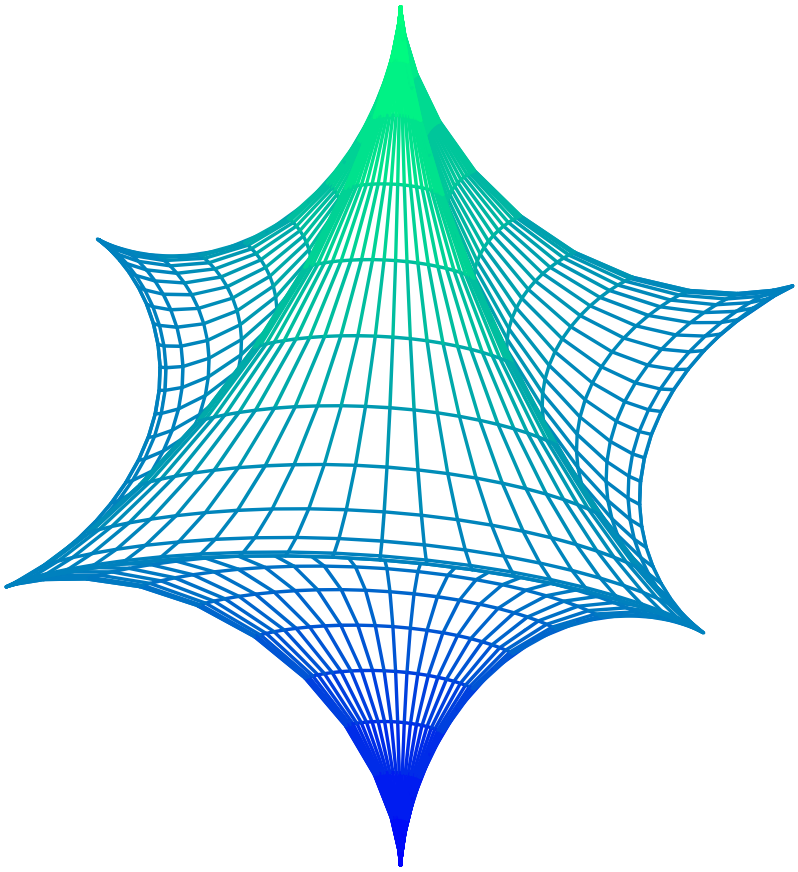
DĂNUȚ ZAHARIEA

LIMBAJE DE PROGRAMARE STRUCTURATĂ

APLICAȚII MATLAB



IAȘI, 2017

Dănuț ZAHARIEA

LIMBAJE DE PROGRAMARE STRUCTURATĂ
APLICAȚII
MATLAB

Iași, 2017

CAPITOLUL 1
OPERAŢII ARITMETICE CU SCALARI

1.1. DECLARAREA VARIABILELOR

Declararea variabilelor se realizează prin operaţia de atribuire („=”), prin care unei variabile oarecare, programul îi va atribui o anumită expresie:

```
NumeVariabilă=expresie
```

1.2. OPERATORI

Programul lucrează cu trei tipuri de operatori: operatori aritmetici, operatori relaţionali şi operatori logici:

- Principalii operatorii aritmetici utilizaţi la definirea expresiilor aritmetice sunt:

Operator	Descriere
+	Adunare, `x+y`
-	Scădere, `x-y`
*	Înmulţire, `x*y`
/	Împărţire, `x/y=x:y`
\	împărţire la stânga, `x\y=y:x`
^	ridicare la putere, `x^y`
'	Transpunere, `x'`

- Principalii operatori relaţionali utilizaţi la compararea variabilelor sunt:

Operator	Descriere
<	Mai mic
>	Mai mare
<=	Mai mic sau egal
>=	Mai mare sau egal
==	Egal
~=	Diferit

- Principalii operatori logici utilizaţi pentru obţinerea expresiilor logice sunt:

Operator	Descriere
&	AND
|	OR
~	NOT

Ordinea de precedenţă a operatorilor stabileşte ordinea în care se vor executa operaţiile aritmetice, relaţionale şi logice din cadrul unei expresii. În cadrul aceluiaşi nivel de precedenţă, operatorii având acelaşi nivel de precedenţă se vor evalua de la stânga la dreapta, mai puţin parantezele care se vor evalua de la interior la exterior.

Principalele reguli de precedență sunt:

Nivel de precedență		Operație
1		Paranteze, `()`
2		Transpunere (`'`) și ridicare la putere (`^`)
3		NOT (~)
4		Înmulțire (`*`) și împărțire (`/`, `\`)
5		Adunare (+) și scădere (-)
6		Operatorul :
7		Operatorii relaționali, <, <=, >, >=, ==, ~=
8		AND
9		OR

1.3. FUNCŢII MATEMATICE ELEMENTARE

Funcțiile matematice elementare definite în program se grupează, în principal, în următoarele categorii: funcții exponențiale, funcții trigonometrice, funcții hiperbolice, funcții pentru manipularea numerelor complexe, funcții pentru aproximarea numerelor.

- **Funcții exponențiale**:

Funcția	Descriere
`exp`	Funcția exponențială, `exp(x)` =e^x
`expm1`	Calculează `exp(x)-1`
`log`	Logaritm natural, `log(x)` =ln(x)
`log10`	Logaritm în baza 10, `log10(x)` =$\log_{10}(x)$
`log2`	Logaritm în baza 2, `log2(x)` =$\log_2(x)$
`sqrt`	Rădăcina pătrată `sqrt(x)` =$\sqrt{x}$
`pow2(e)`	Calculează numărul `pow2(e)` = 2^e
`pow2(m,e)`	Calculează numărul real în virgulă mobilă `pow2(m,e)` =$m \cdot 2^e$

- **Funcții trigonometrice**:

Funcția	Descriere
`sin(x)`	Funcții trigonometrice directe având argumentul x exprimat în radiani.
`cos(x)`	
`tan(x)`	
`cot(x)`	
`sec(x)`	
`csc(x)`	
`sind(x)`	Funcții trigonometrice directe având argumentul x exprimat în grade.
`cosd(x)`	
`tand(x)`	
`cotd(x)`	
`secd(x)`	
`cscd(x)`	
`asin(x)`	Funcții trigonometrice inverse furnizând rezultatul în radiani.
`acos(x)`	
`atan(x)`	
`acot(x)`	
`asec(x)`	
`acsc(x)`	

`asind(x)`
`acosd(x)`
`atand(x)`
`acotd(x)`
`asecd(x)`
`acscd(x)`
	Funcții trigonometrice inverse furnizând rezultatul în grade.

- **Funcții hiperbolice:**

`sinh(x)`
`cosh(x)`
`tanh(x)`
`coth(x)`
`sech(x)`
`csch(x)`
	Funcții hiperbolice directe.
	$\sinh x = \frac{e^x - e^{-x}}{2}$; $\cosh x = \frac{e^x + e^{-x}}{2}$; $\tanh x = \frac{\sinh x}{\cosh x}$
	$\coth x = \frac{1}{\tanh x}$; $\operatorname{sech} x = \frac{1}{\cosh x}$; $\operatorname{csch} x = \frac{1}{\sinh x}$

`asinh(x)`
`acosh(x)`
`atanh(x)`
`acoth(x)`
`asech(x)`
`acsch(x)`
	Funcții hiperbolice inverse.
	$\operatorname{asinh} x = \ln\left(x + \sqrt{x^2 + 1}\right)$; $\operatorname{acosh} x = \ln\left(x + \sqrt{x^2 - 1}\right)$
	$\operatorname{atanh} x = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$; $\operatorname{acoth} x = \operatorname{atanh}\left(\frac{1}{x}\right)$
	$\operatorname{asech} x = \operatorname{acosh}\left(\frac{1}{x}\right)$; $\operatorname{acsch} x = \operatorname{asinh}\left(\frac{1}{x}\right)$

- **Funcții pentru manipularea numerelor complexe:**

Funcția	Descriere
`i, j`	Unitatea imaginară implicită
`real(z)`	Determină partea reală a numărului complex `z`
`imag(z)`	Determină partea imaginară a numărului complex `z`
`z=complex(a,b)` `z=a+bi`	Definește numărul complex `z` având partea reală `a` și partea imaginară `b`
`zc=conj(z)`	Calculează conjugatul numărului complex z
`r=abs(z)`	Calculează modulul `r` al numărului complex `z=x+iy` $r = \sqrt{x^2 + y^2}$
`f=angle(z)`	Calculează argumentul `f` al numărului complex `z=x+iy` $f = \operatorname{arctg}\left(\frac{y}{x}\right)$
`z=r*exp(i*f)` `z=r*(cos(f)+i*sin(f))`	Definește forma polară a numărului complex `z`
`isreal(z)`	Verifică dacă numărul z este real. Rezultatul funcției este 1 dacă numărul `z` este real, și 0 dacă numărul `z` este complex.

- **Funcții pentru aproximarea numerelor:**

Funcția	Descriere
`round(x)`	Rotunjește numărul `x` la cel mai apropiat număr întreg
`floor(x)`	Rotunjește numărul `x` la cel mai apropiat număr întreg spre -∞
`ceil(x)`	Rotunjește numărul `x` la cel mai apropiat număr întreg spre +∞
`fix(x)`	Rotunjește numărul `x` la cel mai apropiat număr întreg spre 0
`rats(x)`	Aproximarea numărului `x` folosind exprimarea cu numere raționale
`rat(x)`	Aproximarea numărului `x` folosind exprimarea cu fracții continue

1.4. PROBLEME

Problema 1.1

Se consideră variabilele:

$$a = 2$$
$$b = 3$$
$$c = 4$$

Să se calculeze:

a) $u = \frac{a+b}{c}$

b) $v = \frac{a+b}{c-1}$

c) $v = \frac{a+b}{c} - 1$

d) $k = \frac{1}{a} + \frac{1}{b} + \frac{1}{c-1}$

e) $g = a^{\frac{b-1}{b}}$

f) $w = e^{a+b}$

g) $u = 2^a + \frac{a+b}{c^2}$

h) $\beta = \frac{a}{b + \frac{1}{c}}$

i) $a + b = r$

j) $f = \frac{a}{b} + \frac{1}{c} - q$

k) $e = k^a + (k+1)^b + (k-1)^c$

l) $h = \frac{a}{1 + \frac{b}{1 + \frac{1}{c}}}$

Problema 1.2

Se consideră variabilele:

$$x = 2.3$$
$$y = 3.213$$
$$z = 7.05$$

Să se calculeze:

a) $a = \sqrt{x + \frac{y}{z}}$

b) $b = \sqrt[3]{x + y - z}$

c) $c = \pi^{x+1} - \frac{y+z}{\pi}$

d) $d = \sqrt{a^2 + b^2}$

Problema 1.3

Se consideră variabilele:

$$a = e$$
$$b = 1.21$$
$$c = 3.9$$

Să se calculeze:

a) $x = \log_2 a$

b) $y = \log_{10} a$

$$z = \ln a$$

c)

$$u = \frac{\log_2 b}{\log_{10} c} \ln\left(\frac{a}{b+c}\right)$$

d)

$$v = \frac{1 + \ln b}{c^3 + \log_2 e}$$

e)

$$w = \frac{\ln a \ln b \ln c}{1 + \log_2(abc)}$$

f)

Problema 1.4

Se consideră variabilele:

$$a = 2$$
$$b = 10$$
$$x = 42.85$$

Să se verifice relațiile:

$$\log_a x = \frac{\ln x}{\ln a}$$

a)

$$\log_a x = \frac{\log_b x}{\log_b a}$$

b)

$$\log_a b = \frac{1}{\log_b a}$$

c)

$$\ln a^b \cdot \frac{\ln(a \cdot b)}{\ln \frac{a}{b}} = b \ln a \cdot \frac{\ln a + \ln b}{\ln a - \ln b}$$

d)

Problema 1.5

Se consideră variabilele:

$$x = 2$$
$$y = 5$$

Să se calculeze:

$$z = \frac{x^3 y}{x - y}$$

a)

$$u = \frac{2x^3}{x^3 - 1} + \frac{2y^3}{y^3 - 1} + \frac{xy}{xy - 1}$$

b)

$$v = \frac{3}{2}xy + \frac{1}{5}\frac{x}{y} - \frac{8}{3}\frac{y}{x}$$

c)

$$p = 3x^{\frac{1}{3}} + 5y^{1/5} + \frac{1}{4}x^{0.25} - y^{1.4}$$

d)

$$w = \left(\frac{1}{\frac{1}{x} + 1}\right)^2 - \left(\frac{1}{\frac{1}{y} + 1}\right)^{\frac{1}{2}}$$

e)

$$q = \left(\frac{3}{4}x - \frac{2}{5}y\right)^{-3/4} \cdot \left(\frac{x}{y+1}\right)^{2/5} + \frac{1}{xy}$$

f)

Problema 1.6

Se consideră variabilele:

$$a = 1.12; b = 2.28; c = 5.12; d = 0.23; f = 3.85.$$

Să se calculeze:

$$x = 1 + \frac{a}{b} + \frac{c}{f^2} - \frac{a+b}{c-f}$$

a)

$$y = \frac{b-a}{d-c} \Big/ \frac{b-c}{a-d}$$

b)

$$z = \frac{1}{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}}$$

c)

$$u = ab \cdot \frac{1}{c} \cdot \frac{f^3}{3},\ p = \frac{\sqrt{a+1}}{\frac{\sqrt{b+1}}{\sqrt{c+1}}}$$

d)

$$v = \frac{1}{2} a\sqrt{2b^3} \cdot \sqrt[3]{\frac{a}{2b}}$$

e)

$$w = \sqrt[2]{2a+1} - \sqrt[4]{\frac{2a}{b-1}} + \sqrt[3]{f^{2a}}$$

f)

Problema 1.7

Se consideră variabilele:

$$x = 2.7; y = e;$$

Să se calculeze:

$$u = \log_2 x + \log_{10} x - \frac{\log_2(x+y)}{\ln y}$$

a)

$$v = e^{x+1} - \frac{e^y}{\ln y}$$

b)

$$w = e^{x+y} + x^{\frac{y+1}{y}}$$

c)

$$p = \frac{1}{x}\ln(y+1) - \frac{1}{y}\ln(x+1)$$

d)

Problema 1.8

Se consideră variabilele:

$$\alpha = 30°$$
$$\beta = \frac{\pi}{2}$$

Să se calculeze:

$$b = \cos\frac{\beta}{5}\left(1 - \sin\frac{\alpha}{2}\right)$$

a)

$$c = \frac{\pi}{3}\sin(\alpha+\beta)$$

b)

$$d = \frac{1-\sin\alpha}{1+\cos\alpha}$$

c)

$$d = \left(\frac{1}{2} + \operatorname{tg}\alpha\right)\sin\left(2\pi\frac{\alpha}{\beta}\right)$$

d)

$$e = \frac{\sin(2\pi\alpha+\beta)}{\cos(\pi\alpha-\beta)}$$

e)

$$u = \sin(\alpha+\pi)\cdot\cos(\beta/3)$$

f)

Problema 1.9

Se consideră variabila:

$$\alpha = 35°$$

Să se verifice următoarele relații:

$$\sin 3\alpha = 3\sin\alpha - 4\sin^3\alpha$$

a)

$$\cos 3\alpha = -3\cos\alpha + 4\cos^3\alpha$$

b)

$$\sin 2\alpha = 2\sin\alpha\cos\alpha = \frac{2\operatorname{tg}\alpha}{1+\operatorname{tg}^2\alpha}$$

c)

$$\cos 2\alpha = 2\cos^2\alpha - 1 = 1 - 2\sin^2\alpha = \cos^2\alpha - \sin^2\alpha = \frac{1-\operatorname{tg}^2\alpha}{1+\operatorname{tg}^2\alpha}$$

d)

$$\sin 4\alpha = 8\cos^3\alpha\cdot\sin\alpha - 4\cos\alpha\sin\alpha$$

e)

$$\cos 4\alpha = 8\cos^4\alpha - 8\cos^2\alpha + 1$$

f)

Problema 1.10

Se consideră variabilele:

$$\alpha = 25.13°$$
$$\beta = 48.5°$$

Să se verifice următoarele relații:

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cdot\cos\frac{\alpha-\beta}{2}$$ a)

$$\sin\alpha - \sin\beta = 2\cos\frac{\alpha+\beta}{2}\cdot\sin\frac{\alpha-\beta}{2}$$ b)

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cdot\cos\frac{\alpha-\beta}{2}$$ c)

$$\cos\alpha - \cos\beta = -2\sin\frac{\alpha+\beta}{2}\cdot\sin\frac{\alpha-\beta}{2}$$ d)

$$\mathrm{tg}\,\alpha + \mathrm{tg}\,\beta = \frac{\sin(\alpha+\beta)}{\cos\alpha\cos\beta}$$ e)

$$\mathrm{tg}\,\alpha - \mathrm{tg}\,\beta = \frac{\sin(\alpha-\beta)}{\cos\alpha\cos\beta}$$ f)

Problema 1.11

Se consideră variabila:

$$\alpha = 45.87°$$

Să se verifice următoarele relații:

$$\sin\alpha = -i\cdot\sinh(i\alpha)$$ a)

$$\cos\alpha = \cosh(i\alpha)$$ b)

$$\tan\alpha = -i\cdot\tanh(i\alpha)$$ c)

$$\mathrm{ctg}\,\alpha = i\cdot\mathrm{ctgh}(i\alpha)$$ d)

$$\sec\alpha = \mathrm{sech}(i\alpha)$$ e)

$$\csc\alpha = i\cdot\mathrm{csch}(i\alpha)$$ f)

Problema 1.12

Se consideră numărul complex definit prin:

$$x = 3,\ y = 7,\ z = x + iy$$

Să se verifice următoarele relații:

$$\sin z = \sin x\cosh y + i\cos x\sinh y$$ a)

$$\cos z = \cos x\cosh y - i\sin x\sinh y$$ b)

$$\mathrm{tg}\,z = \frac{\sin 2x + i\sinh 2y}{\cos 2x + \cosh 2y}$$ c)

$$\cot z = \frac{\sin 2x - i\sinh 2y}{-\cos 2x + \cosh 2y}$$ d)

Problema 1.13

Se consideră numărul complex definit prin:

$$x = 2{,}5,\ y = 4,\ z = x + iy$$

Să se calculeze modulul r și argumentul φ al numărului complex.

Să se verifice următoarele relații:

$$|\sin z| = \sqrt{\sin^2 x + \sinh^2 y}$$ a)

$$|\cos z| = \sqrt{\cos^2 x + \sinh^2 y}$$ b)

$$|\mathrm{tg}\,z| = \sqrt{\frac{\cosh 2y - \cos 2x}{\cosh 2y + \cos 2x}}$$ c)

Problema 1.14.

Se consideră variabilele:

$$x_1 = 2.1283$$
$$x_2 = 2.73$$
$$x_3 = -3.856$$
$$x_4 = -3.19$$

Să se determine:

- Aproximarea la cel mai apropiat număr întreg;
- Aproximarea la cel mai apropiat număr întreg spre -∞;
- Aproximarea la cel mai apropiat număr întreg spre +∞;
- Aproximarea la cel mai apropiat număr întreg spre 0;
- Aproximarea folosind exprimarea cu numere raționale;
- Aproximarea folosind exprimarea cu fracții continue.

Problema 1.15.

Să se scrie relațiile matematice corespunzătoare următoarelor expresii MATLAB:

- `Cp=P/(rho/2*pi*R^2*Vinf^3)`
- `T=Ct*pi*D^2/4*rho/2*Vinf^2`
- `Q=pi*D^2/4*C*E*sqrt(2/r*dp)`
- `v=(x-0.25)/(x+1)+1/x/(x+1)+1`
- `pm=(p2-p1)/2*(1-x/sqrt(R^2+x^2))`
- `u=(x+1/(y-1))*exp(x^((k-1)/k))+1`
- `t=(1/(1/x+1))^2-(1/(1/y+1))^(1/3)`
- `vr=-1/2*a*Vinf*R^2*r/(x^2+r^2)^(3/2)`
- `f=B*(1-L*x)*sqrt(1+L^2)/(2*L)*(1-r/R)`
- `Dh=Dp+1/2*rho*omega^2*r^2-Zd/2*rho*U^2`
- `G=k*c*Vinf*sqrt(1+1/L^2*(r/R)^2)*L/(r/R)`
- `vx=-1/2*a*Vinf*R^2*x/(x^2+y^2+z^2)^(3/2)`
- `w=log(x+1)+1/exp(x+y)+log10(2*x+1)/log2(y)`
- `W=sqrt(Vinf^2*(1+a)^2+OMEGA^2*r^2*(1-ft)^2)`
- `eta=(1-Cx/Cz*tan(beta0))/(1+Cx/Cz*cot(beta0))`
- `C3=45*pi/256*(1-Z^2)^(3/4)*(1-sin(alphaD))^(3/2)`
- `CT=2*k^2*(1+2*k^2)/(1+k^2)-4*k^4*log((1+k^2)/k^2)`
- `p=sin(x+1/pi)/cos(2*y+pi)+tan(2*pi*(x+1))+1/sin(3*pi+x/y)`
- `L=sin(beta)*(2*cos(beta)-1)/((1+2*cos(beta))*(1-cos(beta))`

Problema 1.16.

Se consideră variabilele:

$$a = 2$$
$$b = 3$$
$$c = 4$$

Să se calculeze:

$$u_1 = \sqrt{1+2^{\pi}} + \frac{a+\pi}{2c^2+1} - e^{\frac{a+1}{a}} + \sqrt[3]{2a+b+\frac{1}{c}} + \ln\frac{2+a}{b+c}$$

$$u_2 = \sqrt{b^{a+1}} + \frac{a+1}{1+bc^a} + e^{\frac{a+1}{b}} + \sqrt[3]{2\pi+\frac{b+1}{c}-1} + \ln\frac{a+b}{b+c}$$

$$u_3 = \sqrt{\pi^{a+1}} + \frac{b+a^c}{b+c^a} + e^{\frac{a+1}{b}} + \sqrt[3]{2a+\frac{b}{c+1}+1} + \ln\frac{a+2b}{2b+c}$$

$$u_4 = \sqrt{\pi^{a+b}} + \frac{1+a^c}{\pi-c^a} + e^{\frac{\pi+1}{b}} + \sqrt[3]{2\pi+\frac{b}{c-1}+1} + \ln\frac{2a+b}{b+2c}$$

$$u_5 = \sqrt{\pi^{1+b}} + \frac{1+a^c}{a(\pi-c^a)} + e^{\frac{\pi+1}{b-1}} + \sqrt[3]{\pi^2+\frac{a+b}{c-1}+1} + \ln\frac{a(a+b)}{b+c}$$

$$u_6 = \sqrt{\pi^{1+\pi}} + \frac{1+a^{\pi}}{a(b+c^a)} + e^{\frac{1}{\pi-1}} + \sqrt[3]{\pi^3+\frac{a+1}{b+1}+c} + \ln\frac{a(a+b)}{b(b+c)}$$

$$u_7 = \sqrt{a^{\pi-1}} + \frac{a+b^{\pi}}{\pi(b-c^{\pi})} + e^{1+\frac{1}{\pi}} + \sqrt[3]{\pi^{1/a}+\frac{a+1/b}{b+1/c}+1} + \ln\frac{2(a^c+b)}{3(b+c^a)}$$

$$u_8 = \sqrt{(a+1)^{\pi}} + \frac{c+a^{\pi}}{b+a^{\pi}} + e^{\frac{2a+1}{b}} + \sqrt[3]{\frac{1}{\pi}+\frac{2a+1}{2b+1}+1} + \ln\frac{3(a^{\pi}+1)}{7(c^{\pi}+1)}$$

$$u_9 = \sqrt{(1+a^2)^b} + \frac{1+a+b}{c(a+b^{\pi})} + e^{\frac{\pi+1}{\pi^2}} + \sqrt[3]{\frac{\pi}{a}+\frac{3\pi+1}{\pi(a+1)}+1} + \ln\frac{c+\pi}{\pi^2}$$

$$u_{10} = \sqrt{1+\frac{a}{b+a}} + \frac{a}{\pi(1+\pi^{0.5})} + e^{\frac{a+\pi}{a-\pi}} + \sqrt[3]{\frac{1}{\pi}+\frac{\pi+a}{b(c-1)}} + \log_{10}\frac{2a+3b}{c-1}$$

$$u_{11} = \sqrt{\pi+\frac{a}{b+c}} + \frac{\pi+1}{c(1+\pi^{2a})} + e^{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}} + \sqrt[3]{\frac{\pi-1}{\pi+1}} + \log_2\frac{1+\pi+\pi^2}{a+b+c}$$

$$u_{12} = \sqrt{1+\frac{b}{1+a}} + \frac{1+a}{c(1+a^b)} + e^{\frac{a+b}{2}} + \sqrt[3]{c+\frac{a+1}{b+1}} + \log_2\frac{1+a+b}{\sqrt{c}}$$

$$u_{13} = \sqrt{\frac{a^2}{b^2+c^2}} + \frac{\sqrt{a}}{a(b+c)} + e^{a+b+1/c} + \sqrt[3]{\frac{a+b}{c+\pi^2}} + \log_2\frac{a+b+1/c}{\pi c^2}$$

$$u_{14} = \sqrt{\frac{e^a}{e^b+e^c}} + \frac{\sqrt{\pi+1}}{a+b+c} + e^{\frac{a+b}{c}} + \sqrt[3]{a^2+b^2+c^2} + \log_2\frac{a+b+c}{\sqrt{\pi+1}}$$

CAPITOLUL 2
FIȘIERE DE TIP SCRIPT. FUNCȚII

Principalele moduri de lucru în limbajul de programare MATLAB sunt:

- Modul de lucru în linie de comandă;
- Modul de lucru bazat pe fișiere de tip `script`;
- Modul de lucru bazat pe definirea funcțiilor:
  - Funcții definite prin metoda fișierelor de tip `function`,
  - Funcții definite prin metoda `anonymous`.

2.1. MODUL DE LUCRU BAZAT PE FIȘIERE SCRIPT

Un fișier `script` conține o succesiune de instrucțiuni care se vor executa automat la lansarea fișierului în execuție ca și cum ar fi tastate independent în fereastra de comenzi a programului. Folosirea fișierelor `script` este cu atât mai indicată cu cât creste complexitatea instrucțiunilor de executat.

Fișierele `script` nu au definite nici variabile de intrare și nici variabile de ieșire, însă toate variabilele definite sau calculate în structura fișierului, rămân după executarea fișierului `script`, în spațiul de lucru al programului. Fișierele `script` pot utiliza datele create în structura fișierului, sau datele existente în spațiul de lucru al programului.

La utilizarea fișierelor `script` trebuie să se considere următoarele observații generale:

- La salvarea unui fișier `script` trebuie avut în vedere faptul că extensia fișierului este în mod implicit .m și nu trebuie modificată. Extensia implicită este introdusă automat la salvare.
- Numele fișierelor `script` trebuie să înceapă cu o litera. Numele fișierelor poate să conțină caracterul _, dar nu trebuie să conțină spatii libere.
- În mod implicit, fișierele `script` se salvează în directorul curent implicit (`C:\Users\UserName\Documents\MATLAB`) care, în general, este adăugat în structura relevantă de directoare a programului. În cazul în care salvarea fișierelor se face în alte directoare se va verifica dacă acestea sunt adăugate în căile de căutare ale programului.
- În structura fișierelor `script` pot fi introduse atât instrucțiuni, grupuri de instrucțiuni reunite în secțiuni de calcul, grupuri de instrucțiuni definite ca funcții, precum și linii de text explicativ. O secțiune de program poate conține la rândul său, atât instrucțiuni, text explicativ, cât și funcții. De asemenea, funcțiile pot conține instrucțiuni, text explicativ, secțiuni de calcul, precum și alte funcții.
- Un text care începe cu `%` este considerat text explicativ, imediat după caracterul `%` și până la sfârșitul liniei curente.

- O zonă de program care începe cu `%%` este considerată o secțiune de program. O secțiune începe deci cu `%%` și se termină fie la întâlnirea, din nou, a caracterelor `%%` (moment în care începe o nouă secțiune), fie la sfârșitul fișierului script respectiv.
- Se recomandă definirea secțiunilor pentru fiecare din grupurile principale de instrucțiuni ale unui fișier script. De exemplu:
    - Secțiunea cu titlul programului, datele de identificare ale autorului, grupul de instrucțiuni `clc`, `clear all`, `close all`.
    - Secțiunea datelor inițiale ale programului.
    - Secțiunea calculelor preliminare.
    - Secțiunea calculelor principale.
    - Secțiunea reprezentărilor grafice.
- Lansarea în execuție a unui fișier script se poate realiza prin mai multe metode:
    - Comanda `Run` (F5) – se execută toate instrucțiunile din fișier.
    - Comanda `Run and Advance` – se execută toate instrucțiunile din secțiunea curentă și se avansează apoi la secțiunea următoare a fișierului.
    - Comanda `Run Section` – se execută toate instrucțiunile din secțiunea curentă.
    - Comanda `Run and Time` – se execută toate instrucțiunile din fișier și se determină timpul total de calcul, precum și timpul de calcul pentru fiecare instrucțiune din fișier.
- Identificarea erorilor nu poate fi realizată decât odată cu lansarea în execuție a fișierului `script`. Rezultă deci, necesitatea lansării repetate în execuția a fișierului `script`, pe întreg parcursul dezvoltării acestuia, pentru a identifica din timp eventualele erori logice sau de sintaxă. Scrierea tuturor instrucțiunilor și apoi lansarea în execuție este o metodă de lucru contraproductivă.
- După editarea și verificarea unui fișier `script`, acesta poate fi ulterior îmbunătățit și eventual distribuit altor utilizatori.

Principalele avantaje ale modului de lucru bazat pe fișiere de tip `script` sunt:

- Posibilitatea de a scrie coduri sursă complexe, cu foarte multe linii.
- Posibilitatea de utilizare repetată a instrucțiunilor din structura fișierului `script`.
- Posibilitatea introducerii de secțiuni pentru gruparea pe anumite criterii a instrucțiunilor din structura fișierului `script`.
- Posibilitatea introducerii de text explicativ pentru adnotarea instrucțiunilor din structura fișierului `script`.
- Posibilitatea apelării fișierelor `script` în structura altor fișiere.
- Posibilitatea transferului rapid al fișierelor `script` către alți utilizatori.

2.2. EDITORUL DE FIȘIERE SCRIPT

Deschiderea unui nou fișier `script` se poate face prin selectarea comenzii `New Script` (sau `New/Script`) din toolbarul meniului `HOME`, sau prin apăsarea tastelor `CTRL+N`.

Se deschide astfel fereastra de editare a fișierelor `script`, în care se vor introduce, la poziția curentă a cursorului, linie după linie, toate informațiile specifice fișierului respectiv, figura 2.1.

Principalele zone ale ferestrei de lucru a editorului de fișiere `script` sunt:

- Bara de titlu a fișierului respectiv. Conține numele fișierului și comenzile pentru manipularea ferestrei grafice: `Minimise` (comanda pentru minimizarea ferestrei), `Maximise` (comanda pentru maximizarea ferestrei) și `Close` (comanda pentru închiderea ferestrei).
- Zona de lucru care definește conținutul fișierului `script`. În această zonă se introduc linie după linie instrucțiunile și comentariile necesare pentru rezolvarea unei anumite probleme. Numerotarea liniilor este automată.
- Meniurile `EDITOR`, `PUBLISH` și `VIEW`.

Principalele elemente ale interfeței editorului de fișiere `script` inclusiv ale toolbarului `EDITOR` sunt prezentate în figura 2.1. Se observă comenzile: `Insert Section` (introducerea unei noi secțiuni în fișier); `Comment` (`CTRL+R`, introducerea unui nou comentariu); comenzile pentru depanarea fișierului grupate în meniul `Breakpoints`; comenzile pentru lansarea în execuție a fișierului (`Run`, `Run and Advance`, `Run Section`, `Run and Time`);

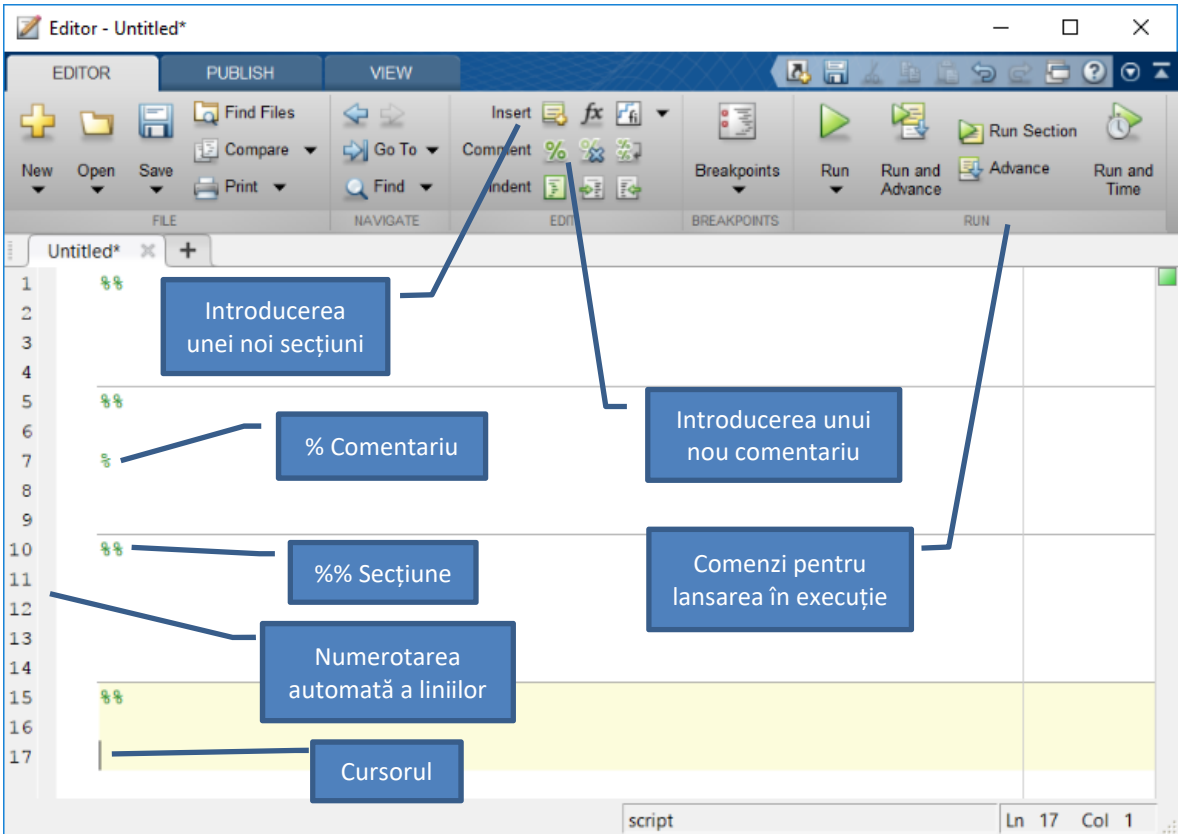


Figura 2.1. Interfața editorului de fișiere. Toolbarul meniului `EDITOR`.

Principalele elemente ale toolbarului `PUBLISH` sunt prezentate în figura 2.2.

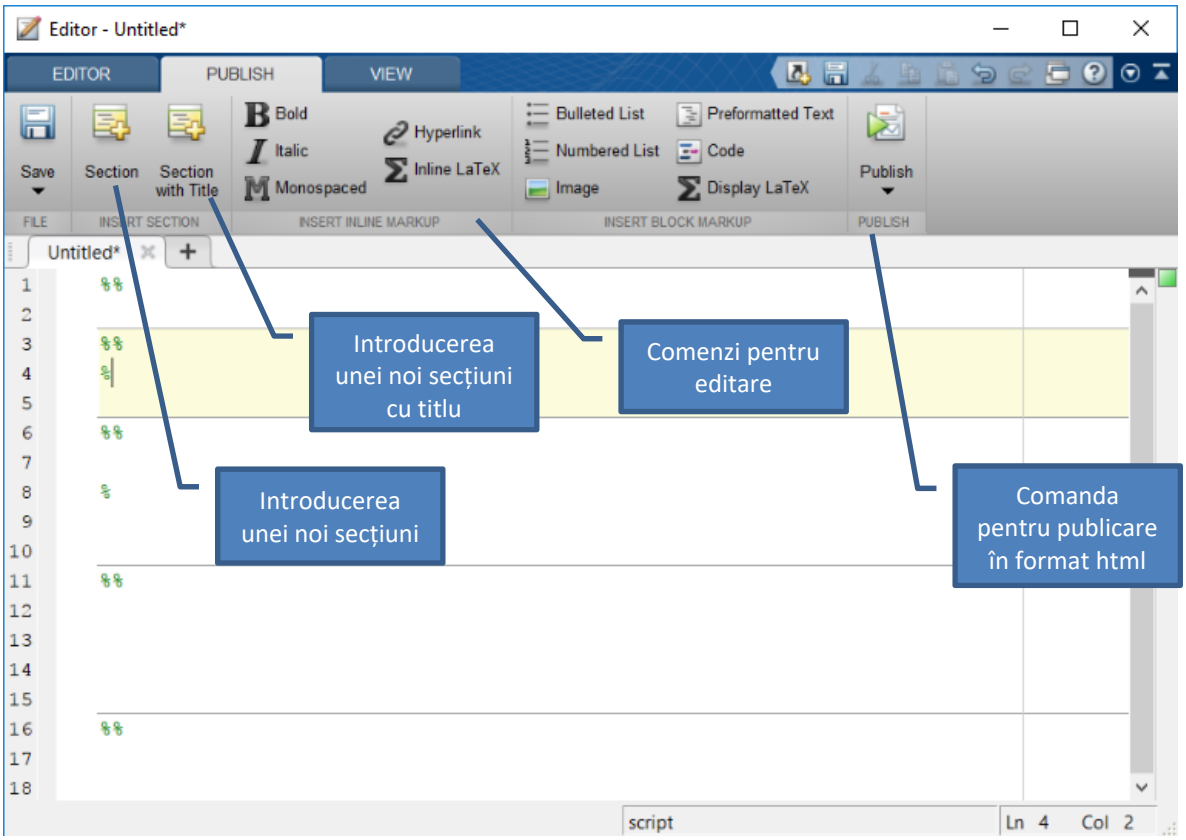


Figura 2.2. Interfața editorului de fișiere. Toolbarul meniului `PUBLISH`.

Principalele elemente ale toolbarului `VIEW` sunt prezentate în figura 2.3.

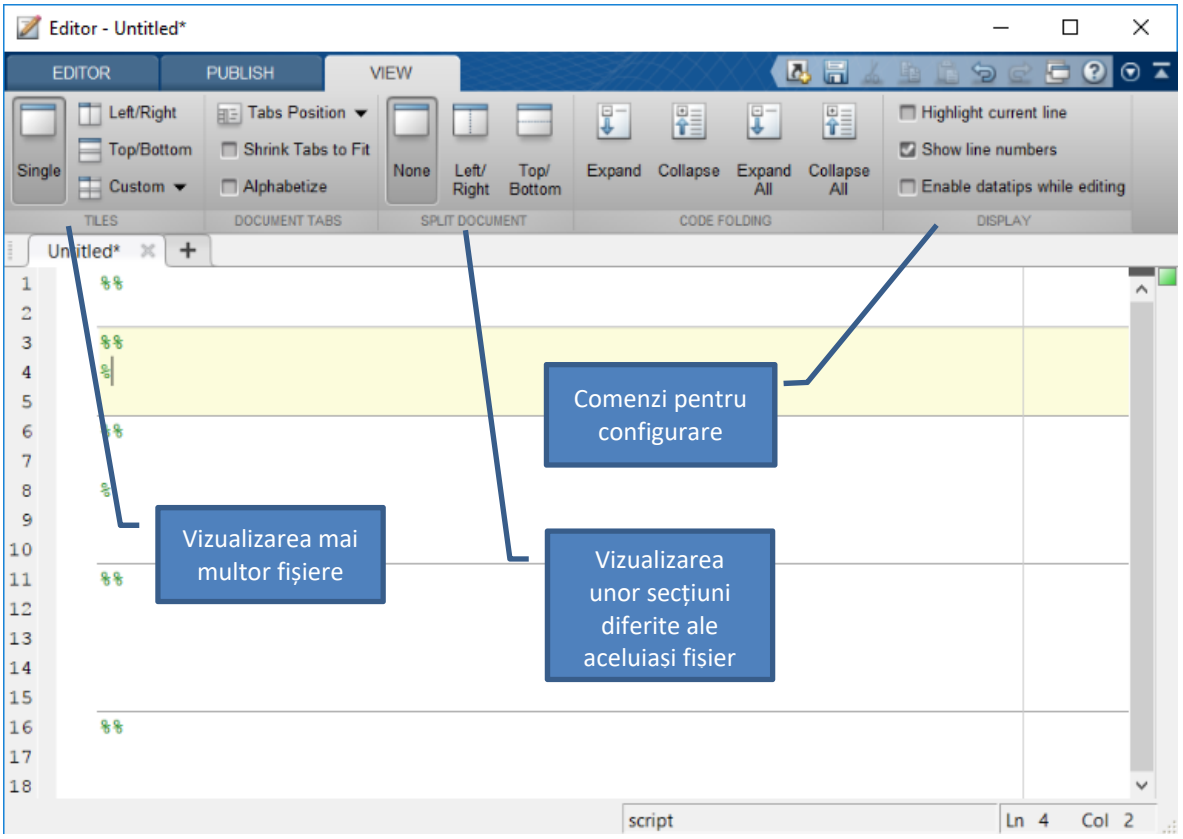


Figura 2.3. Interfața editorului de fișiere. Toolbarul meniului `VIEW`.

2.3. MODUL DE LUCRU BAZAT PE FUNCȚII DEFINITE ÎN FIȘIERE `function`

Deseori apare necesitatea creării unor fișiere particulare (fișierele de tip `function`) care să rezolve o anumită problemă specifică de calcul și care să poată fi utilizate în mod repetat în cadrul unor programe complexe.

Principalele caracteristici ale fișierelor de tip `function` sunt:

- Fișierul de tip `function` este un fișier `script` particular care are proprietatea că își construiește un spațiu de lucru independent, cu variabile independente de spațiul de lucru al programului (variabile locale). Fișierul `function` poate avea însă, în anumite condiții, și variabile globale. În structura unui fișier `function` se pot introduce instrucțiuni, text explicativ și secțiuni de program, ca și în cazul fișierelor de tip `script`.
- Fișierul de tip `function` poate fi apelat în cadrul unui al fișier de tip `function`, în cadrul unui fișier de tip `script`, sau direct în fereastra de comenzi.
- Deschiderea unui nou fișier function se poate face prin selectarea comenzii `New/Function` din toolbarul meniului `HOME`. Se deschide astfel fereastra de editare a fișierelor `function`, care este identică cu fereastra de editare a fișierelor `script`, cu excepția primei linii a fișierului în care apare cuvântul rezervat `function` și a ultimei linii a fișierului în care apare cuvântul rezervat `end`. Cuvântul `function` este un cuvânt rezervat care nu se poate utiliza decât în acest context. Dacă în structura unui fișier `function`, cuvântul rezervat `function` apare de mai multe ori, exceptând prima apariție, toate celelalte apariții definesc sub-funcții. Sub-funcțiile (sub-funcții locale) nu sunt vizibile în exteriorul funcției în care au fost definite.
- Structura generală a unui fișier de tip `function` este prezentată în figura 2.4.

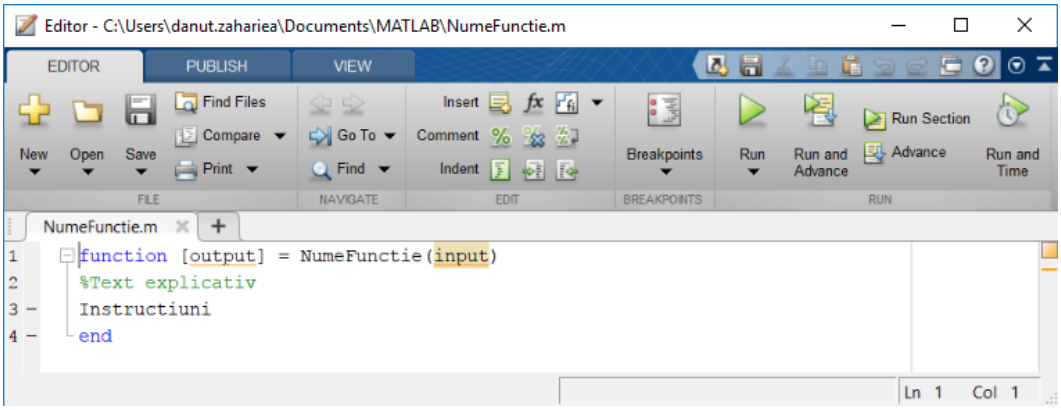


Figura 2.4. Structura generală a unui fișier de tip `function`.

Principalele componente din structura generală a fișierelor de tip `function` sunt:

- `output` reprezintă lista variabilelor de ieșire ale funcției,
- `input` reprezintă lista variabilelor de intrare ale funcției,
- `NumeFunctie` este numele funcției și în același timp și numele fișierului în care se va salva funcția (`NumeFunctie.m`).
- `Text explicativ` reprezintă comentarii necesare pentru documentarea funcției.
- `Instrucțiuni` reprezintă lista instrucțiunilor specifice funcției.

2.4. MODUL DE LUCRU BAZAT PE FUNCȚII DE TIP ANONYMOUS

Funcțiile pot fi definite atât prin metoda fișierelor de tip `function` (funcții cu mai multe variabile de intrare, mai multe variabile de ieșire și mai mult de o singură instrucțiune), cât și prin metoda funcțiilor `anonymous` (funcții cu mai multe variabile de intrare, o singură variabilă de ieșire și doar o singură instrucțiune).

Funcțiile de tip `anonymous` sunt funcții locale care pot fi definite în fereastra de comenzi la prompterul MATLAB, în structura unui fișier de tip `script`, sau într-un fișier de tip `function`.

Structura generală a unei funcții de tip `anonymous` este următoarea:

```
output=@(input) Expresie
```

în care:

- `output` reprezintă variabila de ieșire a funcției și în același timp și numele funcției.
- `input` reprezintă lista variabilelor de intrare ale funcției.
- `Expresie` este expresia matematică de definire a funcției.

2.5. PROBLEME

Problema 2.1

Se consideră două numere: $x_1 = 8$ și x_2=4.

Să se calculeze media aritmetică a celor două numere prin următoarele metode:

- Metoda de lucru în linie de comandă
- Metoda de lucru bazată pe folosirea fișierelor `script`
- Metoda de lucru bazată pe folosirea funcțiilor definite în fișiere
- Metoda de lucru bazată pe folosirea funcțiilor `anonymous`.

Relația pentru calculul mediei aritmetice este:

$$m_a = \frac{x_1 + x_2}{2}$$

Rezolvarea problemei prin metoda de lucru în linie de comandă este prezentată în figura 2.5.

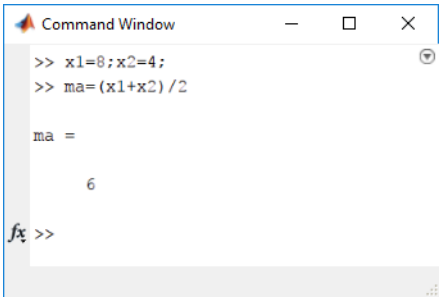
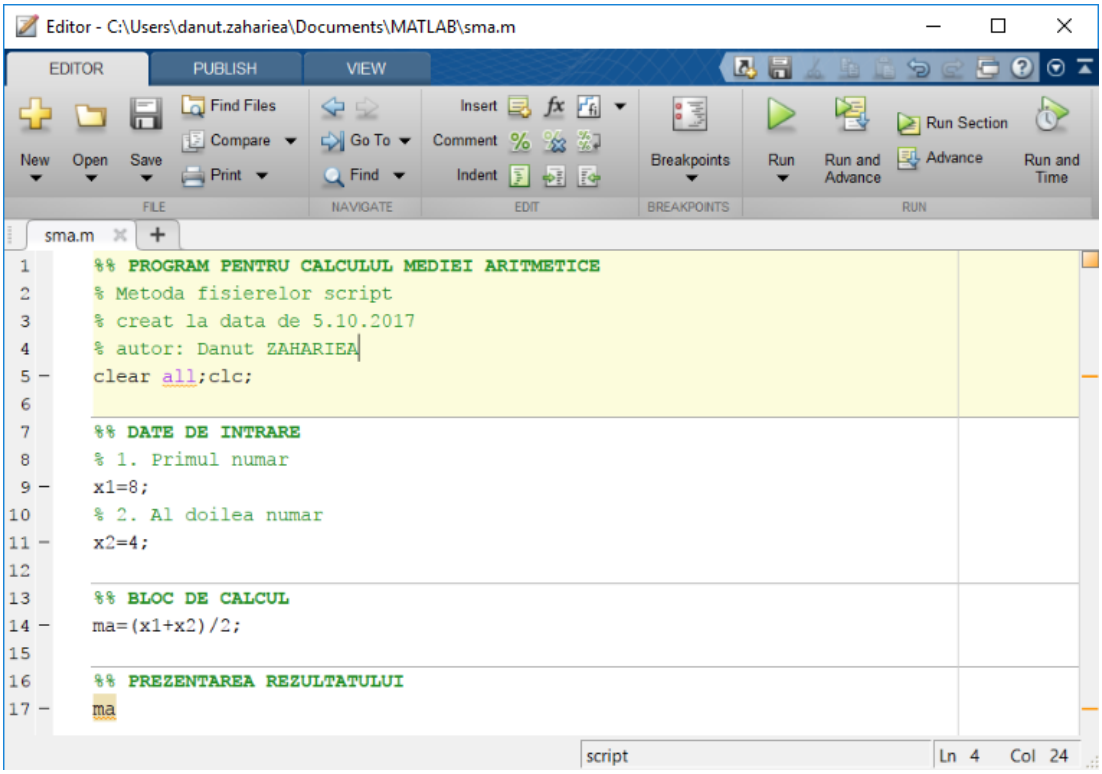
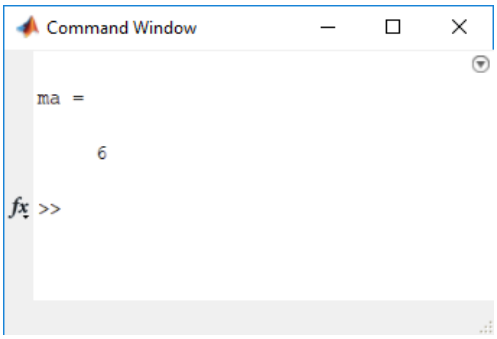


Figura 2.5. Rezolvarea prin metoda de lucru în linie de comandă.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea fișierelor de tip `script` este prezentată în figura 2.6, a) (fișierul `script` propriu-zis în care se definesc variabile de intrare și se introduce instrucțiunea de calcul) și figura 2.6, b) (rezultatul lansării în execuție a fișierului `script`).



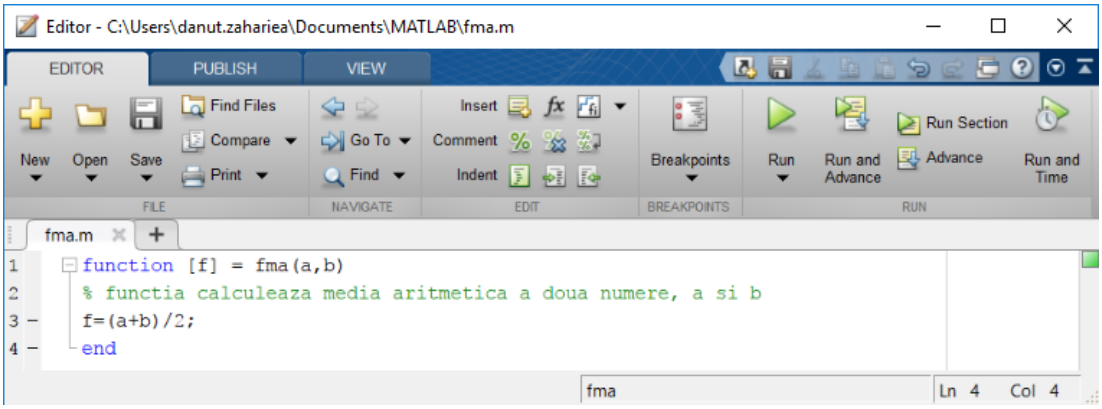
a) fișierul `script`



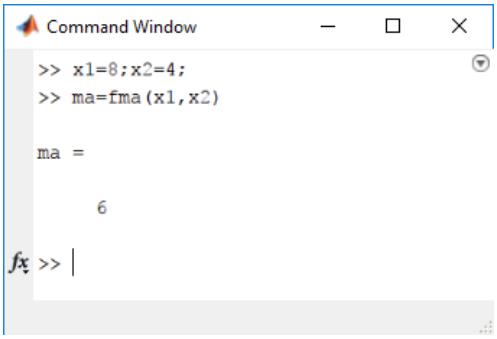
b) rezultatul obținut

Figura 2.6. Rezolvarea prin metoda de lucru bazată pe fișiere de tip `script`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor definite în fișiere de tip `function` este prezentată în figura 2.7, a) (fișierul `function` propriu-zis în care se introduce instrucțiunea de calcul) și figura 2.7, b) (apelarea funcției și rezultatul obținut).



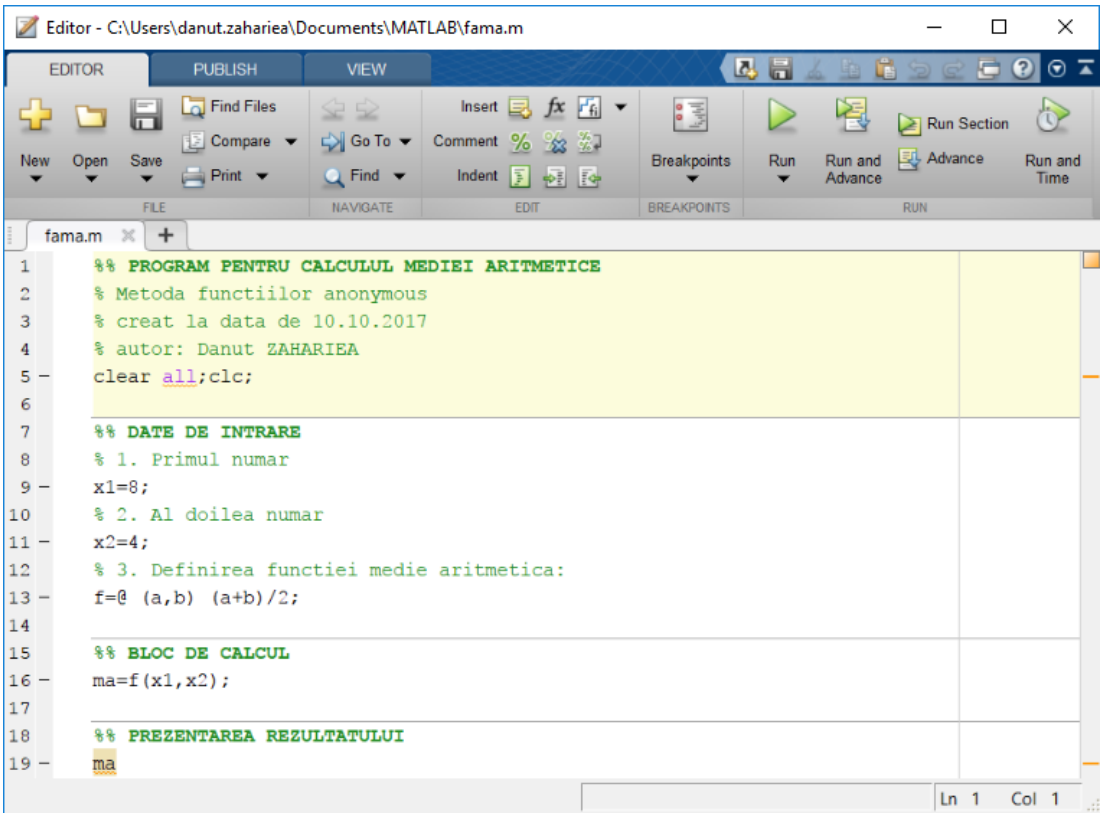
a) fișierul `function`



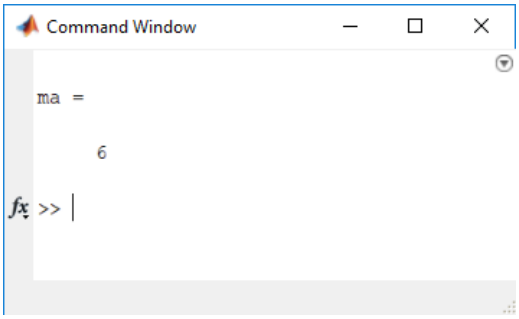
b) rezultatul obținut

Figura 2.7. Rezolvarea prin metoda de lucru bazată funcții definite în fișiere de tip `function`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor de tip `anonymous` este prezentată în figura 2.8, a) (fișierul `script` propriu-zis în care se definesc variabilele de intrare, se introduce instrucțiunea de calcul prin intermediul funcției `anonymous`, după care se apelează funcția) și figura 2.8, b) (rezultatul lansării în execuție a fișierului `script`).



a) fișierul `script`



b) rezultatul obținut

Figura 2.8. Rezolvarea prin metoda de lucru bazată funcții de tip `anonymous`.

Problema 2.2

Se consideră două numere: $x_1 = 18$ și x_2=34.

Să se calculeze media geometrică a celor două numere prin următoarele metode:

- Metoda de lucru în linie de comandă
- Metoda de lucru bazată pe folosirea fișierelor `script`
- Metoda de lucru bazată pe folosirea funcțiilor definite în fișiere
- Metoda de lucru bazată pe folosirea funcțiilor `anonymous`.

Relația pentru calculul mediei geometrice este:

$$m_g = \sqrt{x_1 \cdot x_2}$$

Rezolvarea problemei prin metoda de lucru în linie de comandă este prezentată în figura 2.9.

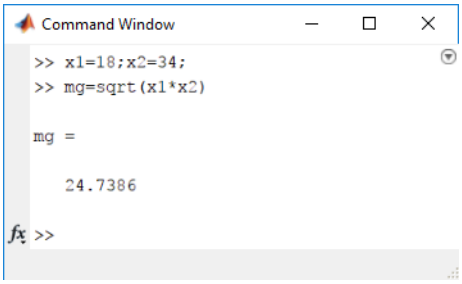
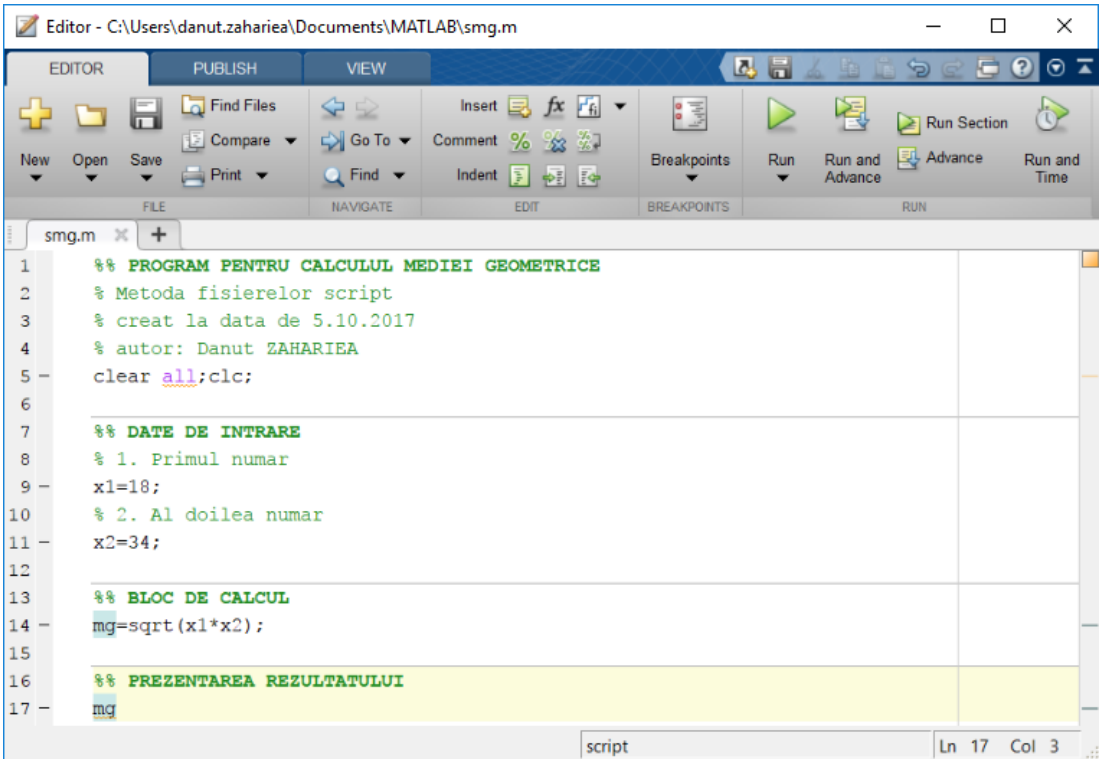
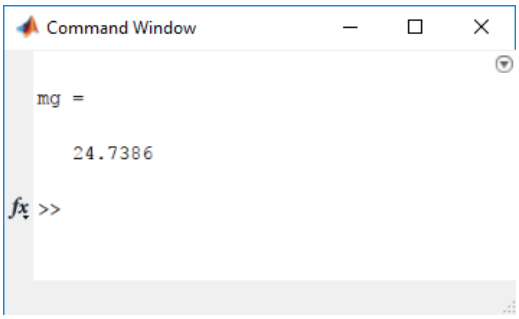


Figura 2.9. Rezolvarea prin metoda de lucru în linie de comandă.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea fișierelor de tip `script` este prezentată în figura 2.10, a) (fișierul `script` propriu-zis în care se definesc variabile de intrare și se introduce instrucțiunea de calcul) și figura 2.10, b) (rezultatul lansării în execuție a fișierului `script`).



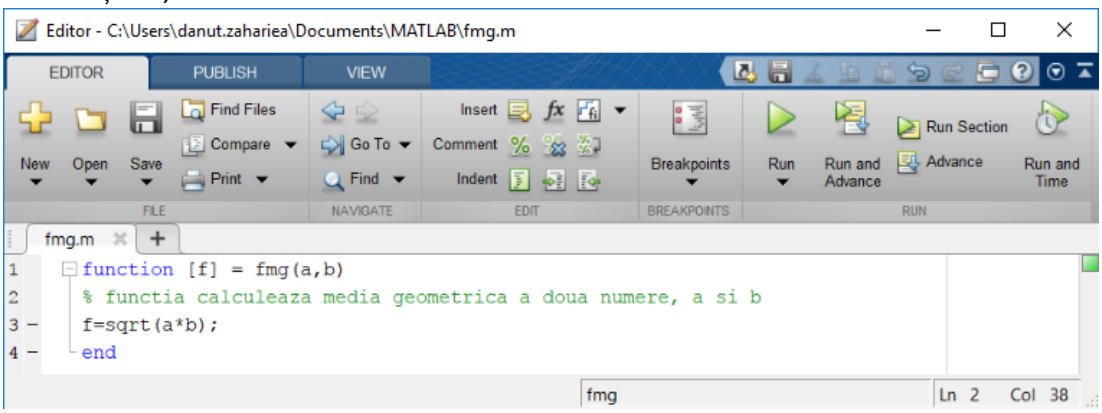
a) fișierul `script`



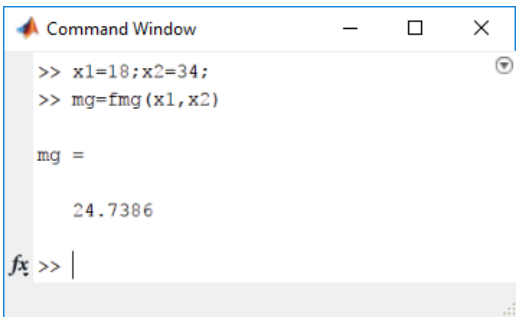
b) rezultatul obținut

Figura 2.10. Rezolvarea prin metoda de lucru bazată pe fișiere de tip `script`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor definite în fișiere de tip `function` este prezentată în figura 2.11, a) (fișierul `function` propriu-zis în care se introduce instrucțiunea de calcul) și figura 2.11, b) (apelarea funcției și rezultatul obținut).



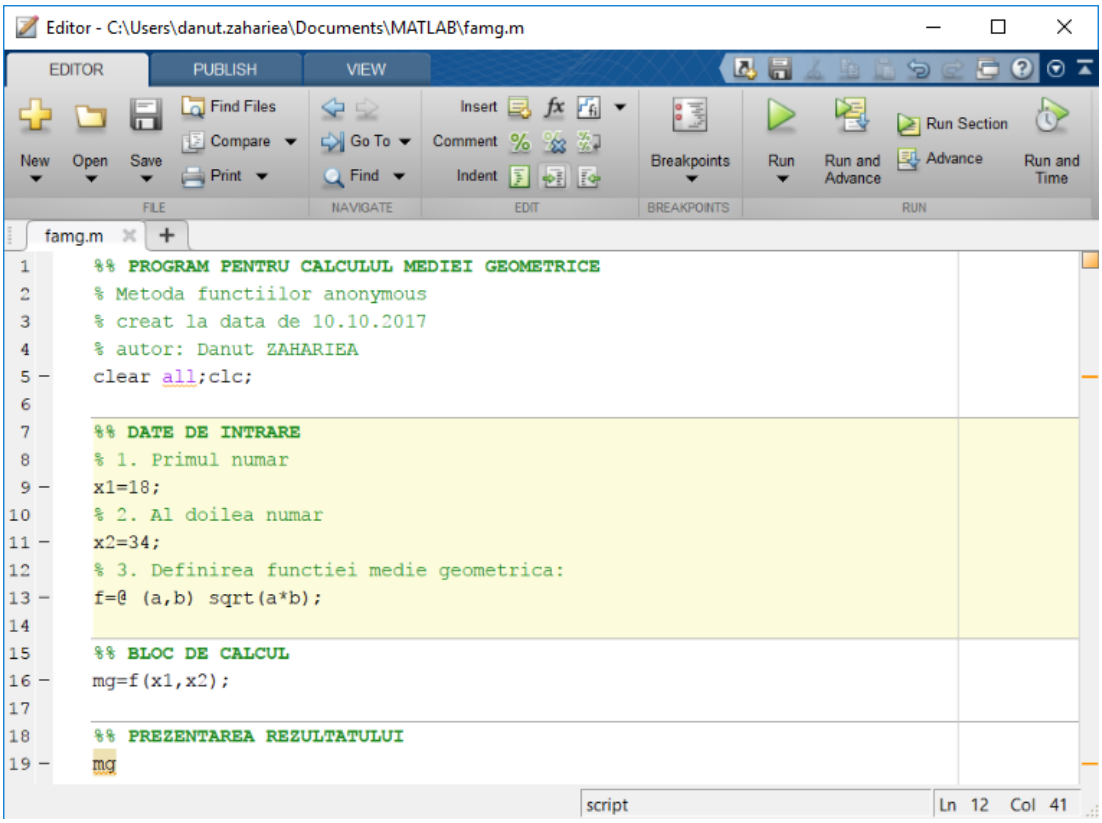
a) fișierul `function`



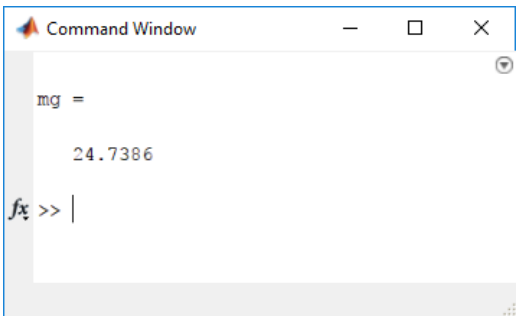
b) rezultatul obținut

Figura 2.11. Rezolvarea prin metoda de lucru bazată funcții definite în fișiere de tip `function`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor de tip `anonymous` este prezentată în figura 2.12, a) (fișierul `script` propriu-zis în care se definesc variabilele de intrare, se introduce instrucțiunea de calcul prin intermediul funcției `anonymous`, după care se apelează funcția) și figura 2.12, b) (rezultatul lansării în execuție a fișierului `script`).



a) fișierul `script`



b) rezultatul obținut

Figura 2.12. Rezolvarea prin metoda de lucru bazată funcții de tip `anonymous`.

Problema 2.3

Se consideră două numere: $x_1 = 72$ și x_2=52.

Să se calculeze media armonică a celor două numere prin următoarele metode:

- Metoda de lucru în linie de comandă
- Metoda de lucru bazată pe folosirea fișierelor `script`
- Metoda de lucru bazată pe folosirea funcțiilor definite în fișiere
- Metoda de lucru bazată pe folosirea funcțiilor `anonymous`.

Relația pentru calculul mediei armonice este:

$$m_r = \frac{2x_1x_2}{x_1 + x_2}$$

Rezolvarea problemei prin metoda de lucru în linie de comandă este prezentată în figura 2.13.

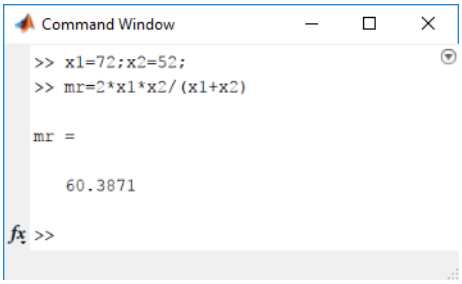
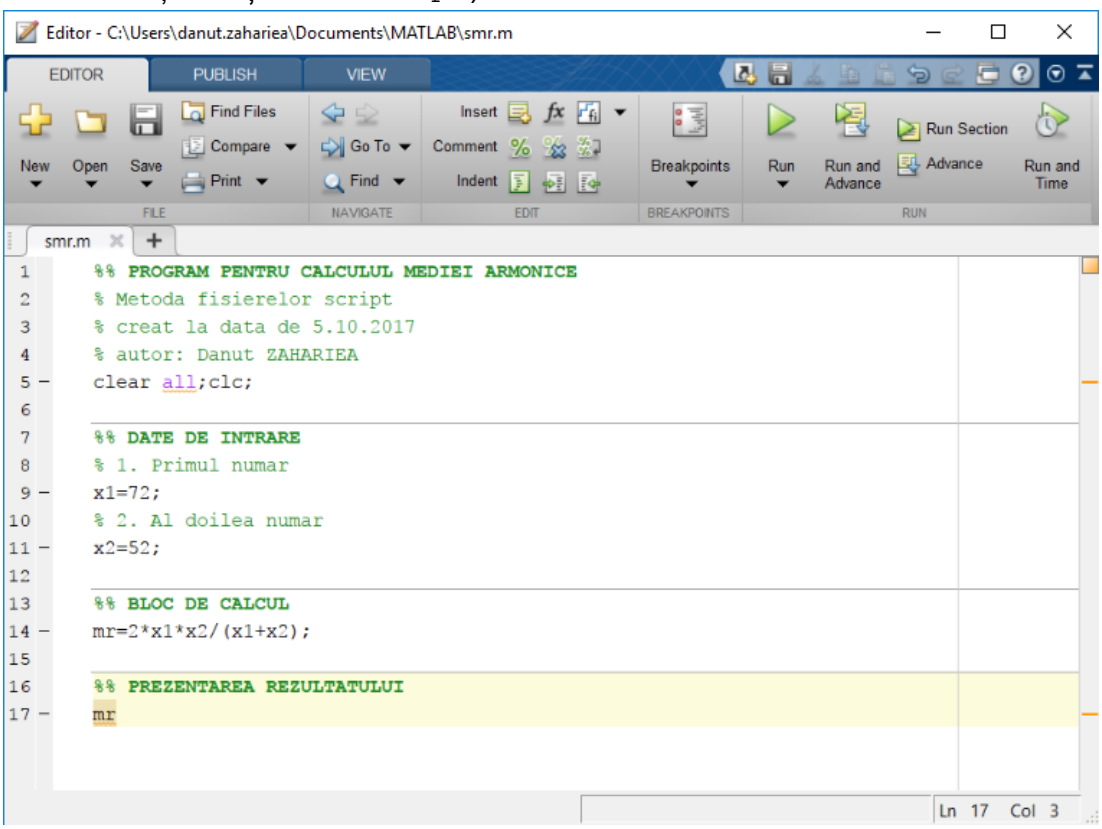
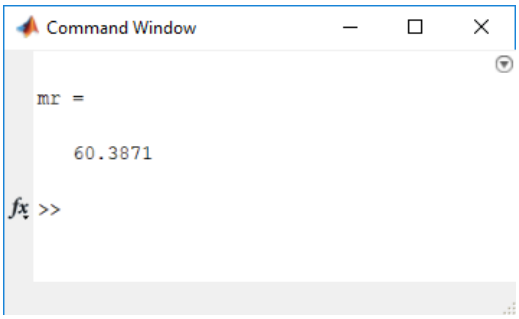


Figura 2.13. Rezolvarea prin metoda de lucru în linie de comandă.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea fișierelor de tip `script` este prezentată în figura 2.14, a) (fișierul `script` propriu-zis în care se definesc variabile de intrare și se introduce instrucțiunea de calcul) și figura 2.14, b) (rezultatul lansării în execuție a fișierului `script`).



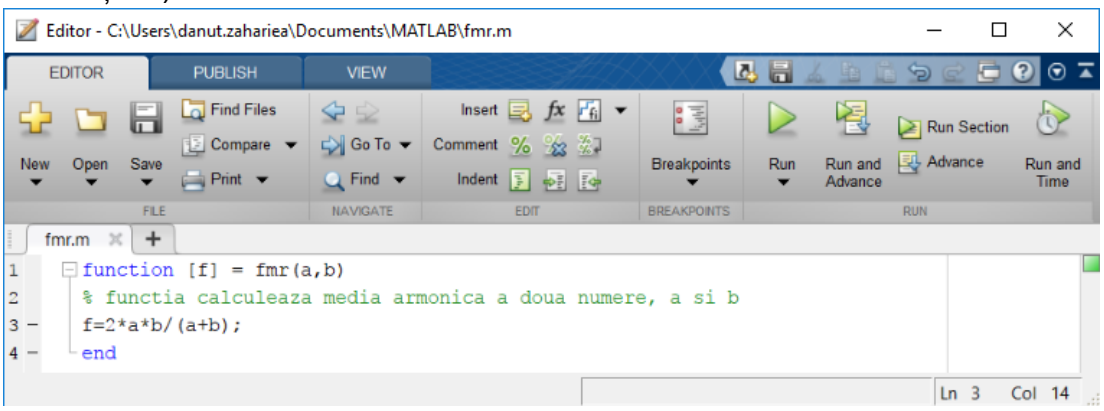
a) fișierul `script`



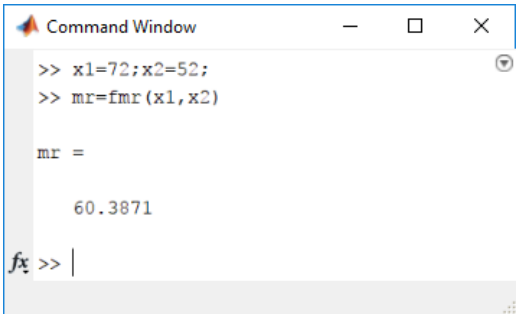
b) rezultatul obținut

Figura 2.14. Rezolvarea prin metoda de lucru bazată pe fișiere de tip `script`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor definite în fișiere de tip `function` este prezentată în figura 2.15, a) (fișierul `function` propriu-zis în care se introduce instrucțiunea de calcul) și figura 2.15, b) (apelarea funcției și rezultatul obținut).



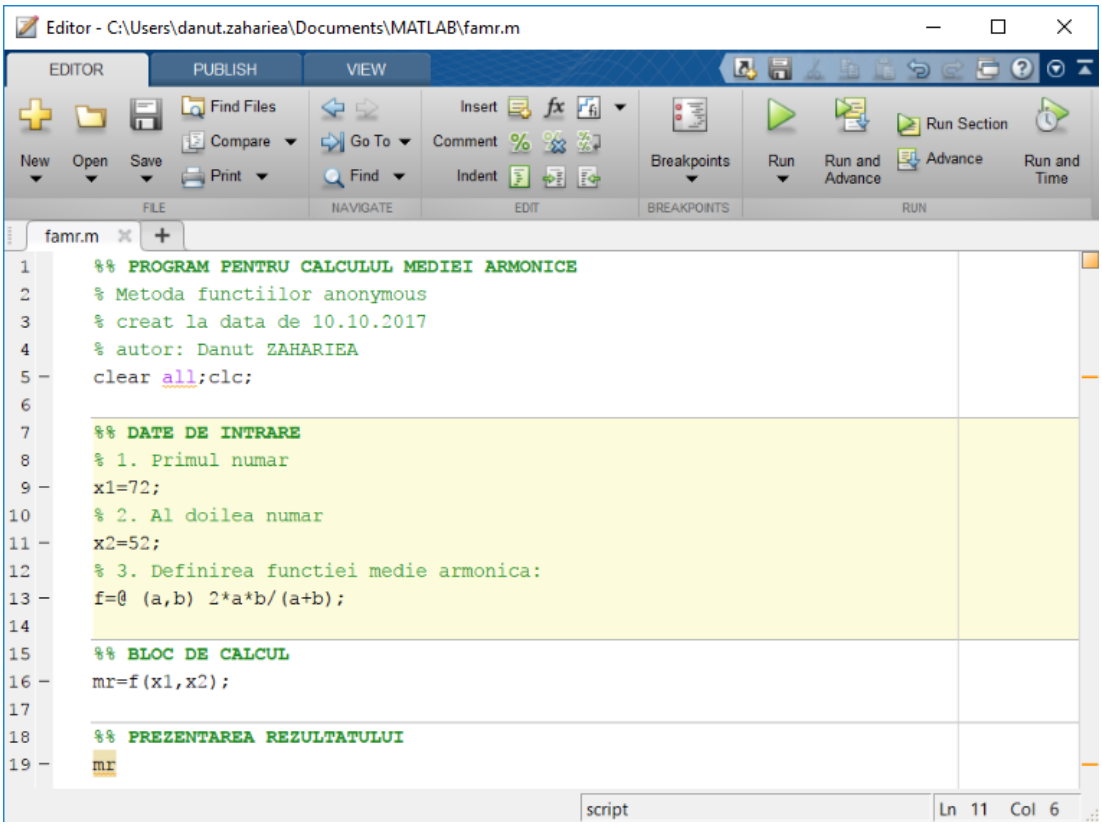
a) fișierul `function`



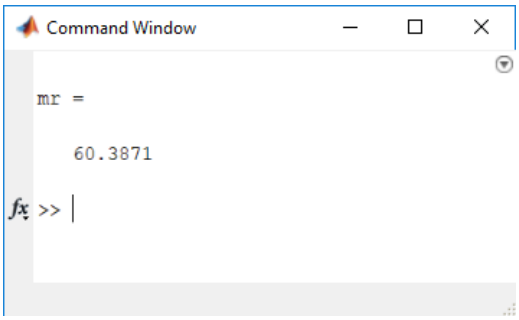
b) rezultatul obținut

Figura 2.15. Rezolvarea prin metoda de lucru bazată funcții definite în fișiere de tip `function`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor de tip `anonymous` este prezentată în figura 2.16, a) (fișierul `script` propriu-zis în care se definesc variabilele de intrare, se introduce instrucțiunea de calcul prin intermediul funcției `anonymous`, după care se apelează funcția) și figura 2.16, b) (rezultatul lansării în execuție a fișierului `script`).



a) fișierul `script`



b) rezultatul obținut

Figura 2.16. Rezolvarea prin metoda de lucru bazată funcții de tip `anonymous`.

Problema 2.4

Se consideră două numere: $x_1 = 7$ și x_2=2.

Să se calculeze media pătratică a celor două numere prin următoarele metode:

- Metoda de lucru în linie de comandă
- Metoda de lucru bazată pe folosirea fișierelor `script`
- Metoda de lucru bazată pe folosirea funcțiilor definite în fișiere
- Metoda de lucru bazată pe folosirea funcțiilor `anonymous`.

Relația pentru calculul mediei pătractice este:

$$m_p = \sqrt{\frac{x_1^2 + x_2^2}{2}}$$

Rezolvarea problemei prin metoda de lucru în linie de comandă este prezentată în figura 2.17.

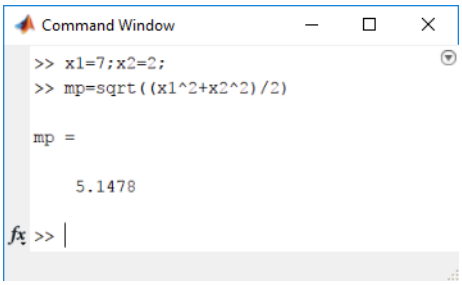
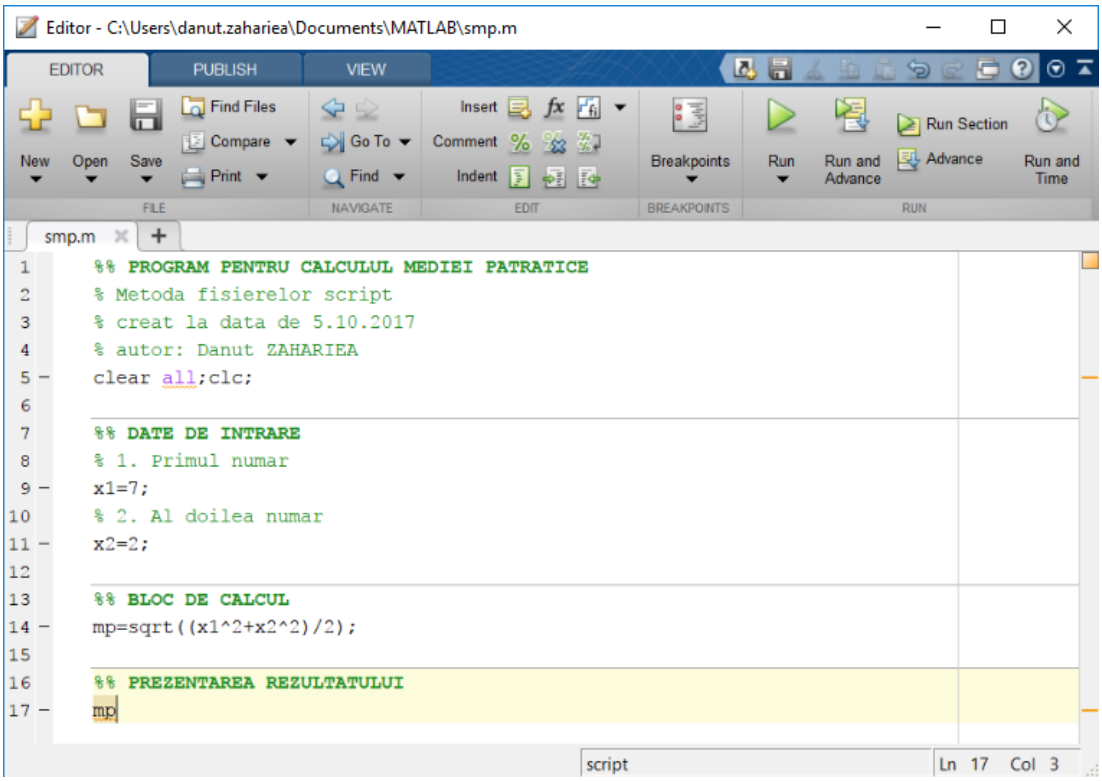
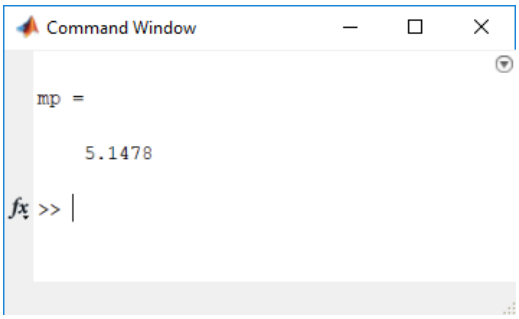


Figura 2.17. Rezolvarea prin metoda de lucru în linie de comandă.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea fișierelor de tip `script` este prezentată în figura 2.18, a) (fișierul `script` propriu-zis în care se definesc variabile de intrare și se introduce instrucțiunea de calcul) și figura 2.18, b) (rezultatul lansării în execuție a fișierului `script`).



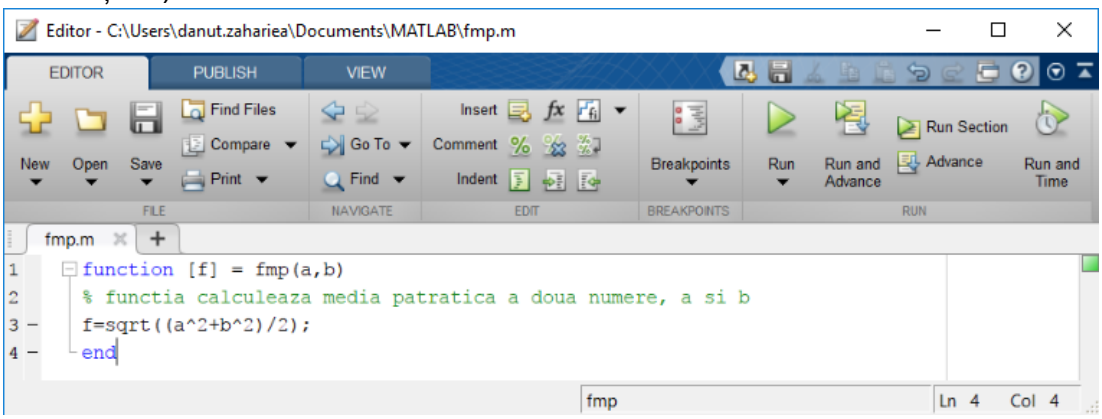
a) fișierul `script`



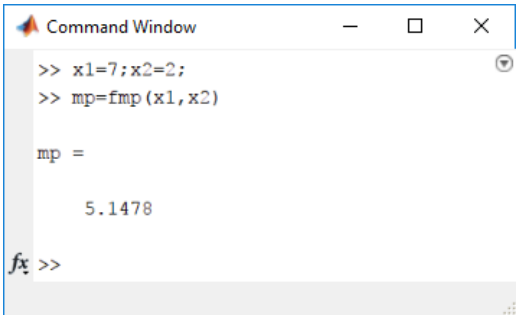
b) rezultatul obținut

Figura 2.18. Rezolvarea prin metoda de lucru bazată pe fișiere de tip `script`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor definite în fișiere de tip `function` este prezentată în figura 2.19, a) (fișierul `function` propriu-zis în care se introduce instrucțiunea de calcul) și figura 2.19, b) (apelarea funcției și rezultatul obținut).



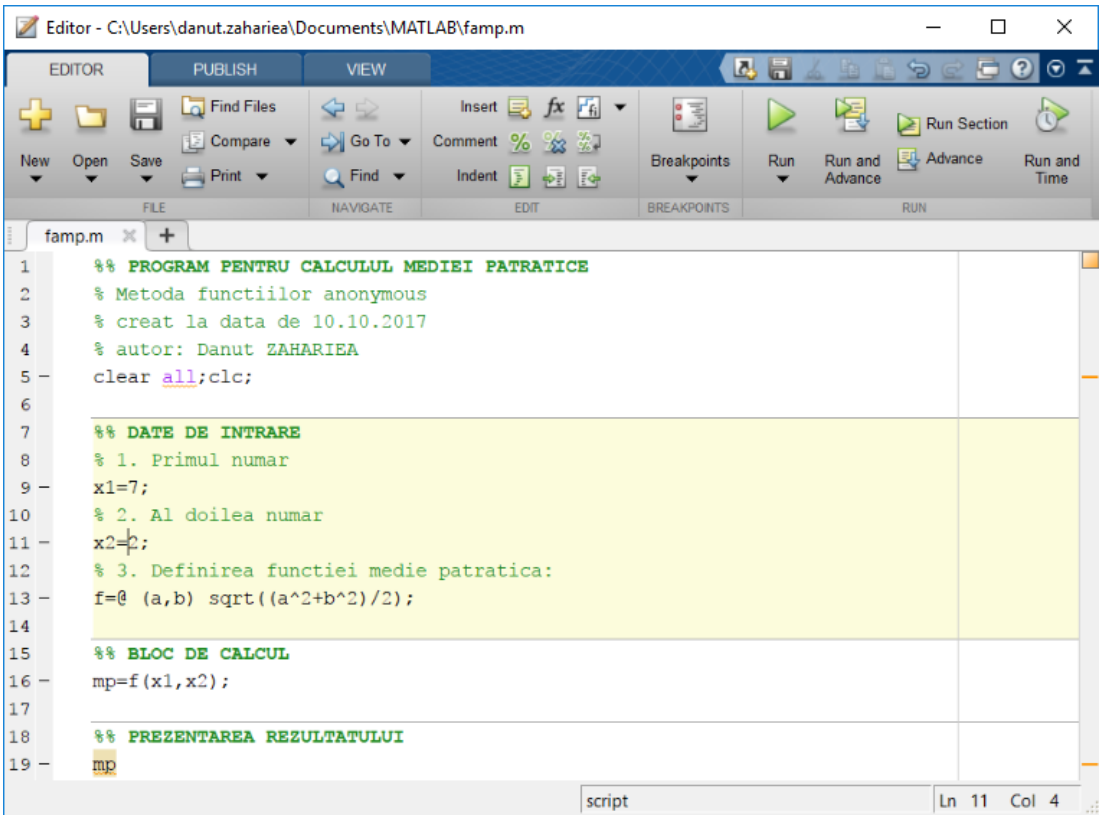
a) fișierul `function`



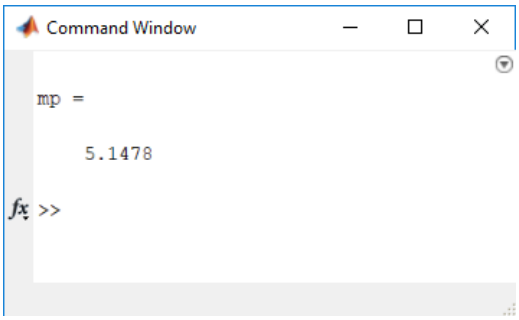
b) rezultatul obținut

Figura 2.19. Rezolvarea prin metoda de lucru bazată funcții definite în fișiere de tip `function`.

Rezolvarea problemei prin metoda de lucru bazată pe folosirea funcțiilor de tip `anonymous` este prezentată în figura 2.20, a) (fișierul `script` propriu-zis în care se definesc variabilele de intrare, se introduce instrucțiunea de calcul prin intermediul funcției `anonymous`, după care se apelează funcția) și figura 2.20, b) (rezultatul lansării în execuție a fișierului `script`).



a) fișierul `script`



b) rezultatul obținut

Figura 2.20. Rezolvarea prin metoda de lucru bazată funcții de tip `anonymous`.

CAPITOLUL 3
ALGORITMI ȘI SCHEME LOGICE

3.1. DEFINIȚII. PROPRIETĂȚI

Algoritmul reprezintă o succesiune finită de etape prin care, pornind de la datele de intrare care definesc o anumită problemă, se ajunge la o serie de date de ieșire, care reprezintă soluția problemei respective, figura 3.1.

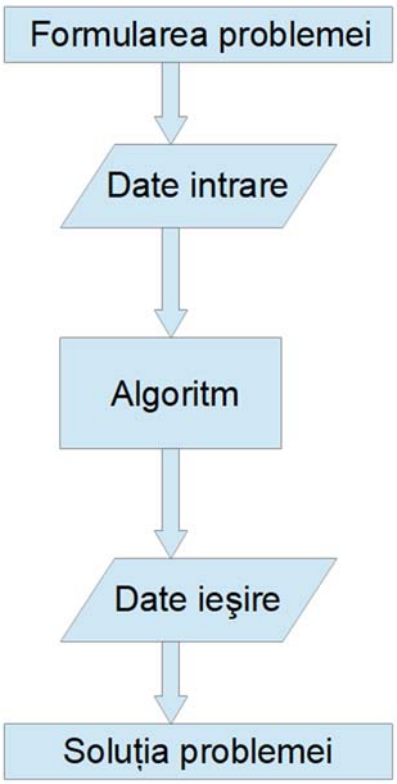


Figura 3.1. Algoritmul și calea de rezolvare a unei probleme.

Datele de intrare reprezintă toate informațiile necesare pentru rezolvarea problemei respective. Datele de intrare rezultă dintr-o bună înțelegere a problemei de rezolvat. Reducerea numărului sau aproximarea unor date de intrare poate conduce la reducerea complexității algoritmului, dar și la creșterea gradului de aproximarea a soluției astfel obținută.

Datele de ieșire reprezintă datele finale care rezultă în urma aplicării algoritmului de calcul asupra datelor de intrare. Algoritmul de calcul poate să furnizeze și o serie de rezultate intermediare, care au relevanță doar în contextul etapelor intermediare ale algoritmului. Utilizatorul, pe baza problemei de rezolvat, va decide care din rezultatele intermediare reprezintă și datele de ieșire relevante pentru soluția problemei.

Principale proprietăți ale algoritmilor sunt:

- **Determinarea**. Algoritmul trebuie să fie astfel conceput încât toate etapele ca și ordinea lor de executare să fie clare, concise și fără ambiguități.
- **Generalitatea**. Algoritmul trebuie să permită rezolvarea unei familii de probleme și nu a unei probleme particulare.
- **Finitudinea**. Algoritmul poate să conțină un număr oricât de mare de etape, dar un număr finit.

Principalele obiecte care intră în structura algoritmilor sunt:

- **Constantele** - reprezintă date (numerice, alfanumerice, logice) care nu se modifică pe parcursul executării unui algoritm.
- **Variabilele** - reprezintă date care se modifică pe parcursul executării unui algoritm.
- **Operatorii** - reprezintă operațiile care se execută asupra constantelor și variabilelor unui algoritm. Principalele tipuri de operatori sunt:
  - Operatori aritmetici: adunare +, scădere -, înmulțire *, împărțire /.
  - Operatori relaționali: mai mare >, mai mic <, mai mare sau egal >=, mai mic sau egal <=, egal =, diferit ~=.
  - Operatori logici: conjuncție logică AND, disjuncție logică OR, negație logică NOT.
- **Expresiile** - reprezintă ansamblul constantelor, variabilelor și al operatorilor dintre acestea. Expresiile pot fi: aritmetice, relaționale și logice.

3.2. METODE DE REPREZENTARE A ALGORITMILOR

Reprezentarea algoritmilor este independentă de un anumit limbaj de programare și se realizează prin două metode: metoda schemelor logice și metoda pseudocod.

Metoda schemelor logice – reprezintă un limbaj în care diferitele elemente ale unui algoritm sunt reprezentate cu ajutorul unor simboluri grafice.

Principalele simboluri grafice specifice realizării schemelor logice sunt prezentate în figura 3.2.

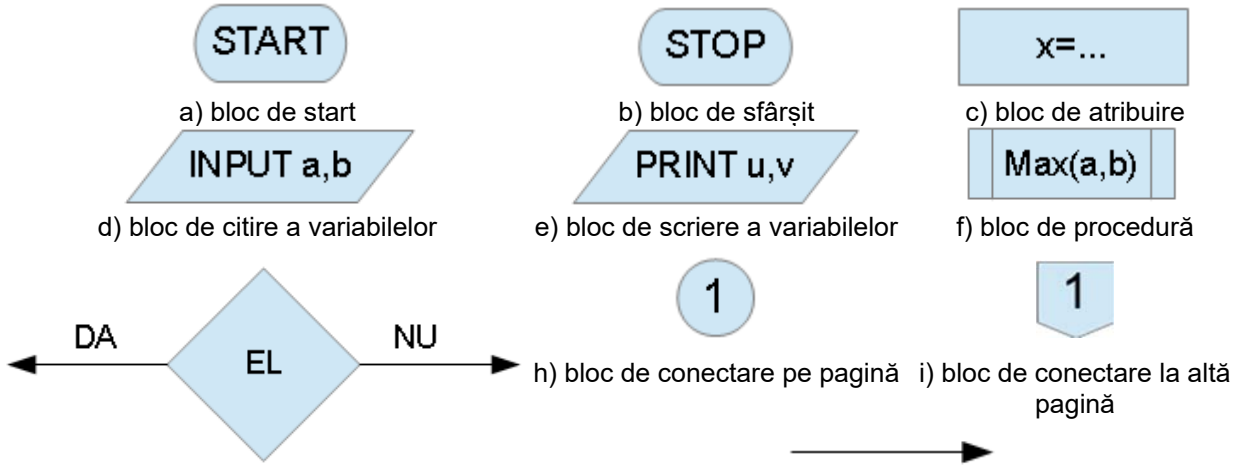


Figura 3.2. Simboluri grafice pentru metoda schemelor logice.

Observații

- Blocurile de start (START) și de sfârșit (STOP) (figura 3.2,a și b), denumite și blocuri de tip terminal, au doar rolul de a defini punctele terminale (de început și de sfârșit) ale schemei logice.
- Blocul de atribuire (figura 3.2, c) permite definirea unor variabile prin intermediul expresiilor.
- Blocurile de citire (INPUT) și de scriere (PRINT) (figura 3.2, d și e), a variabilelor sunt responsabile cu introducerea datelor de intrare (a, b), respectiv cu prezentarea datelor de ieșire (u, v) obținute după executarea schemei logice.

- Blocul de procedură (figura 3.2, f) permite compactarea unei porțiuni a algoritmului într-un singur bloc, asigurând astfel modularizarea schemelor logice. În bloc se specifică numele simbolic al procedurii, variabilele de intrare și cele de ieșire.
- Blocul condițional (figura 3.2, g), introduce în schema logică un punct de ramificare pe baza unei expresii logice. În funcție de valoarea de adevăr a acestei expresii logice, blocul condițional realizează o ieșire alternativă, pe una sau cealaltă dintre căile de ieșire ale blocului.
- Blocurile de conectare pe pagină (figura 3.2, h), respectiv la altă pagină (figura 3.2, i) permit gruparea a două intrări într-o singură ieșire, respectiv continuarea schemei logice pe aceeași pagină sau pe altă pagină. Blocurile de conectare sunt identificate prin numere și ajută la modularizarea schemei logice.
- Săgeata de conectare (figura 3.2, j), asigură conexiunea dintre diferitele blocuri ale unei scheme logice. Săgeata de conectare asigură transferul informației într-un singur sens (specificat prin orientarea săgeții de la capătul liniei).

