

MATLAB

LEHENENGO URRATSAK

FLORENCIO GARRIDO URIARTE

LEYRE ORMAETXEA BUTRON

*Ene guraso maitatuen omenez,
zuenganako maitasunak betiko
iraungo du. Eskerrik asko guz-
tiagatik, eta, Jainkoak gorde
zaitzatela. ♡*

Aurkibide orokorra

| | |
|---|-----------|
| Irudien aurkibidea | iii |
| Taulen aurkibidea | v |
| 1 Sarrera | 1 |
| 1.1 Zer da Matlab? | 1 |
| 1.2 Nola egiten du behar? | 2 |
| 1.3 Liburuaren antolaketa | 3 |
| 2 Lehenengo urratsak | 5 |
| 2.1 Matlab-eko lan-ingurunea. Leihoak | 5 |
| 2.2 Aginduetarako leihoa | 6 |
| 2.3 Laguntza-motak | 9 |
| 2.4 Aldagaiak | 10 |
| 2.5 Bektoreak eta matrizeak | 12 |
| 2.5.1 Bektoreak eta matrizeak sortzea | 12 |
| 2.5.2 Bektore eta matrizeekin eragiketak egitea | 14 |
| 2.5.3 Bektore eta matrize zatiak atzitzea | 17 |
| 2.5.4 Bektoreen eta matrizeen dimentsioa | 21 |
| 2.5.5 Matrizeen manipulazioa | 22 |
| 2.5.6 Eragiketa gehiago bektore eta matrizeekin | 24 |
| 2.5.7 Gelaxka-bektoreak eta egiturak | 27 |
| 3 Funtzioak eta komando-fitxategiak | 31 |
| 3.1 Funtzioak sortzea | 31 |
| 3.1.1 Funtzioen egitura | 31 |
| 3.1.2 Funtzioak egituratzeko beste era batzuk | 35 |
| 3.2 Funtzioen argumentu aldakorrak | 38 |
| 3.3 Funtzioen argumentuak beste funtzio batzuk izatea | 42 |
| 3.4 Aginduzko fitxategiak edo scripts | 44 |
| 4 Eragileak eta fluxuaren kontrola | 47 |
| 4.1 Erlazio-eragileak eta eragile logikoak | 47 |
| 4.1.1 Erlazio-eragileak | 48 |
| 4.1.2 Eragile logikoak | 50 |
| 4.2 Fluxuaren kontrola | 55 |
| 4.2.1 Erabaki-egiturak | 56 |

| | | |
|----------|--|------------|
| 4.2.2 | Begiztak | 60 |
| 5 | Datuen maneia | 65 |
| 5.1 | Sarrerako eta irteerako datuak | 65 |
| 5.1.1 | Erabiltzaileen sarrerak | 65 |
| 5.1.2 | Irteerak pantailatik | 67 |
| 5.1.3 | Sarrerak eta irteerak fitxategi batean | 71 |
| 5.2 | Datu-inportazioa eta esportazioa | 74 |
| 6 | Bi eta hiru dimentsioko grafikoak | 77 |
| 6.1 | Bi dimentsioko grafikoak | 77 |
| 6.1.1 | Adierazpen grafikoaren formaturako aginduak | 84 |
| 6.1.2 | Irteera grafiko motak | 87 |
| 6.1.3 | fplot komandoa | 91 |
| 6.2 | Hiru dimentsioko grafikoak | 92 |
| 6.2.1 | Lerro-grafikoak | 92 |
| 6.2.2 | Sareta- eta gainazal-grafikoak | 93 |
| 6.2.3 | Grafiko bereziak | 99 |
| 6.3 | Objektu grafikoak, identifikatzaileak eta propietateak | 102 |
| 6.3.1 | Grafikoak inprimatzea | 107 |
| 7 | Polinomioak. Doikuntza eta interpolazioa | 111 |
| 7.1 | Polinomioak | 111 |
| 7.2 | Doikuntza-kurbak | 115 |
| 7.2.1 | Polinomioen bidezko egokitzapen-kurbak | 115 |
| 7.3 | Interpolazioa | 119 |
| 8 | Kalkulu sinbolikoa | 125 |
| 8.1 | Objektu eta adierazpen sinbolikoak | 126 |
| 8.2 | Kalkulu sinbolikoaren aplikazio matematikoak | 133 |
| 8.2.1 | Adierazpen sinbolikoaren irudikapen grafikoak. | 143 |
| 8.2.2 | Maple-ko funtzioen erabilera | 146 |
| | Bibliografia | 149 |

Irudien indizea

| | | |
|-----|--|-----|
| 2.1 | Matlab mahaigaina. | 5 |
| 3.1 | Funtzio baten ohiko egitura. | 32 |
| 5.1 | Excel-eko kalkulu-orri esportatua. | 76 |
| 6.1 | Plot aginduaren aplikazio sinpleak zehaztaileak modifikatuz. | 79 |
| 6.2 | Grafiko bateko propietateen eta balioen aldaketak. | 80 |
| 6.3 | Zenbait grafiko leiho berean. | 81 |
| 6.4 | figure aginduaren erabilera. | 82 |
| 6.5 | Zenbakikuntza-adibide bat subplot aginduarekin. | 82 |
| 6.6 | subplot-en erabilera. | 83 |
| 6.7 | Grafikoen formatuari buruzko etsenplua. | 86 |
| 6.8 | Objektu grafikoen egitura hierarkikoa. | 103 |
| 6.9 | Grafiko batetik abiatuta, objektu grafikoen propietateak aldatzea. | 110 |
| 7.1 | Datuen askotariko doikuntza polinomialak, $n = 1$ etik $n = 6$ ra. | 117 |
| 7.2 | Hasierako datuen interpolazio lineala, <i>spline</i> interpolazioa eta interpolazio kubikoa. | 122 |
| 8.1 | Adierazpen sinbolikoen askotariko 2D eta 3D irudikapen grafikoak. | 145 |

Taulen indizea

| | | |
|-----|---|-----|
| 2.1 | Eragiketa aritmetikoak. | 7 |
| 2.2 | Eragiketen arteko lehentasunak. | 8 |
| 2.3 | Aurretik definituriko aldagaien izenak eta deskribapenak. | 11 |
| 2.4 | Aldagaien ezabatzea eta informazioa. | 11 |
| 2.5 | Aurrez definituriko matrize batzuk. | 14 |
| 2.6 | Slash-en eta backslash-en portaera. | 16 |
| 2.7 | Bektore eta matrizeekiko beste oinarritzko agindu batzuk. | 27 |
| 4.1 | Erlazio-eragileak. | 48 |
| 4.2 | Aurredefinitutako eragile logikoak. | 50 |
| 7.1 | Interpolazio-prozesurako hasierako datuak | 121 |

1. Kapitulu Sarrera

1.1 Zer da Matlab?

Matlab kalkulu zientifikoa egiteko lan-ingurune bat da. Izan ere, software-pakete hau Math Works Inc.-ek komertzializatzen du, eta erabiltzaileari kalkulu boteretsua gauzatzeko tresna osoa eta atsegina ematen ahalegintzen da. Jatorriz, Cleve Molerrek 70eko hamarkadan programatu zuen, bi helburu gogoan zituelarik: batetik, irakaskuntzaren arloan erreminta gisa erabiltzea; bestetik, matrize-analisan LINPACK eta EISPACK bibliotekak atzitzeko era erraza eraikitzea. Kontuan hartu Matlab "Matrix Laboratory"-ren laburdura dela.

Hastapenean Fortran lengoaia erabili bazen ere, denboraren poderioz C lengoia estandarizaturik geratu zen gaurdaino. Egun, mundu osoan zehar, Matlab bai unibertsitate-eremuetan, bai industria-eremuetan ere, arras zabalduta da, bereziki ingeniarietan. *Toolbox-ak* deritzenek areagotu egiten dute software honen potentzia; hain zuzen, *toolboxak* Matlab-en berezko lengoia idatzitako bibliotekak baino ez dira, eta ebatz daitezkeen problema kopurua handiagotzea lortzen dute.

Matlab ingurunean maneiataileak agindu sinpleak askoz konplikatuekin batera nahas ditzake. Esate baterako, badaude berezko funtzioak programazio barik erabil daitezkeenak eta barbarako, adarkatze- edo baldin-orduan-bestela instrukzioekin elkartu daitezkeenak. Aipatutako etorrizko funtzioak lagun, integralak edo ekuazio diferentzialak edo datu esperimentalei egindako doikuntzak ebatz daitezke. Halan eta guztiz, haren aplikazioak are zabalagoak dira, eta erabiltzaileak diseinatutako programak ere egiteko malgutasuna eskaintzen du, eskaini ere.

Programazio-lengoiairekin batekin trebatuta dauden pertsonen Matlab-en aplikatzen den filosofia naturaltasunez bereganatuko dute. Gainerakoentzat, egileen irudikoz, ondoko orrialdeetan aurkituko dituzten informazioa oso interesgarria izango da, bai Matlab bera bai beste programazio-lengoaia bat erabiltzeko beharrean egonez gero.

Lehen aipatu denez, Matlab-ek bere ahalmena hainbat esparrutan, ingeniarietan eta matematikan batik bat, erakusten du:

- zenbakizko aljebra linealean,
- irteera grafikoetan,
- datu-prozesaketetan,

- sistema dinamikoen simulazioan,

besteak beste. Halaber, Fortran edo C lengoien aurrean abantailak egon badaude, bai programen idazkeran, bai zenbakizko bibliotekak eskueran izateko tenorean ere; halatan, ezaugarri aipagarrienen artean:

- goi-mailako lengoaia idazteko erraztasuna ematen du,
- interfaze elkarreragilearen bitartez erraz esperimenta daiteke eta erroreak araztu daitezke,
- goi-mailako grafikoak sortzeko zailtasun handirik eza, bistaratzeko eta itxuraldatzeko erosotasuna eskuragarri,
- beste plataforma anitzetara joateko, M izeneko fitxategien eramangarritasuna dago,
- Internet-en askotariko problemak ebazteko, M fitxategi ugari dago, eskura eta dohain.
- Toolbox-i esker, kalkulu sinbolikoa edo optimizazioa edota deribatu partzialak egiteko aukera paregabea dago, besteak beste.

1.2 Nola egiten du behar?

Matlab-eko programazio-lengoien sintaxia ohiko lengoaiena baino malguxeagoa da. Egia esan, hasierako aldagai-erazagupena ez da beharrezkoa, eta bektoreak edo matrizeak dimentsioak zehaztu gabe sar daitezke, baita sortu ahala berrizendatu edo birdimentsionatu ere. Horregatik guztiagatik programazio eragabeagoa zilegia da, batzuetan kode eranginkorra izan ez arren.

Esan gabe doa, zenbakizko metodoak -kopuru ikaragarrian- agindu simple baten bitartez implementatuak daudenez, Matlab-ek, maiz, kutxa beltz baten moduko erabilera izan deza keela. Erabiltzaileak zerbait itaundu eta segituan ihardesten dio ordenagailuak, eta tartean erabilitako eragiketa motak ez zaizkio axola izaten. Halarik ere, zenbaitetan komeni da baliaturiko metodoak edo haien aukerako argumentuak eta bere esangurak kontrolpean izatea, dudarik ez.

Matlab-en funtzio-tipo bi bereiz daitezke: funtzio konpilatuak (*built in functions*) eta funtzio ez-konpilatuak. Lehenengoetan optimizaturik daude, eta erabiltzailearentzat kodea ikusi ezinekoa da:

- oinarrizko eragiketak +, *, ...
- funtsezko funtzio matematikoak (sin, cos, tan, exp, log, ...)
- aljebra linealeko algoritmo esentzialak (inv, det, lu, qr, ...)
- irteera grafikoak (plot, pcolor, contour, surf, ...)

Funtzio ez-konpilatuak Matlab-en berezko programazio-lengoaian idatzita daude, eta *.m fitxategietan gorde ohi dira; "m" fitxategi-luzapen estandarra da.

1.3 Liburuaren antolaketa

Liburu hau zortzi kapitulutan antolatuta dago, eta bigarrenetik zortziraino Matlab programa erakusten da. Errepaso zabala da eta oinarritzko hasieratik abiatuta, kontzeptu garrantzitsuak jorratzen dira. Berbarako, idazkiak planoko eta espazioko grafikoak hartzen ditu, baita leku bat ere kalkulu sinbolikorako, besteak beste. Kronologiaren aldetik, software-irakaskuntza mailaz mailakoa eta progresiboa da, bigarrenetik aurrera.

Bigarren kapituluan programaren oinarritzko nondik norakoak plasmutzen dira. Horrela, lan-inguruneke leihon nagusien deskribapena egiten da. Bertan lehenengo funtsezko eragiketara aritmetikoak azaltzen dira, baita zenbakizko irteeren formatuak ere. Programa guztietan bezala, hainbat aginduren laguntza nola lortu zehazten da. Matlab-en egituran aldagaiek, bektoreek eta matrizeek paper garrantzizkoa dute. Hori dela eta, datuen gordeiluak direnez, berak sortzeko moduak, funtsezko eragiketak eta propietate garrantzitsuenak aurkezten dira. Hor, aukerak oso zabalak dira, eta elementuz elementuko eragiketak egiteko molde erabilgarria agertzen da, baita elementuak atzitzeko eta ezabatzeko teknikak ere.

Hirugarrenari dagokionez, pauso bat aurrerago doa, eta erabiltzaileak definituriko funtzioak erakusten dira. Bereziki, barneko egitura nola osatu erakusten da, eta gero askotariko posibilitateak deskribatzen dira. Izan ere, fitxategi berean funtzio batzuk ager daitezke, betiere haietako batek funtzio nagusiaren papera harturik. Gainera sarrerako eta irteerako argumentuen kopuruak aldakorak izan daitezke, *varargin*, *varargout*, *nargin* eta *nargout* aginduei esker, eta funtzioen definizioei malgutasuna ematen diete. Baina posibilitate-espektra zabalagoa da, ezen funtzio baten argumentua beste funtzio bat izan daiteke; hor, *handle* kontzeptuaz idatziko da. Kapituluaren amaieran, aginduzko fitxategiak zer diren azaltzen da.

Matlab-en, posible da programa baten fluxu normala aldatzea edo soilik zati bat zenbatnahi bider errepikatzea. Laburbilduz, horri guztiari buruzkoa da laugarren kapitulua. Horretarako tresnak existitzen dira, bereziki baldintzazko aginduak eta iterazio-sententziak. Haien eragile logikoz eta erlazio-eragilez baliatzen dira. Kontuan izan eragileok egiazko eta gezurrezko kontzeptuak erabiltzen dituztela eta, adierazpen matematikoetan agertzeaz gain, programaren fluxuan eragin handia izan dezaketeela. Izan ere, programa bat inplementatzeko, teknika bat baino gehiago enplegatzen dira, eta egikaritze-ordena arrunta formaldatzeko, esaterako, erabaki-egiturak, *if* edo *switch* egiturak nagusiki usatzen dira. Bestetuzetan, komando askoren exekuzio kopurua finkatzen da; berbarako, adierazpen matematiko bat balioesteko. Han begiztak direlakoak agertzen dira; adibidez, *for* eta *while* aginduak portaera horren erakusgaiak dira.

Datuen iturria esperimendu batetik jasokoak izan daitezke, edo adierazpen matematikoetatik eskuratutakoak. Dударik ez, datuen kudeaketa arrazionaler arduratzen diren tresnak, bai inportaziorako bai esportaziorako enplegatzen direnak, garrantzitsuak dira. Batzuetan, datuak fitxategi batean gordetzen dira edo fitxategi batetik irakurtzen dira edo pantailan erakusten dira, besterik gabe, geroko azterketa egiteko. Haietaz guztietaz bosgarren kapitulua arduratzen da. Azkenik, Matlab-ek Excel programa ezagunarekin daukan zerikusia eta harremana erakusten da.

Betidanik ikerlarien interesa datuen irudikapen grafikoak egitea izan da, batik bat, oro har grafikoaren bitartez datuen ondoko analisisa esanguratsuagoa izaten baita. Seigarren

kapituluan adierazpide grafikoen sorkuntzei eta aldakuntzei buruzkoa da. Bertan bi dimentsioko grafikoetatik hasita eta hiru dimentsiokoetan bukatuta, askotariko aginduak erakusten dira hainbat eta hainbat grafiko-mota irudikatze aldera. Hala ere, han ez da kapitulu honen edukia geratzen, eta nola sartu elementuak eta nola aldatu haien propietateak ere azaltzen da. Askotan menuen bitartez efektu bera egin ahal izanagatik ere, aginduen erabilera azaltzea hobetsi da. Bestalde, adierazpideen objektu grafikoez eta propietateez ere mintzaten da, eta haiei esker grafikoaren kontrol osoa eta posibilitate inobreak irakurlearen eskueran ipintzen dira. Haria aldaturik, behin irudikapena osatuta nola esportatu eta disko gogorrean nola gorde erakusten da, formatuak formatu.

Funtzio sinpleenak, askotan portaera zailagoak modelizatzen dituztenak, funtzio polinomikoak dira. `Matlab`-ek funtzio horiek aljebraikoki manipulatzeko zenbait agindu dauzka. Horrela, barbarako, erroak edo eragiketa aljebraikoak edo intergralak etab. ebazten dira, osagai oro polinomioak izanik. Bestalde, zazpigarren kapituluan funtzioen doikuntza eta interpolazio-prozesuak ere aztertzen dira. Lehenengoari dagokionez, doikuntza linealak, esponentzialak eta logaritmikoak proposatzen dira, besteak beste, eta ezlinealtasunaren kasuan nola linealizatu azaltzen da. Halaber, polinomio-doikuntzak gauzatzeko, karratu minimoen teknika erakusten da. Interpolazioan, berriz, egokitzapen-prozesuan ez bezala, osatzen den funtzioa puntu guztietatik igarotzen da, eta dimentsiobakarrekoa eta bi dimentsiokoa gauzatu daiteke metodo desberdinak erabiliz.

Azken kapituluan kalkulu sinbolikoaren gainekoa da. Aurreko kapituluetan ez bezala, honetan era sinboliko baten ikuspuntutik aztertzen dira problema aljebraikoen ebazpenak. Aintzat hartzekoa da adierazpen horietan agertzen diren aldagaiek ez dutela zenbakizko baliorik gordetzen. Kapitulu honetan ezaugarri hori net garrantzitsua da. Areak ere, honaino iritsi bitartean ikusitako oro zenbakizko bektoreak eta matrizeak izango ziren. Oro har, objektu sinbolikoak manipula daitezke, barbarako, faktore komuna ateratzeko edo sinplifikazioak lortzeko. Areago kalkulu sinboliko matematikoarekin ohiko eragiketak gauzatu ahal dira, hala nola integrazio sinplea, anizkoitza, ekuazio diferentzialen ebazpena, Taylor-en garapenak, limiteen soluzioak eta abar. Kalkuluaren arlotik kanpo aljebra hel dakiok, alderantzizko matrizeak edo heinak edo bektore eta balio propioak kalkulatzearren. Azkenekoz, alde batetik, adierazpen sinbolikoen irudikapen grafikoez aritzen da, eta, beste alde batetik, `Maple`-ren eta `Matlab`-en arteko harremana agerian uzten da.

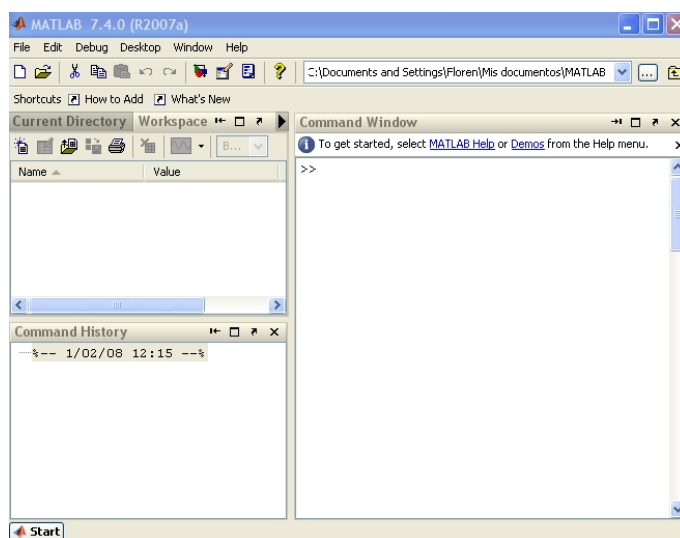
Berriaz aipatu behar da, liburu honetan agertzen diren zenbait informazio, grafiko eta adibide, honako bi liburu hauetatik aterata daudela: bibliografiako [6] eta [9] liburuetatik, hain justu.

2. Kapitulua

Lehenengo urratsak

2.1 Matlab-eko lan-ingurunea. Leihoak

Matlab zabaltzen den lehenengo aldian 2.1 Irudia topatzen dugu, eta bertan honako leiho grafiko hauek aurkitzen dira:



2.1 Irudia: Matlab mahaigaina.

- *Command Window* edo aginduetarako leihoa: Bertan aginduak idatz daitezke eta irteerak ikusi. Izan ere, Matlab-eko leiho nagusia da, eta beti zabalik edukitzea komenigarria da. Azken batean, leiho hori zerbitzu guztietarako abiapuntua da, dudarik ez.
- *Command History* edo aginduen historia: komandoetarako leihoan sartutako aginduak gorde eta bistaratu daitezke. Saioa bukatu bitartean, historian pilatutako sententziak berriro ere exekutatu daitezke, saguarekin *Command Window*-era arrastatuz edo zuzenean sakatuz.
- *Workspace Window* edo lan-esparrua: hari esker, sartutako aldagaiei buruzko informazioa, tamaina, tipoa, balioak, besteak beste, memoria horretan metatu eta gorde-

tzen dira. Azken batean, gogoan izan behar da memoria aldakorra dela eta programa ixtearekin batera desagertzen dela.

- *Current Directory Window* edo laneko uneko direktorioa: uneko direktorioko fitxategiak ikusgai egongo dira, eta bertan dauden programak exekutatu ahal izango dira.

Halarik ere, era laburrean bada ere, hauetaz guztiez aparte, aipatzekoak dira beste leiho interesgarri batzuk: laguntza-leihoa, editore-leihoa, irudi-leihoa. Oso-oso intuitiboak dira, eta, liburua garatu ahala, nola edo hala agertuko dira.

2.2 Aginduetarako leihoa

Lehen aitzinatu denez, `Matlab` programan komando-leihoa leiho nagusia da. Eskuarki, bertan askotariko aginduak exekutatzeko dira, edo beste leiho batzuk ireki edo erabiltzaileak idatzitako programak egikaritu, edota `Matlab`-eko softwarea ere kudeatzen da. Jarraian *Command Windows*-en ezaugarri nagusiak xehatuko dira:

- `<<` ikurra *prompt* deritzona da, eta haren ostean sententziak idazten dira, baita exekutatu ere.
- Hala, zerbait exekutatu asmoa izanez gero, `Enter` sakatu behar da.
- Lerro batean komando bat baino gehiago idatz daiteke, koma bereizlea izanik, eta kontuan hartu exekuzio-ordena ezkerretik eskuinera doala.
- Aurretik idatzitako lerroak edo instrukzio-taldea `↑` eta `↓` geziak bitartez berreskuratu eta berriro aktiba daitezke.
- Sententzia bat luzea baldin bada, eta beste lerro batera pasatu nahi izanez gero, eten-puntuak idatzi, haustura modura: gero, `Enter` sakatu, eta beheko lerroan idatzen jarrai daiteke. Era horretan, exekutzeko dauden lerro guztiak batera eta bloke bat izango balira bezala egikaritzen dira.
- `%` ikurrari esker, iruzkinak tartekatzea lortzen da, eta jakin behar da ez daukala inolako eraginik komandoaren exekuzioan. Aurrean edo amaieran jar daiteke, eta, bereziki programak inplementatzen direnean, oso hedatua da iruzkinak informazio gisa ipintzeko ohitura.

`Matlab` ingurunean lan egiten denetan, aintzat hartzekoa da honako informazio hau:

- `Matlab`-ek maiuskulak eta minuskulak bereizten ditu. Xehetasun hori aldagaien izendapenean hartu behar da kontuan.
- `Matlab`-eko agindu oro minuskulaz idazten dira, eta haien argumentuak parentesien artean eta komaz bereizirik izkiriartzen dira.
- Sententzia baten amaieran puntu eta koma jarritz gero, exekuzioaren irteera ez da pantailatik bistaratzen, egikarituta egon arren. Egindakoaren zantzua *Command Window*-en soilik agertu eta gordeko da.

- Komando-leihoan aginduak tekleatzen diren heinean, pixkanaka-pixkanaka sententziaz betetzen da. Irteera eta sarrera guztiak pantailatik ezabatzeko, alde batetik, `clc` aginduarekin dena begi-bistatik desagertzen da, nahiz eta *Command Window*-eko edo *Workspace*-ko informazioa galdu ez; bestetik, `home` aginduarekin prozesu bera gertatzen da, baina, oraingoan, idatzirikoa ikusgai dago, betiere korritze-barra mugitzen baldin bada.

Estetikaren aldetik, ikusten da Matlab-en interfazea pobre samarra dela, Derive, Octave, Maple edo Mathematica aplikazioetan gertatzen den legez. Usu, oinarritzko funtzionamenduari begira, hasierako erabilera kalkulagailuarena izan daiteke.

```
>> 23+1/3-2/3
ans =
    22.6667
>> 2^1.3-1/(3*2)
ans =
    2.2956
>> ans*24
ans =
    55.0949
```

Ohartu irteera orotan `ans` (*answer*) aldagai lehenetsia itzultzen dela. Beranduago deskribatu arren, azken finean, hark azken irteera gordetzen du, eta gero haren balioa kalkuluak egiteko erabil daiteke.

Eskalarrekin eragiketa aritmetikoak (2.1 Taula) usatzen dira, eta badago horien artean erre-gelak edo lehentasun-ordenak aintzakotzat hartu behar direnak:

| Eragiketa | Ikurra | Adibidea |
|-----------------|--------|-----------------------|
| Batuketa | + | $5 + 3$ |
| Kenketa | - | $5 - 3$ |
| Biderketa | * | $5 * 3$ |
| Eskuin-zatiketa | / | $5/3$ |
| Ezker-zatiketa | \ | $5 \setminus 3 = 3/5$ |
| Berreketa | ^ | $5^3(5^3 = 125)$ |

2.1 Taula: Eragiketa aritmetikoak.

Kontuan izan hemen agertzen diren ikurrak edozein kalkulagailutan aurkitu daitezkeela, ezker-zatiketa salbu. Izan ere, gero ikusiko denez, eragiketa hori gehienbat eskalarrekin erabili beharrean, matrizeekin erabili ohi da. Ordena-exekuzioari dagokionez, arau hauek edozein programazio-lengoiatan ere aplikatzen dira (2.2 Taula):

| Lehenetasuna | Eragiketa matematikoa |
|--------------|--|
| Lehenengoa | Parentesia. Parentesi habiaratuak baldin badaude, barrukoena egikaritze-ko lehenengoa da, eta handik kanporantz. |
| Bigarrena | Berreketa. |
| Hirugarrena | Biderketa eta zatiketa (lehenetasun-maila berean). |
| Laugarrena | Batuketa eta kenketa. |

2.2 Taula: Eragiketen arteko lehenetasunak.

Dena den, bi eragiketak aurrekotasun bera baldin badute, adierazpena ezkerretik eskumara exekutatu da.

2.1 Ariketa: *Konturatu eragiketa hauek desberdinak direla:*

$$6/6-1 \quad 6/(6-1) \quad 2^3*4 \quad 2^(3*4)$$

Programa gehienetan bezala, Matlab-en zenbakiak idazkera zientifikoan sar daitezke. Gainera, zenbaki berezi bi: `inf` eta `NaN`. Lehenengoarekin kantitate infinitua adierazten da, (∞), eta bigarrenarekin "ez da zenbaki bat" (*Not a Number*)-en laburdura da; esaterako, eragiketa ez-definitu bat (0/0 kasu) egiten denean.

```
>> 0/0
ans =
      NaN
>> inf*0
ans =
      NaN
```

Oro har, zenbakien errepresentazioan lau zenbaki zehatzak agertzen dira, baina jakinaren gainean egon eragiketa oro doitasun bikoitzetan exekututzen direla. Matlab-eko hainbat parametrotan bezala, hori ere `format` aginduari esker berrerratu daiteke, segidako instrukzioek erakusten duten legez:

```
>> format short
>> exp(1) % e zenbakia
ans =
    2.7183
>> format long % doitasun maximoa, luzeena
>> exp(1)
ans =
    2.718281828459046
>> format rat % zatikien erara
>> exp(1)
ans =
    1457/536
```

```
>> format bank % zenbaki hamartarren kopuru finkatua
>> exp(1)
ans =
    2.72
```

2.3 Laguntza-motak

Matlab-eko agindu eta *demo* askori buruzko informazio zabalerako sarbidea, leiho nagusiko *Laguntza* menutik edo ”?” ikonotik lortu ahal da. Zer esanik ez, aski intuitiboa eta osoa da; alabaina, bilaketa errazte aldera, bertan arakatzeko-leiho bat dago, baita edukia eta aurkibidea ere. Horrezaz gain, *prompt*-etik *help* idatziz gero, ikus daiteke nola dauden eraturik laguntzarako direktorio-zerrenda eta Matlab-eko funtzioak.

```
>> help
HELP topics:
```

| | |
|-------------------|--|
| matlab\general | - General purpose commands. |
| matlab\ops | - Operators and special characters. |
| matlab\lang | - Programming language constructs. |
| matlab\elmat | - Elementary matrices and matrix manipulation. |
| matlab\elfun | - Elementary math functions. |
| matlab\specfun | - Specialized math functions. |
| matlab\matfun | - Matrix functions - numerical linear algebra. |
| matlab\datafun | - Data analysis and Fourier transforms. |
| matlab\polyfun | - Interpolation and polynomials. |
| matlab\funfun | - Function functions and ODE solvers. |
| matlab\sparfun | - Sparse matrices. |
| matlab\scribe | - Annotation and Plot Editing. |
| matlab\graph2d | - Two dimensional graphs. |
| matlab\graph3d | - Three dimensional graphs. |
| matlab\specgraph | - Specialized graphs. |
| matlab\graphics | - Handle Graphics. |
| matlab\uitools | - Graphical User Interface Tools. |
| matlab\strfun | - Character strings. |
| matlab\imagesci | - Image and scientific data input/output. |
| matlab\iofun | - File input and output. |
| matlab\audiovideo | - Audio and Video support. |
| matlab\timefun | - Time and dates. |
| matlab\datatypes | - Data types and structures. |
| matlab\verctrl | - Version control. |
| matlab\codetools | - Commands for creating and debugging code. |
| matlab\helptools | - Help commands. |
| matlab\winfun | - Windows Operating System Interface Files (COM/DDE) |
| matlab\demos | - Examples and demonstrations. |
| matlab\timeseries | - Time series data visualization and exploration. |

```
matlab\hds          - (No table of contents file)
matlab\guide        - Graphical User Interface Tools.
matlab\plottools    - Graphical User Interface Tools.
```

Exekuzioaren ostean eta zerrenda bistaraturik, berbarako, aukeratu izen bat, `help` direktorioaren izena tekleatu eta sakatu. Segituan, haren barruan definiturik dauden funtzioen deskribapen laburrak agertuko dira. Behinik behin, agindu bakoitzari buruzko azaleko ideia izateko lagungarri izan daiteke. Zer esanik ez, era berean *Komando-leihotik* laguntza zehatza eska daiteke arrapaladan.

2.2 Ariketa: *Erkatu* `help sqrt`, `helpwin sqrt` eta `doc sqrt` eta kontuan hartu desberdintasunak, baldin badaude.

Azkenik, egoera batzuetan `lookfor` agindua ere oso erabilgarria izan daiteke, zeren exekuzioaren irteeretan hitzarekin zerikusia duten funtzioen izenak topatzen baitira. Egia esan, geroago deskribatuko denez gero, normalean ia-ia funtzio guztien goiburuetan zenbait iruzkin badaude; bertan, `lookfor`-en bilaketak hitzarekin parekotasunak aurkitzen ahalegiten dira.

2.3 Ariketa: *Exekutatu* `bessel` eta `matrix` eta *erreparatu* irteera-kopuruari.

2.4 Aldagaiak

Aldagai bat hitz batzuez osatutako izen bat besterik ez da, zenbakizko balioak edo testuak, esaterako, esleitu zaizkiona. Behin balioren bat gorderik, eragiketak egiteko edo funtzioetan erabiltzeko balio du, edo beste komando batean sar daiteke. Hastapenetan aldagai guztiak errealak dira, eta haien erabilera aski da deklaratuak izan daitezen. Hau da, aldezturik ez dago zertan deklaratu. Egia da aldagaietan testua sar daitekeela, baina portaera berbera mantentzen da.

```
>> a=sin(pi/12);b=cos(2.);c=2;
>>format long, log(a)*abs(b)-c^2+4
ans =
    -0.562474937787112
```

Halere, Matlab-en berezitasun horrek eragozpenak izan ditzake. Gerta liteke aldagai baten idazkeran akats bat izatea, eta errore-mezurik egon ezik, nahi gabe, exekuzio-errorea izatea. Dena den, aldagai baten izendapenean honako erregela hauek errespetatu behar dira:

- Aldagai batek gehienez 63 karaktere izan ditzake.
- Lehenengo karaktereak letra bat izan behar du, gero zenbakiak ere agertu ahal izanagatik ere.
- Letra maiuskulak eta minuskulak bereizten dira. Beraz, ez da gauza bera AA, Aa, aA edo aa idaztea.

- Ahal izanez gero, programako funtzioen edo aurrez finkatutako aldagaien izenak ez esleitu. Pasadizoan 2.3 Taulan erreserbaturiko zenbait izen deskribatzen dira:

ans Lehenespenez, ans aldagaien adierazpen baten balioa gordetzen da.
 pi π zenbakia da.
 eps Bi zenbakien arteko kendura maximoa, 2^{-52} ordenagailuaren arabera.
 inf ∞ ikurra da.
 i -1 en erro karratua, hau da: $0+1.000i$
 j i -ren kidekoa da, konplexuen unitate irudikaria alegia.
 NaN "Not a number"-ren laburdura da, eta Matlab-ek eragiketa baten balioa ezagutzen ez duenean azaltzen da.

2.3 Taula: Aurretik definituriko aldagaien izenak eta deskribapenak.

Aldagaiak osatu ahala, interesgarria izan daiteke edozein unetan haiei buruzko informazioa jakitea edo ezabatzea. Segidako aginduak, bai aldagaiak ezabatzeko, bai definitutako aldagai bati buruzko informazioa lortzeko baliak daitezke. Komandoak idatzi eta Enter tekleaturik, 2.4 Taula-k bildutako hurrengo atazak exekutatzeko dira:

| Agindua | Emaitza |
|---------------|---|
| clear | Memoriako aldagai oro ezabatzen dira, <i>Workspace</i> -n gordeak. |
| clear a b c d | Soil-soilik memoriako a, b, c eta d aldagaiak ezabatzen dira. |
| who | <i>Workspace</i> -n gordetako aldagai-zerrenda bistaratzen da. |
| whos | who-ren antzekoa, baina gehi tamaina, gehi aldagai mota eta luzera. |

2.4 Taula: Aldagaien ezabatzea eta informazioa.

Interakziozko moduan, *Workspace*-ko edozein A aldagairen gainean klik eginez gero, haren balioak erakuts daitezke. Gisa berean, beste aukera bat `openvar('A')` agindua exekutatzeko da, eta berriro ere, Excel-en moduko editore batean agertzen dira balioak.

```
>> A=1:3
A =
     1     2     3
>> openvar('A')
```

Askotan, lan-saio baten ostean, aldagaien balioak beste saio baterako gorde nahi izaten dira. Kasu horretan `save` agindua erabili behar da, eta honako sintaxi orokor hau dauka:

```
save fitxategiaren izena ald1, ald2, ald3, ....
```

Fitxategiaren luzapena `mat` da, eta barruan zenbatgura aldagai gorde daitezke. Geroxeago, nahi izanez gero, Matlab itxi daiteke, eta, pizten den hurrengo aldian gordetako aldagaiak berriro ere *Workspace*-n kargatzeko, `load fitxategiaren izena` besterik ez da tekleatu behar. Adibidez, exekutatu sententzia hauek:

```
>> A=1:3;
>> B=2*A;
>> C=A-B;
>> save fitxa A B C % konturatu fitxa-ren luzapenaz
>> clear
>> load fitxa
```

2.5 Bektoreak eta matrizeak

Hasteko, $m \times n$ matrize bat egitura matematiko bat da, zenbakiz beteriko m errenkadaz eta n zutabez osatua. Kasu partikularrak $m = 1$ edo $n = 1$ direnean, errenkada-bektoreak eta zutabe-bektoreak agertzen dira, hurrenez hurren. Matlab-en matrizeak funtsezkoak dira, aljebra linealeko problemetarik at aritu arren, datu askoren metagailuak direla; horregatik, etengabe erabiltzen dira.

2.5.1 Bektoreak eta matrizeak sortzea

Bektoreak eta matrizeak Matlab-eko objektu garrantzitsuenetarikoak dira; haiek eratzeko, askotariko moldeak daude, baita, zergatik ez, trukoak ere. Dena den, erne izan, zeren, laster ikusiko denez, jadanik definituriko matrizeen elementuak berreskuratzeko eta haien posizioak zehazteko, beste lengoaietan ez bezala, zenbaki arruntak erabiltzen baitira, besterik ez.

Eskuarki matrize bat definitzen denean, elementuak kortxete artean idazten dira. Areago, errenkadaz aldatzeko ”;“ bereizlea erabiltzen da, eta errenkada batean zutabeko elementuak bereizteko, hutsuneak edo komak usatzen dira.

```
>> A=[1 2 3; 4 5 6; 7 8 9]; % hiru ordenako matrize erreala
>> B=[-1,-2,-3]; % errenkada-bektorea
>> C=[3; 2; 1]; % zutabe-bektorea
```

Batetik, Matlab-ek errenkada- eta zutabe-bektoreak desberdintzen ditu; batik bat, baturak edo biderkadurak egiteko unean kontuan hartzekoa da ezaugarri hori. Bestetik, errenkada batean zutabeko elementuen arteko hutsunea aldagarria izan daiteke, baina matrize bera adierazten da. Orobat, eta ikuspuntu praktikoari begira, matrizearen dimentsioa ezagutzen baldin bada, baina ez haren elementuak, esaterako, geroxeago sartuko baitira, komenigarria gerta daiteke era honetan hastea:

```
<< D=zeros(20,10)
```

Hala memorian posizio kopuru zehaztua erreserbatzen da, eta gero datuak gordetzea eta atzitzea azkarragoa izango da. Edozein unetan, kasu honetan bezala, zeroz osatutako matrizea izan arren, parentesiei eta posizioari esker, haren elementuak alda daitezke.

```
D(1,8)=3; D(2,3)=4; D(12,9)=-1;
```

Zernahi gisaz, Matlab-en definitu gabeko matrize baten posizioari balio bat esleitzen bazaio, bi efektu nabaritzen dira; bateko, Matlab-en iritziz, elementuak sartu ahala matrizearen tamaina egokituz doala, eta besteko, oraino definitzeke dauden elementuak zero bihurtzen dituela.

```
<< clear all % Workspace Window-eko balio oro desagertzen dira.
<< A(3,2)=-1 % Orain A-ren dimentsioa 3x2 da.
```

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

Maiz ezaugarri horrek programazio malgua eta lasaia egiten uzten badu ere, neurritz erabiltzekoa, da ezen hautemateko zailak diren akatsak eragin baititzake.

