Lucrarea 1

Introducere în GNU Octave / MATLAB și Phython

Scopul lucrării este de a prezenta principiile de bază privind programarea în mediul GNU Octave / MATLAB și în Phython

1. Noțiuni teoretice

1.1 GNU Octave / MATLAB

Numele Matlab este o prescurtare a cuvintelor "matrix laboratory". Aceasta deoarece, inițial programul a fost destinat pentru calculul cu matrici. Limbajul a evoluat si a devenit un standard în universități când este vorba de cursuri introductive sau avansate de matematică sau inginerie. Funcțiile specifice unui anumit domeniu sunt grupate în colecții de funcții sau "toolboxes".

MATLAB (MATrix LABoratory) este un pachet de programe de înalta performanță dedicat calculului numeric și a reprezentărilor grafice. SIMULINK-ul este parte integrantă a acestui pachet soft. Simulink permite modelarea, simularea și analiza dinamică a sistemelor.

GNU Octave reprezintă o alternativă la popularul mediu de programare MATLAB. Sintaxele celor două programe sunt în mare parte identice, iar formatul fișierelor salvate este același (fișiere cu extensia .m). Avantajul GNU Octave este că acesta este oferit în mod gratuit sub licență publică. Un neajuns al acestui program este lipsa unui echivalent pentru Simulink.

Fereastra principală a programului permite accesul direct la interpretorul de comenzi. Acesta este un instrument care execută o secvență de cod linie cu linie. Secvența de cod poate fi introdusă direct de la tastatură, iar după fiecare linie se apasă tasta Enter sau poate fi scrisă într-un fișier de tip text, care se salvează cu extensia ".M" si se execută prin simpla scriere a numelui fișierului.

Limbajul MATLAB respectă principiile programării structurale, astfel că există o foarte mare asemănare între sintaxa si structurile sale cu cea a limbajului C.

1.2 Phython

Python este un limbaj de programare orientat pe obiecte ce poate fi comparat cu alte alte limbaje de programare, ca de exemplu Ruby, Perl sau Java. Python a fost gândit cu o sintaxă elegantă și ușor de folosit.

În prezent este printre cele mai populare limbaje de programare grație versatilității sale. Pe lângă biblioteca standard bogată, este posibil ca aceasta să fie extinsă prin intermediul pachetelor specializate. De la lansare, Python a cunoscut mai multe variante oficiale. Cea mai recentă, Python 3.0, a fost lansată în 2008 și nu este compatibilă în totalitate cu versiunile anterioare.

1.3 Variabile scalare

Declararea și efectuarea operațiilor cu scalari este exemplificată mai jos. În cazul Python trebuie folosită o funcție suplimentară pentru vizualizarea rezultatului ultimei operații.

GNU Octave / MATLAB

```
x=2;
y=3;
z=x+y
```

Python

```
x=2
y=3
z=x+y
print(z) #afisare
```

1.4 Variabile predefinite

Există o serie de variabile și constante predefinite utilizate des în matematică și inginerie care sunt deja definite în cele două limbaje.

GNU Octave / MATLAB

```
true #variabila booleana adevarat
false #variabila booleana fals

pi #constanta pi
e #constanta e

inf #reprezentare pentru infinit
```

Python

```
import math #importa pachet

True #variabila booleana adevarat
False #variabila booleana fals

print(math.pi) #constanta pi
print(math.e) #constanta e

print(math.inf) #reprezentare pentru
infinit
```

Observație. Pentru exemplul Python se observă apelarea unor variabile din pachetul math. Secțiunea următoare conține explicații suplimentare privind folosirea pachetelor în programe.

1.5 Variabile vectoriale (vectori)

Declararea vectorilor se face astfel:

GNU Octave / MATLAB

```
x=[1 2 3]
y=[4 5 6]
```

Python

```
import numpy as np

x = np.array([1,2,3])
y = np.array([4,5,6])

print(x)
print(y)
```

În cazul Phython este necesară utilizarea funcției array din pachetul numpy. Prima linie de cod din exemplu importă pachetul numpy și definește *np* ca fiind prescurtarea folosită pentru a apela funcții din acesta în program. Evident se poate defini orice prescurtare se dorește, dar este de reținut că pachetele des utilizate în aplicațiile Python (ex: numpy) au prescurtări deja încetățenite în rândul dezvoltatorilor. Din acest motiv este recomandată folosirea prescurtărilor standard.

