HÖHERE LEHRANSTALT FÜR ELEKTROTECHNIK
INFORMATIONSTECHNIK, ENERGIETECHNIK & INDUSTRIELLE ELEKTRONIK
A-3101 St. Pölten, Waldstrasse 3, Tel: +43(0)2742-75051-300, Fax -230
Homepage: http://et.htlstp.ac.at E-Mail: et@htlstp.ac.at

COMA
Regelungstechnik mit Maxima

(COntrol engineering with MAxima)

Wilhelm Haager
HTL St. Pölten, Abteilung Elektrotechnik
wilhelm.haager@htlstp.ac.at

Version 1.6, 17.3.2011

Inhaltsverzeichnis

1 Einführung **1**
1.1 Begriffe . . . 1
1.2 wxMaxima Benutzeroberfläche . . . 2
1.3 Grundlegendes zum Paket COMA . . . 2

2 Plotroutinen **3**
2.1 Optionen des Gnuplot-Interfaces Draw . . . 3
2.2 Zusätzliche Optionen von COMA . . . 4
2.3 Plot . . . 4
2.4 Isolinien . . . 6

3 Übertragungsfunktionen **7**

4 Laplace Transformation, Sprungantwort **11**

5 Frequenzgänge **13**

6 Untersuchungen in der s-Ebene **17**
6.1 Pol/Nullstellen-Verteilung . . . 17
6.2 Wurzelortskurven . . . 18

7 Stabilitätsverhalten **20**
7.1 Stabilität . . . 20
7.2 Stabilitätsgüte . . . 22

8 Optimierung **23**

9 Reglerentwurf **24**

10 Zustandsraum **26**

11 Diverse Funktionen **29**

Literaturverzeichnis **31**

1 Einführung

1.1 Begriffe

Maxima: Open-Source Abkömmling des Computeralgebra-Systems *Macsyma*, das ursprünglich 1967–1982 im Auftrag des US-Energieministeriums am MIT entwickelt wurde. 1989 wurde eine Version von Macsyma mit dem Namen *Maxima* unter der *GNU General Public Licence* veröffentlicht und wird nun von einer unabhängigen Gruppe von Anwendern weiterentwickelt. Maxima ist in Lisp geschrieben und enthält selbst viele Elemente der funktionalen Programmierung.

Aufgrund seiner Leistungsfähigkeit und freien Verfügbarkeit gibt es eigentlich keinen Grund, es *nicht* zu verwenden.

wxMaxima: Eine von mehreren grafischen Benutzeroberflächen für Maxima. Es ermöglicht die Eingabe und das Editieren von Ausdrücken, sowie das Dokumentieren von Rechengängen mit Text und Bildern in einem Arbeitsfenster. Rechenergebnisse und (auf Wunsch) Grafiken werden von Maxima in dieses Arbeitsfenster ausgegeben.

Arbeitssitzungen können abgespeichert, geladen und wiederausgeführt werden; die wichtigsten Befehle können (für notorische Mausklickser) über Menüs und Befehlsschaltflächen aufgerufen werden. Arbeitssitzungen können als eine HTML-Filestruktur oder als LaTeX-File exportiert werden. Bei Export nach HTML wird jede Maxima-Ausgabe als einzelne GIF-Grafik abgespeichert. Beim Export nach LaTeX-Export ist mitunter etwas händisches Nacheditieren des resultierenden TEX-Files notwendig.

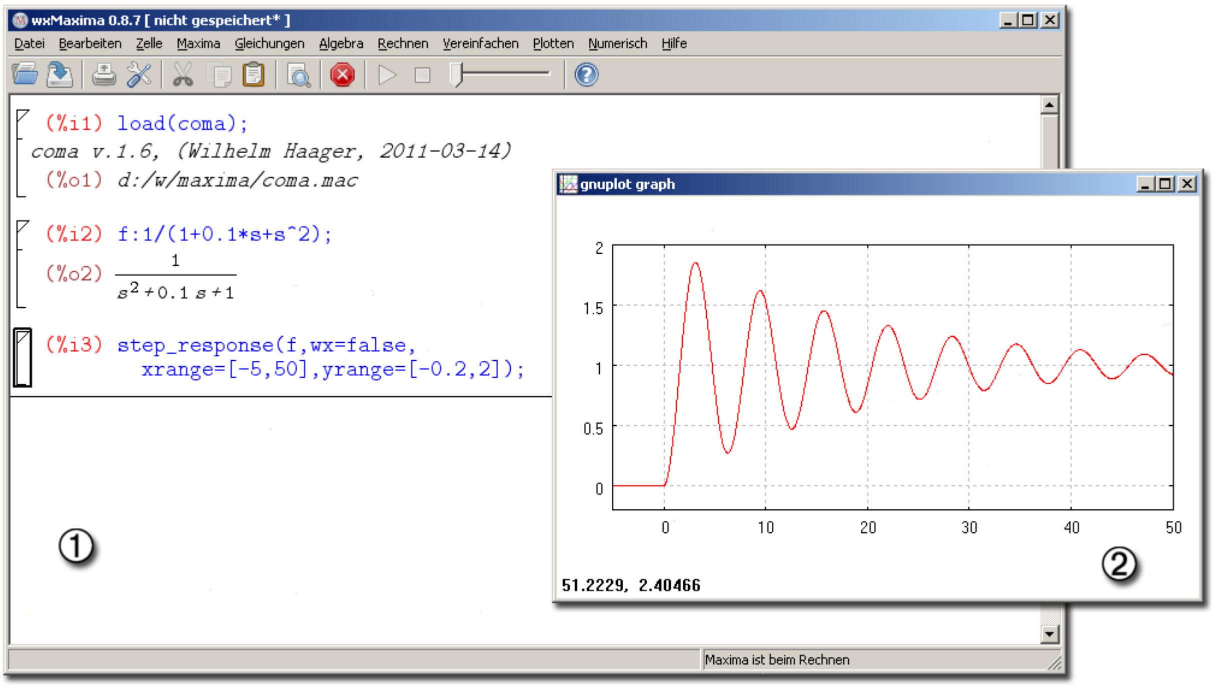
Bei der Installation von Maxima wird wxMaxima automatisch als Standard-Interface mitinstalliert.

COMA (*COntrol engineering with MAxima*):

Regelungstechnik-Paket für Maxima, enthält grundlegende Verfahren zur Sytemanalyse im Zeit-, Frequenz- und Laplacebereich, Reglerentwurfsverfahren, sowie Verfahren im Zustandsraum (in Entwicklung):

- Laplace-Rücktransformation von Übertragungsfunktionen beliebiger Ordnung (die in Maxima eingebaute Funktion versagt im Allgemeinen bei höherer als zweiter Ordnung)
- Sprungantworten
- Ortskurven und Bodediagramme
- Pol- und Nullstellen, Wurzelortskurven
- Stabilitätsuntersuchungen: Stabilitätsgrenze, Hurwitzkriterium, Stabilitätsbereiche in der Parameterebene, Phasenrand, Amplitudenrand
- Optimierung und Reglerentwurf: ISE-Kriterium, Betragsoptimum
- Zustandsraum: Umwandlung in eine Übertragungsfunktion, Normalformen, Steuerbarkeit, Beobachrbarkeit

1.2 wxMaxima Benutzeroberfläche



① . . . Arbeitsbereich zur Ein- und Ausgabe
② . . . Gnuplot-Ausgabefenster

1.3 Grundlegendes zum Paket COMA

- Die Laplacevariable ist immer s, die Zeitvariable immer t; bei Funktionen die den Frequenzgang betreffen, ist die Kreisfrequenz ω. Bei Frequenzgängen wird s gegebenenfalls automatisch durch $j\omega$ ersetzt. Übertragungsfunktionen sind gebrochen rationale Funktionen in s, Totzeiten werden nicht unterstützt, können aber mit Padénäherungen approximiert werden.
- Allen Funktionen, die eine Übertragungsfunktion als Parameter übernehmen, kann auch (ohne besondere Erwähnung) eine *Liste* von Übertragungsfunktionen als Parameter übergeben werden; der Rückgabewert ist dann ebenfalls eine Liste, deren einzelne Elemente den jeweiligen Übertragungsfunktionen entsprechen. Dies ist insbesonders bei Grafiken wichtig, wenn mehrere Kurven in ein einziges Diagramm gezeichnet werden sollen.

  Die zu plottende Liste kann nicht nur Funktionen (Übertragungsfunktionen) enthalten, sondern auch Grafikobekte des Gnuplot-Interfaces *Draw* (`explicit`, `parametric`, `implicit`, `polar`, `points`, `polygon`, `rectangle`, `ellipse`, `label`). Damit können beispielsweise Diagramme mit Beschriftungen und sonstigen Grafikelementen versehen werden, sowie berechnete Kurvenverläufe mit Messpunkten verglichen werden.
- Alle Plotroutinen können (zusätzlich zu den zu plottenden Funktionen) optionale Parameter der Form *option* = *value* enthalten, mit denen die Grafik in Bezug auf Strichstärken, Größe, Darstellungsbereich, Ausgabeziel, etc. konfiguriert werden kann.

2 Plotroutinen

Zu Darstellung von Grafiken verwendet Maxima das Programm *Gnuplot* [2], das bei der Erstellung der Grafik aufgerufen wird. Die Grafik wird dabei entweder in einem eigenen Gnuplot-Fenster dargestellt (bei Aufruf der Routinen `plot2d`, `plot3d`,...) oder direkt im Ausgabefenster von wxMaxima (bei Aufruf der Routinen `wxplot2d`, `wxplot3d`,...).

Zwei verschiedene Gnuplot-Interfaces stehen zur Verfügung, die Standardfunktionen von Maxima mit dem Wortstamm `plot` in den Funktionsnamen, sowie die Funktionen des Zusatzpaketes *Draw* mit dem Wortstamm `draw` in den Funktionsnamen.

Die Plotroutinen von *COMA* verwenden nicht die Standardfunktionen von Maxima (`plot2d`, `plot3d`, `wxplot2d`, `wxplot3d`), sondern die Funktionen des Zusatzpaketes *Draw* (`draw2d`, `draw3d`, `wxdraw2d` und `wxdraw3d`), siehe [1], Kapitel 48. Diese sind zwar etwas komplizierter in der Anwendung, bieten aber mehr Möglichkeiten, die Grafiken mit Hilfe von Optionen an die eigenen Wünsche anzupassen.

Alle Plotroutinen übernehmen als Parameter eine einzige Funktion oder eine *Liste* von Funktionen, zusätzlich optionale Parameter in der Form *option=value*.

