# D

# MATLAB, Simulink alapok

Ebben a függelékben nagyon röviden összefoglaljuk a MATLAB® és a Simulink® legfontosa jellemzőit, kezelését, programozási lehetőségeit. A MATLAB® általános használatával kapcsolatban a [16] művet ajánljuk az Olvasó figyelmébe. Az irányítástechnikai problémák megoldásához szükséges parancsok használatát a programcsomag online súgójából célszerű elsajátítani. Emellett a függelékben bemutatjuk néhány nagyon egyszerű szabályozástechnikai probléma megoldását.

## D.1. Változók és utasítások

A MATLAB® egy olyan scriptnyelv, melynek alapértelmezett változótípusa a mátrix. Ebben a kontextusban egy skalárértékű változó 1x1-es mátrixként értelmezhető.

Az első lehetőség MATLAB® használatára a parancsablak (Command window), melyben a » jel után kell az utasítást írni, majd Enter-rel aktiválni. Formája a következő:

```
>>változó=kifejezés
```

ahol a változó konstans, vektor, vagy mátrix lehet, amit a jobb oldali kifejezéssel definiálunk. Például:

```
»a=15;
```

Egy mátrix elemeit többféleképpen adhatjuk meg, vagy az egyes sorok végét „;”-vel jelölve, vagy Enter-rel elválasztva.

```
»A=[1 2 3; 4 5 6; 7 8 9];
```

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

Fontos, hogy a MATLAB® megkülönbözteti a kisbetűt a nagybetűtől, azaz „a” és „A” értékei különböznek egymástól. Nincs külön típusdefiníció, a mátrix mérete és típusa a beírt értékek alapján automatikusan képződik. Az egyes elemek függvényként is megadhatók. Például:

```
»A=[1 sqrt(2); sin(pi/2) exp(0.8)];
```

Természetesen a mátrix értelmezhetőségéről, illetve a mátrixműveletek elvégezhetőségéről a felhasználónak kell gondoskodnia.

A kifejezésben használható matematikai operátorok a következők:

+: összegzés;
-: kivonás;
*: szorzás;
/: jobbról szorzás a mátrix inverzével, skalár változókra osztás;
^: hatványozás.

A fenti matematikai műveletek az egész mátrixra érvényesek, azonban lehetőség van a mátrix elemeire vonatkoztatott műveletek elvégzése is (a műveleti jelek elé pontot téve):

.*: elemenkénti szorzás;
./: elemenkénti osztás;
.^: elemenkénti hatványozás.

Például legyenek

```
»A=[1 2; 3 4]; B=[ 3 4; 1 2];
»A*B=[5 8; 13 20];
»A.*B=[3 8; 3 8];
```

A legfontosabb trigonometriai és elemi matematikai függvények a következők: `sin(x)`: szinusz függvény;
`cos(x)`: koszinusz függvény;
`tan(x)`: tangens függvény;
`abs(x)`: abszolút érték képzés;
`sqrt(x)`: négyzetgyök képzés;
`imag(x)`: képzetes rész értéke;
`real(x)`: valós rész értéke;
`conj(x)`: komplex konjugált képzés;
`log(x)`: természetes alapú logaritmus képzés;
`log10(x)`: 10-es alapú logaritmus képzés;
`exp(x)`: exponenciális képzés.

A MATLAB® a matematikai számításait minden esetben dupla pontossággal hajtja végre, eredményeit viszont alapértelmezésben egyszeres pon-

tossággal írja a képernyőre. A képernyőn megjelenő információ formátuma szintén kiválasztható. Az alábbi példában a pi értékét (ami a MATLAB®-ben beépített, definiált konstans) írjuk ki alapértelmezésben, illetve különböző formátumok szerint.

```
  pi: 3.1416
format long; pi 3.14159265358979
format short e; pi 3.1416e+000
format long e; pi 3.141592653589793e+000
```

Bármelyik MATLAB® utasításról kérhetünk segítséget a parancsablakban. A `help parancsnév` (például `help plot`) és Enter begépelése után részletes leírást kapunk a használat módjáról, a be- és kimenő paraméterekről.

