

2.6. Simbolių eilutės

Simbolių eilutė – tai simbolių seka. Pavyzdžiui, simbolių eilutėmis bus: ‘sveiki’, ‘Petras Petraitis’, ‘12’ ir t.t. Nors paskutinis pavyzdys yra skaičius 12, tačiau, kadangi jis yra tarp apostrofų, MATLAB‘as jį interpretuoja kaip simbolių eilutę. Kiekvienas simbolis turi savo ASCII kodą. Ryšį tarp simbolių ASCII skaitinių kodų ir juos atitinkančių simbolių nusako MATLAB‘o komandos *double* ir *char*.

Komanda *double* keičia argumento simbolinį formatą į dvigubo tikslumo skaitinį formatą, kuris pagal nutylėjimą yra pagrindinis MATLAB‘o formatas.

Komanda *char* atlieka priešingą veiksmą, t.y. skaičių keičia simboliu.

Pavyzdžiai.

```
>> alph = 'ABCDE' % alph priskiriame simbolių eilutę „ABCDE“
alph = ABCDE
>> num = double(alph) % simbolių „ABCDE“ eilutė keičiama ASCII kodais
num = 65 66 67 68 69
>> char(num) % ASCII kodai keičiami simbolių „ABCDE“ eilutė
ans = ABCDE
>> char(num+5)
ans = FGHIJ
```

Veiksmai su eilutėmis. Surinkę komandų lange komandą *help strfun*, ekrane pamatysime funkcijas skirtas darbui su eilutėmis. Pagrindinės funkcijos pateiktos 2.15 lentelėje.

2.15 lentelė. Pagrindinės darbo su simbolių eilutėmis funkcijos

Funkcija	Paskirtis
<i>double</i>	Simbolių eilutę keičiama į skaitinį dvigubo tikslumo formatą
<i>char</i>	Skaičius keičiamas į simbolius (komanda priešinga <i>double</i>)
<i>isspace</i>	Simbolių eilutėje randami tarpo simbolio elementai
<i>isletter</i>	Simbolių eilutėje randami raidės simbolio elementai
<i>findstr</i>	Simbolių eilutėje randama nurodytą simbolių seką
<i>strrep</i>	Nurodyta simbolių seka keičiama kita simbolių seka
<i>int2str</i>	Sveikasis skaičius keičiamas simbolių eilute
<i>num2str</i>	Realusis skaičius keičiamas simbolių eilute

Pavyzdžiai.

1. Raskime tarpo ir tabuliacijos simbolius simbolių eilutėje ‘Man patinka MATLABas’.

```
>> str = 'Man patinka MATLABas'
str =
Man patinka MATLABas
>> isspace(str)
ans = 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
```

Vienetukai žymi tarpų vietas simbolių eilutėje.

2. Galima surasti ir raidžių vietas.

```
>> isletter(str)
ans = 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
```

Vienetukai žymi raidžių vietas simbolių eilutėje.

3. Simbolių eilutėje konkrečių simbolių seką randa komanda

pos=findstr (simbolių eilutė, ieškoma simbolių seka),
čia
simbolių eilutė – turima simbolių eilutė,
ieškoma simbolių seka - konkreti – ieškoma simbolių seka,
pos - ieškomų simbolių sekos pirmojo simbolio pozicijų
numerinių masyvas.

```
>> pos = findstr('Man patinka MATLABas','patinka')
pos = 5
```

Simbolių eilutėje 'Man patinka MATLABas' žodžio 'patinka' pirmasis simbolis yra 5-je pozicijoje, t.y. žodis 'patinka' prasideda nuo 5-os pozicijos.

```
>> pos = findstr('Man patinka MATLABas','a') % ieškoma raidė 'a'
pos = 2      6      11      19
```

Raidė 'a' rasta 2, 6, 11 ir 19 pozicijose.

4. Komanda **strrep** keičia nurodytą simbolių seką kita simbolių seka.

```
>> strrep('Man patinka MATLABas','Man','Jiems')
ans = Jiems patinka MATLABas
```

Simbolius galima keisti ir visoje eilutėje, kur tik jie sutinkami.

```
>> strrep('Man patinka MATLABas','n','N')
ans = MaN patiNka MATLABas
```

5. Iš eilutės simboliai šalinami pakeičiant juos tuščia simbolių seka, t.y. " arba [].

```
>> strrep('Man patinka MATLABas','as','')
ans = Man patinka MATLAB
```

Skaičių keitimas simbolių eilute. Funkcijos *num2str* ir *int2str* skaičius keičia simbolių eilute.

Komanda *int2str* sveikąjį skaičių keičia simbolių eilute.

Pavyzdys.

```
>> for i = 1:3
disp(['Ciklo numeris ' int2str(i) ' viso 3'])
end
```

```
Ciklo numeris 1 viso 3
Ciklo numeris 2 viso 3
Ciklo numeris 3 viso 3
```

Funkcija *num2str* realųjį arba menamąjį skaičių keičia simbolių eilute.

Pavyzdys.

```
for i = 1:3
disp(['Indeksas ' int2str(i) ', saknis = ' num2str(sqrt(i))'])
end
```

```
Indeksas 1, saknis = 1
Indeksas 2, saknis = 1.4142
Indeksas 3, saknis = 1.7321
```

Antrasis funkcijos *num2str* parametras nurodo skaičiaus, keičiamo eilute, išvedamų reikšminių skaitmenų skaičių.

Pavyzdžiui, slankaus kablelio formate

```
>> num2str(pi,2)
```

```
ans = 3.1
```

```
>> num2str(pi,15)
```

```
ans = 3.14159265358979,
```

arba eksponentiniame formate:

```
>> num2str(pi,'%12.5e')
```

```
ans = 3.14159e+00
```

```
>> num2str(-pi,'%12.5e')
```

```
ans = -3.14159e+00
```

```
>> num2str(pi*1e100,'%12.5e')
```

```
ans = 3.14159e+100.
```

Antrasis funkcijos *num2str* parametras leidžia rašyti ir tekstą. Pavyzdžiui,

```
>> num2str(exp(1),'Naturinio logaritmo pagrindas yra %12.5e')
```

```
ans = Naturinio logaritmo pagrindas yra 2.71828e+000
```

2.7 MATLAB'o redaktorius

MATLAB'o redaktorius skirtas *m*-failams (*m-file*) kurti. *m*-failų turinys yra MATLAB'o komandų ar funkcijų seka, realizuojanti konkretų algoritmą. *m*-failus kuria vartotojas.

Naujo *m*-failo sukūrimas. Naują *m*-failą galima sukurti dvejopai:

1) MATLAB'o įrankių juostoje kairiuoju pelės mygtuko paspaudimu aktyvinama komandų seka: *File- New- m-file*.

2) MATLAB'o įrankių juostoje kairiuoju pelės mygtuko paspaudimu aktyvinama tuščio popieriaus lapo piktograma.

Pastaba. MATLAB'o *m*-failas visuomet turi prievardį *.m*, t.y. *failo vardas.m*.

Esamo *m*-failo atidarymas. Atidaryti esamą *m*-failą taip pat galima dviem būdais:

1) MATLAB'o komandų lango įrankių juostoje aktyvinama komandų seka: *File-Open...*;

2) komandinio lango įrankių juostoje aktyviname failo atidarymo piktogramą. Pažymėtas reikiamas *m*-failas aktyvinamas *Open* mygtuku ir MATLAB'o redaktoriaus lange bus patalpintas *m*-failo turinys.

Darbas su keliais failais. Redaktorius leidžia vienu metu redaguoti kelis failus. Redaktoriaus lango apačioje pateikiamas atidarytų failų sąrašas. Norėdami redaguoti kitą failą aktyviname jo vardą atitinkančią piktogramą.

***m*-failų išsaugojimas.** Norėdami išsaugoti *m*-failą aktyviname *File* meniu eilutę *Save*. Jei *m*-failas yra naujas, tai *Save* dialogo lange reikia nurodyti failo vardą ir katalogą į kurį *m*-failas bus patalpintas.

Save As... meniu naudojamas, jei jau esančiam failui suteikiamas kitas vardas.

***m*-failai** yra dviejų tipų: *scenarijai* (script) ir *funkcijos* (function).

Scenarijus – tai seka komandų, atliekančių skaičiavimus su duomenimis iš darbo srities. Scenarijai neturi įėjimo ir išėjimo parametrų.

Funkcija – tai seka komandų, atliekančių skaičiavimus su duomenimis, kurie perduodami per įėjimo parametrus, ir rezultatus grąžinanti per išėjimo parametrus.

Kitaip tariant, tai savarankiška programa, atliekanti konkrečius skaičiavimus – konkrečią funkciją, ir su kuria bendraujama per formaliuosius (įėjimo ir išėjimo) parametrus.

2.8 Funkcijos

Programuojant dažnai tenka kartoti vienodų komandų, tačiau su skirtingais kintamaisiais, seką. Vienas iš būdų, leidžiančių išvengti tokių sekų pasikartojimo, yra funkcijos, realizuojančios tą komandų seką, sudarymas.

MATLAB'as turi labai vertingą savybę: jis leidžia vartotojui lanksčiai kurti savo funkcijas, kurios greta su MATLAB'o funkcijomis gali būti naudojamos įvairiuose skaičiavimuose. Skirtingai nuo daugelio kitų programavimo kalbų, MATLAB'e nėra skirtumo tarp procedūrų ir funkcijų. **Vartotojo sukurta funkcija saugoma m-faile, kurio vardas turi sutapti su funkcijos vardu.** Nuo scenarijaus (*script*), taip pat saugomo m-faile, funkcija skiriasi tuo, kad priima duomenų sąrašą ir grąžina rezultatų sąrašą. Funkcijos duomenų bei rezultatų sąrašų elementai atitinkamai vadinami įėjimo ir išėjimo formaliaisiais parametrais. Funkciją galima iškviesti tiek iš m-failo, tiek iš komandų lango. Kviečiant funkciją, vietoje formalių parametrų rašomi faktiniai parametrai, su kuriais procedūroje atliekami skaičiavimai. Funkcijos viduje naudojami kintamieji vadinami lokaliaisiais kintamaisiais ir jie galioja tik funkcijos viduje.