Selectarea specifică a elementelor dintr-un vector diferă între cele două limbaje de programare:

GNU Octave / MATLAB

```
x(1) #returneaza primul element al vectorului x
y(3) #returneaza al treilea element al vectorului y
```

Python

```
print(x[0]) #returneaza primul element al vectorului x
print(y[2]) #returneaza al treilea element al vectorului y
```

Pe lângă ușoara diferență de sintaxă trebuie reținut că în **GNU Octave / MATLAB** numerotarea elementelor dintr-un vector se face **pornind de la 1**, în timp ce în **Python** numerotarea **începe de la 0**.

Dacă pentru vectorii de dimensiuni reduse nu este nicio problemă ca elementele lor să fie definite individual, pentru vectorii lungi cu zeci sau sute de elemente această definiție nu mai este practică. Pentru aceștia există următorul mod de definire:

GNU Octave / MATLAB

```
x=[1:0.5:50] #vector cu primul element 1, ultimul element 50 si pas de 0.5 (99 de elemente)
```

Python

```
import numpy as np

x = np.arange(1,50,0.5) #vector cu primul element 1, ultimul element 50 si pas de 0.5 (98 de elemente)

print(x)
```

Observație. Cele două limbaje tratează diferit definiția ultimului element din vector. În cazul GNU Octave / MATLAB ultimul element este inclus în vector, în timp ce în Python nu.

În ambele limbaje pasul implicit este de 1. Din acest motiv, dacă pasul dorit este 1, acesta nu mai trebuie menționat la definirea variabilei:

GNU Octave / MATLAB

```
x=[1:50] #vector cu primul element 1, ultimul element 50 si pas implicit de 1 (50 de elemente)
```

Python

```
import numpy as np

x = np.arange(1,50) #vector cu primul element 1, ultimul element 50 si pas implicit de 1 (49 de elemente)

print(x)
```

Modificarea vectorilor deja definiți se realizează diferit în funcție de limbajul folosit. În cazul GNU Octave / MATLAB există o singură instrucțiune. În cazul Python există mai multe instrucțiuni ce pot fi utilizate.

GNU Octave / MATLAB

```
x(51:100) = [50:-1:1] #am definit elementele 51-100 ale vectorului in completarea celor deja existente
```

Python

```
x=np.append(x,(20,58,9))#adauga la finalul vectorului x elementele definite in instructiune
print(x)

x=np.insert(x,9,(44,84,75))#adauga elementele din instructiune dupa al 9-lea element din vector
print(x)
```

```
x[3]=7 #modifica valoarea elementului
3
print(x)

x=np.delete(x,4)#sterge elementul 4
print(x)

y=([87,54,12,65])
z=np.concatenate((x,y))   #lipeste   2
vectori
print(z)
```

Pentru definirea unui vector cu un anumit număr de elemente aflate la distanță egală unul față de celălalt se folosesc comenzile:

GNU Octave / MATLAB

```
y=linspace(0,50,25) #vector de 25 de
elemente cu primul element 0 si cu
ultimul element 50
```

Python

```
import numpy as np

x = np.linspace(0,50,25) #vector de 25
de elemente cu primul element 0 si cu
ultimul element 50
print(x)
```

Numărul implicit de elemente dintr-un vector astfel definit este 100 în GNU Octave / MATLAB și 50 în Python. Prin urmare, dacă se dorește definirea unor vectori de aceste dimensiuni, este suficient să se menționeze doar primul și ultimul element din vector.

