

LABORATOR NR.6: SIMULAREA SISTEMELOR DINAMICE LINIARE ȘI NELINIARE CU TOOLBOX-UL SIMULINK

Obiectivul lucrării:

Obiectivul lucrării este deprinderea abilităților de lucru cu mediul de simulare Simulink – Matlab.

1 Introducere

Toolbox-ul Simulink este un instrument puternic al mediului științific MATLAB. Principalele caracteristici funcționale ale toolbox-ului sunt:

- construcția user friendly a schemelor funcționale, numai cu ajutorul mouse-ului;
- simularea sistemelor liniare dar și neliniare, inclusiv în regim nestaționar
- simularea sistemelor mixte analogice și numerice

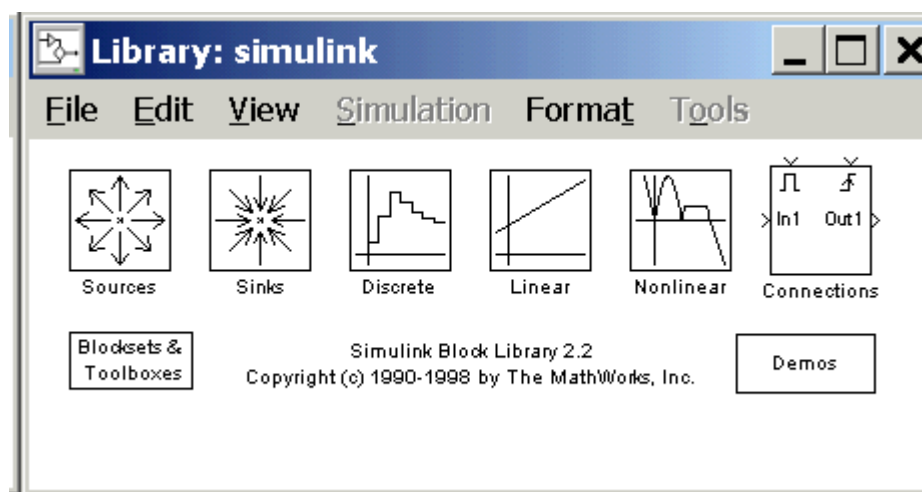


Fig. 1 – Biblioteca standard Simulink.

Prin elaborarea schemelor, Simulink-ul folosește o interfață prietenoasă ceea ce-l face mult mai atractiv decât setul de funcții dedicate aceluiași scop din Matlabul propriu-zis. Cu toate acestea cele 2 softuri rămân complementare și legate unul de celălalt, Simulink-ul nefiind din punct de vedere ierarhic decât un instrument al Matlab-ului.

Interfața utilizând Simulink-ul este utilă pentru introducerea directă a schemelor funcționale aferente sistemelor dinamice. Deși teoretic Simulink-ul nu este indispensabil ca și utilizare, în 90% din cazuri se preferă rezolvarea problemelor prin apelarea sa. Lansarea în execuție se face fie prin shortcut-ul disponibil în bara meniului din fereastra de comandă a Matlabului fie prin lansare directă tastând:

simulink

o fereastră nouă, numită *simulink* va apărea. Ea reprezintă în fapt o bibliotecă de blocuri funcționale standard reunită în 7 sau mai multe grupuri (numărul depinde de versiunea de Matlab sub care se

lucrează) – vezi figura 1. Se remarcă bibliotecile lineare și neliniare conținând fiecare o colecție de blocuri, evident liniare și neliniare. Fidel filozofiei Matlab, Simulink-ul este deschis (forțând semantica i se poate asocia atributul de *open source*) în sensul în care utilizatorul poate concepe blocuri proprii ce pot fi organizate la rândul-le în biblioteci personale ce pot fi adăugate și apoi utilizate în cadrul browserului *simulink*.

În primă abordare, chestiunea menționată anterior nu este un imperativ în sine, utilizarea Simulink-ului la capacitate și coerență maximă revendicând o bună cunoaștere a Matlab-ului și utilizarea acestuia în conexiune permanentă cu Simulink-ul. Trebuie cunoscut faptul că orice bloc Simulink poate fi parametrizat în manieră simbolică adică definirea parametrilor în spațiul de lucru al Matlab-ului, fie în manual, fie executând un fișier *script*. În aceste cazuri, mai multe fișiere de parametrizare sunt create și apoi lansate în Matlab înaintea simulării propriu-zise din Simulink. Documentarea acestor fișiere permite raționalizarea și organizarea unui număr mare de simulări. Astfel, competențele în programare (organizarea, descompunerea și analiza problemei, structurarea) ale utilizatorului devin determinante. Toolbox-urile complementare (Real Time Workshop, Matlab C Compiler) permit în acest context, de a genera codul C al schemei de simulare, astfel încât fișierul obiect pentru procesoarele de semnal (DSP dSpace de ex.) să poată fi programat relativ ușor în C. În consecință, performanțele în termeni de timp de calcul vor fi radical îmbunătățite.

Combinăția între Simulink și un sistem de achiziție în timp real este astfel posibilă dar destul de delicată și limitată în rapiditate din cauza lentorii și complexității sistemului de operare sub care sunt dezvoltate în prezent mediile respective (variantele în Unix, Linux s-au dezvoltat susținut doar în ultimii ani).

Programarea grafică constând în introducerea directă a schemelor funcționale este un instrument remarcabil oferit de Simulink. Totuși, limitele acestui gen de programare sunt atinse relativ repede fiindcă sistemele sunt relativ complexe; în acest caz se preferă programarea în de tip *text* în maniera clasică, creându-se așa numitele S-function în schimbul interconectării blocurilor prin intermediul mouse-ului. Funcțiile *S-function* au un format particular, propriu Simulink-ului dar se scriu cam în aceeași manieră ca și funcțiile Matlab, constând în programarea ecuațiilor diferențiale ale sistemului de simulat.

2. Exemplu introductiv : simularea unui sistem dinamic liniar

Ne propunem de a ilustra utilizarea Simulink-ului prin exemplul clasic al unui motor de c.c. cu excitație separată. *Se poate ignora absența unor cunoștințe despre mașinile electrice, funcționarea mașinii fiind guvernată de ecuații diferențiale de ordinul 1 ce vor fi considerate ca atare.*

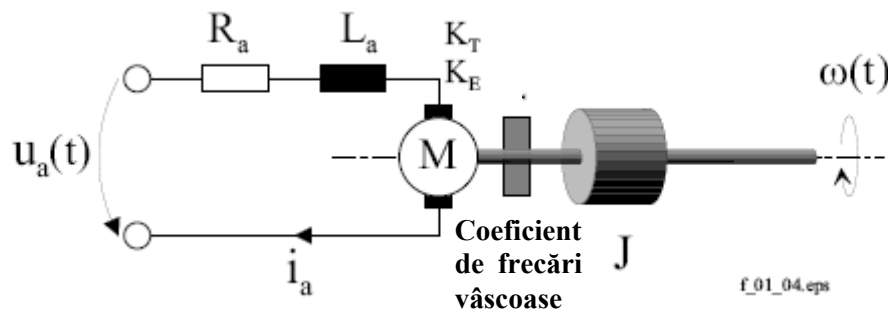


Fig. 2. Mașina de c.c. în reprezentare tip circuit electromecanic (schemă tehnologică)

Se admite că frecarea este proporțională cu viteza prin coeficientul de frecări vâscoase R_f . Ipotezele de lucru pleacă de la premiza unui sistem liniar fără perturbații.