Matrizeak sortzeko unean, jakin beharrekoa da matrizeak blokeka ere sor daitezkeela. Matrize habiaratuak dira horren adibideak. Ikusi segidako bloke-matrizea; 2.5 Taulako zenbait matrize erabiltzen ditu:

```
>> A=[1,2 ;3 4]
A =
     1     2
     3     4
>> B=[A,zeros(2) ; ones(2) eye(2)]
B =
     1     2     0     0
     3     4     0     0
     1     1     1     0
     1     1     0     1
```

Blokekako matrize diagonalak osatzeko beharra arras arrunta da eta orduan soluzio sinplea blkdiag komandoa erabiltzen da:

```
>> C=blkdiag(A,ones(2)) % diagonal nagusiko elementuak matrizeak dira
C =
     1     2     0     0
     3     4     0     0
     0     0     1     1
     0     0     1     1
```

Zenbaitetan, matrize batean blokeak errepikatuak izaten dira. Egitura berezi hori osatzeko, repmat(A,m,n) egitura erabil daiteke, $m \times n$ blokeetan A-ren kopiak agertzen dira:

```
> A=[1 2; 0 0]
A =
     1     2
     0     0
>> repmat(A,2,3)
```

```
ans =
     1     2     1     2     1     2
     0     0     0     0     0     0
     1     2     1     2     1     2
     0     0     0     0     0     0
```

Azkenik azpisekzioa bukatze aldera, egitura partikularrak eta ezagunak dituzten matrize batzuen adibideak 2.5 Taulan zerrendatzen dira:

| Matrizea | Adierazpen matematikoa |
|-------------|--|
| eye(n) | $\text{eye}(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ |
| ones(m,n) | $\text{ones}(2,3) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ |
| zeros(m,n) | $\text{zeros}(2,3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ |
| hilb(n) | $\text{hilb}(2) = \begin{pmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{pmatrix}$; $\text{hilb}(i,j) = \frac{1}{i+j-1}$ |
| magic(n) | $\text{magic}(3) = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ errenkadak batu edo zutabeak batu emaitza bera lortzen da. |
| hadamard(n) | $\text{hadamard}(2) = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ zutabeak ortogonalak dira. |

2.5 Taula: Aurrez definituriko matrize batzuk.

2.5.2 Bektore eta matrizeekin eragiketak egitea

Matlab-en matrizeen arteko ohiko eragiketak inplementatuta daude, hala nola, batuketak, biderketak, berreketak eta determinanteak, besteak beste.

```
>> A1=[1 -2 3; 0 -1 1; 3 2 -2];
>> A2=[-2 1 3; 1 -1 0; -3 -2 2];
>> A1+A2
ans =
    -1    -1     6
```

```

    1    -2    1
    0     0    0
>> A1*A2
ans =
   -13    -3     9
    -4    -1     2
     2     5     5

```

Betiko biderketaz aparte, kron komandoa erabiliz, bi matrizeren arteko Kronecker-en biderketa egin daiteke. Biderketa horretan, A matrizea $m \times n$ baldin bada, eta B matrizea $p \times q$ bada, emaitza $mp \times nq$ dimentsiokoa da, eta biderketaren bloke bakoitza $a_{ij}B$ da. Erreparatu adibide honi:

```

>> A=[1 2; 3 4];
>> B=[-4 -3];
>> kron(A,B)
ans =
   -4    -3    -8    -6
  -12   -9   -16   -12

```

Berezitasunak berezitasun, berreketak-eta ere egin daitezke:

```

>> A1^2-A2^3
ans =
    11     1     7
     4    14     0
   -17   -25    31
>> det(A1)^2 % A1-en determinantearen karratua
ans =
     9

```

Bi bektoreren arteko biderketa eskalarra eta bektoriala ere egin daiteke dot eta cross agin- duei esker, hurrenez hurren:

```

> x=[1 3 4];y=[-2 1 0.5];
>> dot(x,y) % biderkadura eskalarra
ans =
     3
>> cross(x,y) % biderkadura bektoriala
ans =
   -2.5000   -8.5000    7.0000

```

Honaino iritsitakoan, badago berezitasun bat aipatu behar dena: Matlab-en bektore-programazioa egin daiteke, hau da, matrizeen arteko betiko eragiketak gauzatu beharrean, elementuz elementuko eragiketak aplika daitezke, eta, horretarako, delako eragiketaren aitzinean". " ikurra ipintzen da. Aurrera begira, bektore-programazioa oso lagungarria izango da programen idazkerak laburtzeko. Hori guztia hobeto ulertze aldera, begiratu adibide honi, -A1 eta A2 matrizeak lehen definituak dira-:

```

>> A1.*A2 % elementuz elementuko biderketa
ans =
    -2    -2     9
     0     1     0
    -9    -4    -4
>> A1./A2 % elementuz elementuko zatiketa
ans =
   -0.5000   -2.0000    1.0000
           0    1.0000     Inf
   -1.0000   -1.0000   -1.0000
>> A1.^2 % elementuz elementuko berreketa
ans =
     1     4     9
     0     1     1
     9     4     4
>> A1.^A2
ans =
    1.0000   -2.0000   27.0000
           0   -1.0000    1.0000
    0.0370    0.2500    4.0000

```

Igarri-igarrian dago, ez dela gauza bera, esate baterako, $A1 * A2$ biderketa arrunta eta $A1 . * A2$ emaitza. Hain zuzen, aurrekoa ez bezala, azken horretan elementuz elementuko biderketa exekutatu da. Hori bektore-programazioaren etsenplu argia da.

2.4 Ariketa: Sartu M errenkada-bektore bat eta exekutatu: $M.^3$, M' eta $M+1$. Irteeren aitzinean zer da ' ikurraren papera?. Hirugarren eragiketak zentzurik al du?. Ikertu ' eta . ' ikurrak eta deduzitu ea berdin jokatzen duten ala ez.

Ikusitakoez aparte, beste agindu garrantzitsuak hauek dira:

inv / \ /. \.

Lehenengoa lagun, matrize baten alderantzikoa kalkulatzen da. Bestalde, "/" eta "\" (slash eta backslash) 2.6 Taulan deskribatzen direnak, benetan bereziak dira, portaeraren aldetik desberdinak baitira:

| Notazioa | Esangura matematikoa |
|----------------------------------|----------------------------|
| Eskuin-zatiketa: a/b | $\frac{a}{b} = a * b^{-1}$ |
| Ezker-zatiketa: $a \backslash b$ | $\frac{b}{a} = a^{-1} * b$ |

2.6 Taula: Slash-en eta backslash-en portaera.

Taularen arabera, puntua erabiliko balitz, eragiketak elementuz elementu egingo lirateke; hori bai, nahi eta nahi ez, dimentsio bereko bektore edo matrizeak izan beharko lukete.

2.5.3 Bektore eta matrize zatiak atzitzea

Matrizeen eta bektoreen pusketak edo blokeak atzitzea sententzia sinpleen bidez egiten da. Hasieran, litekeena da arraro samar izatea, baina praktikatzea aski da sintaxi berria bereganatzeko. Izan ere, hemen : ikurrak erantzuna dauka, eta horri esker, bloke desberdinak osa daitezke. Lehendabizi, ikusi etsenplu hau:

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> A(3,2) % (3,2) elementua atera
ans=
     5
>> A(2,:) % A-ren bigarren errenkada atera
ans=
     4     5     6
>> A(:,3) % A-ren hirugarren zutabea atera
ans=
     3
     6
     9
```

Orobat, zutabe edo errenkada batzuen zatiak atzi daitezke ”:“ edo kortxeteak erabiliz, aukeran:

```
>> A(3,1:2) % 3. errenkadako 1. eta 2. zutabeak atera
ans =
     7     8
>> A(3,[1 3]) % 3. errenkadako 1. eta 3. zutabeak atera
ans =
     7     9
>> A(:,2:3) % errenkada guztietatik 2. eta 3. zutabeak atera
ans =
     2     3
     5     6
     8     9
>> A(:,[2 1]) % errenkada guztietatik 2. eta 1. zutabeak
      (ordena horretan) atera
ans =
     2     1
     5     4
     8     7
```

Bai bektore batean edo matrize batean azken elementua zehazteko, end aginduaz balia daiteke. Edonola ere, horren bitartez, dimentsioak dimentsio, beti-beti objektuaren azken errenkada edo zutabea seinalatzen da.

```
>> C=blkdiag(A,ones(2))
```

```
C =
     1     2     0     0
     3     4     0     0
     0     0     1     1
     0     0     1     1
```

```
>> C(3:end,3:end)
```

```
ans =
     1     1
     1     1
```

```
>> C(end,:)
```

```
ans =
     0     0     1     1
```

Bestalde, bektoreak eta matrizeak sortzeko ”:“ ikurra ere erabil daiteke, betiere honako egiturari jarraikiz: *hasiera:urratsa:amaiera* non elementuen kopurua

$$\frac{\textit{amaiera} - \textit{hasiera}}{\textit{urratsa}} + 1$$

den. Aintzakotzat hartu, urratsa unitatea baldin bada, normalean adierazpenean ez dela idazten, eta bakarrik hasiera eta amaiera ipini ohi direla.

```
>> B=3:2:9
```

```
B =
     3     5     7     9 % hirutik bederatzira binan-binan B matrizeko
                        % elementuak sortu
```

```
>> C=[1:5;-2:2:6]
```

```
C =
     1     2     3     4     5 % 1. errenkada batetik bostera (urratsa=1)
    -2     0     2     4     6 % 2. errenkadan -2tik 6ra eta (urratsa=2)
```

Askotan, asmoa izaten da $[a, b]$ tartea lerroarte berdineko partiketa bat egitea. Kasu horretan, Matlab-ek `linspace` funtzioa dauka, eta orduan erraza da lerroarte berdineko n puntu sortzea:

```
>> linspace(1,2,9)
```

```
ans =
     1.0000     1.1250     1.2500     1.3750     1.5000     1.6250     1.7500
     1.8750     2.0000
```

Hau da, $[1, 2]$ tartea luzera bereko bederatzita minus bat zatitan deskonposatu egin da eta bederatzita puntu itzultzen dira. Oro har $[a, b]$ tartea baldin bada, urratsaren kalkuluari dagokionez, hau da prozedura:

$$\text{linspace}(a, b, n) \rightarrow \text{urratsa} = \frac{b - a}{n - 1}$$

Bestela, kortxeteak lagun, A matrizeko errenkada eta zutabe espezifikoak atera daitezke:

```
>> A([1 3],[1 2]) % 1. eta 3. errenkadetan eta 1. eta 2. zutabeetan
      % dauden elementuak
```

```
ans =
     1     2
     7     8
```

Software honen bitartez bektore edo matrize bateko elementuetan askotariko permutazioak egin daitezke. Eskuarki, p zenbaki arruntez beteriko bektorea baldin bada, v edozein bektore izanda ere, $v(p)$ exekutuz bada $[v(p(1)), v(p(2)), \dots, v(p(n))]$ itzultzen da. Kasurako:

```
>> v=[2:-2:-6]
v =
     2     0    -2    -4    -6
>> p=[3 1 5]
p =
     3     1     5
>> v(p)
ans =
    -2     2    -6
```

Aplikazio hori erabilgarria izan daiteke bektore edo matrizeetan elementuak lekuz permutatzeko.

2.5 Ariketa: Exekutatu beheko sententziak

```
>> N=[0 2 -3 4; 2 -1 0 3; -1 8 3 7; 2 1 0 -4];
>> p=[2 4 3]; q=[4 2 4];
>> N(p,q)
```

Zehazki zer egiten dute?

Berebat, nabarmentzekoa da ”:“ ikurraren erabilera zeren, zein ataletan idazten den, eragin desberdina du. Horrela, lehen definituriko A matrizea abiapuntu harturik:

```
>> B=A(:)
B =
     1
     4
     7
     2
     5
     8
     3
     6
     9
```

Portaera horrek aipamen berezi bat merezi du. Orain B bektorea zutabe-bektorea da, eta haren elementuak A matrizeko zutabeak dira lehenengotik hasita azkeneraino. Oстера,

ezker-atalean ”:“ idazten baldin bada, portaera desberdina da. Imajinatu A matrizearen dimentsioa ezagutzen dela, eta dagoeneko definiturik dagoela; esaterako, aurreko A matrizea da. Kasu honetan, ”:“ ezkerrean izkiriaturik, zutabez zutabeko balioak birdefinitzeko balia daiteke:

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

Ohartu nola gordetzen diren balioak, zutabeka berriro ere. Ikusitakoa laburbilduz, ”:“ eskuinean ipinita, matrizeak edo bektoreak zutabe-bektore bihurtzen dira; ezkerrean, ordea, balioak zutabeka antolatzen dira.

Matrizeen idazkeran ”[]“ kortxeteek zenbait aplikazio izan ditzakete, bitxiak gainera. Aurrena, matrize bati errenkada edo zutabe bat erants dakioke. Kasu baterako, jo lehen definituriko A matrizea:

```
>> A=[A,[0; 0; 0]] % A-n gaineratu 4. zutabea
A =
    -1    -2    -3     0
    -4    -5    -6     0
    -7    -8    -9     0
>> A=[A; [1 1 1 1]] % A-n gaineratu 4. errenkada
A =
    -1    -2    -3     0
    -4    -5    -6     0
    -7    -8    -9     0
     1     1     1     1
```

Hots, A matrizeari zutabe berri bat itsasten zaio, eta gero errenkada berri bat. Baina kortxeteak bektore edo matrizeetan elementuak gaineratzeaz gain, ezabatzeko ere balia daitezke; berbarako, bigarren errenkada eta laugarren zutabea desagerrarazteko, hain justu:

```
>> A(2,:)=[]
A =
    -1    -2    -3     0
    -7    -8    -9     0
     1     1     1     1
>> A(:,4)=[]
A =
    -1    -2    -3
    -7    -8    -9
     1     1     1
```

Izan ere, ”[]“ notazioarekin 0x0 dimentsioko matrizea definitzen da, eta, adibidean garbi ikusten denez, batik bat matrize bateko errenkadak edo zutabeak kentzeko erabiltzen dira.

2.5.4 Bektoreen eta matrizeen dimentsioa

Matlab-ko bektoreen edo matrizeen dimentsioak eta luzerak `size` eta `length` komandoen bitartez kalkulatu dira. Lehenengoari dagokionez, `size` aginduaren irteerak 1×2 tamainako bektorea itzultzen du: lehenengo osagaia bektoreko edo matrizeko errenkada kopurua da, eta bigarrena zutabe kopurua. Bigarrenari dagokionez, `length` aginduak bektore edo matrize baten luzera itzultzen du. Bektorea baldin bada, esanahia garbi dago; matrizeekin, aldiz, errenkada kopuruaren eta zutabe kopuruaren arteko maximoa hartzen da kontuan. Hori guztia argitzearren, jabetu adibide hauei:

```
>> F=[1:3;2:2:6];
>> size(F)
ans =
     2     3
>> length(F)
ans =
     3
>> F=F'; % F iraultzen da
>> size(F)
ans =
     3     2
>> [e,z]=size(F) % e errenkada kopurua eta z zutabe kopurua
e =
     3
z =
     2
>> length(F) % 3 eta 2 zenbakien maximoa
ans =
     3
```

Behin matrize bat definiturik, badago birdimentsionatzeko era simple bat, noizik behin baliagarria izan daitekeena, eta `reshape` aginduaz baliatzen dena:

```
>> F
F =
     1     2
     2     4
     3     6
>> reshape(F,2,3) % 2x3 dimentsioko matrize berri bat
ans =
     1     3     4
     2     2     6
>> reshape(F,1,6) % 1x6 dimentsioko errenkada-bektorea
ans =
     1     2     3     2     4     6
```

```
>> reshape(F,6,1) % 6x1 dimentsioko zutabe-bektorea
ans =
     1
     2
     3
     2
     4
     6
```

Ohar zaitetz: batetik, elementuen kokapena zutabeka egiten da, eta, bestetik, F jatorrizko matrizearen elementu kopuruak eta reshape funtzioarekin osatzen den matrize berriaren elementu kopuruak bat etorri behar dute.

Bestalde, elementu kopuruaren berri izateko era erosoena numel agindua eskueran dago.

```
>> A=zeros(100,12);
>> numel(A)
ans =
    1200
```

2.5.5 Matrizeen manipulazioa

Noiz edo noiz, matrize bateko zona jakin batzuk, ez soilik azpimatriziak -lehen erakutsi denez-, manipulatu nahi izaten dira. Helburu horrekin, jadanik existitzen diren funtzio askoren artean, honako komando baliagarri hauek daude:

```
diag blkdiag triu tril fliplr flipud rot90
```

Lehenengo diag aginduari dagokionez, askotariko erabilerak eman dakizkioke. Eman dezagun x bektorea dugula; orduan, diag(x) aginduarekin x bektoreko elementuak diagonal nagusian dituen matrize diagonalak osatzen da:

```
>> x=[1 3 -2];
>> diag(x)
ans =
     1     0     0
     0     3     0
     0     0    -2
```

Baina agindua askozaz orokorragoa da, ezen diag(x,k) aginduak, $k > 0$ izanik, diagonal nagusiaren gaineko k. diagonalean x bektorea ipiniko du, eta, aldiz, $k < 0$ baldin bada, diagonal nagusiaren azpiko k. diagonalean:

```
>> diag(x,2)                                >> diag(x,-2)
ans =                                         ans =
     0     0     1     0     0                0     0     0     0     0
     0     0     0     3     0                0     0     0     0     0
```

```

0      0      0      0      -2      1      0      0      0      0
0      0      0      0      0      0      3      0      0      0
0      0      0      0      0      0      0      -2     0      0

```

Beste aplikazio batean, x aldagaia matrizea baldin bada, `diag(x)` aginduarekin, zutabe-bektore gisan, diagonal nagusia ateratzea lortzen da. Halaber, `diag(diag(x))` tekletuz gero, matrize diagonal bat sortzen da, eta x matrizeko diagonalak agertzen da:

```
>> x=[1 -2 3; 5 -3 2; 2 -1 0];
>> diag(x)
```

```
ans =
```

```
1
-3
0
```

```
>> diag(diag(x))
```

```
ans =
```

```
1      0      0
0     -3      0
0      0      0
```

Bestela ere, matrize bateko zati triangeluarrak ateratzeko, `tril` eta `triu` komandoak daude. Lehenengoaren idazkera osoa `tril(A,k)` da. Hari esker, irteera matrize behe-triangeluarra da, A matrizeko diagonal nagusiko k . diagonalatik beherako elementuekin. Era berean, funtzionamenduaren aldetik, `triu` aginduarekin kontu analogoa gertatzen da, baina k . diagonalatik gorako elementuekin:

```
A =
```

```
1      2      3
-1     -2     -3
5      6      7
```

```
>> tril(A,0)
```

```
ans =
```

```
1      0      0
-1     -2      0
5      6      7
```

```
>> triu(A,0)
```

```
ans =
```

```
1      2      3
0     -2     -3
0      0      7
```

```
>> tril(A,-1)
```

```
ans =
```

```
0      0      0
-1     0      0
5      6      0
```

```
>> triu(A,1)
```

```
ans =
```

```
0      2      3
0      0     -3
0      0      0
```

Segidan, `blkdiag` komandoa lagun, bloke-matrizeak nola eraiki erakusten da. Idazkera `blkdiag(A1,A2, ...)` da; $A1, A2, \dots$ matrizeak matrize berriko diagonal nagusian kateatzen dira, eta blokekako matrizea agertzen da:

```
>> A=[1 2;3 4]; B=[-1 -2 ;-3 -4];
>> blkdiag(A,B)
ans =
     1     2     0     0
     3     4     0     0
     0     0    -1    -2
     0     0    -3    -4
```

Halaber, baliteke `fliplr` eta `flipud` aginduekin matrize bateko errenkadak edo zutabeak norabide baten arabera aldatzea. Lehenengoak, `fliplr` aginduak, zutabeak ezkerretik eskuinera mugitzen ditu, eta bigarrenak, `flipud` aginduak, errenkadak goitik behera mugitzen ditu.

Biraketa bat jarraituz, elementuen kokapena aldatzeko, `rot90` aginduak matrize bateko elementuetan erloju-orratzen noranzkoaren kontrako 90 graduko mugimendua eragiten du:

```
A =
     1     2     3
     4     5     6
     7     8     9
>> fliplr(A)                                >> flipud(A)
ans =                                         ans =
     3     2     1                           7     8     9
     6     5     4                           4     5     6
     9     8     7                           1     2     3
>> rot90(A)
ans =
     3     6     9
     2     5     8
     1     4     7
```

2.5.6 Eragiketa gehiago bektore eta matrizeekin

Orain arte ikusitako eragiketez eta komandoez gain, MatLab-ek baditu eragiketa eta funtzio gehiago inplementatuak. Esate baterako, kalkuluarekin eta aljebra linealarekin zerikusia dutenak eta erakutsiko direnak oso eragiketa arruntak dira; haien artean, hauek dira nabarmentzekoak:

- `sum`: Objektua bektorea baldin bada, bektoreko elementuak batzen ditu. Matrizeen kasuan, osteraz, lehenespenez, batura zutabez zutabe gauzatzen da. Horratik ere, nola ez, errenkadaz errenkada ere egin daiteke:

```
>> x=[-2 2 1 -1];A=[1 2 3; 4 5 6; 7 8 9];
>> sum(x)
ans =
     0
>> sum(A)    % zutabetik batura
```

```
ans =
    12    15    18
>> sum(A,2) % errenkadatik batura
ans =
     6
    15
    24
```

2.6 Ariketa: *Nola batuko lirateke matrize baten elementu guztiak?. Nola aurkitu elementu maximoak, minimoak eta haien posizioak?.*

- **prod:** sum agindua bezalakoa da, baina biderkadurak lortzen dira.
- **max:** Objektua bektorea baldin bada, bektoreko elementu maximoa kalkulatzen du. Matrizeen kasuan, berriz, lehenespenez, zutabez zutabe maximoa itzultzen da. Orobat, bai bektoreetan, bai matrizeetan, elementu maximo horien posizioak eskura daitezke. Berriro ere, nola ez, matrizeetan ere errenkadaz errenkada maximoen kalkulua eskura daiteke:

```
>> x
x =
    -2     2     1    -1
>> max(x)
ans =
     2
>> [m,p]=max(x) % m balio maximoa eta p bere posizioa x bektorean
m =
     2
p =
     2
>> A
A =
     1     2     3
     4     5     6
     7     8     9
>> max(A) % zutabez zutabeko maximoak
ans =
     7     8     9
>> [M,P]=max(A) % zutabez zutabe A-ren balio maximoak, M-n gordeak,
                    % eta P bektorean bere posizioak zutabe bakoitzean
M =
     7     8     9
P =
     3     3     3
>> max(A,[],2) % errenkadaz errenkada balio maximoak
ans =
```

```

3
6
9

```

Ohartu azkeneko instrukzioan ”[]“ kortexeteak erabili behar izan direla, zeren max agindua tamaina bereko matrize bi eratzeko ere erabil baitaiteke:

```

>> M=[-1 2; 3 4];N=[2 1; 2 7];
>> max(M,N)
ans =
     2     2
     3     7

```

Izan ere, [] hutsunea txertatzen baldin bada, Matlab-i adierazten zaio ez dagoela bigarren matrizea, eta maximoaren bilaketari errenkadaz errenkada ekiten zaio.

- **min:** max agindua bezala erabiltzen da, baina minimoa kalkulatzen du ikusitako era berean.
- **norm:** Bektore edo matrize baten norma kalkulatzen du, eta aurredefinituriko norma batzuk aukeratu daitezke:

```

x =
    -2     2     1    -1
>> [norm(x) norm(x,1) norm(x,inf)]
    % norma 2, norma 1 eta norma infinitua, hurrenez hurren
ans =
    3.1623    6.0000    2.0000
>> A=[1 2; 3 4];
>> [norm(A) norm(A,1) norm(A,inf)]
    % A-ren norma matritzialak dira
ans =
    5.4650    6.0000    7.0000

```

- **rank:** Matrize baten heina, hau da, errenkada edo zutabe linealki independenteen kopurua da emaitza:

```

>> B=[1 2 3; -1 1 2; -1 4 7]
B =
     1     2     3
    -1     1     2
    -1     4     7
>> rank(B) % B-ren heina 2 da
ans =
     2

```

- `sort`: Bektore baten elementuak, txikienetik handienara, edo alderantziz, goranzko edo beheranzko ordena mantenduz sailkatzen dira:

```
>> x
x =
    -2     2     1    -1
>> sort(x) % ordena gorakorra
ans =
    -2    -1     1     2
>> -sort(-x) % ordena beherakorra
ans =
     2     1    -1    -2
```

- `diff`: Bektore baten ondoz ondoko elementuen kendurak egiten ditu. Hots, n luzerako x bektorea baldin bada, emaitzak $n - 1$ osagai izango ditu eta i . osagaia $x(i + 1) - x(i)$ kendura izango da:

```
>> x
x =
    -2     2     1    -1
>> diff(x)
ans =
     4    -1    -2
>> diff(diff(x))
ans =
    -5    -1
>> diff(diff(diff(x)))
ans =
     4
```

- Ondoko 2.7 Taulan bektoreak eta matrizeak aztertzeko beste eragiketa batzuk deskribatzen dira, eta haien implementazioa irakurleen esku uzten da.

| Agindua | Esanahi matematikoa |
|----------------------|--------------------------------|
| <code>mean</code> | Batezbesteko aritmetikoa |
| <code>std</code> | Desbiderazio estandarra |
| <code>var</code> | Bariantza |
| <code>cumsum</code> | Elementuen batura pilatua |
| <code>cumprod</code> | Elementuen biderkadura pilatua |

2.7 Taula: Bektore eta matrizeekiko beste oinarriko agindu batzuk.

2.5.7 Gelaxka-bektoreak eta egiturak

Datu askoren artean, Matlab-en gelaxka-bektorea edo *cell array* existitzen da. Datu-tipo bat da, oso malgua da, eta erabileraren aldetik, bektoreetan deskribatu diren erabileratik

oso gertu dago. Tipo honetan sarrerak tipo askotakoak izan daitezke, hau da, ez daude tipo berekoak izatera beharturik, bektoreekin gertatzen den legez. Beraz, idazkera nahasian, zenbakiak, testuak, matrizeak, konstante logikoak, inline funtzioak ager daitezke, besteak beste.

Gelaxka-bektore bat sortzeko, nahikoa da zuzenean balioak esleitzea:

```
>> gela={12,[1:10],[1 2;3 4],'testua',inline('x-sin(x)')}
gela =
    [12]    [1x10 double]    [2x2 double]    'testua'    [1x1 inline]
>> whos gela % Gelaxkaren deskripzioa lortzeko.
Name      Size      Bytes  Class  Attributes
gela      1x5       1264   cell
```

Orain, sententziari begira, kortxeteak erabili beharrean, giltzak baliatu dira. Era berean, beste era batean, gelaxkako osagaiak elementuz elementu sar daitezke:

```
>> gela{1}=12;
>> gela{2}=[1:10];
>> gela{3}=[1 2; 3 4];
>> gela{4}='testua';
>> gela{5}=inline('x-sin(x)');
```

Berebat, baina sintaxiaren aldetik zenbait desberdintasun aintzakotzat harturik, honela jardun daiteke:

```
>> gela(1)={12};
>> gela(2)={[1:10]};
>> gela(3)={[1 2; 3 4]};
>> gela(4)={'testua'};
>> gela(5)={inline('x-sin(x)')};
```

Edozelan ere, behin bektorea definiturik, elementuak atzitzeko, haien posizioak giltza bidez seinalatzen dira.

```
>> gela{3}
ans =
     1     2
     3     4
>> gela{5}
ans =
    Inline function:
    ans(x) = x-sin(x)
>> gela{5}([pi/2 2*pi]) % funtzioaren balioztapena
ans =
    0.5708    6.2832
```

Orobat, gordetako guztia celldisp aginduari esker pantailatik erakuts daiteke:

```

>> celldisp(gela)
gela{1} =
    12
gela{2} =

     1     2     3     4     5     6     7     8     9    10
gela{3} =

     1     2
     3     4
gela{4} =
testua
gela{5} =
    Inline function:
    (x) = x-sin(x)

```

Egia bada ere informazioa pilatzeko datu-tipo hori oso malgua dela, Matlab-ek informazioa eraginkortasun gutxiagoarekin maneiatzen du, eta, horrenbestez, datu-tipo horren neurriko erabilera egitea komeni da.

Beste modu batean askotariko datu-tipoak taldekatzeko egiturak edo *struct* daude. Gelaxken propietate ugari partekatzeaz gain, badago alde nabaria: gelaxkak indexatuta daude; alegia, osagai bakoitzari zenbaki bat elkartzen zaio, posizioa seinalatzen duena. Egiturretan, ostera, "eremu" zenbakitugabeak erabiltzen dira. Esaterako, Pelikula egiturak karaktere-kateak diren Titulua eta Zuzendaria eremuak izan ditzake, baita zenbakizko Urtea eremua ere. Ikusi nola sartu eta nola atzitu eremu konkretu bat". "eragileaz baliatuz.

```

>> Pelikula.Titulua='Shrek';
>> Pelikula.Zuzendaria='Almodovar'
Pelikula =
    Titulua: 'Shrek'
    Zuzendaria: 'Almodovar'
>> Pelikula.Urtea=2005
Pelikula =
    Titulua: 'Shrek'
    Zuzendaria: 'Almodovar'
    Urtea: 2005

```

Ikusten denez, Matlab-en egituraren patroia ez da beharrezkoa alde aurretik definitzea. Aski da eremu batzozteko balioak sartzea. Egitura beste era batean defini daiteke:

```

>> Pelikula=struct('Titulua','La vida es bella','Zuzendaria',
    'Roberto Bellini', 'Urtea',2002)
Pelikula =
    Titulua: 'La vida es bella'
    Zuzendaria: 'Roberto Bellini'
    Urtea: 2002

```

Halaber, errazki, egiturazko bektoreak sor daitezke:

```
>> Pelikula(3)=struct('Titulua','La vida es bella','Zuzendaria',
                    'Roberto Bellini','Urtea',2002)
```

```
Pelikula =
1x3 struct array with fields:
    Titulua
    Zuzendaria
    Urtea
```

Horrela 1x3 bektore bat eratzen da, eta hirugarren osagaia zehaztutako balioekin betetzen da. Portaera hori bektore eta matrizeen kasuan nabarmendu da. Hutsik dauden gainontzeko eremuak betetzeko, honelako sententziak idatz daitezke,

```
>> Pelikula(2).Titulua='Conan';
>> Pelikula(3).Zuzendaria='Michael Curtiz';
```

eta eremu baten elementu guztiak begietaratzeko, nahikoa da hau tekletzea:

```
>> Pelikula.Titulua
ans =
[]
ans =
Conan
ans =
La vida es bella
```

Betiere, egiturari noiznahi erants dakioke eremu berri bat,

```
>> Pelikula(1).Hizkuntza='Euskera'
Pelikula =
1x3 struct array with fields:
    Titulua
    Zuzendaria
    Urtea
    Hizkuntza
```

edo sortu beste eremu berri bat, egitura dena. Kasu horretan, kontuan hartu behar da barneko egiturako eremuak erabiltzeko eta atzitzeko ". " eragilea bi aldiz erabili beharko dela.

Egiturei lotutako funtzio franko daude, bereziki egiturekin eragiketak egiteko. Hauek dira haietako batzuk, gehien agertzen direnak:

```
fieldsnames getfield isfield rmfield setfield
```

3. Kapituluia

Funtzioak eta komando-fitxategiak

Lehenengoz funtzioei buruzko ideia irudikatzeko, imajinatu funtzio matematiko bat. Normalean, funtzio matematiko batean argumentu batzuk sartzen dira, eta, behin manipulaturik funtzioaren bitartez, bestetzuk ateratzen dira. Matlab-en jadanik programaturiko hainbat eta hainbat funtzio, aldez aurretik programatuak deritzenak, aurkitu daitezke, hala nola, $\sin(x)$, $\tan(x)$, \sqrt{x} eta $\exp(x)$. Dena den, programazioa egiten denean, askotan beste funtzio batzuen premia izaten da, erabiltzaileak sortzen dituenena, hain zuzen.

Azken finean, erabiltzaileak definituriko funtzio bat Matlab-eko programa bat baino ez da, maneiatzaileak idazten duena eta disko gogorrean gordetzen duena. Behin hori guztia eginda, funtzio horien eta aurredefinitutakoen erabilera oso antzekoa da. Funtzioek sarrerak eta irteerak dituzte; hori dute ezaugarri nagusia. Hartara, barruan exekututzen diren kalkuluak sarrerako datuei esker egiten direla ohartzekoa da, eta irteeraren bitartez, kalkuluen emaitzak pantailatik zein fitxategi batetik ateratzen dira. Bai sarreran, bai irteeran, osagaiak eskalarrak zein bektoreak, matrizeak edo testuak izan daitezke, edozein dimentsiotakoak izanda ere. Halatan, erabilera malguagoa eta zabalagoa ematen dute, dudarik ez.

3.1 Funtzioak sortzea

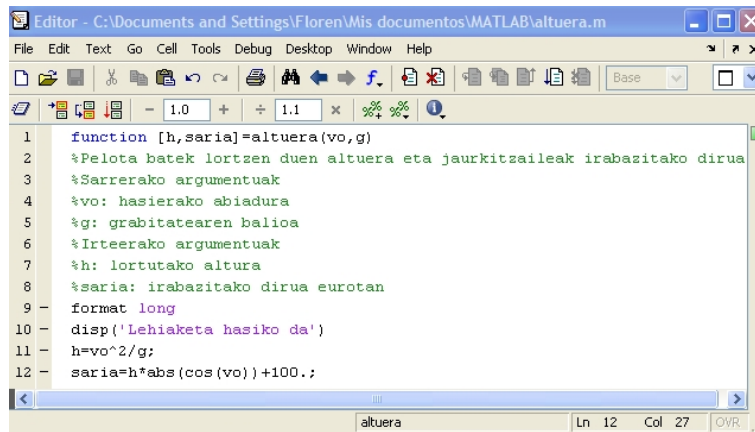
Funtzio berri bat sortzeko eta editatzeko, Matlab-eko leiho nagusitik abiaturik, *File* menua aukeratu eta *New* aukera zabaldu eta gero, *M-file* hautatu. Aurrez aurre, *Edizio Leihoa* agertuko da. Editore normal baten antza dauka. Aukeran, prozesu desberdina baina onartzen dena, *Aginduetarako Leihoan* edit funtzioaren izena idatzi eta *Enter* sakatzea da. Hala, berriro ere, *Edizio Leihora* ailegaten da.

3.1.1 Funtzioen egitura

Funtzio baten ohiko egitura 3.1 Irudian erakusten da. Hain zuzen, joko baten etsenplu honetan, pertsona batek gora jaurtitutako pelota batek iristen duen altuera eta, horren ondorioz, berak irabazten duen dirua kalkulatzeko dituzten funtzioak.

Funtzioaren idazkerari erreparatuz, lehenengo lerroan funtzioaren beraren definizioa agertu behar da; osterantzean, gero ikusiko denez, komando-fitxategi soil bat baino ez litzateke izango, ez funtzio bat. Lerroa aztertuz gero, hauek dira ezaugarri nagusiak eta aipagarriak:

- Hasteko, nahi eta nahi ez, `function` hitza agertzen da, minuskulaz gainera. Horrela, Matlab-i jakinarazten zaio erabiltzaileak eratutako kodea funtzio bat dela.
- Segidan, irteerako argumentuak kortexete artean idatzi behar dira. Funtzioaren emaitzak dira, eta, behin funtzioa egikariturik, Workspace-n gorderik lotzen dira.
- Hurrengoa funtzioaren izena da, erabiltzaileak eman behar duena, era bakarrean funtzioa identifikatuko duena.
- Azkenik sarrerako argumentuak, zenbatgura, eta parentesi artean. Funtzioari deitzen zaion bakoitzean, aldagai horiek edozein motatako balioak esleituak izan behar dituzte.



```

1 function [h,saria]=altuera(vo,g)
2 %Pelota batek lortzen duen altuera eta jaurkitzaileak irabazitako dirua
3 %Sarrerako argumentuak
4 %vo: hasierako abiadura
5 %g: grabitatearen balioa
6 %Irteerako argumentuak
7 %h: lortutako altura
8 %saria: irabazitako dirua eurotan
9 format long
10 disp('Lehiaketa hasiko da')
11 h=vo^2/g;
12 saria=h*abs(cos(vo))+100.;

```

3.1 Irudia: Funtzio baten ohiko egitura.

Halaber, eta 3.1 Irudiarekin jarraikiz, iruzkinduriko zenbait lerro, bitik zortzira hain justu, idatzita daude. Egia esan, beharrezkoak ez badira ere, komenigarria da programa eta funtzio oro iruzkintzea. Izan ere, iruzkinetan lehenengo esaldia esaldi berezia da, eta *HI* lerroa deitzen zaio. Hartan, funtzioari buruzko informazio orokorra, deskribapenak-eta idatzi ohi dira. Izan ere, gero edozein unetan, esaterako, *Aginduetarako Leihotik* lookfor hitz bat exekutatzen den bakoitzean, Matlab-ek funtzio guztien multzoan hitz hori *HI* lerroan duten funtzio guztiak itzultzen ditu. Halaber, definituriko funtzioaren informazio zehatzagoa lortzeko, beti `help funtzioaren izena` lagungarri dago. Kasu honetan, bi eta zortzi lerroren arteko informazioa pantailan ageriko da, erabiltzaileak funtzioaren barruan idatzitakoa, hain zuzen. Begiratu honako adibide honi:

```
>> help atanh
```

```
ATANH Inverse hyperbolic tangent.
```

```
ATANH(X) is the inverse hyperbolic tangent of the elements of X.
```

Hemen `atanh` funtzio aurredefinituaren oinarriko laguntza azaltzen da, baina etsenpluko `altuera` funtzioarekin beste hainbeste gertatuko zen.

Azalpenez azalpen, jarraian funtzioaren gorputza bera hasten da. Kontuan izan beste iruzkin batzuk ager daitezkeela, baita lerro hutsak ere. Dena dela, bereziki komandoak eta sententziak idazteko lekua eta unea da, eta, harik eta `return` agindua edo funtzioaren amaiera topatu arte, exekutatu egingo dira. Alta, `return` edozein posiziotan agertuta ere, jakin behar da funtziotik irteten dela.

Aintzat hartu funtzioaren barruko aldagai eta balioak **lokalak** direla, hots, barneko aldagaien balioak soil-soilik funtzioan bertan ezagutuko direla, sarrerakoak eta irteerakoak salbu. Matlab balioak gordetzeko memoria-zona berezi batez baliatzen da, eta ez da *Workspace*-n erabilia. Halatan, funtzioek ez dituzte funtziotik kanpoko aldagaiak ezagutzen. Halandaze, funtzio bakoitzak bere aldagai lokalak ditu, beste funtzio batzuekin, esate baterako, partekatzen ez dituenak. Halan eta guztiz ere, baliteke aldagaiak sortzea, gero, bai beste zenbait funtziotan, bai *Workspace*-n usatu ahal direnak. Hori gauzatzeko, global-en bitartez, eta era horretan, global izen bat, aldagai **orokorrak** deklaratu dira.

- Funtzio batzuek ezagutu eta partekatuko dituzten aldagaiak funtzio bakoitzean global atributuarekin deklaratu behar dira.
- Funtzioek aldagaia erabili orduko, guztietan global komandoa agertu behar da.
- Aldagai bati, nahiz eta orokorra izan, edonoiz balio berriak eslei eta berreslei dakizkioke.

Funtzioa disko gogorrean gordetzeko tenorean, komeni da fitxategiaren izena eta lehenengo lerroan idatzitakoa bat etortzea, beharrezkoa ez bada ere. Hori bai, izenak desberdinak hautatuz gero, jakin behar da exekuzioaren unean Matlab-ek fitxategiaren izenari lehentasuna ematen diola.

Azpirarratzekoa da, bestalde, *Aginduetarako Leihoan* funtzioa usatu ahal izateko, laneko direktorioan sartu behar dela, eta fitxategiaren deskribapena izena `.m` izango dela. Izan ere, `M` fitxategiak deitzen dira. Behin hori eginda, funtzioa *Aginduetarako Leihotik* erabili ahal da:

```

1 >> abiadura=2;
2 >> grabitate=9.8;
3 >> altuera(abiadura,grabitate)
4 Lehiaketa hasiko da % disp aginduaren irteera
5 ans =                % altuera bakarrik
6     0.408163265306122
7 >> [altu,diru]=altuera(abiadura,grabitate)
8 Lehiaketa hasiko da
9 altu =                % altuera
10    0.408163265306122
11 diru =                % dirua
12    1.001698558516519e+002

```

Lehenengo eta bigarren sententzietan altuera funtzioaren sarrerako bi argumentuen balioak definitu eta *Workspace*-n gordetzen dira. Hirugarrenean, funtzioari berari deitzen

zaio. Sarrerako aldagaien izenak edozein izan daitezke, hots, ez dute nahitaez funtzioaren definizioan agertzen direnak izan behar. Halaber, ohartzekoa da bosgarrenean irteera bat baino ez dela lortzen. Kontua da, orokorki besterik idazten ez bada, eta lehenespen gisa, funtzioaren definizioan irteerako ezkerreko aldagaiaren balioa bakarrik itzultzen dela, goiko adibidean pelotaren altuera, hain justu. Portaera hori Matlab-eko edozein funtziotan ageri da. Zazpigarren lerroan, berriz, erabiltzaileak funtzioa irteerako emaitza guztiak ateratzera behartzen du, lehen ez bezala. Bestela ere, xehetasun txikia izanagatik ere, programa honetan `disp` agindua agertzen da. Haien bitartez, barruan sartutako testua pantailan ateratzen da, laugarren esaldia kasu.

Funtzioen osaketan aukera ugari dago. Kasu baterako, ikusi beste adibide bat, zeinean funtzio bat eta beraren deribatua balioztatzen diren, sarrerako eta irteerako argumentuak bektoreak edo eskalarrak direlarik:

```
function [f,fprima]=fun(x,a)
% FUN-ek parametro baten menpeko funtzioa eta beraren deribatua puntu
% batzuetan balioztatzen ditu.
% x funtzioaren aldagai erreala edo bektoriala
% a parametro erreala
% f funtzioa
% fprima f-ren deribatua
f=x.*(1+a*x.^2)-exp(a*x); % notazio bektoriala erabiliz
fprima=1+3*a*x.^2-a*exp(x); % notazio bektoriala erabiliz
return;
```

Kasu honetan, notazio bektorialaren erabilerari erreparatu, zeren biderketa `.` ikurrari esker elementuz elementu gauzatzen baita. Izan ere, goikoa egikaritzen baldin bada, `x` aldagaia bektore bat izan daiteke, funtzioaren definizioan eragiketa bektorialak egiteko `”.` ikurra sartu baita:

```
>> [f,fderib]=fun(1:3,2)
f =
   -4.3891   -36.5982  -346.4288
fderib =
    1.5634    10.2219    14.8289
```

Ohartu, funtzioaren definizioan `a`-ren eta `x`-ren arteko biderkaduran notazio bektoriala baliatu ez denez, `a` parametroa ezin dela bektore bat izan.

Azpilekzio hau bukatze aldera, aipatu beharra dago posible dela argumentu gabeko funtzioak eraikitzea, bitxia bada ere, eta irakurleari kode hau exekutatzeko proposatzen zaio:

```
function agurra
disp('Ongi etorri')
disp('unibertsitatera')
disp('izan ondo eta ikasi')
return
```

Igarrian da hemen ez dagoela sarrerako argumenturik, ez eta irteerako argumenturik ere.

3.1.2 Funtzioak egituratzeko beste era batzuk

Segidan, Matlab-ek duen malgutasuna adieraztearren, eta aurrekoa gogoan harturik, orain funtzio bat definitzeko askotariko erak eta posibilitateak ikusiko dira, baita propietate batzuk azpimarratu ere.

1. Fitxategi batean funtzio zenbait sar daitezke, ez soilik bakar bat. Kasu honetan, fitxategitik kanpo (*Aginduetarako leihotik* edo beste funtzio batetik) lehenengo funtzioa bakarrik deitu daiteke. Izan ere, gainontzeko funtzioak barruko funtzioak dira, eta gogoan izan funtzio nagusiak bere kalkuluak egiteko erabiltzen dituela. Xehetasun hori kontuan hartzekoa da, batik bat, moduluzko programazioa prestatzen denean. Mota honetako programazioan, funtzio printzipalak funtzio-multzo bat baino erabiltzen ez badu, denak batera fitxategi berean sar daitezke. Horrela, laneko karpeta alferrikako fitxategiz betetzea ekiditzen da. Begiratu adibide hau:

```

1  function pondera=batezbeste(fak,x)
2  m=max(x);
3  n=faktor(fak);
4  pondera=sum(x)*m*100/n;
5  return
6
7  function p=faktor(q)
8  p=exp(q);
9  return

```

Aurreko etsenpluan fitxategi berean funtzio bi daude. Lehenengoa, batezbeste funtzioa, funtzio nagusia da, eta gorri seinalaturik dago. Bertan bigarren mailako faktor funtzioa agertzen da, biak ala biak fitxategi berean. Azpimarratu behar da, funtzio horiek artean aldagaien balioak ez dituztela partekatzen; beraz, bigarren funtzioaren idazkeran, esaterako, p-ren ordez m letra ipin zitekeen. Baieztapen hori egiaztatzea proposatzen da.

2. Matlab-eko funtzio bat sartzeko, badago beste era bat, baina, lehen ez bezala, jakin behar da ez duela .* luzapenik behar. Horretarako, *Aginduetarako Leihoan inline* erabiliz, funtzio-mota bat definitzen da,

```

>> f=inline('cos(x)-sin(2*x)', 'x')
f =
    Inline function:
    f(x) = cos(x)-sin(2*x)

```

eta f-ek $\cos(x) - \sin(2x)$ funtzioa espezifikatzen du. Behin definiturik, funtzio berria balioesteko, gainontzeko funtzioetan bezala egiten da:

```

>> f(pi)
ans =

```

```

-1.0000
>> f(pi:pi:4*pi)
ans =
-1.0000    1.0000   -1.0000    1.0000

```

Aldagai batekoa baldin bada, bai eskalarrak bai bektoreak sar daitezke. Praktikan, `inline` exekututzea, batetik, "f" izeneko funtzio bat editatzea eta horri dagokion kodea sartzea, bestetik, baliokidea da. Dena den, `f`-ren definizioa memorian gordetzen da, baina, behin saioa Matlab-en itxita, aplikazioa amatatuta, alegia, funtzioa galdu eta desagertzen da. Dударik ez, horixe da funtzio bat forma horretan definitzeak duen desabantaila nagusietakoa; alabaina, aldi berean, ataza sinpleak egiteko aukera interesgarria ere izan daiteke. Ez dago inolako eragozpenik argumentu gehiago dituen beste `inline`-funtzio bat zehazteko,

```

>> g=inline('x*y*tan(x-2)-abs(x-y)', 'x', 'y')
g =
  Inline function:
  g(x,y) = x*y*tan(x-2)-abs(x-y)

```

baina, ohar gaitezen, orain funtzioa ez dago bektorizaturik; horrenbestez, behean erakusten denez, ez ditu bektoreak argumentu gisa onartzen:

```

>> g(1:2,2:3)
??? Error using ==> inlineeval at 15
Error in inline expression ==> x*y*tan(x-2)-abs(x-y)
??? Error using ==> mtimes
Inner matrix dimensions must agree.

```

Bat-bateko arazoa konpontzearen, bi era irispidean jar daitezke. Bateko, idatzi funtzioa, baina . ikurra erabiliz, hau da, `inline` aginduaren barnean bektore-notazioa erabili,

```

>> g=inline('x.*y.*tan(x-2)-abs(x-y)', 'x', 'y')
g =
  Inline function:
  g(x,y) = x.*y.*tan(x-2)-abs(x-y)

```

eta, handik aurrera, problema soluzionaturik geratzen da,

```

> g(1:2,-1:0)
ans =
-0.4426   -2.0000

```

edo bestela, bektorizatu gabeko `inline` formatik abiatu, `vectorize` komandoa erabiliz:

```
>> g=inline('x*y*tan(x-2)-abs(x-y)', 'x', 'y')
g =
    Inline function:
    g(x,y) = x*y*tan(x-2)-abs(x-y)    % ez bektorizatua
>> g=vectorize(g)
g =
    Inline function:
    g(x,y) = x.*y.*tan(x-2)-abs(x-y) % bektorizatua
```

Ohartu zaitetz agindu hori aplikatutakoan, Matlab-ek, leku egokietan ”.” erantsiz, funtzioa birdefinitzen duela.

Dena den, Matlab 7.0 bertsiotik aurrera badago beste metodo bat, bai *Aginduetarako Leihotik*, bai beste funtzio baten barruan, funtzio berri bat sortzeko, eta sintaxia honakoa da:

```
fun_izena=@(argumentuak) adierazpena
```

Horrela definituriko funtzioa funtzio **anonimoa** deitzen da, eta egitura hori jarraituz, aurreko adibideetan sartutako funtzio berdintsuak espezifika daitezke:

```
>> f=@(x) cos(x)-sin(2*x); % ez bektorizatua
>> g=@(x,y) x.*y.*tan(x-2)-abs(x-y) % bektorizatua
```

Ostean, funtzio horien balioztapena Matlab-eko beste funtzioetan lez egiten da:

```
>> g([1 2], [-1 0])
ans =
    -0.4426    -2.0000
```

Segurik, funtzioetarako idazkera horren bitxikeria bakarra zera da, ez duela `vectorize` agindua onartzen. Halatan, egitura hau `inline` egituraren beste alternatiba bat da, eta biak arrapaladan aplika daitezke funtzio berri bat definitzeko.

3. Gaur egungo programazio-lengoaia askotan bezala, Matlab-ek errekurtsibitatea uzten du, hau da, funtzio batek zenbat-nahi aldiz dei diezaioke. Menturaz, adibide sinpleena eta klasikoena faktorial funtzioarena da:

$$n! = n.(n - 1).(n - 2) \dots 1$$

Era sinple batean, zenbaki baten faktoriala era errekurtsibo batean defini daiteke:

$$n! = \begin{cases} n.(n - 1)! & \text{baldin } n > 0 \\ 1 & \text{baldin } n = 0 \end{cases}$$

Berbarako, Matlab-en funtzio faktorialaren implementazioari dagokionez, hau izan daiteke:

```

1   function f=fakt(n)
2   % FAKT funtzio faktoriala
3   % n: zenbaki arrunta
4   % f: n-ren faktoriala
5   if n<0 % n negatiboa bada
6       disp('Balio ez-egokia')
7       f=[]; % ez da ezer itzultzen, matrize hutsa
8   elseif n==0
9       f=1; % n=0 bada
10  else
11      f=n*fakt(n-1); % Hemen errekursibitatea agertzen da
12                          % n-1 argumentua delarik
13  end
14  return

```

Errekursibitatea hamaikagarren lerroan islatzen da, berriz ere `fakt` funtzioari deitzearekin. Argi dago funtzio faktoriala ez dela programatzeko modu egokiena eta eraginkorra. Izan ere, Matlab-ek horretarako `factorial` agindua dauka, eta bere kodea edit `factorial` tekletuz gero, ikusgai dago. Programazio-mota hori neurritz kanpo ez erabiltzea gomendatzen da, kode ez-eraginkorra sortzen baita. Etengabeko deiak direla eta ez direla, ordenagailuko memoriaren kostua oso altua da, eta beraz, teknika hori neurritz eta tentuz aplikatzekoa da.

3.2 Funtzioen argumentu aldakorrak

Jarraian, irteerako eta/edo sarrerako argumentu kopuru aldakorra duten funtzioak nola programatu daitezkeen erakutsiko da. Ezaugarri horrek Matlab-en egiten den programazioari malgutasun handiagoa ematen dio. Nagusiki, kopuru ez finkatua inplementatzeko beharrezko aginduak hauek dira:

```
varargin  nargin  varargout  nargout
```

Izan ere, funtzio bati deitzen zaion bakoitzean, `nargin` eta `nargout` komandoek (*number of input arguments*, eta *number of output arguments*) sarrerako eta irteerako aldagaien kopuruen berri ematen dute. Erreparatu honako funtzio honi:

```

1  function [erroa] = sqrtM(a,n)
2  % sqrtM: a zenbakiaren n. erroa
3  % a: errokizun positiboa eta erreala
4  % n: errotzaile arrunta
5  % erroa: emaitza
6  if nargin<2
7      n=2; % n ez badago erro karratua egin
8  end
9  if a<0

```

```

10     disp(['Errokizuna negatiboa da. Sartu zenbaki positibo bat']);
11     return
12 else
13     erroa=a^(1./n);
14 end
15 return

```

Aurreko adibidean nargin komandori dagokion adibide bat erakusten da. Hain zuzen, seigarren lerroan agertzen da eta, horri esker, `sqrtM` funtzioaren sarrerako argumentu kopurua aldakorra izan daiteke. Kasu honetan, argumentu bat baizik ez sartzea onartzen da, zeren eta, orduan, `n` errotzaileari 2 zenbakia esleitzen baitzaio. Halaber, erreparatu erro-kizuna negatiboa baldin bada funtziotik kanpo irteten dela eta pantailan azalpen-mezu bat ateratzen dela.

Irteerako argumentuetarako beste hainbeste nargout komandoari dagokionez, aplikazioaren ideia nargin komandoaren parekoa baita. Ikusi adibide hau:

```

1 function [sail,batez,media,desbide]=means(x)
2 % MEANS: estatistiko batzuk kalkulatu eta sarrera ordenatu
3 % x: sarrerako datua
4 % sail: x bektorea ordenatua txikienetik handienera
5 % batez: batez besteko aritmetikoa
6 % media: mediana
7 % desbide: desbideratze tipikoa
8 sail=sort(x);
9 if nargout>1, batez=mean(x); end
10 if nargout>2, media=median(x); end
11 if nargout>3, desbide=std(x); end
12 return

```

Aurreko etsenpluan, `nargout` gorri nabarmendurik agertzen da. Izan ere, funtzioan irteerak pantailan azaltzeko unerako, bederatzi, hamar eta hamaika lerroak dira horretaz arduratzen direnak. Kopuru horren arabera, esaterako, `nargout=2` baldin bada, `x` bektorearen ordenazioa eta batez besteko aritmetikoa lortzen dira; edo `nargout=1` baldin bada, oster, ordenazioa baino ez da pantailan agertuko. Erabiltzaileak funtzioa idazten duenean, behean azaltzen denez, zenbat irteera nahi dituen seinlatu behar du:

```

>> means([2 -2 4 0 1 -5]) % soilik ordenazioa
ans =
    -5    -2     0     1     2     4
>> [s,b]=means([2 -2 4 0 1 -5]) % bi irteera
s =
    -5    -2     0     1     2     4
b =
     0
>> [s,b,m]=means([2 -2 4 0 1 -5]) % hiru irteera
s =

```

```

    -5    -2     0     1     2     4
b =
     0
m =
    0.5000

```

Komando hauetaz gain, funtzio batek argumentu zehatz batzuk, aurreko sekzioan egin den bezala, eta beste argumentu hautazko batzuk onar ditzake. Azken aldagai horiek atzitzeko eta funtzioari deitzeko, `varargin` (*variable input arguments*) eta `varargout` (*variable output arguments*) aginduak giltza bidez baliatzen dira. Kontua da `varargin` eta `varargout` aldagaiak *cell array* motakoak direla, eta, aurreko kapituluaren ikusienez, behar-beharrezkoa dela idazkeran datuak sartzeko giltzak ipintzea. Begiratu honako adibide esanguratsu honi:

```

1 function varargout=momen(x)
2 % MOMEN: bektore baten momentuak
3 % x: hasierako bektorea
4 % varargout: irteera aldakorra j. ordenako momentu zentrala
5 for i=1:nargout
6     varargout{i}=sum(x.^i)/length(x);
7 end

```

Bosgarren eta seigarren lerroetan, `nargout` irteerako aldagaien kopuruaren arabera, eta `for` begizta baten bitartez, hain justu, *cell* tipoko aldagaiei ordena desberdinetako momentu zentralak esleitzen zaizkie. Jarraian, erabiltzaileak zehaztutako irteerako aldagaien kopuruaren arabera, ikusi nola lortzen diren irteerak:

```

1 >> momen([1:4])
2 >> a=momen([1:4]) % 1. ordenako momentua
3 a =
4     2.5000
5 >> [a,b]=momen([1:4]) % 1. eta 2. ordenako momentuak
6 a =
7     2.5000
8 b =
9     7.5000
10 >> [a,b,c]=momen([1:4]) % 1. 2. eta 3. ordenako momentuak
11 a =
12     2.5000
13 b =
14     7.5000
15 c =
16     25

```

Adibide honetan, bigarren lerroan irteerako `a` argumentu bat zehazten da, eta Matlab-ek `varargout{1}` itzultzen du. Bosgarrenean berriz, `a` eta `b` irteerak esplizitatzen dira, hau da,

`varargout{1}` eta `varargout{2}` dira irteerak, 1. eta 2. ordenako momentu zentralak, hain zuzen.

Bestalde, hurrengo adibidean, matrize baten LU deskonposizioa egiteko, aztertzen ari diren lau aginduok batera nahasten eta aurkezten dira:

```

1 function varargout=LUdeskonposizio(A,varargin)
2 %LUDESKONPOSIZIO: A matrize baten LU deskonposizio matriziala, A=LU
3 %[L,U]=LUdeskonposizio(A)
4 %     L behe-tringeluarra eta
5 %     U goi-triangeluarra itzultzen dira
6 %[L,U,X]=LUdeskonposizio(A,b)
7 %     L behe-triangeluarra eta
8 %     U goi-triangeluarra
9 %     X AX=b sistemaren soluzioa
10 [L,U]=lu(A); %LU deskonposizioa
11 if nargin==1 && nargout==2
12     varargout{1}=L;
13     varargout{2}=U;
14 elseif nargin==2 && nargout==3
15     b=varargin{1};
16     varargout{1}=L;
17     varargout{2}=U;
18     varargout{3}=U\ (L\b);
19 end
20 return

```

Funtzioko lehenengo lerroari gainbegirada bat ematen baldin bazaio, `varargout` eta `varargin` aurkitzen dira. Horrek adierazten du argumentuen irteerak eta sarrerak aldakorrak izan daitezkeela. Iruzkindutako hasierako lerroak irakurtzen baldin badira, funtzioaren jomuga matrize baten LU deskonposizioa egitea da. Berbarako, kodean bertan sarrerako aldagai- kopurua 1 baldin bada eta irteerako argumentu kopurua 2 baldin bada, L eta U irteerak, `varargout{1}` eta `varargout{2}` *cell-arrays* aldagaietan gordetzen dira. Hamabostean, A matrizetik aparte, `varargin` aginduari esker, funtzioaren sarreran beste aldagai bat onartzen da, hain zuzen, $Ax=b$ sistemaren b bektore independentea. Hamasei eta hamazortzi bitarteko lerroetan, lehen iruzkindu den legez, lortutako balioak hiru `varargout` aldagaietan gorde eta pantailatik ateratzen dira. Hona hemen saiakera batzuk zenbait exekuzio desberdinen emaitzak agerian uztearren:

```

>> LUdeskonposizio([1 2 3; -1 0 3; 0 -2 1])
% nargout=0 eta nargin=1, beraz ez dago irteerarik
>> [L,U]=LUdeskonposizio([1 2 3; -1 0 3; 0 -2 1])
% nargout=2 eta nargin=1
L =          U =
     1         0         0         1         2         3
    -1         1         0         0         2         6
     0        -1         1         0         0         7

```

```
>> [L,U,X]=LUdeskonposizio([1 2 3; -1 0 3; 0 -2 1],[-1;2;0])
% nargout=3 eta nargin=2
L =          U =          X =
    1         0         0         1         2         3        -1.5714
   -1         1         0         0         2         6         0.0714
    0        -1         1         0         0         7         0.1429
```

3.3 Funtzioen argumentuak beste funtzio batzuk izatea

Kapitulu honetan zehar, sarrerako argumentuak beti-beti bektoreak edo matrizeak izan dira. Halarik ere, programazio aurreratuan ohikoa da funtzioek, bere exekuzioan, beste funtzio batzuk behar izatea. Horregatik, Matlab-en funtzio bat beste funtzio baten argumentua izan daiteke, eta, behar izanez gero, barruan sartutako funtzioa balioztatu ahal da `feval` komandoa lagun.

Adibide baten bitartez erreparatu egin eta jarraitu behar diren pausuei. Funtzio bat argumentua izan dadin, lehenik eta behin, funtzio nagusia, hau da, argumentu gisa beste funtzio batzuk erabiliko dituen, definitu behar da:

```
1 function grafiko(fun,a,b) % FUNTZIO NAGUSIA
2 % GRAFIKO: [a,b] tarteko funtzio biren grafikoak
3 % a,b: tartearen beheko eta goiko mugak
4 % fun: funtzio bat, grafiko izeneko funtzioaren argumentua
5 n=100;
6 x=linspace(a,b,n);
7 [y,z]=feval(fun,x); % fun funtzioaren balioztapena
8 plot(x,y,x,z);
9 return
```

Kasu honetan funtzio nagusia grafiko deritzona da. Bigarren pausoa ahaztu barik, beste fitxategi aparteko batean fun funtzioa, argumentua dena, idatzi behar da:

```
function [m,n]=fun(x)
m=x.^2-cos(x);
n=x.^2+sin(x);
return
```

Aipatu beharra dago funtzio nagusiko lehenengo lerroko fun argumentua funtzio bat dela eta idazkeraren aldetik, funtzio nagusiaren beste argumentuen kontra, ez dagoela inolako berezitasunik. Exekuzioari dagokionez, programa puntu batera ailegatzen da zeinetan fun funtzioa balioztatzeko beharrean aurkitzen den, eta, horretarako, `feval` agindua laguntzaile dago; hau da sintaxia:

```
feval(funtzioaren izena, argumentuak)
```

Goiko programa nagusiko zazpigarren lerroan `feval` agindua erabiltzen da. `fun` funtzioaren definizioan irteerako argumentu bi agertzen direnez, bi horiek dira, funtzio nagusian, `y` eta `z` aldagaiak, `fun` funtzioaren irteerak jasotzen dituztenak eta gero programaren garapenean erabil daitezkeenak.

Halarik ere, `feval` komandoaren erabilera zabalagoa da, hemen, adibidez, *inline* funtzio batean erakusten denez:

```
>> g=inline('a.*x.^2','x','a')
g =
    Inline function:
    g(x,a) = a.*x.^2
>> g([1,2],[-1,-2])
ans =
    -1    -8
>> feval(g,[1,2],[-1,-2])
ans =
    -1    -8
```

Izan ere, *inline* funtzio bat definitzen baldin bada, jadanik ikusitako eran baliozta daiteke, edo `feval` komandoari esker ere egin daiteke. Hemen, eta `g`-ri definizioari begira, `[1,2]` balioak `x` bektoreari dagozkio eta `a` bektoreari `[-1,-2]` balioak. Era berean, `feval` agindua `Matlab`-en aurredefinituriko funtzioak ere balioesteko erabil daiteke,

```
1 >> feval('cos',[0,pi/2,pi])
2 ans =
3     1.0000     0.0000    -1.0000
4 >> feval(@sin,[0,pi/2,pi])
5 ans =
6         0     1.0000     0.0000
7 >> feval('log',[exp(1),exp(2),exp(-2)])
8 ans =
9     1     2     -2
10 >> feval(@log,[exp(1),exp(2),exp(-2)])
11 ans =
12     1     2     -2
```

alabaina, kasu honetan badago bitxikeria bat, kontuan hartu beharrekoa. Funtzioak balioz-tatu egin dira, baina `''` edo `"@"` ikurrak erabiliz. Hemen, *inline* funtzioetan ez bezala, ikur horietaz baliatu beharra dago, eta, dudarik ez, puntu honek azalpen bat merezi du.

Funtzio-*handle* bat `Matlab`-eko datu-tipo bat da, eta, funtzioa ebaluatzeko, horri buruzko informazio guztia gordetzen du. Funtzio-*handle* bat funtzio baten izenari `"@"` karakterea erantsirik sortzen da. Aurrean `cos` edo `log` funtzioak *Aginduetarako leihotik* ebaluatu nahi izan dira, eta, horretarako, nahitaez, *handle* idatzi behar dira exekuzioa bidera dadin.

Hurrena, pentsatu behar da erabiltzaileak definituriko funtzio nagusiak, argumentuen artean beste funtzioak dituztenak, nola ebalua daitezkeen. Egoera bat baino gahiago aurkitu

daitezke. Aurrena *Aginduetarako Leihoan* egonez gero, edo, osterantzean, beste funtzio nagusi batean funtzio nagusia delakoa egonez gero. Bietan erantzuna emateko, 42. orrialdeko grafiko programa idatzia aprobetxatuko da. Azken finean, koska hori hirutara burutu eta soluzionatu ahal da:

```
1 >> grafiko('fun',1,2)
2 >> grafiko(@fun,1,2)
3 >> feval(@grafiko,'fun',1,2)
```

Lehenengo eta bigarren ilaretan grafiko programa *Aginduetarako Leihoan* betiko lez egikaritzen da, eta grafiko funtzio nagusitik kanpo egiten denez, nahi eta nahi ez, fun funtzio-argumentuan '' edo @ ikurrak erabili behar dira. Bigarren aukera *Aginduetarako Leihoan* edo beste funtzio nagusi baten barruan aplikatu daiteke, eta feval komandoaz baliatzea da. Hor, grafiko funtzioa kanpokoa da, eta haren argumentu bat beste funtzio bat da. Bietan, *handleak* izkiriatzeko premia dago; osterantzean, zerbait idatzi ezean, Matlab-ek erroreak itzuliko ditu:

```
>> feval(@grafiko,fun,1,2)
??? Input argument "x" is undefined.
Error in ==> fun at 2
m=x.^2-cos(x);
```

```
>> feval(grafiko,fun,1,2)
??? Error using ==> grafiko
Too many output arguments.
```

```
>> feval(grafiko,@fun,1,2)
??? Error using ==> grafiko
Too many output arguments.
```

3.4 Aginduzko fitxategiak edo scripts

Kapitulua bukatu orduko, aipatzekoa da Matlab-en *scripts* izenekoak edo aginduzko fitxategiak existitzen direla. Berrito ere, funtzioekin gertatzen zen legez, sortzea oso erraza da, eta edit izena teklatzea aski eta sobera da, horrelako fitxategia izateko. Desberdintasun bakarra dago: fitxategia horiek Matlab-eko sententziaz eta aginduz betea dago, besterik ez. Azken batean, ez dute funtzioen egitura, eta era interaktiboan *Aginduetarako Leihoan* banan-banan sar daitezkeen komandoak, orain, fitxategi bakar batean bilduak daude guztiak. Hemen, ostera, *scripts* horien luzapena "m" da, eta, fitxategia exekutatzeko denean, horretarako izena aski eta sobera delarik, lerroz lerro idatzitakoa egikaritzeko joango da. Pantailan informazio larregi ez izateko, sententzia baten ostean puntu eta koma ipintzea komeni da. Halaber, *Aginduetarako Leihoan* exekuzioa gauzatu ahala, balioak *Workspace* memoriari gordez doaz, funtzioetan ez bezala. Alabaina, *script*-a beste funtzio baten barnean badago, haren balioak funtzioaren ingurunean baino ezin dira aprobetxatu.

Zenbaitetan, *script* bateko sententziak pantailan inprimatzea komeni da; horretarako, `echo` agindua dago. Komando horrek zenbait aukera ditu, eta hurrengo lerroetan batzuk aurkezten dira:

```
echo on          script guztietan echo aktibatzen da
echo off        echo desaktibatzen da
echo file on    file izeneko funtzioan echo aktibatzen da
echo file off   file funtzioan echo desaktibatzen da
echo file       on-etik off-era pasatzeko eta alderantziz fitxategian
echo on all     funtzio guztietan echo aktibatzen da
echo off all    funtzio guztietan echo desaktibatzen da
```

Komando horri buruzko ideia garbiak izateko, etsenplu bi ematen dira eta irakurleari goiko aukerekin praktikatzea proposatzen zaio. Lehenengo *script*-ari dagokionez,

```
% zorizko mxn ordenako matrizea sortzeko
% matrizearen dimentsioa erabiltzaileak hautatzen du
n=input('Sartu matrizeko erreskada kopurua');
m=input('Sartu matrizeko zutabe kopurua');
A=rand(n,m); % zorizko matrizea sortzen da
```

eta, bigarrenaz bezainbatean, honako hau:

```
p=[1 -1 -1 1];  $x^3 - x^2 - x + 1$  polinomioa
roots(p) % p polinomioaren erroak
x=-2:0.1:2;
y=polyval(p,x); % polinomioaren balioztapena x bektorean
plot(x,y) % irudikapen grafikoa
xlabel('x');ylabel('y');title('x^3-x^2-x+1') % etiketak eta titulua
```

Azkenez, goian polinomio batekin erabilitako aginduak zazpigarren kapituluan lantzen dira.

4. Kapitulum

Eragileak eta fluxuaren kontrola

Oro har, edozein ordenagailu-programa agindu sail bat baino ez da, instrukzio edo sententzia deritzenez osatua. Programa sinple batean aginduak banan-banan eta idatzitako ordenan exekututzen dira. Halarik ere, egoera anitzetan programa landuagoak egin behar dira, zeinetan idatzitako ordenan komandoak ez diren egikaritzen. Berbarako, enpresa batek bidalitako paketeen kostuaren kalkulua pisuaren eta tamainaren arabera egin daiteke, eta programan bertan adarkatze-aginduen premia aurkitu daiteke. Horrenbestez, sententzia egokien bidez programa horren fluxu normala aldatu behar da.

Matlab-ek, programa baten fluxua kontrolatu ahal izateko, askotariko aginduak ditu. Baldintzazko instrukzioek, switch programazio-egiturek, tarteko, salto bat eginez, sententzia batzuk ez exekutatzeko, edo soilik, komando espezifiko batzuen exekuzioa egitea ahalbidetzen dute. Bestela ere, iterazio-sententziek, beharrezkoa baldin bada, komando-segida bat hamaika aldiz egikaritzen uzten dute.

Esan gabe doa, programa baten fluxua aldatzeko eta erabaki doia hartzeko, prozesuren bat behar da programaren barruan. Ordenagailuak berak erabaki behar du hurrengo komandoa exekutatzeko duen, edo ondorengoa aintzat ez edukitzea eta programako beste lerro batean implementazioa jarraitzea. Aldagaien balioak erkatuz, programak erabaki horiek hartzen ditu, eta, hori gauzatzeko, nagusiki eragile logikoak eta erlazio-eragileak deritzenak erabiltzen dira, eta eginkizun garrantzitsua dute.

4.1 Erlazio-eragileak eta eragile logikoak

Erlazio-eragileek bi zenbaki alderatzen dituzte, eta alderaketaren enuntziatua egiazkoa edo gezurrezkoa denentz jakiteko erabiltzen dira, $5 < 8$ edo $3 == -1$, kasu baterako. Egiazko kasuan erkaketari esleturiko balioa 1 izango da, eta gezurrezko kasuan emaitza 0 izango da. Portaera hori eragile guztietan errepikatzen da. Izan ere, adibidez, sententzia bi egiazkoak baldin badira, Y(AND) eragile logikoak 1 zenbakia itzultzen du. Bai eragile logikoak, bai erlazio-eragileak, adierazpen matematikoetan erabil daitezke. Are gehiago, geroxeago erakutsiko denez, beste komando batzuekin batera eragin handia izan dezakete programaren fluxuan.

4.1.1 Erlazio-eragileak

Matlab-eko erlazio-eragileak 4.1 Taulak biltzen ditu. Ohar zaitez: "berdin" eragilea adierazteko, "=" ikur bi (hutsunerik gabe) idazten dira. Arian ere, "=" ikurra esleipen-ikurra da, eta haren jokabidea balioak esleitzea da. Modu berean, kontuan hartu behar da gainontzeko ikur bikoitzek ez dutela hutsunerik onartzen.

| Erlazio-eragilea | Deskribapena |
|------------------|----------------------|
| < | Txikiago. |
| > | Handiago. |
| <= | Txikiago edo berdin. |
| >= | Handiago edo berdin. |
| == | Berdin. |
| ~= | Desberdin. |

4.1 Taula: Erlazio-eragileak.

Segidan, erreparatu behar diren erlazio-eragileen gaineko xehetasun batzuk deskribatzen dira:

- Erkaketak eskalarren artean egiten direnean, erlazioa egiazkoa baldin bada, 1 balio *logikoa* lortzen da; osterantzean, 0 balio *logikoa*. Aintzat hartu balio *logikoez* ari dela, hau da, datu-tipo logikoaz, hain justu. Hurrengo adibidea exekutatzeko baldin bada, *Workspace*-n gordetako `z` aldagaia *logical*-tipokoa da, hots, -eta hau garrantzitsua da- erlazio-eragiketa baten emaitza emaitza logikoa da. Horrek ondorio garrantzitsuak ditu; adibidez, bektore edo matrize baten elementuak erauztea izan daiteke aplikazio interesgarrietako bat.

```
>> 5 > 8
ans =
     0          % 0 logikoa
>> x=3; y=-1;
>> z=x~=y
ans =
     1          % 1 logikoa
```

Goian `z` aldagai logikoari 1 balioa dagokio, `x` eta `y` desberdinak baitira, eta egiazko motakoa dela interpretatu behar da.

- Era berean, alderaketak dimentsio bereko bektore edo matrizeekin egiten baldin badira, berdinketa elementuz elementu egiten da. Irteera ere dimentsio bereko bektore edo matrizea da, kasu bakoitzaren arabera, 0 eta 1 zenbakiez betea:

```
>> A=[1 -2 3 6];
>> B=[0 -2 1 6];
>> C=A==B
C =
     0     1     0     1
```

Hor berriro ere, C bektore-aldagaia logikoa da, 0 zenbakiak gezurrezko-motakoak eta 1 zenbakiak egiazko motakoak dituen. Izan ere, imajinatu C aurreko adibidekoa:

```
>> D=[1 2 3 4 5 6] % bektore arrunt bat
D =
     1     2     3     4     5     6
>> D(C)
ans =
     2     4
```

Etsenpluan argi erakusten denez, azken irteeran 2 eta 4 zenbakiak agertzen dira. Portaera horren zergatia C bektore logikoan topatu behar da. Alta, C bektorearen lehenengo osagaia 0 da, hau da, gezurra; horregatik, D matrizeko lehenengo elementua ez da pantailan agertzen. Gisa berean, C bektorearen bigarren osagaia 1 da, hau da, egia, eta orain bai, D matrizeko bigarren elementua azaltzen da. Arrazoi beragatik 4 agertzen da, eta ez gainontzekoak. Jakin beharra dago Matlab-eko aldagai bat, zero edo bat zenbakiez betea, `logical` aginduari esker logiko-tipokoa bihurtu daitekeela; bestela, izan erne:

```
>> b=[2 4 6 8 10]; p=[1 0 0 0 1];
>> b(p)
??? Subscript indices must either be real positive
integers or logicals.
```

Errore-mezua saiheste aldera, p bektore logikoa definiturik helburua eskuratzen da:

```
>> b=[2 4 6 8 10]; p=[1 0 0 0 1];p=logical(p);
>> b(p)
ans =
     2    10
```

- Eskalar bat bektore edo matrize batekin alderatzen baldin bada, eskalarra bektoreko edo matrizeko elementu guztiekin konparatuko da, eta emaitza bektore edo matrize logikoa izango da, dimentsio berekoa:

```
>> a=3;A=[2 3 4; 2 -1 6; 0 4 1]
A =
     2     3     4
     2    -1     6
     0     4     1
>> B=a<A % matrize logikoa
ans =
     0     0     1
     0     0     1
     0     1     0
>> A(B)
```

```
ans =
     4
     4
     6
```

Azken exekuzioan A matrizean B matrize logikoa aplikatzen da, eta 1 zenbakiari dagozkion A matrizeko elementuak soilik itzultzen dira, hau da, nolabait esateko, egiazko balioak.

- Zenbakizko bektoreek eta matrizeek ez dute bektore eta matrize logikoekin zerikusirik. Izan ere, zenbakizko bektoreak ezin dira erabili beste bektore batzuetako elementuak atzitzeko. Dena den, bektore logikoak eragiketa matematikoetan erabil daitezke:

```
>> 3+(4<16)/2 % hemen 4<16-ren emaitza 1 da
ans =
     3.5000
>> 3+4<16/2 % hemen 3+4=7 eta 16/2=8 beraz, 7<8-ren emaitza
           egiazkoa da
ans =
     1
```

- Adibide honetan erakutsi nahi denez, ohartzekoa da adierazpen matematiko batean erlazio-eragileak eta eragile aritmetikoak batera agertuz gero eragiketa aritmetikoek (+, -, *, /) erlazio-eragileek baino lehentasun handiagoa dutela.

4.1.2 Eragile logikoak

Matlab-en erlazio-eragileak usatuz gero, aurreko azpisekzioan ikusi denez, datu-tipo logikoak agertzen dira. Are gehiago, Matlab-en badira aurredefinituriko eragile logikoak, 4.2 Taulan biltzen direnak, eta alderaketak egiteko aplikatu daitezkeenak. Emaitzeen aldetik, irteerak, berriz, logikoak dira.

| Eragile logikoak | Izena | Deskribapena |
|------------------|----------|---|
| &, Adib. A&B | Eta(AND) | A eta B. Bi elementu behar dira. Biak egiazkoak badira, emaitza egiazkoa da; bestela, emaitza faltsua da (0). |
| , Adib. A B | Edo(OR) | A edo B. Bi elementu behar dira. Bata edo bestea egiazkoak badira, emaitza egiazkoa da; bestela, emaitza faltsua da (0). |
| ~, Adib. ~A | Ez(NOT) | Ez A. Elementu bat behar da. Elementuaren ukapena da, hau da, A faltsua bada, egiazkoa (1), eta A egiazkoa bada, faltsua (0). |

4.2 Taula: Aurredefinitutako eragile logikoak.

Eragile logikoen ezaugarri nagusien artean hauek gailentzen dira:

- Eragile logikoek zenbakiak erabiltzen dituzte. Gainera, ez-nulua den zenbaki bat egiazkoa da, eta zero zenbakia beti-beti faltsua da.
- Eragile logikoak eragile aritmetiko gisa erabili ahal dira. Areago, lehen deskribatu denez, emaitzak matrize bateko elementuak ateratzeko ere usa daitezke.
- Eragile logikoak eskalarrekin eta bektoreekin edo matrizeekin erabil daitezke. Elementu biak eskalarrak izanez gero, emaitza 1 edo 0 logikoa izango da. Matrizeen kasuan, elementuz elementuko berdinketak egingo dira, eta emaitza zero eta bat zenbaki logikoez baterikoa izango da.

Aurrekoa hobeto erakusteko, hona hemen adibide zenbait:

```
>> 8&-1
ans =      % irteera logikoa, ez zenbakizkoa
      1
>> 8&0
ans =
      0
>> 8|0
ans =
      1
>> ~-2
ans =
      0
>> ~0
ans =
      1
>> a=10*((12&0)+(~0)+(0|5)) % adierazpen matematikoa: 10*(0+1+1)
a =
      20

>> x=[0 3 4 1]; y=[0 2 0 -3];
>> x&y
ans =
      0      1      0      1
>> x|y
ans =
      0      1      1      1
>> x+y
ans =
      0      5      4      -2
>> ~(x+y)
```

```
ans =
     1     0     0     0
```

Horrez gain, Matlab-ek baditu ikusi berri diren eragile logikoen baliokideak diren funtzio batzuk:

```
and(A,B)  A&B
or        A|B
not(A,B)  ~A
```

Horratik ere, baliokidetasunak baliokidetasun, alderaketa logikoak errazki ete beste zentzu batean eratzeko, badira aurrez definituriko beste funtzio batzuk, eta interesgarriak dira oso:

- `any(A)`: A bektorea baldin bada, eta A bektoreko elementu ez-nuluren bat existitzen baldin bada, irteera 1 (egiazkoa) da, eta osterantzean 0 (gezurrezkoa). Halaber A matrizearen kasuan eta behean igartzen denez, erkaketa zutabez zutabe gauzatzen da bektoreen jarraibide bera mantenduz.

```
>> A=[ 0 0 2; 0 1 -3; 0 2 9]
A =
```

```
     0     0     2
     0     1    -3
     0     2     9
```

```
>> any(A)
ans =
     0     1     1
```

- `all(A)`: A bektore kasuan, A-ko elementu guztiak egiazkoak (ez-nuluak) badira, 1 logikoa itzultzen da, edo, bestela, 0 zenbaki logikoa. Halaber A matrizea baldin bada, eragileak A-ko zutabeak aztertzen ditu, eta portaera berbera jarraitzen du:

```
>> x=[0 3 4 1];
>> any(x),all(x)
ans =
```

```
     1
```

```
ans =
     0
```

```
>> A=[ -2 0 2; 0 1 -3; 0 2 9]
A =
```

```
    -2     0     2
     0     1    -3
     0     2     9
```

```
>> all(A)
ans =
     0     0     1
```

- `xor(A,B)`: 0 eskusiboa da. Elementuetako bat egiazkoa baldin bada eta bestea gezurrezkoa, irteera 1 zenbaki logikoa da; osterantzean, 0 da irteera.

```
>> xor(-2,0)
ans =
     1
>> xor(3,1/2)
ans =
     0
```

- `find(A)`: A bektorea bada, ez-nuluak diren elementuen indizeak edo posizioak agertzen dira. A matrizea bada, zutabez zutabe eta lehenengo zutabetik hasita, A-ko elementu ez-nuluaren posizioak lortzen dira.

```
>> x=[0 1 2];
>> find(x)
ans =
     2     3
>> A=[ 0 0 2; 0 1 -3; 0 2 9]
A =
     0     0     2
     0     1    -3
     0     2     9
>> find(A)
ans =
     5
     6
     7
     8
     9
```

Aurreko adibidean lehenengo elementu ez-nulua 1 da, eta bosgarren posizioan aurkitzen da.

- `find(A>d)`: A bektorea baldin bada, d baino handiagokoak diren A-ko elementuen indizeak itzultzen dira. Kontuan hartu > ikurraren ordez beste edozein erlazio-eragile idatz daitekeela.