Duomenys funkcijai gali būti perduodami ne tik per įėjimo parametrus, bet ir per globaliuosius kintamuosius, kurie aprašomi komanda **global**.

Funkcijos pirmoji eilutė - antraštė užrašoma taip:

function [išėjimo parametrų sąrašas] = funkcijos_ vardas (įėjimo parametrų sąrašas);

Pavyzdžiui, **function [out_1, ..., out_m] = mkm(in_1, ..., in_n);**
čia

function	- raktinis žodis,
in_1, ..., in_n	- įėjimo parametrų – funkcijai perduodamų duomenų sąrašas,
out_1, ..., out_m	- išėjimo parametrų – funkcijos grąžinamų rezultatų sąrašas, rašomas tarp laužtinių skliaustų,
mkm	- funkcijos vardas.

Tiek įėjimo, tiek ir išėjimo parametrai vienas nuo kito skiriami kableliais.

Jei funkcija grąžina tik vieną parametą, pvz., **out**, tai funkcijos antraštės eilutė užrašoma taip: **function out = funkcijos_vardas(in_1, ..., in_n).**

Jei funkcija parametrų negrąžina (pavyzdžiui, skirta spausdinti), tai jo antraštė yra: **function funkcijos_vardas(in_1, ..., in_n).**

Po funkcijos antraštės paprastai rašomos viena arba kelios komentarų eilutės, paaiškinančios funkcijos paskirtį ir parametrus. Šios eilutės vadinamos įžanga arba prologu.

Pirmoji komentarų eilutė yra vadinama H1 eilute ir ji turi prasidėti funkcijos vardu. Tie komentarai, atspausdinami MATLAB'o komandų lange įvykdžius komandą: **help funkcijos_vardas**.

Funkcijos iškvietimas užrašomas taip:

[out_1, ..., out_m] = funkcijos_vardas(in_1, ..., in_n);

Jei funkcija grąžina vieną parametą arba jų negrąžina, atitinkamai užrašoma:

out = funkcijos_vardas(in_1, ..., in_m);

funkcijos_vardas(in_1, ..., in_m);

Funkcijos darbas baigiamas, kai atliekamos visos jos komandos arba sutinkama komanda **return**.

1.Pavyzdys. Tarkime, kad funkcijos pirmoji eilutė yra:
function out = mano_funkcija(x,y,z)

Ši eilutė aprašo funkciją vardu *mano_funkcija* su x , y ir z įėjimo parametrais. Funkcijos rezultatas priskiriamas kintamajam "out", esančiam funkcijos programoje. Šios funkcijos programos m -failo vardas turi būti "*mano_funkcija.m*", o pats failas turi būti saugomas MATLAB'o darbiname kataloge (arba, priešingu atveju, naudojant *path* komandą, turi būti nurodomas šios funkcijos radimo kelias).

Funkcija iškviečiama komandinės eilutės pagalba:
reikšmė = mano_funkcija(a,b,c)

Ši komanda reiškia, kad funkcijai, vardu *mano_funkcija*, perduodamos išorinių kintamųjų a , b , c reikšmės. Funkcijoje šios reikšmės suteikiamos vidiniams x , y , z kintamiesiems. Pažymėtina, kad jei x yra masivas, tai a taip pat turi būti masivas.

2.Pavyzdys. Sudarykime kvadratinės lygties $ax^2 + bx + c = 0$ realiųjų šaknų radimo funkciją.

Pasirinkime funkcijos vardą – *kvsaknys*. Aišku, kad funkcijos įėjimo parametrai bus kvadratinės lygties koeficientai a , b ir c , o išėjimo parametrai – realiųjų šaknų x_1 ir x_2 reikšmės ir požymis p . Parametro p reikšmė bus lygi 1, jei kvadratinės lygties šaknys yra realieji skaičiai, ir p reikšmė bus lygi 0, jei kvadratinė lygtis realiųjų šaknų neturi – jos šaknys yra kompleksiniai skaičiai.

Kvadratinės lygties realiųjų šaknų radimo funkcija kvsaknys

```
function [x1,x2,p] = kvsaknys(a,b,c)
% kvsaknys - funkcija, apskaičiuojanti kvadratinės lygties realiąsias šaknis,
% a, b, c - kvadratinės lygties koeficientai,
% x1, x2 - kvadratinės lygties realiosios šaknys,
% p - p=1, jei kvadratinės lygties šaknys yra realieji skaičiai,
%       p=0, jei kvadratinė lygtis realiųjų šaknų neturi.
```

```
d=b*b-4*a*c;
if d > 0
    t=sqrt(abs(d));
    x1=(-b+t)/(2*a); x2=(-b-t)/(2*a);
    p=1;
else
    p=0;
    disp('kvadratinė lygtis realiųjų šaknų neturi');
end
```

Funkcija feval. MATLAB'o įėjimo parametrai gali būti ne tik kintamieji, bet ir išorinių funkcijų vardai. Pavyzdžiui, norime sudaryti lygties $f(x)=0$ šaknies s , priklausančios intervalui (a,b) , tikslinimo pusiaukirtos metodu procedūrą – funkciją, vardu *pskrt*. Aišku, kad funkcija *pskrt* turi būti tokia, kad įgalintų spręsti bet kokią lygtį. Vadinasi, vienas jos įėjimo parametrų turėtų būti vardas funkcijos, apskaičiuojančios lygties kairiąją pusę. Todėl tikslinga imti tokius šios funkcijos įėjimo parametrus:

func – vardas funkcijos, apskaičiuojančios $f(x)$ – lygties kairiosios pusės reikšmę,
 a,b – šaknies s izoliacijos intervalo galai,
epsilon – norimas šaknies tikslumas,
maxit – didžiausias leistinas iteracijų – atkarpos dalijimo pusiau skaičius.

Išėjimo parametrai turėtų būti:

s - tikslumu *epsilon* apskaičiuota lygties $f(x)=0$ šaknies reikšmė,

it - atliktų iteracijų skaičius.

Kitaip tariant, funkcijos antraštė būtų tokia:

function [*s,it*]=*pskrt*(*func,a,b, epsilon,maxit*);

Procedūros *pskrt* kūne (žr. 3.1.2 paragrafą), norėdami apskaičiuoti $f(x)$ reikšmę taške $x=a$, naudosisime standartinę MATLAB'o funkciją *feval* ir šį veiksmą užrašysime $fa=feval(func,a)$, čia $fa=f(a)$.

Funkcijos *feval* įėjimo parametrai yra funkcijos *pskrt* įėjimo parametro funkcijos *func* vardas ir argumento, kuriame norime apskaičiuoti šią funkciją, reikšmė.

Bendru atveju, kreipinys į funkciją *feval* užrašomas taip:

feval(*fun,p1,p2,p3,...*),

čia *fun* – simbolinė eilutė, reiškianti funkcijos vardą,

p1,p2,p3,... – parametrai, reikalingi apskaičiuoti funkciją *fun*.

Pavyzdžiui, komandos

feval ('sin',pi/4) ir *sin*(pi/4) yra ekvivalenčios.

Kreipiantis į funkciją *pskrt*, vietoje formalaus parametro *func* tarp apostrofų įrašysime faktinį išorinės funkcijos, apskaičiuojančios $f(x)$ reikšmę, vardą. Pavyzdžiui, jei išorinės funkcijos, apskaičiuojančios $f(x)$ reikšmę, antraštė yra

function *fv* = *fx1*(*x*); ,

tai kreipinio į funkciją *pskrt* komanda būtų tokia:

[*s,it*]=*pskrt*('fx1',*a,b, epsilon,maxit*); .

Aišku, kad prieš kreipinį, parametrų *a, b, epsilon, maxit* reikšmės turi būti apibrėžtos.

Funkcija inline. MATLAB'o 5-je ir aukštesnėse versijose yra palaikomas objektiškai orientuotas programavimas. Viena iš objektiškai orientuoto programavimo patogių savybių yra **inline funkcijos objektas** (toliau *inline* funkcija).

Funkcija *inline* lengvai sudaroma ir įgalina išvengti labai trumpų, anksčiau nagrinėtų *m*-failo funkcijų, kurios skirtos apskaičiuoti paprastas išraiškas arba formules. Išraiškoje yra vienas ar keli kintamieji, kurie suprantami kaip įėjimo parametrai.

Inline funkcija užrašoma taip:

funkcijos_vardas = **inline**('išraiška','p1','p2', ... , 'pn'),

čia

funkcijos_vardas - *inline* funkcijos vardas – kintamasis,

inline - raktinis žodis ,

išraiška - formulė, pagal kurią atliekami skaičiavimai,

p1,p2, ... , pn - įėjimo parametrai – išraiškos kintamieji.

Tiek išraiška, tiek ir įėjimo parametrai rašomi tarp apostrofų ir vienas nuo kito skiriami kableliais.

Pavyzdys. Sudarykime **inline** funkciją, kuri atliktų skaičiavimus pagal formulę

$f = \sin(x) * y$, čia x ir y gali būti masyvai.

Norima *inline* funkcija bus:

f = *inline*('sin(x).*y','x','y');

Ši *inline* funkcija yra analogiška komandai: $f=\sin(x).*y$. Skirtumas yra tas, kad pagal šią formulę atliekame skaičiavimą tik su x ir y fiksuotomis reikšmėmis, o į *inline* funkciją galime kreiptis su įvairiais faktiniais įėjimo parametrais.

