TU LIBEREC
Hálkova 6
461 17 Liberec 1, CZ

HOCHSCHULE
ZITTAU/GÖRLITZ (FH)
University of Applied Science
Theodor-Körner-Alle 16, D-02763
Zittau

Fakulta mechatroniky a
mezioborových inženýrských studií

Fachbereich Elektro – und
Informationstechnik

MATLAB
Anwendung in der Regelungstechnik

Arbeitsversion

Doc. Ing. Osvald Modrlák, CSc.

Juni, 2004

KATEDRA ŘÍDICÍ TECHNIKY

Katedra řídicí techniky

Inhalt

Dieser Studientext unterstützt die Vorlesungsreihe „Einführung in MATLAB und eine Anwendung in der Regelungstechnik“ im Rahmen der bilateralen Zusammenarbeit der TU Liberec und FH Zittau/Görlitz im Programm Erasmus.

1 MATHEMATISCHE MODELLE UND MODELLTRANSFORMATION

Ziele der ersten Kapitel:

1) Für lineare zeitinvariante Modelle, die als LTI Objekte bezeichnet sind, möchten wir den Leser einige ausgewählten Grundfunktionen für Bildung der ***mathematischen Modelle*** vorstellen.
2) ***Modelltransformationen (Model Conversion)***.
3) Bekanntmachung mit Werkzeugen für ***Zeitverlauf der Ausgänge*** von LTI-Objekte für definierte Eingänge.

Eine genaue Beschreibung aller Funktionen finden Sie in „help“ oder in den umfangreichen Handbüchern [1,2,3,4].

1.1 ANWEISUNGEN FÜR BILDUNG VON LTI MODELLEN

Das dynamische Verhalten von einem System kann allgemein durch differential Gleichungen beschrieben werden. In der Regelungstechnik werden lineare zeitinvariante Systeme im Bildbereich durch Übertragungsfunktion beschrieben.

1.1.1 Eingrößensysteme

Das Laplace- oder Z- Bild des Ausgangs aus einem SISO System ist gleich

$$Y(s) = F(s)U(s), \quad Y(z) = F(z)U(z),$$

wo ist, $U(s), U(z)$ die Laplace-, Z- Transformation von $u(t)$,
s, z komplexe Variable,
$z = \exp(sT_S)$ komplexe Variablen,
T_S die Abtastperiode,
$F(s)$ die Übertragungsfunktion, $F(z)$ die diskrete Übertragungsfunktion mit der Abtastperiode T_S,

$$F(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} = \frac{Y(s)}{U(s)}$$

$$F(z) = \frac{B(z)}{A(z)} = \frac{b_m z^m + b_{m-1} z^{m-1} + \cdots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0} = \frac{Y(z)}{U(z)}.$$

Die Übertragungsfunktion kann nach dem Fundamentalsatz der Algebra in folgender Form geschrieben werden:

$$F(s) = \frac{B(s)}{A(s)} = \frac{b_m (s - s_{1B})(s - s_{2B}) \cdots (s - s_{mB})}{a_n (s - s_1)(s - s_2) \cdots (s - s_n)} = k_0 \frac{\prod_{j=1}^{m} (s - s_{jB})}{\prod_{k=1}^{n} (s - s_k)} = \frac{Y(s)}{U(s)},$$

$$F(z) = \frac{B(z)}{A(z)} = \frac{b_m (z - z_{1B})(z - z_{2B}) \cdots (z - z_{mB})}{a_n (z - z_1)(z - z_2) \cdots (z - z_n)} = k_0 \frac{\prod_{j=1}^{m} (z - z_{jB})}{\prod_{k=1}^{n} (z - z_k)} = \frac{Y(z)}{U(s)},$$

wobei ist, k_0 ... ein Faktor,
s_{jB} , z_{jB} ...j-te Nullstelle,

$s_{,k}$, z_k ...j-te Polstelle.

Diese Form wird als ***Pol-Nullstelle-Form*** bezeichnet.

1.1.2 Mehrgrößensysteme.

Der Ausgangsvektor eines MIMO- System gleicht

$$\boxed{\boldsymbol{y}(s) = \boldsymbol{F}(s)\boldsymbol{u}(s),}$$

wobei ist, $\mathbf{u}(s) = L\{\mathbf{u}(t)\}$,

$\mathbf{F}(s)$ die Übertragungsmatrix vom Typ (p, q) in der Form

$$\boldsymbol{F}(s) = \begin{bmatrix} F_{11}(s) & F_{12}(s) & \cdots & F_{1q}(s) \\ F_{21}(s) & F_{22}(s) & \cdots & F_{2q}(s) \\ & \cdots & & \\ F_{p1}(s) & F_{p2}(s) & \cdots & F_{pq}(s) \end{bmatrix} \boldsymbol{y}(s) = \begin{bmatrix} Y_1(s) \\ Y_2(s) \\ \vdots \\ Y_q(s) \end{bmatrix}, \boldsymbol{u}(s) = \begin{bmatrix} U_1(s) \\ U_2(s) \\ \vdots \\ U_p(s) \end{bmatrix}.$$

Das Element $F_{ij}(s)$ der Übertragungsmatrix ist diejenige Übertragungsfunktion, welche die i- te Eingangsgröße mit der j- ten Ausgangsgrößen verknüpft. Daraus folgt

$$F_{ij}(s) = \frac{Y_i(s)}{U_j(s)} \rightarrow Y_i(s) = F_{ij}(s)U_j(s).$$

Für dir LTI- Modellbildung wurden für diesen Text nur die wichtigsten Funktionen erklärt.

1.1.3 Bildung einer Übertragungsfunktion durch die Funktion tf

Funktion tf **Bildet aus dem Zähler und Nenner die Übertragungsfunktion (tf-transfer function)**

a) Eingrößensysteme (SISO Systeme)

Syntax der Funktion

```
sys = tf (B, A)
sys = tf (B, A, 'variable','p')
sys = tf (B,A,Ts)
```

(1.1-1)

wo ist **sys** ein Lineares Time Invariant (LTI) Objekt

B ein Polynom $B(s) = b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0$ im Zähler , der die Form $B = [b_m, b_{m-1}, \cdots, b_1, b_0]$ hat, wo B ein Vektor ist.

A ein Polynom im Zähler in der Form $A = [a_n, a_{n-1}, \cdots, a_1, a_0]$,
($A(s) = a_n s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0$),

Ts die Abtastperiode der diskreten Übertragungsfunktion,

'variable' definiert die Variable, die als String angegeben wird. Der String für Übertragungsfunktion ist *s* oder *p*, für diskrete Übertragungsfunktion *z, z^-1* oder *q*.

Anwendung der Funktion **tf** ist im Beispiel 1 dargestellt.

Beispiel 1.1-1a

```
» B=[1 2 2];
» A=[1 2 2 1];
» s=tf(B,A)

Transfer function:
    s^2 + 2 s + 2
---------------------
s^3 + 2 s^2 + 2 s + 1
```

Beispiel 1.1-1b

```
» B1=[1 -1];
» A1=[1 3 2 1];
» s1=tf(B1,A1,'variable','p')

Transfer function:
        p - 1
---------------------
p^3 + 3 p^2 + 2 p + 1
```

Beispiel 1.1-1c

```
» B=[1 0];
» A=[1 -1.5 0.5];
» sd=tf(B,A,1)

Transfer function:
        z
-----------------
z^2 - 1.5 z + 0.5

Sampling time: 1
```

Achten Sie darauf, wenn Sie ein LTI Objekt gebildet haben oder wenn Sie ein LTI Objekt zur Verfügung im Arbeitsspeicher haben, dass die Vektoren und Matrizen, die die Polynome *A, B* (Pol- und Nullstellen, Faktoren, die Zustandsmatrizen) darstellen, allgemein für Sie nicht zugänglich sind. Die Polynome der Übertragungsfunktion eines Modells können Sie mit Hilfe der Funktion **tfdata** zugänglich machen.

Funktion tfdata

Gibt den Zähler und Nenner der Übertragungsfunktion LTI Objekts (TF, ZPK, SS) zurück

Syntax der Funktion **[B,A] = tfdata (sys,'v')** (1.1-2)

Wo ist, **sys** ein LTI Objekt,

B der Nenner ist, der in der Form $B = [b_m, b_{m-1}, \cdots, b_1, b_0]$,

A der Zähler in der Form $A = [a_n, a_{n-1}, \cdots, a_1, a_0]$. (Siehe Beispiel 1d)

Die Anwendung von der Funktion **tfdata** ist sehr einfach und wird am folgendenden Beispiel gezeigt.

Beispiel 1.1-1d

```
» [A,B]=tfdata(s,'v')

A =

     0     1     2     2

B =

     1     2     2     1
```

```
» [A,B]=tfdata(s1,'v')

A =

     0     0     0     1    -1

B =

     1     3     2     1     0
```

b) **Mehrfachsysteme (MIMO Systems)**

```
sys = tf (num, den),
sys = tf (num, den , 'variable','p'),
sys = tf (num, den ,Ts),
```
(1.1-3)

wobei ist,

sys ein Linear Time Invariant (LTI) Mehrfachobjekt,

num, den Feldmatrizen (cell array). Die Feldmatrizen haben soviel Zeilenvektorren, wie viel Ausgänge das Modell hat. Sie haben so viel Spalten, wie viel Eingänge das Modell hat. Struktur der Feldmatrizen:

{[…] […] […] … ; […] […] […] … ; […] […] […] … ; …}

num eine Feldmatrix des Zählers – Zählermatrix (cell array),

den eine Feldmatrix des Nenners – Nennermatrix (cell array).

Anwendung der Funktion **tf** für MIMO Systeme werden wir auf dem folgenden Beispiel demonstrieren. Die Koeffizienten der Übertragungsfunktionen sind direkt in dem Beispiel eingetragen.

Beispiel 1.1-2 Auf dem Bild 2a,b,c sind Strukturen der drei Mehrfachsysteme dargestellt.

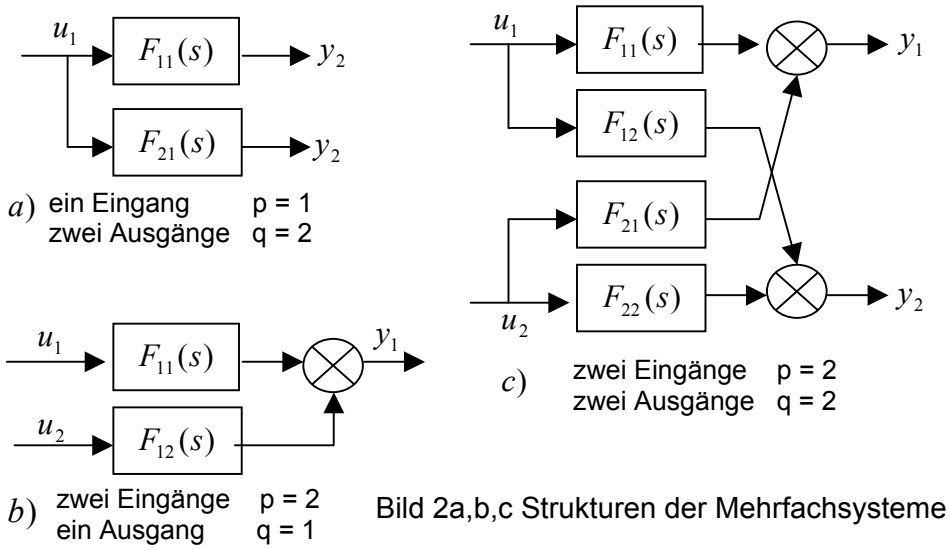


Bild 2a,b,c Strukturen der Mehrfachsysteme

Aufgabenstellung: Bilden Sie für die Strukturen im Bild 2 die entsprechenden Übertragungsfunktionen der LTI Modelle.

```
2a)
nums={2 ;[-1 2] };
denums={[4 4 1];[1 2 1]};
G21=tf(nums, denums)
```

```
2b)
nums1={2 [-1 2] };
denums1={[2 2 1] [1 2 1]};
G12=tf(nums1, denums1)
```

```
Transfer function from input to output...
                2
F11 = #1:  ---------------
           4 s^2 + 4 s + 1

2c)           -s + 2
F21 = #2:  -------------
           s^2 + 2 s + 1
```

```
Transfer function from input 1 to output:
          2
F11 = ---------------
      2 s^2 + 2 s + 1

Transfer function from input 2 to output:
        -s + 2
F12 = -------------
      s^2 + 2 s + 1
```

```
c)
nums2={2 [1 0.5];[1 0] 2 };
denum2={[4 4 1] [2 1 2];[1 1] [1 2 1]};
Gp22=tf(nums2, denum2,'variable','p')
```

```
Transfer function from input 1 to output...
                2
F12 = #1:  ---------------
           4 p^2 + 4 p + 1

                p
F12 = #2:  -------
           p + 1

Transfer function from input 2 to output...
             p + 0.5
F21 = #1:  -------------
           2 p^2 + p + 2

                2
F22 = #2:  -------------
           p^2 + 2 p + 1
```

1.1.4 Bildung einer Übertragungsfunktion durch die Funktion zpk

Funktion zpk **Bildet die Übertragungsfunktion in der Null-Pol-Form**

Syntax der Funktion

```
sys = zpk(Z,P,K)
sys = zpk(Z,P,K,Ts)
```

(1.1-4)

wobei ist, **sys** ein LTI Objekt ist,

Z ein Zeile-Vektor **Z** ist, der die vorgeschriebenen Nullstellen in der Form $Z = [s_{Bm}, s_{Bm-1}, \cdots, s_{B1}]$ enthält,

P ein Zeile-Vektor **P** ist, der die vorgeschriebenen Polstellen in der Form $P = [s_n, s_{n-1}, \cdots, s_1]$ enthält,

K ein Zeile-Vektor ist, der die vorgeschriebene Verstärkung enthält.

Anwendung der Funktion **zpk** für realen und komplexe Polstellen und Nullstellen ist im Beispiel 2a,b,c demonstriert. Die ergebnise sind im Bild 2 dargestellt.

Beispiel 1.1-3a

```
» Z=[];
» P=[-1 -0.5 -2];
» K=[2];
» sz=zpk(Z,P,K)

Zero/pole/gain:
          2
-------------------
(s+1) (s+0.5) (s+2)
```

Beispiel 1.1-3c

```
» Z2=[0];
» P2=[0.5 0.5];
» K2=[2.5];
» zs3=zpk(Z2,P2,K2,1,'variable','z^-1')

Zero/pole/gain:
   2.5 z^-1
---------------
(1 - 0.5z^-1)^2

Sampling time: 1
»
```

Beispiel 1.1-3b

```
» Z1=[-0.5];
» P1=[-1+i -1-i -0.5];
» K1=[10];
» sz1=zpk(Z1,P1,K1)

Zero/pole/gain:
      10 (s+0.5)
----------------------
(s+0.5) (s^2 + 2s + 2)
```

Bild 2 Anwendungen der Funktion **zpk**

Funktion zpkdata **Gibt die Null- und Polstellen und den Faktor einer Übertragungsfunktion des LTI Objekts in der Pol-und Null-Form zurück**

Syntax der Funktion **[B,A] = zpkdata (sys,'v')** (1.1-5)

Wo ist, **sys** ein LTI Objekt,
B der Nenner ist, der in der Form $B = [b_m, b_{m-1}, \cdots, b_1, b_0]$,
A der Zähler in der Form $A = [a_n, a_{n-1}, \cdots, a_1, a_0]$. (Siehe Beispiel 1d)

Die Anwendung von der Funktion **tfdata** folgt..

Beispiel 1.1-3d

```
» [Z1,P1,K1]=zpkdata(sz1,'v')

Z1 =

   -0.5000


P1 =

  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
  -0.5000


K1 =

    10
```

```
» [ZT,PT,KT]=zpkdata(s,'v')

ZT =

  -1.0000 + 1.0000i
  -1.0000 - 1.0000i


PT =

  -1.0000
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i


KT =

     1
```

1.1.5 Bildung einer Übertragungsfunktion durch die Funktion ss

Allgemeine Form der Zustandsgleichung eines LTI kontinuierlichen and diskreten Systems hat die Form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(k+1) = \mathbf{M}\mathbf{x}(k) + \mathbf{N}\mathbf{u}(k),$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \qquad \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k),$$

dabei ist ***x*** der *n*- gliedrige Zustandsvektor, ***u*** der *q*- gliedrige Eingangsvektor und ***y*** der *p*-gliedrige Ausgangsvektor. Die Matrizen

(n, n)- **Systemmatrix**	***A***,	*(n, n)*- **diskrete Systemmatrix**	***M***,
(n, q)- **Eingangsmatrix**	***B,***	*(n, q)*- **diskrete Eingangsmatrix**	***N,***
(p, n)- **Ausgangsmatrix**	***C,***	*(p, n)*- **diskrete Ausgangsmatrix**	***C,***
(p, q)- **Durchgangsmatrix**	***D,***	*(p, q)*- **diskrete Durchgangsmatrix**	***D,***

bestehen aus konstanten Elementen.

Funktion SS **Bildet ein LTI Modell in der Zustandsform, kontinuierlich oder diskret**

Syntax der Funktion

```
sys = ss(A,B,C,D)

sys = ss(M,N,C,D,Ts)
```

(1.1-6)

Wo **A** - Systemmatrix
B - Eingangsmatrix
C - Ausgangsmatrix
D - Durchgangsmatrix
M - diskrete Systemmatrix
N - diskrete Eingangsmatrix
Ts - Abtastperiode

Eine Anwendung der Funktion **ss** werden wir auf dem Beispiel 3 demonstrieren.

Beispiel 1.1-4 Dynamische Eingeschafften einer Regelstrecke sind durch folgende Übertragungsfunktion approximiert $F(s)=\dfrac{2}{s^3+3s^2+s+1} \rightarrow y'''=-3y''-y'-y+2u.$

Aufgabe: Bilden Sie ein LTI Objekt in der Form einer Zustandsgleichung.
Zuerst ist es nötig, die Matrizen der Zustandsgleichung zu finden.

Wenn wir wählen wir die Zustandgröße:

$$\begin{aligned} x_1 &= y & \dot{x}_1 &= y' = x_2, \\ x_2 &= y' \rightarrow & \dot{x}_2 &= y'' = x_3, \\ x_3 &= y'' & \dot{x}_3 &= y''' = -3x_3 - x_2 - x_1 + 2u, \end{aligned}$$

Dann die Zustandsgleichung hat folgende Form

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \cdot u = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot u,$$

und der Ausgang ist gleich:

$$y = [1\ 0\ 0] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + 0 \rightarrow y = \mathbf{C}\mathbf{x}.$$

wobei ist $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -3 \end{bmatrix}$; $\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$; $\mathbf{C} = [1\ 0\ 0]$; $D = 0.$

```
» A=[0 1 0;0 0 1;-1 -1 -3];B=[0;0;2];
» C=[1 0 0];D=0;
» ss1=ss(A,B,C,D)

a =
                    x1          x2          x3
           x1        0           1           0
           x2        0           0           1
           x3       -1          -1          -3

b =
                    u1
           x1        0
           x2        0
           x3        2

c =
                    x1          x2          x3
           y1        1           0           0

d =
                    u1
           y1        0

Continuous-time model.
» |
```

Funktion ssdata **Gibt die Zustandsmatrizen eines LTI Objekts zurück**

Syntax der Funktion

```
[A,B,C,D] = ssdata (sys,'v')
[M,N,C,D,Ts] = ssdata (sys,'v')
```

(1.1-7)

Wo ist, **sys** ein LTI Object

A - Systemmatrix **M** - diskrete Systemmatrix
B - Eingangsmatrix **N** - diskrete Eingangsmatrix
C - Ausgangsmatrix **Ts** - Abtastperiode
D - DurchgangsmatrixDie Anwendung von der Funktion **tfdata** folgt

1.1.6 Bildung eines LTI Objekts mit der Totzeit

Im MATLAB kann die Totzeit am Eingang oder Ausgang des Modells stehen (siehe Bild x1).

