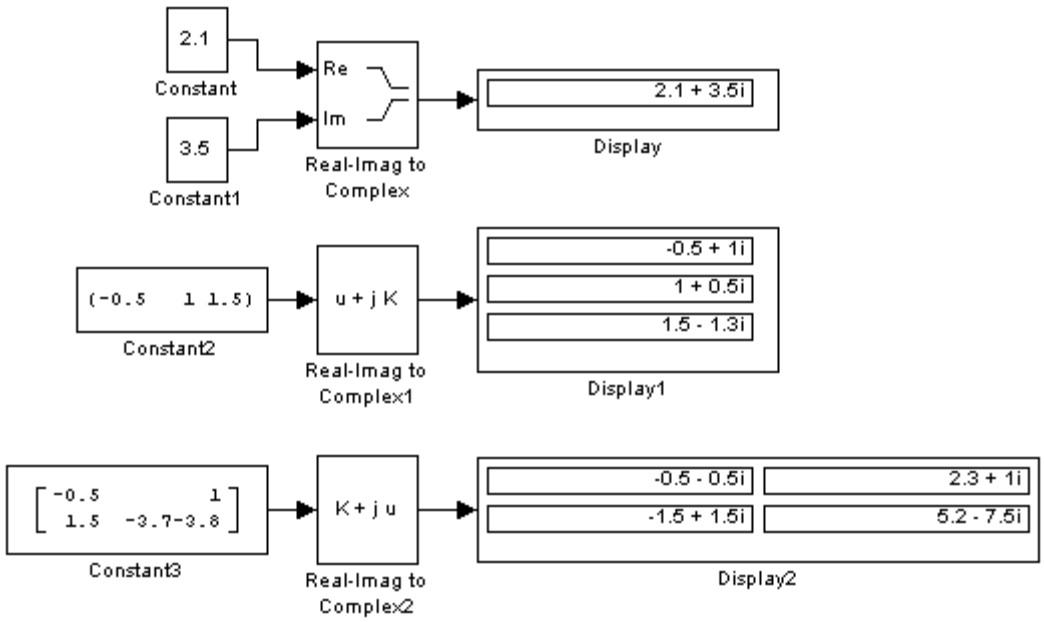
2. **RealAndImage –** həqiqi və xəyali hissə

3. **Image part** – xəyali hissə. Bu parametr o zaman açıq olur ki, əgər **Input** parametri **Real** kimi elan edilib.

4. **Real part** –Həqiqi hissə. Bu parametr o zaman açıq olur ki, əgər **Input** parametri **Image** kimi elan edilmiş olsun.

Blokun giriş siqnalları həm skalyar, həm vektor, həm də matris tipli ola bilərlər. **Image part və Real part** parametrləri ya vektor, ya da matris kimi verilməlidirlər, amma bunun üçün giriş siqnalın mütləq vektor və ya matris şəklində verilməsi lazımdır.

Blokun istifadəsinə aid nümunələr aşağıdakı şəkil 5.62-də göstərilmişdir.



