SZÁMÍTÓGÉPES MODELLEZÉS ÉS SZIMULÁCIÓ 1

című tananyag

DR. SIMON JÁNOS, Főiskolai docens

Szegedi Tudományegyetem - Mérnöki Kar

Műszaki Intézet

2018.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Tartalomjegyzék

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

Készült az EFOP-3.5.1-16-2017-00004 pályázat támogatásával.

Írta:
Dr. Simon János

Szerkesztette:
Dr. Simon János

Lektorálta:
Dr. Szakáll Tibor

Kiadta:

Szegedi Tudományegyetem, Mérnöki Kar – Szeged (MAGYARORSZÁG), 2018

ISBN: 978-963-306-618-8

ELŐSZÓ

A tananyag segítséget nyújt a Szegedi Tudományegyetem - Mérnöki Kar hallgatóinak a Számítógépes modellezés és szimuláció 1 című tantárgy elméleti és gyakorlati részét való elsajátításához. Tartalmilag teljes mértékben lefedi a tantárgy akkreditált programját. A kézikönyv az elméleti rész elsajátítását példák bemutatásával segíti. További példák az mellékelt anyag ismereteinek elmélyítését szolgálják. Több esetben, a szakmai körökben elterjedt SCILAB számítógépes program alkalmazása is bemutatásra kerül. Külön köszönet azoknak a hallgatóknak, akik munkájukkal hozzájárultak a jegyzet elkészítéséhez.

A „Számítógépes modellezés és szimuláció 1” jegyzet 126 oldalon foglalja össze a témakör elméleti és gyakorlati elemeit. A jegyzet 9 fejezetből épül fel. Az elméleti részhez és a változatos témakörből válogatott, kidolgozott gyakorlati feladatokhoz megfelelő mennyiségű és minőségű illusztráció társul, amely nagymértékben hozzájárul a tananyag elsajátíthatóságához. A szöveg megfelelően szerkesztve, átláthatóan foglalja össze a tananyagot. Szerkezeti, szakmai és nyelvhelyességi szempontból megfelel egy egyetemi jegyzet iránt támasztható követelményeknek. Tartalmilag egy átfogó leírást tartalmaz a SCILAB alkalmazáscsomagról, amely egy mérnöki és tudományos számításokat támogató környezet. A bemutatásra kerülő egységek és feladatok kiválasztása tükrözik a szerző témában való jártasságát és a több éves szakmai tapasztalatát. A jegyzet gördülékenyen mutatja be – változatos területekről gyűjtött példákon keresztül – a témával kapcsolatos alapokat, valamint a lényeges és a gyakorlatban fontosnak tartott dolgokat. Elméleti és gyakorlati elemeken keresztül ad képet a SCILAB modellezésre és szimulációra való felhasználásáról és gyakorlati alkalmazásáról. A munka 9 fejezetében megtalálhatóak mindazok az elemek, amelyek szükségesek ahhoz, hogy egy átfogó képet kapjunk a SCILAB környezet számítógépes modellezésre és szimulációra történő felhasználásának fontosabb aspektusairól.

Az első két fejezetben bevezetőként bemutatásra kerülnek a SCILAB csomag alapjai, az általa kezelt adattípusok, az aritmetikai kifejezések és vezérlőszerkezetek. A fejezetek ellenőrző kérdésekkel zárulnak.

A harmadik fejezetben a grafikus lehetőségek vannak bemutatva, ahol a példafeladatok az adatok és függvények 2D és 3D bemutatására helyezik a hangsúlyt.

A negyedik fejezet az integrált fejlesztőkörnyezetben történő programozást mutatja be gépészet (ferde hajítás, fogaskerék áttétel), fizika (ferde hajítás) és más területekről gyűjtött

feladatokon keresztül. A szimulációk eredménye az előző fejezetben megismert grafikon elem felhasználásával történt.

Az ötödik és hatodik fejezet az XCOS eszközt mutatja be, amely mechanikai rendszerek, hidraulikus körök, szabályzó és irányító rendszerek modellezésre alkalmas. Az elméleti rész az átviteli függvényekről és a PID szabályzókról szól. A kidolgozott feladatok pedig a modell felállítást és az ezeken történő vezérlési szimulációkat ismertetik több mint tíz példán keresztül.

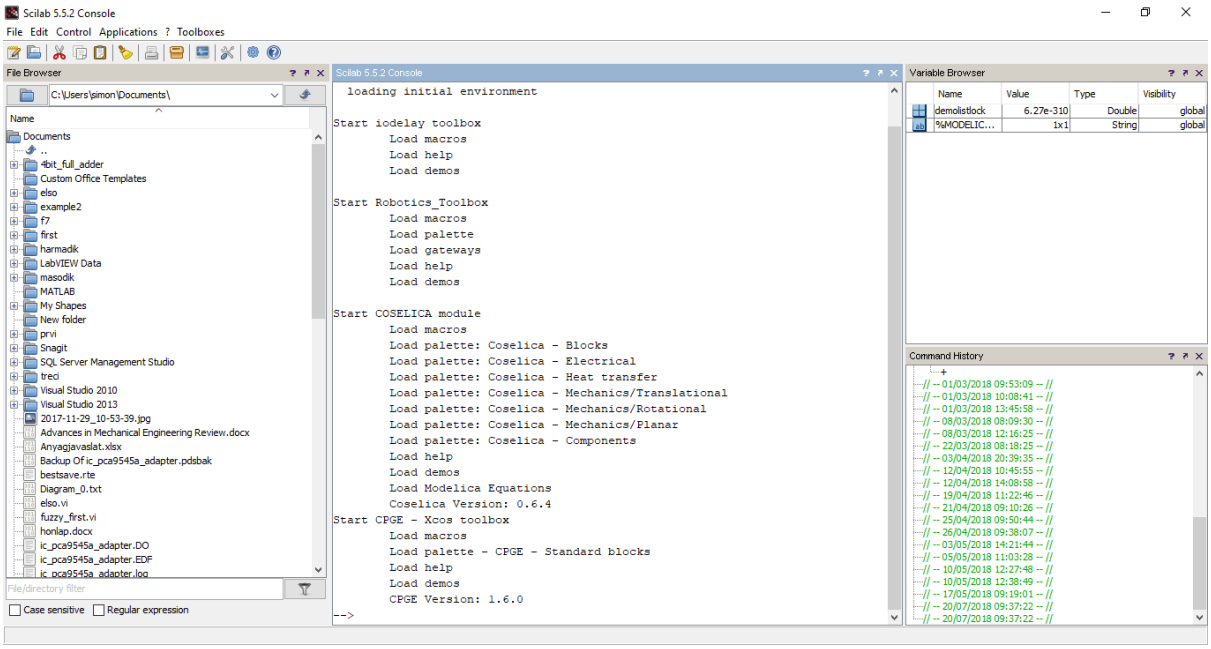
A hetedik fejezet a mechatronikai rendszerek dinamikus modellezésére felhasználható COSELICA nyelvet és modelleket tárgyalja. A fejezet közel 20 elektronika, gépészeti és termikus modell szimulációját tartalmazza.

A nyolcadik és kilencedik fejezet a robotika területén használatos ROBOTICS TOOLBOX függvényeit és felhasználását tárgyalja. A nyolcadik fejezet példái adott konfigurációjú robotkarok modelljének tervezését, míg a kilencedik a robotkarok mozgásterveit dogozza fel.

Dr. Simon János

1. SCILAB ALAPOK

A SCILAB nyílt forráskódú, ingyenes, interaktív, a tudományos és műszaki számítások, valamint a számítási eredmények vizuális megjelenítésének támogatására alkalmas szoftver. Olyan eszközöket foglal magában, amelyek alkalmassá teszik a numerikus analízis, a mátrixalgebra, a jelfeldolgozás és a grafikus ábrázolás leggyakoribb feladatainak megoldására [3]. Alapvető adattípusa a mátrix, amelyet nem kell deklarálni. Segítségével összetett numerikus problémákat oldhatunk meg úgy, hogy a feladatot úgyszólván a matematikában szokásos módon írjuk le, mátrix és vektorműveleteket is alkalmazva, megkímélve magunkat hagyományos értelemben vett programírás időrabló munkájától. így a feladataink megoldásához annak az időnek a töredéke szükséges csak, mint ami a megfelelő C-program előállításához kellene.



1.1 ábra A SCILAB munkafelülete

A Scilab elindításakor a parancsértelmező ablakban az alábbi üzenetet jelenik meg,

```
Startup execution:
loading initial environment
-->
```

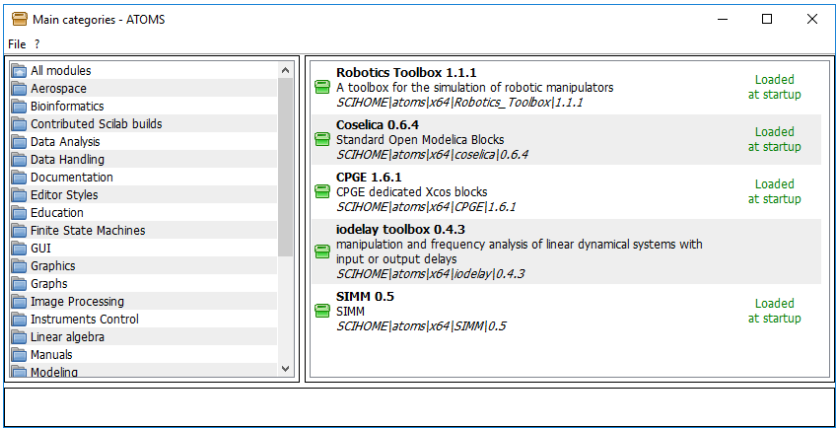
A Scilab különbséget tesz nagy- és kisbetűk között. Ha nem szeretnénk, hogy a Scilab minden utasítás végrehajtása után beszámoljon az eredményről, akkor az utasítás után tegyünk „;”-őt (pontosvesszőt).

Bármely változó értékét egyszerűen lekérdezhetjük nevének begépelésével, A rendelkezésre álló változók a Scilab felületének Variable Browser részében táblázatos formában kerülnek nyilvántartásra.

A jegyzetben bemutatott feladatok sikeres futtatásához legalább 5.5.2 Scilab 32bit/64bit szükséges az alábbi eszközökkel együtt:

- Robotics toolbox
- Coselica
- CPGE
- Iodelay toolbox
- SIMM

A Coselica igényel egy külső C fordítót is, amely lehet egy Visual Studio / Visual C++ Express 2013 vagy egy LCC-Windows C fordító.



1.2 ábra A SCILAB ATOMS telepítő rendszer

Nagyon fontos, hogy a beépített változók elé mindig % jelet kell tenni, például:

```
%pi = 3.14159265358979311599800
```

A legtöbb utasításhoz, adott típusú műveletet végző függvények csoportjához és a modulokhoz is tartoznak mintapéldák, a működés megértését elősegítő egyszerű programok.

Ellenőrző kérdések:

1. Mi a SCILAB?
2. Hogyan történik az eszköztárak letöltése?
3. Miként hivatkozunk a beépített változókra?
4. Milyen külső C fordítót igényel a Coselica?
5. Hol találhatóak a felhasználó által definiált változók?

2. ADATTÍPUSOK, ARITMETIKAI ÉS LOGIKAI KIFEJEZÉSEK

A SCILAB alapvető adattípusa a mátrix. A mátrix lehet négyzetes vagy téglalap alakú, valós, vagy komplex, teljesen kitöltött vagy ritka. Skalárok, vektorok, szövegek ábrázolása ugyanazzal a mátrixstruktúrával történik. Így a skalárnak csak egy sora és egy oszlopa van [14].

Figyelem! A mátrixindexek kezdőértéke a SCILAB-ban mindig 1.

A változók egy névvel ellátott tárolóban megőrizhetők értékek, melyek előhívhatók és felhasználhatók későbbi számításokhoz. Pl.:

```
-->a=2
a  =


    2.
```

A mátrix megadásához a sor komponenseit, üres hellyel vagy vesszővel, az egyes sorokat „;” jellel választjuk el egymástól. Pl.:

```
--> A=[1 0 3 4; 4 3 0 1; 1 0 3 -4]
A  =

    1.    0.    3.    4.
    4.    3.    0.    1.
    1.    0.    3.  - 4.
```

Az alapvető matematikai műveletekhez az alábbi műveleti jeleket használhatjuk.

Művelet	**Jele**
Összeadás	+
Kivonás	-
Szorzás	*
Osztás	/
Hatványozás	^
Transzponálás	`

A műveletek precedenciája a következő táblázatban adott.

Műveletek sorrendje	**Műveletek**
Elsődleges	Negáció, hatványozás
Másodlagos	Szorzás, osztás
Harmadlagos	Összeadás, kivonás

Amennyiben el akarunk térni a precedencia szabálytól, zárójelet kell alkalmaznunk.

A következő táblázatban a rendszer által ismert, a leggyakrabban használatos függvények kerülnek felsorolásra.

Függvény	**Leírása**	**Függvény**	**Leírása**
abs	abszolút érték	nnz	mátrix nem nulla elemeinek száma
acos	arkusz koszinusz	norm	mátrix norma
and	(&) logikai és	not	(~) logikai nem
ceil	kerekítés felfelé	ones	egyesekkel feltöltött mátrix
conj	komplex szám konjugáltja	or	(|) logikai vagy
cos	koszinusz	prod	szorzat
cotg	kotangens	rand	véletlen szám generátor
exp	e^x	real	valós rész
eye	egység mátrix	round	kerekítés
factorial	faktoriális	sign	előjel
fix	kerekítés nulla felé	sin	szinusz
imag	képzetes rész	gsort	rendezés csökkenő sorrendbe
int	egész rész	sqrt	négyzetgyök
log	természetes logaritmus	sum	sum (sor összegzés, oszlop összegzés) vektor/mátrix
log10	tízes alapú logaritmus	tan	tangens
log2	kettes alapú logaritmus	zeros	zérus mátrix
max	maximum	pmodulo	pozitív aritmetikai maradék képzése
min	minimum		

If ... then ... else ... vezérlőszerkezet

Ha az if után szereplő kifejezés logikai értéke igaz, akkor a *then* utáni utasítások futnak le, egyébként az *else* utániak.

Példa:

0 és 100 közzé eső véletlen számok páros-páratlan vizsgálata.

```
n=round(100*rand());
if modulo(n,2)==0 then
    disp("paros szam ",n)
else
    disp("paratlan szam ",n)
end
```

A ciklusok szervezésére a SCILAB két utasítást is tartalmaz, a *for* és a *while* utasításokat.

A FOR ciklus

A *for* utasítás előre meghatározott változó, az ún. ciklusváltozó alapján szervezi a ciklust előre megadott lépésszámra.

Példa:

A megadott szám faktoriálisának kiszámítása.

```
n=input("faktorialis: ");
fakt=1;
if n < 12 then
    for f=1:1:n,
        fakt=fakt*f;
    end
else hiba = "n értéke nagyobb 11-nél!"
    exit // kilépés a programból
end
disp(fakt)
```

Töltsük fel egy a(5,5) mátrix főátlóját 2-sel, a főátló feletti részt 3-mal, és az alatti részt 1-el! Ehhez két egymásba ágyazott for ciklust használunk az alábbi példában látható módon.

```
for i = 1:1:5
    for j = 1:1:5
        if i == j then a(i,j) = 2;
        elseif i > j then a(i,j) = 1;
        else a(i,j) = 3;
        end
    end
end
disp(a)
```

Eredményként az alábbi mátrixot kapjuk.

```
    2.    3.    3.    3.    3.
    1.    2.    3.    3.    3.
    1.    1.    2.    3.    3.
    1.    1.    1.    2.    3.
    1.    1.    1.    1.    2.
```

A WHILE ciklus

A *while* utasítás használata esetén nem kell lerögzítenünk a ciklus ismétlődésének számát. A *while* ciklus befejezését feltétel(ek) vizsgálatával vezérelhetjük.

Példa:

Véletlen számok /rand() utasítás/ összeadása addig, míg az összeg kisebb, mint 100.

```
n=0;
while n<100
    n=n+rand();
end
disp(n)
```

A SELECT programszerkezet

Többirányú elágazás egy egész típusú aritmetikai kifejezés lehetséges értékei szerint. A lehetséges értékek egész típusú konstansok. A *case* utáni egész konstansoknak különbözőeknek kell legyenniük. Arra a *case*-ra kerül a vezérlés, melynek értékével egyezik a kifejezés értéke. Ha egyikkel sem, a *default* ágra kerül a vezérlés.

Példa:

Számológép alapművelet végrehajtása két megadott érték közöztt.

```
hibajelzes = "Hibás a műveleti jel";
a=input("1. adat: ");
b=input("2. adat: ");
muv=input("műveleti jel: ","s");
select muv
    case '+' then ered = a+b
    case '-' then ered = a-b
    case '*' then ered = a*b
    case '/' then ered = a/b
else hibajelzes
end
disp(ered)
```

Adatok kiíratása fájlba

Gyakran előfordul, hogy egy program működéséhez szükséges adatokat egy másik program korábban már előállította, vagy a mi programunk által kapott valamilyen eredményt egy másik program használ a későbbiekben. Ilyenkor, hogy ne kelljen kézzel lejegyezni azokat és újra és újra begépelni, külső adathordozóra érdemes tárolni az adatokat, illetve onnan betölteni, beolvasni.

Fájl megnyitása írásra

```
[fájl változó neve]=mopen([fájl teljes elérési útja,
neve],'wt')
```

Írás a megnyitott fájlba

```
mfprintf([fájl változó neve],"[formátum]", [változó]);
```

Fájl lezárása

```
mclose([fájl változó neve])
```

Példa:

10 véletlen szám kiíratása fájlba

```
fd=mopen("random.txt",'wt');
for i=1:10
    szam=rand();
    mfprintf(fd,"%g\n",szam)
end
mclose(fd);
```

Mivel nincs elérési út megadva, így a konzolban aktuális könyvtárban hozza létre a random.txt állományt.

Konzol műveletek

A konzolról bekérhetünk adatokat az input parancs segítségével és eredményt írhatunk ki a write parancs segítségével. Az alábbi példa ezeket a lehetőségeket mutatja be egy háromszög területét kiszámoló program segítségével a bevitt adatok alapján.

```
clear,clc;
   write(%io(2),'Ez egy interaktív demó.');
   write(%io(2),'Meg kell adni a háromszög bázisát');
   write(%io(2),'es a magasságát. Miután a Scilab');
   write(%io(2),'kiszámolja a háromszög területet.');
   write(%io(2),' ');          // Üres sor
b = input('Add meg a háromszög bázisát: ');
h = input('Add meg a háromszög magasságát: ');
   write(%io(2),' ');         // Üres sor
disp(['A háromszög területe = ' string(b*h/2)])
```

Átviteli függvények kivizgálása SCILAB környezetben

Az átviteli függvények a következőképpen is definiálhatóak.

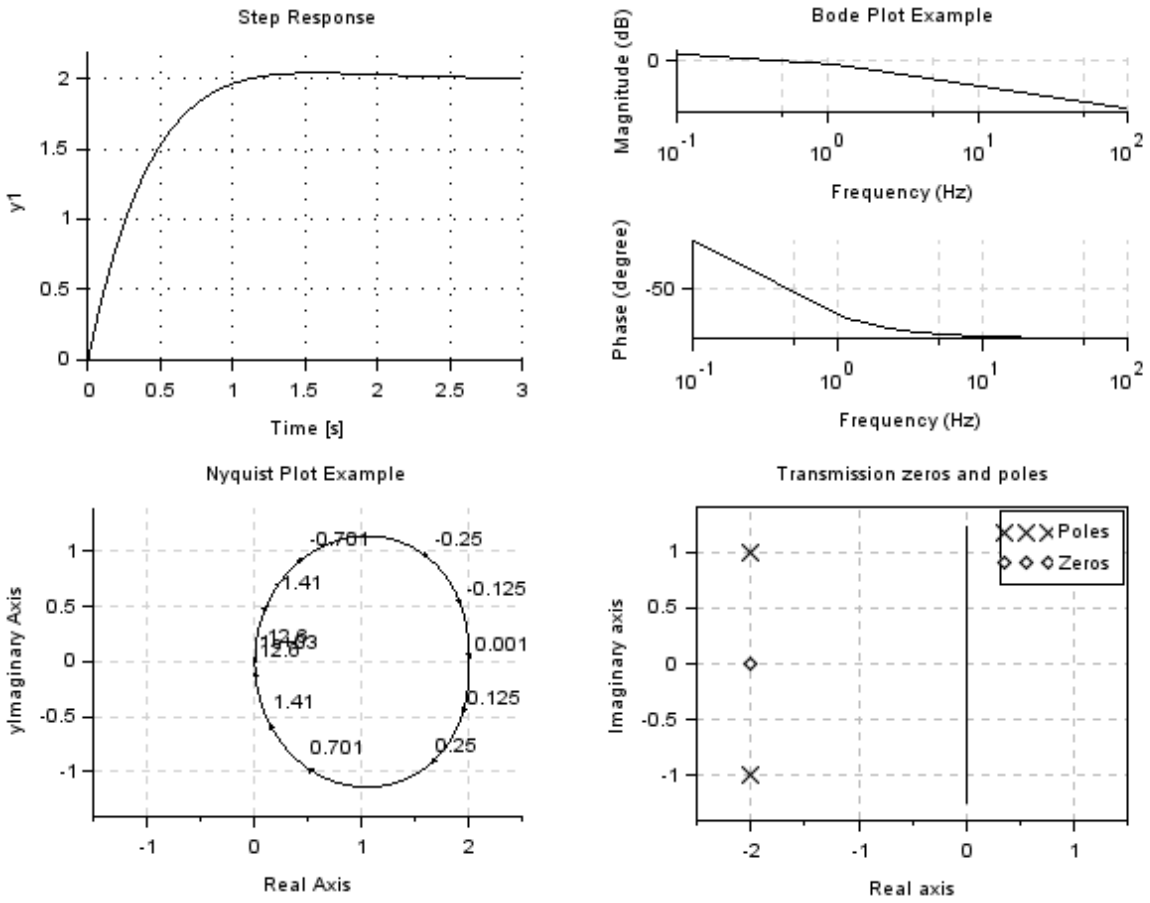
```
-->s=%s;
-->num = poly([0,-1,-2],'s')
 num  =

                  2    3
      2s + 3s + s

-->den=poly([-.5,-1.5,-2.5,-3.5],'s')
 den  =

                                    2     3     4
      6.5625 + 22s + 21.5s + 8s + s
```

```
-->fr=num/den
 fr  =

                                  2   3
                      2s + 3s + s
       ---------------------------------------
                                    2      3   4
      6.5625 + 22s + 21.5s + 8s + s
```

A következő feladat egy adott átviteli függvény kivizsgálását teszi lehetővé, egység ugrás, Nyquist, Bode valamint pólusok és zérusok kimutatásával.

```
s = poly(0,'s');                                        // Define
the complex number frequency parameter.
// Alternatively we can use s = %s.
TF = syslin("c", (5*s + 10) / (s^2 +4*s +5))        // Define
the linear continuous transfer function
disp(TF)

// Converting Transfer Functions to/from State Space
SS = tf2ss(TF)                      // TF -> SS
disp(SS)

// Convert back to transfer functions
TFx = clean(ss2tf(SS))                 // SS -> TF conversion.
Clean removes rounding errors and is recommended.
disp(TFx)

// Step Response
t=0:0.01:3;                          // Define a time range for
the step test
subplot(221)
plot2d(t, csim('step',t,TF));         // csim applies the step
test and plot2d produces the graphical output
xlabel("Time [s]");                   // Add a title and label
axis
```

```
ylabel("y1");
title("Step Response");
xgrid(1, 1, 10);                    // Define a nice grid for
the plot to make it easier to read
subplot(222)
// Bode Plots
//clf();                              // Clears the plotting
window
f = 0.1:100;                          // Set-up the frequency
range we want
bode(TF, f);                        // Generate the Bode plot
title("Bode Plot Example");         // Add a title

// Nyquist Plot
//clf();
                           // Clears the plotting window
subplot(223)
nyquist(TF);                        // Generate the Bode plot
title("Nyquist Plot Example");       // Add a title and label
axis
xlabel("Real Axis");
ylabel("yImaginary Axis");

// Poles zeros
//clf();
subplot(224)
plzr(TF)
```



2.1 ábra Futtatási eredmény

Ellenőrző kérdések:

1. Hogyan definiálunk változókat?
2. Mi a „;” szimbólum szerepe?
3. Milyen ciklusokat használhatunk SCILAB környezetben?
4. Miként történik a létrehozozz adatok mentése szöveges állomány formájában?
5. Hogyan közlünk és fogadunk adatokat a konzolról?

3. A SCILAB GRAFIKUS LEHETŐSÉGEI

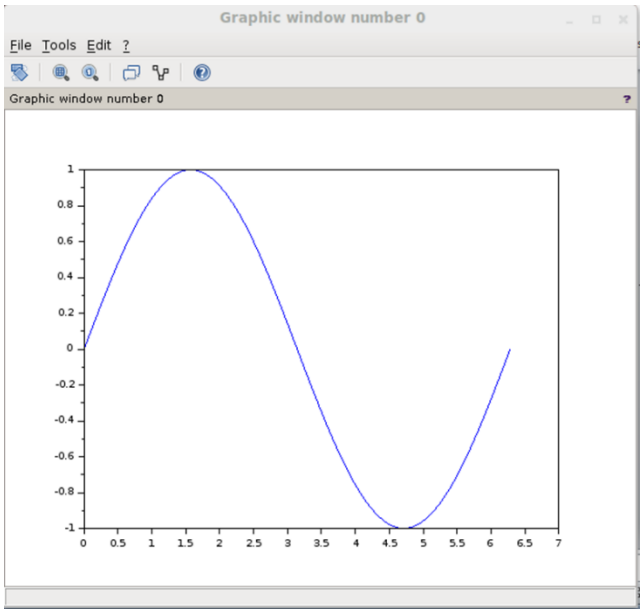
A SCILAB lehetőséget ad 2D és 3D grafikai objektumok létrehozására és megjelenítésére, Ebben a fejezetben csak a kétdimenziós megjelenítéssel foglalkozunk, melyhez a plot(x,y) utasítást kell használni [7]. Az utasítás segítségével az aktuális grafikus ablakba (ha nincs ilyen, akkor a plot utasítás létrehoz egyet) megjeleníthetjük a kívánt függvényt.

2D grafikus megjelenítés

A plot(x,y) Kirajzolja az y vektort az x vektornak megfelelően, vagyis valós pontpárokat ábrázolja az x,y koordináta-rendszerben. Most lássunk néhány példát a plot utasítás használatára!

```
-->x=0:2*%pi/100:2*%pi;
-->plot(x,sin(x))
```

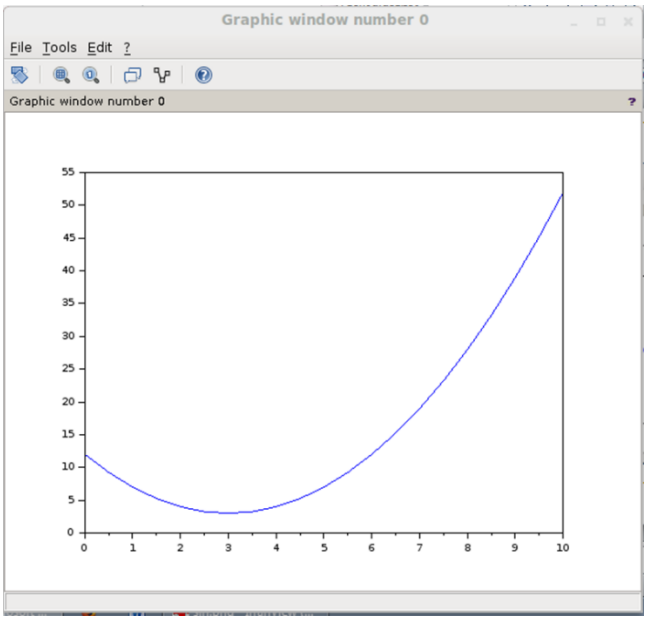
Ekkor a plot(x,y) utasítás hatására az alábbi ábrát kapjuk.



3.1 ábra Futtatási eredmény

A SCILAB használatával bárki egyszerűen kirajzolhat egy olyan grafikont, amely pontjai az összetartozó koordinátákkal van megadva. Ezt a plot utasítással érhetjük el. A plot utasítás számos paramétert tartalmazhat.

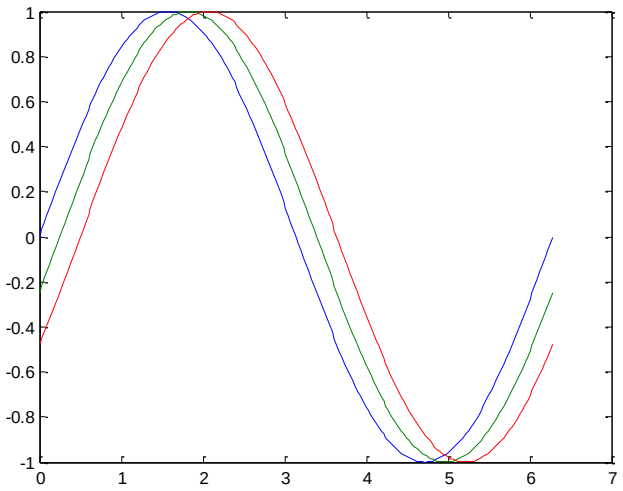
```
-->x=0:0.5:10;
-->plot(x,(x-3).^2+3)
```



3.2 ábra Futtatási eredmény

Lehetőség van több eredményt is kimutatni ugyanazon az ábrán amennyiben igény van rá.

```
-->t = 0:%pi/100:2*%pi;
-->y = sin(t);
-->y2 = sin(t-.25);
-->y3 = sin(t-.5);
-->plot(t,y,t,y2,t,y3)
```

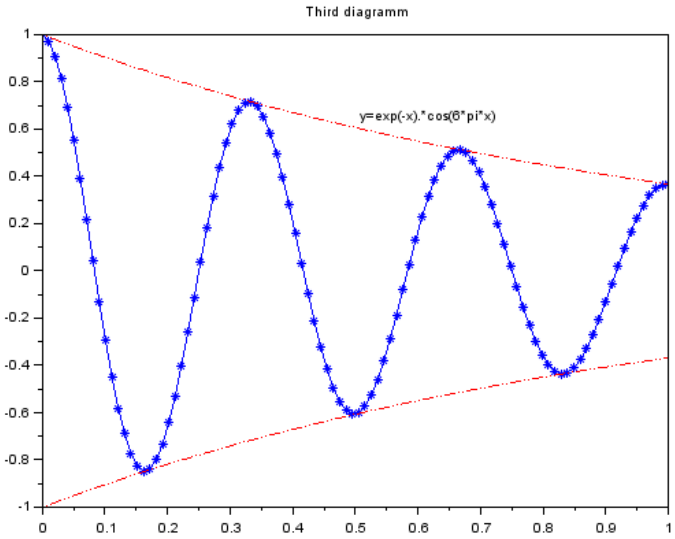


3.3 ábra Futtatási eredmény

Ha nem vagyunk teljesen elégedettek az ábránk megjelentetésével, tudunk változtatni akár a vízszintes, akár a függőleges tengelyek mentén. Az alábbiakban a leggyakrabban használt lehetőségeket foglaljuk össze.

```
-->x=linspace(0,1,100);
-->y=exp(-x).*cos(6*%pi*x); plot(x,y)
-->w=exp(-x);
```

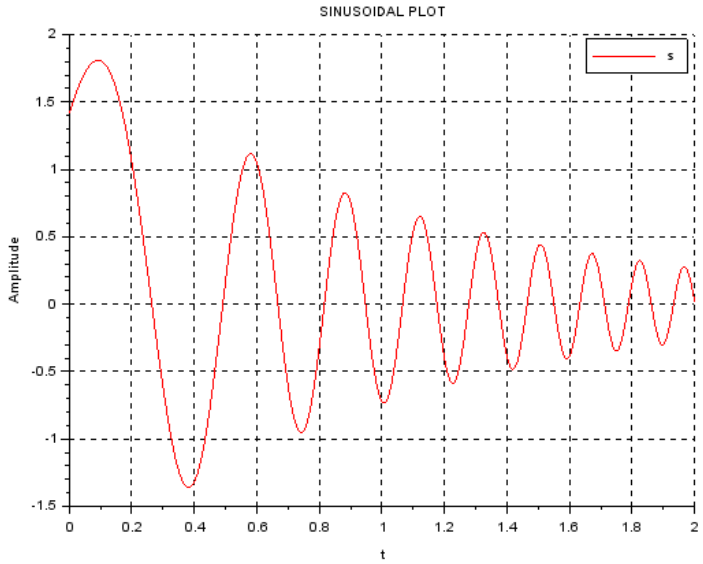
```
-->z=-w;
-->plot(x,y,'b*',x,w,'r:',x,z,'r:'),title('Third diagramm')
-->xstring(0.5,0.6,["y=exp(-x).*cos(6*pi*x)"])
```



3.4 ábra Futtatási eredmény

A vízszintes és a függőleges tengelyekre az xlabel és az ylabel parancsokkal, míg a rajz tetejére az xtitle paranccsal tudunk szöveget kiíratni. Az xsting parancs segítségével pedig a rajz bármelyik, koordinátáival megadott pontjára feliratot, szöveget helyezhetünk. Rajzoljuk ki a megadott függvényt és tegyünk szöveget a tengelyekre, valamint adjunk címet a rajznak!

```
// Plots a sinusoidal function of the type                    /
// s = A(t)(sin(wt+x(t)+phi)), where w = angular              /
// velocity, x(t) = frequency modulation,  phi =              /
// phase shift, and A(t) = amplitude                          /

clear, clc, clf;
f = 1;                                  // Frequency
w = 2*%pi*f;
phi = %pi/4                             // Initial phase shift
fin = (4*%pi)/w;                        // End of plot
t = linspace(0,fin,1000);
A = 2*exp(-t);
s = A.*sin(w*t + 10*t^2 + phi);
plot2d(t,s,5)
xgrid()
xtitle('SINUSOIDAL PLOT')
xlabel('t')
ylabel('Amplitude')
legend('s',1)
```



3.5 ábra Futtatási eredmény

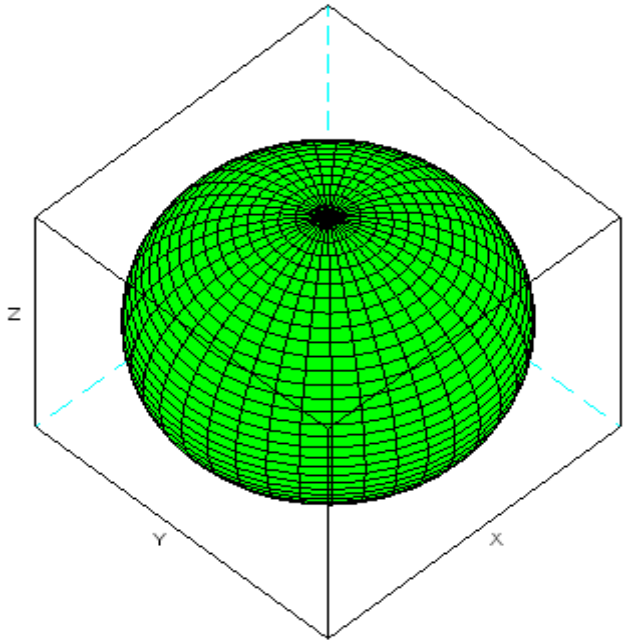
A grid paranccsal egy olyan rácsot illeszthetünk a koordináta-rendszerre, amely illeszkedik a tengelyek beosztására.

3D grafikus megjelenítés

A SCILAB különböző lehetőségeket nyújt háromdimenziós rajzok készítéséhez. Rajzolhatunk háromdimenziós görbéket, felületeket, hálós felületeket, megvilágított felületeket.

A plot3(x,y,z); kirajzolja, és egy vonallal összeköti az x,y,z vektorok által megadott összes (xi,yi,zi) pontot a háromdimenziós koordináta- rendszerben. A vektoroknak azonos elemszámúaknak kell lenniük.

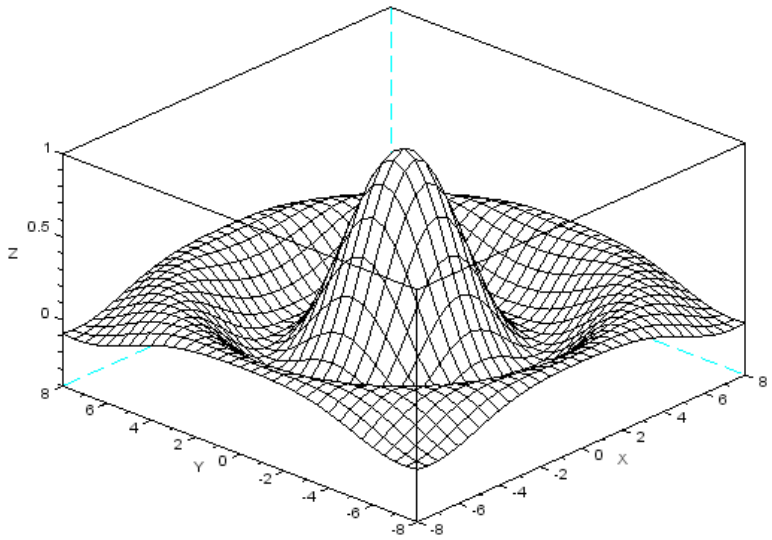
```
// The sphere
clear
clf
u = linspace(-%pi/2, %pi/2, 40);
v = linspace(0, 2*%pi, 40);
x = cos(u)' * cos(v);
y = cos(u)' * sin(v);
z = sin(u)' * ones(v);
[xx,yy,zz] = nf3d(x,y,z);
plot3d(xx, yy, zz, flag=[3 4 3])
```



3.6 ábra Futtatási eredmény

A SCILAB a megadott háromdimenziós adatok alapján egy hálószerű felületet definiál a z-koordináták alapján az x, y vektorok által meghatározott téglalaprács fölött. Egyenes vonallal összeköti a szomszédos pontokat, így olyan eredményt kapunk, mintha egy olyan hálót borítottunk volna a felületre, amelynek a csomópontjai a megadott pontok, és csak a hálót látnánk.