GNU Octave / MATLAB

```
y=linspace(0,200) #vector de 100 de
elemente cu primul element 0 si cu
ultimul element 200
```

Python

```
import numpy as np

x = np.linspace(0,200) #vector de 50
de elemente cu primul element 0 si cu
ultimul element 200
print(x)
```

În ceea ce privește operațiile matematice ce pot fi efectuate asupra vectorilor:

GNU Octave / MATLAB

```
a=3; #definire scalar
x=[5 3 9]; #definire vector
y=[4 7 5];

z=x+y #suma a doi vectori

u=a*x #produsul unui vector cu un
scalar

v=dot(x,y) #produsul scalar al
vectorilor x si y

w=cross(x,y) #produsul vectorial al
vectorilor x si y
```

Python

```
import numpy as np

a=3 #definire scalar
x=np.array([5,3,9]) #definire vector
y=np.array([4,7,5])

z=x+y #suma a doi vectori
print(z)

u=a*x  #produsul  unui  vector  cu  un
scalar
print(u)

v=np.dot(x,y)  #produsul  scalar  al
vectorilor x si y
print(v)
```

```
w=np.cross(x,y)  #produsul  vectorial
al vectorilor x si y
print(w)
```

1.6 Variabile matriceale (matrici)

Deoarece vectorii sunt doar un caz particular al matricilor, definiția și operațiile efectuate asupra lor sunt similare.

GNU Octave / MATLAB

```
A=[1 2 3; 4 5 6; 7 8 9] #definirea
unei matrici cu 3 linii si 3 coloane
```

Python

```
import numpy as np

A =
np.array([[1,2,3],[4,5,6],[7,8,9]])
#definirea unei matrici cu 3 linii si
3 coloane

print(A)
```

O parte din operațiile matematice ce pot fi efectuate cu matrici sunt:

GNU Octave / MATLAB

```
A=[1 2 3; 4 5 6; 7 8 9]; #definirea
unei matrici cu 3 linii si 3 coloane.
s=5; #definire scalar

B=A'  #matricea  B  este  egala  cu
transpusa matricei A.

C=A+B #suma a doua matrici

D=A*B #produsul a doua matrici

E=s*A #produsul unei matrice cu un
scalar

x=det(A) #x este egal cu determinantul
matricei A
```

Python

```
import numpy as np
from scipy import linalg

A=np.array([[1,2,3],[4,5,6],[7,8,9]])
#definirea unei matrici cu 3 linii si
3 coloane
s=5 #definire scalar

B=np.transpose(A)  #matricea  B  este
egala cu transpusa matricei A
print(B)

C=A+B #suma a doua matrici
print(C)

D=np.matmul(A,B)  #produsul  a  doua
matrici
print(D)

E=s*A #produsul unei matrice cu un
scalar
print(E)

x=linalg.det(A)  #x  este  egal  cu
determinantul matricei A
print(x)
```

Observație. Din cauza modului în care este implementat pachetul numpy în locul uni determinantului egal cu 0 se va afișa o valoare foarte mică (apropiată de zero). Din această cauză este recomandată folosirea funcției de calculare a determinantului din **pachetul scipy, sub-pachetul linalg**.

Există comenzi și pentru definirea matricilor speciale:

GNU Octave / MATLAB

```
A=ones(3,4) #matrice cu 3 randuri si
4 coloane cu toate elementele 1

B=zeros(4,5) #matrice cu 4 randuri si
5 coloane cu toate elementele 0

I=eye(3,3) #matrice unitate cu 3
randuri si coloane

v=[1 -9 5 -6 12] #definire vector
D=diag(v) #matrice diagonala cu
elementele vectorului v pe diagonala
principala
```

Python

```
import numpy as np

A=np.ones((3,4)) #matrice cu 3 randuri
si 4 coloane cu toate elementele 1
print(A)

B=np.zeros((4,5)) #matrice cu 4
randuri si 5 coloane cu toate
elementele 0
print(B)
I=np.eye(3,3) #matrice unitate cu 3
randuri si coloane
print(I)

v=([1,-9,5,-6,12]) #definire vector
D=np.diag(v) #matrice diagonala cu
elementele vectorului v pe diagonala
principala
print(D)
```

1.7 Reprezentări grafice

Ambele limbaje prezentate până acum sunt capabile să genereze o sumedenie de reprezentări grafice. Cea mai utilizată este reprezentarea pe două axe Ox și Oy. Ca exemplu se ia reprezentarea funcției sinus.