2.1 Optionen des Gnuplot-Interfaces Draw

`terminal=`*target*	Ziel für die Grafikausgabe, mögliche Werte: `screen` (default), `jpg`, `png`, `eps`, `eps_color`
`file_name=`*string*	Name des Ausgabefiles, Default: `maxima_out.`*ext*
`color=`*c*	Linienfarbe
`line_width=`*w*	Strichstärke
`xrange=[`*x1*`,`*x2*`]`	Darstellungsbereich in x-Richtung
`yrange=[`*y1*`,`*y2*`]`	Darstellungsbereich in y-Richtung
`zrange=[`*z1*`,`*z2*`]`	Darstellungsbereich in z-Richtung
`logx=true/false`	logarithmische Skalierung der x-Achse
`logy=true/false`	logarithmische Skalierung der y-Achse
`logz=true/false`	logarithmische Skalierung der z-Achse
`grid=true/false`	Darstellung eines Kordinatengitters
`enhanced3d=true/false`	Einfärbung der Fläche bei 3D-Plots
`dimensions=[`*width*`,`*height*`]`	Breite und Höhe der Grafik

Wichtige Optionen des Gnuplot-Interfaces Draw

Eine vollständige Liste der Optionen findet sich im Maxima Manual [1]).

2.2 Zusätzliche Optionen von COMA

`wx=true/false`	legt die Art der Ausgabe fest: `true` ... Ausgabe in das wxMaxima-Arbeitsfenster `false` ... Ausgabe in ein Gnuplot-Fenster
`aspect_ratio=`*value*	Verhältnis Höhe/Breite des Diagrammes; der Wert -1 bewirkt gleiche Skalierung von x-Achse und y-Achse
`color=[`*c1*`,`*c2*`,...]`	Liste von Farbwerten, die in entsprechender Reihenfolge für die einzelnen zu plottenden Elemente angewandt werden
`line_width=[`*w1*`,`*w2*`,...]`	Liste von Strichstärken für die einzelnen Elemente
⋮	
Globale Variable:	
`plot_defaults`	Liste mit Optionen der Form *option=value*

Zusätzliche Optionen von COMA

Die Variable `plot_defaults` ist eine Liste mit Defaultwerten für Plot-Einstellungen in Form von Schlüssel-Wert Paaren. Im Gegensatz zur Liste `draw_defaults` des Gnuplot-Interfaces *Draw* kann `plot_defaults` auch andere Optionen enthalten , die *nicht* Bestandteil von *Draw* sind.

2.3 Plot

Die Funktion `plot` dient zur xy-Darstellung von Funktionen $f(x)$in einer Variablen oder zur dreidimensionalen Darstellung von Funktionen $f(x,y)$ in zwei Variablen.

`plot(`*f(x), opts*`)`	Plotten der Funktion $f(x)$ in einem zweidimensionalen Koordinatensystem
`plot(`*f(x,y), opts*`)`	Plotten der Funktion $f(x,y)$ in 3D-Darstellung
	Anstelle einer einzigen Funktion f kann auch eine *Liste* von Funktionen `[`f_1`,`f_2`,...]` geplottet werden.

Plotfunktion für 2D und 3D Grafiken

Intern werden die Funktionen des Paketes *Draw* (`wxdraw2d` oder `draw2d` bzw. `wxdraw3d` oder `draw3d`) mit entsprechenden Parametern aufgerufen. Der (bequeme) Aufruf von

```
plot([f(x),g(x)],xrange=[0,10],color=[red,blue])
```

entspricht genau dem (weniger bequem aufzurufenden) Befehl

```
wxdraw2d(xrange=[0,10],color=red,explicit(f(x),x,0,10),
          color=blue,explicit(g(x),x,0,10)).
```

Die Listenelemente der Optionen `color` und `linewidth` werden entsprechend der Syntax dieser Routinen vor die einzelnen Funktionen gesetzt, die anderen Optionen werden an den Anfang der Parameterliste gesetzt. Der Wert der Option `aspect_ratio`, die es in dieser Form in den Routinen des Paketes *Draw* nicht gibt, wird in der Option `user_preamble` in entsprechender Form an Gnuplot weitergereicht.

Defaultwerte für die Grafik-Optionen

```
(%i1) plot_defaults;
(%o1) [grid=true,dimensions=[440,270],wx=true,
aspect_ratio=0.6,color=[red,blue,green,goldenrod,
violet,gray50,dark-cyan,dark-orange,sea-green,
dark-pink]]
```

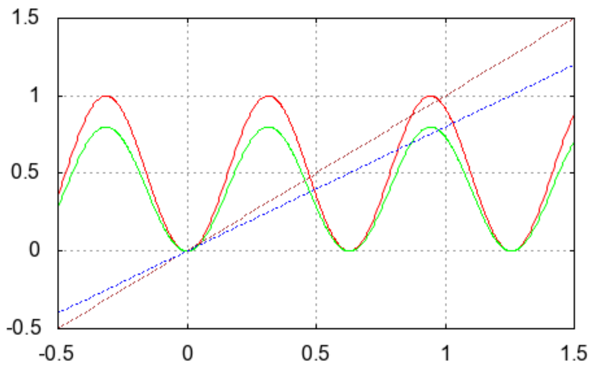
Liste mit Farbwerten

```
(%i2) col:[red,green,brown,blue];
(%o2) [red,green,brown,blue]
```

Plotten einer Liste von vier Funktionen in jeweils *einer einzigen* Variablen; intern wird die Funktion `wxdraw2d` aufgerufen. Die Variablennamen können auch unterschiedlich sein.

```
(%i3) plot([sin(5*x)**2,0.8*sin(5*y)**2,x,0.8*y],
      xrange=[-0.5,1.5],color=col,
      line_type=[solid,solid,dots,dots])$
```

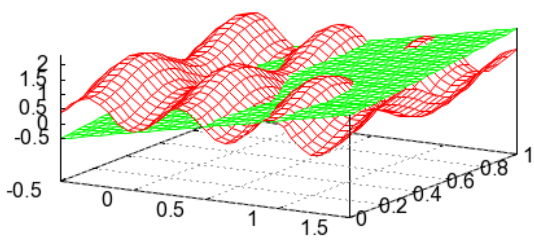
(%t3)



Plotten einer Liste von zwei Funktionen in jeweils *zwei* Variablen; intern wird die Funktion `wxdraw3d` aufgerufen. Die Option `surface_hide=true` unterdrückt verdeckte Linien.

```
(%i4) plot([sin(5*x)**2+0.8*sin(5*y)**2,x+0.8*y],
      xrange=[-0.5,1.5],surface_hide=true,color=col)$
```

(%t4)



`plot` evaluiert die zu plottende Funktion f *bevor* Kurvenpunkte berechnet werden. Soll die Funktion für jeden einzelnen Kurvenpunkt neu evaluiert werden, muss sie *quotiert* werden. Das ist z.B. für Kenngrößen von Übertragungsfunktionen notwendig (siehe Abschnitt 7), die nur numerisch berechnet werden können.

Übertragungsfunktion mit einer besonderen Abhängigkeit der Dämpfung vom Parameter a

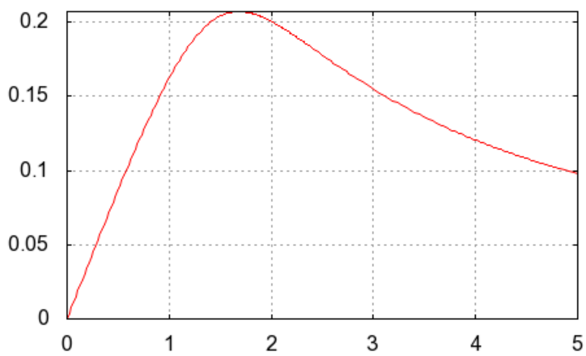
```
(%i5) f:(s+a)/(s^3+a*s^2+2*s+a);
```

(%o5) $$\frac{s+a}{s^3+a\,s^2+2s+a}$$

Die Dämpfung kann nur *numerisch* berechnet werden, was nur dann gelingt, wenn a einen fixen Wert hat. f muss daher quotiert werden, um eine frühzeitige Evaluation zu verhindern.

(%i6) `plot('damping_ratio(f),xrange=[0,5]);`

(%t6)



(%o6)

2.4 Isolinien

Die Funktion `contourplot` zeichnet Isolinien einer Funktion $f(x,y)$. Im Gegensatz zur Funktion `contour_plot`, die Bestandteil von Maxima ist, verwendet sie wie alle Plotroutinen des Paketes *COMA* das Gnuplot Interface *Draw* und hat auch die selben Optionen.

`contourplot(`*f(x,y)*,x,y,*opts*`)`	Plotten von Isolinien der Funktion $f(x,y)$
`contours=[`*z1*,*z2*,...`]`	Festlegen der Funktionswerte für die Isolinien

Isolinien

Übertragungsfunktion mit zwei freien Parametern a und b

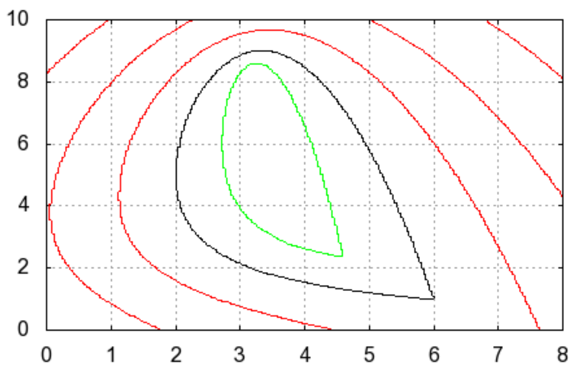
(%i7) `f:1/(s^5+s^4+6*s^3+a*s^2+b*s+1);`

(%o7) $$\frac{1}{s^5+s^4+6s^3+a s^2+b s+1}$$

Isolinien für die Dämpfung in Abhängigkeit der Parameter a und b, die schwarze Linie bei der Dämpfung 0 kennzeichnet die Stabilitätsgrenze, die grünen Linien kennzeichnen den stabilen Bereich, die roten den instabilen Bereich.

(%i8)
```
contourplot('damping(f),a,b,
xrange=[0,8], yrange=[0,10],
contours=[-0.3,-0.2,-0.1,0,0.1],
color=[red,red,red,black,green])$
```

(%t8)



3 Übertragungsfunktionen

Zur bequemen Erzeugung von Übertragungsfunktionen, in erster Linie zu Testzwecken und zum Experimentieren, stellt *COMA* folgende Funktionen bereit:

`rantranf(`n`)`	Zufalls-Übertragungsfunktion n-ter Ordnung, deren Zähler- und Nennerkoeffizienten Zufallswerte zwischen 1 und 10 sind
`stable_rantranf(`n`)`	Stabile Zufalls-Übertragungsfunktion (nur bis 6. Ordnung)
`gentranf(`c,k,d,n`)`	Übertragungsfunktion n-ter Ordnung, (Zählerpolynom k-ter Ordnung) mit den Zählerkoeffizienten c_i und den Nennerkoeffizienten d_i
`tranftype(`$F(s)$`)`	Typ der Übertragungsfunktion $F(s)$ als String
`ntranfp(`$F(s)$`)`	Liefert `true`, wenn alle Koeffizienten der Übertragungsfunktion $F(s)$ Zahlen (und keine symbolischen Ausdrücke) sind.
`closed_loop(`$Fo(s)$`)`	Übertragungsfunktion $F_W(s)$ des geschlossenen Regelkreises aus der offenen Regelschleife $F_O(s)$
`open_loop(`$Fw(s)$`)`	Übertragungsfunktion $F_O(s)$ der offenen Regelschleife aus dem geschlossenen Regelkreis $F_W(s)$
`time_delay(`$T,n,[k]$`)`	Padénäherung für ein Totzeitsystem der Ordnung n. Die Zählerordnung k ist optional.
`impedance_chain(`$Z1,Z2,\ldots[n]$`)`	Übertragungsfunktion einer Impedanzkette mit den Impedanzen $Z1$, $Z2$, … und einem (optionalen) Wiederholungsfaktor n
`transfer_function(`$eqs,vars,u,y$`)`	Berechnung der Übertragungsfunktion aus den Gleichungen *eqs* in den Variablen *vars* mit der Eingangsgröße u und der Ausgangsgröße y
`standard_form(`$F(s),n$`)`	Bringt die Übertragungsfunktion $F(s)$ in Abhängigkeit von n in eine der vier Standardformen

Erzeugung von Übertragungsfunktionen

Die Funktion `stable_rantranf(n)` sucht per Zufallsgenerator so lange Nennerkoffizienten zwischen 1 und 10, bis eine *stabile* Übertragungsfunktion gefunden wird, was mit steigender Systemordnung immer schwieriger wird, die Rechenzeit steigt bei siebter Ordnung extrem an. `stable_rantranf` funktioniert daher nur bis zur 6. Ordnung.

