LAB-TS-2. BAZELE MATLAB ȘI SIMULINK. MODELAREA ȘI SIMULAREA SISTEMELOR. CONEXIURI De SISTEMELO

A. OBIECTIVELE LUCRĂRII. 1. Familiarizarea cu elementele de bază ale Matlab și Simulink.
2. Modelarea și simularea sistemelor în mediul Matlab/Simulink.

B. Bazele MATLAB [1]

Matlab este un software matematic, produs de firma The MathWorks, Inc. (http://www.mathworks.com), destinat calculului numeric, programării, modelării și simulării numerice, prelucrărilor de date și reprezentărilor grafice în știință și inginerie. Există un tool de bază și o colecție de scripturi și toolbox-uri disponibile. Funcționalitățile mediului Matlab îl fac potrivit pentru modelare, simulare, calcule avansate și proiectarea sistemelor de reglare automată. Obiectele de bază cu care puteți interacționa în Matlab sunt:

- instrucțiuni de control și variabile,
- matrice,
- elemente grafice și
- scripturi.

După pornirea Matlab, în cadrul liniei de comandă se va evidenția prompterul reprezentat de simbolul ">>".

1. Instrucțiuni de control și variabile

Matlab folosește caracterul de atribuire "=" pentru a atribui o expresie unei variabile:

```
>>variabilă=expresie
```

De exemplu, instrucțiunea prin care este introdusă o matrice cu două linii și două coloane are expresia:

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

A =
     1     2
     3     4
```

De reținut că Matlab este case sensitive. După apăsarea tastei ENTER, este evaluată fiecare instrucțiune. Dacă instrucțiunile au fost corecte, ele se execută imediat; în caz contrar se afișează mesaje de eroare. Operatorul punct și virgulă (;) poate fi folosit ca și în expresia anterioară pentru a separa liniile din matrice sau pentru a nu afișa rezultatul în linia de comandă.

Operatorii matematici folosiți în Matlab sunt: adunarea (+), scăderea (–), înmulțirea (*), împărțirea la dreapta (/), împărțirea la stânga (\) și ridicarea la putere (^). Acești operatori pot fi folosiți în cazul diferitelor tipuri de obiecte (după cum va fi exemplificat în cele ce urmează).

Câteva dintre variabilele predefinite sunt *pi*, *Inf*, *NaN*, *i* și *j*. *NaN* este pentru not-a-number, iar acesta rezultă din operațiuni nedefinite. *Inf* reprezintă infinitul +∞, în timp ce *pi* reprezintă numărul π. Variabilele $i = \sqrt{-1}$ și *j* sunt folosite pentru a reprezenta numerele complexe. Comanda "`who`" ilustrează variabilele care sunt stocate în spațiul de lucru (workspace):

```
>> who
Your variables are:
A
```

Comanda "`whos`" oferă informații suplimentare despre variabilele workspace-ului:

```
>> whos
Name        Size              Bytes  Class     Attributes

A           2x2                  32  double
```

În tabelul 1 sunt prezentate cele mai frecvent utilizate funcții predefinite în Matlab.

Tabelul 1. Funcții predefinite frecvent utilizate în Matlab

Funcție	Utilizare
sin(x)	Sinusul elementelor vectorului x
cos(x)	Cosinusul elementelor vectorului x
asin(x)	Arcsinusul elementelor vectorului x
acos(x)	Arccosinusul elementelor vectorului x
tan(x)	Tangenta elementelor vectorului x
atan(x)	Arctangenta elementelor vectorului x
atan2(x,y)	Arctangentă în patru cadrane a elementelor reale ale vectorilor x și y
abs(x)	Valoarea absolută a elementelor vectorului x
sqrt(x)	Rădăcina pătrată a elementelor lui x
imag(x)	Partea imaginară a elementelor vectorului x
real(x)	Partea reală a elementelor vectorului x
conj(x)	Conjugata complexă al elementelor vectorului x
log(x)	Logaritmul natural al elementelor vectorului x
log10(x)	Logaritmul zecimal al elementelor vectorului x
exp(x)	Exponențiala elementelor vectorului x

Formatul implicit de afișare a numerelor în Matlab este în virgulă fixă cu 5 cifre. Alte formate sunt în virgulă fixă cu 15 cifre, în virgulă mobilă cu 5 cifre și în virgulă mobilă cu 15 cifre, conform exemplului de mai jos:

```
>> pi

ans =

    3.1416

>> format long; pi

ans =
```

```
   3.141592653589793

>> format short e; pi

ans =

  3.1416e+000

>> format long e; pi

ans =

    3.141592653589793e+000
```

Formatul de afișare **nu afectează** calculele care sunt efectuate cu precizie dublă.

2. Instrucțiuni de control și variabile

- Memoria pentru matrice este alocată automat, deci nu trebuie să fie prespecificată.
- Operatorii matematici de bază sunt supraîncărcați pentru matrice. Împărțirea este un caz special de înmulțire matriceală, astfel încât împărțirea la dreapta A/B înseamnă A*inv(B), în care inv se notează inversa matricei. Împărțirea la stânga A\B înseamnă inv(A)*B.
- **Este foarte important ca toate operațiile cu matrice să fie compatibile dimensional**.
- Operațiile cu tablouri (vectori sau matrice) sunt similare operațiilor matricele din cazul calculului element cu element.
- Operația de transpunere matriceala este realizată folosind caracterul apostrof " ' "

În operațiile vectoriale și matriceale este posibil calculul element cu element utilizând operatorii matematici corespunzători precedați de caracterul punct ".". De exemplu, doi vectori coloană **x** și **y** având același număr de elemente pot fi înmulțiți element cu element folosind "**x.*y**".

Notația două puncte este un instrument important în Matlab care permite generarea vectorilor de tip linie/coloană care să conțină elemente numerice începând cu o valoare dată x_i până la o valoare finală x_f cu incrementul dx. Acest lucru este ilustrat în exemplul următor de cod:

```
>>  x = [-1:0.2:1]';y=x.*sin(x);
>> [x y]

ans =

   -1.0000    0.8415
   -0.8000    0.5739
   -0.6000    0.3388
   -0.4000    0.1558
   -0.2000    0.0397
         0         0
    0.2000    0.0397
    0.4000    0.1558
    0.6000    0.3388
    0.8000    0.5739
    1.0000    0.8415
```

3. Elemente de grafică

În tabelul 2 sunt prezentate cele mai des utilizate comenzi pentru generarea graficelor.

Tabelul 2. Comenzi grafice frecvent utilizate în Matlab.

Funcția	Utilizare
plot(x,y)	grafice în coordonate X-Y carteziene (liniare)
semilogx(x,y)	grafice în coordonate X-Y semilogaritmice, axa X este în $\log_{10}$
semilogy(x,y)	grafice în coordonate X-Y semilogaritmice, axa Y este în $\log_{10}$
loglog(x,y)	grafice în coordonate X-Y logaritmice (în baza 10)

În tabelul 2 sunt prezentate diverse funcții Matlab auxiliare pentru personalizarea graficelor.

Tabelul 3. Funcții Matlab auxiliare pentru reprezentări grafice.

Funcția	Utilizare
title('text')	inserarea titlului 'text' pentru reprezentarea grafică
xlabel('text')	inserarea etichetei 'text' pentru sistemul de coordonate pentru abscisă
ylabel('text')	inserarea etichetei 'text' pentru sistemul de coordonate pentru ordonată
text(p1,p2,'text','sc')	afișează "text" în cazul coordonatelor ecranului la (p1,p2), (0.0,0.0) fiind în partea stângă jos a ecranului și (1.0,1.0) fiind în partea dreaptă sus a ecranului
subplot	împărțirea ferestrei de reprezentare grafică în mai multe regiuni grafice
grid	suprapunerea unei rețele de linii pe grafic

Cea mai des utilizată comandă este funcția predefinită "**plot**". Se pot folosi mai multe argumente pentru fiecare dintre comenzile (funcțiile predefinite) de mai sus. Comanda "**subplot**" este folosită și în cazul creării subgraficelor, aspect evidențiat în exemplul următor:

```
>>subplot(121),plot(x,x,'b-',x,y,'k:'),xlabel('x'),ylabel('x
vs.x and x vs. x*sin(x)'),
>>subplot(122),plot(x,sin(x),'r'),ylabel('sin(x)'),grid
```

ce va genera graficele din fig. 1.

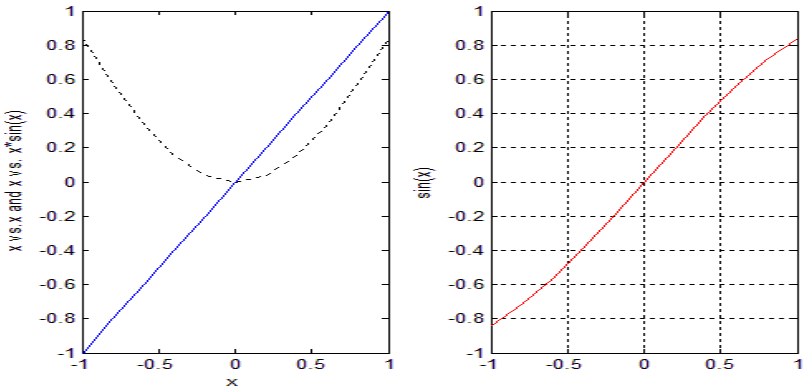


Fig. 1. Graficele funcțiilor din exemplul anterior.

Comanda "subplot(*mnp*)" împarte fereastra grafică în $m \times n$ subgrafice în timp ce valoarea întreagă pozitivă "*p*" selectează subgraficul. Diverse markere pot fi utilizate pentru graficele specificate prin "*cm*", în care "*c*" desemnează culoarea, iar "*m*" reprezintă tipul marker-ului.

4. Tipuri de fișiere în Matlab

Fișierele Matlab (fișierele .m) sunt de două tipuri:
- fișiere funcție
- fișiere script.

Scripturile (sau fișierele .m) reprezintă o colecție de comenzi într-un singur fișier și sunt de obicei create folosind editorul de fișiere .m disponibil în Matlab. Scripturile pot apela alte scripturi. Scripturile care implementează o funcție trebuie să aibă numele fișierului identic cu numele funcției. Cu toate acestea, mai multe funcții pot fi declarate în interiorul scripturilor. Scripturile pot fi bine documentate și comentate folosind simbolul "**%**". Textul comentat apare de culoare verde.

De menționat că deși Matlab este și un limbaj de programare, comenzile complexe pot fi implementate folosind instrucțiuni de decizie precum "**if-then-else**", "**for**" și "**while**" și așa mai departe.

În cele ce urmează este prezentat un exemplu de funcție:

```
function [mean,stdev] = stat(x)
%STAT Statistici interesante.
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);

%------------------------
function mean = avg(x,n)
%subfunctia medie
mean = sum(x)/n;
```

Rularea fișierului funcție se face dând valori vectorului x și scriind următoarea linie de cod în linia de comandă Matlab:

```
>> x=-1:0.1:1;
>> [mean,stdev] = stat(x)
```

Într-un fișier de tip script, variabilele au o sferă și o vizibilitate globală. Într-o funcție variabila are vizibilitate locală și există doar atunci când funcția este executată. Într-un fișier de tip funcție, poate fi definită o altă funcție numită **subfuncție**. O subfuncție nu este vizibilă în afara funcției și nu poate fi invocată în Matlab.

C. Modelarea și simularea sistemelor în Matlab.

1. Sistemul de tip masă-arc-amortizor

În fig. 2 este prezentată schema bloc a sistemului.

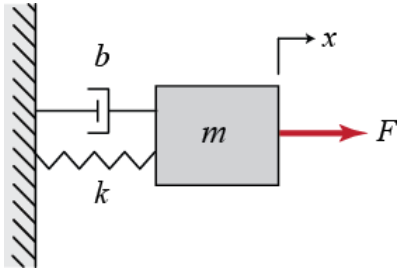


Fig. 2. Schema bloc a sistemului de tip masă-arc-amortizor [2].

Forța arcului este proporțională cu deplasarea masei, notată cu *x*, iar forța de amortizare vâscoasă este proporțională cu viteza masei și anume $v = \dot{x}$. Aceste două forțe se opun mișcării masei generate de forța de intrare *F*. Poziția $x = 0$ corespunde arcului neîntins. În fig. 3 sunt ilustrate forțele.

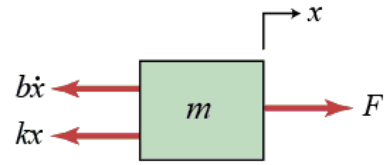


Fig. 3. Evidențierea forțelor în cadrul schemei bloc a sistemului de tip masă-arc-amortizor.

Aplicând a doua lege a lui Newton și subliniind că variabilele depind de timp (*t*) rezultă (presupunând că deplasarea inițială este nulă)

$$F(t) - b\,\dot{x}(t) - k\,x(t) = m\,\ddot{x}(t), \qquad (1.1)$$

cunoscută și sub numele de ecuația primară care descrie complet comportamentul dinamic al sistemului. Evoluția dinamicii este o funcție a forței de intrare $F(t)$ și poate fi calculată în diverse moduri. Pentru a scrie o reprezentare de stare a sistemului, se pornește de la ecuația diferențială de ordinul al doilea care trebuie transformată într-un

set de două ecuații diferențiale de ordinul întâi. Considerând poziția și viteza ca variabile de stare, modelul de stare este

$$\dot{\mathbf{x}}(t)=\begin{pmatrix}\dot{x}(t)\\ \ddot{x}(t)\end{pmatrix}=\begin{pmatrix}0 & 1\\ -k/m & -b/m\end{pmatrix}\begin{pmatrix}x(t)\\ \dot{x}(t)\end{pmatrix}+\begin{pmatrix}0\\ 1/m\end{pmatrix}F(t),$$
$$y(t)=\begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}x(t)\\ \dot{x}(t)\end{pmatrix}. \qquad (1.2)$$

Remarci:

1) Variabilele de stare corespund energiei potențiale în arc și respectiv energiei cinetice a masei. De regulă este util să se urmărească elementele de stocare a energiei din sistem pentru a alege variabilele de stare ale sistemului și pentru a deduce ordinul sistemului.
2) În ecuația ieșirii, ieșirea este de fapt deplasarea masei. Dacă pentru ieșire ne interesează reglarea vitezei masei deplasate, atunci matricea de ieșire se poate înlocui cu (0 1).

Crearea modelului sistemului în Matlab

Pentru a introduce ecuațiile de mai sus, se va utiliza un script de tip .m:

```
m = 1; % masa in kg
k = 1; % constanta arcului in N/m
b = 0.2; % constant de amortizare in Ns/m
F = 1; % forta de intrare in N

A = [0 1; -k/m -b/m];
B = [0 1/m]';
C = [1 0];
D = [0];

sys = ss(A,B,C,D)
```

Rularea scriptului va genera ieșirea următoare:

```
a =
       x1    x2
x1      0     1
x2     -1  -0.2

b =
     u1
x1    0
x2    1

c =
     x1  x2
y1    1   0

d =
     u1
y1    0

Continuous-time state-space model.
```

Aplicarea transformării Laplace ecuației de ordinul al doilea în condiții inițiale nule conduce la:

$$m\,s^2\,X(s)+b\,s\,X(s)+k\,X(s)=F(s), \qquad (1.3)$$

rezultând următoarea funcție de transfer:

$$\frac{X(s)}{F(s)}=\frac{1}{m\,s^2+b\,s+k}. \qquad (1.4)$$

Pentru a introduce modelul ca funcție de transfer, există două variante de lucru. În primul rând, variabila simbolică "s" va fi utilizată conform fișierului de tip .m:

```
s=tf('s');
sys=1/(m*s^2+b*s+k)
```

Rularea scriptului va genera ieșirea următoare:

```
sys =

          1
  ---------------
  s^2 + 0.2 s + 1

Continuous-time transfer function.
```

Pentru a asigura interfața cu Simulink, este utilă folosirea unei alte variante pentru reprezentarea funcțiilor de transfer, conform script-ului următor:

```
num = [1];
den = [m b k];
sys = tf(num,den)
```

Acest script va genera exact aceeași ieșire pentru obiectul "*sys*" cu cea din cazul în care este definită în prealabil o variabilă simbolică.

Intrarea sistemului (1.1), adică forța aplicată masei în cazul acestui exemplu, se mai numește și **funcție de forțare**. Dinamica sistemului supusă funcției de forțare zero și doar deplasării inițiale x(0) se numește **răspuns dinamic neforțat**. Răspunsul sistemului în general este format din răspunsul dinamic neforțat (liber) și din răspunsul dinamic forțat, iar acest lucru se deduce din rezolvarea ecuațiilor diferențiale ordinare. Ecuația diferențială în cazul de față este (1.1).

În cazul funcției de transfer din relația (1.4), rădăcinile numărătorului se numesc **zerouri** ale sistemului, în timp ce rădăcinile numitorului sunt **polii** sistemului. **Ecuația caracteristică** este obținută prin egalarea cu zero a polinomului de la numitor. Funcția Matlab pentru găsirea rădăcinilor unui polinom este "**roots**(*v*)", în care *v* este vectorul ce conține coeficienții polinomului în ordine descrescătoare după puterile variabilei polinomiale. Considerând rădăcinile unui polinom, polinomul poate fi reconstruit folosind funcția "**poly**(*r*)", în care *r* este vectorul rădăcinilor. Înmulțirea polinomului ce rezultă cu un alt polinom se realizează folosind funcția **conv**.

Polinomul este evaluat folosind funcția **polyval**. O detaliere completă legată de operațiile polinomiale este făcută prin tastarea comenzii "**help poly**" în cadrul liniei de comandă.

2. Un sistem electric

În acest paragraf este luat în considerare un circuit electric simplu pentru a fi modelat. Circuitul este format din trei elemente pasive: un rezistor, o bobină și un condensator. Acest circuit este cunoscut sub numele de circuit RLC cu diagrama prezentată în fig. 4.

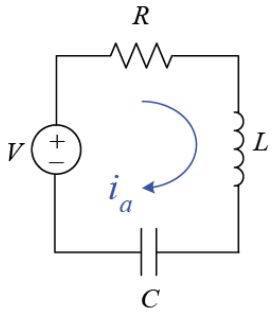


Fig. 4. Circuitul RLC.

Din legea tensiunii (a doua lege) a lui Kirchhoff și legea curenților (prima lege) a lui Kirchhoff, ecuația primară poate fi scrisă în forma:

$$V(t) - L\frac{di(t)}{dt} - R\,i(t) - \frac{1}{C}\int i(t)dt = 0. \tag{1.5}$$

În cele ce urmează este prezentă o analogie interesantă cu sistemul mecanic anterior. Sistemul este de ordinul al doilea, sarcina (integrala curentului) corespunde deplasării, inductanța corespunde masei, rezistența corespunde amortizării vâscoase, iar capacitatea electrică inversă corespunde rigidității arcului. Această analogie este foarte utilă deoarece poate fi utilizată în mod similar și în cazul altor sisteme și astfel permite un cadru general de analiză. În cazul alegerii sarcinii și curentului ca variabile de stare, reprezentarea de stare a sistemului este

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} i(t) \\ \dfrac{di(t)}{dt} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -R/L & -1/(LC) \end{pmatrix}\begin{pmatrix} q(t) \\ i(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1/L \end{pmatrix} V(t),$$
$$y(t) = \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} q(t) \\ i(t) \end{pmatrix}. \tag{1.6}$$

Funcția de transfer poate fi calculată fie prin aplicarea transformării Laplace în cazul ecuației primare (în majoritatea cazurilor condițiile inițiale sunt presupuse a fi nule), fie prin utilizarea formulei de transformare a modelului de stare în funcție de transfer:

$$\frac{I(s)}{V(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} = \frac{s}{L\,s^2 + R\,s + 1/C}. \tag{1.7}$$

3. Conversia între reprezentările de sistem

Din funcția de transfer dată, de exemplu, în [3]:

```
num = [2 1];
den = [4 3 2];
G = tf(num,den)
```

în urma execuției secvenței de mai sus, se obține

```
G =

     2 s + 1
  ---------------
  4 s^2 + 3 s + 2

Continuous-time transfer function.
```

Un model de stare poate fi extras folosind funcția predefinită *ssdata*:

```
[A,B,C,D] = ssdata(G)
```

rezultând că

```
A =
   -0.7500   -0.5000
    1.0000         0
B =
     1
     0
C =
    0.5000    0.2500
D =
     0
```

Această reprezentare poate fi stocată într-o variabilă de sistem echivalentă *H* care oferă practic reprezentarea modelului de stare:

```
H = ss(A,B,C,D)
```

În urma execuției secvenței de mai sus, se obține:

```
H =

  a =
         x1      x2
   x1  -0.75    -0.5
   x2      1       0

  b =
       u1
   x1   1
   x2   0
```

```
  c =
         x1     x2
   y1   0.5   0.25

  d =
       u1
   y1   0
```

```
Continuous-time state-space model.
```

Din această variabilă de sistem poate fi extrasă o reprezentare zerouri-poli-coeficienți de transfer folosind comanda Matlab:

```
[z,p,k] = zpkdata(H,'v')
```

În urma execuției secvenței de mai sus, se obține:

```
z =
   -0.5000
p =
  -0.3750 + 0.5995i
  -0.3750 - 0.5995i
k =
    0.5000
```

Argumentul 'v' forțează returnarea a zerourilor și a polilor în variantă vectorială, utilă pentru sistemele SISO. Reprezentarea zerouri-poli-coeficienți de transfer poate fi folosită din nou pentru o altă afișare într-o nouă variabilă de sistem 'K':

```
K = zpk(z,p,k)
K =

      0.5 (s+0.5)
  -------------------
  (s^2 + 0.75s + 0.5)

Continuous-time zero/pole/gain model.
```

Reprezentarea de tip funcție de transfer poate fi în sfârșit reconstruită folosind comanda **tfdata**, care returnează coeficienții numărătorului și numitorului funcției de transfer:

```
[num,den] = tfdata(K,'v')
num =
         0    0.5000    0.2500
den =
    1.0000    0.7500    0.5000
```

De la modelul de stare la funcția de transfer

Conversia de la modelul matematic de stare cu matricele **A**, **B**, **C** și **D**, la funcția de transfer se poate face prin rularea script-ului următor:

```
A = [0   1
     0   -0.05];

B = [0;
     0.001];

C = [0   1];

D = 0;

[num,den]=ss2tf(A,B,C,D)
```

În urma execuției secvenței de mai sus, se obține:

```
num =
   1.0e-03 *
         0    1.0000         0
den =
    1.0000    0.0500         0
```

Observații utile în cazul curent:

- Numărătorul *num* va avea atâtea linii câte ieșiri sunt (sau câte linii sunt în matricea C).
- Vectorii *num* și *den* returnează coeficienții polinoamelor aferente numărătorului și numitorului în ordinea inversă a puterilor lui *s*.
- Numărătorul și numitorul trebuie verificați **de fiecare dată**, deoarece zerourile la infinit pot produce funcții de transfer eronate.

Un zero la infinit poate fi observat cu ușurință în exemplul următor, în care numerele "−0.0000" și/sau "0" din *num* nu sunt chiar zero ci numere foarte mici:

```
num =
         0         0    1.8182   -0.0000  -44.5460
         0         0    4.5455   -7.4373   -0.0000
den =
    1.0000    0.1818  -31.1818    6.4786         0
```

Aceste numere foarte mici trebuie înlocuite cu zerouri adevărate pentru a elimina inconsecvența.

Obținerea modelului de stare din funcția de transfer se poate face prin funcția predefinită Matlab **tf2ss**.

În mod similar, există funcții pentru a realiza conversia de la modelul de stare la reprezentarea zerouri-poli-coeficienți de transfer sau de la funcția de transfer la reprezentarea zerouri-poli-coeficienți de transfer sau invers. În tabelul 4 sunt prezentate aceste funcții.

Observație: Toate funcțiile și remarcile de mai sus sunt valabile și pentru sistemele în timp discret.

Pentru a determina care este semnificația parametrilor ieșirilor funcțiilor utilizate, este recomandată utilizarea comenzii *help* pentru fiecare funcție predefinită:

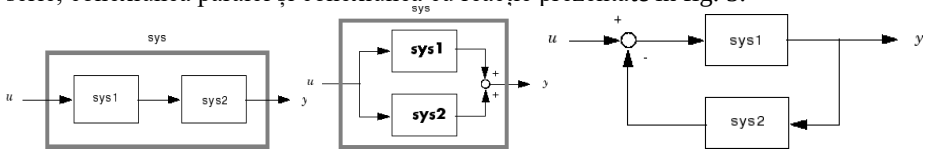
```
>>help numele_funcției
```

Tabelul 4. Funcții Matlab utile în realizarea conversiei model de stare – funcție de transfer – reprezentare zerouri-poli-coeficienți de transfer.

De la \ La	model de stare	funcție de transfer	reprezentare zerouri-poli-coeficienți de transfer
model de stare		ss2tf(A,B,C,D)	ss2zp(A,B,C,D)
funcție de transfer	tf2ss(num,den)		tf2zp(num,den)
reprezentare zerouri-poli-coeficienți de transfer	zp2ss(z,p,k)	zp2tf(z,p,k)	

4. Scheme bloc în Matlab

Conexiunile de bază ale sistemelor în reprezentarea schemelor bloc sunt conexiunea serie, conexiunea paralel și conexiunea cu reacție prezentate în fig. 5.



(a) (b) (c)

Fig. 5. Conexiunile de bază ale sistemelor: conexiunea serie (a), conexiunea paralel (b), conexiunea cu reacție (c) [2].

Presupunând funcțiile de transfer în forma

$$sys1(s) = \frac{num1(s)}{den1(s)}, . \tag{1.8}$$

$$sys2(s) = \frac{num2(s)}{den2(s)}, \tag{1.9}$$

funcția de transfer echivalentă după reducerea fiecărei structuri este:

$$sys(s) = \frac{y(s)}{u(s)} = sys1(s) \cdot sys2(s) \ - \text{în cazul conexiunii serie}, \tag{1.10}$$

$$sys(s) = \frac{y(s)}{u(s)} = sys1(s) + sys2(s) \ - \text{în cazul conexiunii paralel}, \tag{1.11}$$

$$sys(s) = \frac{y(s)}{u(s)} = \frac{sys1(s)}{1 \pm sys1(s) \cdot sys2(s)} \ - \text{în cazul conexiunii cu reactie}, \tag{1.12}$$

iar în acest ultim caz dacă la numitor este semnul "+" atunci este reacție negativă, iar dacă la numitor este semnul "**–**" atunci este reacție pozitivă.

Dacă toți coeficienții polinoamelor de la numărător și numitor sunt dați în Matlab în vectorii *num1*, *num2*, *den1* și *den2*, atunci pentru a calcula tipul conexiunii poate fi folosit codul Matlab următor:

```
>> sys1=tf(num1,den1);sys2=tf(num2,den2);
>> series(sys1,sys2)

ans =

          1
  -----------------
  s^3 + 3 s^2 + 2 s

Continuous-time transfer function.

>> parallel(sys1,sys2)

ans =

    s^2 + 3 s + 1
  -----------------
  s^3 + 3 s^2 + 2 s

Continuous-time transfer function.