Metode de tip pseudocod – reprezintă un limbaj în care diferitele elemente ale unui algoritm sunt reprezentate cu ajutorul unor cuvinte cheie. Cuvintele cheie au în corespondență instrucțiuni specifice în diferite limbaje de programare.
Principalele cuvinte cheie (în limba română) ale metodei de tip pseudocod sunt prezentate în figura 3.3.

<table>
<tr><td>start
stop</td><td>citește a,b
scrie u,v</td><td>x=expresie</td></tr>
<tr><td>a) comenzile de start și de sfârșit</td><td>b) comenzile de citire și de scriere a variabilelor</td><td>c) comanda de atribuire</td></tr>
<tr><td colspan="2">cât timp expresie logică=TRUE execută
instrucțiune 1
instrucțiune 2
sfârșit cât timp</td><td>repetă
instrucțiune 1
instrucțiune 2
până când expresie logică=TRUE</td></tr>
<tr><td colspan="2">d) structura cât timp</td><td>e) structura repetă până când</td></tr>
<tr><td colspan="2">dacă expresie logică=TRUE atunci
instrucțiune 1
altfel
instrucțiune 2
sfârșit dacă</td><td>dacă expresie logică=TRUE atunci
instrucțiune 1
instrucțiune 2
sfârșit dacă</td></tr>
<tr><td colspan="2">f) structură condițională 1</td><td>g) structură condițională 2</td></tr>
<tr><td colspan="3">pentru variabilă=valoare inițială, valoare finală execută
instrucțiune 1
instrucțiune 2
sfârșit pentru</td></tr>
<tr><td colspan="3">h) structura pentru</td></tr>
</table>

Figura 3.3. Cuvinte cheie specifice metoda de tip pseudocod.

Observații

- Cuvintele cheie **start** și **stop** corespund începutului și sfârșitului algoritmului.
- Comanda de atribuire nu conține nici un cuvânt cheie.
- Cuvintele cheie **citește** și **scrie** corespund comenzilor de introducere a datelor de intrare (a, b), respectiv de prezentare a datelor de ieșire (u, v).
- Structura **cât timp** corespunde unui ciclu repetitiv cu test inițial.
- Structura **repetă până când** corespunde unui ciclu repetitiv cu test final.
- Structura **pentru** corespunde unui ciclu repetitiv cu număr cunoscut de pași.
- În cazul structurii condiționale 1, instrucțiunea 1 se execută doar în cazul în care expresia logică este adevărată. În caz contrar, se va executa instrucțiunea 2.
- În cazul structurii condiționale 2, cele două instrucțiuni 1 și 2 se vor executa doar dacă este îndeplinită expresia logică, în caz contrar, structura **dacă** nu se ia în considerare.

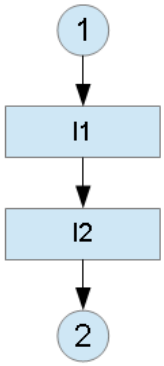
3.3. STRUCTURILE DE CONTROL ȘI SCHEMELE LOR LOGICE

Teorema programării structurate. Orice algoritm cu un singur punct de început și un singur punct de sfârșit poate fi descris cu ajutorul doar a trei tipuri de structuri de control:

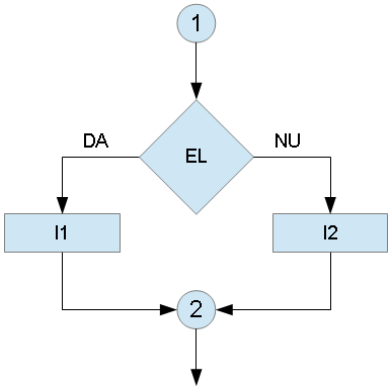
- **structura secvențială**. Instrucțiunile se execută una după alta, exact în ordinea în care apar în program. Schema logică a unei structuri secvențiale cu 2 instrucțiuni `I1` și `I2` este prezentată în figura 3.4, a).
- **structura alternativă (condițională)**. Instrucțiunile se execută în mod alternativ, după cum o anumită expresie logică este adevărată sau falsă. Structurile alternative pot fi: cu două ramuri (figura 3.4, b) sau cu o singură ramură (figura 3.4, c). Valoarea de adevăr a expresiei logice `EL` poate fi adevărat sau fals și în funcție de această urmează execuția alternativă a instrucțiunilor.
- **structura iterativă (repetitivă)**. Instrucțiunile se execută în mod iterativ, în funcție de valoarea curentă a unui contor `i`, care la inițializare are valoarea `i=imin`. Ieșirea din bucla iterativă se realizează după un număr necunoscut de iterații printr-un test inițial (figura 3.4, d), sau final (figura 3.4, e), respectiv după un număr cunoscut de iterații, caz în care se cunoaște valoarea maximă a contorului (figura 3.4, f).

Observații

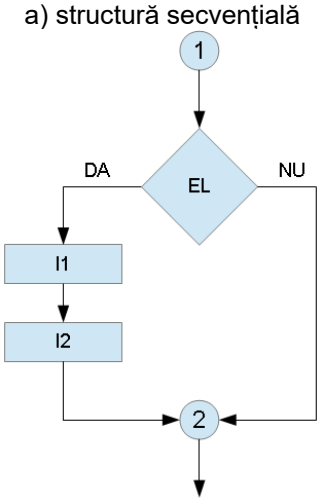
- În cazul structurii secvențiale (figura 3.4, a), cele două instrucțiuni `I1` și `I2` se execută o singură dată, în ordinea specificată în schema logică: după terminarea instrucțiuni `I1` se execută instrucțiunea `I2`. Dacă instrucțiunile sunt dependente între ele, instrucțiunile trebuie să fie executate într-o anumită ordine. Dacă, de exemplu instrucțiunea `I2` depinde de instrucțiunea `I1`, atunci ordinea de execuție a celor două instrucțiuni trebuie să fie în mod obligatoriu `I1`→`I2`.
- În cazul structurii alternative cu două ramuri (figura 3.4, b), dacă expresia logică `EL` este adevărată se execută instrucțiunea `I1`, iar dacă expresia logică `EL` este falsă atunci se va executa instrucțiunea `I2`.



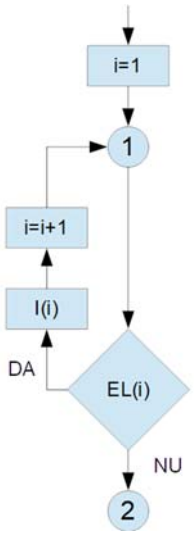
a) structură secvențială



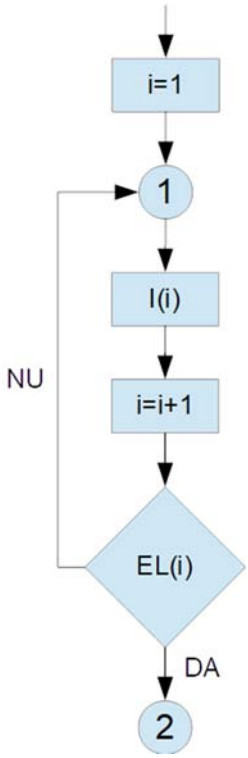
b) structură alternativă cu două ramuri



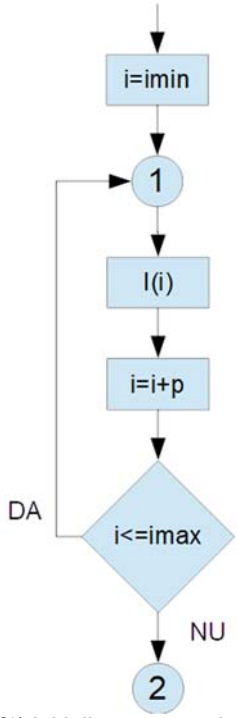
c) structură alternativă cu o ramură



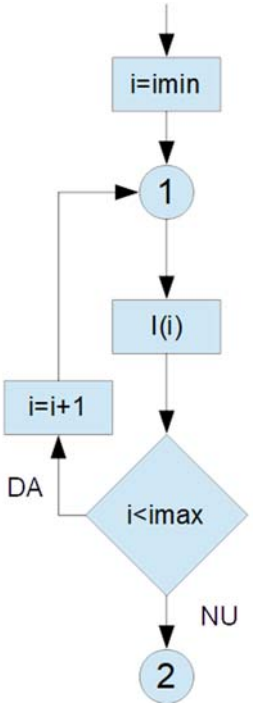
d) structură iterativă cu număr necunoscut de iterații și test inițial



e) structură iterativă cu număr necunoscut de iterații și test final



f1) inițializare-execuție instrucțiuni-incrementare-testare contor



f2) inițializare-execuție instrucțiuni-testare contor-incrementare

f) structură iterativă cu număr cunoscut de iterații

Figura 3.4. Structuri de control secvențiale, alternative și iterative.

- În cazul structurii alternative cu o singură ramură (figura 3.4, c), dacă expresia logică `EL` este adevărată se execută în mod secvențial cele două instrucțiuni `I1` și `I2`, iar dacă expresia logică `EL` este falsă atunci se părăsește structura alternativă fără a se executa nici o instrucțiune.
- În cazul structurii iterative cu număr necunoscut de iterații și test inițial (figura 3.4, d), dacă expresia logică `EL(i)` este falsă atunci se părăsește structura iterativă fără a se executa nici măcar o singură dată instrucțiunea `I(i)`. Atât timp însă, cât expresia logică `EL(i)` va fi adevărată, se va executa instrucțiunea `I(i)`.
- În cazul structurii iterative cu număr necunoscut de iterații și test final (figura 3.4, e), instrucțiunea `I(i)` se va executa cel puțin o singură dacă, deoarece abia după execuția instrucțiunii `I(i)` urmează evaluarea expresiei logice `EL(i)`. Dacă aceasta este adevărată se părăsește structura iterativă, în caz contrar se va repeta executarea instrucțiunii `I(i)`.
- În cazul structurilor iterative cu număr necunoscut de iterații și test inițial (figura 3.4, d), respectiv cu test final (figura 3.4, e), numărul de iterații este exprimat prin intermediul unui contor notat cu `i`, care la începutul structurii iterative se inițializează `i=1` și apoi, în interiorul structurii iterative, se incrementează `i=i+1`.
- În cazul structurii iterative din figura 3.4, f1), se definește un contor `i` care, inițial primește valoarea inițială (`i=imin`, faza de inițializare a contorului). Apoi se execută instrucțiunile `I(i)` (faza de execuție a corpului structurii iterative), după care urmează faza de incrementare a valorii contorului (`i=i+p`) cu o valoare caracteristică denumită pas (`p`). În acest caz pasul poate avea orice valoare (`p=1`, sau `p≠1`). Urmează apoi condiția de verificare a valori contorului în funcție de valoarea finală impusă (`i<=imax`). În cazul în care valoarea curentă a contorului este mai mică sau egală cu valoarea sa maximă (`i<=imax`), se rămâne în corpul structurii iterative. În momentul în care valoarea curentă a contorului este mai mare decât valoarea sa maximă (`i>imax`) se părăsește corpul structurii iterative. Ordinea de execuție a fazelor structurii iterative este: inițializare contor-execuție instrucțiuni-incrementare contor-testare contor.
- În cazul structurii iterative din figura 3.4, f2), se definește un contor `i` care, inițial primește valoarea inițială (`i=imin`, faza de inițializare a contorului). Apoi se execută instrucțiunile `I(i)` (faza de execuție a corpului structurii iterative), după care urmează condiția de verificare a valori contorului în funcție de valoarea finală impusă (`i<imax`). În cazul în care valoarea curentă a contorului este mai mică decât valoarea sa maximă `(i<imax`), se rămâne în corpul structurii iterative. În momentul în care valoarea curentă a contorului este egală cu valoarea sa maximă (`i=imax`) se părăsește corpul structurii iterative. Urmează apoi faza de incrementare a contorului care presupune incrementarea valorii contorului (`i=i+p`, `p=1`) cu o valoare caracteristică denumită pas (`p`). Această variantă se utilizează doar pentru `p=1`. Ordinea de execuție a fazelor structurii iterative este: inițializare contor-execuție instrucțiuni-testare contor-incrementare contor.
- În cazul structurilor iterative, datorită erorilor logice de stabilire a condițiilor de verificare este posibilă apariția unor bucle infinite.

3.4. INSTRUCȚIUNI DE CONTROL MATLAB

Principalele instrucțiuni MATLAB pentru realizarea structurilor de control sunt: instrucțiunea condițională `if`; instrucțiunea iterativă `for` și instrucțiunea iterativă `while`. Cu ajutorul acestor instrucțiuni se pot realiza toate structurile de control secvențiale, alternative și iterative.

3.4.1. Instrucțiunea condițională `if`

Instrucțiunea condițională `if` permite realizarea unei structuri alternative pe baza valorii de adevăr a unei condiții logice. Definirea condițiilor logice se face cu ajutorul operatorilor relaționali și logici specifici, reprezentați în tabelul 3.1.

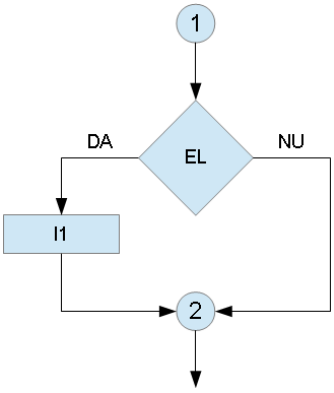
Tabel 3.1. Operatori relaționali și logici.

Operatori relaționali	
Semnificația	Simbol MATLAB
Mai mic	<
Mai mare	>
Mai mic sau egal	<=
Mai mare sau egal	>=
Identic	==
Diferit	~=

Operatori logici	
Semnificația	Simbol MATLAB
AND	&
OR	|
NOT	~

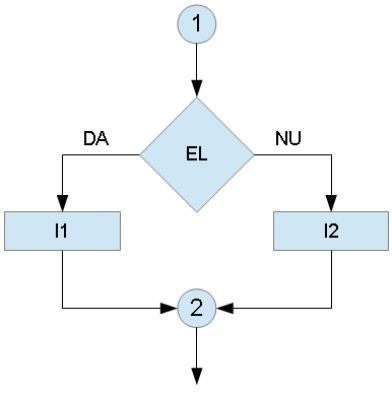
Principalele implementări ale instrucțiunii condiționale `if` sunt: `if`; `if-else`; `if-elseif`; `if-elseif-else` (figura 3.5).

```
if expresie logică
     instrucțiune 1;
end
```



a) varianta `if`

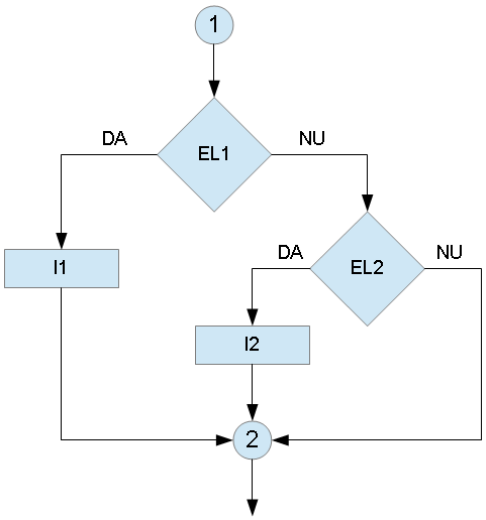
```
if expresie logică
     instrucțiune 1;
else
     instrucțiune 2;
end
```



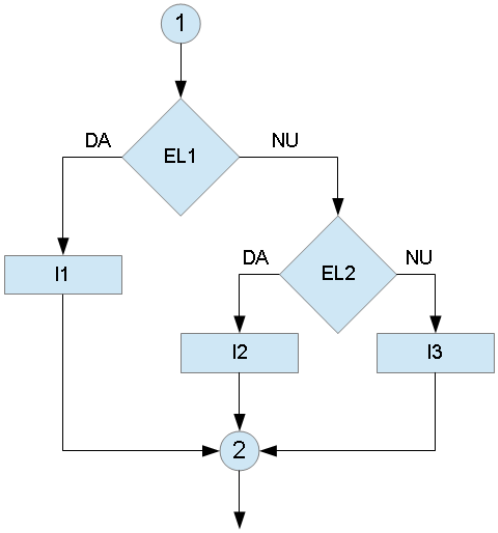
b) varianta `if-else`

```
if expresie logica 1
     instrucțiune 1;
elseif expresie logică 2
     instrucțiune 2;
end
```

```
if expresie logica 1
     instrucțiune 1;
elseif expresie logică 2
     instrucțiune 2;
else
     instrucțiune 3;
end
```



c) varianta `if-elseif`



d) varianta `if-elseif-else`

Figura 3.5. Instrucțiunea condițională `if`.

Observații

- În cazul variantei `if` (figura 3.5, a), dacă expresia logică `EL` este adevărată, atunci se execută instrucțiunea `I1`, în caz contrar se părăsește structura condițională.
- În cazul variantei `if-else` (figura 3.5, b), dacă expresia logică `EL` este adevărată, se execută instrucțiunea 1, în caz contrar se execută instrucțiunea `I2`.
- În cazul variantei `if-elseif` (figura 3.5, c), dacă expresia logică `EL1` este adevărată se execută instrucțiunea `I1`. Dacă expresia logică `EL1` este falsă, atunci se testează suplimentar expresia logică `EL2`: dacă expresia logică `EL2` este adevărată se execută instrucțiunea `I2`, în caz contrar se părăsește structura condițională. Dacă ambele expresii logice `EL1` și `EL2` sunt false atunci se părăsește structura condițională fără a se executa nici o instrucțiune.
- În cazul variantei `if-elseif-else` (figura 3.5, d), dacă expresia logică `EL1` este adevărată se execută instrucțiunea `I1`. Dacă expresia logică `EL1` este falsă, atunci se testează suplimentar expresia logică `EL2`: dacă expresia logică `EL2` este adevărată se execută instrucțiunea `I2`, în caz contrar se execută instrucțiunea `I3`. Indiferent de valoarea de adevăr a celor două expresii logice `EL1` și `EL2`, cel puțin o instrucțiune din cadrul structurii condiționale va fi executată.

3.4.2. Instrucțiunea iterativă `for`

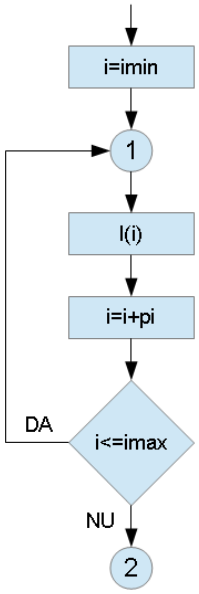
Instrucțiunea `for` se utilizează pentru repetarea unor instrucțiuni de un număr n_i finit, cunoscut de ori. În acest scop se definește o variabilă specială, denumită contor și notată de exemplu cu litera i, care va lua anumite valori între valoarea minimă i_{min} și valoarea maximă i_{max}. În cazul în care pasul contorului (diferența dintre oricare două valori consecutive) este cunoscut și notat cu p_i, atunci numărul n_i de ori pentru care se va realiza repetarea executării instrucțiunilor se poate determina cu relația:

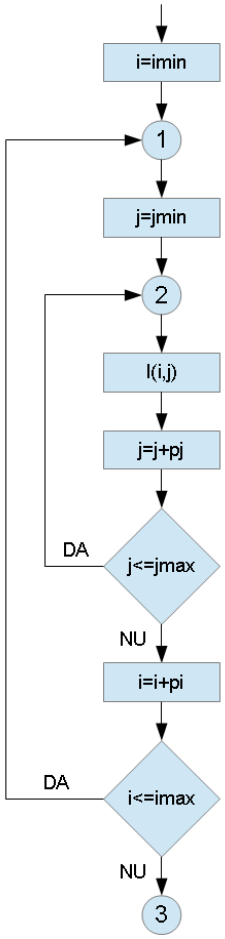
$$n_i = \left[\frac{i_{max} - i_{min}}{p_i}\right] + 1$$

Principalele implementări ale instrucțiunii `for` sunt prezentate în figura 3.6.

```
for i=imin:pi:imax
     instrucțiuni I(i);
end
```

```
for i=imin:pi:imax
     for j=jmin:pj:jmax
          instrucțiuni I(i,j);
     end
end
```





a) instrucțiunea `for` simplă

b) instrucțiunea `for` dublă

Figura 3.6. Instrucțiunea iterativă `for`.

Observații

- În cazul instrucțiunii `for` simple (figura 3.6, a), după execuția structurii iterative se obține un vector având `n` elemente. Principalele faze ale algoritmului sunt: inițializarea contorului `i` (`i=imin`), execuția instrucțiunilor `I(i)` dependente de valoarea contorului `i`, incrementarea contorului `i` cu valoarea pasului `pi` (`i=i+pi`) și testarea contorului `i` (`i<=imax`).
- În cazul instrucțiunii for duble (figura 3.6, b), după execuția structurii iterative se obține o matrice cu n_i linii și n_j coloane. Principalele faze ale algoritmului sunt: inițializarea contorului `i` (`i=imin`), inițializarea contorului `j` (`j=jmin`), execuția instrucțiunilor `I(i,j)` dependente de valorile contoarelor `i` și `j`, incrementarea contorului `j` cu valoarea pasului `pj` (`j=j+pj`), testarea contorului `j` (`j<=jmax`), incrementarea contorului `i` cu valoarea pasului `pi` (`i=i+pi`) și testarea contorului `i` (`i<=imax`).

3.4.3. Instrucțiunea iterativă `while`

Instrucțiunea iterativă `while` se utilizează pentru realizarea unei structuri iterative cu test inițial.

Implementarea instrucțiunii `while` este prezentată în figura 3.7.

```
i=1;
while EL(i)
     instrucțiuni I(i);
     i=i+1;
end
```

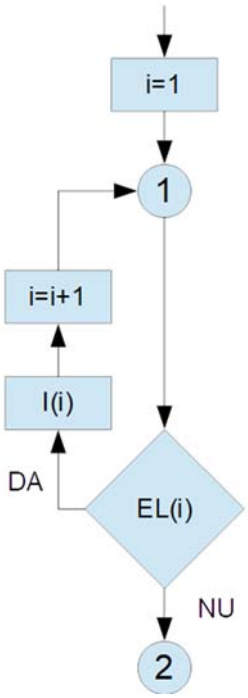


Figura 3.7. Instrucțiunea iterativă `while`.

Observații

- Structura iterativă cu test inițial presupune definirea și inițializarea unui contor, `i=1`.
- Pentru prima valoare a contorului (`i=1`) se evaluează expresia logică `EL(i)`.
- Dacă expresia logică `EL(i)` este falsă atunci se părăsește structura iterativă fără a se executa nici măcar o singură dată instrucțiunile `I(i)`.
- În caz contrar se execută instrucțiunile din corpul structurii `I(i)`, se incrementează contorul (`i=i+1`), după care urmează din nou evaluarea expresiei logice `EL(i)`. Atât timp, cât expresia logică `EL(i)` va fi adevărată, se va executa instrucțiunea `I(i)`.
- Numărul de ori de reperare a execuției instrucțiunii din corpul structurii nu este cunoscut la începutul executării structurii iterative. Atunci când valoarea de adevăr a expresiei logice `EL(i)` este fals, se părăsește corpul structurii iterative. Abia acum se poate determina de câte ori s-a executat repetarea instrucțiunii `I(i)`.
- În cazul în care valoarea de adevăr a expresiei logice `EL(i)` este adevărat, indiferent de câte ori se incrementează contorul se intră într-o buclă infinită, din care se poate ieși prin combinația de taste `CTRL+C`.

3.5. PROBLEME - STRUCTURI SECVENȚIALE

Problema 3.1

Se consideră două numere reale a=8 și b=15.

Să se realizeze o schemă logică pentru calculul următoarele medii:

- Media aritmetică:

$$m_a = \frac{a+b}{2}$$

- Media geometrică:

$$m_g = \sqrt{ab}$$

- Media armonică:

$$m_r = \frac{2ab}{a+b}$$

- Media pătratică:

$$m_p = \sqrt{\frac{a^2+b^2}{2}}$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.8.

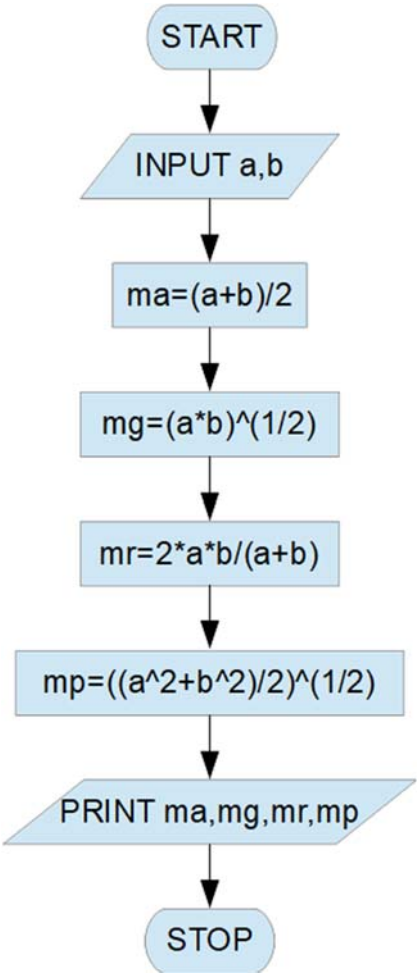
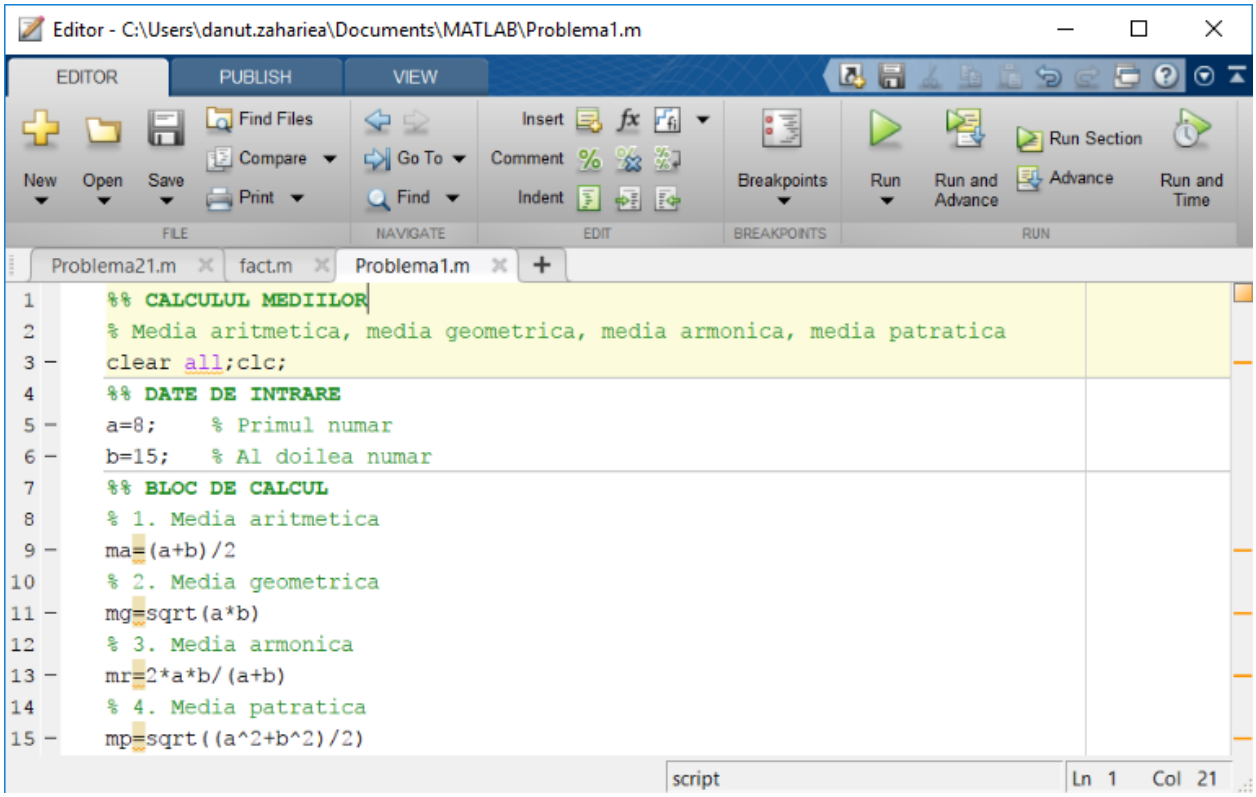


Figura 3.8. Schema logică pentru calculul mediilor.

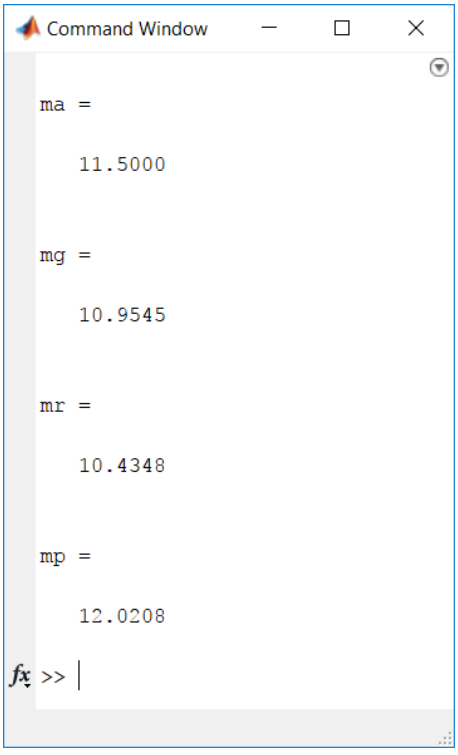
Observații

- Mediile aritmetică, geometrică, armonică și pătratică se calculează cu o structură secvențială.
- Datele de intrare sunt cele două numere `a` și `b`.
- Ordinea de executare a instrucțiunilor de calcul a celor patru medii poate fi modificată fără alterarea rezultatelor.
- Datele obținute în urma executării algoritmului sunt cele patru medii, `ma`, `mg`, `mr` și `mp` care reprezintă rezultatul final și care constituie deci, datele de ieșire ale algoritmului.

Fișierul de tip `script` pentru calculul mediilor este prezentat în figura 3.9, a). Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul celor patru medii. Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.9, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.9. Fișier `script` pentru calculul mediilor.

Problema 3.2

Se consideră un rezervor cilindric vertical având raza r=0,5 m și înălțimea h=1,5 m, figura 3.10.

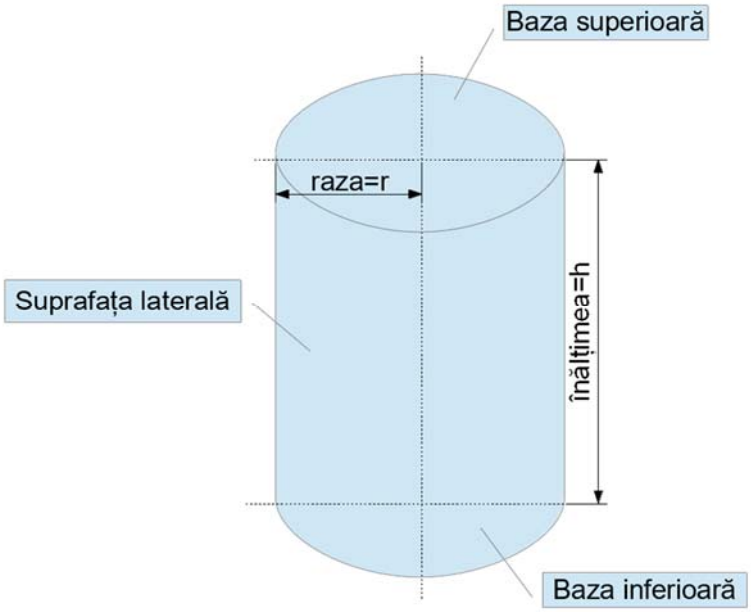


Figura 3.10. Rezervor cilindric vertical.

Să se realizeze o schemă logică pentru calculul volumului și a ariei totale a rezervorului cilindric vertical cunoscând:

- Aria bazei inferioare:

$$A_{bi} = \pi r^2$$

- Aria bazei superioare:

$$A_{bs} = \pi r^2$$

- Aria laterală:

$$A_l = 2\pi rh$$

- Aria totală:

$$A_t = A_{bi} + A_{bs} + A_l$$

- Volumul:

$$V = \pi r^2 h$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Observație

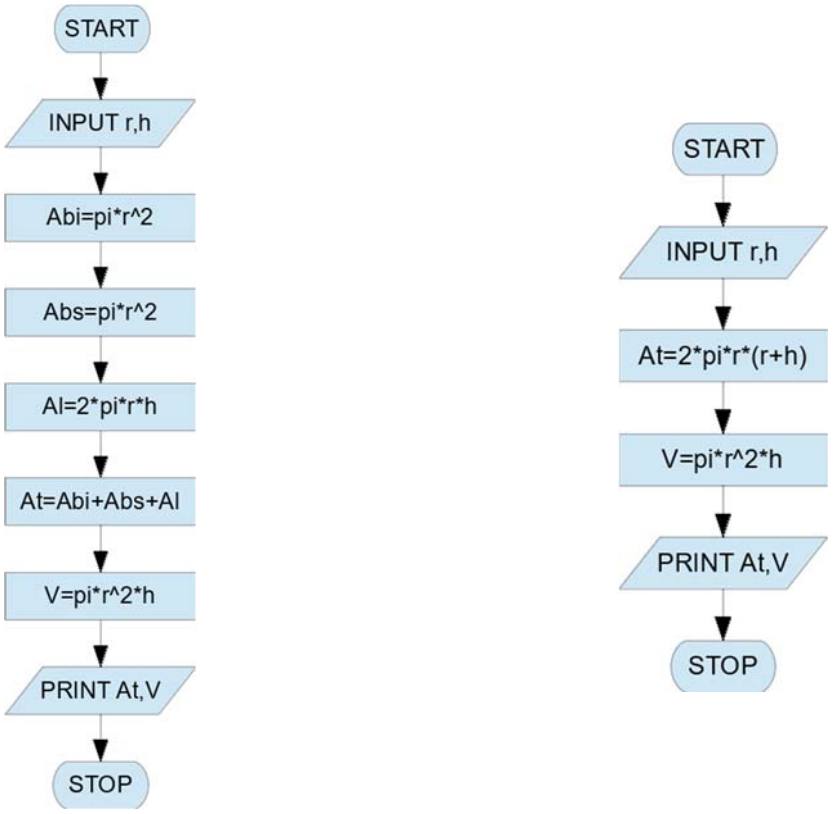
Efectuând calculele în relația ariei totale se obține:

$$A_t = A_{bi} + A_{bs} + A_l = \pi r^2 + \pi r^2 + 2\pi rh = 2\pi r(r + h)$$

Cum în problemă nu se solicită calculul separat al ariei celor două baze, respectiv a ariei laterale, ci doar calculul ariei totale, se poate simplifica schema logică prin eliminarea instrucțiunilor de calcul separat al acestor trei arii și calcularea directă a ariei totale, obținându-se astfel un algoritm la fel de eficace, însă mai eficient.

Rezolvare

Schemele logice sunt prezentate în figura 3.11.



a) calculul separat al ariilor b) calculul direct al ariei totale

Figura 3.11. Scheme logice pentru calculul volumului și a ariei totale a unui rezervor cilindric vertical.

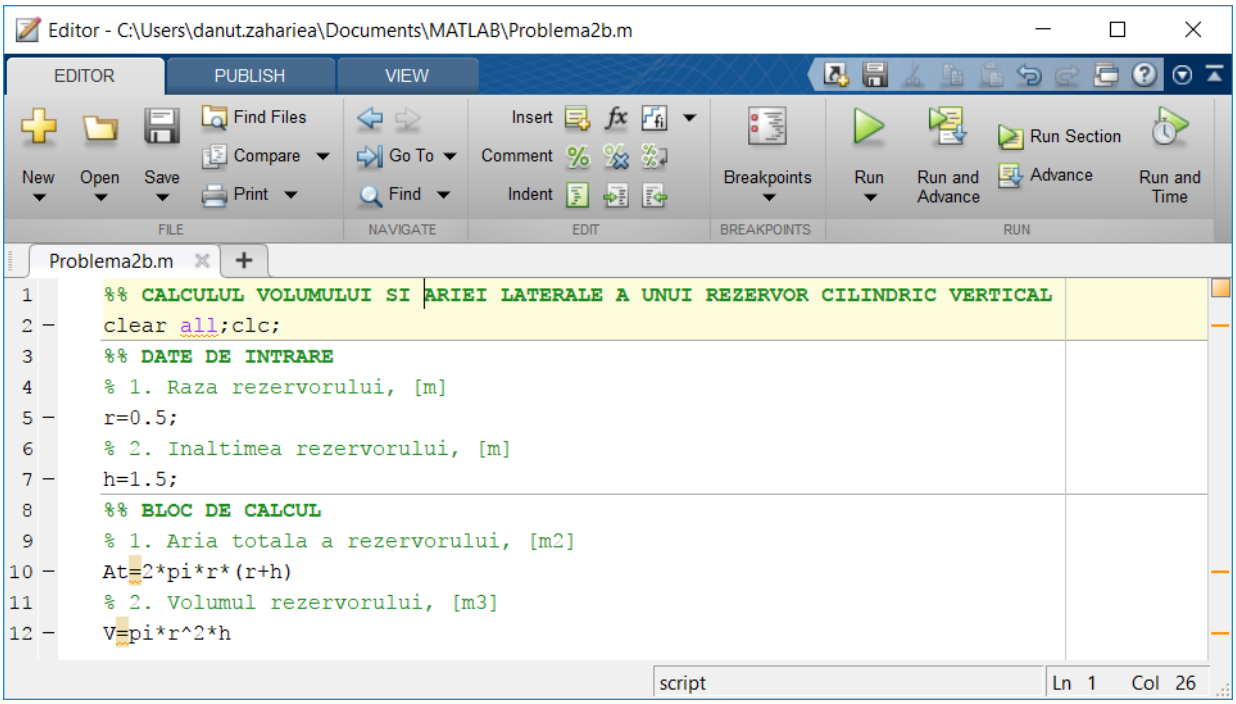
Observații

- Volumul și aria laterală a rezervorului cilindric vertical se calculează cu o structură secvențială.
- Datele de intrare sunt raza `r` și înălțimea `h`.
- Datele de ieșire ale algoritmului sunt aria totală `At` și volumul cilindrului `V`.
- Ordinea de executare a instrucțiunilor de calcul a celor trei arii (aria bazei inferioare, aria bazei superioare, aria laterală) poate fi modificată fără alterarea rezultatelor.
- Instrucțiunea pentru calculul ariei totale trebuie plasată după calculul ariilor bazelor și a ariei laterale.
- Instrucțiunea pentru calculul volumului rezervorului cilindric vertical poate fi plasată oriunde, între instrucțiunile de citire a datelor `INPUT`, respectiv de scriere a rezultatelor `PRINT`.

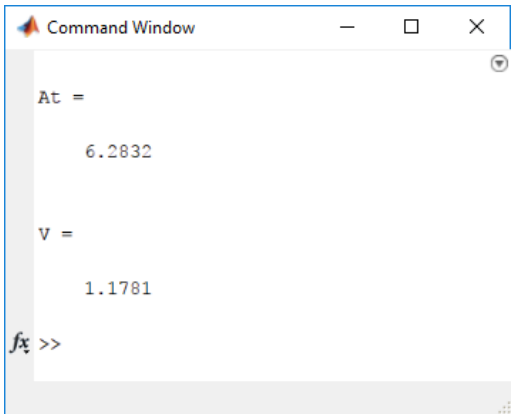
Fișierul `script` pentru calculul volumului și a ariei totale a rezervorului cilindric vertical bazat pe schema logică din figura 3.11, b) este prezentat în figura 3.12, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul ariei totale și a volumului rezervorului cilindric vertical.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.12, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.12. Fișier `script` pentru calculul volumului și a ariei totale a unui rezervor cilindric vertical.

Problema 3.3

Se consideră mișcarea cu suprafață liberă a apei într-un canal cu secțiune transversală dreptunghiulară având lățimea b=1,25 m și adâncimea h=0,8 m, figura 3.13.

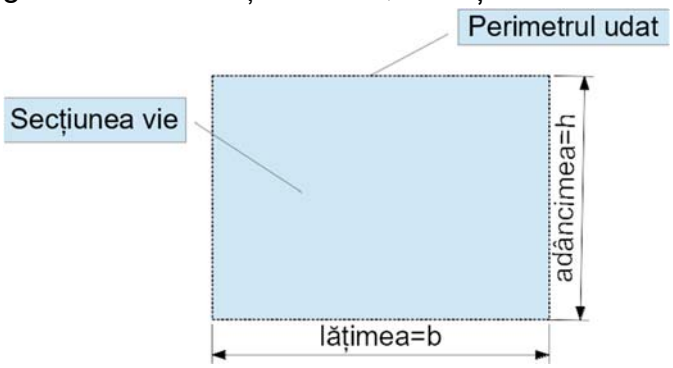


Figura 3.13. Secțiunea unui canal dreptunghiular.

Să se realizeze o schemă logică pentru calculul razei hidraulice a canalului cunoscând:

- aria secțiunii vii:

$$A = b \cdot h$$

- perimetrul udat:

$$P = b + 2 \cdot h$$

- raza hidraulică:

$$R = \frac{A}{P}$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.14.

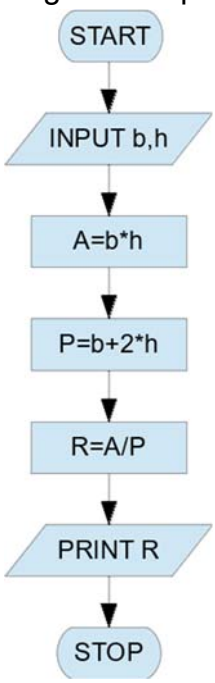


Figura 3.14. Schema logică pentru calculul razei hidraulice.

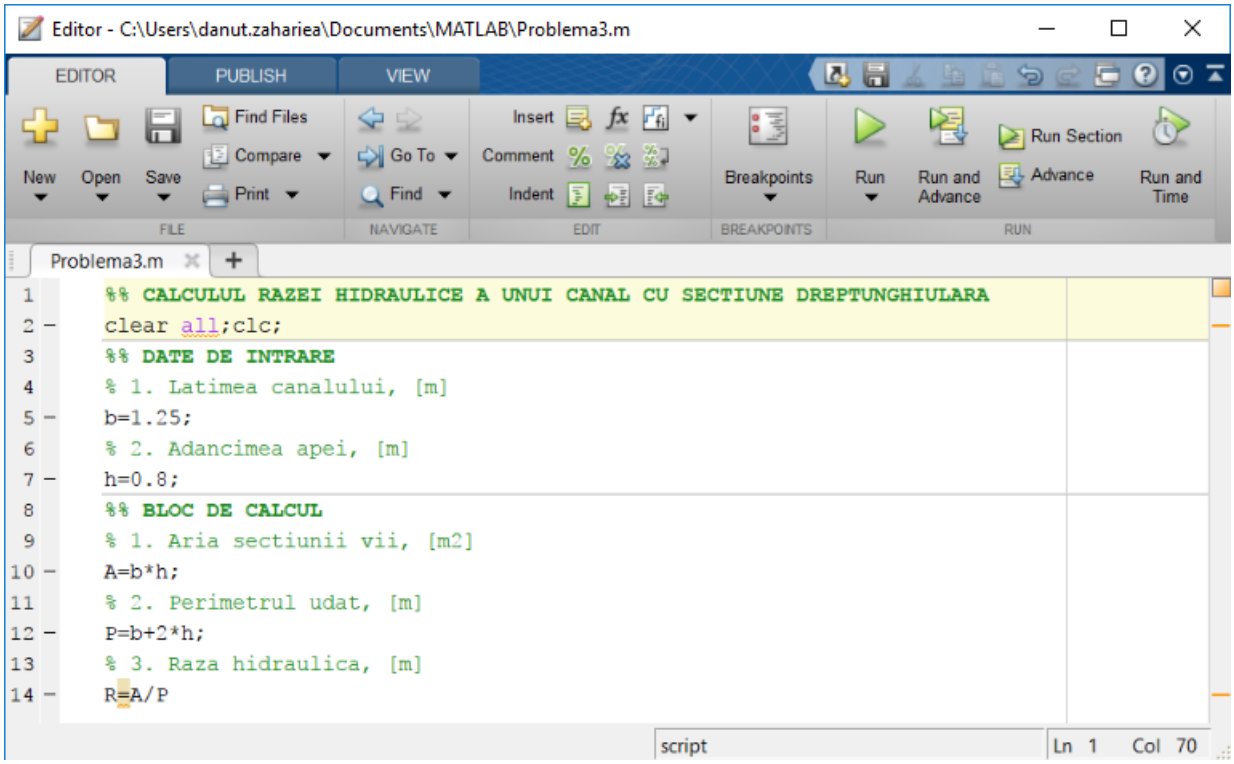
Observații

- Raza hidraulică se calculează cu o structură secvențială.
- Datele de intrare sunt lățimea `b` și adâncimea `h`.
- Ordinea de executare a instrucțiunilor pentru calculul ariei secțiunii vii `A` și a perimetrului udat `P` poate fi modificată, însă instrucțiunea pentru calculul razei hidraulice `R` trebuie obligatoriu plasată după calculul parametrilor `A` și `P`.
- Datele obținute în urma executării algoritmului sunt: aria secțiunii vii `A` și perimetrul udat `P`, care reprezintă rezultatele intermediare și raza hidraulică `R`, care reprezintă rezultatul final și care constituie deci, data de ieșire a algoritmului.

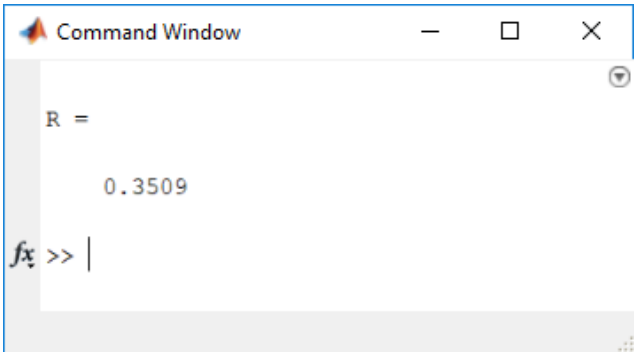
Fișierul de tip `script` pentru calculul razei hidraulice a unui canal cu secțiune dreptunghiulară este prezentat în figura 3.15, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul ariei secțiunii vii, a perimetrului udat și a razei hidraulice.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.15, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.15. Fișier `script` pentru calculul razei hidraulice a unui canal cu secțiune dreptunghiulară.

Problema 3.4

Se consideră un rezervor cilindric orizontal având raza interioară R=1,2 m și lungimea L=2,5 m în care se găsește apă cu densitatea $\rho = 1000$ kg/m^3, figura 3.16.

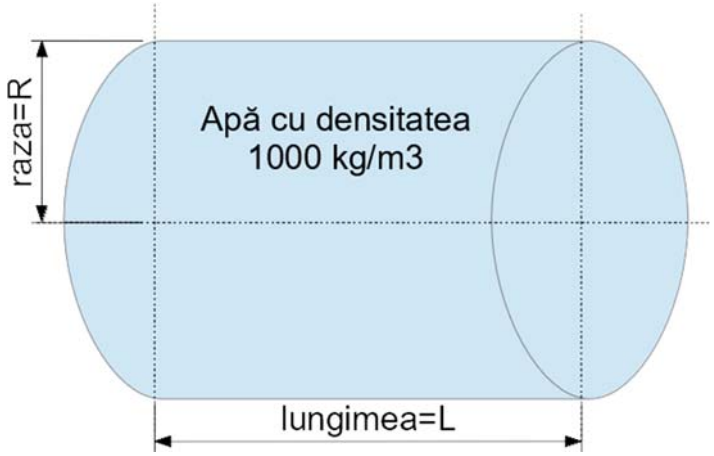


Figura 3.16. Rezervor cilindric orizontal.

Să se realizeze o schemă logică pentru calculul masei apei din interiorul rezervorului cunoscând:

- relația pentru calculul volumului rezervorului:

$$V = \pi R^2 L$$

- relația pentru calculul masei apei din rezervor:

$$m = \rho V$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.17.

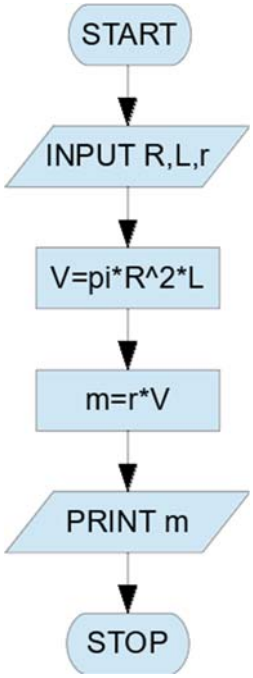


Figura 3.17. Schema logică pentru calculul masei apei din rezervorul cilindric orizontal.

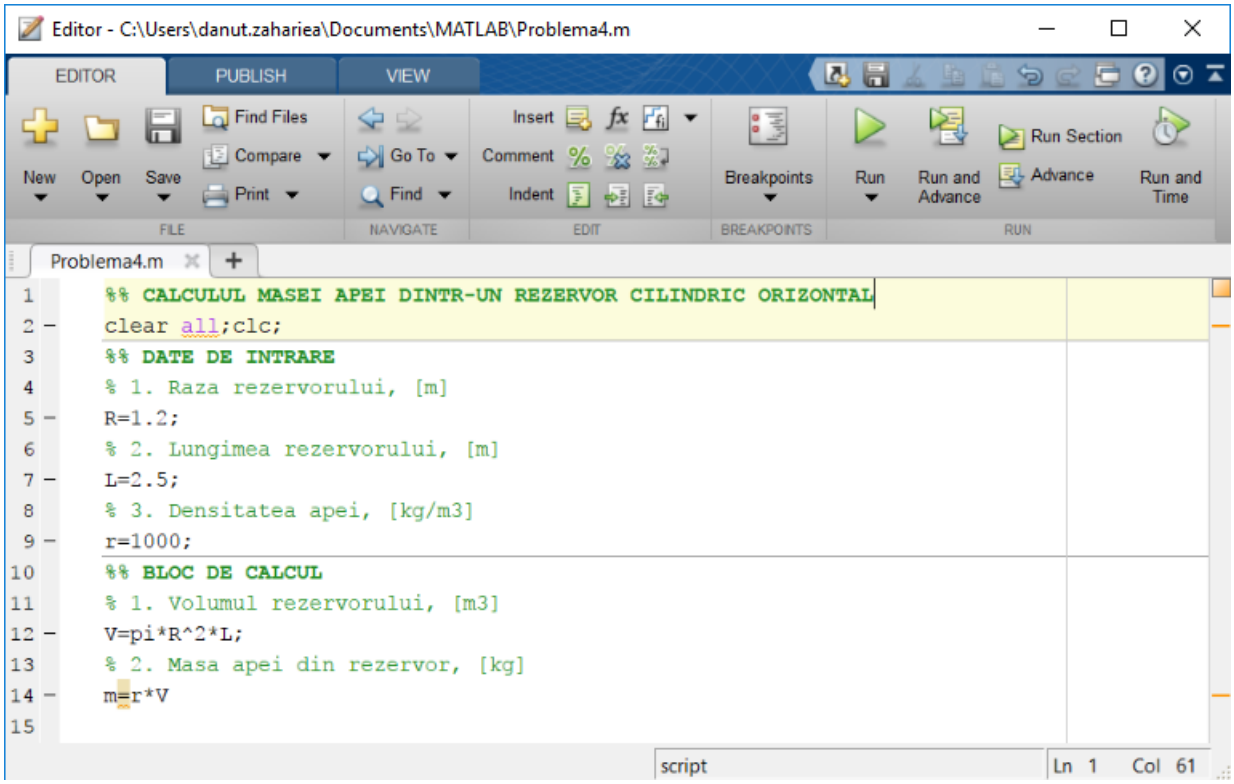
Observații

- Masa apei din rezervor se calculează cu o structură secvențială.
- Datele de intrare sunt raza interioară a rezervorului `R`, înălțimea rezervorului `H` și densitatea lichidului `r`.
- Datele obținute în urma executării algoritmului sunt: volumul interior al rezervorului `V`, care reprezintă un rezultat intermediar și masa de lichid `m`, care reprezintă rezultatul final și care constituie deci, data de ieșire a algoritmului.
- Densitatea lichidului ρ s-a notat în schema logică cu `r`.

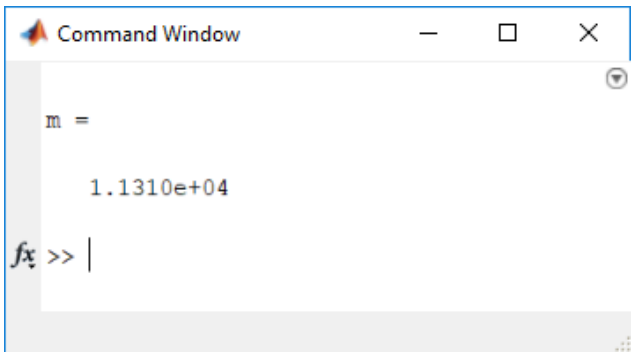
Fișierul de tip `script` pentru calculul masei apei dintr-un rezervor cilindric orizontal este prezentat în figura 3.18, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul volumului rezervorului și a masei apei din rezervor.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.18, b).



a) fișierul `script`

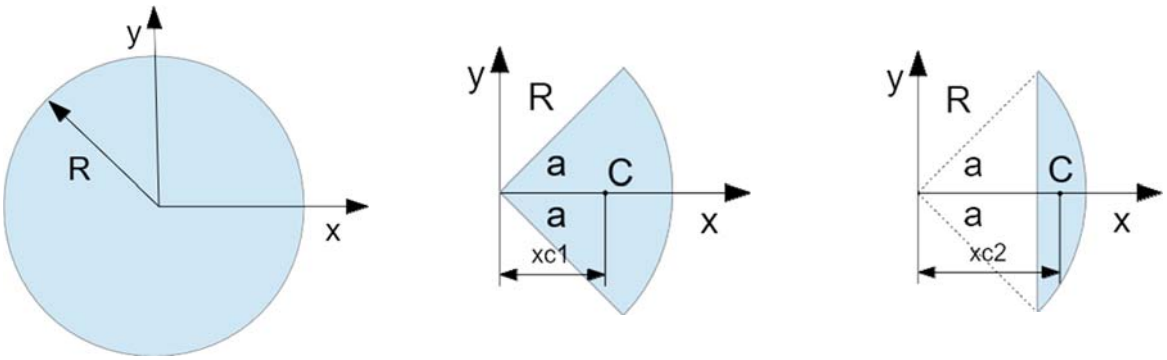


b) rezultatul obținut

Figura 3.18. Fișier `script` pentru calculul masei apei dintr-un rezervor cilindric orizontal.

Problema 3.5

Se consideră o placă circulară omogenă având raza R=1 m, figura 3.19, a).



a) placă circulară omogenă b) sector de cerc c) segment de cerc

Figura 3.19. Placă circulară, sectorul și segmentul de cerc.

Să se realizeze o schemă logică pentru calculul coordonatelor centrului de greutate pentru:

1) Sectorul de cerc (figura 3.19, b) având semi-unghiul la vârf a=$\pi/6$, dispus simetric față de axa x:

$$x_{c1} = \frac{2}{3} R \frac{\sin a}{a}$$

2) Segmentul de cerc (figura 3.19, c) având semi-unghiul la vârf a= $\pi/6$, dispus simetric față de axa x:

$$x_{c2} = \frac{2}{3} R \frac{(\sin a)^3}{a - \sin a \cos a}$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.20.

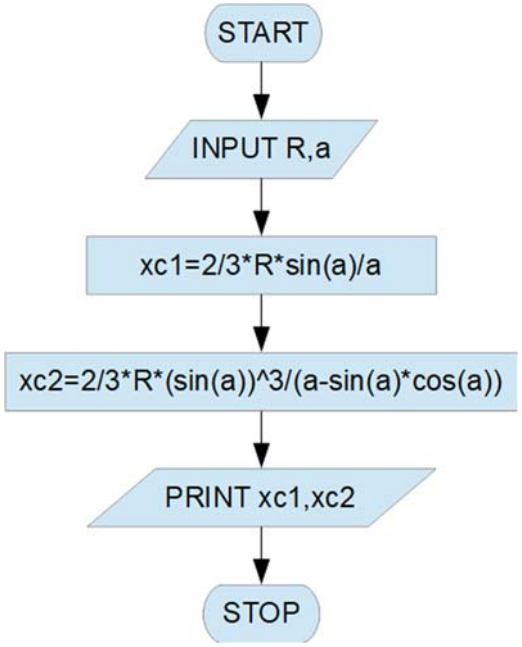


Figura 3.20. Schema logică pentru calculul coordonatelor centrului de greutate.

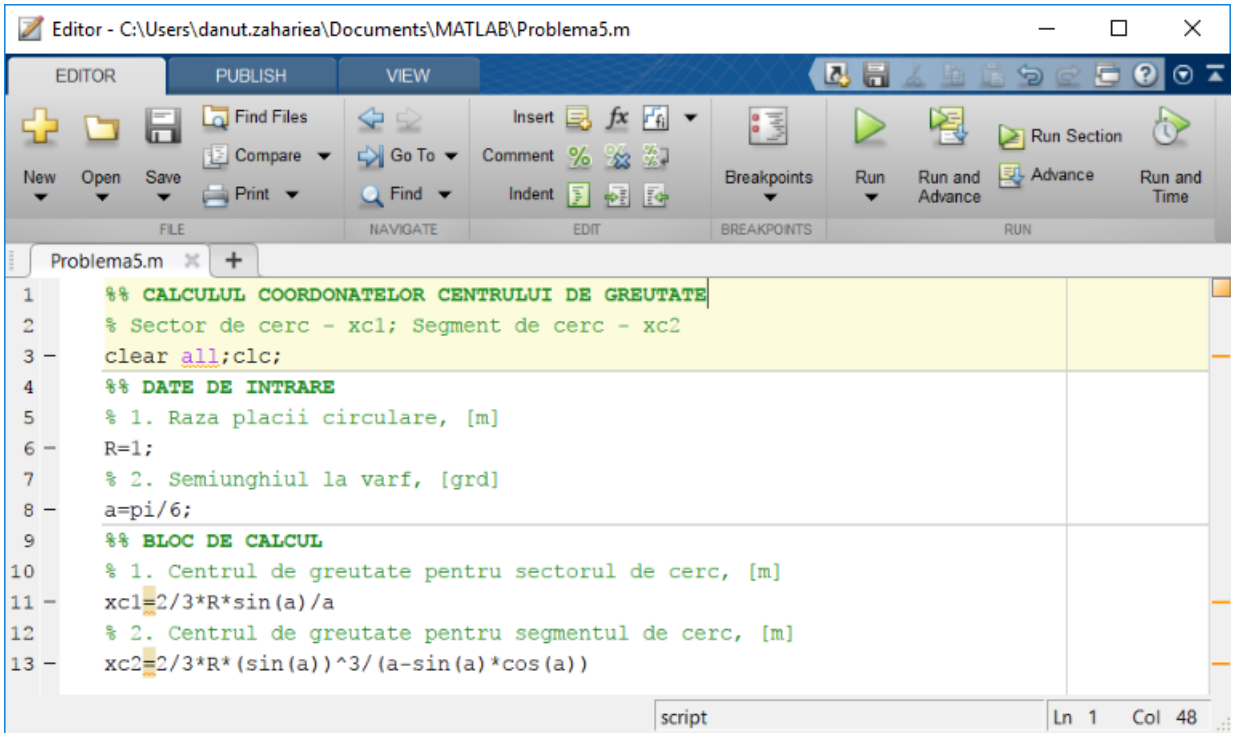
Observații

- Coordonatele centrelor de greutate se calculează cu o structură secvențială.
- Datele de intrare sunt raza cercului `R` și semi-unghiul la vârf `a`.
- Datele obținute în urma executării algoritmului sunt coordonatele centrului de greutate pentru cele două cazuri: `xc1` pentru sectorul de cerc, respectiv `xc2` pentru segmentul de cerc. Cele două valori astfel calculate reprezintă datele de ieșire ale algoritmului.
- Ordinea de executare a instrucțiunilor de calcul a celor două coordonate `xc1` și `xc2` poate fi modificată fără alterarea rezultatelor.

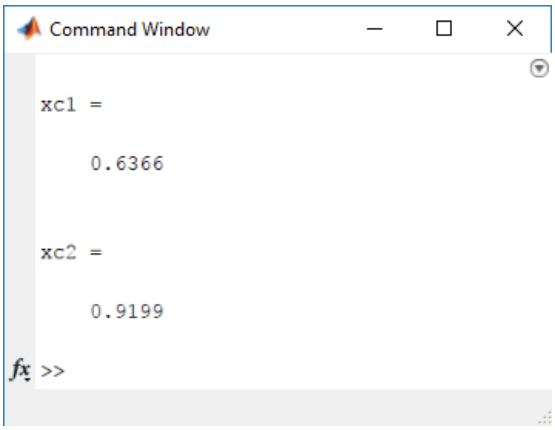
Fișierul de tip `script` pentru calculul coordonatelor centrului de greutate al unui sector de cerc, respectiv al unui segment de cerc este prezentat în figura 3.21, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul coordonatelor centrului de greutate pentru cele două figuri geometrice: sectorul de cerc, respectiv segmentul de cerc.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.21, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.21. Fișier `script` pentru calculul coordonatelor centrului de greutate al unui sector de cerc (`xc1`) și al unui segment de cerc (`xc2`).

Problema 3.6

Se consideră un punct material de masă m care este lansat în câmp gravitațional cu viteza inițială v_0=25 m/s după o direcție care face unghiul α=60° cu planul orizontal, figura 3.22.

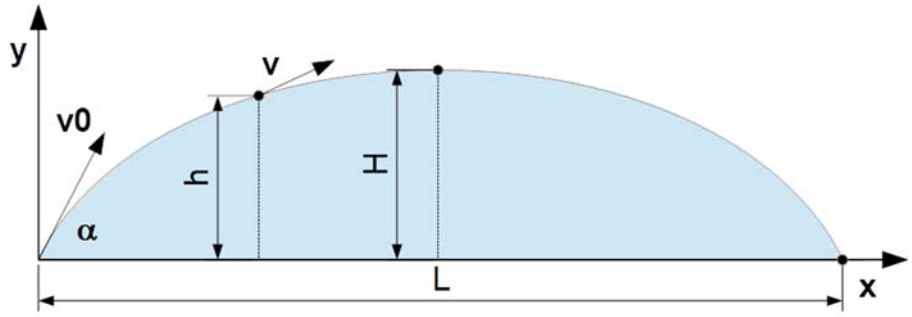


Figura 3.22. Aruncarea oblică.

Să se realizeze o schemă logică pentru calculul următorilor parametri: bătaia, înălțimea maximă și timpul total până la impact. Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Neglijând rezistența aerului, relațiile de calcul pentru parametrii mișcării sunt:

- Bătaia traiectoriei:

$$L = v_0^2/g \cdot \sin 2\alpha$$

- Înălțimea maximă a traiectoriei:

$$H = v_0^2/g \cdot (\sin \alpha)^2$$

- Timpul până la impact:

$$t = 2v_0/g \cdot \sin \alpha$$

Rezolvare

Schema logică este prezentată în figura 3.23.

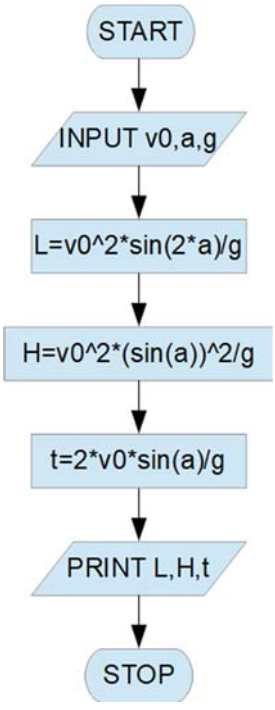


Figura 3.23. Schema logică pentru calculul parametrilor aruncării oblice.

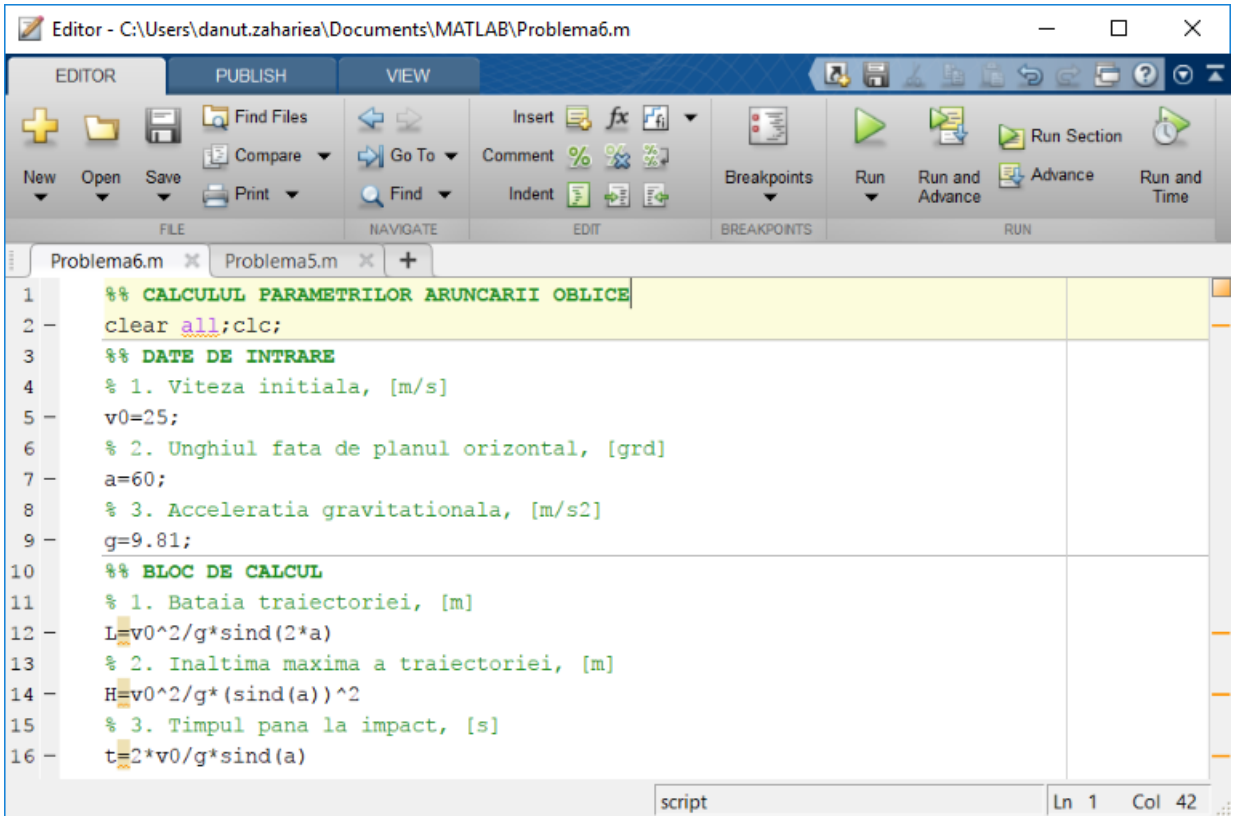
Observații

- Parametrii mișcării se calculează cu o structură secvențială.
- Datele de intrare sunt viteza inițială `v0`, unghiul de lansare `a` și accelerația gravitațională `g`.
- Ordinea de executare a instrucțiunilor poate fi modificată fără alterarea rezultatelor
- Datele obținute în urma executării algoritmului sunt bătaia `L`, înălțimea maximă a traiectoriei `H` și timpul până la impact `t`, care reprezintă de altfel și datele de ieșire ale algoritmului.

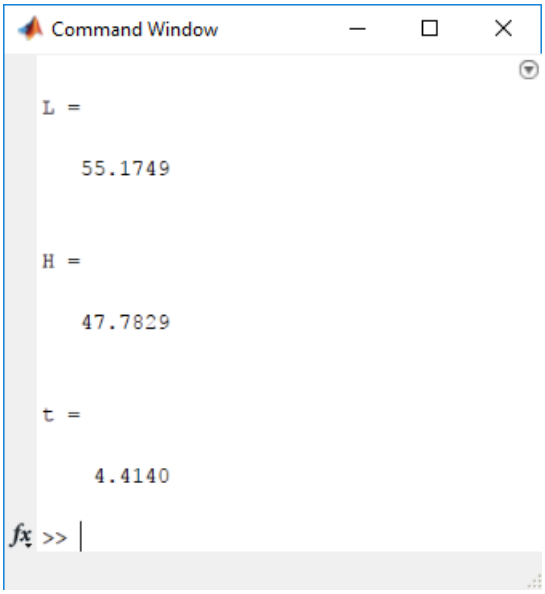
Fișierul de tip `script` pentru calculul parametrilor aruncării oblice este prezentat în figura 3.24, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul parametrilor aruncării oblice: bătaia, înălțimea maximă și timpul total până la impact.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.24, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.24. Fișier `script` pentru calculul parametrilor aruncării oblice: bătaia, înălțimea maximă și timpul total până la impact.

Problema 3.7

Troposfera este zona atmosferei din imediata vecinătate a pământului având grosimea $H\cong$10000 m, figura 3.25.

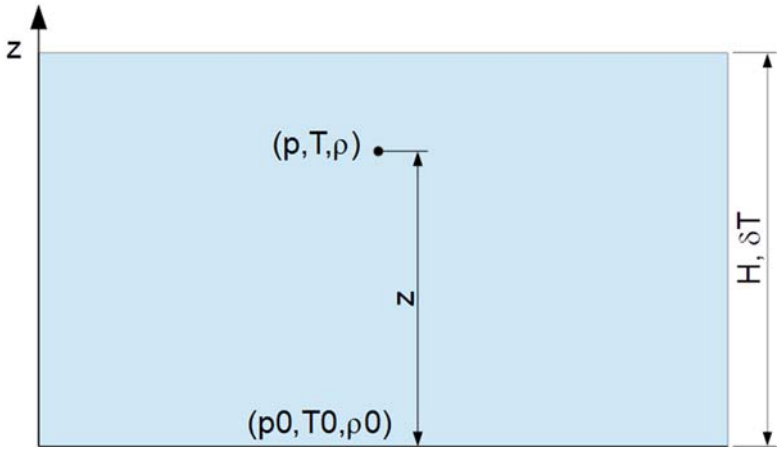


Figura 3.25. Parametrii atmosferici în troposferă.

Să se realizeze o schemă logică pentru calculul parametrilor atmosferici (temperatura, densitatea și presiunea aerului) la înălțimea z=8000 m cunoscând:

- pentru troposferă se consideră o variație liniară a temperaturii cu înălțimea caracterizată prin gradientul vertical de temperatură:
$$\delta T\text{=0.0065 °C/m}$$
- parametrii atmosferei standard la nivelul mării sunt:
  - presiunea p_0=101325 Pa
  - temperatura T_0=288.15 K
- constanta specifică a aerului are valoarea:
$$R\text{=287 J/KgK}$$
- accelerația gravitațională la nivelul mării are valoarea:
$$g\text{=9.806 m/s}^2$$
- densitatea aerului la nivelul mării [kg/m^3]:
$$\rho_0 = \frac{p_0}{RT_0}$$
- variația temperaturii cu altitudinea:
$$T = T_0 - \delta T \cdot z$$
- presupunând o atmosferă politropă, exponentul politropic se calculează cu relația:
$$n = \frac{1}{1 - \frac{R\delta T}{g}}$$
- variația densității cu altitudinea:
$$\rho = \rho_0 \cdot \left(1 - \frac{n-1}{n} \cdot \frac{gz}{RT_0}\right)^{\frac{1}{n-1}}$$
- variația presiunii cu altitudinea:
$$p = p_0 \cdot \left(\frac{\rho}{\rho_0}\right)^n$$

Să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.26.

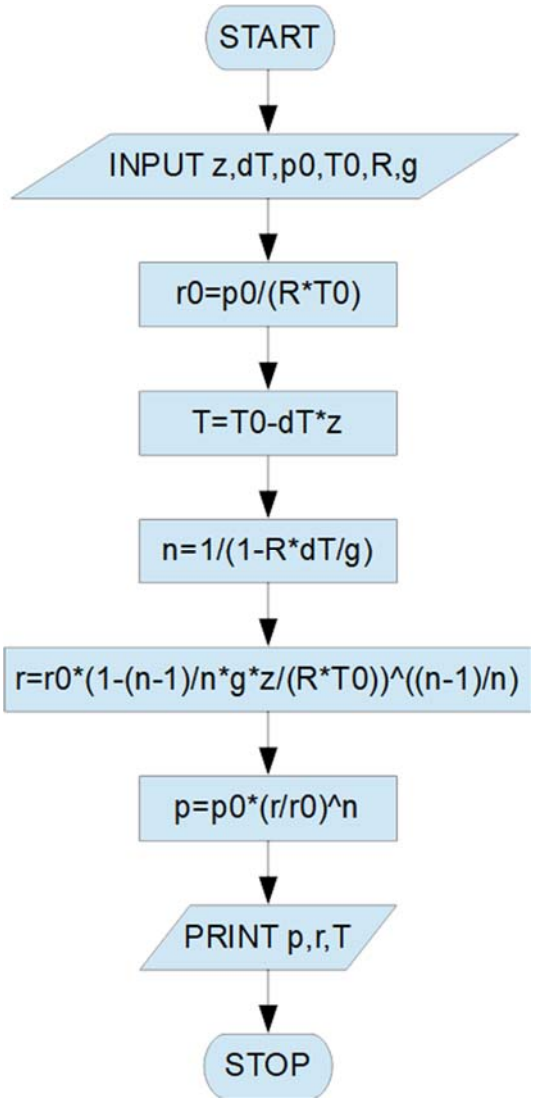


Figura 3.26. Schema logică pentru calculul parametrilor atmosferici.

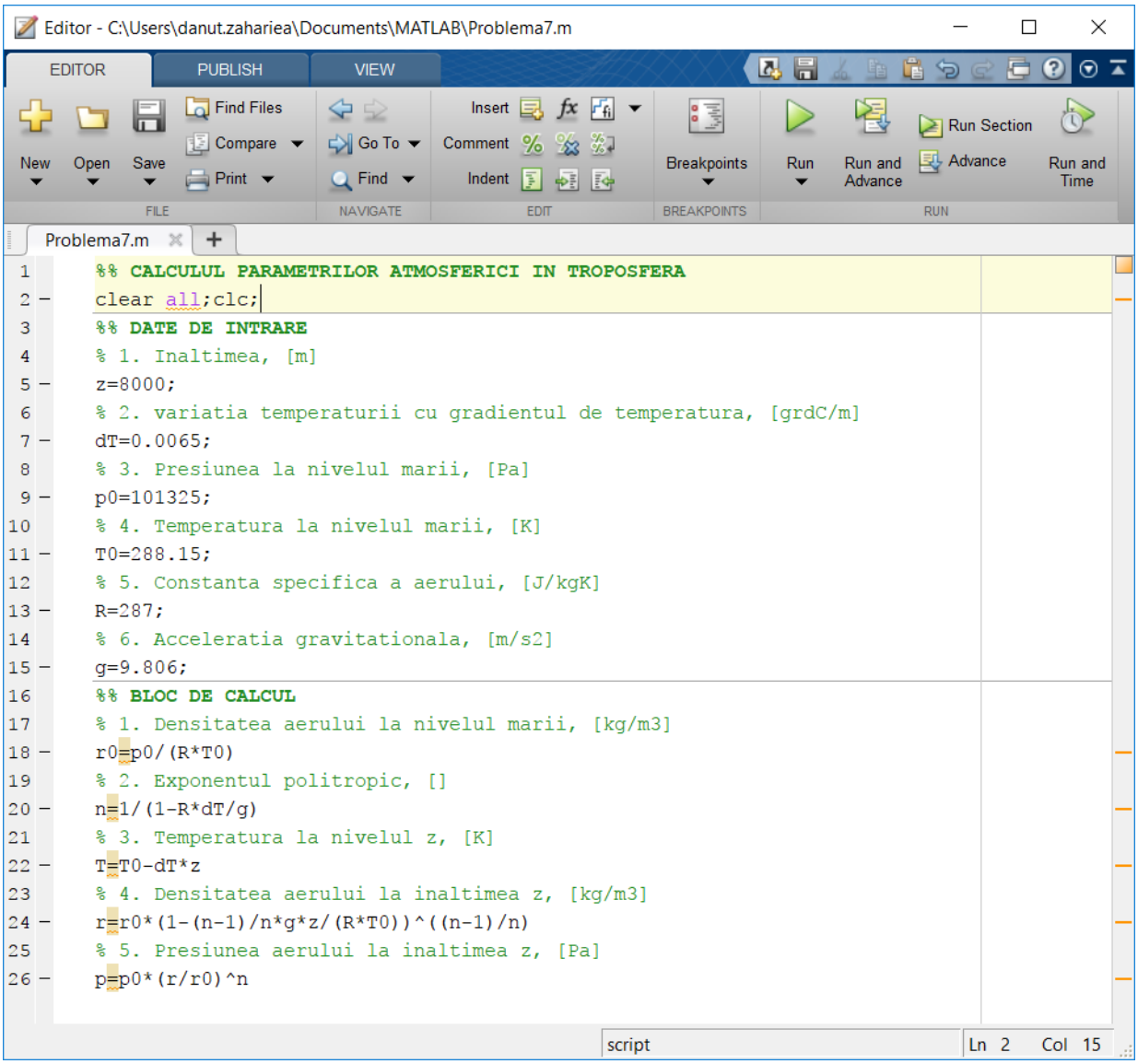
Observații

- Parametrii atmosferici se calculează cu o structură secvențială.
- Datele de intrare sunt înălțimea `z`, gradientul vertical de temperatură `dT`, presiunea `p0` și temperatura `T0` la nivelul mării, constanta specifică a aerului `R` și accelerația gravitațională `g`.
- Ordinea de executare a instrucțiunilor de calcul pentru densitatea `r0` la nivelul mării, pentru temperatura `T` la înălțimea `z` și pentru exponentul politropic `n`, poate fi modificată fără alterarea rezultatelor, însă calculul presiunii `p` și a densității `r` la înălțimea `z` trebuie să se facă după determinarea exponentului politropic `n`.
- Datele obținute în urma executării algoritmului sunt densitatea la nivelul mării `r0` și exponentul politropic `n`, care reprezintă rezultate intermediare, precum și temperatura `T`, densitatea `r` și presiunea `p` la înălțimea `z`, care reprezintă datele de ieșire ale algoritmului.

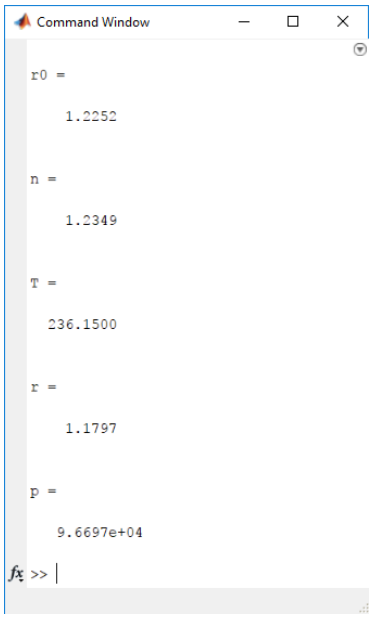
Fișierul de tip `script` pentru calculul parametrilor atmosferici în troposferă este prezentat în figura 3.27, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul coordonatelor centrului de greutate pentru cele două figuri geometrice: sectorul de cerc, respectiv segmentul de cerc.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.27, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.27. Fișier `script` pentru calculul parametrilor atmosferici în troposferă.

Problema 3.8

Transmiterea mișcării de rotație și a puterii între arborele motorului electric de acționare și arborele unui ventilator centrifugal se realizează prin intermediul unei transmisii prin curea, figura 3.28. Transmisia este formată din două roți de curea, roata conducătoare 1 și roata condusă 2, și respectiv o curea lată 3, montată cu pretensionare.

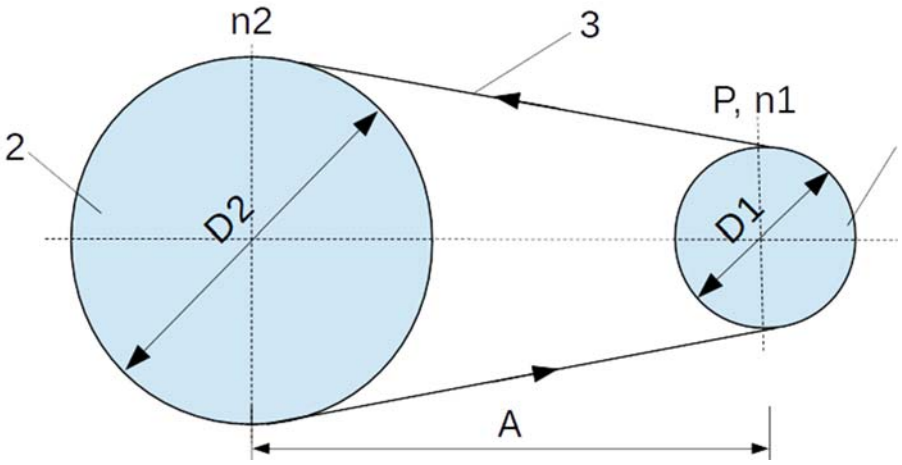


Figura 3.28. Transmisie prin curea lată.
1-roată conducătoare; 2-roată condusă; 3-curea lată

Cunoscând puterea motorului electric de acționare P=350 W, turația motorului n_1=1450 rpm, turația ventilatorului n_2=500 rpm, alunecarea relativă ε=1÷2% și distanța A=700 mm dintre axele celor două roți de curea, să se determine raportul de transmitere teoretic, viteza teoretică a curelei și lungimea necesară a curelei.

Rezolvarea problemei presupune parcurgerea următoarelor etape:

- Calculul raportului de transmitere teoretic:

$$i = \frac{n_1}{n_2}$$

- Calculul diametrului roții conducătoare, [m]:

$$D_1 = (0{,}09 \dots 0{,}11) \cdot \sqrt[3]{\frac{P}{n_1}}$$

- Calculul vitezei teoretice a curelei:

$$V = \frac{\pi D_1 n_1}{60}$$

- Calculul diametrului roții conduse, [m]:

$$D_2 = \frac{n_1}{n_2} D_1 (1 - \varepsilon)$$

- Calculul diametrului mediu al roților de curea, [m]:

$$D_m = \frac{D_1 + D_2}{2}$$

- Lungimea necesară a curelei, [m]:

$$L = 2A + \pi D_m + \frac{(D_2 - D_1)^2}{4A}$$

Să se realizeze o schemă logică pentru rezolvarea problemei și să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.29.

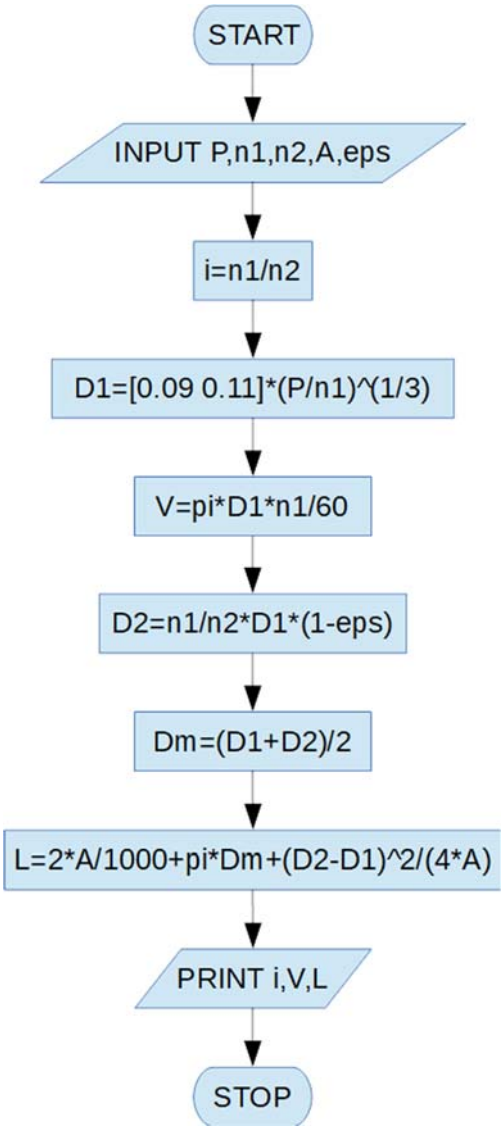


Figura 3.29. Schema logică pentru calculul parametrilor transmisiei prin curele late.

Observații

- Parametrii transmisiei prin curele late se calculează cu o structură secvențială.
- Datele de intrare sunt puterea `P`, turația celor două roți de curea `n1` și `n2`, distanța dintre cele două roți de curea `L` și alunecarea relativă `eps`.
- Prezența intervalului `[0,09 0,11]` implică adoptarea unei anumite valori din intervalul rezultat în urma calculului diametrului `D1`.
- Parametrul `eps` este de asemenea un interval, deci și în acest caz se impune adoptarea unei anumite valori din intervalul rezultat în urma calculului diametrului `D2`.
- Datele obținute în urma executării algoritmului sunt raportul de transmitere teoretic `i`, diametrul roții conducătoare `D1`, viteza teoretică a curelei `V`, diametrul roții conduse `D2`, diametrul mediu al roților de curea `Dm` și lungimea necesară a curelei `L`.
- Datele de ieșire se referă în acest caz doar la `i`, `V` și `L`.

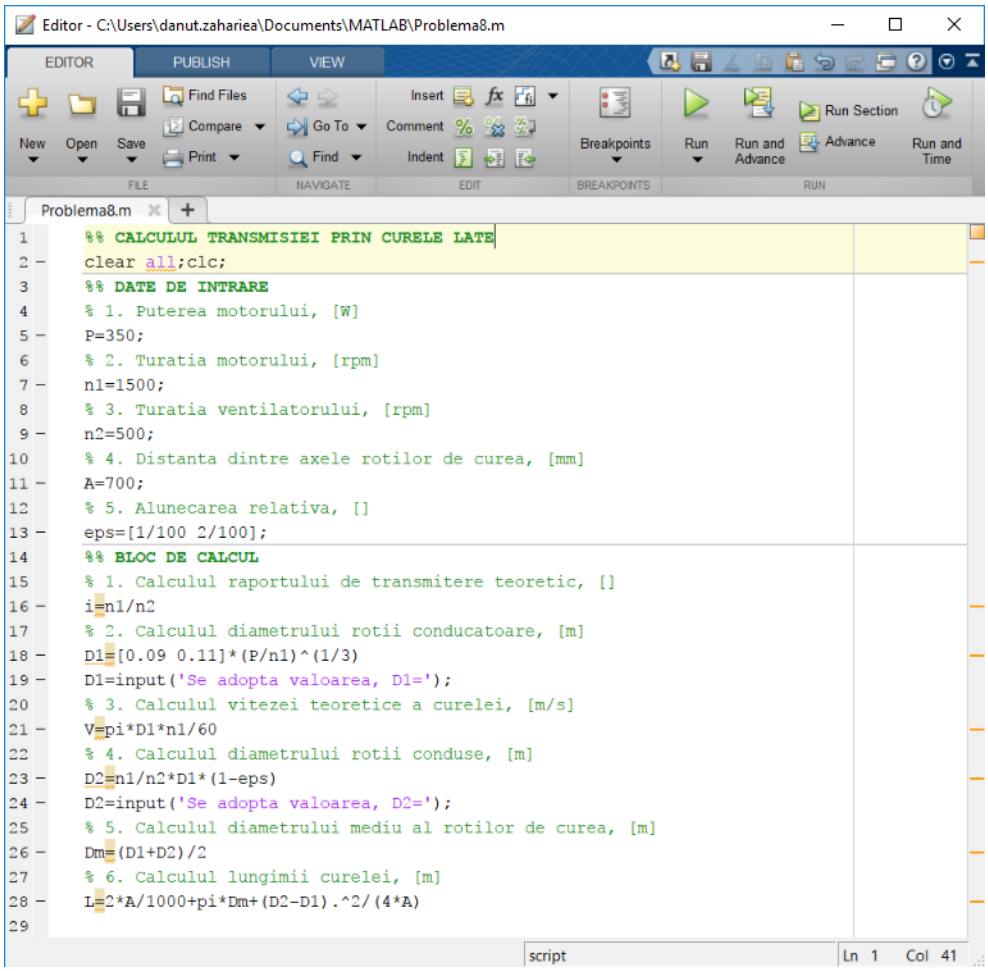
Fișierul de tip `script` pentru calculul parametrilor transmisiei prin curele este prezentat în figura 3.30, a). Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul tuturor parametrilor transmisiei prin curele late.

Adoptarea unei anumite valori pentru cele două diametre `D1` și `D2`, din intervalele rezultate în urma calculelor se face în mod interactiv folosind instrucțiunile:

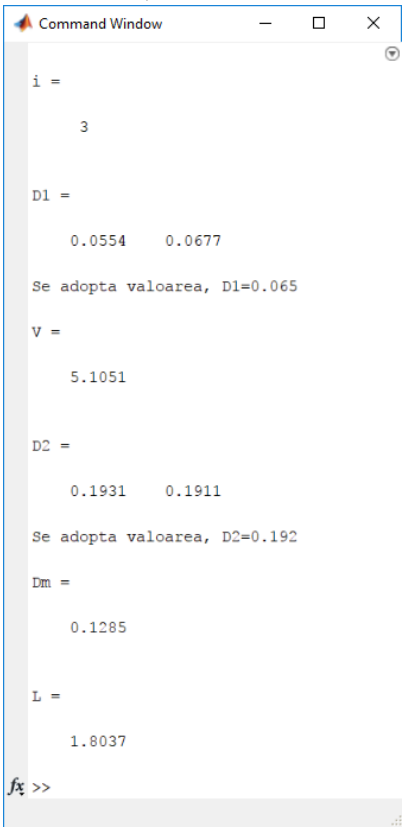
```
D1=input('Se adopta valoarea, D1=');
D2=input('Se adopta valoarea, D2=');
```

Execuția programului se oprește până la introducerea de către utilizator a valorii dorite din intervalul specificat.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.30, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.30. Fișier `script` pentru calculul parametrilor transmisiei prin curele late.

Problema 3.9

Să se dimensioneze arborele unei pompe centrifuge pentru vehicularea apei reci cunoscând: debitul Q=40 m^3/h, sarcina H=20 m, turația n=1450 rpm, densitatea apei ρ=1000 kg/m^3, randamentul global al pompei η=0,75, materialul din care este confecționat arborele pompei OLC 45 având tensiunea admisibilă la torsiune τ_{at}=20 MPa.

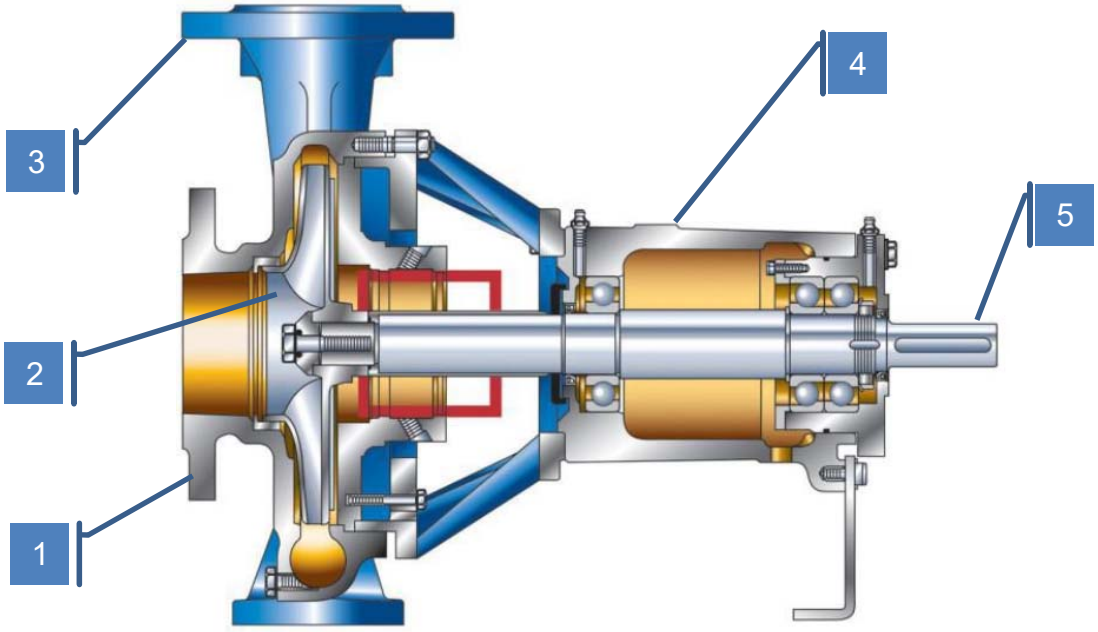


Figura 3.31. Pompa centrifugă, [www.sulzer.com].
1-racord de aspirație; 2-rotor; 3-racord de refulare; 4-carcasă rulmenți; 5-arbore

Rezolvarea problemei presupune parcurgerea următoarelor etape:

- Calculul puterii utile a pompei:

$$P_u = \rho g Q H$$

- Calculul puterii de antrenare:

$$P = \frac{P_u}{\eta}$$

Puterea de antrenare se adoptă din șirul puterilor standard ale motoarelor electrice [3 4 5,5 7,5] kW.

- Calculul vitezei unghiulare:

$$\omega = \frac{2\pi n}{60}$$

- Calculul momentului de torsiune la arbore:

$$M_t = \frac{P}{\omega}$$

- Calculul diametrului arborelui:

$$d = \sqrt[3]{\frac{16 M_t}{\pi \tau_{at}}}$$

Valoarea finală a diametrului arborelui se va adopta, acoperitor, în funcție de valoarea rezultată în urma calculelor.

Să se realizeze o schemă logică pentru rezolvarea problemei și să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.32.

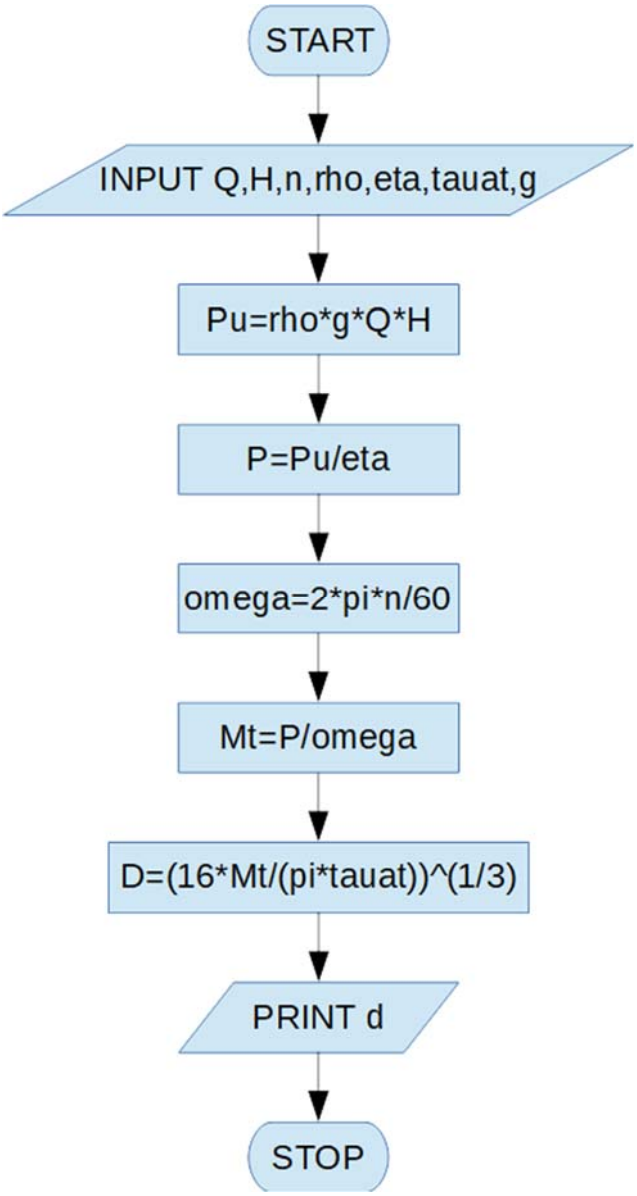


Figura 3.32. Schema logică pentru dimensionarea arborelui unei pompe centrifuge.

Observații

- Diametrul arborelui pompei centrifuge se calculează cu o structură secvențială.
- Datele de intrare sunt debitul `Q`, sarcina `H`, turația `n`, densitatea apei `rho`, randamentul global `eta`, tensiunea admisibilă la torsiune a materialului `tauat`, accelerația gravitațională `g`.
- Obținerea unor rezultate corecte din punct de vedere dimensional implică transformarea unităților de măsură a debitului și a tensiunii admisibile a materialului din m^3/h și MPa în m^3/s și Pa.
- Datele obținute în urma executării algoritmului sunt: puterea utilă `Pu`, puterea de antrenare `P`, viteza unghiulara `omega`, momentul de torsiune `Mt`, diametrul arborelui `d`.
- Arborele pompei centrifuge se dimensionează acoperitor, deci valoarea finală a diametrului arborelui se va adopta, acoperitor, în funcție de valoarea rezultată în urma calculelor.
- Datele de ieșire se referă în acest caz doar la diametrul arborelui `d`.

