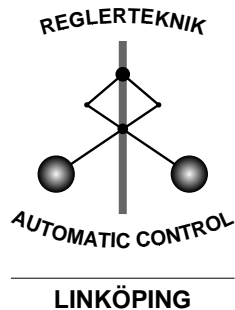


Identification for Control Package
(version 1.1b5, August 2001)
– A Guided Tour –

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

September 2000. Update: August 2001



Report no.: LiTH-ISY-R-2294

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 `2294.pdf`.

Identification for Control Package (version 1.1b5, 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, <http://www.control.isy.liu.se/~wolle/>

Abstract

This software package provides algorithms for control oriented identification, based on measured data. The techniques implemented in particular are non-stationary stochastic embedding, model error modeling using prediction error methods and set membership estimation. Analysis of the obtained model sets is possible using ν -gap methods. A robust controller design with respect to parametric and/or dynamical errors is implemented as well. Most of these methods rely heavily on the usage of orthonormal basis functions, which are described in an object oriented fashion in this package.

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”.

The underlying procedures for stochastic embedding have been implemented by Julio H. Braslavsky (Universidad Nacional de Quilmes, Bernal, Argentina, jbrasla@unq.edu.ar). The basic procedures for set membership estimation have been implemented by Andrea Garulli (Universita di Siena, Siena, Italy, garulli@ing.unisi.it). The original sources are edited and modified by now, in order to synchronize them to the IDDATA format and the object-oriented description of basis functions. The original author is stated in the source code of all m-files. The total amount of source code is about 5000 lines by now, of which are about 2100 and 800 originally written by Andrea Garulli and Julio Braslavsky respectively. Therefore, this package may only be used with permission of all three authors. When using this package, a citation like [15] is welcome. For recent developments of this package, check its homepage at <http://www.wolfgang-reinelt.de/i4c/>. Any comments, bugs, suggestions are welcome and should be directed to me@wolfgang-reinelt.de.

*Supported by the European Union within the European Research Network in System Identification (ERNSI) under contract number ERB FMRX CT98 0206, which is gratefully acknowledged.

2 Installation

1. Unzip the file `i4c.tgz`. It extracts in a directory `i4c-YYMM`. It was packed using `gnu-tar: gtar -czf i4c.tgz i4c-YYMM`. To unpack use for example: `gtar -xzf i4c.tgz`. You may then rename this directory to `i4c`, or set a symbolic link, in order to be able to type `help i4c` instead of `help i4c-YYMM` (for the remainder, we assume that you do something like this; see below)
2. The contents of the directory is then the following:

```
mim.isy.liu.se{wolle}: ls
Contents.m  basisfunc  doc          auxiliary  cd-tools  id-tools
```

The contents of the subdirectories is given at the end of the file `Contents.m`.

3. Add this directory, and the subdirectories `auxiliary`, `basisfunc`, `id-tools` and `cd-tools` to your MATLAB-path using the `addpath`-command, or add it directly in your `~/matlab/startup.m` file. For example:

```
addpath /home/rt/wolle/sysID/software/i4c/i4c
addpath /home/rt/wolle/sysID/software/i4c/i4c/auxiliary
addpath /home/rt/wolle/sysID/software/i4c/i4c/basisfunc
addpath /home/rt/wolle/sysID/software/i4c/i4c/id-tools
addpath /home/rt/wolle/sysID/software/i4c/i4c/cd-tools
```

4. The package was developed using the following Software environment, which is also necessary to use all functions:

MATLAB Toolbox	Version 5.3 (R11)	15-Jan-1999
System Identification Toolbox	Version 5.0 beta 2 (R12 beta 4)	14-Nov-1999
Control System Toolbox	Version 4.2 (R11)	15-Jul-1998
LMI Control Toolbox	Version 1.0.5 (R11)	09-Sep-1998
Mu-Analysis and Synthesis Toolbox	Version 3.0.4 (R11)	10-Apr-1998
Optimization Toolbox	Version 2.0 (R11)	09-Oct-1998
Robust Control Toolbox	Version 2.0.6 (R11)	10-Apr-1998

3 Related/necessary software

3.1 Matlab's System Identification Toolbox, version 5

In order to proceed, make yourself familiar with the usage of `IDDATA` objects from the System Identification Toolbox, version 5.0, beta2, November 1999 [11]. Read the manual or type

```
>> idhelp
>> help ident
>> help iddata
>> help idmodels
>> help idprops
```

in the MATLAB prompt.

