ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՏԱՐՈՆ ՀԱՐՈՒԹՅՈՒՆՅԱՆ

ՌԱԴԻՈՖԻԶԻԿԱԿԱՆ ԽՆԴԻՐՆԵՐԻ ՀԱՄԱԿԱՐԳՉԱՅԻՆ ՄՈԴԵԼԱՎՈՐՈՒՄԸ MATLAB ՄԻՋԱՎԱՅՐՈՒՄ

MATLAB ՄԻՋԱՎԱՅՐԻ ԿԱՌՈՒՑՎԱԾՔԸ

ԵՐԵՎԱՆ
ԵՊՀ ՀՐԱՏԱՐԱԿՉՈՒԹՅՈՒՆ
2016

ՀՏԴ 537.86:004(07)
ԳՄԴ 32.841+32.81g7
Հ 422

Հրատարակության է երաշխավորել
ԵՊՀ ռադիոֆիզիկայի ֆակուլտետի գիտական խորհուրդը

Գրախոսներ՝

ԵՊՀ թվային անալիզի և մաթեմատիկական մոդելավորման ամբիոնի վարիչ,
ֆիզմաթ. գիտությունների դոկտոր, պրոֆեսոր ***Յու. Ռ. Հակոբյան***

ՀԱՊՀ տեղեկատվական տեխնոլոգիաների և ավտոմատացման ամբիոնի դոցենտ,
տեխնիկական գիտությունների թեկնածու ***Դ. Ա. Ղազարյան***

Ֆիզմաթ. գիտությունների դոկտոր ***Դ. Լ. Հովհաննիսյան***

Հարությունյան Տարոն
Հ 422 Ռադիոֆիզիկական խնդիրների համակարգչային մոդելավորումը MATLAB միջավայրում/MATLAB միջավայրի կառուցվածքը/Հարությունյան Տարոն:
-Եր., ԵՊՀ հրատ., 2016, 254 էջ:

Գիրքը նվիրված է ռադիոֆիզիկայի և ֆիզիկայի խնդիրների համակարգչային մոդելավորմանը MatLab միջավայրում: Գրքի առաջին մասն ընդգրկում է MatLab միջավայրի մանրամասն նկարագրությանը: Նախատեսվում է ԵՊՀ և այլ բուհերի ուսանողների, ինչպես նաև բոլոր նրանց համար, ովքեր ցանկանում են աշխատել MatLab փաթեթով:

ՀՏԴ 537.86:004(07)
ԳՄԴ 32.841+32.81g7

ISBN 978-5-8084-2063-2

ՆԵՐԱԾՈՒԹՅՈՒՆ

Համակարգչային մոդելավորումը ֆիզիկայի և ռադիոֆիզիկայի խնդիրների ուսումնասիրության ամենակարևոր փուլերից մեկն է: Չնայած, որ մոդելավորումը չի բերում նոր ֆիզիկական երևույթների հայտնագործության՝ այն հնարավորություն է տալիս ավելի տեսանելի դարձնել և ավելի լավ պատկերացնել տվյալ ֆիզիկական երևույթը: Համակարգչային մոդելավորումը լայնորեն օգտագործվում է նաև նոր փորձերի նախագծման ընթացքում: Այսպիսով, ֆիզիկայի խնդիրների համակարգչային մոդելավորումը շատ օգտակար է ինչպես գիտնականների և ինժեներների, այնպես էլ ուսանողների համար, ովքեր հնարավորություն կստանան ավելի հեշտ յուրացնել ֆիզիկան:

MatLab ծրագրային փաթեթը հնարավորություն է տալիս մոդելավորել ֆիզիկայի և ռադիոֆիզիկայի խնդիրները: Այն 4–րդ սերնդի ծրագրավորման լեզու է իր առանձին գրաֆիկական միջավայրով: **MatLab**–ը (**Mat**rix **Lab**oratory) ստեղծվել է 1984 թվականին **«MathWorks»** ընկերության կողմից: Ավելի քան 30 տարվա ընթացքում **MatLab**–ի հնարավորություններն անընդհատ ավելանում են, ալգորիթմները և գրադարանները՝ կատարելագործվում: Այսօր միլիոնից ավելի գիտնականներ և ինժեներներ տարբեր խնդիրների լուծման համար որպես հզոր գործիք օգտագործում են **MatLab**–ը:

MatLab–ն ունի ճկուն գրաֆիկական միջավայր, որը թույլ է տալիս հեշտությամբ կատարել տարբեր հաշվարկներ, խնդիրների արդյունքները գրաֆիկորեն ներկայացնել: Բացի գրաֆիկական միջավայրից, **MatLab**–ը նաև ինքնուրույն ծրագրավորման լեզու է:

MatLab–ը հարմար է կիրառել մատրիցական հաշվարկներ կատարելիս, ազդանշանների և պատկերների մշակման խնդիրներում, մաթեմատիկական ֆիզիկայի խնդիրներ լուծելիս, օպտիմալացման խնդիրներում և այլն: Բացի հիմնական միջավայրից, **MatLab**–ն ունի նաև առանձին ենթափաթեթներ (**ToolBox**), որոնցից յուրաքանչյուրն ունի իր սեփական գրաֆիկական միջավայրը: Դրանք նախատեսված են կոնկրետ տիպի խնդիրների մոդելավորման համար: Օրինակ, **MatLab**–ում ընդգրկված **Simulink** փաթեթը նախատեսված է դինամիկ համակարգերի մոդելավորման համար, **PDE Toolbox** ենթափաթեթը՝ մաթեմատիկական ֆիզիկայի խնդիրների մոդելավորման համար և այլն:

Այս ձեռնարկի նպատակն է ծանոթացնել ֆիզիկայի և ռադիոֆիզիկայի տարբեր խնդիրների մոդելավորման հիմնական քայլերի և սկզբունքների հետ, ինչպես նաև ցույց տալ, թե ինչպես կարելի է հետազոտել մոդելավորված խնդիր-

ների արդյունքները, դրանք գրաֆիկորեն պատկերել և այլն: Ձեռնարկը բաղկացած է երկու մասից:

Առաջին մասում նկարագրվում է **MatLab** միջավայրը և հիմնական խնդիրները, որոնք կարելի է լուծել այդ միջավայրում: Այդ խնդիրների թվին են պատկանում տարբեր մաթեմատիկական հաշվարկներ ինչպես սովորական թվերի, այնպես էլ վեկտորների և մատրիցների հետ: Ներկայացված են **MatLab** միջավայրում գրաֆիկների ու մակերևույթների կառուցման և հետազոտման եղանակները: Մանրամասն նկարագրված է **MatLab** ծրագրավորման լեզուն, ներկայացված են տրանսցենդենտ և դիֆերենցիալ հավասարումների լուծման, որոշյալ ինտեգրալների հաշվման, կորերով մոտարկման և ինտերպոլացման տարբեր թվային հայտնի եղանակներ և դրանց կիրառությունը **MatLab**–ում, ինչպես նաև սիմվոլային (անալիտիկ) արտահայտությունների հետ աշխատանքներ: Ձեռնարկի առաջին մասը բաղկացած է 8 գլխից: Յուրաքանչյուր գլխում բերված է համապատասխան տեսական մասը՝ բազմաթիվ օրինակներով: Բոլոր գլուխների վերջում բերված են վարժություններ և խնդիրներ՝ նյութն ավելի լավ յուրացնելու համար:

Ձեռնարկի երկրորդ մասը նվիրված է ֆիզիկայի և ռադիոֆիզիկայի տարբեր ոլորտների առանձին խնդիրների համակարգչային մոդելավորմանը: Յուրաքանչյուր խնդիրը ներկայացված է իր մանրամասն նկարագրությամբ և տեսական լուծմամբ, այնուհետև բերված են այդ խնդիրների համակարգչային մոդելավորումները **MatLab** միջավայրում, լուծման գրաֆիկական ներկայացումները և հետազոտությունները: Երկրորդ մասում ևս բերված են տարբեր վարժություններ և խնդիրներ ինքնուրույն լուծման և մոդելավորման համար:

Հեղինակը շնորհակալություն է հայտնում գրախոսներ, ֆիզմաթ. գիտությունների դոկտոր, պրոֆեսոր Յու.Ռ.Հակոբյանին, տեխնիկական գիտությունների թեկնածու Դ. Ա. Ղազարյանին և ֆիզմաթ գիտությունների դոկտոր Դ. Լ. Հովհաննիսյանին, ինչպես նաև Ա.Ս.Հարությունյանին, Գ.Ս.Հաջյանին, Շ.Ռ.Մելիքյանին և Ա. Գ. Ալավերդյանին աշխատանքի ընթացքում ցուցաբերած օգնության և օգտակար դիտողությունների համար:

1. MATLAB ԾՐԱԳՐԻ ՀԻՄՆԱԿԱՆ ՀԱՍԿԱՑՈՒԹՅՈՒՆՆԵՐԸ

1.1. MATLAB ԾՐԱԳՐԻ ԱՇԽԱՏԱՆՔԱՅԻՆ ՄԻՋԱՎԱՅՐԸ

MatLab ծրագրի գրաֆիկական միջավայրը բաղկացած է մի շարք պատուհաններից, որոնցից յուրաքանչյուրն ունի իր առանձին նշանակությունը (նկ. 1)։ Այս ձեռնարկում նկարագրվում է **MatLab2015a** տարբերակի միջավայրը, որը, սակայն քիչ է տարբերվում **MatLab** ծրագրի այլ տարբերակներից։

MatLab ծրագիրը համակարգչում տեղադրելուց (**install**) և առաջին անգամ բացելուց հետո հայտնվում է հետևյալ բաժիններից կազմված գրաֆիկական միջավայրը։

1. **Գործիքների գոտի (Ribbon):** Այստեղ ներկայացվում են **MatLab** միջավայրում աշխատելու համար նախատեսված տարբեր հրամաններ։ Գործիքների գոտին բաղկացած է երեք բաժնից՝ **HOME**, **PLOTS** և **APPS**։ **HOME** բաժնում ներկայացված են հիմնական հրամանները, օրինակ՝ սկրիպտային ֆայլերի և ֆայլ ֆունկցիաների ստեղծում, արդեն ստեղծված ֆայլերի բացում, ֆայլերի պահպանում, ֆայլերն աշխատացնելու հրամանը, օգնության պատուհանը բացելու հրամանը և այլն։ **PLOTS** բաժնում բերված են հրամաններ, որոնք հեշտացնում են գրաֆիկների հետ աշխատանքը։ **APPS** բաժինը պարունակում է **MatLab** ծրագրի հետ ներդրված տարբեր ենթածրագրեր, օրինակ՝ կորերով մոտարկան **Curve Fitting Toolbox** կամ սիմվոլային արտահայտությունների հետ աշխատելու համար նախատեսված **MuPad** ենթածրագրերը։

2. **Հրամանի պատուհան (Command Window)**։ Սա այն հիմնական պատուհանն է, որտեղ գրվում են **MatLab**–ի հրամանները։ Դրանք գրվում են հրամանի տողերում, այսպես կոչված, հրավերի նշանից (>>) հետո։

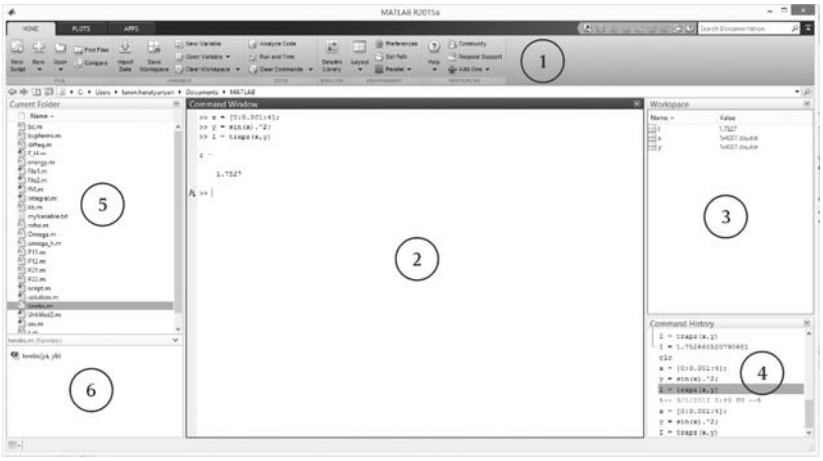
3. **Աշխատանքային միջավայրի (Workspace) պատուհան**։ Այս պատուհանը պարունակում է աշխատանքի ընթացքում սահմանված բոլոր փոփոխականները և դրանց հատկությունները (տիպը, արժեքների տիրույթը և այլն)։

4. **Հրամանների պատմության (Command History) պատուհան**։ Այստեղ բերվում են հրամանի պատուհանում նախկինում գրված հրամանները։

Այդ հրամանները կարելի է պարզապես դիտել կամ կրկին աշխատացնել՝ անհրաժեշտ հրամանի վրա մկնիկի ձախ կոճակը երկու անգամ սեղմելով:

5. **Ընթացիկ թղթապանակի (Current Directory) պատուհան**: Այս պատուհանը պարունակում է տվյալ խնդրի համար անհրաժեշտ ֆայլերն ու ֆունկցիաները, նկարները, տեքստային ֆայլերը և այլն:

6. **Տվյալների (Details) պատուհան:** Եթե ընթացիկ թղթապանակի պատուհանում բերված ֆայլերի ցուցակից ընտրենք կամայական ֆայլ, ապա այս պատուհանում կբերվի այդ ֆայլի նկարագրությունը:



Նկ.1

Բացի հրամանի պատուհանից, մնացած պատուհաններից յուրաքանչյուրն անհրաժեշտության դեպքում կարելի է փակել՝ սեղմելով համապատասխան պատուհանի վերևի աջ անկյունում գտնվող կոճակը և ընտրելով **Close**: Կարելի է նաև սեղմել անհրաժեշտ պատուհաններից կամայականի որևէ հատվածում և ստեղնաշարից սեղմել **Ctrl+W**: Եթե **MatLab**–ը բացելիս կամ աշխատանքի ընթացքում գրաֆիկական միջավայրը չունի նկ.1–ում պատկերված տեսքը՝ պակասում են պատուհաններից մեկը կամ մի քանիսը, ապա այդ տեսքին բերելու համար անհրաժեշտ է **MatLab**–ի գործիքների գոտու **HOME** բաժնից ընտրել **Layout → Default** հրամանը: Բացի **Default** հրամանից, **Layout**–ում նախատեսված են նաև **MatLab**–ի գրաֆիկական միջավայրի այլ ներկայացումներ ևս:

Նկ.1–ում **Layout** բաժնից ընտրված է նաև **Հրամանների պատմության (Command History) պատուհանը**: Վերջինս պարունակում է հրամանի պատուհանում նախկինում գրված հրամանները: Ինչպես ասվեց, այդ հրամանները կարելի է

կրկին իրականացնել՝ անհրաժեշտ հրամանի վրա մկնիկի ձախ կոճակը երկու անգամ սեղմելով: Կարելի է նաև նշել որևէ հրաման, և մկնիկի ձախ կոճակը սեղմած՝ այն տանել դեպի հրամանի պատուհան: Այս դեպքում հրամանը կարելի է խմբագրել, այնուհետև՝ իրականացնել:

MatLab–ն ունի այլ պատուհաններ ևս, օրինակ, գրաֆիկների պատուհանը, սկրիպտային ֆայլեր գրելու համար նախատեսված պատուհանը, օգնության պատուհանը և այլն: Սրանք առավել մանրամասն կնկարագրենք հետագայում:

1.2. ՊԱՐԶԱԳՈՒՅՆ ՀԱՇՎԱՐԿՆԵՐ

Պարզագույն դեպքում **MatLab**–ը կարելի է օգտագործել որպես հզոր հաշվիչ, որը մեծ ճշտությամբ կատարում է տարբեր թվային հաշվարկներ և ունի բազմաթիվ ներդրված ֆունկցիաներ: **MatLab**–ի յուրաքանչյուր հրաման գրվում է հրամանի պատուհանում, հրավերի (>>) նշանից հետո: Հրամանն իրականացվում է, եթե այն գրելուց հետո սեղմում ենք **Enter** ստեղնը: Քանի դեռ **MatLab**–ը չի ավարտել տվյալ հրամանի իրականացումը, ապա նոր տողում չի հայտնվում հրավերի նշանը, և նոր հրաման գրել հնարավոր չէ:

Հրամանը կատարվելուց հետո հաջորդ տողում երևում է հրամանի արդյունքը, սակայն եթե հրամանից հետո դնում ենք կետ–ստորակետ (;), ապա հրամանի արդյունքը հաջորդ տողում չի երևում: Հրամաններից հետո կետ–ստորակետ դնելը հարմար է, օրինակ, այն դեպքերում, երբ ծրագիրը բաղկացած է մի քանի տողից, և անհրաժեշտ է տեսնել ծրագրի միայն կոնկրետ հրամանների, այլ ոչ բոլոր հրամանների արդյունքները:

Օրինակ.

Որպես պարզագույն օրինակ՝ հաշվենք 3 և 8 թվերի գումարը: Դրա համար անհրաժեշտ է հրամանի տողում գրել 3+8 և սեղմել **Enter**.

```
>> 3+8
   ans = 11
>>
```

Ինչպես տեսնում ենք արդյունքն անմիջապես երևում է հաջորդ տողում (քանի որ 3+8 հրամանից հետո կետ–ստորակետ դրված չի), և դրանից հետո միայն հայտնվում է նոր (>>) նշանը, այսինքն՝ **MatLab**–ն ավարտեց հրամանը և սպասում է հաջորդ հրամանին:

Հրավերի նշանը չի հայտնվում նաև այն ժամանակ, երբ ծրագիրը գտնվում է, այսպես կոչված, «կախված» վիճակում: Եթե անհրաժեշտ է ընդհատել չավարտ-

ված հրամանը կամ **MatLab**–ը դուրս բերել «կախված» վիճակից, ապա ստեղնաշարի վրա պետք է սեղմել **Ctrl+C**:

*Հետագա շարադրանքում որևէ հրաման գրելիս ամեն անգամ չենք շեշտի, որ անհրաժեշտ է հրամանից հետո սեղմել **Enter**:*

Հրամանի պատուհանում ամեն հրամանից հետո իրականանում է միայն ամենավերջին հրամանը, իսկ դրան նախորդող բոլոր հրամանները մնում են անփոփոխ: Եթե անհրաժեշտ է փոփոխել նախկինում գրված որևէ հրաման, ապա պետք է այդ և դրան հաջորդող բոլոր հրամանները հերթով մեկ անգամ ևս կանչել: Ընդ որում, նախկինում գրված որևէ հրաման նորից իրականացնելու կամ այդ հրամանի մեջ որևէ փոփոխություն կատարելու համար կարիք չկա կրկին գրել այդ հրամանը: Եթե դատարկ հրամանի տողում անհրաժեշտ քանակությամբ անգամ սեղմենք ստեղնաշարի ↑ կամ ↓ ստեղները, ապա այդ տողում հերթով կհայտնվեն նախկինում գրված հրամանները: Անհրաժեշտ հրամանին հասնելուց հետո այն կարելի է պարզապես կրկին աշխատեցնել կամ խմբագրել նախքան աշխատեցնելը:

MatLab–ի հրամանի պատուհանը կարելի է մաքրել՝ հրամանի տողում գրելով **clc** հրամանը կամ մկնիկի աջ կոճակը սեղմել հրամանի պատուհանի որևէ կետում և ընտրել **Clear Command Window** հրամանը: Ընդ որում, պետք է նկատի ունենալ, որ այս դեպքում ընդամենը մաքրվում է հրամանի պատուհանը. նախկինում գրված բոլոր հրամանները, ինչպես նաև մինչ այդ սահմանված բոլոր փոփոխականները մնում են հիշողության մեջ, ինչում կարելի է համոզվել՝ նայելով հրամանների պատմության և աշխատանքային միջավայրի պատուհաններին:

1.2.1. Թվաբանական գործողություններ

MatLab–ում սահմանված են հետևյալ թվաբանական գործողությունները.

Գործողությունը	**MatLab–ի օպերատորը**	**Օրինակ**
Գումարում	+	5 + 2 = 7
Հանում	–	13 – 8 = 5
Բազմապատկում	*	4 * 9 = 36
Աջ բաժանում	/	6/3 = 2
Ձախ բաժանում	\	5\15 = 3 (համարժեք է 15/5 աջ բաժանմանը)
Աստիճան	^	6^3 = 216

Արտահայտությունների մեջ թվաբանական գործողությունների կատարման կարգը հետևյալն է.

1. աստիճան բարձրացնելը՝ ձախից աջ հանդիպելու հերթականությամբ,
2. բազմապատկումներն ու բաժանումները՝ ձախից աջ հանդիպելու հերթականությամբ,
3. գումարումներն ու հանումները՝ ձախից աջ հանդիպելու հերթականությամբ:

Եթե անհրաժեշտ է փոխել գործողությունների կատարման կարգը, ապա արտահայտության մեջ պետք է օգտագործել կլոր փակագծեր: Այդ դեպքում առաջինը կկատարվեն փակագծերի միջի արտահայտությունները: Եթե արտահայտության մեջ կան ներդրված փակագծեր, ապա առաջինը կկատարվի ամենաներդրված փակագծի միջի արտահայտությունը:

Երկար արտահայտություններն անհարմար են, որովհետև ընթեռնելի են: Երբեմն ավելի հարմար է արտահայտության մի մասը գրել մի տողում, մյուս մասը շարունակել հաջորդ տողում: Եթե արտահայտության ցանկացած հատվածում դնենք երեք կետ և սեղմենք **Enter**, ապա **MatLab**–ը կհասկանա, որ արտահայտությունն ավարտված չի, և հաջորդ տողում չի բերի (>>) նշանը:

Օրինակ.

```
>> 5*(8+6) – 16/(4+79) ...
   + 3*9 – 6
   ans = 90.8072
```

1.2.2. Փոփոխականներ

MatLab–ում փոփոխականները սահմանվում են ծրագրավորման լեզուներին բնորոշ եղանակով՝ օգտագործելով վերագրման (=) օպերատորը: Փոփոխականների անունները կարող են բաղկացած լինել լատիներեն մեծատառերից ու փոքրատառերից, թվերից և ընդգծման գծիկից (_), ընդ որում դրանք անպայման պետք է սկսվեն տառով և չեն կարող բացատ պարունակել: Անհրաժեշտ է նկատի ունենալ, որ **MatLab**–ը զգայուն է մեծատառերի և փոքրատառերի նկատմամբ, հետևաբար, օրինակ, *AA*, *Aa*, *aA* և *aa*–ն 4 տարբեր փոփոխականներ են:

Օրինակ.

```
>> Ab_4c = 8
```

հրամանն աշխատեցնելիս կսահմանվի *Ab_4c* անունով փոփոխականը, որին կվերագրվի 8 արժեքը:

Սահմանված փոփոխականները երևում են աշխատանքային միջավայրի պատուհանում: Եթե անհրաժեշտ է հիշողությունից մաքրել սահմանված որևէ փոփոխական, ապա հրամանի տողում պետք է գրել **clear** հրամանը և բացատով ան-

ջատված՝ փոփոխականի անունը: Մի քանի փոփոխական մաքրելու համար անհրաժեշտ է գրել **clear** և ստորակետներով կամ բացատներով իրարից անջատված՝ համապատասխան փոփոխականների անունները: Սահմանված բոլոր փոփոխականները հիշողությունից հեռացնելու համար գրվում է **clear all** կամ պարզապես **clear**: Փոփոխականները հիշողությունից կարելի է մաքրել նաև առանց **clear** հրամանից օգտվելու: Դա անելու համար պարզապես պետք է աշխատանքային միջավայրի պատուհանում ընտրել անհրաժեշտ փոփոխականը (կամ փոփոխականները) և սեղմել ստեղնաշարի **Delete** ստեղնը:

Օրինակ.

Վերևում սահմանված *Ab_4c* փոփոխականը հիշողությունից ջնջելու համար անհրաժեշտ է հրամանի տողում գրել

```
>> clear Ab_4c
```

MatLab–ում սահմանված է **ans** անունով հատուկ ներդրված փոփոխականը: Եթե որևէ արտահայտության արժեք չի վերագրվում որոշակի փոփոխականի, ապա այն վերագրվում է **ans**–ին:

Օրինակ.

```
>> 15*(6+8)
   ans = 210
```

Քանի որ **ans**–ն ինքը փոփոխական է, ապա իրենից կարելի է օգտվել ճիշտ նույն կերպ, ինչ մնացած բոլոր փոփոխականներից:

Օրինակ.

Նախորդ օրինակում սահմանված **ans**–ը բազմապատկենք 2–ով.

```
>> ans*2
   ans = 420
```

Ինչպես տեսնում ենք, այս գործողության արդյունքը ևս վերագրվեց **ans**–ին, քանի որ այն չէր վերագրվել մեկ այլ փոփոխականի, և **ans**–ի արժեքը փոխվեց: Այսպիսով, **ans**–ը բուֆերային փոփոխական է և ընդունում է ամենավերջին գործողության արդյունքը, եթե այդ գործողությունը բացահայտ կերպով չի վերագրվում որոշակի փոփոխականի:

Հաճախ ծրագիր գրելու ընթացքում կարիք է լինում տեղեկություն ստանալ արդեն սահմանված փոփոխականների և դրանց հատկությունների վերաբերյալ: Դա անելու համար կամ պետք է այդ փոփոխականների ցանկը նայել աշխատանքային միջավայրի պատուհանում, կամ օգտվել **MatLab**–ի **who** կամ **whos** հրամաններից:

- Եթե հրամանի տողում գրենք **who** հրամանը, ապա հաջորդ տողում կերևան սահմանված բոլոր փոփոխականների անունները:
- Եթե հրամանի տողում գրենք **who** հրամանը և բացատից հետո որևէ փոփոխականի անուն, ապա **MatLab**–ը կվերադարձնի այդ փոփոխականի անունը, եթե վերջինս սահմանված է և ոչինչ չի վերադարձնի, եթե այդպիսի փոփոխական սահմանված չէ: Եթե ցանկանում ենք տեղեկություն ստանալ մի քանի փոփոխականի մասին, ապա **who** հրամանից հետո պետք է գրել դրանց անունները՝ բացատներով կամ ստորակետներով իրարից անջատված:
- Եթե հրամանի տողում գրենք **whos** հրամանը, ապա հաջորդ տողում կերևան սահմանված բոլոր փոփոխականների անունները և դրանց մասին տեղեկություն (տիպը, չափը, արժեքների տիրույթը և այլն):
- Եթե հրամանի տողում գրենք **whos** հրամանը և բացատից հետո սահմանված որևէ փոփոխականի անուն, ապա հաջորդ տողում կերևա այդ փոփոխականի անունն ու իր մասին տեղեկությունը: Եթե ցանկանում ենք տեղեկություն ստանալ մի քանի փոփոխականի մասին, ապա **whos** հրամանից հետո պետք է գրել դրանց անունները՝ բացատներով կամ ստորակետներով իրարից անջատված:

Օրինակ.

```
>> x = 15; y = 26; z = x*(y+8);
>> 6*(x/y + 16);
>> who
   Your variables are:
   ans  x    y    z
>> who x y
   Your variables are:
   x  y
>> who k
>> whos
   Name      Size            Bytes  Class     Attributes
   ans       1x1                 8  double
   x         1x1                 8  double
   y         1x1                 8  double
   z         1x1                 8  double
>> whos ans
   Name      Size            Bytes  Class     Attributes
   ans       1x1                 8  double
```

Այս օրինակում սահմանված են *x*, *y*, *z* փոփոխականները, որոնցից առաջինին վերագրվել է 15, երկրորդին՝ 26, իսկ երրորդին՝ *x*(*y*+8) արտահայտության արժեքները: Սահմանված է նաև 6(*x*/*y*+16) արտահայտությունը, որը, քանի որ մեկ այլ փոփոխականի չի վերագրվել, լռելյայն վերագրվում է **ans**–ին: Հետևաբար, երբ գրում ենք **who** հրամանը, **MatLab**–ը վերադարձնում է սահմանված բոլոր փոփոխականների անունները (Your variables are ans, x, y, z): Այժմ ենթադրենք, ցանկանում ենք իմանալ՝ արդյո՞ք սահմանված են *x* և *y* փոփոխականները: Այդ նպատակով գրվում է **who x y** հրամանը: Քանի որ երկու փոփոխականներն էլ սահմանված են, **MatLab**–ը հաջորդ տողում վերադարձնում է այդ փոփոխականների անունները (Your variables are x y): Նույնը չի կարելի ասել *k* անունով փոփոխականի մասին: Երբ գրում ենք **who k** հրամանը, **MatLab**–ը ոչինչ չի վերադարձնում, քանի որ այդպիսի փոփոխական չկա սահմանված: Հաջորդ տողում գրվում է **whos** հրամանը, որը վերադարձնում է սահմանված բոլոր փոփոխականների անունները և դրանցից յուրաքանչյուրի մասին ինֆորմացիան: Ինչպես տեսնում ենք, սահմանված բոլոր փոփոխականները սկալյար են (նրանք ունեն 1×1 չափը), յուրաքանչյուրը հիշողության մեջ զբաղեցնում է 8 բայթ և իրական տիպի են (double): Եթե ցանկանում ենք տեղեկություն ստանալ միայն **ans** փոփոխականի մասին, գրում ենք **whos ans** հրամանը, ինչն էլ գրված է օրինակում՝ որպես վերջին հրաման:

1.3. ԹՎԵՐԻ ՏԻՊԵՐԸ ԵՎ ԴՐԱՆՑ ՆԵՐԿԱՅԱՑՄԱՆ ՖՈՐՄԱՏՆԵՐԸ

Ծրագրավորման լեզուներում փոփոխականներ հայտարարելիս անհրաժեշտ է նախօրոք հայտարարել այդ փոփոխականի տիպը, որպեսզի ըստ դրա կոմպիլյատորն այդ փոփոխականի համար հիշողության մեջ տեղ հատկացնի (տիպից կախված): **MatLab**–ն այդ իմաստով տարբերվում է այլ ծրագրավորման լեզուներից: Ինչպես տեսանք, այստեղ մենք կարող ենք կամայական փոփոխականի արժեք վերագրել՝ առանց նախօրոք այդ փոփոխականի տիպը հայտարարելու: Պատճառն այն է, որ նախօրոք փոփոխականի տիպը չնշելու դեպքում **MatLab**–ը թվային տվյալները հայտարարում է որպես կրկնակի ճշտությամբ իրական թիվ՝ double (double precision floating point number), որի համար հիշողության մեջ հատկացնում է 8 բայթ (այսինքն՝ 64 բիթ): Դրանում արդեն համոզվեցինք վերջին օրինակում, երբ **whos** հրամանով դիտում էինք նախօրոք տիպը չհայտարարված *x*, *y*, *z* փոփոխականների հատկությունները:

Քանի որ կրկնակի ճշտությամբ իրական թվերի քանակը վերջավոր է, հնարավոր չէ ցանկացած թիվ ներկայացնել այս ճշտությամբ: Այս ճշտությունը որոշելու համար նախատեսված է ներդրված **eps** փոփոխականը, որը ցույց է տալիս 1 թվի և դրան հաջորդող կրկնակի ճշտությամբ ամենամոտ թվի հեռավորությունը: Այդ հեռավորությունը հավասար է 2.220446049250313e–016 կամ, որ նույնն է՝ 2^{-52} :

Անհրաժեշտության դեպքում **MatLab**–ը հնարավորություն է տալիս փոխել փոփոխականի տիպը: Այն կարելի է դարձնել սովորական իրական թիվ (single), ամբողջ թիվ, առանց նշանի ամբողջ թիվ և այլն: **MatLab**–ում սահմանված են մի շարք ֆունկցիաներ, որոնք թույլ են տալիս փոխել փոփոխականի տիպը: Այդ ֆունկցիաների ցանկն ու նկարագրությունը բերված են հաճախ օգտագործվող ներդրված ֆունկցիաների բաժնում (տե՛ս, բաժին 1.5.1):

single տիպի համար նախատեսված է 32 բիթ, հետևաբար այն ավելի քիչ հիշողություն է պահանջում, քան double տիպը, սակայն double տիպն ավելի ճշգրիտ է: Որպես double տիպի կարող են պահվել բոլոր այն թվերը, որոնք ընկած են մոտավորապես $[-3.4\cdot10^{38}; 3.4\cdot10^{38}]$ միջակայքում:

Այժմ տեսնենք, ինչպես են թվային տվյալները գրվում **MatLab**–ի միջավայրում: Ամբողջ թվերը գրվում են սովորական ձևով (օրինակ՝ 17 կամ –89): Տասնորդական կոտորակները գրվում են՝ ամբողջ մասը կոտորակայինից կետով անջատելով (օրինակ՝ 3.84): Ընդ որում, եթե տասնորդական կոտորակի ամբողջ մասը հավասար է զրոյի, կարելի է գրել միայն կետը և կոտորակային մասը: Օրինակ, հավասար չափով կիրառելի են 0.125 և .125 գրելաձևերը:

Բացի այդ, **MatLab**–ը թույլ է տալիս նաև թվերի էքսպոնենցիալ գրելաձևը, որն առավել հարմար է շատ մեծ կամ շատ փոքր թվերը ավելի կարճ գրելու համար:

Օրինակ.

1000000000 թիվը արտահայտությունների մեջ գրելը այնքան էլ հարմար չէ: Իհարկե, կարելի է այս թիվը գրել 10^9 տեսքով: Բայց շատ ավելի հարմար է օգտվել էքսպոնենցիալ գրելաձևից: Այս գրելաձևում նախ գրվում է թվի հիմքը, այնուհետև, առանց բացատ դնելու գրվում է e տառը, որին, դարձյալ առանց բացատի, հետևում է 10 թվի աստիճանը: Այսինքն՝ 1000000000 թիվը կգրվի 1e9 տեսքով: Նման ձևով, օրինակ, 536790000 թիվը կգրվի 5.3679e8 կամ 5.3679e+8 տեսքով (+ նշանը ցույց է տալիս, որ 10–ի աստիճանը դրական է): Շատ փոքր թվեր գրելիս դրանք կարելի է ներկայացնել 10–ի բացասական աստիճանների տեսքով:

Օրինակ.

–0.00004654 թվի փոխարեն ավելի հարմար է գրել –4.654e–5: 10 թվի բացասական աստիճան գրելու ժամանակ e տառից հետո – նշանը դնելը պարտադիր է, մինչդեռ 10–ի դրական աստիճանի դեպքում + նշանը կարելի է նաև չգրել: Անհրաժեշտ է նկատի ունենալ, որ e տառից հետո կարող է գրվել միայն ամբողջ թիվ, կոտորակային թվեր չեն թույլատրվում (օրինակ՝ սխալ է 7.34e1.5 գրելաձևը): Ընդ որում, այս գրելաձևում e տառը պետք չէ շփոթել բնական լոգարիթմի հիմքի հետ. այն պարզապես էքսպոնենցիալ (ցուցչային) բառի առաջին տառն է:

Կոմպլեքս թվերը գրելու համար որպես կեղծ միավոր **MatLab**–ում օգտագործվում են *i* կամ *j* տառերը, բայց պատասխաններում միշտ վերադարձվում է *i* տառը: Օրինակ՝ կարելի է գրել 5+3*i* կամ 6–4*j*: Բազմապատկելիս, բաժանելիս կամ աստիճան բարձրացնելիս կոմպլեքս թվերն անպայման պետք է ներառել կլոր փակագծերի մեջ, օրինակ՝ (6–4*i*)^2 կամ (3.6+1.2*i*)*6: Եթե այս օրինակներում կոմպլեքս թվերը չգրվեին փակագծերում, ապա աստիճան բարձրացնելու կամ բազմապատկման գործողությունները կկատարվեին միայն թվի կեղծ մասի հետ: Կոմպլեքս թվի համալուծը որոշելու համար նախատեսված է շտրիխ (′) օպերատորը, որը դրվում է անմիջապես կոմպլեքս թվից հետո (ընդ որում՝ կոմպլեքս թիվը կարելի է դնել փակագծերի մեջ կամ չդնել):

Օրինակ.

```
>> 4.56+9i′
ans = 4.5600 – 9.0000i
>> (3–9j)′
ans = 3.0000 + 9.0000i
```

Կոմպլեքս թվերի հետ աշխատելու համար նախատեսված են մի շարք ֆունկցիաներ, որոնք թույլ են տալիս որոշել կոմպլեքս թվի իրական, կեղծ մասերը, արգումենտը, փուլը և այլն: Դրանց հետ կծանոթանանք 1.5.1 բաժնում:

MatLab–ում թվային գործողությունների արդյունքները կարելի է ներկայացնել տարբեր ֆորմատներով: Դրա համար հրամանի տողում պետք է գրել **format** բառը և դրանից հետո՝ անհրաժեշտ ֆորմատի անունը: Թվերի ներկայացման մի շարք ֆորմատներ ներկայացված են հետևյալ աղյուսակում:

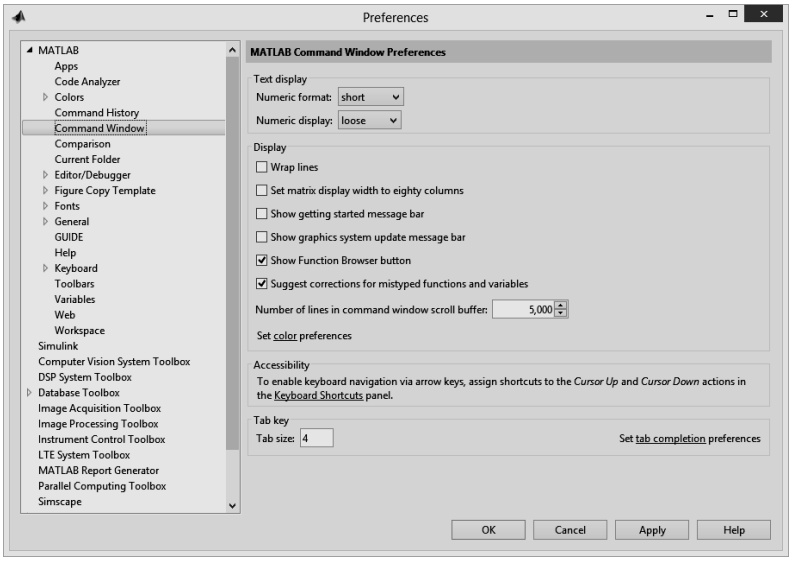
Անվանումը	**Նկարագրությունը**
format short	Թիվը ներկայացվում է տասնորդական կոտորակի տեսքով, կոտորակային մասը պարունակում է 4 նիշ:
format long	Թիվը ներկայացվում է տասնորդական կոտորակի տեսքով, կոտորակային մասը պարունակում է 15 նիշ:
format short e	Թիվը ներկայացվում է էքսպոնենցիալ տեսքով, որի հիմքը գրվում է տասնորդական կոտորակի տեսքով, կոտորակային մասը պարունակում է 4 նիշ:
format long e	Թիվը ներկայացվում է էքսպոնենցիալ տեսքով, որի հիմքը գրվում է տասնորդական կոտորակի տեսքով, կոտորակային մասը պարունակում է 15 նիշ:
format bank	Թիվը ներկայացվում է տասնորդական կոտորակի տեսքով, կոտորակային մասը պարունակում է 2 նիշ:
format rat	Թվի մոտավոր տեսքը ներկայացվում է երկու ամբողջ թվերի հարաբերության տեսքով:

Օրինակ, եթե **MatLab**–ի հրամանի տողում գրենք **format short**, ապա հետագա բոլոր գործողություններում արդյունքները կներկայացվեն տասնորդական կոտորակի տեսքով, որի կոտորակային մասը պարունակում է 4 նիշ: Այս ֆորմատը կպահպանվի, քանի դեռ հրամանի տողում չի գրվել մեկ այլ ֆորմատի հրաման:

Օրինակներ.

```
>> format short
>> 351/7
   ans = 50.1429
>> format long
>> 351/7
   ans  = 50.142857142857146
>> format short e
>> 1265/87
   ans = 1.4540e+001
>> format long e
>> 451/3
   ans  = 1.503333333333333e+002
>> format bank
>> 251/8
   ans = 31.38
>> format rat
>> 4.51289
   ans = 1751/388
```

Թվերի ներկայացման ֆորմատները կարելի է փոխել նաև առանց հրաման գրելու: Դրա համար անհրաժեշտ է **MatLab**–ի գործիքների գոտու **HOME** բաժնից ընտրել **Preferences**, և բացված պատուհանի ձախ մասում գտնվող ծառից ընտրել **Command Window**: Աջ մասում բացված պատուհանում **Text display** բաժնից պետք է ընտրել անհրաժեշտ ֆորմատը **Numeric format** տողում (նկ. 2):



Նկ. 2

1.4. ՆԵՐԴՐՎԱԾ ՓՈՓՈԽԱԿԱՆՆԵՐ

MatLab–ում սահմանված են մի շարք ներդրված փոփոխականներ, որոնք կարելի է օգտագործել հաշվարկների ժամանակ: Այդ փոփոխականներից են վերևում սահմանված **ans**–ը և **eps**–ը: Այդ և ուրիշ հաճախ օգտագործվող ներդրված փոփոխականներ բերված են աղյուսակում.

Անվանումը	**Նկարագրությունը**
ans	Ամենավերջին պատասխանը: Եթե ընթացիկ արտահայտությունը չի վերագրվել որևէ փոփոխականի, այն լռելյայն վերագրվում է **ans** ներդրված փոփոխականին:
pi	π թիվը
eps	Կրկնակի ճշտությամբ երկու իրական թվերի ամենափոքր տարբերությունը
Inf	Անվերջություն (∞)
NaN	Անորոշություն
i կամ j	Կեղծ միավորը ($\sqrt{-1}$)

Օրինակներ.

```
>> pi
   ans = 3.1416
>> pi^(1/3)
   ans = 1.4646
>> eps
   ans = 2.2204e-016
>> 1/0
   ans = Inf
>> Inf+13
   ans = Inf
>> 14/Inf
   ans = 0
>> 0/0
   ans = NaN
```

Այս օրինակներում նախ գրվել է **pi** հրամանը, որը վերադարձնում է π թվի արժեքը: Երկրորդ հրամանով որոշվում է π թվի 1/3 աստիճանի արժեքը: Հաջորդ հրամանով որոշվում է **eps** փոփոխականի արժեքը: 1/0 հրամանը վերադարձնում է **Inf**, այսինքն՝ ∞: Եթե անվերջությանը գումարենք 13, դարձյալ կստանանք անվերջություն (5–րդ հրամանը): 6–րդ հրամանում 14 թիվը բաժանած է անվերջության վրա, հետևաբար արդյունքում ստացվում է 0: Վերջին հրամանում 0 թիվը բաժանած է 0–ի վրա, որի արդյունքն անորոշություն է (**NaN**):

Անհրաժեշտ է ուշադիր լինել և փոփոխականներ սահմանելիս չօգտագործել ներդրված փոփոխականների անունները: Այդ անուններով փոփոխականներ սահմանելիս **MatLab**–ը սխալ չի վերադարձնի, սակայն նոր արժեք ընդունելով՝ դրանք կկորցնեն իրենց արժեքը:

1.5. ՖՈՒՆԿՑԻԱՆԵՐ

Ինչպես բարձր կարգի ծրագրավորման այլ լեզուներում, **MatLab**–ում ևս կա ֆունկցիայի գաղափարը, որը թույլ է տալիս մեծ ծրագիրը բաժանել առանձին ենթածրագրերի, ինչպես նաև ծրագրի կրկնվող հատվածներն առանձին սահմանել ու դրանց անուն տալ, և ամեն անգամ կրկնվող հատվածը գրելու փոխարեն օգտագործել այդ անունը: **MatLab**–ում կան բազմաթիվ ներդրված ֆունկցիաներ, բայց ֆունկցիան կարող է ստեղծել նաև օգտագործողը: Թե ինչպես են ստեղծվում ֆունկցիաները, կծանոթանանք 6.2 բաժնում:

Այս գլխում կներկայացնենք սովորական սկալյար թվերի հետ աշխատելու համար նախատեսված առավել հաճախ օգտագործվող ներդրված ֆունկցիաների ցանկը և նկարագրությունը:

Մեկ փոփոխականից կախված ֆունկցիան կանչելու համար անհրաժեշտ է գրել ֆունկցիայի անունը և կլոր փակագծի մեջ՝ մուտքային արգումենտը: Մի քանի արգումենտից կախված ֆունկցիա կանչելու համար գրվում է ֆունկցիայի անունը և կլոր փակագծի մեջ՝ ստորակետներով իրարից անջատված մուտքային արգումենտները: Ֆունկցիայի անվան և բացվող կլոր փակագծի միջև բացատ չի դրվում:

Օրինակ 1.

Թվի քառակուսի արմատը որոշելու համար սահմանված է **sqrt** անունով ֆունկցիան, որն ունի մեկ մուտքային արգումենտ (այն թիվը, որի քառակուսի արմատը փնտրում ենք): Հետևաբար, որևէ թվի, օրինակ՝ 16–ի քառակուսի արմատը հաշվելու համար հրամանի տողում պետք է գրվի **sqrt(16)**: Եթե **sqrt(16)**–ը չի վերագրվում որևէ փոփոխականի, այն կվերագրվի ներդրված ans փոփոխականին, և վերջինիս կվերագրվի 4 թիվը: Եթե **sqrt(16)**–ը ուզում ենք վերագրել, օրինակ *x* փոփոխականին, ապա պետք է գրենք **x=sqrt(16)**:

Օրինակ 2.

Կամայական երկու թվերի հարաբերության մնացորդը որոշելու համար սահմանված է **rem** անունով ֆունկցիան, որն ունի երկու մուտքային արգումենտ (առաջինը՝ բաժանելին, երկրորդը՝ բաժանարարը): Եթե ցանկանում ենք որոշել, օրինակ 5 և 2 թվերի հարաբերության մնացորդը և արդյունքը վերագրել *y* փոփոխականին, անհրաժեշտ է հրամանի տողում գրել **y=rem(5, 2)**:

Այս օրինակներում ներկայացված ֆունկցիաները ունեին միայն մեկ ելքային արգումենտ՝ մի դեպքում դա թվի քառակուսի արմատի արժեքն էր, մյուս դեպքում՝ երկու թվերի հարաբերության մնացորդի արժեքը: Սակայն **MatLab**–ում սահմանված են նաև ֆունկցիաներ, որոնք ունեն ոչ թե մեկ, այլ մի քանի ելքային արգումենտներ: Այսպիսի ֆունկցիաներն էլ կարելի է վերագրել միայն մեկ փոփոխականի, միայն թե այս դեպքում փոփոխականին կվերագրվի միայն առաջին ելքային արգումենտի արդյունքը, իսկ մնացած ելքային արգումենտները որևէ փոփոխա-

կանի չեն վերագրվի: Որպեսզի բոլոր ելքային արգումենտներն էլ վերագրվեն որոշակի փոփոխականների և հետագայում դրանք հնարավոր լինի օգտագործել, անհրաժեշտ է վերագրման գործողության ձախ կողմում քառակուսի փակագծերի մեջ ստորակետներով անջատված գրել անհրաժեշտ փոփոխականների անունները:

Օրինակ 3.

Թվերի հաջորդականության առավելագույն արժեքը և այդ արժեքի ինդեքսը որոշելու համար սահմանված է **max** անունով ֆունկցիան, որը ունի մեկ մուտքային արգումենտ (հաջորդականության անունը) և երկու ելքային արգումենտ (առաջինը՝ առավելագույն արժեքը, երկրորդը՝ ինդեքսը): Եթե, օրինակ, ենթադրենք, որ հաջորդականության անունը *a* է, և **max(a)**–ն վերագրենք որևէ *m* փոփոխականի (հրամանի տողում գրենք **m=max(a))**, ապա, քանի որ **max** ֆունկցիայի առաջին ելքային արգումենտը հաջորդականության առավելագույն արժեքն է, ապա հենց այդ արժեքը կվերագրվի m–ին: Հաջորդականության առավելագույն արժեքի ինդեքսի համարը ոչ մի փոփոխականի չի վերագրվել, հետևաբար, այն հնարավոր էլ չի օգտագործել: Իսկ եթե ցանկանում ենք իմանալ նաև ինդեքսը, ապա հարկավոր էր հրամանի տողում գրել **[m, k]=max(a)**: Այսպիսի վերագրման դեպքում m–ին կվերագրվի հաջորդականության առավելագույն արժեքը, իսկ *k*–ին՝ վերջինիս ինդեքսը հաջորդականության մեջ: Նկատի ունենանք, որ այս ֆունկցիան գրված է որպես մի քանի ելքային արգումենտով ֆունկցիայի օրինակ: Քանի որ այս գլխում թվերի հաջորդականությունները դեռևս չենք դիտարկում, այս ֆունկցիայի մանրամասն գրելաձևի մասին կխոսենք հաջորդ գլխում:

1.5.1. Հաճախ օգտագործվող ներդրված ֆունկցիաներ

MatLab–ում սահմանված են բազմաթիվ ներդրված ֆունկցիաներ, որոնք էապես հեշտացնում են օգտագործողի աշխատանքը: Այս բաժնում կներկայացնենք թվաբանական գործողություններում առավել հաճախ օգտագործվող ներդրված ֆունկցիաները և դրանց նկարագրությունը՝ օրինակներով:

Հետևյալ աղյուսակում բերված են **MatLab**–ում ներդրված լոգարիթմական, էքսպոնենցիալ և այլ ցուցչային ֆունկցիաների ցանկը:

Ցուցչային ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
sqrt(x)	*x* արգումենտի քառակուսի արմատը	sqrt(9)=3
exp(x)	*e* թվի *x* աստիճանը	exp(4)=54.5982
log(x)	*x* արգումենտի բնական լոգարիթմը	log(45)=3.8067
log2(x)	*x* արգումենտի 2 հիմքով լոգարիթմը	log2(8)=3
log10(x)	*x* արգումենտի 10 հիմքով լոգարիթմը	log10(10000)=4
pow2(x)	2 թվի *x* աստիճանը	pow2(5)=32
nextpow2(x)	վերադարձնում է առաջին *p* թիվն այնպես, որ $2^p \geq \lvert x \rvert$	nextpow2(16)=4

MatLab–ում սահմանված են եռանկյունաչափական և հակադարձ եռանկյունաչափական ֆունկցիաներ: Դրանք բաժանված են երկու խմբի: Եռանկյունաչափական ֆունկցիաների դեպքում առաջին խմբի ֆունկցիաների արգումենտները պետք է գրվեն աստիճաններով, եթե ֆունկցիայի վերջին տառը d–ն է (օրինակ՝ **sind** կամ **cosd**), երկրորդ խմբի արգումենտները՝ ռադիաններով (օրինակ՝ **sin** կամ **cos**): Հակադարձ եռանկյունաչափական ֆունկցիաների դեպքում առաջին խմբի ֆունկցիաներն արդյունքը վերադարձնում են աստիճաններով, եթե ֆունկցիայի վերջին տառը d–ն է (օրինակ՝ **asind** կամ **acosd**), երկրորդ խմբի ֆունկցիաները՝ ռադիաններով (օրինակ՝ **asin** կամ **acos**):

Եռանկյունաչափական և հակադարձ եռանկյունաչափական ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
sin(x)	*x*–ի սինուսը	sin(pi/3) = 0.8660
sind(x)		sind(30)=0.5
cos(x)	*x*–ի կոսինուսը	cos(pi/3) = 0.5
cosd(x)		cosd(90)=0
tan(x)	*x*–ի տանգենսը	tan(pi/4) = 1
tand(x)		tand(45)=1
cot(x)	*x*–ի կոտանգենսը	cot(pi/4) = 1
cotd(x)		cotd(0)=Inf
sec(x)	*x*–ի սեկանսը	sec(pi/6) = 1.1547
secd(x)		secd(60)=2
csc(x)	*x*–ի կոսեկանսը	csc(pi/6) = 2
cscd(x)		cscd(90)=1
asin(x)	*x*–ի արկսինուսը	asin(1)=1.5708
asind(x)		asind(1) = 90
acos(x)	*x*–ի արկկոսինուսը	acos(–1) = 3.1416
acosd(x)		acosd(–1) = 180
atan(x)	*x*–ի արկտանգենսը	atan(1) = 0.7854
atand(x)		atand(1) = 45
acot(x)	*x*–ի արկկոտանգենսը	acot(–1) = –0.7854
acotd(x)		acotd(–1) = –45
asec(x)	*x*–ի արկսեկանսը	asec(1) = 0
asecd(x)		asecd(1) = 0
acsc(x)	*x*–ի արկկոսեկանսը	acsc(1) = 1.5708
acscd(x)		acscd(1) = 90

MatLab–ում սահմանված հիպերբոլական և հակադարձ հիպերբոլական ֆունկցիաները ներկայացված են հետևյալ աղյուսակում:

Հիպերբոլական և հակադարձ հիպերբոլական ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
sinh(x)	*x*–ի հիպերբոլական սինուսը	sinh(2) = 3.6269
cosh(x)	*x*–ի հիպերբոլական կոսինուսը	cosh(0) = 1
tanh(x)	*x*–ի հիպերբոլական տանգենսը	tanh(3) = 0.9951
coth(x)	*x*–ի հիպերբոլական կոտանգենսը	coth(1) = 1.3130
sech(x)	*x*–ի հիպերբոլական սեկանսը	sech(0) = 1
csch(x)	*x*–ի հիպերբոլական կոսեկանսը	csch(0) = Inf
asinh(x)	*x*–ի հիպերբոլական արկսինուսը	asinh(2) = 1.4436
acosh(x)	*x*–ի հիպերբոլական արկկոսինուսը	acosh(1) = 0
atanh(x)	*x*–ի հիպերբոլական արկտանգենսը	atanh(3) = 0.3466+1.5708i
acoth(x)	*x*–ի հիպերբոլական արկկոտանգենսը	acoth(1) = Inf
asech(x)	*x*–ի հիպերբոլական արկսեկանսը	asech(0) = Inf
acsch(x)	*x*–ի հիպերբոլական արկկոսեկանսը	acsch(1) = 0.8814

Հաջորդ աղյուսակում ներկայացված են իրական թվերի կլորացման համար նախատեսված ֆունկցիաները: Իրական թվերի կլորացումը կարող է լինել տարբեր եղանակներով՝ կլորացում դեպի ամենամոտ ամբողջ թիվը, դեպի 0 ամենամոտ ամբողջ թիվը, այսինքն՝ թվի ամբողջ մասի առանձնացումը և այլն:

Կլորացման ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
round(x)	*x*–ը կլորացնում է դեպի ամենամոտ ամբողջ թիվը	round(7.4)=7 round(–15.8)= –16
fix(x)	*x*–ը կլորացնում է դեպի 0 ամենամոտ ամբողջ թիվը	fix(9.7)=9 fix(–4.3)= –4
floor(x)	*x*–ը կլորացնում է դեպի –∞ ամենամոտ ամբողջ թիվը	floor(9.7)=9 floor(–4.3)= –5
ceil(x)	*x*–ը կլորացնում է դեպի +∞ ամենամոտ ամբողջ թիվը	ceil(9.7)=10 ceil(–4.3)= –4

Հաջորդ աղյուսակում ներկայացված են կոմպլեքս թվերի հետ աշխատելու համար նախատեսված ֆունկցիաները: **MatLab**–ում սահմանված են ֆունկցիաներ, որոնք թույլ են տալիս վերադարձնել կոմպլեքս թվի իրական կամ կեղծ մասերը, մոդուլը կամ փուլը, կոմպլեքս համալուծը և այլն:

Կոմպլեքս թվերի հետ աշխատող ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
real(z)	վերադարձնում է *z*–ի իրական մասը	real(7+4i)=7
imag(x)	վերադարձնում է *z*–ի կեղծ մասը	imag(9–8i)= –8
abs(x)	վերադարձնում է *z*–ի մոդուլը	abs(3+4i)=5
angle(x)	վերադարձնում է *z*–ի փուլը ռադիաններով	angle(7–2i)= –0.2783
complex(x, y)	վերադարձնում է *x+iy* կոմպլեքս թիվը	complex(4, 9)=4+9i
conj(z)	վերադարձնում է *z*–ի կոմպլեքս համալուծը	conj(2+i)=2–1i

Ինչպես արդեն ասել ենք 1.3. բաժնում, եթե թվային փոփոխականի տիպը հստակ չի նշվում, ապա այն **MatLab**–ում սահմանվում է որպես կրկնակի ճշտությամբ իրական թիվ (double): **MatLab**–ը թույլ է տալիս անհրաժեշտության դեպքում փոխել թվային փոփոխականի տիպը՝ դարձնել այն ամբողջ, սովորական իրական թիվ, առանց նշանի ամբողջ թիվ և այլն: Ստորև բերված են այն ֆունկցիաները, որոնք արգումենտում գրված թվային փոփոխականը փոխակերպում են նշված տիպի փոփոխականի: Այստեղ նկատի ունենանք, որ double և single տիպերի արժեքների տիրույթը նույնն է, սակայն single տիպն ավելի քիչ հիշողություն է պահանջում:

Տվյալների տիպերի փոխակերպման ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Արժեքների տիրույթը**
single(x)	սովորական իրական թիվ	$[-3.4\cdot10^{38}; 3.4\cdot10^{38}]$
double(x)	կրկնակի ճշտությամբ իրական թիվ	$[-3.4\cdot10^{38}; 3.4\cdot10^{38}]$
int8(x)	8 բիթանոց, նշանով ամբողջ թիվ	$[-2^7; 2^7-1]$
int16(x)	16 բիթանոց, նշանով ամբողջ թիվ	$[-2^{15}; 2^{15}-1]$
int32(x)	32 բիթանոց, նշանով ամբողջ թիվ	$[-2^{31}; 2^{31}-1]$
int64(x)	64 բիթանոց, նշանով ամբողջ թիվ	$[-2^{63}; 2^{63}-1]$
uint8(x)	8 բիթանոց, առանց նշանի ամբողջ թիվ	$[0; 2^8-1]$
uint16(x)	16 բիթանոց, առանց նշանի ամբողջ թիվ	$[0; 2^{16}-1]$
uint32(x)	32 բիթանոց, առանց նշանի ամբողջ թիվ	$[0; 2^{32}-1]$
uint64(x)	64 բիթանոց, առանց նշանի ամբողջ թիվ	$[0; 2^{64}-1]$

Հետևյալ աղյուսակում բերված են հաճախ օգտագործվող մի քանի այլ ֆունկցիաներ, որոնք առանձին դասակարգման չեն ենթարկվում:

Այլ ֆունկցիաներ		
Անվանումը	**Նկարագրությունը**	**Օրինակ**
abs(x)	*x*–ի բացարձակ արժեքը	abs(7)=7 abs(–12.5)=12.5
sign(x)	*x*–ի նշանի ֆունկցիան	sign(–7.2)= –1 sign(0)=0 sign(128)=1
rem(x, y)	*x* և *y* հարաբերության մնացորդը	rem(25,4) = 1
factorial(x)	*x*–ի ֆակտորիալը	factorial(5)=125

Այստեղ **sign**–ը նշանի ֆունկցիան է, որը վերադարձնում է 1՝ դրական արգումենտի դեպքում, –1՝ բացասական արգումենտի դեպքում և 0, եթե արգումենտը հավասար է 0–ի:

Բացի ներկայացրած ֆունկցիաներից, **MatLab**–ում սահմանված են բազմաթիվ այլ օգտակար ֆունկցիաներ: Դրանց ամբողջական ցանկը այս գրքում ներկայացնել հնարավոր չէ: Մենք սահմանափակվում ենք առավել հաճախ օգտագործվող ֆունկցիաներով: Հաջորդ գլուխներում կծանոթանանք կոնկրետ խնդիրների համար նախատեսված այլ ներդրված ֆունկցիաների հետ ևս, իսկ ներդրված բոլոր ֆունկցիաներին կարելի է ծանոթանալ **MatLab**–ի տարբեր ձեռնարկներում կամ հետևյալ հղումից.

http://www.mathworks.com/help/MatLab/functionlist.html?refresh=true

Ինչպես ներդրված փոփոխականների դեպքում, պետք է ուշադիր լինել գոյություն ունեցող ֆունկցիայի անունով փոփոխական սահմանելիս: Եթե սահմանենք այդ անունով նոր փոփոխական, **MatLab**–ը սխալ չի վերադարձնի, սակայն կկորի ֆունկցիայի իմաստը: Եթե պատահմամբ սահմանվել է որևէ ֆունկցիայի անունով փոփոխական, այն կարելի է մաքրել հիշողությունից՝ օգտագործելով **clear** հրամանը:

Օրինակ.

```
>> sin = 1
   sin = 1
>> sin(pi)
   ??? Subscript indices must either be real positive integers or
   logicals.
>> clear sin
>> sin(pi)
   ans = 1.2246e-016
```

Այս օրինակում հայտարարվում է sin անունով փոփոխական, որին վերագրվում է 1 արժեքը: Չնայած այն բանին, որ sin–ը ներդրված ֆունկցիայի անուն է (հաշվում է արգումենտի սինուսը), **MatLab**–ը սխալ չվերադարձրեց և կատարեց վերագրման գործողությունը: Սակայն այդպիսի վերագրումից հետո արդեն հնարավոր չէ sin–ը օգտագործել որպես ֆունկցիա: Իրոք, օրինակի երկրորդ հրամանում փորձ է արվում հաշվել π թվի սինուսը, և **MatLab**–ը սխալ է վերադարձնում: Սակայն, երբ սահմանված sin փոփոխականը մաքրում ենք հիշողությունից (clear sin), այն դարձյալ հնարավոր է դառնում օգտագործել որպես ֆունկցիա:

1.6. ՄԵԿՆԱԲԱՆՈՒԹՅՈՒՆՆԵՐ

Ծրագրավորման այլ լեզուների նման **MatLab**–ը ևս հնարավորություն է տալիս ծրագրերում մեկնաբանություններ անել: Ինչպես հայտնի է, դրանք տեքստեր են, որոնք նկարագրում են գրված ծրագիրը կամ ծրագրի առանձին տողերը: Մեկ-

նաբանությունները թույլ են տալիս, որ ուրիշները ևս հասկանան ծրագիրը, իսկ ծրագիրը գրողը հետագայում կարողանա այն հեշտությամբ վերհիշել:

Մեկնաբանությունները **MatLab**–ում գրելու համար նախատեսված է տոկոսի (%) սիմվոլը: Եթե որևէ տեղ հանդիպի այդ սիմվոլը, ապա այդտեղից մինչև տվյալ տողի վերջը **MatLab**–ը կհասկանա որպես մեկնաբանություն:

Օրինակ.

```
>> % Definitions of variables x and y
>> x = 8;
>> y = sin(x);
```

Այս օրինակում առաջին տողը մեկնաբանություն է, որը նկարագրում է հաջորդ տողերում գրված հրամանները:

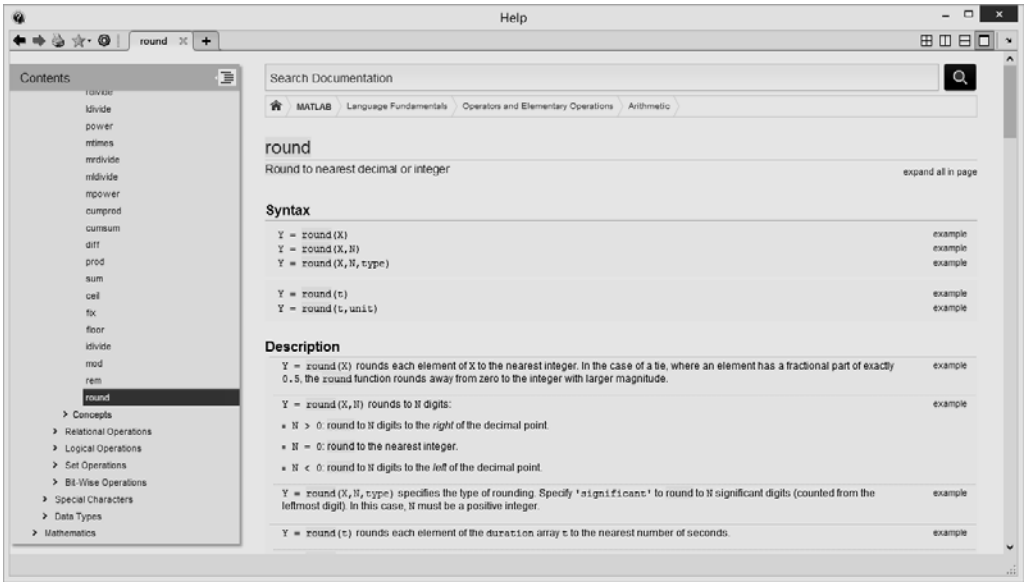
Մեկնաբանությունները կարող են օգտագործվել նաև ծրագրի որոշակի հատվածներ չիրականացնելու համար: Սա հաճախ օգտագործվում է ծրագրի տեստավորման ժամանակ: Այդ դեպքերում ոչ անհրաժեշտ հրամաններից առաջ դնում են % սիմվոլը, և այդ տողերը դիտվում են որպես մեկնաբանություններ:

Ծրագրի որևէ բլոկ կամ մի քանի իրար հաջորդող տողեր որպես մեկնաբանություն գրելու համար կարիք չկա յուրաքանչյուր տողից առաջ գրել % սիմվոլը: **MatLab**–ում նախատեսված է նաև բլոկային մեկնաբանության գաղափարը: Ծրագրի որոշակի հատված մեկնաբանություն դարձնելու համար այդ հատվածից առաջ, դատարկ տողում գրում են տոկոսի և բացվող ձևավոր փակագծի (%{) սիմվոլները, իսկ այդ հատվածից հետո, դարձյալ դատարկ տողում՝ փակվող ձևավոր փակագծի և տոկոսի (}%) սիմվոլները: Այս դեպքում ծրագրի այդ հատվածը դիտվում է որպես մեկնաբանություն:

1.7. Օգնության հնարավորությունը MatLab–ում

Եթե **MatLab**–ով գրված ծրագրի մեջ անհրաժեշտ է օգտագործել որևէ ներդրված ֆունկցիա կամ հրաման, որի անունը մենք գիտենք, սակայն չգիտենք, թե այն ինչպես է պետք օգտագործել (օրինակ, չգիտենք, թե տվյալ ներդրված ֆունկցիան քանի մուտքային արգումենտ է պահանջում կամ քանի ելքային արգումենտ պետք է վերադարձնի), ապա այդ ֆունկցիայի մասին կարելի է տեղեկություն ստանալ **MatLab**–ի օգնության պատուհանից (**Help**): Վերջինս կարելի է բացել տարբեր եղանակներով՝ **MatLab**–ի գործիքների գոտու **HOME** բաժնից սեղմելով  հրամանը, նույն բաժնից ընտրելով **Help → Documentation** կամ ստեղնաշարի վրա սեղմելով F1 ստեղնը: Բացված պատուհանի որոնման տողում գրելով անհրա-

ժեշտ ֆունկցիայի անունը՝ օգնության պատուհանում կհայտնվի այդ ֆունկցիայի մանրամասն նկարագրությունը, օրինակներ և այլն (նկ. 3)։



Նկ. 3

Բացի այդ, անհրաժեշտ ֆունկցիայի մասին տեղեկություն կարելի է ստանալ ավելի հեշտ եղանակով։ Եթե **MatLab**–ի հրամանի տողում գրենք **help** և բացատով անջատված՝ անհրաժեշտ ֆունկցիայի անունը, ապա ֆունկցիայի նկարագրությունը կտեսնենք հենց **MatLab**–ի հրամանի պատուհանում։

Օրինակ.

Ենթադրենք, անհրաժեշտ է տեղեկություն ստանալ **cos** ֆունկցիայի մասին, որն, ինչպես գիտենք, հաշվում է արգումենտի կոսինուսը։ **MatLab**–ի հրամանի տողում կգրենք.

```
>> help cos
   cos    Cosine of argument in radians.
      cos(X) is the cosine of the elements of X.
      See also acos, cosd.
      Overloaded methods:
         codistributed/cos
      Reference page in Help browser
         doc cos
```

Ինչպես տեսնում ենք՝ հրամանի արդյունքում բերվեց **cos** ֆունկցիայի նկարագրությունը և որոշ ընդգծված հղումներ։ Սեղմելով, օրինակ, **acos** կամ **cosd** հղումների վրա՝ տեղեկություն կստանանք այդ ֆունկցիաների մասին։ Դրանք բերվել են, քանի որ

անմիջական կապ ունեն այն ֆունկցիայի հետ, որի մասին ցանկանում էինք տեղեկություն ստանալ՝ դրանցից մեկը թույլ է տալիս հաշվել արգումենտի արկկոսինուսը, մյուսը՝ կոսինուսը, եթե արգումենտն արտահայտված է աստիճաններով: Ամենավերջում բերվել է **doc cos** հղումը, որի վրա սեղմելով՝ կբացվի օգնության պատուհանը՝ **cos** ֆունկցիայի առավել մանրամասն նկարագրությամբ:

Եթե հրամանի տողում գրեինք ոչ թե **help cos**, այլ **doc cos**, ապա միանգամից կբացվեր օգնության պատուհանը, **cos** ֆունկցայի նկարագրության էջում:

Եթե անհրաժեշտ է որոնել և տեղեկություն ստանալ որևէ տառով կամ տառերի խմբով սկսվող ներդրված ֆունկցիաների մասին, ապա **MatLab**–ի հրամանի տողում կարելի է գրել **help**, բացատով անջատված՝ տառը կամ տառերի խումբը և սեղմել **Tab** ստեղնը: Արդյունքում կբացվի փոքր պատուհան, որում սյան տեսքով բերված են նշված տառով կամ տառերի խմբով սկսվող ներդրված ֆունկցիաների ցանկը: Այդ ցանկից ընտրելով անհրաժեշտ ֆունկցիան՝ հրամանի պատուհանում կստանանք դրա նկարագրությունը:

Օրինակ.

Եթե ցանկանում ենք տեսնել c տառով սկսվող ներդրված ֆունկցիաների ցանկը՝ հրամանի տողում կարող ենք գրել **help c** և սեղմել **Tab** ստեղնը: Արդյունքում փոքր պատուհանով էկրանին կհայտնվեն c–ով սկսվող բոլոր ներդրված ֆունկցիաների ցանկը (**c166demos**, **c166editprefs**, **c166prefshelp** և այլն):

Եթե ցանկանում ենք տեսնել co տառերով սկսվող ներդրված ֆունկցիաների ցանկը՝ հրամանի տողում կարող ենք գրել **help co** և սեղմել **Tab** ստեղնը: Արդյունքում փոքր պատուհանով էկրանին կհայտնվեն co–ով սկսվող բոլոր ներդրված ֆունկցիաների ցանկը (**co2di**, **code2html**, **codec** և այլն):

Հնարավոր են դեպքեր, երբ մենք հստակ չգիտենք, թե անհրաժեշտ ֆունկցիայի անունն ինչ է: Ենթադրենք, անհրաժեշտ է կառուցել որևէ ֆունկցիայի գրաֆիկ, սակայն չգիտենք, թե **MatLab**–ում գրաֆիկ կառուցելու բազմաթիվ ներդրված ֆունկցիաներից որն է տվյալ խնդրի համար ավելի նախընտրելի: Այդ դեպքում կարելի է օգտվել **MatLab**–ի **lookfor** հրամանից: Հայտնի է, որ գրաֆիկ բառն անգլերենում plot–ն է: **MatLab**–ի հրամանի տողում գրելով **lookfor plot** հրամանը՝ կստանանք գրաֆիկներ կառուցելու համար նախատեսված ներդրված ֆունկցիաների ամբողջական ցանկը և դրանցից յուրաքանչյուրի կարճ նկարագրությունը: Բերված ֆունկցիաների անունները երևում են հղումների տեսքով: Սեղմելով հղումներից որևէ մեկի վրա՝ կստանանք տվյալ ֆունկցիայի առավել մանրամասն նկարագրությունը:

1.8. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Առանց **MatLab**–ից օգտվելու՝ որոշեք արտահայտությունների արժեքները.

- 1+2*3,
- 2*2^3,
- 4/2*2,
- 2*3\3,
- 1+2/4,
- 2^(1+2)/3,
- 1+2\4,
- 1/2e–1:

2. **MatLab**–ով գրված հետևյալ արտահայտության մեջ գտեք սխալը կամ սխալները.

(2(3+4)/(5*(6+1))^2:

3. Որոշեք՝ արդյոք հետևյալ թվերը **MatLab**–ի գրելաձևով ճիշտ են.

- 9,87,
- .0,
- 25.82,
- –356231,
- 3.57*e2,
- 3.57e2.1,
- 3.57e+2,
- 3,57e–2:

4. Որոշեք՝ կարելի՞ է արդյոք **MatLab**–ում նշված անուններով փոփոխականներ հայտարարել.

- b4,
- c.6,
- var_x,
- 2a,
- min*2,
- inf,
- _x_1,
- var x,
- miXedUp,
- 'a'one:

5. Հաշվեք հետևյալ արտահայտությունների արժեքները.

- $\frac{3+4}{5+6}$,
- $2\pi^2$,
- e,
- 2^{3^2},
- $\frac{3.6^2+4.5^3}{15.1^{4/3}}$,
- $\frac{1}{e}$:

6. Օգտվելով ցուցչային գրելաձևից՝ հաշվեք հետևյալ արտահայտությունների արժեքները.

- $(0.0000123+5.678\cdot10^{-3})\cdot0.4567\cdot10^{-4}$,
- $(256100048000-6.12\cdot10^{-12})-(150000+0.006121\cdot10^{-8})$:

7. Հաշվեք հետևյալ արտահայտությունների արժեքները, եթե x=5, y=8, z=2, a=4 և t=12.5.

- $x-\frac{x^3}{3!}+\frac{x^5}{5!}$,
- x^{y^z},
- $4\,\mathrm{tg}^{-1}x$,
- $(e^{3t}+t^2\sin(4t))\cos^2(3t)$,
- $(x^y)^z$,
- $\ln(x+x^2+a^2)$,
- $\mathrm{ctg}^{-1}\left|\frac{x}{a}\right|$,
- $x^2+y^2-\frac{x^2}{y^2}$:

8. Գրեք ծրագիր, որը կորոշի $ax^2+bx+c=0$ հավասարման լուծումը, եթե $a=2$, $b=-10$, $c=12$:

9. Գրեք ծրագիր, որը $F = 9C/5 + 32$ բանաձևով Ցելսիուսով արտահայտված ջերմաստիճանը կփոխարինի Ֆարենհեյթով արտահայտված ջերմաստիճանի, եթե $C = 0, 100, -40, 37$:

10. Հաշվեք արտահայտությունների արժեքները.

- $\dfrac{35.7 \cdot 64 - 7^3}{45 + 5^2}$,
- $\dfrac{3^7 \ln 76}{7^3 + 546} + \sqrt[3]{910}$,
- $\dfrac{5}{4} \cdot 7 \cdot 6^2 + \dfrac{3^7}{9^3 - 652}$,
- $43 \cdot \dfrac{(\sqrt[4]{250} + 23)^2}{e^{45-3^3}}$,
- $(2+7)^3 + \dfrac{273^{2/3}}{2} + \dfrac{55^2}{3}$,
- $\cos^2 \dfrac{5\pi}{6} \sin\left(\dfrac{7\pi}{8}\right)^2 + \dfrac{\mathrm{tg}\left(\dfrac{\pi}{6} \ln 8\right)}{\sqrt{7}}$,
- $2^3 + 7^3 + \dfrac{273^3}{2} + 55^{3/2}$,
- $\cos\left(\dfrac{5\pi}{6}\right)^2 \sin^2 \dfrac{7\pi}{8} + \dfrac{\mathrm{tg}\left(\dfrac{\pi}{6} \ln 8\right)}{7 \cdot \dfrac{5}{2}}$,
- $\sqrt{\mathrm{ch} \dfrac{1.5}{\sqrt{2\pi}}}$,
- $15 \dfrac{\sqrt{10} + 3.12^2}{\lg 145 + 2.74}$,
- $(e+8)\sin(\pi\sqrt{1.8})$,
- $(-3.5)^{2/3} + \dfrac{e^5}{\ln 125} + 201^{1/3}$,
- $\arccos e^{-\sqrt[3]{3 \cdot 0.5}}$,
- $\dfrac{\mathrm{tg}\, 12^\circ}{\cos^2 12^\circ} - \dfrac{3 \sin 80^\circ}{\sqrt[5]{0.8}}$,
- $\sqrt{4.8}\, \mathrm{tg} \dfrac{\pi \cdot 0.9^3}{2}$,
- $\dfrac{3.2^4 - 1.45}{\sqrt{(1 - 3.24)^2 + 6^{3/2}}}$:

11. Առանց **MatLab**–ից օգտվելու որոշեք՝ ինչի հավասար կլինեն x և a փոփոխականները հետևյալ գործողություններից հետո.

```
>> a = 0;
>> i = 1;
>> x = 0;
>> a = a+i;
>> x = x+i/a;
>> a = a+i;
```

```
>> x = x+i/a;
>> a = a+i;
>> x = x+i/a;
>> a = a+i;
>> x = x+i/a;
```

12. Հաշվեք հետևյալ արտահայտությունների արժեքները, եթե $x = 13.5$ և $z = 8.1$.

- $\lg |x^2 - x^3|$,
- $\dfrac{443z}{2x^3} + \dfrac{e^{-xz}}{x+z}$,
- $\dfrac{\sqrt{14x^3}}{e^{3x}}$,
- $xz^2 - \left(\dfrac{2z}{3x}\right)^{\frac{3}{5}}$:

13. Ենթադրենք՝ $a = 15.62$, $b = -7.08$, $c = 62.5$, $d = 0.5(ab - c)$: Հաշվեք հետևյալ արտահայտությունների արժեքները.

- $a+\dfrac{ab}{c}\dfrac{(a+d)^2}{\sqrt{|ab|}}$,
- $de^{d/2}+\dfrac{\dfrac{ad+cd}{20/a+30/b}}{a+b+c+d}$:

14. Միայն մեկ հրամանով հաշվեք 350 մ^3 ծավալով գնդի r շառավիղը: Օգտվելով ստացված արդյունքից՝ հաշվեք այդ գնդի գնդային մակերևույթի մակերեսը:

15. Ենթադրենք՝ $x=\dfrac{5}{24}\pi$: Առանձին հաշվելով տրված նույնությունների աջ և ձախ կողմերի արտահայտությունները՝ ապացուցեք, որ

- $\cos\dfrac{x}{2}=\sqrt{\dfrac{1+\cos x}{2}}$,
- $\operatorname{tg}\dfrac{x}{2}=\sqrt{\dfrac{1-\cos x}{1+\cos x}}$,
- $\operatorname{tg}2x=\dfrac{2\operatorname{tg}x}{1-\operatorname{tg}^2 x}$,
- $\sin^3 x=\dfrac{1}{4}(3\sin x-\sin 3x)$:

16. Ենթադրենք՝ $\alpha=\dfrac{5\pi}{9}$, $\beta=\dfrac{\pi}{7}$: Ապացուցեք, որ $\cos\alpha-\cos\beta=2\sin\dfrac{\alpha+\beta}{2}\sin\dfrac{\beta-\alpha}{2}$:

17. $(x_0,\ y_0)$ կետի հեռավորությունը $Ax+By+C=0$ ուղղից որոշվում է $d=\dfrac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}}$ արտահայտությամբ: Հաշվեք $(2,\ -3)$ կետի հեռավորությունը $3x+5y-6=0$ ուղղից:

18. Հայտնի է, որ $\int\cos^2(ax)dx=\dfrac{x}{2}-\dfrac{\sin 2ax}{4a}$: Օգտվելով այս բանաձևից՝ հաշվեք $\int\limits_{\pi/7}^{4\pi/3}\cos^2(0.7x)dx$ ինտեգրալի արժեքը:

19. Հայտնի է, որ $\log_a b=\dfrac{\log_c b}{\log_c a}$: Հաշվեք $\log_5 342$ լոգարիթմի արժեքը՝ օգտվելով **MatLab**–ում ներդրված **log10** և **log** ֆունկցիաներից:

20. Մեկ տուփի մեջ տեղավորվում է 12 գրիչ: Որոշեք՝ քանի՞ տուփ է անհրաժեշտ 751 գրիչ տեղավորելու համար:

21. a, b, c կողմերով եռանկյան c կողմը որոշվում է $c^2=a^2+b^2-2ab\cos\gamma$ բանաձևով, որտեղ γ–ն a և b կողմերի կազմած անկյունն է: Միայն մեկ հրաման օգտագործելով՝ որոշել γ–ն, եթե հայտնի է, որ $a=18$ սմ, $b=35$ սմ, $c=50$ սմ:

22. Ռիխտերի սանդղակով երկրաշարժի M ամպլիտուդը որոշվում է $M=\dfrac{2}{3}\lg\dfrac{E}{E_0}$ բանաձևով, որտեղ E–ն երկրաշարժի ժամանակ անջատված էներգիան է, իսկ E_0–ն հաստատուն է և հավասար է $E_0=10^{4.4}$: Որոշեք՝ Ռիխտերի սանդղակով 7.2 բալանոց երկրաշարժի ժամանակ անջատված էներգիան քանի՞ անգամ է մեծ 5.3 բալանոց երկրաշարժի ժամանակ գրանցված էներգիայից:

23. K իներցիալ հաշվարկի համակարգում դադարի վիճակում գտնվող, x առանցքին զուգահեռ տեղադրված ձողի երկարությունը l_0 է: K' իներցիալ համակարգում, որը K համակարգի նկատմամբ շարժվում է հաստատուն V արագությամբ x առանցքի երկայնքով, ձողի երկարությունը կրճատվում է և հավասար է $l = l_0\sqrt{1-\frac{V^2}{c^2}}$, որտեղ c-ն լույսի արագությունն է՝ $c = 3\cdot 10^8$ մ/վ: Ենթադրենք՝ $l_0 = 4$ մ: Որոշեք ձողի երկարությունը K' համակարգում, եթե $V = 8000$ մ/վ:

24. R դիմադրությունից, L ինդուկտիվությունից և E էլշու ունեցող հոսանքի աղբյուրից կազմված փակ շղթայի միացումից t ժամանակ անց հոսանքի ուժի փոփոխության օրենքն ունի $I = \frac{E}{R}\left(1 - e^{-c^2 Rt/L}\right)$ տեսքը, որտեղ c-ն լույսի արագությունն է՝ $c = 3\cdot 10^8$ մ/վ: Որոշեք հոսանքի ուժը շղթայում միացումից 0.04վ հետո, եթե $R = 200$ Օհմ, $L = 0.5$ Հ և $E = 150$ Վ:

25. Ունենք երկու բարձրախոս, որոնք արձակում են միևնույն ձայնը: Առաջին բարձրախոսը ձայնն արձակում է W_1 հզորությամբ, մյուսը՝ W_2 հզորությամբ: Երկու ձայների հզորությունների տարբերությունը դեցիբելներով (dB) որոշվում է $L = 10\lg\frac{W_1}{W_2}$ բանաձևով: Ենթադրենք՝ $L = 60$ dB: Որոշեք՝ երկրորդ բարձրախոսի արձակած ձայնի հզորությունը քանի անգամ է մեծ առաջին բարձրախոսի արձակած ձայնի հզորությունից:

2. ՎԵԿՏՈՐՆԵՐ

Ընդհանրապես, **MatLab**–ը տվյալներն ընդունում և դրանց հետ աշխատում է որպես զանգվածներ, իսկ **MatLab**–ի յուրաքանչյուր տվյալ մատրից է: Հենց **MatLab** անունն ինքը գալիս է **Matrix Laboratory** (մատրիցային լաբորատորիա) անվանումից: Այս լեզվի բոլոր կանոններն ու հրամանները հիմնված են մատրիցական գործողությունների վրա:

Մաթեմատիկայից հայտնի է, որ պարզագույն դեպքում մատրիցը տողերից և սյուներից կազմված ուղղանկյուն աղյուսակ է, որի հանգույցներում գտնվում են տվյալները: Մատրիցի յուրաքանչյուր տող կամ սյուն կոչվում է նաև վեկտոր: m հատ տող և n հատ սյուն ունեցող մատրիցը սովորաբար նշանակվում է $m \times n$ տեսքով: Այն ցույց է տալիս մատրիցի չափը և տարրերի քանակը ($m \cdot n$): Մատրիցներում տարրերն ինդեքսավորվում են, ինչը հարմար է դարձնում յուրաքանչյուր տարրին դիմելը:

Սովորական թիվը կամ սկալյարը 1×1 չափի (1 տողից և 1 սյունից բաղկացած) մատրից է: Սկալյարների հետ աշխատանքը մենք դիտարկել ենք նախորդ գլխում: Հիշենք, որ սովորական սկալյար սահմանելիս (ենթադրենք՝ $x = 15$) աշխատանքային միջավայրի պատուհանում կամ **whos x** հրամանի օգնությամբ այս թվի չափը երևում էր հենց 1×1 տեսքով:

Ինչպես ասացինք՝ մատրիցի յուրաքանչյուր տող կամ սյուն կոչվում է նաև վեկտոր: Հետևաբար, վեկտորը ևս մատրից է, որն ունի $1 \times n$ կամ $n \times 1$ չափը, այսինքն՝ վեկտորի տողերի կամ սյուների քանակը հավասար է 1–ի: Առաջին դեպքում վեկտորը կոչվում է տող–վեկտոր, երկրորդ դեպքում՝ սյուն–վեկտոր:

MatLab–ում վեկտորների և մատրիցների տարրերի ինդեքսավորումը սկսվում է 1–ից, ի տարբերություն, օրինակ, C/C++ լեզվի, որտեղ ինդեքսավորումը սկսվում է 0–ից:

Այս և 3–րդ գլխում կծանոթանանք վեկտորների և դրանց գրաֆիկական ներկայացումների հետ, իսկ մատրիցների հետ աշխատանքին՝ 4–րդ գլխում:

2.1. ՎԵԿՏՈՐԻ ՍԱՀՄԱՆՈՒՄԸ

Այսպիսով, վեկտորը մատրիցի մասնավոր դեպքն է, որն ունի մեկ տող կամ մեկ սյուն: Եթե մատրիցն ունի մեկ տող, այն կոչվում է տող-վեկտոր, իսկ եթե մեկ սյուն՝ սյուն վեկտոր:

Ընդհանուր դեպքում տող-վեկտորը **MatLab**-ում սահմանվում է հետևյալ կերպ.

