2. gaia

MATLABi buruzko oinarrizko nozioak

Kapitulu honetan ikusiko dugu MATLAB paketea programa matematikoen multzo bat dela eta matrizeen erabileran oinarritzen dela. MATLAB izenak MATrix LABoratory esan nahi du, zeren bere oinarrizko datu-gaia matrize bat baita. MATLAB kalkulu teknikorako lengoaia ahaltsua da, eta oso erabilia da unibertsitateetan, Matematika, Zientzia eta Ingeniaritza ikasketetan. Pakete horrek zenbakizko programen eta marrazketa bidimentsionala eta tridimentsionala egiteko programa grafikoen bilduma zabal bat dauka. Gainera, goimailako lengoaia erabiliz, programa gehigarriak idaztea onartzen du.

MATLAB paketearen instrukzioak `idazteko makinaren` letraz idatziko dira. Adibideetan ikusiko dugu MATLABeko lan-leiho batek (ingelesez: *command window*) erakusten diguna. Honelako >> ikur baten jarraian agertzen da sartutako datua edo instrukzioa. Nahi duguna idatzi eta gero, "sartu" tekla sakatu behar dugu; orduan, ordenagailuak eragiketa egingo du eta emaitza erakutsiko du honela: `ans =`. Programarekin erabiltzeko gidak eta erreferentziako gidak datoz; eta laguntzako leihoan ere aurki daitezke instrukzioei buruzko eta haien aukerei buruzko informazio gehiago eta adibideak.

2.1. Eragiketa aritmetikoak

+	Batuketa
−	Kenketa
*	Biderketa
/	Eskuineko zatiketa
\	Ezkerreko zatiketa
^	Berreketa
pi, e, i	Konstanteak

2.1. adibidea.

```
>>(2+3*pi)/2
ans =
      5.7124
```

2.2. Programari gehitutako funtzioak

Jarraian, MATLAB paketeko funtzio erabilgarri batzuen zerrenda labur bat eskaintzen da:

`abs(#)`	`cos(#)`	`exp(#)`	`log(#)`	`log10(#)`	`cosh(#)`	`sum(#)`	`length(#)`
`sin(#)`	`tan(#)`	`sqrt(#)`	`acos(#)`	`tanh(#)`	`floor(#)`	`ceil(#)`	`round(#)`

Programak berak funtzio erabilgarriei buruzko informazioa eskaintzen du. Funtzio horien zerrenda izateko, idatzi "`help elfun`". Honako adibide honek argitzen du nola erabiltzen eta nola konbinatzen diren eragiketa aritmetikoak eta funtzioak.

2.2. adibidea.

```
>>3*cos(sqrt(4.7))
ans=
     -1.6868
```

Normalean, emaitzan bost zifra hamartar esangarri erakusten dira; `format long` instrukzioari esker, hamabost zifra hamartar esangarri arte lor ditzakegu.

2.3. adibidea.

```
>> format long
>> 3*cos(sqrt(4.7))
ans=
     -1.68686892236893
```

Jarraian, formatu-instrukzioekin agertzen da taula bat. Haiek `format mota` sintaxiarekin idazten dira.

Mota	Emaitza	Adibidea
`short`	Koma finkoa 5 digiturekin	3.1416
`long`	Koma finkoa 15 digiturekin	3.14159265358979
`short e`	Koma mugikorraren formatuan 5 digiturekin	3.1416e+000
`long e`	Koma mugikorraren formatuan 15 digiturekin	3.14159265358979e+000

2.3. Esleipen-instrukzioak

Berdintza ikurraren bidez, adierazpen baten kalkuluaren emaitzari izen bat eslei diezaiokegu.

2.4. adibidea.

```
>>a=3-floor(exp(2.9))
a=
     -15
```

Adierazpen baten bukaeran puntu eta koma idazten dugunean, konputagailuak dagozkion eragiketak egiten ditu eta bere emaitza gordetzen du guk esleitutako izenpean. Hurrengo adibidean, ez da erakusten `b`-ren balioa.

2.5. adibidea.

```
>>b=sin(a);
>>2*b^2
ans=
     0.8457
```

2.4. Matrizeak

MATLAB paketean, aldagai guztiak matrizeak dira. Matrizeak zuzen sartzen dira.

2.6. adibidea.

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

";" ikurrak matrizeko lerroak bereizten ditu, eta lerro bateko gaiak hutsune-espazio baten bidez (edo koma batez) bereizi behar ditugu. Matrizeak lerroz lerro ere sar ditzakegu.

2.7. adibidea.

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

Gehitutako funtzio batzuk erabiliz, matrize berezi batzuk sor ditzakegu.

2.8. adibidea.

```
>>Z=zeros(3,5);              (zerozko 3 x 5 dimentsioko matrize bat sortzen du)
>>X=ones(3,5);               (batezko 3 x 5 dimentsioko matrize bat sortzen du)
>>Y=0:0.5:2                  (1 x 5 dimentsioko matrize hau sortzen du:)
Y=
0 0.5000 1.0000 1.5000 2.0000

>>sin(Y)                     (Y matrizeko gai bakoitzaren sinua hartuz, 1 x 5
                             dimentsioko matrize hau sortzen du:)
ans=
     1.0000 0.8776 0.5403 0.0707 -0.4161
```

`linspace(a,b,n)` instrukzioak a eta b puntuen artean tarte berak bereizitako n puntu sortzen ditu, muturrak sartuta. Aurreko adibideko `Y` bektore berdina ematen du, hau idazten badugu:

```
» linspace(0,2,5)
```

Ez badugu jartzen `n`, instrukzioak 100 puntu sortuko ditu automatikoki.