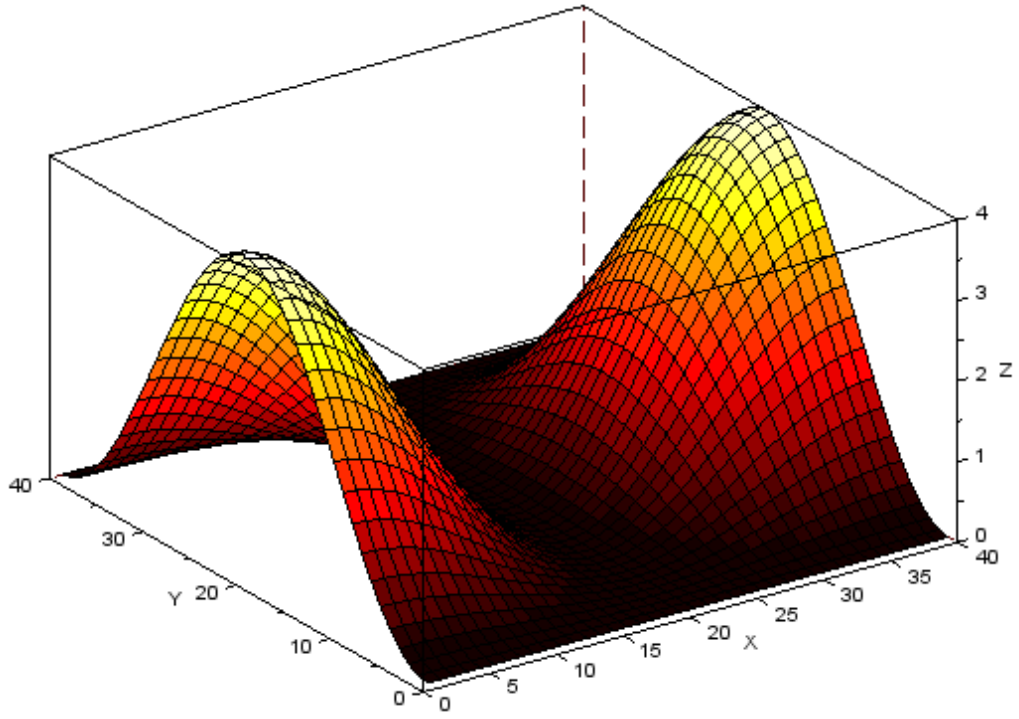
```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + 0.0001;
Z = sin(R)./R;
mesh(X,Y,Z)
```



3.7 ábra Futtatási eredmény

A SCILAB lehetővé teszi a felhasználó számára, hogy a háromdimenziós felületek rajzolásakor maga állítsa be a színeket és a megvilágítást. A *surf* utasítás állítja be a felület rajzolási módját.

```
u = linspace(0, 2*%pi, 40);
z=(1-cos(u))'*(1+cos(u));
surf(z);
gr = gcf();
gr.color_map = hotcolormap(32);
```

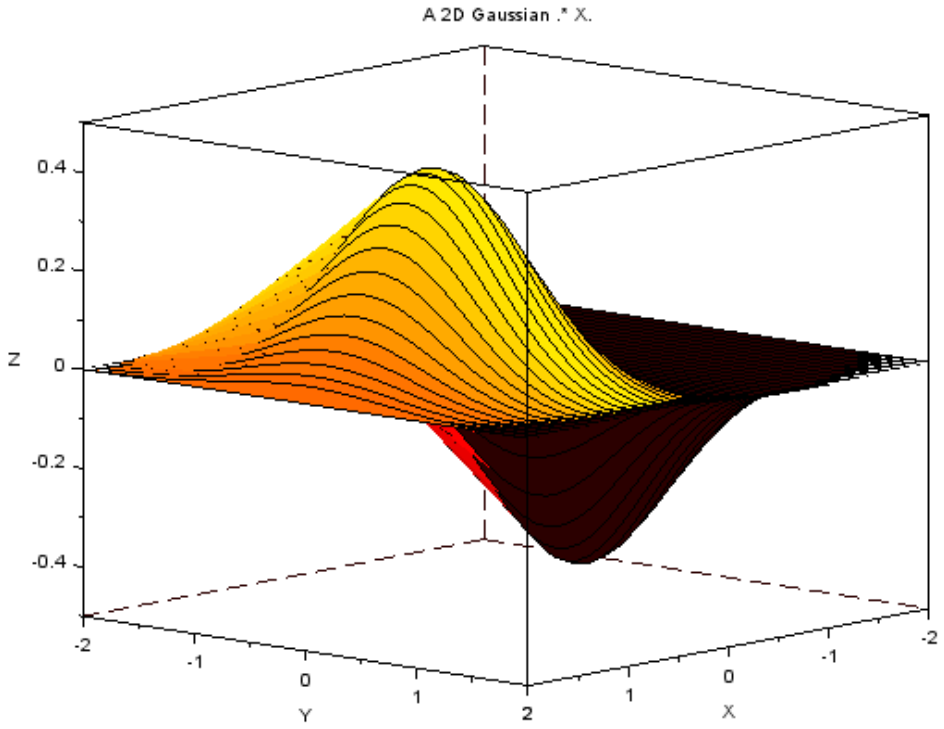


3.9 ábra Futtatási eredmény

Lehet úgy rajzolni, hogy a hálóvonalak látszódjanak, de anélkül is lehet ábrázolni. Ezenkívül a kiszínezés fajtáját is beállíthatjuk. Lehet úgy, hogy az egyes kis felületek egyszínűek legyenek, de lehet úgy is, hogy bizonyos színinterpolációt használunk a csomópontok figyelembe vételével.

```
Xvals = -2 : 0.1 : 2;          // Generate axis vectors.
Yvals = -2 : 0.1 : 2;
[X, Y] = meshgrid(Xvals, Yvals)     // Make a mesh grid.
Z = X .* exp(-(X.^2 + Y.^2));   // A 2D Gaussian .* X.
// Plotting & plot control:
//-------------------------------
drawlater();             // Suppress plotting until ready
   plot3d(X,Y,Z);      // (Suppressed) plot function
  f=gcf();                 // Get Figure handle
```

```
  f.color_map  =  hotcolormap(64);       //  Set  color  table
coppercolormap() hsvcolormap()
  h=gca();                                  // Get Axes handles
  h.rotation_angles=[87,42];   // Set angle of observation
  h.children.color_flag=1;        // Use current color table
   xtitle('A 2D Gaussian .* X.','X','Y','Z');              //
Title & legend
drawnow();                                  // Plot now
//mesh(X, Y, Z);         // Display the plot as a mesh.
//surf(X, Y, Z);         // Display the plot as a surface.
//contour(X, Y, Z);      // Display the plot as a contour.
```



3.9 ábra Futtatási eredmény

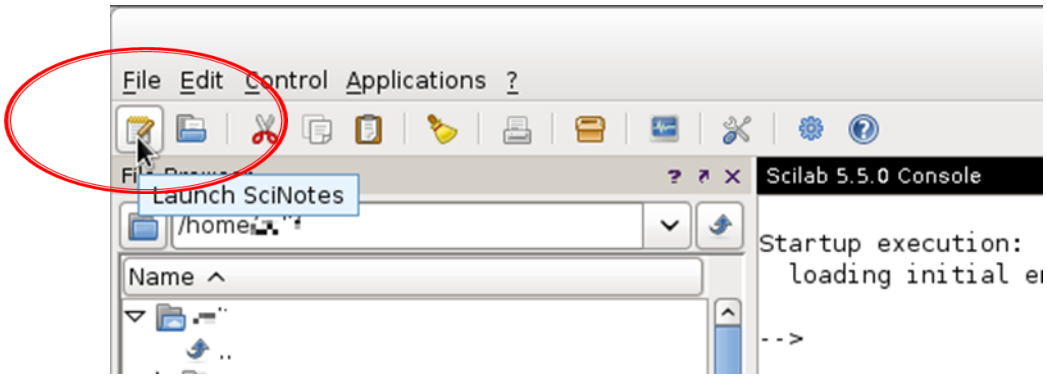
Ellenőrző kérdések:

1. Mire szolgál a plot parancs?
2. Hogyan jelenítünk meg több adatot egy ábrán?
3. Miként jelenítünk meg feliratokat az ábrán?
4. Melyik paranacs teszi lehetővé a 3D ábrázolást?
5. Milyen 3D megjelenítési formákat ismer?

4. A SCILAB PROGRAMOZÁSA

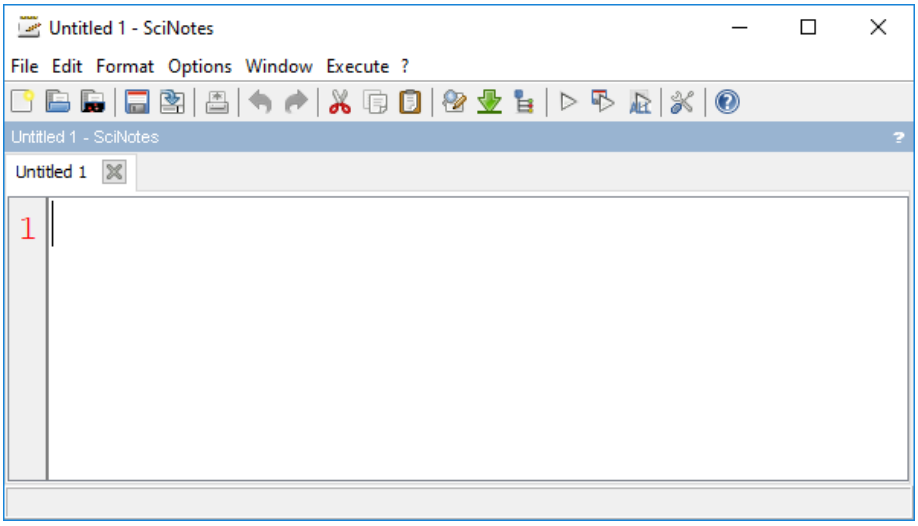
A függvényeket újabb függvényekbe és programokba szervezhetjük. A C-nvelvhez hasonlóan használhatók feltételes utasítások (if, else) és szervezhetők ciklusok (for, while). Léteznek relációs és logikai függvények (find, exist, isnan, isempty, type, strcmp, stb.).

Utasítások <név>.sce fájlba foglalhatók, és az exec <név>.sce parancs hatására végrehajthatók, Kialakíthatók a szokásos módon függvények <név>.sce fájlban a function kulcsszó után, és meghívhatok a <függvény neve> névvel paramétereket átadva. A függvények és szkriptek fejlesztése a Scilab saját szöveges szerkesztője, a SciNotes segítségével célszerű, ahol a hibakeresést és javítást segítő szokásos eszközök rendelkezésre állnak.



4.1 ábra Scinotes indítása

A SciNotes a Scilab beépített szövegszerkesztője. A beírt utasítások lefuttathatók a Scilab-ban. Elmenthető, újra megnyitható (programkód)



4.2 ábra Scinotes ablak

A SCILAB programokat rendszerint *.sce kiterjesztésű fájlokban tároljuk. Ezek a fájlok SCILAB utasításokból állnak, és a fájl nevének utasításként való megadása esetén a SCILAB rendszer a fájl sorait egymás utáni sorrendben végrehajtja.

Programozói feladat (random, sort)

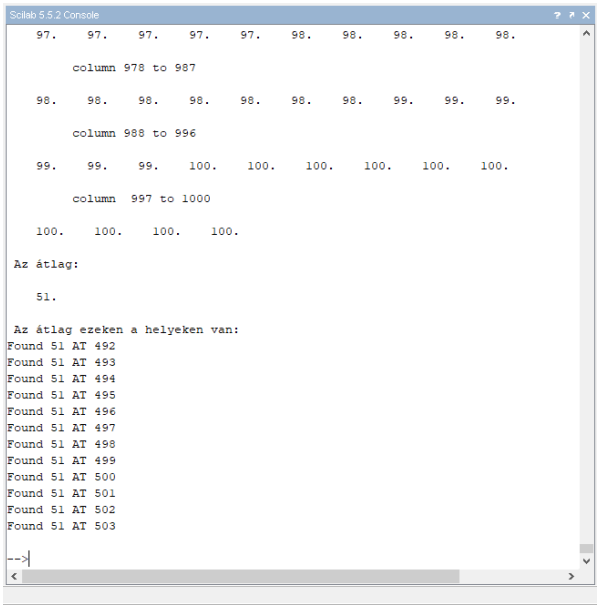
Létrehozni egy 1000 elemből álló vektort, amely véletlenszerűen generált számokat tartalmaz 1 és 100 közötti tartományban! Csak egész számokat használhatunk. A vektor létrehozása után az elemeket nagyság szerint növekvő sorrendbe kell rakni és kiírni a konzolra. Meghatározni a vektor elemeinek átlagát és kimutatni, hogy hol helyezkednek el a vektoron belül az átlag értékkel megegyező számok!

Megoldás:

```
function [a1]=bubble(a,n)
    i = 1;
    j = 1;
    temp = 0;

    for i = 1:n-1
        for j = 1:n-1
            if(a(j)>a(j+1))
                temp = a(j);
                a(j) = a(j+1);
                a(j+1) = temp;
            end
            j = j+1
        end
        i = i+1;
    end

    a1=a;
    disp(a1,"Sorted array is:");
endfunction

function []=search(a,n,ele)
    i = 1;
    j = 0;

    for i=1:n
        if(a(i)==ele)
            printf("Found %d AT %d\n",ele,i);
            j = 1;
        end
    end

    if(j==0)
        disp("%d NOT FOUND",ele);
    end
endfunction
```

```
//Calling Routine:
a = round(rand(1,1000)*100);
disp(a,"Given array");

a1 = bubble(a,1000);

summa = 0;

for i = 1:size(a,2)
        summa = summa+a1(i);
end

atlag = round(summa/1000);
disp(atlag,"Az átlag: ");

ok = 0;
j = 1;
ok2 = 0;
k = 1;

while(ok==1)
    for i=1:1000
        if atlag==a1(i)
            ok = 1;
        else
            ok = ok;
        end
    end

    if ok==0
        while(ok2==1)
            for i=1:1000
                if (atlag-j)==a1(i)
                    ok2 = 1;
                else
                    ok2 = ok2;
                end
            end

            j = j+1;
        end

        while(ok==1)
            for i=1:1000
                if (atlag+k)==a1(i)
                    ok = 1;
                else
                    ok = ok;
                end
            end

            k = k+1;
```

```
        end

        if j>k
            atlag = atlag+k;
        else
            atlag = atlag-j;
        end
    end
    disp(atlag,"Módosított átlag: ");
end

disp("Az átlag ezeken a helyeken van: ")
search(a1,1000,atlag);
```



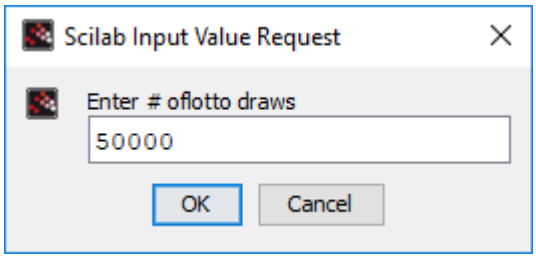
4.3 ábra Futtatási eredmény

Programozói feladat (Lottó számok, statisztika)

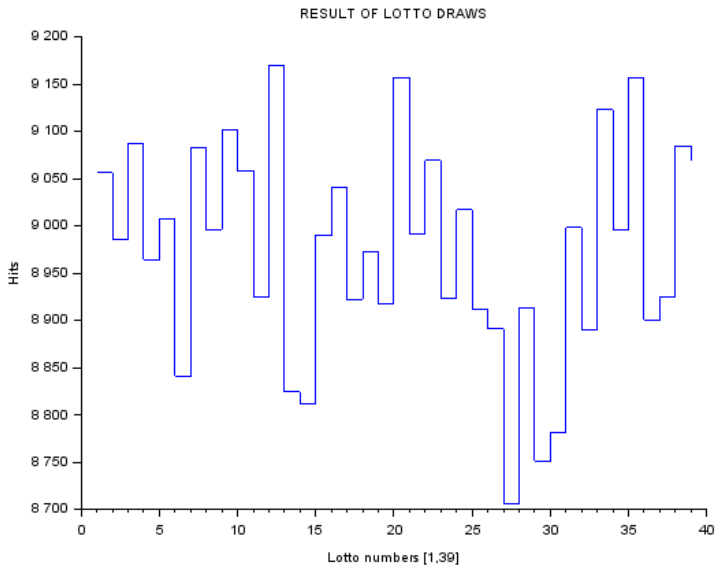
Írjunk egy programot, amely a lottó számok sorsolására alkalmas! A számok nem ismétlődhetnek! Paraméterként megadható hogy hányszor történjen a sorsolás majd grafikusan kimutatható hogy melyik szám milyen gyakran fordult elő.

```
function out=lottodraw(in)
    dt=getdate(); // Pick current date
    rand('seed',1000*dt(9)+dt(10)); // Initialize random generator
    out = floor(1+39*rand(1,in));     // Draw Lotto row (out variable)
    while(length(unique(out))<in)    // If number repeats in row,
        out = floor(1+39*rand(1,in));  // then a new row is drawn
    end
endfunction

// (MAIN) Call subroutine, update histogram, plot:
//-----------------------------------------------------------------
M = evstr(x_dialog('Enter # of...     // Open dialog box
```

```
lotto draws ',''));
N = 7;                                          // Lotto numbers to draw
columns = zeros(1,39);                  // Initiate collecting vector
for k = 1:M
    numbers = lottodraw(N);             // Call to subroutine
    columns(numbers)=columns(numbers)+1;
    // Add 1 for drawn number
end

x = linspace(1,39,39);                      // Define x axis
plot2d2(x,columns,style=2)              // Plot as step functions
xtitle('RESULT OF LOTTO DRAWS')  // Add title & labels
xlabel('Lotto numbers [1,39]')
ylabel('Hits')
```



4.4 ábra Paraméter megadása



4.5 ábra Futtatási eredmény

Programozói feladat (Otto motor körfolyamat)

Scilab segítségével készítsünk el egy szabadon parametrizálható Otto-körfolyamat szimulációját! A szükséges alapadatok megadása után a program számolja ki a négy állapothoz tartozó adatokat.

Alapadatok amelyeket használni lehet:

- kompresszióviszony: ε=5
- nyomás: p1=1, p3=30 [bar]
- hőmérséklet: t1=100 [°C]
- tömeg: m=1 [kg]

- gázállandó: R=0,287 [kJ/kgK]
- adiabatikus kitevő: κ=1,41
- izochor fajhő: cv=0,7002 [kJ/kgK]
- izobar fajhő: cp=0,9873 [kJ/kgK]
- entrópia: s1=1 [kJ/kgK]

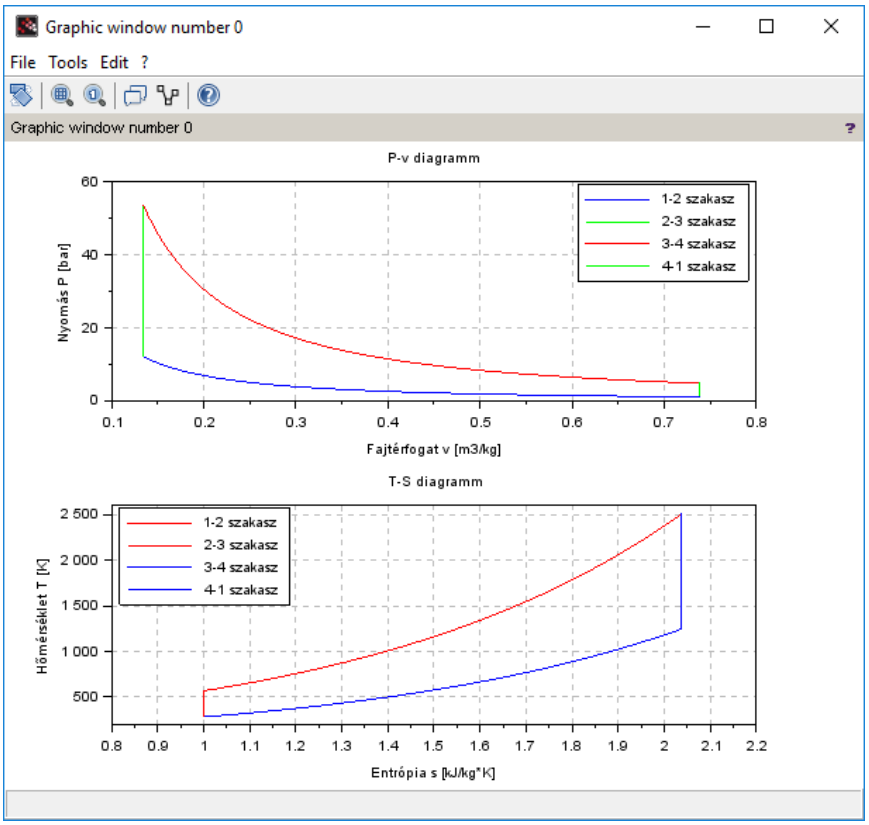
Megoldás:

```
// Hő és áramlástani feladat megoldó program
txt = ['Hőmérséklet:';'Nyomás:';'Kompresszió viszony ℰ:';'Nyomás emelkedési
tényező λ:'];
adatok          =          x_mdialog('Az          Otto          körfolyamat
adatai',txt,['10';'1.1';'5.5';'4.4']);
T(1)=(evstr(adatok(1)))+273,15; //kezdeti hőmérséklet
P(1)=evstr(adatok(2)); //kezdeti nyomás
E=evstr(adatok(3)); // epszilon a kompresszió viszony
L=evstr(adatok(4)); //λ a nyomás emelkedési tényező
K=1.41; //Kappa adiabatikus kitevő a számoláshoz
R0=8.314; //Univerzális gázállandó kJ/kmol*K
Ml=28.96/1000; //levegő moltömege kg/mol
R=R0/Ml; //Gázállandó
v=0;
v1=0;
cv=0.7;
cp=0.987;
s(1)=1;
Q=0;
W=0;
i=1;
s2=0;
t2=0;
t3=0;
nyomasgorbe_1=0;
nyomasgorbe_2=0;
nyomasgorbe_3=0;
nyomasgorbe_4=0
v_beosztas=0;
v_beosztas_2=0;
v_beosztas_3=0;
v_beosztas_4=0;

t_beosztas_1=0;
s_beosztas_1=0;
t_beosztas_3=0;
s_beosztas_3=0;

function [v]=fajterfogat(T,P)
    v=(R*T)/(P*100000);
endfunction

function [P]=nyomas(P,v,v1)
    P=(P*v^K)/(v1^K);
endfunction
```

```
function [v]=kompresszio_viszony(v1)
    v=v1/E;
endfunction

function [T]=homerseklet(p,v)
    T=((p*100000)*v)/R;
endfunction

function [P]=nyomas_emelkedes(p)
    P=L*p;
endfunction

function [u]=belso_energia(T)
    u=cv*T;
endfunction

function [h]=entalpia(T)
    h=cp*T;
endfunction

function [s]=etropia(s2,t2,t3)

    s=s2+cv*(log(t3/t2));
endfunction

//1-es pont számítása
v(1)=fajterfogat(T(1),P(1));
//2-es pont számítása
v(2)=kompresszio_viszony(v(1));
P(2)=nyomas(P(1),v(1),v(2));
T(2)=homerseklet(P(2),v(2));
// 3-as pont Izohor állapotváltozás
v(3)=v(2);
P(3)=nyomas_emelkedes(P(2));
T(3)=homerseklet(P(3),v(3));
//4-es pont adiabatikus expanzió
v(4)=v(1);
P(4)=nyomas(P(3),v(3),v(4));
T(4)=homerseklet(P(4),v(4));
//Fajlagos entropia
s(2)=s(1);
s(3)=s(2)+cv*(log(T(3)/T(2)))
s(4)=s(3)
for i=1:4
    u(i)=belso_energia(T(i));
    h(i)=entalpia(T(i))
    select i>0  //A hőenergia kiszámítása
       case i==1 then q(i)=0;
       case i==2 then q(i)=cv*(T(3)-T(2));
       case i==3 then q(i)=0;
       case i==4 then q(i)=cv*(T(1)-T(4));
    end
```

```
end
for i=1:4
    select i>0  //A munka kiszámítása
       case i==1 then w(i)=(u(2)-u(1))*-1;
       case i==2 then w(i)=0;
       case i==3 then w(i)=(u(4)-u(3))*-1;
       case i==4 then w(i)=0;
     end
    Q=Q+q(i); //Összes hőenergia
    W=W+w(i);  // Összes munka

end

hatasfok=(W/q(2))*100; //A hatásfok
// Wait bar csak a látvány kedvéért
winH=waitbar('Most számolgatok.....');
realtimeinit(0.3);
for j=0:0.1:1,
  realtime(3*j);
  waitbar(j,winH);
end
close(winH);
clf();
for i=1:4
  disp(v(i));
  disp(P(i));
  disp(T(i));
  disp(u(i));
  disp(h(i));
  disp(q(i));
  disp(w(i));
end
  disp(W);
  disp(Q);
  disp(q(2));
  disp(hatasfok);
  // Az 1-es és kettes pontok közé eső pontok kiszámítása
      p_beosztas=linspace(0,P(2),100);
      v_beosztas=linspace(v(1),v(2),100);
  //Nyomásadatok kiszámítása és külön sorvektorba való tárolása, mert
különben rossz adatok keletkeznek
subplot(211);
 set(gca(), 'grid', [1 1]*color('gray')) // rácsháló beállítáa
xlabel(gettext("Fajtérfogat v [m3/kg]")) //Az y tengely cimkézése
ylabel(gettext("Nyomás P [bar]")) //Az x tengely cimkézése
     for i=1:100
     nyomasgorbe_1(i)=nyomas(P(1),v(1),v_beosztas(i));
     end
      // A sorvektor adatainak kiíratása
      plot(v_beosztas,nyomasgorbe_1,'b'),title('P-v diagramm');


// A 2-es és hármas pontok közé eső pontok kiszámítása
```

```
nyomasgorbe_2=linspace(P(2),P(3),100);
v_beosztas_2=linspace(v(2),v(3),100);
plot(v_beosztas_2,nyomasgorbe_2,'g')

//A hármas és negyes pontok közötti görbe kiszámítása
v_beosztas_3=linspace(v(3),v(4),100);
   for i=1:100
     nyomasgorbe_3(i)=nyomas(P(3),v(3),v_beosztas_3(i));
     end
plot(v_beosztas_3,nyomasgorbe_3,'r')
//A 4-es és az egyes pont közötti szakasz rajzolása
nyomasgorbe_4=linspace(P(4),P(1),100);
v_beosztas_4=linspace(v(4),v(1),100);
plot(v_beosztas_4,nyomasgorbe_4,'g')
hl=legend(['1-2 szakasz';'2-3 szakasz';'3-4 szakasz';'4-1 szakasz'],1);

// T-S diagramm rajzolása
subplot(212);
 set(gca(), 'grid', [1 1]*color('gray')) // rácsháló beállítáa
xlabel(gettext("Entrópia s [kJ/kg*K]")) //Az y tengely cimkézése
ylabel(gettext("Hőmérséklet T [K]")) //Az x tengely cimkézése
t_beosztas_1=linspace(T(1),T(2),100);
s_beosztas_1=linspace(s(1),s(2),100);
plot(s_beosztas_1,t_beosztas_1,'r'),title('T-S diagramm');

// A 2 és 3 pont közötti szakasz rajzolása

for i=1:100
    homerseklet_2(i)=homerseklet(nyomasgorbe_2(i),v_beosztas_2(i));
    s_ertekek(i)=etropia(s(2),T(2),homerseklet_2(i));
  end
plot(s_ertekek,homerseklet_2,'r');

// A 3 és 4 pont közötti szakasz rajzolása
t_beosztas_3=linspace(T(3),T(4),100);
s_beosztas_3=linspace(s(3),s(4),100);
plot(s_beosztas_3,t_beosztas_3,'b');

// A 4 és 1 pont közötti szakasz rajzolása

for i=1:100
    homerseklet_3(i)=homerseklet(nyomasgorbe_4(i),v_beosztas_4(i));
    s_ertekek_2(i)=etropia(s(4),T(4),homerseklet_3(i));
  end
plot(s_ertekek_2,homerseklet_3,'b');
hl=legend(['1-2 szakasz';'2-3 szakasz';'3-4 szakasz';'4-1 szakasz'],2);
```



4.6 ábra Futtatási eredmény

Programozói feladat (Ferde hajítás)

Scilab segítségével készítsünk el egy szabadon parametrizálható grafikus felülettel rendelkező ferde hajítás szimuláló programot! A szükséges alapadatok megadása után a program számolja ki és vizualizálja a ferde hajítás pályáját!

Megoldás:

```
function Kosarlabda()

    // megjelenés
    f = figure( ...
    "dockable", "off", ...
    "infobar_visible", "off", ...
    "toolbar_visible", "off", ...
    "menubar_visible", "off", ...
    "visible","off",...
    "default_axes", "on");
    f.figure_size = [800,600];
    f.figure_position = [400,50];
    f.figure_name = "Kosárlabda szimuláció";
    f.icon = "applications-system";

    //  Tengelyek inicializálása
    handles.dummy = 0;
```

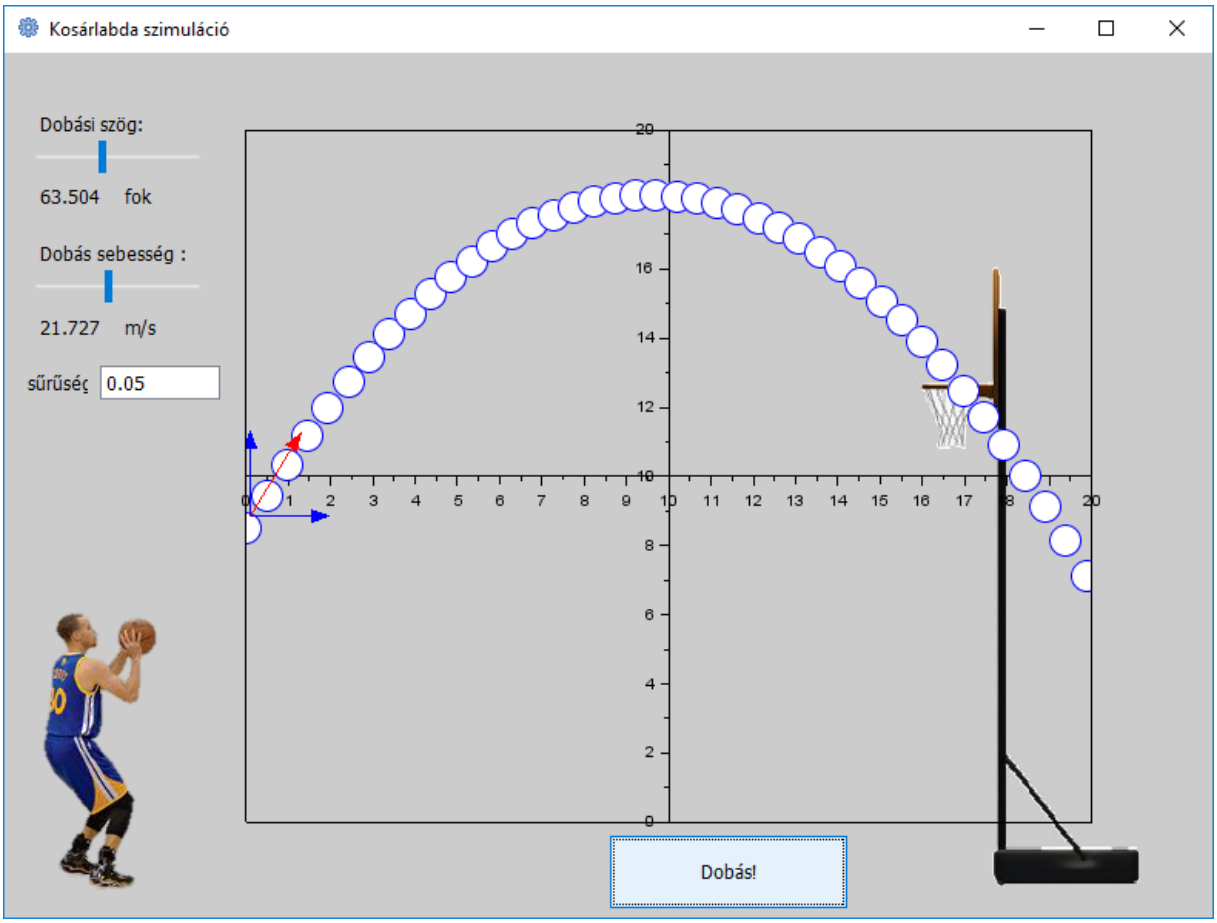
```
    handles.axes2= newaxes();
    handles.axes2.margins = [ 0 0 0 0];
    handles.axes2.axes_bounds = [0.75,0.23,0.2,0.75];
    load('rim');
    Matplot(S,'080');
    handles.axes1= newaxes();
    handles.axes1.margins = [ 0 0 0 0];
    handles.axes1.axes_bounds = [0.2,0.09,0.7,0.8];
    handles.axes1.x_location = 'middle';
    handles.axes1.y_location = 'middle';
    handles.axes1.filled = 'off'
    plot(0,0)
    handles.axes1.data_bounds(1) = 0;
    handles.axes1.data_bounds(2) = 20;
    handles.axes1.data_bounds(3) = 0;
    handles.axes1.data_bounds(4) = 20;
    handles.axes1.auto_scale = 'off';

    // uicontrol

handles.imgPlayer=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.
8,0.8],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[1
2],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.02,0.02,0.12,0.5],'Relief','default','SliderStep',[0.01,0.1],'String
','p1.png','Style','image','Value',[1,1,0,0,0],'VerticalAlignment','middle'
,'Visible','on','Tag','imgLogo','Callback','')
    handles.pbLaunch=uicontrol(f,'unit','normalized','BackgroundColor',[-
1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'
FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Posi
tion',[0.5,0.01,0.2,0.0875],'Relief','default','SliderStep',[0.01,0.1],'Str
ing','Dobás!','Style','pushbutton','Value',[0],'VerticalAlignment','middle'
,'Visible','on','Tag','pbLaunch','Callback','pbLaunch_callback(handles)')

handles.txt1=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0.8
],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'F
ontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.03,0.9,0.15,0.04],'Relief','default','SliderStep',[0.01,0.1],'String
','Dobási
szög:','Style','text','Value',[0],'VerticalAlignment','middle','Visible','o
n','Tag','txt1','Callback','')

handles.sldAngle=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8
,0.8],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12
],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[90],'Min',[45],'Posi
tion',[0.02,0.86,0.15,0.04],'Relief','default','SliderStep',[0.01,0.1],'Str
ing','Angle','Style','slider','Value',[45],'VerticalAlignment','middle','Vi
sible','on','Tag','sldAngle','Callback','sldAngle_callback(handles)')
```

```
handles.txtAngle=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8
,0.8],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12
],'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Posi
tion',[0.03,0.82,0.05,0.03],'Relief','default','SliderStep',[0.01,0.1],'Str
ing','45','Style','text','Value',[0],'VerticalAlignment','middle','Visible'
,'on','Tag','txtAngle','Callback','')

handles.txt3=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0.8
],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'F
ontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.1,0.82,0.06,0.03],'Relief','default','SliderStep',[0.01,0.1],'String
','fok','Style','text','Value',[0],'VerticalAlignment','middle','Visible','
on','Tag','txt3','Callback','')

handles.txt2=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0.8
],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'F
ontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.03,0.75,0.15,0.04],'Relief','default','SliderStep',[0.01,0.1],'Strin
g','Dobás                                                    sebesség
:','Style','text','Value',[0],'VerticalAlignment','middle','Visible','on','
Tag','txt2','Callback','')

handles.sldVel=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0
.8],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],
'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[30],'Min',[15],'Posi
tion',[0.02,0.71,0.15,0.04],'Relief','default','SliderStep',[0.01,0.1],'Str
ing','Velocity','Style','slider','Value',[15],'VerticalAlignment','middle',
'Visible','on','Tag','sldVel','Callback','sldVel_callback(handles)')

handles.txtVel=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0
.8],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],
'FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Posi
tion',[0.03,0.67,0.05,0.03],'Relief','default','SliderStep',[0.01,0.1],'Str
ing','15','Style','text','Value',[0],'VerticalAlignment','middle','Visible'
,'on','Tag','txtVel','Callback','')

handles.txt4=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0.8
],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'F
ontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.1,0.67,0.06,0.03],'Relief','default','SliderStep',[0.01,0.1],'String
','m/s','Style','text','Value',[0],'VerticalAlignment','middle','Visible','
on','Tag','txt4','Callback','')

handles.txt0=uicontrol(f,'unit','normalized','BackgroundColor',[0.8,0.8,0.8
],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'F
ontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','center','ListboxTop',[],'Max',[1],'Min',[0],'Posi
```

```
tion',[0.02,0.6,0.05,0.04],'Relief','default','SliderStep',[0.01,0.1],'Stri
ng','sűrűség
:','Style','text','Value',[0],'VerticalAlignment','middle','Visible','on','
Tag','txtAngle','Callback','')
    handles.editalfa=uicontrol(f,'unit','normalized','BackgroundColor',[-
1,-1,-
1],'Enable','on','FontAngle','normal','FontName','Tahoma','FontSize',[12],'
FontUnits','points','FontWeight','normal','ForegroundColor',[-1,-1,-
1],'HorizontalAlignment','left','ListboxTop',[],'Max',[1],'Min',[0],'Positi
on',[0.08,0.6,0.1,0.04],'Relief','default','SliderStep',[0.01,0.1],'String'
,'0.05','Style','edit','Value',[0],'VerticalAlignment','middle','Visible','
on','Tag','editalfa','Callback','')

    // nyilak kirajzolása
    sca(handles.axes1);
    xarrows([0 0],[8.854  8.854],8,2);
    handles.c1 = gce();
    xarrows([0 0],[8.854  8.854],8,2);
    handles.c2 = gce();
    xarrows([0 0],[8.854  8.854],8,5);
    handles.c3 = gce();

    // figurák láthatóvá tétele
    f.visible = 'on';
    draw_compass(handles);
    handles=resume(handles);

endfunction

// egyéb funkciók

function draw_compass(handles);
    vel = handles.sldVel.value/8;
    th = handles.sldAngle.value;
    xarrows([0.1 2],[8.854  8.854],8,2);
    handles.c1 = gce();
    xarrows([0.1 0.1],[8.854  11.354],8,2);
    handles.c2 = gce();
    xarrows([0.1 0.1+vel*cosd(th)],[8.854  8.854+vel*sind(th)],8,5);
    handles.c3 = gce();
    handles=resume(handles);
endfunction

function update_compass(handles);
    vel = handles.sldVel.value/8;
    th = handles.sldAngle.value;
    handles.c1.data = [0.1 2;8.854  8.854]';
    handles.c2.data = [0.1 0.1;8.854  11.354]';
    handles.c3.data = [0.1 0.1+vel*cosd(th);8.854  8.854+vel*sind(th)]';
endfunction

// Számítások
```

```
function pbLaunch_callback(handles)
    // Belső
    deff('out  =  s(beta)','out  =  sqrt((x(beta)-(L-Dr/2)).^2  +  (y(beta)-
h).^2)')
    deff('out = betalow(beta)','out = s(beta).^2 - (Db/2).^2');
    deff('out = betahigh(beta)','out = x(beta) - L + ((Db-Dr)/2)');
    deff('out = tf(beta)','out = L/(v0*cosd(beta))');
    deff('out = x(beta)','out = v0*cosd(beta)*T(i)');
    deff('out = y(beta)','out = v0*sind(beta)*T(i)+ (1*g*T(i).^2)');

    // konstansok
    H=2,15; // játékos magassága méterben
    h=4-1.25*H; //karok hossza méterben
    L = 12; // palánk távolsága méterben
    Db = 0.24; // labda átmérője méterben
    Dr = 0.45; // gyűrű átmérője méterben
    g = -9.81; // gravitáció méter/sec

    // változtatható értékek
    beta = handles.sldAngle.value;
    v0 = handles.sldVel.value;
    alfa = strtod(handles.editalfa.string);

    // tengelyek újrainicializásála
    if ~isempty(handles.axes1.children) then
        delete(handles.axes1.children);
    end

    // szimuláció
    T = 0:alfa:tf(beta)*10;
    N=length(T);
    plot([20 20],[20 20],'r')

    plot(-ones(1,N),-ones(1,N),'ob','MarkerSize',20);
    l1 = gce();

    sca(handles.axes1);
    draw_compass(handles);

    handles.imgPlayer.string = 'p1.png';sleep(100);
    handles.imgPlayer.string = 'p2.png';sleep(100);
    handles.imgPlayer.string = 'p3.png';sleep(100);
    handles.imgPlayer.string = 'p4.png';sleep(100);

    for i=1:N
        Y=y(beta);
        X=x(beta);
        l1.children.data(i,:) = [X,Y+10-h];
        if Y+10-h < 0
            break
        end
    end
```

```
    handles.imgPlayer.string = 'p1.png';sleep(100);
    handles = resume(handles);
endfunction

function sldAngle_callback(handles)
    handles.txtAngle.string = string(handles.sldAngle.value);
    update_compass(handles);
endfunction


function sldVel_callback(handles)
    handles.txtVel.string = string(handles.sldVel.value);
    update_compass(handles);
endfunction

Kosarlabda();
```



4.7 ábra Futtatási eredmény

Programozói feladat (Fogaskerekes áttétel számítás)

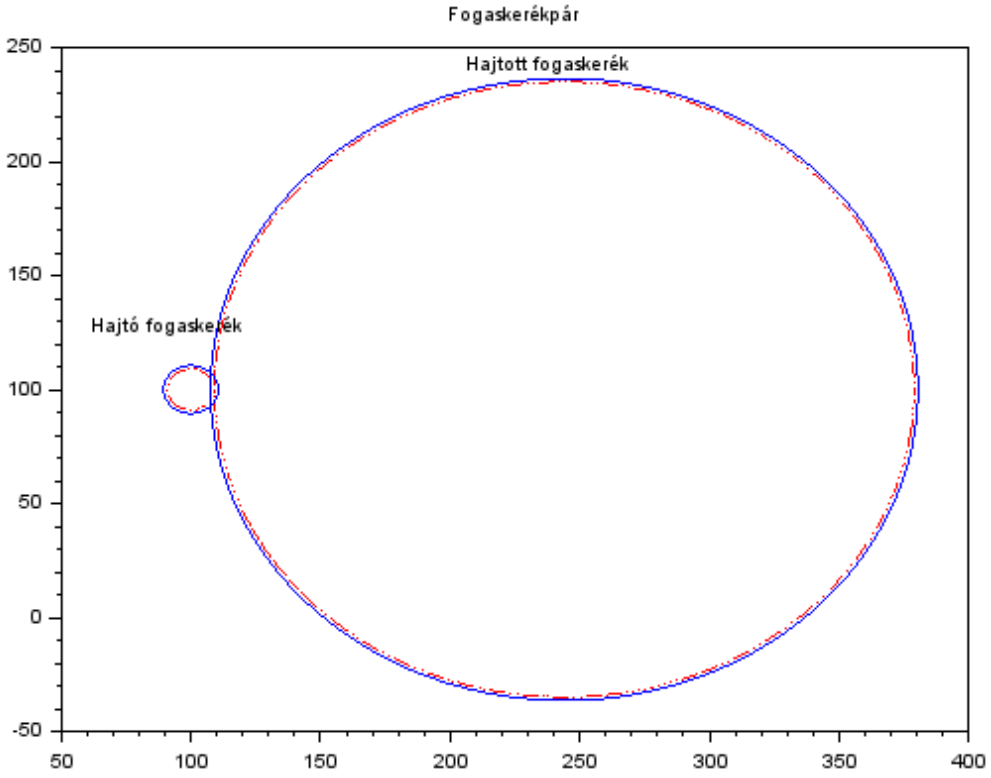
Scilab segítségével készítsünk el egy szabadon parametrizálható fogaskerék áttételének kiszámítására alkalmas szimuláló programot! A szükséges alapadatok megadása után a program számolja ki és vizualizálja a megfelelő fogaskerék párt!