Höhere Ordnungen können durch Multiplikation mehrerer Übertragungsfunktionen erreicht werden, die Koeffizienten sind dann allerdings nicht mehr auf den Bereich 1...10 beschränkt.

Erzeugung einer Liste von Zufalls-Übertragungsfunktionen vierter Ordnung, die Zählerordnung ist mindestens um eins niedriger.

(%i1) `fli:makelist(rantranf(3),k,1,4);`

(%o1) $$[\frac{6s^2+10s+2}{5s^3+6s^2+5s+3}, \frac{7}{s^3+6s^2+6s+4}, \frac{10s^2+10s+7}{8s^3+5s^2+10s+1}, \frac{2s^2+s+4}{7s^3+7s^2+7s+10}]$$

Prüfung der Stabilität der Übertragungsfunktionen (siehe Abschnitt 7)

(%i2) `stablep(fli);`

(%o2) `[true,true,true,false]`

Erzeugung einer Liste von stabilen Zufalls-Übertragungsfunktionen

(%i3) `fli:makelist(stable_rantranf(3),k,1,4);`

(%o3) $$[\frac{6}{7s^3+9s^2+10s+10}, \frac{6s+7}{5s^3+9s^2+8s+4}, \frac{8}{5s^3+6s^2+7s+8}, \frac{7s^2+7s+3}{3s^3+9s^2+7s+2}]$$

Alle sind stabil.

(%i4) `stablep(fli);`

(%o4) `[true,true,true,true]`

Erzeugung einer Liste von Übertragungsfunktionen

(%i5) `fo:[k/s,5/(s*(s+3)),1-b/s];`

(%o5) $$[\frac{k}{s}, \frac{5}{s\,(s+3)}, 1-\frac{b}{s}]$$

Berechnung der Übertragungsfunktionen der geschlossenen Regelkreise

(%i6) `fw:closed_loop(fo);`

(%o6) $$[\frac{k}{s+k}, \frac{5}{s^2+3s+5}, \frac{s-b}{2s-b}]$$

Ermittlung des Typs der Übertragungsfunktionen als Textstrings

(%i7) `tranftype(fw);`

(%o7) `[PT1,PT2,PDT1]`

Überprüfung, ob alle Koeffizienten der Übertragungsfunktionen numerische Werte sind

(%i8) `ntranfp(fw);`

(%o8) `[false,true,false]`

Zurückrechnung auf die offenen Regelschleifen

(%i9) `open_loop(fw);`

(%o9) $$[\frac{k}{s}, \frac{5}{s^2+3s}, \frac{s-b}{s}]$$

`gentranf(a,k,b,n)` erzeugt eine allgemeine Übertragungsfunktion mit indizierten Koeffizienten der Form

$$\frac{a_0+a_1s+a_2s^2+\cdots+a_ks^k}{b_0+b_1s+b_2s^2+\cdots+b_ns^n}.$$

Übertragungsfunktion mit allgemeinen Koeffizienten a_i und b_i

(%i10) `gentranf(a,3,b,5);`

(%o10) $$\frac{a_3\,s^3+a_2\,s^2+a_1\,s+a_0}{b_5\,s^5+b_4\,s^4+b_3\,s^3+b_2\,s^2+b_1\,s+b_0}$$

Totzeitsysteme haben transzendente Übertragungsfunktionen, eine Rücktransformation totzeitbehafteter Regelkreise ist im Allgemeinen in geschlossener Form nicht möglich.

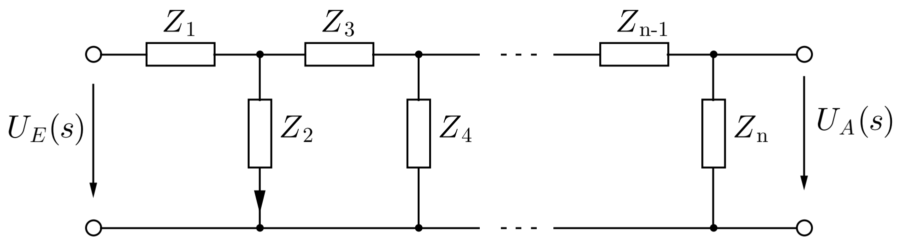
`time_delay(T,n,k)` liefert eine Padénäherung n-ter Ordnung für ein Totzeitsystem mit der Übertragungsfunktion $G(s)=e^{-sT}$. Die Angabe k der Ordnung des Zählerpolynoms ist optional, ihr Defaultwert ist $n-1$.

Padénäherung fünfter Ordnung für die Übertragungsfunktion einer Totzeit $G(s)=\exp(-sT)$

(%i11) `time_delay(T,5);`

(%o11) $$\frac{5s^4T^4-120s^3T^3+1260s^2T^2-6720sT+15120}{s^5T^5+25s^4T^4+300s^3T^3+2100s^2T^2+8400sT+15120}$$

`impedance_chain` erzeugt die Übertragungsfunktion einer Impedanzkette mit beliebig vielen Impedanzen (gerader Anzahl); der letzte (optionale ganzzahlige) Parameter gibt an, wie oft sich die Impedanzkette wiederholt.



Übertragungsfunktion einer Impedanzkette mit vier Elementen

(%i12) `impedance_chain(R,1/(s*C),s*L+R,1/(s*C));`

(%o12) $$\frac{1}{s^2C^2R^2+\left(s^3C^2L+3sC\right)R+s^2CL+1}$$

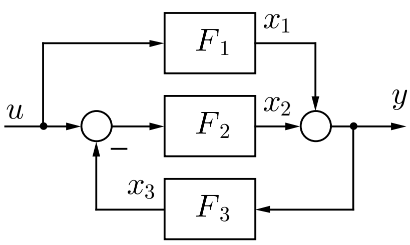
Übertragungsfunktion einer Impedanzkette mit viermal den gleichen Elementen

(%i13) `impedance_chain(R,1/(s*C),4);`

(%o13) $$\frac{1}{s^4C^4R^4+7s^3C^3R^3+15s^2C^2R^2+10sCR+1}$$

`transfer_function(`*eqs*`,`*vars*`,`*u*`,`*y*`)` berechnet die Übertragungsfunktion aus einer Liste mit linearen Gleichungen *eqs*, beispielsweise aus einem Blockschaltbild. *vars* ist die Liste der verwendeten Variablen, nach denen das Gleichungssystem aufzulösen ist, u ist die Eingangsgröße, y die Ausgangsgröße.

Auch Mehrgrößensysteme sind berechenbar; sind u und y *Listen* mit Variablen, so wird die entsprechende Übertragungs*matrix* berechnet.



$$\begin{aligned} x_1 &= F_1\cdot u\\ x_2 &= F_2\cdot(u-x_3)\\ x_3 &= F_3\cdot y\\ y &= x_1+x_2 \end{aligned}$$

Aufstellen eines Gleichungssystems aus dem Blockschaltbild

(%i14) `eqs:[x1=F1*u,x2=F2*(u-x3),x3=F3*y,y=x1+x2];`

(%o14) $[x1=u\,F1,x2=(u-x3)\,F2,x3=y\,F3,y=x2+x1]$

Berechnung der Übertragungsfunktion aus dem Gleichungssystem

```
(%i15) transfer_function(eqs,[x1,x2,x3,y],u,y);
```

(%o15) $\frac{F2+F1}{F2\,F3+1}$

`standard_form`(*F(s)*,*n*) dividiert den Zähler und Nenner von *F(s)* in Abhängigkeit von n durch einen bestimmten Koeffizienten, sodass dieser zu 1 wird:

$n=1$ … führender Zählerkoeffizient von $F(s)$
$n=2$ … niedrigster Zählerkoeffizient von $F(s)$
$n=3$ … führender Nennerkoeffizient von $F(s)$
$n=4$ … niedrigster Nennerkoeffizient von $F(s)$ (default)

```
(%i16) F:(2*s+3)/(4*s^2+5*s+6);
```

(%o16) $\frac{2s+3}{4s^2+5s+6}$

Standardformen, die den führenden oder niedrigsten *Zähler*koeffizienten zu 1 machen:

```
(%i17) [standard_form(F,1),standard_form(F,2)];
```

(%o17) $[\frac{s+1.5}{2.0s^2+2.5s+3.0},\frac{0.667s+1}{1.3333s^2+1.6667s+2.0}]$

Standardformen, die den führenden oder niedrigsten *Nenner*koeffizienten zu 1 machen:

```
(%i18) [standard_form(F,3),standard_form(F,4)];
```

(%o18) $[\frac{0.5s+0.75}{s^2+1.25s+1.5},\frac{0.333s+0.5}{0.667s^2+0.833s+1}]$

4 Laplace Transformation, Sprungantwort

Zur Laplace-Transformation stellt Maxima die Funktion `laplace(f,t,s)` zur Verfügung, zur Rücktransformation `ilt(f,s,t)`. Die Koeffizienten der Zähler- und Nennerpolynome können dabei auch symbolische Werte haben. Allerdings versagt `ilt` bereits bei Nennerpolynomen 3. Grades, wenn keine Nullstellen analytisch gefunden werden.

Bei der in *COMA* enthaltenen Funktion `nilt` werden die Nullstellen des Nennerpolynoms mit `allroots` numerisch berechnet, es können daher gebrochen rationale Funktionen (fast) beliebiger Ordnung rücktransformiert werden, die Polynomkoeffizienten müssen dann jedoch reine Zahlenwerte sein.