## D.2. Programozás

A második lehetőség a MATLAB® programok futtatása. A programok első csoportja az úgynevezett script fájlok (speciális „m” kiterjesztésű fájlok), melyek az elvégzendő utasításokat szekvenciális sorrendben (egymás után) tartalmazzák. Egy script fájl a nevével aktiválható. A MATLAB® a scriptben lévő utasításokat a megadott sorrendben automatikusan végrehajtja. A script fájl egy ASCII típusú szövegfájl, amely tetszőleges - txt formátumot támogató - szövegszerkesztővel létrehozható, illetve módosítható. Természetesen a MATLAB® is kínál saját szerkesztőt (Editor) a programjaink megírására. A script fájlban megjegyzések is tehetők % jelölés alkalmazásával. A script lényeges jellemzője az, hogy a benne lévő valamennyi változó globális értelmezésű, azaz a script lefutása után valamennyi változó a memória területre kerül.

A MATLAB® másik program típusa a function (függvény), ami abban különbözik a scripttől, hogy a benne lévő változók alapvetően lokálisak, azaz a function lefutása után a memória területre nem kerülnek. Természetesen kivételt képeznek a function bemenő és kimenő adatai, melyeket a function fejlécében kell definiálni. Nézzünk példát a functionra:

```
function y=mean(x)
% Átlagérték számítását biztosítja az x vektorra.
[m,n]=size(x);
if (m>1 && n>1)
disp('Kérem, hogy vektor bemenetet adjon meg!');
return;
```

```
end
if m==1
y=sum(x)/n;
else
y=sum(x)/m;
end
```

A function x bemeneti és y kimeneti változója globális, a többi mind lokális.

A globális változók értéke a parancsablakban közvetlenül is megjeleníthető, például nevének beírásával, majd Enter leütésével. A memória tartalma, azaz az adott alkalmazásban értéket kapott összes változó a „whos" paranccsal listázható ki. Emellett a programjaink futása során létrehozott összes változó és értéke megtekinthető a „munkatérben" (Workspace) is.

Számos törlésre szolgáló parancs található a MATLAB®-ban. A munkatér változói a `clear all` paranccsal törölhetők. A `clear változónév` egy adott változót töröl. A `close all` parancs bezárja az összes MATLAB®-ból megnyitott ablakot, kivéve a főablakot és a szerkesztőt. A parancsablakban megjelenített tartalmak a `clc` paranccsal törölhetők.

## D.3. Grafika

A MATLAB® lehetőséget biztosít az eredmények megjelenítésére is. A legfontosabb rajzolási formátumok a következők:

`plot(x,y)`: x függvényében y értékeit kirajzolja. Számos opcionális bemenő paraméterével testre szabható a görbe megjelenése.

`semilogx(x,y)`: az (x,y) függvény értékeit tízes alapú logaritmikus x tengelyű koordináta rendszerben rajzolja ki.

`semilogy(x,y)`: az (x,y) függvény értékeit tízes alapú logaritmikus y tengelyű koordináta rendszerben rajzolja ki.

`loglog(x,y)`: az (x,y) függvény értékeit olyan koordináta rendszerben rajzolja ki, melynek mindkét tengelye tízes alapú logaritmikus.

Megválasztható a függvény színe és a vonaltípusa. A vonaltípus a következő lehet: „-": folytonos;
„- -": szaggatott vonal (a két kötőjel között nincs szóköz);
„.-": pont-vonal;
„:": pontokból álló görbe.

A görbe karakteres is lehet: *, o, x, + karakterekkel. A `plot` parancs további opcionális paramétere a `'Linewidth'`, mellyel az adott görbe vo-

nalvastagságát adhatjuk meg pixelben. A vastagságot vesszővel elválasztva kell megadni, lásd a példában.

