

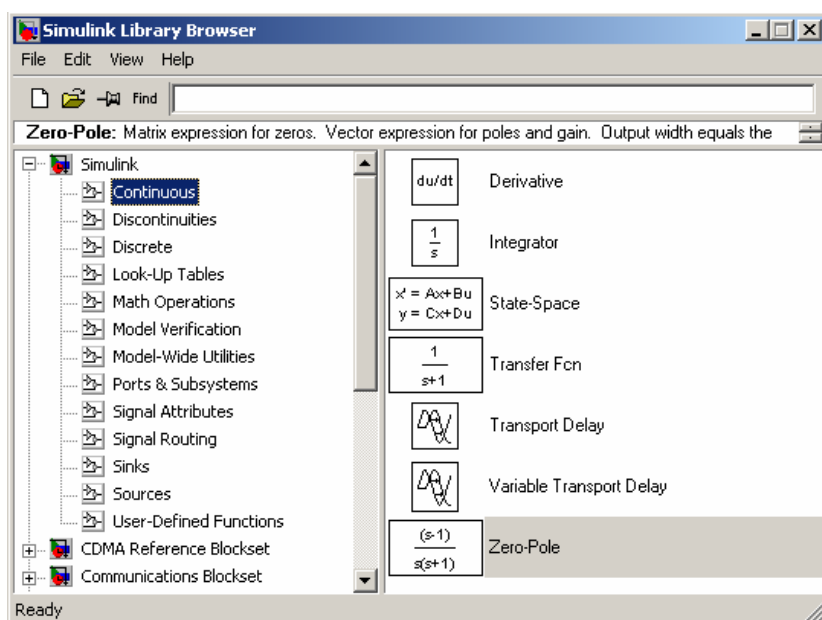


Universitat de Girona
ESCOLA POLITÈCNICA SUPERIOR

PRÀCTICA M-1 REGULACIÓ AUTOMÀTICA

3^{er} ETIEI

INTRODUCCIÓ AL MATLAB-SIMULINK



INTRODUCCIÓ AL MATLAB

El Matlab és un paquet informàtic interactiu per al càlcul numèric. L'element bàsic de Matlab (MATrix LABoratory) és una matriu que no requereix dimensionament, permetent així la resolució de problemes numèrics sense necessitat d'escriure un programa en un llenguatge com Fortran, Basic o C, encara que el propi Matlab estigui escrit precisament en C.

Una de les característiques del Matlab és el seu immens conjunt de funcions. N'hi ha de dos tipus:

- Intrínseques (*built-in*), com les funcions standard sin, cos, log, exp, sqrt, i moltes altres. També té definides les constants com 'pi', o 'j' (números complexos).
- Exteriors (*M-files*). Agrupades en *Toolboxes* o dissenyades per l'usuari.

1. ÚS DE MATLAB

En entrar a Matlab, ens apareix la indicació **>>** seguida del cursor. Aquest símbol ens assenyalava la *línia de comandes*, on escriurem les operacions que ens interressi avaluar.

En Matlab es pot treballar de dues maneres:

- Introduint les instruccions en la línia de comandes. El programa interpreta la línia i n'executa les instruccions després de prémer ENTER. Una vegada executada la comanda, podem introduir la següent.
- Construint, amb l'ajut d'un editor, uns programes anomenats fitxers M-files (amb extensió .m), on col·locarem seqüencialment les instruccions que posaríem en la línia de comandes, i posteriorment cridant aquest programa des de la línia de comandes de Matlab.

S'anomena espai de treball de Matlab (*Workspace*) a l'espai on Matlab guarda la informació de les variables de l'actual sessió de treball. Hi ha dues comandes per a obtenir informació sobre la llista de les variables en l'espai de treball, **who** i **whos**. **who** ens retorna una llista de les variables utilitzades. **whos** ens retorna la llista de variables, però amb més informació sobre la mida (dimensions i espai de memòria ocupat) de cada una.

La informació referent a l'actual espai de treball es pot guardar per a una posterior sessió de treball mitjançant **save** i especificant el nom del fitxer (Matlab hi afegirà l'extensió .mat). Si no s'especifica cap nom, es crea un fitxer anomenat **matlab.mat**. Si es vol, també es poden especificar només algunes variables determinades per a guardar. Per exemple:

```
>>save sessio_1 (Guardarà en el fitxer sessio_1.mat totes les variables del Workspace)
>>save          (Guardarà en el fitxer Matlab.mat totes les variables del Workspace)
>>save sessio_1 A a x (Guardarà en el fitxer sessio_1.mat les variables 'A', 'a' i 'x' del Workspace)
```

Per a restaurar la informació guardada amb **save** haurem d'utilitzar la instrucció **load**, seguint les mateixes regles comentades anteriorment. Per exemple:

```
>>load sessio_1
```

Per esborrar tot l'espai de treball, o una part, utilitzarem la instrucció **clear**. Per exemple:

```
>>clear
```

```
>>clear A a x
```

Cal recordar que Matlab distingeix entre majúscules i minúscules. És a dir que 'A' i 'a' són considerades com dues variables diferents.

1.1. COMANDES D'EDICIÓ DE LÍNIA

Matlab és un llenguatge interpretat (*expression language*). Així, tota expressió entrada des de la línia de comandes és interpretada i avaluada pel sistema Matlab. D'aquesta manera una sentència Matlab té una forma

```
>> variable=expressió
```

Exemple:

```
>> E=m*C^2
```

o simplement:

```
>> expressió
```

Exemple:

```
>> m*C^2
```

Tota variable o expressió ha de començar obligatòriament amb un caràcter alfabètic, seguit de qualsevol altre tipus de dígit. Si no s'assigna cap variable a una expressió, **ans** és automàticament assignat. Per exemple:

```
>>sin(pi/4)
```

al prémer ENTER, ens respondrà amb:

```
ans=
```

```
0.7071
```

A vegades volem executar una comanda, però no volem que es visualitzi el resultat; en aquest cas cal acabar la línia amb un punt i coma (;). Així,

```
>>t=100
```

assigna a la variable t el valor 100 i visualitza el seu contingut. En canvi,

```
>>t=100;
```

fa la mateixa assignació però sense visualitzar el contingut de t.