`laplace`(*ft*,*timevar*,*lapvar*)	Laplace-Transformation der Funktion *ft* (Bestandteil von Maxima)
`ilt`(*fs*,*lapvar*,*timevar*)	Laplace-Rücktransformation von *fs* (Bestandteil von Maxima)
`nilt`(*fs*,*lapvar*,*timevar*)	Laplace-Rücktransformation von *fs* mit numerisch berechneten Polstellen
`step_response`(*F(s)*,*opts*)	Plotten der Sprungantwort der Übertragungsfunktion $F(s)$

Laplace-Transformation

Laplace-Transformierte einer Funktion

```
(%i1) laplace(t^2*sin(a*t),t,s);
```

(%o1) $$\frac{8\,a\,s^2}{\left(s^2+a^2\right)^3}-\frac{2\,a}{\left(s^2+a^2\right)^2}$$

Laplace-Rücktransformation

```
(%i2) ilt(1/(s^3+2*s^2+2*s+1),s,t);
```

(%o2) $$e^{-\frac{t}{2}}\left(\frac{\sin\left(\frac{\sqrt{3}\,t}{2}\right)}{\sqrt{3}}-\cos\left(\frac{\sqrt{3}\,t}{2}\right)\right)+e^{-t}$$

Die Koeffizienten können auch *symbolisch* sein

```
(%i3) ilt(1/((s+a)^2*(s+b)),s,t);
```

(%o3) $$\frac{e^{-b\,t}}{b^2-2\,a\,b+a^2}+\frac{t\,e^{-a\,t}}{b-a}-\frac{e^{-a\,t}}{b^2-2\,a\,b+a^2}$$

Die Rücktransformation versagt, wenn keine Nullstellen des Nenners analytisch gefunden werden können.

```
(%i4) ilt(1/(s^3+2*s^2+3*s+1),s,t);
```

(%o4) $$\mathrm{ilt}\left(\frac{1}{s^3+2\,s^2+3\,s+1},s,t\right)$$

`nilt` ermittelt die Nenner-Nullstellen numerisch, Nennerpolynome beliebiger Ordnung sind daher möglich.

```
(%i5) nilt(1/(s^3+2*s^2+3*s+1),s,t);
```

(%o5) $$-0.148\,e^{-0.785\,t}\sin(1.3071\,t)-0.545\,e^{-0.785\,t}\cos(1.3071\,t)+0.545\,e^{-0.43\,t}$$

Aufbau einer Liste aus PT2-Elementen mit steigendem Dämpfungsgrad

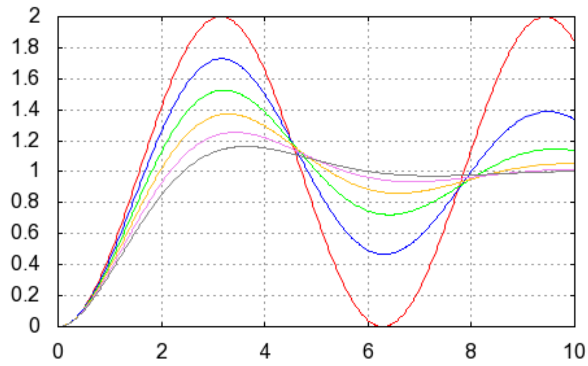
(%i6)
```
pt2li:create_list(1/(s^2+2*d*s+1), d,
        [0.0001,0.1,0.2,0.3,0.4,0.5]);
```

(%o6) $$[\frac{1}{s^2+2.0\,10^{-4}\,s+1},\frac{1}{s^2+0.2\,s+1},\frac{1}{s^2+0.4\,s+1},\frac{1}{s^2+0.6\,s+1},\frac{1}{s^2+0.8\,s+1},\frac{1}{s^2+1.0\,s+1}]$$

Plotten der Sprungantworten; wird die Option `xrange` nicht angegeben, so erfolgt die Wahl des dargestellten Zeitbereiches automatisch.

(%i7)
```
step_response(pt2li)$
```

(%t7)



PT1-Element mit zusätzlicher Totzeit in Padénäherung

(%i8)
```
F:time_delay(2,5)*1/(1+s);
```

(%o8) $$\frac{5\,s^4-60\,s^3+315\,s^2-840\,s+945}{(s+1)\,(2\,s^5+25\,s^4+150\,s^3+525\,s^2+1050\,s+945)}$$

Berechnung der exakten Sprungantwort des totzeitbehafteten PT1-Elementes

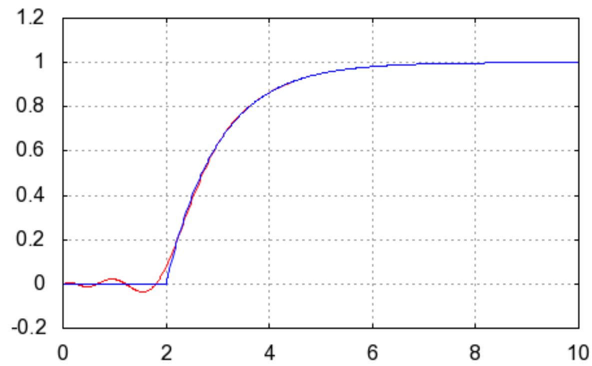
(%i9)
```
ft:unit_step(t-2)*ev(ilt(1/(s*(1+s)),s,t),
t=t-2);
```

(%o9) $$\left(1-e^{2-t}\right)\,\mathrm{unit\_step}\,(t-2)$$

Vergleich der exakten Sprungantwort mit der Padénäherung; die berechnete exakte Sprungantwort ist als Grafikobjekt `explicit` angegeben.

(%i10)
```
step_response([F,explicit(ft,t,0,10)],
yrange=[-0.2,1.2])$
```

(%t10)



5 Frequenzgänge

`bode_plot(`*F(s),opts*`)`	Bodediagramm von $F(j\omega)$
`magnitude_plot(`*F(s),opts*`)`	Amplitudengang des Bodediagrammes von $F(j\omega)$
`logy=false`	Option für `magnitude_plot`, bewirkt eine *lineare* Skalierung des Amplitudenganges
`phase_plot(`*F(s),opts*`)`	Phasengang des Bodediagrammes von $F(j\omega)$
`phase(`*F(s)*`)`	Argument des Frequenzganges $F(j\omega)$ in Grad
`nyquist_plot(`*F(s),opts*`)`	Frequenzgangsortskurve von $F(j\omega)$

Frequenzgänge

Als Parameter sind Übertragungsfunktionen $F(s)$ mit der Laplace-Variable s anzugeben, die von den Plotroutinen automatisch durch $j\omega$ ersetzt wird.

Werden in den Plotfunktionen die Optionen `xrange` und `yrange` nicht angegeben, so wird der dargestellte Bereich automatisch ermittelt. Mit den Optionen `logx=false` und `logy=false` können die Achsen von Bodediagrammen linear skaliert werden. `bode_plot` benötigt für die Option `yrange` eine *Liste mit zwei Bereichen*, jeweils einen für den Amplitudengang und einen für den Phasengang.

Bei der Ortskurve (`nyquist_plot`) ist die Skalierung in x-Richtung und in y-Richtung standardmäßig gleich (`aspect_ratio=-1`), was zu einer unverzerrten Darstellung führt.

Im Gegensatz zur Maxima-Funktion `carg`, die das Argument einer komplexen Zahl (im Bogenmaß) immer im Intervall $-\pi \ldots \pi$ berechnet, ermittelt `phase` die tatsächliche Phasendrehung zwischen Eingangs- und Ausgangsgröße, die beliebig hohe Werte annehmen kann; jede Nullstelle und jede Polstelle bewirkt eine Phasendrehung um $\pi/2$ (bzw. 90 Grad) mit entsprechendem Vorzeichen.

Bei Auftreten von hohen Resonanzen findet in einem kleinen Frequenzbereich eine sehr große Phasendrehung statt. Um auch in solchen Fällen – insbesondere bei der Ortskurve – eine saubere glatte Kurve zu erhalten, muss unter Umständen die Anzahl der Stützstellen für die Plotroutine erhöht werden, was mit der Option `nticks=`*value* erfolgen kann (Defaultwert 500).

Liste aus PT2-Elementen mit steigendem Dämpfungsgrad

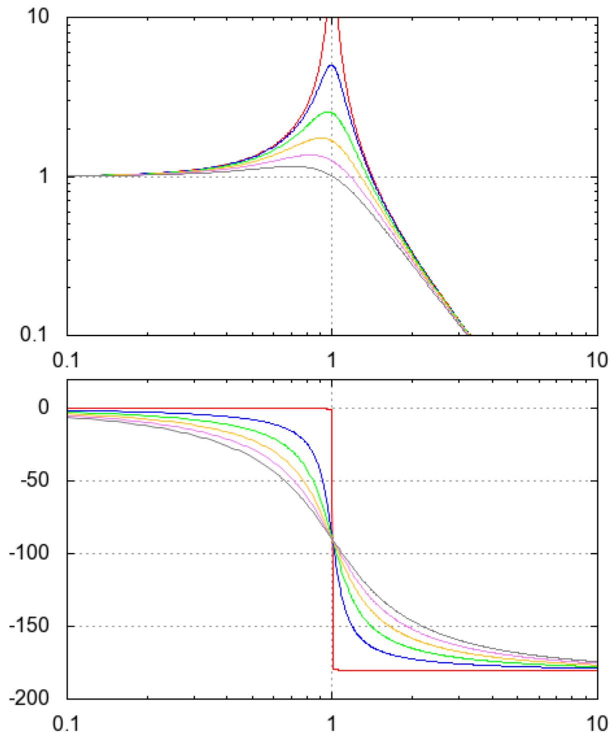
```
(%i1) fli:create_list(1/(s^2+2*d*s+1), d,
         [0.0001,0.1,0.2,0.3,0.4,0.5]);
```

(%o1) $$\left[\frac{1}{s^2+2.0\,10^{-4}\,s+1}, \frac{1}{s^2+0.2\,s+1}, \frac{1}{s^2+0.4\,s+1}, \frac{1}{s^2+0.6\,s+1}, \frac{1}{s^2+0.8\,s+1}, \frac{1}{s^2+1.0\,s+1}\right]$$

Bodediagramm der PT2-Elemente; die Bereiche für die y-Achsen sind in einer *Liste* anzugeben.

(%i2)
```
bode_plot([fli],yrange=[[0.1,10],[-200,20]])$
```

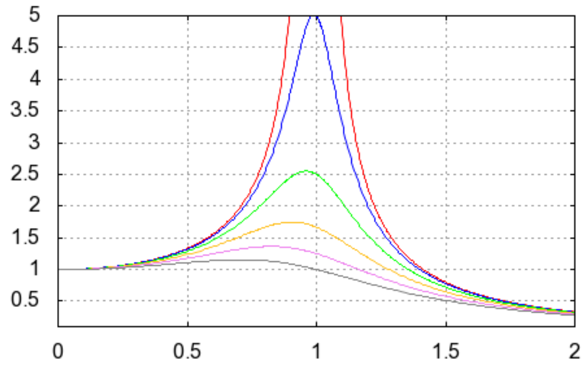
(%t2)



Amplitudengänge der PT2-Elemente mit linearer Skalierung beider Achsen

(%i3)
```
magnitude_plot(fli,xrange=[0,2],
yrange=[0.1,5],logx=false,logy=false)$
```

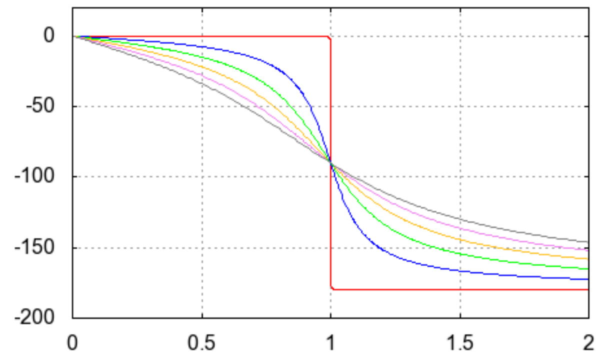
(%t3)



Phasengänge der PT2-Elemente mit linearer Skalierung beider Achsen, die y-Achse ist hier standardmäßig linear skaliert.

