Lab Nr. 1, Probability and Statistics

Introduction to Matlab I

Basic notions

Pentru cei ce vor să-şi instaleze Matlab-ul pe laptop sau acasă, etc., ATENŢIE!, trebuie neapărat să aibă şi Statistics Toolbox, care nu face parte din pachetul de Toolbox-uri obişnuit.

Când deschidem Matlab, se deschide o fereastră în care sunt mai multe subferestre: Current Directory (sau Folder), Command History (în stânga, sus, respectiv, jos) şi Command Window în partea din dreapta (şi Workspace in versiunile mai noi). (Acesta este default-ul, se poate schimba sau reveni, de la Toolbar-ul de sus, de la Desktop $->$ Desktop Layout, sau Layout in versiunile mai noi.) Fereastra Command Window este interactivă, aici, la prompterul specific Matlab-ului, >>, putem face calcule, atribuiri, etc., al căror rezultat este afişat imediat:

```
>> 1+1
```

care produce rezultatul

```
ans =

     2
```

Cum nu i-am dat niciun nume (nicio atribuire) rezultatului, Matlab l-a numit *ans* de la answer (răspuns). Dacă vrem să-i dăm un nume:

```
>> a=1+1

a =

     2
```

sau o simplă atribuire:

```
>> x=1

x =

     1
```

Motivul pentru care apare de două ori $x = 1$ este că Matlab-ul afişeaza pe ecran rezultatul oricărei execuţii (chiar dacă este doar o simplă atribuire), dacă nu se pune simbolul semicolon (punct şi virgulă, **;**) la sfârşit. Dacă nu dorim această afişare, punem **;** la sfârşit:

```
>> x=1;
>>
```

Obs: Chestiunea cu ";" la sfârşit va fi importantă când vom avea volum mare de date, pe care NU dorim să le vedem afişate pe ecran (ci doar rezultatele, ce vrem să facem cu ele).

Acum, vorbind de atribuiri: numele MATLAB vine de la MATrix LABoratory, adică lucrează foarte bine şi uşor cu matrici. Un scalar e, de fapt, interpretat ca o matrice 1×1. Matricile se definesc cu paranteze pătrate [,], elementele aceleiaşi linii fiind despărţite de comma or space (i.e. , sau ␣). **Atenţie** să se potrivească dimensiunile, altfel dă eroare:

```
>> A=[ 1 2 3; 4, 5, 6; 7 8 9; 0 0 0]

A =

     1     2     3
     4     5     6
     7     8     9
     0     0     0

>> a=[1 2 3; 4 5 6; 7 8]
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

De asemenea, de menţionat, când e nevoie de paranteze într-o expresie, se folosesc **doar paranteze rotunde, simple**, cele pătrate fiind rezervate pentru definirea matricilor.

Matricile pot fi, uneori, definite mai simplu, fără a lista fiecare element, ci doar initialvalue : step : endvalue. Simbolul : e pe post de "de la" sau "până la" sau "cu pasul":

```
>> x=1:2:10

x =

     1     3     5     7     9
```

Pasul poate fi şi negativ, iar dacă nu se precizează o valoare, defaul-ul este 1:

```
>> x=5:-0.5:2

x =

    5.0000    4.5000    4.0000    3.5000    3.0000    2.5000    2.0000

>> y=1:7

y =

     1     2     3     4     5     6     7
```

De asemenea, simbolul : poate fi folosit şi pentru a defini o matrice:

```
>> A=[1:3;4:-2:0; 1:3:8; 0:2]
```

```
A =

     1     2     3
     4     2     0
     1     4     7
     0     1     2
```

sau pentru a obţine o submatrice (inclusiv un element) a unei matrici deja definite:

```
>> A(1:3,2:3)

ans =

     2     3
     2     0
     4     7
```

conţine liniile $1, 2, 3$ şi coloanele $2, 3$ ale matricii A,

```
>> A(:,2:3)

ans =

     2     3
     2     0
     4     7
     1     2
```

conţine toate liniile şi coloanele $2, 3$ ale matricii A, iar