Megoldás:

```
m=input('Kérem írjon be egy modult: ');   //Moldul bekérése.
//Szabványos modulok.
i=[0.05,0.06,0.08,0.1,0.12,0.2,0.25,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,1.25,1.5,
2,2.5,3,4,5,6,8,10,12,16,20,25,32,40,50,60];

while m <> i     //Addíg kéri a modulokat, amíg nem írunk szabványosat.
    disp('Nem szabványos modul.');
    m=input('Írjon be új modult: ');
end

z1=input(' Kérem írja be fogszámot: ');   //Fogaskerék számokkal ugyan ez,
szabványos adatot kér.
j=[12:1:70];
while z1 <> j
    disp('Nem szabványos fogszám.');
    z1=input('Írjon be új fogszámot: ');
end

d1=z1*m;      //Hajtó fogaskerék átmérője[mm]-ben.
disp('a d1-es fogaskerék átmérője[mm]. ',d1);
r1=d1/2;

xc1=100;       //Hajtó fogaskerék középpontjának koordinátái.
yc1=100;
a=linspace(0,2*%pi,100);

x1=xc1+r1*cos(a);           //Hajtó fogaskerék képlete.
y1=yc1+r1*sin(a);

da1=d1+(2*m);              //Hajtó fogaskerék osztóköre.
ra1=da1/2;
x1ra=xc1+ra1*cos(a);
y1ra=yc1+ra1*sin(a);

n1=600/60;          //Forulatszám [1/s]-ben.
n2=40/60;

i=n1/n2;      //Fogaskerék áttétel. Ez egy viszonyszám mértékegység nélkül.
disp('a fogaskerekek áttétele. ',i);

z2=i*z1;           //Adott hajtó fogaskerék fogszámából és az áttételből
kiszámoljuk a hajtott fogaskerék fogszámát.
```

```
d2=z2*m;        //Hajtott fogaskerék átmérője[mm]-ben.
disp('a d2-es fogaskerék átmérője[mm]. ',d2);
r2=d2/2;
//Hajtott fogaskerék középpontjának koordinátái.
xc2=xc1+(r1+r2);        //A körök érintkezése szempontjából szükséges ez a
képlet.
yc2=yc1;

x2=xc2+r2*cos(a);         //Hajtott fogaskerék képlete.
y2=yc2+r2*sin(a);

da2=d2+(2*m);
ra2=da2/2;
x2ra=xc2+ra2*cos(a);
y2ra=yc2+ra2*sin(a);

plot(x1,y1,'r:',x2,y2,'r:');                  //Fogaskerékpár  osztókörének
kirajzolása.

plot(x1ra,y1ra,'b',x2ra,y2ra,'b');               //Fogaskerékpár fejkörének
kirajzolása.
xstring(xc2-40,yc2+r2,["Hajtott fogaskerék"]);     //Felirat.
xstring(xc1-40,yc1+20,["Hajtó fogaskerék"]);       //Felirat.
xtitle('Fogaskerékpár')      //Felirat.

omegahajto=n1*2*%pi;      //Hajtó fogaskerék szögsebessége.

v=(omegahajto*r1)/1000;       //Fogaskerékpár közös sebessége.
disp('a fogaskerekek közös sebessége[m/s]. ',v);
```

```
Kérem írjon be egy modult: 1.5
Kérem írja be fogszámot: 12

    18.
  a d1-es fogaskerék átmérője[mm].

    15.
  a fogaskerekek áttétele.

    270.
  a d2-es fogaskerék átmérője[mm].

    0.5654867
  a fogaskerekek közös sebessége[m/s].
```



4.8 ábra Futtatási eredmény

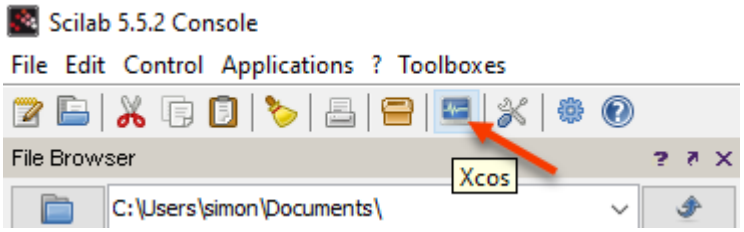
Ellenőrző kérdések:

1. Hogyan kell saját függvényt készíteni SCILAB-ban?
2. Miként tudunk dialog box-on kereszül adatot bekérni a felhasználótól?
3. Hogyan tudunk grafikus felhasználói felületet késziteni a programunkhoz?
4. Hogyan kell több részre osztani a grafikus ábrázoló feluletet?
5. Készítsen interaktív, különböző 3D geometriai alakzatok felülete és térfogata számítására alkalmas programot.

5. AZ XCOS RENDSZER

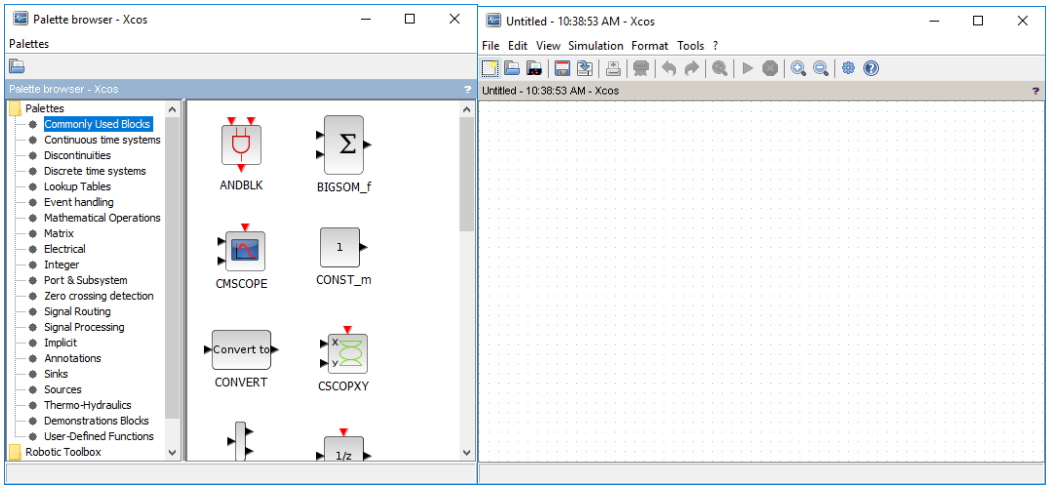
Az XCOS – (Hybrid dynamic systems modeler and simulator), mechanikai rendszerek, hidraulikus körök, szabályzó és irányító rendszerek modellezésére alkalmas eszköz.

Az XCOS a SCILAB része, mely a dinamikus rendszerek modellezésére és vizsgálatára alkalmas. Az XCOS különösen hasznos olyan esetben, ahol a folytonos idejű és a diszkrét idejű komponensek egyaránt megtalálhatóak a modellben. Az XCOS esetében előre elkészített modulokból, blokkokból épül fel a vizsgálni kívánt modell. Az egymáshoz tartozó blokkok könnyen almodellekbe szervezhetünk, melynek köszönhetően egy bonyolult rendszer is áttekinthetővé válik.



5.1 ábra Az XCOS ikon

Nagyon sok előre elkészített modul és blokk áll rendelkezésre az XCOS-on belül. Ezek a blokkok többségében elemi műveletek, melyek szinte minden dinamikus rendszer felépítéséhez szükségesek. Illetve itt is, ahogy SCILAB esetében az "ATOMS portál" segítségével lehetőség van egy-egy speciális területhez (például Robotics Toolbox) tartozó blokkok letöltésére. Ha saját blokkra van szükség, lehetséges a SCILAB saját nyelvén megírni a kívánt függvényt, és ezt, mint blokkot használni a modellben. Természetesen van mód a hiányzó blokk függvényének C alapú nyelven történő megírására is.

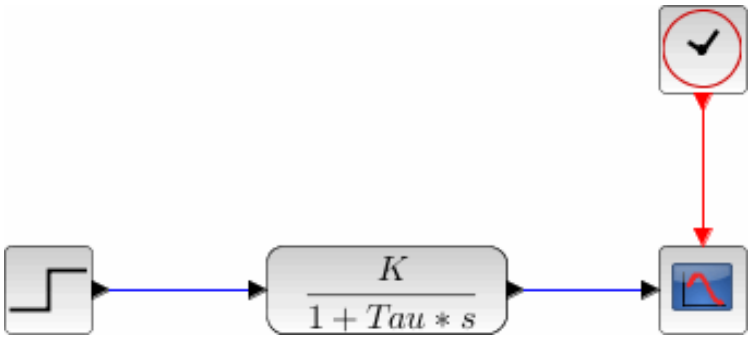


5.2 ábra Az XCOS felület

Fontos megjegyezni, hogy az XCOS több mind egy modellezésre és szimulációra használható környezet, nagyon sok olyan egyéb funkciója van, amely a tervezőt segíti a modell paramétereinek optimalizálásban, a modell validálásában, de lehetőség van C-programot generálni az elkészült modellből.

Első szimuláció

A szimulációhoz szükséges objektumokat a Palette Browser ablakban találhatjuk csoportokra bontva. Első szimulációként egy egyszerű átviteli függvényt fogunk megvizsgálni.

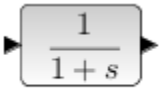


5.3 ábra Az XCOS modell

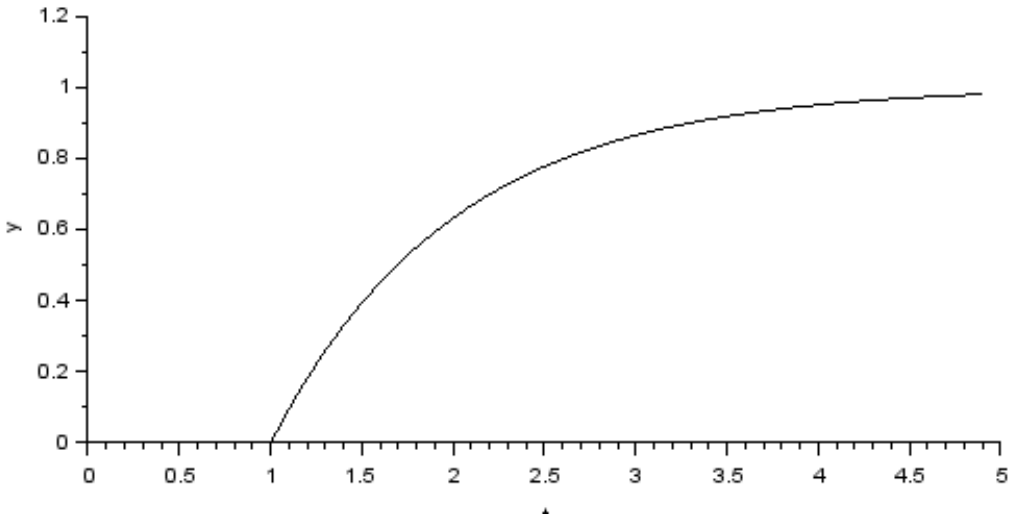
A Simulation / Set context menüpontban megadhatjuk a változók értékeit.

```
K = 1
Tau = 1
```

A szükséges elemek az alábbi helyen találhatók.

Megnevezés	Ikon	Helye
Step function	 STEP_FUNCTION	Palette / Sources
Clock	 CLOCK_c	Palette / Sources
Transfer function	 CLR	Palette / Continuous time systems
Scope	 CSCOPE	Palette / Sinks

Miután megvan a modellünk, megkezdhetjük a szimuláció futtatását a Play gombra kattintva.



5.4 ábra Futtatási eredmény

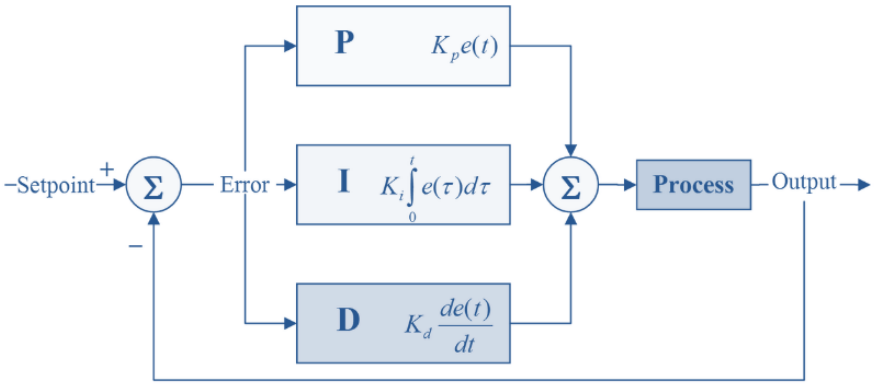
A Scope eredményét egy külön ablakba tekinthetjük meg.

PID szabályozó

A PID szabályozó egy lineáris rendszerek szabályozásánál gyakran alkalmazott, párhuzamos kompenzáción alapuló szabályozótípus. A PID rövidítés a szabályozó elvére utal, a szabályozó által kiadott végrehajtójel:

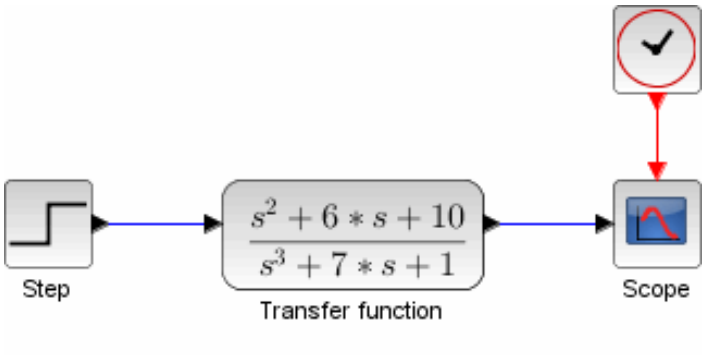
- a hibajellel (P: proportional),
- a hibajel integráljával (I: integral), valamint
- a hibajel változási sebességével, deriváltjával (D: derivative)

Arányos tagokból adódik össze, azaz a végrehajtójel a jelenlegi hiba, a múltbeli hibák és a várható hibák függvénye. Ezen tagok közül nem mindig valósítják meg mindet, ilyenkor beszélhetünk P, PI, PD szabályozókról. A végrehajtójel használható a folyamat vezérlésére, például egy fűtési rendszer energiaforrásának szabályozására.



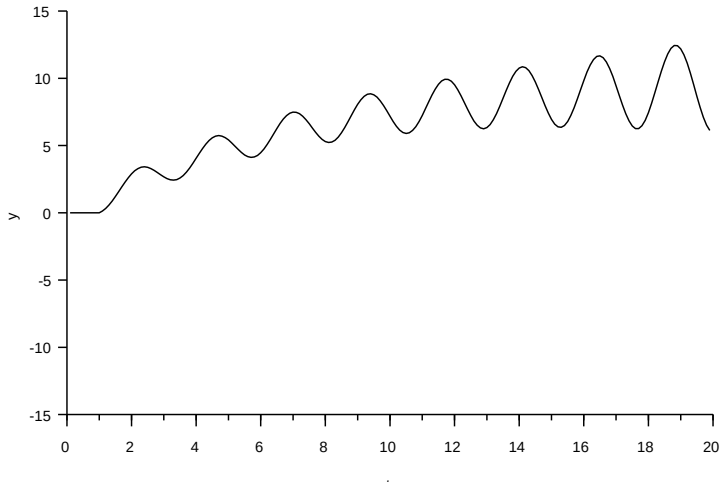
5.5 ábra PID szabályozó

A PID szabályozó működését az alábbi átviteli függvény segítségével mutatjuk be.



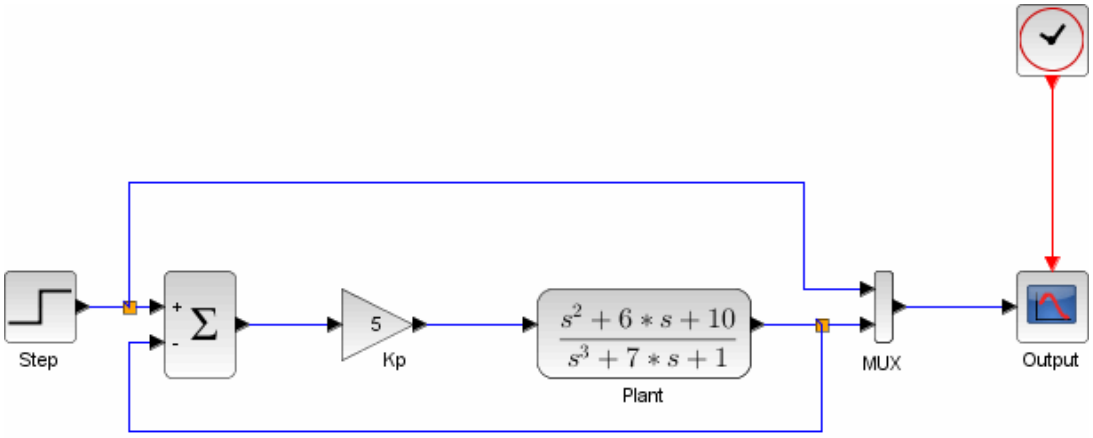
5.6 ábra XCOS modell

Szimuláció futtatása után megfigyelhető az átviteli függvény viselkedése szabályozás nélkül.



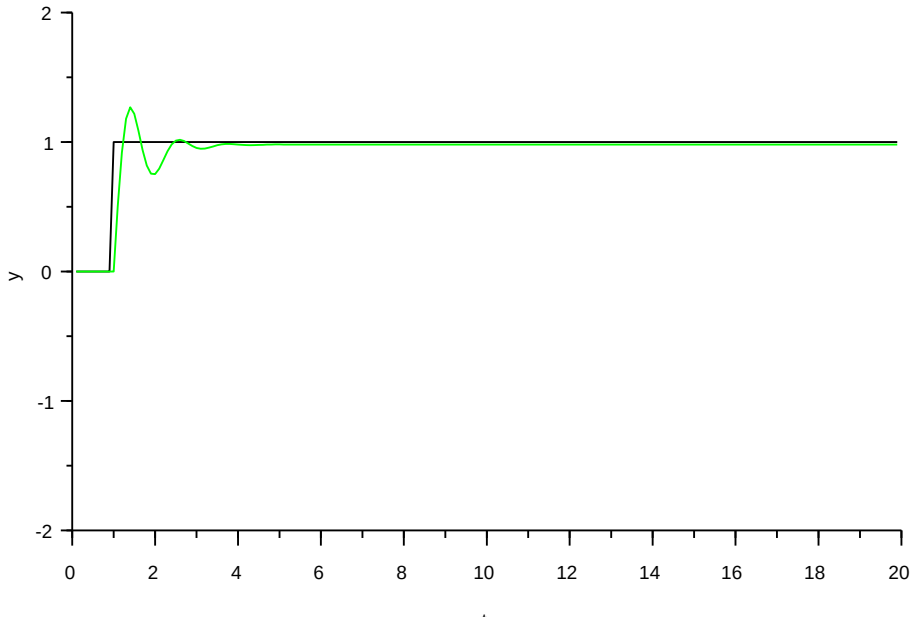
5.7 ábra Futtatási eredmény

Lépésről lépésre fogjuk megszerkeszteni a PID szabályozót tagonként implementálva. Először a proporcionális tag kerül megvalósításra.



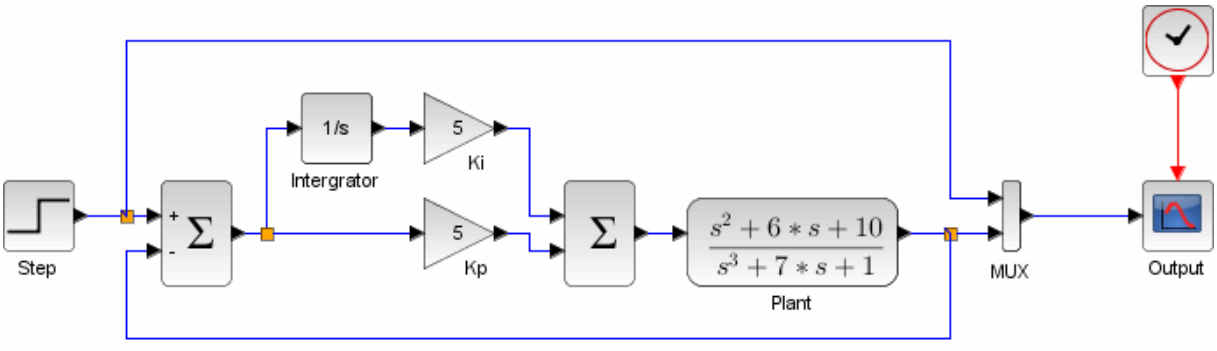
5.8 ábra P tag

Az ábrán látható a kimenő jel abban az esetben, ha P szabályozót alkalmazunk, vagy PID-t (de az integráló és differenciáló tag értéke 0). Az arányos tagnak köszönhetően a felfutási idő csökken.



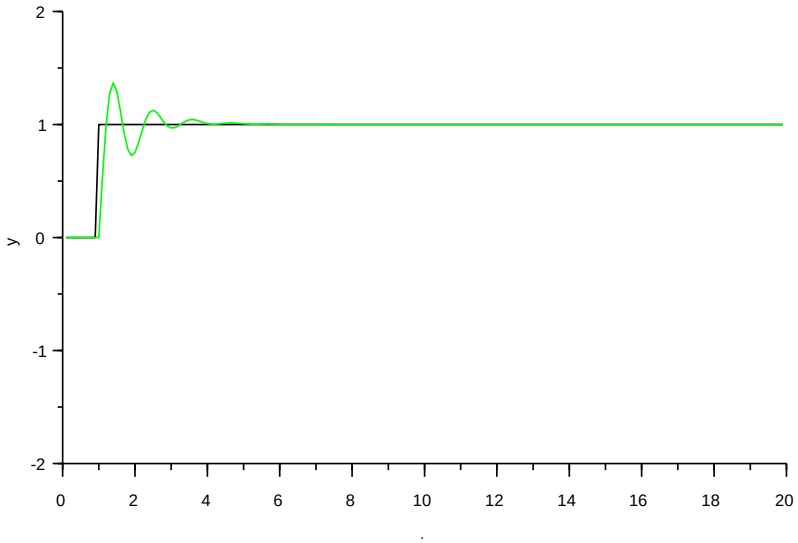
5.9 ábra Futtatási eredmény

Majd a következő lépésben az I tag is megvalósításra kerül. Az integráló tagok állandósult állapotában a kimeneten a bemenőjel integráljával arányos jel jelenik meg. A kimeneten a jel állandóan nő, ha a bemenetre pozitív jel kerül. Az integráló tagot a pontosság jelentős növelésére szokás beiktatni a szabályozási körbe.



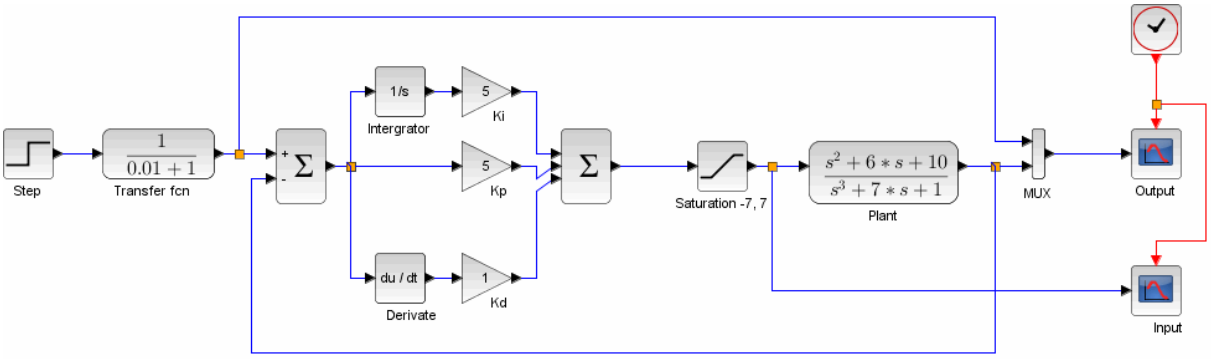
5.10 ábra PI tagok

PI kompenzációt alkalmazunk, ha a rendszer működésének stabilizálására vagy lengési hajlamának csökkentésére van szükség, és ugyanakkor a szabályozási idő növekedése nem hátrányos. Ha állandósult állapotban pontos beállást követelünk meg egységugrás alapjelre, integráló hatást iktatunk be.



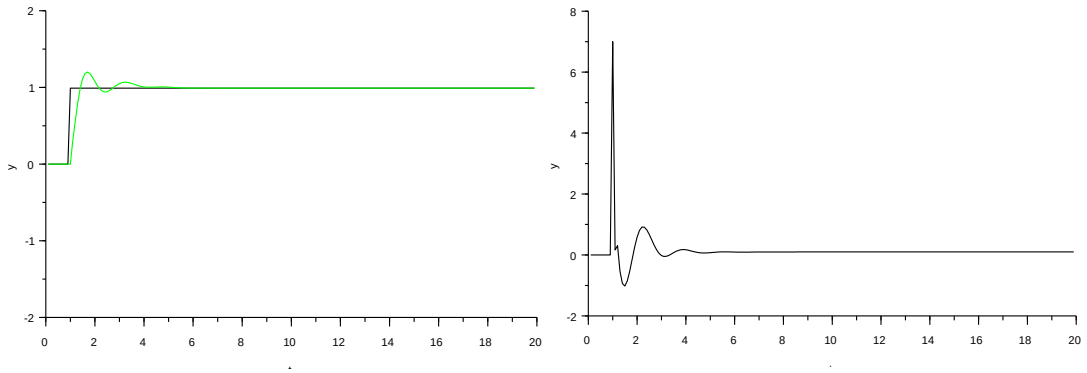
5.11 ábra Futtatási eredmény

A P, I és D tag után a szabályozó hangolása marad.



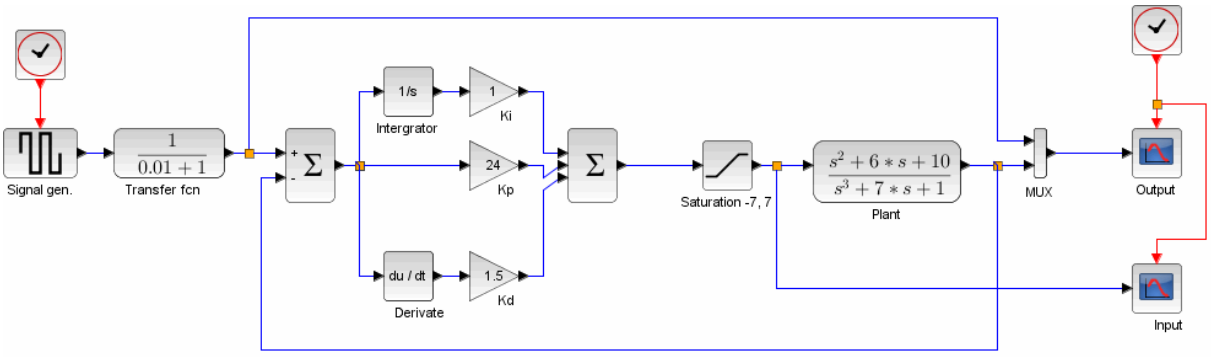
5.12 ábra PID tagok

Az ábrán látható a kimenő jel abban az esetben, ha PID szabályozót alkalmazunk. A PI és a PD szabályozó előnyeit ötvözi. Tehát ha fontos, hogy pontos legyen a szabályozás (integráló tag), ellenben a szabályozási idő megnövekedése nem engedhető meg.



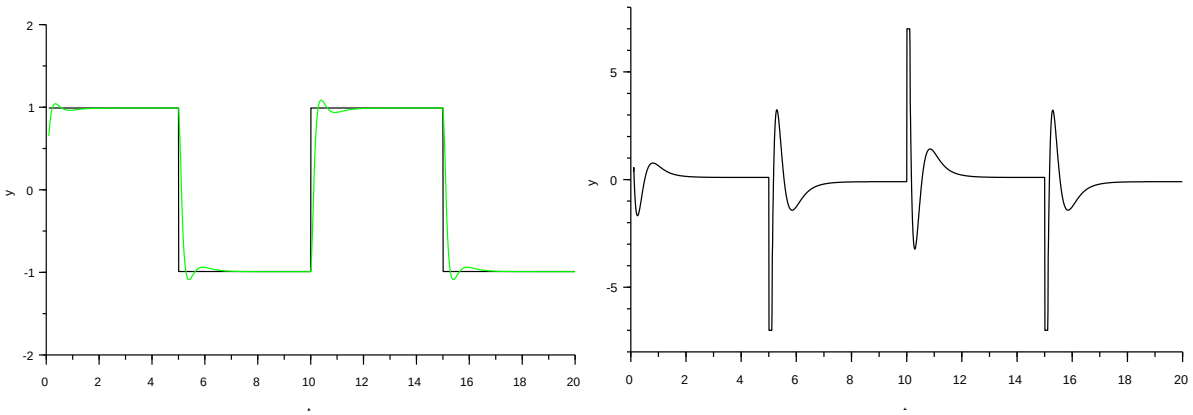
5.13 ábra Futtatási eredmény

Miután behangoltuk a szabályozót, a bemenetre egy jelgenerátort kapcsolunk.



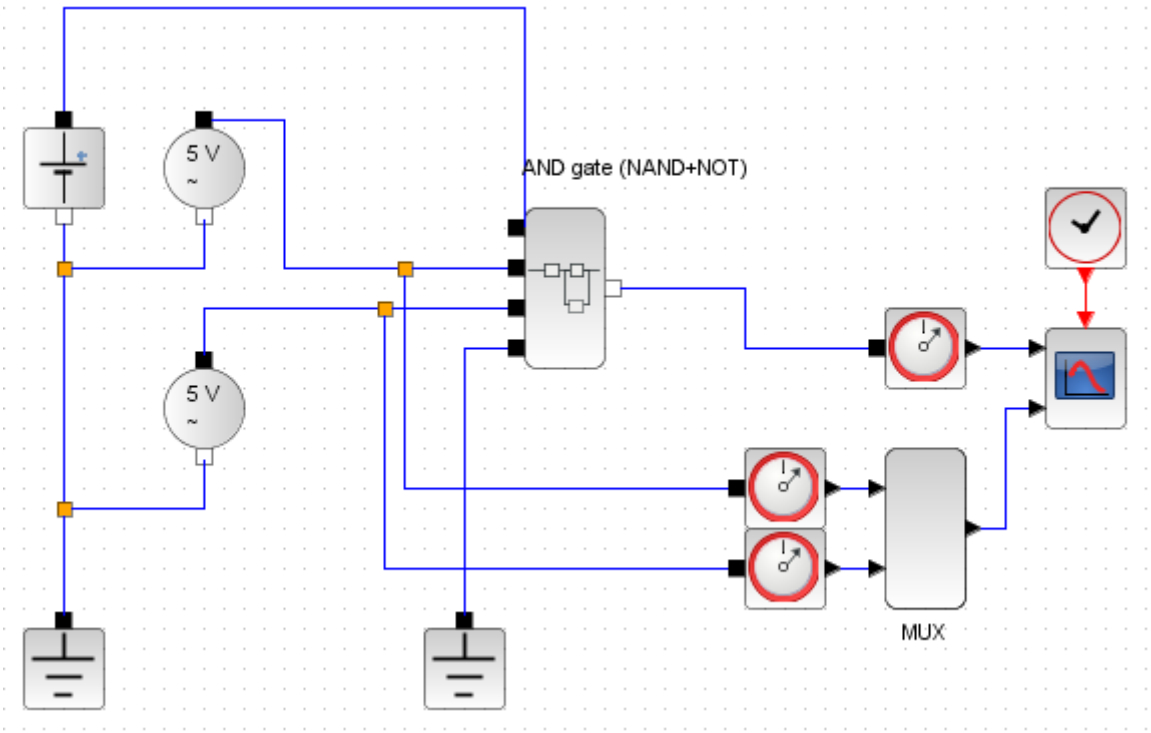
5.14 ábra A PID szabályozó jelgenerátorral

Látható, hogy a jelgenerátor szépen kezelhető. A rugalmas és személyreszabható felépítés fontos cél.



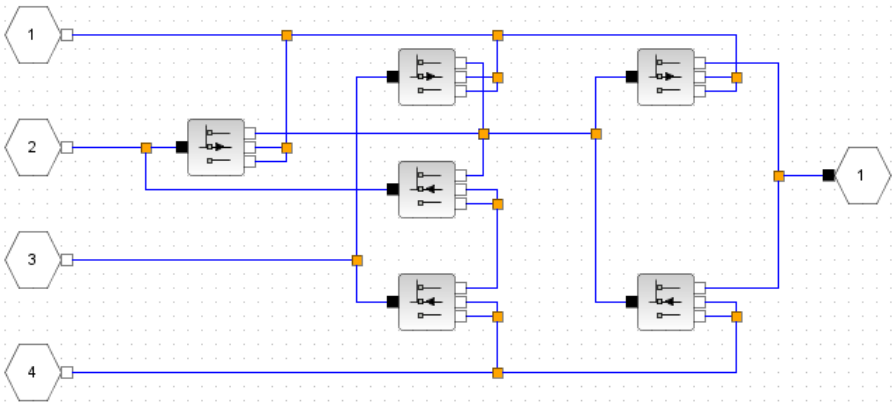
5.15 ábra Futtatási eredmény

A következő feladatban egy ÉS kapu kerül megvalósításra.



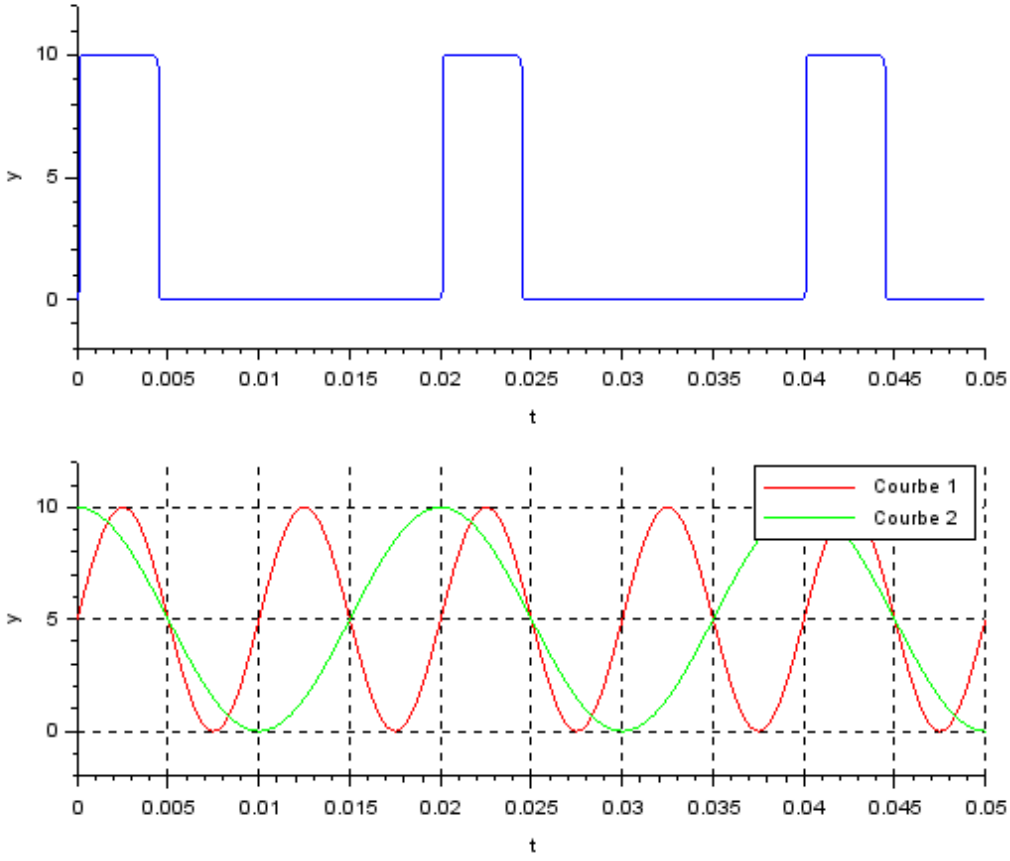
5.16 ábra Logikai ÉS kapu szimulációs környezete

A bemeneten két szinuszjel található, π/2 fázieltolással.



5.17 ábra Logikai ÉS kapu megvalósítása

A kimeneten csak akkor jelenik meg a logikai magas szintű jel, ha mindkét bemeneten pozitív feszültség van.



5.18 ábra Futtatási eredmény

Ellenőrző kérdések:

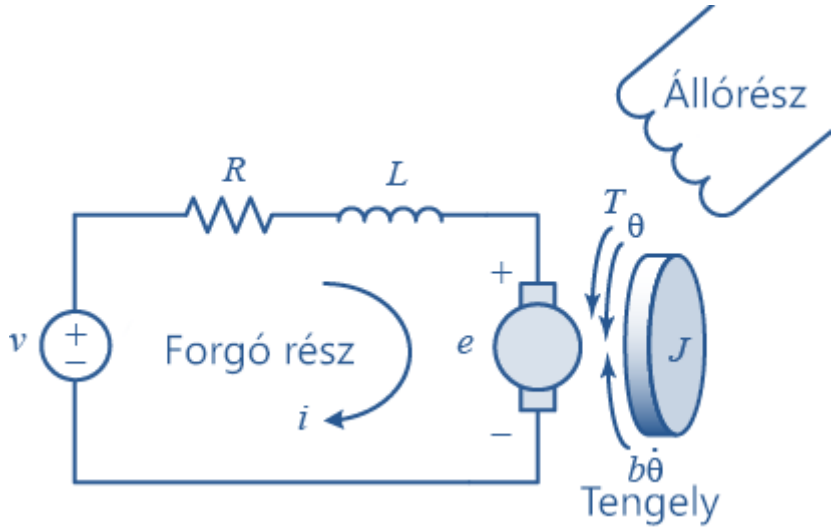
1. Mi az XCOS rendszer?
2. Hogyan készithetö és futtatható egy szimulációs modell?
3. Ismertesse a PID szabályozó működési elvét.
4. Hogyan szerkeszthetünk PD szabályzót alapelemeiből?
5. Milyen hangolási technikákat ismer a PID szabályozóhoz?

