HTBLuVA St. Pölten
Elektrotechnik

3100 St. Pölten | Waldstrasse 3 | 02742-75051-300
et@htlstp.ac.at | http://et.htlstp.ac.at
Höhere Abteilung | Fachschule | Abendschule

COMA
Control Engineering with Maxima

Wilhelm Haager
HTL St. Pölten, Department Electrical Engineering
wilhelm.haager@htlstp.ac.at

Version 1.8, April 18, 2017

Contents

1 Introduction

1.1 Notions

Maxima: Open-Source descendant of the computeralgebra system *Macsyma*, which initially was developed 1967–1982 at the MIT by order of the US Department of Energy. 1989 a version of Macsyma was published with the name *Maxima* under the *GNU General Public Licence*, which is now being developed further by an independent group of users. Maxima is written in Lisp and contains many aspects of functional programming.

Due to its power and free availability there is no reason to use it *not*.

wxMaxima: One of several graphical user interfaces for Maxima. It enables input and editing expressions in a working window, as well as the documenting calculations with text and images. Maxima outputs results and (on demand) graphics into that working window.

Working sessions can be saved, loaded and re-executed; the most common commands are accessible via menus and control buttons (for notorious mouse-clickers) . Working sessions can be exported as HTML or as LaTeX-file. Export to LaTeX may require some corrections by hand of the resulting TEX-file.

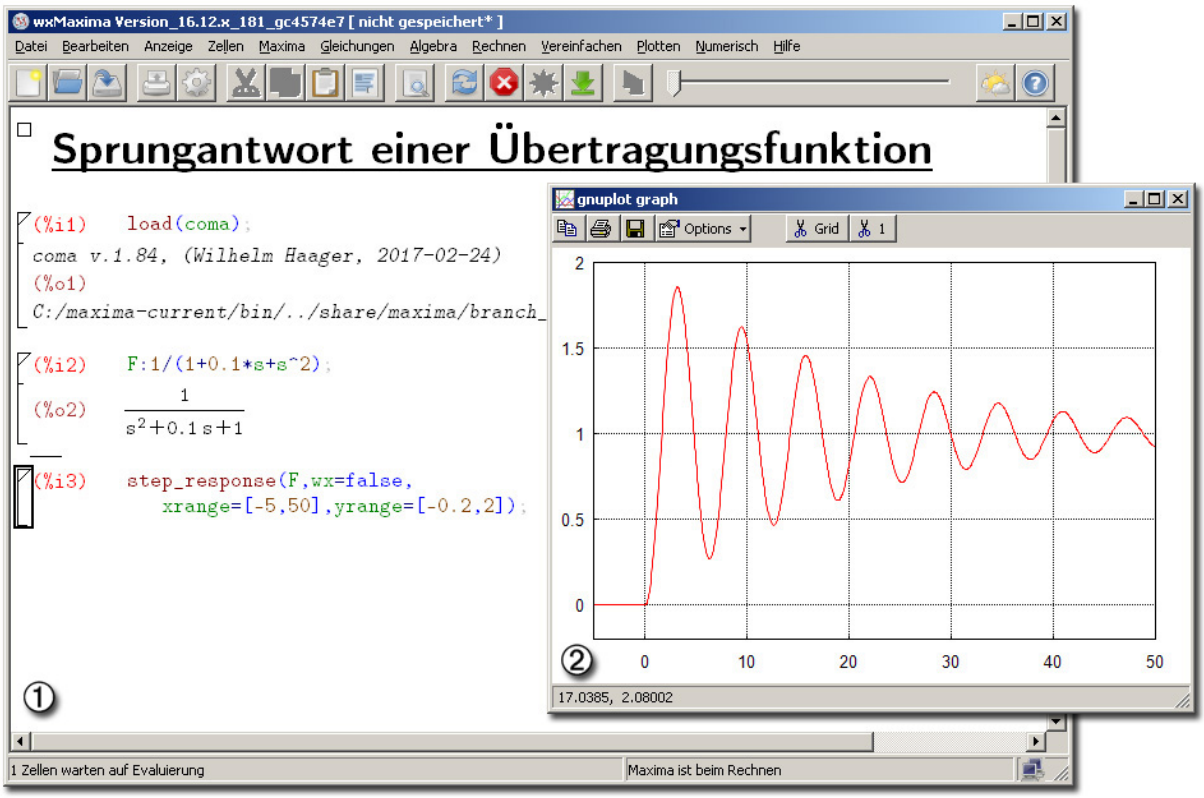
On installing Maxima, wxMaxima is automatically installed as the standard user interface.

COMA (*COntrol engineering with MAxima*):

Control engineering package for Maxima, it comprises basic methods for system analysis in time-, frequency- and Laplace-domain, controller design, as well as state space methods:

- Inverse Laplace-transform of transfer functions of arbitrary order (the built-in function `ilt` in general fails at orders higher than two).
- Unit step responses
- Nyquist diagrams and Bode plots
- Poles and Zeros, root locus plots
- Stability investigations: stability limit, Hurwitz criterion, stable regions in the parameter plane, phase margin, gain margin
- Optimization and controller design: ISE-criterion (integral of squared error), gain optimum
- State space: Conversion into a transfer function, canonical forms, controllability, observability

1.2 wxMaxima User Interface



① . . . Working window for input and output
② . . . Gnuplot output window

1.3 Basic Concepts of the Package COMA

- The Laplace variable is considered to be always s, the time variable always t; in functions concerning the frequency response, the angular frequency is ω. In frequency responses s is automatically replaced by $j\omega$. Transfer functions are rational functions in the variable s, time delays are not supported, but can be approximated by Padé approximations.
- All functions, which take a transfer function as a parameter, can also take (without explicit notice) a *list* of transfer functions as a parameter; in that case the result will also be a list, of which the elements correspond to the particular transfer functions. That is particularly important in graphics, where a couple of curves shall be drawn into a single diagram.

  The list to be plotted need not have only *functions* (transfer functions), but can also contain graphic objects of the Gnuplot interface *Draw* (`explicit`, `points`, `implicit`, `parametric`, `polar`, `polygon`, `rectangle`, `ellipse`, `label`). Thus diagrams can be provided with labels, legends and other graphical elements; furthermore a direct comparison with measured values is possible.
- Additional to the plotted functions, all plot routines can have optional parameters in the form *option* = *value*, which allow to adapt the graphic with respect to colors, line widths, scale, graphic type, output etc.
- Priot to its first usage, the package has to be laded using the command `load`.

2 Plot routines

Maxima uses the program *Gnuplot* [2] for drawing graphics, which is called at the generation of the graphic. Herein the graphic is drawn either in a seperate Gnuplot window (using the option `wx=false`) or directly into the working window of wxMaxima (using the option `wx=true`, default).

The plot routines of *COMA* don't use the standard functions of Maxima (`plot2d`, `plot3d`, `wxplot2d`, `wxplot3d`), but the functions of the additional package *Draw* (`draw2d`, `draw3d`, `wxdraw2d` und `wxdraw3d`), refer [1], [3]. Those functions are a little bit more complicated in their application, but they offer much more possibilities to adapt the graphics according to special requirements by the use of options.

All plot routines take a single function as parameter or a *list* of functions; additional optional parameters in the form *option*=*value*. Options, which apply to particular graphic objects, (`color`, `line_width etc.`), can be given in a list, of which the elements correspond to the respective graphic objects: *option*=[val_1,val_2, ...].

2.1 Options of the Gnuplot-Interface Draw

`terminal=`*target*	output target, possible values: `screen` (default), `jpg`, `png`, `pngcairo`, `eps`, `eps_color`
`file_name=`*string*	name of the output file, default: `maxima_out`.*ext*
`color=`*c*	plot color
`line_width=`*w*	line width
`xrange=`[x_1,x_2]	plot range in x-direction
`yrange=`[x_1,y_2]	plot range in y-direction
`zrange=`[z_1,z_2]	plot range in z-direction
`logx=true/false`	logarithmic scale of the x-axis
`logy=true/false`	logarithmic scale of the y-axis
`logz=true/false`	logarithmic scale of the z-axis
`grid=true/false`	inclusion of grid lines
`enhanced3d=true/false`	coloring of surfaces in 3D-plots

Important options of the Gnuplot-interface Draw

A complete list of the options can be found in in the Maxima manual [1].

