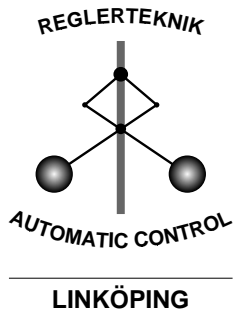


**Constrained Control Package**  
**(version 1.1b2, August 2001)**  
**– A Guided Tour –**

Wolfgang Reinelt

Department of Electrical Engineering  
Linköping University, 581 83 Linköping, Sweden  
WWW: <http://www.control.isy.liu.se/~wolle/>  
Email: [wolle@isy.liu.se](mailto:wolle@isy.liu.se)

August 2001



Report no.: LiTH-ISY-R-2351

Technical reports from the Automatic Control group in Linköping are available by anonymous ftp at the address [ftp.control.isy.liu.se](ftp://ftp.control.isy.liu.se). This report is contained in the portable document format file `2351.pdf`.

# Constrained Control Package (version 1.1b2, August 2001) – A Guided Tour –

Wolfgang Reinelt\*

Division of Automatic Control, Dept of Electrical Engineering,  
Linköping University, 581 83 Linköping, Sweden,  
email: [wolle@isy.liu.se](mailto:wolle@isy.liu.se), <http://www.control.isy.liu.se/~wolle/>

## Abstract

This software package provides algorithms for analysis and design of saturation avoiding control systems. The design of linear controllers for linear, possibly multivariable or uncertain, plants is considered. To avoid a conservative design, the external signals are hard bounded in amplitude and rate. During the example sessions, only reference signals are considered as external signals, but this can be extended quite easily.

## 1 Introduction & Disclaimer

This manual does not intend to describe the ideas or the theoretical background behind the implemented algorithms – therefore, reference to appropriate literature is given. Moreover, only the basic usage of all functions is described, for more options and additional parameters, see the online help of each function. This manual only gives an idea of how to use all functions “in a row”.

When using this package, a citation like [9] is welcome. For recent developments of this package, check its homepage at <http://www.wolfgang-reinelt.de/cc/>. Any comments, bugs, suggestions are welcome and may be directed to [me@wolfgang-reinelt.de](mailto:me@wolfgang-reinelt.de).

## 2 Installation

1. Unzip the file `cc-YYMM.tgz`. It extracts in a directory `cc-YYMM`. It was packed using `gnu-tar: gtar -czf cc-YYMM.tgz cc-YYMM`. To unpack use for example: `gtar -xzf cc-YYMM.tgz`
2. The contents of the directory should be the following:
3. Add these directories to your MATLAB-path using the `addpath`-command, or add it directly in your `/matlab/startup.m` file.
4. The main part is coded in Matlab, but the package also contains some `c-files`. Executable `MEX-files` are present for the following environments: Solaris 5.6, Linux 2.x. The `MEX-files` will

---

\*Partly supported by German research council DFG, while being with Paderborn University, Paderborn, Germany, which is gratefully acknowledged.

work under a not too old OS version and any Matlab 5.x version. If not or if your OS type is not present, you have to create the MEX-files: simply change to the directory `hidden/` and type

```
mim.isy.liu.se{wolle}: mex ccBalken.c
mim.isy.liu.se{wolle}: mex ccEvalInvLaplace.c
mim.isy.liu.se{wolle}: mex ccBuschel.c
```

For troubleshooting see the Matlab User's Guide. For detailed information on the precompiled MEX-files, see end of file.

