A 2008. január 7-én, 10-12 között megtartott Szabályozástechnika konzultáció anyaga (a konzultációt Kiss Bálint tartotta)

A leírtak helyességéért a felelősséget nem tudom vállalni, de remélem azért hasznotokra válik.

Mintavizsga 1. feladatának megoldása (2-DOF tervezés)

A feladat 1/s^2, ennél egyszerűbbet nem nagyon lehet adni, esetleg gyengén csillapított kéttárolós lengőtag is lehet. (1/(s^2+0.1s+1))
A feladat szerint a szabályzó tartalmaz integrátort, ezért beleteszünk egyet, l=1.
T= 1 sec adott volt.

s1 és s2 számítható az adott ksi és wo értékekből. Soinf szintén adott.

1.,
Ekvivalens specifikáció előállítása diszkrét időben:
S1 és s2 átszámítása z1 és z2-re, soinfet is áttesszük zoinfbe. (z=e^sT)
//ebben a feladatban T=1, tehát nem felejtik el a hallgatók beírni… ☺

Ezzel kész is a diszkrétidejű specifikáció. A lapra le kell írni s1,s2 számítását, valamint a z-re megkapott értékeket.

2.,
Megadjuk a szakaszt.
nums=1; dens=[1 0 0]; [B,A]=c2dm(nums,dens,1,'zoh');
a kapott eredményeket felírjuk a papírunkra!
A számlálót fel kell bontanunk → ehhez kiszámoljuk a számláló gyökeit. Az egyetlen gyök most -1. Ez nem kiejthető. Bplus=1; Bminus=B;

3.
Bm/Am=Bmv*Bminus/Am → Bmv konstans!
Fokszámfeltételek megtalálhatóak a könyvben!

Visszatérés az első feladathoz:
Ha megvan Am és Ao fokszáma, ezeket legyártjuk. Ao-ba kerül zoinf, Ac-be pedig a domináns pólusok.
A kapott értékeket a papírra vetjük.
P=Am és Ao szorzata, tehát ezt convval csináljuk meg.

A harmadik feladatrészben kért referencia modellhez szükséges Bm' meghatározása is.
Bm'*Bminus(1)=Am(1) → Bm'=Am(1)/Bminus(1);
A Bm'-re megkapott eredményt is papírra vetjük. ☺

4., Diophantosi egyenlet felírása.
A: szakasz diszkrétidejű átviteli függvényének nevezője.
R1v még nem ismert, de fokszáma adott. S még nem ismert, de fokszáma adott.

A*(z-1)*R1v + Bminus*S = P/(Am*Ao)

R1' elsőfokú, ezért R1'(z)=z+r0
S másodfokú, ezért S(z)=s0*z^2+s1*z+s2;

$$\begin{bmatrix} 1 & 0.5 & & \\ -3 & 0.5 & 0.5 & \\ 3 & & 0.5 & 0.5 \\ -1 & & & 0.5 \end{bmatrix} \begin{bmatrix} r1 \\ s0 \\ s1 \\ s2 \end{bmatrix} = \begin{bmatrix} -1.9896+3 \\ 1.2373-3 \\ -0.2355+1 \\ 0.0138 \end{bmatrix}$$

A jobb oldalon p együtthatói szerepelnek, de a vezető egyest ki kell szedni!!!
A kapott vektorból ki kell vonnunk a nagy mátrix első oszlopának együtthatóit, de ott is elhagyjuk a vezető egyest!
Az oszlopokat Matlabban is összerakjuk.

```
>> M=[AA' [0.5 0 0; 0.5 0.5 0; 0 0.5 0.5; 0 0 0.5]]
>> C=P(2:end)'-[AA(2:end)';0]
```

Az egyenlet megoldását megkapjuk.

```
>> rs=inv(M)*C;
>> rs

rs =

    0.4405
    1.1398
   -2.0224
    0.9086
```

R1'=z+0.4405
s=1.139z^2 - 2.0224z + 0.9086

Számítanunk kell R-et. R=R1'(z-1)Bplus

```
>> R=conv([1 rs(1)],[1 -1])
R =

    1.0000   -0.5595   -0.4405
```

5. feladat
A szabályozó összefüggése a feladatban adott. Ru=Tr-Sy.
Ebből már R-et és S-et meghatároztuk, T=Bm'*Ao
T =

   0.0347   -0.0094   0.0006
Ru visszatranszformálva időtartományban:

u[k]-0.5595u[k-1]-0.4405u[k-2] =
=0.034r[k] – 0.0094 r[k-1] + 0.0006r[k-2]-1.1398y[k]+2.0224y[k-1]+0.9086y[k-2]

A beavatkozójel számításához szükséges képlet.

