# 1.4. Programarea în MATLAB

## 1.4.1. Fluxul de control

MATLAB are patru structuri de control: instrucţiunea `if`, instrucţiunea de ciclare `for`, instrucţiunea de ciclare `while` şi instrucţiunea `switch`. Cea mai simplă formă a instrucţiunii `if` este

`if` *expresie*
*instrucţiuni*
`end`

unde secvenţa *instrucţiuni* este executată dacă părţile reale ale elementelor lui `expresie` sunt toate nenule. Secvenţa de mai jos interschimbă `x` şi `y` dacă `x` este mai mare decât `y`:

```
if x > y
   temp = y;
   y = x;
   x = temp;
end
```

Atunci când o instrucţiune `if` este urmată în aceeaşi linie de alte instrucţiuni, este nevoie de o virgulă pentru a separa `if`-ul de instrucţiunea următoare:

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

Alternativa se implementează cu `else`, ca în exemplul

```
a = pi^exp(1); c = exp(pi);
if a >= c
   b = sqrt(a^2-c^2)
else
   b = 0
end
```

În fine, se pot introduce teste suplimentare cu `elseif` (de notat că nu este nici un spaţiu între `else` şi `if`):

```
>> if a >= c
      b = sqrt(a^2-c^2)
   elseif a^c > c^a
      b = c^a/a^c
   else
      b = a^c/c^a
   end
```

Într-un test `if` de forma „`if` *condiţie1* & *condiţie2*”, *condiţie2* nu este evaluată dacă *condiţie1* este falsă (aşa-numită evaluare prin scurtcircuit). Acest lucru este util când evaluarea lui *condiţie2* ar putea da o eroare — probabil din cauza unei variabile nedefinite sau a unei depăşiri de indice.

Ciclul `for` este una dintre cele mai utile construcţii MATLAB, deşi codul este mai compact fără ea. Sintaxa ei este

`for` *variabilă* = *expresie*

*instrucţiuni*
`end`

De obicei, *expresie* este un vector de forma `i:s:j`. Instrucţiunile sunt executate pentru *variabilă* egală cu fiecare element al lui *expresie* în parte. De exemplu, suma primilor 25 de termeni ai seriei armonice $1/i$ se calculează prin

```
>> s = 0;
>> for i = 1:25, s = s + 1/i; end, s
s =
    3.8160
```

Un alt mod de a defini *expresie* este utilizarea notaţiei cu paranteze pătrate:

```
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
```

Ciclurile `for` pot fi imbricate, indentarea ajutând în acest caz la creşterea lizibilităţii. Editorul-debuger-ul MATLAB poate realiza indentarea automată. Codul următor construieşte o matrice simetrică 5 pe 5 , `A`, cu elementul $(i, j)$ egal cu $i/j$ pentru $j \geq i$:

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

Expresia din ciclul `for` poate fi o matrice, în care caz lui *variabilă* i se atribuie succesiv coloanele lui *expresie*, de la prima la ultima. De exemplu, pentru a atribui lui `x` fiecare vector al bazei canonice, putem scrie `for x=eye(n), ..., end`.

Ciclul `while` are forma
`while` *expresie*
*instrucţiuni*
`end`

Secvenţa *instrucţiuni* se execută atât timp cât *expresie* este adevărată. Exemplul următor aproximează cel mai mic număr nenul în virgulă flotantă:

```
x = 1;
while x>0, xmin = x; x = x/2; end, xmin
xmin =
  4.9407e-324
```

Execuţia unui ciclu `while` sau `for` poate fi terminată cu o instrucţiune `break`, care dă controlul primei instrucţiuni de după `end`-ul corespunzător. Construcţia `while 1`, ..., `end`, reprezintă un ciclu infinit, care este util atunci când nu este convenabil să se pună testul la începutul ciclului. (De notat că, spre deosebire de alte limbaje, MATLAB nu are un ciclu „repeat-until”.) Putem rescrie exemplul precedent mai concis prin

```
x = 1;
while 1
    xmin = x;
    x = x/2;
    if x == 0, break, end
end
xmin
```

Într-un ciclu imbricat un `break` iese în ciclul de pe nivelul anterior.

Instrucţiunea `continue` cauzează trecerea controlului la execuţia unui ciclu `for` sau `while` următoarei iteraţii, sărind instrucţiunile rămase din ciclu. Un exemplu trivial este:

```
for i=1:10
   if i < 5, continue, end
   disp(i)
end
```

care afişează întregii de la 5 la 10.

Structura de control cu care încheiem este instrucţiunea `switch`. Ea constă din „`switch` *expresie*” urmată de o listă de instrucţiuni „`case` *expresie instrucţiuni*”, terminată opţional cu „`otherwise` *instrucţiuni*” şi urmată de `end`. Exemplul următor evaluează $p$-norma unui vector `x` pentru trei valori ale lui $p$:

```
switch(p)
   case 1
       y = sum(abs(x));
   case 2
       y = sqrt(x'*x);
   case inf
       y = max(abs(x));
   otherwise
       error('p poate fi 1, 2 sau inf.')
end
```

Funcţia `error` generează un mesaj de eroare şi opreşte execuţia curentă. Expresia ce urmează după `case` poate fi o listă de valori delimitate de acolade. Expresia din `switch` poate coincide cu orice valoare din listă:

```
x = input('Enter a real number: ')
switch x
   case {inf, -inf}
      disp('Plus or minus infinity')
   case 0
      disp('Zero')
   otherwise
      disp('Nonzero and finite')
end
```

Construcţia `switch` din MATLAB se comportă diferit de cea din C sau C++ : odată ce MATLAB a selectat un grup de expresii `case` şi instrucţiunile sale au fost executate, se dă controlul primei instrucţiuni de după `switch`, fără a fi nevoie de instrucţiuni `break`.