2.2 Additional Options of COMA

`wx=true/false`	determines the output: `true` ... output into the wxMaxima working window `false` ... output into a seperate Gnuplot window
`aspect_ratio=`*value*	ratio height/width of the diagram; the value −1 results in same scale of the x-axis and the y-axis
`color=[`c_1`,`c_2`,...]`	list of colors, which are applied for the particular elements to be plotted respectively
`line_width=[`w_1`,`w_2`,...]`	list of linewidths, applied to the elements respectively
`dimensions=[`*width*`,`*height*`]`	width and height of the graphic
⋮	
Global variable:	
`coma_defaults`	list containing options in the form *option=value*

Additional options of COMA

The variable `coma_defaults` is a list containing default values for settings in the form of key-value pairs. Contrary to the list `draw_defaults` of the Gnuplot-interface *Draw* `coma_defaults` can contain also othe options, which are *not* part of *Draw*.

2.3 Plot

The function `plot` performs a two-dimensional depiction of functions $f(x)$ in one variable or a three-dimensional depiction of functions $f(x, y)$ in two variables.

`plot(`*f(x)*, *opts*`)`	plotting the function $f(x)$ in a two-dimensional coordinate system
`plot(`*f(x,y)*, *opts*`)`	plotting the function $f(x, y)$ in 3D-representation
	Instead of a single function f, also a *list* of functions `[`f_1`, `f_2`, ...]` can be plotted.

Plot routines for 2D und 3D graphics

The functions of the package *Draw* (`wxdraw2d`, `draw2d`, `wxdraw3d`, `draw3d`) are called internally with appropriate parameters. Thus the (convenient) call of

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

exactly corresponds to the (less convenient) command

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

The value of the option `aspect_ratio`, which does not exist in the routines of the package *Draw*, is passed to Gnuplot via the option `user_preamble` in appropriate form.

Loading the control engineering package

```
(%i1) load(coma)$
coma v.1.84, (Wilhelm Haager, 2017-02-24)
```

List containing default values for the graphics options

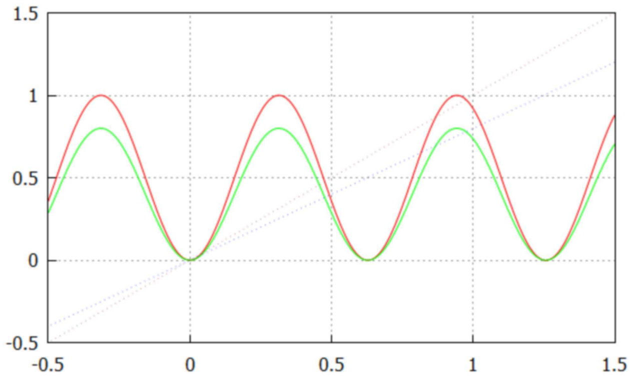
```
(%i2) coma_defaults ;
(%o2) [grid=true , wx=true , dimensions =[600 ,350] ,
user_preamble =set grid linetype 3 lc '#444444' ,
color=[red , blue , dark-green , goldenrod , violet ,
gray40 , dark-cyan , orange , brown , sea-green ]]
```

List of color names

```
(%i3) col:[red,green,brown,blue];
(%o3) [red , green , brown , blue ]
```

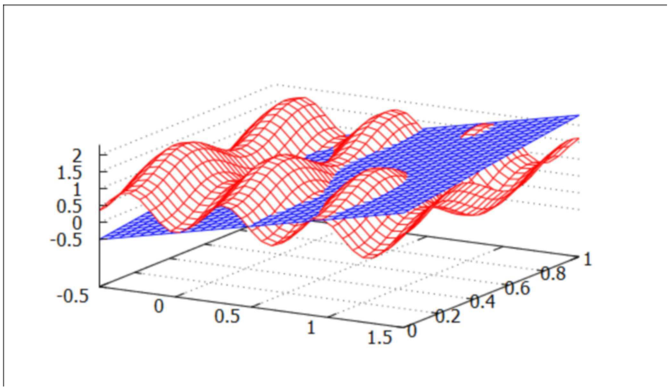
Plotting a list of four functions in *one single* variable; internally the function `wxdraw2d` is called. The names of the variables can be different in each function.

```
(%i4) 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]);
(%t4)
```



Plotting a list of two functions in *two* variables each; internally the function `wxdraw3d` is called. The option `surface_hide=true` suppresses hidden lines.

```
(%i5) plot([sin(5*x)**2+0.8*sin(5*y)**2,x+0.8*y],
      xrange=[-0.5,1.5],surface_hide =true)$
(%o4)
(%t5)
```



`plot` evaluates the function to be plotted f *before* points are calculated. In order to evaluate the function for every particular point, it has to be *quoted*. That is especially important e.g. for characteristic values of transfer functions (section 7), which can only be calculated numerically.

Transfer function with a dependence on a parameter a

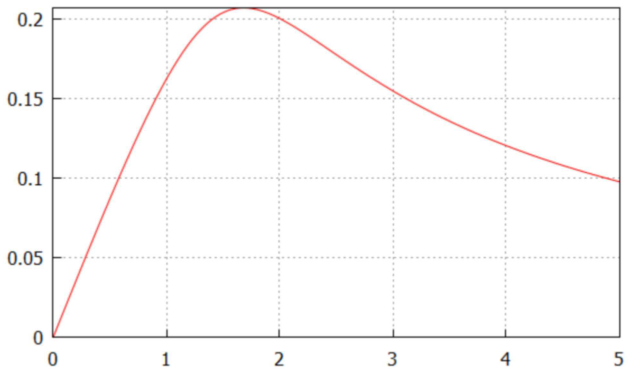
```
(%i6) f:(s+a)/(s^3+a*s^2+2*s+a);
```

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

The damping can only be calculated numerically, which requires a to have a fixed value. Thus f has to be quoted to avoid being evaluated too early.

(%i7) `plot('damping_ratio(f),xrange=[0,5]);`

(%t7)



2.4 Contour Lines

The function `contourplot` draws isolines of a function $f(x, y)$. Contrary to `contour_plot`, which is part of Maxima, it uses the Gnuplot interface *Draw*, like all plot routines of the package *COMA*. It also has the same options.

`contourplot`(*f(x,y)*, x, y, *opts*)	Plotting of isolines of the function $f(x, y)$
`contours=[`$z_1, z_2, \ldots$`]`	Determining the function values for the isolines

Contour Lines

Transfer function with two parameters a and b

(%i8) `f:1/(s^5+s^4+5*s^3+a*s^2+b*s+1);`

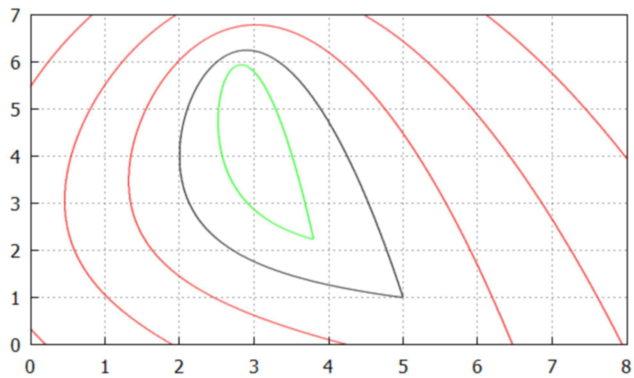
(%o7)

(%o8) $$\frac{1}{s^5+s^4+5\,s^3+a\,s^2+b\,s+1}$$

Isolines for the damping in dependence on the parameters a and b. The black line at damping 0 represents the stability limit, green lines represent stable areas, red lines unstable areas.

(%i9)
```
contourplot('damping(f),a,b,xrange=[0,8],
yrange=[0,7],contours=[-0.3,-0.2,-0.1,0,0.1],
color=[red,red,red,black,green],
ip_grid_in=[20,20])$
```

(%t9)



3 Transfer functions

COMA provides the following functions for convenient generation of transfer functions, primarily for testing and experimenting:

`rantranf(`n`)`	n-th order random transfer function, of wich the numerator and denominator coefficients are numbers between 1 and 10
`stable_rantranf(`n`)`	Stable random transfer function (only up to 6th order)
`gentranf(`c, k, d, n`)`	n-th order transfer function, (numerator of k-th order) with the numerator coefficients c_i and the denominator coefficients d_i
`tranftype(`$F(s)$`)`	Type of the transfer function $F(s)$ as a string
`ntranfp(`$F(s)$`)`	Yields `true`, if all coefficients of the transfer function $F(s)$ evaluate to numbers.
`closed_loop(`$Fo(s)$`)`	Calculation of the closed loop transfer function $F_W(s)$ from the open loop $F_O(s)$
`open_loop(`$Fw(s)$`)`	Calculation of the open loop transfer function $F_O(s)$ from the closed loop $F_W(s)$
`time_delay(`$T, n,$ `[`k`])`	n-th order Padé approximation for a time delay system. The order of the numerator k is optional.
`impedance_chain(`$Z_1, Z_2, \ldots [n]$`)`	transfer function of an impedance chain with the impedances Z_1, Z_2, ... and an (optional) repeat factor n
`transfer_function(`$eqs, vars, u, y$`)`	Calculation of the transfer function from the equations *eqs* in the variables *vars* with the inputs u and the outputs y
`sum_form(`$F(s), n$`)`	One of four canonical forms *F(s)* (in depoendence of n)
`product_form(`$F(s)$, `[`n`])`	Splitting up *F(s)* into linear und quadratic factors

Generation of transfer cunctions

The function `stable_rantranf(n)` searches denominator coefficients randomly between 1 and 10, until a *stable* transfer function is found, which is becoming more difficult at higher orders, at seventh order computing time is increasing heavily. Thus `stable_rantranf` is working only for transfer functions up to sixth order.