Per ajudar en la utilització de la línia de comandes, podem utilitzar les tecles següents:

↑	Recupera la línia anterior
↓	Recupera la línia següent
→	Mou el cursor a la dreta
←	Mou el cursor a l'esquerra
Ins	Commuta entre el mode d'inserció i sobrescritura
Del	Esborra el caràcter sobre el cursor
Backspace	Esborra el caràcter a l'esquerra del cursor

Les tecles del cursor del teclat poden ser utilitzades per a editar errors en l'entrada de comandes o bé per tornar a cridar línies de comandes prèvies. Si, per exemple, hem entrat:

```
>>log (sqt (atan2(3,4)))
```

on hem escrit `sqt` i no `sqrt` que és la forma correcta de l'arrel quadrada, Matlab respondrà amb un missatge d'error. Per corregir l'error no cal a tornar a escriure la línia, sinó que prement la tecla `↑`, aquesta tornarà a pantalla i podrem moure el cursor per sobre la línia amb les tecles `→` i `←` o amb el ratolí fins al lloc apropiat per corregir l'error :

```
>> log (sqrt(atan2(3,4)))  
ans =  
-0.2204
```

La utilitat `help` subministra informació sobre la gran majoria dels elements que componen el Matlab. Si només entrem `help`, ens donarà el contingut dels directoris que utilitza el Matlab. Per exemple:

```
>>help help  
>>help  
>>help inv  
>>help :
```

1.2. ARXIUS .M

Matlab, a més de treballar en la línia de comandes, també pot executar seqüències de comandes prèviament emmagatzemades en un fitxer. Aquests fitxers, anomenats `.m` o *M-files*, consisteixen en una seqüència de sentències Matlab, amb la possibilitat de crida a altres fitxers `.m` o a aquest mateix.

Hi ha dos tipus d'arxius `.m`: els arxius d'instruccions i els arxius de funcions. Tant els uns com els altres han d'ésser fitxers de text que contenen caràcters ASCII. La diferència entre els arxius d'instruccions i els de funcions és que en aquests darrers les variables són internes i per tant totalment independents de les de l'espai de treball.

A més de les comandes habituals de Matlab, el fet de treballar amb aquests fitxers permet crear bucles i condicions, com també estructures recursives i crides a altres programes o a ell mateix, utilitzant les sentències típiques per a control de fitxers.

1.2.1. PROCEDIMENTS O ARXIUS D'INSTRUCCIONS

Un *script file* o arxiu d'instruccions és un fitxer que fa executar seqüencialment les comandes que conté i permet entrades des de teclat. Les sentències executades operen globalment en les variables de l'espai de treball, i, per tant, poden modificar el seu valor.

Per poder obtenir una entrada de dades des de teclat, mentre s'executa un *M-file*, utilitzarem `input`. Exemple:

```
n=input('Entra el nombre de columnes: ');
```

Com es pot veure, les cadenes de caràcters s'han d'entrar entre cometes, quedant el text carregat dintre d'un vector, d'un caràcter per cada element.

Exemple:

```
>>s='Matlab'
```

```
s=
```

```
Matlab
```

```
>>size(s)
```

```
ans=
```

```
1      6
```

ens diu la grandària del vector s (1 fila, 6 columnes).

Podem concatenar diferents cadenes de caràcters mitjançant l'ús dels claudàtors. Per exemple:

```
>>s1=['Cadenes de caràcters en ',s]
```

```
s1=
```

```
Cadenes de caràcters en Matlab
```

1.2.2. FUNCIONS

L'arxiu d'instruccions o *function file* admet transferència d'arguments. Les seves variables internes estan definides com a locals, és a dir, no afecten ni són afectades per les operacions globals realitzades en l'espai de treball. Veiem per exemple la funció següent que forma part de Matlab:

```
function y = mean(x)
```

```
%MEAN      Average or mean value.
```

```
%      For vectors, MEAN(X) is the mean value of the elements in X.
```

```
%      For matrices, MEAN(X) is a row vector containing the mean value  
%      of each column.
```

```
%
```

```
%      See also MEDIAN, STD, MIN, MAX.
```

```
%      Copyright (c) 1984-94 by The MathWorks, Inc.
```

```
[m,n] = size(x);
```

```
if m == 1
```

```
    m = n;
```

```
end
```

```
y = sum(x) / m;
```

Podem veure que:

- La primera línia declara el nom de la funció i els arguments d'entrada/sortida.
- El símbol % indica que la resta de la línia és un comentari.
- Les línies % després de la declaració de la funció seran presentades en pantalla quan s'utilitzi help.
- Les variables m, n, x i y són locals.

La funció pot retornar varis arguments.

2. INTRODUCCIÓ DE MATRIUS

Matlab treballa essencialment amb un sol tipus d'element: una matriu numèrica rectangular amb possibilitat d'utilitzar números complexos. Casos particulars són les matrius de 1x1 (escalars) o les matrius d'una sola fila o columna (vectors).

Els números complexos es poden entrar utilitzant com a $\sqrt{-1}$ les variables predefinides *i* o *j* indistintament. També es poden tornar a definir si cal fent

```
>> i=sqrt(-1);
```

Hi ha diverses maneres d'entrar una matriu:

- Entrar una llista explícita d'elements
- Generar-les mitjançant sentències o funcions pròpies
- Generar-les amb fitxers .m
- Carregar-les des de fitxers exteriors

La manera més senzilla, per una matriu de dimensions reduïdes, és utilitzant una llista explícita. Els convenis que s'utilitzen són els següents:

- La matriu es relaciona entre claudàtors: [i].
- Els elements se separen amb espais en blanc o amb comes ,
- El final d'una fila s'indica amb un punt i coma ;

Així,

```
>>A=[1 2 3; 4 5 6;7 8 9]
```

representa una matriu que Matlab ens retornarà com:

A=

```
1 2 3
4 5 6
7 8 9
```

i que és guardada per un ús posterior.

Per entrar una matriu des d'un fitxer .m, cal crear aquest fitxer amb les dades ordenades, i amb claudàtors que indiquin el començament i el final de la matriu. Per exemple, si escrivim un fitxer, en ASCII, anomenat LAB.M, de la forma:

```
A= [ 1 2 3
    4 5 6
    7 8 9]
```

cada vegada que es cridi LAB des de la línia de comandes, es generarà la matriu A.

Una matriu pot ésser definida utilitzant matrius més petites com a elements. Per exemple:

```
>> B=[10 11 12];
```

```
>> A=[A;B]
ens modifica l'anterior matriu A, i la deixa com
A=
     1     2     3
     4     5     6
     7     8     9
    10    11    12
```

Com es pot veure, no cal dimensionar la matriu. La mida de la matriu s'incrementa per tal d'adequar-se als nous elements.

3. OPERACIONS AMB MATRIUS

A part de totes les operacions amb escalars que podem trobar en qualsevol calculadora científica, el Matlab incorpora operacions matricials. La majoria de les vegades s'indiquen de la mateixa manera que en la notació matemàtica tradicional.