(%i4)
```
phase_plot(fli,xrange=[0,2],yrange=[-200,20],
logx=false)$
```

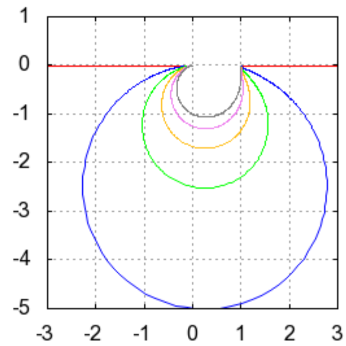
(%t4)



Ortskurven der PT2-Elemente; es kann notwendig sein, die Anzahl der Stützstellen mit der Option `nticks` zu erhöhen.

(%i5)
```
nyquist_plot(fli,xrange=[-3,3],yrange=[-5,1],
    dimensions=[300,300],nticks=2000)$
```

(%t5)



Übertragungsfunktion dritter Ordnung.

(%i6)
```
f:2/(1+2*s+2*s^2+s^3);
```

(%o6) $$\frac{2}{s^3+2s^2+2s+1}$$

Argument von $F(j\omega)$ durch Zerlegung des Frequenzganges in lineare und quadratische Faktoren und Addition der Teilargumente.

(%i7)
```
phase(f);
```

(%o7)
$$57.296\left(-1.0\,\mathrm{atan2}\left(1.0\,\omega,1.0-1.0\,\omega^2\right)-1.0\,\mathrm{atan}\left(1.0\,\omega\right)\right)$$

Mit den Grafikobjekten `points` und `label` kann eine Ortskurve mit ω-Werten markiert und beschriftet werden; die Positionen der Beschriftungen können mit einigen trickreichen Überlegungen geeignet festgelegt werden: die Beschriftung eines Punktes liegt in einem bestimmten Abstand auf einer Normalen zum Richtungsvektor in diesem Punkt.

Beachte: Punkte und Vektoren sind hier als *Listen* (und nicht als *Matrizen*) definiert.

Liste mit ω-Werten zum Markieren der Ortskurve

(%i8)
```
omegali:makelist(0.1*k,k,1,10);
```

(%o8) $[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]$

Umwandlung in den Frequenzgang durch Ersetzen von s durch $j\omega$

(%i9)
```
fom:ev(f,s=%i*omega);
```

(%o9) $$\frac{2}{-i\,\omega^3-2\,\omega^2+2\,i\,\omega+1}$$

Position eines Punktes auf der Ortskurve (Liste aus x- und y-Koordinate):

```
(%i10) dot:[realpart(fom),imagpart(fom)];
```

(%o10) $$\left[\frac{2\left(1-2\,\omega^2\right)}{\left(2\,\omega-\omega^3\right)^2+\left(1-2\,\omega^2\right)^2},\frac{2\left(\omega^3-2\,\omega\right)}{\left(2\,\omega-\omega^3\right)^2+\left(1-2\,\omega^2\right)^2}\right]$$

Die Ableitung gibt einen Richtungsvektor auf der Ortskurve an.

```
(%i11) abl:ratsimp(diff(dot,omega));
```

(%o11) $$\left[\frac{16\,\omega^7-12\,\omega^5-8\,\omega}{\omega^{12}+2\,\omega^6+1},-\frac{6\,\omega^8-20\,\omega^6-6\,\omega^2+4}{\omega^{12}+2\,\omega^6+1}\right]$$

Einheitsvektor normal auf die Ortskurve

```
(%i12) ovec:ratsimp([-abl[2],abl[1]]/
           sqrt(abl[1]^2+abl[2]^2));
```

(%o12) $$\left[\frac{6\,\omega^8-20\,\omega^6-6\,\omega^2+4}{\sqrt{36\,\omega^4+16\,\omega^2+16}\,\sqrt{\omega^{12}+2\,\omega^6+1}},\frac{16\,\omega^7-12\,\omega^5-8\,\omega}{\sqrt{36\,\omega^4+16\,\omega^2+16}\,\sqrt{\omega^{12}+2\,\omega^6+1}}\right]$$

Position der Beschriftung für einen Punkt

```
(%i13) lab:dot+0.3*ovec$
```

Die zu markierenden Ortskurvenpunkte werden als Grafikobjekt `points` definiert.

```
(%i14) dots:points(map(lambda([u],ev(dot,omega=u)),
       omegali))$
```

Die zugehörigen Beschriftungen werden als Grafikobjekt `label` definiert.

```
(%i15) labs:apply(label,map(lambda([u],[string(u),
       ev(lab[1],omega=u),ev(lab[2],omega=u)]),
       omegali))$
```

Einheiskreis als Kurve in Parameterdarstellung

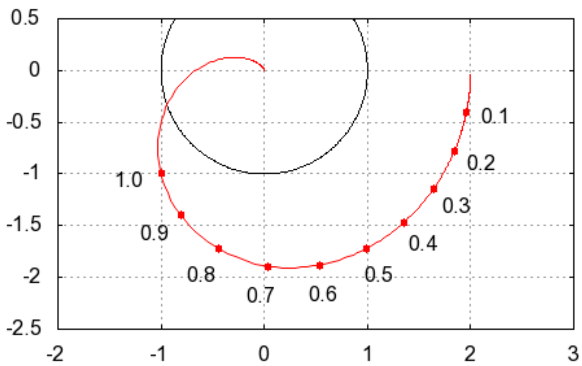
```
(%i16) circ:parametric(cos(t),sin(t),t,0,2*%pi);
```

(%o16) $\mathrm{parametric}\left(\cos(t),\sin(t),t,0,2\,\pi\right)$

Ortskurve mit markierten Ortskurvenpunkten und dem Einheitskreis

```
(%i17) nyquist_plot([circ,f,dots,labs],
       xrange=[-2,3],yrange=[-2.5,0.5],
       point_type=7,color=[black,red,red,black])$
```

(%t17)



6 Untersuchungen in der s-Ebene

6.1 Pol/Nullstellen-Verteilung

`poles`($F(s)$)	Polstellen der Übertragungsfunktion $F(s)$
`zeros`($F(s)$)	Nullstellen der Übertragungsfunktion $F(s)$
`poles_and_zeros`($F(s)$,*opts*)	Darstellung der Pol/Nullstellen-Verteilung einer Übertragungsfunktion $F(s)$ in der s-Ebene

Pol/Nullstellen-Verteilung

Die Funktionen `poles` und `zeros` geben die Pol- bzw. Nullstellen der Übertragungsfunktion als Liste aus, `poles_and_zeros` zeichnet die Pol- und Nullstellen in der s-Ebene. Dabei werden die Polstellen durch ein ×, die Nullstellen durch ein ∘ dargestellt. Standardmäßig ist die Skalierung in x-Richtung und in y-Richtung gleich (`aspect_ratio=-1`), was zu einer unverzerrten Darstellung führt.

Erzeugung einer Liste aus Zufalls-Übertragungsfunktionen

(%i1) `fli:makelist(stable_rantranf(5),k,1,2);`

(%o1) $$\left[\frac{4s^2+9s+6}{2s^5+6s^4+9s^3+10s^2+5s+1}, \frac{3s^2+5s+10}{2s^5+2s^4+10s^3+6s^2+4s+1}\right]$$

Nullstellen

(%i2) `zeros(fli);`

(%o2) $[[0.484\,i-1.125, -0.484\,i-1.125], [1.6245\,i-0.833, -1.6245\,i-0.833]]$

Polstellen

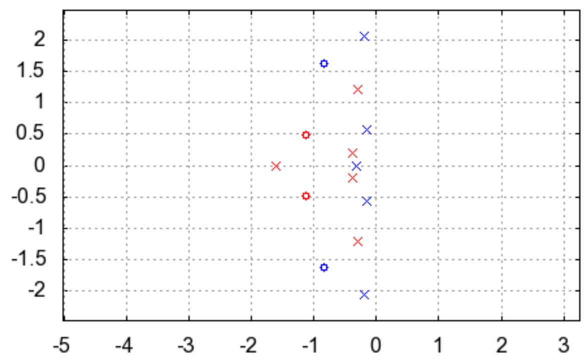
(%i3) `poles(fli);`

(%o3) $[[0.208\,i-0.39, -0.208\,i-0.39, 1.2235\,i-0.305, -1.2235\,i-0.305, -1.609], [0.576\,i-0.151, -0.576\,i-0.151, -0.327, 2.0685\,i-0.185, -2.0685\,i-0.185]]$

Pol/Nullstellenverteilungen in der komplexen s-Ebene

(%i4)
```
poles_and_zeros(fli)$
```

(%t4)



6.2 Wurzelortskurven

`root_locus(`*F(s,k),opts*`)`	Wurzelortskurve einer Übertragungsfunktion $F(s,k)$ mit einem freien Parameter k in der s-Ebene
`trange=[`*min,max*`]`	Bereich für den freien Parameter k, default: `[0.001,100]`
`nticks=`n	Anzahl der berechneten Ortskurvenpunkte, default: 500

Wurzelortskurven

`root_locus` zeichnet die Wurzelortskurve einer Übertragungsfunktion $F(s)$ in Abhängigkeit eines freien Parameters k, der aber nicht (wie bei den „klassischen" Wurzelortskurven) die Verstärkung der offenen Regelschleife sein muss, sondern ein beliebiger Parameter, der die Übertragungsfunktion beeinflusst. Werden mehrere Übertragungsfunktionen angegeben, können die Namen der Parameter unterschiedlich sein, der Parameter*bereich* ist aber für alle Übertragungsfunktionen gleich, angegeben durch die Option `trange`.

Die Berechnung der Ortskurvenpunkte erfolgt über den Parameterbereich logarithmisch verteilt, der Parameter darf daher nur positive Werte annehmen.

Die Anfangspunkte der Wurzelortskurven werden durch ein $\times$, die Endpunkten durch ein $\circ$ dargestellt. Wenn der Parameter die Verstärkung der offenen Regelschleife $F_O(s)$ ist, sein Startwert hinreichend klein und sein Endwert hinreichend groß gewählt wird, entspricht das den Polstellen bzw. den Nullstellen von $F_O(s)$.