`logspace(a,b,n)` instrukzioak `linspace(10^a,10^b,n)` instrukzioaren berdina ematen du.

Ez badugu jartzen `n`, instrukzioak 50 puntu sortuko ditu automatikoki.

Matrize bateko gaiekin hainbat eratara joka dezakegu.

2.9. adibidea.

```
>>A(2,3)                    (A-ren (2,3) lekuko gaia aukeratzen du)
ans=
     6
>>A(1:2,2:3)                (A-ren azpimatrize bat aukeratzen du)
ans=
     2 3
     5 6
>>A([1 3],[1 3])            (beste era bat A-ren azpimatrize bat aukeratzeko)
ans=
     1 3
     7 9
>>A(2,2)=tan(7.8);          (beste balio bat esleitzen dio A-ren (2,2) lekuan
                            dagoen gaiari)
>>A(2,:)
ans=
     4 5 6
>>A(:,3)
ans=
    3
    6
    9
```

Laguntzaren leihoak beste funtzio matrizialei buruzko informazioa eskaintzen digu.

2.5. Eragiketa matrizialak

+	Batuketa
−	Kenketa
/	Eskuineko zatiketa
\	Ezkerreko zatiketa
∗	Biderketa
^	Berreketa
'	Irauli konjokatua

2.10. adibidea.

```
>>B=[1 2;3 4];
>>C=B'                          (C B-ren iraulia da)
C=
    1 3
    2 4
>>3*(B*C)^3
ans=
     13080 29568
     29568 66840
```

Azken hori $3(BC)^3$ da.

2.11. adibidea.

```
>>a=[1 2 3];
>>b=[2;4;6];
>>a*b
ans=
     28
>>b*a
ans=
    2   4   6
    4   8  12
    6  12  18
>>a*A
ans=
```

```
    30 36 42
>>c=[4 5 6]';
>>A*c
ans=
    32
    77
   122
>>A*a
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>>A/pi
ans=
   0.3183  0.6366  0.9549
   1.2732  1.5915  1.9099
   2.2282  2.5465  2.8648
>>2\[1 2;3 4]
ans=
    0.5  1
    1.5  2
```

`A\B` idazten badugu, $A^{-1}B$ emango digu, eta `B/A` idatziz gero, BA^{-1}. Ondorioz, `4\1`-ek eta `1/4`-ek emaitza bera dute: 0.25.

Zer emango du MATLABek `2/[1 2;3 4]` idazten badugu?

2.6. Gaiez gai egiten diren eragiketak

MATLAB paketearen ezaugarri erabilgarrienetariko bat da matrize batez gaiez gai eragiten duten funtzio kopuru handi bat daukala. Aurreko adibide batean ikusi dugu 1×5 matrize baten gai bakoitzeko sinua kalkulatzen duela. Batuketa, kenketa eta eskalar bateko biderketa eragiketa matrizialak gaiez gai egiten dira; aldiz, hori ez da gertatzen biderketa, zatiketa eta berreketa eragiketa matrizialekin. Azken hiru eragiketak gaiez gai egin ditzakegu eragiketaren aurrean puntu bat idazten badugu, hau da: $.*$, $./$, $.\backslash$ eta $.\hat{}$. Oso garrantzitsua da jakitea nola eta noiz erabili behar ditugun eragiketa horiek, zeren gaiez gaiko eragiketak oso garrantzitsuak baitira zenbakizko programak eta grafiko programak MATLAB paketearekin eraginkorki diseinatzeko eta inplementatzeko orduan.

2.12. adibidea.

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

```
>>A^2              (AA biderketa kalkulatzen du)
ans=
     7 10
     15 22
>>A.^2                          (A-ren gai bakoitza karratura jasotzen du)
ans=
     1 4
     9 16
>>2.\A
ans=
    0.5000 1.0000
    1.5000 2.0000
>>cos(A./2)                     (A-ren gai bakoitza 2rekin zatitu eta gero,
                                kosinua kalkulatzen du)
ans=
     0.8776  0.5403
     0.0707 -0.4161
```

2.6.1. MATLABek emandako funtzioak

Ikus 2.2. atalean MATLABeko oinarrizko funtzioak. Funtzio horietako propietate garrantzitsu bat da haietako gehienek bektoreen eta matrizeen gainean eragiten dutela, aurreko puntua jarri gabe.

2.13. adibidea.

```
>>log(A)
ans=
    0.0000   0.6931
    1.0986   1.3863
>>B=sqrt(A)
B=
    1.0000   1.4142
    1.7321   2.0000
>>round(B)   (B-ren gaiak zenbaki oso hurbilenera biribiltzen ditu)
ans=
    1   1
    2   2
>>ceil(B)  (B-ren gaiak goiko zenbaki oso hurbilenera biribiltzen ditu)
```

```
ans =
    1   2
    2   2
>>floor(B) (B-ren gaiak beheko zenbaki oso hurbilenera biribiltzen ditu)
ans=
    1   1
    1   2
>>F=[3 5 4 6 1];
>>sum(F)
ans =
    19
>>min(F),max(F),mean(F),prod(F),sort(F)
ans =
    1
ans =
    6
ans =
    3.8000
ans =
    360
ans =
    1   3   4   5   6
```

2.7. Grafikoak

MATLAB paketeak kurben eta gainazalen marrazketa bidimentsionalak eta tridimentsionalak egin ditzake. Laguntza-paketean kontsulta daitezke instrukzio-grafikoen aukera eta alderdi gehigarriak.