```
>> A(4,2)

ans =

     1
```

este elementul de pe linia a $4-$a şi coloana a $2-$a a matricii A. Evident că $A(:,:)$ va fi acelaşi lucru cu matricea A.

M-files, scripts and Matlab functions

Tot ceea ce am făcut până acum în Command Window (interactiv şi distractiv...) se va şterge în momentul când se încheie sesiunea Matlab. Cum lucrăm cu fişiere, ca să putem salva ceea ce am lucrat? Fişierele Matlab se numesc M-files şi sunt de două tipuri: script (fişiere simple) şi funcţii. Din colţul de sus, de la File, select $\text{New} -> \text{M-File}$ (script sau function). Se deschide o nouă fereastră Editor-Untitled (deocamdată nu are nume fişierul!).

Într-un fişier script, listăm toate comenzile pe care vrem să le execute programul la lansarea acestuia (fără o linie de început sau de sfârşit a programului). De exemplu:

```
x=1;
y=2;
z=x+y
t=x*y
```

Observăm că variabilelor x şi y (când le-am atribuit valori) le-am pus ; la sfârşit, pentru ca să nu le mai afişeze valorile pe ecran (puteam lăsa să le afişeze, dacă voiam), iar la definirea variabilelor noi z şi t nu le-am mai pus ; la sfârşit ca să le putem vedea valorile. Salvăm fişierul, de la Save din File, sau dând click pe dischetă. Să şi le salveze în contul lor, cel mai bine, să-şi facă un director pentru Laboratorul de P&S, unde NEAPARAT SĂ-ŞI SALVEZE TOATE laboratoarele (vor avea nevoie de ele pe parcurs şi, MAI ALES, la examenul practic de la sfârşitul semestrului, să le spui asta!!!).

Acuma, sub ce nume se poate salva: orice nume care NU E OCUPAT deja în Matlab (funcţiile obişnuite, sin, cos, log, abs, exp, input, max, min, etc), numele poate conţine numerale, dar NU poate începe cu, sau conţine doar, numerale. De asemenea, nu poate conţine operaţii elementare Deci poate fi de ex. "l1" sau labst1, dar nu "1" sau "1lab", sau "lab-ps-1". Asta ca să nu se confunde cu o variabilă sau cu o comandă de operaţie atunci când se lansează execuţia. ATENŢIE: dacă se salvează cu un nume care e deja o funcţie în Matlab, NU VA DA EROARE, se va salva, dar se va schimba funcţia respectivă şi se poate da totul peste cap, deci MARE ATENŢIE!! După ce i-am dat nume, fişierul va avea automat extensia .m (M-file).

Cum se lansează execuţia: fie din fereastra editorului, de sus de la Debug $->$ Run, fie din Command Window, pur şi simplu tastând numele fişierului, fără extensia .m, după prompter. Cea de-a doua metodă e mai simplă, dar trebuie avut grijă ca in Command Window, sus, la Current Directory să fie directorul corect, unde s-a salvat fişierul (se caută de la butonul galben din dreapta).

```
>> lab1ps

z =

     3


t =

     2
```

Altfel, Matlab-ul nu-l găseşte.

```
>> lab1ps
??? Undefined function or variable 'lab1ps'.
```

Obs: de aceea nu putea fi numele fişierului un numeral sau ceva ce începe cu un numeral, pentru că la tastarea numelui în Command Window, Matlab-ul l-ar fi confundat cu un simplu număr, cu o variabilă sau cu o operaţie pe care n-ar fi înţeles-o. Bun, dacă vreau să execut programul respectiv şi pentru alte valori ale lui x şi y, le pot cere de la user, în interiorul programului, cu comanda *input*. Se poate vedea cum funcţionează cu

```
>> help input
 INPUT  Prompt for user input.
    R = INPUT('How many apples') gives the user the prompt in the
    text string and then waits for input from the keyboard.
    The input can be any MATLAB expression, which is evaluated,
```

```
    using the variables in the current workspace, and the result
    returned in R.  If the user presses the return key without
    entering anything, INPUT returns an empty matrix.