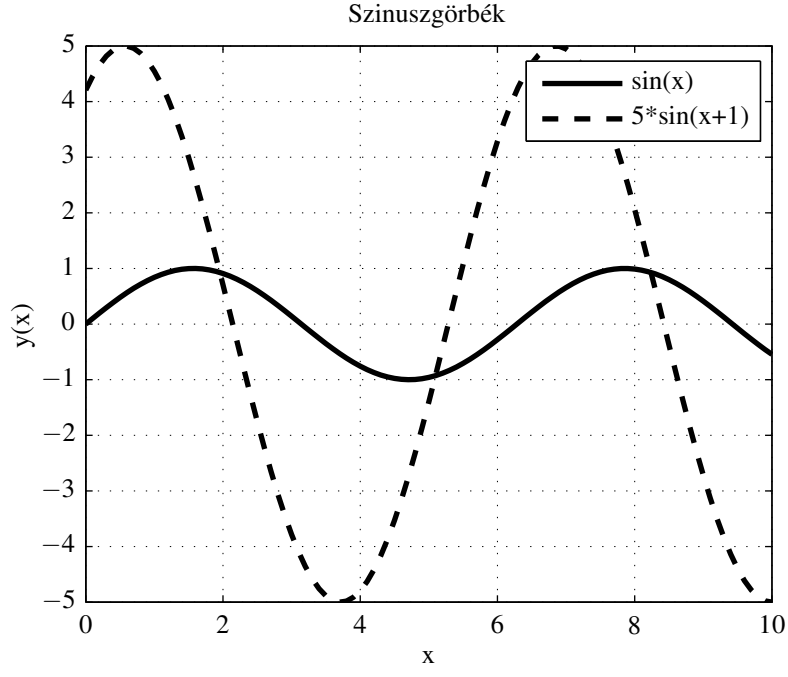
A grafikus ábra a felhasználó igényének megfelelően egyéb információkkal is kiegészíthető.

`title('text')`: ábra címe,
`xlabel('label')`: x tengely felirata,
`ylabel('label')`: y tengely felirata,
`text(p1,p2,'text')`: a grafikus ábra (p1,p2) koordinátájára megadott szöveg kiírása,
`legend('text1', 'text2')`: jelmagyarázat elhelyezése,
`subplot`: az ábra felosztása több koordinátarendszerre (egy ablakban több ábra),
`grid on`: rács megjelenítése az ábrában (ellentéte a `grid off`).
`hold on`: ez a parancs jelzi, hogy az utána következő grafikonokat mindaddig ugyanabba az ábrába kell rajzolni, amíg a `hold off` parancs nem következik, vagy új ábrát nem nyitunk a figure paranccsal.

Például rajzoljunk két szinusz görbét ugyanabba az ábrába más-más vonaltípussal. Jelenítsük meg a jelmagyarázatot is.

1. példa

```
x=[0:0.1:10]';
y1=sin(x);
y2=5*sin(x+1);
figure;
plot(x,y1,'k-','LineWidth', 2);
hold on;
plot(x,y2,'k--','LineWidth', 2);
title('Szinuszgorbek');
xlabel('x');
ylabel('y(x)');
grid on;
legend ('sin(x)', '5*sin(x+1)');
hold off;
```




D.1. ábra. 1. példa: Szinuszgörbék rajzolása

## D.4. Szabályozási feladat

Egy szabályozott rendszer modelljét megadhatjuk átviteli függvény alakban. Például az alábbi átviteli függvény esetén $G(s) = \dfrac{20s+10}{s^2+4s+3}$ a modell megadása a következő:

```
num=[20 10];
den=[1 4 3];
```

A rendszer pólusai és zérusai a következő utasítással határozhatók meg:

```
[z,p,k]=tf2zp(num,den);
```

ahol z a zérusokat, p a pólusokat tartalmazó vektor, míg k az erősítés értéke.

A példában `z=-0.5; p=[-3; -1]; k=20;`

A `sys=tf(num,den)` paranccsal létrehozható egy átmeneti függvény, mely bemenő paraméterként szolgál további parancsoknak.

A rendszer időtartományi analízisében a súlyfüggvény és az átmeneti függvény játszik fontos szerepet. Ezek a következő utasítással számíthatók ki.

```
[g,tg]=impulse(sys);
[v,t]=step(sys);
```

A fenti utasításokban eredményül két-két vektort kapunk, melyek az súlyfüggvény (átmeneti függvény) értékeit, valamint az időértékeket tartalmazzák.

Az időtartományi függvények jellemzői az állandósult állapotbeli érték, a beállási idő (az időtartam ami után a kimenőjel az állandósult állapot 5%-os tűrésén belül marad), a növekedési idő (az időtartam ami ahhoz szükséges, hogy a kimenőjel az állandósult állapot 10%-áról és 90%-ra eljut), a maximális túllendülés időpontja és annak százalékos értéke.