վեկտորի_անուն = [*տարր*1 *տարր*2 … *տարր*N]

կամ

վեկտորի_անուն = [*տարր*1, *տարր*2, … *տարր*N]

Վեկտորի անունը կարող է բաղկացած լինել լատիներեն մեծատառերից ու փոքրատառերից, թվերից և ընդգծման գծիկից (_), չի կարող պարունակել բացատ և պետք է անպայման սկսվի տառով: Տարրերի արժեքները, որոնք գրվում են վերագրման օպերատորից հետո քառակուսի փակագծերում, իրարից անջատվում են բացատներով կամ ստորակետներով:

Օրինակ.

```
>> a = [3 8 –2.5 12 0 –10 1 621]
   a  =  3.0000      8.0000     –2.5000     12.0000   0   –10.0000
         1.0000      621.0000
>> whos a
  Name      Size             Bytes  Class     Attributes

  a         1x8                 64  double
```

Այս օրինակում սահմանված է 8 տարրից բաղկացած *a* տող–վեկտորը: Քանի որ սահմանումից հետո չի դրվել կետ–ստորակետ, սահմանված վեկտորը երևում է ծրագրի հաջորդ տողում: **whos a** հրամանի օգնությամբ տեղեկություն ենք ստանում *a* վեկտորի մասին: Ինչպես տեսնում ենք՝ այն ունի 1×8 չափը, այսինքն՝ սահմանված տող–վեկտորը **MatLab**–ում պահվում է 1 տող և 8 սյուն ունեցող մատրիցի տեսքով, իսկ հիշողության մեջ իր համար հատկացվում է 64 բայթ: Իրոք, քանի որ *a* վեկտորի տարրերը double տիպի են, ապա դրանցից յուրաքանչյուրի համար հատկացվում է 8 բայթ, իսկ քանի որ վեկտորն ինքը բաղկացած է 8 տարրից, ապա ընդհանուր բայթերի քանակը 64 է:

Սյուն–վեկտորի սահմանումն ընդհանուր դեպքում նման է տող–վեկտորի սահմանմանը, միայն թե տարրերի արժեքները սահմանման մեջ իրարից անջատվում են կետ–ստորակետներով.

վեկտորի_անուն = [*տարր*1*;* *տարր*2*;* … *տարր*N]

Օրինակ.

```
>> b = [15; 24.5; 0.1; –12; 0; 75]
   b =
      15.0000
      24.5000
       0.1000
```

```
    -12.0000
           0
     75.0000
>> whos b
  Name      Size             Bytes  Class     Attributes

  b         6x1                 48  double
```

Տող–վեկտորը կարելի է դարձնել սյուն–վեկտոր կամ հակառակը՝ օգտվելով տրանսպոնացման (′) օպերատորից, որը դրվում է վեկտորի անունից անմիջապես հետո:

Օրինակ.

```
>> a'
   ans =
       3.0000
       8.0000
      -2.5000
      12.0000
            0
     -10.0000
       1.0000
     621.0000
>> b'
   ans =
      15.0000   24.5000    0.1000  -12.0000  0   75.0000
```

Տրանսպոնացման (′) օպերատորից հարմար է օգտվել միայն այն իրական տարրերից կազմված վեկտորների դեպքում: Եթե վեկտորների տարրերը պարունակում են կոմպլեքս թվեր, ապա (′) օպերատորը ոչ միայն կտրանսպոնացնի վեկտորը, այլև վեկտորի մեջ պարունակվող բոլոր կոմպլեքս թվերը կփոխարինի վերջիններիս կոմպլեքս համալուծներով (հիշենք, որ (′) օպերատորը նաև կոմպլեքս համալուծի օպերատորն է):

Օրինակ.

```
>> a = [5 4 3-4i 5+6i 2 0 1+9i]
>> a'
   ans =
      5.0000
      4.0000
      3.0000 + 4.0000i
      5.0000 - 6.0000i
      2.0000
      0
      1.0000 - 9.0000i
```

Այս դժվարությունից խուսափելու համար **MatLab**–ում սահմանված է **transpose** ներդրված ֆունկցիան, որը որպես արգումենտ ստանում է անհրաժեշտ վեկտորը և վերադարձնում վերջինիս տրանսպոնացված վեկտորը՝ վեկտորի մեջ պարունակվող կոմպլեքս թվերը թողնելով անփոփոխ: Բացի **transpose** ֆունկցիայից կարելի է օգտվել նաև տրանսպոնացման (.′) օպերատորից: Վերջինս ևս, ի տարբերություն նախկինում նկարագրված (′) օպերատորի, վեկտորի մեջ պարունակվող կոմպլեքս տարրերը չի փոխարինում դրանց համալուծով:

Օրինակ.

```
>> a = [5 4 3–4i 5+6i 2 0 1+9i]
>> a.'
   ans =
      5.0000
      4.0000
      3.0000 – 4.0000i
      5.0000 + 6.0000i
      2.0000
      0
      1.0000 + 9.0000i
>> transpose(a)
   ans =
      5.0000
      4.0000
      3.0000 – 4.0000i
      5.0000 + 6.0000i
      2.0000
      0
      1.0000 + 9.0000i
```

Բացի բերված ընդհանուր սահմանումներից, գոյություն ունեն վեկտորների սահմանման այլ եղանակներ ևս: Դրանք ավելի են հեշտացնում վեկտորների հայտարարությունը՝ հնարավորություն տալով բոլոր տարրերը հերթով չգրել սահմանման մեջ:

1. Եթե հայտնի են վեկտորի առաջին և վերջին տարրերի արժեքները, և բոլոր հարևան տարրերի միջև հաստատուն քայլը, ապա վեկտորը կարելի է սահմանել հետևյալ կերպ.

վեկտորի_անուն = [*m*:*p*:*n*]

կամ

վեկտորի_անուն = *m*:*p*:*n*

Այս սահմանման մեջ *m*–ը վեկտորի առաջին տարրի արժեքն է, *n*–ը՝ վերջին, իսկ *p*–ն՝ երկու հարևան տարրերի միջև քայլը:

Օրինակներ.

```
>> x = [2:2:10]
   x = 2     4     6     8    10
>> y = [1:2:13]
   y = 1     3     5     7     9    11    13
>> z = [1.5:0.1:2.1]
   z = 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000 2.1000
```

Եթե երկու հարևան տարրերի միջև քայլը հավասար է 1–ի, այն կարելի է սահմանման մեջ բաց թողնել.

վեկտորի_անուն = [*m*:*n*]

Օրինակ.

```
>> t = [-3:7]
   t = -3  -2  -1   0   1   2   3   4   5   6   7
```

Երկու հարևան տարրերի միջև քայլը կարող է լինել նաև բացասական: Այս դեպքում վեկտորի առաջին տարրի արժեքը պետք է մեծ լինի վերջին տարրի արժեքից, հակառակ դեպքում կսահմանվի դատարկ վեկտոր:

Օրինակներ.

```
>> f = [21:-3:6]
   f = 21    18    15    12     9     6
>> g = [5:-8:12]
   g = Empty matrix: 1-by-0
>> whos f g
   Name      Size            Bytes  Class     Attributes
   f         1x5                40  double
   g         1x0                 0  double
```

Եթե *m*, *n*, *p* թվերը տրվում են այնպես, որ *m*-ին *p*-եր գումարելիս *n* թիվը չի ստացվում, ապա վեկտորի վերջին արժեքը հավասար է լինում *n*-ին չգերազանցող ամենավերջին թվին:

Օրինակ.

```
>> f = [1:2:10]
   f = 1     3     5     7     9
```

Ինչպես տեսնում ենք՝ վեկտորի առաջին տարրի արժեքը 1 է, վերջինը՝ 10, իսկ բոլոր հարևան տարրերի միջև քայլը՝ 2: Քանի որ 1–ից 2 քայլով 10 թիվը չի ստացվում, ապա վեկտորի վերջին արժեքը հավասար է 9–ի:

2. Եթե հայտնի են վեկտորի առաջին և վերջին տարրերի արժեքները և տարրերի քանակը, ապա վեկտորը կարելի է սահմանել՝ օգտվելով **linspace** ֆունկցիայից.

վեկտորի_անուն = linspace(m, n, q),

որտեղ *m*–ը և *n*–ը համապատասխանաբար վեկտորի առաջին և վերջին տարրերի արժեքներն են, իսկ *q*–ն՝ տարրերի քանակը, ընդ որում տարրերը բաշխվում են իրարից հավասար հեռավորություններով: **linspace** ֆունկցիայի երրորդ արգումենտը գրելը պարտադիր չի: Եթե այն բաց է թողնվում, ապա տարրերի քանակը ընդունվում է հավասար 100–ի: Եթե *q*=1, ապա **linspace** ֆունկցիան վերադարձնում է *n*–ը:

Օրինակներ.

```
>> va = linspace(0,8,6)
   va = 0    1.6000    3.2000    4.8000    6.4000    8.0000
>> vb = linspace(30,10,11)
   vb = 30   28   26   24   22   20   18   16   14   12   10
>> vc = linspace(10,62,1)
   vc = 62
```

Նշված եղանակներով սահմանվում են միայն տող–վեկտորները: Դրանք կարելի է դարձնել սյուն-վեկտոր՝ օգտվելով տրանսպոնացման օպերատորից:

Օրինակներ.

```
>> d = [7:2:14]
   d = 7    9    11    13
>> e = d'
   e =
        7
        9
       11
       13
>> f = [2:4:18]'
   f =
        2
        6
       10
       14
```

```
>> g = linspace(0,16,11)'
   g =
         0
    1.6000
    3.2000
    4.8000
    6.4000
    8.0000
    9.6000
   11.2000
   12.8000
   14.4000
   16.0000
```

2.2. ՎԵԿՏՈՐՆԵՐՈՎ ԱՇԽԱՏԵԼՈՒ ՀԱՄԱՐ ՆԱԽԱՏԵՍՎԱԾ ՆԵՐԴՐՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐ

Վեկտորներով աշխատելիս հաճախ անհրաժեշտ է լինում որոշել վեկտորի տարրերի առավելագույն կամ նվազագույն արժեքները, հաշվել վեկտորի տարրերի գումարը կամ արտադրյալը, վեկտորի տարրերը դասավորել աճման կամ նվազման կարգով և այլն: **MatLab**–ում կան բազմաթիվ ներդրված ֆունկցիաներ, որոնք իրականացնում են այս և բազմաթիվ այլ գործողություններ: Առավել հաճախ հանդիպող ֆունկցիաները ներկայացված են հետևյալ աղյուսակում:

Վեկտորների հետ աշխատելու համար նախատեսված ֆունկցիաներ	
Անվանումը	**Նկարագրությունը**
lengh(a)	վերադարձնում է *a* վեկտորի երկարությունը, այսինքն՝ տարրերի քանակը
sum(a)	վերադարձնում է *a* վեկտորի տարրերի գումարը
prod(a)	վերադարձնում է *a* վեկտորի տարրերի արտադրյալը
mean(a)	վերադարձնում է *a* վեկտորի տարրերի միջին թվաբանականը
sort(a)	*a* վեկտորի տարրերը դասավորում է աճման կարգով
max(a)	վերադարձնում է *a* վեկտորի առավելագույն արժեքով տարրը և այդ տարրի ինդեքսը
min(a)	վերադարձնում է *a* վեկտորի նվազագույն արժեքով տարրը և այդ տարրի ինդեքսը
transpose(a)	վերադարձնում է *a* վեկտորի տրանսպոնացված վեկտորը

Բացի **max** և **min** ֆունկցիաներից, մնացած բոլոր ֆունկցիաները վերագրվում են մեկ փոփոխականի: Եթե **max** և **min** ֆունկցիաները վերագրվում են մեկ փոփոխականի, ապա դրանք վերադարձնում են համապատասխանաբար միայն վեկտորի առավելագույն և նվազագույն արժեքները, իսկ եթե վերագրվում են երկու փոփոխականների (քառակուսի փակագծերում, ստորակետերով անջատված), վերադարձվում են նաև այդ տարրերի ինդեքսները:

Օրինակ.

```
>> c = [6 12.5 –8 24 3 0.1 –9.2]
   c = 6.0000   12.5000   –8.0000   24.0000    3.0000    0.1000
      –9.2000
>> l = length(c)
   l = 7
>> s = sum(c)
   s = 28.4000
>> p = prod(c)
   p = 39744
>> mn = mean(c)
   mn = 4.0571
>> cs = sort(c)
   cs = –9.2000   –8.0000    0.1000    3.0000    6.0000   12.5000
        24.0000
>> mx = max(c)
   mx = 24
>> [mx, mxi] = max(c)
   mx = 24
   mxi = 4
>> minc = min(c)
   minc = –9.2000
>> [minc, minci] = min(c)
   minc = –9.2000
   minci = 7
```

Այս օրինակում նախ սահմանվում է *c* տող–վեկտորը: Այնուհետև, **length** ֆունկցիայի օգնությամբ *l* փոփոխականին վերագրվում է *c* վեկտորի տարրերի քանակը (վեկտորի երկարությունը), որը հավասար է 7–ի: *s* փոփոխականին վերագրվում է *c* վեկտորի տարրերի գումարը, *p*–ին՝ արտադրյալը, իսկ *mn*–ին՝ միջին թվաբանականը: *mx* փոփոխականին վերագրվում է *c* վեկտորի տարրերի առավելագույն արժեքը: Ինչպես տեսնում ենք, քանի որ **max(c)** ֆունկցիան վերագրվել է միայն մեկ փոփոխականի (*mx*–ին), ապա **max(c)** ֆունկցիան առավելագույն արժեքի ինդեքսը չի վերագրել որևէ փոփոխականի: Հաջորդ հրամանով ստանում ենք նաև առավելագույն արժեքի ինդեքսը, որը վերագրվում է *mxi* փոփոխականին: Նման ձևով *minc* փոփոխականին վերագրվում է *c* վեկտորի նվազագույն արժեքը, իսկ վերջին հրամանով ստանում ենք ինչպես փոքրագույն արժեքը, այնպես էլ վերջինիս ինդեքսը:

2.3. ՎԵԿՏՈՐԻ ՏԱՐՐԵՐԻՆ ԴԻՄԵԼԸ

Վեկտորի առանձին տարրին կարելի է դիմել՝ գրելով վեկտորի անունը և, կլոր փակագծի մեջ, այդ տարրի ինդեքսը: Վեկտորի տարրին դիմելով՝ կարելի է ստանալ վերջինիս արժեքը, այն վերագրել այլ փոփոխականի, օգտագործել տարբեր արտահայտություններում, վեկտորի մեջ փոխել այդ տարրի արժեքը կամ այն հեռացնել վեկտորից: Ընդ որում, այստեղ անհրաժեշտ է հիշել, որ **MatLab**–ում վեկտորի տարրերի ինդեքսավորումը սկսվում է 1–ից:

Օրինակ.

```
>> x = [1 8 –3 0 9 12 24 6 7 2 99]
   ans = 1    8    –3    0    9    12    24    6    7    2    99
>> x(4)
   ans = 0
>> x(1)
   ans = 1
>> x(8)
   ans = 6
>> x(7) + x(2)
   ans = 32
```

Եթե վեկտորի որևէ տարրի նոր արժեք վերագրենք, ապա վեկտորը կմնա նույնը, միայն թե նշված ինդեքսով տարրը կընդունի նոր արժեքը.

Օրինակ.

```
>> x(5) = 7
   x = 1  8  –3  0  7  12  24  6  7  2  99
```

Այստեղ *x*–ը նախորդ օրինակում սահմանված վեկտորն է: Երբ իր 5–րդ ինդեքսով տարրին վերագրում ենք 7 արժեքը, ապա նոր վեկտորն անմիջապես երևում է հաջորդ տողում, ընդ որում իր 5–րդ տարրի արժեքը արդեն ոչ թե 9 է, այլ 7:

Եթե ցանկանում ենք աշխատել վեկտորի ոչ թե մեկ, այլ մի քանի տարրերի հետ, ապա անհրաժեշտ տարրերի ինդեքսները կարելի է որպես առանձին վեկտոր սահմանել և որպես հիմնական վեկտորի ինդեքս գրել այդ նոր վեկտորը:

Օրինակ.

Ենթադրենք, ցանկանում ենք վերևում սահմանված *x* վեկտորը փոխել այնպես, որ իր 2–րդ ինդեքսն ունենա –6 արժեքը, 7–րդ ինդեքսը՝ 3 արժեքը, իսկ 10–րդ ինդեքսը՝ 36 արժեքը: Դրա համար նախ սահմանենք ind անունով նոր տող–վեկտոր, որի արժեքները կլինեն վերոնշյալ ինդեքսների արժեքները.

```
>> ind = [2 7 10];
```

Այժմ, որպես 2–րդ քայլ, *x* վեկտորի *ind* ինդեքսով արժեքներին վերագրենք –6, 3 և 36 թվերը՝

```
>> x(ind) = [-6 3 36]
x =
     1   -6   -3    0    7   12    3    6    7   36   99
```

MatLab–ը հնարավորություն է տալիս տող կամ սյուն–վեկտորների հաջորդաբար դասավորված տարրերի խմբի հետ գործողություններ կատարել ավելի հարմար եղանակով՝ օգտվելով վեկտորների հասցեավորման վերջակետի (:) օպերատորից:

- Եթե վեկտորի ինդեքսում գրում ենք պարզապես (:) օպերատորը (օրինակ՝ x(:)), ապա ընտրվում են այդ վեկտորի բոլոր տարրերը, ընդ որում վեկտորը վերադարձվում է սյան տեսքով:
- Եթե վեկտորի ինդեքսում գրում ենք m:n, որտեղ *m*–ն ու *n*–ը թվեր են (օրինակ՝ x(m:n)), ապա ընտրվում են այդ վեկտորի՝ *m*–ից *n* միջակայքում ընկած տարրերը:
- Եթե վեկտորի ինդեքսում գրում ենք m:p:n, որտեղ *m*–ը, *p*–ն ու *n*–ը թվեր են (օրինակ՝ x(m:p:n)), ապա ընտրվում են այդ վեկտորի *m*–ից *n* միջակայքում ընկած այն տարրերը, որոնք իրարից բաժանված են *p* քայլով:
- Եթե վեկտորի ինդեքսում գրում ենք m:end (կամ m:p:end), որտեղ *m*–ը և *p*–ն թվեր են (օրինակ՝ x(m:p:end)), ապա ընտրվում են այդ վեկտորի *m*–րդից մինչև վերջին ինդեքսով այն տարրերը, որոնք իրարից բաժանված են *p* քայլով:

Օրինակ.

```
>> a = [7 2 8 -1 6 45 -3 0 6]
   a = 7     2     8    -1     6    45    -3     0     6
>> a(:)
   ans =
        7
        2
        8
```

```
        -1
         6
        45
        -3
         0
         6
>> a(2:7) = [3:2:13]
   a = 7     3     5     7     9    11    13     0     6
>> a(4:2:8)
   ans = 7    11     0
>> a(3:end)
   ans = 5     7     9    11    13     0     6
```

Վեկտորների տարրերն իրենք կարող են լինել վեկտորներ, այսինքն՝ կարելի է իրար միացնել 2 կամ ավելի վեկտորներ (այս գործողությունը ծրագրավորման մեջ կոչվում է նաև կոնկատենացման գործողություն): Ընդ որում՝ այս դեպքում բոլոր վեկտորները պետք է լինեն կամ միայն սյուն, կամ միայն տող:

Օրինակ.

```
>> a = [1 2 3]
   a = 1     2     3
>> b = [4 5 6]
   b = 4     5     6
>> c = [7 8 9]
   c = 7     8     9
>> d = [a b c]
   d = 1     2     3     4     5     6     7     8     9
```

Վեկտորի որոշակի տարր կամ տարրեր կարելի է հեռացնել վեկտորից, եթե համապատասխան ինդեքսով կամ ինդեքսներով տարրերին վերագրենք դատարկ քառակուսի փակագծեր:

Օրինակ.

```
>> d(8) = []
   d = 1     2     3     4     5     6     7     9
>> d(3:7) = []
   d = 1     2     9
```

2.4. ԹՎԱԲԱՆԱԿԱՆ ԳՈՐԾՈՂՈՒԹՅՈՒՆՆԵՐ ՎԵԿՏՈՐՆԵՐԻ ՀԵՏ

2.4.1. Գումարում և հանում

Վեկտորի և թվի գումարումը կամ հանումն անդամ առ անդամ գործողություն է: Եթե a վեկտորին աջից կամ ձախից գումարենք (կամ a վեկտորից հանենք) որևէ x թիվ, ապա արդյունքում կստանանք մի նոր b վեկտոր, որն ունի նույն երկարությունը, ինչ a–ն, և որի յուրաքանչյուր տարրի արժեքը հավասար է a վեկտորի համապատասխան տարրի և x թվի գումարին (կամ տարբերությանը):

Օրինակներ.

```
>> a = [7 5 –1 12 0 6]
   a = 7     5    –1    12     0     6
>> x = 2
   x = 2
>> a + x
   ans = 9     7     1    14     2     8
>> x + a
   ans = 9     7     1    14     2     8
>> a – x
   ans = 5     3    –3    10    –2     4
>> x – a
   ans = –5    –3     3   –10     2    –4
```

Երկու վեկտոր կարելի է գումարել կամ իրարից հանել միայն այն դեպքում, երբ դրանք ունեն նույն երկարությունը, ինչպես նաև եթե դրանք երկուսն էլ կամ տող են, կամ սյուն: *a* և *b* երկու վեկտորների գումարումը և հանումը ևս անդամ առ անդամ գործողություն է, այսինքն՝ արդյունքում ստացվում է նույն երկարության վեկտոր, որի յուրաքանչյուր տարրի արժեքը հավասար է *a* վեկտորի և *b* վեկտորի համապատասխան տարրերի գումարին կամ տարբերությանը:

Օրինակներ.

```
>> a = [5 4 8 9 11 0 6];
>> b = [1 9 4 –3 –2 7 1];
>> a + b
   ans = 6    13    12     6     9     7     7
>> b + a
   ans = 6    13    12     6     9     7     7
>> a – b
   ans = 4    –5     4    12    13    –7     5
>> b – a
   ans = –4     5    –4   –12   –13     7    –5
```

2.4.2. Սկալյարի և վեկտորի բազմապատկում և բաժանում

Վեկտորի և սկալյարի բազմապատկումը անդամ առ անդամ գործողություն է: Եթե *a* վեկտորը աջից կամ ձախից բազմապատկենք որևէ *x* թվով, ապա արդյունքում կստանանք մի նոր վեկտոր, որն ունի նույն երկարությունը, ինչ *a*–ն, և որի յուրաքանչյուր տարրի արժեքը հավասար է *a* վեկտորի համապատասխան տարրի և *x* թվի արտադրյալին:

Օրինակ.

```
>> a = [8 –9 12 4 3 0 1];
>> x = 6;
>> a*x
   ans = 48   –54    72    24    18     0     6
```

```
>> x*a
   ans = 48   -54    72    24    18     0     6
```

Վեկտորը կարելի է բաժանել սկալյարի վրա: Այս գործողությունը ևս կատարվում է անդամ առ անդամ, այսինքն, եթե որևէ a վեկտոր բաժանենք x թվի վրա, կստանանք a վեկտորի երկարությունն ունեցող նոր վեկտոր, որի յուրաքանչյուր տարրի արժեքը հավասար է a վեկտորի համապատասխան տարրի և x–ի քանորդին:

Օրինակ.

```
>> a = [4 9 1 26 32 14 7];
>> x = 2;
>> a/x
   ans = 2.0000  4.5000  0.5000 13.0000 16.0000  7.0000  3.5000
```

Սակայն **MatLab**–ում թիվը հնարավոր չէ բաժանել վեկտորի վրա: Պատճառն այն է, որ բաժանման (/) գործողությունը մատրիցների դեպքում նախատեսված է xA=B հավասարման x=B/A լուծումը որոշելու համար: Վեկտորների դեպքում այդ գործողությունը սահմանված չէ: Այս մասին ավելի մանրամասն կխոսենք մատրիցների հետ գործողությունները դիտարկելիս:

Օրինակ.

```
>> a = [8 4 6 11 -23];
>> x = 3;
>> x/a
   ??? Error using ==> mrdivide
   Matrix dimensions must agree.
```

2.4.3. Վեկտորների բազմապատկումը

Ինչպես արդեն նշել ենք, վեկտորները **MatLab**–ը պահում է որպես մատրիցներ: Տող–վեկտորների դեպքում այդ մատրիցն ունի 1 տող, սյուն–վեկտորի դեպքում՝ 1 սյուն: Հայտնի է, որ երկու մատրից կարելի է բազմապատկել միայն այն դեպքում, երբ առաջին մատրիցի սյուների թիվը հավասար է երկրորդ մատրիցի տողերի թվին, իսկ արդյունքում ստացվում է մի նոր մատրից, որի տողերի քանակը հավասար է առաջին մատրիցի տողերի քանակին, իսկ սյուների քանակը՝ երկրորդ մատրիցի սյուների քանակին: Այսինքն, եթե $m \times n$ չափի մատրիցը բազմապատկենք $n \times p$ չափի մատրիցով, կստանանք $m \times p$ չափի մատրից: Վեկտորներն այս պայմանին կարող են բավարարել, երբ դրանք ունեն նույն երկարությունը, և դրանցից մեկը տող է, մյուսը՝ սյուն, հետևաբար միայն այս դեպքում դրանք կարելի է բազմապատկել:

Տող–վեկտորի և սյուն–վեկտորի արտադրյալը

Ենթադրենք՝ n երկարությունն ունեցող a տող–վեկտորը բազմապատկվում է նույն երկարությունն ունեցող b սյուն–վեկտորով: Այս դեպքում, քանի որ տող–վեկտորը $1\times n$ չափի է, իսկ սյուն–վեկտորը՝ $n\times 1$ չափի, ապա արդյունքում կունենանք 1×1 չափի մատրից, այսինքն՝ թիվ: Եթե օգտվենք մատրիցական բազմապատկման կանոնից, այս արտադրյալը կարելի է գրել հետևյալ բանաձևով.

$$a\cdot b=\sum_{i=1}^{n}a_ib_i \tag{2.1}$$

Մաթեմատիկայից հիշենք, որ այս բանաձևն իրենից ներկայացնում է երկու վեկտորների սկալյար արտադրյալը: Իրոք, ենթադրենք ունենք երկու վեկտոր՝ $\vec{a}=\hat{i}a_x+\hat{j}a_y+\hat{k}a_z$ և $\vec{b}=\hat{i}b_x+\hat{j}b_y+\hat{k}b_z$: Այս վեկտորների սկալյար արտադրյալը որոշվում է հետևյալ բանաձևով.

$$\vec{a}\cdot\vec{b}=a_xb_x+a_yb_y+a_zb_z: \tag{2.2}$$

Եթե x, y, z ինդեքսների փոխարեն օգտագործենք 1, 2, 3 նշանակումը, ապա սկալյար արտադրյալը կընդունի հետևյալ տեսքը.

$$\vec{a}\cdot\vec{b}=a_1b_1+a_2b_2+a_3b_3=\sum_{i=1}^{3}a_ib_i: \tag{2.3}$$

Այսինքն, եռաչափ վեկտորների դեպքում (2.1) արտահայտությունը համարժեք է երկու վեկտորների սկալյար արտադրյալին: Ընդհանուր դեպքում «սկալյար արտադրյալ» տերմինը **MatLab**–ում վերաբերում է կամայական երկարության վեկտորներին: Այն կարելի է հաշվել՝ օգտագործելով սովորական բազմապատկման գործողությունը (a*b): Բացի դրանից, **MatLab**–ում երկու վեկտորների սկալյար արտադրյալը հաշվելու համար սահմանված է նաև ներդրված **dot(a, b)** ֆունկցիան:

Օրինակ.

Ենթադրենք՝ տրված են $\vec{a}=5\hat{i}-4\hat{j}+\hat{k}$ և $\vec{b}=-8\hat{i}+3\hat{j}+9\hat{k}$ վեկտորները: Հաշվենք այս վեկտորների սկալյար արտադրյալը՝ օգտվելով բազմապատկման գործողությունից, (2.3) բանաձևից և **dot** ֆունկցիայից: Դրա համար $\vec{a}$ վեկտորը կսահմանենք որպես տող–վեկտոր, որի տարրերը $\vec{a}$ –ի x, y, z բաղադրիչներն են, իսկ $\vec{b}$ վեկտորը՝ որպես սյուն–վեկտոր, որի տարրերն էլ $\vec{b}$ –ի x, y, z բաղադրիչներն են:

```
>> a = [5 –4 1];
>> b = [–8; 3; 9];
>> p1 = a*b
   p1 = –43
>> p2 = a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
```

```
   p2 = -43
>> p3 = dot(a, b)
   p3 = -43
```

Ինչպես տեսնում ենք՝ երեք եղանակով էլ ստանում ենք միևնույն արդյունքը:

Սյուն–վեկտորի և տող–վեկտորի արտադրյալը

Այժմ ենթադրենք՝ a սյուն–վեկտորն ենք բազմապատկում նույն երկարությունն ունեցող b տող–վեկտորով: Այս դեպքում, քանի որ տող–վեկտորը $n\times 1$ չափի է, իսկ սյուն–վեկտորը՝ $1\times n$ չափի, ապա արդյունքում կունենանք $n\times n$ չափի քառակուսի մատրից:

Օրինակ.

```
>> a = [7; 3; 8; -1];
>> b = [-6 2 4 3];
>> p = a*b
   p =
      -42    14    28    21
      -18     6    12     9
      -48    16    32    24
        6    -2    -4    -3
```

Մատրիցների հետ աշխատանքին առավել մանրամասն կծանոթանանք 4-րդ գլխում:

Երկու վեկտորների վեկտորական արտադրյալը

Վեկտորական արտադրյալը **MatLab**–ում սահմանված է միայն եռաչափ վեկտորների համար և համարժեք է մաթեմատիկայում երկու վեկտորների վեկտորական արտադրյալին, որն, ինչպես հայտնի է, որոշվում է հետևյալ բանաձևով.

$$\vec{a}\times\vec{b}=\hat{i}(a_yb_z-a_zb_y)+\hat{j}(a_zb_x-a_xb_z)+\hat{k}(a_xb_y-a_yb_x):\qquad(2.4)$$

Եթե դարձյալ x, y, z ինդեքսների փոխարեն օգտագործենք 1, 2, 3 նշանակումը, (2.4) բանաձևը կընդունի հետևյալ տեսքը.

$$\vec{a}\times\vec{b}=\hat{i}(a_2b_3-a_3b_2)+\hat{j}(a_3b_1-a_1b_3)+\hat{k}(a_1b_2-a_2b_1):\qquad(2.5)$$

Չնայած վեկտորական արտադրյալը կարելի է հաշվել (2.5) բանաձևով, սակայն **MatLab**–ում նախատեսված է նաև $\vec{a}$ և $\vec{b}$ վեկտորների վեկտորական արտադրյալը հաշվող **cross(a, b)** ֆունկցիան:

Օրինակ.

Տրված են $\vec{a}=4\hat{i}+9\hat{j}-3\hat{k}$ և $\vec{b}=8\hat{i}+\hat{j}+6\hat{k}$ վեկտորները: Հաշվենք այս վեկտորների վեկտորական արտադրյալը՝ օգտվելով (2.5) բանաձևից և **cross** ֆունկցիայից:

```
>> a = [4 9 -3];
>> b = [8 1 6];
```

```
>> p1=[a(2)*b(3)–a(3)*b(2) a(3)*b(1)–a(1)*b(3) a(1)*b(2)–a(2)*b(1)]
   p1 = 57   –48   –68
>> p2 = cross(a, b)
   p2 = 57   –48   –68
```

Երեք վեկտորների արտադրյալը

Երեք վեկտորներ բազմապատկելու համար մաթեմատիկայում սահմանված են խառն արտադրյալը և կրկնակի վեկտորական արտադրյալը: Դրանք կարելի է **MatLab**–ում հաշվել՝ օգտվելով **dot** և **cross** ֆունկցիաներից:

Երեք վեկտորների խառն արտադրյալը, ինչպես հայտնի է, ունի $\vec{a}\cdot(\vec{b}\times\vec{c})$ տեսքը և բավարարում է ցիկլիկ տեղափոխության կանոնին.

$$\vec{a}\cdot(\vec{b}\times\vec{c})=\vec{c}\cdot(\vec{a}\times\vec{b})=\vec{b}\cdot(\vec{c}\times\vec{a}): \tag{2.6}$$

Օրինակ.

Ենթադրենք՝ տրված են $\vec{a}=2\hat{i}-3\hat{j}+8\hat{k}$, $\vec{b}=9\hat{i}+7\hat{j}+5\hat{k}$ և $\vec{c}=-4\hat{i}+\hat{j}+3\hat{k}$ վեկտորները: Հաշվենք այս վեկտորների խառն արտադրյալը և համոզվենք ցիկլիկ տեղափոխության կանոնի իրավացիության մեջ:

```
>> a = [2 –3 8];
>> b = [9 7 5];
>> c = [–4 1 3];
>> p1 = dot(a, cross(b,c))
   p1 = 469
>> p2 = dot(c, cross(a,b))
   p2 = 469
>> p3 = dot(b, cross(c,a))
   p3 = 469
```

Երեք վեկտորների կրկնակի վեկտորական արտադրյալը ունի $\vec{a}\times(\vec{b}\times\vec{c})$ տեսքը և բավարարում է հետևյալ կանոնին.

$$\vec{a}\times(\vec{b}\times\vec{c})=\vec{b}(\vec{a}\cdot\vec{c})-\vec{c}(\vec{a}\cdot\vec{b}): \tag{2.7}$$

Օրինակ.

Հաշվենք նախորդ օրինակում սահմանված $\vec{a}$, $\vec{b}$, $\vec{c}$ վեկտորների կրկնակի վեկտորական արտադրյալը և համոզվենք, որ (2.7) բանաձևը ճիշտ է.

```
>> p4 = cross(a, cross(b,c))
   p4 = 265    54   –46
>> p5 = b*dot(a,c) – c*dot(a,b)
   p5 = 265    54   –46
```

2.4.4. Անդամ առ անդամ բազմապատկում, բաժանում, աստիճանի բարձրացում

Հաճախ անհրաժեշտ է լինում վեկտորների հետ կատարել անդամ առ անդամ բազմապատկման, բաժանման և աստիճան բարձրացնելու գործողություններ: Երկու վեկտորների գումարումը, հանումը կամ սկալյարի ու վեկտորի բազմապատկումը, ինչպես տեսանք անդամ առ անդամ գործողություններ էին, սակայն բազմապատկման կամ բաժանման դեպքում այդպես չէ, քանի որ վեկտորները **MatLab**–ում պահվում են մատրիցների տեսքով և, հետևաբար, այս գործողությունները կատարվում են մատրիցական հանրահաշվի օրենքներով:

Սակայն **MatLab**–ում նախատեսված են նաև բազմապատկման, բաժանման և աստիճան բարձրացնելու անդամ առ անդամ գործողություններ ևս: Որպեսզի այս գործողությունները կատարվեն անդամ առ անդամ, բազմապատկման, բաժանման և աստիճան բարձրացնելու օպերատորներից առաջ պետք է դնել կետ (առանց բացատ դնելու): Այդ օպերատորները ներկայացված են հետևյալ աղյուսակում.

Անդամ առ անդամ գործողություններ	
Օպերատորը	**Նկարագրությունը**
.*	վեկտորների անդամ առ անդամ բազմապատկում
./	վեկտորների անդամ առ անդամ աջ բաժանում
.\	վեկտորների անդամ առ անդամ ձախ բաժանում
.^	վեկտորների անդամ առ անդամ աստիճանի բարձրացում

Անդամ առ անդամ գործողություններ կատարելու ժամանակ անհրաժեշտ է հաշվի առնել, որ վեկտորները պետք է լինեն կամ միայն տող, կամ միայն սյուն և ունենան նույն երկարությունը:

Օրինակներ.

```
>> a = [2 4 3 0];
>> b = [1 3 –2 –1];
>> c = a.*b
   c = 2    12    –6     0
>> d = a./b
   d = 2.0000    1.3333    –1.5000    0
>> f = a.\b
   f = 0.5000    0.7500    –0.6667    –Inf
>> g = a.^b
   g = 2.0000   64.0000    0.1111     Inf
```

2.5. ՖՈՒՆԿՑԻԱՅԻ ԱՐԺԵՔՆԵՐԻ ԱՂՅՈՒՍԱԿԱՅԻՆ ՆԵՐԿԱՅԱՑՈՒՄ

Հաճախ պետք է լինում ֆունկցիայի արժեքը որոշել ոչ թե արգումենտի մեկ արժեքի համար, այլ արգումենտների որևէ հաջորդականության համար, այսինքն՝ ֆունկցիայի արժեքները ներկայացնել աղյուսակային տեսքով: Այդ խնդիրները լուծվում են երկու քայլով.

1. Ստեղծվում է մուտքային արգումենտի վեկտորը տողի կամ սյան տեսքով, որի տարրերը արգումենտի այն արժեքներն են, որոնց համար ուզում ենք հաշվել ֆունկցիայի արժեքները,
2. Սահմանված յուրաքանչյուր արգումենտի համար որոշվում է ֆունկցիայի արժեքը, և ստացված արդյունքները ներկայացվում են վեկտորի տեսքով: Քանի որ ֆունկցիայի արժեքները հաշվում ենք արգումենտի յուրաքանչյուր արժեքի համար, ապա բոլոր թվաբանական գործողությունները պետք է կատարվեն անդամ առ անդամ:

***Օրինակ** 1.*

Որոշենք $y = x^2 - 4x$ ֆունկցիայի արժեքները x արգումենտի 1, 2, ... 8 արժեքների համար:

```
>> x = [1:8];
>> y = x.^2-4*x;
>> x
   x = 1     2     3     4     5     6     7     8
>> y
   y = -3    -4    -3     0     5    12    21    32
```

***Օրինակ** 2.*

Որոշենք $y = \dfrac{z^3 + 5z}{4z^2 - 10}$ ֆունկցիայի արժեքները x արգումենտի 1, 3, 5, ... 15 արժեքների համար:

```
>> z = [1:2:15];
>> y = (z.^3+5*z)./(4*z.^2-10);
>> z
   z = 1     3     5     7     9    11    13    15
>> y
   y = -1.00  1.42  1.56  1.95  2.40  2.87  3.35   3.84
```

2.6. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Հայտարարեք հետևյալ տարրերից կազմված տող–վեկտորը.
32, 4.81, $e^{2.5}$, 63, $\cos\frac{\pi}{3}$, 14.12:

2. Հայտարարեք հետևյալ տարրերից կազմված սյուն–վեկտորը.
55, 14, $\ln 51$, 987, 0, $5\sin(2.5\pi)$:

3. Հայտարարեք տող–վեկտոր, որի առաջին տարրի արժեքը 1 է, վերջինը՝ 33, իսկ տարրերի միջև հեռավորությունը՝ 2:

4. Հայտարարեք սյուն–վեկտոր, որի առաջին տարրի արժեքը 15 է, վերջին տարրի արժեքը՝ –25, իսկ տարրերի միջև հեռավորությունը՝ 5:

5. Հայտարարեք հավասար հեռավորությամբ դասավորված, 15 տարրերից բաղկացած տող–վեկտոր, որի առաջին տարրի արժեքը 7 է, վերջինի արժեքը՝ 40:

6. Հայտարարեք հավասար հեռավորությամբ դասավորված, 12 տարրերից բաղկացած սյուն–վեկտոր, որի առաջին տարրի արժեքը –1 է, վերջինի արժեքը՝ –15:

7. Օգտվելով միայն **sort** ֆունկցիայից՝ 25, –14, 0, 69.2, –7, –12.9, 78 տարրերից բաղկացած վեկտորը դասավորեք նվազման կարգով:

8. Տրված են երկու վեկտոր՝ $\vec{a} = 4\hat{i} + 9\hat{j} - 5\hat{k}$, $\vec{b} = -3\hat{i} + 6\hat{j} - 7\hat{k}$: Հաշվեք $\vec{a}\cdot\vec{b}$ սկալյար արտադրյալը, $\vec{a}\times\vec{b}$ վեկտորական արտադրյալը և $\vec{a}$ վեկտորի մոդուլը:

9. Տրված են $\vec{a} = 3\hat{i} + 4\hat{j} - 5\hat{k}$, $\vec{b} = 15\hat{i} - 6\hat{j} + 3\hat{k}$ և $\vec{c} = 8\hat{i} + 4\hat{j} - 10\hat{k}$ վեկտորները: Համոզվեք, որ

- $\vec{a}\times\vec{b} + \vec{b}\times\vec{a} = 0$,
- $(\vec{a}\times\vec{b})\cdot(\vec{c}\times\vec{d}) = (\vec{a}\cdot\vec{c})(\vec{b}\cdot\vec{d}) - (\vec{a}\cdot\vec{d})(\vec{b}\cdot\vec{c})$,
- $(\vec{a}\times\vec{b})\times(\vec{c}\times\vec{d}) = (\vec{a}\cdot(\vec{b}\times\vec{d}))\vec{c} - (\vec{a}\cdot(\vec{b}\times\vec{c}))\vec{d} = (\vec{a}\cdot(\vec{c}\times\vec{d}))\vec{b} - (\vec{b}\cdot(\vec{c}\times\vec{d}))\vec{a}$:

10. Ջրհորի d խորությունը կարելի է որոշել՝ քարը նետելով նրա մեջ և հաշվելով այն ժամանակը, որի ընթացքում կլսվի քարի ձայնը $d = \frac{gt^2}{2}$ բանաձևով, որտեղ $g = 9.81$ մ/վ^2: Որոշեք ջրհորների խորությունները, եթե քարի ձայնը լսվել է $t \in 1, 2, \ldots, 10$ վ ժամանակ հետո:

11. $x \in [-2.5;\ 3]$ միջակայքում $\Delta x = 0.5$ քայլով որոշեք $y = (x^3 - 2)^4 x^3$ ֆունկցիայի արժեքները:

12. $x \in [0.2;\ 2.5]$ միջակայքում $\Delta x = 0.2$ քայլով որոշեք $y = \dfrac{\sin^3 x}{1+\cos x} + e^{-2x} \ln x$ ֆունկցիայի արժեքները:

13. Որոշեք $z = \dfrac{xy + \dfrac{y}{x}}{(x+y)^{(y-x)}} + 12^{\frac{x}{y}}$ ֆունկցիայի արժեքները, եթե $x = [2,\ 4,\ 6,\ 8,\ 10]$ և $y = [3,\ 6,\ 9,\ 12,\ 15]$:

14. Որոշեք $T = \dfrac{xyz}{(h+k)^{\frac{k}{5}}} + \dfrac{ke^{\left(\frac{z}{x}+y\right)}}{z^h}$ ֆունկցիայի արժեքները, եթե $h = 0.9$, $k = 12.5$, $x = [1,\ 2,\ 3,\ 4]$, $y = [0.9,\ 0.8,\ 0.7,\ 0.6]$ և $z = [2.5,\ 3,\ 3.5,\ 4]$:

15. Հայտնի է, որ e թիվը կարելի է ներկայացնել $e = \lim\limits_{n\to\infty}\left(1+\dfrac{1}{n}\right)^n$ սահմանի տեսքով: $n = [1,\ 10,\ 100,\ 500,\ 1000,\ 2000,\ 8000]$ արժեքների համար հաշվեք $\left(1+\dfrac{1}{n}\right)^n$ ֆունկցիայի արժեքները և համոզվեք, որ n թվի մեծանալուն զուգընթաց դրանք ձգտում են e թվի արժեքին:

16. Հայտնի է, որ $\sum\limits_{n=1}^{\infty} \dfrac{1}{n^2} = \dfrac{\pi^2}{6}$: n=[50, 100, 500, 5000, 10000] արժեքների համար հաշվեք այս գումարը և համոզվեք, որ n –ի մեծանալուն զուգընթաց այն ձգտում է $\dfrac{\pi^2}{6}$ –ին:

17. Հայտնի է, որ $\sum\limits_{n=0}^{\infty} \dfrac{1}{(2n+1)(2n+2)} = \ln 2$: n=[20, 100, 200, 800, 1000, 2000, 5000] արժեքների համար հաշվեք այս գումարը և համոզվեք, որ n –ի մեծանալուն զուգընթաց այն ձգտում է $\ln 2$ –ին:

3. ԳՐԱՖԻԿՆԵՐ

Հաճախ շատ ավելի հարմար է տվյալները և դրանց վարքագիծը տեսնել գրաֆիկի, քան պարզապես թվերի տեսքով: **MatLab**–ն օժտված է տվյալների գրաֆիկական ներկայացման լայն հնարավորություններով: Այն թույլ է տալիս կառուցել ինչպես հայտնի, անալիտիկ տեսքով տրված բացահայտ կամ անբացահայտ ֆունկցիաների գրաֆիկներ, այնպես էլ գրաֆիկորեն պատկերել, օրինակ, որևէ փորձի արդյունքում ստացված տվյալները: Բացի այդ, գրաֆիկները հնարավոր է կառուցել գծային կամ լոգարիթմական մասշտաբներով, բևեռային կոորդինատներով, կարելի է ստեղծել շարժապատկերներ և այլն:

Այս գլխում կծանոթանանք մեկ փոփոխականից կախված ֆունկցիաների գրաֆիկական ներկայացման տարբեր եղանակների հետ: Սակայն, մինչ այդ ծանոթանանք ավելի պարզ գրաֆիկական պատկերների հետ:

3.1. ԿԵՏԻ ԵՎ ԿԵՏԵՐԻ ՀԱՋՈՐԴԱԿԱՆՈՒԹՅԱՆ ԳՐԱՖԻԿԱԿԱՆ ՆԵՐԿԱՅԱՑՈՒՄՆԵՐԸ

Որևէ (x, y) կոորդինատներով կետ գրաֆիկորեն պատկերելու համար նախատեսված է **plot(x,y)** ֆունկցիան:

Օրինակ.

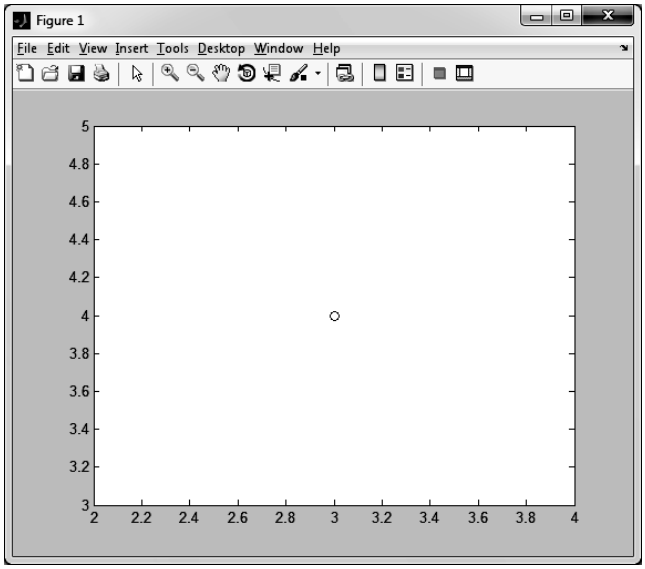
```
>> plot(3,4)
```

Այս հրամանի իրականացման արդյունքում կհայտնվի նոր պատուհան, որը կոչվում է գրաֆիկների պատուհան: Այդ պատուհանն ունի **Figure** անվանումը, որին հետևում է բացված պատուհանի համարը (օրինակ՝ **Figure 1**): Գրաֆիկների պատուհանում երևում են x, y առանցքները և $(3, 4)$ կոորդինատներով կետը: Եթե հատուկ չի նշվում, կետը պատկերվում է կապույտ գույնով: Սակայն այս օրինակում գրված հրամանով ստացված պատկերն ունի թերություն՝ կետը շատ փոքր է և գրեթե չի երևում գրաֆիկական պատուհանում: **MatLab**–ը հնարավորություն է տալիս կետի փոխարեն օգտագործել այլ սիմվոլներ, օրինակ, այն պատկերել քառակուսու, օղակի, շեղանկյան և այլ տեսքերով, ինչպես նաև փոխել կետի գույնը: Այս հատկություններին մանրամասն կծանոթանանք հաջորդ՝ ֆունկցիաների գրաֆիկական ներկայացման բաժնում: Այստեղ դիտարկենք մի պարզ դեպք: Ենթադրենք՝ ցանկանում ենք $(3, 4)$ կոորդինատներով կետը պատկերել սև գույնով և օղակի տեսքով: Դրա համար անհրաժեշտ է գրել հետևյալ հրամանը.

```
>> plot(3,4,'ko')
```

Ինչպես տեսնում ենք, **plot** ֆունկցիան կարող է պարունակել նաև երրորդ արգումենտ, որը գրվում է ապոստրոֆներում: Հետագայում կտեսնենք, որ **MatLab**–ում

ապոստրոֆներում գրվում են սիմվոլային փոփոխականները: Հետևաբար, **plot** ֆունկցիան իր երրորդ արգումենտն ընդունում է որպես տող: Այս տողի մեջ k տառը վերաբերում է կետի գույնին (անգլերեն black բառի վերջին տառը), իսկ o տառը՝ կետի տեսակին (օղակաձև): Գրաֆիկական պատուհանում պատկերվող կետերը կոչվում են նաև մարկերներ: Հետագա շարադրանքում հիմնականում կօգտագործենք «մարկեր» տերմինը: Վերևում գրված հրամանի արդյունքը պատկերված է նկ. 4–ում:



Նկ. 4

Եթե ունենք (x_i, y_i) կոորդինատներով կետերի բազմություն, ապա դրանք ևս **plot** ֆունկցիայի օգնությամբ կարելի է պատկերել գրաֆիկական պատուհանում: Այս դեպքում անհրաժեշտ է կատարել հետևյալ քայլերը.

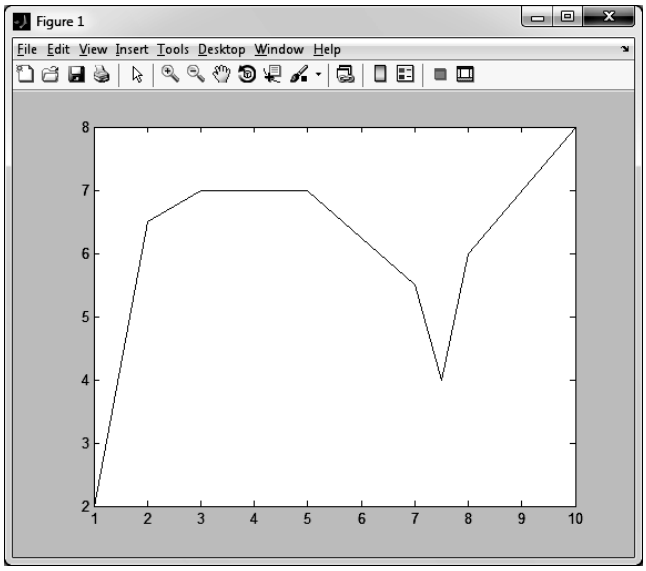
1. որպես վեկտոր սահմանել կետերի x_i կոորդինատները,
2. որպես վեկտոր սահմանել կետերի y_i կոորդինատները,
3. **plot** ֆունկցիայի օգնությամբ սահմանված կետերը պատկերել գրաֆիկական պատուհանում:

Օրինակ.

Ենթադրենք, ցանկանում ենք գրաֆիկորեն պատկերել $(1,\ 2)$, $(2,\ 6.5)$, $(3,\ 7)$, $(5,\ 7)$, $(7,\ 5.5)$, $(7.5,\ 4)$, $(8,\ 6)$, $(10,\ 8)$ կոորդինատներով կետերը: Վերևում նշված քայլերի իրականացումը ծրագրում կունենա հետևյալ տեսքը.

```
>> x = [1 2 3 5 7 7.5 8 10];
>> y = [2 6.5 7 7 5.5 4 6 8];
>> plot(x,y)
```

Ստացված գրաֆիկը պատկերված է նկ. 5–ում:



Նկ. 5

Ինչպես տեսնում ենք, **plot(x,y)** հրամանի արդյունքում սահմանված կետերը (մարկերները) չեն երևում: Դրա փոխարեն **plot** ֆունկցիան յուրաքանչյուր երկու հարևան կետերն իրար է միացրել ուղիղ գծերով: Եթե հատուկ չի նշվում, ապա գրաֆիկը պատկերվում է կապույտ, հոծ գծով: **plot** ֆունկցիան կարելի է գրել նաև այնպես, որ երևան միայն մարկերները (ուղիղ գծերը չերևան) կամ երևան և՛ մարկերները, և՛ վերջիններս միացնող գծերը: Դրա համար դարձյալ պետք է օգտվել **plot** ֆունկցիայի ոչ պարտադիր, երրորդ արգումենտից: Կրկին, առանց մանրամասների մեջ մտնելու (մանրամասները կբերվեն հաջորդ բաժնում), դիտարկենք երկու դեպք:

Եթե ցանկանում ենք, որ կետերը պատկերվեն աստղանիշի տեսքով և միացնող գիծը չերևա, ապա ծրագիրը կգրվի հետևյալ կերպ.

```
>> x = [1 2 3 5 7 7.5 8 10];
>> y = [2 6.5 7 7 5.5 4 6 8];
>> plot(x,y,'*')
```

Ծրագրի արդյունքը պատկերված է նկ. 6–ում: Ինչպես տեսնում ենք, մարկերները միացնող գծերը գրաֆիկական պատուհանում չեն երևում: Պատճառն այն է, որ մենք **plot** ֆունկցիայի երրորդ արգումենտում նշել ենք միայն մարկերի տեսակը (*): Եթե երրորդ արգումենտում նշվում է մարկերի տեսակը, ապա, որպեսզի դրանք իրար միացնող գիծը ևս երևա, դա նույնպես պետք է հայտարարել երրորդ արգումենտում: Հոծ գիծ պատկերելու համար երրորդ արգումենտում աստղանիշի կողքին պետք է ավելացնել հոծ գծի (–) նշանը.

```
>> x = [1 2 3 5 7 7.5 8 10];
>> y = [2 6.5 7 7 5.5 4 6 8];
>> plot(x,y,'-*')
```

Ծրագրի իրականացման արդյունքում ստացված գրաֆիկը բերված է նկ. 7–ում:

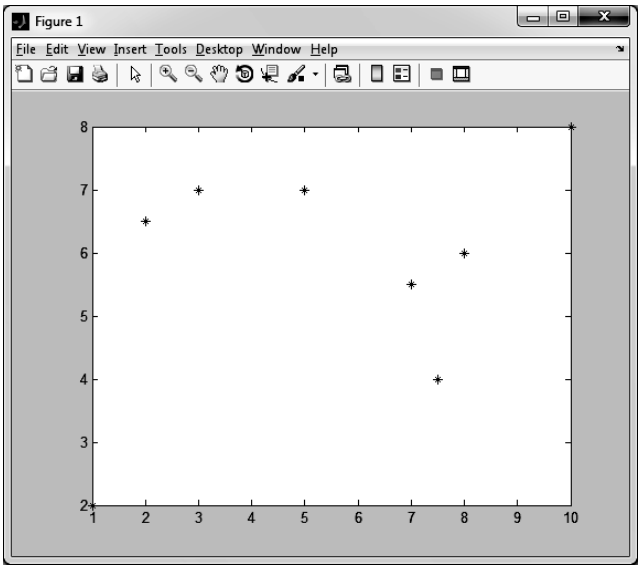


Figure 1
File Edit View Insert Tools Desktop Window Help
8
7
6
5
4
3
2
1 2 3 4 5 6 7 8 9 10

Նկ. 6

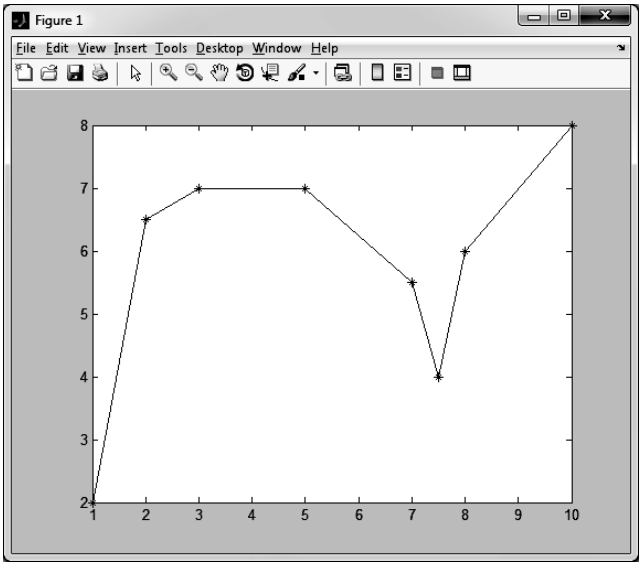


Figure 1
File Edit View Insert Tools Desktop Window Help
8
7
6
5
4
3
2
1 2 3 4 5 6 7 8 9 10

Նկ. 7

3.2. ՖՈՒՆԿՑԻԱՆԵՐԻ ԳՐԱՖԻԿԱԿԱՆ ՆԵՐԿԱՅԱՑՈՒՄՆԵՐԸ

Նախորդ բաժնում շարադրվածից պարզ է դառնում, թե ինչպես կարելի է ստանալ մեկ փոփոխականից կախված որևէ $y(x)$ ֆունկցիայի գրաֆիկը: Այն կառուցելու համար պարզագույն դեպքում պետք է կատարենք հետևյալ քայլերը.

1. սահմանենք x մուտքային արգումենտի արժեքները վեկտորի տեսքով (դրանք կլինեն այն կետերը, որի համար ցանկանում ենք որոշել ֆունկցիայի արժեքները),
2. յուրաքանչյուր մուտքային արգումենտի համար որոշենք $y(x)$ ֆունկցիայի արժեքը, և ստացված արդյունքները ներկայացնենք y վեկտորի տեսքով (հիշենք, որ այս դեպքում բոլոր թվաբանական գործողությունները պետք է կատարել անդամ առ անդամ),
3. **plot** ֆունկցիայի օգնությամբ կառուցենք ֆունկցիայի գրաֆիկը:

Գրաֆիկի կառուցման սկզբունքն այս դեպքում ամբողջությամբ նույնն է, ինչ կետերի բազմության գրաֆիկական պատկերման դեպքում, որը դիտարկեցինք նախորդ բաժնում (իրոք, այստեղ սահմանված արգումենտների և ֆունկցիաների արժեքները, ի վերջո, ևս կետերի բազմություն են): (x,y) հարթության մեջ հերթով դրվում են սահմանված (x_i,y_i) կետերը, և բոլոր հարևան կետերը միացվում են ուղիղ գծերով: Ակնհայտ է, որ որքան շատ լինեն սահմանված (x_i,y_i) կետերը, այսինքն՝ մեծ լինեն x և y վեկտորների երկարությունները, այնքան գրաֆիկն ավելի ճշգրիտ տեսք կունենա: Ինչպես արդեն նշել ենք, եթե հատուկ չի նշվում, ապա գրաֆիկը կառուցվում է հոծ, կապույտ գծով՝ առանց սահմանված (x_i,y_i) կետերը (մարկերները) պատկերելու:

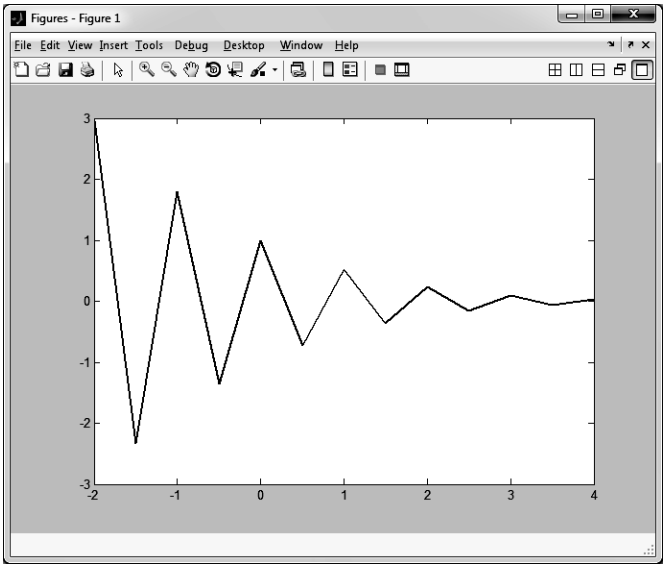
***Օրինակ** 1.*

Կառուցենք $y(x)=3.5^{-0.5x}\cos 6x$ ֆունկցիայի գրաֆիկը $-2\le x\le 4$ միջակայքում, $\Delta x=0.5$ քայլով:

```
>> x = [-2:0.5:4];
>> y = 3.5.^(-0.5*x).*cos(6*x);
>> plot(x, y)
```

Ֆունկցիայի գրաֆիկը պատկերված է նկ. 8–ում: Ինչպես տեսնում ենք, ստացված գրաֆիկը բավական կոպիտ է և այնքան էլ ճշգրիտ չի ցույց տալիս ֆունկցիայի վարքը: Պատճառն այն է, որ այս օրինակում շատ քիչ կետեր վերցրեցինք ($\Delta x=0.5$ քայլով $-2\le x\le 4$ միջակայքում կա ընդամենը 13 կետ), և այդ քանակությունը բավարար չէ ֆունկցիայի ճիշտ վարքագիծը գրաֆիկորեն պատկերելու համար: Հիմա ցույց

տանք, որ փոքրացնելով Δx քայլը՝ կարելի է ավելի ճշգրիտ ու սահուն գրաֆիկ ստանալ:



Նկ. 8

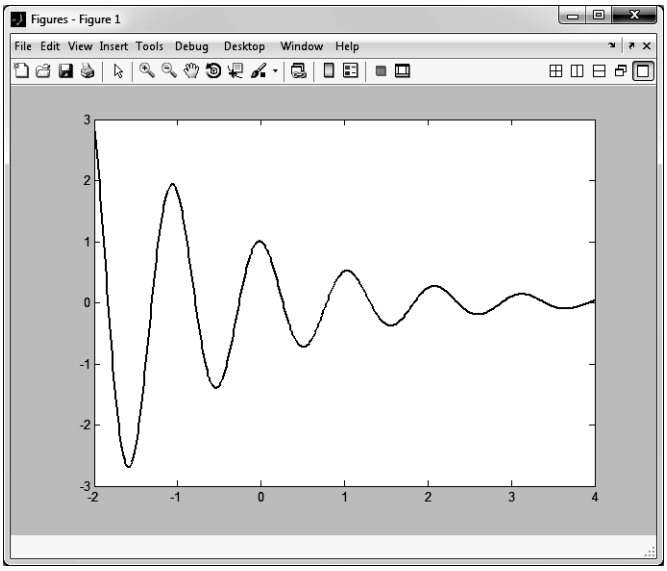
***Օրինակ** 2.*

Դարձյալ $-2 \le x \le 4$ միջակայքում, բայց այս անգամ ավելի փոքր՝ $\Delta x = 0.01$ քայլով կառուցենք նախորդ օրինակում սահմանված ֆունկցիայի գրաֆիկը.

```
>> x = [-2:0.01:4];
>> y = 3.5.^(-0.5*x).*cos(6*x);
>> plot(x, y)
```

Ի տարբերություն նախորդ օրինակի, այս դեպքում սահմանված կետերը բավականին շատ են ($\Delta x = 0.01$ քայլով $-2 \le x \le 4$ միջակայքում կա 601 կետ): Սա արդեն բավական է ֆունկցիայի վարքագիծն ավելի ճշգրիտ պատկերելու համար: Թեև **MatLab**–ը սահմանված 601 կետերն իրար է միացնում ուղիղ գծերով, գրաֆիկը բավականին սահուն տեսք է ունենում (նկ. 9): Որքան կետերի քանակը շատացնենք (այսինքն՝ քայլը փոքրացնենք), այնքան գրաֆիկն ավելի ճշգրիտ տեսք կստանա, սակայն համակարգչի համար կարող է երկար ժամանակ պահանջվել խնդիրը լուծելու համար:

Այժմ ավելի մանրամասն տեսնենք, թե ինչպես կարելի է փոխել գրաֆիկի գծի ձևը, գույնը և ավելացնել տարբեր տեսքի մարկերներ: Նախորդ օրինակներից տեսանք, որ գրաֆիկի այդ հատկությունները գրվում են **plot** ֆունկցիայի երրորդ, ոչ պարտադիր արգումենտում՝ տողի տեսքով, այսինքն՝ ապոստրոֆներում:



Նկ. 9

Գրաֆիկների հատկությունները բերված են հետևյալ աղյուսակում.

Գրաֆիկի գծի ձևը	
–	հոծ գիծ (default)
– –	գծիկավոր
:	կետիկավոր
–.	կետագծիկավոր
Գրաֆիկի գծի ձևը	
b	կապույտ (**b**lue)
r	կարմիր (**r**ed)
g	կանաչ (**g**reen)
k	սև (blac**k**)
m	մորեգույն (**m**agenta)
y	դեղին (**y**ellow)
c	երկնագույն (**c**yan)
w	սպիտակ (**w**hite)

Մարկերի տեսակը	
+	գումարման նշան
o	օղակ
*	աստղանիշ
.	կետ
s	քառակուսի
d	շեղանկյուն
p	հնգաթև աստղ
h	վեցաթև աստղ
x	խաչաձև
v	ներքև ուղղված եռանկյուն
^	վերև ուղղված եռանկյուն
<	ձախ ուղղված եռանկյուն
>	աջ ուղղված եռանկյուն

Գրաֆիկի հատկություններն ավելացնելիս պետք է հաշվի առնել, որ

1. դրանք գրվում են **plot** ֆունկցիայի երրորդ արգումենտում որպես տող, այսինքն՝ ապոստրոֆների մեջ,
2. դրանց հերթականությունն այդ տողի մեջ կարևոր չի: Օրինակ՝ **plot(x, y, ´ –go´)**, **plot(x, y, ´ g–o´)** կամ **plot(x, y, ´ o –g´)** հրամանները հա-

մարժեք են: Երեք դեպում էլ գրաֆիկը կառուցվում է հոծ գծով, կանաչ գույնի և օղակաձև մարկերներով,

3. դրանք գրելը պարտադիր չէ: Դա նշանակում է, որ **plot** ֆունկցիան կարող է պարունակել 3, 2 կամ 1 հատկություն, ինչպես նաև կարող է ընդհանրապես որևէ հատկություն չպարունակել: Անհրաժեշտ է նաև հիշել, որ երբ գրվում է երրորդ արգումենտը, որում նշված է մարկերի տեսակը, բայց բացահայտ գրված չէ գրաֆիկի գծի ձևը, ապա գրաֆիկը կկառուցվի առանց գծի՝ կերևան միայն մարկերները:

Օրինակներ.

1. **plot(x, y)** – այս հրամանում գրաֆիկի որևէ հատկություն գրված չէ: Հետևաբար, գրաֆիկը կկառուցվի հոծ, կապույտ գծով, առանց մարկերների:
2. **plot(x, y, 'r')** – այս հրամանում գրաֆիկի հատկությունների արգումենտը պարունակում է միայն գույնը (r): Հետևաբար, գրաֆիկը կկառուցվի հոծ, կարմիր գծով, առանց մարկերների:
3. **plot(x,y,'––y')** – այս հրամանում գրաֆիկի հատկությունների արգումենտը պարունակում է գծի տեսակը (––) և գույնը (y): Հետևաբար, գրաֆիկը կկառուցվի գծիկավոր, դեղին գույնի և առանց մարկերների:
4. **plot(x, y, 'x')** – այս հրամանում գրաֆիկի հատկությունների արգումենտը պարունակում է միայն մարկերը (x): Հետևաբար, գրաֆիկի պատուհանում կերևան միայն կապույտ գույնի խաչաձև մարկերներ:
5. **plot(x, y, 'g–.d')** – այս հրամանում գրաֆիկի հատկությունների արգումենտը պարունակում է բոլոր հատկությունները՝ գույնը (g), գծի ձևը (–.) և մարկերի տեսակը (d): Հետևաբար, գրաֆիկը կկառուցվի կանաչ գույնի կետագծիկավոր և շեղանկյան տեսք ունեցող մարկերներով:

3.3. ԳՐԱՖԻԿՆԵՐԻ ԿԱՌՈՒՑՈՒՄԸ ԼՈԳԱՐԻԹՄԱԿԱՆ ՄԱՍՇՏԱԲՈՎ

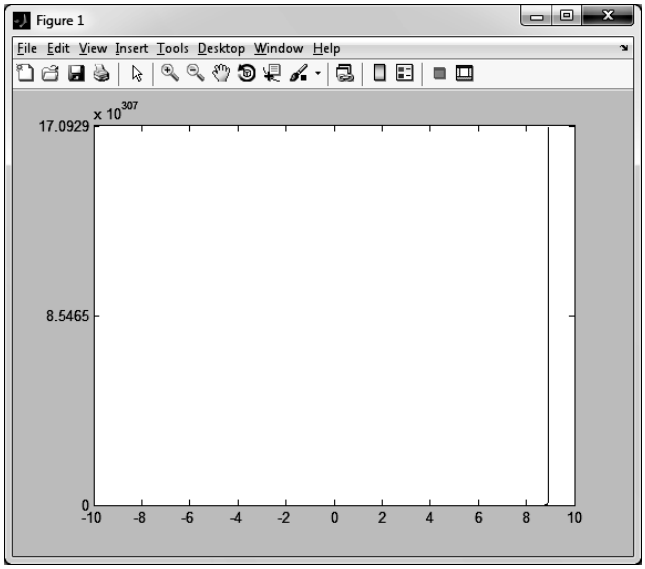
plot ֆունկցիան գրաֆիկները կառուցում է գծային մասշտաբներով: Սակայն ոչ միշտ է դա հարմար: Երբեմն տվյալների թվային արժեքներն այնպիսի մեծ տիրույթ են ընդգրկում, որ գծային մասշտաբում գրաֆիկը կառուցելիս դրանց վարքագիծը հասկանալ չի լինում: **MatLab**–ում սահմանված են գրաֆիկներ կառուցելու երեք ֆունկցիաներ՝ **semilogx**, **semilogy** և **loglog**, որոնք թույլ են տալիս տվյալները ներկայացնել լոգարիթմական մասշտաբներով: Որոշ խնդիրների դեպքում այս մասշտաբներով շատ ավելի հասկանալի է դառնում տվյալների վարքը: **semilogx**, **semilogy** և **loglog** ֆունկցիաների գրելաձևը **MatLab**–ում նույնն է, ինչ **plot** ֆունկցիայի գրելաձևը: **semilogx** ֆունկցիան միայն աբսցիսների առանցքն է ներկայաց-

նում լոգարիթմական մասշտաբով, **semilogy**–ը՝ միայն օրդինատների առանցքը, իսկ **loglog**–ը՝ երկուսն էլ:

Օրինակ.

Ենթադրենք՝ $-10 \le x \le 10$ միջակայքում ցանկանում ենք կառուցել $y = e^{x^3}$ ֆունկցիայի գրաֆիկը: Եթե դիմենք **plot** ֆունկցիային, ապա ծրագիրը կունենա ստորև բերված տեսքը, իսկ իրականացման արդյունքը պատկերված է նկ. 10–ում.

```
>> x = [-10:0.01:10];
>> y = exp(x.^3);
>> plot(x,y)
```

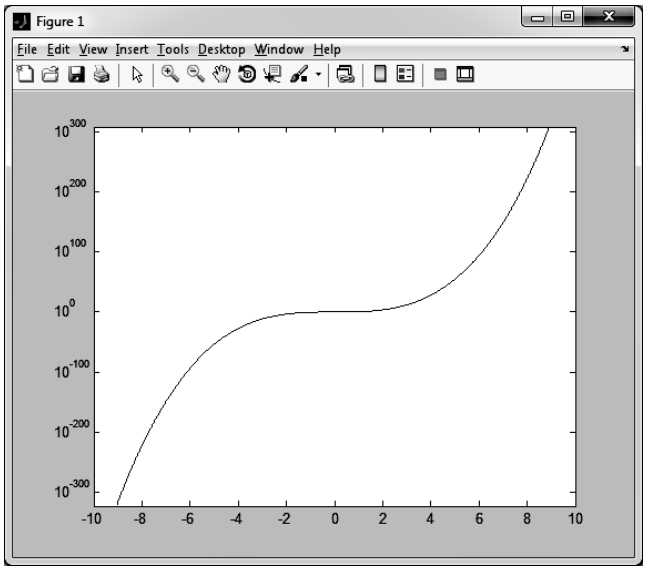


Նկ. 10

Ինչպես տեսնում ենք, գծային մասշտաբների օգտագործումը ոչ մի տեղեկություն չի հաղորդում ֆունկցիայի վարքի մասին: Սակայն, եթե օրդինատների առանցքը ներկայացնենք լոգարիթմական մասշտաբով (աբսցիսների առանցքը թողնելով գծային), ապա գրաֆիկը ֆունկցիայի վարքագծի մասին ավելի շատ տեղեկություն կհաղորդի: Դրա համար անհրաժեշտ է ծրագիրը թողնել նույնը, ընդամենը **plot** ֆունկցիայի փոխարեն գրել **semilogy**:

```
>> x = [-10:0.01:10];
>> y = exp(x.^3);
>> semilogy(x,y)
```

Ծրագրի իրականացման արդյունքը ներկայացված է նկ. 11–ում:



Նկ. 11

3.4. Մի քանի գրաֆիկի կառուցումը նույն գրաֆիկական պատուհանում

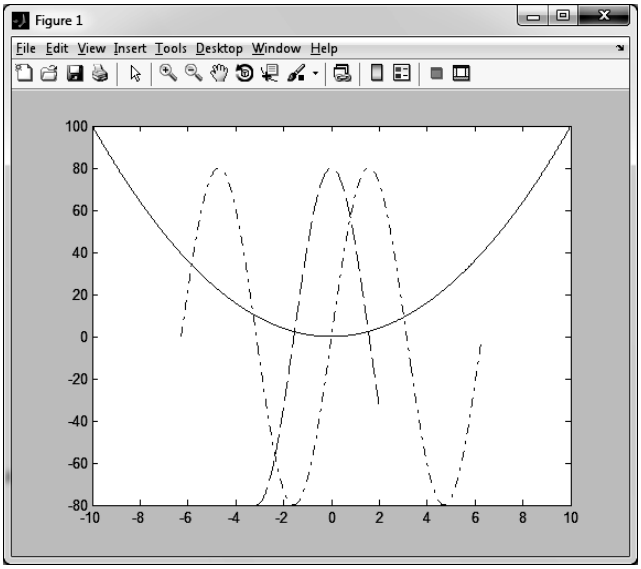
Երբ ծրագրի մեջ հանդիպում է ֆունկցիայի գրաֆիկ կառուցելու **plot** հրամանը, **MatLab**–ը բացում է գրաֆիկական պատուհանը, որի մեջ պատկերվում է **plot** հրամանի իրականացման արդյունքում ստացված գրաֆիկը: Սակայն, երբ նույն ծրագրի մեջ հանդիպում է մեկ այլ **plot** հրաման, **MatLab**–ը գրաֆիկական պատուհանից մաքրում է առաջին գրաֆիկը և գծում նորը: Երբեմն անհրաժեշտ է լինում նույն գրաֆիկական պատուհանում կառուցել ոչ թե մեկ, այլ մի քանի ֆունկցիաների գրաֆիկներ: Այստեղ, կոնկրետ օրինակի վրա, կծանոթանանք նույն գրաֆիկական պատուհանում մի քանի գրաֆիկ կառուցելու երկու եղանակի հետ:

Օրինակ.

Ենթադրենք, ուզում ենք միևնույն պատուհանում կառուցել այս երեք ֆունկցիայի գրաֆիկները՝ $y(x)=x^2$, երբ $-10\le x\le 10$, $f(u)=80\sin u$, երբ $-2\pi\le u\le 2\pi$ և $h(t)=80\cos t$, երբ $-\pi\le u\le 2$: Դրա համար նախ անհրաժեշտ է սահմանել x, u, t անկախ փոփոխականները՝ որպես վեկտորներ, այնուհետև դրանց օգնությամբ՝ y, f և h ֆունկցիաները՝ որպես վեկտորներ: Վերջում անհրաժեշտ է գրել **plot** ֆունկցիան, որի արգումենտում, ստորակետներով անջատված, հերթով կգրվեն յուրաքանչյուր մուտքային արգումենտի վեկտորի և համապատասխան ֆունկցիայի վեկտորի անունները: Այսինքն՝ գրաֆիկը գծելու հրամանը կունենա **plot(x,y,u,f,t,h)** տեսքը: Կարելի է նաև յուրաքանչյուր գրաֆիկի համար գրել նաև իր հատկությունները, օրինակ՝

plot(x,y,'b–',u,f,'–.k',t,h,'––r'): Այս հրամանով $y(x)$ ֆունկցիան կկառուցվի կապույտ, հոծ գծով, $f(u)$ ֆունկցիան՝ սև, կետագծիկներով, իսկ $h(t)$ ֆունկցիան՝ կարմիր, գծիկավոր: Ծրագիրը կունենա ստորև բերված տեսքը, իսկ ծրագրի արդյունքը բերված է նկ. 12–ում:

```
>> x = [-10:0.01:10];
>> y = x.^2;
>> u = [-2*pi:0.01:2*pi];
>> f = 80*sin(u);
>> t = [-pi:0.01:2];
>> h = 80*cos(t);
>> plot(x,y,'b-',u,f,'k-.',t,h,'r--')
```



Նկ. 12

Միևնույն պատուհանում մի քանի գրաֆիկ պատկերելու համար կա մեկ այլ, ավելի հարմար ու ավելի հաճախ օգտագործվող եղանակ: Վերևում բերված առաջին եղանակը հարմար է այն դեպքում, երբ գրաֆիկները նույն պատուհանում պետք է կառուցվեն ծրագրի նույն տեղում: Սակայն ավելի հաճախ առաջին և երկրորդ գրաֆիկների հրամանների միջև գրվում են այլ ծրագրային հրամաններ: Նախորդ օրինակում նախ սահմանեցինք բոլոր երեք ֆունկցիաները և նոր միայն կառուցեցինք դրանց գրաֆիկները: Բայց եթե սահմանեինք առաջին ֆունկցիան, կառուցեինք դրա գրաֆիկը, իսկ երկրորդ ֆունկցիան սահմանեինք դրանից հետո, առաջին եղանակն այլևս կիրառելի չէր լինի: Այսպիսի դեպքերում օգտվում են **MatLab**–ի **hold on** և **hold off** հրամաններից: Երբ առաջին գրաֆիկը կառուցելուց հետո **MatLab**–ը հանդիպում է ծրագրում գրված **hold on** հրամանին, ապա հետագայում մեկ այլ **plot** ֆունկցիայի հանդիպելիս չի մաքրի առաջին գրաֆիկը, այլ երկ-

րորդ գրաֆիկը կկառուցի առաջինի հետ միասին: Դա տեղի կունենա այնքան ժամանակ, քանի դեռ ծրագրում չի հանդիպել **hold off** հրամանը: **hold off** հրամանից հետո առաջին իսկ **plot** ֆունկցիան հանդիպելիս կմաքրվեն նախկինում կառուցված բոլոր գրաֆիկները, և նոր գրաֆիկը կկառուցվի դատարկ պատուհանում:

Նախորդ օրինակն այս եղանակով կունենա հետևյալ տեսքը.

```
>> x = [-10:0.01:10];
>> y = x.^2;
>> plot(x,y,'b-')
>> hold on
>> u = [-2*pi:0.01:2*pi];
>> f = 80*sin(u);
>> plot(u,f,'k-.')
>> t = [-pi:0.01:2];
>> h = 80*cos(t);
>> plot(t,h,'r--')
>> hold off
```

Երբեմն անհրաժեշտ է միևնույն գրաֆիկական պատուհանում կառուցել այնպիսի ֆունկցիաների գրաֆիկներ, որոնց արժեքները միմյանցից խիստ տարբերվում են: Եթե նկարագրված եղանակներով կառուցենք այս ֆունկցիաների գրաֆիկները, ապա այն ֆունկցիայի գրաֆիկը, որի արժեքները շատ փոքր են մյուս ֆունկցիայի արժեքներից, գրեթե չի երևա՝ կխառնվի աբսցիսների առանցքին:

Օրինակ.

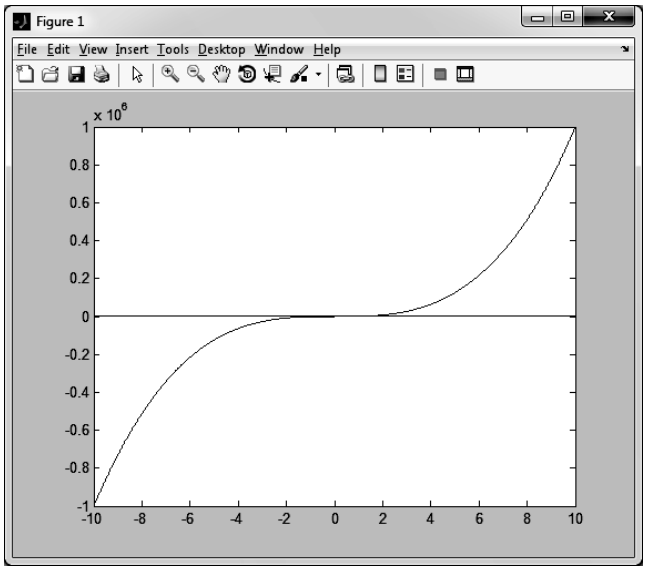
Նույն գրաֆիկական պատուհանում $-10 \le x \le 10$ միջակայքում կառուցենք $y1 = x^2$ և $y2 = 1000x^3$ ֆունկցիաների գրաֆիկները.

```
>> x = [-10:0.01:10];
>> y1 = x.^2;
>> y2 = 1000*x.^3;
>> plot(x,y1,x,y2)
```

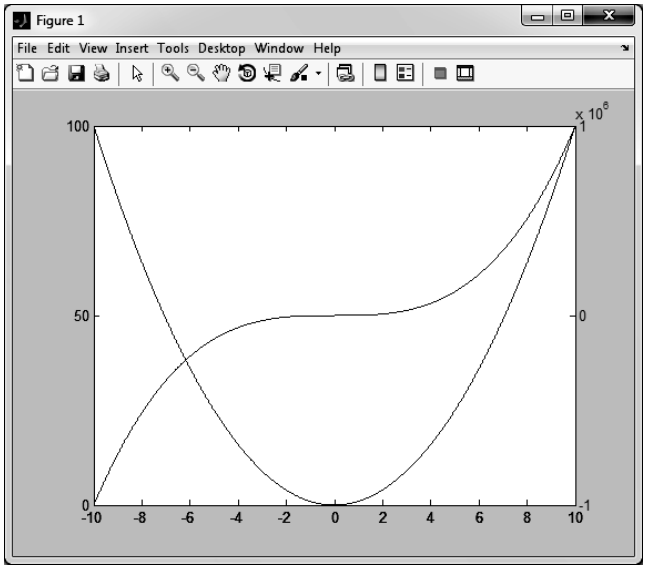
Ինչպես տեսնում ենք նկ. 13–ից, $y1 = x^2$ ֆունկցիայի գրաֆիկը գրեթե չի երևում՝ այն կարծես համընկնում է աբսցիսների առանցքի հետ, քանի որ իր արժեքները անհամեմատ փոքր են $y2 = 1000x^3$ ֆունկցիայի արժեքներից: Այս բարդությունից խուսափելու համար **plot** ֆունկցիայի փոխարեն օգտվում են **MatLab**–ի ներդրված **plotyy** ֆունկցիայից, որը գրաֆիկական պատուհանը կառուցում է համապատասխան ձևով մասշտաբավորված երկու ուղղահայաց առանցքներով.

```
>> x = [-10:0.01:10];
>> y1 = x.^2;
>> y2 = 1000*x.^3;
>> plotyy(x,y1,x,y2)
```

Ինչպես երևում է նկ. 14–ից, արդեն հստակ երևում են երկու ֆունկցիաների գրաֆիկն էլ, ընդ որում ուշադրություն դարձրեք, որ յուրաքանչյուր գրաֆիկի գույնը համընկնում է իրեն համապատասխան ուղղահայաց առանցքի գույնի հետ:



Նկ. 13



Նկ. 14

Երբեմն անհրաժեշտ է լինում նոր ֆունկցիայի գրաֆիկը կառուցել մեկ այլ գրաֆիկական պատուհանում՝ հին պատուհանը թողնելով նույնը: Եթե ծրագրի որևէ հատվածում գրենք **figure** հրամանը, ապա կբացվի նոր, դատարկ գրաֆիկական պատուհան, և այն կդառնա ընթացիկ պատուհանը, այսինքն՝ գրաֆիկներին վերաբերող հետագա բոլոր ֆունկցիաները կվերաբերվեն այդ պատուհանին: Եթե **figure** հրամանի արգումենտում գրենք թիվ, օրինակ՝ **figure(1)** կամ **figure(3)**, ապա

բացված գրաֆիկական պատուհանը կունենա այդ համարը: Եթե, ենթադրենք, աշխատում ենք **figure(3)** պատուհանի հետ և անհրաժեշտ է հետագայում կրկին վերադառնալ առաջին պատուհանին, ապա կգրենք **figure(1)** հրամանը, և այն դարձյալ կդառնա ընթացիկ պատուհանը:

3.5. ԳՐԱՖԻԿԱԿԱՆ ԱՌԱՆՑՔՆԵՐԻ ՀԱՏԿՈՒԹՅՈՒՆՆԵՐԸ

Ինչպես տեսանք, գրաֆիկ կառուցելիս **MatLab**–ն ավտոմատ մասշտաբավորում է առանցքներն այնպես, որ այն ամբողջությամբ ընդգրկի կառուցվող տվյալները: Սակայն երբեմն անհրաժեշտ է լինում գրաֆիկը տեսնել այլ տիրույթներում: **MatLab**–ի **axis** հրամանի օգնությամբ կարելի է փոխել առանցքների հատկությունները: Առավել հաճախ օգտագործվող հրամանները ներկայացված են հետևյալ աղյուսակում:

Գրաֆիկական առանցքների հատկությունները	
Ֆունկցիան	**Նկարագրությունը**
axis([xmin xmax ymin ymax])	**axis** ֆունկցիայի արգումենտում տրվում են x և y առանցքների նվազագույն և առավելագույն արժեքները: Գրաֆիկական պատուհանի առանցքներն ընդունում են այդ սահմանները:
axis auto	առանցքների մասշտաբավորումը բերվում է «լռելյայն» տեսքին, այսինքն՝ այնպես, որ ամբողջությամբ ընդգրկի կառուցվող տվյալները:
axis square	առանցքները կառուցում է քառակուսի չափերով
axis equal	x և y առանցքների մասշտաբավորումը դարձնում է նույնը
axis on	առանցքները երևում են
axis off	առանցքները չեն երևում

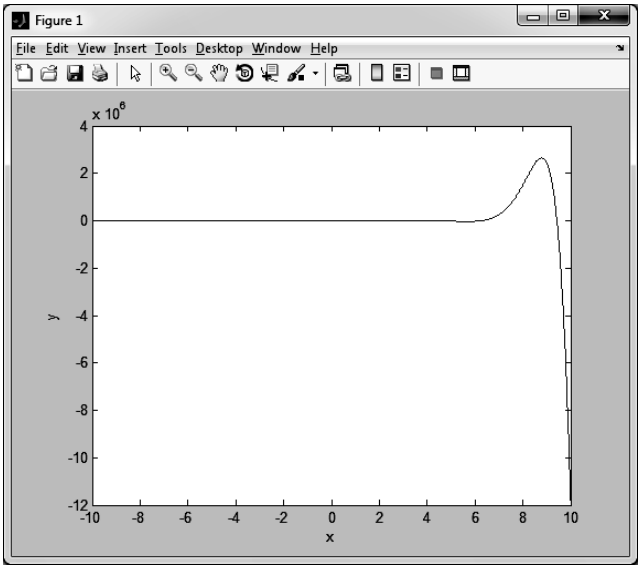
Եթե **MatLab**–ի հրամանի տողում գրենք **help axis** հրամանը, ապա կբերվեն առանցքների հատկությունների ամբողջական ցանկը և նկարագրությունները:

Երբեմն գրաֆիկ կառուցելիս անհրաժեշտ է լինում աշխատել ոչ ամբողջական գրաֆիկի, այլ դրա՝ տվյալ խնդրի համար ավելի կարևոր մի որևէ հատվածի հետ: Իհարկե, միշտ կարելի է գրաֆիկը կառուցելուց հետո տեսնել, թե որ կետերն են մեզ անհրաժեշտ, և ծրագիրը գրել սկզբից՝ միայն այդ կետերի համար ֆունկցիան սահմանելով և գրաֆիկը կառուցելով: Բայց դա միշտ չէ, որ հարմար է անել, ժամանակատար է և, որ ամենակարևորն է, միշտ չէ, որ կարող է ստացվել այն, ինչ ցանկանում էինք: Դրա փոխարեն շատ ավելի հարմար է օգտվել **axis** հրամանից:

Օրինակ.