GNU Octave / MATLAB

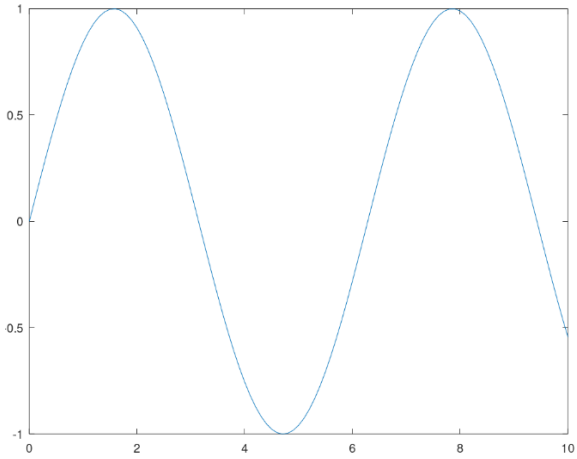
```
t=[0:0.1:10]; #definire vector timp
x=sin(t); #definirea functiei x(t)

plot(t,x) #generarea unui grafic al
funcției x(t)
```

Python

```
import numpy as np
import matplotlib.pyplot as plt

t=np.arange(0,10,0.1) #definire
vector timp
x=np.sin(t) #definirea functiei x(t)

plt.plot(t,x) #generarea unui grafic
al funcției x(t)
plt.show() #afisarea graficului
```

În cazul GNU Octave / MATLAB, graficele generate pot fi salvate prin comanda saveas(gcf,'nume_fișier.extensie'). În cazul Python salvarea graficelor se poate face prin intermediul butonului de salvare din fereastra ce conține graficul.

GNU Octave / MATLAB



Python

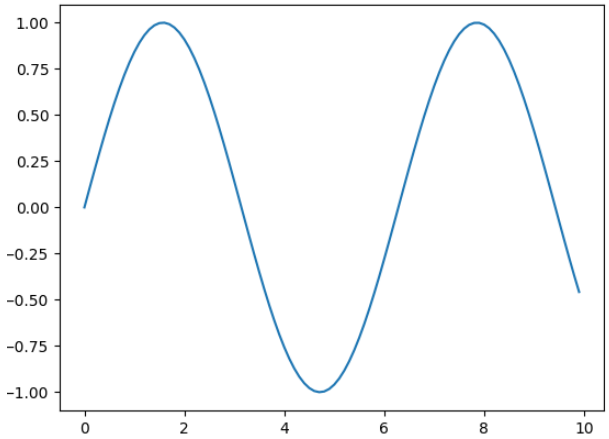


Figura 1. Reprezentarea funcției sinus

Dacă se dorește reprezentarea mai multor curbe pe același grafic, trebuie folosită funcția hold on în GNU Octave / MATLAB. În cazul Python nu este nevoie de alte comenzi suplimentare.

GNU Octave / MATLAB

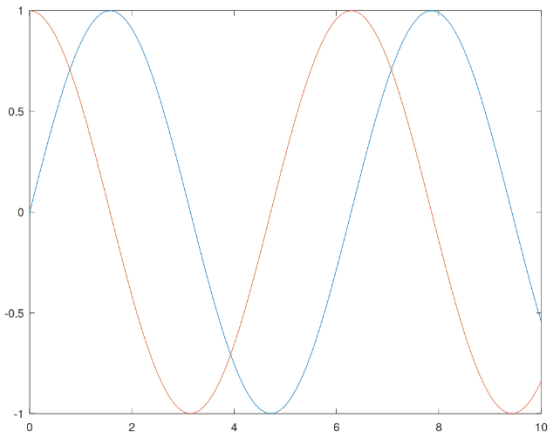
```
t=[0:0.1:10]; #definire vector timp
x=sin(t); #definirea functiei x(t)
y=cos(t); #definirea functiei x(t)
plot(t,x)  #generarea unui grafic al
functiei x(t)

hold on; #mentine deschisa fereastra
cu graficul reprezentat
plot(t,y) #adauga graficului existent
reprezentarea functiei y(t)
```