3.1. OPERACIONS ARITMÈTIQUES

3.1.1.- TRANSPOSICIÓ

Podem trobar la trasposta d'una matriu utilitzant el signe d'apostrofar '. Per exemple:

```
>>A=[1 2 3;4 5 6;7 8 9]
>>B=A'
B=
     1     4     7
     2     5     8
     3     6     9
```

Si Z és una matriu complexa, en aquest cas Z' serà la trasposta de la conjugada de Z .

3.1.2.-SUMA I RESTA

La suma i la resta de matrius s'indiquen mitjançant els signes respectius de + i -. Es pot fer l'operació sempre que les dimensions de les matrius siguin iguals. Per exemple:

```
>>s=A+B
s=
     2     6    10
     6    10    14
    10    14    18
>>r=A-B
r=
```

0	-2	-4
2	0	-2
4	2	0

La suma i la resta també estan definides quan un dels operands és un escalar (matriu d'1x1). En aquest cas, l'escalar és sumat o restat a tots els elements de l'altre operand. Per exemple:

```
>>e=[1 2 3];
>>e1=e-1
e1=
    0    1    2
```

3.1.3.-MULTIPLICACIÓ I DIVISIÓ

La multiplicació entre matrius de dimensions compatibles s'indica amb el símbol *. Per exemple:

```
>>m=A*B
m=
    14    32    50
    32    77   122
    50   122   194
```

Un escalar (matriu d'1x1) pot multiplicar o ser multiplicat per una matriu. Per exemple:

```
>>me1=2*A;
>>me2=A*2;
Evidentment, els dos resultats són iguals:
me1=
     2     4     6
     8    10    12
    14    16    18
```

La divisió entre matrius s'indica amb dos símbols: \ i /:

- $x=a \backslash b = \text{inv}(a) * b$ és la solució de $a * x = b$
- $x=a / b = b * \text{inv}(a)$ és la solució de $x * a = b$

Exemples:

```
>>d1=A \ B
d1=
   -0.2334   -0.3333   -1.1344
    0.4667   -2.3333   -3.7311
    0.1000    3.0000    5.1989
>>d2=A / B
```


d2=

-0.2334	0.4667	0.1000
-0.3333	-2.3333	3.0000
-1.1344	-3.7311	5.1989

3.1.4.- POTÈNCIA

La potènciació A^p està definida si A és una matriu quadrada i p és un escalar. Exemple:

```
>>p=A^3
```

p=

468	576	684
1062	1305	1548
1656	2034	2412

Podem comprovar que es correspon amb $A*A*A$

3.2. OPERACIONS AMB LLISTES

Operacions amb llistes (*Array operations*) és un terme utilitzat per referir-se a operacions aritmètiques element a element, en contraposició de les operacions matricials algebraiques lineals indicades pels símbols $*$, \backslash , $/$, $^$.

Expressions com $\exp(A)$, $\log(A)$ i $\text{sqrt}(A)$ són operacions amb llistes, definides sobre cada element individual de la matriu A .

Per matrius quadrades, MATLAB també pot calcular funcions transcendents com la matriu exponencial, $\text{expm}(A)$, la matriu logaritme $\text{logm}(A)$ ó la matriu arrel quadrada $\text{sqrtm}(A)$. Com es pot veure, s'ha d'afegir m al nom de la funció ordinària.

El següent exemple reflexa la diferència entre unes i altres:

```
>>a=[2 4; 8 9];
```

```
>>sqrt(a)
```

ans=

1.4142	2.0000
2.8284	3.0000

```
>>sqrtm(a)
```

ans=

0.8259+0.8190i	1.0481-0.3227i
2.0962-0.6454i	2.6601+0.2543i

Podem veure que, $a=\text{sqrtm}(a)*\text{sqrtm}(a)$

Si un dels símbols *,\,/ o ^ ve precedit d'un punt ., indica que és una operació element a element, no matricial. Evidentment, per la suma i la resta no hi haurà cap diferència amb l'operació matricial respectiva.

Per exemple donats els vectors $y=[1 \ 2 \ 3 \ 4 \ 5 \ 6]$ i $x=[6 \ 5 \ 4 \ 3 \ 2 \ 1]$, si es vol com a resultat un vector que el seu primer component sigui el producte entre el primer component de y i de x, el segon component sigui el producte entre el segon component de y i de x, i així successivament, s'haurà de fer:

```
» y=[1 2 3 4 5 6];
» x=[6 5 4 3 2 1];
» y.*x
```

ans =

```
6 10 12 12 10 6
```

3.3. OPERACIONS AMB POLINOMIS

Matlab té una sèrie de funcions que operen primordialment amb vectors i que permeten treballar amb polinomis. Els polinomis es representen com vectors, on els coeficients s'ordenen segons potència descendent.

Per trobar les arrels d'una equació, podem utilitzar **roots**. Per exemple si es volen trobar les arrels de s^3+2s^2+3s+4 , s'introduirà el vector format pels coeficients del polinomi de la forma que segueix:

```
» roots([1 2 3 4])
ans =
-1.6506
-0.1747 + 1.5469i
-0.1747 - 1.5469i
```

La multiplicació i la divisió entre polinomis la podem realitzar amb **conv** i **deconv**, respectivament.

Exemples:

```
>>a=[1 2 3];
>>b=[4 5 6];
>>mp=conv(a,b)
```

```
mp=
4 13 28 27 18
```

```
>>[q,r]=deconv(mp,a)
q=
```

```
4 5 6
r=
0 0 0 0 0
```

ens dona el quocient i la resta de l'operació.

3.4 OPERACIONS RELACIONALS

Existeixen sis operadors relacionals per comparar dos matrius d'iguals dimensions:

<	Més petit que
<=	Més petit o igual
>	Més gran que
>=	Més gran o igual
==	igual
~=	diferent (no igual)

La comparació es realitza entre cada parell corresponent d'elements, i el resultat és una matriu de 1 (veritat) i de 0 (fals).

3.5- OPERADORS LÒGICS

Existeixen tres operadors lògics elementals:

&	AND
	OR
~	NOT

Aquests operadors normalment són utilitzats amb matrius de 1 i 0. Si la matriu té valors diferents, Matlab treballarà prenent els zeros 0 i la resta com 1.

Exemple:

```
>>p=[1 0;1 0];  
>>p1=~p;  
>>p2=p|p1;  
>>p3=p&p1;
```

4. MANIPULACIÓ DE MATRIUS

Matlab permet la manipulació de files, columnes, elements individuals i submatrius.

Per subindexar s'utilitzen vectors que es generen mitjançant la notació de dos punts (*Colon Notation*). Per exemple:

```
>>x=1:5  
genera un vector fila, de 1 a 5, amb increment unitari:  
x=
```

```
1    2    3    4    5
```

D'altra banda:

```
>>x=9:-2:1  
genera un vector de 9 a 1, amb decrement 2.  
x=
```

9 7 5 3 1

Amb `linspace`, podem especificar el nombre de punts del vector, mentre que amb `logspace` podem generar vectors amb intervals logarítmics.