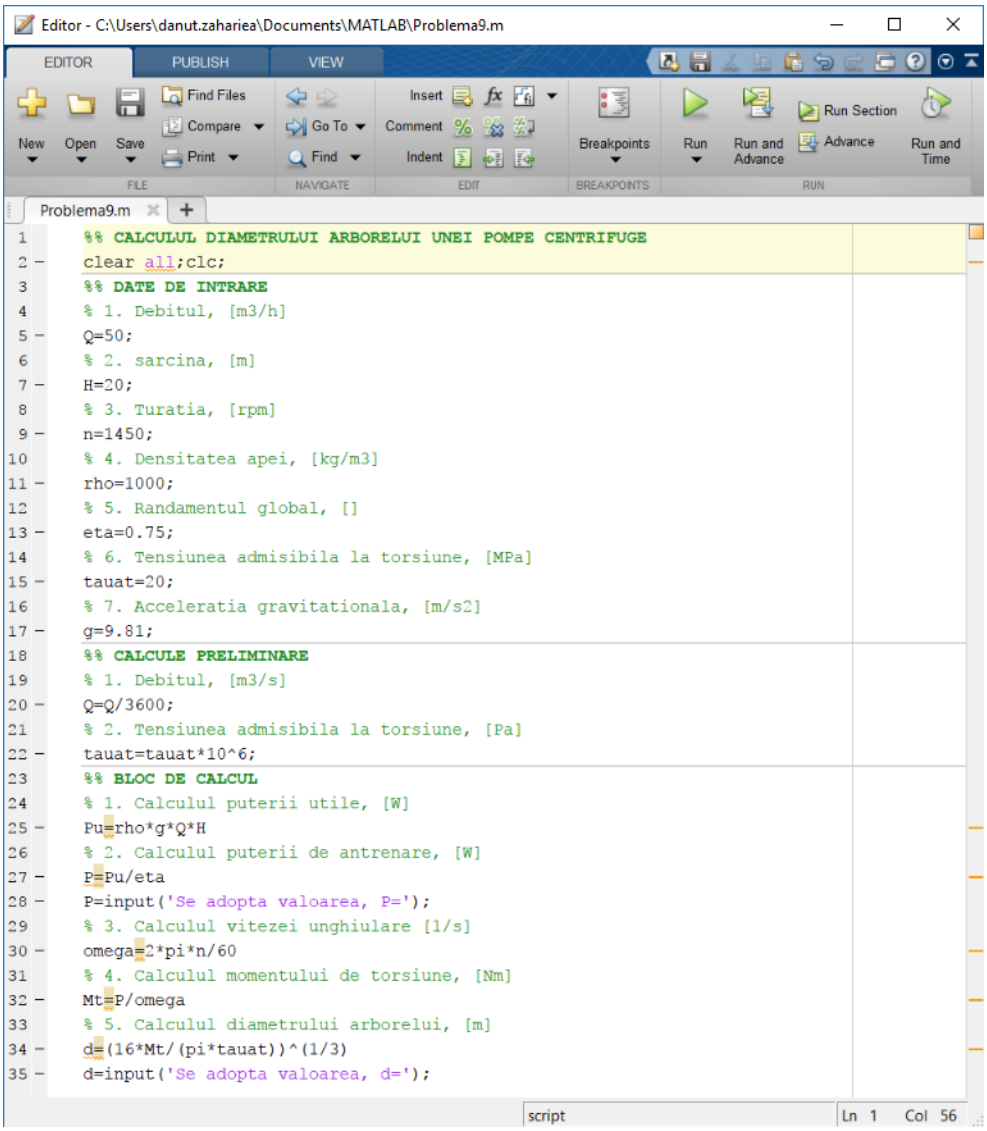
Fișierul de tip `script` pentru calculul diametrului arborelui pompei centrifuge este prezentat în figura 3.33, a). Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul tuturor parametrilor necesari dimensionării arborelui.

Adoptarea unei anumite valori pentru puterea de antrenare `P` și pentru diametrul arborelui `d` se face în mod interactiv folosind instrucțiunile:

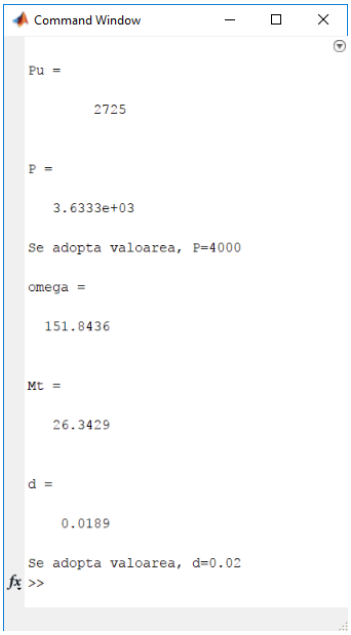
```
P=input('Se adopta valoarea, P=');
d=input('Se adopta valoarea, d=');
```

Execuția programului se oprește până la introducerea de către utilizator a valorii dorite din intervalul specificat.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.33, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.33. Fișier `script` pentru dimensionarea arborelui unei pompe centrifuge.

Problema 3.10

Se consideră o întreprindere care urmărește fabricarea unui produs al cărui preț de vânzare unitar este estimat la p=7,4 lei/buc. Pentru realizarea produsului sunt necesare cheltuieli variabile unitare de v=3,4 lei/buc și cheltuieli fixe estimate la suma de C_f=80000 lei. Admițând că prețul de vânzare unitar și cheltuielile variabile unitare rămân constante și presupunând că întreprinderea obține venituri doar din vânzarea produselor (corespunzătoare cifrei de afaceri), să se calculeze profitul întreprinderii pentru un volum al producției egal cu q=22000 bucăți.

Rezolvarea problemei presupune parcurgerea următoarelor etape:

- Calculul cifrei de afaceri, [lei]:

$$CA = p \cdot q$$

- Calculul cheltuielilor variabile, [lei]:

$$C_v = q \cdot v$$

- Calculul cheltuielilor totale, [lei]:

$$CT = C_f + C_v$$

- Calculul profitului, [lei]:

$$P = CA - CT$$

Să se realizeze o schemă logică pentru rezolvarea problemei și să se scrie un fișier de tip `script` pentru implementarea schemei logice.

Rezolvare

Schema logică este prezentată în figura 3.34.

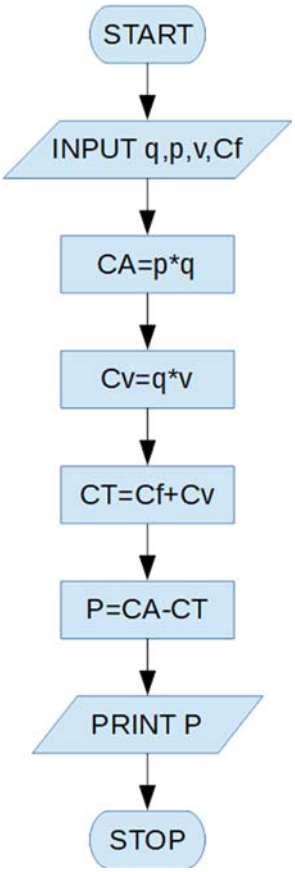


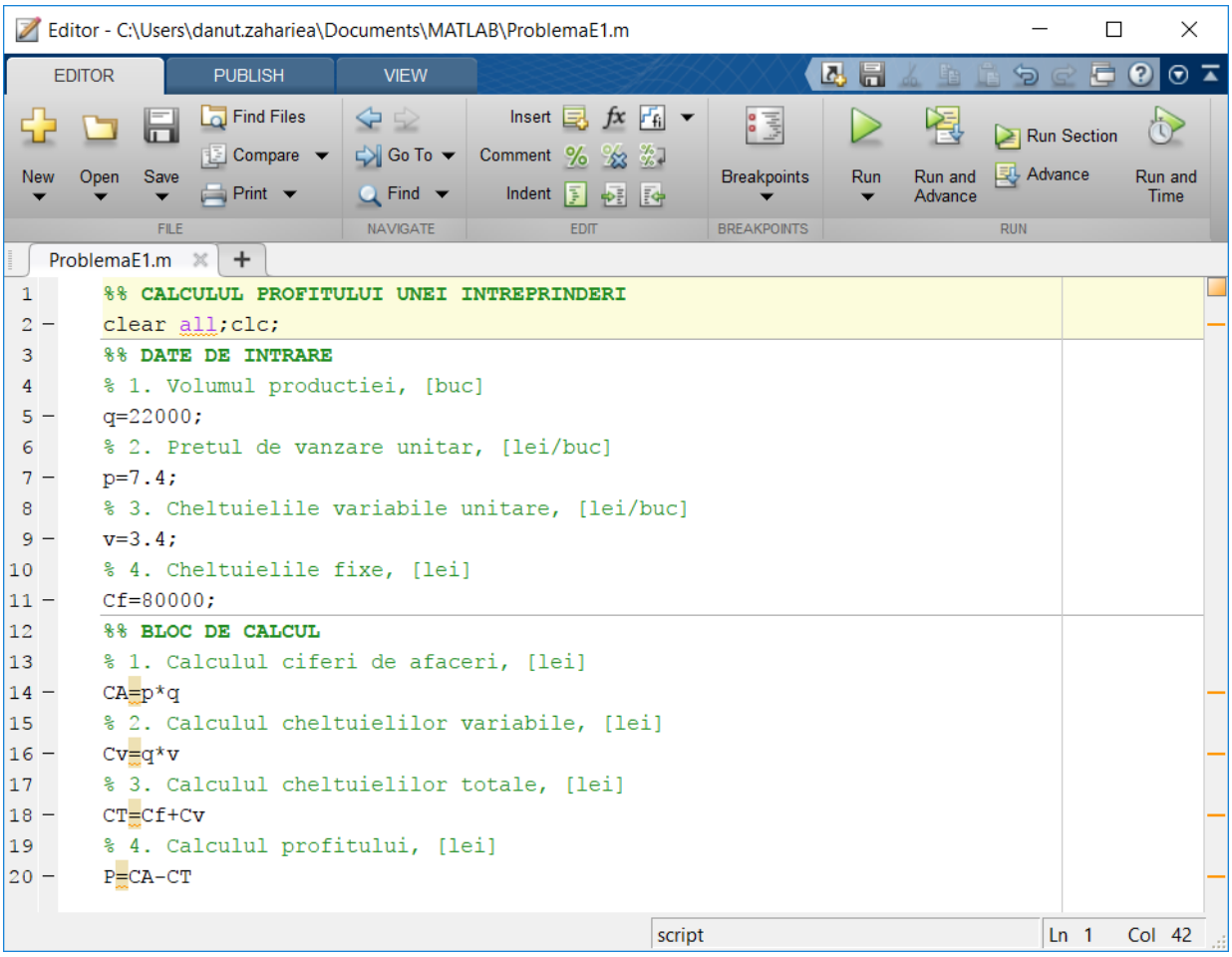
Figura 3.34. Schema logică pentru calculul profitului întreprinderii.

Observații

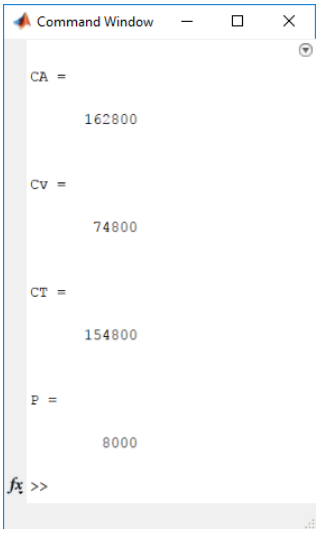
- Profitul întreprinderii se calculează cu o structură secvențială.
- Datele de intrare sunt volumul producției `q`, prețul de vânzare unitar `q`, cheltuielile variabile unitare `v` și cheltuielile fixe `Cf`.
- Urmează apoi o structură secvențială care cuprinde patru blocuri de atribuire prin care se calculează cifra de afaceri `CA=p*q`, cheltuielile variabile `Cv=q*v`, cheltuielile totale `CT=Cf+Cv` și profitul `P=CA-CT`.
- Ordinea de executare a instrucțiunilor din cadrul structurii secvențiale este importantă; calculul profitului se poate face doar după calculul cifrei de afaceri, respectiv a cheltuielilor totale.
- Datele de ieșire ale algoritmului se referă la valoarea profitului `P`.

Fișierul de tip `script` pentru calculul profitului întreprinderii este prezentat în figura 3.35, a). Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei; secțiunea de calcul în care se află instrucțiunile pentru calculul cifrei de afaceri, a cheltuielilor variabile, a cheltuielilor totale și a profitului.

Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.35, b).



a) fișierul `script`



b) rezultatul obținut

Figura 3.35. Fișier `script` pentru calculul profitului întreprinderii.

3.6. PROBLEME - STRUCTURI ALTERNATIVE

Problema 3.11

Se consideră două numere reale `a` și `b`.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a valorii minime `min(a,b)` a celor două numere.

Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazul `a=18` și `b=6`.

Rezolvare

Schema logică este prezentată în figura 3.36.

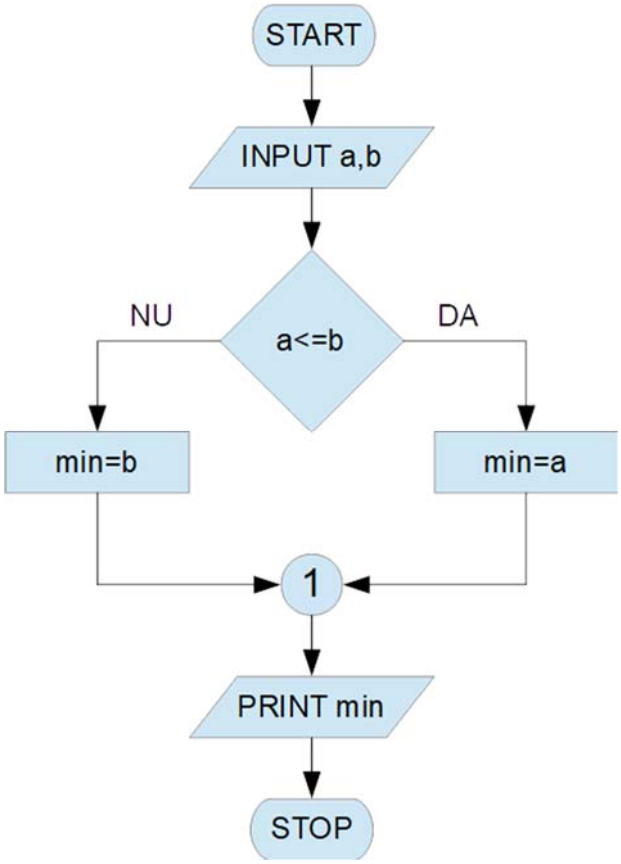


Figura 3.36. Schema logică pentru determinarea valorii minime a două numere.

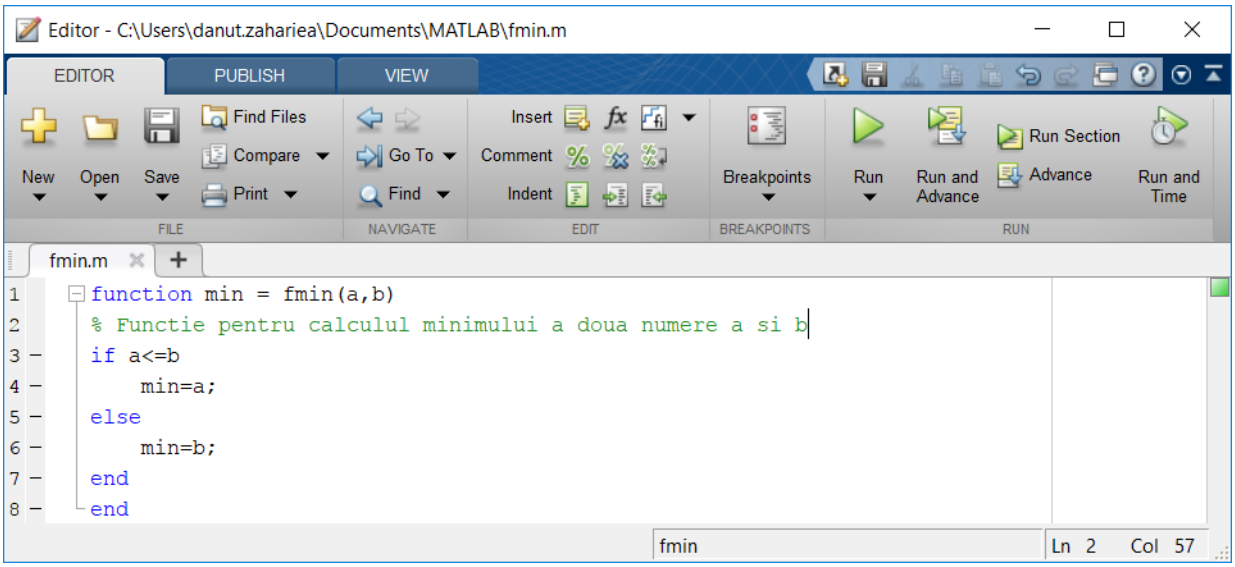
Observații

- Datele de intrare sunt cele două numere reale `a` și `b`.
- După introducerea datelor de intrare, urmează o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `a<=b`. Dacă expresia logică este adevărată atunci minimul celor două numere este `a`, în caz contrar este `b`.
- Pe cele două ramuri ale structurii alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `min` valoarea numărului `a`, sau a numărului `b`, după cum expresia logică `a<=b` este sau nu adevărată.
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea minimă dintre cele două numere.

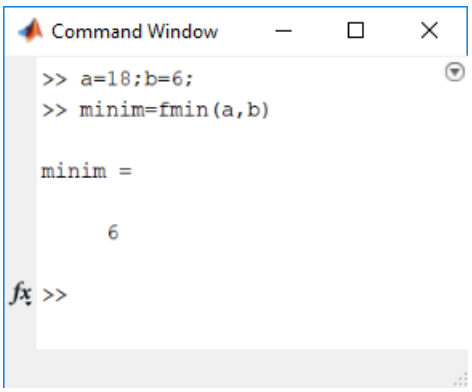
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.37, a). Numele funcției este `fmin` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmin.m`. Funcția acceptă doi parametri de intrare, respectiv numerele `a` și `b`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea minimă atribuită variabilei `f`. Variabila `f` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea prealabilă a celor două numere `x1=18` și `x2=6`. La apelare, cele două numere `x1` și `x2` se transferă parametrilor de intrare ai funcției `a` și `b`.

Rezultatul apelării funcției este prezentat în figura 3.37, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.37. Fișier `function` pentru determinarea valorii minime a două numere.

Problema 3.12

Se consideră două numere reale `a` și `b`.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a valorii maxime `max(a,b)` a celor două numere.

Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazul `a=18` și `b=6`.

Rezolvare

Schema logică este prezentată în figura 3.38.

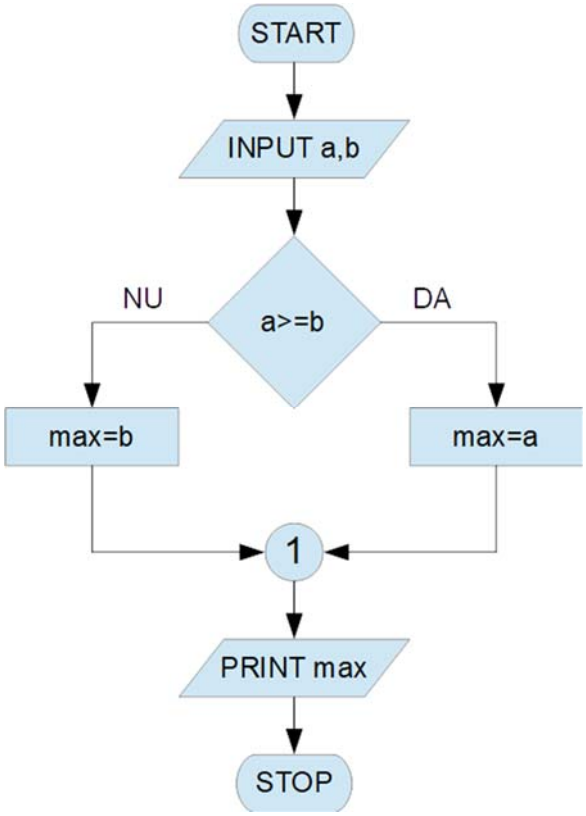


Figura 3.38. Schema logică pentru determinarea valorii maxime a două numere.

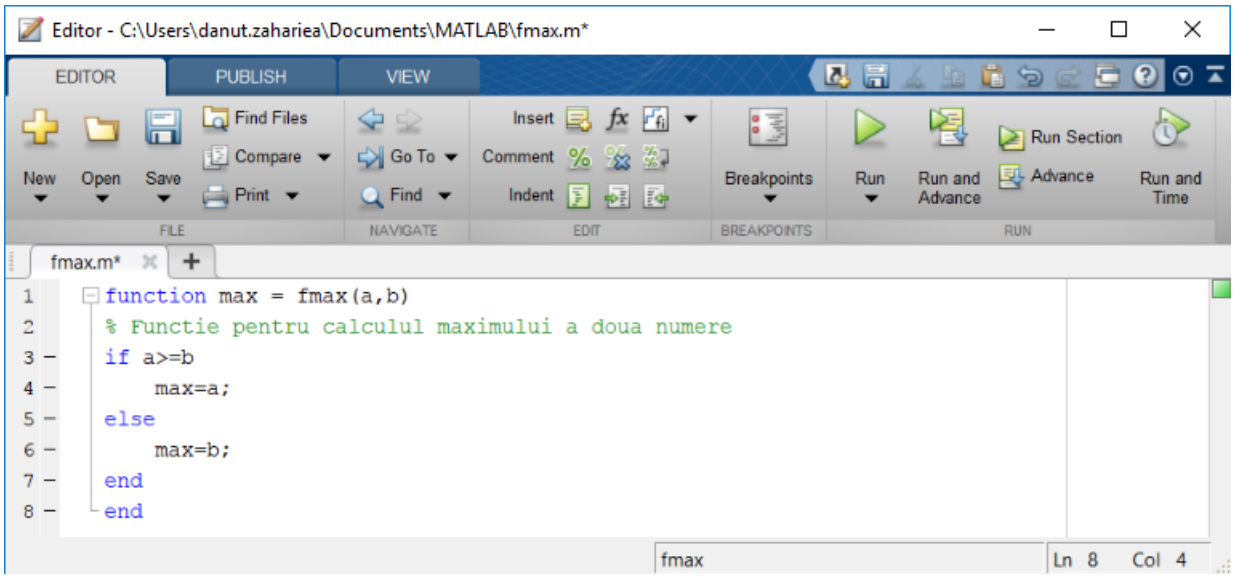
Observații

- Datele de intrare sunt cele două numere reale `a` și `b`.
- După introducerea datelor de intrare, urmează o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `a>=b`. Dacă expresia logică este adevărată atunci maximul celor două numere este `a`, în caz contrar este `b`.
- Pe cele două ramuri ale structurii alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `max` valoarea numărului `a`, sau a numărului `b`, după cum expresia logică `a>=b` este sau nu adevărată.
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea maximă dintre cele două numere.

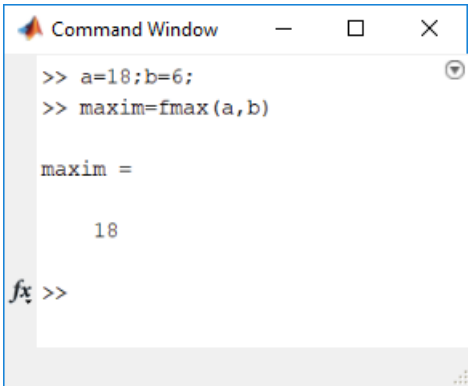
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.39, a). Numele funcției este `fmax` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmax.m`. Funcția acceptă doi parametri de intrare, respectiv numerele `a` și `b`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea maximă atribuită variabilei `max`. Variabila `max` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea prealabilă a celor două numere `a=18` și `b=6`. La apelare, cele două numere `a` și `b` se transferă parametrilor de intrare ai funcției.

Rezultatul apelării funcției este prezentat în figura 3.39, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.39. Fișier `function` pentru determinarea valorii maxime a două numere.

Problema 3.13

Se consideră trei numere reale `a, b` și `c`. Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a valorii minime `min(a,b,c)` a celor trei numere. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazul `a=18, b=6` și `c=14`.

Rezolvare

Schema logică este prezentată în figura 3.40.

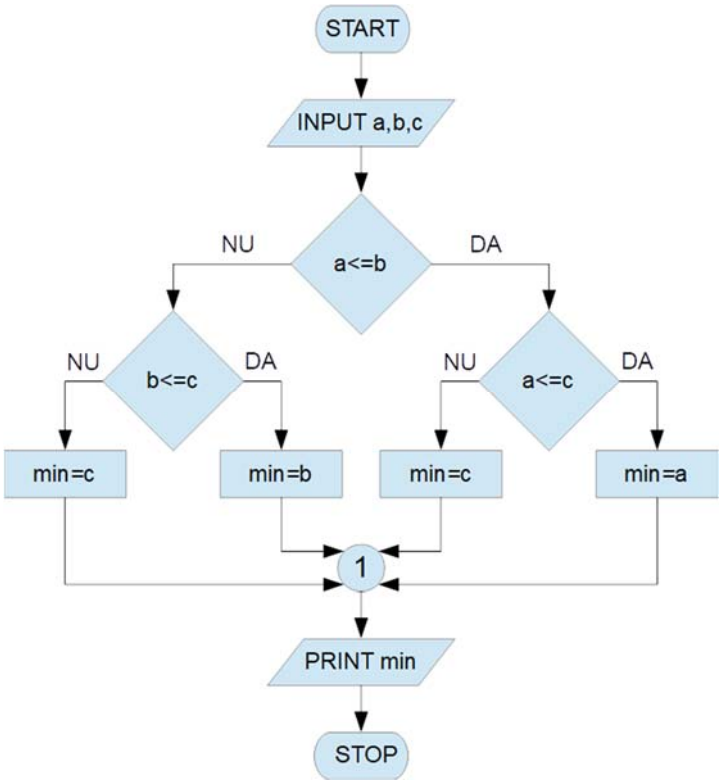


Figura 3.40. Schema logică pentru determinarea valorii minime a trei numere.

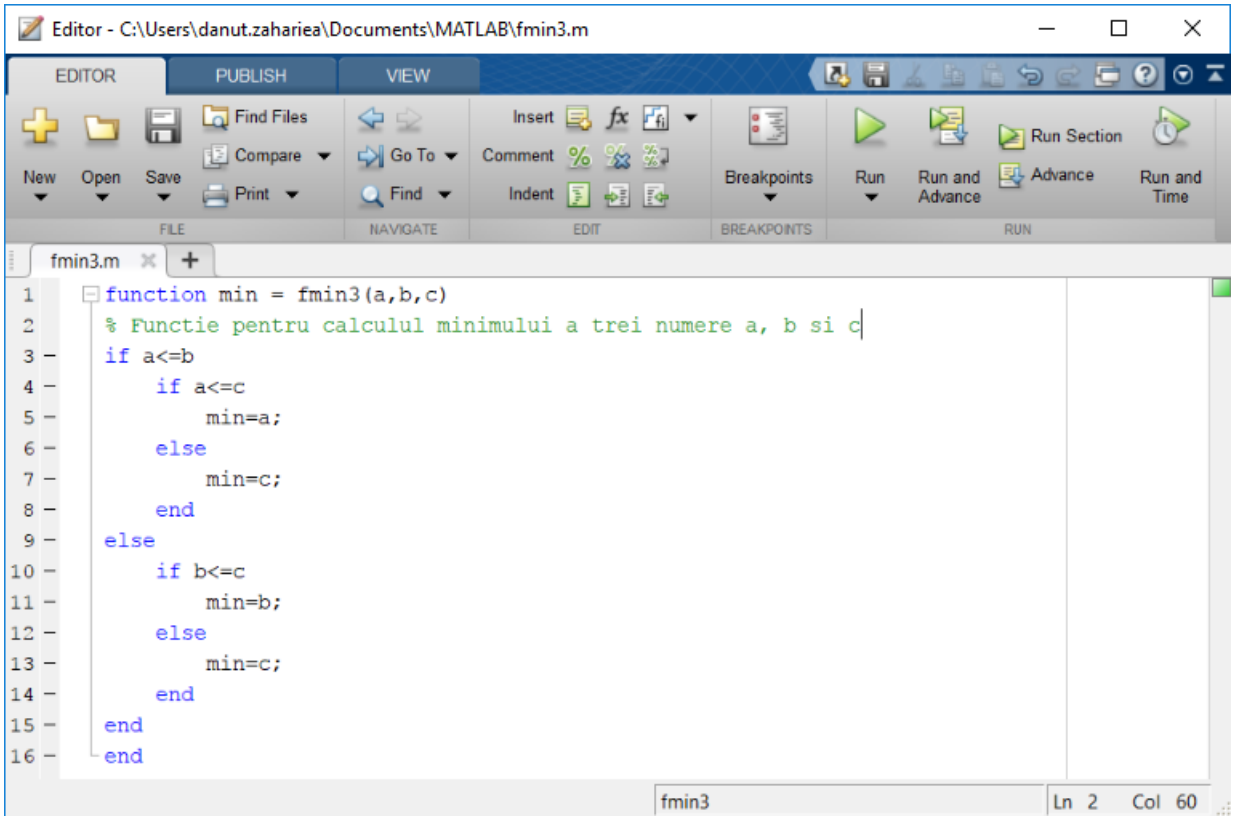
Observații

- Datele de intrare sunt cele trei numere reale `a, b` și `c`.
- După introducerea datelor de intrare, urmează o primă structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `a<=b`. Dacă expresia logică este adevărată atunci se verifică suplimentar, cu o a doua structură alternativă, dacă `a<=c`. Dacă și această expresie logică este adevărată atunci minimul celor trei numere este `a`, în caz contrar este `c`. Pe cele două ramuri ale celei de-a doua structuri alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `min` valoarea numărului `a`, sau a numărului `c`, după cum expresia logică `a<=c` este sau nu adevărată.
- Dacă prima expresie logică `a<=b` este falsă, se verifică suplimentar, cu o a treia structură alternativă, dacă `b<=c`. Dacă această expresie logică este adevărată atunci minimul celor trei numere este `b`, în caz contrar este `c`. Pe cele două ramuri ale celei de-a treia structuri alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `min` valoarea numărului `b`, sau a numărului `c`, după cum expresia logică `b<=c` este sau nu adevărată.

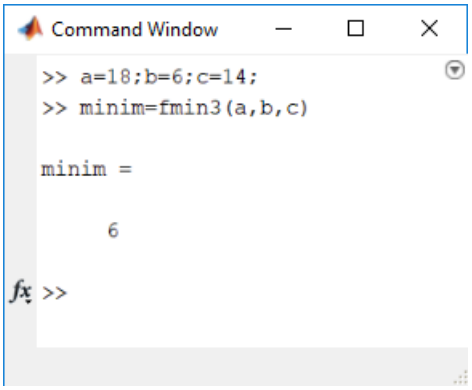
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea minimă dintre cele trei numere.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.41, a). Numele funcției este `fmin3` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmin3.m`. Funcția acceptă trei parametri de intrare, respectiv numerele `a`, `b` și `c`. În corpul funcției se găsesc trei structuri alternative de tip `if-else` prin care se determină valoarea minimă atribuită variabilei `min`. Variabila `min` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea prealabilă a celor două numere `a=18`, `b=6` și `c=14`. La apelare, cele trei numere `a`, `b` și `c` se transferă parametrilor de intrare ai funcției. Rezultatul apelării funcției este prezentat în figura 3.41, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.41. Fișier `function` pentru determinarea valorii minime a trei numere.

Problema 3.14

Se consideră trei numere reale `a`, `b` și `c`. Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a valorii maxime `max(a,b,c)` a celor trei numere. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazul `a=18`, `b=6` și `c=14`.

Rezolvare

Schema logică este prezentată în figura 3.42.

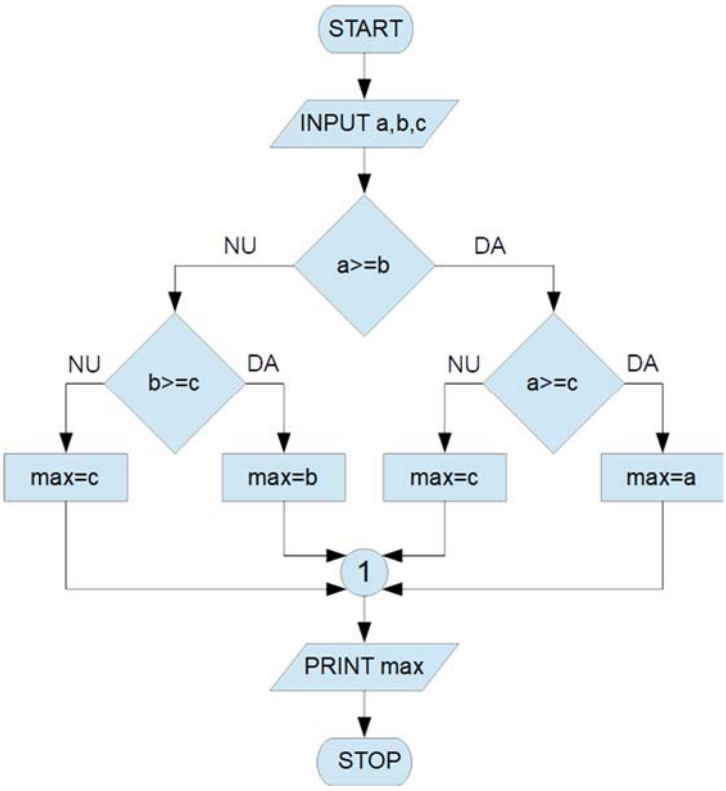


Figura 3.42. Schema logică pentru determinarea valorii maxime a trei numere.

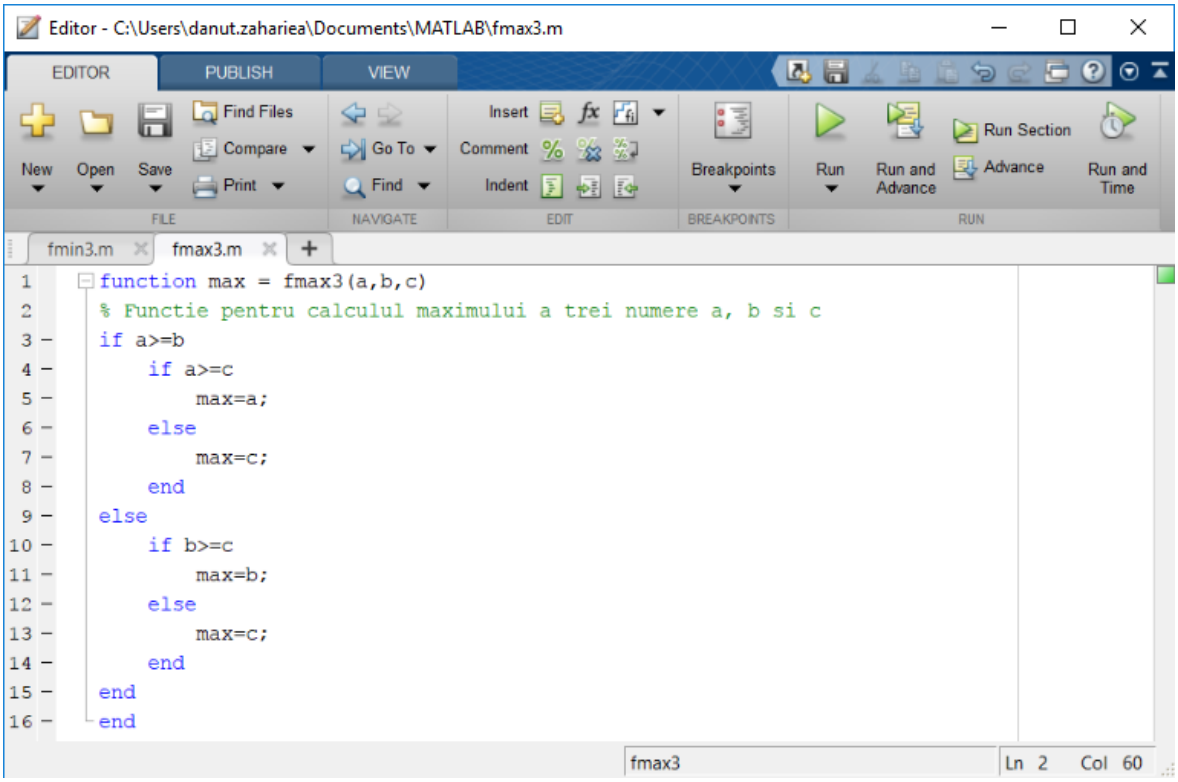
Observații

- Datele de intrare sunt cele trei numere reale `a`, `b` și `c`.
- După introducerea datelor de intrare, urmează o primă structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `a>=b`. Dacă expresia logică este adevărată atunci se verifică suplimentar, cu o a doua structură alternativă, dacă `a>=c`. Dacă și această expresie logică este adevărată atunci maximul celor trei numere este `a`, în caz contrar este `c`. Pe cele două ramuri ale celei de-a doua structuri alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `max` valoarea numărului `a`, sau a numărului `c`, după cum expresia logică `a>=c` este sau nu adevărată.
- Dacă prima expresie logică `a>=b` este falsă, se verifică suplimentar, cu o a treia structură alternativă, dacă `b>=c`. Dacă această expresie logică este adevărată atunci maximul celor trei numere este `b`, în caz contrar este `c`. Pe cele două ramuri ale celei de-a treia structuri alternative se găsește câte o comandă de atribuire prin care se transferă variabilei `max` valoarea numărului `b`, sau a numărului `c`, după cum expresia logică `b>=c` este sau nu adevărată.

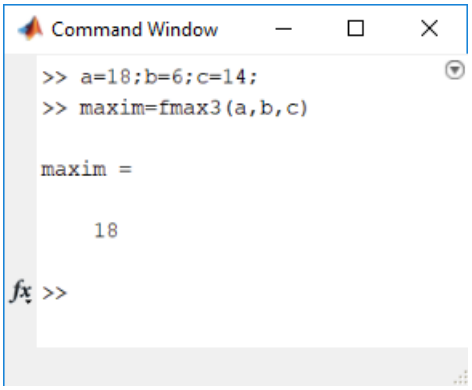
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea maximă dintre cele două numere.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.43, a). Numele funcției este `fmax3` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmax3.m`. Funcția acceptă trei parametri de intrare, respectiv numerele `a`, `b` și `c`. În corpul funcției se găsesc trei structuri alternative de tip `if-else` prin care se determină valoarea maximă atribuită variabilei `max`. Variabila `max` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea prealabilă a celor două numere `a=18`, `b=6` și `c=14`. La apelare, cele trei numere `a`, `b` și `c` se transferă parametrilor de intrare ai funcției. Rezultatul apelării funcției este prezentat în figura 3.43, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.43. Fișier `function` pentru determinarea valorii maxime a trei numere.

Problema 3.15

Să se realizeze o schemă logică pentru definirea simbolului lui Kronecker:

$$\delta_{ij} = \begin{cases} 1, \text{dacă } i = j \\ 0, \text{dacă } i \neq j \end{cases}$$

Să se definească un fișier de tip `function` pentru implementarea schemei logice. Să se verifice pentru cazul `i=1` și `j=3`.

Rezolvare

Schema logică este prezentată în figura 3.44.

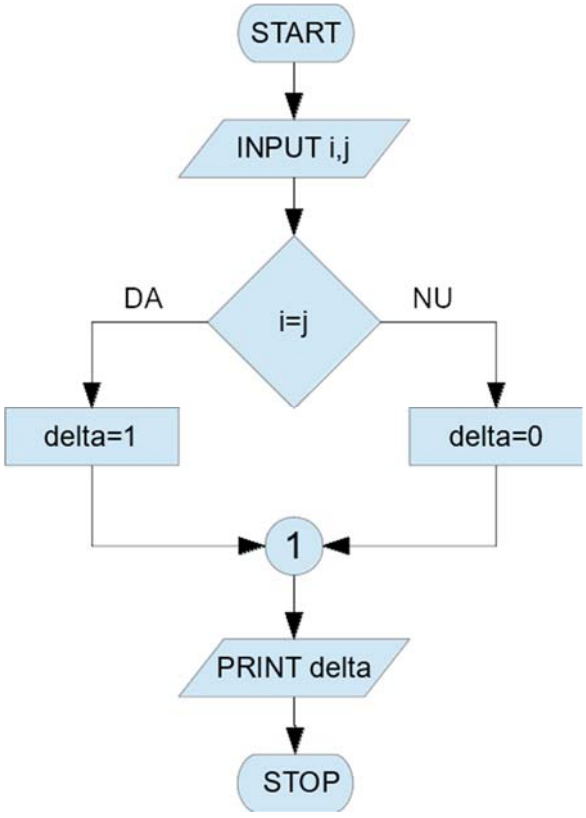


Figura 3.44. Schema logică pentru definirea simbolului lui Kronecker.

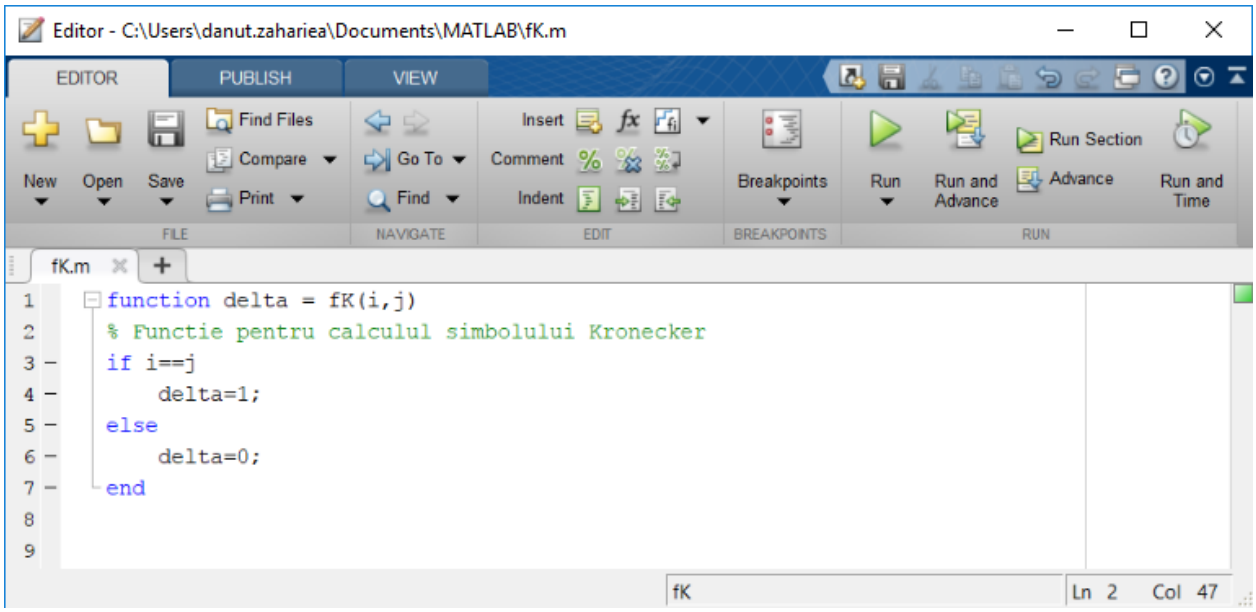
Observații

- Datele de intrare sunt cele două numere reale `i` și `j`.
- După introducerea datelor de intrare, urmează o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a egalității `i=j`. Dacă expresia logică este adevărată atunci simbolul lui Kronecker va avea valoarea `delta=1`. În caz contrar, simbolul lui Kronecker va avea valoarea `delta=0`.
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea simbolului lui Kronecker, `delta`.

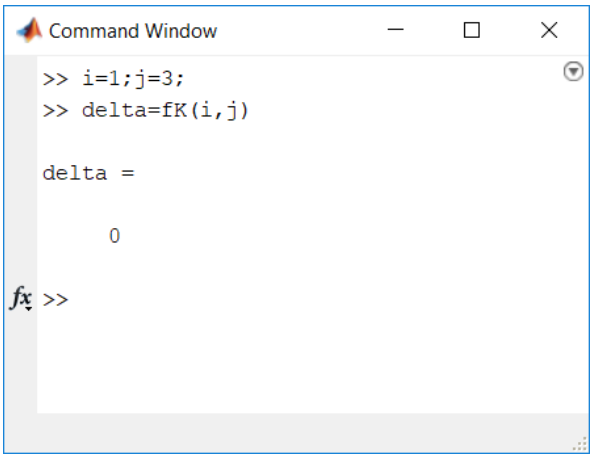
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.45, a). Numele funcției este `fK` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fK.m`. Funcția acceptă doi parametri de intrare, respectiv numerele `i` și `j`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea simbolului lui Kronecker atribuită variabilei `delta`. Variabila `delta` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea prealabilă a celor două numere `i=1` și `j=3`. La apelare, cele două numere `i` și `j` se transferă parametrilor de intrare ai funcției, având aceleași notații `i` și `j`.

Rezultatul apelării funcției este prezentat în figura 3.45, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.45. Fișier `function` pentru definirea simbolului lui Kronecker.

Problema 3.16

Să se realizeze o schemă logică pentru definirea funcției modul:

$$|x| = \begin{cases} x, \text{dacă } x \geq 0 \\ -x, \text{dacă x} < 0 \end{cases}$$

Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazul `x=1` și `x=-1`.

Rezolvare

Schema logică este prezentată în figura 3.46.

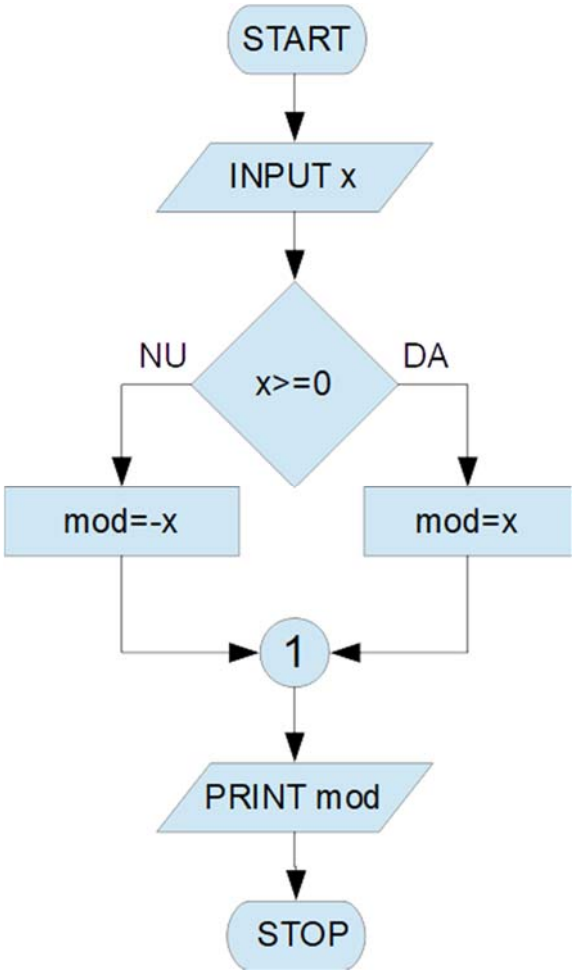


Figura 3.46. Schema logică pentru definirea funcției modul.

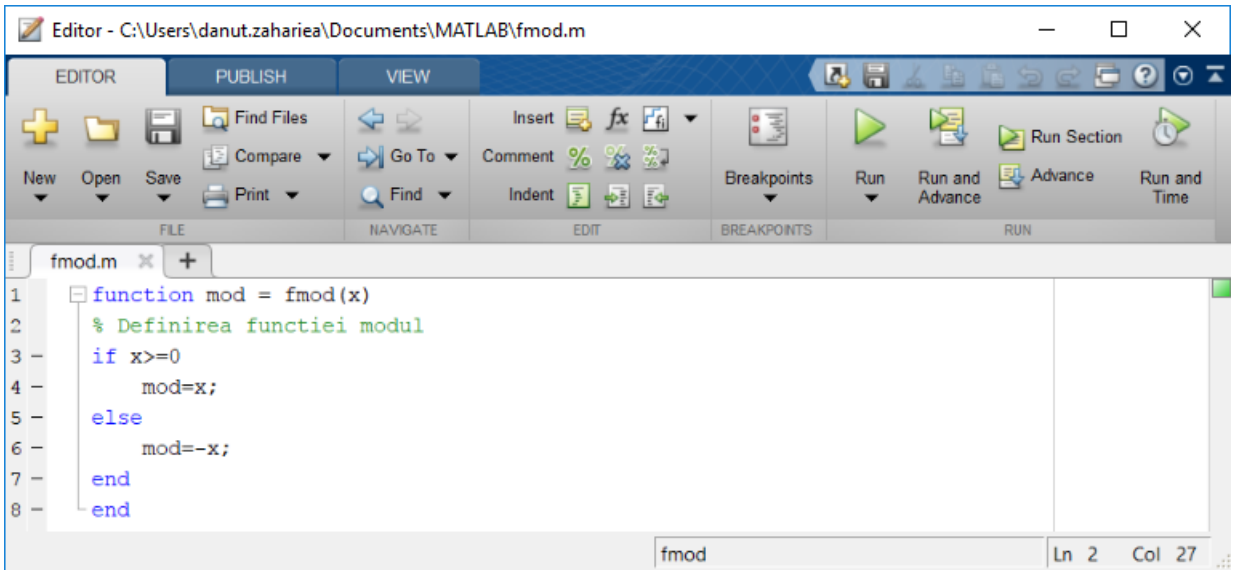
Observații

- Datele de intrare se referă la valoarea numărului `x`.
- După introducerea numărului `x`, urmează o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a egalității `x>=0`. Dacă expresia logică este adevărată atunci funcția modul va avea valoarea `mod=x`. În caz contrar, funcția modul va avea valoarea `mod=-x`.
- După părăsirea structurii alternative se prezintă rezultatul final al algoritmului, respectiv valoarea funcției modul, `mod`.

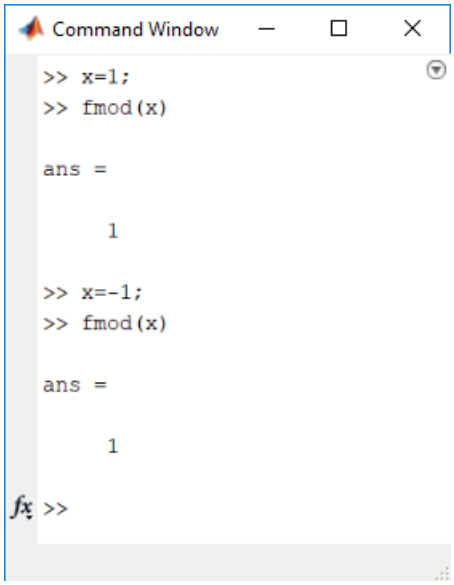
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.47, a). Numele funcției este `fmod` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmod.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea funcției modul atribuită variabilei `mod`. Variabila `mod` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, în mod repetat pentru fiecare valoare dorită a numărului, `x=1`, respectiv `x=-1`. La apelare, numărul `x` se transferă parametrului de intrare al funcției.

Rezultatul apelării funcției este prezentat în figura 3.47, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.47. Fișier `function` pentru definirea funcției modul.

Problema 3.17

Să se realizeze o schemă logică pentru definirea funcției semn (signum):

$$\text{sgn}(x) = \begin{cases} -1, \text{dacă } x < 0 \\ 0, \text{dacă } x = 0 \\ +1, \text{dacă } x > 0 \end{cases}$$

Să se definească un fișier de tip `function` pentru implementarea schemei logice. Să se verifice pentru cazul `x=2`, `x=0` și `x=-2`.

Rezolvare

Schema logică este prezentată în figura 3.48.

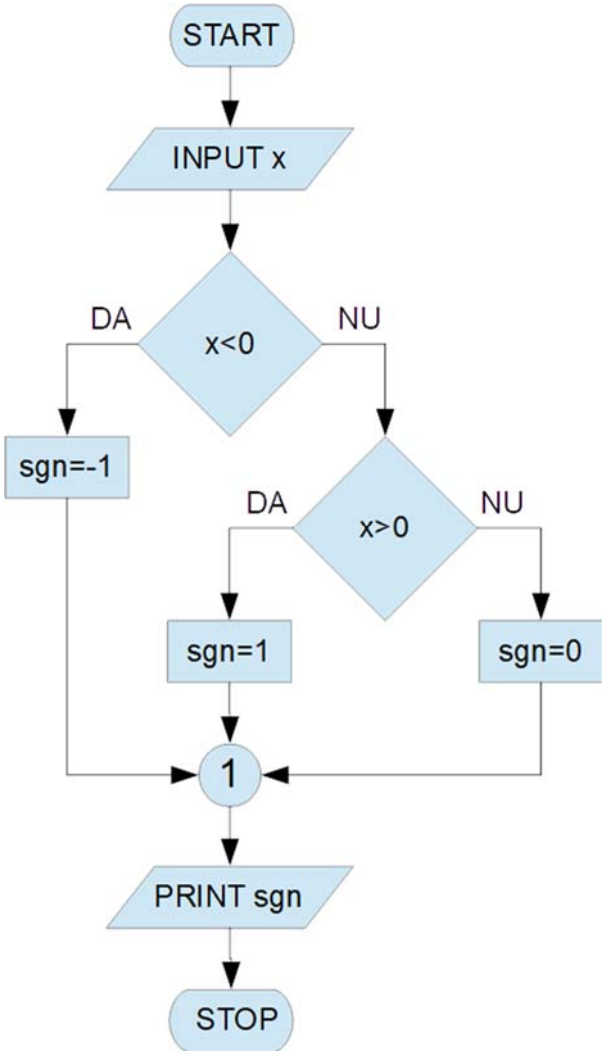


Figura 3.48. Schema logică pentru definirea funcției semn.

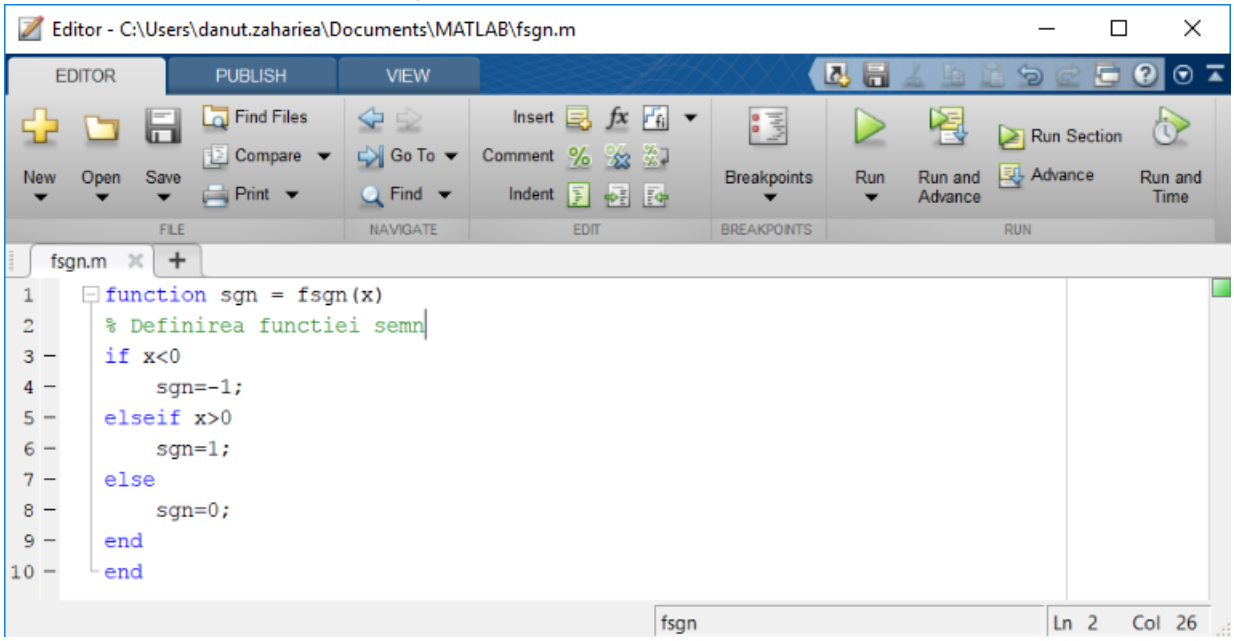
Observații

- După introducerea datelor de intrare, în acest caz valoarea numărului `x`, urmează prima structura alternativă cu două ramuri care verifică dacă `x<0`.
- Dacă expresia logică `x<0` este adevărată, atunci funcția semn capătă valoarea `sgn=-1`.
- În caz contrar, algoritmul verifică dacă `x>0` prin intermediul celei de-a doua structuri alternative.
- Dacă expresia logică `x>0` este adevărată, atunci semn capătă valoarea `sgn=1`.
- În caz contrar, adică pentru `x=0`, funcția semn capătă valoarea `sgn=0`.
- Indiferent însă de valoarea numărului `x`, în urma structurilor alternative se obține o anumită valoare pentru parametrul de ieșire al algoritmului, respectiv funcția semn, `sgn`.

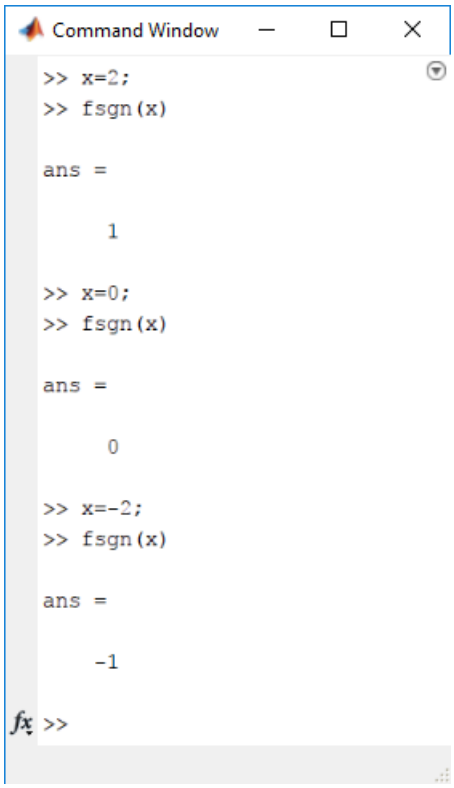
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.49, a). Numele funcției este `fsgn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsgn.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`. În corpul funcției se găsește o structură alternativă de tip `if-elseif-else` prin care se determină valoarea funcției semn atribuită variabilei `sgn`. Variabila `sgn` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, în mod repetat pentru fiecare valoare dorită a numărului, `x=2`, `x=0`, respectiv `x=-2`. La apelare, numărul `x` se transferă parametrului de intrare al funcției.

Rezultatul apelării funcției este prezentat în figura 3.49, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.49. Fișier `function` pentru definirea funcției semn.

Problema 3.18

Se consideră ecuația de gradul 1 cu coeficienți reali:

$$ax + b = 0$$

Să se realizeze schema logică pentru rezolvarea ecuației.

Să se definească un fișier de tip `function` care să rezolve problema ecuației de gradul 1. Să se verifice pentru cazurile:

- `a=0`, `b=0`
- `a=0`, `b=2`
- `a=4`, `b=2`

Observații

a) Dacă $a \neq 0$, atunci ecuația admite soluția unică $x = -b/a$.
b) Dacă $a = 0$ și $\mathrm{b} = 0$, atunci ecuația este nedeterminată.
c) Dacă $a = 0$ și $b \neq 0$, atunci ecuația este incompatibilă.

Rezolvare

Schema logică este prezentată în figura 3.50.

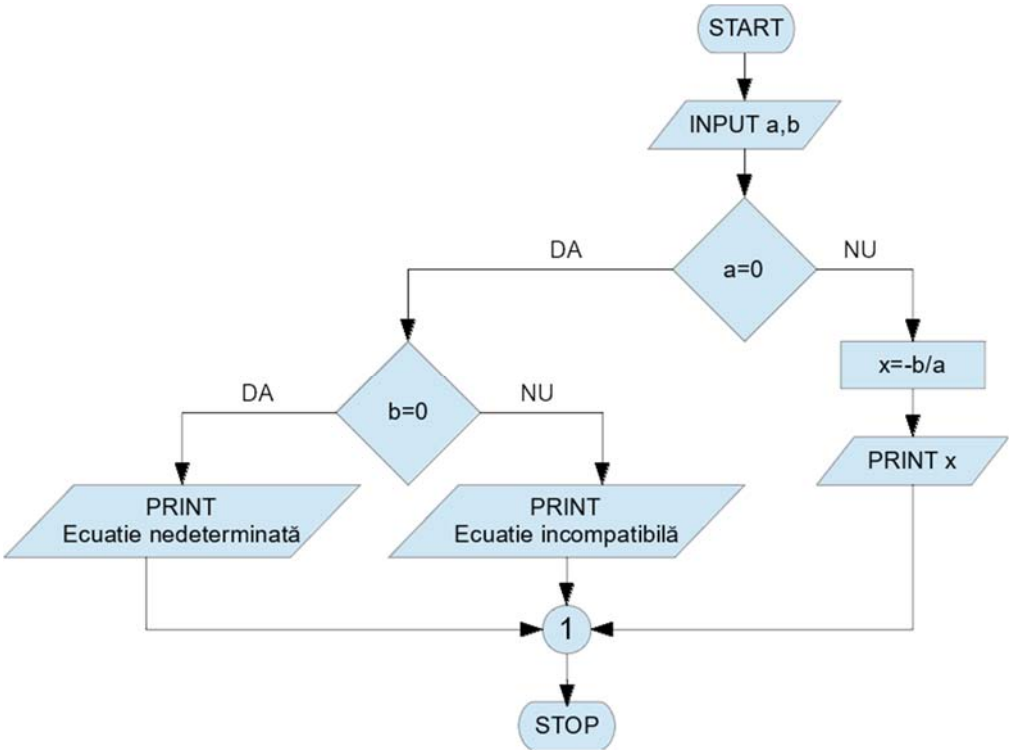


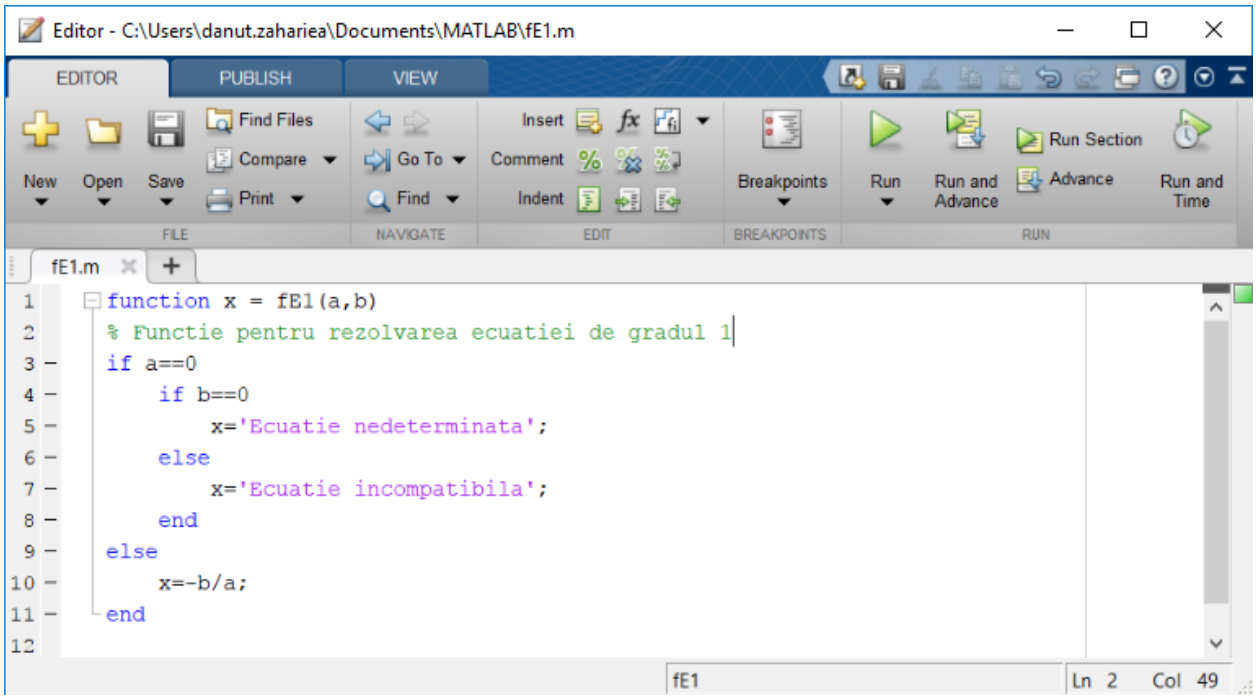
Figura 3.50. Schema logică pentru rezolvarea ecuației de gradul 1.

Observații

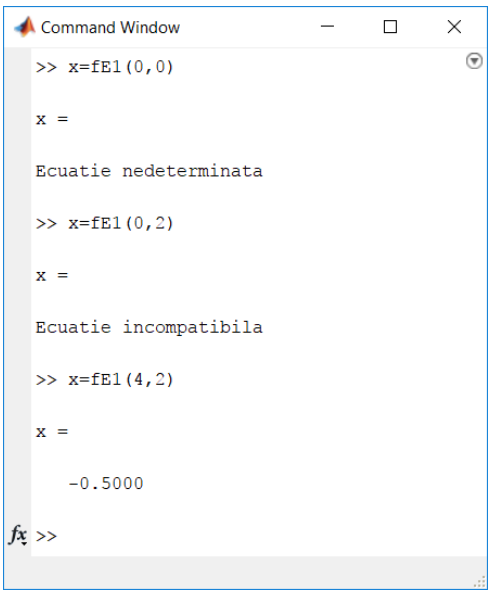
- După introducerea datelor de intrare `a` și `b`, urmează o structură alternativă cu două ramuri.
- Expresia logică urmărește determinarea valorii de adevăr a egalității `a=0`.
- Dacă expresia logică este falsă atunci se calculează soluția unică a ecuației `x=-b/a`. În caz contrar, se evaluează cea de-a doua expresie logică, `b=0`. În funcție de valoarea de adevăr a expresiei `b=0`, ecuația poate fi nedeterminată, respectiv incompatibilă.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.51, a). Numele funcției este `fE1` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fE1.m`. Funcția acceptă doi parametri de intrare, respectiv numerele `a` și `b`. În corpul funcției se găsesc două structuri alternative de tip `if-else`, una în interiorul celeilalte, prin care se determină soluția ecuației de gradul 1, atribuită variabilei `x`. Variabila `x` este și parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare din perechile de valori `a` și `b` dorite. La apelare, cele două numere se transferă parametrilor de intrare ai funcției, `a` și `b`. Rezultatul apelării funcției este prezentat în figura 3.51, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.51. Fișier `function` pentru rezolvarea ecuației de gradul 1.

Problema 3.19

Se consideră ecuația de gradul 2 cu coeficienți reali, $a \neq 0$:

$$ax^2 + bx + c = 0$$

Să se realizeze schema logică pentru rezolvarea ecuației.

Să se definească un fișier de tip `function` care să rezolve problema ecuației de gradul 2. Să se verifice pentru cazurile:

- `a=2, b=5, c=2`
- `a=2, b=4, c=2`
- `a=2, b=4, c=3`

Observații

Se calculează discriminantul $\Delta = b^2 - 4ac$

- Dacă $\Delta \geq 0$, atunci ecuația admite soluțiile reale $x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$.
- Dacă $\Delta < 0$ atunci ecuația admite soluțiile complexe $x_{1,2} = \frac{-b \pm i\sqrt{|\Delta|}}{2a}$.

Rezolvare

Schema logică este prezentată în figura 3.52.

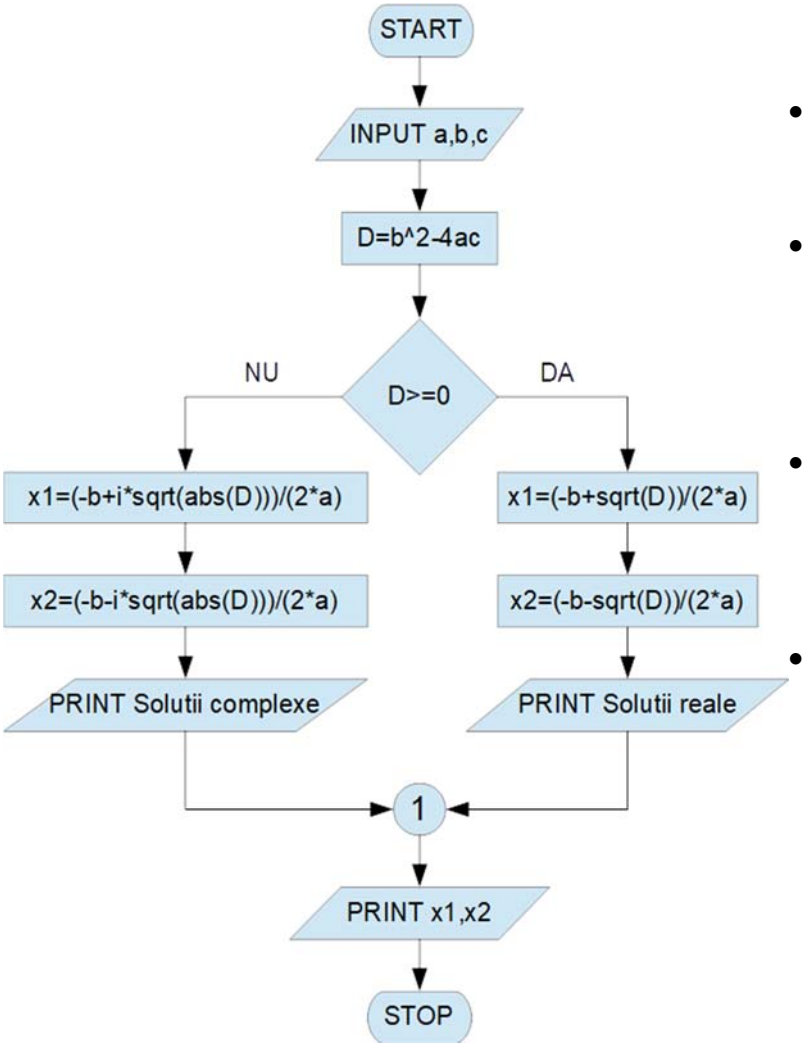


Figura 3.52. Schema logică pentru rezolvarea ecuației de gradul 2.

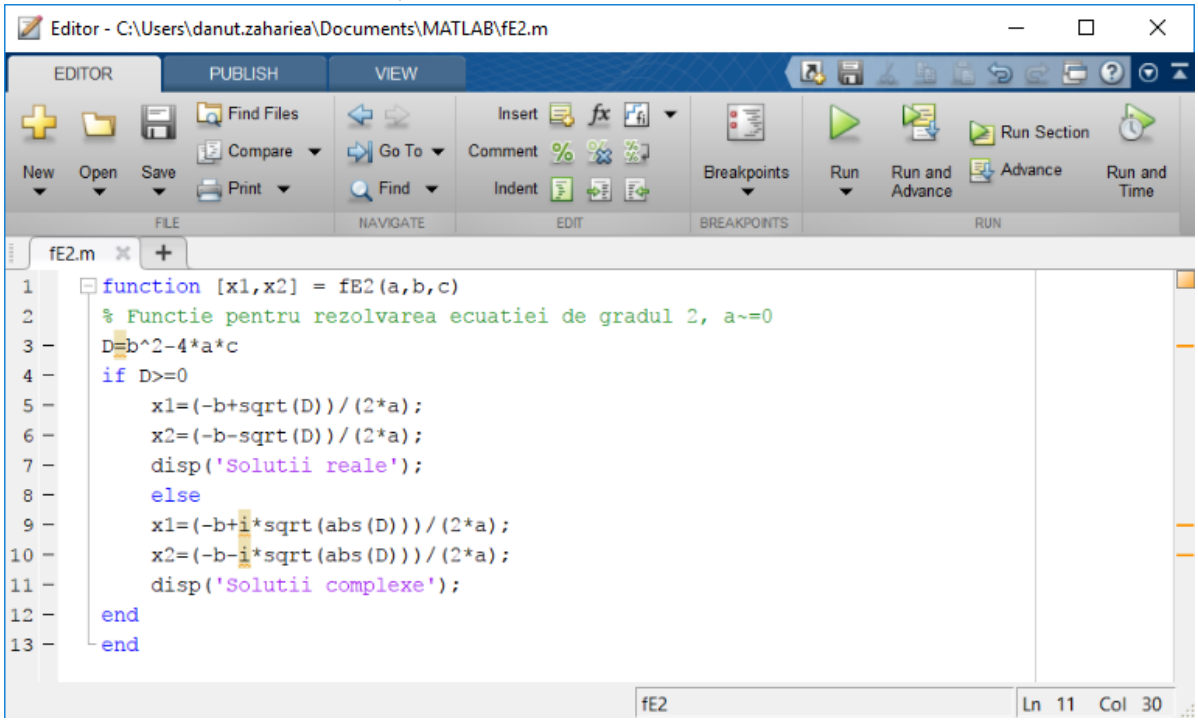
Observații

- După introducerea datelor de intrare `a`, `b` și `c`, se calculează determinantul `D`.
- Urmează apoi o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `D>=0`.
- Dacă expresia logică este adevărată atunci se calculează soluțiile reale ale ecuației, $x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$.
- În caz contrar, se calculează soluțiile complexe ale ecuației, $x_{1,2} = \frac{-b \pm i\sqrt{|\Delta|}}{2a}$.

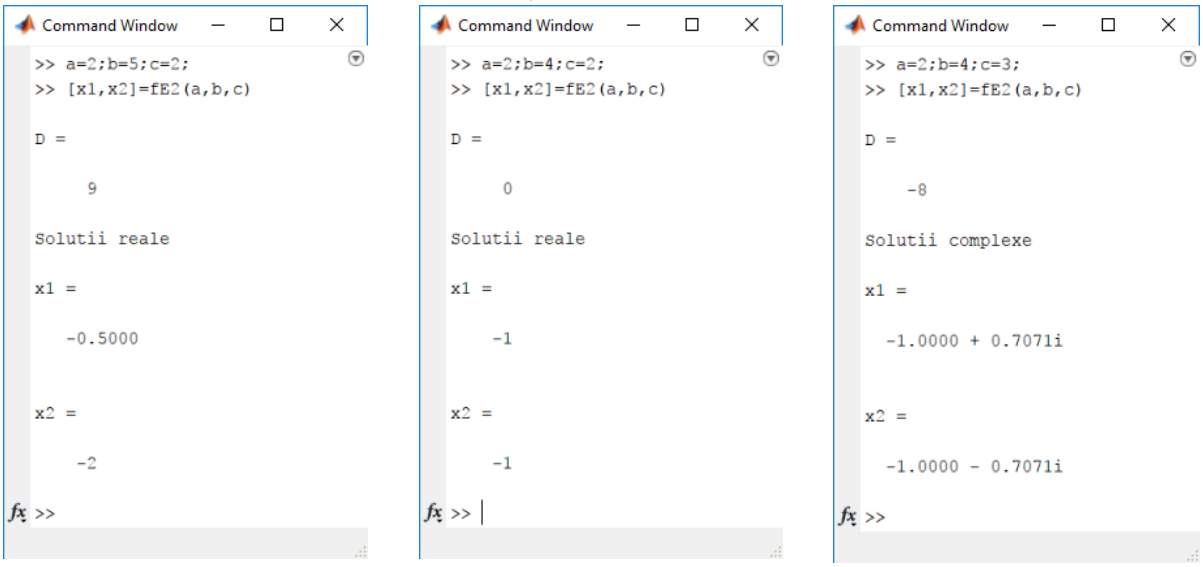
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.53, a). Numele funcției este `fE2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fE2.m`. Funcția acceptă trei parametri de intrare, respectiv numerele `a`, `b` și `c` cu ajutorul cărora se calculează determinantul `D`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care, în funcție de valoarea determinantului, se determină soluțiile reale (`D>=0`), respectiv complexe (`D<0`) ale ecuației de gradul 2, atribuite variabilelor `x1` și `x2`. Variabilele `x1` și `x2` sunt și parametrii de ieșire ai funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare din perechile de valori `a`, `b` și `c` dorite. La apelare, cele trei numere se transferă parametrilor de intrare ai funcției, `a`, `b` și `c`.

Rezultatul apelării funcției este prezentat în figura 3.53, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.53. Fișier `function` pentru rezolvarea ecuației de gradul 2.

Problema 3.20

Se consideră un punct P dat prin coordonatele (x,y), x≠0, y≠0.

Să se realizeze o schemă logică care să determine cadranul în care se află punctul respectiv, figura 3.54.

Pe baza schemei logice să se rezolve problema în MATLAB folosind metoda fișierelor `script`, respectiv metoda funcțiilor definite în fișiere de tip `function`.

Să se verifice pentru punctele: P1(2,4); P2(3,-5); P3(-3,1); P4(-5,-3).

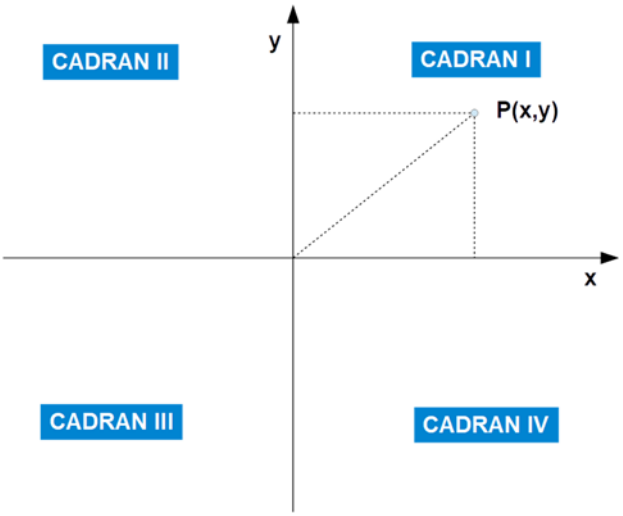


Figura 3.54. cadranele cercului trigonometric.

Rezolvare

Schema logică este prezentată în figura 3.55.

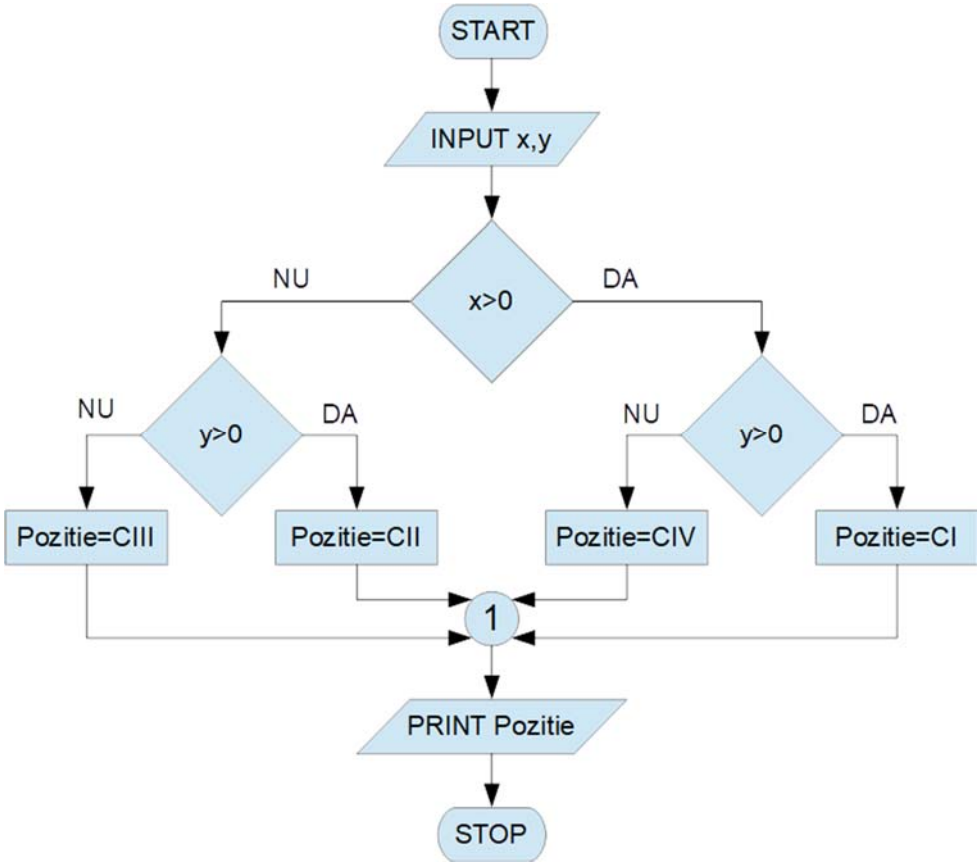


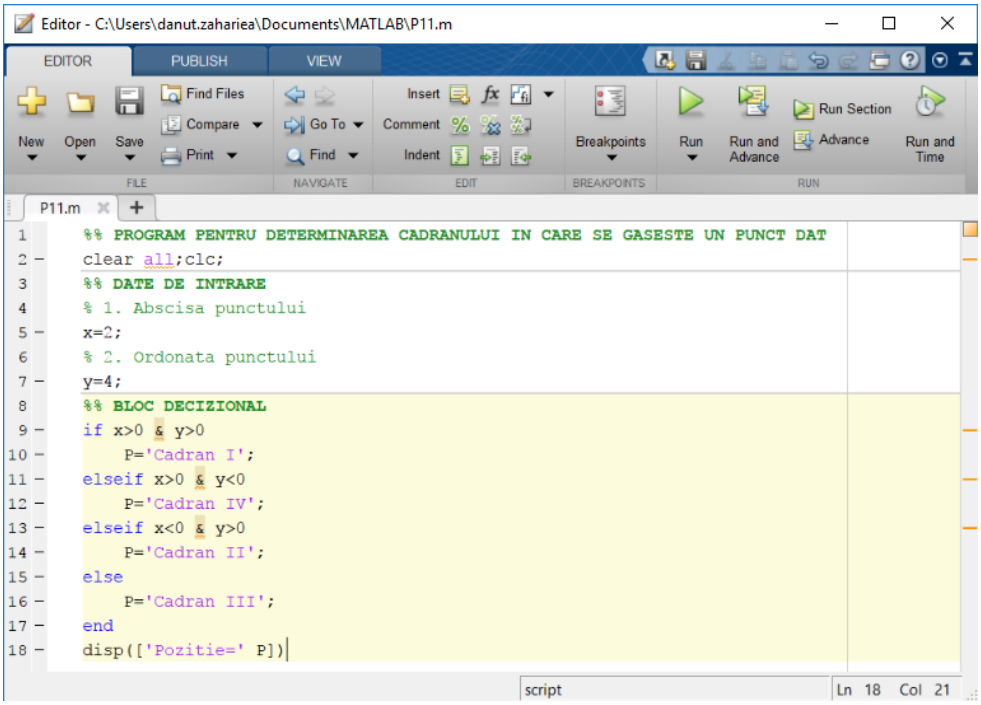
Figura 3.55. Schema logică pentru determinarea cadranului în care se află punctul P(x,y).

Observații

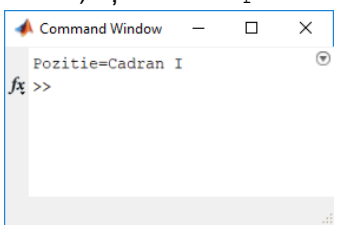
- Datele de intrare sunt abscisă `x` și ordonata `y`.
- Urmează structura alternativă principală, cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `x>0`.
- Dacă expresia logică este adevărată, atunci se verifică suplimentar condiția `y>0` (structură alternativă secundară). Dacă și această condiție este adevărată, atunci punctul se găsește în cadranul I. Se definește o variabilă `Poziție` care capătă valoarea `CI`. În caz contrar, punctul se află în cadranul IV și `Poziție=CIV`.
- În mod similar se verifică și cadranele II și III.
- După părăsirea structurilor alternative se prezintă rezultatul final al algoritmului, respectiv valoarea variabile `Poziție`.

Fișierul de tip `script` pentru determinarea cadranului în care se află punctul P(x,y) este prezentat în figura 3.56, a).

Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei (abscisa și ordonata punctului); secțiunea blocului decizional care verifică și stabilește cadranul corespunzător în funcție de semnul abscisei și ordonatei. Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.56, b).



a) fișierul `script`

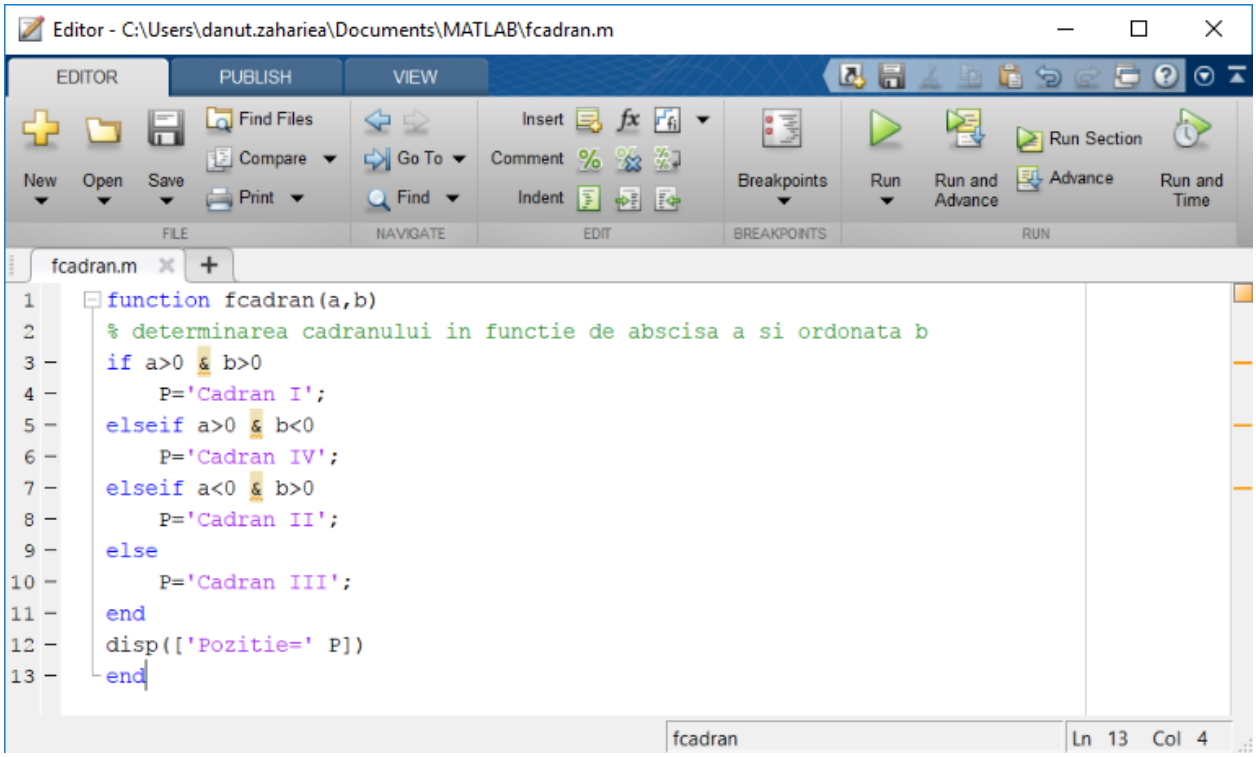


b) rezultatul obținut

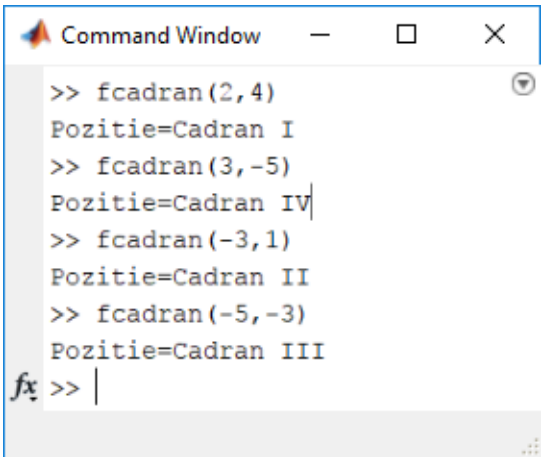
Figura 3.56. Fișier `script` pentru determinarea cadranului în care se află un punct oarecare `P(x,y)`.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.57, a). Numele funcției este `fcadran` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fcadran.m`. Funcția acceptă doi parametri de intrare, respectiv abscisa `a` și ordonata `b`. În corpul funcției se găsește o structură alternativă care verifică și stabilește cadranul corespunzător în funcție de semnul abscisei și ordonatei. Variabila `P` este o variabilă locală. Funcția nu are parametri de ieșire.

Funcția se apelează, în acest caz din fereastra de comenzi, în mod repetat, pentru fiecare din perechile de valori `a` și `b` dorite. La apelare, cele două numere se transferă parametrilor de intrare ai funcției, `a` și `b`. Rezultatul apelării funcției este prezentat în figura 3.57, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.57. Fișier `function` pentru determinarea cadranului în care se află un punct oarecare P(x,y).

Problema 3.21

Se consideră funcția $f: R \to R$ definită prin:

$$f(x) = \begin{cases} x^2 + 1, \text{pentru } x \geq 1 \\ \dfrac{1}{x^2 + 1}, \text{pentru } x < 1 \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$ într-un singur punct al domeniului său de definiție. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazurile `x=3` și `x=-3`.

Rezolvare

Schema logică este prezentată în figura 3.58.

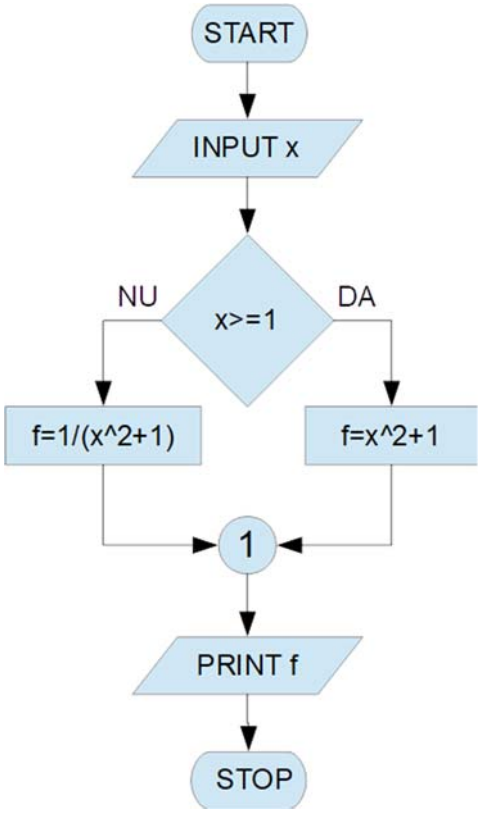


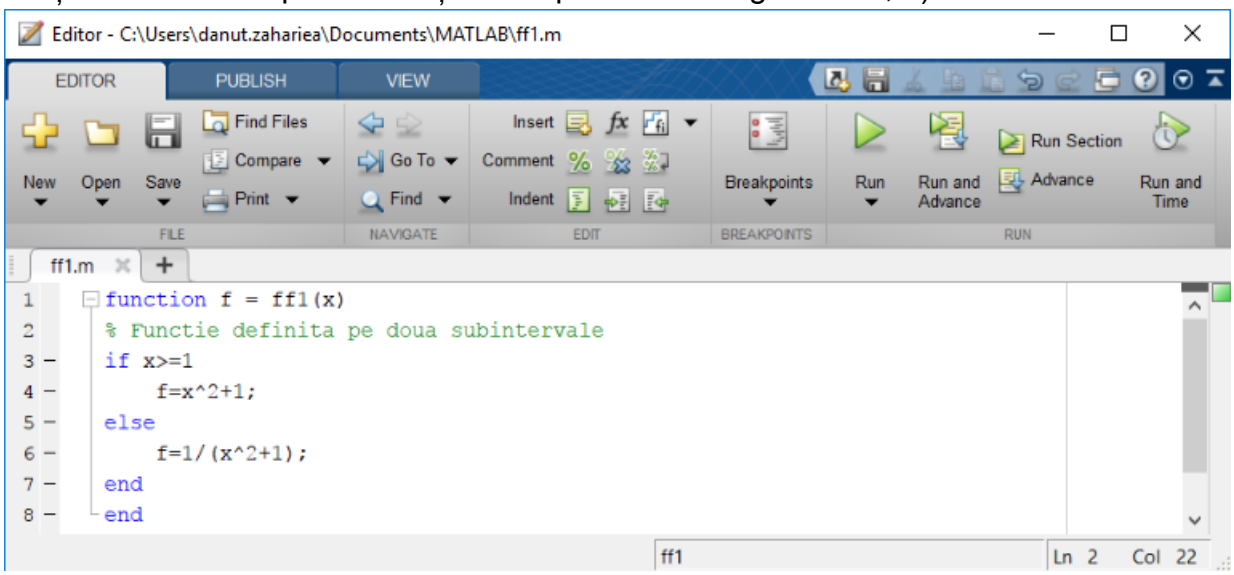
Figura 3.58. Schema logică pentru evaluarea unei funcții definite pe două subintervale.

Observații

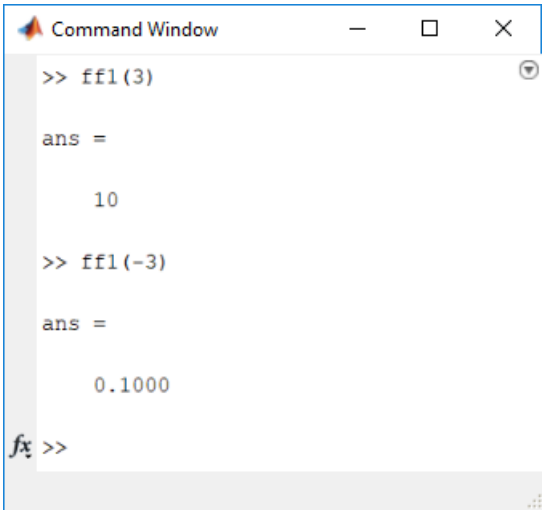
- Pentru evaluarea funcției `f` se utilizează o structură alternativă cu două ramuri.
- După introducerea datelor de intrare, în acest caz valoarea numărului `x`, urmează structura alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `x>=1`.
- Dacă expresia logică este adevărată, atunci funcția `f` se calculează cu relația $f(x) = x^2 + 1$.
- În caz contrar, funcția `f` se calculează cu relația $f(x) = \frac{1}{x^2+1}$.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.59, a). Numele funcției este `ff1` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff1.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`, cu ajutorul căruia se va evalua funcția `f`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care, în funcție de valoarea numărului `x`, se va utiliza una din cele două relații de calcul. Rezultatul obținut în urma evaluării funcției `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare valoare `x` dorită. La apelare, numărul `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.59, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.59. Fișier `function` pentru evaluarea unei funcții definite pe două subintervale.

Problema 3.22

Se consideră funcția $f: R \to R$ definită prin:

$$f(x) = \begin{cases} x \cdot \sin\left(\frac{1}{x}\right), \text{pentru } x > 0 \\ x^3 + x + 1, \text{pentru } x \leq 0 \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$ într-un singur punct al domeniului său de definiție. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazurile `x=3` și `x=-3`.

Rezolvare

Schema logică este prezentată în figura 3.60.

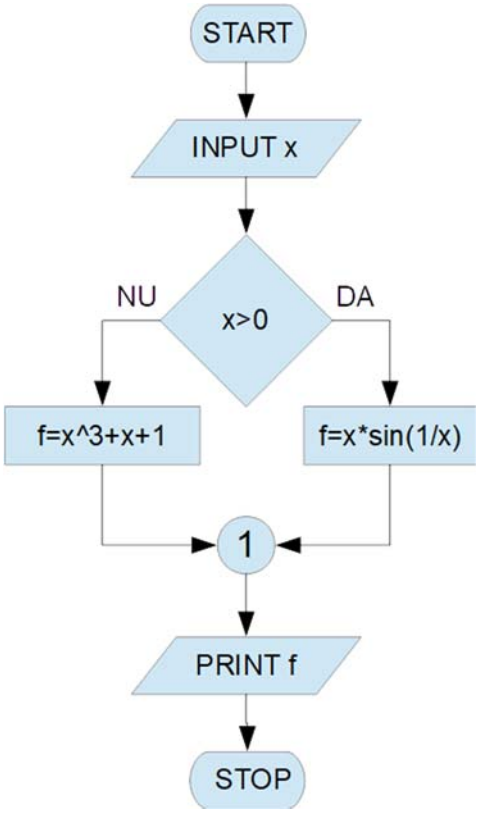


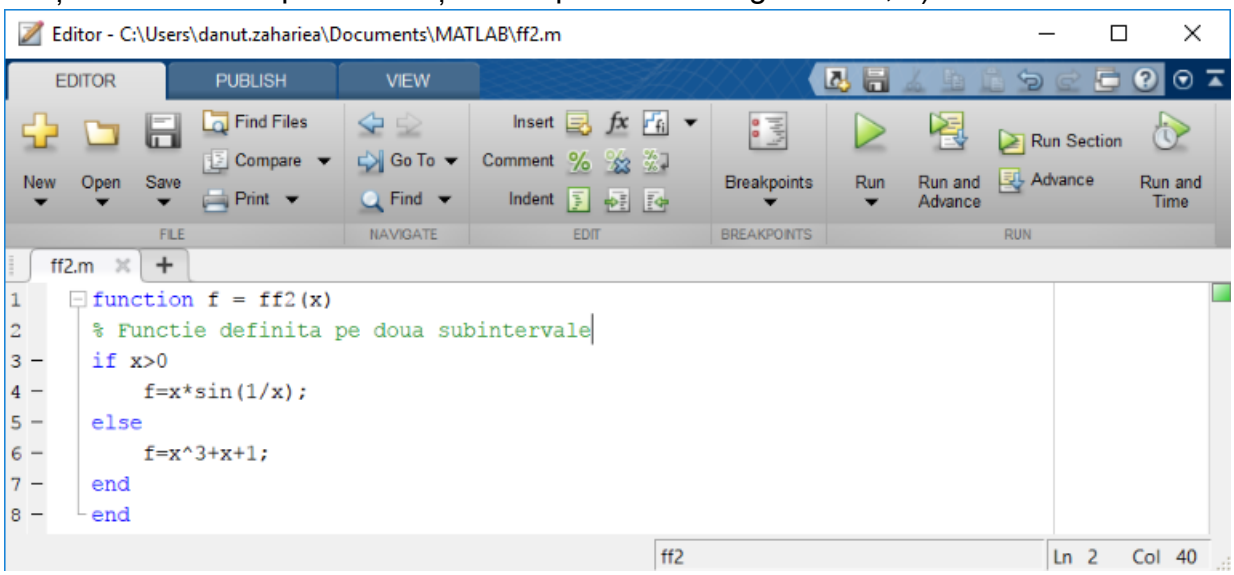
Figura 3.60. Schema logică pentru evaluarea unei funcții definite pe două subintervale.