Erzeugung einer Liste von Regelkreisen aus Übertragungsfunktionen mit jeweils einer anderen Nullstelle a und der veränderlichen Verstärkung k

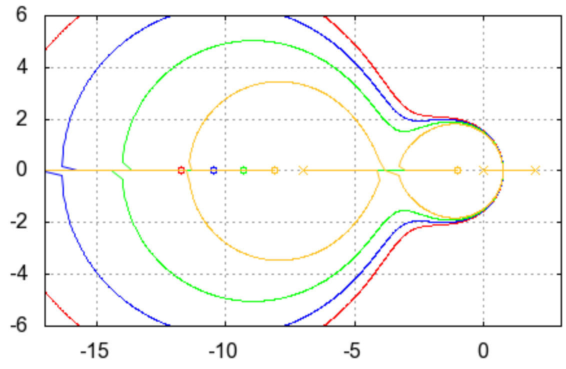
(%i5)
```
fli:closed_loop(makelist(k*((s-a)*(s+1))
    /(s*(s-2)*(s+7)),a,-11,-8));
```

(%o5)
$$\left[\frac{k\,s^2+12\,k\,s+11\,k}{s^3+(k+5)\,s^2+(12\,k-14)\,s+11\,k},\frac{k\,s^2+11\,k\,s+10\,k}{s^3+(k+5)\,s^2+(11\,k-14)\,s+10\,k},\frac{k\,s^2+10\,k\,s+9\,k}{s^3+(k+5)\,s^2+(10\,k-14)\,s+9\,k},\frac{k\,s^2+9\,k\,s+8\,k}{s^3+(k+5)\,s^2+(9\,k-14)\,s+8\,k}\right]$$

Wurzelortskurven in Abhängigkeit der „wandernden“ Nullstelle

(%i6)
```
root_locus(fli,xrange=[-17,3],
      yrange=[-6,6],nticks=5000)$
```

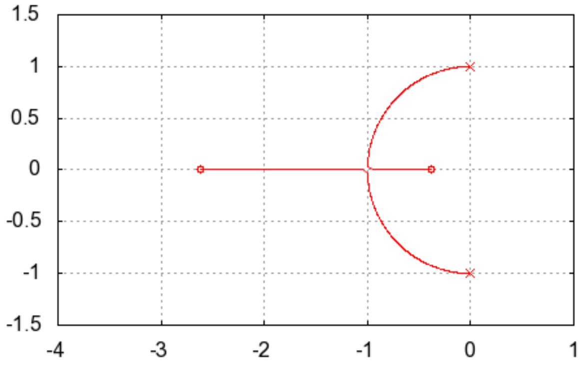
(%t6)



Wurzelortskurve eines PT2-Elementes mit dem Dämpfungsgrad als Parameter

(%i7)
```
root_locus(1/(s**2+a*s+1),xrange=[-4,1],
      trange=[1e-4,3],nticks=5000)$
```

(%t7)



7 Stabilitätsverhalten

7.1 Stabilität

`stablep(`*F(s)*`)`	Überprüft das Sytem mit der Übertragungsfunktion $F(s)$ auf Stabilität
`stability_limit(`*F(s,k)*`)`	Ermittlung der Stabilitätsgrenze der Übertragungsfunktion $F(s,k)$ in Bezug auf den Parameter k
`hurwitz(`*p(s)*`)`	Berechnung der Hurwitz-Determinanten des Polynoms $p(s)$
`stable_area(`*F(s,a,b),a,b,opts*`)`	Darstellung des Stabilitätsbereiches der Übertragungsfunktion $F(s,a,b)$ in der a/b-Parameterebene

Stabilität

Die Funktion `stability_limit(`*F(s,k)*`)` liefert Bedingungen in der Form

k=value, `omega=`*value*,

für rein imaginäre Pole, was für gängige Systeme mit der Stabilitätsgrenze gleichzusetzen ist. Dabei ist der Wert von `omega` die Kreisfrequenz der Dauerschwingung an der Stabilitätsgrenze.

Eine exakte Stabilitätsaussage liefert das *Hurwitz-Kriterium*: Alle Nullstellen des Polynoms $p(s)$ haben genau dann einen negativen Realteil (d. h. die Übertragungsfunktion mit dem Nenner $p(s)$ ist genau dann stabil), wenn alle Hurwitz-Determinanten einen Wert größer Null ergeben. Die Funktion `hurwitz(`*p(s)*`)` liefert eine Liste der Hurwitz-Determinanten, die Koeffizienten von $p(s)$ können dabei auch symbolische Werte haben.

`stable_area` stellt die Stabilitätsgrenze einer Übertragungsfunktion in Abhängigkeit zweier Parameter a und b in der a/b-Parameterebene dar. Werden die Optionen `xrange` und `yrange` nicht angegeben, so erfolgt die Darstellung jeweils im Bereich $0\ldots1$.

Die Beurteilung der Stabilitätsgüte kann mit Hilfe des Phasenrandes α_r oder des Amplitudenrandes A_r erfolgen, die zugehörigen Kreisfrequenzen sind die Durchtrittsfrequenz ω_D bzw. die Phasenschnittfrequenz ω_r.

Zufalls-Übertragungsfunktion

```
(%i1) f:stable_rantranf(5);
```

(%o1) $$\frac{4s^2+9s+6}{2s^5+6s^4+9s^3+10s^2+5s+1}$$

Berechnung des geschlossenen Regelkreises mit einer Reglerverstärkung k

```
(%i2) fw:closed_loop(k*f);
```

(%o2) $$\frac{4ks^2+9ks+6k}{2s^5+6s^4+9s^3+(4k+10)s^2+(9k+5)s+6k+1}$$

Ermittlung der Stabilitätsgrenze; das Ergebnis ist eine Liste mit der Grenzverstärkung k und der Kreisfrequenz der Dauerschwingung ω.

```
(%i3) lim:stability_limit(fw,k);
```

(%o3) $[[k = 0.468, \omega = 1.2554]]$

Das Hurwitz-Kriterium liefert exakte Aussagen; der Regelkreis ist genau dann stabil, wenn alle Listenelemente positiv sind.

```
(%i4) ratsimp(hurwitz(denom(fw)));
```

(%o4) $[34-8k, -32k^2-196k+172, -288k^3-988k^2-776k+610, -1728k^4-6216k^3-5644k^2+2884k+610]$

Erstellung einer Liste von Verstärkungen; oberhalb, an und unterhalb der Stabilitätsgrenze

```
(%i5) kli:ev([1.1*k,k,0.9*k],lim);
```

(%o5) $[0.515, 0.468, 0.422]$

Berechnung der Übertragungsfunktionen der zugehörigen offenen Regelschleifen,

```
(%i6) foli:create_list(k*f,k,kli);
```

(%o6) $$\left[\frac{0.515\left(4s^2+9s+6\right)}{2s^5+6s^4+9s^3+10s^2+5s+1}, \frac{0.468\left(4s^2+9s+6\right)}{2s^5+6s^4+9s^3+10s^2+5s+1}, \frac{0.422\left(4s^2+9s+6\right)}{2s^5+6s^4+9s^3+10s^2+5s+1}\right]$$

sowie der geschlossenen Regelkreise

```
(%i7) fwli:closed_loop(foli)$
```

Stabilitätsprüfung, `stablep` liefert hier für den grenzstabilen Fall `false`

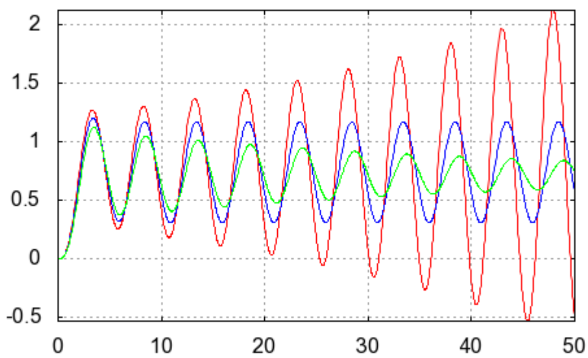
```
(%i8) stablep(fwli);
```

(%o8) `[false,true,true]`

Kontrolle der Rechnung an Hand der Sprungantworten, die Periodendauer der Dauerschwingung an der Stabilitätsgrenze gemäß $T = 2\pi/\omega$ ergibt ziemlich exakt den Wert 5.

```
(%i9) step_response(fwli,xrange=[0,50])$
```

(%t9)



Liste von PT5-Elementen mit zwei freien Parametern a und b

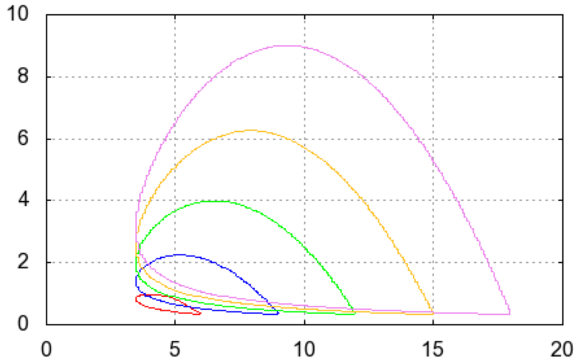
```
(%i10) fli:makelist(1/(s^5+3*s^4+k*s^3+a*s^2+b*s+1),k,2,6);
```

(%o10) $$\left[\frac{1}{s^5+3s^4+2s^3+as^2+bs+1}, \frac{1}{s^5+3s^4+3s^3+as^2+bs+1}, \frac{1}{s^5+3s^4+4s^3+as^2+bs+1}, \frac{1}{s^5+3s^4+5s^3+as^2+bs+1}, \frac{1}{s^5+3s^4+6s^3+as^2+bs+1}\right]$$

Darstellung des Stabilitätsbereiches in Abhängigkeit der Parameter a und b

(%i11) `stable_area(fli,a,b,xrange=[0,20],yrange=[0,10])$`

(%t11)



7.2 Stabilitätsgüte

`gain_crossover`$(F(s))$	Berechnung der Durchtrittsfrequenzen ω_D von $F(j\omega)$ für die gilt: $\lvert F(j\omega_D)\rvert = 1$
`phase_margin`$(F(s))$	Berechnung des Phasenrandes α_r von $F(j\omega)$ in Grad
`phase_crossover`$(F(s))$	Berechnung der Phasenschnittfrequenz ω_r von $F(j\omega)$ für die gilt: $\arg F(j\omega_r) = -\pi$
`gain_margin`$(F(s))$	Berechnung des Amplitudenrandes A_r von $F(j\omega)$
`damping`$(F(s))$	Dämpfung von $F(j\omega)$ (negativer Realteil des rechtesten Poles)
`damping_ratio`$(F(s))$	minimaler Dämpfungs*grad* aller Polpaare von $F(j\omega)$

Stabilitätsgüte

Durchtrittsfrequenzen ω_D, können (insbesondere bei Resonanz) auch mehrfach auftreten.

(%i12) `float(gain_crossover(foli));`

(%o12) $[[\omega = 1.3052], [\omega = 1.2554], [\omega = 1.1755]]$

Phasenränder α_r in Grad, instabile Regelkreise haben einen negativen Wert.

(%i13) `phase_margin(foli);`

(%o13) $[-9.1522, 8.63161\,10^{-6}, 15.194]$

Phasenschnittfrequenzen ω_r, können auch mehrfach auftreten.

