Regelstruktur symbolisch vorgegeben

Gert-Helge Geitner, Dresden

Privatdozent Dr.-Ing. habil. G.-H. Geitner ist Mitarbeiter am Lehrstuhl Elektrische Antriebe und Grundlagen der Elektroenergietechnik des Elektrotechnischen Instituts der Technischen Universität Dresden.
Hauptarbeitsfelder: Digitale Regelung elektrischer Antriebe, Reglersoftware, vernetzte Antriebe, Bewegungssteuerungen

Die Berechnung von Reglerparametern nach dem Digitalen Betragsoptimum (BOD) wurde bisher durch unterschiedliche MATLAB m-Files für Einzelregelkreise, Kaskadenstrukturen und Zustandsregelstrukturen unterstützt. Durch Einführung weniger Definitionen zur symbolischen Vorgabe von Streckenstruktur im Laplace-Bereich und Regelstruktur im Z-Bereich kann mit Hilfe eines universellen m-Files sowohl die Optimierung von Reglerparametern für Einzelregelkreise, als auch für Kaskaden- oder Zustandsregelstrukturen erfolgen. Im vorliegenden Beitrag werden die notwendigen Vereinbarungen angegeben, mögliche m-File Übergabeparameter vorgestellt und die Anwendung des neuen Werkzeugs anhand von mehreren Beispielen erläutert.

Till now several MATLAB m-files in regard to single-loop control, cascade control and state control structures were necessary to calculate controller parameters according to the Digital Amount Optimum (Digitales Betragsoptimum - BOD). If there are introduced a few definitions only for a symbolic input of the plant structure in Laplace domain and the control structure in Z domain it is possible to create an universal m-file which optimises controller parameters not only of single and cascade control but also of state control structures. In this paper are necessary definitions given, possible m-file input parameters presented and applications of the new tool discussed by means of different examples.

1. Einführung

Eine Berechnung von Parametern diskontinuierlicher Regler (Mikrorechner-Regler) für elektrische Antriebe nach dem Digitalen Betragsoptimum (BOD) [1] bietet u.a. die Möglichkeit des Verzichts auf quasikontinuierliche Näherungen bei einfacher Berücksichtigung von Totzeiten und Mittelwertmessungen [2]. Es entfallen Forderungen bezüglich der Relation zwischen Abtastzeit und Streckenzeitkonstante. Stellgliedmodelle unter Einschluß von Abtastern sind vorteilhaft einsetzbar.
Für typische Strecken- und Regelstrukturen wurden zur on-line Adaption geeignete einfache Berechnungsvorschriften in [1] angegeben. In vielen Fällen ist jedoch eine off-line Berechnung der Reglerparameter notwendig, z.B. um unterschiedliche Strukturen zu untersuchen oder Lerndaten für Künstliche Neuronale Netze (KNN) zu berechnen. Hierfür standen bisher die m-Files bod.m (PID-Typ-Regler für Einzelregelkreise), kaskade.m [alt: casc.m / caspa.m] (PID-Typ-Regler für Kaskadenstrukturen) sowie zust_au.m [alt: bran_au.m] (Beispiel für Zustandsregelstrukturen) unter MATLAB zur Verfügung [3, 4]. Der Nutzer mußte den Aufruf unterschiedlicher m-Files beherrschen. Beim Entwurf von m-Files für Zustandsregelstrukturen anhand des Beispiels waren Berechnungsalgorithmus und Anwenderstruktur verknüpft. Im vorliegenden Beitrag wird eine Lösung zur einheitlichen Behandlung von Einzelregelkreisen, Kaskadenregelungen und Zustandsregelstrukturen vorgestellt. Wesentliche Randbedingungen sind:

- Nutzung der durch MATLAB [5] gegebenen Definitionen
- Beschränkung notwendiger neuer Definitionen zur Beschreibung der Strecken- und Regelstruktur auf ein Minimum
- Einfacher m-File Aufbau mittels weniger Übergabeparameter, aber Möglichkeiten zur wahlweisen Beeinflussung des internen Ablaufs durch zusätzliche Parameter bei Bedarf
- Konsequente Trennung von Aufgabenstellung (Struktur) und Berechnung
- Berechnung von Ergebnisparameterfeldern ohne Mehraufwand

Typische Beispiele demonstrieren die Möglichkeiten des neuen Softwarewerkzeugs unter MATLAB.

2. M-File Anforderungen

Unabhängig von den Eigenschaften des durch das Digitale Betragsoptimum (BOD) resultierenden Gleichungssystems zur Berechnung der Reglerparameter wird die MATLAB-Funktion fsolve.m zur Lösung eingesetzt. Für den Nutzer soll das weitestgehend unbemerkt geschehen. Hieraus ergeben sich folgende Forderungen:

- Aufruf des neuen Werkzeugs mittels *eines* neuen m-Files, aber Bereitstellung des zu lösenden Gleichungssystems für fsolve.m durch ein weiteres, *nicht* vom Nutzer geschriebenes und aufgerufenes m-File.
- Symbolische Hinterlegung der Strecken- und Regelstruktur in ein vorgefertigtes „Formular“-m-File durch den Nutzer.
- Laufzeit und Empfindlichkeit der in fsolve.m eingebauten Algorithmen kann durch die Gradientenberechnung beeinflußt werden. Hierzu soll alternativ zur fsolve.m internen Gradientenschätzung ein weiteres, speziell auf das BOD Optimierungsgleichungssystem [1] ausgerichtetes, m-File zur Verfügung stehen. Nutzeraufwand: Setzen eines Steuerbits.
- Wahlweise Übergabe des im "Optimization Toolbox" Handbuch beschriebenen OPTIONS-Vektors. Bei Bedarf können damit unterschiedliche Algorithmen in fsolve.m eingesetzt werden (Bits 5/7).
- Wahlweise Abschaltung eines Zufallszahlengenerators und Vorgabe der für fsolve.m notwendigen Anfangswerte für die unbekannten Reglerparameter durch den Nutzer.

