# 

# Óbudai egyetem

Bánki Donát

Gépész és Biztonságtechnikai Mérnöki Kar

Mechatronikai Mesterképzés

Rendszer és irányítás elmélet

Egy motor pozíció szabályzása digitális szabályzóval MATLAB SIMULINK használatával

|  |
| --- |
| Tanár Dr. Pletl Szilveszter |
|  |
|  |

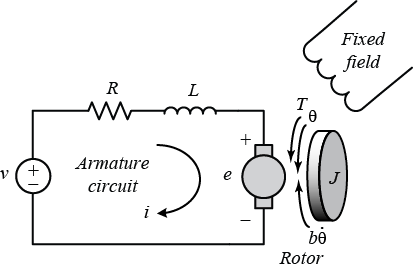
Tanuló: Bajúsz Péter

Neptún Kód: B9FVUK

# DC motor pozícionálása

## Fizikai felépítés:

A vezérlő rendszerekben a leggyakrabban használt végrehajtó szerv a DC motor. Forgó mozgást végez, amit kerekek vagy egyébb eszközök segítségével tranlációs mozgássá tudunk alakítani. A motor felépítése a következő ábrán látható (1. ábra).



1. ábra

A példa bemutatásához a motor következő fizikai paramétereit vesszük fel:

J a forgó rész tehetetlenségi nyomatéka 3.2284E-6 kg.m^2

b a motor viszkózus surlódási konstansa 3.5077E-6 N.m.s

Kb elektromos erő állandó 0.0274 V/rad/sec

Kt motor nyomaték állandó 0.0274 N.m/Amp

R villamos ellenállás 4 Ohm

L elektromos induktivitás 2.75E-6H

Feltételezzük, hogy a rendszer bemenete a motor armatúráján létrehozott feszültség külömbség (V), a kimenet pedig a tengely pozíciója théta (θ). A rotor és a tengely merevnek tekinthető, nincs csavarodás, valamint feltételezzük, hogy a surlódási nyomaték arányos a tengley szögsebességével.

## Rendszeregyenletek:

Általában a DC motor által létrehozott nyomaték arányos az armatúra áramerősségével és a mágneses tér erősségével. Ebben a példában azt feltételezzük, hogy a mágneses mező konstans, és ezért a motor nyomatéka csak az „i” armatúra áramával és Kt tényezővel arányos, amint azt az alábbi egyenlet mutatja. Ezt fegyvervezérelt motornak nevezik.

Az elektro motoros erő a tengely szögsebességével és egy Kb tényezővel arányos.

Az SI mértékegységben a motor elektro motoros ereje és nyomatéka egyenlőek, ezért a továbbiakban csak egy K együthatót használunk. A fenti ábra segítségével, felhasználva Newton 2 és Kirchof törvényét a következő egyenletet kapjuk:

### Átviteli függvény:

A fenti két egyenletet Laplace tranformálva:

A nyílt hurkú átviteli fügvényt az Is elminálásával kapjuk meg úgy, hogy a V armatúra feszültség legyen a bemenet, a kimenet pedig a szögsebesség.

A feladatban a pozíciót tekintjük kimenetnek, ezért a fenti egyenletet a sebesség integrálásával kaphatjuk meg.

### Állapot tér

A fentiektől eltérő differenciálegyenleteket az állapot-tér alakban is kifejezhetjük úgy, hogy a motor helyzetét, a motor fordulatszámát és az armatúra-áramot választjuk az állapotváltozóként. Ismét a tápfeszültséget úgy kezeljük, mint bemenetet, és a forgáspozíciót kimenetként választjuk ki.

### Követelmények:

Nagyon pontosan szeretnénk elhelyezni a motort, így a motor pozíciójának állandó állapota nulla, ha parancsot adunk. Azt is szeretnénk, hogy az egyensúlyi zavar miatt fennálló egyensúlyi hiba 0 legyen. A másik teljesítménykövetelmény az, hogy a motor nagyon gyorsan érje el a végső helyzetét túlzott túllendülés nélkül. Ebben az esetben azt szeretnénk, hogy a rendszer 40 ms alatt áljon be, és a túllendülés 16% alatt legyen.

### MATLAB PREZENTÁCIÓ:

Az átviteli függvény a MATLAB-ban:

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.

Állapot tér:

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

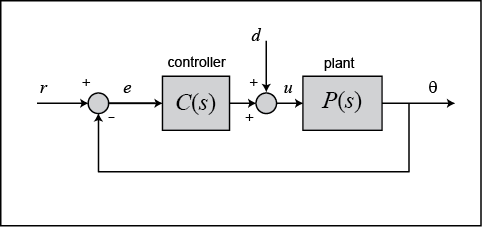
y1 0

Continuous-time state-space model.