Exemple:

```
>>x=linspace(-1,1,5)
```

x=

```
-1.0000    -0.5000     0.0000     0.5000     1.0000
```

Exemple:

```
>>y=logspace(-2,1);
```

genera 50 punts (és el nombre de punts per defecte, tant per `linspace` com per `logspace`) entre 0.01 i 10 (10^{-2} i 10^1)

En el cas de matrius, els vectors indexats permeten accedir a submatrius contigües i no-contigües. En general, si v i w són vectors amb components enters, $a(v,w)$ és la matriu que s'obté prenent els elements de a amb files definides per v i columnes per w .

Exemples:

```
>>A=[1 2 3;4 5 6;7 8 9]
```

```
>>A(1:2,3)
```

ans=

```
3
6
```

```
>>A(1:2,1:2)
```

ans=

```
1    2
4    5
```

```
>>A(:,2)
```

ans=

```
2
5
8
```

en aquest cas, `:` indica “tots els elements”.

```
>>A(1:2,:)
```

ans=

```
1    2    3
4    5    6
```

Una utilització especial és `A(:)`, que converteix la matriu en un vector columna. Exemple:

```
>>B=A(:)
```

```
B=
```

```
1
4
7
2
5
8
3
6
10
```

5. CONTROL DE FLUX

Matlab té unes sentències de control de flux, igual que la majoria de llenguatges informàtics, que li permeten passar de ser un simple calculador a convertir-se en un llenguatge matricial d'alt nivell.

5.1.- FOR

El bucle for, en Matlab, pren la forma general

```
>> for v=expressió
      sentències
end
```

Tot bucle for ha de tenir forçosament un **end** al final. Exemple:

```
>>for i=1:n
      k(i)=0
end
```

o, el que seria el mateix,

```
>>for i=1:n, k(i)=0, end
```

que, en aquest cas, per cada valor de *i* entre 1 i *n*, executa la sentència *x(i)=0* inclosa entre **for** i **end**.

Si volem sortir del bucle si es compleix una condició utilitzarem la sentència **break**.

5.2.- WHILE

El bucle while permet repetir una sentència, o un grup, sempre que es compleixi una certa condició lògica. La forma general és:

```
>> while expressió
      sentència
end
```

Exemple:

```
>>n=1;
>>while prod(1:n)<1e100
        n=n+1;
    end
>>n
    70
```

ens calcula quin és el primer enter que té com a factorial un número de 100 dígit.

5.3.- IF

Amb la sentència if, igual que en la majoria dels llenguatges de programació, farem que Matlab executi una sentència sempre que es compleixi una condició determinada. Exemple:

```
>>if n>0
    b=inv(a)
end
```

La sentència if també permet la utilització de else i de elseif. Exemple:

```
>>if n>0
    b=inv(a);
else
    b=a';
end
```

Exemple utilitzant elseif:

```
>>if n>0
    b=inv(a)
elseif n==0
    b=a'
else
    b=a*a
end
```

6. GRÀFICS X-Y

La representació gràfica de les dades, que en permet un examen complementari, posseeix un ampli ventall d'opcions que inclouen diversos tipus de gràfics en dues i en tres dimensions, amb possibilitat de variar moltes de les seves característiques bàsiques.

Existeixen diferents comandaments que permeten seleccionar el tipus de gràfic x-y que es vol crear:

<code>plot</code>	gràfic lineal, x-y
<code>loglog</code>	gràfic logarítmic, $\log(x)$ - $\log(y)$
<code>semilogx</code>	gràfic semi-logarítmic, $\log(x)$ -y
<code>semilogy</code>	gràfic semi-logarítmic, x- $\log(y)$

Amb la comanda `plot(y)`, es genera un gràfic x-y lineal dels elements de `y` enfront de l'índex de cada element de `y`. Exemple:

```
>>y1=[0.1 0.48 0.84 1 0.91 0.6 1.04];
>>plot(y1)
```

Si `x` i `y` són vectors de la mateixa longitud, podem crear un gràfic x-y, amb els elements de `x` enfront dels elements de `y`, utilitzant `plot(x,y)`, `loglog(x,y)`, `semilogx(x,y)`, `semilogy(x,y)`. Per exemple:

```
>>x1=[1e0 1e1 1e2 1e3 1e4 1e5 1e6];
>>plot(x1,y1)
>>semilogx(x1,y1)
```

Donant a `plot` múltiples arguments `plot(X1,Y1,X2,Y2,.....,Xn,Yn)` on les variables `Xi,Yi` són parells de valors que representen un gràfic, generarem múltiples línies en un sol gràfic. Cada línia és representada en un color o un tipus de línia diferent. Per veure les diferents opcions feu `help plot`.

Exemple:

```
>> semilogx(x1,y1,x1,sin(y1),x1,exp(y1),x1,log(y1))
```

Matlab fa un escalat automàtic dels eixos del gràfic, però aquest escalat es pot canviar mitjançant la comanda `axis`. Executant `axis` directament, es conserva l'actual escalat per posteriors gràfics, no tornant a l'auto-escalat fins a executar novament `axis`. Exemple:

```
>>ax1=[1 20 15 18];
>>plot(ax1)
>>ax=axis
>>ax2=[1 2 0.5 5];
>>plot(ax2)
>>axis(ax)
```

Podem definir l'escalat manualment indicant els seus límits amb la comanda :

```
axis([x_min, x_max, y_min, y_max])
```

Exemple:

```
>>plot(ax1)
>>axis([0 4 0 30])
```

Una vegada dibuixat un gràfic, s'hi poden posar el títol, els comentaris, les etiquetes dels eixos `x` i `y`, i també es pot crear un engrallat de fons, mitjançant les comandes:

<code>title</code>	títol del gràfic
<code>xlabel</code>	etiqueta de l'eix x
<code>ylabel</code>	etiqueta de l'eix y
<code>text</code>	posicionat de text arbitrari

<code>gtext</code>	posicionat de text amb ratolí (mouse)
<code>grid</code>	engraellat de fons

Exemple:

```
>>t=[0:0.05:4*pi];  
>>plot(t,sin(t))  
>>title('Funció sinus')  
>>xlabel('Radiants')  
>>ylabel('Amplitud')  
>>grid
```

Amb la comanda `gtext('comentari')`, podem escriure qualsevol comentari en el lloc del gràfic que designem mitjançant el cursor que apareix en invocar la comanda.