>> feedback(sys1,sys2,-1)

ans =

        s^2 + 2 s
  --------------------
  s^3 + 3 s^2 + 2 s + 1

Continuous-time transfer function.
```

Observație: Conexiunea cu reacție este extrem de des întâlnită în proiectarea sistemelor de reglare automată, deoarece aceasta reprezintă conexiunea dintre procesul condus (sys2) și regulator (sys1).

Conform celor menționate mai sus, baza o reprezintă cele trei tipuri de conexiuni, însă în practică pot apărea topologii pentru diagrame mai complexe. În fig. 6 sunt reprezentate în manieră grafică diverse operații, în care semnul "≡" indică echivalența.

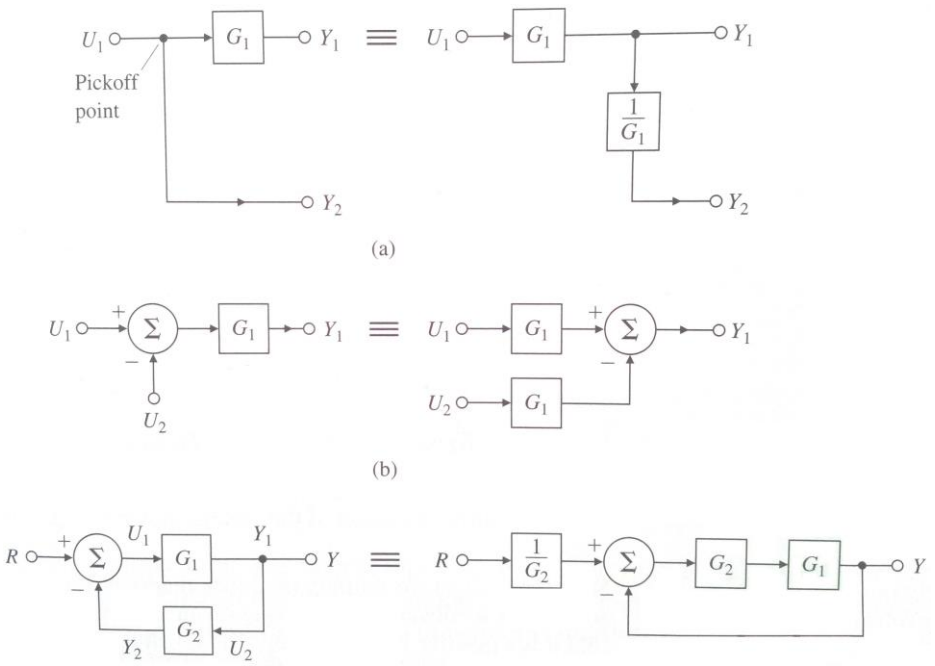


Fig. 6. Reprezentări echivalente ale tipurilor de conexiuni [4].

În cazul fig. 6 (a), blocul cu funcția de transfer G_1 poate fi mutat înaintea nodului pe ramurile care provin din U_1 și Y_2 sub forma G_1 și respectiv $1/G_1$. În fig. 6 (b), blocul G_1 poate fi mutat înaintea sumatorului atât la U_1 cât și la U_2. În conexiunea cu reacție din fig. 6 (c), G_2 poate fi, de asemenea, mutat așa cum se arată în imagine, fără a afecta relațiile matematice dintre intrarea R și ieșirea Y. Aceste ajustări sunt foarte utile pentru reducerea (simplificarea) schemelor bloc.

D. Modelarea și simularea sistemelor în Simulink.
Pentru introducerea în problematică este necesară consultarea materialului anexat **Simulink.pdf**.

Exemplul 1: În cazul sistemului de tip masă-arc-amortizor modelat anterior utilizând mediul Matlab, în fig. 7 sunt prezentate trei implementări posibile ale modelului în Simulink.

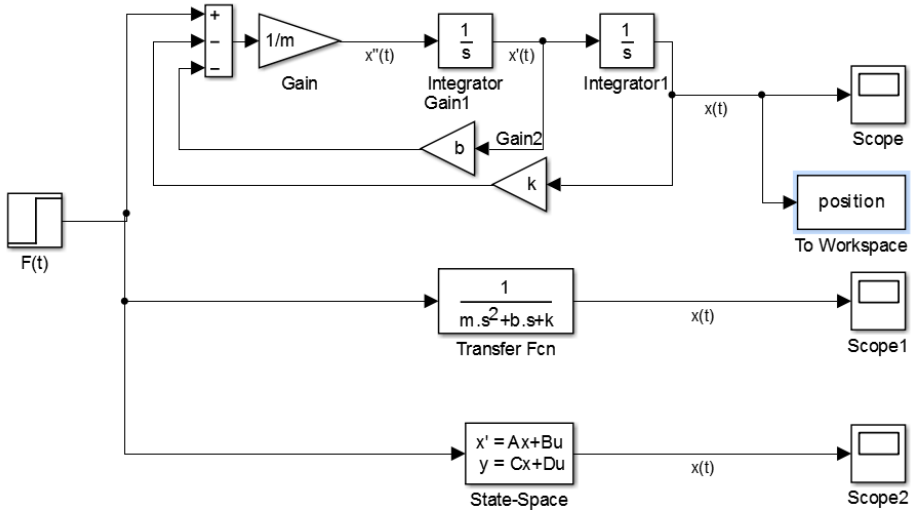


Fig. 7. Implementări în Simulink ale sistemului de tip masă-arc-amortizor.

Cele trei moduri de a implementa procesul de tip arc-amortizor-masă din fig. 7 sunt (în ordine de sus în jos):
- prin ecuațiile matematice explicite (EME),
- prin funcția de transfer (FDT),
- prin modelul de stare (MS).

În primul caz este implementată în Simulink relația (1.1), în cazul funților de transfer este implementată în Simulink relația (1.4), iar în cazul modelului de stare este implementată în Simulink relația (1.2). Variabilele *m*, *b*, *k* sunt definite în spațiul de lucru (Workspace) Matlab, iar Simulink le va prealua din Workspace. De menționat din nou că Matlab este case sensitive. Comportamentul celor trei variante de implementare ale sistemului de tip masă-arc-amortizor din fișierul Simulink „*.mdl*" este identic. Poziția $x(t)$ a sistemului mecanic este privită ca ieșire, iar forța mecanică $F(t)$ este considerată ca intrare, ambele fiind etichetate în fig. 7. Blocul de tip "To-Workspace" etichetat ca și "poziție" retrimite datele în Workspace-ul Matlab pentru postprelucrare.