Kreipinys į *inline* funkciją yra:

kintamasis = *funkcijos_vardas*(*išraiškos argumentai*);

Pavyzdžiui, norėdami apskaičiuoti $r = \sin(\pi/4) * 5$, remdamiesi sukurta *inline* funkcija f , rašysime: $r = f(\pi/4, 5)$, o norėdami apskaičiuoti $a = \sin(z) * t$, čia z ir t , pvz., yra vienodo formato masyvai, rašysime: $a = f(z, t)$.

Inline funkcija skiriasi nuo anksčiau išnagrinėtų *m*-failo funkcijų pirmiausia tuo, kad *inline* funkcija galioja tik programoje, kurioje ji buvo sudaryta. Kitaip tariant, ***inline funkcijas, analogiškai kitoms aukšto lygio programavimo kalboms, galima laikyti vidinėmis funkcijomis.***

Kitas *inline* funkcijų skirtumas nuo *m*-failų funkcijų yra tas, kad *inline* funkcijos rezultatas gali būti naudojamas kaip kitų funkcijų įėjimo parametras.

Pavyzdžiui, tarkime, kad x ir y yra to paties formato vektoriai ir norime apskaičiuoti sumą: $s = \sum_{i=1}^n \sin(x_i) * y_i$, čia n – vektoriaus x elementų skaičius. Tada, pasinaudodami sukurta *inline* funkcija f , sumą s apskaičiuosime komandos $s = \text{sum}(f(x, y))$; pagalba.

Pirminės ir antrinės funkcijos. MATLAB'o 5-je ir aukštesnėse versijose viename *m*-faile galima patalpinti kelias funkcijas: pirmoji funkcija yra ***pirminė*** ir *m*-failo vardas sutampa su ***šios funkcijos*** vardu, o kitos funkcijos, patalpintos po pirminės funkcijos, vadinamos ***antrinėmis arba subfunkcijomis.***

Vien ***tik pirminė funkcija*** gali būti iškviesta komandiniame lange arba ją gali iškviesti ***kitų m-failų*** funkcijos.

Antrinės funkcijas gali iškviesti tiek pirminė, tiek ir antrinės ***to paties m-failo*** funkcijos.

Antrinių funkcijų vartojimas turi keletą privalumų.

1. Giminingų funkcijų patalpinimas viename *m*-faile sumažina programos bendrą *m*-failų skaičių.
2. Kadangi antrinės funkcijas gali naudoti tik funkcijos, esančios tame pačiame *m*-faile, tai antrinių funkcijų vardai gali būti vartojami kitų *m*-failų funkcijoms pavadinti. Kitaip tariant, tą patį antrinės funkcijos vardą gali naudoti skirtingos pirminės funkcijos.
3. Antrinės funkcijos leidžia viename *m*-faile kompaktiškiau sukaupti didesnę programos bloką. Tas įgalina lengviau skaityti ir derinti pirminę funkciją.

Pavyzdys. Sudarykime funkciją, kuri įgalintų apskaičiuoti duotų skritulių plotus bei jų apskritimų ilgius, kai žinomi tų apskritimų spinduliai.

```
function [q, p]=skritulys(x);
% SKRITULYS, tai funkcija, apskaičiuojanti
% skritulių plotus ir jų apskritimų ilgius.
%
% Formalus parametrai
% x - duotų apskritimų spindulių masyvas,
% q - skritulių plotų masyvas,
% p - apskritimų ilgių masyvas.

i=0;
for r=x
    i=i+1;
    q(i)=plotas(r);
    p(i)=ilgis(r);
end
```

```
% Antrine funkcija plotas
function q = plotas(r);
% Apskaičiuojamas r spindulio
% skritulio plotas
q=pi*r^2;

% Antrine funkcija ilgis
function p= ilgis(r);
% Apskaičiuojamas r spindulio
% apskritimo ilgis
p=2*pi*r;
```

Komandų lange užrašę komandas:

```
x=[1 4 8 10]
[q, p]=skritulys(x) ,
turėsime rezultatą:
x = 1 4 8 10
q = 3.1416 50.2655 201.0619 314.1593
p = 6.2832 25.1327 50.2655 62.8319
```

Funkcija eval. MATLAB'o funkcijos *eval* antraštė yra:

eval('simbolių_eilutė').

Ši funkcija skirta apskaičiuoti išraiškas, aprašytas simbolių eilute. Jos įėjimo parametras yra simbolių eilutė: '*simbolių_eilutė*', kuri vykdoma kaip MATLAB'o komanda.

Pavyzdžiui, tai galima iliustruoti tokių simbolių eilučių išraiškų apskaičiavimu:

```
>>str='2+3'
str = 2+3
>>eval (str)
ans = 5
>> eval('sin(pi*30/180)') %Apskaičiuoja sin(30°)
ans = 0.5000
>> str = 'v = 1:5'
str =
v = 1:5
>> eval(str)
v = 1 2 3 4 5
```

Naudojant funkciją *eval*, patogiu suformuoti skirtingų vardų ir formatų vektorių ar matricių rinkinį. Tarkime, reikia suformuoti tokius vektorius $v_i = 1, 2, \dots, i$, kai $i=1,2,\dots,10$. Šiuos vektorius galime suformuoti su tokia komandų seka:

```
v1 = 1;
v2 = 1:2;
v3 = 1:3;
.....
v10 = 1:10;
```

Tačiau komanda *eval* čia puikiai padeda: ji įgalina minėtus vektorius suformuoti trumpiau.

```
clear
for i = 1:10
str = ['v' int2str(i) ' = 1:i;'];
eval(str)
end
```

Šis programos fragmentas sukurs norimus masyvus: v_1, \dots, v_{10} . Tuo galime įsitikinti komandos *whos* pagalba:

```
>> whos
Name      Size      Bytes  Class
v1         1x1         8     double
v2         1x2         8     double
v3         1x3        16     double
.....
v10        1x10       80     double
```


i	1x1	8	double array
str	1x10	20	char array
v1	1x1	8	double array
v10	1x10	80	double array
v2	1x2	16	double array
v3	1x3	24	double array
v4	1x4	32	double array
v5	1x5	40	double array
v6	1x6	48	double array
v7	1x7	56	double array
v8	1x8	64	double array
v9	1x9	72	double array

Grand total is 66 elements using 468 bytes

Pavyzdžiui, vektoriaus v6 elementai įgijo tokias reikšmes:

```
>> v6
v6 = 1      2      3      4      5      6
```

Panagrinėkime kitą pavyzdį. Tegul reikia suformuoti kvadratinės vienetinės matricos $VIENETINE_{ii}(i,i)$, $i=1,2,3$. Matricas suformuos tokia komandų seka.

```
>> for n = 1:3
eval(['VIENETINE' int2str(n) int2str(n) ' = eye(n,n)'])
end
VIENETINE11 = 1
VIENETINE22 = 1      0
              0      1
VIENETINE33 = 1      0      0
              0      1      0
              0      0      1
```

2.9 Duomenų failai: duomenų skaitymas ir rašymas

Duomenų įvedimas iš klaviatūros. Jei įvedamų duomenų yra nedaug, tai juos patogų įvesti iš klaviatūros funkcijos **input** pagalba.

Funkcijos **input** įėjimo parametras yra simbolių eilutė, kuri sufleruoja, kokio kintamojo reikšmę reikia įvesti. Pagal nutylėjimą **input** funkcija grąžina skaitinę reikšmę: skaičių, vektorių, matricą ir pan. Pavyzdžiui, komanda

```
>> x=input('įveskite matricą x')
```

komandiniame lauke atspausdina pranešimą

```
įveskite matricą x.
```

Tarkim, iš klaviatūros surinkus: [1 2 3; 4 5 6], ekrane turėsime pranešimą:

```
x =
     1     2     3
     4     5     6.
```

Norėdami iš klaviatūros įvesti simbolių eilutę, **input** funkcijai reikia nurodyti du įėjimo parametrus: **sufleruojančia simbolių eilutę ir po kablelio simbolį 's'**, kuris reiškia, kad įvedama informacija yra simbolių eilutė. Pavyzdžiui, komanda

```
>> vardas=input('įveskite savo vardą','s')
```

komandiniame lauke atspausdina pranešimą

```
įveskite savo vardą.
```

Pavyzdžiui, iš klaviatūros surinkus: Jonas, ekrane turėsime pranešimą:

```
vardas = Jonas
```

Duomenų spausdinimas ekrane. Mažą informacijos kiekį arba trumpus pranešimus patogų spausdinti ekrane. Tam tikslui naudojama funkcija **disp**.

Funkcija **disp** turi tik vieną įėjimo parametą, kuris gali būti arba simbolinė, arba skaitinė matrica.

Norint atspausdinti trumpą pranešimą, jį turime užrašyti tarp apostrofų. Pavyzdžiui, komanda

```
>> disp('ivedami duomenys yra klaidingi')
```

ekrane atspausdins

```
ivedami duomenys yra klaidingi.
```

Šalia tekstinio pranešimo dažnai reikia užrašyti skaitinę arba simbolinę reikšmę. Kadangi *disp* funkcija turi tik vieną įėjimo parametą, tai *pranešimą ir kintamojo reikšmę* reikia apjungti į vieną eilutę. Pavyzdžiui, jei kintamojo vardas reikšmė yra simbolinė eilutė Jonas, tai komanda

```
>> disp(['jūsų vardas yra ',vardas])
```

ekrane atspausdins

```
jūsų vardas yra Jonas.
```

Tarkime, kad kintamojo sak reikšmė yra skaičius 2. Tada komanda:

```
>> disp(['šaknies reikšmė =', num2str(sak)])
```

ekrane atspausdins:

```
šaknies reikšmė =2.
```

Pastaba. Funkcija *num2str* yra paaiškinta 2.15 lentelėje. Ji realųjį skaičių verčia simbolių eilute.