Higher orders can be attained by multiplication of several lower order transfer functions. However, in that case the coefficients are not confined to the range 1...10 any more.

Generation of a list of fourth order random transfer functions, the orders on the numerators are lower, at least by one.

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

(%o1) $$\left[\frac{4s^2+8s+10}{4s^3+s^2+4s+10},\frac{9s+10}{7s^3+7s^2+2s+4},\frac{2s^2+10s+7}{4s^3+5s^2+9s+7},\frac{2s^2+s+10}{10s^3+10s^2+2s+5}\right]$$

Stability test of the transfer functions (section 7)

(%i2) `stablep(fli);`

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

Generation of a list of stable random transfer functions

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

(%o3) $$\left[\frac{3}{2s^3+10s^2+7s+4},\frac{4s^2+10s+1}{2s^3+3s^2+6s+4},\frac{8s+7}{2s^3+9s^2+s+1},\frac{10}{4s^3+10s^2+4s+5}\right]$$

All are stable.

(%i4) `stablep(fli);`

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

List of transfer functions

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

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

Calculation of the closed loop transfer functions

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

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

Determining the types of the transfer functions as strings

(%i7) `tranftype(fw);`

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

Check, whether all coefficients of the transfer functions evaluate to numbers

(%i8) `ntranfp(fw);`

(%o8) [false, true, false]

Back-calculation to the open loop transfer functions

(%i9) `open_loop(fw);`

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

`gentranf(a,k,b,n)` produces a general transfer function with indexed coefficients in the form

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

transfer function with general coefficients a_i and b_i

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

(%o10) $$\frac{a_3s^3+a_2s^2+a_1s+a_0}{b_5s^5+b_4s^4+b_3s^3+b_2s^2+b_1s+b_0}$$

Time delay systems have transcendental transfer functions, inverse Laplace transform of control loops containing time delays is not possible analytically in general.

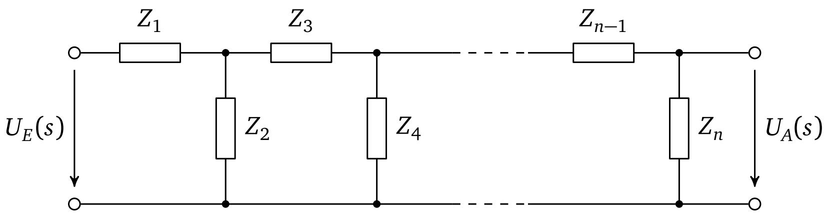
`time_delay(T,n,k)` yields a n-th order Padé approximation of a time delay system with the transfer function $G(s)=e^{-sT}$. The declaration k of the numerator order is optional, its default is $n-1$.

Fourth order Padé approximation of the transfer function of a time delay $G(s)=\exp(-sT)$

(%i11) `time_delay (T,4);`

(%o11) $$\frac{-4\,T^3\,s^3+60\,T^2\,s^2-360\,T\,s+840}{T^4\,s^4+16\,T^3\,s^3+120\,T^2\,s^2+480\,T\,s+840}$$

`impedance_chain` calculates the transfer function of an impdance chain with arbitrary impedances (of even number); the last (optional integer valued) parameter determines the number of repetitions of the impedance chain.



transfer function of an impedance chain consisting of four elements

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

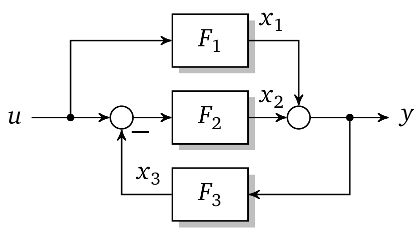
(%o12) $$\frac{1}{C^2\,L\,R\,s^3+\left(C^2\,R^2+C\,L\right)\,s^2+3\,C\,R\,s+1}$$

transfer function of an impedance chain repeated four times

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

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

`transfer_function`(*eqs*,*vars*,*u*,*y*) calculates the transfer function from a list of linear equations *eqs*, e.g. from a block diagram. *vars* is a list containing the used variables according to which the system of equations is to be solved, *u* is the input, *y* is the output.
Also multivariable systems can be calculated: When *u* and *y* are *lists* of variables, the corresponding transfer *matrix* is calculated.



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

System of linear equations from the block diagram

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

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

Calculation of the transfer function from the equations

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

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

`sum_form`(*F(s)*,*n*) divides the numerator and the denominator of *F(s)* in dependence of *n* by a partitcular coefficient, thus making one of the leading or last coefficients to 1:

$n=1$ … leading numerator coefficient of $F(s)$
$n=2$ … last numerator coefficient of $F(s)$
$n=3$ … leading denominator coefficient of $F(s)$
$n=4$ … last denominator coefficient of $F(s)$ (default)

(%i16) `F:(2*s+3)/(4*s^3+5*s^2+6*s+7);`

(%o16) $$\frac{2\,s+3}{4\,s^3+5\,s^2+6\,s+7}$$

Canonical forms, making the leading or last *numerator* coefficient to 1:

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

(%o17) $$\left[\frac{s+1.5}{2\,s^3+2.5\,s^2+3\,s+3.5},\frac{0.66667\,s+1}{1.3333\,s^3+1.6667\,s^2+2\,s+2.3333}\right]$$

Canonical forms, making the leading or last *denominator* coefficient to 1:

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

(%o18) $$\left[\frac{0.5\,s+0.75}{s^3+1.25\,s^2+1.5\,s+1.75},\frac{0.28571\,s+0.42857}{0.57143\,s^3+0.71429\,s^2+0.85714\,s+1}\right]$$

`product_form` splits numerator and denominator of the transfer function into linear and quadratic factors:

Product form of a transfer function

(%i19) `product_form(F);`

(%o19) $$\frac{0.42857\,\left(0.66667\,s+1.0\right)}{\left(0.82799\,s+1.0\right)\,\left(0.69014\,s^2+0.029158\,s+1.0\right)}$$

4 Laplace Transformation, Step Response

Maxima provides the function `laplace(f,t,s)` for the Laplace transform; the inverse Laplace transform is calculated with `ilt(f,s,t)`. The coefficients of the numerator and denominator polynomials can have symbolic values. However, `ilt` fails at denominator polynomials of third or higher order, if no zeros can be found analytically.

The function `nilt` of the package *COMA* calculates the zeros of the denominator polynomial *numerically* using `allroots`, thus rational functions of (nearly) arbitrary order can be back-transformed; however, the polynomial coefficients have to evaluate to numbers in that case.

`laplace`(*ft*, *timevar*, *lapvar*)	Laplace transform of the function *ft* (part of Maxima)
`ilt`(*fs*, *lapvar*, *timevar*)	inverse Laplace transform of *fs* (part of Maxima)
`nilt`(*fs*, *lapvar*, *timevar*)	inverse Laplace transform of *fs* with numerically calculated poles
`step_response`(*F*(*s*), *opts*)	Plotting the unit step response of the transfer function *F*(*s*)

Laplace transform

Laplace transform of a function

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

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

Inverse Laplace transform

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

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

The coefficients can also have *symbolic* values.