3.2 SOCP/SP

In order to use the routines for analysis with ν -gap and for controller design, some LMI solvers are needed. So far, standard LMI problems are solved with Matlab's LMI toolbox. To solve Second Order Cone Problems, the software SOCP [12] is used. It is available at <http://www.stanford.edu/~boyd/SOCP.html> and the routines have to be added to the MATLAB-path:

```
addpath /home/rt/wolle/matlab/socpfun/
```

In future, semidefinite programming problems might be solved with SP, consult <http://www.ee.ucla.edu/~vandenbe/sp.html> for closer information.

4 i4c Quick start

Type

```
>> help i4c
```

Online help for all the functions is available as well. Links to other functions are mentioned.

5 Orthonormal Basis functions

Functions, related to handling of orthonormal basis functions:

basisdata creates/describes a BASISDATA object (empty now)

basis generation of basis: calculated state space description

polechoice2 best pole of Laguerre functions for a fixed order (approximation of input by Laguerre-series [4])

nombasis compute model in basis parameterization

kautzpar generation of Kautz parameters, given the poles

All relevant information about the orthonormal basis functions, cf. [14, 19], are contained in a object produced by the command **basisdata**. Well, right now, it's only a structure, which should be organized as follows:

BasisData.type 'Laguerre' or 'Lag' for a Laguerre basis.

'Kautz' or 'Kau' for a Kautz basis .

'GOB' for a generalized orthonormal basis.

'free' for a MISO system consisting of 1st/2nd order systems with poles as specified and numerator 1.

BasisData.Ts empty or 0: for continuous time bases.

positive: Sampling time for discrete time bases.

negative: ERROR, state sample time in discrete time case!

BasisData.order order of the basis, except for the 'free' type of basis, where it is the number of basis functions (i.e. = number of poles specified plus probably one, for the FeedForward function)

BasisData.pole *p* (scalar): when *p* is the pole of the Laguerre-basis (the absolute value in the continuous time case, as we assume stable basis functions).

[*c,d*]: the Kautzpair for a Kautz-basis, requires even **BasisData.order**, see **kautzpar**.

[*p1...pn*] arbitrary stable poles for generalizes orthonormal bases (the absolute value in the continuous time case, as we assume stable basis functions). Here, **BasisData.order=n** should hold.

BasisData.PoleAccuray **basis** performs a test, if the poles are there where they should be. PoleAccuray is the treshold for this test (absolute value of difference). Default:0.001

BasisData.FeedForward specifies, if the basis has feedforward (D_{i0}) or not. Default:0 (D-vector of ss-representation=0). If $i0$, $D(1)=\text{FeedForward}$ is choosen (this is a dirty solution) In the case of the free basis, an extra constant channel (=1) is added.

Currently supported are the state space-descriptions of

- discrete time Laguerre, Kautz and GOB
- continuous-time Laguerre and GOB

and the direct frequency evaluation of continuous-time Laguerre bases, a feature that might die in future. The idea is to pass only this information between the functions, and not the actual state space-descriptions. Moreover, it enables validation of all parameters, when the object is described (when the **basisdata** command is ready), and not later during runtime.

For more information on that type

```
>> help basisdata
>> help basis
```

We produce a 7th order Laguerre basis with poles in -1.6139 in continuous time. Note, that you have to omit the minus sign, as we only regard stable Laguerre functions:

```
>> LagBase.Ts=0;
>> LagBase.type='Laguerre';
>> LagBase.pole=1.6139;
>> LagBase.order=7;
```

and a state space-description of that by

```
>> BaseSS=basis(LagBase);
```

which we can check by:

```
>> eig(BaseSS.a)
ans =
-1.6139 -1.6139 -1.6139 -1.6139 -1.6139 -1.6139 -1.6139
```

6 Identification

The three methods, implemented here, are described, discussed and compared in [17, 7, 16]. Consult the papers for detailed information.