## PID szabályzás

A nyílt hurkú átviteli függvényből megkapjuk:

A vezérlési rendszer stuktúrája a következő ábrán látható:



2. ábra

A PID szabályzáshoy először létre kell hozni egy m-fájlt MATLAB-ban ahova a következő értékeket visszük be:

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));

A PID vezérlés átviteli függvénye:

Az első lépésként a P proporcionális szabályzás értékét kell megfelelően beállítanunk, hogy eleget tegyen az előírt feltételek mindegyikének.

Kp = 1;

for i = 1:3

C(:,:,i) = pid(Kp);

Kp = Kp + 10;

end

sys\_cl = feedback(C\*P\_motor,1);

A P értékét 1 és 21 között vizsgáljuk:

t = 0:0.001:0.2;

step(sys\_cl(:,:,1), sys\_cl(:,:,2), sys\_cl(:,:,3), t)

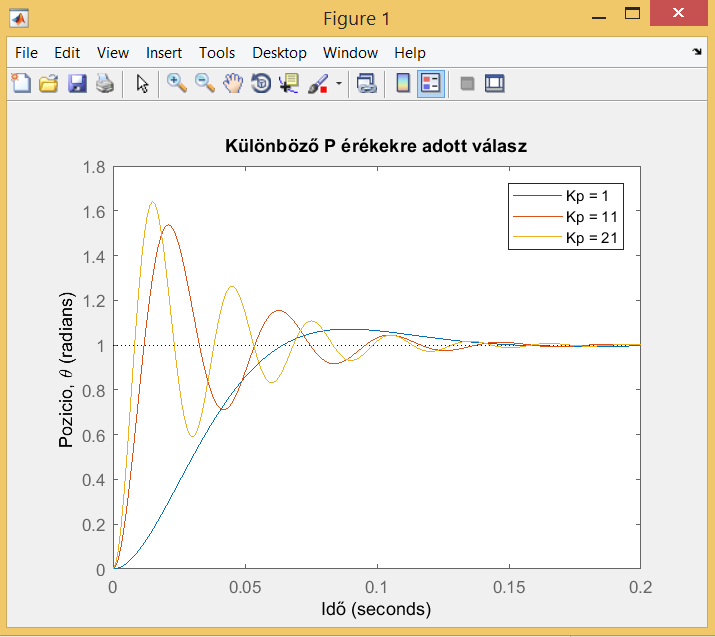
ylabel('Pozíció, \theta (radians)')

xlabel('Idő')

title('Külömböző P értékekre adott válasz')

legend('Kp = 1', 'Kp = 11', 'Kp = 21')

A következő ábrát kapjuk:



3. ábra

Az ábrán az adott értékek közül számunkra a Kp= 21 adott legmegfelelőbb választ, ezért most az I tagot Ki-t kell meghatározni.

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;

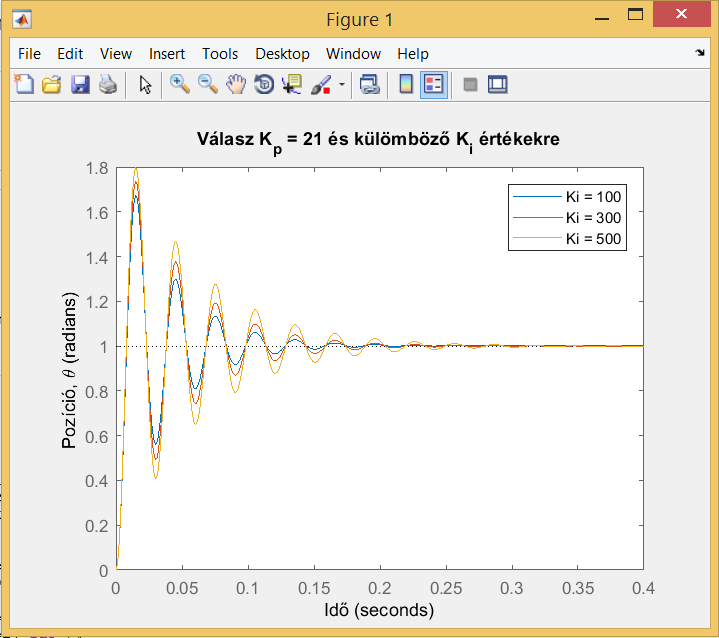
step(sys\_cl(:,:,1), sys\_cl(:,:,2), sys\_cl(:,:,3), t)

ylabel('Pozíció, \theta (radians)')

xlabel('Idő')

title('Válasz K\_p = 21 és külömböző K\_i értékekre')

legend('Ki = 100', 'Ki = 300', 'Ki = 500')



4. ábra

Most meg kell vizsgálni a zavarás értékéneek nagyságát:

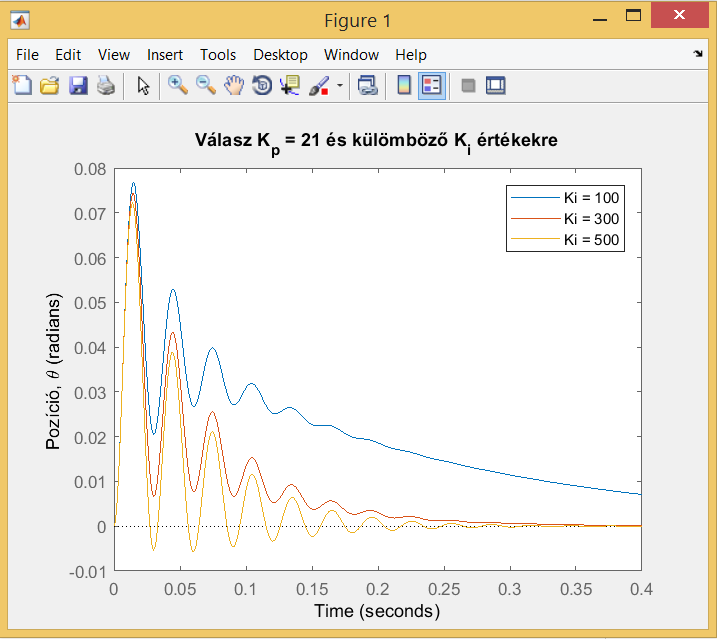
dist\_cl = feedback(P\_motor,C);

step(dist\_cl(:,:,1), dist\_cl(:,:,2), dist\_cl(:,:,3), t)

ylabel('Pozíció, \theta (radians)')

title('Válasz K\_p = 21 és külömböző K\_i értékekre')

legend('Ki = 100', 'Ki = 300', 'Ki = 500')



5. ábra

A fenti ábrából látszik, hogy a Ki=500 érték felel meg az előírt feltételeknek, mivel ezen érték reagál legjobban a hiba értékére. Az integrált vezérlés csökkentette az egyensúlyi hibát nullára, még akkor is, ha egy lépcsőzetes zavar van; ez volt a cél. Megpróbáljuk csökkenteni az ülepedési időt és a túllövést úgy, hogy derivatív vezérlést adunk hozzá a vezérlőhöz.

A Kd értékét 0,05 és 0,25 között vizsgáljuk.

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);

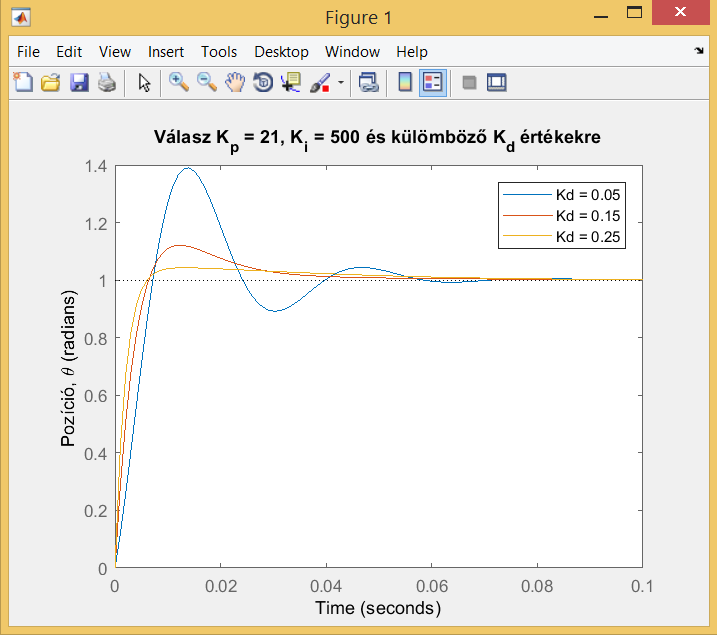
t = 0:0.001:0.1;

step(sys\_cl(:,:,1), sys\_cl(:,:,2), sys\_cl(:,:,3), t)

ylabel('Pozíció, \theta (radians)')

title('Válasz K\_p = 21, K\_i = 500 és külömböző K\_d értékekre')

legend('Kd = 0.05', 'Kd = 0.15', 'Kd = 0.25')



6. ábra

Most figyeljük meg, melyik kd értékre kapunk megfelelő hiba választ:

dist\_cl = feedback(P\_motor,C);