2. példa: ábrázoljuk a $G(s) = \dfrac{2}{s^2 + s + 9.25}$ átviteli függvénnyel adott rendszer súly- és átmeneti függvényeit! Számítsuk ki a következő időtartományi paramétereket: állandósult érték, maximális túllendülés, növekedési idő, beállási idő.

```
num=2;
den=[1 1 9.25];
sys=tf(num,den);
% Suly- es atmeneti fuggvény szamitasa
[g,tg]=impulse(sys);
[y,t]=step(sys);
figure;
plot(tg,g, 'k-','LineWidth', 2);
xlabel('t (s)');
ylabel('g(t)');
title('Súlyfüggvény');
figure;
plot(t,y, 'k-','LineWidth', 2);
xlabel('t (s)');
ylabel('v(t)');
title('Átmeneti függvény');
% Allandosult ertek szamitasa
y_all=polyval(num,0)/polyval(den,0);
% Maximalis tullendules
[Y,k]=max(y);
```

```
t_tullend=t(k);
p_tullend_szazalek=100*(Y-y_all)/y_all;
% Novekedesi ido
n=1;
while y(n)<0.1*y_all
n=n+1; end;
m=1;
while y(m)<0.9*y_all
m=m+1; end;
t_novek=t(m)-t(n);
% Beallasi ido
l=length(t);
while (y(l)>0.95*y_all) & (y(l)<1.05*y_all)
l=l-1; end;
t_beall=t(l);
```

Jellemző értékek:

- állandósult állapotbeli érték: 0.2162
- beállási idő: 5.515 s
- növekedési idő: 0.39 s
- max. túllendülés időpontja: 1.05 s
- max. túllendülés százaléka: 59.2%

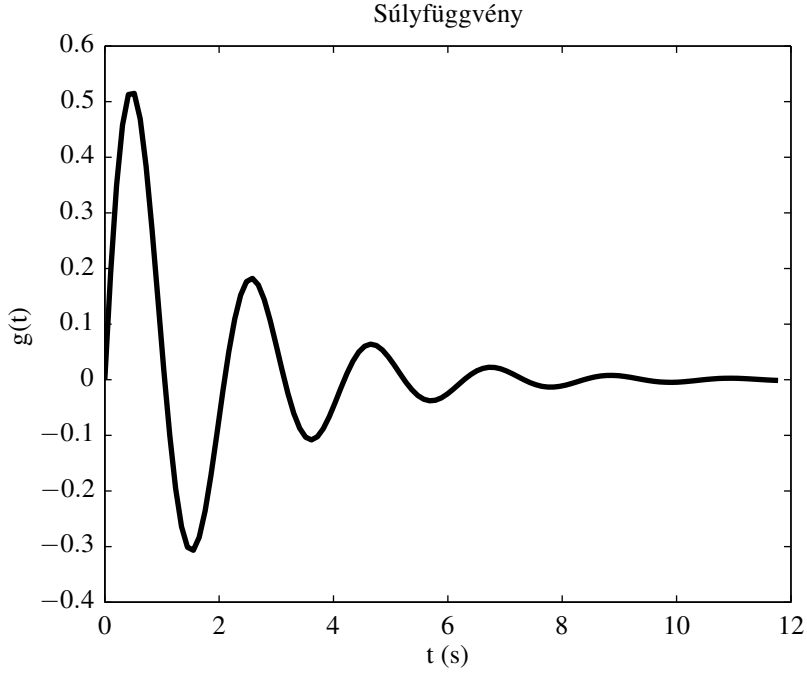
A rendszer frekvenciatartományi analízise a Nyquist- és a Bode-diagramok alapján történik, melyek számítása a következőképpen történik:

```
[mag,phase,w]=bode(num,den);
[re,im]=nyquist(num,den);
```

ahol mag az amplitudó, phase a fázis, w a frekvencia tartomány, re a valós rész, míg im a képzetes rész.