(%i14) `float(phase_crossover(foli));`

(%o14) $[[\omega = 1.2554], [\omega = 1.2554], [\omega = 1.2554]]$

Amplitudenränder A_r, instabile Regelkreise haben einen Wert kleiner 1.

(%i15) `gain_margin(foli);`

(%o15) $[0.909, 1.0, 1.1111]$

Die Dämpfung ist der negative Realteil des am weitest rechts liegenden Poles oder Polpaares.

(%i16) `damping(fwli);`

(%o16) $[-0.0232, 1.54999\,10^{-8}, 0.0249]$

8 Optimierung

Das Güteintegral nach dem ISE-Kriterium kann gemäß dem Parseval-Theorem direkt im Laplace-Bereich berechnet werden [6]:

$$I_{ISE} = \int_0^\infty e^2(t)\,\mathrm{d}t = \frac{1}{2\pi j}\int_{-j\infty}^{j\infty} E(s)\cdot E(-s)\,\mathrm{d}s$$

Dabei ist $e(t)$ eine mit steigender Zeit gegen Null strebende Funktion, bei Optimierungsaufgaben üblicherweise die Regelabweichung. Gemäß [5] gelingt die Berechnung des Integrals rein algebraisch.

`ise`$(E(s))$	Güteintegral der Funktion $e(t)$ nach dem ISE-Kriterium

Integrale Gütemaße

Ableiten des Integrals nach freien Parametern (z. B. Reglerparametern) und Nullsetzen der Ableitungen liefert die optimalen Werte der Parameter.

Übertragungsfunktion mit zwei freien Parametern a und b	(%i1)	`f:1/(s**3+a*s**2+b*s+1);`
	(%o1)	$\frac{1}{s^3+a\,s^2+b\,s+1}$
Berechnung der Abweichung vom Stationärwert bei Anregung mit der Sprungfunktion	(%i2)	`x:ratsimp((1-f)/s);`
	(%o2)	$\frac{s^2+a\,s+b}{s^3+a\,s^2+b\,s+1}$
Güteintegral nach dem ISE-Kriterium	(%i3)	`iise:ise(x);`
	(%o3)	$\frac{a\,b^2-b+a^2}{2\,a\,b-2}$
Ableiten nach den Parametern a und b (Berechnung der Jacobimatrix)	(%i4)	`abl:ratsimp(jacobian([iise],[a,b]));`
	(%o4)	$\begin{pmatrix}\frac{a^2\,b-2\,a}{2\,a^2\,b^2-4\,a\,b+2} & \frac{a^2\,b^2-2\,a\,b-a^3+1}{2\,a^2\,b^2-4\,a\,b+2}\end{pmatrix}$
Beschränkung auf reelle Lösungen beim Lösen von Gleichungssystemen	(%i5)	`realonly:true;`
	(%o5)	$true$
Auflösen nach a und b, die Ableitungen werden hier automatisch 0 gesetzt.	(%i6)	`res:solve(abl[1],[a,b]);`
	(%o6)	$[[a=1,b=2]]$
Einsetzten in f liefert die „optimale“ Übertragungsfunktion.	(%i7)	`fopt:ev(f,res);`
	(%o7)	$\frac{1}{s^3+s^2+2\,s+1}$

9 Reglerentwurf

`gain_optimum(`*Fs(s),Fr(s)*`)`	Dimensionierung eines Reglers nach dem Betragsoptimum.

Reglerentwurfsverfahren

`gain_optimum` ermittelt einen betragsoptimalen Regler $F_R(s)$ zu einer gegebenen Regelstecke $F_(s)$. Die Struktur des Reglers und die Namen für die Reglerparameter sind grundsätzlich frei wählbar. Ob tatächlich Lösungen für die Reglerparameter gefunden werden, hängt natürlich von der Sinnhaftigkeit der Wahl ab (ein PT1-Regler wird beispielsweise vermutlich keine Lösung ergeben).

Übertragungsfunktion einer Regelstrecke

```
(%i1) fs:2/((1+5*s)*(1+s)**2*(1+0.3*s));
```

(%o1) $$\frac{2}{(0.3\,s+1)\,(s+1)^2\,(5\,s+1)}$$

Liste bestehend aus einem I-, PI- und einem PID-Regler

```
(%i2) [fri,frpi,frpid]:[1/(s*Ti),
      kr*(1+1/(s*Tn)),(1+s*Ta)*(1+s*Tb)/(s*Tc)];
```

(%o2) $$[\frac{1}{s\,Ti},kr\left(\frac{1}{s\,Tn}+1\right),\frac{(s\,Ta+1)\,(s\,Tb+1)}{s\,Tc}]$$

Betragsoptimum für den I-Regler

```
(%i3) g1:gain_optimum(fs,fri);
```

(%o3) $$[Ti=\frac{146}{5},Ti=0]$$

Betragsoptimum für den PI-Regler; die Reglernullstelle kompensiert näherungsweise die größte Streckenzeitkonstante.

```
(%i4) g2:gain_optimum(fs,frpi);
```

(%o4) $$[kr=\frac{206057}{349320},Tn=\frac{206057}{40190}]$$

Betragsoptimum für den PID-Regler; die Reglernullstellen kompensieren näherungsweise die beiden größten Streckenzeitkonstanten.

```
(%i5) g3:gain_optimum(fs,frpid);
```

(%o5) $$[Ta=\frac{4019\,\sqrt{53}\,\sqrt{67}\,\sqrt{1297}\,\sqrt{35765423}-55265693971}{730\,\sqrt{53}\,\sqrt{67}\,\sqrt{1297}\,\sqrt{35765423}-12006960570},$$
$$Tb=\frac{\sqrt{164722913143681}+22787789}{7126660},Tc=\frac{1289808}{356333}]$$

Einsetzen der Ergebnisse in die Regler-Übertragungsfunktionen

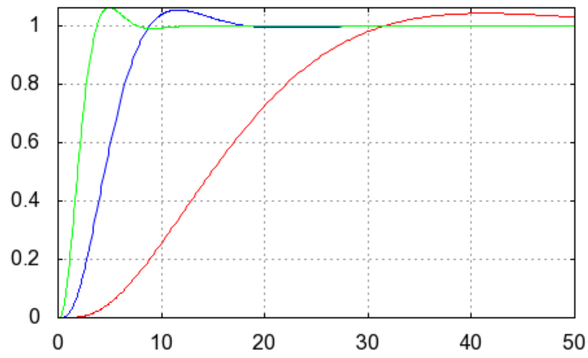
```
(%i6) reli:float(ev([fri,frpi,frpid],[g1,g2,g3]));
```

(%o6) $$[\frac{0.0342}{s},0.59\left(\frac{0.195}{s}+1.0\right),$$
$$\frac{0.276\,(1.3966\,s+1.0)\,(4.9984\,s+1.0)}{s}]$$

Die Sprungantworten bestätigen ca. 5% Überschwingen und Anregelzeiten im Ausmaß von etwa 4,7 mal der Summe der verbleibenden Zeitkonstanten.

(%i7)
```
step_response(float(ev(closed_loop(reli*fs),
res)));
```

(%t7)



(%o7)

Die Regelstrecke kann auch symbolische Koeffizienten haben.

(%i8)
```
fs:2/((1+a*s)*(1+s**2)*(1+b*s));
```

(%o8) $$\frac{2}{(a\,s+1)\,(b\,s+1)\,\left(s^2+1\right)}$$

Das Ergebnis sind Formeln für die optimalen Reglerparameter.

(%i9)
```
gain_optimum(fs,frpi);
```

(%o9) $$[kr=\frac{b^2+a^2-1}{4\,a\,b},$$
$$Tn=\frac{b^3+a\,b^2+\left(a^2-1\right)\,b+a^3-a}{b^2+a\,b+a^2-1}]$$

10 Zustandsraum

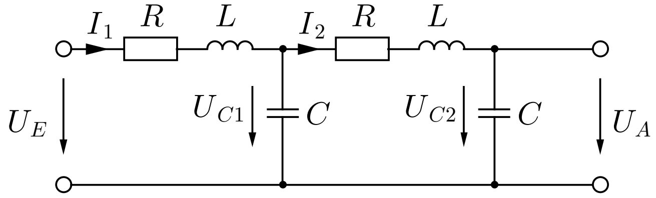
System: [A,B,C,D]	Definition eines linearen Systems als Liste aus den Zustandsmatrizen A, B, C und D
`systemp(`A,B,C[,D]`)` `systemp(`*System*`)`	Überprüft, ob *System* ein gültiges lineares System aus Zustandsmatrizen ist.
`nsystemp(`A,B,C[,D]`)` `nsystemp(`*System*`)`	Überprüft, ob *System* ein gültiges lineares System ist, bei dem alle Matrizenelemente Zahlenwerte sind.
`transfer_function(`A,B,C[,D]`)` `transfer_function(`*System*`)`	Berechnung der Übertragungsfunktion (Übertragungsmatrix) aus der Zustandsraumdarstellung
`controller_canonical_form(`f`)`	Berechnung der Regelungsnormalform aus der Übertragungsfunktion f
`observer_canonical_form(`f`)`	Berechnung der Beobachtungsnormalform aus der Übertragungsfunktion f
`controllability_matrix(`A,B`)` `controllability_matrix(`*System*`)`	Berechnung der Steuerbarkeitsmatrix
`observability_matrix(`A,C`)` `observability_matrix(`*System*`)`	Berechnung der Beobachtbarkeitsmatrix

Zustandsraumdarstellung

Unter *System* ist eine Zusammenfassung der vier Zustandsmatrizen **A** (Systemmatrix), **B** (Eingangsmatrix), **C** (Ausgangsmatrix) und **D** (Durchgangsmatrix) in einer Liste zu verstehen. Bei Systemen ohne Durchgriff kann **D** entfallen, bei Eingrößensystemen kann **D** ein Skalar d sein.

Allen Funktionen, denen ein *System* als Parameter übergeben wird, können auch direkt die entsprechenden Zustandsmatrizen (ohne Zusammenfassung zu einer Liste) übergeben werden.

Elektrischer Vierpol mit Zustandsgleichungen:



$$\begin{aligned}
\frac{\mathrm{d}i_1}{\mathrm{d}t} &= \frac{1}{L}\cdot(u_e - R\cdot i_1 - u_{C1})\\
\frac{\mathrm{d}i_2}{\mathrm{d}t} &= \frac{1}{L}\cdot(u_{C1} - u_{C2} - R\cdot i_2)\\
\frac{\mathrm{d}u_{C1}}{\mathrm{d}t} &= \frac{1}{C}\cdot(i_2 - i_1)\\
\frac{\mathrm{d}u_{C2}}{\mathrm{d}t} &= \frac{1}{C}\cdot i_2
\end{aligned}$$

Aus den Zustandsgleichungen ergeben sich direkt die Zustandsmatrizen **A**, **B** und **C**; ein direkter Durchgriff von der Eingangsspannung U_E zur Ausgangsspannung U_A ist nicht vorhanden, daher gilt $\mathbf{D} = 0$, das daher weggelassen werden kann.