5. Main developing and testing platform is Solaris 5.7 and Matlab 6.0.

### 3 The testdata set

The directory `doc` contains, additionally to this documentation, a dataset that contains some simple transfer functions and bounds to test the basis routines and the data for the aircraft example in [1], used later.

```
>> load testdata
>> Your variables are:
G1          U1          Udot1          aircraft      weight_s
G2          U2          Udot2          w_c
G3          U3          Udot3          weight
```

### 4 Worst Case Output of Systems

There are four methods implemented in order to calculate the maximum possible output amplitude of an LTI system, with input bounded in amplitude and rate. They are described in [5]. The main routines are, however, `ccmaxout` (using Linear Programming), which handels SISO and MIMO systems in discrete and continuous time, all with the same notation. For the SISO case, there are alternatives implemented the “constructive” approach to the worst case input signal, described in [3, 2] and implemented as `ccmaxoutC` and `ccmaxoutD` respectively. These two algorithms are added to the package for “historical” reasons, and if there is any difference in the result, we believe that the Linear Programming in `ccmaxout` gives the more accurate result.

The counterpart for uncertain SISO systems (in both discrete and continuous time) is `ccsetout`.

We start up with *SISO, continuous time systems*. We have two options:

```
>> [ymax,umax,time]=ccmaxout(G3,U3,Udot3,1); % using LP techniques
>> [ymaxC,tptsC]=ccmaxoutC(G3,U3,Udot3,1); % using Reichel's algorithm
```

The results for the `testdata` set are quite the same:

```
case 1: ymax = 0.7370; ymaxC = 0.7610
case 2: ymax = 0.0302; ymaxC = 0.0596
case 3: ymax = 0.1278; ymaxC = 0.1272
```

We look at *SISO systems in discrete time* and observe, that the notation for `ccmaxout` does not change at all:

```
>> G1d=c2d(G1,0.1); % etc
>> [ymax,umax,time]=ccmaxout(G1d,U1,Udot1,1); % using LP techniques
>> [ymaxD,tptsD]=ccmaxoutD(G1d,U1,Udot1,1); % using Peng's algorithm
```

The results for the testdata are:

```
case 1: ymax = 0.7787; ymaxD = 0.7858
case 2: ymax = 0.0677; ymaxD = 0.0714
case 3: ymax = 0.1278; ymaxD = 0.1237
```

*MIMO systems* can be handled by `ccmaxout` in any time domain. We create system with 2 input and 3 outputs:

```
>> Gm=tf([G1,G2;G3,G2;G1*G1, G3]);
>> ymaxM=ccmaxout(Gm,U1,Udot1,1)
>> ymaxM = [1.1274;1.7116;1.7475]
```

A similar result can be obtained, when looping through all entries with `ccmaxoutC` (or `ccmaxoutD`).

*Uncertain systems* have to be parameterised as  $G + [B_1, \dots, B_n] \cdot \theta$ , where  $G$  is a SISO system,  $B = [B_1, \dots, B_n]$  is a set of basis functions and the parameter vector  $\theta$  is living in a box. The problem is then solved by Quadratic Programming invoking `ccsetout`. The set of basis functions  $B$  can be any LTI object with  $n$  inputs and 1 output or, an I4CBASIS object, as used in the `i4c` package [6] (when choosing this option, you'll then have to install this software).

```
>> BasDat.pole=1;BasDat.order=3;BasDat.Ts=0;BasDat.type='lag';
>> thbox=[0.9 1.1];
>> % using the I4CBASIS object BasDat:
>> [ymaxU,umax,time,thetamaxU]=ccsetout(G1,BasDat,thbox,U1,Udot1,1);
>> % which is equivalent to
>> Bas=basis(BasDat); % Bas is a 'normal' ss-object now
>> minfo(Bas)
MATLAB ss object:   3 states      1 outputs      3 inputs
>> [ymaxU,umax,time,thetamaxU]=ccsetout(G1,Bas,thbox,U1,Udot1,1);
```

The result, independent which description for the basis we pick, is  $ymaxU = 2.6793$ ;  $thetamaxU = [1.1000; 1.1000; 1.1000]$ .