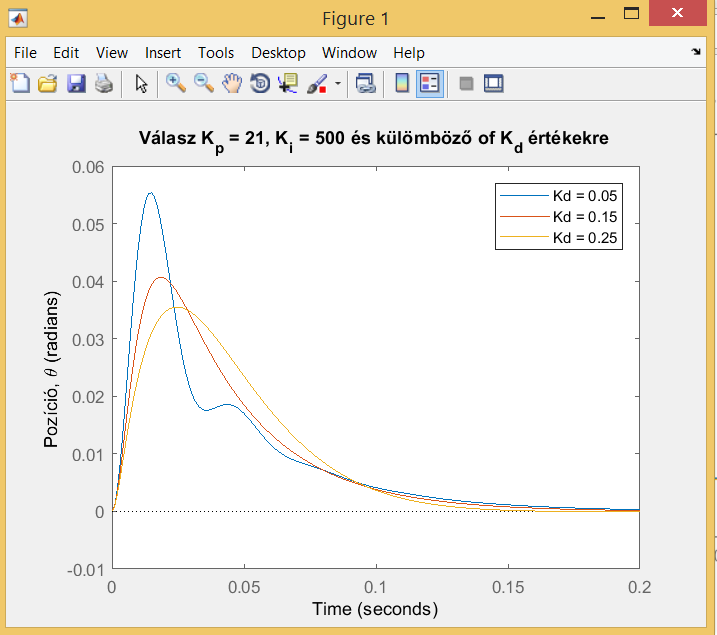
t = 0:0.001:0.2;

step(dist\_cl(:,:,1), dist\_cl(:,:,2), dist\_cl(:,:,3), t)

ylabel('Pozíció, \theta (radians)')

title('Válasz K\_p = 21, K\_i = 500 és külömböző K\_d' értékekre)

legend('Kd = 0.05', 'Kd = 0.15', 'Kd = 0.25')



7. ábra

Az ábrából leolvasható, hogy a Kd=0,15 érték ad elégíti ki legmegfelelőbben az általunk szabott feltételeket. Most nézzük meg a karakterisztikáját stepinfo segítségével:

stepinfo(sys\_cl(:,:,2))

ans =

struct with fields:

RiseTime: 0.0046

SettlingTime: 0.0338

SettlingMin: 0.9103

SettlingMax: 1.1212

Overshoot: 12.1175

Undershoot: 0

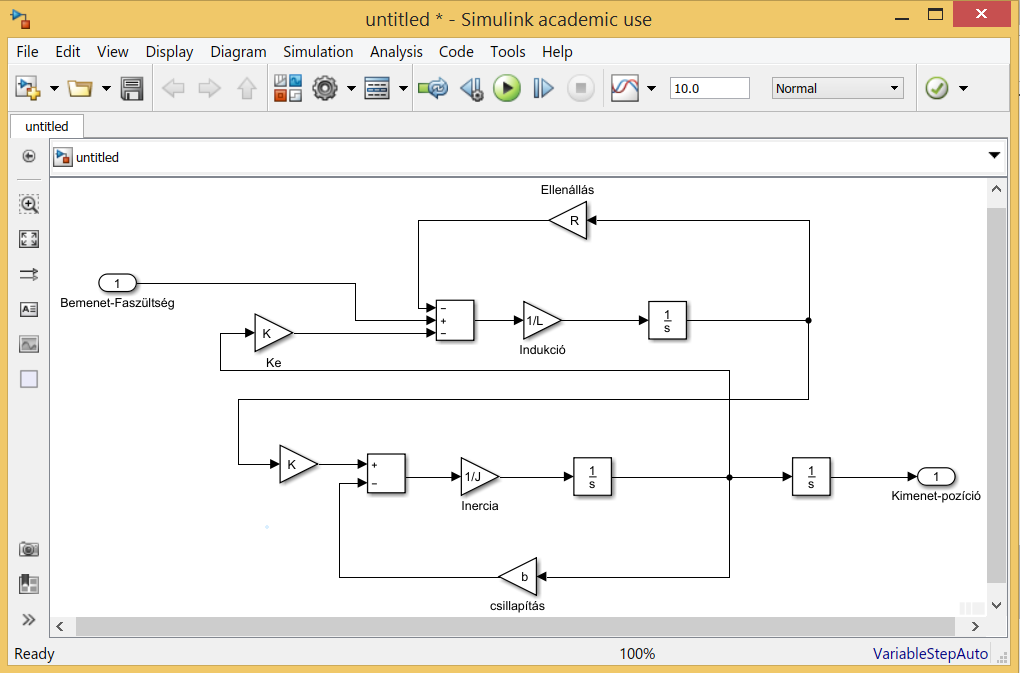
Peak: 1.1212

PeakTime: 0.0122

Innen látszik, hogy minden feltételünk teljesült, a rendszer 34 ms alatt beáll és a túllendülés 12% os. Tehát a motor pozíció szabalyzásának PID értékei: Kp=21, Ki=500 és Kd=0,15.