Systemmatrix **A** des Schaltkreises

(%i1) `A:matrix([-R/L,0,-1/L,0],[0,-R/L,1/L,-1/L],[1/C1,-1/C1,0,0],[0,1/C1,0,0]);`

(%o1) $$\begin{pmatrix} -\frac{R}{L} & 0 & -\frac{1}{L} & 0 \\ 0 & -\frac{R}{L} & \frac{1}{L} & -\frac{1}{L} \\ \frac{1}{C1} & -\frac{1}{C1} & 0 & 0 \\ 0 & \frac{1}{C1} & 0 & 0 \end{pmatrix}$$

Eingangsmatrix **B** des Schaltkreises

(%i2) `B:matrix([1/L],[0],[0],[0]);`

(%o2) $$\begin{pmatrix} \frac{1}{L} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Ausgangsmatrix **C** des Schaltkreises

(%i3) `C:matrix([0,0,0,1]);`

(%o3) $\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$

Zusammenfassen der Zustandsmatrizen zu einem *System*

(%i4) `circuit:[A,B,C];`

(%o4) $$\left[\begin{pmatrix} -\frac{R}{L} & 0 & -\frac{1}{L} & 0 \\ 0 & -\frac{R}{L} & \frac{1}{L} & -\frac{1}{L} \\ \frac{1}{C1} & -\frac{1}{C1} & 0 & 0 \\ 0 & \frac{1}{C1} & 0 & 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{L} \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}\right]$$

Die Liste mit den Zustandsmatrizen ist ein gültiges *System* …

(%i5) `systemp(circuit);`

(%o5) *true*

… die Matrizen haben aber nicht ausschließlich Zahlenwerte als Elemente.

(%i6) `nsystemp(circuit);`

(%o6) *false*

`transfer_function` berechnet aus den Zustandsmatrizen die Übertragungsfunktion (oder die Übertragungsmatrix im Mehrgrößenfall). Diese Funktion ist in gewisser Weise polymorph: werden als Parameter keine Zustandsmatrizen sondern eine Liste von linearen Gleichungen und Listen von Variablen übergeben, so wird aus diesen Gleichungen eine Übertragungsfunktion ermittelt (siehe Abschnitt 3).

Berechnung der Übertragungsfunktion; sowohl die Angabe eines *Systems*, …

(%i7) `f:transfer_function(circuit);`

(%o7) $$\frac{1}{s^2\,C1^2\,R^2 + \left(2\,s^3\,C1^2\,L + 3\,s\,C1\right)\,R + s^4\,C1^2\,L^2 + 3\,s^2\,C1\,L + 1}$$

... als auch die Angabe der einzelnen Zustandsmatrizen ist möglich.

(%i8) `transfer_function(A,B,C);`

(%o8)
$$\frac{1}{s^2\,C1^2\,R^2+\left(2\,s^3\,C1^2\,L+3\,s\,C1\right)\,R+s^4\,C1^2\,L^2+3\,s^2\,C1\,L+1}$$

Die direkte Berechnung als Impedanzkette liefert erwartungsgemäß das selbe Ergebnis.

(%i9) `f:impedance_chain(R+s*L,1/(s*C1),2);`

(%o9)
$$\frac{1}{s^2\,C1^2\,R^2+\left(2\,s^3\,C1^2\,L+3\,s\,C1\right)\,R+s^4\,C1^2\,L^2+3\,s^2\,C1\,L+1}$$

Die Berechnung der Zustandsmatrizen aus der Übertragungsfunktion kann nach der Regelungsnormalform oder der Beobachtungsnormalform erfolgen:

Berechnung der Zustandsmatrizen nach der Regelungsnormalform

(%i10) `circ1:controller_canonical_form(f);`

(%o10)
$$\left[\begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -\frac{1}{C1^2\,L^2} & -\frac{3\,R}{C1\,L^2} & -\frac{C1^2\,R^2+3\,C1\,L}{C1^2\,L^2} & -\frac{2\,R}{L}\end{pmatrix},\begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix},\begin{pmatrix}\frac{1}{C1^2\,L^2} & 0 & 0 & 0\end{pmatrix},0\right]$$

Berechnung der Zustandsmatrizen nach der Beobachtungsnormalform

(%i11) `circ2:observer_canonical_form(f);`

(%o11)
$$\left[\begin{pmatrix}0 & 0 & 0 & -\frac{1}{C1^2\,L^2}\\ 1 & 0 & 0 & -\frac{3\,R}{C1\,L^2}\\ 0 & 1 & 0 & -\frac{C1^2\,R^2+3\,C1\,L}{C1^2\,L^2}\\ 0 & 0 & 1 & -\frac{2\,R}{L}\end{pmatrix},\begin{pmatrix}\frac{1}{C1^2\,L^2}\\ 0\\ 0\\ 0\end{pmatrix},\begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix},0\right]$$

Steuerbarkeitsmatrix

(%i12) `h1:ratsimp(controllability_matrix(A,B));`

(%o12)
$$\begin{pmatrix}\frac{1}{L} & -\frac{R}{L^2} & \frac{C1\,R^2-L}{C1\,L^3} & -\frac{C1\,R^3-2\,L\,R}{C1\,L^4}\\ 0 & 0 & \frac{1}{C1\,L^2} & -\frac{2\,R}{C1\,L^3}\\ 0 & \frac{1}{C1\,L} & -\frac{R}{C1\,L^2} & \frac{C1\,R^2-2\,L}{C1^2\,L^3}\\ 0 & 0 & 0 & \frac{1}{C1^2\,L^2}\end{pmatrix}$$

Beobachtbarkeitsmatrix

(%i13) `h2:observability_matrix(circuit);`

(%o13)
$$\begin{pmatrix}0 & 0 & 0 & 1\\ 0 & \frac{1}{C1} & 0 & 0\\ 0 & -\frac{R}{C1\,L} & \frac{1}{C1\,L} & -\frac{1}{C1\,L}\\ \frac{1}{C1^2\,L} & \frac{R^2}{C1\,L^2}-\frac{2}{C1^2\,L} & -\frac{R}{C1\,L^2} & \frac{R}{C1\,L^2}\end{pmatrix}$$

Das System ist steuerbar und beobachtbar.

(%i14) `[rank(h1),rank(h2)];`

(%o14) $[4,4]$

11 Diverse Funktionen

Die Routinen von *COMA* verwenden einige Funktionen, die nicht speziell regelungstechnischer Natur sind, die aber für den Anwender in verschiedensten Bereichen nützlich sein können.

`chop`(x)	Ersetzen aller Zahlen im Ausdruck x, die kleiner als 10^{-10} sind, durch 0
`coefficient_list`(p,x)	Liste der Koeffizienten des Polynoms p in der Variablen x
`set_option`(*name*=*val*,*list*)	Setzen oder Hinzufügen eines Listenelements als Hasheintrag zur Liste *list*
`delete_option`(*name*,*list*)	Löschen des Hasheintrags mit dem Namen *name* aus der Liste *list*
`option_exists`(*name*,*list*)	Überprüft, ob ein Hasheintrag mit dem Namen *name* in der Liste *list* existiert
`list_option_exists`(*name*,*list*)	Überprüft, ob der Hasheintrag mit dem Namen *name* existiert und eine Liste ist

Diverse Funktionen

`coefficient_list` erstellt eine Liste von Polynomkoeffizienten in steigender Reihenfolge:

Polynom in der Variablen x

(%i1) `p:5*(x+y)^2+a*x^5;`

(%o1) $5\,(y+x)^2+a\,x^5$

Liste mit den Polynomkoeffizienten

(%i2) `coefficient_list(p,x);`

(%o2) $[5\,y^2, 10\,y, 5, 0, 0, a]$

Die Funktion `chop` entfernt aus einem Ausdruck alle Zahlen, die kleiner als 10^{-10} sind. Damit lassen sich numerische Fehler „ausbügeln“:

Wenn die Numerik verrückt spielt, ...

(%i3) `x3:expand((s^2-1.1*s+1.1)`
`*(s^2+1.1*s+1.1)*(s^2+1));`

(%o3) $s^6+1.99\,s^4+2.22045\,10^{-16}\,s^3+2.2\,s^2+1.21$

... können kleine (auf Grund von numerischen Fehlern) auftretende Zahlenwerte beseitigt werden.

(%i4) `chop(x3);`

(%o4) $s^6+1.99\,s^4+2.2\,s^2+1.21$

Ein assoziatives Array (Hash), bestehend aus Schlüssel-Wert-Paaren, wird in Maxima mit einer Liste realisiert. Es eignet sich unter anderem besonders zum Verwalten von Standardeinstellungen („Optionen“) und für benannte Parameter von Funktionen (z. B. für Grafikroutinen).

Einige Routinen erleichteren den Umgang mit assoziativen Arrays:

Liste mit Schlüssel-Wert-Paaren (Hash-Einträgen)	`(%i5) opts:[color=blue,height=700,width=400];` `(%o5) [color=blue,height=700,width=400]`
Ändern eines Hash-Wertes	`(%i6) set_option(color=red,opts);` `(%o6) [height=700,width=400,color=red]`
Existiert ein Schlüssel nicht, so wird ein neuer Eintrag hinzugefügt.	`(%i7) set_option(title="Test",opts);` `(%o7) [height=700,width=400,color=red,title=Test]`
Entfernen eines Hash-Eintrags	`(%i8) delete_option(color,opts);` `(%o8) [height=700,width=400,title=Test]`
Überprüfen, ob ein Hash-Eintrag existiert	`(%i9) option_exists(height,opts);` `(%o9) true`
Lesen eines Wertes aus dem Hash	`(%i10) get_option(height,opts);` `(%o10) 700`
Zurückgabe eines Defaultwertes, wenn ein Schlüssel nicht existiert	`(%i11) get_option(color,opts,red);` `(%o11) red`

Die Funktion `get_option` zum Lesen eines Wertes wäre prinzipiell nicht notwendig, da es zu diesem Zweck die Maxima-Funktion `assoc` gibt. Im Gegensatz zu `assoc` akzeptiert `get_option` auch Listen, die nicht nur Schlüssel-Wert-Paare enthalten, sondern auch beliebige Ausdrücke (wovon von anderen COMA-Funktionen intern Gebrauch gemacht wird).

Literaturverzeichnis

[1] Maxima Development Team: *Maxima Reference Manual V.5.23.* 2010.

[2] Diverse Autoren: *Gnuplot, An Interactive Plotting Program.* 2007.

[3] Ameling W.: *Laplace-Transformation.* Bertelsmann Universitätsverlag Düsseldorf 1975.

[4] Haager W.: *Regelungstechnik.* Hölder Pichler Tempsky Wien 1997.

[5] Newton G., Gould L., Kaiser J.: *Analytical Design of Linear Feedback Control.* Wiley, New York 1957.

[6] Unbehauen H.: *Regelungstechnik I.* Vieweg Braunschweig, Wiesbaden 1984.

[7] Föllinger O.: *Regelungstechnik.* Elitera Verlag Berlin 1978.

[8] Zak S.: *Systems and Control.* Oxford University Press, New York, Oxford 2003.