    R = INPUT('What is your name','s') gives the prompt in the text
    string and waits for character string input.  The typed input
    is not evaluated; the characters are simply returned as a
    MATLAB string.

    The text string for the prompt may contain one or more '\n'.
    The '\n' means skip to the beginning of the next line. This
    allows the prompt string to span several lines. To output
    just a '\' use '\\'.

    See also keyboard.

    Reference page in Help browser
       doc input
```

(În general, de reţinut, se tastează *help topic* şi apar informaţiile necesare. De cele mai multe ori, e mai simplu decât de căutat prin meniul de Help de sus.) Bun, revenind la *input*. Editǎm în fişier:

```
x=input('give the value for x= ');
y=input('give the value for y= ');
z=x+y
t=x*y
```

pe ecran apare (se aşteaptă valori de la tastatură)

```
>> lab1ps
give the value for x= 2
give the value for y= 3

z =

     5


t =

     6
```

Dacă la input dorim o valoare nenumerică, un şir de caractere, se mai adaugă opţiunea 's' (string of characters), după cum se vede in help input, mai sus.

Celălalt tip de M-file e o funcţie Matlab. Aceasta se defineşte exact pe formatul în care sunt definite funcţiile intrinseci ale Matlab-ului. Deschidem un nou M-file. Aici, de la început trebuie specificat că va fi o funcţie:

```
function y=myfun(x)
y=x+1;
```

Trebuie precizaţi parametrii de intrare (în cazul nostru, x), între paranteze rotunde, parametrii de ieşire (în cazul nostru, y), adică ce anume vrem să ne returneze funcţia respectivă, numele pe care îl dăm funcţiei (în cazul nostru, $myfun$), iar în corpul funcţiei trebuie atribuite valori tuturor parametrilor de ieşire (ca să ştie ce să returneze). Un fişier de tip funcţie se salvează DOAR cu numele pe care l-am dat funcţiei, altfel nu-l găseşte. Deci, când să salvăm acest fişier, automat va apare numele $myfun.m$. Apelul (în Command Window, sau într-un alt M-file) se face în mod obişnuit, cu precizarea, că pentru datele de intrare trebuie să fie atribuite deja valori.

```
>> myfun(3)

ans =

     4
```

sau se poate atribui valoarea funcţiei unei variabile (nu neapărat acelaşi nume cu cel din M-file)

```
>> x=0;
>> myanswer=myfun(x)

myanswer =

     1
>>
```

dar nu se poate simbolic (deci fără o valoare bine definită):

```
>> myfun(y)
??? Undefined function or variable 'y'.
```

Pot fi mai multe date de intrare (ca o funcţie de mai multe variabile), între paranteze rotunde, sau/şi mai multe date de ieşire, între paranteze pătrate (deci un vector). Tuturor variabilelor de ieşire trebuie să li se atribuie valori:

```
function [z,t]=myfun(x,y)
z=x+y;
t=x*y;
```

Când se face apelul unei astfel de funcţii (cu mai multe date de ieşire), pentru a le vedea pe toate, trebuie făcut apelul complet, adică

```
>> [a,b]=myfun(2,3)

a =

     5


b =

     6
```

altfel, cu apel simplu, se va returna DOAR valoarea PRIMEI variabile de ieşire:

```
>> myfun(2,3)

ans =

     5
```

Working with arrays, matrices; Matrix and dot operations

Am menţionat mai devreme că Matlab-ul lucrează foarte uşor cu matrici şi vectori. Astfel, de multe ori se poate evita folosirea unui *for*, se poate lucra direct matricial. În Command Window, să vedem câteva exemple:

```
>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10

>> x+2

ans =

     3     4     5     6     7     8     9    10    11    12

>> a=[1 2; 3 4]

a =

     1     2
     3     4

>> b=[1 0; 0 1]

b =

     1     0
     0     1

>> a+b

ans =

     2     2
     3     5

>> a*b

ans =

     1     2
     3     4
```

```
>> a/2

ans =

    0.5000    1.0000
    1.5000    2.0000
```

Deci se poate opera direct pe matrici, iar operaţiile sunt cele MATRICIALE (se observă la înmulţire). Deci trebuie avut grijă ca dimensiunile matricilor să fie potrivite pentru operaţia respectivă. Altfel, dă eroare:

```
>> a+x
??? Error using ==> plus
Matrix dimensions must agree.