Blokkséma!!!
ITT NEM MATLAB KÓDOT KELL ODAÍRNI, HANEM A FENTI SZÁMÍTÁST.

6. feladat
Dyr(z) eredő átviteli függvény.
A modellünk átviteli függvénye a Bm(z)/Am(z), már mindkettőt megkaptuk.
Dyr(z)=Bm'*Bminus/Am
>> Bm=Bmv*Bminus

Bm =

       0   0.0174   0.0174

>> Am

Am =

   1.0000   -1.7189   0.7537

Dyr(z)=0.0174z+0.0174/z^2-1.7189z+0.7537

A pólusok, zérusok, gyökök azokat, amiket előírtunk. A túllövés itt a step utasítás nélkül is meghatározható, ennek a túllövése annak a kéttárolós lengőtagnak a túllövése lesz, amelynek wo-ja és ksi-je adott.

Meghatározás steppel:
dstep(Bm,Am)

a szükséges idők leolvashatóak! vagy használhatjuk a kéttárolós lengőtagra vonatkozó képleteket.
A MATLAB által kiírt Settling time-ot meg kell szorozni a mintavételi periódusidővel, ez itt pont 1.
Peak response-t kiválasztva megkapjuk az overshoot értéket is.

Nehezítések / könnyítések

- magasabb fokszámú rendszer, a mátrix nagyobb lehet
- lehet olyan másodfokú rendszert megadni, ahol szükség van scinf pólusra
- lehet variálni azzal, hogy van-e szükség integrátorra

2. feladat

A feladat diszkrétidejű állapotteres szabályozó tervezése. A szakasz szintén egy kettős integrátor.

5. rész:
valósidejű szempontok figyelembevétele: két lépcsőre bontjuk:
a kimenet megérkezése előtt elkezdhető számítások, a kimenet megérkezésekor → eszerint felbontjuk

állapotvisszacsatolásnál előírtuk a póluspárokat, tehát a megadott ksi és wo adja ezeket a pólus-zérus értékeket.

A feladat megoldása végig:

W(s)=1/s^2. ksi és wo itt is adott az állapotvisszacsatoláshoz. A megfigyelő sajátértékei is meg vannak adva.

1., feladat
Állapotegyenlet meghatározása folytonos időben, szabályozó alakban. → tf2ss használata.
Ez a folytonosidejű állapotegyenlet mátrixait adja meg. Meg kell adnunk neki a számlálót és a nevezőt.

```
>> [A,B,C,D]=tf2ss(1,[1 0 0])

A =

     0     0
     1     0


B =

     1
     0


C =

     0     1


D =

     0
```

Folytonos időben megkaptuk az állapotegyenleteket.

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

$$y = Cx + 0u$$

Diszkrétidejű megfeleltetés:
plant=ss(A,B,C,D) → folytonosidejű szakasz
Ezt átalakítjuk diszkrét időbe (c2d, 0.1s mintavételezéssel, 'zoh'-hal)
A későbbiekben hivatkoznunk kell a mátrixokra, ezért ezeket ebből a struktúrából kiszedjük. (ssdata)

```
>> plantd=c2d(plant,0.1,'zoh')
>> [Ad,Bd,Cd,Dd]=ssdata(plantd)
```

Az attérés a C és D mátrixokat nem befolyásolja, a többit a Matlab megadta.

```
>> [Ad,Bd,Cd,Dd]=ssdata(plantd)

Ad =

    1.0000         0
    0.1000    1.0000


Bd =

    0.1000
    0.0050


Cd =

     0     1


Dd =

     0
```

2., feladat.
Diszkrétidejű specifikációk meghatározása, karakterisztikus egyenletek megadása.
A fokszámok itt nem kérdésesek. (2, mivel a zrke…)

```
>> wo=2;
>> ksi=0.7;
>> soinf=-4;

>> s1=-wo*ksi+j*wo*sqrt(1-ksi^2)

s1 =

  -1.4000 + 1.4283i

>> s2=conj(s1)

s2 =

  -1.4000 - 1.4283i
```

Ezeket áttranszformáljuk z-be.

```
>> Ts=0.1;
>> z1=exp(s1*Ts); z2=exp(s2*Ts); zoinf=exp(soinf*Ts);
>> z1

z1 =

  0.8605 + 0.1237i

>> z2

z2 =

  0.8605 - 0.1237i

>> zoinf

zoinf =

   0.6703
```

Fel kell írnunk a z.r.k.e-t és a megfigyelő karakterisztikus egyenletét.
fiC gyökei: z1, z2.
fiC(z)=(z-z1)(z-z2)
fiO gyökei: zoinf (másodfokú rendszerhez tervezzük! ha nagyobb rendszer lenne, nagyobb hatványon lenne)
[ha nagyobb rendszer lenne, akkor z1 és z2 mellé egy zcinf-et is be kellene venni fiC-be]
fiO(z)=(z-zoinf)^2

A kapott értékeket a matlabból átírjuk.