Um die neue Funktion möglichst anwenderfreundlich zu gestalten und eine universelle Anwendbarkeit zu gewährleisten, wurden zusätzlich folgende allgemeine Forderungen aufgestellt:

- Wahlweise Möglichkeit der Übergabe von Parametern an die symbolisch hinterlegte Strecken- und Regelstruktur
- Wahlweise Umschaltung auf Anzeige der Pole und Nullstellen der definierten Teilstrecken
- Überprüfung der Übergabeparameter sowie der Strecken- und Regelstruktur in sinnvollem Maß

Auf Grundlage dieser Anforderungen wurden die in Tabelle 1 angegebenen MATLAB m-Files aufgestellt. Zur Ausführung einer Berechnung müssen folglich mindestens 3 m-Files geladen sein (bod_es.m, bod_esg.m, strucxxx.m). Die Hilfeausschrift von bod_es.m ist in Bild 1 angegeben. Wird beim Aufruf von bod_es.m SV(2)=1 vorgegeben, so muß zusätzlich bod_esp.m vorhanden sein. Struc999.m in Bild 2 stellt ein Formular-File für den Nutzer dar. In struc000.m sind alle Definitionen erläutert. Der einfachste Aufruf einer Reglerparameterberechnung für die in struc010.m beschriebene Struktur erfolgt mittels:

```
REG=bod_es('struc010',T,PAR);
```

Vorher muß sichergestellt sein, daß der Vektor PAR die Zahlenwerte der Streckenparameter in der in struc010.m festgelegten Reihenfolge enthält und der Skalar T dem Zahlenwert der Abtastzeit entspricht.

3. Streckenstrukturdefinitionen (kontinuierlicher Teil)

Zuerst wird der Signalflußplan der gesamten Regelstruktur im Laplace-Bereich dargestellt. Danach werden die für die MATLAB-Funktion connect.m notwendigen Definitionen in einem Anwenderfile strucxxx.m vorgenommen. Die Streckenübertragungsfunktionen sind dabei jedoch symbolisch vorzugeben und zwei weitere Vektoren mit Angaben zur Transformation und zu Namenvereinbarungen beizufügen.

Damit ergeben sich folgende Schritte:

1, Durchgehende Numerierung der Streckenübertragungsfunktionen (im weiteren Block bezeichnet) mit Ausnahme von Mittelwertmessungen sowie Abtastern und Haltegliedern; *vollständige* Definition der entsprechenden Zähler und Nenner als String mit fortlaufender Numerierung **sz1** bis **szx** bzw. **sn1** bis **snx**;

2, Aufstellung einer Blockverbindungsmatrix **Q** entsprechend connect.m
[Zeilenanzahl = Blockanzahl; 1. Spalte = fortlaufende Nummer der Blöcke; weitere Spalten = vorzeichenbehaftete Nummern der jeweiligen Vorgängerblöcke des betrachteten Blocks]

3, Definition von: **IN** = Nummer des Eingangsblocks der gesamten Strecke;
OU = Vektor mit Nummern der interessierenden Ausgangsblöcke in beliebiger Reihenfolge

4, Definition einer Matrix **MT** mit Angaben zur Z-Transformation der Teilstrecken mit den Ausgängen nach Vektor OU (Zeilenanzahl = Ausgangsanzahl entsprechend OU in gleicher Reihenfolge; 1. Spalte = *echte* nicht ganzzahlige Zeitverschiebung; 2. Spalte = Kode für Kopplung und Messung bezüglich dieser Teilstrecke:

0 = Abtaster (T) und Augenblickswertmessung (A))
2 = Halteglied (H) und Mittelwertmessung (M)
1 = T/M oder H/A

- siehe auch trans.m in [3])

5, Aufstellung eines String-Namenvektors **NV** der die Reihenfolge der in sz1 bis szx und in sn1 bis snx verwendeten Streckenparameter (Verstärkungen, Zeitkonstanten, etc.) bei der Übergabe von Zahlenwerten festlegt. Die Namen *müssen* durch Semikolon *und* Leerzeichen abgeschlossen werden. In NV brauchen nur die Streckenparameter symbolisch definiert werden, die von "außen", also über den Aufruf von bod_es.m, beeinflußbar sein sollen.

Ganzzahlige Totzeiten innerhalb des Streckenmodells können an die Schnittstellen zwischen kontinuierlichen und diskontinuierlichen Blöcken geschoben werden. Damit ist für die Struktureingabe eine Zuordnung zu den Reglerblöcken möglich. Im Bedarfsfall sind Umschaltungen im Streckenteil durch die wahlfreien Übergabeparameter P1 und P2 von bod_es.m möglich. Durch die Definition der Matrix MT können auf einfache Weise sowohl Stellgliedmodelle auf Abtasterbasis als auch Mittelwertmessungen in inneren Regelkreisen berücksichtigt werden.

4. Regelstrukturdefinitionen (diskontinuierlicher Teil)

Die Vorgehensweise ähnelt der unter Punkt 3 Beschriebenen, jedoch sind einige Besonderheiten und Erweiterungen notwendig. Die fortlaufende Numerierung der diskontinuierlichen Übertragungsfunktionen (im weiteren Block bezeichnet) erfolgt dabei so, daß entsprechend der Anzahl der für den kontinuierlichen Teil angegebenen Ausgänge im Vektor OU die ersten Blocknummern der diskontinuierlichen Struktur für die diskretisierten Streckenübertragungsfunktionen reserviert sind. Folgende Schritte sind notwendig:

1, ***Vollständige*** Definition der Blöcke durch Zähler *und* Nenner als String mit fortlaufender Numerierung **rzy** bis **rzz** bzw. **rny** bis **rnz**. Es gilt also y=length(OU)+1.

Hinweis: Um eine spätere Umnumerierung zu vermeiden, sollten von Anfang an alle eventuell für die Optimierung interessierenden Ausgänge der kontinuierlichen Streckenstruktur angegeben werden. Es müssen nicht notwendig immer alle verwendet werden.

Beispiel: Sind für die kontinuierliche Strecke 5 Ausgangsblocknummern in Vektor OU enthalten und der diskontinuierliche Teil enthält 4 Blöcke, so gilt für die Numerierung Letzterer: rz6 bis rz9 und rn6 bis rn9.