`plot` instrukzioaz kurba lauen grafikoak sor daitezke. Adibide honetan, $[0, \pi]$ tarteko $y = \cos(x)$ eta $y = \cos^2(x)$ funtzioen grafikoak lor ditzakegu.

2.14. adibidea.

```
>>x=0:0.1:pi;
>>y=cos(x);
>>z=cos(x).^2;
>>plot(x,y,x,z,'o')
```

Lehenengo lerroan zehazten dira eremua eta 0.1 urratseko tamaina. Hurrengo bi lerroetan funtzioak definitzen dira. Ohartu lehenengo hiru lerroak puntu eta koma batez bukatzen direla; puntu eta koma hori erabiltzen da ez agertzeko pantailan `x`, `y` eta `z` matrize bakoitzeko 32 gaiak. Laugarren lerroak grafikoak ematen duen marrazketa-instrukzioa dauka. Lehenengo bi gaiek, `x` eta `y`, $y = \cos(x)$ funtzioa marrazten dute. Hirugarren eta laugarren gaiek, `x` eta `z`, $z = \cos^2(x)$ funtzioa marrazten dute. Azken gaiak, `'o'`, behartzen du marraztera 'o' puntu hauetan: (x_k, z_k), non $z_k = \cos^2(x_k)$.

Hirugarren lerroan "`.^`" gaiez gaiko eragiketa-adierazlea erabiltzea oinarrizkoa da; izan ere, lehenik gai bakoitzaren kosinua kalkulatzen da eta, gero, `cos(x)` matrizearen gai bakoitza karratura jasotzen da `.^` instrukzioa erabiliz.

`fplot` marrazketa-instrukzioa `plot` instrukziorako aukera erabilgarria da. Instrukzio horren sintaxia `fplot('izena',[a,b],n)` da. Honek `izena.m` funtzioaren grafikoa ematen digu, haren balioa zehaztuz $[a, b]$ tarteko n puntutan. Ez bada ematen n-ren balioa, $n = 25$ da.

2.15. adibidea. *Honek* $[-2, 2]$ *tartean* $y = \tanh$ *marrazten du:*

```
>>fplot('tanh',[-2,2])
```

`plot` eta `plot3` instrukzioak erabiltzen dira kurba parametrizatu bidimentsionalak eta tridimentsionalak marrazteko.

2.16. adibidea. $c(t) = (2\cos(t), 3\sin(t))$ *elipsearen marrazketa,* $0 \leq t \leq 2\pi$*, instrukzio hauekin lortzen da:*

```
>>t=0:0.2:2*pi;
>>plot(2*cos(t),3*sin(t))
```

2.17. adibidea. $c(t) = (2\cos(t), t^2, 1/t)$ *kurbaren marrazketa,* $0.1 \leq t \leq 4\pi$*, instrukzio hauekin lortzen da:*

```
>>t=0.1:0.1:4*pi;
>>plot3(2*cos(t),t.^2,1./t)
```

`meshgrid` instrukzioaz marrazketa tridimentsionalak lortzeko funtzioaren eremu angeluzuzen bat zehaztu behar dugu, eta, gero, grafikoa lortu `mesh` edo `surf` instrukzioekin.

2.18. adibidea.

```
>>x=-pi:0.1:pi;
>>y=x;
>>[x,y]=meshgrid(x,y);
>>z=sin(cos(x+y));
>>mesh(z)
```

`hold on` instrukzioak datu eta propietate guztiekin gordetzen ditu marrazkiak eta, orain, beste instrukzio batzuk gehi daitezke.

`hold off` instrukzioak leiho grafiko hori bukatzen du.

`subplot(m,n,p)` instrukzioak leiho grafikoa zatitzen du m $\times$ n zatitan, eta p. azpileihoa (ezkerretik eskuinera eta goitik behera zenbatuta) aukeratzen du, oraingo grafikoa marrazteko.

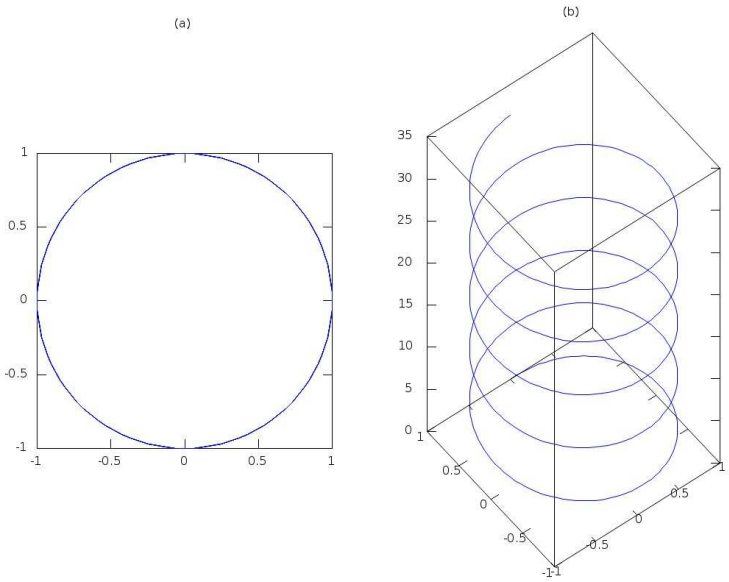
2.19. adibidea.

```
>>t=0:pi/50:10*pi;
>>subplot(1,2,1);
>>plot(sin(t),cos(t));
>>axis square
>>title('(a)')
>>subplot(1,2,2);
>>plot3(sin(t),cos(t),t);
>>title('(b)')
```