Şəkil 5.62. **Real-Imag to Complex** blokunun istifadəsinə aid nümunələr

5.8.13. Modul və arqumentinə görə kompleks ədədin hesablanması- Magnitude-Angle to Complex

Təyinatı: *Modul və arqumentinə görə kompleks ədədi bərpa edir.*

Parametrlər

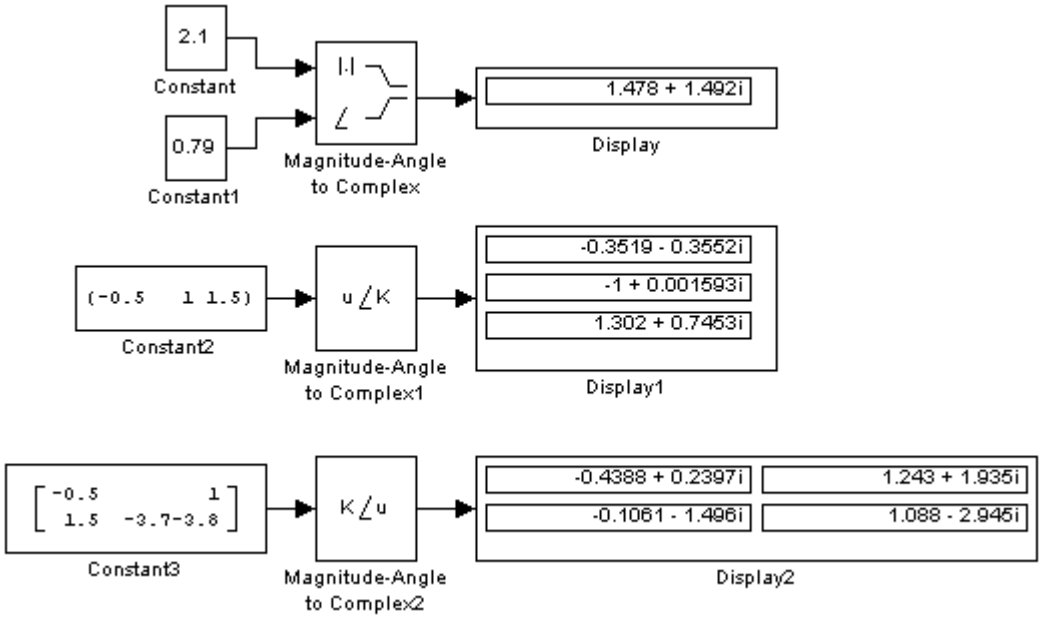
1. **Input – Giriş siqnal** (siyahidan seçilir):

   - Magnitude – Modul

- **Angle –** arqument
- **MagnitudeAndAngle –** modul və arqument

2. **Angle** – Arqument. Bu parametr o zaman açıq olur ki, əgər **Input** parametri **Magnitude** kimi elan edilibsə.
3. **Magnitude** – Modul. Bu parametr isə o zaman açıq olur ki, əgər **Input** parametri **Angle** kimi elan edilibsə.

Giriş siqnalları skalyar, vektor və matris tipli ola bilərlər. **Angle** və **Magnitude** parametrlər ya vektor, ya da matris şəklində verilə bilər. Bunun üçün giriş siqnalının vektor və matris olması lazımdır. **Magnitude-Angle to Complex** blokunun istifadə olunmasına aid nümunə aşağıda şəkil 5.63-də göstərilmişdir.



Şəkil 5.63. **Magnitude-Angle to Complex** blokunun istifadəsinə aid nümunələr

5.8.14. Minimal və maksimal qiymətləri təyin edən blok - MinMax

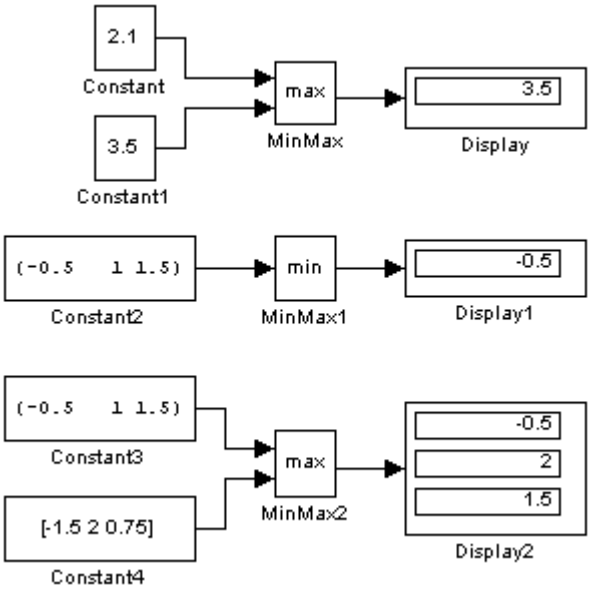
Təyinatı: girişinə daxil olan siqnalların minimal və maksimal qiymətlərini təyin edir.

Parametrlər

1. **Function** – çıxış parametr. Siyahıdan seçilir:
   - **min** – minimal qiymət
   - **max** – maksimal qiymət
2. **Number of input ports** – giriş portlarının sayı.

Blokun giriş siqnalları ya skalyar, ya da vektor tipli ola bilərlər. Bu blok onun gırışinə daxil olan bütün skalyar siqnalların minimal və maksimal qiymətlərini hesablayır. Əgər giriş siqnallar vektor şəklində olsa, o zaman blok element-element əməliyyatları yerinə yetirir.

Bu halda vektorların ölçüləri üst-üst düşməlidir. Əgər giriş portların sayı 1-ə bərabərdirsə, o zaman blok giriş vektorunda minimal və maksimal qiymətləri hesablayacaqdır. **MinMax** blokunun istifadəsinə aid nümunələr aşağıdakı şəkil 5.64-də göstərilmişdir.