Achtung:

Das Polynomgleichsetzen bei Führungsgrößenfiltern (Reglerzählerpolynom = Filternennerpolynom) sowie das stationäre Verhalten des Filters muß unmittelbar bei Definition der entsprechenden Übertragungsfunktionen beachtet werden!

2, Aufstellung einer Blockverbindungsmatrix **QQ** für die diskontinuierliche Struktur analog zu Punkt 3/Schritt 2. Dabei stehen die ersten "length(OU)" Blocknummern, wie oben erläutert, für die mit IN/OU beschriebenen und transformierten Ein-/Ausgangsmodelle der Gesamtstrecke. Daraus folgt, daß die ersten "length(OU)" Blöcke stets die gleiche Eingangsblockbelegung in der QQ Matrix haben müssen!

3, Definition von Matrizen **BME** und **BMA** mit den Nummern des jeweiligen Eingangs- (BME) bzw. Ausgangsblockes (BMA) für die zu optimierende Teilstruktur und Berechnungsvariante.

Definition des Spaltenaufbaus:

a, Jede Spalte entspricht einer Berechnungsvariante. Die Anzahl der von Null verschiedenen Elemente einer Spalte entspricht der Anzahl der Berechnungsläufe für die jeweilige Variante.

b, Der Inhalt eines Elementes bezeichnet einen Eingangs- (BME) bzw. Ausgangsblock (BMA)

Definition des Zeilenaufbaus:

a, Die Berechnung (Optimierung) erfolgt beginnend mit Zeile 1 einer ausgewählten Spalte. BME liefert den Eingangsblock, BMA den Ausgangsblock der betrachteten Regel-(Teil-)Struktur.

b, Beispiel 1:

BME=[3]; BMA=[1]; *Eine* Berechnungs-(Optimierungs-)variante mit einem Durchlauf. Eingangsblock für die Optimierung ist Nr. 3, Ausgangsblock ist Nr. 1

c, Beispiel 2:

```
BME=[1  1  7   BMA=[2  1  3   Es werden drei Optimierungsvarianten untersucht.
     7  1  0        4  2  0   Die zweite erfordert 3, die erste 2 und die dritte
     0  7  0];      0  3  0]; Variante 1 Optimierungslauf.
```

Ablauf von Variante eins: Erste Berechnung mit Block 1 als Eingang, Block 2 als Ausgang; zweite Berechnung mit Block 7 als Eingang, Block 4 als Ausgang.

4, Auswahl einer Berechnungsvariante aus den BME/BMA-Spalten durch Skalar **BRF**.

Hinweis: Eine Umschaltung von "außen" kann mittels der wahlfreien Übergabeparameter P1 und P2 von bod_es.m erfolgen.

5, Definition einer Matrix **BM** vom Aufbau der Matrizen BME/BMA. Jedoch muß mindestens das letzte Element einer Spalte Null sein. Damit besitzt die Matrix BM stets eine Zeile mehr als die Matrizen BME/BMA. Der Inhalt der Elemente von BM gibt die Anzahl der zu berechnenden Unbekannten pro Berechnungslauf an.

6, Aufstellung eines String-Namenvektors **NV** der die Reihenfolge der in rzy bis rzz und in rny bis rnz verwendeten Reglerparameter bei der Zuordnung von Zahlenwerten festlegt. Zusammen mit der Matrix BM liegt damit auch die Berechnungsreihenfolge fest. Die Namen *müssen* durch Semikolon *und* Leerzeichen abgeschlossen werden.

Auch für den diskontinuierlichen Teil gilt, daß die wahlfreien Übergabeparameter P1 und P2 von bod_es.m im Bedarfsfall Umschaltungen ermöglichen. Beispiele sind die wahlweise Verwendung eines Führungsgrößenfilters (Optimierung für verzögerte oder unverzögerte Eingangsgrößen) oder der Parametervergleich bei unterschiedlichen Reglerstrukturen (PI/PID).

5. Beispiele

Die nachfolgenden Beispiele sollen die einfache Handhabung der wenigen Definitionen demonstrieren. Die Beschreibung der Strukturen für die Beispiele 3 bis 5 erfolgt durch entsprechende SIMULINK [6] Simulationsmodelle.

Beispiel 1

Einfacher Einzelregelkreis mit einem Verzögerungsglied, Mittelwertmessung und PI-Regler. In einem File struc010.m werden folgende Definitionen eingetragen:

```
%*** Beginn  Streckendefinition
sz1='VS;';      sn1='[T1 1];';
Q=[1 0]; IN=1; OU=[1];
MT= [0 2];
NV=['VS; T1; '];
%****** Ende
```

```
%*** Beginn Reglerdefinition
rz2='[VR VRd1];';     rn2='[1 -1];';
QQ=[1   2
      2  -1];
BME=[2];
BMA=[1];
BM  =[2
        0];   BRF=1;
NV=['VR; VRd1; '];
%****** Ende
```

Der Aufruf

```
reg=bod_es('struc010',1,[0.2  5]);
```

liefert dann

```
reg=[14.1591   -11.5962]
```

Das heißt bei Mittelwertmessung und einer Abtastzeit von 1 ms wird für ein Verzögerungsglied $\frac{VS}{1+p\cdot T1}=\frac{0.2}{1+p\cdot 5ms}$ ein PI-Regler $\frac{VR+VRd1\cdot z^{-1}}{(1-z^{-1})}=\frac{14.16(1-0.8190z^{-1})}{(1-1z^{-1})}$ berechnet - vgl. Beispiel 1/Abschnitt 3 in [1].

Interessiert z.B. der Einfluß unterschiedlicher Abtastzeiten auf die Reglerparameter, so erhält man über

```
T=[0.8   1    1.2];     Par=[0.2   5; 0.2   5; 0.2   5];  und
reg=bod_es('struc010',T,Par);
```

sofort ohne Zusatzaufwand das Ergebnis

```
reg=[17.24   -14.70; 14.16   -11.60; 12.12   -9.54]
```

Beispiel 2

Einfacher Einzelregelkreis mit 3 Verzögerungsgliedern und wahlweise PI- oder PID-Regler sowie Optimierung für unverzögerte oder verzögerte Eingangssignale. Da keine Variation für die Streckenstruktur vorgesehen ist, genügt die Definition einer Gesamtübertragungsfunktion - vgl. Bild 3. Die Streckenverstärkung wird fest vorgegeben, die drei Zeitkonstanten sind durch den Übergabeparameter Par von bod_es.m vorgebbar und das Streckennennerpolynom wird durch die Nennerzeichenkette sn1 in Bild 3 ermittelt. Mit Hilfe des wahlfreien Übergabeparameters P2 sind Regelstruktur und Optimierungsvariante festzulegen.