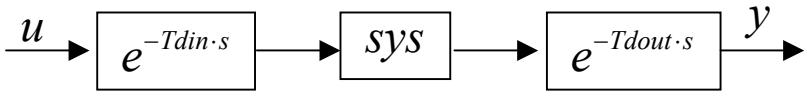


Bild x1 Struktur der Totzeit in MATLAB

Die gesamte Totzeit ist:

$$F(s) = \exp(-T_{din}.s)F_0(s)\exp(-T_{dout}.s) = F_0(s)\exp(-T_d.s),$$

wo ist, $T_d = T_{din} + T_{dout}$.

Die Totzeitglieder werden in serieller Verbindung am Eingang und Ausgang des Modells des LTI Objekts mit Hilfe der Funktion **sys.input** und **sys.output** angeschlossen.

Syntax der Funktion

```
sys .input = Tdin
sys .output = Tdout
td = totaldelay( sysf )
```

(1.1-8)

wo ist,

sys	ein LTI Objekt,
Tdin	die Totzeit am Eingang,
Tdout	die Totzeit am Ausgang,
Td	die gesamte Totzeit des Objekts**.**

Ein Anwendungsbeispiel folgt.

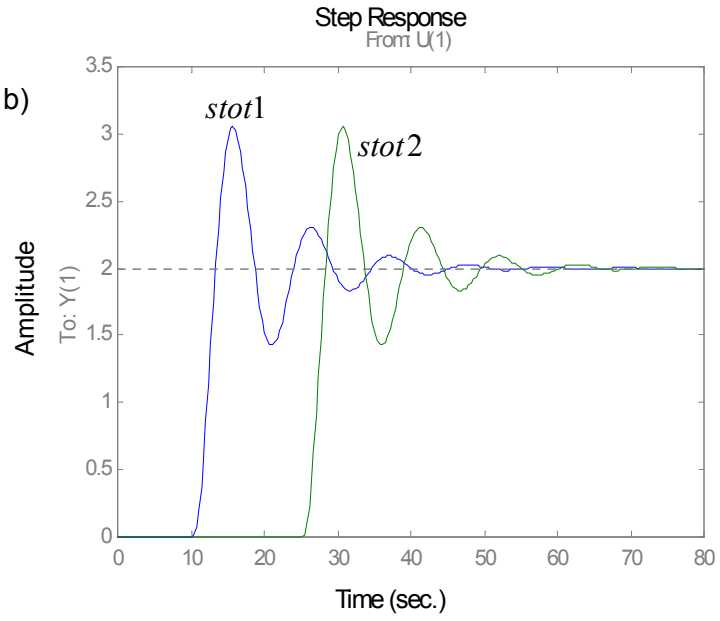
Beispiel 1.1-5 Ein LTI System ist durch Übertragungsfunktion $F = \frac{B}{A}$ bestimmt.

Aufgabestellung: Es sollen zwei LTI Objekte in MATLAB gebildet. Ein hat die Totzeit *Tdin = 10sec* am Eingang, der zweite hat eine Totzeit *Tdin = 10sec* am Eingang, die zweite *Tdout = 15sec* ist am Ausgang. Gesamte Totzeit *td = 25sec*. Das Programm ist im Bild X.Ya, die Einheitssprungantworten von beiden LTI Objekte ist im Bild X.Yb dargestellt.

a)

```
%B_Totzeit1
A=[ 1 3 1 1];
B=[2];
stot1=tf(B,A);
stot1.inputd=10;
td1=totaldelay(stot1)

stot2=tf(B,A);
stot2.inputd=10;
stot2.outputd=15;
td2=totaldelay(stot2)
step(stot1,stot2)
```

b)



1.2 MODELLTRANSFORMATIONEN

Für Bildung der LTI Modelle wurden nur die wichtigsten Funktionen beschrieben. Die Modelle können stetig oder diskret sein. Alle Modellformen können gegenseitig überführt werden. Dazu dienen Funktionen für die Modell- Transformation.

1.2.1 Transformation des LTI- Objekts in das Übertragungsfunktionsmodell

Ein LTI Objekt kann in die Form der Übertragungsfunktion mit Hilfe der Funktion **tf** transformiert werden**.**

Syntax der Funktion **sys = tf (sysf)** (1.2-1)

Wo ist, **sys** ein LTI Objekt in der Form einer **Übertragungsfunktion**
sysf ein LTI Objekt in der Form eines **Zustandsmodells oder Pol- Nullstellen-Modells**

Beispiel 1.2-1

In dem Arbeitspeicher ist ein LTI Modell **sz** in der Pol- Nullstellen Form und LTI Objekt **ss1** als ein Zustandsmodell. Durch einfache Verwendung von der Funktion **tf** werden diese Modelle in die Übertragungsform überführt.

```
» szpk=tf(sz)

Transfer function:
              2
-------------------------
s^3 + 3.5 s^2 + 3.5 s + 1
```

```
» sss1=tf(ss1)

Transfer function:
              1
-------------------
s^3 + 2 s^2 + s + 1
```

1.2.2 Transformation des LTI- Objekts in die Pol- Nullstelle- Form

Ein LTI Objekt kann man in die Pol- Nullstelle -Form direkt mit der Funktion **zpk** überführen.

Syntax der Funktion **sys = zpk (sysf)** (1.2-2)

Wo ist, **sys** ein LTI Objekt in der **Pol- Nullstelle- Form**
sysf ein LTI Objekt in der Form eines **Zustandsmodells oder eine Übertragungsfunktion**

Beispiel 1.2-2

In dem Arbeitspeicher ist ein LTI Modell **s** in der Übertragungsfunktion Form und LTI Objekt **ss1** als ein Zustandsmodell. Durch einfache Verwendung von der Funktion **zpk** werden diese Modelle in die **Pol- Nullstelle- Form** überführt.

```
» szpk1=zpk(s)

Zero/pole/gain:
  (s^2 + 2s + 2)
------------------
(s+1) (s^2 + s + 1)
```

```
» szpk2=zpk(ss1)

Zero/pole/gain:
                1
---------------------------------
(s+1.755) (s^2 + 0.2451s + 0.5698)
```

1.2.3 Transformation des LTI- Objekts in das Zustandsmodell

Ein LTI Objekt kann man in das Zustandsmodell direkt mit der Funktion **ss** überführen.

Syntax der Funktion

sys = ss (sysf) (1.2-3)

Wo ist, **sys** ein LTI Objekt als **Zustandsmodell**
sysf ein LTI Objekt in der **Pol- Nullstelle- Form oder in der Übertragungsfunktionsform**

Beispiel 1.2-3

In dem Arbeitspeicher ist ein LTI Modell **s** in der Übertragungsfunktion Form. Durch einfache Verwendung von der Funktion **ss** wird dieses Modelle in das Zustandsmodell überführt.

```
» ss1s=ss(s)

a =
                x1          x2          x3
    x1          -2          -1        -0.5
    x2           2           0           0
    x3           0           1           0

b =
                u1
    x1           2
    x2           0
    x3           0

c =
                x1          x2          x3
    y1         0.5         0.5         0.5

d =
                u1
    y1           0

Continuous-time model.
»
```

1.2.4 Transformation eines kontinuierlichen LTI Objekts in das diskrete

Bei der Diskretirrung werden verschiedene Methoden benutzt. Allgemeine Struktur der Diskretirrung eines LTI Objekts ist im Bild 1.2.1 dargestellt. Wichtigen Einfluss auf die diskrete Übertragungsfunktion hat das Halteglied oder die Approximation von der komplexen Variable „s“ (bilineare- Tustin Approximation usw.). Die in MATLAB benutzten Diskretirrungsmethoden sind ausführlich in [6] beschrieben.

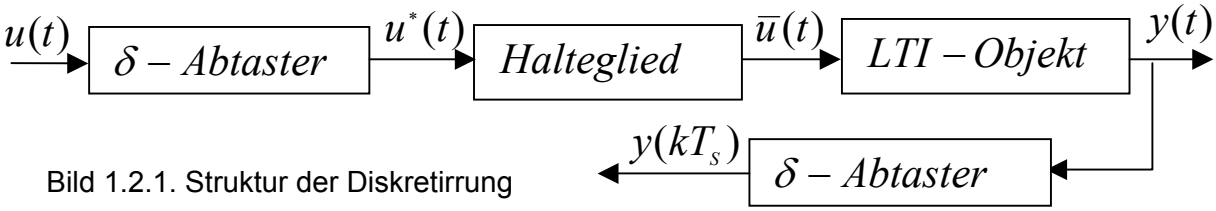


Bild 1.2.1. Struktur der Diskretirrung

In MATLAB wird dazu die Funktion **c2d** benutzt.

Funktion c2d **Transformiert ein kontinuierliches LTI Objekt in das diskrete Objekt**

Syntax der Funktion

```
sysd = c2d(sys,Ts)
sysd = c2d(sys,Ts,'method')
```

(1.2-4)

Wo ist, **sys** ein **kontinuierliches** LTI Objekt,
sd ein diskretes LTI Objekt,
Ts gewählte Abtastperiode,
'method' definiert, welche Diskretisierung- Methode benutzt wird**.** Wenn keine **'method'** angegeben wird, wird automatisch **'zoh'** eingesetzt.

Die String '**method**' ist in der Tab.2 definiert.

'method'	**Bedeutung**
'zoh'	Das Abtast-Halteglied von Null- Ordnung (Zero-Order Hold)
'foh'	Das modifizierte Dreieck Approximation von dem Abtast-Halteglied erster Ordnung (Triangle Approximation modified first-order hold)
'tustin'	Bilineare Tustin- Approximation
'matched'	Modifizierte Pol-Nullstellen- Methode (nur für SISO- Systeme [2])

Tab.2

Eine Anwendung der Transformationsfunktion **c2d** ist im Beispiel 1.2-4 gezeigt.

Beispiel 1.2-4

Diskkretisieren Sie das LTI Modell „s“ mit der Abtastperiode 1 sec und mit dem Halteglied erster Ordnung. Anweisungen folgen, der Einheitssprung ist im Bild 1.2.2 dargestellt.

```
Transfer function:
    s^2 + 2 s + 2
---------------------
s^3 + 2 s^2 + 2 s + 1

zoh Transfer function:
      0.9724 z^2 - 0.3803 z + 0.1436
---------------------------------
z^3 - 1.154 z^2 + 0.657 z - 0.1353

Sampling time: 1

Foh     Transfer function:
        0.4941 z^3 + 0.3084 z^2 - 0.136 z + 0.06923
        ------------------------------------------
            z^3 - 1.154 z^2 + 0.657 z - 0.1353

        Sampling time: 1

tustin  Transfer function:
        0.4762 z^3 + 0.2857 z^2 - 0.09524 z + 0.09524
        ---------------------------------------------
             z^3 - 1.19 z^2 + 0.7143 z - 0.1429

        Sampling time: 1

matched Transfer function:
          0.9972 z^2 - 0.3964 z + 0.135
        ---------------------------------
        z^3 - 1.154 z^2 + 0.657 z - 0.1353

        Sampling time: 1
```

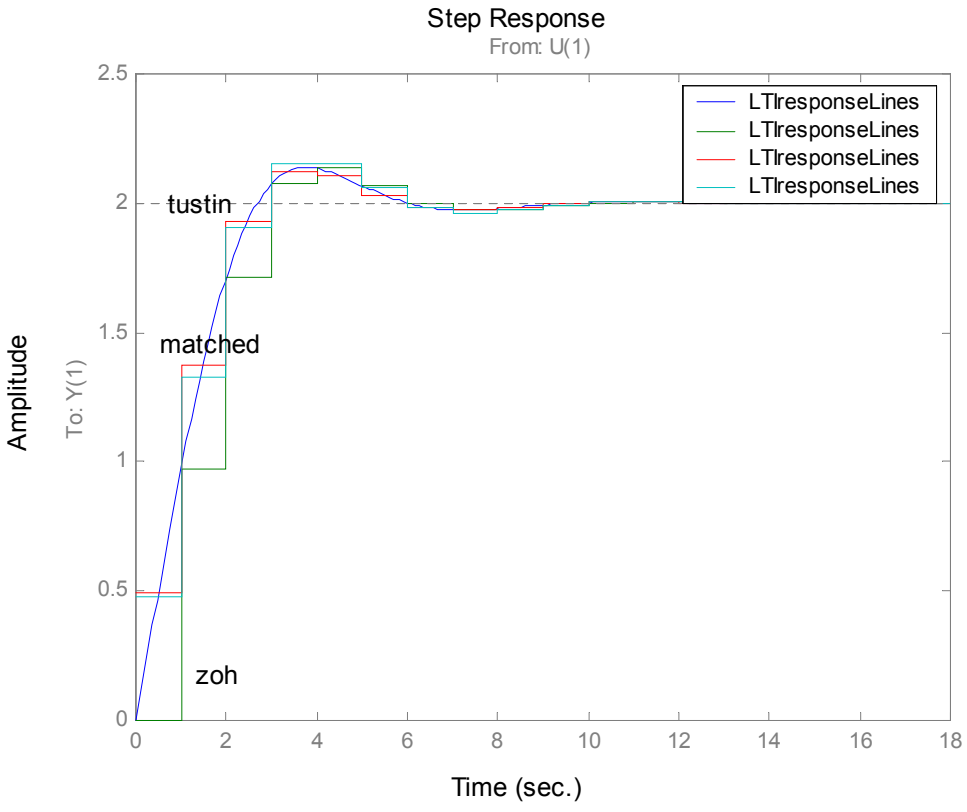


Bild 1.2.1 Diskrete Einheitssprünge der diskreten LTI Systeme

1.2.5 Transformation eines diskreten LTI- Modells in das kontinuierliche Modell.

In MATLAB kann ein diskretes LTI- Modell in das kontinuierliche Modell einfach transformiert werden. Dazu dient die Funktion **d2c**. Die Transformationsmethoden sind in [2] beschrieben.

Funktion d2c **Transformiert ein kontinuierliches LTI Objekt in das diskrete Objekt**

Syntax der Funktion

```
sysc = d2c(sysd)
sysc = d2c(sysd,'method')
```

(1.2-5)

Wo ist, **sysc** ein **kontinuierliches** LTI Objekt,
sysd ein diskretes LTI Objekt,
'method' definiert, welche Methode zur Transformation benutzt wird**.** Wenn keine **'method'** angegeben wird, wird automatisch **'zoh'** eingesetzt. Die String '**method**' ist in der Tab.3 definiert.

'method'	**Bedeutung**
'zoh'	Das Abtast-Halteglied von nullter Ordnung
'tustin'	Bilineare Tustin- Approximation mit Rücksicht zu der Ableitung [2]
'matched'	Modifizierte Pol-Nullstellen- Methode (nur für SISO- Systeme [2])

Tab. 3

Eine Anwendung der Transformationsfunktion **d2c** ist im Beispie l.2-5 demonstriert.

Beispiel 1.2-5

Die diskrete Übertragungsfunktion sd wurde im Beispiel 1.2-4 gefunden. Die Aufgabe ist es, mit Hilfe der Funktion **d2c** ein kontinuierliches Modell zurückzubekommen. Alle drei Transformationsmethoden sollen geprüft werden. Die Wirkung der einzelnen Methoden an den Zeitverlauf des Einheitssprungs sind in dem Bild 1.2.3 sichtbar.

```
%B_d2c1
B=[1 2 2];
A=[1 2 2 1];
s=tf(B,A)    %kont. System
sd=c2d(s,1) %dikretr System
sds=d2c(sd) %d2c transformationen
sds2=d2c(sd,'matched')
sds1=d2c(sd,'tustin')
step(s,sd,sds1,sds2)
```

```
Transfer function:
    s^2 + 2 s + 2
---------------------
s^3 + 2 s^2 + 2 s + 1

Transfer function:
1.005 s^2 + 1.922 s + 2
-----------------------
 s^3 + 2 s^2 + 2 s + 1

Transfer function:
-0.5079 s^3 - 0.1095 s^2 + 1.252 s + 1.998
------------------------------------------
    s^3 + 2.098 s^2 + 2.166 s + 0.999
```

```
Transfer function:
    s^2 + 2 s + 2
--------------------
s^3 + 2 s^2 + 2 s + 1

Transfer function:
  0.9724 z^2 - 0.3803 z + 0.1436
---------------------------------
z^3 - 1.154 z^2 + 0.657 z - 0.1353
```

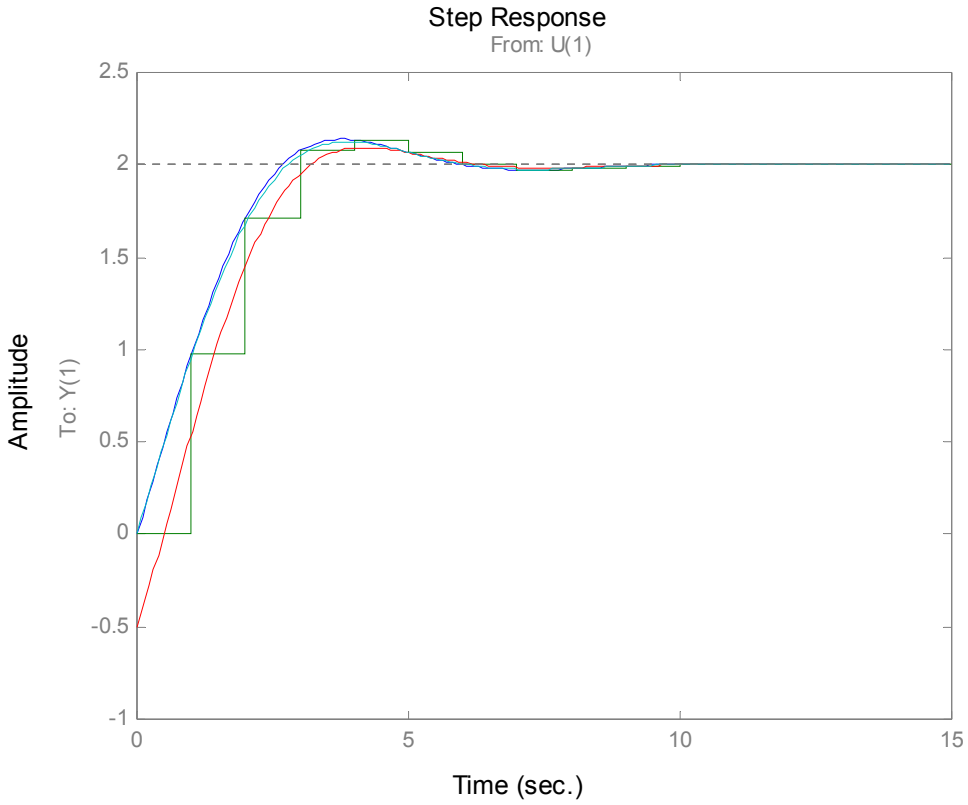


Bild 1.2.3 Zeitverlauf der Einheitssprünge

2 ZEIT-UND FREQUENZVERHALTEN DER LTI OBJEKTE

Als Testsignale wird in der Regelungstechnik der Einheitssprung, Dirac-Impuls und das harmonisches Signal (z.B. Sinussignal) benutzt. Diesen entsprechen die Sprungantwort, Impulsantwort und die Sinusantwort. In diesem Kapitel werden zuerst die zeitlichen Verläufe der Ausgangsgröße eines LTI Objekts, anschließend dann die Antworten auf ein komplexes harmonisches Eingangsignal behandelt. Es werden die wichtigste MATALB Funktionen vorgestellt und ihre Anwendung auf Beispielen erklärt.

2.1 ZEITVERHALTEN DER LTI – OBJEKTE

Die Sprungantwort ist der zeitliche Verlauf des Ausgangssignals auf einer sprungförmige Änderung des Eingangssignals. Wenn das Eingangssignal ein Einheitssprung ist, dann bildet der Ausgang eines LTI- Objekts eine Übergangsfunktion.

> Die Übergangsfunktion $h(t)$ eines LTI- Objekts ist seine Antwort auf den Einheitssprung. Graphische Darstellung kann als Übergangscharakteristik bezeichnet werden.