$-10 \le x \le 10$ միջակայքում կառուցենք $y = e^x x^3 \sin x$ ֆունկցիայի գրաֆիկը:

```
>> x = [-10:0.01:10];
>> y = exp(x).* (x.^3).*sin(x);
>> plot(x,y)
```

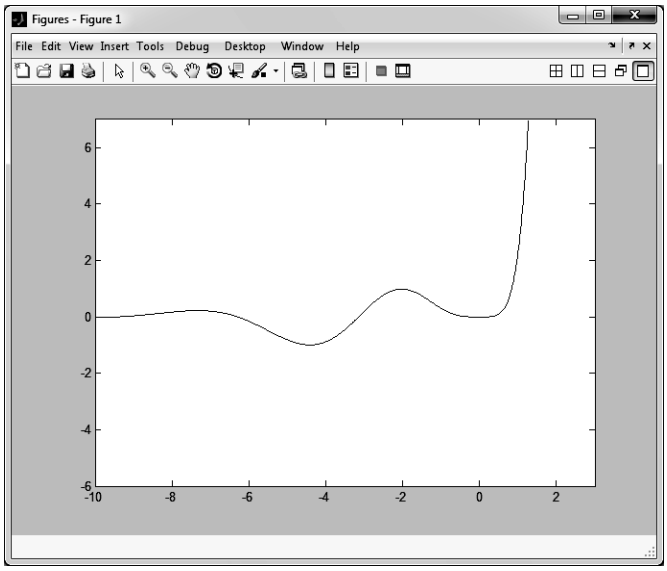


Նկ. 15

Գրաֆիկը պատկերված է Նկ. 15–ում: Ինչպես տեսնում ենք, այն $-10 \le x \le 3$ միջակայքում ուղիղ գծի տեսք ունի: Պատճառն այն է, որ այս տիրույթում $y = e^x x^3 \sin x$ ֆունկցիայի արժեքները միավորի կարգի են, իսկ մոտավորապես $6 \le x \le 9$ տիրույթի մոտ կարգով հասնում են մինչև $\sim 10^6$ (ինչպես տեսնում ենք՝ առանցքի վերին հատվածում գրված է $\times 10^6$ ընդհանուր բազմապատկիչը): Որպեսզի երևա, թե իրականում ինչ վարք ունի ֆունկցիան $-10 \le x \le 3$ միջակայքում, կարող ենք մեծացնել այդ տիրույթը և դիտել ֆունկցիայի վարքագիծը՝ օգտվելով **axis** ֆունկցիայից: Առանցքները փոխենք այնպես, որ երևան միայն $-10 \le x \le 3$ և $-6 \le y \le 7$ տիրույթները: Դրա համար գրաֆիկը կառուցելուց հետո կավելացնենք **axis([–10 3 –6 7])** հրամանը:

```
>> x = [-10:0.01:10];
>> y = exp(x).* (x.^3).*sin(x);
>> plot(x,y)
>> axis([-10 3 -6 7])
```

Ինչպես տեսնում ենք, այս տիրույթում գրաֆիկն իրականում ունի Նկ. 16–ում բերված տեսքը: Եթե ցանկանում ենք վերադառնալ Նկ. 15–ում տեսքին, անհրաժեշտ պահին ծրագրում կավելացնենք **axis auto** հրամանը:



Նկ. 16

3.6. ԳՐԱՖԻԿԱԿԱՆ ՊԱՏՈՒՀԱՆԻ ՁԵՎԱՎՈՐՈՒՄԸ

MatLab–ն ունի ներդրված մի շարք ֆունկցիաներ, որոնք թույլ են տալիս տեքստեր ավելացնել գրաֆիկական պատուհանի տարբեր հատվածներում՝ պատկերվածը դարձնելով ավելի հասկանալի և ակնառու:

xlabel('text') և **ylabel('text')** ֆունկցիաները նախատեսված են գրաֆիկական պատուհանի համապատասխանաբար x և y առանցքներին անուններ տալու համար: **xlabel** ֆունկցիայի արգումենտում գրված տեքստը տեղադրվում է x առանցքի ներքևում, իսկ **ylabel** ֆունկցիայի արգումենտում գրված տեքստը՝ y առանցքի ձախ կողմում:

title('text') ֆունկցիայի արգումենտը ևս տող է և նախատեսված է գրաֆիկին վերնագիր տալու համար: **title** ֆունկցիայի արգումենտում գրված տեքստը տեղադրվում է գրաֆիկի վերևում:

text(X,Y,'text') ֆունկցիան նախատեսված է գրաֆիկական պատուհանում կամայական տեքստ տեղադրելու համար: **text** ֆունցկիայի երրորդ արգումենտում գրված տեքստը տեղադրվում է X, Y կոորդինատներով կետում:

gtext('text') ֆունկցիայի իմաստը նույնն է, ինչ **text** ֆունկցիայինը: Տարբերությունն այն է, որ **gtext** ֆունկցիայի արգումենտում գրված տեքստը մկնիկի օգնությամբ տեղադրվում է գրաֆիկական պատուհանի կամայական կետում: Երբ ծրագիրը հանդիպում է **gtext** ֆունկցիային, ապա այն կանգ է առնում, գրաֆիկա-

կան պատուհանում հայտնվում է տեքստային կուրսոր, և ծրագիրը չի շարունակում աշխատանքն այնքան ժամանակ, քանի դեռ մկնիկի ձախ կոճակը չենք սեղմում պատուհանի անհրաժեշտ կետում:

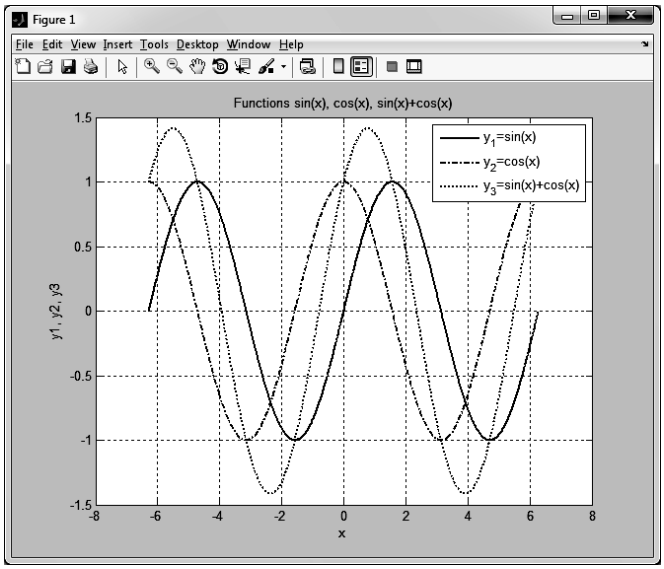
legend('text1', 'text2',... 'textN') ֆունկցիան թույլ է տալիս գրաֆիկական պատուհանում տալ կառուցված գրաֆիկների նկարագրությունները (լեգենդը): Սա շատ հարմար է այն դեպքերում, երբ կառուցվել է մի քանի գրաֆիկ՝ տարբեր գույներով ու տեսքերով, և դժվար է հասկանալ, թե դրանցից յուրաքանչյուրը որ ֆունկցիայի գրաֆիկն է: Եթե կառուցվել է N հատ գրաֆիկ նույն պատուհանում, ապա լեգենդի արգումենտում, ստորակետներով անջատված, գրվում է N հատ տեքստ: Այդ տեքստերը հայտնվում են գրաֆիկական պատուհանում նույն հերթականությամբ, ինչ հերթականությամբ կառուցվել են գրաֆիկները, և յուրաքանչյուր տեքստի ձախ կողմում դրվում է համապատասխան գրաֆիկի գծի տեսակն ու գույնը՝ հնարավորություն տալով գրաֆիկներն իրարից տարբերել:

Երբեմն շատ հարմար է նաև գրաֆիկական պատուհանի ներսում կառուցել կոորդինատների ցանց: Այդ նպատակի համար նախատեսված են **grid on** և **grid off** հրամանները, որոնք համապատասխանաբար տեղադրում և հեռացնում են ցանցը գրաֆիկական պատուհանից:

Օրինակ.

$-2\pi \le x \le 2\pi$ միջակայքում կառուցենք $y1 = \sin x$, $y2 = \cos x$ և $y3 = \sin x + \cos x$ ֆունկցիաների գրաֆիկները միևնույն պատուհանում: Պատուհանը ձևավորենք այնպես, որ երևա կոորդինատների ցանցը (**grid on**): Այնուհետև, x առանցքին տանք "x" անունը, y առանցքին՝ "y1, y2, y3" անունը, գրաֆիկի վերնագիրը դնենք "Functions sin(x), cos(x), sin(x)+cos(x)" և գրենք լեգենդն այնպես, որ $y1$ ֆունկցիայի նկարագրությունը լինի "y1=sin(x)", $y2$ ֆունկցիայի նկարագրությունը՝ "y2=cos(x)", իսկ $y3$ ֆունկցիայինը՝ "y3=sin(x)+cos(x)": Գրաֆիկը կունենա նկ. 17–ում ներկայացված տեսքը:

```
>> x = [-2*pi:0.01:2*pi];
>> y1 = sin(x);
>> y2 = cos(x);
>> y3 = sin(x) + cos(x);
>> plot(x,y1,'k-')
>> hold on
>> plot(x,y2,'r-.')
>> plot(x,y3,'b:')
>> grid on
>> xlabel('x')
>> ylabel('y1, y2, y3')
>> title('Functions sin(x), cos(x), sin(x)+cos(x)')
>> legend('y_1=sin(x)', 'y_2=cos(x)', 'y_3=sin(x)+cos(x)')
```



Նկ. 17

Ինչպես տեսանք, գրաֆիկական պատուհանում տարբեր տեքստային մեկնաբանություններ ավելացնելը բավականին հեշտ է և հարմար: Մինչ այժմ մենք խոսել ենք այնպիսի տեքստերի մասին, որոնց սիմվոլները կարելի է գտնել համակարգչի ստեղնաշարի վրա: Սակայն շատ հաճախ կարիք է զգացվում տեքստերում ավելացնել նաև հատուկ սիմվոլներ, օրինակ, հունարեն տառեր, տարբեր մաթեմատիկական սիմվոլներ, ինչպես նաև՝ ինդեքսներ կամ ցուցիչներ: **MatLab**–ը հնարավորություն է տալիս տեքստերը ձևափոխել այնպես, որ դրանք ստանան անհրաժեշտ սիմվոլների տեսքը: Դրա համար տեքստերի մեջ պետք է գրել **TeX** ստանդարտի համապատասխան հրամանները: Երբ **MatLab**–ը տեքստի մեջ հանդիպում է \ սիմվոլին, դրան հաջորդող սիմվոլը հասկանում է որպես **TeX** ստանդարտի հրաման: Օրինակ, ըստ այդ ստանդարտի, հունարեն α տառը գրվում է \alpha հրամանով: Եթե այժմ ցանկանանք գրաֆիկի վերնագիրը գրել, օրինակ, $y = \sin\alpha$ տեսքով, անհրաժեշտ է **MatLab**–ում գրել **title('y=sin\alpha')** հրամանը:

Հետևյալ աղյուսակում բերված են **TeX** ստանդարտի հաճախ օգտագործվող սիմվոլները:

TeX հրամանը	**Արդյունքը**	**TeX հրամանը**	**Արդյունքը**	**TeX հրամանը**	**Արդյունքը**
\alpha	α	\chi	χ	\leq	$\leq$
\beta	β	\psi	ψ	\geq	$\geq$
\gamma	γ	\omega	ω	\neq	$\neq$

\delta	δ	\Gamma	Γ	\infty	∞
\epsilon	ε	\Delta	Δ	\leftrightarrow	↔
\eta	η	\Theta	Θ	\leftarrow	←
\theta	θ	\Lambda	Λ	\uparrow	↑
\kappa	κ	\Phi	Φ	\rightarrow	→
\lambda	λ	\Xi	Ξ	\downarrow	↓
\mu	μ	\Pi	Π	\circ	°
\nu	ν	\Sigma	Σ	\pm	±
\xi	ξ	\Psi	Ψ	\propto	∝
\rho	ρ	\Omega	Ω	\bullet	•
\pi	π	\forall	∀	\oslash	∅
\sigma	σ	\exists	∃	\cap	∪
\tau	τ	\approx	≈	\cup	∩
\phi	φ	\nabla	∇	\equiv	≡

Բացի այս սիմվոլներից, **TeX** հրամաններով հնարավոր է փոխել տեքստի ոճը՝ դարձնել թեք (italic) կամ թավ (bold), փոխել տեքստի տառատեսակը (font) և այլն: Այս գործողությունների համար անհրաժեշտ է օգտվել աղյուսակում բերված ցուցակից:

TeX հրամանը	**Նկարագրությունը**
\bf	այս հրամանին հաջորդող տեքստը դառնում է թավ
\it	այս հրամանին հաջորդող տեքստը դառնում է թեք
\rm	այս հրամանին հաջորդող տեքստը գրում է նորմալ տառատեսակով
^	այս հրամանին հաջորդող սիմվոլը գրում է ցուցիչում
_	այս հրամանին հաջորդող սիմվոլը գրում է ինդեքսում
\fontname{*fontname*}	այս հրամանին հաջորդող տեքստը գրվում է ձևավոր փակագծում գրված տառատեսակով
\fontsize{*fontsize*}	այս հրամանին հաջորդող տեքստը գրվում է ձևավոր փակագծում գրված չափով

Ինչպես տեսնում ենք, բացի ցուցիչի և ինդեքսի հրամաններից, մնացած բոլոր հրամանները ազդում են իրենց հաջորդող ամբողջ տեքստի վրա: Եթե այդ հատկությունները պետք է ազդեն տեքստի միայն մի մասի վրա, ապա համապատասխան տեղում պետք է գրել ձևավորման նոր հրամանը: Ի տարբերություն սրանց՝ ցուցիչը և ինդեքսն ազդում են միայն հրամանին հաջորդող մեկ սիմվոլի վրա: Եթե ցանկանում ենք մի քանի իրար հաջորդող սիմվոլ գրել որպես ցուցիչ կամ ինդեքս, ապա այդ սիմվոլները պետք է ներդնել ձևավոր փակագծերում: Վերջին երկու հրամանները վերաբերում են համապատասխանաբար տառատեսակի

անվանն ու չափին: Տառատեսակի անունը (fontname) պետք է լինի գոյություն ունեցող տառատեսակի անուն, օրինակ՝ arial: Տառատեսակի չափը (fontsize) դրվում է տառատեսակի միավորներով, օրինակ՝ 12:

Օրինակներ.

TeX հրամանը	**Արդյունքը**
\fontname{arial}\fontsize{14}\phi=0 to 4\pi	$\varphi = 0$ to 4π
\itx^{4.8}	$x^{4.8}$
u_5	u_5
\bf\it\Plot of Function \rmsin(\itx)	***Plot of Function*** sin$x$

3.7. ՄԻ ՔԱՆԻ ԳՐԱՖԻԿԱԿԱՆ ԵՆԹԱՊԱՏՈՒՀԱՆՆԵՐ ՄԵԿ ԸՆԴՀԱՆՈՒՐ ՊԱՏՈՒՀԱՆՈՒՄ

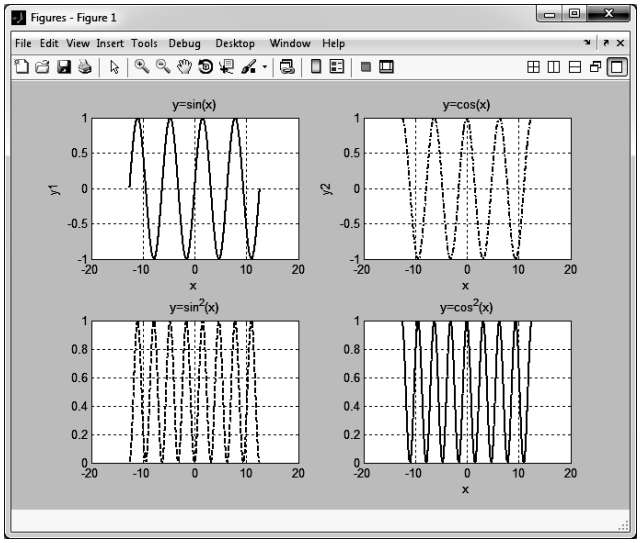
MatLab–ը հնարավորություն է տալիս գրաֆիկական պատուհանը տրոհել առանձին ենթապատուհանների, որոնք հիմնական պատուհանում դասավորվում են աղյուսակի (մատրիցի) տեսքով, և առանձին–առանձին աշխատել դրանցից յուրաքանչյուրի հետ: Սա շատ հարմար է իրար հետ կապ ունեցող գրաֆիկներն առանձին պատուհանում կառուցելու և իրար հետ համեմատելու համար: Այս նպատակի համար նախատեսված է **subplot** ֆունկցիան, որը ստանում է երեք արգումենտ՝ **subplot(m,n,k)**: Այստեղ *m*–ը և *n*–ը համապատասխանաբար ուղղահայաց և հորիզոնական ուղղություններով դասավորված ենթապատուհանների քանակն է, *k*–ն՝ այն ենթապատուհանի համարը, որը պետք է դառնա ընթացիկը: Օրինակ, **subplot(2,3,5)** հրամանը հիմնական պատուհանում ստեղծում է 6 հատ ենթապատուհան, որոնք դասավորված են երկու տողով՝ յուրաքանչյուր տողում երեք հատ, և 5–րդ ենթապատուհանն ընտրվում է որպես ընթացիկ: Ընթացիկ պատուհանի համարակալումը սկսվում է վերևի ձախ անկյունից. նախ համարակալվում են առաջին տողի ենթապատուհանները՝ ձախից աջ հերթականությամբ, այնուհետև երկրորդ տողի: **subplot** ֆունկցիան գրելուց հետո արդեն կարելի է օգտվել գրաֆիկներին վերաբերող, նախորդ բաժիններում ներկայացված կամայական ֆունկցիայից: Այդ բոլոր գործողությունները կկատարվեն **subplot** ֆունկցիայի երրորդ արգումենտում գրված ենթապատուհանի մեջ:

Օրինակ.

Հիմնական գրաֆիկական պատուհանը տրոհենք 4 ենթապատուհանների՝ դասավորված երկու տողով և երկու սյունով, և այդ ենթապատուհաններում $-4\pi \le x \le 4\pi$ միջակայքում կառուցենք համապատասխանաբար $y1 = \sin x$, $y2 = \cos x$, $y3 = \sin^2 x$ և $y4 = \cos^2 x$ ֆունկցիաների գրաֆիկները:

Ծրագրի իրականացման արդյունքում ստացված գրաֆիկները բերված են նկ. 18–ում:

```
>> x = [-4*pi:0.01:4*pi];
>> y1 = sin(x);
>> y2 = cos(x);
>> y3 = sin(x).^2;
>> y4 = cos(x).^2;
>> subplot(2,2,1)
>> plot(x,y1,'r')
>> grid on
>> title('y=sin(x)')
>> xlabel('x')
>> ylabel('y1')
>> subplot(2,2,2)
>> plot(x,y2,'b-.')
>> grid on
>> title('y=cos(x)')
>> xlabel('x')
>> ylabel('y2')
>> subplot(2,2,3)
>> plot(x,y3,'r--')
>> grid on
>> title('y=sin^2(x)')
>> xlabel('x')
>> ylabel('y3')
>> subplot(2,2,4)
>> plot(x,y4,'g-')
>> grid on
>> title(''y=cos^2(x)')
>> xlabel('x')
>> ylabel('y4')
```



Նկ. 18

3.8. ՊԱՐԱՄԵՏՐԱԿԱՆ ՏԵՍՔԻ ԳՐԱՖԻԿՆԵՐԻ ԿԱՌՈՒՑՈՒՄԸ

Ենթադրենք՝ ցանկանում ենք կառուցել $y(x)$ կախվածության գրաֆիկն այն դեպքում, երբ x և y փոփոխականներն իրենց հերթին կախված են երրորդ, t փոփոխականից: Այսպիսի կախվածությունը կոչվում է պարամետրական, իսկ t փոփոխականը՝ պարամետր: Պարամետրական ֆունկցիայի գրաֆիկը կառուցելու համար անհրաժեշտ է կատարել հետևյալ քայլերը.

1. որպես վեկտոր սահմանել t փոփոխականը,

2. t պարամետրի միջոցով սահմանել x և y վեկտորները, ընդ որում գործողությունները պետք է կատարվեն անդամ առ անդամ,
3. **plot(x,y)** ֆունկցիայի օգնությամբ կառուցել $y(x)$ կախվածությունը:

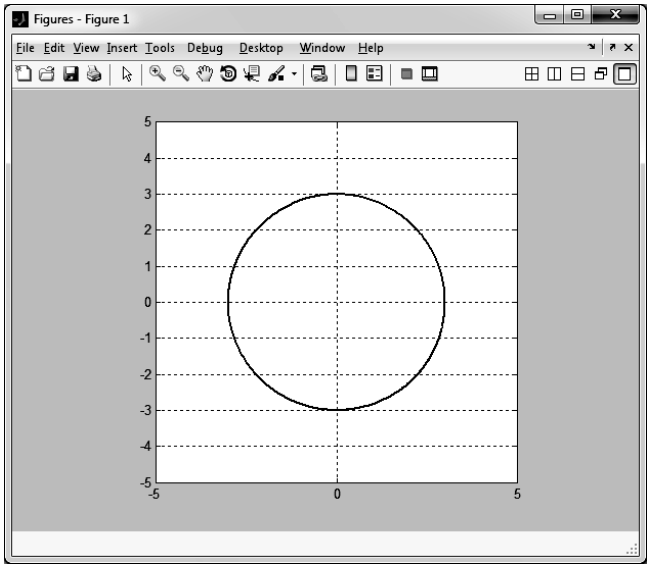
Օրինակ.

R շառավղով շրջանագծի հավասարումը պարամետրական տեսքով որոշվում է $x = R\cos t$, $y = R\sin t$ առնչություններով: Սրանց օգնությամբ կառուցենք $R = 3$ շառավղով շրջանագիծ: Ծրագիրը կգրվի հետևյալ տեսքով.

```
>> R = 3;
>> t = [0:0.01:2*pi];
>> x = R*cos(t);
>> y = R*sin(t);
>> plot(x,y)
>> axis([–5 5 –5 5])
>> axis equal
>> grid on
```

Ինչպես արդեն նշել ենք, **MatLab**-ը լռելյայն ավտոմատ մասշտաբավորում է առանցքներն այնպես, որ այն ամբողջությամբ ընդգրկի կառուցվող տվյալները: Դրա համար գրել ենք **axis([–5 5 –5 5])** հրամանը, որպեսզի այս տիրույթում տեսնենք շրջանագիծը, այլապես այն կտեսնեինք [–3 3 –3 3] տիրույթում: **axis equal** հրամանն ավելացրել ենք, որպեսզի **MatLab**–ը x և y առանցքների մասշտաբավորումը դարձնի համարժեք: Հակառակ դեպքում, շրջանագծի փոխարեն պատկերը կերևար սեղմված տեսքով:

Ծրագրի արդյունքը բերված է նկ. 19–ում:



Նկ. 19

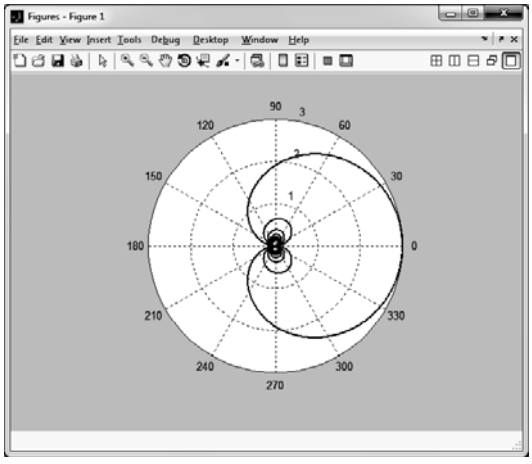
3.9. ԳՐԱՖԻԿՆԵՐԻ ԿԱՌՈՒՑՈՒՄԸ ԲԵՎԵՌԱՅԻՆ ԿՈՈՐԴԻՆԱՏԱԿԱՆ ՀԱՄԱԿԱՐԳՈՒՄ

MatLab–ը հնարավորություն է տալիս գրաֆիկներ կառուցել նաև (r,φ) բևեռային կոորդինատական համակարգում, որտեղ յուրաքանչյուր կետ նկարագրվում է r շառավիղ–վեկտորով և φ բևեռային անկյունով: Ինչպես գիտենք, դեկարտյան և բևեռային կոորդինատները կապված են $x = r\cos\varphi$ և $y = r\sin\varphi$ առնչություններով: Բևեռային կոորդինատական համակարգում $r(\varphi)$ ֆունկցիայի գրաֆիկը կառուցելու համար **MatLab**–ում **plot**–ի փոխարեն անհրաժեշտ է օգտվել **polar** ֆունկցիայից: Ծրագիրը գրելու սկզբունքն այս դեպքում նույնն է՝ նախ անհրաժեշտ է սահմանել *phi* վեկտորը, այնուհետև, վերջինիս օգնությամբ, r վեկտորը և վերջում՝ գրել **polar(phi, r)** հրամանը: Ինչպես **plot** ֆունկցիայի դեպքում, այստեղ ևս կարելի է գրել ոչ պարտադիր, երրորդ արգումենտը, որում նշվում են գրաֆիկի գույնը, մարկերի տեսակն ու գրաֆիկի ձևը: Այս հատկությունները նույնն են, ինչ **plot** ֆունկցիայի դեպքում: Գրաֆիկի ձևավորման մյուս հատկությունները, օրինակ՝ գրաֆիկի կամ առանցքների վերնագրելը, կոորդինատային ցանցի կառուցումը, **subplot** ֆունկցիայի կիրառումը և այլն, ևս չեն փոխվում:

Օրինակ.

Բևեռային կոորդինատական համակարգում կառուցենք $r = 3\dfrac{\sin\varphi}{\varphi}$ ֆունկցիայի գրաֆիկը, որտեղ $-8\pi \le \varphi \le 8\pi$: Ծրագիրը կունենա հետևյալ տեսքը, իսկ ստացված գրաֆիկը պատկերված է նկ. 20–ում:

```
>> phi = [-8*pi:0.01:8*pi];
>> r = 3*sin(phi)./phi;
>> polar(phi, r, 'k')
>> grid on
```



Նկ. 20

3.10. ԿԵՏԻ ՇԱՐԺՈՒՄԸ ՀԱՐԹՈՒԹՅԱՆ ՄԵՋ

Երբեմն հարմար է գրաֆիկական պատուհանում ոչ թե պարզապես պատկերել ֆունկցիայի գրաֆիկը, այլ հետևել դրա փոփոխության ընթացքին: **MatLab**–ում նախատեսված է **comet** ներդրված ֆունկցիան, որը հնարավորություն է տալիս շարժման մեջ քայլ առ քայլ հետևել ֆունկցիայի գրաֆիկի կառուցմանը կոորդինատական հարթության մեջ: **comet** ֆունկցիայի գրելաձևը նման է **plot** ֆունկցիայի գրելաձևին՝ այն որպես առաջին արգումենտ ստանում է անկախ փոփոխականի վեկտորը, իսկ որպես երկրորդ արգումենտ՝ ֆունկցիայի վեկտորը:

Օրինակ.

Գրենք ծրագիր, որը $0 \le t \le 4\pi$ միջակայքում թույլ կտա հետևել $y = \sin t$ ֆունկցիայի փոփոխության ընթացքին: Ծրագիրը կունենա հետևյալ տեսքը.

```
>> t = [0:0.01:4*pi];
>> y = sin(t);
>> comet(t,y)
```

Ծրագրի իրականացման ժամանակ գրաֆիկական պատուհանում հայտնվում է օղակաձև մարկեր, որը նախ գտնվում է $t = 0$ պահին $y = \sin 0$ կոորդինատով կետում, այնուհետև այն հերթով հայտնվում է $t = 0.01,\ 0.02,\ldots,\ 4\pi$ պահերին $y = \sin t$ կոորդինատներով կետերում՝ յուրաքանչյուր կետը նախորդին միացնելով ուղիղ գծով: Այսպիսով՝ այն կարելի է դիտարկել որպես կետ, որը շարժվում է հարթության մեջ՝ իր հետևից հետք թողնելով: Հասնելով վերջին կետին՝ գրաֆիկական պատուհանում երևում է $y = \sin t$ ֆունկցիայի գրաֆիկը:

Հասկանալի է, որ կետի շարժման արագությունը կախված է վեկտորի քայլերի քանակից: Որքան քայլը լինի ավելի փոքր, այնքան շատ կետերով այն կանցնի և, հետևաբար, շարժումն ավելի դանդաղ կլինի:

3.11. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Կառուցեք հետևյալ ֆունկցիաների գրաֆիկները նշված միջակայքում, Δx քայլով.

- $f(x) = 0.6x^5 - 5x^3 + 9x + 2$, եթե $-4 \le x \le 4$, $\Delta x = 0.1$,
- $f(x) = \sqrt{\operatorname{ch}\dfrac{x}{2\pi}}$, եթե $0.5 \le x \le 1.5$, $\Delta x = 0.1$,
- $f(x) = x^{2x+1} + x^3 - 2x$, եթե $1 \le x \le 5$, $\Delta x = 0.4$,
- $f(x) = \arccos e^{-\sqrt[3]{3x}}$, եթե $0.2 \le x \le 0.5$, $\Delta x = 0.03$,

- $f(x)=693.8-68.8\,\mathrm{ch}\dfrac{x}{99.7}$, եթե $-300\le x\le 300$, $\Delta x=0.1$,
- $f(x)=\dfrac{a^x-b^x}{\lg\dfrac{a}{b}}\sqrt[3]{ab}$, եթե $3.2\le x\le 6.2$, $a=0.4$, $b=0.8$, $\Delta x=0.6$,
- $f(x)=\sqrt[4]{|x^2-2.5|}+\sqrt[3]{\lg x^2}$, եթե $1.25\le x\le 2.75$, $\Delta x=0.3$,
- $f(x)=\dfrac{\arccos(x^2-b^2)}{\arcsin(x^2-a^2)}$, եթե $0.2\le x\le 0.95$, $a=0.05$, $b=0.06$, $\Delta x=0.15$,
- $f(x)=\dfrac{\sqrt[3]{a}+\mathrm{tg}^{4.5}(bx)}{\sqrt[5]{b}+\mathrm{ctg}^{2.7}(ax)}$, եթե $0.33\le x\le 1.23$, $a=0.1$, $b=0.5$, $\Delta x=0.18$:

2. Միևնույն գրաֆիկական պատուհանում կառուցեք $f(x)=3x\sin x-2x$ ֆունկցիայի, ինչպես նաև այդ ֆունկցիայի առաջին և երկրորդ կարգի ածանցյալների գրաֆիկները $-2\pi\le x\le 2\pi$ միջակայքում:

3. Կառուցեք $f(x)=\dfrac{1.5x}{x-4}$ ֆունկցիայի գրաֆիկը: Քանի որ այս ֆունկցիան որոշված չէ $x=4$ կետում, ապա գրաֆիկը կառուցեք երկու վեկտոր սահմանելով ($-10\le x1\le 3.7$ և $-4.3\le x2\le 10$): Սահմանված վեկտորներից յուրաքանչյուրի համար միևնույն գրաֆիկական պատուհանում կառուցեք $f(x)$ ֆունկցիայի գրաֆիկները:

4. Ռիխտերի սանդղակով երկրաշարժի մագնիտուդը որոշվում է $M=\dfrac{2}{3}\lg\dfrac{E}{10^{4.4}}$ օրենքով, որտեղ E-ն երկրաշարժի ժամանակ անջատված ջրուլներով արտահայտված էներգիան է: Կառուցեք $E(M)$ կախվածության գրաֆիկը $3\le M\le 8$ միջակայքում:

5. Կառուցեք $y1=\sqrt[3]{x^2+3}\cos\dfrac{\pi x}{2}$ և $y2=\sqrt[3]{x^3+4}\sin\dfrac{\pi x}{2}$ ֆունկցիաների գրաֆիկները $1\le x\le 2.5$ միջակայքում 0.15 քայլով: $y1$ ֆունկցիայի գրաֆիկը կառուցեք կարմիր, քառակուսի մարկերներով և հոծ գծով: $y2$ ֆունկցիայի գրաֆիկը կառուցեք սև, կլոր մարկերներով և կետագծերով: Գրաֆիկի վերնագրեք “Functions y1 and y2”, լեգենդները՝ “Function y1” և “Function y2”:

6. Բևեռային կոորդինատական համակարգում կառուցեք հետևյալ ֆունկցիաների գրաֆիկները (բոլոր կորերում φ պարամետրը փոխվում է 0.01 քայլով).

- $r=\varphi$, $0\le\varphi\le 8\pi$,
- $r=8(\sin(\cos(\mathrm{tg}\,\varphi)))$, $-8\pi\le\vartheta\le 8\pi$,
- $r=2\cos 4\varphi$, $0\le\varphi\le 2\pi$,
- $r=e^{\cos\varphi}-2\cos 4\varphi+\sin^5\dfrac{\varphi}{12}$, $0\le\vartheta\le 75.4$:

7. Կառուցեք հետևյալ նշանավոր կորերը, որոնց հավասարումները տրված են պարամետրական տեսքով (բոլոր կորերում t պարամետրը փոխվում է 0.01 քայլով).

- **պարույրագիծ** $x = t\sin t$, $y = t\cos t$, $0 \le t \le 5\pi$,
- **դելտոիդ** $x = 2\cos t + \cos 2t$, $y = 2\sin t - \sin 2t$, $0 \le t \le 2\pi$,
- **աստրոիդ** $x = 2\sin^3 t$, $y = 2\cos^3 t$, $0 \le t \le 2\pi$,
- **հիպոցիկլոիդ**

  ա) $x = 20\left(\cos t + \dfrac{\cos 5t}{5}\right)$, $y = 20\left(\sin t - \dfrac{\sin 5t}{5}\right)$, $0 \le t \le 2\pi$,

  բ) $x = 24\left(\cos t + \dfrac{\cos(6t)}{6}\right)$, $y = 24\left(\sin t - \dfrac{\sin(6t)}{6}\right)$, $0 \le t \le 10\pi$,
- **կարդիոիդ** $x = (1+\cos t)\cos t$, $y = (1+\cos t)\sin t$, $0 \le t \le 2\pi$,
- **նեֆրոիդ** $x = 6\cos t - 4\cos^3 t$, $y = 4\sin^3 t$, $0 \le t \le 2\pi$,
- **էպիցիկլոիդ**

  ա) $x = 8\left(\cos t - \dfrac{\cos 4t}{4}\right)$, $y = 8\left(\sin t - \dfrac{\sin 4t}{4}\right)$, $0 \le t \le 2\pi$,

  բ) $x = 13\left(\cos t - \dfrac{\cos 6.5t}{6.5}\right)$, $y = 13\left(\sin t - \dfrac{\sin 6.5t}{6.5}\right)$, $0 \le t \le 4\pi$,

  գ) $x = 6.2\left(\cos t - \dfrac{\cos 3.1t}{3.1}\right)$, $y = 6.2\left(\sin t - \dfrac{\sin 3.1t}{3.1}\right)$,
- **թիթեռ**

  $x = \sin t\left(e^{\cos t} - 2\cos 4t + \sin^5 \dfrac{t}{12}\right)$,

  $y = \cos t\left(e^{\cos t} - 2\cos 4t + \sin^5 \dfrac{t}{12}\right)$, $0 \le t \le 12\pi$:

8. Հարթ կորի էվոլյուտ է կոչվում այդ կորին տարած նորմալների ընտանիքի պարուրիչը: Եթե կորը էլիպս է, որը նկարագրվում է $x = a\cos\varphi$, $y = b\sin\varphi$ պարամետրական օրենքով, ապա կարելի է ցույց տալ, որ իր էվոլյուտը նկարագրվում է $X = \dfrac{a^2 - b^2}{a}\cos^3\varphi$, $Y = \dfrac{b^2 - a^2}{b}\sin^3\varphi$ պարամետրական օրենքով: $-2\pi \le \varphi \le 2\pi$ միջակայքում կառուցեք $a = 2$ մեծ և $b = 1$ փոքր կիսաառանցքներով էլիպսի և իր էվոլյուտի գրաֆիկները միևնույն գրաֆիկական պատուհանում:

9. Այն հետագծերը, որոնցով շարժվում է մասնիկը՝ միաժամանակ մասնակցելով երկու փոխուղղահայաց տատանումների, կոչվում են Լիսաժուի կորեր: Ենթադրենք՝ ունենք երկու փոխուղղահայաց ներդաշնակ տատանումներ, որոնցից մեկը տեղի

է ունենում x առանցքի, մյուսը՝ y առանցքի ուղղությամբ: Այս երկու տատանումներն ընդհանուր դեպքում կնկարագրվեն $x = a_1 \sin(\omega_1 t + \varphi_1)$ և $y = a_2 \sin(\omega_2 t + \varphi_2)$ օրենքներով, որտեղ a_1 –ը և a_2 –ը համապատասխանաբար առաջին և երկրորդ տատանումների լայնույթներն են, ω_1 –ը և ω_2 –ը՝ հաճախությունները, իսկ φ_1 –ը և φ_2 –ը՝ սկզբնական փուլերը: Եթե ω_1 և ω_2 հաճախությունները հարաբերվում են որպես ամբողջ թվեր, ապա շարժման հետագիծն ունենում է փակ տեսք: Ենթադրենք՝ $a_1 = 1.4$, $a_2 = 1.2$, $\varphi_1 = \varphi_2 = 0$: Օգտագործելով **subplot** ֆունկցիան՝ կառուցեք Լիսաժուի կորերը հաճախությունների $\frac{\omega_1}{\omega_2} = \frac{1}{2}, \frac{1}{3}, \frac{2}{3}$ և $\frac{3}{4}$ հարաբերությունների համար:

4. ՄԱՏՐԻՑՆԵՐ

Ինչպես արդեն նշել ենք, **MatLab**–ը յուրաքանչյուր տվյալ ընդունում և դրանց հետ աշխատում է երկչափ զանգվածի նման, այսինքն՝ **MatLab**–ի յուրաքանչյուր տվյալ մատրից է: Սովորական թվերը (սկալյարները) և վեկտորները, որոնց հետ աշխատեցինք նախորդ գլուխներում, մատրիցների մասնավոր դեպքերն են: Սկալյարը 1×1 չափի (1 տողից և 1 սյունից բաղկացած) մատրից է, տող–վեկտորը՝ $1\times n$ չափի (1 տողից և n սյունից բաղկացած), իսկ սյուն–վեկտորը՝ $n\times1$ չափի (n տողից և 1 սյունից բաղկացած) մատրից: Ընդհանուր դեպքում մատրիցները կարող են բաղկացած լինել կամայական քանակությամբ տողերից և սյուներից: **MatLab**–ն օժտված է մատրիցների հետ աշխատելու լայն հնարավորություններով: Այս գլխում կծանոթանանք կամայական m հատ տող և n հատ սյուն ունեցող ($m\times n$ չափի) մատրիցների հետ աշխատանքին:

4.1. ՄԱՏՐԻՑԻ ՍԱՀՄԱՆՈՒՄԸ

Եթե մատրիցը բաղկացած է m հատ տողից և n հատ սյունից ($m\times n$ չափի), ապա **MatLab**–ում այդ մատրիցը հայտարարելու համար անհրաժեշտ է այն ներմուծել տող առ տող: Ընդ որում դա անելու համար գոյություն ունի երկու եղանակ:

Առաջին եղանակով բացող քառակուսի փակագծից հետո ներմուծվում է առաջին տողը որպես սովորական տող–վեկտոր (այսինքն՝ տարրերն իրարից անջատվում են բացատով կամ ստորակետներով), այնուհետև, առանց փակագիծը փակելու, դրվում է կետ–ստորակետ, ներմուծվում երկրորդ տողը և այդպես շարունակ: Վերջին տողը ներմուծելուց հետո դրվում է փակող քառակուսի փակագիծը: Կարելի է ասել, որ մատրիցը սահմանվում է որպես սյուն–վեկտոր, որի յուրաքանչյուր տարրը տող–վեկտոր է:

Օրինակ.

$A=\begin{pmatrix}5 & 35 & 43\\ 4 & 76 & 81\\ 21 & 32 & 40\end{pmatrix}$ մատրիցն առաջին եղանակով կհայտարարվի այսպես.

```
>> A = [5 35 43; 4 76 81; 21 32 40];
```

Երկրորդ եղանակով մատրիցը հայտարելիս վերագրման նշանից հետո դնում են բացող քառակուսի փակագիծ, ներմուծում մատրիցի առաջին տողը տող–վեկտորի տեսքով, այնուհետև, առանց կետ–ստորակետ դնելու, սեղմում են **Enter**, հաջորդ տողում ներմուծում երկրորդ տողը և այդպես շարունակ: Վերջին տողը ներմուծելուց հետո դնում են փակող քառակուսի փակագիծը:

Օրինակ.

Նախորդ օրինակի մատրիցը երկրորդ եղանակով կհայտարարվի այսպես.

```
>> A = [5 35 43
        4 76 32
        43 81 40];
```

Ուշադրություն դարձնենք, որ երկրորդ եղանակով մատրիցը ներմուծելու ժամանակ յուրաքանչյուր տողն ավարտելուց հետո **Enter** սեղմելիս հաջորդ տողում չի հայտնվում >> նշանը, այսինքն՝ **MatLab**–ը հրամանը չի համարում ավարտված այնքան ժամանակ, քանի դեռ չի հանդիպել փակող քառակուսի փակագծին:

Հետագայում մատրիցների անունները սահմանելիս միշտ կօգտվենք լատինատառերից, իսկ սկալյարներն ու վեկտորները սահմանելիս՝ փոքրատառերից:

Ակնհայտ է, որ մատրիցի տարրերը կարող են լինել ոչ միայն թվեր, այլև փոփոխականների անուններ, ֆունկցիաներ կամ ամբողջական արտահայտություններ:

Օրինակ.

```
>> k = 6; e = 3; h = 4;
>> M = [e, k*h, cos(pi/3); h^2, sqrt(h*h/k), 14]
   M =
       3.0000   24.0000    0.5000
      16.0000    1.6330   14.0000
```

Քանի որ մատրիցի տողերը ներմուծվում են որպես տող–վեկտորներ, ապա դրանք կարելի է սահմանել՝ օգտվելով նաև տող–վեկտորների սահմանման մյուս եղանակներից (տես գլուխ 2):

Օրինակ.

```
>> B = [1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 7 41 12];
   B =
        1     3     5     7     9    11
        0     5    10    15    20    25
       10    20    30    40    50    60
       67     2    43     7    41    12
```

Երբեմն մատրիցներ սահմանելիս այնքան էլ հարմար չէ հերթով ներմուծել մատրիցի տարրերը, մանավանդ այն դեպքերում, երբ մատրիցի չափերը մեծ են: **MatLab**–ը հնարավորություն է տալիս ստեղծել նաև մի քանի հատուկ մատրիցներ, որոնց սահմանումից և հետագա ձևափոխությունից հետո ավելի է հեշտանում անհրաժեշտ մատրիցի ստանալը: Այդ հատուկ մատրիցները սահմանվում են հետևյալ ներդրված ֆունկցիաների օգնությամբ.

- **zeros(m, n)** ֆունկցիան թույլ է տալիս ստեղծել $m \times n$ չափի զրոյական մատրից, այսինքն՝ մատրից, որի բոլոր տարրերը հավասար են զրոյի,
- **ones(m, n)** ֆունկցիան թույլ է տալիս ստեղծել $m \times n$ չափի 1–երից կազմված մատրից,
- **eye(n)** ֆունկցիան թույլ է տալիս ստեղծել $n \times n$ չափի միավոր մատրից, այսինքն՝ քառակուսի մատրից, որի գլխավոր անկյունագծի տարրերը մեկեր են, իսկ մնացած տարրերը՝ զրոներ,
- **rand(m, n)** ֆունկցիան թույլ է տալիս ստեղծել $m \times n$ չափի մատրից, որի տարրերը 0–ից 1 միջակայքում ընկած պատահական թվեր են:

Օրինակ.

```
>> zeros(3,4)
   ans =
        0     0     0     0
        0     0     0     0
        0     0     0     0
>> ones(5,6)
   ans =
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
>> eye(7)
   ans =
        1     0     0     0     0     0     0
        0     1     0     0     0     0     0
        0     0     1     0     0     0     0
        0     0     0     1     0     0     0
        0     0     0     0     1     0     0
        0     0     0     0     0     1     0
        0     0     0     0     0     0     1
>> rand(3,6)
   ans =
       0.2575    0.8143    0.3500    0.6160    0.8308    0.9172
       0.8407    0.2435    0.1966    0.4733    0.5853    0.2858
       0.2543    0.9293    0.2511    0.3517    0.5497    0.7572
```

B մատրիցը կոչվում է A մատրիցի տրանսպոնացված մատրից, եթե այն ստացվում է A–ի տողերի և սյուների տեղերը փոխելուց: **MatLab**–ում մատրիցը կարելի է տրանսպոնացնել՝ օգտվելով (′) օպերատորից:

Օրինակ.

```
>> A = [2 55 44 8; 21 5 -3 8; 41 64 3 2]
   A =
        2    55    44     8
       21     5    -3     8
       41    64     3     2
```

```
>> B = A'
   B =
        2    21    41
       55     5    64
       44    -3     3
        8     8     2
```

Եթե մատրիցի տարրերի մեջ պարունակվում են կոմպլեքս թվեր, ապա տրանսպոնացման (′) օպերատորը կիրառելիս կստանանք սխալ արդյունք: Իրոք, (տես §2.1), այս դեպքում ոչ միայն մատրիցը կտրասպոնացվի, այլև իր կոմպլեքս տարրերը կփոխարինվեն իրենց կոմպլեքս համալուծով: Ինչպես վեկտորների դեպքում, այստեղ ևս նմանատիպ իրավիճակներում պետք է օգտվել կամ **transpose** ներդրված ֆունկցիայից, որը որպես արգումենտ ստանում է անհրաժեշտ մատրիցը, կամ տրանսպոնացման (.′) օպերատորից:

4.2. ՄԱՏՐԻՑՆԵՐՈՎ ԱՇԽԱՏԵԼՈՒ ՀԱՄԱՐ ՆԱԽԱՏԵՍՎԱԾ ՆԵՐԴՐՎԱԾ ՖՈՒՆԿՑԻԱՆԵՐ

Ինչպես վեկտորների դեպքում, մատրիցների հետ աշխատելու համար ևս նախատեսված են մի շարք ներդրված ֆունկցիաներ: Դրանք հիմնականում նույն ֆունկցիաներն են, ինչ վեկտորների համար նախատեսված ֆունկցիաները: Դա բնական է, քանի որ վեկտորները մատրիցների մասնավոր դեպքն են: Հաճախ օգտագործվող ֆունկցիաները և դրանց նկարագրությունները ներկայացված են աղյուսակում:

Մատրիցների հետ աշխատելու համար նախատեսված ֆունկցիաներ	
Անվանումը	**Նկարագրությունը**
size(A)	վերադարձնում է *A* մատրիցի չափը՝ որպես 2 տարրից բաղկացած տող–վեկտոր: Այդ վեկտորի տարրերը հավասար են համապատասխանաբար *A* մատրիցի տողերի և սյուների քանակին,
sum(A)	վերադարձնում է *A* մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան սյան տարրերի գումարին,
sum(A,2)	վերադարձնում է *A* մատրիցի տողերի քանակին հավասար երկարությամբ սյուն–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան տողի տարրերի գումարին,
prod(A)	վերադարձնում է *A* մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան սյան տարրերի արտադրյալին,

prod(A,2)	վերադարձնում է *A* մատրիցի տողերի քանակին հավասար երկարությամբ սյուն–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան տողի տարրերի արտադրյալին,
mean(A)	վերադարձնում է *A* մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան սյան տարրերի միջին թվաբանականին,
mean(A,2)	վերադարձնում է *A* մատրիցի տողերի քանակին հավասար երկարությամբ սյուն–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան տողի տարրերի միջին թվաբանականին,
sort(A)	*A* մատրիցի սյուներից յուրաքանչյուրի տարրերը դասավորում է աճման կարգով,
sort(A,2)	*A* մատրիցի տողերից յուրաքանչյուրի տարրերը դասավորում է աճման կարգով,
max(A)	վերադարձնում է *A* մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան սյան տարրերի առավելագույն արժեքին,
min(A)	վերադարձնում է *A* մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր, որի յուրաքանչյուր տարրը հավասար է *A* մատրիցի համապատասխան սյան տարրերի նվազագույն արժեքին,
diag(u)	եթե *u*–ն վեկտոր է, ապա ստեղծվում է *u* –ի երկարության քանակով տողերից և սյուներից կազմված քառակուսի մատրից, որի անկյունագծային տարրերը հավասար են *u* վեկտորի տարրերին, իսկ մնացած տարրերը՝ զրոյի,
diag(A)	եթե *A*–ն մատրից է, ապա ստեղծվում է այդ մատրիցի անկյունագծային տարրերից բաղկացած վեկտոր,
det(A)	վերադարձնում է *A* քառակուսի մատրիցի որոշիչը,
transpose(A)	վերադարձնում է *A* մատրիցի տրանսպոնացված մատրիցը:

Ինչպես տեսնում ենք՝ **sum**, **prod**, **mean** և **sort** ֆունկցիաները կարող են ստանալ լրացուցիչ, 2 արժեքն ունեցող երկրորդ արգումենտը, որը թույլ է տալիս գումարի, արտադրյալի, միջինացման և աճման կարգով դասավորելու գործողությունները կատարել ոչ թե սյուների, այլ տողերի հետ: Եթե այս ֆունկցիաների երկրորդ արգումենտում գրեինք ոչ թե 2, այլ 1, ապա դա համարժեք կլինի երկրորդ արգումենտը չգրելուն, այսինքն՝ գործողությունները կկատարվեն սյուների հետ: Օրինակ՝ **sum(A)** և **sum(A,1)** գրելաձևերը համարժեք են: Եթե երկրորդ արգումենտում գրենք որևէ այլ թիվ, օրինակ, **prod(A,4)**, ապա որևէ գործողություն չի կատարվի, և արդյունքում կստացվի նույն *A* մատրիցը:

Ինչպես վեկտորների դեպքում, **max** և **min** ֆունկցիաները կարող են ունենալ երկու ելքային արգումենտ: Եթե այս ֆունկցիաները վերագրվում են ոչ թե մեկ, այլ երկու փոփոխականների, ապա դրանցից առաջինին վերագրվում է մատրիցի սյու-

ների քանակին հավասար տող–վեկտոր, որի յուրաքանչյուր տարրի արժեքը հավասար է մատրիցի համապատասխան սյան տարրերի առավելագույն կամ նվազագույն արժեքներին, իսկ երկորդին՝ նույն երկարությամբ տող վեկտոր, որի յուրաքանչյուր տարրը հավասար է տվյալ սյան առավելագույն կամ նվազագույն արժեքների ինդեքսներին: Ընդ որում, եթե մեծագույն կամ փոքրագույն արժեքները տվյալ սյան վրա հանդիպում են մի քանի անգամ, վերադարձվում է առաջին հանդիպածի ինդեքսը:

Օրինակ.

```
>> A = [1 4 –5 6; 3 –7 8 9; 12 3 8 1]
   A =
        1     4    –5     6
        3    –7     8     9
       12     3     8     1
>> size(A)
   ans = 3     4
>> sum(A)
   ans = 16     0    11    16
>> sum(A,2)
   ans =
        6
       13
       24
>> prod(A)
   ans = 36   –84  –320    54
>> prod(A,2)
   ans =
         –120
        –1512
          288
>> mean(A)
   ans = 5.3333    0    3.6667    5.3333
>> mean(A,2)
   ans =
       1.5000
       3.2500
       6.0000
>> sort(A)
   ans =
        1    –7    –5     1
        3     3     8     6
       12     4     8     9
>> sort(A,2)
   ans =
       –5     1     4     6
       –7     3     8     9
        1     3     8    12
>> max(A)
   ans =
       12     4     8     9
```

```
>> [m,k] = max(A)
   m = 12     4     8     9
   k = 3      1     2     2
>> min(A)
   ans = 1    -7    -5     1
>> [n,t] = min(A)
   n = 1     -7    -5     1
   t = 1      2     1     3
```

4.3. ՄԱՏՐԻՑԻ ՏԱՐՐԵՐԻՆ ԴԻՄԵԼԸ

Մատրիցի որևէ տարրի կարելի է դիմել՝ գրելով մատրիցի անունը և, կլոր փակագծերում, ստորակետներով անջատած, ինդեքսները: Ինչպես վեկտորների դեպքում, մատրիցի տարրին դիմելով՝ կարելի է ստանալ այդ տարրի արժեքը, այն օգտագործել տարբեր արտահայտություններում, մատրիցի մեջ փոխել այդ տարրի արժեքը կամ այն մատրիցից հեռացնել: Այստեղ ևս հիշենք, որ **MatLab**–ում մատրիցի տարրերի ինդեքսավորումը սկսվում է 1–ից:

Օրինակ.

```
>> M = [3 11 6 5; 4 7 10 2; 13 9 0 8]
   M =
        3    11     6     5
        4     7    10     2
       13     9     0     8
>> M(3,1)
   ans = 13
>> M(2,3) = 20
   M =
        3    11     6     5
        4     7    20     2
       13     9     0     8
>> M(2,4) - M(1,2)
   ans = -9
```

Մատրիցներից կարելի է առանձնացնել ոչ միայն առանձին տարրեր, այլև բլոկներ: Դրա համար օգտվում են մատրիցների հասցեավորման (:) օպերատորից, որը նման է վեկտորների համար սահմանված նույն օպերատորին:

- Եթե մատրիցի հասցեավորումը գրվում է A(:, n) տեսքով, որտեղ *n*–ը բնական թիվ է, այն վերաբերում է այդ մատրիցի *n*–րդ սյան բոլոր տարրերին:
- Եթե մատրիցի հասցեավորումը գրվում է A(m, :) տեսքով, որտեղ *m*–ը բնական թիվ է, այն վերաբերում է այդ մատրիցի *m*–րդ տողի բոլոր տարրերին:

- Եթե մատրիցի հասցեավորումը գրվում է A(:, m:n) տեսքով, որտեղ *m*–ն ու *n*–ը բնական թվեր են, ապա այն վերաբերում է մատրիցի բոլոր տողերի *m*–ից *n*–րդ սյուների տարրերին:
- Եթե մատրիցի հասցեավորումը գրվում է A(m:n, :) տեսքով, որտեղ *m*–ն ու *n*–ը բնական թվեր են, ապա այն վերաբերում է մատրիցի *m*–ից *n*–րդ տողերի բոլոր սյուների տարրերին:
- Եթե մատրիցի հասցեավորումը գրվում է A(m:n, p:q) տեսքով, որտեղ *m*–ը, *n*–ը, *p*–ն և *q*–ն բնական թվեր են, ապա այն վերաբերում է մատրիցի *m*–ից *n*–րդ տողերի *p*–ից *q*–րդ սյուների տարրերին:
- Հասցեավորման A(:, :) գործողությունը կվերադարձնի հենց *A* մատրիցը:

Օրինակ.

```
>> A = [1 3 5 7 9; 2 4 6 8 10; 3 6 9 12 15; 4 8 12 16 20; 5 10 15
20 25]
   A =
        1     3     5     7     9
        2     4     6     8    10
        3     6     9    12    15
        4     8    12    16    20
        5    10    15    20    25
>> b = A(:, 3)
   b =
        5
        6
        9
       12
       15
>> c = A(2, :)
   c = 2     4     6     8    10
>> E = A(2:4, :)
   E =
        2     4     6     8    10
        3     6     9    12    15
        4     8    12    16    20
>> F = A(1:3, 2:4)
   F =
        3     5     7
        4     6     8
        6     9    12
```

Ինչպես վեկտորների դեպքում, եթե մատրիցի ինդեքսներից որևէ մեկը գրենք m:end (կամ m:p:end), որտեղ *m*–ը, *p*–ն ու *n*–ը բնական թվեր են (օրինակ՝ A(m:end, n)), ապա այն կվերաբերի այդ մատրիցի *m*–րդ տողից (կամ սյունից) մինչև վերջին ինդեքսով տարրերին:

Մատրիցների մեջ կարելի է ավելացնել նոր տարրեր, ամբողջական տողեր և սյուներ կամ նույնիսկ այլ մատրիցներ: Դրա համար ևս օգտվում են հասցեավորման գործողությունից:

Օրինակ.

Հետևյալ օրինակում E մատրիցը բաղկացած է 2 տողից և 4 սյունից: Եթե մենք այդ մատրիցի 3–րդ տողի բոլոր տարրերին վերագրենք E մատրիցի սյուների քանակին հավասար երկարությամբ տող–վեկտոր (ծրագրի 2–րդ հրամանը), ապա E մատրիցը կունենա 3 տող և 4 սյուն, իսկ 3–րդ տողում գրված կլինեն սահմանված տող–վեկտորի տարրերը: Ծրագրի 3–րդ հրամանով սահմանված է 3×3 չափի անկյունագծային մատրից, որը, 4–րդ հրամանով աջից միացված է E մատրիցին: Արդյունքում ստացվում է 3 տող և 7 սյուն ունեցող մատրից:

```
>> E = [1 2 3 4; 5 6 7 8]
   E =
        1     2     3     4
        5     6     7     8
>> E(3, :) = [10:4:22]
   E =
        1     2     3     4
        5     6     7     8
       10    14    18    22
>> K = eye(3)
   K =
        1     0     0
        0     1     0
        0     0     1
>> [E K]
   ans =
        1     2     3     4     1     0     0
        5     6     7     8     0     1     0
       10    14    18    22     0     0     1
```

Եթե մատրիցը $m\times n$ չափի է, և նրան ավելացվում է նոր անդամ, որի ինդեքսը մեծ է մատրիցի չափերից, ապա **MatLab**–ն այնպես է ավելացնում մատրիցի չափերը, որ կարողանա տեղավորել նոր մատրիցը: Ազատ մնացած մյուս բոլոր ինդեքսների տարրերն ընդունում են 0 արժեքները:

Օրինակ.

```
>> D = [1 5 –3; 8 4 2]
   D =
        1     5    –3
        8     4     2
>> D(5,6) = 45
   D =
        1     5    –3     0     0     0
        8     4     2     0     0     0
        0     0     0     0     0     0
        0     0     0     0     0     0
        0     0     0     0     0    45
```

Մատրիցի առանձին տողեր կամ սյուներ կարելի է հեռացնել՝ վերագրման օպերատորի աջ մասում գրելով դատարկ քառակուսի փակագծեր:

Օրինակ.

```
>> C = [5 7 8 49; 4 10 36 60; 56 13 59 8]
   C =
        5     7     8    49
        4    10    36    60
       56    13    59     8
>> C(:, 2:3) = []
   C =
        5    49
        4    60
       56     8
```

4.4. ԹՎԱԲԱՆԱԿԱՆ ԳՈՐԾՈՂՈՒԹՅՈՒՆՆԵՐ ՄԱՏՐԻՑՆԵՐԻ ՀԵՏ

Մատրիցների հետ կարելի է կատարել տարբեր թվաբանական գործողություններ: Դրանց մանրամասն նկարագրությունը բերված է այս բաժնում:

4.4.1. Գումարում և հանում

A և *B* երկու մատրիցների գումարումը կամ հանումը անդամ առ անդամ գործողություն է, ընդ որում այս գործողության դեպքում *A*–ն և *B*–ն պետք է ունենան միևնույն չափը: Արդյունքում ստացված *C* մատրիցը ևս ունի նույն չափը, ինչ *A*–ն և *B*–ն, և դրա յուրաքանչյուր տարրի արժեքը հավասար է *A* և *B* մատրիցների համապատասխան տարրերի գումարին կամ տարբերությանը:

Օրինակ.

```
>> A = [1 4 8 0; 2 -5 12 1; 3 7 9 -4]
   A =
        1     4     8     0
        2    -5    12     1
        3     7     9    -4
>> B = [15 2 8 7; -9 4 -5 1; 4 4 6 -2]
   B =
       15     2     8     7
       -9     4    -5     1
        4     4     6    -2
>> A + B
   ans =
       16     6    16     7
       -7    -1     7     2
        7    11    15    -6
>> A - B
   ans =
      -14     2     0    -7
       11    -9    17     0
       -1     3     3    -2
```

4.4.2. Մատրիցների բազմապատկումը

Մատրիցի և թվի բազմապատկումը կատարվում է նույն կերպ, ինչ վեկտորի ու թվի բազմապատկումը: Այն անդամ առ անդամ գործողություն է:

Երկու մատրիցներ կարող են բազմապատկվել միայն այն դեպքում, երբ առաջին մատրիցի սյուների քանակը հավասար է երկրորդ մատրիցի տողերի քանակին: Արդյունքում ստացված մատրիցն ունի առաջին մատրիցի տողերի քանակին հավասար տողեր և երկրորդ մատրիցի սյուների քանակին հավասար սյուներ: Այսինքն, եթե $m \times n$ չափի մատրիցը բազմապատկենք $n \times p$ չափի մատրիցով, կստանանք $m \times p$ չափի մատրից:

Ենթադրենք՝

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \quad \text{և} \quad B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ . & . & \dots & . \\ b_{n1} & b_{n2} & \dots & b_{np} \end{pmatrix}:$$

Այս դեպքում $A \cdot B$ արտադրյալի արդյունքում ստացվում է

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ . & . & \dots & . \\ c_{m1} & c_{m2} & \dots & c_{mp} \end{pmatrix}$$

մատրիցը, որտեղ

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \ \ (i = 1,2,...,m \, ; \ \ j = 1,2,...,p \,): \tag{4.1}$$

Օրինակ.

```
>> A = [1 8 4 5; 6 7 9 -2; 1 3 4 7]
   A =
        1     8     4     5
        6     7     9    -2
        1     3     4     7
>> B = [7 8; -1 4; 3 9; 12 1]
   B =
        7     8
       -1     4
        3     9
       12     1
```

```
>> C = A*B
   C =
       71    81
       38   155
      100    63
```

4.4.3. Հակադարձ մատրից: Մատրիցների բաժանումը

Մատրիցների բաժանման գործողությունը **MatLab**–ում նախատեսված է հանրահաշվական հավասարումների համակարգերի լուծման համար: Նախքան այս հավասարումներին անդրադառնալը, անհրաժեշտ է գաղափար ունենալ հակադարձ մատրիցի մասին:

Արդեն ասել ենք, որ **MatLab**–ում **eye(n)** ֆունկցիան նախատեսված է $n \times n$ չափի միավոր մատրից ստեղծելու համար: Ինչպես մաթեմատիկայից հայտնի է, երբ միավոր մատրիցը աջից կամ ձախից բազմապատկվում է մեկ այլ մատրիցով (մատրիցների բազմապատկման կանոնները հաշվի առնելով), ապա վերջինս մնում է անփոփոխ: Եթե զուգահեռներ տանենք սովորական սկալյար թվերի հետ, ապա կարելի է ասել, որ այս գործողությունը համարժեք է թիվը աջից կամ ձախից մեկով բազմապատկելուն: Միավոր մատրիցը սովորաբար նշանակվում է I տառով: Հետևաբար, կարելի է գրել, որ $AI = A$ կամ $IB = B$: Եթե A –ն քառակուսի մատրից է, ապա $AI = IA = A$:

B քառակուսի մատրիցը կոչվում է A քառակուսի մատրիցի հակադարձ մատրից ($B = A^{-1}$), եթե դրանց արտադրյալը միավոր մատրից է, այսինքն՝ $AB = BA = I$: A քառակուսի մատրիցի հակադարձ մատրիցը որոշելու համար **MatLab**–ում նախատեսված է **inv(A)** ֆունկցիան: Չնայած այս ֆունկցիայի առկայությանը, հակադարձ մատրիցը **MatLab**–ում կարելի է որոշել նաև A^(–1) հրամանով:

Օրինակ.

```
>> A = [2 1 4; 4 1 8; 2 -1 3]
   A =
        2     1     4
        4     1     8
        2    -1     3
>> B = inv(A)
   B =
       5.5000   -3.5000    2.0000
       2.0000   -1.0000         0
      -3.0000    2.0000   -1.0000
>> A*B
   ans =
        1     0     0
        0     1     0
        0     0     1
```

```
>> B*A
   ans =
        1     0     0
        0     1     0
        0     0     1
>> A*A^(-1)
   ans =
        1     0     0
        0     1     0
        0     0     1
```

Այժմ խոսենք հավասարումների համակարգերի լուծման և մատրիցների բաժանման գործողության մասին: Ինչպես հայտնի է,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases} \tag{4.2}$$

գծային հավասարումների համակարգը կարելի է ներկայացնել

$$AX = B \tag{4.3}$$

տեսքով, որտեղ

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_m \end{pmatrix}, \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{pmatrix}:$$

Եթե (4.3) հավասարման երկու կողմը ձախից բազմապատկենք A մատրիցի հակադարձ մատրիցով, ապա կստանանք.

$$A^{-1}AX = A^{-1}B\,, \tag{4.4}$$

որտեղից, հիշելով հակադարձ և միավոր մատրիցների հատկությունները՝ կստանանք.

$$IX = A^{-1}B \;\text{ և }\; X = A^{-1}B: \tag{4.5}$$

Հետևաբար, **MatLab**–ում (4.2) համակարգը լուծելու համար նախ անհրաժեշտ է սահմանել A, B մատրիցները և որոշել X մատրիցը, որը կներկայացվի սյուն–վեկտորի տեսքով: X–ը կարելի է որոշել՝ **MatLab**–ում գրելով X=A^(–1)*B, X=inv(A)*B կամ X=A\B հրամաններից որևէ մեկը:

Այսպիսով՝ ձախ բաժանումը (\) մատրիցների դեպքում օգտագործվում է (4.2) կամ (4.3) համակարգը լուծելու համար: Ցույց տանք, որ աջ բաժանումը ևս կարելի

է օգտագործել համակարգ լուծելիս: Իրոք, (4.2) համակարգը կարելի է գրել ոչ միայն (4.3) տեսքով, այլ նաև

$$XA = B \tag{4.6}$$

տեսքով: Միայն թե այս դեպքում, ինչպես հետևում է մատրիցների բազմապատկման կանոնից, X–ը և B–ն ոչ թե սյուն–վեկտորներ են, այլ տող–վեկտորներ՝ $X = \begin{pmatrix} x_1 & x_2 & \dots & x_m \end{pmatrix}$, $B = \begin{pmatrix} b_1 & b_2 & \dots & b_m \end{pmatrix}$: Եթե այժմ (4.6)–ի երկու կողմն աջից բազմապատկենք A մատրիցի հակադարձ մատրիցով, ապա կստանանք.

$$XAA^{-1} = BA^{-1}, \tag{4.7}$$

որտեղից կստանանք.

$$XI = BA^{-1} \text{ և } X = BA^{-1}: \tag{4.8}$$

Հետևաբար, եթե X–ը և B–ն սահմանել ենք որպես տող **MatLab**–ում (4.2) հավասարման լուծումը կարելի է որոշել X=B*A^(–1), X=B*inv(A) կամ X=B/A հրամաններից որևէ մեկով:

Այսպիսով՝ աջ և ձախ բաժանման գործողությունները **MatLab**–ում օգտագործվում են համապատասխանաբար (4.4) և (4.3) հավասարումները լուծելու համար: Ինչպես տեսանք, այստեղ անհրաժեշտ է ուշադիր լինել B մատրիցը սահմանելիս: Աջ բաժանումը կիրառելիս այն պետք է սահմանել որպես տող–վեկտոր, ձախ բաժանումը կիրառելիս՝ որպես սյուն–վեկտոր:

Օրինակ.

Որպես օրինակ, լուծենք $\begin{cases} 4x - 2y + 6z = 8 \\ 2x + 8y + 2z = 4 \\ 6x + 10y + 3z = 0 \end{cases}$ հավասարումների համակարգը՝ օգտագործելով ձախ բաժանումը:

```
>> A = [4 –2 6; 2 8 2; 6 10 3]
   A =
        4    –2     6
        2     8     2
        6    10     3
>> B = [8; 4; 0]
   B =
        8
        4
        0
>> X = A\B
   X =
      -1.8049
       0.2927
       2.6341
```

4.4.4. Անդամ առ անդամ գործողություններ

Քանի որ վեկտորները մատրիցների մասնավոր դեպքն են, ապա երկու մատրիցների անդամ առ անդամ բազմապատկման, բաժանման և աստիճանի բարձրացման գործողությունները կատարվում են ճիշտ նույն կերպ, ինչ վեկտորների դեպքում: Այս դեպքում պետք է ուշադիր լինել, որ երկու մատրիցներն էլ ունենան միևնույն չափը և օգտագործել .*, ./, .\, .^ օպերատորները:

Օրինակ.

```
>> A = [1 8 4 6; -6 7 2 0; 3 9 1 8]
   A =
        1     8     4     6
       -6     7     2     0
        3     9     1     8
>> B = [-3 7 -5 3; 7 8 9 10; 15 1 0 -7]
   B =
       -3     7    -5     3
        7     8     9    10
       15     1     0    -7
>> A.*B
   ans =
       -3    56   -20    18
      -42    56    18     0
       45     9     0   -56
>> A./B
   ans =
      -0.3333    1.1429   -0.8000    2.0000
      -0.8571    0.8750    0.2222         0
       0.2000    9.0000       Inf   -1.1429
>> B./A
   ans =
      -3.0000    0.8750   -1.2500    0.5000
      -1.1667    1.1429    4.5000       Inf
       5.0000    0.1111         0   -0.8750
>> A.^B
   ans =
     1.0e+007 *
       0.0000    0.2097    0.0000    0.0000
      -0.0280    0.5765    0.0001         0
       1.4349    0.0000    0.0000    0.0000
```

4.5. ՄԱՏՐԻՑՆԵՐԻ ՏԱՐԲԵՐԻ ՆԿԱՏՄԱՄԲ ՄԱԹԵՄԱՏԻԿԱԿԱՆ ՖՈՒՆԿՑԻԱՆԵՐԻ ԿԻՐԱՌՈՒՄԸ

Եթե հայտարարված է որևէ A մատրից, ապա, եթե վերջինիս նկատմամբ կիրառենք որևէ մաթեմատիկական ֆունկցիա, օրինակ՝ **sin(A)**, ապա **MatLab**–ը կստեղծի մի նոր, A մատրիցի չափն ունեցող մատրից, որի յուրաքանչյուր տարրի արժեքը հավասար է A մատրիցի համապատասխան տարրի սինուսի արժեքին:

Օրինակ.

```
>> A = [7 6.5 8 12 14; 9 4 1 5 -7; 9 8 1.2 -4 -1; 1 9 0.5 7 22]
   A =
       7.0000    6.5000    8.0000   12.0000   14.0000
       9.0000    4.0000    1.0000    5.0000   -7.0000
       9.0000    8.0000    1.2000   -4.0000   -1.0000
       1.0000    9.0000    0.5000    7.0000   22.0000
>> B = sin(A)
   B =
       0.6570    0.2151    0.9894   -0.5366    0.9906
       0.4121   -0.7568    0.8415   -0.9589   -0.6570
       0.4121    0.9894    0.9320    0.7568   -0.8415
       0.8415    0.4121    0.4794    0.6570   -0.0089
```

4.6. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Օգտագործելով **zeros** և **ones** ֆունկցիաները՝ հայտարարեք 4×5 չափի մատրից, որի առաջին երկու տողերի տարրերի արժեքները զրոներ են, վերջին երկու տողերինը՝ մեկեր:

2. Հայտարարեք 6×6 չափի մատրից այնպես, որ ունենա հետևյալ տեսքը.

$$M=\begin{pmatrix}0&0&1&1&0&0\\0&0&1&1&0&0\\1&1&1&1&1&1\\1&1&1&1&1&1\\0&0&1&1&0&0\\0&0&1&1&0&0\end{pmatrix}:$$

3. Առանց մատրիցների առանձին տարրերը մուտքագրելու՝ օգտվելով վեկտորներ սահմանելու տարբեր եղանակներից՝ հայտարարեք հետևյալ մատրիցները.

$$A=\begin{pmatrix}0&3&6&9&12&15&18\\750&650&550&450&350&250&150\\0&0.8333&1.6667&2.5&3.3333&4.1667&5\end{pmatrix},$$

$$B=\begin{pmatrix}1&4&7&10&13&16&19&22&25\\72&66&60&54&48&42&36&30&24\\0&0.125&0.25&0.375&0.5&0.625&0.75&0.875&1\end{pmatrix},\ C=\begin{pmatrix}1&0&4\\2&0&4\\3&0&4\\4&0&4\\5&0&4\end{pmatrix}:$$

4. Առանց մատրիցների առանձին տարրերը մուտքագրելու՝ հայտարարեք հետևյալ մատրիցները.

$$A=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&5\end{pmatrix},\quad B=\begin{pmatrix}0&0&0&0&0\\0&0&0&7&7\\0&0&0&7&7\end{pmatrix},$$

$$C=\begin{pmatrix}0&0&0&0&0\\0&0&1&2&3\\0&0&4&5&6\\0&0&7&8&9\end{pmatrix},\quad D=\begin{pmatrix}0&0&0&0&0\\0&0&1&10&20\\0&0&2&8&26\\0&0&3&6&32\end{pmatrix}$$

5. Հայտարարեք $A=\begin{pmatrix}6&43&2&11&87\\12&6&34&0&5\\34&18&7&41&9\end{pmatrix}$ մատրիցը և դրա միջոցով կազմեք

- 5 տարրերից բաղկացած $u1$ տող–վեկտորը, որը կազմված կլինի A մատրիցի 2–րդ տողի տարրերից,
- 3 տարրերից բաղկացած $u2$ տող–վեկտորը, որը կազմված կլինի A մատրիցի 4–րդ սյան տարրերից,
- 10 տարրերից բաղկացած $u3$ տող–վեկտորը, որը կազմված կլինի A մատրիցի 1–ին և 3–րդ տողերի տարրերից,
- 6 տարրերից բաղկացած $u4$ տող–վեկտորը, որը կազմված կլինի A մատրիցի 2–րդ և 5–րդ սյուների տարրերից,
- 3 տարրերից բաղկացած $c1$ սյուն–վեկտորը, որը կազմված կլինի A մատրիցի 3–րդ սյան տարրերից,
- 5 տարրերից բաղկացած $c2$ սյուն–վեկտորը, որը կազմված կլինի A մատրիցի 2–րդ տողի տարրերից,
- 10 տարրերից բաղկացած $c3$ սյուն–վեկտորը, որը կազմված կլինի A մատրիցի 1–ին և 2–րդ տողերի տարրերից:

6. Օգտագործելով **zeros**, **ones** և **eye** ֆունկցիաները՝ հայտարարեք հետևյալ մատրիցները.

$$X=\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\end{pmatrix},\quad Y=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\quad Z=\begin{pmatrix}1&1\\1&1\\1&1\end{pmatrix}:$$

7. Օգտագործելով **zeros** ֆունկցիան՝ հայտարարեք 7×7 չափի զրոյական մատրից: Այնուհետև, օգտագործելով հասցեավորման օպերատորը՝ հայտարարված մատրիցը ձևափոխեք այնպես, որ այն ունենա հետևյալ տեսքը.

$$A=\begin{pmatrix}4&4&4&0&5&5&5\\4&4&4&0&5&5&5\\3&3&3&0&5&5&5\\0&0&0&0&0&0&0\\8&8&7&0&9&9&9\\8&8&7&0&9&9&9\\8&8&7&0&9&9&9\end{pmatrix}:$$

8. Օգտագործելով **zeros** և **ones** հրամանները՝ հայտարարեք 3×5 չափի մատրից, որի 1–ին, 2–րդ, 5–րդ սյուների տարրերի արժեքները հավասար են 0–ի, իսկ 3–րդ, 4–րդ սյուների տարրերի արժեքները՝ 1–ի:

9. Հայտարարեք 5×7 չափի մատրից, որի 1–ին տողը բաղկացած է 1–ից 7 ամբողջ թվերից, 2–րդ տողը՝ 8–ից 14 ամբողջ թվերից, 3–րդ տողը 15–ից 21 ամբողջ թվերից և այլն: Այս մատրիցի օգնությամբ հայտարարեք նոր 3×4 չափի մատրից, որը կազմված կլինի 1–ին մատրիցի 2–րդից 4–րդ տողերից և 3–րդից 6–րդ սյուներից:

10. Հայտարարեք 3×3 չափի մեկերից կազմված A և 2×2 չափի 5–երից կազմված B մատրիցները: B մատրիցը միացնել A մատրիցին այնպես, որ վերջինս ունենա հետևյալ տեսքը.

$$A=\begin{pmatrix}1&1&1&0&0\\1&1&1&0&0\\1&1&1&0&0\\0&0&0&5&5\\0&0&0&5&5\end{pmatrix}:$$

11. Տրված են հետևյալ մատրիցները.

$$A=\begin{pmatrix}5&2&4\\1&7&-3\\6&-10&0\end{pmatrix},\quad B=\begin{pmatrix}11&5&-3\\0&-12&4\\2&6&1\end{pmatrix},\quad C=\begin{pmatrix}7&14&1\\10&3&-2\\8&-5&9\end{pmatrix}:$$

Այս մատրիցների օգնությամբ

- ցույց տվեք, որ $A+B=B+A$,
- ցույց տվեք, որ $A+(B+C)=(A+B)+C$,
- ցույց տվեք, որ $5(A+C)=5A+5C$,
- ցույց տվեք, որ $A(B+C)=AB+AC$,
- որոշեք՝ արդյո՞ք $AB=BA$,
- որոշեք՝ արդյո՞ք $A(BC)=(AB)C$,
- որոշեք՝ արդյո՞ք $(AB)^T=B^TA^T$, (T –ն տրանսպոնացված մատրիցն է),
- որոշեք՝ արդյո՞ք $(A+B)^T=A^T+B^T$,
- ստացեք A^2 մատրիցը,
- ստացեք A մատրիցի հակադարձ մատրիցը,
- ստացեք A^2+B^2-AB մատրիցը,
- ստացեք A, B և AC մատրիցների որոշիչները:

12. Լուծեք հետևյալ հավասարումների համակարգերը՝ օգտվելով մատրիցների բաժանման գործողությունից և Կրամերի եղանակից.

- $\begin{cases} y-3z=-7 \\ 2x+3y-z=9 \\ 4x+5y-2z=15 \end{cases}$,
- $\begin{cases} 2x-y=10 \\ -x+2y-z=0 \\ -y+z=-50 \end{cases}$,
- $\begin{cases} 2x+y+z-t=12 \\ x+5y-5z+6t=35 \\ -7x+3y-7z-5t=7 \\ x-5y+2z+7t=21 \end{cases}$,
- $\begin{cases} 5x+4y-2z+6t=4 \\ 3x+6y+6z+1.5t=13.5 \\ 6x+12y-2z+16t=20 \\ 4x-2y+2z-4t=6 \end{cases}$:

5. ԳՐԱՖԻԿՆԵՐԸ ԵՌԱՉԱՓ ՏԱՐԱԾՈՒԹՅԱՆ ՄԵՋ

5.1. ԵՌԱՉԱՓ ԳՐԱՖԻԿՆԵՐ

3–րդ գլխում ծանոթացանք պարամետրից կախված ֆունկցիաների երկչափ գրաֆիկների կառուցմանը: Հիշեցնենք, որ եթե հայտնի են t պարամետրից կախված $x(t)$ և $y(t)$ ֆունկցիաները, ապա $y(x)$ կախվածությունը կարելի է պատկերել **plot** ներդրված ֆունկցիայի օգնությամբ: Նման ձևով, եթե հայտնի են t պարամետրից կախված $x(t)$, $y(t)$ և $z(t)$ ֆունկցիաները, ապա այս կախվածությունը կարելի է պատկերել եռաչափ տարածության մեջ: Այս նպատակի համար նախատեսված է ներդրված **plot3** ֆունկցիան, որի գրելաձևը գրեթե չի տարբերվում **plot** ֆունկցիայի գրելաձևից: Տարբերությունն այն է, որ **plot3** ֆունկցիան որպես մուտքային արգումենտ ստանում է երեք վեկտոր, այսինքն՝ գրվում է **plot3(x,y,z)** տեսքով: **plot** ֆունկցիայի մնացած, ոչ պարտադիր արգումենտները ևս կիրառելի են **plot3** ֆունկցիայի համար: Մասնավորապես, եթե եռաչափ գրաֆիկը ցանկանում ենք կառուցել սև գույնի, կետագծիկավոր, ապա հրամանը պետք է գրվի **plot3(x,y,z,'k–.')** տեսքով:

Եռաչափ գրաֆիկը կառուցելու համար նախ անհրաժեշտ է հայտարարել t պարամետրը վեկտորի տեսքով, վերջինիս օգնությամբ հայտարարել x, y և z վեկտորները ու գրել **plot3** ֆունկցիան:

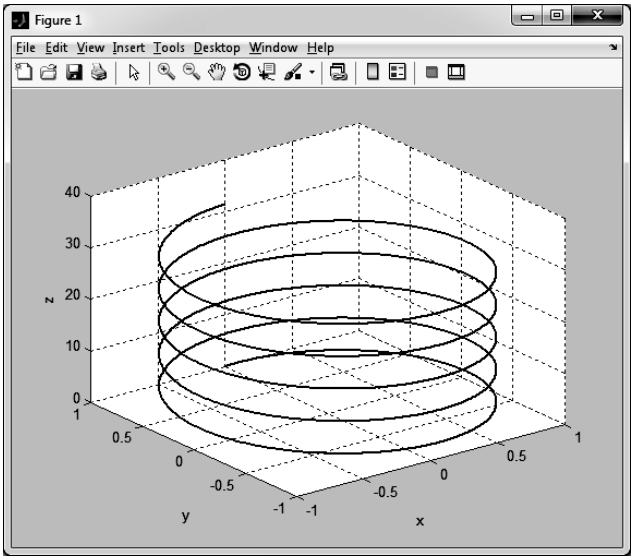
Օրինակ.

Կառուցենք $x = \sin t$, $y = \cos t$, $z = t$ պարամետրական ֆունկցիայի գրաֆիկը, որտեղ $t \in [0;10]$ (նկ. 21):

```
>> t = [0:0.01:10*pi];
>> x = sin(t);
>> y = cos(t);
>> z = t;
>> plot3(x,y,z,'k')
>> grid on
>> xlabel('x')
>> ylabel('y')
>> zlabel('z')
```

Բացի **plot3** ֆունկցիայից, **MatLab**–ում սահմանված է նաև **comet3** ֆունկցիան, որը հնարավորություն է տալիս շարժման մեջ քայլ առ քայլ հետևել ֆունկցիայի գրաֆիկի կառուցմանը եռաչափ տարածության մեջ: **comet3** ֆունկցիայի գրելաձևը գրեթե չի տարբերվում հարթության մեջ կետի շարժումը դիտարկելու համար նախատեսված **comet** ֆունկցիայի գրելաձևից, որը մանրամասն դիտարկել

ենք 3.10. բաժնում: Տարբերությունն այն է, որ **comet3** ֆունկցիան որպես մուտքային արգումենտ ստանում է x, y և z վեկտորները:



Նկ. 21

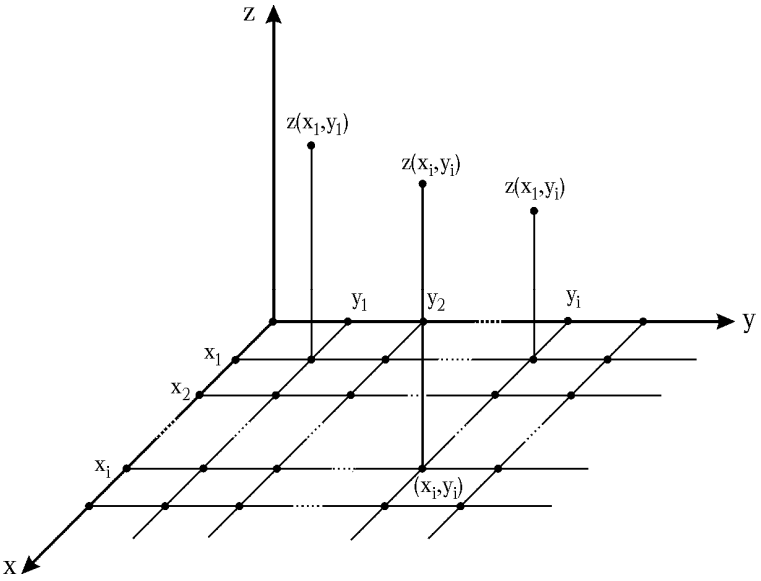
5.2. ՄԱԿԵՐԵՎՈՒՅԹՆԵՐ

5.2.1. Մակերևույթ կառուցելու հիմնական քայլերը

Տեսնենք, թե գրաֆիկորեն ինչպես կարելի է ներկայացնել երկու՝ x և y փոփոխականներից կախված $z = f(x,y)$ ֆունկցիան: Եռաչափ տարածության մեջ այս ֆունկցիաների գրաֆիկները մակերևույթներ են: $z = f(x,y)$ ֆունկցիայի մակերևույթը կառուցելիս պետք է հաշվի առնել, որ յուրաքանչյուր x և y զույգի պետք է համապատասխանի ֆունկցիայի որոշակի z արժեք: Նկատի ունենալով սա՝ մակերևույթ կառուցելու համար անհրաժեշտ է կատարել հետևյալ քայլերը.