Neformatuotas duomenų skaitymas ir rašymas. Skaitant duomenis iš failo reikia prisiminti, kad MATLAB'e veiksmai atliekami tik su matricomis. Tai reiškia, kad duomenų faile visi įrašai turi turėti vienodą laukų skaičių, pvz., visos eilutės turi turėti vienodą stulpelių skaičių.

Dažniausiai tenka dirbti su tekstinio formato (ASCII) failais. Pavyzdžiui, jei norimas nuskaityti duomenų rinkinys yra faile vardu *points.dat*, jis nuskaitymas komanda:

```
load points.dat
```

Nuskaityti duomenys priskiriami MATLAB'o kintamajam *points*. Tegul faile *points.dat* yra du duomenų stulpeliai: pirmajame stulpelyje yra aukščių reikšmės, o antrajame – atstumai. Tada *load.dat* komandos rezultatas bus :

```
points = 1.0200  2.6500  
         3.5400  4.2600  
         5.6800  6.7700,
```

o komandų - *aukstis* = *points(:,1)*

```
aukstis = 1.0200  
         3.5400  
         5.6800
```

atstumas = *points(:,2)*

```
atstumas = 2.6500  
         4.2600  
         6.7700.
```

Norint įrašyti šiuos duomenis į ASCII formato failą vardu *duomenys.dat*, reikia surinkti:

```
save duomenys.dat aukstis atstumas -ascii.
```

Tada failo *duomenys.dat* turinys bus:

```
duomenys = 1.0200  
         3.5400  
         5.6800  
         2.6500  
         4.2600  
         6.7700.
```

Formatuotas duomenų skaitymas. Norint nuskaityti duomenų failą, kuris yra sukurtas kita programine įranga nei MATLAB^{as}, tenka parašyti funkciją arba programą (*m-failą*), kuri tai atliktų. Tarkime, kad turime tokį duomenų failą:

```
10/06 11:18:00 -34.855 151.3057 216.4 70.91 -61.23 0.29
10/06 11:18:01 -34.85554 151.30649 214.8 71.38 -60.8 -0.88
10/06 11:18:02 -34.85609 151.30727 212.7 71.86 -60.64 -1.64
10/06 11:18:03 -34.85664 151.30807 210.8 72.4 -60.35 -1.67
10/06 11:18:04 -34.85717 151.30887 209.7 72.83 -60.06 -1.33.
```

Pirmajame duomenų failo lauke (stulpelyje) yra data su slešo skyrikliu, antrajame – laikas su dvitaškio skyrikliu, o likusiuose šešiuose – skaičiai, atskirti tarpais. Šie duomenys nuskaitomi iš failo naudojama *fscanf* funkcija. Tarkime šio failo vardas yra *rezultatai.dat*.

Pirmiausia šis failas *fopen* komanda atidaromas skaitymui:

```
fid = fopen('rezultatai.dat');
```

Komanda *fopen* grąžina failo identifikatorių *fid*, kuris įgyja sveiką skaičiaus reikšmę. Šis failo identifikatorius susietas su failo vardu *rezultatai.dat* ir jį visuomet būtina nurodyti kaip pirmąjį argumentą failo skaitymo ar rašymo funkcijose. Jei failo neįmanoma atidaryti, t.y. nurodytu vardu failas neegzistuoja arba MATLAB^{as} jo neranda kataloge, grąžinama *fid* = -1 reikšmė. Jei failas atidarytas sėkmingai, jo turinys nuskaitomas tokia komanda:

```
a = fscanf(fid, '%d/%d %d:%d %d %g%g%g%g%g%g');
```

Komanda *fscanf* nuskaitė visus faile esančius duomenis:

```
size(a)
ans = 55 1
```

Failo *rezultatai.dat* įrašas (eilutė) turi 11 skaitinių reikšmių: 2 reikšmės - žymi datą, 3- laiką ir 6 – likusią informaciją. Kadangi failas turi 5 įrašus (eilutes), tai viso turime 55 skaičių reikšmes, patalpintas vektoriuje – stulpelyje, kurio vardas *a*. Formato eilutė '%d/%d %d:%d %d %g%g%g%g%g%g', kuria nuskaityti failo duomenys, paaiškinama taip: „*skaityk du dešimtainius skaičius atskirtus slešu, praleisk keletą tarpų, skaityk tris dešimtainius skaičius atskirtus dvitaškiais, praleisk keletą tarpų, skaityk šešis slankaus kablelio formatu pateiktus skaičius*“. Plačiau apie komandą *fscanf* galima sužinoti MATLAB^o pagalbos žinyne. Kad vektorius *a* įgautų failo *rezultatai.dat* struktūrą, jį reikia pertvarkyti į matricą, turinčią 11 stulpelių. Tam tikslui rašome komandas:

```
N = length(a)/11; a = reshape(a,11,N);
```

Reikia pažymėti, kad MATLAB^{as} duomenis, pvz. matricą, iš failo skaito eilutėmis (įrašais) ir juos nuosekliai priskiria vektoriaus elementams. Kad nuskaityto iš failo duomenų struktūra atitiktų faile esančią duomenų struktūrą, funkcijos *reshape* pagalba pertvarkant vektorius *a* į matricą *a*, pastarąją dar reikia ir transponuoti (transponavimo veiksmą nusako simbolis *'*). Patikriname, ar tikrai matrica *a* atitinka pradinių duomenų struktūrą.

Datos ir laiko reikšmės yra pirmuose penkiuose matricos stulpeliuose:

```
a(:,1:5)
ans = 10 6 11 18 0
      10 6 11 18 1
      10 6 11 18 2
      10 6 11 18 3
      10 6 11 18 4.
```

Likusios reikšmės:

```

a(:,6:11)
ans =
-34.8550 151.3057 216.4000 70.9100 -61.2300 0.2900
-34.8555 151.3065 214.8000 71.3800 -60.8000 -0.8800
-34.8561 151.3073 212.7000 71.8600 -60.6400 -1.6400
-34.8566 151.3081 210.8000 72.4000 -60.3500 -1.6700
-34.8572 151.3089 209.7000 72.8300 -60.0600 -1.3300.

```

Jei failas *rezultatai.dat* tolesniuose skaičiavimuose nebus naudojamas, jis uždaromas komanda *fclose*: *fclose(fid)*

Formatuotas duomenų rašymas. Norint į failą rašyti formatuotus duomenis, naudojama *fprintf* komanda.

Tarkime, reikia sukurti tekstinį failą *exp.txt* ir į jį įrašyti eksponentinės funkcijos e^x , kai x kinta nuo nulio žingsniu 0.2 iki vieneto, reikšmių lentelę. Norimą failą formuoja tokia komandų seka:

```

x = 0:0.2:1;
y = [x; exp(x)];
fid = fopen('exp.txt', 'w');
fprintf(fid, '%6.2f %12.8f\n', y);
fclose(fid)

```

Šių komandų rezultatas - tekstinis failas vardu *exp.txt*, kuriame yra eksponentinės funkcijos reikšmių lentelė:

```

0.00 1.00000000
0.20 1.22140276
0.40 1.49182470
0.60 1.82211880
0.80 2.22554093
1.00 2.71828183.

```

Bendru atveju aukščiau naudotos komandos turi tokį pavidalą:

failo_identifikatorius=fopen('failo_vardas', veiksmas).

Dažniausiai naudojami veiksmai:

- 'r' – skaitymas (numatytasis veiksmas),
- 'w' – rašymas (jei reikia, failas sukuriamas),
- 'a' – papildymas (jei reikia, failas sukuriamas).

Duomenų skaitymą iš failo nurodytu formatu atlieka tokia komanda:

fscanf(failo_identifikatorius, 'formatas', kintamieji).

Duomenų rašymą į failą nurodytu formatu atlieka komanda:

fprintf(failo_identifikatorius, 'formatas', kintamieji) .

Failo identifikatorius – skaičius, gaunamas įvykdžius *fopen* komandą.

Jei failo identifikatorius praleistas arba jo reikšmė lygi 1, tai spausdinama į ekraną.

Darbo pradžioje komandos *fopen* pagalba failas turi būti atvertas rašymui, o darbo pabaigoje – komanda *fclose* - užvertas.

'*formateas*' žymi vietą, kurioje rašomas norimas tekstas, eilutės valdymo simboliai, pertvarkymo specifikatoriai bei kiti simboliai.

Dažniausiai naudojami valdymo simboliai:

- \n – perėjimas į kitą eilutę,
- \t – tabuliacijos įvedimas.

Populiariausi pertvarkymo specifikatoriai:
 $\%d$ – dešimtainis formatas su ženklu;
 $\%f$ – formatas skaičiams su fiksuotu tašku;
 $\%e$ – formatas su dešimtainiu daugikliu;
 $\%s$ – simbolių eilutė.
 $\%nd$ – n sveikasis teigiamas skaičius, nurodantis ilgį lauko, kuris skiriamas spausdinamajam elementui, pavyzdžiui, $\%3d$.
 $\%n.mf$ – n nurodo visą lauko ilgį, o
 m – skaitmenų po kablelio skaičių, pavyzdžiui, $\%8.2f$.
Komanda: *fclose(failo_identifikatorius)* uždaro atvertą failą.

Panagrinėkime kelis *fprintf* komandos panaudojimo pavyzdžius.