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

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

Inverse Laplace transform fails, if no zeros of the denominator polynomial can be found analytically.

```
(%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` calculates the zeros of the denominator numerically, thus arbitrary order transfer functions can be back-transformed.

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

(%o5) $-0.14795\ \%e^{-0.78492\,t}\sin\left(1.3071\,t\right)-0.54512\ \%e^{-0.78492\,t}\cos\left(1.3071\,t\right)+0.54512\ \%e^{-0.43016\,t}$

Generation of a list of PT2-elements with increasing damping ratio

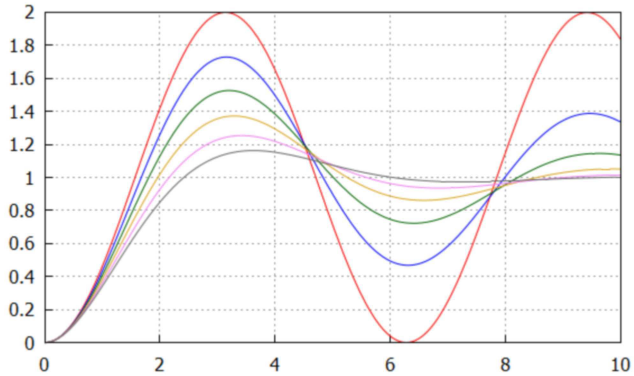
(%i6)
```
pt2li:create_list(1/(s^2+2*d*s+1), d,
    [0.0001,0.1,0.2,0.3,0.4,0.5]);
```

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

Plotting the step responses; unless the option `xrange` is given explicitely, the time range is chosen automatically.

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

(%t7)



PT1-element with additional time delay in Padé approximation

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

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

Calculation of the exact step response of a PT1-element with additional time delay

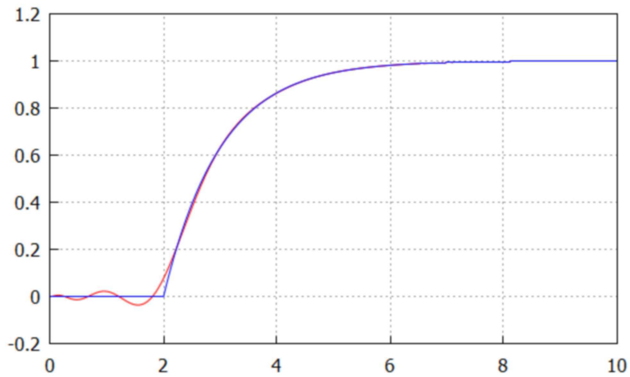
(%i9)
```
ft:unit_step(t-2)*ev(ilt(1/(s*(1+s)),s,t),
t=t-2);
```

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

Comparison of the exact step response with the Padé approximation; the exact step response is included as a graphical object `explicit`.

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

(%t10)



5 Frequency Responses

`bode_plot`(*F(s)*, *opts*)	Bode plot of $F(j\omega)$
`magnitude_plot`(*F(s)*, *opts*)	Magnitude plot of the Bode diagram of $F(j\omega)$
`logy=false`	Option for `magnitude_plot`, yields *linear* scale of the magnitude
`phase_plot`(*F(s)*, *opts*)	Phase plot of the Bode diagram of $F(j\omega)$
`phase`(*F(s)*)	Phase shift of the frequency responsees $F(j\omega)$ in degree
`asymptotic`(*F(s)*)	Asymptotic characteristic of the frequency response $F(j\omega)$
`nyquist_plot`(*F(s)*, *opts*)	Frequency response locus of $F(j\omega)$

Frequency responses

Parameters are transfer functions $F(s)$ depending on the Laplace variable s; the plot routines replace s by $j\omega$ automatically.

Unless the options `xrange` and `yrange` are declared explicitely, the scale is chosen automatically. The axes of Bode plots plot can be linearly-scaled using the option `logx=false` and `logy=false`. `bode_plot` requires a *list of two ranges* for the option `yrange`, one for the magnitude plot and one for the phase plot each.

Frequency response locus plots (`nyquist_plot`) have the same scale in x-direction and y-direction by default (`aspect_ratio=-1`), which results in an undistorted image.

Contrary to the Maxima function `carg`, which calculates the argument of a complex number (in radiant) always in the interval $-\pi \dots \pi$, `phase` calculates the actual phase shift between input and output, which can attain arbitrarily high values; every pole and every zero produce a phase shift of $\pi/2$ (or 90 degrees) with the appropriate sign.

At resonance a significant phase shift is occuring in a small frequency range. In order to attain – especially in frequency response locus plots – a smooth curve, the number of primarily calculated points has to be increased explicitely, which can be set with the option `ntricks=`*value* (default 500).

List of PT2-elements with increasing daming ratio

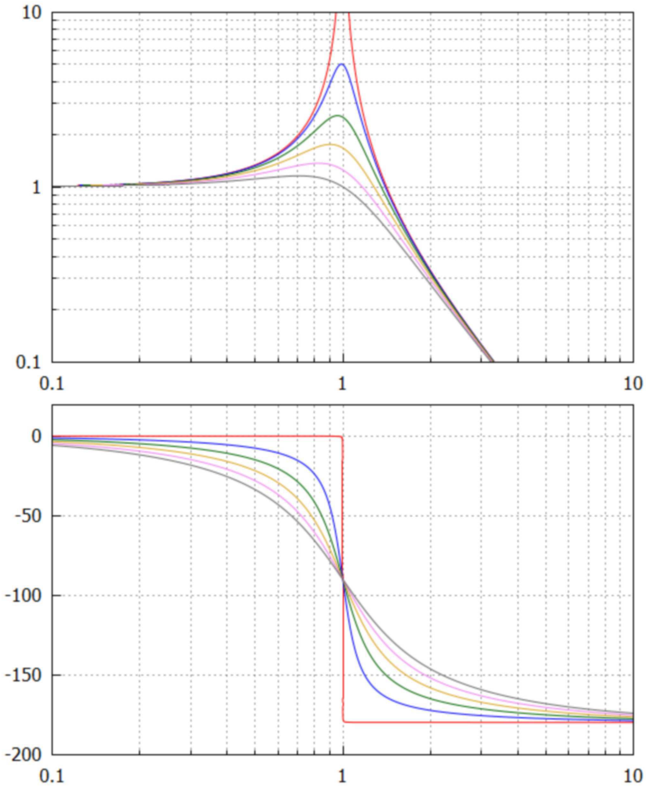
```
(%i1) fli:create_list (1/(s^2+2*d*s+1), d,
          [0.0001,0.1,0.2,0.3,0.4,0.5]);
```

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

Bode plots of the PT2-elements, the ranges of the y-axes have to be declared in a *list*.

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

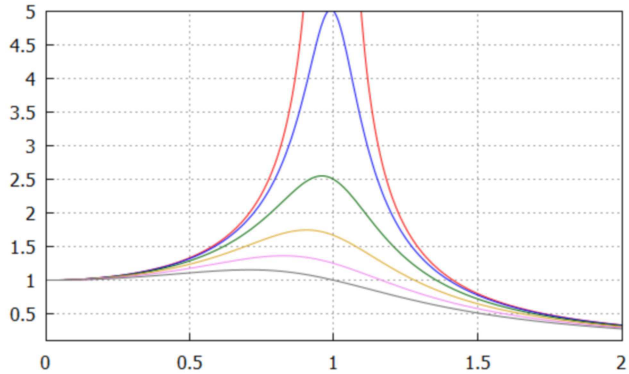
(%t2)



Magnitude plots of the PT2-elements with both axes scaled linearly

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

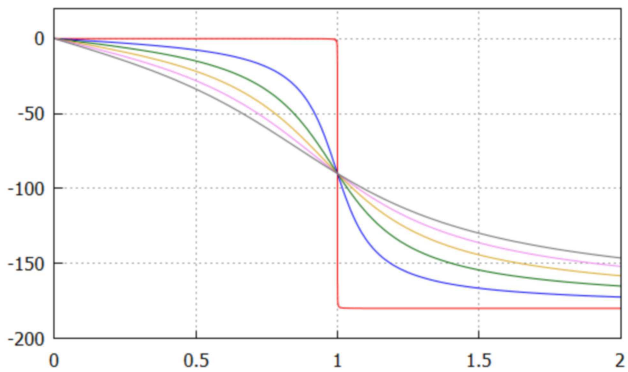
(%t3)



Phase plots of the PT2-elements with both axes scaled linearly; the y-axis is scaled linearly by default.

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

(%t4)



Transfer function of a PID-controller

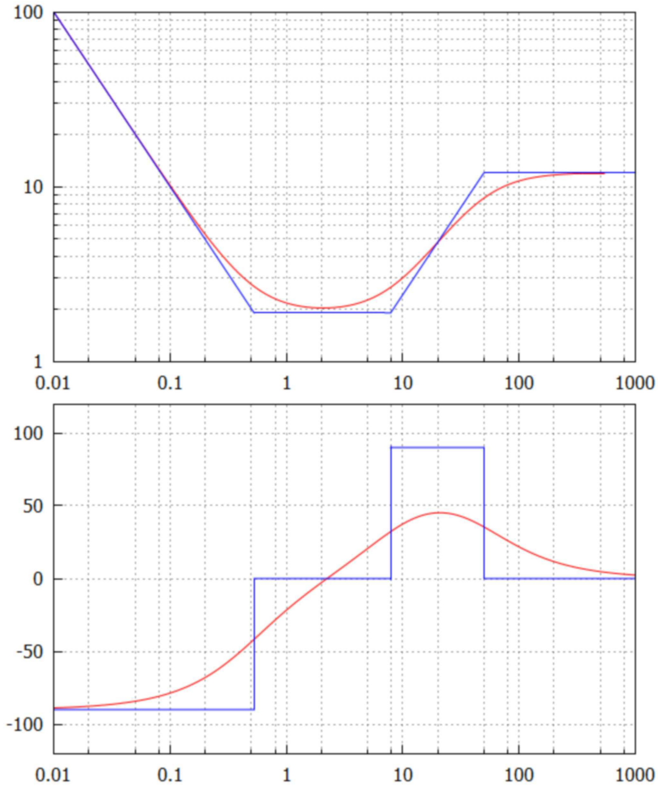
(%i5)
```
Fr:2*(1+1/(2*s)+0.1*s/(1+0.02*s));
```

(%o5) $2\left(\frac{0.1\,s}{0.02\,s+1}+\frac{1}{2\,s}+1\right)$

Bode plot of the PID-controller: exact (red) and asymptotic characteristic (blue)

(%i6)
```
bode_plot ([Fr,asymptotic (Fr)],
xrange=[0.01,1000],yrange=[[1,100],[-120,120]])$
```

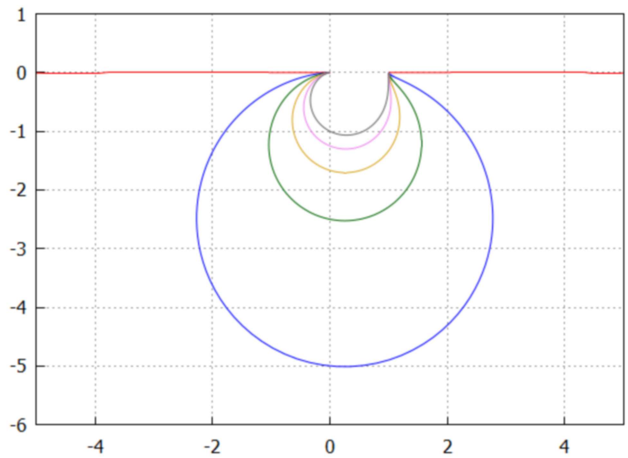
(%t6)



Frequency response locus plots of the PT2-elements; eventually the number of primarily calculated points has to be increased with the option `nticks`.

(%i7)
```
nyquist_plot (fli,xrange=[-5,5],yrange=[-6,1],
    nticks=2000)$
```

(%t7)



Third order transfer function

(%i8)
```
f:2/(1+2*s+2*s^2+s^3);
```

(%o8) $\frac{2}{s^3+2\,s^2+2\,s+1}$

Phase shift of $F(j\omega)$ by splitting the frequency response into linear and quadratic factors and addition of the partial phase shifts.