1. հայտարարել x և y արգումենտների որոշման տիրույթները (x և y վեկտորները),
2. հայտարարված x և y վեկտորների օգնությամբ XY հարթության մեջ կառուցել ցանց, որի հանգույցները (x,y) զույգերն են,
3. կառուցված ցանցի հանգույցներից յուրաքանչյուրում որոշել z ֆունկցիայի արժեքը,
4. կառուցել $z = f(x,y)$ ֆունկցիայի մակերևույթը,
5. անհրաժեշտության դեպքում ձևավորել մակերևույթը՝ ընտրել մակերևույթի գունային տիրույթը, վերնագրեր ավելացնել առանցքերին կամ մակերևույթին, փոխել մակերևույթի դիտման անկյունը և այլն:

x և y արգումենտները սահմանվում են վեկտորների տեսքով ճիշտ այնպես, ինչպես սահմանել ենք մինչ այժմ: Ցանցը XY հարթության մեջ ֆունկցիայի որոշման տիրույթում այն կետերի բազմությունն է, որոնք օգտագործվելու են z ֆունկցիայի արժեքները որոշելու համար (նկ. 22): Ցանցի խտությունը, այսինքն՝ հարթության մեջ կետերի քանակը, որոշվում է x և y վեկտորների տարրերի քանակով: Ցանցը կառուցվում է **MatLab**–ի **meshgrid** ներդրված ֆունկցիայի օգնությամբ: **meshgrid** ֆունկցիան որպես արգումենտներ ստանում է x և y վեկտորները և վերադարձնում երկու մատրից, որոնցից մեկը պարունակում է ցանցի բոլոր կետերի x կոորդինատները, իսկ մյուսը՝ y կոորդինատները:



Նկ. 22

Կառուցված ցանցն ավելի հստակ պատկերացնելու համար դիտարկենք հետևյալ օրինակը:

Օրինակ.

Կառուցենք ցանց $x \in [-1;2]$ և $y \in [0;4]$ տիրույթում, որտեղ x –ը և y –ը փոխվում են 1 քայլով:

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

```
Y =

     0     0     0     0
     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
```

Ցանցի հանգույցները որոշվում են X և Y մատրիցներով. X մատրիցը պարունակում է ցանցի բոլոր կետերի x կոորդինատները, իսկ Y մատրիցը՝ y կոորդինատները: X մատրիցի բոլոր տողերը բաղկացած են նույն տարրերից, որովհետև ցանցի y առանցքին զուգահեռ գծերի վրա բոլոր հանգույցների x կոորդինատները նույնն են: Այդ նույն պատճառով նույն տարրերից են բաղկացած նաև Y մատրիցի բոլոր սյուները:

Եթե ցանցը քառակուսի է, ապա **meshgrid** ֆունկցիան կարելի է գրել մեկ արգումենտով, այսինքն՝ **meshgrid(x)** և **meshgrid(x,x)** ֆունկցիաներն իրար համարժեք են:

5.2.2. Մակերևույթ կառուցելու համար նախատեսված mesh և surf ֆունկցիաները

Ցանցը կառուցելուց հետո վերջինիս հանգույցներից յուրաքանչյուրում որոշում են $z = f(x, y)$ ֆունկցիայի արժեքները և կառուցում դրա մակերևույթը: Ծրագրում z ֆունկցիայի արգումենտները X և Y կոորդինատ–մատրիցներն են, իսկ գործողությունները կատարվում են անդամ առ անդամ: Մակերևույթը կառուցելու համար նախատեսված են մի շարք ֆունկցիաներ, որոնցից առավել հաճախ օգտագործվում են **mesh** և **surf** ֆունկցիաները: **mesh** ֆունկցիան մակերևույթը կառուցում է z ֆունկցիայի կետերը միացնող գծերի տեսքով, այսինքն՝ դատարկ վանդակների տեսքով, որոնց եզրերի գույները համապատասխանում են տվյալ կետում ֆունկցիայի արժեքներին, իսկ **surf** ֆունկցիան այդ վանդակները ներկում է տրված կետում ֆունկցիայի արժեքին համապատասխանող գույնով: Այս ֆունկցիաները գրվում են **mesh(X,Y,Z)** և **surf(X,Y,Z)** տեսքերով, որտեղ X –ը և Y –ը մատրիցներ են, որոնք պարունակում են **meshgrid** ֆունկցիայի օգնությամբ կառուցված ցանցի կոորդինատները, իսկ Z –ը նույն չափի մատրից է, որը պարունակում է z ֆունկցիայի արժեքը այդ ցանցի յուրաքանչյուր հանգույցում:

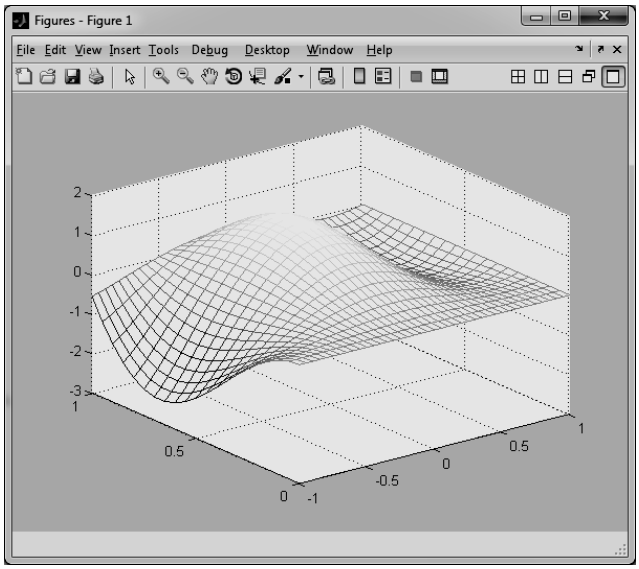
Օրինակ.

Կառուցենք $z = 2\cos \pi x \cdot \sin 3y \cdot (1 - x^3)y$ մակերևույթը $-1 \le x \le 1$, $0 \le y \le 1$ տիրույթում՝ x և y կոորդինատների քայլը վերցնելով 0.05: Մակերևույթը կկառուցենք ինչպես **mesh**, այնպես էլ **surf** ֆունկցիաների օգնությամբ:

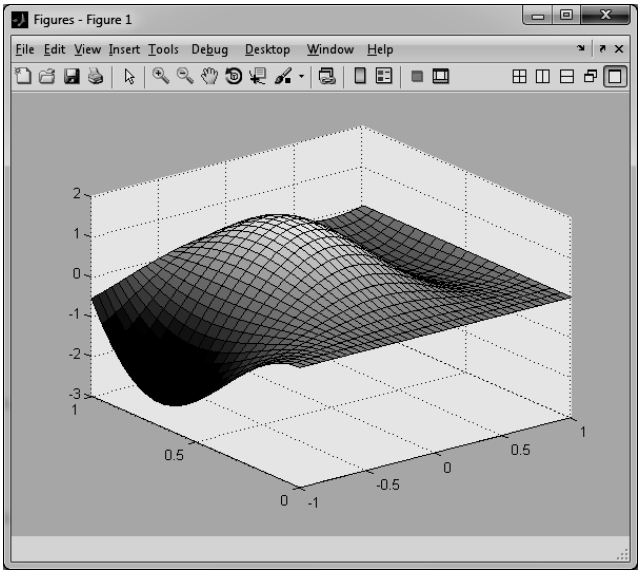
```
>> x = [-1:0.05:1];
>> y = [0:0.05:1];
```

```
>> [X,Y] = meshgrid(x,y);
>> Z = 2*cos(pi*X).*sin(3*Y).*(1–X.^3).*Y;
>> mesh(X,Y,Z)
>> figure
>> surf(X,Y,Z)
```

mesh ֆունկցիայի օգնությամբ ստացված ֆունկցիայի մակերևույթը ներկայացված է նկ. 23–ում, իսկ **surf** ֆունկցիայի օգնությամբ ստացված մակերևույթը՝ նկ. 24–ում:



Նկ. 23



Նկ. 24

mesh և **surf** ֆունկցիաները մակերևույթի գունավորման միջոցով մոտավոր պատկերացում են տալիս ֆունկցիայի արժեքների մասին: **MatLab**–ում սահմանված է նաև **colorbar** հրամանը, որը գրվում է **mesh** և **surf** ֆունկցիաներից հետո: Այս հրամանի օգնությամբ կառուցված մակերևույթի կողքին հայտնվում է գունային սանդղակ, որը համապատասխանություն է մտցնում ֆունկցիայի արժեքների և մակերևույթի գույների միջև:

Ինչպես տեսնում ենք **mesh** և **surf** ֆունկցիաների օգնությամբ մակերևույթ կառուցելիս **MatLab**–ը պատկերում է միայն դրա տեսանելի մասը: Մակերևույթը կարելի է թափանցիկ դարձնել, եթե այն կառուցելուց հետո ավելացնենք **hidden off** հրամանը: Նախկին տեսքին բերելու համար պետք է գրել **hidden on** հրամանը:

surf ֆունկցիան մակերևույթը կառուցում է վանդակավոր ցանցի տեսքով: Վանդակներից յուրաքանչյուրը ներկվում է տվյալ կետում ֆունկցիայի արժեքին համապատասխանող գույնով: **MatLab**–ը հնարավորություն է տալիս մակերևույթը կառուցել այնպես, որ ցանցը չերևա, փոխարենը երևա միայն գունավորված մակերևույթը: Դրա համար մակերևույթը կառուցելուց հետո պետք է ավելացնել **shading flat** կամ **shading interp** հրամանները: Սրանցից առաջինը հեռացնում է ցանցը, իսկ երկրորդը դրանից բացի մակերևույթը ներկում է սահուն գունային անցումներով: Մակերևույթը նախկին տեսքին բերում է **shading faceted** հրամանը:

5.2.3. Մակերևույթների ձևավորումը

Գրաֆիկների նման, մակերևույթները ևս կարելի է ձևավորել կառուցելուց հետո: Ձևավորելու համար կարելի է օգտվել գրեթե բոլոր այն ֆունկցիաներից, որոնք օգտագործել ենք գրաֆիկները ձևավորելիս: Այստեղ մանրամասն չենք նկարագրի մակերևույթների ձևավորման ֆունկցիաները՝ այդ մասին մանրամասն խոսվել է 3–րդ գլխում: Փոխարենը, պարզապես կթվարկենք այն ֆունկցիաները, որոնցից կարելի է օգտվել մակերևույթները ձևավորելու համար:

Մասնավորապես, մակերևույթին վերնագիր ավելացնելու համար օգտագործվում է **title** ֆունկցիան: x, y և z առանցքներին անուն կարելի է տալ համապատասխանաբար **xlabel**, **ylabel** և **zlabel** ֆունկցիաներով: Մակերևույթին լեգենդ կարելի է ավելացնել՝ օգտագործելով **legend** ֆունկցիան, իսկ գրաֆիկական պատուհանում սովորական տեքստեր՝ **text** կամ **gtext** ֆունկցիաների օգնությամբ: Տեքստերը կարելի է գրել նաև **TeX** ստանդարտի հրամաններով, որոնք ևս նկարագրել ենք 3–րդ գլխում:

Ինչպես սովորական գրաֆիկների դեպքում, մի քանի մակերևույթ միևնույն գրաֆիկական պատուհանում կարելի է տեղադրել **hold on** և **hold off** հրամանների

օգնությամբ: Կարելի է օգտվել նաև **subplot** ֆունկցիայից՝ գրաֆիկական պատուհանը ենթապատուհանների տրոհելու համար:

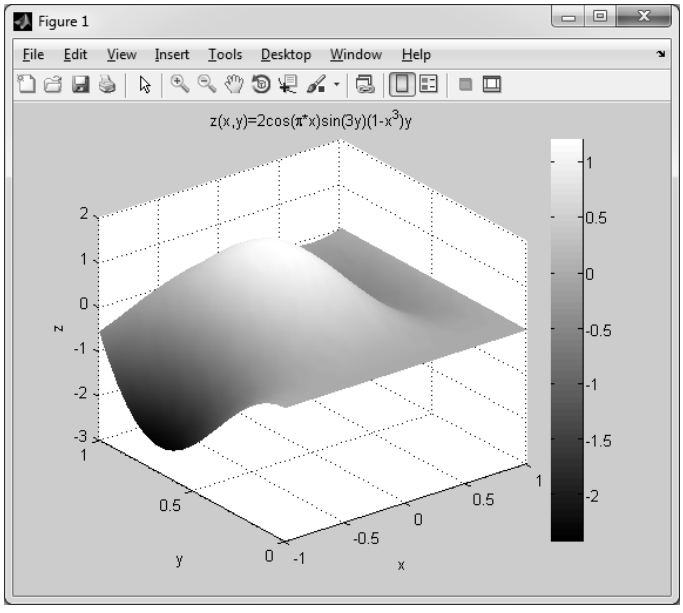
MatLab–ը հնարավորություն է տալիս փոխել նաև մակերևույթի գունային տիրույթը: Դրա համար նախատեսված է **colormap** ֆունկցիան, որի արգումենտում գրվում է անհրաժեշտ գունային տիրույթի հրամանը: Գունային տիրույթների հրամանների ցանկը բերված է հետևյալ աղյուսակում:

Մակերևույթի գունային տիրույթի հրամանները	
Անվանումը	**Նկարագրությունը**
autumn	Սահուն գունային անցում է կատարում կարմիր–նարնջագույն–դեղին տիրույթում:
bone	Մակերևույթը կառուցում է մոխրագույնին մի փոքր կապույտ խառնված գունային տիրույթում:
colorcube	Յուրաքանչյուր գույն փոխվում է մուգ երանգից մինչև բաց երանգ:
cool	Մակերևույթը կառուցվում է երկնագույնի և մուգ կարմիրի երանգներով:
copper	Մակերևույթը կառուցվում է պղնձագույնի երանգներով:
flag	Մակերևույթը կառուցվում է կարմիր–սպիտակ–կապույտ–սև գույների ցիկլային փոփոխությամբ:
gray	Մակերևույթը կառուցվում է մոխրագույնի երանգներով:
hot	Սահուն գունային անցում է կատարում սև–կարմիր–նարնջագույն–դեղին–սպիտակ գունային տիրույթում:
hsv	Սահուն գունային անցում է կատարում ծիածանի գունային տիրույթում:
jet	Սահուն գունային անցում է կատարում կապույտ–բաց կապույտ–դեղին–նարնջագույն–կարմիր գունային տիրույթում:
pink	Մակերևույթը կառուցում է մոխրագույնին մի փոքր շագանակագույն խառնված գունային տիրույթում:
prism	Մակերևույթը կառուցվում է կարմիր–նարնջագույն–դեղին–կանաչ–կապույտ–մանուշակագույն գույների ցիկլային փոփոխությամբ:
spring	Մակերևույթը կառուցվում է մուգ կարմիրի և դեղինի երանգներով:
summer	Մակերևույթը կառուցվում է կանաչի և դեղինի երանգներով:
vga	Մակերևույթը կառուցվում է **Windows** օպերացիոն համակարգի 16 գույնից բաղկացած գունային տիրույթում:
white	Մակերևույթը կառուցվում է սպիտակ գույնով:
winter	Մակերևույթը կառուցվում է կապույտի և կանաչի երանգներով:

Օրինակ.

Կառուցենք նախորդ խնդրի $z = 2\cos \pi x \cdot \sin 3y \cdot (1 - x^3)y$ ֆունկցիայի մակերևույթը, որտեղ $-1 \le x \le 1$, $0 \le y \le 1$, իսկ հանգույցների կոորդինատները փոխվում են 0.05 քայլով: Այստեղ կօգտվելով վերևում սահմանված **colorbar** ֆունկցիայից, որը գունային սանդղակ է տեղադրում մակերևույթի կողքին, **shading interp** հրամանից, որը հեռացնում է մակերևույթի վրայի վանդակները և այն ներկում սահուն փոխվող գույներով, ինչպես նաև **xlabel**, **ylabel**, **zabel** և **title** հրամաններից, որոնք անուններ կտան x, y և z առանցքներին և վերնագիր կավելացնեն մակերևույթին: Նկատի ունենանք, որ **title** ֆունկցիան տվյալ օրինակում պարունակում է նաև **TeX** ստանդարտի հրամաններ, որոնք թույլ են տալիս վերնագրին ավելացնել աստիճան և հունարեն տառ: Մակերևույթը կկառուցենք **gray** գունային տիրույթում՝ օգտվելով **colormap(gray)** ֆունկցիայից (նկ. 25):

```
x = [-1:0.05:1];
y = [0:0.05:1];
[X,Y] = meshgrid(x,y);
Z = 2*cos(pi*X).*sin(3*Y).*(1-X.^3).*Y;
surf(X,Y,Z)
colorbar
shading interp
colormap(gray)
xlabel('x')
ylabel('y')
zlabel('z')
title('z(x,y)=2cos(\pi*x)sin(3y)(1-x^3)y')
```



Նկ. 25

5.2.4. Մակերևույթ կառուցելու համար նախապատրաստված այլ ֆունկցիաներ

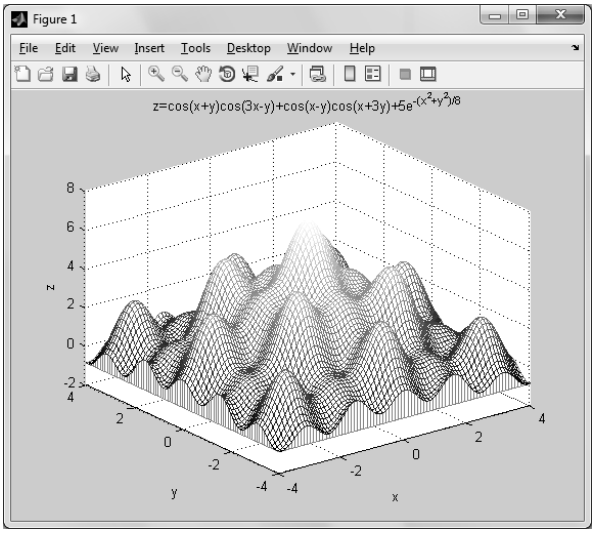
Բացի **mesh** և **surf** ֆունկցիաներից՝ **MatLab**–ում մակերևույթներ կառուցելու համար սահմանված են մի շարք ֆունկցիաներ ևս: Ստորև կբերենք դրանց նկարագրությունը և յուրաքանչյուրով, որպես օրինակ կկառուցենք

$$z = \cos(x+y)\cos(3x-y) + \cos(x-y)\sin(x+3y) + 5e^{-\frac{(x^2+y^2)}{8}}$$

մակերևույթը $x \in [-4;4]$ և $y \in [-4;4]$ տիրույթում: Ծրագիր կգրենք միայն **meshz** ֆունկցիայի համար: Մնացած ֆունկցիաների դեպքում ծրագիրն ունի նույն տեսքը, միայն **meshz(X,Y,Z)** ֆունկցիան է փոխարինվում դրանցից որևէ մեկով:

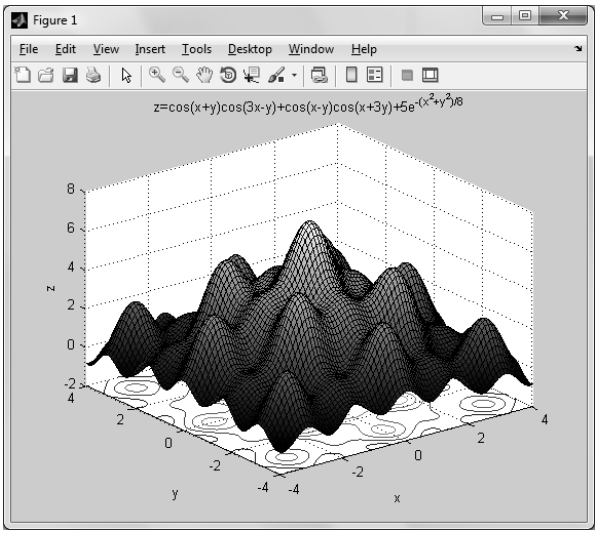
- **meshz** ֆունկցիան նման է **mesh** ֆունկցիային, միայն մակերևույթի եզրային կետերից z առանցքին զուգահեռ ուղիղ գծեր են իջնում XY հարթության վրա՝ շեշտելով մակերևույթի կտրվածքները (նկ. 26):

```
x = [-4:0.1:4];
y = [-4:0.1:4];
[X,Y] = meshgrid(x,y);
Z=cos(X+Y).*cos(3*X-Y)+cos(X-Y).*cos(X+3*Y)+ ...
  5*exp(-X.^2+Y.^2)/8);
meshz(X,Y,Z)
xlabel('x'); ylabel('y'); zlabel('z')
title('z=cos(x+y)cos(3x-y)+cos(x-y)cos(x+3y)+ ...
      5e^{-(x^2+y^2)/8}')
grid on
colormap(gray)
```



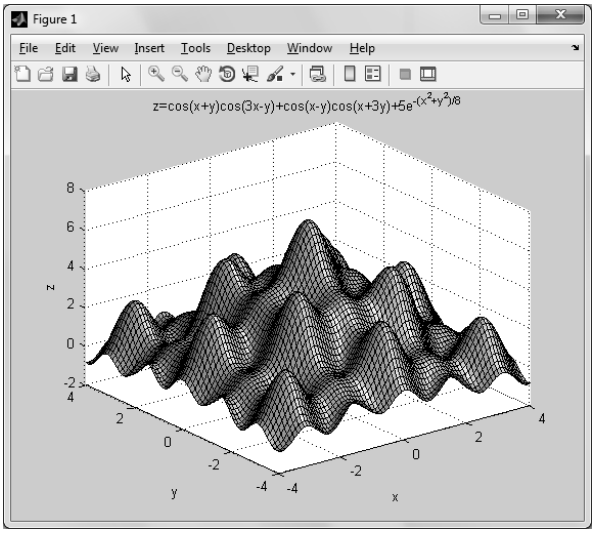
Նկ. 26

- **meshc** և **surfc** ֆունկցիաները նման են **mesh** և **surf** ֆունկցիաներին, միայն թե մակերևույթ կառուցելուց բացի *XY* հարթության մեջ կառուցում են մակերևույթի կոնտուրային պրոյեկցիաները (նկ. 27)։



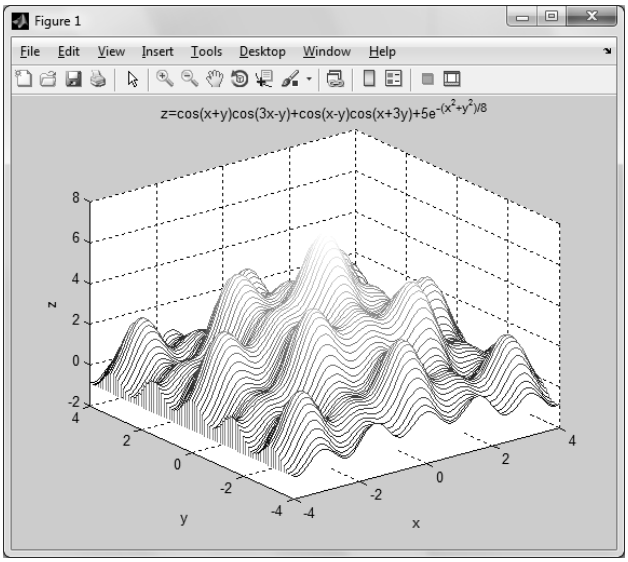
Նկ. 27

- **surfl** ֆունկցիան կառուցում է լուսավորված մակերևույթ (նկ. 28)։



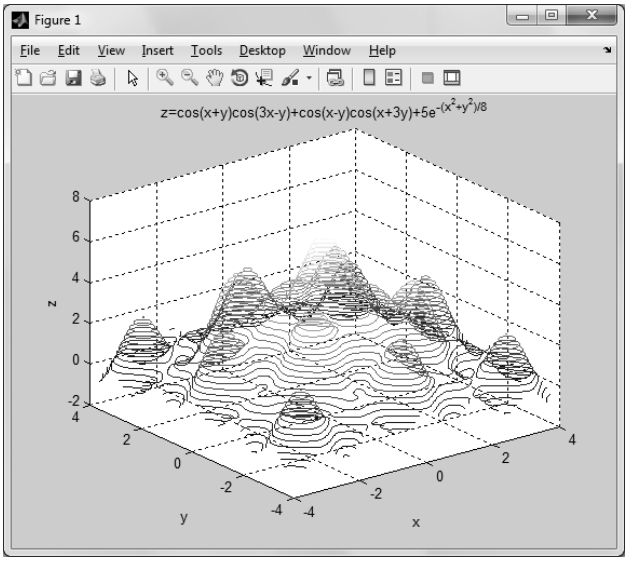
Նկ. 28

- **waterfall** ֆունկցիան մակերևույթը կառուցում է ոչ թե վանդակներով, այլ զուգահեռ գծերի տեսքով (նկ. 29)։



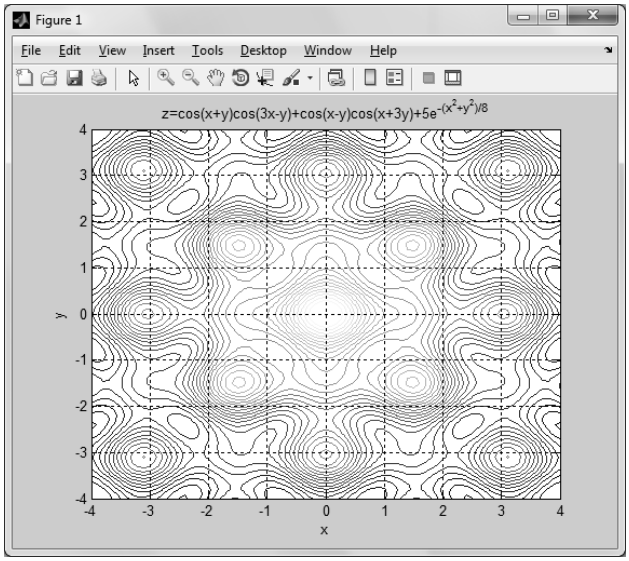
Նկ. 29

- **contour3** ֆունկցիան մակերևույթը կառուցում է եռաչափ կոնտուրների տեսքով: Այս ֆունկցիան կարող է ստանալ ոչ պարտադիր չորրորդ արգումենտ՝ **contour3(X,Y,Z,n)**, որտեղ *n*–ը կոնտուրների թիվն է: Նկ. 30-ում պատկերված մակերևույթը կառուցվել է 30 եռաչափ կոնտուրներով:



Նկ. 30

- **contour** ֆունկցիան կառուցում է կոնտուրային մակերևույթի պրոյեկցիան XY հարթության մեջ: Այս ֆունկցիան ևս կարող է ստանալ ոչ պարտադիր չորրորդ արգումենտ՝ **contour(X,Y,Z,n)**, որտեղ n–ը կոնտուրների թիվն է: Նկ. 31–ում բերված կոնտուրային մակերևույթը կառուցվել է 30 կոնտուրներով:



Նկ. 31

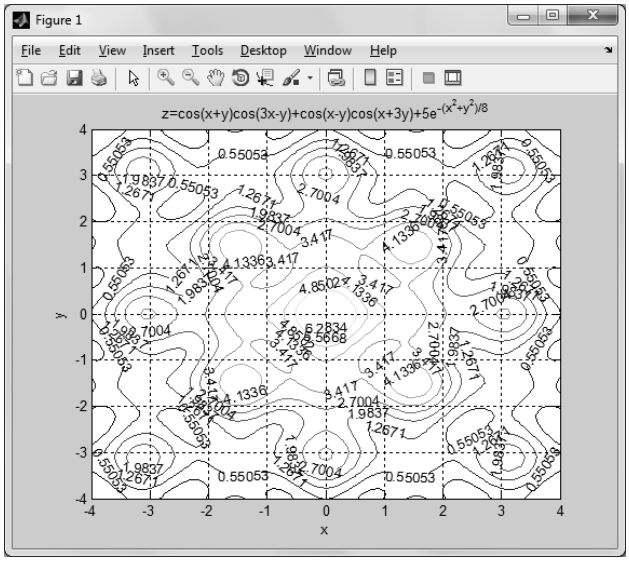
Կոնտուրային գրաֆիկները մեզ քիչ տեղեկություն են տալիս, քանի որ յուրաքանչյուր կոնտուրում մենք չգիտենք ֆունկցիայի արժեքը՝ չնայած այն բանին, որ դրանք ներկված են համապատասխան գույներով: Իհարկե, կարելի է ավելացնել **colormap** հրամանը և տեսնել, թե որ գույնին ֆունկցիայի որ արժեքն է համապատասխանում: Սակայն դա էլ ճշգրիտ արժեքը ստանալու հնարավորություն չի տալիս: Որպեսզի ավելի լավ տեղեկություն ստանանք ֆունկցիայի արժեքների մասին, կարելի է օգտվել ներդրված **clabel** ֆունկցիայից: **contour** ֆունկցիան, բացի այն, որ կառուցում է կոնտուրային գրաֆիկ, նաև որպես ելք վերադարձնում է երկու արգումենտ, որոնք գրելով **clabel** ֆունկցիայի մուտքային արգումենտներում՝ կստանանք ֆունկցիայի արժեքները կոնտուրներից յուրաքանչյուրի վրա:

Օրինակ.

Ստորև բերված ծրագրում կառուցվել է նախորդ օրինակներում բերված ֆունկցիայի կոնտուրային գրաֆիկը, այս անգամ 10 կոնտուրով և **clabel** ֆունկցիայի օգնությամբ ավելացվել են ֆունկցիայի արժեքները յուրաքանչյուր կոնտուրի վրա (նկ. 32):

```
x = [-4:0.1:4];
y = [-4:0.1:4];
```

```
[X,Y] = meshgrid(x,y);
Z=cos(X+Y).*cos(3*X-Y)+cos(X-Y).*cos(X+3*Y)+5*exp(-X.^2+Y.^2)/8);
[C,H] = contour(X,Y,Z,10)
clabel(C,H)
xlabel('x')
ylabel('y')
zlabel('z')
title('z=cos(x+y)cos(3x-y)+cos(x-y)cos(x+3y)+5e^{-(x^2+y^2)/8}')
grid on
```

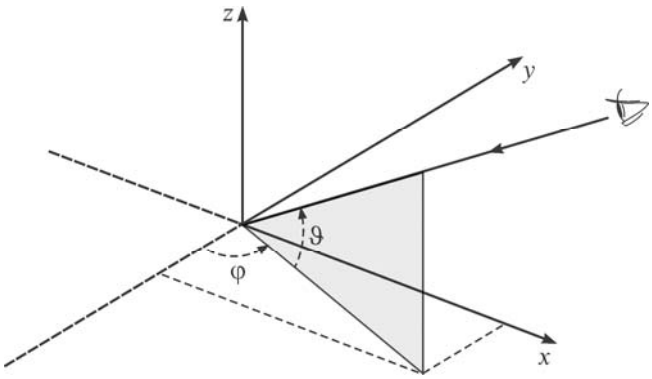


Նկ. 32

5.2.5 Մակերևույթի դիտման անկյան ընտրությունը

MatLab–ում մակերևույթի ուղղությունը որոշվում է φ լայնական և ϑ երկայնական անկյուններով (նկ. 33): φ անկյունը *XY* հարթության մեջ է և չափվում է բացասական *y* կիսաառանցքից, իսկ ϑ–ն դիտման ուղղության անկյունն է *XY* հարթությունից: Երկու անկյունն էլ չափվում են աստիճաններով:

Եթե հատուկ չի նշվում, ապա մակերևույթը երևում է $\varphi = -37.5^{o}$ և $\vartheta = 30^{o}$ անկյունների տակ: Մակերևույթի դիտման անկյունը փոխելու համար նախատեսված է **view(phi, teta)** ներդրված ֆունկցիան, որտեղ phi փոփոխականը ներկայացնում է φ լայնական անկյունը, իսկ teta–ն՝ ϑ երկայնական:



Նկ. 33

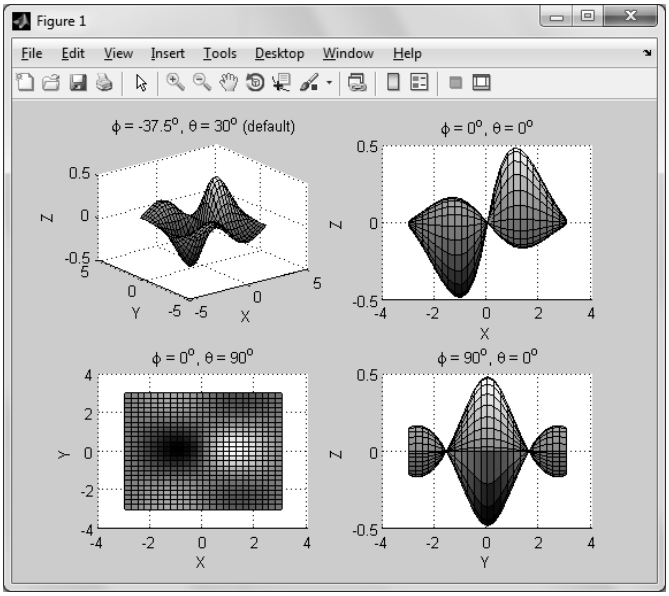
Լայնական և երկայնական անկյունների համապատասխան ընտրությամբ կարելի է դիտել մակերևույթի պրոյեկցիաները տարբեր կոորդինատական հարթություններում: Օրինակ, երբ $\varphi = 0^o$, $\vartheta = 90^o$, ապա մակերևույթը կտեսնենք վերևից՝ XY հարթության մեջ: Եթե $\varphi = 0^o$, $\vartheta = 0^o$, մակերևույթը կտեսնենք կողքից՝ XZ հարթության մեջ, իսկ $\varphi = 90^o$, $\vartheta = 0^o$ դեպքում մակերևույթը դարձյալ կտեսնենք կողքից՝ այս անգամ YZ հարթության մեջ:

Օրինակ.

Կառուցենք $z = 2^{-0.8\sqrt{x^2+y^2}} \cos y \sin x$ մակերևույթը $-3 \le x \le 3$, $-3 \le y \le 3$ տիրույթում՝ x և y կոորդինատների քայլը վերցնելով 0.25: Օգտագործելով **subplot** ներդրված ֆունկցիան՝ նույն գրաֆիկական պատուհանի 4 ենթապատուհաններում դիտենք այս մակերևույթը նախ առանց **view** ֆունկցիայից օգտվելու ($\varphi = -37.5^o$, $\vartheta = 30^o$ անկյան տակ), այնուհետև՝ $\varphi = 0^o$, $\vartheta = 0^o$ (XZ հարթության մեջ), $\varphi = 0^o$, $\vartheta = 90^o$ (XY հարթության մեջ) և $\varphi = 90^o$, $\vartheta = 0^o$ անկյունների տակ (YZ հարթության մեջ) (նկ. 34):

```
x = [-3:0.25:3]; y = [-3:0.25:3];
[X,Y] = meshgrid(x,y);
Z=2.^(-0.8*sqrt(X.^2 + Y.^2)).*cos(Y).*sin(X);
subplot(2,2,1)

surf(X,Y,Z)
xlabel('X'); ylabel('Y'); zlabel('Z')
title('\phi = -37.5^o, \theta = 30^o (default)')

subplot(2,2,2)
surf(X,Y,Z)
view(0,0)
xlabel('X'); ylabel('Y'); zlabel('Z')
title('\phi = 0^o, \theta = 0^o')

subplot(2,2,3)
```

```
surf(X,Y,Z)
view(0,90)
xlabel('X'); ylabel('Y'); zlabel('Z')
title('\phi = 0^o, \theta = 90^o')

subplot(2,2,4)
surf(X,Y,Z)
view(90,0)
xlabel('X'); ylabel('Y'); zlabel('Z')
title('\phi = 90^o, \theta = 0^o')
```



Նկ. 34

5.2.6. Պարամետրական տեսքով տրված ֆունկցիաների մակերևույթները

Եթե x, y, z կոորդինատները կապված են ոչ թե $z = f(x,y)$ տիպի բացահայտ ֆունկցիոնալ կապով, այլ որոշակի u և v պարամետրերից կախված $x = x(u,v)$, $y = y(u,v)$, $z = z(u,v)$ անբացահայտ ձևով, ապա ասում են, որ մակերևույթի հավասարումը տրված է պարամետրական տեսքով: Այսպիսի ֆունկցիաների մակերևույթները կառուցելու համար նախ անհրաժեշտ է u և v պարամետրերը սահմանել վեկտորների տեսքով, դրանցով կառուցել ցանց **meshgrid** ֆունկցիայի օգնությամբ, այնուհետև ստացված մատրիցներով սահմանել x, y, z ֆունկցիաները և կառուցել ֆունկցիայի մակերևույթը 5.2.2. , 5.2.4. կետերում նկարագրված ֆունկցիաներից որևէ մեկով: Հարմար է u և v վեկտորները սահմանել

linspace ֆունկցիայի օգնությամբ, որպեսզի դրանք ունենան միևնույն երկարությունը:

Օրինակ.

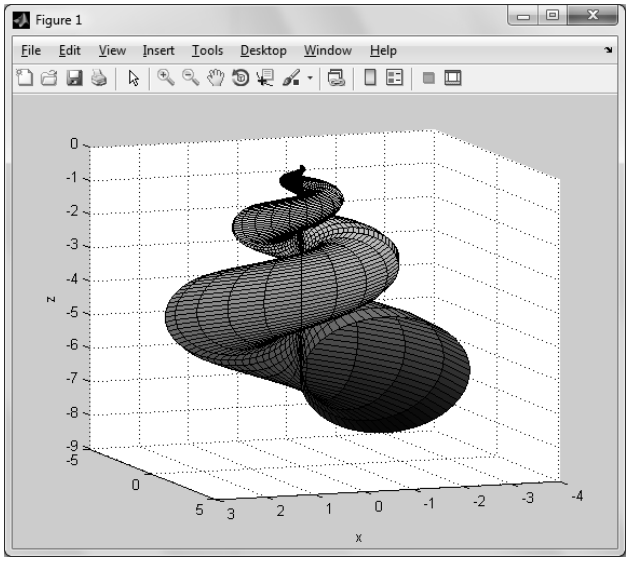
Կառուցենք

$$x = 2\left(1 - e^{u/(6\pi)}\right)\cos u \cos^2\frac{v}{2},\ \ y = 2\left(-1 + e^{u/(6\pi)}\right)\sin u \cos^2\frac{v}{2},$$

$$z = 1 - e^{u/(3\pi)} - \sin v + e^{u/(6\pi)}\sin v$$

պարամետրական տեսքով տրված ֆունկցիայի մակերևույթը, որտեղ $u \in [0; 6\pi]$, $v \in [0; 2\pi]$ և դիտենք այս մակերևույթը $\varphi = 160^o$ և $\vartheta = 10^o$ անկյունների տակ (նկ. 35):

```
u = linspace(0,6*pi,60);
v = linspace(0,2*pi,60);
[U,V] = meshgrid(u,v);
x = 2*(1-exp(U/(6*pi))).*cos(U).*cos(V/2).^2;
y = 2*(-1+exp(U/(6*pi))).*sin(U).*cos(V/2).^2;
z = 1-exp(U/(3*pi))-sin(V)+exp(U/(6*pi)).*sin(V);
surf(x,y,z)
grid on
view(160,10)
xlabel('x'); ylabel('y'); zlabel('z')
```



Նկ. 35

5.3. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Կառուցեք հետևյալ եռաչափ գրաֆիկները.

- $x = e^{-t/20}\cos t$, $y = e^{-t/20}\sin t$, $z = t$, որտեղ $t \in [0;10\pi]$,
- $x = (2 + 4\cos t)\cos t$, $y = (2 + 4\cos t)\sin t$, $z = t^2$, որտեղ $t \in [0;20]$,
- $x = (4 - 0.1t)\sin(0.8t)$, $x = (4 - 0.1t)\cos(0.8t)$, $z = 0.4t^{3/2}$, որտեղ $t \in [0;30]$:

Գրաֆիկները դիտեք տարբեր անկյուններով՝ $\varphi = -10^o$ և $\vartheta = 10^o$, $\varphi = -9^o$ և $\vartheta = 50^o$, $\varphi = 0^o$ և $\vartheta = 90^o$, $\varphi = 90^o$ և $\vartheta = 0^o$:

2. Շրջանաձև աստիճանահարթակը նկարագրվում է հետևյալ օրենքով.

$$x = R\cos\left(2\pi n\frac{t}{h}\right),\ y = R\sin\left(2\pi n\frac{t}{h}\right),\ z = \frac{t}{h},$$

որտեղ h –ը շենքի հարկերի բարձրությունն է, n –ը՝ աստիճանների պտույտների թիվը: Ենթադրենք՝ շենքի հարկերի բարձրությունը 3մ է: Դիտարկեք երկու տարբեր դեպք՝ $R = 1.5$ մ, $n = 3$ և $R = 4$ մ, $n = 2$: Երկու դեպքի համար կառուցեք եռաչափ գրաֆիկները միևնույն պատուհանում:

3. Կառուցեք հետևյալ ֆունկցիաների մակերևույթները.

- $z(x,y) = 3.2^{-1.2\sqrt{x^2+y^2}}\cos x\sin 3y$, եթե $-2 \le x \le 2$, $-2 \le y \le 2$,
- $z(x,y) = \cos(x+y)\sin(3x-y) + \cos(x-y)\sin(x+3y) + 5e^{-(x^2+y^2)/8}$, եթե $-3 \le x \le 3$, $-3 \le y \le 3$,
- $z(x,y) = \frac{x^2}{3} + 2\sin 3y$, եթե $-3 \le x \le 3$, $-3 \le y \le 3$,
- $z(x,y) = 2|x| + 2|y|$, եթե $-2 \le x \le 2$, $-2 \le y \le 2$,
- $z(x,y) = \frac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$, եթե $-10 \le x \le 10$, $-10 \le y \le 10$,
- $z(x,y) = -\frac{x^2}{4} - \frac{y^2}{4}$, եթե $-4 \le x \le 4$, $-4 \le y \le 4$,
- $z(x,y) = -xye^{-2(x^2+y^2)}$, եթե $-\pi \le x \le \pi$, $-\pi \le y \le \pi$,
- $z(x,y) = \sin x\cos y$, եթե $0 \le x \le 2\pi$, $0 \le y \le \pi$,
- $z(x,y) = (y+3)^2 + 2x^2 - x^2y$, եթե $-3 \le x \le 3$, $-3 \le y \le 3$,
- $z(x,y) = \cos(xy)\cos\sqrt{x^2+y^2}$, եթե $-\pi \le x \le \pi$, $-\pi \le y \le \pi$:

4. Կառուցեք հետևյալ ֆունկցիաների կոնտուրային գրաֆիկները.

- $z(x,y)=1.8^{-1.5\sqrt{x^2+y^2}}\sin x\cos 0.5y$, եթե $-3\le x\le 3$, $-3\le y\le 3$,
- $z(x,y)=\sin(x^2-y^2)$, եթե $-\frac{\pi}{2}\le x\le\frac{\pi}{2}$, $-\frac{\pi}{2}\le y\le\frac{\pi}{2}$,
- $z(x,y)=xe^{-(x^2+y^2)}$, եթե $-\pi\le x\le\pi$, $-\pi\le y\le\pi$,
- $z(x,y)=\frac{xy^2}{x^2+y^2}$, եթե $-1\le x\le 3$, $1\le y\le 4$:

5. Կառուցեք հետևյալ նշանավոր մակերևույթները, որոնց հավասարումները տրված են պարամետրական տեսքով.

- **կոնական մակերևույթ**
  $x=R\cos\varphi$, $y=R\sin\varphi$, $z=R$, եթե $0\le R\le 1$, $0\le\varphi\le 2\pi$,
- **գնդային մակերևույթ**
  $x=R\sin\vartheta\cos\varphi$, $y=R\sin\vartheta\sin\varphi$, $z=R\cos\vartheta$,
  եթե $0\le\vartheta\le\pi$, $0\le\varphi\le 2\pi$, $R=5$,
- **էլիպսոիդ**
  $x=a\cos u\sin v$, $y=b\sin u\sin v$, $z=c\cos v$, եթե $0\le u\le 2\pi$, $0\le v\le\pi$:
  Մակերևույթը կառուցեք տարբեր a, b, c գործակիցներով,
- **հիպերբոլոիդ**
  $x=a\,\mathrm{ch}\,u\cos v$, $y=b\,\mathrm{ch}\,u\sin v$, $z=c\,\mathrm{sh}\,u$, եթե $-2\le u\le 2$, $0\le v\le 2\pi$:
  Մակերևույթը կառուցեք տարբեր a, b, c գործակիցներով,
- **հիպերբոլոիդ**
  $x=a\,\mathrm{sh}\,u\cos v$, $y=b\,\mathrm{sh}\,u\sin v$, $z=c\,\mathrm{ch}\,u$, եթե $-2\le u\le 2$, $0\le v\le 2\pi$:
  Մակերևույթը կառուցեք տարբեր a, b, c գործակիցներով,
- **Լիսաժուի մակերևույթ**
  $x=\sin u$, $y=\sin v$, $z=\sin((d-au-bv)/c)$,
  եթե $-\pi\le u\le\pi$, $-\pi\le v\le\pi$, $a=1$, $b=1$, $c=1$, $d=0$,
- **Սթեյների մակերևույթ**
  $x=\sin 2u\cos^2 v$, $y=\sin u\sin 2v$, $z=\cos u\sin 2v$, եթե $0\le u\le\pi$, $-\frac{\pi}{2}\le v\le\frac{\pi}{2}$,
- **փսևդոգունդ**
  $x=\cos u\sin v$, $y=\sin u\sin v$, $z=\cos v+\ln\left(\mathrm{tg}\frac{v}{2}\right)$, եթե $0\le u\le 2\pi$, $0\le v\le\pi$,
- **կատենոիդ**
  $x=a\cos u\,\mathrm{ch}\frac{v}{a}$, $y=a\sin u\,\mathrm{ch}\frac{v}{a}$, $z=v$, եթե $0\le u\le 2\pi$, $-c\le v\le c$, $c>0$:
  Մակերևույթը կառուցեք տարբեր a, c գործակիցներով:

- **տորոիդ**

  $x=(a+b\cos v)\cos u$, $y=(a+b\cos v)\sin u$, $z=b\sin v$, եթե $0\le u\le 2\pi$, $0\le v\le 2\pi$: Մակերևույթը կառուցեք տարբեր a , b գործակիցներով:

- **տիեզերանավ**

  $x=-u+\dfrac{2w^2\,\mathrm{ch}(au)\,\mathrm{sh}(au)}{d}$, $y=2w\,\mathrm{ch}(au)\dfrac{-w\cos v\cos(wv)-\sin v\sin(wv)}{d}$,

  $z=2w\,\mathrm{ch}(au)\dfrac{-w\sin v\cos(wv)+\cos v\sin(wv)}{d}$, եթե $-13.4\le u\le 13.4$, $-37.2\le v\le 37.2$,

  $a=0.4$, $w=\sqrt{1-a^2}$, $d=a\left((w\,\mathrm{ch}(au))^2+(a\sin(wv))^2\right)$:

6. Գլանային կոորդինատական համակարգի (ρ,φ,z) կոորդինատները դեկարտյան (x,y,z) կոորդինատների հետ կապված են $x=\rho\cos\varphi$, $y=\rho\sin\varphi$, $z=z$ առնչություններով, որտեղ $0\le\rho\le+\infty$, $0\le\varphi\le 2\pi$, $-\infty\le z\le+\infty$: Միևնույն կոորդինատական առանցքի վրա կառուցեք $\rho=const$, $\varphi=const$, $z=const$ գլանային կոորդինատական համակարգի կոորդինատային մակերևույթները ($\rho=const$ մակերևույթները z առանցքին կոաքսիալ գլաններ են, $\varphi=const$ մակերևույթները՝ z առանցքով սահմանափակված կիսահարթություններ, իսկ $z=const$ մակերևույթները՝ z առանցքին ուղղահայաց հարթություններ):

7. Գնդային կոորդինատական համակարգի (r,ϑ,φ) կոորդինատները դեկարտյան (x,y,z) կոորդինատների հետ կապված են $x=r\sin\vartheta\cos\varphi$, $y=r\sin\vartheta\sin\varphi$, $z=r\cos\vartheta$ առնչություններով, որտեղ $0\le r\le+\infty$, $0\le\vartheta\le\pi$, $0\le\varphi\le 2\pi$: Միևնույն կոորդինատական առանցքի վրա կառուցեք $r=const$, $\vartheta=const$, $\varphi=const$ գնդային կոորդինատական համակարգի կոորդինատային մակերևույթները ($r=const$ մակերևույթները r շառավղով գնդային մակերևույթներ են, որոնց կենտրոնն ընկած է կոորդինատների սկզբնակետում, $\vartheta=const$ մակերևույթները՝ z առանցքով կոներ, իսկ $\varphi=const$ մակերևույթները՝ z առանցքով սահմանափակված կիսաշրջանագծեր):

6. ԾՐԱԳՐԱՎՈՐՈՒՄԸ MATLAB–ՈՒՄ

Ծրագիր գրելիս **MatLab**–ի ներդրված ֆունկցիաները բավականին պարզեցնում են օգտագործողի աշխատանքը, քանի որ դրանց՝ նախօրոք սահմանված չլինելու դեպքում, վերջինս ստիպված էր լինելու ինքը սահմանել անհրաժեշտ ֆունկցիան, ինչը հաճախ կլիներ բարդ և ժամանակատար: Միևնույն ժամանակ, չնայած բազմաթիվ ներդրված ֆունկցիաների առկայությանը, ավելի բարդ ծրագրեր գրելիս, միևնույնն է, հարկ է լինում դիմել ծրագրավորման տարրերին: Ծրագրավորելը հնարավորություն է տալիս օգտվելու պայմանի և ցիկլի շատ կարևոր օպերատորներից, կառուցելու տվյալ խնդրին բնորոշ սեփական ֆունկցիաներ, որոնք ներդրված ֆունկցիաների նման կարելի է օգտագործել այլ խնդիրներում և այլն: Օգտագործելով **MatLab**–ի այս հնարավորությունները՝ ծրագրավորողը հնարավորություն է ստանում գրել ավելի բարդ և ավելի ճկուն ծրագրեր:

6.1. ՍԿՐԻՊՏԱՅԻՆ ՖԱՅԼԵՐ (M–ՖԱՅԼԵՐ)

Մինչ այժմ բոլոր ծրագրերը գրում էինք **MatLab**–ի հրամանի պատուհանի հրամանի տողում՝ հրավերի (>>) նշանից հետո: Հրամանի տողում ծրագրերը գրելը հեշտ է և արդյունավետ, երբ լուծում ենք պարզ խնդիրներ: Սակայն, երբ ծրագրում հրամանների թիվն ավելանում է, հրամանի պատուհանում ծրագրեր գրելը դառնում է անհարմար: Պատճառն այն է, որ հրամանի պատուհանում **MatLab**–ով գրված ծրագրերը պահպանելու և հետագայում նորից կանչելու հնարավորություն նախատեսված չէ: Բացի այդ, ամեն անգամ հրամանն իրականացնելիս կատարվում է միայն ամենավերջին գրված հրամանը, իսկ դրան նախորդող բոլոր հրամանները մնում են անփոփոխ: Եթե անհրաժեշտ է փոփոխել նախորդող որևէ հրաման, ապա անհրաժեշտ է այդ և դրան հաջորդող բոլոր հրամանները հերթով մեկ անգամ ևս իրականացնել:

Այս բարդությունը հաղթահարելու համար **MatLab**–ը հնարավորություն է տալիս ծրագիրը գրել առանձին տեքստային ֆայլում, այնուհետև **MatLab**–ի հրամանի տողում որպես հրաման պարզապես գրել այդ ֆայլի անունը: Արդյունքում տեքստային ֆայլում գրված հրամանները հերթով կիրականան ճիշտ այնպես, ինչպես եթե դրանք գրեինք հրամանի պատուհանում: Տեքստային ֆայլի առավելությունն այն է, որ այն կարելի է պահպանել և հետագայում նորից դիմել իրեն: Բացի այդ, կարելի է այդ ֆայլի մեջ հեշտությամբ փոփոխություններ կատարել և նորից աշխատեցնել:

Այսպիսի տեքստային ֆայլերը կոչվում են սկրիպտային ֆայլեր կամ M–ֆայլեր: Վերջին անվանումը կապված է այդ ֆայլերի *.m ընդլայնված անվան հետ (օրինակ, program1.m): Քանի որ M–ֆայլերը տեքստային ֆայլեր են, ապա դրանք

կարելի է ստեղծել և խմբագրել կամայական տեքստային խմբագրիչում (օրինակ՝ **Notepad**–ում, **WordPad**–ում, **UltraEdit**–ում և այլն): Սակայն **MatLab**–ն ունի իր սեփական տեքստային խմբագրիչը, որը կարելի է բացել՝ **MatLab**–ի գործիքների գոտու **HOME** բաժնից ընտրելով **New**, այնուհետև **Script** հրամանը կամ ստեղնաշարի վրա սեղմելով **Ctrl+N**: Այն կարելի է բացել նաև **MatLab**–ի հրամանի տողում պարզապես գրելով **edit** հրամանը:

M–ֆայլերը ստեղծելուց և պահպանելուց (**save**) հետո այն կարելի է աշխատեցնել՝ **MatLab**–ի խմբագրիչում սեղմելով  (**Run)** կոճակը կամ, ինչպես վերևում ասվեց, **MatLab**–ի հրամանի տողում որպես հրաման գրելով ֆայլի անունը:

M–ֆայլում գրվող ծրագրում կարելի է օգտվել բոլոր այն փոփոխականներից, որոնք տվյալ պահին սահմանված են **MatLab**–ում, այսինքն՝ գրված են **MatLab**–ի աշխատանքային պատուհանում (**Workspace**): Բացի այդ, M–ֆայլն աշխատեցնելուց հետո դրա ներսում սահմանված բոլոր փոփոխականները ևս կհայտնվեն **MatLab**–ի աշխատանքային պատուհանում, այսինքն՝ դրանք հնարավոր կլինեն օգտագործել **MatLab**–ի այլ ծրագրերում ևս:

M–ֆայլն աշխատեցնելիս նրա մեջ գրված բուն հրամանները չեն երևում հրամանի պատուհանում՝ երևում են միայն այդ հրամանների արդյունքները: Եթե ցանկանում ենք հրամանի պատուհանում բացի արդյունքներից տեսնել նաև հրամանները, կարելի է հրամանի տողում կամ M–ֆայլում գրել **echo on** հրամանը: Այս հրամանն իրականանալուց հետո հետագա բոլոր հրամանները կերևան հրամանի պատուհանում այնքան ժամանակ, քանի դեռ չի գրվել **echo off** հրամանը:

MatLab–ում սահմանված են մի շարք ներդրված ֆունկցիաներ և հրամաններ, որոնք հաճախ են օգտագործվում ծրագրեր գրելիս: Չնայած այս ֆունկցիաները կարելի է օգտագործել նաև հրամանի տողում փոքր ծրագրեր գրելիս, սակայն դրանց կիրառումն այդ դեպքում այնքան էլ իմաստավորված չէ, փոխարենը դրանք շատ հարմար են M–ֆայլերում: Դրանց նկարագրությունը բերված է հետևյալ աղյուսակում:

Ֆունկցիան	**Նկարագրությունը**
disp(A)	հրամանի պատուհանում առանց *A* փոփոխականի անունը գրելու՝ ցույց է տալիս դրա արժեքը
disp(′text′)	հրամանի պատուհանում ցույց է տալիս արգումենտում գրված տեքստը
error(′text′)	հրամանի պատուհանում սխալի տեսքով ցույց է տալիս արգումենտում գրված տեքստը, և ծրագիրը դադարեցնում է աշխատանքը՝ չի իրականացնում այս ֆունկցիայից հետո գրված մնացած հրամանները

warning(′text′)	հրամանի պատուհանում զգուշացման տեսքով ցույց է տալիս արգումենտում գրված տեքստը, սակայն, ի տարբերություն error ֆունկցիայի, ծրագիրը չի դադարեցնում աշխատանքը
echo on	թույլ է տալիս հրամանի պատուհանում տեսնել M–ֆայլում գրված ծրագրի հրամանները
echo off	հրամանի պատուհանում երևում են միայն M–ֆայլում գրված ծրագրի հրամանների արդյունքները՝ առանց բուն ծրագիրը ցույց տալու (default)
input(′prompt′)	ծրագիրը դադարեցնում է աշխատանքը և չի շարունակում այնքան ժամանակ, քանի դեռ օգտագործողը չի ներմուծել որևէ տվյալ
pause	ծրագիրը դադարեցնում է աշխատանքն այնքան ժամանակ, քանի դեռ օգտագործողը չի սեղմել ստեղնաշարի որևէ ստեղն
pause(n)	ծրագիրը *n* վայրկյան դադարեցնում է աշխատանքը

Այստեղ ներկայացված **input** ֆունկցիան օգտագործվում է ծրագրի իրականացման ընթացքում փոփոխականներին արժեքներ վերագրելու համար: Օրինակ, եթե M–ֆայլի որևէ տողում հանդիպի

```
x = input('Enter the value of variable x: ');
```

հրամանը, ապա **MatLab**–ի հրամանի տողում կհայտնվի *Enter the value of variable x:* տեքստը, և ծրագիրը չի շարունակի աշխատանքն այնքան ժամանակ, քանի օգտագործողը չի ներմուծել որևէ թիվ: Ներմուծելուց և **Enter** սեղմելուց հետո այդ թիվը կվերագրվի *x* փոփոխականին, իսկ ծրագիրը կշարունակի աշխատանքը: Նշենք, որ այս օրինակում *x*–ին արժեք վերագրելուց հետո այդ արժեքը չի երևա ծրագրի հաջորդ տողում, քանի որ հրամանից հետո դրվել է կետ–ստորակետ:

M–ֆայլի անունն ընտրելիս պետք է հետևել նույն կանոններին, ինչ փոփոխականներ սահմանելիս, այսինքն՝ այն պետք է բաղկացած լինի լատիներեն մեծատառերից ու փոքրատառերից, թվերից ու ընդգծման գծիկից, ինչպես նաև անպայման պետք է սկսվի տառով:

Եթե M–ֆայլը պահպանենք որևէ անունով, և նույն անունով փոփոխական ևս սահմանված լինի աշխատանքային միջավայրում կամ հենց նույն M–ֆայլում, ապա դա կբերի սխալների: Բանն այն է, որ M–ֆայլն աշխատեցնելու համար հրամանի տողում պետք է գրել այդ ֆայլի անունը որպես հրաման: Եթե մինչ այդ որևէ այլ ծրագրով սահմանվել է նույն անունով փոփոխական, ապա հրամանի տողում այդ հրամանը գրելիս կվերադարձվի նախկինում սահմանված փոփոխականի արժեքը: Եթե աշխատանքային միջավայրում նախկինում սահմանված չի եղել այդ անունով փոփոխական, բայց այդպիսին կա M–ֆայլի ծրագրի մեջ, ապա այն առաջին անգամ աշխատեցնելիս ծրագիրը ճիշտ կաշխատի, սակայն այդ փոփոխականը կհայտնվի աշխատանքային պատուհանում, և երկրորդ անգամ աշ-

խատեցնելու փորձ անելիս կաշխատի ոչ թե M–ֆայլը, այլ կվերադարձվի այդ փոփոխականի արժեքը: Իհարկե, կարելի է յուրաքանչյուր անգամ M–ֆայլն աշխատացնելուց առաջ մաքրել աշխատանքային պատուհանը **clear** հրամանների օգնությամբ, սակայն ավելի լավ է խուսափել նույն անուններով ֆայլ և փոփոխական օգտագործելուց:

Խորհուրդ չի տրվում նաև որպես M–ֆայլի անուն օգտագործել **MatLab**–ի հրամանների կամ ֆունկցիաների անունները: Կարելի է նախապես ստուգել՝ արդյոք այդպիսի անունով հրաման կամ ֆունկցիա կա սահմանված **MatLab**–ում՝ հրամանի տողում գրելով **which** հրամանը, որին պետք է հետևի այն անունը, որը ցանկանում ենք ստուգել (օրինակ՝ which program1):

Ընդհանրապես, ծրագրեր գրելիս (հրամանի տողում կամ M–ֆայլում) միշտ պետք է հաշվի առնել, որ ծրագրում կարող են լինել սխալներ: Սխալները բաժանվում են մի քանի խմբի:

Ամենահաճախ հանդիպող սխալը ուղղագրական սխալն է (**syntax error**): Հնարավոր է, որ ծրագրավորողը սխալ գրած լինի որևէ ֆունկցիայի անուն, արտահայտություններում բաց թողնի փակագծեր, սխալ անունով փոփոխական սահմանի, բաց թողնի որևէ օպերատոր և այլն: Այս սխալներից յուրաքանչյուրի դեպքում **MatLab**–ը վերադարձնում է տարբեր հաղորդագրություններ, և ծրագիրը դադարեցնում է աշխատանքը: Հետևյալ օրինակներում բերված են տարբեր ուղղագրական սխալներով հրամաններ, և դրանցից յուրաքանչյուրի համար բերված հաղորդագրությունները:

Օրինակներ.

- Պակասում է փակող փակագիծը:

```
>> 7*(8+6
??? 7*(8+6
          |
Error:  Expression  or  statement  is  incorrect--possibly
unbalanced (, {, or [.
```

- Պակասում է բազմապատկման օպերատորը:

```
>> 9(6-7)
??? 9(6-7)
     |
Error: Unbalanced or unexpected parenthesis or bracket.
```

- **MatLab**–ի կանոններին չբավարարող անունով փոփոխական է սահմանվել:

```
>> 8a = 3*(6+1)
??? 8a = 3*(6+1)
     |
Error: Unexpected MATLAB expression.
```

- **sqrt** ֆունկցիայի անունը սխալ է գրվել:
```
>> x = srqt(9)
??? Undefined function or method 'srqt' for input arguments
of type 'double'.
```

Երկրորդ տիպի սխալները ծրագրի աշխատանքի ընթացքում ի հայտ եկող սխալներն են (**Run–Time Error**): Դրանք առաջանում են, երբ որևէ գործողության արդյունքը ստացվում է **NaN**, **Inf**, դատարկ զանգված և այլն:

Եվ վերջապես, շատ հաճախ են հանդիպում, այսպես կոչված, տրամաբանական սխալները (**Logic Error**): Այս սխալների դեպքում ծրագիրը նորմալ աշխատում է, բայց վերադարձնում է սխալ արդյունք: Դժվարը տրամաբանական սխալները հայտնաբերելը և ուղղելն է: Սխալը հայտնաբերելու համար պետք է կատարել ծրագրի տեստավորում: Օրինակ՝ ծրագիրը կարելի է տրոհել փոքր հատվածների, հերթով աշխատեցնել, ստուգել ծրագրի այդ հատվածների արդյունքները և աստիճանաբար հասնել ծրագրի այն հատվածին, որտեղ առաջացել է սխալը: Տեստավորման ժամանակ ճիշտ կլինի հանել հրամանների վերջում դրված կետ–ստորակետները, որպեսզի ծրագրի յուրաքանչյուր հրամանի արդյունքը տեսանելի լինի:

M–ֆայլն աշխատեցնելիս պետք է ուշադիր լինել, որ անհրաժեշտ սկրիպտային ֆայլը պարունակվի **MatLab**–ի ընթացիկ թղթապանակում (**Current directory**): Որպեսզի ընտրենք այն թղթապանակը (**Folder**), որը պարունակում է անհրաժեշտ ֆայլը, կարելի է օգտվել **MatLab**–ի **cd** հրամանից, որին պետք է հետևի անհրաժեշտ թղթապանակի ճանապարհը (օրինակ՝ cd D:\Work): Կարելի է նաև **MatLab**–ի մենյուի ներքևի տողից ընտրել անհրաժեշտ ճանապարհը՝ օգտվելով **Current Directory** տողից (նկ. 36): Ուշադրություն դարձրեք, որ **MatLab**–ի ընթացիկ թղթապանակի պատուհանում (**Current Directory**) երևում է ընտրված թղթապանակի պարունակությունը:

Նկ. 36

6.2. ՖԱՅԼ–ՖՈՒՆԿՑԻԱՆԵՐ (M–ՖՈՒՆԿՑԻԱՆԵՐ)

Ֆունկցիայի գաղափարը **MatLab**–ում նույնն է, ինչ ծրագրավորման այլ լեզուներում: Դրանք ծրագրի կտորներ են (ենթածրագրեր), որոնք թույլ են տալիս մեծ ծրագիրը տրոհել առանձին մասերի, որոնցից յուրաքանչյուրն ունակ է լուծելու որևէ կոնկրետ խնդիր: Մեծ ծրագիրը ենթածրագրերի բաժանելը հարմար է, քանի

որ հիմնական ծրագիրը դառնում է ավելի ընթեռնելի, իսկ կրկնվող հատվածները ամեն անգամ գրելու փոխարեն գրում են մեկ անգամ, պահպանում առանձին ֆայլում և ծրագրի մեջ ամեն անգամ կանչում են պարզապես այդ ֆայլի անունը:

Ֆունկցիաները սովորաբար վերցնում են որևէ տվյալ կամ տվյալներ, դրանց հետ կատարում մի շարք գործողություններ և վերադարձնում արդյունք: Մեկ անգամ ֆունկցիան սահմանելով՝ դրանք կարելի է անընդհատ օգտագործել, ինչպես ներդրված ֆունկցիաները:

Օգտագործողի կողմից գրվող ֆունկցիան նման է M–ֆայլին, քանի որ երկուսն էլ տեքստային ֆայլեր են և ունեն *.m ընդլայնումը: Այդ պատճառով դրանք երբեմն կոչվում են նաև M–ֆունկցիաներ: Ֆունկցիաների ներսում սահմանված փոփոխականները լոկալ են, այսինքն՝ դրանք հասանելի են միայն տվյալ ֆունկցիայի ներսում:

M–ֆունկցիա գրելիս պետք է հետևել հետևյալ կանոններին:

- **Ֆունկցիայի սահմանումը**: Յուրաքանչյուր ֆայլ–ֆունկցիա պետք է սկսվի այսպիսի տողով.

  ```
  function [output variables] = function_name(input variables)
  ```

  Այսպիսով՝ այն սկսվում է **function** բառով, այնուհետև քառակուսի փակագծերում գրվում են ելքային փոփոխականները (ելքային արգումենտները), վերագրման նշանին հետևում է ֆունկցիայի անունը և, կլոր փակագծերում՝ մուտքային փոփոխականները (մուտքային արգումենտները): Եթե ելքային արգումենտը մեկ հատ է, քառակուսի փակագծերը կարելի է բաց թողնել:

  Օրինակ.
  ```
  function C = F2D(F);
  ```

  Այս ֆունկցիայում *C*–ն ելքային արգումենտն է, *F*2*D*–ն՝ ֆունկցիայի անունը, *F*–ը՝ մուտքային արգումենտը:

  Չնայած պարտադիր չէ, սակայն ավելի ճիշտ է ֆունկցիան ավարտել **end** բառով:

- **Ֆունկցիայի պահպանումը**

  Ֆունկցիան պահպանելիս ֆունկցիայի ֆայլի անունը պետք է անպայման լինի նույնը, ինչ ֆունկցիայի անունը: Նախորդ օրինակի ֆունկցիայի ֆայլը պետք է անպայման պահպանել *F*2*D* անունով:

- **Մեկնաբանություններ**

  Սովորաբար ծրագրի մեջ որևէ ֆունկցիա գրելուց առաջ կարիք է զգացվում տեղեկություն ստանալ, թե այդ ֆունկցիան ինչ գործողություն է կատարում, քանի մուտքային արգումենտ է պահանջում և քանի ելքային արգումենտ կարող է վերադարձնել: Դա իմանալու ամենահարմար տարբերակը **MatLab**–ի հրամանի տողում **help** հրամանը գրելն է, որին հետևում է ֆունկցիայի անունը: Օրինակ, եթե ցանկանում ենք տեղեկություն ստանալ ներդրված **max** ֆունկցիայի մասին, հրամանի տողում գրում ենք **help max**: Նույնը կարելի է անել օգտագործողի սահմանած ֆունկցիաների հետ: Դրա համար ֆունկցիայի հայտարարության տողից հետո լավ սովորություն է մի քանի տողով մեկնաբանություն–բացատրություն գրել սահմանվող ֆունկցիայի մասին (% նշանից հետո): Հետագայում **help** հրամանի օգնությամբ այդ մեկնաբանությունը որպես տեղեկություն կարտածվի հրամանի պատուհանում:

- **Վերադարձվող արժեքները**

  Արժեքի տեսքով ֆունկցիաները վերադարձնում են միայն ելքային արգումենտը (կամ արգումենտները): Հետևաբար, ֆունկցիայի մարմնում ելքային փոփոխականին պետք է անպայման վերագրման գործողություն կատարվի: Իհարկե, պետք է նկատի ունենալ նաև, որ ֆունկցիան կարող է որևէ ելքային արգումենտ չունենալ: Այդ դեպքում ֆունկցիայի հայտարարության մեջ ելքային արգումենտի հատվածում ոչինչ չի գրվում: Օրինակ, ելքային արգումենտ կարող է չունենալ այնպիսի ֆունկցիան, որի նպատակն է պարզապես որևէ կախվածության գրաֆիկ կառուցել:

- **Լոկալ փոփոխականներ**

  Ֆունկցիան կանչելիս **MatLab**–ի աշխատանքային միջավայրի պատուհանում երևում են միայն ֆունկցիայի մուտքային և ելքային արգումենտները: Ֆունկցիայի մարմնում կարելի է սահմանել ցանկացած քանակությամբ միջանկյալ փոփոխականներ, սակայն դրանք հասանելի չեն լինի **MatLab**–ի աշխատանքային պատուհանում: Այսպիսի փոփոխականները կոչվում են լոկալ փոփոխականներ: Այսպիսով, յուրաքանչյուր ֆունկցիա ունի իր սեփական աշխատանքային միջավայրը, որն անկախ է **MatLab**–ի աշխատանքային միջավայրից:

- **Գլոբալ փոփոխականներ**

  Չնայած, որ բացի ֆունկցիայի մուտքային և ելքային արգումենտներից, ֆունկցիայի ներսում մնացած բոլոր փոփոխականները լոկալ են և հա-

սանելի չեն **MatLab**–ի աշխատանքային միջավայրին, **MatLab**–ը հնարավորություն է տալիս այդ փոփոխականները սահմանել գլոբալ, այսինքն՝ այնպես, որ դրանք աշխատանքային միջավայրում ևս հասանելի լինեն: Այդպես սահմանված գլոբալ փոփոխականները կարելի է օգտագործել նաև **MatLab**–ի այլ ծրագրերում և ֆունկցիաներում: Որպեսզի ֆունկցիայի մարմնում սահմանված որևէ փոփոխական հայտարարենք գլոբալ, անհրաժեշտ է նախքան դրան որևէ արժեք վերագրելը ֆունկցիայի մեջ գրել **global** բառը, որին կհետևի այդ փոփոխականի անունը կամ, մի քանի փոփոխական հայտարարելու դեպքում՝ դրանց անունները՝ ստորակետներով անջատված: Օրինակ, ֆունկցիայի մարմնում fg անունով փոփոխականը գլոբալ հայտարարելու համար նախքան դրան արժեք վերագրելը պետք է գրել **global fg** հրամանը: Գլոբալ փոփոխական հայտարարելուց հետո այն կարելի է օգտագործել ցանկացած այլ ծրագրում, **MatLab**–ի հրամանի պատուհանում կամ ուրիշ ֆունկցիայում: Միայն թե այստեղ ևս, նախքան օգտագործելը պետք է գրել **global fg** հրամանը: Լավ կլինի, որ այս հրամանը գրվի **MatLab**–ի ծրագրերի և ֆունկցիաների սկզբում, որպեսզի տեսանելի լինի, թե որ փոփոխականն է հայտարարված գլոբալ: Տարբեր ծրագրերում և ֆունկցիաներում գլոբալ հայտարարված նույն փոփոխականի արժեքը դրանցից յուրաքանչյուրում կարելի է փոխել:

- **Մի քանի ելքային արգումենտներ**

  Եթե ֆունկցիան վերադարձնում է ոչ թե մեկ, այլ մի քանի ելքային արգումենտ, ապա դրանք պետք է գրել քառակուսի փակագծերում և իրարից անջատել ստորակետերով:

  Օրինակ.

```
function [x, y, z] = coordinate(r)
```

- **Մի քանի մուտքային արգումենտներ**

  Եթե ֆունկցիան ստանում է ոչ թե մեկ, այլ մի քանի մուտքային արգու–մենտներ, ապա դրանք գրվում են կլոր փակագծերում և իրարից ան–ջատվում ստորակետներով:

  Օրինակ.

```
function F1 = my_function(a, b, c)
```

- **Կետ–ստորակետներ**

  Կետ–ստորակետի նպատակը ֆունկցիայում նույնն է, ինչ **MatLab**–ի ցանկացած հրամանի դեպքում: Այն թույլ չի տալիս, որ տվյալ հրամանի արդյունքը երևա **MatLab**–ի հրամանի պատուհանում:

M–ֆունկցիա ստեղծելու համար կարելի է բացել դատարկ M–ֆայլ, ինչպես նկարագրված էր նախորդ բաժնում, դրա մեջ գրել ֆունկցիայի հայտարարության տողը, մեկնաբանությունները և ֆունկցիայի մարմինը, այնուհետև պահպանել սահմանված ֆունկցիայի անունով:

MatLab–ում նախատեսված է M–ֆունկցիա սահմանելու ավելի հարմար եղանակ: Դրա համար անհրաժեշտ է **MatLab**–ի գործիքների գոտու **HOME** բաժնից ընտրել **New**, այնուհետև **Function** հրամանը: Բացված ֆայլն արդեն պարունակում է ֆունկցիայի հայտարարության տողն ու մեկնաբանության բաժինը հետևյալ տեսքով.

```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

end
```

Այստեղ մնում է միայն ավելացնել ֆունկցիայի մարմինը: Ավելին, երբ ցանկանում ենք պահպանել այս եղանակով բացված M–ֆունկցիան, **MatLab**–ն ինքնուրույն սահմանված ֆունկցիայի անունը գրում է որպես ֆայլի անուն: Այսինքն, սխալ անունով պահպանելու հավանականությունը փոքրանում է:

Օրինակ.

Որպես օրինակ, ստեղծենք *C2FK* անունով ֆունկցիա, որը որպես մուտքային արժեք կստանա Ցելսիուսով արտահայտված ջերմաստիճանը, իսկ որպես ելքային արժեքներ կվերադարձնի դրան համապատասխան Ֆահրենհեյթով և Կելվինով արտահայտված ջերմաստիճանները: Ինչպես հայտնի է Ֆարենհեյթով և Կելվինով արտահայտված ջերմաստիճանները Ցելսիուսով արտահայտված ջերմաստիճանի հետ կապված են համապատասխանաբար $^{o}F = \frac{9}{5}.^{o}C + 32$ և $^{o}K = ^{o}C + 273.15$ առնչություններով:

Նախքան ֆունկցիան սահմանելը, ստուգենք՝ արդյոք **MatLab**–ում նախապես սահմանված չի *C2FK* անունով ֆունկցիա կամ հրաման: Դրա համար **MatLab**–ի հրամանի տողում գրենք **which C2FK** հրամանը:

```
>> which C2FK
   'C2FK' not found.
```

Ինչպես տեսնում ենք, հրամանի արդյունքում **MatLab**–ը վերադարձրեց *'C2FK' not found.* հաղորդագրությունը: Դա նշանակում է, որ այդ անունով հրաման կամ ֆունկցիա նախկինում սահմանված չի եղել, և մենք հանգիստ կարող ենք այն օգտագործել:

Այժմ վերևում նշված երկու եղանակներից որևէ մեկով ստեղծենք M–ֆունկցիայի ֆայլ և այդ ֆայլում գրենք հետևյալ ծրագիրը.

```
function [F, K] = C2FK(C)
    % C2FK gets its input as a temperature in Celsius and returns
    % corresponding temperatures in Fahrenheit and Kelvin.
    % It gets one input argument and returns 2 output arguments

    F = 9/5*C + 32;
    K = C + 273.15;
end
```

Այս ֆունկցիայի առաջին տողը ֆունկցիայի հայտարարության տողն է: Ինչպես տեսնում ենք, այն սահմանվել է *C2FK* անունով, ունի *C* անունով մեկ մուտքային արգումենտ, որը կհամապատասխանի Ցելսիուսով արտահայտված ջերմաստիճանին, և երկու ելքային արգումենտ՝ *F* և *K* անուններով, որոնք կհամապատասխանեն Ֆահրեհեյթով և Կելվինով արտահայտված ջերմաստիճաններին: Ֆունկցիայի հաջորդ երեք տողերը մեկնաբանություններ են, որոնք նկարագրում են ֆունկցիայի աշխատանքն ու տեղեկություն տալիս մուտքային և ելքային արգումենտների մասին: Այնուհետև, *F* և *K* փոփոխականներին վերագրվել են այն բանաձևերը, որոնցով իրենք կապված են Ցելսիուսով արտահայտված ջերմաստիճանի հետ: Վերջապես, գրվել է **end**՝ ցույց տալով, որ ֆունկցիայի հայտարարությունն ավարտված է:

Ֆունկցիան սահմանելուց հետո ֆայլը պետք է պահպանել *C2FK* անունով: Պահպանելուց հետո ստուգենք՝ ֆունկցիան իրոք սահմանված է, թե ոչ: Դարձյալ գրենք **which C2FK** հրամանը:

```
>> which C2FK
   D:\Work\C2FK.m
```

Ինչպես տեսնում ենք այն արդեն գոյություն ունի և գտնվում է D:\Work թղթապանակում: Օգտվելով **help** հրամանից՝ կարող ենք տեղեկություն ստանալ սահմանված ֆունկցիայի մասին.

```
>> help C2FK
    C2FK gets its input as a temperature in Celsius and returns
    corresponding temperatures in Fahrenheit and Kelvin.
    It gets one input argument and returns 2 output arguments
```

Հրամանի արդյունքում վերադարձվեց այն մեկնաբանությունը, որը գրվել էր ֆունկցիայի հայտարարության տողից հետո:

Ֆունկցիան սահմանելուց և պահպանելուց հետո այն կարելի է հետագայում օգտագործել ճիշտ այնպես, ինչպես ներդրված ցանկացած ֆունկցիա: Օրինակ, հրամանի տողում գրենք.

```
>> [F, K] = C2FK(40)
   F = 104
   K = 313.1500
```

Ինչպես տեսնում ենք 40^0C–ին համապատասխանում են 104^0F և 313.15^0K ջերմաստիճանները: Որպես ֆունկցիայի արգումենտ կարելի է օգտագործել նաև վեկտոր-

ներ: Ընդհանրապես պետք է ուշադիր լինել, և եթե ցանկանում ենք, որ ֆունկցիան ճիշտ աշխատի նաև վեկտորների հետ, պետք է անհրաժեշտության դեպքում օգտվել անդամ առ անդամ գործողություններից: Այս խնդրում դրա անհրաժեշտությունը չկար, քանի որ մուտքային արգումենտն ընդամենը բազմապատկվում էր սկալյարով:

Ֆունկցիաները **MatLab**–ում կարող են վերադարձնել մի քանի արգումենտ, բայց երբ այդպիսի ֆունկցիան վերագրում ենք միայն մեկ փոփոխականի, ապա այն վերադարձնում է միայն առաջին ելքային արգումենտը: Օրինակ, եթե հրամանի տողում պարզապես գրեինք **C2FK(40)** հրամանը, ապա ֆունկցիան կվերադարձներ միայն Ֆահրենհեյթով արտահայտված ջերմաստիճանը: Այսպիսի դեպքի հանդիպել ենք, օրինակ, վեկտորների և մատրիցների համար **max** կամ **min** ֆունկցիաները նկարագրելիս:

MatLab–ը հնարավորություն է տալիս նաև M–ֆունկցիաներն օգտագործել որպես որոշ հրամանների արգումենտներ: Մասնավորապես, **MatLab**–ում սահմանված է ներդրված **fplot** ֆունկցիան, որը բավական հարմար է ինչպես օգտագործողի կողմից սահմանված M–ֆունկցիաների, այնպես էլ ներդրված ֆունկցիաների գրաֆիկները կառուցելու համար: Ի տարբերություն 3–րդ գլխում սահմանված **plot** ֆունկցիայի, որը գրաֆիկը կառուցում էր երեք քայլով (մուտքային արգումենտի վեկտորի սահմանում, ֆունկցիայի սահմանում և **plot** ֆունկցիայի կանչ), **fplot** ֆունկցիան թույլ է տալիս կառուցել ֆունկցիայի գրաֆիկն ընդամենը մեկ հրամանով: Այն որպես առաջին արգումենտ ստանում է ֆունկցիայի անունը՝ գրված ապոստրոֆների մեջ (կամ առանց ապոստրոֆի՝ այս դեպքում ֆունկցիայի անունը գրվում է @ սիմվոլից հետո), իսկ որպես երկրորդ արգումենտ՝ քառակուսի փակագծերում, բացատներով անջատած, գրվում է այն տիրույթը, որում ցանկանում ենք կառուցել գրաֆիկը: Կարելի է գրել ինչպես միայն աբսցիսների տիրույթը, այնպես էլ միաժամանակ աբսցիսների և օրդինատների տիրույթները: Որպես երրորդ ոչ պարտադիր արգումենտ կարելի է գրել գրաֆիկի հատկությունները (գույնը, գծի ձևը և մարկերները).

Օրինակներ.

```
>> fplot('sin', [-2*pi 2*pi])
```

Այս օրինակում գրված հրամանով կառուցվում է **sin** ֆունկցիայի գրաֆիկը $[-2\pi; 2\pi]$ տիրույթում: Գրաֆիկը կառուցվում է հոծ, կապույտ գծով:

```
>> fplot(@cos, [-4*pi 4*pi -1.5 1.5])
```

Այս օրինակում գրված հրամանով կառուցվում է **cos** ֆունկցիայի գրաֆիկը $x \in [-4\pi; 4\pi]$ և $y \in [-1.5; 1.5]$ տիրույթում: Գրաֆիկը կառուցվում է հոծ, կապույտ գծով:

```
>> fplot(@C2FK, [-40 40])
>> fplot('C2FK', [-40 40])
```

Այս երկու հրամաններն իրար համարժեք են: Դրանց օգնությամբ կառուցվում է վերևում սահմանված $C2FK$ ֆունկցիայի գրաֆիկը՝ Ցելսիուսով արտահայտված -40^oC –ից 40^oC ջերմաստիճանների տիրույթում:

fplot ֆունկցիայի առավելություններից է նաև այն, որ այստեղ կարիք չկա նշելու, թե ինչ քայլով պետք է տրվեն ֆունկցիայի արժեքները: **fplot**–ն ինքն է որոշում ֆունկցիայի վարքագիծը և ըստ դրա յուրաքանչյուր տիրույթում ընտրում օպտիմալ քայլի արժեքը:

6.3. ՀԱՄԵՄԱՏՈՒԹՅԱՆ ԵՎ ՏՐԱՄԱԲԱՆԱԿԱՆ ՕՊԵՐԱՏՈՐՆԵՐ

Մինչև այժմ մեր գրած ծրագրերը հրամանների հաջորդականություններ էին, որոնք իրականանում էին նույն հերթականությամբ, ինչ հերթականությամբ գրված էին ծրագրի մեջ: Սակայն ավելի բարդ խնդիրներ դիտարկելիս կարելի է ավելի ճկուն և հարմար ծրագրեր ստեղծել, եթե ծրագրավորողը հնարավորություն ունենա խախտել հրամանների հերթականությունը՝ դիմելով ծրագրի տարբեր կտորների, կամ էլ ծրագրի որոշ հատվածներ կրկնի մի քանի անգամ: Ինչպես այլ ծրագրավորման լեզուներում, առաջին տիպի խնդիրները լուծելիս օգտվում են պայմանի (**if**) կամ ընտրության (**switch**) օպերատորներից, երկրորդ տիպի խնդիրները լուծելիս՝ ցիկլի (**for** կամ **while**) օպերատորներից:

Այս դեպքերում անհրաժեշտ է ծրագրի մեջ ստուգել որևէ պայման: Պայմանից կախված՝ ծրագիրն ընտրում է հետագա քայլերի հաջորդականությունը, օրինակ, իրականացնի հաջորդ հրամանը, թե բաց թողնի մի քանի հրաման և շարունակի ծրագրի մեկ այլ տողից: Այս ընտրությունը կատարվում է համեմատությունների միջոցով՝ օգտագործելով համեմատության և տրամաբանական օպերատորներ: Նախքան պայմանի և ցիկլի օպերատորներին անցնելը՝ ծանոթանանք այս օպերատորներին:

Համեմատության օպերատորը համեմատում է երկու թիվ և որոշում՝ արդյոք համեմատությունը ճիշտ է, թե սխալ: Օրինակ՝ 5>2 համեմատությունը ճիշտ է, քանի որ 5–ը մեծ է 2–ից, իսկ 7<4 համեմատությունը՝ սխալ, քանի որ 7–ը փոքր չէ 4–ից: Եթե համեմատության արդյունքը ճիշտ է, գործողության արդյունքում **MatLab**–ը վերադարձնում է 1, իսկ սխալի դեպքում՝ 0: Տրամաբանական օպերատորները գործողություններ են կատարում ճիշտ կամ սխալ արտահայտությունների հետ և վերադարձնում ճիշտ (1) կամ սխալ (0):

MatLab–ում սահմանված համեմատության օպերատորները ներկայացված են հետևյալ աղյուսակում.

Օպերատորը	Նկարագրությունը	Օրինակ	Վերադարձրած արժեքը
>	մեծ է	7 > 8	0
<	փոքր է	15 < 24.8	1
>=	մեծ է կամ հավասար	8 >= 8	1
<=	փոքր է կամ հավասար	5 <= 4	0
==	հավասար է	15 == 24	0
~=	հավասար չէ	9 ~= 10	1

Համեմատության օպերատորներն ունեն հետևյալ հատկությունները.