1. Komanda

*fprintf('Vienetinio apskritimo perimetras %g.\n', 2*pi)*
ekrane atspausdins eilutę: *Vienetinio apskritimo perimetras 6.283186.*

2. Teksto eilutėje norint patalpinti apostrofo simbolį, jį reikia įvesti du kartus.

Komanda
fprintf(1, 'It''s Friday.\n') ekrane pateiks: *It's Friday.*

3. Komandos

$B = [8.8 \ 7.7; 8800 \ 7700]$
fprintf(1, 'X yra %6.2f metrai arba %8.3f mm\n', 9.9, 9900, B)
ekrane atspausdins:
X yra 9.90 metrai arba 9900.000 mm
X yra 8.80 metrai arba 8800.000 mm
X yra 7.70 metrai arba 7700.000 mm

2.10 MATLAB'o grafika

Grafinės funkcijos - vienos iš svarbiausių MATLAB'o funkcijų. Pagrindinės MATLAB'o grafikų braižymo funkcijos skirtos tiek atskiros kreivės $y=f(x)$, tiek ir kelių kreivių $y_1=f(x_1)$, $y_2=f(x_2)$ ir t.t. grafikų vaizdavimui viename grafiniame lange. Dviejų nepriklausomų kintamųjų funkcijos $z=f(x,y)$ grafiškai atvaizduojamos arba funkcijos vienodų reikšmių kontūrais (lygio linijomis), arba paviršiumi.

2.10.1. Vieno kintamojo funkcijų grafinis vaizdavimas

MATLAB'o funkcijos, skirtos kreivių grafikų braižymui, lengvai įsisavinamos - tereikia nurodyti argumentų x ir jas atitinkančių funkcijos reikšmių $y=f(x)$ vektorius.

Pavyzdžiui, komandų seka:

$x = \dots$ % *apibrėžiamas vektorius x*
 $y = \dots$ % *apibrėžiamas vektorius y*
plot(x,y) % *brėžiamas funkcijos grafikas*
išpieš funkcijos $y = f(x)$ grafiką.

Funkcijos grafikas, tai nuosekliai atkarpomis sujungtų taškų $((x_i, y_i), i = \overline{1, n})$, čia n vektoriaus x elementų skaičius, visuma.

Apibendrintoji funkcijos *plot* išraiška yra

plot(xduomenys, yduomenys, simbolis, p1, p2, ...), čia

xduomenys, yduomenys – argumento ir funkcijos reikšmių to paties formato masyvai;

„*simbolis*“ – simbolinis kintamasis (bendru atveju nebūtinai). Simbolinis kintamasis, gali turėti 3 specialius simbolius, nurodančius linijos tipą, taško tipą bei linijos spalvą. Galima nurodyti tik kai kurias iš šių charakteristikų.

Jei parametro „*simbolis*“ nėra, tai grafiko linija ištisinė, grafikas vaizduojamas pikseliais, o spalva kreivėms parenkama tokia tvarka: mėlyna, žalia, raudona, žydra, violetinė, geltona, juoda, balta;

p_1, p_2, \dots – parametrai (bendru atveju nebūtinai), aprašantys kitas grafinio vaizdo charakteristikas, pavyzdžiui, linijos storį, taško formos matmenys bei jo spalvą ir panašiai.

2.16 lentelėje pateikti galimi linijų tipai, spalvos ir taškai, bei nurodyti juos aprašantys simboliai.

2.16 lentelė. Linijos charakteristikos

Linijos tipas	Spalva	Taško forma
- ištisinė	y geltona	. taškas
: taškinė	m violetinė	o apskritimas
-- brūkšninė	c žydra	x x-žymė
-. taškinė-brūkšninė	r raudona	+ plius
	g žalia	* žvaigždutė
	b mėlyna	p penkiakampis
	w balta	d rombas
	k juoda	s kvadratas
		h šešiakampis
		^ rodyklė į viršų
		> rodyklė dešinė
		< rodyklė kairė

Kiekvienas naujas *plot* komandos vykdymas ištrina prieš tai buvusį grafinį vaizdą. Tarkime norime išbrėžti tiesinės funkcijos $y=3x$ grafiką, kai $x \in [0,100]$. Tam tikslui sukuriame *m*-failą:

```
x = 0:0.1:100;
y = 3*x;
plot(x,y)
```

Nubrėžtas grafikas pavaizduotas 2.1 paveiksle.

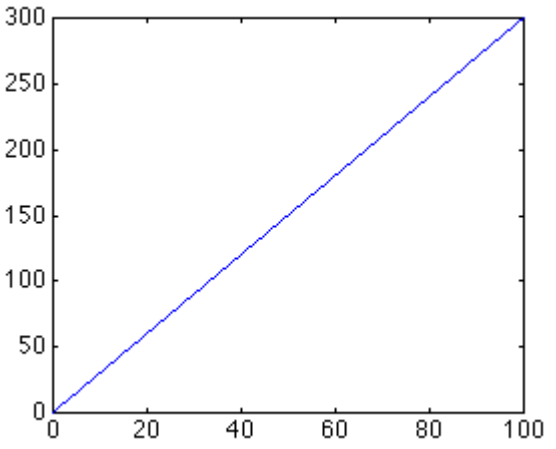
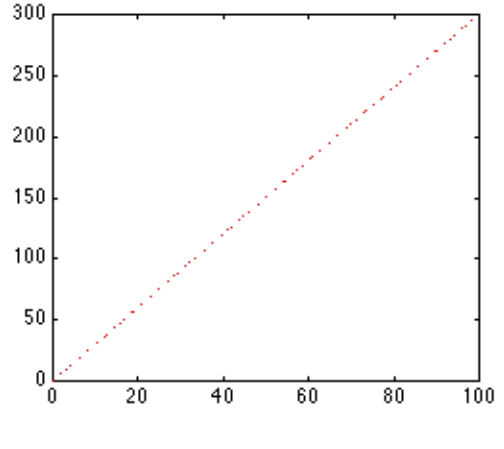
Naudojant *plot* komandą reikia turėti omenyje, tai, kad vektoriai *x* ir *y* turi turėti tą patį elementų skaičių. Priešingu atveju MATLAB'as duos klaidos pranešimą.

Grafinis apipavidalinimas. Grafiko spalva ir grafiko taškų vietą žymintys simboliai brėžinyje aprašomi trečiu *plot* komandos parametru – „*simbolis*“, kuris rašomas tarp apostrofų, ir, kaip buvo minėta anksčiau, gali turėti 3 specialius simbolius.

Pavyzdžiui, norint nubrėžti prieš tai pateiktos funkcijos grafiką raudona taškine linija, *m*-failą tektų pertvarkyti:

```
x = 0:0.1:100;
y = 3*x;
plot(x,y,'r:')
```

Grafikas pavaizduotas 2.2 paveiksle.

	
<p>2.1 pav. Funkcijos $y=3x$ grafikas, kai grafiko linija – ištisinė</p>	<p>2.2 pav. Funkcijos $y=3x$ grafikas, kai grafiko linija yra raudona ir taškinė.</p>

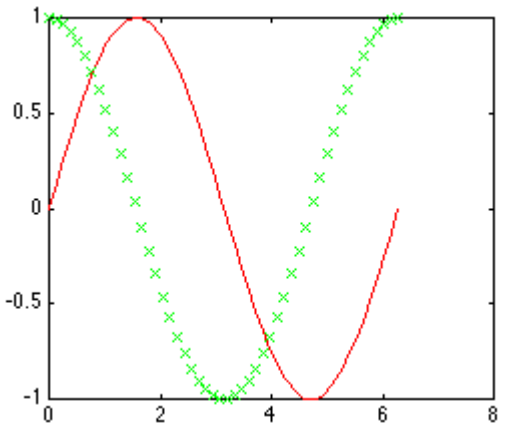
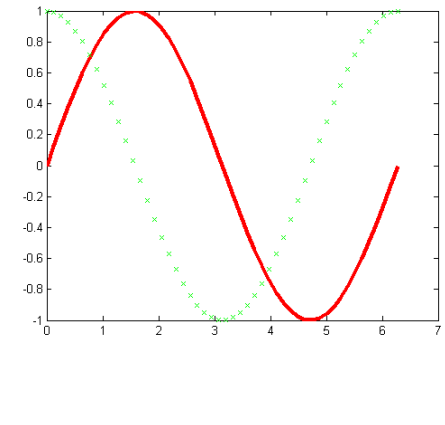
Galimi grafiko spalvų, linijų ir taškų formų, tipai pavaizduoti anksčiau patalpintoje 2.14 lentelėje.

Ketvirtasis *plot* komandos parametras *pl* skirtas linijos storiui. Šio parametro vardas yra simbolių eilutė '*LineWidth*', po kurios eina sveikasis skaičius, nusakantis linijos storį pikseliais.

Viename grafiniame lange (tose pačiose koordinačių ašyse) galima išbrėžti kelių funkcijų grafikus. Tarkime, viename grafiniame lange reikia pavaizduoti skirtingomis spalvomis ir linijų tipais sinuso ir kosinuso grafikus. Tai atliks komandų seka:

```
x = linspace(0,2*pi,50);
y = sin(x); z = cos(x);
plot(x,y,'r', x,z,'gx')
```

Brėžinyje sinuso grafikas pateiktas ištisine linija, o kosinuso – sudarytas iš „x“ simbolių (žiūrėk 2.3 paveikslą).