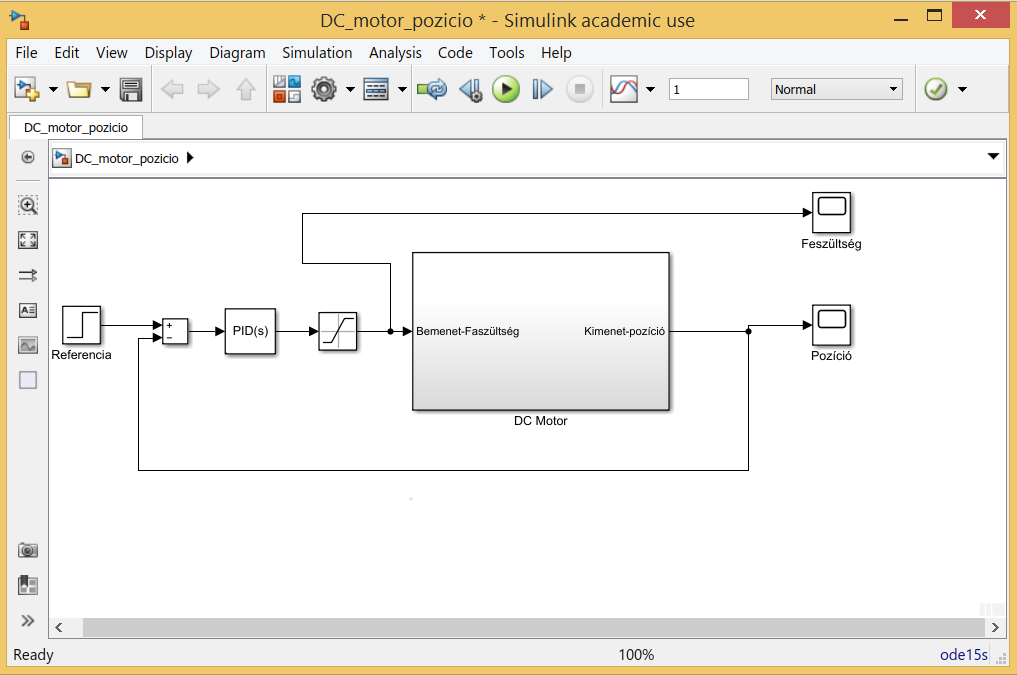
## Megvalósítás SIMULINKBEN

A motor SIMULINK felépítése:



8. ábra

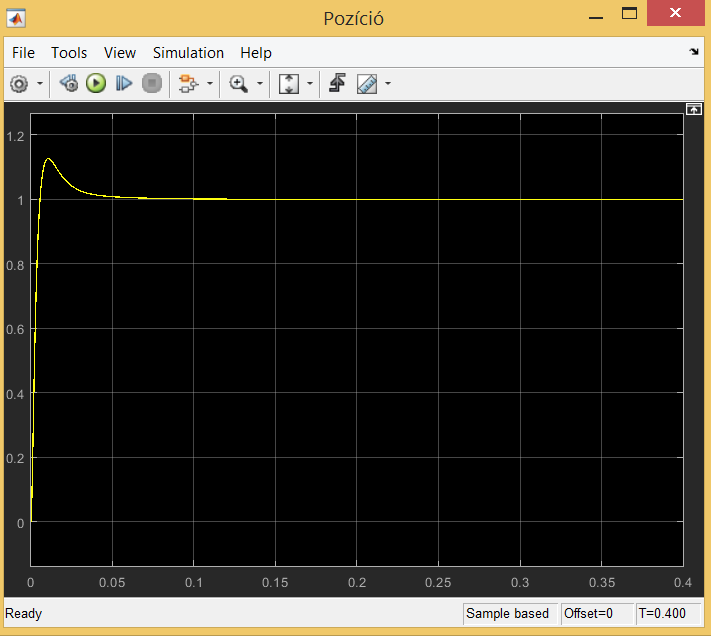
A motor PID szabályzós SIMULINK felééítése:



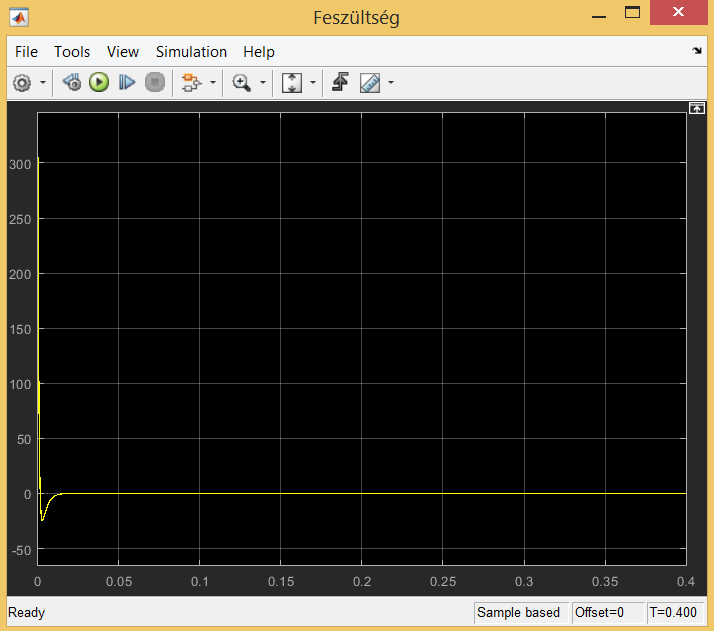
9. ábra

A szabályzóba még beépítettem egy szatiráció szabályzót, hogy a motor ne legyen túlvezérleve. Ennek következményeként a szabályzás nem áll be időben.