Der zeitliche Verlauf des Ausgangs eines LTI –Objektes auf Dirac- Impuls wird mit einer Gewichtsfunktion (Impulsfunktion) beschrieben.

> Die Gewichtsfunktion *g(t)* eines LTI- Objekts ist seine Antwort auf den Dirac Impuls δ(t). Die Gewichtsfunktion ist die zeitliche Ableitung der Übergangsfunktion.

2.1.1 Darstellung der Übergangsfunktion

Der Zeitverlauf der Übergangsfunktion wird im MATLAB mit Hilfe der Funktion **step** berechnet und graphisch dargestellt.

Funktion step — **Berechnet und stellt den Zeitverlauf der Übergangsfunktion dar**

Syntax der Funktion

step(sys),	-Berechnung und Zeichnung der Übergangsfunktion des LTI- Objektes **sys.** Simulationszeit wird automatisch bestimmt.
step(sys,t),	- Berechnung und Zeichnung der Übergangsfunktion des LTI- Objektes **sys,** Vektor **t** bestimmt die Zeitlänge der Simulation
step(sys1,sys2,…,sysN)	- Berechnung und Zeichnung der Übergangsfunktionen der LTI- Objekte **sys1,sys2,…,sysN**
[y,t]=step(sys)	-Berechnung und Zeichnung der Übergangsfunktion des LTI- Objektes **sys.** Vektor **y** enthält die Ordinatenwerte, der Vektor **t** enthält die Zeit.

Anwendung der Funktion **step** ist im Beispiel 2.1-1 demonstriert.

Beispiel 2.1-1

In dem Arbeitsspeicher sind zwei LTI- Objekte s, s1 gespeichert. Den Zeitverlauf der Übergangsfunktionen wird durch **step** Funktion dargestellt (siehe Bild 2.1.1).

```
» s

Zero/pole/gain:
2 (s+0.5) (s-0.75)
------------------
  (s+1) (s+2)^2

 » step(s,s1)
```

```
» B=3;
» A=[1 1 3];
» s1=tf(B,A)

Transfer function:
     3
-----------
s^2 + s + 3
```

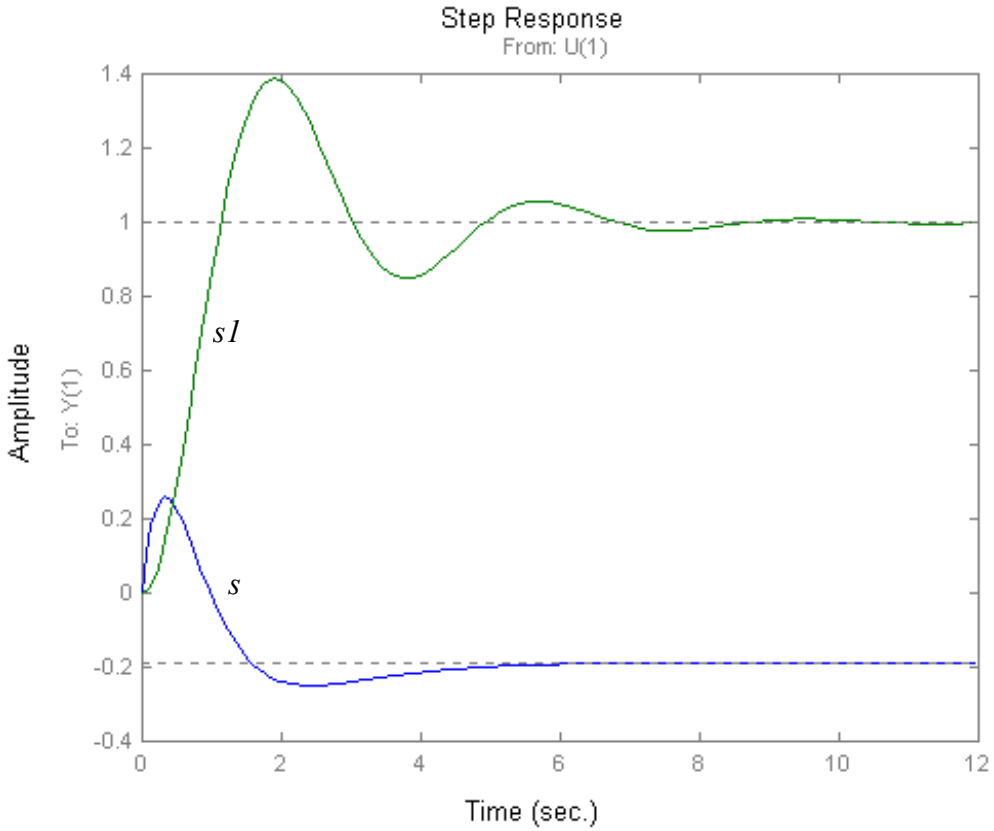


Bild 2.1.1 Darstellung der Übergansfunktionen s,s1.

2.1.2 Darstellung der Impulsfunktion

Berechnung und Darstellung der Impulsfunktion is im MATLAB durch die Funktion **impuls** gesichert.

Funktion impulse — **Berechnet und stellt den Zeitverlauf der Impulsfunktion dar.**

Syntax der Funktion

impulse (sys)	-Berechnung und Zeichnung der Impulsfunktion des LTI Objektes **sys.** Simulationszeit wird automatisch bestimmt.
impulse (sys,t)	-Berechnung und Zeichnung der Impulsfunktion des LTI Objektes **sys.** Simulationszeit wird durch den Vektor **t** bestimmt.
impulse (sys1,…,sysN)	-Berechnung und Zeichnung der Impulsfunktionen der LTI Objekte **sys1,…,sysN.**
[y,t] = impulse (sys)	-Berechnung und Zeichnung der Impulsfunktion des LTI Objektes **sys,** der Vektor **y** enthält die Ordinatenwerte, der Vektor **t** die Zeit.

Anwendung der Funktion **step** ist im Beispiel 2.1-2 demonstriert.

Beispiel 2.1.2 In dem Arbeitsspeicher sind drei LTI- Objekte s, stot1 und sz gespeichert. Zeichnen Sie die Impulsfunktionen und sichern Sie, dass die Impulsfunktion von jedem LTI- Objekt sich durch Farbe und Linetyp unterscheiden, siehe das Bild 2.1.2.

```
» impulse(s,'b:',stot1,'r--',sz,'g-')
»
```

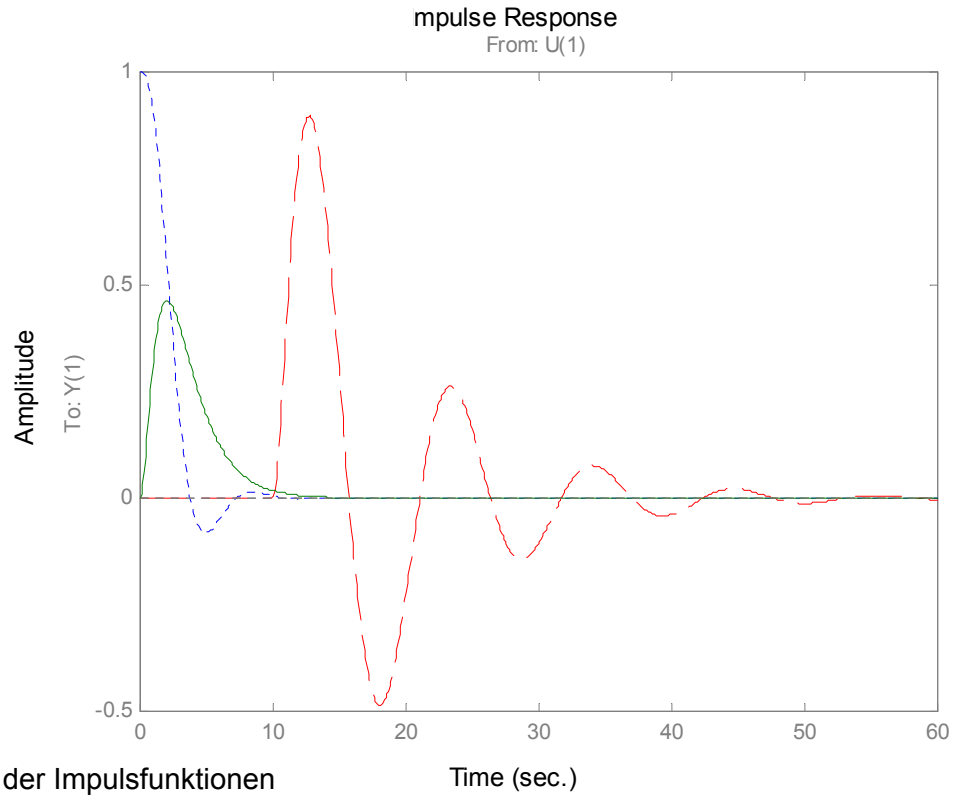


Bild 2.1.2
Darstellung der Impulsfunktionen

2.1.3 Gemeinsame Darstellung der Zeit- und Frequenzantworten

Bei Analyse und Synthese der Regelkreise kann es nützlich sein, wie die Zeitantworten so die Frequenzantworten in einem Bild darzustellen. Dazu dient in MATLAB die Funktion ltiview.

Funktion ltiview — **Berechnet und stellt den Zeit- oder Frequenzverlauf eines LTI Objekts dar**

Syntax der Funktion

```
ltiview
ltiview (plottype,sys)
ltiview (plottype,sys,extras)

ltiview (plottype,sys1,sys2, … ,sysN)
```

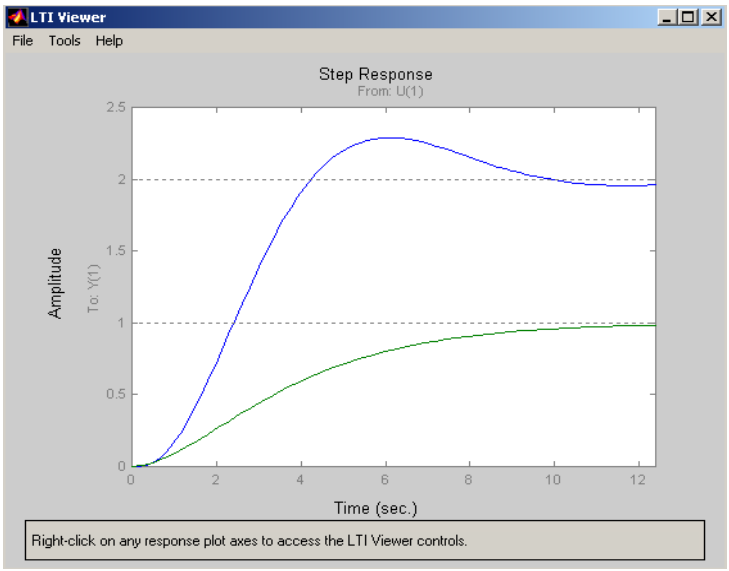
wobei **extras** kann z.B. die Gesamtzeit dr Simulation sein **Tsim**
plottype ein String ist,
'**step**' Zeitverlauf des Einheitssprunges,
'**impulse**' Zeitverlauf des Impulsantwort,
'**bode**' Darstellung der logarithmischen, Amplituden- und Phasengänge,
'**nyquist**' Frequenzgangdarstellung bezeichnen
'**lsim**' Antwort auf beliebiges Signal mit **lsim**
und andere

.

Anwendung der Funktion **ltiview** ist im Beispiel 2.1.3 und 2.1.4 dargestellt

Beispiel 2.1.3

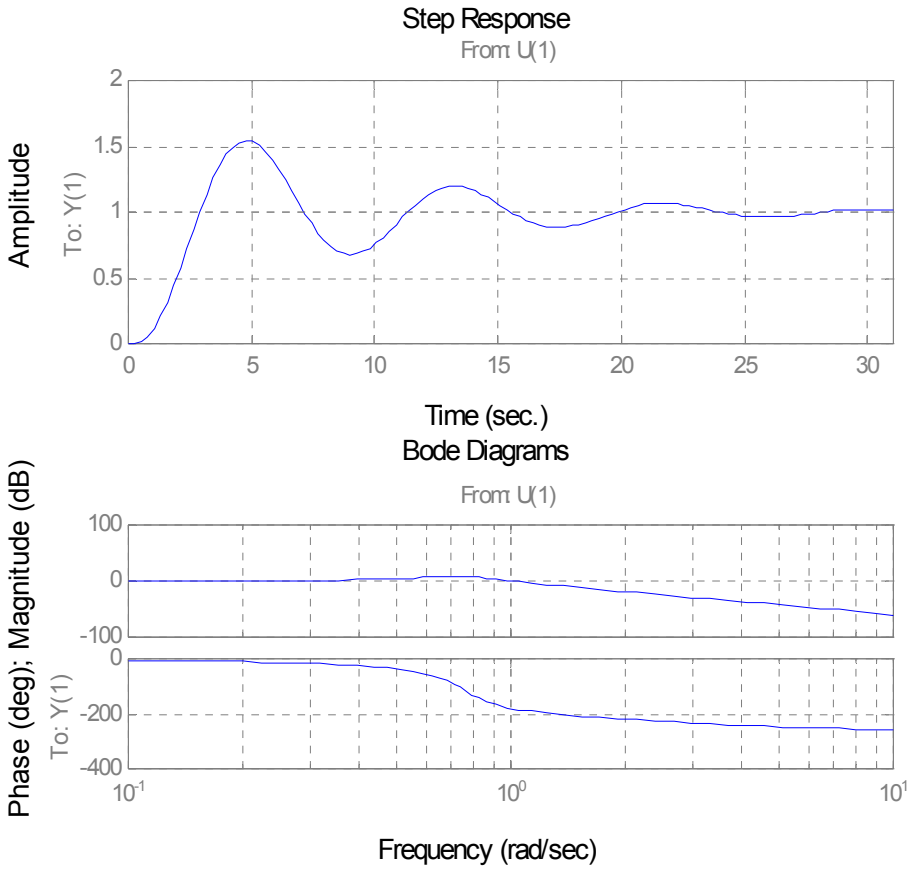
```
» load 23MAI
» s

Transfer function:
          2
----------------------
s^3 + 3 s^2 + 2 s + 1

» s1

Transfer function:
       1
---------------
4 s^2 + 4 s + 1

» ltiview('step',s,s1)
```



Beispiel 2.1.4

Für das LTI- Objekt ss1 soll die Übergangsfunktion und Frequenzgang dargestellt werden. Anwendung der Funktion **ltiview** wird demonstriert.

```
» ltiview({'step';'bode'},ss1)
»
```



2.2 FREQUENZVERHALTEN DER LTI OBJEKTE

Nehmen wir an, dass ein LTI Objekt durch ein harmonisches Signal $u(t) = \sin(\omega \cdot t)$ erregt wird.

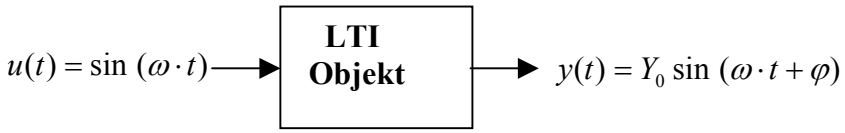


Bild 2.2.1 Erregung eines LTI Objekt durch harmonisches Signal

Der Ausgang ist wieder ein harmonisches Signal

$$y(t) = Y_0 \sin(\omega \cdot t + \varphi),$$

wo ist, $Y_0 = Y_0(\omega)$ die Amplitude ,
$\varphi = \varphi(\omega)$ die Phase.

Das dynamische Verhalten eines LTI- Objekts, das in der Form einer Übertragungsfunktion beschrieb ist, wird allgemein durch die Frequenzganggleichung beschrieben

$$F(i\omega) = \frac{b_{m-1}(i\omega)^{m-1} + \cdots + b_1(i\omega) + b_0}{(i\omega)^n + a_{n-1}(i\omega)^{n-1} + \cdots + a_1(i\omega) + a_0} = \left|F(i\omega)\right| \cdot \exp i\varphi(\omega),$$

wo ist, $\left|F(i\omega)\right| = \sqrt{\Re e\{F(i\omega)\}^2 + \Im m\{F(i\omega)\}^2}$ die Amplitude,

$$\varphi(\omega) = arctg\frac{\Im m\{F(i\omega)\}}{\Re e\{F(i\omega)\}}$$ die Phase

Die Frequenzganggleichung stellt eigentlich eine komplexe Zahl, die von dem Parameter ω abhängig ist. Eine komplexe Zahl in der s- Ebene kann durch den Real- und Komplexenteil oder durch den absoluten Wert $\left|F(i\omega)\right|$ und die Phase $\varphi(\omega)$ dargestellt werden.

Wenn der Parameter ω den Bereich von $-\infty$ bis $+\infty$ durchläuft, dann bekommen wir in der s-Ebene eine Ortskurve d.h. eine graphische Darstellung des Frequenzganges. Graphische Darstellung des Frequenzganges wird als Frequenz- Kennlinien genannt. Frequenz-Kennlinien werden entweder in der komplex Ebene oder als logarithmischer Amplituden- und Phasengang dargestellt.

Für Analyse und Synthese in den Frequenzbereich kann man in MATLAB folgende Anweisungen benutzen: **nyquist, bode, evalfr, freqresp, margin.**

2.2.1 Anweisungen für Frequenzgangdarstellung und -berechnung

Funktion nyquist — **Frequenzgangdarstellung und -berechnung**

Syntax der Funktion

nyquist (sys)	- Frequenzgangberechnung- und Darstellung des LTI Objektes **sys**
nyquist (sys1,…,sysN)	- Frequenzgangberechnung und Darstellung der LTI Objekte **sys1,…,sysN**
nyquist (sys,w)	- Frequenzgangberechnung und Darstellung des LTI Objektes **sys** für Frequenzbereich **w.**
[re,im,w] = nyquist (sys)	- Berechnung des Real- und Imaginärteiles des LTI – Objekts und Speicherung in die Vektore **re,im,w**

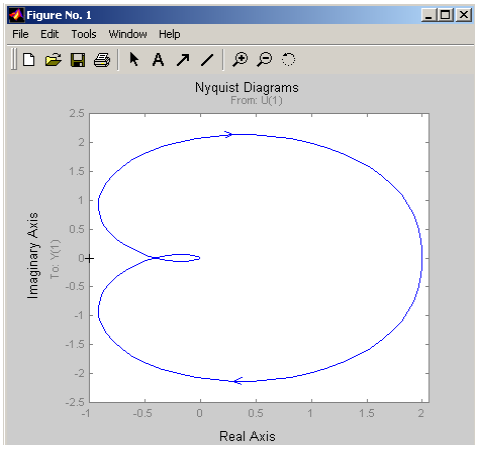
wo **sys** ein LTI Objekt ist,
sys1,sys2,…,sysN LTI Objekte sind,
w Parameter **w** einzugeben ist: **w = {w_{min} w_{max}},**
re,im Vektoren sind, wo Real- und Imaginärteile gespeichert sind.

Anwendung der Funktion **nyquist** ist im Beispiel 2.2.1 demonstriert.

Beispiel 2.2.1

```
» load 23MAI
» s

Transfer function:
          2
---------------------
s^3 + 3 s^2 + 2 s + 1

» w

w =

    [0.1000]    [100]

» nyquist(s,s1)
» nyquist(s)
»
```

```
» s1

Transfer function:
       1
---------------
4 s^2 + 4 s + 1
```



2.2.2 Anweisung zur Berechnung des Frequenzganges

Funktion freqresp **Berechnung des Frequenzganges eines LTI Systems**

Beispiel 2.2.2

```
» w=[0.1 0.5 1];
» H=freqresp(s,w)
H(:,:,1) =
   1.9786 - 0.4059i
H(:,:,2) =
   0.6038 - 2.1132i
H(:,:,3) =
  -0.8000 - 0.4000i
```

2.2.3 Anweisung zur Berechnung des Frequenzganges von komplexe Frequenz

Funktion evalfr **Berechnung des Frequenzganges für a komplex Zahl**

Syntax der Funktion

```
frsp=evalfr(sys,f)
```

wo ist, **sys** ein LTI Objekt
f eine komplexe Zahl.