	
<p>2.3 pav. Funkcijų $y=\sin(x)$ ir $y=\cos(x)$ grafikai</p>	<p>2.4 pav. Skirtingo storio funkcijų $y=\sin(x)$ ir $y=\cos(x)$ grafikai</p>

Tame pačiame grafiniame lange galima nubraižyti kiek norima daug funkcijų grafikų. Tereikia *plot* komandoje aprašyti jų parametrų rinkinius. Šiuo atveju patogiu

funkcijų grafikus pavaizduoti skirtingomis spalvomis ir linijų tipais. Tą patį rezultatą galima gauti naudojant *hold on* ir *hold off* komandas. Komandų seka

```
x = linspace(0,2*pi,50);
y = sin(x);
hold on
plot(x,y,'r','LineWidth',3)
z = cos(x);
plot(x,z,'gx')
hold off
```

toje pačioje koordinatinėje plokštumoje išpieš $y=\sin(x)$ ir $y=\cos(x)$ grafikus. Be to čia pirmoji kreivė (sinusinė) pavaizduota storesne linija (žiūrėk 2.4 paveikslą).

Reikia turėti omenyje, kad visi grafikai, brėžiami po komandos *hold on*, pateikiami toje pačioje koordinatinių plokštumoje, nenaikinant jame esančių grafikų. Ankstesnieji grafikai, esantys grafiniame lange, naikinami komandos *hold off* pagalba.

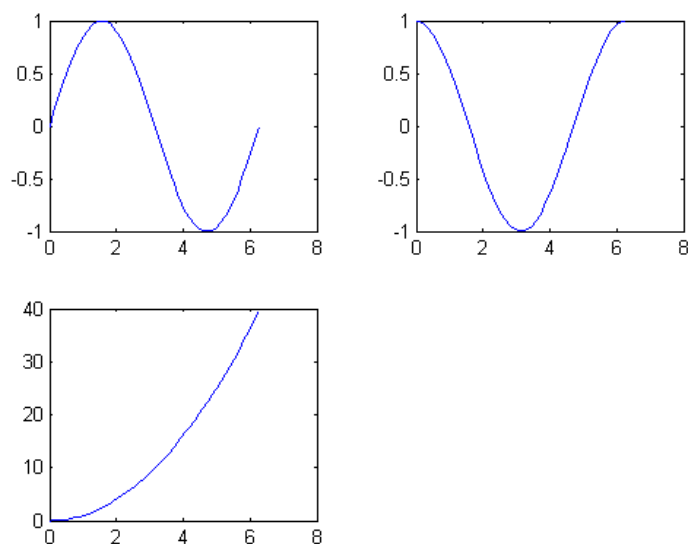
Komanda *grid on* koordinatinių plokštumą padengia stačiakampiu tinkleliu, o komanda *grid off* – tinklelį naikina. Nenurodžius *grid* komandos MATLAB'o numatytoji komanda yra *grid off*.

Grafinio lango skaidymas. Kartais norima viename grafiniame lange pavaizduoti keletą grafikų skirtingose koordinatinių ašyse. Tam skirta komanda *subplot*, kuri rašoma prieš grafikų braižymo komandas.

Komanda *subplot(m,n,p)* suskaido grafinį langą į m eilučių (horizontalių polangių) ir n stulpelių (vertikalių polangių) matricą. Tai leidžia viename grafiniame lange pavaizduoti $m*n$ brėžinių. Brėžinio vietą lange nusako kintamasis p . Tarkime, tame pat lange, bet skirtinguose brėžiniuose norime pavaizduoti sinuso, kosinuso ir parabolės grafikus. Tai atliks toks scenarijus:

```
x = linspace(0,2*pi,50);
y = sin(x); z = cos(x); w = x.*x;
subplot(2,2,1), plot(x,y)
subplot(2,2,2), plot(x,z)
subplot(2,2,3), plot(x,w)
```

Funkcijų grafikai pavaizduoti 2.5 paveiksle.



2.5 pav. Funkcijų $y=\sin(x)$, $y=\cos(x)$ ir $y=x^2$ grafikai

Kaip matyti, pateikiami tik trys brėžiniai, nors lango suskaidymo matrica yra

2 x 2, t.y. skirta 4 brėžiniams. Tai reiškia, kad ne visi lango suskaidymo matricos elementai turi būti užpildyti. Pirmasis brėžinys yra viršutiniame kairiajame lango kampe, kitas - dešiniau. Užpildžius eilutės visus stulpelius, ta pačia tvarka pildomi žemiau esančios eilutės stulpeliai. Lango brėžiniai numeruojami iš eilės einančiais natūraliaisiais skaičiais eilutė po eilutės iš kairės į dešinę, pradedant pirmąja eilute.

Jei programoje yra daugiau *plot* komandų, nei lange brėžiniams išskirta vietos, tai grafikai brėžinių, kurių eilės numeriai didesni už langelių skaičių, nuosekliai talpinami vienas ant kito paskutiniame langelyje. Aišku, kad jame matysis tik paskutinis brėžinys, - kiti bus panaikinti.

Ši problema išsprendžiama atidarant naują grafinį langą komandos *figure* pagalba.

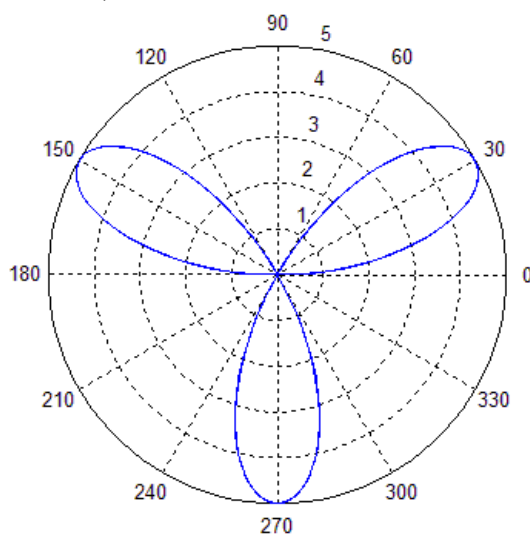
Reikia pabrėžti, kad MATLAB'as turi funkciją, įgalinančią brėžti grafikus polinėje koordinačių plokštumoje (funkcija **polar**), o taip eilę specializuotų brėžimo funkcijų, skirtų vaizduoti kompleksinius skaičius (funkcijos **compass**, **feather**), histogramoms (funkcijos **hist**) ir kitas (pvz., **semilogx**, **loglog**, **area**, **fill**, **set** ir kt.). Plačiau apie šias funkcijas galima sužinoti surinkus komandą: *help funkcijos vardas*.

Pavyzdys. Išbrėžkime funkcijos $\rho = 5 \sin(3\varphi)$, $\varphi \in [0, 2\pi]$ grafiką.

Komandų lauke surinkus komandas:

```
t=linspace(0,2*pi,200);
r=5*sin(3*t);
polar(t,r)
```

turėsime funkcijos $\rho = 5 \sin(3\varphi)$ grafiką polinėje koordinačių sistemoje (žr. 2.6 paveikslą).



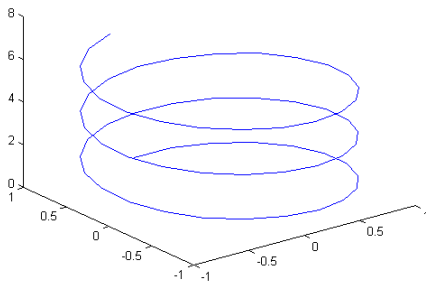
2.6 pav. Funkcijos $\rho = 5 \sin(3\varphi)$ grafiką polinėje koordinačių sistemoje

3D grafika. Erdviniai grafikai brėžiami komanda *plot3*, kuri yra analogiška komandai *plot*.

Panagrinėkime pavyzdį, - komandų seką:

```
x = 0:0.1:2*pi;
y = cos(3*x);
z = sin(3*x);
plot3(z,y,x)
```

Šios komandos išbrėš erdvinę apskritiminę spiralę (2.7 paveikslas).



2.7 pav. Erdvinė apskritiminė spiralė

Gautąjį erdvinės spiralės grafinį vaizdą galima pasukti, aktyvuojant koordinatinių ašių pasukimo mygtuką, esantį grafinio lango įrankių juostoje. Po to belieka žymeklį patalpinti grafiniame lange ir, paspaudus pelės mygtuką, pelę paslinkti. Tai leidžia grafinį vaizdą matyti esant skirtingoms koordinatinių ašių padėtimis.

Kartais tenka viename grafiniame lange pavaizduoti kelių funkcijų grafikus, kurių skaitinės reikšmės skiriasi skaičių eilėmis. Šiuo atveju patogiau naudoti MATLAB'o funkciją *plotyy*, kuri sukuria dvi funkcijos reikšmių ašis, kairėje ir dešinėje. Funkcijos parametrai analogiški funkcijos *plot* parametrų.

Panagrinėkime pavyzdį:

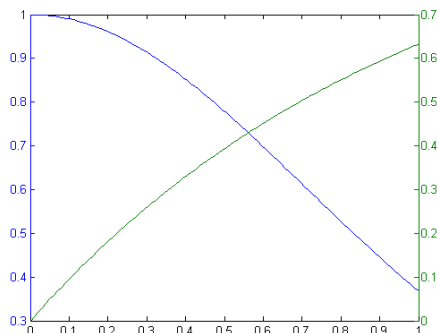
```
f = inline('exp(-x.^2)'); % vidinė funkcija, apskaičiuojanti  $y = e^{-x^2}$ 
g = inline('1 - exp(-x)'); % vidinė funkcija, apskaičiuojanti  $y = 1 - e^{-x}$ 
x = linspace(0,1); %  $x=0:0.01:1$ 
plotyy(x,f(x),x,g(x)) % brėžiami funkcijų  $y = e^{-x^2}$  ir  $y = 1 - e^{-x}$  grafikai.
```

Ši komandų seka koordinatinėje plokštumoje su skirtingomis y -ų ašimis išpieš funkcijų $y = e^{-x^2}$ ir $y = 1 - e^{-x}$ grafikus (2.8 paveikslas).