6. RENDSZEREK MODELLEZÉSE SCILAB KÖRNYEZETBEN

A feladat célja egy szabályzó megtervezése XCOS-ban, amely egy kefés DC motor pozíció szabályzását végzi. A feladat bemutatja a motor felépítését, működését és modellezését, a szabályzó megtervezését és a kapott eredményeket.

A szénkefés DC motor fizikai modellje

A szénkefés DC motor az egyik legelterjedtebb aktuátor. Két részből áll: az armatúrából, vagyis a rotorból, illetve a sztátorból, vagyis az állórészből.



6.1 ábra Szénkefés DC motor vázlata

Az állórész egy állandó mágneses mezőt generál, amely körülveszi az armatúrát. Az armatúra tekercsei keféken keresztül csatlakoznak a tápfeszültséghez. A tekercsre adott feszültség hatására eletromágneses vonzás, taszítás alakul ki a sztátor és az armatúra között, és ez kényszeríti elfordulásra a motor tengelyét. A kefés DC motor fordulatszáma csak a bemenő tápfeszültségtől függ.

A fenti ábrán levő motor a következő tulajdonságokkal rendelkezik:

- R – a motor ohmikus ellenállása
- L – a motor induktivitása
- V – a tápfeszültség
- i – a motoron átfolyó áram
- T - a motor tengelyének aktuális szöge
- B - a motor súrlódása

Az ábrán nincsenek feltűntetve a következő tulajdonságok:

- Ke – a motor elektromos erő állandója
- Kt – a motor nyomaték állandója
- J – a motor tehetetlensége

A motor modelljének leírása

A motor tengelyének aktuális szöge a motor áramától és a motor nyomaték állandójától függ:

$$T = K_t i$$

Az elektromotoros erő *e* a szögsebesség és a Ke állandó szorzatával egyenlő:

$$e = K_e \dot{\theta}$$

Az SI mértékrendszerben a két állandó értéke egyforma ezért:

$$K_e = K_t = K$$

A fenti ábrából Newton 2. törvénye és Kirchhoff-feszültség törvénye alapján a következő egyenletet tudjuk levezetni:

$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

Átviteli függvény

Laplace-transzformáció alkalmazása után, a fenti egyenletek a következőképpen alakulnak:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V(s) - Ks\Theta(s)$$

Az *I*(s) eliminálása után a fenti egyenletekből a következő nyílt hurkú átviteli függvény írható fel, ahol a forgási sebesség a kimenet, míg a kapocsfeszültség a bemenet.

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2} \quad \left[\frac{rad/sec}{V}\right]$$

Mivel mi a pozíciót keressük, tehát a kimeneten a pozícióra van szükségünk, amelyet úgy érünk el, hogy integráljuk a sebességet, tehát a fenti egyenletet el kell osztanunk s-sel, és a következőt kapjuk:

$$P(s) = \frac{\Theta(s)}{V(s)} = \frac{K}{s((Js + b)(Ls + R) + K^2)} \quad \left[\frac{rad}{V}\right]$$

A fenti egyenlet a motor átviteli egyenlet, amit SCILAB-ban a változók paramétereinek deklarálásával és a függvények definiálásával adunk meg. A programkód a következő ábrán látható:

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2))
P_motor =

0.0274
-----------------------------------------
8.878e-12 s^3 + 1.291e-05 s^2 + 0.0007648 s

Continuous-time transfer function.
```

Állapotegyenletek

Az állapot változók (pozíció, forgási sebesség és elektromos áram) kiválasztásával a fenti egyenleteket állapottéri egyenletekbe írjuk fel. A bemenet a kapocsfeszültség, a kimenet a pozíció.

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & -\frac{b}{J} & \frac{K}{J}\\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{1}{L}\end{bmatrix}V$$

$$y = \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}$$

A rendszert modellezhetjük állapot egyenletekkel is. A következő ábrán bemutatjuk, hogy ezt SCILAB-ban hogyan lehet elkészíteni, és futtatni:

```
A = [0 1 0
     0 -b/J K/J
     0 -K/L -R/L];
B = [0 ; 0 ; 1/L];
C = [1 0 0];
D = [0];

motor_ss = ss(A,B,C,D)
motor_ss =

  a =
                x1           x2           x3
   x1            0            1            0
   x2            0       -1.087         8487
   x3            0        -9964  -1.455e+06

  b =
               u1
   x1           0
   x2           0
   x3  3.636e+05

  c =
       x1  x2  x3
   y1   1   0   0

  d =
       u1
   y1   0

Continuous-time state-space model.
```

A fent bemutatott modell legenerálható a következő paranccsal is:

```
motor_ss = ss(P_motor);
```

Tervezési követelmények

A rendszer kimenetének meg kell felelnie a következő követelményeknek:

- A beállási idő legyen kevesebb, mint 40 ms
- A túllövés legyen kevesebb, mint 16%
- Állandósult állapotban ne legyen hiba, még ha a bemeneten zavar (terhelés) is van.

A rendszer modellezése xcos-ban

A rendszert úgy modellezhetjük, hogy összeadjuk a rotor tehetetlensége ellen ható nyomatékokat és integráljuk a rotor szöggyorsulását a pillanatnyi kerületi sebességgé és ezt a sebességet tovább integrálva megkaphatjuk a motor pozícióját. Kirchoff törvényeit is

alkalmazzuk az armatúra terheléseinek kiszámítására. Először a motor gyorsulását és az armatúra áramváltozását integráljuk:

$$\iint \frac{d^2\theta}{dt^2} dt = \int \frac{d\theta}{dt} dt = \theta$$

$$\int \frac{di}{dt} dt = i$$

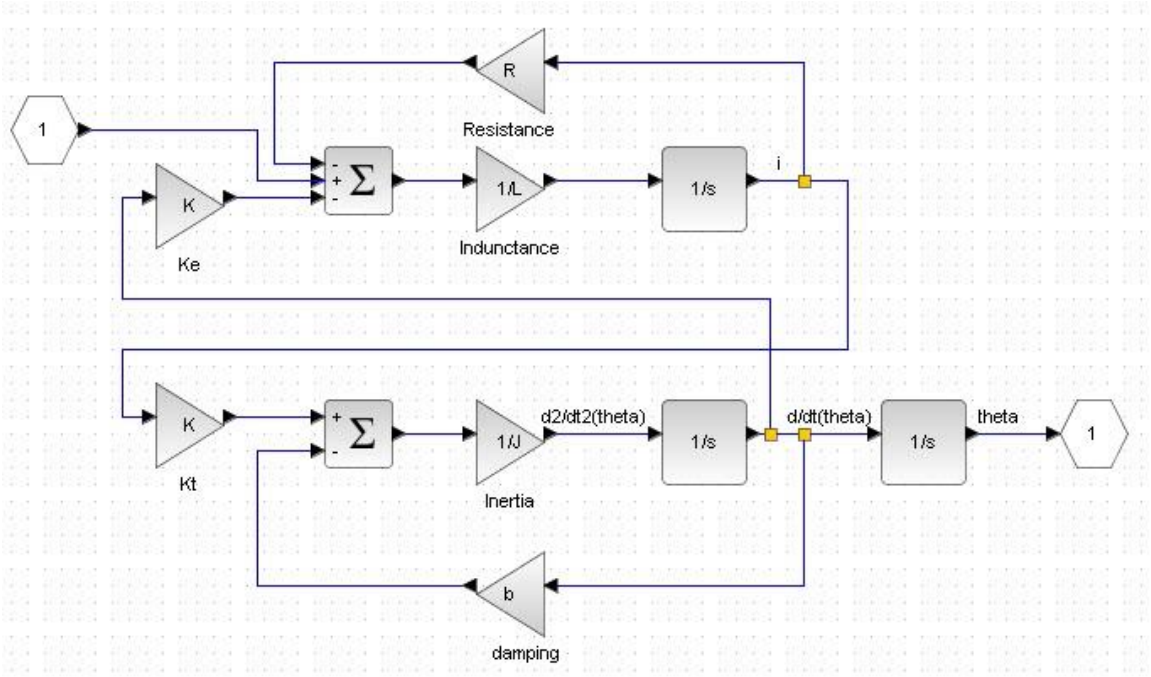
Ezután xcos-ban berakjuk a szükséges integrator blokkokat. Alkalmazzuk Kirhoff törvényeit a motor mehanikai és elektromos rendszerére is. A mehanikai rendszerre a következő képletet írjuk fel:

$$J\frac{d^2\theta}{dt^2} = T - b\frac{d\theta}{dt} \Rightarrow \frac{d^2\theta}{dt^2} = \frac{1}{J}(K_t i - b\frac{d\theta}{dt})$$

Az elektromos rendszerre a következő képletet írjuk fel:

$$L\frac{di}{dt} = -Ri + V - e \Rightarrow \frac{di}{dt} = \frac{1}{L}(-Ri + V - K_b\frac{d\theta}{dt})$$

A szöggyorsulás egyenlő 1/J megszorozva két tényezővel. Hasonlóan az áram deriváltja egyenlő 1/L megszorozva három tényezővel. Ezután hozzáadjuk a nyomatékokat, melyeket Newton törvényeiből kapunk. Először a súrlódást, majd pedig az armatúra nyomatékát. A feszültségre vonatkozó elemeket ezután adjuk hozzá, először a feszültségesést az armatúra ohmikus ellenállásán kereszül, majd pedig az elektromotoros erőt. Ezek után már csak a bemenetet és a kimenetet kell a rendszerhez illeszteni.

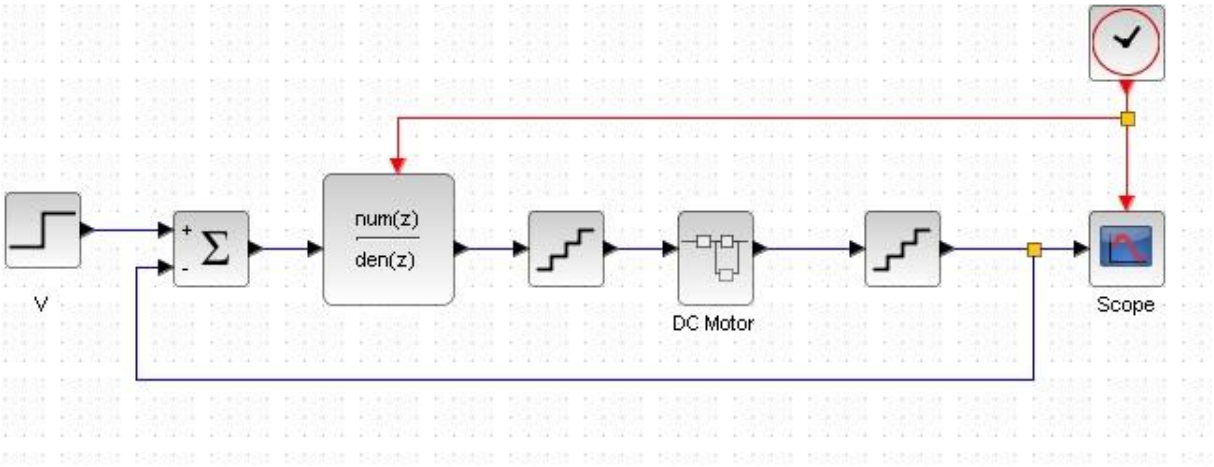


6.1 ábra Bemenet és kimenet hozzáadása Xcos-ban

Az összes elemet először is elmentjük egy alrendszerbe, mégpedig a következő módon:

- jelöljük ki az összes elemet (Ctrl+A),
- majd az Edit menüpontban kattinsunk a Create Subsystem menüpontra.

Így a modell a következőképpen fog kinézni:



6.3 ábra A kész rendszer Xcos-ban

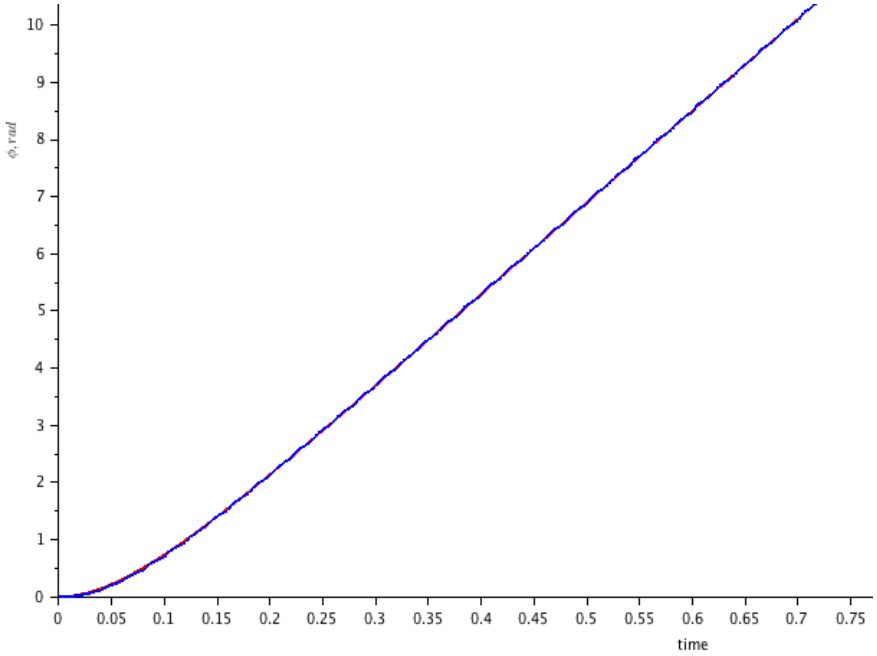
A szimulációnál a következő értékeket használtuk a motor modellezéséhez:

- J=3.2284E-6 kg.m^2
- b=3.5077E-6 Nms
- Ke=0.0274 V/rad/sec
- Kt=0.0274 Nm/Amp
- R=4 ohm
- L= 2.75E-6H

A szimulációt három módon tudjuk futtatni:

- Megnyomjuk a Ctrl+T billentyűkombinációt,
- Simulation menüből a Start menüpontra kattintva, illetve
- a menüsorból a Start Simulation nyilacska megnyomásával.

Miután elindítottuk a szimulációt, a kimenetet a Scope segítségével tudjuk követni, méghozzá úgy, hogy duplán kattintunk a Scope blokkra. Méretezzük át a Scope felületét, hogy a teljes tartalmat láthassuk! Ezt úgy tehetjük meg, hogy az eszközsorban a távcső ikonra kattintunk (Autoscale), vagy jobb klikk után az almenüből kiválasztjuk az Autoscale menüpontot.



6.4 ábra Eredmény megtekintése a Scope-on

Rendszer analízis

Az egyenáramú motornál a dinamikus egyenletek Laplace tartományban és a nyílt hurkú átviteli függvény a következők:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V(s) - Ks\Theta(s)$$

$$P(s) = \frac{\Theta(s)}{V(s)} = \frac{K}{s((Js + b)(Ls + R) + K^2)} \quad \left[\frac{rad}{V}\right]$$

Ahhoz, hogy egy lépésben 1 rad/sec legyen a referens érték, a tervezési kritériumok a következők:

- A beállási idő kevesebb, mint 40ms
- A túllövés kevesebb, mint 16%
- Állandósult állapotban ne legyen hiba, még ha a bemeneten zavar (terhelés) is van.

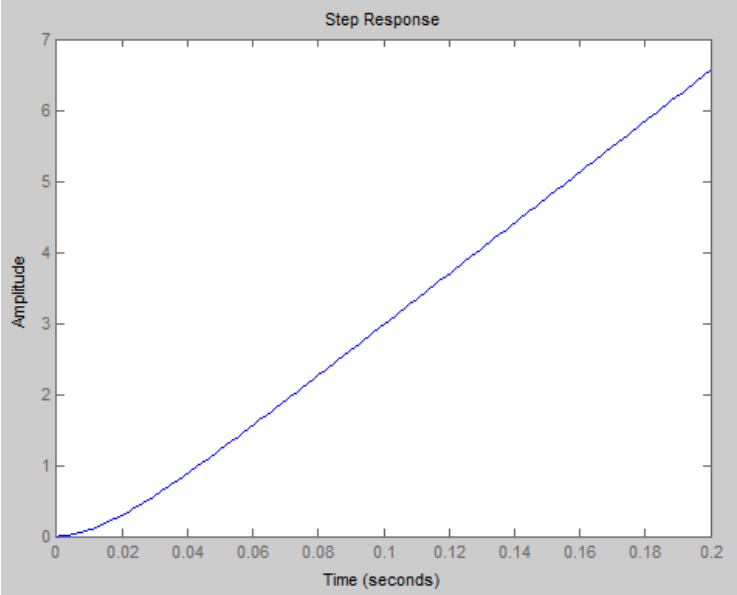
Nyílt hurkú válasz

Először is hozzunk létre SCILAB-ban egy új sce-fájlt, és írjuk be a következőket:

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2));
```

Most nézzük meg, hogy a rendszer hogy viselkedik, ha bekapcsoljuk! Megadjuk a t időkorlát függvényét, majd a motor modelljével lefuttatjuk a step függvényt.

```
t = 0:0.001:0.2;
step(P_motor,t)
```



6.5 ábra A rendszer viselkedése

A fenti képen látható, hogy a bemenetre adott 1V hatására a motor határtalanul gyorsul. Ez megfelel az adott követelményeknek, amikor a motoron nincs terhelés, viszont a rendszer ilyenkor instabil. A rendszer stabilitását az isstable paranccsal lehet ellenőrizni, aminek a válasza 1 ha a rendszer stabil, 0, ha nem stabil a rendszer.

```
isstable(P_motor)
ans =
     0
```

A rendszer stabilitása szintén meghatározható a rendszer pólusaiból, amiket a pole paranccsal kapunk.

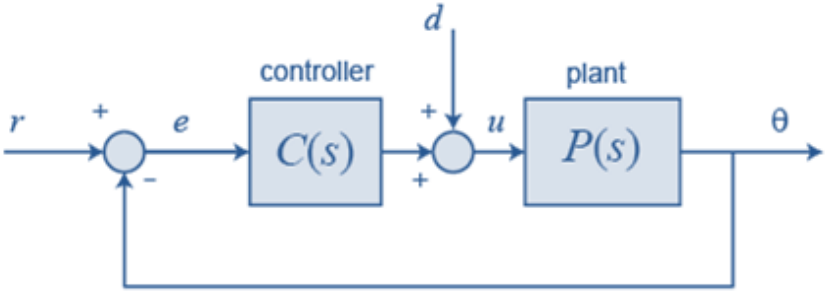
```
pole(P_motor)
ans =
   1.0e+06 *
         0
   -1.4545
   -0.0001
```

Amint látható, az egyik pólus imaginárius, a másik kettő a komplex sík bal oldalán helyezkedik el. Az imaginárius pólus azt mutatja, hogy rendszer válasza nem fog határtalanul növekedni, viszont nem is fog a nulla felé tartani. Attól még, hogy a válasz nem fog határtalanul nőni, egy rendszer egy pólussal az imaginárius tengelyen tarthat a végtelenbe attól függetlenül, hogy a bemenet korlátos-e. Az előző grafikonon is ezt láthattuk. Ebben az esetben a középpontban lévő pólus(0) úgy hat, mint egy integrátor, vagyis ha a rendszer bemenetére jelet adunk, a kimenete elkezd nőni a végtelenségig, mint amikor egy konstans értéket integrálunk.

PID szabályzó tervezése

$$P(s) = \frac{\Theta(s)}{V(s)} = \frac{K}{s((Js+b)(Ls+R)+K^2)} \quad \left[\frac{rad}{V}\right]$$

A rendszer szerkezeti formája az alábbi ábrán látható:



6.6 ábra A rendszer szerkezeti formája, PID szabályzó tervezése

Ahhoz, hogy egy lépésben 1 rad/sec legyen a referencia, a tervezési kritériumok a következők:

- A beállási idő kevesebb, mint 40ms
- A túllövés kevesebb, mint 16%
- Állandósult állapotban ne legyen hiba, még ha a bemeneten zavar (terhelés) is van.

Nyitunk egy új m-fájlt, és a következő parancsokat adjuk meg:

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2));
```

Átviteli függvény a PID vezérlőhöz:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

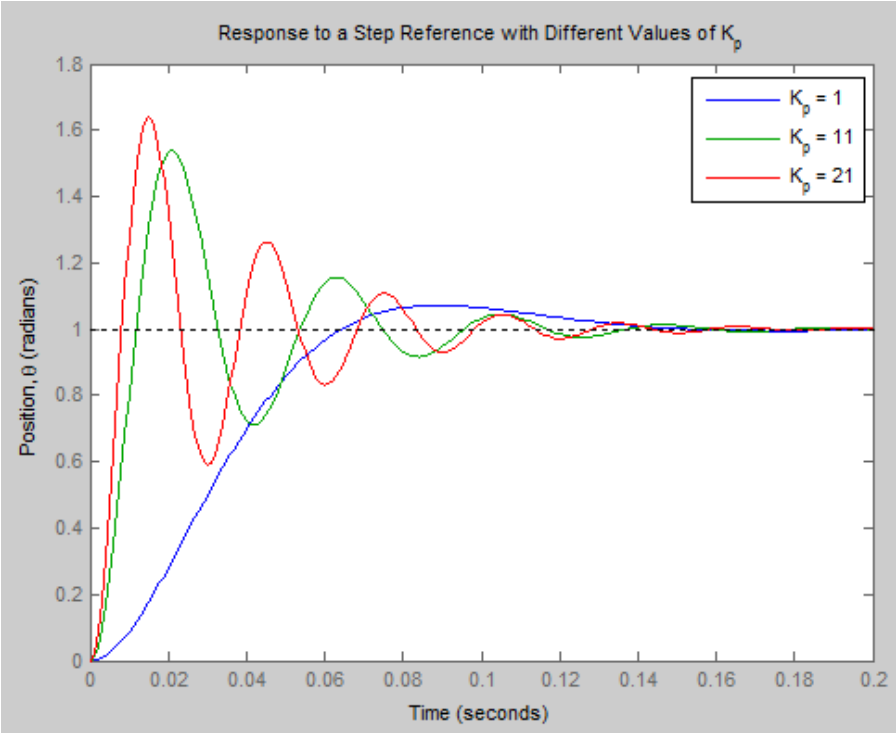
Arányos szabályozás

Ahhoz, hogy meghatározzuk a zárt hurkú szabályzó átviteli függvényét, használjuk a *feedback* parancsot!

```
Kp = 1;
for i = 1:3
    C(:,:,i) = pid(Kp);
    Kp = Kp + 10;
end
sys_cl = feedback(C*P_motor,1);
```

Most vizsgáljuk meg a zárt hurkú szabályzó átviteli függvényének ugrásválaszát:

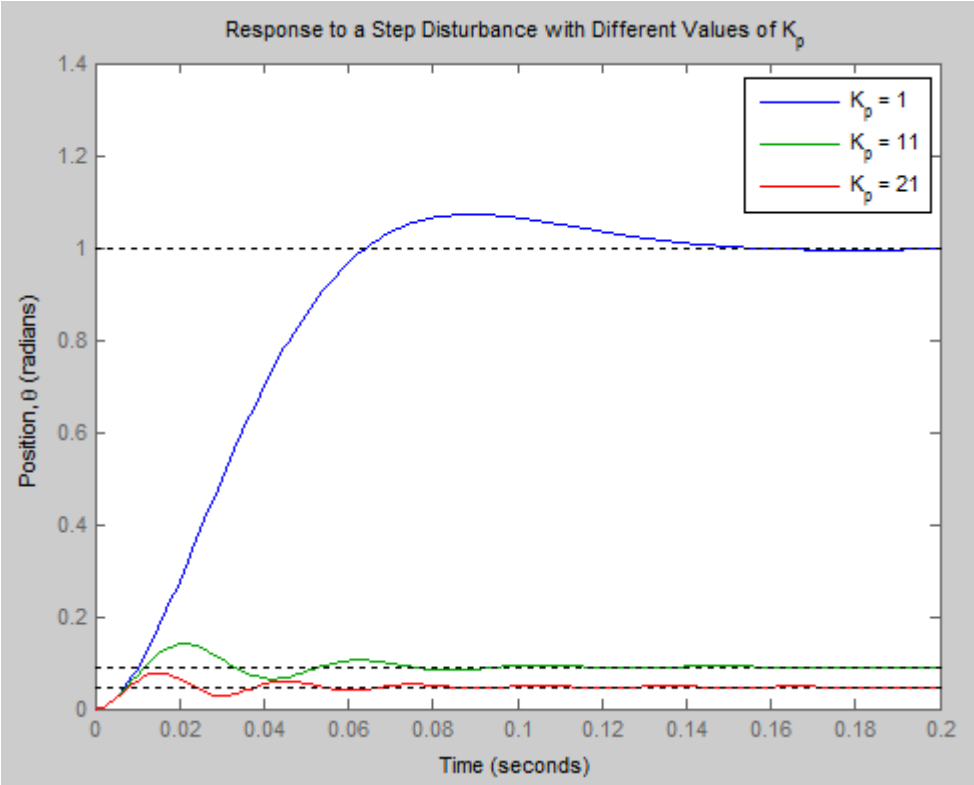
```
t = 0:0.001:0.2;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with Different Values of K_p')
legend('K_p = 1',  'K_p = 11',  'K_p = 21')
```



6.6 ábra A zárt hurkú szabályzó átviteli függvényének lépés válasza

Most vizsgéljuk meg a rendszer viselkedését terhelés alatt! Ebben az esetben a zero referenciát, és megfigyeljük, hogyan válaszol a rendszer magára a terhelésre. A feedback parancs kell, hogy zárt hurkú átvitelt hajtsunk végre, ahol negatív visszacsatolás van, habár most csak az átviteli P(s) függvény van a kimenet felé és a vezérlő C(s) pedig visszacsatolásban. Most nézzük meg újra a fenti blokk ábrát az oldal tetején, hogy lássuk a rendszer felépítését! Adjuk hozzá az m-fileunk végére a következőket, és futtassuk le:

```
dist_cl = feedback(P_motor,C);
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with Different Values of K_p')
legend('K_p = 1', 'K_p = 11','K_p = 21')
```



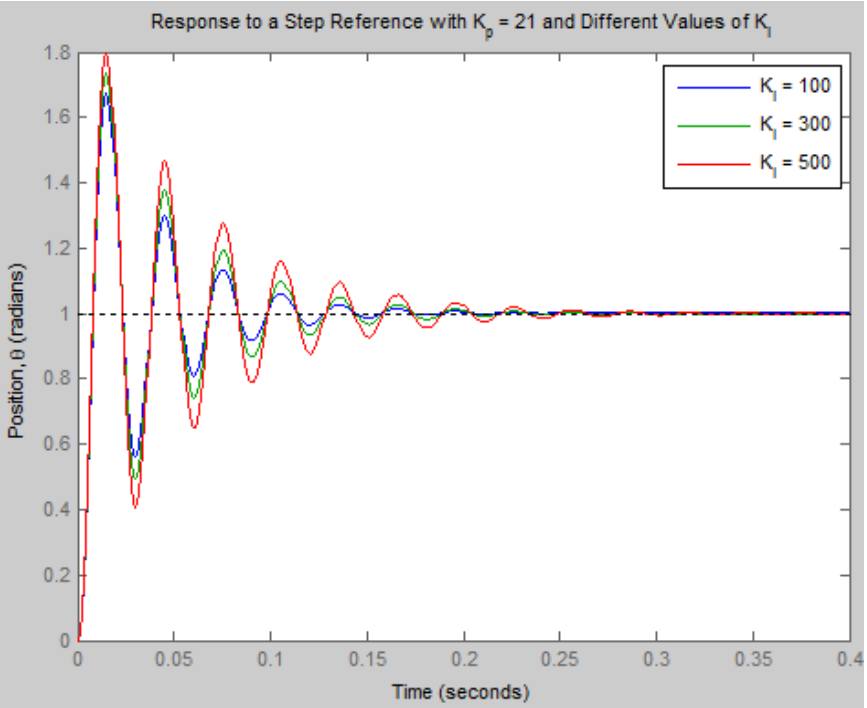
6.7 ábra A rendszer viselkedése terhelés alatt

A fenti ábrákon látható hogy nincs egyensúlyi állapot hiba az egység bementről, figyelmen kívül hagyva a Kp erősítést. Ez k=1 esetén látszik. Látható, hogy a rendszer hasonlóan viselkedik, ha egyensúlyi állapotban terhelés alatt van. Abban az esetben, ha a referenciát és a terhelést összeadjuk, az egyenlő lesz a két grafikon összegével. Ezt követi a szuperpozíció, ami használható a lineáris rendszerkre. Ebből kiindulva, ha akarunk egy egyensúlyi állapot hibát a terhelés jelenlétekor, a terhelésnek változnia kell, tartania kell a nulla felé. Minél nagyobb a K_p értéke, annál kisebb lesz az egyensúlyi állapot hibája, de soha nem éri el a nullát. Persze minél nagyobb értékkel dolgozunk, annál nagyobb lesz a túllövés és annál hosszabb lesz a beállási idő.

PI szabályzás

Először egy PI szabályzóval próbáljuk meg kiküszöbölni a terhelés egyensúlyi hibáját. A K_p-re 21-et veszünk és teszteljük az erősítést a K_i-vel 100-tól 500-ig. Az m állomány tartalmát cseréljük ki a következőre, és futtassuk a parancssorban! Az alábbi ábra fog kiraljzolódni:

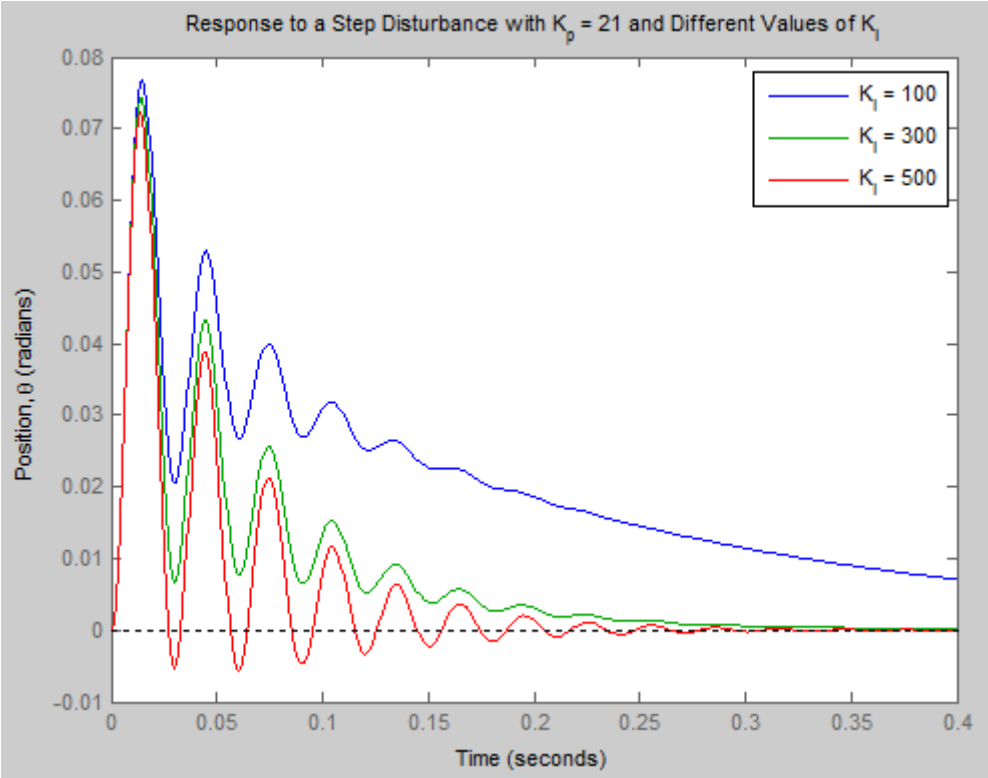
```
Kp = 21;
Ki = 100;
for i = 1:5
    C(:,:,i) = pid(Kp,Ki);
    Ki = Ki + 200;
end

sys_cl = feedback(C*P_motor,1);
t = 0:0.001:0.4;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with K_p = 21 and Different Values
of K_i')
legend('K_i = 100', 'K_i = 300', 'K_i = 500')
```



6.8 ábra Egyensúlyi hiba kiküszöbölése

Most nézzük meg milyen lesz a válasz a bemenet terhelésére!

```
dist_cl = feedback(P_motor,C);
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with K_p = 21 and Different Values
of K_i')
legend('K_i = 100',  'K_i = 300',  'K_i = 500')
```



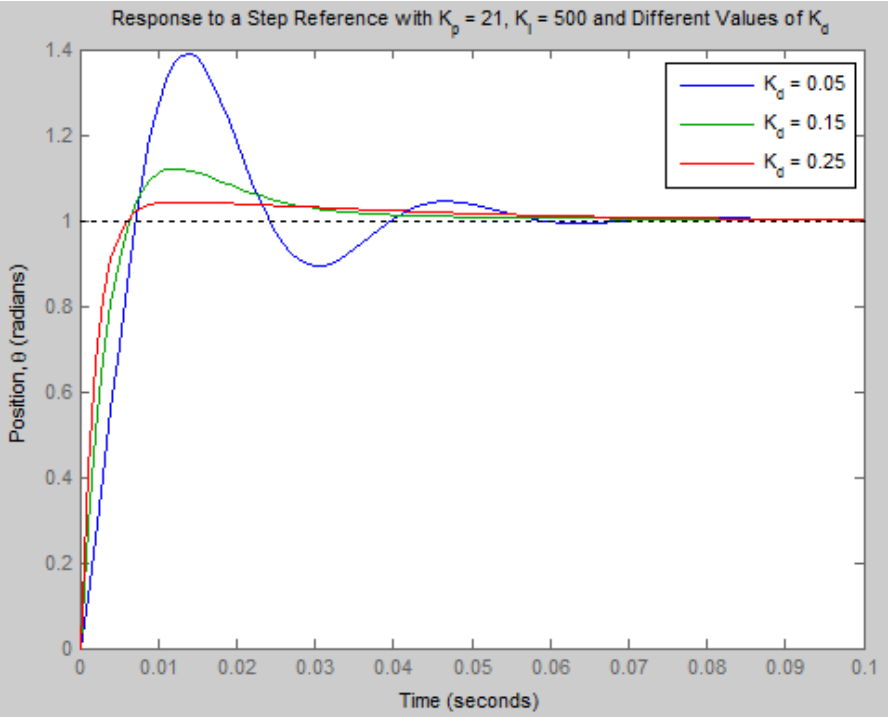
6.9 ábra Bemenet terhelése

Az integrális rész lecsillapította az egyensúlyi hibát nullára, még úgy is, hogy a bemeneti terhelés jelen van, ez volt a célja az integrál rész hozzáadásának. Amint látható az összes válasz hasonló a referens grafikonhoz, azzal hogy az oszcilláció nőtt, ahogy egyre nagyobb K_i-t vettünk, viszont látható, hogy terhelés jelentősen változott, ahogy a K_i-t változtattuk. Amikor különösen nagy erősítést használtunk a hiba sokkal gyorsabban tartott a nulla felé. Azért választottuk a K_i=500-at, mert a terhelés által okozott hiba nagyon gyorsan tart a nulla felé, még akkor is, ha a rendszernek több beállási idő kell, és több lesz az oszcilláció. Az oszcillációt és a beállási időt a deriváló tag hozzáadásával érhetjük el.

PID szabályzás

Próbáljuk a PID-t kis K_d értékekkel bővíteni!

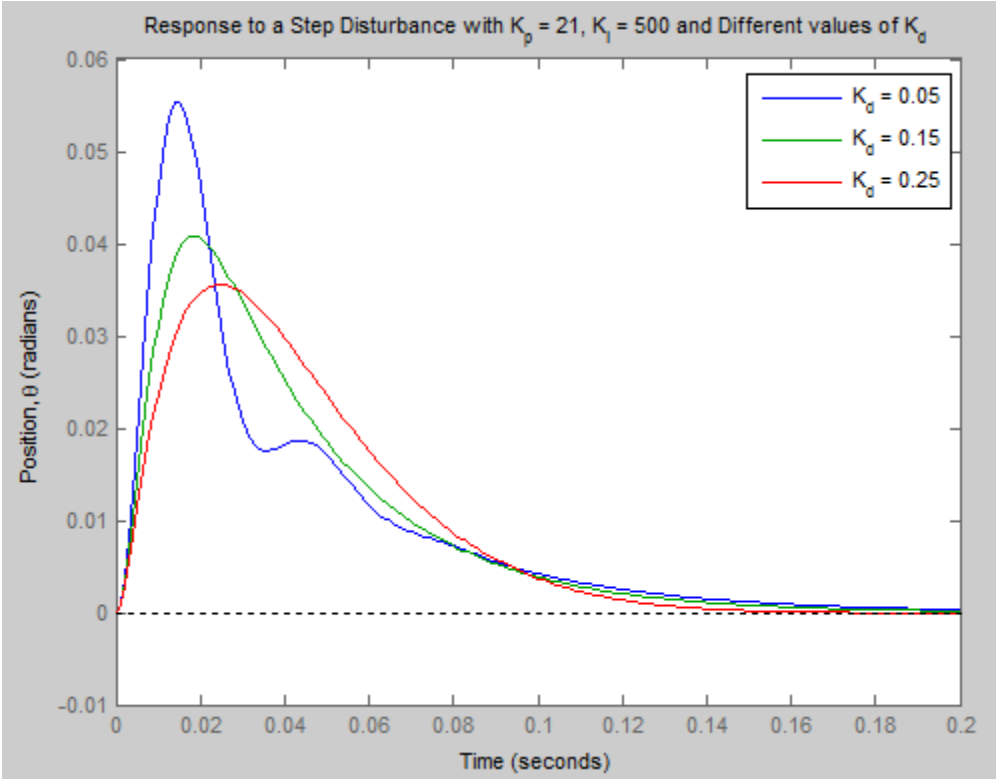
```
Kp = 21;
Ki = 500;
Kd = 0.05;

for i = 1:3
    C(:,:,i) = pid(Kp,Ki,Kd);
    Kd = Kd + 0.1;
end

sys_cl = feedback(C*P_motor,1);
t = 0:0.001:0.1;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with K_p = 21, K_i = 500 and
Different Values of K_d')
legend('K_d = 0.05', 'K_d = 0.15', 'K_d = 0.25')
```



6.10 ábra Kd érték hozzáadása a rendszerhez

Vizsgáljuk meg mi történik a gerjesztés válasszal, ha az m-file-unkba a következőt írjuk be!