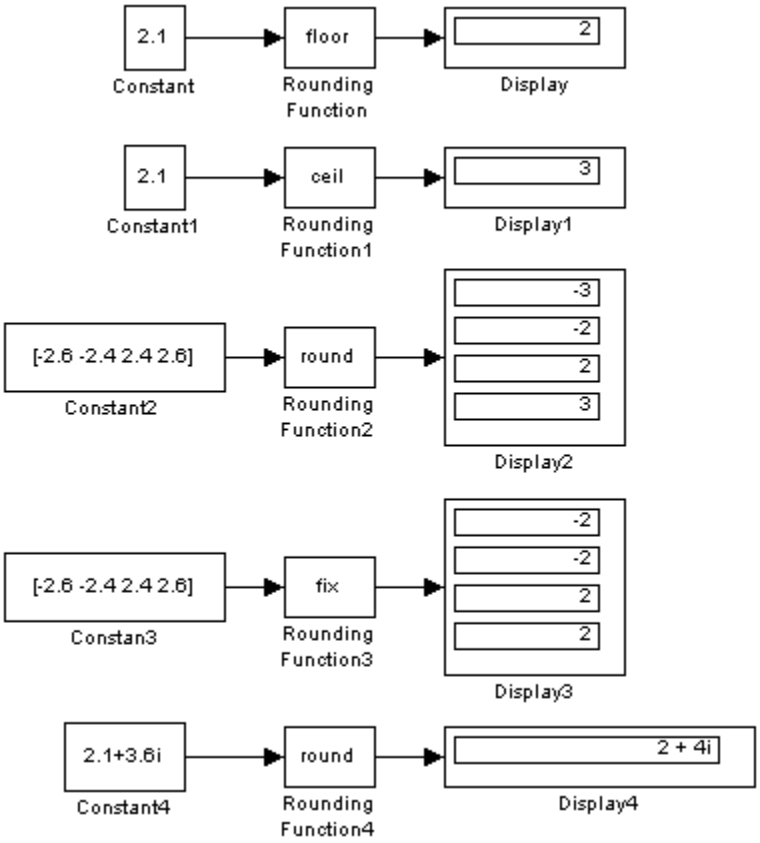
Şəkil 5.64. **MinMax** blokunun istifadəsinə aid nümunələr

5.8.15. Ədədlərin yuvarlaqlaşdırması bloku - Rounding Function

Təyinatı: Ədədi qiymətlərin yuvarlaqlaşdırılmasını yerinə yetirir.
Parametrlər: yuvarlaqlaşdırma üsulları aşağıdakı **Function** siyahısından seçilir:
Function-da olan üsullar bunlardır:

- **floor** – ən yaxın kiçik tam ədəd kimi yuvarlaqlaşdırma
- **ceil** – ən yaxın böyük tam ədəd kimi yuvarlaqlaşdırma
- **round** – ən yaxın tam ədədə kimi yuvarlaqlaşma
- **fix** – kəsir hissəni atmaqla yuvarlaqlama

Blokun giriş siqnalları həqiqi və kompleks tipli həm skalyar, vektor, həm də matris tipli olurlar. Əgər girişdə vektor və matris tipli siqnal varsa, o zaman blok element-element əməliyyatı yerinə yetirir. Blokun çıxış siqnalı **double** və ya **single** tipli olur. Bu blokun istifadəsinə aid nümunələr aşağıdakı şəkil 5.65-də göstərilmişdir.



Şəkil 5.65. **Rounding Function** blokunun istifadəsinə aid nümunələr

5.8.16. Münasibət operatorunu yerinə yetirən blok - Relational Operator

Təyinatı: Giriş siqnallarını müqayisə edən blok.

Parametrlər: **Relational Operator** – münasibətlər əməliyyatlarının tipi aşağıdakı siyahıdan seçilir:

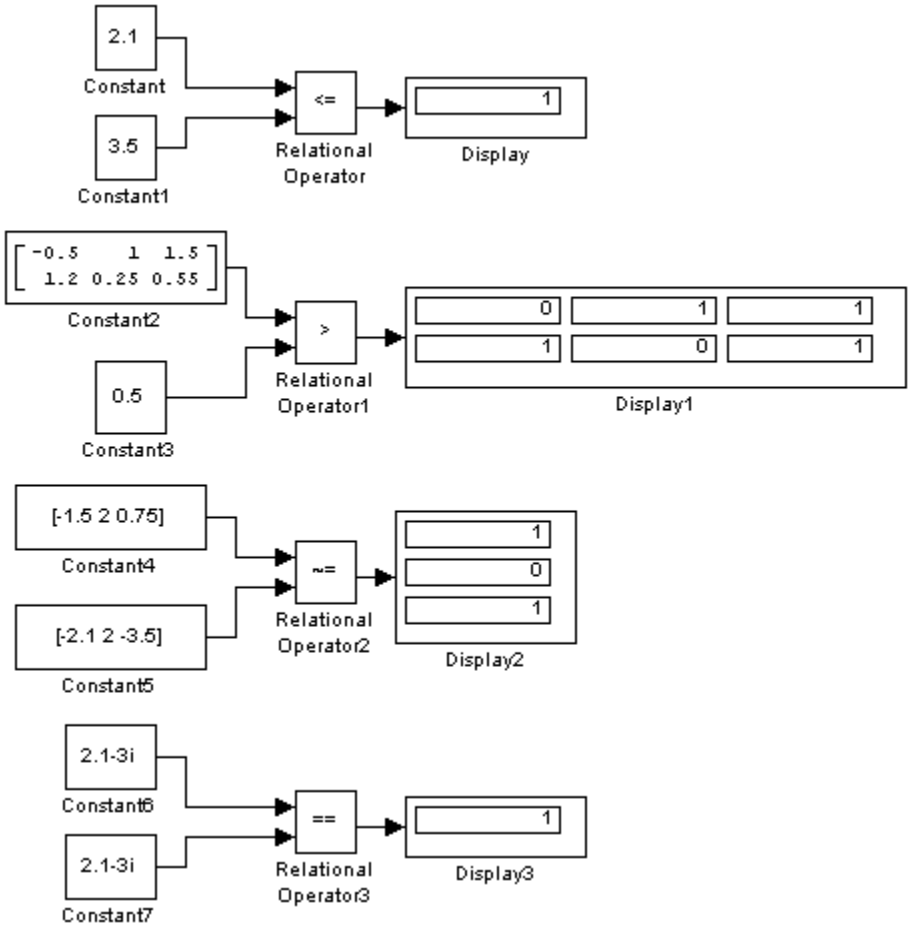
- = = - eyniliklə doğru.
- ~ = - bərabər deyil.
- < - kiçik.
- < = - kiçikdır və ya bərabərdir.
- > = - böyükdür və ya bərabərdır.
- > - böyükdür.

Münasibət operatorunda və ya əməliyyatında birinci operand blokun birinci girişinə verilən siqnal, ikinci operand isə blokun ikinci və ya aşağı girişinə verilən siqnaldır. Blokun çıxış siqnalı o zaman məntiqi 1 olur ki, münasibət operatoru "doğrudur", sıfır isə tam tərsi.

Blokun giriş siqnalı həm skalyar, həm vektor, həm də matris tipli ola bilər. Əgər hər iki siqnal vektor və ya matrisdirsə, onda blok element-element müqayisə edir. Bu zaman giriş siqnalların ölçüləri üst-üstə düşməlidir. Əgər iki giriş siqnallarından biri ya vektor və ya matrissə, digər siqnal isə skalyardırsa, o zaman blok massivin hər bir elementi ilə skalyar

giriş sıqnalı müqayisə edir. Çıxış siqnalın ölçüsü bu halda blokun girişinə verilən vektor və ya matris siqnalının ölçüsü ilə təyin edilir. = = (eynilikdə bərabərdir) və ~ = (bərabər deyil) əməliyyatları kompleks giriş siqnalları üçün istifadə edilə bilinər. Giriş siqnalları həm də məntiq tipli ola bilərlər (**boolean**).

Relational Operator blokunun istifadəsinə aid nümunələr şəkil 5.66-da göstərilmişdir.