6.1 A sample dataset

A dataset with the real plants is contained in `doc/sampleddata.mat` (`BasisModel`, `NominalE11`, `Shi`, `Slow`, `Srob`, `WDelta`, `W_S` are used later on in the controller design section):

```
>> load sampledata
>> whos
Name                Size          Bytes  Class

BasisModel          1x1            536   struct array
Glinear              1x1           2610   zpk object
Gnom                 1x1           2426   zpk object
Gspa                 4-D           12980  idfrd object
NominalE11          1x1            794   struct array
Shi                  1x1           3156   ss object
Slow                 1x1           3156   ss object
Srob                 1x1           2844   ss object
WDelta               1x1           4216   ss object
W_S                  1x1           2844   ss object
data1                3000x1x1      52696  iddata object
data2                3000x1x1      52694  iddata object
```

Grand total is 14156 elements using 140952 bytes

`data1` are 3000 samples with transients, `data2` is without transients (for stochastic embedding). `Glinear` is an (Control Systems toolbox) LTI object, `Gspa` is a System Identification Toolbox IDFRD object, that contains a spectral analysis of data using `spa`. The latter two objects can be used by the function `addreal` for comparison of identified models with the “real plant”.

Explore the datasets `data1`, `data2`:

```
>> get(data1)
ans =
    Domain: 'Time'
    Name: 'Linear plant with saturation, first samples'
    OutputData: [3000x1 double]
    y: 'Same as OutputData'
    OutputName: {'Noisy Output, nonlinear plant'}
    OutputUnit: {''}
    InputData: [3000x1 double]
    u: 'Same as InputData'
    InputName: {'Sum of 53 sinusoids'}
    InputUnit: {''}
```

```

        Period: Inf
    InterSample: 'zoh'
        Ts: 0.0400
        Tstart: 0
    SamplingInstants: [3000x0 double]
        TimeUnit: 's'
    ExperimentName: 'Clearly nonlinear, lhl'
        Notes: 'Userdata are: [excited frequencies].'
    UserData: [1x1 struct]

```

The input signal is a multi-sinusoid one, the excited 53 frequencies are contained in the `UserData`. This construction is expected by the stochastic embedding routine `nsse` and the least squares fit in the frequency domain `lsebasis` (which is actually the first step in stochastic embedding).

```

>> data1.UserData
ans =
    frequencies: [1x53 double]

```

6.2 Sample session with non-stationary stochastic embedding

Functions, related to stochastic embedding:

nsse stochastic embedding with different random-walk strategies

addreal adding the real plants to Bode/Nyquist plots

orderchoice best order for fixed Laguerre-pole (minimum error in `nsse`), cf. [1]

polechoice best pole for fixed order (minimum error in `nsse`), cf. [1]

tfpe calculate transfer function point estimates

tfpmerge merges two `tfpe`'s on different frequency sets

The stochastic embedding technique is described in [2, 8, 9]. We are using dataset `data2`, in order to get rid of initial transients.

First, get a transfer function point estimate and a plot of them:

```

>> PointEst=tfpe(data2,1);

```

You may look at the results:

```

>> PointEst
PointEst =
    Ghat: [53x1 double]
    SigmaV: [106x106 double]
    sigmaVhat: 27.0166

```

```
frequencies: [53x1 double]
Comment: 'Produced by tfpe. Time stamp 05-Apr-2000 15:54:05'
```

The actual estimates are stored in `Ghat`, the corresponding covariance matrix `SigmaV` and an unbiased estimate `sigmaGhat` of the variance of the measurement noise, assumed Gaussian. The frequencies used are also restored from the dataset. Estimates from two different frequency sets may be merged using `tfpemerge`.