Ein einfaches Beispiel folgt.

Beispiel 2.2.3

```
» f=2+i;
» evalfr(s,f)

ans =

   0.3984 - 0.1820i
```

2.2.4 Berechnung der logarithmischen Amplituden- und Phasengänge

Funktion bode

Berechnung der logarithmischen Amplituden- und Phasengänge

Syntax der Funktion

bode (sys)	- Darstellung und Berechnung der Amplituden- und Phasengänge des LTI Objektes **sys**
bode (sys1,…,sysN)	- Darstellung und Berechnung der Amplituden der LTI Objekte **sys1,…,sysN**
bode (sys,w)	- Darstellung und Berechnung der Amplituden- und Phasengänge des LTI Objektes **sys** für den Frequenzbereich **w.**
[mag,phase,w] = =bode (sys)	- Berechnung der Amplituden und Phasen des LTI Objektes **sys** und Speicherung in die Vektoren **mag,phase,w**

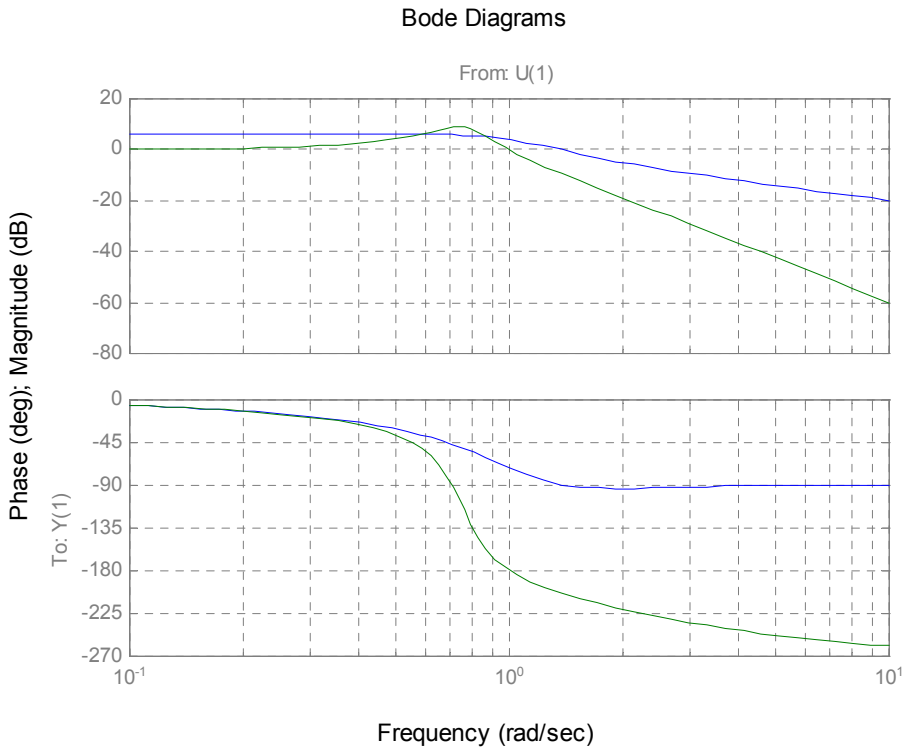
wo **sys** ein LTI Objekt ist,
sys1, sys2, …, sysN LTI Objekte sind,
w Parameter **w** einzugeben ist: **w = {w_{min} w_{max}},**
mag, phase, w Vektore sind, wo Amplituden, Phasen und Frequenzen gespeichert werden.

Die Anwendung der Anweisung wird auf folgendem Beispiel gezeigt.

Beispiel 2.2.4

Berechnen Sie logarithmische Amplituden – und Phasengänge für LTI- Objekte s, ss1.

```
» bode (s,ss1)
»
```



2.2.5 Berechnung von Amplituden und - Phasenrand

Funktion margin **Berechnung von Amplituden und - Phasenrand**

Syntax der Funktion

[Gm,Pm,Wcg,Wcp] = margin (sys)

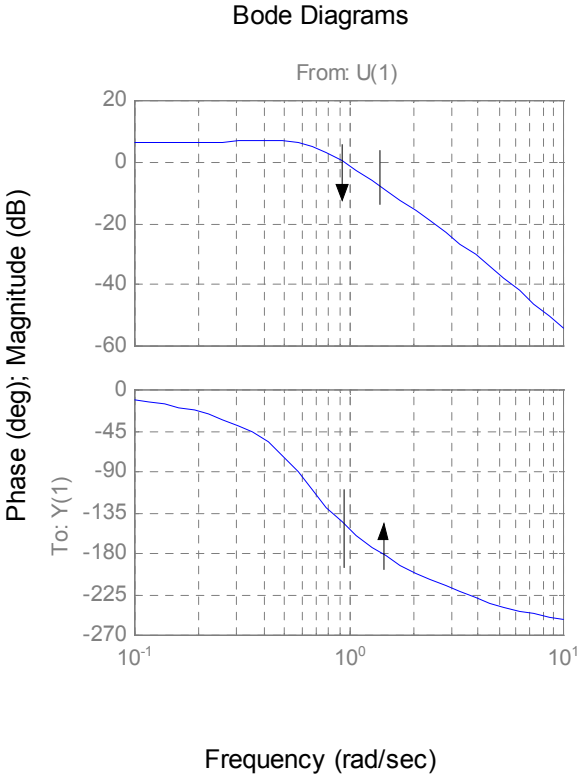
wobei **sys**… ein LTI Objekt ist, **Wcg**… Durchtrittfrequenz des Amplitudenrandes
Pm… Phasenrand **Wcp**… Durchtrittfrequenz des Phasenrandes
Gm… Amplitudenrand

Die Anwendung der Anweisung wird auf folgendem Beispiel gezeigt.

Beispiel 2.2.5

Berechnen Sie den Amplituden und – Phasenrand für $F(s) = \dfrac{2}{s^3 + 3s^2 + 2s + 1}$.

```
%B_marg1
A1=[1 3 2 1];
B1=[2];
figure(1)
s1=tf(B1,A1)
[Bm1,Pm1,Wcg1,Wcp1]=margin(s1)
bode(s1)
```




```
» s

Transfer function:
          2
--------------------
s^3 + 3 s^2 + 2 s + 1

» [Gm,Pm,Wcg,Wcp]=margin(s)

Gm =          Pm =

    2.5000      31.4238
Wcg =         Wcp =

    1.4142       0.9499
```

3 WURZELORTSVERFAHREN

Das Verfahren zur Bestimmung der Bewegung der Pole in der s-Ebene des geschlossenen Regelkreises in Abhängigkeit von einem veränderlichen Parameter z.B. dem Verstärkungsfaktor, ist als Wurzelortsverfahren bekannt. Es ist die zweite klassische Methode für Analyse und Synthese der linearen Regelkreise.

3.1 HINWEISE ZUM WURZELORTSVERFAHREN

Bei dieser Methode werden von den bekannten Eigenschaften des offenen Regelkreises die noch unbekannten Eigenschaften des geschlossenen Regelkreises hergeleitet.

Die Übertragungsfunktion eines linearen offenen Regelkreises hat allgemein folgende Form:

$$F_0(s) = \frac{b_m s^m + b_{(m-1)} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{(n-1)} s^{(n-1)} + \ldots + a_1 s + a_0} \qquad (3.1\text{-}1)$$

In folgende Form kann sie umgewandelt werden:

$$F_0(s) = k \frac{(s - s_{b1})(s - s_{b2})\ldots(s - s_{bm})}{(s - s_{a1})(s - s_{a2})\ldots(s - s_{an})} = \mathrm{k}\ \frac{\mathsf{Z_0(s)}}{\mathsf{N_0(S)}}, \quad \mathrm{k} > 0 \qquad (3.1\text{-}2)$$

dabei sind s_{bi} die Nullstellen (Wurzeln) und s_{aj} die Polstellen des offenen Kreises, welche komplexe Werte sein können und k ein Parameter ist.

Der **offene Regelkreis** wird durch folgende Übertragungsfunktion $F_0(s)$ beschrieben (siehe Bild 3.1.1a):

$$F_0(s) = R(s)F(s) = Gain \cdot R_0(s)F(s) = Gain * \frac{Z(s)}{N(s)}, \qquad (3.1\text{-}3)$$

dabei sind :

$F_0(s)$ ... die Übertragungsfunktion des offenen Regelkreises,

$F(s)$ ... die Übertragungsfunktion der erweiterten Regelstrecke,

R(s) ...die Übertragungsfunktion des Reglers,

R_0(s)... die Übertragungsfunktion der Grundstruktur des Reglers mit n_{RP} Polstellen und m_{RN} Nullstellen

Gain ... Gesamtverstärkung des Reglers (beeinflusst alle Anteile des Reglers), entspricht dem Parameter k .

N(s)... Polynom im Nenner der Übertragungsfunktion $F_0(s)$,

Z (s)... Polynom im Zähler der Übertragungsfunktion $F_0(s)$.

Die Pol- und Nullstellen der Übertragungsfunktion des offenen Regelkreises werden gebildet von den n_P Pol und m_N Nullstellen der **erweiterten Strecke** und den n_{RP} Pol und m_{RN} Nullstellen **des Reglers**.

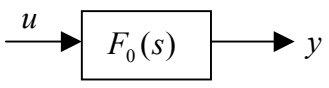


Bild 3.1.1a Offene Regelstrecke

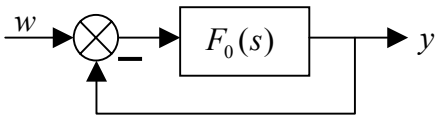


Bild 3.1.1b Geschlossene Regelstrecke

Die Übertragungsfunktion des **geschlossenen Regelkreises** für das **Führungsübertragungsverhalten** siehe (Bild 3.2.1b) lautet

$$F_{GE}(s) = \frac{F_0(s)}{1 + F_0(s)} = \frac{Gain \cdot R_0(s)F(s)}{1 + Gain \cdot R_0(s)F(s)} = \frac{Gain * Z(s)}{N(s) + Gain * Z(s)}. \qquad (3.1\text{-}4)$$

Die Polstellen dieser Funktion sind somit die Lösungen folgender Gleichung :

$$N(s) + Gain * Z(s) = 0 \qquad (3.1 - 5)$$

Die Nullstellen werden durch den Zähler von (3.1 - 4) bestimmt

$$Gain \cdot R_0(s)F(s) = Gain * Z(s) = 0 . \qquad (3.1 - 6)$$

Der geometrische Ort (in der komplexen Ebene) aller Lösungen (Wurzeln) der Gleichung (3.1 – 5) in Abhängigkeit von einem reellen Parameter Gain $\langle 0; \infty)$ wird als **Wurzelortkurve** (**WOK**) bezeichnet.
Die Anzahl der Lösungen ist somit gleich der Anzahl der einzelnen Bahnen (Zweige) in der WOK- Darstellung. Auf jeder Bahn existiert für ein bestimmtes Gain = k_i ein Funktionswert.
Weitere Aussagen über die Bahnen und Konstruktion der WOK sind in der Literatur [5,7,8] enthalten.

In diesem Versuch steht für die Darstellung und das Arbeiten mit WOK ein MATLAB / SIMULINK Programmpaket zur Verfügung [5]. Dieses stellt u.a. die komplexe Ebene mit den Polen, den Nullstellen und den Bahnen der WOK der Führungsübertragungsfunktion graphisch dar. Dabei werden die Null- und Polstellen des Reglers rot und die der Strecke blau gekennzeichnet. Auf jeder Bahn der WOK ist der aktuelle Wert des Parameters Gain = k_a gekennzeichnet. Für diesen k_a (Gain) wird die Führungsübergangsfunktion vom Programmpaket graphisch dargestellt.

Die Bedeutung der WOK bei der Auslegung von Regelkreisen:
Das Verhalten eines Regelkreises wird von der Lage seiner Pol- und Nullstellen der Übertragungsfunktion (3.1 – 4) in der komplexen Ebene bestimmt. Als Beispiel das Stabilitätskriterium (Hurwitz): alle Nullstellen des Nennerpolynoms müssen links der imaginären Achse liegen.
Diese und weitere Aussagen lassen sich über das Verhalten des geschlossenen Kreises an Hand der WOK in Abhängigkeit des Verstärkungsfaktors Gain machen. Für einen Regelkreis, bestehend aus einer gegebenen Strecke (Übertragungsfunktion) und der Art des Reglers (P,PID usw.), die durch die Anzahl der Null und Polstellen der Reglerübertragungsfunktion festgelegt wird, kann die Lage der Null und Polstellen des Reglers und die WOK dargestellt werden, wobei die WOK die Lage der Nullstellen und Pole der Strecke und des Reglers in der Führungsübertragungsfunktion des Regelkreises in Abhängigkeit des Parameters Gain =k darstellt. Für ein günstiges Regelverhalten lassen sich bestimmte anzustrebende Positionen der Pol und Nullstellen zu einander herleiten. Zum Beispiel bei einer schwingenden Strecke sollten die Nullstellen möglichst nahe an den Polstellen liegen und links von ihnen bleiben.

Im MATLAB / SIMULINK Programmpaket können die k-Werte (Gain) sowohl mittels Tastatur als auch durch Verschieben des aktuellen k-Wertes (Gain) entlang seiner Bahn mittels der Maus eingegeben werden.
Nach jeder Änderung von (Gain) oder eines Parameters des Regelkreises kann sofort die Übergangsfunktion des geschlossenen Regelkreises für Führungsverhalten dargestellt werden. Weitere Darstellungsarten, wie z.B. das Bodediagramm sind möglich.

Bei der Synthese wird durch die Pol und Nullstellen der Übertragungsfunktion $R_0(s)$ des Reglers die Struktur des Reglers definiert. Durch die Veränderung der Gesamtverstärkung des Reglers *Gain* wird die gewünschte Dynamik des geschlossenen Regelkreises gesucht.

Durch ein überlegtes, sukzessives Vorgehen lassen sich günstige Werte für die Pole und Nullstellen des Reglers finden. Dabei ist die Lage der Pole und Nullstellen des Reglers überlegt zu ändern.

Aus den gefundenen Werten der Pol- und Nullstellen des Reglers und dem k-Wert (Gain) für das gesuchte Regelverhalten (Führungsübergangfunktion) werden die Reglereinstellwerte berechnet.
Zur Bestimmung der Reglereinstellwerte muß der k-Wert (Gain) in jede Komponente des Reglers multipliziert werden.
Eine Beschreibung dieses Verfahrens mit einer Anwendung von MATLAB findet man in [5].

In MATLAB stehen zur Verfügung drei Funktionen Tab.3, die das Wurzelortsverfahren unterstützen.

rlocus	Stellt die Wurzelortkurve von einem LTI Objekts dar
C	Berechnet von dem gegebenen Punkt der Wurzelortkurve die entsprechende Verstärkung- Gain.
rltool	Ermöglicht Entwurf eines Regelkreises mit Hilfe des Wurzelortsverfahren

Tabelle 3

Anwendung der ersten zwei Funktionen (**rlocus, rlocus**) ist einfach, wir werden uns auf die Funktion **rltool** konzentrieren

3.2 ANWENDUNG DES MATLAB PROGRAMMPAKETS MIT DEM WURZELORTSVERFAHREN

Für die Synthese des Regelkreises wird in diesem Versuch die Funktion **rltool** benutzt. Die Übertragungsfunktion der erweiterten Regelstrecke und die Struktur des Reglers (z.B.: PI, PID, PD) werden dazu benötigt. Die Struktur des Reglers wird durch Eingabe der Pol- und Nullstellen der Übertragungsfunktion des Reglers $R_0(s)$ festgelegt.

Programm rltool

Start des Programms: in das Fenster MATLAB " **rltool** " eintragen und mit Enter starten. Dadurch öffnet sich das Fenster „**Root Locus Design**" (Bild 3.2.2). Den offene Regelkreis bilden: der Regler **K**, die Strecke **P** und die Messeinrichtung **H**. Der Vorfilter **F** wird für diesen Versuch auf 1 gesetzt.

K,P,H sind LTI – Objekte (Linear Time Invariant) diese müssen sich in dem Workspace (Arbeitsspeicher) befinden.

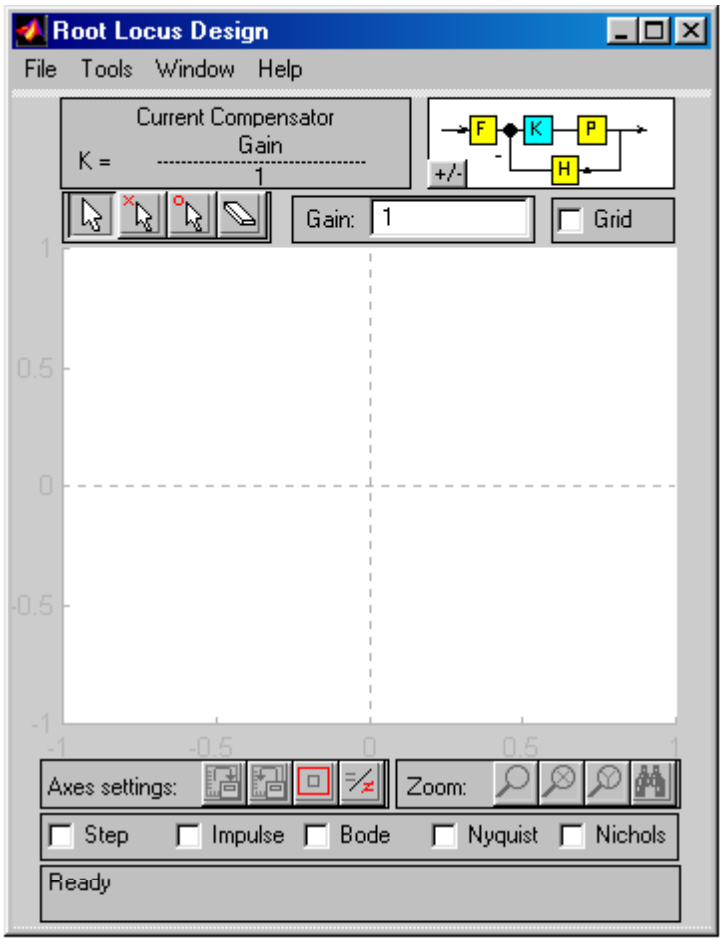


Bild.3.2.2 Fenster *„Root Locus Design "*

1) **Import der LTI Modelle ins Programm rltool**:
   - Öffnen von „File" in Fenster „Root Locus Design" (Bild 3.2.2)
     → Menu "File" (Bild 3.2.4a)
   - Öffnen von „Import Modell.."
     → Fenster "Import LTI Design Model" (Bild3.2.3)

Wesentliche Bestandteile des Fensters "Import LTI Design Model" (Bild3.2.3):

a) In dem Teil „Feedback Structure" wird durch die Taste „*Other*" die Rückführungsstruktur gewählt..
b) In dem Teil „ Import From " wird angewählt, woher das Modell transportiert werden soll.
c) In dem Teil „ Workspace Contents " ist eine Liste von LTI -Objekten, die durch anklicken benutzt werden.
d) In dem Teil „ Design model " wird der Name des geschlossenen Regelkreissees eingegeben. Durch Klick auf den entsprechenden Pfeil → bei der Bezeichnung P,H,F wird diesem Block das markierte LTI Model aus „ Workspace Contents " zugeordnet.