Implementarea prin ecuațiile matematice explicite este cea mai generală, întrucât permite descrierea sistemelor neliniare. Celelalte două implementări pot descrie doar comportamentul sistemelor liniare. Combinațiile celor trei implementări pot fi folosite fără probleme. Există, de asemenea, un alt mod de a descrie comportamentul complex al sistemelor folosind așa-numitele **S-functions**, dar acestea nu vor fi discutate în cadrul lucrării curente de laborator.

Observația 1: Implementarea modelului de stare și implementarea prin ecuațiile matematice explicite permit stabilirea unor condiții inițiale nenule (în notație vectorială pentru modelul de stare și individual pentru fiecare integrator pentru

implementarea prin ecuațiile matematice explicite). Condițiile inițiale nule sunt presupuse pentru implementarea prin funcții de transfer.

Observația 2: Blocurile de tip "Scope" sunt utile pentru vizualizarea rezultatelor simulării imediat după simulare. În Simulink sunt disponibile diverse tipuri de blocuri pentru vizualizarea rezultatelor simulării. O altă metodă oarecum mai elaborată de afișare a rezultatelor care necesită un efort suplimentar de scris cod este trimiterea semnalelor către Workspace-ul Matlab (folosind blocurile "To Workspace") și apoi afișarea rezultatelor grafice folosind comanda "plot".

Tema de casă 1: Implementați sistemul electric din paragraful C.2 în cele trei forme prezentate în cazul sistemului de tip masă-arc-amortizor. Simulați modelul/sistemul și observați răspunsul în timp al intensității curentului $i(t)$ pentru diferite intrări $v(t)$, în speță semnalul de tip impuls, semnalul de tip treaptă, semnalul de tip rampă și semnalul de tip sinusoidal. Parametrii modelului sunt R=1 kΩ, C=1 mF și L=1 H.

Diverse modele (cum sunt, de exemplu, cele rezentate în primul laborator) pot fi simulate folosind Simulink. În acest context este propusă tema următoare:
Tema de casă 2: Implementați și simulați modelul dinamicii studenților folosind Matlab/Simulink. În Matlab, acest lucru poate fi realizat prin implementarea unui script de tip funcție. În Simulink, implementarea va fi grafică folosind blocurile disponibile. În fiecare caz, vor fi utilizate grafice pentru a afișa evoluția în timp a numerelor de boboci și de absolvenți.

References

[1] R.H. Bishop, *Modern Control Systems Analysis and Design Using Matlab*, Addison-Wesley, Boston, MA, USA, 1993.

[2] http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=SystemModeling

[3] http://ctms.engin.umich.edu/CTMS/index.php?aux=Extras_Conversions

[4] G.F. Franklin, J.D. Powell, A. Emami-Naeini, *Feedback Control of Dynamic Systems, 5th Edition*, Prentice Hall, Upper Saddle River, NJ, USA, 2009.