Eine Vorgabe T=1; Par=[10 8 2]; P2=[1 0];

führt über reg=bod_es('struc041',T,Par,[],[],[1],[],P2);

zu reg=[1.1936 -0.9362]

Folglich wird durch Anwendung von BOD bei einer Abtastzeit von 1 ms für die gegebene Strecke ein störoptimaler PI-Regler $\frac{1{,}19 - 1{,}12z^{-1}}{1 - z^{-1}}$ mit Vorfilter $\frac{0{,}06}{1 - 0{,}94z^{-1}}$ berechnet.

Durch Setzen von OPT(1)=1 im Beispielaufruf kann die Arbeit des Suchalgorithmus beobachtet werden.

Beispiel 3

Die Definition der in Bild 4e dargestellten 3-schleifigen Kaskadenstruktur ist als Auszug von struc000.m in Tabelle 2, Spalte a enthalten. Dabei werden innere und mittlere Regelgröße als Mittelwert (MWM) gemessen und der mittlere Regler für verzögerte Eingangsgrößen mit Vorfilter ausgelegt (störoptimiert). Das Sprungantwortverhalten der Regelkreise veranschaulichen die Bilder 4a bis 4d. Aus Bild 4e ablesbare Reglerkoeffizienten sind für T=1ms gültig.

Beispiel 4

Eine 2-schleifige Kaskadenstruktur nach Bild 5b ist in struc001.m ohne Polkompensation und in struc002.m mit Polkompensation - durch den äußeren Regler - programmiert. Die Strecken- und Regelstrukturdefinitionen aus struc002.m sind in Tabelle 2, Spalte b angegeben.

Mit Eingabe von

```
reg=bod_es('struc002',2,[1  3.75  1  2.5  5],[0  1]);
```

erfolgt zunächst die Bestimmung sämtlicher Pole. Danach werden die beiden nur im äußeren Regelkreis enthaltenen Pole in P1 geladen und durch

```
reg=bod_es('struc002',2,[1  3.75  1  2.5  5],[],[],[],P1);
```

die in Bild 5b ersichtlichen Reglerkoeffizienten ausgerechnet - vgl. Beispiel/Abschnitt 5 [1]. Bei Anwendung der Polkompensation ist dabei allgemein zu beachten, daß die Zahlenwerte der Pole im jeweiligen strucxxx.m File in Zeichenketten umgewandelt werden. Ein Vergleich des Verhaltens mit Polkompensation, ohne Polkompensation sowie der ungeregelten Strecke erfolgt in Bild 5a. Die Regelkreise weisen ein typisches Überschwingen von 4,8% bzw. 7,2% auf.

Beispiel 5

Zur Programmierung der in [2] Bild 4 angegebenen Zustandsregelstruktur werden die hierfür wesentlichen Blöcke in Bild 6 numeriert. Sich hieraus ergebende Definitionen sind in Tabelle 2, Spalte c eingetragen. Mit Hilfe eines entsprechenden Files struc020.m können die in [2] Tabelle 1 aufgelisteten Rückführkoeffizienten nachgerechnet werden.

6. Zusammenfassung

Die erläuterte symbolische Definition von Strecken- und Regelstrukturen zum Einsatz des Digitalen Betragsoptimums (BOD) für die Reglerkoeffizientenberechnung von Einzelregelkreisen, Kaskadenregelungen und Zustandsregelstrukturen ermöglicht eine einheitliche Vorgehensweise. Durch MATLAB gegebene Vereinbarungen (z.B. connect.m) werden ausgenutzt, notwendige neue Definitionen sind auf ein Minimum beschränkt. Ganzzahlige und nichtganzzahlige Totzeiten sowie Mittelwertmessungen sind ohne Aufwand leicht zu berücksichtigen. Die Methodik läßt prinzipiell auch einen Austausch gegen andere Optimierungsverfahren ausgehend von transformierten Übertragungsfunktionen zu.
Verfügbare Beispiele und SIMULINK-Modelle sind in Tabelle 3 zusammengefaßt. Alle m-Files zur Berechnung, Definitionsbeispiele und Simulationsmodelle sind per Internet erhältlich [7, 8]. Aus den oben vorgestellten Beispielen lassen sich u.a. folgende Hinweise zusammenfassen:

- Reglerparameterfelder zu berechnen erfordert bei entsprechender Vorgabe der bod_es.m Übergabeparameter T und Par keinen Zusatzaufwand - vgl. Beispiel 1.
- Die Zeichenkettendefinitionen für Zähler und Nenner der Blöcke können selbst Berechnungsvorschriften (z.B. conv.m) enthalten - vgl. Beispiel 2.
- Polkompensation ist möglich. Entsprechende Zahlenwerte sind im strucxxx.m File in Zeichenketten umzuwandeln - vgl. Beispiel 4. Die Berechnung von Parameterfeldern kann dann mit Hilfe eines einfachen überlagerten Steuerfiles erfolgen.
- Nur die Parameter variierbar programmieren, die auch variabel verwendet werden. Das gilt sowohl für die Vorgabe von Streckenparametern mittels Par an bod_es.m, als auch für die Übergabe von Informationen mittels P1 und P2 an strucxxx.m oder für Regler mit Polkompensation.

- Prinzipiell kann eine Multiplikation von Unbekannten programmiert werden. Vorzugsvariante ist jedoch ein *Nichtausklammern* der Reglerverstärkung - vgl. Beispiel 2 mit P2(1)=0 bzw. P2(1)=1.
- Bei Totzeitvariation empfiehlt es sich schrittweise vorzugehen und die Ergebnisse vorheriger Berechnungen bei kleineren Totzeiten als Anfangswerte bei vergrößerter Totzeit zu verwenden - vgl. struc030.m.