Es poden tenir diverses pantalles gràfiques obertes simultàniament utilitzant la funció `figure`. Aquesta mateixa funció ens permet indicar a quina d'aquestes pantalles es faran les següents representacions gràfiques.

La comanda `hold` permet que l'actual gràfic quedi en pantalla, de manera que els successius gràfics es representin sobre aquest, amb els eixos ja establerts. Per tornar al funcionament normal cal tornar a executar `hold`.

INTRODUCCIÓ AL SIMULINK

El Simulink és una eina per a modelitzar, analitzar i simular una gran varietat de sistemes: lineals i no lineals, continus i discrets, etc. Es un annex del Matlab que afegeix a la generalitat d'aquest les especificitats dels sistemes dinàmics. Els sistemes es representen gràficament, amb tots els avantatges que això suposa, mitjançant diagrames de blocs.

Hi ha dues fases en l'ús del Simulink: definició de models i anàlisi de models. Qualsevol sessió de treball amb Simulink comença construint un model o cridant un model ja creat i segueix amb l'anàlisi d'aquest model.

Definir el model vol dir construir-lo a partir d'elements bàsics o altres models construïts prèviament. Per a la definició de models Simulink es treballa amb finestres. Dibuixarem el nostre model en una finestra i per a fer-ho utilitzarem els blocs ja definits que es troben en una altra "finestra biblioteca" anomenada *block diagram window* en la qual hi ha la majoria de blocs necessaris per a la definició de models. Aquests blocs estan agrupats en diferents grups (*sinks, sources, linear, discrete, Linear, Nonlinear, Connections i Extras*).

Per a analitzar els models podem fer una simulació directament des del menú de la finestra en que treballem o bé podem utilitzar les funcions del Matlab treballant des de la finestra anomenada *Matlab command window*.

La definició del model i l'anàlisi del model s'influencien mútuament l'un sobre l'altre: segons la definició que fem els resultats de l'anàlisi seran uns o altres i segons aquests resultats es pot modificar el model fins obtenir el comportament desitjat.

1. ÚS DE SIMULINK

Per entrar a Simulink, des de la finestra de comandes del Matlab, executarem `simulink` i una nova finestra anomenada *Simulink library browser* ens apareixerà a la pantalla (Veure Figura 1). Seguidament haurem d'obrir una altra finestra per poder-hi dibuixar el nostre sistema. Per fer-ho anirem al menú *File* i seleccionarem l'opció *New*.

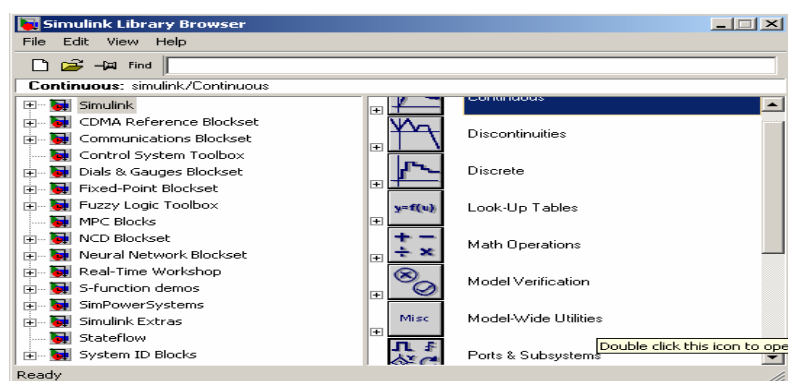



Figura 1. *Simulink library browser*

També la podem obrir des de la barra superior de la finestra Matlab prement el botó



1. EXEMPLE 1: GENERACIÓ D'ONES

L'objectiu d'aquest exemple és agafar un generador d'ones sinusoidals i veure la seva sortida amb l'ajuda d'un oscil·loscopi. Per això des de *Simulink library browser* farem File/New/model, o bé premerem el botó . Aquesta acció ens crea una finestra en blanc amb el nom *Untitled*, que després podrem guardar amb el nom que ens interessi.

Ara ja podem construir el nostre primer model. Necessitarem un generador d'ones sinusoidals (*sine wave*) que trobarem a *Sources* i un oscil·loscopi (*Scope*) que trobarem a *Sinks*.

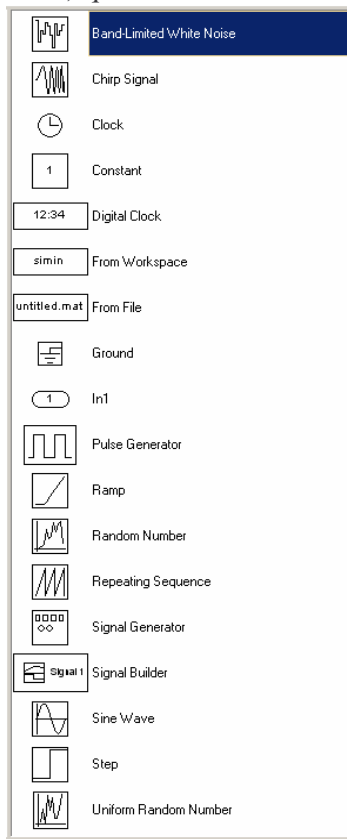



Figura 2. Llibreria *Sources*


En primer lloc ens hem de situar a la finestra *Simulink library browser* i fem doble clic a la icona anomenada *Sources*. Una nova finestra amb blocs individuals ens apareix a la pantalla (Veure Figura 2). Per seleccionar el que ens interessa pitgem el botó esquerre del ratolí quan el cursor estigui a sobre del bloc *Sine Wave* i, sense deixar de pitjar, el situem en la nostra finestra de treball *Untitled*. De la mateixa manera podem moure'l per la finestra. L'altre bloc que necessitem és el *Scope* de la llibreria *Sinks*. El copiarem de la mateixa manera que l'anterior.

El següent pas serà col·locar adequadament els blocs: a l'esquerra hi posarem la font de senyal i a la dreta la sortida.

Cada bloc té uns paràmetres característics als quals haurem de donar valors, per fer-ho farem doble clic a sobre d'ells i ens apareixerà una sub-finestra de diàleg en la qual els paràmetres tindran uns valors per defecte.

Hi ha dues maneres de donar valors als paràmetres. La primera i més senzilla és la de donar els valors que volem als paràmetres directament a la finestra de diàleg. La segona és la de parametritzar el bloc substituint els valors pels noms de les variables. En aquest cas els haurem de donar valors des del Matlab. Si no ho fem ens apareixerà un missatge d'error. Per a blocs on no és freqüent canviar els valors dels paràmetres és aconsellable utilitzar el primer mètode, però si hem de canviar els paràmetres amb freqüència és millor fer servir el segon. És el cas de voler fer una simulació moltes vegades en la que cada cop variem el valor d'un mateix paràmetre.

