SZABADKAI MŰSZAKI SZAKFŐISKOLA

SZABADKA

DC motor pozíciószabályozása

- projektum -

Hallgatók:	Pálfi Szuzanna	09213002	Tárgy:	Digitális irányítórendszerek
	Szegedi Mihály	09213003	Tanár:	Dr. Pletl Szilveszter

Szabadka, 2014

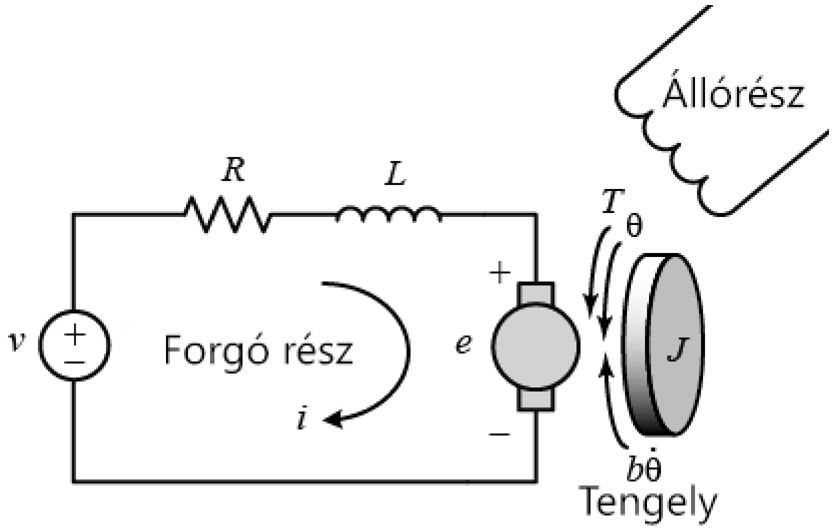
Tartalomjegyzék

A feladat leírása

A feladat célja egy szabályzó megtervezése SIMULINK-ben, 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 kefés DC motor fizikai modellje

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



1. ábra - Kefés 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

Rendszeregyenletek

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 MATLAB-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 MATLAB-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.

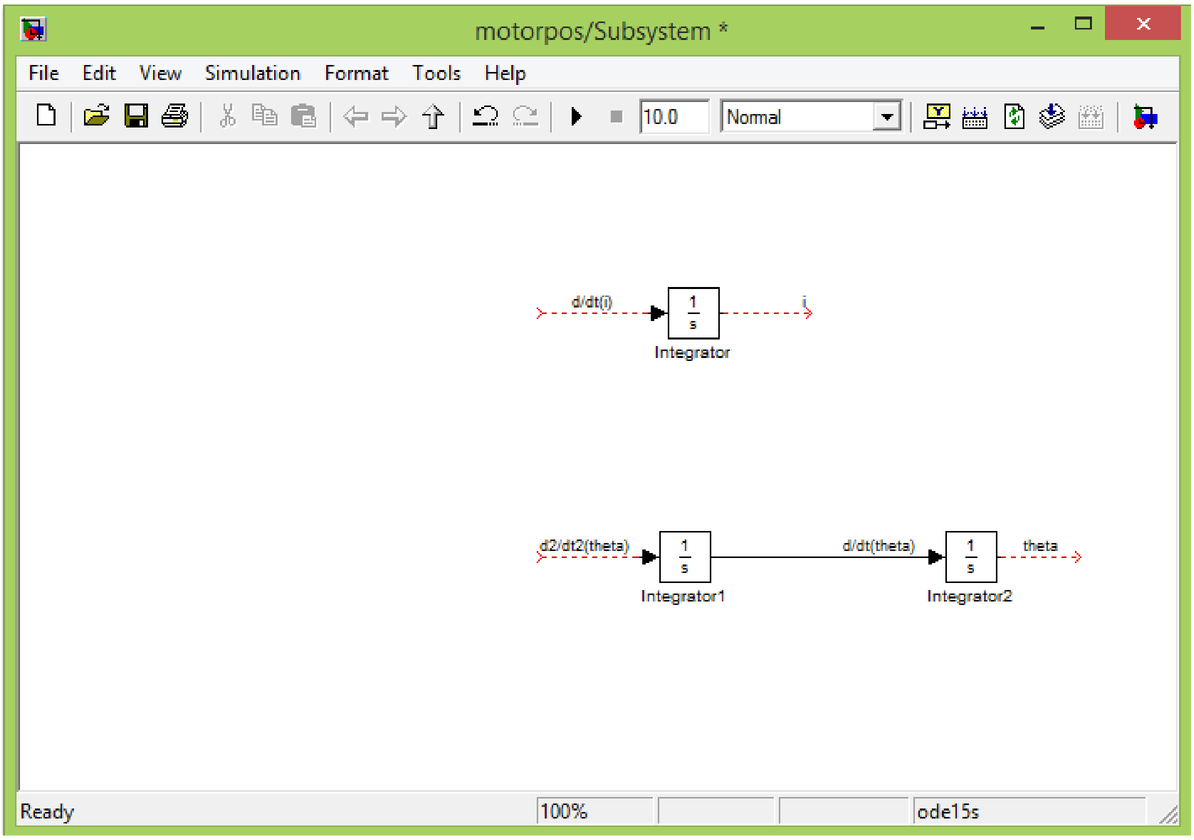
A rendszer modellezése simulinkben

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 simulinkben berakjuk a szükséges integrator blokkokat.



2. ábra – Integrátorok bevitele Simulinkben

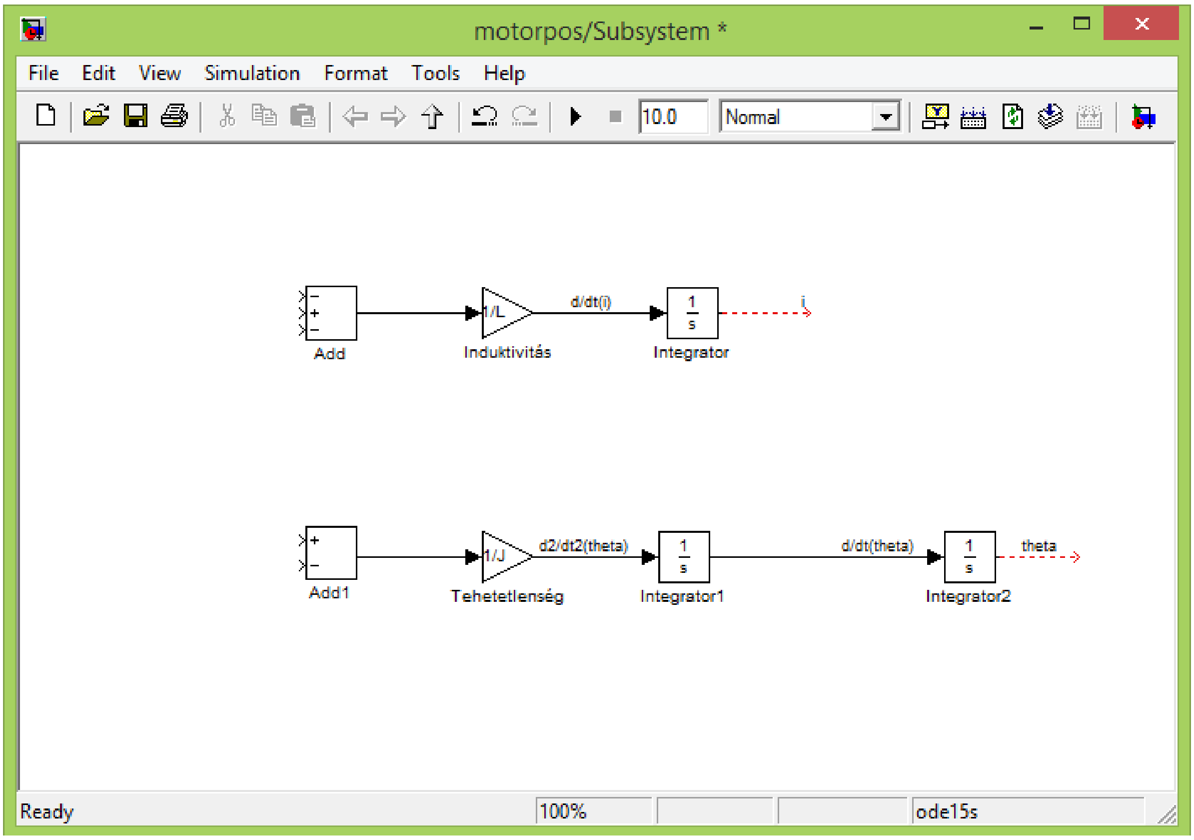
Ezután alkalmazzuk Kirhoff törvényeit a motor mehanikai és elektromos rendszerére is. A mehanikai rendszerre a következő képletet írjuk fel:

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

Az elektromos rendszerre a következő képletet írjuk fel:

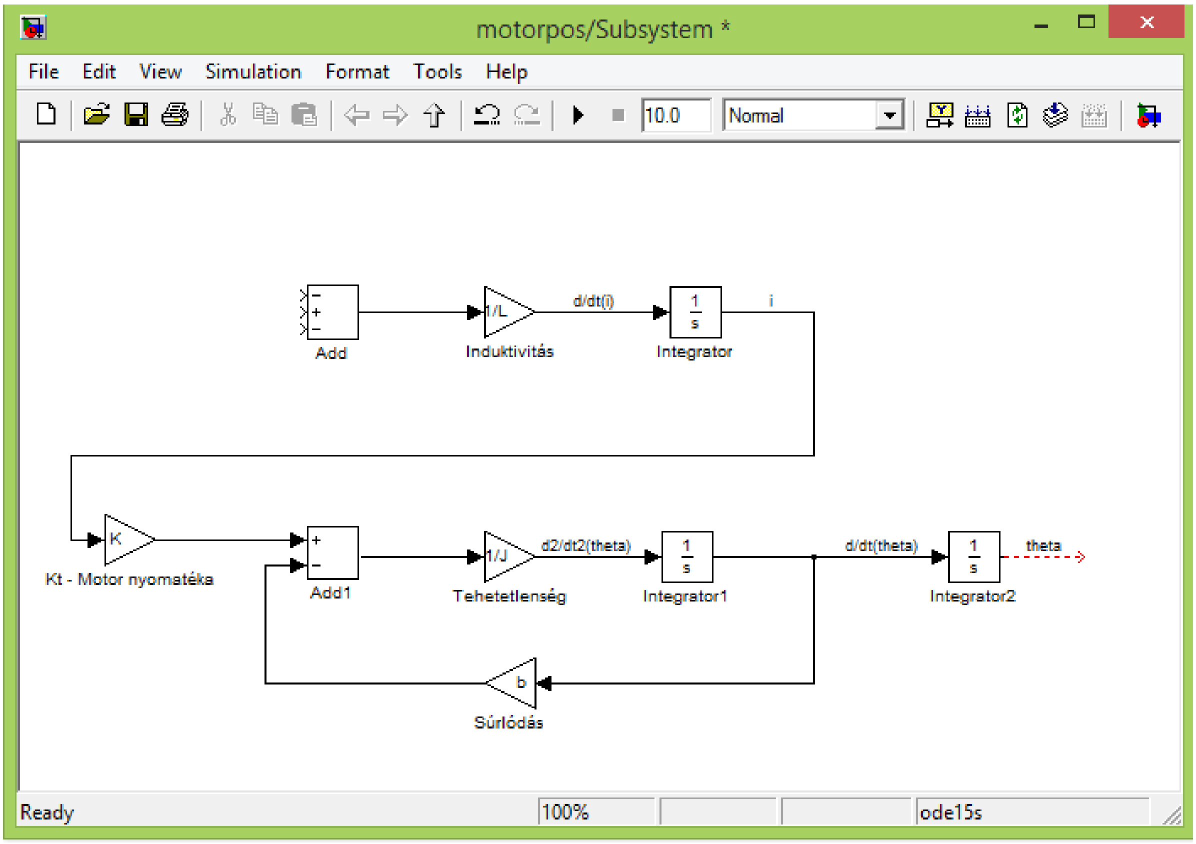
$$L\frac{di}{dt} = -Ri + V - e \Rightarrow \frac{di}{dt} = \frac{1}{L}(-Ri + V - K_b\frac{d\theta}{dt})$$

A szöggyorsulás egyenlő 1/J megszorozva két tényezővel. Hasonlóan az áram deriváltja egyenlő 1/L megszorozva három tényezővel.



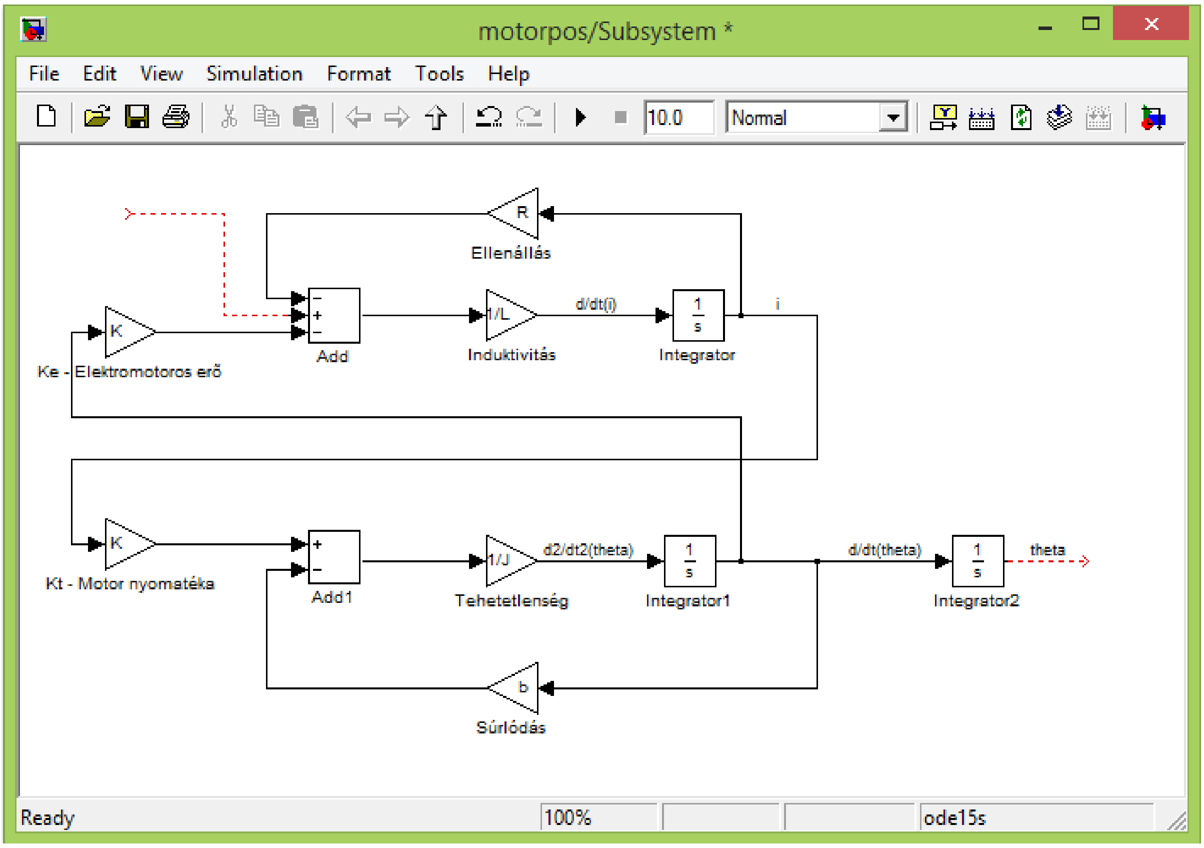
3. ábra – Induktivitás és tehetelenség hozzáadása Simulinkben

Ezután hozzáadjuk a nyomatékokat, melyeket Newton törvényeiből kapunk. Először a súrlódást, majd pedig az armatúra nyomatékát.



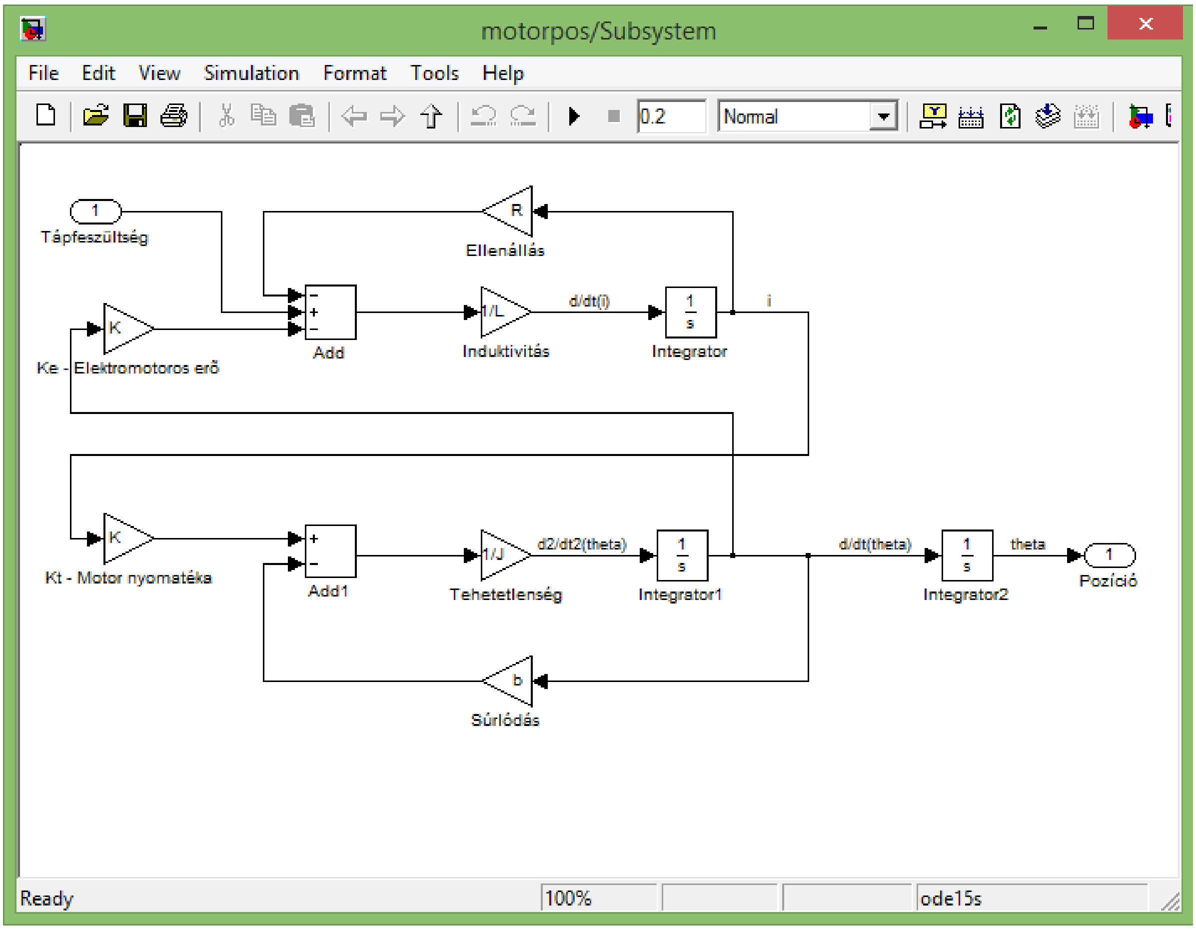
4. ábra – Súrlódás és a motor nyomatékának hozzáadása Simulinkben

Ezután a feszültségre vonatkozó elemeket adjuk hozzá, először a feszültségesést az armatúra ohmikus ellenállásán keresztül, majd pedig az elektromotoros erőt.



5. ábra – Elektronotoros erő és feszültségesés

Ezek után már csak a bemenetet és a kimenetet kell a rendszerhez illeszteni.