2.1 Modelare

Modelele în continuu (timp, t) și complex (variabila s) sunt descrise de ecuațiile:

$$\begin{cases} u_A = R_A i_A + L_A \frac{di_A}{dt} + K_E \omega \\ J \frac{d\omega}{dt} = T_{em} - R_f \omega; T_{em} = K_T i_A \end{cases} \xrightarrow{\text{Laplace}} \begin{cases} U_A(s) = R_A I_A(s) + s L_A I_A(s) + K_E \Omega(s) \\ s J \Omega(s) = K_T I_A(s) - R_f \Omega(s) \end{cases}$$

În care condițiile inițiale sunt considerate nule.

După scrierea ecuațiilor, schema funcțională detaliată este ilustrată în fig.3.

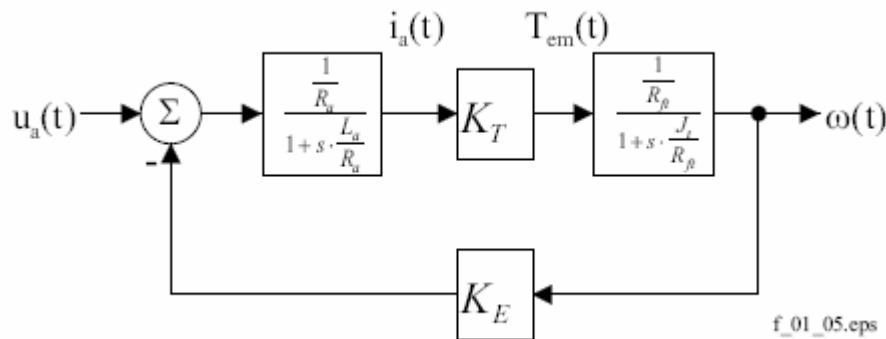
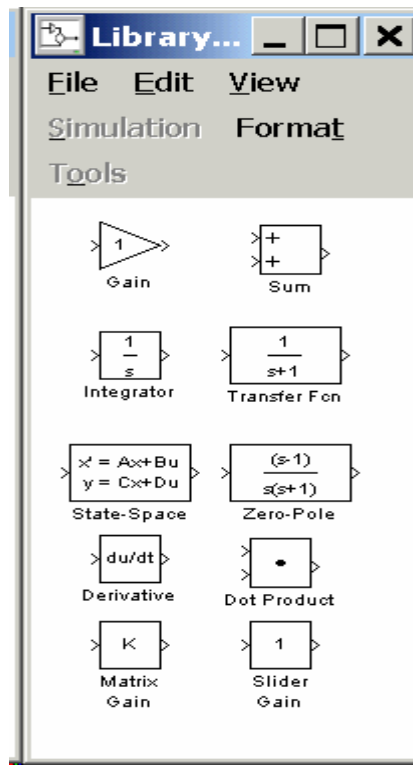