A kimenet értéke szaturáció nélkül:



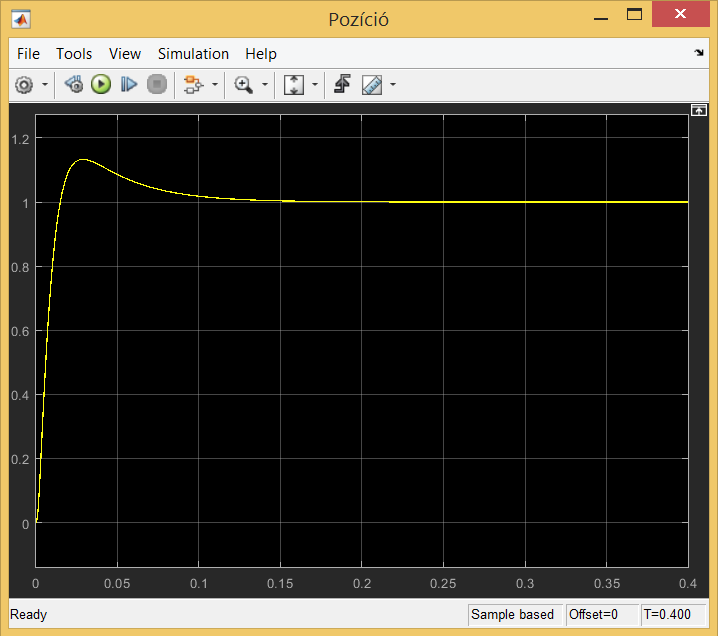
10. ábra



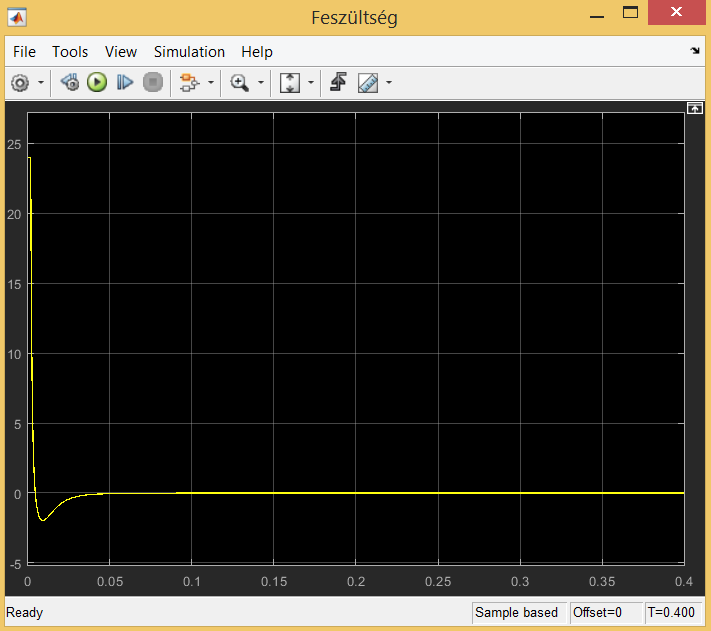
11. ábra

Az ábrából leolvasható, hogy a kezdeti állapotban, 300 V fölé is felugrik a bemeneti érték, de ezt egy 24 V os motor nem bírná el, ezért van szükség szaturációra.

A kimenet értéke szaturációval:

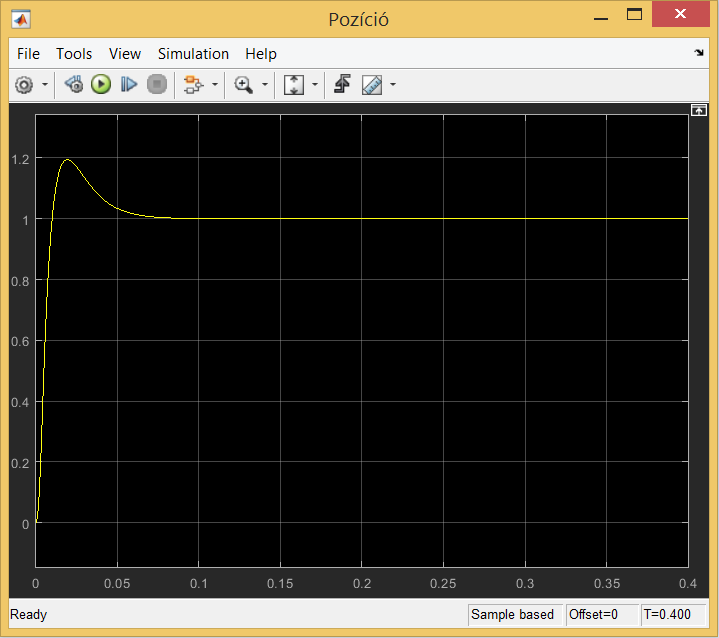


12. ábra



13. ábra

A szaturációval sikerült a motor túlfeszültség védelme, viszont az időállandónk elcsúszott, az szaturáció nélküli 34 ms-os beállás helyett közel 200 ms-ra van szüksége a rendszernek, hogy beáljon, ezért újra kell hangolni a PID szabályzó értékeit.