A Laguerre basis `LagBase` was already defined in Section 5; we will use the object `LagBase` rather than the state space description, to pass information.

Now, some definitions for the identification.

```
>> Fn=logspace(-1,2); % according to injected freq's
>> CL=80; % confidence level
>> Walk='rw'; % use random walk
```

Invoke stochastic embedding and add the “real plants” to the estimation. Note, that `Glinear`, may be either continuous or discrete time. It is converted using `d2c` in the latter case. Using a spectral analysis for comparison, the (discrete) frequencies are converted to continuous ones.

```
>> [Gbar,Qhat,Stats]=nsse(PointEst,LagBase,Walk,Fn,CL);
>> addreal(data2,Glinear)
>> %or
>> [Gbar,Qhat,Stats]=nsse(PointEst,LagBase,Walk,Fn,CL);
>> addreal(data2,Gspa)
```

The nominal model is now contained in the LTI-object `Gbar`, `Qhat` are the coefficients of the basis expansion, thus, the following gives us the nominal model as well:

```
>> Gbar2=nombasis(Qhat,LagBase)
```

No idea, how to pick the pole? At least for Laguerre-expansions, there are two guidelines: The first is, to minimize the error in terms of the random walk [1]. Here one might choose another interval of frequencies `Fnsearch` to emphasize important frequencies, maybe only up to the crossover.

```
>> Fnsearch=Fn;
>> optipole1=polechoice(PointEst,LagBase.order,...
                        linspace(0.1,1.2,10),Walk,Fnsearch)
```

Alternatively, one can expand the input signal in a truncated Laguerre series, and pick the pole that minimizes this approximation error [4]:

```
>> orderLAGEXP=7;
>> optipole2=polechoice2(data2,orderLAGEXP,linspace(0.5,4.5,80),1)
```

Having found a better pole, you might then feed it back to your basis:

```
>> LagBase.pole=optipole2; % hope this is better...
```

A somewhat related function is `orderchoice`, which picks the order by a criterion related to that used in `polechoice`, see [1]. Having a nonlinear system or a low signal to noise ration, this function normally picks the highest possible order.

6.3 Sample session with model error modeling, using prediction error methods

Functions, related to model error modeling, using prediction error methods and least squares techniques:

mempem Model Error Modeling using PEM

addreal adding the real plants to MEMPEM and SM-plots

lsebasis Least Squares estimate of model using a certain basis, time or frequency data

Using the `pem` procedure of the System Identification Toolbox, there is not much to add for the model error modeling setup, which is fully described in [10].

Get a nominal model of order 7, display the real plants as well and look at the residual analysis:

```
>> Gbar=pem(data1,'nb',6,'nf',7);
>> bode(Gbar,'sd',3,'fill','A'),grid
>> addreal(dataFirst,Glinear)
>> resid(data1,Gbar);
```

The actual model error modeling procedure is then to pick a clever structure (which usually needs some trials) for the model error model, and calculate the stuff

```
>> pemmem=mempem(data1,Gbar,[1,9,5,5,10,3]);
>> addreal(data1,Glinear)
```

6.4 Sample session with Set Membership Estimation

Functions, related to Set Membership Estimation:

smeoe SM identification of basis-parameterized output error functions in time or frequency domain (ellipsoids)

smpoe SM identification of basis-parameterized output error functions in time domain (parallelotopes)

smband uncertainty band of basis-parameterized output error functions in bode plot (ellipsoids and parallelotopes)

smbandplot plots nominal model and uncertainty band (basis-parameterized)

smbandmem Bode envelope of nominal model and model error model (basis-parameterized)

addreal adding the real plants to MEMPEM and SM-plots

smrpe Restricted projection estimate in l_∞ norm

simpolechoice selects Laguerre pole for restricted projection (nominal model and central estimate with ellipsoidal uncertainty as *unfalsified* uncertainty region.)

smearx SM identification of ARX model in time domain (ellipsoids)

smparx SM identification of ARX model in time domain (parallelotopes)

smpqarx SM identification of ARX model in time domain (parallelotopes - block processing)

Somewhat classical references for the Unknown-but-bounded (UBB) approach are [5, 13, 20]. Conditional algorithms are described in [6] and the parameterization using orthonormal basis functions is discussed in [17].