Fig.3. Schema funcțională derivată din cea tehnologică

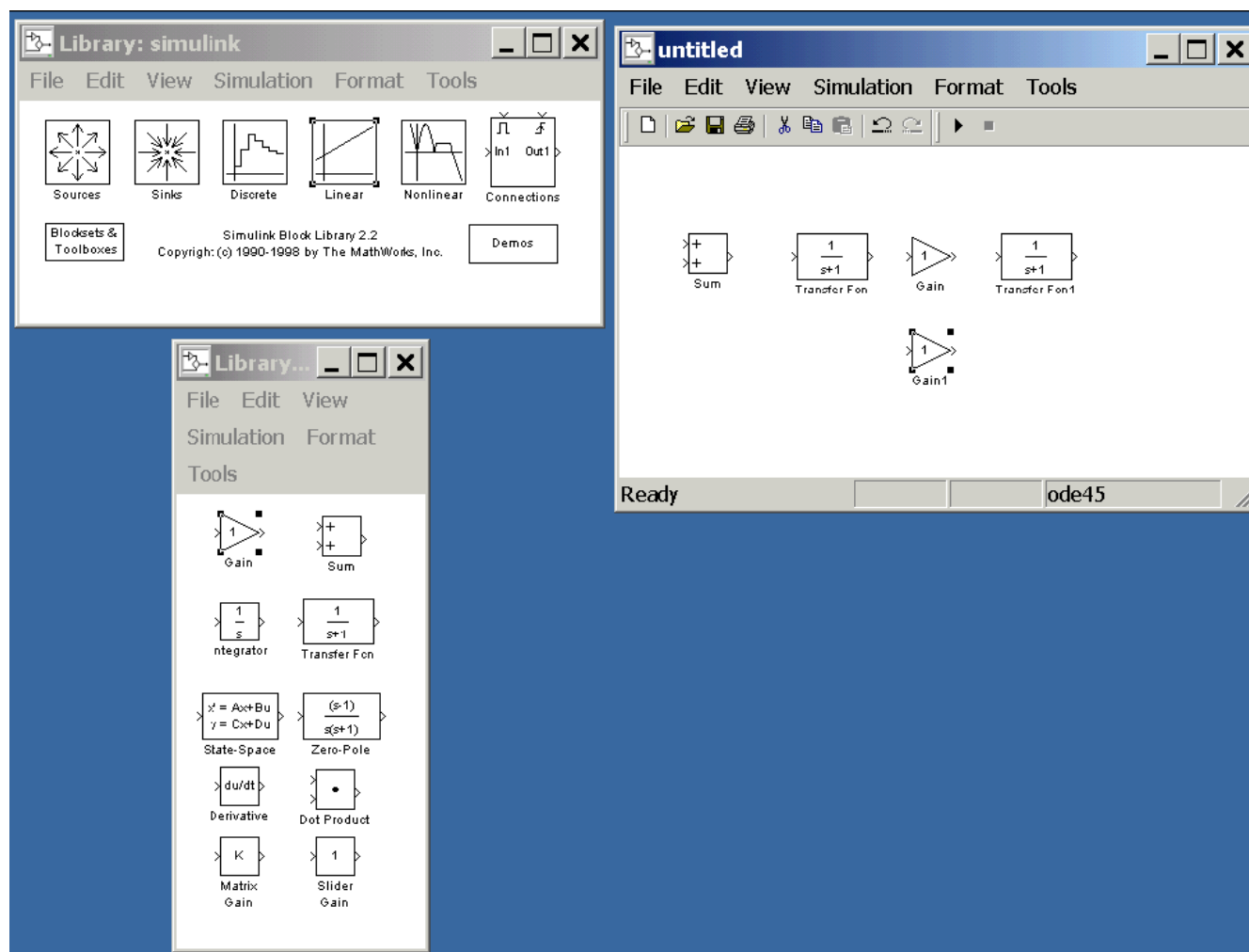
2.2 Conceperea schemei în Simulink



Pentru introducerea schemei funcționale din figura 3 în SIMULINK, se procedează prin apelarea în workspace-ul Matlab-ului a comenzii *simulink* sau lansarea prin apăsarea shortcut-ului afreent browserului Simulink. Pentru definirea ferestrei de lucru, se selectează **New** din meniul **File**. O fereastră nouă corespunzătoare unui fișier de tip model va fi lansată sub denumirea implicită de **untitled**. Scheme va fi construită în această fereastră – fișier plecând de la blocurile conținute în biblioteca Simulink. Pentru conformitate, sunt prezentate capturi ale ferestrelor de lucru, metodologia de abordare fiind de tipul **drag and drop** familiară unui utilizator obișnuit de PC.

În continuare sunt prezentate aceste ferestre al căror conținut este lesne de intuit și înțeles în perspectiva unei implementări a exemplului ca temă de laborator. Manipularea și diversele atribute grafice ale blocurilor utilizate vor fi vizualizate și testate de asemenea în laborator, o descriere coerentă a modului de lucru putând deveni în

contextul acestei prezentări greoaie și redundantă în absența calculatorului.



După realizarea interconectărilor se va obține schema din figura 4.

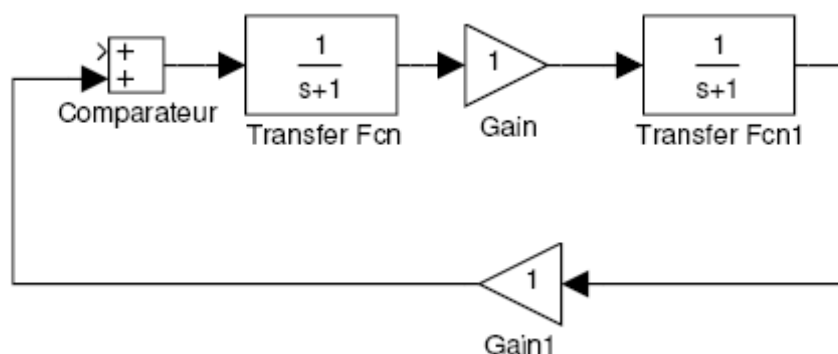
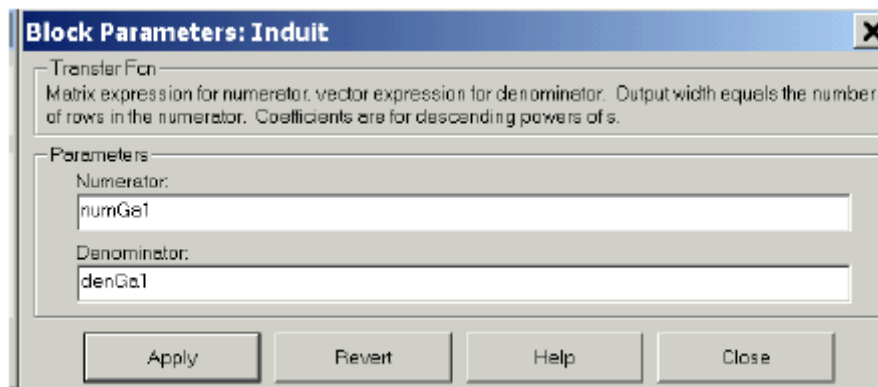
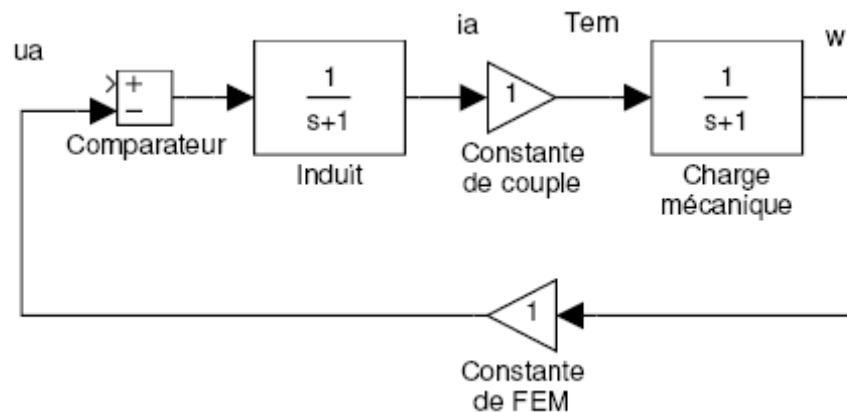


Fig.4. Schema principală a aplicației.

Pentru ca schema de mai sus să devină funcțională se vor activa pe rând blocurile componente înlocuindu-se parametrii implicați cu cei reali (dați fie la nivel formal, fie efectiv – în cele ce urmează se ia în considerare varianta formală). Prin manipulări succesive, se vor obține pașii de mai jos ilustrați la nivel de captură. Accesarea blocuri se face prin dublu click pentru parametrizare sau click dreapta

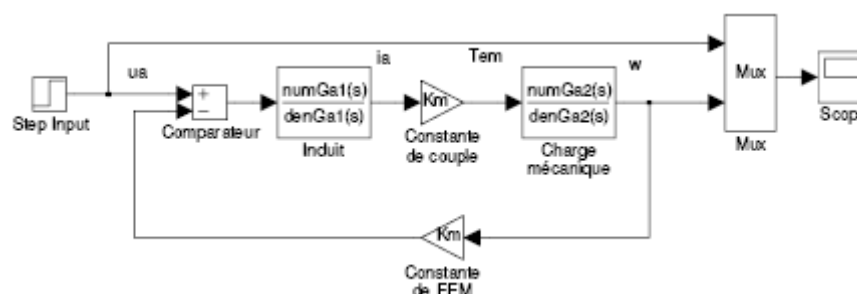
pentru schimbarea atributelor grafice.