Irudia gorde dezakegu, hau idatziz:

```
>>print -dpdf irudiizena.pdf;
```

edo hau:



2.1. irudia. Zirkulu bidimentsional bat eta helize tridimentsional bat, bi zatitako irudi batean.

```
>>print -djpg irudiizena.jpg;
```

Beste aukera batzuk ere badaude; begiratu laguntza-leihoan.

2.8. MATLABez programatzen: M-fitxategiak

2.8.1. Instrukzioen fitxategiak

MATLABen instrukzioen segida bat fitxategi batean gordetzen dugunean, horrelako instrukzioen M-fitxategi bat dugu. Fitxategi horiek erabilgarriak dira, behin baino gehigotan erabiltzen badugu instrukzioen segida hori. Era hauetan egikari daiteke:

- Aginduen leihoan fitxategiaren izena idatziz.
- Fitxategia editatuz eta editorearen leihotik exekutatuz.

2.20. adibidea. *Garatu instrukzioen fitxategi bat, puenting-jauzilariaren abiadura kalkulatzeko.*

Ebazpena:

```
g=9.81; m=68.1; t=12; cd=0.25;
v=sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t)
```

Hau `instrukadib.m` izeneko fitxategi batean gordetzen badugu eta, gero, aginduen leihoan hau idazten badugu:

```
>>instrukadib
```

emaitza hau izango da:

```
v=
   50.6175
```

Parametro baten balioa jakin nahi badugu, adibidez `g`-rena, hau idatziko dugu:

```
>>g
g=
 9.8100
```

2.8.2. Funtzio-fitxategiak

Guk funtzio berriak defini ditzakegu MATLAB paketearekin erabiltzeko. Horretarako, `m` luzapeneko testu-fitxategi bat idatzi behar dugu MATLABek berak duen editorea erabiliz: M-fitxategi bat. Behin definituz gero, beste edozein funtzio bezala erabil daiteke.

2.21. adibidea. *$f(x) = -x^2/4 + x + 1$ funtzioa definituko dugu, eta* `fun.m` *izeneko M-fitxategi batean gordeko dugu. Horretarako, testu-editoreaz hau idatziko dugu:*

Ebazpena:

```
function y=fun(x)
% Funtzio honen bidez f(x)=a*x^2+b*x+c polinomioa kalkula dezakegu.
a=-1/4; b=1; c=1;
y=a*x^2+b*x+c
```

Geroago, funtzio hori baliozta dezakegu aginduen leihoan `x` argumentuaren balio baterako. Hots, $f(2)$ kalkulatzeko, nahikoa da honako hau idaztea:

```
>>fun(2)
ans=
    2.0000
```

Gainera, funtzio honi buruz laguntza eska dezakegu:

```
>>help fun
Funtzio honen bidez f(x)=a*x^2+b*x+c polinomioa kalkula dezakegu.
```

Aldagaietarako letra desberdinak erabil ditzakegu eta funtzioari izen desberdina eman diezaiokegu, baina formatu berdina izan behar dute. Funtzio hori `fun.m` fitxategi batean gorde eta gero, MATLABeko beste edozein funtzio bezala erabil dezakegu.

```
>>cos(fun(3))
ans=
     -0.1782
```

Bestalde, `feval` instrukzioaz baliozta ditzakegu funtzioak era erabilgarri eta eraginkor batean.

```
>>feval('fun',4)
ans=
     1
```

Geroago, funtzioaren izena ahazten badugu, guk honela erabil dezakegu `lookfor` instrukzioa:

```
>>lookfor polinomioa
```

eta honako informazio hau jasoko dugu:

```
 fun.m: % Funtzio honen bidez f(x)=a*x^2+b*x+c polinomioa kalkula dezakegu.
```

Aldiz, hau gertatuko da:

```
>>a
??? Undefined function or variable 'a'.
```

Horren arrazoia zera da: `a` aldagaiak -1/4 balioa hartzen du M-fitxategiaren barnean, hots, aldagai *lokala* da, eta aldagai lokalak ezabatu egiten dira fitxategia exekutatu eta gero. Aldiz, instrukzio M-fitxategi baterako aldagaiak lan-espazioan gordetzen dira, banan-banan instrukzio bakoitza aginduen leihoan eskuz sartuko bazenu bezala; hori gertatzen da, adibidez, `instrukadib.m` M-fitxategiarekin. Ariketa gisa, egiazta ezazu hori.

M-fitxategiek emaitza bat baino gehiago itzul ditzakete. Adibidez, `estatiskoak.m` fitxategi honek bektore bati dagozkion gaien batezbestekoa eta desbideratze estandarra kalkulatzen ditu:

```
function [bbest,dest]=estatistikoak(x)
n=length(x);
bbest=sum(x)/n;
dest=sqrt(sum((x-bbest).^2/n));
```

Orduan, honela joka dezakegu:

```
>>y=[8 5 10 12 6 7.5 4];
>>[b,d]=estatistikoak(y)
b =
  7.5000
d =
  2.6049
```

Instrukzio M-fitxategiak gutxi erabiltzen dira; funtzio M-fitxategiak erabiliko ditugu batez ere eta, ondorioz, M-fitxategiak izen laburtuarekin adieraziko ditugu hemendik aurrera horiek.