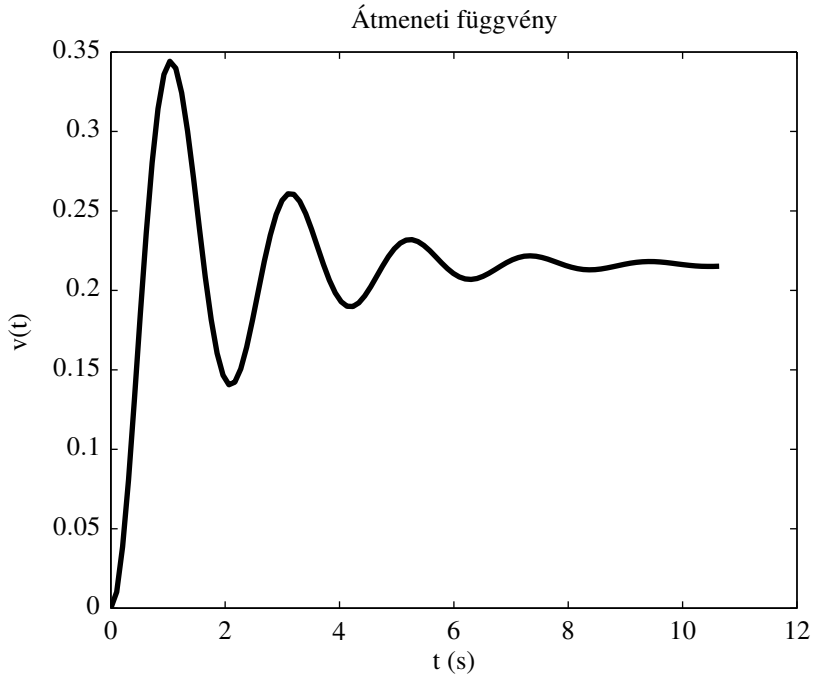
3. példa: rajzoljuk fel a $G(s) = \dfrac{20}{s^2+s+9.25}$ átviteli függvény frekvencia függvényeit.

```
   % Atviteli függvény
num=20;
den=[1 1 9.25];
% Nyquist diagram
```




D.2. ábra. 2. példa: Súlyfüggvény rajzolása

```
[re,im]=nyquist(num,den);
figure;
plot(re,im,'k-','LineWidth', 2);
xlabel('Re');
ylabel('Im');
grid off;
title('Nyquist-diagram');
% Bode diagram
w=logspace(-1,2,100);
[mag,phase]=bode(num,den,w);
mag=20*log10(mag);
figure;
subplot(2,1,1);
semilogx (w,mag,'k-','LineWidth', 2);
```




D.3. ábra. 2. példa: Átmeneti függvény rajzolása

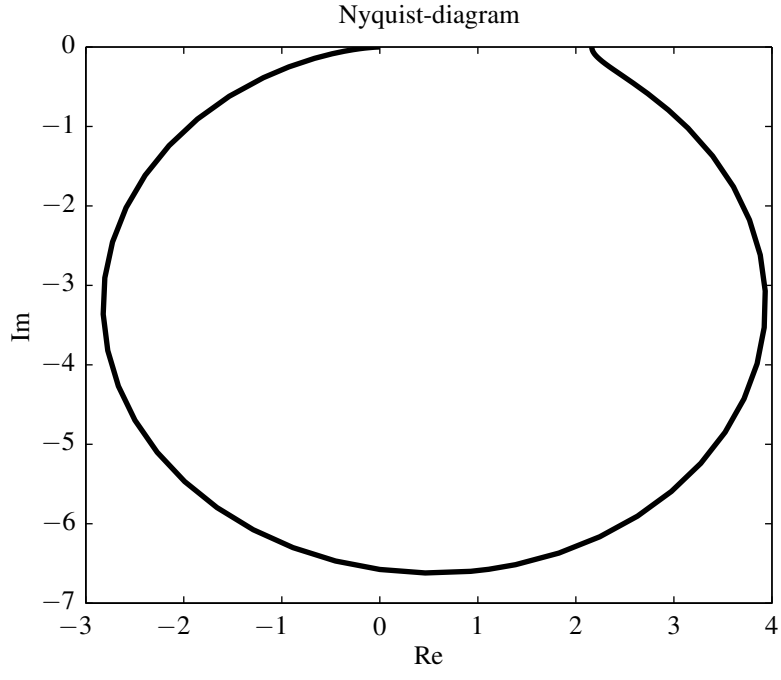
```
title('Bode-diagram');
ylabel('Amplitudó (dB)');
subplot(2,1,2);
semilogx(w,phase,'k-','LineWidth', 2);
xlabel('Frekvencia (rad/s)');
ylabel('Fázis (fok)');
```

Megadható a rendszer állapottér-reprezentációs alakban is:

```
A=[-4 -3; 1 0];
b=[ 1; 0];
c=[20 10];
d=0;
```

Az állapottér reprezentációs alakból egy utasítás kiadásával áttérhetünk átviteli függvényre

```
[num,den]=ss2tf(A,b,c,d);
```




D.4. ábra. 3. példa: Nyquist-diagram rajzolása

és viszont az átviteli függvény egy utasítással állapottér reprezentációs alakba vihető.