S-a preferat ilustrarea cu parametrii formali, cei efectivi fiind definiți în spațiu de lucru al Matlab-ului.

```
numGa1 = 1/Ra;  
denGa1 = [La/Ra, 1];
```

Rămâne de definit tipul semnalului de intrare, respectiv tensiunea u_A , care se alege ca funcție treaptă (echivalentul conectării la o sursă de c.c. a mașinii) și maniera de vizualizare prin intermediul ieșirii de tip *Scope*. **Ca temă se vor prelucra grafic semnalele de curent și turație utilizând facilitățile de reprezentare grafică din Matlab, prin exportul datelor în spațiu de lucru și tratarea acestora ca vectori și/sau matrici.**



2.3 Inițializarea parametrilor

Pentru inițializarea parametrilor, se propune crearea unui fișier MATLAB de tip .m file, (script), adică un fișier care să conțină grupul de comenzi necesare. Ținându-se cont de cunoștințele de MATLAB anterioare, se obține rapid fișierul, ca mai jos:

```
%/*
%*****
%                               INI_01 .M
%*****
%      Fișier script conținând comenzile necesare inițializării parametrilor motorului de curent-continuu
%
%
%*****
%*****
%
%
%*****
%*****
%*/
%      Inițializare parametrii motor curent-continuu
%
%
Ra = 1.84;
La = 0.00077;
KT = 0.9;
KE = KT;
Rf = 0.001;
J = 0.0061;

%      Funcția de transfer
numGa1 = 1/Ra;
denGa1 = [La/Ra, 1];

numGa2 = 1/Rf;
denGa2 = [J/Rf, 1];
```

Execuția acestui fișier se face introducând numele său ini_01 în fereastra de lucru. Se poate verifica inițializarea variabilelor prin intermediul comenzii (instrucțiunii), who , introdusă în fereastra de lucru a MATLAB-ului, (figura 10)

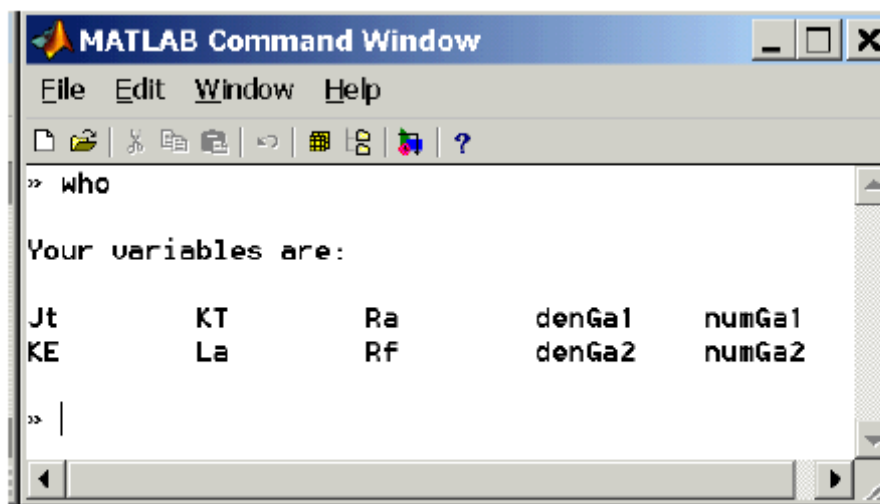


FIG. 10 –

2.4 Simularea

Se poate acum reveni la fereastra Simulink ex_01 (dacă nu mai este deschisă, se introduce numele său

în MATLAB), și se intră în meniul Simulation, opțiunea Parameters, unde se pot face mai multe setări importante, figura 11.

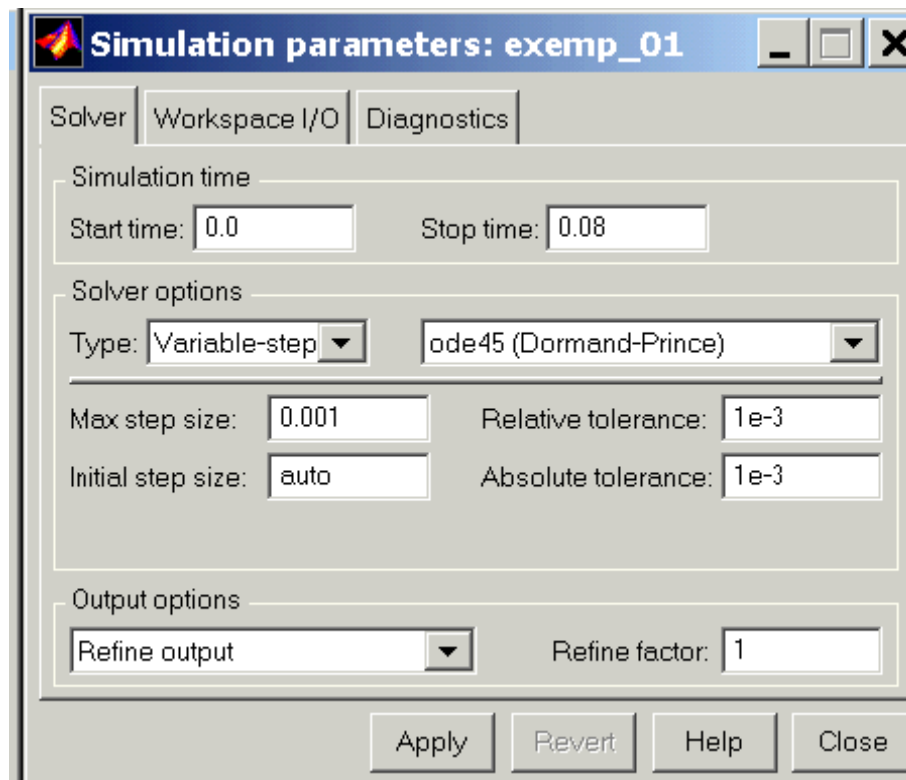


Fig.11

În acest meniu se pot face următoarele setări:

- metoda de integrare numerică a ecuațiilor diferențiale. Se poate alege implicit Runge-Kutta 45;
- momentul începerii simulării ;
- sfârșitul simulării ;
- pasul de integrare maxim este ajustat cu atenție deoarece are o influență importantă în ceea ce privește precizia și timpul de calcul.

Înainte de a începe simularea se poate determina rapiditatea fenomenului, în cazul studiului nostru prin calculul constantelor de timp mecanică T_m și electrică T_e .

$$T_m = \frac{R_a \cdot J}{K_T \cdot K_E} = \frac{1.84 [\Omega] \cdot 0.0061 [kg \cdot m^2]}{(0.9 [\frac{N \cdot m}{A}])^2} = 0.0139 [s]$$

$$T_e = \frac{L_a}{R_a} = \frac{7.7 [mH]}{1.84 [\Omega]} = 0.0042 [s]$$

Pasul de integrare ales este de ordinul 0.001 [s], cu o precizie fixată de parametrul **tolerance**.