Observații

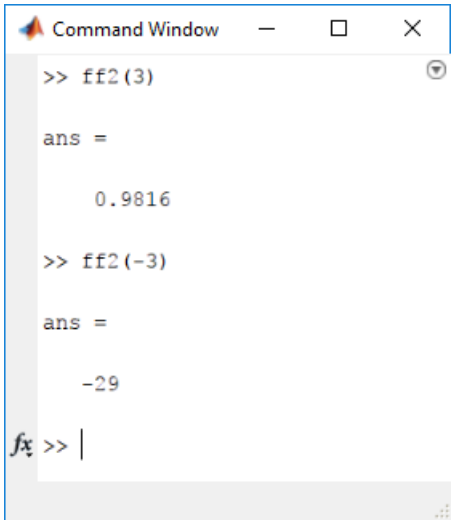
- Pentru evaluarea funcției `f` se utilizează o structură alternativă cu două ramuri.
- După introducerea datelor de intrare, în acest caz valoarea numărului `x`, urmează structura alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a inegalității `x>0`.
- Dacă expresia logică este adevărată, atunci funcția `f` se calculează cu relația $f(x) = x \sin\left(\frac{1}{x}\right)$.
- În caz contrar, funcția `f` se calculează cu relația $f(x) = x^3 + x + 1$

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.61, a). Numele funcției este `ff2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff2.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`, cu ajutorul căruia se va evalua funcția `f2`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care, în funcție de valoarea numărului `x`, se va utiliza una din cele două relații de calcul. Rezultatul obținut în urma evaluării funcției `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare valoare `x` dorită. La apelare, numărul `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.61, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.61. Fișier `function` pentru evaluarea unei funcții definite pe două subintervale.

Problema 3.23

Se consideră funcția $f: R \to R$ definită prin:

$$f(x) = \begin{cases} e^x, \text{pentru } x > 0 \\ e, \text{pentru } x = 0 \\ e^{-x}, \text{pentru } x < 0 \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$ într-un singur punct al domeniului său de definiție. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazurile `x=2`, `x=0` și `x=-2`.

Rezolvare

Schema logică este prezentată în figura 3.62.

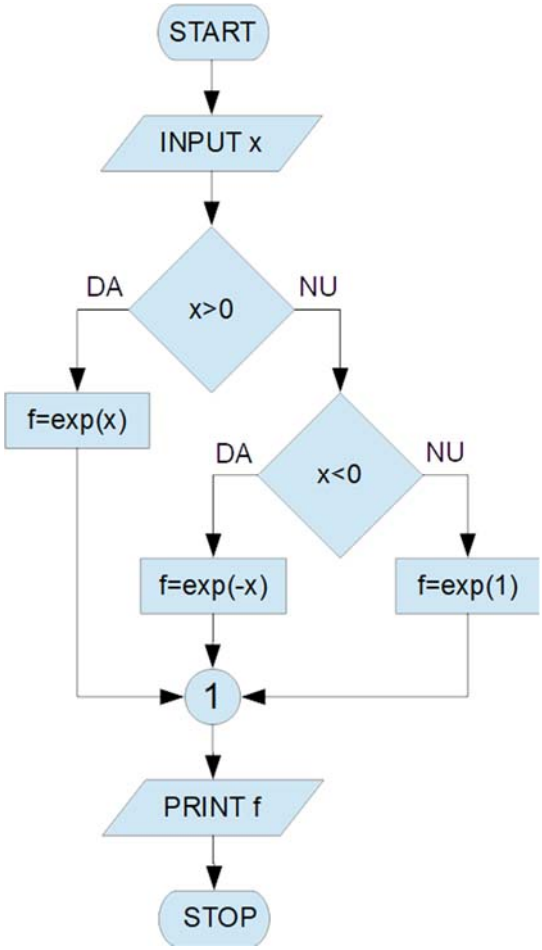


Figura 3.62. Schema logică pentru evaluarea unei funcții definite pe trei subintervale.

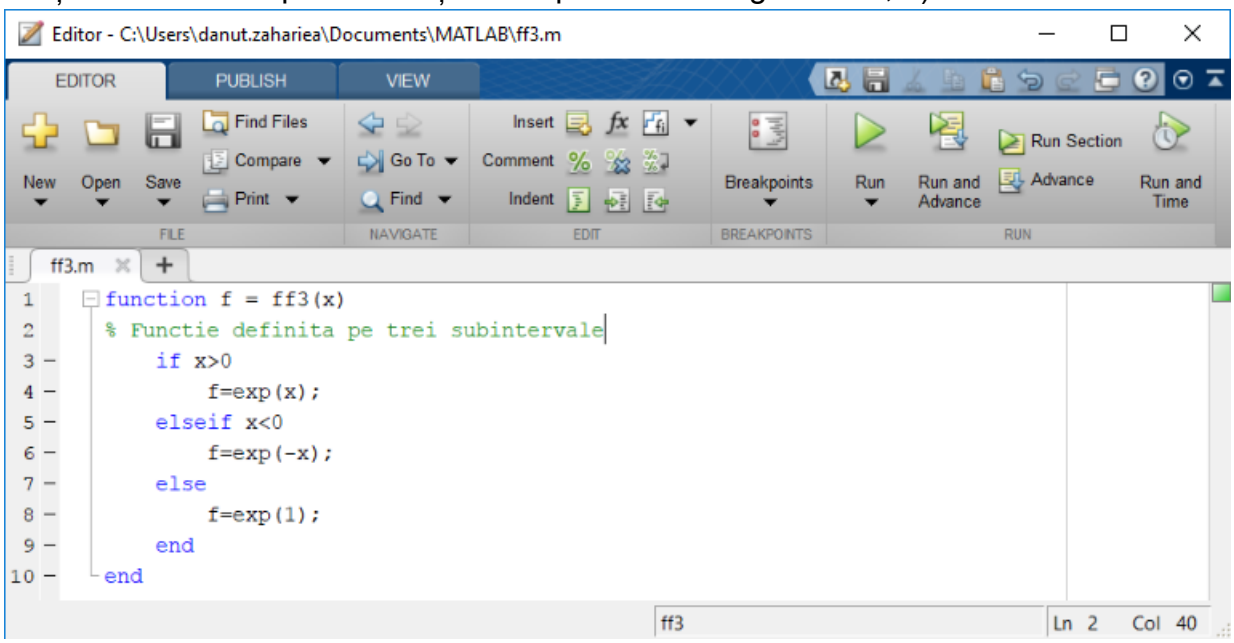
Observații

- Pentru evaluarea funcției `f` se utilizează două structuri alternative.
- După introducerea datelor de intrare, în acest caz valoarea numărului `x`, urmează prima structura alternativă cu două ramuri care verifică dacă `x>0`.
- Dacă expresia logică `x>0` este adevărată, atunci funcția `f` se calculează cu relația $f(x) = e^x$.
- În caz contrar, algoritmul verifică dacă `x<0` prin intermediul celei de-a doua structuri alternative.
- Dacă expresia logică `x<0` este adevărată, atunci funcția `f` se calculează cu relația $f(x) = e^{-x}$.
- În caz contrar, adică pentru `x=0`, funcția `f` se calculează cu relația $f(x) = e$.
- Indiferent însă de valoarea numărului `x`, în urma structurilor alternative se obține o anumită valoare pentru parametrul de ieșire al algoritmului, `f`.

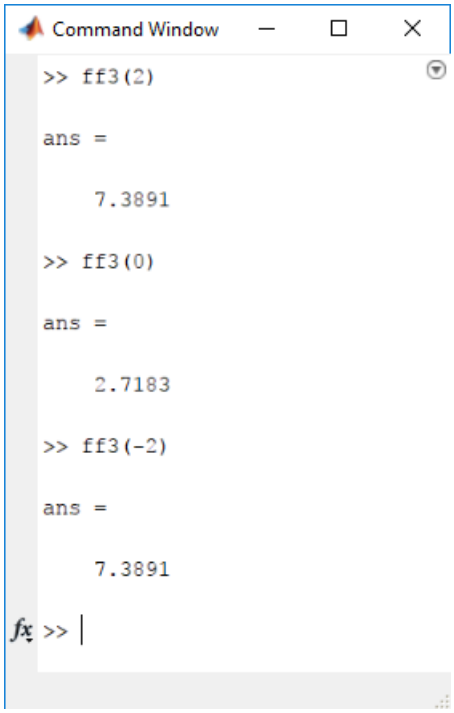
Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.63, a). Numele funcției este `ff3` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff3.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`, cu ajutorul căruia se va evalua funcția `f`. În corpul funcției se află o structură alternativă de tip `if-elseif-else` prin

care, în funcție de valoarea numărului `x`, se va utiliza una din cele trei relații de calcul. Rezultatul obținut în urma evaluării funcției `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare valoare `x` dorită. La apelare, numărul `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.63, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.63. Fișier `function` pentru evaluarea unei funcții definite pe trei subintervale.

Problema 3.24

Se consideră funcția $f: R \to R$ definită prin:

$$f(x) = \begin{cases} 2^{\frac{1}{x-1}}, \text{pentru } x < 1 \\ 0, \text{pentru } x = 1 \\ \ln(x^2 - 2x + 2), \text{pentru } x > 1 \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$.

Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru cazurile `x=2`, `x=1` și `x=-2`.

Rezolvare

Schema logică este prezentată în figura 3.64.

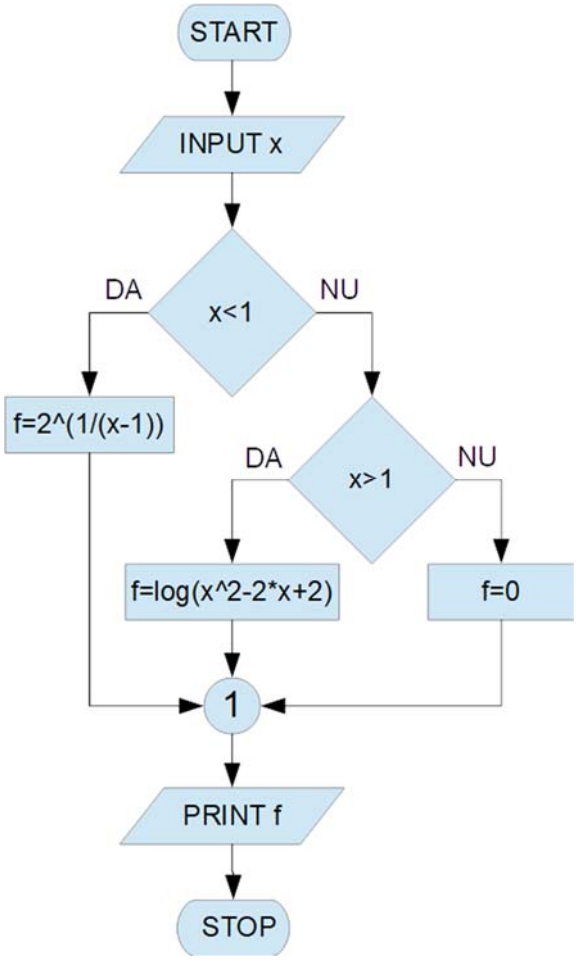


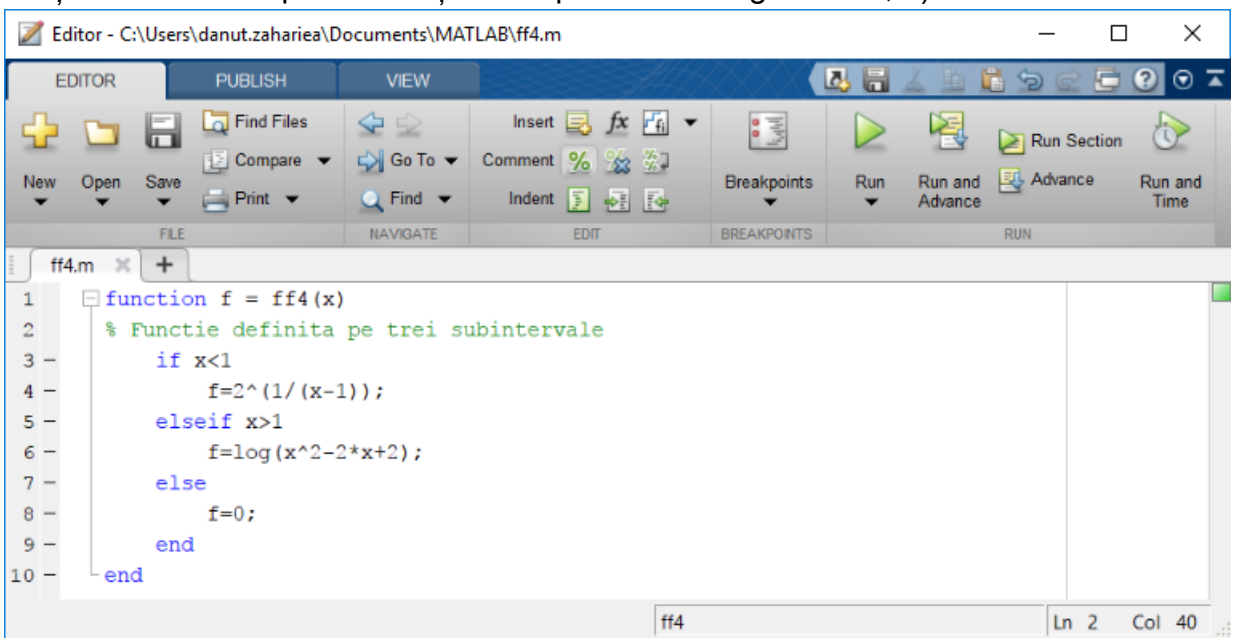
Figura 3.64. Schema logică pentru evaluarea unei funcții definite pe trei subintervale.

Observații

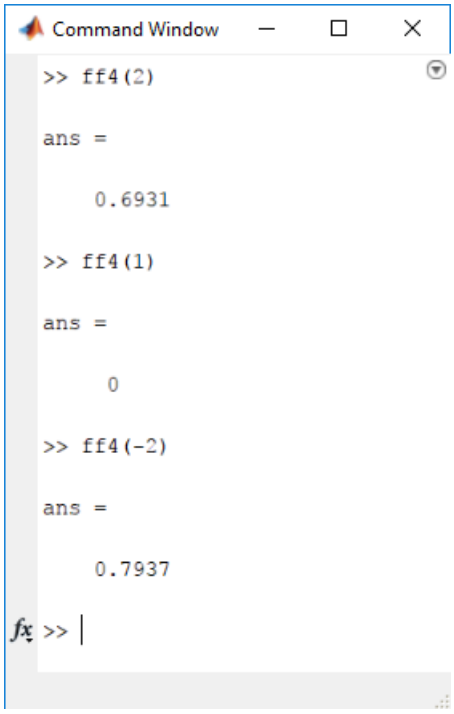
- Pentru evaluarea funcției `f` se utilizează două structuri alternative.
- După introducerea datelor de intrare, în acest caz valoarea numărului `x`, urmează prima structura alternativă cu două ramuri care verifică dacă `x<1`.
- Dacă expresia logică `x<1` este adevărată, atunci funcția `f` se calculează cu relația $f(x) = 2^{\frac{1}{x-1}}$.
- În caz contrar, algoritmul verifică dacă `x>1` prin intermediul celei de-a doua structuri alternative.
- Dacă expresia logică `x>1` este adevărată, atunci funcția `f` se calculează cu relația $f(x) = \ln(x^2 - 2x + 2)$.
- În caz contrar, adică pentru `x=1`, funcția `f` se calculează cu relația $f(x) = 0$.
- Indiferent însă de valoarea numărului `x`, în urma structurilor alternative se obține o anumită valoare pentru parametrul de ieșire al algoritmului, `f`.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.65, a). Numele funcției este `ff4` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff4.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `x`, cu ajutorul căruia se va evalua funcția `f`. În corpul funcției se află o structură alternativă de tip `if-elseif-else` prin care, în funcție de valoarea numărului `x`, se va utiliza una din cele trei relații de calcul. Rezultatul obținut în urma evaluării funcției `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare valoare `x` dorită. La apelare, numărul `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.65, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.65. Fișier `function` pentru evaluarea unei funcții definite pe trei subintervale.

Problema 3.25

Se consideră o întreprindere care urmărește să pună în fabricație un nou produs pentru care s-au calculat prețul unitar de vânzare p [lei/buc], cheltuielile variabile unitare v [lei/buc] și cheltuielile fixe C_f [lei].

Determinarea volumului producției q [buc] se face în corelare cu profitul estimat P [lei], prin raportare la un volum critic al producției q_{cr} [buc] după cum urmează:

$$\begin{cases} q > q_{cr} \Rightarrow P > 0 \\ q = q_{cr} \Rightarrow P = 0 \\ q < q_{cr} \Rightarrow P < 0 \end{cases}$$

în care: dacă P>0 întreprinderea va avea profit, dacă P=0 întreprinderea va fi la punctul critic, iar dacă P<0 întreprinderea va înregistra pierderi.

Se cunosc relațiile de calcul pentru:

- Volumului critic al producției:

$$q_{cr} = \frac{C_f}{p - v}$$

- Profitul:

$$P = q(p - v) - C_f$$

Să se realizeze o schemă logică pentru evaluarea profitului obținut P în funcție de volumul producției q. Să se definească un fișier de tip `function` pentru implementarea schemei logice. Să se verifice pentru cazul `p=7.4, v=3.4, Cf=80000` și trei valori ale volumului producției `q=[14000 20000 24000]`.

Rezolvare

Schema logică este prezentată în figura 3.66.

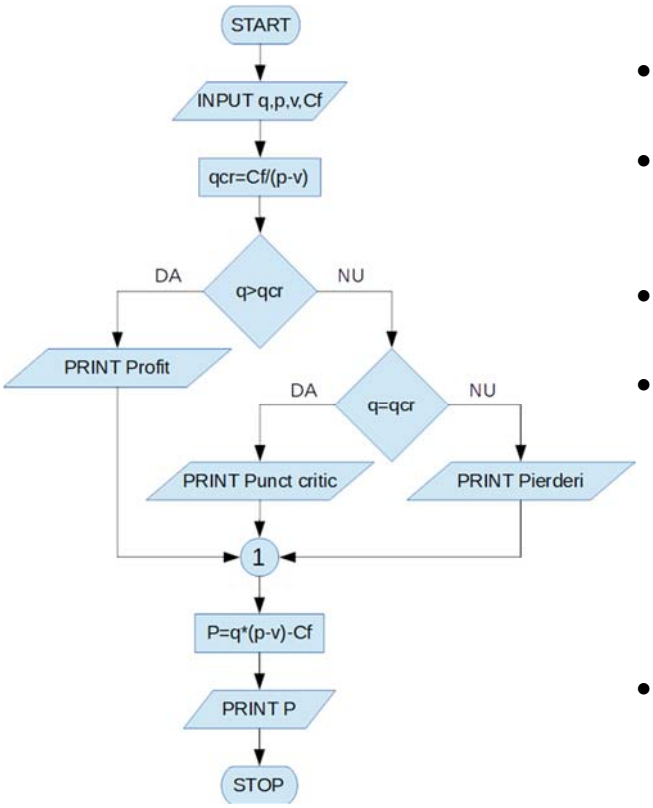


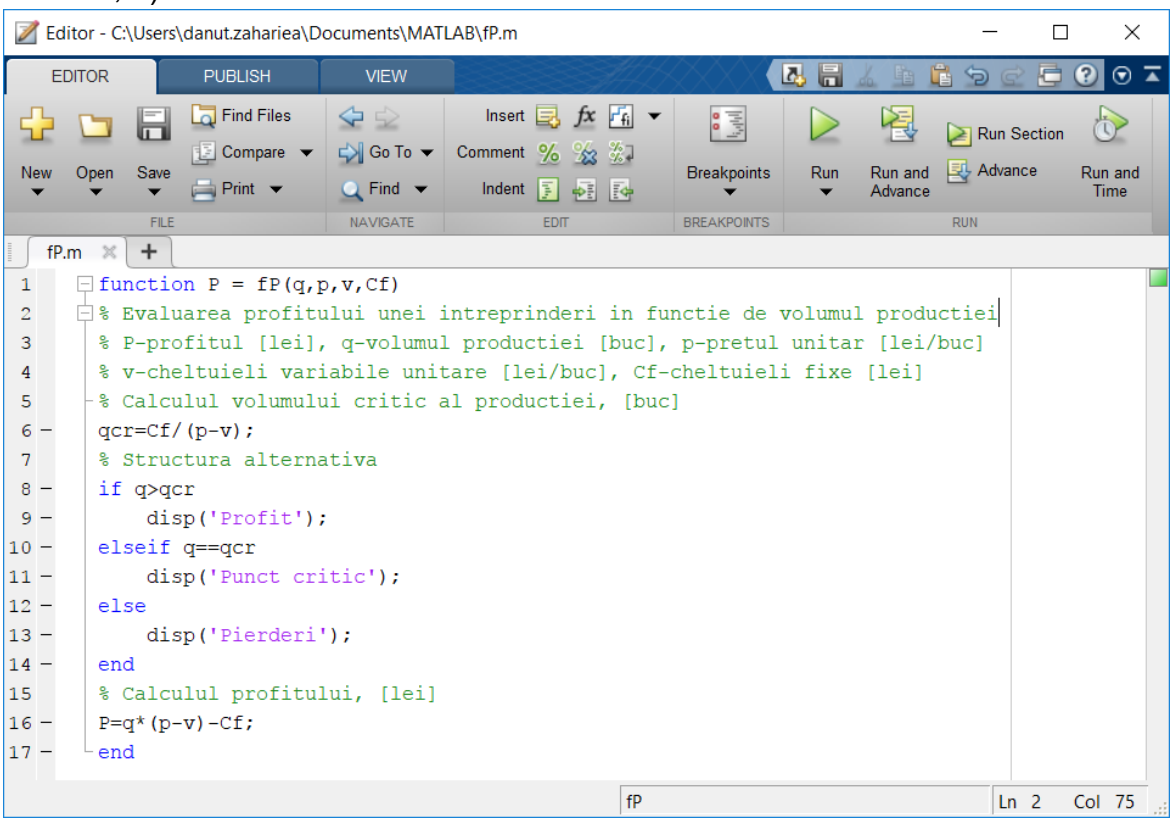
Figura 3.66. Schema logică pentru evaluarea profitului unei întreprinderi în funcție de volumul producției.

Observații

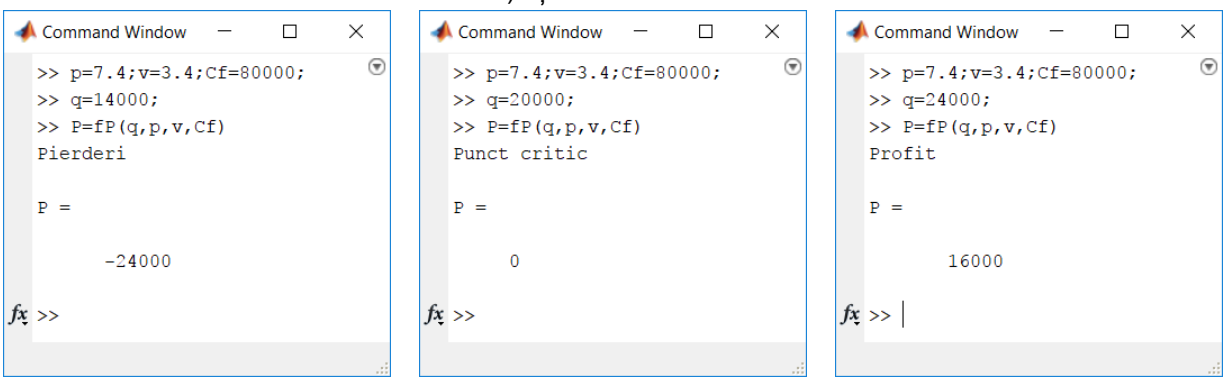
- Pentru evaluarea profitului `P` se utilizează două structuri alternative.
- Datele de intrare sunt volumul producției `q`, prețul de vânzare unitar `q`, cheltuielile variabile unitare `v` și cheltuielile fixe `Cf`.
- Se calculează volumul critic al producției, `qcr`.
- Urmează cele două structuri alternative cu două ramuri care verifică dacă `q>qcr` și apoi dacă `q=qcr`. În funcție de valoarea de adevăr a celor două expresii logice, întreprinderea va avea profit, va fi la punctul mort, respectiv va înregistra pierderi.
- După cele două structuri alternative urmează calculul efectiv al profitului `P`, care reprezintă și parametrul de ieșire al algoritmului.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.67, a). Numele funcției este `fP` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fP.m`. Funcția acceptă patru parametri de intrare: volumul producției `q`, prețul de vânzare unitar `q`, cheltuielile variabile unitare `v` și cheltuielile fixe `Cf`. După calculul volumului critic al producției `qcr=Cf/(p-v)`, urmează o structură alternativă de tip `if-elseif-else` prin care, în funcție de valoarea efectivă a volumului de producție `q`, raportată la valoarea sa critică `qcr` se stabilește dacă întreprinderea va avea profit, dacă va fi la punctul mort, respectiv dacă va înregistra pierderi. După această evaluare urmează calculul efectiv al profitului `P`, care reprezintă și parametrul de ieșire al algoritmului.

Funcția se apelează, în acest caz, din fereastra de comenzi în mod repetat, pentru fiecare valoare a volumului de producție `q`. Rezultatele apelării funcției sunt prezentate în figura 3.67, b).



a) fișierul `function`



b) rezultatele obținute

Figura 3.67. Fișier `function` pentru evaluarea profitului unei întreprinderi în funcție de volumul producției.

3.7. PROBLEME - STRUCTURI ITERATIVE

Problema 3.26

Se consideră mulțimea primelor n numere naturale: $M = \{1, 2, 3, \dots, n\}$

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei elementelor mulțimii M:

$$S = 1 + 2 + 3 + \cdots + n = \sum_{i=1}^{n} i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `n=5`, `n=10` și `n=15`.

Să se verifice rezultatele obținute folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.68.

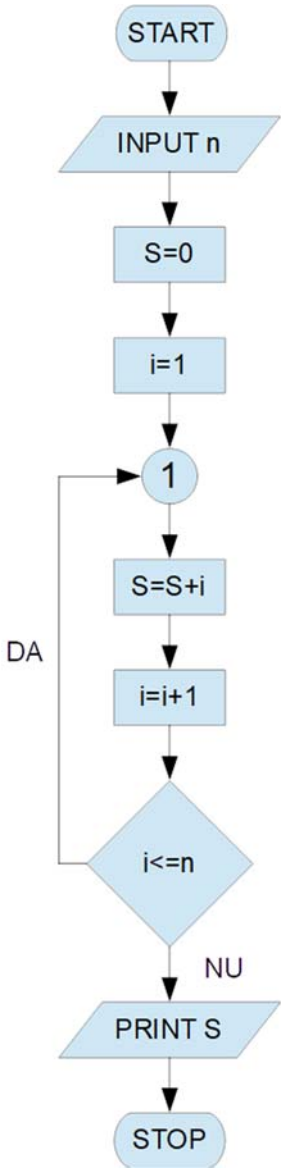


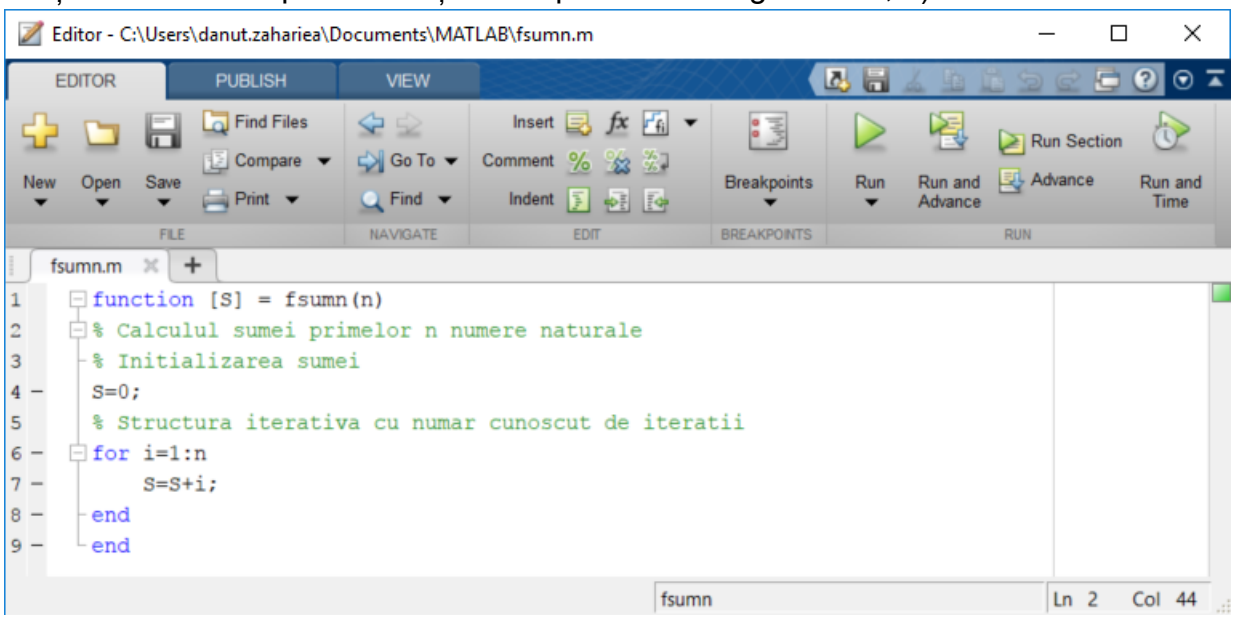
Figura 3.68. Schema logică pentru calculul sumei primelor `n` numere naturale folosind o structură iterativă cu număr cunoscut de iterații.

Observații

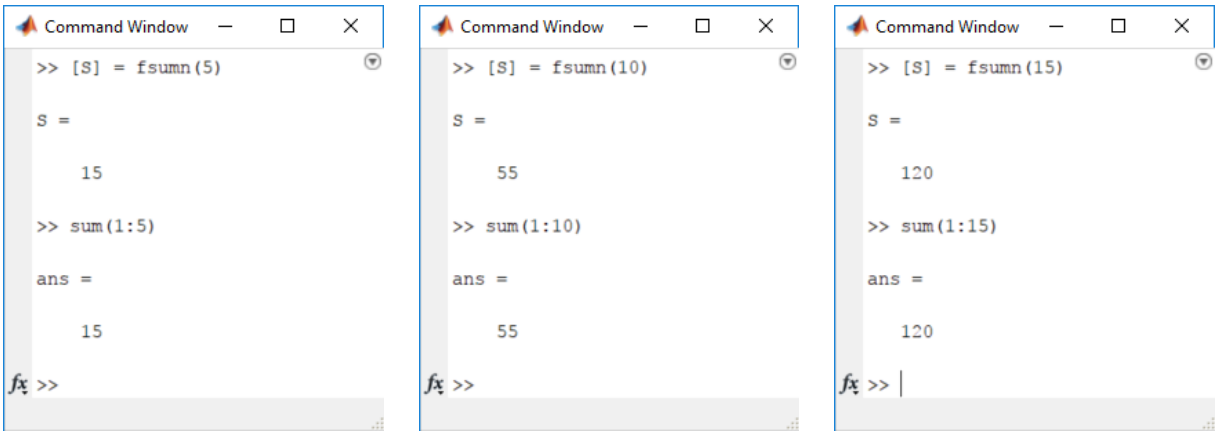
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru sumă `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toate cele `n` numere naturale declarate în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugare valorii curente a contorului la valoarea anterioară a sumei (`S=S+i`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă și se afișează ultima valoare calculată a sumei.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.69, a). Numele funcției este `fsumn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumn.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. Faza de inițializare conține doar inițializarea sumei `S=0`, (elementul neutru pentru adunare). La fiecare iterație, identificată prin valoarea curentă a contorului `i`, suma `S` se recalculează prin adăugarea valorii curente a contorului la valoarea anterioară a sumei, `S=S+i`. Parametrul de ieșire al funcției este suma `S`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.69, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.69. Fișier `function` pentru calculul sumei primelor `n` numere naturale.

Problema 3.27

Se consideră mulțimea primelor n numere naturale: $M = \{1, 2, 3, \dots, n\}$

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei pătratelor elementelor mulțimii M:

$$S = 1^2 + 2^2 + 3^2 + \cdots + n^2 = \sum_{i=1}^{n} i^2$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `n=5`, `n=10` și `n=15`.

Să se verifice rezultatele obținute folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.70.

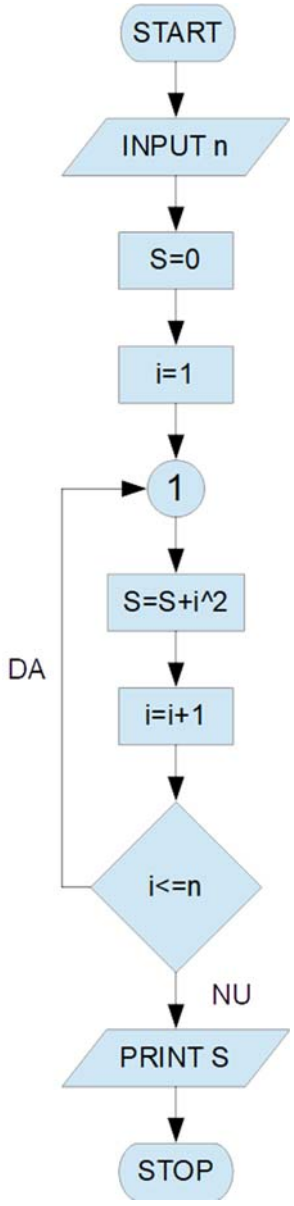


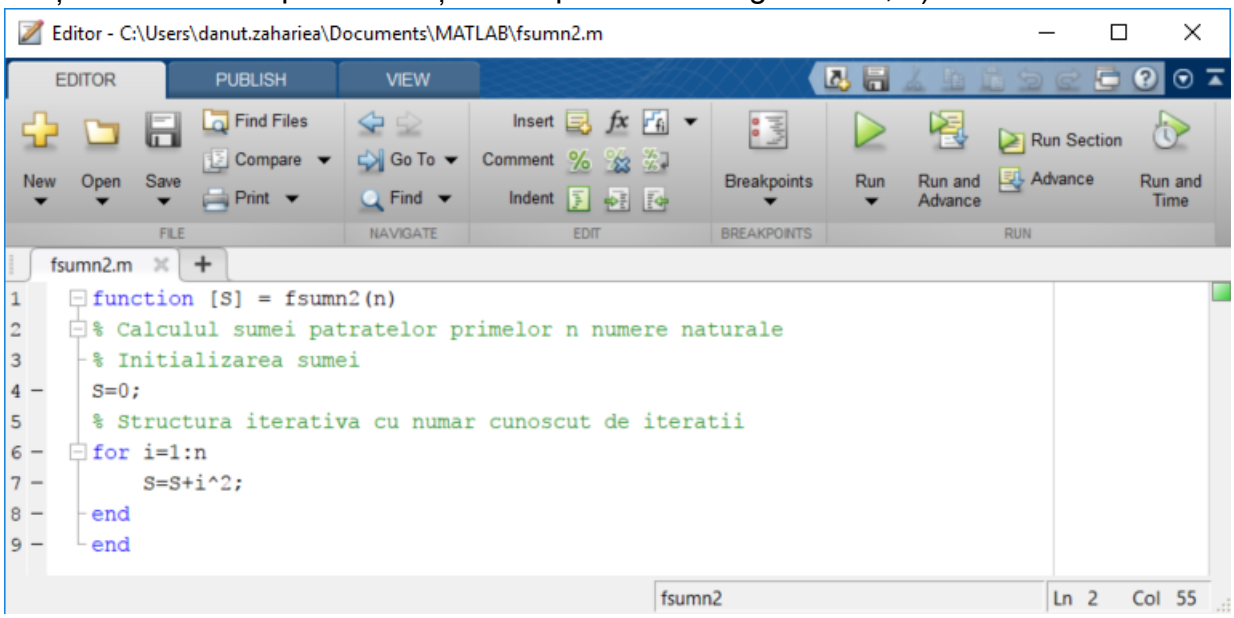
Figura 3.70. Schema logică pentru calculul sumei pătratelor primelor `n` numere naturale folosind o structură iterativă cu număr cunoscut de iterații.

Observații

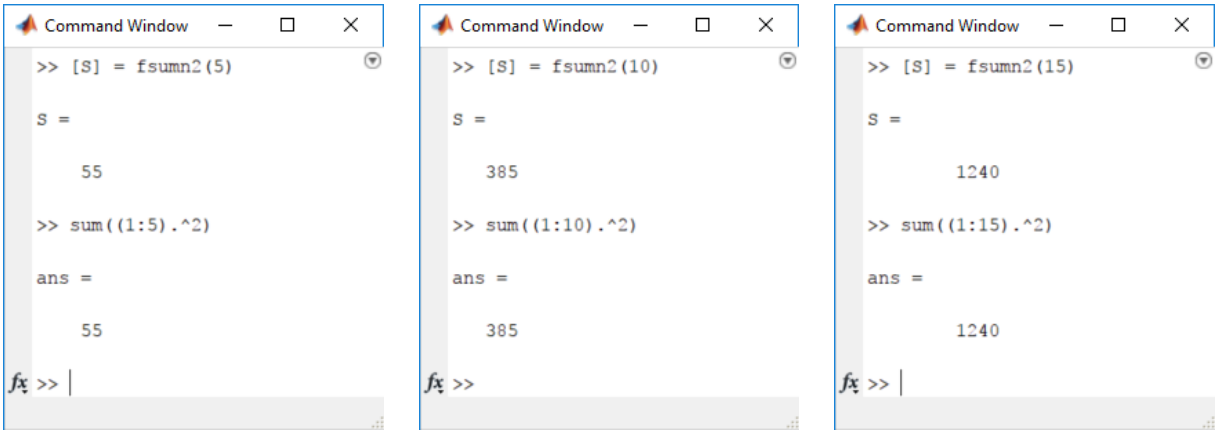
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru sumă `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toate cele `n` numere naturale declarate în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugare pătratului valorii curente a contorului la valoarea anterioară a sumei (`S=S+i^2`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă și se afișează ultima valoare calculată a sumei.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.71, a). Numele funcției este `fsumn2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumn2.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. Faza de inițializare conține doar inițializarea sumei `S=0`, (elementul neutru pentru adunare). La fiecare iterație, identificată prin valoarea curentă a contorului `i`, suma `S` se recalculează prin adăugarea pătratului valorii curente a contorului la valoarea anterioară a sumei, `S=S+i^2`. Parametrul de ieșire al funcției este suma `S`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.71, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.71. Fișier `function` pentru calculul sumei pătratelor primelor `n` numere naturale.

Problema 3.28

Se consideră mulțimea primelor n numere naturale: $M = \{1, 2, 3, \dots, n\}$

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei cuburilor elementelor mulțimii M:

$$S = 1^3 + 2^3 + 3^3 + \cdots + n^3 = \sum_{i=1}^{n} i^3$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `n=5`, `n=10` și `n=15`.

Să se verifice rezultatele obținute folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.72.

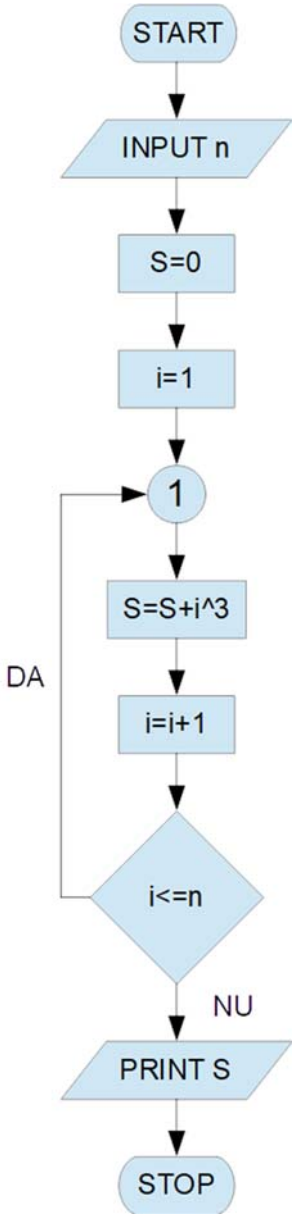


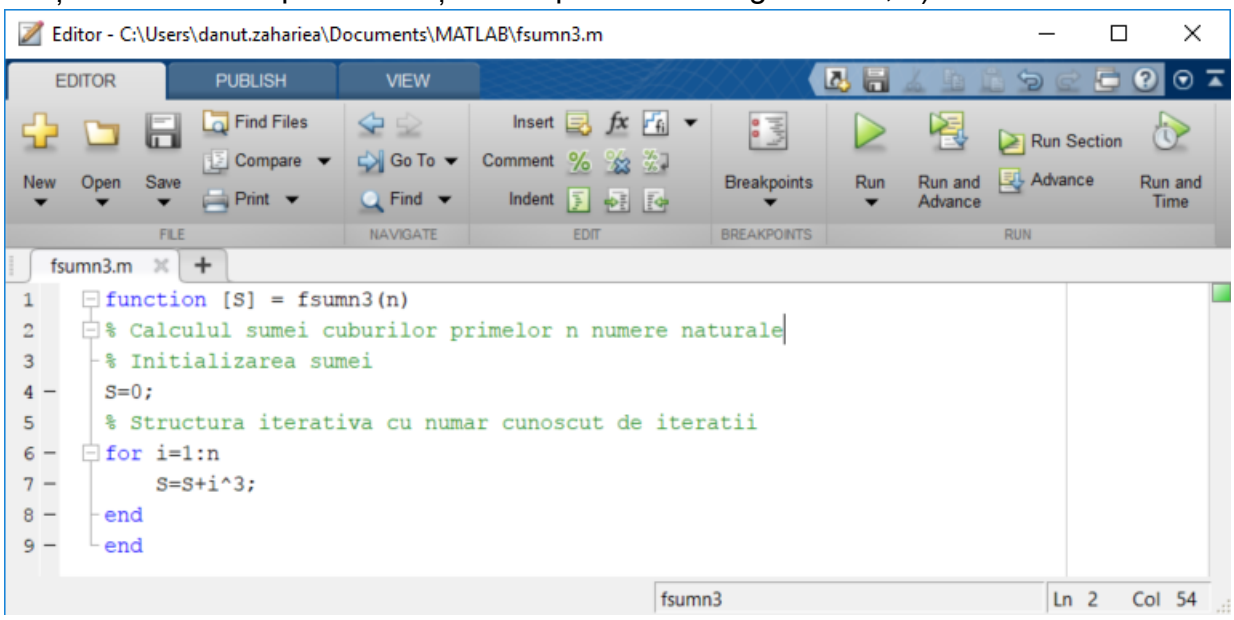
Figura 3.72. Schema logică pentru calculul sumei cuburilor primelor `n` numere naturale folosind o structură iterativă cu număr cunoscut de iterații.

Observații

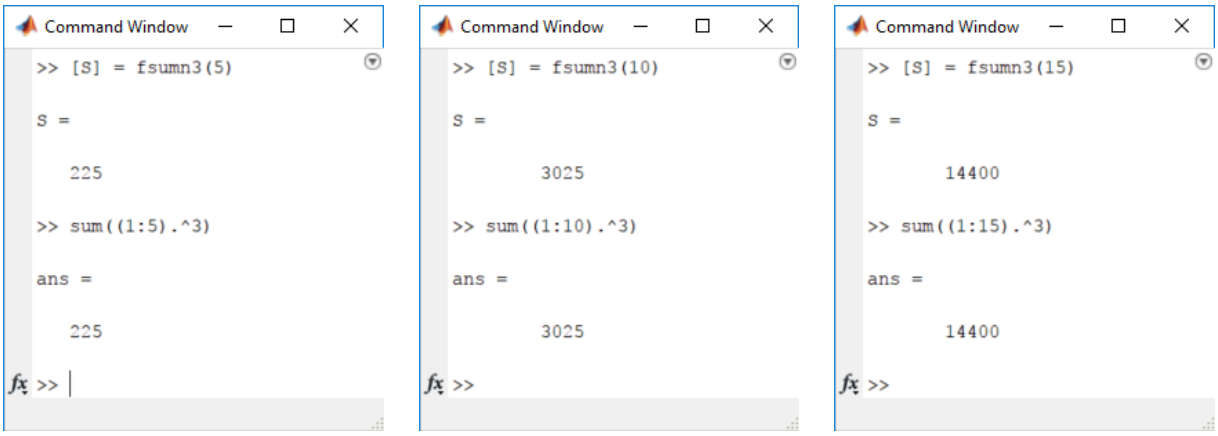
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru suma `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toate cele `n` numere naturale declarate în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugare cubului valorii curente a contorului la valoarea anterioară a sumei (`S=S+i^3`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează ultima valoare calculată a sumei.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.73, a). Numele funcției este `fsumn3` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumn3.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. Faza de inițializare conține doar inițializarea sumei `S=0`, (elementul neutru pentru adunare). La fiecare iterație, identificată prin valoarea curentă a contorului `i`, suma `S` se recalculează prin adăugarea cubului valorii curente a contorului la valoarea anterioară a sumei, `S=S+i^3`. Parametrul de ieșire al funcției este suma `S`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.73, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.73. Fișier `function` pentru calculul sumei cuburilor primelor `n` numere naturale.

Problema 3.29

Se consideră mulțimea primelor n numere naturale: $M = \{1, 2, 3, \dots, n\}$

Folosind o structură iterativă cu test inițial să se realizeze o schemă logică pentru descrierea algoritmului de determinare a produsului elementelor mulțimii M:

$$P = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{i=1}^{n} i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `n=5`, `n=10` și `n=15`.

Să se verifice rezultatele obținute folosind funcția MATLAB predefinită `prod`.

Rezolvare

Schema logică este prezentată în figura 3.74.

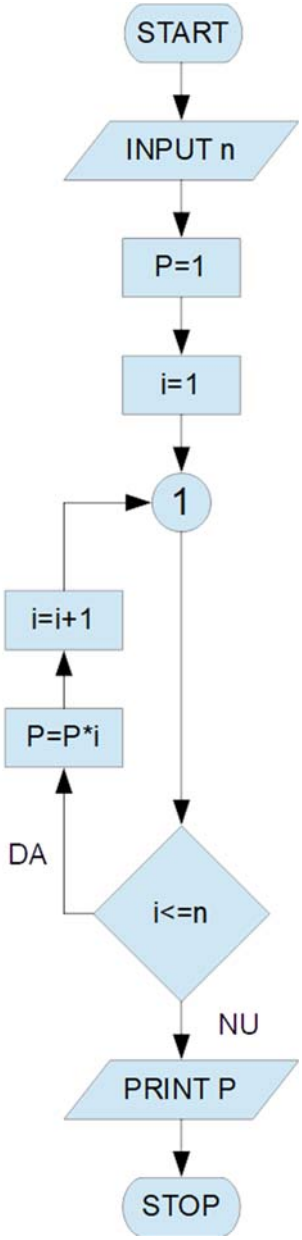


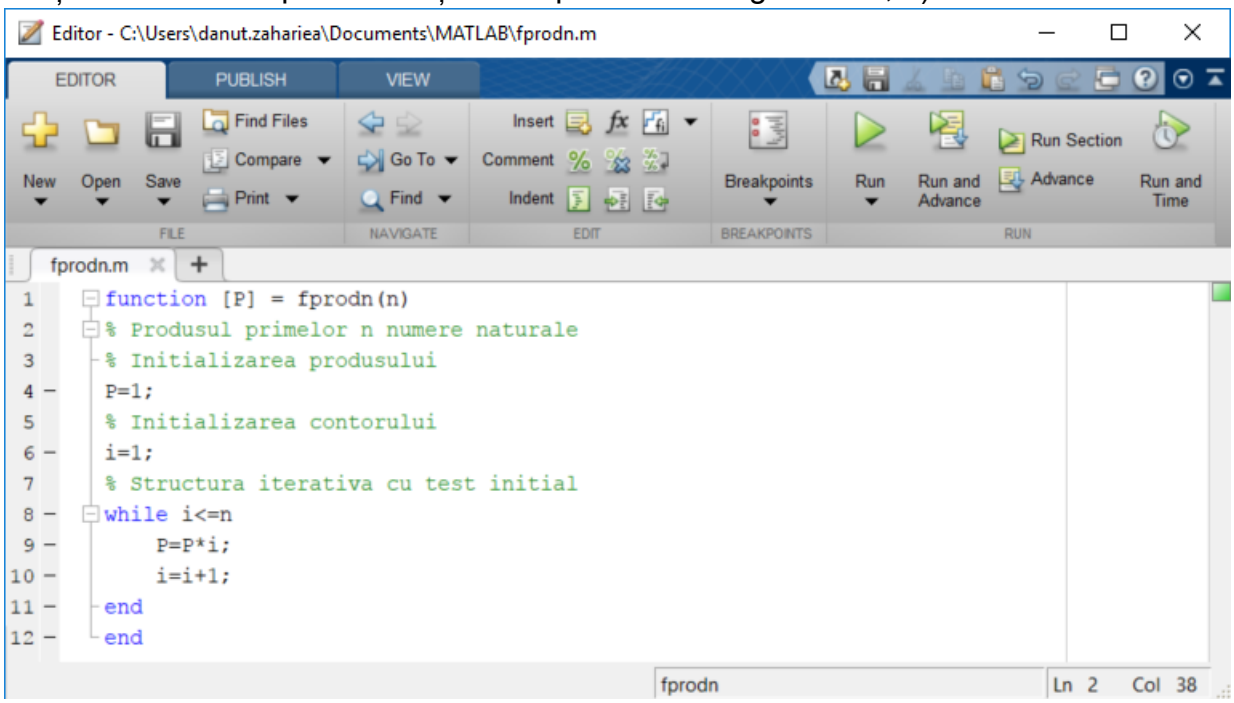
Figura 3.74. Schema logică pentru calculul produsului primelor `n` numere naturale folosind o structură iterativă cu test inițial.

Observații

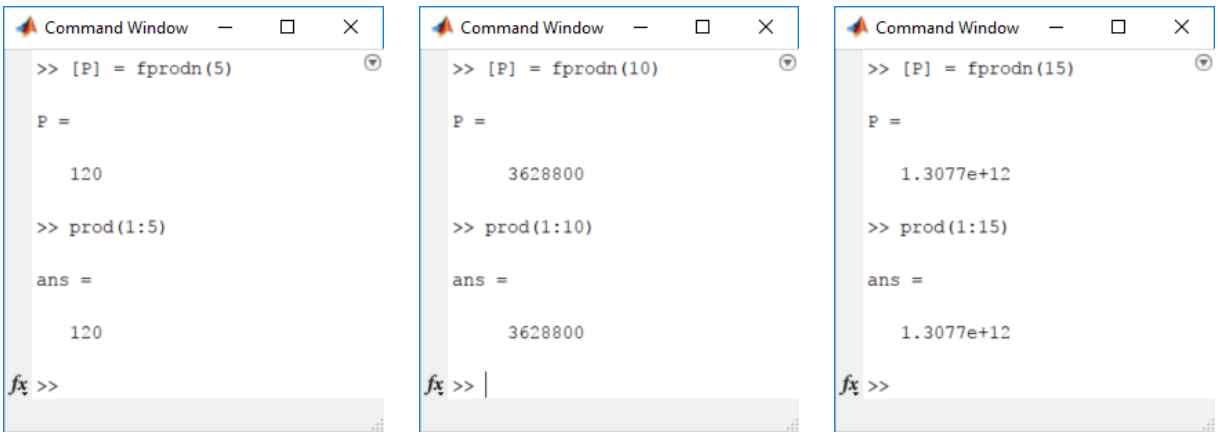
- Schema logică se bazează pe o structură iterativă cu test inițial.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru produs `P=1` și pentru contorul `i=1` al structurii iterative care va parcurge toate cele `n` numere naturale declarate în faza de introducere a datelor de intrare.
- După faza de inițializare a algoritmului urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se intră în corpul structurii iterative și se recalculează valoare produsului prin înmulțirea valorii curente a contorului cu valoarea anterioară a produsului (`P=P*i`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu test inițial și se afișează ultima valoare calculată a produsului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.75, a). Numele funcției este `fprodn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fprodn.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. Calculul produsului `P` se efectuează cu o structură iterativă cu test inițial. Faza de inițializare conține pe lângă inițializarea produsului `P=1`, (elementul neutru pentru înmulțire) și inițializarea contorului `i=1`. La fiecare iterație, identificată prin valoarea curentă a contorului `i`, produsul `P` se recalculează prin înmulțirea valorii curente a contorului cu valoarea anterioară a produsului, `P=P*i`. Parametrul de ieșire al funcției este produsul `P`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.75, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.75. Fișier `function` pentru calculul produsului primelor `n` numere naturale.

Problema 3.30

Se consideră numărul pozitiv n. Factorialul numărul n este definit prin relația:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n = \prod_{k=1}^{n} k$$

Să se realizeze schema logică pentru descrierea algoritmului de determinare a factorialului numărului n folosind o structură iterativă cu număr cunoscut de iterații, respectiv o structură iterativă cu test inițial.

Să se realizeze în plus o schemă logică pentru descrierea algoritmului de determinare a factorialului numărului n folosind metoda recursivă definită prin relația:

$$n! = n \cdot (n-1)!$$

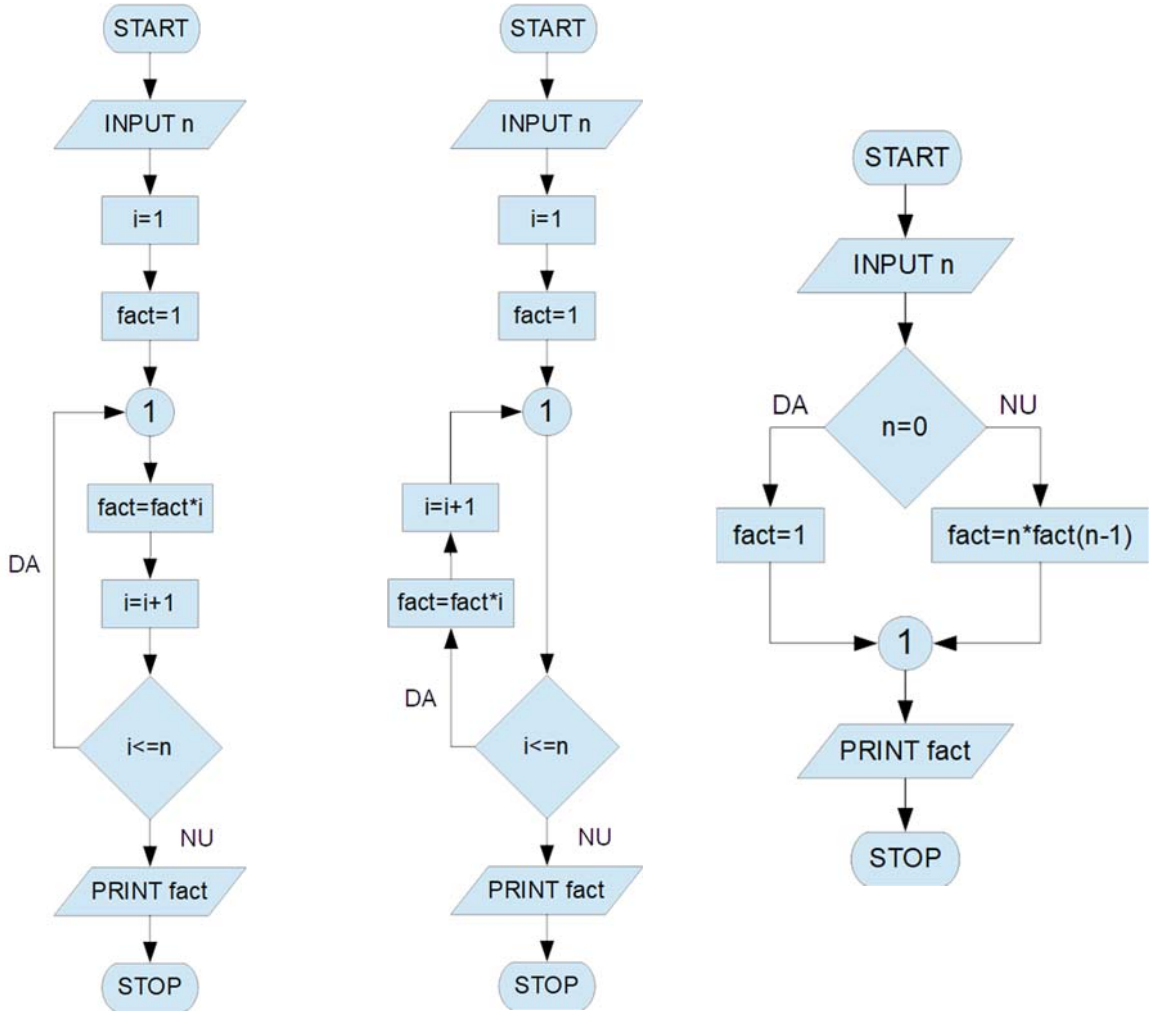
Să se scrie câte un fișier de tip `function` pentru implementarea schemelor logice.

Să se verifice pentru `n=5`, `n=6` și `n=7`.

Să se verifice rezultatele folosind funcția MATLAB predefinită `factorial`.

Rezolvare

Schemele logice sunt prezentate în figura 3.76, a) pentru cazul structurii iterative cu număr cunoscut de iterații, în figura 3.76, b) pentru cazul structurii iterative cu test inițial, respectiv în figura 3.76, c) pentru cazul exprimării recursive a factorialului.



a) structură iterativă cu număr cunoscut de iterații

b) structură iterativă cu test inițial

c) exprimare recursivă

Figura 3.76. Scheme logice pentru calculul funcției factorial.

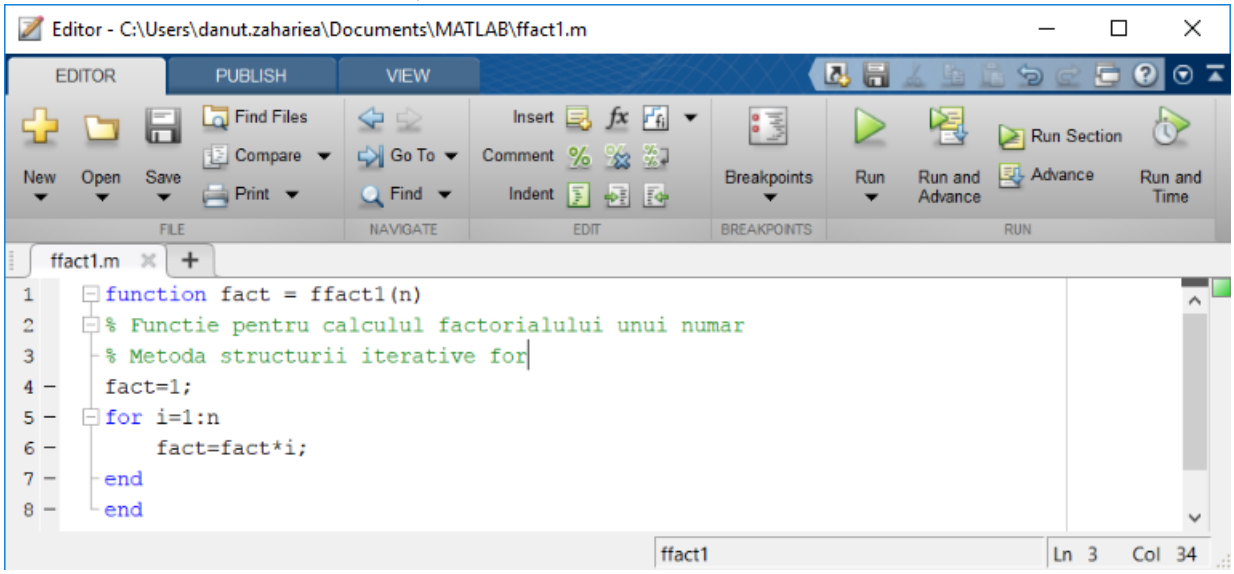
Observații

- Datele de intrare se referă la numărul `n` pentru care se dorește calcularea funcției factorial.
- Pentru primele două scheme logice faza de inițializare constă în inițializarea contorului `i=1`, respectiv în inițializarea funcției factorial `fact=1` (element neutru pentru înmulțire).
- Pentru primele două scheme logice urmează apoi blocul de calcul iterativ care este structurat în mod diferit:
    - Pentru schema logică din figura 3.76, a), la fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare variabilei `fact` prin înmulțirea valorii sale anterioare cu valoarea curentă a contorului (`fact=fact*i`), după care urmează faza de incrementare a contorului cu o unitate, `i=i+1`. După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`. Dacă expresia logică `i<=n` este adevărată se continuă recalcularea factorialului și incrementarea în continuare a contorului. Dacă expresia logică `i<=n` este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează ultima valoare calculată a variabilei `fact`.
    - Pentru schema logică din figura 3.76, b), la fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`. Dacă expresia logică `i<=n` este adevărată se intră în corpul structurii iterative și se recalculează valoare variabilei `fact` prin înmulțirea valorii sale anterioare cu valoarea curentă a contorului (`fact=fact*i`), după care urmează faza de incrementare a contorului cu o unitate, `i=i+1`. Dacă expresia logică `i<=n` este falsă, se părăsește structura iterativă cu test inițial și se afișează ultima valoare calculată a variabilei `fact`.
- Pentru cea de-a treia schemă logică (figura 3.76, c), după introducerea numărului `n`, urmează o structură alternativă cu două ramuri. Expresia logică urmărește determinarea valorii de adevăr a egalității `n=0`. Dacă expresia logică este adevărată atunci funcția factorial va avea valoarea `fact=1`. În caz contrar, funcția factorial se va apela pe ea însăși în formula recursivă `fact=n*fact(n-1)`. La rândul său, funcția `fact(n-1)` se va apela din nou pe ea însăși în formula recursivă `fact(n-1)=(n-1)*fact(n-2)`. Acest apel recursiv continuă până la evaluarea formulei recursive `fact(1)=1*fact(0)`, în care pentru evaluarea funcției `fact(0)` se trece pe cealaltă ramură a structurii alternative conform căreia `fact(0)=1`. Conform acestui apel recursiv se obține relația `fact=n*(n-1)*...*2*1`, ceea ce corespunde de altfel relației de definiție a funcției factorial.
- Datele de ieșire se referă la valoarea variabilei `fact`, reprezentând factorialul numărului dat `n`.

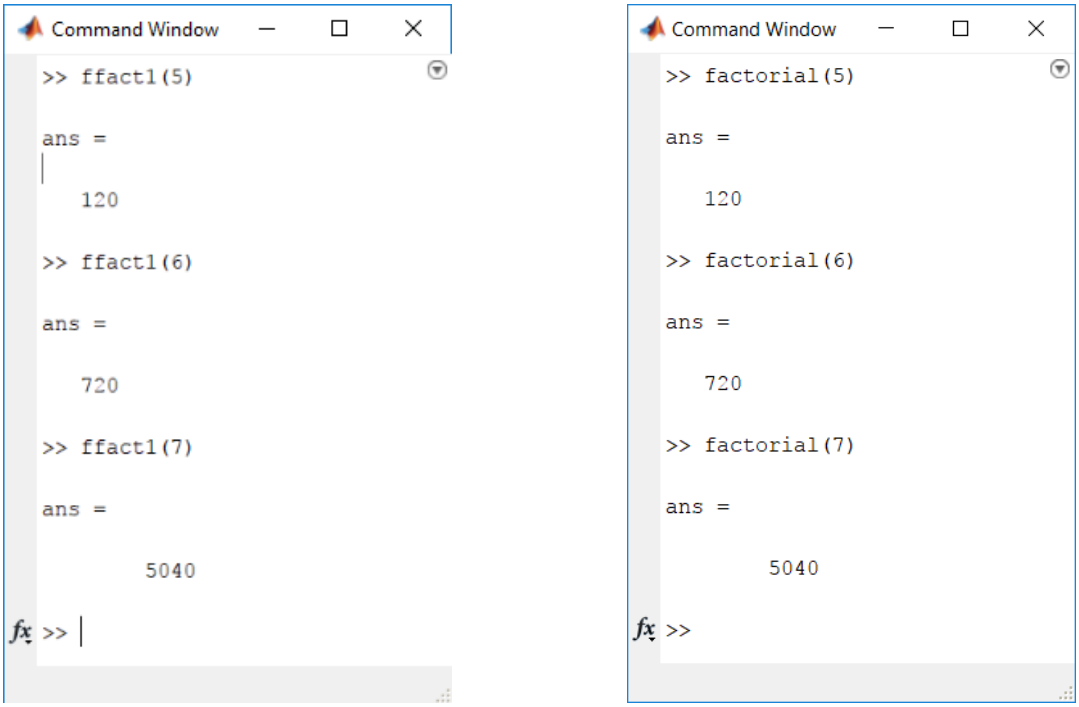
Funcția care implementează algoritmul descris în schema logică din figura 3.76, a) este definită în fișierul de tip `function` prezentat în figura 3.77, a). Numele funcției este `ffact1` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ffact1.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. După inițializarea factorialului (`fact=1`, elementul neutru pentru înmulțire) urmează o structură iterativă cu număr cunoscut de iterații prin care la fiecare iterație se recalculează valoarea factorialului (`fact=fact*i`). Variabila `fact` este parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, în mod repetat pentru fiecare valoare dorită a numărului, `n=5`, `n=6`, respectiv `n=7`. La apelare, numărul `n` se transferă parametrului de intrare al funcției.

Rezultatul apelării funcției este prezentat în figura 3.77, b).



a) fișierul `function`

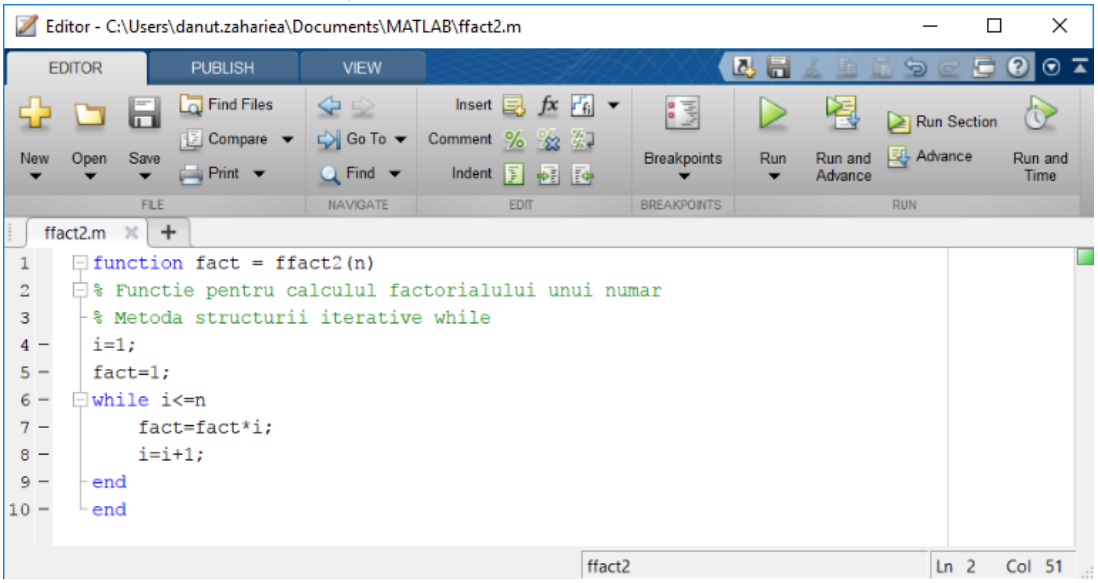


b) rezultatul obținut

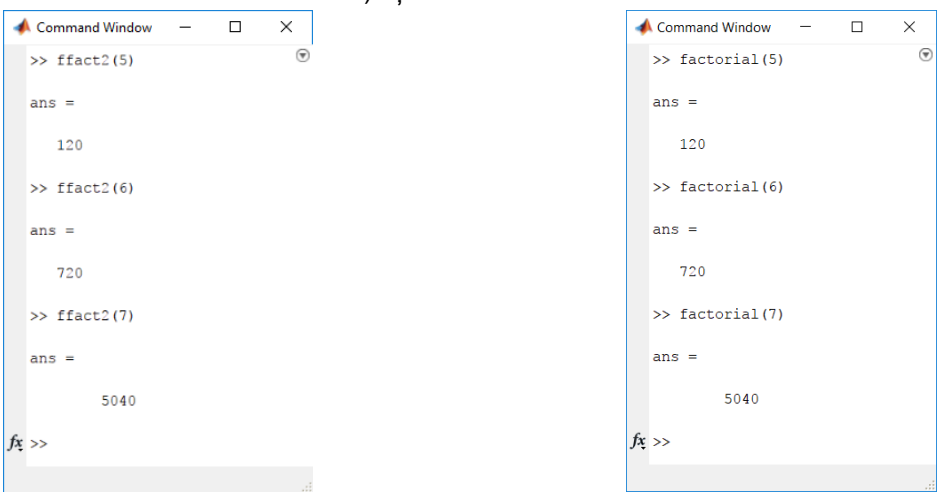
Figura 3.77. Fișier `function` pentru definirea funcției factorial pe baza unei structuri iterative cu număr cunoscut de iterații.

Funcția care implementează algoritmul descris în schema logică din figura 3.76, b) este definită în fișierul de tip `function` prezentat în figura 3.78, a). Numele funcției este `ffact2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ffact2.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. După inițializarea contorului (`i=1`) și a factorialului (`fact=1`, elementul neutru pentru înmulțire) urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`. Dacă expresia logică este adevărată se intră în corpul structurii iterative și se recalculează valoare factorialului prin înmulțirea valorii curente a contorului cu valoarea anterioară a factorialului (`fact=fact*i`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`. Dacă expresia logică este falsă, se părăsește structura iterativă cu test inițial și se afișează ultima valoare calculată a factorialului. Variabila `fact` este parametrul de ieșire al funcției. Funcția se apelează, în acest caz, din fereastra de comenzi, în mod repetat pentru fiecare valoare dorită a numărului, `n=5`, `n=6`, respectiv `n=7`. La apelare, numărul `n` se transferă parametrului de intrare al funcției.

Rezultatul apelării funcției este prezentat în figura 3.78, b).



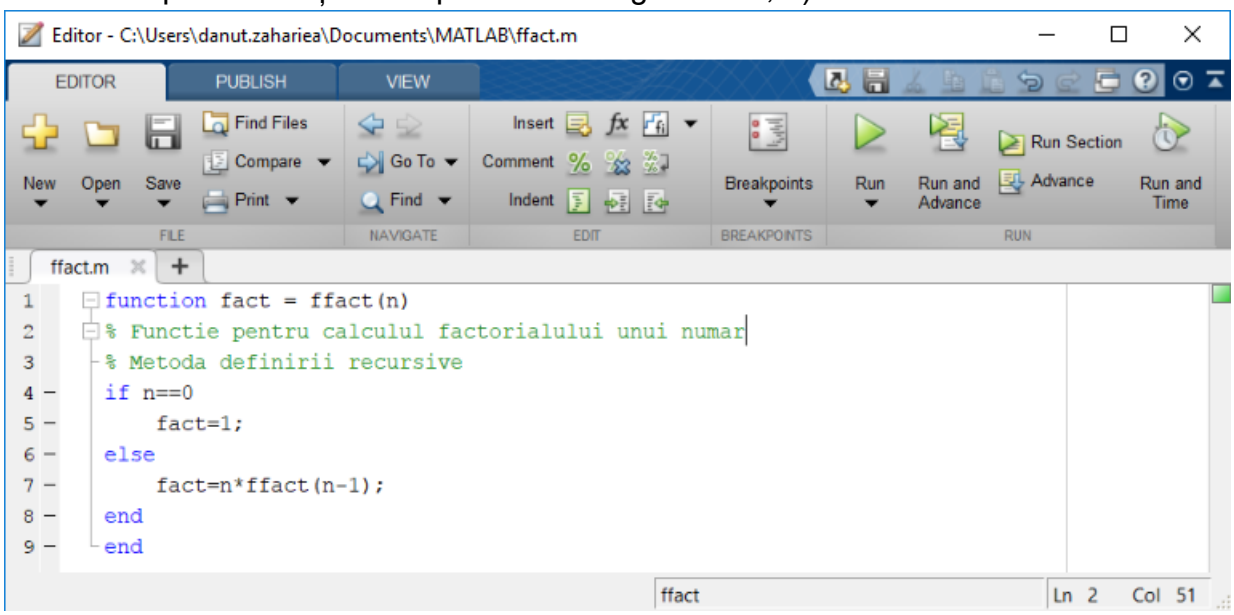
a) fișierul `function`



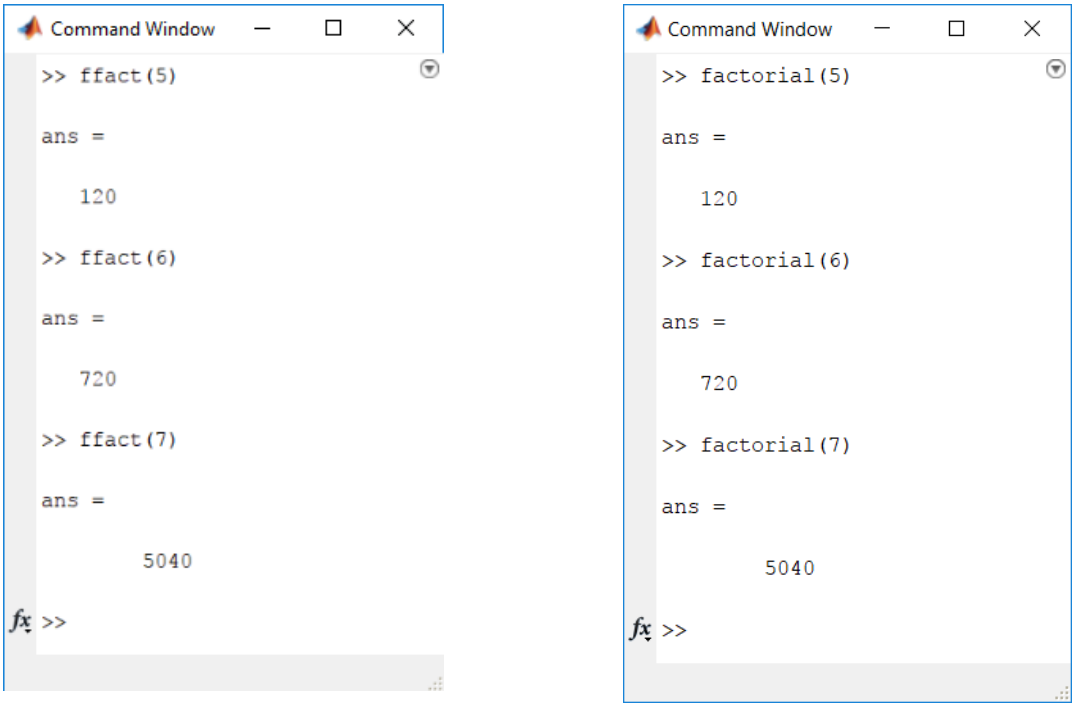
b) rezultatul obținut

Figura 3.78. Fișier `function` pentru definirea funcției factorial pe baza unei structuri iterative cu test inițial.

Funcția care implementează algoritmul descris în schema logică din figura 3.76, c) este definită în fișierul de tip `function` prezentat în figura 3.79, a). Numele funcției este `ffact` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ffact.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea funcției factorial prin apelarea recursivă. Rezultatul obținut se atribuie variabilei `fact`, care reprezintă și parametrul de ieșire al funcției. Funcția se apelează, în acest caz, din fereastra de comenzi, în mod repetat pentru fiecare valoare dorită a numărului, `n=5`, `n=6`, respectiv `n=7`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.79, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.79. Fișier `function` pentru definirea funcției factorial pe baza exprimării recursive.

Problema 3.31

Se consideră două numere naturale n și m, $n > m$. Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a următorilor parametri:

- Permutări de n:

$$P_n = n!$$

- Aranjamente de n luate câte m:

$$A_n^m = \frac{n!}{(n-m)!}$$

- Combinări de n luate câte m:

$$C_n^m = \frac{n!}{m! \cdot (n-m)!}$$

în care factorialul numărul n este definit prin relația: $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n = \prod_{k=1}^{n} k$

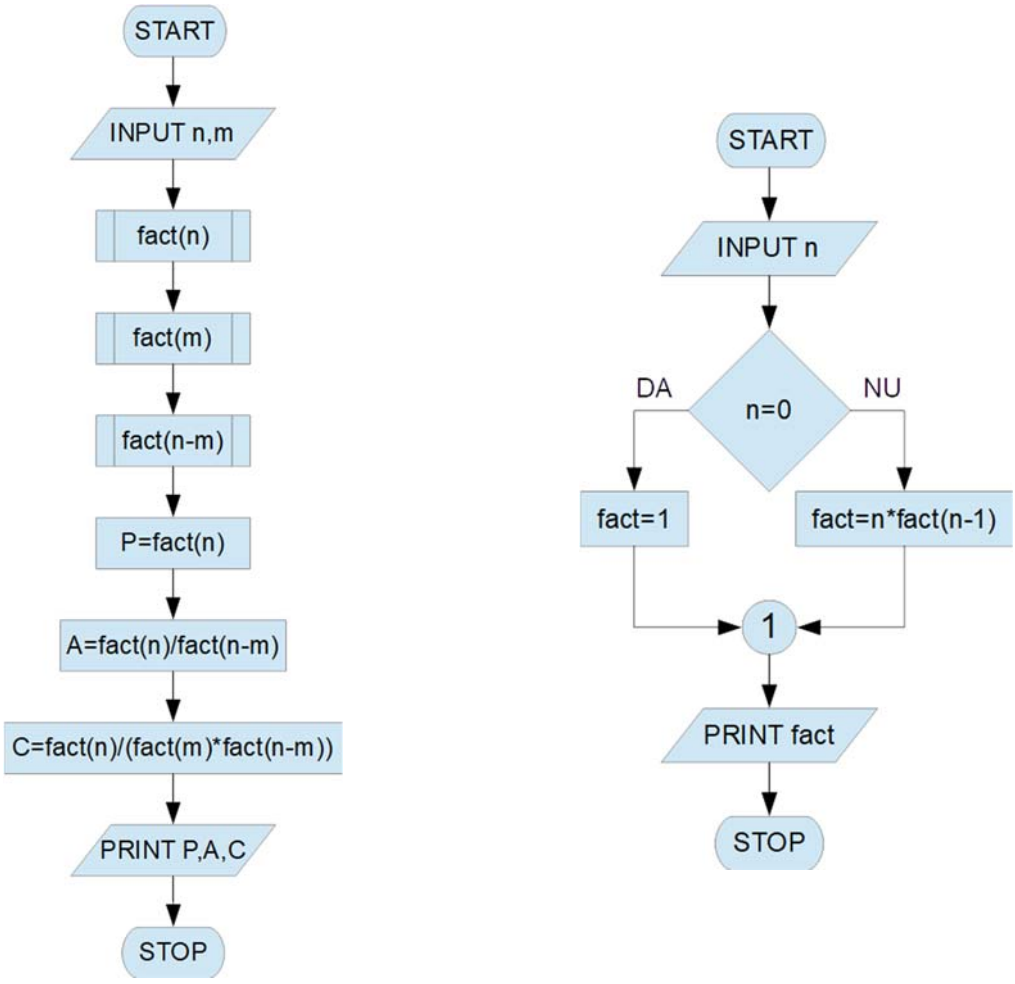
Să se scrie un fișier de tip `function` pentru implementarea schemei logice de determinare a factorialului și un fișier de tip `script` pentru apelarea funcției și calculul celor trei parametri.

Să se verifice pentru `n=8` și `m=5`.

Să se verifice rezultatele folosind funcția MATLAB predefinită `factorial`.

Rezolvare

Schemele logice sunt prezentate în figura 3.80.



a) schema logică principală b) schema logică secundară (procedura `fact`)

Figura 3.80. Schema logică pentru calculul parametrilor P_n, A_n^m și C_n^m.

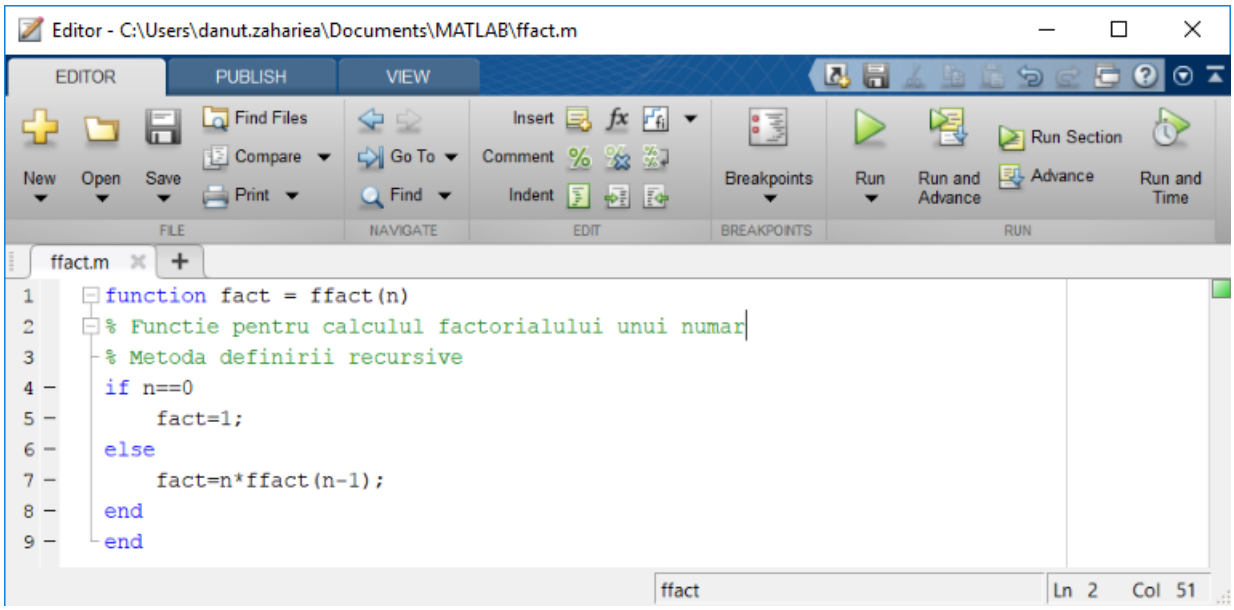
Observații

- Algoritmul pentru determinarea celor trei parametri P_n, A_n^m și C_n^m este descris cu ajutorul a două scheme logice: schema logică principală (figura 3.80, a) care apelează în mod repetat procedura `fact` de calculul a factorialului și care calculează apoi cei trei parametri, și respectiv schema logică secundară care reprezintă descrierea procedurii `fact` (figura 3.80, b).
- Datele de intrare ale schemei logice principale se referă la numerele `n` și `m` pentru care se dorește calcularea celor trei parametri P_n, A_n^m și C_n^m.
- Se apelează apoi procedura pentru calculul factorialului, denumită `fact`, pentru determinarea expresiilor $n!$, $m!$ și $(n-m)!$, expresii care intervin în calculul celor trei parametri P_n, A_n^m și C_n^m.
- Se calculează apoi cei trei parametri P_n, A_n^m și C_n^m în conformitate cu relațiile lor de definiție, folosind expresiile calculate anterior cu procedura `fact`.
- Datele de ieșire ale algoritmului prezentat în schema logică principală se referă la valorile variabilelor `P`, `A` și `C`, reprezentând valorile celor trei parametri P_n, A_n^m și C_n^m.
- Schema logică secundară, pentru procedura de calcul a factorialului, este prezentată în figura 3.80, b). S-a utilizat metoda de determinare a factorialului numărului n folosind exprimarea recursivă definită prin relația $n! = n \cdot (n-1)!$. Datele de intrare ale schemei logice secundare se referă la numărul n pentru care se dorește calcularea factorialului $n!$. Datele de ieșire ale schemei logice secundare se referă la valoarea factorialului numărului respectiv, $n!$.

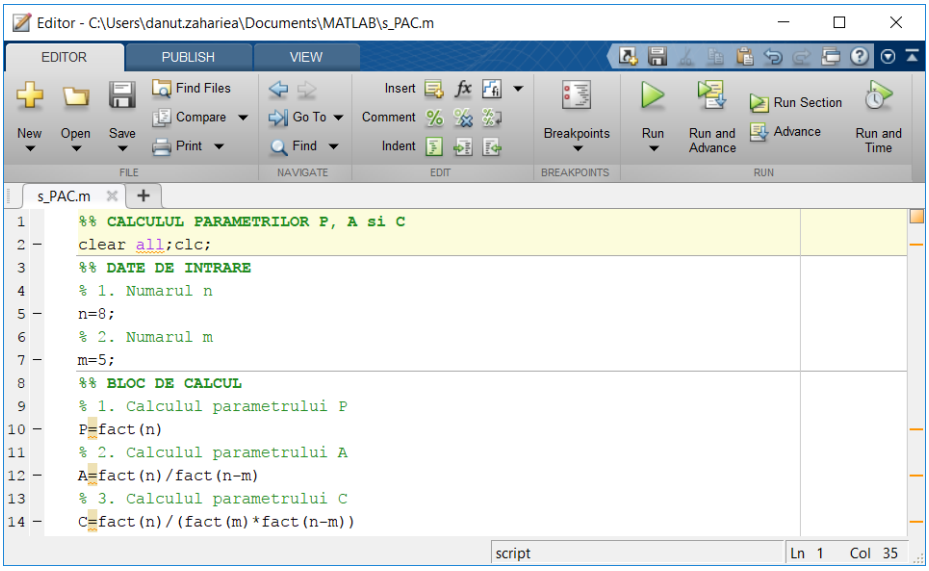
Funcția care implementează algoritmul descris în schema logică secundară din figura 3.80, b) este definită în fișierul de tip `function` prezentat în figura 3.81, a). Numele funcției este `ffact` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ffact.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se găsește o structură alternativă de tip `if-else` prin care se determină valoarea funcției factorial prin apelarea recursivă. Rezultatul obținut se atribuie variabilei `fact`, care reprezintă și parametrul de ieșire al funcției.

Funcția pentru calculul factorialului se apelează, în acest caz, dintr-un fișier de tip `script`, prezentat în figura 3.81, b). Fișierul `script` conține 3 secțiuni: secțiunea de titlu care include și instrucțiunile `clear all` și `clc`; secțiunea de introducere a datelor de intrare ale problemei (cele două numere `n=8` și `m=5`); secțiunea de calcul în care se află instrucțiunile care apelează procedura `fact` (de fapt se apelează funcția pentru calculul factorialului definită în fișierul `ffact.m`) și prin care se calculează cei trei parametri `P`, `A` și `C`.

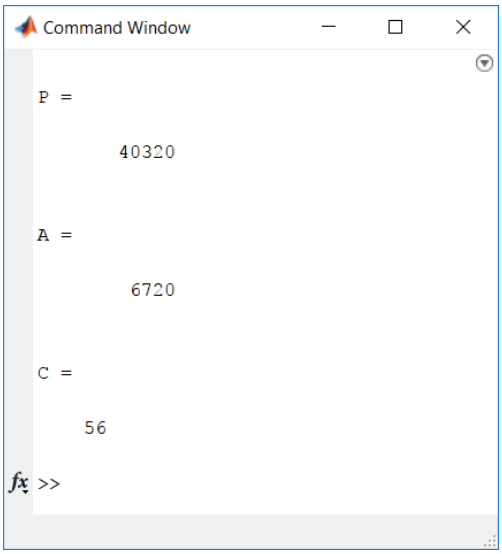
Rezultatul lansării în execuție a fișierului `script` este prezentat în figura 3.81, c). Rezultatele obținute se pot verifica folosind funcția MATLAB predefinită pentru calculul factorialului, denumită `factorial`. În urma fazei de verificare se constată că rezultatele obținute sunt corecte, prin urmare atât algoritmul cât și modul de implementare al acestuia prin cele două fișiere (fișierul `function` și fișierul `script`) sunt corecte.



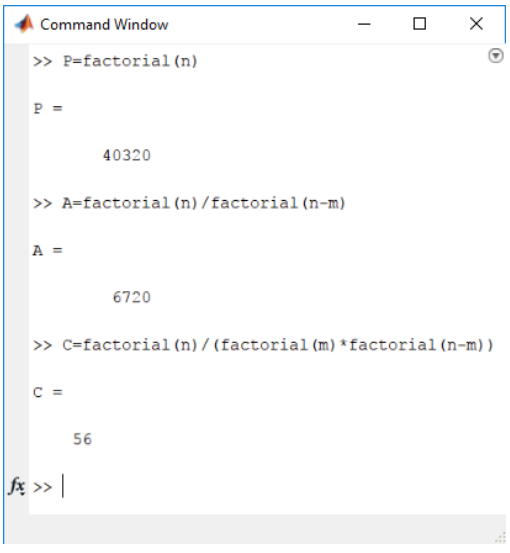
a) fișierul `function` pentru definirea schemei logice secundare



b) fișierul `script` pentru definirea schemei logice principale



c) rezultatele obținute



d) verificarea rezultatelor

Figura 3.81. Calculul parametrilor P_n, A_n^m și C_n^m.

Problema 3.32

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei primelor `n+1` numere naturale pare:

$$S_p = 0 + 2 + 4 + \cdots + 2n = \sum_{k=0}^{n} 2k$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

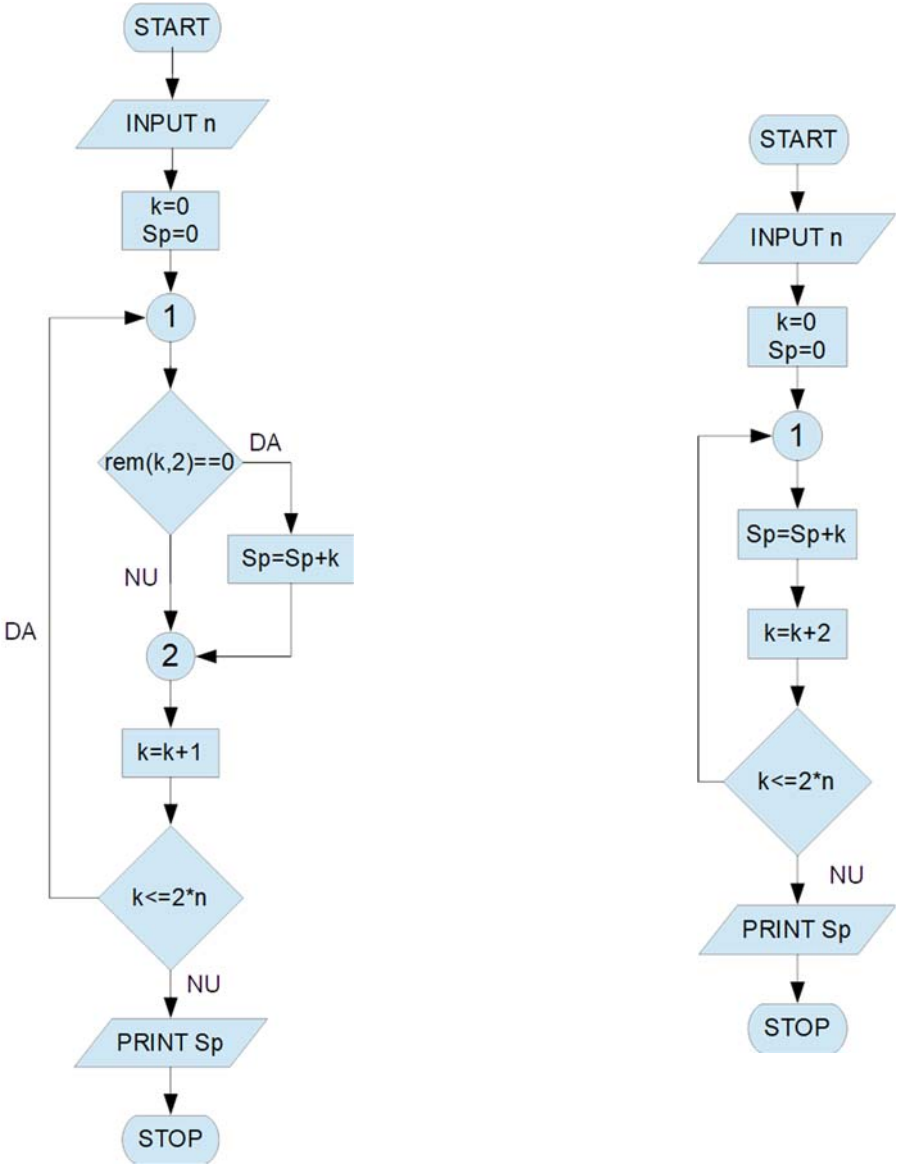
Să se verifice pentru `n=5`, `n=10` și `n=15`.

Rezolvare

Rezolvarea problemei se poate face prin două metode:

- **Metoda `rem`**. Parcurgerea fiecărui număr natural de la 0 la 2n și adunarea doar a acelor elemente k pentru care restul împărțirii la 2 este 0 (adică doar a numerelor pare). Restul împărțirii la 2 a numărul k se calculează cu instrucțiunea rem(k,2).
- **Metoda k=k+2**. Parcurgerea doar a elementelor pare de la 0 la 2n prin utilizarea unei instrucțiuni de incrementare de forma k=k+2.

Schemele logice sunt prezentate în figura 3.82.

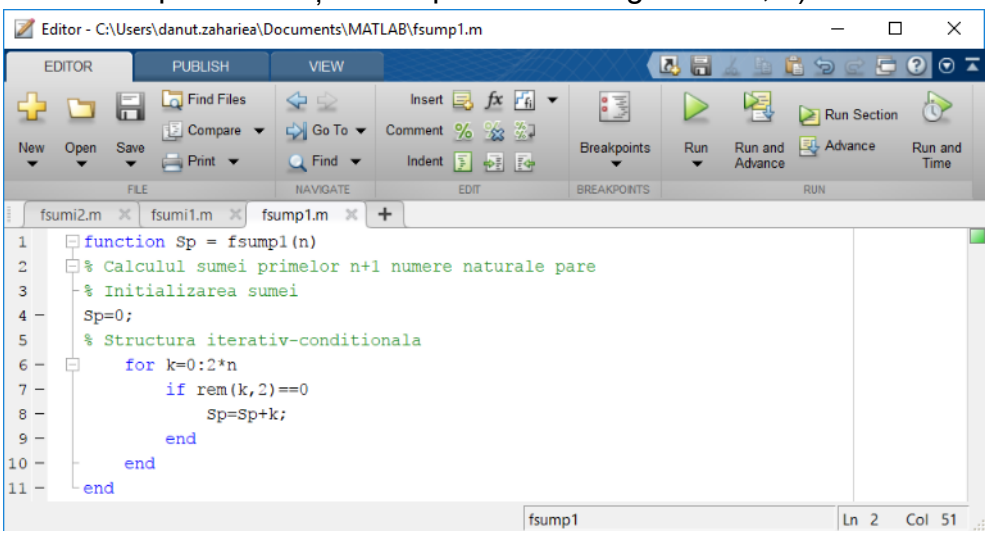


a) metoda `rem` b) metoda `k=k+2`

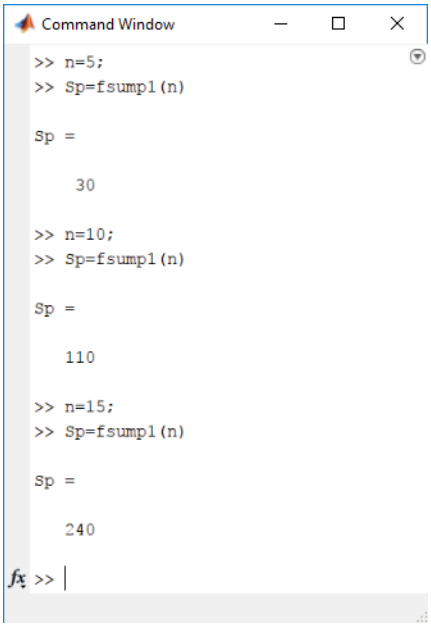
Figura 3.82. Schema logică pentru calculul sumei elementelor pare.