Ašių mastelis. Brėžinio ašių mastelis parenkamas komanda *axis*. Ši komanda aprašo funkcijos ir argumento ašių intervalą, kuriame pateikiamas grafikas. Komanda *axis* rašoma po *plot* (ar kitos grafinio vaizdo išvedimo) komandos:

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

čia *xmin*, *xmax*, *ymin*, *ymax* reiškia, kad brėžinyje bus pavaizduotas stačiakampys, kurį apibrėžia koordinatės: $x=xmin$, $x=xmax$ ir tiesės: $y=ymin$ ir $y=ymax$.



2.8 pav. Funkcijų $y = e^{-x^2}$ ir $y = 1 - e^{-x}$ grafikai
kai turime dvi skirtingas y -ų ašis

Tarkime, reikia nubraižyti funkcijos $y = e^{5t} - 1$ grafiką, kai $t \in [0;5]$.

Atlikus komandas,

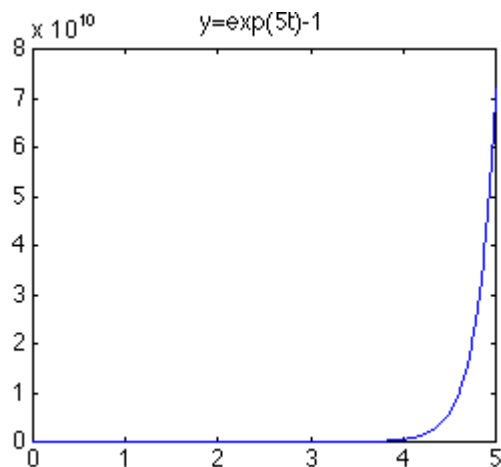
```
t=0:0.01:5;  
y=exp(5*t)-1;  
plot(t,y),
```

brėžinys atrodys (2.9 paveikslas).

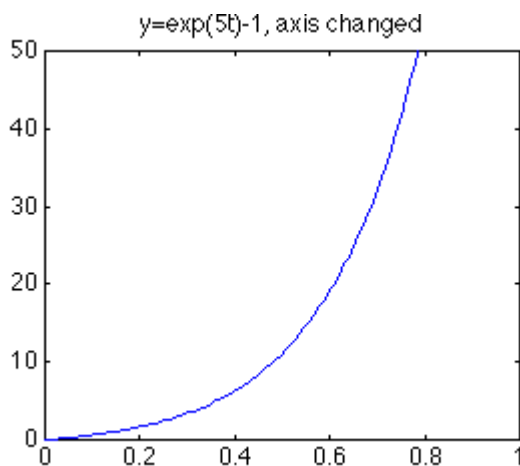
Kaip matyti iš 2.9 brėžinio, kai $t \in [0;4]$, tai funkcijos reikšmės yra labai mažos lyginant jas su funkcijos reikšmėmis, kai $t \in [4;5]$. Tarkime mus domina funkcijos grafikas intervale $(0,1)$. Surinkę komandą: `axis([0, 1, 0, 50])`, ekrane matysime tokį vaizdą (2.10 paveikslas). 2.10 paveikslas žymiai informatyvesnis.

Ašių mastelis parenkamas pagal funkcijos pobūdį. Naudojant `subplot` komandą, ašių mastelis parenkamas kiekvienam brėžiniui atskirai. Šiuo atveju prieš kiekvieną `subplot` komandą rašoma `axis` komanda.

Ašių parametrus valdančiosios komandos pateiktos 2.17 lentelėje.



2.9 pav. Funkcijos $y = e^{5t} - 1$ grafikas be apribojimų



2.10 pav. Funkcijos $y = e^{5t} - 1$ grafikas su apribojimais

Teksto įterpimas. Technikoje svarbu reikiamai apipavidalinti brėžinį. Brėžiniui suteikiamas pavadinimas, koordinatinių ašių pavadinimai bei aiškinamasis tekstas.

Brėžinio pavadinimas užrašomas viršuje per vidurį komanda: `title('brėžinio pavadinimas')`.

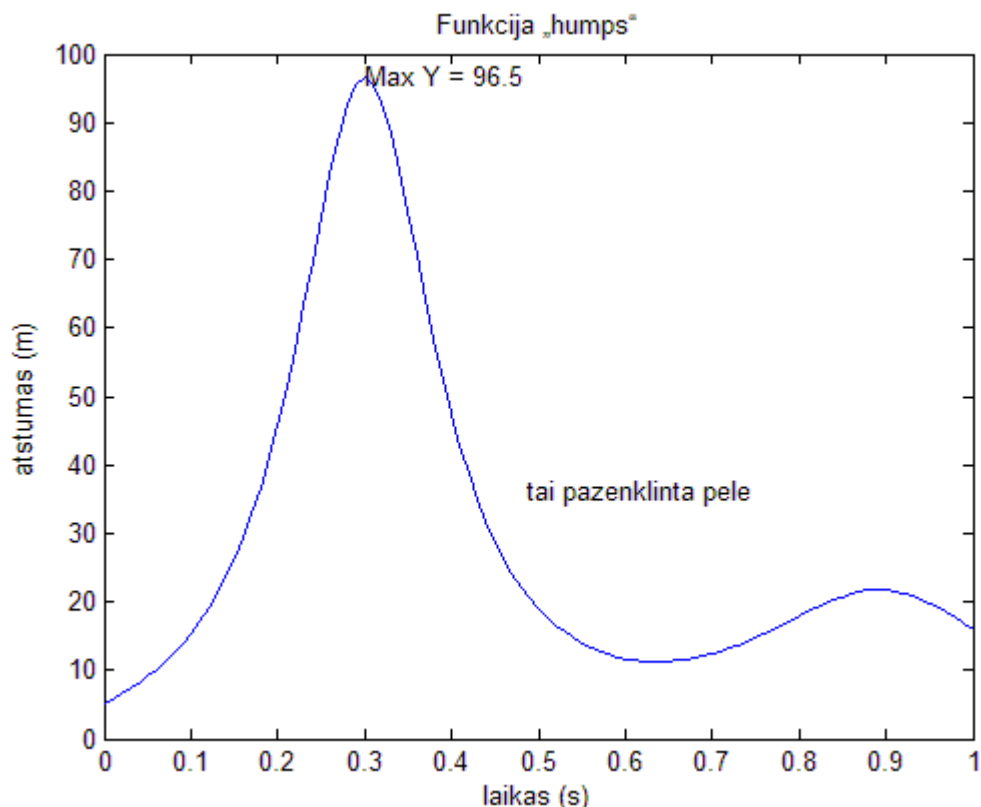
x ašies pavadinimas užrašomas komanda: `xlabel('x-ašies pavadinimas')`, o y – ašies: `ylabel('y-ašies pavadinimas')`.

Tekstas brėžinyje rašomas vienu iš dviejų būdų: `text` ir `gtext` komandų pagalba. Komandoje `text` nurodomos teksto patalpinimo brėžinyje koordinatės, pav.,
`text(xcor,ycor,'tekstas')` parametrai `xcor,ycor` nurodo teksto pradžios koordinatas.

Komanda `gtext('tekstas')` nereikalauja tikslų teksto pradžios koordinatų. Teksto vieta nurodoma pelės pagalba: žymeklis patalpinamas norimoje brėžinio vietoje ir paspaudžiamas pelės mygtukas. Pvz.

```
x = 0:0.01:1.;
y = 1./((x-0.3).^2 + 0.01)+1./((x-0.9).^2+0.04)-6;
plot(x,y)
[ym,i] = max(y);
str = ['Max Y = ',num2str(ym)];
text(x(i),ym,str)
title('Funkcija „humps“')
xlabel('laikas (s)')
ylabel('atstumas (m)')
gtext('tai pazenklinta pele')
```

Tekstas '*tai pazenklinta pele*', paspaudus pelės mygtuką, patalpinamas dešiniau žymeklio. Brėžinio apipavidalinimas parodytas 2.11 paveiksle.

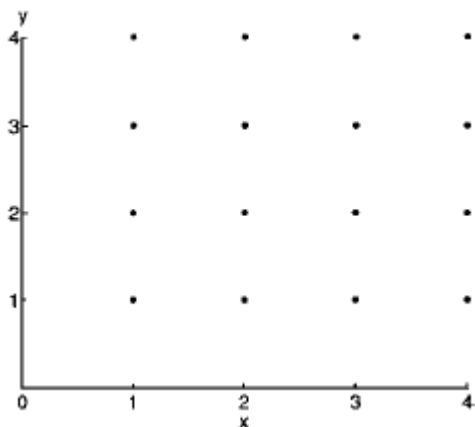


2.11 pav. Apipavidalintas brėžinys

2.10.2. Dviejų kintamųjų funkcijų grafinis vaizdavimas

Dviejų nepriklausomų kintamųjų funkcijos $z=f(x,y)$ grafikas, tai paviršius trimatėje erdvėje. MATLAB'e paviršius aprašomas taškais (x_i, y_i, z_i) .

Tarkime, turime (x,y) taškų koordinačių rinkinį, aprašytą matricomis;
 $x = [1\ 2\ 3\ 4; 1\ 2\ 3\ 4; 1\ 2\ 3\ 4; 1\ 2\ 3\ 4]$; $y = [1\ 1\ 1\ 1; 2\ 2\ 2\ 2; 3\ 3\ 3\ 3; 4\ 4\ 4\ 4]$;
 Grafiškai tai atrodytų taip (2.12 pav.):



2.12 pav. Plokštumos taškai, apibrėžti x ir y matricomis

MATLAB'e taškų (x,y) koordinačių matricas X ir Y generuoja funkcija **meshgrid**:

```
[X,Y] = meshgrid(1:4,1:4);
```

Tarkime, kad šioje srityje paviršių aprašanti funkcija yra $z = (x^2 + y^2)^{1/2}$.