1. Եթե համեմատվում է երկու թիվ, համեմատության արդյունքը ևս թիվ է (1 կամ 0):

   Օրինակ.

```
>> 15 >= 64
   ans = 0
>> a = 5 < 10
   a = 1
```

2. Դրանք կարող են օգտագործվել մաթեմատիկական արտահայտություններում:

   Օրինակ.

```
>> y = (6 < 10) + (7 > 8) + (5 * 3 == 60/4)
   y = 2
```

3. Եթե համեմատվում են նույն չափն ունեցող երկու զանգված (վեկտոր կամ մատրից), ապա համեմատությունը կատարվում է անդամ առ անդամ, այսինքն՝ հերթով համեմատվում են երկու զանգվածների համապատասխան տարրերը: Արդյունքում ստացվում է նույն չափն ունեցող, 1–երից և 0–ներից կազմված մի նոր զանգված, որի յուրաքանչյուր տարրի արժեքը ստացվում է երկու զանգվածների համապատասխան տարրերի համեմատության արդյունքից:

   Օրինակ.

```
>> b = [15 9 6 4 11 7 14];
>> c = [8 20 9 2 19 7 10];
>> d = c >= b
   d = 0     1     1     0     1     1     0
>> b == c
   ans = 0     0     0     0     0     1     0
>> b ~= c
   ans = 1     1     1     1     1     0     1
>> f = b – c
   f = 7   –11    –3     2    –8     0     4
>> f = b – c > 0
   f = 1     0     0     1     0     0     1
```

4. Եթե թիվը համեմատվում է զանգվածի (վեկտորի կամ մատրիցի) հետ, ապա այն հերթով համեմատվում է զանգվածի տարրերի հետ և արդյունքում վերադարձնում զանգվածի չափի, 1–երից և 0–ներից կազմված նոր զանգված, որի յուրաքանչյուր տարրի արժեքը ստացվում է թվի և զանգվածի համապատասխան տարրերի համեմատության արդյունքից:

```
>> a = 5;
>> b = [7 9 –1 0 91 2 4 6];
>> a <= b
   ans = 1     1     0     0     1     0     0     1
```

5. **Վեկտորների հետ համեմատության գործողությունների արդյունքում ստացված, 1–երից և 0–ներից կազմված վեկտորը կոչվում է տրամաբանական վեկտոր**: Այն կարելի է օգտագործել վեկտորների հասցեավորման համար: Եթե տրամաբանական վեկտորն օգտագործվում է որևէ վեկտորի հասցեավորման համար, վերջինիցս առանձնացնում է այն ինդեքսներով տարրերը, որոնց համապատասխան տրամաբանական վեկտորի տարրերը հավասար են 1–ի:

Օրինակ.

```
>> x = [7 –1 9 12 24 0 8 3 –4];
>> y = (x <= 4)
   y =
      0     1     0     0     0     1     0     1     1
>> a = x(y)
    a =
       –1     0     3     –4
```

Անհրաժեշտ է, սակայն, նկատի ունենալ, որ սովորական, 1–երից ու 0–ներից կազմված վեկտորը տրամաբանական վեկտոր չէ, և այն չի կարելի օգտագործել վեկտորների հասցեավորման համար: Իրենք՝ տրամաբանական վեկտորները, կարող են օգտագործվել թվաբանական գործողություններում:

Համեմատության օպերատորները թույլ են տալիս ստուգել պարզ պայմաններ: Օրինակ՝ a > 5 գործողությունը ստուգում է ընդամենը մեկ պայման: Սակայն հաճախ անհրաժեշտ է լինում միաժամանակ ստուգել ոչ թե մեկ, այլ մի քանի պայման, այսինքն՝ կառուցել ավելի բարդ պայմանի գործողություններ: Այսպիսի դեպքերում, բացի համեմատության օպերատորներից, օգտագործվում են տրամաբանական օպերատորները և տրամաբանական ֆունկցիաները:

Տրամաբանական օպերատորները գործողություններ են կատարում տարբեր ճիշտ և սխալ արտահայտությունների հետ և վերադարձնում 1 (ճիշտ) և 0 (սխալ): **MatLab**–ում սահմանված տրամաբանական օպերատորները բերված են հետևյալ աղյուսակում.

Օպերատորը	Անվանումը	Նկարագրությունը	Աղյուսակը
& (օրինակ, A&B)	տրամաբանական ԵՎ (AND)	Ունի երկու օպերանդ: Վերադարձնում է 1, եթե երկու օպերանդն էլ տրամաբանորեն ճիշտ են, հակառակ դեպքում՝ 0:	ճիշտ & ճիշտ = 1 ճիշտ & սխալ = 0 սխալ & ճիշտ = 0 սխալ & սխալ = 0
| (օրինակ, A|B)	տրամաբանական ԿԱՄ (OR)	Ունի երկու օպերանդ: Վերադարձնում է 1, եթե երկու օպերանդներից գոնե մեկը տրամաբանորեն ճիշտ է, հակառակ դեպքում՝ 0:	ճիշտ | ճիշտ = 1 ճիշտ | սխալ = 1 սխալ | ճիշտ = 1 սխալ | սխալ = 0
~ (օրինակ, ~A)	տրամաբանական ՈՉ կամ տրամաբանական ժխտում (NOT)	Ունի մեկ օպերանդ: Վերադարձնում է 1, եթե այդ օպերանդը տրամաբանորեն սխալ է, հակառակ դեպքում՝ 0:	~ճիշտ = 0 ~սխալ = 1

Տրամաբանական օպերատորներն ունեն հետևյալ հատկությունները.

1. Այս օպերատորների օպերանդները թվեր են: Ոչ զրոյական թվերը համարվում են տրամաբանորեն ճիշտ, զրոն համարվում է տրամաբանորեն սխալ:

   Օրինակ.

```
>> 7 & 8
   ans = 1
>> 5 | 0
   ans = 1
>> ~64
   ans = 0
```

2. Համեմատության օպերատորների պես տրամաբանական օպերատորները ևս կարող են օգտագործվել տարբեր թվաբանական արտահայտություններում: Սրանց արդյունքը կարելի է օգտագործել այլ մաթեմատիկական արտահայտություններում, զանգվածների հասցեավորման մեջ և, որ ամենակարևորն է, ծրագրի ընթացքը ղեկավարող հրամաններում (օրինակ՝ **if** պայմանի կամ **while** ցիկլի օպերատորներում՝ որպես պայման):

   Օրինակ.

```
>> x = 15 + 75*((14&0) + (~7) – (84|6))
    x = –60.00
```

3. Տրամաբանական օպերատորների օպերանդները կարող են լինել ոչ միայն սկալյար թվեր, այլև զանգվածներ: Եթե & և | օպերատորների երկու օպերանդներն էլ թվեր են, արդյունքը ևս թիվ է (1 կամ 0): Եթե երկու օպերանդներն էլ հավասար չափի զանգվածներ են, ապա տրամաբանական գործողությունները կատարվում են անդամ առ անդամ, և արդյունքում ստացվում է նույն չափի, 1–երից ու 0–ներից կազմված զանգված: Եթե տրամաբանական օպերատորների օպերանդներից մեկը թիվ է, մյուսը՝ զանգված, ապա գործողությունը կատարվում է թվի և զանգվածի յուրաքանչյուր տարրի հետ, որի արդյունքում ստացվում է զանգվածի չափի, 1–երից ու 0–ներից կազմված զանգված:

Օրինակ.

```
>> a = [7 12 –1 0 0 34];
>> b = [–5 0 1 7 45 68];
>> a & b
   ans = 1     0     1     0     0     1
>> a | b
   ans = 1     1     1     1     1     1
>> a | 0
   ans = 1     1     1     0     0     1
>> b & 2
   ans = 1     0     1     1     1     1
```

4. Տրամաբանական ՈՉ օպերատորն ունի միայն մեկ օպերանդ: Եթե այն թիվ է, արդյունքը ևս թիվ է: Եթե այն զանգված է, արդյունքը նույն չափի մեկ այլ զանգված է, որի տարրերը հավասար են զրոյի, եթե օպերանդի համապատասխան ինդեքսով տարրերը ոչ զրոյական են, և մեկի, եթե դրանք հավասար են զրոյի:

Օրինակ.

```
>> ~a
   ans = 0     0     0     1     1     0
>> ~b
   ans = 0     1     0     0     0     0
```

Ինչպես ասացինք համեմատության և տրամաբանական օպերատորները կարող են օգտագործվել մաթեմատիկական տարբեր արտահայտություններում, թվաբանական օպերատորների հետ միասին: 1–ին գլխում գրել ենք արտահայտություններում թվաբանական գործողությունների կատարման կարգի մասին: Այժմ անհրաժեշտ է ընդհանրացնել այս կարգը, որպեսզի համեմատության և տրամաբանական օպերատորների կիրառման դեպքում ևս կարողանանք ստանալ ճիշտ արդյունք:

Գործողությունների կատարման կարգը **MatLab**–ում հետևյալն է.

1. Փակագծերը (եթե արտահայտության մեջ հանդիպում են ներդրված փակագծեր, ապա առաջնահերթությունը տրվում է ամենաներդրված փակագծին),
2. Աստիճանի բարձրացումները՝ ձախից աջ հանդիպելու հերթականությամբ,
3. Տրամաբանական ՈՉ օպերատորները՝ ձախից աջ հանդիպելու հերթականությամբ,
4. Բազմապատկումներն ու բաժանումները՝ ձախից աջ հանդիպելու հերթականությամբ,
5. Գումարումներն ու հանումները՝ ձախից աջ հանդիպելու հերթականությամբ,
6. Համեմատության օպերատորները՝ ձախից աջ հանդիպելու հերթականությամբ,
7. Տրամաբանական ԵՎ օպերատորները՝ ձախից աջ հանդիպելու հերթականությամբ,
8. Տրամաբանական ԿԱՄ օպերատորները՝ ձախից աջ հանդիպելու հերթականությամբ:

Օրինակ 1.

Ենթադրենք՝ ցանկանում ենք ստուգել $-9 \le x < -2$ պայմանը, որտեղ $x = -4$: Մաթեմատիկորեն ճիշտ գրված այս անհավասարությունը նույն տեսքով սխալ է գրել ծրագրավորելիս: Չնայած ձևական առումով –9<=x<–2 հրամանը ճիշտ է, սակայն այն կվերադարձնի սխալ արդյունք: Իրոք, համեմատության օպերատորներն արտահայտություններում իրականանում են ձախից աջ հանդիպելու հերթականությամբ: Հետևաբար, **MatLab**–ը նախ կիրականացնի –9<=x համեմատության գործողությունը, որի արդյունքը տրամաբանորեն ճիշտ է, այսինքն՝ հավասար է 1–ի: Այնուհետև ստացված 1–ը կհամեմատվի (–2) թվի հետ, այսինքն՝ կստուգվի 1<–2 պայմանը, որի արդյունքը տրամաբանորեն սխալ է, այսինքն՝ վերջնական պատասխանը կստացվի հավասար 0–ի: Այսպիսով, չնայած ձևականորեն $-9 \le x < -2$ պայմանն իրավացի է, սակայն **MatLab**–ում նույն տեսքով գրելիս արդյունքը կլինի սխալ:

Այսպիսի անհավասարությունները ճիշտ գրելու համար օգտվում են տրամաբանական ԵՎ օպերատորից և պայմանը գրում (–9<=x) & (x<–2) տեսքով: Քանի որ համեմատության օպերատորներն ունեն առաջնահերթություն տրամաբանական օպերատորների նկատմամբ, ապա նախ կստուգվեն 2 համեմատության գործողությունները, որոնց արդյունքները տրամաբանորեն ճիշտ են (=1), այնուհետև ստացված 1–երի հետ կկատարվի տրամաբանական ԵՎ գործողությունը, և վերջնական արդյունքը կլինի տրամաբանորեն ճիշտ:

Օրինակ 2.

Ենթադրենք՝ $z = 12$: ~(z<20) հրամանի արդյունքը կլինի 0, քանի որ փակագծերն ունեն առաջնահերթություն տրամաբանական ՈՉ օպերատորի նկատմամբ: Մինչդեռ ~z<20 հրամանը կվերադարձնի 1, քանի որ տրամաբանական ՈՉ օպերատորն ունի առաջնահերթություն համեմատության օպերատորի նկատմամբ: Այս դեպքում նախ կիրականանա ~z գործողությունը, որի արդյունքը հավասար կլինի 0–ի ($z = 12$, հետևաբար այն տրամաբանորեն ճիշտ է), իսկ 0<=20 համեմատության արդյունքը հավասար է 1–ի:

Վերջում դիտարկենք **MatLab**–ում սահմանված որոշ ներդրված տրամաբանական ֆունկցիաներ, որոնք ներկայացված են հետևյալ աղյուսակում.

Ֆունկցիան	**Նկարագրությունը**	**Օրինակ**
and(x, y)	համարժեք է x&y գործողությանը	`>> and(12, -64)` `ans = 1`
or(x, y)	համարժեք է x|y գործողությանը	`>> or(12, 0)` `ans = 1`
not(x)	համարժեք է ~x գործողությանը	`>> not(24)` `ans = 0`
xor(x, y)	Բացառող ԿԱՄ: Վերադարձնում է 1 միայն այն դեպքում, եթե մի օպերանդը 1 է, մյուսը՝ 0:	`>> xor(15, 1)` `ans = 0`

all(x)	Վերադարձնում է 1, եթե *x* վեկտորի բոլոր տարրերը տրամաբանորեն ճիշտ են, 0՝ եթե գոնե մի տարրը հավասար է 0–ի:
	Եթե արգումենտը մատրից է, ապա իր սյուները դիտարկվում են որպես վեկտորներ, և արդյունքում ստացվում է 1–երից և 0–ներից կազմված տող–վեկտոր:
		`>> x = [1 0 0 1 9 7 12];`
		`>> all(x)`
		`ans = 0`

any(x)	Վերադարձնում է 1, եթե *x* վեկտորի առնվազն մի տարրը տրամաբանորեն ճիշտ է, 0՝ եթե բոլոր տարրերը հավասար են 0–ի: Եթե արգումենտը մատրից է, ապա իր սյուները դիտարկվում են որպես վեկտորներ, և արդյունքում ստացվում է 1–երից և 0–ներից կազմված տող–վեկտոր:	`>> any(x)` `ans = 1`

find(x)	Եթե *x*–ը վեկտոր է, վերադարձնում է ոչ զրոյական տարրերի ինդեքսները	`>> find(x)` `ans = 1 4 5 6 7`
find(x<a)	Եթե *x*–ը վեկտոր է, վերադարձնում է այն տարրերի ինդեքսները, որոնք բավարարում են արգումենտում գրված պայմանին: Ակնհայտ է, որ արգումենտում կարող է գրվել կամայական տրամաբանական արտահայտություն:	`>> find(x>2)` `ans = 5 6 7`

isfinite(x)	Եթե *x*–ը զանգված է, վերադարձնում է նույն չափի այլ զանգված, որի տարրերը հավասար են 1–ի, եթե *x* զանգվածի համապատասխան ինդեքսներով տարրերը թվեր են և 0–ի, եթե դրանք թվեր չեն (հավասար են Inf կամ NaN):	`>> x = [7 Inf NaN 0 4 ];`
		`>> isfinite(x)`
		`ans = 1 0 0 1 1`

isinf(x)	Եթե *x*–ը զանգված է, վերադարձնում է նույն չափի այլ զանգված, որի տարրերը հավասար են 1–ի, եթե *x* զանգվածի համապատասխան ինդեքսներով տարրերը հավասար են Inf–ի և 0–ի, եթե դրանք թվեր են:	`>> isinf(x)` `ans = 0 1 0 0 0`
isnan(x)	Եթե *x*–ը զանգված է, վերադարձնում է նույն չափի այլ զանգված, որի տարրերը հավասար են 1–ի, եթե *x* զանգվածի համապատասխան ինդեքսներով տարրերը հավասար են NaN–ի և 0–ի, եթե դրանք թվեր են:	`>> isnan(x)` `ans = 0 0 1 0 0`

isprime(x)	Եթե *x*–ը զանգված է, վերադարձնում է նույն չափի այլ զանգված, որի տարրերը հավասար են 1–ի, եթե *x* զանգվածի համապատասխան ինդեքսներով տարրերը պարզ թվեր են և 0–ի, եթե դրանք պարզ չեն:	`>> x = [1 8 0 1 9 7 12];`
		`>> isprime(x)`
		`ans = 0 0 0 0 0 1 0`

isreal(x)	Եթե *x*–ը զանգված է, վերադարձնում է նույն չափի այլ զանգված, որի տարրերը հավասար են 1–ի, եթե *x* զանգվածի համապատասխան ինդեքսներով տարրերը իրական թվեր են և 0–ի, եթե դրանք կոմպլեքս են:	`>> x = [1+3i 8 2-4i];`
		`>> isreal(x)`
		`ans = 0 1 0`

6.4. ՃՅՈՒՂԱՎՈՐՄԱՆ ՕՊԵՐԱՏՈՐՆԵՐ

Ինչպես գիտենք, եթե հատուկ չի նշվում, **MatLab**–ում ծրագրերն իրականանում են տող առ տող, այսինքն՝ այն հերթականությամբ, որով գրված է ծրագրի մեջ: Սակայն հաճախ անհրաժեշտ է լինում ծրագիրը գրել այնպես, որ որևէ հրամանի կամ հրամանների մի խումբ իրականանա միայն որոշակի պայմանի դեպքում, իսկ հակառակ դեպքում **MatLab**–ը դրանք բաց թողնի և իրականացնի մնացած հրամանները: Այս նպատակով օգտագործվում են պայմանի **if** և ընտրության **switch** օպերատորները:

6.4.1. if պայմանի օպերատոր

if պայմանի օպերատորը ծրագրում հետագա քայլերն ընտրելու հնարավորություն է տալիս. կամ իրականանում են այդ օպերատորից հետո գրված հրամանները, կամ դրանք անտեսվում են: Ընտրությունը կախված է **if** օպերատորից հետո գրված պայմանի ճիշտ կամ սխալ լինելուց: Պայմանի օպերատորին կարելի է դիմել և՛ M–ֆայլերում, և՛ M–ֆունկցիաներում: Այդ պայմանը կարող է լինել կամայական բարդության տրամաբանական արտահայտություն:

Օրինակներ.

```
if x <=y
if c == d
if (a<b) & (c>d)
if (a~=5) | (z<x)
```

Յուրաքանչյուր **if** պայմանի օպերատոր պետք է անպայման ավարտվի **end** բառով: Պարզագույն պայմանի օպերատորը **if–end** օպերատորն է: Գոյություն ունեն ավելի բարդ ճյուղավորումներ ևս՝ **if–else–end** և **if–elseif–else–end** օպերատորները: Ծանոթանանք սրանցից յուրաքանչյուրին:

if–end պայմանի օպերատոր:

Այս պայմանի օպերատորի կառուցվածքը հետևյալն է.

```
...
... MatLab–ով գրված հրամանների 1–ին խումբ
...
if պայման
        ...
        ... MatLab–ով գրված հրամանների 2–րդ խումբ
        ...
end
...
... MatLab–ով գրված հրամաններ 3–րդ խումբ
...
```

if–end պայմանի օպերատոր պարունակող ծրագիրն իրականանում է տող առ տող, քանի դեռ չի հասել **if** հայտարարությանը (**MatLab**–ով գրված հրամաննե-

րի 1–ին խումբ): Երբ ծրագիրը հասնում է **if** հայտարարությանը, այն ստուգում է **if**–ից հետո գրված պայմանը: Եթե այդ պայմանը տեղի ունի (հավասար է 1–ի), ծրագիրը շարունակում է իրականացնել այն հրամանները, որոնք գրված են **if** հայտարարությունից հետո (**MatLab**–ով գրված հրամանների 2–րդ խումբ): Եթե պայմանը տեղի չունի (հավասար է 0–ի), ապա **MatLab**–ը այդ հրամանները բաց է թողնում, և ծրագիրը շարունակվում է **end** բառից հետո գրված հրամաններից (**MatLab**–ով գրված հրամանների 3–րդ խումբ):

Օրինակ.

Գրենք ծրագիր, որը տրված a, b, c թվերի համար կորոշի $ax^2+bx+c=0$ քառակուսի հավասարման արմատները և կվերադարձնի զգուշացում, եթե այդ հավասարման $D=b^2-4ac$ տարբերիչը (դիսկրիմինանտը) բացասական է:

Իհարկե, այս խնդիրը կարելի է լուծել որպես սովորական **MatLab**–ի ծրագիր՝ գրված M–ֆայլում: Սակայն, որպեսզի ծրագիրն ավելի ճկուն դառնա, հարմար է սահմանել առանձին M–ֆունկցիա, որը որպես մուտքային արգումենտներ կստանա հավասարման a, b, c գործակիցները, իսկ որպես ելքային արգումենտներ՝ կվերադարձնի $x1$, $x2$ լուծումները: Ֆունկցիան նախ a, b, c գործակիցների օգնությամբ կհաշվի D տարբերիչի արժեքը, այնուհետև կստուգի վերջինիս բացասական լինելը: Եթե այն կլինի բացասական, ապա ֆունկցիան կվերադարձնի զգուշացում, որ հավասարման լուծումները կոմպլեքս են, հակառակ դեպքում՝ առանց զգուշացման կվերադարձնի լուծումները:

```
function [x1, x2] = roots_eq(a, b, c)

    D = b^2 – 4*a*c;

    if D < 0
        warning('Equation has complex roots')
    end

    x1 = (–b + sqrt(D))/(2*a);
    x2 = (–b – sqrt(D))/(2*a);

end
```

Քանի որ ֆունկցիային անվանել ենք roots_eq, ապա ֆունկցիան պետք է պահպանել հենց այդ անունով: Այստեղ օգտագործել ենք **warning** (զգուշացում) ներդրված ֆունկցիան, որի մասին խոսել ենք սկրիպտային ֆայլերը նկարագրելիս: Այն թույլ է տալիս բացասական տարբերիչի դեպքում զգուշացման *Equation has complex roots* տողն արտածել հրամանի պատուհանում:

Այժմ, ֆունկցիան սահմանելուց և պահպանելուց հետո այն կարելի է կանչել հրամանի տողում՝ արգումենտում գրելով կամայական գործակիցներ:

```
>> [x1, x2] = roots_eq(2, 10, 3)
   x1 = –0.32
   x2 = –4.68

>> [x1, x2] = roots_eq(2, 1, 3)
```

```
Warning: Equation has complex roots
> In roots_eq at 6

x1 = -0.2500 + 1.1990i
x2 = -0.2500 - 1.1990i
```

Ինչպես տեսնում ենք առաջին դեպքում տարբերիչը դրական է, հետևաբար, առանց զգուշացման ստանում ենք հավասարման լուծումները: Երկրորդ դեպքում տարբերիչը բացասական է, և **MatLab**–ը վերադարձնում է ֆունկցիայում գրված զգուշացումը, սակայն ծրագիրը շարունակում է աշխատանքը և վերադարձնում ստացված կոմպլեքս լուծումները:

if–else–end պայմանի օպերատոր:

Այս պայմանի օպերատորը թույլ է տալիս հրամանների երկու հնարավոր խմբերից ընտրել մեկը և իրականացնել հրամանների այդ խումբը՝ կախված **if**–ից հետո գրված պայմանի իրականանալուց: **if–else–end** պայմանի օպերատորի կառուցվածքը հետևյալն է.

```
...
... MatLab–ով գրված հրամանների 1–ին խումբ
...

if պայման
    ...
    ... MatLab–ով գրված հրամանների 2–րդ խումբ
    ...
else
    ...
    ... MatLab– ով գրված հրամանների 3–րդ խումբ
    ...
end

...
... MatLab– ով գրված հրամանների 4–րդ խումբ
...
```

Նախ ծրագիրը հերթով իրականացնում է **if** հայտարարությունից առաջ գրված հրամանները (**MatLab**–ով գրված հրամանների 1–ին խումբ): Հասնելով **if** հայտարարությանը՝ ստուգվում է **if**–ից հետո գրված պայմանը: Եթե այն ճիշտ է (հավասար է 1–ի), ծրագիրը շարունակում է իրականացնել այն հրամանները, որոնք գրված են **if** հայտարարությունից հետո մինչև **else** բառը (**MatLab**–ով գրված հրամանների 2–րդ խումբ): Եթե պայմանը ճիշտ չէ (հավասար է 0–ի), ապա **MatLab**–ը իրականացնում է **else** բառից մինչև **end** բառը գրված հրամանները (**MatLab**–ով գրված հրամանների 3–րդ խումբ): Երկու դեպքում էլ ծրագիրը շարունակում է իրականացնել **end** բառից հետո գրված հրամանները (**MatLab**–ով գրված հրամանների 4–րդ խումբ):

Օրինակ.

Սահմանենք $y = \begin{cases} x, & x \le 0 \\ \sin x, & x > 0 \end{cases}$ ֆունկցիան և կառուցենք այս ֆունկցիայի գրաֆիկը $[-10; 10]$ միջույթում:

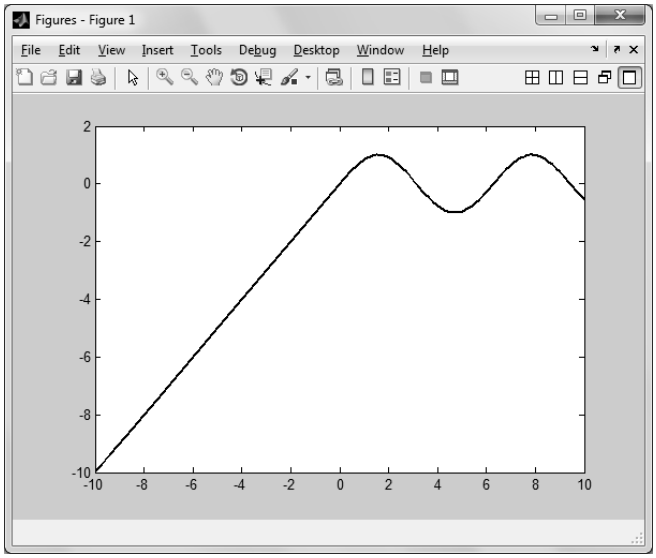
Ֆունկցիան պետք է ունենա հետևյալ տեսքը.

```
function y = mfun(x)
    if x <= 0
        y = x;
    else
        y = sin(x);
    end
end
```

Այնուհետև, **fplot** ֆունկցիան օգտագործելով, հրամանի տողում գրենք գրաֆիկը կառուցելու հետևյալ հրամանը.

```
>> fplot('mfun', [-10 10])
```

Սահմանված mfun ֆունկցիայի գրաֆիկը պատկերված է նկ. 37–ում.



Նկ. 37

Նկատենք, որ կտոր առ կտոր սահմանված ֆունկցիայի գրաֆիկը **plot** ֆունկցիայով կառուցելու դեպքում սխալ արդյունք կստանանք: Պատճառը համեմատության գործողությունների առանձնահատկությունն է վեկտորների դեպքում, որը մանրամասն քննարկել ենք 6.3. -ում: Ստուգելու համար mfun ֆունկցիան սահմանելուց հետո գրենք՝

```
x = [-10:0.01:10];
y = mfun(x);
plot(x,y)
```

հրամանները և համոզվենք, որ ստացված գրաֆիկը տարբերվում է **fplot** ֆունկցիայով կառուցված գրաֆիկից:

Հետագայում, **for** ցիկլի օպերատորը սահմանելիս, ցույց կտանք, թե ինչպես կարելի է **plot** ֆունկցիայի օգնությամբ ստանալ ճիշտ գրաֆիկը:

if–elseif–else–end պայմանի օպերատոր:

Այս պայմանի օպերատորը թույլ է տալիս իրականացնել առավել բարդ ճյուղավորումներ: **if–elseif–else–end** պայմանի օպերատորի պարզագույն տեսքը հետևյալն է.

```
...
... MatLab–ով գրված հրամանների 1–ին խումբ
...
if պայման 1
    ...
    ... MatLab–ով գրված հրամանների 2–րդ խումբ
    ...
elseif պայման 2
    ...
    ... MatLab– ով գրված հրամանների 3–րդ խումբ
    ...
else
    ...
    ... MatLab– ով գրված հրամանների 4–րդ խումբ
    ...
end

...
... MatLab– ով գրված հրամանների 5–րդ խումբ
...
```

Ծրագիրը նախ հերթով իրականացնում է **if** հայտարարությունից առաջ գրված հրամանները (**MatLab**–ով գրված հրամանների 1–ին խումբ): Հասնելով **if** հայտարարությանը՝ ստուգվում է **if**–ից հետո գրված պայմանը (պայման 1): Եթե այն ճիշտ է, ծրագիրը շարունակում է իրականացնել այն հրամանները, որոնք գրված են **if** հայտարարությունից հետո մինչև **elseif** հայտարարությունը (**MatLab**–ով գրված հրամանների 2–րդ խումբ): 1–ին պայմանի սխալ լինելու դեպքում **MatLab**–ը ստուգում է **elseif** հայտարարությունից հետո գրված պայմանը (պայման 2): Եթե վերջինս ճիշտ է, ապա իրականանում են մինչև **else** հայտարարությունը գրված հրամանները (**MatLab**–ով գրված հրամանների 3–րդ խումբ): Եթե 2–րդ պայմանը ևս սխալ է, ապա իրականանում են **else** բառից մինչև **end** բառը գրված հրամանները (**MatLab**–ով գրված հրամանների 4–րդ խումբ): Եթե **else** հայտարարությունը բացակայում է, ապա **if** և **end** տիրույթի որևէ հրաման չի իրակա-

նանում: Իսկ **end** բառից հետո գրված հրամանները (**MatLab**–ով գրված հրամանների 5–րդ խումբ) բոլոր դեպքերում իրականանում են:

Օրինակ.

Սահմանենք $y=\begin{cases}\sin^2 x, & x<-\pi\\ x, & -\pi\le x<\pi\\ \cos^2 x, & x\ge\pi\end{cases}$ ֆունկցիան և կառուցենք այս ֆունկցիայի գրաֆիկը $[-3\pi;3\pi]$ միջույթում:

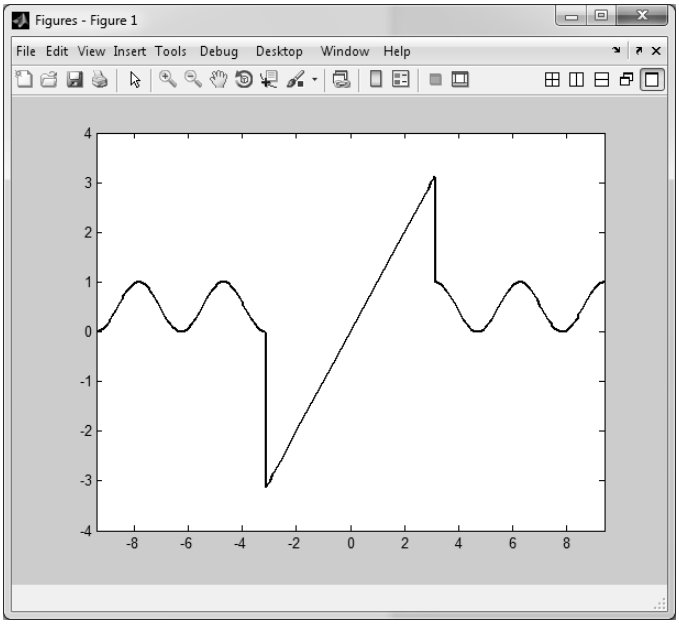
Ֆունկցիան պետք է ունենա հետևյալ տեսքը.

```
function y = mfun1(x)
    if x < -pi
        y = sin(x).^2;
    elseif (x >= -pi) & (x < pi)
        y = x;
    else
        y = cos(x).^2;
    end
end
```

Այժմ հրամանի տողում գրենք գրաֆիկը կառուցելու հետևյալ հրամանը.

```
>> fplot('mfun1', [-3*pi 3*pi])
```

Սահմանված mfun1 ֆունկցիայի գրաֆիկը բերված է նկ. 38–ում:



Նկ. 38

6.4.2. switch ընտրության օպերատոր

Ծրագրի իրականացման ընթացքը կարելի է փոխել ոչ միայն **if** պայմանի օպերատորի, այլև ընտրության **switch–case** օպերատորի օգնությամբ: Այս օպերատորը թույլ է տալիս մի քանի հնարավոր հրամանների խմբից ընտրել և իրականացնել մի խումբը: Ընտրության օպերատորի սխեման ունի հետևյալ տեսքը.

```
...
...  MatLab–ով գրված հրամանների 1–ին խումբ
...
switch արտահայտություն
    case արժեք 1
        ...
        ... MatLab–ով գրված հրամանների 2–րդ խումբ
        ...
    case արժեք 2
        ...
        ... MatLab–ով գրված հրամանների 3–րդ խումբ
        ...
    ...
    case արժեք N
        ...
        ... MatLab–ով գրված հրամանների N–րդ խումբ
        ...
    otherwise
        ...
        ... MatLab–ով գրված հրամանների (N+1)–րդ խումբ
        ...
end

...
...  MatLab–ով գրված հրամանների (N+2)–րդ խումբ
...
```

Այսպիսով՝ նախ անհրաժեշտ է գրել **switch** հայտարարությունը, որին հետևում է որևէ արտահայտություն: Այս արտահայտության ներքո հասկացվում է կամ պարզապես նախապես որևէ արժեք ունեցող փոփոխականի անուն, կամ որևէ մաթեմատիկական արտահայտություն՝ բաղկացած մեկ կամ մի քանի փոփոխականներից, որոնց ևս նախապես արժեքներ են վերագրված: Այնուհետև գրվում են **case** բառերը, որոնցից յուրաքանչյուրի կողքին գրվում է որևէ արժեք: Գրված արտահայտության արժեքը հերթով համեմատվում է այս արժեքների հետ: Եթե կա որևէ համընկում, ապա իրականանում է համապատասխան **case**–ի ներսում գրված հրամանների խումբը: Օրինակ, եթե սխեմայում գրված արտահայտությունը հավասար է արժեք 2–ին, ապա իրականանում են միայն դրանից հետո գրված հրամանները (**MatLab**–ով գրված հրամանների 2–րդ խումբ): Որևէ այլ հրամանների

խումբ չի իրականանում: Բացի **case** հայտարարություններից, **switch**–ը կարող է պարունակել նաև **otherwise** հայտարարությունը: Եթե որևէ համընկում չի լինում, ապա իրականանում է **otherwise** բառից մինչև **end** բառն ընկած հրամանների խումբը (**MatLab**–ով գրված հրամանների (N+1)–րդ խումբ): Եթե որևէ համընկում չկա, իսկ **otherwise** հայտարարությունը բացակայում է, ապա ընտրման օպերատորի որևէ հրաման չի իրականանում: Թվարկած դեպքերից յուրաքանչյուրի ժամանակ ծրագիրը շարունակում է աշխատանքը՝ իրականացնելով **end** բառից հետո գրված հրամանները (**MatLab**–ով գրված հրամանների (N+2)–րդ խումբ):

case հայտարարություններից յուրաքանչյուրը կարող է պարունակել ոչ միայն մեկ, այլև մի քանի արժեք (օրինակ՝ ***case*** *արժեք1 արժեք2 արժեք3*):

Օրինակ.

M–ֆայլում գրենք ծրագիր, որը թույլ կտա ներմուծել 1–ից 7 միջակայքում որևէ թիվ և վերադարձնել դրան համապատասխան շաբաթվա օրվա անվանումը (1–ը կհամապատասխանի երկուշաբթի օրվան, 2–ը՝ երեքշաբթի օրվան և այլն): Եթե ներմուծված թիվը չի պատկանում 1–ից 7 միջակայքին, ապա ծրագիրը կվերադարձնի զգուշացում:

Ծրագիրը կունենա հետևյալ տեսքը.

```
x = input('x = ');

switch x
    case 1
        disp('Monday')
    case 2
        disp('Tuesday')
    case 3
        disp('Wednesday')
    case 4
        disp('Thursday')
    case 5
        disp('Friday')
    case 6
        disp('Saturday')
    case 7
        disp('Sunday')
    otherwise
        warning('Invalid Week Day')
end
```

Աշխատեցնենք ծրագիրը 2 անգամ՝ մի անգամ *x*–ին վերագրելով 5 արժեքը, որը համապատասխանում է ուրբաթ օրվան, մյուս անգամ՝ 8 արժեքը, որը չի համապատասխանում շաբաթվա որևէ օրվա: Հրամանի պատուհանում կստանանք.

```
x = 5
Friday

x = 8
Warning: Invalid Week Day
```

6.5. ՑԻԿԼԻ ՕՊԵՐԱՏՈՐՆԵՐԸ

Ցիկլի օպերատորները հնարավորություն են տալիս ծրագրի որևէ հրաման կամ հրամանների մի խումբ իրականացնել ոչ թե մեկ, այլ մի քանի անգամ: Այդ հրամանների յուրաքանչյուր իրականացումը կոչվում է իտերացիա: Ցիկլի օպերատորների հրամանները իրականանում են, քանի դեռ որոշակի պայման բավարարվում է, հակառակ դեպքում այդ հրամանները բաց են թողնվում: Եթե այդ պայմանը ոչ մի քայլում չի խախտվում, ապա ցիկլի օպերատորը չի դադարեցնում աշխատանքը: Այդպիսի ցիկլերը կոչվում են անվերջ ցիկլեր: Այս բաժնում կծանոթանանք ցիկլի **while** և **for** օպերատորների, ինչպես նաև ներդրված ցիկլերի և ցիկլերի ընդհատման **break** և **continue** օպերատորների հետ:

6.5.1. while ցիկլի օպերատոր

while ցիկլի օպերատորն օգտագործվում է այն դեպքերում, երբ ցիկլերում իտերացիաների քանակը նախապես հայտնի չէ: Ցիկլի ներսում գրված հրամանները իրականանում են այնքան ժամանակ, քանի դեռ ցիկլի իրականացման պայմանը ճիշտ է: **while** օպերատորը կարելի է ներկայացնել հետևյալ կերպ.

```
...
...  MatLab–ով գրված հրամանների 1–ին խումբ
...

while պայման
    ...
    ... MatLab–ով գրված հրամանների 2–րդ խումբ
    ...
end

...
...  MatLab– ով գրված հրամանների 3–րդ խումբ
...
```

Երբ ծրագիրը հասնում է **while** հայտարարությանը, ստուգվում է այդ հայտարարությունից հետո գրված պայմանը: Եթե այդ պայմանը տրամաբանորեն սխալ է (հավասար է 0–ի), ապա **while** և **end** հայտարարությունների միջև գրված հրամանները (**MatLab**–ով գրված հրամանների 2–րդ խումբ) բաց են թողնվում, և **MatLab**–ը շարունակում է իրականացնել **end** բառից հետո գրված հրամանները (**MatLab**–ով գրված հրամանների 3–րդ խումբ): Եթե պայմանը տրամաբանորեն ճիշտ է (հավասար է 1–ի), ապա **MatLab**–ն իրականացնում է **while** և **end** հայտարարությունների միջև գրված հրամանները (**MatLab**–ով գրված հրամանների 2–րդ խումբ): Այնուհետև ծրագիրը դարձյալ վերադառնում է **while** հայտարարությանը և

դարձյալ ստուգվում է պայմանը: Քանի դեռ պայմանը տրամաբանորեն ճիշտ է, իտերացիաները հաջորդաբար շարունակվում են: Այն պահին, երբ պայմանը դառնում է տրամաբանորեն սխալ, **MatLab**–ը բաց է թողնում **while** և **end** հայտարարությունների միջև գրված հրամանները, և անցնում է **end** հայտարարությունից հետո գրված հրամաններին: Եթե որևէ պատճառով պայմանը սխալ չի դառնում, իտերացիաներն անընդհատ շարունակվում են, ստացվում է անվերջ ցիկլ:

Որպեսզի ցիկլը վերջավոր լինի, **while** հայտարարությունից հետո գրված պայմանը պետք է պարունակի առնվազն մեկ փոփոխական (այս փոփոխականը կոչվում է ցիկլի փոփոխական), ընդ որում վերջինիս նախապես պետք է որևէ սկզբնական արժեք վերագրվի, իսկ հետո, յուրաքանչյուր իտերացիայի ժամանակ (**while** և **end** հայտարարությունների միջև) նոր արժեք ստանա: Եթե **while** հայտարարությունից հետո գրված պայմանը մի քանի փոփոխական պարունակող արտահայտություն է, ապա դրանցից յուրաքանչյուրին նախապես պետք է սկզբնական արժեք վերագրված լինի, իսկ վերջավոր ցիկլ ստանալու համար պետք է դրանցից առնվազն մեկին ցիկլի օպերատորի ներսում նոր արժեք վերագրել: Հակառակ դեպքերում ցիկլի իրականացման պայմանը երբեք չի խախտվի:

Օրինակ.

Գրենք ծրագիր, որն էկրանին կարտածի բոլոր այն դրական ամբողջ x թվերի քառակուսիները, քանի դեռ իրականանում է $x^3 < 1500$ պայմանը:

```
x = 1;
while x^3<1500
    disp(x^2)
    x = x+1;
end
```

Ստորև բերված են ծրագրի իրականացման արդյունքում ստացված արժեքները:

```
   1
   4
   9
  16
  25
  36
  49
  64
  81
 100
 121
```

6.5.2. for ցիկլի օպերատոր

Ի տարբերություն ցիկլի **while** օպերատորի, ցիկլի **for** օպերատորն ավելի հարմար է օգտագործել այն դեպքերում, երբ ցիկլերի քանակը նախապես հայտնի է: **for** օպերատորի տեսքը հետևյալն է.

```
...
... MatLab–ով գրված հրամանների 1–ին խումբ
...
for k = m:p:n

    ...
    ... MatLab–ով գրված հրամանների 2–րդ խումբ
    ...
end

...
... MatLab– ով գրված հրամանների 3–րդ խումբ
...
```

Այստեղ *k*–ն ցիկլի փոփոխականն է, *m*–ը՝ *k*–ի արժեքը ցիկլի սկզբում (առաջին իտերացիայում), *n*–ը՝ վերջին արժեքը, իսկ *p*–ն՝ *k*–ի փոփոխության քայլը: Երբ ծրագիրը հասնում է **for** հայտարարությանը, *k* ցիկլի փոփոխականին վերագրվում է *m* սկզբնական արժեքը (k=m), և **MatLab**–ը սկսում է իրականացնել **for** և **end** հայտարարությունների միջև գրված հրամանները (**MatLab**–ով գրված հրամանների 2–րդ խումբ): Երբ ծրագիրը հասնում է **end** հայտարարությանը, այն վերադառնում է **for** տողին, *k*–ն ընդունում է նոր արժեք (k=m+p), և **MatLab**–ը դարձյալ սկսում է իրականացնել **for** և **end** հայտարարությունների միջև գրված հրամանները: Ցիկլի այս պրոցեսը շարունակվում է այնքան ժամանակ, քանի դեռ *k*–ն չի ընդունել իր վերջնական *n* արժեքը: Այս դեպքում ծրագիրը չի վերադառնում **for** տողին, այլ սկսում է իրականացնել **end** հայտարարությունից հետո գրված հրամանները: Օրինակ, **for k = 3:2:11** հայտարարության դեպքում *k*–ն կընդունի 3, 5, 7, 9, 11 արժեքները, իսկ ցիկլը կիրականանա 5 անգամ:

Ցիկլի **for** օպերատորն օգտագործելուց պետք է հիշել, որ

1. ցիկլի **for** օպերատորը պետք է ավարտվի **end** հայտարարությամբ;
2. ծրագրավորման մեջ հաճախ ցիկլի փոփոխականը նշանակում են *i* կամ *j* տառերով: Սակայն **MatLab**–ում պետք է խուսափել այդ անուններն օգտագործելուց, քանի որ այս տառերով, ինչպես հայտնի է, սահմանված է կոմպլեքս թվերի կեղծ միավորը:
3. Ցիկլի փոփոխականի քայլի *p* արժեքը կարող է լինել ոչ միայն դրական, այլև բացասական: Օրինակ՝ **for k = 8:–3:–4** հայտարարության մեջ *k*–ն

կընդունի 8, 5, 2, –1, –4 արժեքները, և ցիկլը կիրականանա 5 անգամ: Ծրագրավորման մեջ դրական p–ն անվանում են ինկրեմենտ, բացասական p–ն՝ դեկրեմենտ:

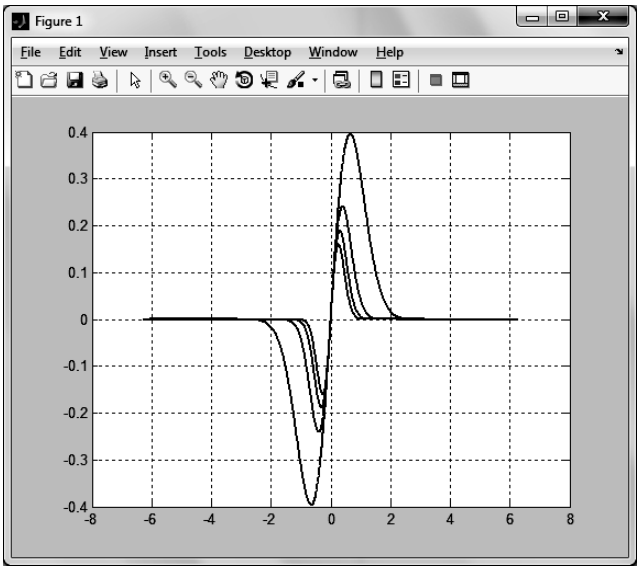
4. Անհրաժեշտ է նկատի ունենալ, որ $p>0$ և $m>n$ կամ $p<0$ և $m<n$ դեպքերում ցիկլը չի իրականանում:
5. Եթե $m=n$, ցիկլն իրականանում է միայն մեկ անգամ:
6. Եթե $p=1$, ապա ցիկլի հայտարարության մեջ այն կարելի է բաց թողնել: Օրինակ՝ **for k = 3:8** հայտարարության մեջ k–ն կընդունի 3, 4, 5, 6, 7, 8 արժեքները, և ցիկլը կիրականանա 6 անգամ:
7. Եթե m, p, n թվերն ընտրված են այնպես, որ k–ն չի ընդունում n արժեքը, ապա ցիկլի վերջին իտերացիան իրականանում է k–ի՝ մինչև n–ը ընդունած ամենավերջին արժեքի դեպքում: Օրինակ՝ **for k = 3:2:10** դեպքում k–ն չի ընդունում 10 արժեքը, հետևաբար վերջին իտերացիայում k–ի արժեքը կլինի 9–ը (k = 3, 5, 7, 9):
8. k փոփոխականին նոր արժեք չի կարող վերագրվել **for** ցիկլի ներսում: Այն իր արժեքները պետք է ստանա միայն **for** հայտարարության մեջ:
9. **for** ցիկլի հայտարարության մեջ k–ն կարող է ունենալ հատուկ արժեքներ, որոնք տրվում են վեկտորի տեսքով: Օրինակ՝ **for k = [3 –8 0 2 7 19]**:
10. **for** ցիկլի ավարտից հետո k–ն հավասար է իրեն վերագրված ամենավերջին արժեքին:

Օրինակ.

Կառուցենք $y = e^{-kx^2}\sin x$ ֆունկցիայի գրաֆիկները k=1, 3, 5, 7 արժեքների համար $x \in [-2\pi;\ 2\pi]$ տիրույթում: Ծրագիրը կունենա հետևյալ տեսքը.

```
x = [-2*pi:0.01:2*pi];

for k = 1:2:7
    y = exp(-k*x.^2).*sin(x);
    plot(x, y)
    grid on
    hold on
end
```

Ֆունկցիայի գրաֆիկները բերված են նկ. 39–ում:



Նկ. 39

for ցիկլի օպերատորի օգնությամբ լուծվող որոշ խնդիրներ կարելի է լուծել նաև վեկտորների հետ անդամ առ անդամ կատարվող գործողություններով (տե՛ս գլուխ 2)։ Սա **MatLab**–ի առավելություններից մեկն է ծրագրավորման այլ լեզուների նկատմամբ, քանի որ անդամ առ անդամ գործողությունները համակարգիչն իրականացնում է ավելի արագ, քան ցիկլերը։ Հետևաբար, նման խնդիրների հանդիպելիս խորհուրդ է տրվում օգտագործել անդամ առ անդամ գործողությունները։

Օրինակ.

Ենթադրենք՝ անհրաժեշտ է հաշվել $s = \sum_{k=0}^{100} \frac{(-1)^k}{2k+1}$ գումարը։ Ցանկացած ծրագրավորման լեզվով այս խնդիրը կլուծեն **for** ցիկլի օպերատորի օգնությամբ։ **MatLab**–ում լուծման այդ եղանակը կունենա հետևյալ տեսքը.

```
s = 0;
for k = 0:100
  s = s + (-1).^k * 1./(2*k+1);
end
s
s = 0.7879
```

Շատ ավելի հեշտ և իրականացման տեսակետից ավելի արագ կարելի է նույն խնդիրը լուծել անդամ առ անդամ գործողություններով։ Այս եղանակով ծրագիրը կգրվի հետևյալ կերպ.

```
>> k = [0:100];
>> y = (-1).^k * 1./(2*k+1);
>> s = sum(y)
   s = 0.7879
```

Օրինակ.

Սահմանենք $y=\begin{cases} x, & x \le 0 \\ \sin x, & x > 0 \end{cases}$ ֆունկցիան և կառուցենք այս ֆունկցիայի գրաֆիկը $[-10;10]$ տիրույթում:

Այս խնդրին անդրադարձել ենք **if** պայմանի օպերատորը նկարագրելիս: Հիշեցնենք, որ եթե ֆունկցիան սահմանում էինք միայն պայմանի օպերատոր օգտագործելով, ապա ֆունկցիայի գրաֆիկը հնարավոր է կառուցել միայն **fplot** ֆունկցիայի օգնությամբ: **plot** ֆունկցիան այս դեպքում ճիշտ արդյունք չէր վերադարձնում այն պատճառով, որ վեկտորների հետ համեմատության օպերատորները հատուկ ձևով են կիրառվում: Այժմ սահմանենք միևնույն ֆունկցիան՝ պայմանի օպերատորի հետ կիրառելով նաև ցիկլի **for** օպերատորը և ֆունկցիայի գրաֆիկը կառուցենք սովորական **plot** ֆունկցիայի օգնությամբ: Ֆունկցիան կունենա հետևյալ տեսքը.

```
function y = mfun(x)

    L = length(x);

    for i = 1:L
       if x(i) <= 0
           y(i) = x(i);
       else
           y(i) = sin(x(i));
       end
    end
end
```

Ֆունկցիան սահմանելուց հետո հրամանի տողում գրենք հետևյալ ծրագիրը.

```
>> x = [-10:0.01:10];
>> y = mfun(x);
>> plot(x,y)
```

Ինչպես տեսնում ենք, ֆունկցիան այսպես սահմանելու դեպքում **plot** ֆունկցիայի օգնությամբ գրաֆիկը ստանում է նույն տեսքը, ինչ **fplot** ֆունկցիայի օգնությամբ՝ ֆունկցիան միայն **if** պայմանի օպերատորով սահմանելու դեպքում (տես Նկ. 37):

6.5.3. Ներդրված ցիկլեր

Երբեմն հնարավոր են դեպքեր, երբ ցիկլերի կամ պայմանի օպերատորները լինում են ներդրված, օրինակ, ցիկլի **for** օպերատորի ներսում գրվում է մեկ այլ **for** օպերատոր: Ներդրված օպերատորների քանակի որևէ սահմանափակում չկա: Ներդրված կարող են լինել **if**, **case**, **while** և **for** օպերատորները: Այստեղ անհրաժեշտ է հիշել, որ յուրաքանչյուր ցիկլի կամ պայմանի օպերատոր պետք է ավարտվի **end** հայտարարությամբ:

Օրինակ.

Հաշվենք $s=\sum_{k=1}^{10}\sum_{m=1}^{8}(m^3+k^3)$ կրկնակի գումարը: Նկատենք, որ այս գումարը հաշվելիս k թվի յուրաքանչյուր արժեքի համար անհրաժեշտ է հաշվել $\sum_{m=1}^{8}(m^3+k^3)$ գումարը և տարբեր k–երի համար հաշվված այդ գումարների արժեքները գումարել իրար: Հետևաբար, ըստ k–ի արտաքին գումարը կլինի հիմնական ցիկլը, իսկ ըստ m–ի ներքին գումարը՝ ներքին ցիկլը: Ծրագիրը կունենա հետևյալ տեսքը.

```
s = 0;
for k = 1:10
    for m = 1:8
        s = s + (m^3 + k^3);
    end
end
s
```

Այս ծրագրում k=1 դեպքում m-ը կընդունի 1, 2, ... 8 արժեքները և յուրաքանչյուր քայլում s-ին կգումարվի m^3+1 ընթացիկ արժեքը: Այնուհետև k–ն կհավասարվի 2-ի, այս դեպքում ևս m-ը կընդունի 1, 2, ..., 8 արժեքները, և s-ին կվերագրվի m^3+4 ընթացիկ արժեքը: Ցիկլերը կավարտվեն այն ժամանակ, երբ s-ի մեջ կպարունակվեն կրկնակի գումարի բոլոր տարրերը:

6.6. BREAK ԵՎ CONTINUE ՀՐԱՄԱՆՆԵՐԸ

break և **continue** հրամանները նախատեսված են ցիկլերն ընդհատելու համար: Եթե **break** հրամանը հանդիպում է **for** կամ **while** ցիկլի ներսում, ապա **MatLab**–ն այլևս չի իրականացնում ցիկլի ներսում գրված որևէ հրաման, այլ միանգամից անցնում է ցիկլի **end** հայտարարությանը և շարունակում **end**–ից հետո (այսինքն՝ ցիկլից հետո) գրված հրամանների իրականացմանը: Եթե **break** հրամանը հանդիպում է ներդրված ցիկլի ներսում, այն ընդհատում է միայն ներդրված ցիկլի աշխատանքը: **break** հրամանը կարող է օգտագործվել ոչ միայն ցիկլում, այլև ծրագրի կամ M–ֆունկցիայի կամայական տողում: Այդ դեպքում ծրագիրը պարզապես դադարեցնում է աշխատանքը: Ցիկլերի ներսում **break** հրամանը սովորաբար գրվում է որոշակի պայմանի հետ և իրականանում է այդ պայմանը ճիշտ լինելու դեպքում, որից հետո ցիկլն ընդհատվում է:

Ի տարբերություն **break** հրամանի, **continue** հրամանը ընդհատում է ցիկլի միայն տվյալ իտերացիայի աշխատանքը և դարձյալ վերադառնում է **for** կամ **while** տողին:

Օրինակ.

Գրենք ծրագիր, որը օգտագործողին հնարավորություն կտա մուտքագրել 10 հատ դրական թիվ և հաշվել դրանց միջին թվաբանականը: Ծրագիրը կդադարեցնի թվերի

մուտքագրումը այն դեպքում, երբ օգտագործողը կներմուծի բացասական թիվ և կվերադարձնի մինչ այդ ներմուծած թվերի միջին թվաբանականը:

```
s = 0;
n = 0;

for i = 1:10
    num = input('num = ')
    if num < 0
        break
    end
    s = s + num;
    n = n + 1;
end

average = s/n
```

Այս ծրագրում *s*–ով նշանակված է մուտքագրված դրական թվերի գումարը, իսկ *n*–ով՝ ներմուծած դրական թվերի քանակը: **for** ցիկլի օպերատորը թույլ է տալիս օգտագործողին մուտքագրել 10 թիվ **input** ֆունկցիայի օգնությամբ: Մուտքագրված թվերը վերագրվում են *num* փոփոխականին: Հերթական թիվը մուտքագրելուց հետո ծրագիրը ստուգում է այդ թվի բացասական լինելու պայմանը: Եթե թիվը բացասական չէ, ապա **if** պայմանի օպերատորի պայմանը չի իրականանում, և ծրագիրը հաշվում է հերթական ներմուծած թվի գումարը և քանակը: Եթե, հակառակը, ներմուծված թիվը եղել է բացասական, ապա ծրագիրն իրականացնում է **if** պայմանի օպերատորի ներսում գրված **break** հրամանը, և ցիկլը դադարեցնում է աշխատանքը, իսկ ծրագիրն անմիջապես անցնում է *average* փոփոխականի արժեքը հաշվելուն, որը հավասար է մինչ այդ ներմուծած թվերի միջին թվաբանականին: Ուշադրություն դարձրեք, որ թվի բացասական լինելու պայմանը ստուգվում է նախքան գումարի և դրական թվերի քանակի հաշվարկը, քանի որ, հակառակ դեպքում, ներմուծած բացասական թիվը ևս կգումարվեր մինչ այդ ներմուծված թվերի գումարին, ինչպես նաև ներմուծած թվերի քանակն էլ կավելանար 1–ով:

Այժմ խնդրի պայմանը փոխենք այնպես, որ ծրագիրը ոչ թե հաշվի մինչև առաջին բացասական թիվը ներմուծված դրական թվերի միջին թվաբանականը, այլ ներմուծված բոլոր դրական միջին թվաբանականը՝ անտեսելով ներմուծված բացասական թվերը: Խնդիրն այսպես փոխելիս, բացասական թիվ ներմուծելուց հետո ծրագիրը ոչ թե պետք է անցնի *average* փոփոխականի արժեքը հաշվելուն, այլ պարզապես այդ թիվը հաշվի չառնի, և օգտագործողը նոր թիվ ներմուծի: Դրա համար ընդամենը պետք է գրված ծրագրում **break** հրամանը փոխարինել **continue** հրամանով:

6.7. ԱՇԽԱՏԱՆՔ ՏՈՂԵՐԻ ՀԵՏ

MatLab–ը հիմնականում նախատեսված է թվային տվյալների հետ աշխատելու համար, սակայն աշխատանքի ընթացքում հաճախ անհրաժեշտություն է առաջանում գործ ունենալ տեքստերի հետ: Նման դեպքի մենք հանդիպել ենք, երբ կառուցված գրաֆիկին և նրա առանցքներին անուններ էինք վերագրում: «Տեքստ» տերմինը **MatLab**–ում վերաբերում է ինչպես առանձին սիմվոլներին, այնպես էլ ամբողջ տողերին:

Տողերը (կամ առանձին սիմվոլները) **MatLab**–ում սահմանվում են թվային փոփոխականների նման, միայն թե գրվում են ապոստրոֆների մեջ:

Օրինակ.

```
>> s = 'Hello, World!'
   s = Hello, World!
```

Այս օրինակում սահմանվեց *s* անունով տեքստային փոփոխականը, որին վերագրվեց *Hello, World!* տողը: *s* փոփոխականի մասին տեղեկություն կարելի է ստանալ՝ օգտվելով **whos** հրամանից:

```
>> whos s
   Name      Size            Bytes  Class    Attributes
   s         1x13               26  char
```

Ինչպես տեսնում ենք, *s* տեքստային փոփոխականը սիմվոլային (**char**) տիպի է, բաղկացած է 13 սիմվոլներից, 1×13 չափի տող–վեկտոր է և հիշողության մեջ զբաղեցնում է 26 բայթ, այսինքն՝ յուրաքանչյուր սիմվոլ զբաղեցնում է 2 բայթ:

Այսպիսով՝ տեքստերը զանգվածներ են, հետևաբար զանգվածներին վերաբերող շատ ֆունկցիաներ կարելի է կիրառել տեքստերի նկատմամբ: Օրինակ՝ տողի առանձին սիմվոլին կամ սիմվոլների որոշ հաջորդականության դիմելու համար կարելի է օգտագործել վեկտորների ինդեքսավորման գործողությունները.

```
>> s(8)
   ans = W
>> s1 = [s(1:5) s(13)]
   s1 = Hello!
```

Այս օրինակում նախ դիմեցինք վերևում սահմանված *s* տողի 8–րդ ինդեքսով տարրին, որի արժեքը *W*–ն է: Այնուհետև *s*1 փոփոխականին վերագրեցինք *s* փոփոխականի 1–ից 5–րդ տարրերի ինդեքսները և 13–րդ ինդեքսով արժեքը, այսինքն՝ *s*1–ին վերագրվեց *Hello!* բառը:

Մի քանի տող իրար միացնելը ծրագրավորման մեջ կոչվում է նաև կոնկատենացիա: *s*1 փոփոխականին վերագրված *Hello!* բառը կոնկատենացիայի օրինակ է: Բացի կոնկատենացիայի այս եղանակից, **MatLab**–ում կա նաև ներդրված **strcat** ֆունկցիան, որը կոնկատենացնում է արգումենտում ստորակետներով անջատված տողերը: Օրինակ, վերևում սահմանված *s*1 փոփոխականը **strcat** ֆունկցիայի օգնությամբ կսահմանվի հետևյալ կերպ.

```
>> str1 = s(1:5);
>> str2 = s(13);
>> s1 = strcat(str1, str2)
   s1 = Hello!
```

strcat ֆունկցիայի արգումենտում կարող են լինել 2–ից ավելի տողեր:

Բացի **strcat** ֆունկցիայից, **MatLab**–ը ունի տողերի հետ աշխատելու համար նախատեսված այլ ներդրված ֆունկցիաներ ևս: Դիտարկենք հետևյալ տողը.

```
>> s = 'This is an example of text variable.';
```

Ենթադրենք, ցանկանում ենք այս տողի մեջ որոնել *example* բառը: Դրա համար նախատեսված է **findstr(s, str)** ներդրված ֆունկցիան, որտեղ *str*–ը այն տեքստն է, որը ցանկանում ենք որոնել *s* տողի մեջ: *example* բառը *s* տողի մեջ որոնելու համար, այսպիսով, կգրենք.

```
>> str = 'example';
>> findstr(s, str)
   ans = 12
```

Նշանակում է, որ *str* տեքստը, այսինքն՝ *example* բառը *s* տողի մեջ հանդիպում է 12–րդ ինդեքսով դիրքում: Եթե *str* բառը հանդիպի մի քանի անգամ, ելքում վեկտորի տեսքով կվերադարձվեն բոլոր հանդիպումների ինդեքսները:

Երկու՝ *s*1 և *s*2 տող համեմատելու համար նախատեսված է **strcmp(s1, s2)** ֆունկցիան: Այս ֆունկցիայի ելքում ստացվում է տրամաբանական 1, եթե *s*1 և *s*2 տողերը համընկնում են, 0՝ հակառակ դեպքում:

Օրինակ.

```
>> s1 = 'a345fg';
>> s2 = 'a345fg';
>> strcmp(s1, s2)
   ans = 1
```

Բացի **strcmp** ֆունկցիայից, **MatLab**–ում սահմանված է նաև **strncmp(s1, s2, n)** ֆունկցիան, որը թույլ է տալիս համեմատել *s*1 և *s*2 տողերի առաջին *n* սիմվոլները՝ վերադարձնելով 1, եթե դրանք համընկնում են և 0՝ հակառակ դեպքում:

Օրինակ.

```
>> s1 = 'This is an Example of String';
>> s2 = 'This is an';
>> strncmp(s1, s2, 10)
   ans = 1
```

Հաճախ անհրաժեշտ է լինում *s* տողի մեջ որևէ *s*1 բառ փոխարինել այլ՝ *s*2 բառով: Դրա համար սահմանված է **strrep(s, s1, s2)** ֆունկցիան:

Օրինակ.

```
>> s = 'This is an Example of String';
>> s1 = 'String';
>> s2 = 'Text Variable.';
>> s = strrep(s, s1, s2)
   s = This is an Example of Text Variable.
```

eval(s) ֆունկցիան թույլ է տալիս *s* տողն իրականացնել որպես **MatLab**–ի արտահայտություն:

Օրինակ.

```
>> s = '7^2 + 6*sin(pi/4)'
   s = 7^2 + 6*sin(pi/4)
>> k = eval(s)
   k = 53.2426
```

Բացի թվարկված ֆունկցիաներից, **MatLab**–ում սահմանված են տողերի հետ աշխատելու համար նախատեսված այլ ֆունկցիաներ ևս, որոնց մի մասը նկարագրված է հետևյալ աղյուսակում:

Ֆունկցիան	**Նկարագրությունը**	**Օրինակ**

ischar(s)	վերադարձնում է 1, եթե *s*–ը սիմվոլային զանգված է, 0՝ հակառակ դեպքում,	`>> s1 = 'Example of string';`
		`>> s2 = 745;`
		`>> ischar(s1)`
		`ans = 1`
		`>> ischar(s2)`

isletter(s)	վերադարձնում է *s* սիմվոլային զանգվածի չափի զանգված, որի տարրերը հավասար են 1–ի, եթե *s*–ի համապատասխան ինդեքսով սիմվոլն այբուբենի տառ է, 0՝ հակառակ դեպքում,	`>> s = 'a4b5ky';`
		`>> isletter(s)`
		`ans = 1 0 1 0 1 1`

isspace(s)	վերադարձնում է *s* սիմվոլային զանգվածի չափի զանգված, որի տարրերը հավասար են 1–ի, եթե *s*–ի համապատասխան ինդեքսով սիմվոլը բացատ է, 0՝ հակառակ դեպքում,	`>> s = 'a 45 lk'`
		`>> isspace(s)`
		`ans = 0 1 0 0 1 0 0`

lower(s)	*s* սիմվոլային զանգվածի բոլոր մեծատառերի փոխարեն վերադարձնում է դրանց համապատասխան փոքրատառերը, իսկ մնացած սիմվոլները թողնում է անփոփոխ,	`>> s = 'Example of String.';`
		`>> lower(s)`
		`ans =`
		`example of string.`

upper(s)	*s* սիմվոլային զանգվածի բոլոր փոքրատառերի փոխարեն վերադարձնում է դրանց համապատասխան մեծատառերը, իսկ մնացած սիմվոլները թողնում է անփոփոխ,	`>> s = 'Example of String.';`
		`>> upper(s)`
		`ans =`
		`EXAMPLE OF STRING.`

num2str(x)	*x* թվային զանգվածը փոխարինում է նույն սիմվոլներից բաղկացած տողով: Այս ֆունկցիան հարմար է **xlabel**, **ylabel**, **title**, **gtext** ֆունկցիաներում օգտագործելու համար:	

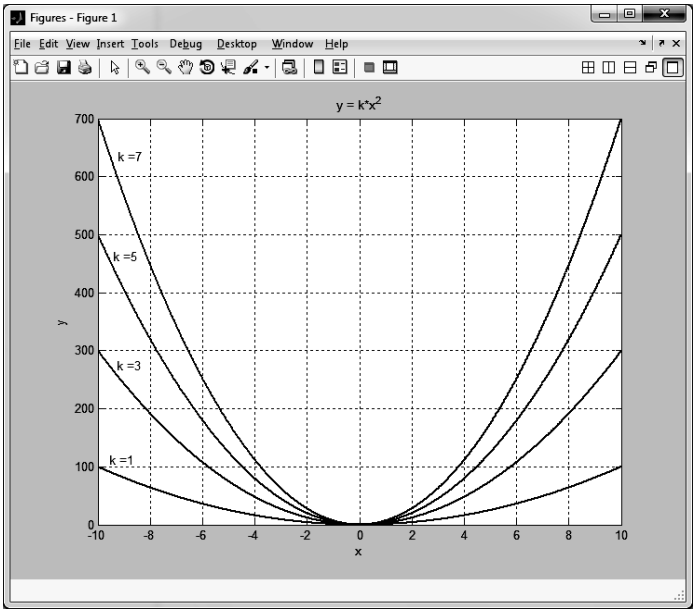
Օրինակ.

$-10 \le x \le 10$ միջակայքում կառուցենք $y = kx^2$ ֆունկցիայի գրաֆիկները $k = 1, 3, 5, 7$ արժեքների համար և յուրաքանչյուր կորի մոտ ավելացնենք տեքստ՝ նշելով k գործակցի համապատասխան արժեքը:

Խնդիրը լուծելու համար կօգտվենք **gtext** ֆունկցիայից, որը գրաֆիկը կառուցելուց հետո թույլ է տալիս տեքստ ավելացնել գրաֆիկական պատուհանում: Ինչպես արդեն

գիտենք, **gtext** ֆունկցիայի արգումենտը տող տիպի փոփոխական է: Քանի որ մեր օրինակում ստացված կորերից յուրաքանչյուրը կառուցվում է *k*–ի տարբեր արժեքների համար, **gtext**–ի արգումենտում մեկ հաստատուն տող գրելով՝ նպատակին չենք հասնի: Խնդիրը լուծվում է **strcat** ֆունկցիայի օգնությամբ, որի առաջին արգումենտում կգրենք պարզապես *'k = '* տողը, իսկ երկրորդ արգումենտում պետք է լինի *k* փոփոխականի համապատասխան արժեքը: Այդ արժեքը որպես տող կգրենք **strcat** ֆունկցիայի երկրորդ արգումենտում՝ օգտվելով թիվը տողի փոխարինող **num2str** ֆունկցիայից: Այսինքն՝ **gtext** ֆունկցիան կունենա **gtext(strcat('k = ', num2str(k)))** տեսքը: Ծրագրի արդյունքը բերված է նկ. 40–ում:

```
x = [-10:0.01:10];

for k = 1:2:7
    y = k*x.^2;
    plot(x,y)
    gtext(strcat('k = ', num2str(k)))
    hold on
    grid on
end
```



Նկ. 40

6.8. ԱՇԽԱՏԱՆՔ ՖԱՅԼԵՐԻ ՀԵՏ

Եթե աշխատում ենք քիչ թվով տվյալների հետ, ապա դրանք ստեղնաշարից մուտքագրելը շատ հարմար է: Սակայն երբ տվյալների քանակը մեծ է, ապա սովորական մուտքագրումը խիստ անհարմար է: Ավելին, հաճախ անհրաժեշտ է

լինում միևնույն տվյալներն օգտագործել տարբեր ծրագրերում: Նման դեպքերում, ստեղնաշարից հերթով տվյալները մուտքագրելու փոխարեն շատ ավելի հարմար է տվյալները պահել որևէ տեքստային ֆայլում և ծրագրի մեջ կանչել այդ ֆայլը: Այդպիսով՝ ֆայլի ներսում գրված տվյալները հասանելի են դառնում ծրագրին ճիշտ այնպես, ինչպես եթե դրանք մուտքագրված լինեին ստեղնաշարից: Բացի տվյալները ֆայլից կարդալուց, հնարավոր է նաև ծրագրի աշխատանքի արդյունքում ստացված արդյունքները ևս պահպանել տեքստային ֆայլի մեջ և հետագայում օգտագործել: Ինչպես ծրագրավորման այլ լեզուներ, **MatLab**–ում ևս հնարավոր է տեքստային կամ այլ ֆայլերից տվյալներ կարդալ ու դրանցում տվյալներ գրելը: Այս բաժինը նվիրված է **MatLab**–ում ֆայլերի հետ աշխատանքին:

Տեքստային ֆայլերի հետ աշխատելիս (ֆայլից տվյալներ կարդալիս կամ ֆայլի մեջ տվյալներ գրելիս) անհրաժեշտ է կատարել երեք գործողություն՝ բացել այդ ֆայլը, կատարել ֆայլից կարդալու կամ ֆայլի մեջ գրելու գործողություններ և փակել ֆայլը: Ծանոթանանք այս գործողություններից յուրաքանչյուրին:

6.8.1. Տեքստային ֆայլի բացելն ու փակելը

Նախքան տեքստային ֆայլի հետ որևէ գործողություն կատարելը անհրաժեշտ է բացել այդ ֆայլը: Տեքստային ֆայլը բացելու համար **MatLab**–ում նախատեսված է **fopen** ֆունկցիան, որի գրելաձևը հետևյալն է.

fid = fopen('ֆայլը', 'ֆայլին դիմելու ձևը')

կամ

[fid, message] = fopen('ֆայլը', 'ֆայլին դիմելու ձևը')։

fopen ֆունկցիան ունի երկու մուտքային արգումենտ: Այդ արգումենտները տեքստային են, այսինքն՝ դրանք պետք է գրվեն ապոստրոֆների մեջ: Առաջին արգումենտը ֆայլի անվանումն է: Սակայն այստեղ կա սխալվելու հավանականություն: Եթե գրենք միայն ֆայլի անունը (օրինակ՝ 'MyFile.txt'), ապա **MatLab**–ն այն կփնտրի ընթացիկ թղթապանակում, և եթե ֆայլը այդ թղթապանակում չգտնվի, ապա չի գտնի այն: Այդ պատճառով ավելի նախընտրելի է առաջին արգումենտում գրել ֆայլի գտնվելու լրիվ ճանապարհը, այլ ոչ թե միայն անվանումը (օրինակ՝ 'D:\Work\MyFile.txt'):

fopen ֆունկցիայի երկրորդ արգումենտում նշվում է ֆայլին դիմելու ձևը: Այն կարող է ընդունել հետևյալ աղյուսակում բերված արժեքները.

Ֆայլին դիմելու ձևը	Նկարագրությունը
'r'	Ֆայլը բացվում է՝ նրանից տվյալներ կարդալու համար:
'w'	Ֆայլը բացվում է տվյալներ գրելու համար: Եթե նշված անունով ֆայլ համակարգչում գոյություն չունի, ստեղծվում է այդ անունով նոր ֆայլ: Եթե այդ անունով ֆայլ համակարգչում արդեն կար, ֆայլի պարունակությունը մաքրվում է, դրանից հետո միայն նոր տվյալները գրվում են դրա մեջ:
'a'	Նախորդ դեպքի պես ֆայլը բացվում է՝ նրանում տվյալներ գրելու համար: Եթե նշված անունով ֆայլ համակարգչում գոյություն չունի, ստեղծվում է այդ անունով նոր ֆայլ: Եթե այդ անունով ֆայլ համակարգչում արդեն գոյություն ուներ, ֆայլի նախկին պարունակությունը չի մաքրվում, և նոր տվյալները գրվում են հին տվյալներից հետո:
'r+'	Ֆայլը բացվում է՝ նրանից տվյալներ կարդալու և նրանում տվյալներ գրելու համար:
'w+'	Ֆայլը բացվում է՝ նրանում տվյալներ գրելու կամ նրանից տվյալներ կարդալու համար: Եթե նշված անունով ֆայլ համակարգչում գոյություն չունի, ստեղծվում է այդ անունով նոր ֆայլ: Եթե այդ անունով ֆայլ համակարգչում արդեն գոյություն ուներ, ֆայլի պարունակությունը մաքրվում է, դրանից հետո միայն գրվում են նոր տվյալները:
'a+'	Նախորդ դեպքի նման ֆայլը բացվում է՝ մեջը տվյալներ գրելու կամ դրանից տվյալներ կարդալու համար: Եթե նշված անունով ֆայլ համակարգչում չկա, ստեղծվում է այդ անունով նոր ֆայլ: Եթե այդ անունով ֆայլ համակարգչում արդեն գոյություն ունի, ֆայլի նախկին պարունակությունը չի մաքրվում, և նոր տվյալները գրվում են հին տվյալներից հետո:

fopen ֆունկցիայի հայտարարությունից երևում է, որ այն կարող է վերադարձնել մեկ կամ երկու ելքային արգումենտ: Առաջին ելքային արգումենտով (*fid*) որոշվում է՝ արդյոք ֆայլը հնարավոր է եղել բացել: Այն ընդունում է ամբողջ արժեքներ և կոչվում է ֆայլի իդենտիֆիկատոր: Եթե ֆայլի իդենտիֆիկատորը հավասար է լինում (–1)–ի, նշանակում է, որ ֆայլը բացել չի հաջողվել: Երկրորդ ելքային արգումենտը (*message*) տողային փոփոխական է, որը հաղորդագրություն է տալիս՝ արդյոք ֆայլը հնարավոր է եղել բացել: Եթե ֆայլը հաջողվել է նորմալ բացել, այն վերադարձնում է դատարկ տող:

Օրինակ.

Ենթադրենք՝ MyTextFile.txt տեքստային ֆայլը գոյություն ունի և գտնվում է D:\Work թղթապանակում: Փորձենք բացել այս ֆայլը նրանից տվյալներ կարդալու համար՝ օգտվելով **fopen** ֆունկցիայից.

```
>> [FID, message] = fopen('MyTextFile.txt', 'r')
   FID = –1
   message = No such file or directory
```

fopen ֆունկցիայի երկրորդ մուտքային արգումենտը 'r' է, ինչը նշանակում է, որ ֆայլը բացում ենք՝ նրանից տվյալներ կարդալու համար: Ինչպես տեսնում ենք, ֆայլի իդենտիֆիկատորը՝ *FID*=–1, այսինքն՝ ֆայլը բացել չի հաջողվել: Նույնը տեսնում ենք նաև *message* փոփոխականի արժեքից՝ *No such file or directory* (այսպիսի ֆայլ կամ թղթապանակ գոյություն չունի): Այս օրինակում սխալի պատճառն այն է, որ մենք չենք նշել ֆայլի գտնվելու լրիվ ճանապարհը (D:\Work\MyTextFile.txt), այլ միայն ֆայլի անունն ու ընդլայնումը, իսկ **MatLab**–ը ֆայլը փնտրում է իր ընթացիկ թղթապանակում, որը չի համապատասխանում ֆայլի գտնվելու թղթապանակի հետ:

Այժմ գրենք նույն հրամանը՝ այս անգամ նշելով ֆայլի գտնվելու լրիվ ճանապարհը.

```
>> [fid, message] = fopen('D:\Work\MyTextFile.txt', 'r')
   fid = 8
   message = ''
```

Ինչպես տեսնում ենք, ֆայլը հաջողվեց բացել, ինչի մասին վկայում են ֆայլի իդենտիֆիկատորի՝ (–1)–ից տարբեր արժեքը (*fid*=8) և *message* արգումենտի՝ դատարկ տող լինելը: Այժմ ֆայլը բացված է, և նրանից կարող ենք տվյալներ կարդալ:

Տեքստային ֆայլը բացելուց և իր հետ գործողություններ կատարելուց հետո անպայման այն պետք է փակել: Ֆայլը փակելու համար նախատեսված է **fclose** ֆունկցիան, որը որպես մուտքային արգումենտ ստանում է ֆայլի իդենտիֆիկատորի արժեքը: **fclose** ֆունկցիան ելքում վերադարձնում է 0, եթե ֆայլը հաջողվել է նորմալ փակել, (–1)՝ հակառակ դեպքում:

Օրինակ.

Նախորդ օրինակում բացված ֆայլը փակելու համար անհրաժեշտ է գրել հետևյալ հրամանը.

```
>> status = fclose(fid)
   status = 0
```

Ինչպես տեսնում ենք, **fclose** ֆունկցիայի ելքային *status* փոփոխականը ստացավ 0 արժեքը, ինչը նշանակում է, որ ֆայլը հաջողվեց նորմալ փակել:

6.8.2. Տեքստային ֆայլից կարդալը

fopen ֆունկցիայի օգնությամբ ֆայլը բացելուց հետո այդ ֆայլի տվյալները կարելի է տող առ տող կարդալ **fgetl** ֆունկցիայի օգնությամբ: Ամեն անգամ կանչելիս **fgetl** ֆունկցիան ֆայլից կարդում է հերթական տողը. առաջին անգամ կանչելիս առաջին տողը, երկրորդ անգամ՝ երկրորդ տողը և այլն: **fgetl** ֆունկցիան ունի հետևյալ գրելաձևը.

line = fgetl(fid)

Որպես մուտքային արգումենտ **fgetl** ֆունկցիան ստանում է ֆայլի իդենտիֆիկատորը, իսկ ելքային line փոփոխականը տեքստային փոփոխական է, որը պարունակում է ֆայլի ընթացիկ տողը:

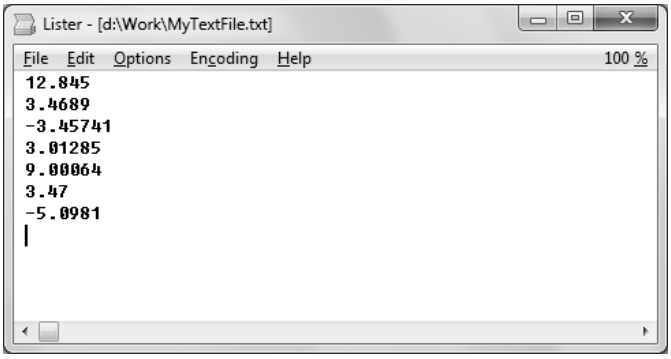
Այսպիսով, որպեսզի ֆայլն ամբողջությամբ տող առ տող կարդանք, անհրաժեշտ է **fgetl** ֆունկցիան կանչել այնքան անգամ, որքան տող պարունակվում է ֆայլի մեջ: Հարմար է **fgetl** ֆունկցիան գրել **while** ցիկլի մեջ: Ցիկլի պայմանը պետք է ճիշտ լինի այնքան ժամանակ, քանի դեռ ֆայլի տողերը լրիվ չեն կարդացվել: Ֆայլի տողերի ավարտի մասին կարող ենք իմանալ **MatLab**–ում ներդրված **feof** ֆունկցիայի օգնությամբ, որը որպես մուտքային արգումենտ ստանում է ֆայլի իդենտիֆիկատորը, իսկ ելքում վերադարձնում է 1, եթե ֆայլի տողերն ավարտվել են և 0՝ հակառակ դեպքում: Այսպիսով՝

```
while(~feof(fid))
   line = fgetl(fid)
end
```

հայտարարությամբ կարող ենք կարդալ ֆայլի բոլոր տողերը: **while** ցիկլի պայմանը կարելի է գրել նաև **while(feof(fid)==0)**, որը համարժեք է վերևում գրվածին:

Օրինակ.

Դարձյալ դիտարկենք MyTextFile.txt ֆայլը, որը պահպանված է D:\Work թղթապանակում: Այս ֆայլը պարունակում է նկ. 41–ում բերված տվյալները: Գրենք ծրագիր, որը թույլ կտա բացել այդ ֆայլը, կարդալ այդ ֆայլի տվյալները և փակել այն:



Նկ. 41

```
fid = fopen('D:\Work\MyTextFile.txt', 'r');
if fid~=-1
   LineArray = [];
   while ~feof(fid)
      line = fgetl(fid);
      LineArray = [LineArray; str2num(line)];
   end
else
   disp('No such file.')
end
LineArray
fclose(fid);
```

Այս ծրագրում նախ բացվում է անհրաժեշտ ֆայլը, որի իդենտիֆիկատորը վերագրվում է *fid* փոփոխականին: Ուշադրություն դարձնենք, որ **fopen** ֆունկցիայի երկրորդ ելքային արգումենտը չենք գրել, քանի որ այստեղ բավական է միայն *fid*–ի արժեքը ստուգելը: **if** պայմանի օպերատորի օգնությամբ նախ ստուգում ենք՝ արդյոք ֆայլը հաջողվել է նորմալ բացել: Եթե fid փոփոխականը հավասար չէ (–1)-ի, նշանակում է ֆայլը բացված է: Եթե *fid*=–1, այդ դեպքում ծրագիրը **disp** ֆունկցիայի օգնությամբ պետք է էկրանին արտածի "*No such file*" տողը:

Եթե ֆայլը հաջողվել է նորմալ բացել, ապա նախ հայտարարվում է *LineArray* դատարկ վեկտորը, որի մեջ հետագայում պետք է գրվեն ֆայլից կարդացված տողերը: Այնուհետև գրված է **while** ցիկլի օպերատորը, որի ներսում գրված հրամանները կկատարվեն այնքան ժամանակ, քանի դեռ ֆայլի բոլոր տողերը չեն կարդացվել: Յուրաքանչյուր տողը կարդացվում է **line=fgetl(fid)** հրամանի օգնությամբ: *line* փոփոխականը տեքստային է, և որպեսզի այն դարձնենք թիվ, հաջորդ հրամանում օգտվել ենք **str2num** ֆունկցիայից, որը որպես մուտքային արգումենտ ստանում է տող և այն դարձնում իրական թիվ: Կարդացված տողերից յուրաքանչյուրն, այսպիսով, փոխակերպվում է իրական թվի և միացվում *LineArray* սյուն–վեկտորին: Վերջում **MatLab**–ի էկրանին կպատկերվեն կարդացված տվյալները *LineArray* հրամանի օգնությամբ և կփակվի ֆայլը:

Ծրագրի իրականացման արդյունքում կստանանք հետևյալ պատասխանը.

```
LineArray =
   12.8450
    3.4689
   -3.4574
    3.0128
    9.0006
    3.4700
   -5.0981
```

Սա նշանակում է, որ ֆայլի բոլոր տվյալները հաջողությամբ կարդացվել են (համեմատելու համար տես նկ. 41):

6.8.3. Տեքստային ֆայլի մեջ գրելը

Տեքստային ֆայլից կարելի է ոչ միայն տվյալներ կարդալ, այլև կարելի է դրանց մեջ գրել: Ֆայլի մեջ տեքստ կարելի է ներմուծել **fprintf** ֆունկցիայի օգնությամբ, որն ունի հետևյալ գրելաձևը.

fprintf(fid, 'տեքստ')

Այս հրամանի օգնությամբ *fid* իդենտիֆիկատորով ֆայլի մեջ գրվում է երկրորդ արգումենտում գրված տեքստը: Ակնհայտ է, որ որպես երկրորդ փոփոխական կարող է լինել ոչ միայն տեքստ՝ գրված ապոստրոֆներում, այլև նախօրոք սահմանված որևէ տեքստային փոփոխական:

Եթե միևնույն ֆայլի մեջ երկրորդ անգամ տեքստ ավելացնենք (երկրորդ անգամ գրենք **fprint** ֆունկցիան), ապա նոր տեքստը կգրվի ոչ թե նոր տողից, այլ առաջին տեքստից հետո որպես շարունակություն:

Օրինակ.

```
>> fid = fopen('D:\Work\MyTextFile.txt', 'w');
>> fprintf(fid, 'This is an example of text.')
   ans = 27
>> fprintf(fid, 'This is an example of another text.')
   ans = 35
>> fclose(fid);
```

Այս օրինակում նախ MyTextFile.txt ֆայլը բացվում է՝ նրանում տվյալներ գրելու համար (**fopen** ֆունկցիայի երրորդ արգումենտում գրված է 'w'): Այնուհետև երկու տեքստ է գրվում ֆայլի մեջ՝ '*This is an example of text.*' և '*This is an example of another text.*': Նախ նկատենք, որ **fprint** ֆունկցիան որպես ելք վերադարձնում է այն սիմվոլների քանակը, որոնք պարունակվում են տեքստերի մեջ՝ համապատասխանաբար 27 և 35 առաջին և երկրորդ դեպքերում: Եթե այժմ որևէ տեքստային խմբագրիչով բացենք MyTextFile.txt ֆայլը, ապա կտեսնենք, որ դրա մեջ գրվել է հետևյալ տեքստը.

This is an example of text.This is an example of another text.