3., feladat
Irányíthatósági mátrixát és a K mátrix meghatározása Ackermann képlettel.
Az irányíthatósági mátrixot Mc-vel jelöljük. Általánosan: B, AB, …. A^(n-1)B
Mivel itt n=2 → Mc=[B AB];
ctrb utasítás: előállítja az irányíthatósági mátrixot (controllability)
ctrb nélkül: Mc=[Bd, Ad*Bd]

Ackermann képlettel meg kell határoznunk a K-t.
Használhatjuk az acker utasítást, vagy a képletet valósítjuk meg.
Az acker utasításhoz meg kell adnunk a zárt körbe előírt sajátértékeket!!! z1 és z2

```
>> K=acker(Ad,Bd,[z1 z2])

K =

   2.6160    3.4772
```

Nx és Nu meghatározása is feladat volt.

Figyelni kell, hogy folytonos vagy diszkrét idejű rendszerre végezzük a tervezést, mert az egyenlet különbözik a két esetben!

$$\begin{bmatrix} Ad - I & Bd \\ Cd & 0 \end{bmatrix} \begin{bmatrix} Nx \\ Nu \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Egységmátrix: eye utasítással.

[0; 0; 1]
0 mérete nxm-es…

```
>> N=inv([Ad-eye(2,2) Bd; Cd 0])*[0; 0; 1]

N =

    0
    1
    0
```

Nx az első két elem, [0;1], Nu=0; (Nu mindig skalár SISO rendszerben.)

4., Aktuális állapotmegfigyelő differenciaegyenlete.

xkkalap=F xkalap[k-1] + G y[k] + H u[k-1]
(y aktuális értékét használjuk fel)
Attól aktuális, hogy az aktuális kimenettől függ, amit az előző állapotok becslője alapján határoz meg.

H=Bd-GCdBd (ha nem aktuális lenne a megfigyelő, akkor H=Bd)
F=Ad-GCdAd

G meghatározása Ackermann képlet segítségével. Az F mátrix sajátértékei az előírt **fio(z) polinom gyökei** legyenek. (Nem határozhatjuk meg közvetlenül, hanem a transzponáltakat kell használni.)

F'=Ad'-Ad'Cd'G'
(a transzponálásra nézve a sajátértékek invariánsak)

```
>> Gt=acker(Ad',Ad'*Cd',[zoinf zoinf])

Gt =

   1.869      0.5507
```

A G ennek transzponáltja lesz.
>> G=Gt'

G =

1.0869
0.5507

Mivel az összes mátrixot kérték, ezért a H és F mátrixok számítása is szükséges.

>> H=Bd-G*Cd*Bd

H =

0.0946
0.0022

>> F=Ad-G*Cd*Ad

F =

0.8913 -1.0869
0.0449 0.4493

Bonyolíthatóság: nem elég akt. megfigyelés tervezése, hanem egy terhelésbecslőt is bele kellhet tenni a megfigyelőbe, ebben az esetben az F mátrix 3x3-as lesz, a H és G oszlopvektorok is megnagyobbodnak.
Integráló szabályozás esetében az ackermann képlet módosul, amellyel a K meghatározására szolgál, ebbe bele kell venni az integrátor visszacsatolást.
Ha integrátor van a szabályozóban, akkor nincs szükség Nu-ra.

„Dobjunk ehhez egy terhelésbecslőt!"

4., másik lehetséges megoldása terhelésbecsléssel
Bevezetünk egy új állapotot, xd,k=a terhelés értéke. Feltesszük, hogy ez konstans, tehát deriváltja 0.

$$\begin{bmatrix} xk+1 \\ xdk+1 \end{bmatrix} = \begin{bmatrix} Ad & Bd \\ 0 & 1 \end{bmatrix} \begin{bmatrix} xk \\ xdk \end{bmatrix} + \begin{bmatrix} Bd \\ 0 \end{bmatrix} uk$$

$$yk = [Cd \quad 0] \begin{bmatrix} xk \\ xdk \end{bmatrix}$$

Bővített állapotegyenlet, a mátrixokat hullámokkal jelöljük.
Ugyanezt az aktuális megfigyelőt tervezzük, de a bővített rendszerre. Ez ennek a bővített rendszernek az állapotát fogja becsülni, ez megadja a szakasz állapotait (becslését), és a zavarásnak a becslését is.