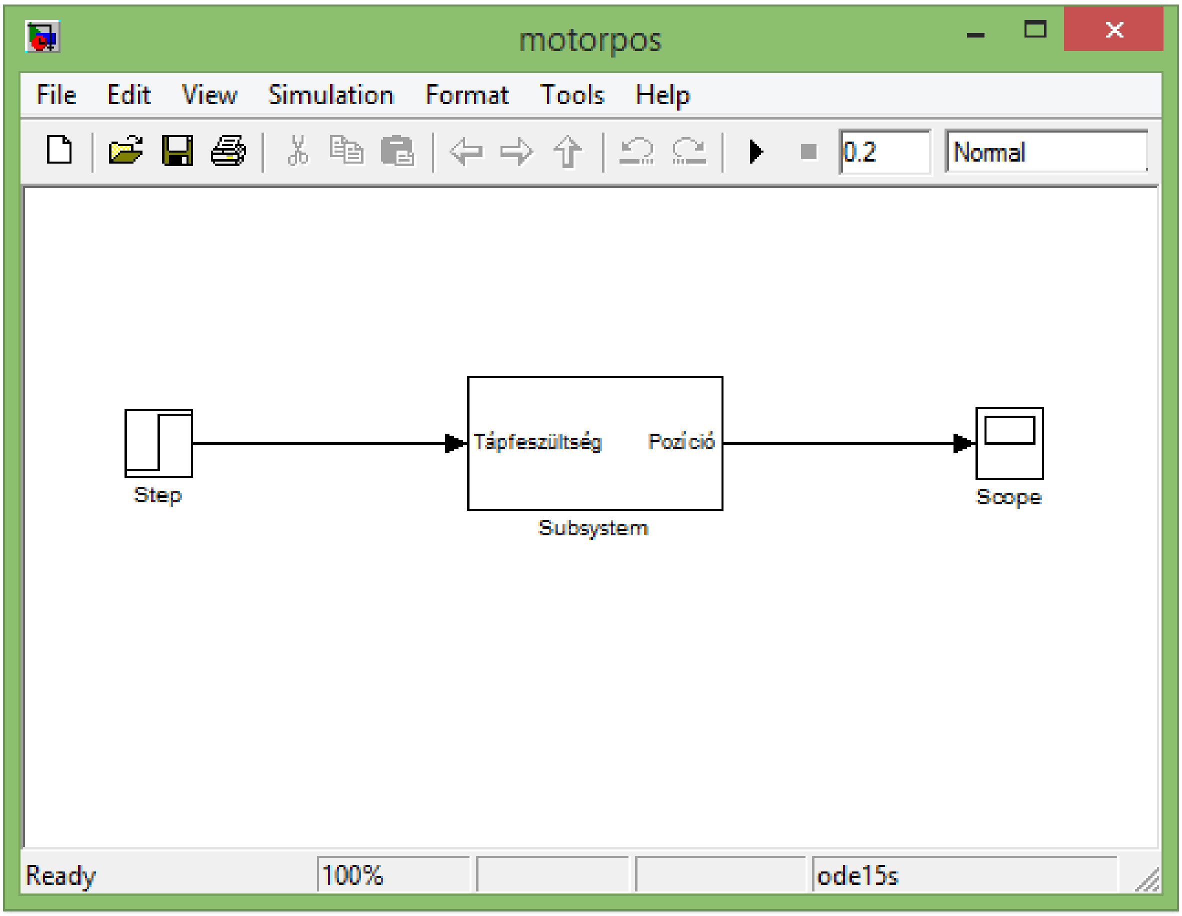


6. ábra – Bemenet és kimenet hozzáadása Simulinkben

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:



7. ábra – A kész rendszer Simulinkben

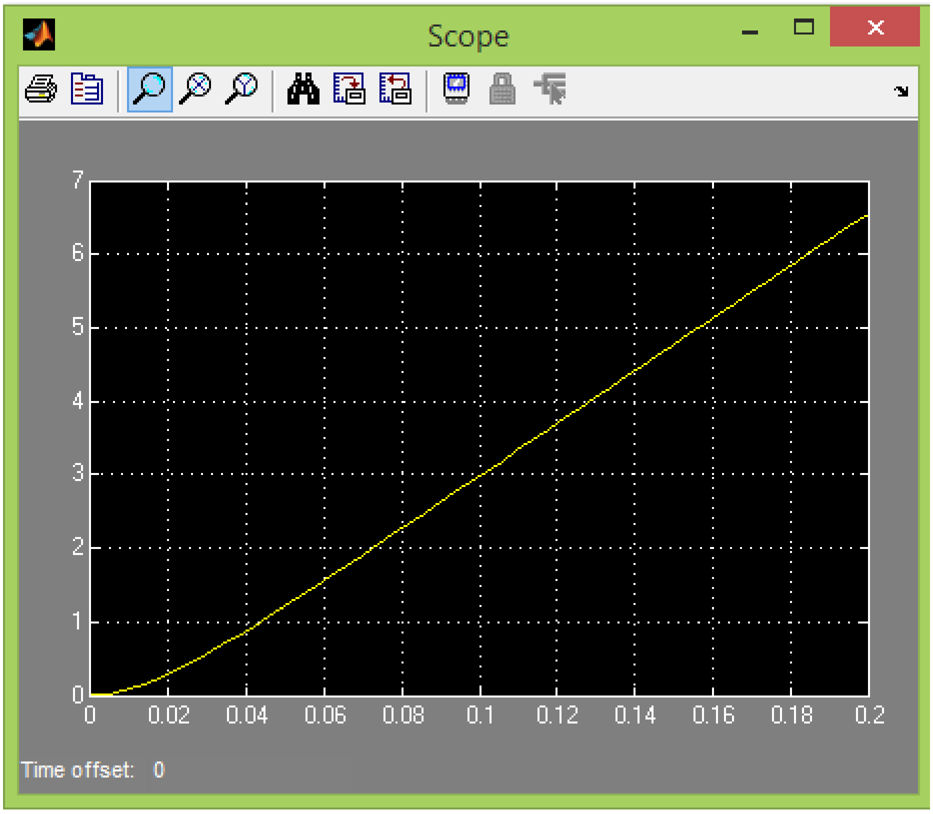
A szimulációnál a következő értékeket használtuk a motor modellezéséhez:

- 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.



8. 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.