```
(%i9) phase(f);
```

(%o9) $$\frac{180\left(-\mathrm{atan}\left(1.0\,\omega\right)-\mathrm{atan2}\left(1.0\,\omega,1.0-1.0\,\omega^2\right)\right)}{\pi}$$

A frequency response locus can be marked and labelled with graphical objects `points` and `label`; arbitrary positions of the labels can be determined with some tricky considerations: the label for a point lies in a certain distance from the point in orthogonal direction of the curve. Attention: points and vectors are defined here as *lists*, not as *matrices*.

List of ω-values for marking and labelling of the frequency response locus

```
(%i10) omegali:makelist(0.1*k,k,1,10);
(%o10) [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0
]
```

Replacement of s by $j\omega$

```
(%i11) fom:ev(f,s=%i*omega);
```

(%o11) $$\frac{2}{-\%i\,\omega^3-2\,\omega^2+2\,\%i\,\omega+1}$$

Position of a point on the curve (list of x- and y-coordinate):

```
(%i12) dot:[realpart(fom),imagpart(fom)];
```

(%o12) $$\left[\frac{2\left(1-2\,\omega^2\right)}{\left(2\,\omega-\omega^3\right)^2+\left(1-2\,\omega^2\right)^2},\frac{2\left(\omega^3-2\,\omega\right)}{\left(2\,\omega-\omega^3\right)^2+\left(1-2\,\omega^2\right)^2}\right]$$

Differentiation gives the direction of the curve.

```
(%i13) abl:ratsimp(diff(dot,omega));
```

(%o13) $$\left[\frac{16\,\omega^7-12\,\omega^5-8\,\omega}{\omega^{12}+2\,\omega^6+1},-\frac{6\,\omega^8-20\,\omega^6-6\,\omega^2+4}{\omega^{12}+2\,\omega^6+1}\right]$$

Unit vector orthogonal to the curve

```
(%i14) ovec:ratsimp([-abl[2],abl[1]]/
              sqrt(abl[1]^2+abl[2]^2));
```

(%o14) $$\left[\frac{6\,\omega^8-20\,\omega^6-6\,\omega^2+4}{\sqrt{36\,\omega^4+16\,\omega^2+16}\,\sqrt{\omega^{12}+2\,\omega^6+1}},\frac{16\,\omega^7-12\,\omega^5-8\,\omega}{\sqrt{36\,\omega^4+16\,\omega^2+16}\,\sqrt{\omega^{12}+2\,\omega^6+1}}\right]$$

Position for the label of a point

```
(%i15) lab:dot+0.3*ovec$
```

The marks are defined as graphic object `points`.

```
(%i16) punkte:points(map(lambda([u],ev(dot,omega=u)),
       omegali))$
```

The labels are defined as graphic object `label`.

```
(%i17) labs:apply(label,map(lambda([u],[string(u),
       ev(lab[1],omega=u),ev(lab[2],omega=u)]),
       omegali))$
```

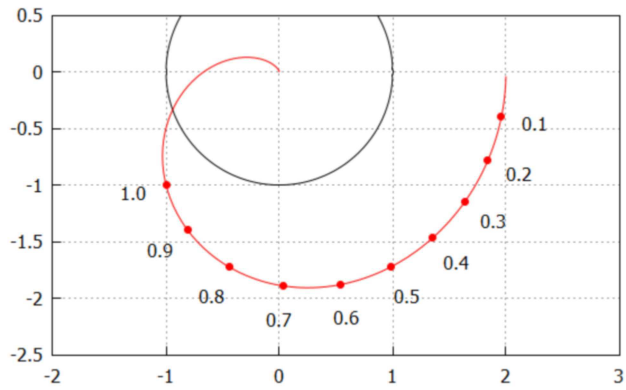
Unit circle as parametric curve

```
(%i18) circle:parametric(cos(t),sin(t),t,0,2*%pi);
```

(%o18) $\mathrm{parametric}\left(\cos\left(t\right),\sin\left(t\right),t,0,2\,\pi\right)$

Frequency response locus plot with labelled points and the unit circle

```
(%i19) nyquist_plot([circle,f,punkte,labs],
       xrange=[-2,3],yrange=[-2.5,0.5],
       point_type=7,color=[black,red,red,black])$
(%t19)
```



6 Investigations in the Complex s-Plane

6.1 Poles/Zeros-Distribution

`poles` $(F(s))$	Poles of the transfer function $F(s)$
`zeros` $(F(s))$	Zeros of the transfer function $F(s)$
`poles_and_zeros` $(F(s), opts)$	Image of the Poles/zeros-distribution of the transfer function $F(s)$ in the complex s-plane

Poles/Zeros-Distribution

The function `poles` and `zeros` return the poles and zeros of the transfer function in al list. `poles_and_zeros` draws the poles/zeros distribution in the complex s-plane. Herein a pole is indicated by a × mark, a zero by a ∘ mark. In order to attain an undistorted image, the scales in x-direction any y-direction are the same by default (`aspect_ratio=-1`).

List of random transfer functions

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

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

Zeros

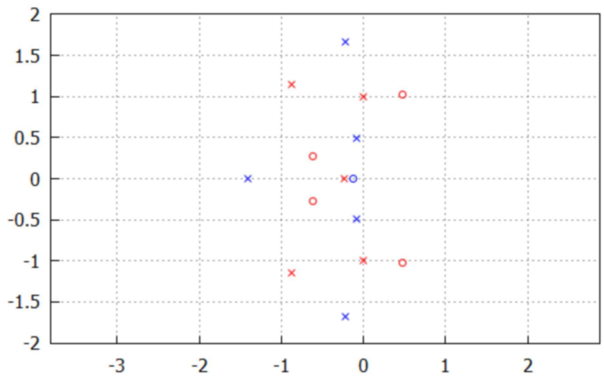
```
(%i2) zeros(fli);
(%o2) [[0.2688 %i-0.61289, -0.2688 %i-0.61289,
0.47003-1.0271 %i, 1.0271 %i+0.47003], [-0.125]]
```

Poles