Python

```
import numpy as np
import matplotlib.pyplot as plt
t=np.arange(0,10,0.1)        #definire
vector timp
x=np.sin(t) #definirea functiei x(t)
y=np.cos(t) #definirea functiei y(t)

plt.plot(t,x) #generarea unui grafic
al funcției x(t)
plt.plot(t,y)    #adaugarea    curbei
pentru functia y(t)

plt.show() #afisarea graficului
```

Graficele obținute sunt prezentate mai jos:

GNU Octave / MATLAB



Python

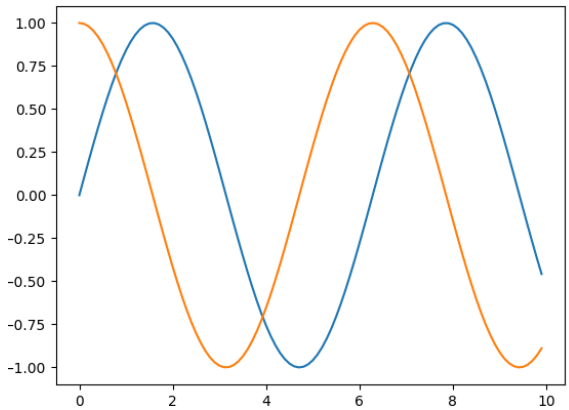


Figura 2. Grafice cu două funcții reprezentate

Graficelor generate li-se pot adăuga diferite elemente suplimentare:

GNU Octave / MATLAB

```
title('Reprezentare grafica')
#Titlul graficului
xlabel('Timp (s)') #Text axa Ox
ylabel('f(t)') #Text axa Oy

legend({'sin(t)','cos(t)'},'font
size',14,'location','southwest')
#adauga legenda cu font de
marime 14 in stanga jos
```

Python

```
plt.title("Reprezentare grafica")
#Titlul graficului
plt.xlabel("Timp(s)") #Text axa Ox
plt.ylabel("f(t)") #Text axa Oy

plt.legend(["sin(t)","cos(t)"],fon
tsize=14,loc="lower left") #adauga
legenda cu font de marime 14 in
stanga jos
```

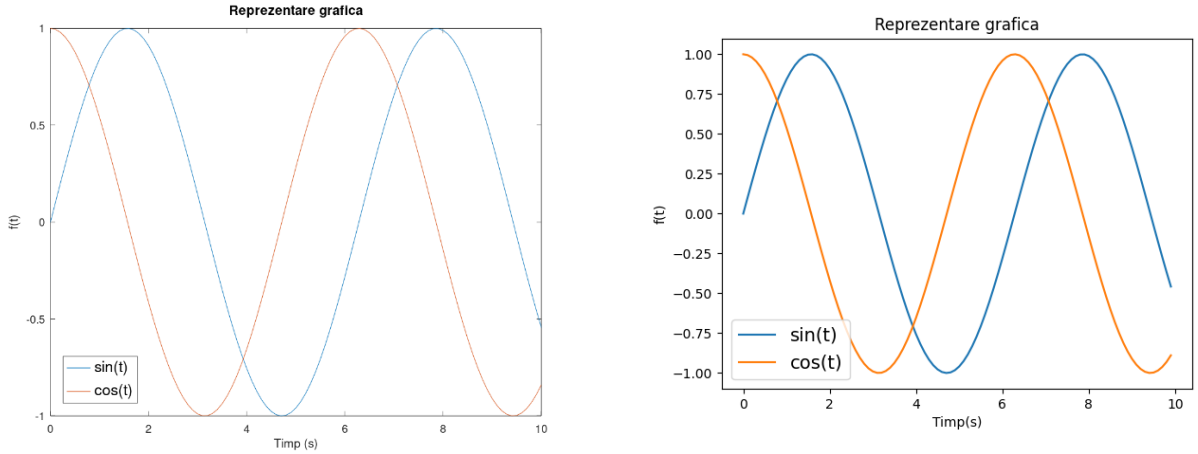


Figura 3. Graficele completate cu elementele suplimentare

Aspectul curbelor reprezentate poate fi și el modificat.