2.8.3. Azpifuntzioak

Funtzio batek beste funtzio batzuk barnera ditzake. Horrelako funtzioak M-fitxategi berezitu moduan existitu arren, M-fitxategi bakar batean sartuta egon daitezke.

Har dezagun, berriro, jauzilariaren 2.20. adibidea; kalkula dezagun abiadura ere M-fitxategi honen bitartez:

```
function v=jauzilari(t,m,cd)
```

```
v=abi(t,m,cd);
end

function v=abi(t,m,cd)
g=9.81;
v=sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t)
end
```

`end` funtzio bakoitzaren muga adierazteko erabiliko dugu, baina ez da beharrezkoa. Fitxategi hori `jauzilari.m` izenarekin gordeko dugu. Lehenengo funtzioari *funtzio nagusia* (*main function*) deritzogu. Funtzio hori aginduen leihoan bakarrik eskura dezakegu. Beste funtzioak, kasu honetan `abi`, *azpifuntzioak* (*subfunctions*) dira. Azpifuntzio bat funtzio nagusirako M-fitxategiaren barnean bakarrik da eskuragarria. Aginduen leihotik exekutatuz, hau dugu:

```
>>jauzilari(12,68.1,0.25)
ans=
   50.6175
```

Hala ere, `abi` azpifuntzioa exekutatzen saiatzen bagara, honelako errore mezu bat aterako da:

```
>>abi(12,68.1,0.25)
??? Undefined command/function 'abi'.
```

2.8.4. Input-output

Funtzioan aginduen leihoaren bidez soilik ez dira sartzen edo ateratzen parametroen balioak edo beste informazio bat; badaude beste bi bide ere lan hori egiteko.

- `input` funtzioa. Funtzio honen bidez erabiltzaileak balio batzuk zuzenean eman ditzake; adibidez, honela idatziz M-fitxategian:

  ```
  m=input('Masa (kg): ')
  ```

  Lerro hau exekutatzen denean, monitorean hau agertuko da:

  ```
  Masa (kg):
  ```

  Erabiltzaileak balio bat sartuko du, eta orduan hura esleituko zaio `m` aldagaiari.

  Funtzio honen bidez hitzak (*strings*) ere sar ditzakegu; horretarako, `'s'` gehituko diogu funtzioaren argumentuen zerrendari. Adibidez,

```
izena=input('Sartu zure izena: ','s')
```

- `disp` funtzioa. Funtzio honek era praktiko bat ematen du parametro baten balioa edo hitza (*string*) ikusi ahal izateko. Bere sintaxia `disp(parametroa)` da.

2.22. adibidea.

```
function jauzilaria
% jauzilaria: jauzilariaren abiadura kalkulatzeko era interaktibo bat
g=9.81;
m=input('Masa (kg): ');
cd=input('Erresistentzia-koefizientea (kg/m): ');
t=input('Denbora (s): ');
disp('  ')
disp('Abiadura (m/s):')
disp(sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t))
```

`jauzilaria.m` izeneko fitxategian gordez gero, eta hau idatziz:

```
>> jauzilariia

Masa (kg): 68.1
Erresistentzia-koefizientea (kg/m): 0.25
Denbora (s): 12

Abiadura (m/s):
    50.6175
```

- `fprintf` funtzioa. Funtzio honen bidez kontrolatzen dugu monitorean ateratzen den informazioa. Haren sintaxia, labur emanda, hau da:

  `fprintf('formatua',x,...)` non `formatua`-ren bidez adierazten dugu nola ikusi nahi baitugu `x`-ren balioa. Adibidez,

```
 >>fprintf('Abiadura %8.4f m/s da\n',abiadura)
Abiadura   50.6175 m/s da
```

  Normalean, hauek dira `fprintf` funtzioarekin erabiltzen diren formatuen eta kontrolen kodeak:

Formatuen kodeak	Deskripzioa
%d	Zenbaki osoaren formatuan
%e	Formatu zientifikoa e minuskularekin
%E	Formatu zientifikoa e maiuskularekin
%f	Formatu hamarrenarekin
%g	%e edo %f formatuen trinkoena
Kontrolen kodea	**Deskripzioa**
\n	Hasi lerro berri bat
\t	Tabuladorea

Ikus ditzagun, orain, bi adibide funtzio honen erabilera hobe ulertzeko.