Si fem doble clic a la icona *Scope*, ens apareix una nova finestra que representa un oscil·loscopi. Els paràmetres que hi podem canviar els podrem modificar des del menú que ens surt al premer el botó  de la finestra *Scope*. Hi posarem valors adients a l'amplitud i la freqüència del senyal

sinusoïdal. També tenim el botó  per fer un autoescalat dependent del senyal que hi tinguem. Aquesta finestra no la tancarem ja que ens servirà per veure els resultats de la simulació, per tant l'haurem de situar a la pantalla de manera que no ens molesti. Amb això haurem acabat la definició dels blocs. Ara els haurem de connectar entre ells.

Les entrades i sortides dels blocs estan indicades amb fletxes (>). Les connexions entre els blocs les farem creant línies entre ells. Per a això, ens situem a sobre de la sortida del bloc *Sine Wave*, pitgem el botó dret del ratolí i, sense deixar de prémer, ens situem a l'entrada del bloc *Scope*. Ens apareix una línia entre els dos blocs. La fletxa ens indica el sentit del flux de senyal.

També podem canviar els noms dels blocs i posar-hi els que nosaltres vulguem per poder entendre millor el diagrama. Això ho podem fer fent clic a sobre el nom del bloc. Per exemple, podem canviar *Scope* per *Oscil·loscopi*.

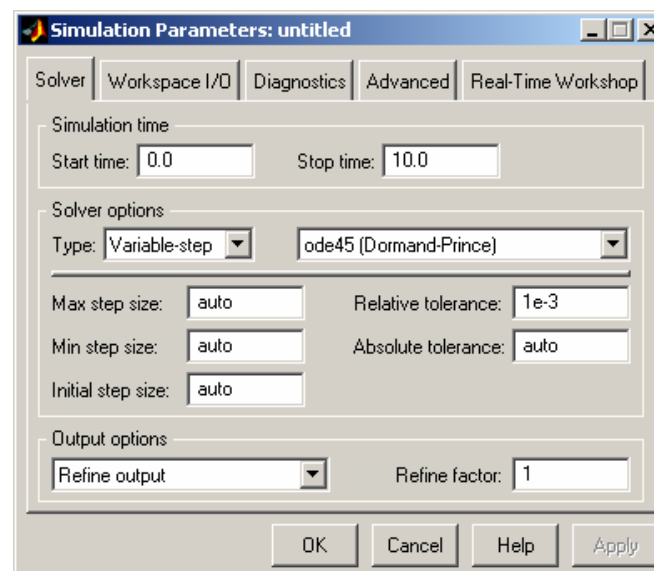


Figura 3. Finestra de paràmetres de simulació

Per poder fer la simulació del nostre sistema haurem de canviar els valors dels paràmetres de simulació que hi ha per defecte. Per fer-ho anem al menú *Simulation* i escollim la opció *Parameters*, amb la qual cosa se'ns obre una finestra de diàleg en la qual podem canviar l'algorisme d'integració i una sèrie de paràmetres (Veure Figura 3).

Si volem guardar el nostre model seleccionem l'opció *Save* del menú *File*. Al nom que li donem, se li afegirà l'extensió *.m*.

Ara ja podem dur a terme la simulació: anem al menú de simulació i escollim *Start*. Podem veure els resultats immediatament a la finestra *Scope*. També des de la barra de menú de la finestra

Untitled podem prémer el botó .

2. EXEMPLE 2: CIRCUIT ELÈCTRIC

Suposem que volem simular el comportament del circuit elèctric de la Figura 4.

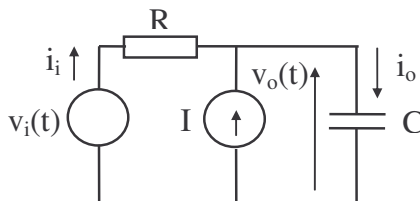


Figura 4. Circuit elèctric

Aplicant la Llei de nusos de Kirchhoff tenim:

$$i_o = i_i + I$$

$$C \frac{dv_o}{dt} = \frac{v_i - v_o}{R} + I$$

$$\frac{dv_o}{dt} = \frac{1}{C} \left(\frac{v_i - v_o}{R} + I \right)$$

on:

$$C = 1 \text{ mF.}$$

$$R = 1 \text{ k}\Omega.$$

Hem de crear, per tant, un conjunt de blocs en el Simulink que ens representin el comportament del circuit elèctric. Aquests blocs poden tenir una estructura com els de la Figura 5. Els blocs sumadors i de guany es troben a la llibreria *Math Operations*, mentre que els ports d'entrada i sortida a la *Ports & Subsystems*.

Comproveu que aquest model es correspon amb l'equació que hem obtingut anteriorment que descriu el comportament del circuit elèctric.

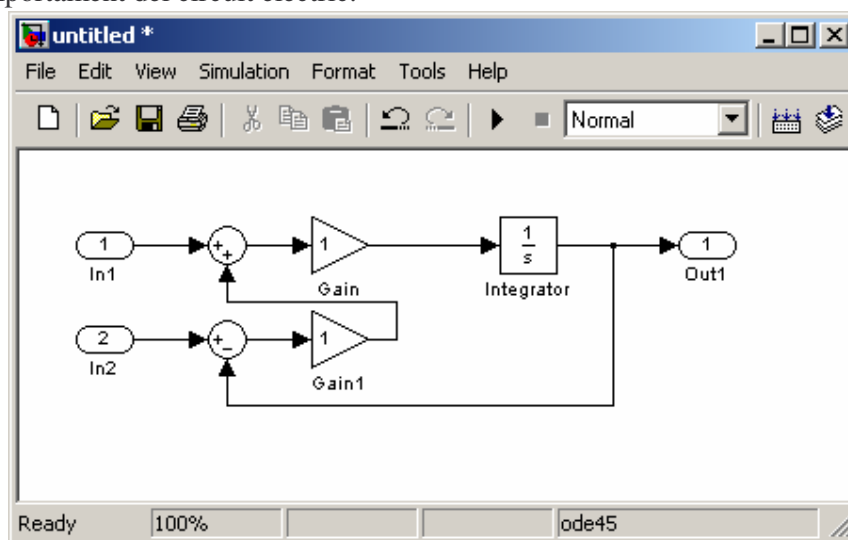


Figura 5. Model del circuit elèctric

Utilitzarem aquest model en un problema de control. Es tracta de controlar la tensió de sortida V_o mitjançant la font d'intensitat I . En concret, es desitja que v_o es mantingui al voltant de 15 V quan

$V_i(t)=10+2\sin 2t$. Per aconseguir-ho farem el control amb un relé que ens connectarà o no la font d'intensitat a la resta del circuit. El diagrama en Simulink d'aquest tipus de control pot ser el que mostra la Figura 6.