Şəkil 5.66. **Relational Operator** blokunun istifadəsinə aid nümunələr

5.8.17. Məntiqi əməliyyatları yerinə yetirən blok - Logical Operation

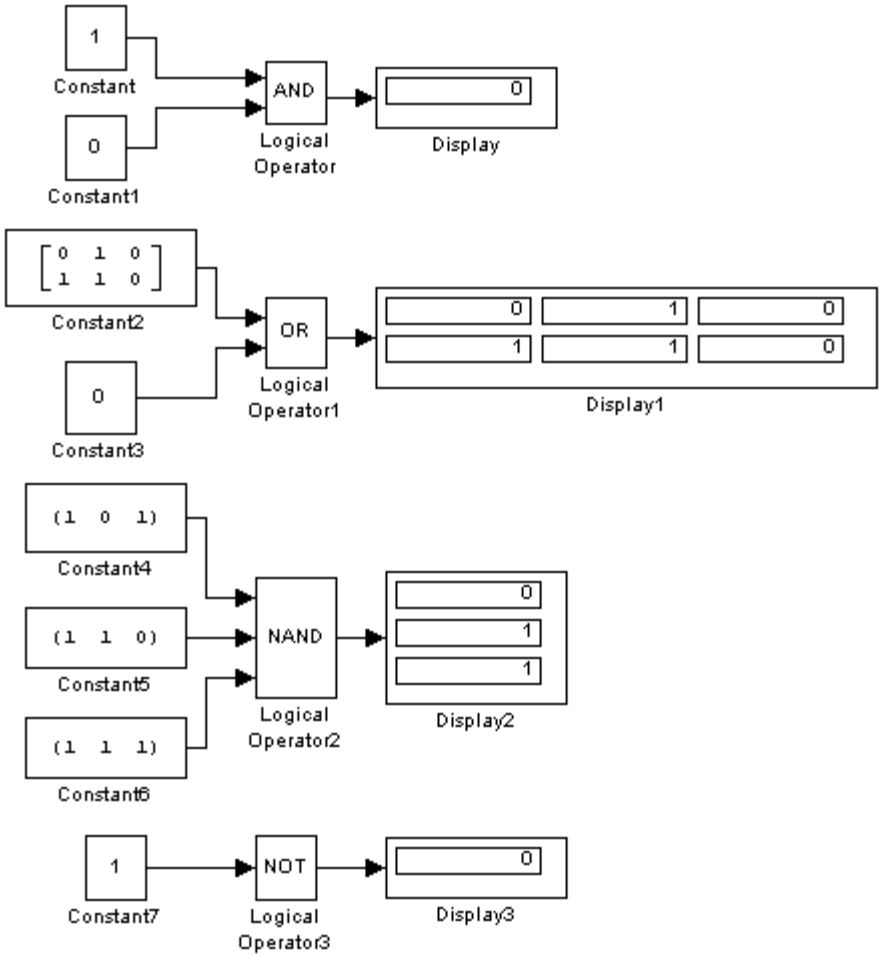
Təyinatı: Əsas məntiqi əməliyyatları yerinə yetirir.

Parametrlər

1. **Operator** – yerinə yetirilən məntiqi əməliyyatlar(siyahıdan seçilir):
- **AND** – məntiqi vurma
- **OR** – məntiqi toplama.
- **NAND** – VƏ-YOX əməliyyatı
- **NOR** – VƏ YA –YOX əməliyyatı
- **XOR** – istisna VƏ YA(2 modula görə toplama)
- **NOT** – Məntiqi YOX

2. **Number of input ports** – giriş portlarının sayı

Bu blokun çıxış sıqnalı o zaman məntiqi 1 olur ki, məntiqi əməliyyatın nəticəsi **“doğru”** olsun. Əks halda isə nəticə məntiqi 0 olur. Məntiqi inkar əməliyyatını yerinə yetirdiyiniz zaman blokun sadəcə bir giriş portu vardır. Giriş siqnalları həm həqiqi həm də məntiqi tipli ola bilərlər. Bu blokun istifadəsinə aid nümunələri aşağıdakı şəkil 5.67-də görə bilərsiniz.