We start with identifying output error models, parameterized by basis functions. A basis for the nominal model would be for example:

```
>> BasisN.type='Laguerre'; BasisN.Ts=data1.Ts;
>> BasisN.order=7; BasisN.pole=0.95;
```

Generating Kautz bases, `kautzpar` assists in calculating the Kautz-parameters from the pole locations.

Define the noise bound:

```
>> delta=2.5;
```

Get the SMI nominal model (central estimate) and plot this and add the real models:

```
>> [Gbar,FinalE11,LSEstimate]=smeoe(data1,delta,BasisN);
>> smband(FinalE11,BasisN);
>> addreal(data1,Glinear)
```

The central estimate `Gbar` is delivered as an LTI-object, and `FinalE11` is a structure describing the ellipsoid. This information is then passed to `smband`, that calculates and plots the uncertainty regions in the Bode plot, compare `smbandplot`.

```
>> FinalE11
FinalE11 =
    type: 'Ellipsoid'
   center: [7x1 double]
  matrix: [7x7 double]
  volume: 3.9258e-04
  Comment: 'Produced by smeoe. Time stamp 06-Apr-2000 10:12:33'
```

Compare the central estimate `Gbar` to the Least Squares estimate, which comes with `smeoe`, or separately with `lsebasis`:

```
>> GbarLS=nombasis(LSEstimate,BasisN);
>> bode(Gbar,'b-',GbarLS,'r--');
>> title('Bode plots: SME central estimate (b-), LS estimate (r--)')
```

Alternatively, `smeoe` will derive a central estimate from frequency data, delivered by `tfpe`. Another option is the generation of parallelotopic uncertainties, instead of ellipsoids, which can be done using `smpeo`.

For people who do not like basis function, the estimation of ARX models is possible using one of `smearx`, `smparx`, `smpqarx`, for instance:

```
>> Garx = smearx (data1,15,[3 3 1])
```

Having an uncertain output error model, with coefficients located in an ellipsoid, `dbenvetf` displays this uncertainty in the Bode-plot.

Not satisfied with the uncertainty region delivered by SMI so far? You might proceed with the model error modeling approach. Therefore, we need the residual data:

```
>> resdata=data1;ynom=lsim(idpoly(Gbar),data1);
>> resdata.y=data1.y-ynom.y;
```

and a basis for the model error model, additionally a noise bound:

```
>> BasisE.type='Laguerre'; BasisE.Ts=BasisN.Ts;
>> BasisE.order=12; BasisE.pole=0.92; deltaE=0.75;
```

Now, calculate the model error, together with its uncertainty ellipsoid and plot it, together with the real plants:

```
>> [MEM,FinalElle]=smeoe(resdata,deltaE,BasisE);
>> smbndmem(Gbar,FinalElle,BasisE);
>> addreal(data1,Glinear)
```

Another possibility is to jointly identify restricted projection estimate *and* unfalsified model error model, see [7] for details. First, you have to pick order of nominal and model error model:

```
>> order=[4 12];
```

and then a range for the Laguerre pole and noise bound, where you intend to search:

```
>> poles=[0.9:0.01:0.99]; bounds=[0.2 2.5];
```

The syntax for the actual identification procedure will then return optimal pole location, jointly for nominal and model error model, along with the minimum, non-falsifying noisebound. You may add the spectral analysis for comparison to the plot as well:

```
>> [optipole,deltamin]=smpolechoice(data1,order,poles,bounds,1);
>> addreal(data1,Gspa)
```

The restricted projection estimate alone (i.e. without model error model) can be calculated using `smrpe`.

7 Representation of uncertain (parametric) models in the frequency domain

Functions, related to representation of uncertain models in the frequency domain:

nsse stochastic embedding with different random-walk strategies

mempem Model Error Modeling using PEM

smbandplot, **smbandmem** plots nominal model and uncertainty band, possibly for the model error model as well (basis-parameterized set membership models)

dbenvetf Bode envelope of discrete time output error model, parameters are contained in ellipsoids.