Funcție care implementează algoritmul descris în schema logică din figura 3.82, a) este definită în fișierul de tip `function` prezentat în figura 3.83, a). Numele funcției este `fsump1` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsump1.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se află o structură iterativă de tip `for` care va parcurge cu pasul `1` toate numerele `k` de la `0` la `2*n` și o structură alternativă cu o singură ramură de tip `if` prin care se vor identifica doar elementele pare, care se vor aduna iterativ, obținându-se în final suma elementelor pare. Identificarea numerelor pare se realizează prin calculul restului împărțirii dintre valoarea curentă `k` a contorului și 2. Rezultatul `Sp`, obținut în urma evaluării funcției, reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.83, b).



a) fișierul `function`

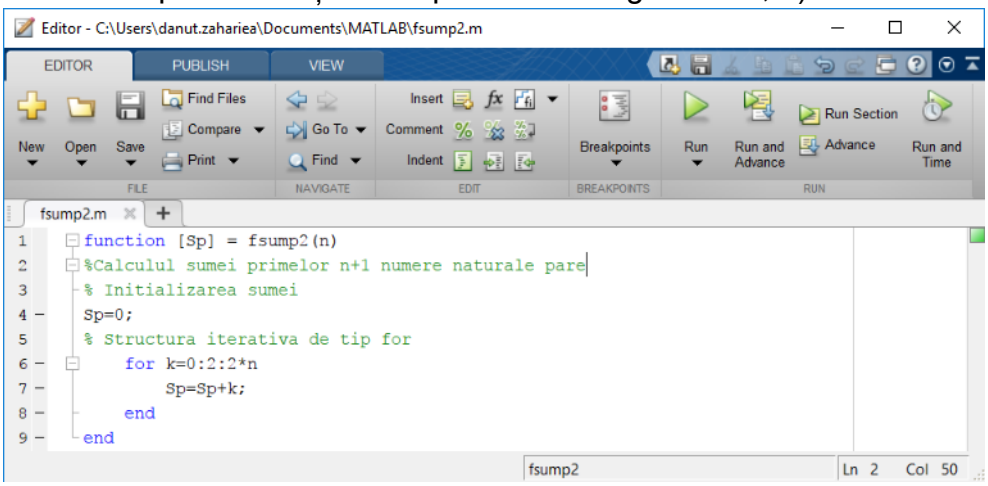


b) rezultatul obținut

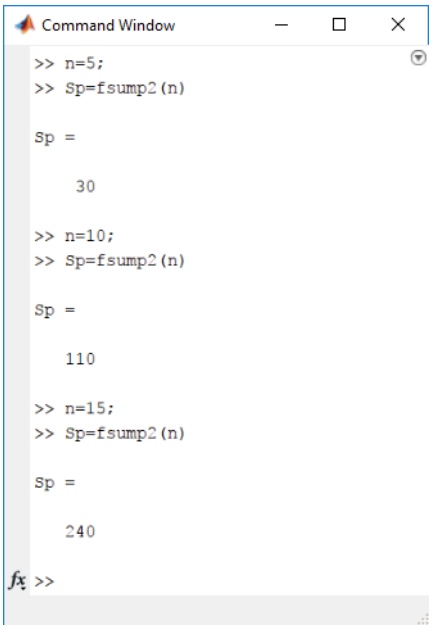
Figura 3.83. Fișier `function` pentru calculul sumei elementelor pare prin metoda `rem`.

Funcție care implementează algoritmul descris în schema logică din figura 3.82, b) este definită în fișierul de tip `function` prezentat în figura 3.84, a). Numele funcției este `fsump2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsump2.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se află o structură iterativă de tip `for` care va parcurge cu pasul `2` toate numerele `k` de la `0` la `2*n`, deci doar elementele pare, care se vor aduna iterativ, obținându-se în final suma elementelor pare. Identificarea numerelor pare se realizează prin inițializarea `k=0` și incrementarea particulară a contorului, `k=k+2`. Rezultatul `Sp`, obținut în urma evaluării funcției, reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.84, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.84. Fișier `function` pentru calculul sumei elementelor pare prin metoda `k=k+2`.

Problema 3.33

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei primelor `n+1` numere naturale impare:

$$S_i = 1 + 3 + \cdots + (2n+1) = \sum_{k=0}^{n} (2k+1)$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

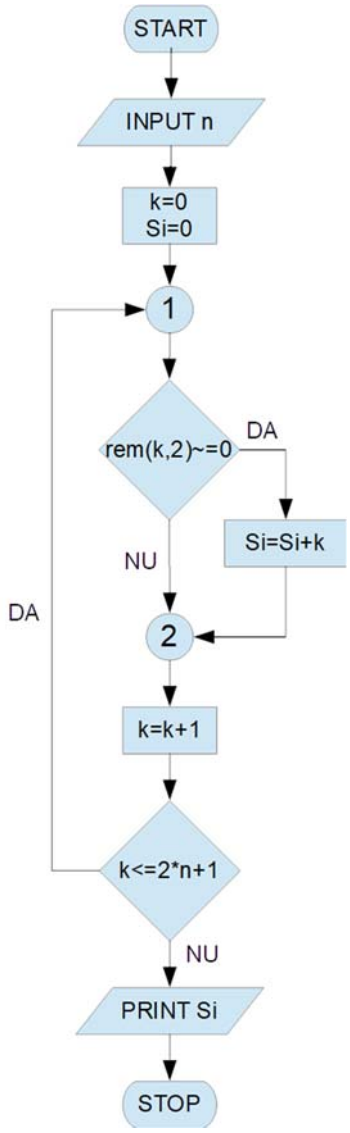
Să se verifice pentru `n=5`, `n=10` și `n=15`.

Rezolvare

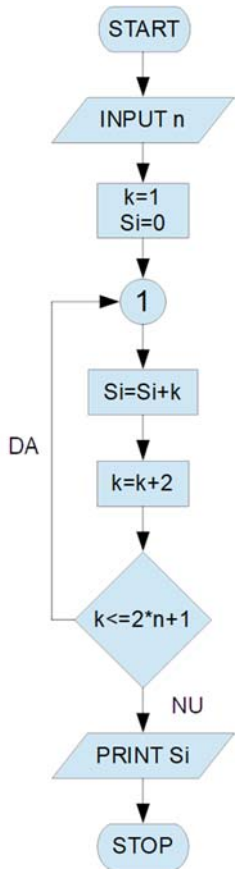
Rezolvarea problemei se poate face prin două metode:

- **Metoda `rem`**. Parcurgerea fiecărui număr natural de la 0 la 2n+1 și adunarea doar a acelor elemente k pentru care restul împărțirii la 2 este diferit de 0 (adică doar a numerelor impare). Restul împărțirii la 2 a numărul k se calculează cu instrucțiunea rem(k,2).
- **Metoda k=k+2**. Parcurgerea doar a elementelor impare de la 1 la 2n+1 prin utilizarea unei instrucțiuni de incrementare de forma k=k+2.

Schemele logice sunt prezentate în figura 3.85.



a) metoda `rem`

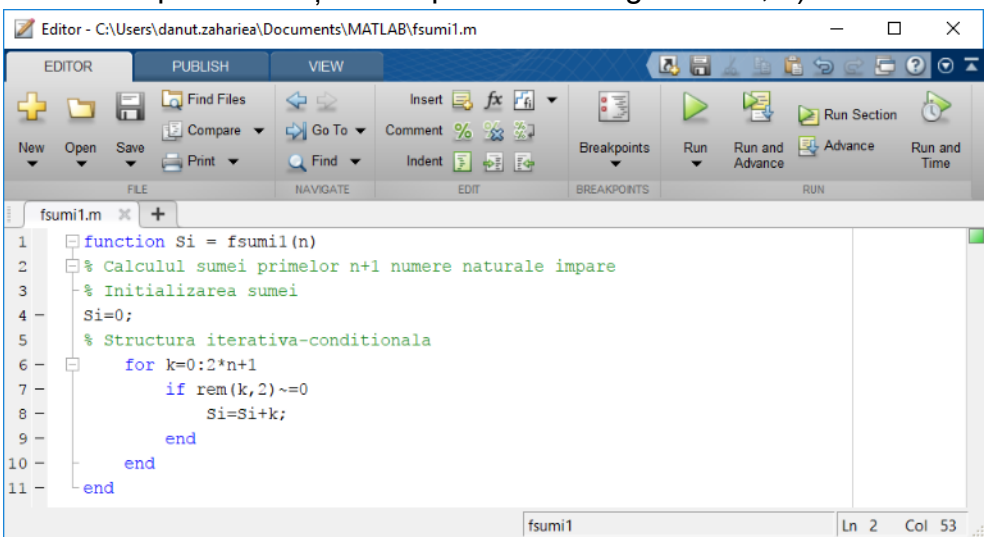


b) metoda `k=k+2`

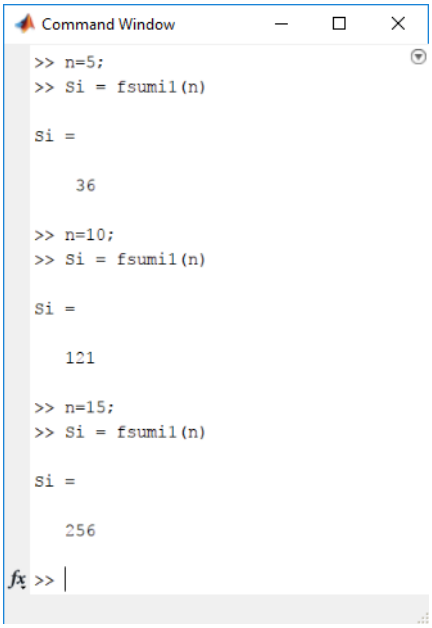
Figura 3.85. Schema logică pentru calculul sumei elementelor impare.

Funcție care implementează algoritmul descris în schema logică din figura 3.85, a) este definită în fișierul de tip `function` prezentat în figura 3.86, a). Numele funcției este `fsumi1` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumi1.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se află o structură iterativă de tip `for` care va parcurge cu pasul `1` toate numerele `k` de la `0` la `2*n+1` și o structură alternativă cu o singură ramură de tip `if` prin care se vor identifica doar elementele impare, care se vor aduna iterativ, obținându-se în final suma elementelor impare. Identificarea numerelor impare se realizează prin calculul restului împărțirii dintre valoarea curentă `k` a contorului și 2. Rezultatul `Si`, obținut în urma evaluării funcției, reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.86, b).



a) fișierul `function`

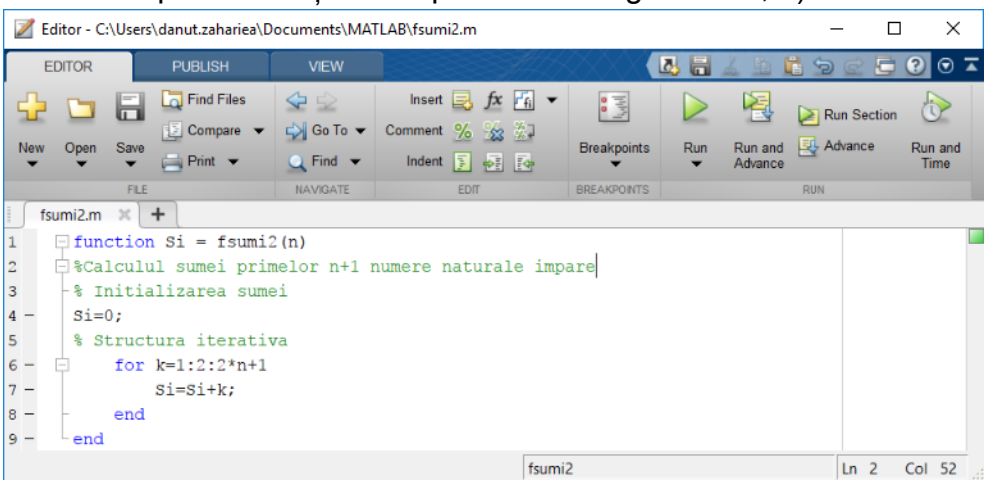


b) rezultatul obținut

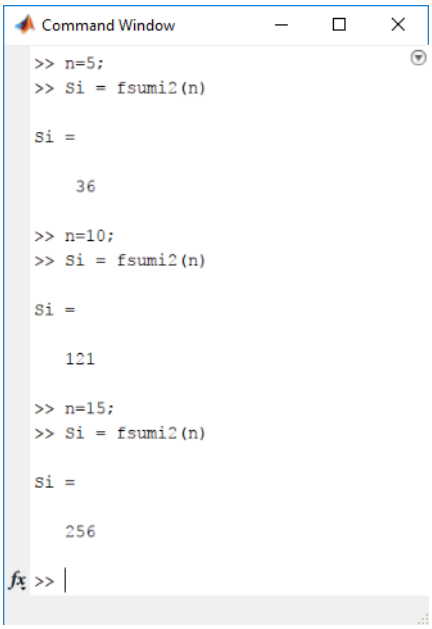
Figura 3.86. Fișier `function` pentru calculul sumei elementelor impare prin metoda `rem`.

Funcție care implementează algoritmul descris în schema logică din figura 3.85, b) este definită în fișierul de tip `function` prezentat în figura 3.87, a). Numele funcției este `fsumi2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumi2.m`. Funcția acceptă un singur parametru de intrare, respectiv numărul `n`. În corpul funcției se află o structură iterativă de tip `for` care va parcurge cu pasul `2` toate numerele `k` de la `1` la `2*n+1`, deci doar elementele impare, care se vor aduna iterativ, obținându-se în final suma elementelor impare. Identificarea numerelor impare se realizează prin inițializarea `k=1` și incrementarea particulară a contorului, `k=k+2`. Rezultatul `Si`, obținut în urma evaluării funcției, reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a numărului `n`. La apelare, numărul `n` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.87, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.87. Fișier `function` pentru calculul sumei elementelor impare prin metoda `k=k+2`.

Problema 3.34

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei elementelor vectorului a:

$$S = a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[2 4 6 8 10]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.88.

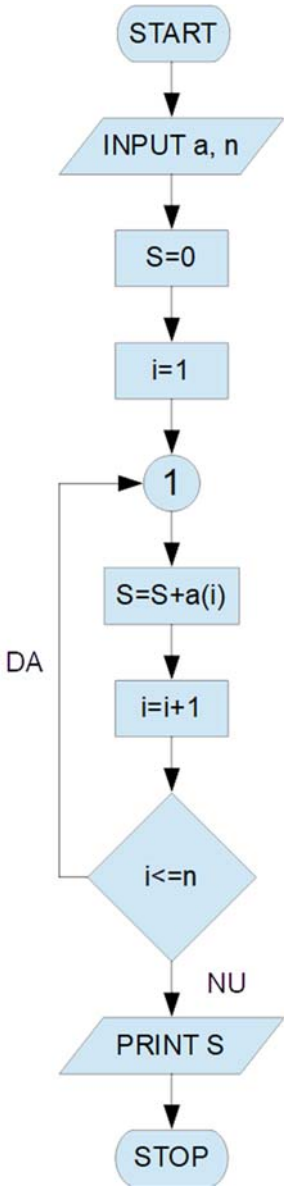


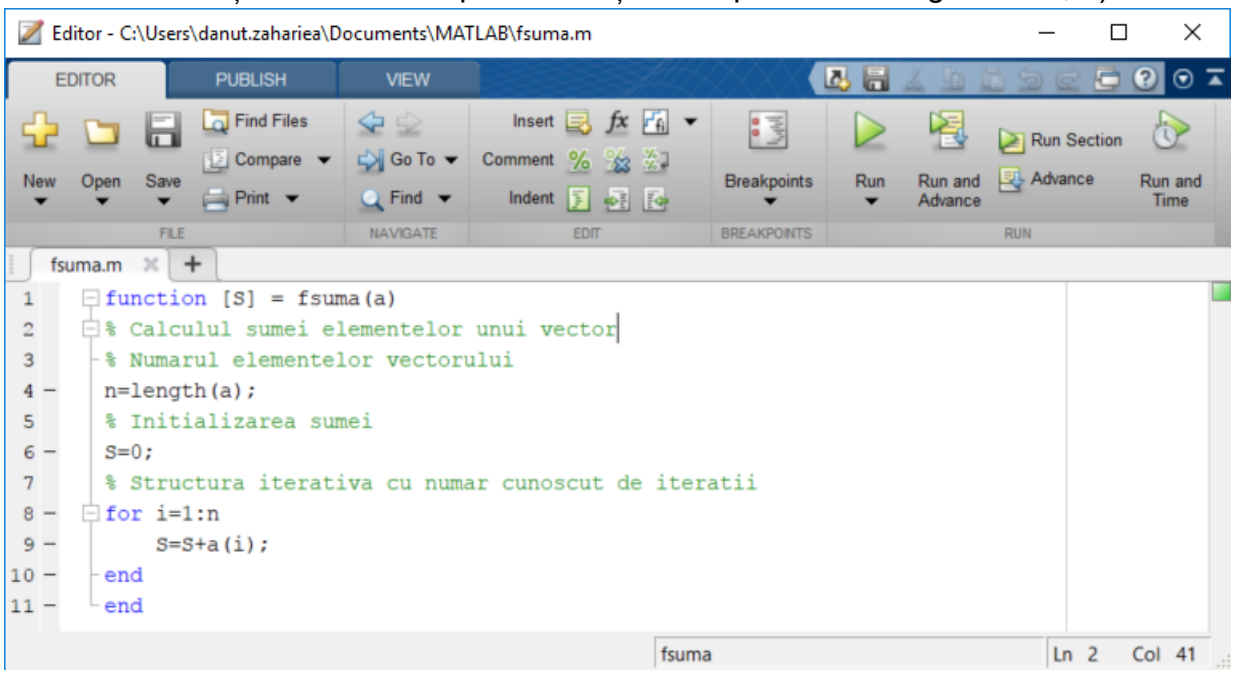
Figura 3.88. Schema logică pentru calculul sumei elementelor unui vector folosind o structură iterativă cu număr cunoscut de iterații.

Observații

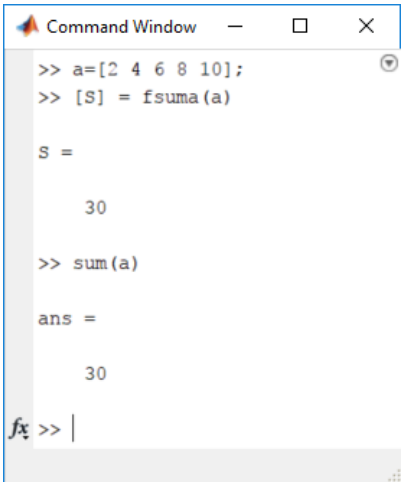
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru suma `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugare acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)`) la valoarea anterioară a sumei (`S=S+a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează ultima valoare calculată a sumei.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.89, a). Numele funcției este `fsuma` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsuma.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. Determinarea numărului de elemente ale vectorului `a` se realizează cu instrucțiunea `n=length(a)`. Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. Faza de inițializare conține doar inițializarea sumei `S=0`, (elementul neutru pentru adunare). La fiecare iterație, identificată prin valoarea curentă a contorului `i`, suma `S` se recalculează prin adăugarea valorii curente a vectorului la valoarea anterioară a sumei, `S=S+a(i)`. Parametrul de ieșire al funcției este suma `S`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.89, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.89. Fișier `function` pentru calculul sumei elementelor unui vector.

Problema 3.35

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu test inițial să se realizeze o schemă logică pentru descrierea algoritmului de determinare a produsului elementelor vectorului a:

$$P = a_1 \cdot a_2 \cdot \dots \cdot a_n = \prod_{i=1}^{n} a_i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 3 5 7 9]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `prod`.

Rezolvare

Schema logică este prezentată în figura 3.90.

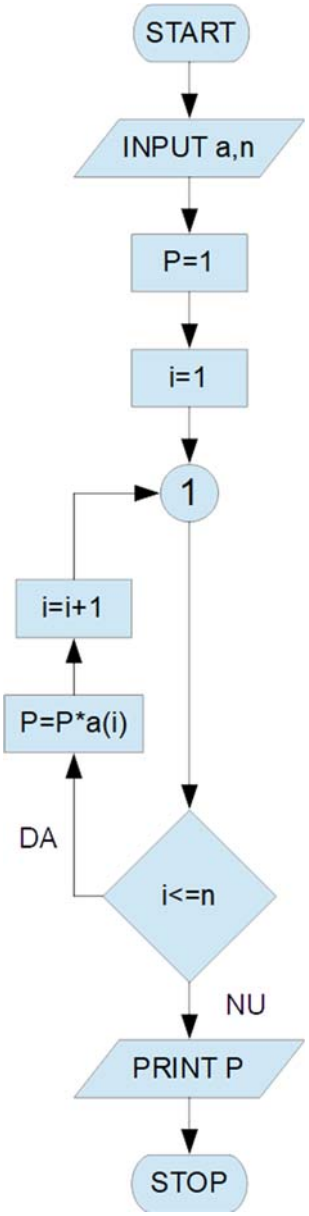


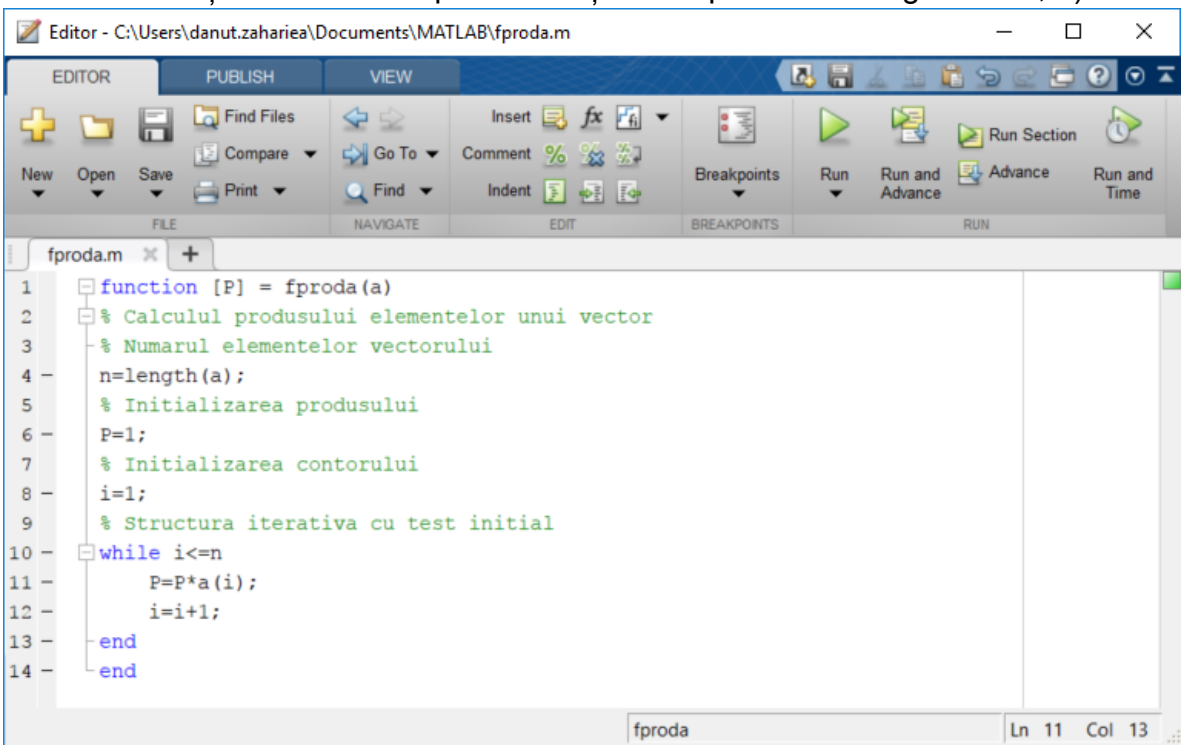
Figura 3.90. Schema logică pentru calculul produsului elementelor unui vector folosind o structură iterativă cu test inițial.

Observații

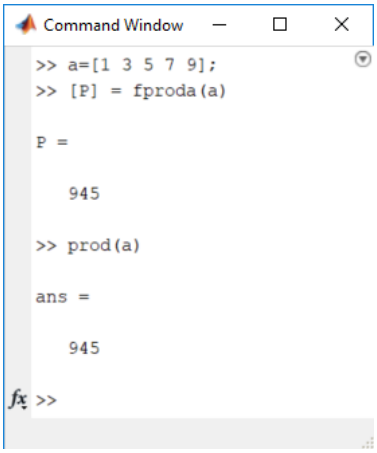
- Schema logică se bazează pe o structură iterativă cu test inițial.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru produsul `P=1` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- După faza de inițializare a algoritmului urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se intră în corpul structurii iterative și se recalculează valoare produsului prin înmulțirea acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)`) cu valoarea anterioară a produsului (`P=P*a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu test inițial și se afișează ultima valoare calculată a produsului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.91, a). Numele funcției este `fproda` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fproda.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. Determinarea numărului de elemente ale vectorului `a` se realizează cu instrucțiunea `n=length(a)`. Calculul produsului `P` se efectuează cu o structură iterativă cu test inițial. Faza de inițializare conține pe lângă inițializarea produsului `P=1`, (elementul neutru pentru înmulțire) și inițializarea contorului `i=1`. La fiecare iterație, identificată prin valoarea curentă a contorului `i`, produsul `P` se recalculează prin înmulțirea valorii curente a vectorului cu valoarea anterioară a produsului, `P=P*a(i)`. Parametrul de ieșire al funcției este produsul `P`.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.91, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.91. Fișier `function` pentru calculul produsului elementelor unui vector.

Problema 3.36

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a mediei aritmetice a elementelor vectorului a:

$$m_a = \frac{a_1 + a_2 + \cdots + a_n}{n} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `mean`.

Rezolvare

Schema logică este prezentată în figura 3.92.

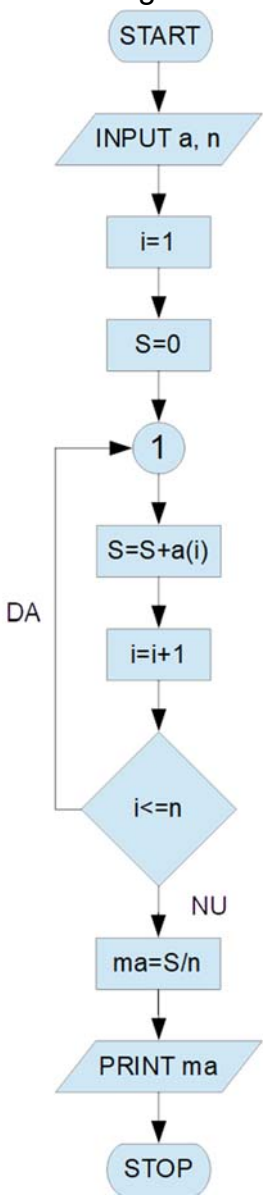


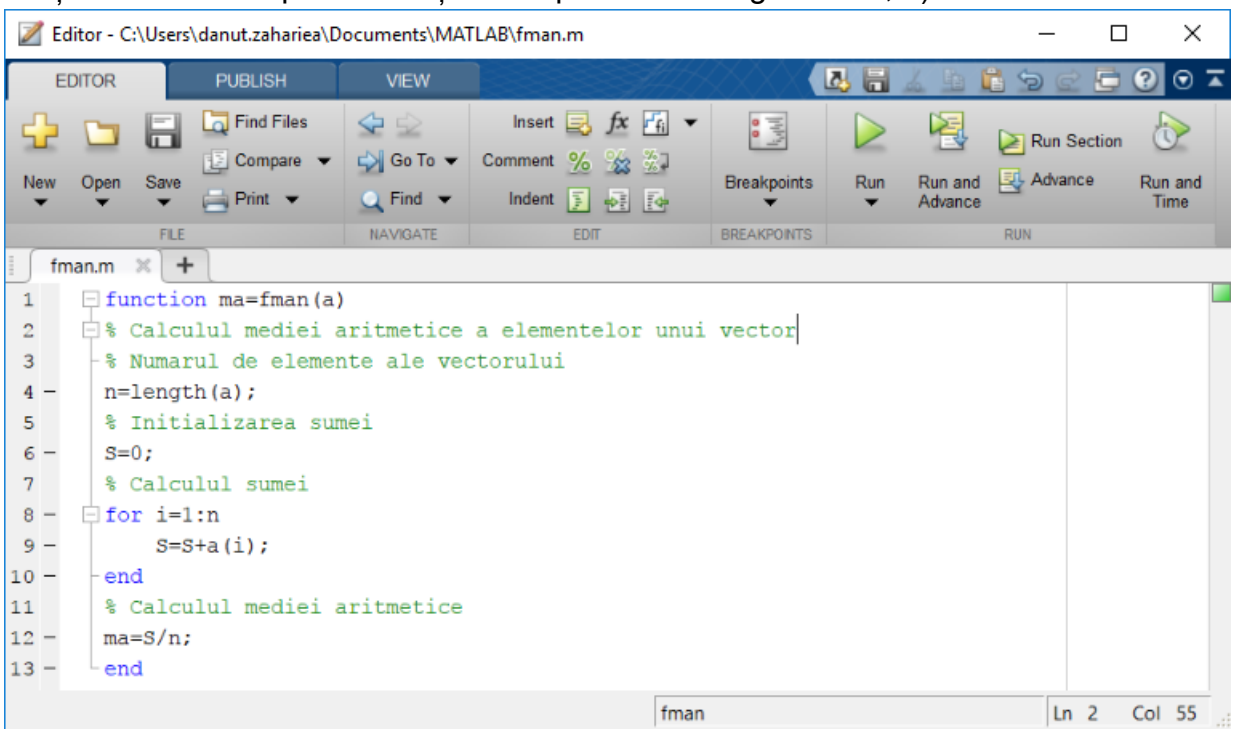
Figura 3.92. Schema logică pentru calculul mediei aritmetice a elementelor unui vector.

Observații

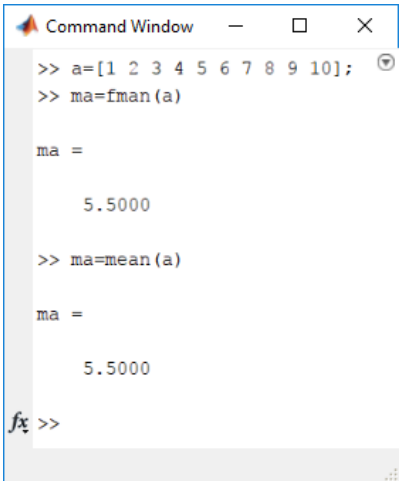
- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare pentru calculul sumei `S` a elementelor vectorului `a`, urmată de calculul propriu zis al mediei aritmetice `ma`.
- Datele de intrare ale algoritmului sunt elementele `a` și dimensiunea `n` a vectorului.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru suma `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugare acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)`) la valoarea anterioară a sumei (`S=S+a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel suma elementelor vectorului `a`.
- Urmează apoi calculul efectiv al mediei aritmetice `ma=S/n`, media aritmetică `ma` fiind și rezultatul final al algoritmului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.93, a). Numele funcției este `fman` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fman.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea sumei (`S=0`, elementul neutru pentru adunare). Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. După obținerea sumei elementelor vectorului se calculează media aritmetică cu instrucțiunea `ma=S/n`. Variabila `ma` este parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.93, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.93. Fișier `function` pentru calculul mediei aritmetice a elementelor unui vector.

Problema 3.37

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a mediei geometrice a elementelor vectorului a:

$$m_g = \sqrt[n]{a_1 \cdot a_2 \cdot \dots \cdot a_n} = \left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}}$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `prod`.

Rezolvare

Schema logică este prezentată în figura 3.94.

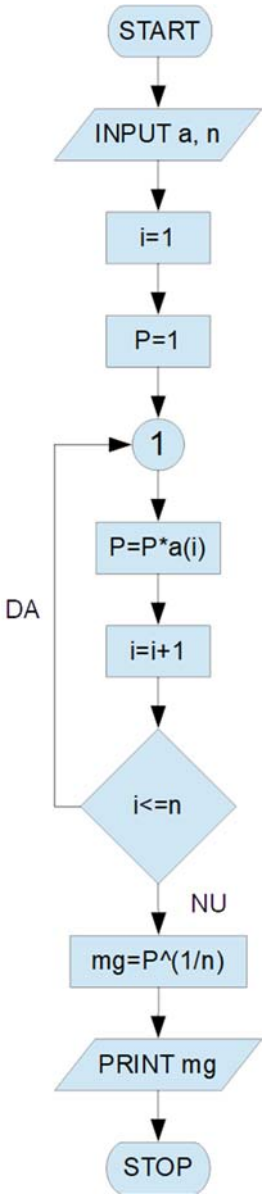


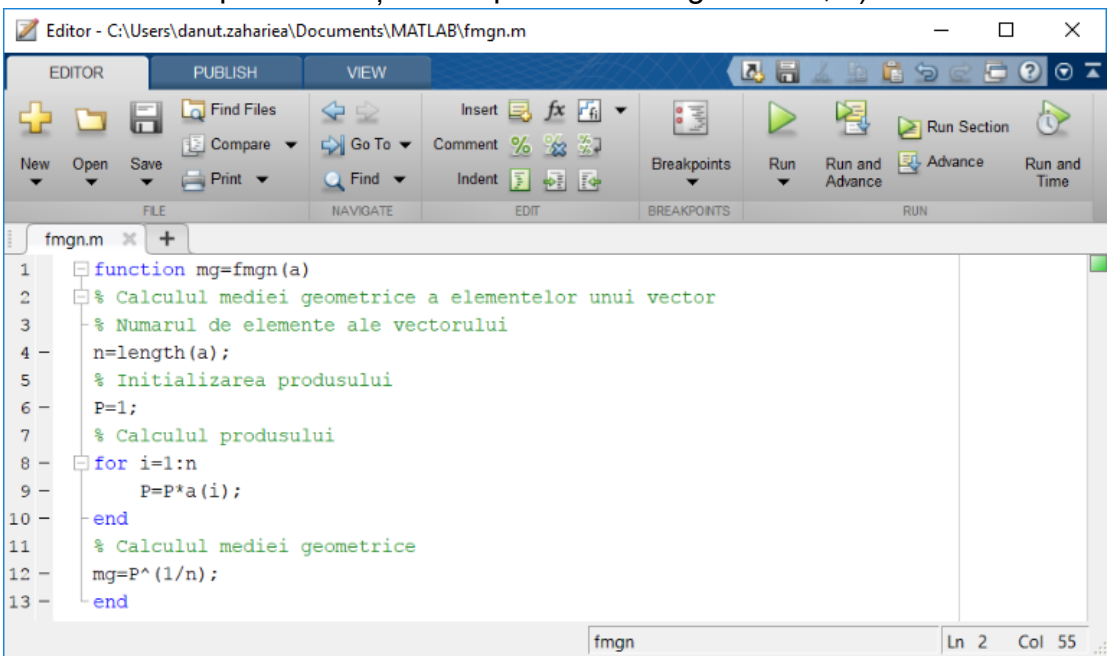
Figura 3.94. Schema logică pentru calculul mediei geometrice a elementelor unui vector.

Observații

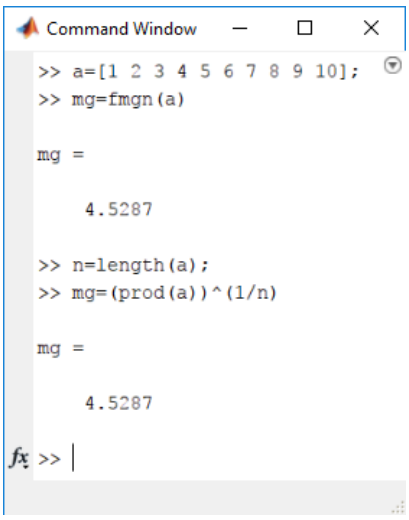
- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare pentru calculul produsului `P` al elementelor vectorului `a`, urmată de calculul propriu zis al mediei geometrice `mg`.
- Datele de intrare ale algoritmului sunt elementele `a` și dimensiunea `n` a vectorului.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru produsul `P=1` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare produsului prin înmulțirea acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)`) cu valoarea anterioară a produsului (`P=P*a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea produsului și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel produsul elementelor vectorului `a`.
- Urmează apoi calculul efectiv al mediei geometrice `mg=P^(1/n)`, media geometrică `mg` fiind și rezultatul final al algoritmului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.95, a). Numele funcției este `fmgn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmgn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea produsului (`P=1`, elementul neutru pentru înmulțire). Calculul produsului `P` se efectuează cu o structură iterativă cu număr cunoscut de iterații. După obținerea produsului elementelor vectorului se calculează media geometrică cu instrucțiunea `mg=P^(1/n)`. Variabila `mg` este parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.95, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.95. Fișier `function` pentru calculul mediei geometrice a elementelor unui vector.

Problema 3.38

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a mediei armonice a elementelor vectorului a:

$$m_r = \frac{n}{\sum_{i=1}^{n} \frac{1}{a_i}}$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.96.

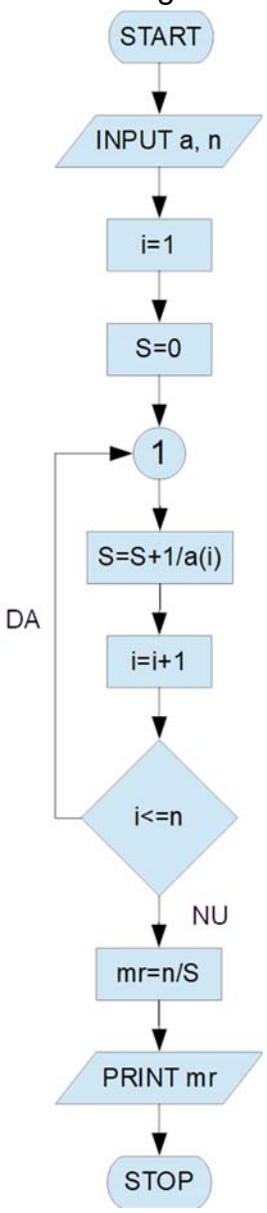


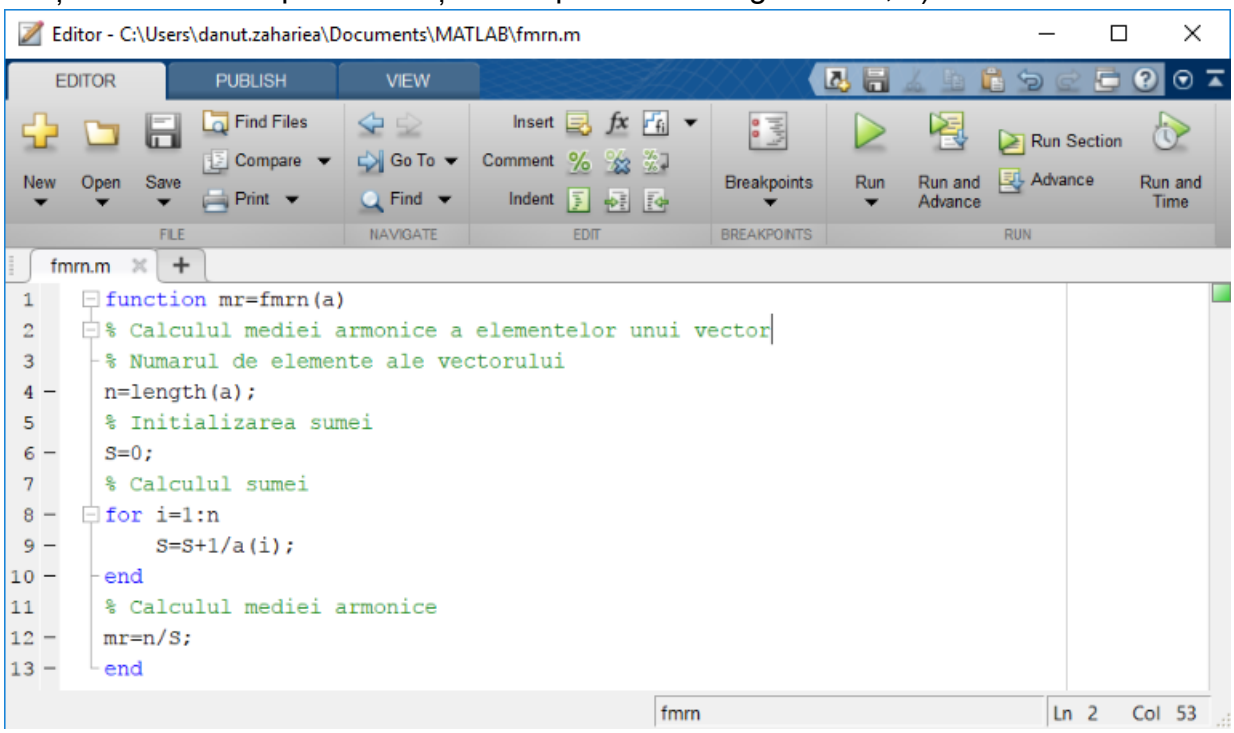
Figura 3.96. Schema logică pentru calculul mediei armonice a elementelor unui vector.

Observații

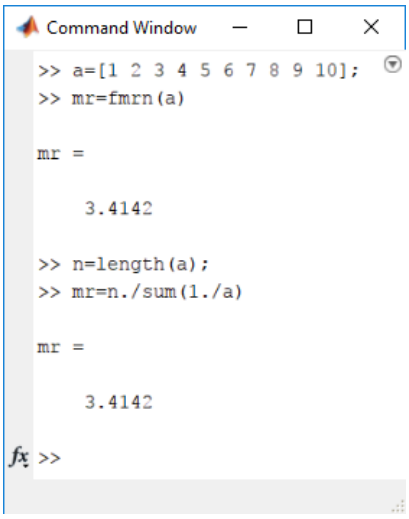
- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare pentru calculul sumei `S` a inverselor elementelor vectorului `a`, urmată de calculul propriu zis al mediei armonice `mr`.
- Datele de intrare ale algoritmului sunt elementele `a` și dimensiunea `n` a vectorului.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru suma `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugarea inversului acelei valori a vectorului care corespunde valorii curente a contorului (`1/a(i)`) la valoarea anterioară a sumei (`S=S+1/a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel suma inverselor elementelor vectorului `a`.
- Urmează apoi calculul efectiv al mediei armonice `mr=n/S`, media armonică `mr` fiind și rezultatul final al algoritmului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.97, a). Numele funcției este `fmrn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmrn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea sumei (`S=0`, elementul neutru pentru adunare). Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. După obținerea sumei inverselor elementelor vectorului se calculează media armonică cu instrucțiunea `mr=n/S`. Variabila `mr` este parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.97, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.97. Fișier `function` pentru calculul mediei armonice a elementelor unui vector.

Problema 3.39

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind o structură iterativă cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a mediei pătratice a elementelor vectorului a:

$$m_p = \sqrt{\frac{a_1^2 + a_2^2 + \cdots + a_n^2}{n}} = \left(\frac{1}{n}\sum_{i=1}^{n} a_i^2\right)^{\frac{1}{2}}$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Să se verifice rezultatul obținut folosind funcția MATLAB predefinită `sum`.

Rezolvare

Schema logică este prezentată în figura 3.98.

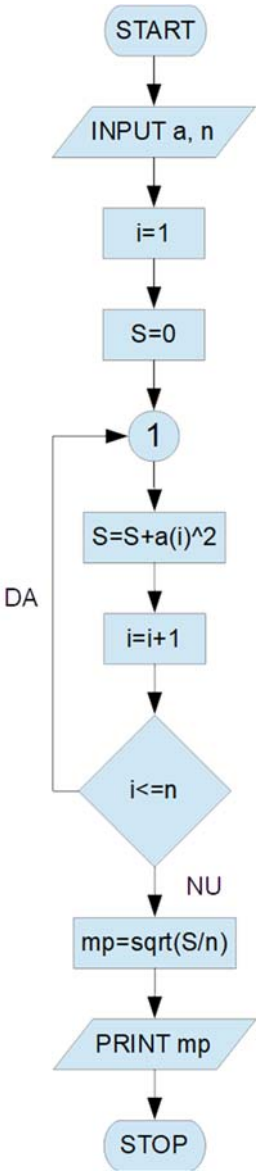


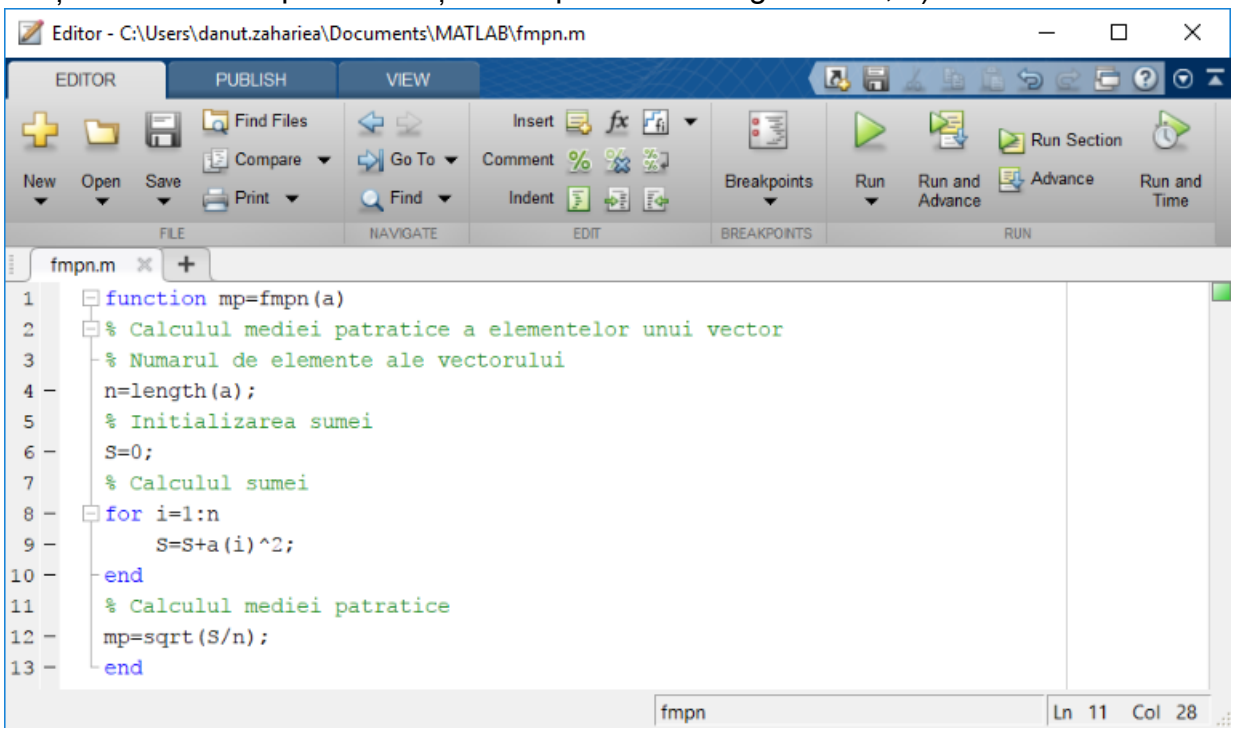
Figura 3.98. Schema logică pentru calculul mediei pătratice a elementelor unui vector.

Observații

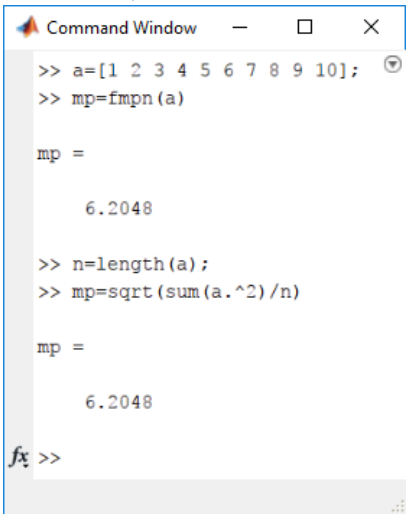
- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare pentru calculul sumei `S` a pătratelor elementelor vectorului `a`, urmată de calculul propriu zis al mediei pătratice `mp`.
- Datele de intrare ale algoritmului sunt elementele `a` și dimensiunea `n` a vectorului.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru suma `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei prin adăugarea pătratului acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)^2`) la valoarea anterioară a sumei (`S=S+a(i)^2`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea sumei și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel suma inverselor elementelor vectorului `a`.
- Urmează apoi calculul efectiv al mediei pătratice `mp=sqrt(S/n)`, media pătratică `mp` fiind și rezultatul final al algoritmului.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.99, a). Numele funcției este `fmpn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmpn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea sumei (`S=0`, elementul neutru pentru adunare). Calculul sumei `S` se efectuează cu o structură iterativă cu număr cunoscut de iterații. După obținerea sumei pătratelor elementelor vectorului se calculează media pătratică cu instrucțiunea `mp=sqrt(S/n)`. Variabila `mp` este parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi, după definirea vectorului `a`. La apelare, elementele vectorului `a` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.99, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.99. Fișier `function` pentru calculul mediei pătratice a elementelor unui vector.

Problema 3.40

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Folosind structuri iterative cu număr cunoscut de iterații să se realizeze o schemă logică pentru descrierea algoritmului de determinare a mediei aritmetice și a abaterii medii pătratice a elementelor vectorului a:

$$\bar{a} = \frac{a_1 + a_2 + \dots + a_n}{n} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

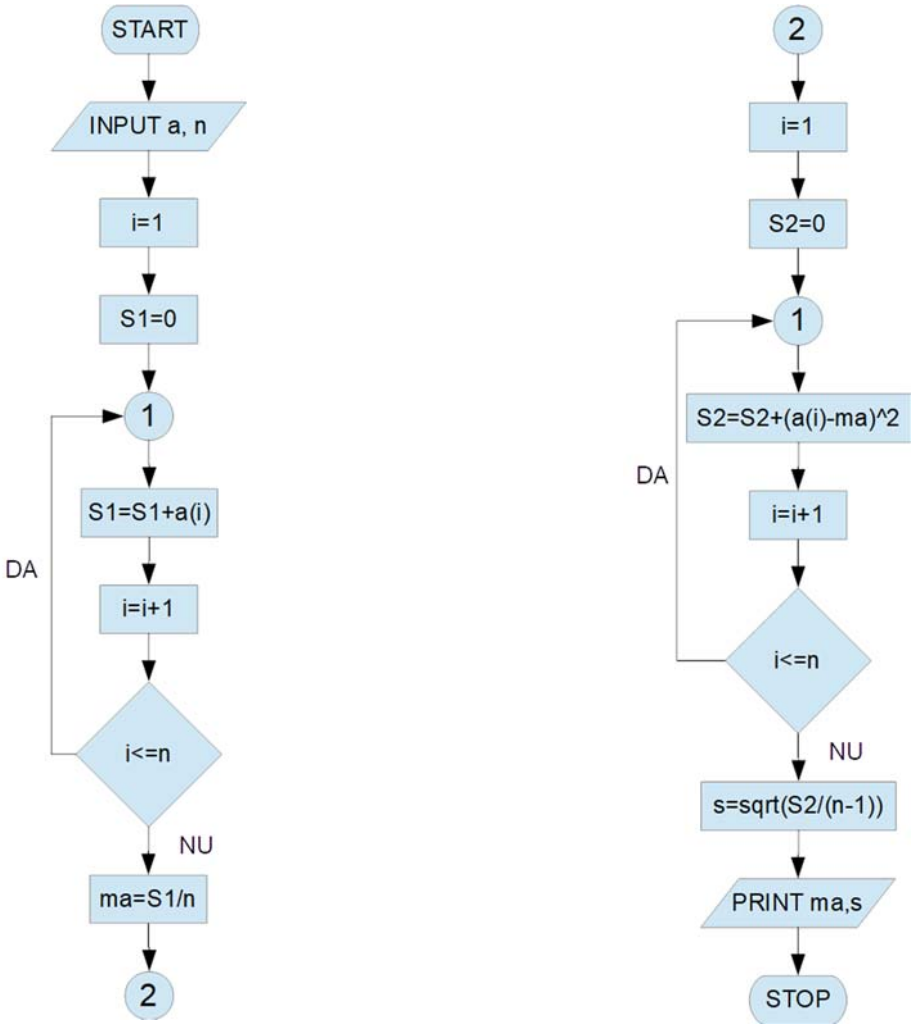
$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2}$$

Să se scrie câte un fișier de tip `function` pentru calculul mediei aritmetice, respectiv a abaterii medii pătratice. Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Să se verifice rezultatele folosind funcțiile MATLAB predefinite `mean` și `std`.

Rezolvare

Schema logică este prezentată în figura 3.100.



a) datele de intrare și calculul mediei aritmetice

b) calculul abaterii medii pătratice și datele de ieșire

Figura 3.100. Schema logică pentru calculul mediei aritmetice și abaterii medii pătratice a elementelor unui vector.

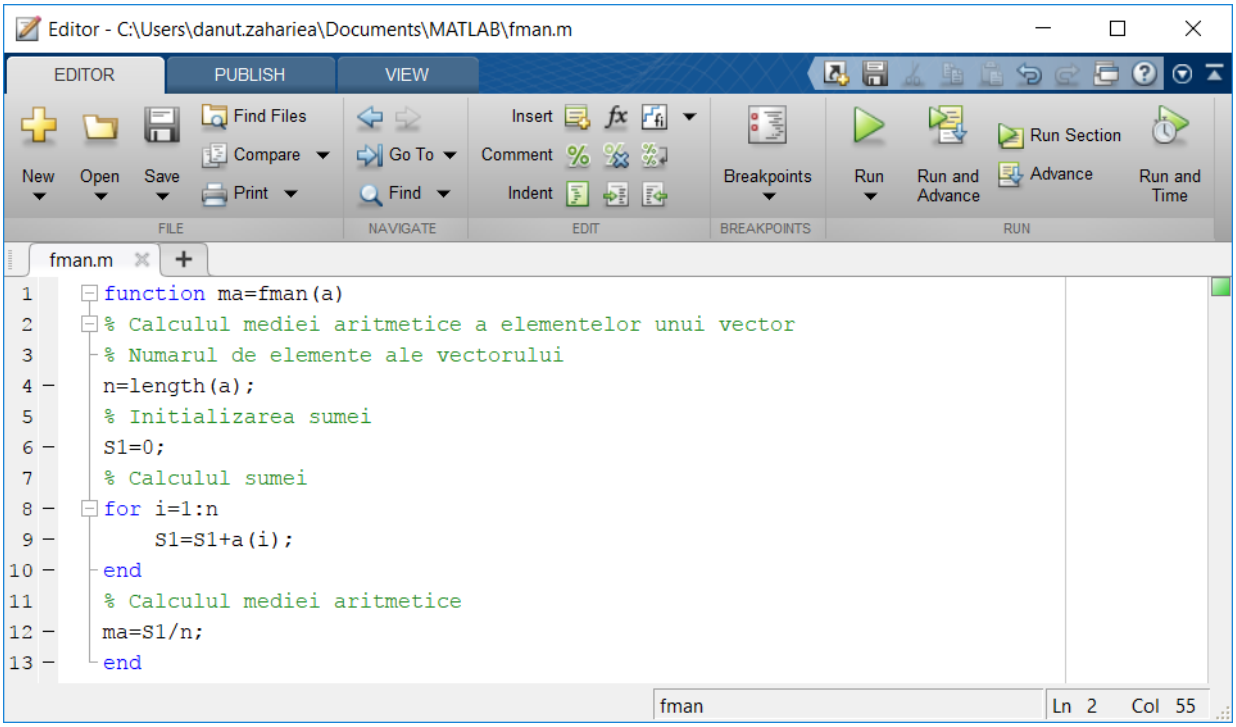
Observații

- Schema logică se bazează pe două structuri iterative cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare: prima structură iterativă pentru calculul mediei aritmetice, iar cea de-a doua structură iterativă pentru calculul abaterii medii pătratice.
- Datele de intrare ale algoritmului sunt elementele `a` și dimensiunea `n` a vectorului.
- Principalele etape ale primei structuri iterative sunt:
  - Faza de inițializare cuprinde două comenzi de atribuire, pentru suma `S1=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
  - La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei `S1` prin adăugare acelei valori a vectorului care corespunde valorii curente a contorului (`a(i)`) la valoarea anterioară a sumei (`S1=S1+a(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
  - După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
  - Dacă expresia logică este adevărată se continuă recalcularea sumei `S1` și incrementarea în continuare a contorului.
  - Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel suma `S1`.
  - Urmează apoi calculul efectiv al mediei aritmetice `ma=S1/n`.
- Principalele etape ale celei de-a doua structuri iterative sunt:
  - Faza de inițializare cuprinde două comenzi de atribuire, pentru suma `S2=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul declarat în faza de introducere a datelor de intrare.
  - La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare sumei `S2` prin adăugare termenului `(a(i)-ma)^2` la valoarea anterioară a sumei, după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
  - După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
  - Dacă expresia logică este adevărată se continuă recalcularea sumei `S1` și incrementarea în continuare a contorului.
  - Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel suma `S2`.
  - Urmează apoi calculul efectiv al abaterii medii pătratice folosind instrucțiunea `s=sqrt(S2/(n-1))`.
- Rezultatele finale ale algoritmului sunt media aritmetică `ma` și abaterea media pătratică `s`.

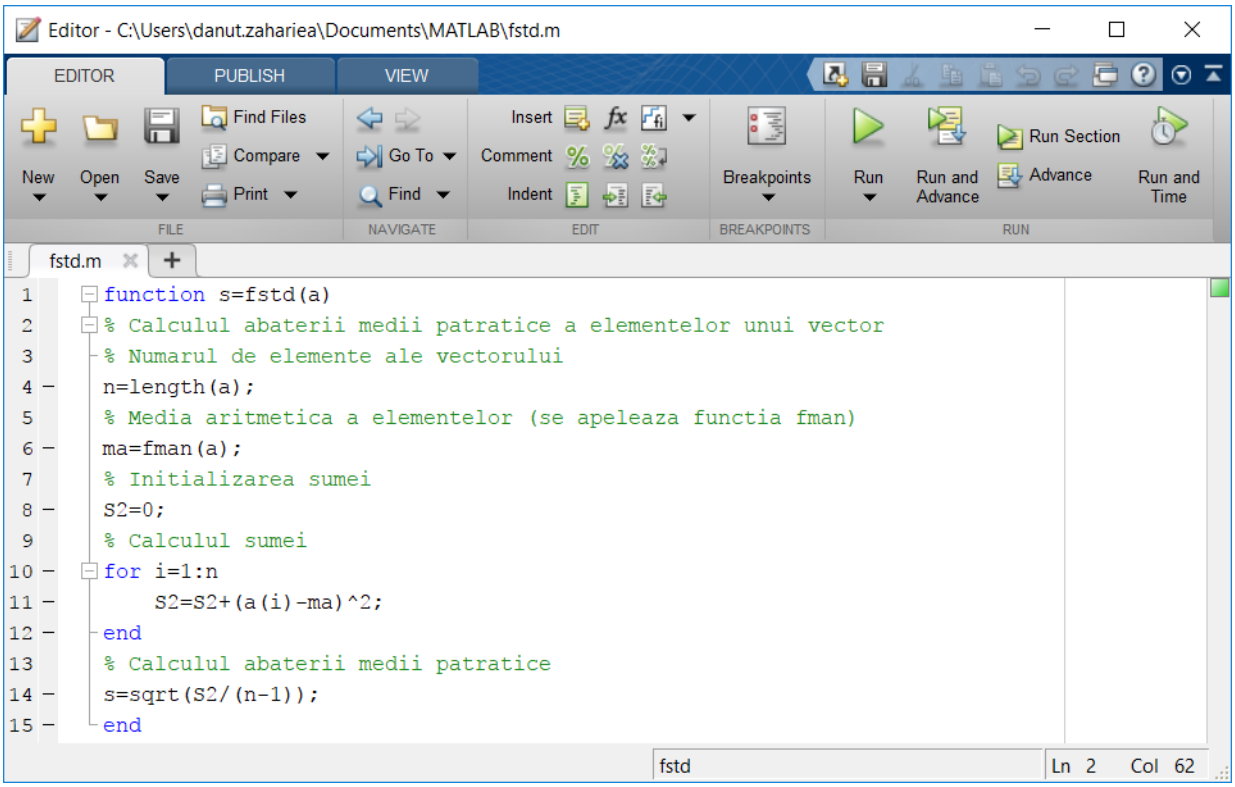
Funcția care implementează algoritmul pentru determinarea mediei aritmetice este definită în fișierul de tip `function` prezentat în figura 3.101, a). Numele funcției este `fman` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fman.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea sumei (`S1=0`, elementul neutru pentru adunare). Calculul sumei `S1` se efectuează cu o structură iterativă cu număr cunoscut de iterații. După obținerea sumei elementelor vectorului se calculează media aritmetică cu instrucțiunea `ma=S1/n`. Variabila `ma` este parametrul de ieșire al funcției.

Funcția care implementează algoritmul pentru determinarea abaterii medii pătratice este definită în fișierul de tip `function` prezentat în figura 3.101, b). Numele funcției este `fstd` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fstd.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea sumei (`S=0`, elementul neutru pentru adunare). Calculul sumei `S2` se efectuează cu o structură iterativă cu număr cunoscut de iterații. La calculul sumei intervine și media aritmetică pentru determinarea căreia s-a apelat funcția `fman`. După obținerea sumei elementelor vectorului se calculează abaterea medie pătratică cu instrucțiunea `ma=sqrt(S2/(n-1))`. Variabila `s` este parametrul de ieșire al funcției.

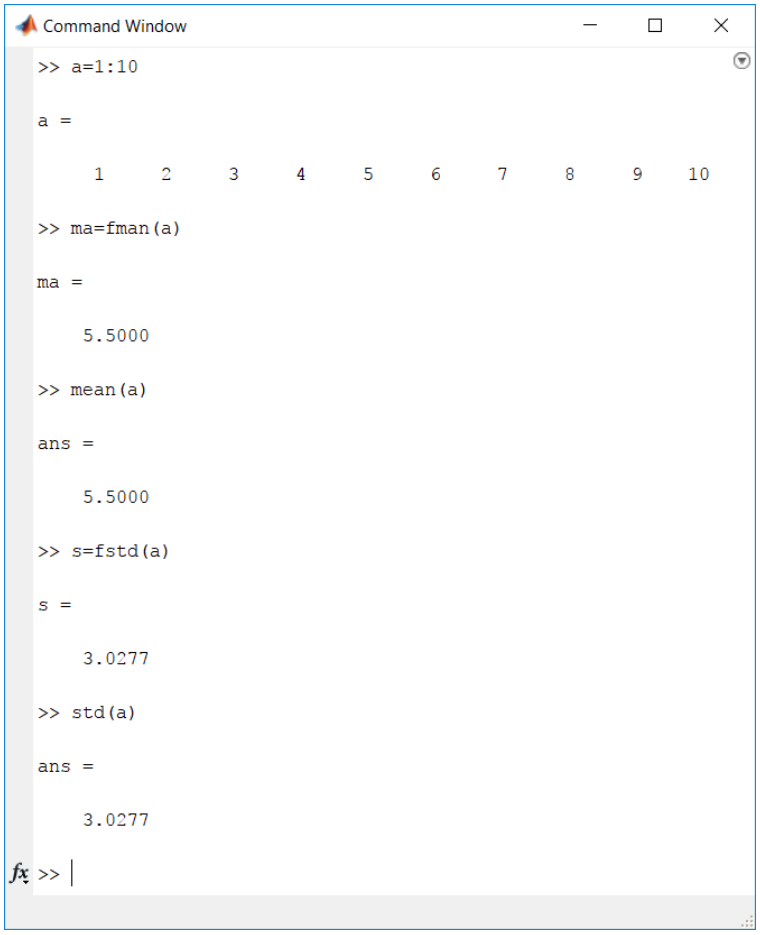
Apelarea celor două funcții se realizează din fereastra de comenzi, după după definirea prealabilă a vectorului `a`. Se apelează apoi pe rând cele două funcții, `fman` pentru determinarea mediei aritmetice și `fstd` pentru determinarea abaterii medii pătratice. Rezultatul apelării celor două funcției este prezentat în figura 3.101, c). Verificarea rezultatelor obținute se efectuează folosind funcțiile MATLAB predefinite `mean` (pentru calculul mediei aritmetice) și `std` (pentru calculul abaterii medii pătratice).



a) fișierul `function` pentru calculul mediei aritmetice



b) fișierul `function` pentru calculul abaterii medii pătratice



c) rezultatul obținut

Figura 3.101. Fișiere `function` pentru calculul mediei aritmetice și abaterii medii pătratice.

Problema 3.41

Se consideră un vector cu n elemente $a = [a_1\ a_2 \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a numărului de elemente pozitive (P), a numărului de elemente negative (N), respectiv a numărului de elemente egale cu zero (O) ale vectorului a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[-1 2 0 4 -5 -6 0 8 9 10]`.

Rezolvare

Schema logică este prezentată în figura 3.102.

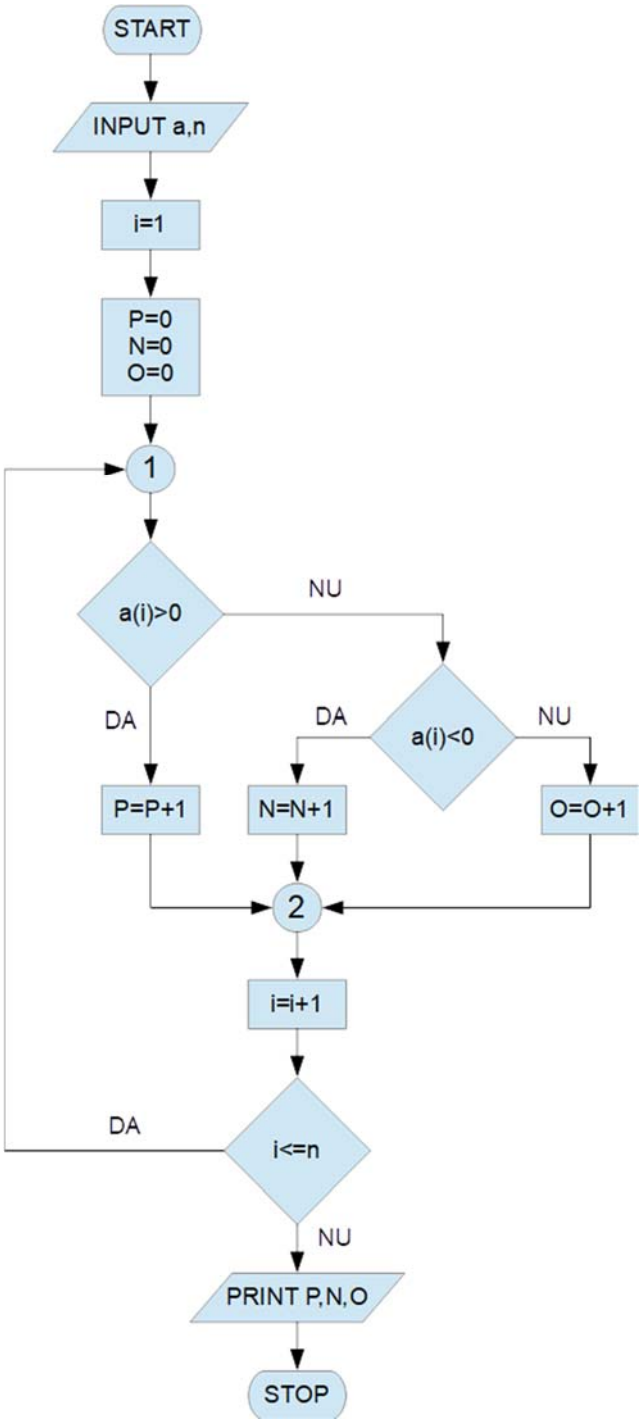
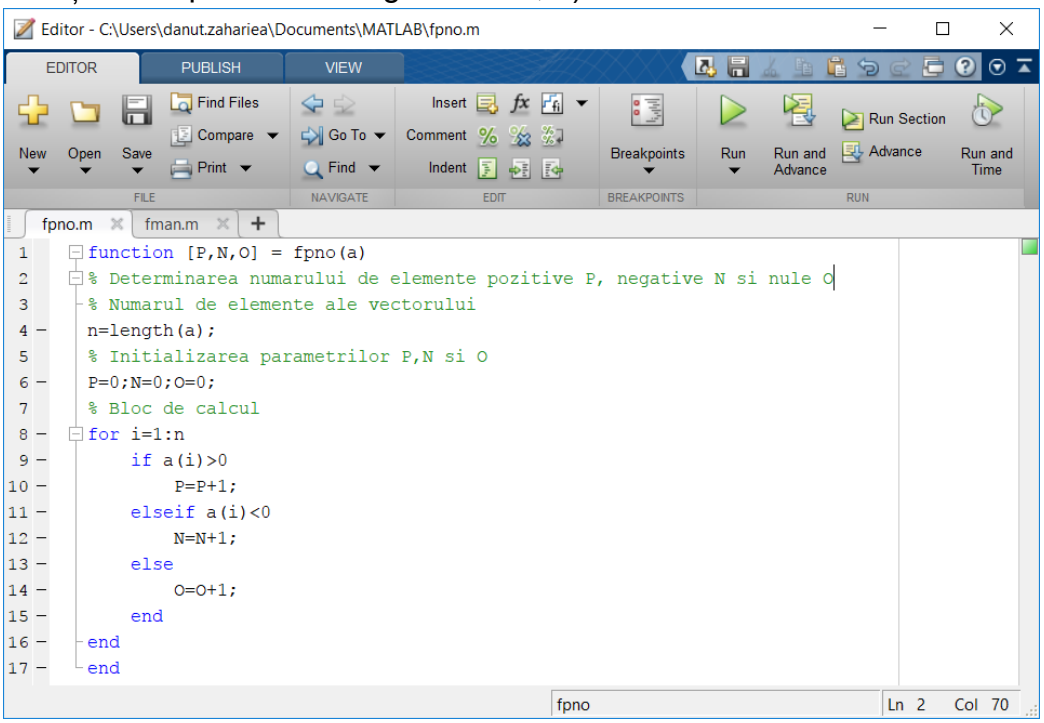
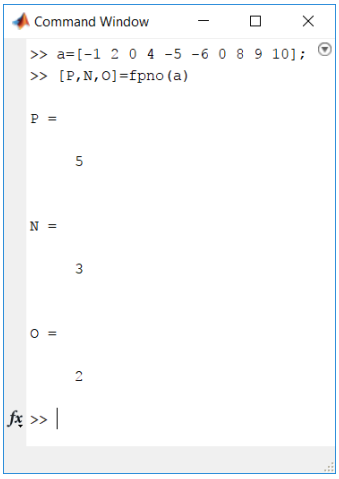


Figura 3.102. Schema logică pentru determinarea numărului de elemente pozitive, negative și nule dintr-un vector.

Funcția care implementează algoritmul pentru determinarea numărului de elemente pozitive, negative și nule dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.103, a). Numele funcției este `fpno` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fpno.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)` urmează inițializarea celor trei parametri `P=0`, `N=0` și `O=0`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if-elseif-else` se verifică fiecare valoare a vectorului și se decide dacă este un număr pozitiv (caz în care se incrementează cu o unitate valoarea parametrului `P`), negativ (caz în care se incrementează cu o unitate valoarea parametrului `N`) sau nul (caz în care se incrementează cu o unitate valoarea parametrului `O`). Valorile obținute pentru cei trei parametri `P`, `N` și `O` sunt parametrii de ieșire ai funcției. Rezultatul apelării funcției este prezentat în figura 3.103, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.103. Fișier `function` pentru determinarea numărului de elemente pozitive, negative și nule dintr-un vector.

Problema 3.42

Se consideră un vector cu n elemente $a = [a_1\ a_2 \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a numărului k și a poziției p a elementelor nule din vectorul a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[-1 2 0 4 -5 -6 0 8 9 0]`.

Rezolvare

Schema logică este prezentată în figura 3.104.

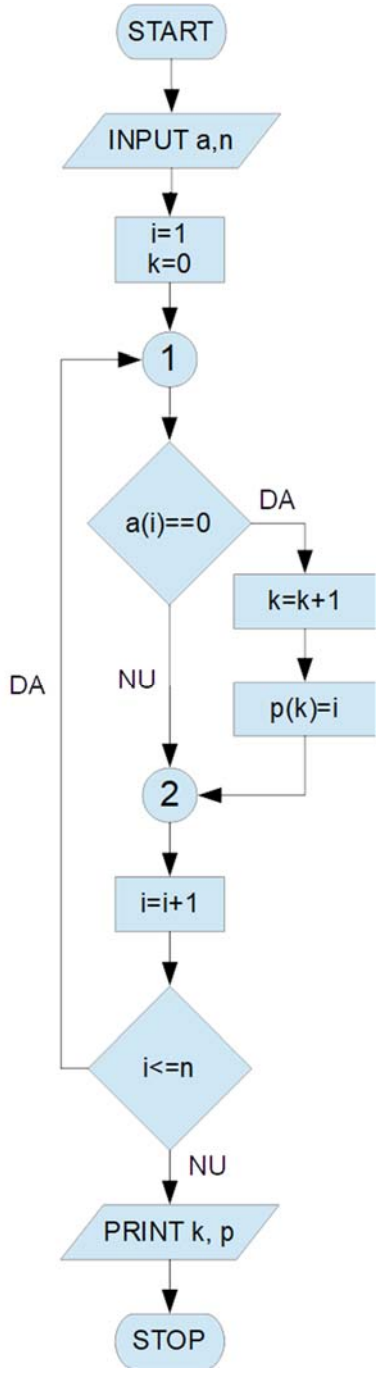
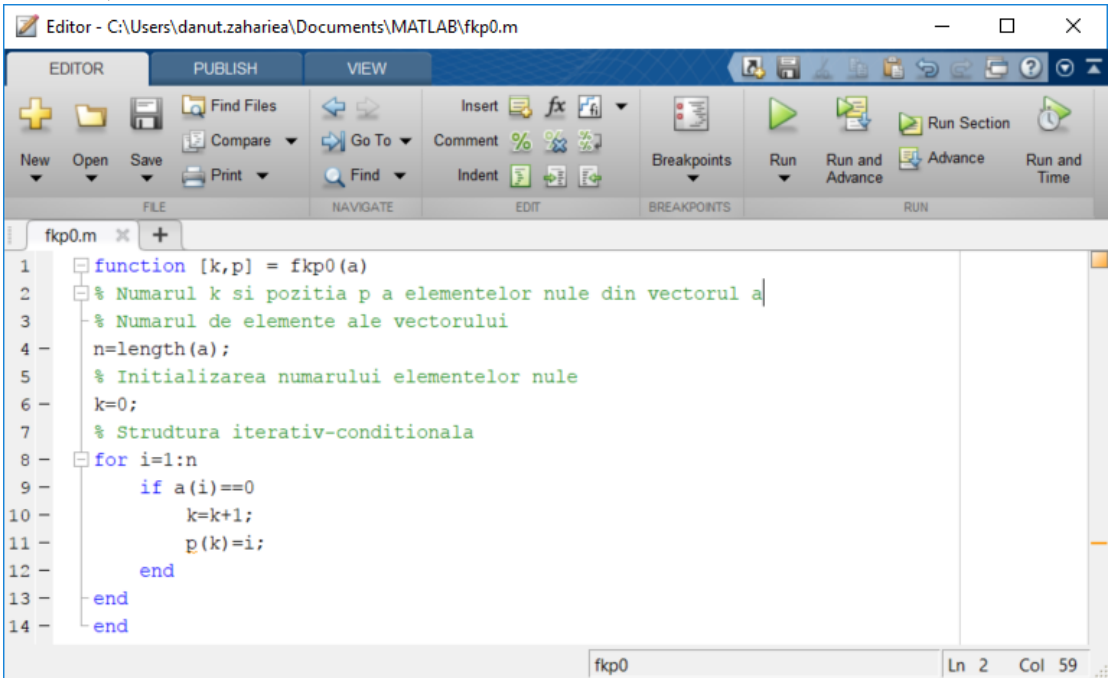


Figura 3.104. Schema logică pentru determinarea numărului k și a poziției p a elementelor nule dintr-un vector.

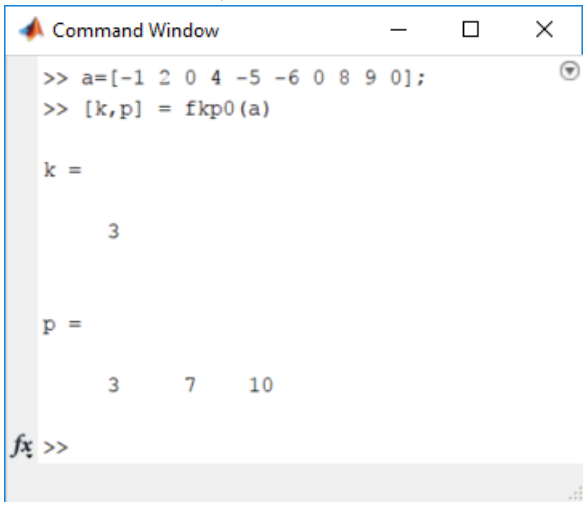
Observații

- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
- Datele de intrare ale algoritmului sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru numărul elementelor nule `k=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este sau nu zero, `a(i)==0`. Dacă valoarea este zero, se incrementează cu o unitate parametrul `k`. Pentru stocarea informației referitoare la poziția elementelor zero se definește o variabilă `p`, care va avea `k` valori, anume indicii `i` ai elementelor nule. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel numărul `k` al elementelor nule și pozițiile `p` ale acestora în vectorul inițial.

Funcția care implementează algoritmul pentru determinarea numărului și poziției elementelor nule dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.105, a). Numele funcției este `fkp0` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fkp0.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează inițializarea parametrului `k=0`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if` se verifică fiecare valoare a vectorului și se decide dacă este zero, caz în care se incrementează cu o unitate valoarea parametrului `k`, și în plus se adaugă indicele curent `i` parametrului `p` care va stoca informația referitoare la pozițiile elementelor nule. Valorile obținute pentru cei doi parametri `k` și `p` sunt parametrii de ieșire ai funcției. Rezultatul apelării funcției este prezentat în figura 3.105, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.105. Fișier `function` pentru determinarea numărului k și a poziției p a elementelor nule dintr-un vector.

Problema 3.43

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

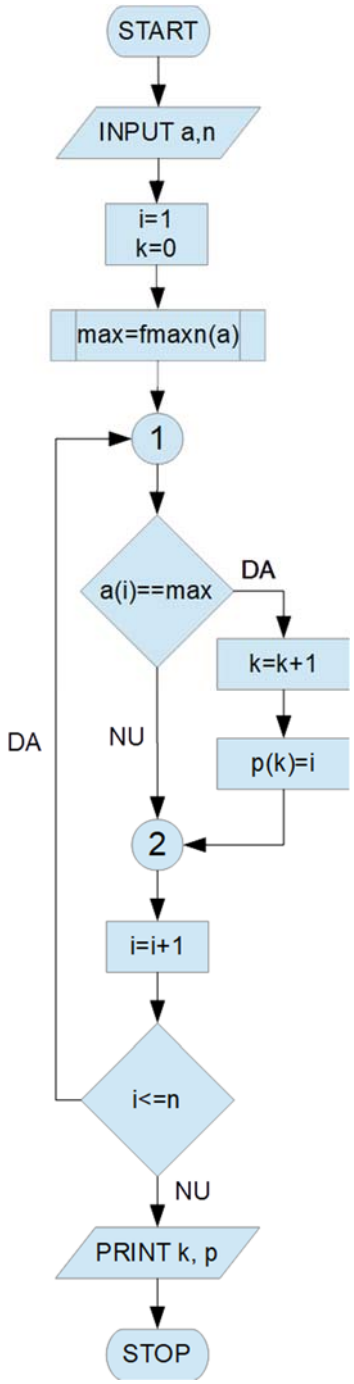
Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a elementului maxim al vectorului a. Să se determine în plus numărul k de elemente maxime, precum și pozițiile p ale acestora în vectorul inițial.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

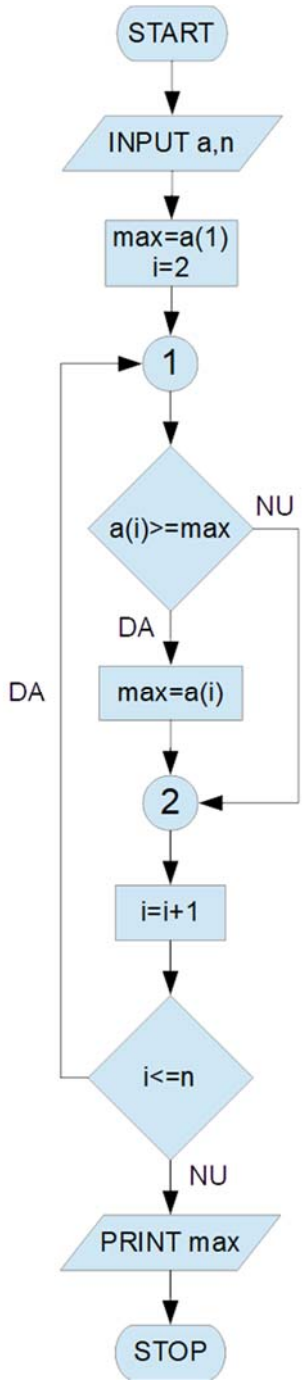
Să se verifice pentru `a=[-1 2 9 4 -5 -6 0 8 9 0]`.

Rezolvare

Schemele logice sunt prezentate în figura 3.106.



a) schema logică principală



b) schema logică secundară (procedura `fmaxn`)

Figura 3.106. Schema logică pentru determinarea elementului maxim, a numărului k de elemente maxime, precum și a pozițiilor p ale acestora.

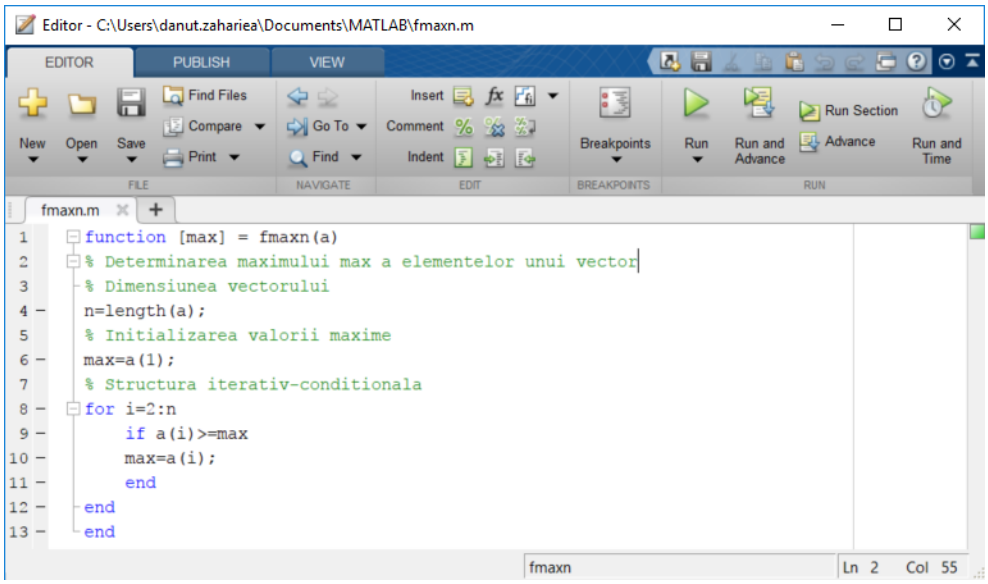
Observații

- Algoritmul pentru determinarea elementului maxim, a numărului `k` de elemente maxime, precum și a pozițiilor `p` ale acestora este descris cu ajutorul a două scheme logice:
  - schema logică principală (figura 3.106, a) care apelează procedura `fmaxn` de calculul a valorii maxime și care calculează apoi cei doi parametri `k` și `p`,
  - schema logică secundară care reprezintă algoritmul de determinare a valorii maxime `max` pentru elementele vectorului `a` (figura 3.106, b).
- Pentru schema logică principală:
  - Datele de intrare ale schemei logice principale sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
  - Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru numărul elementelor maxime `k=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.
  - Se apelează apoi procedura pentru calculul valorii maxime, denumită `fmaxn`.
  - Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
  - La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este egală cu valoarea maximă, `a(i)==max`, caz în care se incrementează cu o unitate parametrul `k`. Pentru stocarea informației referitoare la poziția elementelor maxime se definește o variabilă `p`, care va avea `k` valori, anume indicii `i` ai elementelor maxime. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
  - După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
  - Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
  - Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel numărul `k` al elementelor maxime și pozițiile `p` ale acestora în vectorul inițial.
- Pentru schema logică secundară:
  - Datele de intrare ale schemei logice secundare sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
  - Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru valoarea maximă care se inițializează chiar cu prima valoarea a vectorului `max=a(1)` și pentru contorul `i=2` al structurii iterative care va parcurge indicii `2÷n` ai valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.

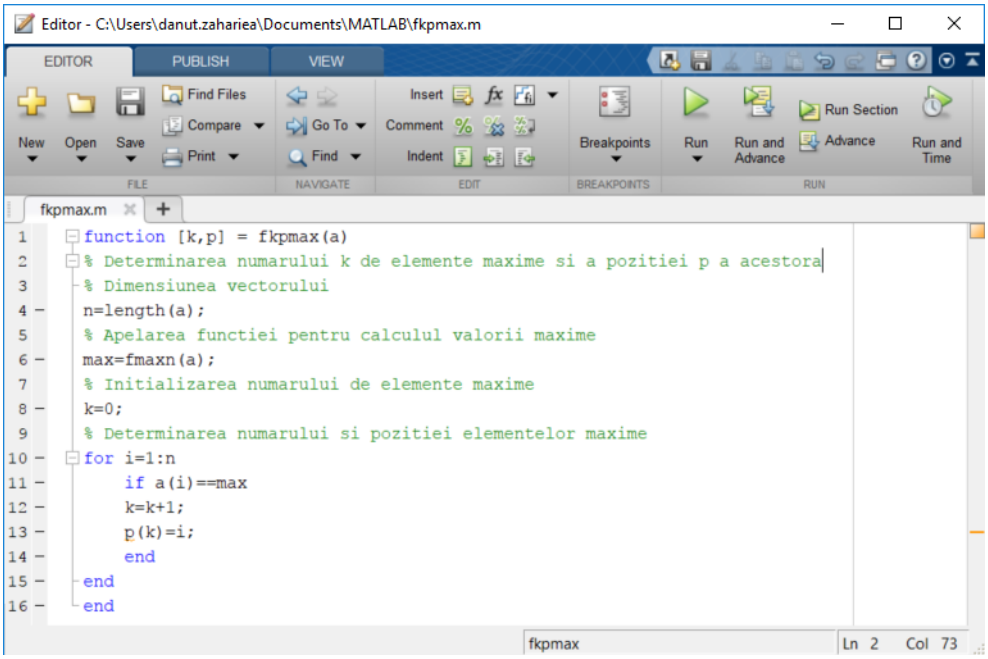
- Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este mai mare sau egală cu valoarea maximă, `a(i)>=max`, caz în care valoarea `a(i)` va fi atribuită variabilei `max`. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel valoarea maximă `max` a elementelor vectorului `a`.

Funcția care implementează algoritmul pentru determinarea valorii maxime dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.107, a). Numele funcției este `fmaxn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fmaxn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează inițializarea valorii maxime `max=a(1)`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if` se verifică celelalte `n-1` valori rămase ale vectorului și se decide dacă există valori mai mari decât valoarea maximă inițializată. Dacă se identifică o astfel de valoare, atunci aceasta va fi atribuită variabilei `max`, care reprezintă și parametrul de ieșire al funcției. Rezultatul apelării funcției este prezentat în figura 3.107, c).

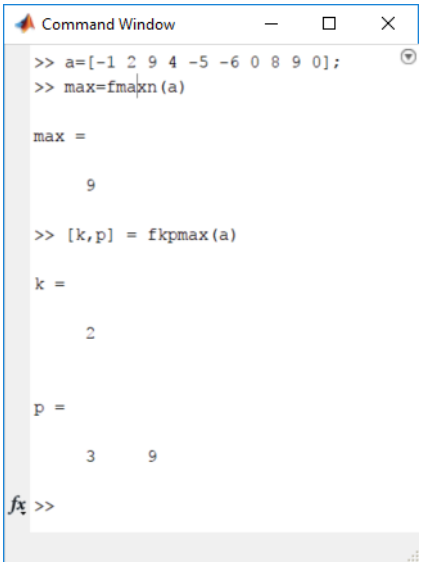
Funcția care implementează algoritmul pentru determinarea numărului `k` și a poziției `p` a elementelor maxime dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.107, b). Numele funcției este `fkpmax` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fkpmax.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează calculul valorii maxime prin apelarea funcției `fmaxn` și inițializarea apoi a parametrului `k=0`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if` se verifică fiecare valoare a vectorului și se decide dacă este egală cu valoarea maximă, caz în care se incrementează cu o unitate valoarea parametrului `k`, și în plus se adaugă indicele curent `i` parametrului `p` care va stoca informația referitoare la pozițiile elementelor maxime. Valorile obținute pentru cei doi parametri `k` și `p` sunt parametrii de ieșire ai funcției. Rezultatul apelării funcției este prezentat în figura 3.107, c).



a) fișierul `function` pentru calculul valorii maxime



b) fișierul `function` pentru calculul numărului și a poziției elementelor maxime



c) rezultatul obținut

Figura 3.107. Fișiere `function` pentru determinarea elementului maxim, a numărului k de elemente maxime, precum și a pozițiilor p ale acestora.

Problema 3.44

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

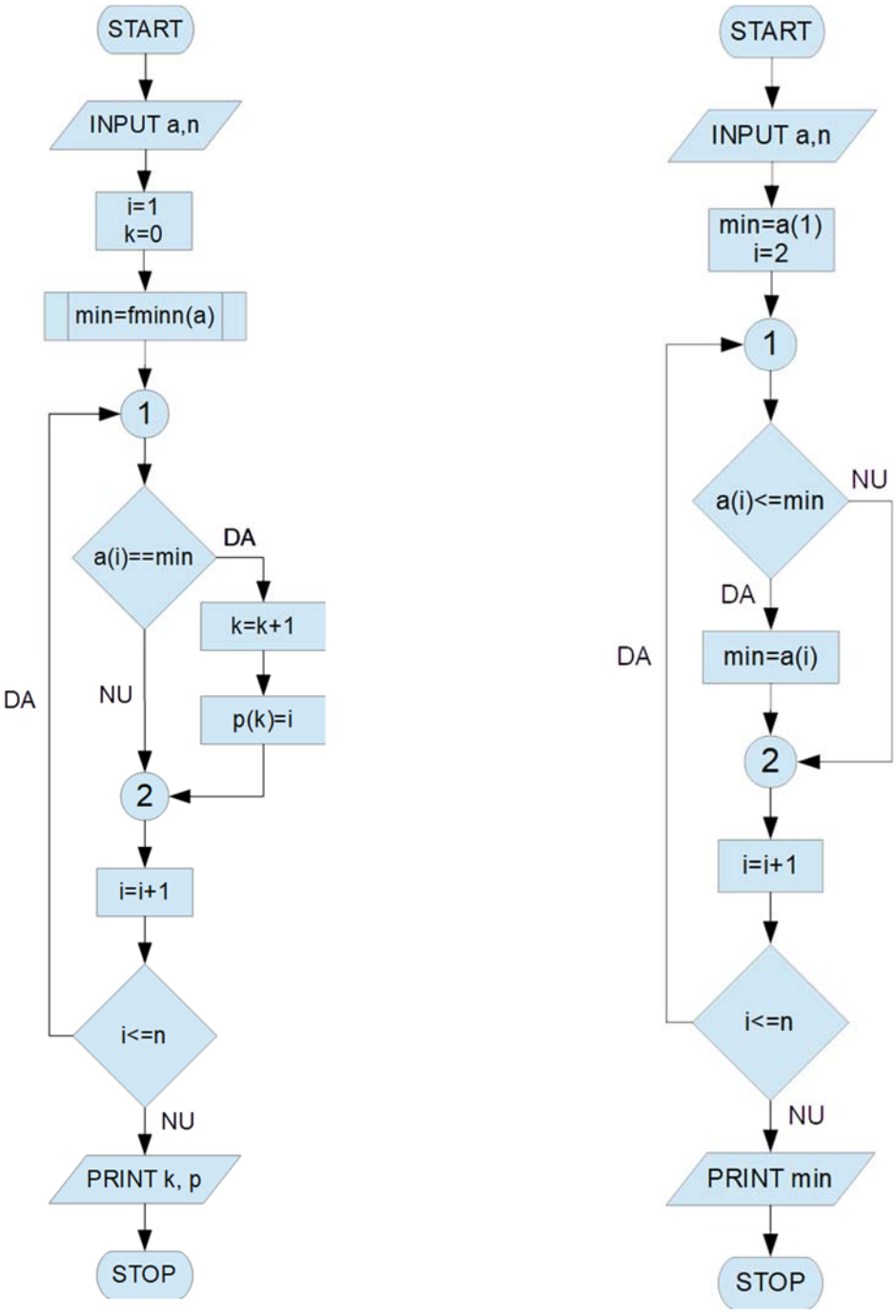
Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a elementului minim al vectorului a. Să se determine în plus numărul k de elemente minime, precum și pozițiile p ale acestora în vectorul inițial.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[5 2 9 4 5 6 2 8 9 2]`.

Rezolvare

Schemele logice sunt prezentate în figura 3.108.



a) schema logică principală b) schema logică secundară (procedura `fminn`)

Figura 3.108. Schema logică pentru determinarea elementului minim, a numărului k de elemente minime, precum și a pozițiilor p ale acestora.

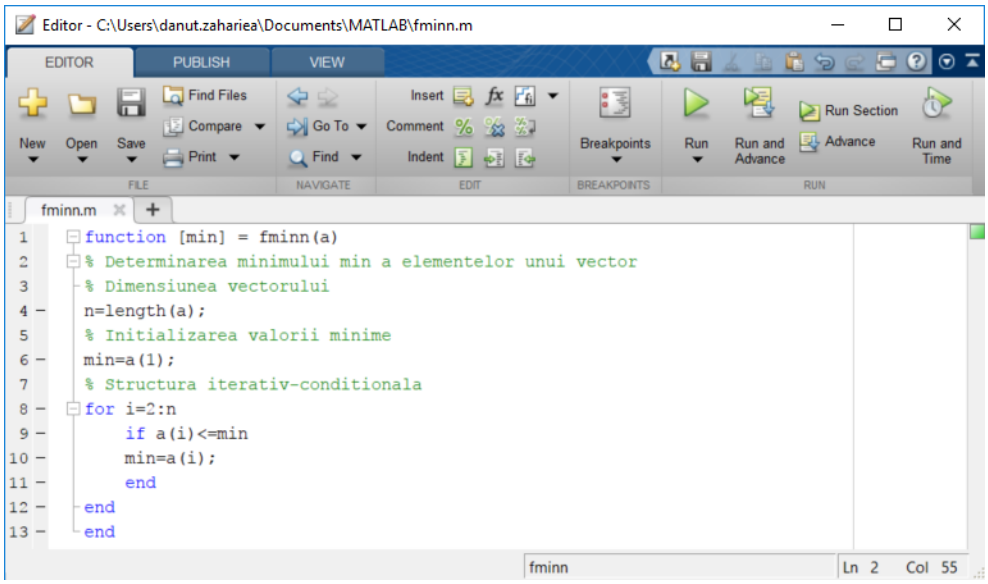
Observații

- Algoritmul pentru determinarea elementului minim, a numărului `k` de elemente minime, precum și a pozițiilor `p` ale acestora este descris cu ajutorul a două scheme logice:
  - schema logică principală (figura 3.108, a) care apelează procedura `fminn` de calculul a valorii minime și care calculează apoi cei doi parametri `k` și `p`,
  - schema logică secundară care reprezintă algoritmul de determinare a valorii minime `min` pentru elementele vectorului `a` (figura 3.108, b).
- Pentru schema logică principală:
  - Datele de intrare ale schemei logice principale sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
  - Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru numărul elementelor maxime `k=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.
  - Se apelează apoi procedura pentru calculul valorii minime, denumită `fminn`.
  - Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
  - La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este egală cu valoarea minimă, `a(i)==min`, caz în care se incrementează cu o unitate parametrul `k`. Pentru stocarea informației referitoare la poziția elementelor minime se definește o variabilă `p`, care va avea `k` valori, anume indicii `i` ai elementelor minime. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
  - După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
  - Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
  - Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel numărul `k` al elementelor minime și pozițiile `p` ale acestora în vectorul inițial.
- Pentru schema logică secundară:
  - Datele de intrare ale schemei logice secundare sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
  - Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru valoarea minimă care se inițializează chiar cu prima valoarea a vectorului `min=a(1)` și pentru contorul `i=2` al structurii iterative care va parcurge indicii `2÷n` ai valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.

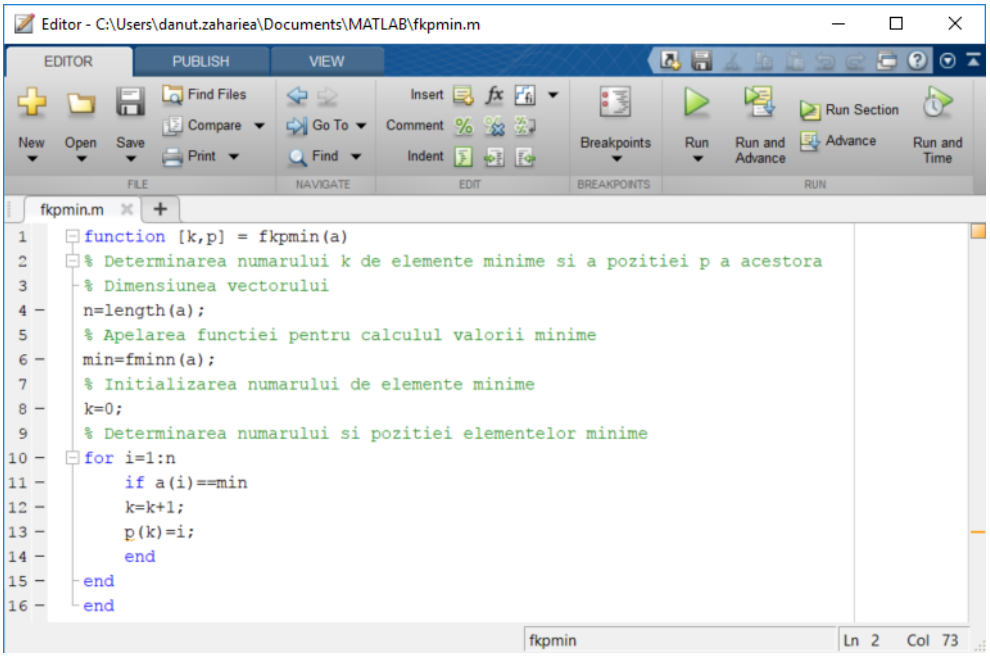
- Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este mai mică sau egală cu valoarea minimă, `a(i)<=min`, caz în care valoarea `a(i)` va fi atribuită variabilei `min`. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel valoarea minimă `min` a elementelor vectorului `a`.

Funcția care implementează algoritmul pentru determinarea valorii minime dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.109, a). Numele funcției este `fminn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fminn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează inițializarea valorii minime `min=a(1)`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if` se verifică celelalte `n-1` valori rămase ale vectorului și se decide dacă există valori mai mici decât valoarea minimă inițializată. Dacă se identifică o astfel de valoare, atunci aceasta va fi atribuită variabilei `min`, care reprezintă și parametrul de ieșire al funcției. Rezultatul apelării funcției este prezentat în figura 3.109, c).