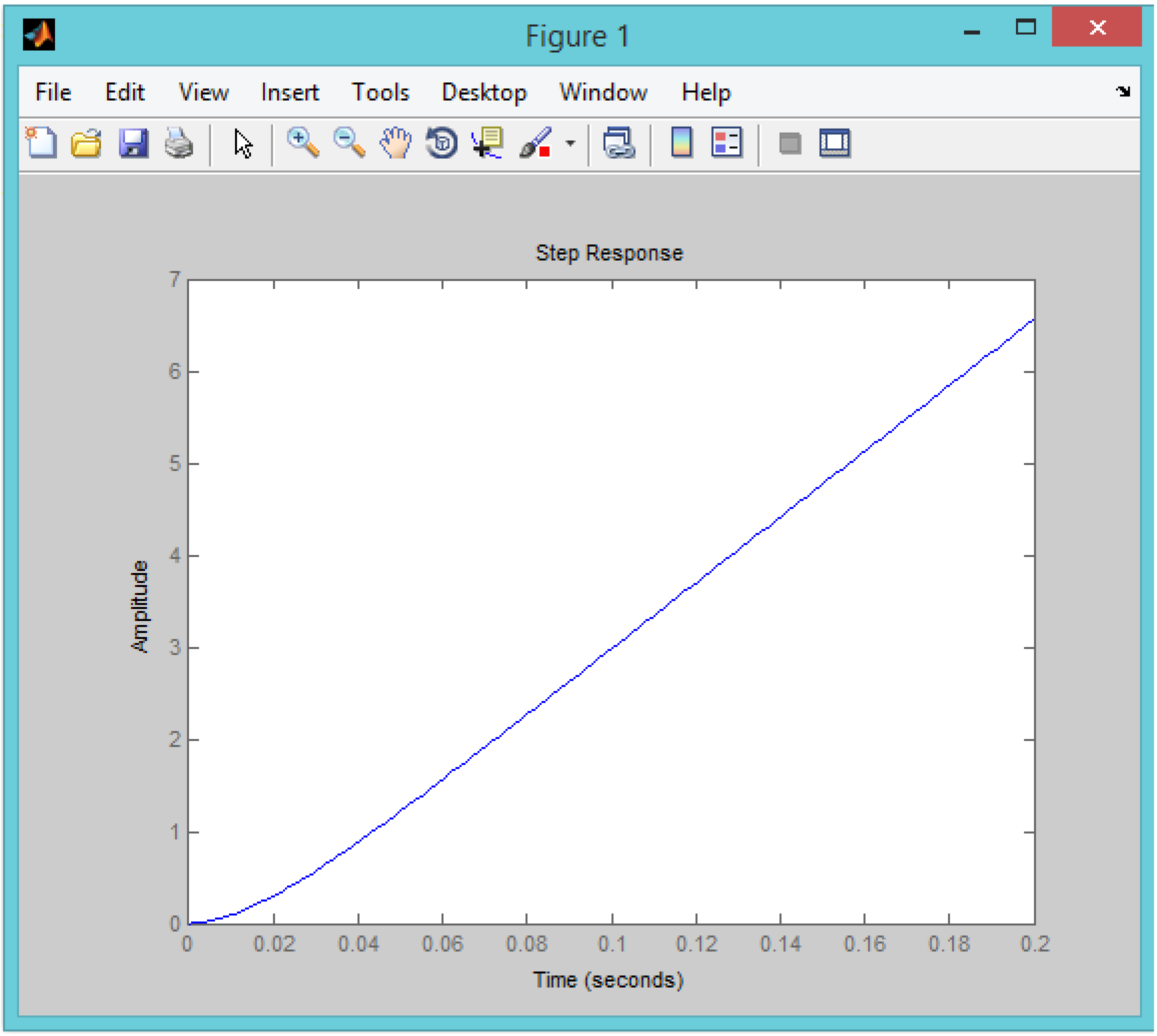
Nyílt hurkú válasz

Először is hozzunk létre MATLAB-ban egy új m-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)
```



9. 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.

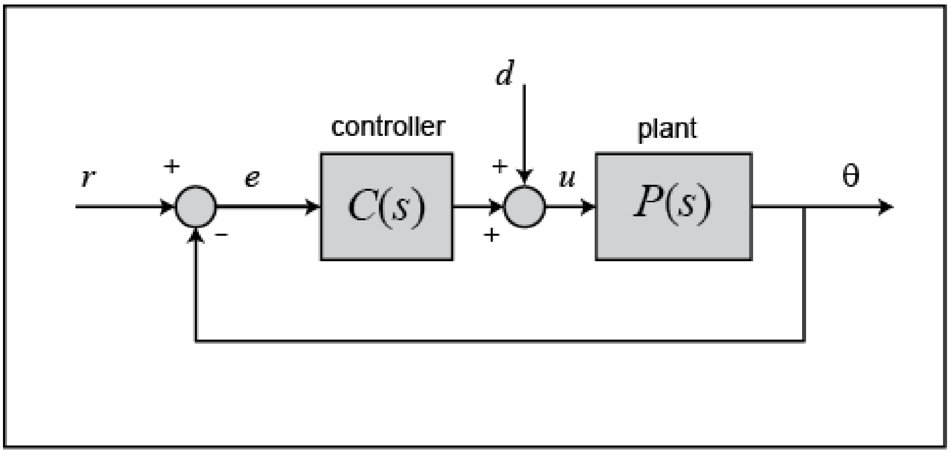
```
pole(P_motor)
ans =
   1.0e+06 *
         0
   -1.4545
   -0.0001
```

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:



10. 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:

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

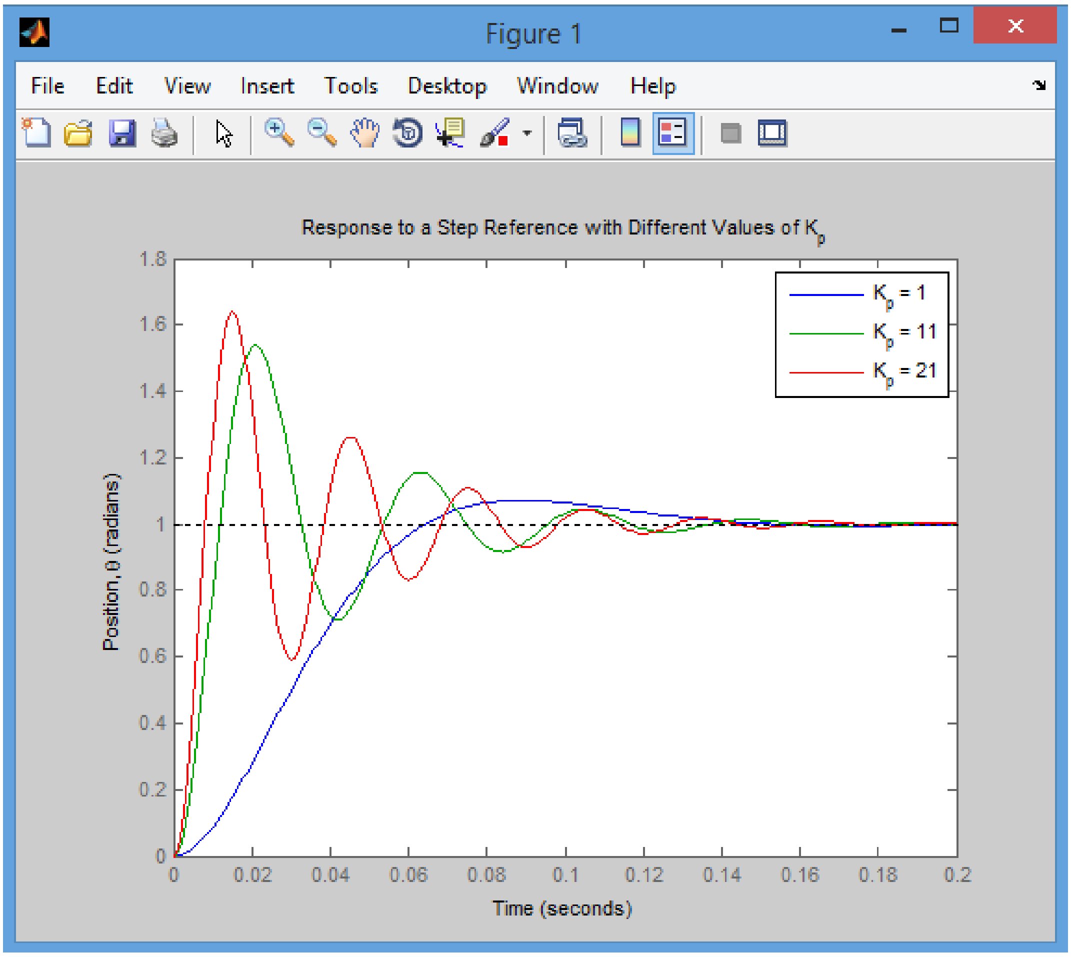
Arányos szabályozás

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 lépés válaszát:

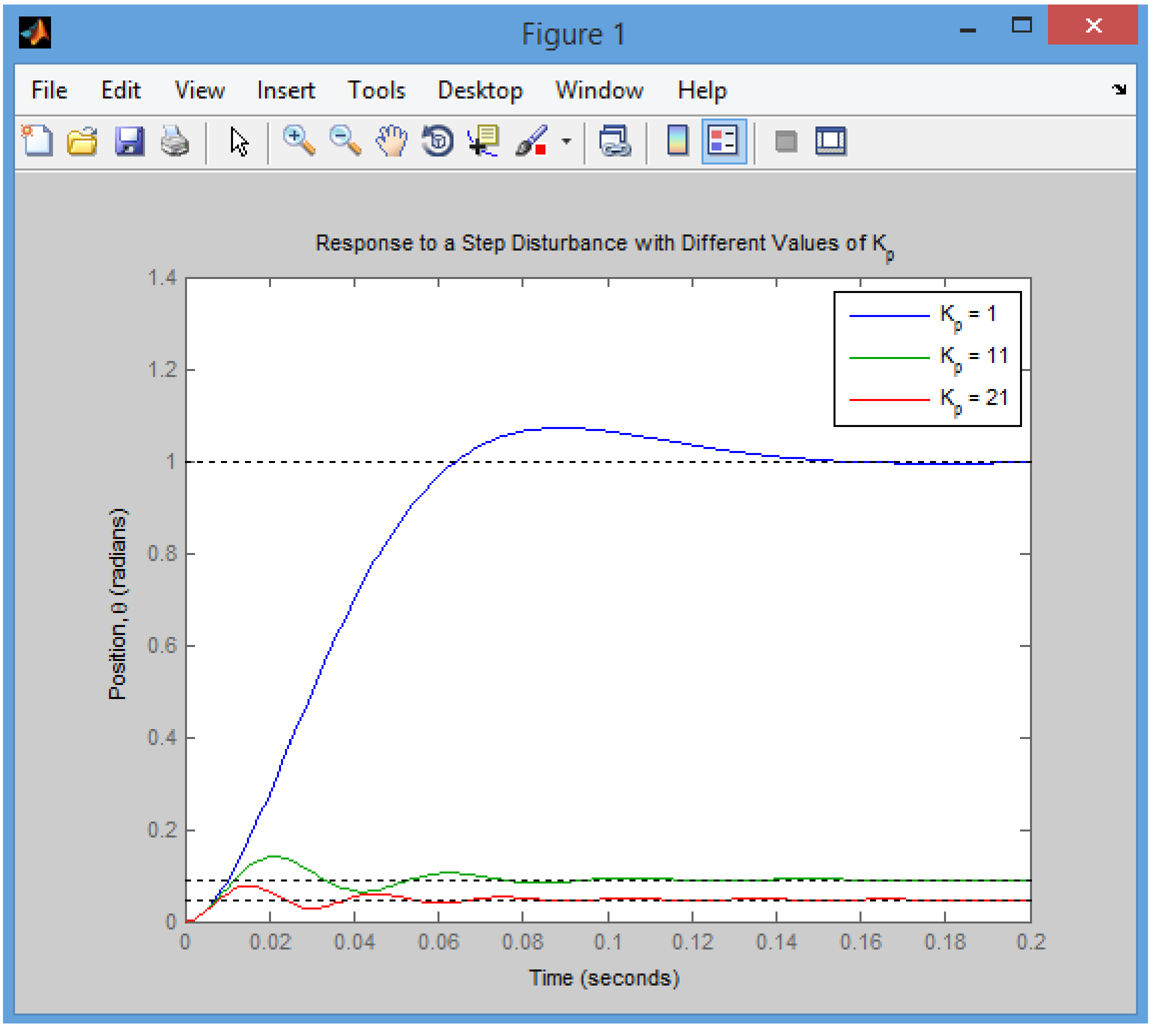
```
t = 0:0.001:0.2;
step(sys_cl(:,:,1), sys_cl(:,:,2), sys_cl(:,:,3), t)
ylabel('Position, \theta (radians)')
title('Response to a Step Reference with Different Values of K_p')
legend('K_p = 1',  'K_p = 11',  'K_p = 21')
```



11. ábra – A zárt hurkú szabályzó átviteli függvényének lépés válasza

Most nézzük 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')
```