$$\begin{bmatrix} xkkalap+1 \\ xdkkalap+1 \end{bmatrix} = \tilde{F} \begin{bmatrix} xk-1 \\ xdk-1 \end{bmatrix} + \tilde{G}yk + \tilde{H}uk-1$$

Hh=Bdh-GhCdhBdh
Fh=Adh-GhCdhAdh

Annyi 0 van, ahány oszlopa van Ad-nek, tehát most 2. A mátrix 3x3 lesz.

```
>> Adh=[Ad Bd; 0 0 1;]

Adh =

    1.0000         0    0.1000
    0.1000    1.0000    0.0050
         0         0    1.0000

>> Bdh=[Bd; 0]

Bdh =

    0.1000
    0.0050
         0

>> Cdh=[Cd 0]

Cdh =

     0     1     0

>> Ght=acker(Adh',Adh'*Cdh',[zoinf zoinf zoinf])

Ght =

    2.7232    0.6988    3.5833
```

még egyszer be kell vennünk zoinf-et, mivel eggyel nagyobb lett a mátrix.

```
>> Gh=Ght'

Gh =

    2.7232
    0.6988
    3.5833


>> Hh=Bdh-Gh*Cdh*Bdh

Hh =

    0.0864
    0.0015
   -0.0179
```

```
>> Fh=Adh-Gh*Cdh*Adh

Fh =

    0.7277   -2.7232    0.0864
    0.0301    0.3012    0.0015
   -0.3583   -3.5833    0.9821
```

5., feladat
A szabályozó hatásvázlatának felrajzolása.

Rajz.

A valósidejű szempontok figyelembevételével két blokkra bontjuk ezt szét, ez a 306. oldalon benne van a könyvben.
Realizálás:

Terhelésbecslős hatásvázlat: 11.19 ábra

A Simulink modellt most nem valósítjuk meg, a túllövés és az első maximum eléréséhez idő meghatározható a wo és ksi felhasználásával, ezek benne vannak a kéttárolós lengőtagos leírásban.

Stabilitás, Bode-k, mintavételezés (nyquist), szabályozótervezés (állapotteres és nem állapotteres), utolsó anyagrész (identifikációs modellek, költségfüggvény … 😐)