GNU Octave / MATLAB

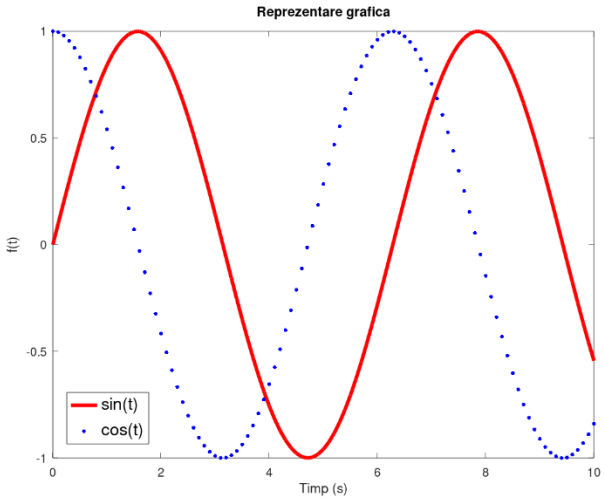
```
t=[0:0.1:10]; #definire vector
timp
x=sin(t); #definirea functiei
x(t)
y=cos(t); #definirea functiei
x(t)

plot(t,x,'linewidth',3,'color','
red') #generarea unui grafic al
functiei x(t) cu o grosime de 3
si de culoare rosie

hold on;

plot(t,y,'.','color','blue','mar
kersize',10) #generarea unui
grafic al functiei y(t) de tip
linie punctata albastra cu
marcatori de dimensiune 10

title('Reprezentare grafica')
xlabel('Timp (s)')
```

Python

```
import numpy as np
import matplotlib.pyplot as plt

t=np.arange(0,10,0.1) #definire
vector timp
x=np.sin(t) #definirea functiei
x(t)
y=np.cos(t) #definirea functiei
y(t)

plt.plot(t,x,linewidth=3,color='re
d') #generarea unui grafic al
functiei x(t) cu o grosime de 3 si
de culoare rosie
plt.plot(t,y,'.',color='blue')
#generarea unui grafic al functiei
y(t) de tip linie punctata
albastra cu marcatori de
dimensiune 10

plt.title("Reprezentare grafica")
plt.xlabel("Timp(s)")
```

```
ylabel('f(t)')
legend({'sin(t)','cos(t)'},'font
size',14,'location','southwest')
```

```
plt.ylabel("f(t)")
plt.legend(["sin(t)","cos(t)"],
fontsize=14,loc="lower left")

plt.show()
```