14. ábra

Hiába a PID szabályzó maximális hangolása, szaturációval a kívánt feltételeket nem tudjuk teljesíteni.

## Digitális szabályzás:

### PID-el

Ahhoz, hogy a DC motort digitálisan szabályozni tudjuk, szükség van az átviteli függvény diszkrét alakjára, amit a következő képen kapunk meg:

P\_motor

P\_motor =

0.0274

-------------------------------------------

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

Continuous-time transfer function.

>> T0=0.01

T0 =0.0100

DP\_motor=c2d(P\_motor,0.01)

DP\_motor =

0.08791 z^2 + 0.07221 z + 5.548e-10

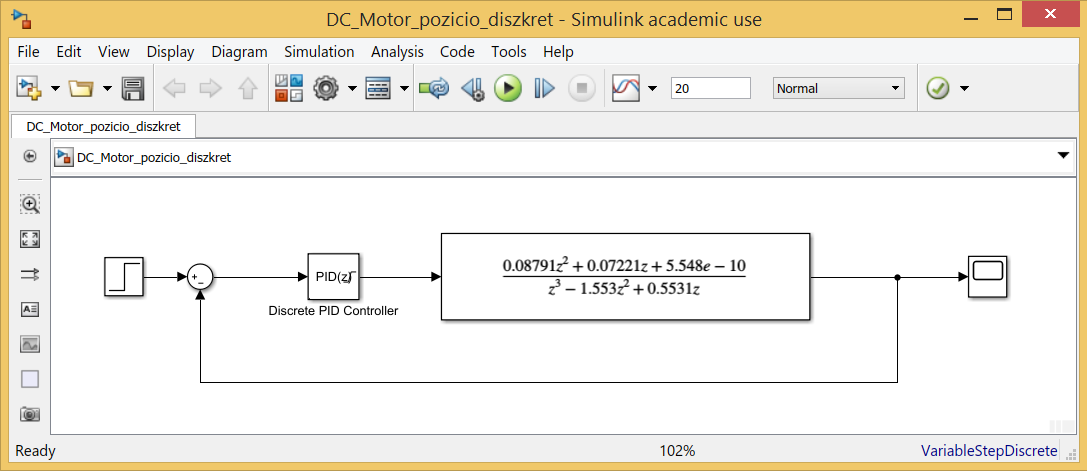
-----------------------------------

z^3 - 1.553 z^2 + 0.5531 z

Sample time: 0.01 seconds

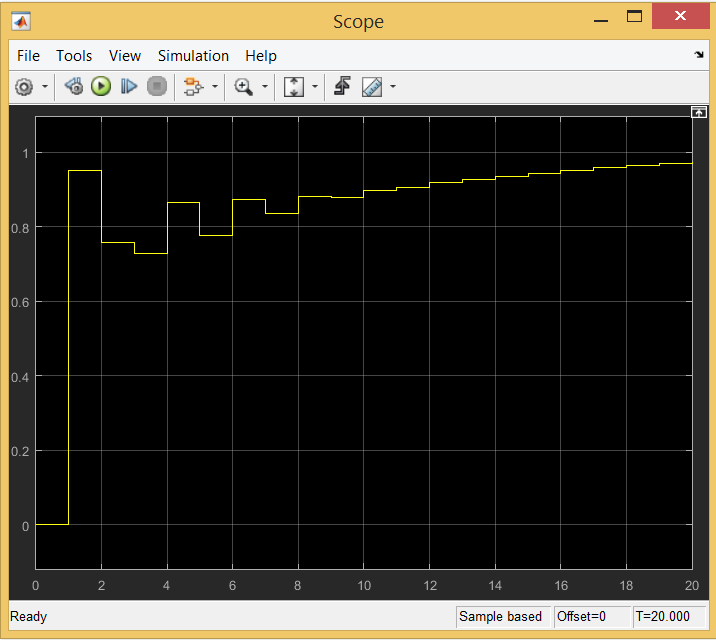
Discrete-time transfer function.

Ezt a diszkretizált átviteli fügvény, a következő képen használtam fel MATLAB SIMULINK-ben:



15. ábra

A PID szabályzó hangolása után, a következő eredményeket kaptam:



16. ábra

Ebben a feladatban a szaturációt magába a szabályzóba építettem bele. A diszkretizáció végett, késés lép fel (1 s) és a PID szabályzót sem lehet, úgy hangolni, hogy a megadott feltételeknek elegedet tegyen. Ezt a rendszert nem lehet megfelelően PID szabályzással vezérelni, ezért a megvalósításához vagy más vezérlési feltételeket kell szabni, vagy pedig más vezérlési metódust alkalmazni.