```
>> x=[-1 1 2]; m=find(x>0)
m =          % bigarren eta hirugarren posizioak
     2     3
>> x(m)=[9 9]
x =
    -1     9     9
>> A=[ 0 0 2; 0 1 -3; 0 2 9]
A =
```

```

    0    0    2
    0    1   -3
    0    2    9
>> n=find(A==0)
n =
    1
    2
    3
    4
>> A(n)=[10 10 10 10]
A =
   10    10     2
   10     1    -3
   10     2     9

```

Azkenik, aipatzekoa da Matlab-en askotariko erabileretarako `is*` motako funtzio logikoak daudela:

| | |
|---------------------------|--|
| <code>ischar(A)</code> | Egia A karaktere-katea (<i>string</i>) bada |
| <code>isempty(A)</code> | Egia A elementu hutsa bada |
| <code>isequal(A,B)</code> | Egia A eta B elementuz elementu berdinak badira |
| <code>isfinite(A)</code> | Egia Ako elementu oro finituak badira |
| <code>isinf(A)</code> | Egia Ako elementu guztiak infintituak badira |
| <code>islogical(A)</code> | Egia A matrize logikoa bada |
| <code>isnan(A)</code> | Egia A-n NaN (<i>Not a Number</i>) agertzen bada |
| <code>isreal(A)</code> | Egia A matrize erreala baldin bada |

Horrelako `is*` funtzio gehiago egon badaude, eta, guztiak begietaratzeko, aski da `doc is` idaztea. Goian zerrendatutako en artean, `isnan`-ek berezitasun aipagarri bat dauka: `x==NaN` testak beti 0 (gezurra) itzultzen du, `x NaN` izanda ere. Honatx horri guztiari buruzko adibideak:

```

>> Izen='Matlab'
Izen =
Matlab
>> ischar(Izen)
ans =
    1
>> A=[];
>> isempty(A)
ans =
    1
>> isempty([1 2])
ans =
    0
>> isempty([0 0])

```

```
ans =
    0
>> isfinite(Inf)
ans =
    0
>> isfinite(-10)
ans =

    1
>> x=[2:0.1:10];y=[2:0.1:10 10.1];
>> isequal(x,y)
ans =
    0
>> isfinite(x) % elementuz elementuko erkaketa
ans =
Columns 1 through 21
    1    1    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1    1    1    1
Columns 22 through 42
    1    1    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1    1    1    1
Columns 43 through 63
    1    1    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1    1    1    1
Columns 64 through 81
    1    1    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1
>> isinf([2 inf 3])
ans =
    0    1    0
>> isnan([2 NaN 3])
ans =
    0    1    0
>> isreal([1+j 2 -3*j]) % lehenespenez, hemen j unitate irudikaria da
ans =
    0
```

4.2 Fluxuaren kontrola

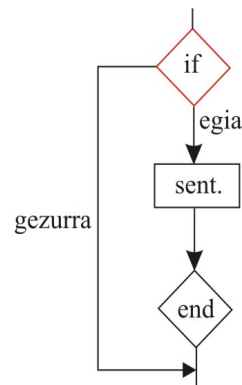
Jadanik aipatu denez, Matlab programa zenbakizko *script*-ak egiteko ere pentsatuta dago, ondo pentsatuta ere. Orobat, programazio-lengoaia gisa, beste lengoaien aldean, baditu hainbat aukera, eta ez hain zailak. Lehenik eta behin, jakin beharra dago programa baten exekuzioa ez dela lerroz lerro zertan egikaritu. Hori dela eta, jarraian inplementazioaren fluxu arrunta aldatzeko eta manipulatzeko zenbait agindu deskribatuko dira.

4.2.1 Erabaki-egiturak

Matlab-eko erabaki-egiturak garrantzitsuenetarikoak dira, hainbat programazio-lengoaiatan bezala, eta ordezkari nagusiak `if` eta `switch` komandoak dira, segidan deskribatzen direnak:

- `if`: Baldintzako agindu honen era errazena hauxe da:

```
if adierazpena
sententziak
end
```



non, adierazpeneko elementuak ez-nuluak baldin badira, hots, programa baten fluxuan adierazpena egiazkoa baldin bada, sententzia-multzoa exekututzen den. Esaterako, segidako kodean, bigarren instrukzioan, hain zuzen, erabaki-egitura bat aurkitzen da, eta `x` aldagaia `y` baino handiagoa denez, haien balioak trukutzen dira. Erreparatu baldintza erabakitzekeo erlazio-eragile bat usatu dela:

```
>> x=1;y=-3;
>> if x>y
tmp=y;
y=x;
x=tmp;
end
>> x
x =
    -3
>> y
y =
     1
```

Halaber, jakin behar da `if` aginduaren lerro berean idatz daitekeela, eta, horretarako, ”,“ bereizlea erabiltzen dela sententzia-multzoak bereizteko:

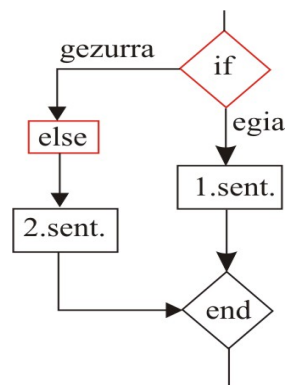
```
>> x=225; if x>0, x=sqrt(x), end
>> x=
    15
```

Bestalde adierazpena faltsua izan arren, `else` aukerari esker badago beste sententzia-sail bat exekutatzeko abagunea, non:

```

if adierazpena
1. sententziak
else
2. sententziak
end

```



Hau da, lehenengo adierazpena egiazkoa baldin bada, `if` eta `else` bitarteko 1. sententziak egikaritzen dira; osterantzean, gezurrezko kasuan, fluxuaren kontrola 2. sententzietara pasatzen da. Segidako erakusgaia adibidetzat hartu:

```

>> if 2^e > e^2
disp('2^e handiagoa da') % karaktere-katea edo string
else
disp('e^2 handiagoa da') % karaktere-katea edo string
end
e^2 handiagoa da

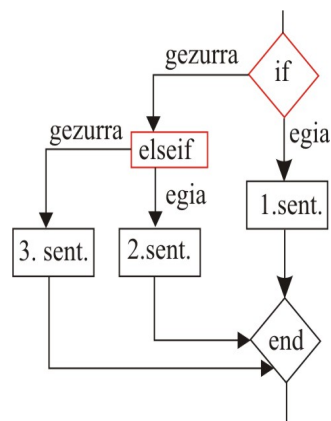
```

Areago, egitura konplexuagoa izan eta osa daiteke. Izan ere, `elseif` aginduari esker, bigarren sententzia-multzoko gezurrezko kasuan test desberdina proposa daiteke; berriro ere, egiazko edo gezurrezko adarrak biltzen dituztenak:

```

if 1. adierazpena
1. sententziak
elseif 2. adierazpena
2. sententziak
else
3. sententziak end

```



Ahalezko egituraren konplexutasuna agerian adieraztearren, erreparatu, era berean, `if` aginduarekin batera nahi beste `elseif` bloke habiaratu daitezkeela, hau da, erabaki anizkoitzak ematen dituztenak osa daitezkeela.

```

>>x=0;
>> if x<0
    f=x^2;
    disp('x negatiboa')
elseif x<sqrt(pi)

```

```

    f=sin(x^2);
    disp('x [0,sqrt(pi)] tartean')
elseif x<2*sqrt(pi)
    f=(x-sqrt(pi))^2;
    disp('x [sqrt(pi) 2*sqrt(pi)] tartean')
else
    f=pi*cos(x^2);
    disp('x [2*sqrt(pi) infinitua] tartean')
end
x [0,sqrt(pi)] tartean % hau if blokearen emaitza da

```

- `switch`: Agindu hau erabaki-egitura bat implementatzeko beste hautabide bat da. Sententzia multzo baten artean, zenbait sententzia soilik exekutatzeko aukera ematen du. Aitortu behar da, funtsean, komando hau eta `if` habiratuak oso antzekoak direla:

```

switch adierazpena
case 1.balioak
1. sententziak
case 2.balioak
2. sententziak
.....
.....
otherwise
azken sententziak
end

```

Adierazpenari dagokionez, eskuarki, adierazpena eskalarra edo karaktere-katea izan arren, baliteke adierazpen matematiko balioztagarria izatea. Nolahani ere, behin adierazpenaren balioa ezaguturik, `case`-aren arabera, adierazpenarekin bat datorren instrukzio-multzoa baino ez da egikaritzen. Baina adierazpenaren balioa ezein kasutan bat etorri ezik, `otherwise` komandoaren osteko sententziak exekutatzen dira.

- Ohartu `otherwise` agindua aukerakoa dela. Izan ere, bat-etortzeak aurkitzen ez badira eta `otherwise` idatzirik ez badago, sententzia bat bera ere ez da exekutatu, eta fluxua `switch` blokeko ondorengo sententzietara pasatzen da.
- Halaber, `case` sententzia batek adierazpenarekin erkatzeko balio bat baino gehiago izan ditzake; horiek guztiak deklaratzeko, nahikoa da giltza artean eta komaz bereizturik idaztea:

```

>> x=input('sartu zenbaki erreal bat ')
% Erabiltzaileak teklatutik zenbaki erreal bat sartu

sartu zenbaki erreal bat -1
x =
    -1

```

```
>> switch x % erabaki-egitura hasten da
    case {inf,-inf}
        disp('Plus minus infinitua')
    case 0
        disp('Zero')
    otherwise
        disp('Ez-nulua eta finitua')
    end
Ez-nulua eta finitua % hau emaitza da
```

- Gogoan izan switch blokean, harik eta case batekin lehenengo kointzidentzia lortu arte, bilaketa egiten dela. Gero, sententziak exekutatu ostean, fluxuak switch-etik at jarraitzen du. Esaterako, C lengoaiak ez du jokabide hori, han zenbait case blokeak egikari baitaitezke. Horrenbestez, xehetasun hori, Matlab-en jazotzen dena, kontuan hartzekoa da.

```
>> x=[-3 2 4 -5 1];
>> p=inf; % kasu infinitua
>> switch p
    case 1
        y=sum(abs(x)) % norma 1
    case 2
        y=sqrt(x'*x) % norma 2
    case inf
        y=max(abs(x)) % norma infinitua
    otherwise
        % hurrengo errore-mezu bat baino ez da
        error('p 1 edo 2 edo inf izan behar du')
    end
y =
     5 % kasu honetan irteera norma infinitua da
>> p=3; % orain otherwise-en ondokoa exekutatuko da
>> switch p
    case 1
        y=sum(abs(x))
    case 2
        y=sqrt(x'*x)
    case inf
        y=max(abs(x))
    otherwise
        error('p 1 edo 2 edo inf izan behar du')
    end
??? p 1 edo 2 edo inf izan behar du % kasu honetan emaitza
```

Garbi dago aurreko kasuak, switch-en ordeztu, if aginduarekin ere era daitezkeela. Hemen ariketa interesgarri bat dago, dudarik ez.

4.2.2 Begiztak

Begiztak programaren fluxua kontrolatzeko beste metodo bat dira. Begiztek komando baten edo askoren exekuzioa behin eta berriro errepikatzea ahalbidetzen dute. Errepikapen bakoitzari iterazio edo urrats deitzen zaio. Matlab-en begizta-mota desberdin bi definitu ahal dira: batetik, `for`-end egitura, non iterazio kopurua begiztaren hasieratik definiturik dagoen, eta bestetik, `while`-end egitura, non hasieran iterazio kopurua ez den ezagutzen, baldintza baten menpe baitago. Hala ere, bai batean, bai bestean, edozein unetan iterazioak egiteari utziaz, `break` komandoaren bitartez irten daiteke begizta horietatik. Berebat, esate baterako, `continue` aginduari esker, hurrengo iteraziora pasatzea ere posible da, interesatzen baldin bada, jakina.

- Lehenengoaz bezainbatean, `for` aginduaren egitura orokorra hau da:

```
for k = h:u:a
    sententziak
end
```

non h elementua k -ren hasierako balioa den, u urratsa eta a amaiera. Hots, u urratsa positiboa dela joz gero, hau da k -ren iterazio-multzoa:

```
k = h
k = h+u
k = h+2u
k = h+3u
.....
k ≤ a
```

Azken lerroan erakusten denez, k -ren balioa a -rena baino handiagoa denean, iterazioak egiten bertan behera bukatzen da.

- u urratsa negatiboa izan daiteke; kasu horretan, k -ren balioa a -rena baino txikiagoa denean, iterazioak bertan behera lagako dira. Esate baterako, $k=25 : -5 : 10$ idatziz gero, begiztan $k=25, 20, 15, 10$ balioetarako bost iterazio izango dira, ordena hori mantenduz.
- h eta a berdinak baldin badira, iterazio bat baino ez da izango, edozein urrats izanda ere.
- $h > a$ eta $u > 0$, edo $h < a$ eta $u < 0$ gertatzen baldin badira, begizta ez da inondik inora exekutatu.
- Kontuan hartu behar da $k-k$ ez duela bakarrik eskalarra izan behar. Aldiz, k aldagaiak balio zehatz batzuk hartu ahal ditu, urratsik gabe edo bai. Adibidez, $k=[1 \ 2 \ 3 \ 6 : 10]$ non k -ren balioak 1, 2, 3, 6, 7, 8, 9, 10 izango diren. Erreparatu kasu honetan ez dagoela urratsik.
- Begiztaren barruan k -ren balioa birdefinitzen baldin bada, arazoak ager daitezke, batik bat, begiztaren funtzionamenduaren aldetik; horrenbestez, ez abusatu egokiera horretaz.

- Nola ez, for aginduak habiaratu daitezke, betiere bakoitzak bere end duelarik. Horrela, egitura konplexuagoak osa daitezke.
- Automatikoki k aldagaiaren balioa ez da pantailan ateratzen, soilik sententzia gisa izkiriartzen baldin bada agertuko da balioa. Ezaugarri hori interesgarria izan daiteke programa bat arazteko.
- Iterazioak amaitzen direnean, k aldagaiari azken pausuan hartutako balioa dagokio, eta beste adierazpen batzuetan erabil daiteke, inolako arazorik gabe.

Jarraian adibide batzuen bitartez for-en zenbait erabilera erakutsiko dira. Lehenengoan, $\sum \frac{1}{n^2}$ seriearen batura partzial baten balioa, lehenengo hogeita bost batugaiena, kalkulatzeko da:

```
>> s=0;
>> for i=1:25
    s=s+1/i^2; % gai orokorra
end
>> s
s =
    1.6057
```

Bigarrenean, begizta balio zehatz batzuetan baino exekutatzeko ez delako etsenplu bat:

```
>> for x=[pi/6 pi/4 pi/3]
    disp([x sin(x)])
end
    0.5236    0.5000

    0.7854    0.7071

    1.0472    0.8660
```

- Bigarrenaz bezainbatean, while aginduaren egitura orokorra, beharbada for egiturarena baino zerbait malguagoa, hau da:

```
while baldintzazko adierazpena
    sententziak
end
```

Egitura honetako lehenengo lerroan baldintzazko adierazpena azaltzen da. Programa lerro horretara iristen denean, delako adierazpena aztertzeke unea da. Gezurrezkoa (0) baldin bada, begiztako instrukziorik ez da exekutatu. Ohera, baldintzazkoa egiazkoa baldin bada, begiztaren barruko sententziak egikaritzen dira. Behin iterazioa buruturik, while agindura atzeratzen da, eta, berriro ere, gisa berean, baldintzazkoa aztertzen da. Era horri jarraikiz, adierazpen faltsua izan bitartean, iterazio-prozesuak segitu egingo du.

- Baldintzazko adierazpenean aldagai baten balioa agertu behar da, bederik.
- Baldintzazko adierazpeneko aldagaien balioa aldatu ezean, begizta infinitua izan daiteke, adibide honetan erakusten den legez:

```
x=1;
>> while 1 % 1 beti egia da
xmin=x;
x=x/20;
end
```

- Dena den, noiznahi iteratzeari uzteko break agindua eskura dago beti.

```
>> while 1
xmin=x;
x=x/20;
if x<=0.001, break, end
end
>> xmin
xmin =
    0.0025
```

Adibide honetan xmin-en azken balioa 0.0025 da, ezen $x=x/20$ zenbakia 0.001 baino txikiagoa da eta, break lagun, begiztatik ateratzen da.

- Erroreak direla eta ez direla, edozein unetan programa begizta infinitu batean sartuta baldin badago, erabiltzaileak Ctrl+C edo Ctrl+Break laster-tekleen bitartez geldiaraz dezake exekuzioa.

Hurrengo adibidean e^x funtzioaren Taylor-en garapenari hurbilketa egingo zaio, non $e^x = \sum \frac{x^n}{n!}$ formula ezaguna aplikatuko den:

```
>> x=input('Sartu x-en balioa: ');
Sartu x-en balioa: 0.1
>> n=1;an=1;Batura=an;
>> while abs(an)>=0.0001 & n<=50 % eragile logiko baten erabilera
an=x^n/factorial(n);
Batura=Batura+an;
n=n+1;
end
>> Batura % batura hurbildua
Batura =
    1.1052
>> exp(0.1) % balio doia
ans =
    1.1052
```

Aurreko adibideko begiztan, an-en balio absolutua handiagoa edo berdin 0.0001 eta n txikiagoa edo berdin 50 diren bitartean jarraitzen da iteratzen. Ohartu n-ri unitate

bat erantsiz, batugaien kopurua gorantz doala eta, beraz, baldintzako adierazpena ere aldatzen dela. Bestalde, jabetu 0.1 puntuko hurbilketa esponontzialaren zenbatespena eta zenbaki doia lau hamartarrekin bat datozela.

Azpireskzio hau bukatu orduko, aipatzekoa da `continue` agindua. Komando horren aplikazioa, bai `for`, bai `while` begiztetan, uneko iterazioa geldiaraztea da, eta nolabait, begiztaren hurrengo iterazioa egitera behartzea. Hona hemen adibide adierazgarri eta sinple bat:

```
>> for i=1:10
if i<5
continue
end
disp(i)
end
    5 % emaitzak
    6
    7
    8
    9
    10
```

non bost eta hamar bitarteko zenbakiak azaltzea lortzen den. Askotan, `continue` sententzia-bloke bat ez egikaratzeko `if` baten barruan joaten da.

5. Kapitulua

Datuen maneia

Edozein esperimentutan datuak lortzen dira, diskretuak gehienbat, eta askotan kopuru ino-break izaten dira. Hori dela eta, garrantzitsua da tresna egokiak izatea datuen kudeaketa arrazionala egiteko. Kapitulu honen asmoa da, batetik, erakustea programa baten exekuzioan erabiltzaileak, pantailan edo fitxategi batean, datuak nola sartu eta atera ditzakeen; bestetik, datu-inportazioa eta esportazioa landuko da, eta Matlab-ek Excel programarekin daukan konexio zuzena deskribatuko da.

5.1 Sarrerako eta irteerako datuak

Sekzio honetan datuak sartzeko eta ateratzeko askotariko erak erakutsiko dira. Hori guztia gauzatzeko, bide bat baino gehiago dago: batzuetan, erabiltzaileak datuak pantailatik bertatik kontrolatuko ditu, eta, bestetuetan, datuak fitxategi batetik irakurriko edo fitxategi batera gordeko dira.

5.1.1 Erabiltzaileen sarrerak

Erabiltzaileak, input komandoari esker, noiznahi datuak sar ditzake. Alean ere, gero egiaztatzen denez, honen egikaritzean programan edo Matlab-eko *prompt*-en geldiune bat suertatzen da, harik eta balioak, errealak, bektoreak, matrizeak edo karaktereak sartu arte. Horren ostean, datuak gorde eta beste adierazpen batzuetan erabil daitezke:

```
>> x=input('Eman hasierako puntua: ')
Eman hasierako puntua: 25
x =
    25
>> y=input('Eman hasierako puntuak: ')
Eman hasierako puntuak: [1 2 3; 4 5 6; 7 8 9]
y =
     1     2     3
     4     5     6
     7     8     9
```

Halaber, input komandoaren erabilera zabalagoa da eta karaktere-kate bat gordetzeko bitara erabil daiteke; alde batetik, zenbakiak legez, baina komatxo sinpleen artean,

```
>> k=input('Sartu izena: ')
Sartu izena: 'Ander'
k =
Ander
```

eta orain k karaktere-aldagai da, edo bestetik, input aginduaren idazkeran karaktere bat sartuko delako zehaztasuna idatzi:

```
>> k=input('Sartu izena: ','s')
Sartu izena: Ander
k =
Ander
```

Orain erreparatu, 's' karaktere-atributua (*string*) idatzi dela. Zernahi gisaz, karaktereak zuzenean edo komatxo sinple bidez sar daitezke.

Bestalde, *M script* baten exekuzioa keyboard aginduaren bitartez geldi daiteke, eta kontrola *Aginduetarako Leihora* pasatzen da. Lehenik eta behin, *prompt*-en moldea aldatzen dela igartzen da. Bertan aldagaien balioak edo beste komando batzuk erabili eta sar daitezke, eta, nahi denean, *return* aginduari esker kontrola *script*-era pasatzen da. Imajinatu "fitxa" izeneko fitxategia, sententzia hauek dituen:

```
1 x=21;
2 y=0;
3 z=x+y
4 keyboard
5 z=x+y
```

Fitxategia exekutatu ostean, eta hirugarren lerroa iristen denerako, z-ren balioa 21 da. Segidan, fitxategitik irten eta *Aginduetarako Leihoan* aldaketak egin daitezke. Suposatu bertan y-ri -11 balioa esleitzen zaiola eta *return* idatzi eta sakatzen dela. Berrito ere, programari kontrola itzultzen zaio, eta bosgarren lerroa exekutatzen da, orain z-ren irteera 10 delarik.

```
>> fitxa % fitxategiaren izena
z =
    21
K>> y=-11 % erreparatu prompt-en moldea aldatu dela
y =
   -11
K>> return % orain fitxategiko z=x+y eragiketa egiten da
z =
    10
```

Halatan, programa bat araztu beharrez, artxibotik kanpo joan ahal izateko eta hutsegiteak detektatzeko interesgarria izan daiteke.

5.1.2 Irteerak pantailatik

Matlab-ek pantailatik automatikoki agindu batzuen irteerak itzultzen ditu, esaterako, aldagai bati balio bat esleitzen zaionean eta `Enter` sakatzen denean. Nolanahi ere, Matlab-ek komando batzuk ditu, irteera osoagoak, formatuari buruzkoa tarteko, lortzeko erabil daitezkeenak. Hemen, batik bat, agindu `bi`, `disp` eta `fprintf`, komentatuko dira.

Lehenengoari dagokionez, irteera beti pantailatik izaten da, eta bigarrenarekin, datuak pantailan erakuts daitezke edo fitxategi batean gorde daitezke. Nabarmentzekoa da, biak Matlab-eko *Aginduetarako Leihoan* edo *script* batean erabil daitezkeela.

- Lehenik, aldagaiaren izena eta gero = ikurra tekleatu beharrean, `disp` aginduak, u-neko formatua errespetatuz, aldagaien balioak bistaratzen ditu.

```
>> a=input('sartu lehenengo nota: ')
sartu lehenengo nota: 7.2
a =
    7.2000
>> b=input('sartu bigarren nota: ')
sartu bigarren nota: 3.1
b =
    3.1000
>> disp('') % lerro huts bat uzteko

>> disp('Batez bestekoa: ')
Batez bestekoa:
>> bab=mean([a b]) % aurredefinituriko funtzioa
bab =
    5.1500
```

Maiz, datuak taula-forman erakusteko beharra egoten da. Kasu horretarako, aurkeztu nahi diren datuak dituen matrizea idazten da, eta gero `disp`-ren bitartez bistaratzen dira taula gisa:

```
urte=[2000 2001 2002 2003 2004];
pop=[100 120 130 110 105];
taula(:,1)=urte;
taula(:,2)=pop;
disp('        urteak        populazioa')
disp('')
disp(taula) % taula izeneko datu-matrizea
```

Goiko "taula" izeneko *script*-a exekutatzeko baldin bada, taularen goiburukoa eta datuak denak batera taularatzen dira:

```
>> taula % script-en izena
        urteak        populazioa
```

| | |
|------|-----|
| 2000 | 100 |
| 2001 | 120 |
| 2002 | 130 |
| 2003 | 110 |
| 2004 | 105 |

- Gehienbat programen irteerak, testua edo datuak, pantailatik begiratzeko edo fitxategi batean gordetzeko, `fprintf` agindua erabili ohi da. Hemen, `disp` komandoan ez bezala, `fprintf`-ren bidez, erabiltzaileak irteerari formatu zehatza eman diezaioke. Horrezaz gain, lerro berean testua eta zenbakizko balioak tartekatu daitezke, baita formatua aldatu ere. Komando horren osaketa luzea eta korapilatsua izan daitekeenez, erabilera astiro-astiro erakutsiko da:

- Aplikazio xumeena testu-mezuak bistartzekoa da,

```
fprintf('karakterek')
```

```
esaterako,
>> fprintf('Ezin da burutu ')
Ezin da burutu >>
```

Kasu honetan *prompt*-i erreparatzen baldin bazaio, agerikoa da bera eta karaktere-katea lerro berean daudela. Jakin behar da `fprintf` aginduarekin lerro berri bat has daitekeela. Hori burutzeko, lerro berriko karakterearen aurrean `\n` ikurrak txertatzen dira.

```
>> fprintf('Ezin da burutu. \n Beste egokiera bat daukazu. \n')
Ezin da burutu.
Beste egokiera bat daukazu
```

Irteerari dagokionez, orain bi lerro desberdin lortzen dira. Askotan `\n` karakterari ihes-karakterea deitzen zaio, eta irteera kontrolatzeko karaktereetako bat baino ez da. Horrezaz gain, kate baten barruan beste ihes-karaktere batzuk txertatu daitezke, hala nola:

```
\b ezabatzeko karakterea
\t tabulazio horizontala
```

Programa batean `fprintf` bat baino gehiago agertzen baldin badira, irteera jarraitua da, hau da, ez dago lerro-jauzirik:

```
fprintf('Egiaztatu datuak, mesedez. ')
x=6; y=3;
fprintf('Exekutatu programa beranduago.')
```

Hala, *script* hori egikaritzean, `fprintf` aginduen artean beste instrukzio bat egon arren, *Aginduetarako Leihotik* hitzez hitz hauxe itzultzen da:

```
Egiaztatu datuak, mesedez.Exekutatu programa beranduago.
```

Efektu gogaikarri hori ekiditeko, \n karakterea karaktere-katearen aitzinean idatzi beharko zen.

- fprintf-ren beste aplikazio bat da datuak eta testua dena batera begietaratzea, eta, kasu baterako, honako sintaxi hau dagokio:

```
fprintf('Testua %-5.2f testu gehigarria',aldagai baten izena)
```

non % ikurrak testuaren barruko aldagaiaren balioa txertatuko den lekua adierazten duen. Gainera, horren ondoan eta gorriz seinaturik, kasu honetan adibide bat baino ez delarik, zenbaki batzuk eta letra bat aurkezten dira. Espresuki, hauek dira formatu-elementuak deritzenak, eta haien zati nagusiak hauek dira:

-5.2f

Lehenengoa bandera (*flag*) bat da eta guztiz aukerakoa da:

| Bandera | Deskribapena |
|------------|---|
| - ikurra | Eremuaren barruan zenbakiaren ezker-lerrokadura |
| + ikurra | Zenbakiaren aurrean + edo - bistaratzen da |
| 0 zenbakia | Zenbakia eremua baino txikiagoa baldin bada, zeroak eransten dira |

Gero bigarren zatia eremuaren zabalaren espezifikazioari eta doitasunari dagokie, 5.2 kasu honetan, eta berriro ere aukerakoa da. Lehenengo zenbakia, 5, eremuaren zabalera da, eta bisualizazioan digitu kopuru txikiena finkatzen du. Zenbakia eremu-zabalera baino txikiagoa baldin bada, zenbakiaren aurrean zeroak edo hutsuneak erantsiko dira. Bigarren zenbakia, 2, doikuntzari dagokio, eta koma dezimalaren ondoko digitu kopurua zehazten du.

```
>> x=-21.12345;
>> fprintf('Emaitzaren balioa %+6.4f da \n',x)
Emaitzaren balioa -21.1234 da
>> fprintf('Emaitzaren balioa %+6.3f da \n',x)
Emaitzaren balioa -21.123 da
>> fprintf('Emaitzaren balioa %+6.2f da \n',x)
Emaitzaren balioa -21.12 da
```

Etsenplu honetan eremuaren zabalera 6 da, eta hamartarren kopurua lautik bira aldatzen da; horrenbestez irteeren aurrean atera kontuak.

Azken formatu-elementua, f, formatu-bihurketari dagokio. Aurrekoak ez bezala, elementu hori nahitaezkoa da, eta segidan, gehien erabiltzen diren bihurketa-karakterek zerrendatzen dira,

| | |
|---|---|
| e | Minuskulaz notazio esponentziala (1.789e + 001) |
| E | Maiuskulaz notazio esponentziala (1.789E + 001) |
| f | Koma finkoko notazioa (17.89) |
| g | e edo f notazioei dagokien formatu laburra |
| G | E edo f notazioei dagokien formatu laburra |
| i | Zenbaki osoa (17) |

eta notazio hori jarraituz, zenbait aplikazio aurkezten dira:

```
>> fprintf('Emaitzaren balioa %+4.2f da \n',x)
Emaitzaren balioa -21.12 da
>> fprintf('Emaitzaren balioa %+4.2g da \n',x)
Emaitzaren balioa -21 da
>> fprintf('Emaitzaren balioa %+4.2G da \n',x)
Emaitzaren balioa -21 da
>> fprintf('Emaitzaren balioa %+4.2e da \n',x)
Emaitzaren balioa -2.11e+001 da
>> fprintf('Emaitzaren balioa %+4.2E da \n',x)
Emaitzaren balioa -2.11E+001 da
>> fprintf('Emaitzaren balioa %i da \n',3)
Emaitzaren balioa 3 da
```

Halarik ere, etsenplu orokorragoa zenbat-nahi zenbakizko balio txertatzea da. Hori egiteko, nahikoa da testuan zenbakiak agertu behar duten posizioetan % eta ondoan dagozkien formatuak teklatzea:

```
fprintf('testua....%g....%f....testua....%e'
        ,1.aldagai,2.aldagaia,3.aldagaia)
```

Ezkerretik eskumara, lehenengo aldagaia lehenengoa da agertzen, eta lehenengo formatua dagokio. Ikusi adibide hau:

```
>> x=1;y=0.010;z=3.2113
>> fprintf('x=%i, y=%3.2f, z=%4.3e \n',x,y,z)
x=1, y=0.01, z=3.211e+000
```

Edozelan ere, fprintf aginduaren erabileraren gaineko iradokizun batzuk aintzak hartu behar dira, eta segidan zehazten dira:

- * Sintaxiaren aldetik, testu baten barruan komatxo sinplea txertatzeko, komatxo bikoitza teklatu behar da.
- * Orobat, fprintf aginduak bektore-notazioa onartzen du; osterantzean, erreparatu adibide honi:

```
>> x=2:5;
>> y=log(x);
>> T=[x;y]; % Bi errenkada
>> fprintf('Zenbakia %i baldin bada, bere logaritmoa
           %f da \n',T)
```

```
Zenbakia 2 baldin bada, bere logaritmoa 0.693147 da
Zenbakia 3 baldin bada, bere logaritmoa 1.098612 da
Zenbakia 4 baldin bada, bere logaritmoa 1.386294 da
Zenbakia 5 baldin bada, bere logaritmoa 1.609438 da
Horretan, nabarmena da T matrizeko elementuen irakurketa zutabez zutabe gauzatzen dela.
```

- Berebat, `fprintf` komandoaren azken aplikazioa zera da, programa batean sortutako irteera zehaztuak fitxategi batean sartzea. Ondorengo azpisekzioan, beste komando batzuekin batera haren aplikazioa azaltzen da.

5.1.3 Sarrerak eta irteerak fitxategi batean

Fitxategi bateko datuen kudeaketa adibideekin erakutsiko da. Ataza horretarako, `fopen`, `fclose`, `fprintf` eta `fscanf` aginduak erabiltzen dira. Lehenik, fitxategi bat manipulatu orduko, `fopen` komandoarekin ireki behar da. Gogoan izan fitxategi berri huts bat sortzen dela edo dagoeneko existitzen den fitxategia irekitzen dela, gero harekin lan egiteko erabiltzeko dena. Sintaxia hau da:

```
fid=fopen('fitxategiaren izena','baimenak')
```

non `fid` artxibo-identifikadorea izeneko aldagaia den. Hark irekitako fitxategiari dagokion eskalar bat gordetzen du. Kontuan izan idazkeran artxiboaren izena bere luzapenarekin batera idazten dela. Baimenei dagokienez, kode batzuek sistemari artxiboa nola ireki behar den esaten diote. Gehienetan maizago erabiltzen diren baimenak hauek dira:

- 'r' Fitxategia irakurtzeko zabaltzen da. Balio lehenetsia.
- 'w' Fitxategia idazteko irekitzen da. Fitxategia existitzen baldin bada, edukia ezabatuko da; osterantzean, hutsik sortuko da.
- 'a' 'w' legez, baina datuak fitxategiaren amaieran erantsiko dira. Horrenbestez, jadanik existitzen diren datuak ez dira ezabatuko.

Behin artxiboa zabalik dagoela ziurtaturik, datuak bertatik idatzi edo irakurri ahal izango dira.

- Fitxategi batean datuak idazteko, `fprintf` agindua usatzen da. Aurreko azpisekzioan azaldutako modu berean erabiltzen da; dena den, desberditasun bakarra da artxibo-indikadorea jarri behar dela,

```
fprintf(fid,'testua %-5.2f testua eransgarria', aldagaiaren izena)
```

non `fid` artxiboaren indikadorea den. Azkenik, fitxategian datu guztiak idaztea bukatu denean, artxiboa `fclose(fid)` tekleturik ixten da, eta edukia ikusteko parada dago. Orain, esandako guztia adibide baten bitartez erakutsiko da:

```
>> A=[10 20 30 40];
>> B=A*0.1;
>> f=fopen('irteera.txt','w') % artxiboaren luzapena txt da
f =
```

```
>> fprintf(f,'%i kilo ume % g \n',[A;B])
ans =
    60
>> fclose(f)
ans =
    0
```

Aintzat hartu etsenpluan fitxategiaren luzapena `txt` dela eta `fprintf` aginduak `[A;B]` matrizeko elementuak zutabez zutabe izkiriartzen dituela. Segidan lorturiko *"irteera.txt"* fitxategia editatzen baldin bada, hona emaitza:

```
10 kilo ume 1
20 kilo ume 2
30 kilo ume 3
40 kilo ume 4
```

Azpimarratzekoa da `fclose` agindua exekutatu bitartean sortutako artxiboa editatzen baldin bada edukirik ez dela agertzen.

- Halan eta guztiz ere, alderantzizko prozesua egitea posible da. Alegia, fitxategi batetik datuak irakurtzea; horretarako, `fscanf` komandoa erabil daiteke. Oinarri-oinarrian haren idazkera sinpleena honako hau da:

```
fscanf(fid, formatu-elementua)
```

non bai `fid`, bai `formatu-elementua` lehen deskribatuak diren. Suposatu *"hondar"* izeneko fitxategi batean segidako edukia dagoela, eta datuak bertatik irakurri eta gorde nahi direla:

```
2.23
-1.23
0.12
1.00
```

Esaterako, balio hauek guztiak bektore batean sartzeko zera egin behar da: lehenengo, zabaldu, gero irakurri eta, azkenez, itxi.

```
>> fid=fopen('hondar.txt','r') % 'r' irakurketa-atributua
fid =
    5
>> C=fscanf(fid,'%f') % koma finkoa, lau hamartarrekin
C =
    2.2300
   -1.2300
    0.1200
```

```

1.0000
>> fclose(fid)
ans =
    0

```

Nabarmenezkoa da, batetik, fitxategiko datu kopurua ez dela `fscanf` aginduan zehazten, eta bestetik, `fscanf` aginduaren irteera zutabe-bektore eran lortzen dela. Bada beste planteamendu bat: datuak matrize moduan idatzirik daudela jotzea eta berreskuratu nahi izatea. Orduan, matrizearen dimentsioa ezagutzen baldin bada, `fscanf` aginduaren barruan `size` hautatuta zehaztu daiteke:

```
fscanf(fid, formatu-elementua, size)
```

non `size` matrizearen dimentsio ezaguna den. Kasu baterako *"hondar.txt"* izeneko fitxategiko edukia honako hau baldin bada,

```

2.23    3
-1.23   4
0.12    5
1.00    6

```

balio guztiak irakurri eta aldagai bakar batean gordetzeko, hurrengo pausoak exekutatu orduko, konturatu behar da `fscanf` komandoak lerroak errenkadaz errenkada irakurtzen dituela eta, aldi berean, barrutik balioak zutabeka biltzen dituela:

```

>> fid=fopen('hondar.txt','r')
fid =
    5
>> C=fscanf(fid,'%f %f',[2 4]) % zutabe bi daude
C =
    2.2300   -1.2300    0.1200    1.0000
    3.0000    4.0000    5.0000    6.0000
>> C' % matrize iraulia
ans =
    2.2300    3.0000
   -1.2300    4.0000
    0.1200    5.0000
    1.0000    6.0000
>> fclose(fid)
ans =
    0

```

Halaber, fitxategi batean balio errealak eta testu-karaktereak tartekatzen direnean, beste posibilitate bat da zenbait balio erreal soilik gorde nahi izatea. Kontuan hartu honako egoera hau: *"froga.txt"* artxibotik erradioaren eta perimetroaren balioak aldagai batean bakarrik gorde nahi direla.

```
Erradioa
0.123
Perimetroa
2.101
```

Prozedimendua aurrekoaren antzekoa da, baina, orain, `fscanf`-ren idazkeran, fitxategiko hitzez hitzeko karaktereek agertu behar dute:

```
>> fi=fopen('froga.txt','r') % zabaldu irakurtzeko
fi =
    5
>> D=fscanf(fi,'Erradio \n %f \n Perimetroa \n %f')
D =
    0.1230 % erradioa
    2.1010 % perimetroa
>> fclose(fi)
ans =
    0
```

5.2 Datu-inportazioa eta esportazioa

Eskuarki, Matlab iturri desberdinetako datu esperimentalen analisiak egiteko erabiltzen da. Kanpoko datu horiek prozesatzeko funtsezko modua Matlab-era inportatzea da. Gisa berean, Matlab-en kalkulaturako datuak beste aplikazio batzuetara esporta daitezke. Sekzio honetan bakarrik zenbakizko datuak nola inportatu edo esportatu deskribatuko da. Bestelako datu-tipoetarako, irakurleari Matlab-eko laguntza proposatzen zaio abiapuntu.

Lehenik eta behin, Excel-eko kalkulu-orri batetik, edo orri batera, datuak nola transferitu erakutsiko da. Gaur egun, Excel datuak kudeatzeko oso tresna hedatua da, bereziki maneiatzen dituen datuak beste ohiko aplikazio askorekin bateragarriak direlako. Oro har, Matlab-ek .csv, ASCII eta Lotus formatuetarako datu-transferentzia onartzen du, besteak beste.

- Inportazioari dagokionez, Excel-etik inportazioa gauzatzeko, `xlsread` agindua erabiltzen da. Komando horrek Excel kalkulu-orriko datuak matrize-motako aldagai batera inportatzen ditu. Haren erabilera sinpleena honako hau da:

```
aldagaia=xlsread('fitxategia')
```

Excel-en zenbatgura orri ager daitezkeenez, bi xehetasun ezagutu behar dira:

- Kasu honetan, `fitxategia` Excel artxioboaren izena da, eta, enplegatutako ahalizateko haren kokalekuak laneko direktorioa izan behar du.
- Dena den, jakin beharra dago inportaturako Excel artxioba kalkulatu orri bat baino gehiago edukiz gero, bakarrik lehenengo kalkulatu orriko datuak inportatuko direla.

Prozesua argitzeko, imajinatu segidako "exam1" izeneko Excel fitxategia, A1 : C3 barrutia zenbakiz beterik duena,

```
23 1.2    3
 1  0    10
-1 -3    0.1
```

orain, inportazioa egingo bada, *Aginduetarako Leihotik* zera exekutatu behar da:

```
>> dat=xlsread('exam1')
dat =
 23.0000    1.2000    3.0000
  1.0000         0   10.0000
-1.0000   -3.0000    0.1000
```

Jakin beharra dago Excel fitxategian kalkulu-orri bat baino gehiago izanez gero inportatu nahi den orria `xlsread` aginduaren barruan zehaztu behar dela:

```
aldagaia=xlsread('fitxategia','orriaren izena')
```

Horrezaz gain, badago soilik kalkulu-orriko barruti berezi bat inportatzeko beste aukera bat:

```
aldagaia=xlsread('fitxategia','orriaren izena','barruti zehatza')
```

non barrutia, hala nola, C3 : F5, Excel nomenklaturan idatzita dagoen. Horregatik, jakin behar da C3 : F5 barrutiak 4x3 dimentsioko eskualde bat irudikatzen duela, 3, 4 eta 5 errenkadek eta C, D, E eta F zutabeek mugatzen dutena.

- Matlab-eko datuak Excel orri batera igortzea alderantzizko pausoa da. Prozesua gauzatzeko, hau da, Matlab-etik esportazioa egiteko, `xlswrite` komandoa behar da, eta hau da haren sintaxi laburra:

```
xlswrite('fitxategiaren izena',aldagaia)
```

- Artxiboaren izena Excel-eko fitxategia da, zeinetara datuak esportatu nahi diren.
- Matlab-eko datuak dituen aldagaiaren izena kontuan hartu behar da.
- `xlsread` aginduan aipatu den bezalaxe, orriaren izena eta barruti zehatza parametroak erants daitezke.

Adibidez, A matrizea Excel-eko *exam2* fitxategiko "azterketa" izeneko orri batera esportatzeko, sententzia hauek egin behar dira:

```
>> A=[1:6;2:2:12;-1:-2:-11]
A =
     1     2     3     4     5     6
     2     4     6     8    10    12
    -1    -3    -5    -7    -9   -11
>> xlswrite('exam2',A,'azterketa','C2:H4')
```

Berriro ere, datuen kokapena zehaztu daiteke, betiere Excel nomenklaturaren arabera; hemen, 2., 3. eta 4. errenkadak eta *C*, *D*, *E*, *F*, *G* eta *H* zutabeak seinatzen dira. Jarraian, 5.1 Irudian esportazioaren emaitza agerian azaltzen da:

| | A | B | C | D | E | F | G | H |
|---|---|---|----|----|----|----|----|-----|
| 1 | | | | | | | | |
| 2 | | | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | | | 2 | 4 | 6 | 8 | 10 | 12 |
| 4 | | | -1 | -3 | -5 | -7 | -9 | -11 |
| 5 | | | | | | | | |

5.1 Irudia: Excel-eko kalkulu-orri esportatua.

Azkenez azpimarratu behar da Matlab-en eta Excel-en arteko harremanak estuak direla, eta erakutsitako komando horien gaineko beste aukera batzuk bilatzea irakurleari uzten zaiola, jakin-mina izanez gero, noski.

6. Kapituluia

Bi eta hiru dimentsioko grafikoak

Hasteko, aipatzekoa da Matlab programa grafikoak sortzeko oso trebea dela. Izan ere, era askotariko irudiak erraz eratu daitezke, eta, horrezaz gain, erabiltzaileak era eroso eta sistematikoan alda ditzake grafiko horien propietateak. Areago, Matlab-eko komandoekin grafiko estandarrak, eskala linealak edo logaritmikoak dituztenak, barra-grafikoak, polarak, sare-grafikoak, hiru dimentsioko grafikoak, maila-kurbak, eta abar egin daitezke. Gero, behin sorkuntza sinplea eginik, zenbait parametro modifikatuz, haien itxurak goitik behera personaliza daitezke. Hala, trazuaren aldetik, kolorea, lodiera eta tipoa alda daitezke, besteak beste, edo izenburu bat eman edo testua sartu, edota legenda bat ipini. Ikusten denez, aukera franko daude; haiek guztiak eta propietate batzuk ere deskribatuko dira.

6.1 Bi dimentsioko grafikoak

Noiz edo noiz, datuen azterketa esanguratsua egiteko, bi dimentsioko grafikoak egiteko premia agertzen da; horretarako, plot agindu arruntena dago, Mathematica, Maple edo Octave programetan bezala. Erabilera sinpleena hau da:

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

non x eta y bektoreak diren. Bektoreek edozein izen har dezakete, eta gainera, trazua (x,y) bikoteak biltzen dituzten zuzenkiek osatzen dute. Halaber, plot aginduaren lehenengo argumentuak abzisa-ardatza definitzen du, eta bigarrenak ordenatu-ardatza. Marrazketaren aldetik, ardatz bietan eskala linealak enplegatzen dira, eta irudikatutako eskualdearen mugan balio lehenetsiak aplikatzen dira.

Esan gabe doa, nahi izanez gero, plot komandoaren barruko argumentu batzuk erantsiz, grafikoak personalizatu egin daitezke. Esaterako, trazuaren ezaugarriak aldatu edo grafiko-marka zenbait sartu. Haiek, dudarik ez, plot aginduaren sintaxia zabaltzen dute eta puntuak hobeto adierazten lagundu dezakete, informazio gehiago ateraz:

```
plot(x,y, 'trazu-zehazpenak', 'propietateak', 'balioak')
```

- **Trazu-zehazpenak:** Elementuok aukerakoak dira, eta trazuaren kolorea zein trazu-estiloa edo marka-motak definitzeko enplegatzen dira. Trazu-estiloari dagokionez, haiek ditu zehaztaile nagusiak:

| Trazu-estiloa | Zehaztailea |
|-----------------------|--------------------|
| solidoa (lehenespena) | - |
| etena | -- |
| punteaduna | : |
| marra eta puntukoa | -. |

Kolorearen aldetik hauek dira zehaztatzaile nagusiak:

| Trazu-kolorea | Zehaztailea |
|----------------------|--------------------|
| gorria | r |
| berdea | g |
| urdina | b |
| zian | c |
| magenta | m |
| horia | y |
| beltza | k |
| zuria | w |

Orobat, ondorengo zehaztaileak lagun, grafiko batean grafiko-markak sar daitezke:

| Marka-tipoa | Zehaztatzailea |
|---------------------|-----------------------|
| plus ikurra | + |
| zirkulua | <i>o</i> |
| izartxo | * |
| puntua | . |
| karratua | s |
| diamantea | d |
| bost puntako izarra | p |
| sei puntako izarra | h |

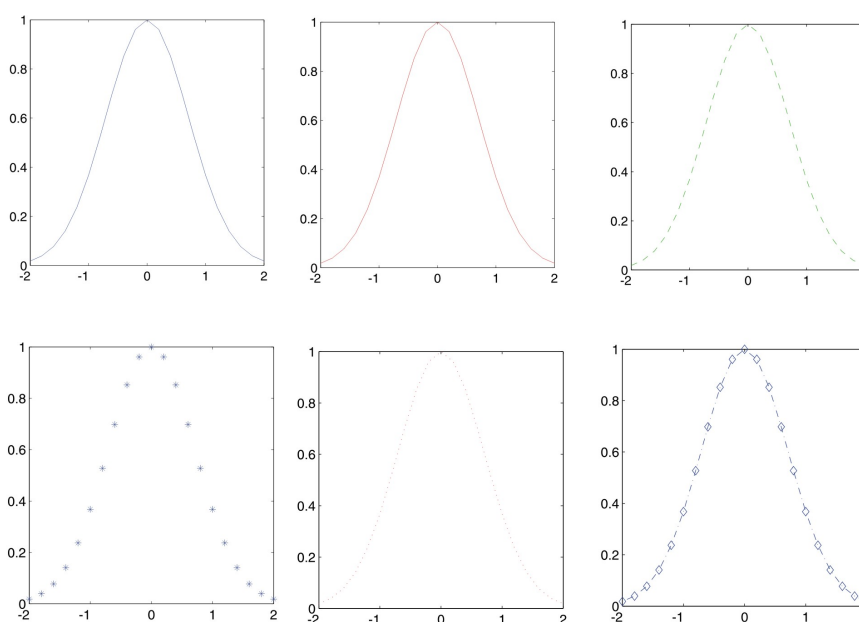
Erreparatu zehaztaileen gaineko oharrei:

- Zehaztaileak plot aginduaren barruan kate gisa sartzen dira.
- Katean zehaztaileak edozein ordenatan idatz daitezke; beraz, ordenarekiko independentzia egon badago.
- Lehen aipatu denez, elementuok aukerakoak dira, hau da, inoiz ez, edo behin, edo birritan ager daitezke.

Esandakoa argitzearren, jarraian etsenplu batzuk zerrendatzen dira, iruzkinak barne:

| | |
|-------------------------------|--|
| <code>plot(x,y)</code> | Trazu solido urdina, eta zehaztaile barik. Kasu sinpleena. |
| <code>plot(x,y,'r')</code> | Trazu jarraitua eta gorria. |
| <code>plot(x,y,'-g')</code> | Trazu etena eta berdea. |
| <code>plot(x,y,'*')</code> | Trazuan soilik izartxo motako zehaztaileak. |
| <code>plot(x,y,'r:')</code> | Trazu punteaduna eta gorria. |
| <code>plot(x,y,'g-.d')</code> | Trazia marra eta puntuduna, berdea eta diamante tipoko zehaztaileak. |

Etsenplu horien exekuzioetan lortzen diren irudikapen grafikoak 6.1 Irudian erakusten dira, hurrenez hurren:



6.1 Irudia: Plot aginduaren aplikazio sinpleak zehaztaileak modifikatuz.

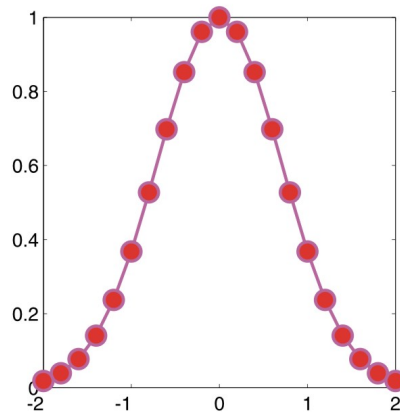
- **Propietateak eta balioak:** Matlab-en badago beste maila bat, marrazkiaren zehaztasun konkretuak atzitzeko, hala nola marken kolorea eta tamaina, trazuaren lodiera eta abar. Haietako batzuk hauek dira,

| | |
|------------------------------|-----------------------|
| <code>color</code> | Lerroaren kolorea. |
| <code>linewidth</code> | Trazuaren lodiera. |
| <code>marker</code> | Marka-tipoa. |
| <code>MarkerEdgeColor</code> | Marken mugen kolorea. |
| <code>MarkerFaceColor</code> | Marken kolorea. |
| <code>MarkerSize</code> | Marken tamaina. |

Hala, aztertu adibide hau, goiko batzuk usatzen dituen:

```
plot(x,y,'-mo','linewidth',2,'markersize',12,'markerfacecolor','r')
```

Grafikoa simplea izan beharrea, aldaketa horiek sartuta kalitatearen aldetik hobea izango da, eta egindako transformazioak 6.2 Irudian islatzen dira:



6.2 Irudia: Grafiko bateko propietateen eta balioen aldaketak.

non trazuaren kolorea magenta den, lodiera 2pt, eta markaren tamaina eta bere kolorea modifikatu egin diren.

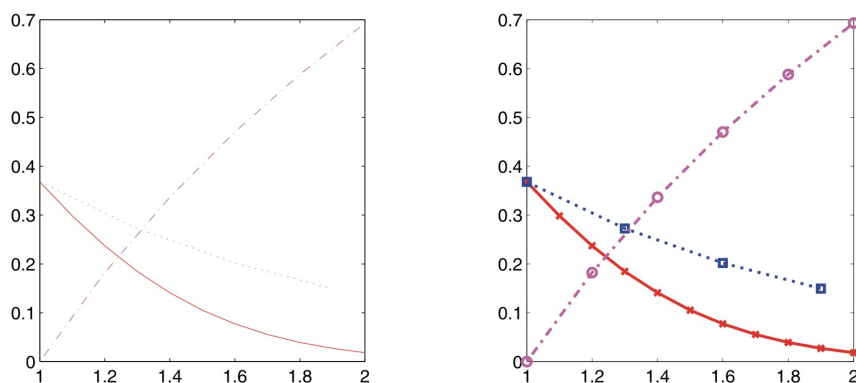
Grafikoen aurkezpenari begira, aukera batzuk baliabide gisa daude:

- Grafiko bat baino gehiago leiho bakar batean irudikatu nahi baldin badira, bitara gauzatu daiteke; batetik, plot aginduan bertan, (x,y) bikote desberdinak espezifikatuz:

```
plot(x1,y1,'prop1',x2,y2,'prop2',x3,y3,'prop3',...)
```

horrela marko grafiko berean y1 vs. x1, y2 vs. x2, etab. irudikatzea lortzen da, 6.3 Irudian azaltzen denez:

```
>> x1=[1:0.1:2];y1=exp(-x1.^2);
>> x2=[1:0.3:2];y2=exp(-x2);
>> x3=[1:0.2:2];y3=log(x3);
>> plot(x1,y1,'r-',x2,y2,'b:',x3,y3,'m-.')
>> plot(x1,y1,'r-x',x2,y2,'b:s',x3,y3,'m-.o','linewidth',2)
```



6.3 Irudia: Zenbait grafiko leiho berean.

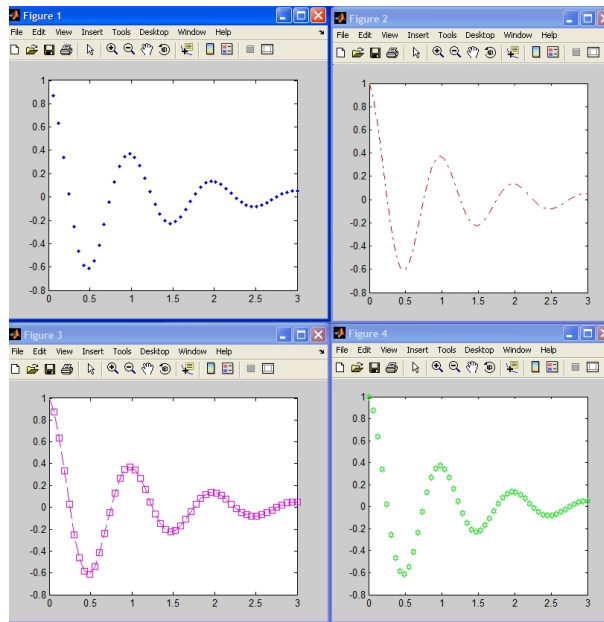
Era honetan plot aginduaren idazkera luzea eta nahasia izan daiteke. Hori dela eta, beste metodo bat aurkezten da, zeinetan hold instrukzioak pantailan grafikoaren gainjartzea aktibatzen duen. Hau da, dagoeneko existitzen diren trazuei beste berri batzuk erants dakizkieke zaharrak ezabatu gabe:

```
>> hold on % gainjartzea aktibatuta
>> plot(x1,y1,'r-x')
>> plot(x2,y2,'b:s')
>> plot(x3,y3,'m-.o')
```

Sententzia horiek exekutatu gero, 6.3 Irudian dagoen eskuineko grafikoa errepikatzen da. Dena dela, kontuan izan agindu honen erabilera askozaz zabalagoa dela, ezen bestelako komandoen bidez eraikitako grafikoak gainjartzea posible baita. Gainjartzea bertan behera izateko, hold off teklatu behar da: eta hala marrazki berriak aurrekoak ezabatuko ditu, eta lehenetsitako erabilerara itzuliko da.

- Berrito irudikapen grafikoez bezainbatean eta batzuk irudikatu beharrez, beste planteamendu bat da grafiko bakoitza bere leiho propioan itxuratzea. Horretarako, figure agindua dago: leiho grafiko bat zabaldu eta zenbaki bat esleitzen zaio. Batzuk zabalik baldin baziren, irekitako leihoetan zehar mugitzeko aukera dago figure erabiliz (6.4 Irudia).

```
>> x=linspace(0,3,50);
>> y=exp(-x).*cos(2*pi*x);
>> figure(1); plot(x,y,'.')
>> figure(2); plot(x,y,'r-.')
>> figure(3); plot(x,y,'sm--')
>> figure(4); plot(x,y,'hg')
```

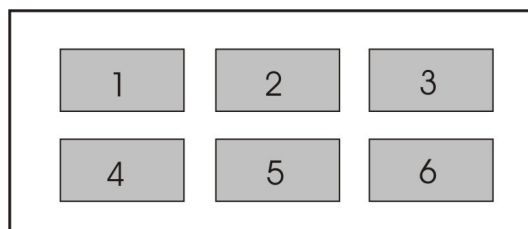


6.4 Irudia: figure aginduaren erabilera.

- Azkenik subplot aginduari esker, leiho nagusi bakar batean askotariko azpileiho grafikoak bistaratu daitezke. Izan ere, komando horren idazkeran berehala matrize-idazkera hautematen da. Berbarako, imajinatu sententzia hau:

```
subplot(231)
```

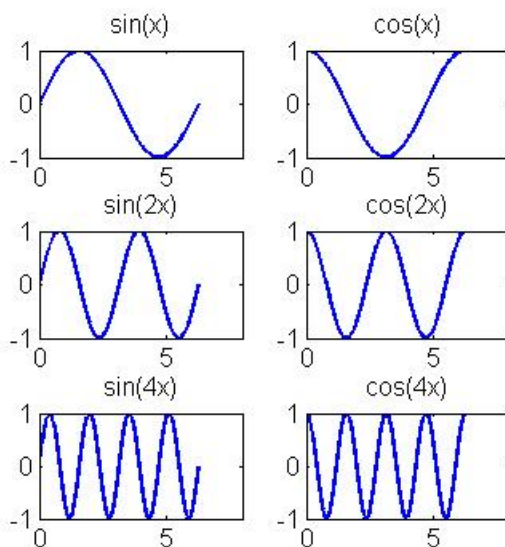
Era honetan "figure" berean sei eskualde definiturik lotzen dira, 2x3, ostean haiei dagozkien irteera grafikoak sartu ahal izateko. Halaber, azken zenbakiari begira, kasu honetan lehenengo zona baino ez da atzitzen. Izan ere, adibide honetan jarraitzen den zenbakikuntza 6.5 Irudian erakusten da, non kasu baterako subplot(235) bosgarren laukia izango den.



6.5 Irudia: Zenbakikuntza-adibide bat subplot aginduarekin.

Segidan, suposatu honako sententzia hauek, 3x2 dimentsioko sarea bat definitzen dutenak,

```
>> clear
>> x=linspace(0,2*pi,200);
>> subplot(321) % lehenengo zona
>> plot(x,sin(x),'linewidth',2);
>> title('sin(x)','fontsize',14)
>> subplot(322)
>> plot(x,cos(x),'linewidth',2);
>> title('cos(x)','fontsize',14)
>> subplot(323)
>> plot(x,sin(2*x),'linewidth',2);
>> title('sin(2x)','fontsize',14)
>> subplot(324)
>> plot(x,cos(2*x),'linewidth',2);
>> title('cos(2x)','fontsize',14)
>> subplot(325)
>> plot(x,sin(4*x),'linewidth',2);
>> title('sin(4x)','fontsize',14)
>> subplot(326) % seigarren zona
>> plot(x,cos(4*x),'linewidth',2);
>> title('cos(4x)','fontsize',14)
```



6.6 Irudia: subplot-en erabilera.

zeinek 6.6 Irudia sortzen duten. Hor, 3×2 dimentsioko grafiko-matrize bat eratzen da, eta subplot aginduarekin posizio bakoitzari dagokion adierazpidea gauzatzen da.

6.1.1 Adierazpen grafikoen formaturako aginduak

Behin irudikapen grafikoa eginda, ohikoa izaten da grafikoaren alderdi batzuk aldatzea edo eranstea, hala nola titulu bat, etiketak, ardatzetan eskala pertsonalizatuak, legendak, sareta, testua, eta abar. Egia da *Grafiko-Leihotik* ere aldaketak egin daitezkeela, baina hemen bereziki komandoak azpimarratuko dira, programa batean sar daitezkeenak eta grafikoa osatzean erabiltzaileari kontrol handiagoa ematen diotenak.

- Uneko leihoan zer edo zer baldin badago, trazuak edo testuak, `clf`-rekin dena ezabatzea lortzen da.
- `xlabel` eta `ylabel`: Ardatzetan etiketak ipintzeko balio dute. Hona haien sintaxia:

```
xlabel('testua'); % abzisa ardatzean
ylabel('testua'); % ordenatu ardatzean
```

- `title`: Goiko partean grafikoari titulu nagusi bat esleitzen zaio. Sintaxia:

```
title('testua')
```

- `text`: Komando honekin grafikoan etiketa bat jar daiteke, eta, gainera, bi aldaera onartzen dira:

```
text(x,y,'testua');
gtext('testua');
```

Lehenengoarekin, `x` eta `y` koordenatuetan aukeratutako testua ipintzen da. Bigarrena erabiliz gero, sagua mugitu eta erabiltzaileak hautatuko lekuan, klik egin ostean, posizio horretan jartzen da testua.

- `legend`: Agindu honen bitartez irudikapen grafikoan legenda bat ipintzen da. Agin-dua egikariturik, legendaren bistaratzean eta dena kolore desberdinez bereizturik, ikusten da bi zati elkarren ondoan daudela: batetik, trazuaren lerro-mota, eta bestetik, etiketa-identifikadorea.

```
legend('1.katea','2.katea',...,posizioa)
```

Grafikoetan kateak etiketak baino ez dira, lerro-tipoaren ondoan agertzen direnak. Orobat, kontuan hartu etiketa horiek definitutako funtzioen ordena bera mantendu behar dutela, korrelazioa logikoa izan dadin. Bestalde, `posizio` aldagaia zenbaki osoa da, eta, horren arabera legendaren kokapena aldatzen da, taula honek esplizitatzen duen bezala:

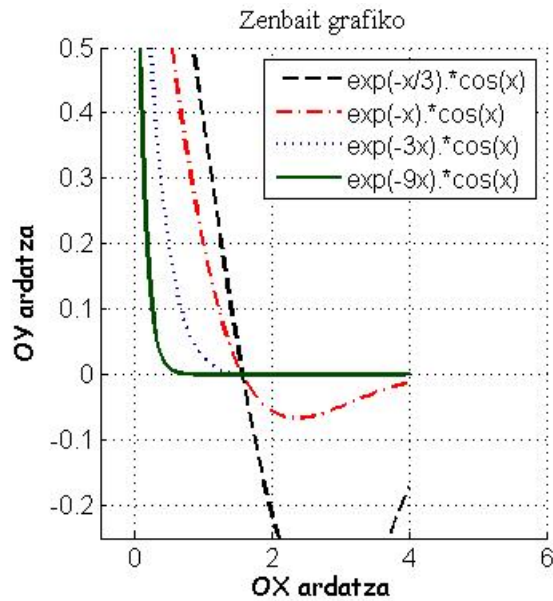
| | |
|-------------------------|---|
| <code>posizio=-1</code> | Legenda eremu grafikotik kanpo eskuinean kokatzen da. |
| <code>posizio=0</code> | Leku eroso automatiko batean eta eremu grafikoaren barruan. |
| <code>posizio=1</code> | Goi-eskuineko izkinan jarri (aukera lehenetsia). |
| <code>posizio=2</code> | Goi-ezkerreko izkinan jarri. |
| <code>posizio=3</code> | Behe-ezkerreko izkinan. |
| <code>posizio=4</code> | Behe-eskuineko izkinan. |

- `grid`: Bi aldaera onartzen ditu: `grid on` erabiliz gero, pantailan sareta bat erakusten da; `grid off` aginduarekin, kontrako prozesua da, eta sareta ezabatzen da.
- `axis`: Komando hau zail samarra izan daiteke. Kasurako, OX-ren eta OY-ren arteko proportzioa kontrolatzen du, edo pantailatik agertzen den grafikoaren zatia, edo erabilitako koordenatuak, besteak beste. Ondoren, xehetasun batzuk baino ez dira emango:

| | |
|--|--|
| <code>axis([xmin xmax ymin ymax])</code> | Ardatzen mugak zehazten dira, minimoak eta maximoak, hain justu. |
| <code>axis auto</code> | Mugen kalkulua automatikoa. |
| <code>axis tight</code> | Mugak datu-eremuaren arabera jartzen dira. |
| <code>axis ij</code> | Koordenatu-jatorria goi-ezkerreko izkian. |
| <code>axis xy</code> | Ardatzak koordenatu cartesiarretan. |
| <code>axis equal</code> | Ardatz bietan eskala bera. |
| <code>axis square</code> | Ardatzen eremua karratu batean. |
| <code>axis off</code> | Etiketak, markak eta ardatzen lerroak kendu. |
| <code>axis on</code> | Etiketak, markak eta ardatzen lerroak ipini. |

Une honetan, ikusitako formaturako aginduak praktikatze aldera, erreparatu adibide honi (emaitza 6.7 Irudian erakusten da):

```
figure(1) % Lehenengo leihoa prest
x=linspace(0,4,90);
f=inline('exp(-n*x).*cos(x)', 'n', 'x') % funtzio bektorizatua
hold on
y=f(1/3,x);
plot(x,y,'k--', 'linewidth', 2)
y=f(1,x);
plot(x,y,'r-.', 'linewidth', 2)
y=f(3,x);
plot(x,y, ':', 'color', [0,0,0.5], 'linewidth', 2)
y=f(9,x);
plot(x,y, '-', 'color', [0,0.3,0], 'linewidth', 2)
grid on
xlim([-0.5,6]),ylim([-0.25,0.5]);
xlabel('OX ardatza', 'fontname', 'Comic Sans Ms', 'fontsize', 12);
ylabel('OY ardatza', 'fontname', 'Comic Sans Ms', 'fontsize', 12);
title('Zenbait grafiko', 'fontname', 'Times new roman', 'fontsize', 16);
legend('exp(-x/3).*cos(x)', 'exp(-x).*cos(x)', 'exp(-3x).*cos(x)',
       'exp(-9x).*cos(x)');
```



6.7 Irudia: Grafikoen formatuari buruzko etsenplua.

Adibide honetan, aztertutako formaturako aginduez gain, eta gorritz seinalaturik, letra-tipoa eta tamaina nola aldatu agertzen da. Era berean, testuaren propietateak alda daitezke, eta bai `text-en`, bai `xlabel-en`, `ylabel-en` zein `title-n` honako identifikadore hauek erabil daitezke:

| | |
|-------------------------|--|
| <code>Fontname</code> | Letra-tipoa zehazteko. |
| <code>FontSize</code> | Letraren tamaina. |
| <code>Color</code> | Testuaren kolorea. |
| <code>LineWidth</code> | Testuaren inguruko kaxa laukizuzeneko ertzaren lodiera. |
| <code>FontWeight</code> | Karaktereen trazua da, eta balioak <code>light</code> (mehea), <code>normal</code> , <code>demi</code> edo <code>bold</code> (lodia) dira. |
| <code>FontAngle</code> | Bere balioak <code>normal</code> , <code>italic</code> edo <code>oblique</code> dira. Parametro honekin letra-tipoaren makurdura kontrolatzen da. |
| <code>Rotate</code> | Testuak izango duen angelua zehazten da. Balio lehenetsia 0 da, idazkera horizontala, alegia. Era berean, 90 baldin bada testua bertikala izango da. |

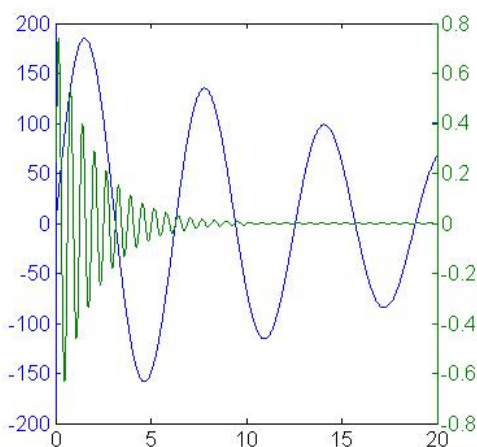
Une honetan aipatu komando bi, interesgarriak direnak oso. Bata lehen aipatutako `clf` da; uneko leihoan dagoen eduki osoa ezabatzeke erabiltzen da. Bigarrena `cla` da, eta uneko *axes* kentzea lortzen da. Kontuan hartu, Matlab-en terminologiaren arabera, leihoko zatian *axes*-ek marrazteke erabiltzen den zona seinalatzen dutela; hori dela eta, agindu horri zailtasuna hortik datorkio, menturaz.

6.1.2 Irteera grafiko motak

Orain arte, plot aginduaren erabilera espezifikatu da, eta propietateak erakutsi dira. Jarraian, bi dimentsioko irudikapen grafikoarekin zerikusia duten zenbait agindu aurkezten dira. Guztien maneiua eta plot-ena oso antzekoak direla berretsiko da, propietateak barne, eta grafiko motak banan-banan adibide batekin batera azalduko dira.

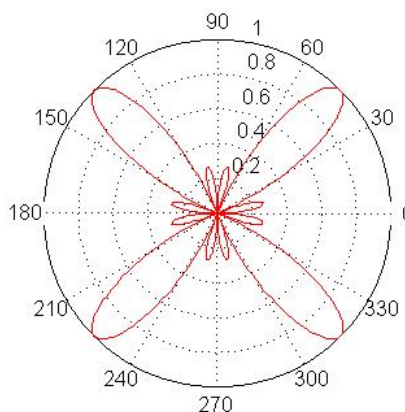
- `plotyy`: Leiho berean grafiko bi, ezkerrean OY ardatz bat eta eskuinean beste OY bat.

```
>> x = 0:0.01:20;
y1 =
200*exp(-0.05*x).*sin(x);
y2 =
0.8*exp(-0.5*x).*sin(10*x);
plotyy(x,y1,x,y2,'plot');
```



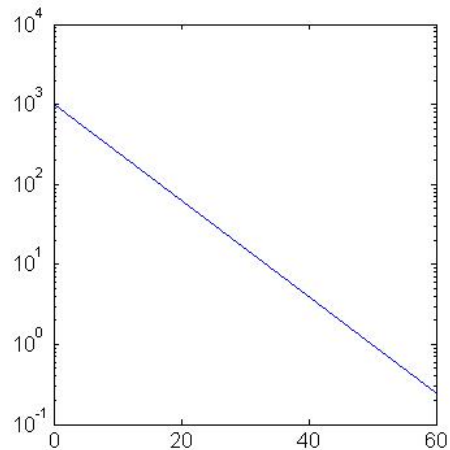
- `polar`: Koordenatu polarretan kurbak marrazteko komandoa.

```
>> t = 0:.01:2*pi;
>>
polar(t,sin(2*t).*cos(4*t)
,'r')
```



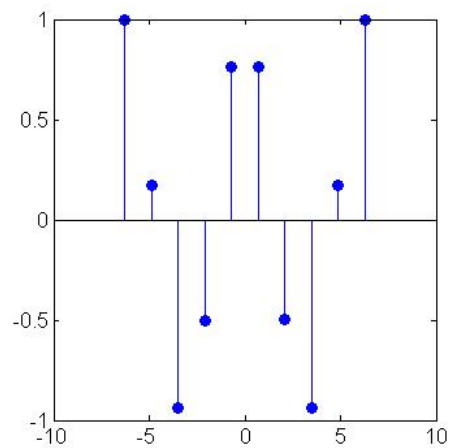
- `semilogx` eta `semilogy`: plot aginduaren berdinak, baina eskala logaritmikoa OX ardatzean eta OY ardatzean erabiltzen da, hurrenez hurren.

```
>>
x=linspace(0.1,60,1000);
>> y=2.^(-0.2*x+10);
>> semilogy(x,y)
```



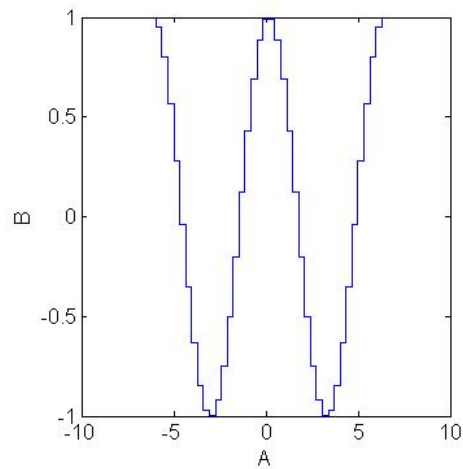
- **loglog**: Ardatz bietan eskala logaritmikoa aplikatzen da, besterik gabe. Berbarako suposatu aurreko adibidea baina loglog aginduarekin.
- **stem**: Puntuak marrazten ditu eta zuzen bertikalen bitartez OX ardatzarekin konektatzen ditu.

```
>> t =
linspace(-2*pi,2*pi,10);
>> h =
stem(t,cos(t),'fill');
```



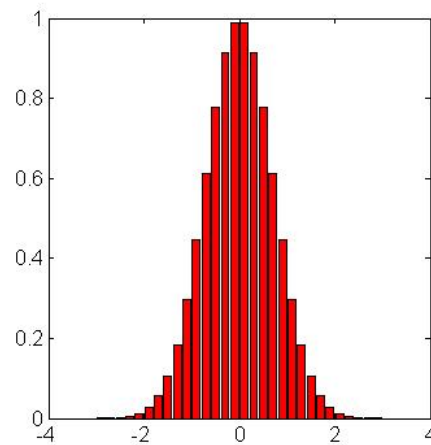
- **stairs**: Eskailera-formako grafikoa sortzen da.

```
>> x =
linspace(-2*pi,2*pi,40);
stairs(x,cos(x))
xlabel('A'),ylabel('B')
```



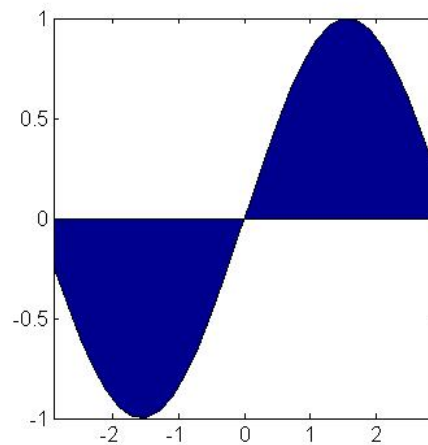
- `bar`, `barh`, `bar3`: Barrazko grafikoak agertzen dira. Biziki egokiak datu estatistikoak irudikatzeko; histograma modukoak agertzen dituzte.

```
>> x = -2.9:0.2:2.9;
bar(x,exp(-x.*x),'r')
```



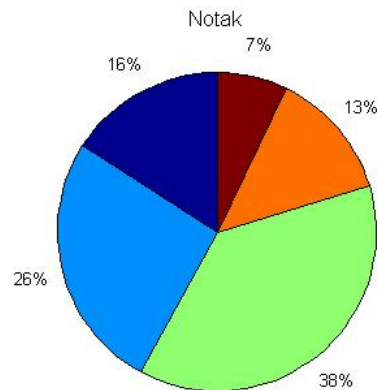
- `area`: Funtzioaren eta OX ardatzaren arteko hutsunea kolore batez betetzen da.

```
>> x = -2.9:0.2:2.9;
area(x,sin(x))
```



- `pie`: Grafiko zirkularrak egiteko, esaterako ekonomia-arloan erabiliak, batik bat.

```
>> grd=[11 18 26 9 5];
>> pie(grd)
>> title('Notak')
```



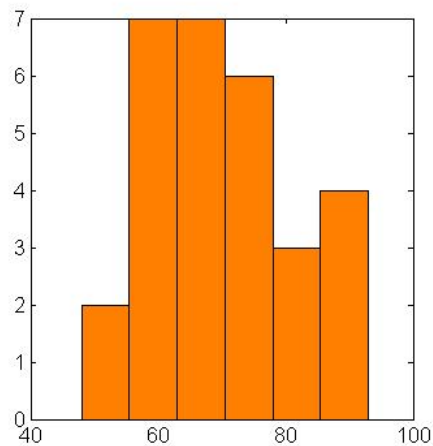
- `hist`: Estatistikan datu-sail baten banaketa erakusteko histiogramak izenekoak erabiltzen dira. Datuen eremua azpitartetan banatzen da, eta histiogramek tarte bakoitzean zenbat puntu dauden erakusten dute. Histiograma barra bertikalez osatutako grafikoa da: barra bakoitzaren zabalera azpitartearen luzera da, eta altuera azpitarteko puntu kopurua da. Histiogramak `hist` aginduari esker sortzen dira, eta sintaxi sinpleena hau da:

`hist(y)`

non `y` bektoreak puntuak datuak jasotzen dituen. Lehenespenez, Matlab-ek datuak luzera bereko hamar azpitartetan banatzen ditu, besterik esplizitatzen ez bada. Nola ez, `x` aukerari esker, tarte kopurua alda daiteke.

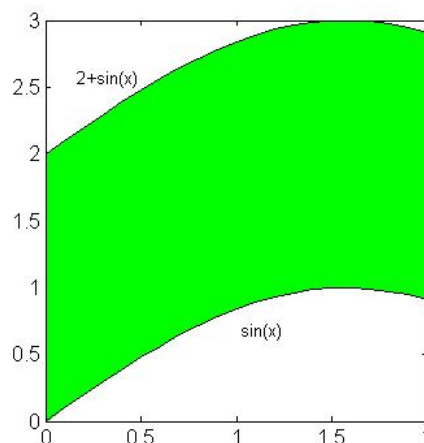
`hist(y,x)`

```
>> y=[58 73 73 50 48 56
73
73 66 69 63 74 82 84
91 93 89 91 80 59 69
56 64 63 66 64 74 63
69];
>> hist(y,6) % 6 tarte
```



- `fill`: Poligono itxiak marraztu eta koloreztatzen ditu. Trebetasuna izanez gero, poligonoaren itxiera osoa lortze aldera, ideia da eskualdearen mugako puntu guztien koordinatuak zehaztea.

```
>> x=[0:0.1:2,2:-0.1:0];
>> y=[sin(x(1:21)),
2+sin(x(22:end))];
>> fill(x,y,'g')
>> gtext('sin(x)')
>> gtext('2+sin(x)')
```



- `line`: (x, y) bikoteak zuzenkien bidez konektatzen dira, eta sintaxiaren aldetik:

`line(x,y,propietateak)`

non `propietateak` trazu-motari edo haren koloreari edo lodierari buruzkoak diren.

Horietaz guztiez aparte, Matlab-en erraz irudika daitezke kalkulu sinbolikoari lotutako funtzioak. Horietarikoak dira `ezplot` eta `ezpolar`. Erabilera erraza da, baina erabiltzaileak ez dauka `plot` aginduarekin adinako kontrola, eta xehetasun hori aintzat hartzekoa da.

6.1.3 `fplot` komandoa

Datuak bistaratzeaz bezainbatean, `fplot` aginduak erabiltzaileak tarte batean zehaztutako $y = f(x)$ funtzioa irudikatzen du. Hau da agindu horren sintaxia:

`fplot('funtzioa',mugak,trazuaren propietateak)`

- `funtzioa`: Aginduaren barruan funtzioa zuzenean karaktere-kate gisa teklea daiteke. Berbarako, funtzioa $f(x) = 8x^2 + 5\cos(x)$ baldin bada, funtzio hau `'8*x^2+5*cos(x)'` sartzen da. Funtzioa Matlab-eko funtzio aurredefinitua izan daiteke edo erabiltzaileak osatutakoak.

```
>> f=inline('x.^2-x+1') % erabiltzaileak definitua
f =
    Inline function:
    f(x) = x.^2-x+1
```

```
>> fplot(f,[-2,2]) % [-2,2] tartea
edo
>> fplot('sqrt',[1,3]) % Matlab-eko sqrt funtzio aurredefinitua
```

Erreparatu funtzioan ezin dela aurrean definituriko aldagai baten baliorik erabili. Hau da, ez da posible aldagai bati balio bat esleitzea eta gero `fplot` aginduan usatzea.

- mugak: hemen x aldagaiaren mugak seinatzeko bi posizioko `[xmin,xmax]` bektorea enplegatzen da, edo lau posiziokoa `[xmin,xmax,ymin,ymax]`, x -ren eta y -ren mugak zehazte aldera.
- Trazuaren propietateak: `plot` aginduan deskribaturikoen berdinak.

6.2 Hiru dimentsioko grafikoak

Datuek bi aldagai baino gehiago dituztenean, hiru dimentsioko grafikoak agertzen dira, *3D* direlakoak hain justu. `Matlab`-ek, irudikapen horiek gauzatzeko, askotariko aukerak eta funtzioak ematen ditu. Halaber, hemen berriro ere, grafikoari espresibitate handiagoa eman dakioke, itxura aldatu eta zenbait efektu hautatu. Bi dimentsioko grafikoak ez bezala, sarri hiru dimentsiokoak ez dira agertzen. Bestalde, aipatu behar da sekzio honetan maila-kurbak dituzten grafikoak aztertzen direla, bi dimentsiokoak izan arren.

6.2.1 Lerro-grafikoak

Hiru dimentsioko espazioan, lerro batek bildutako puntuek *3D* lerro-grafikoa sortzen dute. Horrelakoak egiteko, era sinplea eta oinarrizkoa `plot3`-ren bidezkoa da. Ikusten da haren sintaxia eta lehen ikusitako `plot`-ena oso antzekoak direla:

```
plot3(x,y,z,'trazuaren zehaztaileak','propietateak','balioak')
```

- Lehenengo hiru bektoreek elementu-kopuru bera izan behar dute.
- Aipatu trazuaren zehaztaileak eta bi dimentsioko grafikoetarako 78. orrialdekoak berdinak direla.

Adibidez, suposatu honako kurba parametrizatu hau dagoela, eta hiru dimentsiotan irudikatze guraria dagoela:

$$x = \sqrt{t} \sin(2t)$$

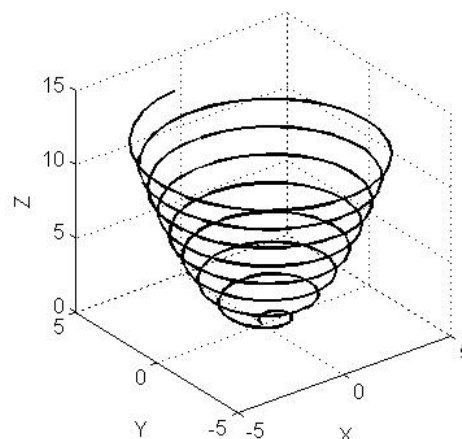
$$y = \sqrt{t} \sin(2t)$$

$$z = 0.5t$$

```

>> t=0:0.1:8*pi;
>> x=sqrt(t).*sin(2*t);
>> y=sqrt(t).*cos(2*t);
>> z=0.5*t;
>> plot3(x,y,z,'k','linewidth',2)
>> grid on
>> xlabel('X');ylabel('Y');
zlabel('Z');

```



6.2.2 Sareta- eta gainazal-grafikoak

Bai sareta-grafikoak, bai gainazal-grafikoak hiru dimentsioko grafikoak dira, eta $z = f(x, y)$ motako funtzioak irudikatzeko erabiltzen dira. Hemen z mendeko aldagaia da, eta x eta y aldagai askeak dira. Horrelako grafikoak sortzeko, hiru pauso nagusi eman behar dira. Aurrena, OXY planoko sareta sortu behar da, z funtzioaren definizio-eremua estaltzen duena. Hurrena, saretako puntu guztietan funtzioaren balioa kalkulatu behar da, eta, azkenik, agindu egokiren baten bitartez adierazpide grafikoa etortzen da.

Sareari dagokionez, imajinatu $z = f(x, y)$ funtzioa eremu laukizuzen batean irudikatu nahi dela, $x = (x_1, x_2, \dots, x_n)$ eta $y = (y_1, y_2, \dots, y_m)$ puntuak erabiliz. Sareta (x_i, y_j) puntuek eratzen dute eta sarea sortzeko Matlab-en meshgrid agindua existitzen da:

$$[X, Y] = \text{meshgrid}(x, y)$$

non X eta Y honako $m \times n$ ordenako matrize bi diren:

$$X = \begin{bmatrix} x_1 & \cdots & x_n \\ \vdots & & \vdots \\ x_1 & \cdots & x_n \end{bmatrix} \quad Y = \begin{bmatrix} y_1 & \cdots & y_m \\ \vdots & & \vdots \\ y_m & \cdots & y_m \end{bmatrix}$$

Jakinaren gainean egon, X -eko m errenkadak x bektorearen kopiak direla, eta Y -ko n zutabeak y bektorearen kopiak direla. Horrela, sareta $(X(i, j), Y(i, j))$ puntuez osaturik dago. Adibidez,

```

>> x=[0.1 0.2 0.3 0.4];y=[-2 0 2];
>> [X,Y]=meshgrid(x,y)
X =

```

```

    0.1000    0.2000    0.3000    0.4000
    0.1000    0.2000    0.3000    0.4000
    0.1000    0.2000    0.3000    0.4000

```

```

Y =

```

$$\begin{array}{cccc} -2 & -2 & -2 & -2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 2 & 2 & 2 & 2 \end{array}$$

Hurrengo urratsa puntu horietan $z = f(x, y)$ funtzioa balioztatzea da, hots:

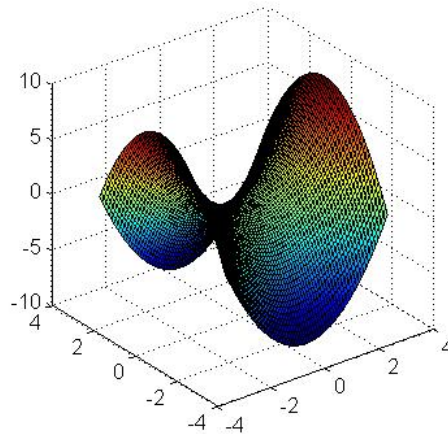
$$Z = f(\mathbf{X}, \mathbf{Y}) = \begin{bmatrix} f(x_1, y_1) & \dots & f(x_n, y_1) \\ \vdots & & \vdots \\ f(x_1, y_m) & \dots & f(x_n, y_m) \end{bmatrix}$$

Adibidez, imajinatu definituriko saretan $z = x^2 - y^2$ funtzioa balioetsi nahi dela. Horretarako, eta \mathbf{X} eta \mathbf{Y} matrizeak aprobetxatuz, honako definizio hau erabiltzen da:

$$Z = \mathbf{X}.^2 - \mathbf{Y}.^2;$$

Azkenekoz, sare-grafikoak eta gainazal-grafikoak `mesh(X,Y,Z)` edo `surf(X,Y,Z)` komandoen bitartez irudikatzen dira, hurrenez hurren.