Most of the functions mentioned above are discussed in Section 6, together with their identification routines. Additionally, **dbenvetf** computes and calculates the Bode envelope of discrete time output error model, parameters are contained in ellipsoids, see online-help for usage. A complete collection of tools for computation of exact bounds for the frequency response of an uncertain plant with ellipsoidal perturbations is available from the authors of [3] and therefore not treated here.

8 Robust Controller Design

Functions, related to robust controller design:

pypaugment Augment SISO plant with parametric and dynamic error.

pycdesign Design robust controller with performance specs on the sensitivity function.

pyccheck Check robustness of performance specs.

i4cyoula Performs Youla parameterization, used for controller design.

lti2sys/sys2lti Transform LTI object to (μ Toolbox-) SYSTEM variable and vice versa.

The method, underlying these functions is fully described in [18, ?]. Following the strategy described there, suppose, that a nominal plant has been identified using Set Membership techniques, i.e. based on the 2nd order Laguerre basis `BasisModel` (present in the `sampledata` file):

```
>> BasisModel
BasisModel =
  type: 'Laguerre'
  Ts: 0.0800
  order: 2
  pole: 0.9600
```

A nominal model `Gnom` is identified, along with an parametric uncertainty: the two coefficients for the Laguerre expansion live in the ellipsoid `NominalEll`. This identification can be carried out using `smeoe` and represented in the frequency domain using `smband`:

```
>> [Gnom,NominalEll]=smeoe(data,4.9,BasisModel);
>> smband(NominalEll,BasisModel);
```

Based on the nominal model, we can identify a dynamic and unstructured uncertainty (or model error model), based on residual data:

```
>> resdata=data;ynom=lsim(idpoly(Gnom),data);resdata.y=data.y-ynom.y;
>> % error basis
>> BasisE=BasisModel;BasisE.order=25;BasisE.pole=0.95;
>> deltaE=1.3;% noise bound
>> [GdynerrorID,FinalEllE]=smeoe(resdata,deltaE,BasisE);
>> [mag_e,pha_e,range_e]=smbandmem(Gnom,FinalEllE,BasisE);
```

The remaining task is to come up with a transfer function that describes/approximates the maximum amplitude of the model error, stored in the second row of `range_e`. This is simple curve matching, you can also treat this as an identification problem. The most simple way is probably to match/overbound the amplitude with the GUI `magshape` (μ -analysis toolbox). Tools that assist in converting from continuous frequencies to discrete ones and back and to transform μ -analysis toolbox SYSTEM variables to LTI objects are `df2cf`, `sys2lti` and their “inverse” functions. An approximation of the maximum error in this case is already stored in the LTI object `WDelta`. The upper bound for the “robust” sensitivity function, has to be specified as an LTI object. The sample dataset already contains such an upper bound stored in `W.S`.

The first step towards the controller design is to produce an augmented plant, based on the identified model and the weight for robust sensitivity, and prepare a structure `Plant`:

```
>> rho=1; % rhs for SMI ellipsoid
>> [Plant,Paug]=pypaugment(Gnom,BasisModel,NominalEll,rho,WDelta,Srob);
```

You do not necessarily need all types of uncertainties or performance specifications, you may simply skip arguments. You may note, that `Plant` is a structure, storing all relevant information about type of uncertainty, used basis, etc for you:

```
>>Plant
Plant =
  structure: 'deltaAddErrPerf'
          T1: [4x1 ss]
          T2: [4x1 ss]
  BasisData: [1x1 struct]
          Gbar: [1x1 zpk]
          Kc: [2x2 ss]
```