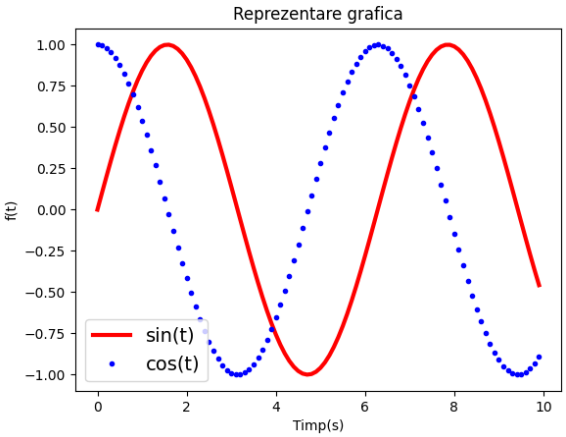


Figura 4. Graficele generate în urma rulării secvențelor de cod anterioare

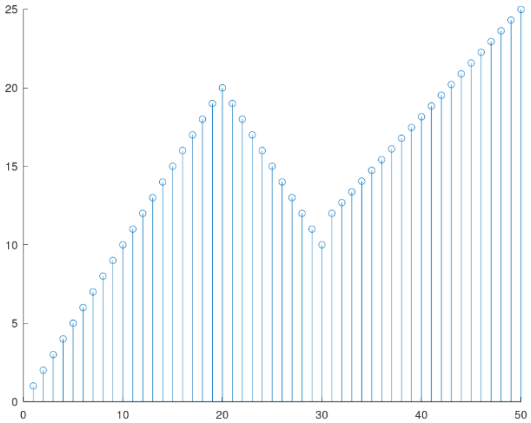
Un alt tip de grafic, potrivit pentru reprezentarea datelor discrete, poate fi generat prin comanda stem(x).

GNU Octave / MATLAB

```
v=[1:20];
v(21:30)=[19:-1:10];
v(31:50)=linspace(12,25,20);
#definire vector

stem(v)
```

Python

```
import numpy as np
import matplotlib.pyplot as plt

v1=np.arange(1,21)
v2=np.arange(19,9,-1)
v3=np.linspace(12,25,20)
v=np.concatenate((v1,v2,v3))
#definire vector

plt.stem(v)
plt.show()
```



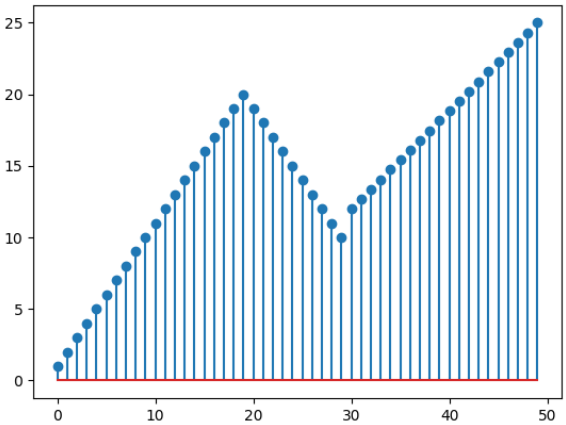


Figura 5. Graficele generate prin funcția stem

2. Modul de lucru

Se parcurg următoarele cerințe, notându-se comanda executată și rezultatul acesteia. Pentru afișarea rezultatelor folosiți funcția print()

Variabile și vectori

1) Definiți variabilele nn (numărul de litere al numelui), pn (numărul de litere al celui mai scurt prenume)
2) Declarați un vector x cu nn elemente și un vector y cu pn elemente.
3) Declarați un vector z cu primul element nn, ultimul element nn+pn și pn elemente.

Matrici

4) Definiți o matrice A cu nn rânduri și pn coloane.
5) Definiți o matrice B cu toate elementele 1 cu pn linii și coloane.
6) Calculați matricea C ca fiind înmulțirea matricilor A și B.
7) Definiți o matrice D cu 3 linii și coloane. Elementele matricei sunt: nn, pn-20, nn*0.42, nn+pn, pn*2, 1, nn-pn, nn/pn și 3.
8) Calculați variabila d ca fiind determinantul matricei D. (Folosiți sub-pachetul linalg din pachetul scipy)

Reprezentări grafice

9) Definiți un vector t cu primul element 0, ultimul element 20 și pas de nn/100.
10) Definiți funcția s ca fiind sin(t+π/nn). (Folosiți pachetul math pentru variabila pi)
11) Definiți funcția u ca fiind 0.1*(nn+pn)*cos(t- π/pn). (Folosiți pachetul math pentru variabila pi)
12) Reprezentați grafic funcția s(t).
13) Adăugați graficului anterior reprezentarea funcției u(t).
14) Adăugați graficului anterior un titlu și text pentru ambele axe.
15) Adăugați graficului anterior o legendă cu mărimea fontului de 14 poziționat astfel încât să se suprapună cât mai puțin cu funcțiile deja reprezentate.
16) Reprezentați pe un grafic discret vectorii z, x și y (respectați această ordine)