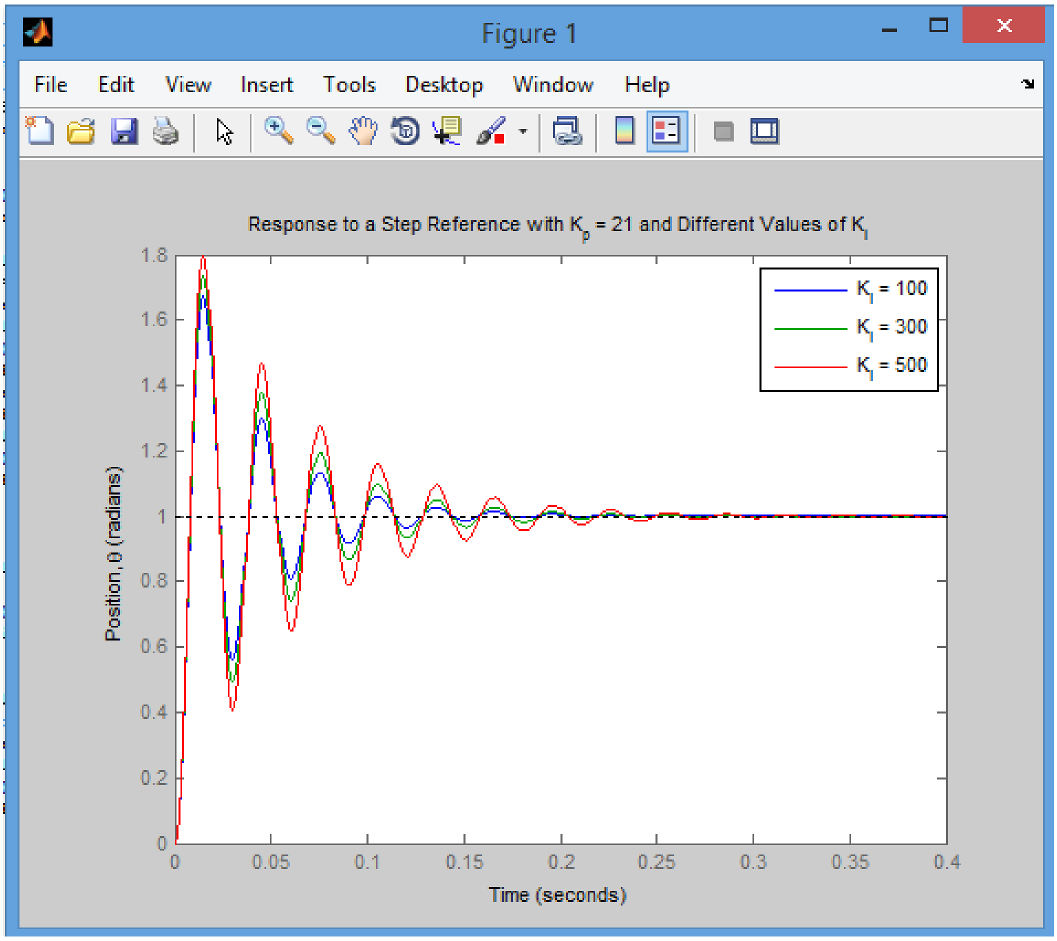
12. ábra – A rendszer viselkedése terhelés alatt

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 fileunk tartalmát cseréljük ki a következőre, és futtassuk a parancssorban. Az alábbi ábra fog kiraljzolódni:

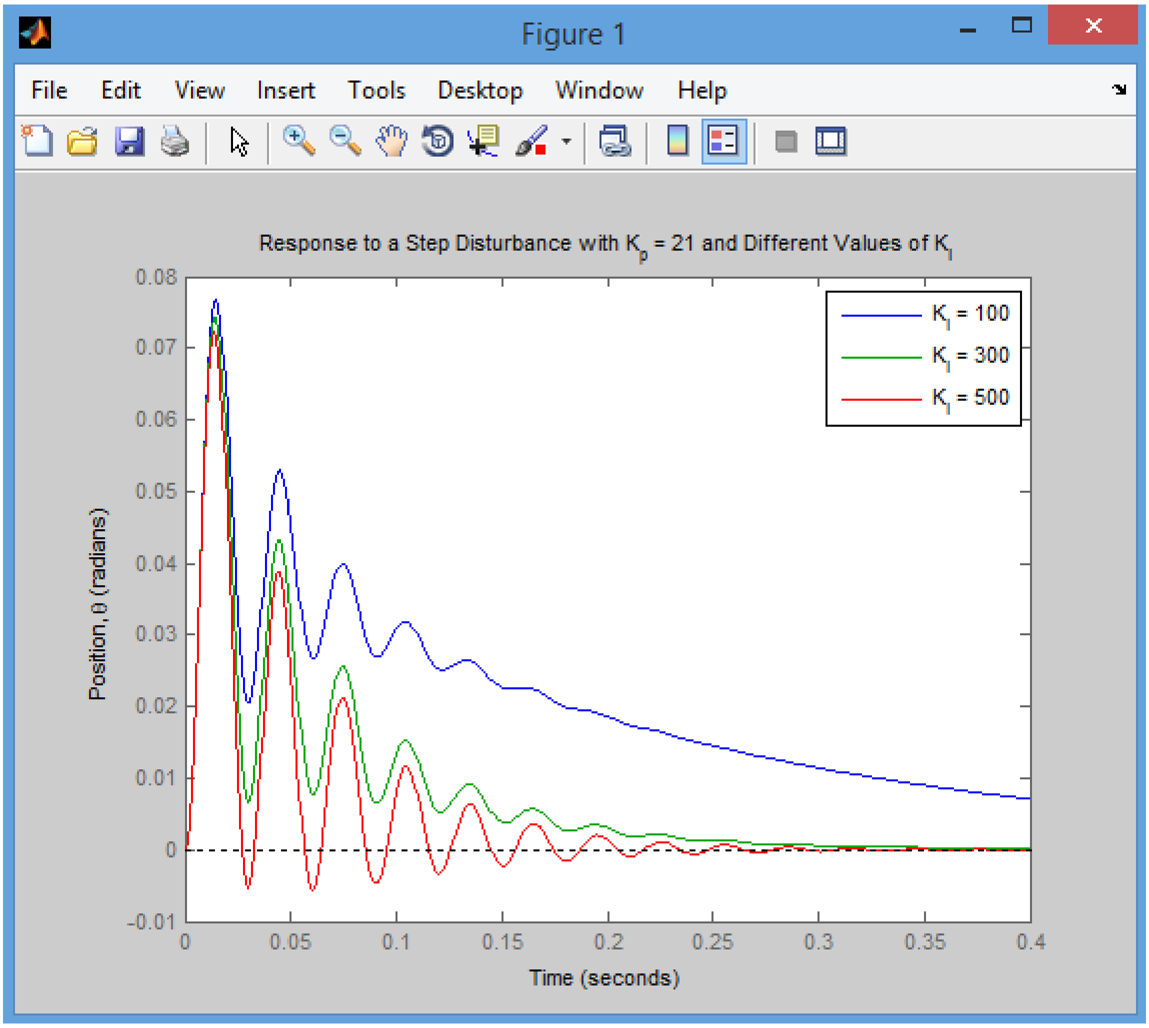
```
Kp = 21;
Ki = 100;
for i = 1:5
    C(:,:,i) = pid(Kp,Ki);
    Ki = Ki + 200;
end

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')
```



13. 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')
```



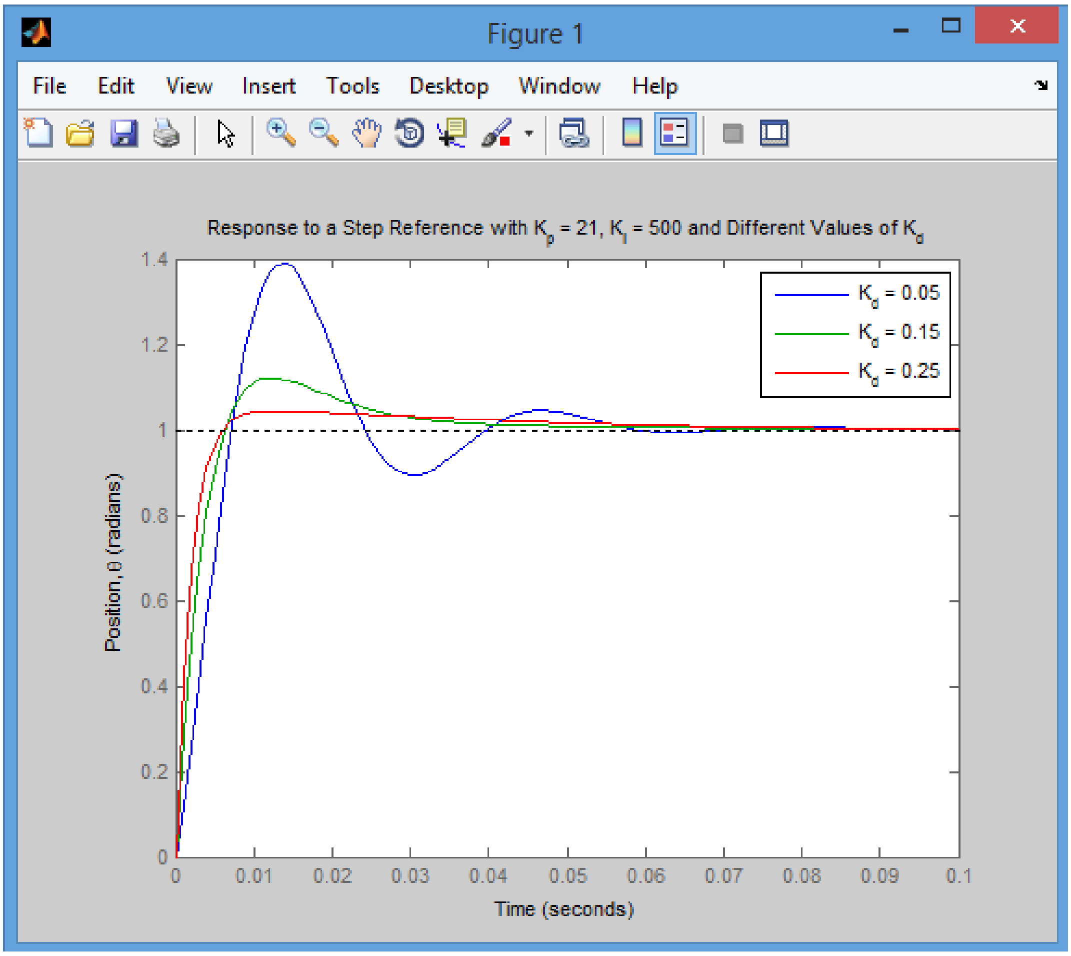
14. ábra – Bemenet terhelése

Az integrális vezérlés 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.

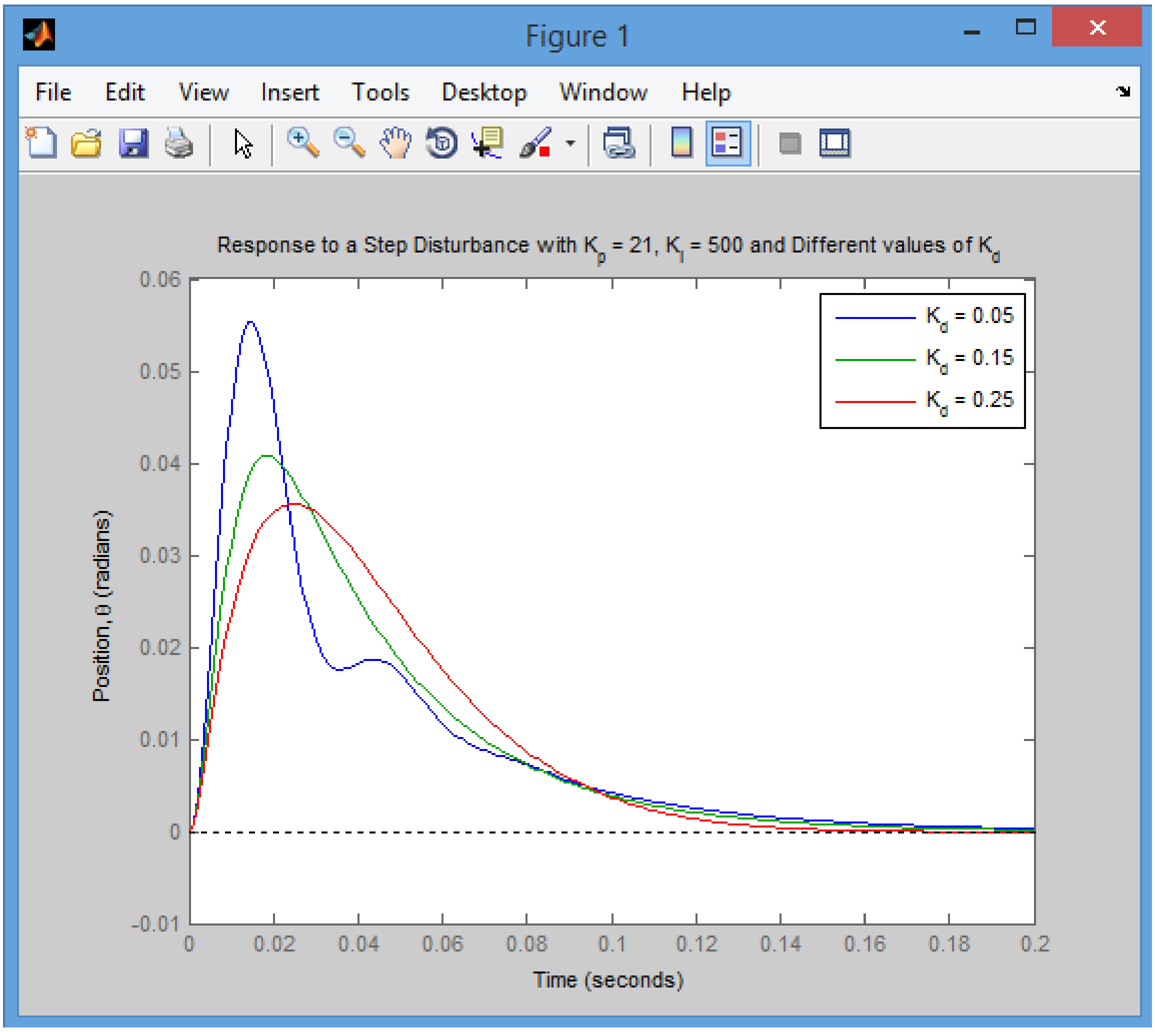
```
Kp = 21;
Ki = 500;
Kd = 0.05;