2.23. adibidea.

```
>> fprintf('%5d %10.3f %8.5e\n',100,2*pi,pi);
  100      6.283  3.14159e+000
```

2.24. adibidea.

```
function fprintfdemo
x=[1 2 3 4 5];
y=[20.4 12.6 17.8 88.7 120.4];
z=[x;y];
fprintf('     x        y\n');
fprintf('%5d %10.3f\n',z);
```

Hau da horren emaitza:

```
>> fprintfdemo

    x       y
    1    20.400
    2    12.600
    3    17.800
    4    88.700
    5   120.400
```

2.8.5. Fitxategiak sortu eta fitxategietan sartu

MATLABek datu-fitxategiak irakur eta idatz ditzake. Modu errazenak fitxategi bitar berezi bat erabiltzen du, MAT-fitxategi deritzona; hori espresuki gauzatzen da MATLABen barnean

inplementatzeko. Horrelako fitxategiak `save` instrukzioaz sortzen dira; `load` instrukzioaz fitxategian sar daitezke eta datuak irakurri. `fitxategiizena.mat` datu-fitxategia sortzeko edo irakurtzeko, sintaxi hau erabiliko dugu:

```
save fitxategiizena var1 var2 ...  varn

load fitxategiizena var1 var2 ...  varn
```

Ez baditugu `var1 var2 ... varn` aldagaiak idazten, lan-espazioko aldagai guztiak gordeko dira (`save` kasuan) edo kargatuko dira (`load` kasuan).

Adibidez,

```
>> g=9.81;m=80;t=5;
>> cd=[.25 .267 .1245 .28 .273]';
>> v=sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);
```

Orduan, abiadura- eta erresistentzia-koefizienteak gorde ditzakegu `abikoef.mat` fitxategian, hau idatziz:

```
>> save abikoef v cd
```

`clear` instrukzioa idazten badugu, lan-espazioko aldagai guztiak ezabatuko dira. Aldagai horiek berreskuratzeko nahikoa da hau idaztea:

```
>> load abikoef v cd
```

Lan-espazioan zein aldagai ditugun jakiteko, hau idatziko dugu:

```
>> who

Your variables are:
cd   v
```

Horrelako fitxategiak oso erabilgarriak dira MATLABekin lan egiteko; baina, agian, beste programa batzuekin hobe izango da testu-fitxategi bat izatea, hau da, ASCII fitxategi bat. Horretarako, nahikoa da `save abikoef v cd -ascii` idaztea, eta `abikoef.txt` fitxategi batean gordeko du. Datuak zehaztasun bikoitzarekin gorde nahi baditugu, `-ascii -double` idatziko dugu. Adibidez,

```
>> A=[5 7 9 2;3 6 3 9];
>> save simpmatrix.txt -ascii -double
```

Horrelako fitxategi bat beste programa batzuek irakur dezakete; esate baterako, Excelek eta Word-ek. Bestalde, `simpmatrix.txt` ez denez MAT-fitxategi bat, behin MATLABek irakurriz gero, MATLABek zehaztasun bikoitzeko matrize bat sortuko du, honela:

```
>> load simpmatrix.txt
>> simpmatrix
simpmatrix =
     5    7    9    2
     3    6    3    9
```

Gainera, `load` instrukzioa funtzio gisa erabil dezakegu. Adibidez,

```
>> A = load(simpmatrix.txt)
```

2.8.6. Programazio egituratua

Begiztak eta adarkadurak

Erlazio eragileak:

Eragilea	Erlazioa	Adibidea
$==$	Zerbaiten berdin	`x == 0`
$\sim=$	Zerbaiten desberdin	`unitate ~= 'm'`
$<$	Zerbait baino txikiago	`a < 0`
$>$	Zerbait baino handiago	`s > t`
$<=$	Zerbait baino txikiago edo berdin	`2.8 <= a/3`
$>=$	Zerbait baino handiago edo berdin	`r >= 0`

Eragile logikoak:

$\sim$	Ez	(Egiazkoa, baldin eta soilik baldin, proposizioa faltsua bada)
&	Eta	(Egiazkoa, baldin bi proposizioak egiazkoak badira)
|	Edo	(Egiazkoa, baldin bi proposizioetako bat egiazkoa bada)

Booleko balioak:

1	Egiazkoa
0	Faltsua

`for`, `if`, `switch` eta `while` egiturek MATLAB paketean eragiten dute beste programazio-lengoaia batzuetan egiten duten antzeko era batean. Instrukzio horiek oinarrizko sintaxi hau hartzen dute:

`for` (begiztaren aldagaia=begiztaren heina)

(adierazpenak)

`end`

Baina MATLABen, batzuetan, ez dugu `for` begizta erabiltzeko beharrik. Adibidez,

```
i=0;
for t=0:0.02:50
   i=i+1;
   y(i)=cos(t);
end
```

adieraz daiteke *era bektorizatu* batean, honela:

```
t=0:0.02:50
y=cos(t);
```

MATLABek automatikoki handitzen du bektoreen eta matrizeen luzera. Adibidez,

```
t=0:0.01:5;
for i=1:length(t)
  if t(i)>1
    y(i)=1/t(i);
```

```
  else
    y(i)=1;
  end
end
```

```
if (baldintza)
        (adierazpenak)
end
if (baldintza)
        (adierazpenak)
    else
        (adierazpenak)
end
```

Baldintza bat baino gehiago daudenean, `if...elseif...else...end` egitura erabiltzen dugu:

```
if (baldintza)
        (adierazpenak)
    elseif (baldintza)
        (adierazpenak)
    elseif (baldintza)
        (adierazpenak)
            .
            .
            .
    else
        (adierazpenak)
end
```

"`error`" funtzioa erabiltzeko, `if` oso egokia da. Adibidez, hau dugu:

```
function f=errortest(x)
if x==0
   error('zero balioa aurkitu du');
end
f=1/x
```

Hau da, zatiketa kalkulatuko da, argumentua zero ez denean bakarrik.

```
>> errortest(10)
ans =
    0.1000
```