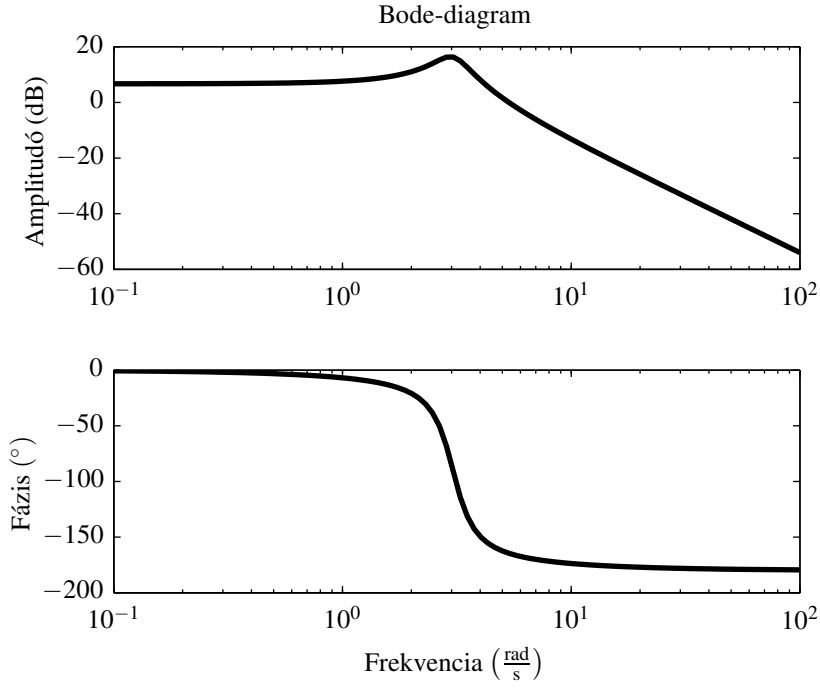
```
[A,b,c,d]=tf2ss(num,den);
```

Továbbá az állapottér reprezentációs alakok között egy nemszinguláris T transzformációs mátrix megadásával hasonlósági transzformáció hajtható végre.

```
[At,bt,ct,dt]=ss2tf(A,b,c,d,T);
```

4. példa: végezetül határozzuk meg a fenti rendszer irányíthatóságát és megfigyelhetőségét.

```
   % Allapotter reprezentacio
A=[-4 -3; 1 0];
b=[ 1; 0];
c=[20 10];
d=0; % Iranyithatosag vizsgalata
```




D.5. ábra. 3. példa: Bode-diagram rajzolása

```
model_rend=length(b);
C=ctrb(A,b);
rang_control=rank(C);
if model_rend==rang_control
disp('A rendszer iranyithato');
else
disp('A rendszer nem iranyithato');
end;
% Megfigyelhetoseg
O=obsv(A,c);
rang_obser=rank(O);
if (model_rend==rang_obser)
disp('es megfigyelheto');
else
```

```
disp('es nem megfigyelheto');
end;
```

Eredmény: a rendszer irányítható és megfigyelhető.