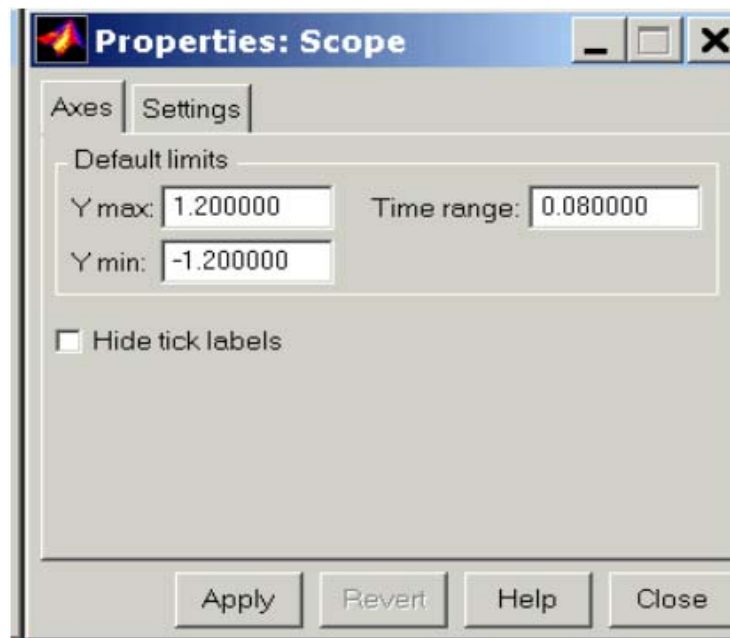


FIG. 12 –

Prin « dublu-clik » pe osciloscop (scope), se poate ajusta dimensiunea scalei pe orizontală și verticală, în cazul nostru alegându-se 0.08 [s] și respectiv 1.2 [rad/s], figura 12.

Lansarea simulării se face alegând opțiunea **Start** din meniul **Simulation**.

Rezultatul simulării este prezentat în figura 13. Pentru o bună interpretare a rezultatelor se poate apela la o serie de opțiuni, cum ar fi spre exemplu zoom-ul general, zoom-ul pe verticală sau orizontală, figura 13.

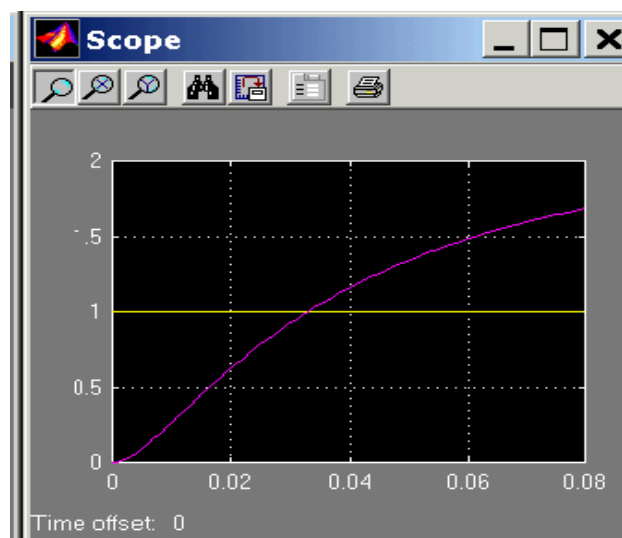


Fig. 13

2.5 Salvarea semnalelor și prelucrarea acestora în MATLAB

Pentru exemplificarea operației de salvare și prelucrare a unui semnal din Simulink, alegem tensiunea

$u_A(t)$, cuplul electromagnetic $T_{em}(t)$ și viteza unghiulară $\omega(t)$. Pentru a efectua această operație nu trebuie să se omită informația despre timp t , care se poate obține utilizând blocul Clock din grupul Sources, biblioteca Simulink. Cele trei semnale, $u_A(t)$, $T_{em}(t)$ și $\omega(t)$, sunt multiplexate, (reunite), într-un semnal de tip vector, cu ajutorul unui nou bloc Mux din grupul Connection. Semnal obținut la ieșirea blocului Mux este transmis la un alt bloc ce asigură legătura Simulink-ului cu MATLAB-ul, To Workspace ce se găsește în grupul Sinks. Acest ultim bloc are ca parametrii (figura 14), numele variabilei MATLAB în care rezultatele vor fi salvate și o indicație relativă a numărului maxim de pași de calcul salvati. Dacă durata pasului este mică și cea a simulării este mare se ajunge la saturarea spațiului de memorie. Pentru a evita această problemă se poate specifica în blocul To Workspace de a nu se salva decât intervale regulate, de durată mai mare decât pasul de integrare.

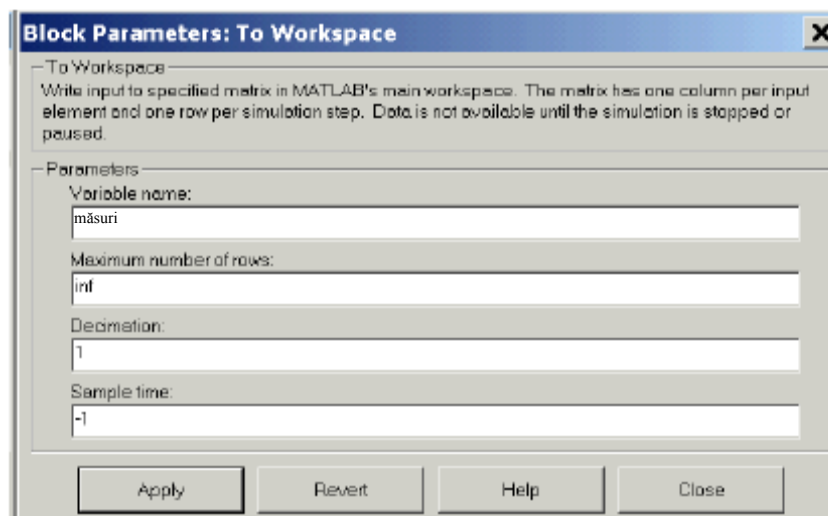


FIG. 14 –

Schema implementată în Simulink este prezentată în figura 15 și este salvată cu numele ex_02.mdl.