## 5 Controller Design via Loop Shaping

As a first design technique, Loop Shaping [1] for SISO or MIMO systems is considered. By this, we mean clever adjustment of the design weight during “classical” Loop Shaping such that the hard bound on each control channel are met. We start with a simpler example, controller design in the SISO case, which is also outlined in [4, Sec.6.2].

```
>> % the plant & its input constraints
>> plant=tf([1 -1],[1 2 0]);
>> w1=ss(0.0817);
>> R=7; Rdot=2;
>> pf=1; % performance factor for the Loop Shaping procedure
>> [u_max,e_max,K]=ccls(plant,R,Rdot,w1,[],pf,[1e-5 1e5]);
```

As a result, we obtain:  $u_{\max} = 0.9866$ ;  $e_{\max} = 0.6860$  and  $K(s) = (0.086661s + 0.17322)/(s + 2.249)$  as controller via `tf(K)`. A dynamical weight can be considered as well, of course:

```
>> % new weight:
>> w1new=tf([0.03841 0.002289],[1 0.001]);
>> [u_max,e_max,K]=ccls(plant,R,Rdot,w1new,[],pf,[1e-5 1e5]);
>> tf(K)
Transfer function:
0.0683 s^3 + 0.1417 s^2 + 0.01027 s + 0.0001235
-----
s^3 + 2.18 s^2 + 0.09815 s + 9.597e-05
```

Here, we end up with a maximum control signal  $u_{\max} = 0.9213$  and a stability margin  $e_{\max} = 0.4902$ . For adjustment of the design weights, `magshape` ( $\mu$  Toolbox) for instance is an appropriate tool and `sys2lti` (i4c-package) is a nice tool to convert the mutools SYSTEM variable to an LTI object. For a controller design in the MIMO case, see [4, Sec.6.4] or [7], an aircraft model and some weights is already contained in `testdata`:

```
>> minfo(aircraft) % aircraft model
MATLAB ss object: 5 states      3 outputs      3 inputs
>> minfo(weight_s) % adaped weight, converted from magshape:
MATLAB ss object: 3 states      3 outputs      3 inputs
```

As pointed out, there are different ways to produce the weighting functions. Define as transfer functions:

```
>> w_c=tf([1 0.4],[1 0]);
>> weight=[24*w_c,0,0;0,12*w_c,0;0,0,24*w_c];
>> % or fetch diagonal entries wi from magshape:
>> weight_s=[sys2lti(w1),0,0;0,sys2lti(w2),0;0,0,sys2lti(w3)]
```

Having these weighs at hand, the actual controller design runs as in the SISO case (from the syntax point-of-view):

```
>> R=[1 1 1]; Rdot=[5 11 3]; pf=1.1;
>> [u_max,e_max,K]=ccls(aircraft,R,Rdot,[],weight,pf,[1e-3 1e3]);
>> [u_max_s,e_max_s,K_s]=ccls(aircraft,R,Rdot,[],weight_s,pf,[1e-3 1e3]);
```

The results are somewhat different to those given in [4, Sec.6.4], as we use another implementation for determining the maximum output amplitude  $u_{\max}$  here:

```
u_max = [26.5273; 10.4946; 61.4195]; e_max = 0.3786;
u_max,s = [20.3149; 8.1607; 49.1611]; e_max,s = 0.3525.
```

## 6 Design of Optimal Controllers

In this section, we would like to discuss a method for designing optimal controllers. Optimality refers to the fact that we achieve the smallest possible worst case error (i.e. difference between reference signal and plant output) during runtime. Moreover, the presented framework allows to assess feasibility of the constraint control problem. For details, we refer to [8]. This approach relies upon parameterisation of all stabilising controllers via Youla parameterisation. Suppose a plant to be controlled, along with bounds on amplitude and rate of the reference signal:

```

>> R=1; Rdot=0.8;
>> Gnom=tf([1 -1],[1 -2 0]);
>> Qorder=6;          % define order of the Youla parameter
>> Alpha=[0:0.1:1];% grid on alpha
>> [Result]=optidesign (Gnom,R,Rdot,Qorder,Alpha);

```

The structure `Result` then contains the solution of the Pareto-optimal problem on the grid `alpha`: Youla parameter, controller and the values for maximum control amplitude and maximum error amplitude. Solving this problem involves non-linear optimisation and will take some time (in this example about 30 minutes per grid point in order to arrive at a reasonable stage).