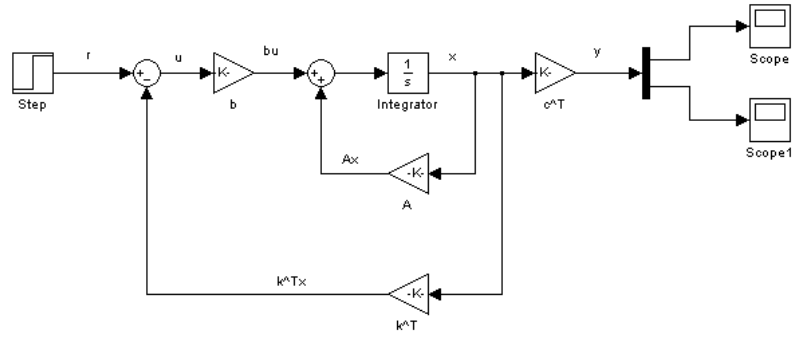
## D.5. Simulink

Az előzőekben a MATLAB® program irányítástechnikai alkalmazásainak egyes lehetőségei kerültek bemutatásra. Ezek segítségével rendszerek elemzésére és szabályozás tervezésére soros kódban megírt programok készíthetőek. Lineáris rendszerek analízisére és szabályozására ezek a módszerek tökéletesen megfelelőek lehetnek, viszont bonyolultabb esetben néhol nehézkesen használhatók.

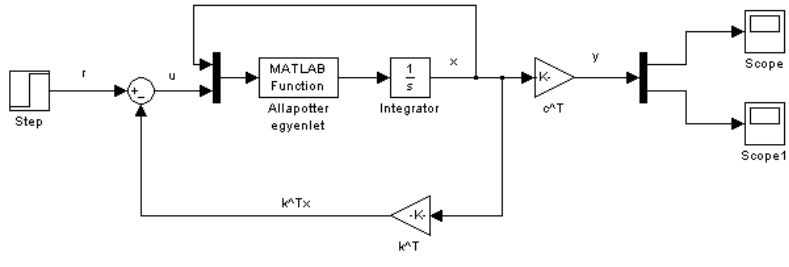
A MATLAB® lehetőséget nyújt a Simulink® programcsomag segítségével az általunk a jármű és közlekedéstechnikában használt rendszerek szemléletesebb modellezésére és szabályozására. A MATLAB® Simulink® egy felhasználóbarát, könnyen kezelhető programcsomag, amelyben a rendszer egyes részei és vizsgáló elemei blokkokból épülnek fel. Ilyen blokkok lehetnek például az erősítések (Gain), multiplexerek (Mux), vizsgáló dobozok (Scope), jelek integrálását és differenciálását elvégző elemek (Integrator és Derivative), jelgenerátorok (Sine Wave, Step) stb. A blokkokat irányított vonalak kötik össze, melyek vektorba rendezett jelek.

A következő ábrákon az irányítástechnikában használatos teljes állapot visszacsatolással megvalósított általános irányítási blokkséma látható kétféleképpen megvalósítva Simulink®-ben. A D.6 esetben az állapottér felírásban szereplő $A$, $b$, $c^T$ és $k$ vektorok/mátrixok Gain blokkban jelennek meg számszerűleg. Az Integrator blokk feladata $\dot{x}$ állapotvektor derivált integrálása, melyben a rendszer állapotának kezdeti értékei beállíthatók. A mért $y$ jelek Demux blokkal különválaszthatók, és egyenként Scope-pal vizsgálhatók. A D.7 esetben a rendszer egyes részei soros kódban jelennek meg a Simulink® modell fájlon belül. A MATLAB® Function blokk alá beilleszthető egy általunk megírt m-fájl, ami a model fájllal azonos könyvtárban szerepel. Ennek minden esetben függvénynek kell lennie, tehát egy ehhez hasonló első sorral (preambulummal) kell rendelkeznie: `function output=fuggveny(input)`, ahol `function` a függvény parancs, `output` a MATLAB® Function kimenete, `input` a bemenete, `fuggveny` pedig a függvény (m-fájl) neve. D.7 esetében a MATLAB® Function tartalmazza $A$ és $b$ mátrixokat, és $Ax$ számításához szükséges visszavezetni a függvény bemeneteként az állapotokat. Mivel $bu$ számítása is szükséges a

függvényen belül, ezért $x$ és $u$ egyaránt szükséges, mint bemenet. Mivel a MATLAB® Function blokknak egyetlen bemenő vektora van, ezért egy Mux blokkal $x$ és $u$ egyesítése szükséges, mely a függvényen belül ismét szétbontható. Ugyanez igaz a kimenetekre is, azaz a függvényen belül az egyes kimenetek egy vektorba rendezése szükséges.




D.6. ábra. Rendszer Gain blokkokban




D.7. ábra. Rendszer függvény felhasználásával