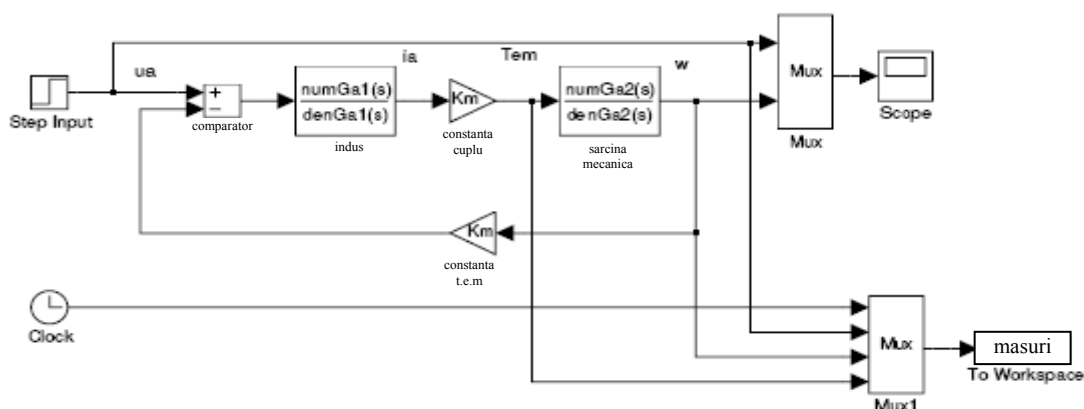


FIG. 15 –

După finalizarea simulării exemplului din figura 15, la revenirea în MATLAB se poate observa existența variabilei măsurî, creată de blocul To Workspace.

Fișierul MATLAB cal_01.m extrage informațiile și trasează cele trei curbe $u_A(t)$, $T_{em}(t)$ și $\omega(t)$, după care calculează puterea mecanică $P_{mec}(t)$.

```
%/*
%*****
%                               CAL_01.M
%*****
%                               Fișier script conținând comenzile necesare interpretării variabilelor
%                               obținute prin ex_02.m
%
%                               Obținerea timpului t și a celor 3 semnale  $u_a(t)$ ,  $T_{em}(t)$  și  $\omega(t)$ 
%
%                               t = mesures (:,1);
%                               ua = mesures (:,2);
%                               omega = mesures (:,3);
%                               Tem = mesures (:,4);
%
%                               Calculul puterii mecanice
%
%                               Pmec = Tem .* omega;
%
%                               Instrucțiuni afisaj
%                               figure(gcf)
%                               subplot(411), plot(t, ua)
%                               xlabel('t_[s]')
%                               ylabel('u_a(t)')
%                               title('MASURI')
%
%                               subplot(412), plot(t, omega)
%                               xlabel('t_[s]')
%                               ylabel('\omega(t)')
%
%                               subplot(413), plot(t, Tem)
%                               xlabel('t_[s]')
%                               ylabel('T_{em}(t)')
%
%                               subplot(414), plot(t, Pmec)
%                               xlabel('t_[s]')
%                               ylabel('P_{mec}(t)')
%
%                               subplot(111)
```

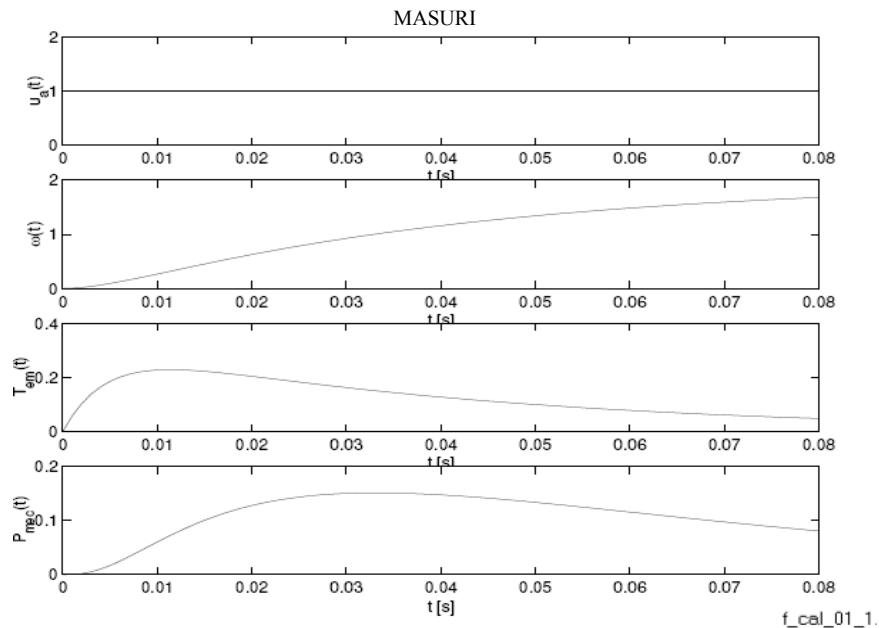


FIG. 16 –

2.6 Pornirea simulării din Matlab

Simulările efectuate pot fi lansate plecând de la spațiul (fereastra) de lucru MATLAB, fără a deschide fereastra cu schema Simulink. Linia de comandă este următoarea :

```
[t,x,y] = sim('ex_01',tfinal,options,u);
```

În cazul nostru :

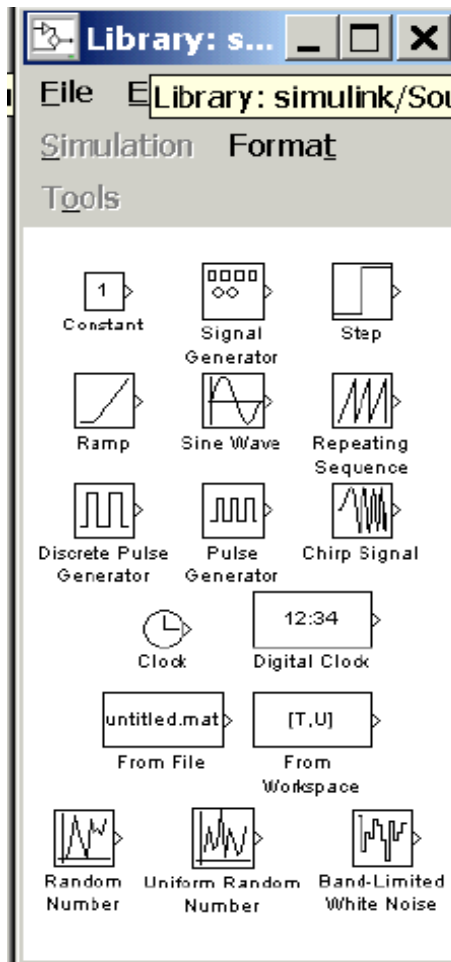
```
ini_01  
[t,x,y] = sim('exemp_01',0.08,[]);
```

3. Biblioteca standard

3.1 SOURCES

Aceasta bibliotecă (figura 17) conține elemente generatoare de semnal cum ar fi semnal treaptă, sinusoidal, fișiere de puncte, variabile MATLAB, zgomot, secvențe, timpul curent de simulare, etc, fără să uităm generatorul de semnal însuși.

Putem crea propriul generator de semnal, aceasta prezentând un real interes pentru parametrizarea acestor blocuri.