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')
```



15. ábra – Kd érték hozzáadása a rendszerhez

Nézzük 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')
```



16. ábra – Gerjesztés válasz

Úgy néz ki, mikor Kd=0.15, mint amikor a rendszerünk követelményeit kértük. Hogy pontosan meghatározzuk a karakterisztikáját a gerjesztés válasznak a jobb-klikk menüben ki kell választani a step response plot-ot, vagy a stepinfo parancsot 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);
```

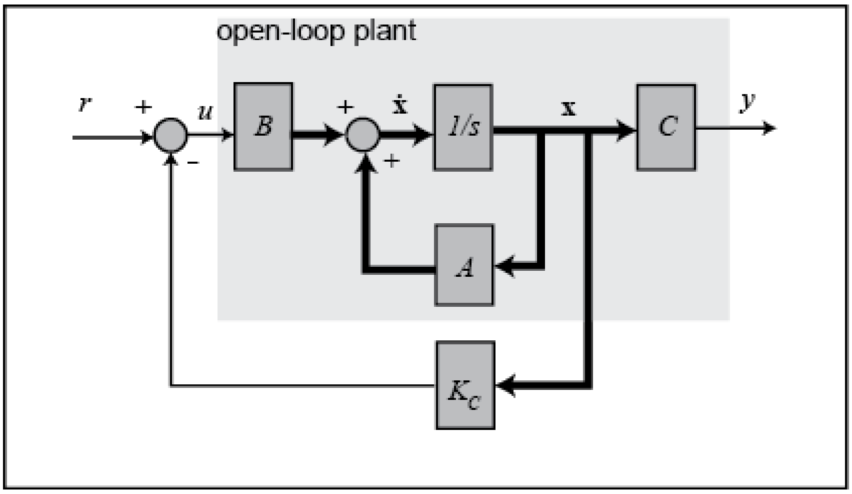
Visszacsatolt állapottér szabályozó tervezése

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.



17. 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 ^ 2B* ...]. A *ctrb* MATLAB parancs felépíti az *A* és *B* irányíthatósági mátrixot, továbbá a *rank* parancs az adott mátrix rangját. 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 MATLAB 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:

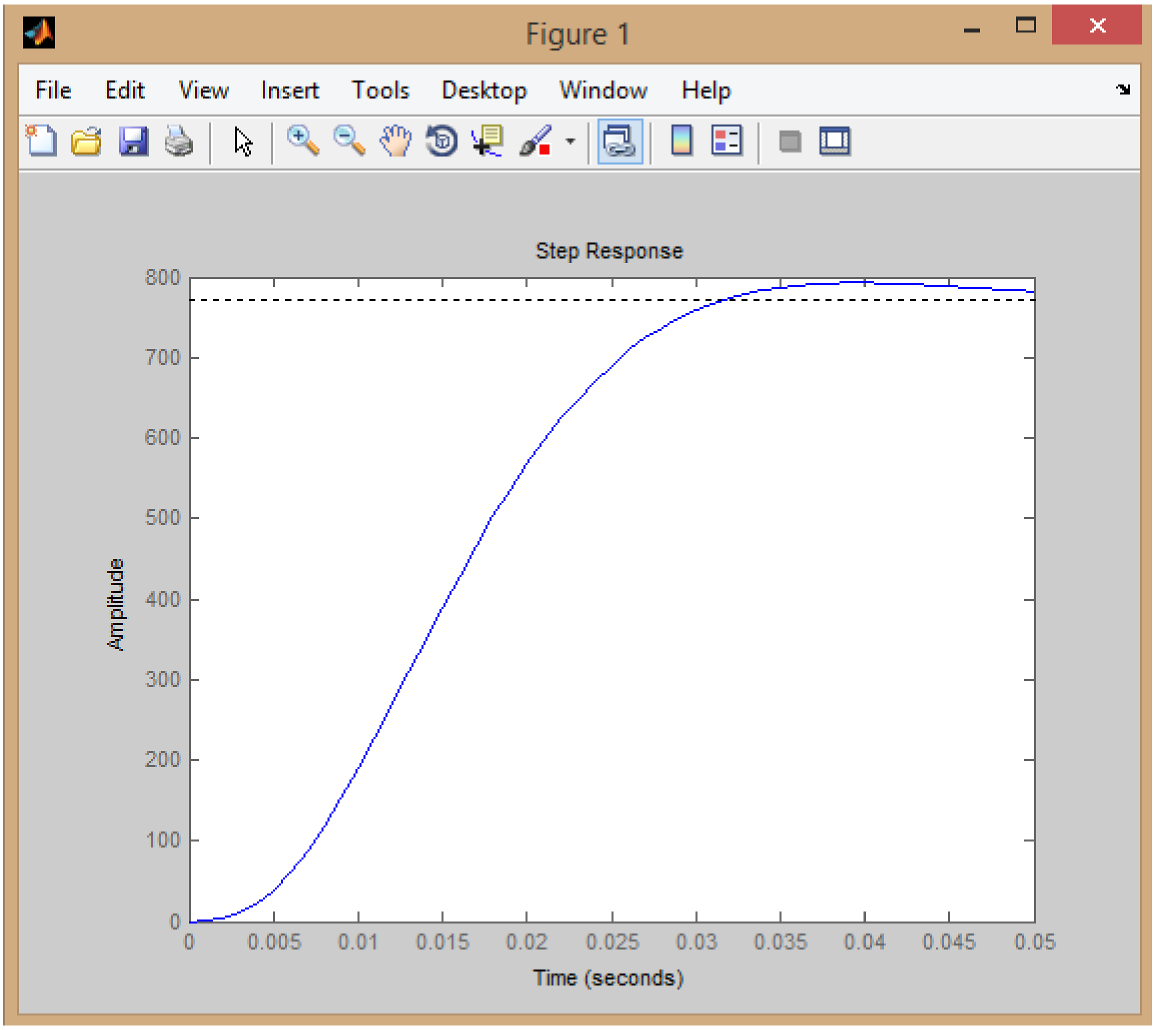
```
p1 = -100+100i;
p2 = -100-100i;
p3 = -200;
Kc = place(A,B,[p1, p2, p3])
Kc =
    0.0013   -0.0274   -3.9989
```

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)
```



18. 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) tartószervet feltételezünk.