Şəkil 5.67. **Logical Operation** blokunun istifadəsinə aid nümunələr

5.8.18. Bit-bit məntiqi əməliyyatları yerinə yetirən Bitwise Logical Operator bloku

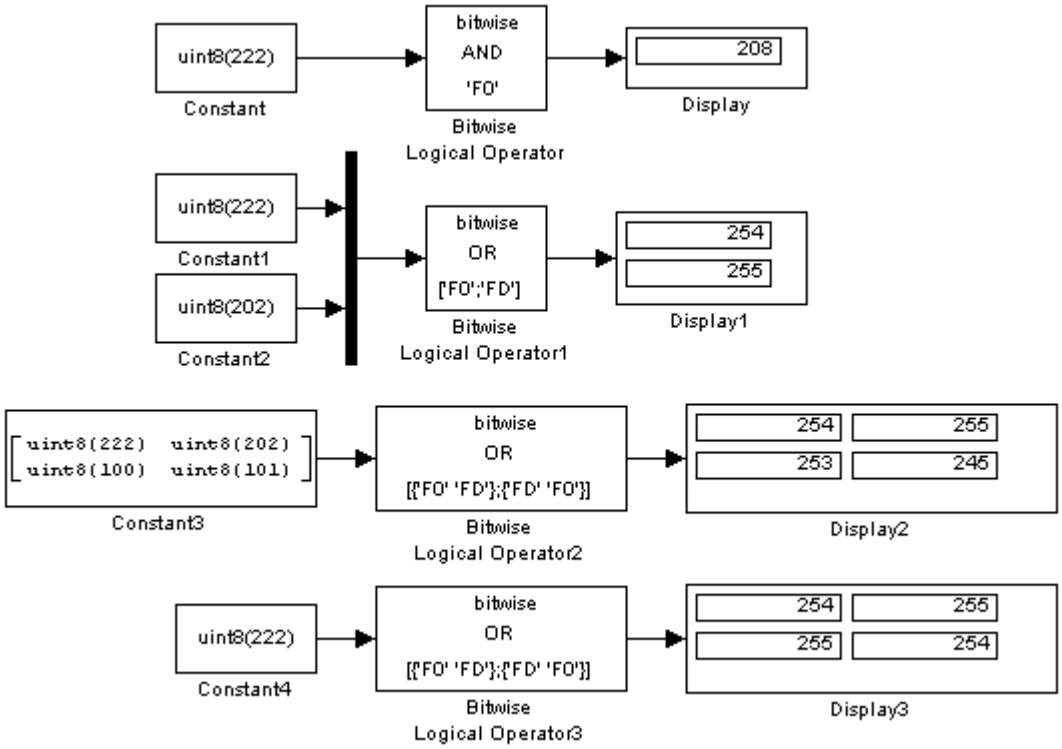
Təyinatı: İkilik say sistemində verilmiş tam ədədlər üzərində məntiqi əməliyyatı yerinə yetirir.

Parametrlər: **Bitwise operator –** Məntiqi əməliyyatların tipi (siyahıdan seçilir):

- **AND** – məntiqi vurma (AND əməliyyatı).
- **OR** – məntiqi toplama (OR əməliyyatı).
- **XOR** – istisnalı OR (2 moduluna görə toplama).
- **NOT** – məntiqi inkar (NOT).
- **SHIFT_LEFT –** mərtəbələrin sola sürüşdürülməsi.
- **SHIFT_RIGHT –** mərtəbələrin sağa sürüşdürülməsi.

Second operand – 2-ci operand. Simvol şəklində verilmiş onaltılıq say sistemində verilir.

Birwise Logical Operator blokunun iki operandı verilir, onlardan birinci operand blokun girişinə verilən siqnaldır, ikinci operand isə **Second operand** blokunun parametridir. Giriş siqnalları kimi **uint8, uint16** və **uint32** tipli işarəsiz dəyişənlər olmalıdır. Giriş siqnalı həm skalyar həm vektor həm də matris tipli ola bilər. Əgər giriş və çıxış siqnalları vektor və ya matris tiplidirsə, o zaman blok məntiqi əməliyyatı element-element yerinə yetirir. Bu zaman operandların ölçüləri üst-üstə düşməlidirlər. Əgər operandlardan biri vektor və ya matris, o birisi isə skalyar isə onda blok, vektor və ya matrisin hər elementi üçün skalyar operand üçün nəzərdə tutulmuş məntiqi əməliyyatı yerinə yetirir. Bu halda çıxış sıqnalının ölçüləri vektor və matris operandının ölçüləri ilə təyin edilir. Məntiqi inkar əməliyyatını yerinə yetirən zaman sadəcə bir giriş siqnalı lazımdır. **Bitwise Logical Operator** blokunun istifadəsinə aid nümunələr aşağıda şəkil 5.68-də göstərilmişdir.



Şəkil 5.68. **Bitwise Logical Operator** blokunun istifadəsinə aid nümunələr

5.8.19. Kombinator məntiq bloku- Combinatorical Logic

Təyinatı: Giriş siqnalların doğruluq cədvəlinə uyğun çevirir.
Parametrlər: **Truth table – doğruluq cədvəli**.