Da der eingesetzte MATLAB Suchalgorithmus fsolve.m zum Lösen nichtlinearer Gleichungssysteme nur eine Lösung liefert, kann in bestimmten Fällen eine gezielte Beeinflussung über externe Anfangswertvorgabe notwendig werden. Eine Beobachtung der Suche ist bei Vorgabe von OPT(1)=1 möglich. SV(2)=1 führt in der Regel mit weniger Suchschritten zum Ziel, benötigt jedoch mehr Rechenzeit pro Schritt. Die von MATLAB über OPTIONS angebotene Algorithmenumschaltung ist über OPT von bod_es.m zugänglich.
Voraussetzung für den Aufruf von bod_es.m sind die MATLAB-Toolboxen Control System und Optimization sowie die m-Files trans.m und minreal1.m aus der Anwender-Toolbox BOD [4]. Für weiterführende Anregungen, Informationen zur Anwendung oder die Übergabe neuer strucxxx.m m-Files zur Erweiterung der Beispielsammlung ist der Autor jederzeit dankbar (Email: geitner@eti.et.tu-dresden.de).

Literatur:

[1] Geitner, G.-H.: Entwurf digitaler Regler für elektrische Antriebe. VDE-Verlag GmbH, Berlin und Offenbach 1996.
[2] Geitner, G.-H.: Berechnung von Reglern nach dem Digitalen Betragsoptimum. Automatisierungstechnische Praxis (atp) 38 (1996) 5, H.5, S.58/61
[3] http://www.eti.et.tu-dresden.de/ae/ae_1_0_2.htm
[4] ftp://www.eti.et.tu-dresden.de/pub/ae/MATLAB/BOD/TB_V20
[5] http://www.mathworks.de/products/matlab
[6] http://www.mathworks.de/products/simulink
[7] http://www.eti.et.tu-dresden.de/ae/ae_1_2.htm
[8] ftp://www.eti.et.tu-dresden.de/pub/ae/MATLAB/BOD/SYMBDEF

```
BOD_ES   Betragsoptimierung für in "struc..." definierte Regelstruktur
         [Reg]=bod_es(name,T,Par(,SV)(,AWV)(,OPT)(,P1,P2))
          name=Filename als string; File muß Definition der Strecken- und
               Regelstruktur enthalten (Formular:struc999.m/Def.:struc000.m)
          T=   Skalar mit Abtastzeit
               Spaltenvektor bei Parametervarianten
          Par= Parameterzeilenvektor mit Zahlenwerten der Streckenparameter
               (Reihenfolge und Bedeutung muß in File name definiert sein.)
               Matrix bei Parametervariation
          Reg= Reglerparameter-Zeilenvektor
              (Reihenfolge und Bedeutung muß in File name definiert sein.)
               Matrix bei Parametervariation;
               falls SV(2)=1: Streckennullstellen und Pole im Z-Bereich

      (,SV=Steuervektor für Arbeitsweise von bod_es - nur bei Bedarf!
        SV(1)=0 MATLAB-interne Gradientenschätzung (Standardeinstellung)
                Achtung: OPTIONS(9)=1 hierfür nicht zulässig !
              =1 Gradientenberechnung für strucxxx mittels bod_esp
                nur für Parameterprodukte mit Potenz "1" zulässig
         SV(2)=1 Keine Optimierung, sondern Berechnung der Nullstellen
                und Pole der transformierten Strecken (ohne zus. Totzeiten)
                Spalte_1=Nullst./Pol(0/inf); Spalte_2=jeweilige Anzahl
                Spalte_3=Werte ab hier)
       (,AWV=Anfangswertevorgabe von außen - nur bei Bedarf!)
      (,OPT=OPTIONS-Vektor siehe:Optimization Handbuch - nur bei Bedarf!)
       (,P1,P2=Parameterübergabe an STRUCxxx - nur bei Bedarf!)

Achtung: *  Voraussetzungen: Optimization Toolbox sowie bod_esg.m
            trans.m, minreal1.m, (bod_esp.m nur bei SV(1)=1)
         *  Bei Berechnung mehrerer Parametervarianten müssen length(T)
            und Zeilenanzahl von Par übereinstimmen
```