`Plant.structure` tells you in this case, that there is a parametric uncertainty `delta`, an additive model error `AddErr` and specifications on the robust performance `Perf` subject to the to uncertainty sources. `T1`,`T2` is the Youla parameterization of the nominal plant, `Kc` the central controller. The Youla parameter making up the *robust* controller is the ratio of two \mathcal{H}_∞ functions α and β , which we parameterize by means of Laguerre bases

```
>> BasisAlpha.type='Laguerre';BasisAlpha.Ts=BasisModel.Ts;
>> BasisBeta=BasisAlpha;
>> BasisAlpha.order=8;BasisAlpha.pole=-0.8;
>> BasisBeta.order=8;BasisBeta.pole=0.6;
```

The specs on lower and upper bound (transfer functions) of the nominal sensitivity function are stored in a structure

```
>> Specs.Slo=Slow; Specs.Shi=Shi;
```

You may want to change some of the default parameters for the design procedure:

```
>> Tuning.StartOmega=logspace(-3,log10(3.14/BasisBeta.Ts),50);
>> Tuning.OIB=2;Tuning.NoOmegas=70;
```

The big moment: controller design

```
>> Q=pycdesign(Plant,Specs,BasisAlpha,BasisBeta,1,Tuning);
>> % add upper bound for robust sensitivity by hand:
>> wvec=logspace(-3,log10(3.14/BasisBeta.Ts),200);
>> hold on,subplot(212),loglog(wvec,squeeze(bode(W_S,wvec)),'g-.');
```

The controller, in the case of a stable plant is then given by:

```
>> K=-Q*inv(1-Gnom*Q);%% controller for stable Gnom
```

We obtain a quite high order controller for the stable, second order plant, which is certainly overkill. Hence, we reduce this controller (we have to switch to continuous time and back in order to use Matlab's model reduction routines), which works for instance like that:

```
>> Kcont=minreal(d2c(K));
>> Kconty=schmr(Kcont,1,2);
>> bode(Kconty,Kcont);
>> Ky=minreal(c2d(Kconty,K.Ts));
```

In the same fashion, we can procure a first order controller, called K_n . Now, we have three controllers on the market. An amplitude plot shows the the reduced order controllers just remove the “bubbles” in the the mid-frequency range, that most likely appear due to parametrization via Youla and basis functions. We know (by construction), that controller K fulfills the performance specs and guarantees robustness against the uncertain parameters. But how about the other two guys? Any controller can be checked using the following function:

```
>> pyccheck(Plant,Specs,Kn,Tuning.StartOmega);
```

It turns out that K_y does fulfill the requirements, while K_n does not. K does also fulfill the requirements, which is not crystal clear from a numerical point of view, as the controller is re-parameterized, another frequency grid might be used, some thresholds may not hold anymore.

9 Misc Functions

Functions, that assist in “trivial” things:

df2cf/cf2df Transform discrete time to continuous time frequencies and vice versa.

dftanalysis Get raw frequency data from time data.

detectextrema Extract resonances from frequency data records.

lti2sys/sys2lti Transform LTI object to (μ Toolbox-) SYSTEM variable and vice versa.

i4cyoula Performs Youla parameterization, used for controller design.

10 Todo List

Basis:

- * object BASISDATA with crosschecks
- * add missing cont time cases
- * add Legendre?
- * MIMO case
- ? add or kill frequency evaluation?

General:

- * check input arguments in functions
- * remove possible zig-zags in detectextrema

Control:

- * check pole location for Youla parameters

ID:

- * extract LS-part from nsse1+2
- * update LS
- * NSSE for discrete time?
- * check usage of IDFRD, (instead of PointEst, see TFPE+NSSE. Combine with SMERROR?)
- * object SMERROR, containing SM ellipsoids and parallelotopes, see SM{E,P}OE

New features:

- * implement ν -gap analysis stuff

References

- [1] J. H. Braslavsky. Model error quantification via non-stationary stochastic embedding: Hairdryer example. CIDAC, Dept of Electrical Engineering, Univ of Newcastle, Callaghan, Australia. Manual for NSSE-package, Mar. 1999.
- [2] J. H. Braslavsky and G. C. Goodwin. A note on non-stationary stochastic embedding for modelling error quantification in the estimation of resonant systems. Technical Report EE99014, CIDAC, Dept of Electrical Engineering, Univ of Newcastle, Callaghan, Australia, Jan. 1999.