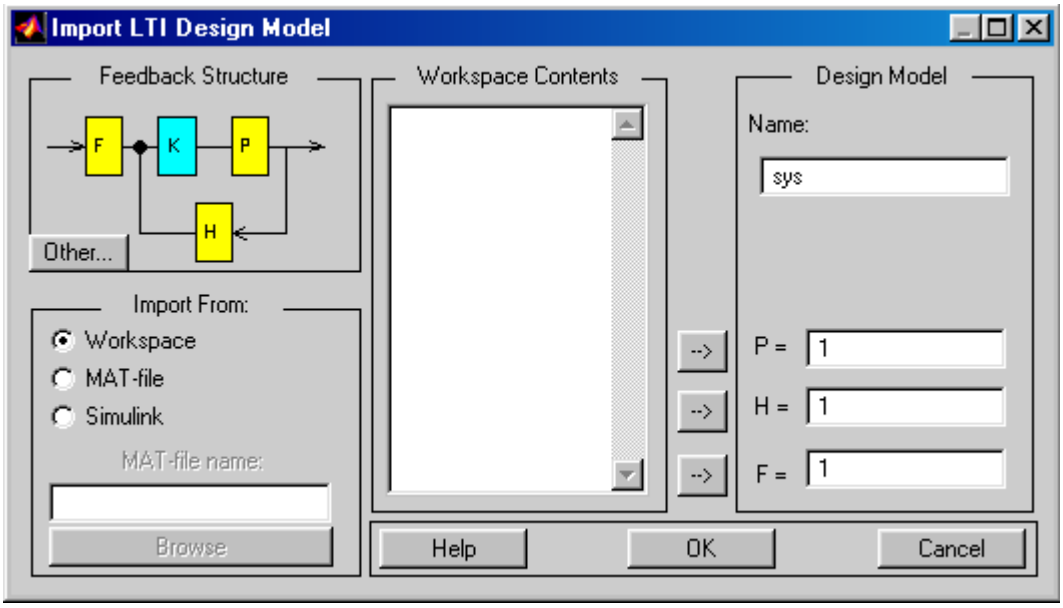
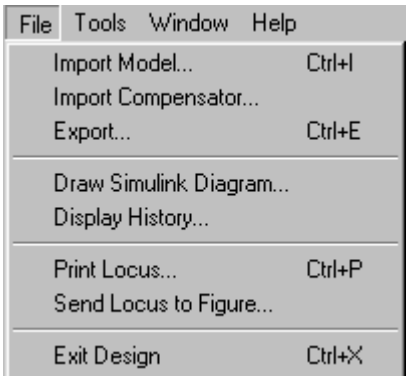


Bild.3.2.3 *Fenster „Import LTI Design Model“*

Mit „OK“ wird die Aktion durchgeführt, und das Fenster „ *Import LTI Design Model* “ wird geschlossen.

a)



b)

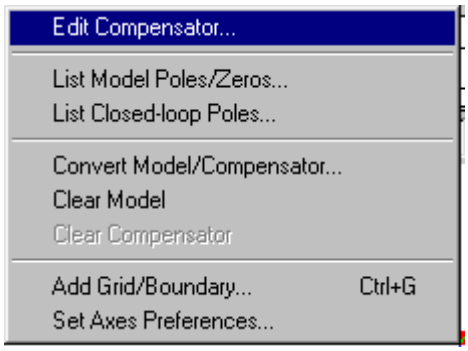


Bild 3.2.4a *Menu „File“*

Bild 3.2.4b *Menu „Tools“*

2) Das Interaktive Wurzelortverfahren:

Nach dem Import öffnet sich das Fenster „ Root Locus Design UOB1“, in dem die Wurzelortskurve dargestellt ist (Bild 3.2.6). Durch Anklicken von „Tools“ öffnet sich das in Bild 3.2.4b dargestellte Rollmenü. Durch das Klicken auf „Edit Compensator“ ist es möglich, einen Kompensator (Regler) zu editieren (Bild 3.2.5).

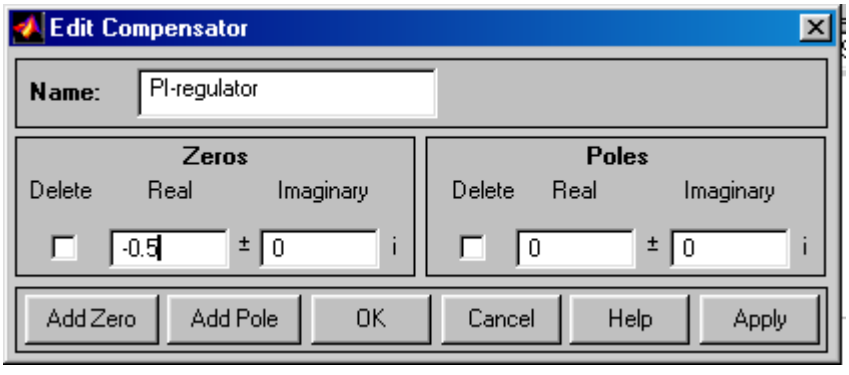


Bild 3.2.5 *Fenster „Edit Compensator“*

In dem Fenster „Edit Compensator“ ist es möglich den Namen des Kompensators (Reglers) und die Pole und Nullstellen einzugeben (siehe Bild 3.2.5). Im Rollmenü Tool ist es möglich, die „List Model Poles/Zero“ zu öffnen. Auf dem Bild 3.2.6a ist eine solche Liste der

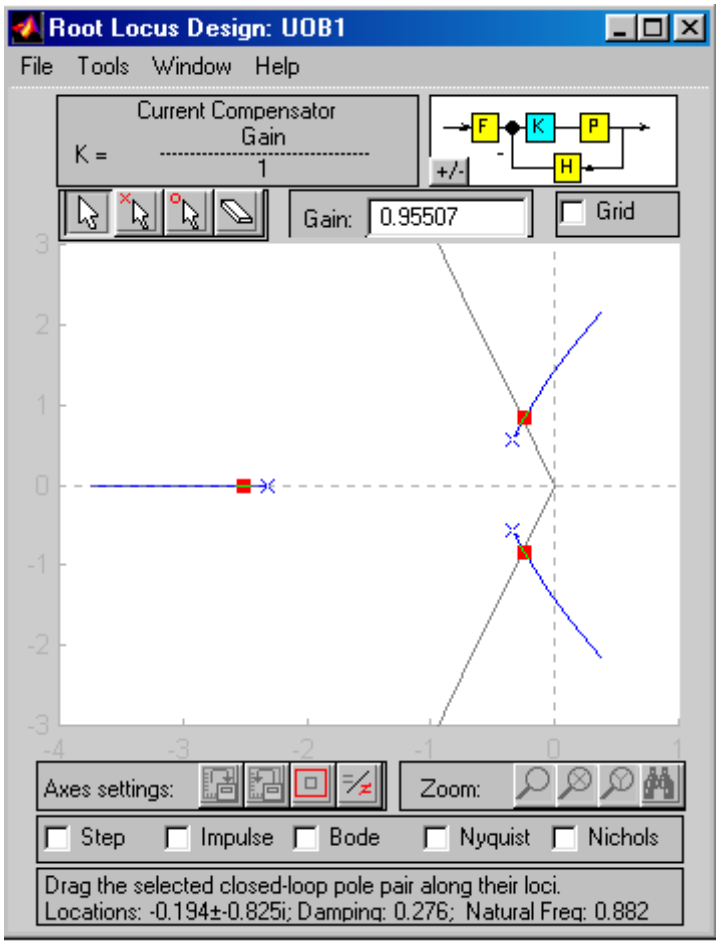


Bild 3.2.6 *Root Locus design*

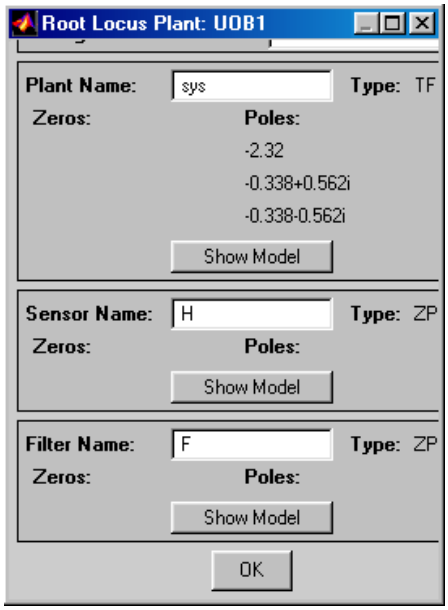


Bild 3.2.6a *Fenster „Root Locus Plant“*

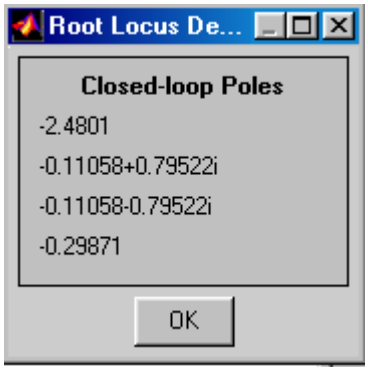


Bild 3.2.6b *Fenster List Closed –loop Poles*

Nullstellen und Pole der Strecke (sys), der Messeinrichtung (H) und des Filters (F) dargestellt.

Im Rollmenü „Tools“ kann man u.a. folgende Fenster öffnen :

- " *List Closed-loop Polles* " , (Bild 3.2.6b)
- " *Add Grid / Boundry*" (Bild 3.2.6c) dient zur Eingabe der relativen Dämpfung ξ (*Damping Ratio*), Einschwingzeit (*Settlig Time*), Überschwingweite (*Peak Overshoot*). Die relative Dämpfung ξ, die als eine Gerade in der komplexen Ebene „s“ dargestellt ist, geht vom Anfangspunkt der Koordinaten aus (Bild 3.2.6)
- „Grid and Constrain Option“ (Bild 3.2.6c)
- „Set Axes Preferences“ (Bild 3.2.6d)

Bild 3.2.6c *Fenster Grid and Constrain Option*

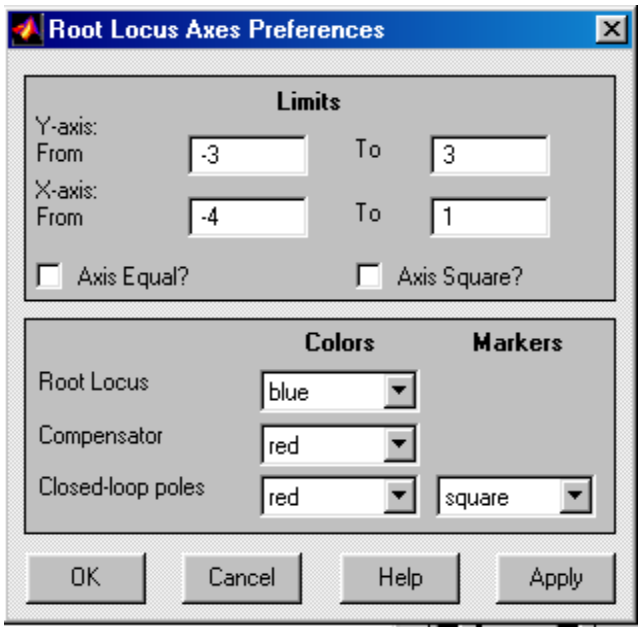


Bild 3.2.6d *Fenster Set Axes Preferences*

3) Durchführung des Wurzelortverfahrens

Nach dem Import des Streckenmodells (Übertragungsfunktion) werden die Reglereinstellwerte nach folgender Verfahrensweise für einen PID - Regler bestimmt

1) ***Die I- und D-Anteile des Reglers werden Null gesetzt, indem die Nullstelle und Polstelle nicht hinzugefügt werden (Bild 3.2.5).***
2) ***Die Verstärkung „Gain" wird so eingestellt, damit der geschlossene Regelkreis die geforderte Schnelligkeit (die Einschwingzeit T_R) besitzt.***
3) ***Der D-Anteil wird eingestellt durch das Einfügen einer Nullstelle in die Übertragungsfunktion des Reglers (Bild 3.2.5). Die Lage der Nullstelle wird variiert bis die gewünschte Dämpfung des geschlossenen Regelkreises erreicht wird (PD - Regler).***
4) ***Die bleibende Regelabweichung*** $e_w(\infty) = \lim_{t\to\infty} e_w(t) = 0$ ***wird durch Zufügung von dem Pol*** $s_R = 0$ ***beseitigt (PI-Regler).***
5) ***Um einen PID –Regler zu erstellen, ist es noch nötig, eine zweite Nullstelle hinzuzufügen.***
6) ***Wenn eine schwingungsfähige Strecke zu regeln ist, werden die beiden Nullstellen des Reglers in die Nähe der komplexen Pole der Übertragungsfunktion der Strecke gelegt.***

Vor der Beendigung des Programms „rltool" sind die Ergebnisse in die *Workspace* mit Hilfe der Anweisung „*Export*" in dem Rollmenü „*File*" zu transportieren.

Auswertung :

Das dynamische Verhalten des Regelkreises kann mit Hilfe der Sprungantwort (*Step*), der Impulsfunktion (*Impulse*) oder durch die Ortskurve des Frequenzgangs (*Nyquist*) bzw. der Frequenzkennliniendarstellung - Bodediagramm (*Bode*), (*Nichols*) dargestellt werden. Diese dynamische Charakteristiken werden einfach durch Öffnen der entsprechenden Fenster in „*Root Locus Design*“ (siehe Bild 3.2.6) dargestellt.
Die Überschwingweite *%P* in % (*Peak Overshoot*), die Einschwingzeit T_s (*Settlig Time*) und die relative Dämpfung ξ können mit Hilfe des Fensters „*Fenster Grid and Constrain Option“ (Bild 3.2.6c)* bestimmt werden.

Diese Synthese wird an Beispiel 1 gezeigt.

Beispiel 3.1 Die Übertragungsfunktion der Strecke $F_U(s)$ ist: $F_U(s) = \dfrac{1}{s^2 + 2s + 1}$.

Gesucht wird die Reglereinstellung für:

a) I Regler, ***%P <20%,***
b) PI-Regler ***%P <20%*** und $\boldsymbol{T_s < 6\ sec}$.

Lösung: a) Für die gewünschte Überschwingweite *%P* wird gewählt ξ=0,6.
Die Übertragungsfunktion $R_0(s) = \dfrac{1}{s}$, das heißt , der Pol des Reglers liegt bei $s_R = 0$. Wo die Wurzelortskurve die Gerade der Dämpfung schneidet, dieser Gain-Wert erzeugt die gewünschte Überschwingweite ξ=0,6.
Die gesuchte Gesamtverstärkung des Reglers (siehe Bild 3.2.6e,f): *Gain=0,323*

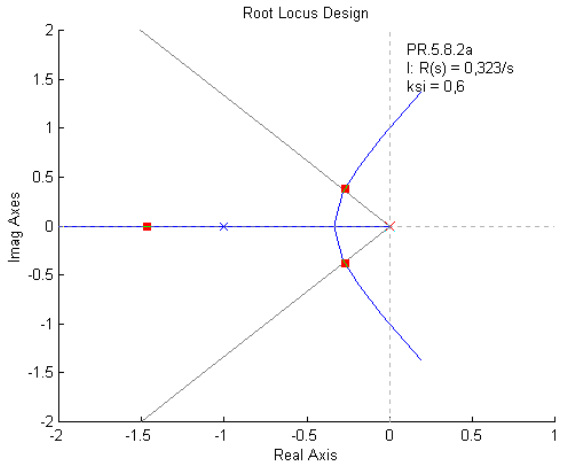


Bild 3.2.6e *Wurzelortskurve für I Reg*

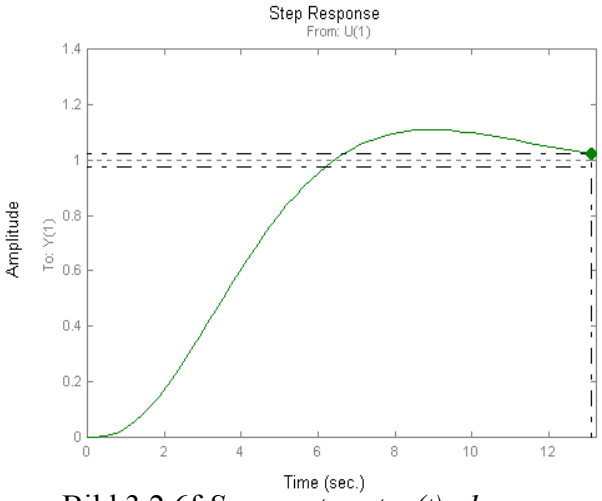


Bild 3.2.6f *Sprugantwort w(t)=1*

Lösung b) ***Für PI Regler ist die Einschwingzeit*** $\mathbf{T_s}$ ***<6sec.***

Der PI- Regler: $R_0(s) = \dfrac{r_0 s + r_1}{s} = \dfrac{Gain \cdot (s - s_{RB})}{s}$. Die Nullstelle s_{BR} der Übertragungsfunktion $R_0(s)$ wurde auf $s_{RB} = -0{,}73$ eingestellt. Die gesuchte Gesamtverstärkung des Reglers Gain liegt in dem Punkt, in dem sich die Gerade für $\xi = 0{,}6$ und die Gerade für Ts=6sec

mit der Wurzelortskurve schneiden, siehe Bil

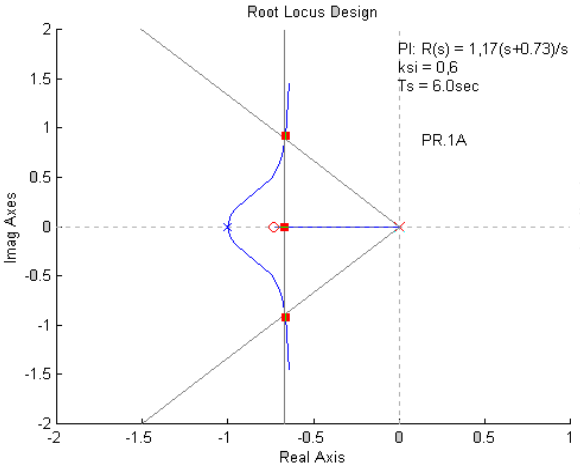


Bild 3.2.6a *Wurzelortkurve für PI Reg.*

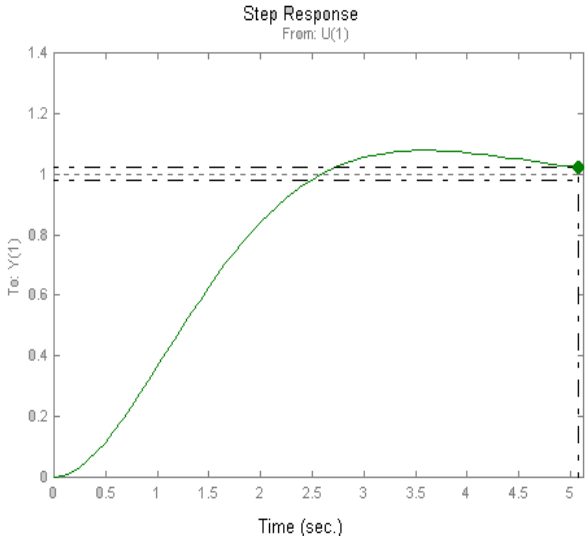


Bild 3.2.6b *Sprugantwort w(t)=1*

Der PI Regler wurde folgend eingestellt: $R(s) = Gain\frac{s - s_{BR}}{s} = 1{,}17\frac{s + 0{,}73}{s} = 1{,}17 + \frac{0{,}85}{s}$.

> B*emerkung: Bei der Versuchsdurchführung werden die dimensionslose eingegeben. Aber alle Berechnungen erfolgen in Sekunden.*

4 OPTIMIERUNGSANWEISUNGEN

Unter Optimierung versteht man die Ermittelung des Minimums oder Maximums einer Funktion endlich vieler Variablen mit oder ohne Nebenbedienungen. Für eine Optimierungsaufgabe kann der Ansatz geschrieben werden

$$\min_x f(\mathbf{x}), \quad \mathbf{g}(\mathbf{x}) = \mathbf{0},$$

wo ist, $f(\mathbf{x})$ eine Funktionsgleichung von *n*- Variablen,
$\mathbf{x}$ *n*- dimensionaler Vektor der Variablen x_i,
$\mathbf{g}(\mathbf{x}) = \mathbf{0}$ die Nebenbedienungen.

Dazu stehen in MATLAB folgende Funktionen zur Verfügung, siehe Tab.4 und noch andere.