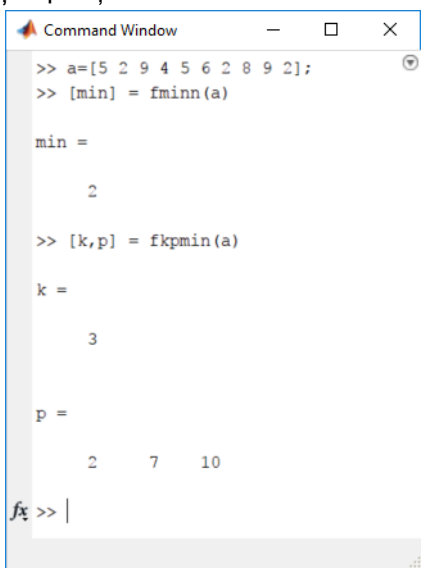
Funcția care implementează algoritmul pentru determinarea numărului `k` și a poziției `p` a elementelor minime dintr-un vector este definită în fișierul de tip `function` prezentat în figura 3.109, b). Numele funcției este `fkpmin` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fkpmin.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează calculul valorii minime prin apelarea funcției `fminn` și inițializarea apoi a parametrului `k=0`. Prin intermediul unei structuri iterative de tip `for` și a unei structuri alternative de tip `if` se verifică fiecare valoare a vectorului și se decide dacă este egală cu valoarea minimă, caz în care se incrementează cu o unitate valoarea parametrului `k`, și în plus se adaugă indicele curent `i` parametrului `p` care va stoca informația referitoare la pozițiile elementelor minime. Valorile obținute pentru cei doi parametri `k` și `p` sunt parametrii de ieșire ai funcției. Rezultatul apelării funcției este prezentat în figura 3.109, c).



a) fișierul `function` pentru calculul valorii minime



b) fișierul `function` pentru calculul numărului și a poziției elementelor minime



c) rezultatul obținut

Figura 3.109. Fișiere `function` pentru determinarea elementului minim, a numărului k de elemente minime, precum și a pozițiilor p ale acestora.

Problema 3.45

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei elementelor pozitive din vectorul a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[-1 2 0 4 -5 -6 0 8 9 10]`.

Rezolvare

Schema logică este prezentată în figura 3.110.

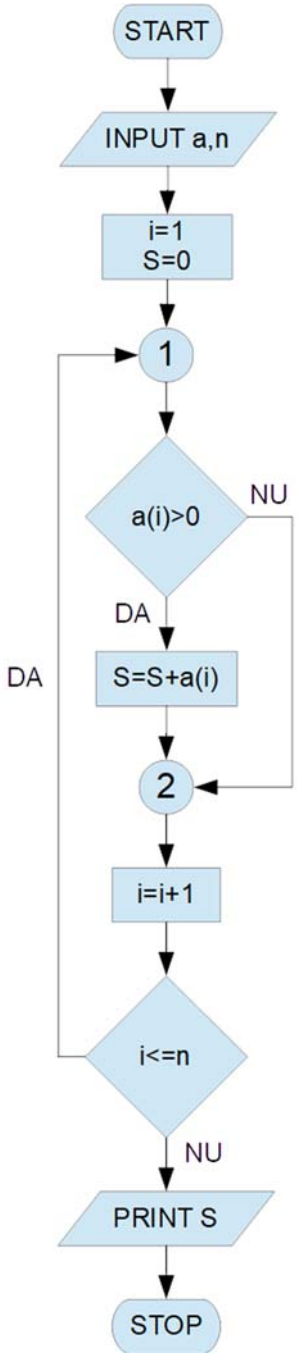
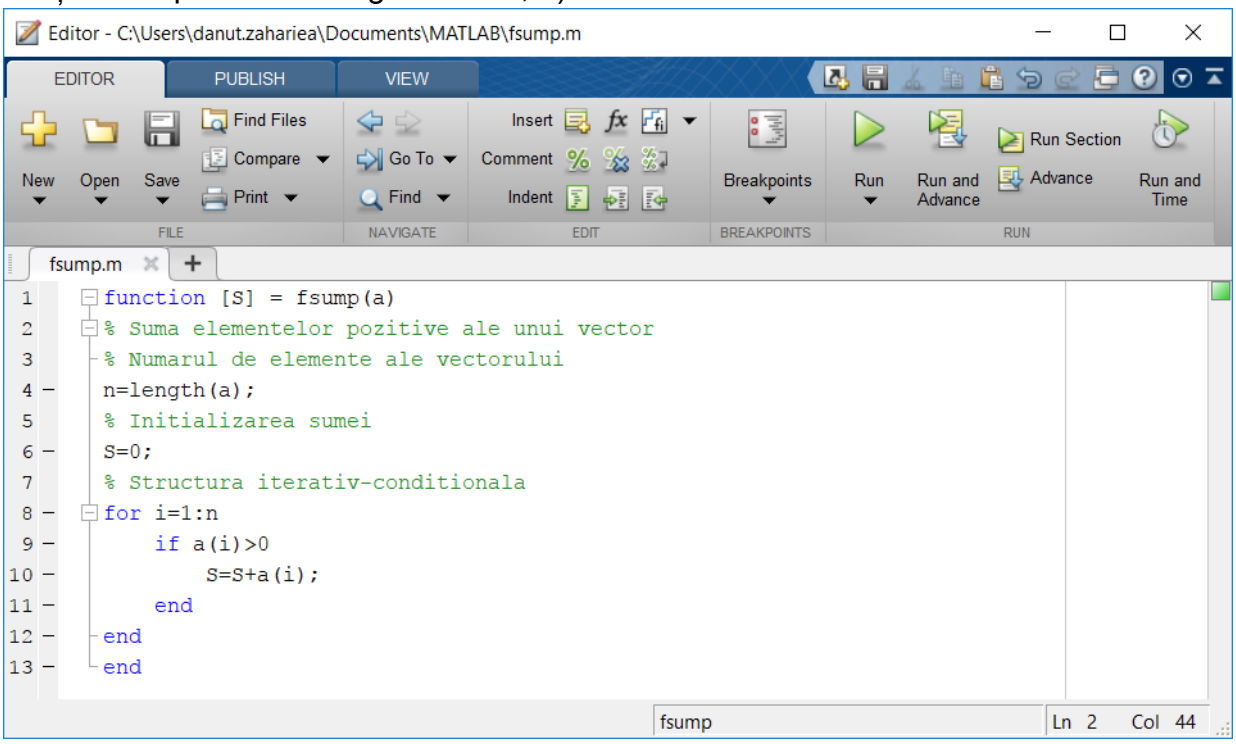


Figura 3.110. Schema logică pentru determinarea sumei elementelor pozitive dintr-un vector.

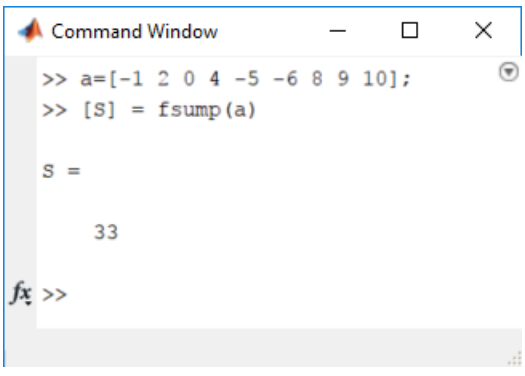
Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru suma elementelor pozitive care se inițializează cu 0 (element neutru pentru adunare) `S=0` și pentru contorul `i=1` al structurii iterative care va parcurge indicii tuturor valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.
- Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este pozitivă, `a(i)>0`, caz în care se recalculează suma elementelor pozitive prin adăugarea valorii curente la valoarea anterioară a sumei `S=S+a(i)`. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel valoarea sumei elementelor pozitive ale vectorului.

Funcția care implementează algoritmul pentru determinarea sumei `S` a elementelor pozitive ale unui vector este definită în fișierul de tip `function` prezentat în figura 3.111, a). Numele funcției este `fsump` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsump.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativ-condițională (o structură iterativă de tip `for` și o structură alternativă de tip `if`) prin care se verifică fiecare valoare a vectorului și se decide dacă este pozitivă (`a(i)>0`), caz în care se recalculează suma `S` prin adăugarea valorii curente `a(i)` la valoarea anterioară a sumei, `S=S+a(i)`. Valoarea finală a sumei `S` reprezintă parametrul de ieșire al funcției. Rezultatul apelării funcției este prezentat în figura 3.111, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.111. Fișier `function` pentru determinarea sumei elementelor pozitive dintr-un vector.

Problema 3.46

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a produsului elementelor nenule din vectorul a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[-1 2 0 4 -5 -6 0 8 9 0]`.

Rezolvare

Schema logică este prezentată în figura 3.112.

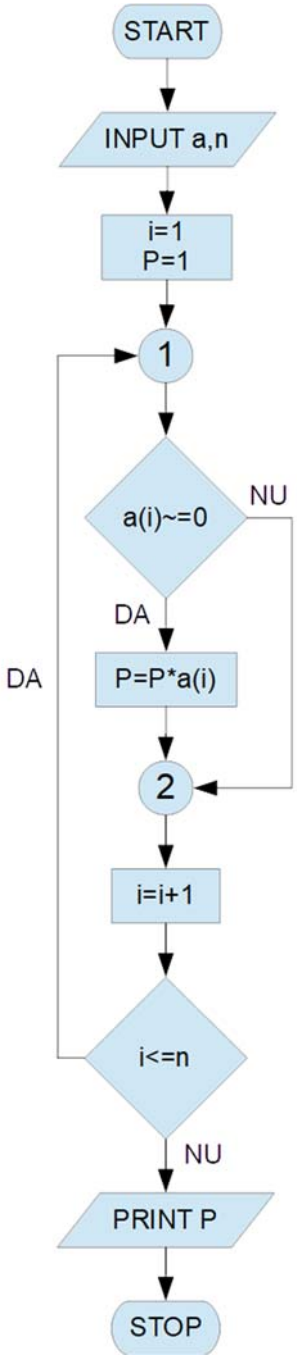
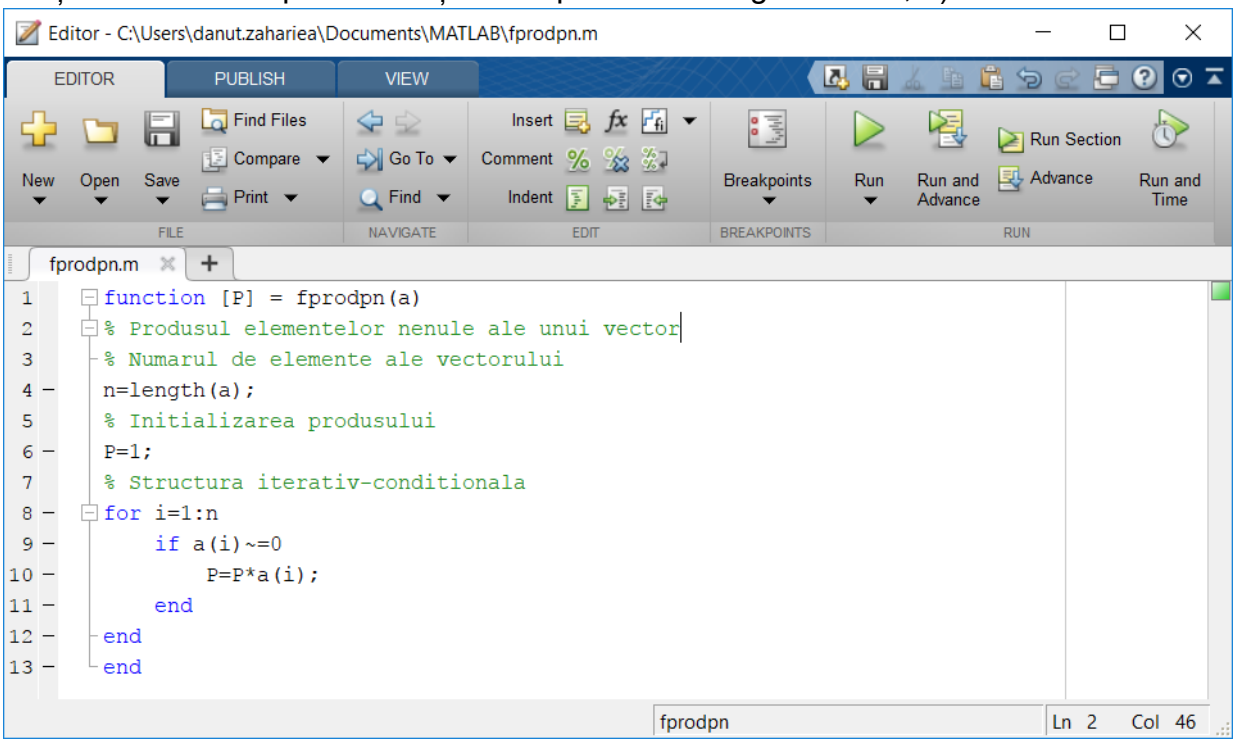


Figura 3.112. Schema logică pentru determinarea produsului elementelor nenule dintr-un vector.

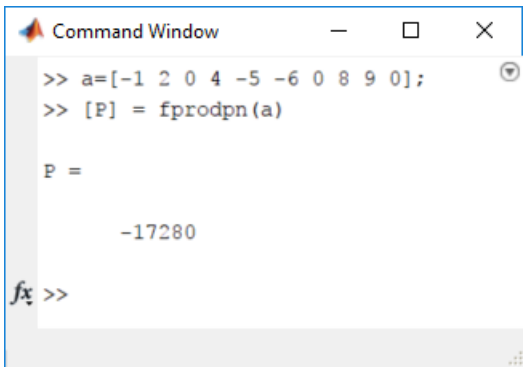
Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru produsul elementelor nenule care se inițializează cu 1 (element neutru pentru înmulțire) `P=1` și pentru contorul `i=1` al structurii iterative care va parcurge indicii tuturor valorilor din vectorul `a` declarat în faza de introducere a datelor de intrare.
- Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare, în interiorul căreia se găsește o structură alternativă cu o ramură.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului este diferită de zero, `a(i)~=0`, caz în care se recalculează produsul elementelor nenule prin înmulțirea valorii curente cu valoarea anterioară a produsului `P=P*a(i)`. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel valoarea produsului elementelor nenule ale vectorului.

Funcția care implementează algoritmul pentru determinarea produsului `P` al elementelor nenule ale unui vector este definită în fișierul de tip `function` prezentat în figura 3.113, a). Numele funcției este `fprodpn` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fprodpn.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativ-condițională (o structură iterativă de tip `for` și o structură alternativă de tip `if`) prin care se verifică fiecare valoare a vectorului și se decide dacă este diferită de zero (`a(i)~=0`), caz în care se recalculează produsul `P` prin înmulțirea valorii curente `a(i)` cu valoarea anterioară a produsului, `P=P*a(i)`. Valoarea finală a produsului `P` reprezintă parametrul de ieșire al funcției. Rezultatul apelării funcției este prezentat în figura 3.113, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.113. Fișier `function` pentru determinarea produsului elementelor nenule dintr-un vector.

Problema 3.47

Se consideră un vector cu n elemente $a = [a_1\ a_2 \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a sumei elementelor vectorului a care aparțin domeniului $[b_1\ b_2]$.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]` și `b=[4 8]`.

Rezolvare

Schema logică este prezentată în figura 3.114.

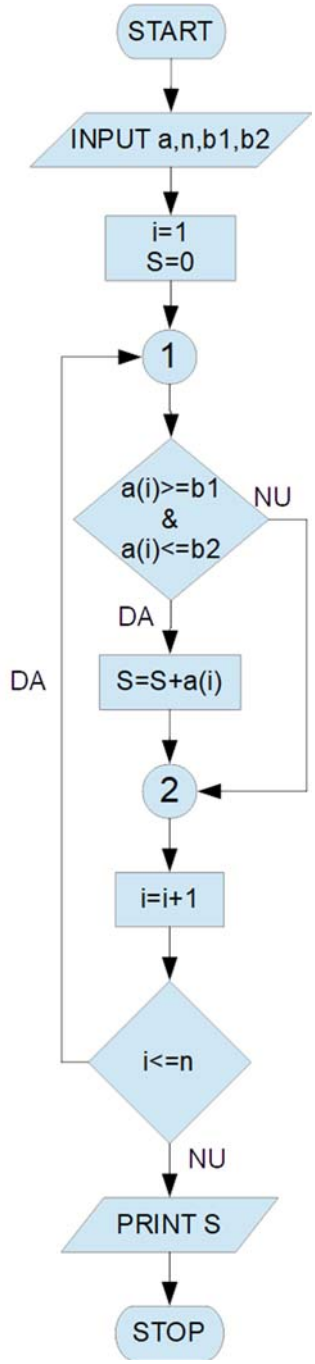
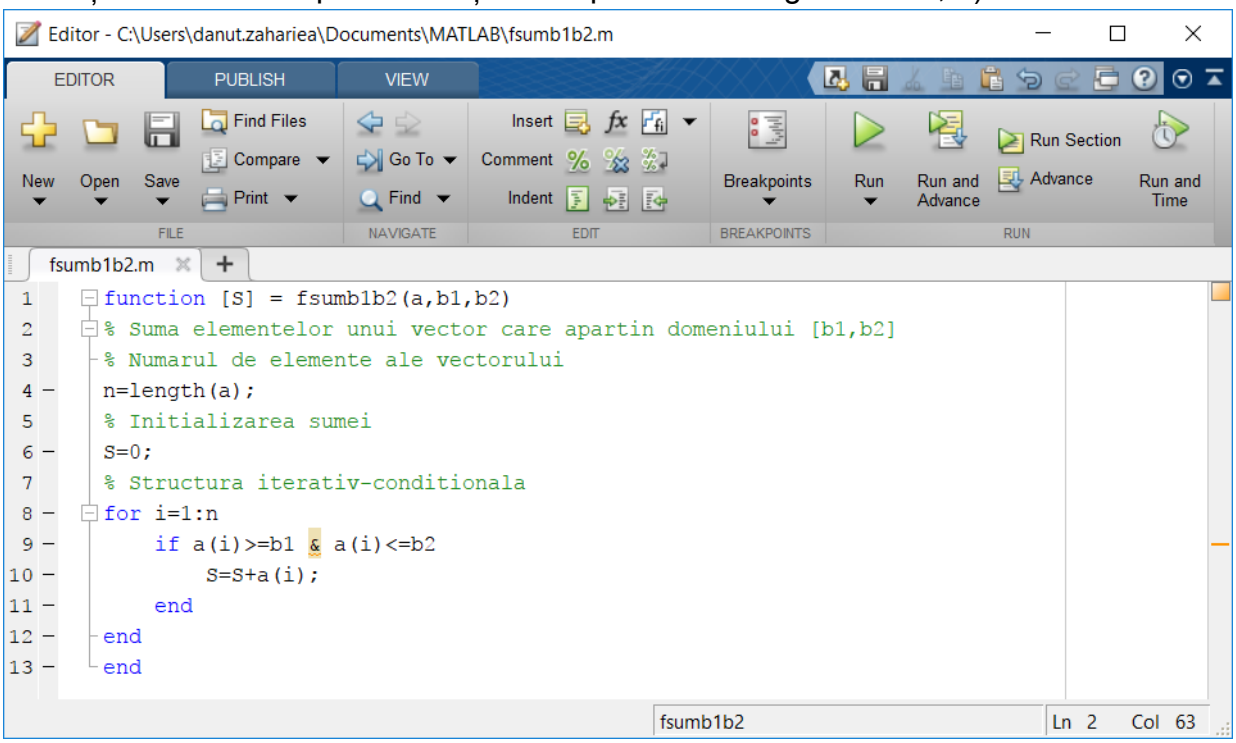


Figura 3.114. Schema logică pentru determinarea sumei elementelor vectorului a care aparțin domeniului $[b_1\ b_2]$.

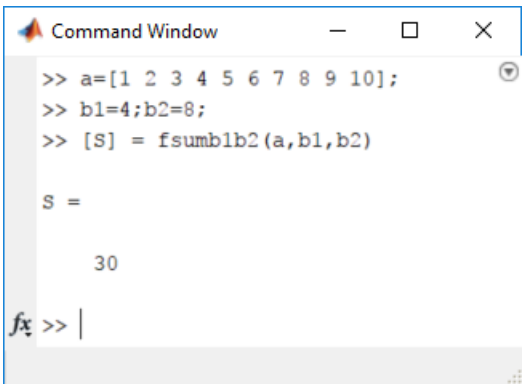
Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia, precum și cele două limite ale domeniului `b1` și `b2`.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire: pentru suma elementelor din domeniul specificat care se inițializează cu 0 (element neutru pentru adunare) `S=0` și pentru contorul `i=1` al structurii iterative.
- Urmează apoi o structură iterativă cu număr cunoscut de iterații, în interiorul căreia se găsește o structură alternativă cu o ramură.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se verifică dacă valoarea corespunzătoare a vectorului aparține domeniului specificat, `a(i)>=b1 & a(i)<=b2`, caz în care se recalculează suma `S` prin adăugarea valorii curente `a(i)` la valoarea anterioară a sumei `S=S+a(i)`. La ieșirea din structura alternativă, fie pe ramura `DA`, fie pe ramura `NU`, urmează incrementarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n`.
- Dacă expresia logică este adevărată se continuă verificarea următorului element al vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel valoarea sumei elementelor din interiorul domeniului specificat.

Funcția care implementează algoritmul pentru determinarea sumei `S` a elementelor vectorului `a` care aparțin domeniului `[b1,b2]` este definită în fișierul de tip `function` prezentat în figura 3.115, a). Numele funcției este `fsumb1b2` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsumb1b2.m`. Funcția acceptă următorii parametri de intrare: variabila vectorială `a` și cele două limite ale domeniului, `b1` și `b2`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativ-condițională (o structură iterativă de tip `for` și o structură alternativă de tip `if`) prin care se verifică fiecare valoare a vectorului și se decide dacă aparține domeniului specificat (`a(i)>=b1 & a(i)<=b2`), caz în care se recalculează suma `S` prin adunarea valorii curente `a(i)` la valoarea anterioară a sumei, `S=S+a(i)`. Valoarea finală a sumei `S` reprezintă parametrul de ieșire al funcției. Rezultatul apelării funcției este prezentat în figura 3.115, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.115. Fișier `function` pentru determinarea sumei elementelor vectorului a care aparțin domeniului $[b_1\ b_2]$.

Problema 3.48

Se consideră un vector cu n elemente $a = [a_1\ a_2 \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de ordonare în sens crescător a elementelor vectorului a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[5 4 1 6 3 9 7 2 10 8]`.

Rezolvare

Schema logică este prezentată în figura 3.116.

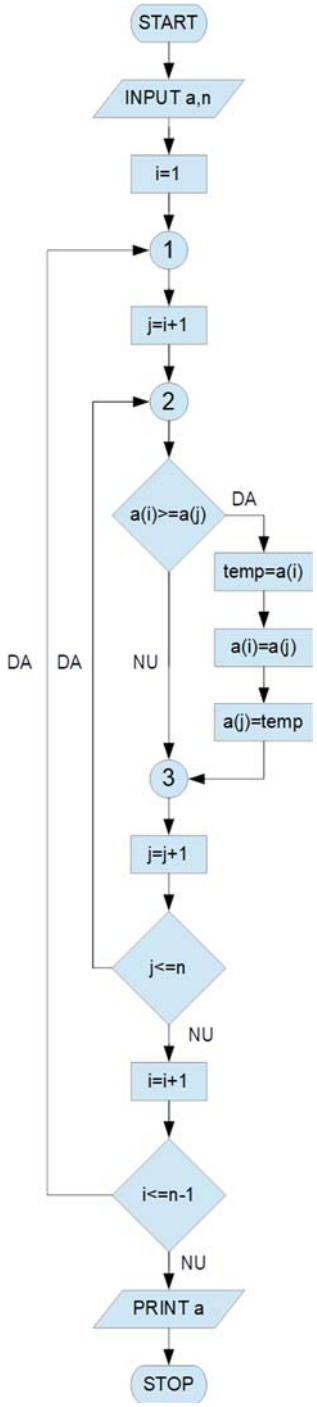
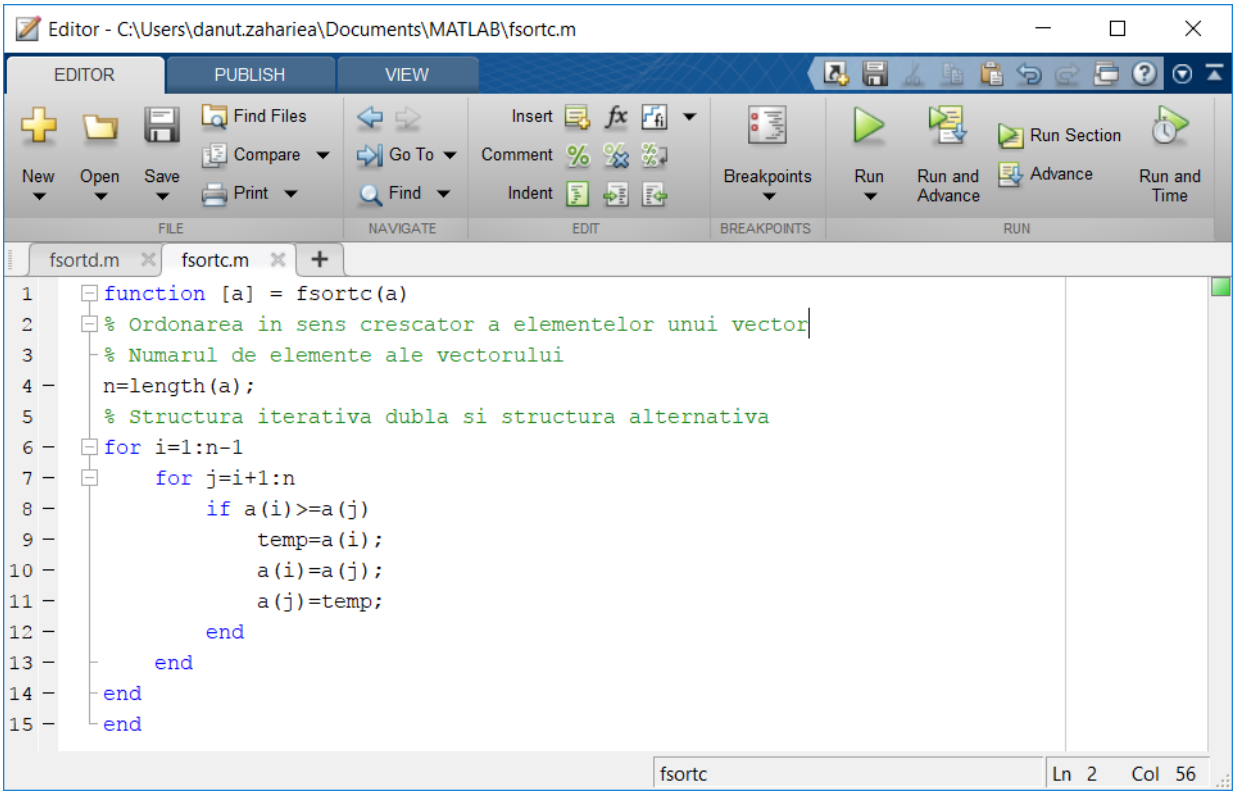


Figura 3.116. Schema logică pentru ordonarea în sens crescător a elementelor unui vector.

Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Schema logică se bazează pe metoda interschimbării și conține două structuri iterative cu număr cunoscut de iterații, în interiorul cărora se află o structură alternativă.
- Prima structură iterativă este definită prin contorul `i` care parcurge indicii de la `1` la `n-1`.
- A doua structură iterativă este definită prin contorul `j` care parcurge indicii de la `i+1` la `n`.
- Pentru o anumită valoare a contorului `i`, valoarea corespunzătoare a vectorului `a(i)` se compară cu toate celelalte valori rămase ale vectorului `a(j)`, cu `j` de la `i+1` la `n`. Dacă se găsește vreo valoare `a(j)` mai mică decât valoare `a(i)` se schimă cele doua valori între ele.
- Interschimbarea valorilor `a(i)` și `a(j)` se realizează prin intermediul unei variabile temporare, denumită `temp` care va stoca, temporar, valoarea `a(i)`. După interschimbare, `a(i)=a(j)`, variabilei `a(j)` i se atribuie valoarea stocată în variabila temporară, `a(j)=temp`.
- După parcurgerea indicilor celor două structuri iterative, se afișează variabila de ieșire a algoritmului, respectiv vectorul `a` cu elementele sortate în sens crescător.

Funcția care implementează algoritmul pentru ordonarea în sens crescător a elementelor vectorului `a` este definită în fișierul de tip `function` prezentat în figura 3.117, a). Numele funcției este `fsortc` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsortc.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativă dublă în interiorul căreia se află o structură condițională (o structură iterativă de tip `for` dublă și o structură alternativă de tip `if`) prin care se verifică dacă `a(i)>=a(j)`, cu `i=1:n-1` și `j=i+1:n`. În cazul în care condiția logică este adevărată, se procedează la interschimbarea celor două valori, astfel încât la finalizarea parcurgerii tuturor valorilor contorului `j`, valoarea `a(i)` va fi valoarea cea mai mică față de toate celelalte valori rămase ale vectorului. După parcurgerea indicilor celor două structuri iterative, se obține variabila de ieșire a funcției, respectiv vectorul `a` cu elementele sortate în sens crescător. Rezultatul apelării funcției este prezentat în figura 3.117, b).



a) fișierul `function`

b) rezultatul obținut

Figura 3.117. Fișier `function` pentru ordonarea în sens crescător a elementelor unui vector.

Problema 3.49

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de ordonare în sens descrescător a elementelor vectorului a.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[5 4 1 6 3 9 7 2 10 8]`.

Rezolvare

Schema logică este prezentată în figura 3.118.

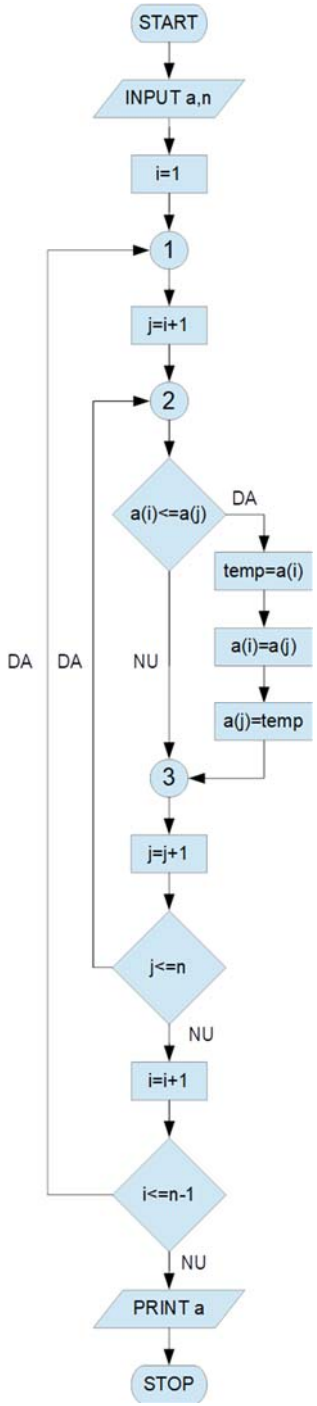
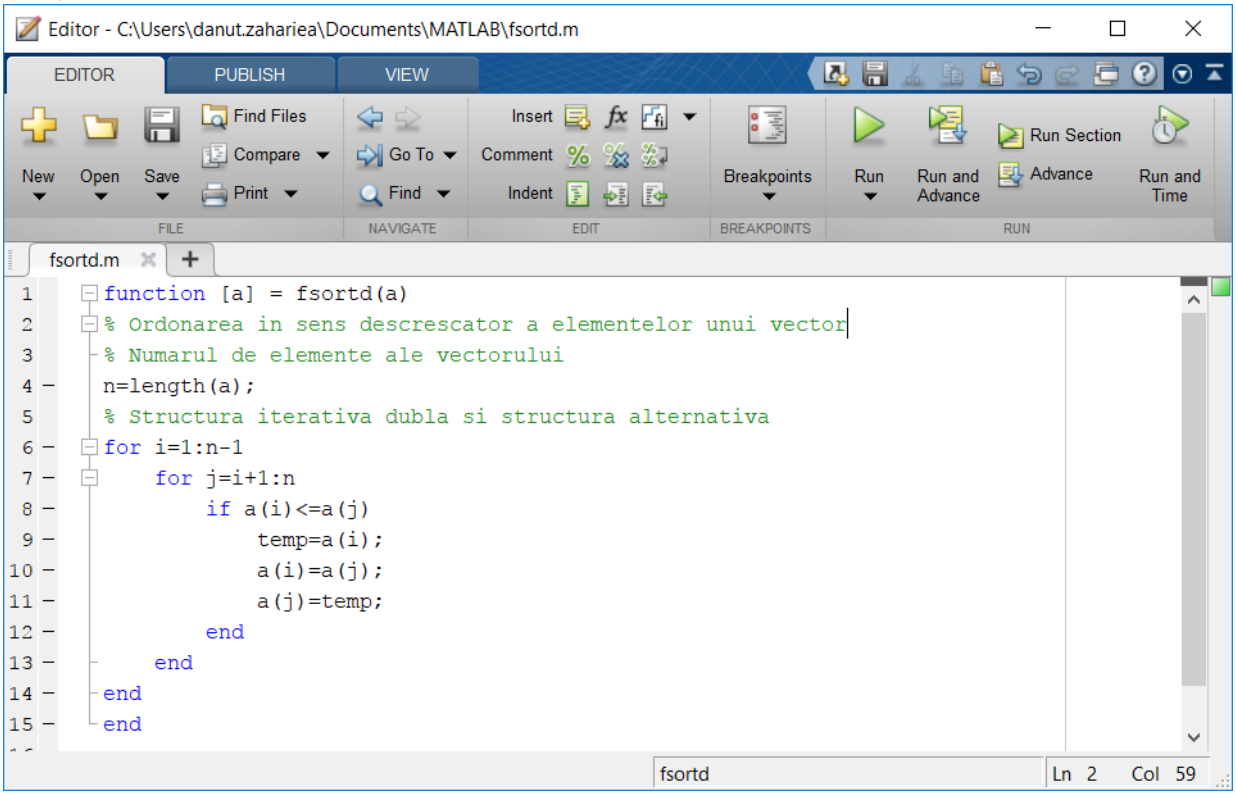


Figura 3.118. Schema logică pentru ordonarea în sens descrescător a elementelor unui vector.

Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Schema logică se bazează pe metoda interschimbării și conține două structuri iterative cu număr cunoscut de iterații, în interiorul cărora se află o structură alternativă.
- Prima structură iterativă este definită prin contorul `i` care parcurge indicii de la `1` la `n-1`.
- A doua structură iterativă este definită prin contorul `j` care parcurge indicii de la `i+1` la `n`.
- Pentru o anumită valoare a contorului `i`, valoarea corespunzătoare a vectorului `a(i)` se compară cu toate celelalte valori rămase ale vectorului `a(j)`, cu `j` de la `i+1` la `n`. Dacă se găsește vreo valoare `a(j)` mai mare decât valoare `a(i)` se schimă cele doua valori între ele.
- Interschimbarea valorilor `a(i)` și `a(j)` se realizează prin intermediul unei variabile temporare, denumită `temp` care va stoca, temporar, valoarea `a(i)`. După interschimbare, `a(i)=a(j)`, variabilei `a(j)` i se atribuie valoarea stocată în variabila temporară, `a(j)=temp`.
- După parcurgerea indicilor celor două structuri iterative, se afișează variabila de ieșire a algoritmului, respectiv vectorul `a` cu elementele sortate în sens descrescător.

Funcția care implementează algoritmul pentru ordonarea în sens descrescător a elementelor vectorului `a` este definită în fișierul de tip `function` prezentat în figura 3.119, a). Numele funcției este `fsortd` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsortd.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativă dublă în interiorul căreia se află o structură condițională (o structură iterativă de tip `for` dublă și o structură alternativă de tip `if`) prin care se verifică dacă `a(i)<=a(j)`, cu `i=1:n-1` și `j=i+1:n`. În cazul în care condiția logică este adevărată, se procedează la interschimbarea celor două valori, astfel încât la finalizarea parcurgerii tuturor valorilor contorului `j`, valoarea `a(i)` va fi valoarea cea mai mare față de toate celelalte valori rămase ale vectorului. După parcurgerea indicilor celor două structuri iterative, se obține variabila de ieșire a funcției, respectiv vectorul `a` cu elementele sortate în sens descrescător. Rezultatul apelării funcției este prezentat în figura 3.119, b).



a) fișierul `function`

b) rezultatul obținut

Figura 3.119. Fișier `function` pentru ordonarea în sens descrescător a elementelor unui vector.

Problema 3.50

Se consideră un vector cu n elemente $a = [a_1\ a_2\ \dots a_{n-1}\ a_n]$.

Să se realizeze o schemă logică pentru descrierea algoritmului de interschimbare a ordinii elementelor vectorului a astfel încât să se obțină vectorul $a = [a_n\ a_{n-1}\ \dots a_2\ a_1]$.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru `a=[1 2 3 4 5 6 7 8 9 10]`.

Rezolvare

Schema logică este prezentată în figura 3.120.

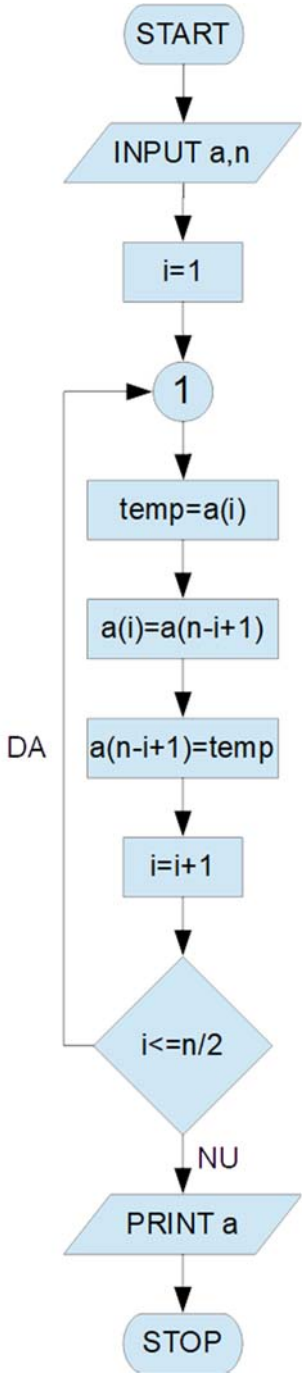
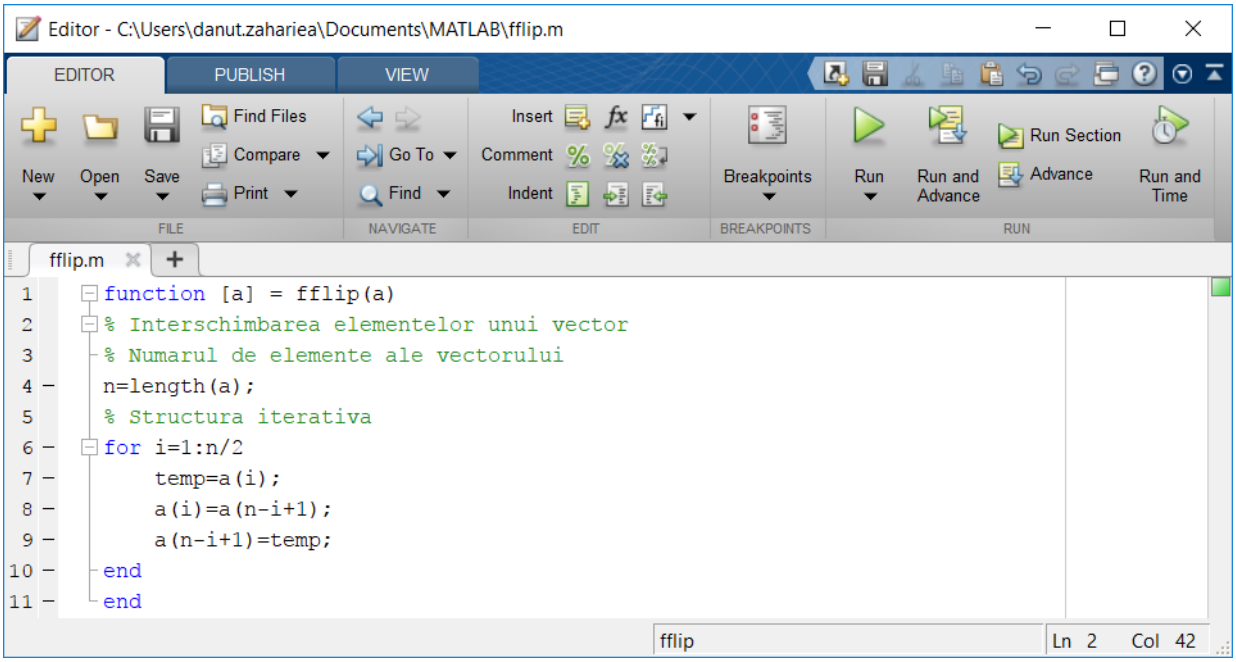


Figura 3.120. Schema logică pentru schimbarea ordinii elementelor unui vector.

Observații

- Datele de intrare ale schemei logice sunt elementele vectorului `a` și dimensiunea `n` a acestuia.
- Faza de inițializare a algoritmului cuprinde o comandă de atribuire `i=1`, pentru contorul `i` al structurii iterative.
- Urmează apoi o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare prin care se va aplica algoritmul de interschimbare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, valoarea elementului `a(i)` se stochează într-o variabilă temporară, `temp`. Interschimbarea se realizează între elementele `a(i)` și `a(n-i+1)` și presupune execuția a două instrucțiuni: `a(i)=a(n-i+1)` prin care se obține noua valoare a elementului `a(i)`, și `a(n-i+1)=temp` prin care valoarea stocată temporar în variabila `temp` (adică `a(i)`) se atribuie valorii `a(n-i+1)`.
- După realizarea interschimbării, urmează faza de incrementare a contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=n/2`.
- Dacă expresia logică este adevărată se continuă interschimbarea următoarelor două elemente ale vectorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații, obținându-se astfel vectorul a cu elemente interschimbate.

Funcția care implementează algoritmul de interschimbare a elementelor unui vector este definită în fișierul de tip `function` prezentat în figura 3.121, a). Numele funcției este `fflip` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fflip.m`. Funcția acceptă un singur parametru de intrare, respectiv variabila vectorială `a`. După determinarea numărului de elemente ale vectorului `a` cu instrucțiunea `n=length(a)`, urmează o structură iterativă de tip `for` prin care se aplică algoritmul de interschimbare între valorile `a(i)` și `a(n-i+1)` prin intermediul variabilei temporare `temp`. După parcurgerea valorilor contorului `i=1:n/2`, se obține variabila de ieșire a funcției, respectiv vectorul `a` cu elementele interschimbate. Rezultatul apelării funcției este prezentat în figura 3.121, b)



a) fișierul `function`

b) rezultatul obținut

Figura 3.121. Fișier `function` pentru schimbarea ordinii elementelor unui vector.

Problema 3.51

Se consideră seria numerică $\sum_{k=1}^{\infty} a_k$ având termenul general:

$$a_k = \frac{1}{k^2}$$

și valoarea exactă a sumei seriei:

$$S_\infty = \sum_{k=1}^{\infty} a_k = \lim_{n\to\infty} S_n = \frac{\pi^2}{6}$$

a) Pentru un număr impus de termeni n=10, să se determine viteza de convergență a șirului sumelor parțiale calculând și analizând toate rezultatele parțiale:
   - șirul sumelor parțiale S_k, $\forall k = 1 \div n$:
   $$S_1 = \sum_{k=1}^{1} a_k,\ S_2 = \sum_{k=1}^{2} a_k,\ \ldots,\ S_n = \sum_{k=1}^{n} a_k$$
   - restul de ordin k definit prin relația:
   $$r_k = |S_k - S_\infty|,\ \forall k = 1 \div n$$
   - eroarea relativă de ordin k definită prin relația:
   $$e_k = \frac{|S_k - S_\infty|}{S_\infty} \cdot 100\ [\%]\ ,\forall k = 1 \div n$$

b) Să se determine numărul minim de termeni k_{min} ai şirului sumelor parţiale astfel încât eroarea relativă dintre valoarea exactă și valoarea aproximativă să fie mai mică decât valoarea critică e_{cr}=3%. Să se vizualizeze toate rezultatele parțiale.

c) Să se realizeze schemele logice și să se scrie fișierele `script` pentru implementarea acestora.

Rezolvare

Schemele logice sunt prezentate în figura 3.122, a) pentru cazul numărului de termeni impus, respectiv în figura 3.122, b) pentru cazul erorii relative impuse.

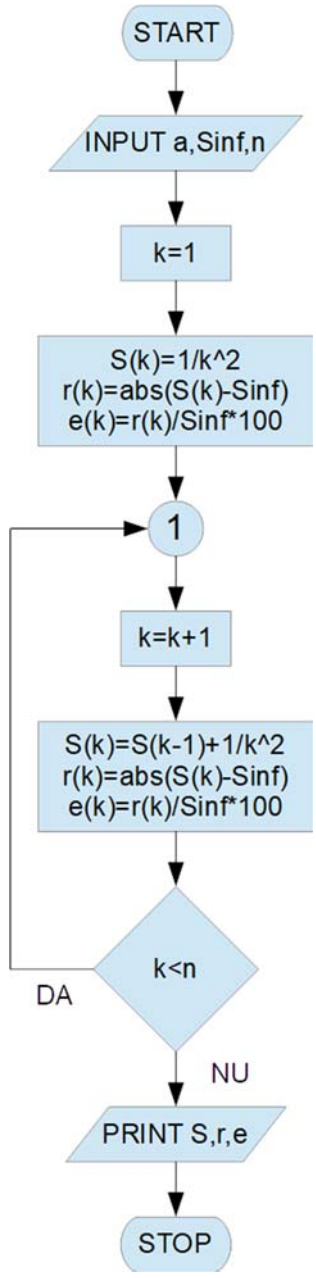
Observații

- Datele de intrare se referă la expresia termenului general al seriei `a(k)`, la valoarea exactă a sumei seriei `Sinf`, precum și la numărul `n` impus de termeni ai șirului sumelor parțiale (figura 3.122, a), respectiv la eroarea relativă critică `ecr` impusă (figura 3.122, b).
- Pentru ambele scheme logice faza de inițializare constă în inițializarea contorului `k=1`, respectiv în inițializarea rezultatelor parțiale (suma parțială de ordin `k=1`, `S(1)`; restul de ordin `k=1`, `r(1)`; eroarea relativă de ordin `k=1`, `e(1)`).
- Urmează apoi blocul de calcul iterativ care este structurat în mod diferit pentru cele două scheme logice:
  - Pentru schema logică din figura 3.122, a), intrarea în structura iterativă se face prin incrementarea contorului, `k=k+1`, după care, la fiecare iterație, definită prin valoarea curentă `k` a contorului, se calculează noua valoare `S(k)=S(k-1)+a(k)` din șirul sumelor parțiale, restul de ordin `k`, `r(k)=abs(S(k)-Sinf)`, precum și eroarea relativă fată de valoarea exactă a sumei seriei `e(k)=r(k)/Sinf*100`. Ieșirea din structura

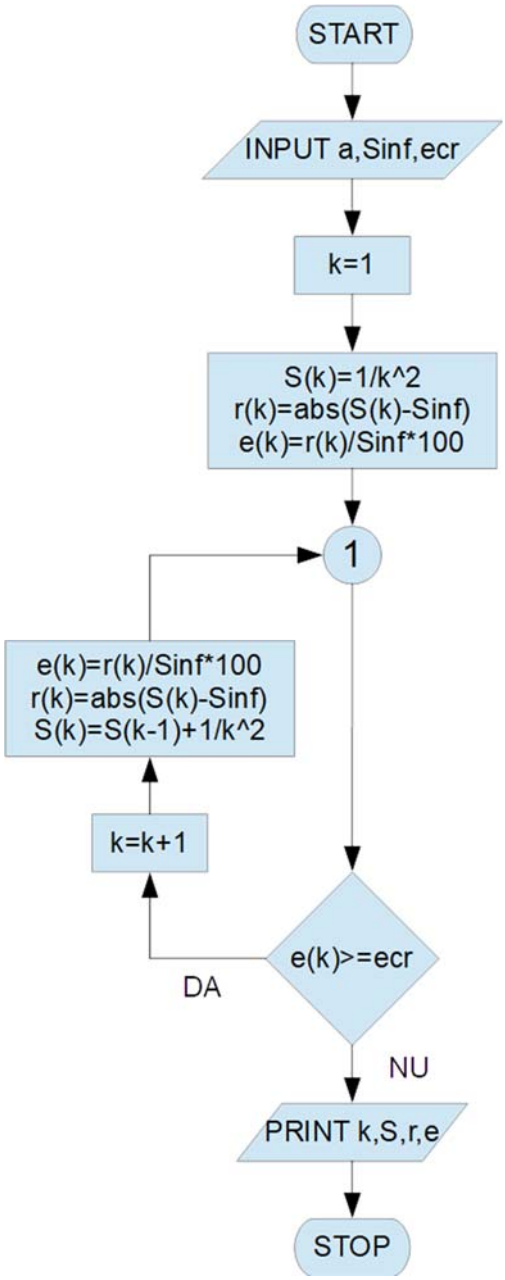
iterativă se face prin testarea valorii curente a contorului: pentru `k<n` se continuă iterațiile, în caz contrar se părăsește structura iterativă.

- Pentru schema logică din figura 3.122, b), intrarea în structura iterativă se face prin testarea valorii curente a erorii relative: dacă `e(k)>=ecr` se încep iterațiile, în caz contrar se părăsește structura iterativă. La fiecare iterație, definită prin valoarea curentă `k` a contorului, se calculează noua valoare `S(k)=S(k-1)+a(k)` din șirul sumelor parțiale, restul de ordin `k`, `r(k)=abs(S(k)-Sinf)`, precum și eroarea relativă față de valoarea exactă a sumei seriei `e(k)=r(k)/Sinf*100`.

- Datele de ieșire se referă la valorile sumelor parțiale de ordin `k`, la valorile resturilor de ordin `k`, respectiv la valorile erorilor relative de ordin `k`. În plus, pentru schema logică din figura 3.122, b) se afișează și numărul de iterații după care se părăsește structura iterativă.



a) număr impus de termeni



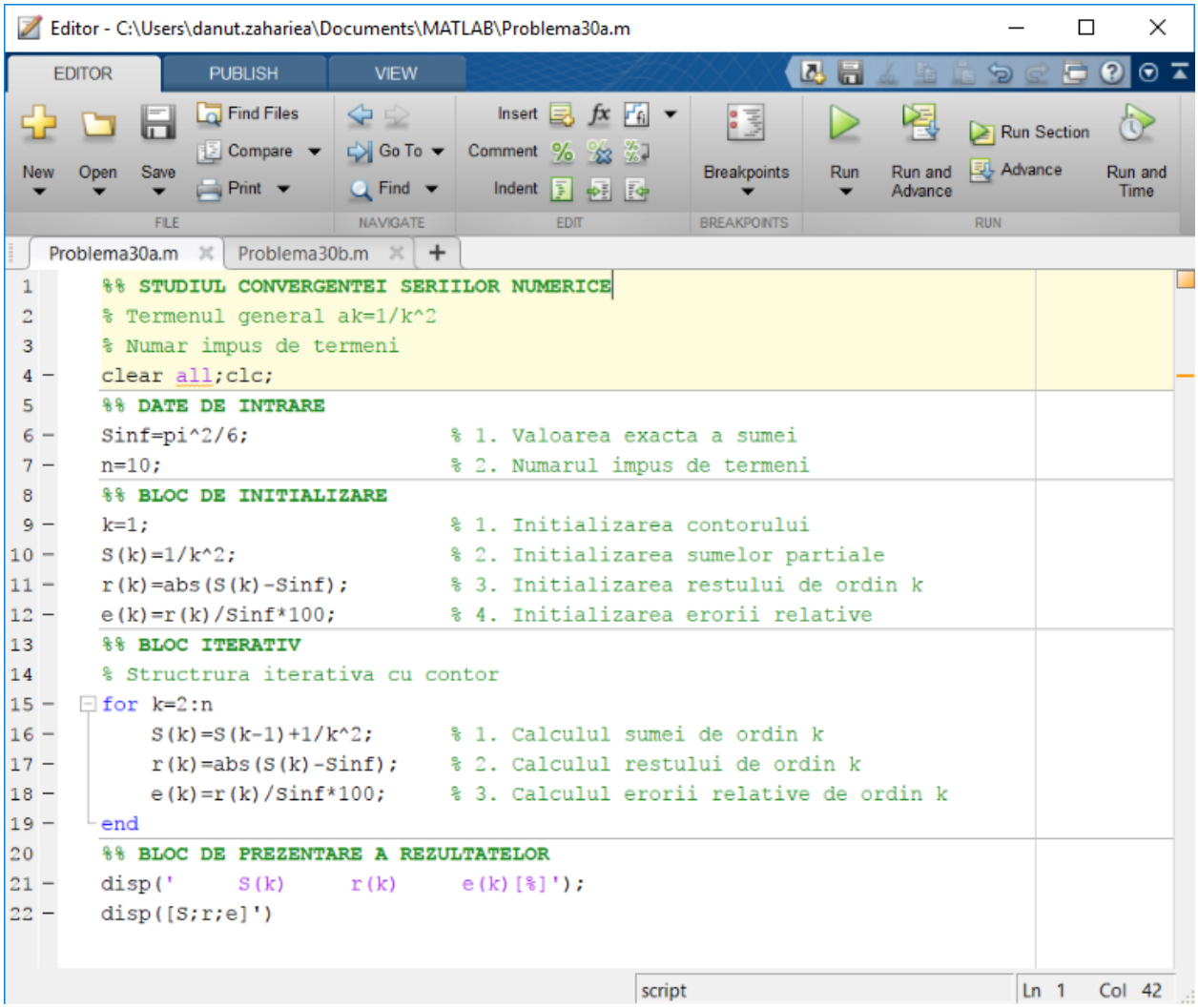
b) eroare relativă impusă

Figura 3.122. Scheme logice pentru studiul seriilor numerice.

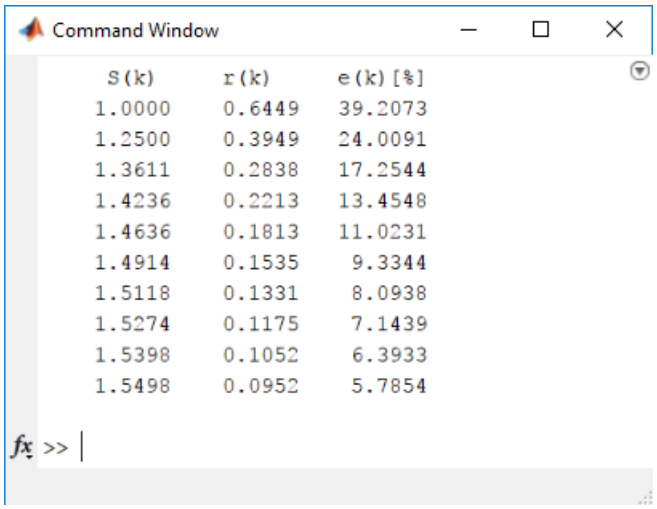
Fișierul de tip `script` pentru analiza seriei numerice cu număr impus de termeni este prezentat în figura 3.123, a). Fișierul `script` conține 5 secțiuni:

- Secțiunea de titlu care include și instrucțiunile `clear all` și `clc`.
- Secțiunea de introducere a datelor de intrare ale problemei (valoarea exactă a sumei și numărul impus de termeni).
- Secțiunea blocului de inițializare care conține inițializarea contorului `k=1` dar și a celor trei rezultate parțiale (suma parțială, restul și eroarea relativă de ordin `k=1`).
- Secțiunea de calcul iterativ. Cum rezultatele parțiale de ordin 1 au fost deja evaluate în blocul de inițializare, calculul iterativ va începe de la valoarea `k=2` a contorului. La fiecare iterație `k` se calculează rezultatele parțiale de ordin `k`. Iterațiile continuă inclusiv până la atingerea valorii impuse a numărului de termeni, `k=n`.
- Secțiunea de prezentare a rezultatelor conține două instrucțiuni, prima pentru afișarea unui text pe ecran reprezentând capul de tabel, iar cea de-a doua instrucțiune pentru prezentarea ordonată pe coloane a rezultatelor parțiale.

Rezultatul executării fișierului `script` este prezentat în figura 3.123, b).



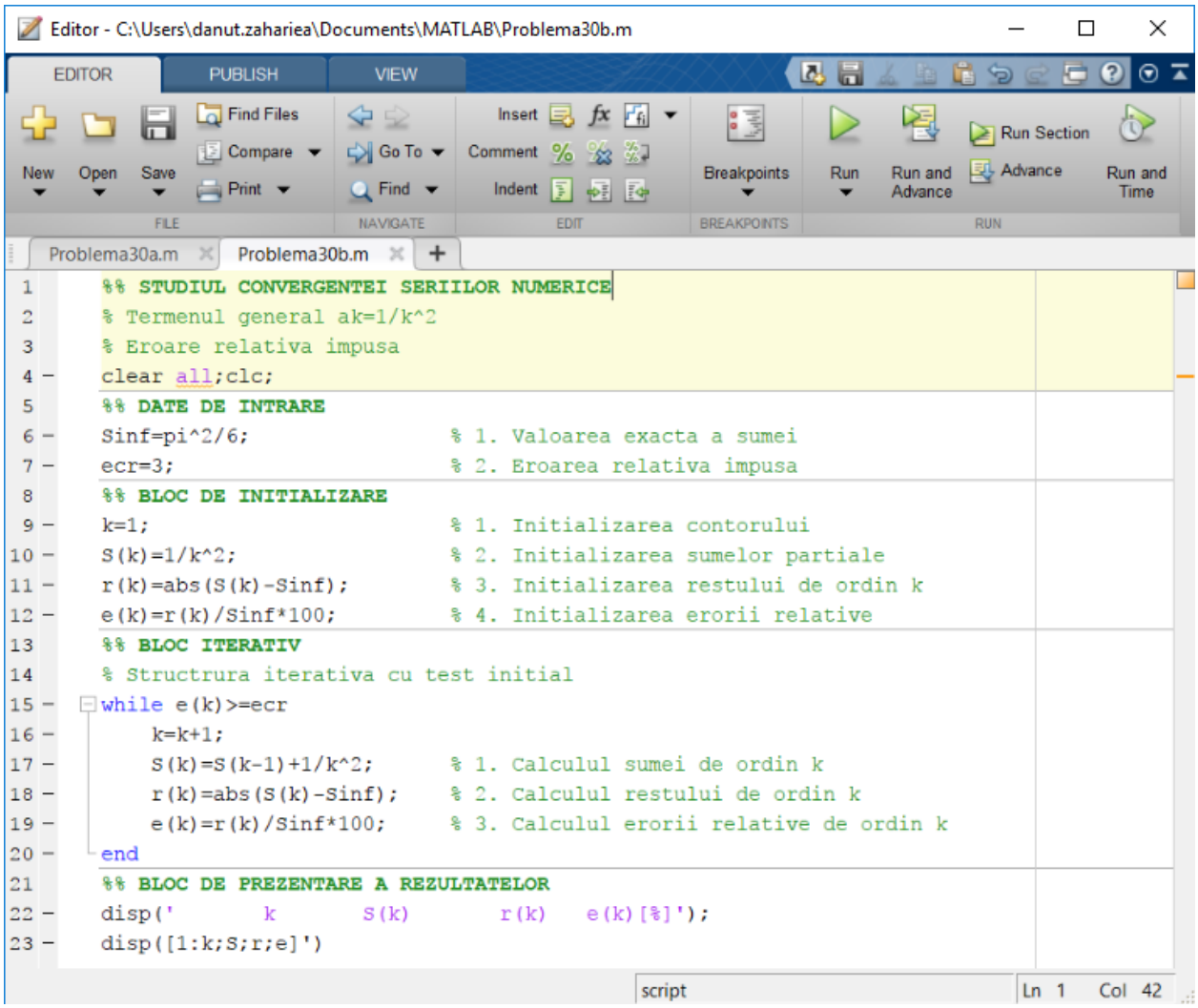
a) fișierul `script`



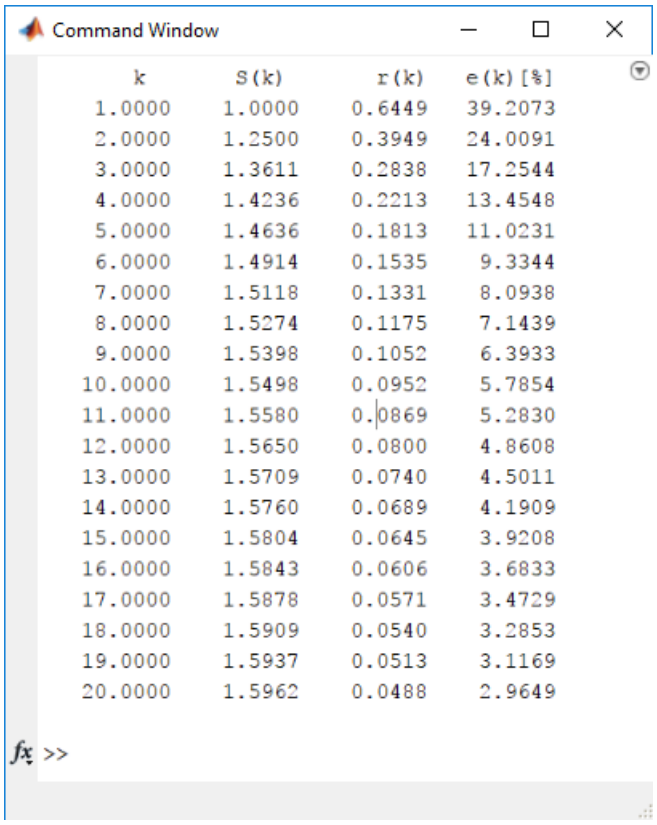
b) rezultatul obținut

Figura 3.123. Fișier `script` pentru studiul seriei numerice cu număr impus de termeni.

Fișierul de tip `script` pentru analiza seriei numerice cu eroare relativă impusă este prezentat în figura 3.124, a).



a) fișierul `script`



b) rezultatul obținut

Figura 3.124. Fișier `script` pentru studiul seriei numerice cu eroare relativă impusă.

Fișierul `script` conține 5 secțiuni:

- Secțiunea de titlu care include și instrucțiunile `clear all` și `clc`.
- Secțiunea de introducere a datelor de intrare ale problemei (valoarea exactă a sumei și eroarea relativă impusă).
- Secțiunea blocului de inițializare care conține inițializarea contorului `k=1` dar și a celor trei rezultate parțiale (suma parțială, restul și eroarea relativă de ordin `k=1`).
- Secțiunea de calcul iterativ. Rezultatele parțiale de ordin 1 au fost deja evaluate în blocul de inițializare, deci se poate testa valoarea erorii relative față de valoarea impusă. Calculul iterativ va începe dacă eroarea relativă obținută este mai mare sau egală cu valoarea impusă. La fiecare iterație se incrementează valoarea contorului `k=k+1`, după care se recalculează rezultatele parțiale. Iterațiile continuă până ce eroarea relativă devine mai mică față de valoarea impusă, `e(k)<ecr`, moment în care se obține și numărul necesar de iterații.
- Secțiunea de prezentare a rezultatelor conține două instrucțiuni, prima pentru afișarea unui text pe ecran reprezentând capul de tabel, iar cea de-a doua instrucțiune pentru prezentarea ordonată pe coloane a rezultatelor parțiale.

Rezultatul executării fișierului `script` este prezentat în figura 3.124, b).

Problema 3.52

Se consideră funcția $f: R \to R$ definită prin:

$$f(x) = \begin{cases} e^x + x - 1, \text{pentru } x \leq 1 \\ x^{1/(x-1)}, \text{pentru } x > 1 \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$ în mai multe puncte ale domeniului său de definiție. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru următoarele elemente ale domeniului de definiție:

```
x=[0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2].
```

Rezolvare

Schema logică este prezentată în figura 3.125.

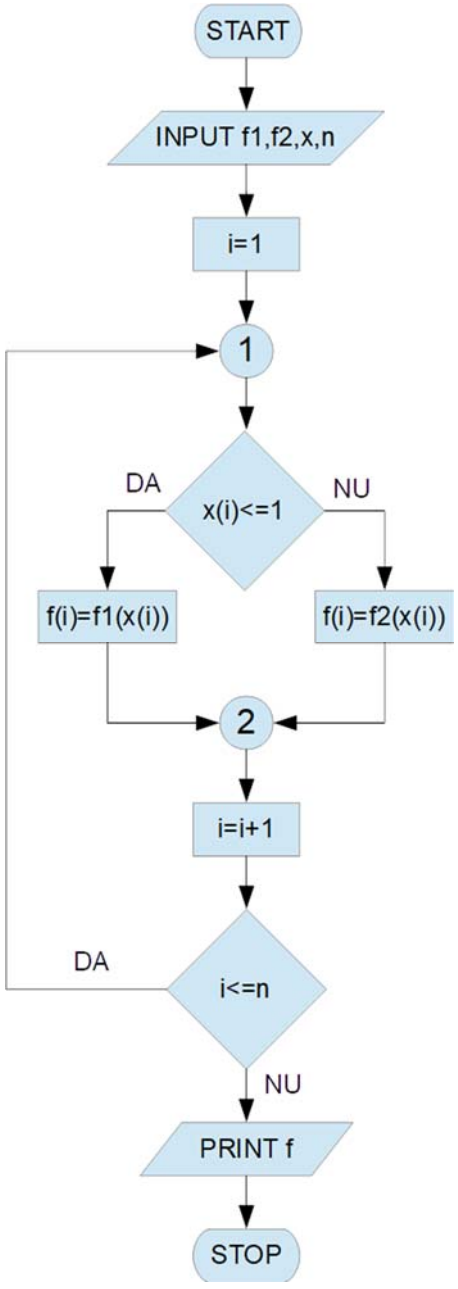


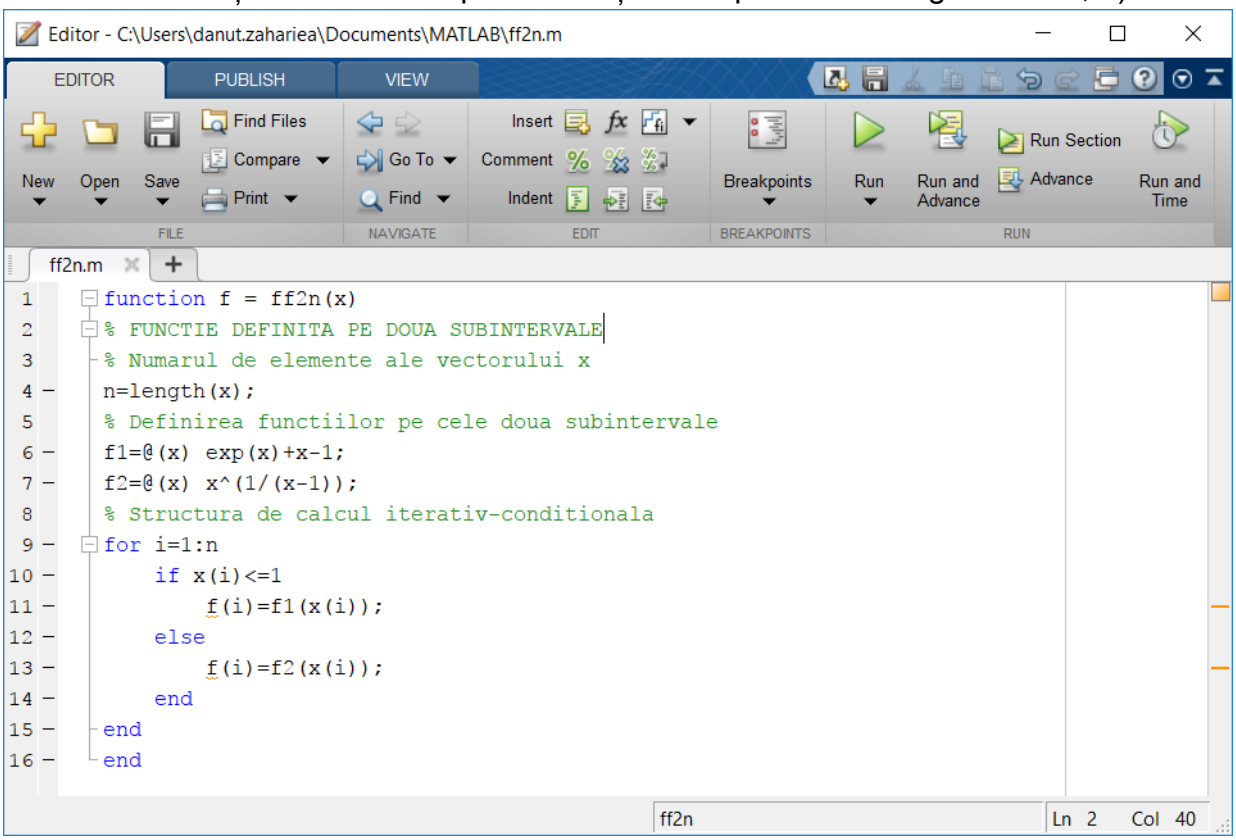
Figura 3.125. Schema logică pentru evaluarea unei funcții definite pe două subintervale.

Observații

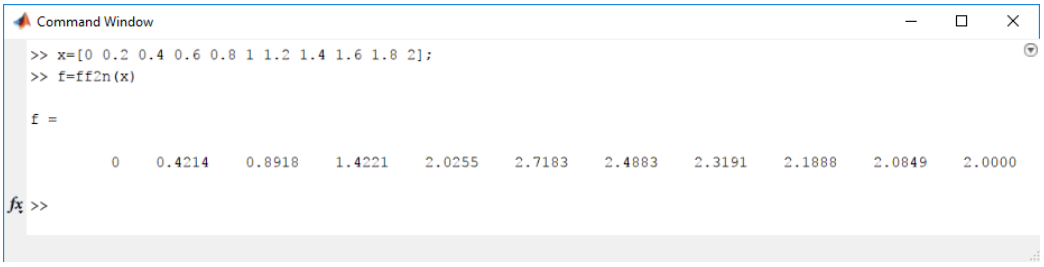
- Pentru evaluarea funcției `f` se utilizează o structură iterativă exterioară și o structură alternativă cu două ramuri inclusă în structura iterativă.
- Introducerea datelor de intrare constă în specificarea expresiilor funcțiilor pe cele două subintervale $f_1(x) = e^x + x - 1$ și $f_2(x) = x^{1/(x-1)}$, precum și a numărul n de elemente și a valorilor vectorului x.
- Schema logică conține apoi o structură exterioară iterativă în varianta inițializare-execuție instrucțiuni-incrementare-testare care prin valorile contorului `i`, parcurge toate elementele vectorului `x(i)`, și o structură interioară alternativă cu două ramuri care verifică, la fiecare iterație `i`, intervalul în care se găsește valoarea `x(i)` și în funcție de acesta, calculează valoarea funcției folosind expresia corespunzătoare intervalului respectiv.
- Indiferent însă de valorile vectorului `x`, în urma structurii iterativ-alternative se obțin valorile corespunzătoare ale funcției `f(x)`, care reprezintă și parametrul de ieșire al algoritmului.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.126, a). Numele funcției este `ff2n` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff2n.m`. Funcția acceptă un singur parametru de intrare, respectiv valorile vectorului `x`. Funcțiile specifice celor două subintervale se definesc ca funcții `anonymous` direct în fișierul `function`. În corpul funcției se află o structură iterativă de tip `for` și o structură alternativă de tip `if-else` prin care se va determina intervalul corespunzător fiecărei valori `x(i)`, și apoi în funcție de interval se va utiliza relația corespunzătoare de calcul pentru evaluarea funcției. Rezultatul obținut în urma evaluării funcției, `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a vectorului `x`. La apelare, elementele vectorului `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.126, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.126. Fișier `function` pentru evaluarea unei funcții definite pe două subintervale.

Problema 3.53

Se consideră funcția $f:[0,3] \to R$ definită prin:

$$f(x) = \begin{cases} e^x, \text{pentru } x \in [0,1) \\ \sin x, \text{pentru } x \in [1,2) \\ \ln x, \text{pentru } x \in [2,3] \end{cases}$$

Să se realizeze o schemă logică pentru evaluarea funcției $f(x)$ în mai multe puncte ale domeniului său de definiție. Să se definească un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru următoarele elemente ale domeniului de definiție:
`x=[0 0.25 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3].`

Rezolvare

Schema logică este prezentată în figura 3.127.

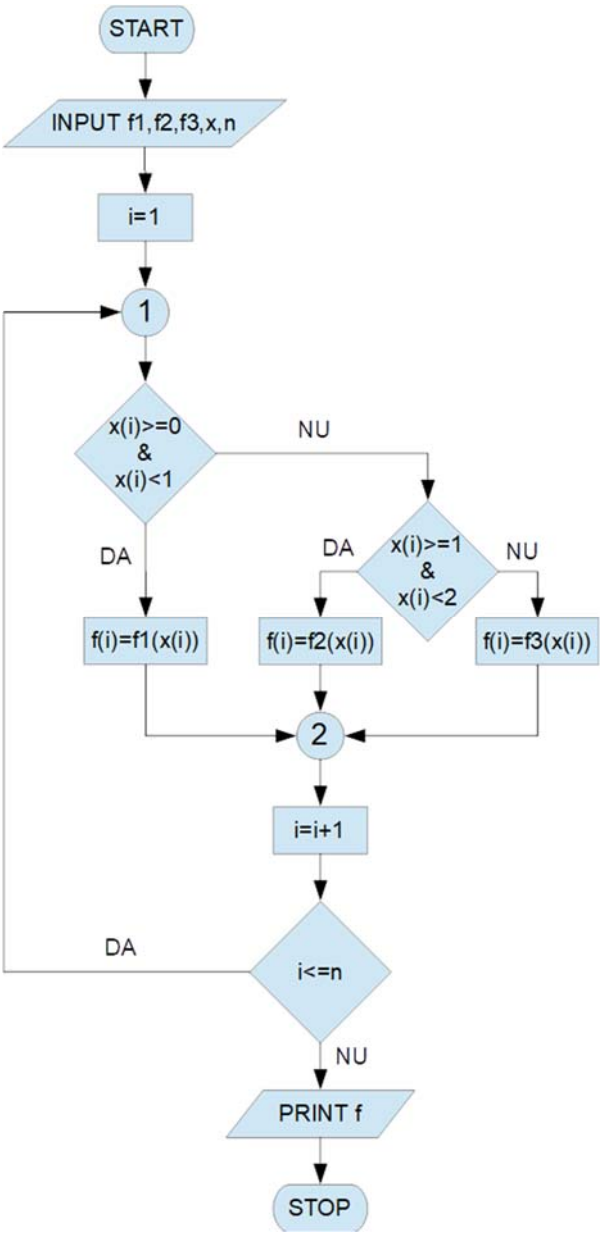


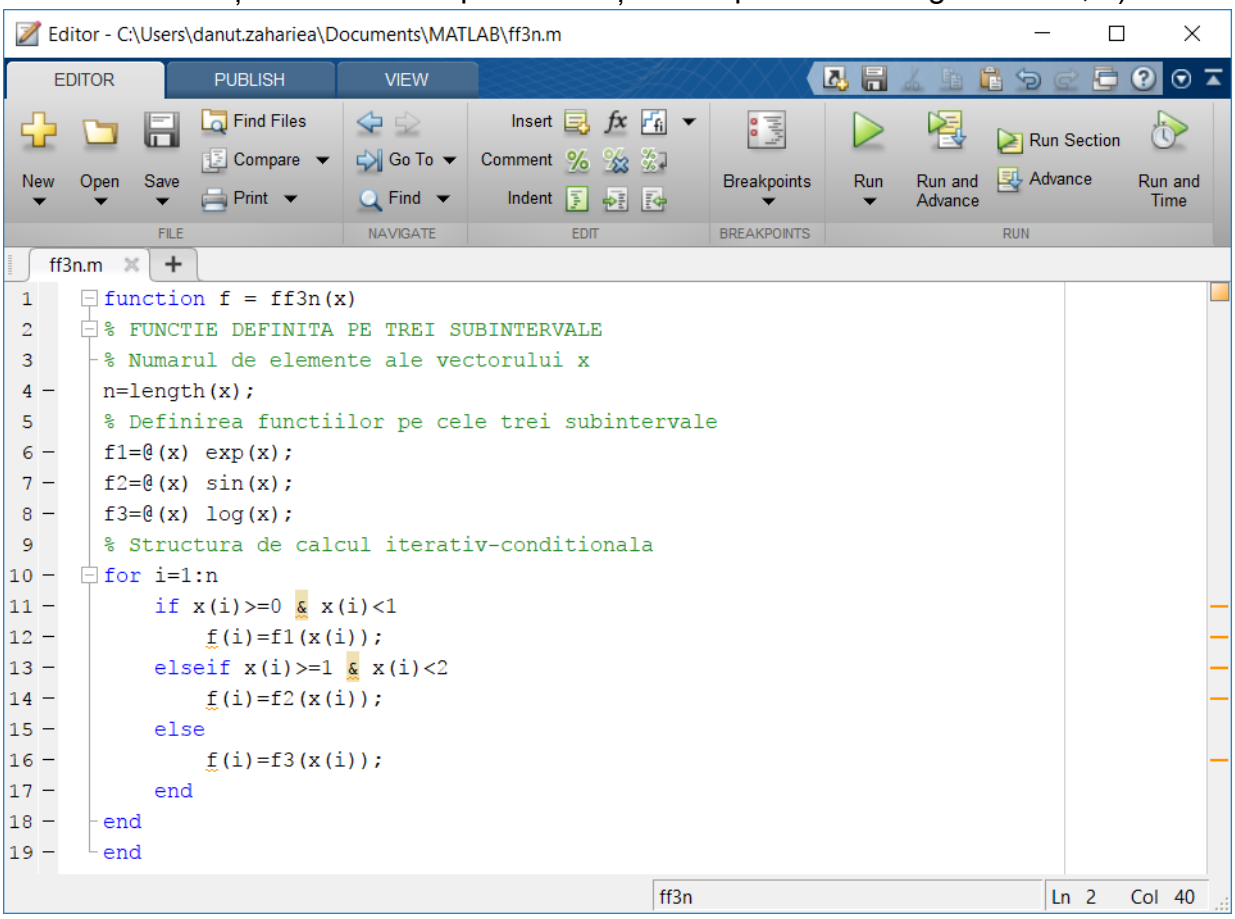
Figura 3.127. Schema logică pentru evaluarea unei funcții definite pe trei subintervale.

Observații

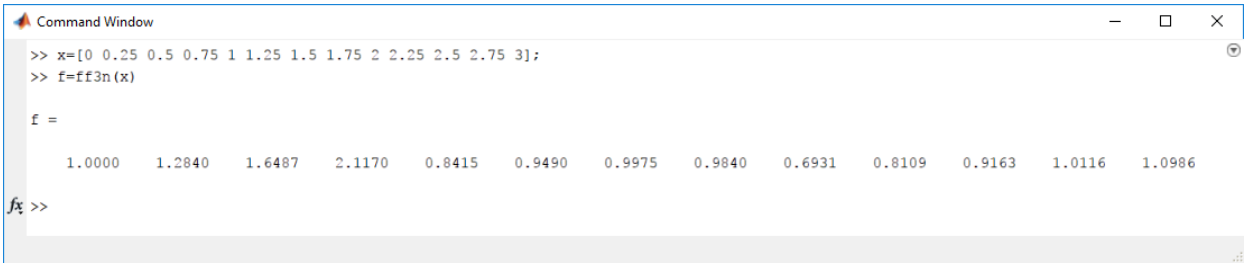
- Pentru evaluarea funcției `f` se utilizează o structură iterativă exterioară și două structuri alternative incluse în structura iterativă.
- Introducerea datelor de intrare constă în specificarea expresiilor funcțiilor pe cele trei subintervale $f_1(x) = e^x$, $f_2(x) = \sin x$ și $f_3(x) = \ln x$, precum și a numărul n de elemente și a valorilor vectorului x.
- Schema logică conține apoi o structură exterioară iterativă în varianta inițializare-execuție instrucțiuni-incrementare-testare care prin valorile contorului `i`, parcurge toate elementele vectorului `x(i)`, și două structuri interioare alternative care verifică, la fiecare iterație `i`, intervalul în care se găsește valoarea `x(i)` și în funcție de acesta, calculează valoarea funcției folosind expresia corespunzătoare intervalului respectiv.
- Indiferent însă de valorile vectorului `x`, în urma structurii iterativ-alternative se obțin valorile corespunzătoare ale funcției `f(x)`, care reprezintă și parametrul de ieșire al algoritmului.

Funcție care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.128, a). Numele funcției este `ff3n` și în mod corespunzător numele fișierului în care se va salva funcția va fi `ff3n.m`. Funcția acceptă un singur parametru de intrare, vectorul `x`. Funcțiile specifice celor trei subintervale se definesc ca funcții `anonymous` direct în fișierul `function`. În corpul funcției se află o structură iterativă de tip `for` și o structură alternativă de tip `if-elseif-else` prin care se va determina intervalul corespunzător fiecărei valori `x(i)`, și apoi în funcție de interval se va utiliza relația corespunzătoare de calcul pentru evaluarea funcției. Rezultatul obținut în urma evaluării funcției, `f` reprezintă parametrul de ieșire al funcției.

Funcția se apelează, în acest caz, din fereastra de comenzi după definirea prealabilă a vectorului `x`. La apelare, elementele vectorului `x` se transferă parametrului de intrare al funcției. Rezultatul apelării funcției este prezentat în figura 3.128, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.128. Fișier `function` pentru evaluarea unei funcții definite pe trei subintervale.

Problema 3.54

Se urmărește rambursarea unui împrumut în sumă de I=180000 lei contractat pe o perioadă de t=5 ani, cu rata dobânzii de r_d=10% pe an.

Să se calculeze:

- Sumele rămase de rambursat la începutul fiecărui an:

$$R_1 = I,\ R_{i+1} = R_i - a_i,\ \forall i = 1 \div t$$

- Dobânda de plată anuală:

$$d_i = R_i \cdot r_d,\ \forall i = 1 \div t$$

- Dobânda totală:

$$D = \sum_{i=1}^{t} d_i$$

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a parametrilor rambursării împrumutului: sumele rămase de rambursat, dobânzile anuale, dobânda totală. Să se scrie un fișier de tip `function` pentru implementarea schemei logice. Să se verifice pentru:

- Amortismente egale `a=[36000 36000 36000 36000 36000]` lei
- Amortismente inegale `a=[50000 50000 40000 30000 10000]` lei.

Rezolvare

Schema logică este prezentată în figura 3.129.

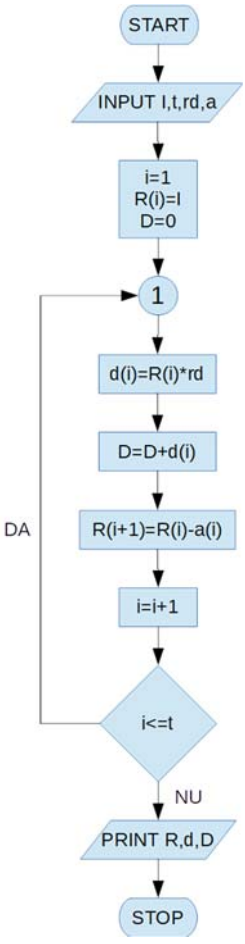


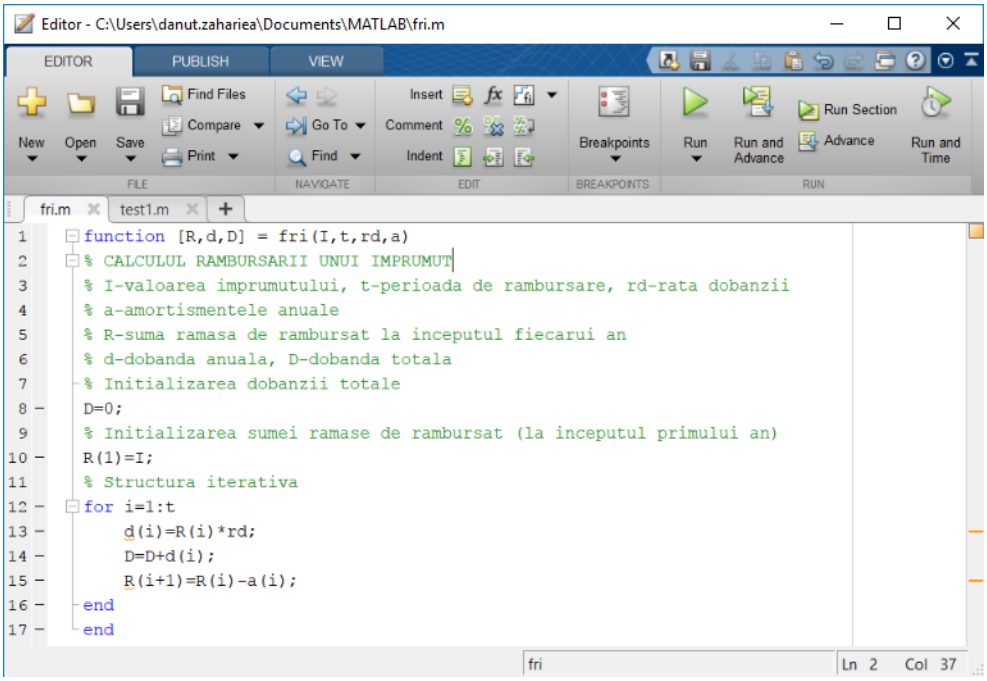
Figura 3.129. Schema logică pentru calculul rambursării unui împrumut.

Observații

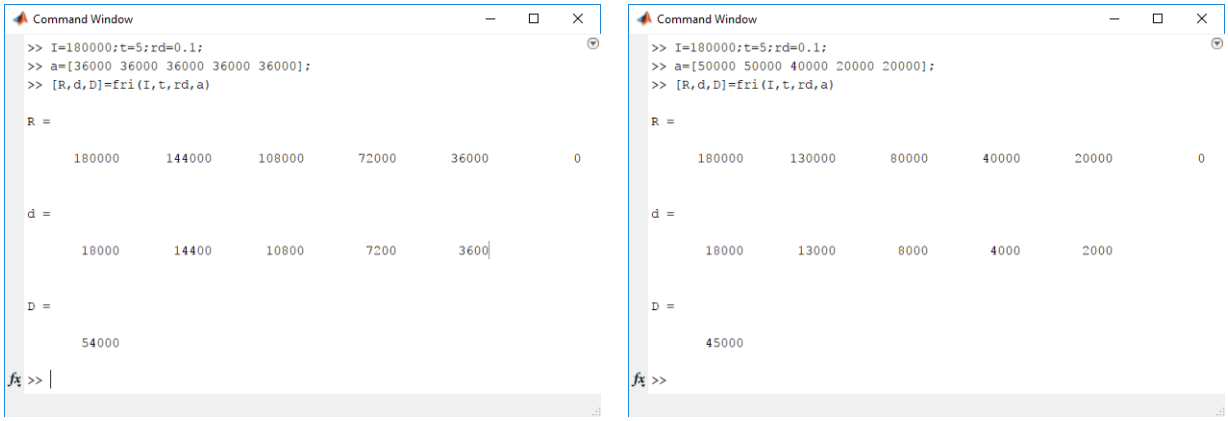
- Schema logică conține o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Datele de intrare ale algoritmului sunt valoarea împrumutului `I`, perioada de rambursare `t`, rata dobânzii `rd` și amortismentele `a`.
- Faza de inițializare a algoritmului cuprinde trei comenzi de atribuire, pentru contorul `i=1` al structurii iterative, pentru suma rămasă de plată la începutul primului an `R(1)=I` (chiar valoarea împrumutului) și pentru dobânda totală `D=0`.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se calculează dobânda anuală `d(i)=R(i)*rd`, dobânda totală `D=D+d(i)` și suma rămasă de plată la începutul anului următor `R(i+1)=R(i)-a(i)`, după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă `i` a contorului este sau nu mai mică decât valoarea sa maximă, `i<=t`. Dacă expresia logică este adevărată se continuă recalcularea parametrilor și incrementarea contorului. Dacă expresia logică este falsă, se părăsește structura iterativă.
- Parametrii de ieșire sunt sumele rămase de plată `R`, dobânzile anuale `d` și dobânda totală `D`.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.130, a). Numele funcției este `fri` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fri.m`. Funcția acceptă patru parametri de intrare, valoarea împrumutului `I`, perioada de rambursare `t`, rata dobânzii `rd` și amortismentele `a`. După inițializarea dobânzii totale (`D=0`) și a sumei rămase de rambursat la începutul primului an `R(1)=I`, urmează o structură iterativă cu număr cunoscut de iterații prin care se calculează dobânzile anuale `d(i)`, dobânda totală `D`, respectiv sumele rămase de rambursat la începutul anului următor `R(i+1)`. Variabilele `R`, `d` și `D` reprezintă și parametrii de ieșire ai funcției.

Funcția se apelează din fereastra de comenzi, în mod repetat, după definirea amortismentelor `a`. Rezultatul apelării funcției este prezentat în figura 3.130, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.130. Fișier `function` pentru calculul rambursării unui împrumut.

Problema 3.55

Se consideră doi vectori a şi b având aceeași dimensiune n și elementele:

$$a = [a_1\ a_2\ \dots a_n]$$
$$b = [b_1\ b_2\ \dots b_n]$$

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a vectorilor:

$$s = [s_1\ s_2\ \dots s_n]$$
$$d = [d_1\ d_2\ \dots d_n]$$

pentru care elementele s_i și d_i $(i = 1 \div n)$ se determină cu relațiile:

$$s_i = a_i + b_i$$
$$d_i = a_i - b_i$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru vectorii a=[1 3 5 7 9] şi b=[2 4 6 8 10].

Rezolvare

Schema logică este prezentată în figura 3.131.

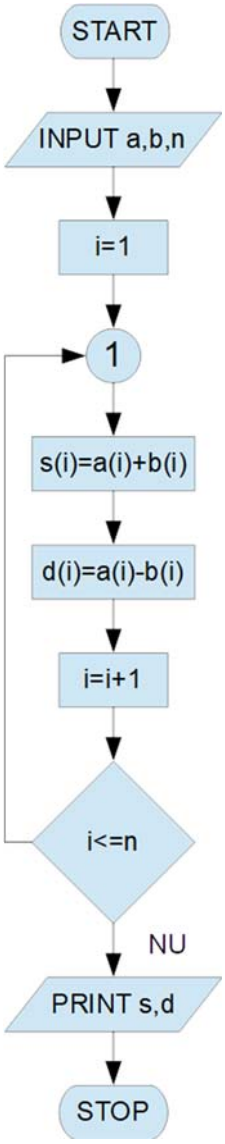


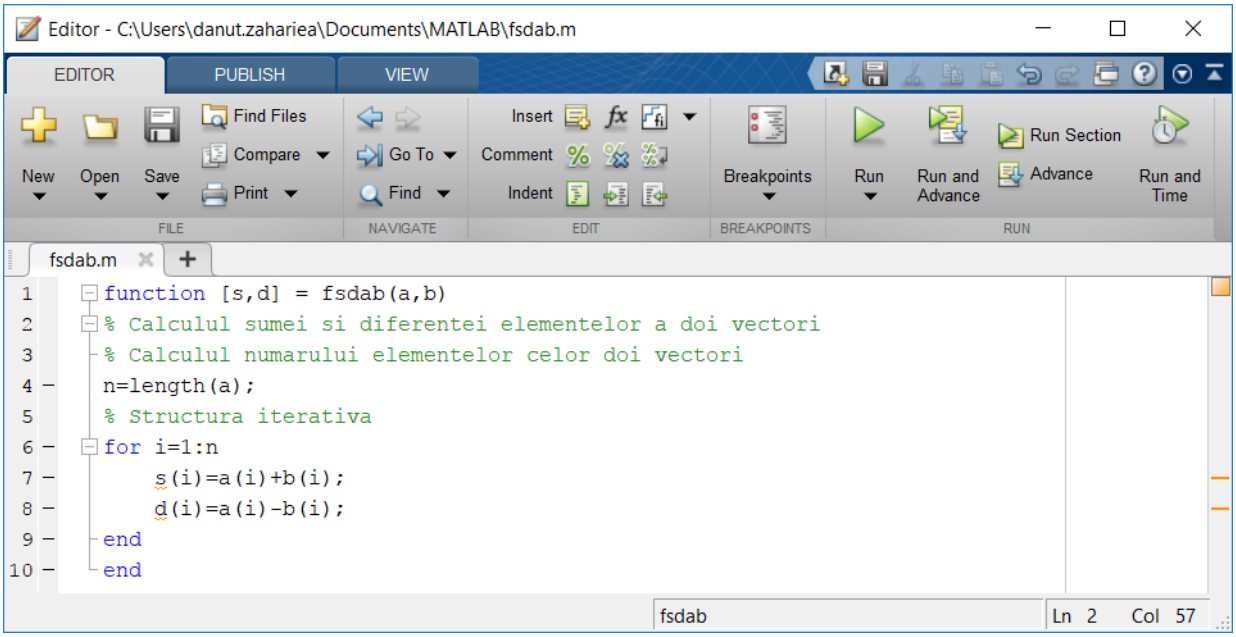
Figura 3.131. Schema logică pentru calculul sumei și diferenței elementelor a doi vectori.

Observații

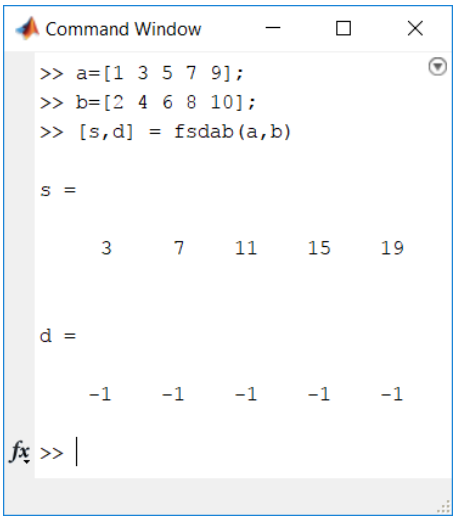
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde o singură comandă de atribuire, pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorii `a` și `b` declarați în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se calculează valoare sumei `s(i)` și diferenței `d(i)` dintre elementele având același indice `i` din cei doi vectori `a` și `b` folosind relațiile `s(i)=a(i)+b(i)` și `d(i)=a(i)-b(i)`, după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă calcularea sumei și diferenței următoarelor două elemente ale vectorilor `a` și `b`, după care urmează incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează toate valorile calculate pentru suma `s` și diferența `d`.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.132, a). Numele funcției este `fsdab` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fsdab.m`. Funcția acceptă doi parametri de intrare, respectiv variabilele vectoriale `a` și `b`. Determinarea numărului de elemente ale vectorului `a` se realizează cu instrucțiunea `n=length(a)`. Calculul sumei `s` și a diferenței `d` se efectuează cu o structură iterativă cu număr cunoscut de iterații cu ajutorul relațiilor `s(i)=a(i)+b(i)` și `d(i)=a(i)-b(i)`. Parametrii de ieșire ai funcției sunt suma `s` și diferența `d`.

Funcția se apelează, în acest caz, din fereastra de comenzi (`Command Window`) după definirea prealabilă a celor doi vectori, `a` și `b`. La apelare, elementele vectorilor `a` și `b` se transferă parametrilor de intrare ai funcției. Rezultatul apelării funcției este prezentat în figura 3.132, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.132. Fișier `function` pentru calculul sumei și diferenței elementelor a doi vectori.

Problema 3.56

Se consideră doi vectori a şi b având elementele:

$$a = [a_1\ a_2\ \dots a_n]$$
$$b = [b_1\ b_2\ \dots b_n]$$

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a vectorilor:

$$p = [p_1\ p_2\ \dots p_n]$$
$$c = [c_1\ c_2\ \dots c_n]$$

pentru care elementele p_i și c_i $(i = 1 \div n)$ se determină cu relațiile:

$$p_i = a_i \cdot b_i$$
$$c_i = \frac{a_i}{b_i}$$

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru vectorii a=[1 3 5 7 9] şi b=[2 4 6 8 10].

Rezolvare

Schema logică este prezentată în figura 3.133.

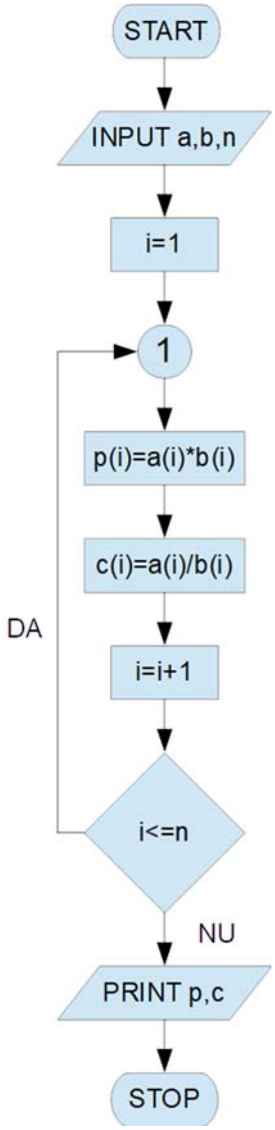


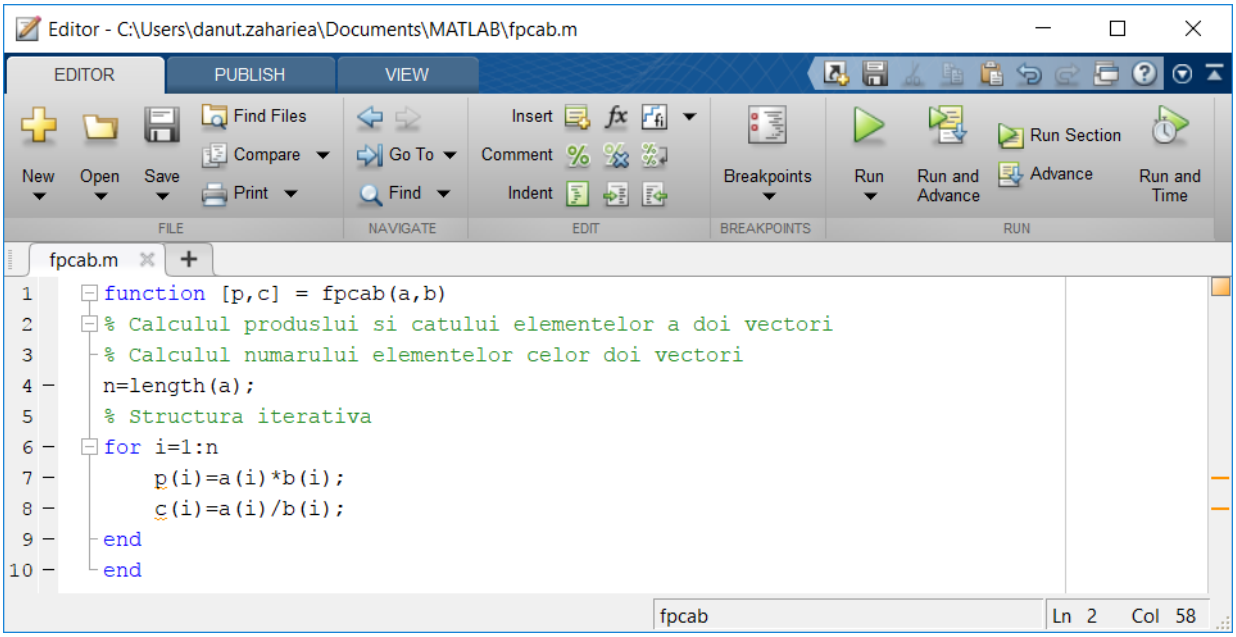
Figura 3.133. Schema logică pentru calculul produsului și câtului elementelor a doi vectori.