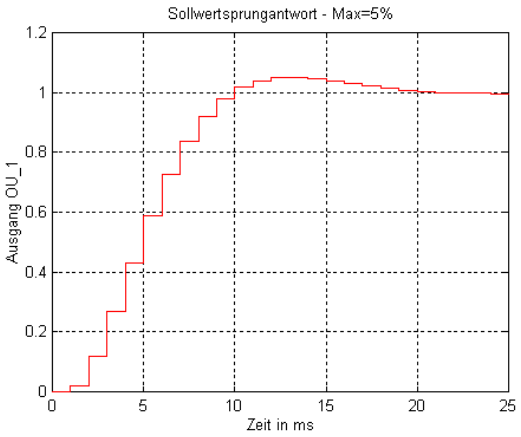
Bild 1 Hilfeausschrift von bod_es.m

```
function [ZVe,NVe,Q,IN,OU,MT,NV,BRF]=struc999(SOR,OU,P1,P2)
%STRUC999 Formularfile zur Strukturdefinition für bod_es.m
%                Zur Erläuterung der Definitionen siehe struc000.m
%ANWENDUNG:- Dieses File kopieren und dabei "999" im Namen sowie fileintern
%                      durch eine dreistellige Zahl ersetzen.
%                    - Zwischen den mit %****** eingerahmten Stellen die jeweilige
%                      Strecken- und Regelstruktur definieren.
%                    - Die Zeilen 2 bis 10 dieses Files streichen und durch Kurzbe-
%                      schreibung der betrachteten Struktur ersetzen.
%                    - Den neuen Filenamen bei Aufruf von bod_es verwenden
%HILFSFILE zur symbolischen Definition - kein selbständiger Aufruf !!!
%               SOR:  Schalter - zur Auswahl der Strecken oder Regelstruktur
%               OU:   Vektor mit fortlaufenden Nummern der Ausgänge der diskreti-
%                     sierten Streckenzustandsdarstellung
%              P1,P2: Parameterweitergabemöglichkeit über bod_es.m
% SCHLUESSELWORTE: Digitales Betragsoptimum, Symbolische Strecken-
%                        und Regel-Strukturdefinition für bod_es/bod_esg
%  Copyright (c) 1996 Dr. Gert-Helge Geitner; TU Dresden, Fak. ET,
%                        Elektrotechnisches Institut (ETI); Mommsenstr. 13;
%                        D-01062 Dresden, Germany;
%                        Tel./Fax ++49-351-463-2917/7280
%                        e-mail: geitner@eeiwzb.et.tu-dresden.de
%  Literatur zum Digitalen Betragsoptimum: Geitner, G.-H.:
%                        Entwurf digitaler Regler für elektrische Antriebe.
%                        VDE-Verlag GmbH Berlin und Offenbach 1996

if SOR==0  %Streckendefinition
% ****** Beginn ******
  sz1=             sn1=
   .          .
  szx=             snx=
  Q=
  IN=             OU=
  MT=
  NV=
% ****** Ende ******

s=0; zae=0; ZVe=[]; NVe=[]; BRF=0;
while s==0 zae=zae+1;  eval(['ZVe=[ZVe sz' num2str(zae) '];'],'s=1;');
                       eval(['NVe=[NVe sn' num2str(zae) '];'],'s=1;'); end
else  %Regelstrukturdefinition

% ****** Beginn ******
  rzy=             rny=
   .          .
  rzz=             rnz=
  QQ=
  BME=
  BMA=
  BM=
  BRF=
  NV=
% ****** Ende ******

s=0; zae=length(OU); ZVe=[]; NVe=[];
while s==0 zae=zae+1;  eval(['ZVe=[ZVe rz' num2str(zae) '];'],'s=1;');
                       eval(['NVe=[NVe rn' num2str(zae) '];'],'s=1;'); end
Q=QQ;IN=BME;OU=BMA;MT=BM;
end
```

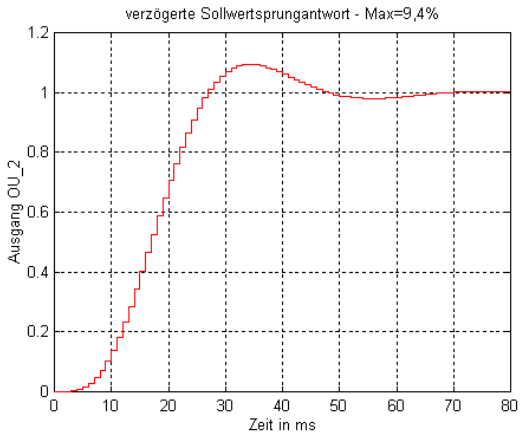
Bild 2 Formularfile struc999.m

```
% ****** Beginn Streckendefinition
   sz1='1;';  sn1='conv(conv([TN1 1],[TN2 1]),[TN3 1]);';
   Q=[1 0];   IN=[1];   OU=[1];
   MT=[0 1];
   NV=['TN1; TN2; TN3; '];
% ****** Ende
...
%******* Beginn Regelstrukturdefinition
rn2='[1 -1];';              % Definitionen für alle Varianten
BMA=[1]; BRF=1;
if P2(1)==0                 % Def. für unverzögertes Eingangssignal
  QQ=[1  2
      2 -1];
  BME=[2];
  if P2(2)==0               % Def. für unv. Eing. und PI-Regler
    rz2='[VR VRd1];';
    BM=[2
        0];
    NV=['VR; VRd1; '];
  else                      % Def. für unv. Eing. und PID-Regler
    rz2='[VR VRd1 VRd2];';
    BM=[3
        0];
    NV=['VR; VRd1; VRd2; '];
  end
else                        % Def. für verzögertes Eingangssignal
  QQ=[1  2 0
      2 -1 3
      3  0 0];
  BME=[3];
  if P2(2)==0               % Def. für ver. Eing. und PI-Regler
    rz2='VR*[1 d1];';
    rz3='(1+d1);';          rn3='[1 d1];';
    BM=[2
        0];
    NV=['VR; d1; '];
  else                      % Def. für ver. Eing. und PID-Regler
    rz2='VR*[1 d1 d2];';
    rz3='(1+d1+d2);';       rn3='[1 d1 d2];';
    BM=[3
        0];
    NV=['VR; d1; d2; '];
  end
end
%****** Ende
```

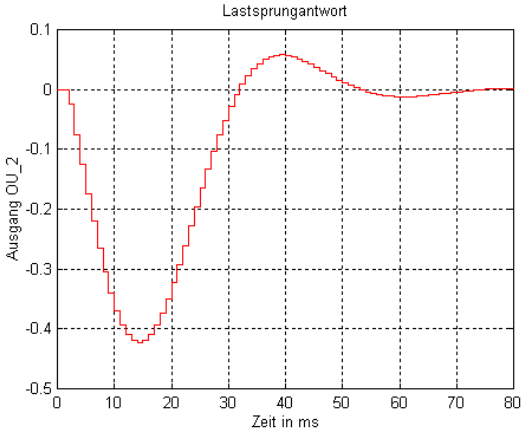
Bild 3 Beispiel für Berechnungsumschaltung in einem File - struc041.m mit Auswahl von Regler und Optimierung



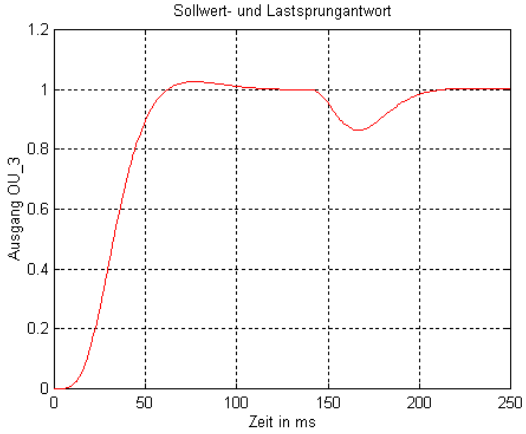
a, Sollwertsprungantwort des inneren Regelkreises