Anweisung	Funktion
fmins fminsearch	Minimierung ohne Nebenbedienungen $\min_x f(\mathbf{x})$
fminbnd	Minimierung mit den Nebenbedienungen $\mathbf{x}_L < \mathbf{x} < \mathbf{x}_H$
fmincon	Minimierung mit nichtlinearen Nebenbedienungen
fminimax	Minimax- Optimierung

Tab.4

4.1 MINIMIERUNG EINER FUNKTION MIT MEHRER REALEN VARIABLEN

Um einfache Optimierung durchführen zu können, wird aus dem „Optimization Toolbox“ für Minimierung die Funktion **fmins** erklärt und beschrieben.

Funktion fmins

Minimierung einer Funktion mit mehrerer realen Variablen.

Syntax der Funktion

```
fmins('fun',x0)
fmins('fun',x0,options)
```

wobei **'fun'** der Name der Funktion (Datei.m) ist, die minimiert werden soll,
x0 der Parametervektor ist, damit die Minimierung gestartet wird,
x der Parametervektor ist, wo die Ergebnisse der Minimierung gespeichert sind
options Steuerparametervektor ist:
options(2) …definiert die kontrollierbare Genauigkeit fun(x) a x
options(14)… definiert die maximale Zahl der Minimierungsschritten

Als Beispiel einer Programmerarbeitung sollte ein Programm für parametrische Identifikation dienen.

4.2 IDENTIFIKATION - DATENAUSWERTUNG

In unserem Studientext verstehen wir unter Identifikation die Bestimmung der Parameter und Struktur einer Übertragungsfunktion auf Grund der Messung an einer Realen Strecke.

4.2.1 Voraussetzungen, Modellstruktur

Es wird vorausgesetzt, dass das gemessene Signal mit einem Messrauschsignal überlagert ist

$$y = y_0 + \nu,$$

wo y gemessenes Signal mit Messrauschsignal,
y_0 Nutzsignal,
ν Messrauschsignal, das den mittleren Wert $E\{\nu\} = 0$ hat.

Die Struktur der Übertragungsfunktion wird gewählt. Die meist gebrauchten sind:

a) $F_0(s) = \dfrac{Ke^{-T_D s}}{\tau s + 1}$, K …Verstärkung, T_D …Totzeit, τ … Zeitkonstante,

b) $F_1(s) = \dfrac{K}{(\tau_1 s + 1)(\tau_2 s + 1)}$, K …Verstärkung, τ_1, τ_2… Zeitkonstanten

c) $F_2(s) = \dfrac{K}{(\tau s + 1)^n}$, K… Verstärkung, τ …Zeitkonstante,
n … gewählte Streckeordnung

d) $F_3(s) = \dfrac{K}{(\tau_1 s + 1)\left[(\tau s)^2 + 2\xi\tau s + 1\right]}$, K, τ_1, τ, ξ...eine schwingungsfähige Strecke

e) $F_4(s) = \dfrac{K}{(\tau_1 s + 1)(\tau_2 s + 1)(\tau_3 s + 1)}$, K, τ_1, τ_2, τ_3

a) $F_5(s) = \dfrac{b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}$, **P** = [$b_0$, a_0, a_1, a_2, a_3]
Parameter – Anfangsvektor
soll Stabilität sichern!

g) $F_6(s) = \dfrac{b_0 + b_1 s}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0}$, **P** = [$b_0$, b_1, a_0, a_1, a_2, a_3]
Parameter – Anfangsvektor
soll Stabilität sichern!

Auf Grund der Messung und unseren Erfahrungen wird die Struktur der Übertragungsfunktion gewählt. Zum Beispiel

$$F(s) = \frac{K}{(Ts+1)^n} = \frac{x(1)}{[x(2)s+1]^n} = \frac{B(s)}{A(s)} \quad \ldots \qquad (4\text{-}1)$$

T … Zeitkonstante, K… Verstärkungsfaktor, x(1), x(2) die zu suchende Parameter des Modells

4.2.2 Arbeitspunkt, Struktur der Optimierung

Weil bei der Übertragungsfunktion nach der Definition die Anfangsbedingungen Null sein muss, ist in dem ersten Schritt ein Arbeitspunkt zu finden.
1) Für diesen Arbeitspunkt müssen die Messdaten nach folgenden Formeln umgerechnet werden:

$$\Delta y(k) = y(k) - y_0 = yP(k) = y(k) - Py\,,\ \Delta u(k) = u(k) - u_0 = uP(k) = u(k) - Pu\,,$$

y(k), u(k) … gemessene Regel – und Stellgröße,
y_0, u_0 … Arbeitspunkte der Regel – und Stellgröße
yP(k), uP(k) … Programmvariable für gemessene Regel – und Stellgröße
Py, Pu … Programmvariable für Arbeitspunkte der Regel – und Stellgröße

2) Zwei Parameter werden gesucht
$x(1) = K, x(2) = T$.

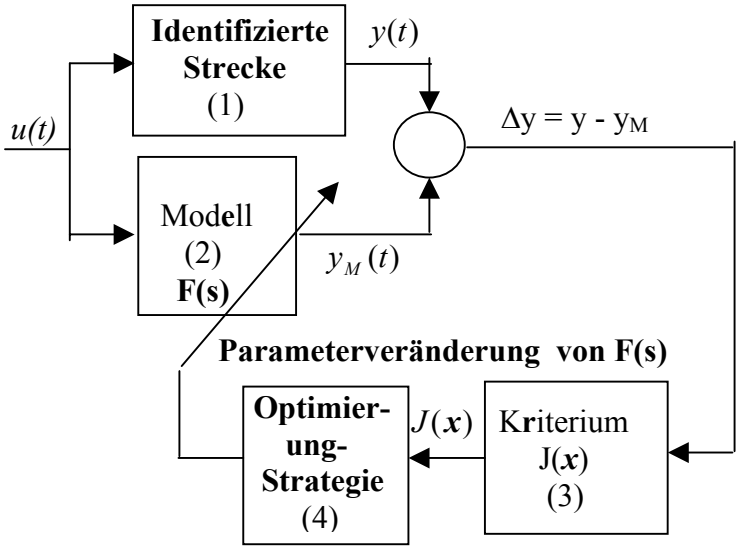


Bild 4.5 *Struktur der Identifikation*

Der Hauptgedanke besteht darin, dass bei gleichen Eingangssignalen $u(t)$ die Ausgangssignale aus der identifizierten Strecke-Block (1) (eine Messungsdatei) und des Modells (2) miteinander verglichen werden, siehe Bild 4.5.Für die entstehende Abweichung Δy zwischen gemessenen $y(t)$ und berechneten $y_M(t)$ dynamischen Verlauf der Regelstrecke soll der für das quadratische Kriterium (3) gelten:

$$J(X) = \sum_k (y(k) - y_M(k))^2 = \sum_k (yP(k) - yi(k))^2 = \sum_k (\Delta y(k))^2 = \rightarrow Min$$

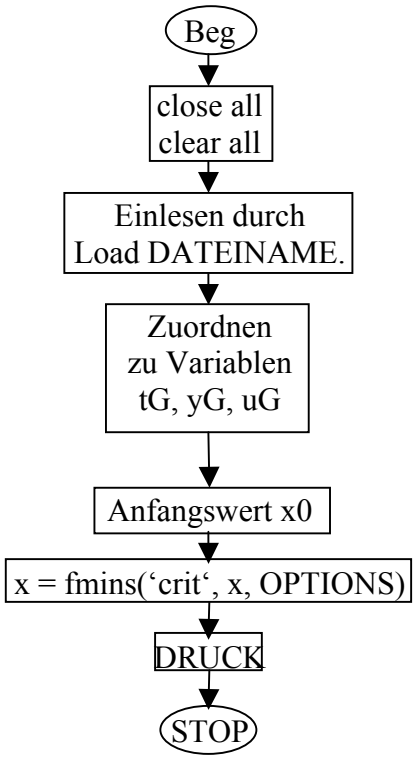
Diese Minimierung wird durch Parametereinstellung des Modells, so genannte Optimierungs-Strategie (4), erzielt. Für diese Minimierung wird die Funktion **fmins** benutzt.

4.2.3 Beschreibung des Identifikation- Programmes

1) Eine Messung wurde auf einer realen Strecke durchgeführt. Die gemessenen Daten werden für Arbeitspunkt umgerechnet und sind dann unter den Dateiname **`tyu1.m`** zugänglich.
2) Die Datei hat die Daten in der Matrix-Variable **`dat1`** gespeichert. In der ersten Zeile ist die Abtastzeit, in der zweiten Zeile ist das gemessene Ausgangsignal und in der dritten Zeile ist das Eingangsignal.
3) Die Struktur des mathematischen Models ist die Übertragungsfunktion
$F(s) = \dfrac{K}{(Ts+1)^n}$, ***K***…Verstärkung, ***T*** … Zeitkonstant, ***n***…gewählte Streckeordnung
4) Als Kriterium wurde das quadratische Kriterium genommen:
$J = \sum_{i=1}^{N} (y_i - yM_i)^2 \rightarrow Minimum$. $y_i \cdots$ Streckeausgang $yM_i \cdots$ Modellausgang

5) Diese Summe wird in der Funktion function crit(x) berechnet.
6) Für die Minimierung ist die MATLAB's Funktion **fmins** benutzt**.**
   Vektor `x` enthält die zu optimierenden Parameter. Das Unterprogramm, dass das Kriterium berechnet, ist **function crit(x).**

Für von uns gewähltes Beispiel kann ein Programmablauf gezeichnet werden.




```
function f=critT(x)
global tG yG uG N
K=x(1);
T=x(2);
xT=[T 1];
A=xT;
for i=1:N;
   A=conv(A,xT);
end
sys=tf(K,A);
lsim(sys,uG,tG);
[yi,ti]=lsim(sys,uG,tG);
f=sum((yG-yi).*(yG-yi));
```

critT.m - stáhni soubor

```
%idT.m, ohne Suchen von %Ar-
beitspunkt
close all
clear all
global tG uG yG N
load tyu1
tG=dat1(1,:)';
uG=dat1(3,:)';
yG=dat1(2,:)';
N=2
T=0.5
K=1
x=[T 1];
A=x;
for i=1:N;
   A=conv(A,x);
end
s=tf(K,A)
roots(A)
x=[K T]
critT(x)
disp('running...')
OPTIONS(2)=1e-1;
OPTIONS(14)=50;
x=fmins('critT',x,OPTIONS);
disp('Optimierter Vektor x')
x
disp('Kriteriumwert J')
critT(x)
Ax=[x(2) 1];
A=Ax;
for i=1:N;
   A=conv(A,Ax);
end
K=x(1);
roots(A)
tf(x(1),A)
sys=tf(x(1),A);
[yi,ti]=lsim(sys,uG,tG);
plot(tG,yG,ti,yi)
```

idT.m - stáhni soubor

4.3 OPTIMALE EINSTELLUNG EINES PID- REGLERS

In der Kapitel 4.2 haben wir ein Identifikationsverfahren als Optimierungsaufgabe vorgestellt. Ähnlich kann die optimale Parametereinstellung eines PID- Reglers gesucht werden.

4.3.1 Struktur der Optimierung

Dazu kann die allgemeine Optimierungsstruktur auf dem Bild 4.6 benutzt werden. Das Modell des geschlossenen Regelkreises mit Störungen ($d(t)$,$d_U(t)$) und der Sollwert $w(t)$ anschließlicht der Kriteriumsberechnung bildet ein SIMULINK Programm. Die Optimierunkstrategie wird mit Hilfe der Funktion **fminsearch** das MATLAB- Programm durchgeführt. Auf dem Bild 4.7a,b ist der vereinfachte Programmablaufplan dargestellt.

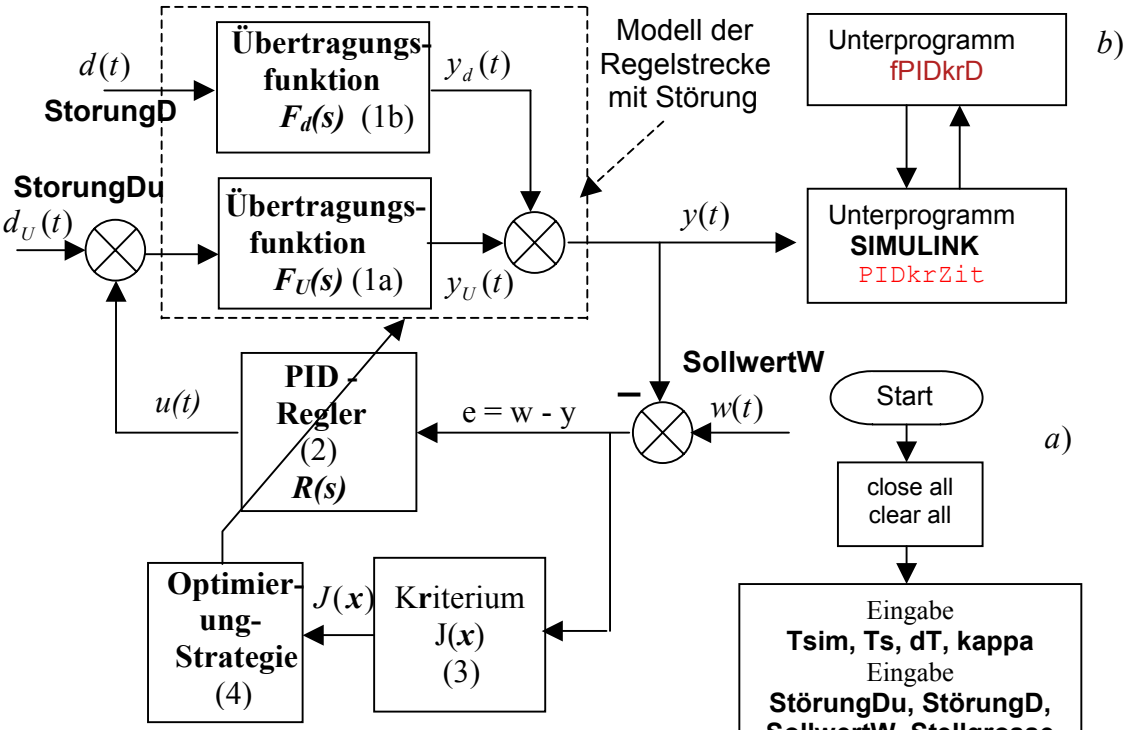
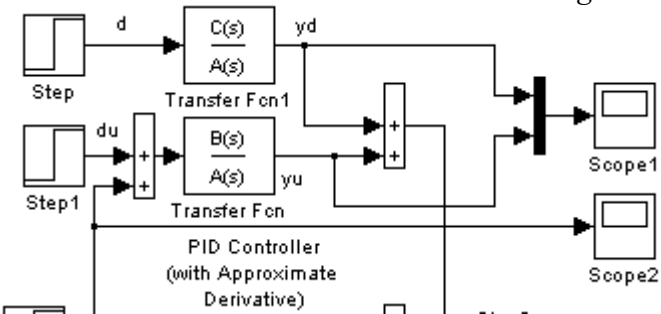


Bild 4.6 *Struktur der optimalen PID- Reglereinstellung*

Bild 4.7 *Programmstruktur*

Das SIMULINK Programm **PIDkrZit,** wo das Kriterium berechnet wird, ist auf dem Bild 4.8 dargestellt. Das SIMULINK Programm **PIDsimZit** und die Auflistung des MATLAB - Programms **PIDoptZit** sind auf den Bildern 4.9, 4.10 dargestellt. Die Simulationsparameter **Tsim, Ts, dT, kappa** und **StörungDu, StörungD, SollwertW, Stellgrosse** und der Anfangsvektor **x0** werden im Programm **PIDoptZit** eingegeben.



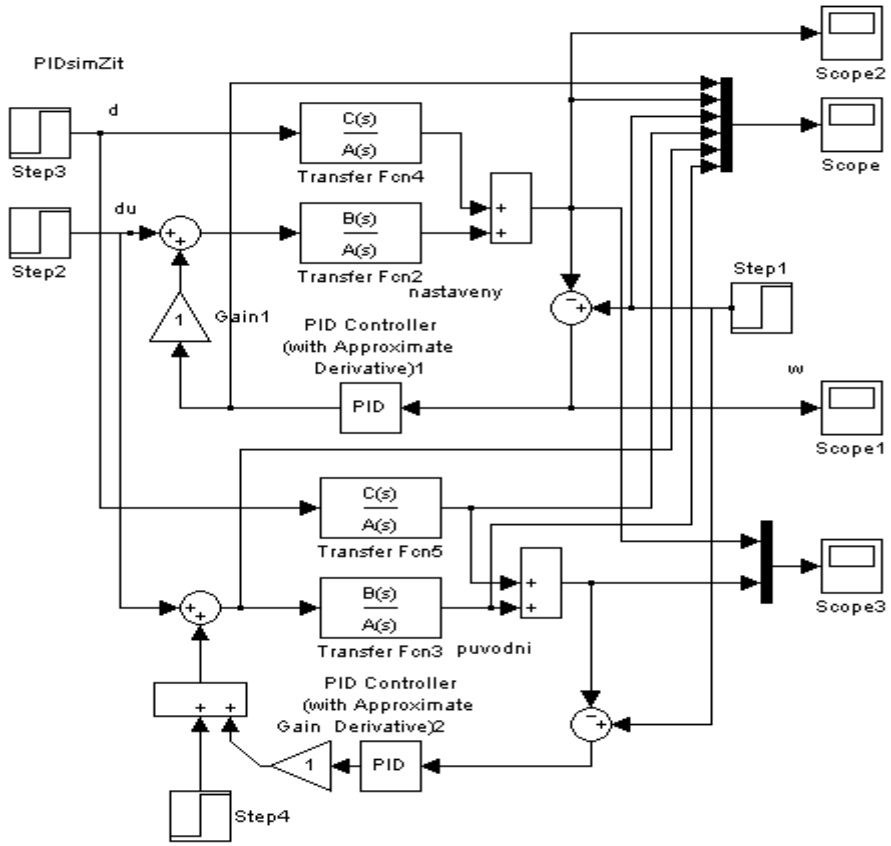


Bild 4.9 SIMULINK Programm PIDsimZit

4.3.2 Das Programm PIDoptZit

Das Kriterium bei der Optimierung hat die Form

$$J(\mathbf{x}) = J(P,I,D) = \int_0^{Tsim} \{e(t)^2 + \kappa \cdot [u(t) - u(\infty)]^2\} dt \rightarrow MIN\ .$$

Wo ist,
- e(t) die Regelabweichung,
- u(t) die Stellgröße,
- u(∞) der Beharrungszustand der Stellgröße,
- κ ein wählbarer Koeffizient, damit die Dämpfung der Stellgröße gesichert wird,
- P, I, D einstellbare Parameter des PID Reglers.

```
%PIDoptZit:  integral{e^2 + kappa * [u-u(unendlich)^2)]}
clear all;
close all;

global P I D Tsim
A=[4 8 5 1];
B=[0 0 0 1];
C=[0 0 0 1];
Ts = 0.05;
Tsim=50;           %Simulationszeit
dT=0.01;           %Schritt der Simulation
Kappa=1
P0=1; I0=0.5;D0=0.5; N=20;     %Anfangeinstellung des PID Reglers
StorungDu=0;
%US=B(n)/A(n)*StorungDu;       %Umrechnung StorungDu
StorungD=0;
%US=C(n)/B(n)*StorungDu;       %Umrechnung StorungD
SollwertW=1;
Stellgrosse=0;
n=length(A);
US=-A(n)/B(n)*SollwertW;       %Umrechnung SollwertW
P=P0; I=I0; D=D0;
sim('PIDkrZit',Tsim);
disp('Wert des quadratischen Kriteriums für x0');
krit1
pause
x=[P I D]
OPTIONS=optimset('TolFun',1e-10,'MaxFunEvals',50);
x = fminsearch('fPIDkrD',x,OPTIONS)
sim('PIDkrZit',Tsim);
disp('Wert des quadratischen Kriteriums für den optimalen PID Regler:');
krit1
StorungDu=0;
SollwertW=1;
StorungDu=0;
sim('PIDsimZit',Tsim);
PIDsimZit;
```

```
%function f=fPIDkrD(x)
function f=fPIDkrD(x)
global P I D Tsim
P=x(1);
I=x(2);
D=x(3);
sim('PIDkrZit',Tsim);
f=krit1;
```

Bild 4.10 Auflistung des MATLAB's Programms **PIDoptZit**

5 FUZZY LOGIC CONTROL

Die Entwicklung des Fuzzy-Reglers wird in der Software von MATLAB durchgeführt. Dazu dient die Fuzzy Logic Toolbox [4]. In MATLAB werden Simulationen und Real-Time Messungen an physikalischen Systemen mit Hilfe der Simulationssoftware SIMULINK realisiert. Die nachfolgende Beschreibung erhebt keinesfalls den Anspruch die Toolbox Fuzzy Logic, SIMULINK ausführlich zu beschreiben. Für ausführliche Informationen wird auf die Literatur [1, 2,3, 4] verwiesen. Eine Hilfefunktion in dem **MATLAB Command Windows** (in der Menüleiste **Help**) steht Ihnen bei jeder Arbeit zur Verfügung. Help ist in englischer Sprache und wird durch Klicken geöffnet.