3.2. SINKS

Aceasta bibliotecă (figura 18) conține elemente receptoare de semnal, cum ar fi fișiere, variabile

MATLAB si osciloscop. Acesta din urma poate fi considerat ca o unealta cu ajutorul căreia se poate verifica buna funcționare a simulării. Combinația osciloscopului cu un multiplexor analogic (biblioteca CONNECTIONS) permite gruparea mai multor semnale pe o singura linie si de asemenea crearea unui osciloscop multi trace. Existenta unui numar mare de osciloscoape intr-o schema încetinește mult simularea si de asemenea salvarea sistematică a semnalelor in variabile MATLAB.

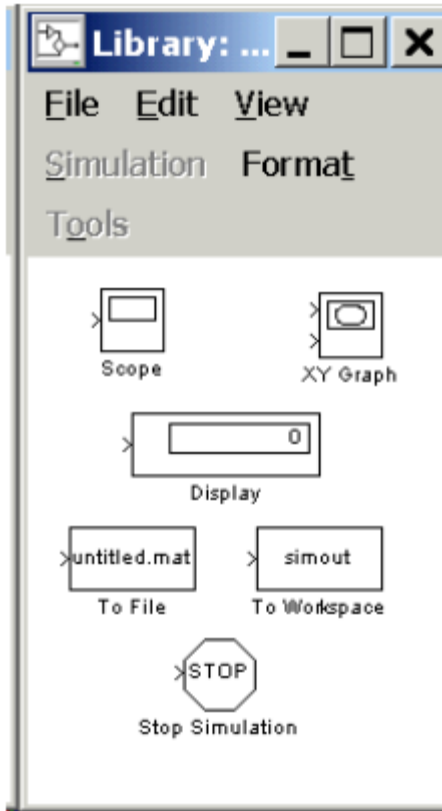


Fig 18

In cadrul unui proiect, este preferabila utilizarea acestei soluții , analiza semnalelor putând fi efectuată in MATLAB după terminarea simulării. In felul acesta avem la dispoziție de toate funcțiile MATLAB pentru analiza si tratarea semnalelor (filtrare, interpolare, transformata Fourier, calculul puterii, etc.). Este o regula simpla care merita subliniata : pentru simulare trebuie utilizat Simulink-ul, iar pentru analiza si tratament offline Matlabul.

De exemplu, daca o schema de simulare furnizează semnalele de tensiune $u(t)$ si de curent $i(t)$, este mai bine ca aceste doua semnale sa fie salvate in spațiul de lucru MATLAB la intervale regulate si puterea instantanee $p(t)=u(t)*i(t)$ sa fie calculata după ce s-a terminat simularea si nu in schema de simulare. In acest ultim caz, cantitatea de calcule de efectuat in cursul simulării creste inutil si in plus se adaugă o neliniaritate schemei. In general, Simulinkul trebuie utilizat pentru simularea părții dinamice a sistemelor si trebuie scutit de alte calcule. In teoria reprezentării sistemelor in spațiul de stare aceasta înseamnă limitarea schemei de simulare la ecuația de stare diferențiala.

Exemplu :

$$\frac{d}{dt}\vec{x} = f(\vec{x}) + g(\vec{u}) \quad ,$$

A se omite ecuația de observație :

$$\vec{y} = h(\vec{x}) + i(\vec{u})$$

3.3 Discrete

Aceasta bibliotecă (fig 19) elemente discrete lineare (în exclusivitate discretizarea timpului) cum ar fi amplificările, sistemele dinamice liniare (transmitanța izomorfa, ecuații de stare). Integratorul limitat, curios prezent în această bibliotecă, este de fapt un element neliniar.

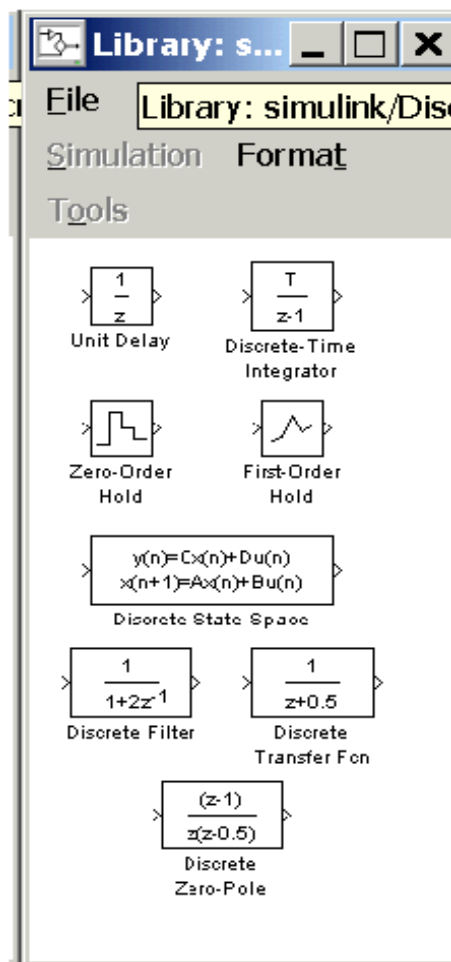


Fig 19

3.4 Liniar

Această bibliotecă (fig 20) conține elemente analogice care efectuează toate operații liniare asupra semnalelor, cum ar fi amplificările, sumatoarele, sistemele dinamice liniare reprezentate prin funcția lor de transfer sau prin ecuațiile de stare.

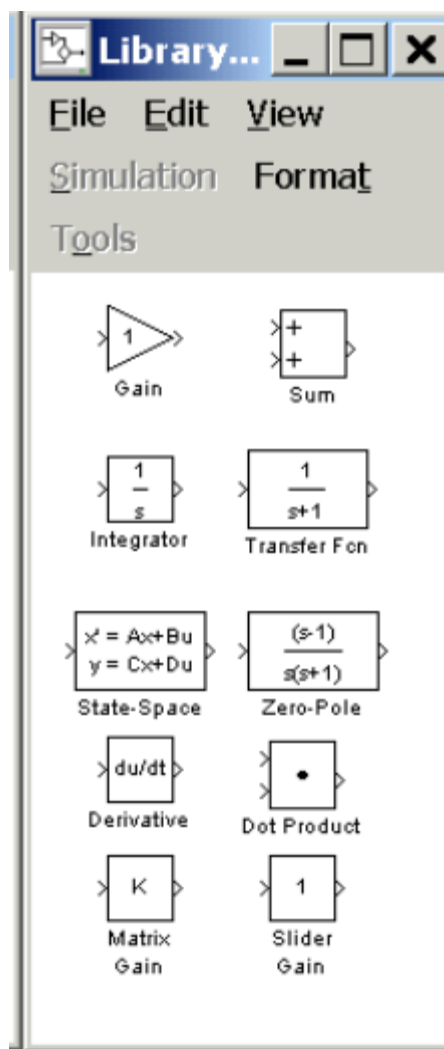


Fig.20

3.5 nonlinear

Aceasta bibliotecă (fig 21) conține elemente neliniare statice sau dinamice. Este una dintre cele mai interesante pentru că datorită diferitelor blocuri puse la dispoziție putem simula de exemplu :