Komanda: $z = \text{sqrt}(X.^2 + Y.^2)$;

apskaičiuoja funkcijos $z = (x^2 + y^2)^{1/2}$ reikšmes. Tada paviršių $z=f(x,y)$ grafiškai galima pavaizduoti komandos: `mesh(X,Y,Z)` pagalba (2.13 pav.).

Komandos: $[X,Y] = \text{meshgrid}(-10:10, -10:10)$;

```
z = sqrt(X.^2 + Y.^2);
```

```
mesh(X,Y,Z)
```

ši paviršių, kuris yra kūgis su viršūne taške $(0, 0, 0)$, pavaizduoja didesnėje srityje (žr. 2.14 paveikslą)

<p>2.13 pav. Funkcijos $z = (x^2 + y^2)^{1/2}$ paviršius srityje $x \in [1,4], y \in [1,4]$</p>	<p>2.14 pav. Funkcijos $z = (x^2 + y^2)^{1/2}$ paviršius srityje $x \in [-10,10], y \in [-10,10]$</p>

Funkcijos paviršius gali būti vaizduojamas skirtingais būdais:

- funkcijos **mesh(x,y,z)** ir **meshc(x,y,z)** paviršių vaizduoja tinkleliu,
- funkcijos **surf(x,y,z)**, **surfc(x,y,z)** ir **surf1(x,y,z)** pateikia paviršių diskrečiais elementais, kurie nuspalvinti pagal funkcijos reikšmę, apšvietimą ar kitą parametą.

Funkcijos **meshc(x,y,z)** ir **surfc(x,y,z)** kartu su paviršiumi pateikia ir nagrinėjamos funkcijos vienodų reikšmių kontūrus (lygio linijas).

Visos aukščiau paminėtos funkcijos turi tą pačią kreipinio struktūrą:

surf(Xmatrica,Ymatrica,Zmatrica).

MATLAB'as leidžia dviejų kintamųjų funkciją pavaizduoti ir lygio linijomis, t.y. funkcijos paviršiaus vienodų reikšmių kontūrais. Šie kontūrai pateikiami (x,y) koordinačių plokštumoje. Funkcijų vaizdavimas lygio linijomis yra plačiai taikomas topografijoje, pvz. vaizduojant vandenynų gylį ar kalnų aukštį.

Į lygio linijas vaizduojančią funkciją **contour** galima kreiptis keliais būdais, pvz.:

contour(Xmatrica,Ymatrica,Zmatrica,nkont),

contour(Xmatrica,Ymatrica,Zmatrica,zkont).

Norimas lygio linijų skaičius nusakomas kintamojo **nkont** reikšme.

Funkcijos vienodų reikšmių kontūrus galima pavaizduoti nurodant funkcijos reikšmes. Šios funkcijos reikšmės priskiriamos vektoriui **zkont**.

Vaizduojant paviršius, gali būti naudojama funkcija **colormap**. Ši funkcija paviršių nuspalvina pasirinkta spalvine gama, ją susiedama su funkcijos (Z matricos) reikšmėmis. Kreipinys į šią funkciją yra: **colormap('spalvinė_gama')**.

MATLAB turi eilę spalvinių gamų, kurių vardai yra: **'bone'**, **'cool'**, **'gray'**, **'hot'** ir **'jet'**.

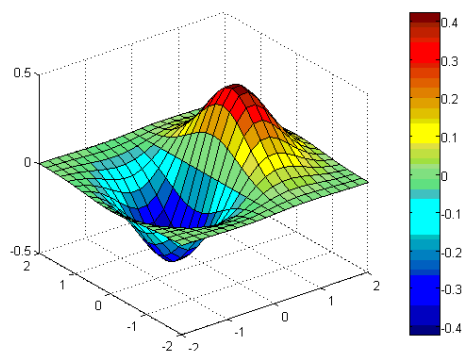
Funkcijos paviršiui galima suteikti ir atspalvius. Tai atlieka MATLAB funkcija **shading('atspalvis')**.

Atspalviai nurodomi simbolių eilutėmis: **'flat'**, **'faceted'** ir **'interp'**. Į funkciją **shading** kreipiamasi po kreipinio į vieną iš paviršiaus vaizdavimo funkcijų.

Pavyzdys. Grafiškai pavaizduokime funkciją $z = f(x_1, x_2), x_1 \in [-2; 2], x_2 \in [-2; 2]$.

2.15 ir 2.16 paveiksluose ši funkcija pavaizduota skirtingais grafiniais būdais.

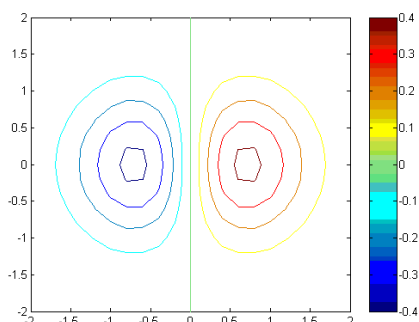
```
x=-2:.2:2;
y=-2:.2:2;
[X,Y] = meshgrid(x,y);
Z = X.*exp(-X.^2-Y.^2);
surf(X,Y,Z)
colorbar
```



2.15 pav. Funkcijos $Z = f(x_1, x_2) = x_1 e^{(-x_1^2 - x_2^2)}$ paviršius su funkcijos reikšmės nurodančiu spalvinės gamos stulpeliu

figure

```
contour (X1,X2,Z) ;
colorbar
```



2.16 pav. Funkcijos $Z = f(x_1, x_2) = x_1 e^{(-x_1^2 - x_2^2)}$ lygio linijos su funkcijos reikšmės nurodančiu spalvinės gamos stulpeliu

Pagrindinės darbo su grafiniais objektais funkcijos pateiktos 2.17 lentelėje.

2.17. lentelė. Pagrindinės darbo su grafiniais objektais funkcijos

Funkcija	Paskirtis
	Grafikų braižymo funkcijos
<i>plot</i>	Brėžia funkcijos grafiką plokštumoje
<i>plotyy</i>	Sukuria dvi funkcijos reikšmių ašis, kairėje ir dešinėje
<i>plot3</i>	Brėžia funkcijos grafiką erdvėje
<i>loglog</i>	Taip pat kaip ir <i>plot</i> , tik abiejų ašių mastelis yra logaritminis
<i>semilogx</i>	Taip pat kaip ir <i>plot</i> , tik <i>x</i> -ašies mastelis yra logaritminis
<i>semilogy</i>	Taip pat kaip ir <i>plot</i> , tik <i>y</i> -ašies mastelis yra logaritminis
	Paviršių braižymo funkcijos
<i>contour</i>	Funkcija vaizduojama kontūro linijomis
<i>surf</i>	Funkcija vaizduojama diskrečiais elementais
<i>surfc</i>	Funkcija vaizduojama diskrečiais elementais ir kontūro linijomis
<i>surfl</i>	Funkcija vaizduojama diskrečiais elementais su apšvietimu
<i>mesh</i>	Funkcijos paviršius vaizduojamas tinkleliu
<i>meshc</i>	Funkcija vaizduojama paviršiaus tinkleliu ir lygio linijomis
<i>meshgrid</i>	3D brėžiniams generuojamos argumentų matricos
	Ašių parametrų valdymo funkcijos
<i>axis square</i>	Suvienodinamas ašių ilgis
<i>axis equal</i>	Suvienodinamas ašių mastelis
<i>axis tight</i>	Ašių mastelis suderinamas su duomenų reikšmėmis
<i>axis auto</i>	Ašių mastelį parenka MATLAB'as
<i>axis off</i>	Brėžinyje ašys nevaizduojamos
<i>axis on</i>	Brėžinyje ašys vėl vaizduojamos
<i>grid on</i>	Brėžinyje pavaizduojamas tinklelis
<i>grid off</i>	Brėžinyje panaikinamas tinklelis
<i>box off</i>	Brėžinyje panaikinamas ašių rėmelis
<i>box on</i>	Brėžinyje pavaizduojamas ašių rėmelis
	Grafinio lango valdymo funkcijos

<i>figure</i>	Atidaromas naujas grafinis langas; taip išsaugomas ankstesniame lange esantis grafinis objektas
<i>close</i>	Uždaromas aktyvusis brėžinio langas
<i>hold on</i>	Aktyviajame lange vaizduojamas papildomas grafinis objektas
<i>hold off</i>	Aktyviajame lange esantys grafiniai objektai naikinami ir vietoje jų vaizduojamas naujas grafinis objektas
<i>subplot</i>	Grafinio lango skaidymas į stačiakampius su koordinčių ašimis
<i>clf</i>	Grafiniame lange panaikinami visi grafiniai objektai; langas tampa tuščias
	<i>Brėžinio apipavidalinimo funkcijos</i>
<i>title</i>	Grafinio objekto (brėžinio) pavadinimas
<i>xlabel</i>	X koordinčių ašies pavadinimas
<i>ylabel</i>	Y koordinčių ašies pavadinimas
<i>zlabel</i>	Z koordinčių ašies pavadinimas
<i>text</i>	Teksto rašymas koordinčių ašių plokštumoje
<i>gtext</i>	Teksto talpinimas koordinčių ašių plokštumoje pelės pagalba
<i>colorbar</i>	Pavaizduojamas su funkcijos reikšmėmis susijusių spalvų stulpelis
<i>camlight('right')</i>	Grafinio objekto apšvietimas iš dešinės
<i>colormap(flipud(colormap))</i>	Grafinio objekto spalvinės gamos nustatymai
<i>lighting phong</i>	Grafinio objekto apšvietimo būdas
<i>shading</i>	Grafinio objekto atspalvio nustatymas