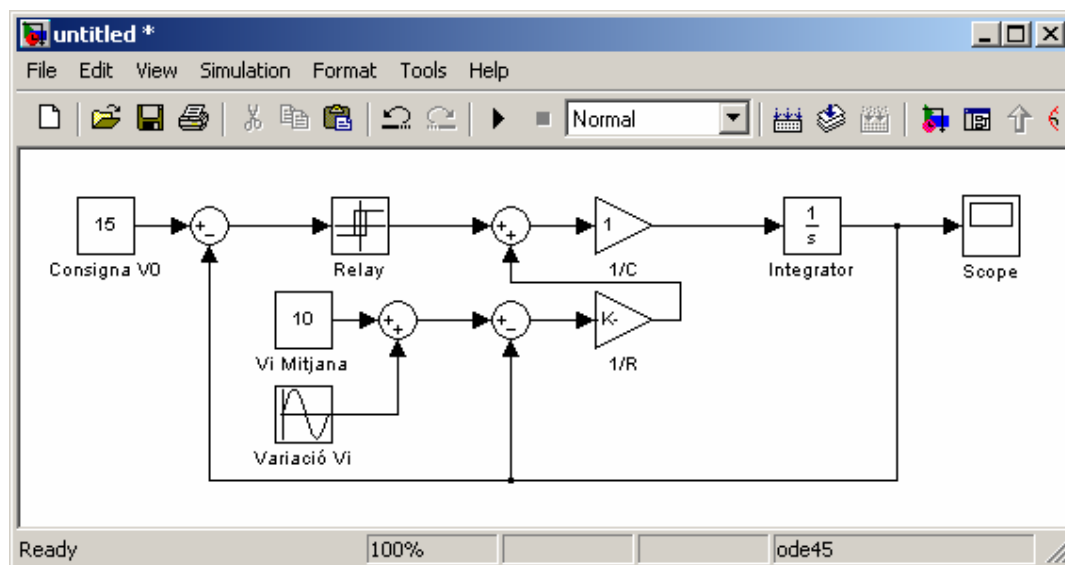


Figura 6. Diagrama del circuit elèctric controlat

En aquesta figura, el relé dóna una sortida de 0.02 A quan l'error supera 1 V i dóna 0 A quan l'error baixa de -1 V. El relé (Relay) es troba a la llibreria *Discontinuities*.

Un cop definits el model del sistema i el model del controlador, podem començar la simulació. Per a això haurem d'ajustar *Parameters* del menú *Simulation*. Podem fer simulacions de 20 s, per exemple, posant un *Max step size* de 0.1 s.

EXEMPLE 3. MODELACIÓ D'UN MOTOR D'IMAN PERMANENT.

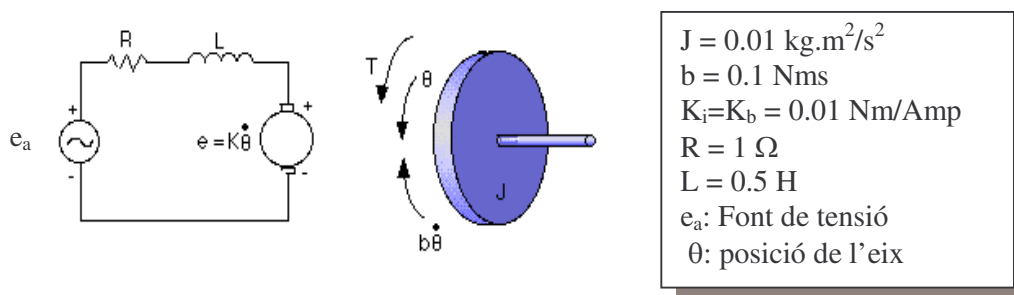


Figura 7. Motor d'iman permanent

Al aplicar una tensió $e_a(t)$ el motor gira un angle $\theta(t)$ a una velocitat angular $\omega = \dot{\theta}$ provocant un parell $T_m(t)$. El parell, T_m , es relaciona amb el corrent d'armadura, i_a , amb un factor K_i , i la força electromotriu, e , està relacionada amb la velocitat angular ω_m per una constant K_b . D'aquesta manera es té

$$T_m(t) = K_i i_a(t) \quad (1)$$

$$e(t) = K_b \omega_m(t) \quad (2)$$

Basant-nos en les lleis de Newton i les de Kirchhoff, de l'esquema del motor donat es pot expressar:

$$T_m(t) = J \frac{d\omega_m(t)}{dt} + b\omega_m(t) \quad (3)$$

$$e_a(t) = i_a(t)R + L \frac{di_a(t)}{dt} + e(t) \quad (4)$$

Utilitzant la transformada de Laplace i combinant les equacions (1), (2), (3) i (4) es pot obtenir

$$G_m(s) = \frac{\Omega_m(s)}{E_a(s)} = \frac{K_i}{(Ls + R)(Js + b) + K_b K_i}$$

on es pren com ha sortida del sistema la velocitat angular ω_m .

Per a introduir el model del motor mitjançant Matlab, i veure com respon en llaç obert a una entrada graó unitari, es farà amb un fitxer m-file, o introduint les instruccions següents una a una des de la línia de comandes.

```
% Valor dels paràmetres
J=0.01;
b=0.1;
Ki=0.01;
Kb=0.01;
R=1;
L=0.5;
num=Ki;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+Ki*Kb)];

% Representació de la resposta al graó dels 3 primers segons.
t=[0:0.01:3];
step(num,den,0:0.1:3)
title('Resposta al grao del sistema en llaç obert')
```

Per a fer el mateix amb Simulink, es pot partir de les equacions proposades anteriorment i d'aquesta manera construir el diagrama de blocs que les representa (Figura 8)