Combinatorical Logic bloku giriş siqnalını doğruluq cədvəlində təyin edilmiş qaydalara görə başqa şəklə çevirir. Doğruluq cədvəli çıxış qiymətlərinin bütün mümkün qiymətlərini özündə əks edir. Qurğuların bu cür təsvir edilməsi sonlu avtomatlar nəzəriyyəsində qəbul edilmiş bir qaydadir. Doğruluq cədvəlində sətirlərin sayı aşağıdakı ifadə ilə təyin edilir:

number of rows = 2 ^ (number of inputs)

Burada **number of inputs** – giriş siqnallarının sayı, **number of rows – doğruluq cədvəlinin sətirlərin sayı.**

Doğruluq cədvəlinin tərtib edilməsində giriş siqnalları verilmiş hesab edilir. Onlar indeks, yəni sətrin nömrəsini təyin edir. Bura blokun çıxış qiymətləri yazılır. Hər bir sətrin indeksi aşağıdakı ifadə ilə müəyyən edilir:

$$row\ index = 1 + u(m) * 2^0 + u(m-1) * 2^1 + \cdots + u(1) * 2^{m-1}$$

Burada **row index** – sətir indeksi,**m** – giriş siqnalların sayı (giriş vektorunda olan elementlər), **u(1)** – 1-ci giriş siqnalı (giriş vektorunun 1-ci elementi), **u(m)** – sonuncu giriş siqnalı (sonuncu giriş siqnalı).

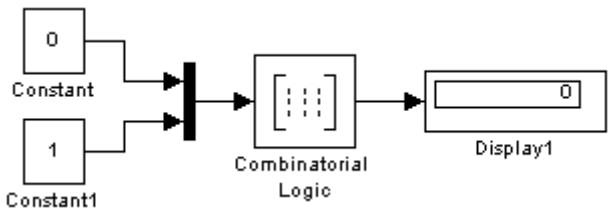
Məsələn iki operand üçün məntiqi VƏ (**AND**) üçün istifadə edilərkən, sətrin indeksini təyin etmək üçün lazım olan ifadə aşağıdakı şəkildə olacaqdır:

$$row\ index = 1 + u(2) * 2^0 + u(1) * 2^1$$

Aşağıdakı şəkildə iki operand üçün istifadə olunan məntiqi VƏ (AND) üçün doğruluq cədvəlinin formalaşması nümunəsi göstərilmişdir:

Giriş 2	Giriş 1	Sətir indeksi üçün istifadə olunan ifadə	Sətir indeksinin qiymətləri	Doğruluq cədvəli (çıxış)
0	0	$1+0\cdot 2^0+0\cdot 2^1$	1	0
1	0	$1+1\cdot 2^0+0\cdot 2^1$	2	0
0	1	$1+0\cdot 2^0+1\cdot 2^1$	3	0
1	1	$1+1\cdot 2^0+1\cdot 2^1$	4	1

Aşağıdakı şəkil 5.69-da məntiqi AND–in **Combinatorical Logic** blokunun köməyi ilə realizə edilməsi nümunəsi göstərilmişdir. Doğruluq cədvəlinin (**Truth table)** parametri **[0;0;0;1]** ifadəsi ilə verilir.



Şəkil 5.69. **Combinatorical Logic** blokundan istifadə olunma nümunəsi

5.8.20. Cəbri kontur bloku - Algebraic Constraint

Təyinatı: Cəbri tənliyin köklərini axtarır.

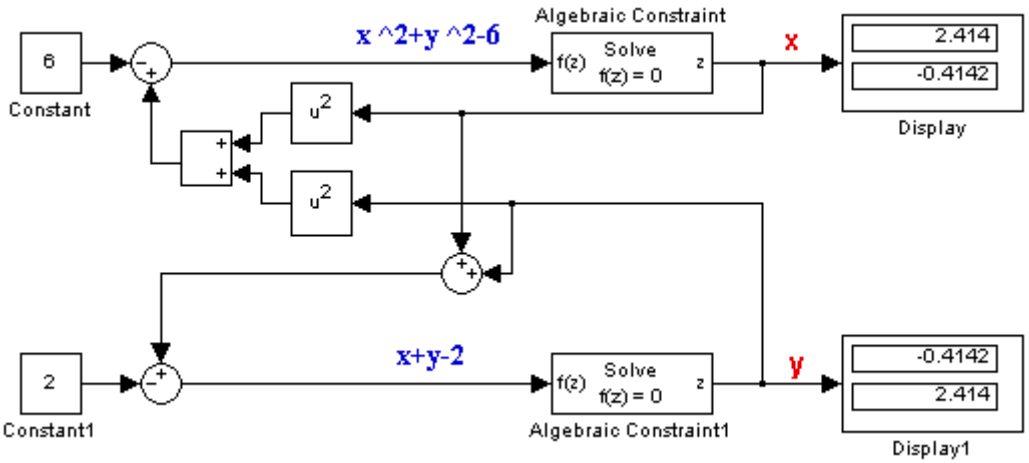
Parametrlər: **Initial guess –** çıxış sıqnalının başlanğıc qiymətləri.