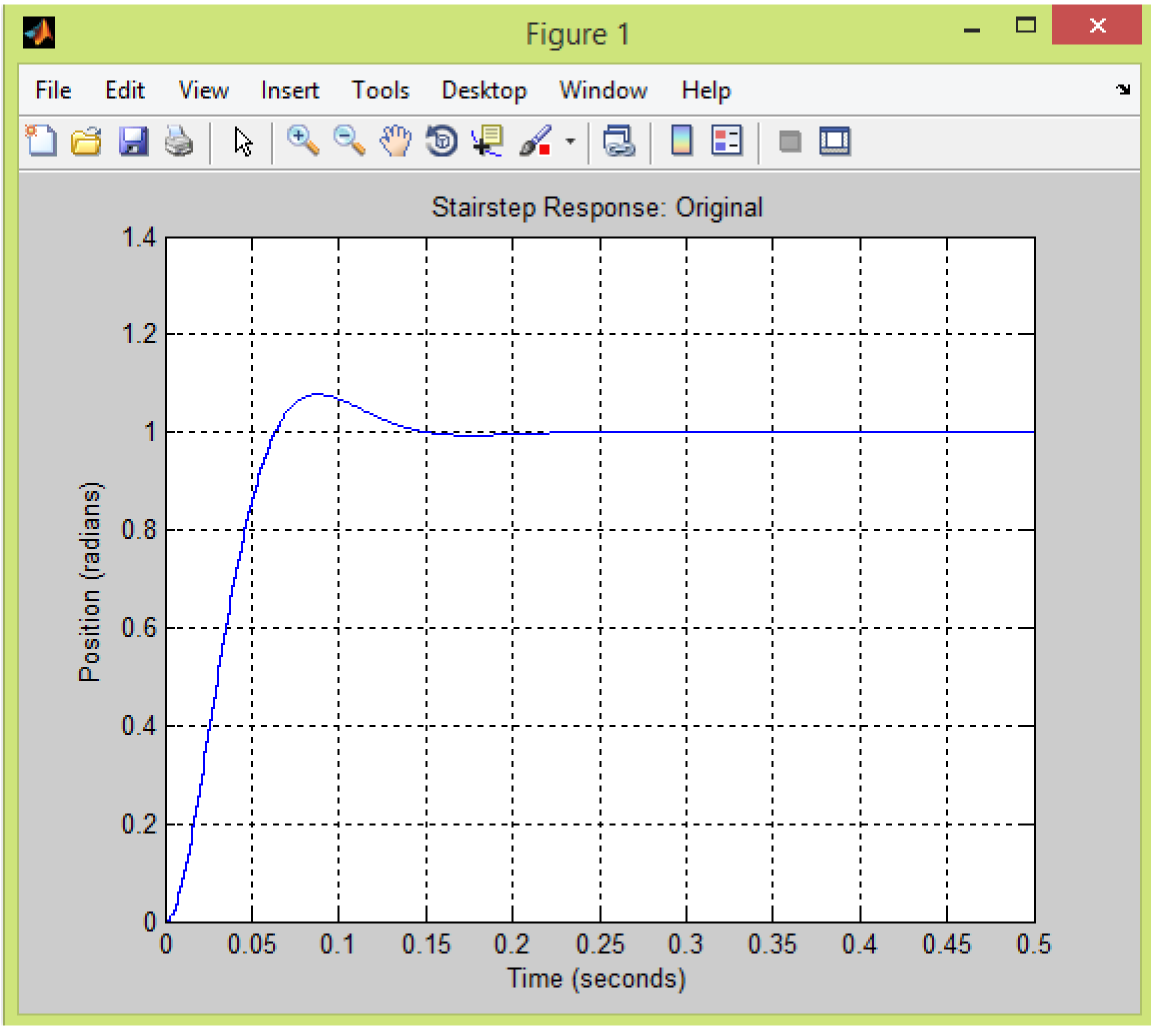
```
Ts = 0.001;
dP_motor = c2d(P_motor, Ts, 'zoh');
zpk(dP_motor)


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 MATLAB-ban. A mineral parancs használata következő ábrán látható:

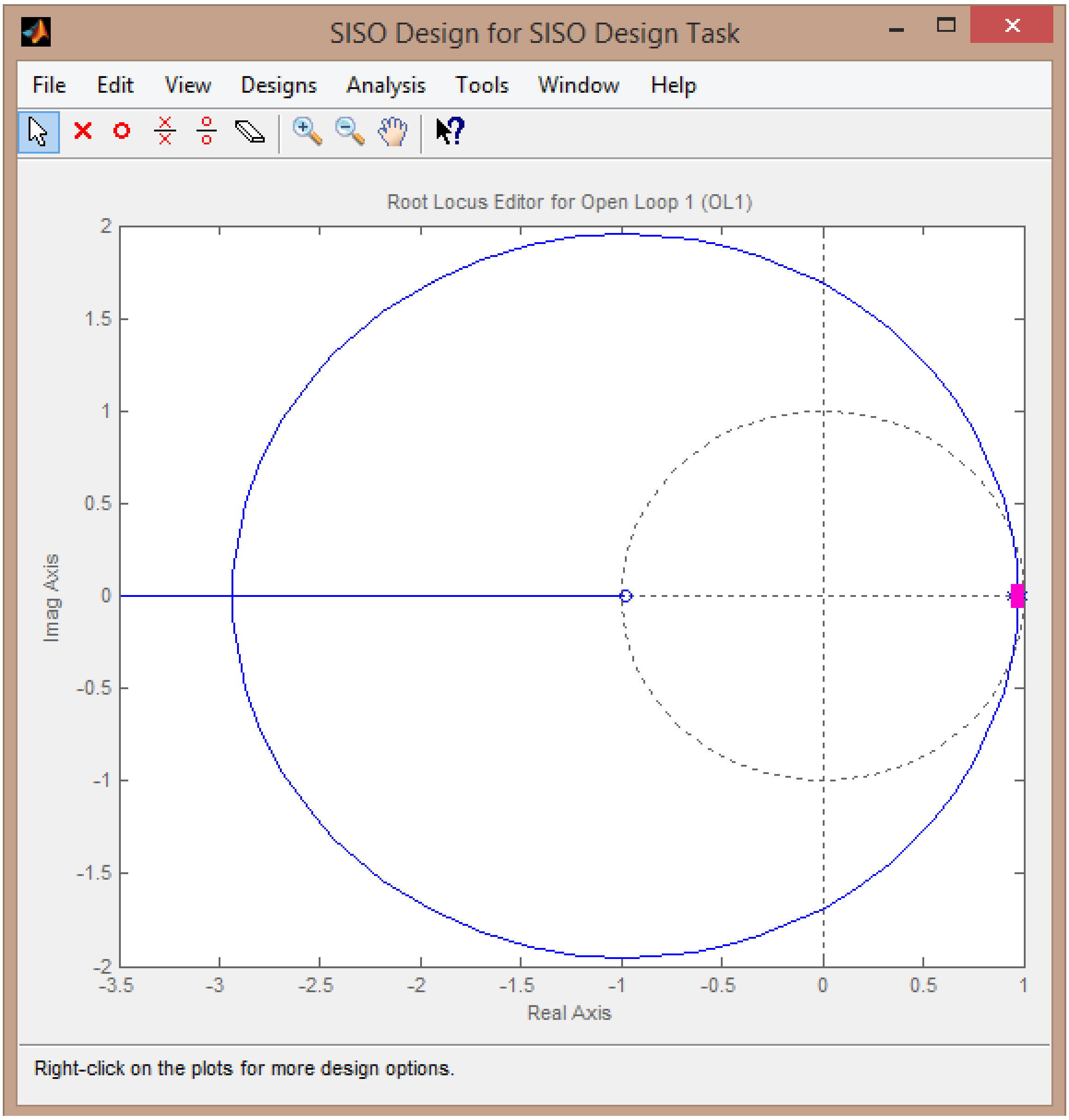
```
dP_motor = minreal(dP_motor,0.001);
zpk(dP_motor)
ans =

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

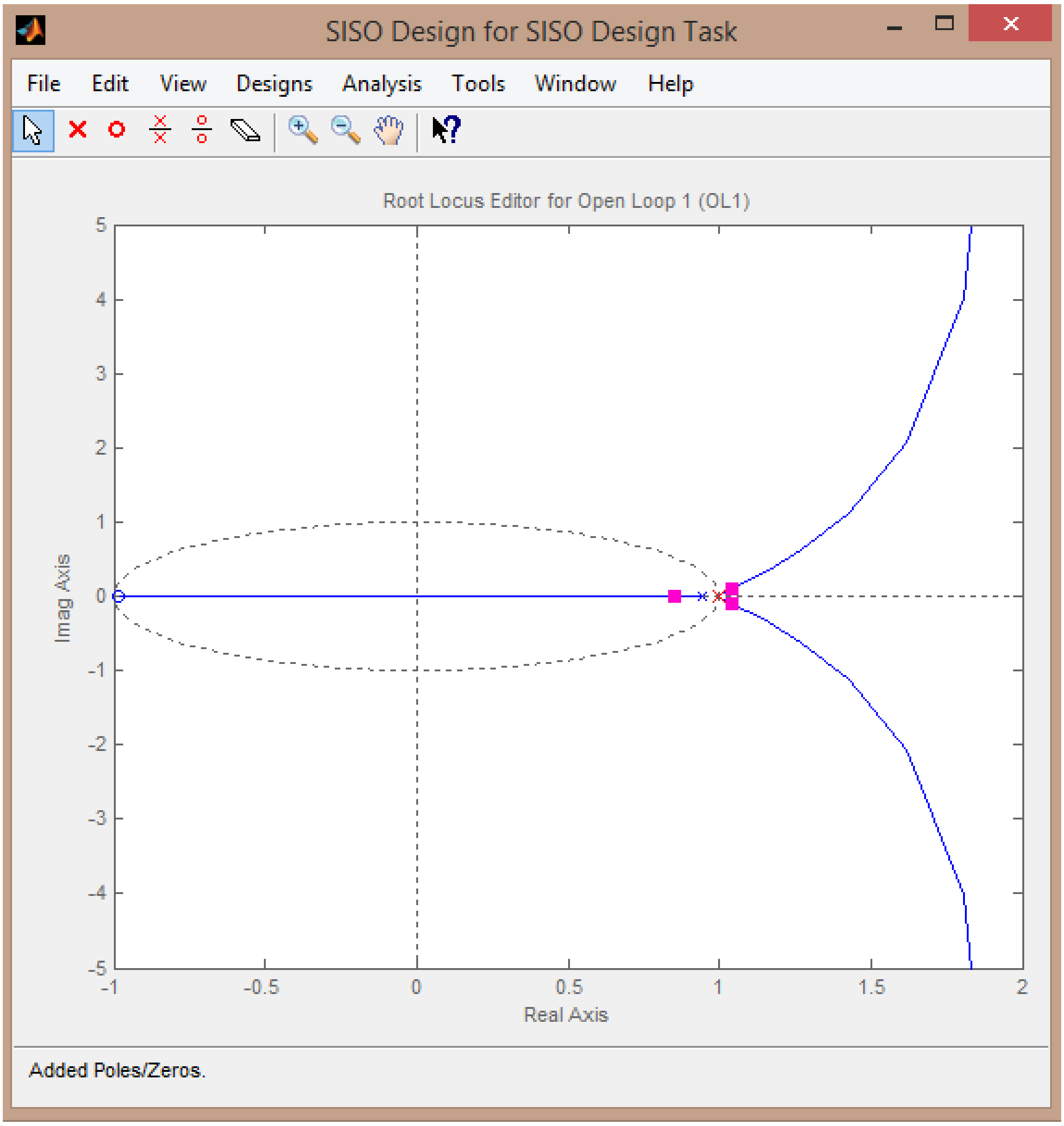
```
sys_cl = feedback(dP_motor,1);
[x1,t] = step(sys_cl,.5);
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Position (radians)')
title('Stairstep Response: Original')
grid
```