```
(%i3) poles(fli);
(%o3) [[-0.2408, 1.0 %i, -1.0 %i, -1.1414 %i-0.8796,
1.1414 %i-0.8796], [0.49432 %i-0.079394, -0.49432 %i-
0.079394, -1.6717 %i-0.21887, -1.4035, 1.6717 %i-
0.21887]]
```

Pole/zero distribution in the complex s-plane

```
(%i4) poles_and_zeros(fli)$
(%t4)
```



6.2 Root Locus Plots

root_locus(*F(s,k)*,*opts*)	Root locus plot of a transfer function $F(s,k)$ with one free parameter k in the s-plane
trange=[*min*,*max*]	Range for the free parameter k, default: [0.001,100]
nticks=*n*	Number of calculated points, default: 500

Root locus plots

root_locus draws the root locus of a transfer function $F(s)$ in dependence of a parameter k, which need not be (unlike in "classical" root loci) the open loop gain, but can be an arbitrary parameter influencing the transfer function. If several transfer functions are given in a list, the names of the parameters can be different, nervertheless their *ranges* have to be the same for all transfer functions, determinded by the option trange.

The plot points are distributed over the parameter range logarithmically, thus only positive values for the range are allowed.

The starting points of the root loci are indicated by a × mark, the endpoints are indicated by a ∘ mark. If the free parameter is the open loop gain, its starting value is sufficiently small, its end value is sufficiently large, starting and ending points represent the poles and zeros of the open loop transfer function respectively.

List of transfer functions with various zeros a and a variating gain k

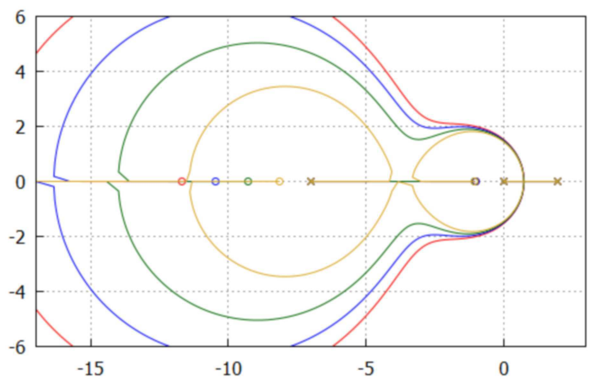
```
(%i5) fli:closed_loop(makelist(k*((s-a)*(s+1))
        /(s*(s-2)*(s+7)),a,-11,-8));
```

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

Root locus plots in dependence on the open loop gain k with various values of the open loop zero a

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

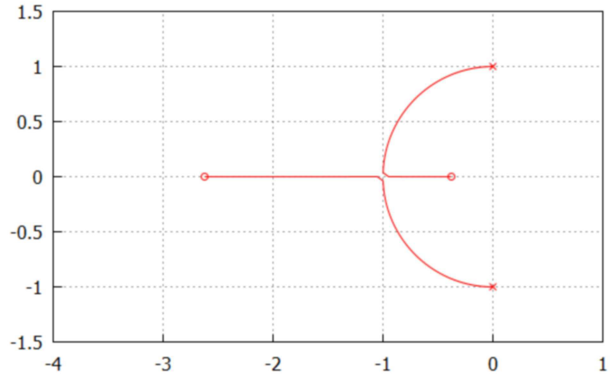
(%t6)



Root locus plot of a PT2-element with the damping ratio as parameter

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

(%t7)



7 Stability Behavior

7.1 Stability

`stablep(`*F(s)*`)`	Checks the stability of the system with the der transfer function $F(s)$
`stability_limit(`*F(s), k*`)`	Calculation of the stability limit of the transfer function $F(s,k)$ with respect to the parameter k
`hurwitz(`*p(s)*`)`	Calculation of the Hurwitz-determinants of the polynomial $p(s)$
`stable_area(`*F(s, a, b), a, b, opts*`)`	Plot of the stability limit of the transfer function $F(s,a,b)$ in the a/b-parameter plane

Stability

The function `stability_limit(`*F(s), k*`)` yields conditions for imaginary poles in the form

$$k = value,\ \omega = value,$$

which is equivalent to the stability limit for common systems. Herein the value of ω is the angular frequency of the undamped oscillaion at the stability limit. Those conditions can be fulfilled also for *more than one* value of ω. Which condition in fact corresponds to the stability limit, has to be checked by further considerations.

Exact conditions for stability are provided by the *Hurwitz-criterion*: All zeros of the polynomial $p(s)$ have a negative realpart (i. e. in exactly that case the transfer function with the denominator $p(s)$ is stable), if all Hurwitz determinants have a value greater zero. The function `hurwitz(`*p(s)*`)` yields a list of the Hurwitz determinants, the coefficients of $p(s)$ can have symbolic values.

`stable_area` plots the stability limit of a transfer function with respect to two parameters a and b in the a/b-parameter plane. Unless the options `xrange` and `yrange` are given explicitely, the axes range from 0 to 1.

Random transfer function

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

(%o1) $\frac{9}{s^5+3s^4+9s^3+10s^2+7s+6}$

Calculation of the closed loop transfer function with a controller gain of k

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

(%o2) $\frac{9k}{s^5+3s^4+9s^3+10s^2+7s+9k+6}$

Calculation of the stability limit; the result can be more tan one condition for imaginary poles.

```
(%i3) lim:stability_limit(fw,k),numer;
(%o3) [[k=-13.709,ω=2.8531],[k=0.042326,ω=
0.92733]]
```

The Hurwitz criterion provides exact results; the system is stable if and only if all elements of the resulting list are positive.

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

(%o4) $[16-81\,k, 39-666\,k, -81\,k^2-1107\,k+47, -81\,k^2-1107\,k+47]$

Generation of a list of gains: above, at and below the stability limit

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

(%o5) [0.046559 , 0.042326 , 0.038093]

Calculation of the transfer functions of the corresponding *open* loops,

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

(%o6) $[\frac{0.41903}{s^5+3\,s^4+9\,s^3+10\,s^2+7\,s+6}, \frac{0.38093}{s^5+3\,s^4+9\,s^3+10\,s^2+7\,s+6}, \frac{0.34284}{s^5+3\,s^4+9\,s^3+10\,s^2+7\,s+6}]$

as well as of the *closed* loops.

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

Checking the stability, `stablep` yields `true` in the limit case.

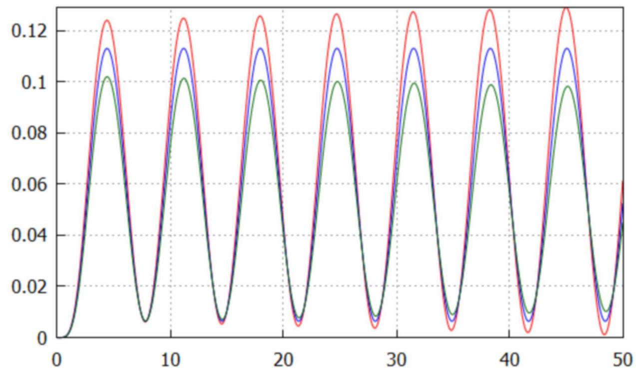
```
(%i8) stablep(fwli);
```

(%o8) [false , true , true]

Step responses; the period of the undamped oscillation at the stability limit according to $T = 2\pi/\omega$ yields a value of about 6.7.

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

(%t9)



List of PT5-elements with two free parameters a and b

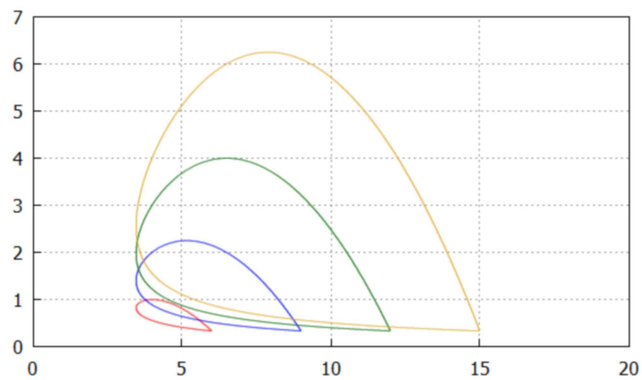
```
(%i10) fli:makelist(1/(s^5+3*s^4+k*s^3+a*s^2+b*s+1)
       ,k,2,5);
```

(%o10) $[\frac{1}{s^5+3\,s^4+2\,s^3+a\,s^2+b\,s+1}, \frac{1}{s^5+3\,s^4+3\,s^3+a\,s^2+b\,s+1}, \frac{1}{s^5+3\,s^4+4\,s^3+a\,s^2+b\,s+1}, \frac{1}{s^5+3\,s^4+5\,s^3+a\,s^2+b\,s+1}]$

Plotting the borders of the stable areas in the a/b-parameter plane