>> a*x
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Dacă dorim ca o anumită operaţie să NU se facă matricial, ci scalar (termen cu termen, ca adunarea), operaţia respectivă se defineşte CU PUNCT în faţă (dot operations). E vorba despre operaţiile care provin din înmulţire, singura care e definită altfel la matrici, decât termen cu termen. Deci avem * şi .* (înmulţire), ^ şi .^ (ridicare la putere), / şi ./ (împărţire, adică înmulţire cu inversa). Exemple:

```
>> x.^2

ans =

     1     4     9    16    25    36    49    64    81   100

>> a.*b

ans =

     1     0
     0     4
```

Obs:
1. Chiar dacă într-un caz sunt posibile ambele operaţii (cu şi fără punct), evident că rezultatele nu coincid, $a * b \neq a. * b$, deci atenţie!!
2. Şi în cadrul fişierelor, se poate lucra direct matricial (cu aceeaşi menţiune, atenţie la dimensiunile matricilor şi la tipul operaţiilor):

```
>> lab1ps
give the value for x= a
give the value for y= b

z =

     2     2
     3     5
```

```
t =

     1     2
     3     4
```

sau

```
>> [C,D]=myfun(a,b)

C =

     2     2
     3     5


D =

     1     2
     3     4
```

dar nu

```
>> [C,D]=myfun(1:10,1:10)
??? Error using ==> mtimes
Inner matrix dimensions must agree.

Error in ==> myfun at 3
t=x*y;
```

Special matrices

Cum Matlab-ul lucrează foarte bine cu matrici, are câteva matrici speciale, care pot fi generate uşor, de orice ordin:
$-zeros(m, n)$ - generează o matrice de zerouri de dimensiunea $m \times n$. Dacă dorim doar un vector, una dintre dimensiuni va fi 1 (vector-linie sau vector-coloană). Dacă dăm o singură dimensiune, matricea generată va fi pătratică de acel ordin:

```
>> zeros(3,4)

ans =

     0     0     0     0
     0     0     0     0
     0     0     0     0

>> zeros(1,3)

ans =

     0     0     0
```

```
>> zeros(2,1)

ans =

     0
     0

>> zeros(3)

ans =

     0     0     0
     0     0     0
     0     0     0
```

$-ones(m, n)$ - la fel ca mai sus, doar că valorile sunt de 1, nu 0:

```
>> ones(2,3)

ans =

     1     1     1
     1     1     1
```

Această matrice, în special, ne va folosi mai târziu, la statistică. Vor fi cazuri când un vector X va trebui sa conţină distribuţia frecvenţelor unui set de date, ceva de genul: valoarea 10 apare de 6 ori, valoarea 12 de 15 ori, etc... În loc să tastăm fiecare valoare de atâtea ori (chiar cu copy-paste), e mai simplu de folosit matricea *ones*:

```
>> X=[10*ones(1,6), 12*ones(1,15)]

X =

  Columns 1 through 11

    10    10    10    10    10    10    12    12    12    12    12

  Columns 12 through 21

    12    12    12    12    12    12    12    12    12    12
```

Despre asta vom mai discuta la timpul respectiv.