```
>> x=linspace(-3,3,70);
>> y=linspace(-3,3,70);
>> [X,Y]=meshgrid(x,y);
>> Z=X.^2-Y.^2;
>> surf(X,Y,Z)
```



OHARRA: Adierazpideetan, kolorea Z matrizearen magnitudearen arabera da. Edozein aldaketa sartzeko, `colormap(C)` agindua erabil daiteke, C m errenkadako eta 3 zutabeko matrizea delarik. Errenkada bakoitzak kolore gorriaren, berdearen eta urdinaren intentsitateak adierazten ditu, hurrenez hurren; alabaina, RGB eskala usatzen da. Balio horiek $[0, 1]$ tartean daude, 0 intentsitate minimoa eta 1 intentsitate maximoa izanik. Hauek dira tonu ohikoenak:

```
C=[0 0 0] beltza   C=[1 0 0] gorria   C=[0 1 0] berdea
C=[0 0 1] urdina  C=[1 1 0] horia    C=[1 0 1] magenta
C=[0.5 0.5 0.5] grisa
```

Dena den, burua nekatu barik, aurrez definituriko formatu batzuk existitzen dira,

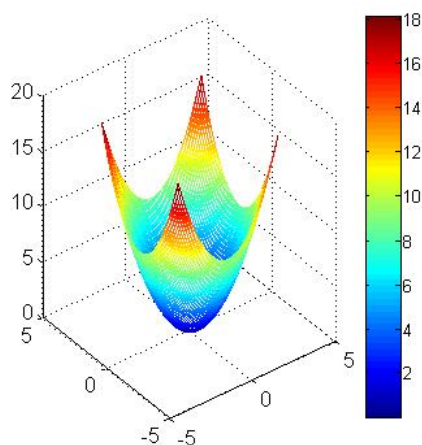
```
autumn  bone    colorcube  cool    cooper
flag    gray    hot        hsv     jet
lines   pink    prism     spring  summer
white   winter  default
```

eta, formatua kargatzeko, aski da agindu hau exekutatzea:

```
colormap('pink')
```

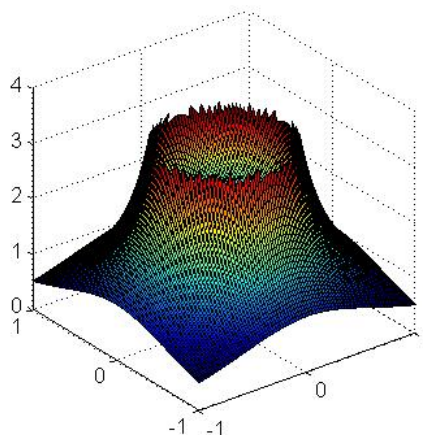
Izan ere, askotan hiru dimentsioko grafikoaren ondoan kolore-barra bat ipintzea komeni da, bereziki barra horretan koten balioak begi-bistan izateko. Horretarako, `colorbar` erabiltzen da. Liburuan agertu ez arren, komando horrek zenbait konfigurazio onartzen ditu.

```
>> x=linspace(-3,3,70);
>> y=linspace(-3,3,70);
>> [X,Y]=meshgrid(x,y);
>> Z=X.^2+Y.^2;
>> mesh(X,Y,Z)
>> colorbar
```



Grafikoen gaiarekin jarraituz, hurrengo adibidean NaN *Not a Number*-en aplikazio bitxi bat erakutsiko da. Trikimailu hori eragiketa ezinezkoak edo infinituak ekiditeko aplikatu daitezke, hala nola $1/0$ eragiketa egin behar denean, errore-mezua ez agertzeko eta irudikatu ahal izateko, betiere singularitasunak saihestuz.

```
>> x=linspace(-1,1,101);
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=1./(X.^2+Y.^2);
>> id=X.^2+Y.^2<0.3;
>> Z(id)=NaN;
>> surf(X,Y,Z)
```

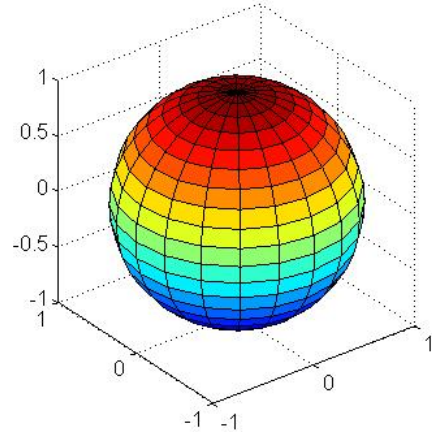
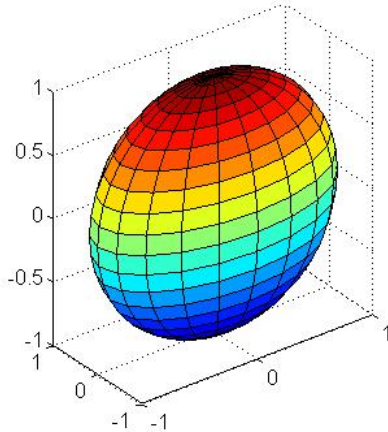


Alegia, etsenplu honetan `Z` funtzioaren puntu singularra $(0,0)$ da eta bosgarren sententzian eta erlazio-eragile bati esker, haren inguruko puntu batzuk bilatzen dira eta seigarren instrukzioan NaN balioa esleitzen zaie. Irudikapen grafikoan ikusten denez, $(0,0)$ puntuaren inguruan ez da ezer azaltzen, zulo bat baizik; indeterminazioa ekidin egin da, hain zuzen.

Bestalde 3D inguruneko beste agindu bat daspect da. Hari esker, marrazkian dauden ardatzen proportzionaltasuna kontrolatzen da. Berbarako nahikoa da,

```
daspect([1 1 1])
```

teklatzea OX, OY eta OZ ardatzen proportzioak berdinak izan daitezzen.



Hemen, ezkerreko grafikoan, `daspect([0.5 1 0.5])` aplikatu da, eta eskuinekoan, berriz, `daspect([1 1 1])`; beraz, kontuak atera agindu horrek zer eragin duen.

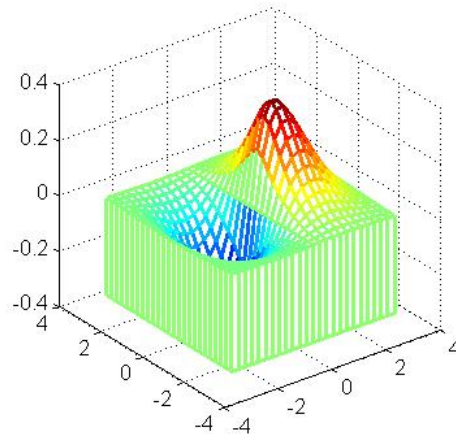
Era berean, ardatzen limiteak zehaztea interesgarria izan daiteke; horretarako, berriro ere, `axis` agindua laguntzaile dago. Sintaxiari dagokionez, bi dimentsioko grafikoena bezalako da, baina z hirugarren osagaia kontuan harturik:

```
axis([xmin xmax ymin ymax zmin zmax])
```

Orain arte Matlab-en ikusitakoen antzeko beste hiru dimentsioko grafiko batzuk gauzatu daitezke. Segidan komando nagusiak, etsenpluak barne, deskribatzen dira:

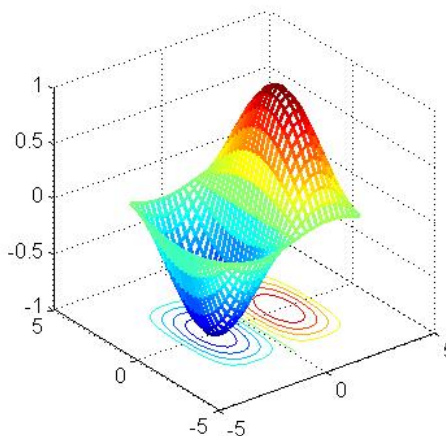
- `meshz`: Saretaren inguruan kortina bat marrazten da, gainazaletik OXY planora doana.

```
>> x=-3:0.25:3;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=1.8.^
(-1.5*sqrt(X.^2+Y.^2))
.*cos(0.5*Y).*sin(X);
>> meshz(X,Y,Z)
>>
xlabel('x');ylabel('y');
zlabel('z');
```



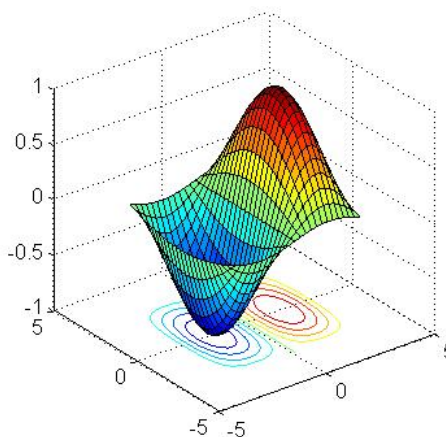
- `meshc`: Saretaren azpian gainazalarekin batera bere maila-kurbak agertzea lortzen da.

```
>> x=-3:0.25:3;  
>> y=x;  
>> [X,Y]=meshgrid(x,y);  
>> Z=cos(0.5*Y).*sin(X);  
>> meshc(X,Y,Z)
```



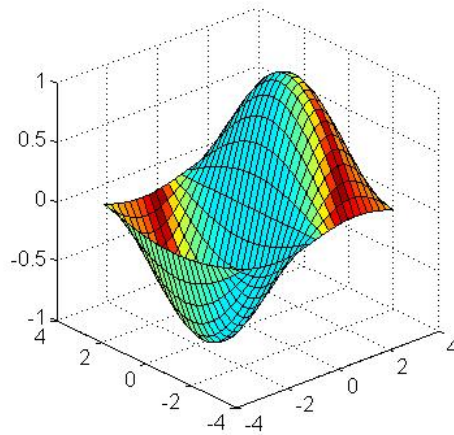
- `surf`: Gainazal-grafiko bat lortzen da, eta, horrekin batera, OXY planoan maila-kurbak marrazten dira. Irteera horren aitzinean, hori erkatu mesh aginduari esker exekutatzen den emaitzarekin.

```
>> x=-3:0.25:3;  
>> y=x;  
>> [X,Y]=meshgrid(x,y);  
>> Z=cos(0.5*Y).*sin(X);  
>> surf(X,Y,Z)
```



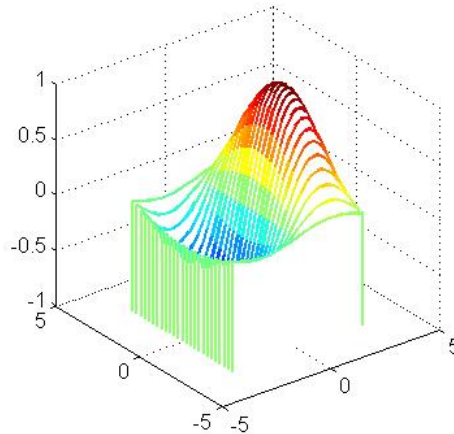
- `surf1`: Gainazal-grafiko argitua irudikatzen da. Argitasuna kanpoko argi-puntu baten menpe dago eta konfiguratu egin daiteke. Irakurleari ikerketa bat egitea proposatzen zaio.

```
>> x=-3:0.25:3;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=cos(0.5*Y).*sin(X);
>> surf1(X,Y,Z)
```



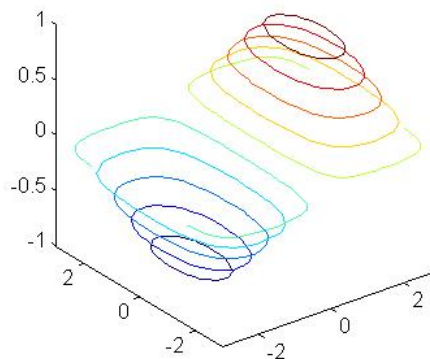
- waterfall: Ur-jauzien moduko grafikoa da, eta sareta norabide bakarrekoa da.

```
>> x=-3:0.25:3;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=cos(0.5*Y).*sin(X);
>> waterfall(X,Y,Z)
```



- contour3: Gainazalaren maila-kurbak hiru dimentsiotan . Maila-kurben kopurua nahierara alda daiteke. Adibide honetan kopurua 10 da.

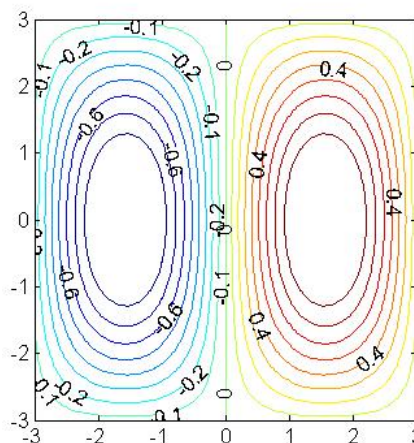
```
>> x=-3:0.25:3;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=cos(0.5*Y).*sin(X);
>> contour3(X,Y,Z,10)
>> grid off
```



- contour: Aurrekoaren antzekoa baina gainazalaren maila-kurbak planoan proiektatu eta irudikatzen dira. Maila-kurben kopurua finkatzeaz aparte, erabiltzaileak aukera du

marrastu nahi dituen kotak seinatzeko, baita etiketatzeko ere. Erreparatu adibide honi:

```
>> x=linspace(-3,3,100);
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=cos(0.5*Y).*sin(X);
>> cvals=[-0.8:0.1:0.8];
>>
[C,h]=contour(X,Y,Z,cvals);
>> clabel(C,h,cvals([3 7
8 9
13]))
```



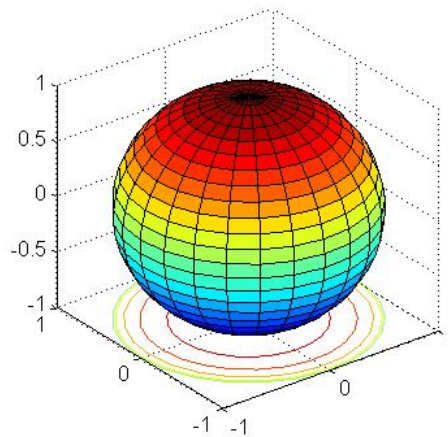
Aurreko grafikoa sortzeko, bosgarren sententzian `cvals` agindua erabili da. Hari esker zehaztu dira agertu behar duten maila-kurben kotak. Are gehiago, zazpigarren instrukzioari esker, `clabel` lagun kota horietatik zeintzuk etiketatu nahi diren esplizitatzen da; kasu honetan, OXY planoan $-0,6$, $-0,2$, $-0,1$, 0 eta $0,4$ kotak dituzten puntuen proiektzioak. Halaber, `clabel(C,h,'manual')` idatziz gero, erabiltzaileak, saguarekin klik eginez, nahi beste etiketa ipin ditzake. Saiatu, interesgarria da eta.

6.2.3 Grafiko bereziak

Oraindaino 3D irudikapen grafikoak eratzeko deskribatutako aginduez aparte, Matlabek hiru dimentsioko grafiko bereziak sortzeko funtzio batzuk ditu eskura. Aipatzekoa da hurrengo adibideetan komando horiek erabil ditzaketan aukerak ez direla deskribatuko. Xehez, informazio gehiagoren premian izanez gero, *Aginduetarako Leihotik help* aginduaren izena teklatu edo doc agindua besterik ez da egin behar, esplizitatutako komandoaren gaineko informazio sakonagoa izateko.

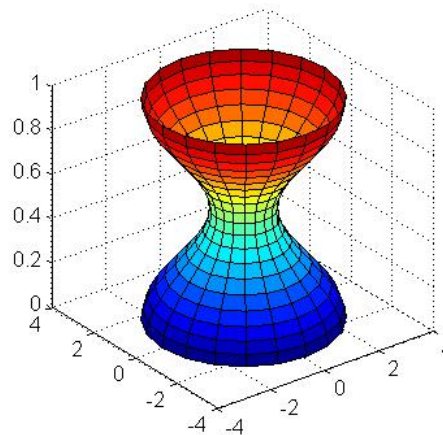
- **sphere**: Bat erradioko esfera baten koordenatuak lortzen dira, eta argumentu gisa komandoaren barnean zenbat xerra agertuko diren espezifika daiteke. Gero, informazioa bildu ostean, `mesh` edo `surf` aginduekin irudikapena gauzatzen da.

```
>> [X,Y,Z]=sphere(25);
>> surfc(X,Y,Z)
```



- cylinder: Bat altuerako biraketa-gainazal baten koordenatuak itzultzen ditu, eta, haren soslaia zehazteko bektore bat erabiltzen da. Gero, irudikapen grafikoak lortzeko, lehen ikusitako aginduren bat usatu besterik ez da behar.

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] =
cylinder(2+cos(t));
>> surf(X,Y,Z)
```



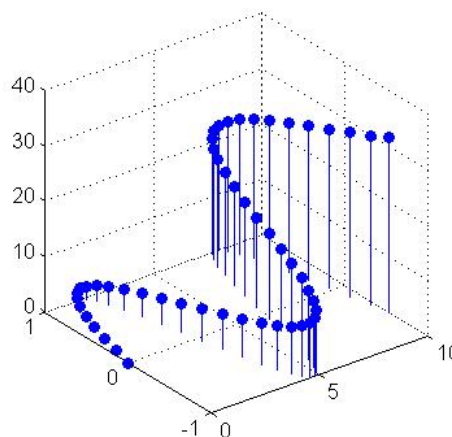
Paradoxikoa iruditu arren, `cylinder(2:0.1:10,20)` sententziak enbor-kono bat sortzen du. Zergatik? Ahalegindu eta arrazoiak eman.

- stem3: Hiru dimentsiotan zurtoin-grafikoak egiteko erabilia. Lehen stem agindua-ekin gertatzen zen legez espazioko kurbako puntuak OXY planoarekin konektatzen dira.

```

>> t=[0:0.2:10];
>> x=t;
>> y=sin(t);
>> z=t.^1.5;
>> stem3(x,y,z,'fill')
>> grid on
>> box off

```

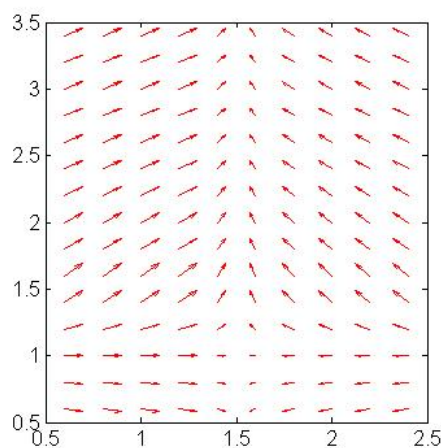


- `quiver`: Planoko puntuei bektoreak egokitzen zaizkie; hala, `quiver(X,Y,U,V)` aginduan (x_i, y_i) puntuari (u_i, v_i) bektorea esleitzen zaio, eta matematikoki bektore-eremu bat eratzen da.

```

>> x=0:0.2:4;y=x;
>> xp=x.*(3-y-x);
>> yp=y.*(x-1);
>>
up=xp./(sqrt(xp.^2+yp.^2)+eps)
>>
vp=yp./(sqrt(xp.^2+yp.^2)+eps)
>> [X,Y]=meshgrid(x,y);
>> [U,V]=meshgrid(up,vp);
>> quiver(X,Y,U,V,0.5)
>> axis([0.5 2.5 0.5
3.5])

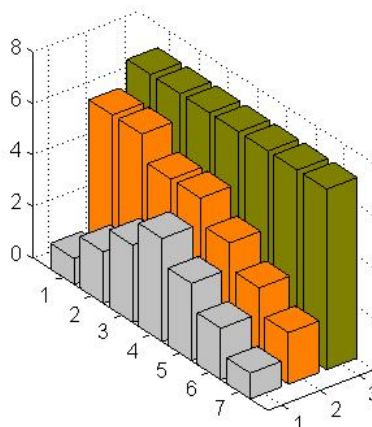
```



Nabarmendu beharra dago, kasu honetan eremu bektorialaren adierazpenean, $1/0$ indeterminazioa ekiditzearren, izendatzaileei "eps" kantitatea gehitzen zaiela. Alta, eps magnitudea 1-en eta hurrengo koma mugikorreko zenbakiaren arteko distantzia da, $2.2e - 016$ gutxi-asko.

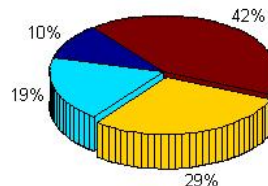
- `bar3`: Agindu honen argumentua matrize bat da, non elementu bakoitza barra bat den eta datuen irakurketa zutabez zutabe egiten den.

```
>> Y=[1 6 7;2 6 7;
3 5 7; 4 5 7;
3 4 7; 2 3 7;
1 2 7];
>> bar3(Y)
```



- **pie**: Komando honen idazkera `pie3(X,e)` da, eta, e eta X bektoreek luzera bera dute. Hala, e bektorea bat eta zero zenbakiez beterikoa da. Zenbakia bat baldin bada, tartaren grafikoan horri dagokion zatia zer edo zer bananduko da, eta zero kasuan ez da ezertarko bananduko. Etsenpluan hirugarren zatia zerbait urruntzen da.

```
>> X=[5 9 14 20];
>> e=[0 0 1 0];
>> pie3(X,e)
```



Erabiltzaleari grafikoak osatzea errazteko asmotan, Matlab-ek, agindu erabilerrazak ditu, normalean *easy use commands* izenekoak, hain justu.

```
ezmesh    ezmeshc    ezsurf    ezsurf
ezcontour ezcontourf ezplot3
```

Kasu honetan, kontrola da arazo nagusia: erabiltzaileak ez du aukera handirik grafikoaren itxura kontrolatzeko.

6.3 Objektu grafikoak, identifikatzaileak eta propietateak

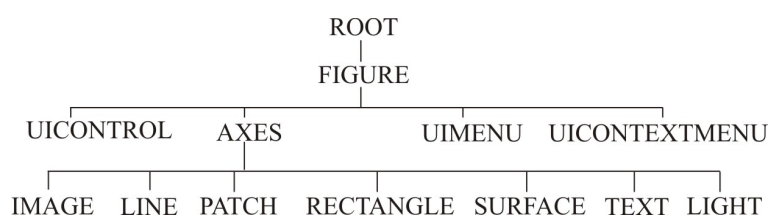
Matlab-eko sistema grafikoaren oinarrian objektu grafikoak daude. Objektu horiek, 6.8 Irudian jasotzen denez, arbola-tipoko hierarkia-egitura batean antolatuta daude.

AXES objektuek irudi grafikoaren barruan eremu bat definitzen dute, zeinetan irudiak (IMAGE), bi eta hiru dimentsioko marrazkiak (LINE,PATCH,RECTANGLE,SURFACE), testuak (TEXT) eta abar agertuko diren.

UNICONTROL objektuak testu-koadroak eta botoiak dira; haiei esker, ekintza zehatzak egin daitezke. Normalean, erabiltzaileak programatuak izaten dira eta saguarekin aktibatzen dira.

UIMENU objektuak erabiltzaileak sortutako menuak dira, eta, nagusiki, irudiaren goiko partean dagoen menura erantzen dira.

UICONTEXTMENU objektuak saguaren eskuineko botoiarekin aktibatzen diren menuak dira. Gainera, elementuok aurreko objektuekin elkartu daitezke.



6.8 Irudia: Objektu grafikoaren egitura hierarkikoa.

Objektu grafiko hauek sortzen direnean, Matlab-ek identifikadore gisa zenbaki bat, *handle* deritzona, esleitzen die. Berbarako, ROOT objektuaren identifikatzailea zero zenbakia da beti. Edo FIGURE objektuaren identifikadorea `figure(n)` komandoaren bidez esleituriko zenbakia da. Gainontzeko identifikatzaileak zenbaki errealak dira, eta Matlab-ek sortutako informazioa gordetzen dute. Balio horiek aldagaietan sar daitezke geroxeago erabiltzeko. Bestalde, Matlab-en badaude hiru komando erraz identifikadore batzuk atzitzeko:

- **gcf:** Uneko adierazpide grafikoaren FIGURE-ren identifikatzailea itzultzen du.
- **gca:** Uneko AXES-en identifikadorea ematen du.
- **gco:** Saguarekin seinalaturiko objektuaren identifikadorea lortzen da. Hori interesgarria izan daiteke, adibidez, 2D trazu bateko puntuen (x, y) koordenatuak erazteko.

Funtzio horiek beste funtzio batzuen sarrerako argumentuak izan daitezke, batik bat, objektuen propietateak aldatze aldera.

OBJEKTU GRAFIKOEN PROPIETATEAK: Objektu grafiko guztiak propietatez beterik daude, eta propietateok objektuak pantailan agertzeko era kontrolatzen dute. Izan ere, berez eta besterik esan ezean, propietate horiek balio lehenetsiak hartzen dituzte. Zernahi gisaz, eta betiko legez, Matlab-en beste modu batean konfigura daitezke. Horretarako, bereziki, `set`, `get` eta `findobj` aginduak oso lagungarriak dira.

Laburbilduz, *Aginduetarako Leihotik* edo funtzio edo *script* batetik, esaterako, era orokorrean bada ere, propietateak `get`-en bitartez bistaratu daitezke, eta `set`-en bidez edita

daitezke. Adibidez, suposatu h aldagaian objektu baten identifikadorea gordeta dagoela, orduan

```
>> get(h)
```

sententziarekin objektu horren propietateak uneko balioekin batera erakusten dira pantailan:

```
>> x=[0:1:10];y=x;
>> h=plot(x,y) % handle-a
h =
    159.0023
>> get(h)
        Color: [0 0 1]
        EraseMode: 'normal'
        LineStyle: '-'
        LineWidth: 0.5000
        Marker: 'none'
        MarkerSize: 6
        MarkerEdgeColor: 'auto'
        MarkerFaceColor: 'none'
        XData: [0 1 2 3 4 5 6 7 8 9 10]
        YData: [0 1 2 3 4 5 6 7 8 9 10]
        ZData: [1x0 double]
        BeingDeleted: 'off'
        ButtonDownFcn: []
        Children: [0x1 double]
        Clipping: 'on'
        CreateFcn: []
        DeleteFcn: []
        BusyAction: 'queue'
        HandleVisibility: 'on'
        HitTest: 'on'
        Interruptible: 'on'
        Selected: 'off'
        SelectionHighlight: 'on'
        Tag: ''
        Type: 'line'
        UIContextMenu: []
        UserData: []
        Visible: 'on'
        Parent: 158.0018
        DisplayName: ''
        XDataMode: 'manual'
        XDataSource: ''
        YDataSource: ''
        ZDataSource: ''
```

Aurreko etsenpluan `h`-ren izaera ez da ezagutu. Horregatik, eta informazio hori azalertzeko, `findobj` teklatzen da grafikoko objektuen *handle*-zerrenda zein den jakiteko,

```
>> findobj
ans =
     0
  1.0000
 158.0018
 159.0023
```

eta objektu grafikoei dagozkien *handle*-ak erakusten dira. Dena dela, aurreko kasuan `plot`-en irteera zer objektu grafiko den zehazteko, aski da haue teklatzea:

```
>> get(h, 'type')
ans =
line
```

Kasu honetan, `h=159.0023` identifikatzailea LINE tipokoa da, eta propietateak eta uneko balioak aurreko zerrendan agertzen dira.

Beste ikuspuntu batetik, grafiko baten objektuak eta objektu grafikoaren propietateak nola lortu deskribatzen da.

```
1 >> x=[0:10];y=x;
2 >> plot(x,y)
3 >> h=findobj
4 h =
5     0
6   1.0000
7  158.0018
8  159.0029
9 >> get(h, 'type')
10 ans =
11   'root'
12   'figure'
13   'axes'
14   'line'
```

Behin irudikapen grafikoa eginda, hirugarren lerroan adierazpide grafikoaren objektu grafikoaren identifikadore guztiak lortzen dira. Aldi berean, bederatzigarren sententzia exekutatu gero, identifikatzaileen izenak agertzen dira. Aurreko `h` *handle*-a LINE tipokoa zen, baina orain bektore bat da, eta, esaterako, `h(3)` AXES-en identifikadorea da. Beharbada era hori bestea baino askoz erosoagoa eta osoagoa da.

Hurrengo urratsa objektu grafiko baten propietate bat aldatzea izan daiteke. Horretarako, lehendabizi propietatearen uneko balioa ezagutu behar da. Suposatu `h` identifikatzailea dela, orduan zera teklatu behar da:

```
get(h, 'propietatearen izena')
```

Adibidez, aurreko adibidean `h(3)` zenbakiak AXES objektua deskribatzen zuen. Aparte, lehen erakutsi denez, `get(h(3))` exekutatzaren bada, propietate guztiak, uneko balioak barne, zerredatzen dira. Imajinatu definitu nahi dena XLim propietatea dela. Lehenik, pantailan uneko balioa erakusteko,

```
>> get(h(3), 'XLim')
ans =
     0     10
```

tekleatzen da, eta balioa birdefinitzeko `set` komandoa ondoko eran erabili behar da:

```
>> set(h(3), 'XLim', [1 3])
```

non ikus daitekeen ostera `get(h(3), 'XLim')` egikaritzuz gero OX ardatzaren mugak [1, 3] direla.

Prozedimendu hori hainbat eta hainbat propietaterekin errepika daiteke. Irakurleari saiakerak egitea proposatzen diogu. Bestalde, objektuaren uneko propietateak eta balioak beste modu batean lortu eta erakuts daitezke, eta uneko balioak eta zenbait posibilitate agertzen dira. Aurreko adibidearekin jarraikiz, `h(4)` *handle*-a LINE objektua da, eta honako hau exekutaturik:

```
>> set(h(4))
Color:
    EraseMode: 4x1 cell
    LineStyle: 5x1 cell
    LineWidth:
    Marker: 14x1 cell
    MarkerSize:
    MarkerEdgeColor: 2x1 cell
    MarkerFaceColor: 2x1 cell
    XData:
    YData:
    ZData:
    ButtonDownFcn:
    Children:
    Clipping: 2x1 cell
    CreateFcn:
    DeleteFcn:
    BusyAction: 2x1 cell
    HandleVisibility: 3x1 cell
    HitTest: 2x1 cell
    Interruptible: 2x1 cell
    Selected: 2x1 cell
    SelectionHighlight: 2x1 cell
    Tag:
    UIContextMenu:
    UserData:
```

```

    Visible: 2x1 cell
    Parent:
  DisplayName:
    XDataMode: 2x1 cell
  XDataSource:
  YDataSource:
  ZDataSource:

```

propietateak ere zerrendatzen dira, eta hau tekleturik,

```

>> get(h(4), 'XData')
ans =
    0     1     2     3     4     5     6     7     8     9    10

```

haren uneko balioak -kasu honetan, x aldagaiaren balioak- berreskura daitezke.

Sekzioaren lehenengo zatia bukatu orduko esandakoaren gaineko zenbait etsenplu irudikatu dira 6.9 Irudian. Jakin-mina asebetetzeko, propietateei eta posibilitate zabalei buruzko ikerketa sakonagoa egitea proposatzen diogu irakurleari. Kontuan izan behar da adierazpide honetan sortutako lehenengo irudia (1, 1) posizioan kokatuta dagoela eta errenkadaz errendaka aldaketak eransten zaizkiola.

6.3.1 Grafikoak inprimatzea

Grafikoak sortzeko erak eta hainbat aukera deskribatuta, emaitzak gordetzea interesgarria izan daiteke. Aurrena, jakin behar da Matlab-eko irudien luzapen lehenetsia `fig` luzapena dela. Gordetzeko, *Irudietarako Leihotik* goitibeherako *File* menua zabaldu eta bertan *Save as* aukeratu. Han, disko gogorrean gordeko den irudiaren luzapen lehenetsia `fig` dela azaltzen da. Alabaina, luzapen-menua irekitzen baldin bada, luzapen franko begi-bistan geratuko dira, hala nola, ezagunenak aipatzearen, `bmp`, `jpg`, `pdf` eta `eps`, besteak beste. Segidan, irudiari izen bat eta atzizkia egokiturik, disko gogorrean gordeko da.

Esan gabe doa *Aginduetarako Leihotik* prozesu hori gauzatu daitekeela. Alde batetik, `print` agindua exekutatzeko baldin bada, kontuan izan behar da egindako irudia inprimagailu lehenetsian inprimaketa lortzen dela. Izan ere, `print` komandoak aukera batzuk ditu, eta, guztiak bistartzeko, aski da `help print` tekletzea. Haue da sintaxi nagusia:

```

    print -aukerak      EDO      print -aukerak irudiaren izena

```

Berbarako,

```

>> print -dwin

```

tekletatuz gero, irudiaren zuri-beltzeko kopia bat inprimagailu lehenetsira bidaltzen da. Edo

```

>> print -dwinc

```

exekutatzuz gero, aurreko prozesua errepikatzen da, baina orain berriz, koloretako kopia lortzen da. Orobat

```
>> print -deps
```

eginez gero, inprimagailutik ez da ezer jadetsiko. Inprimatu beharrean, irudia eps luzapeneko fitxategi batean gordeta egongo da; artxiboaren izena Matlab-ek hautatzen du, eta beheko mezu hau itzultzen du:

```
Warning: Files produced by the 'eps' driver cannot be sent to printer.
File saved to disk under name 'figure1.eps'.
```

Prozesua beste hainbeste aldiz errepikatzen da artxiboaren atzizkiak desberdinak baldin badira:

```
>> print -djpeg  jpg luzapena
>> print -dtiff  tiff luzapena
>> print -dpdf   pdf luzapena
>> print -depsc  eps baina koloretako kopia
>> print -dbmp   bmp luzapena
```

Bestalde matematika arloan, askotan *LateX* makroak formulak-eta idazteko enplegatzen dira. Hala, behin irudia gorderik, dokumentuan kapsulaturik egoteko, nahikoa da *LateX*-eko editorean honako kode hau idaztea:

```
\documentclass{article}
\usepackage[dvips]{graphicx}
.....
\begin{document}
.....
\includegraphics[width=7cm,height=!]{irudia.eps}
.....
\end{document}
```

Nola ez, esportatutako irudiaren izena zehaztu egin daiteke, baita beste aukera batzuk esplizitatu ere. Esaterako, gerta daiteke zenbait leiho irekita egotea, baina bakar bat gordetze-ko guraria izatea. Ondorengo kodean hirugarren leihoa gordeko da, fitxategiaren luzapena epsc izango da -hots, eps koloretan-, izena irudiM, eta bereizmena 300 izango da.

```
>> print -depsc -f3 -r300 irudiM
```

```
edo
```

```
print('-depsc', '-f1', '-r300', 'irudiM')
```

Aipatzekoa da, goiko elementuen ordena ez dela garrantzitsua; kasu honetan, atzizkia, bereizmena, izena eta irudiaren zenbakia.

Hori dela eta, erreparatu azken adibide honi, zeinetan for komandoari esker, era sekuentzialean bost irudi sortu eta gordetzen diren: irudi1.eps,...,irudi5.eps:

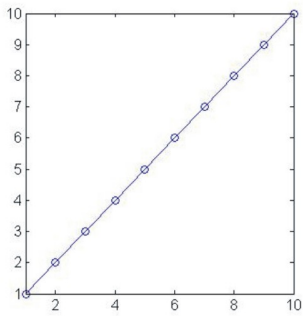
```
>> x=linspace(0,2*pi,100);
>> for i=1:5
    plot(x,cos(i*x));
    print('-depsc',['irudi' int2str(i) '.eps'])
end
```

Ohartzekoa da `int2str` agindua erabili behar izan dela {1, 2, 3, 4, 5} zenbakiak karaktere (*strings*) izan daitezzen, hau da, zenbaki oso batetik karaktere batera (integer to string) pasatzeko.

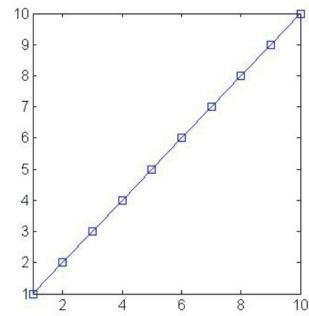
Bestaldetik, `Matlab`-en `saveas` agindua dago, eta haren bitartez, berriro ere, irudia fitxategi batean gorde daiteke; alabaina, gero `open` komandoarekin `Matlab`-en bertan birkargatu daiteke. Erreparatu etsenplu honi:

```
>> saveas(gcf, 'IRUDI', 'jpeg') % gorde uneko irudia  
>> open('IRUDI.jpg') % kargatu
```

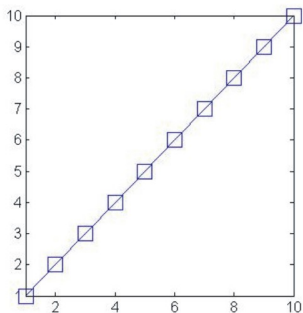
Azkenekoz, gogoratu `Matlab`-en sortutako irudiak `Word` edo `Power Point` programetan itsas daitezkeela. Hau da, `Matlab`-eko iruditik, hautatu `Edit` menu, zabaldu eta `Copy` as-en gainean klikatu. Hala, irudia `Windows` arbelera igarotzen da, eta beste programa batean itsatsi besterik ez da egin behar.



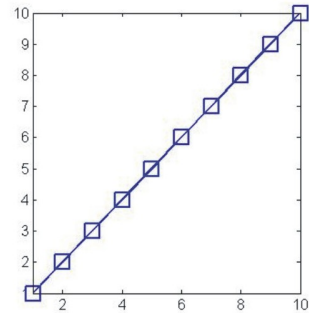
```
plot(1:10,'o-')
h=findobj
```



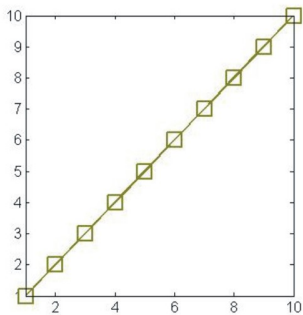
```
set(h(4),'Marker','square')
```



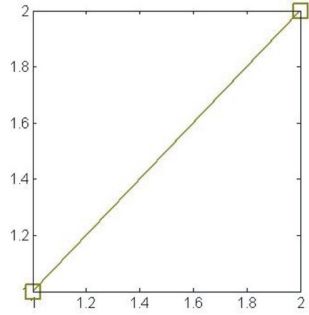
```
set(h(4),'MarkerSize',16)
```



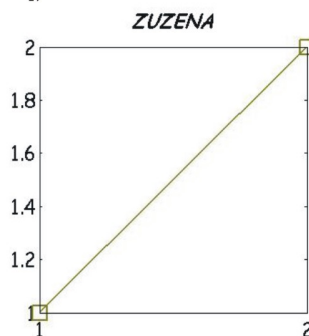
```
set(h(4),'LineWidth',2)
```



```
set(h(4),'Color',[0.5 0.5 1])
```



```
set(h(3),'XLim',[1 2])
set(h(3),'YLim',[1 2])
```



```
set(h(3),'fontsize',16)
set(h(3),'fontname','ComicSans ms')
p=title('ZUZENA')
set(p,'fontangle','oblique')
set(p,'fontweight','bold')
```

6.9 Irudia: Grafiko batetik abiatuta, objektu grafikoaren propietateak aldatzea.

7. Kapituluia

Polinomioak. Doikuntza eta interpolazioa

Kalkulu zientifikoetan, usu, polinomioak agertu ohi dira. Izan ere, funtzio sinpleenak dira, eta, kasu askotan, problemak ebazteko ekuazio polinomialak agertzen dira. Matlab-ek funtzio ugari du polinomiak manipulatzeko.

Bestalde, doikuntza-kurbak, normalean, datu esperimentalak modelizatzeko enplegatzen dira. Aintzakotzat hartzekoa da fabrikatutako kurbak ez duela zertan datu guztietatik pasatu, eta hurbilketa-errorea ahalik eta txikiena izatea kontrolatu behar dela. Hasiera-hasieratik, ohartzekoa da ez dagoela datuen egokitzapen-kurbaren ekuazioaren gaineko murrketarik, hau da, doikuntza polinomiala, edo esponentziala edota berreturazkoa izan daitekeela.

Interpolazio-prozesuan, datu batzuetatik abiatuz, beste balio batzuk zenbatesten dira. Interpolazio sinpleenean, puntuak konektatzeko, zuzenkiak erabiltzen dira; interpolazio konplexuagoetan, jatorrizko puntuez landa, zenbait datu osagarri gehiago usatzen dira. Prozedura horretan interpolazio-grafikoei jatorrizko puntuetatik igaro behar dute.