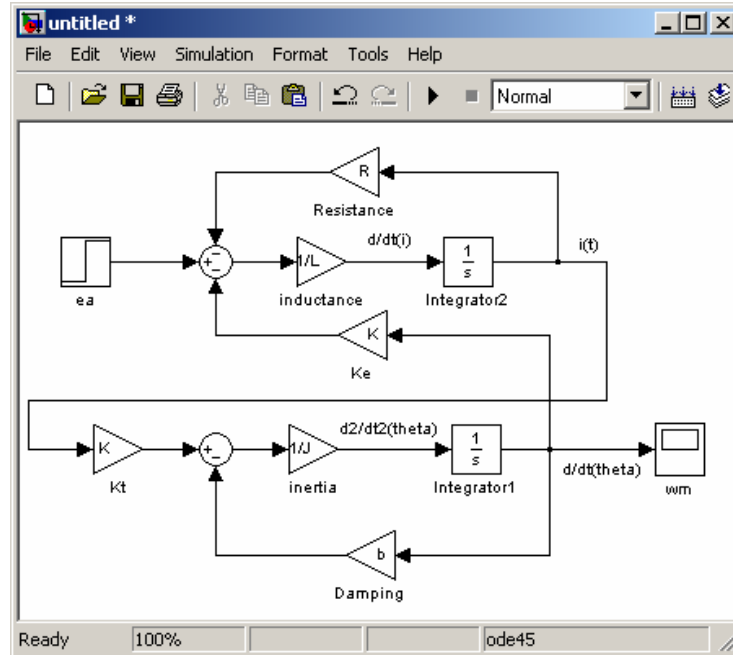


Figura 8. Diagrama Simulink de les equacions del motor d'iman permanent

També es pot modelar el sistema buscant primer la funció de transferència $\frac{\Omega_m(s)}{T_m(s)}$ amb (3), i després afegir els blocs representats per les altres equacions per arribar a un sistema amb entrada $e_a(t)$ i sortida $\omega_m(t)$ (Figura 9)

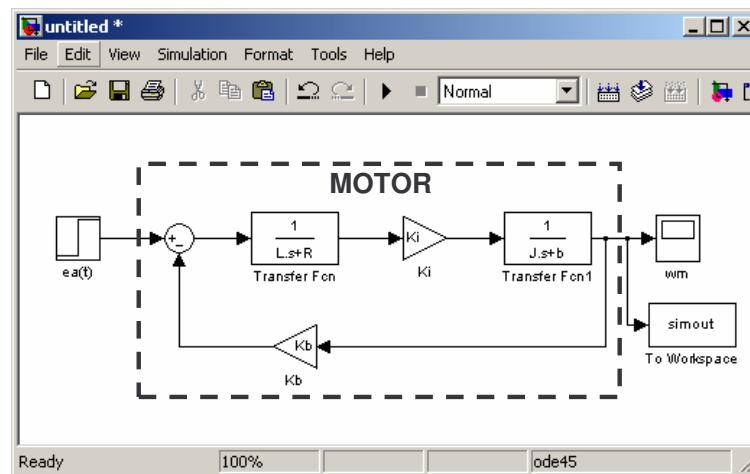


Figura 9. Representació Simulink del motor d'iman permanent utilitzant funció de transferència.

Ara podeu comprovar que la resposta del motor enllaç obert obtinguda a una entrada graó unitari, és la mateixa per al model en Matlab i les dues representacions en Simulink.

Per a simplificar la representació del model es poden agrupar tots els elements que componen el bloc motor en un sol bloc. Per a fer això, seleccioneu tots els elements del motor amb el cursor i seleccioneu *Create Subsystem* de l'opció *Edit* de la finestra *Untitled* (Figura 10).

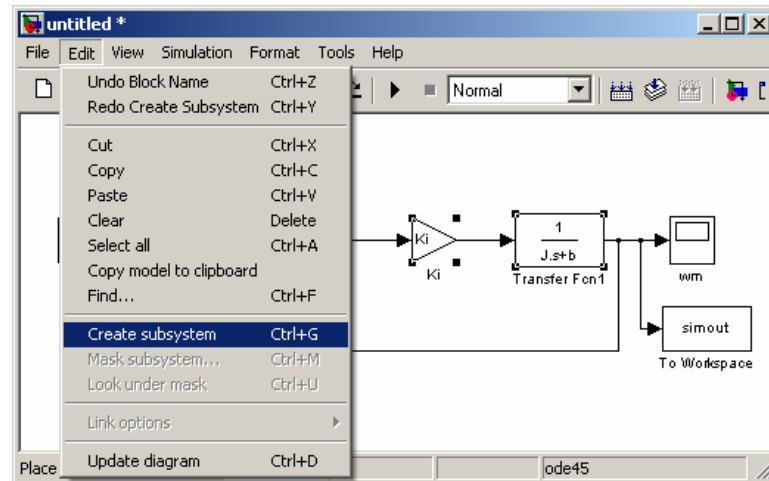


Figura 10. Agrupació en un sol bloc dels blocs del motor

Quedarà l'esquema mostrat a la Figura 11 on *Subsystem* conté tots els blocs corresponents al motor.

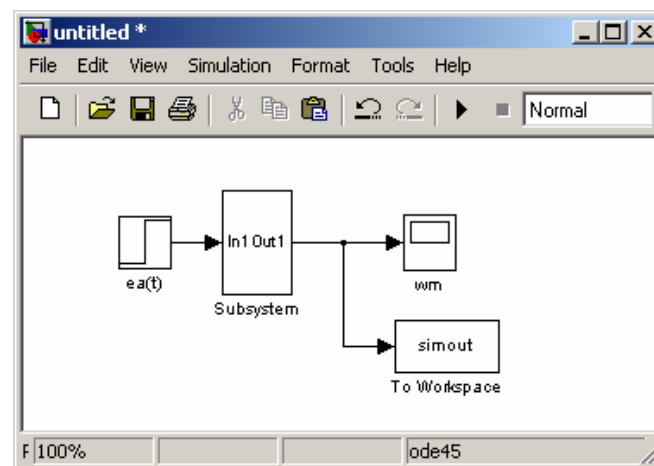


Figura 11. Resultat de l'agrupació dels blocs del motor.

Per a fer un exemple de control de velocitat sobre el motor anterior, es proposa introduir un controlador al sistema amb una funció de transferència

$$G_C(s) = \frac{50s + 50}{s + 0.01}$$

Aprofitant el bloc del motor, realitzeu amb Simulink el sistema global amb el controlador i el procés seguint l'esquema proposat a la Figura 12.

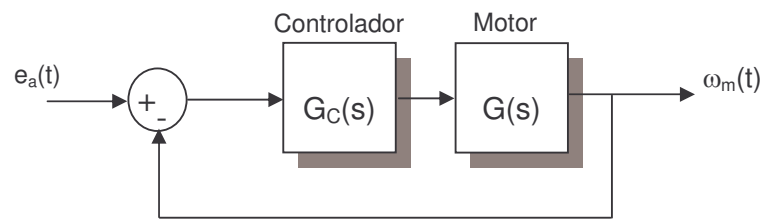


Figura 12. Sistema amb controlador

Comproveu la diferència en la resposta del sistema amb controlador i en llaç obert.

Implementeu el mateix sistema de control de la Figura 12, utilitzant ara instruccions de Matlab, tal com *feedback*, *cloop*, *conv*, *step*, etc. Partiu de la funció de transferència del motor $G(s)$.