Bestela, mezu hau emango digu:

```
 >> errortest(0)
??? Error using ==> errortest
zero balioa aurkitu du
```

`switch` egitura `if...elseif...else...end` egituraren antzekoa da. Hala ere, banakako baldintzak jarri beharrean adarkatzea test-adierazpen txikiagoetan oinarritzen da:

```
switch (test-adierazpena)
    case (balioa);
            (adierazpenak)
    case (balioa);
            (adierazpenak)
                .
                .
                .
    otherwise
            (adierazpenak)
```

```
   end
```

Adibidez,

```
kalifikazioa = 'B';
switch kalifikazioa
  case 'A'
    disp('Bikain')
  case 'B'
    disp('Oso ondo')
  case 'C'
    disp('Ondo')
  case 'D'
    disp('Nahiko')
  case 'E'
    disp('Txarto')
  otherwise
    disp('Oso txarto')
end
```

Kode hori exekutatzen badugu, "`Oso ondo`" emango digu.

`while` egituran begizta bat errepikatzen da, baldintza logikoa egia bada, hots:

```
while (baldintza)
                (adierazpenak)
end
```

Adibidez,

```
x=8
while x>0
  x=x-3;
  disp(x)
end
```

Kode hori exekutatzen denean, emaitza hau da:

```
x=
     8
```

```
    5
    2
   -1
```

Hurrengo adibidean erakusten da nola sar ditzakegun begizta batzuk elkarren gainka, matrize bat sortzeko. Testu-lerroak `habia.m` fitxategi batean gordez, orduan, MATLAB paketeko lan-leihoan `habia` idazten dugun bakoitzean, `A` matrizea lortuko dugu. Oharataraziko dugu `A` matrizeko gaiek, goiko ezkerreko izkinatik hasiz, Pascalen triangelua sortzen dutela.

2.25. adibidea.

```
for i=1:5
  A(i,1)=1;
  A(1,i)=1;
end
for i=2:5
  for j=2:5
    A(i,j)=A(i,j-1)+A(i-1,j);
  end
end
A
```

`break` instrukzioa azken (`for` edo `while`) begiztatik ateratzeko, hau bete baino lehen erabiliko dugu.

2.26. adibidea.

```
for k=1:100
  x=sqrt(k);
  if ((k>10)&(x-floor(x)==0))
    break
  end
end
k
```

`disp` instrukzioa erabiliko dugu testu-lerro bat edo matrize bat erakusteko.

2.27. adibidea.

```
n=10;
k=0;
while k<=n
  x=k/3;
  disp([x x^2 x^3]);
  k=k+1;
end
```

`pause` instrukzioa erabiltzen da, programa baten exekuzioa gelditu nahi badugu leku egoki batera heltzen denean. `pause(n)` idazten badugu, **n** segundotan zehar geldi egongo da. `tic` instrukzioak oraingo denbora gordetzen du, eta `toc` instrukzioak monitorean erabilitako denbora emango digu. `beep` instrukzioak, berriz, ordenagailuaren "beep" soinu bat emango digu. Adibidez,

```
tic
beep
pause(5)
beep
toc
```

Kode hau exekutatzen denean, "beep" bat emango digu. Bost segundo geroago beste "beep" bat entzungo dugu, eta monitorean mezu hau izango dugu:

```
Elapsed time is 5.006306 seconds.
```

Programa baten exekuzioa moztu nahi badugu, nahikoa da **Ctrl+C** jotzea.

2.28. adibidea. Egitura habiaratuak. *Adierazpen honen bidez kalkula ditzakegu* $f(x) = ax^2 + bx + c$ *ekuazio koadratikoaren erroak:*

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

Garatu funtzio bat, formula hori inplementatzeko koefizienteen balioak finkatu eta gero.

Ebazpena:

```
function errokoad(a,b,c)
% errokoad: ekuazio koadratikoaren erroak
%     errokoad(a,b,c): ekuazio koadratikoaren erro erreal eta konplexuak
% input:
%   a=2. mailako koefizientea
%   b=1. mailako koefizientea
%   c=0. mailako koefizientea
% output:
%   r1=1. erroaren zati erreala
%   i1=1. erroaren zati irudikaria
%   r2=2. erroaren zati erreala
%   i2=2. erroaren zati irudikaria
if a==0
        % kasu bereziak
        if b~=0
                % erro erreal bakuna
                r1=-c/b
        else
                % soluzio nabaria
                error('Soluzio nabaria. Saiatu berriro')
        end
else
        % formula koadratikoa
        d=b^2-4*a*c;                    % diskriminantea
        if d>=0
                % erro errealak
                r1=(-b+sqrt(d))/(2*a)
                r2=(-b-sqrt(d))/(2*a)
        else
                % erro konplexuak
                r1=-b/(2*a)
                i1=sqrt(abs(d))/(2*a)
                r2=r1
                i1=-i1
        end
end
```

Orain, kalkulu hauek egin ditzakegu:

```
>> errokoad(1,1,1)
r1=
  -0.5000
i1=
```

```
   0.8660
r2=
  -0.5000
i2=
  -0.8660
>> errokoad(1,5,1)
r1=
  -0.2087
r2=
  -4.7913
>> errokoad(0,5,1)
r1=
  -0.2000
>> errokoad(0,0,0)
??? Error using ==> errokoad
Soluzio nabaria. Saiatu berriro
```

2.8.7. M-fitxategiei funtzioak igorri

M-funtzio bat garatu daiteke ekuazio berri bakoitzerako, baina aukera hobea da funtzio generiko bat diseinatzea eta analizatu nahi dugun ekuazioa argumentu gisa pasatzea. MATLABen hizkeran, horrelako funtzioek `funtzio funtzioak` izena dute.