7.1 Polinomioak

Polinomioen forma nagusia eta ezaguna hau da:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

non $a_n, a_{n-1}, \dots, a_1, a_0$ zenbaki errealak koefizienteak diren, eta n zenbaki arrunta polinomioaren maila den.

Matlab-en polinomioak errenkada-bektoreak dira, eta bektoreetako elementuak polinomioaren koefizienteak dira: $a_n, a_{n-1}, \dots, a_1, a_0$. Hots, lehenengo elementua maila handieneko x aldagaiaren koefizientea da. Gainera, osatutako bektoreak kontuan eduki behar ditu tartean nuluak diren koefizienteak ere. Kasu baterako:

| Polinomioa | Matlab-eko bektore-adierazpena |
|-----------------|--------------------------------|
| $-2x + 5$ | <code>p=[-2 5]</code> |
| $x^2 - 2x + 12$ | <code>p=[1 -2 12]</code> |
| $-3x^2 - 1$ | <code>p=[-3 0 -1]</code> |
| $4x^4 - 2x$ | <code>p=[4 0 0 -2 0]</code> |

- **Polinomio baten balioztapena.** Erreparatu x puntuko polinomio baten balioztapena `polyval` funtzioaren bitartez gauzaten dela eta sintaxi hau duela:

$$\text{polyval}(p, x)$$

non p polinomioaren koefizienteez osatutako bektorea den, eta x balio bat edo bektore bat izan daitekeen, aukeran.

```
>> p=[-2 0 1 -1];
>> x=[-1 0 1];
>> polyval(p,x)
ans =
     0     -1     -2
```

Adibidean ikusten denez, polinomioa $-2x^3 + x - 1$ da eta $x = -1, 0, 1$ puntuetan batera balioztatzen da, eta irteera bektoriala lortzen da.

- **Polinomio baten erroak.** Polinomio baten erroak irudi nulua duten x balioak dira. Hala, $f(x) = x^2 - 5x + 6$ polinomioaren erroak $x = 2$ eta $x = 3$ dira. Matlab-ek, erroak lortzeko, `roots` izeneko funtzioa dauka:

$$\text{erro}=\text{roots}(p)$$

non p polinomioa den. Irteeran erro zutabe-bektorea ateratzen da, non, etsenplu honetan,

```
>> p=[1 -12 40 -17 -71 35];
>> erro=roots(p)
erro =
     6.4553
     3.8764
     2.3724
    -1.1967
     0.4927
```

erro guztiak errealak diren.

- **Erroetatik abiaturik, osatu polinomioa.** Polinomio baten erro oro ezagutzen baldin badira, alderantziko prozesua gauzatu daiteke, eta `poly(r)` erabiliz, hasierako polinomioa berreraiki daiteke, aginduaren idazkeran r bektoreak erroak dituelarik. Ikusi erakusbide hau:

```
>> erro=[1 -2 0 1];
>> p=poly(erro)
p =
     1     0    -3     2     0
```

Beraz, bilatutako polinomioa $p(x) = x^4 - 3x^2 + 2x$ da, eta haren erroak $\{1, -2, 0, 1\}$ dira. Bestalde, poly aginduaren argumentua A matrize bat baldin bada, matrizeari elkartutako polinomio karakteristikoa lortzen da, hau da, $\det(A - \lambda I)$.

```
>> A=[1 3 4; 0 1 -1; 0 1 1];
>> poly(A)
ans =
     1     -3     4     -2
```

Kasu honetan A -ren polinomio karakteristikoa $x^3 - 3x^2 + 4x - 2$ da.

- **Polinomioen arteko batura, biderkadura eta zatidura.** Bi polinomio batzeko, maila bereko koefizienteak batzen dira. Maila desberdineko polinomioak baldin badira, izan erne, zeren bektore laburrena (maila baxuagoko polinomioa) modifikatu behar baita, eta behar beste zero ipintzen baitira, polinomio biak maila berekoak izateko. Adibidez,

$$f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40 \quad f_2(x) = 3x^3 - 2x - 6$$

aurreko polinomioak batzeko ezinbestean hurrengo eran egin behar da:

```
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[0 0 0 3 0 -2 -6];
>> p=p1+p2
p =
     3     15     0     -7     -3     13    -46
```

Baturaz gain, biderkadura egiteko, Matlab-ek conv funtzioa dauka, eta, sintaxiari dagokionez, hau da:

$$\text{bider} = \text{conv}(a, b)$$

non a eta b bektoreek polinomioen koefizienteak dituzten. Biderkaduraren kasuan, polinomioen mailek ez dute berdinak izan behar, eta hala, aginduaren aplikazioa zabalitzen da:

```
>> p1=[1 1 1];
>> p2=[-2 1];
>> bider=conv(p1,p2)
bider =
     -2     -1     -1     1
```

Era berean, polinomio batzuen ondoz ondoko biderkadurak egiteko, conv komandoa behar beste aldiz errepikatu behar da.

Azkenik, zatidurari dagokionez, deconv agindua erabiltzen da. Sintaxi hau du:

```
[q, r]=deconv(u, v)
```

non u bektoreak zenbakitzailearen koefizienteak dituen, v bektoreak izendatzailearenak, q bektoreak zatiduraren koefizienteak eta r bektoreak hondarrarenak gordetzen dituen.

```
>> p1=[1 0 0 -1];
>> p2=[1 -1];
>> [q, r]=deconv(p1, p2)
```

```
q =
     1     1     1
r =
     0     0     0     0
```

hau da $\frac{x^3 - 1}{x - 1} = x^2 + x + 1$, kasu honetan hondarra nulua delarik.

- **Polinomio baten deribatua.** Polinomio baten deribatua egiteko, polyder komandoa dago. Agindu horrek hiru sintaxi ditu, zeren polinomio bakar bat, edo polinomioen arteko biderkadura edo zatidura deribatuzko erabil baitaiteke.

| | |
|------------------------|---|
| der=polyder(p) | Polinomio bakar baten deribatua kalkulatzeko. |
| der=polyder(a, b) | a eta b polinomioen biderkaduraren deribatua. |
| [ze, iz]=polyder(u, v) | ze u/v zatiduraren deribazioa egin osteko frazioaren zenbakitzailea, eta iz frazioaren izendatzailea. |

- **Frakzio baten deskonposizioa frakzio sinpleetan.** Matlab-ek, frakzio sinpleetako deskonposizioa egiteko, residue agindua dauka. Lehenengoan, [r, p, k]=residue(p1/p2) sententziarekin p1/p2 frakzioa zatiki sinpleen batura gisa deskonposatzen da. Matematikoki, eta residue aginduaren irteerarekin erkatuz gero,

$$\frac{p_1(x)}{p_2(x)} = \frac{r_1}{x - p(1)} + \frac{r_2}{x - p(2)} + \dots + \frac{r_n}{x - p(n)} + k(x)$$

hemen erraz identifikatzen da:

- r bektorean frakzio sinpleen zenbakitzaileak gordetzen dira.
- p bektorean izendatzaileen erroak gordetzen dira.
- k bektorean polinomio independentearen koefizienteak pilatzen dira.

Horrenbestez, $\frac{x^3 + x^2 + 1}{x^3 - 3x^2 + 4}$ frakzioa deskonposatu beharrez, honako kodea sartu behar da,

```
>> p1=[1 1 0 1];
>> p2=[1 -3 0 4];
```

```

>> [r,p,k]=residue(p1,p2)
r =
    3.8889
    4.3333
    0.1111
p =
    2.0000
    2.0000
   -1.0000
k =
     1
>> rats(r) % arrazionalizatu r-ren balioak
ans =
    35/9
    13/3
     1/9

```

non emaitzaren interpretazioa zera den:

$$\frac{x^3 + x^2 + 1}{x^3 - 3x^2 + 4} = \frac{35}{9(x-2)} + \frac{13}{3(x-2)^2} + \frac{1}{9(x+1)} + 1$$

7.2 Doikuntza-kurbak

Doikuntza-kurben kalkulua funtzio baten bitartez datu-multzo bat doitzean edo hurbiltzean datza. Gero, eraikitako funtzioa datuetarako eredu matematiko gisa erabil daiteke. Prozesua konplexua izan daiteke, kurba mota anitz daudelako: funtzio linealak, esponentzialak, logaritmikoak eta abar, besteak beste. Noiz edo behin, aldeztu aurretik zer funtzio-tipo interesatzen den informazioa baldin badago, horrek prozesua sinplifika dezake. Edozelan ere, normalean ez dago inondik inora informaziorik egoten, eta orduan askotariko adierazpide grafikoak egin ohi dira, erabiliko den funtzio-mota egokia hautatzeko.

7.2.1 Polinomioen bidezko egokitzapen-kurbak

Imajinatu planoan n puntu daudela; kasu horretan baliteke $n - 1$ mailako edo maila txikiagoko polinomio bat definitzea, nahi eta nahi ez puntuetatik pasatu behar ez duena. Doikuntza onena izateko, karratu txikienen metodoa da gehienbat aplikatzen dena. Metodo honetan soluzio-polinomioko koefizienteak lortzeko, datuen hondar-balioen karratuen batura minimizatzen da. Hemen, datu baten hondar-balioa datu horretako polinomioaren balioa minus datu erreala da. Adibidez, lau datu daude eta eman dezagun kalkulatu nahi dela zein den zuzenaren ekuazioa puntu horietara ondoen doitzen dena. Puntuak (x_1, y_1) , (x_2, y_2) , (x_3, y_3) , (x_4, y_4) baldin badira eta funtzio ezezaguna $f(x) = ax + b$ baldin bada, hondar-balioen karratuen batura-funtzioa gisa honetan eraikitzen da:

$$H = (f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + (f(x_3) - y_3)^2 + (f(x_4) - y_4)^2$$

eta, segidan, funtzio hori minimizatuz gero (deribatu partzialak nuluak direla exijituz), a eta b ezezagun erreal egokienak kalkulaten dira, betiere karratu txikien teknika aplikatuz.

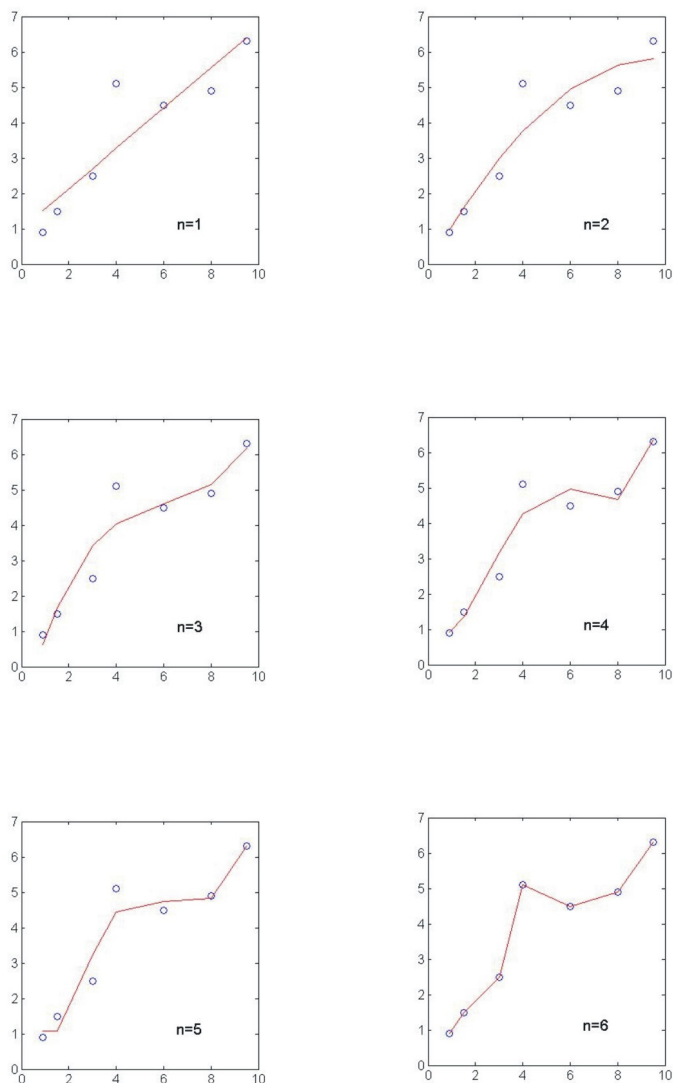
Garapen matematikoa alde batera utziaz, Matlab-ekin polinomio-doikuntzak erraz egin daitezke `polyfit` aginduari esker. Hau da sintaxia:

$$p = \text{polyfit}(x, y, n)$$

non p polinomio doitua den, x eta y datuen abzisak eta ordenatuak diren, eta n haututako doikuntza-polinomioaren maila den. Hala, $n = 1$ baldin bada, polinomioa zuzen bat izango da, edo $n = 2$ doikuntza koadratikoa egiten dela esaten da. Halere, n puntu baldin badira, polinomioaren maila maximoa $n - 1$ da, eta han dago muga aljebraikoa.

Gauzak horrela, 7.1 Irudian hasierako datuak (0.9, 0.9), (1.5, 1.5), (3, 2.5), (4, 5.1), (6, 4.5), (8, 4.9) eta (9.5, 6.3) dira; doikuntza-prozesuan maila desberdineko polinomioak erabiltzen dira, eta mailak letik 6ra aldatzen dira. Grafiko guztiak egiteko, aurretik deskribatutako `polyfit` agindua enplegatzen da, eta honatx erabilitako sententziak:

```
>> A=[0.9,0.9,1.5,1.5,3,2.5,4,5.1,6,4.5,8,4.9,9.5,6.3];
>> x=A(1:2:end); % abzisak
>> y=A(2:2:end); % ordenatuak
>> plot(x,y,'o') % datu errealak
>> p1=polyfit(x,y,1); % 1. mailako polinomio doitua
>> yp1=polyval(p1,x);
>> hold on,plot(x,yp1)
>> gtext('n=1')
>> p2=polyfit(x,y,2); % 2. mailako polinomio doitua
>> yp2=polyval(p2,x);
>> hold on, plot(x,yp2)
>> p3=polyfit(x,y,3); % 3. mailako polinomio doitua
>> yp3=polyval(p3,x);
>> hold on, plot(x,yp3)
>> p4=polyfit(x,y,4); % 4. mailako polinomio doitua
>> yp4=polyval(p4,x);
>> hold on, plot(x,yp4)
>> p5=polyfit(x,y,5); % 5. mailako polinomio doitua
>> yp5=polyval(p5,x);
>> hold on, plot(x,yp5)
>> p6=polyfit(x,y,6); % 6. mailako polinomio doitua
>> yp6=polyval(p6,x);
>> hold on, plot(x,yp6)
```



7.1 Irudia: Datuen askotariko doikuntza polinomialak, $n = 1$ etik $n = 6$ ra.

Argiro igartzen denez, polinomioaren maila 6 denean doikuntza zehatza da, polinomioa puntu guztietatik iragaten baita. Zenbat eta maila handiagoa izan, orduan eta doikuntza hobea lortzen da, dudarik ez.

Zernahi gisaz, polinomioez gain, egoera desberdinetan eta datuei begira, doikuntza egiteko bestelako funtzioak egokiagoak izan daitezke. Segidan, egokitzapenak egiteko ohikoak diren funtzio esponentzialak edo logaritmikoak, besteak beste, deskribatzen dira.

| | |
|--|------------------------|
| $y = bx^m$ | berretura-funtzioa |
| $y = be^{mx}$ edo $y = b10^{mx}$ | funtzio esponentziala |
| $y = m \ln(x) + b$ edo $y = m \log(x) + b$ | funtzio logaritmikoa |
| $y = \frac{1}{mx + b}$ | alderantzizko funtzioa |

Funtzio horien bidez eta `polyfit` erabiliz, doikuntzak egitea erraza da, betiere funtzioak era linealean berridazten baldin badira, hau da, $y = mx + b$ tankerako bihurtzen baldin badira, transformazio aljebraiko sinpleen bidez.

| | |
|---|------------------------|
| $\ln(y) = \ln(b) + m \ln(x)$ | berretura-funtzioa |
| $\ln(y) = \ln(b) + mx$ edo $\log(y) = \log(b) + mx$ | funtzio esponentziala |
| $y = m \ln(x)$ edo $y = m \ln(x) + b$ | funtzio logaritmikoa |
| $\frac{1}{y} = mx + b$ | alderantzizko funtzioa |

Nolabait esateko, linealizatu ostean, orain bai, m eta b konstanteak bilatzeko `polyfit(x,y,1)` komandoa aplikatu daiteke:

| Funtzioa | | <code>polyfit</code> aginduaren erabilera |
|----------------|------------------------|---|
| Potentzia | $y = bx^m$ | <code>polyfit(log(x), log(y), 1)</code> |
| Esponentziala | $y = be^{mx}$ | <code>polyfit(x, log(y), 1)</code> |
| Esponentziala | $y = b10^x$ | <code>polyfit(x, log10(y), 1)</code> |
| Logaritmikoa | $y = m \ln(x) + b$ | <code>polyfit(log(x), y, 1)</code> |
| Logaritmikoa | $y = m \log(x) + b$ | <code>polyfit(log10(x), y, 1)</code> |
| Alderantzizkoa | $y = \frac{1}{mx + b}$ | <code>polyfit(x, 1./y, 1)</code> |

Edozelan ere, doikuntza-funtzioa hautatzeko unean xehetasun batzuei arreta jarri behar zaie:

- Funtzio esponentzialak ez dira jatorritik pasatzen.
- Funtzio esponentzialak datu guztiak positiboak edo negatiboak direnean baino ezin dira erabili.
- Datuen abzisa negatiboa bada, funtzio logaritmikoa ez usatu, tarte horretan logaritmoa ez baita existitzen.
- Berretura-funtzian $x = 0$ ri $y = 0$ dagokio.
- $y = 0$ izanez gero, alderantzizko funtzioa ez enplegatu, indeterminazioa agertzen baita.

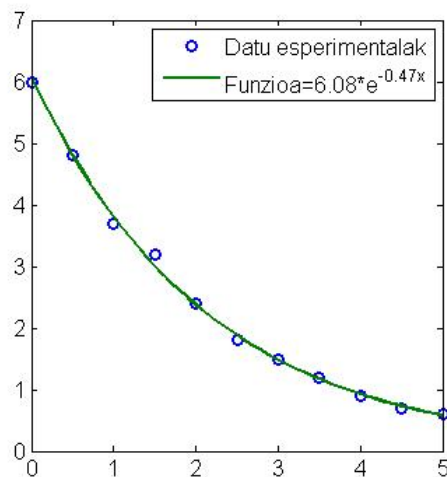
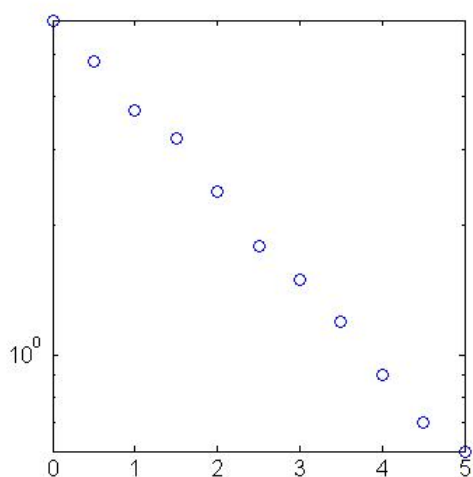
Datuen adierazpen grafikoari begira, hurrengo ariketan doikuntza esponentziala egitea deliberatzen da. Lehenik, `semilogy` aginduaren bitartez, hots, eskala logaritmikoa ordenatuetan eta lineala abzisetan, portaera linealaz ohartzen da; horren ondorioz, `polyfit` aginduaren bitartez, $y = b * e^{mx}$ funtzioaren b eta m parametroak doitzen dira.

```

>> x=[0:0.5:5];
>> y=[6 4.8 3.7 3.2 2.4 1.8...
      1.5 1.2 0.9 0.7 0.6];
>> semilogy(x,y,'o')
>> p=polyfit(x,log(y),1) % Polinomio lineala
p =
   -0.4707    1.8049
>> m=p(1)
m=
   -0.4707
>> b=exp(p(2))
b=
    6.0791
>> xm=[0:0.1:5];
>> ym=b*exp(m*xm);
>> plot(x,y,'o',xm,ym,'linewidth',2)
>> legend('Datu esperimentalak',['Funtzioa=', '6.08*e^-0.47x'])

```

Aurreko kodean doikuntza egiteaz aparte, segidako grafiko bi lortzen dira. Ezkerrekoan, OY ardatzean datuak eskala logaritmikoan irudikatzen dira, eta OX ardatzean eskala linealean; eskuinekoan, gainera, datuak eta lortutako doikuntza-kurba, biak batera eta eskala linealean. Izan ere, kasu honetan datuei so, esponentziala egokiena dela ondorioztatzen da.



7.3 Interpolazioa

Interpolazio-prozesuan ezaguturiko puntuen artean, beste balio batzuk zenbatesten dira. Matlab-en prozedura burutan ateratzeko, polinomioetan eta Fourier-en transformatuetan oinarrituriko funtzioak existitzen dira. Interpolazioan kontuan hartzeko da, eraikitzen den polinomioa hasierako puntuetik iragaten dela eta hori dela egokitzapen-teknikarekin duen

desberdintasun handiena. Segidan, labur bada ere, dimentsio bakarreko eta bi dimentsioko interpolazioa deskribatuko dira.

Dimentsio bakarreko interpolazioari dagokionez, imajinatu adibide sinpleena, hau da, bi puntu daudela. Kasu horretan, zuzen baten bitartez bil daitezke; izan ere, gero, zuzen horren ekuazioa beste puntu batzuen balioak zehazteko erabil daiteke. Hori da interpolazio-prozesua. Areago, hiru edo lau puntu ezaguturik, bigarren edo hirugarren mailako polinomioak osa daitezke, hurrenez hurren. Zenbat eta puntu gehiago, orduan eta maila altuagoko polinomioa izan behar da puntu orotatik pasatzeko. Bitxia bada ere, polinomioaren maila altua izateak ez du esan nahi balioen zenbatespenak hobeak izango direnik.

Dena den, ezaguturiko puntu guztiak erabili beharrean, interpolazioa egingo den ingurune baten datuak bakarrik enplegatzen badira eta bertan interpolazio polinomiala usatuz gero, teknika askozaz doiagoa da: segmentu-interpolazioa edo *spline* deitzen da. Hala, polinomioen maila baxuagoa izatea eta soilik datu gutxiri aplikatzea lortzen da. Emaitzen alde-tik, *spline* kubikoen erabilerak oso soluzio egokiak eta onak ematen ditu.

Beraz, prozesua nolabait zatika eginez gero, esaterako (x_i, y_i) eta (x_{i+1}, y_{i+1}) puntuetatik honako zuzen hau eraiki daiteke, eta segmentazio-interpolazio lineala hau da:

$$y = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}x + \frac{y_i x_{i+1} - y_i x_i}{x_{i+1} - x_i}$$

Bi puntuetatik pasatzen den zuzenak malda konstantea du, baina puntuz puntu aldatuz doa. Era berean, polinomio koadratikoak edo kubikoak erabiliz gero, kurba askoz samurragoa lor daiteke. Metodo horiek segmentazio-interpolazio koadratikoa edo kubikoa deitzen dira, hurrenez hurren. Polinomio horien koefizienteak tartez tartez kalkultzen dira, baina liburu honetan metodo teorikoa ez da deskribatuko. Zernahi gisaz, prozesuari buruzko informazioa zenbakizko kalkuluaren arloko edozein liburutan topa daiteke, dudarik ez.

Matlab-en dimentsio bakarreko interpolazioa `interp1` aginduaren bidez gauzatzen da, eta honako sintaxi hau du:

```
yi=interp1(x,y,xi,'metodoa')
```

non

- y_i balio interpolatuak diren.
- x bektoreak sarrerako puntuen koordenatu horizontalak dituen, y bektoreak sarrerako puntuen koordenatu bertikalak dituen eta x_i bektorearen osagaiak interpolazio-puntuen abzisak diren.
- 'metodoa' interpolazio-metodoa da, aukerakoa, eta honako karaktere hauek onartzen ditu:
 - 'nearest': puntu interpolatuetatik gertuen dauden puntuen balioak itzultzen dira.
 - 'linear': segmentazio-interpolazio lineala erabiltzen da.

- 'spline': segmentazio-interpolazio koadratikoa edo kubikoa enplegatzen da.
 - 'pchip': Hermite-ren interpolazio kubikoa usatzen da (cubic ere onartzen da).
- 'nearest' eta 'linear' metodoak aplikatzen direnean, xi-ren balioek x-en eremuaren barruan egon behar dute. Ostera, 'spline' edo 'pchip' usatuz gero, xi bektorearen osagai batzuk x bektorearen eremutik kanpo ager daitezke.
 - Lehenespenez, metodorik esplizitatu ez bada, 'linear' aplikatzen da.

Adibidez, hurrengo 7.1 Taulan segidako puntuetatik abiatuak, lortu 0:0.1:5 puntuen ordenatu interpolatuak, eta, horretarako, 'nearest', 'spline' eta 'pchip' metodoak aplikatu, hurrenez hurren.

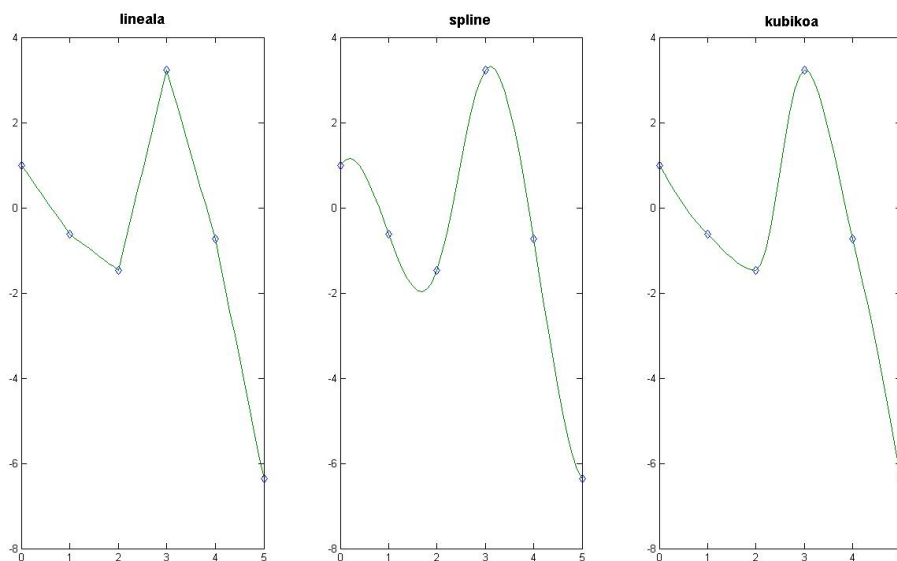
| x | 0 | 1 | 2 | 3 | 4 | 5 |
|---|-----|-------|-------|------|-------|-------|
| y | 1.0 | -0.62 | -1.47 | 3.24 | -0.74 | -6.37 |

7.1 Taula: Interpolazio-prozesurako hasierako datuak .

Taula honetan bost puntu daude, eta interpolazio-teknika 51 puntuetan aplikatuko da, betiere soluzioak hasierako puntuetatik pasatuz. Era sistematikoan problemaren enuntziatua hurrengo pausook ebatzen dute,

```
>> x=[0:1:5];
>> y=[1 -0.62 -1.47 3.24 -0.74 -6.37];
>> xi=0:0.1:5;
>> yilin=interp1(x,y,xi,'linear');
>> yispl=interp1(x,y,xi,'spline');
>> yikub=interp1(x,y,xi,'pchip');
>> subplot(1,3,1)
>> plot(x,y,'d',xi,yilin)
>> subplot(1,3,2)
>> plot(x,y,'d',xi,yispl)
>> subplot(1,3,3)
>> plot(x,y,'d',xi,yikub)
>> gtext('lineala')
>> gtext('spline')
>> gtext('kubikoa')
```

eta hona hemen 7.2 Irudian lorturiko exekuzioaren irteera grafikoa:



7.2 Irudia: Hasierako datuen interpolazio lineala, *spline* interpolazioa eta interpolazio kubikoa.

Azkenik, bi dimentsioko interpolazioa egiteko, `interp2` agindua dago, eta haren sintaxia hau da:

`Z matrizea=interp2(X,Y,Z,X matrizea, Y matrizea, 'metodoa')`

den. Kasu honetan, `jo` sarrerako datuek taulaturik egon behar dutela:

- `X` eta `Y` matrizeek sarrerako datuen sareta bat definitzen dute.
- `Z` matrizeak `X` eta `Y` matrizeei dagozkien balioak gordetzen ditu.
- `X` matrizeak eta `Y` matrizeak `Z`-ren balio zenbatetsiak non zehaztu nahi diren markatzen dute.
- '`metodoak`' interpolazioan erabiltzen den metodoa seinalatzen du, eta, aukera horretan, bai '`linear`' (interpolazio bilineala), bai '`cubic`' (interpolazio bikubikoa) metodoak onartzen dira.
- `Z` matrizeak balio zenbatetsiak ditu.

Ideiak argitzeko, demagun ondorengo sarrera biko taula hau dagoela eta bi dimentsioko interpolazioa usatuz (3, 2) puntuari dagokion `Z`-ren balio interpolatua bilatu nahi dela.

| <code>y \ x</code> | 2 | 4 |
|--------------------|----------|----------|
| 1 | 3 | 15 |
| 3 | 11 | 23 |

Sententzia hauek egikaritzuz gero, ebatzi egiten da:

```
>> x=[2 4];  
>> y=[1 3];  
>> [X,Y]=meshgrid(x,y);  
>> Z=[3 15; 11 23];  
>> Zestim=interp2(X,Y,Z,3,2)  
Zestim =  
    13
```

Beraz, (3, 2) puntuari dagokion z zenbatetsia 13 da. Adibide honetan interpolazio bilineala erabili da, ezen cubic aukerak gutxienez 3x3 dimentsioko sareta behar du. Halaber, hiru dimentsioko interpolazioa egin daiteke, eta horretarako, `interp3` agindua dago. Dena den, erabilera arras antzekoa bada ere, haren aplikazioa liburutik kanpo geratzen da.

8. Kapituluua

Kalkulu sinbolikoa

Honaino egindako eta aurkeztutako eragiketa matematikoak zenbakizkoak izan dira. Askotariko adierazpenak idatzi dira, eta zenbakiak eta aldagaiak agertzen ziren. Aplikazio matematiko, zientifiko eta tekniko askotan eragiketa sinbolikoak behar dira, hots, aldagai sinbolikoz osatutako adierazpenekin eragiketa matematikoak egin behar dira. Aldagai sinbolikoak exekutatzeko direnean, ez dago inolako zenbakizko baliorik. Adierazpen sinbolikoaren emaitza aldagai sinbolikoak dituen adierazpen matematikoa da. Horren gaineko adibide xume bat hau da: imajinatu $ax - b = 0$ ekuazioa, a , b eta x aldagai sinbolikoak dituen, eta hortik x aldagaia askatu nahi dela. Garbi dago emaitza $x = b/a$ dena, baina prozesuan aldagaiek ez dute zenbakizko baliorik gordetzen, hau da, ebazpena sinbolikoa da. Beste adibide batzuk $x^3 - x + 1$ polinomioaren deribazioa edo integrazioa izan daitezke.

Matlab-ek, kalkulu sinbolikoa eta matematikoa egiteko, *Symbolic Math* toolbox-a dauka: biblioteka berezi bat, hain justu. Azken finean, liburutegia Matlab-eko funtzio-bilduma bat baino ez da, eta eragiketa sinbolikoak gauzatzeko enplegatzen da. Horri esker, erraz formulak eta adierazpen aljebraikoak manipulatu eta eragiketa gehienak egin daitezke. Berbarako, polinomio eta adierazpen arrazionalak zein trigonometrikoak faktORIZATU, sinplifikatu edo garatu egin daitezke. Halaber, ekuazio polinomikoen edo sistemen soluzio aljebraikoak aurki daitezke. Era berean, deribatuak balioztatu daitezke, edo integral sinbolikoa egin, edo ekuazio diferentzialen soluzioak bilatu daitezke. Serieak batu, edo limiteak lortu, eta abar; horiek dira, besteak beste, erraminta horren aplikazioak.

Ohartzekoa da, adierazpen sinbolikoan zenbakizko zatia agertuz gero balio zehatza erabiltzen dela, hau da, ez dagoela zenbakizko balioen hurbilketarik. Esaterako, $x/3$ eta $x/4$ batzen baldin badira emaitza $0.5833x$ izan beharrean, $7/12x$ da. Orobat, eragiketa sinbolikoetan erabiltzen diren komando eta funtzioen estiloa eta sintaxia, eta zenbakizko eragiketetan azaltzen dena antzekoak dira. Jakin beharra dago, barrutik kalkulu sinbolikoa Maple[®]-k egiten duela, software hori horretarako diseinatutako programa baita. Izan ere, Maple Matlab-en inkorporatuta dago, eta eragiketa sinbolikoren bat exekutzatzen den bakoitzean automatikoki aktibatzen da. Esan beharra dago Maple software-a independentea ere badela baina beraren egitura eta komandoak desberdinak direla. Ordenagailuan eragiketa sinbolikoen biblioteka instalatuta dagoenez jakiteko, aski da `ver` tekletzea. Horrela erabil daitezkeen *toolbox* guztiak zerrendatzen dira, *Symbolic toolbox* barne.

8.1 Objektu eta adierazpen sinbolikoak

Objektu sinboliko bat, aldagai bat, edo zenbaki bat, edo adierazpena bat aldagai edo zenbaki sinbolikoez eratua izan daiteke. Adierazpen sinbolikoak objektu sinbolikoak dituzten adierazpen matematikoak dira. Izan ere, adierazpen sinbolikoak eta zenbakizkoak antzekoak dira teklatean direnean. Halere, objektu sinbolikoak agertzen direnez geroztik, Matlab-ek era sinbolikoan exekutatuko ditu eta berezitasun hori kontuan hartu behar da.

Esan berri denez, objektu sinbolikoak aldagaiak edo zenbakiak izan daitezke, eta haiek sortzeko erabiltzen diren aginduak `sym` edo `syms` dira. Objektu sinboliko bat baino ez bada, `sym` erabiliko da sintaxi honi jarraituz:

```
izena=sym('katea')
```

non `katea` honako hau izan daitekeen:

- Letraz osatutako `katea`, hala nola `'a'`, edo `'ab'`.
- Letraz eta zenbakiz eratutako `katea`, betiere lehenengo letra bat dela, kasurako, `'s1s2'` edo `'obi1'`.
- Zenbaki bat, esaterako, `'123'`.

Horrela `izena` deritzon objektua sortzen da, `katea` objektu sinbolikoa gordetzen duena. Bestalde, gogoan izan da komeni dela objektuaren `izena` eta `katearena` berdinak izatea, nahitaezkoa ez bada ere.

```
>> a=sym('a')
a =
a
>> b=sym('12')
b =
12
>> c=12
c =
12
```

Ikusten denez, zenbakizko aldagaietan ez bezala, komando horien irteeran ez dago koskarik. Era berean, aldagai sinboliko bat baino gehiago definitu nahi izanez gero, `syms` komandoa usatzen da, aldagaiak bereizlerik gabe bananduta.

```
syms aldag1 aldag2 aldag3....
```

```
>> syms a b c d e f
```

Aipatzekoa da aldagai sinbolikoak sortzen direla baina ez direla automatikoki bistaratzen, amaieran puntu eta koma jarri ez arren.

Adierazpen sinbolikoetan aldagai sinbolikoak agertzen dira; izan ere, adierazpen sinbolikoak berak objektu sinbolikoak dira, eta honela eraiki daitezke:

adierazpena=adierazpen matematikoa

Hona hemen zenbait adibide:

```
>> syms x y a b
>> fun=a*x^2-b*x+a*b % orain fun objektu sinbolikoa da
fun =
a*x^2-b*x+a*b
```

edo

```
>> deri=2*a/3*x^2-4*3/7*x-a*b*2*7/4
deri =
2/3*a*x^2-12/7*x-7/2*a*b
```

Azken horretan eta portaera orokortzat harturik, Matlab-ek eragiketak sententziak sartu ahala egiten ditu, baina zatikiak itzultzen dira, eta ez zenbakizko balioak. Segidako adibidean, aldagai sinbolikoekin eta zenbakizko aldagaiekin eragiketa bera egingo da, kalkulu doiaren eta hurbilduaren arteko desberdintasuna agerian uzteko:

```
>> a=sym('12');b=sym('20');
>> c=b/a-log(10) % kalkulu sinbolikoa
c =
-2147941490884513/3377699720527872
>> d=12;e=20;
>> f=e/d-log(10) % zenbakizko kalkulua
f =
-0.6359
```

Adierazpen batean, eta era nahasian, zenbakizko objektuak eta sinbolikoak ager daitezke, baina, orduan, eragiketa matematiko guztiak zehatzak izango dira, azken batean emaitza adierazpen sinbolikoa izango baita.

Objektu sinboliko baten balioa, era zehatzean idatzita dagoena, zenbakizko erara transformatzeko, double agindua dago.

```
>> y=sin(pi/23)-sym('12')
y =
-213719821833263227/18014398509481984
>> y2=double(y)
y2 =
-11.8638
>> y3=sin(pi/23)-12
y3 =
-11.8638
>> isequal(y2,y3) % biak alderatzeko
ans =
1
```

Hamargarren lerroan, `isequal` komandoa enplegatu da objektu sinbolikoaren zenbakizko balioa eta zenbakizko adierazpenaren emaitza berdinak direla egiaztatzeko.

Halaber, badago beste modu bat adierazpen sinbolikoa balioztatzeko eta zenbakizko emaitza lortzeko: `var` agindua (*variable precision arithmetic*) erabiltzea. Harekin, erabiltzaileak irteeraren zehaztasuna hautatzen du; lehenetsitakoa hogeita hamabi zifra da.

```
>> vpa(1/12*exp(pi))
ans =
1.9283910527316054928093080889084
>> vpa(1/12*exp(pi),3)
ans =
1.93
>> vpa(1/12*exp(pi),5)
ans =
1.9284
>> f = vpa('70!',200)
f =
11978571669969891796072783721689098
73645893814254642585755536286462800
95827898453196800000000000000000.
```

Bestalde, aldezturik objektu sinboliko moduan ez zehaztutako aldagaiek objektu sinboliko bat definitu ahal dute, esate baterako,

```
>> f=sym('a*x^3-b*x+c')
f =
a*x^3-b*x+c
```

baina, orain kontuz, `a`, `b`, `c` eta `x` aldagaiak objektu sinboliko independenteak ez baitira; horrenbestez, objektuko aldagai bakoitzarekin elkartutako eragiketa sinbolikoak egiterik ez dago.

Bestenaz, adierazpen sinboliko batean dauden aldagai sinbolikoak identifikatzeko, `findsym` agindua dago.

```
>> syms a b c d e x y
>> f=1;g=2;
>> h=f*a-b*c*g-x*g-sin(2)*y*d
h =
a-2*b*c-2*x-4095111552621091/4503599627370496*y*d
>> findsym(h)
ans =
a, b, c, d, x, y
>> findsym(h,3)
ans =
x,y,d
```

Komando honen egikaritzean ikusi denez, aldagai sinboliko batzuk edo guztiak topatzen dira; esaterako, bederatzi lerroan gertatzen denez, lehenengo hirurak x -tik abiaturik itzultzen dira.

Oro har, adierazpen sinbolikoak manipulatu egin daitezke, eta, berbarako, berretzaile bereko monomioak taldekatzea edo identitate trigonometrikoak sinplifikatzea, edo faktore komuna ateratzea dira, besteak beste, gauzatu daitezkeen aldaketak. Segidan, horietako batzuk deskribatuko dira:

- `collect` aginduak adierazpen bateko berretzaile bereko elementuak taldekatzen ditu, eta elementuak potentziaren ordena beherakorrean sailkatuko dira.