```
(%i11) stable_area(fli,a,b,xrange=[0,20],
       yrange=[0,7],ip_grid_in=[20,20])$
```

(%t11)



7.2 Relative Stability

The relative stability can be validated with the phase margin α_R or the gain margin A_R. The corresponding angular frequencies are the gain crossover frequency ω_D and the phase crossover frequency ω_r respectively.

gain_crossover($F(s)$)	Calculation of the gain crossover frequencies ω_D of $F(j\omega)$ for which holds $\lvert F(j\omega_D)\rvert = 1$
phase_margin($F(s)$)	Calculation of the phase margin α_R of $F(j\omega)$ in degree
phase_crossover($F(s)$)	Calculation of the phase crossover frequency ω_r of $F(j\omega)$ for which holds $\arg F(j\omega_r) = -\pi$
gain_margin($F(s)$)	Calculation of the gain margin A_R of $F(j\omega)$
damping($F(s)$)	(absolute) damping of $F(j\omega)$ (negative realpart of the rightmost pole)
damping_ratio($F(s)$)	minimum damping ratio of all pole pairs of $F(j\omega)$

Relative Stability

gain_crossover yields a list containing the gain crossover frequencies ω_D, at which the absolut value of the frequency response is equal to 1. phase_margin yields the corresponding phase margins, the differences to $-180°$. Which of those values is actually significant for stability, has to be checked by further considerations (that can be nore than one values on principle).

Gain crossover frequencies ω_D; multiple values are possible (especially at occurence of resonance).

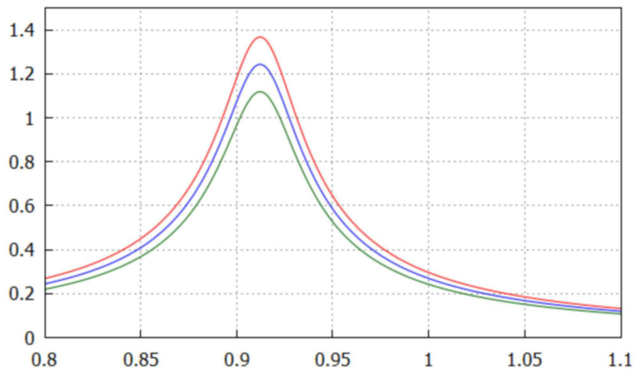
```
(%i12) gain_crossover (foli);
```

(%o12) $[[\omega=0.89261, \omega=0.93126], [\omega=0.89673, \omega=0.92733], [\omega=0.90173, \omega=0.92252]]$

Magnitude plots with more then one gain crossover frequencies

```
(%i13) magnitude_plot (foli,xrange=[0.8,1.1],
          yrange=[0,1.5],logx=false,logy=false)$
```

(%t13)



Phase margins α_R in degree, unstable control loops have a negative value.

```
(%i14) phase_margin (foli);
```

(%o14) $[[81.347, -6.7748], [74.56, -5.1512\ 10^{-5}], [64.484, 10.065]]$

Phase crossover frequencies ω_r, multiple values are possible.

```
(%i15) phase_crossover (foli);
```

(%o15) $[[\omega=\frac{\sqrt{9-\sqrt{53}}}{\sqrt{2}}], [\omega=\frac{\sqrt{9-\sqrt{53}}}{\sqrt{2}}], [\omega=\frac{\sqrt{9-\sqrt{53}}}{\sqrt{2}}]]$

Gain margins A_R, unstable control loops have a value less than 1.

```
(%i16) gain_margin(foli);
(%o16) [[0.90909],[1.0],[1.1111]]
```

`damping` calculates the *damping* σ of a transfer function. That is the negative realpart of the outmost right pole or pole pair. It indicates the speed of decaying (or stoking up) of a transient process. The *damping ratio* D is the relative damping, related to the natural angular frequency ω_n of a pole pair. `damping_ratio` calculates the minimum damping ratio of all pole pairs of a transfer function. Stable transfer functions have positive values of σ and D, unstable transfer functions have negative values

Damping of three transfer functions

```
(%i17) damping(fwli);
(%o17) [-0.0020036,0,0.0020153]
```

Damping ratio of three transfer functions

```
(%i18) damping_ratio(fwli);
(%o18) [-0.0021573,0.0,0.0021766]
```

8 Optimization

The performance index according to the ISE-criterion can be calculated according to Parseval's theorem directly in the Laplace domain:

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

Herein $e(t)$ is a function tending to zero with increasing time, usually the deviation of the controlled variable from its stationary value. According to [6] the integral can be calculated algebraically.

`ise(`$E(s)$`)`	Performance index of the function $e(t)$ according to the ISE-criterion

Integral performance indexes

Differentiation of the integral with respect to the free parameters (e. g. the controller parameters) and setting the results zo zero yield the optimum values of the parameters.

transfer function with two free parameters a and b	(%i1)	`f:1/(s**3+a*s**2+b*s+1);`
	(%o1)	$\frac{1}{s^3+a\,s^2+b\,s+1}$
Calculation of the deviation from the stationary value at input step function	(%i2)	`xs:ratsimp((1-f)/s);`
	(%o2)	$\frac{s^2+a\,s+b}{s^3+a\,s^2+b\,s+1}$
Preformance index according to the ISE-criterion	(%i3)	`iise:ise(xs);`
	(%o3)	$\frac{a\,b^2-b+a^2}{2\,a\,b-2}$
Differentiation with respect to the parameters a and b (Calculation of the Jacobian matrix)	(%i4)	`abl:ratsimp(jacobian([iise],[a,b]));`
	(%o4)	$\begin{bmatrix}\frac{a^2\,b-2\,a}{2\,a^2\,b^2-4\,a\,b+2} & \frac{a^2\,b^2-2\,a\,b-a^3+1}{2\,a^2\,b^2-4\,a\,b+2}\end{bmatrix}$
Confinement to real solutions of systems of equations	(%i5)	`realonly:true;`
	(%o5)	`true`
Solving the equations with respect to a and b, the expressions are assumed to be set to zero.	(%i6)	`res:solve(abl[1],[a,b]);`
	(%o6)	`[[a=1,b=2]]`
Substituting the solutions into f yields the "optimum" transfer function.	(%i7)	`fopt:ev(f,res);`
	(%o7)	$\frac{1}{s^3+s^2+2\,s+1}$

9 Controller Design

`gain_optimum(`*Fs(s)*,*Fr(s)*`)`	Calculation of a controller according to the gain optimum.

Controller Design

`gain_optimum` calculates the parameters of an optimum controller $F_R(s)$ for a given plant $F_(s)$. The strucuture of the controller and the names of its parameters are freely chooseable on principal. It depends on the reasonableness of the assumtions for the controller, whether solutions for the controller paraneters are found actually (e.g. a PT1-controller will presumeably not yield soutions).

Transfer function of a plant

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

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

List of an I-, PI- and a PID-controller

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

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

Gain optimum for the I-controller

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

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

Gain optimum for the PI-controller; the zero of the controller approximately compensates the dominating pole of the plant.

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

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

Gain optimum for the PID-controllerr; the zeros of the controller approximately compensate the two dominating poles of the plant.

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

(%o5) $$[Tc=3.6197,\,Ta=4.9984,\,Tb=1.3966]$$

Substituting the results into the controllers

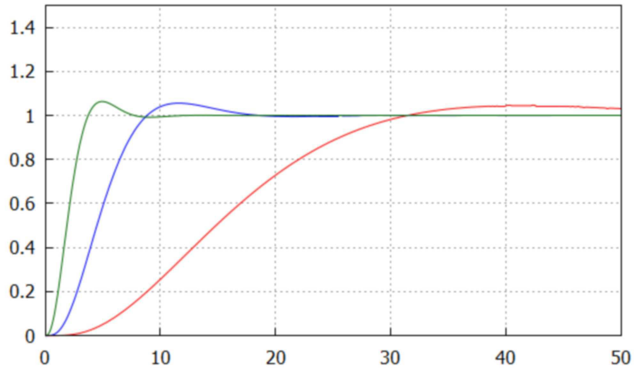
(%i6) `reli:float(ev([fri,frpi,frpid],[g1,g2,g3]));`

(%o6) $$\left[\frac{0.034247}{s},\,0.58988\left(\frac{0.19504}{s}+1.0\right),\,\frac{0.27627\,(1.3966\,s+1.0)\,(4.9984\,s+1.0)}{s}\right]$$

The step responses confirm about 5% overshoot and rise times in the amount of about 4.7 times the sum of the remaining time constants.

(%i7)
```
step_response (float (ev(closed_loop (reli*fs),
    res)),yrange=[0,1.5])$
```

(%t7)



The plant can also have symbolic coefficients.

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

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

The results are formulas for the optimum controller parameters.

(%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 State Space

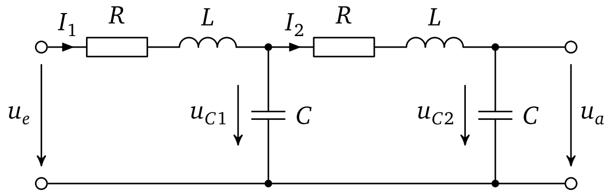
System: $[A, B, C, D]$	Definition of a linear system as a list of state matrices A, B, C und D
`systemp`$(A, B, C\,[, D])$ `systemp`$(system)$	Checks, whether *system* is a valid system constiting of state matrices.
`nsystemp`$(A, B, C\,[, D])$ `nsystemp`$(system)$	Cecks, whether *system* forms a valid linear system, wherein all matrix elements evaluate to numbers.
`transfer_function`$(A, B, C\,[, D])$ `transfer_function`$(System)$	Calculation of the transfer function (or transfer matrix) from the state matrices
`controller_canonical_form`(f)	Calculation of the state matrices according to the controller canonical from the transfer function f
`observer_canonical_form`(f)	Calculation of the state matrices according to the observer canonical from the transfer function f
`controllability_matrix`(A, B) `controllability_matrix`$(System)$	Calculation of the controllability matrix
`observability_matrix`(A, C) `observability_matrix`$(System)$	Calculation of the observability matrix

State space representation

The notion *system* means a subsumption of the four state matrices **A** (system matrix), **B** (input matrix), **C** (output matrix) and **D** (transit matrix) into a list. The transit matrix **D** can be omitted for systems without feedthrough, for systems with one input and one output **D** can be a scalar value d.

All funcions, which can have a *system* as the parameter, can also receive the particular state matrices as parameters (without subsumption into a list).

Electrical quadripole with state equations:



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

The state matrices **A**, **B** and **C** result directly from the state equations; a direct feedthrough from the input voltage u_e to the output voltage u_a does not exist, thus $\mathbf{D} = 0$ and can be omitted.

System matrix **A** of the circuit

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

Input matrix **B** of the circuit

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

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

Output matrix **C** of the circuit

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

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

Subsumption of the state matrices into a *system*

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

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

The list of state matrices are forming a valid *System* . . .