5.1 DIE IMPLEMENTIERUNG DER FUZZY LOGIK UND DIE REGELUNG IN SIMULINK

Das Simulationsprogramm SIMULINK hat in dem Rollmenü "Simulink Library Browser" ein Fuzzy Logic Toolbox, das bei der Aktivierung zwei **Fuzzy – Zentraleinheiten** als **Fuzzy-Programmblöcke** enthält:

Fuzzy Logic Controller
Fuzzy Logic Controller with Ruleviewer (wird nicht erklärt)

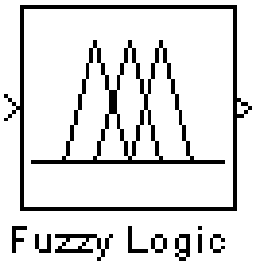


Bild 5.1

Der Block "Fuzzy Logic Controller" (siehe Bild 5.1) wird im SIMULNK als normaler Block verwendet. Dadurch können verschiedene Typen von Fuzzy Regler (**PD, PI, PI+PD, PI+D**) realisiert werden. In dem Block „Fuzzy Logic Controller" sind Fuzzyfizierung, Defuzzyfikation und Inferenz enthalten. Das Aufstellen der Fuzzy-Mengen für die Ein- und Ausgänge, die Zugehörigkeitsfunktionen und die Fuzzy-Regelmengen wird für den konkreten "Fuzzy Logic Controller" mit Hilfe des Graphical User Interface (GUI) durchgeführt.

Eine Struktur des geschlossenen Regelkreises in SIMULINK mit einer Strecke zweiter Ordnung und mit einem einfachen Fuzzy PD-Regler, der einen diskretisierten Eingang hat, ist im Bild 5.2 dargestellt. Der Fuzzy PD-Regler hat zwei Eingänge: die Regelabweichung e(t) und die Änderung der Regelabweichung pro Abtastperiode de(t). Der Ausgang des Reglers ist u(t). Die Abtastperiode beträgt T = 0.05s.

Im Regelkreis sind Sollwert- und Störgrößenänderungen möglich (Sollwert wird über Block Constant und Störgröße über Block Stepp eingegeben). Am Regler können an den Eingängen des "Fuzzy Logic Controllers" die Koeffizienten K_e = Ke und K_{de} = Kde und am Ausgang die Reglerverstärkung K_u = Ku eingestellt werden. Durch diese drei Koeffizienten wird der Fuzzy PD-Regler eingestellt.

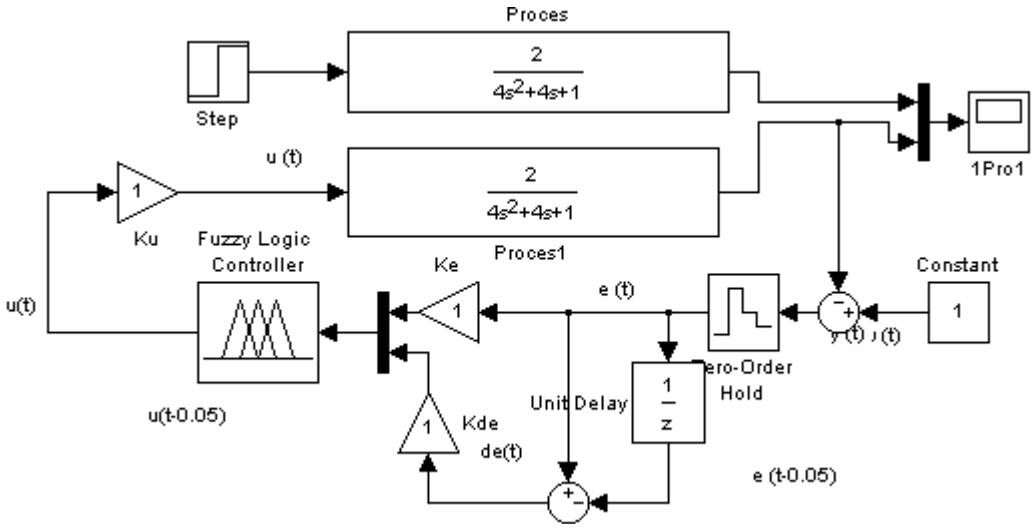


Bild 5.2. *Geschlossener Regelkreis im SIMULINK mit einem Fuzzy PD-Regler*

Folgendes ist zu beachten:

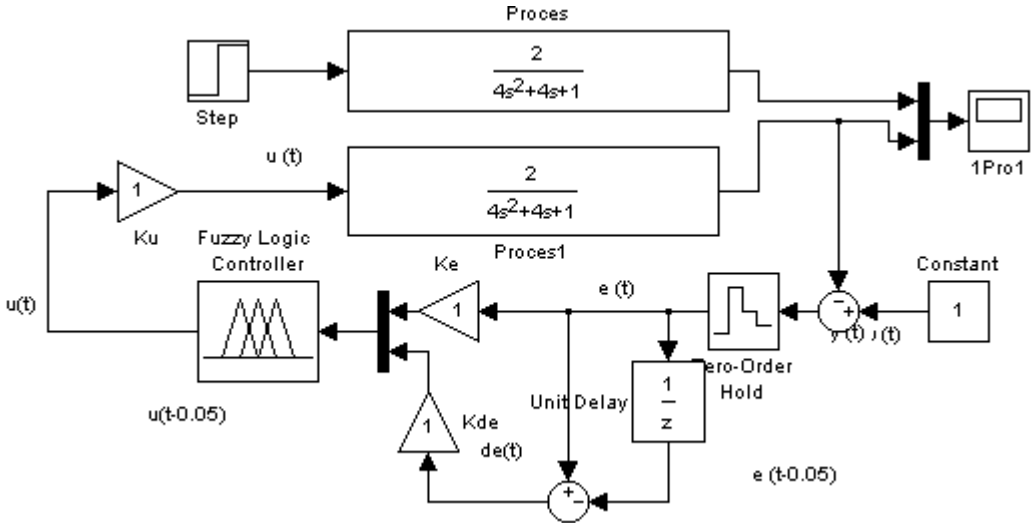


Bild 5.2. *Geschlossener Regelkreis im SIMULINK mit einem Fuzzy PD-Regler*

Folgendes ist zu beachten:

1) Der Typ des Fuzzy Reglers wird durch die Kopplungen mit dem "Fuzzy Logic Controller" und die zugeführten Eingänge bestimmt.

2) Die Eigenschaften des konkreten "Fuzzy Logic Controllers" werden mit Hilfe des Graphical User Interfaces (GUI) in SIMULINK eingegeben. (Siehe Abschnitt 5.2)

5.2 ENTWURF EINES FUZZY REGLERS MIT DEM FUZZY INTERFACE SYSTEM

Bei dem Entwurf eines "Fuzzy Logic Controller" ist es nötig folgendes zu definieren: die Ein- und Ausgänge und deren Fuzzy-Mengen, die Zugehörigkeitsfunktionen, die Fuzzy-Regelmenge, die Inferenz- und Defuzzifikationsmethode. Diese Forderungen werden mit Hilfe der interaktiven Umgebung **Fuzzy Inference System (FIS)** erfüllt. Als Blockbild ist das **FIS** in Bild 5.3 dargestellt.

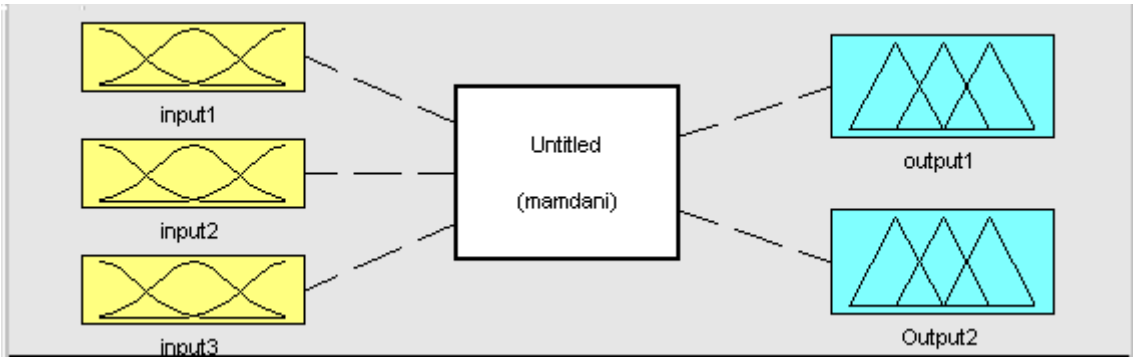


Bild 5.3 *Fuzzy Inference System FIS*

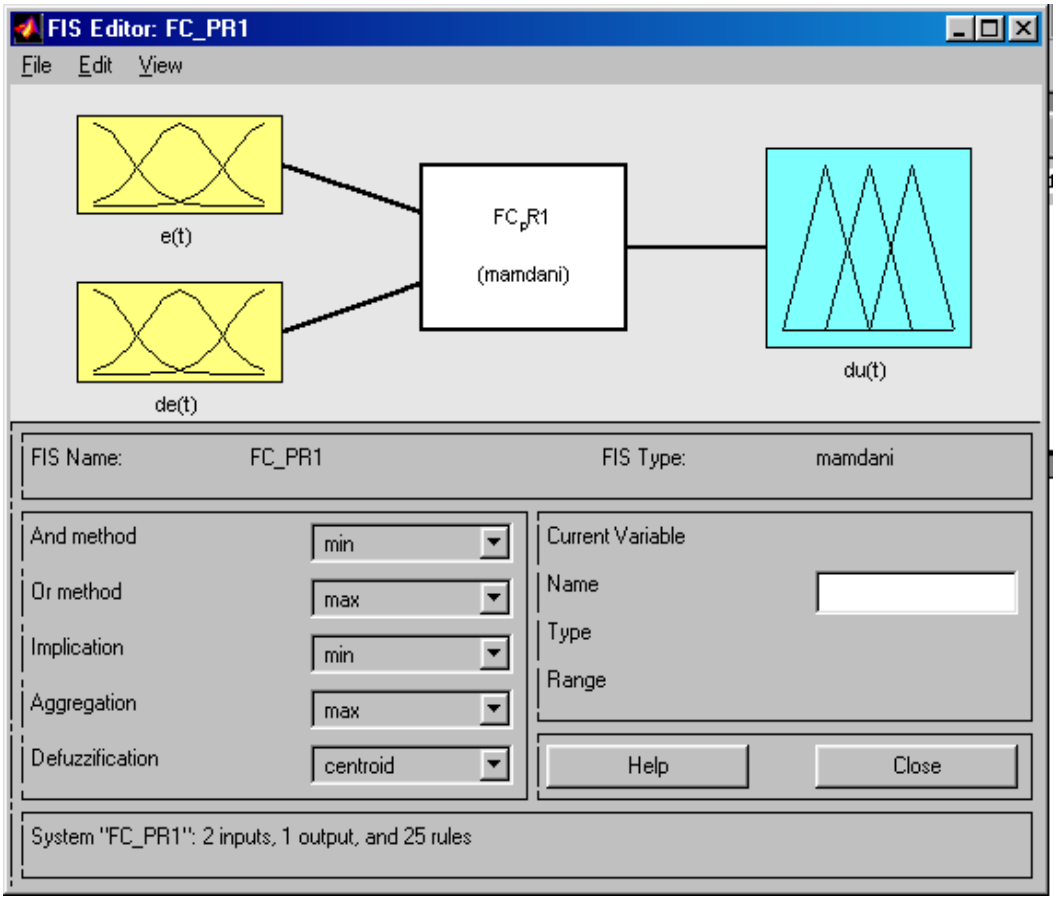


Bild 5.4 *FIS Edi-*

Das Fuzzy Inference System FIS hat folgende 3 Editoren: FIS Editor (Bild 5.4), Membership Function Editor (Bild 5.5) und Rule Editor (Bild 5.6). Außerdem besitzt es den Rule und den Surface Viewer (Bilder 5.8, 5.8).

Das Fuzzy Inference System FIS wird bei einem neuen Entwurf durch die Anweisung `fuzzy` aktiviert. Eine Aktivierung von schon existierenden FIS Systemen wird durch die Anweisung `fuzzy Name.fis` bewirkt.

5.2.1 Der FIS Editor

Der FIS Editor wird in Bild 5.4 als ein zweigeteiltes Fenster dargestellt. In dem linken unteren Teil können mit Hilfe von Menüs die Parameter der `And method, Or method, Implication, Aggregation und Defuzzification` eingestellt werden. In dem rechten unteren Teil kann man den Namen der Veränderlichen, den Typ und den Messbereich (`Range`) eingeben.

Die Menüleiste im Bild 5.4 enthält File, Edit, View, die das Arbeiten mit Dateien, die Edition von Fuzzy Logic Controllern und die Darstellung von Inferenz und den Steuerungsraum ermöglichen.

Wahl der Inferenzmethode

Klicken Sie in der Menüleiste File an und wählen Sie aus der Auswahlliste: New Mamdani FIS oder New Sugeno FIS.

Festlegung der Zahl der Ein- und Ausgangsgrößen

Klicken Sie in der Menüleiste Edit an und wählen Sie aus der Auswahlliste: `Add input, Add output` oder `Remove variable`(die zu entfernenden Variable müssen bezeichnet werden).

Eingabe und Veränderung des Namens einer Größe

Wählen Sie die entsprechenden Eingangs- oder Ausgangsgröße aus und klicken Sie die dazugehörige Ikone an (die Ikone und die Beschriftung wird rot). Dann schreiben Sie in das Textfeld, welches mit Name gekennzeichnet ist, den Namen der Größe.

Beispiel 5.1.

Bestimmen Sie ein Fuzzy Inference System (FIS) eines Fuzzy Systems mit 2 Eingangsgrößen *e(t)*, *de(t)* und einer Ausgangsgröße *u(t)*. Speichern Sie das FIS unter den Namen FC_PR1.

Bearbeitungsschritte:

1) Aktivieren des FIS Systems durch die Anweisung `fuzzy`.

2) Klicken Sie `Edit` an und wählen Sie (anklicken) `Add Input.` Damit erhöht sich die Zahl der Eingänge auf zwei.

3) Klicken Sie input1 an und ersetzen Sie in dem Textfeld, das die Bezeichnung Name hat, input1 durch e(t). Dann folgt "Return"

4) Klicken Sie input2 an und ersetzen Sie in dem Textfeld, das die Bezeichnung Name hat, input2 durch de(t). Dann folgt "Return".
5) Klicken Sie output1 an und ersetzen Sie in dem Textfeld, das die Bezeichnung Name hat, output1 durch u(t). Dann folgt "Return".
6) Klicken Sie das Menü File an und wählen Sie aus der Auswahlliste Save to disk as.
7) Geben Sie den Namen FC_PR1 ein. Es folgt "Return".

Im graphischen Fenster des FIS Editors werden entsprechende Ikonen der Ein- und Ausgangsgrößen dargestellt. In der Mitte steht eine Ikone mit Inferenzmethode (mamdani) und mit dem Namen des FIS. Wenn die Verbindungslinien zwischen diesen Blöcke **gestrichelt sind**, dann sind die Blöcke nicht parametriert oder die Parametrierung ist nicht ordentlich beendet (siehe 5.2.2). ***In einem solchen Fall ist es nicht möglich den FIS Block zu verbinden oder zu starten***.

Das Menü View dient zur Aktivierung

a) des Membership Function Editors (Editor der Zugehörigkeitsfunktionen),
b) Rule Editors, (Regelmenge-Editor),
c) Rule Viewers (graphische Darstellung der Infernz)
d) Surface Viewers (graphische Darstellung der Stellgrößenfläche).

5.2.2 Der Membership Function Editor (MF Editor)

Aktivierung des MF Editors

1) Im FIS Editor (Bild 5.4) durch Doppelklick auf die Ikone „Input (e(t),de(t) oder Output du(t) " Welche?
2) Klicken Sie `View` an und wählen Sie aus der Auswahlliste Edit eine Membership function . Das Fenster Membership Function Editor ist in Bild 5.5 dargestellt und ist ähnlich aufgebaut wie das Fenster des FIS Editors.

Es hat ein graphisches Fenster, in dem in der linken oberen Ecke Ikonen der Ein- und Ausgangsgrößen angebracht sind. Außerdem werden alle Zugehörigkeitsfunktionen der ausgewählten Ein-Ausgangsgrößen mit der Bezeichnung in diesem Fenster dargestellt.

In dem unteren Teil des Fensters Membership Function Editor sind die Fenster Current Variable, Current Membership Function (click on MF to select) und das Fenster Ready und die Tasten Help, Close angeordnet.

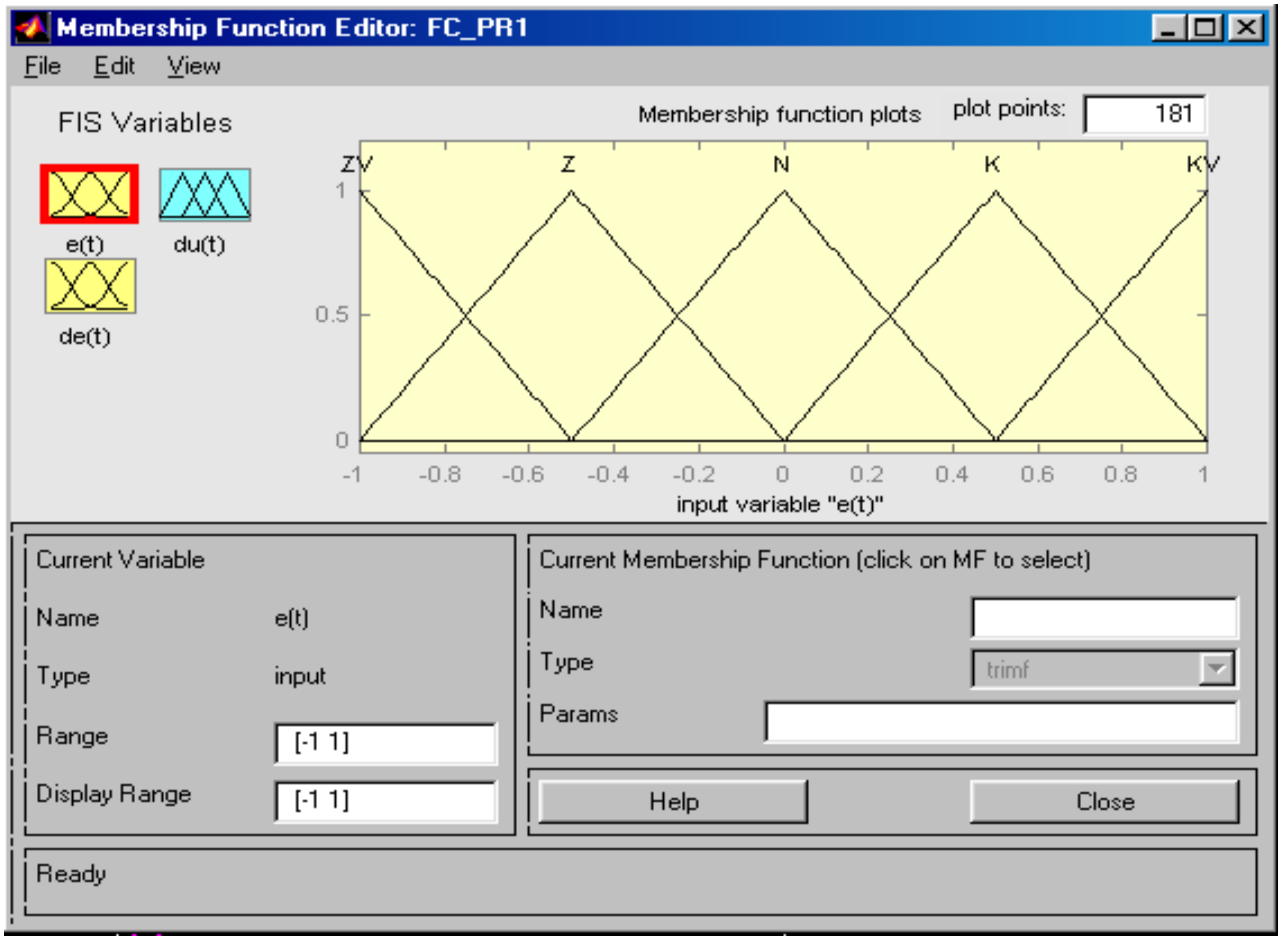


Bild 5.5 *Membership Function Editor*

Parametrieren der Zugehörigkeitsfunktion

1. Die Ikone der gewählten Eingangsgröße wird angeklickt. Die gewählte Ikone wird rot hinterlegt.

2. Sind die Zugehörigkeitsfunktionen schon festgelegt, so werden alle entsprechenden Zugehörigkeitsfunktionen mit den eingestellten Parametern und mit den Namen der Veränderlichen- als **Membership function plots** dargestellt.