În fine,
$-eye(m, n)$ - la fel ca mai sus, matricea unitate (*eye* vine de la identity, nu de la ochi...). Dacă dimensiunile nu sunt egale (cum ştim noi că trebuie să fie matricea identitate), se completează cu zerouri, liniile sau coloanele care sunt în plus:

```
>> eye(5)

ans =
```

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

>> eye(2,3)

ans =

     1     0     0
     0     1     0

>> eye(4,2)

ans =

     1     0
     0     1
     0     0
     0     0
```

Display of results, *fprintf* **and** *format*

Am văzut că dacă nu punem ; la sfârşitul unei atribuiri sau operaţii, Matlab-ul afişează conştiincios toate rezultatele efectuării comenzilor. Dar, când la statistică vom avea volum mare de date, vom dori afişarea rezultatelor într-un mod mai relevant şi mai uşor de citit. Acest lucru se poate face cu comanda $fprintf$.

```
>> help fprintf
 FPRINTF Write formatted data to file.
    COUNT = FPRINTF(FID,FORMAT,A,...) formats the data in the real
    part of array A (and in any additional array arguments), under
    control of the specified FORMAT string, and writes it to the file
    associated with file identifier FID.  COUNT is the number of bytes
    successfully written. FID is an integer file identifier obtained
    from FOPEN. It can also be 1 for standard output (the screen) or 2
    for standard error. If FID is omitted, output goes to the screen.

    FORMAT is a string containing C language conversion specifications.
    Conversion specifications involve the character %, optional flags,
    optional width and precision fields, optional subtype specifier, and
    conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s.
    For more details, see the FPRINTF function description in online help
    (search by Function Name for FPRINTF), or look up FPRINTF in a C
    language manual.

    The special formats \n,\r,\t,\b,\f can be used to produce linefeed,
    carriage return, tab, backspace, and formfeed characters respectively.
    Use \\ to produce a backslash character and %% to produce the percent
    character.
```

```
FPRINTF behaves like ANSI C with certain exceptions and extensions.
These include:
1. Only the real part of each parameter is processed.
2. ANSI C requires an integer cast of a double argument to correctly
   use an integer conversion specifier like d. A similar conversion
   is required when using such a specifier with non-integral MATLAB
   values. Use FIX, FLOOR, CEIL or ROUND on a double argument to
   explicitly convert non-integral MATLAB values to integral values
   if you plan to use an integer conversion specifier like d.
   Otherwise, any non-integral MATLAB values will be outputted using
   the format where the integer conversion specifier letter has been
   replaced by e.
3. The following non-standard subtype specifiers are supported for
   conversion characters o, u, x, and X.
   t    - The underlying C datatype is a float rather than an
          unsigned integer.
   b    - The underlying C datatype is a double rather than an
          unsigned integer.
   For example, to print out in hex a double value use a format like
   '%bx'.
4. FPRINTF is "vectorized" for the case when A is nonscalar. The
   format string is recycled through the elements of A (columnwise)
   until all the elements are used up. It is then recycled in a similar
   manner through any additional array arguments.

For example, the statements

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

create a text file containing a short table of the exponential function:

    0.00    1.00000000
    0.10    1.10517092
         ...
    1.00    2.71828183

See the reference page in the online help for other exceptions,
extensions, or platform-specific behavior.

See also fscanf, sprintf, fwrite, disp, diary, save, input.

Overloaded functions or methods (ones with the same name in other directories)
   help serial/fprintf.m
   help icinterface/fprintf.m

Reference page in Help browser
   doc fprintf
```

Deocamdată să vedem afişarea rezultatelor pe ecran, în Command Window (standard output), deci nu dăm niciun fid (file identifier). Pentru fiecare variabilă pe care vrem să o afişăm, trebuie să dăm un format, cu % în faţă. Formatul este d pentru întreg, f pentru real (floating), e pentru forma exponenţiala, cam astea le folosim noi (s pentru string of caharacters, variabilă non-numerică). Formatul se dă între ghilimele simple, ‘ ’. Apoi virgulă şi inşirarea variabilelor ce se vor afişate:

```
>> x=5

x =

     5

>> y=0.5

y =

    0.5000

>> z=12000

z =

       12000

>> fprintf('%d %f %e', x,y,z)
5 0.500000 1.200000e+004>>
```

Formatul poate fi mai explicit, să spună nu doar de ce tip sunt datele, ci şi din câte cifre, câte înainte, după virgulă, etc. TOT ce apare între ’ ’ şi NU e un format (deci nu e cu %), apare afişat de asemenea. Deci pot fi mesaje, spaţii, virgule, semne de punctuaţie, paranteze, etc., tot ce vrem pentru a face display-ul să arate mai bine şi mai uşor de citit. De asemenea, este \n pentru newline, \t pentru tab (pentru aranjarea frumoasă a datelor una sub alta), etc,. se poate citi de mai sus din *help*.