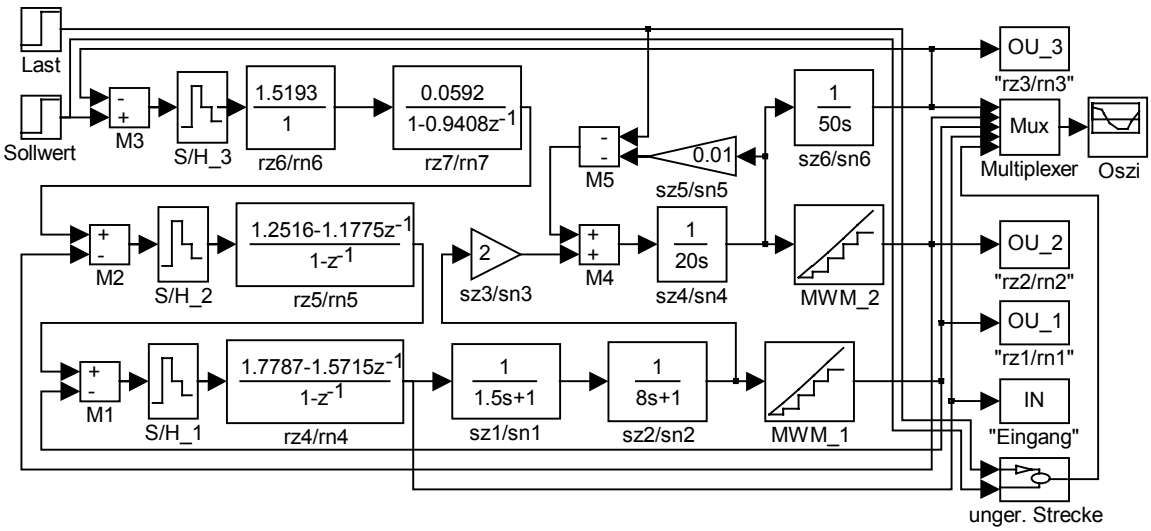
b, Sollwertsprungantwort des mittleren Regelkreises



c, Lastsprungantwort des mittleren Regelkreises

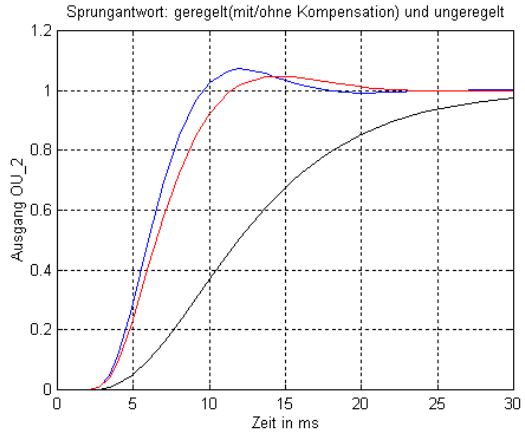


d, Sollwert- und Lastsprungantwort des äußeren Regelkreises

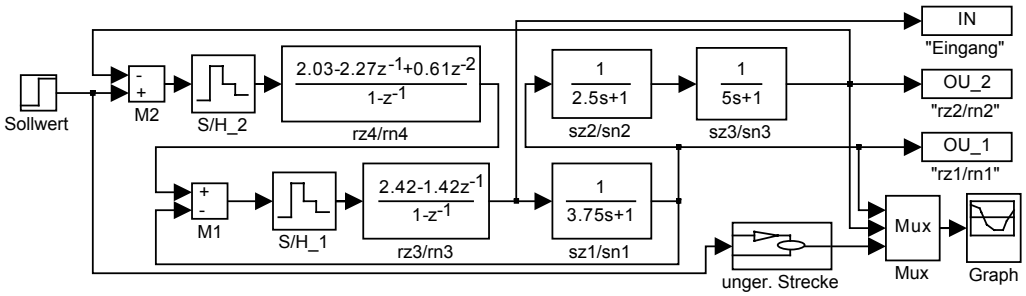


e, Simulationsmodell sist000.m

Bild 4 SIMULINK-Modell und Sprungantwortverhalten einer 3-schleifigen Kaskadenstruktur - struc000.m

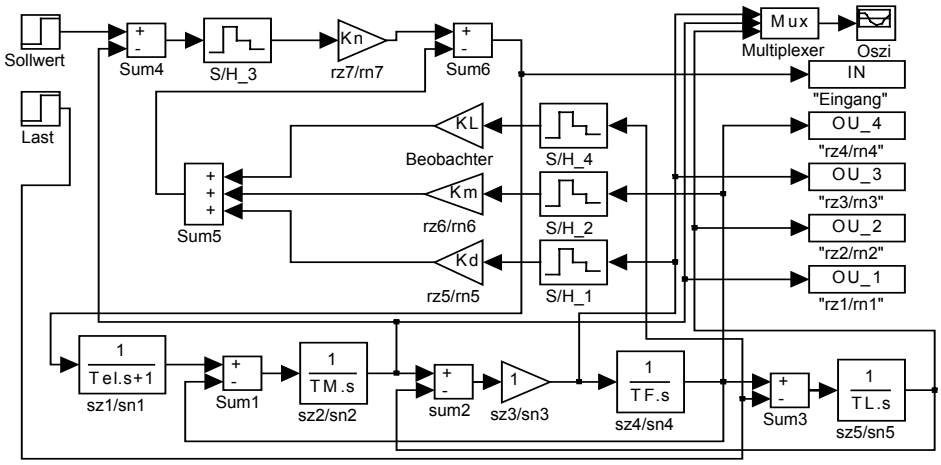


a, Sollwertsprungantwort mit und ohne Polkompensation im Vergleich zu ungeregelter Strecke



b, Simulationsmodell

Bild 5 SIMULINK-Modell und Sprungantwortverhalten einer 2-schleifigen Kaskadenstruktur – struc001.m / struc002.m



Simulationsmodell sist020.m

Bild 6 SIMULINK-Modell einer Zustandsregelstruktur - struc020.m [2]