Observații

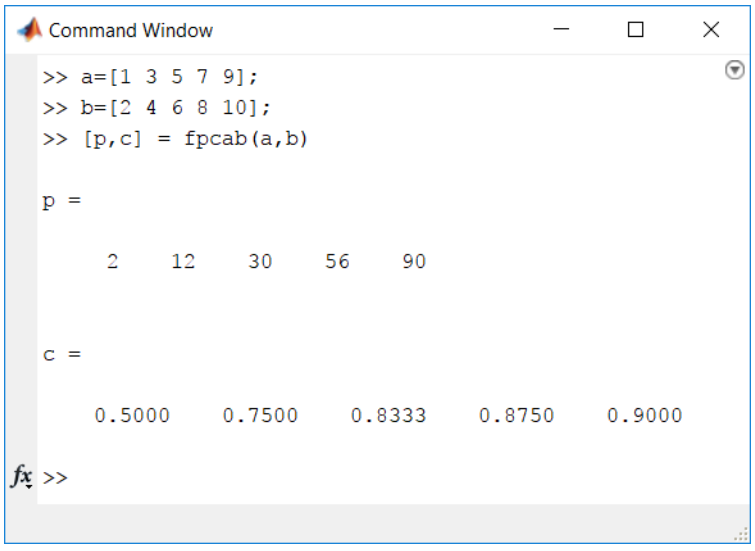
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde o singură comandă de atribuire, pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorii `a` și `b` declarați în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se calculează valoare produsului `p(i)` și câtului `c(i)` dintre elementele având același indice `i` din cei doi vectori `a` și `b` folosind relațiile `p(i)=a(i)*b(i)` și `c(i)=a(i)/b(i)`, după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă calcularea produsului și câtului următoarelor două elemente ale vectorilor `a` și `b`, după care urmează incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează toate valorile calculate pentru produsul `p` și câtul `c`.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.134, a). Numele funcției este `fpcab` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fpcab.m`. Funcția acceptă doi parametri de intrare, respectiv variabilele vectoriale `a` și `b`. Determinarea numărului de elemente ale vectorului `a` se realizează cu instrucțiunea `n=length(a)`. Calculul produsului `p` și a câtului `c` se efectuează cu o structură iterativă cu număr cunoscut de iterații cu ajutorul relațiilor `p(i)=a(i)*b(i)` și `c(i)=a(i)/b(i)`. Parametrii de ieșire ai funcției sunt produsul `p` și câtul `c`.

Funcția se apelează, în acest caz, din fereastra de comenzi (`Command Window`) după definirea prealabilă a celor doi vectori, `a` și `b`. La apelare, elementele vectorilor `a` și `b` se transferă parametrilor de intrare ai funcției. Rezultatul apelării funcției este prezentat în figura 3.134, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.134. Fișier `function` pentru calculul produsului și câtului elementelor a doi vectori.

Problema 3.57

Se consideră doi vectori a şi b având elementele:

$$a = [a_1\ a_2\ \dots a_n]$$
$$b = [b_1\ b_2\ \dots b_n]$$

Produsul scalar al celor doi vectori se calculează cu relația:

$$PS = \sum_{i=1}^{n} a_i b_i$$

Să se realizeze o schemă logică pentru descrierea algoritmului de determinare a produsul scalar al celor doi vectori.

Să se scrie un fișier de tip `function` pentru implementarea schemei logice.

Să se verifice pentru vectorii a=[1 3 5 7 9] şi b=[2 4 6 8 10].

Rezolvare

Schema logică este prezentată în figura 3.135.

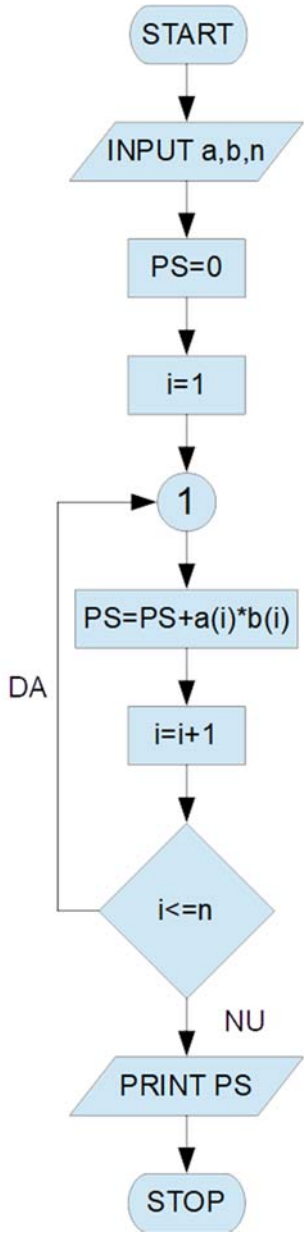


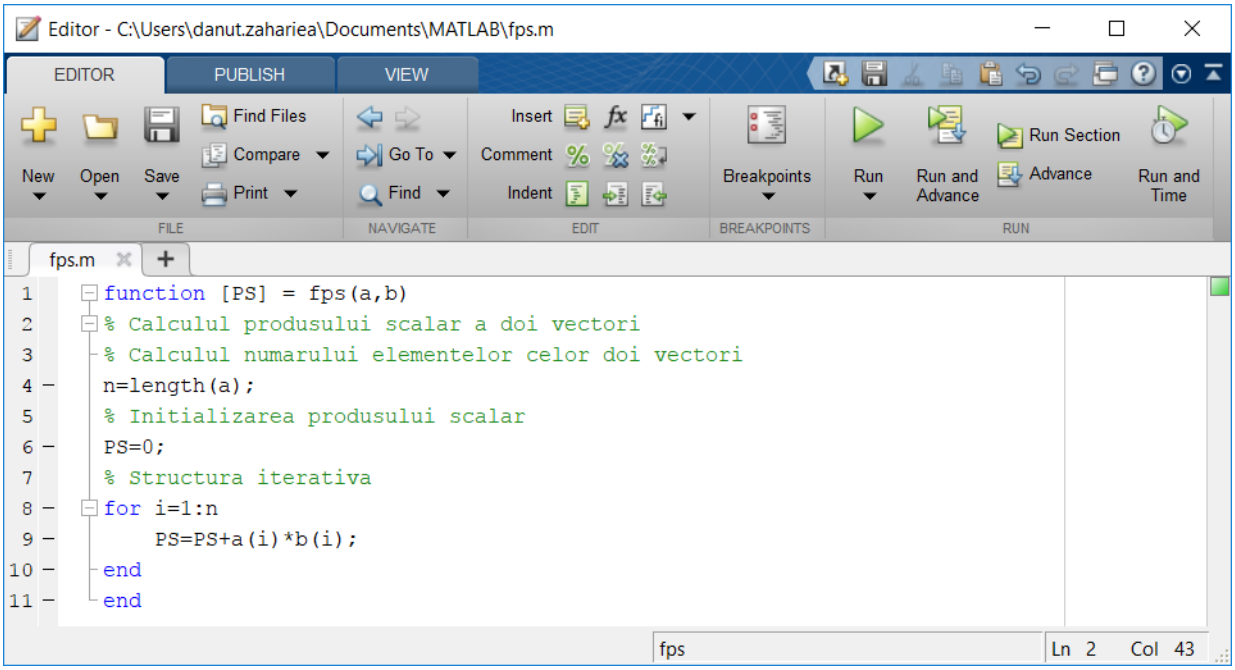
Figura 3.135. Schema logică pentru calculul produsului scalar a doi vectori.

Observații

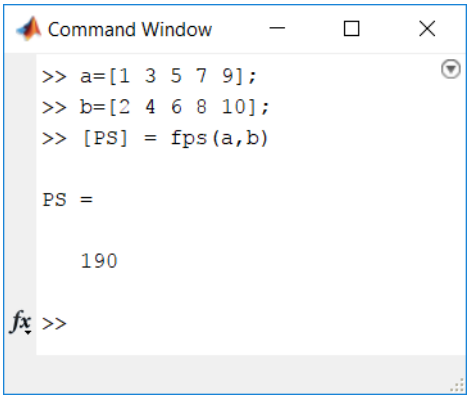
- Schema logică se bazează pe o structură iterativă cu număr cunoscut de iterații în varianta inițializare-execuție instrucțiuni-incrementare-testare.
- Faza de inițializare a algoritmului cuprinde două comenzi de atribuire, pentru produsul scalar `PS=0` și pentru contorul `i=1` al structurii iterative care va parcurge toți cei `n` indici ai valorilor din vectorii declarați în faza de introducere a datelor de intrare.
- La fiecare iterație, definită prin valoare curentă `i` a contorului, se recalculează valoare produsului scalar prin adăugare produsului dintre valorile celor doi vectori corespunzătoare indicelui curent `i` (`a(i)*b(i)`) la valoarea anterioară a produsului scalar (`PS=PS+a(i)*b(i)`), după care se incrementează valoarea contorului cu o unitate, `i=i+1`.
- După fiecare iterație urmează o structură alternativă care verifică dacă valoarea curentă a contorului este sau nu mai mică decât valoarea maximă a contorului, `i<=n`.
- Dacă expresia logică este adevărată se continuă recalcularea produsului scalar și incrementarea în continuare a contorului.
- Dacă expresia logică este falsă, se părăsește structura iterativă cu număr cunoscut de iterații și se afișează ultima valoare calculată a produsului scalar.

Funcția care implementează algoritmul descris în schema logică este definită în fișierul de tip `function` prezentat în figura 3.136, a). Numele funcției este `fps` și în mod corespunzător numele fișierului în care se va salva funcția va fi `fps.m`. Funcția acceptă doi parametri de intrare, respectiv variabilele vectoriale `a` și `b`. Determinarea numărului de elemente ale vectorului `a` se realizează cu instrucțiunea `n=length(a)`. Calculul produsului scalar `PS` se efectuează cu o structură iterativă cu număr cunoscut de iterații cu ajutorul relației `PS=PS+a(i)*b(i)`. Parametrul de ieșire al funcției este produsul scalar al celor doi vectori `PS`.

Funcția se apelează, în acest caz, din fereastra de comenzi (`Command Window`) după definirea prealabilă a celor doi vectori, `a` și `b`. La apelare, elementele vectorilor `a` și `b` se transferă parametrilor de intrare ai funcției. Rezultatul apelării funcției este prezentat în figura 3.136, b).



a) fișierul `function`



b) rezultatul obținut

Figura 3.136. Fișier `function` pentru calculul produsului scalar a doi vectori.

CAPITOLUL 4
REPREZENTAREA GRAFICĂ A FUNCŢIILOR 2D

4.1. INSTRUCŢIUNEA `plot`

4.1.1. Reprezentarea unei singure funcţii.

Se consideră funcţia: $f: [x_{min}, x_{max}] \to \mathbb{R}$, definită prin: $y = f(x)$.

Pentru reprezentarea grafică a dependenţei $y = f(x)$ se utilizează instrucţiunea:

```
plot(x,y,'pf')
```

în care: `x` reprezintă valorile de pe abscisă, `y` valorile de pe ordonată, iar `pf` sunt parametrii de formatare ai graficului.

Formatarea curbelor se face prin intermediul a trei caracteristici:

- tip marker (`+`, `o`, `*`, `x`, `s`, `d`, `^`, `v`, `<`, `>`, `p`, `h`),
- tip linie (`-` linie continuă, `--` linie întreruptă, `:` linie punctată, `-.` linie punct),
- culoare (`r` roşu, `g` verde, `b` albastru, `m` mov, `y` galben, `k` negru, `w` alb).

4.1.2. Reprezentarea a două funcţii în ferestre grafice diferite

Se consideră funcţiile:

$$f_1: [x_{min}, x_{max}] \to \mathbb{R},\ y_1 = f_1(x)$$
$$f_2: [x_{min}, x_{max}] \to \mathbb{R},\ y_2 = f_2(x)$$

Pentru reprezentarea grafică a celor două dependenţe $y_1 = f_1(x)$ şi $y_2 = f_2(x)$ în două ferestre grafice diferite se utilizează instrucţiunile:

```
figure
plot(x,y1,'-k');
grid on;
figure
plot(x,y2,'-.k');
grid off;
```

Instrucțiunea `figure` generează un nou obiect grafic de tip `figure`.

4.1.3. Reprezentarea a două funcții în același obiect grafic `axes`

Se consideră funcţiile:

$$f_1: [x_{min}, x_{max}] \to \mathbb{R},\ y_1 = f_1(x)$$
$$f_2: [x_{min}, x_{max}] \to \mathbb{R},\ y_2 = f_2(x)$$

Pentru reprezentarea grafică a celor două dependenţe $y_1 = f_1(x)$ şi $y_2 = f_2(x)$ pe aceleaşi axe se poate utiliza una din variantele:

```
plot(x,y1,'-r',x,y2,'-.b');
```

sau

```
plot(x,y1,'-k');
hold on;
plot(x,y2,'-.k');
hold off;
```

În cazul reprezentării unui număr ridicat de funcţii se preferă cea de-a doua metodă. Pentru obţinerea rezultatului final instrucţiunea `plot` a fost aplicată, în acest caz, de două ori, odată pentru trasarea graficului funcţiei $y_1 = f_1(x)$ şi a două oară pentru obţinerea graficului funcţiei $y_2 = f_2(x)$. În acest caz, cele două instrucţiuni `plot` trebuie să afişează rezultatele în aceeaşi fereastră grafică şi în acelaşi obiect grafic de tip `axes`, fiind obligatorie intercalarea între cele două instrucţiuni `plot` a instrucţiunii:

```
hold on
```

Formatarea celor două curbe ajută la identificarea precisă a fiecăreia şi este necesară ori de câte ori pe aceeaşi fereastră grafică se reprezintă mai multe curbe. Astfel, funcţia $y_1 = f_1(x)$ se reprezintă cu linie continuă de culoare roşie, iar funcţia $y_2 = f_2(x)$ se reprezintă cu linie punct de culoare albastră.

Identificarea clară a graficelor se realizează prin intermediul adnotării de tip legendă. Aplicarea adnotării de tip legendă se face cu instrucţiunea:

```
legend('y_1=f_1(x)','y_2=f_2(x)')
```

în care specificaţiile curbelor care reprezintă parametrii instrucţiunii `legend` trebuie introduse între apostrofuri şi separate prin virgulă.

4.2. INSTRUCŢIUNEA `PLOTYY`

Pentru realizarea obiectelor grafice de tip `axes` cu două axe `y`, una plasată la stânga şi cealaltă plasată la dreapta obiectului grafic `axes` se utilizează instrucţiunea `plotyy`. Această instrucţiune este foarte utilă în cazul în care se urmăreşte reprezentarea grafică în acelaşi obiect grafic de tip `axes` a două funcţii f_1 şi f_2 care au acelaşi domeniu de definiţie:

$$f_1: [x_{min}, x_{max}] \to \mathbb{R}$$
$$f_2: [x_{min}, x_{max}] \to \mathbb{R}$$

dar care au valori $y_1 = f_1(x)$ şi $y_2 = f_2(x)$ mult diferite între ele.

Astfel, pentru reprezentarea grafică a celor două dependenţe $y_1 = f_1(x)$ şi $y_2 = f_2(x)$ în aceeaşi fereastră grafică şi pe aceleaşi axe se utilizează instrucţiunile:

```
plotyy(x,y1,x,y2);grid;
plotyy(x,y1,'-b',x,y2,'-.r');grid;
```

Se observă în cel de-al doilea caz introducerea şi a parametrilor de formatare ci celor două curbe: prima curbă va fi reprezentată cu linie continuă de culoare albastră, în timp ce a doua curbă va fi reprezentată cu linie întreruptă de culoare roşie.

4.3. GRAFICE ÎN COORDONATE LOGARITMICE

Pentru reprezentarea grafică a funcţiilor utilizând coordonate logaritmice (logaritm în baza 10) se pot utiliza următoarele instrucţiuni:

- `loglog`. Permite realizarea graficelor 2D cu ambele axe trasate în coordonate logaritmice.
- `semilogx`. Permite realizarea graficelor 2D doar cu abscisa trasată în coordonate logaritmice.
- `semilogy`. Permite realizarea graficelor 2D doar cu ordonata trasată în coordonate logaritmice.

4.4. GRAFICE ÎN TREPTE

Se consideră funcţia: $f: [x_{min}, x_{max}] \rightarrow \mathbb{R}$, definită prin: $y = f(x)$.

Pentru reprezentarea grafică a funcţiei $y = f(x)$ folosind metoda graficului în trepte se utilizează instrucţiunea:

```
stairs(x,y)
```

4.5. GRAFICE CU BARE

Se consideră funcţia: $f: [x_{min}, x_{max}] \rightarrow \mathbb{R}$, definită prin: $y = f(x)$.

Pentru reprezentarea grafică a funcţiei $y = f(x)$ folosind metoda graficului cu bare verticale se utilizează instrucţiunea:

```
bar(x,y)
```

Pentru reprezentarea grafică a unei funcţiei $y = f(x)$ folosind metoda graficului cu bare orizontale se utilizează instrucţiunea:

```
barh(x,y)
```

În cazul reprezentării grafice cu bare a două funcţii care au acelaşi domeniu de definiţie, atunci trebuie definită o matrice având pe fiecare coloană valorile funcţiilor respective. În acest caz, opţiunile de reprezentare ale comenzii `bar (barh)` sunt `group` și `stack`.

4.6. REPREZENTAREA GRAFICĂ A HISTOGRAMEI

Se consideră un eşantion de volum n format din valorile ordonate crescător:

$x_1, x_2, \dots, x_{n-1}, x_n$ cu $x_{min} = x_1$ şi $x_{max} = x_n$

Amplitudinea sondajului, definită prin $A = x_{max} - x_{min}$, se împarte într-un număr k de intervale egale, numite clase. Amplitudinea unei clase este: $a = A/n$.

Frecvenţa absolută a claselor $i = 1 \dots k$ se notează cu n_i, $i = 1 \dots k$ şi reprezintă numărul de rezultate ale eşantionului care se găsesc în fiecare interval.

Reprezentarea grafică a frecvenţelor absolute în funcţie de intervalele de valori corespunzătoare poartă numele de histograma repartiţiei.

Pentru realizarea grafică a histogramei se pot utiliza următoarele instrucţiuni:

- Pentru coordonate carteziene:

  ```
  hist(x,k)
  ```

  în care x reprezintă vectorul valorilor eşantionului de analizat, iar k reprezintă numărul de clase considerat. În cazul în care parametrul k se omite, atunci se consideră în mod implicit că numărul de clase este k=10.

- Pentru coordonate polare:

  ```
  rose(t,k)
  ```

  în care t [rad] reprezintă vectorul valorilor unghiulare ale eşantionului de analizat, iar k reprezintă numărul de clase considerat. În cazul în care parametrul k se omite, atunci se consideră în mod implicit că numărul de clase este k=20.

4.7. REPREZENTAREA GRAFICĂ A ERORILOR

Se consideră că la în urma efectuării unui experiment rezultatele obţinute se pot scrie sub forma:

$$y = \bar{x} \pm e$$

în care $\bar{x}$ reprezintă valorile medii ale rezultatelor experimentale obţinute în fiecare punct de măsură x, iar e reprezintă intervalul de eroare corespunzător fiecărui punct de măsurare. Valorile $\bar{x}$ şi e rezultă în urma repetării procesului de măsurare în condiţii practic identice de un anumit număr de ori, pentru fiecare punct de măsură x şi prin efectuare apoi a analizei statistice a datelor experimentale. Rezultatele obţinute în urma analizei statistice a datelor experimentale sunt valabile în condiţiile impunerii unei anumite valori a nivelului de încredere P [%] (de obicei 95% sau 99%).

Reprezentarea grafică a intervalelor de eroare e asociate setului de date $\{x, \bar{x}\}$ se realizează cu ajutorul instrucţiunii:

```
errorbar(x,xm,e)
```

4.8. GRAFICE DE TIP `AREA`

Pentru reprezentarea grafică de tip area se utilizează instrucţiunea:

```
area(y)
```

în care y reprezintă un vector de date. Suprafaţa dintre curba obţinută şi axa absciselor va fi colorată. În cazul în care y este o matrice se reprezintă fiecare coloană a matricei sub forma unor curbe suprapuse una peste alta, colorându-se suprafeţele dintre curbe.

În cazul instrucţiunii:

```
area(x,y)
```

rezultatul este identic cu cel al utilizării instrucţiunii `plot`, cu deosebirea că se colorează suprafaţa de sub curbă.

4.9. GRAFICE DE TIP `PIE`

Pentru reprezentarea grafică a unui set de date în care fiecare element reprezintă un procent dintr-un întreg, se recomandă utilizarea instrucţiunii:

```
pie(y)
```

în care y reprezintă un vector având n date care are proprietatea că suma elementelor reprezintă un întreg (de exemplu, 100%), astfel încât fiecare element al vectorului de date y este pus în corespondenţă cu valoarea sa procentuală.

În cazul instrucţiunii:

```
pie(y,explode)
```

în care `explode` este un vector având aceeaşi dimensiune ca vectorul de date y, dar având doar elemente de 0 şi 1, rezultatul este identic cu cel al utilizării instrucţiunii `pie(y)`, cu deosebirea că toate sectoarele cărora le corespund valorile 1 vor fi separate din suprafaţa circulară a figurii (sectoare explodate).

În cazul instrucţiunilor:

```
pie3(y)
pie3(y,explode)
```

rezultatul este identic cu cel al utilizării instrucțiunilor `pie(y)` și `pie(y,explode)` cu deosebirea că toate sectoarele vor fi reprezentate în vedere tridimensională.

4.10. GRAFICE DE TIP `FILL`

Se consideră un număr de n puncte având coordonatele x_i şi y_i, $i = 1 \dots n$. Pentru reprezentarea grafică a suprafeţei interioare conturului poligonal definit de punctele considerate $P_i(x_i, y_i), i = 1 \dots n$ şi colorarea suprafeţei respective folosind culoarea c, se utilizează instrucţiunea:

```
fill(x,y,'c')
```

Aria suprafeţei interioare a poligonului astfel definit se poate obţine folosind instrucţiunea:

```
A=polyarea(x,y)
```

în care x şi y sunt vectorii coordonatelor colţurilor poligonului.

4.11. GRAFICE ÎN COORDONATE POLARE

Instrucţiunea `polarplot` permite reprezentarea grafică a funcţiilor în coordonate polare (r, θ). Forma generală de implementare a instrucţiunii este:

```
polarplot(theta,r)
```

în care `theta` reprezintă unghiul în radiani dintre raza vectoare și axa Ox, iar `r` reprezintă lungimea razei vectoare.

Pentru graficele în coordonate polare instrucţiunile de adnotare ale axelor sau ale rețelei `grid` nu au relevanță.

4.12. INSTRUCŢIUNEA `SUBPLOT`

Reprezentarea a două funcții în același obiect grafic `figure` dar în două obiecte grafice de tip `axes` diferite se realizează prin crearea unei structuri de obiecte grafice de tip `axes`, adică a mai multor obiecte grafice de tip `axes` dispuse în interiorul aceluiaşi obiect grafic de tip `figure`. În acest scop se utilizează instrucţiunea:

```
subplot(m,n,p)
```

care realizează împărţirea ferestrei grafice curente într-o matrice cu `m` rânduri şi `n` coloane de obiecte grafice de tip `axes` și selectarea obiectului grafic `axes` cu identificatorul numeric `p` ca fiind cel curent.

4.13. INSTRUCŢIUNEA `AXIS`

Controlul principalelor proprietăţi ale obiectului grafic `axes` se poate face şi cu instrucţiuni specifice de tip `axis`. Astfel, principalele moduri de configurare pentru aplicarea instrucţiunii `axis` sunt:

- `axis auto`. Varianta implicită pentru care limitele celor două axe se determină în mod automat pe baza valorilor minime şi maxime ale datelor x şi y.
- `axis tight`. Limitele celor două axe sunt strict egale cu limitele valorilor `x` şi `y`.
- `axis off`. Anularea afișării obiectului grafic `axes`.
- `axis xy`. Originea sistemului de coordonate este amplasată în colţul din stânga-jos, axa `x` este orizontală având sensul crescător al valorilor de la stânga la dreapta, axa `y` este verticală având sensul crescător al valorilor de jos în sus.
- `axis ij`. Originea sistemului de coordonate este amplasată în colțul stânga-sus, axa `i` este verticală având sensul crescător al valorilor de sus în jos, axa `j` este orizontală având sensul crescător al valorilor de la stânga la dreapta.

- `axis square`. Redimensionarea lungimii axelor `x` şi `y` astfel încât suprafața obiectului grafic `axes` să aibă forma unui pătrat.
- `axis equal`. Redimensionarea lungimii axelor astfel încât unitatea de măsură în direcţia axei `x` să fie egală cu unitatea de măsură din direcţia axei `y`.
- `axis image`. Redimensionarea lungimii axelor astfel încât unitatea de măsură în direcţia axei `x` să fie egală cu unitatea de măsură din direcţia axei `y` şi în plus limitele celor două axe sunt strict egale cu limitele valorilor `x` şi `y`.

4.14. UTILIZAREA CARACTERELOR SPECIALE

Limbajul de programare MATLAB permite aplicarea în cadrul obiectelor grafice de tip `axes` a mai multor tipuri de adnotări. Adnotările care au ca obiect manipularea şirurilor de caractere sunt:

1. Adnotarea de etichetare a axelor:
```
xlabel('valoare proprietate')
ylabel('valoare proprietate')
zlabel('valoare proprietate')
```
2. Adnotării de tip titlu:
```
title('valoare proprietate')
```
3. Adnotarea de tip legendă:
```
legend('pic1','pic2',...)
```
în care `pic1` şi `pic2` reprezintă parametrii de identificare ai curbelor 1, 2, etc.
4. Adnotarea de tip text:
```
text(xt,yt,'text')
```
în care `xt` și `yt` sunt coordonatele de amplasare a textului, iar `'text'` reprezintă textul propriu-zis.

Pentru toate aceste tipuri de adnotări se pot utiliza caractere speciale. Principalele tipuri de caractere speciale sunt: indicii inferiori, indicii superiori, simbolurilor matematice şi literele greceşti.

Obţinerea indicelui inferior simplu se realizează prin utilizarea caracterului special „_” care are rolul de a plasa caracterul imediat următor pe poziţia de indice inferior. În cazul în care se doreşte obţinerea poziţiei de indice superior simplu se va folosi caracterul special „^”. În cazul în care pe poziţia de indice inferior, respectiv superior se doreşte plasarea mai multor caractere (indice inferior şi superior de tip complex) se impune includerea acestora între acolade.

Introducerea simbolurilor matematice şi a caracterelor greceşti se face prin utilizarea unui interpretor de caractere speciale. Sunt implementate două seturi de caractere speciale: TEX şi LATEX. Selectarea interpretorului de caractere speciale se face prin modificarea proprietății `Interpreter` la una din valorile particulare `tex`, `latex`, respectiv `none`. Interpretorul implicit este TEX.

Spre exemplu, introducerea unui text cu folosirea interpretorului de tip LATEX se face printr-o instrucţiune de forma:
```
text('Interpreter','latex',……)
```

Modul de obținere a caracterelor greceşti este exemplificat în tabelul:

Simbol	Sintaxa	Simbol	Sintaxa	Simbol	Sintaxa
α	\alpha	ν	\nu	Γ	\Gamma
β	\beta	ξ	\xi	Δ	\Delta
γ	\gamma	ο	o	Θ	\Theta
δ	\delta	π	\pi	Λ	\Lambda
ε	\epsilon	ρ	\rho	Ξ	\Xi
ζ	\zeta	σ	\sigma	Π	\Pi
η	\eta	τ	\tau	Σ	\Sigma
θ	\theta	υ	\upsilon	Υ	\Upsilon
ι	\iota	φ	\phi	Φ	\Phi
κ	\kappa	χ	\chi	Ψ	\Psi
λ	\lambda	ψ	\psi	Ω	\Omega
μ	\mu	ω	\omega		

Modul de obţinere a simbolurilor matematice este exemplificat în tabelul:

Simbol	Sintaxa	Simbol	Sintaxa	Simbol	Sintaxa
≡	\equiv	∩	\cap	∞	\infty
≅	\cong	∪	\cup	∂	\partial
≈	\approx	⊃	\supset	°	\circ
~	\sim	⊇	\supseteq	∇	\nabla
≠	\neq	⊂	\subset	∅	\oslash
≤	\leq	⊆	\subseteq	∫	\int
≥	\geq	∈	\in	ℜ	\Re
±	\pm	⊥	\perp	ℑ	\Im
÷	\div	⊗	\otimes	ℵ	\aleph
·	\cdot	⊕	\oplus	℘	\wp

4.15. PROBLEME

Problema 4.1

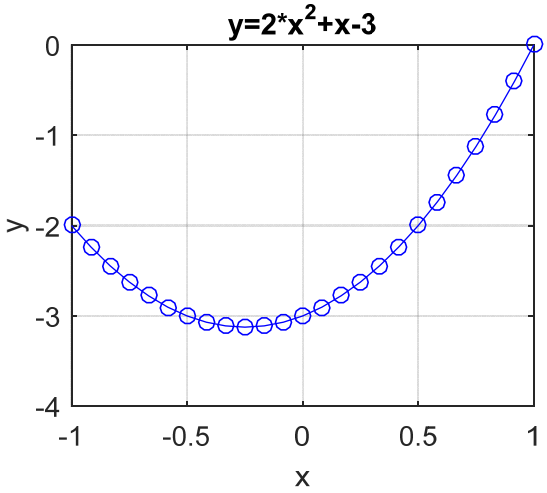
Se consideră funcţia: $f:[-1,1] \to \mathbb{R}$, definită prin: $y = 2x^2 + x - 3$.

a) Să se reprezinte grafic în coordonate carteziene funcţia $y = f(x)$, figura 4.1.

b) Să se aplice parametrii de formatare corespunzători diferitelor tipuri de adnotări (axe, grid, titlu).

c) Să se verifice rezultatul aplicării diferitelor combinaţii de parametri de formatare ai reprezentării grafice (tip linie, culoare, marker).

```
%% DATE DE INTRARE
% 1. Domeniul de definitie
xmin=-1;xmax=1;nx=25;
x=linspace(xmin,xmax,nx);
% 2. Calculul valorilor functiei
y=2*x.^2+x-3;
%% REPREZENTARE GRAFICA
figure
plot(x,y,'-ob');
grid on;
xlabel('x');
ylabel('y');
title('y=2*x^2+x-3');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 4.1. Reprezentarea grafică a funcției $y = 2x^2 + x - 3$.

Problema 4.2

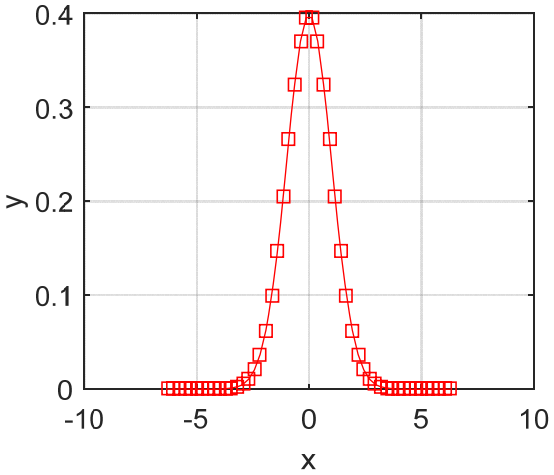
Se consideră funcţia: $f:[-2\pi, +2\pi], \to \mathbb{R}$, definită prin: $y = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

a) Să se reprezinte grafic în coordonate carteziene funcţia $y = f(x)$, figura 4.2.

b) Să se aplice parametrii de formatare corespunzători diferitelor tipuri de adnotări (axe, grid, titlu).

c) Să se verifice rezultatul aplicării diferitelor combinaţii de parametri de formatare ai reprezentării grafice (tip linie, culoare, marker).

```
%% DATE DE INTRARE
% 1. Domeniul de definitie
xmin=-2*pi;xmax=2*pi;nx=50;
x=linspace(xmin,xmax,nx);
% 2. Calculul valorilor functiei
y=1/sqrt(2*pi)*exp(-x.^2/2);
%% REPREZENTARE GRAFICA
figure
plot(x,y,'-sr');grid on;
xlabel('x');ylabel('y');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 4.2. Reprezentarea grafică a funcției $y = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$.

Problema 4.3

Se consideră funcţiile:

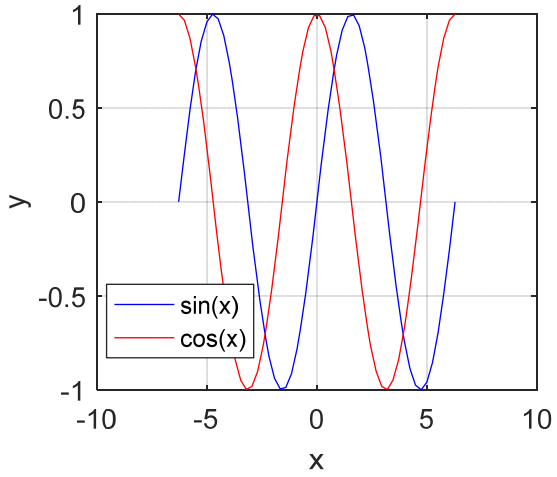
$$f_1: [-2\pi, 2\pi] \to \mathbb{R},\ f_1(x) = \sin x.$$
$$f_2: [-2\pi, 2\pi] \to \mathbb{R},\ f_2(x) = \cos x$$

Să se reprezinte grafic cele două funcţii:

- în două ferestre grafice diferite
- în aceeaşi fereastră grafică şi în aceleași axe, figura 4.3.

```
%% DATE DE INTRARE
% 1. Domeniul de definitie
xmin=-2*pi;xmax=2*pi;nx=50;
x=linspace(xmin,xmax,nx);
% 2. Calculul functiei sinus
y1=sin(x);
% 2. Calculul functiei cosinus
y2=cos(x);
%% REPREZENTARE GRAFICA
figure
plot(x,y1,'-b');hold on;
plot(x,y2,'-r');hold off;
grid on;xlabel('x');ylabel('y');
legend('sin(x)','cos(x)');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.3. Reprezentarea grafică a funcțiilor sinus și cosinus.

Problema 4.4

Se consideră funcția:

$$f: [-1, +1] \to \mathbb{R},\ f(x) = 2x + 1$$

Să se calculeze:

$$g(x) = f(x)/2 + 3,\ h(x) = [f(x)]^{3/2}$$
$$u(x) = \frac{2f(x)+1}{3f(x)+2} + 1$$
$$v(x) = 1 + 2[f(x)]^2 + 3[g(x)]^3$$
$$w(x) = f(x) \cdot g(x) + [3 + u(x)]^{-1}$$

Să se reprezinte grafic funcțiile:

$$f(x),\ g(x),\ h(x),\ u(x),\ v(x),\ w(x)$$

Problema 4.5

Se consideră funcția:

$$f: [-2\pi, +2\pi] \to \mathbb{R},\ f(\theta) = \sin\theta$$

Să se calculeze:

$$g(\theta) = [1 - f(\theta)]^2,\ h(\theta) = \theta \cdot f(\theta) + \theta \cdot \left(\theta + \frac{\pi}{2}\right)$$
$$u(\theta) = \frac{\theta}{2 + f(\theta)} + \frac{3\pi}{2}$$
$$v(\theta) = \frac{\pi}{2} f(\theta) - \frac{\pi}{3}[f(\theta) + \theta^2 \cdot f(\theta)]$$

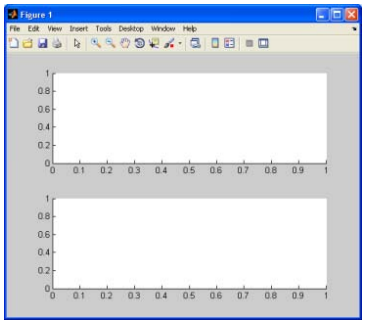
$$w(\theta) = f(\theta) \cdot [1 + g(\theta)] + \frac{u(\theta)}{v(\theta)}$$

Să se reprezinte grafic funcțiile:

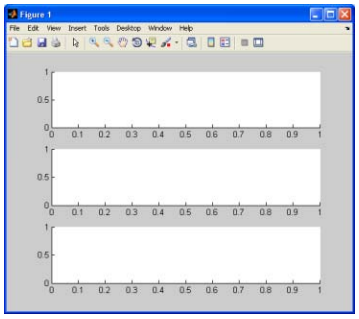
$$f(\theta),\ g(\theta),\ h(\theta),\ u(\theta),\ v(\theta),\ w(\theta)$$

Problema 4.6

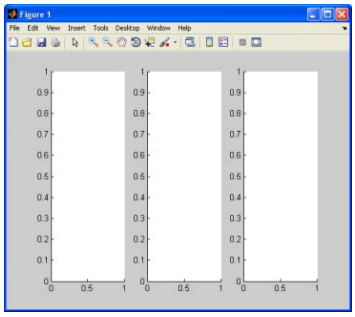
Utilizând instrucțiunea `subplot`, să se construiască structurile de axe prezentate în figura 4.4.



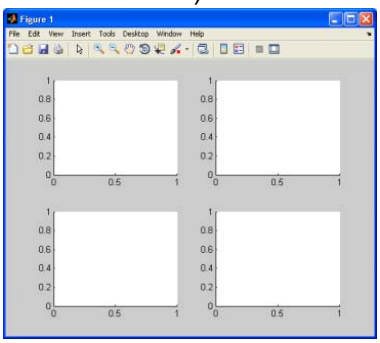
a)



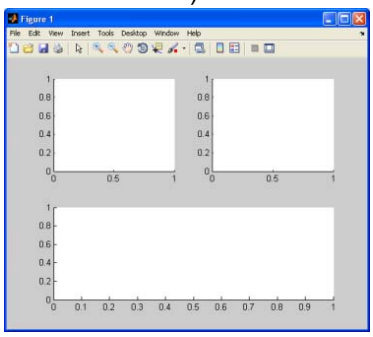
b)



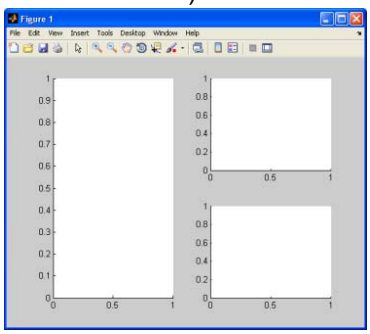
c)



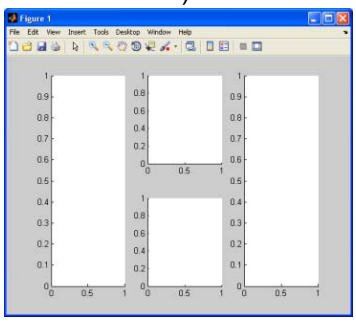
d)



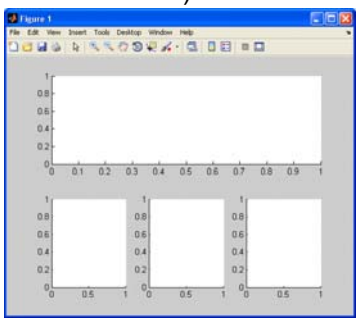
e)



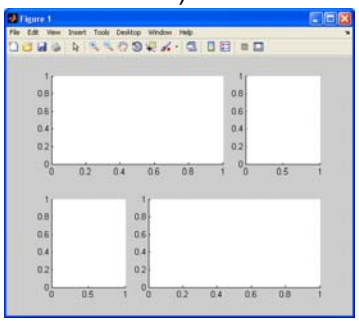
f)



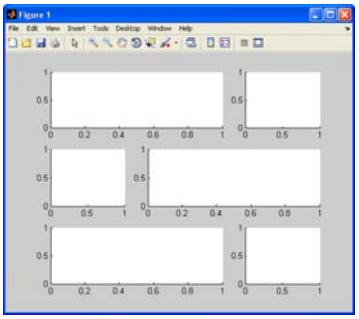
g)



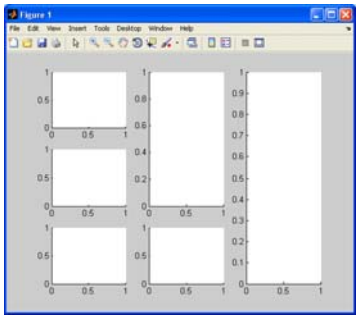
h)



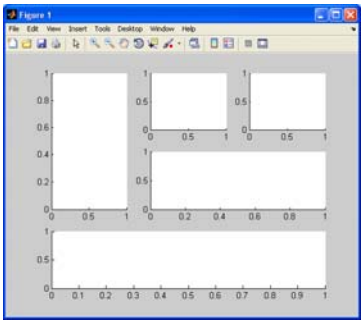
i)



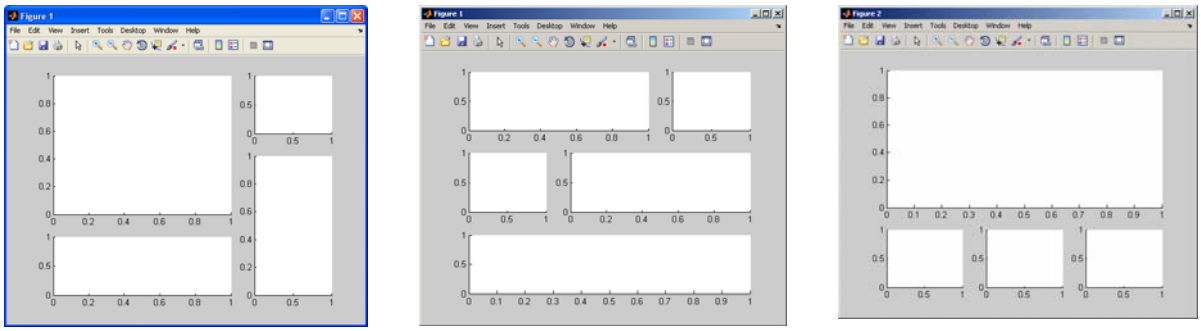
j)



k)



l)



m) n) o)

Figura 4.4. Structuri de axe obținute cu instrucțiunea `subplot`.

Problema 4.7

Să se reprezinte grafic și să se analizeze comparativ rezultatele obținute cu aplicarea instrucțiunilor `plot` (ferestre grafice diferite, aceeași fereastră grafică și axe diferite, aceeași fereastră grafică și aceleași axe) și `plotyy` pentru următoarele funcții:

a)

$$f_1: [-2\pi, 2\pi] \to \mathbb{R},\ f_1(x) = x \sin 10x.$$

$$f_2: [-2\pi, 2\pi] \to \mathbb{R},\ f_2(x) = 1000 \sin(x/2)$$

b)

$$s = 2.5, m = 0$$

$$f_1: [-3s, 3s] \to \mathbb{R},\ f_1(x) = x^{0.4} \sin 10x.$$

$$f_2: [-3s, 3s] \to \mathbb{R},\ f_2(x) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m}{s}\right)^2}$$

c)

$$f_1: [-4\pi, 4\pi] \to \mathbb{R},\ f_1(x) = x^{0.4} \sin 10x.$$

$$f_2: [-4\pi, 4\pi] \to \mathbb{R},\ f_2(x) = 100 \sin x$$

Problema 4.8

Se consideră funcția: $f: [0,4\pi] \to \mathbb{R}$, definită prin:

$$f(x) = \sin x.$$

Să se calculeze:

$$g(x) = a \cdot x \cdot f(x),\ a = \pi/2.$$

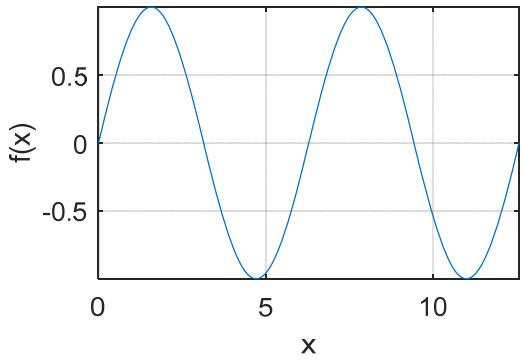
Să se reprezinte grafic și să se analizeze comparativ funcțiile $f(x)$ și $g(x)$.

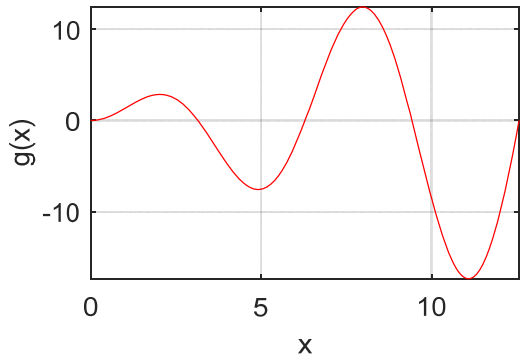
Cele două funcții se vor reprezenta:

- În ferestre grafice diferite, figura 4.5, b).
- În aceeași fereastră grafică și în aceleași axe, figura 4.5, c).
- În aceeași fereastră grafică și în axe diferite, axe suprapuse (figura 4.5, d) și axe alăturate (figura 4.5, e).

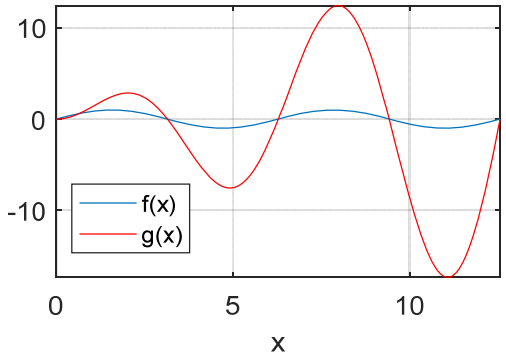
```
%% DATE DE INTRARE
xmin=0;xmax=4*pi;nx=500;
x=linspace(xmin,xmax,nx);
a=pi/2;
%% DEFINIREA FUNCTIILOR
f=sin(x);
g=a*x.*f;
%% FERESTRE GRAFICE DIFERITE
figure
plot(x,f);grid on;
xlabel('x');ylabel('f(x)');
figure
plot(x,g,'r');
grid on;
xlabel('x');ylabel('g(x)');
%% ACELEASI AXE
figure
plot(x,f);hold on;
plot(x,g,'r');hold off;
grid on;xlabel('x');
legend('f(x)','g(x)');
%% AXE SUPRAPUSE
figure
subplot(2,1,1);plot(x,f);
grid on;
xlabel('x');ylabel('f(x)');
subplot(2,1,2);plot(x,g,'r');
grid on;
xlabel('x');ylabel('g(x)');
%% AXE ALATURATE
figure
subplot(1,2,1);plot(x,f);
grid on;
xlabel('x');ylabel('f(x)');
subplot(1,2,2);plot(x,g,'r');
grid on;
xlabel('x');ylabel('g(x)');
```

a) fișierul `script`

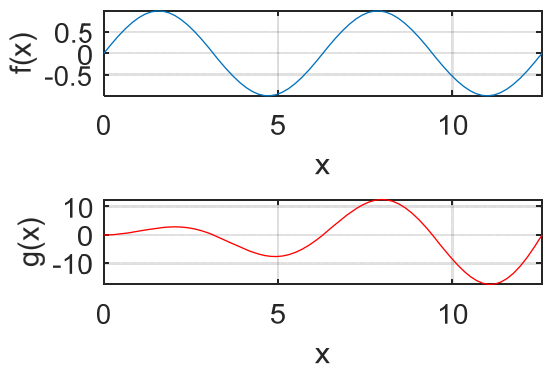




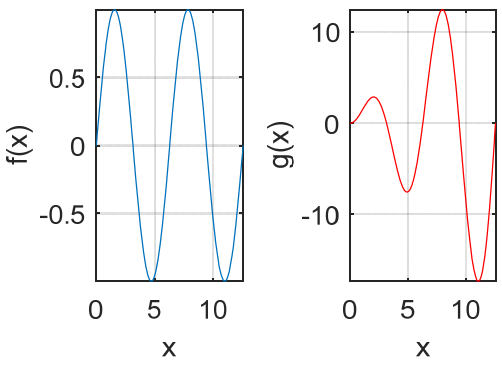
b) ferestre grafice diferite



c) aceeași fereastră grafică, aceleași axe



d) aceeași fereastră grafică, axe suprapuse



e) aceeași fereastră grafică, axe alăturate

Figura 4.5. Reprezentări grafice comparative.

Problema 4.9

Se consideră funcția: $f: [0,4\pi] \to \mathbb{R}$, definită prin:

$$f(x) = \sin x.$$

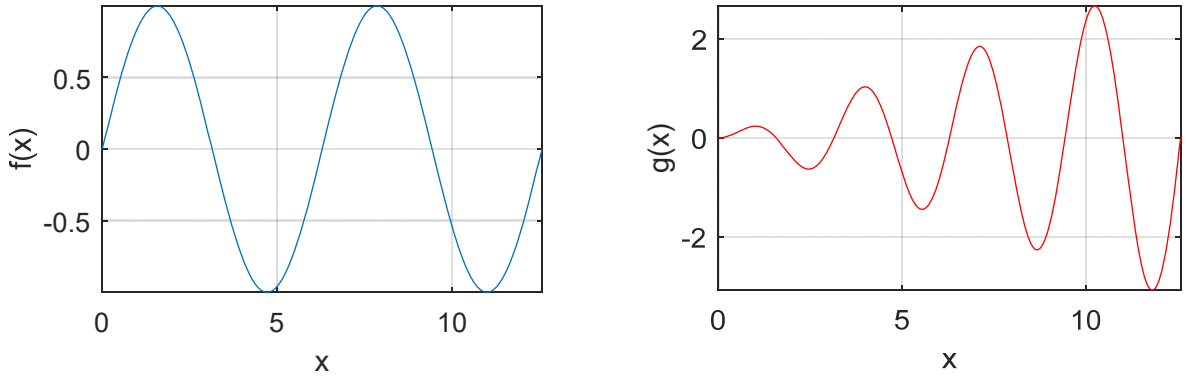
Să se calculeze:

$$g(x) = a \cdot x \cdot \cos x \cdot f(x),\ a = \pi/6.$$

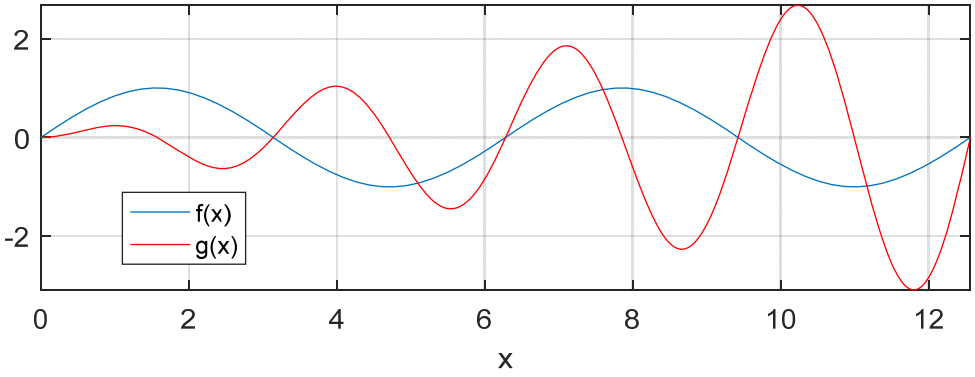
Să se reprezinte grafic și să se analizeze comparativ funcțiile $f(x)$ și $g(x)$.

Cele două funcții se vor reprezenta:

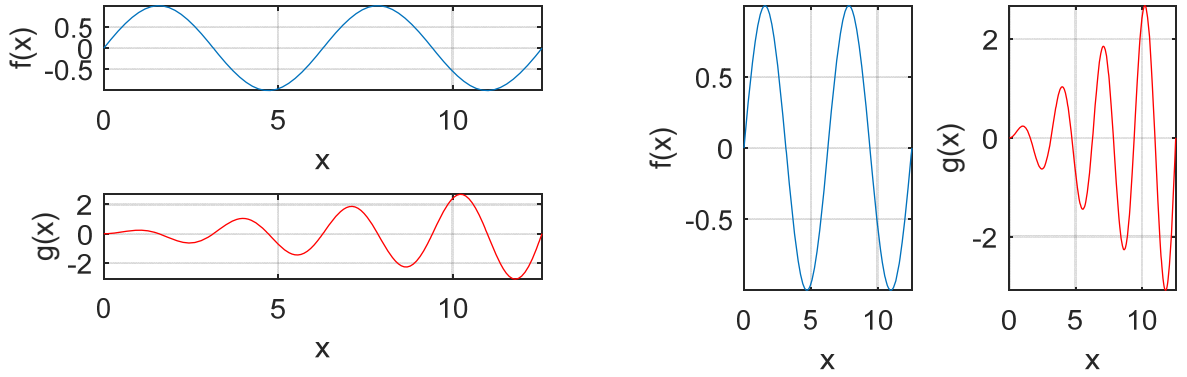
- În ferestre grafice diferite, figura 4.6, a).
- În aceeași fereastră grafică și în aceleași axe, figura 4.6, b).
- În aceeași fereastră grafică și în axe diferite, axe suprapuse (figura 4.6, c) și axe alăturate (figura 4.6, d).



a) ferestre grafice diferite



b) aceeași fereastră grafică, aceleași axe



c) aceeași fereastră grafică, axe suprapuse

d) aceeași fereastră grafică, axe alăturate

Figura 4.6. Reprezentări grafice comparative.

Problema 4.10

Se consideră funcția: $f:[0,4\pi] \to \mathbb{R}$, definită prin:

$$f(x) = \sin x.$$

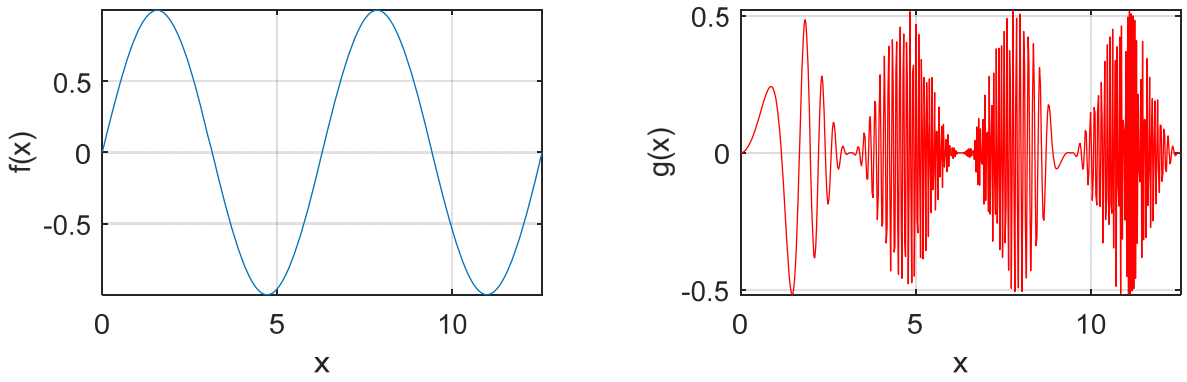
Să se calculeze:

$$g(x) = a \cdot \sin x \cdot \cos x^3 \cdot f(x),\ a = \pi/6.$$

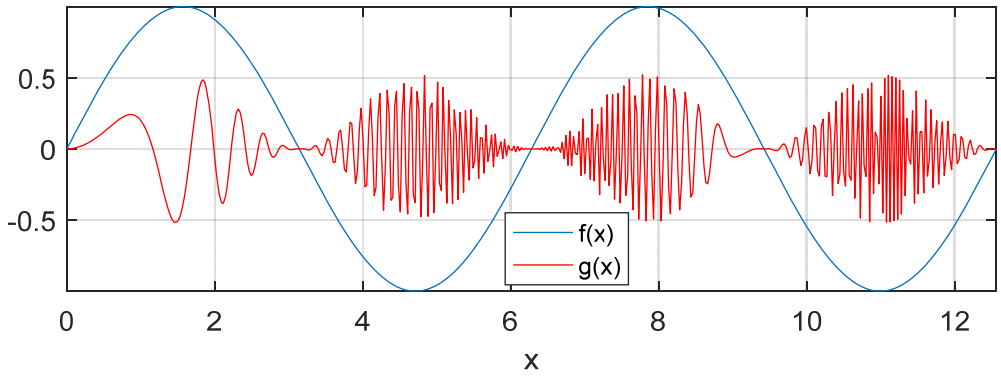
Să se reprezinte grafic și să se analizeze comparativ funcțiile $f(x)$ și $g(x)$.

Cele două funcții se vor reprezenta:

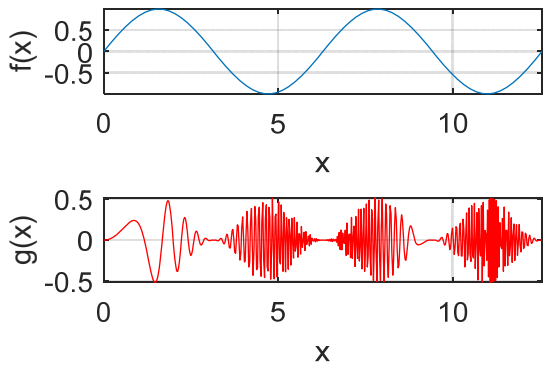
- În ferestre grafice diferite, figura 4.7, a).
- În aceeași fereastră grafică și în aceleași axe, figura 4.7, b).
- În aceeași fereastră grafică și în axe diferite, axe suprapuse (figura 4.7, c) și axe alăturate (figura 4.7, d).



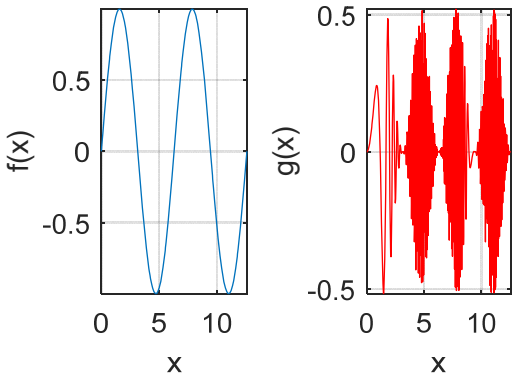
a) ferestre grafice diferite



b) aceeași fereastră grafică, aceleași axe



c) aceeași fereastră grafică, axe suprapuse



d) aceeași fereastră grafică, axe alăturate

Figura 4.7. Reprezentări grafice comparative.

Problema 4.11

Se consideră funcția: $f: [0,4\pi] \to \mathbb{R}$, definită prin:

$$f(x) = \sin x.$$

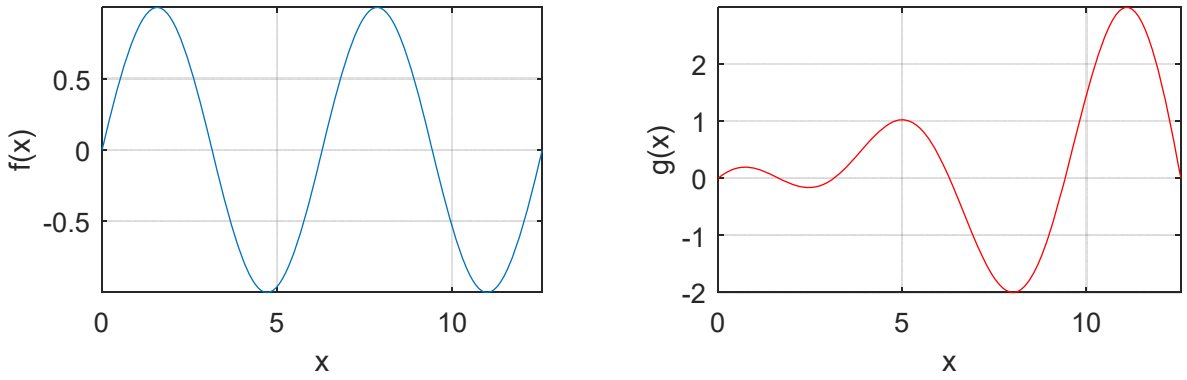
Să se calculeze:

$$g(x) = a \cdot f(x) - \frac{x}{6a} \cdot \sin x,\ a = \pi/6.$$

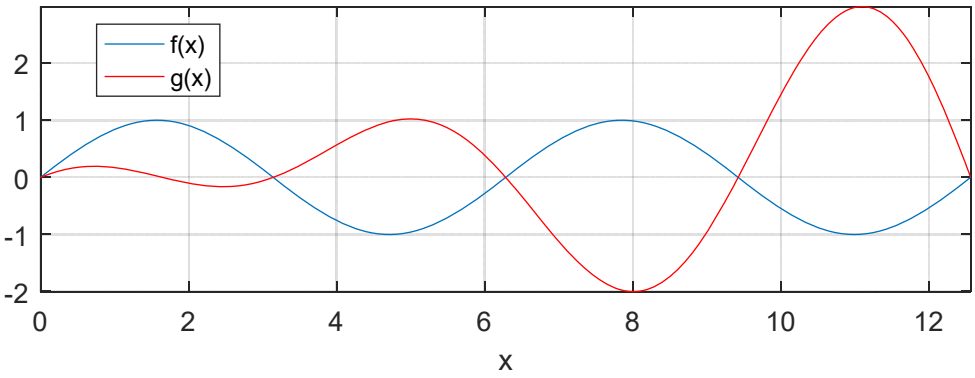
Să se reprezinte grafic și să se analizeze comparativ funcțiile $f(x)$ și $g(x)$.

Cele două funcții se vor reprezenta:

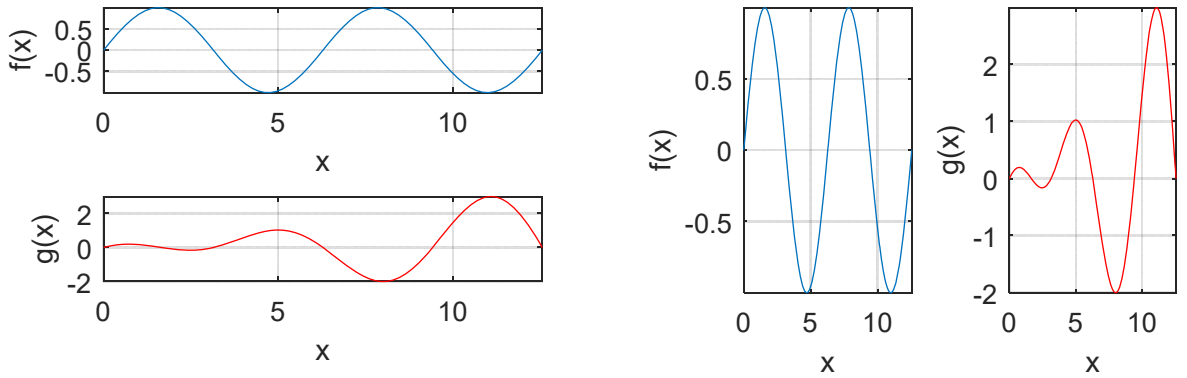
- În ferestre grafice diferite, figura 4.8, a).
- În aceeași fereastră grafică și în aceleași axe, figura 4.8, b).
- În aceeași fereastră grafică și în axe diferite, axe suprapuse (figura 4.8, c) și axe alăturate (figura 4.8, d).



a) ferestre grafice diferite



b) aceeași fereastră grafică, aceleași axe



c) aceeași fereastră grafică, axe suprapuse

d) aceeași fereastră grafică, axe alăturate

Figura 4.8. Reprezentări grafice comparative.

Problema 4.12

Să se reprezinte grafic următoarele funcţii definite parametric:

- în patru ferestre grafice diferite
- în aceeaşi fereastră grafică cu o structură de axe în diferite configuraţii (1,4); (4,1); (2,2); etc.

a) Cicloida, figura 4.9:

$$\begin{cases} x = at - h\sin t \\ y = a - h\cos t \end{cases}$$
$$a = 1,\ h = 1,\ t = [-2\pi, 2\pi]$$

b) Astroida, figura 4.10:

$$\begin{cases} x = a(\cos t)^3 \\ y = a(\sin t)^3 \end{cases}$$
$$a = 1,\ t = [-\pi, +\pi]$$

c) Epicicloida, figura 4.11:

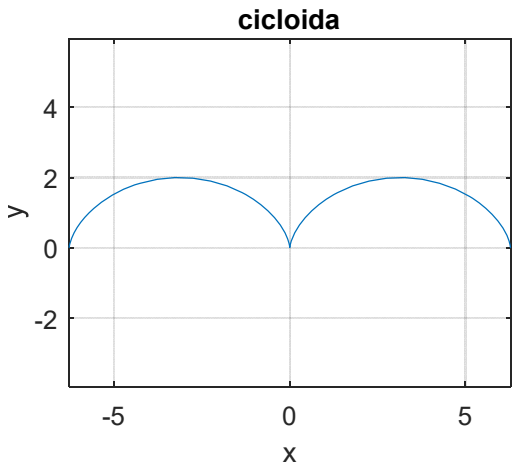
$$\begin{cases} x = (a+b)\cos t - b\cos\left[\left(\frac{a}{b}+1\right)t\right] \\ y = (a+b)\sin t - b\sin\left[\left(\frac{a}{b}+1\right)t\right] \end{cases}$$
$$a = 8,\ b = 5,\ t = [0, +10\pi]$$

d) Epitrohoida, figura 4.12:

$$\begin{cases} x = (a+b)\cos t - c\cos\left[\left(\frac{a}{b}+1\right)t\right] \\ y = (a+b)\sin t - c\sin\left[\left(\frac{a}{b}+1\right)t\right] \end{cases}$$
$$a = 5,\ b = 3,\ c = 5,\ t = [0, +6\pi]$$

```
%% CICLOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-2*pi;tmax=2*pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;h=1;
% 3. Definirea functiei
x=a*t-h*sin(t);
y=a-h*cos(t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('cicloida');
```

a) fișierul `script`

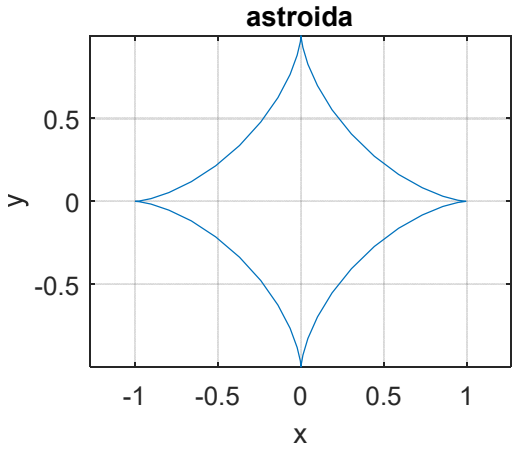


b) reprezentarea grafică obținută

Figura 4.9. Reprezentarea grafică a cicloidei.

```
%% ASTROIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-pi;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
x=a*(cos(t)).^3;
y=a*(sin(t)).^3;
%% REPREZENTARE GRAFICA
figure
plot(x,y);
grid on;
axis equal;
xlabel('x');
ylabel('y');
title('astroida');
```

a) fișierul `script`

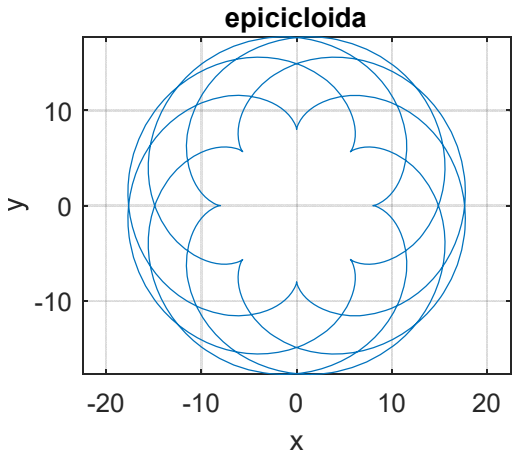


b) reprezentarea grafică obținută

Figura 4.10. Reprezentarea grafică a astroidei.

```
%% EPICICLOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=10*pi;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=8;b=5;
% 3. Definirea functiei
x=(a+b)*cos(t)-b*cos((a/b+1)*t);
y=(a+b)*sin(t)-b*sin((a/b+1)*t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);
grid on;
axis equal;
xlabel('x');
ylabel('y');
title('epicicloida');
```

a) fișierul `script`

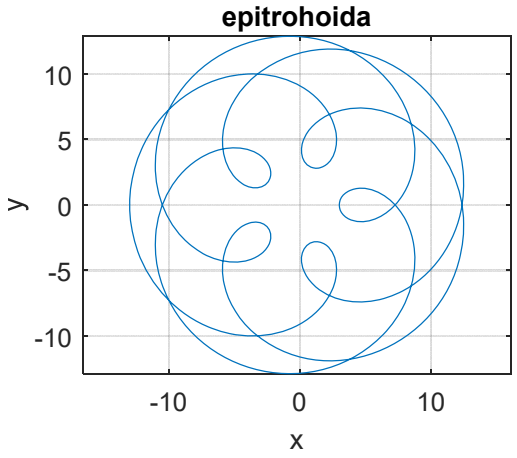


b) reprezentarea grafică obținută

Figura 4.11. Reprezentarea grafică a epicicloidei.

```
%% EPITROHOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=6*pi;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=5;b=3;c=5;
% 3. Definirea functiei
x=(a+b)*cos(t)-c*cos((a/b+1)*t);
y=(a+b)*sin(t)-c*sin((a/b+1)*t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('epitrohoida');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.12. Reprezentarea grafică a epitrohoidei.

Problema 4.13

Să se reprezinte grafic următoarele funcţii definite parametric:

- în cinci ferestre grafice diferite
- în aceeaşi fereastră grafică cu o structură de axe în diferite configuraţii (1,5); (5,1); (2,3); etc.

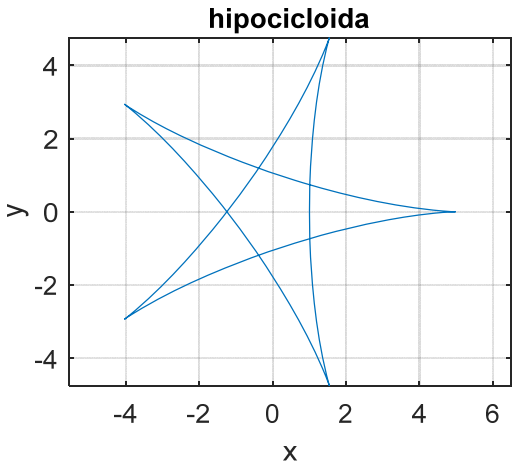
a) Hipocicloida, figura 4.13:

$$\begin{cases} x = (a-b)\cos t + b\cos\left[\left(\frac{a}{b}-1\right)t\right] \\ y = (a-b)\sin t - b\sin\left[\left(\frac{a}{b}-1\right)t\right] \end{cases}$$

$$a = 5,\ b = 3,\ t = [-3\pi\ ,3\pi]$$

```
%% HIPOCICLOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-3*pi;tmax=3*pi;nt=100;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=5;b=3;
% 3. Definirea functiei
x=(a-b)*cos(t)+b*cos((a/b-1)*t);
y=(a-b)*sin(t)-b*sin((a/b-1)*t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('hipocicloida');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.13. Reprezentarea grafică a hipocicloidei.

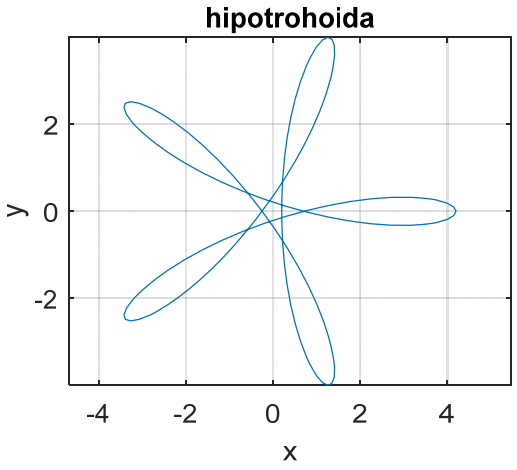
b) Hipotrohoida, figura 4.14:

$$\begin{cases} x = (a-b)\cos t + c\cos\left[\left(\frac{a}{b}-1\right)t\right] \\ y = (a-b)\sin t - c\sin\left[\left(\frac{a}{b}-1\right)t\right] \end{cases}$$

$$a = 5,\ b = 7,\ c = 2.2,\ t = [0, +6\pi]$$

```
%% HIPOTROHOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=6*pi;nt=100;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=5;b=3;c=2.2;
% 3. Definirea functiei
x=(a-b)*cos(t)+c*cos((a/b-1)*t);
y=(a-b)*sin(t)-c*sin((a/b-1)*t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('hipotrohoida');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.14. Reprezentarea grafică a hipotrohoidei.

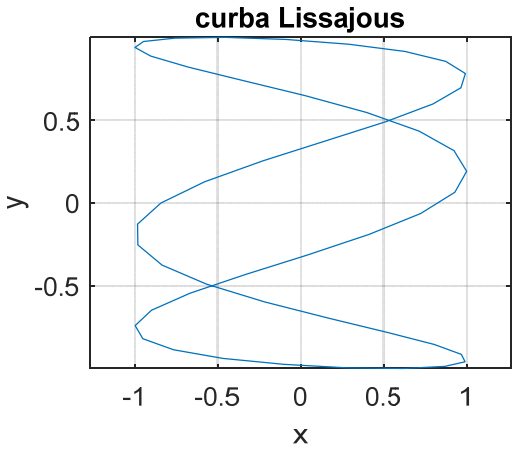
c) Curba Lissajous, figura 4.15:

$$\begin{cases} x = a\sin(nt + c) \\ y = b\sin t \end{cases}$$

$$a = 1,\ b = 1,\ c = 1, n = 3,\ t = [-\pi, +\pi]$$

```
%% CURBA LISSAJOUS
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-pi;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;b=1;c=1;n=3;
% 3. Definirea functiei
x=a*sin(n*t+c);
y=b*sin(t);
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('curba Lissajous');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.15. Reprezentarea grafică a curbei Lissajous.

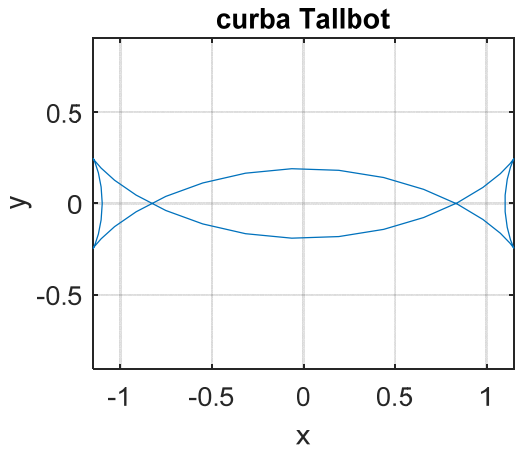
d) Curba Tallbot, figura 4.16:

$$\begin{cases} x = [a^2 + f^2(\sin t)^2]\dfrac{\cos t}{a} \\ y = [a^2 - 2f^2 + f^2(\sin t)^2]\dfrac{\sin t}{b} \end{cases}$$

$$a = 1.1,\ b = 0.5,\ f = 1,\ t = [-\pi, +\pi]$$

```
%% CURBA TALLBOT
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-pi;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1.1;b=0.5;f=1;
% 3. Definirea functiei
x=(a^2+f^2*(sin(t)).^2).*cos(t)/a;
y=(a^2-
2*f^2+f^2*(sin(t)).^2).*sin(t)/a;
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('curba Tallbot');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.16. Reprezentarea grafică a curbei Tallbot.

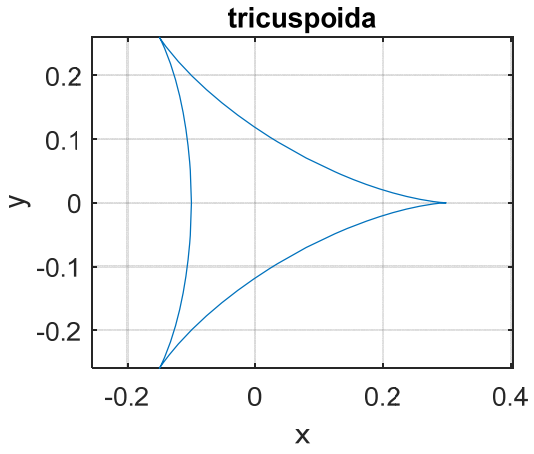
e) Tricuspoida, figura 4.17:

$$\begin{cases} x = a(2\cos t + \cos 2t) \\ y = a(2\sin t - \sin 2t) \end{cases}$$

$$a = 0.1,\ t = [-\pi, +\pi]$$

```
%% TRICUSPOIDA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-pi;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=0.1;
% 3. Definirea functiei
x=a*(2*cos(t)+cos(2*t));
y=a*(2*sin(t)-sin(2*t));
%% REPREZENTARE GRAFICA
figure
plot(x,y);grid on;axis equal;
xlabel('x');ylabel('y');
title('tricuspoida');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.17. Reprezentarea grafică a tricuspoidei.

Problema 4.14

Se consideră profilul aerodinamic de tip Gö 364 definit prin coordonatele intradosului şi extradosului conform tabelului de valori:

x	0	1.5	2.5	5	7.5	10	15	20	30	40	50	60	70	80	90	95	100
y^-	0.85	0	0.05	0.35	0.55	0.65	1.05	1.3	1.7	1.85	1.8	1.55	1.25	0.9	0.45	0.2	0.1
y^+	0.85	4.05	5.45	7.03	8.6	9.65	11	11.85	12.5	12.1	11.1	9.5	7.55	5.35	2.9	1.55	0.1

Să se reprezinte grafic profilul Gö 364 utilizând pentru intrados $y^-(x)$ linia continuă de culoare albastră şi pentru extrados $y^+(x)$ linia punctată de culoare roşie.

Să se verifice rezultatul aplicării tuturor parametrilor de configurare ai instrucțiunii `axis`, figura 4.18: `auto, tight, off, xy, ij, square, equal, image.`

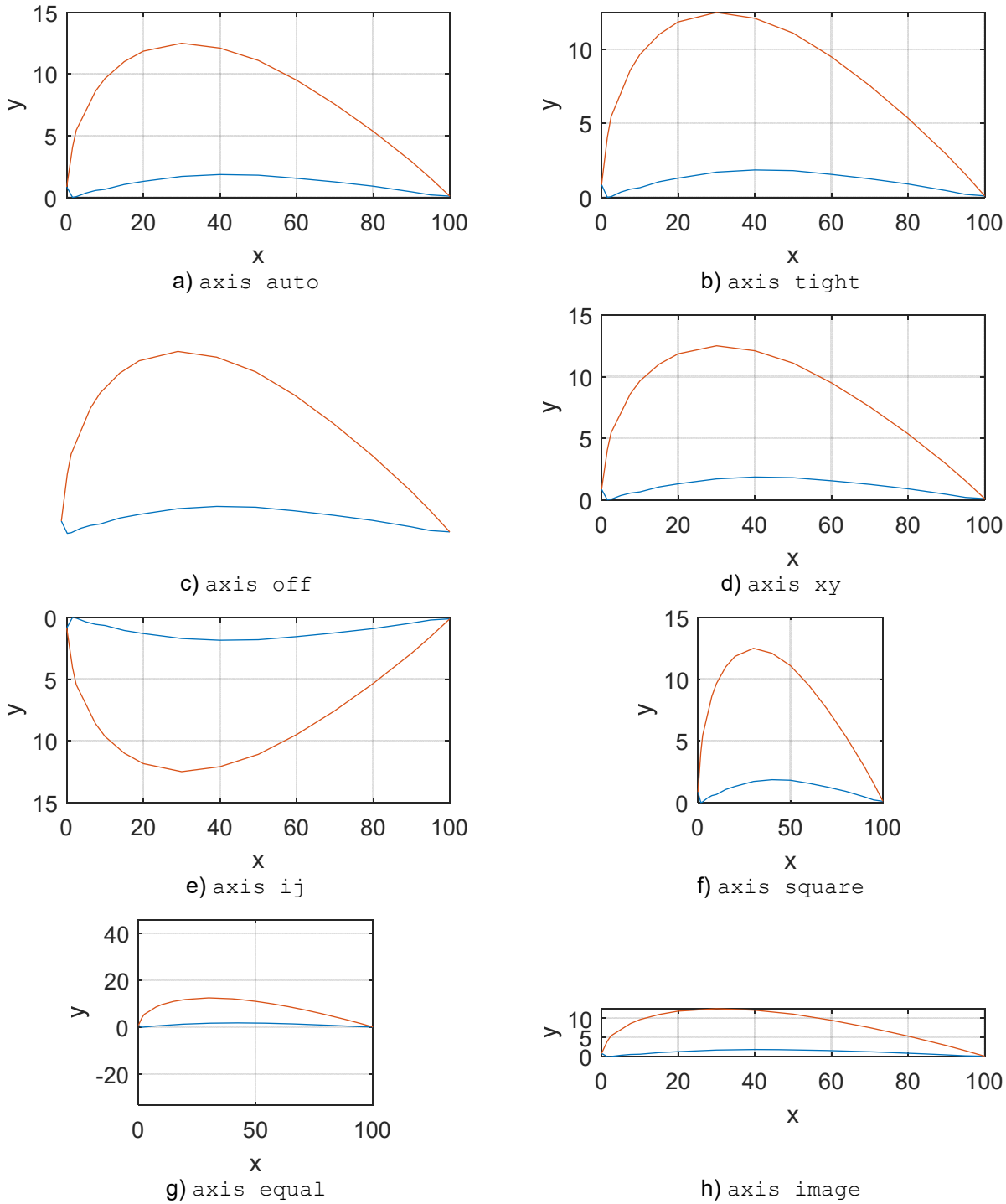


Figura 4.18. Instrucțiunea `axis`.

Problema 4.15

Să se reprezinte grafic următoarele funcţii definite în coordonate polare:

a) Spirala lui Arhimede, figura 4.19:

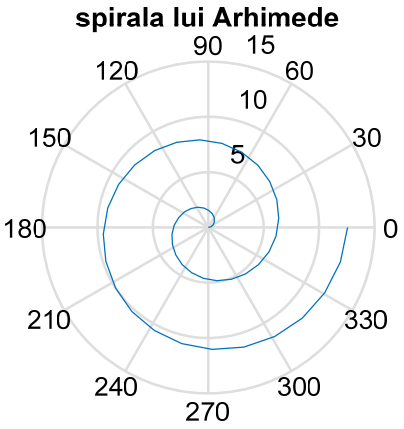
$$r = a\theta$$
$$a = 1,\ \theta = [0, +4\pi]$$

```
%% SPIRALA LUI ARHIMEDE
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=4*pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r=a*t;

%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('spirala lui Arhimede');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.19. Reprezentarea grafică a spiralei lui Arhimede.

b) Spirala logaritmică (echiunghiulară), figura 4.20:

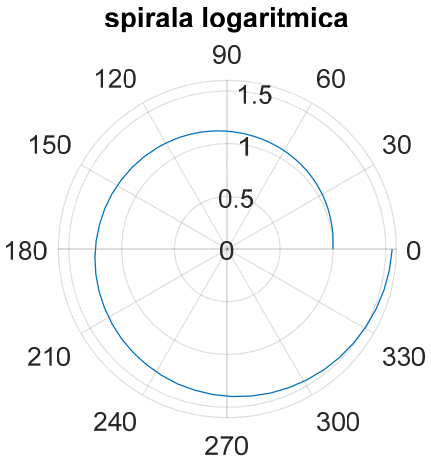
$$r = ae^{\theta\ \mathrm{ctg}\, b}$$
$$a = 1,\ b = 1.5\ \theta = [0, +2\pi]$$

```
%% SPIRALA LOGARITMICĂ
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=2*pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;b=1.5;
% 3. Definirea functiei
r=a*exp(t*cot(b));

%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('spirala logaritmica');
```

a) fișierul `script`



b) reprezentarea grafică obținută

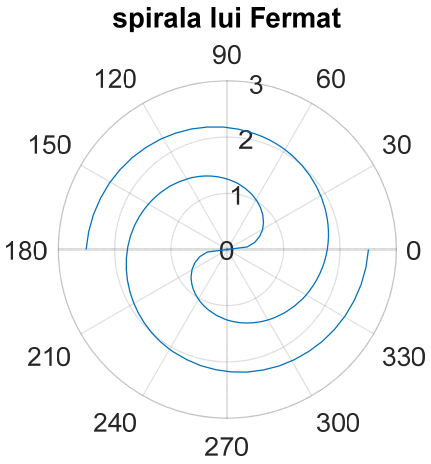
Figura 4.20. Reprezentarea grafică a spiralei logaritmice (echiunghiulare).

c) Spirala lui Fermat, figura 4.21:

$$r^2 = a^2\theta$$
$$a = 1,\ \theta = [0, +2\pi]$$

```
%% SPIRALA LUI FERMAT
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=2*pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r1=a*sqrt(t);
r2=-a*sqrt(t);
%% REPREZENTARE GRAFICA
figure
h1=polarplot(t,r1);hold on;
h2=polarplot(t,r2);hold off;
h1.Color='[0 0.447 0.741]';
h2.Color='[0 0.447 0.741]';
title('spirala lui Fermat');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.21. Reprezentarea grafică a spiralei lui Fermat.

d) Spirala sinusoidală, figura 4.22:

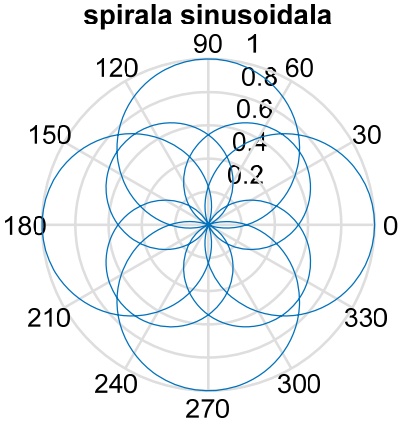
$$r = a(\cos p\theta)^{1/p}$$
$$a = 1,\ p = 0.8,\ \theta = [0, +10\pi]$$

```
%% SPIRALA SINUSOIDALA
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=10*pi;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;p=0.8;
% 3. Definirea functiei
r=a*(cos(p*t)).^(1/p);

%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('spirala sinusoidala');
```

a) fișierul `script`



b) reprezentarea grafică obținută

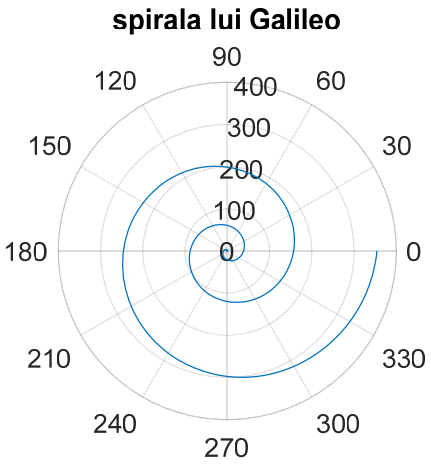
Figura 4.22. Reprezentarea grafică a spiralei sinusoidale.

e) Spirala lui Galileo, figura 4.23:

$$r = a\theta^2 - l$$
$$a = 1, l = 1, \theta = [0, +6\pi]$$

```
%% SPIRALA LUI GALILEO
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=6*pi;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;l=1;
% 3. Definirea functiei
r=a*t.^2-l;
%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('spirala lui Galileo');
```

a) fișierul `script`



b) reprezentarea grafică obținută

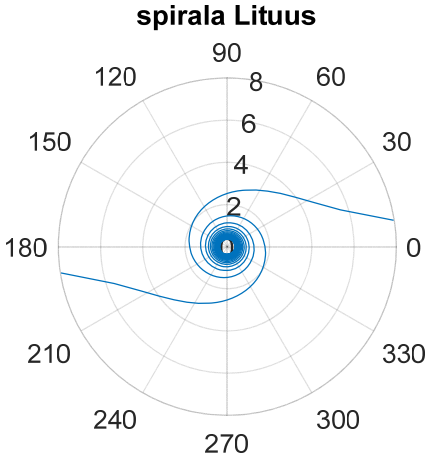
Figura 4.23. Reprezentarea grafică a spiralei lui Galileo.

f) Spirala Lituus, figura 4.24:

$$r^2 = \frac{a}{\theta}$$
$$a = 10, \theta = [0, +25\pi]$$

```
%% SPIRALA LITUUS
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=25*pi;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=10;
% 3. Definirea functiei
r1=sqrt(a./t);
r2=-sqrt(a./t);
%% REPREZENTARE GRAFICA
figure
h1=polarplot(t,r1);hold on;
h2=polarplot(t,r2);hold off;
h1.Color='[0 0.447 0.741]';
h2.Color='[0 0.447 0.741]';
title('spirala Lituus');
```

a) fișierul `script`



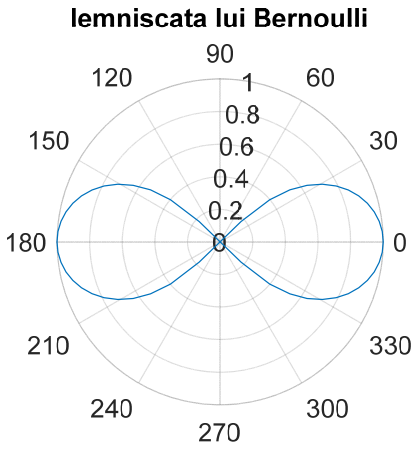
b) reprezentarea grafică obținută

Figura 4.24. Reprezentarea grafică a spiralei Lituus.

g) Lemniscata lui Bernoulli, figura 4.25:

$$r^2 = a^2 \cos 2\theta$$
$$a = 1,\ \theta = [0, +\pi]$$

```
%% LEMNISCATA LUI BERNOULLI
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r1=a*sqrt(cos(2*t));
r2=-a*sqrt(cos(2*t));
%% REPREZENTARE GRAFICA
figure
h1=polarplot(t,r1);hold on;
h2=polarplot(t,r2);hold off;
h1.Color='[0 0.447 0.741]';
h2.Color='[0 0.447 0.741]';
title('lemniscata lui Bernoulli');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 4.25. Reprezentarea grafică a lemniscatei lui Bernoulli.

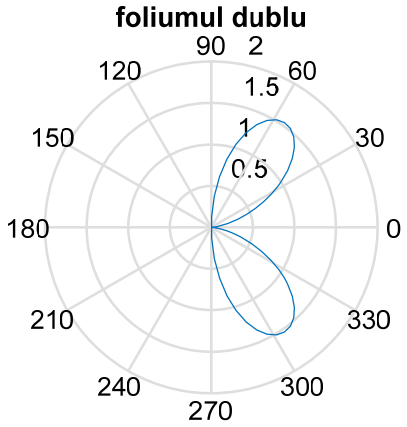
h) Foliumul dublu, figura 4.26:

$$r = 4a \cos\theta\ (\sin\theta)^2$$
$$a = 1,\ \theta = [0, +\pi]$$

```
%% FOLIUMUL DUBLU
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r=4*a*cos(t).*(sin(t)).^2;

%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('foliumul dublu');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 4.26. Reprezentarea grafică a foliumului dublu.

i) Foliumul triplu, figura 4.27:

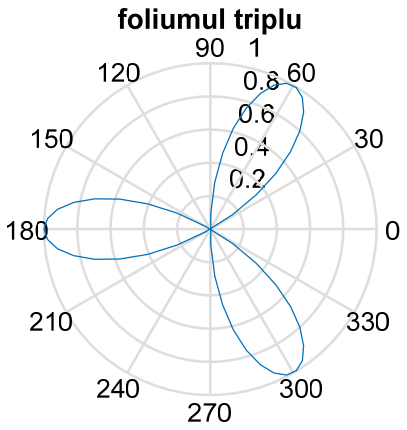
$$r = a\cos\theta\,[4(\sin\theta)^2 - 1]$$
$$a = 1,\ \theta = [0, +\pi]$$

```
%% FOLIUMUL TRIPLU
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;tmax=pi;nt=50;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r=a*cos(t).*(4*(sin(t)).^2-1);

%% REPREZENTARE GRAFICA
figure
polarplot(t,r);
title('foliumul triplu');
```

a) fișierul `script`



b) reprezentarea grafică obținută

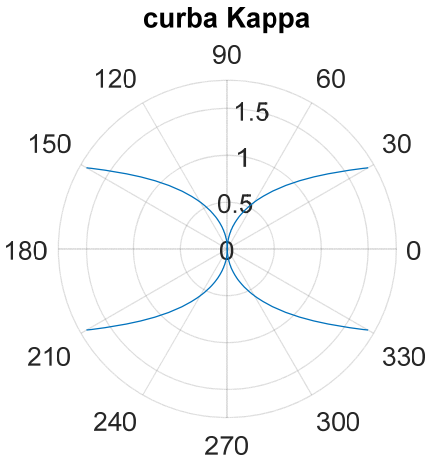
Figura 4.27. Reprezentarea grafică a foliumului triplu.

j) Curba Kappa, figura 4.28:

$$r = \pm a\,\mathrm{ctg}\,\theta$$
$$a = 1,\ \theta = [\pi/6, 5\pi/6]$$

```
%% CURBA KAPPA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=pi/6;tmax=5*pi/6;nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=1;
% 3. Definirea functiei
r1=a*cot(t);
r2=-a*cot(t);
%% REPREZENTARE GRAFICA
figure
h1=polarplot(t,r1);hold on;
h2=polarplot(t,r2);hold off;
h1.Color='[0 0.447 0.741]';
h2.Color='[0 0.447 0.741]';
title('curba Kappa');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.28. Reprezentarea grafică a curbei Kappa.

Problema 4.16

Se consideră funcţia:

$$f:[-\pi,\pi]\to\mathbb{R},\ f(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}.$$

Să se reprezinte grafic funcţia $f(x)$ folosind următoarele tipuri de grafice:

a) Grafic în coordonate carteziene (`plot`), figura 4.29.

b) Grafic în coordonate logaritmice (`loglog`, `semilogx`, `semilogy`).

c) Grafic cu bare verticale (`bar`).

d) Grafic în trepte (`stairs`).

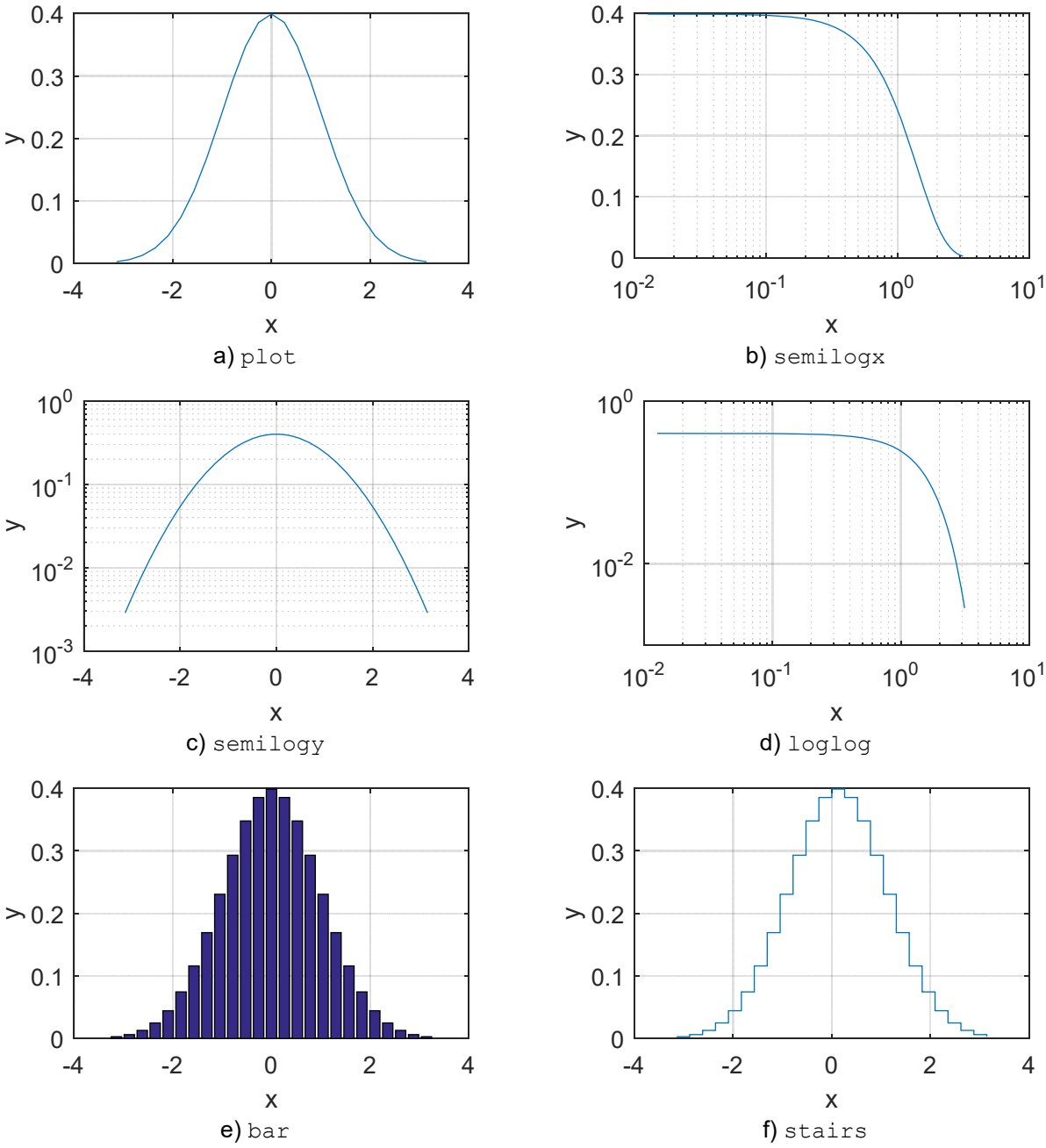


Figura 4.29. Reprezentarea grafică a funcției $f(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$.