Այսպիսով, եթե հատուկ չնշենք, **MatLab**–ը ֆայլի մեջ **fprintf** ֆունկցիայով տեքստերը գրում է իրար կողքի: Որպեսզի կարողանանք երկրորդ տեքստը գրել նոր տողից, անհրաժեշտ է առաջին տողի վերջում կամ երկրորդ տողի սկզբում գրել նոր տողի '\n' հատուկ սիմվոլը: Այսինքն՝ նախորդ օրինակը պետք է ձևափոխել հետևյալ կերպ.

```
>> fid = fopen('D:\Work\MyTextFile.txt', 'w');
>> fprintf(fid, 'This is an example of text.\n')
>> fprintf(fid, 'This is an example of another text.')
>> fclose(fid);
```

կամ

```
>> fid = fopen('D:\Work\MyTextFile.txt', 'w');
>> fprintf(fid, 'This is an example of text.')
>> fprintf(fid, '\nThis is an example of another text.')
>> fclose(fid);
```

Երկու դեպքում էլ տեքստային ֆայլի պարունակությունը կլինի.

This is an example of text.
This is an example of another text.

Բացի նոր տողի '\n' սիմվոլից **MatLab**–ում կան նաև այլ հատուկ սիմվոլներ: Դրանց ցանկը բերված է հետևյալ աղյուսակում.

Հատուկ սիմվոլը	**Նշանակությունը**
\n	նոր տող
\t	հորիզոնական տաբուլյացիա (tab)
\b	մեկ սիմվոլի չափով ետ է գալիս (backspace)
\f	նոր էջ
\\	'\' սիմվոլը
\" կամ "	ապոստրոֆ
%%	տոկոսի նշան

Այսպիսով՝ մենք տեսանք, թե սովորական տեքստն ինչպես կարելի է ավելացնել տեքստային ֆայլի մեջ: Բացի սովորական տեքստից, շատ հաճախ անհրաժեշտ է լինում տեքստի փոխարեն ֆայլի մեջ գրել թվեր, որոնք ստացվել են **MatLab**–ում որոշակի հաշվարկների արդյունքում: Ավելին, երբեմն հարկ է լինում տեքստն ու թվերն իրար հետ ավելացնել ֆայլի մեջ: Թվերը ֆայլի մեջ գրելու համար անհրաժեշտ է հստակ նշել, թե ինչ ֆորմատով են դրանք գրվելու ֆայլում: Տեսնենք, թե թվերի ֆորմատն ինչպես կարելի է նշել **fprintf** ֆունկցիայի մեջ:

Ենթադրենք՝ fid իդենտիֆիկատորով ֆայլում պետք է գրանցենք թվային փոփոխականներ: Այս դեպքում **fprintf** ֆունկցիայի գրելաձևը հետևյալն է.

fprintf(fid, 'ֆորմատ', թվային փոփոխականները);

Եթե, օրինակ, ֆայլի մեջ պետք է գրվի որևէ տեքստ (տեքստ1), այնուհետև՝ թիվ, դրանից հետո՝ այլ տեքստ (տեքստ2) և մեկ ուրիշ թիվ, ապա **fprintf** ֆունկցիայի գրելաձևը կլինի այսպիսին.

fprintf(fid, 'տեքստ1 ֆորմատ տեքստ2 ֆորմատ', թվային փոփոխականները)

fprintf ֆունկցիայի կիրառություններն ամենից հարմար է նկարագրել օրինակներով:

Օրինակ.

Ենթադրենք՝ $x = \frac{\pi}{8}$ արժեքի համար անհրաժեշտ է որոշել $y = \sin x$ ֆունկցիայի արժեքն ու արդյունքը գրել D:\Work\FormatExample.txt տեքստային ֆայլում: Ընդ որում, $y = \sin x$ ֆունկցիայի արժեքը կգրենք ստորակետից հետո 4 նիշի ճշտությամբ իրական թվի տեսքով: Ծրագիրը կունենա այսպիսի տեսք.

```
fid = fopen('D:\Work\FormatExample.txt', 'w');
x = pi/8;
y = sin(x);
fprintf(fid, '%-6.4f', y);
fclose(fid);
```

Ծրագրի առաջին տողում բացվում է անհրաժեշտ տեքստային ֆայլը՝ նրանում տվյալներ գրելու համար: Այնուհետև սահմանվում է x փոփոխականը և y ֆունկցիան: Ծրագրի 4–րդ տողը y ֆունկցիայի արժեքը բացված տեքստային ֆայլում գրելու հրամանն է **fprintf** ֆունկցիայի օգնությամբ: Այս ֆունկցիան որպես առաջին արգումենտ ստանում է բացված ֆայլի *fid* իդենտիֆիկատորը, երկրորդ արգումենտում գրվում է այն ֆորմատը, որով ցանկանում ենք թիվը գրել ֆայլի մեջ, իսկ երրորդ արգումենտում՝ այն թվային փոփոխականը, որի արժեքը պետք է գրվի ֆայլում երկրորդ արգումենտում նշված ֆորմատով:

Մանրամասն դիտարկենք երկրորդ արգումենտում գրված ֆորմատի գրելաձևը: **fprintf** ֆունկցիայում օգտագործվող ցանկացած թվային ֆորմատ պետք է գրվի որպես տեքստային փոփոխական (գրվի ապոստրոֆներում) և սկսվի տոկոսի (%) նշանով: Այս օրինակում տոկոսի նշանին հաջորդում է մինուսի նշանը, որը ցույց է տալիս, որ տեքստային փոփոխականում թվի համար նախատեսված դաշտի մեջ թիվը պետք է գրվի ձախ կողմից (left–justified): Մինուսի նշանին հաջորդում է 6.4 թիվը: Այստեղ 6 թիվը նշանակում է, որ թվի համար նախատեսված դաշտի երկարությունը 6–ն է, այսինքն՝ թիվը ֆայլի մեջ կարող է գրվել առավելագույնը 6 նիշով, իսկ կետին հաջորդող 4 թիվը նշանակում է, որ թվի կոտորակային մասը բաղկացած է 4 նիշից: Ֆորմատի ամենավերջում գրված f տառը նշանակում է, որ թիվը պետք է գրվի իրական թվի տեսքով: Այսպիսով, ծրագրի իրականացման արդյունքում $y = \sin x$ ֆունկցիայի արժեքը տեքստային ֆայլում կունենա 0.3827 տեսքը:

Այժմ դիտարկենք այն դեպքը, երբ տեքստային ֆայլի մեջ մեկ անգամ **fprintf** ֆունկցիան օգտագործելով՝ ցանկանում ենք գրել ոչ միայն թիվ, այլև որևէ տեքստ: Դարձյալ դիտարկենք քննարկված օրինակը, միայն թե այս անգամ ծրագիրը ձևափոխենք այնպես, որ տեքստային ֆայլի մեջ գրվի ոչ թե միայն 0.3827 թիվը, այլ *The value of sin(x) is equal to 0.3827* տողը: Այս դեպքում **fprintf** ֆունկցիան կգրվի հետևյալ կերպ.

```
fprintf(fid, 'The value of sin(x) is equal to %–6.4f ', y);
```

Ինչպես տեսնում ենք, **fprintf** ֆունկցիայի երկրորդ արգումենտում կարելի է գրել ցանկացած տեքստ, սակայն թվի արժեքը գրելիս պետք է անպայման նշել, թե ինչ ֆորմատով է այն գրվելու ֆայլում: Թվային փոփոխականի անունը դարձյալ գրվում է **fprintf** ֆունկցիայի երրորդ արգումենտում:

Վերջապես, դիտարկենք այն դեպքը, երբ մեկ անգամ **fprintf** ֆունկցիան օգտագործելով ֆայլի մեջ անհրաժեշտ է գրել ոչ թե մեկ, այլ մի քանի թվային փոփոխականի արժեք: Ենթադրենք, բացի $y = \sin x$ ֆունկցիայի արժեքից, ֆայլի հաջորդ տողում ցանկանում ենք գրել $z = \cos x$ ֆունկցիայի արժեքը այնպես, որ ֆայլի պարունակությունն ունենա հետևյալ տեսքը.

```
y = 0.3827
z = 0.92388
```

Ծրագիրն այս դեպքում կընդունի հետևյալ տեսքը.

```
fid = fopen('D:\Work\FormatExample.txt', 'w');
x = pi/8;
y = sin(x);
z = cos(x);
fprintf(fid, 'y = %–6.4f\nz = %–7.5f', y, z);
fclose(fid);
```

Ինչպես տեսնում ենք, **fprintf** ֆունկցիայի երկրորդ արգումենտում գրվել են "y = " և "z = "տեքստերը, նոր տողի "\n" սիմվոլը և անհրաժեշտ երկու թվային փոփոխականների ֆորմատները: Եթե ֆայլի մեջ պետք է գրվեն երկու կամ ավելի թվեր, ապա դրանցից յուրաքանչյուրի ֆորմատը պետք է գրել առանձին **fprint** ֆունկցիայի երկրորդ արգումենտում: Յուրաքանչյուր տոկոսի սիմվոլ **MatLab**–ը հասկանում է որպես հերթական թվային փոփոխականի ֆորմատի սկիզբ: **fprintf** ֆունկցիայի երրորդ և չորրորդ արգումենտներում գրված են այն թվային փոփոխականների անուն-

ները, որոնք անհրաժեշտ է գրել ֆայլի մեջ, ընդ որում, անունների հերթականությունը և փոփոխականների քանակը պետք է համընկնի երկրորդ արգումենտում ֆորմատների հերթականության և քանակի հետ:

Այսպիսով, *y* թվային փոփոխականի համար նախատեսված է 6 նիշից բաղկացած դաշտ, որից 4 նիշը նախատեսված է կոտորակային մասի համար, իսկ *z* փոփոխականի համար՝ 7 նիշից բաղկացած դաշտ, որից 5 նիշը՝ կոտորակային մասի համար:

Այժմ ավելի մանրամասն ծանոթանանք թվերի ֆորմատի գրելաձևի հետ: Ինչպես տեսանք, ցանկացած ֆորմատ անպայման պետք է սկսվի տոկոսի (%) նշանով: Դրան հաջորդում է, այսպես կոչված, դրոշակը (flag), ընդ որում, այն գրելը պարտադիր չէ: Այն կարող է լինել հետևյալ սիմվոլներից մեկը.

Սիմվոլը	**Նշանակությունը**
–	թվի համար նախատեսված դաշտում այն գրվում է ձախից (left–justified)
+	թվի առջևից գրվում է թվի նշանը (+ կամ –)
0	եթե թիվն ավելի կարճ է, քան իր համար նախատեսված դաշտը, թիվը ձախից լրացվում է զրոներով

Դրոշակին հաջորդում է թվի համար նախատեսված դաշտի երկարությունը, կետի նշանը և թվի կոտորակային մասի նիշերի քանակը (վերևի օրինակներում *y* փոփոխականի համար գրված էր 6.4, իսկ *z* փոփոխականի համար՝ 7.5): Դաշտի երկարությունը և կոտորակային մասի նիշերի քանակը գրելը ևս պարտադիր չէ: Դաշտի երկարությունն այն փոքրագույն թվանշանների քանակն է, որն անհրաժեշտ է գրել ֆայլի մեջ: Եթե դաշտն ավելի մեծ է տրված, քան թվի թվանշանների քանակն է, այն ձախից լրացվում է բացատներով կամ զրոներով:

Ֆորմատի վերջին սիմվոլը թիվը գրելու ֆորմատն է: Այն կարող է լինել հետևյալ սիմվոլներից մեկը.

Սիմվոլը	**Նշանակությունը**
e	փոքրատառ e տառով էքսպոնենցիալ գրելաձև (օրինակ՝ 3.64e+002)
E	մեծատառ E տառով էքսպոնենցիալ գրելաձև (օրինակ՝ 3.64E+002)
f	թիվը ներկայացնում է իրական թվի տեսքով (օրինակ՝ 34.1523)
g	ընտրում է e և f ֆորմատներից ավելի կարճը
G	ընտրում է E և f ֆորմատներից ավելի կարճը
i	թիվը ներկայացնում է ամբողջ թվի տեսքով

fprintf ֆունկցիան կարելի է օգտագործել ոչ միայն տեքստը կամ թիվը ֆայլի մեջ գրելու համար, այլև էկրանին արտածելու համար: Այս դեպքում պարզապես չպետք է գրվի առաջին արգումենտը՝ ֆայլի իդենտիֆիկատորը:

6.8.4. save և load հրամանները

MatLab ծրագիրը փակելուց հետո վերջինիս հիշողությունից կորում են աշխատանքի ընթացքում սահմանված բոլոր փոփոխականները: Դրանք վերականգնելու համար, անհրաժեշտ է կրկին աշխատացնել այն բոլոր հրամանները, որոնցով դրանք հայտարարվել էին: Խնդիրն էապես հեշտանում է, եթե ծրագիրը գրված է M–ֆայլում. պարզապես պետք է աշխատացնել այդ ֆայլը: Սակայն եթե ծրագիրը գրվել էր հրամանի պատուհանում, ապա, ինչպես գիտենք, ծրագրի բոլոր հրամանները պետք է հերթով իրականացվեն: Որպեսզի յուրաքանչյուր անգամ **MatLab**–ը բացելուց հետո ստիպված չլինենք այս գործողություններն ամեն անգամ կրկնել, կարելի է օգտվել ներդրված **save** և **load** հրամաններից:

save հրամանը թույլ է տալիս առանձին ֆայլի մեջ պահպանել սահմանված փոփոխականները: Ընդ որում, ֆայլի մեջ կարելի է պահպանել ինչպես սահմանված բոլոր փոփոխականները, այնպես էլ դրանցից միայն մի քանիսը: **save** հրամանն ունի հետևյալ գրելաձևը.

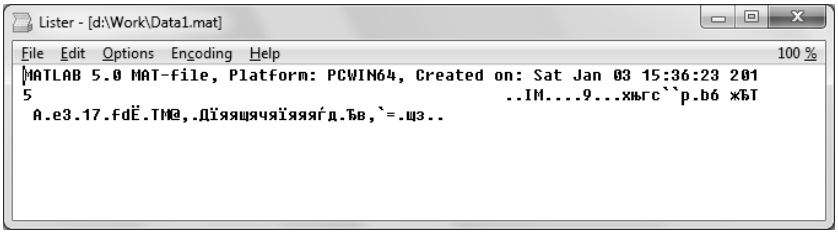
save ֆայլի անուն կամ *save('ֆայլի անուն')*

Այս հրամանի օգնությամբ աշխատանքային միջավայրում երևացող բոլոր փոփոխականները կպահպանվեն **save** հրամանի կողքին գրված ֆայլի մեջ: Ստեղծված ֆայլը երկուական համակարգով ներկայացված ֆայլ է և ունի *.mat ընդլայնումը: Այն նախատեսված է միայն **MatLab** ծրագրի համար, և այլ ծրագրերի համար ընթեռնելի չէ: Այս ֆայլի մեջ երկուական համակարգում գրվում են յուրաքանչյուր փոփոխականի անունը, չափը, տիպը և արժեքը:

Օրինակ.

```
>> x = [–5:5];
>> save D:\Work\Data1
```

Այս օրինակում սահմանվեց *x* փոփոխականը և պահպանվեց Data1.mat ֆայլի մեջ: Եթե այժմ ցանկանանք բացել այս ֆայլը որևէ տեքստային խմբագրիչով, ապա կտեսնենք, որ իր պարունակությունն ունի նկ. 42–ում բերված տեսքը.



Նկ. 42

Ֆայլը կարելի է պահպանել այնպես, որ այն հնարավոր լինի կարդալ նաև այլ ծրագրերով: **MatLab**–ը թույլ է տալիս այն պահպանել ASCII ֆորմատով: Սակայն այս դեպքում ֆայլի մեջ պահպանվում են միայն սահմանված փոփոխականների արժեքները՝ իրարից բացատներով անջատված: Փոփոխականների անունը, չափը և տիպը չեն պահպանվում: Այս դեպքում **save** հրամանն անհրաժեշտ է գրել հետևյալ կերպ.

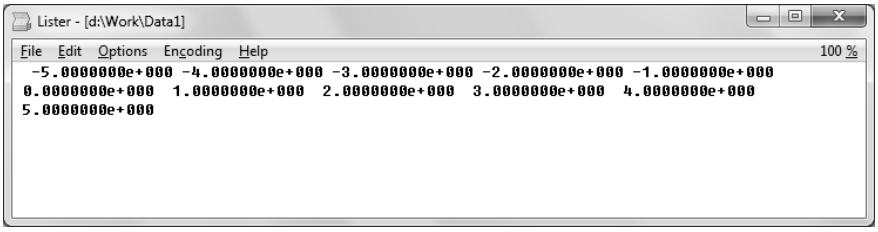
save –ascii ֆայլի անուն

Օրինակ.

Նախորդ օրինակում սահմանված *x* փոփոխականը պահպանենք այս ֆորմատով.

```
>> save –ascii D:\Work\Data1
```

Այս դեպքում ստեղծված ֆայլը բացելով որևէ տեքստային խմբագրիչով՝ կտեսնենք, որ այն ունի նկ. 43–ում բերված տեսքը.



Նկ. 43

save հրամանի օգնությամբ *.mat ընդլայնումով ֆայլում պահպանված փոփոխականները կարելի է վերականգնել **MatLab**–ի աշխատանքային միջավայրում **load** հրամանի օգնությամբ, որն ունի հետևյալ գրելաձևը.

load ֆայլի անուն կամ *load('ֆայլի անուն')*

Այս դեպքում **MatLab**–ի աշխատանքային միջավայրում վերականգնվում են ֆայլում պահպանված բոլոր փոփոխականները: Եթե անհրաժեշտ է վերականգնել ֆայլում պահպանված փոփոխականներից մի քանիսը, ապա **load** հրամանը պետք է գրել հետևյալ կերպ.

load ֆայլի անուն u1 u2 u3 կամ *load('ֆայլի անուն', 'u1', 'u2', 'u3')*

Այս գրելաձևի օգնությամբ ֆայլից կվերականգնվեն միայն *u*1, *u*2 և *u*3 անուններով փոփոխականները:

Բացի սրանից, **load** հրամանի օգնությամբ կարելի է վերականգնել նաև տեքստային (*.txt ընդլայնումով) կամ ASCII ֆորմատով պահպանված այլ ֆայլերի տվյալները: Այսպիսի ֆայլերում, ինչպես տեսանք **save** հրամանը դիտարկելիս, գրվում են միայն փոփոխականների արժեքները, այլ ոչ թե դրանց անունը, տիպը

կամ չափը: Պետք է ուշադիր լինել, որ այս դեպքում ֆայլում պահպանված տվյալները **MatLab**–ի տվյալների տեսքով լինեն: Օրինակ, եթե ֆայլում գրված է միայն մեկ թիվ, ապա **load** հրամանի օգնությամբ այդ թիվը **MatLab**–ում կներկայացվի որպես սկալյար: Եթե տվյալները գրված են մեկ սյան կամ մեկ տողի տեսքով, ապա դրանք կներկայացվեն համապատասխանաբար սյուն–վեկտորի կամ տող–վեկտորի տեսքով: Վերջապես, աղյուսակի տեսքով գրված տվյալները կներկայացվեն մատրիցի տեսքով: Անհրաժեշտ է ուշադիր լինել, որ մատրիցի դեպքում բոլոր տողերում հավասար քանակությամբ տարրեր լինեն գրված, հակառակ դեպքում **load** հրամանի օգնությամբ տվյալները չեն վերականգնվի:

6.8.5. Excel ծրագրի միջավայրից տվյալների կարդալը և գրելը

Հաճախ այն տվյալները, որոնք պետք է ուսումնասիրվեն և մշակվեն **MatLab**–ում, գրված են լինում ոչ թե տեքստային ֆայլում, այլ ուրիշ ծրագրերում: Օրինակ՝ որևէ փորձի արդյունքում ստացված տվյալները հարմար է լինում ներկայացնել աղյուսակային տեսքով՝ **Excel** ծրագրի միջավայրում: **MatLab**–ում սահմանված են ֆունկցիաներ, որոնք թույլ են տալիս կարդալ **Excel** ծրագրում գրված տվյալները կամ, հակառակը, **MatLab**–ում ստացված տվյալները գրել **Excel**–ի աղյուսակներում:

Հիշենք, որ **Excel** միջավայրում աշխատանքային գիրքը (Workbook) բաղկացած է տարբեր թերթերից (sheet), որոնցից յուրաքանչյուրն իրենից ներկայացնում է աղյուսակ՝ բաղկացած մի քանի միլիարդ վանդակներից: Աղյուսակի տողերը համարակալված են թվերով (1, 2, 3, ...), իսկ սյուները՝ տառերով (*A*, *B*, *C*, ..., *X*, *Y*, *Z*, *AA*, *AB*, *AC*, ...): Վանդակներից յուրաքանչյուրին կարելի է դիմել իր հասցեով՝ գրելով համապատասխան սյան և տողի համարը: Օրինակ՝ աղյուսակի երկրորդ սյան երրորդ տողի հասցեն *B*3–ն է: Եթե ցանկանում ենք դիմել ոչ թե մի վանդակի, այլ աղյուսակում որևէ տիրույթի, ապա պետք է գրել այդ տիրույթի առաջին և վերջին վանդակների հասցեները և դրանց միջև դնել վերջակետի (:) սիմվոլը: Օրինակ՝ *C*4:*C*11 տիրույթը կվերաբերի աղյուսակի երրորդ սյան 4–րդից 11–րդ տողերի բոլոր վանդակներին, իսկ *A*1:*D*14 տիրույթը՝ 1–ին սյան 1–ին տողից մինչև 4–րդ սյան 14–րդ տողի միջև ընկած ուղղանկյուն տիրույթի բոլոր վանդակներին:

MatLab–ում ներդրված **xlsread** ֆունկցիան նախատեսված է **Excel** միջավայրի աղյուսակներից տվյալներ կարդալու համար: Ընդհանուր դեպքում այն ունի հետևյալ գրելաձևը.

u = xlsread('ֆայլի անուն', 'թերթի անուն', 'տիրույթ')

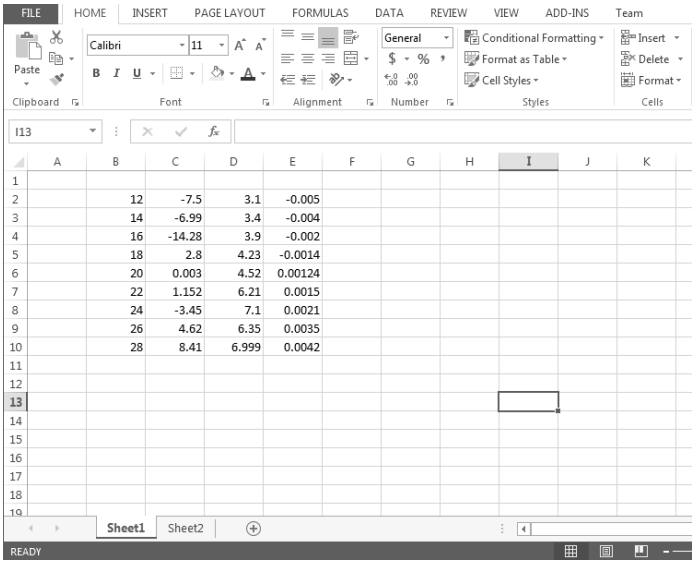
Այս ֆունկցիայի բոլոր արգումենտները գրվում են տողերի տեսքով: Առաջին արգումենտում տողի տեսքով գրվում է այն **Excel**–ի ֆայլի անունը, որից ցանկանում ենք կարդալ տվյալները, երկրորդ արգումենտում՝ **Excel**–ի աշխատանքային միջավայրի այն թերթի անունը, որի մեջ գրված են մեզ անհրաժեշտ տվյալները, իսկ երրորդ արգումենտում՝ նշված թերթի անհրաժեշտ տիրույթը: Կարդացված տվյալները **MatLab**–ում կվերագրվեն u զանգվածին:

Օրինակ.

Ենթադրենք՝ **Excel**–ի ֆայլը գտնվում է D:\Work թղթապանակում և ունի ExcelData1.xlsx անվանումը: Վերջինիս առաջին թերթում (sheet1) գրված են նկ. 44–ում բերված տվյալները: Եթե ցանկանում ենք **MatLab**–ի միաջավայրից կարդալ այս թերթի, օրինակ, երրորդ սյան բոլոր տվյալները (այսինքն՝ $C2$–ից $C10$ տիրույթի տվյալները), ապա **MatLab**–ի միջավայրում պետք է գրենք հետևյալ հրամանը.

```
>> u = xlsread('D:\Work\ExcelData1.xlsx', 'Sheet1', 'C2:C10')
   u =
      -7.5000
      -6.9900
     -14.2800
       2.8000
       0.0030
       1.1520
      -3.4500
       4.6200
       8.4100
```

Ինչպես տեսնում ենք, u սյուն–վեկտորին վերագրվեցին անհրաժեշտ տվյալները:



Նկ. 44

Excel–ից տվյալներ կարդալու համար նախատեսված **xlsread** ֆունկցիան կարելի է օգտագործել նաև ավելի պարզ տեսքով: Պարզագույն դեպքում այն ստանում է միայն մեկ արգումենտ՝ անհրաժեշտ ֆայլի անունը.

u = xlsread('ֆայլի անուն')

Այս դեպքում *u* զանգվածին կվերագրվեն **Excel**–ի ֆայլի բոլոր տվյալները, եթե ֆայլը բաղկացած է միայն մի թերթից: Եթե **Excel**–ի ֆայլն ունի մի քանի թերթ, ապա կկարդացվեն առաջին թերթի բոլոր տվյալները: Եթե ցանկանում ենք կարդալ ոչ թե առաջին թերթի, այլ մեկ ուրիշ թերթի բոլոր տվյալները, ապա **xlsread** ֆունկցիան գրվում է երկու արգումենտով.

u = xlsread('ֆայլի անուն', 'թերթի անուն')

MatLab–ը թույլ է տալիս նաև տվյալներ գրել **Excel**–ի ֆայլի մեջ: Դրա համար նախատեսված է **xlswrite** ֆունկցիան, որն ընդհանուր դեպքում ունի հետևյալ գրելաձևը.

xlswrite('ֆայլի անուն', փոփոխականի անուն, 'թերթի անուն', 'տիրույթ')

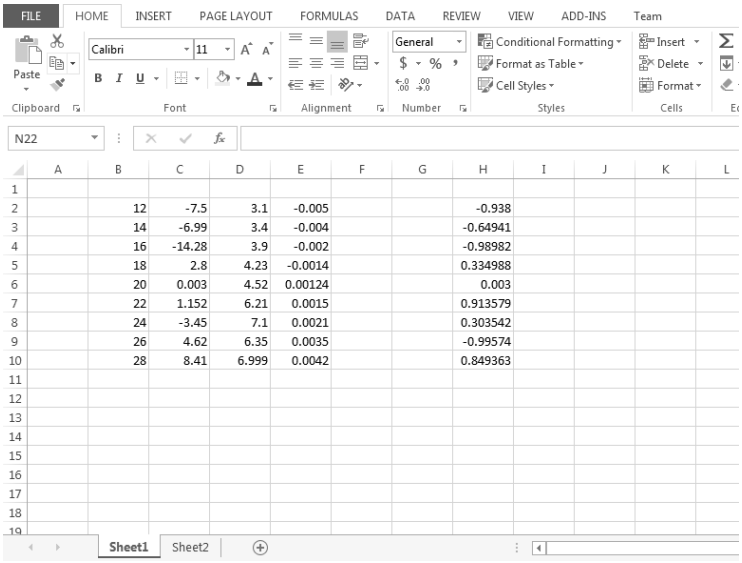
Ինչպես տեսնում ենք, բացի փոփոխականի անունից, որի արժեքը ցանկանում ենք գրել **Excel**–ի ֆայլի մեջ, մնացած բոլոր արգումենտները գրվում են տողերի տեսքով: Փոփոխականը կարող է լինել ինչպես սկալյար, այնպես էլ վեկտոր կամ մատրից:

Օրինակ.

Նախորդ օրինակում (նկ. 44) ExcelData1.xlsx ֆայլից կարդացինք *C*2:*C*10 տիրույթի տվյալները: Այժմ ծրագիրը ձևափոխենք այնպես, որ այդ տվյալները կարդալուց հետո **MatLab**–ը որոշի այդ տվյալների սինուսներն ու ստացված արդյունքները գրի նույն ֆայլի *H*2:*H*10 վանդակներում: Ծրագիրը կընդունի հետևյալ տեսքը.

```
>> v = xlsread('D:\Work\ExcelData1.xlsx', 'Sheet1', 'C2:C10');
>> s = sin(v);
>> xlswrite('D:\Work\ExcelData1.xlsx', s, 'Sheet1', 'H2:H10')
```

Ծրագիրն աշխատեցնելուց հետո ExcelData1.xlsx ֆայլը բացելով՝ կտեսնենք, որ Sheet1 թերթի *H*2:*H*10 տիրույթում գրվել են անհրաժեշտ տվյալները (նկ. 45):



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		12	-7.5	3.1	-0.005			-0.938				
3		14	-6.99	3.4	-0.004			-0.64941				
4		16	-14.28	3.9	-0.002			-0.98982				
5		18	2.8	4.23	-0.0014			0.334988				
6		20	0.003	4.52	0.00124			0.003				
7		22	1.152	6.21	0.0015			0.913579				
8		24	-3.45	7.1	0.0021			0.303542				
9		26	4.62	6.35	0.0035			-0.99574				
10		28	8.41	6.999	0.0042			0.849363				
11												
12												
13												
14												
15												
16												
17												
18												
19												

Նկ. 45

6.9. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Սահմանեք M–ֆունկցիա, որը մուտքային x փոփոխականի համար կվերադարձնի $y = -3x^4 + e^{-0.2x}x^3 + 4\sin x - 6$ ֆունկցիայի արժեքը: M–ֆունկցիան սահմանեք այնպես, որ այն ճիշտ լինի նաև x –ի՝ վեկտոր լինելու դեպքում: Սահմանված ֆունկցիայի օգնությամբ

- հաշվեք $y(7.8)$ և $y(4)$ արժեքները,
- կառուցեք $y(x)$ ֆունկցիայի գրաֆիկը, եթե $-5 \le x \le 12$:

2. Սահմանեք M–ֆունկցիա, որը թույլ կտա կմ/ժ–ով արտահայտված արագությունը վերածել մ/վ–ով արտահայտված արագության: Ֆունկցիան պետք է ունենա ms= =kmh2ms(kmh) տեսքը, որտեղ *kmh2ms*–ը ֆունկցիայի անվանումն է, *kmh* մուտքային արգումենտը կմ/ժ–ով արտահայտված արագությունը, իսկ *ms* ելքային արգումենտը՝ մ/վ–ով արտահայտված արագությունը: Օգտագործելով սահմանված ֆունկցիան՝ մ/վ–ով արտահայտեք 90կմ/ժ և 120կմ/ժ արագությունները:

3. Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել եռանկյան մակերեսը, եթե հայտնի են վերջինիս կողմերը: Ֆունկցիան պետք է ունենա S=area(a, b, c) տեսքը, որտեղ *area*–ն M–ֆունկցիայի անվանումն է, *a*–ն, *b*–ն և *c*–ն եռանկյան կողմերն են, իսկ *S*–ը՝ մակերեսը: Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $a = 3$, $b = 7$, $c = 9$ կողմերով եռանկյան մակերեսը:

4. Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել երկու եռաչափ վեկտորների սկալյար արտադրյալը: Ֆունկցիան պետք է ունենա p=skprod(a, b) տեսքը, *skprod*–ը M–ֆունկցիայի անվանումն է, *a*–ն և *b*–ն համապատասխանաբար $\vec{a}$ և $\vec{b}$ վեկտորների բաղադրիչներից կազմված վեկտորը, իսկ *p*–ն՝ դրանց սկալյար արտադրյալի

արժեքը: Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $\vec{a} = 3\hat{i} + 4\hat{j} - 5\hat{k}$ և $\vec{b} = 4.5\hat{i} - 3\hat{j} + 6\hat{k}$ վեկտորների սկալյար արտադրյալը:

5. Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել երկու եռաչափ վեկտորների վեկտորական արտադրյալը: Ֆունկցիան պետք է ունենա d=vprod(a, b) տեսքը, *vprod*–ը M–ֆունկցիայի անվանումն է, *a*–ն և *b*–ն համապատասխանաբար $\vec{a}$ և $\vec{b}$ վեկտորների բաղադրիչներից կազմված վեկտորը, իսկ *p*–ն՝ դրանց վեկտորական արտադրյալի արդյունքում ստացված վեկտորի բաղադրիչներից կազմված վեկտորը: Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $\vec{a} = \hat{i} - 2.3\hat{j} + 6.4\hat{k}$ և $\vec{b} = 7\hat{i} - 8\hat{j} - \hat{k}$ վեկտորների վեկտորական արտադրյալը:

6. Ցածրհաճախային կոչվում է այն ֆիլտրը, որը թողարկում է միայն տրված հաճախությունից փոքր հաճախությամբ ազդանշանները: Ցածրհաճախային RC ֆիլտրի ելքային և մուտքային լարումների բացարձակ արժեքների հարաբերությունը որոշվում է $K = \left|\frac{U_{out}}{U_{in}}\right| = \frac{1}{\sqrt{1 + (\omega RC)^2}}$ օրենքով, որտեղ ω –ն մուտքային ազդանշանի հաճախությունն է: Սահմանեք M–ֆունկցիա, որը տրված *R*, *C*, ω պարամետրերի համար կորոշի այս հարաբերությունը: Ֆունկցիան պետք է ունենա K=lowpass(R, C, omega) տեսքը: Մուտքային արգումենտներից *R* դիմադրությունը արտահայտված է օհմերով, *C* ունակությունը՝ ֆարադներով, իսկ ω հաճախությունը՝ ռադ/վ–ով: Ֆունկցիան սահմանեք այնպես, որ ճիշտ աշխատի նաև ω հաճախության՝ վեկտոր լինելու դեպքում:

M–ֆայլում գրեք ծրագիր, որը նախ պետք է պահանջի դիմադրության և ունակության արժեքները, այնուհետև, սահմանված lowpass ֆունկցիայի օգնությամբ պատկերի $K(\omega)$ կախվածության գրաֆիկը $10^{-12} \le \omega \le 10^{6}$ ռադ/վ տիրույթում: Գրաֆիկը կառուցեք $R = 1200$ Օհմ և $C = 8$ մկՖ արժեքների համար: *K* առանցքը կառուցեք գծային, իսկ ω առանցքը՝ լոգարիթմական մասշտաբներով:

7. Շերտավոր կոչվում է այն ֆիլտրը, որը թողարկում է միայն տրված հաճախությունների տիրույթի ազդանշանները: Շերտավոր RCL ֆիլտրի ելքային և մուտքային լարումների բացարձակ արժեքների հարաբերությունը որոշվում է $K = \left|\frac{U_{out}}{U_{in}}\right| = \frac{\omega RC}{\sqrt{(1 - \omega^2 LC)^2 + (\omega RC)^2}}$ օրենքով, որտեղ ω –ն մուտքային ազդանշանի հաճախությունն է: Սահմանեք M–ֆունկցիա, որը տրված *R*, *C*, *L*, ω պարամետրերի համար կորոշի այս հարաբերությունը: Ֆունկցիան պետք է ունենա K=bandpass(R,C,L,omega) տեսքը: Մուտքային արգումենտներից *R* դիմադրությունը արտահայտված է օհմերով, *C* ունակությունը՝ ֆարադներով, *L* ինդուկտիվությունը՝ հենրիներով, իսկ ω հաճախությունը՝ ռադ/վ–ով: Ֆունկցիան սահմանեք այնպես, որ ճիշտ աշխատի նաև ω հաճախության՝ վեկտոր լինելու դեպքում:

M–ֆայլում գրեք ծրագիր, որը նախ պետք է պահանջի դիմադրության և ունակության արժեքները, այնուհետև, սահմանված bandpass ֆունկցիայի օգնությամբ

պատկերի $K(\omega)$ կախվածության գրաֆիկը $10^{-2} \le \omega \le 10^{7}$ ռադ/վ տիրույթում: Գրաֆիկը կառուցեք $R = 1100$ Օհմ, $C = 9$ մկՖ, $L = 7$ մՀ և $R = 500$ Օհմ, $C = 300$ մկՖ, $L = 400$ մՀ արժեքների համար: K առանցքը կառուցեք գծային, իսկ ω առանցքը՝ լոգարիթմական մասշտաբներով:

8. $f(x)$ ֆունկցիայի ածանցյալի մոտավոր արժեքը x_0 կետում կարելի է որոշել $\dfrac{df(x)}{dx} = \dfrac{f(x_0 - 2h) - f(x_0 - h) + f(x_0 + h) - f(x_0 + 2h)}{12h}$ բանաձևով, որտեղ h –ը փոքր թիվ է: Սահմանեք M–ֆունկցիա, որն այս բանաձևով թույլ կտա հաշվել $f(x)$ ֆունկցիայի ածանցյալի մոտավոր արժեքը x_0 կետում: Ֆունկցիան պետք է ունենա dfdx=Fdiff(f, x0) տեսքը, որտեղ *f*–ը տրված ֆունկցիան է, *x*0–ն՝ այն կետը, որում ուզում ենք հաշվել *f* ֆունկցիայի ածանցյալը: Ընդունելով, որ $h = \dfrac{x_0}{10}$ և օգտագործելով սահմանված ֆունկցիան՝ որոշեք $f(x) = x^3 e^{2x}$ ֆունկցիայի ածանցյալը $x_0 = 0.25$ կետում և $f(x) = \dfrac{3^x}{2x}$ ֆունկցիայի ածանցյալը $x_0 = 2.8$ կետում: Ստացված արդյունքները համեմատեք անալիտիկ եղանակով ստացված արդյունքների հետ:

9. XY հարթության մեջ կետի կոորդինատներն են (x_0, y_0): Եթե XY հարթությունը առանցքի շուրջ պտտենք ϑ անկյունով ժամացույցի սլաքի ուղղությամբ, ապա կետի նոր կոորդինատները կլինեն (x_1, y_1) և կորոշվեն $x_1 = x_0 \cos\vartheta - y_0 \sin\vartheta$ և $y_1 = x_0 \sin\vartheta + y_0 \cos\vartheta$ բանաձևերով: Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել կետի կոորդինատները XY հարթությունը առանցքի շուրջ ϑ անկյունով պտույտից հետո: Ֆունկցիան պետք է ունենա [x1, y1]=rotation(x0, y0, teta) տեսքը, որտեղ *x*0–ն և *y*0–ն կետի կոորդինատներն են նախքան հարթության պտույտը, իսկ *teta*–ն պտտման անկյունն է՝ արտահայտված աստիճաններով:

- Օգտագործելով սահմանված ֆունկցիան՝ որոշեք (6.5, 2.1) կետի նոր կոորդինատները՝ XY հարթությունը առանցքի շուրջը $\vartheta = 25^{\circ}$ –ով պտտելու դեպքում;
- Տրված է $y = (x-7)^2 + 1.5$ ֆունկցիան, որտեղ $5 \le x \le 9$: Գրեք ծրագիր, որը, օգտագործելով սահմանված rotation ֆունկցիան, կպտտի հարթությունը $\vartheta = 20^{\circ}$ –ով և կպատկերի ֆունկցիայի գրաֆիկը նախքան պտույտը և պտույտից հետո:

10. Սահմանեք $f(x) = 0.01x^5 - 0.03x^4 + 0.4x^3 - 2x^2 - 6x + 5$ M–ֆունկցիան և կառուցեք այդ ֆունկցիայի գրաֆիկը $-4 \le x \le 6$ միջակայքում՝ օգտագործելով ներդրված **fplot** ֆունկցիան:

11. Սահմանեք $f(x) = x^2 + 4\sin 2x - 1$ M–ֆունկցիան և կառուցեք այդ ֆունկցիայի գրաֆիկը $-3 \le x \le 3$ միջակայքում՝ օգտագործելով ներդրված **fplot** ֆունկցիան:

12. Առանց **MatLab**–ից օգտվելու՝ որոշեք, թե ինչի հավասար կլինեն հետևյալ արտահայտությունների արժեքները և պատասխանը ստուգեք **MatLab**–ով.

- `7-3 > 25*4`,
- `2*3>10/5+2>3^2`,
- `4*(3>15/2)+(4>8)^3`,
- `2*3>10/5+2>3^2`,
- `5*(4>7/2)-(3<12)^4`,
- `7*8-3*4 >= ~3*2-4-~0`:

13. Տրված են $a=2$, $b=5$, $c=-4$ փոփոխականները: Առանց **MatLab**–ից օգտվելու՝ որոշեք, թե ինչի հավասար կլինեն հետևյալ արտահայտությունների արժեքները և պատասխանը ստուգեք **MatLab**–ով.

- `a-b > a+b < c`,
- `-5 < c < -2`,
- `b+c >= c > a/b`,
- `a+c == ~(c+a ~= a/b-b)`:

14. Տրված են a=[3 –3 –5 –2 0 8 3 2] և b=[0 5 0 6 –3 2 –1 –4] վեկտորները: Առանց **MatLab**–ից օգտվելու՝ որոշեք, թե ինչի հավասար կլինեն հետևյալ արտահայտությունների արժեքները և պատասխանը ստուգեք **MatLab**–ով.

- `~(~a)`,
- `a == b`,
- `b-a < b`,
- `a-(a<b)`:

15. Առանց **MatLab**–ից օգտվելու՝ որոշեք, թե ինչի հավասար կլինեն հետևյալ արտահայտությունների արժեքները և պատասխանը ստուգեք **MatLab**–ով.

- `5 & ~1`,
- `~-2 > -1 & 11 >= ~0`,
- `3-2/0.5 & 7<5 | -3`,
- `3 | -2 & ~3*-4 | 0`:

16. Պայմանի օպերատորի օգնությամբ որոշեք տրված չորս իրարից տարբեր փոփոխականներից մեծագույնի արժեքը:

17. Տրված x փոփոխականի և a պարամետրի համար որոշեք

$$f(x)=\begin{cases}1 & (x\ge a)\\ \dfrac{1}{4}+\dfrac{1}{2\pi}\arcsin\dfrac{x}{a} & (-a\le x\le a)\\ 0 & (x\le -a)\end{cases}$$ ֆունկցիայի արժեքը:

18. Օգտվելով Կրամերի եղանակից՝ որոշեք

$$\begin{cases}ax+by=c\\ dx+ey=f\end{cases}$$

հավասարումների համակարգի լուծումները x և y փոփոխականների նկատմամբ, a, b, c, d, e, f պարամետրերի տրված արժեքների համար: Եթե համակարգը լուծում չունի, ծրագիրը պետք է վերադարձնի սխալի հաղորդագրություն:

19. Տրված են չորս փոփոխական: Գրեք ծրագիր, որը թույլ կտա դրանցից փոքրագույնը փոխարինել մնացած փոփոխականների գումարով:

20. Տրված են չորս փոփոխական: Գրեք ծրագիր, որը թույլ կտա այն փոփոխականների արժեքները, որոնք հավասար չեն 5–ի կամ 12–ի, փոխարինել զրոներով:

21. Տրված են քառակուսու կողմն ու շրջանի շառավիղը: Գրեք ծրագիր, որը թույլ կտա հաշվել այդ պատկերների մակերեսները և կվերադարձնի դրանցից մեծի արժեքը:

22. Գրեք ծրագիր, որը թույլ կտա որոշել $ax^2+bx+c=0$ հավասարման արմատները: Ծրագիրն աշխատացնելիս օգտագործողը պետք է մուտքագրի a, b, c գործակիցները: Այնուհետև ծրագիրը պետք է հաշվի քառակուսի հավասարման տարբերիչը և վերադարձնի

- "*Two real roots*" տողը և իրական արմատների արժեքները, եթե տարբերիչը դրական է,
- "*One real root*" տողը և հավասարման իրական արմատը, եթե տարբերիչը հավասար է զրոյի,
- "*No real roots*" զգուշացման տողը, եթե տարբերիչը բացասական է:

Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $2x^2+8x+8=0$, $-5x^2+3x-4=0$ և $-2x^2+7x+4=0$ քառակուսի հավասարումների արմատները:

23. Օգտագործելով ցիկլի **while** օպերատոր՝ հաշվեք $S=\sqrt{12}\sum_{k=0}^{m}\frac{(-1/3)^k}{2k+1}$ գումարը $m=5$, $m=10$ և $m=30$ արժեքների համար:

24. Օգտագործելով ցիկլի **while** օպերատոր՝ հաշվեք $P=2\prod_{k=1}^{m}\frac{(2k)^2}{(2k)^2-1}$ արտադրյալը $m=0$, $m=10000$, $m=1000000$ արժեքների համար և ստացված արժեքներից յուրաքանչյուրը համեմատեք π թվի հետ:

25. Օգտագործելով ցիկլի **for** օպերատորը՝ որոշեք հետևյալ գումարները (գումարելիների քանակը և x փոփոխականի արժեքը պետք է ներմուծվեն ծրագրավորողի կողմից).

- $S=x+\frac{x^5}{5}+\frac{x^9}{9}+\frac{x^{13}}{13}+\ldots,$
- $S=\sqrt{\frac{2x}{\pi}}\left(\frac{x}{3}-\frac{x^3}{7\cdot 3!}+\frac{x^5}{11\cdot 5!}-\frac{x^7}{15\cdot 7!}+\ldots\right),$
- $\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots,$
- $\mathrm{bei}\,x=\frac{(x/2)^6}{(1!)^2}-\frac{(x/2)^8}{(3!)^2}+\frac{(x/2)^{10}}{(5!)^2}-\ldots,$
- $\mathrm{Si}\,x=x-\frac{x^3}{3\cdot 3!}+\frac{x^5}{5\cdot 5!}-\frac{x^7}{7\cdot 7!}+\ldots,$
- $\ln\frac{1+x}{1-x}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\ldots\right)$, եթե $|x|<1$:

26. R շառավղով և ρ ծավալային խտությամբ հավասարաչափ լիցքավորված գնդի ստեղծած էլեկտրաստատիկ դաշտի լարվածության մեծությունը տարածության որևէ r կետում որոշվում է հետևյալ օրենքով.

$$E(r)=\begin{cases}\frac{4\pi}{3}\rho r, & (r\le R)\\ \frac{4\pi R^3\rho}{3}\frac{1}{r^3}, & (r\ge R)\end{cases}:$$

Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել տրված *rho* ծավալային խտությամբ և *R* շառավղով գնդի էլեկտրաստատիկ դաշտի լարվածությունը տարածության r կետում: Ֆունկցիան պետք է ունենա E=E_Sphere(r) տեսքը: *rho* և *R* մեծությունները սահմանեք որպես գլոբալ փոփոխական:

Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $R = 9$ սմ շառավղով և $\rho = 2$ CGSE/սմ3 ծավալային խտությամբ գնդի ստեղծած էլեկտրաստատիկ դաշտի լարվածությունը $0 \le r \le 20$ սմ տիրույթում: Օգտագործելով **fplot** ֆունկցիան՝ կառուցեք $E(r)$ կախվածության գրաֆիկը:

27. Նախորդ խնդրում սահմանված *E_Sphere* ֆունկցիան ձևափոխեք այնպես, որ $E(r)$ կախվածության գրաֆիկը ճիշտ կառուցվի **plot** ֆունկցիայի օգնությամբ:

28. Տրված է x=[–4 2 3 –7 0 6 –3 –8 1 5] վեկտորը: Օգտագործելով պայմանի և ցիկլի օպերատորներ՝ գրեք ծրագիր, որը կհայտարարի *p* և *q* երկու վեկտորներ, որոնցից առաջինը բաղկացած կլինի *x* վեկտորի դրական տարրերից, երկրորդը՝ բացասական: *p* և *q* վեկտորների տարրերի հերթականությունը պետք է լինի նույնը, ինչ *x* վեկտորի տարրերի հերթականությունը:

29. Տրված է x=[–4 6 2 7 9 0 1 2 –3 –6 12 8] վեկտորը: Առանց օգտագործելու **MatLab**–ում ներդրված **sort** ֆունկցիան՝ գրեք ծրագիր, որը թույլ կտա այս հաջորդականության տարրերը դասավորել աճման և նվազման կարգով:

30. Օգտագործելով ներդրված ցիկլի օպերատոր՝ հայտարարեք 5×8 չափի *A* մատրից, որի յուրաքանչյուր տարրի արժեքը հավասար է իր տողի և սյան գումարի հարաբերությանը սյան համարի քառակուսուն՝ $A_{mn} = (m+n)/n^2$:

31. Պասկալի համաչափ մատրիցը քառակուսի մատրից է, որի յուրաքանչյուր տարրի արժեքը որոշվում է $P_{mn} = \dfrac{(m+n-2)!}{(m-1)!(n-1)!}$ բանաձևով: Օգտագործելով ներդրված ցիկլի օպերատոր՝ հայտարարեք 5×5 և 8×8 չափերի Պասկալի համաչափ մատրիցներ:

7. ԹՎԱՅԻՆ ԵՂԱՆԱԿՆԵՐԻ ԿԻՐԱՌՈՒԹՅՈՒՆԸ MATLAB–ՈՒՄ

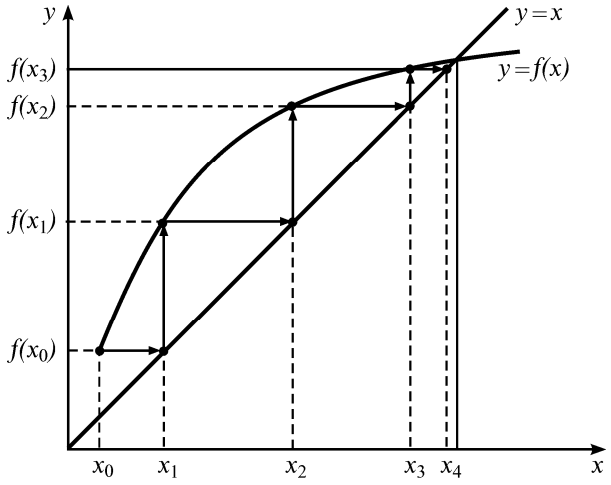
7.1. ՄԵԿ ՓՈՓՈԽԱԿԱՆՈՎ ՀԱՎԱՍԱՐՈՒՄՆԵՐ

Մեկ փոփոխականով հավասարումն ընդհանուր դեպքում կարելի է ներկայացնել $F(x)=0$ տեսքով: Եթե $F(x)$ –ը հանրահաշվական ֆունկցիա է, այսինքն՝ ինչ–որ բազմանդամի տեսք ունի, ապա $F(x)=0$ հավասարումը կոչվում է հանրահաշվական հավասարում, մնացած դեպքերում, երբ $F(x)$ -ի մեջ կան ցուցչային, լոգարիթմական, ուղիղ և հակադարձ եռանկյունաչափական ֆունկցիաներ, $F(x)=0$ հավասարումը կոչվում է տրանսցենդենտ հավասարում: Հավասարման արմատը x փոփոխականի այն արժեքն է, որի դեպքում $F(x)$ ֆունկցիայի գրաֆիկը հատում է աբսցիսների առանցքը, այսինքն՝ փոխում է նշանը: Հավասարումը կարող է մեկ, մի քանի, նույնիսկ անվերջ թվով արմատներ ունենալ: Որոշ հավասարումներ անալիտիկ եղանակով լուծվում են, և ճշգրիտ ստացվում են x –ի այն արժեքները, որոնց դեպքում $F(x)$ ֆունկցիան զրոյի է հավասարվում: Մնացած դեպքերում դիմում են թվային լուծման եղանակներին և ստանում մոտավոր լուծումները՝ ճշգրիտ արմատներին x փոփոխականի հնարավորին չափ մոտ արժեքները: Ամենատարածված թվային եղանակներից մեկը հաջորդական մոտավորությունների կամ իտերացիաների եղանակն է: **MatLab**–ում սահմանված **fzero** ֆունկցիան մեկ փոփոխականով հավասարումների արմատները որոշում է այս եղանակով:

Համառոտ ներկայացնենք իտերացիաների եղանակի սկզբունքը՝ առանց մանրամասն մաթեմատիկական արտածումների և ապացույցների: Նախ $F(x)=0$ հավասարումը ներկայացվում է $x=f(x)$ տեսքով: $F(x)=0$ հավասարումը միշտ հնարավոր է բերել այդպիսի տեսքի, ընդ որում տարբեր եղանակներով, օրինակ՝ աջ և ձախ կողմերին գումարելով x: Ենթադրենք՝ նախապես հայտնի է, որ $[a,b]$ միջակայքում կա հավասարման մեկ արմատ: Կառուցենք թվերի $x_i=f(x_{i-1})$ իտերացիոն հաջորդականությունը, և որպես x –ի սկզբնական արժեք վերցնենք $[a,b]$ տիրույթի որևէ կետի, օրինակ, $x_0=(b-a)/2$ միջնակետի կոորդինատը և, ըստ իտերացիոն բանաձևի, $x_1=f(x_0)$, $x_2=f(x_1)$, ..., $x_i=f(x_{i-1})$, ... թվերի հաջորդականությունը: Կախված $x=f(x)$ ֆունկցիայի վարքից՝ հնարավոր է, որ ստացված կետերի բազմությունը յուրաքանչյուր իտերացիայում ավելի մոտենա կամ ավելի հեռանա $F(x)=0$ հավասարման ճշգրիտ լուծումից: Եթե յուրաքանչյուր իտերացիայում ստանում ենք հավասարման ճշգրիտ լուծմանն ավելի ու ավելի մոտ ար-

ժեքներ, ապա ասում են, որ այս դեպքում իտերացիաների եղանակը զուգամիտում է: Եթե, հակառակը, յուրաքանչյուր քայլում ավելի ենք հեռանում հավասարման ճշգրիտ արմատից, ապա այս դեպքում ասում են, որ իտերացիաների եղանակը տարամիտում է: Ցույց է տրվում, որ x_i հաջորդականությունը զուգամիտում է հավասարման արմատին, եթե $|f'(x)|<1$ և տարամիտում, եթե $|f'(x)|>1$:

Իտերացիաների եղանակն ակնառու պատկերացնելու համար այն հարմար է ներկայացնել գրաֆիկորեն (նկ. 46): Դիտարկենք $|f'(x)|>1$ դեպքը: $x=f(x)$ հավասարման ճշգրիտ արմատը կլինի $y=x$ ուղղի և $y=f(x)$ գրաֆիկի հատման կետի աբսցիսը: Ենթադրենք, x_0-ն զրոյական մոտավորությամբ արմատն է: $y=f(x)$ գրաֆիկի վրա գտնենք $f(x_0)$ կետը: Այժմ այդ կետից x առանցքին զուգահեռ ուղիղ տանենք՝ մինչև այն կհատի $y=x$ ուղիղը: Հատման կետը կգտնվի $x_1=f(x_0)$ աբսցիսով կետում: Ստացված առաջին մոտավորությամբ արմատի համար նույն կերպ կստանանք $f(x_1)$ օրդինատով կետը, որտեղից x առանցքին զուգահեռ ուղիղ տանելով կգտնենք $x_2=f(x_1)$ հաջորդ մոտավորությամբ արմատը: Պրոցեսը շարունակենք այս հերթականությամբ քանի դեռ տրված ε փոքր թվի համար չի իրականացել $|x_i-x_{i-1}|\le\varepsilon$ պայմանը: Գործողությունների ընթացքը **Նկ. 46**-ում ցույց է տրված սլաքներով: Ինչպես տեսնում ենք, գտնված $x_1, x_2, x_3, \ldots$ կետերի հաջորդականությունը ձգտում է հավասարման իրական արմատին:



Նկ. 46

MatLab-ում մեկ փոփոխականով հավասարումները իտերացիաների եղանակով լուծելու համար նախատեսված է **fzero** ֆունկցիան, որը պարզագույն դեպքում ունի

x = fzero('function', x0)

տեսքը: Այս ֆունկցիայի առաջին արգումենտում տողի տեսքով (ապոստրոֆներում) գրվում է $F(x)=0$ հավասարման ձախ կողմը: $x0$–ն x արգումենտի այն արժեքն է, որին մոտ փնտրվում է հավասարման լուծումը: Նշենք, որ մենք ընդամենը գրելու ենք $F(x)$ ֆունկցիայի տեսքը: $x=f(x)$ տեսքի բերելը և մնացած հաշվարկները իրականացնում է **MatLab**–ը:

fzero ֆունկցիայի օգնությամբ հավասարման լուծումը որոշելու համար հարմար է կատարել հետևյալ քայլերը:

1. Սահմանել $F(x)$ –ը որպես առանձին M–ֆունկցիա:
2. Կառուցել $F(x)$ ֆունկցիայի գրաֆիկը և տեսնել, թե ֆունկցիան մոտավորապես որ կետում կամ կետերում է հատում x առանցքը: Գրաֆիկն ավելի հարմար է կառուցել **fplot** ֆունկցիայի օգնությամբ:
3. Օգտագործելով **fzero** ֆունկցիան՝ որոշել հավասարման լուծումը: **fzero**–ի առաջին արգումենտում պետք է տողի տեսքով գրել սահմանված ֆունկցիայի անունը, իսկ երկրորդ արգումենտում՝ գրաֆիկի վրա երևացող այն մոտավոր x արժեքը, որի մոտ ֆունկցիան հավասար է զրոյի: Եթե գրաֆիկը մի քանի անգամ է հատում x առանցքը, այսինքն՝ հավասարումն ունի մեկից ավելի արմատ, ապա **fzero** ֆունկցիան պետք է գրել մի քանի անգամ՝ յուրաքանչյուր անգամ որոշելով մեկ արմատ:

Հարկ է նշել, որ հավասարումը լուծելու համար առանձին ֆունկցիա սահմանելը պարտադիր չի: Կարելի է հավասարումը տողի տեսքով գրել **fzero** ֆունկցիայի առաջին արգումենտում՝ որպես մաթեմատիկական արտահայտություն: Նշենք, որ այն աբսցիսները, որոնցում ֆունկցիայի գրաֆիկը շոշափում է x առանցքը, հավասարման արմատներ չեն համարվում: Եթե հավասարումը արմատ չունի, **fzero** ֆունկցիան վերադարձնում է **NaN** արժեքը:

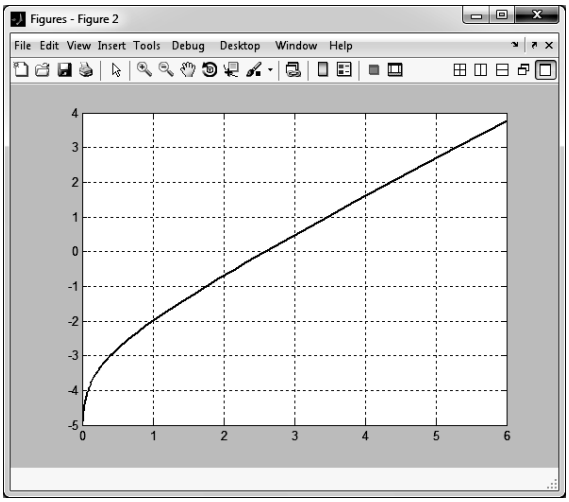
Օրինակ.

Իտերացիաների եղանակով լուծենք $\lg x+x=3$ հավասարումը: Դրա համար նախ այն բերենք $F(x)=0$ տեսքի և $F(x)$ –ը սահմանենք որպես M–ֆունկցիա:

```
function y = yfun(x)
    y = log10(x) + x – 3;
end
```

Այժմ կառուցենք սահմանված ֆունկցիայի գրաֆիկը, օրինակ՝ $[0;6]$ միջույթում: Գրաֆիկը կարելի է կառուցել **plot** ֆունկցիայի օգնությամբ, սակայն ավելի հարմար է **fplot** ֆունկցիան, որը գրաֆիկն ավելի ճշգրիտ է կառուցում (Նկ. 47):

```
>> fplot('yfun', [0 6])
>> grid on
```



Նկ. 47

Ինչպես տեսնում ենք, ֆունկցիայի գրաֆիկը x առանցքը հատում է $x = 2.5$ կետի մոտակայքում, այսինքն՝ $\lg x + x = 3$ հավասարման լուծումը այդ կետի մոտ է: Հետևաբար, **fzero** ֆունկցիայի երկրորդ արգումենտում կգրենք այդ արժեքը:

```
>> x = fzero('yfun', 2.5)
   x = 2.587174309874031
```

7.2. ՖՈՒՆԿՑԻԱՅԻ ՆՎԱԶԱԳՈՒՅՆ ԵՎ ԱՌԱՎԵԼԱԳՈՒՅՆ ԱՐԺԵՔՆԵՐԻ ՈՐՈՇՈՒՄԸ

MatLab–ը $y = f(x)$ ֆունկցիայի նվազագույն և առավելագույն արժեքները որոշելու միջոցներ ունի: Ինչպես հայտնի է, ֆունկցիայի էքստրեմումի կետերը որոշելու համար անհրաժեշտ է զրոյի հավասարեցնել $f(x)$ ֆունկցիայի ածանցյալը, և ստացված հավասարման x արմատը տեղադրել $f(x)$ ֆունկցիայի մեջ:

MatLab–ում ներդրված **fminbnd** ֆունկցիան նախատեսված է ֆունկցիայի նվազագույն արժեքը որևէ $[a, b]$ միջակայքում որոշելու համար: Այն ունի հետևյալ տեսքը.

x_min = fminbnd('function', a, b)

Այս ֆունկցիայի առաջին արգումենտը, ինչպես **fzero** ֆունկցիայի դեպքում, տողի տեսքով գրված այն M–ֆունկցիայի անունն է, որի մեջ հայտարարված է $f(x)$ ֆունկցիան: *x_min*–ը արգումենտի այն արժեքն է, որի համար $f(x)$ ֆունկցիայի արժեքը նվազագույնն է: Ֆունկցիայի նվազագույն արժեքը գրվում է **fminbnd** ֆունկցիայի երկրորդ, ոչ պարտադիր ելքային արգումենտում.

[x_min y_min] = fminbnd('function', a, b)

fminbnd ֆունկցիայի օգնությամբ հայտնաբերվում են $y = f(x)$ ֆունկցիայի լոկալ նվազագույն արժեքները: Եթե տրված միջակայքում այդպիսիք գոյություն չունեն, վերադարձվում է այդ տիրույթի a կամ b ծայրակետը՝ կախված այն բանից, թե դրանցից որ կետում է ֆունկցիայի արժեքն ավելի փոքր:

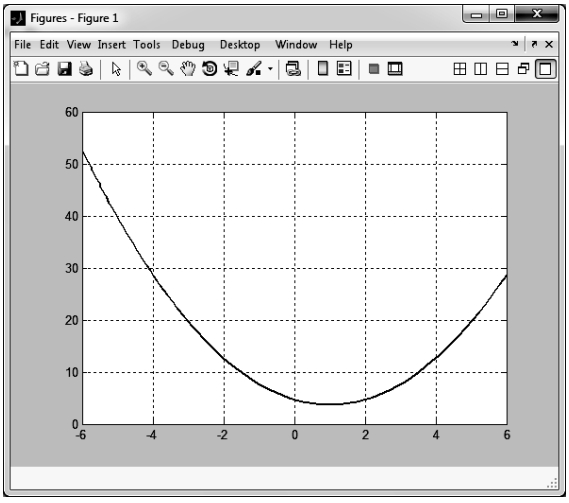
Օրինակ.

Որոշենք $y = x^2 - 2x + 4.68$ ֆունկցիայի նվազագույն արժեքը: Դրա համար նախ սահմանենք *F_Min* անունով M–ֆունկցիա, որի մեջ հայտարարվում է $y = x^2 - 2x + 4.68$ ֆունկցիան:

```
function y = F_Min(x)
    y = x.^2 - 2*x + 4.68;
end
```

Այժմ կառուցենք այս ֆունկցիայի գրաֆիկը, օրինակ [–6, 6] միջակայքում (նկ. 48):

```
>> fplot('F_Min', [-6 6])
>> grid on
```



Նկ. 48

Ինչպես տեսնում ենք, այս տիրույթում ֆունկցիան ունի նվազագույն արժեք: Այժմ այդ տիրույթում որոշենք նվազագույն արժեքը և դրա արգումենտը:

```
>> [x_min y_min] = fminbnd('F_Min', -6, 6)
   x_min = 1.0000
   y_min = 3.6800
```

Ֆունկցիայի առավելագույն արժեքի որոշման համար **MatLab**–ում հատուկ ֆունկցիա սահմանված չի: Այն որոշելու համար անհրաժեշտ է ֆունկցիան բազմապատկել (–1)–ով և, կիրառելով **fminbnd** ֆունկցիան՝ գտնել դրա նվազագույն արժեքը:

7.3. ԲԱԶՄԱՆԴԱՄՆԵՐ

Ընդհանուր դեպքում n–րդ կարգի բազմանդամ է կոչվում

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

տեսքի ֆունկցիան, որտեղ $a_n, a_{n-1}, ..., a_1, a_0$ թվերը բազմանդամի գործակիցներն են: **MatLab**–ում n–րդ կարգի բազմանդամը սահմանվում է վեկտորի տեսքով, որի տարրերը $a_n, a_{n-1}, ..., a_1, a_0$ գործակիցներն են: Այդ վեկտորի առաջին տարրը պետք է լինի x–ի ամենաբարձր աստիճանն ունեցող անդամի գործակիցը: Եթե բազմանդամի մեջ կան զրոյի հավասար գործակիցներով անդամներ, ապա **MatLab**–ում վեկտորի տեսքով պետք է գրվեն բոլոր, ներառյալ զրոյի հավասար գործակիցները:

Օրինակներ.

- $p = 3x^2 + 4x - 1$ 2–րդ կարգի բազմանդամը **MatLab**–ում սահմանվում է p = [3 4 –1] վեկտորի տեսքով:
- $q = 5x^3 + 6$ 3–րդ կարգի բազմանդամը սահմանվում է q = [5 0 0 6] վեկտորի տեսքով:
- $q = 7x^6 + 3x^2 - 2x$ 6–րդ կարգի բազմանդամը սահմանվում է q = [7 0 0 0 3 –2 0] վեկտորի տեսքով:

Սահմանված բազմանդամի արժեքը կամայական x կետում կարելի է ստանալ **polyval(p,x)** ֆունկցիայի օգնությամբ, որտեղ p–ն սահմանված բազմանդամն է, իսկ x–ը՝ այն կետը, որի համար ցանկանում ենք ստանալ p բազմանդամի արժեքը: Եթե x–ը ոչ թե թիվ է, այլ զանգված (վեկտոր կամ մատրից), ապա բազմանդամի արժեքը որոշվում է այդ զանգվածի յուրաքանչյուր տարրի համար, և արդյունքում ստացվում է x զանգվածի չափի մի նոր զանգված, որում գրված են համապատասխան կետերում բազմանդամի արժեքները:

Օրինակ.

Որոշենք $p = 7x^8 + 3x^7 - 9x^5 + 6x^4 - x^2 + 2$ բազմանդամի արժեքները $x = -2$ կետում և $5 \le x \le 12$ տիրույթում՝ x–ի ամբողջ արժեքներով կետերում:

```
>> p = [7 3 0 –9 6 0 –1 2];
>> x = –2;
>> t = polyval(p, x)
   t = –892
>> x = [5:12];
>> k = polyval(p,x)
   k = 588872   2089148   6098192   15432698   35020424   72915992
       141601088  259604342
```

$p(x)$ բազմանդամի արմատները x արգումենտի այն արժեքներն են, որոնց համար $p(x)=0$: Ինչպես հայտնի է, բազմանդամի արմատների քանակը հավասար է բազմանդամի կարգին: Օրինակ՝ $3x^2-2x+1$ բազմանդամն ունի երկու արմատ, իսկ x^4-3x^2+6x-1 բազմանդամը՝ չորս արմատ: **MatLab**–ում բազմանդամի արմատները որոշելու համար նախատեսված է **roots(p)** ֆունկցիան, որտեղ p–ն բազմանդամն է:

Օրինակ.

Որոշենք $p=x^2+2x+5$ բազմանդամի արմատները:

```
>> p = [1 2 5];
>> x = roots(p)
   x =
     -1.0000 + 2.0000i
     -1.0000 - 2.0000i
```

MatLab–ը թույլ է տալիս նաև, իմանալով բազմանդամի արմատները՝ վերականգնել բազմանդամի գործակիցները: Դրա համար նախատեսված է **poly(x)** ֆունկցիան, որտեղ x–ը p բազմանդամի արմատներից կազմված վեկտորն է:

Օրինակ.

Նախորդ օրինակի բազմանդամի x արմատներով վերականգնենք p բազմանդամը:

```
>> p = poly(x)
   p = 1     2     5
```

MatLab–ը հնարավորություն է տալիս կատարելու գումարման, հանման, բազմապատկման և բաժանման գործողություններ բազմանդամների հետ, ինչպես նաև ածանցել բազմանդամները:

Քանի որ բազմանդամները **MatLab**–ում ներկայացվում են վեկտորների տեսքով, ապա դրանց գումարումը և հանումը կարելի է կատարել սովորական ձևով՝ որպես երկու վեկտորների գումարում կամ հանում: Եթե գումարվող կամ հանվող բազմանդամները նույն կարգի են, ապա այս գործողությունները որևէ խնդիր չեն առաջացնում: Սակայն եթե դրանց կարգերը տարբեր են, ապա ցածր կարգ ունեցող բազմանդամին ձախից պետք է 0 արժեք ունեցող տարրեր ավելացնել այնքան, մինչև որ այն հավասարվի բարձր կարգ ունեցող բազմանդամի երկարությանը, քանի որ **MatLab**–ում գումարել կամ հանել կարելի է միայն նույն երկարության վեկտորները:

Օրինակ.

Որոշենք $p=5x^2-2x+1$ և $q=6x^3+4x^2-2x$ բազմանդամների գումարն ու տարբերությունը: Քանի որ p բազմանդամը 2–րդ կարգի է, իսկ q–ն՝ 3–րդ, ապա p բազմանդամը ներկայացնող վեկտորին ձախից պետք է ավելացնել 1 հատ զրո արժեք ունեցող տարր:

```
>> p = [0 5 –2 1];
>> q = [6 4 –2 0];
>> s = p + q
   s =
        6     9    –4     1
>> d = p – q
   d =
       –6     1     0     1
```

p և q երկու բազմանդամների արտադրյալը հաշվելու համար նախատեսված է **conv(p, q)** ֆունկցիան: Արդյունքում ստացվում է նոր վեկտոր, որն իրենից ներկայացնում է p և q բազմանդամների արտադրյալի գործակիցները: Ի տարբերություն գումարման գործողության, բազմապատկվող բազմանդամների կարգը կարող է տարբեր լինել: Երեք կամ ավելի բազմանդամներ բազմապատկելիս **conv** ֆունկցիան օգտագործում են հաջորդաբար:

p և q երկու բազմանդամների հարաբերությունը որոշելու համար նախատեսված է **deconv(p, q)** ֆունկցիան, որը վերադարձնում է երկու արգումենտ: Առաջին արգումենտը p և q բազմանդամների քանորդի վեկտորն է, երկրորդը՝ մնացորդի:

Օրինակ.

Բազմապատկենք և բաժանենք $p = 6x^4 - 8x^3 - 1$ և $q = 7x^2 + 3x - 1$ բազմանդամները:

```
>> p = [6 –8 0 0 –1];
>> q = [7 3 –1];
>> f = conv(p, q)
   f =
      42   –38   –30     8    –7    –3     1
>> [g h] = deconv(q, p)
   g =
      0.8571   –1.5102    0.7697
   h =
      0   0   0   –3.8192   –0.2303
```

polyder(p) ներդրված ֆունկցիան նախատեսված է p բազմանդամի ածանցյալն ըստ x արգումենտի որոշելու համար: Բացի այդ, **polyder** ֆունկցիան կարելի է օգտագործել p և q երկու բազմանդամների արտադրյալի և հարաբերության ածանցյալները հաշվելու համար: Եթե ցանկանում ենք որոշել p և q բազմանդամների արտադրյալի ածանցյալը, պետք է գրենք **polyder(p, q)** ֆունկցիան երկու մուտքային արգումենտով և այն վերագրենք մեկ փոփոխականի: Եթե ցանկանում ենք որոշել p և q բազմանդամների հարաբերության ածանցյալը, պետք է գրենք **polyder(p, q)** ֆունկցիան երկու մուտքային արգումենտով և այն վերագրենք երկու փոփոխականների:

Օրինակ.

Որոշենք նախորդ օրինակում գրված $p = 6x^4 - 8x^3 - 1$ և $q = 7x^2 + 3x - 1$ բազմանդամների *dp* և *dq*, դրանց արտադրյալի *df* և հարաբերության *dg* ածանցյալները:

```
>> p = [6 -8 0 0 -1];
>> q = [7 3 -1];
>> dp = polyder(p)
   dp =
       24   -24     0     0
>> dq = polyder(q)
   dq =
       14     3
>> df = polyder(p, q)
   df =
      252  -190  -120    24   -14    -3
>> [dg dh] = polyder(p, q)
   dg =
       84    -2   -72    24    14     3
   dh =
       49    42    -5    -6     1
```

7.4. ԿՈՐԵՐՈՎ ՄՈՏԱՐԿՈՒՄ

Կորերով մոտարկումը կամ ռեգրեսիոն վերլուծությունը տրված x և համապատասխան y կետերի բազմությունը որևէ ֆունկցիայով մոտարկելու գործողությունն է: Ենթադրենք, փորձից հայտնի են ինչ–որ անհայտ $y(x)$ կախվածության y_k արժեքները վերջավոր թվով n հատ x_k կետերում: Կորերով մոտարկումը այդ $y(x)$ կախվածության որոնման գործողությունն է, ընդ որում բոլորովին պարտադիր չէ, որ որոնման արդյունքում ստացված ֆունկցիայի գրաֆիկն անցնի բոլոր (x_k, y_k) $(k = 1,2,...,n)$ կետերով: Կորերով մոտարկումը բավականին բարդ գործողություն է: Մոտարկման գործողությունն իրականացնելու համար ընտրում են տարբեր փորձնական ֆունկցիաներ (գծային կամ բազմանդամային, ցուցչային և այլն):

Երբեմն հնարավոր է կռահել կամ նախօրոք հայտնի է, թե մոտավորապես ինչ ֆունկցիայով կարելի է կետերի բազմությունը մոտարկել, սակայն ավելի հաճախ որոնվող ֆունկցիայի մասին ոչինչ հայտնի չի լինում, և տարբեր փորձնական ֆունկցիաներ օգտագործելով՝ նայում են, թե որն է առավել լավ մոտարկում ապահովում:

Այս բաժինը նվիրված է կորերով մոտարկման հիմնական գաղափարներին և **MatLab**–ի համապատասխան ֆունկցիաներին:

7.4.1. Կորերով մոտարկումը բազմանդամային ֆունկցիաներով

Ամենահարմար և առավել կիրառվող մոտարկային ֆունկցիաները բազմանդամներն են: Տվյալները բազմանդամային ֆունկցիայով մոտարկելիս հնարավոր է երկու դեպք: Առաջին դեպքում բազմանդամային ֆունկցիան անցնում է տվյալներից յուրաքանչյուրով, իսկ երկրորդ դեպքում ֆունկցիան ընտրվում է այնպես, որ բոլոր տվյալներով անցնելը պարտադիր չլինի:

Նախ համառոտ դիտարկենք առաջին դեպքը: Եթե տրված են (x_k, y_k) $(k = 1,2,...,n)$ կետերը, ապա, ինչպես հայտնի է, միշտ գոյություն ունի $(n-1)$-րդ կարգի այնպիսի $y_k = \sum_{k=1}^{n} p_k x_k^{n-k}$ բազմանդամ, որն անցնում է այդ բոլոր կետերով: Այդ բազմանդամի p_k $(k = 1, 2, ..., n)$ գործակիցները կորոշվեն՝ (x_k, y_k) զույգերը հերթով տեղադրելով բազմանդամի մեջ: Կստացվի n անհայտով n հատ գծային հավասարումների համակարգ, որի լուծումները կլինեն անհայտ p_k գործակիցները:

$$\begin{cases} y_1 = p_1 x_1^{n-1} + p_2 x_1^{n-2} + ... + p_{n-1} x_1 + p_n \\ y_2 = p_1 x_2^{n-1} + p_2 x_2^{n-2} + ... + p_{n-1} x_2 + p_n \\ ... \\ y_n = p_1 x_n^{n-1} + p_2 x_n^{n-2} + ... + p_{n-1} x_n + p_n \end{cases}$$

Այս եղանակի թերությունն այն է, որ եթե այս բազմանդամի օգնությամբ ցանկանանք ստանալ ոչ թե x_k կետերից որևէ մեկում, այլ կամայական x_k և x_{k+1} կետերի միջև ընկած որևէ այլ կետում բազմանդամի արժեքը, ապա կարող ենք ստանալ սխալ արդյունք, ինչն առավել նկատելի է մանավանդ բարձր կարգի բազմանդամներով մոտարկելու դեպքում:

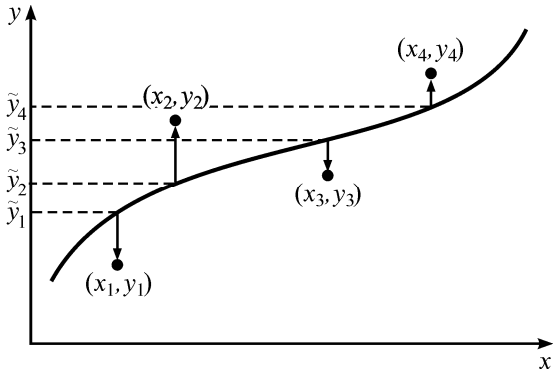
Երկրորդ դեպքում, ինչպես ասացինք, հնարավոր է (x_k, y_k) $(k = 1,2,...,n)$ տվյալները մոտարկել այնպիսի բազմանդամով, որը պարտադիր չանցնի բոլոր այդ կետերով, սակայն ավելի լավ նկարագրի որոնվող ֆունկցիայի վարքը, քան $(n-1)$-րդ կարգի բազմանդամով մոտարկելու դեպքում: Ակնհայտ է, որ այսպիսի բազմանդամը պետք է $(n-1)$-ից ավելի ցածր կարգի լինի: Այսպիսի մոտարկային ֆունկցիա ստանալու ամենահիմնական եղանակը նվազագույն քառակուսիների եղանակն է (**Least–Squares Method**): Ըստ այս եղանակի, որպես լավագույն մոտարկային ֆունկցիա ընտրվում է այն $\tilde{y} = f(x)$ ֆունկցիան, որի դեպքում

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(\widetilde{y}_k - y_k)^2$$

միջին քառակուսային սխալը (**Mean–Squared Error,** կրճատ՝ **MSE**) նվազագույնն է: Հաճախ **MSE**–ի փոխարեն օգտագործում են

$$RMSE = \sqrt{MSE}$$

միջին քառակուսային սխալի արմատը (**Root Mean–Squared Error**): Այս բանաձևերում y_k –ն փորձից հայտնի k –րդ դիսկրետ տվյալի արժեքն է, իսկ $\widetilde{y}_k$ –ն՝ ստացված մոտարկային ֆունկցիայի արժեքը x_k կետում (նկ. 49): Նկատենք, որ **RMSE** մեծությունն ունի y_k մեծության չափողականությունը, մինչդեռ **MSE** մեծությունը՝ y_k մեծության քառակուսու չափողականությունը: $\widetilde{y}_k - y_k$ տարբերությունն, այսպիսով, x_k կետում մոտարկային ֆունկցիայի և փորձնական եղանակով ստացված k –րդ դիսկրետ տվյալի տարբերությունն է (residual):



Նկ. 49

Պարզագույն բազմանդամային ֆունկցիան գծային ֆունկցիան է (1-ին կարգի բազմանդամը): Ենթադրենք, (x_k, y_k) տվյալները ցանկանում ենք մոտարկել $\widetilde{y} = p_1 x + p_2$ գծային ֆունկցիայով: Լավագույն գծային ֆունկցիան որոշելու համար անհրաժեշտ է p_1 և p_2 գործակիցներն ընտրել այնպես, որ **MSE** և **RMSE** մեծություններն ունենան նվազագույն արժեքը: Գծային ֆունկցիայի դեպքում, ըստ սահմանման,

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(\widetilde{y}_k - y_k)^2 = \frac{1}{n}\sum_{k=1}^{n}(p_1 x_k + p_2 - y_k)^2 :$$

Այս մեծության նվազագույն արժեքը որոշելու համար անհրաժեշտ է $\frac{\partial MSE}{\partial p_1}$ և $\frac{\partial MSE}{\partial p_2}$ ածանցյալները հավասարեցնել զրոյի՝

$$\frac{\partial MSE}{\partial p_1} = \frac{1}{n}\sum_{k=1}^{n} 2(p_1 x_k + p_2 - y_k)x_k = \frac{2}{n}\left(p_1 \sum_{k=1}^{n} x_k^2 + p_2 \sum_{k=1}^{n} x_k - \sum_{k=1}^{n} x_k y_k \right) = 0,$$

$$\frac{\partial MSE}{\partial p_2} = \frac{1}{n}\sum_{k=1}^{n} 2(p_1 x_k + p_2 - y_k) = \frac{2}{n}\left(p_1 \sum_{k=1}^{n} x_k + p_2 n - \sum_{k=1}^{n} y_k \right) = 0,$$

և ստացված

$$\begin{cases} p_1 \sum_{k=1}^{n} x_k^2 + p_2 \sum_{k=1}^{n} x_k = \sum_{k=1}^{n} x_k y_k \\ p_1 \sum_{k=1}^{n} x_k + p_2 n = \sum_{k=1}^{n} y_k \end{cases}$$

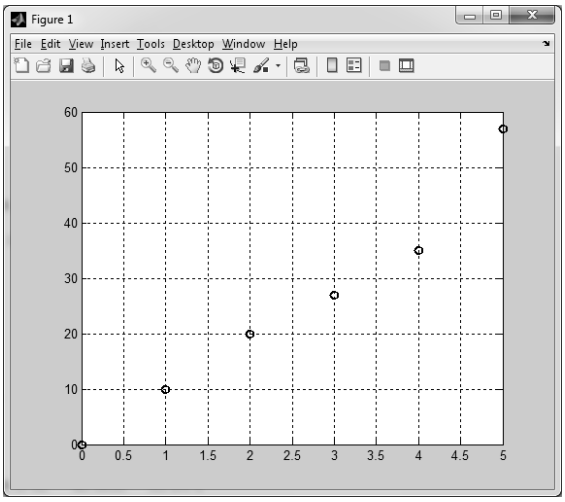
գծային հավասարումների համակարգից որոշել **MSE**–ի նվազագույն արժեքին համապատասխանող p_1 և p_2 գործակիցները, որոնցով հաշվված $\tilde{y} = p_1 x + p_2$ ֆունկցիան լավագույն ձևով է մոտարկում փորձնական կետերը:

MatLab–ում (x_k, y_k) դիսկրետ տվյալները վերը նկարագրված եղանակով լավագույն կերպով մոտարկող գծային ֆունկցիան որոշվում է **polyfit(x,y,1)** ներդրված ֆունկցիայով, որի x և y արգումենտները (x_k, y_k) տվյալներով կառուցված վեկտորներն են, իսկ երրորդ՝ 1 արգումենտը ցույց է տալիս մոտարկվող բազմանդամի կարգը: **polyfit(x,y,1)** ֆունկցիան որպես ելքային տվյալներ վերադարձնում է որոնվող գծային ֆունկցիայի p_1 և p_2 գործակիցները՝ որպես p վեկտոր:

Օրինակ.

Հայտնի են շղթայի տեղամասում չափվող լարման $U = \{0, 10, 20, 27, 35, 57\}$ արժեքները ժամանակի $t = \{0, 1, 2, 3, 4, 5\}$ պահերին: Տեսնենք, թե որ գծային ֆունկցիան է լավագույն ձևով մոտարկում այս (t_k, U_k) արժեքները: Նախ գրաֆիկորեն պատկերենք փորձի արդյունքում ստացված տվյալները: Դրա համար կգրենք հետևյալ ծրագիրը.

```
t = [0:5];
U = [0 10 20 27 35 57];
plot(t, U, 'o')
grid on
```

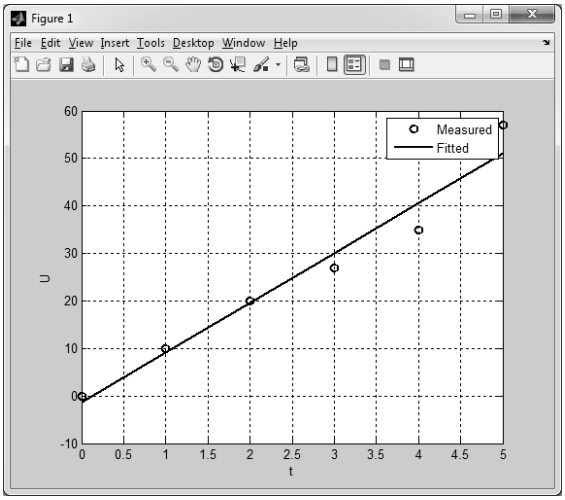


Նկ. 50

Ծրագրի արդյունքը ներկայացված է նկ. 50–ում: Այժմ ստանանք այն գծային ֆունկցիան, որը լավագույն ձևով մոտարկում է այս կետերը: Համապատասխան ծրագիրը և նրա գործարկումից հետո հրամանի պատուհանում ստացվող արդյունքները ունեն այսպիսի տեսք.

```
t = [0:5];
U = [0 10 20 27 35 57];
p = polyfit(t, U, 1)
U_Fit = polyval(p, t)
error = U-U_Fit
MSE = mean(error.^2)
RMSE = sqrt(MSE)
plot(t, U, 'ok')
hold on
plot(t, U_Fit, 'k')
grid on
xlabel('t')
ylabel('U')
legend('Measured', 'Fitted')

p =
   10.4857   -1.3810
U_Fit =
   -1.3810    9.1048   19.5905   30.0762   40.5619   51.0476
error =
    1.3810    0.8952    0.4095   -3.0762   -5.5619    5.9524
MSE =
   13.1175
RMSE =
    3.6218
```

Ծրագրում նախ հայտարարվել են չափումներից ստացված t և U տվյալները: Այնուհետև այդ տվյալները մոտարկել ենք գծային ֆունկցիայով **polyfit(t, U, 1)** ֆունկցիայի օգնությամբ, որի ելքային p_1 և p_2 փոփոխականները վերագրել ենք p վեկտորին: Ինչպես երևում է ծրագրի արդյունքից, $p_1 = 10.4857$ և $p_2 = -1.3810$: Այնուհետև **polyval** ֆունկցիայի օգնությամբ ստացված գծային $U\_Fit = 10.4857t - 1.3810$ ֆունկցիայի մեջ հերթով տեղադրել ենք t փոփոխականի արժեքները և ստացել ենք *U_Fit* լարման արդյունքները ժամանակի տրված պահերին մոտարկված ֆունկցիայի միջոցով: Ինչպես տեսնում ենք, *U_Fit* փոփոխականի արժեքները ճշգրիտ չեն համընկնում համապատասխան պահերին չափումից ստացված U փոփոխականի արժեքների հետ: Որպեսզի որոշենք յուրաքանչյուր t պահին փորձի և մոտարկված ֆունկցիայով ստացված լարումների միջև շեղումը, *error* փոփոխականին վերագրել ենք դրանց տարբերությունը և վերջինիս օգնությամբ հաշվել ենք *MSE* և *RMSE* մեծությունները: Վերջում պատկերել ենք փորձի տվյալները և մոտարկված ֆունկցիայի գրաֆիկները: Ինչպես տեսնում ենք, չնայած այն բանին, որ *U_Fit* ֆունկցիան ճշգրիտ չի համընկնում U մեծության արժեքների հետ, սակայն բավականին լավ մոտարկում է այդ տվյալները (նկ. 51):



Նկ. 51

Տվյալների մոտարկումը կարելի է կատարել նաև ավելի բարձր կարգի բազմանդամներով: Այդ նպատակին դարձյալ ծառայում է **polyfit** ֆունկցիան, որի երրորդ արգումենտում գրվում է մոտարկվող բազմանդամի կարգը: Օրինակ, 2–րդ աստիճանի բազմանդամով (քառակուսային ֆունկցիայով) մոտարկման ֆունկցիան գրվում է **polyfit(x,y,2)**, իսկ 5–րդ կարգի բազմանդամով մոտարկման ֆունկցիան՝ **polyfit(x,y,5)** տեսքով: Ֆունկցիան վերադարձնում է մոտարկվող բազմանդամի գործակիցները վեկտորի տեսքով: Ինչպես արդեն նշել ենք, մոտարկվող բազմանդամի կարգն ավելացնելիս ստացված ֆունկցիան հնարավոր է, որ ավելի վատ մոտարկի տվյալները, քան ցածր կարգի բազմանդամներով մոտարկելու

դեպքում: Ասվածն ավելի ակնառու դարձնելու համար դիտարկենք հետևյալ օրինակը:

Օրինակ.