3. Wenn die Zugehörigkeitsfunktionen noch nicht festgelegt sind, geben Sie in dem Fenster **Current Variable** in das Textfeld Range und Display Range den Größenbereich ein. Außerdem ist noch folgendes einzutragen:
   Name: Name der Größe
   Type: Output oder Input (abhängig von der gewählten Größe, ob das Ein- oder Ausganggröße ist)

4. Dann klicken Sie in der Menüleiste Edit an und aus der Menüliste klicken Sie Add membership function an. Das Fenster **Add membership function** ist im Bild 5.6 dargestellt.

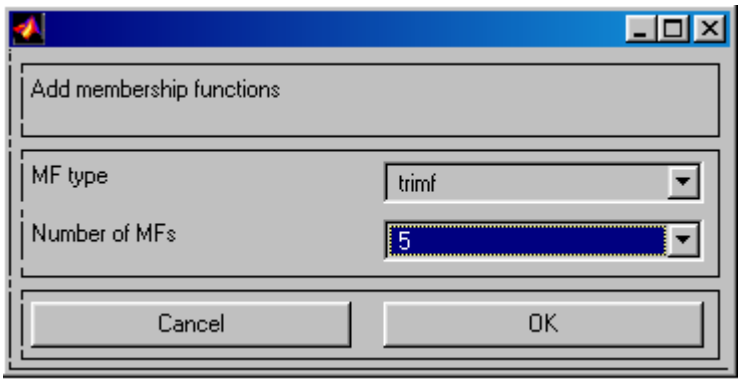


Bild 5.6 *Fenster "Add membership functions"*

5. Im Fenster **Add membership function** wird unter **MF type** mittels Rollbalken die Dreieck- Zugehörigkeitsfunktion „trimf" aus den vorgegebenen Zugehörigkeitsfunktion der Ein- oder Ausganggrößen {trimf, trapmf, gbellmf, gausmf, gaus2mf, pimf, dsigmf, psigmf } gewählt .

6. In dem Rollmenü **Number of FMs** wird Zahl der Zugehörigkeitsfunktionen eingegeben. Wie wird diese bestimmt ? Wir haben das gemacht in Kap.4.1gemacht!

7. ***Parametrisierung der Zugehörigkeitsfunktion:*** Klicken Sie an die ausgewählte Zugehörigkeitsfunktion, die danach rot wird. Dann werden in dem Fenster **Current Membership Function (click on MF to select)** der Type, der Namen und die Parameter in das Textfeld eingegeben.

5.2.3 Der Rule Editor (Regeleditor RE)

Die Bedienoberfläche des Regeleditors ist in Bild 5.7 dargestellt.

Aktivierung des RE Editors (2 Möglichkeiten):

1) Im FIS Editor durch Doppelklick auf die Ikone der Eingangs-Größe .
2) `View` anklicken und aus der Auswahlliste Edit rules auswählen.

Die Menüleiste enthält File, Edit, View und Options. Jede linguistische Ein- und Ausgangsgröße hat ihr Rollemenü, in dem die Bezeichnungen der entsprechenden Termen dargestellt sind. Außerdem enthält es ein großes Textfeld, in dem die Regeln dargestellt sind. In diesem Feld kann man per Hand oder mit Hilfe der folgenden Tasten editieren.

Delete rule : Die Regel wird gelöscht
Add rule : Eine Regel wird zugegeben
Change rule : Eine Regel wird verändert

Eine konkrete Regel kann mit Hilfe des Rollendatenmenüs für die linguistische Ein- und Ausgangsgröße (z.B. *e(t)*, *de(t)* und *u(t)*) einschließlich des gewünschten Gewichtes zu-

sammengestellt werden. Die Termen können mit den Operatoren and oder or verbunden werden. Die einzelnen Terme können auch in der Negation auftreten, was durch den Klick auf das Feld not. durchgeführt wird.

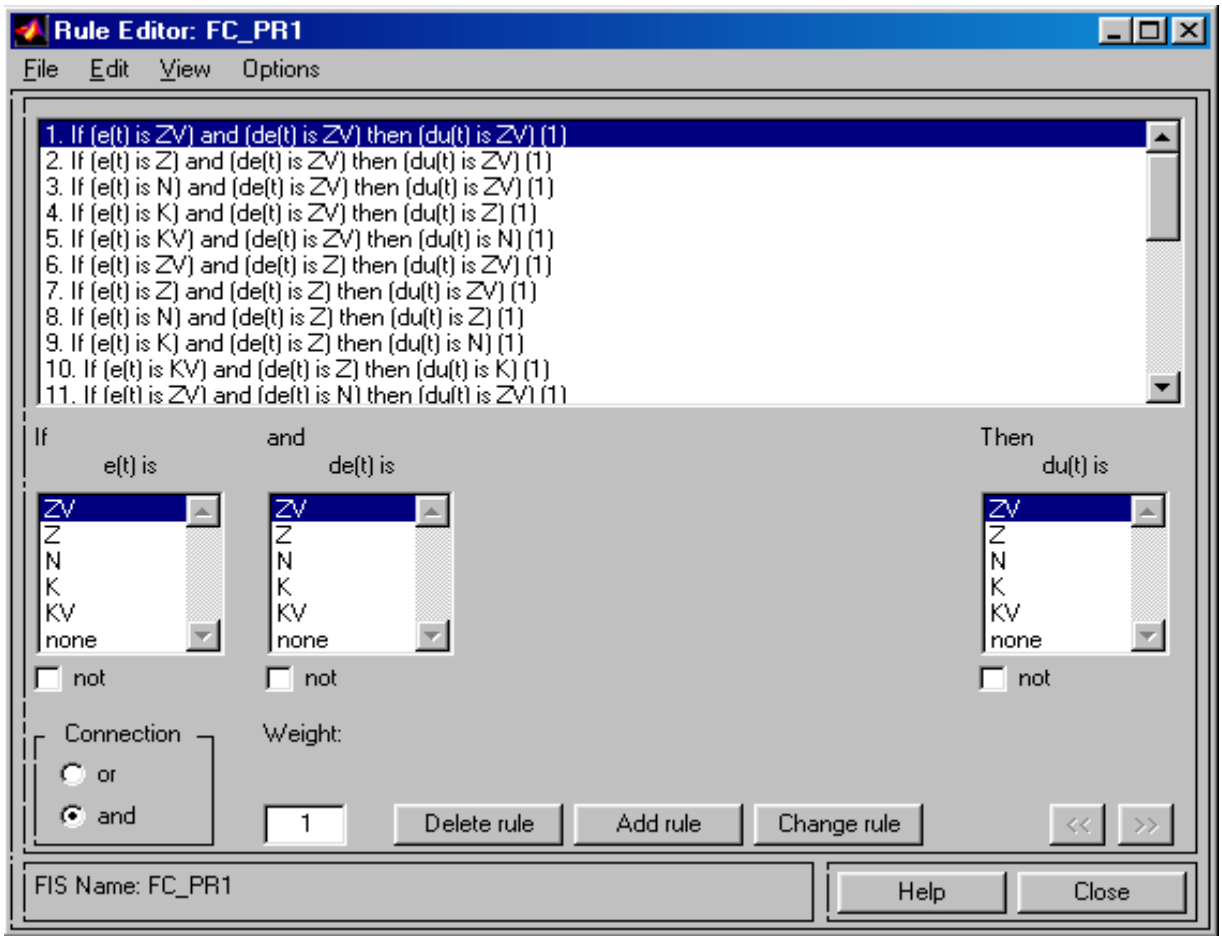


Bild 5.7. *Rule Editor*

5.3 RULE VIEWER, SURFACE VIEWER

Der Rule Viewer stellt alle Regeln und die Zugehörigkeitsfunktionen der Ein- und Ausgangsgrößen dar.

Aktivierung des Rule Viewers (Bild 5.8)

1) Klicken Sie das Leistenmenü View an.
2) Wählen Sie aus dem geöffneten Menü **Rule Viewer**

Der Surface Viewer stellt den Raum der Ausgangsgrößen in der Abhängigkeit von der Eingangsgrößen dar.

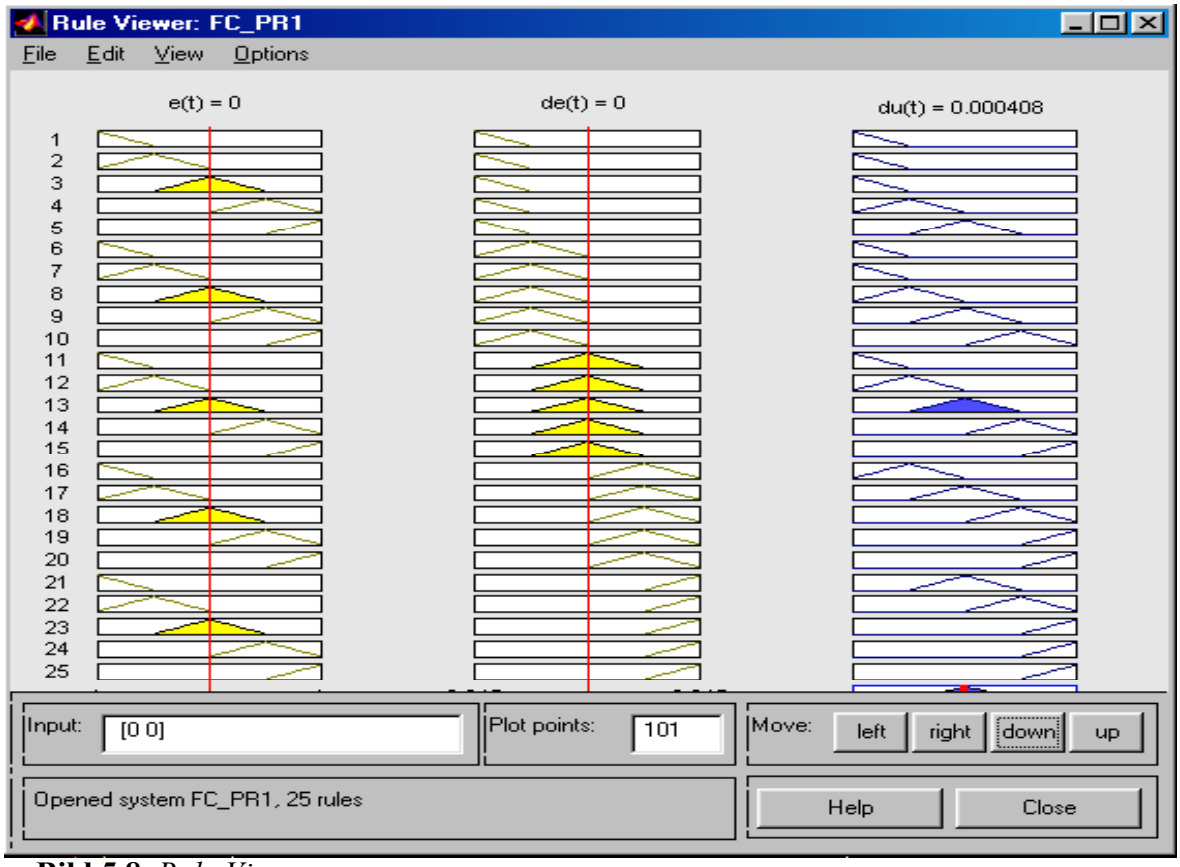


Bild 5.8 *Rule Viewer*

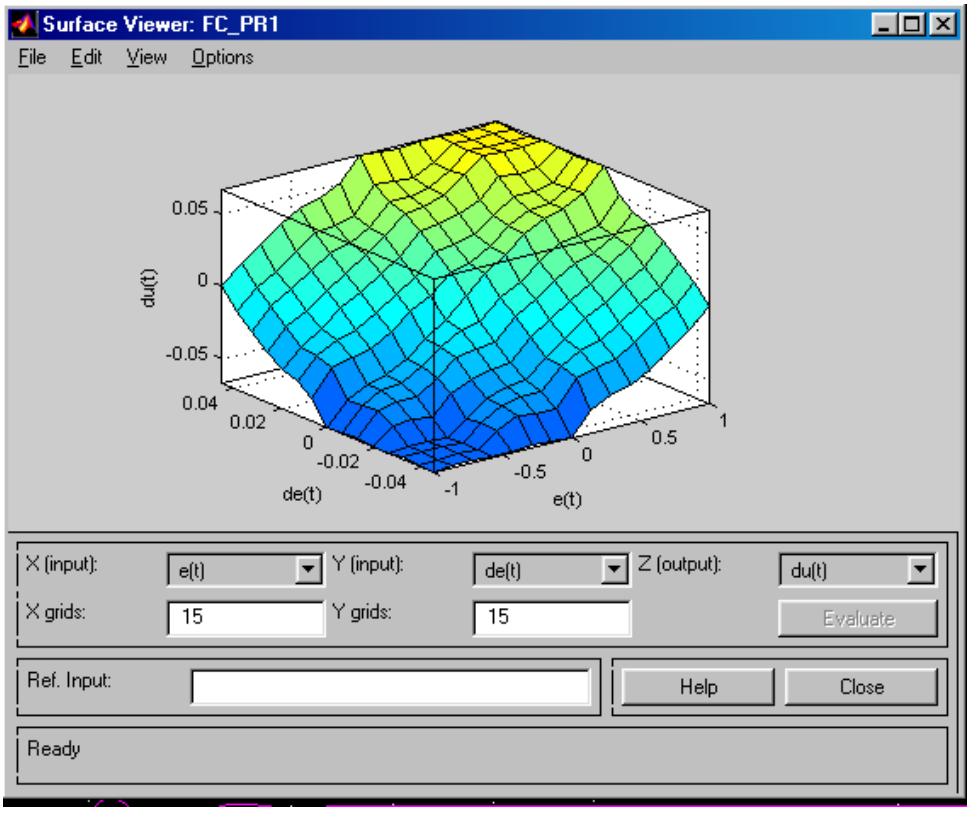


Bild 5.9: *Surface Viewer. Steuerraum*

Aktivierung des Surface Viewers (Bild 5.9)

1) Klick auf **View**
2) Klick auf **Surface Viewer**.

5.4 DIE FIS MATRIX

Damit eine Simulation im SIMULINK stattfindet, ist es nötig das Fuzzy Inference System unter seinem Namen im Arbeitsraum des MATLABs (**Workspace**) zu speichern.

Als Grundlage der Beschreibung [10, 12] und der Informationsspeicherung dient in MATLAB eine Matrix. Deshalb sind die Informationen des Fuzzy Inference Systems FIS als Matrix gespeichert. Diese Matrix wird als **FIS Matrix** bezeichnet.

Speichern der FIS Matrix ins Workspace

1) Klicken Sie das Leistenmenü File im Surface Viewer (Bild 5.9) an.
2) Aufruf des Fensters „Block Parameters: Fuzzy Logic Controller“(Bild 5.10). Klicken Sie Save to workspace an oder in SIMULINK durch einen Doppelklick auf die Ikone des Fuzzy Logic Controller.
3) Geben Sie den Name der FIS Matrix ein

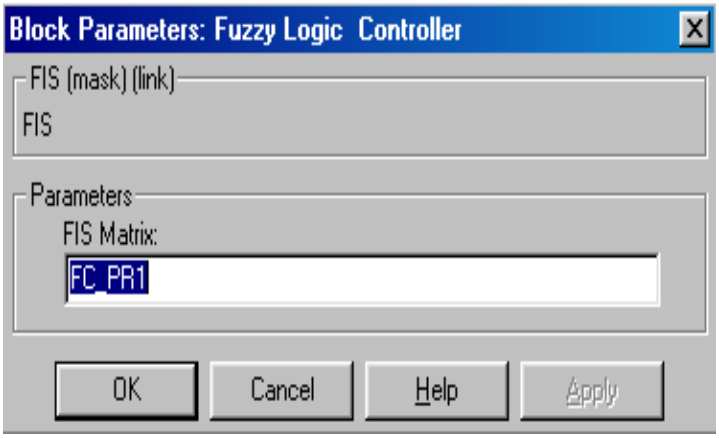


Bild 5.10. *Fenster „Block Parameters: Fuzzy Logic Controller“*

Speichern einer FIS Matrix von dem Disc ins Workspace kann man auch in MATLAB Programm mit Hilfe der Anweisung **readfis**. Als Beispiel wird die Anweisung von dem Programm Init.m in der Beilage 3 dienen.

```
ZFD = readfis('zfd.fis');
ZFPI = readfis('zfpi.fis');
```

6 LITERATUR

[1] *The Student Edition of MATLAB.* Version 4, User's Guide. The Math Works, Inc. 1995, Prentice Hall, Englewood Cliffs. ISBN 0-13-184979-4

[2] *SIMULINK Dynamic System Simulation for MATLAB.Using Simulink*, Version 2 The Math Works, Inc.1997

[3] GRACE, A.-LAUB, J.A-LITTLE, J.N.-THOMPSON, C.M.: *Control System Toolbox.* For Use with MATLAB. User's Guide. The Math Works, Inc.1995.

[4] Gulley, N., Jang,J.S.: *Fuzzy Logic Toolbox. For Use with MATLAB*. The Math Works, Inc.1995

[5] BODE, H.: *MATLAB in der Regelungstechnik. Analyse linearer Systeme.* B.G.Teubner Stuttgart,Leipzig, 1998.

[6] FRANKLIN, F. G., POWEL,J. D., WORKMAN, L. M.: *Digital Control of Dynamic Systems*. Second Edition, Addison Wesley, 1990.

[7] FÖLLINGER,O.: *Regelungstechnik*. Heidelberg, Hüthig, 1984, 4. Auflage.

[8] UNBEHAUEN,H.: *Regelungstechnik I.* Vieweg & Sohn mbH, Braunschweig/ Wiesbaden,1992.

[9] REINISCH, K.: *Kybernetische Grundlagen und Beschreibung kontinuierlicher Systeme*. VEB Verlag Technik, Berlin,1974

[10] NELDER, J. A., MEAD,R.: *A Simplex Method for Function Minimization*. Computer Journal, vol. 7, p.308-313.

[11] HOFFMANN, J.: *MATLAB und SIMULINK. Beispielorientierte Einführung in die Simulation dynamischer Systeme*. Addison – Wesley-Longman, Bon, 1998.