- [3] G. Chesi, A. Garulli, A. Tesi, and A. Vicino. Exact bounds for the frequency response of an uncertain plant with ellipsoidal perturbations. In *Proc. of the System Identification Symposium SYSID*, Santa Barbara, CA, USA, June 2000.
- [4] B. R. Fischer. *System Identification in Alternative Shift Operators with Applications and Some Other Topics*. PhD thesis, Control Engineering Group, Luleå University of Technology, 971 87 Luleå, Sweden, Oct. 1999.
- [5] E. Fogel and F. Huang. On the value of information in system identification – bounded noise case. *Automatica*, 18(12):229–238, Dec. 1982.
- [6] A. Garulli, B. Z. Kacewicz, A. Vicino, and G. Zappa. Error bounds for conditional algorithms in restricted complexity set membership identification. *IEEE Trans. on Automatic Control*, 45(1):160–164, 2000.
- [7] A. Garulli and W. Reinelt. On model error modeling in set membership identification. In *Proc. of the System Identification Symposium SYSID*, pages WeMD1–3, Santa Barbara, CA, USA, June 2000.
- [8] G. C. Goodwin, J. H. Braslavsky, and M. M. Seron. Non-stationary stochastic embedding for transfer function estimation. In *Proc. of the 14th IFAC World Congress*, Beijing, China, July 1999.
- [9] G. C. Goodwin, M. Gevers, and B. Ninness. Quantifying the error in estimated transfer functions with application to model order selection. *IEEE Trans. on Automatic Control*, 37(7):913–928, July 1992.
- [10] L. Ljung. Model validation and model error modeling. In B. Wittenmark and A. Rantzer, editors, *Proc. of the Åström Symposium on Control*, pages 15–42, Lund, Sweden, Aug. 1999. Studentlitteratur, Lund, Sweden.
- [11] L. Ljung. *System Identification Toolbox – for use with MATLAB*. The Mathworks Inc., Natick, MA, USA, 2000.
- [12] M. S. Lobo, L. Vandenberghe, and S. Boyd. *Software for second-order cone programming: User’s Guide*. Dept of EE, Stanford University, Stanford, CA, USA, Apr. 1997.
- [13] M. Milanese, J. P. Norton, H. Piet-Lahanier, and E. Walter, editors. *Bounding Approaches to System Identification*. Plenum Press, New York, NY, USA, 1996.
- [14] B. Ninness and F. Gustafsson. A unifying construction of orthonormal bases for system identification. *IEEE Trans. on Automatic Control*, 42(3):515–521, Apr. 1997.
- [15] W. Reinelt. *i4c: Identification for Control Package (version 1.1b5)*. Linköping University, Linköping, Sweden, Sept. 2000. <http://www.wolfgang-reinelt.de/i4c/>.
- [16] W. Reinelt, A. Garulli, and L. Ljung. Comparing different approaches to model error modeling in robust identification. *Automatica*, Aug. 2000. Provisionally accepted for publication.
- [17] W. Reinelt, A. Garulli, L. Ljung, J. H. Braslavsky, and A. Vicino. Model error concepts in identification for control. In *Proc. of the 38th IEEE Conference on Decision and Control*, pages 1488–1493, Phoenix, AZ, USA, Dec. 1999. Invited Session.

- [18] W. Reinelt and L. Ljung. Robust control of identified models with mixed parametric and non-parametric uncertainties. In *Proc. of the European Control Conference*, pages 3564–3569, Porto, Portugal, Sept. 2001.
- [19] P. M. J. Van den Hof, P. S. C. Heuberger, and J. Bokor. System identification with generalized orthonormal basis functions. *Automatica*, 31(12):1821–1834, 1995.
- [20] E. Walter and H. Piet-Lahanier. Estimation of parameter bounds from bounded-error data: a survey. *Mathematics in Computer and Simulation*, 32:449–468, 1990.