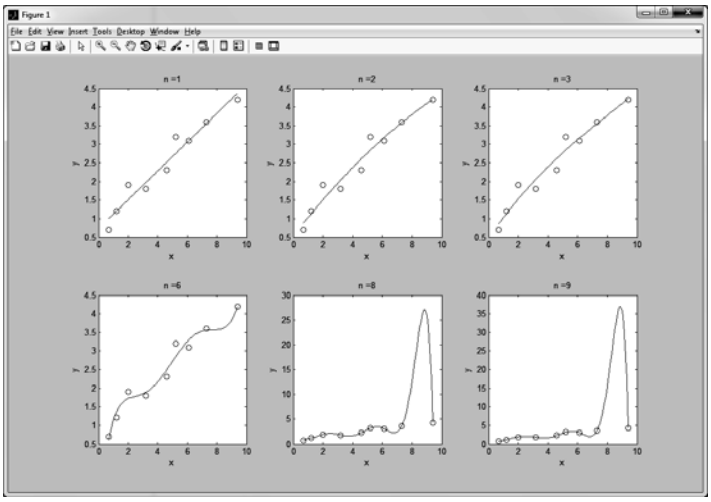
Ենթադրենք, որևէ փորձից ստացել ենք (0.7, 0.7), (1.2, 1.2), (2, 1.9), (3.2, 1.8), (4.6, 2.3), (5.2, 3.2), (2.6, 3.1), (7.3, 3.6), (9.4, 4.2) դիսկրետ տվյալները: Գրենք ծրագիր, որն այս տվյալները հաջորդաբար կմոտարկի 1-ին, 2-րդ, 3-րդ, 6-րդ, 8-րդ, 9-րդ կարգի բազմանդամային ֆունկցիաներով և կպատկերի փորձնական կետերն ու ստացված յուրաքանչյուր բազմանդամի գրաֆիկները:

Ծրագիրը պարզեցնելու համար օգտագործել ենք **for** ցիկլի օպերատորը: Նախ սահմանել ենք դիսկրետ տվյալների կոորդինատները x և y վեկտորների տեսքով: Որպես **for** ցիկլի փոփոխական հայտարարվել է k-ն, որը պարունակում է դիսկրետ տվյալներ (մոտարկվող բազմանդամների կարգերը): Յուրաքանչյուր k-ի համար **polyfit** ֆունկցիայով մոտարկվել են (x, y) տվյալները, այնուհետև **polyval** ֆունկցիայով ստացված բազմանդամների մեջ հերթով տեղադրել ենք xp փոփոխականի արժեքները և յուրաքանչյուրի համար ստացել ենք y_k վեկտորները: xp-ն վեկտոր է, որը որպես սկզբնական և վերջնական արժեք ստանում է x-ի առաջին և վերջին տարրերի արժեքները և փոխվում է 0.1 քայլով: Սա արվել է՝ ֆունկցիան ավելի ողորկ պատկերելու համար:

Այսպիսով, ծրագիրը կունենա հետևյալ տեսքը.

```
x = [0.7 1.2 2 3.2 4.6 5.2 6.1 7.3 9.4];
y = [0.7 1.2 1.9 1.8 2.3 3.2 3.1 3.6 4.2];
xp = [0.7:0.1:9.4];
n = 1;
for k = [1 2 3 6 8 9]
    p_k = polyfit(x,y,k)
    y_k = polyval(p_k,xp)
    subplot(2,3,n)
    plot(x,y,'ok')
    hold on
    plot(xp,y_k,'k')
    xlabel('x')
    ylabel('y')
    title(strcat('n = ', num2str(k)))
    n = n + 1;
end
```

Ծրագրի իրականացման արդյունքում ստացված գրաֆիկները պատկերված են մեկ գրաֆիկական պատուհանի մեջ, յուրաքանչյուր մոտարկման արդյունքը ներկայացված է առանձին ենթապատուհանում (նկ. 52): Ընթացիկ ենթապատուհանի համարը ղեկավարվում է n փոփոխականով:



Նկ. 52

Ցվում է, թե բազմանդամի կարգն ավելացնելով՝ ավելի լավ մոտարկումներ պետք է ստացվեին: Սակայն, ինչպես երևում է նկ. 7-ից, 6-րդ, 8-րդ և 9-րդ կարգի բազմանդամներով մոտարկումները, չնայած այն բանին, որ ավելի մոտ են անցնում դիսկրետ տվյալներով, տալիս են ավելի վատ արդյունք միջանկյալ կետերում, քան ցածր կարգի բազմանդամները: Այսպիսով, եթե, օրինակ, ցանկանանք 9-րդ կարգի բազմանդամով մոտարկված բազմանդամի արժեքը ստանալ կամայական x կետում, կստանանք սխալ արդյունք:

7.4.2. Կորերով մոտարկում այլ ֆունկցիաներով

Երբեմն հարմար է տվյալները մոտարկել ոչ թե բազմանդամային, այլ ուրիշ ֆունկցիաներով: Ընդհանրապես, ցանկացած ֆունկցիայով կարելի է տվյալները մոտարկել, սակայն առավել հաճախ օգտագործվում են աստիճանային, ցուցչային, լոգարիթմական և հակադարձ ֆունկցիաները, որոնք ունեն հետևյալ տեսքերը.

- Աստիճանային ֆունկցիա – $y = ax^m$,
- Ցուցչային ֆունկցիա – $y = ae^{mx}$ կամ $y = a \cdot 10^{mx}$,
- Լոգարիթմական ֆունկցիա – $y = m \ln x + a$ կամ $y = m \lg x + a$,
- Հակադարձ ֆունկցիա – $y = \dfrac{1}{mx + a}$:

Այս ֆունկցիաներով մոտարկելու համար ևս կարելի է օգտվել **polyfit** ներդրված ֆունկցիայից, եթե սրանք բերենք $y = mx + a$ գծային ֆունկցիայի տեսքի: Լոգարիթմական ֆունկցիան արդեն ունի գծային ֆունկցիայի տեսք, միայն թե x–ի փոխարեն գրված է $\ln x$ կամ $\lg x$: Դիտարկենք մնացած ֆունկցիաները:

Լոգարիթմելով աստիճանային և ցուցչային ֆունկցիաները՝ դրանք ևս հեշտությամբ կբերվեն գծային ֆունկցիայի տեսքի: Իրոք, աստիճանային ֆունկցիայի դեպքում կունենանք՝

$$\ln y = m \ln x + \ln a \ ,$$

իսկ ցուցչային ֆունկցիայի դեպքում՝

$$\ln y = mx + \ln a \quad \text{կամ} \quad \lg y = mx + \lg a \ :$$

Հակադարձ ֆունկցիան կբերվի գծային ֆունկցիայի տեսքի, եթե վերցվի այդ ֆունկցիայի հակադարձը, այսինքն՝

$$\frac{1}{y} = mx + a \ :$$

Այսպիսով՝ քննարկվող ֆունկցիաներով կարելի է մոտարկել **polyfit** ներդրված ֆունկցիայի օգնությամբ, եթե նրա արգումենտներն ունենան հետևյալ տեսքերը.

- Աստիճանային ֆունկցիա – **polyfit(log(x), log(y), 1)**,
- Ցուցչային ֆունկցիա – **polyfit(x, log(y), 1)** կամ **polyfit(x, log10(x), 1)**,
- Լոգարիթմական ֆունկցիա – **polyfit(log(x), y, 1)** կամ **polyfit(log10(x), y, 1)**,
- Հակադարձ ֆունկցիա – **polyfit(x, 1./y, 1)**:

Ինչպես սովորական գծային ֆունկցիայի դեպքում, այս ֆունկցիաների ելքը ևս կլինի երկու տարրից բաղկացած վեկտոր, որի առաջին տարրը կպարունակի m գործակցի արժեքը, իսկ երկրորդ տարրը՝ a գործակցի:

Այս ֆունկցիաների գրաֆիկական տեսքը ստանալու համար աստիճանային ֆունկցիաների համար կկիրառվի **loglog(x, y)** ֆունկցիան, որը x և y առանցքները պատկերում է լոգարիթմական մասշտաբներով, էքսպոնենցիալ ֆունկցիայի համար՝ **semilogy(x, y)** ֆունկցիան, որը x առանցքը պատկերում է գծային, իսկ y առանցքը՝ լոգարիթմական մասշտաբներով, լոգարիթմական ֆունկցիայի համար՝ **semilogx(x, y)** ֆունկցիան, որը x առանցքն է պատկերում լոգարիթմական մասշտաբով՝ y առանցքը թողնելով գծային մասշտաբով և, վերջապես, հակադարձ ֆունկցիայի համար՝ սովորական **plot(x, y)** ֆունկցիան:

Այս ֆունկցիաներով մոտարկելիս պետք է հաշվի առնել, որ

- աստիճանային ֆունկցիան հավասար է զրոյի $x = 0$ դեպքում,
- ցուցչային ֆունկցիան չի հատում x առանցքը, և նրա միջոցով կարելի է մոտարկում կատարել, եթե բոլոր y-ները կամ դրական են, կամ բացասական,
- լոգարիթմական ֆունկցիան չի կարող մոտարկել $x \le 0$ տիրույթի տվյալները,
- հակադարձ ֆունկցիան չի կարող մոտարկել $y = 0$ արժեքը:

7.5. ԻՆՏԵՐՊՈԼԱՑՈՒՄ

Ինչպես տեսանք, եթե երկու մեծությունների միջև կախվածությունը մեզ հայտնի է դիսկրետ տվյալների (աղյուսակային) տեսքով և հայտնի չէ այդ կախվածության ֆունկցիոնալ տեսքը, ապա կորերով մոտարկման եղանակով կարելի է գտնել այնպիսի ֆունկցիա, որը նվազագույն սխալով մոտարկում է տրված աղյուսակային ֆունկցիան: Առավել հաճախ որպես մոտարկային ֆունկցիա ընտրում են բազմանդամները: Եթե հայտնի է n դիսկրետ տվյալ, ապա միշտ գոյություն ունի այդ կետերով անցնող $(n-1)$ –րդ կարգի բազմանդամ: Ստացված բազմանդամը, հատկապես եթե բարձր կարգի է, թեև անցնում է աղյուսակային կետերով, սակայն դրանց արանքներում կարող է զգալիորեն տարբերվել իրական ֆունկցիոնալ կախվածությունից: Հետևաբար, բարձր կարգի մոտարկային բազմանդամները բոլոր կետերում անհայտ ֆունկցիայի արժեքները որոշելու համար հարմար չեն:

Այդ խնդիրը լուծելու համար կիրառում են մեկ այլ եղանակ, երբ մոտարկային ֆունկցիան կառուցելիս օգտագործվում են ոչ թե տվյալների բոլոր կետերը, այլ դրանք տրոհվում են ավելի քիչ թվով կետերից բաղկացած ենթաբազմությունների, դրանցից յուրաքանչյուրի համար կառուցվում են ավելի ցածր կարգի մոտարկային բազմանդամային ֆունկցիաներ, և սրանց օգնությամբ արդեն որոշվում են անհայտ կախվածության միջանկյալ արժեքները:

Նկարագրված եղանակը հայտնի է ինտերպոլացում անունով: Նշենք, որ որոշ գրքերում ինտերպոլացում են անվանում նաև 7.4. –ում նկարագրված կորերով մոտարկման եղանակը:

Համառոտ ներկայացնենք ինտերպոլացման մի քանի եղանակ և դրանց իրականացումը **MatLab**–ում:

7.5.1. Գծային ինտերպոլացում

Ինտերպոլացման ամենապարզ եղանակը գծային ինտերպոլացումն է, երբ դիսկրետ տվյալների բազմությունը տրոհվում է երկուական կետերից բաղկացած ենթաբազմությունների, և ենթադրվում է, որ հարևան կետերի միջև որոնվող ֆունկցիայի արժեքներն ընկած են այդ կետերը միացնող ուղղի հատվածի վրա: (x_i, y_i) և (x_{i+1}, y_{i+1}) հարևան կետերը միացնող ուղիղ գծի հավասարումն ունի

$$y_m = y_i + K(x_m - x_i)$$

տեսքը, որտեղ K –ն գծի թեքությունն է՝

$$K = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}:$$

MatLab–ում տվյալների ինտերպոլացման համար սահմանված է **interp1** ֆունկցիան: Գծային ինտերպոլացման համար **interp1** ֆունկցիան գրվում է հետևյալ կերպ.

ym = interp1(xi, yi, xm, ´ linear´)

կամ պարզապես

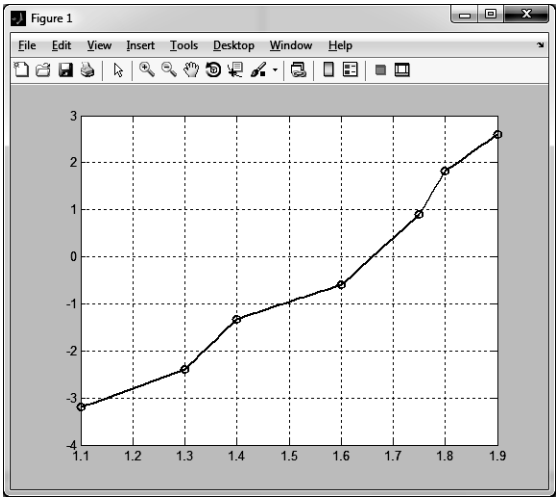
ym = interp1(xi, yi, xm)

տեսքով, որտեղ *xi*–ն և *yi*–ն դիսկրետ տվյալների կոորդինատները ներկայացնող նույն երկարության վեկտորներ են: *xm*–ը *xi* վեկտորի տիրույթին պատկանող կամայական երկարության վեկտոր է՝ կազմված այն կետերի կոորդինատներից, որոնցում ցանկանում ենք գտնել անհայտ ֆունկցիայի *ym* ինտերպոլացված արժեքները: *ym*–ն ունի նույն երկարությունը, ինչ *xm*–ը:

Օրինակ.

Ենթադրենք, տրված են (1.1, –3.2), (1.3, –2.4), (1.4, –1.32), (1.6, –0.6), (1.75, 0.9), (1.8, 1.82), (1.9, 2.6) կոորդինատներն ունեցող դիսկրետ տվյալները: *xi* կոորդինատը ընկած է 1.1–ից 1.9 տիրույթում: Գծային ինտերպոլացման միջոցով այս տիրույթում 0.01 քայլով որոշենք այս կախվածության միջանկյալ արժեքները և կառուցենք տրված կետերի և ինտերպոլացված կետերի գրաֆիկները (նկ. 53):

```
xi = [1.1 1.3 1.4 1.6 1.75 1.8 1.9];
yi = [-3.2 -2.4 -1.32 -0.6 0.9 1.82 2.6];
plot(xi,yi,'o')
xm = [1.1:0.01:1.9];
ym = interp1(xi,yi,xm);
hold on
plot(xm,ym)
grid on
```



Նկ. 53

7.5.2. Խորանարդային սպլայն ինտերպոլացում

Գծային ինտերպոլացման դեպքում, ինչպես տեսանք, յուրաքանչյուր երկու հարևան կետ իրար են միացվում ուղղի հատվածով: Ցանկացած ուղիղ գծի թեքությունը հաստատուն մեծություն է, կետից կետ ուղղի հատվածի ուղղությունը փոխվում է, և ստացվում է բեկյալ գիծ: Հետևաբար, գծային ինտերպոլացման դեպքում ողորկ կորեր չեն ստացվում: Ինտերպոլացման ողորկ կոր ստանալու համար կարելի է դիսկրետ տվյալների բազմության յուրաքանչյուր երկու հարևան կետ իրար միացնել ոչ թե ուղիղ գծով, այլ ավելի բարձր կարգի բազմանդամային ֆունկցիայով այնպես, որ յուրաքանչյուր կետում աջից ու ձախից մոտարկային ֆունկցիաներն ունենան նույն ածանցյալը: Այդ նպատակով օգտագործվում է խորանարդային բազմանդամը, իսկ ինտերպոլացման եղանակը անվանում են խորանարդային սպլայն ինտերպոլացում:

Յուրաքանչյուր (x_i, y_i) և (x_{i+1}, y_{i+1}) հարևան կետերը միացնող խորանարդային սպլայն ինտերպոլացման կորի հավասարումը ունի հետևյալ տեսքը.

$$y_m = a_1(x_m - x_i)^3 + a_2(x_m - x_i)^2 + a_3(x_m - x_i) + a_4:$$

Այս հավասարման մեջ $x_i \le x_m \le x_{i+1}$, իսկ a_1, a_2, a_3 և a_4 գործակիցներն ընտրվում են այնպես, որ բավարարեն հետևյալ երեք պայմաններին.

- Բազմանդամը պետք է անցնի x_i և x_{i+1} կետերով: $x_m = x_i$ կետում պետք է կատարվի $y_m = y_i$ պայմանը, այսինքն՝ $a_4 = y_i$,
- Երկու հարևան միջակայքերում խորանարդային բազմանդամների թեքությունները (առաջին կարգի ածանցյալները) ընդհանուր կետում պետք է հավասար լինեն,
- Երկու հարևան բազմանդամների կորությունները պետք է ընդհանուր կետում նույնը լինեն:

MatLab–ում խորանարդային սպլայն եղանակով տվյալներն ինտերպոլացնելու համար նորից օգտագործվում է **interp1** ներդրված ֆունկցիան, միայն 4–րդ արգումենտում տողի տեսքով պետք է գրել **spline**.

ym = interp1(x, y, xm, 'spline')

Միևնույն գործողությունը կարելի է կատարել ոչ միայն **interp1** ֆունկցիայով, այլև **spline** ներդրված ֆունկցիայով, որն ունի հետևյալ տեսքը.

ym = spline(x, y, xm)

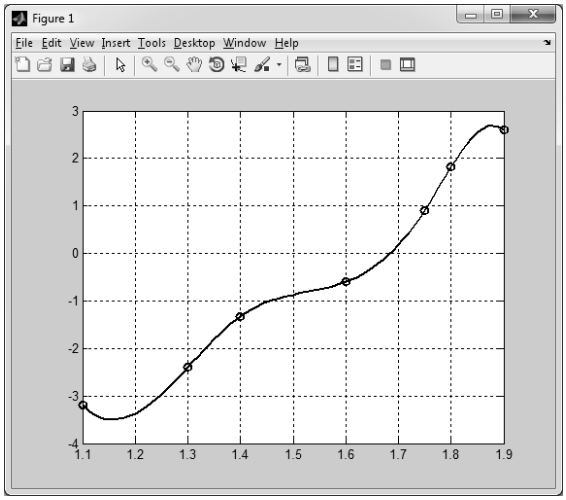
Այստեղ, ինչպես գծային ինտերպոլացման դեպքում *xi*–ն և *yi*–ն հայտնի դիսկրետ տվյալների կոորդինատներն են և անպայման պետք է ունենան նույն երկարությունը: *xm*–ը *xi* վեկտորի տիրույթին պարունակող կամայական երկարության վեկտոր է, որի տարրերը այն կետերի կոորդինատներն են, որոնցում ցանկանում ենք գտնել *ym* ինտերպոլացված արժեքները: Պարզ է, որ *ym* և *xm* վեկտորները նույն երկարությունն ունեն:

Օրինակ.

Նույնությամբ կրկնենք նախորդ օրինակում նկարագրված խնդիրը, միայն հիմա տվյալների ինտերպոլացումը կատարենք խորանարդային սպլայն եղանակով.

```
xi = [1.1 1.3 1.4 1.6 1.75 1.8 1.9];
yi = [–3.2 –2.4 –1.32 –0.6 0.9 1.82 2.6];
plot(xi,yi,'o')
xm = [1.1:0.01:1.9];
ym = spline(xi,yi,xm);
hold on
plot(xm,ym)
grid on
```

Համեմատելով նկ. 54–ում ստացված արդյունքները գծային ինտերպոլացման միջոցով ստացված, նկ. 53–ում պատկերված արդյունքների հետ, տեսնում ենք, որ խորանարդային սպլայն եղանակով ինտերպոլացման արդյունքում ստացված կորը բավական ողորկ է և ավելի լավ է նկարագրում տվյալների բազմությունը:



Նկ. 54

7.5.3. Ինտերպոլացում ըստ ամենամոտիկ հարևանների

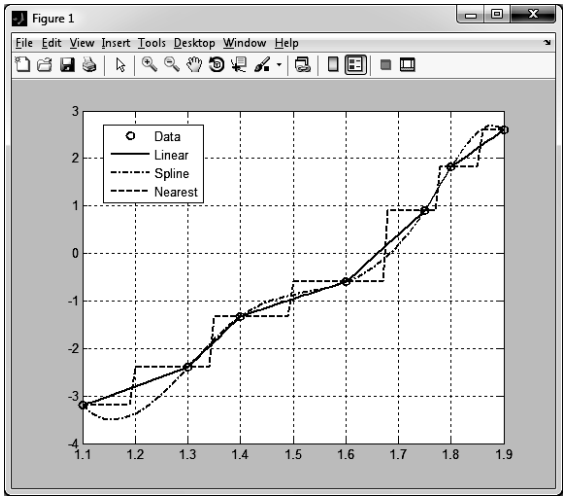
Դիսկրետ տվյալների ինտերպոլացման այս եղանակն ամենապարզն է: Այս դեպքում դարձյալ դիսկրետ տվյալների բազմությունը տրոհվում է երկու հարևան կետերից բաղկացած ենթաբազմությունների, և դրանցից յուրաքանչյուրի միջանկյալ կետերում, որպես անհայտ կախվածության արժեք, ընդունվում է աղյուսակային տվյալներից ամենամոտ գտնվող կետի արժեքը: Այս եղանակով ինտերպոլացում կատարելու համար օգտագործվում է նույն **interp1** ֆունկցիան, որի չորրորդ արգումենտում գրվում է 'nearest' տողը:

Օրինակ.

Դարձյալ դիտարկենք նույն օրինակը, ինչ նախորդ ինտերպոլացման եղանակները դիտարկելիս: Տրված են (1.1, –3.2), (1.3, –2.4), (1.4, –1.32), (1.6, –0.6), (1.75, 0.9), (1.8, 1.82), (1.9, 2.6) կոորդինատներն ունեցող դիսկրետ տվյալները: 1.1–ից 1.9 միրույթում 0.01 քայլով որոշենք այս կախվածության միջանկյալ արժեքները միաժամանակ գծային, խորանարդային սպլայններով և ըստ ամենամոտ հարևանների, կառուցենք տրված կետերի և ինտերպոլացված կետերի գրաֆիկները:

```
xi = [1.1 1.3 1.4 1.6 1.75 1.8 1.9];
yi = [–3.2 –2.4 –1.32 –0.6 0.9 1.82 2.6];
plot(xi,yi,'o')
xm = [1.1:0.01:1.9];
ym_linear = interp1(xi,yi,xm);
ym_spline = spline(xi,yi,xm);
ym_nearest = interp1(xi,yi,xm,'nearest');
hold on
plot(xm,ym_linear)
plot(xm,ym_spline,'–.')
plot(xm,ym_nearest,'––')

legend('Data', 'Linear', 'Spline', 'Nearest')
grid on
```

Ծրագրի արդյունքը պատկերված է նկ. 55–ում:



Նկ. 55

7.6. ԹՎԱՅԻՆ ԻՆՏԵԳՐՈՒՄ

Ինչպես հայտնի է, $y = f(x)$ ֆունկցիայի $I = \int_a^b f(x)dx$ որոշյալ ինտեգրալը երկրաչափորեն $f(x)$ ֆունկցիայի գրաֆիկով, $x = a$, $x = b$ ուղիղներով և x առանցքով սահմանափակված տիրույթի մակերեսն է: Որոշյալ ինտեգրալները հաճախ հնարավոր է հաշվել անալիտիկորեն, սակայն քիչ չեն հանդիպում այնպիսի ֆունկցիաներ, որոնց ինտեգրալները անալիտիկ եղանակով հաշվել հնարավոր չէ: Բացի այդ, երբեմն ենթաինտեգրալային ֆունկցիաները տրված են լինում աղյուսակային տեսքով: Այսպիսի ֆունկցիաների ինտեգրալների մոտավոր արժեքները հաշվելու համար գոյություն ունեն տարբեր թվային եղանակներ:

MatLab–ում սահմանված են ինտեգրալների անալիտիկ և թվային եղանակներով հաշվելու համար նախատեսված մի շարք ներդրված ֆունկցիաներ: Անալիտիկ եղանակով որոշյալ ինտեգրալ հաշվելուն կծանոթանանք 8–րդ գլխում, իսկ այս գլխում համառոտ կներկայացնենք ինտեգրալներ հաշվելու մի քանի թվային եղանակ և դրանցից յուրաքանչյուրի համար նախատեսված **MatLab**–ի համապատասխան ֆունկցիան:

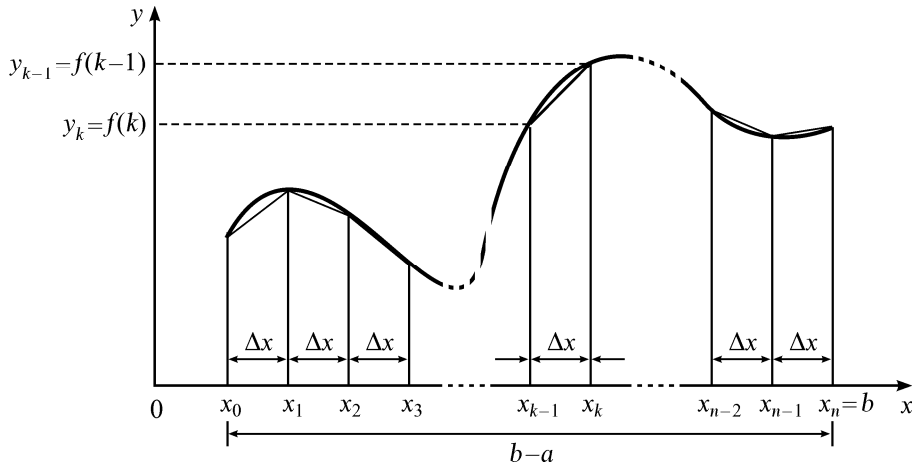
Ակնհայտ է, որ եթե ենթինտեգրալային ֆունկցիայի տեսքը հայտնի է, ապա նախ անհրաժեշտ է դիմել անալիտիկ եղանակին, որը կվերադարձնի ճշգրիտ արդյունք, եթե այդպիսին գոյություն ունի: Եթե անալիտիկ լուծում գտնել չի հաջողվում, նոր միայն պետք է դիմել թվային եղանակներին:

7.6.1. Մեղանների եղանակ

$y = f(x)$–ի ինտեգրման $[a,b]$ տիրույթը $[x_0 = a, x_1, ..., x_k, x_{k+1}, ..., x_n = b]$ կետերով տրոհենք n հավասար հատվածների: Այս հատվածներից յուրաքանչյուրի երկարությունը $\Delta x = (b-a)/n$ է: Ինտեգրալի մոտավոր արժեքը կարող ենք որոշել, եթե հարևան $(x_{k-1}, f(x_{k-1}))$ և $(x_k, f(x_k))$ կետերը միացնենք ուղղի հատվածներով, այնուհետև հաշվենք այդ հատվածներով, $x = x_k$, $x = x_{k+1}$ ուղիղներով ու $y = 0$ առանցքով սահմանափակված սեղանների մակերեսները և գումարենք իրար (նկ. 56):

k–րդ սեղանի մակերեսը հավասար է՝

$$I_k \approx \frac{\Delta x}{2}(f(x_{k-1}) + f(x_k)):$$



Նկ. 56

Գումարելով այս սեղանների մակերեսները՝ կստանանք $[a,b]$ միջակայքում $y=f(x)$ ֆունկցիայի ինտեգրալի մոտավոր արժեքը.

$$I \approx \sum_{k=1}^{n} \frac{\Delta x}{2}(f(x_{k-1})+f(x_k)) = \frac{\Delta x}{2}\left[f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{n-1})+f(x_n)\right]:$$

Ակնհայտ է, որ ինչքան n –ը (տրոհված հատվածների քանակը) մեծացնենք, այնքան որոշյալ ինտեգրալի արժեքն առավել ճշգրիտ կլինի:

MatLab–ում սեղանների եղանակով $y(x)$ ֆունկցիայի որոշյալ ինտեգրալը հաշվելու համար նախատեսվում է **trapz(x,y)** ներդրված ֆունկցիան, որի արգումենտում x և y վեկտորները պետք է ունենան միևնույն երկարությունը:

Օրինակ.

Հաշվենք $I=\int_0^4 \sin^2 x dx$ որոշյալ ինտեգրալը: $\sin^2 x$ ֆունկցիայի որոշյալ ինտեգրալն ունի անալիտիկ լուծում, և (0, 4) տիրույթում նրա թվային արժեքը 1.7526604383 է: Օգտագործելով **trapz** ֆունկցիան՝ հաշվենք այս ինտեգրալի մոտավոր արժեքը և տեսնենք, թե ինչ ճշտությամբ է համընկնում անալիտիկ լուծման հետ:

Նախ բաժանումների թիվը փոքր վերցնենք: Ենթադրենք, x–ը 0.5 քայլով փոխվում է 0–ից մինչև 4:

```
>> x = [0:0.5:4];
>> y = sin(x).^2;
>> I = trapz(x,y)
   I = 1.773623984647617
```

Ինչպես տեսնում ենք, սեղանների եղանակով թվային լուծումը անալիտիկ լուծման հետ համընկնում է ստորակետից հետո մեկ նիշի ճշտությամբ: Եթե բաժանումների թիվը մեծացնենք՝ 0–ից 4 տիրույթում քայլը դարձնելով 0.1, կունենանք.

```
>> x = [0:0.1:4];
>> y = sin(x).^2;
>> I = trapz(x,y)
   I = 1.753485453717139
```

Այս դեպքում արդեն թվային լուծումը անալիտիկ լուծման հետ համընկնում է՝ ստորակետից հետո 2 նիշի ճշտությամբ: Այժմ քայլն ավելի փոքրացնենք՝ ընտրելով այն հավասար 0.001–ի: Ինչպես երևում է ծրագրի արդյունքից՝ թվային լուծումը ճշգրիտ լուծման հետ համընկնում է արդեն 6 նիշի ճշտությամբ.

```
>> x = [0:0.001:4];
>> y = sin(x).^2;
>> I = trapz(x,y)
   I = 1.752660520790681
```

MatLab–ը հնարավորություն է տալիս սեղանների եղանակով հաշվել նաև փոփոխական վերին սահմանով $I = \int_a^x f(u)du$ ինտեգրալը: Դրա համար նախատեսված է **cumtrapz(x,y)** ներդրված ֆունկցիան: Եթե տրված են [*a*, *x*] տիրույթում կետերի քանակը և այդ կետերից յուրաքանչյուրի համար *f*(*u*) ֆունկցիայի արժեքները, ապա ինտեգրումը կատարվում է հաջորդաբար՝ [*a*, *x*] միջակայքի յուրաքանչյուր *x* վերին սահմանի համար և վերադարձվում է վեկտոր, որը պարունակում է յուրաքանչյուր ինտեգրման արդյունքը:

Օրինակ.

Ենթադրենք, անհրաժեշտ է հաշվել $I = \int_1^x \sin x dx$ ինտեգրալի արժեքը, որտեղ *x*–ը փոփոխվում է 1–ից 8 տիրույթում: Օգտագործելով **cumtrapz** ֆունկցիան՝ կարող ենք հաշվել այս ինտեգրալի արժեքները *x*–ի՝ 1, 2, ..., 8 արժեքների համար և յուրաքանչյուր *x*–ի համար ստանալով համապատասխան արժեքը:

```
>> x = [1:8];
>> y = sin(x);
>> I = cumtrapz(x,y)
   I =
      0  0.8754  1.4006  1.0928  0.2349  -0.3843  -0.1955   0.6277
```

7.6.2. Սիմպսոնի եղանակը

Որոշյալ ինտեգրալների թվային ինտեգրման հիմնական եղանակներից է Սիմպսոնի կամ պարաբոլների եղանակը: $I=\int_a^b f(x)dx$ ինտեգրալը այս եղանակով հաշվելիս ինտեգրման $[a,\,b]$ տիրույթը $[x_0=a, x_1, ..., x_k, x_{k+1}, ..., x_{2n-1}, x_{2n}=b]$ կետերով տրոհում են $2n$ զույգ թվով $h=(b-a)/2n$ երկարությամբ հատվածների: Երեք կետ պարունակող յուրաքանչյուր $[x_{2k},\,x_{2k+2}]$ $(k=0,\,1,\,...,\,n-1)$ հատվածում ենթինտեգրալային ֆունկցիան փոխարինվում է $(x_{2k},\,f(x_{2k}))$, $(x_{2k+1},\,f(x_{2k+1}))$, $(x_{2k+2},\,f(x_{2k+2}))$ կետերով անցնող $y=p_0+p_1x+p_2x^2$ պարաբոլով (նկ. 57), որի p_0, p_1, p_2 գործակիցները յուրաքանչյուր տեղամասում գտնելու համար այդ երեք կետերի կոորդինատներն անհրաժեշտ է տեղադրել պարաբոլի հավասարման մեջ և լուծել p_i գործակիցների համար ստացված երեք հանրահաշվական հավասարումների համակարգը: Արդյունքում ստացվում է՝

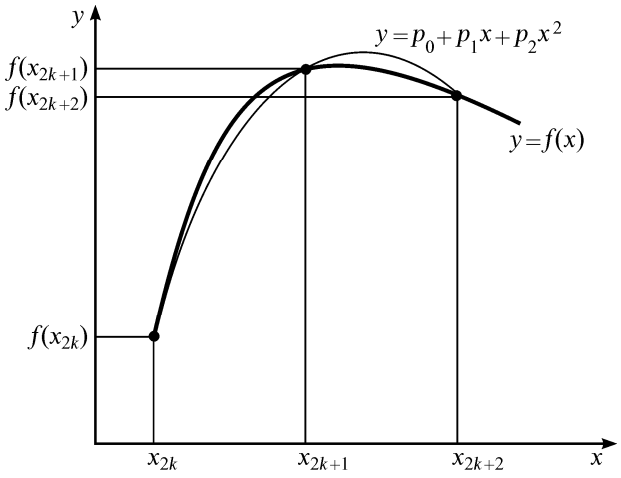
$$p_0=f(x_{2k}),\quad p_1=\frac{-3f(x_{2k})+4f(x_{2k+1})-f(x_{2k+2})}{2h},$$
$$p_2=\frac{f(x_{2k})-2f(x_{2k+1})+f(x_{2k+2})}{2h^2}:$$

$f(x)$ ֆունկցիան պարաբոլով փոխարինելը հնարավորություն է տալիս յուրաքանչյուր $[x_{2k},\,x_{2k+2}]$ տեղամասում $I_{2k}=\int_{x_{2k}}^{x_{2k+2}} f(x)dx$ ինտեգրալը փոխարինել $I_{2k}\approx\int_{x_{2k}}^{x_{2k+2}}(p_0+p_1x+p_2x^2)dx$ հեշտ հաշվվող ինտեգրալով: Հաշվի առնելով p_i գործակիցների համար ստացված արտահայտությունները՝ կունենանք.

$$I_{2k}\approx\frac{h}{3}(f(x_{2k})+4f(x_{2k+1})+f(x_{2k+2})):$$

Բոլոր տեղամասերում հաշվված I_{2k} մեծությունների գումարը կտա որոնվող I ինտեգրալը հաշվելու մոտավոր բանաձևը՝

$$I\approx\sum_{k=1}^{n-1}I_{2k}=\frac{h}{3}(f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+...+2f(x_{2n-1})+f(x_{2n})):$$



Նկ. 57

Այս առնչությունը կոչվում է Սիմպսոնի բանաձև: Ցույց է տրվում, որ Սիմպսոնի եղանակով հաշվման սխալի մեծությունը h^4–ի կարգի է, հետևաբար, այս եղանակով որոշյալ ինտեգրալներն ավելի ճշգրիտ են որոշվում, քան սեղանների եղանակով, որի դեպքում սխալի մեծությունը $\sim h^2$:

Սիմպսոնի եղանակով որոշյալ ինտեգրալ հաշվելու համար նախատեսված է **quad(′function′, a, b)** ներդրված ֆունկցիան: Այստեղ *function*–ը ներդրված կամ օգտագործողի կողմից սահմանված ֆունկցիայի անունն է՝ գրված տողի տեսքով, իսկ *a*–ն ու *b*–ն ինտեգրման ստորին և վերին սահմաններն են:

Օրինակ.

Սիմպսոնի եղանակով հաշվենք $I=\int_0^4 \sin^2 x dx$ ինտեգրալի արժեքը:

```
>> I = quad('sin(x).^2', 0, 4)
   I = 1.752660414250969
```

7.6.3. Կրկնակի ինտեգրալներ

Սիմպսոնի եղանակով կարելի է հաշվել նաև երկու փոփոխականից կախված ֆունկցիաների կրկնակի ինտեգրալները: Կրկնակի ինտեգրալ հաշվելու համար **MatLab**–ում նախատեսված է **dblquad** ֆունկցիան:

$I=\int_a^b\int_c^d f(x,y)dydx$ տեսքի կրկնակի ինտեգրալը հաշվելու համար **dblquad** ֆունկցիան գրվում է հետևյալ տեսքով.

I = dblquad('function',a,b,c,d)

Այստեղ *functon*–ը երկու փոփոխականից կախված ֆունկցիայի անունն է՝ գրված տողի տեսքով, իսկ *a*, *b*–ն և *c*, *d*–ն՝ համապատասխանաբար *x* և *y* փոփոխականների ստորին ու վերին սահմանները:

Օրինակ.

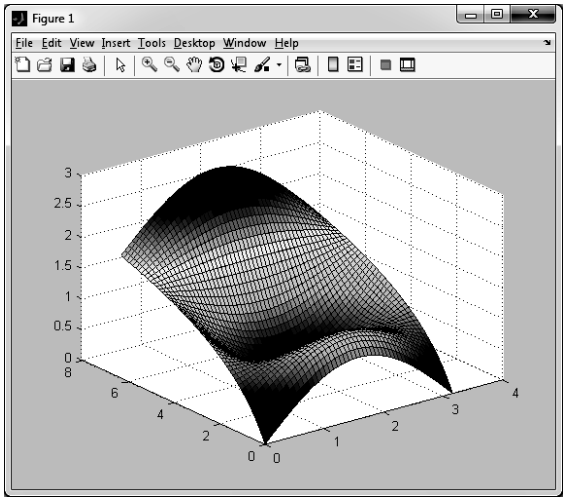
Հաշվենք $I = \int_0^{\pi}\int_0^{2\pi} (\sin x \cos y + \ln(1+y))dydx$ կրկնակի ինտեգրալը:

Նախ առանձին ֆայլում սահմանենք $f(x,y)$ ֆունկցիան.

```
function z = mfun(x, y)
    z = sin(x) .* cos(y) + log(y+1);
end
```

Այժմ հրամանի պատուհանում կամ M–ֆայլում սահմանենք *x* և *y* փոփոխականները անհրաժեշտ տիրույթում, կառուցենք մակերևութային ցանցը և հայտարարենք $z(x,y)$ ֆունկցիան: Այնուհետև կառուցենք սահմանված ֆունկցիայի մակերևույթը (նկ. 58) և հաշվենք ինտեգրալը.

```
>> x = linspace(0,pi,50);
>> y = linspace(0,2*pi,50);
>> [X,Y] = meshgrid(x,y);
>> Z = mfun(X,Y);
>> surf(X,Y,Z)
>> I = dblquad('mfun',0,pi,0,2*pi)
   I = 25.692185406378115
```



Նկ. 58

7.7. ԴԻՖԵՐԵՆՑԻԱԼ ՀԱՎԱՍԱՐՈՒՄՆԵՐԻ ԼՈՒԾՈՒՄԸ

Բազմաթիվ բնագավառների խնդիրներ, մասնավորապես, գիտության և տեխնիկայի զանազան խնդիրներ հաճախ հանգում են դիֆերենցիալ հավասարումների կամ դիֆերենցիալ հավասարումների համակարգերի լուծմանը: Այսպիսով, դիֆերենցիալ հավասարումներն առանցքային դեր են խաղում այս բնագավառներում: Հաճախ հնարավոր է լինում դիֆերենցիալ հավասարումները լուծել անալիտիկորեն, սակայն քիչ չեն դեպքերը, երբ հավասարումը չի ունենում ճշգրիտ լուծում և անհրաժեշտություն է առաջանում դիմել թվային եղանակների օգնությանը: Մշակված են մի շարք թվային եղանակներ, որոնք թույլ են տալիս լուծել դիֆերենցիալ հավասարումները: Սակայն սխալ կլինի մտածել, որ ցանկացած դիֆերենցիալ հավասարման կամ դիֆերենցիալ հավասարումների համակարգի թվային եղանակներով լուծումը հեշտ խնդիր է: Պատճառն այն է, որ գոյություն չունի մշակված այնպիսի թվային եղանակ, որը թույլ կտա լուծել կամայական դիֆերենցիալ հավասարում: Անհրաժեշտ է հստակ պատկերացում ունենալ, թե տրված դիֆերենցիալ հավասարումը լուծելու համար որ թվային եղանակն է ավելի արդյունավետ: **MatLab**–ը պարունակում է մի շարք ներդրված ֆունկցիաներ, որոնք թույլ են տալիս տարբեր եղանակներով լուծել դիֆերենցիալ հավասարումները:

Այս ձեռնարկի նպատակը դիֆերենցիալ հավասարումների լուծման բոլոր թվային եղանակները մանրամասն նկարագրելը չէ: Մենք կսահմանափակվենք ամենից հաճախ կիրառվող թվային եղանակների համառոտ նկարագրությամբ, փոխարենը կներկայացնենք տարբեր թվային եղանակների վրա հիմնված **MatLab**–ի ներդրված ֆունկցիաների ցանկը և կնշենք, թե որ տիպի դիֆերենցիալ հավասարումների լուծման համար որ ֆունկցիան է ամենից հարմար կիրառել:

Հարկ է նկատի ունենալ, որ **MatLab**–ն ունի դիֆերենցիալ հավասարումների լուծման անալիտիկ (ճշգրիտ) լուծման համար նախատեսված ֆունկցիա ևս: Ցանկացած դիֆերենցիալ հավասարում նախ ցանկալի է փորձել լուծել անալիտիկ եղանակով: Եթե **MatLab**–ը չի գտնի այդ հավասարման ճշգրիտ լուծում, ապա նոր միայն անհրաժեշտ կլինի դիմել թվային եղանակների կիրառությանը: Դիֆերենցիալ հավասարումների անալիտիկ լուծման **MatLab**–ի ֆունկցիաները շարադրված են 8-րդ գլխում: Այս գլխում կդիտարկենք միայն թվային եղանակները:

Սովորական դիֆերենցիալ հավասարումը (Ordinary differential equation, կրճատ՝ ODE) այնպիսի հավասարում է, որը պարունակում է անկախ փոփոխական, այդ փոփոխականից կախված այլ փոփոխական և վերջինիս որևէ կարգի ածանցյալ: Մաթեմատիկորեն առաջին կարգի սովորական դիֆերենցիալ հավասարումը շատ հաճախ կարելի է ներկայացնել

$$\frac{dy}{dt} = f(t, y) \tag{7.1}$$

տեսքով, որտեղ t –ն և y –ը համապատասխանաբար անկախ և կախված փոփոխականներն են: Լուծել (7.1) հավասարումը՝ նշանակում է գտնել այնպիսի $y = y(t)$ ֆունկցիա, որը կբավարարի այդ հավասարմանը: (7.1) հավասարումն ունի անվերջ թվով լուծումներ: Յուրաքանչյուր կոռեկտ ձևակերպած խնդրի դեպքում $y(t)$ ֆունկցիայի վրա դրվում է լրացուցիչ պայման (կամ պայմաններ), որը թույլ է տալիս (7.1) հավասարման հնարավոր բոլոր լուծումներից ընտրել տրված խնդրին համապատասխան միակ լուծումը: Այսպիսի պայմանը կոչվում է սկզբնական պայման և կարելի է ներկայացնել հետևյալ տեսքով.

$$y(t_0) = y_0 : \tag{7.2}$$

Այս խնդիրն այլ կերպ անվանում են Կոշու խնդիր:

7.7.1. Առաջին կարգի Ռունգե–Կուտտայի եղանակ

Սովորական դիֆերենցիալ հավասարումների լուծման այս եղանակը հարմար է, որ իր լուծման յուրաքանչյուր y_{i+1} կետը որոշելու համար բավական է իմանալ միայն նախորդ՝ y_i կետի արժեքը: Այսպիսի եղանակները կոչվում են մի քայլանի եղանակներ:

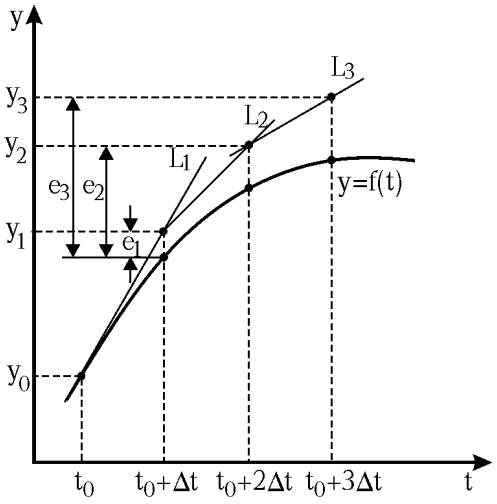
Առաջին կարգի Ռունգե–Կուտտայի եղանակի սկզբունքը հարմար է նկարագրել գրաֆիկորեն (նկ. 59): Ենթադրենք՝ ցանկանում ենք լուծել (7.1) հավասարումը (7.2) սկզբնական պայմանով, այսինքն՝ գտնել հավասարմանը բավարարող $y = F(t)$ միակ կորը: Այս կորի վրա սկզբնական պայմանից մեզ հայտնի է (t_0, y_0) կետը: Որպես դիֆերենցիալ հավասարման լուծման առաջին քայլ՝ այդ կետով տանենք $f(t)$ ֆունկցիայի L_1 շոշափողը: Վերջինիս հավասարումն ունի $y = y_0 + K(t - t_0)$ տեսքը, որտեղ, ինչպես հայտնի է, $K = y_0' = f(t_0, y_0)$: Հաջորդ y_1 կետը կընտրենք $t = t_0 + \Delta t$ և L_1 շոշափողի հատման կետում, այսինքն՝ $y_1 = y_0 + \Delta t \cdot f(t_0, y_0)$:

Այս քայլում սխալի արժեքը հավասար է e_1 –ի: Այժմ (t_1, y_1) կետով տանենք երկրորդ, L_2 շոշափողը, որի հավասարումն է

$$y_2 = y_1 + \Delta t \cdot f(t_1, y_1) :$$

Երկրորդ քայլի սխալի արժեքը հավասար է e_2 –ի: Նման ձևով որոշելով մնացած կետերը՝ կստանանք.

$$y_{i+1} = y_i + \Delta x \cdot f(t_i, y_i) :$$



Նկ. 59

Ինչպես տեսնում ենք, յուրաքանչյուր y_{i+1} կետի արժեքը որոշվում է միայն y_i կետի արժեքն իմանալով: Սա նշանակում է, որ Ռունգե–Կուտտայի առաջին կարգի եղանակը մի քայլանի եղանակ է: Այս եղանակի թերությունն այն է, որ յուրաքանչյուր քայլում e_i սխալը մեծանում է, և արդյունքում լուծման ճշտությունը փոքրանում է:

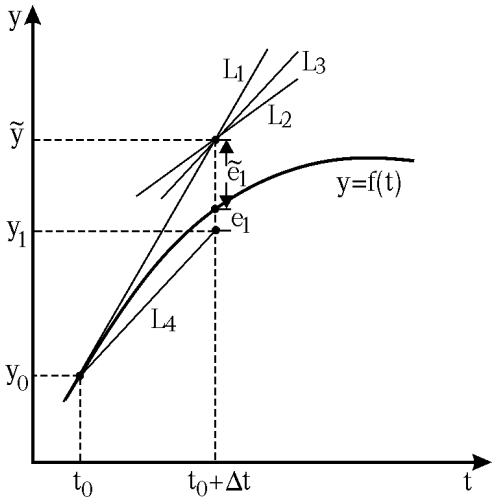
7.7.2. Երկրորդ կարգի Ռունգե–Կուտտայի եղանակ

Առաջին կարգի Ռունգե–Կուտտայի եղանակի դեպքում ի հայտ եկող կուտակվող սխալը փոքրացնելու համար կարելի է վարվել հետևյալ կերպ: Դարձյալ սկզբնական պայմանից հայտնի (t_0, y_0) կետով տարվում է L_1 շոշափողը (նկ. 60): Հաջորդ կետը, որն ընտրում ենք $t_1 = t_0 + \Delta t$ կետի և L_1 շոշափողի հատման կետում, այս անգամ նշանակենք $\tilde{y}$ –ով և դրանով նախկինի պես տանենք L_2 շոշափողը: Այժմ ընտրենք L_3 ուղիղն այնպես, որ իր անկյունային գործակիցը հավասար լինի L_1 և L_2 շոշափողների անկյունային գործակիցների միջին արժեքին՝

$$K_{L_3} = \frac{K_{L_1} + K_{L_2}}{2}:$$

Որոշ ձևափոխություններից հետո կարելի է ցույց տալ, որ

$$K_{L_3} = \frac{1}{2}\big(f(t_0, y_0) + f(t_0 + \Delta t, y_0 + \Delta t \cdot f(t_0, y_0))\big):$$



Նկ. 60

Այժմ (t_0, y_0) կետով տանենք L_4 ուղիղը, որը զուգահեռ կլինի որոշված L_3 ուղղին: Որպես փնտրվող (t_1, y_1) կետ կընտրենք $t_1 = t_0 + \Delta t$ կետի և L_4 ուղղի հատման կետը.

$$y_1 = y_0 + \frac{\Delta t}{2}\big(f(t_0, y_0) + f(t_0 + \Delta t, y_0 + \Delta t \cdot f(t_0, y_0))\big):$$

Նույն կերպ որոշելով մնացած կետերը՝ ընդհանուր դեպքում կստանանք.

$$y_{i+1} = y_i + \frac{\Delta t}{2}\big(f(t_i, y_i) + f(t_i + \Delta t, y_i + \Delta t \cdot f(t_i, y_i))\big):$$

Այս եղանակի օգնությամբ փոքրանում է Ռունգե–Կուտտայի առաջին կարգի եղանակի դեպքում առաջացող կուտակվող սխալը: Ճշգրտված այս եղանակը կոչվում է Ռունգե–Կուտտայի երկրորդ եղանակ:

7.7.3. Չորրորդ կարգի Ռունգե–Կուտտայի եղանակ

Առանց մանրամասն արտածումների, գրենք այն արտահայտությունները, որոնք նկարագրում են չորրորդ կարգի Ռունգե–Կուտտայի եղանակը.

$$y_{i+1} = y_i + \frac{\Delta t}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

որտեղ

$$K_1 = f(t_i, y_i), \quad K_2 = f\left(t_i + \frac{\Delta t}{2}, y_i + K_1\frac{\Delta t}{2}\right),$$

$$K_3 = f\left(t_i + \frac{\Delta t}{2}, y_i + K_2\frac{\Delta t}{2}\right), \quad K_4 = f\big(t_i + \Delta t, y_i + K_3\Delta t\big):$$

Տեսնենք՝ ինչպես է որոշվում (t_{i+1}, y_{i+1}) կետը: Նախ $A_0(t_i, y_i)$ կետով, α_1 անկյան տակ ($\operatorname{tg}\alpha_1 = K_1$) տարվում է L_1 ուղիղը: Այնուհետև որոշվում է L_1 ուղղի և $t_i + \frac{\Delta t}{2}$ կետի օրդինատի հատման A_1 կետը: Այժմ դարձյալ A_0 կետից տարվում է L_2 ուղիղը α_2 անկյան տակ ($\operatorname{tg}\alpha_2 = K_2$) և փնտրվում է L_2 ուղղի և $t_i + \frac{\Delta t}{2}$ կետի օրդինատի հատման A_3 կետը: Այստեղից որոշվում է α_3 անկյունը՝ $\operatorname{tg}\alpha_3 = K_3$: Դարձյալ A_0 կետից, այս անգամ α_3 անկյան տակ տարվում է L_3 ուղիղը և որոշվում է այդ ուղղի և $t_i + \Delta t$ կետի օրդինատի հատման կետը A_4: Այստեղից կորոշվի α_4 անկյունը՝ $\operatorname{tg}\alpha_4 = K_4$: Ստացված տանգենսների արժեքները միջինացնելով՝ չորրորդ կարգի Ռունգե–Կուտտայի բանաձևից կորոշվի (t_{i+1}, y_{i+1}) կետը:

Չորրորդ կարգի Ռունգե–Կուտտայի եղանակով հաշվարկի սխալը որոշվում է կրկնակի հաշվարկի միջոցով.

$$\varepsilon \approx \frac{|\tilde{y}_n - y_n|}{15}:$$

Այստեղ $\tilde{y}_n$–ը $\frac{\Delta t}{2}$ քայլով որոշվող արժեքն է, իսկ y_n–ը՝ Δt քայլով որոշվող արժեքը:

7.7.4. Բազմաքայլ եղանակներ

Ինչպես արդեն նշել ենք, Ռունգե–Կուտտայի եղանակները մի քայլանի եղանակներ են, քանի որ յուրաքանչյուր y_{i+1} կետը որոշելու համար բավական էր իմանալ միայն նախորդ y_i կետի արժեքը: Ի տարբերություն մի քայլանի եղանակների, գոյություն ունեն սովորական դիֆերենցիալ հավասարումների լուծման այնպիսի թվային եղանակներ, որոնց դեպքում հերթական y_{i+1} կետը որոշելու համար անհրաժեշտ է իմանալ նախորդ մի քանի կետերում լուծումների արժեքները: Այսպիսի եղանակները կոչվում են բազմաքայլ: Բազմաքայլ եղանակները կիրառելու համար (7.1) հավասարումը ինտեգրվում է $[t_i, t_{i+1}]$ միջակայքում.

$$\int_{t_i}^{t_{i+1}} \frac{dy}{dt}\,dt = \int_{t_i}^{t_{i+1}} f(t,y)\,dt\,,$$

որտեղից

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t,y)\,dt:$$

$y(t)$ ֆունկցիան որոշելու համար առավել լայն տարածում է գտել Ադամս–Բեշֆորդ–Միլտոնի եղանակը: Այստեղ չենք շարադրի այդ եղանակը: Կնշենք միայն, որ այս եղանակի դեպքում ենթաինտեգրալային արտահայտությունների նկատմամբ կիրառվում է բազմանդամային ինտերպոլացում, որից հետո ստացված բազմանդամն ինտեգրվում է: Ադամս–Բեշֆորդ–Միլտոնի եղանակն առավել արդյունավետ է, քան Ռունգե–Կուտտայի եղանակները, քանի որ այն ապահովում է ավելի բարձր կարգի ճշտություն:

7.7.5. Կոշտ դիֆերենցիալ հավասարումներ

Երբեմն հանդիպում են այնպիսի դիֆերենցիալ հավասարումներ կամ հավասարումների համակարգեր, որոնց նկատմամբ սովորական եղանակները (ինչպես, օրինակ, Ռունգե–Կուտտայի եղանակը) կիրառելի չեն: Առաջին անգամ այսպիսի հավասարումների նկատմամբ հետաքրքրությունն առաջացել է 20-րդ դարի կեսերին, քիմիական կինետիկայի այնպիսի խնդիրներ լուծելիս, երբ միաժամանակ ընթանում էին շատ դանդաղ և շատ արագ քիմիական պրոցեսներ: Այսպիսի խնդիրները բերվում էին այնպիսի դիֆերենցիալ հավասարումների, որոնք ճիշտ արդյունք չէին վերադարձնում Ռունգե–Կուտտայի եղանակներով լուծելիս: Այսպիսով, պարզ դարձավ, որ այն եղանակները, որոնք մինչ այդ համարվում էին շատ կայուն, միշտ չէ, որ արդյունավետ են: Այսպիսի դիֆերենցիալ հավասարումները կոչվում են կոշտ:

Կոշտ դիֆերենցիալ հավասարման օրինակ է $\frac{dy}{dt} = -A(y - \cos t)$ հավասարումը: Եթե այն լուծվի Ռունգե–Կուտտայի եղանակներով, ապա կախված A պարամետրի արժեքներից՝ այն կարող է վերադարձնել ինչպես ճիշտ, այնպես էլ՝ սխալ արդյունք: Օրինակ, եթե $A = 10$, ապա խնդիրը նորմալ լուծվում է, սակայն $A = -40$ դեպքում լուծման տեսքն անճանաչելիորեն փոխվում է: Սա նշանակում է, որ լուծման ալգորիթմը կայուն չէ: Խնդիրը կարելի է լուծել, եթե Ռունգե–Կուտտայի եղանակի քայլերն ավելացնենք այնքան, մինչև որ լուծումը կայունանա: Այսպիսով, տեսնում ենք, որ միևնույն դիֆերենցիալ հավասարումը տարբեր պարամետրերի դեպքում կարող է լինել ինչպես կոշտ, այնպես էլ ոչ կոշտ: Տեսնում ենք նաև, որ թվային եղանակի քայլն ավելացնելով՝ լուծումը հնարավոր է կայունացնել: Սակայն դա ևս ոչ միշտ է հարմար: Վերը բերված հավասարումը շատ կոշտ չէ, և քայլերի քանակը շատ մեծ թվով պարտադիր չէ ավելացնել: Գոյություն ունեն նաև շատ կոշտ դիֆերենցիալ հավասարումներ, որոնց դեպքում լուծումը կայունանում է, եթե քայլերի քանակը ավելացնենք մինչև միլիոն, միլիարդ կամ նույնիսկ ավելի անգամ, ինչը կբերի հսկայական քանակով ռեսուրսների ծախսման:

Գոյություն ունեն կոշտ դիֆերենցիալ հավասարումների լուծման մի շարք թվային եղանակներ, որոնցից առավել լայն տարածում ունի Ռոզենբրոկի մի քայլանի եղանակը:

7.7.6. Դիֆերենցիալ հավասարումների լուծումը MatLab–ի օգնությամբ

MatLab–ում սահմանված են մի շարք ներդրված ֆունկցիաներ, որոնք նախատեսված են տարբեր թվային եղանակներով սովորական դիֆերենցիալ հավասարումները լուծելու համար: Այս ֆունկցիաներին անվանում են նաև սոլվերներ (solver): Ստորև բերված է սոլվերների նկարագրությունը:

- **ode45** ֆունկցիան հիմնված է չորրորդ կարգի Ռունգե–Կուտտայի եղանակի վրա: Դիֆերենցիալ հավասարում լուծելիս ճիշտ է առաջին հերթին օգտվել այս ֆունկցիայից, քանի որ շատ հաճախ այն վերադարձնում է ճիշտ արդյունք:
- **ode23** ֆունկցիան հիմնված է երկրորդ կարգի Ռունգե–Կուտտայի եղանակի վրա: Այն ավելի արագ է լուծում դիֆերենցիալ հավասարումները, սակայն չունի նույն ճշտությունը, ինչ **ode45** ֆունկցիան: Այս ֆունկցիան հարմար է օգտագործել այն դեպքորում, երբ կարիք չկա ստանալ մեծ ճշտությամբ լուծումներ:
- **ode113** ֆունկցիան հիմնված է Ադամս–Բեշֆորդ–Միլտոնի բազմաքայլ եղանակի վրա: Այն հարմար է օգտագործել ոչ կոշտ դիֆերենցիալ հավասարումներ լուծելիս, երբ անհրաժեշտ է ստանալ մեծ ճշտությամբ լուծումներ: Այն նաև նախընտրելի է կիրառել այնպիսի ոչ կոշտ դիֆերենցիալ հավասարումներ կամ դրանց համակարգեր լուծելիս, երբ դրանց աջ մասերը պարունակում են բարդ ֆունկցիաներ:
- **ode15s** ֆունկցիան բազմաքայլ եղանակ է, որը նախատեսված է կոշտ դիֆերենցիալ հավասարումներ լուծելու համար: Այս ֆունկցիան հարմար է օգտագործել այն դեպքերում, երբ **ode45** ֆունկցիան չի կարողանում վերադարձնել ճիշտ արդյունք:
- **ode23s** ֆունկցիան ևս նախատեսված է կոշտ դիֆերենցիալ հավասարումներ լուծելու համար և հիմնված է Ռոզենբրոկի մի քայլանի եղանակի վրա: Երբեմն այս ֆունկցիան թույլ է տալիս լուծել այնպիսի դիֆերենցիալ հավասարումներ, որոնք **ode15s** ֆունկցիան լուծել չի կարողանում:

- **ode23t** ֆունկցիան հարմար է կիրառել ոչ շատ կոշտ դիֆերենցիալ հավասարումներ լուծելու համար:
- **ode23tb** ֆունկցիան նախատեսված է կոշտ դիֆերենցիալ հավասարումների լուծման համար և երբեմն ավելի արդյունավետ է, քան **ode15s**–ը:

Այսպիսով, ոչ կոշտ դիֆերենցիալ հավասարումները լուծելու համար պետք է կիրառել **ode45**, **ode23** կամ **ode113** ֆունկցիաները, ընդ որում նախ և առաջ ցանկալի է փորձել դրանք լուծել **ode45**–ով: Եթե հատուկ չի նշվում, այս ֆունկցիաները դիֆերենցիալ հավասարման լուծումները որոնում են 10^{-3} ճշտությամբ: Ճշտությունը կարելի է ավելի մեծացնել, ինչի մասին կխոսենք հետագայում: Եթե այս ֆունկցիաներից ոչ մեկը չի վերադարձնում սպասվող արդյունքը, ապա հնարավոր է, որ գործ ունենք կոշտ դիֆերենցիալ հավասարման հետ, այսինքն՝ այնպիսի հավասարումների, որոնց լուծումները փոխվում են շատ արագ և շատ դանդաղ և պահանջում են փոքր ժամանակային քայլեր խնդիրը լուծելու համար: Այս դեպքերում պետք է օգտվել **ode15s**, **ode23s**, **ode23t** կամ **ode23tb** ֆունկցիաներից, ընդ որում առաջին հերթին պետք է փորձել օգտվել **ode15s** ֆունկցիայից:

Այժմ տեսնենք, թե ինչպես են կիրառվում այս ֆունկցիաները: Այստեղ չենք սահմանափակվի առաջին կարգի սովորական դիֆերենցիալ հավասարումների լուծմամբ, այլ կդիտարկենք կամայական n–րդ կարգի դիֆերենցիալ հավասարում, որն ընդհանուր դեպքում կներկայացնենք

$$y^{(n)} = f(t, y, y', ..., y^{(n-1)})$$

տեսքով և $t = t_0$ պահին

$$y(t_0) = y_0\,,\ y'(t_0) = y_1\,,\ ...,\ y^{(n-1)}(t_0) = y_{n-1}$$

սկզբնական պայմաններով: Այս դիֆերենցիալ հավասարումները լուծելու համար **MatLab**–ում անհրաժեշտ է կատարել հետևյալ քայլերը.

1. Դիֆերենցիալ հավասարումը բերել առաջին կարգի դիֆերենցիալ հավասարումների համակարգի,
2. Ստացված համակարգի համար սահմանել առանձին M–ֆունկցիա,
3. Օգտագործել վերը նշված սոլվերներից որևէ մեկը,
4. Գրաֆիկորեն պատկերել հավասարման լուծումը:

Օրինակ.

Լուծենք $\frac{d^2y}{dt^2}+5\frac{dy}{dt}-4y=\sin 10t$ երկրորդ կարգի սովորական դիֆերենցիալ հավասարումը $y(0)=0$ և $y'(0)=0$ սկզբնական պայմաններով:

Ինչպես նշեցինք, նախ անհրաժեշտ է նշված դիֆերենցիալ հավասարումը բերել առաջին կարգի դիֆերենցիալ հավասարումների համակարգի: Այդ նպատակով սահմանենք y_1 և y_2 լրացուցիչ ֆունկցիաներ հետևյալ տեսքով.

$$y_1=y,\ y_2=\frac{dy_1}{dt}:$$

Այժմ դիֆերենցիալ հավասարումը և սկզբնական պայմանները կարելի է գրել հետևյալ համակարգերի տեսքով.

$$\begin{cases}\frac{dy_1}{dt}=y_2\\ \frac{dy_2}{dt}=-5y_2+4y_1+\sin 10t\end{cases} \quad \text{և} \quad \begin{cases}y_1(0)=0\\ y_2(0)=0\end{cases}:$$

Որպես երկրորդ քայլ, անհրաժեշտ է սահմանել M–ֆունկցիա, որը պետք է ունենա երկու մուտքային արգումենտ: Առաջինը t վեկտորն է, ըստ որի կատարվում է ածանցումը և y վեկտորը, որի երկարությունը հավասար է անհայտ ֆունկցիաների թվին, տվյալ դեպքում՝ 2–ի: Որպես ելքային արգումենտ վերադարձվում է համակարգի աջ մասերից բաղկացած վեկտորը:

```
function F = diffeq(t, y)
    F = [y(2); -5*y(2)+4*y(1)+sin(10*t)];
end
```

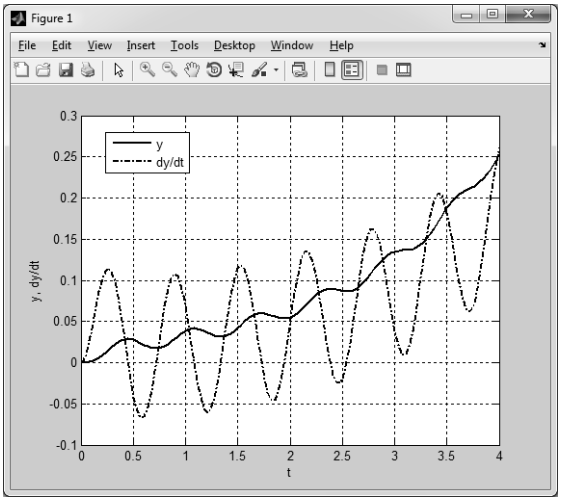
Այժմ լուծենք հավասարումը՝ օգտվելով **ode45** ֆունկցիայից: Պարզագույն դեպքում այս և մնացած սոլվերների արգումենտները երեքն են՝ սահմանված M–ֆունկցիայի անունը, որը պարունակում է դիֆերենցիալ հավասարումների համակարգը, t արգումենտի տիրույթը վեկտորի տեսքով, որում ցանկանում ենք լուծել հավասարումը և սկզբնական արժեքների վեկտորը: Ելքում ստացվում են t անկախ արգումենտի վեկտորը և լուծումը՝ մատրիցի տեսքով, որի առաջին սյունը պարունակում է y ֆունկցիայի արժեքները, իսկ մյուս սյուները՝ այդ ֆունկցիայի ածանցյալները: Տվյալ խնդրի դեպքում մատրիցը կունենա երկու սյուն, որոնցից առաջինը կպարունակի y ֆունկցիայի, իսկ երկրորդը՝ y' ածանցյալի արժեքները:

Հավասարումը լուծելուց հետո հարմար է այն պատկերել գրաֆիկորեն: Ստորև բերված ծրագիրը նախ սահմանում է սկզբնական արժեքների $y0$ վեկտորը, **ode45** ֆունկցիայի օգնությամբ լուծում է դիֆերենցիալ հավասարումը չորրորդ կարգի Ռունգե–Կուտտայի եղանակով, այնուհետև պատկերում y և y' ֆունկցիաների գրաֆիկ-

ները (այդ ֆունկցիաները համապատասխանում են լուծման մատրիցի $y(:, 1)$ առաջին և $y(:, 2)$ երկրորդ սյուների արժեքներին)։

```
y0 = [0; 0];
[t, y] = ode45(@diffeq, [0 4], y0);
plot(t, y(:,1), 'k', 'linewidth', 2)

grid on
hold on

plot(t, y(:,2), 'k-.', 'linewidth', 2)

xlabel('t')
ylabel('y, dy/dt')
legend('y', 'dy/dt')
```

Լուծման արդյունքը պատկերված է նկ. 61–ում, որտեղ հոծ գծով պատկերված է y ֆունկցիայի, իսկ կետ–գծերով՝ y' ֆունկցիայի գրաֆիկները։



Նկ. 61

Օրինակ.

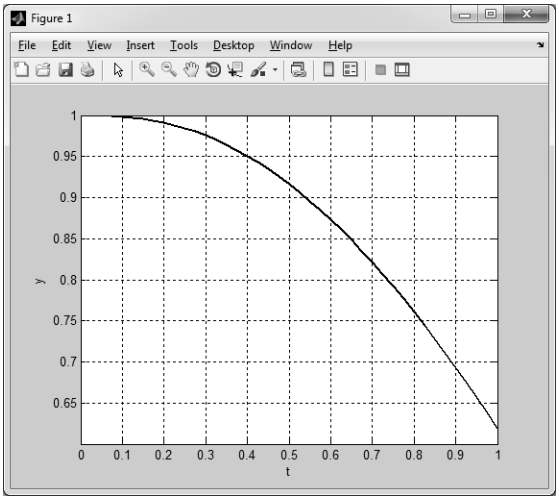
Այս օրինակում կլուծենք $\frac{dy}{dt} = -A(y - \cos t)$ կոշտ դիֆերենցիալ հավասարումը $y(0) = 1$ սկզբնական պայմանով։ Ցույց տանք, որ կախված A պարամետրի արժեքներից՝ **ode45** ֆունկցիան կարող է վերադարձնել սխալ արդյունք։ Իրոք, նախ ենթադրենք, որ $A = 10$: Այս դեպքում **ode45** ֆունկցիան հեշտությամբ լուծում է նշված հավասարումը։

Սահմանված M–ֆունկցիան կունենա հետևյալ տեսքը.

```
function F = stiff_eq(t, y)
    F = -10*(y-cos(t));
end
```

Ֆունկցիան սահմանելուց հետո գրենք ծրագիր, որը **ode45** ֆունկցիայի օգնությամբ կլուծի այս հավասարումը և կպատկերի լուծման գրաֆիկը (նկ. 62)։

```
y0 = [1];
[t, y] = ode45(@stiff_eq, [0 1], y0);
plot(t, y, 'k', 'linewidth', 2)
grid on
xlabel('t')
ylabel('y')
```



Նկ. 62

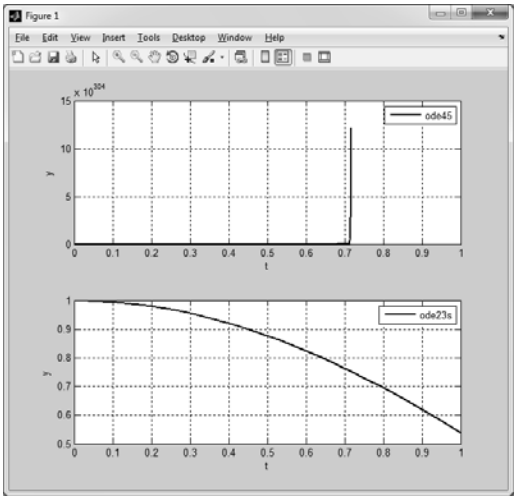
Այժմ լուծենք նույն հավասարումը $A=-1000$ դեպքում **ode45** և **ode23s** ֆունկցիաների օգնությամբ: Նախ գրենք M–ֆունկցիան այս դեպքի համար.

```
function F = stiff_eq(t, y)
    F = 1000*(y-cos(t));
end
```

Գրենք ծրագիր, որը երկու եղանակով կլուծի նշված հավասարումը.

```
y0 = [1];
[t, y] = ode45(@stiff_eq, [0 1], y0);
subplot(2,1,1)
plot(t, y, 'k', 'linewidth', 2)
grid on
xlabel('t')
ylabel('y')
[t, y] = ode23s(@stiff_eq, [0 1], y0);
subplot(2,1,2)
plot(t, y, 'k', 'linewidth', 2)
grid on
xlabel('t')
ylabel('y')
```

Ծրագրի իրականացման արդյունքը պատկերված է նկ. 63–ում: Ինչպես տեսնում ենք, **ode45** ֆունկցիայից օգտվելիս լուծումն անճանաչելիորեն փոխվում է, մինչդեռ **ode23s** ֆունկցիան կարողանում է լուծել կոշտ դիֆերենցիալ հավասարումը:



Նկ. 63

7.8. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Լուծեք հետևյալ հավասարումները.

- $xe^{-x} = 0.7$,
- $e^{0.5x} - \sqrt{x} = 3$,
- $3 + 4\sin x = 0.6x^3$,
- $\cos x = 3x^3$:

2. Որոշեք հետևյալ հավասարումների առաջին երեք դրական արմատները.

- $2\sin x - \sqrt{x} = -2.5$,
- $4\cos 2x - e^{0.5x} + 5 = 0$,
- $x^3 - 8x^2 + 17x + \sqrt{x} = 10$:

3. Որոշեք $x^2 - 5x\sin 3x + 3 = 0$ հավասարման դրական արմատները:

4. Որոշեք հետևյալ ֆունկցիաների փոքրագույն և մեծագույն արժեքները.

- $y(x) = x^3 - 11x^2 + 39x - 34$,
- $y(x) = (x-3)(x+2)$,
- $y(x) = x^3 + 3x^2 - 2x + 1$,
- $y(x) = \dfrac{x-3}{((x-3)^4 + 2)^{1.8}}$:

5. Տրված են $p_1 = x^3 + 7x^2 + 9x - 3$, $p_2 = 8x^4 + 2x^3 - 5x^2 + 3.2x - 10$ և $p_3 = x^5 + 3x^4 - 6x^3 + 12x^2 - x + 2$ բազմանդամները: Օգտագործելով **polyval** ֆունկցիան՝ որոշեք $p_1(3)$, $p_2(8)$ և $p_3(6)$ արժեքները:

6. Տրված է $p = x^3 + 3x^2 - x - 1$ բազմանդամը: Օգտագործելով **polyval** ֆունկցիան՝ որոշեք այս բազմանդամի արժեքները $x = 1, 2, ..., 10$ կետերում:

7. Տրված է $p = x^4 + x^2 - 1$ բազմանդամը: Օգտագործելով **polyval** ֆունկցիան՝ կառուցեք այս բազմանդամի գրաֆիկը $x \in [-2; 2]$ միջակայքում՝ օգտագործելով 50 կետ:

8. Տրված է $p = -2x^4 + 6x^3 + 3x^2 - 4x + 28$ բազմանդամը: Օգտագործելով **polyval** ֆունկցիան՝ կառուցեք այս բազմանդամի գրաֆիկը $x \in [0; 50]$ միջակայքում:

9. Տրված է $p = -0.001x^4 + 0.04x^3 - 0.68x^2 + 3.1x - 1$ բազմանդամը: Օգտագործելով **polyval** ֆունկցիան՝ կառուցեք այս բազմանդամի գրաֆիկը $x \in [1; 15]$ միջակայքում:

10. Օգտագործելով **roots** ֆունկցիան՝ որոշեք հետևյալ բազմանդամների արմատները.

- $p_1(x) = x^7 + 8x^6 + 5x^5 + 4x^4 + 2x^2 + x + 1$,
- $p_2(x) = x^5 - 13x^4 + 10x^3 + 12x^2 + 8x - 15$,
- $p_3(x) = x^3 - 3x^2 + 2x$,
- $p_4(x) = x^4 + 7x^3 + 12x^2 - 25x + 8$:

11. Օգտագործելով **conv** ֆունկցիան՝ որոշեք հետևյալ բազմանդամների արմատները և կառուցեք դրանց գրաֆիկները.

- $(3x^2 + 4)(x^4 + 2x^3 - 5x^2 + 12x - 8)$,
- $(x^3 - 1)(4.5x^2 + 8x + 6)$,
- $(2x^3 - 1)(4x + 7)(9x - 2)$,
- $(x + 45)(x - 8)x(x^2 + 2x + 1)(x - 14)$:

12. Օգտագործելով **deconv** ֆունկցիան՝ $0.7x^5 - 3x^4 + 2x^3 - 32x^2 + 8x - 1$ բազմանդամը բաժանեք $x^3 + 2x^2 - 9x + 1$ բազմանդամի վրա:

13. Օգտագործելով **deconv** ֆունկցիան՝ $x^6 - 3x^4 + 8x^3 - 12x + 9$ բազմանդամը բաժանեք $4x^2 + 8$ բազմանդամի վրա:

14. Երեք հաջորդական թվերի արտադրյալը հավասար է 7.412–ի: Օգտագործելով բազմանդամների համար նախատեսված ֆունկցիաները՝ որոշեք այդ թվերը:

15. Հինգ հաջորդական կենտ թվերի արտադրյալը հավասար է 52632–ի: Օգտագործելով բազմանդամների համար նախատեսված ֆունկցիաները՝ որոշեք այդ թվերը:

16. Սահմանեք M–ֆունկցիա, որը թույլ կտա գումարել կամ հանել կամայական երկու բազմանդամ: Ֆունկցիան անվանեք PAddSub: Այն պետք է ունենա երեք մուտքային արգումենտ (*p*, *q*, *op*) և մեկ ելքային արգումենտ (*f*): *p* և *q* արգումենտները համապատասխանաբար առաջին և երկրորդ բազմանդամների գործակիցներից կազմված վեկտորներն են, իսկ *op*–ը տողային փոփոխական է, որը կարող է ընդունել *add* արժեք, եթե ցանկանում ենք գումարել բազմանդամները և *sub*՝ եթե ցանկանում ենք դրանք իրարից հանել: *f*–ը գումարման կամ հանման գործողության արդյունքում ստացված բազմանդամն է: Եթե *p* և *q* բազմանդամների կար-

գերը չեն համապատասխանում, ապա ֆունկցիան պետք է ցածր կարգով բազմանդամը լրացնի զրոներով: Օգտագործելով սահմանված PAddSub ֆունկցիան՝ գումարեք և հանեք $3x^4-2x^2+7x+12$ և x^3+4x^2-3x-2 բազմանդամները:

17. Սահմանեք M–ֆունկցիա, որը թույլ կտա բազմապատկել կամայական երկու բազմանդամ: Ֆունկցիան անվանեք PMult: Այն պետք է ունենա երկու մուտքային արգումենտ (p, q) և մեկ ելքային արգումենտ (f): p և q արգումենտները համապատասխանաբար առաջին և երկրորդ բազմանդամների գործակիցներից կազմված վեկտորներն են, իսկ f–ը բազմապատկման արդյունքում ստացված բազմանդամն է: Օգտագործելով սահմանված PMult ֆունկցիան՝ բազմապատկեք $2x^3-82x^2-10x+7$ և $x^5+1.2x^2-6x-1$ բազմանդամները: Արդյունքը համեմատեք **conv** ներդրված ֆունկցիայի օգնությամբ ստացված արդյունքի հետ:

18. Սահմանեք M–ֆունկցիա, որը թույլ կտա որոշել ax^2+bx+c քառակուսային ֆունկցիայի մեծագույն և փոքրագույն արժեքները: Օգտագործելով սահմանված ֆունկցիան՝ որոշեք $3x^2-4x+8$ բազմանդամի մեծագույն և փոքրագույն արժեքները:

19. Գլանաձև ալյումինե տարայի արտաքին տրամագիծը հավասար է 70սմ, իսկ բարձրությունը՝ 115սմ: Տարայի պատի հաստությունը հավասար է x–ի, իսկ հիմքերի հաստությունը՝ դրանից 20%–ով ավելի: Ալյումինի խտությունը 0.0026 կգ/սմ^3 է: Որոշեք x–ը, եթե հայտնի է, որ տարայի զանգվածը հավասար է 60կգ:

20. Գլանաձև ալյումինե տարայի ստորին հիմքը հարթ է, իսկ վերին հիմքը՝ կիսագունդ: Տարայի արտաքին տրամագիծը հավասար է 20սմ, գլանաձև հատվածի բարձրությունը՝ 45սմ: Պատերի և կիսագնդի հատվածի հաստությունը հավասար է x–ի, իսկ ստորին հիմքի հաստությունը՝ $1.5x$–ի: Ալյումինի խտությունը 0.0026 կգ/սմ^3 է: Որոշեք x–ը, եթե հայտնի է, որ տարայի զանգվածը հավասար է 28կգ:

21. Օգտագործելով **polyfit** ֆունկցիան՝ մոտարկեք հետևյալ աղյուսակում տրված դիսկրետ տվյալները 1–ին, 3–րդ, 4–րդ, 6–րդ և 8–րդ կարգի բազմանդամային ֆունկցիաներով: Յուրաքանչյուր դեպքի համար աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորերը՝ հոծ գծերով:

x	0	1	2	3	4	5	6	7
y	1	–0.2	0.05	0.01	0.01	0.01	0.005	0.005

22. Օգտագործելով **polyfit** ֆունկցիան՝ մոտարկեք հետևյալ աղյուսակում տրված դիսկրետ տվյալները 1–ին, 2–րդ, 4–րդ, 6–րդ և 7–րդ կարգի բազմանդամային ֆունկցիաներով: Յուրաքանչյուր դեպքի համար աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորերը՝ հոծ գծերով:

x	–5	–3.4	–2	–0.8	0	1.2	2.5	4	5	7	8.5
y	4.4	4.5	4	3.6	3.9	3.8	3.5	2.5	1.2	0.5	–0.2

23. Օգտագործելով **polyfit** ֆունկցիան՝ հետևյալ աղյուսակում բերված դիսկրետ տվյալները մոտարկեք $y = \frac{ax}{b+x}$ տեսքի կոտորակային ֆունկցիայով: Աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորերը՝ հոծ գծերով:

x	1	3	4	7	8	10
y	2.1	4.6	5.4	6.2	6.5	6.7

24. Ժամանակի տարբեր պահերին բակտերիաների թիվը տրված է հետևյալ աղյուսակում.

t (րոպե)	10	20	30	40	50
N	17000	220000	345000	510000	770000

Որոշեք $N = N_0 e^{at}$ տեսքի այն էքսպոնենցիալ ֆունկցիան, որը լավագույն կերպով կմոտարկի այս տվյալները: Ստացված ֆունկցիայի օգնությամբ որոշեք բակտերիաների թիվը ժամանակի $t = 65$ պահին: Աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորը՝ հոծ գծով:

25. Սահմանեք M-ֆունկցիա, որը թույլ կտա դիսկրետ տվյալները մոտարկել $y = ax^m$ տեսքի աստիճանային ֆունկցիայով: Ֆունկցիան պետք է ունենա [a, m] = pow_fit(x,y) տեսքը, որտեղ x–ը և y–ը դիսկրետ տվյալների կոորդինատներից կազմված վեկտորներն են, իսկ a և m ելքային արգումենտները՝ մոտարկման արդյունքում ստացված հաստատուն գործակիցները: Օգտագործելով սահմանված ֆունկցիան՝ մոտարկեք հետևյալ աղյուսակում բերված դիսկրետ տվյալները: Աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորը՝ հոծ գծով:

x	0.5	2.4	3.2	4.8	6.4	7.8
y	0.7	9.2	36.4	67.7	147	182

26. Աշխարհի բնակչությունը 1750 թվից մինչև 2009 թիվը ներկայացված է հետևյալ աղյուսակում:

Տարի	1750	1800	1850	1900	1950	1990	2000	2009
Բնակչ. ($\times 10^6$)	791	980	1260	1650	2520	5270	6060	6800

Մոտարկեք այս տվյալները

- 3–րդ կարգի բազմանդամային ֆունկցիայով,
- էքսպոնենցիալ ֆունկցիայով,
- գծային ինտերպոլացման օգնությամբ,
- խորանարդային սպլայն ինտերպոլացման օգնությամբ:

Յուրաքանչյուր դեպքի համար աղյուսակի տվյալները պատկերեք կլոր մարկերներով, իսկ մոտարկված կորերը՝ հոծ գծով: Յուրաքանչյուր դեպքի համար որոշեք աշխարհի բնակչության թիվը 1980 թվականին և համեմատեք այդ թվականին աշխարհի բնակչության իրական 4453.8 միլիոն թվի հետ:

27. Օգտագործելով հետևյալ աղյուսակում բերված դիսկրետ տվյալները՝ գծային ինտերպոլացման օգնությամբ որոշեք ֆունկցիայի արժեքները $x=5.6$ և $x=10.3$ կետերում:

x	1	3	5	7	9	11
y	–3	4	5	–8	–3	0

28. Օգտագործելով հետևյալ աղյուսակում բերված դիսկրետ տվյալները՝ խորանարդային սպլայն ինտերպոլացման օգնությամբ որոշեք ֆունկցիայի արժեքները $x=-\pi/4$ և $x=\pi/3$ կետերում:

x	$-\pi$	$-\pi/2$	0	$\pi/2$	π
y	–3	4	5	–8	–3

29. Որոշեք հետևյալ որոշյալ ինտեգրալների մոտավոր արժեքները.