## References

- [1] D. C. McFarlane and K. Glover. *Robust Controller Design Using Normalized Coprime Factor Plant Description*. Number 138 in Lecture Notes in Control and Information Science. Springer Verlag, Berlin, Germany, 1989.
- [2] X. Peng. *Rechnerunterstützte Synthese von Abtastregelkreisen mit Beschränkungen*. PhD thesis, Dept of EE, University of Paderborn, 33095 Paderborn, Germany, 1992.
- [3] R. W. Reichel. *Synthese von Regelsystemen mit Beschränkungen bei stochastischen Eingangsgrößen*. PhD thesis, Dept of EE, University of Paderborn, 33095 Paderborn, Germany, 1984.
- [4] W. Reinelt. *Robust Control of Systems subject to Hard Constraints*. PhD thesis, Dept of EE, Univ of Paderborn, 33095 Paderborn, Germany, Apr. 1998.
- [5] W. Reinelt. Maximum output amplitude of linear systems for certain input constraints. In *Proc. of the IEEE Conference on Decision and Control*, pages 1075–1080, Sydney, Australia, Dec. 2000.
- [6] W. Reinelt. *i4c: Identification for Control Package (version 1.1b5)*. Linköping University, Linköping, Sweden, Sept. 2000. <http://www.wolfgang-reinelt.de/i4c/>.
- [7] W. Reinelt. Control of multivariable systems with hard constraints. In *Proc. of the European Control Conference*, pages 2576–2581, Porto, Portugal, Sept. 2001.
- [8] W. Reinelt. Design of optimal control systems with bounded control signals. In *Proc. of the European Control Conference*, pages 348–353, Porto, Portugal, Sept. 2001.
- [9] W. Reinelt. *cc: Constraint Control Package (version 1.1b2)*. Linköping University, Linköping, Sweden, Aug. 2001. <http://www.wolfgang-reinelt.de/cc/>.

## A Compiler options

### A.1 Solaris: \*.mexsol

```

-----
Matlab Version: 5.3.0.10183 (R11)
OS Version: SunOS 5.6 Generic_105181-13 sun4u sparc SUNW,Ultra-1
Host: mim.isy.liu.se

```

Compiled with mex-options:

```
->  MATLAB          = /sw/matlab/5.3
->  CC              = cc
->  CC flags:
      CFLAGS        = -dalign -KPIC
      CDEBUGFLAGS   = -g
      COPTIMFLAGS   = -O -DNDEBUG
      CLIBS         =
      arguments     =
->  FC              = f77
->  FC flags:
      FFLAGS        = -dalign -KPIC
      FDEBUGFLAGS   = -g
      FOPTIMFLAGS   = -O
      FLIBS         =
      arguments     =
->  LD              = /usr/ccs/bin/ld
->  Link flags:
      LDFLAGS       = -G -M /sw/matlab/5.3/extern/lib/sol2/export.map
      LDDEBUGFLAGS  =
      LDOPTIMFLAGS  =
      arguments     =
```

## A.2 Linux: \*.mexlx

-----  
Matlab Version: 5.2 (R10)

OS Version: Linux 2.0.27 #1 Sat Dec 21 23:44:11 EST 1996 i586

Host: magnani.upb.de

Compiled with mex-options:

```
->  MATLAB          = /usr/local/matlab52-V10
->  CC              = gcc
->  CC flags:
      CFLAGS        =
      CDEBUGFLAGS   = -g
      COPTIMFLAGS   = -O -DNDEBUG
      CLIBS         =
      arguments     =
->  FC              = f2c
->  FC flags:
      FFLAGS        =
      FDEBUGFLAGS   = -g
      FOPTIMFLAGS   =
      FLIBS         = -lf2c -Wl,--defsym,MAIN__=mexfunction_
      arguments     =
->  LD              = gcc
->  Link flags:
      LDFLAGS       = -shared -rdynamic
```

LDDEBUGFLAGS =  
LDOPTIMFLAGS =  
arguments =

---