Problema 4.17

Se consideră funcţia:

$$g: [0.1,100] \to \mathbb{R},\ g(x) = \sqrt{x} \cdot \left(x - \sqrt{x}\right).$$

Să se reprezinte grafic funcţia $g(x)$ folosind următoarele tipuri de grafice:

a) Grafic în coordonate carteziene (`plot`), figura 4.30.

b) Grafic în coordonate logaritmice (`loglog`, `semilogx`, `semilogy`).

c) Grafic cu bare verticale (`bar`).

d) Grafic în trepte (`stairs`).

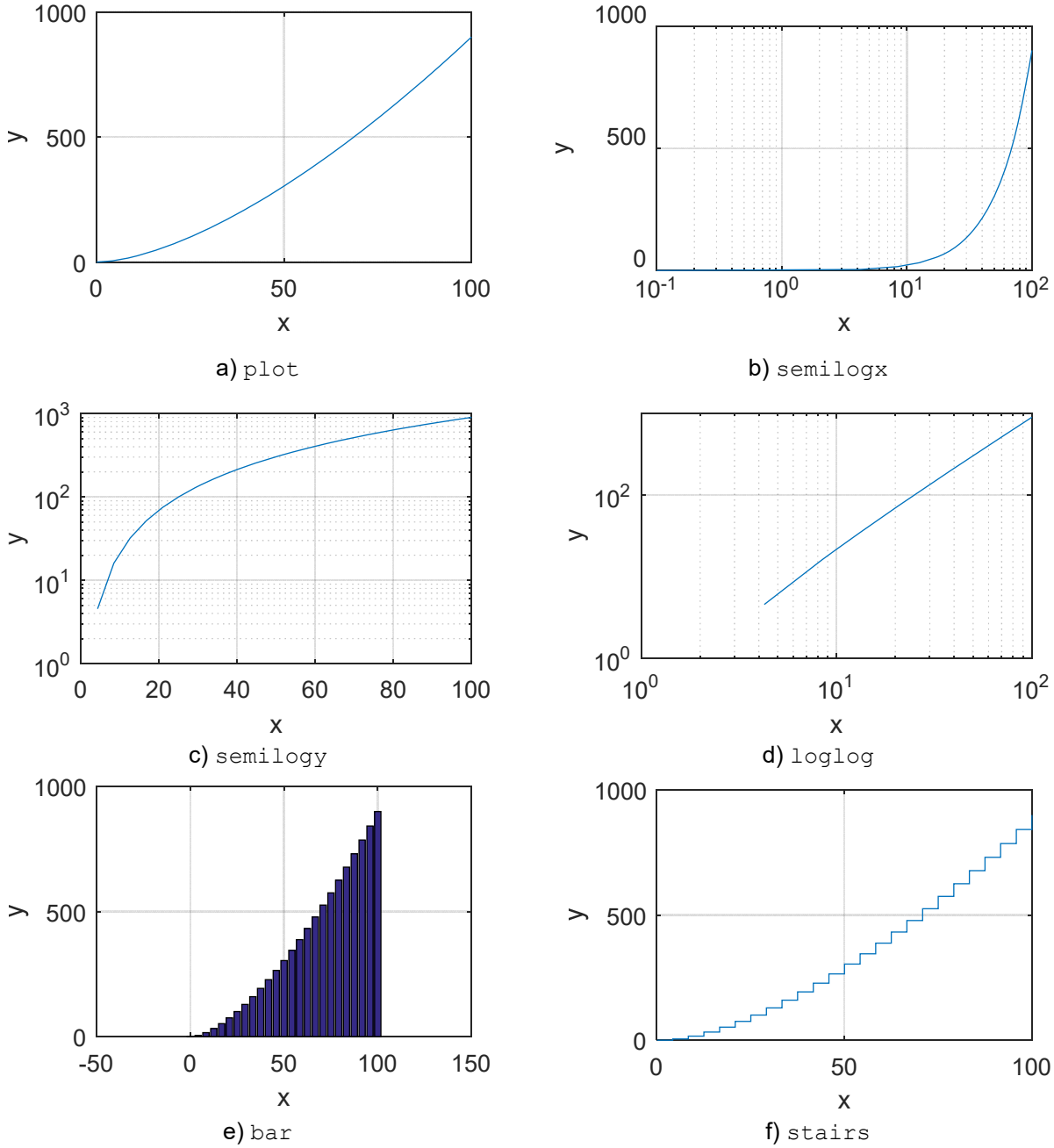


Figura 4.30. Reprezentarea grafică a funcției $g(x) = \sqrt{x} \cdot \left(x - \sqrt{x}\right)$.

Problema 4.18

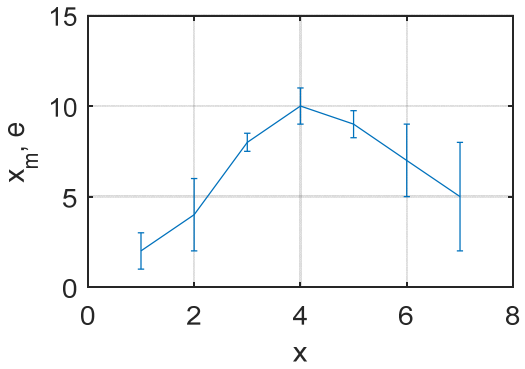
Se consideră următorul set de date rezultat din analiza statistică a unor rezultate experimentale:

x	1	2	3	4	5	6	7
$\bar{x}$	2	4	8	10	9	7	5
e	1.0	2.0	0.5	1.0	0.75	2	3

Să se reprezinte grafic intervalele de eroare e asociate setului de date $\{x, \bar{x}\}$, figura 4.31.

```
%% DATE DE INTRARE
% 1. Punctele de masurare
x=1:7;
% 2. Valorile medii
xm=[2 4 8 10 9 7 5];
% 3. Erorile
e=[1 2 0.5 1 0.75 2 3];
%% REPREZENTARE GRAFICA
figure
errorbar(x,xm,e);grid on;
xlabel('x');ylabel('x_m, e');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.31. Reprezentarea grafică a intervalelor de eroare.

Problema 4.19

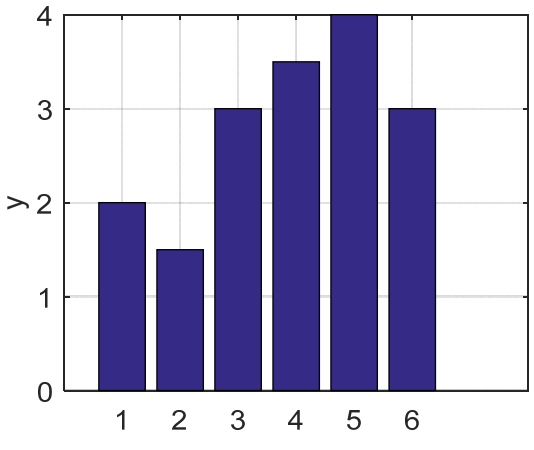
Se consideră următorul set de date:

	1	2	3	4	5	6
y_1	2	1.5	3	3.5	4	3

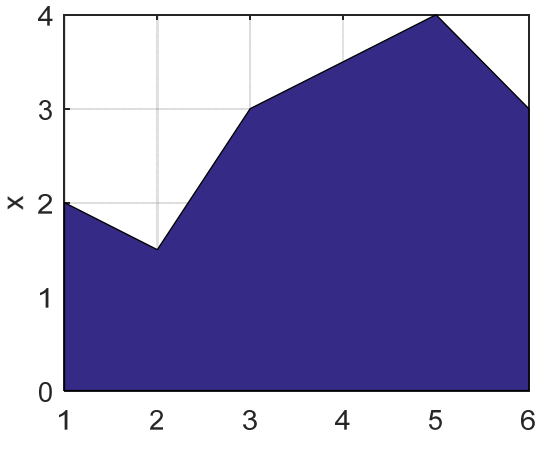
Să se reprezinte grafic folosind următoarele tipuri de grafice:

a) Grafic cu bare verticale (`bar`), figura 4.32, a) și orizontale (`barh`).

b) Grafic de tip `area`, figura 4.32, b).



a) `bar`



b) `area`

Figura 4.32. Reprezentarea grafică de tip `bar` si `area`.

Problema 4.20

Se consideră următorul set de date:

	1	2	3	4	5	6
y_1	2	1.5	3	3.5	4	3
y_2	1.5	1	2.5	3	3.25	3.5
y_3	1.25	1	2.75	3.25	2.75	2.5

Să se reprezinte grafic setul de date folosind următoarele tipuri de grafice:

a) Grafic cu bare verticale (`bar`) și orizontale (`barh`), figura 4.33, a) și b).

b) Grafic cu bare verticale (`bar3`) și orizontale (`bar3h`), figura 4.33, c) și d).

c) Grafic de tip `area`, figura 4.33, e).

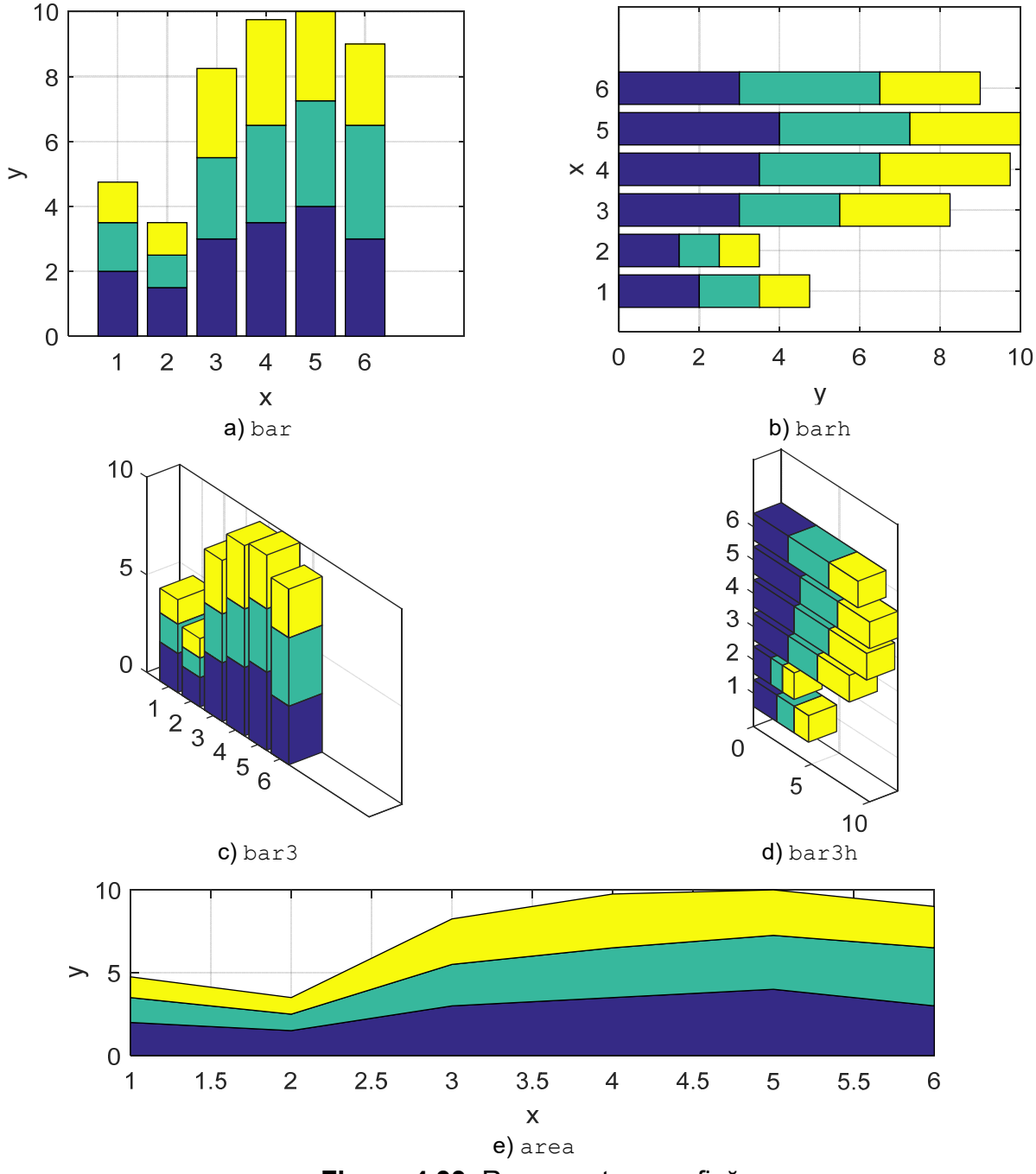


Figura 4.33. Reprezentare grafică.

Problema 4.21

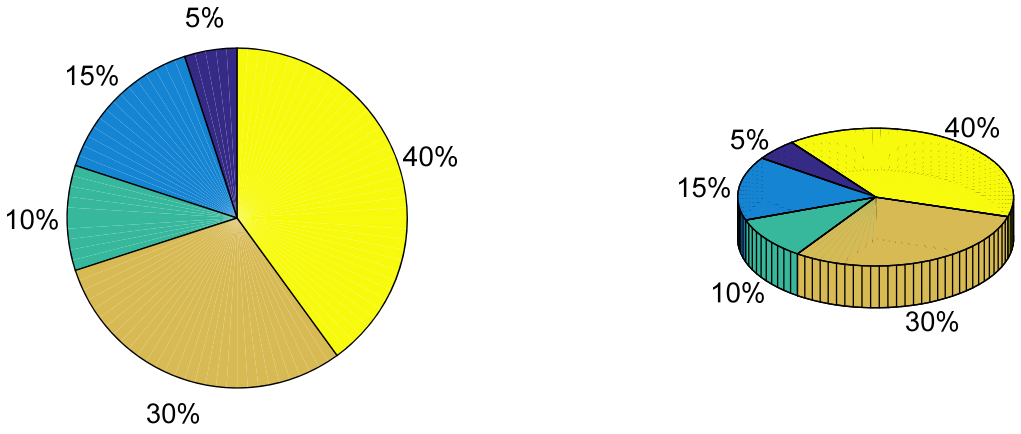
Se consideră următorul set de date:

	1	2	3	4	5	$\sum y$
y	5	15	10	30	40	100
y[%]	5%	15%	10%	30%	40%	100%

Să se reprezinte grafic setul de date folosind următoarele tipuri de grafice:

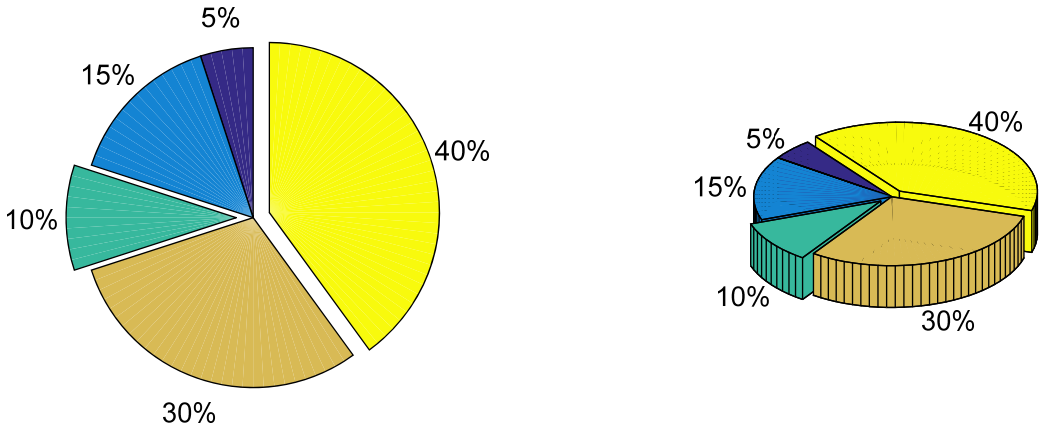
a) Grafice de tip `pie` şi `pie3`, figura 4.34.

b) Să se extragă din graficele obținute sectoarele 3 și 5, figura 4.35.



a) `pie` b) `pie3`

Figura 4.34. Reprezentarea grafică de tip `pie` și `pie3`.



a) `pie` b) `pie3`

Figura 4.35. Reprezentarea grafică de tip `pie` și `pie3` cu sectoare explodate.

Problema 4.22

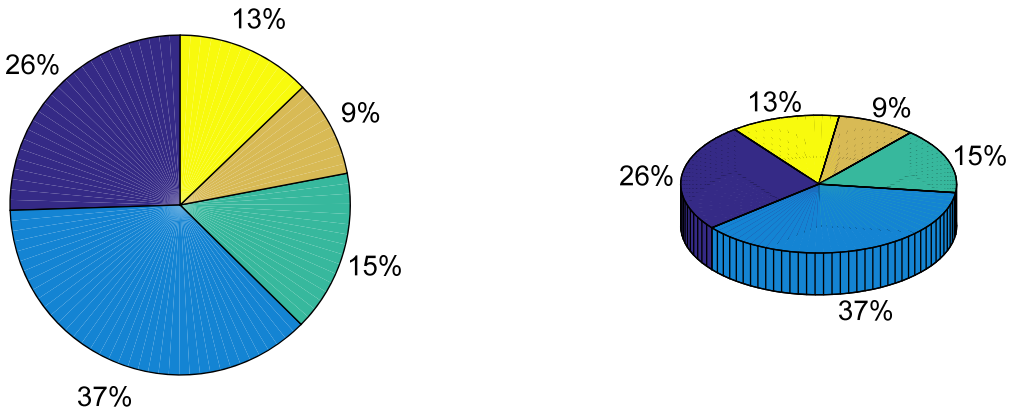
Se consideră următorul set de date:

	1	2	3	4	5	$\sum y$
y	255	371	154	92	128	1000
y[%]	26%	37%	15%	9%	13%	100%

Să se reprezinte grafic setul de date folosind următoarele tipuri de grafice:

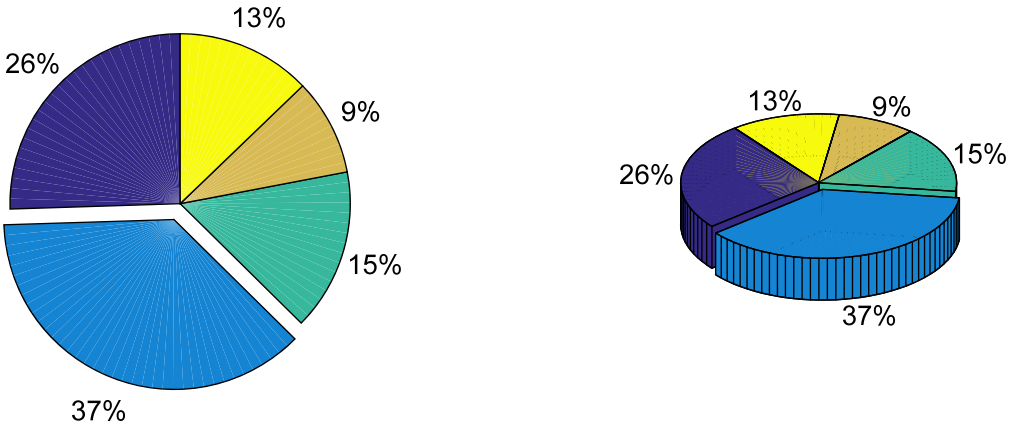
a) Grafice de tip `pie` și `pie3`, figura 4.36.

b) Să se extragă din graficele obținute sectorul 2, figura 4.37.



a) `pie` b) `pie3`

Figura 4.36. Reprezentarea grafică de tip `pie` și `pie3`.



a) `pie` b) `pie3`

Figura 4.37. Reprezentarea grafică de tip `pie` și `pie3` cu sectoare explodate.

Problema 4.23

Se consideră un număr de n=7 puncte $P_i(x_i, y_i), i = 1 \ldots n$ având coordonatele:

n	1	2	3	4	5	6	7
x [m]	1	2	8	6.5	7	5	3.5
y [m]	5	1	2	3	5	6	4.5

a) Să se reprezinte grafic poligonul având colțurile în punctele considerate.
b) Să se folosească diferite culori (alb, roșu, albastru, negru, alb, verde, galben).
c) Să se calculeze aria poligonului și să se afișeze valoarea obținută în fereastra grafică începând cu punctul de coordonate (3.5;4.5), figura 4.38.

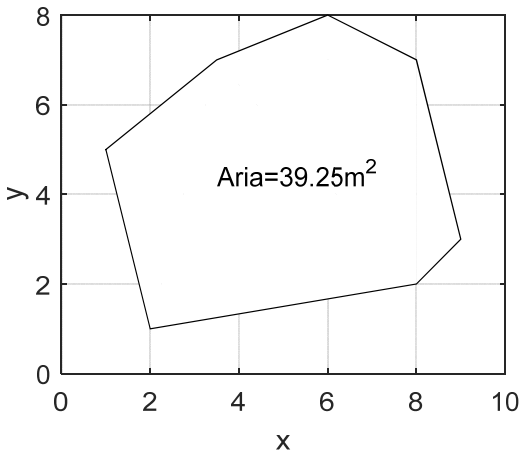
```
%% GRAFIC DE TIP FILL
clear all;clc;close all;

%% DATE DE INTRARE
% 1. Abscisele
x=[1 2 8 9 8 6 3.5];
% 2. Ordonatele
y=[5 1 2 3 7 8 7];

%% ARIA POLIGONULUI
A=polyarea(x,y)

%% REPREZENTARE GRAFICA
figure
fill(x,y,'w');
grid on;
xlabel('x');
ylabel('y');
t=['Aria=' num2str(A) 'm^2'];
text(3.5,4.5,t);
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 4.38. Reprezentarea grafică și calculul ariei unui poligon.

CAPITOLUL 5
REPREZENTAREA GRAFICĂ A FUNCŢIILOR 3D

5.1. INSTRUCŢIUNEA `plot3`

Instrucțiunea `plot3` realizează reprezentarea grafică 3D a unei curbe, prin utilizarea obiectelor grafice de tip linie, a unei funcții matematice definită sub formă parametrică pe un domeniu de definiție oarecare.

Astfel, instrucțiunea:

```
plot3(fx,fy,fz)
```

realizează graficul 3D al funcției parametrice $f_x(t)$, $f_y(t)$ şi $f_z(t)$ pe domeniul de variaţie al variabilei $t = [t_{min}, t_{max}]$. Atunci când variabila independentă parcurge domeniul de variaţie impus, valorile $f_x(t)$, $f_y(t)$ şi $f_z(t)$ reprezintă coordonatele spaţiale ale punctelor graficului.

5.2. INSTRUCŢIUNEA `mesh`

Instrucțiunea `mesh` realizează reprezentarea grafică 3D prin utilizarea obiectelor grafice de tip `wireframe` a unei funcții matematice definită pe un domeniu oarecare.

Astfel, pentru reprezentarea grafică a funcției definită în mod explicit `z=functie(x,y)` pe domeniul oarecare $x \times y = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ se utilizează instrucţiunile:

```
mesh(X,Y,Z)
```

în care X, Y reprezintă matrice determinate cu instrucțiunea:

```
[X,Y]=meshgrid(x,y)
```

iar Z reprezintă matricea calculată conform instrucțiunii:

```
Z=functie(X,Y)
```

În cazul funcțiilor parametrice $f_x(u,v)$, $f_y(u,v)$ şi $f_z(u,v)$ definite pe domeniu oarecare $u \times v = [u_{min}, u_{max}] \times [v_{min}, v_{max}]$ se utilizează instrucţiunile:

```
mesh(X,Y,Z)
```

în care matricele X, Y și Z se calculează conform instrucțiunilor:

```
X=fx(U,V);Y=fy(U,V);Z=fz(U,V)
```

iar matricele U și V se determină cu instrucțiunile:

```
[U,V]=meshgrid(u,v)
```

Instrucțiunea `meshc` realizează reprezentarea grafică 3D, prin utilizarea combinată atât a obiectelor grafice de tip `wireframe` cât și a liniilor de contur, a unei funcții definită atât în mod explicit, cât şi parametric, pe un domeniu de definiție oarecare, conform sintaxei:

```
meshc(X,Y,Z)
```

În expresiile funcțiilor de analizat intervin operatorii specifici operațiilor element cu element „.*”, „./” și „.^”.

Atribuirea unei anumite mape de culoare reprezentării grafice 3D se face cu instrucțiunea:

```
colormap(map)
```

în care `map` reprezintă una din mapele de culoare predefinite: `jet`, `hsv`, `hot`, `cool`, `spring`, `summer`, `autumn`, `winter`, `gray`, `bone`, `cooper`, `pink`, `lines`.

Afişarea scalei culorilor pentru a permite identificarea corespondenței dintre valorile numerice și culorile asociate se face cu instrucțiunea:

```
colorbar
```

5.3. INSTRUCŢIUNEA `surf`

Instrucțiunea `surf` realizează reprezentarea grafică 3D prin utilizarea obiectelor grafice de tip `surface` a unei funcții matematice definită pe un domeniu oarecare.

Astfel, pentru reprezentarea grafică a funcției definită în mod explicit `z=functie(x,y)` pe domeniul oarecare $x \times y = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ se utilizează instrucţiunile:

```
surf(X,Y,Z)
```

în care X, Y reprezintă matrice determinate cu instrucțiunea:

```
[X,Y]=meshgrid(x,y)
```

iar Z reprezintă matricea calculată conform instrucțiunii:

```
Z=functie(X,Y)
```

În cazul funcțiilor parametrice $f_x(u,v)$, $f_y(u,v)$ şi $f_z(u,v)$ definite pe domeniu oarecare $u \times v = [u_{min}, u_{max}] \times [v_{min}, v_{max}]$ se utilizează instrucțiunile:

```
surf(X,Y,Z)
```

în care matricele X, Y și Z se calculează conform instrucțiunilor:

```
X=fx(U,V)
Y=fy(U,V)
Z=fz(U,V)
```

iar matricele U și V se determină cu instrucțiunile:

```
[U,V]=meshgrid(u,v)
```

Instrucțiunea `surfc` realizează reprezentarea grafică 3D, prin utilizarea combinată atât a obiectelor grafice de tip `surface` cât și a liniilor de contur, a unei funcții definită atât în mod explicit, cât și parametric, pe un domeniu de definiție oarecare, conform sintaxei:

```
surfc(X,Y,Z)
```

5.4. INSTRUCŢIUNEA `contour`

Instrucțiunea `contour` realizează reprezentarea grafică 2D prin utilizarea liniilor de contur a unei funcții matematice 3D definită pe un domeniu oarecare.

Astfel, pentru reprezentarea grafică a funcției definită în mod explicit `z=functie(x,y)` pe domeniul oarecare $x \times y = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$ se utilizează instrucţiunile:

```
C=contour(X,Y,Z)
```

în care X, Y reprezintă matrice determinate cu instrucțiunea:

```
[X,Y]=meshgrid(x,y)
```

iar Z reprezintă matricea calculată conform instrucțiunii:

```
Z=functie(X,Y)
```

În cazul funcțiilor parametrice $f_x(u,v)$, $f_y(u,v)$ şi $f_z(u,v)$ definite pe domeniu oarecare $u \times v = [u_{min}, u_{max}] \times [v_{min}, v_{max}]$ se utilizează instrucțiunile:

```
C=contour(X,Y,Z)
```

în care matricele X, Y și Z se calculează conform instrucțiunilor:

```
X=fx(U,V)
Y=fy(U,V)
Z=fz(U,V)
```

iar matricele U și V se determină cu instrucțiunile:

```
[U,V]=meshgrid(u,v)
```

Instrucțiunea `contourf` realizează reprezentarea grafică 2D, prin utilizarea combinată atât a liniilor de contur, cât și a colorării spațiului dintre liniile de contur, conform sintaxei:

```
contourf(X,Y,Z)
```

Pentru reprezentarea grafică 3D prin utilizarea liniilor de contur a unei funcții matematice 3D definită pe un domeniu oarecare se utilizează instrucțiunea:

```
C=contour3(X,Y,Z)
```

Etichetarea liniilor de contur se realizează cu ajutorul instrucțiunii:

```
clabel(C)
```

5.5. PROBLEME

Problema 5.1

Să se reprezinte grafic curbele spațiale corespunzătoare următoarelor funcții definite sub formă parametrică:

a) Elicea elicoidală, (figura 5.1):

$$\begin{cases} x = r\cos t \\ y = r\sin t \\ z = ct \end{cases}$$
$$r = 1,\ c = 1,\ t = [0, +8\pi]$$

b) Spirala conică, (figura 5.2):

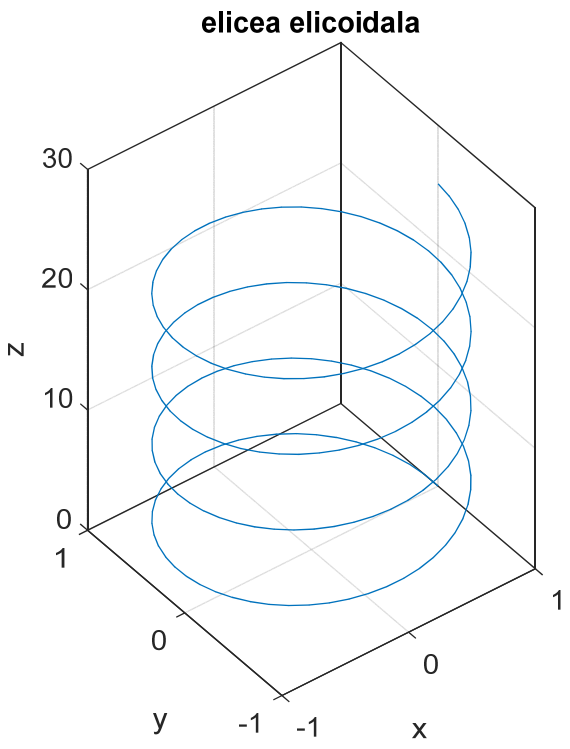
$$\begin{cases} x = tr\cos(at) \\ y = tr\sin(at) \\ z = t \end{cases}$$
$$a = 1,\ r = 1,\ t = [0, +12\pi]$$

c) Spirala sferică, (figura 5.3):

$$\begin{cases} x = \dfrac{\cos t}{\sqrt{1 + a^2t^2}} \\ y = \dfrac{\sin t}{\sqrt{1 + a^2t^2}} \\ z = \dfrac{-at}{\sqrt{1 + a^2t^2}} \end{cases}$$
$$a = 0.1,\ t = [-14\pi, +14\pi]$$

```
%% ELICEA ELICOIDALA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie al
functiei
tmin=0;
tmax=8*pi;
nt=200;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
r=1;c=1;
% 3. Definirea functiei
x=r*cos(t);
y=r*sin(t);
z=c*t;
%% REPREZENTARE GRAFICA
figure
plot3(x,y,z);
grid on;box on;
xlabel('x');
ylabel('y');
zlabel('z');
title('elicea elicoidala');
```

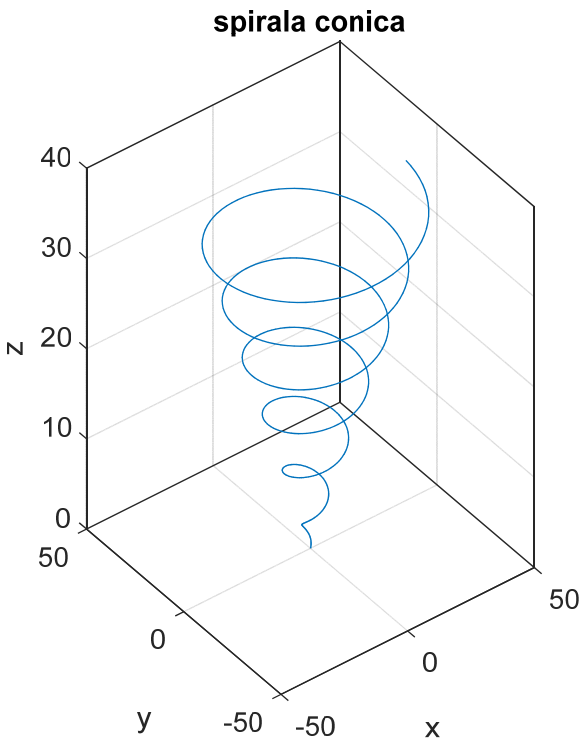
a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.1. Reprezentarea grafică a elicei elicoidale.

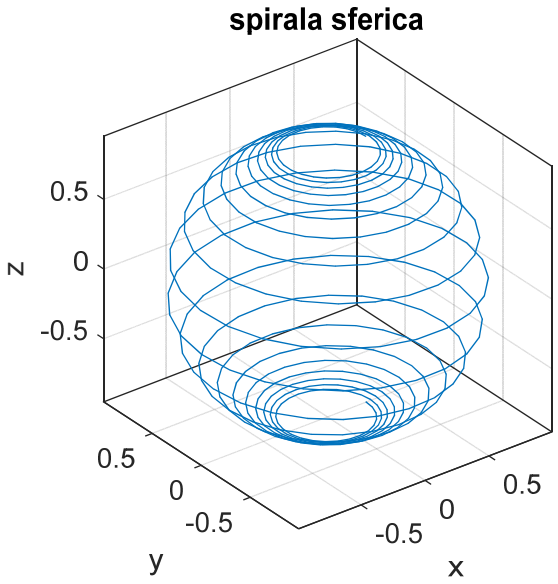
```
%% SPIRALA CONICA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=0;
tmax=12*pi;
nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrii caracteristici
a=1;r=1;
% 3. Definirea functiei
x=t*r.*cos(a*t);
y=t*r.*sin(a*t);
z=t;
%% REPREZENTARE GRAFICA
figure
plot3(x,y,z);
grid on;box on;
xlabel('x');
ylabel('y');
zlabel('z');
title('spirala conica');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.2. Reprezentarea grafică a spiralei conice.

```
%% SPIRALA SFERICA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul de definitie
tmin=-20*pi;
tmax=20*pi;
nt=500;
t=linspace(tmin,tmax,nt);
% 2. Parametrul caracteristic
a=0.05;
% 3. Definirea functiei
x=cos(t)./sqrt(1+a^2*t.^2);
y=sin(t)./sqrt(1+a^2*t.^2);
z=-a*t./sqrt(1+a^2*t.^2);
%% REPREZENTARE GRAFICA
figure
plot3(x,y,z);
grid on;box on;axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
title('spirala sferica');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.3. Reprezentarea grafică a spiralei sferice.

Problema 5.2

Să se reprezinte grafic, prin utilizarea obiectelor grafice de tip `wireframe`, următoarele funcții definite în mod explicit:

a) Paraboloidul hiperbolic (figura 5.4):

$$z(x,y) = \frac{y^2}{b^2} - \frac{x^2}{a^2}$$
$$a = 1,\ b = 1,\ x \times y = [-\pi, +\pi] \times [-\pi, +\pi]$$

b) Suprafața de tip „Crossed Trough”, (figura 5.5):

$$z = x^2 y^2$$
$$x \times y = [-2\pi, +2\pi] \times [-2\pi, +2\pi]$$

c) Suprafața de tip „Monkey Saddle”, (figura 5.6):

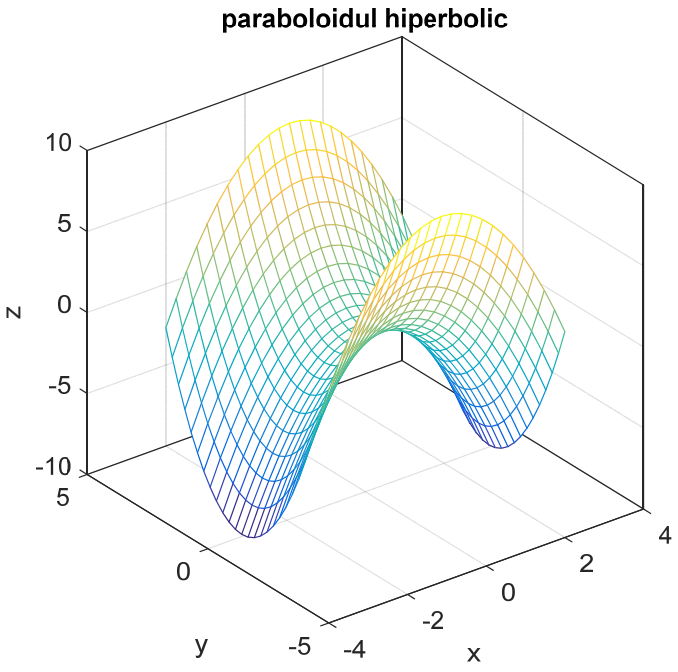
$$z = x(x^2 - 3y^2)$$
$$x \times y = [-2\pi, +2\pi] \times [-2\pi, +2\pi]$$

d) Suprafața de tip „Peano”, (figura 5.7):

$$z = (2x^2 - y)(y - x^2)$$
$$x \times y = [-2\pi, +2\pi] \times [-2\pi, +2\pi]$$

```
%% PARABOLOIDUL HIPERBOLIC
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-pi;xmax=pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-pi;ymax=pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 2. Parametrii functiei
a=1;b=1;
% 3. Definirea functiei
Z=Y.^2/b^2-X.^2/a^2;
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;
xlabel('x');
ylabel('y');
zlabel('z');
title('paraboloidul
hiperbolic');
```

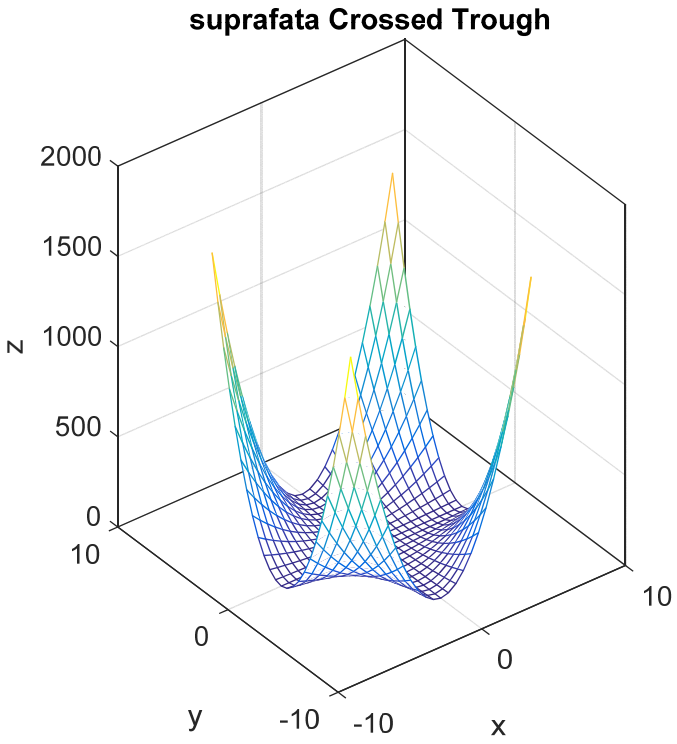
a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.4. Reprezentarea grafică a paraboloidului hiperbolic.

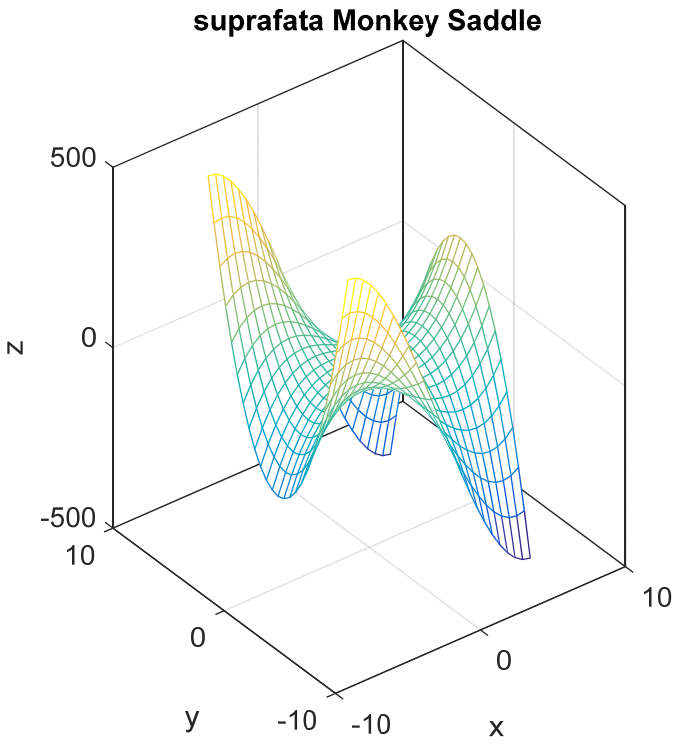
```
%% SUPRAFATA CROSSED TROUGH
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-2*pi;xmax=2*pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-2*pi;ymax=2*pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Definirea functiei
Z=X.^2.*Y.^2;
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;
xlabel('x');
ylabel('y');
zlabel('z');
title('suprafata Crossed
Trough');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.5. Reprezentarea grafică a suprafeței Crossed Trough.

```
%% SUPRAFATA MONKEY SADDLE
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-2*pi;xmax=2*pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-2*pi;ymax=2*pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Definirea functiei
Z=X.*(X.^2-3*Y.^2);
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;
xlabel('x');
ylabel('y');
zlabel('z');
title('suprafata Monkey
Saddle');
```

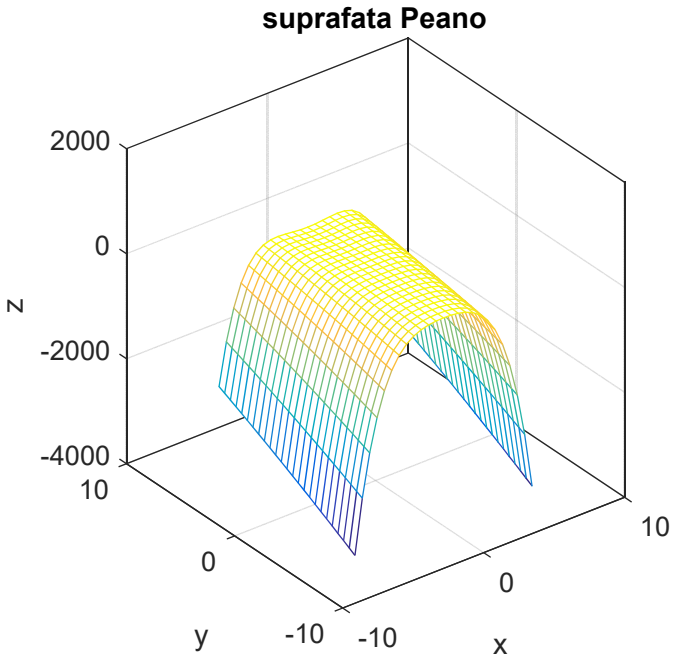


a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.6. Reprezentarea grafică a suprafeței Monkey Saddle.

```
%% SUPRAFATA PEANO
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-2*pi;xmax=2*pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-2*pi;ymax=2*pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Definirea functiei
Z=(2*X.^2-Y).*(Y-X.^2);
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata Peano');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.7. Reprezentarea grafică a suprafeței Peano.

Problema 5.3

Să se reprezinte grafic, prin utilizarea obiectelor grafice de tip `wireframe`, următoarele funcții definite în mod parametric:

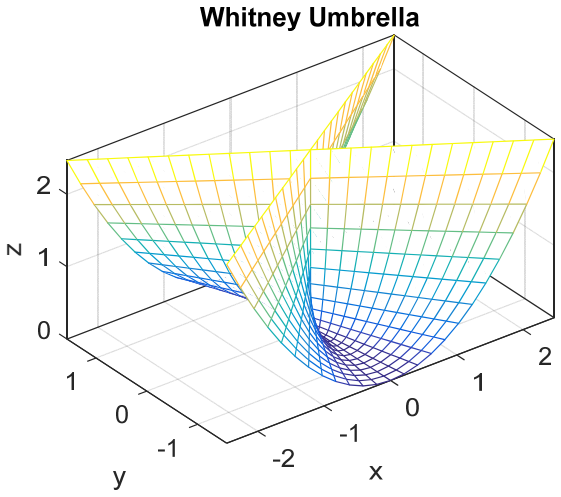
a) Suprafața de tip „Whitney Umbrella”, (figura 5.8):

$$\begin{cases} x = uv \\ y = u \\ z = v^2 \end{cases}$$

$$u \times v = [-\pi/2, +\pi/2] \times [-\pi/2, +\pi/2]$$

```
%% SUPRAFATA WHITNEY UMBRELLA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=-pi/2;umax=pi/2;nu=25;
u=linspace(umin,umax,nu);
vmin=-pi/2;vmax=pi/2;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=U.*V;Y=U;Z=V.^2;
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('Whitney Umbrella');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.8. Reprezentarea grafică a suprafeței Whitney Umbrella.

b) Suprafața de tip „Moebius Strip”, (figura 5.9):

$$\begin{cases} x = [R + u\cos(v/2)]\cos v \\ y = [R + u\cos(v/2)]\sin v \\ z = u\sin(v/2) \end{cases}$$

$$u \times v = [-w, +w] \times [0, +2\pi],\ R = 1,\ w = 0.3$$

c) Suprafața de tip octaedru hiperbolic, (figura 5.10):

$$\begin{cases} x = (\cos u \cdot \cos v)^3 \\ y = (\sin u \cdot \cos v)^3 \\ z = (\sin v)^3 \end{cases}$$

$$u \times v = [-\pi/2, +\pi/2] \times [-\pi, +\pi]$$

d) Suprafața de tip „Eight Surface”, (figura 5.11):

$$\begin{cases} x = \cos u \sin 2v \\ y = \sin u \sin 2v \\ z = \sin v \end{cases}$$

$$u \times v = [0, +2\pi] \times [-\pi/2, +\pi/2]$$

e) Suprafața elicoidală, (figura 5.12):

$$\begin{cases} x = u\cos v \\ y = u\sin v \\ z = cv \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +4\pi]$$

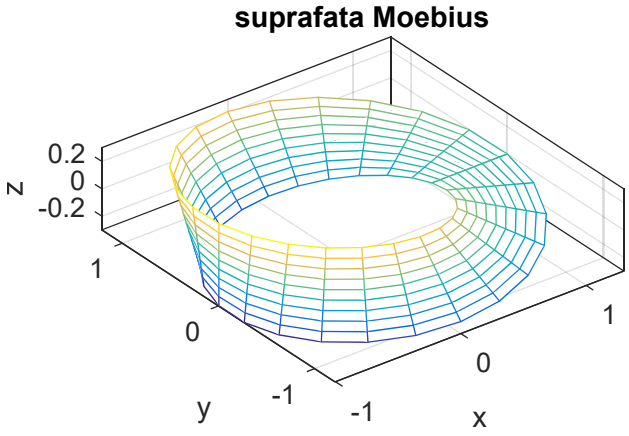
f) Suprafața de tip „Seashell”, (figura 5.13):

$$\begin{cases} x = 2\left[1 - e^{u/6\pi}\right]\cos u\,[\cos(v/2)]^2 \\ y = 2\left[-1 + e^{u/6\pi}\right]\sin u\,[\cos(v/2)]^2 \\ z = 1 - e^{u/3\pi} - \sin v + e^{u/6\pi}\sin v \end{cases}$$

$$u \times v = [0, +6\pi] \times [0, +2\pi]$$

```
%% SUPRAFATA MOEBIUS STRIP
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
w=0.3;R=1;
umin=-w;umax=w;nu=10;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=(R+U.*cos(V/2)).*cos(V);
Y=(R+U.*cos(V/2)).*sin(V);
Z=U.*sin(V/2);
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata Moebius');
```

a) fișierul `script`

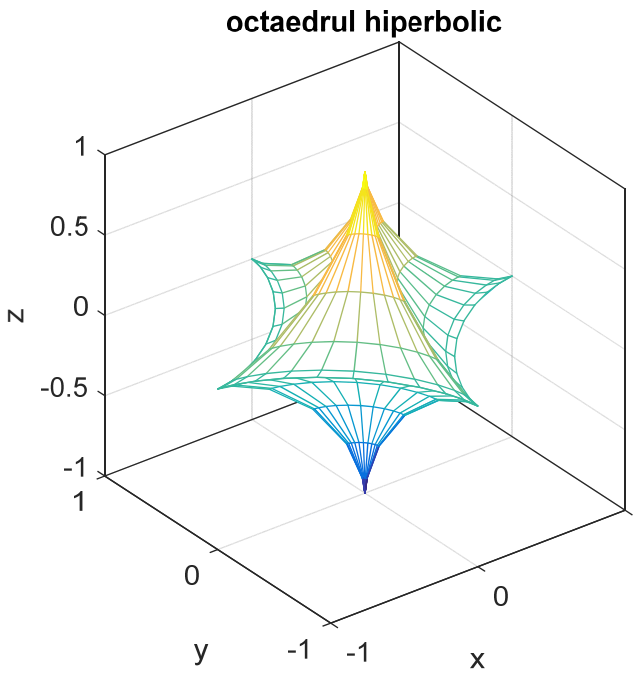


b) reprezentarea grafică obținută

Figura 5.9. Reprezentarea grafică a suprafeței Moebius.

```
%% OCTAEDRUL HIPERBOLIC
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=-pi/2;umax=pi/2;nu=25;
u=linspace(umin,umax,nu);
vmin=-pi;vmax=pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=(cos(U).*cos(V)).^3;
Y=(sin(U).*cos(V)).^3;
Z=(sin(V)).^3;
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('octaedrul hiperbolic');
```

a) fișierul `script`

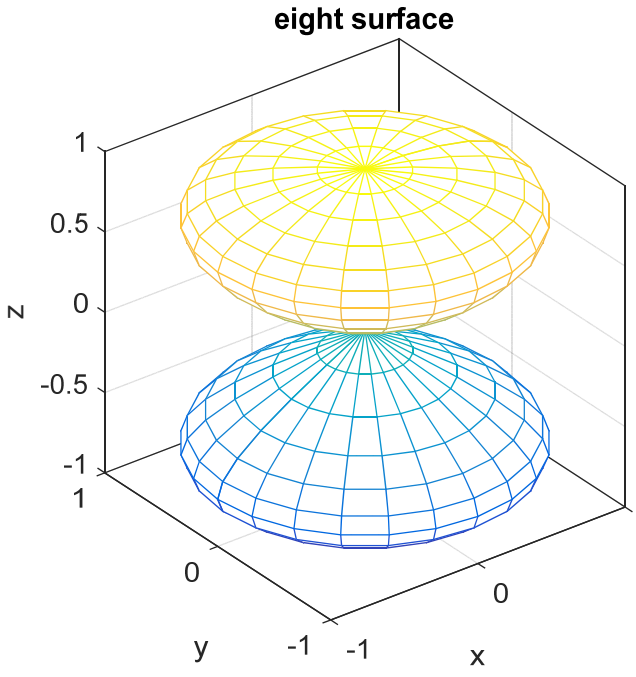


b) reprezentarea grafică obținută

Figura 5.10. Reprezentarea grafică a octaedrului hiperbolic.

```
%% EIGHT SURFACE
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=-pi/2;vmax=pi/2;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=cos(U).*sin(2*V);
Y=sin(U).*sin(2*V);
Z=sin(V);
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('eight surface');
```

a) fișierul `script`

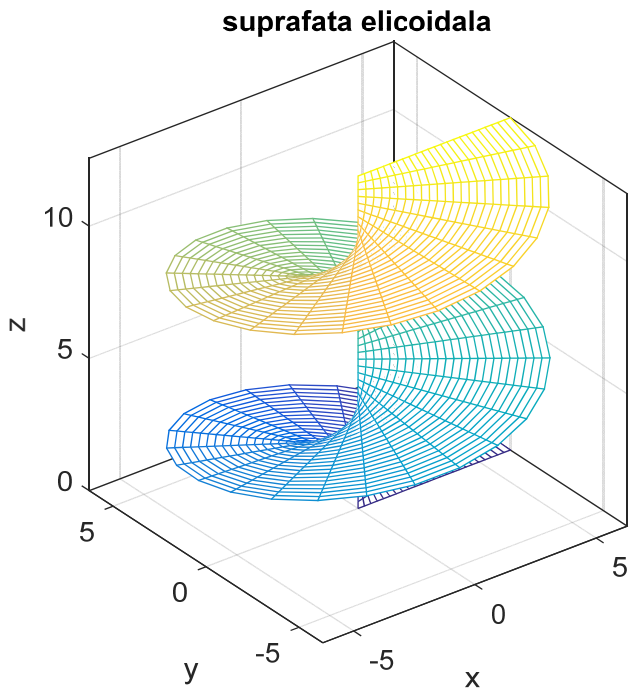


b) reprezentarea grafică obținută

Figura 5.11. Reprezentarea grafică a suprafeței Eight Surface.

```
%% SUPRAFATA ELICOIDALA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=4*pi;nv=50;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=U.*cos(V);
Y=U.*sin(V);
Z=V;
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata elicoidala');
```

a) fișierul `script`

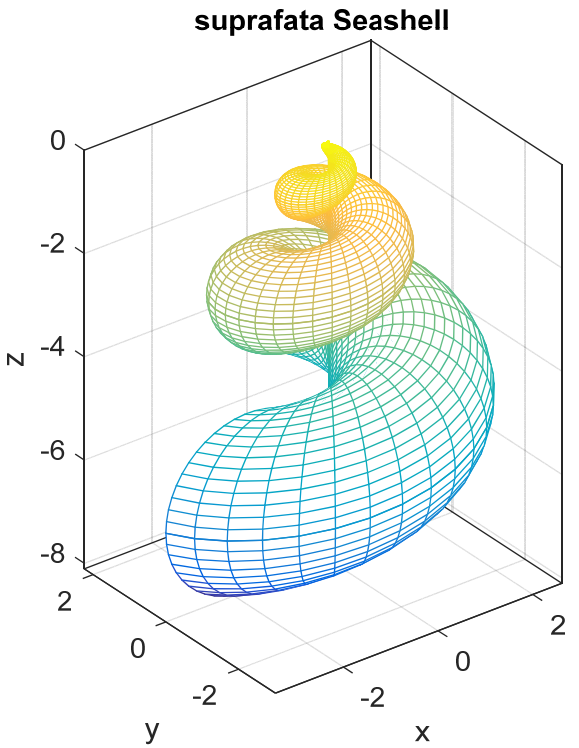


b) reprezentarea grafică obținută

Figura 5.12. Reprezentarea grafică a suprafeței elicoidale.

```
%% SUPRAFATA SEASHELL
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=6*pi;nu=100;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=50;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=2*(1-exp(U./(6*pi))).*cos(U).*
(cos(V/2)).^2;
Y=2*(-1+exp(U./(6*pi))).*sin(U).*
(cos(V/2)).^2;
Z=1-exp(U/(3*pi))-sin(V)+
exp(U/(6*pi)).*sin(V);
%% REPREZENTARE GRAFICA
figure
mesh(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata Seashell');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.13. Reprezentarea grafică a suprafeței Seashell.

Problema 5.4

Să se reprezinte grafic, prin utilizarea combinată atât a obiectelor grafice de tip `wireframe`, cât și a liniilor de contur a următoarelor funcții definite în mod parametric:

a) Suprafața de tip „Cross Cap”, (figura 5.14):

$$\begin{cases} x = 1/2 \cos u \sin 2v \\ y = 1/2 \sin u \sin 2v \\ z = 1/2 \left[(\cos v)^2 - (\cos u)^2 (\sin v)^2 \right] \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +\pi/2]$$

b) Suprafața de tip pseudosferă, (figura 5.15):

$$\begin{cases} x = \cos u \sin v \\ y = \sin u \sin v \\ z = \cos v + \ln \left[\tan \left(\frac{1}{2} v \right) \right] \end{cases},$$

$$u \times v = [0, +2\pi] \times [0, +\pi/2]$$

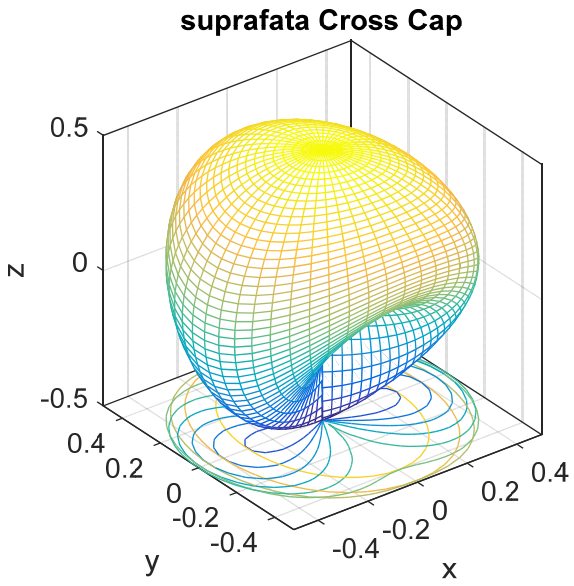
c) Suprafața de tip „Corkscrew”, (figura 5.16):

$$\begin{cases} x = a \cos u \cos v \\ y = a \sin u \cos v \\ z = a \sin v + bu \end{cases}$$

$$u \times v = [0, +\pi] \times [0, +2\pi],\ a = 1,\ b = 1$$

```
%% SUPRAFATA CROSS CAP
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=50;
u=linspace(umin,umax,nu);
vmin=0;vmax=pi/2;nv=50;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=1/2*cos(U).*sin(2*V);
Y=1/2*sin(U).*sin(2*V);
Z=1/2*((cos(V)).^2-(cos(U))
.^2.*(sin(V)).^2);
%% REPREZENTARE GRAFICA
figure
meshc(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata Cross Cap');
```

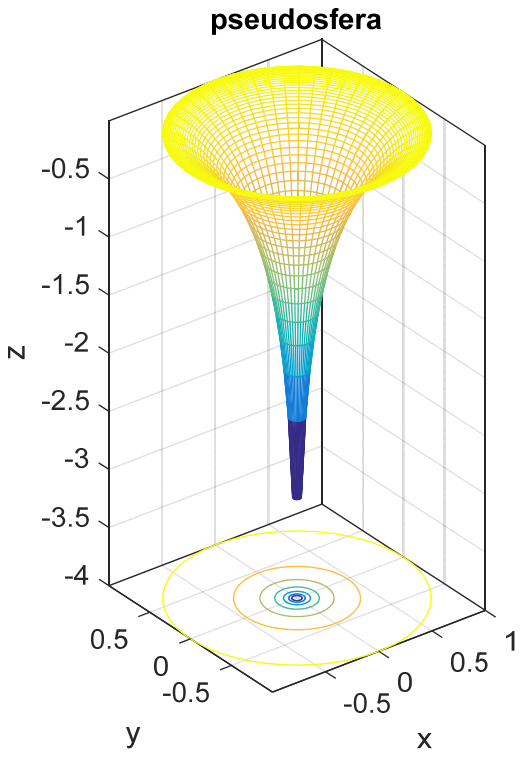
a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.14. Reprezentarea grafică a suprafeței Cross Cap.

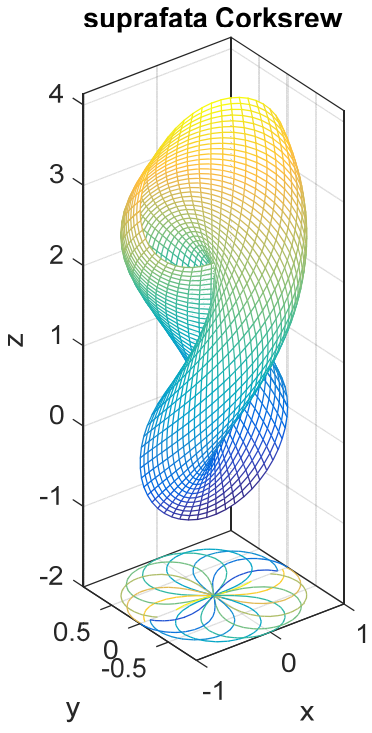
```
%% SUPRAFATA PSEUDOSFERA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=50;
u=linspace(umin,umax,nu);
vmin=0;vmax=pi/2;nv=50;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=cos(U).*sin(V);
Y=sin(U).*sin(V);
Z=cos(V)+log(tan(V/2));
%% REPREZENTARE GRAFICA
figure
meshc(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('pseudosfera');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.15. Reprezentarea grafică a pseudosferei.

```
%% SUPRAFATA CORKSCREW
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=pi;nu=50;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=50;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 3. Definirea functiei
X=cos(U).*cos(V);
Y=sin(U).*cos(V);
Z=sin(V)+U;
%% REPREZENTARE GRAFICA
figure
meshc(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('Corksrew');
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.16. Reprezentarea grafică a suprafeței Corkscrew.

Problema 5.5

Să se reprezinte grafic, prin utilizarea obiectelor grafice de tip `surface`, următoarele funcții definite în mod parametric:

a) Suprafața de tip sferoid:

$$\begin{cases} x = a \sin v \cos u \\ y = a \sin v \sin u \\ z = c \cos v \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +\pi]$$

sferoid de tip sferă ($a = c$): $a = 1$, $c = 1$, (figura 5.17)

sferoid de tip „oblate” ($a > c$): $a = 1.5$, $c = 1$,

sferoid de tip „prolate” ($a < c$): $a = 1$, $c = 1.5$,

b) Suprafața de tip tor:

$$\begin{cases} x = (c + a \cos v) \cos u \\ y = (c + a \cos v) \sin u \\ z = b \sin v \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +2\pi]$$

tor de tip circular $c > a$ şi $b = a$: $a = 1, c = 2.5, b = 1$, (figura 5.18)

tor de tip eliptic $c > a$ şi $b > a$ sau $b < a$: $a = 1, c = 2.25, b = 2$

tor de tip „horn” $a = c$: $a = 2.5, c = 2.5, b = 2.5$

tor de tip „spindle” $c < a$: $a = 2.5, c = 1.5, b = 2.5$

c) Suprafața de tip hiperboloid, (figura 5.19):

$$\begin{cases} x = a\sqrt{1 + u^2} \cos v \\ y = a\sqrt{1 + u^2} \sin v \\ z = cu \end{cases}$$

$$u \times v = [-\pi, +\pi] \times [0, +2\pi],\ a = 1,\ c = 1$$

d) Suprafața de tip „Funnel”, (figura 5.20):

$$\begin{cases} x = u \cos v \\ y = u \sin v \\ z = a \ln u \end{cases}$$

$$u \times v = [0.1, +\pi] \times [0, +2\pi],\ a = 2,$$

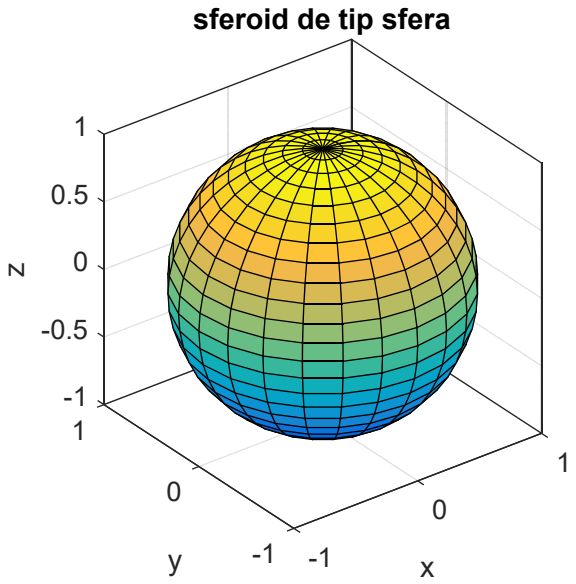
e) Suprafața de tip „Gabriel’s Horn”, (figura 5.21):

$$\begin{cases} x = u \\ y = a\dfrac{\cos v}{u} \\ z = a\dfrac{\sin v}{u} \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +2\pi],\ a = 0.75$$

```
%% SFEROID DE TIP SFERA
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrii caracteristici
a=1;c=1;
% 3. Definirea functiei
X=a*sin(V).*cos(U);
Y=a*sin(V).*sin(U);
Z=c*cos(V);
%% REPREZENTARE GRAFICA
figure
surf(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('sferoid de tip sfera');
```

a) fișierul `script`

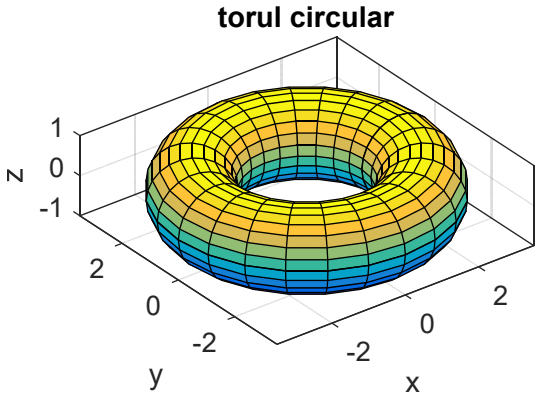


b) reprezentarea grafică obținută

Figura 5.17. Reprezentarea grafică a sferoidului de tip sferă.

```
%% TORUL CIRCULAR
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrii caracteristici
a=1;b=1;c=2.5;
% 3. Definirea functiei
X=(R1+R2*cos(V)).*cos(U);
Y=(R1+R2*cos(V)).*sin(U);
Z=R2*sin(V);
%% REPREZENTARE GRAFICA
figure
surf(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');zlabel('z');
title('torul circular');
```

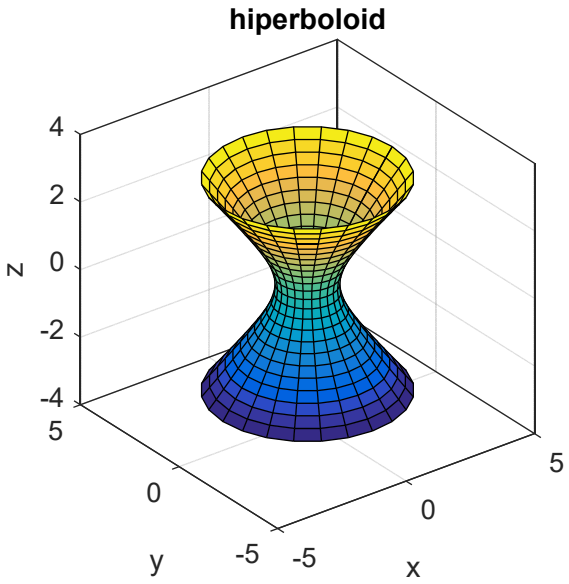
a) fișierul `script`



c) reprezentarea grafică obținută

Figura 5.18. Reprezentarea grafică a torului de tip circular.

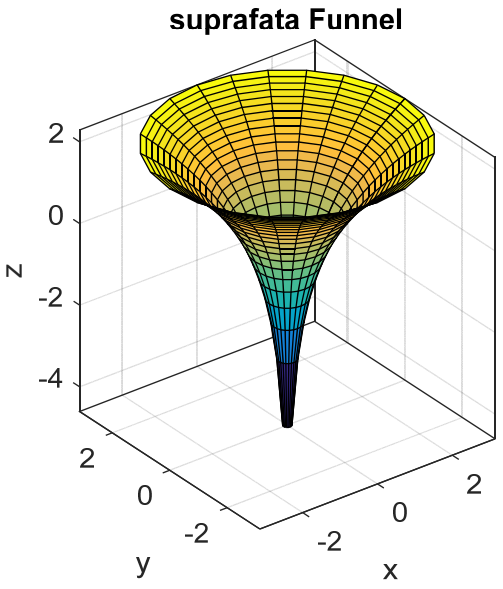
```
%% HIPERBOLOIDUL
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=-pi;umax=pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrii caracteristici
a=1;c=1;
% 3. Definirea functiei
X=a*sqrt(1+U.^2).*cos(V);
Y=a*sqrt(1+U.^2).*sin(V);
Z=c*U;
%% REPREZENTARE GRAFICA
surf(X,Y,Z);
grid on;box on;%axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('hiperboloid');
shading faceted
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.19. Reprezentarea grafică a hiperboloidului.

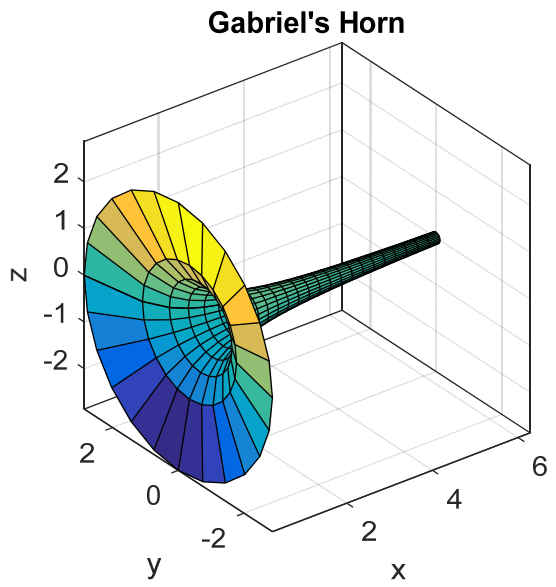
```
%% SUPRAFATA FUNNEL
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0.1;umax=pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrul caracteristic
a=2;
% 3. Definirea functiei
X=U.*cos(V);
Y=U.*sin(V);
Z=a*log(U);
%% REPREZENTARE GRAFICA
surf(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('suprafata Funnel');
shading faceted
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.20. Reprezentarea grafică a suprafeței Funnel.

```
%% SUPRAFATA GABRIEL'S HORN
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrul caracteristic
a=0.75;
% 3. Definirea functiei
X=U;
Y=a*cos(V)./U;
Z=a*sin(V)./U;
%% REPREZENTARE GRAFICA
figure
surf(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('Gabriel''s Horn');
shading faceted
```



a) fișierul `script` b) reprezentarea grafică obținută

Figura 5.21. Reprezentarea grafică a suprafeței Gabriel's Horn.

Problema 5.6

Să se reprezinte grafic, prin utilizarea combinată atât a obiectelor grafice de tip `surface`, cât și a liniilor de contur a următoarelor funcții definite în mod parametric:

a) Suprafața de tip paraboloid, (figura 5.22):

$$\begin{cases} x = a\sqrt{\frac{u}{h}}\cos v \\ y = a\sqrt{\frac{u}{h}}\sin v \\ z = u \end{cases}$$

$$u \times v = [0, +\pi] \times [0, +2\pi]$$

$$a = 1, h = 1$$

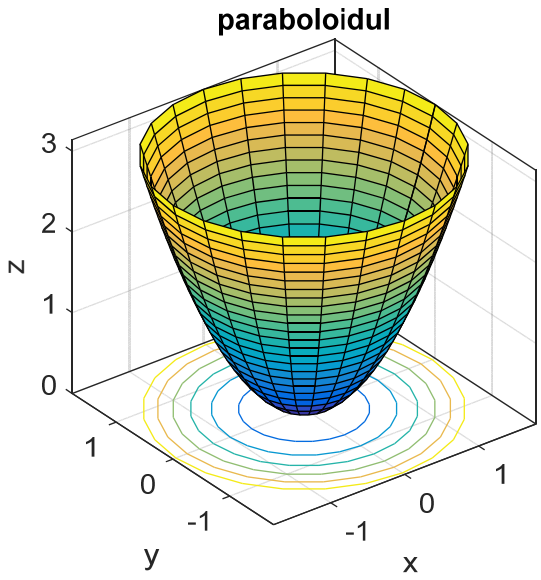
b) Suprafața de tip catenoid, (figura 5.23):

$$\begin{cases} x = c\cosh(v/c)\cos u \\ y = c\cosh(v/c)\sin u \\ z = v \end{cases}$$

$$u \times v = [0, +2\pi] \times [0, +2\pi],\ c = 3$$

```
%% PARABOLOIDUL
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrii caracteristici
a=1;h=1;
% 3. Definirea functiei
X=a*sqrt(U/h).*cos(V);
Y=a*sqrt(U/h).*sin(V);
Z=U;
%% REPREZENTARE GRAFICA
figure
surfc(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('paraboloid');
```

a) fișierul `script`

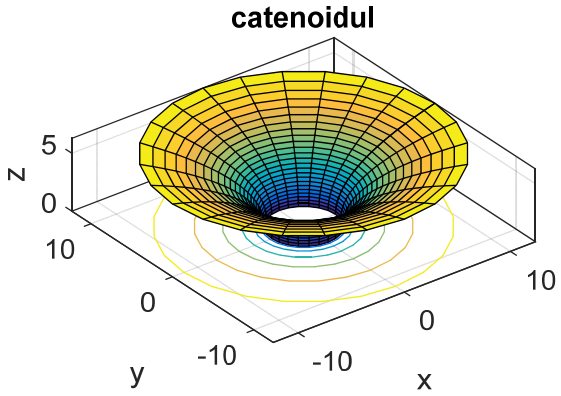


b) reprezentarea grafică obținută

Figura 5.22 Reprezentarea grafică a paraboloidului.

```
%% CATENOIDUL
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
umin=0;umax=2*pi;nu=25;
u=linspace(umin,umax,nu);
vmin=0;vmax=2*pi;nv=25;
v=linspace(vmin,vmax,nv);
% 2. Domeniul matriceal
[U,V]=meshgrid(u,v);
% 2. Parametrul caracteristic
c=3;
% 3. Definirea functiei
X=c*cosh(V/c).*cos(U);
Y=c*cosh(V/c).*sin(U);
Z=V;
%% REPREZENTARE GRAFICA
figure
surfc(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');zlabel(
'z');
title('catenoidul');
shading faceted
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.23. Reprezentarea grafică a catenoidului.

Problema 5.7

Să se reprezinte grafic, prin utilizarea obiectelor grafice de tip `surface`, următoarele funcții definite în mod explicit:

a) Suprafața definită prin ecuația, (figura 5.24):

$$z = -\sin\left(1+\sqrt{x^2+y^2}\right)$$

$$x = [-\pi, +\pi],\ y = [-\pi, +\pi]$$

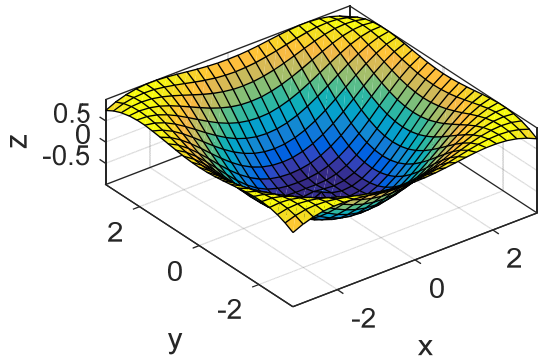
b) Suprafața de tip paraboloid, (figura 5.25):

$$z(x,y) = b(x^2+y^2)$$

$$b = 1/5,\ x \times y = [-\pi, +\pi] \times [-\pi, +\pi]$$

```
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-pi;xmax=pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-pi;ymax=pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Definirea functiei
Z=-sin(1+sqrt(X.^2+Y.^2));
%% REPREZENTARE GRAFICA
figure;surf(X,Y,Z);
grid on;box on;axis equal;
xlabel('x');ylabel('y');
zlabel('z');
```

a) fișierul `script`

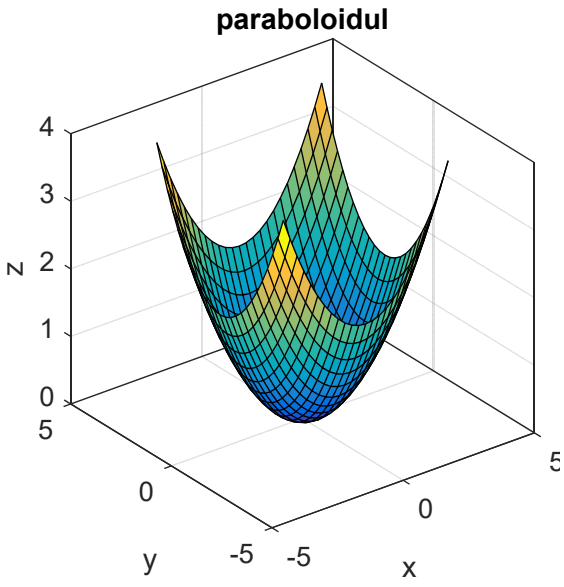


b) reprezentarea grafică obținută

Figura 5.24. Reprezentarea grafică a unei suprafețe.

```
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-pi;xmax=pi;nx=25;
x=linspace(xmin,xmax,nx);
ymin=-pi;ymax=pi;ny=25;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Parametrul caracteristic
b=1/5;
% 4. Definirea functiei
Z=b*(X.^2+Y.^2);
%% REPREZENTARE GRAFICA
figure
surf(X,Y,Z);
grid on;box on;%axis equal;
xlabel('x');ylabel('y');
zlabel('z');
title('paraboloidul');
```

a) fișierul `script`



b) reprezentarea grafică obținută

Figura 5.25. Reprezentarea grafică a paraboloidului.

Problema 5.8

Să se reprezinte grafic, prin utilizarea liniilor de contur 2D și 3D următoarele funcții:

a) Suprafața definită prin ecuația (figura 5.26):

$$z = -x \cdot e^{-x^2-y^2}$$

$$x = [-2, +2],\ y = [-2, +2]$$

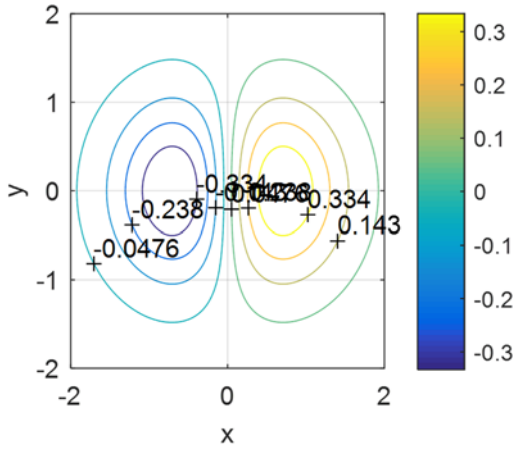
b) Suprafața definită prin ecuația (figura 5.27):

$$z = y^3 + 3e^x \sin(x-3)$$

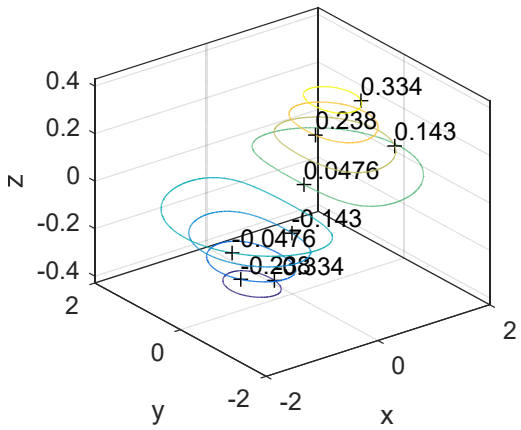
$$x = [-2\pi, +2\pi],\ y = [-2\pi, +2\pi]$$

```
%% INSTRUCTIUNEA CONTOUR
clear all;clc;close all;
%% DATE DE INTRARE
% 1. Domeniul vectorial
xmin=-2;xmax=2;nx=250;
x=linspace(xmin,xmax,nx);
ymin=-2;ymax=2;ny=250;
y=linspace(ymin,ymax,ny);
% 2. Domeniul matriceal
[X,Y]=meshgrid(x,y);
% 3. Definirea functiei
Z=X.*exp(-X.^2-Y.^2);
% 4. Numarul liniilor de contur
nc=8;
%% REPREZENTARE GRAFICA
figure
C=contour(X,Y,Z,nc);
grid on;box on;
clabel(C);
xlabel('x');ylabel('y');
zlabel('z');
colorbar
```

a) fișierul `script`

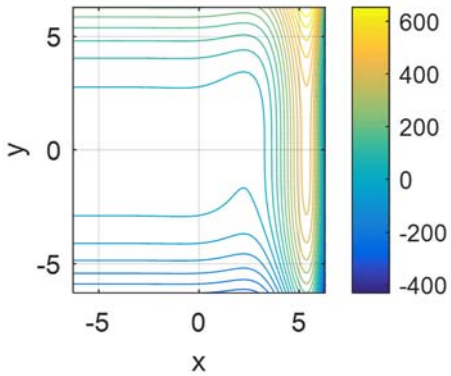


b) instrucțiunea `contour`

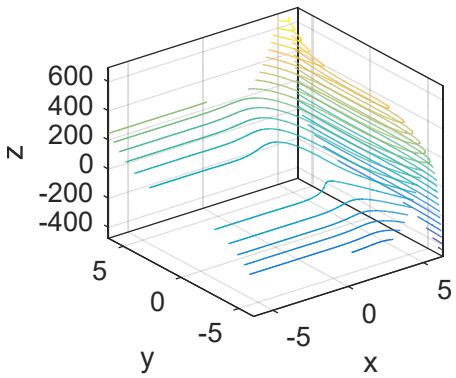


c) instrucțiunea `contour3`

Figura 5.26. Reprezentarea grafică 3D de tip linii de contur (a).



a) instrucțiunea `contour`

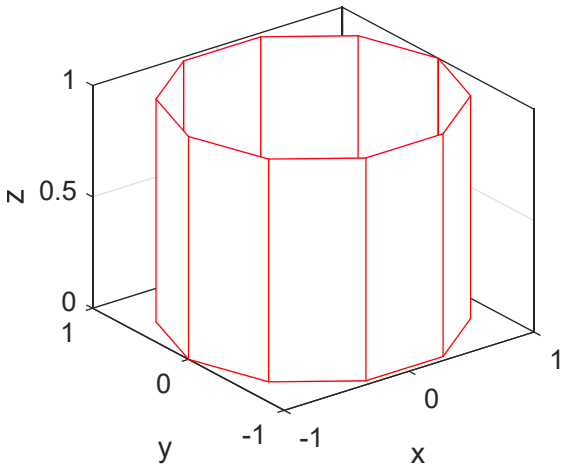


b) instrucțiunea `contour3`

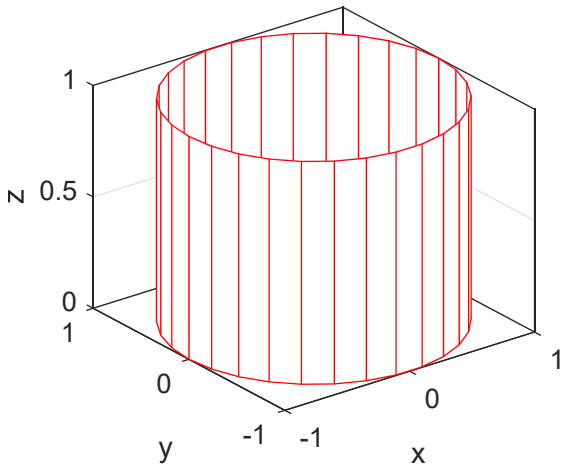
Figura 5.27. Reprezentarea grafică 3D de tip linii de contur (b).

Problema 5.9

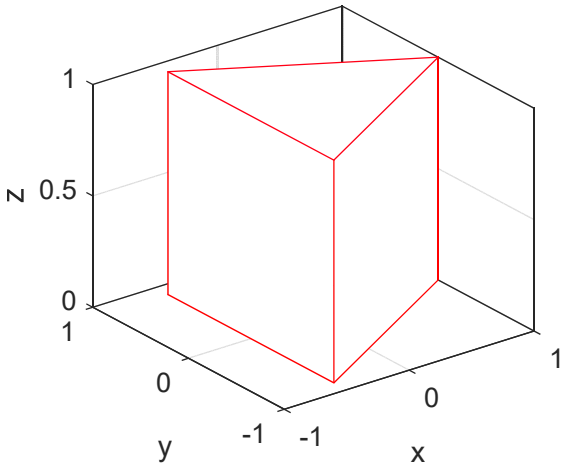
Să se reprezinte grafic cilindrul având raza R=1. Să se analizeze comparativ, influența numărului de puncte de pe circumferința cilindrului, n={10, 30}. Să se reprezinte figurile corespunzătoare valorilor n={3, 4, 6}.



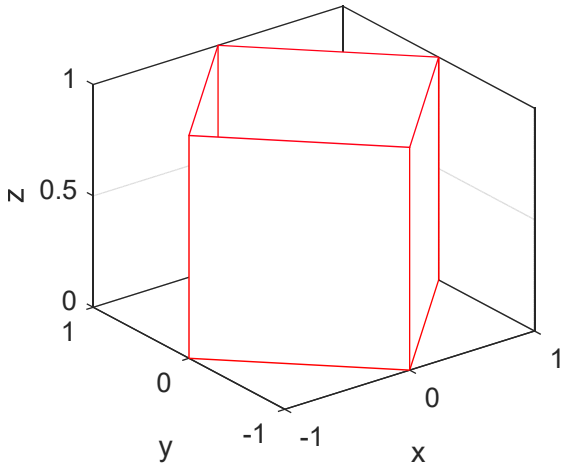
a) n=10



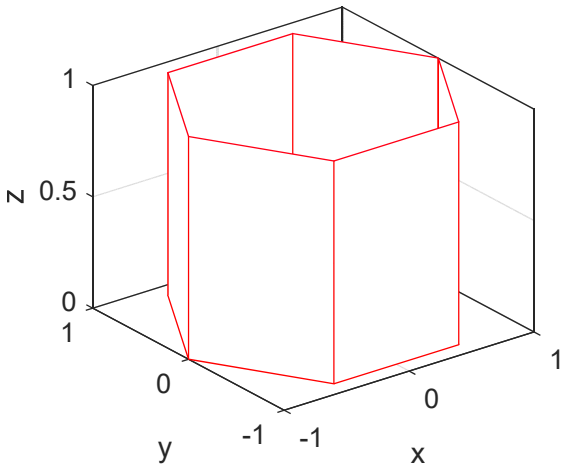
b) n=30



c) n=3



d) n=4



e) n=6

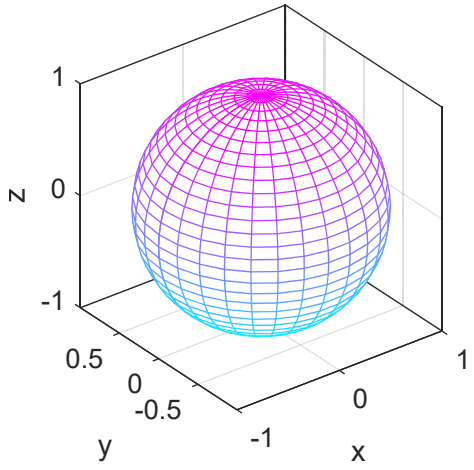
```
R=1;n=30;
[X,Y,Z]=cylinder(R,n);
mesh(X,Y,Z);
grid on;
box on;
xlabel('x');
ylabel('y');
zlabel('z');
```

f) fișierul `script`

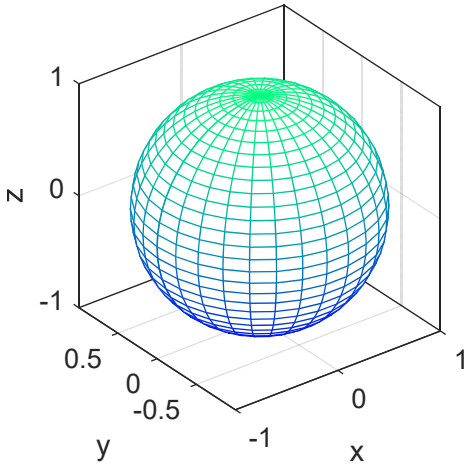
Figura 5.28. Reprezentarea grafică a cilindrului.

Problema 5.10

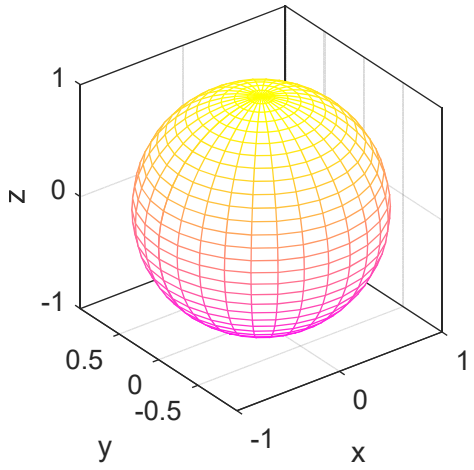
Să se reprezinte grafic sfera având raza R=1 și 30x30 puncte dispuse pe suprafața sferei (n=30). Să se analizeze comparative efectul următoarelor mape de culoare: `cool`, `winter`, `spring`, `summer`, `autumn`.



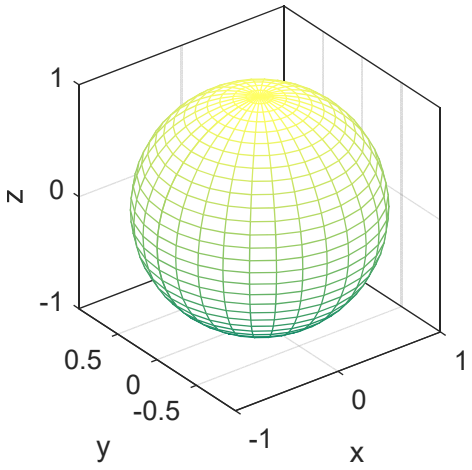
a) colormap cool



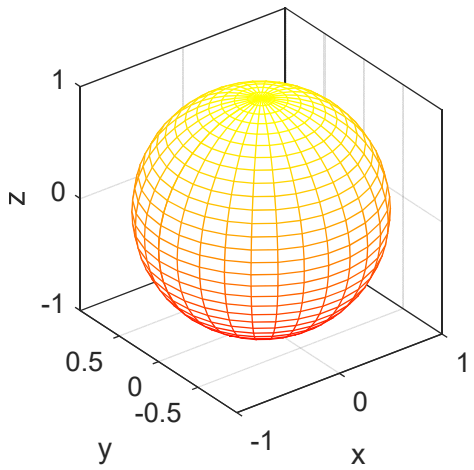
b) colormap winter



c) colormap spring



d) colormap summer



e) colormap sutumn

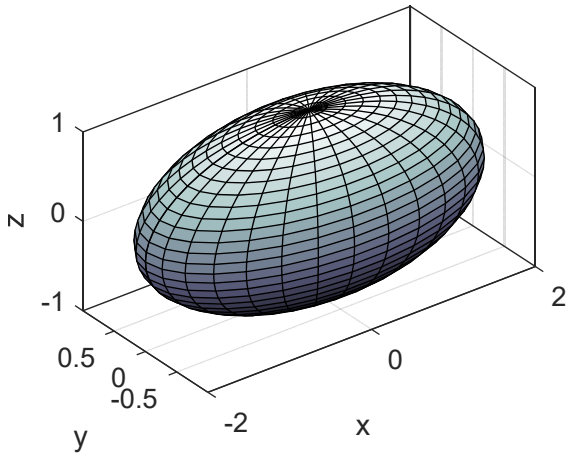
```
n=30;
[X,Y,Z]=sphere(n);
mesh(X,Y,Z);
grid on;
box on;
axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
colormap cool;
```

f) fișierul `script`

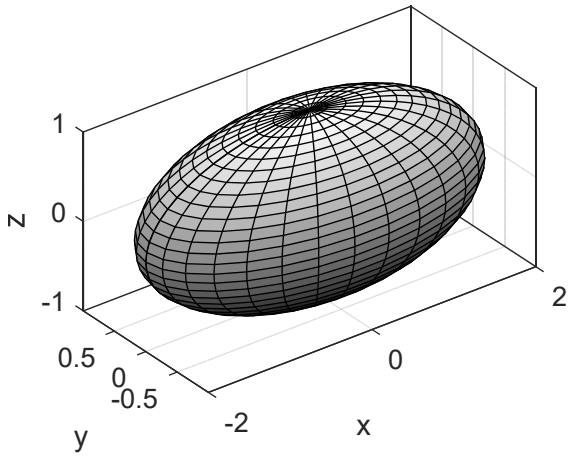
Figura 5.29. Reprezentarea grafică a sferei.

Problema 5.11

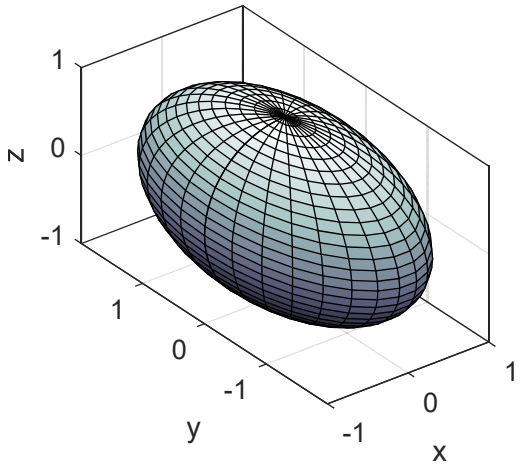
Să se reprezinte grafic elipsoidul cu centrul în originea sistemului de coordonate având semiaxele (2,1,1). Să se utilizeze mapele de culoare `bone` și `gray`. Să se analizeze comparativ elipsoizii având semiaxele (1,2,1), respectiv (1,1,2). Să se reprezinte grafic elipsoidul având semiaxele (1,1,1).



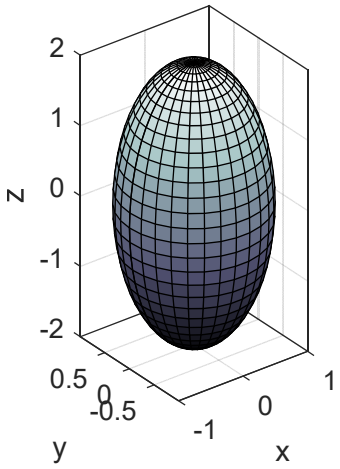
a) semiaxele (2,1,1), colormap `bone`



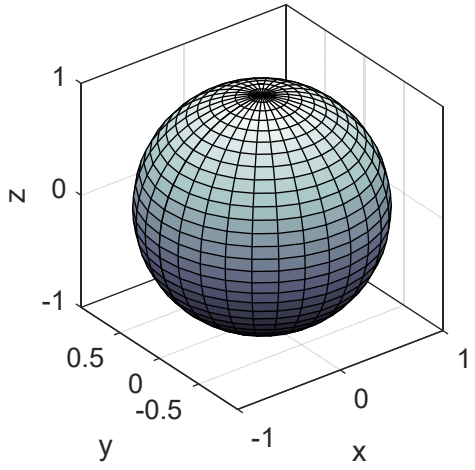
b) semiaxele (2,1,1), colormap `gray`



c) semiaxele (1,2,1)



d) semiaxele (1,1,2)



e) semiaxele (1,1,1)

```
xc=0;yc=0;zc=0;
xr=2;yr=1;zr=1;
n=30;
[X,Y,Z]=ellipsoid(xc,yc,zc,
xr,yr,zr,n);
surf(X,Y,Z);
grid on;
box on;
axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
colormap bone;
```

f) fișierul `script`

Figura 5.30. Reprezentarea grafică a elipsoidului.

BIBLIOGRAFIE

1. Anton R., Probleme de mecanică. E.D.P., București, 1978.
2. Chiriță S., Probleme de matematici superioare. EDP, București, 1989.
3. Cristea V., ș.a., Tehnici de programare. Ed. Teora, București, 1993.
4. Curteanu S., Inițiere în MATLAB. Ed. Polirom, Iași, 2008.
5. Davidescu A., Analiza și procesarea datelor în MATLAB. Ed. Politehnica, Timișoara, 2013.
6. Dodun O., Calcul numeric asistat. Teorie și aplicații în MATLAB. Ed. Performantica, Iași, 2013.
7. Dodun O., Aplicații în MATLAB. Ed. Performantica, Iași, 2014.
8. Florea J., ș.a., Mecanica fluidelor și mașini hidropneumatice. Probleme., E.D.P., București, 1982.
9. Gafițanu M., ș.a., Organe de mașini, Vol. I, II. Ed. Tehnică, București, 1983.
10. Ghinea M., Fireţeanu V., MATLAB. Calcul numeric, grafică, aplicații. Editura Teora, Bucureşti, 2003.
11. Iacob. C., ș.a., Matematici clasice și moderne, Vol. II. Ed. Tehnică, București, 1979.
12. Kilyeni Șt., Metode numerice. Algoritme. Programe de calcul. Aplicații în energetică. Ed. Orizonturi Universitare, Timișoara, 2004.
13. Lăzăroiu Gh., Sisteme de programare pentru modelare și simulare. Ed. Politehnica Press, București, 2005.
14. MathWorks, MATLAB, Primer, 2017.
15. MathWorks, MATLAB, Desktop Tools and Development Environment, 2017.
16. MathWorks, MATLAB, Graphics, 2017.
17. MathWorks, MATLAB, Mathematics, 2017.
18. Mathworks, MATLAB, Programming Fundamentals, 2017.
19. Mathworks, MATLAB, Function Reference, 2017.
20. Mocanu M., Marian Gh., Bădică C., Bădică C., 333 probleme de programare. Ed. Teora, București, 1993.
21. Modică Gh., Probleme de funcții speciale. EDP, București, 1988.
22. Muhs, D., ș.a., Roloff/Matek, Organe de mașini, Vol. I, II. Ed. MATRIX Rom, București, 2008.
23. Năslău P., ș.a., Matematici asistate de calculator. Ed. Politehnica, Timișoara, 2007.
24. Slabu Gh., Slabu V., Exerciții rezolvate și erori simulate în TURBO C++. Ed. Tehnica-Info, Chișinău, 2006.
25. Șabac I.Gh., Matematici speciale. EDP, București, 1965.
26. Turtoiu F., Probleme de trigonometrie. Ed. Tehnică, București, 1979.
27. Voinea R., Voiculescu D., Ceauşu V., Mecanica. E.D.P., București, 1983.
28. Zahariea D., MATLAB. Calcul numeric și simbolic. Editura PIM, Iași, 2014.
29. Zahariea D., Simularea sistemelor fizice în MATLAB. Ed. PIM, Iași, 2010.

CUPRINS