- $\int_0^{\pi/2} \sin^6 x\cos^7 x dx$,
- $\int_1^2 \frac{\cos 2x}{x} dx$,
- $\int_1^2 \sin x^2 e^{\sqrt{x}} dx$,
- $\int_1^2 \frac{e^{2x}}{x} dx$,
- $\int_0^7 (xe^{-x^{0.3}} + 0.1) dx$,
- $\int_{-1}^1 e^{-x^2} dx$,
- $\int_1^6 \frac{2x^2}{\sqrt{1+x}} dx$,
- $\int_0^{20} \cos^2(0.5x)\sin^4(0.5x) dx$:

30. Որոշեք հետևյալ փոփոխական վերին սահմանով ինտեգրալների մոտավոր արժեքները.

- $\int_0^x \frac{\cos x}{\sqrt{1+\sin x}} dx$, որտեղ $0 < x < \frac{\pi}{2}$,
- $\int_0^x x^2 \sin x^3 dx$, որտեղ $0 < x < \frac{\pi}{2}$,
- $\int_1^x \frac{(1+x^{2/3})}{x^{1/3}} dx$, որտեղ $1 < x < 8$,
- $\int_0^x \sqrt{\operatorname{tg} x}\sec^2 x dx$, որտեղ $0 < x < \frac{\pi}{4}$:

31. Օգտագործելով դիֆերենցիալ հավասարումների տարբեր թվային եղանակները՝ լուծեք հետևյալ հավասարումները.

- $\frac{dy}{dt} = (1-t)y$, $t \in [0;5]$ միջակայքում, եթե $y(0)=1$,
- $\frac{dy}{dt} = -y\sqrt{t}$, $t \in [0;1]$ միջակայքում, եթե $y(0)=1$,

- $\frac{dy}{dt} = \frac{t^3 - 2y}{t}$, $t \in [1;3]$ միջակայքում, եթե $y(1) = 4.2$,
- $\frac{dy}{dt} = \sqrt{ty} - 0.5ye^{-0.1t}$, $t \in [0;4]$ միջակայքում, եթե $y(0) = 6.5$,
- $\frac{dy}{dt} = 80e^{-1.6t}\cos 4t - 0.4y$, $t \in [0;4]$ միջակայքում, եթե $y(0) = 0$,
- $\frac{d^2y}{dt^2} + 3y = t$, $t \in [0;1]$ միջակայքում, եթե $y(0) = y'(0) = 0$,
- $\frac{d^2y}{dt^2} = t\cos t$, եթե $y(1) = 0$, $y'(0) = 0$,
- $\frac{d^2y}{dt^2} + 10\frac{dy}{dt} + 5y = 11$, եթե $y(0) = 1$, $y'(0) = -1$,
- $\frac{d^2y}{dt^2} + \frac{dy}{dt} - 3y = 5$, եթե $y(0) = 1$, $y'(0) = 0$,
- $\frac{d^3y}{dt^3} - 2\frac{d^2y}{dt^2} - \frac{dy}{dt} + 2y = 0$, եթե $y(0) = y'(0) = 0$, $y''(0) = 1$:

8. ՍԻՄՎՈԼԱՅԻՆ (ԱՆԱԼԻՏԻԿ) ՀԱՇՎԱՐԿՆԵՐ

8.1. ԹՎԱՅԻՆ ԵՎ ՍԻՄՎՈԼԱՅԻՆ (ԱՆԱԼԻՏԻԿ) ՀԱՇՎԱՐԿՆԵՐ

Մինչ այժմ դիտարկված բոլոր խնդիրները լուծվում էին թվայնորեն, երբ համակարգիչը կարծես վերածվում է հզոր հաշվիչի, որն արագ իրականացնում է թվաբանական և տրամաբանական գործողություններ թվերի և թվերի հաջորդականությունների հետ: Այս գործողությունների արդյունքը ևս թիվ է կամ թվերի խումբ՝ գրված հաջորդականությունների, աղյուսակների (մատրիցների) կամ գրաֆիկների կետերի տեսքով: Սակայն խիստ իմաստով այսպիսի հաշվարկների արդյունքները շատ հազվադեպ են մաթեմատիկորեն ճշգրիտ: Իրոք, գործողությունների մեծ մասը մենք կատարում ենք իրական թվերով, որոնք համակարգիչը միշտ կլորացնում է, որպեսզի կարողանա պահել հիշողության մեջ: Բացի այդ, թվային եղանակների իրականացումը ևս (օրինակ՝ ոչ գծային կամ դիֆերենցիալ հավասարումների և դրանց համակարգերի լուծումը, ինտեգրալների հաշվարկը և այլն) հիմնված է մոտավոր ալգորիթմների վրա: Որքան էլ մոտավոր, այս եղանակները հաճախ տալիս են ընդունելի արդյունք, սակայն քիչ չեն դեպքերը, երբ հաշվարկի սխալների կուտակման պատճառով կորում է հաշվարկի կայունությունը, և լուծումները տարամիտում են:

Բացի թվային հաշվարկներից, **MatLab**–ը հնարավորություն է տալիս կատարել, այսպես կոչված, սիմվոլային կամ անալիտիկ հաշվարկներ: Այս դեպքում հաշվարկները կատարվում են ոչ թե թվային մոտավոր եղանակներով, այլ սիմվոլային (բանաձևային) արտահայտությունների օգնությամբ, որոնց արդյունքը ևս գրվում է բանաձևի տեսքով:

Թվային և սիմվոլային հաշվարկները համեմատելու համար, որպես օրինակ, հաշվենք $\sin^2 x + \cos^2 x = 1$ արտահայտության արժեքը x=1 կետում: Եթե հաշվարկը կատարվում է թվային եղանակով, ապա **MatLab**–ը հաշվում է sin(1) և cos(1) ֆունկցիաների թվային արժեքները, որոնք այդ ընթացքում կլորացվում են, և դրանց քառակուսիները գումարում իրար: Չնայած հայտնի է, որ կամայական արգումենտի համար $\sin^2 x + \cos^2 x = 1$, կլորացման հետևանքով ստացվում ոչ թե 1, այլ, օրինակ, 0.99...9 կամ 1.00...01: Սիմվոլային եղանակով հաշվելու դեպքում համակարգիչը ոչ թե հաշվում է sin(1)–ի և cos(1)–ի արժեքները, այլ խնդիրը ներկայացնում է

բանաձևային տեսքով՝ (sin(1)^2+cos(1)^2), և իր մեջ ներդրված բանաձևերի օգնությամբ վերադարձնում է այս արտահայտության ճշգրիտ 1 արժեքը:

Ավելին, եթե $\sin^2 x + \cos^2 x$ արտահայտության արժեքը ցանկանանք հաշվել ոչ թե արգումենտի կոնկրետ արժեքի, այլ կամայական x–ի համար, ապա թվային եղանակով հաշվող ցանկացած ծրագրավորման լեզու, այդ թվում և **MatLab**–ը, կվերադարձնի զգուշացում, որ x փոփոխականը սահմանված չի, այսինքն՝ x–ին կոնկրետ արժեք վերագրված չէ: Իսկ եթե նշենք, որ օգտվելու ենք **MatLab**–ի ոչ թե թվային, այլ սիմվոլային փաթեթից, ապա կամայական x փոփոխականի դեպքում կստանանք 1 արդյունքը:

Սա պարզ օրինակ էր, և այն լուծելու համար **MatLab**–ի նման լայն ծրագրային միջավայրին դիմելու կարիք չկար: Սակայն գոյություն ունեն բազմաթիվ խնդիրներ, որոնց արդյունքները ևս կարելի է ստանալ բանաձևերի տեսքով (այսինքն՝ ունեն անալիտիկ լուծում), բայց դրանց տեսքը հայտնի չէ: Այս դեպքերում է, որ օգնության է գալիս **MatLab**–ի սիմվոլային փաթեթը, որն իր մեջ ներդրված բանաձևերի հսկայական բազայի օգնությամբ լուծում է այս խնդիրները և արդյունք վերադարձնում: Այս բազան պարբերաբար թարմացվում է, **MatLab**–ի նոր տարբերակներում ավելացվում են նոր բանաձևեր, որոնք թույլ են տալիս լուծել ավելի մեծ թվով խնդիրներ:

Իհարկե, **MatLab**–ի սիմվոլային փաթեթի պաշարներն անսահմանափակ չեն, և ոչ միշտ է հնարավոր ստանալ անալիտիկ արդյունքներ: Եթե հանդիպում են խնդիրներ, որոնք հնարավոր չէ լուծել անալիտիկորեն, ապա **MatLab**–ը վերադարձնում է զգուշացում, որ խնդիրը չունի անալիտիկ լուծում: Այս դեպքում արդեն միակ միջոցը մնում է դիմել թվային եղանակներին և գտնել խնդրի գոնե մոտավոր լուծում:

Այսպիսով, **MatLab**–ի սիմվոլային հաշվարկների փաթեթն իր մեջ ներառում է անալիտիկ հաշվարկների հզոր բազա, որի օգնությամբ կարելի է

- կատարել սիմվոլային ինտեգրում, ածանցում, սահմանների որոշում,
- կատարել տարբեր ձևափոխություններ (Ֆուրիեի, Լապլասի, z– և այլն),
- լուծել գծային հանրահաշվի տարբեր խնդիրներ,
- լուծել հանրահաշվական, դիֆերենցիալ հավասարումներ և դրանց համակարգեր,
- արտահայտություններ պարզեցնել և դրանք ձևափոխել:

8.2. ՍԻՄՎՈԼԱՅԻՆ ԹՎԵՐ

MatLab–ի սիմվոլային փաթեթը հնարավորություն է տալիս սովորական իրական թիվը դարձնել սիմվոլային օբյեկտ, այսինքն՝ թվերի հետ աշխատել ինչպես սիմվոլների հետ: Թիվը սիմվոլային դարձնելու համար օգտվում են **sym** հրամանից:

Օրինակ.

```
>> a = sym('3')
```

Եթե սիմվոլային թվի թվանշանների քանակը չի գերազանցում 15–ը, ապա ապոստրոֆները կարելի է չդնել:

Օրինակ.

```
>> a = sym(3);
```

Որպեսզի պատկերացնենք սովորական թվի և սիմվոլային թվի տարբերությունը, հաշվենք, օրինակ, $\sqrt{3}$ արտահայտության արժեքը իրական 3 թվի և վերևում հայտարարված սիմվոլային a=3 թվի համար.

```
>> b = sqrt(3)
   b = 1.7321
>> c = sqrt(a)
   c = 3^(1/2)
```

Ինչպես տեսնում ենք, իրական թվի հետ աշխատելիս **MatLab**–ը հաշվում է 3 թվի արմատը և վերադարձնում մոտավոր, կլորացրած արդյունք: Ի տարբերություն դրա, սիմվոլային 3 թվի արմատը հաշվելիս որևէ մոտավորություն չի արվում, և արդյունքը ևս գրվում է սիմվոլային տեսքով (3^(1/2)):

MatLab–ը հնարավորություն է տալիս կատարել նաև հակառակ գործողությունը՝ սիմվոլային թիվը դարձնել իրական և վերջինիս հետ կատարել սովորական թվային գործողություններ: Դրա համար նախատեսված է **double** հրամանը, որի արգումենտում գրվում է սիմվոլային թիվը:

Օրինակ.

```
>> double(c)
   ans = 1.7321
```

Կոտորակային սիմվոլային թիվ կարելի է սահմանել, օրինակ, հետևյալ կերպ.

```
>> x = sym(8);
>> y = sym(11);
>> x/y
   ans = 8/11
```

կամ

```
>> z = sym(8/11);
```

Սահմանված կոտորակային թվերի հետ կարելի է կատարել տարբեր մաթեմատիկական գործողություններ:

Օրինակ.

```
>> sym(2/3) + sym(7/12)
   ans = 5/4
```

Տեսնում ենք, սիմվոլային կոտորակային թվերը գումարելիս **MatLab**–ը դրանք բերում է ընդհանուր հայտարարի և գումարում այնպես, ինչպես սովորական ռացիոնալ թվերի հետ աշխատելիս:

Բայց եթե նույն գործողությունը կատարենք թվային եղանակով, ապա **MatLab**–ը նախ կհաշվի 2/3 հարաբերությունը, արդյունքը կկլորացնի, ապա նույնը կկատարի 7/12 հարաբերության հետ և կգումարի արդեն մոտավոր դարձած թվերը.

```
>> 2/3 + 7/12
   ans = 1.2500
```

8.3. Սիմվոլային փոփոխականներ և արտահայտություններ

MatLab–ը հնարավորություն է տալիս փոփոխականները ևս հայտարարել որպես սիմվոլային: Դրա համար գոյություն ունի երկու եղանակ:

Առաջին եղանակով սիմվոլային փոփոխական հայտարարելու համար օգտվում են **sym** ֆունկցիայից, որի արգումենտում ապոստրոֆների մեջ գրվում է անհրաժեշտ անունը:

Օրինակ.

```
>> x = sym('x');
>> y = sym('y');
```

Երկրորդ եղանակով սիմվոլային փոփոխական հայտարարելու համար օգտվում են **syms** հրամանից, որին հետևում են սիմվոլային փոփոխականի անունը կամ, մի քանի փոփոխական հայտարարելու դեպքում, դրանց անունները՝ բացատով իրարից անջատած:

Օրինակ.

```
>> syms x y
```

Սիմվոլային փոփոխականներ հայտարարելու այս երկու եղանակներից յուրաքանչյուրն ունի որոշակի առանձնահատկություններ.

Առաջին եղանակի դեպքում

- անհրաժեշտ է անպայման օգտագործել փակագծեր ու ապոստրոֆներ՝ x = sym(′x′), ըստ որում, ապոստրոֆները կարելի է բաց թողնել միայն այն դեպքում, երբ հայտարարում ենք ոչ թե սիմվոլային փոփոխական, այլ սիմվոլային թիվ, եթե վերջինիս նիշերի քանակը չի գերազանցում 15–ը,
- մեկ հրամանով կարելի է սահմանել ընդամենը մեկ սիմվոլային փոփոխական,
- հնարավոր է սահմանել սիմվոլային թվեր և արտահայտություններ:

Երկրորդ եղանակի դեպքում

- փակագծեր ու ապոստրոֆներ օգտագործելու կարիք չկա,
- մեկ հրամանով կարելի է հայտարարել միանգամից մի քանի փոփոխական:

Եթե հատուկ չի նշվում, սիմվոլային փոփոխականները հայտարարվում են որպես կոմպլեքս փոփոխական: Սակայն կարելի է դրանք հայտարարել որոշակի պայմաններով: **assume** ներդրված ֆունկցիան նախատեսված է սիմվոլային փոփոխականի նկատմամբ պայման դնելու համար, որը գրվում է այդ ֆունկցիայի արգումենտում:

Օրինակ.

```
>> syms x
>> assume(x>=2)
```

Այս օրինակում սահմանվում է *x* սիմվոլային փոփոխականը և ենթադրվում է, որ այն մեծ է կամ հավասար 2–ից:

assume ֆունկցիայի արգումենտում կարելի է տողի տեսքով գրել նաև 'integer', 'real', 'positive' և 'rational' արգումենտները, որոնք նշանակում են, որ տրված փոփոխականը համապատասխանաբար ամբողջ տիպի, իրական տիպի, դրական կամ կոտորակային է:

sym ֆունկցիայով սիմվոլային փոփոխականը հայտարարելիս վերջինիս նկատմամբ կարելի է դնել մեկից ավելի պայմաններ: Սկսած երկրորդ պայմանից՝ դրանք գրվում են ներդրված **assumeAlso** ֆունկցիայի արգումենտում:

Օրինակ.

```
>> syms x
>> assume(x>=2)
>> assumeAlso(x, 'integer')
```

Այս օրինակում հայտարարվեց *x* սիմվոլային փոփոխականը և վերջինիս վրա դրվեց երկու պայման: Ըստ առաջին պայմանի, այն մեծ է կամ հավասար 2–ից, ըստ երկրորդ պայմանի՝ այն նաև ամբողջ տիպի է:

syms հրամանով հայտարարելիս սիմվոլային փոփոխականի նկատմամբ կարելի է կիրառել միայն մեկ պայման:

Օրինակ.

```
>> syms x y z positive
>> syms b real
```

Այս օրինակում սահմանվեցին x, y, z դրական և b իրական սիմվոլային փոփոխականները:

Սիմվոլային փոփոխական հայտարարելուց հետո դրա հետ կարելի է կատարել տարբեր մաթեմատիկական գործողություններ:

Օրինակ.

```
>> x + x + y
   ans = 2*x + y
```

Արդեն հայտարարված սիմվոլային փոփոխականների օգնությամբ կարելի է հայտարարել այլ սիմվոլային արտահայտություններ կամ ֆունկցիաներ, ընդ որում՝ նախօրոք հայտարարված փոփոխականների օգնությամբ սահմանված նոր ֆունկցիաները պարտադիր չէ բացահայտ հայտարարել որպես սիմվոլային:

Օրինակ.

```
>> syms x a b
>> f = (cos(x)+a^3–log(b))/sqrt(a^2+b^2)
   f = (a^3 – log(b) + cos(x))/(a^2 + b^2)^(1/2)
```

Երբեմն գրված արտահայտությունները մեկ տողով գրելը այն դժվար ընթեռնելի է դարձնում, ինչպես վերևի օրինակի f արտահայտությունը: Սիմվոլային արտահայտությունը կարելի է բերել ավելի ընթեռնելի տեսքի՝ օգտագործելով **pretty** հրամանը, որի արգումենտում գրվում է սիմվոլային արտահայտության կամ ֆունկցիայի անունը:

Օրինակ.

```
>> pretty(f)
    3
   a  – log(b) + cos(x)
   --------------------
         2    2 1/2
       (a  + b )
```

8.4. ՍԻՄՎՈԼԱՅԻՆ ՎԵԿՏՈՐՆԵՐ ԵՎ ՄԱՏՐԻՑՆԵՐ

Բացի սիմվոլային սկալյար փոփոխականներից, **MatLab**–ում սահմանվում են նաև սիմվոլային վեկտորներ և մատրիցներ: Այս դեպքում վեկտորի կամ մատրիցի տարրերը պետք է լինեն սիմվոլային փոփոխականներ, սիմվոլային թվեր կամ սիմվոլային արտահայտություններ: Եթե վեկտորները կամ մատրիցները հայտարարվում են որպես սիմվոլային, ապա ցանկացած գործողություն դրանց հետ

կատարվում է անալիտիկորեն: Ընդ որում, սիմվոլային վեկտորներն ու մատրիցները հայտարարվում են նույն կերպ, ինչպես թվային վեկտորներն ու մատրիցները:

***Օրինակ** 1.*

Սահմանենք *a*, *b*, *c*, *d* սիմվոլային փոփոխականներից կազմված *x* տող–վեկտորը և *e*, *f*, *g*, *h* փոփոխականներից կազմված *y* սյուն–վեկտորը:

```
>> syms x a b c d e f g h
>> x = [a b c d]
   x = [ a, b, c, d]
>> y = [e; f; g; h]
   y =
      e
      f
      g
      h
```

Սիմվոլային վեկտորներ և մատրիցներ սահմանելուց հետո նրանց հետ կարելի է կատարել տարբեր մաթեմատիկական գործողություններ: Ընդ որում, գործողություններ կատարելու համար օգտագործվում են 2–րդ և 4–րդ գլխում սահմանված նույն ֆունկցիաները, որոնք սահմանված էին սովորական թվային վեկտորների և մատրիցների հետ աշխատելու համար:

***Օրինակ** 2.*

Նախորդ օրինակում սահմանված վեկտորի տարրերը դասավորենք աճման կարգով:

```
>> sort(x)
   ans = [ a, b, c, d]
```

***Օրինակ** 3.*

Այս օրինակը թույլ է տալիս սահմանել *a*, *b*, *c*, *d* սիմվոլային փոփոխականներից կազմված $A=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ և *e*, *f*, *g*, *h* փոփոխականներից կազմված $B=\begin{pmatrix} e & f \\ g & h \end{pmatrix}$ մատրիցները:

```
>> syms a b c d e f g h
>> A = [a b; c d]
   A =
      [ a, b]
      [ c, d]
>> B = [e f; g h]
   B =
      [ e, f]
      [ g, h]
```

***Օրինակ** 4.*

Սահմանված մատրիցների հետ ևս կարելի է կատարել տարբեր մաթեմատիկական գործողություններ: Այս օրինակում կհաշվենք նախորդ օրինակում սահմանված *A* և *B*

սիմվոլային մատրիցների S գումարը, P արտադրյալը, ինչպես նաև A մատրիցի M հակադարձ մատրիցն ու B մատրիցի k որոշիչը:

```
>> S = A+B
   S =
      [ a + e, b + f]
      [ c + g, d + h]

>> P =A*B
   P =
      [ a*e + b*g, a*f + b*h]
      [ c*e + d*g, c*f + d*h]

>> M = inv(A)
   M =
      [  d/(a*d – b*c), –b/(a*d – b*c)]
      [ –c/(a*d – b*c),  a/(a*d – b*c)]

>> pretty(M)
   +–                          –+
   |       d             b      |
   |   ---------,  – ---------  |
   |   a d – b c     a d – b c  |
   |                            |
   |        c           a       |
   |  – ---------,  ---------   |
   |    a d – b c   a d – b c   |
   +–                          –+

>> k = det(B)
   k = e*h – f*g
```

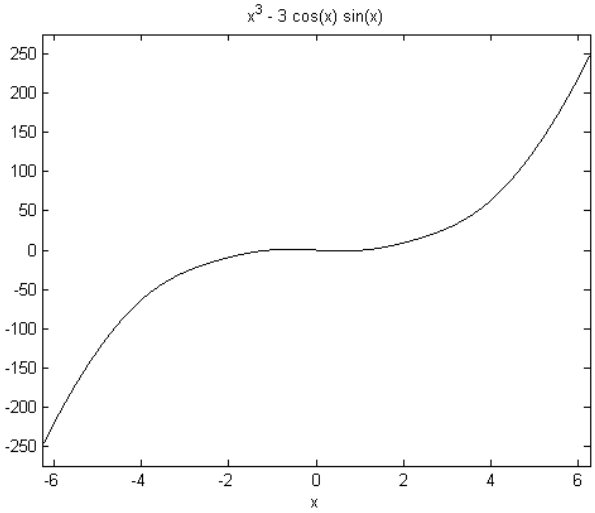
8.5. ԳՐԱՖԻԿՆԵՐ ԵՎ ՄԱԿԵՐԵՎՈՒՅԹՆԵՐ

Մեկ փոփոխականից կախված սիմվոլային ֆունկցիայի գրաֆիկը կառուցելու համար նախատեսված է **ezplot** ներդրված ֆունկցիան, որի արգումենտում գրվում է ֆունկցիայի անունը: Հատուկ չնշելու դեպքում ֆունկցիայի գրաֆիկը կառուցվում է $[-2\pi, 2\pi]$ միջակայքում: Եթե անհրաժեշտ է ֆունկցիայի գրաֆիկը կառուցել մեկ այլ միջակայքում, ապա այդ միջակայքը պետք է գրել որպես **ezplot** ֆունկցիայի երկրորդ արգումենտ՝ քառակուսի փակագծերի մեջ:

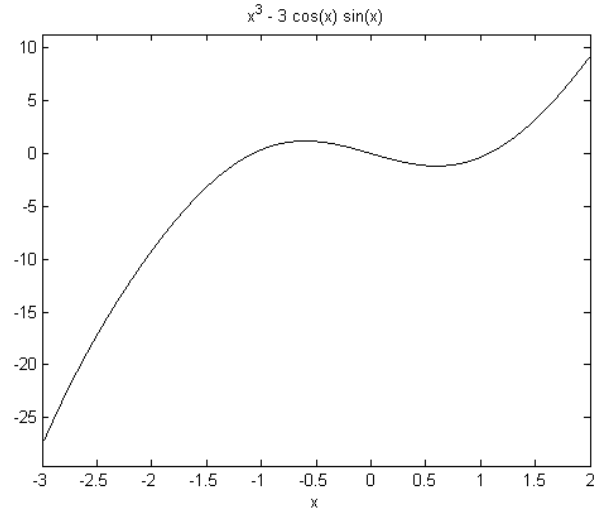
Օրինակ.

Այս օրինակում սահմանվում է x սիմվոլային փոփոխականը և $y = x^3 - 3\cos x \sin x$ սիմվոլային ֆունկցիան: Սկզբում կառուցվում է y ֆունկցիայի գրաֆիկը $[-2\pi, 2\pi]$ միջակայքում (նկ. 64), հետո՝ $[-3; 2]$ միջակայքում (նկ. 65):

```
>> syms x
>> y = x^3–3*cos(x)*sin(x);
>> ezplot(y)
>> figure
>> ezplot(y, [–3 2])
```



նկ. 64



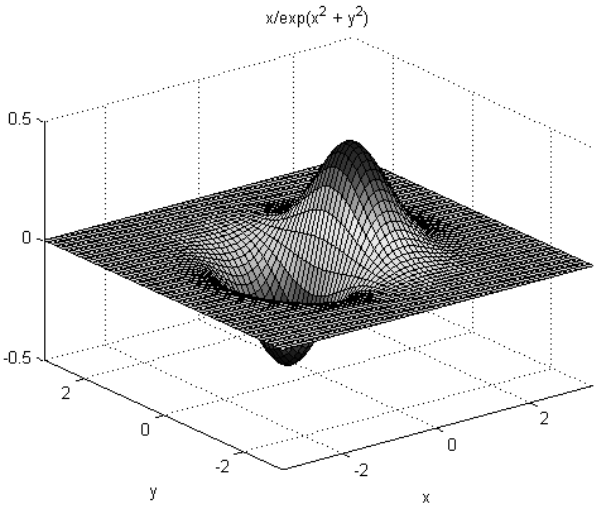
նկ. 65

Երկու փոփոխականից կախված սիմվոլային ֆունկցիայի մակերևույթը կառուցելու համար կարելի է օգտվել, օրինակ, **ezsurf** ֆունկցիայից: Ինչպես **ezplot** ֆունկցիայի դեպքում, այստեղ ևս պետք է որպես արգումենտ գրել ֆունկցիայի անունը: Եթե հատուկ չի նշվում, ապա անկախ փոփոխականների միջակայքն ընտրվում է $[-2\pi, 2\pi]$ տիրույթը: Եթե անհրաժեշտ է ֆունկցիայի գրաֆիկը կառուցել մեկ այլ միջակայքում, ապա այդ միջակայքը պետք է գրել որպես **ezsurf** ֆունկցիայի երկրորդ արգումենտ՝ քառակուսի փակագծերի մեջ:

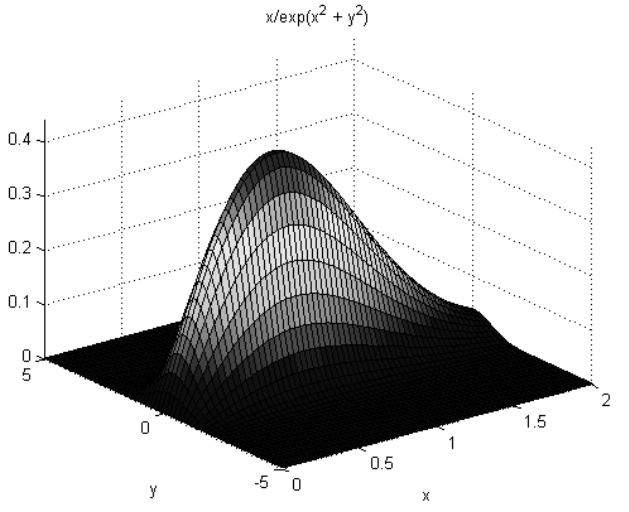
Օրինակ.

Այս օրինակում սահմանվում են x և y սիմվոլային փոփոխականները և $z = xe^{-x^2-y^2}$ սիմվոլային ֆունկցիան: Սկզբում կառուցվում է z ֆունկցիայի մակերևույթը $x \in [-2\pi, 2\pi]$, $y \in [-2\pi, 2\pi]$ միջակայքում (նկ. 66), հետո՝ $x \in [0, 2]$, $y \in [-5, 5]$ միջակայքում (նկ. 67):

```
>>syms x y;
>>z = x*exp(-x^2-y^2);
>>ezsurf(z)
>>ezsurf(z, [0 2 -5 5])
```



նկ. 66



նկ. 67

Բացի **ezplot** և **ezsurf** ֆունկցիաներից, երկչափ և եռաչափ գրաֆիկների կառուցման համար կարելի է օգտվել նաև **ezmesh**, **ezmeshc**, **ezsurfc**, **ezplot3**, **ezpolar** և մյուս ֆունկցիաներից, որոնք սահմանված են թվային եղանակով գրաֆիկներ կառուցելու համար:

8.6. Սիմվոլային արտահայտությունների պարզեցումը

Հաճախ հանրահաշվական, եռանկյունաչափական կամ լոգարիթմական արտահայտությունները անհրաժեշտ է լինում գրել պարզեցված կամ ձևափոխված տեսքով, խմբավորել ըստ որևէ փոփոխականի, վերլուծել արտադրիչների, կատարել փոփոխականների փոխարինումներ և այլն: **MatLab**–ի սիմվոլային հաշվարկման փաթեթը հնարավորություն է տալիս կատարել այս և բազմաթիվ այլ գործողություններ սիմվոլային արտահայտությունների հետ:

Ընդհանրապես, արտահայտությունների պարզեցումը որոշակի կանոնի չի ենթարկվում. խնդրից կախված՝ պարզեցման տակ կարելի է տարբեր բաներ հասկանալ:

Օրինակ՝ $y=(a+b)^3$ և $y=a^3+3a^2b+3ab^2+b^3$ արտահայտությունները համարժեք են: Համարժեք են նաև $z=x^5+13x^4+\frac{215}{4}x^3+\frac{275}{4}x^2-\frac{27}{2}x-18$ և $z=\frac{1}{4}\left((2x+1)(2x-1)(x+6)(x-6)(x+4)(x+3)\right)$ արտահայտությունները: Կոնկրետ խնդիրը կարող է հուշել, թե որ մի տեսքով գրված արտահայտությունն է ավելի հարմար:

Հետևյալ ֆունկցիաները հնարավորություն են տալիս տարբեր կերպ պարզեցնելու կամ ձևափոխելու սիմվոլային արտահայտությունները:

- **simplify(y)** ֆունկցիան հնարավորինս պարզեցնում է արգումենտում գրված y սիմվոլային արտահայտությունը: Այն բավականին հզոր ֆունկցիա է, որը ներառում է ինչպես գումարելու, աստիճան բարձրացնելու, արմատ հանելու, կոտորակ կրճատելու գործողություններ, այնպես էլ բազմաթիվ նույնություններ՝ կապված եռանկյունաչափական, հիպերբոլական, լոգարիթմական և այլ հատուկ ֆունկցիաների հետ:

  ***Օրինակ** 1.*

```
>> syms x
>> f = (1-x^3)/(1-x);
>> simplify(f)
   ans = x^2 + x + 1
```

Օրինակ 2.

```
>> syms x
>> f = sin(x)^2 + cos(x)^2;
>> simplify(f)
   ans = 1
```

Օրինակ 3.

```
>> syms x y
>> f = exp(x)*exp(y);
>> simplify(f)
   ans = exp(x + y)
```

- **expand(y)** ֆունկցիան հնարավորություն է տալիս բացված տեսքով գրել արգումենտում գրված *y* սիմվոլային արտահայտությունը, որը կարող է լինել ինչպես հանրահաշվական, այնպես էլ պարունակել եռանկյունաչափական, հիպերբոլական, աստիճանային կամ լոգարիթմական գործողություններ:

Օրինակ 1.

```
>> syms a b
>> y = (a+b)^3;
>> f = expand(y)
   f = a^3 + 3*a^2*b + 3*a*b^2 + b^3
```

Օրինակ 2.

```
>> syms a x y
>> y = a*(x–y);
>> expand(y)
   ans = a*x – a*y
```

Օրինակ 3.

```
>> syms a b
>> y = cos(a+b);
>> expand(y)
   ans = cos(a)*cos(b) – sin(a)*sin(b)
```

- **factor(y)** ֆունկցիան հնարավորություն է տալիս ռացիոնալ գործակիցներով *y* բազմանդամն արտահայտելու ռացիոնալ գործակիցներով ավելի ցածր կարգի բազմանդամների արտադրյալների տեսքով: Եթե բազմանդամը հնարավոր չէ ներկայացնել այդպիսի արտադրյալների տեսքով, ապա **factor(y)** ֆունկցիայի արդյունքը կլինի հենց *y*–ը:

  Բացի բազմանդամից, *y*–ը կարող է լինել նաև սովորական իրական թիվ: Այս դեպքում **factor(y)** ֆունկցիան կվերադարձնի այդ թվի պարզ արտադրիչները վեկտորի տեսքով:

Օրինակ 1.

```
>> syms x
>> y = x^3 – 6*x^2 + 11*x – 6;
>> factor(y)
   ans = (x – 3)*(x – 1)*(x – 2)
```

Օրինակ 2.

```
>> factor(2782)
   ans = 2    13   107
```

- **collect(y,x)** ֆունկցիան առաջին արգումենտում գրված y սիմվոլային արտահայտությունը դիտարկում է որպես բազմանդամ ըստ երկրորդ արգումենտում գրված x փոփոխականի և միավորում x–ի նույն աստիճաններն ունեցող գործակիցները: Եթե երկրորդ արգումենտը չի գրվում, ապա **collect** ֆունկցիան y արտահայտությունը դիտարկում է որպես այն փոփոխականի բազմանդամ, որի առաջին տառն այբբենական կարգով x–ին ամենամոտն է:

Օրինակ.

```
>> syms x a
>> y = (x+a)^3 + (x–2)^2 + (x–a) + 7*x – 4;
>> pretty(y)
                    2          3
   8 x – a + (x – 2)  + (a + x)  – 4

>> f = collect(y);
>> pretty(f)
    3             2       2           3
   x  + (3 a + 1) x  + (3 a  + 4) x + a  – a

>> g = collect(y,a);
>> pretty(g)
    3          2       2                        2    3
   a  + (3 x) a  + (3 x  – 1) a + 8 x + (x – 2)  + x  – 4
```

Այս օրինակում y սիմվոլային արտահայտությունը դիտարկվում է որպես բազմանդամ մի անգամ ըստ x–ի (f արտահայտությունը), մյուս անգամ՝ ըստ a–ի (g արտահայտությունը): Ընդ որում, f –ը հայտարարելիս **collect** ֆունկցիայի երկրորդ արգումենտը գրելու կարիք չկա, որովհետև գործողությունը միևնույնն է կկատարվի ըստ x–ի, իսկ g–ն հայտարարելիս անպայման պետք է այն նշել:

- **coeffs(y,x)** ֆունկցիան առաջին արգումենտում գրված y սիմվոլային արտահայտությունը դիտարկում է որպես բազմանդամ ըստ երկրորդ արգումենտում գրված x փոփոխականի և վերադարձնում այդ բազմանդամի գործակիցները:

Օրինակ.

```
>> syms x a
>> y = (x+a)^3 + (x–2)^2 + (x–a) + 7*x – 4;

>> coeffs(y,x)
   ans = [ a^3 – a, 3*a^2 + 4, 3*a + 1, 1]

>> coeffs(y,a)
   ans = [ 8*x + (x – 2)^2 + x^3 – 4, 3*x^2 – 1, 3*x, 1]
```

- **subs(y,a,b)** ֆունկցիան y սիմվոլային արտահայտության մեջ a փոփոխականները (կամ արտահայտությունները) փոխարինում է b փոփոխականով (կամ արտահայտությամբ):

Օրինակ.

```
>> syms a b
>> y = (a^2+b^2)/(a^3–b^3);
>> pretty(y)

    2    2
   a  + b
   -------
    3    3
   a  – b

>> f = subs(y,'a','exp(a)+sin(a)');
>> pretty(f)

                    2    2
   (exp(a) + sin(a))  + b
   ----------------------
                    3    3
   (exp(a) + sin(a))  – b

>> f = subs(f,'b','exp(a)–cos(a)');
>> pretty(f)

                    2                      2
   (exp(a) + sin(a))  + (cos(a) – exp(a))
   ---------------------------------------
                    3                      3
   (exp(a) + sin(a))  + (cos(a) – exp(a))
```

8.7. ՍԻՄՎՈԼԱՅԻՆ ԱՐՏԱՀԱՅՏՈՒԹՅՈՒՆՆԵՐԻ ՍԱՀՄԱՆԸ

MatLab–ի սիմվոլային հաշվարկների փաթեթը հնարավորություն է տալիս որոշելու սիմվոլային արտահայտությունների սահմանը: $y(x)$ արտահայտության $\lim\limits_{x\to a} y(x)$ սահմանը հաշվելու համար նախատեսված է **limit(y,x,a)** ֆունկցիան:

Օրինակ 1.

Հաշվենք $\lim\limits_{x\to\infty}\left(1+\frac{1}{x}\right)^{ax}$ սահմանը:

```
>> syms a x
>> y = (1+1/x)^(a*x);
>> limit(y,x,inf)
   ans =exp(a)
```

Օրինակ 2.

Ինչպես հայտնի է, $y(x)$ ֆունկցիայի ածանցյալը $\frac{dy}{dx}=\lim\limits_{h\to 0}\frac{y(x+h)-y(x)}{h}$ սահմանն է, եթե այդ սահմանը գոյություն ունի: Օգտվելով այս սահմանումից՝ հաշվենք $\sin x$ ֆունկցիայի ածանցյալը:

```
>> syms x h
>> y = (sin(x+h)-sin(x))/h;
>> limit(y,h,0)
   ans = cos(x)
```

Բացի սովորական սահմաններից, **MatLab**–ը հնարավորություն է տալիս որոշելու նաև սիմվոլային արտահայտության միակողմանի սահմանները: Դրա համար անհրաժեշտ է որպես **limit** ֆունկցիայի 4–րդ արգումենտ չակերտների մեջ գրել 'left' հրամանը, եթե ցանկանում ենք հաշվել արտահայտության ձախակողմյան $\lim\limits_{x\to a^-} y(x)$ սահմանը և 'right', եթե ցանկանում ենք հաշվել արտահայտության $\lim\limits_{x\to a^+} y(x)$ աջակողմյան սահմանը:

Օրինակ 3.

Հաշվենք $y=\frac{1}{x}$ ֆունկցիայի ձախակողմյան և աջակողմյան սահմանները, երբ x –ը ձգտում է զրոյի:

```
>> syms x;
>> y = 1/x;
>> limit(y,x,0,'left')
   ans =-Inf
>> limit(y,x,0,'right')
   ans =Inf
```

Ընդ որում, ուշադրություն դարձնենք, որ այս ֆունկցիայի սովորական $\lim_{x\to 0}\frac{1}{x}$ սահմանը գոյություն չունի.

```
>> limit(y,x,0)
   ans = NaN
```

8.8. Սիմվոլային արտահայտությունների ածանցումը

MatLab–ի սիմվոլային հաշվարկների փաթեթը հնարավորություն է տալիս իրականացնել ածանցման հետևյալ գործողությունները.

- մեկ փոփոխականներից կախված ֆունկցիաների ածանցում,
- մի քանի փոփոխականներից կախված ֆունցկիաների մասնակի ածանցում,
- երկրորդ և ավելի բարձր կարգի ածանցումներ,
- խառը ածանցումներ:

Սիմվոլային արտահայտությունն ածանցելու համար նախատեսված է **diff** ֆունկցիան: Այս ֆունկցիան որպես պարտադիր արգումենտ պահանջում է սիմվոլային արտահայտության անունը: Բացի այդ, այն կարող է պարունակել ոչ պարտադիր ևս երկու արգումենտ, որոնցից մեկն այն փոփոխականի անունն է, ըստ որի կատարվում է ածանցումը, իսկ մյուսը՝ ածանցման կարգը: Եթե փոփոխականի անունը չի նշվում, ապա ածանցումը կատարվում է ըստ այն փոփոխականի, որի առաջին տառն այբբենական կարգով x–ին ամենամոտն է, իսկ ածանցման կարգը հատուկ չնշելու դեպքում կատարվում է առաջին կարգի ածանցում:

***Օրինակ** 1.*

```
>> syms x
>> f = x^3 + 3*cos(x)^2;
>> diff(f)
   ans = 3*x^2 – 6*cos(x)*sin(x)
```

Այս օրինակում $f = x^3 + 3\cos^2 x$ սիմվոլային արտահայտությունը կախված է միայն x փոփոխականից, հետևաբար այն նշելու կարիք չկա՝ ածանցումը կկատարվի ըստ x–ի: Քանի որ կարգը ևս նշված չի, ապա **diff(f)** ֆունկցիան իրականացնում է f արտահայտության առաջին կարգի ածանցումը:

***Օրինակ** 2:*

```
>> syms y z
>> f = sin(y)^3 + cos(z)^3;
>> dy = diff(f)
   dy = 3*cos(y)*sin(y)^2
>> dz = diff(f, z)
   dz = (–3)*cos(z)^2*sin(z)
>> d4z = diff(f, z, 4)
   d4z = 21*cos(z)^3 – 60*cos(z)*sin(z)^2
```

Այս օրինակում $f = \sin^3 x + \cos^3 x$ սիմվոլային արտահայտությունը կախված է y և z փոփոխականներից: Քանի որ *dy* փոփոխականին վերագրվում է *f* արտահայտության ածանցման արդյունքը, որի արգումենտում նշված չի, թե ըստ որ փոփոխականի պետք է այն ածանցվի, ապա ածանցումը կկատարվի ըստ *y*–ի, որովհետև այբբենական կարգով *y*–ը ավելի մոտ է *x*–ին, քան *z*–ը: Իսկ *dz* փոփոխականին վերագրվում է *f* արտահայտության ըստ *z*–ի ածանցման արդյունքը, քանի որ որպես **diff** ֆունկցիայի երկրորդ արգումենտ նշված է, որ ածանցումը պետք է կատարվի ըստ *z*–ի: Ի տարբերություն *dy*–ի և *dz*–ի, որոնք վերադարձնում էին *f*–ի առաջին կարգի ածանցյալները, *d4z* փոփոխականին վերագրվում է f արտահայտության 4–րդ կարգի ածանցման արդյունքը, քանի որ որպես **diff** ֆունկցիայի երրորդ արգումենտ գրվել է ածանցման կարգը:

Խառն ածանցյալներ հաշվելու համար անհրաժեշտ է միևնույն արտահայտության մեջ գրել անհրաժեշտ քանակով ածանցման հրամաններ:

***Օրինակ** 3.*

```
>> d2yz = diff(diff(f, y), z)
   d2yz = 0
```

Այս օրինակում որոշվեց և *d2yx* փոփոխականին վերագրվեց վերևում սահմանված *f* արտահայտության $\frac{\partial^2 f}{\partial x \partial y}$ խառն ածանցյալը:

8.9. Սիմվոլային արտահայտությունների ինտեգրումը

MatLab–ի սիմվոլային հաշվարկների փաթեթում անորոշ, որոշյալ, կրկնակի և ավելի բարձր կարգի ինտեգրալներ հաշվելու միջոցներ կան: Ինտեգրալներ հաշվելու համար նախատեսված է **int** ֆունկցիան, որը որպես պարտադիր արգումենտ պահանջում է սիմվոլային արտահայտության անունը, իսկ որպես ոչ պարտադիր երկրորդ արգումենտ՝ այն փոփոխականի անունը, ըստ որի կատարվելու է ինտեգրումը: Ինչպես սիմվոլային ածանցման դեպքում, եթե փոփոխականի անունը հատուկ չի նշվում, ապա ինտեգրումը կատարվում է ըստ այն փոփոխականի, որի առաջին տառն այբբենական կարգով *x*–ին ամենամոտն է:

- **Անորոշ ինտեգրալ:** Սիմվոլային արտահայտության անորոշ ինտեգրալը որոշելու համար **int** ֆունկցիան որպես արգումենտ պահանջում է միայն սիմվոլային արտահայտությունը և որպես ոչ պարտադիր երկրորդ արգումենտ՝ ինտեգրման փոփոխականի անունը:

  ***Օրինակ** 1.*

  Հաշվենք $\int x \sin x dx$ անորոշ ինտեգրալը:

```
>> syms x
>> y = x*sin(x);
>> I = int(y,x)
   I =sin(x) – x*cos(x)
```

- **Որոշյալ ինտեգրալ:** Սիմվոլային արտահայտության որոշյալ ինտեգրալը հաշվելու համար **int** ֆունկցիան բացի սիմվոլային արտահայտությունից և ինտեգրման փոփոխականի անունից, պահանջում է նաև լրացուցիչ երկու արգումենտ, որոնցից առաջինը ինտեգրալի ստորին սահմանն է, իսկ երկրորդը՝ վերին:

Օրինակ 2.

Հաշվենք $\int_2^4 x\sin x dx$ որոշյալ ինտեգրալը:

```
>> syms x
>> y = x*sin(x);
>> I = int(y,x,2,4)
   I =2*cos(2) – 4*cos(4) – sin(2) + sin(4)
```

- **Բարձր կարգի ինտեգրալներ:** Սիմվոլային արտահայտության կրկնակի և ավելի բարձր կարգի ինտեգրալները հաշվելու համար առանձին կանոն չկա: Այս խնդիրներում **int** ֆունկցիան պարզապես պետք է օգտագործել անհրաժեշտ թվով անգամ:

Օրինակ 3.

Հաշվենք $\int_a^b\int_c^d y\sin x dxdy$ կրկնակի ինտեգրալը:

```
>> syms x y a b c d
>> z = y*sin(x);
>> Ix = int(z,x,c,d)
   Ix =y*(cos(c) – cos(d))
>> I = int(Ix,y,a,b)
   I =–((a^2 – b^2)*(cos(c) – cos(d)))/2
```

Այս օրինակում նախ ինտեգրվող z ֆունկցիան ինտեգրվում է ըստ x–ի c–ից մինչև d, ստացված արդյունքը վերագրվում է Ix փոփոխականին, այնուհետև Ix–ը ինտեգրվում է արդեն ըստ y–ի a–ից մինչև b:

Ինտեգրալները միշտ չէ, որ հնարավոր է հաշվել անալիտիկորեն: Այդպիսի դեպքերում **MatLab**–ը վերադարձնում է զգուշացում, և միակ տարբերակը մնում է խնդիրը լուծել թվային մոտավոր եղանակներով:

Օրինակ 4.

```
>> syms x
>> y = sin(x^2)*exp(sqrt(x));
>> int(y,x,1,2)
   Warning: Explicit integral could not be found.
   ans = int(sin(x^2)*exp(x^(1/2)), x = 1..2)
```

8.10. Սիմվոլային արտահայտությունների գումարը

MatLab–ի սիմվոլային հաշվարկների փաթեթը թույլ է տալիս հաշվել y սիմվոլային արտահայտության $\sum_{n=a}^{b} y_n$ գումարը, եթե վերջինս գոյություն ունի: Դրա համար նախատեսված է **symsum(y,n,a,b)** ֆունկցիան, որտեղ y–ը սիմվոլային արտահայտությունն է, n–ը՝ այն փոփոխականը, ըստ որի պետք է գումարումը կատարվի, a–ն գումարի ստորին սահմանն է, b–ը՝ վերինն:

Օրինակ 1.

Հաշվենք $S = \sum_{x=1}^{32} x(x+1)$ գումարը.

```
>> syms x
>> y = x*(x+1);
>> S = symsum(y,x,1,32)
   S = 11968
```

Օրինակ 2.

Հաշվենք $S = \sum_{x=1}^{\infty} \frac{1}{n^2}$ գումարը.

```
>> syms n
>> y = 1/n^2;
>> S = symsum(y,n,1,Inf)
   S = pi^2/6
```

8.11. Սիմվոլային արտահայտության վերլուծումը Թեյլորի շարքի

MatLab–ի սիմվոլային հաշվարկների փաթեթում սիմվոլային արտահայտությունը Թեյլորի շարքի վերլուծելու հնարավորություն է նախատեսված որևէ a կետի շուրջը: $f(x)$ ֆունկցիայի Թեյլորի շարքը, ինչպես հայտնի է, ունի հետևյալ տեսքը.

$$\sum_{n=0}^{\infty} (x-a)^n \frac{f^{(n)}(a)}{n!}:$$

Ֆունկցիան Թեյլորի շարքի վերլուծելու համար **MatLab**–ում նախատեսված է **taylor(f,n,x,a)** ֆունցկիան, որը f սիմվոլային արտահայտությունը վերլուծում է (n–1)–րդ կարգի Թեյլորի շարքի ըստ x փոփոխականի՝ a կետի շուրջը: 0 կետի շուրջ շարքի վերլուծելու համար **taylor** ֆունկցիայի 4–րդ արգումենտը կարելի է չգրել:

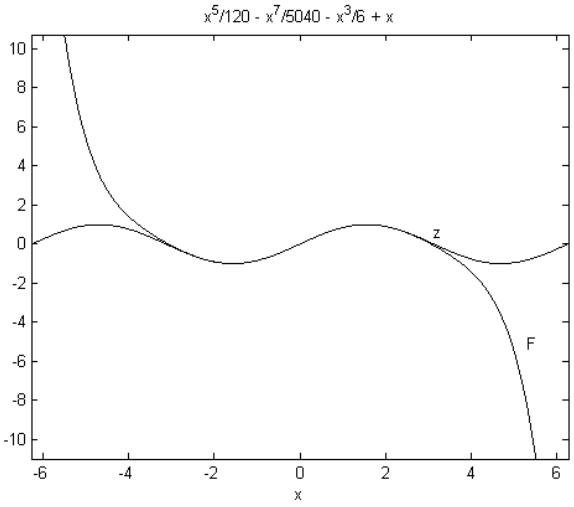
Օրինակ 1.

$\sin x$ ֆունկցիան վերլուծենք 8–րդ կարգի Թեյլորի շարքի 0 կետի շուրջ.

```
>> syms x
>> z = sin(x);
>> F = taylor(z,8,x)
   F =x^5/120 – x^7/5040 – x^3/6 + x
```

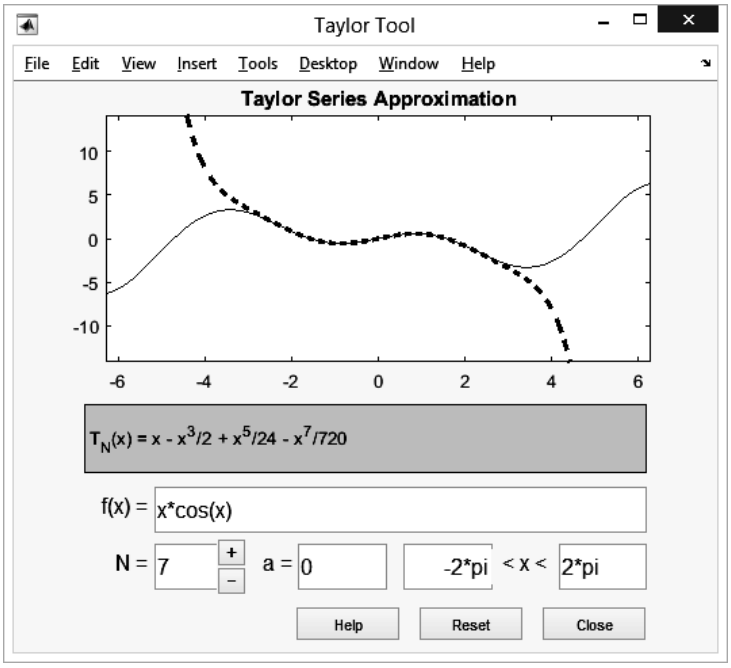
Որպեսզի արդյունքն ավելի տեսանելի լինի, կարելի է կառուցել z ֆունկցիայի և իր F Թեյլորի շարքի գրաֆիկները միևնույն պատուհանում և համեմատել: Ակնհայտ է, որ որքան շարքի կարգը ավելի մեծ լինի, այնքան շարքն ավելի մոտ կլինի z ֆունկցիայի արժեքներին (նկ. 68):

```
>> ezplot(z)
>> hold on
>> ezplot(F)
```



նկ. 68

Ֆունկցիան կարելի է վերլուծել Թեյլորի շարքի և տեսնել այդ ֆունկցիայի և նրա Թեյլորի շարքի գրաֆիկական ներկայացումը նաև **Taylor Tool** ենթածրագրի օգնությամբ: Դրա համար անհրաժեշտ է հրամանի տողում գրել **taylortool** հրամանը: Բացված պատուհանում կարելի է գրել դիտարկվող $f(x)$ ֆունկցիան, այն x տիրույթը, որում պետք է պատկերվի ֆունկցիայի և նրա շարքի գրաֆիկը, այն a կետը, որի շուրջ պետք է ֆունկցիան վերլուծվի Թեյլորի շարքի, ինչպես նաև Թեյլորի շարքի N կարգը: Պատուհանը բացվելիս լռելյայն տրված է $x\cos x$ ֆունկցիան, որը վերլուծված է 7–րդ կարգի Թեյլորի շարքի 0 կետի շուրջ, իսկ գրաֆիկները պատկերված են $[-2\pi, 2\pi]$ տիրույթում (նկ. 69):



նկ. 69

8.12. ՀԱՆՐԱՀԱՇՎԱԿԱՆ ՀԱՎԱՍԱՐՈՒՄՆԵՐԻ ԵՎ ՀԱՄԱԿԱՐԳԵՐԻ ԼՈՒԾՈՒՄ

MatLab–ի սիմվոլային հաշվարկների փաթեթում հանրահաշվական հավասարումները և հանրահաշվական հավասարումների համակարգերը անալիտիկորեն լուծելու հնարավորություն կա: $f(x)=0$ տեսքի հավասարումներ լուծելու համար նախատեսված է **solve(f, x)** ֆունկցիան, որտեղ f–ը հանրահաշվական հավասարման ձախ մասն է՝ գրված սիմվոլային արտահայտության տեսքով, իսկ x սիմվոլային փոփոխականը՝ որոնվող լուծումը: Նկատի ունենանք, որ այս երկրորդ արգումենտը կարելի է չգրել: Այս դեպքում հավասարումը կլուծվի ըստ այն փոփոխականի, որը f սիմվոլային արտահայտության մեջ այբբենական կարգով x–ին ամենամոտն է:

***Օրինակ** 1.*

Լուծենք $ax^2+bx+c=0$ քառակուսի հավասարումը:

```
>> syms a b c x
>> f = a*x^2 + b*x + c;
>> solve(f, x)
   ans =
       -(b + (b^2 – 4*a*c)^(1/2))/(2*a)
       -(b – (b^2 – 4*a*c)^(1/2))/(2*a)
```

Օրինակ 2.

Լուծենք $x^3 = 5x$ հավասարումը:

```
>> syms x
>> f = x^3 - 5*x;
>> solve(f,x)
   ans =
        0
        5^(1/2)
        -5^(1/2)
```

Նախորդ երկու օրինակում սիմվոլային արտահայտությունները բերվում էին $f(x)=0$ տեսքի, և հավասարման ձախ կողմում գրված արտահայտությունն էր գրվում որպես **solve** ֆունկցիայի արգումենտ: Սակայն **MatLab**–ի նոր տարբերակները թույլ են տալիս **solve** ֆունկցիան օգտագործել՝ առանց հավասարման աջ մասը զրոյի հավասարեցնելու, այսինքն՝ հավասարումը գրելով $f(x)=g(x)$ տեսքով: Այդ դեպքում **solve** ֆունկցիայի արգումենտները գրվում են որպես տող (ապոստրոֆներում), ընդ որում առաջին արգումենտում հավասարումը գրվում է լրիվ տեսքով՝ որպես հավասարության նշան օգտագործելով == օպերատորը:

Օրինակ 3.

Նորից դիտարկենք նախորդ օրինակի $x^3 = 5x$ հավասարումը:

```
>> solve('x^3==5*x', 'x')
   ans =
     0
     5^(1/2)
     -5^(1/2)
```

Հանրահաշվական հավասարումների համակարգ լուծելու համար անհրաժեշտ է սահմանել անհրաժեշտ սիմվոլային փոփոխականները և ֆունկցիաները (զրոյի հավասարեցված հավասարումների ձախ մասերը՝ որպես սիմվոլային արտահայտություն), այնուհետև դրանք օգտագործել՝ որպես մեկ ելքային արգումենտ ունեցող **solve** ֆունկցիայի մուտքեր: Ընդ որում, **solve** ֆունկցիայի մուտքային արգումենտներում նախ պետք է գրվեն սիմվոլային ֆունկցիաները, այնուհետև՝ փոփոխականները: Արդյունքն իրենից ներկայացնում է կառուցվածքային տվյալների տիպ, այսինքն՝ խմբավորված փոփոխականների հաջորդականություն, որոնք կլինեն հավասարումների համակարգի լուծումները:

Օրինակ 4.

Լուծենք հետևյալ հավասարումների համակարգը.

$$\begin{cases} x(2-y) = \cos x \cdot e^y \\ 2 + x - y = \cos x + e^y \end{cases}$$

```
>> syms x y
>> f1 = x*(2-y)-cos(x)*exp(y);
>> f2 = 2+x-y-cos(x)-exp(y);
>> s = solve(f1, f2, x, y)
   s =
       x: [1x1 sym]
       y: [1x1 sym]
>> s.x
   ans = 0.739085133215160641655312087673 87
>> s.y
   ans = 0.442854401002388583141327999999 34
```

Ինչպես երևում է օրինակից, **solve** ֆունկցիայի ելքում ստացվում է *s* կառուցվածքը, որի մեջ պարունակվում են *x* և *y* սիմվոլային փոփոխականները: Վերջիններիս արժեքները չեն երևում, քանի դեռ անմիջականորեն չենք դիմում դրանց՝ օգտագործելով համապատասխանաբար *s.x* և *s.y* հրամանները:

8.13. ԴԻՖԵՐԵՆՑԻԱԼ ՀԱՎԱՍԱՐՈՒՄՆԵՐԻ ԵՎ ՀԱՎԱՍԱՐՈՒՄՆԵՐԻ ՀԱՄԱԿԱՐԳԵՐԻ ԼՈՒԾՈՒՄԸ

MatLab–ի սիմվոլային հաշվարկների փաթեթում սահմանված **dsolve** ֆունկցիան նախատեսված է սովորական դիֆերենցիալ հավասարումները և դիֆերենցիալ հավասարումների համակարգերը անալիտիկորեն լուծելու համար: Ընդ որում, հնարավոր է գտնել դիֆերենցիալ հավասարումների և՛ անորոշ հաստատունից կախված ընդհանուր լուծումները, և՛ որոշակի սկզբնական կամ եզրային պայմանների բավարարող լուծումները:

Դիֆերենցիալ հավասարումներ լուծելու համար անհրաժեշտ է որպես արգումենտներ տողերի տեսքով (ապոստրոֆների մեջ) գրել հավասարումը, սկզբնական կամ եզրային պայմանը և անկախ փոփոխականը: Եթե սկզբնական կամ եզրային պայմանը չի գրվում, **dsolve** ֆունկցիան գտնում է դիֆերենցիալ հավասարման ընդհանուր լուծումը, իսկ անկախ փոփոխականը չգրելու դեպքում հավասարումը լուծվում է ըստ *t* փոփոխականի: Արգումենտներում առաջին, երկրորդ և ավելի բարձր կարգի ածանցյալները գրելու համար ֆունկցիայի անվան դիմաց գրվում են *D*, *D*2,... տառերը: Օրինակ, եթե ֆունկցիան *y*–ն է, ապա առաջին կարգի ածանցյալը կգրվի որպես *Dy*, երկրորդ կարգի ածանցյալը՝ *D*2*y* և այլն:

***Օրինակ** 1.*

Լուծենք $\frac{dy}{dx} + y = x^2$ դիֆերենցիալ հավասարումը:

```
>> y = dsolve('Dy+y=x^2', 'x')
   y = x^2 - 2*x + C37/exp(x) + 2
```

Քանի որ հավասարումը լուծվում է առանց սկզբնական պայմանի, *y* փոփոխականին վերագրվում է հավասարման ընդհանուր լուծումը, որը կախված է *C*37 հաստատունից:

Օրինակ 2.

Դարձյալ լուծենք $\frac{dy}{dx}+y=x^2$ դիֆերենցիալ հավասարումը՝ այս անգամ $y(0.5)=1$ սկզբնական պայմանով:

```
>> y = dsolve('Dy+y=x^2', 'y(0.5)=1', 'x')
   y = x^2 – exp(1)^(1/2)/(4*exp(x)) – 2*x + 2
```

Օրինակ 3.

Լուծենք $\frac{d^3u}{dx^3}=u$ երրորդ կարգի դիֆերենցիալ հավասարումը՝ $u(0)=1$, $u'(0)=1$ և $u''(0)=\pi$ սկզբնական պայմաններով:

```
>> y = dsolve('D3u=u', 'u(0)=1', 'Du(0)=–1', 'D2u(0)=pi', 'x')
>> y = (pi*exp(x))/3 – (cos((3^(1/2)*x)/2)*(pi/3–1))/exp(x/2)...
   (3^(1/2)*sin((3^(1/2)*x)/2)*(pi + 1))/(3*exp(x/2))
```

Դիֆերենցիալ հավասարումների համակարգեր լուծելու համար անհրաժեշտ է որպես **dsolve** ֆունկցիայի արգումենտներ տողերի տեսքով նախ գրել համապատասխան հավասարումները, այնուհետև՝ սկզբնական կամ եզրային պայմանները և վերջում՝ անկախ փոփոխականը: Ինչպես հանրահաշվական հավասարումների համակարգի դեպքում, այստեղ ևս ֆունկցիան վերադարձնում է կառուցվածքային տիպ, որում պարունակվում են հավասարումների լուծումները:

Օրինակ 4.

Լուծենք $\begin{cases}\frac{dy}{dt}=3y+4z\\ \frac{dz}{dt}=-4y+3z\end{cases}$ դիֆերենցիալ հավասարումների համակարգը՝ $y(0)=0$ և $z(0)=1$ սկզբնական պայմաններով:

```
>> s = dsolve('Dy=3*y+4*g', 'Dz=–4*y+3*z', 'y(0)=0', 'z(0)=1')
   s =
       z: [1x1 sym]
       y: [1x1 sym]
>> s.z
   ans = exp(3*t)*((16*g)/9 + 1) – (16*g)/9 – (16*g*t*exp(3*t))/3
>> s.y
   ans = (4*g*exp(3*t))/3 – (4*g)/3
```

Ուշադրություն դարձրեք, որ, ի տարբերություն նախորդ օրինակների, այստեղ **dsolve** ֆունկցիայի արգումենտում կարիք չկար գրելու *t* անկախ փոփոխականի անունը, քանի որ այն չգրելու դեպքում դիֆերենցիալ հավասարումը լուծվում է ըստ t փոփոխականի:

8.14. ՎԱՐԺՈՒԹՅՈՒՆՆԵՐ

1. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **factor** ներդրված ֆունկցիան՝ տրված արտահայտությունները վերլուծեք պարզ արտադրիչների.

- $f = x^3 + 4x^2 - 3x - 12$,
- $f = x^4 - 2x + 1$,
- $f = x^2 - 4y^2$,
- $f = x^2 - 16$:

2. Հայտարարեք հետևյալ սիմվոլային թվերը և օգտագործելով **factor** ներդրված ֆունկցիան՝ դրանք ներկայացրեք պարզ թվերի արտադրյալների տեսքով.

- 3542,
- 230010,
- 23012452,
- 1574:

3. Հայտարարեք x և y սիմվոլային փոփոխականները և օգտագործելով **expand** ներդրված ֆունկցիան՝ տրված արտահայտությունները ներկայացրեք բացված տեսքով.

- $(x+y)^2$,
- $\cos(x+y)$,
- e^{2x-y},
- $(x-y)^5$,
- $\sin(x-y)$,
- $\operatorname{ctg} 3x$:

4. Հայտարարեք x և y սիմվոլային փոփոխականները և օգտագործելով **simplify** ներդրված ֆունկցիան՝ պարզեցրած տեսքով ներկայացրեք տրված արտահայտությունները.

- $x^3 + 4x^2 - 3x - 12$,
- $e^{-3\ln x + 1}$,
- $\operatorname{ch}^2 x - \operatorname{sh}^2 x$,
- $3\cos^2(4x) + 3\sin^2(4x)$,
- $\operatorname{tg} x \operatorname{ctg} x$,
- $\ln \dfrac{e^{2x}}{e^{3x} e^{x}}$:

5. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **subs** ներդրված ֆունկցիան՝ $2(x^2+3x-6) + \dfrac{(x^2+3x-6)^4 - 5}{3 + (x^2+3x-6)^7} - (x^2+3x-6)^2$ արտահայտության մեջ կատարեք $z = x^2 + 3x - 6$ փոխարինումը:

6. Հայտարարեք x և a սիմվոլային փոփոխականները և օգտագործելով **collect** ներդրված ֆունկցիան՝ միավորեք $p = (x+a)^4 + (x-1)^3 - (x-a)^2 - ax + x - 3$ բազմանդամի գործակիցները մի անգամ ըստ x փոփոխականի, մի անգամ՝ ըստ a -ի:

7. Հայտարարեք x և a սիմվոլային փոփոխականները և օգտագործելով **coeffs** ներդրված ֆունկցիան՝ ստացեք $p = (x-a)^5 + (x+2a)^4 - (3x-8a)^3 - a^2 x - 2x + 43a$ բազմանդամի գործակիցները մի անգամ ըստ x փոփոխականի, մի անգամ՝ ըստ a –ի:

8. Հայտարարեք x, y սիմվոլային փոփոխականները և օգտագործելով **ezplot** ներդրված ֆունկցիան՝ կառուցեք հետևյալ ֆունկցիաների գրաֆիկները.

- $y = \lg(x^2 - 3x + 2)$,
- $y = \sqrt{1+x^2}\,\ln(1+\sqrt{1+x^2})$,
- $y = \cos^2 \pi x \sin 3\pi x$,
- $y = \sqrt{1-x^2}\,\sin\dfrac{\pi}{x}$,
- $y = x + e^{-x}$,
- $y = |1-x| + |1+x|$:

9. Հայտարարեք x, y, z սիմվոլային փոփոխականները և օգտագործելով **ezsurf** կամ **ezmesh** ներդրված ֆունկցիան՝ կառուցեք հետևյալ ֆունկցիաների մակերևույթները.

- $z = \dfrac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$,
- $z = \sqrt{1-x^2-y^2}$,
- $z = \cos x \cos y$,
- $z = (x^2 - y^2)e^{-x^2-y^2}$,
- $y = \dfrac{\sqrt{|xy|}}{1+x^2+y^2}$,
- $z = \cos(x^2+y^2)$:

10. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **limit** ներդրված ֆունկցիան՝ որոշեք հետևյալ սահմանները.

- $\lim\limits_{x\to 1} \arcsin x$,
- $\lim\limits_{x\to 0} x\,\mathrm{ctg}\,3x$,
- $\lim\limits_{x\to 1} \lg\dfrac{100+x^2}{1+100x^2}$,
- $\lim\limits_{x\to 0^-} \sqrt{10+x}$,
- $\lim\limits_{x\to 0^+} \sqrt{10+x}$,
- $\lim\limits_{x\to\infty} \dfrac{\sqrt{x+\sqrt{x+\sqrt{x}}}}{\sqrt{x+1}}$:

11. Հայտարարեք x, a, b սիմվոլային փոփոխականները և օգտագործելով **diff** ներդրված ֆունկցիան՝ որոշեք հետևյալ ածանցյալները.

- $\dfrac{d}{dx}(e^{ax}\sin bx)$,
- $\dfrac{d}{dx}(x^2 - \sqrt{x} - 3)$,
- $\dfrac{d}{dx}\left((\sin x)^{\cos x} - (\cos x)^{-\sin x}\right)$,
- $\dfrac{d}{dx}\left(\dfrac{(\ln x)^x}{x^{\ln x}}\right)$,
- $\dfrac{d^2}{dx^2}\left(x - \sqrt{1-x^4}\,\arccos x\right)$,
- $\dfrac{d^3}{dx^3}e^{x^8}$,
- $\dfrac{d^2}{dx^2}\left(\dfrac{(\lg x)^x + 15x^2}{x^{\lg x}}\right)$,
- $\dfrac{d^4}{dx^4}(\lg\lg\lg x)$:

12. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **int** ներդրված ֆունկցիան՝ որոշեք հետևյալ անորոշ ինտեգրալները.

- $\int \frac{dx}{1+x+x^2}$,
- $\int \sin \ln x dx$,
- $\int (3x+2)\sin 4x dx$,
- $\int \frac{dx}{\sin^2 x}$,
- $\int \frac{dx}{(1+x^2)\operatorname{arctg}^2 x}$,
- $\int \frac{\arccos x}{\sqrt{1-x^2}} dx$:

13. Հայտարարեք x, y սիմվոլային փոփոխականները և օգտագործելով **int** ներդրված ֆունկցիան՝ որոշեք հետևյալ որոշյալ և կրկնակի ինտեգրալների արժեքները.

- $\int_4^9 \frac{dx}{(\sqrt{x}-1)\sqrt{x}}$,
- $\int_{\pi^2/9}^{\pi^2} \frac{\cos\sqrt{x}}{\sqrt{x}} dx$,
- $\int_4^9 \frac{\operatorname{tg}(x+2)}{\cos^2(x+1)} dx$,
- $\int_{\ln 2}^{\ln 3} \frac{dx}{e^x + e^{-x}}$,
- $\int_2^4 \frac{2x+3}{x(x-1)(x+2)} dx$,
- $\int_0^{\pi/2}\int_0^1 y\sin x dy dx$,
- $\int_1^\infty \frac{dx}{x}$,
- $\int_1^2\int_0^3 (1+8xy)dxdy$:

14. Հայտարարեք n սիմվոլային փոփոխականը և օգտագործելով **symsum** ներդրված ֆունկցիան՝ հաշվեք հետևյալ գումարները.

- $\sum_{n=0}^{20} n^2$,
- $\sum_{n=2}^{\infty} \ln\left(1-\frac{1}{n^2}\right)$,
- $\sum_{n=1}^{\infty} \frac{(-1)^n}{2^n}$,
- $\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$,
- $\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$,
- $\sum_{n=1}^{\infty} \left(\sqrt{n+2} - 2\sqrt{n+1} + \sqrt{n}\right)$:

15. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **taylor** ներդրված ֆունկցիան՝ հետևյալ ֆունկցիաները վերլուծեք Թեյլորի շարքի.

- $y = \operatorname{th} x$ ֆունկցիան՝ 8–րդ կարգի, 0 կետի շուրջ,
- $y = \operatorname{ch} x$ ֆունկցիան՝ 8–րդ կարգի, 0 կետի շուրջ,
- $y = \operatorname{arctg} x$ ֆունկցիան՝ 8–րդ կարգի, 0 կետի շուրջ,
- $y = \cos x$ ֆունկցիան՝ 10–րդ կարգի, 1/2 կետի շուրջ,

- $y = e^x$ ֆունկցիան՝ 10–րդ կարգի, 1/2 կետի շուրջ,
- $y = \ln(1 + x^2)$ ֆունկցիան՝ 10–րդ կարգի, 1 կետի շուրջ:

16. Հայտարարեք x սիմվոլային փոփոխականը և օգտագործելով **solve** ներդրված ֆունկցիան՝ որոշեք հետևյալ հավասարումների արմատները.

- $2x^4 + 8x^3 + 8x^2 - 1 = 0$,
- $2\operatorname{arctg} x - x + 3 = 0$,
- $\ln x = \sin x$,
- $\sin(x - 0.5) - x + 0.8 = 0$,
- $3^{x-1} - 4 - x = 0$,
- $e^{-2x} - 2x + 1 = 0$:

17. Հայտարարեք x, y, z, u սիմվոլային փոփոխականները և օգտագործելով **solve** ներդրված ֆունկցիան՝ որոշեք հետևյալ հավասարումների համակարգերի արմատները.

- $\begin{cases} \cos(y + 0.5) - x = 2 \\ \sin x - 2y = 1 \end{cases}$,
- $\begin{cases} x^2 \sin y = 0 \\ y \cos x^2 = 1 \end{cases}$,
- $\begin{cases} x + 2xy = 3 \\ 4x + y = 2 \end{cases}$,
- $\begin{cases} x + y + z = 3 \\ 5x + 4y + 3z = 11 \\ 10x + 5y + z = 11.5 \end{cases}$,
- $\begin{cases} x(2 - y) = \cos x \cdot e^y \\ \sin x - 2y = 1 \end{cases}$,
- $\begin{cases} 2x + 3y - z + u = -3 \\ 3x - y + z + 4u = 8 \\ x + y + 3z - 2u = 6 \\ -x + 2y + 3z + 5u = 3 \end{cases}$:

18. Հայտարարեք x, y սիմվոլային փոփոխականները և օգտագործելով **dsolve** ներդրված ֆունկցիան՝ լուծեք հետևյալ դիֆերենցիալ հավասարումները.

- $x\dfrac{dy}{dx} + y = \sin x$, եթե $y\left(\dfrac{\pi}{2}\right) = \dfrac{2}{\pi}$,
- $\dfrac{dy}{dx} + y^2 = x^{-2}$, եթե $y(0.5) = -1$,
- $\dfrac{dy}{dx} = -y + e^{3x}$, եթե $y(0) = 2$,
- $\dfrac{dy}{dx} + 5y = 35$, եթե $y(0) = 4$,
- $\dfrac{d^2y}{dx^2} + 7\dfrac{dy}{dx} + 5y = 8$, եթե $y(0) = 1$ և $y'(0) = 2$:

19. Հայտնի է, որ $\varphi(x,y,z)$ սկալյար ֆունկցիայի գրադիենտը որոշվում է $\operatorname{grad}\varphi = \hat{i}\dfrac{\partial\varphi}{\partial x} + \hat{j}\dfrac{\partial\varphi}{\partial y} + \hat{k}\dfrac{\partial\varphi}{\partial z}$ բանաձևով: Օգտվելով այս առնչությունից՝ ցույց տվեք, որ

- $\operatorname{grad} r = \dfrac{\vec{r}}{r}$,
- $\operatorname{grad} r^n = nr^{n-2}\vec{r}$,
- $\operatorname{grad}(\vec{a}\cdot\vec{r}) = \vec{a}$,
- $\operatorname{grad}(\varphi\psi) = \varphi\operatorname{grad}\psi + \psi\operatorname{grad}\varphi$,

որտեղ φ–ն և ψ–ն x, y, z փոփոխականներից կախված սկալյար ֆունկցիաներ են, $\vec{r}$–ը և r–ը՝ համապատասխանաբար շառավիղ–վեկտորը և դրա մոդուլը՝

$$\vec{r} = \hat{i}x + \hat{j}y + \hat{k}z, \qquad r = \sqrt{x^2+y^2+z^2},$$

իսկ $\vec{a}$–ն հաստատուն վեկտոր է ($\vec{a} = \hat{i}a_x + \hat{j}a_y + \hat{k}a_z$):

20. Հայտնի է, որ $\vec{A}(x,y,z)$ վեկտորական ֆունկցիայի դիվերգենցիան որոշվում է $\operatorname{div}\vec{A} = \dfrac{\partial A_x}{\partial x} + \dfrac{\partial A_y}{\partial y} + \dfrac{\partial A_z}{\partial z}$ բանաձևով: Օգտվելով այս առնչությունից՝ ցույց տվեք, որ

- $\operatorname{div}\vec{r} = 3$,
- $\operatorname{div}(\vec{a}\cdot\vec{r})\vec{r} = 4(\vec{a}\cdot\vec{r})$,
- $\operatorname{div}(\vec{a}\times\vec{r}) = 0$,
- $\operatorname{div}(\vec{r}\times(\vec{a}\times\vec{r})) = -2(\vec{a}\cdot\vec{r})$,

որտեղ φ–ն x, y, z կոորդինատներից կախված սկալյար ֆունկցիա է, $\vec{r}$–ը՝ շառավիղ–վեկտորը ($\vec{r} = \hat{i}x + \hat{j}y + \hat{k}z$), իսկ $\vec{a}$–ն՝ հաստատուն վեկտոր ($\vec{a} = \hat{i}a_x + \hat{j}a_y + \hat{k}a_z$):

21. Հայտնի է, որ $\vec{A}(x,y,z)$ վեկտորական ֆունկցիայի ռոտորը որոշվում է $\operatorname{rot}\vec{A} = \hat{i}\left(\dfrac{\partial A_z}{\partial y} - \dfrac{\partial A_y}{\partial z}\right) + \hat{j}\left(\dfrac{\partial A_x}{\partial z} - \dfrac{\partial A_z}{\partial x}\right) + \hat{k}\left(\dfrac{\partial A_y}{\partial x} - \dfrac{\partial A_x}{\partial y}\right)$ բանաձևով: Օգտվելով այս առնչությունից՝ ցույց տվեք, որ

- $\operatorname{rot}\vec{r} = 0$,
- $\operatorname{rot}(\vec{a}\cdot\vec{r})\vec{b} = \vec{a}\times\vec{b}$,
- $\operatorname{rot}(\vec{a}\times\vec{r}) = 2\vec{a}$,
- $\operatorname{rot}(\vec{a}\cdot\vec{r})\vec{r} = \vec{a}\times\vec{r}$,

որտեղ φ–ն x, y, z կոորդինատներից կախված սկալյար ֆունկցիա է, $\vec{r}$–ը՝ շառավիղ–վեկտորը ($\vec{r} = \hat{i}x + \hat{j}y + \hat{k}z$), իսկ $\vec{a}$–ն՝ հաստատուն վեկտոր ($\vec{a} = \hat{i}a_x + \hat{j}a_y + \hat{k}a_z$):

22. Ցույց տվեք, որ

- $\operatorname{div}(\varphi\vec{A}) = \varphi\operatorname{div}\vec{A} + \vec{A}\cdot\operatorname{grad}\varphi$,
- $\operatorname{div}(\vec{A}\times\vec{B}) = \vec{B}\cdot\operatorname{rot}\vec{A} - \vec{A}\cdot\operatorname{rot}\vec{B}$,
- $\operatorname{rot}(\varphi\vec{A}) = \varphi\operatorname{rot}\vec{A} - \vec{A}\times\operatorname{grad}\varphi$,
- $\operatorname{rotrot}\vec{A} = \operatorname{graddiv}\vec{A} - \nabla^2\vec{A}$,

որտեղ φ–ն, $\vec{A}$–ն և $\vec{B}$-ն կախված են x, y, z կոորդինատներից, իսկ $\nabla^2 = \dfrac{\partial^2}{\partial x^2} + \dfrac{\partial^2}{\partial y^2} + \dfrac{\partial^2}{\partial z^2}$–ն Լապլասի օպերատորն է:

23. Էլեկտրաստատիկայում լիցքերի ρ ծավալային խտությունը և նրանց ստեղծած $\vec{E}$ լարվածությունը բավարարում են Մաքսվելի $\operatorname{div}\vec{E} = 4\pi\rho$ հավասարմանը: Որոշեք լիցքերի ρ խտությունը, եթե

- $\vec{E} = (\vec{a}\cdot\vec{r})\vec{a}$,
- $\vec{E} = qr\vec{r}$,
- $\vec{E} = \dfrac{q\vec{r}}{r^3}\left\{1 - \left[1 + \dfrac{2r}{a}\left(1 + \dfrac{r}{a}\right)\right]\exp\left(-\dfrac{2r}{a}\right)\right\}$:

24. Մագնիսաստատիկայում էլեկտրական հոսանքի $\vec{j}$ խտությունը և մագնիսական դաշտի $\vec{H}$ լարվածությունը կապված են Մաքսվելի $\operatorname{rot}\vec{H} = \dfrac{4\pi}{c}\vec{j}$ հավասարումով: Որոշեք հոսանքի $\vec{j}$ խտությունը, եթե

- $\vec{H} = (\vec{a}\cdot\vec{r})(\vec{b}\times\vec{r})$, որտեղ $\vec{a}$–ն և $\vec{b}$–ն հաստատուն վեկտորներ են,
- $\vec{H} = f(r)(\vec{a}\times\vec{r})$, որտեղ $f(r)$–ը կոորդինատներից կախված կամայական ածանցելի ֆունկցիա է, իսկ $\vec{a}$–ն՝ հաստատուն վեկտոր:

25. Մաքսվել–Բոլցմանի հավանականային խտության ֆունկցիան տրվում է $f(u) = \sqrt{\dfrac{2}{\pi}\left(\dfrac{m}{kT}\right)^3}\,u^2\exp\left(-\dfrac{mu^2}{2kT}\right)$ օրենքով, որտեղ m–ը յուրաքանչյուր մոլեկուլի զանգվածն է՝ արտահայտված կիլոգրամներով, u–ն՝ արագությունը՝ մ/վ–ով, T–ն ջերմաստիճանը՝ Կելվինով, իսկ $k = 1.38\cdot10^{-23}$ Ջ/Կ–ը՝ Բոլցմանի հաստատունը: u արագության ամենահավանական u_p արժեքը համապատասխանում է $f(u)$ ֆունկցիայի առավելագույն արժեքին և որոշվում է $\dfrac{df(u)}{du} = 0$ հավասարումից: $f(u)$–ն հայտարարեք որպես սիմվոլային արտահայտություն, ածանցեք ըստ u արագության և ցույց տվեք, որ $u_p = \sqrt{\dfrac{2kT}{m}}$: Որոշեք u_p արագությունը թթվածնի մոլեկուլի համար ($m = 5.3\cdot10^{-26}$ կգ), եթե $T = 300$ Կ: Այս մոլեկուլի համար կառուցեք $f(u)$ ֆունկցիայի գրաֆիկը $0 \le u \le 2500$ մ/վ տիրույթում:

26. Դիֆուզիայի միաչափ հավասարումն ունի $\dfrac{\partial u}{\partial t} = m\dfrac{\partial^2 u}{\partial x^2}$ տեսքը: Ցույց տվեք, որ հետևյալ ֆունկցիաները դիֆուզիայի հավասարման լուծումներ են.

- $u = A\dfrac{1}{\sqrt{t}}\exp\left(-\dfrac{x^2}{4mT}\right) + B$, որտեղ A–ն և B–ն հաստատուններ են,
- $u = Ae^{-\alpha x}\cos(\alpha x - 2m\alpha^2 t + B) + C$, որտեղ A–ն, B–ն, C–ն և α–ն հաստատուններ են:

27. Փոփոխական լարման միջին քառակուսային արժեքը որոշվում է $u_{rms}=\sqrt{\frac{1}{T}\int_0^T u^2(t)dt}$ բանաձևով, որտեղ T –ն լարման պարբերությունն է:

- Լարումը տրված է $u(t)=u_0\cos(\omega t)$ ներդաշնակ օրենքով, որտեղ $\omega=\frac{2\pi}{T}$ –ն լարման հաճախությունն է: Ցույց տվեք, որ այս դեպքում լարման միջին քառակուսային արժեքը կախված չէ հաճախությունից և $u_{rms}=\frac{u_0}{\sqrt{2}}$:
- Լարումը տրված է $u(t)=3.5\cos(450t)+3$ Վ օրենքով: Որոշեք լարման միջին քառակուսային արժեքը:

ՕԳՏԱԳՈՐԾՎԱԾ ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. Getting Started with MatLab R2015a, *The Mathworks, Inc, 2015*.
2. **И. Ануфриев**, Самоучитель MatLab 5.3/6.x, *Санкт Петербург, «БХВ–Петербург», 2002*.
3. **Patrick Marchand, O. Thomas Holland**, Graphics and GUIs with MatLab, *Chapman & Hall / CRC, 2003*.
4. **Won Young Yang, Wenwu Cao, Tae–Sang Chung, John Morri**s, Applied numerical methods using MatLab, *A John Wiley & Sons, Inc., 2005*.
5. **Amos Gilat**, MatLab. An introduction with applications, *A John Wiley & Sons, Inc., 2011*.
6. **Dingyu Xue, Yangquan Chen**, Solving applied mathematical problems with MatLab, *Chapman & Hall / CRC, 2009*.
7. **H.J. Lee, W.E. Schiesser**, Ordinary and partial differential equation routing in C, C++, Fortran, Java, Maple and MatLab, *Chapman & Hall / CRC, 2004*.
8. **John H. Mathiews, Kurtis D. Fink**, Numerical methods using MatLab, 3rd edition, Prenticle Hall, 1999.
9. **O. Beucher, M. Weeks**, Introduction to MatLab & Simulink. A project approach, *Infinity Science Press LLC, 2006*.
10. **K. Atkinson, W. Han, D. Stewart**, Numerical solution of ordinary differential equations, *A John Wiley & Sons, Inc., 2009.*
11. **S.R. Otto, J. P. Denier**, An introduction to programming and numerical methods in MatLab, *Springer–Verlag London, 2005*.
12. **Robert E. White**, Computational mathematics. Models, methods, and analysis with MatLab and MPI, *Chapman & Hall / CRC, 2004*.
13. **Rao V. Dukkipati**, MatLab. An introduction with applications, *New Age International Ltd., 2010*.
14. **Jamal T. Manassah**, Elementary mathematical and computational tools for electrical and computer engineers using MatLab, *CRC Press LLC, 2001*.
15. **С.П. Иглин**, Математические расчеты на базе MatLab, *Санкт Петербург, «БХВ–Петербург», 2005.*
16. **Գ. Բ. Ալավերդյան, Ա. Ս. Հարությունյան, Յու. Լ. Վարդանյան**, Էլեկտրադինամիկայի խնդիրների ժողովածու 1, *ԴԱՐ հրատարակչություն, 2006.*
17. **Գ. Բ. Ալավերդյան**, Վեկտորական և թենզորական հաշվի խնդրագիրք, *Երևան, 1984.*
18. **Г. Корн, Т. Корн**, Справочник по математике для научных работников и инженеров, Изд. *«Наука», Москва 1968*.

ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՏԱՐՈՆ ՀԱՐՈՒԹՅՈՒՆՅԱՆ

ՌԱԴԻՈՖԻԶԻԿԱԿԱՆ ԽՆԴԻՐՆԵՐԻ ՀԱՄԱԿԱՐԳՉԱՅԻՆ ՄՈԴԵԼԱՎՈՐՈՒՄԸ MATLAB ՄԻՋԱՎԱՅՐՈՒՄ

MATLAB ՄԻՋԱՎԱՅՐԻ ԿԱՌՈՒՑՎԱԾՔԸ

Համակարգչային ձևավորումը՝ Ա. Հարությունյան, Կ. Չալաբյանի
Կազմի ձևավորումը՝ Ա. Պատվականյանի
Հրատ. սրբագրումը՝ Գ. Գրիգորյանի

Տպագրված է «Գևորգ-Հրայր» ՍՊԸ-ում:
ք. Երևան, Գրիգոր Լուսավորչի 6

Չափսը՝ 70x100 1/16: Տպ. մամուլը՝ 15.875:
Տպաքանակը՝ 100:

ԵՊՀ հրատարակչություն
ք. Երևան, 0025, Ալեք Մանուկյան 1