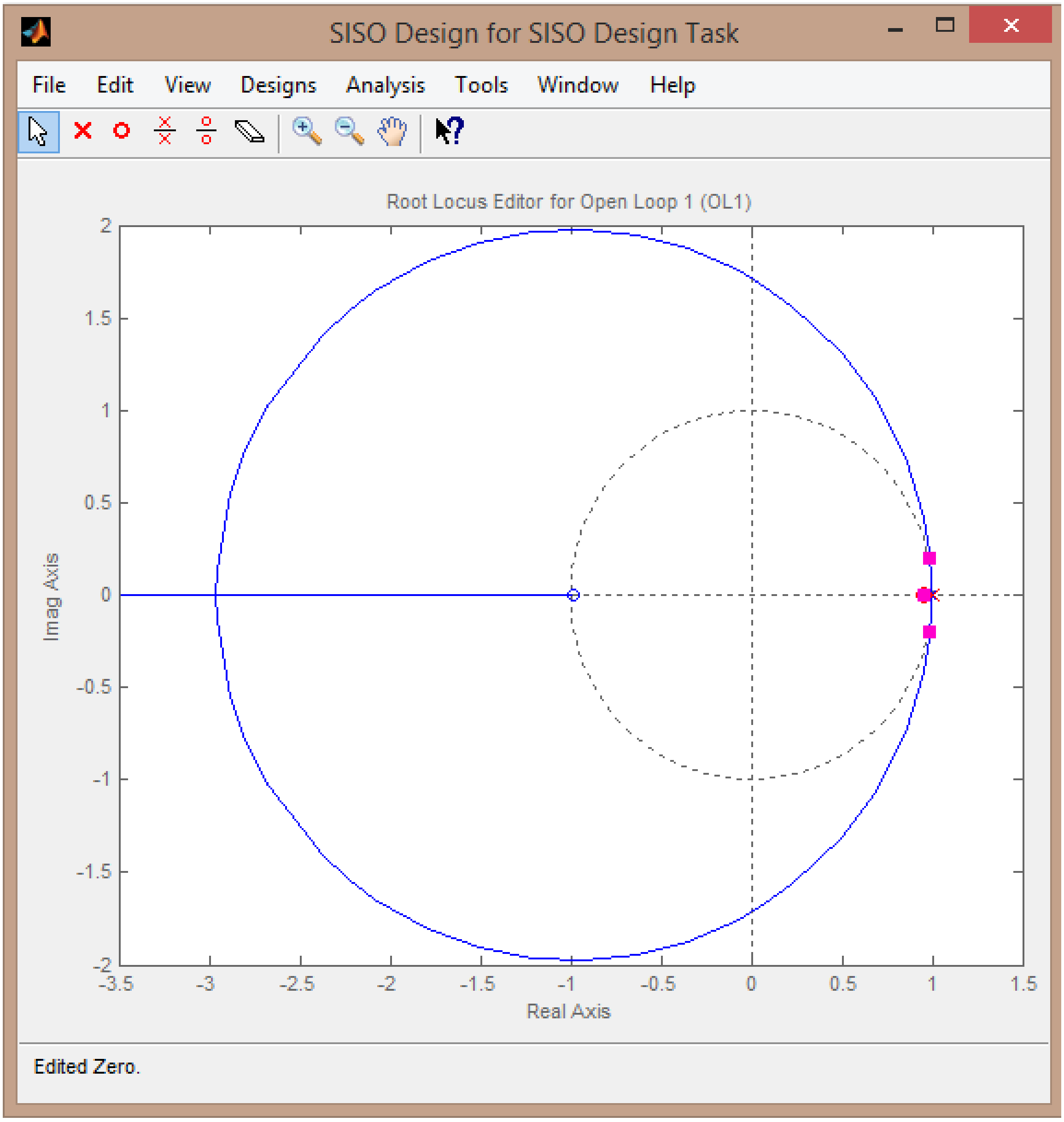
19. ábra – Zárt visszacsatolású rendszer válasza


SISO Design for SISO Design Task
File
Edit
View
Designs
Analysis
Tools
Window
Help
Root Locus Editor for Open Loop 1 (OL1)
Imag Axis
Real Axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
Right-click on the plots for more design options.

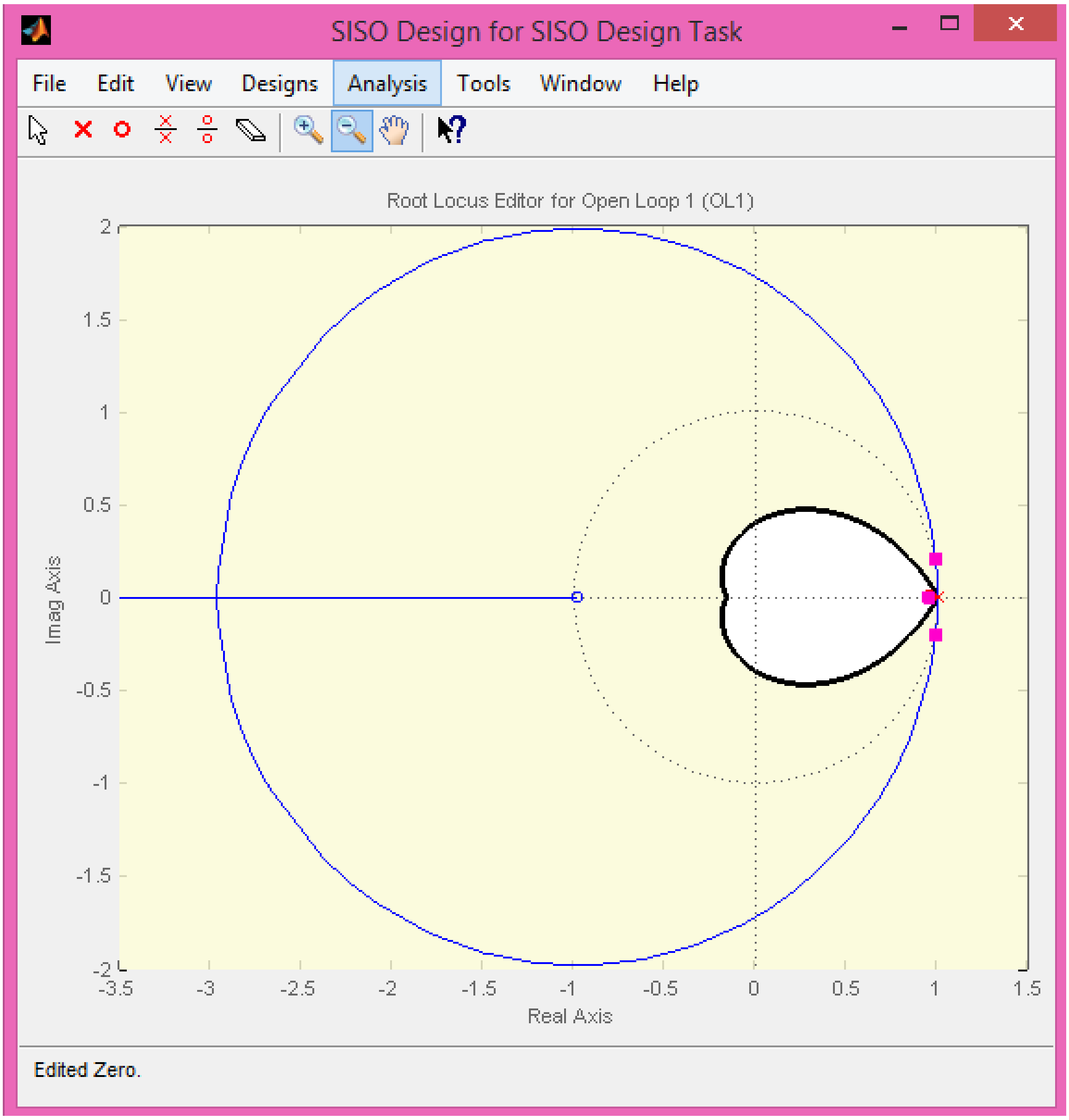
20. ábra


SISO Design for SISO Design Task
File Edit View Designs Analysis Tools Window Help
Root Locus Editor for Open Loop 1 (OL1)
Imag Axis
Real Axis
5
4
3
2
1
0
-1
-2
-3
-4
-5
-1
-0.5
0
0.5
1
1.5
2
Added Poles/Zeros.

21. ábra


SISO Design for SISO Design Task
File
Edit
View
Designs
Analysis
Tools
Window
Help
Root Locus Editor for Open Loop 1 (OL1)
Imag Axis
Real Axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
Edited Zero.

22. ábra


SISO Design for SISO Design Task
File
Edit
View
Designs
Analysis
Tools
Window
Help
Root Locus Editor for Open Loop 1 (OL1)
Imag Axis
Real Axis
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
Edited Zero.

23. ábra

Ábrajegyzék