Funtzio anonimoak

Funtzio anonimoak erabiliz funtzio soil bat sor dezakegu, M-fitxategi bat sortu gabe. Agindu en leiho baten barnean defini daitezke haiek, honelako sintaxi baten bidez:

```
feskuleku = @(argumzerrenda) adierazpena
```

non `feskuleku =` zuk erabil dezakezun funtzio-eskulekua baita, `argumzerrenda` funtzioaren argumentuen zerrenda, komekin bereizita, eta `adierazpena` MATLABeko adierazpen bat. Adibidez,

```
>> f1=@(x,y) x^2 + y^2;
>> f1(3,4)
ans =
    25
```

Beste adibide bat:

```
>> a=4;
>> b=2;
>> f2=@(x) a*x^b;
>> f2(3)
ans =
    36
```

`a` eta `b`-rako balio berriak sartzen baditugu, ez da aldatzen funtzio anonimoaren balioa:

```
>> a=3;
>> f2(3)
ans =
    36
```

Baina, guk birsortzen badugu funtzio bera, balioa aldatuko da:

```
>> f2=@(x) a*x^b;
>> f2(3)
ans =
    27
```

Antzeko emaitzak lortzeko beste bide bat `inline` funtzioa erabiltzea da. Hau da:

```
>> f1=inline('x^2 + y^2','x','y');
```

Funtzio funtzioak

Funtzio funtzioak beste funtzio batzuetan eragiten duten funtzioak dira eta funtzioak argumentu gisa pasatzen dira. Adibide argi bat `fplot` funtzioarena da; honek funtzioen grafikoak marrazten ditu. Hau da bere idazkera labur bat:

`fplot`(*fun,lims*)

non *fun* funtzio matematikoa marraztuko baita *lims*=[*xmin,xmax*] mugen artean. Adibidez:

```
>> vel=@(t) ...
sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t)
>> fplot(vel,[0 12])
```

Honek $t = 0$-tik $t = 12$-rako marrazki bat sortuko du.

2.29. adibidea. *Sortu funtzio funtzioaren M-fitxategi bat, tarte bateko funtzio baten balioen batezbestekoa kalkulatzeko. Hau ikusiko dugu $t >= 0$-tik $t = 12$-rako tarteko puenting-jauzilariaren abiadura erabiliz.*

Ebazpena. Hau da hori lortzeko bide bat:

```
>> t=linspace(0,12);
>> v=sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
>> mean(v)
ans =
   36.0870
```

Beste bide bat:

```
function fbb = funcbb(a,b,n)
%
% funcbb: batezbestekoa kalkulatzen du
% input:
%    a = tartearen ezker muturra
%    b = tartearen eskuin muturra
%    n = tartearen puntuen kopurua
% output:
%    fbb = funtzioaren balioen batezbestekoa
x = linspace(a,b,n);
y = func(x);
fbb = mean(y);
end

function f = func(t)
f=sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
end
```

eta gero,

```
>> funcbb (0,12,60)
ans =
   36.0127
```

Emaitza berdina lortzeko beste era bat da abiadura kalkulatzeko funtzioa argumentu moduan sartzea; hau da:

```
function fbb = funcbb(f,a,b,n)
%
% funcbb: batezbestekoa kalkulatzen du
% input:
%    f = balioztatu nahi dugun funtzioa
%    a = tartearen ezker muturra
%    b = tartearen eskuin muturra
%    n = tartearen zatien kopurua
% output:
%    fbb = funtzioaren balioen batezbestekoa
x = linspace(a,b,n);
y = f(x);
fbb = mean(y);
end
```

Gero, hau idatziko dugu:

```
>> abi=@(t) ...
sqrt(9.81*68.1/0.25)*tanh(sqrt(9.81*0.25/68.1)*t);
>> funcbb(abi,0,12,60)
ans =
   36.0127
```

Funtzioen parametroak aldatzea

Ikertzeko orduan oso arrunta da ikustea nola aldatzen den menpeko aldagaiaren balioa funtzioen parametroak aldatzen ditugunean. Parametroei buruzko ikerketa horri ereduaren *sentsibilitatearen ikerketa* deritzogu.

Aurreko 2.29. adibideko `funcbb` funtzioaren sentsibilitate-analisi bat egin nahi badugu, M-file hau idatz dezakegu:

```
function fbb=funcbb(f,a,b,n,varargin)
x = linspace(a,b,n);
y = f(x,varargin(:));
fbb = mean(y);
```

Geroago, funtzio anonimo hau idatziko dugu:

```
>> abi=@(t,m,cd) sqrt(9.81*m/cd)*tanh(sqrt(9.81*cd/m)*t);
```

eta gero, `m` eta `cd` parametroen balio desberdinetarako abiadura kalkula dezakegu. Adibidez, $m = 68.1$ eta $c_d = 0.25$ badira, hau ematen du:

```
>> funcbb(abi,0,12,60,68.1,0.25)
ans =
   36.0127
```

Gero, $m = 100$ eta $c_d = 0.28$ badira, hau ematen du:

```
>> funcbb(abi,0,12,60,100,0.28)
ans =
   38.9345
```