`collect(adierazp)` edo `collect(adierazp,aldagaia)`

Hala aldagaia islatzen baldin bada, adierazpen sinboliko batean zein aldagairekiko egingo den ordena aukeratu daiteke.

```
>> syms x y
>> z1=(x^2-2*x+1)*(x^4-y^3-y^2+x*y)
z1 =
(x^2-2*x+1)*(x^4-y^3-y^2+x*y)
>> sim=collect(z1) % lehenetsiz x-ekiko ordena beherakorra
sim =
x^6-2*x^5+x^4+y*x^3+(-y^3-y^2-2*y)*x^2+(y+2*y^3+2*y^2)*x-y^3-y^2
>> sim=collect(z1,y) % y-rekiko ordena
sim =
(-x^2+2*x-1)*y^3+(-x^2+2*x-1)*y^2+(x^2-2*x+1)*x*y+(x^2-2*x+1)*x^4
```

- `expand` aginduak, propietate banakorra aplikatuz, adierazpen sinbolikoak garatzen ditu. Halaber, helburu horrekin, behar izanez gero, identitate trigonometrikoak, edo esponenzialak edo logaritmitikoak usatzen ditu.

```
>> syms x y
>> a=sin(x-y)
a =
sin(x-y)
>> expand(a)
ans =
sin(x)*cos(y)-cos(x)*sin(y)
```

- `factor` aginduak adierazpen sinboliko polinomiala faktorizatzen du, eta erro errealak baino ez ditu enplegatzen.

```
>> syms x
>> M=x^4-2*x^3+5*x^2-2*x-2
M =
x^4-2*x^3+5*x^2-2*x-2
>> factor(M)
ans =
(x-1)*(x^3-x^2+4*x+2)
```

- `simplify` eta `simple` aginduak adierazpen sinbolikoak sinplifikatzeko erabiltzen dira. Lehenengoak, adierazpen baten era sinpleena ateratzeko, identitate trigonometrikoak, logaritmikoak, edo eragiketa matematikoak eta abar usatzen ditu.

```
>> syms x y
>> simplify(cos(x)^2-sin(x)^2)
ans =
2*cos(x)^2-1
```

edo

```
>> simplify((x+y)/(1/x+1/y))
ans =
x*y
```

Orobat `simple` komandoak ere irteera ahalik eta karaktere kopuru txikienarekin aurkezten du, beharbada adierazpen sinpleena ez bada ere.

```
>> syms x
>> f=(x^3+x^2-x-1)/(x^2-2*x+1)
f =
(x^3+x^2-x-1)/(x^2-2*x+1)
>> simple(f)
```

```
simplify:
(x^2+2*x+1)/(x-1)
radsimp:
(x^2+2*x+1)/(x-1)
combine(trig):
(x^2+2*x+1)/(x-1)
factor:
(x+1)^2/(x-1)
expand:
1/(x^2-2*x+1)*x^3+1/(x^2-2*x+1)*x^2-1/(x^2-2*x+1)*x-1/(x^2-2*x+1)
combine:
(x^3+x^2-x-1)/(x^2-2*x+1)
convert(exp):
(x^3+x^2-x-1)/(x^2-2*x+1)
```

```

convert(sincos):
(x^3+x^2-x-1)/(x^2-2*x+1)
convert(tan):
(x^3+x^2-x-1)/(x^2-2*x+1)
collect(x):
(x^3+x^2-x-1)/(x^2-2*x+1)
mwcos2sin:
(x^3+x^2-x-1)/(x^2-2*x+1)
ans =
(x+1)^2/(x-1)

```

Komando honen bitartez agerian dago irteera desberdinak lortzen direla, sinplifikazio-metodoaren arabera. Soluzioa hain luzea ez izateko, [irteera metodoa]=simple(adierazpena) egitura erabiltzea proposatzen da. Saia zaitez.

- pretty aginduaren bidez adierazpen sinbolikoak betiko era aljebraikoan izatea lortzen da.

```

>> syms x y
>> M=log(x*y^2)-(sin(x+y))^2
M =
log(x*y^2)-sin(x+y)^2
>> pretty(M)

```

$$\log(x y^2) - \sin(x + y)^2$$

Behin Matlab-ek adierazpen sinbolikoa sortuta, interesgarria izan daiteke aldagai sinbolikoei zenbakiak esleitzea eta ordezkatzeta, bereziki adierazpen horien zenbakizko balioak lortzeko. Horretarako, subs agindua dago, hain zuzen, eta agindua modu batean edo gehiagotan erabil daiteke:

- Adierazpen sinboliko batean balio edo bektore bat ordezkatzeta posible da; kasu horretan, sintaxia hau da:

```
R=subs(S,aldagaia,balioak)
```

non S adierazpeneko aldagaia balioez ordezkutzen den.

```

>> syms x y
>> S1=x^2-x+sin(x)
S1 =
x^2-x+sin(x)
>> subs(S1,x,[1 0]) % jadanik zenbakizko irteera
ans =
    0.8415    0
>> S2=x^2-x*y+y*sin(x-y)

```

```
S2 =
x^2-x*y+y*sin(x-y)
>> subs(S2,x,[1 0]) % irteera sinbolikoa
ans =
[ 1-y-y*sin(-1+y),      -y*sin(y)]
```

Adibidean igartzen denez, azpimarratzekoa da adierazpenean aldagai sinboliko bat baino ez badago zenbakizko emaitza ateratzen dela. Ostera, adierazpenean elementu sinboliko bat baino gehiago agertuz gero, irteera sinbolikoa da; kasu honetan, bektore sinboliko bat, hain zuzen.

- Bestalde adierazpen sinboliko batean aldagai sinboliko baten baino gehiagoren balioak ordezkatu daitezke, idazkera zerbait aldatuagatik ere:

```
R=subs(S,aldag1,aldag2,...,aldagN,balio1,balio2,...,balioN)
```

Idazkera honetan xehetasun batzuk gogoan izan behar dira:

- Balioak, aldagai sinbolikoetan ordezkaturiko direnak, eskalarrak edo bektoreak edo matrizeak izan daitezke. Lehenengo balio1 balioa aldag1 aldagaian ordezkaturiko da, eta gero balio2 aldag2 aldagaian eta prozesu bera n. aldagaira ailegatu arte.
- Ordezkatutako balio oro eskalarrak baldin badira, irteera zenbaki bat edo adierazpen sinboliko bat (aldagai sinbolikoren bat ordezkatu ez bada) izango da.
- Gainera, ordezkatutako balioen bat matrizea baldin bada, kontuan hartu behar da eragiketa guztiak elementuz elementu egingo direla. Horrenbestez, aldagai sinbolikoen balioek tamaina bereko matrizeak izan behar dute.
- Aldagai batzuetan baliteke tamaina bereko matrizeak ordezkatzeko eta beste batzuetan eskalarrak sartzea. Kasu horretan, Matlab-ek barrutik eskalarretarako tamaina bereko matrizeak, elementu errepikatuak dituztenak, eraikitzen ditu. Izan ere, prozedura horri esker, elementuz elementuko eragiketak egitea ahalbidetzen da.

```
>> syms a b c x y
>> f1=a*b*x-y*exp(c)
f1 =
a*b*x-y*exp(c)
>> subs(f1,a,b,c,x,y,{1 1 0 1 -1})
ans =
      2
>> subs(f1,a,c,[0 1 2],[0 -1 1])

ans =

[      -y,  b*x-y*exp(-1), 2*b*x-y*exp(1)]
>> subs(f1,a,b,c,x,y,[1 0],[1 0],[0 1],[1 0],[0,-1])
```

```
ans =
    1.0000    2.7183
```

8.2 Kalkulu sinbolikoaren aplikazio matematikoak

Ohiko eragiketa matematikoak kalkulu sinbolikoaren bidez egin daitezke. Segidan, adibideen bitartez, aplikazio matematikoak deskribatuko dira, baina, puntu honetan aurrera joan orduko, irakurleari Matlab-eko funtool erreminta interesgarria proposatzen zaio, zeren horren bitartez ondoan ikusiko diren eragiketa batzuk modu elkarreragilean gauzatu baitaitezke.

- Lehenengo aplikazioa ekuazioen eta ekuaziozko sistemen ebazpena da. Hori zuzenki gauzatzeko, solve agindua dago. Alde batetik, ekuazioei dagokienez, sintaxi orokorra hau da:

```
solve(ekuazioa,aldagaia)
```

non ekuazio batetik zein den askatu nahi den aldagaia zehaztu daitekeen. Izan ere, ekuazio aljebraiko batean aldagai sinboliko bat baino gehiago ager daitezke, eta aldagai bat baino ez badago, zenbakizko soluzioa lortzen da.

- Ekuazioan $f(x) = g(x)$ formako ekuazioak, = ikurra barne eta kate gisa, sar daitezke.
- Ekuazioak soluzio bat baino gehiago baldin badu, irteera zutabe-bektore sinbolikoa da.

Erreparatu adibide hauei:

```
>> h=solve('x^2-2*y*x+1','x')
h =
    y+(y^2-1)^(1/2)
    y-(y^2-1)^(1/2)

>> pretty(h)
      [      2      1/2]
      [y + (y  - 1)  ]
      [              ]
      [      2      1/2]
      [y - (y  - 1)  ]

>> h=solve('x^2-2*x+1')
h =
    1
    1

>> h=solve('cos(2*x)+3*sin(x)=2')
```

```

h =
  1/2*pi
  1/6*pi
  5/6*pi

```

Bestalde, ekuazio-sistemak ere ebazteko, `solve` erabil daiteke. Aldagai kopurua eta ekuazio kopurua bat etorri gero, zenbakizko soluzioa izango da. Bestetik, aldagai kopurua ekuazio kopurua baino handiagoa baldin bada, soluzio sinbolikoa ateratzen da. Honatx egoera:

```
solve(eku1,eku2,...,ekun,aldag1,aldag2,...,aldagm)
```

Kontuan hartu ekuazioak sinbolikoak direla eta `=` ikurra idatzi ezik, zerora berdindurik daudela suposatzen dela. Berebat, `solve` aginduaren irteera bi formatutan ager daiteke, edo gelaxka-matrize erara edo `Matlab`-eko egitura modura.

Komandoaren irteera gelaxka-matrizea denean, honako sintaxi hau dauka (imajinatu bi ekuazioko sistema):

```
[aldA,aldB]=solve(eku1,eku2)
```

```

>> syms x y a
>> Ek1='2*x-y=0';
>> Ek2='a*x-y=a';
>> [xb,yb]=solve(Ek1,Ek2)
xb =
  1/(a-2)*a
yb =
  2/(a-2)*a

```

Izatez, aurreko adibidean `x` eta `y` aldagaiak askatu egin dira; halarik ere, beste aldagai bi aska zitezkeen:

```

>> [aa,xx]=solve(Ek1,Ek2,a,x)
aa =
  2*y/(y-2)
xx =
  1/2*y

```

Kontuz, ezen goiko etsenpluan `a` eta `x` aldagai sinbolikoak ebazten dira, baina irteeran emaitzak ordena alfabetikoaren arabera itzultzen dira.

Komandoaren irteera `Matlab`-eko egitura denean, hau da sintaxia (imajinatu hiru ekuazioko sistema):

```
EG=solve(eku1,eku2,eku3)
```

non EG egituraren izena den. Agindua exekutatzen denean, egituraren izena eta eremu horren izenak bistartzen dira, baina ez eremu bakoitzaren balioa. Eremu baten balioa bisualizatzeko, erabiltzaileak egituraren izena. eremuaren izena egitura tekleatu behar du.

```
>> syms x y z
>> eku1='2*x-y+z=1';
>> eku2='x+2*y-2*z=0';
>> eku3='-2*x+3*y-6*z=3';
>> Sol=solve(eku1,eku2,eku3) % Sol egitura
Sol =
    x: [1x1 sym]
    y: [1x1 sym]
    z: [1x1 sym]
>> Sol.x % x-en balioa
ans =
    2/5
>> Sol.y % y-en balioa
ans =
   -5/3
>> Sol.z % z-en balioa
ans =
  -22/15
```

- Matlab-en deribatuen kalkulu sinbolikoa egin daiteke, eta `diff` agindua enplegatzen da. Sintaxiari dagokionez, hau da orokorra:

```
diff(S,aldagaia,n)
```

non S adierazpen sinbolikoa aldagaiarekiko n bider deribatzen den. Adierazpean aldagai sinboliko bat baizik ez badago, aski da `diff(S,n)`. Areago, aldagai sinboliko bat agertu eta lehenengo ordenako deribatua egiteko, nahikoa da `diff(S)` exekutatzea. Erreparatu adibide honi:

```
>> syms x y z
>> f=log(x^2+y* 2+z^2);
>> a=diff(f,x,2) % f-ren bigarren ordenako deribatua x-ekiko
a =
    2/(x^2+2*y+z^2)-4*x^2/(x^2+2*y+z^2)^2
>> b=diff(f,y,2) % f-ren bigarren ordenako deribatua y-ekiko
b =
   -4/(x^2+2*y+z^2)^2
```

```

>> c=diff(f,z,2) % f-ren bigarren ordenako deribatua z-ekiko
c =
2/(x^2+2*y+z^2)-4*z^2/(x^2+2*y+z^2)^2
>> a+b+c
ans =
4/(x^2+2*y+z^2)-4*x^2/(x^2+2*y+z^2)^2-4/(x^2+2*y+z^2)^2
-4*z^2/(x^2+2*y+z^2)^2
>> simplify(a+b+c)
ans =
4*(2*y-1)/(x^2+2*y+z^2)^2

```

- Integrazio sinbolikoa, bai integrazio mugagabea, bai integrazio mugatua, `int` aginduari esker egiten da. Orobat, integrazioa sinplea edo anizkoitza izan daiteke. Integrazio mugagabea baldin bada, sintaxia hau da:

$$\text{int}(S, \text{aldagaia})$$

non S adierazpen sinbolikoa den, eta aldagaia parametroak zeinekiko integratzen den islatzen duen.

```

>> syms x t
>> f=2*x^2-t+1;
>> int(f,x) % f-ren integrazioa x-ekiko
ans =
2/3*x^3-t*x+x
>> int(f,t) % f-ren integrazioa t-ekiko
ans =
2*x^2*t-1/2*t^2+t

```

Aurrekoan integrazio sinplea egin da, baina, segidako adibidean erakusten denez, integrazio anizkoitza ere egin daiteke,

```

>> syms x y
>> f=x*cos(y);
>> int(int(f,y),x)
ans =
1/2*x^2*sin(y)

```

non $\int \int x * \cos(y) dy dx$ integral bikoitzaren emaitza lortzen den.

Integral anizkoitzak eta mugatuak ere kalkula daitezke; sintaxiaren aldetik, hona hemen `int` komandoaren idazkera:

$$\text{int}(S, \text{aldagaia}, a, b)$$

zeinetan a eta b integrazio-mugak diren. Erreparatu $\int_0^\pi \cos(x) - 3x^3 dx$ integralaren balioa kalkulatzeko behar den kodeari:

```
>> syms x
>> int(cos(x)-3*x^3,x,0,pi)
ans =
-3/4*pi^4
```

Lehen bezala integral anizkoitz mugatuak baliozta daiteke. Kasu honetan, honako integralaren emaitza hau ematen da:

$$\int_0^\pi \left(\int_0^\pi \sin(x+y) dx \right) dy$$

```
>> syms x y
>> int(int(sin(x+y),x,0,pi),y,0,pi)
ans =
0
```

Integrazioa, askotan, prozesu zaila izan daiteke. Baliteke soluziorik ez existitzea, eta orduan Matlab-ek integrala gehi informazio-mezu bat itzultzen ditu: *Explicit integral could not be found* mezua agertzen da.

- Gisa berean, kalkulu sinbolikoan ekuazio diferentzialak eta sistema diferentzialak ebatz daitezke. Ekuazio diferentzialak `dsolve` aginduari esker ebatzen dira. Areago, `dsolve`-rekin soluzio orokorretik abiatuta, hasierako baldintza espezifikaturik edo mugako baldintza inposaturik, soluzio partikularra lor daiteke. Dena den, baldintzarik gabeko ekuazio diferentziala ebazteko,

`dsolve('ekuazio','aldagaia')`

exekutatu behar da.

- Ebatzi nahi den elementua ekuazioa da. Nahiz eta aldagaiak objektu sinbolikoak izan, karaktere gisa sartu behar dira.
- `dsolve('ekuazio')` bakarrik egikaritzuz gero, aldagai lehenetsia `t` aldagaia da.
- `D` letrak ekuazioan sartzen denean, letra horrek deribazioa adierazten du. Hau da, `y` aldagai dependentea baldin bada eta `x` aldagai independentea baldin bada, `Dy` idazkerak dy/dx adierazten du, edo `D2y`-k d^2y/dx^2 , etab.
- Baldintzagabeko ekuazio diferentzialen soluzioan `C1,C2,...` integrazio-konstanteak agertzen dira, ekuazio diferentzialaren ordenaren arabera.

```
>> dsolve('D2y+3*Dy-2*y=0')
ans =
C1*exp(1/2*(-3+17^(1/2))*t)+C2*exp(-1/2*(3+17^(1/2))*t)
>>
```

Adibide honetan beheko ekuazio diferentzialaren soluzioa lortzen da:

$$\frac{d^2y}{dx^2} + 3\frac{dy}{dx} - 2y = 0$$

Soluzio partikularrak lortzeko, integrazio-konstanteen menpe ez daudenak, baldintza zenbait aplikatu behar dira. Horrela, bat ordenako ekuazio diferentzialak baldintza bat behar du, edo bi ordenakoek bi baldintza, eta horrela ondoz ondo.

```
dsolve('ekuazioa', '1.baldintza', '2.baldintza, . . . .', 'aldagaia')
```

non 'aldagaia' argumentua aukerakoa den, eta ekuazioko aldagai independentea definitzeko enplega daitekeen; dena den, baliorik zehaztu ezean, aldagai independentea t letra dela suposatzen da. Esaterako:

| Era matematikoa | Matlab era |
|-----------------|------------|
| $y(a) = A$ | 'y(a)=A' |
| $y'(a) = A$ | 'Dy(a)=A' |
| $y''(a) = A$ | 'D2y(a)=A' |

Berbarako, honako kodea sartu behar da, $\frac{d^2y}{dt^2} - 2\frac{dy}{dt} + 1 = \cos(t)$, $y(0) = 0$, $y'(0) = 1$ ebaztearren.

```
>> syms y x
>> dsolve('D2y-2*Dy+1=cos(t)', 'y(0)=0', 'Dy(0)=1')
ans =
-2/5*sin(t)-1/5*cos(t)+9/20*exp(2*t)+1/2*t-1/4
```

Bestela ere, sistema diferentzialei begira, eskema oso antzeko da, baina orain bai ekuazioak eta bai baldintzak ere zehaztu behar dira. Osterantzean, ikusi segidako etsenplua,

$$\begin{cases} y' - 11y - 16z = 1 + x \\ z' + 2y + z = 1 - x \end{cases} \quad y(0) = 2, \quad z(0) = 1$$

non kode honek soluzioa aurkitzen duen:

```
>> sol=dsolve('Dy-11*y-16*z=1+x,Dz+2*y+z=1-x', 'y(0)=2,z(0)=1', 'x')
sol = % sol egitura bat da
y: [1x1 sym]
```

```

z: [1x1 sym]
>> sol.y % y soluzioa
ans =
432/49*exp(7*x)-22/3*exp(3*x)+76/147-5/7*x
>> sol.z % z soluzioa
ans =
-108/49*exp(7*x)+11/3*exp(3*x)-68/147+3/7*x

```

Adibidean 'x' aldagai independentea zehaztu da, zeren bestela aldagai independentea t hartzen baita.

- Hurrengo eragiketara pasatuz, limiteen kalkulua `limit` aginduarekin egiten dira, eta horrela,

`limit(funtzioa, x, a, noranzkoa)`

instrukzioarekin $\lim_{x \rightarrow a} \text{funtzioa}$ ebazten da. Hori bai, `noranzkoa` finkatzeko, 'left' edo 'right' aukerak, ezker-limitea edo eskuin-limitea erabiltzen dira, hurrenez hurren.

```

>> syms x a;
>> v = [(1 + a/x)^x, exp(-x)];
>> limit(v,x,inf,'left') % ezker-limitea
ans =
[ exp(a),      0]

```

Irteerari begira, `v` bektorea sinbolikoa izan da. Orobat, limite errepikatuak gauzatu daitezke, eta, horretarako `limit` birritan enplegatu behar da, habiaratuak izenekoak hain justu:

```

>> limit(limit('(x^2-y^2)/(x^2+y^2)', y, 0), x, 0)
ans =
1
>> limit(limit('(x^2-y^2)/(x^2+y^2)', x, 0), y, 0)
ans =
-1

```

hots, ebatzitako limiteak hauek dira:

$$\lim_{x \rightarrow 0} \lim_{y \rightarrow 0} \frac{x^2 - y^2}{x^2 + y^2} = 1 \quad \text{eta} \quad \lim_{y \rightarrow 0} \lim_{x \rightarrow 0} \frac{x^2 - y^2}{x^2 + y^2} = -1$$

- Taylor-en garapenak, edozein ordenatakoak izanda ere, `taylor` komandoarekin e-razki egiten dira.

taylor(funtzioa,n,a)

Hala, a puntuko n-1 ordenako funtzioaren Taylor-en garapena kalkulatzen da, hau da:

$$\sum_{m=0}^{n-1} (x-a)^m \frac{f^{(m)}(a)}{m!}$$

```
>> taylor(log(x),4,1)
ans =
x-1-1/2*(x-1)^2+1/3*(x-1)^3
```

Puntua zehazten ez bada, zeroren inguruko garapena irteten da, Mac-Laurin-en garapena izenekoa, hain zuzen. Beste alde batetik, Matlab-en badago interfaze grafiko bat, Taylor-en garapenak konfiguratzeko eta irudikatzeko, eta `taylor` tool tekleatuz gero agertzen da. Irakurleei hori ikertzea eta saiakerak egitea proposatzen diegu.

- Serien batura `symsum` aginduaren bitartez lortzen da, eta sintaxia hau da:

`symsum(gai orokorra,n,a,b)`

non $\sum_{n=a}^b$ *gaia* batzen den. Kasu baterako, $\sum_{n=1}^{\infty} \frac{1}{n^2}$ eta $\sum_{n=1}^{\infty} x^n$ ebatzearren:

```
>> syms n
>> symsum(1/n^2,n,1,inf)
ans =
1/6*pi^2
>> syms x n
>> symsum(x^n,n,1,inf) % berretura-serie konbergentea.
ans =
-x/(x-1)
```

Etsenpluan erakutsi denez, segida baten gai orokorra eskalarra edo funtzio motakoa izan daiteke. Lehenengo kasuan zenbakizko emaitza lortzen da, eta bigarrena funtzio-seriea dela esaten da; emaitza, batura konbergentea denez, funtzio bat da.

- Egia esan, kalkulu sinbolikoaren arloan, hainbat eta hain funtzio existitzen dira. Berbarako, transformazio integralak egiteko, honako komando nagusi hauek eskura daude,

| | |
|-----------------------|---|
| <code>fourier</code> | Fourier-en transformatua |
| <code>laplace</code> | Laplace-ren transformatua |
| <code>ztrans</code> | Z transformatua |
| <code>ifourier</code> | Fourier-en alderantzizko transformatua |
| <code>ilaplace</code> | Laplace-ren alderantzizko transformatua |
| <code>iztrans</code> | Z transformatuaren alderantzizkoa |

guztiak ere oso transformazio ezagunak dira. Irakurleen esku uzten da azterketa sakonagoa egitea; horretarako, Matlab-eko laguntza erabil daiteke.

- Aljebra arloari begira, askotariko funtzioak aplikatzeko daude. Kasu baterako, A matrize batetik abiaturik,

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

kalkulu sinbolikoari esker, zenbait eragiketa aljebraiko egin daitezke. Halan eta guztiz ere, lehenik eta behin, A matrizea sinboliko bihurtu behar da,

```
>> A
A =

     0     1    -1
     1     1     0
    -1     0     1
>> A=sym(A)
A =
[ 0, 1, -1]
[ 1, 1, 0]
[-1, 0, 1]
```

eta bere determinantea, heina eta alderantzizko matrizea, det, rank eta inv aginduekin kalkula daitezke, hurrenez hurren:

```
>> det(A)
ans =
-2
>> rank(A)
ans =
3
>> B=inv(A)
B =
[-1/2, 1/2, -1/2]
[ 1/2, 1/2, 1/2]
[-1/2, 1/2, 1/2]
>> B*A
ans =
[ 1, 0, 0]
[ 0, 1, 0]
[ 0, 0, 1]
```

Orobat, A matrizearen bektore eta balio propioak eig-ren bidez eskuratu daitezke. Agindu horri dagokion sintaxia hau da,

$$[P,D]=\text{eig}(A)$$

non P iragate-matrizea den, eta D matrize diagonalak A -ren antzekoa den. Balio propioak D matrize diagonaleko elementuak dira, eta balio propioei elkartutako bektore propioak P matrizeko zutabeak dira,

```
>> [P,D]=eig(A)
P =
[ -1,  2,  0]
[ -1, -1,  1]
[  1,  1,  1]
D =
[  2,  0,  0]
[  0, -1,  0]
[  0,  0,  1]
```

non antzekotasuna dela eta, $P^{-1}AP = D$ berdintza betetzen den.

```
>> B=inv(P)*A*P % formularen egiaztapena
B =
[  2,  0,  0]
[  0, -1,  0]
[  0,  0,  1]
>> isequal(D,B)
ans =
     1 % egiazkoa
```

Gainera, `poly` aginduarekin matrize baten polinomio karakteristikoa esplizita daiteke,

```
>> C=poly(A)
C =
x^3-2*x^2-x+2
>> subs(C,x,2)
ans =
     0
>> subs(C,x,-1)
ans =
     0
>> subs(C,x,1)
ans =
     0
```

eta, kasu honetan, `subs` komandoarekin egiaztatu berri da haren erroak $\{2, -1, 1\}$ direla.

Azkenik, matrizen baten Jordan-en forma kanonikoa jordan agindua erabiliz aise lor daiteke.

```
>> A=[1 2 3 0 0;0 1 2 0 0; 0 0 1 2 0;0 0 0 2 1;0 0 0 0 1];
>> [P,J]=jordan(A)
P =
    8.0000    -8.0000    -8.0000    -8.0000    -8.0000
    2.2857         0    -4.0000    -1.0000    -3.2500
    1.1429         0         0    -2.0000    -0.5000
    0.5714         0         0         0    -1.0000
         0         0         0         0         1.0000

J =
    2     0     0     0     0
    0     1     1     0     0
    0     0     1     1     0
    0     0     0     1     1
    0     0     0     0     1

>> inv(P)*A*P
ans =

    2.0000         0         0         0         0
   -0.0000     1.0000     1.0000         0     0.0000
         0         0     1.0000     1.0000         0
         0         0         0     1.0000     1.0000
         0         0         0         0     1.0000
```

non J matrizea goi-triangeluarra den, eta berriro ere antzekotasunez $P^{-1}AP = J$ betetzen den.

8.2.1 Adierazpen sinbolikoen irudikapen grafikoak.

Problema batzuetan, adierazpen sinbolikoak irudikatu behar dira. Izan ere, Matlab-en bai planoan, bai espazioan egin daitezke, eta, horretarako, ez* funtzioak erabiltzen dira, gehienbat. Aurrena, planoan sinbolikoki irudikatzeko, ezplot funtzioa dago. Funtzionamenduaren aldetik, S adierazpen sinbolikoa baldin bada, ald aldagaia duena, Matlab-ek S(ald) funtzio sinbolikoa joko du, eta ezplot aginduak ald/S(ald) grafikoa sortuko du. Gainera, adierazpen sinbolikoan ald1 eta ald2 aldagai sinbolikoak agertzen baldin badira, ezplot komandoarekin S(ald1,ald2)=0 ekuazio inplizitua irudikatuko da. Sintaxiari dagokionez, hauek dira aukerak:

```
ezplot(S) edo
ezplot(S,[xmin,xmax]) % aldagai independentearen eremua edo
ezplot(S,[xmin,xmax,ymin,ymax]) % aldagi independentearen eta
                               dependentearen eremua
```

Honako xehetasun hauek kontuan hartzekoak dira:

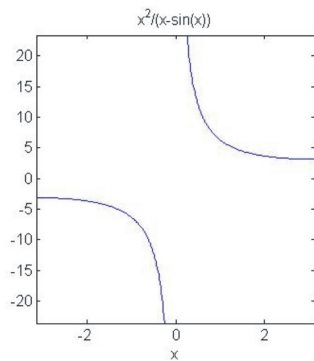
- S adierazpen sinbolikoa irudikatzen da. Zuzenenean sar daiteke, edo existitzen den adierazpen sinboliko baten izena izan daiteke.
- Badago aukera, halaber, zuzenean kate gisako adierazpena ezplot aginduaren barruan sartzeko.
- S-n aldagai sinboliko bat besterik ez badago, aldagaiaren balioak ardatz horizontalean ipintzen dira, eta S(ald)-enak ardatz bertikalean.
- S adierazpenean bi aldagai sinboliko izanez gero, S(ald1,ald2)=0 irudikatuko da, non, ordena alfabetikoari begira, esaterako x, y baldin badira, x abzisa eta y ordenatua izango diren.
- Soil-soilik ezplot(S) idatziz gero, aldagaiaren tartea $(-2\pi, 2\pi)$ izango da. Bi aldagai existituz gero, $ald1 \in (-2\pi, 2\pi)$ eta $ald2 \in (-2\pi, 2\pi)$ beteko da.
- Aipatu behar da, ezplot komandoa era parametrikotan esplizitaturiko funtzio bat irudikatzeko enplega daitekeela. Adibidez, $x = x(t)$ eta $y = y(t)$ baldin bada, eta y aldagaia x-en kontra grafikatu beharrez:

`ezplot(S1,S2,[tmin,tmax])`

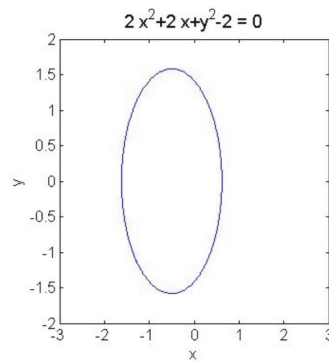
non S1 eta S2 adierazpen sinbolikoek t aldagai sinboliko bera daukaten. Gauzak horrela, S1 OX ardatzean eta S2 OY ardatzean marrazten dira. Berebat, t aldagai sinbolikoaren tartea zehaztu ezik, lehenespenez Matlab-ek $(0, 2\pi)$ tartea usatuko du.

Bestalde, nabarmentzekoa da sinbolikoki irudikatzen den bakoitzean funtzioaren adierazpen matematikoa grafikoaren goiko partean idatzita erakusten dela.

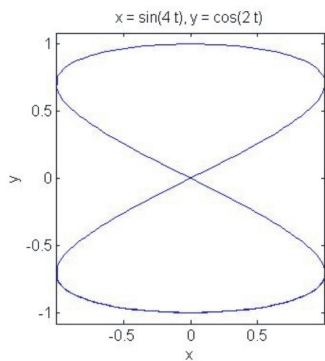
Hiru dimentsioko grafikoak ere lor daitezke. Horretarako, seigarren kapituluan ikusitako funtzioetan aldaketa txiki bat egin behar da, hau da, komandoen aurretik ez erantsi. Horrela, esaterako, ezcontour, ezplot3, ezmesh, ezsurf, eta abar agertzen dira. Komando horien gaineko informazio sakonagoa izateko, irakurleari Matlab-eko laguntza aztertzeke gondita egiten diogu. Segidan, bi dimentsiokoak eta hiru dimentsiokoak 8.1 Irudian batzen dira, eta adierazpen sinbolikoari buruzko zenbait etsenplu erakusten dira.



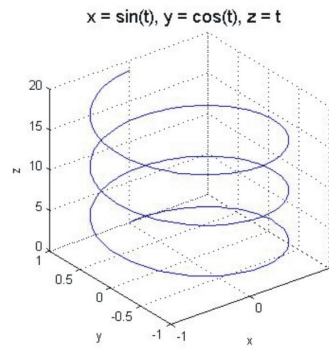
```
syms x;
S=x^2/(x-sin(x));
```



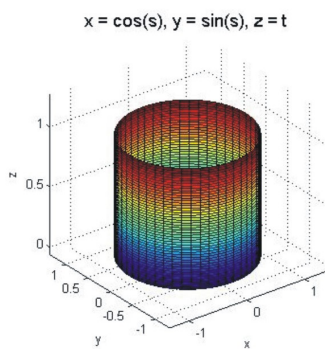
```
syms x y;
S=2*x^2+2*x+y^2-2;
ezplot(S,[-3,3,-2,2])
```



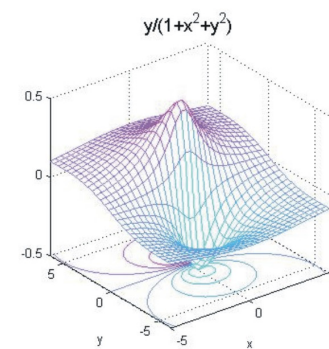
```
syms t;
x=sin(4*t);
y=cos(2*t);
ezplot(x,y,[-2,2])
```



```
syms t;
ezplot3(sin(t),cos(t),t,[0,6*pi])
```



```
syms s t;
ezsurf(cos(s),sin(s),t,[0,2*pi,0,1])
```



```
syms x y;
ezmeshc(y/(1+x^2+y^2),[-5,5,-2*pi,2*pi],30)
colormap(cool)
```

8.1 Irudia: Adierazpen sinbolikoen askotariko 2D eta 3D irudikapen grafikoak.

8.2.2 Maple-ko funtzioen erabilera

Matlab-eko maple funtzioari esker, zuzen-zuzenean Maple-ko funtzioak usa daitezke. Funtzio horien sarrerak sinbolikoak, edo karaktere-kateak edo zenbakiak izan daitezke. Adibide gisa, Maple-ko gcd funtzioaren erabilera erakutsiko da. Funtzio horrek zatitzaile komunetako handiena, bai bi zenbakirena bai bi polinomioarena zehazten du. Jo dezagun, 124 eta 64 zenbakien zenbakitzaile komunetako handiena kalkulatu nahi dela; horretarako, honako kode hau sartu behar da,

```
>> maple('gcd(124,64)')
ans =
4
```

edo $x^2 - y^2$ eta $x^3 - y^3$ polinomioena:

```
>> maple('gcd(x^2-y^2,x^3-y^3)')
ans =
x-y
```

Komando honi buruzko informazioa zabaltzeko, tekleatu doc gcd. Dena den, etengabe maple idatzi beharrean, badago aukera bat gcd funtzioa atzitzeko, eta era sinplea M fitxategia osatzea da:

```
function g=gcd(a,b)
g=maple('gcd',a,b)
```

Gero, exekutatzeko, argumentu desberdinez aldatuz, *Aginduetarako Leihotik* instrukzio hauek sartzen dira:

```
>> z=gcd(x^2-y^2,x^3-y^3) % osagaiak polinomioak
z =
x-y
>> z=gcd(124,64) % osagaiak zenbakiak
z =
4
```

Dena den, eraikitako funtzioa modifikatu egin daiteke matrizeak argumentutzat har ditzan, kode honek azaltzen duenez,

```
function g=gcd(a,b)

if any(size(a)~=size(b))
    error('Sarrerak tamaina berekoak izan behar') % errore-mezu bat
end

for k=1:prod(size(a))
    g(k)=maple('gcd',a(k),b(k));
end
```

```
g=reshape(g,size(a));
```

```
return
```

eta honako irteera hauek lortzen dira:

```
>> A=sym([2 4 6; 8 4 3; 1 3 9]);
>> B=sym([6 2 12; 2 2 5; 1 6 27]);
>> gcd(A,B) % elementuz elementu zatitzaile komunetako handiena
ans =
[2 2 6]
[2 2 1]
[1 3 9]
```

Bestalde, ondorengo etsenpluan, suposatu matrize sinboliko baten sinua kalkulatu nahi dela. Horretarako, lehendabizi `sinM` izeneko *script*-a definitzen da, Maple-ko `sin` funtzioa erabiliz,

```
function y=sinM(x)

for k=1:prod(size(x))
    y(k)=maple('sin',x(k));
end
y=reshape(y,size(x));
return
```

eta jarraian exekuzioan helburua lortzen da:

```
>> C=sym([pi pi/2; pi/4 0]);
>> sinM(C) % C matrizeko elementuen sinuak
ans =
[      0,      1]
[ 1/2*2^(1/2), 0]

>> double(ans) % zenbakizko matrize bihurturik
ans =
      0      1.0000
0.7071      0
```

Bestela, `maple` funtzioak zenbaitetan bi irteera itzultzen ditu, `result` eta `status`. Baldin eta `maple`-ren exekuzioa arrakastatsua izan bada, emaitza `result` argumentuan dago, eta `status`-en argumentua zero da. Exekuzioan, berriz, akatsen bat edo beste agertuz gero, `status`-en balioan zenbaki oso positibo ez-nulu bat eta errore-mezu bat erakutsiko dira.

Ondoko adibidean, polinomio baten diskriminatzailea kalkulatzeko, `discrim` funtzioa enplegatzen da, eta haren sintaxia `discrim(p,x)` da, non `p` `x`-en menpeko polinomioa den.

```
>> syms a b c x
>> [result, status]=maple('discrim',a*x^2+b*x+c)
```

```
result =  
Error, (in discrim) invalid arguments  
status =  
    2
```

Ikusten denez, errore bat detektatzen da, kasu honetan beste argumentu bat behar baita, x aldagai independentea, hain justu. Konponketa egin ostean, orain bai soluzio zuzena erraz agertzen da:

```
>> syms a b c x  
>> [result, status]=maple('discrim',a*x^2+b*x+c,x)  
result =  
-4*a*c+b^2  
status =  
    0    % ez dago akatsik  
>> pretty(result)  
  
                2  
-4 a c + b
```

Bibliografía

- [1] BACKSTROM, G.: *Practical Mathematics using Matlab*. Studentlitteratur. 2000.
- [2] BAILLO MORENO, A., GRANE CHAVEZ, A.: *100 problemas resueltos de Estadística Multivariante (Implementados con Matlab)*. Delta Publicaciones. 2006.
- [3] CARRERA AMURIZA, A. R., MARTINEZ NEBREDA, M.: *Introducción a Matlab y a la creación de interfaces gráficas*: Servicio Editorial UPV-EHU. 2004.
- [4] CHAPMAN, S. J.: *Matlab Programming for Engineers*. Brooks/Cole Publishing Company. 2002.
- [5] COOMBES, K. R.: *Differential equation with Matlab*. John Wiley and sons. 1999.
- [6] DOMINGUEZ BAGUENA, V., RAPUN BANZO, M. L.: *Matlab en cinco lecciones de Numérico*. Universidad de Navarra. 2007.
- [7] ETTER, D. M.: *Solución de Problemas de Ingeniería con Matlab*. Prentice Hall. 1997.
- [8] ETTER, D. M., KUNCICKY, D. C., HULL, D. W.: *Introduction to Matlab 6*. Prentice Hall. 2004.
- [9] GILAT, A.: *Matlab. Una introducción con ejemplos prácticos*. Reverté. 2006.
- [10] GOLUBITSKY, M., DELLNITZ, M.: *Álgebra lineal y ecuaciones diferenciales con Matlab*. Thompson Learning. 1999.
- [11] HIGHAM, D. J., HIGHAM, N. J.: *Matlab Guide*. SIAM. 2000.
- [12] HUNT, B. R., LIPSMAN, R. L., ROSENBERG, J. M.: *A guide to Matlab: for Beginners and Experienced users*. Cambridge University Press. 2001.
- [13] INFANTE, J. A., REY, J. M.: *Métodos numéricos. Teoría, problemas y prácticas con Matlab*. Pirámide. 2006.
- [14] KINCAID, D., CHENEY, W.: *Numerical Analysis*. Brooks/Cole Publishing Company. 1996.
- [15] KNIGHT, A.: *Basics of Matlab and Beyond*. CRC. Press Inc. 2000.
- [16] KOLMAN, B., HILL, D. R.: *Álgebra lineal con aplicaciones y Matlab*. Pearson Prentice Hall. 2005.

- [17] LINDFIELD, G., PENNY, J.: *Numerical Methods using Matlab*. Prentice Hall. 2000.
- [18] MAGRAB, E. B., AZARM, S., BALACHANDRAN, B., DUCAN, J., HEROLD, K., WALSH, G: *An Engineer's Guide to Matlab*. Prentice Hall. 2000.
- [19] MALEK-MADINI, R.: *Advanced Engineering Mathematics with Mathematica and Matlab*. 1. Bolumena. Addison Wesley Longman, Inc. 1998.
- [20] MARCHAN, P., HOLLAND, O.T.: *Graphics and GUIs with Matlab*. Chapman & Hall/CRC. 2003.
- [21] MATHEWS, J. H., FINK, K. D.: *Métodos numéricos con Matlab*. Pearson Prentice Hall. 1999.
- [22] *MATLAB, The Language of Technical Computing, Using Matlab Graphics, Version 6*. The Mathworks Inc. 2002.
- [23] *MATLAB, The Language of Technical Computing, Using Matlab Version 6*. The Mathworks Inc. 2002.
- [24] MOLER, C.: *Numerical computing with Matlab*. SIAM. 2004.
- [25] NAKAMURA, S.: *Análisis numérico y visualización gráfica*. Pearson Prentice Hall. 1997.
- [26] PALM, W. J.: *Introduction to Matlab 7 for engineers*. McGraw-Hill. 2004.
- [27] PEITGEN, H. O., SAUPE, D., JURGENS, H.: *Chaos and Fractals: New frontiers of science*. Springer-Verlag. 1992.
- [28] PEREZ LOPEZ, C.: *Matlab y sus aplicaciones en las ciencias y la ingeniería*. Pearson Prentice Hall. 2002.
- [29] POLKING, J. C., ARNOLD, D.: *Ordinary Differential Equations using Matlab*. Prentice Hall. 2004.
- [30] PORTILLO DE LA FUENTE, A. M., DE UÑA MARTIN, A.: *Prácticas de cálculo numérico con Matlab para ingeniería técnica: ejercicios y aplicaciones*. Universidad de Valladolid. 2005.
- [31] QUARTERONI, A., SALERY, F.: *Scientific computing with Matlab*. Springer. 2003.
- [32] QUINTELA ESTEVEZ, P.: *Introducción a Matlab y sus aplicaciones*. Servicio de Publicaciones de la Universidad de Santiago de Compostela. 1997.
- [33] QUINTELA ESTEVEZ, P.: *Matemáticas en ingeniería con Matlab*. Servicio de Publicaciones de la Universidad de Santiago de Compostela. 2000.
- [34] SIGMON, K., DAVIS, T. A.: *Matlab Primer*. Chapman & Hall/CRC. 2002.
- [35] SUAREZ RODRIGUEZ, M. C., VIEITES RODRIGUEZ, A. M.: *Cálculo integral y aplicaciones con Matlab*. Universidad de Vigo. 2004.