```
>> fprintf('The answer is: \n %2d, %2.3f, %e\n', x,y,y)
The answer is:
  5, 0.500, 5.000000e-001
```

Acest lucru ne va fi foarte folositor la aranjarea datelor sub formă de tabel, interval, etc.

Mai rar vom avea nevoie, dar pentru a afişa rezultatele într-un alt fişier, se precizează fişierul prin fid (file identifier). Se deschide (sau creează) acest fişier cu $fopen$:

```
>> help fopen
 FOPEN  Open file.
    FID = FOPEN(FILENAME) opens the file FILENAME for read access.
    (On PC systems, fopen opens files for binary read access.)

    FILENAME can be a MATLABPATH relative partial pathname.  If the
    file is opened for reading and it is not found in the current
    working directory, FOPEN searches down MATLAB's search path.
```

```
FID is a scalar MATLAB integer, called a file identifier. You
use the fid as the first argument to other file input/output
routines. If FOPEN cannot open the file, it returns -1.

FID = FOPEN(FILENAME,PERMISSION) opens the file FILENAME in the
mode specified by PERMISSION.  PERMISSION can be:

    'r'     read
    'w'     write (create if necessary)
    'a'     append (create if necessary)
    'r+'    read and write (do not create)
    'w+'    truncate or create for read and write
    'a+'    read and append (create if necessary)
    'W'     write without automatic flushing
    'A'     append without automatic flushing
```

Editǎm fişierul în care am lucrat anterior, lab1ps.m mai adǎugǎm:

```
x=input('give the value for x= ');
y=input('give the value for y= ');
z=x+y;
t=x*y;
fid=fopen('results_file.m','a+');
fprintf(fid, '%%The answers are:\n%%_____________________________\n');
fprintf(fid,'%%the value of z is %5.3f\n',z);
fprintf(fid,'%%the value of t is %5.3f\n',t);
fprintf(fid, '%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n');
fclose(fid);
```

care va avea ca urmare creearea fişierului results_file.m cu opţiunea $'a+'$, adică se creează, se poate citi şi, dacă fişierul există deja şi conţine ceva, noile rezultate se adaugă la sfârşit (append). Ca urmare a rulării programului:

```
>> lab1ps
give the value for x= 1.2345
give the value for y= -72.333
```

în fişierul results_file.m va apare

```
%The answers are:
%_________________________
%the value of z is -71.099
%the value of t is -89.295
%%%%%%%%%%%%%%%%%%%%
```

Last remarks

1. Comentariile în Matlab se fac cu simbolul % în faţă.
2. Mai mult, un grup de instrucţiuni pot fi comentate sau des-comentate (!!!) împreună, dacă se selectează, right click, select Comment or Uncomment.
3. Lansarea unui program (execuţia unui fişier) e mai convenabilă din Command Window prin tastrea numelui, pentru că acolo, mergând cu săgeţile up, down, Matlab-ul

merge la comenzile date înapoi sau înainte.
4. Există comenzile *clear all* (şterge variabilele şi funcţiile temporare din directorul curent), *clear var* (şterge variabila var din directorul curent), *clf* (creează o fereastră pentru o figură, ştergând figura dinainte), *clc* (clear Command Window, şterge toate afişajele anterioare din Command Window).
5. Matlab-ul e case-sensitive, face diferenţa între litere mici şi litere mari, deci atenţie!. Şi în legătură cu asta, în mod ciudat, când se tastează *help topic*, comenzile apar cu majuscule, deşi ele funcţionează doar când sunt tastate cu litere mici... o ciudăţenie, dar asta e...