MATLAB KÓD:

```
  To get started, select MATLAB Help or Demos from the Help menu.

>> ksi=0.707; wo=0.2;
>> s1=-wo*ksi+j*wo*sqrt(1-ksi^2)

s1 =

  -0.1414 + 0.1414i

>> s1=-wo*ksi-j*wo*sqrt(1-ksi^2)

s1 =

  -0.1414 - 0.1414i

>> s1=-wo*ksi+j*wo*sqrt(1-ksi^2)

s1 =

  -0.1414 + 0.1414i

>> s2 = -wo*ksi-j*wo*sqrt(1-ksi^2);
>> s2 = -wo*ksi-j*wo*sqrt(1-ksi^2)

s2 =

  -0.1414 - 0.1414i

>> soinf=-2;
>> z1 = exp(s1)

z1 =

   0.8595 + 0.1224i

>> z2=exp(s2)

z2 =

   0.8595 - 0.1224i

>> zoinf=exp(soinf)

zoinf =

    0.1353

>> nums=1; dens=[1 0 0]; [B,A]=c2dm(nums,dens,1,'zoh')
```

```
B =

        0    0.5000    0.5000


A =

     1    -2     1

>>
>> roots(B)

ans =

    -1

>> Bminus=B

Bminus =

        0    0.5000    0.5000

>> Bplus=1

Bplus =

     1

>> Am=poly([z1 z2])

Am =

    1.0000   -1.7189    0.7537

>> Ao=poly([zoinf zoinf])

Ao =

    1.0000   -0.2707    0.0183

>> P=conv(Am,Ao)

P =

    1.0000   -1.9896    1.2373   -0.2355    0.0138

>> Bmv=polyval(Am,1)/polyval(Bminus,1)

Bmv =
```

```
    0.0347

>> %másképp:
>> Bmv=sum(Am)/sum(Bminus)

Bmv =

    0.0347

>> AA=conv(A,[1 -1]);
>> AA

AA =

     1    -3     3    -1

>> Bminus

Bminus =

         0    0.5000    0.5000

>> P

P =

    1.0000   -1.9896    1.2373   -0.2355    0.0138

>> M=[AA' [0.5 0 0; 0.5 0.5 0; 0 0.5 0.5; 0 0 0.5]]

M =

    1.0000    0.5000         0         0
   -3.0000    0.5000    0.5000         0
    3.0000         0    0.5000    0.5000
   -1.0000         0         0    0.5000

>> C=P(2:end)'-[AA(2:end)';0]

C =

    1.0104
   -1.7627
    0.7645
    0.0138

>> rs=inv(M)*C;
>> rs

rs =
```

```
    0.4405
    1.1398
   -2.0224
    0.9086

>> R=conv([1 rs(1)],[1 -1])

R =

    1.0000   -0.5595   -0.4405

>> T=Bmv*Ao;
>> T

T =

    0.0347   -0.0094    0.0006

>> Bm=Bmv*Bminus

Bm =

         0    0.0174    0.0174

>> Am

Am =

    1.0000   -1.7189    0.7537

>> dstep(Bm,Am)
>> tf2ss([1],[1 0 0])

ans =

     0     0
     1     0

>> [A,B,C,D]=tf2ss(1,[1 0 0])

A =

     0     0
     1     0


B =

     1
     0
```

```
C =

     0     1


D =

     0

>> plant=ss(A,B,C,D)

a =
       x1  x2
  x1   0   0
  x2   1   0


b =
       u1
  x1   1
  x2   0


c =
       x1  x2
  y1   0   1


d =
       u1
  y1   0

Continuous-time model.
>> plantd=c2d(plant,0.1,'zoh')

a =
        x1   x2
  x1     1    0
  x2   0.1    1


b =
         u1
  x1     0.1
  x2   0.005


c =
       x1  x2
  y1   0   1
```

```
d =
      u1
  y1   0

Sampling time: 0.1
Discrete-time model.
>> [Ad,Bd,Cd,Dd]=ssdata(plantd)

Ad =

    1.0000         0
    0.1000    1.0000


Bd =

    0.1000
    0.0050


Cd =

     0     1


Dd =

     0

>> wo=2;
>> ksi=0.7;
>> soinf=-4;
>> s1=-wo*ksi+j*wo*sqrt(1-ksi^2)

s1 =

  -1.4000 + 1.4283i

>> s2=conj(s1)

s2 =

  -1.4000 - 1.4283i

>> Ts=0.1;
>> z1=exp(s1*Ts); z2=exp(s2*Ts); zoinf=exp(soinf*Ts); zcinf=exp(scinf*Ts);
??? Undefined function or variable 'scinf'.

>> z1=exp(s1*Ts); z2=exp(s2*Ts); zoinf=exp(soinf*Ts);
```

```
>> z1

z1 =

   0.8605 + 0.1237i

>> z2

z2 =

   0.8605 - 0.1237i

>> zoinf

zoinf =

    0.6703

>> phic=poly([z1 z2])

phic =

    1.0000   -1.7210    0.7558

>> phio=poly([zoinf zoinf])

phio =

    1.0000   -1.3406    0.4493

>> Mc=ctrb(Ad,Bd)

Mc =

    0.1000    0.1000
    0.0050    0.0150

>> K=acker(Ad,Bd,[z1 z2])

K =

    2.6160    3.4772

>> N=inv([Ad-eye(2,2) Bd; Cd 0])*[0; 0; 1]

N =

     0
     1
     0
```

```
>> Gt=acker(Ad',Ad'*Cd',[zoinf zoinf])

Gt =

    1.0869    0.5507

>> G=Gt'

G =

    1.0869
    0.5507

>> H=Bd-G*Cd*Bd

H =

    0.0946
    0.0022

>> F=Ad-G*Cd*Ad

F =

    0.8913   -1.0869
    0.0449    0.4493

>> Adh=[Ad Bd; 0 0 1;]

Adh =

    1.0000         0    0.1000
    0.1000    1.0000    0.0050
         0         0    1.0000

>> Bdh=[Bd; 0]

Bdh =

    0.1000
    0.0050
         0

>> Cdh=[Cd 0]

Cdh =

     0     1     0

>> Ght=acker(Adh',Adh'*Cdh',[zoinf zoinf zoinf])
```

```
Ght =

    2.7232    0.6988    3.5833

>> Gh=Ght'

Gh =

    2.7232
    0.6988
    3.5833

>> Hh=Bdh-G*hCd*Bd
??? Undefined function or variable 'hCd'.

>> Hh=Bdh-Gh*hCd*Bd
??? Undefined function or variable 'hCd'.

>> Hh=Bdh-Gh*Cdh*Bdh

Hh =

    0.0864
    0.0015
   -0.0179

>> Fh=Adh-Gh*Cdh*Adh

Fh =

    0.7277   -2.7232    0.0864
    0.0301    0.3012    0.0015
   -0.3583   -3.5833    0.9821
```