```
dist_cl = feedback(P_motor,C);
t = 0:0.001:0.2;
step(dist_cl(:,:,1), dist_cl(:,:,2), dist_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Disturbance with K_p = 21, K_i = 500 and
Different values of K_d')
legend('K_d = 0.05', 'K_d = 0.15', 'K_d = 0.25')
```



6.11 ábra Gerjesztés válasz

A Kd=0.15 esetében elérhető a rendszer követelményeinek megfelelő válasz. Hogy pontosan meghatározzuk a válaszfüggvény karakterisztikáját, jobb kattintással a menüben ki kell választani a step response plot-ot, vagy a stepinfo parancsot tudjuk használni, mint a következő ábrán:

```
stepinfo(sys_cl(:,:,2))

ans =
        RiseTime: 0.0046
    SettlingTime: 0.0338
     SettlingMin: 0.9183
     SettlingMax: 1.1211
       Overshoot: 12.1139
      Undershoot: 0
            Peak: 1.1211
        PeakTime: 0.0121
```

Az eredményekből látható, hogy a beállási idő 34ms alá csökkent, a túllövés kicsivel 12% fölé, és nincs egyensúly állapot hibánk. Tudjuk tehát hogy egy olyan PID szabályzót kell terveznünk amely a következő értékekkel dolgozik:

- $Kp = 21$

- Ki = 500
- Kd = 0.15.

Állapottér módszerek a szabályozó tervezéséhez

Az adott probléma dinamikai egyenletei állapottér formában a következőképpen alakulnak:

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & -\frac{b}{J} & \frac{K}{J}\\ 0 & -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{1}{L}\end{bmatrix}V$$

$$y = \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\\ i\end{bmatrix}$$

Ezek az állapottér egyenletek szabványos formában:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

Ahhoz, hogy egy lépésben 1 rad/sec legyen a referencia, a tervezési kritériumok a következők:

- A beállási idő kevesebb, mint 40ms
- A túllövés kevesebb, mint 16%
- Állandósult állapotban ne legyen hiba, még ha a bemeneten zavar (terhelés) is van.

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;

A = [0 1 0
    0 -b/J K/J
    0 -K/L -R/L];
B = [0 ; 0 ; 1/L];
C = [1  0  0];
D = 0;
motor_ss = ss(A,B,C,D);
```

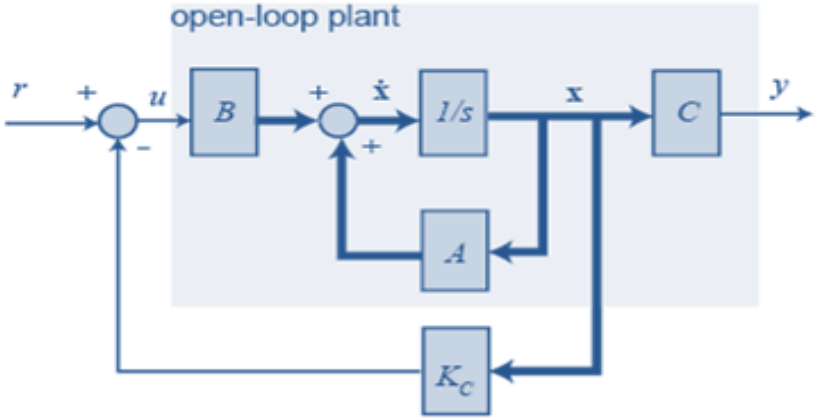
Visszacsatolt állapottér szabályozó tervezése

Mivel mindkét állapottér változó könnyen mérhető (egyszerűen adjunk hozzá egy árammérőt, a tachométert és egy potenciométert a pozíció meghatározásához), ezért nem kell hozzá megfigyelő.

A visszacsatolt állapottér szabályozó rendszerhez a következő törvény írható fel:

$$u = r - Kcx$$

Az alábbiakban a hozzá kapcsolódó sematikus ábra látható.



6.12 ábra Visszacsatolt állapottér szabályozó rendszer

A zárt visszacsatolású rendszert meghatározó karakterisztikus polinom *s I* - (*A* - *B* * *Kc*), ahol az *s* a Laplace változó. Mivel *A* és *B***Kc* mátrixok egyaránt 3x3 mátrixok, így három pólusa van a rendszernek. Az adott rendszer tervezésénél a három pólus elmozdítható. Ahhoz, hogy ellenőrizhessük az adott rendszer irányíthatóságát, ellenőrizzük az irányíthatósági mátrix rangját [*B AB* A^2 *B* ...]. A *ctrb* SCILAB parancs felépíti az *A* és *B* irányíthatósági mátrixot, továbbá a *rank* parancs az adott mátrix rangját adja meg. A következő parancsokkal ellenőrizzük a rendszer rendjét és a rendszer irányíthatóságát:

```
sys_order = order(motor_ss)
determinant = det(ctrb(A,B))


sys_order =
     3
determinant =
  -3.4636e+24
```

Az eredményekből látszik, hogy a rendszerünk szabályozható, hiszen a determinánsa a mátrixnak nem nulla, ezért a rendszer zárt hurok pólusait akárhol elhelyezhetjük az s térben. Először elhelyezzük a pólusokat -200, -100+100i és -100-100i. értékekre, az első pólus hatását elhagyva (mivel ez sokkal gyorsabb, mint a másik két pólus), a domináns pólusok megfelelnek egy másodrendű rendszernek, *zeta* = 0.5, ami megfelel 0.16%-os túllövésnek és

sigma = 100, ami megfelel 0.040-os beállási időnek. Most hogy meghatároztuk a pólusok pozícióját, használhatunk SCILAB parancsokat, hogy ismét meghatározzuk a szabályzó erősítés mátrixát K_c-t, hogy elérjük ezeket a pólusokat. A meghatározást továbbra is numerikusan végezzük. A következő programkódot kell az m-file végére írni:

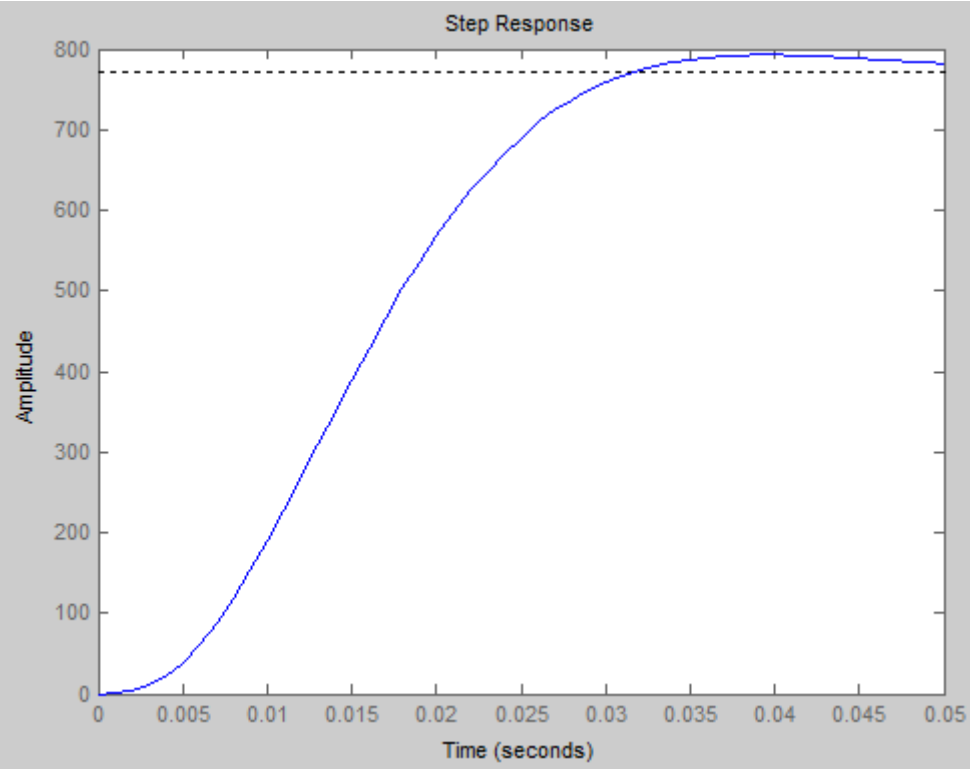
```
p1 = -100+100i;
p2 = -100-100i;
p3 = -200;
Kc = place(A,B,[p1, p2, p3])
Kc =
    0.0013   -0.0274   -3.9989
```

Ha az állapottér egyenletbe behelyettesíjük az $u = r - Kc\,x$ törvényt, akkor a következő kifejezést kapjuk:

$$\dot{x} = (A - BK_c)x + Br$$

$$y = Cx$$

```
t = 0:0.001:0.05;
sys_cl = ss(A-B*Kc,B,C,D);
step(sys_cl,t)
```



6.13 ábra Szabályzott rendszer

Digitális szabályozó tervezése

Ebben a fejezetben a DC motor pozíciószabályozó digitális változatát dolgozzuk fel. Ezt az analóg modell átalakításával fogjuk leírni.

A folytonos nyitott hurkú átviteli függvény a bemeneti feszültség és a kimeneti pozícióból származik:

$$P(s) = \frac{\Theta(s)}{V(s)} = \frac{K}{s((Js+b)(Ls+R)+K^2)} \quad \left[\frac{rad}{V}\right]$$

Ahhoz, hogy egy lépésben 1 rad/sec legyen a referencia, a tervezési kritériumok a következők:

- A beállási idő kevesebb, mint 40ms
- A túllövés kevesebb, mint 16%
- Állandósult állapotban ne legyen hiba, még ha a bemeneten zavar (terhelés) is van.

A rendszer mintavételezett adatmodellje

Egy digitális vezérlő rendszer tervezésének az első lépése az, hogy létrehozzunk egy mintavételezett adat-modellt. Szükséges megválasztani a mintavételezési frekvenciát.

A *zpk* paranccsal az átviteli függvényt egy olyan formára alakítjuk, ahol a nullák, pólusok és az erősítés egyértelműen láthatóak. A választott mintavételezési idő 0,001 másodperc, lényegesen gyorsabb, mint a rendszer dinamikája.

```
J = 3.2284E-6;
b = 3.5077E-6;
K = 0.0274;
R = 4;
L = 2.75E-6;
s = tf('s');
P_motor = K/(s*((J*s+b)*(L*s+R)+K^2));
zpk(P_motor)


ans =

      3086245930.9988
  ------------------------
  s (s+1.454e06) (s+59.23)

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

Ebben az esetben az adott átviteli függvényt a folyamatos Laplace tartományból diszkrét z-tartományba alakítjuk át. Az említett átalakítást a *C2D* parancs révén kaphatjuk meg. A *C2D* parancshoz három tényező szükséges: a rendszer modellje, a mintavételezési idő (Ts), és a tartószerv típusának a meghatározása. Ebben a példában nulladrendű (ZOH - Zero-order Hold) tartószervet feltételezünk.

```
Ts = 0.001;
dP_motor = c2d(P_motor, Ts, 'zoh');
zpk(dP_motor)


ans =

  0.0010389 (z+0.9831) (z+9.256e-07)
  ----------------------------------
          z (z-1) (z-0.9425)

Sample time: 0.001 seconds
Discrete-time zero/pole/gain model.
```

Amint felülről látható, van egy pólus, ami nagyon közel van a nullához, vagyis szinte elhagyható. Ha el akarjuk hagyni, a *mineral* panrancsal megtehetjük, ha a toleranciát 0.001-re állítjuk. Ha ezt a pólust így nullává módosítjuk, akkor csökkentjük az átviteli függvény

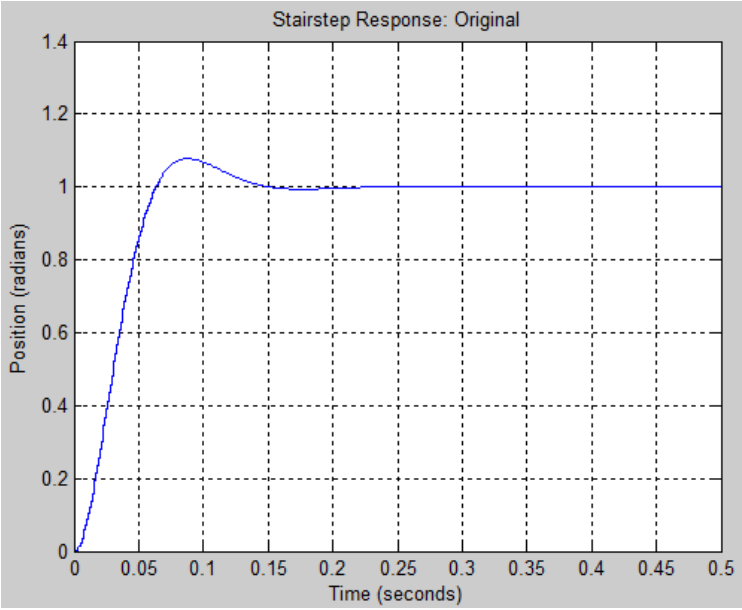
rendjét, és kikerüljük a numerikus nehézségeket a SCILAB-ban. A *minreal* parancs használata következő ábrán látható:

```
dP_motor = minreal(dP_motor,0.001);
zpk(dP_motor)
ans =

  0.0010389 (z+0.9831)
  --------------------
    (z-1) (z-0.9425)

Sample time: 0.001 seconds
Discrete-time zero/pole/gain model.
```

Első megközelítésre a zárt visszacsatolású rendszer válaszát szeretnénk elemezni, bármilyen kompenzáció nelkül. Ehhez be kell zárni a hurkot az átviteli függvényen a *feedback* parancs segítségével.