- un element de tipul totul sau nimic, cu sau fără histerezis (sign, Relay)
- un element care introduce o saturare (Saturation)
- un efect de joc mecanic (backlash)
- cuantificarea amplitudinii adusă de un convertor (Quantizer)
- un efect de prag sau de zonă moartă (Dead zone)
- un efect de limitare a variației în raport cu timpul (Rate limiter)
- o neliniaritate de tip valoare absolută (Abs)

- un element care efectueaza produsul a doua semnale (Product)

Operatii logice simple sau complexe (AND, Combinatorial logic)

Un element care creaza o intarziere pura (Transport delay)

Un element a carui caracteristica statica este disponibila intr-un tabel (look up table)

Etc

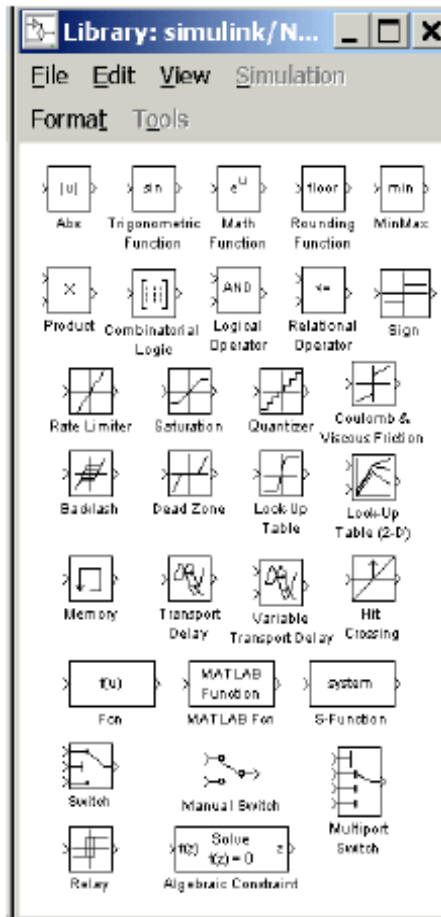


Fig 21

Si in acest caz este de dorit ca aceste blocuri sa fie parametrizate prin intermediul variabilelor definite in MATLAB.

Aceasta lista non-exhaustiva, cu toate ca oferă numeroase posibilități, nu trebuie să ne facă sa uitam posibilitățile fantastice oferite de blocurile

MATLAB Fcn

S-Function

Primul dintre aceste blocuri permite apelarea unei funcții MATLAB standard sau definită de utilizator (liniara sau nu). Aceasta poate fi multivariabilă.

Al doilea bloc oferă aceleași posibilități ca și primul, diferența fiind ca este vorba de o funcție executabilă direct din Simulink : nu este necesară nici o interacțiune prealabilă, funcția S furnizează Simulinkului informațiile cerute, acestea fiind specificate prin parametrii de apel.

Această funcție S poate deci să fie o altă schemă simulink, scrisă explicit de către utilizator într-un fișier care respectă convențiile de apel (simple) ale Simulinkului. În acest ultim caz, funcția poate fi programată fie în limbaj MATLAB sau direct în limbaj C.

3.6. Connections

Această bibliotecă (fig22) conține în special elemente care permit gruparea mai multor semnale pe o aceeași linie (Mux) sau operația inversă (Demux).

Blocurile Import și Export sunt uni-dimensionale nefiind direct utilizabile, cu excepția cazului în care simularea este efectuată în spațiul de lucru MATLAB în linia de comandă, fără să fie deschisă interfața grafică a Simulinkului. Simulinkul le adaugă automat la o schemă când mai multe blocuri sunt grupate într-unul singur, pentru a simplifica schema sau pentru a identifica mai ușor ansamblul de blocuri dedicate unei singure funcții globale.

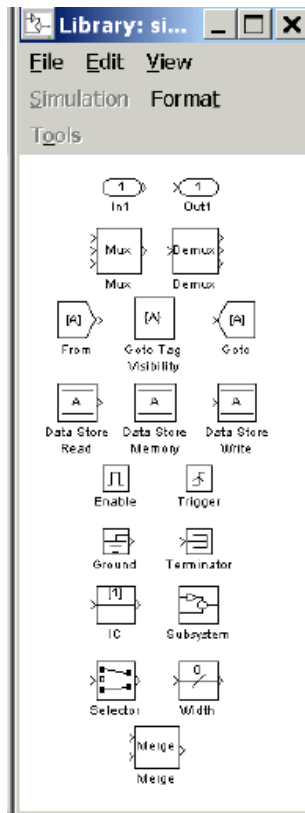


Fig.22

6.4 S-functions in Simulink

Instrumentele Simulink din MATLAB sunt înainte de toate un ansamblu de rutine pentru rezolvarea numerică a ecuațiilor diferențiale date de comportamentul sistemelor dinamice. Printre acestea amintim rk45 (Runge-Kutta de ordinul 4 sau 5). Aceste rutine pot trata fișiere care descriu sistemul de simulat cu condiția ca acesta să aibă un format definit : trebuie să apară ca o S-function.

Nu merge altfel atunci când utilizatorul, în loc să descrie sistemul prin modelul de stare, profita de editorul de scheme Simulink pentru a construi direct schema funcțională a sistemului cu mouse-ul : în spate, editorul Simulink transformă de fapt schema într-o S-function compatibilă cu rutinele de calcul numeric.

Dacă se cunoaște formatul unei S-function, este posibil să nu se utilizeze editorul de scheme Simulink și să se scrie direct un fișier care să satisfacă formatul citat. În anumite cazuri, este mai rapid să se implementeze ecuațiile dinamice sub forma de text decât sub forma grafică. În afara de modelul de stare al sistemului, fișierul S-function va trebui să implementeze protocolul de schimb de date între rutina de calcul numeric și S-function.

În linii mari, putem spune că rutina de calcul interoghează S-function, atribuind valori argumentelor (semnale de intrare și eventual parametri). De exemplu :

```
[sys,x0] = rk45('si_02_02',u,parametre,flag)
```

S-function poate fi scrisă în limbaj MATLAB, C sau Fortran. Forma funcției "S" este foarte generală și se adaptează la fel de bine sistemelor continue, discrete, sau hibride. Deducem astfel că orice sistem dinamic poate fi descris cu o S-function.