Bu blok çıxış siqnallarının elə qiymətlərini tapir ki, onlara uyğun giriş siqnalları sıfıra bərabər olsun. Belə olan halda giriş siqnalı birbaşa və ya dolayısı ilə çıxış siqnalı ilə bağlı olmalıdır.

Aşağıdakı şəkildə qeyri-xətti tənliklər sisteminin həll edilmə nümunəsi göstərilmişdir:

$$\begin{cases} x^2 + y^2 = 6 \\ x + y = 2 \end{cases}$$

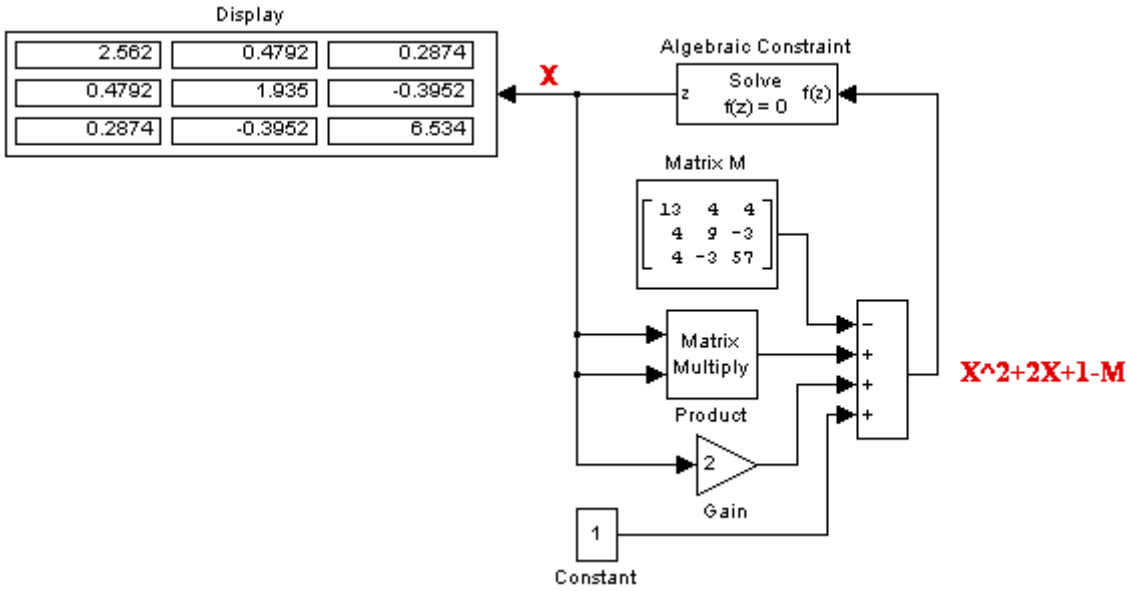
Verilmiş tənliklər sisteminin iki həlli olduğu üçün **Algebraic Constraint** blokunun başlanğıc qiymətləri vektor şəklində verilmişdir. Birinci (üst) blok üçün başlanğıc qiymətlər **[1 -1]** vektoru ilə, ikinci (alt) blok – [-1 1] vektorunun köməyi ilə verilir (şəkil 5.70).



Şəkil 5.70. **Algebraic Constraint** blokundan istifadə olunmasına aid nümunə

Bununla bərabər **Algebraic Constraint** bloku qeyri xətti matris tənliyinin həlli üçün istifadə edilə bilinər. Aşağıdakı şəkildə verilmiş qeyri xətti matris tənliyinin həll edilmə nümunəsi verilmişdir:

$$X^2 + 2 \cdot X + 1 = \begin{vmatrix} 13 & 4 & 4 \\ 4 & 9 & -3 \\ 4 & -3 & 57 \end{vmatrix}.$$



Şəkil 5.71. Qeyri-xətti matris tənliyinin həll edilməsinə aid **Algebraic Constraint** blokundan istifadə olunma nümunəsi