```
sys_cl = feedback(dP_motor,1);
[x1,t] = step(sys_cl,.5);
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Position (radians)')
title('Stairstep Response: Original')
grid
```

A hurok bezárása után megvizsgáljuk a rendszer válasz függvényét nulladrendű tartószervvel (*step* és *stairs* parancs).



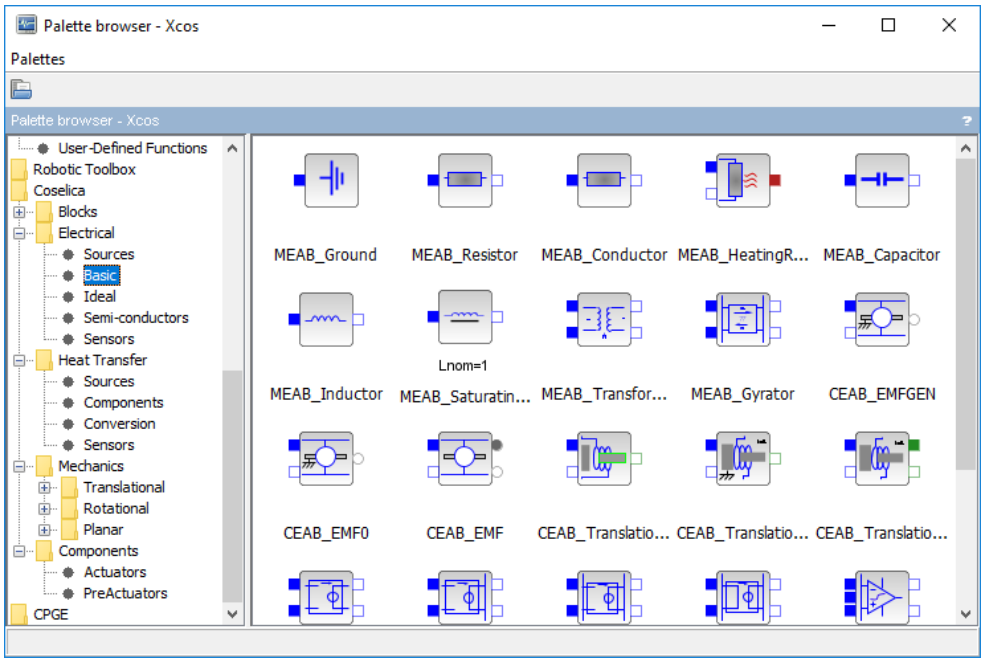
6.14 ábra Zárt visszacsatolású rendszer válasza

Ellenőrző kérdések:

1. Milyen ekvivalens elemekből tevődik össze a DC motor modellje?
2. Milyen visszacsatolású rendszereket ismer?
3. Hogyan kapjuk a DC motor átviteli függvényét?
4. Vázolja fel a visszacsatolt állapottér szabályozó modelljének shematikus rajzát.
5. Tervezzen XCOS rendszerben egy DC motor fordulatszám szabályozót.

7. COSELICA

A mechatronikai rendszerek dinamikus modellezésére számos lehetőség kínálkozik. Az ipari trendeket megfigyelve megállapítható, hogy a szabványos Coselica nyelv nagy áttörés előtt áll az objektum-orientált matematikai modellezés terén. A Coselica szabvány lassan 10 éve tartó folyamatos fejlődését egy nemzetközi nonprofit szervezet irányítja. Ez a modern, objektum-orientált modellező nyelv lehetővé teszi egy komplex mechatronikai rendszer több tudományterületen átívelő leírását, matematikai differenciálegyenletekre alapozva. A folyamat- és irányításorientált komponensek mellett analóg módon támogatja a mechanikai, elektronikai, hidraulikai, pneumatikai és hőtani komponensekből álló komplex rendszerek leírását [6]. A Coselica modellek általános nemlineáris differenciálegyenleteket (ODE), differenciálalgebrai egyenleteket (DAE) valamint állapotgépeket és Petri-hálókat is tartalmazhatnak. A Coselica blokk-komponensek szabványosított egy- vagy kétirányú csatlakozókkal kapcsolhatók egymáshoz. Az utóbbi típusú csatlakozókon történő adatfolyamra a Kirchhoff-féle csomóponti törvények implicite értelmezettek (a csatlakozóba befolyó és az onnan kifolyó folyamok – pl. elektromos áramok – előjeles összege zérus, összekötött csatlakozók potenciál értéke – pl. hőmérséklet – pedig mindig megegyezik).



7.0 ábra A Coselica eszköztára

7.1 Elektronikai szimuláció

Multivibrátorok fogalma, típusai

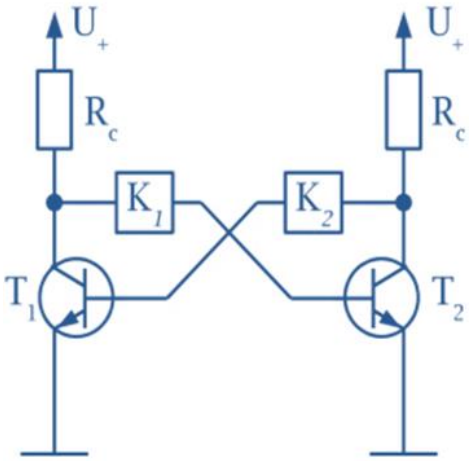
A billenő áramkörök vagy ismertebb nevükön multivibrátorok, pozitívan visszacsatolt univerzális digitális áramkörök, melyeket négyszögjelek előállítására, időzítésre, hiszterézises komparátor, impulzusszélesség moduláció (PWM) használunk [15]. Kimeneti feszültségük nem folytonosan változik, hanem két, az áramköri paraméterek által meghatározott értéket vehet fel (logikai alacsony illetve logikai magas). Az egyes állapotok közötti átbillenés több különböző módon is történhet; ezek szerint létezik:

- **bistabil multivibrátor**: A neve is mutatja, mindkét állapota stabil. A kimeneti jelszint csak akkor változik, ha az átbillenési folyamatot egy bemeneti jel kiváltja. A tápfeszültség meglétéig az egyik állapotban marad, ha nem billentjük át a másik állapotba. Nincs alap állapota, az áramköri aszimmetria dönti el, hogy alaphelyzetben melyik állapotában van.
- **monostabil multivibrátor**: Egy stabil állapota van. A másik (instabil) állapotát egy bemeneti jellel válthatjuk ki, és az csak a visszacsatoló hálózat alkatrészek értékei által meghatározott ideig marad fenn. Ezen idő eltelte után az áramkör automatikusan visszabillen a stabil állapotába. A tápfeszültség meglétéig a stabil állapotát képes megőrizni, ha nem billentjük át az instabil állapotba. Alap állapota a stabil állapot.
- **astabil multivibrátor**: Nincs stabil állapota. Külső vezérlés nélkül, periodikusan változtatja kimeneti feszültségszintjét, „billeg” a két állapota között. A tápfeszültség meglétéig billeg. Alap állapotot itt nem tudunk értelmezni.

A billenő áramköröket általánosan a 7.1. ábra szemlélteti. Ezen áramkör tranzisztorokból (NPN), ellenállásokból és/vagy kondenzátorokból áll. Ezek a billenő áramkörök a TTL áramkörcsalád térhódításával már nyomtatott áramkörökkel is megvalósíthatók. A NE555 egy nagyon sok feladatra alkalmas IC, most multivibrátoros kapcsolásokhoz használjuk fel. A billenőkör típusát a visszacsatolások (K_1, K_2) határozzák meg (lásd 7.1 táblázat)

7.1 Táblázat Billenőkör típusa a visszacsatolás alapján

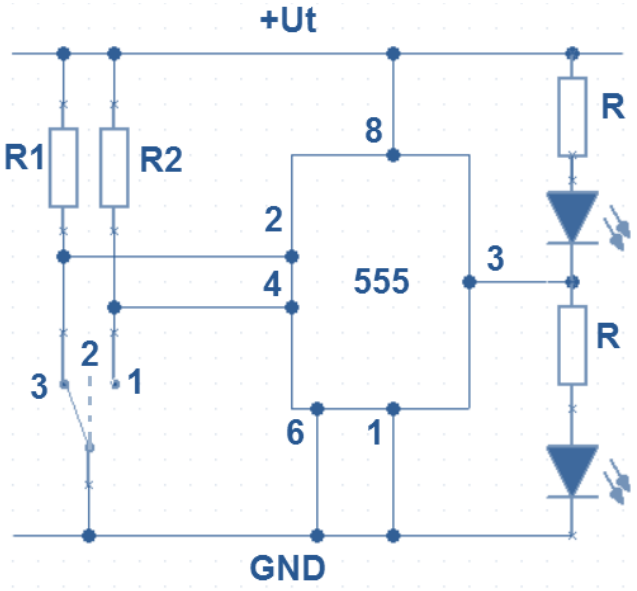
Típus	**K_1 csatoló**	**K_2 csatoló**
Bistabil	Ellenállás (R_1)	Ellenállás (R_2)
Monostabil	Ellenállás (R)	Kondenzátor (C)
Astabil	Kondenzátor (C_1)	Kondenzátor (C_2)



7.1 ábra Billenőkörök blokksémája

Bistabil multivibrátor NE555 IC-vel

A bistabil billenő kör flip-flop-ként működik: a kimenet vagy magas (logikai 1) vagy alacsony (logikai 1). A kapcsolóval a trigger (2) és reset (4) lábakat kapcsolgatjuk a földre, melyek alapból fel vannak húzva a tápfeszültségre. A bistabil multivibrátor kapcsolási rajza NE555 IC-vel a 7.2 ábrán látható. A szemléltetés kedvéért 2db LED-et is lehet kapcsolni a kimenetre.



7.2 ábra Bistabil multivibrátor NE555 IC-vel

Kezdetben, ahogyan a rajzon van, a trigger (2)-t kapcsoljuk a földre (GND). Ekkor az NE555 IC belsejében található K_1 és K_2 komparátor bemenetei közti egyenlőség nem fog teljesülni és az IC kimenetén magas (logikai 1) szint lesz [15]. Ha a kapcsolót átkapcsoljuk az áramkör belső tranzisztoros flip-flop-jának reset-je lép működésbe, azaz a kimenet visszaáll alacsony (logikai 0) szintre. A működés állapotai a 7.2 táblázatban láthatók.

7.2 Táblázat Bistabil multivibrátor működésének állapotai

Kapcsoló	1 (R)	2	3 (S)	2	1 (R)
K_1	Magas	Magas	Alacsony	Magas	Magas
K_2	Magas	Magas	Magas	Magas	Magas
$\bar{Q}$	Magas	Magas	Alacsony	Alacsony	Magas
Q (OUT)	Alacsony	Alacsony	Magas	Magas	Alacsony

Bistabil multivibrátor „időzítése”

A monostabil és az astabil multivibrátorral szemben a bistabil billenőköröknél az időzítését nem tudjuk értelmezni, mert a tápfeszültség meglétéig vagy a külső vezérlőjel érkezéséig az előző állapotában marad a bistabil.

Monostabil multivibrátor NE555 IC-vel

Monostabil, mert csak egy stabil állapota van. Ha átbillentjük a másik állapotba, t_1 idő múlva visszabillen a stabil állapotába.

Időzítésre használható kapcsolás: egy adott hosszúságú impulzus állítható elő a trigger jel segítségével (negatív logika). Amikor bekapcsoljuk a tápfeszültséget a stabil állapotában van a multivibrátor és a kimenet alacsony (logikai 0) szinten van, ha megérkezik a trigger jel (logikai 0), akkor R ellenálláson keresztül C kondenzátor elkezd feltöltődni. Amikor a kondenzátorra eső fezsültség eléri a tápfeszültség 2/3-adát (2/3 Ut-t), kimenet alacsony (logikai 0) szinten lesz és bekapcsol a kisütő tranzisztor (DIS). Ez a stabil állapot, amíg trigger jel nem érkezik, aminek hatására a kimenet magas (logikai 1) szinten lesz és C kondenzátor újból töltődni kezd, míg újból el nem éri a tápfeszültség 2/3-adát (2/3 Ut-t).

A kvázistabil állapot ideje megegyezik a kondenzátor töltési idejével, amely a következő összefüggéssel számítható: $t_1=1,1*R*C$

A monostabil akkor működik helyesen, ha a triggerimpulzus szélessége rövidebb, mint az időzítés.

Stabil állapotban a kimeneten alacsony szint van (kb. 0V), mert az RS tároló kimenete magas, ami a tranzisztort nyitva tartja. A tranzisztor pedig nem engedi feltöltődni a kondenzátort.

Ebből a stabil állapotból úgy tudjuk kibillenteni a multivibrátort, hogy a trigger bemenetre alacsony szintet adunk, ekkor az RS tároló kimenete 0 (az IC kimenete pedig magas), aminek hatására a tranzisztor lezár. A lezárt tranzisztor miatt a C kondenzátor R ellenálláson keresztül elkezd töltődni, ami addig tart, amíg a K_2 komparátor billenéséhez szükséges 2/3 Ut feszültséget el nem éri a kondenzátor feszültsége. Ekkor a K_2 komparátor átbillen, RS tároló resetel, RS tároló kimenete magas (logikai 1) szintre vált, az NPN tranzisztor pedig kinyit, C kondenzátor pedig elkezd kisülni. A kimenet pedig lemegy alacsony (logikai 0) szintre.

Monostabil multivibrátor időzítése

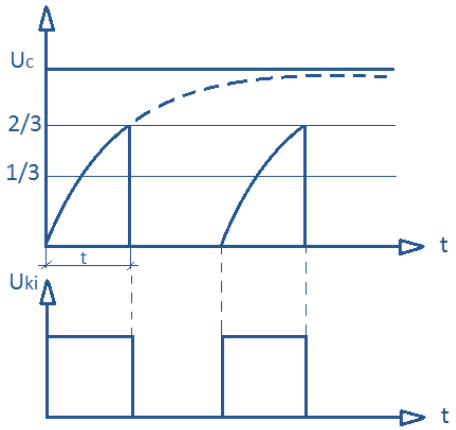
Külső beavatkozás nélkül az egyik állapotában van (stabil) és ebben az állapotban is marad korlátlan ideig. Külső beavatkozásra, a trigger bemenetre adott impulzus hatására, a másik állapotukba billennek egy előre meghatározott ideig, amit a monostabil egyik visszacsatoló áramkörében található R és C elemek szabnak meg. Ezt az időtartamot időzítésnek nevezik. Ezen időzítés letelte után maguktól visszabillennek stabil állapotukba.

$$t_1=R_1*C*\ln 3=1{,}1*R_1*C$$

, ahol t_1 = billenőkör időzítése [s]

R_1 = ellenállás [Ω]

C = kondenzátor [F]



7.3 ábra Monostabil multivibrátor idődiagram

A képlet értelmezése a 7.3 ábra szerint történik. Az IC kimenete magas (logikai 1) szinten van mindaddig, amíg C kondenzátorra eső feszültség értéke el nem éri a tápfeszültség 2/3-adát (2/3 Ut).

A fenti képlet levezetése az alábbi módon történik:

$$t = -RC\ln\left(1-\frac{U_c}{U_0}\right) = RC\ln\left(\frac{1}{1-\frac{U_c}{U_0}}\right)$$

$$t = RC\ln\left(\frac{1}{1-\frac{\frac{2}{3}U_0}{U_0}}\right) = RC\ln\left(\frac{1}{\frac{1}{3}}\right) = RC\ln 3 = RC\,\mathbf{1,1}$$

Ellenállás és kondenzátor kiválasztásának irányelve

- Úgy kell megválasztani az R ellenállást, hogy 20MΩ-nál nagyobb ne legyen, de 1kΩ-nál kisebb se.
- Ha a számítás olyan ellenállásértéket eredményez, amely nem kapható a kereskedelmi forgalomban, akkor soros, illetve párhuzamos kapcsolással kell előállítani, ha a

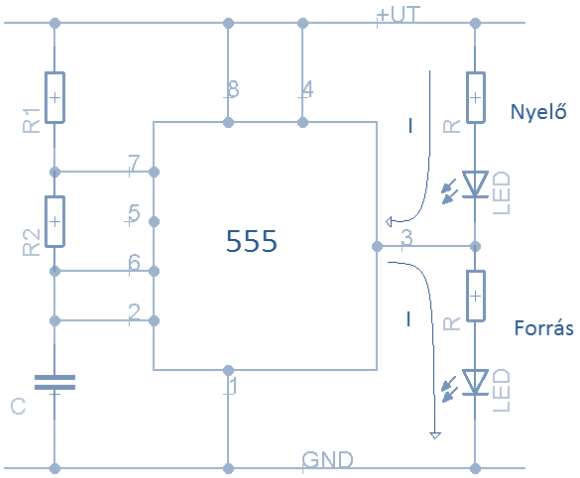
megadott t időhöz 100% pontosan ragaszkodunk.

- A kondenzátort úgy kell megválasztani, hogy ne legyen túl nagy, mert akkor pontatlanságot fog okozni, ami pontos időzítéseknél nem megengedhető.

Astabil multivibrátor NE555 IC-vel

Astabil, mert nincs stabil állapota. Bármelyik állapotba billentjük idővel átbillen az ellenkező állapotba, ezért nincs is szükség külső vezérlő jelre, az áramkör magától billeg a két állapot között, a tápfeszültség meglétéig.

Felhasználható négyszögjel-generátornak, de mérésre is: a kimeneti frekvencia mérésével megmérhető az R ellenállás vagy C kondenzátor értéke. Az IC-re tápfeszültséget kapcsolva, a kimeneten adott időközönként változik a jelszint. A kapcsolási rajza a 7.4 ábrán látható, ahol a szemléltetés kedvéért 2 db LED-et is kapcsoltam a kimenetre.



7.4 ábra Astabil multivibrátor NE555 IC-vel

Astabil multivibrátor működése

Abból kiindulva, hogy a kondenzátor feszültsége 0 V, ez az állapot csak akkor lép fel, amikor a tápfeszültséget rákapcsoljuk a rendszerre.

A C kondenzátor az R_1 és R_2 ellenállásokon keresztül töltődik, amíg a kondenzátor feszültsége eléri a tápfeszültség 2/3-adát (2/3 Ut). Ebben az időpontban a K_2 komparátor kimenete állapotot vált, billenti az RS tárolót. A kimeneti feszültség hozzávetőlegesen 0 V lesz és kinyit az NPN tranzisztor is. A nyitott tranzisztor az R_2 ellenálláson keresztül kezdi kisütni a C kondenzátort. A kisütés addig tart, amíg a kondenzátor feszültsége a tápfeszültség 1/3-ada (1/3 Ut) alá nem csökken. A K_2 komparátor visszabillenti az RS tárolót, a kondenzátor kisütése megszűnik, mert az NPN tranzisztor lezár, az IC kimeneti feszültség pedig visszaáll magas (logikai 1) szintre.

Astabil multivibrátor időzítése

Az előző alfejezetben leírt folyamat periodikusan ismétlődik. Az astabil multivibrátor frekvenciáját a töltés és kisütés idejét meghatározó R_1 ellenállás, R_2 ellenállás és C kondenzátor elemek értékeiből lehet kiszámítani a következő összefüggés szerint:

$$f = \frac{1}{\ln(2) \cdot C \cdot (R_1 + 2R_2)}$$

A kondenzátor töltődik az R1 ás R2 ellenállások soros eredőjén keresztül, amíg el nem éri a felső komparátor komparálási szintjét, a tápfeszültség 2/3-adát (2/3 U_t).

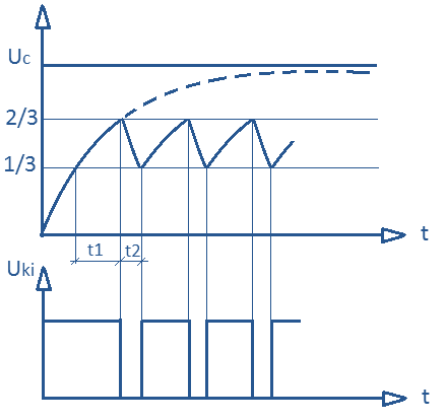
A töltéshez: $t_1=(R_1+R_2)C\ln 2=0{,}693(R_1+R_2)C$ idő kell.

A kisütéshez pedig: $t_2=0{,}693*R_2*C$ idő kell.

Ezzel a rezgési frekvencia:

$$f = \frac{1}{t_1 + t_2} = \frac{1}{0{,}693\ C\ (R_1 + 2R_2)} = \frac{1{,}44}{C(R_1 + 2R_2)} = \frac{1}{\ln 2\ (R_1 + 2R_2)C}$$

A 7.5 ábrán keresztül szemléletesen látható, hogy a kondenzátor feltöltése és kisütése nem azonos időt igényel, továbbá az első bekapcsoláskor a töltéshez nem t_1 idő szükséges, hanem t_1 + x idő.



7.5 ábra Astabil multivibrátor időidiagram

$$T=t_1+t_2 \rightarrow T = \ln 2\ (R_1 + 2R_2)C$$

Ahol:

T = periódusidő [s]

t_1 = kondenzátor töltési ideje [s]

t_2 = kondenzátor kisütési ideje [s]

Ezen két periódusidő levezetése a következőképp néz ki:

A kondenzátor töltéséhez szükséges idő:

$$t_1 = (R_1 + R_2)C * \ln\left(\frac{1}{1 - \frac{\frac{2}{3}U_0}{U_0}}\right) - (R_1 + R_2)C * \ln\left(\frac{1}{1 - \frac{\frac{1}{3}U_0}{U_0}}\right) =$$

$$= (R_1 + R_2)C\left(\ln\left(\frac{1}{\frac{1}{3}}\right) - \ln\left(\frac{1}{\frac{2}{3}}\right)\right) = (R_1 + R_2)C\left(\ln 3 - \ln\frac{3}{2}\right) =$$

$$= 0{,}693 * (R_1 + R_2)C$$

A kondenzátor kisütéséhez szükséges idő pedig a következőképp alakul:

$$t_2 = R_2C * \ln\left(\frac{1}{1 - \frac{\frac{2}{3}U_0}{U_0}}\right) - R_2C * \ln\left(\frac{1}{1 - \frac{\frac{1}{3}U_0}{U_0}}\right) =$$

$$= R_2C\left(\ln\left(\frac{1}{\frac{1}{3}}\right) - \ln\left(\frac{1}{\frac{2}{3}}\right)\right) = R_2C\left(\ln 3 - \ln\frac{3}{2}\right) =$$

$$= 0{,}693 * R_2C$$

Az NE555 ic összeállítása Coselica környezetben

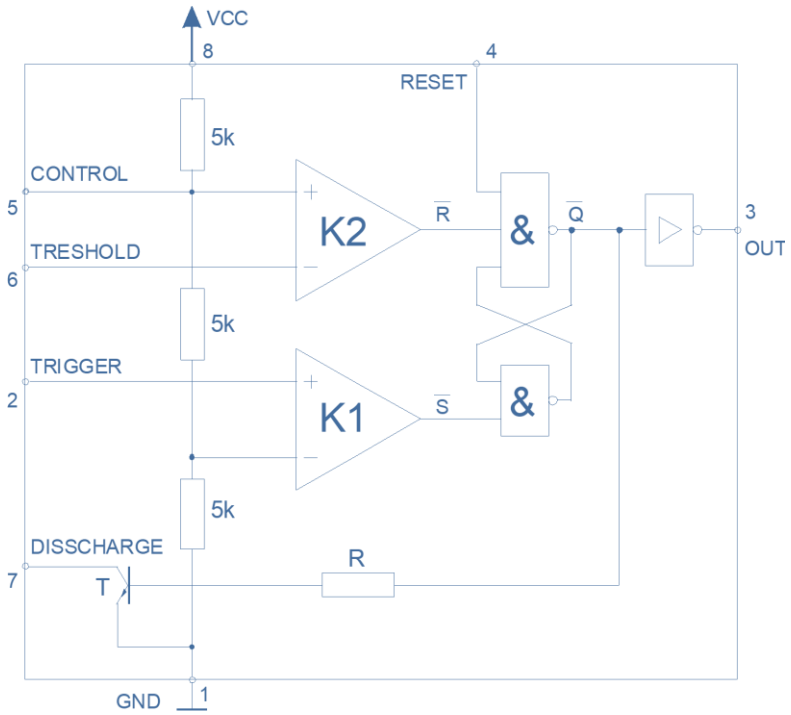
A 7.3 táblázatban láthatók a NE555 IC be illetve kimenetei és azoknak a meghatározása.

7.3 Táblázat NE555 be és kimenetei

Láb	Megnevezés	Leírás
1	GND	Alap referenciafeszültség, föld (0 V)
2	Trigger	K_1 komparátor nem invertáló bemenete
3	OUT	NE 555 kimenete (logikai alacsony vagy magas)
4	Reset	RS tároló engedélyező bemenete
5	Control	K_2 komparátor referencia feszültséget kompenzáló bemenet
6	Treshold	K_2 komparátor invertáló bemenete
7	Disscharge	Úgynevezett kisütő kimenet (kimenet negáltja)
8	V_{CC}	NE555 tápfeszültsége (3 – 15 V)

Az NE555 IC felépítése

Az NE555 IC belső felépítése a 7.6 ábrán látható. A továbbiakban az ábra segítségével lesz ismertetve az IC belső felépítése, paraméterei és működése.



7.6 ábra NE555 IC belső felépítése

- 3 db 5 kΩ ellenállásokból álló feszültségosztó
- 2 db komparátor (K_1 és K_2)
- Komparálási szint K_2 komparátornál (felső komparátor) 2/3 Ut, ami a control (5) bemeneten kis mértékben változtatható
- Komparálási szint K_1 komparátornál (alsó komparátor) 1/3 Ut
- K_2 invertáló, K_1 nem invertáló komparátor
- Külön törlőbemenettel rendelkező RS tároló (felső NAND kapu törlő bemenettel ellátott)
- NPN tranzisztor melyet az RS tároló hajt meg (kondenzátor kisütésére használható)
- A tranzisztor akkor van nyitva, ha a kimenet alacsony (logikai 0) szinten van
- RS tároló kimenete után található egy erősítő, ami egyben invertál is
- 200 mA áramot tud maximálisan kiadni (maximális terhelhetőség)
- Tápra és földre is tud húzni, ellenben az NPN tranzisztor földre, a PNP tranzisztor tápra húz
- Forrásként és nyelőként is tud üzemelni, az NPN tranzisztor csak nyelőként, a PNP tranzisztor csak forrásként tud üzemelni
- Szabad kimenetet zavarszűrő kondenzátoron keresztül a földre kellene kötni (pl 5-ösre)

Az NE555 IC belső felépítése az előző fejezet alapján lett megvalósítva Coselica környezetben. A külön törlő bemenettel rendelkező RS tároló helyett egy szokványos RS

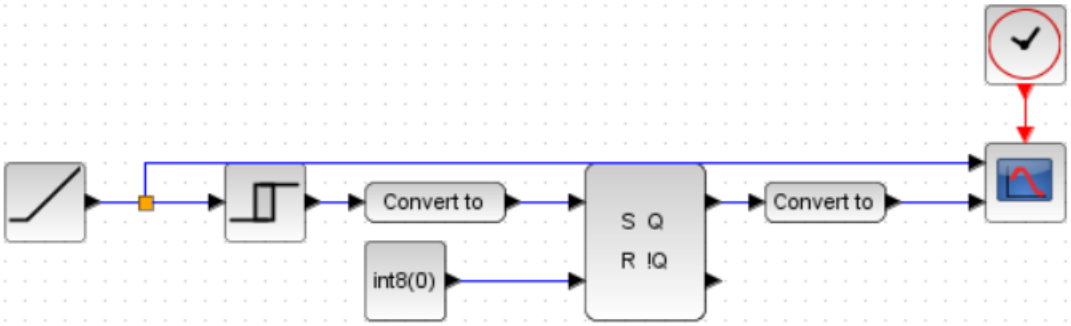
tároló használható, mert az astabil multivibrátoroknál az RS tároló törlő bemenete fel van húzva a tápfeszültségre (logikai 1), ami azt jelenti, hogy beírás/ törlés engedélyezett, vagyis szokványos RS tárolóként használjuk az astabil multivibrátoroknál.

Továbbá Coselica környezetben komparátor nem található, csupán műveleti erősítő. Műveleti erősítő is tud komparátorként működni, de a billenés sebessége valamivel lassabb, ezt most figyelmen kívül hagyjuk. Visszacsatolás nélkül azonban nem üzemeltethető a műveleti erősítő, tehát csak hiszterézises komparátorként tudjuk használni, ami nekünk nem jó vagyis nullkomparátort kellene használnunk.

RS tároló megvalósítása

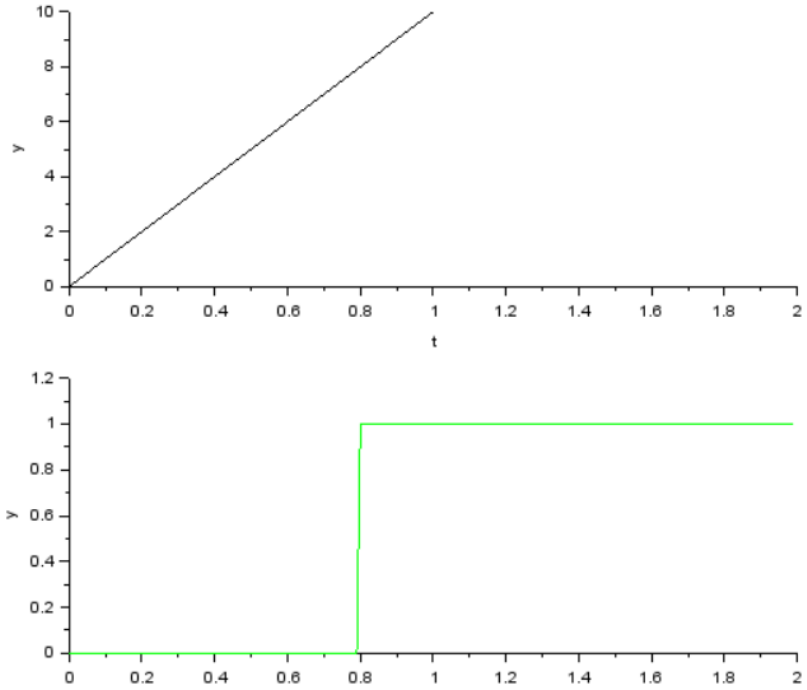
A komprátorok Coselica környezetben lettek modellezve, de az RS tároló már Xcos környezetben, a kettő közötti összeköttetésről is gondoskodni kellett. Az RS tárolót a K_1 és K_2 komparátor hajtja meg (analóg kimenetűek: 0-10 V), az RS tároló pedig bool (alacsony – magas) értékkel működik. Ennek kiküszöbölésére egy hiszterézist került a komparátorok és az RS tároló közé. 8 V fölött logikai 1 (magas) és 5 V alatt logikai 0 (alacsony) szintet ad ki.

A Scilabban összeállított RS tároló a 7.7 ábrán látható. Az RS tároló számára a hiszterézisben előállított (alacsony – magas) értéket még át kell alakítani int típusúra, hogy a tároló értelmezni tudja.



7.7 ábra RS tároló összeállítás

A diagramok a 7.8 ábrán láthatóak. A bemeneti jel egy egység sebesség ugrás függvény. A hiszterézisben beállított 8 V elérésekor billenti a bistabilt. A bemeneti jel ≈ 0,8 s-nál billen a hiszterézisben magas (logikai 1) szintre → és az RS tároló is ekkor billen magas (logikai 1) szintre. Az átmenthez idő kell, a diagramról ez is leolvasható, az átmenet alacsony és magas szint között nem függőleges.

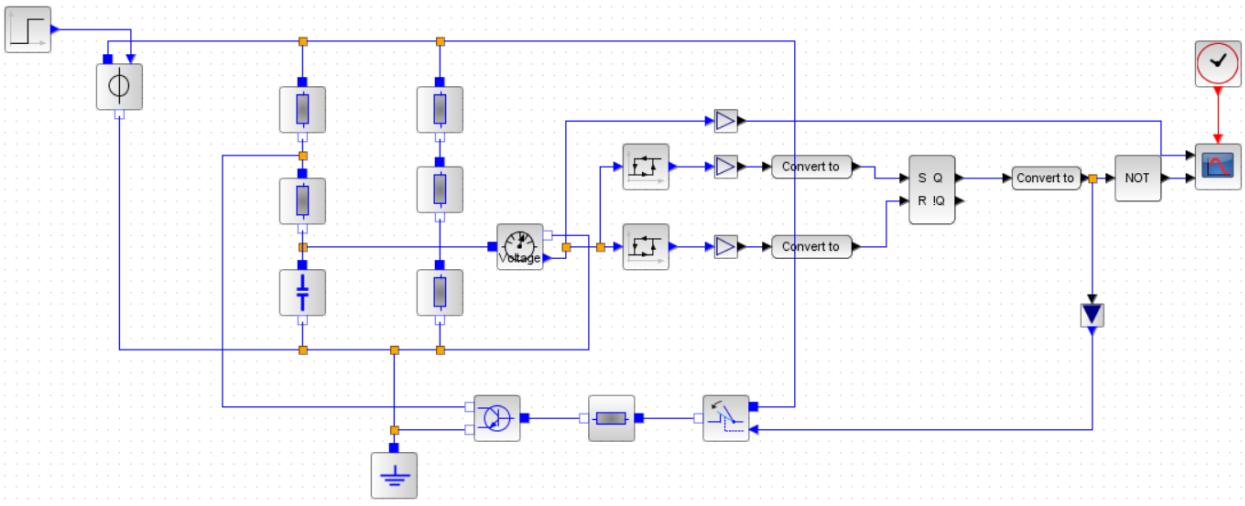


7.8 ábra RS tároló beírás (set)

A tároló törlése (reset) hasonló módon valósul meg, az R bemenetre kell magas (logikai 1) szintet adni és a tároló az ellentétes állapotába billen.

Astabil multivibrátor coselica környezetben

A Coselica környezetben összeállított NE555 IC belső felépítése a 7.9 ábrán látható. Az NE555 IC 10 V egyenfeszültségről üzemel.



7.9 ábra Astabil multivibrátor NE555 IC-vel

Az ábráról látható, hogy a Coselica és Xcos környezet összekapcsolása úgynevezett Potencial sensor és Real input segítségével történik. Az RS tároló kimenete visszacsatolás útján működteti az NPN tranzisztort, tehát Xcos környezetből kell áttérni Coselica környezetbe.

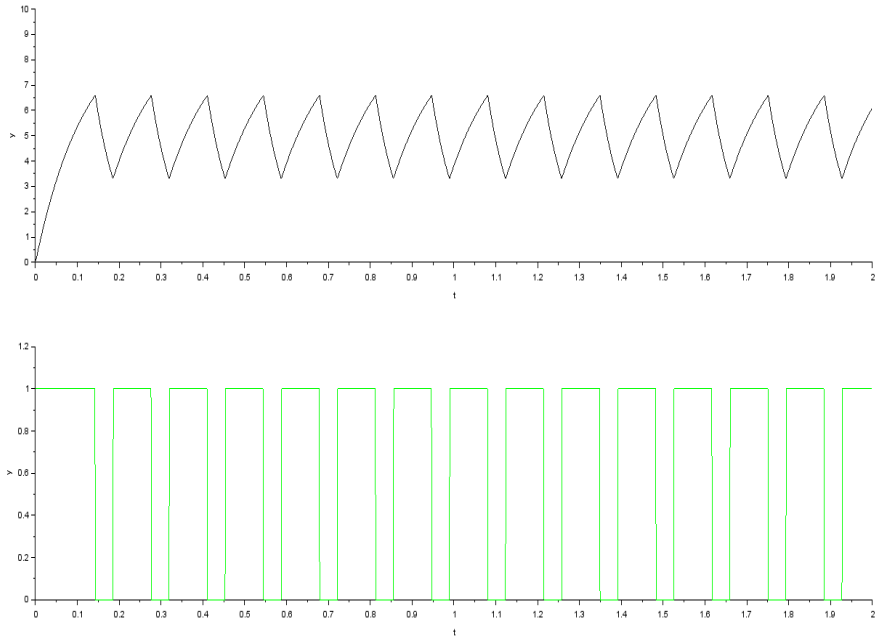
Ehhez egy real output-ot és a záró érintkezőt (kapcsolót) alkalmazhatunk, amit a Real output vezérel, az érintkezők zárása esetén a 10V egyenfeszültség működteti az NPN tranzisztort.
Az előző fejezetben tárgyaltak alapján kijelenthetjük, hogy Coselica és Xcos környezetben komparátor nem valósítható meg műveleti erősítővel. A Coselica eszköztárban nullkomparátor nem, csupán hiszterézises komparátor található.
Az eredeti kapcsolásban a K_2 komparátor billenési szintje $\frac{2}{3}$ U_t, tehát jó közelítéssel 10 V tápfeszültség esetén 6,66 V-nak is vehetjük.

- Az eredeti kapcsolásban a K_1 komparátor billenési szintje $\frac{1}{3}$ U_t, tehát jó közelítéssel 10 V tápfeszültség esetén 3,33 V-nak is vehetjük.
- A K_2 komparátor helyén található hiszterézisnél 6,6V és 6,66 V a felső, illetve az alsó billenési szint. Látható, hogy 0,03 V a hiszterézis (kedvezőtlen), de ez még elfogadható.
- A K_1 komparátor helyén található hiszterézisnél 3,3V és 3,33 V a felső, illetve az alsó billenési szint. Látható, hogy 0,03 V a hiszterézis (kedvezőtlen), de ez még elfogadható.

Bemenetre adott válaszfüggvény 66 % kitöltöttséggel:

A 7.10 ábra felső részén látható az astabil multivibrátor kondenzátorára eső feszültsége cscope-ban, a 7.10 ábra alsó részén pedig a bemenetre adott válaszfüggvénye (logikai alacsony – magas).
A visszacsatoló hálózat paraméterei:

- $R_1 = 100\ \Omega$
- $R_2 = 100\ \Omega$
- $C = 600\ \mu F$



7.10 ábra A szimuláció eredménye

A tápfeszültség (10V DC) 0,45 s után vált magasra (egységugrás). Az ábráról leolvasható, hogy ez nem végtelen gyorsan történik, hanem a váltásra átmeneti időre van szükség. Továbbá az első periódus hosszabb, mint a többi, mert ilyenkor a kondenzátor töltöttsége még 0 V, az első periódus után, $\frac{1}{3}U_t$ és $\frac{2}{3}U_t$ között változik a kondenzátorra eső feszültség. Az ábra alapján az alacsony és magas szint ideje:

- A magas szint ideje: $T_H \approx 83$ ms
- Az alacsony szint ideje: $T_L = 41$ ms
- Periódusidő: $T_p = T_L + T_H = 83$ ms + 41 ms = 124 ms

Ezek alapján a kitöltési tényező (ha $R_1 = R_2$)

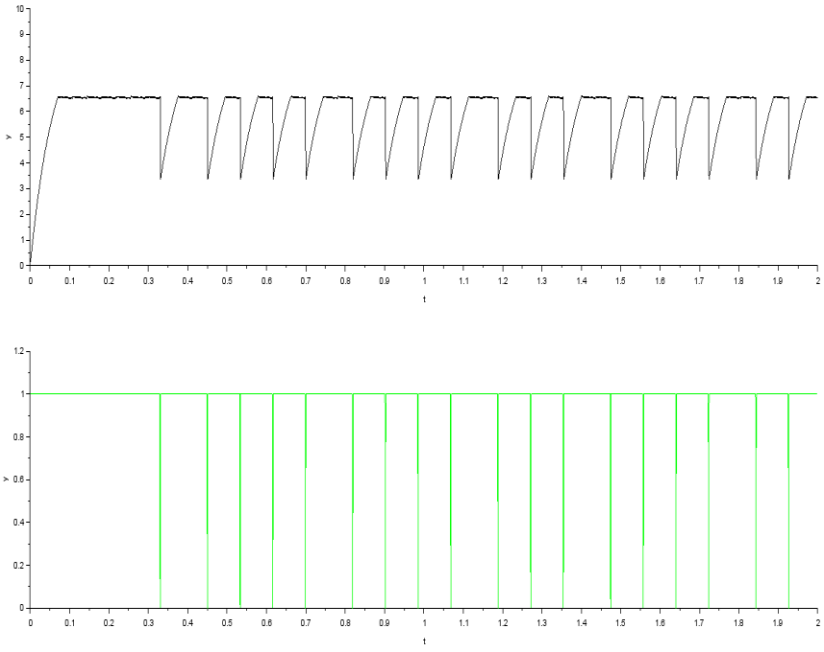
$$k = \frac{T_{ON}}{T_{ON}+T_{OFF}} = \frac{R1}{R1+2R2} \approx 66\%$$

Bemenetre adott válaszfüggvény 100 % kitöltöttséggel:

A következő szimulációban a visszacsatoló hálózat értékei meg lettek változtatva a következőkre:

- $R_1 = 100\ \Omega$
- $R_2 = 0\ \Omega$
- $C = 600\ \mu F$

A 7.11 ábra felső részén látható az astabil multivibrátor kondenzátorára eső feszültsége cscope-ban, a 7.11 ábra alsó részén pedig a bemenetre adott válaszfüggvénye (logikai alacsony – magas), abban az esetben, ha az R_2 ellenállás értéke 0.



7. 11 ábra A szimuláció eredménye

Az ábra alapján az alacsony és magas szint ideje:

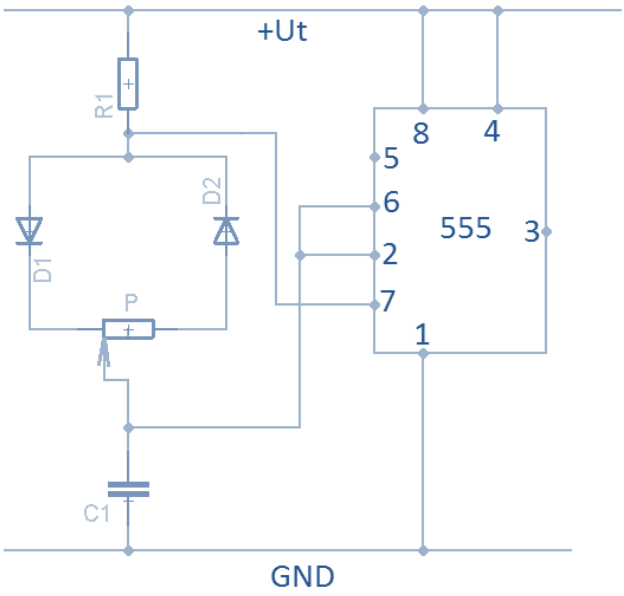
- A magas szint ideje: $T_H \approx 41$ ms
- Az alacsony szint ideje: $T_L = 0$ ms
- Periódusidő: $T_p = T_L + T_H = 41$ ms $+ 0$ ms $= 0$ ms

Ezek alapján a kitöltési tényező (ha $R_1 = 100\ \Omega$ és $R_2 = 0\ \Omega$)

$$k = \frac{T_{ON}}{T_{ON}+T_{OFF}} = \frac{R1}{R1+2R2} \approx 100\%$$

A 7.11 ábrán látható, hogy a kitöltési tényező csak elméleti esetben 100 %, a valóságos modellen bizony ez csak megközelíti a 100 %-ot. Bizonyos időközönként a kimenet lemegy alacsony (logikai 0) szintre. Az ok, amiért mégis használják, 200 mA-t tud leadni a kimenetén, tehát nagyon jól terhelhető.

Mint látható, a kitöltési tényező értéke nem tud 50 % alá menni, ahhoz egy speciális kapcsolás szükséges. A 7.12 ábrán egy ilyen kapcsolás látható, ahol a P potenciométer segítségével 18 %-ig csökkenthető a kitöltési tényező.



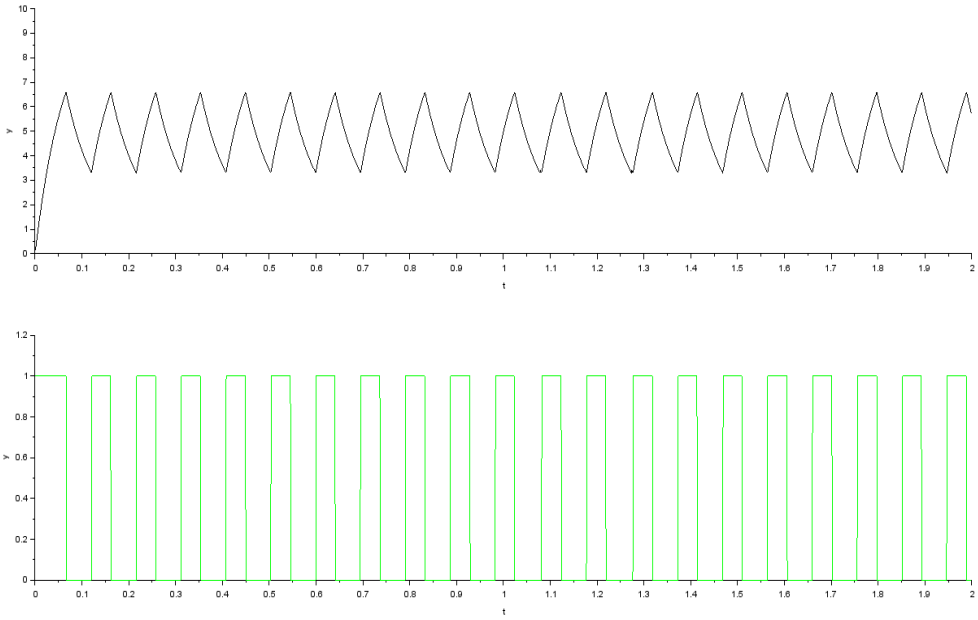
7.12 ábra Speciális kapcsolás

Bemenetre adott válaszfüggvény 50 % kitöltöttséggel:

A következő szimulációban a visszacsatoló hálózat értékei változtak meg a következőkre:

- $R_1 = 0\ \Omega$
- $R_2 = 100\ \Omega$
- $C = 600\ \mu F$

A 7.13 ábra felső részén látható az astabil multivibrátor kondenzátorára eső feszültsége cscope-ban, a 7.13 ábra alsó részén pedig a bemenetre adott válaszfüggvénye (logikai alacsony – magas), abban az esetben, ha az R_1 ellenállás értéke 0.

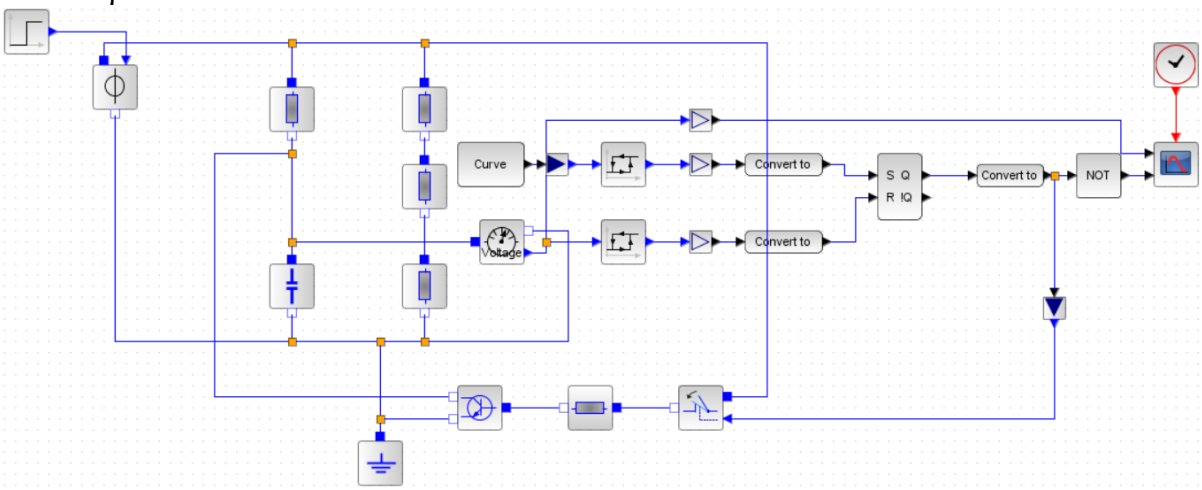


7.13 ábra A szimuláció eredménye

Monostabil multivibrátor

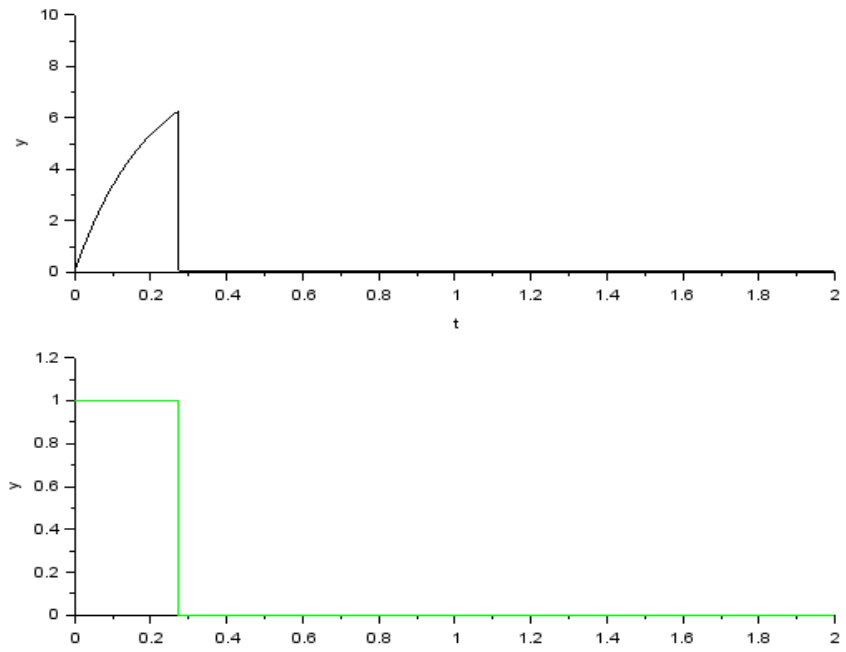
A kapcsolása hasonló az astabil multivibrátoréhoz, de itt már csak R_1 ellenállás és C kondenzátor alkotják a visszacsatoló hálózatot. A megvalósított kapcsolás a 7.14 ábrán látható. A bemeneti impulzust egy egységugrás negáltja váltja ki, amely 0 s-nál érkezik. A visszacsatolás paraméterei:

$R_1 = 330\ \Omega$
$C = 600\ \mu F$



7.14 ábra A megvalósított kapcsolás

A 7.14 ábrán látható az 555 IC idődiagramja, tehát a kimeneti szintek és a kondenzátor töltöttsége.



7.15 ábra Válasz függvény a kondenzátor töltöttsége alapján

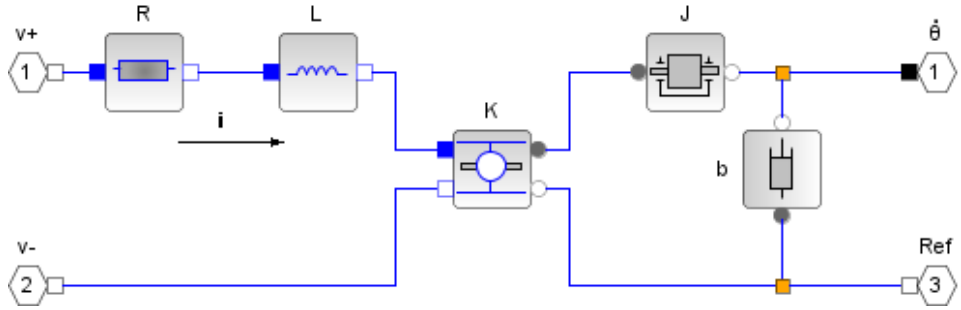
DC motor modellezése

Egyenáramú motornak nevezünk, minden olyan egyenárammal működő gépet, amelyben a gép villamos energiát alakít mechanikai energiává. A motor egy állórészből, és egy tekercselt forgórészből áll, a forgórészre kapcsolt feszültség hatására, a rotort forogni kezd.

(J)	**tehetetlenség**	**0.01 kg.m^2**
(b)	**surlódási együttható**	0.1 N.m.s
(Ke)	**elektromotoros erő konstans**	0.01 V/rad/sec
(Kt)	**motor nyomaték konstans**	0.01 N.m/Amp
(R)	**motor ellenállása**	1 Ohm
(L)	**motor induktivitása**	0.5 H

$$\frac{d^2\theta}{dt^2} = \frac{Ki - b\frac{d\theta}{dt}}{J}$$

$$\frac{di}{dt} = \frac{V - K\frac{d\theta}{dt} - Ri}{L}$$



7.16 ábra Egyenáramú motor modellje

Állapotteres megoldás:

$$\underline{\dot{x}} = A\underline{x} + B\underline{u}$$
$$\underline{y} = C\underline{x} + D\underline{u}$$

n állapotok száma
m bemenetek száma
p kimenetek száma
x=állapotok vektora nx1
u=bemenetek vektora mx1
x'=az állapotvektor deriváltja nx1
A=nxn B=nxm C=pxn D=pxm

$$\underbrace{\begin{bmatrix} \frac{d^2\theta}{dt^2} \\ \frac{di}{dt} \end{bmatrix}}_{\underline{\dot{x}}} = \underbrace{\begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \frac{d\theta}{dt} \\ i \end{bmatrix}}_{\underline{x}} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}}_{B} \underbrace{[V]}_{\underline{u}}$$

$$\underbrace{[y]}_{\underline{y}} = \underbrace{[1 \quad 0]}_{C} \underbrace{\begin{bmatrix} \frac{d\theta}{dt} \\ i \end{bmatrix}}_{\underline{x}} + \underbrace{[0]}_{D} \underbrace{[V]}_{\underline{u}}$$

Átviteli függvény

$$\frac{d^2\theta}{dt^2} = \frac{Ki - b\frac{d\theta}{dt}}{J}$$

$$\frac{di}{dt} = \frac{V - K\frac{d\theta}{dt} - Ri}{L}$$

Laplace transzformáció

$$s^2\Theta = \frac{KI - bs\Theta}{J}$$

$$sI = \frac{V - Ks\Theta - RI}{L}$$

Átrendezve

$$\mathrm{s}(\mathrm{Js} + \mathrm{b})\Theta = KI$$

$$(Ls + R)I = V - Ks\Theta$$

Tovább rendezve

$$(Ls + R)\frac{\mathrm{s}(\mathrm{Js} + \mathrm{b})\Theta}{K} = V - Ks\Theta$$

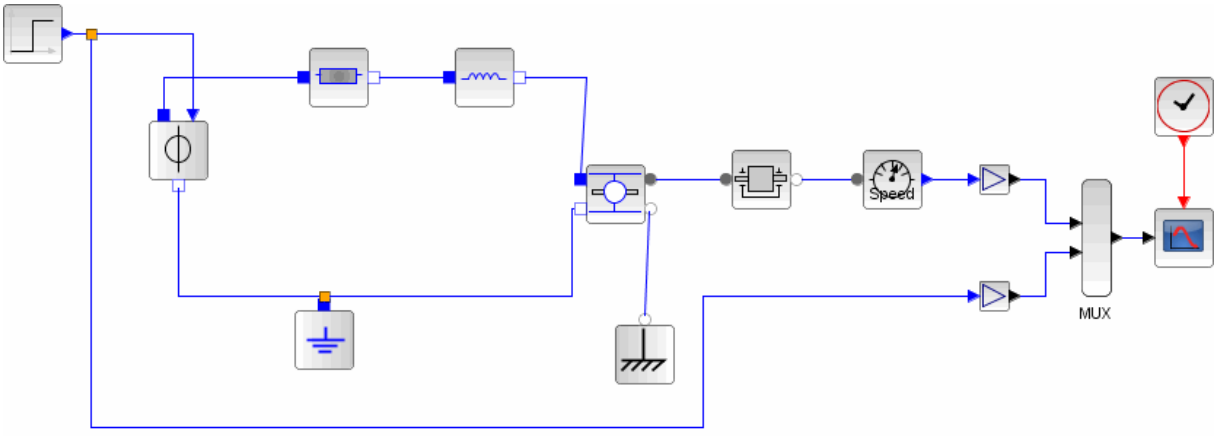
$$\mathrm{s}\Theta = \frac{V}{\left(\frac{(\mathrm{Js} + \mathrm{b})(Ls + R)}{K} + K\right)}$$

$$\mathrm{s}\Theta = \frac{KV}{\left((\mathrm{Js} + \mathrm{b})(Ls + R) + K^2\right)}$$

Végleges átviteli függvény

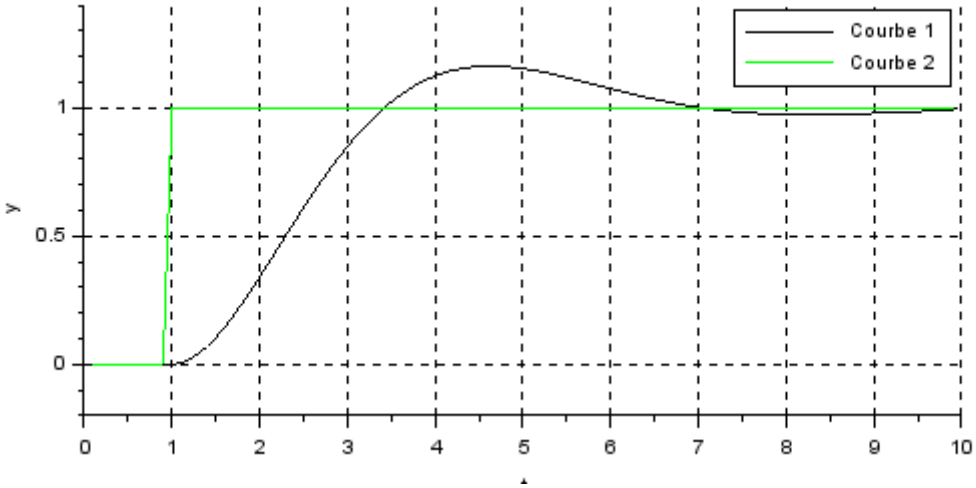
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{JLs^2 + (JR + bL)s + bR + K^2}$$

A DC motrot kétféleképp modellezhetjük, egyrészt a Coselica speciális eszköztára által kínál elemekből, másfelől pedig az állapotteres módszer segítségével kapott egyenletből vezetjük le a motor megvalósítását.



7.17 ábra DC motor modell Coselica megvalósítása

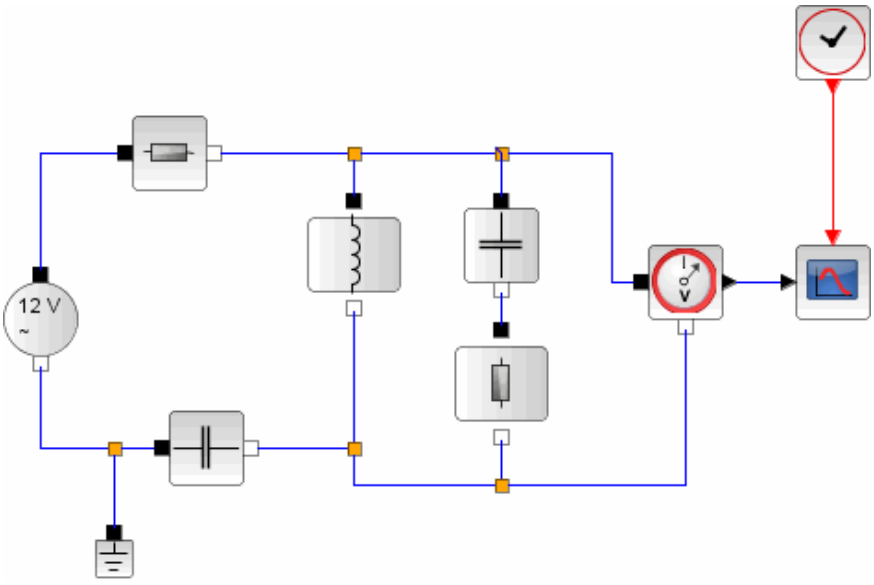
Ahol: R=1, L=1, K=1, J=1



7.18 ábra Szimulációs eredmény

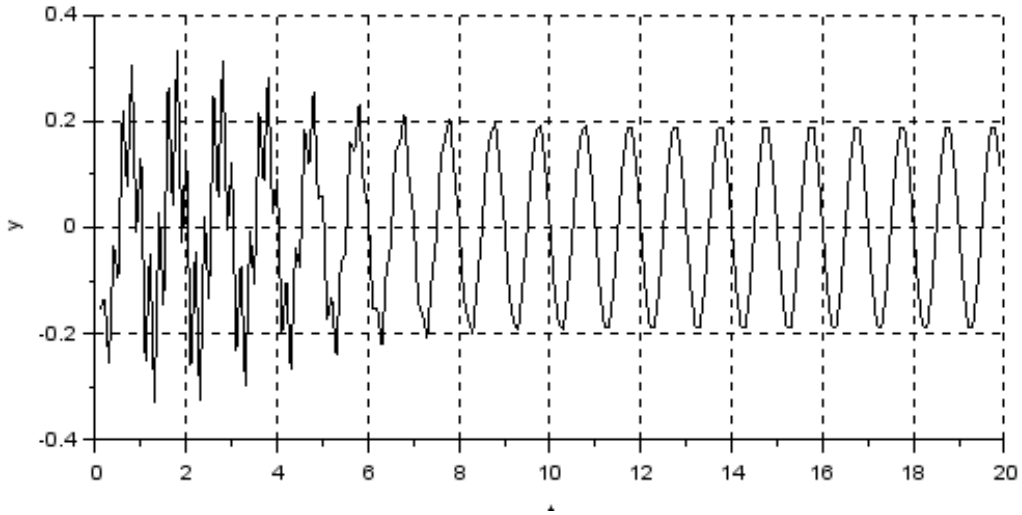
RLC rezgőkör

A soros rezgőkört a rezonancia frekvenciával megegyező frekvencia kiválasztására vagy kiszűrésére használjuk. A frekvencia kiválasztás azt jelenti, hogy az áramkör bemenetére érkező sokféle frekvenciájú jel közül csak egyet használunk fel, vagyis a kimeneten csak egyféle frekvenciájú jel jelenik meg.



7.19 ábra RLC rezgőkör

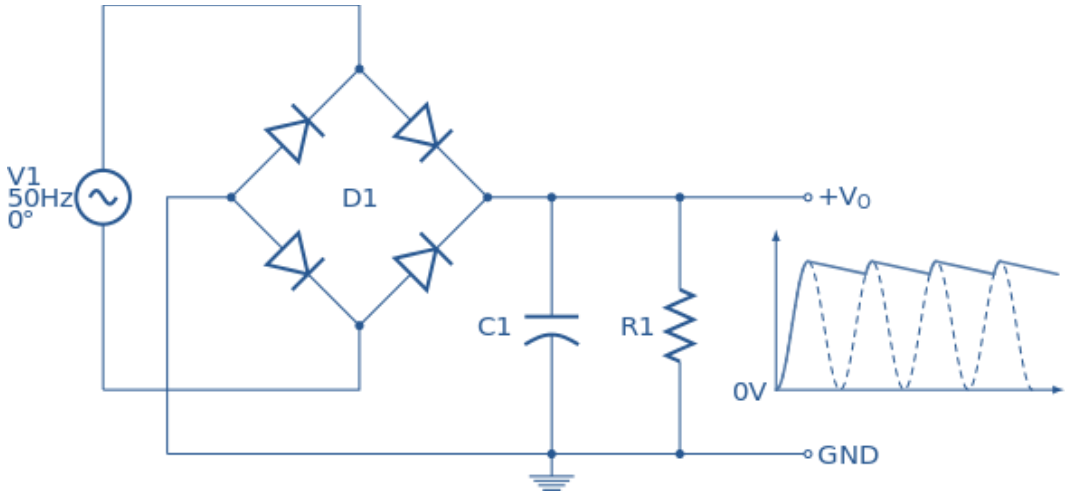
Ahol: R1=2, L=2, C1=2*10^-4, R2=5, C2=3*10^-4



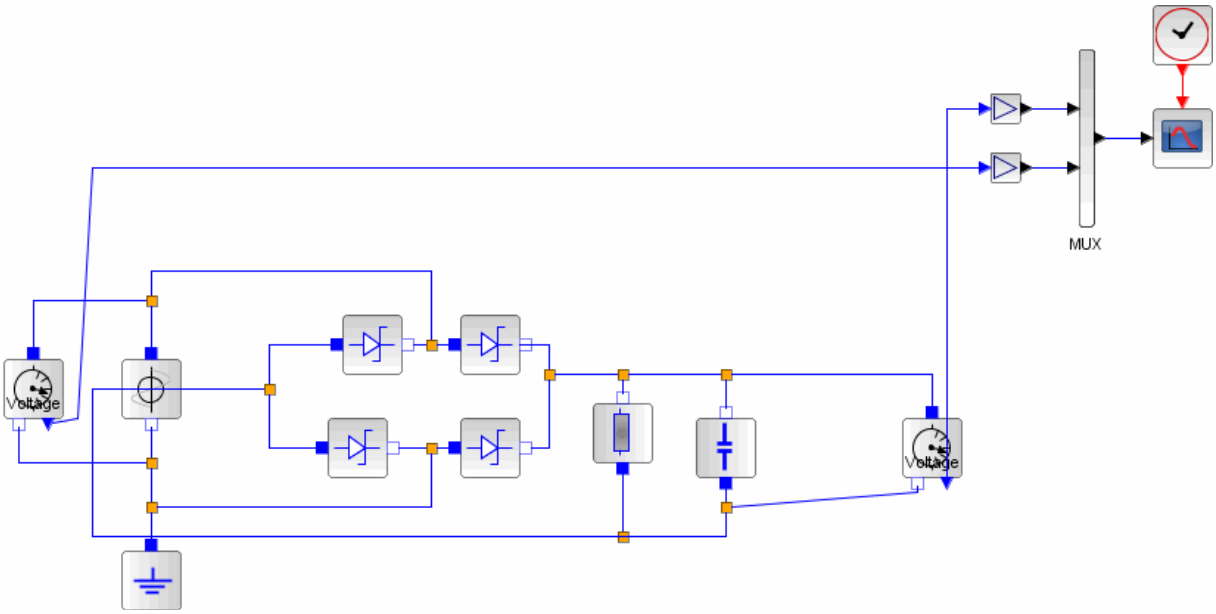
7.20 ábra Szimulációs eredmény

AC/DC átalakító

Az egyenirányító segítségével váltakozó áramot alakítanak át egyenárammá. Alapvetően diódák alkotják, amik újabban szilíciumból készülnek. Lineáris tápegységekben túlnyomórészt négy diódából álló, ún. Graetz-kapcsolású egyenirányítást alkalmaznak. Modellezzük az alábbi kapcsolást Coselica segítségével.

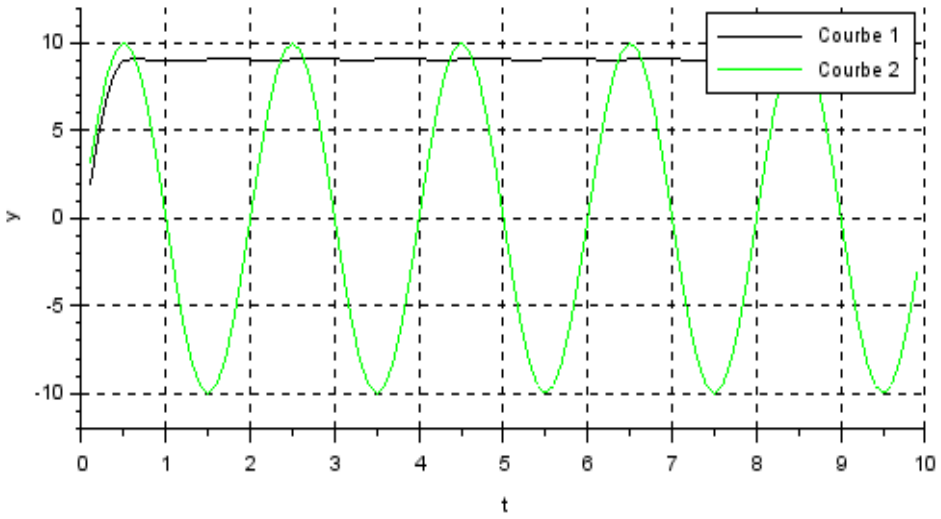


7.21 ábra AC-DC átalakító



7.22 ábra AC-DC átalakító Coselica modellje

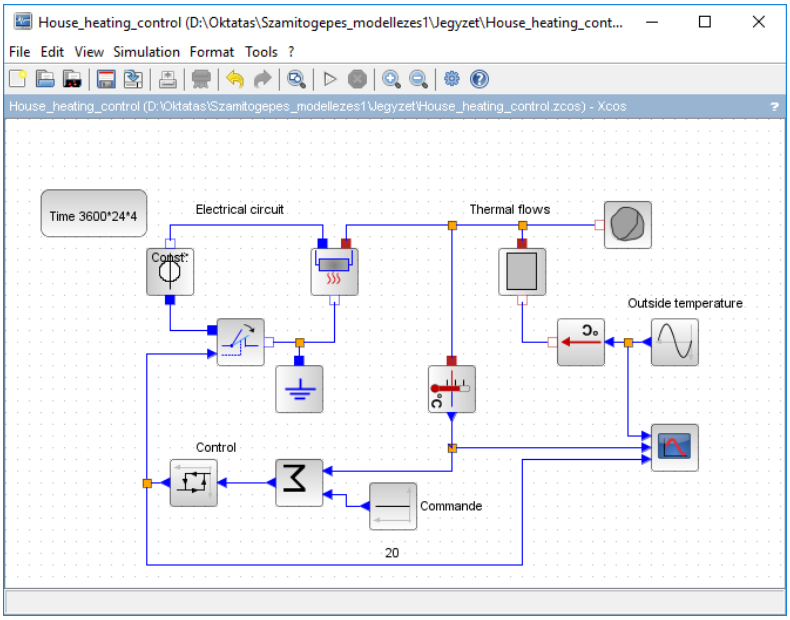
Amint a szimulációs eredményen látszik, a zöld vonal a bemeneten található váltóáramot mutatja, míg a fekete vonal a kimeneten mérhető stabilizált egyenáramot prezentálja.



7.23 ábra Szimulációs eredmény

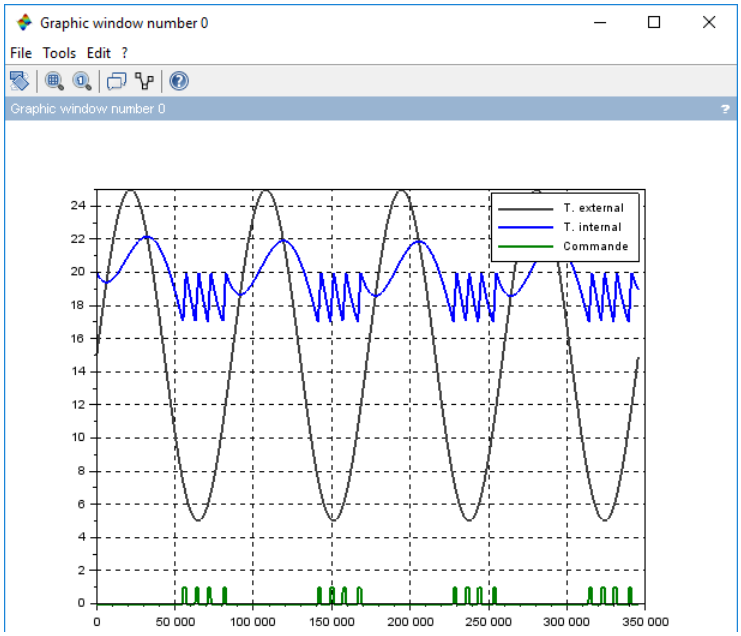
7.2 Termikus szimuláció

A Coselica termikus-szimuláció része egy családi ház fűtésének szabályozásán keresztül kerül bemutatásra. Egy 20 C° referencia érték lesz a szabályzó bemenetén, figyelve a külső hőmérsékletet is. A referencia értéktől való eltérés nem lehet nagyobb 3 C°-nál. Alkalmazva a Coselica blokkjait meg tudjuk valósítani az alábbi modellt.



7.24 ábra Egy családi ház hőmérséklet szabályozása

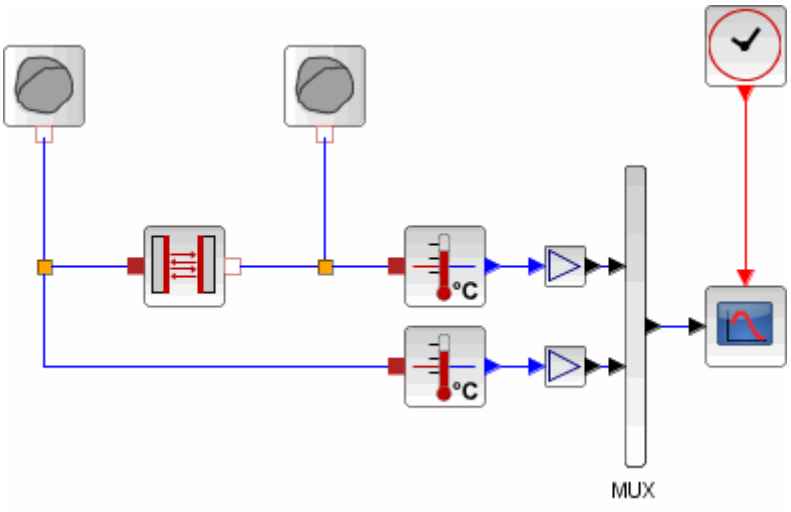
Miután megvalósult a megfelelő modul, a következő ábrát kapjuk.



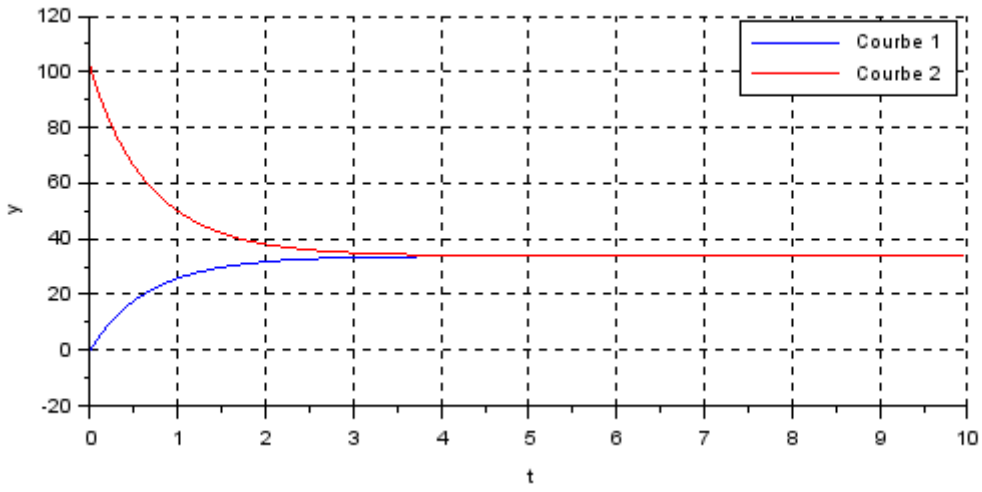
7.25 ábra Szimulációs eredmény

A fenti ábrán látható a külső hőmérséklet változása, a belső hőmérséklet értékeivel, valamint a vezérlőjel aktivitása is.

A következő modell egy egyszerű parametrizálható hőátadást modellez ahol eredményként a két különböző hőmérsékletű közeg kiegyenlítődése szemlélhető.



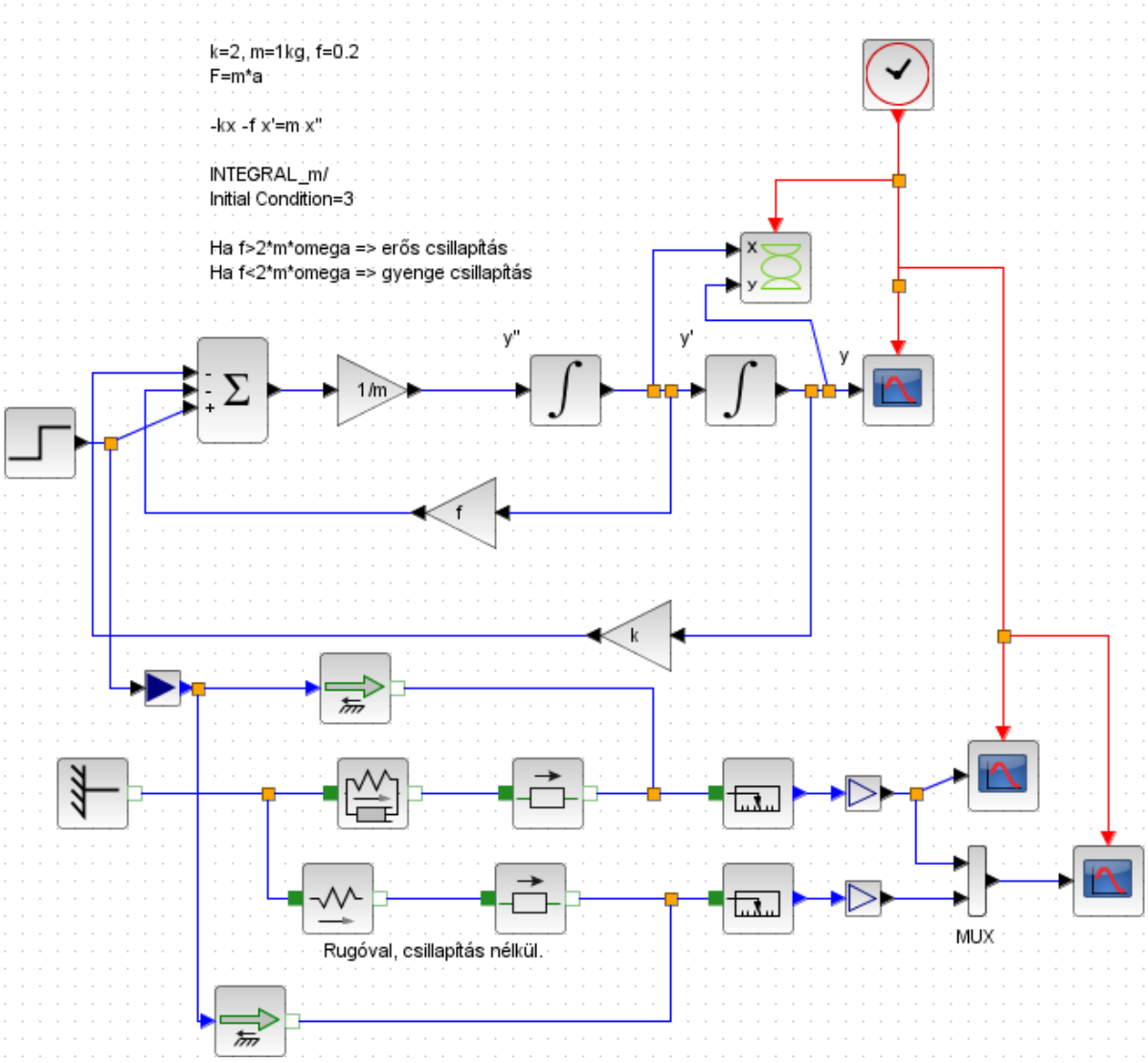
7.26 ábra Két különböző hőmérsékletű közeg



7.27 ábra Szimulációs eredmény

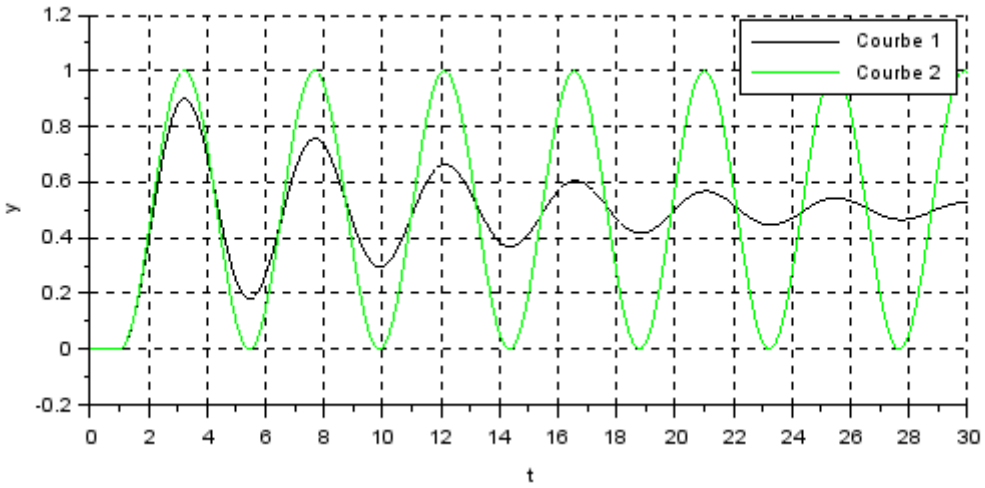
7.3 Gépészeti szimuláció

A Coselica gépészeti-szimuláció része egy csillapított rezgés modellen keresztül kerül bemutatásra.



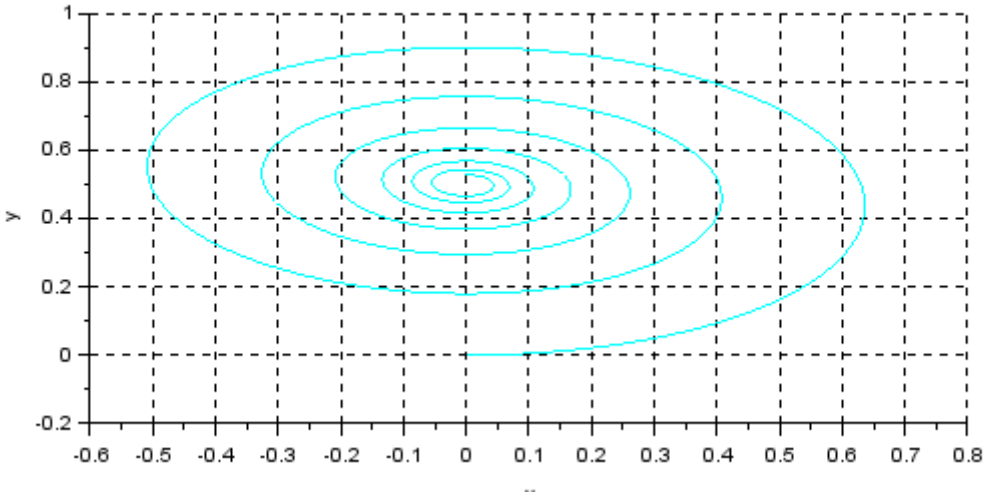
7.28 ábra Szimulációs eredmény

A szimulációs modell tartalmazza az alap XCOS elemekből megszerkesztett modellt, valamint a Coselica elemeit alkalmazva ugyanazon modell eredményeit tudjuk megvizsgálni csillapítással és csillapítás nélkül.



7.29 ábra Szimulációs eredmény

Az XY gráf eredménye az alábbi ábrán látható.



7.30 ábra Szimulációs eredmény

Ellenőrző kérdések:

1. Mi a Coselica?
2. Milyen modelleket készithetünk Coselica segítségével?
3. Vázolja fel az NE555 shematikus rajzát és ismertesse a működési elvét.
4. Ismertesse az RLC rezgőáramkör elméleti hátterét?
5. Hogyan működik az egyenirányító?

8 ROBOTICS TOOLBOX

A Robotics Toolbox sok olyan függvényt szolgátat, amelyek hasznosak a robotikában, beleértve olyan témákat, mint a kinematika, dinamika és a pályagenerálás. A Toolbox éppolyan hasznos szimuláláshoz, mint valódi robotokkal való kísérletezés eredményeinek elemzéséhez. A Toolbox az utóbbi években olyan szintre fejlődött és kibővült, hogy szükségtelenné teszi a „C" nyelvben írt saját modulok/függvények fejlesztését.

A Toolbox az egymás után csatolt manipulátorok kinematikai és dinamikai reprezentálásának egy nagyon általános módszerén alapul. A paraméterek SCILAB objektumokba vannak tárolva. A felhasználó bármely kinematikai lánchoz képes Robot objektumokat létrehozni és számos példa is elérhető a közismert robotok közül, mint például a Puma 560 és a Standford arm. A Toolbox tehát függvényekkel rendelkezik több adattípus kezelésére is: vektorok, homogén transzformációk, unit quaternion-ok, melyek szükségesek a háromdimenziós pozíció és orientáció reprezentálásához.

A Robotics Toolbox ingyenes, és elérhető az ATOMS rendszeren keresztül. Ajánlott az összes fájlt ebből a könyvtárból letölteni, mivel a Toolbox funkció gyakran egymásra támaszkodnak, és összefüggő rendszert alkotnak [10]. A Robotics Toolbox melléklete a *robot.pdf*, amely egy átfogó kézikönyv egy bevezető ismertetővel és az összes Toolbox funkció részleteivel. A rtdem funkcióval egy menüvezérlet alkalmazást hívhatunk meg, amely a toolbox lehetőségeit mutatja be.

Minden szög radiánban értendő. A felhasználó választhat más mértékegységeket is, és ez nyilván befolyásolja a homogénkoordináták, Jakobi-mátrixok, tehetetlenségi- és forgatónyomatékok mértékegységeit is.

Homogén transzformációk	
eul2tr	Euler-szögből homogén transzformációt
oa2tr	orientációból és közelítő vektorból homogén transzformációt
rot2tr	3x3-as rotációs mátrixból homogén transzformációt
rotx	X-tengely körüli rotáció homogén transzformációja
roty	Y-tengely körüli rotáció homogén transzformációja
rotz	Z-tengely körüli rotáció homogén transzformációja
rpy2tr	billentési/csavarási/forgatási szögekből homogén transzformációt
tr2eul	homogén transzformációból Euler-szöget
tr2r0t	homogén transzformációból rotációs almátrixot
tr2rpy	homogén transzformációból billentési/csavarási/forgatási szögeket
transl	beállítja vagy eltávolítja a transzlációs komponensét egy homogén transzformációnak
trnorm	egy homogén transzformáció normalizál

Speciális komplex számok

/	komplex szám osztása komplex számmal vagy skalárral
*	komplex szám szorzása komplex számmal vagy vektorral
inv	komplex szám inverz
norm	komplex szám normája
plot	komplex szám megjelenítése 3D rotációként
q2tr	komplex szám homogén transzformációt
qinterp	komplex szám komplex számokat
unit	egységesít egy komplex számot

Kinematika	
diff2tr	differenciális mozgás-vektorból transzformációt
fkine	direkt kinematikát számit
ikine	inverz kinematikát számit
ikine560	inverz kinematikát számit puma 560 karjaira
jacob0	világkoordináta rendszerben Jacobi mátrixot számit
jacobn	effektor koordináta rendszerben Jacobi mátrixot számit
tr2diff	homogén transzformációból differenciális mozgás-vektorokat
tr2jac	homogén transzformációból Jacobi mátrixba

Dinamika	
accel	direkt dinamikát számit
cintertia	a manipulátor Descartes féle tehetetlenségi mátrixot számítja
coriolis	centripetális/ coriolis nyomatékot számit
friction	a csukló súrlódása
ftrans	erő/nyomaték transzformáció
gravload	gravitációs hatásokat számítja
inertia	manipulátor tehetetlenségi mátrixot számítja
itorque	tehetetlenségi nyomatékot számit
nofriction	kiveszi a súrlódást egy robot objektumból
rne	inverz dinamika

Manipulátor modellek	
link	létrehoz egy robot-szegmens objektumot
puma560	puma 560 adatai
puma560akb	puma 560 adatai módosított
robot	létrehoz egy robot objektumot
standorf	a Standorf kar adatai
twolink	példa két egyszerű szegmensre

Pályagenerálás	
crtaj	Descartes-féle pálya
jtraj	pályaszámítás a csuklókoordináták terében
trinterp	interpolál homogén transzformációkat

Grafika	
drivebot	Manuálisan vezérli a grafikusan megjelenített robotkonfigurációt
plot	megjelenít/animálja a robotot

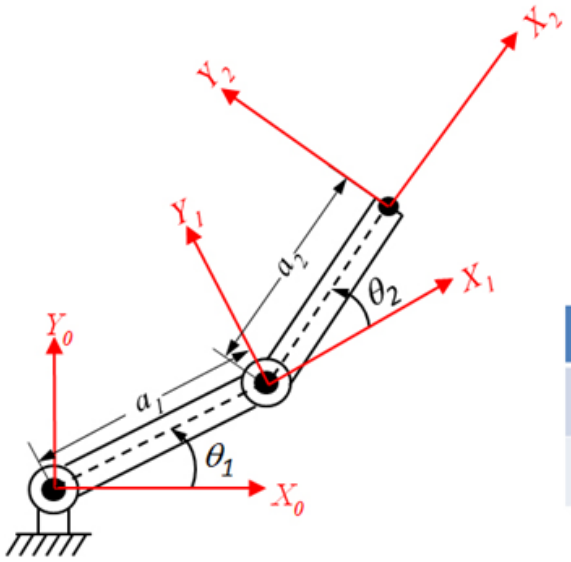
Egyéb	
manipblty	kiszámítja a manipulálhatóságot
rtdemo	a Toolbox demonstrációja
unit	egységesít egy vektort

8.1 Robotkarok tervezése

A következő néhány feladat ismerteti a Robotics toolbox lehetőségeit a robotkarok tervezésének terén.

Robotics toolbox feladat 1 (síkbeli manipulátor)

Modellezni az alábbi ábrán látható két szegmenses síkbeli manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon.

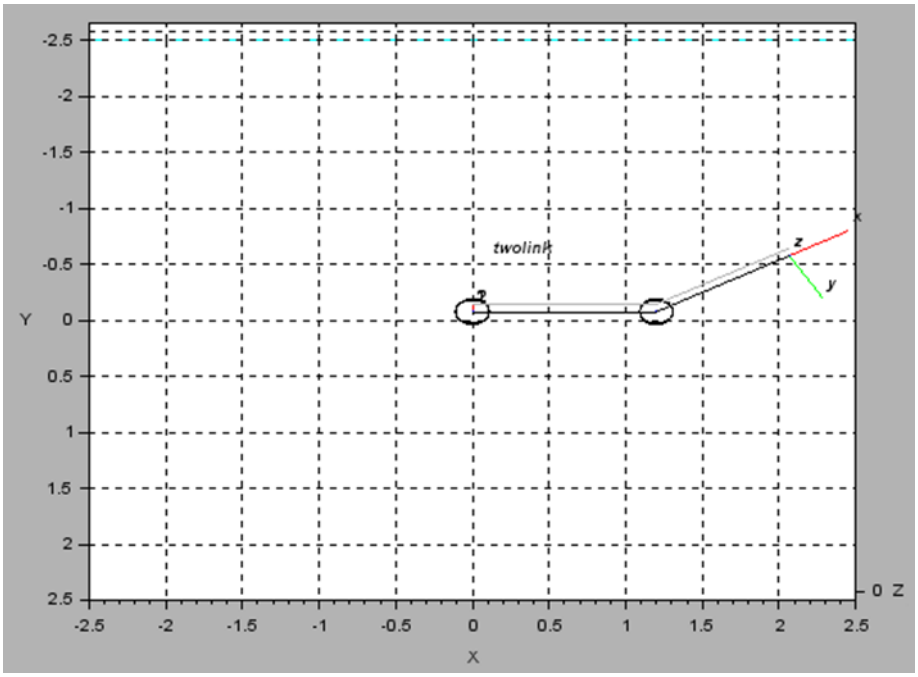


link	θ_i	d_i	a_i	α_i
1	θ_1^*	0	a_1	0
2	θ_2^*	0	a_2	0

8.1 ábra Síkbeli manipulátor

Megoldás:

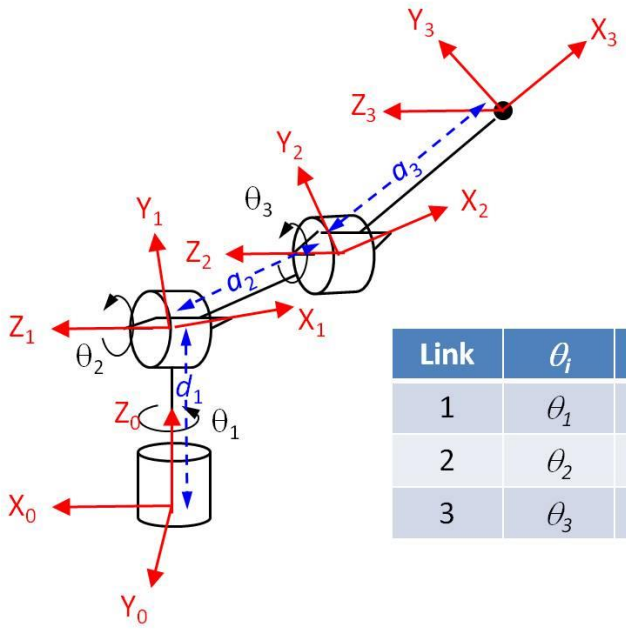
```
clear L;
a1 = 1.2;
a2 = 1;
L1=Link([0 0 a1 0]);
L2=Link([0 0 a2 0]); // theta d a alpha
L=list(L1,L2);
twolink=SerialLink(L);
twolink.name='twolink';
plot_robot(twolink, [0 -%pi/6]);
```



8.2 ábra Szimulációs eredmény

Robotics toolbox feladat 2 (RRR alapkonfiguráció)

Modellezni az alábbi ábrán látható RRR típusú alapkonfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!



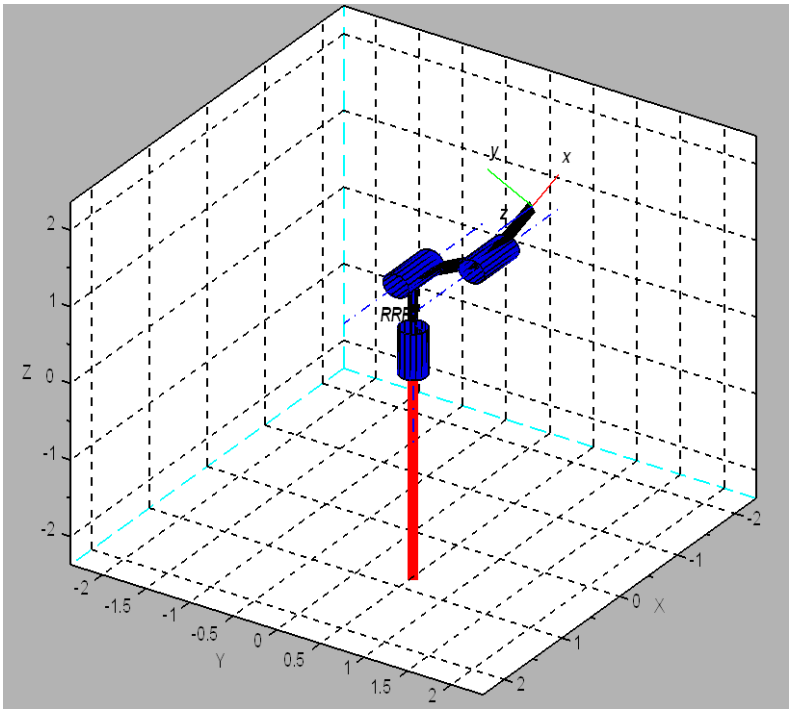
Link	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	90°
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

8.3 ábra RRR típusú alapkonfiguráció

Megoldás:

```
clear L;
d1 = 1;
```

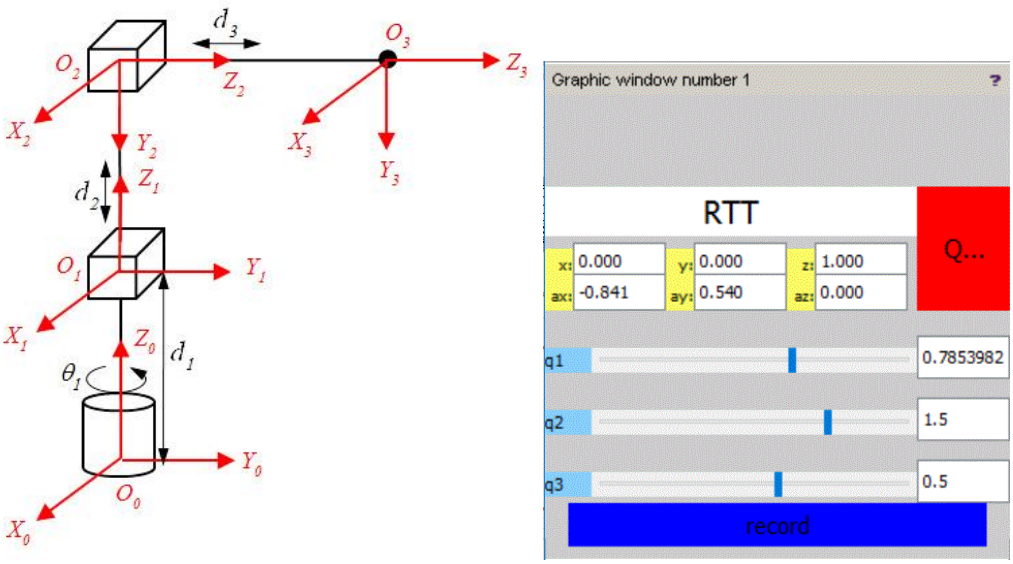
```
a2 = 1;
a3 = 1;
L1= Link([0 d1 0 %pi/2]);
L2=Link([0 0 a2 0]);
L3=Link([0 0 a3 0]); // theta d a alpha
L=list(L1,L2,L3);
rrr_robot = SerialLink(L);
//Robot_Info(rrr_robot);
rrr_robot.name='RRR';
plot_robot(rrr_robot,[%pi/2, %pi/6, %pi/6]);
```



8.4 ábra Szimulációs eredmény

Robotics toolbox feladat 3 (RTT alapkonfiguráció)

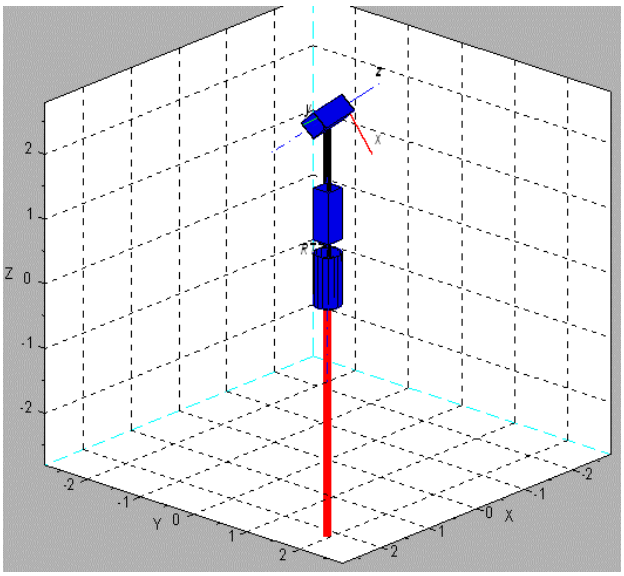
Modellezni az alábbi ábrán látható RTT típusú alapkonfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!



8.5 ábra RRR típusú alapkonfiguráció

Megoldás:

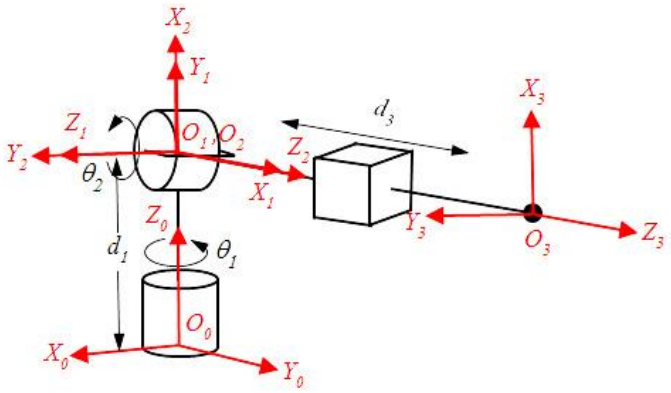
```
clear;
d1 = 1;
d2 = 2; // d1, d2, d3 are joint variables
d3 = 1;
L1=Link([0 d1 0 0]); // theta d a alpha
L2=Link([1 d2 0 -%pi/2 1]); // '1' prismatic joint
L3=Link([1 d3 0 0 1]);
L=list(L1,L2,L3);
rtt_robot=SerialLink(L);
rtt_robot.name='RTT';
plot_robot(rtt_robot,[%pi/4 1.5 0.5]);
teach(rtt_robot)
```



8.6 ábra Szimulációs eredmény

Robotics toolbox feladat 4(RRT alapkonfiguráció)

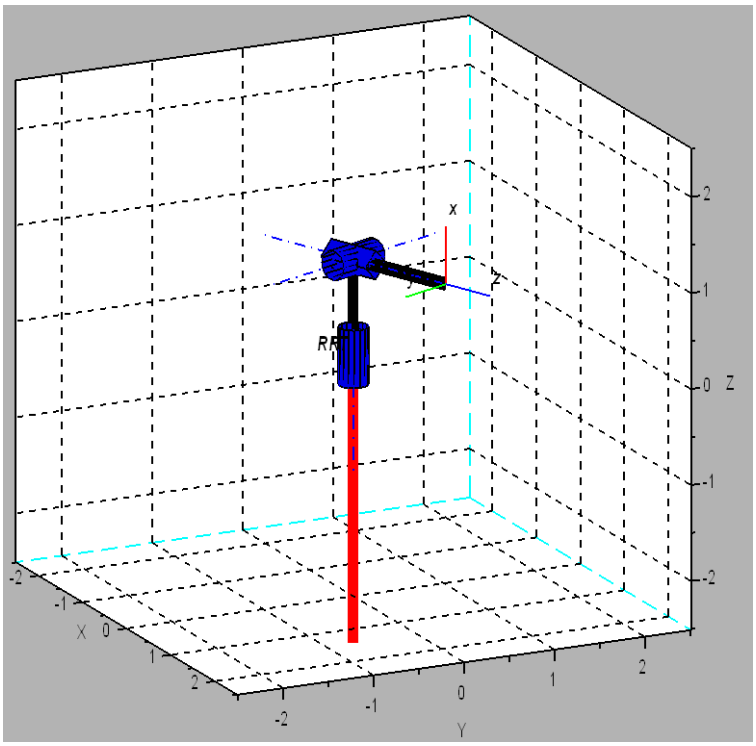
Modellezni az alábbi ábrán látható RRT típusú alapkonfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!



8.7 ábra RRR típusú alapkonfiguráció

Megoldás:

```
clear L;
d1 = 1; // d1 and
d3 = 2; // d3 is joint variable
L1=Link([0 d1 0 %pi/2]);
L2=Link([0 0 0 %pi/2]);
L3=Link([0 d3 0 0 1]);
L=list(L1,L2,L3);
rrt_robot=SerialLink(L);
rrt_robot.name='RRT';
plot_robot(rrt_robot,[1 %pi/2 1.3]);
teach(rrt_robot)
```



8.8 ábra Szimulációs eredmény

Robotics toolbox feladat 5 (SCARA robotkonfiguráció)

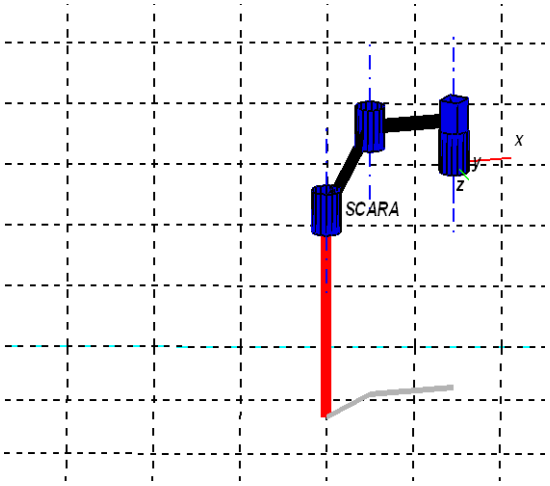
Modellezni az alábbi ábrán látható SCARA típusú robot konfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!

8.9 ábra SCARA típusú robot konfiguráció

Megoldás:

```
clear L;
d1 = 1;
a1 = 1;
a2 = 1;
d3 = 0.2; // joint variable
```

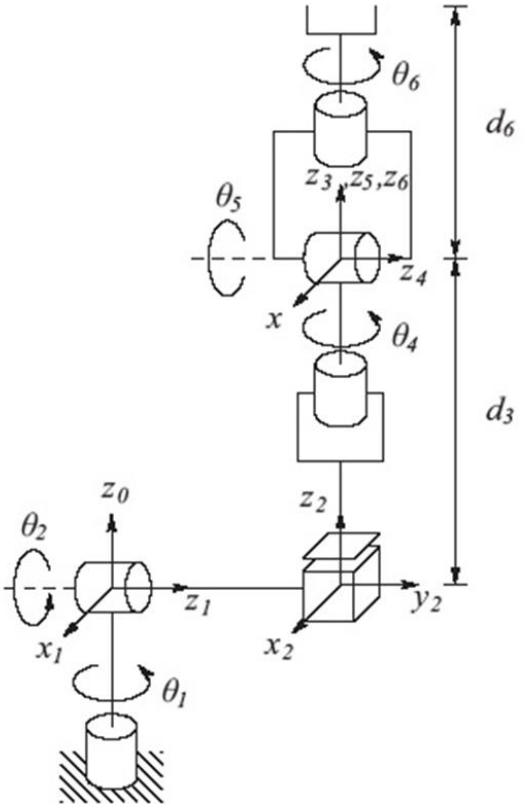
```
d4 = 0.2;
L1=Link([0 d1 a1 0]);
L2=Link([0 0 a2 %pi]);
L3=Link([0 d3 0 0 1]);
L4=Link([0 d4 0 0]);
L=list(L1,L2,L3,L4);
scara_robot=SerialLink(L);
scara_robot.name='SCARA';
plot_robot(scara_robot,[%pi/3 -%pi/4 0.5 0]);
teach(scara_robot)
```



8.10 ábra Szimulációs eredmény

Robotics toolbox feladat 6 (Stanford arm robotkonfiguráció)

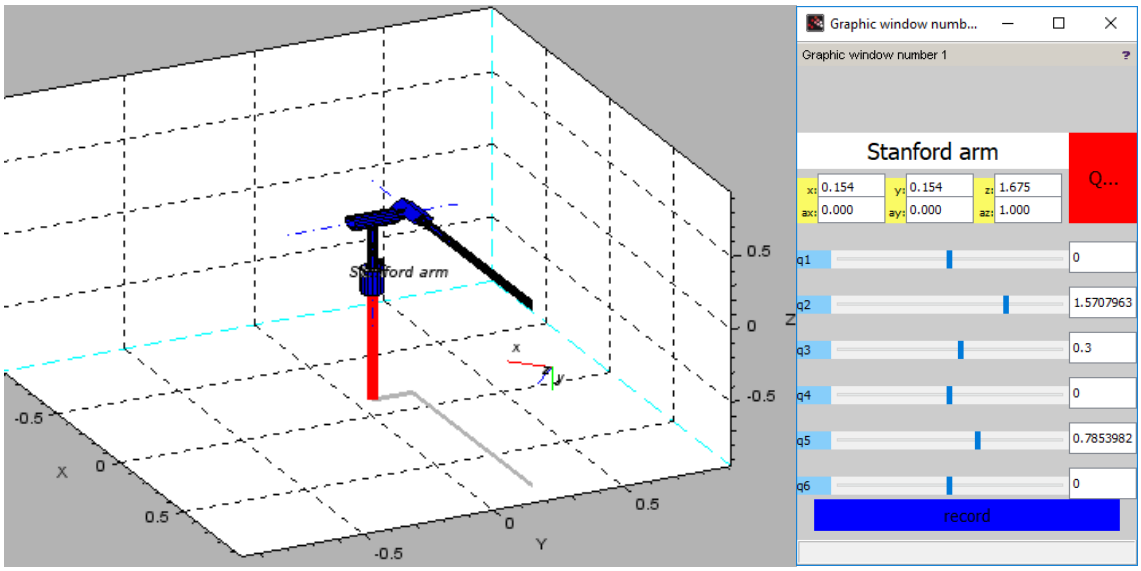
Modellezni az alábbi ábrán látható Stanford arm típusú robot konfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!



8.11 ábra Stanford arm típusú robot konfiguráció

Megoldás:

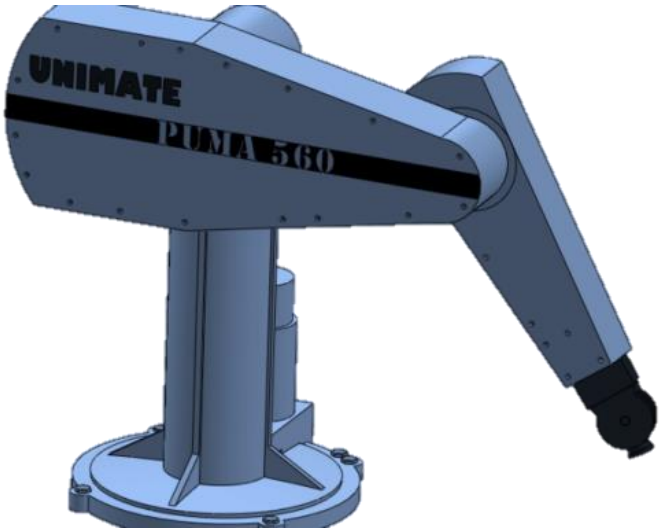
```
L1 = Link([ 0 0.412 0 -%pi/2 0]);
L2 = Link([0 0.154 0 %pi/2 0]);
L3 = Link([-%pi/2 0 0 0 1 1]);
L4 = Link([0 0 0 -%pi/2 0]);
L5 = Link([0 0 0 %pi/2 0]);
L6 = Link([0 0.263 0 0 0]);
L=list(L1,L2,L3,L4,L5,L6);
stanford= SerialLink(L);
stanford.name='Stanford arm';
plot_robot(stanford,[0 %pi/2 0.3 0 %pi/4 0]);
teach(stanford)
```



8.12 ábra Szimulációs eredmény

Robotics toolbox feladat 7 (PUMA robotkonfiguráció)

Modellezze az alábbi ábrán látható PUMA típusú robot konfigurációt a SCILAB Robotics Toolbox segítségével a képen látható módon!

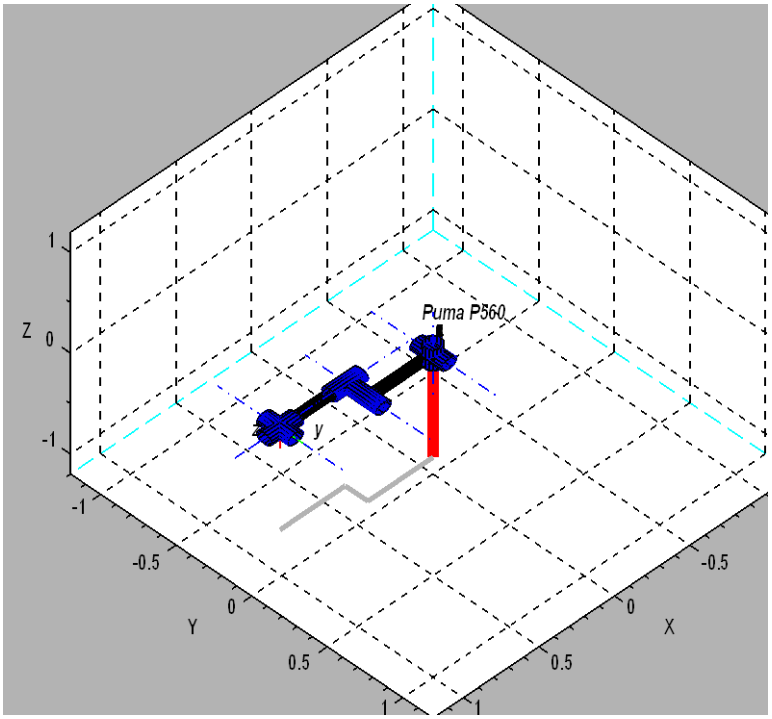


8.13 ábra PUMA típusú robot konfiguráció

Megoldás:

```
// th d a alpha
L1 = Link([ 0 0 0 %pi/2 0]);
L2 = Link([ 0 0 0.4318 0 0]);
L3 = Link([ 0 0.15005 0.0203 -%pi/2 0]);
L4 = Link([ 0 0.4318 0 %pi/2 0]);
L5 = Link([ 0 0 0 -%pi/2 0]);
L6 = Link([ 0 0 0 0 0]);
```

```
L=list(L1,L2,L3,L4,L5,L6);
p560 = SerialLink(L);
p560.name='Puma P560';

q_z = [0 0 0 0 0 0]; // zero angles, L shaped pose
q_r = [0 %pi/2 -%pi/2 0 0 0]; // ready pose, arm up
q_s = [0 0 -%pi/2 0 0 0];
q_n=[0 %pi/4 %pi 0 %pi/4 0];
plot_robot(p560,q_s);
teach(p560)
```



8.14 ábra Szimulációs eredmény

Ellenőrző kérdések:

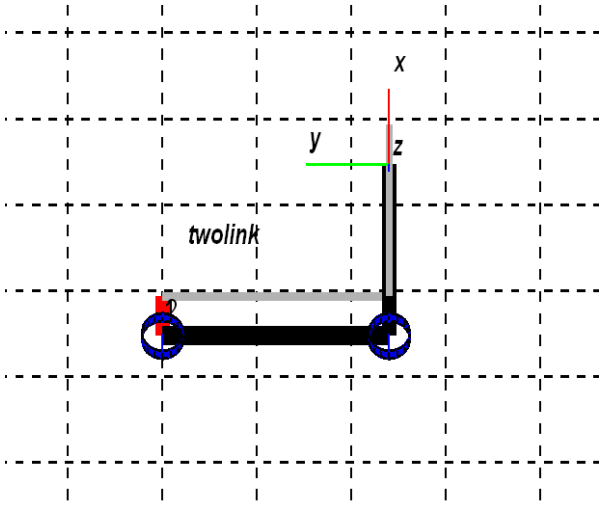
1. Milyen paramétekek definiálnak egy bizonyos szegmenset?
2. Sorolja fel a csuklókat tipus szerint.
3. Hogyan definiálunk egy robot objektumot?
4. Melyik parancs jeleniti meg a robot vezérlő paneljét?
5. Hogyan jelenítjük meg az adott robot konfigurációt?

9 MOZGÁSTERVEK KÉSZÍTÉSE

A következő néhány feladat ismerteti a Robotics toolbox lehetőségeit a mozgástervek készítése terén.

Robotics toolbox feladat 8

Modellezzen egy síkbeli, két szegmenses robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy a minkét csukló teljes kört írjon le!



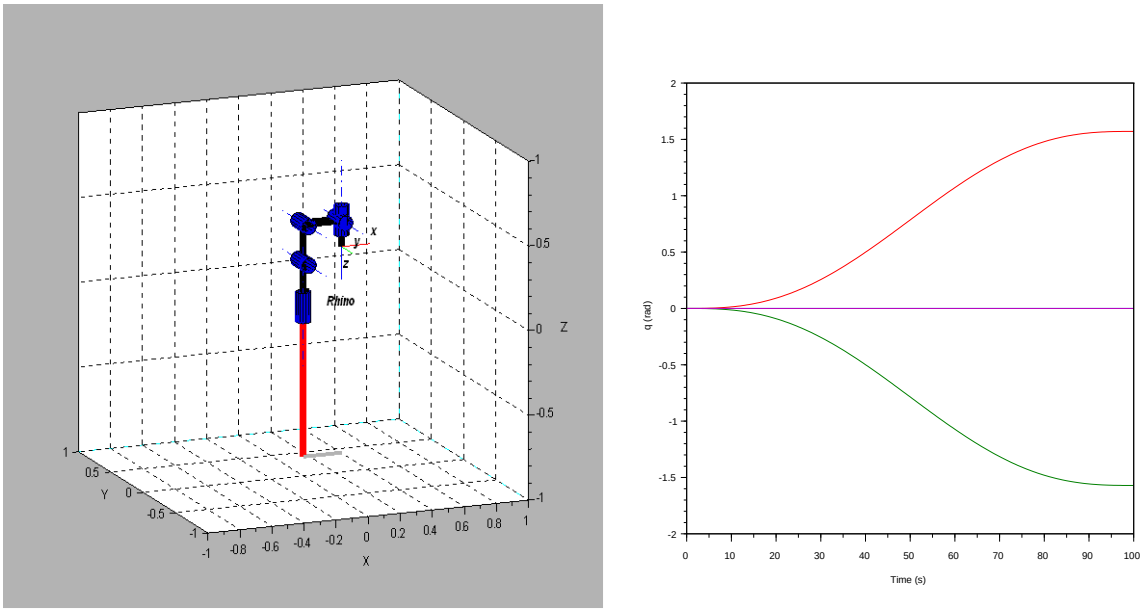
8.15 ábra Szimulációs eredmény

Megoldás:

```
clear L;
a1 = 1.2;
a2 = 1;
L1=Link([0 0 a1 0]);
L2=Link([0 0 a2 0]);
L=list(L1,L2);
twolink=SerialLink(L);
twolink.name='twolink';
// a 2-link manipulator
// generate simple setpoints
// both joints move full circle
t = [0:0.01:1]';
// "time" data
qs = [%pi*t %pi/2*t];
for q=qs'
plot_robot(twolink,q');
end
```

Robotics toolbox feladat 9

Modellezze a RHINO robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy a manipulátor belső szögei kövessék a diagramon ábrázolt értékeket!



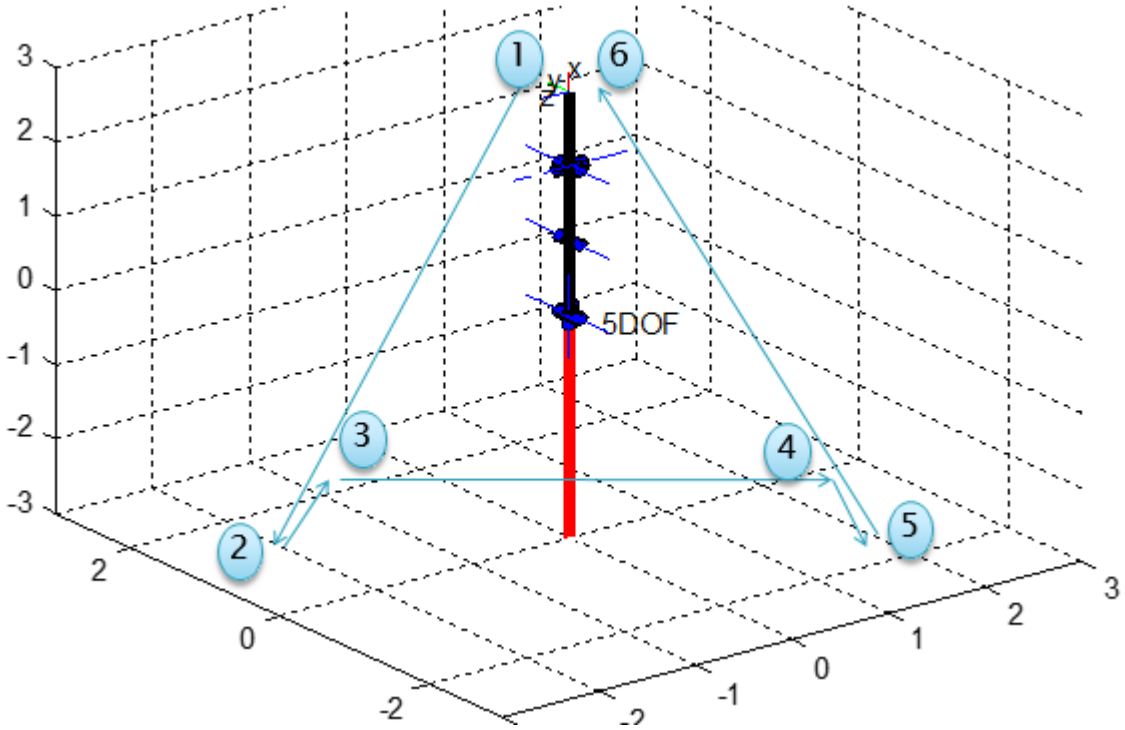
8.16 ábra Szimulációs eredmény

Megoldás:

```
clear all
// th d a alpha
L1 = Link([0 0.26035 0 -%pi/2]);
L2 = Link([-%pi/2 0 0.2286 0]);
L3 = Link([%pi/2 0 0.2286 0]);
L4 = Link([0 0 0.009525 -%pi/2]);
L5 = Link([0 0.15875 0 0 ]);
L=list(L1,L2,L3,L4,L5);
Rhino = SerialLink(L, 'Rhino');
Rhino.name='Rhino';
q0 = [0 0 0 0 0];
qinit = [0 -%pi/2 %pi/2 0 0 ];
t = [0:0.5:100];
q = jtraj(q0, qinit, t);
plot(t,q)
xlabel('Time (s)')
ylabel('q (rad)')
for qq=q'
plot_robot(Rhino,qq')
end
```

Robotics toolbox feladat 10

Modellezze az öt szabadságfokú (5DOF) robot konfigurációt az RTSX (Robotic Tools for Scilab/Xcos) segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy a manipulátor érintse a képen megjelölt pontokat!



8.17 ábra Szimulációs eredmény

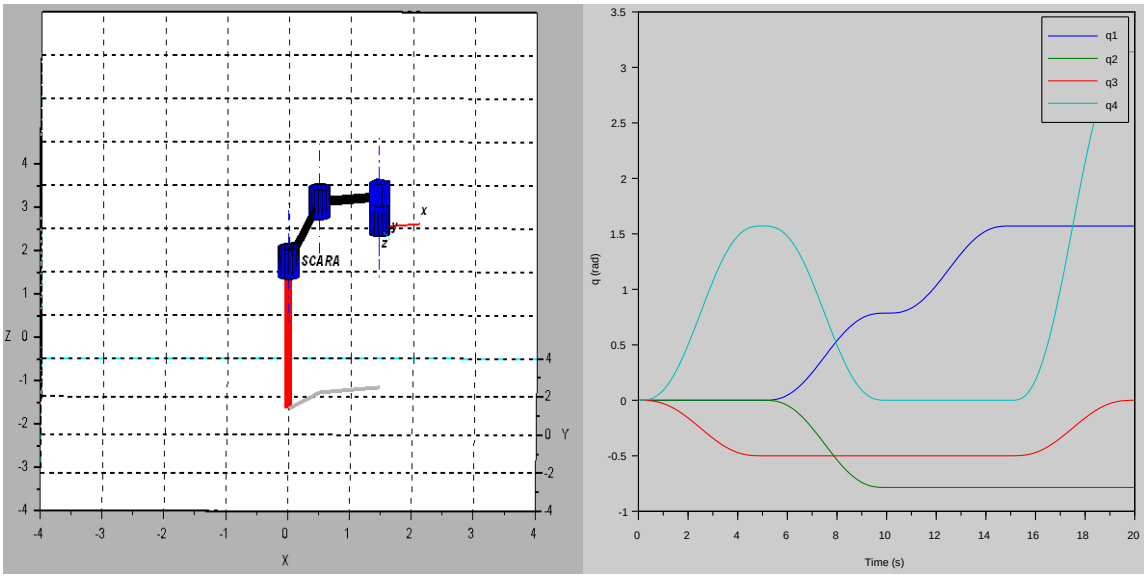
Megoldás:

```
clear all
L1=Link([0 0 0 %pi/2]);
L2=Link([0 0 2 0]);
L3=Link([0 0 2 0]);
L4=Link([0 0 0 %pi/2]);
L5=Link([0 0 2 0])
L=list(L1,L2,L3,L4,L5);
robot_5dof=SerialLink(L)
robot_5dof.name='5DOF';

q1=[0 %pi/2 0 0 0];
q2=[0 %pi/12 -%pi/3 -%pi/8 0 ];
q3=[0 0.66 -1.445 -0.377 0 ];
q4=[%pi/2 0.66 -1.445 -0.377 0 ];
q5=[%pi/2 %pi/12 -%pi/3 -%pi/8 0 ];
q6=[%pi/2 %pi/2 0 0 0]

//plot_robot(robot_5dof, [0 %pi/2 0 0 0])
//teach(robot_5dof)
t=[0:2:100];
while(1)
```

```
//1
q = jtraj(q1, q2, t);
    for qq=q'
        plot_robot(robot_5dof, qq')
    end
//2
q = jtraj(q2, q3, t);
    for qq=q'
        plot_robot(robot_5dof, qq')
    end
//3
q = jtraj(q3, q4, t);
    for qq=q'
        plot_robot(robot_5dof, qq')
    end
//4
q = jtraj(q4, q5, t);
    for qq=q'
        plot_robot(robot_5dof, qq')
    end
//5
q = jtraj(q5, q6, t);
    for qq=q'
        plot_robot(robot_5dof, qq')
    end
//6
q = jtraj(q6, q1, t);
   for qq=q'
        plot_robot(robot_5dof, qq')
    end
end
```

Robotics toolbox feladat 11

Modellezze a SCARA robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy a manipulátor belső szögei kövessék a diagramon ábrázolt értékeket!



8.18 ábra Szimulációs eredmény

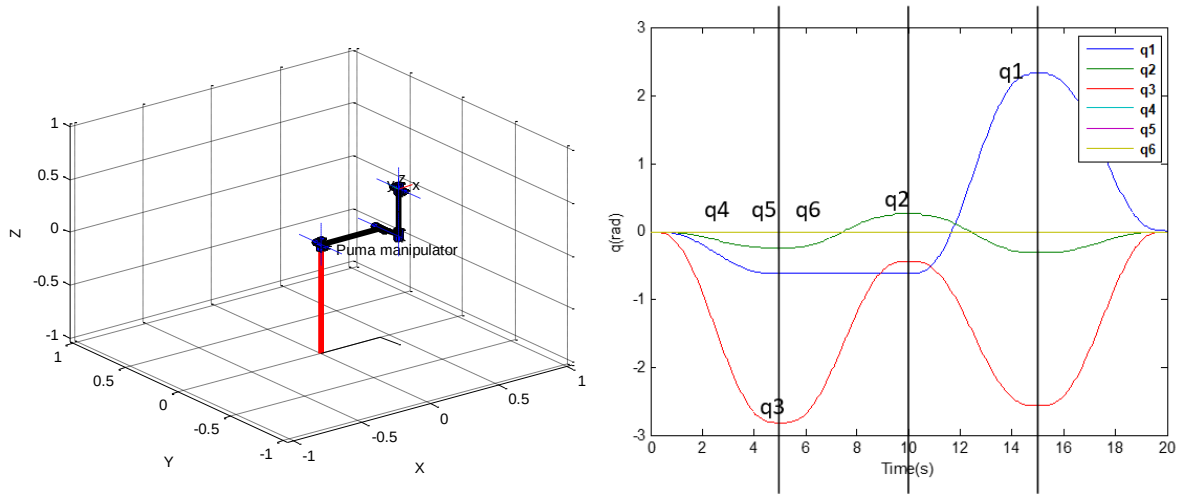
Megoldás:

```
clear L;
d1 = 1;
a1 = 1;
a2 = 1;
d3 = 0.5; // joint variable
d4 = 0;
L1=Link([0 d1 a1 0]);
L2=Link([0 0 a2 %pi]);
L3=Link([0 d3 0 0 1]);
L4=Link([0 d4 0 0]);
L=list(L1,L2,L3,L4);
scara_robot=SerialLink(L);
scara_robot.name='SCARA';
// plotrobot(scara_robot,[pi/3 -pi/4 0.5 0]);
//idosorok definiálása
tab = [0:0.1:5]; t= [tab 5+tab 10+tab 15+tab];
//Koordináták generalasa
aq = jtraj([0 0 0 0] ,[0 0 -0.5 %pi/2], tab);
bq = jtraj([0 0 -0.5 %pi/2],[%pi/4 -%pi/4 -0.5 0], tab);
cq = jtraj([%pi/4 -%pi/4 -0.5 0],[%pi/2 -%pi/4 -0.5 0], tab);
dq = jtraj([%pi/2 -%pi/4 -0.5 0],[%pi/2 -%pi/4 0 %pi], tab);
// a négy muvelet álapotsorozatainak
// konkatenálása (sorozatba szedese)
// egy állapotsorozattá:
q = [aq; bq; cq; dq];
// Plot
figure(1);
plot(t,q);
xlabel('Time (s)');
ylabel('q (rad)');
legend({'q1','q2','q3','q4'},'FontSize',8,'FontWeight','bold')
```

```
for qq=q'
plot_robot(scara_robot,qq')
end
```

Robotics toolbox feladat 12

Modellezze a PUMA robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy a manipulátor belső szögei kövessék a diagramon ábrázolt értékeket!



8.19 ábra Szimulációs eredmény

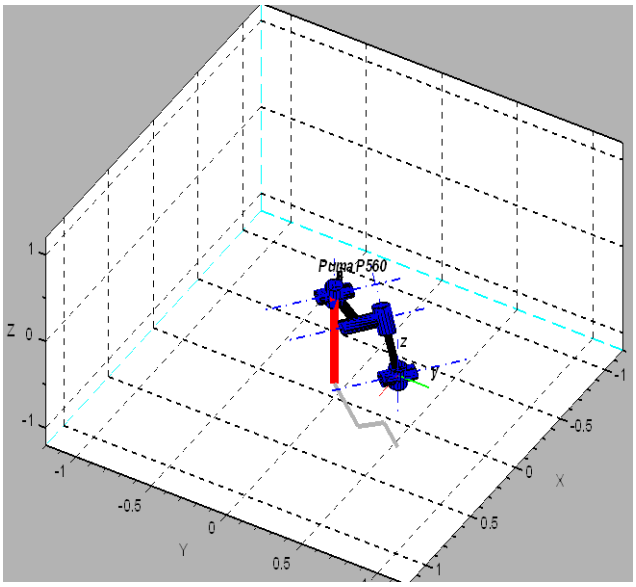
Megoldás:

```
clear L;
//             th    d       a          alpha
L1 = Link([ 0     0       0          %pi/2    0]);
L2 = Link([ 0     0       0.4318      0       0]);
L3 = Link([ 0     0.15005 0.0203    -%pi/2    0]);
L4 = Link([ 0     0.4318  0          %pi/2    0]);
L5 = Link([ 0     0       0          -%pi/2   0]);
L6 = Link([ 0     0       0           0       0]);
L=list(L1,L2,L3,L4,L5,L6);
p560 = SerialLink(L);
p560.name='Puma P560';
//plotrobot(p560,q_r)
g = 9.81;
r.gravity = [0 0 g];
tab = [0:0.05:5];
t = [tab 5+tab 10+tab 15+tab];
aq = jtraj([0 0 0 0 0 0],[-0.62832 -0.25133 -2.8274 0 0 0],tab);
bq = jtraj([-0.62832 -0.25133 -2.8274 0 0 0],[-0.62832 0.25133 -0.43982 0 0
0],tab);
cq = jtraj([-0.62832 0.25133 -0.43982 0 0 0],[2.3248 -0.31416 -2.5761 0 0
0],tab);
```

```
dq = jtraj([2.3248 -0.31416 -2.5761 0 0 0],[0 0 0 0 0 0],tab);
q = [aq;bq;cq;dq];

// Plot
figure(1);
plot(t,q);
xlabel('Time (s)');
ylabel('q (rad)');
legend({'q1','q2','q3','q4','q5','q6'},'FontSize',8,'FontWeight','bold')

for qq=q'
    plot_robot(p560,qq')
end
```

Robotics toolbox feladat 13

Modellezze a PUMA robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy az inverz kinematikai függvényt használva két tetszőleges pont közötti legrövidebb útvonalon haladjon végig az efektor!



8.20 ábra Szimulációs eredmény

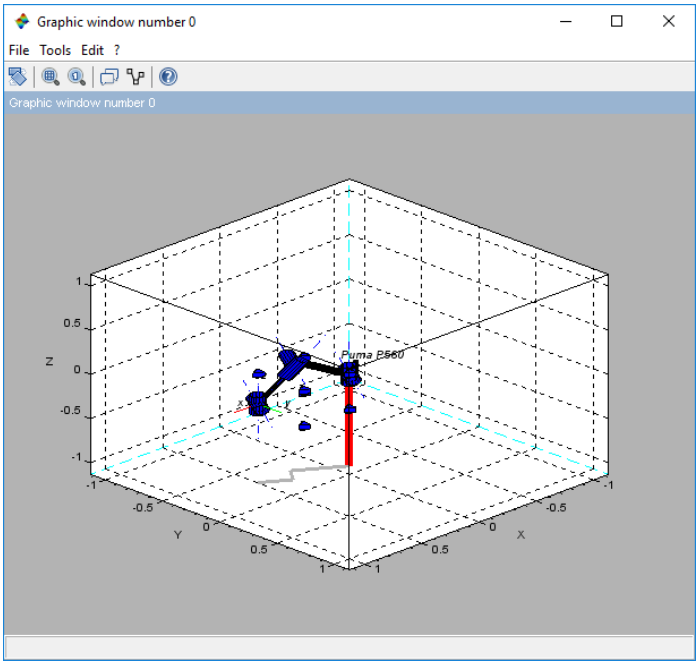
Megoldás:

```
clear L; // szegmensek definiálása az értékek theta d a alpha
L1 = Link([ 0 0 0 %pi/2 0]);
L2 = Link([ 0 0 0.4318 0 0]);
L3 = Link([ 0 0.15005 0.0203 -%pi/2 0]);
L4 = Link([ 0 0.4318 0 %pi/2 0]);
L5 = Link([ 0 0 0 -%pi/2 0]);
L6 = Link([ 0 0 0 0 0]);
L=list(L1,L2,L3,L4,L5,L6); //Robot definiálása
p560 = SerialLink(L);
```

```
p560.name='Puma P560';
t = [0:.5:100]; // idővektor
T1 = transl(0.3, -0.6, 0.4) // kezdő pont(X,Y,Z)
T2 = transl(0.3, 0.6, -0.2) // végpont (X,Y,Z)
T = ctraj(T1, T2, length(t)); // pálya kiszámítása
q = ikine6s(p560, T); // Inverz kinematika
for p=q' //animálás
plot_robot(p560,p');
end
```

Robotics toolbox feladat 14

Modellezze egy PUMA robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy az inverz kinematikai függvényt használva egy tetszőleges kocka élein haladjon végig az efektor!



8.21 ábra Szimulációs eredmény

Megoldás:

```
clear all;
L1=Link([0 0 0 %pi/2 0]);
L2=Link([0 0 0.4318 0 0]);
L3=Link([0 0.15005 0.0203 -%pi/2 0]);
L4=Link([0 0.4318 0 %pi/2 0]);
L5=Link([0 0 0 -%pi/2 0]);
L6=Link([0 0 0 0 0]);
L=list(L1,L2,L3,L4,L5,L6);
p560=SerialLink(L);
p560.name='Puma P560';
t=[0:5:100];
T1=transl(0.2, 0.2, 0.2);
T2=transl(0.2, 0.2, -0.2);
T3=transl(0.2, -0.2, -0.2);
```

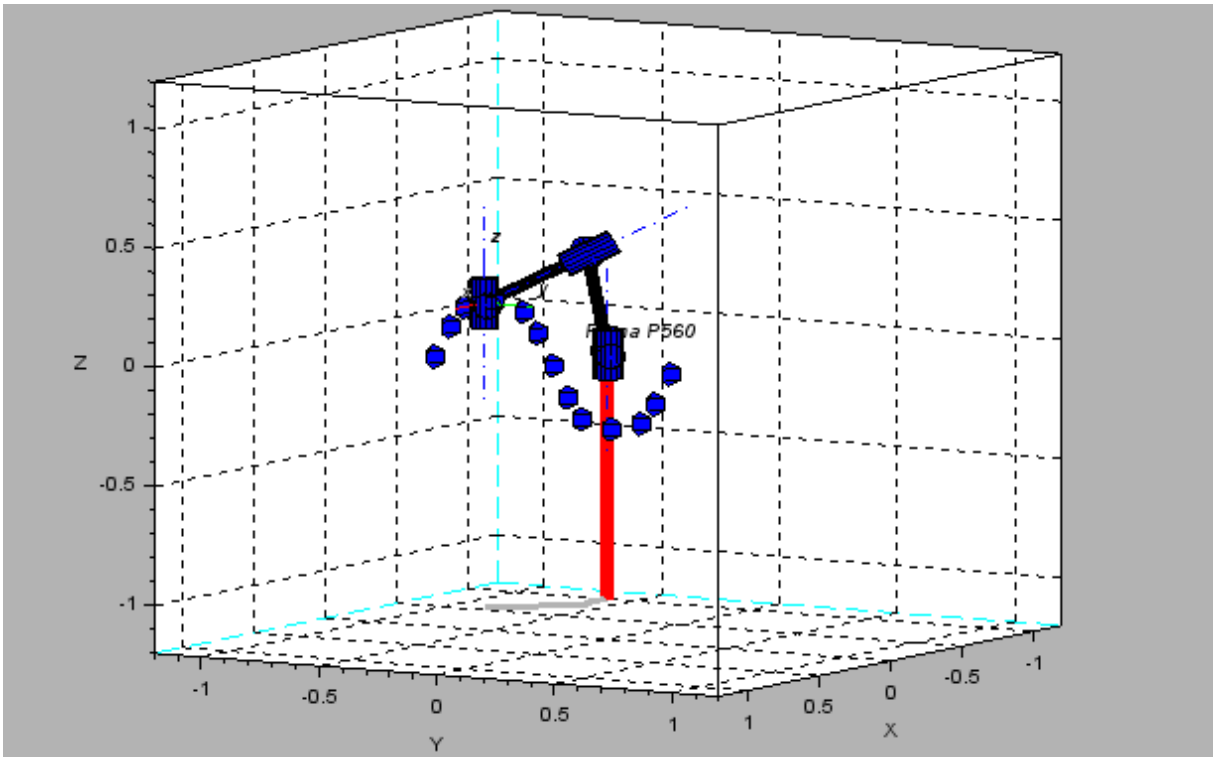
```
T4=transl(0.2, -0.2, 0.2);
T5=transl(0.6, 0.2, 0.2);
T6=transl(0.6, 0.2, -0.2);
T7=transl(0.6, -0.2, -0.2);
T8=transl(0.6, -0.2, 0.2);
T9=transl(0.4, 0, 0.4)

deff("[x,y,z]=sph(alp,tet)",["x=r*cos(alp).*cos(tet)+orig(1)*ones(tet)";
    "y=r*cos(alp).*sin(tet)+orig(2)*ones(tet)";
    "z=r*sin(alp)+orig(3)*ones(tet)"]);
r=0.05; orig=[0.2 0.2 0.2];

[x1,y1,z1]=eval3dp(sph,linspace(-%pi/2, %pi/2, 4), linspace(0,%pi*2,4));
orig=[0.2 0.2 -0.2];
[x2,y2,z2]=eval3dp(sph,linspace(-%pi/2,%pi/2,4), linspace(0,%pi*2,4));
orig=[0.2 -0.2 -0.2];
[x3,y3,z3]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.2 -0.2 0.2];
[x4,y4,z4]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.6 0.2 0.2];
[x5,y5,z5]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.6 0.2 -0.2];
[x6,y6,z6]=eval3dp(sph,linspace(-%pi/2, %pi/2, 4), linspace(0,%pi*2,4));
orig=[0.6 -0.2 -0.2];
[x7,y7,z7]=eval3dp(sph,linspace(-%pi/2,%pi/2,4), linspace(0,%pi*2,4));
orig=[0.6 -0.2 0.2];
[x8,y8,z8]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));

while(1)
    T=ctraj(T9,T1,length(t));
    q=ikine6s(p560,T);

    for p=q'
        plot_robot(p560,p')
    end

    plot3d(x1,y1,z1)
    plot3d(x2,y2,z2)
    plot3d(x3,y3,z3)
    plot3d(x4,y4,z4)
    plot3d(x5,y5,z5)
    plot3d(x6,y6,z6)
    plot3d(x7,y7,z7)
    plot3d(x8,y8,z8)
    T=ctraj(T1,T2,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T2,T3,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
```

```
end
    T=ctraj(T3,T4,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
    T=ctraj(T4,T1,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
    T=ctraj(T1,T5,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T5,T6,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T6,T2,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T2,T9,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T9,T5,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T5,T8,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T8,T7,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T7,T6,length(t));
q=ikine6s(p560,T);
for p=q'
    plot_robot(p560,p')
end
T=ctraj(T6,T9,length(t));
q=ikine6s(p560,T);
```

```
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T9,T4,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T4,T8,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T8,T9,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T9,T3,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T3,T7,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
    T=ctraj(T7,T9,length(t));
    q=ikine6s(p560,T);
    for p=q'
        plot_robot(p560,p')
    end
end
```

Robotics toolbox feladat 15

Modellezzen egy PUMA robot manipulátort a SCILAB Robotics Toolbox segítségével a képen látható módon, és tervezze meg a mozgása útvonalát oly módon, hogy az inverz kinematikai függvényt használva egy tetszőleges szinusz jel alakját követi végig az efektor!



8.22 ábra Szimulációs eredmény

Megoldás:

```
close;
clear all;
L1=Link([0 0 0 %pi/2 0]);
L2=Link([0 0 0.4318 0 0]);
L3=Link([0 0.15005 0.0203 -%pi/2 0]);
L4=Link([0 0.4318 0 %pi/2 0]);
L5=Link([0 0 0 -%pi/2 0]);
L6=Link([0 0 0 0 0]);
L=list(L1,L2,L3,L4,L5,L6);
p560 = SerialLink(L);
p560.name='Puma P560';
t=[0:2:100];
T1 =transl(0.4,0-0.5,0)                    //a pont
T2 =transl(0.4,0.0647-0.5,0.13)          //b pont
T3 =transl(0.4,0.125-0.5,0.2165)           //c pont
T4 =transl(0.4,0.25-0.5,0.25)              //d pont
T5 =transl(0.4,0.375-0.5,0.2154)           //e pont
T6 =transl(0.4,0.4353-0.5,0.13)          //f pont
T7 =transl(0.4,0.5-0.5,0)                  //g pont
T8 =transl(0.4,0.5647-0.5,-0.13)         //h pont
```

```
T9 =transl(0.4,0.624-0.5,-0.2165)           //i pont
T10=transl(0.4,0.75-0.5,-0.25)              //j pont
T11=transl(0.4,0.875-0.5,-0.2165)           //k pont
T12=transl(0.4,0.9353-0.5,-0.13)          //l pont
T13=transl(0.4,1-0.5,0)                     //m pont

deff("[x,y,z]=sph(alp,tet)",["x=r*cos(alp).*cos(tet)+orig(1)*ones(tet)";
"y=r*cos(alp).*sin(tet)+orig(2)*ones(tet)";
"z=r*sin(alp)+orig(3)*ones(tet)"]);
r=0.05;
orig=[0.4,0-0.5,0];
[x1,y1,z1]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.0647-0.5,0.13];
[x2,y2,z2]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.125-0.5,0.2165];
[x3,y3,z3]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.25-0.5,0.25];
[x4,y4,z4]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.375-0.5,0.2154];
[x5,y5,z5]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.4353-0.5,0.13];
[x6,y6,z6]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.5-0.5,0];
[x7,y7,z7]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.5647-0.5,-0.13];
[x8,y8,z8]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.624-0.5,-0.2165];
[x9,y9,z9]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.75-0.5,-0.25];
[x10,y10,z10]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.875-0.5,-0.2165];
[x11,y11,z11]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,0.9353-0.5,-0.13];
[x12,y12,z12]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
orig=[0.4,1-0.5,0];
[x13,y13,z13]=eval3dp(sph,linspace(-%pi/2,%pi/2,4),linspace(0,%pi*2,4));
while(1)
T=ctraj(T1, T2, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
plot3d(x1,y1,z1)
plot3d(x2,y2,z2)
plot3d(x3,y3,z3)
plot3d(x4,y4,z4)
plot3d(x5,y5,z5)
plot3d(x6,y6,z6)
plot3d(x7,y7,z7)
plot3d(x8,y8,z8)
plot3d(x9,y9,z9)
plot3d(x10,y10,z10)
plot3d(x11,y11,z11)
```

```
plot3d(x12,y12,z12)
plot3d(x13,y13,z13)
T=ctraj(T2, T3, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T3, T4, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T4, T5, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T5, T6, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T6, T7, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T7, T8, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T8, T9, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T9, T10, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T10, T11, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T11, T12, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T12, T13, length(t));
```

```
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
T=ctraj(T13, T1, length(t));
q=ikine6s(p560,T);
for p=q'                          //animálás
plot_robot(p560, p')
end
end
```

Ellenőrző kérdések:

1. Milyen mozgástervezési módszereket ismer?
2. Mi az inverz kinematika előnye?
3. Hogyan történik egy mozgásterv elkészítése?
4. Hogyan jelenítjük meg a mozgáster karakterisztikus pontjait?
5. Hogyan számoljuk ki a két megadott pont közötti legrövidebb útvonalat?

IRODALOMJEGYZÉK

[1] Alain Vande Wouwer Philippe Saucez Carlos Vilas: Simulation of ODE/PDE Models with MATLAB, OCTAVE and SCILAB, Springer, 2014.

[2] Matlab - http://www.mathworks.com/products/matlab/

[3] Scilab - http://www.scilab.org/

[4] Scilab-Xcos Pour L'enseignement Des Sciences De L'ingénieur, Scilab Enterprises, 2013.

[5] Simulink - http://www.mathworks.com/products/simulink/

[6] Stephen L. Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah: Modeling and Simulation in Scilab/Xcos, Springer, 2010.

[7] Stoyan Gisbert: MATLAB. Typotex, 2005.

[8] Xcos - https://www.scilab.org/scilab/gallery/xcos

[9] J.-P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikoukhah, and S. Steer. Introduction a Scilab, Deuxieme Edition. Springer, 2007.

[10] Peter I. Corke, A Robotics Toolbox for Matlab Release 8, 2008.

[11] Lantos Béla, Irányítási rendszerek elmélete és tervezése I. Budapest: Akadémiai Kiadó, 2009.

[12] Ajtonyi István, Automatizálási és kommunikációs rendszerek. Miskolc: Miskolci Egyetemi Kiadó, 2006.

[13] Yassine Ariba, Introduction to Scilab application to feedback control, Brno University of Technology, 2015.

[14] Michael Baudin, Programming in Scilab, Consortium Scilab - Digitéo, 2011.

[15] Nagy Gergely, Az 555-ös időzítő használata mikrokontrolleres tervezésben, Egyetemi jegyzet, Budapest, 2003.

[16] Lovassy Rita, Analóg és Digitális Technika I, Egyetemi jegyzet, Budapest, 2010.