Filename	Aufruf durch Nutzer	Eingriff durch Nutzer	Funktion
bod_es.m	x	-	Reglerparameterberechnung nach BOD
bod_esg.m	-	-	Bereitstellung eines Optimierungsgleichungssystems für fsolve.m
bod_esp.m	-	-	Gradientenberechnung - auf BOD Gleichungssystem zugeschnitten
struc999.m	-	-	Formular zum Kopieren und Eintragen einer Strecken- und Regelstruktur an zwei markierten Stellen - ohne Erläuterung der Definitionen
struc000.m	-	-	Vollständige Erläuterungen zur symbolischen Definition der Strecken- und Regelstruktur mit Beispiel 3-schleifige Kaskadenstruktur
struc010.m	-	x	Anwendungsbeispiel: Einzelregelkreis mit PI-Regler; (Erstellt durch Kopieren von struc999.m, Ändern der Strukturnummer von 999 auf z.B. 010 und Eintragen der Anwenderstruktur)

Tabelle 1 M-Files zur betragsoptimalen Reglereinstellung bei symbolischer Strecken- und Reglervorgabe

Strukturdefinition	Funktion	SIMULINK-Modell
struc000	K 1, 3-schleifig	sist000
struc001	K 2a, 2-schleifig	sist001
struc002	K 2b, wie K2a aber Polkompensation	
struc010	E 1a, PI-Regler, unverzögerte Eingänge	sist010
struc011	E 1b, PI-Regler, verzögerte Eingänge	
struc012	E 2a, PID-Regler, unverzögerte Eingänge	
struc013	E 2b, PID-Regler, verzögerte Eingänge	
struc020	Z 1, Berechnung von 3 Koeffizienten	sist020
struc021	Z 2, Berechnung von 4 Koeffizienten	sist021
struc030	E 3, mit Totzeit vorwärts	sist030
struc031	E 4a, mehrere Totzeiten, unverzögerte Eingänge	sist031
struc032	E 4b, mehrere Totzeiten, verzögerte Eingänge	
struc041	E 5, mit Berechnungsumschaltungen	sist041

Tabelle 3 In [7, 8] verfügbare Beispiele für Einzelregelkreise (E), Kaskadenstrukturen (K), Zustandsregelstrukturen (Z) und Modelle

```
% ****** Beginn  Streckendefinition
 sz1='V11;';                          sn1='[T11 1];';
 sz2='V12;';                          sn2='[T12 1];';
 sz3='V13;';                          sn3='1;';
 sz4='1;';                            sn4='[T21 0];';
 sz5='V21;';                          sn5='1;';
 sz6='1;';                            sn6='[T31 0];';
 Q=[1 0  0
      2 1  0
      3 2  0
      4 3 -5
      5 4  0
      6 4  0];          IN=[1];          OU=[2 4 6];
 MT=[0 2
       0 2
       0 1];
 NV=['V11; T11; V12; T12; V13; T21; V21; T31; '];
% ****** Ende
...
%******* Beginn Regelstrukturdefinition
 rz4='VR1*[1 d11];';            rn4='[1 -1];';
 rz5='VR2*[1 d12];';            rn5='[1 -1];';
%Vorgabe für Filter: VFi=1+d12 (Limes) und bf1=d12
   (Polynomgleichsetzen)
 rz6='VR3;';                    rn6='1;';
 rz7='(1+d12);';                rn7='[1 d12];';
QQ=[1  4 0
       2  4 0
       3  4 0
       4 -1 5
       5 -2 7
       6 -3 0
       7  6 0];
BME=[4 7 6]';   BMA=[1 2 3]';   BRF=1;   BM=[2 2 1
0]';
NV=['VR1; d11; VR2; d12; VR3; '];
%****** Ende
```

a, Kaskade 3-schleifig (struc000.m)

```
%****** Beginn Streckendefinition
sz1='VS1;';                  sn1='[T1 1];';
sz2='VS2;';                  sn2='conv([T2 1],[T3 1]);';
 Q=[1 0
      2 1];          IN=[1];    OU=[1 2];
 MT=[0 1
       0 1];         NV=['VS1; T1; VS2; T2; T3; '];
%****** Ende
...
%****** Beginn Regelstrukturdefinition
rz3='VR1*[1 d11];';                          rn3='[1 -
1];';
%Übergabe von P1 und P2 nur an STRUCxxx mög-
lich
   nicht an bod_esg:
   p1=num2str(P1(1)); p2=num2str(P1(2));
rz4=['VR2*conv([1 -' p1 '],[1 -' p2 ']);'];        rn4='[1 -
1];';
QQ=[1  3  0
       2  3  0
       3  4 -1
       4 -2  0];
BME=[3
        4];
BMA=[1
         2];
 BM=[2
        1
        0];          BRF=1;          NV=['VR1; d11; VR2; '];
%****** Ende
```

b, Kaskade 2-schleifig (struc002.m)

```
% ****** Beginn Streckendefinition
 sz1='1;';                            sn1='[Tel 1];';
 sz2='1;';                            sn2='[TM 0];';
 sz3='1;';                            sn3='1;';
 sz4='1;';                            sn4='[TF 0];';
 sz5='1;';                            sn5='[TL 0];';
 Q=[1 0  0
      2 1 -4
      3 2 -5
      4 3  0
      5 4  0];    IN=[1];   OU=[2 5 3 4];
 MT=[0 1
       0 1
       0 1
       0 1];     NV=['Tel; TM; TF; TL; '];
% ****** Ende
...
%******* Beginn Regelstrukturdefinition
 rz5='Kd;';             rn5='1;';
 rz6='Km;';             rn6='1;';
 rz7='Kn;';             rn7='1;';
QQ=[1 -5 -6 7
       2 -5 -6 7
       3 -5 -6 7
       4 -5 -6 7
       5  3  0 0
       6  4  0 0
       7 -1  0 0];
BME=[1 1
         7 7];
BMA=[4 4
         1 2];
 BM=[2 2
        1 1
        0 0];       BRF=2;      NV=['Kd; Km; Kn; '];
%****** Ende
```

c, Zustandsrückführung (struc020.m)

Tabelle 2 Beispiele zur symbolischen Definition mehrschleifiger Strukturen