```
(%i5) systemp(circuit);
(%o5) true
```

. . . however, their elements do not eveluate to numbers.

```
(%i6) nsystemp(circuit);
(%o6) false
```

`transfer_function` calculates the transfer function (or the transfer matrix in multivariable systems) from the state matrices. That function is “polymorphic” in a certain sense: if the parameters are not state matrices but a list of linear equations and lists of variables, the transfer function is calculated from those equations (section 3).

Calculation of the transfer function; providing a *system*, ...

(%i7) f:transfer_function (circuit);

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

... as well as particular state matrices is possible.

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

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

Direct calculation of an impedance chain yields the same result expectedly.

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

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

The state matrices can be calculated from the transfer function according to the controller canonical form or the observer canonical form:

Controller canonical form of the state matrices

(%i10) circ1:controller_canonical_form (f);

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

Observer canonivcal foem of the state matrices

(%i11) circ2:observer_canonical_form (f);

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

Controllability matrix

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

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

Observability matrix

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

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

The system is controllable and observable.

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

(%o14) [4,4]

11 Various Functions

The package *COMA* provides several auxiliary functions which are not specific to control engineering, but can be useful in various calculations.

R_1 `//` R_2	Parallel connection of two resistors R_1 and R_2
Z `/_` φ	Polar coordinate representation of a complex number $z = Z\angle\varphi$ (Z „cis“ φ)
z `cf`	Postfix-Operator, outputs the complex number z in polar coordinate representation $Z\angle\varphi$
`chop(`*x*`)`	Replaces all numbers in the expression x, which are less than 10^{-10}, by 0
`coefficient_list(`*p*,*x*`)`	List of the coefficients of the polynomial p in the variable x
`set_option(`*name*=*val*, *list*`)`	Setting or adding an element as a key-value pair to *list*
`delete_option(`*name*, *list*`)`	Deleting of a key-value pair *name* from *list*
`option_exists(`*name*, *list*`)`	Checks, whether a key-value pair *name* exists in *list*
`list_option_exists(`*name*, *list*`)`	Checks, whether a key-value pair with the name *name* exists in *list* and its value is a list itself

Various functions

„//“ and „/_“ are two operators common in electrical enigneering: for the parallel connection of resistors and for complex quantities in polar coordinate representation. The postfix operator `cf` outputs a complex quantity in polar coordinates.

Parallel connection of impedances

```
(%i1) R // 1/(s*C) // s*L + R1;
```

(%o1) $\frac{L\,R\,s}{C\,L\,R\,s^2+L\,s+R}+R1$

Polar coordinate represenation of complex quantities

```
(%i2) [U1,U2,U3]:[230,230/_240,230/_120];
(%o2) [230,-199.19 %i-115.0,199.19 %i-115.0]
```

Output of a complex quantity in polar coordinates

```
(%i3) U1+U3 cf;
(%o3) 230.0 /_ 60.0
```

`coefficient_list` builds a list of the polynomial coefficients in increasing order:

Polynomial in the variable x

```
(%i4) p:5*(x+y)^2+a*x^5;
```

(%o4) $5\left(y+x\right)^2+a\,x^5$

List containing the polynomial coefficients	(%i5)	`coefficient_list (p,x);`
	(%o5)	$[5\,y^2, 10\,y, 5, 0, 0, a]$

The function chop removes all numbers from an expression, which are less than 10^{-10}. That is useful for "ironing out" numeric bugs:

When the numeric fails, ...	(%i6)	`x3:expand((s^2-1.1*s+1.1) *(s^2+1.1*s+1.1)*(s^2+1));`
	(%o6)	$s^6+1.99\,s^4+2.2204\,10^{-16}\,s^3+2.2\,s^2+1.21$
... erroneously emerging small numbers can be removed.	(%i7)	`chop(x3);`
	(%o7)	$s^6+1.99\,s^4+2.2\,s^2+1.21$

An associative array (hash), consisting of key-value pairs, is implemented as a list. It is well suited for saving preferences ("options") and for named parameters of functions (e. g. graphic routines).
Some routines facilitate the handling of associative arrays:

List of key-value pairs	(%i8)	`opts:[color=blue,xrange=[0,10]];`
	(%o8)	[color = blue , xrange = [0 , 10]]
Replacing a value	(%i9)	`set_option (color=red,opts);`
	(%o9)	[xrange = [0 , 10] , color = red]
Unless a key exists, a new key-value pair is generated.	(%i10)	`set_option (title="Test",opts);`
	(%o10)	[xrange = [0 , 10] , color = red , title = *Test*]
Removing a key-value pair	(%i11)	`delete_option (color,opts);`
	(%o11)	[xrange = [0 , 10] , title = *Test*]
Checking, whether a key exists	(%i12)	`option_exists (xrange,opts);`
	(%o12)	true
Reading a hash value	(%i13)	`get_option (title,opts);`
	(%o13)	*Test*
Returning a default value, unless a key exists	(%i14)	`get_option (color,opts,red);`
	(%o14)	red

The function `get_option` to read out a value would not be not neccessary on principal, for that purpose the Maxima function `assoc` is available. Contrary to `assoc`, `get_option` accepts also lists, which not only contain key-value pairs, but also any arbitrary expression (which is used internally by other by other COMA-functions).

Bibliography

[1] Maxima Development Team: *Maxima Manual Version 5.39.0,* `http://maxima.sourceforge.net` 2016 .

[2] Various Authors: *Gnuplot 4.6, An Interactive Plotting Program,* `http://www.gnuplot.info` 2014 .

[3] Rodríguez Riotorto, M.: *A Maxima-Gnuplot Interface,* `http://riotorto.users.sourceforge.net/Maxima/gnuplot/index.html`.

[4] Haager W.: *Computeralgebra mit Maxima,* Hanser München, Leipzig 2014.

[5] Haager W.: *Regelungstechnik kompetenzorientiert,* Hölder Pichler Tempsky Wien 2016.

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