## Diszkretizált digitális szabályzás

A digitális LTI (Linearizált idő invariáns) vezérléshez szükségünk van a rendszer állapot tér egyenleteire:

DD=c2d(motor\_ss, 0.01)

DD =

A =

x1 x2 x3

x1 1 0.007546 4.403e-05

x2 0 0.5531 0.003227

x3 0 -0.003789 -2.211e-05

B =

u1

x1 0.08791

x2 16.01

x3 0.1403

C =

x1 x2 x3

y1 1 0 0

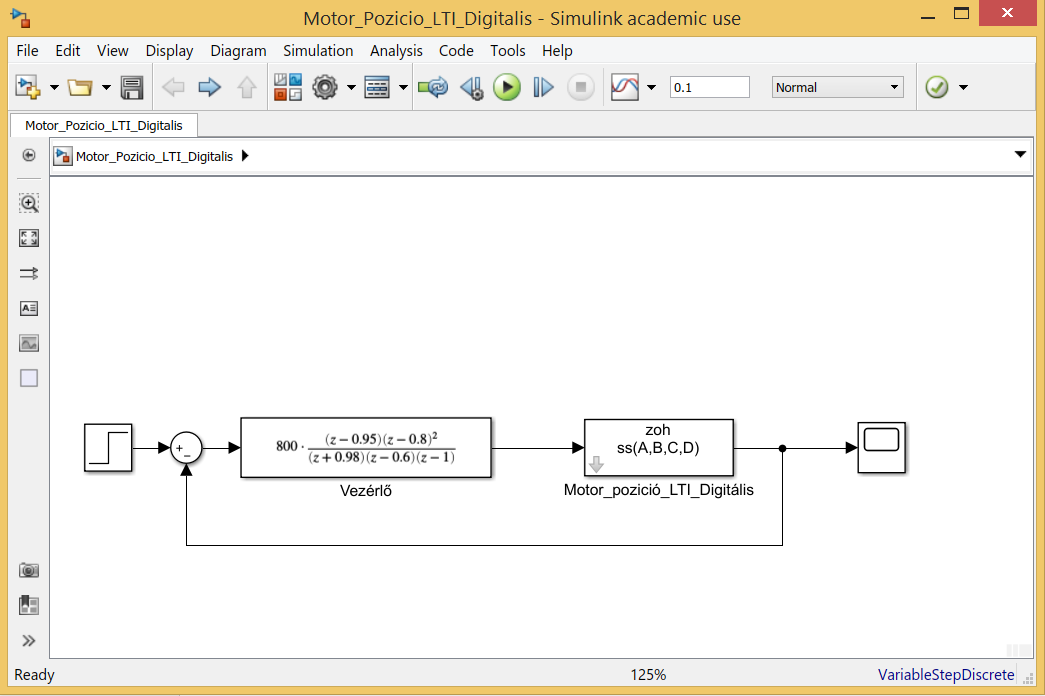
D =

u1

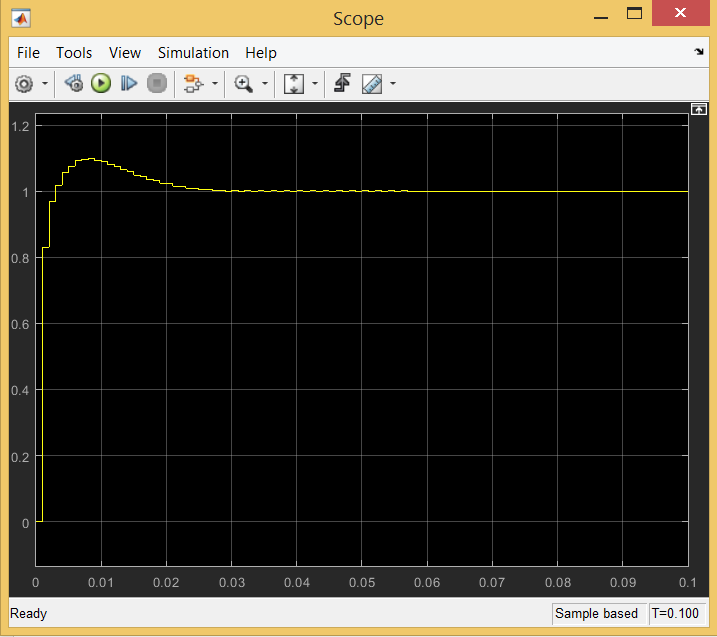
y1 0

Sample time: 0.01 seconds

A fenti képleteket és állapot tér elemeket felhasználva a SIMULINK:



17. ábra



18. ábra

Az ábárn jól látható, hogy a digitális szabályzással eleget tettünk a feltételek mindegyikének.