MOOSE2: Model Based Optimal Input Design Toolbox for MATLAB® (version 2)

User's Guide

Mariette Annergren
mariette.annergren@ee.kth.se

Christian Larsson
christian.y.larsson@scania.com

April 15, 2016

Contents

Notation

$\det\{X\}$	determinant of a square matrix X
$\mathrm{E}\{x\}$	expected value of x
$\mathscr{E}_{app}(\gamma)$	application ellipsoid evaluated for γ
$\mathscr{E}_{SI}(\alpha)$	system identification ellipsoid evaluated for α
η	process-model parameters
FIR	finite impulse response
$\lambda_{max}\{X\}$	largest eigenvalue of matrix X
$\mathscr{M}$	model
P	covariance matrix of estimated parameters
Φ_x	power spectrum of signal x
q	time-shift operator
$\mathscr{S}$	true system
t	time
θ	process- and noise-model parameters
$\hat{\theta}_N$	estimated process- and noise-model parameters using N data points
θ^0	true process- and noise-model parameters
$\mathrm{trace}\{X\}$	trace of a square matrix X
$V_{app}(\theta)$	application cost function evaluated at θ
ω	frequency
X^*	complex conjugate transpose of matrix X

1 Introduction

1.1 MOOSE2

MOOSE2 is a MATLAB®-based toolbox for solving applications oriented input design problems. The difference between MOOSE2 and MOOSE is that the former is function-based and uses YALMIP, a toolbox for modeling and optimization in MATLAB® [16], to solve the optimization problems while the latter is based on keywords and uses `cvx`, a package for specifying and solving convex programs [7].

This user guide requires some knowledge about optimal input design. For a brief theoretical background, introduction of the notation used and references, see Section 5.

1.2 Problem formulation

The system set-up considered in MOOSE2 is depicted in Figure 1. MOOSE2 handles MIMO systems and closed-loop identification. However, some of the constraints are limited to SISO systems and currently the parameters of the controller F_y are *fixed*.

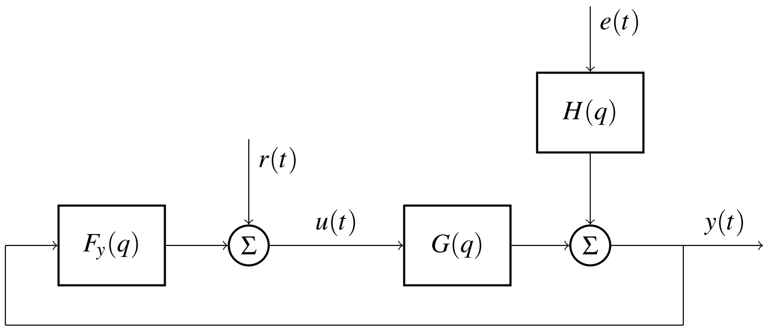

Figure 1: The system set-up. Here, $r(t)$ is the excitation signal, $u(t)$ is the input signal, $e(t)$ is the noise signal, $y(t)$ is the output signal, and $F_y(q)$, $G(q)$ and $H(q)$ are the transfer functions of the system.

The applications oriented input design problem has the following formulation in MOOSE2:

$$\begin{array}{ll} \underset{\Phi_r(\omega)}{\text{minimize}} & a_r \int_{-\pi}^{\pi} \Phi_r(\omega) \mathrm{d}\omega + a_u \int_{-\pi}^{\pi} \Phi_u(\omega) \mathrm{d}\omega + a_y \int_{-\pi}^{\pi} \Phi_y(\omega) \mathrm{d}\omega, \\ \text{subject to} & \text{user constraints.} \end{array}$$

Here, Φ_r is the spectrum of the excitation signal used in the identification experiment. The symbols Φ_u and Φ_y denote the spectra of the input signal u and output signal y, respectively, that originates from the excitation signal r. Note that in open-loop identification, $F_y = 0$, we have $r = u$.

In addition to the objective function above, MOOSE2 supports the classical A-, E- and D-optimality criteria. That is, the minimization of trace$\{P\}$, $\lambda_{max}\{P\}$, det$\{P\}$, respectively, where P is the covariance matrix of the estimated parameters.

The constraints in the input design problem can be any number or combination of

1. *Spectrum constraints:*

$$r_{lb}(\omega) \leq \Phi_r(\omega) \leq r_{ub}(\omega),$$
$$u_{lb}(\omega) \leq \Phi_u(\omega) \leq u_{ub}(\omega),$$
$$y_{lb}(\omega) \leq \Phi_y(\omega) \leq y_{ub}(\omega).$$

2. *Power constraints:*

$$r_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_r(\omega) \mathrm{d}\omega \leq r_{ub}^{pow},$$
$$u_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_u(\omega) \mathrm{d}\omega \leq u_{ub}^{pow},$$
$$y_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_y(\omega) \mathrm{d}\omega \leq y_{ub}^{pow}.$$

3. *Application constraints:*

$$V_{app}(\theta) \leq 1/\gamma_a \text{ with probability } \alpha.$$

4. *Quality (weighted trace) constraints:*

$$\operatorname{trace}(W(\omega)P) \leq \gamma_q \text{ for all } \omega,$$

where

$$W(\omega) = V(\omega)V(\omega)^* \text{ for all } \omega.$$

5. *Ellipsoidal quality constraints:*

$$F(\omega,\eta) \leq \gamma_e \text{ for all } \omega \text{ and } \eta \in \mathscr{E}_{SI},$$

where

$$F(\omega,\eta) = \frac{(W_n G(\eta) + X_n)^* \mathscr{Y}_n (W_n G(\eta) + X_n) + \mathscr{K}_n}{(W_d G(\eta) + X_d)^* \mathscr{Y}_d (W_d G(\eta) + X_d) + \mathscr{K}_d}.$$

1.3 Quick tutorial

This tutorial presents the process of declaring and solving an optimal input design problem in MOOSE2.

1.3.1 Setting up the problem

Consider input design for the system

$$y(t) = G(q,\theta) + e(t),$$

where

$$G(q,\theta) = \theta_1 u(t-1) + \theta_2 u(t-2),$$

with true parameter values $\theta^0 = [10 \ \ -9]$ and noise variance $\mathrm{var}\{e\} = \lambda = 1$. The identification experiment is to be done in open-loop. The objective is to solve the optimization problem

$$\begin{aligned} \underset{\Phi_u(\omega)}{\text{minimize}} \quad & \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_u(\omega)\mathrm{d}\omega \\ \text{subject to} \quad & \mathscr{E}_{SI}(0.95) \subseteq \mathscr{E}_{app}(100) \\ & \Phi_u(\omega) \geq 0, \quad \forall\omega \end{aligned}$$

using an FIR parametrization of the spectrum with 20 lags, that is 20 coefficients in the spectral density function defined in Section 5.3, 100 samples of data and the ellipsoidal relaxation. A MATLAB® implementation of the problem is as follows:

```
%% EXAMPLE OF PROBLEM SETTING
% ----- SYSTEM
theta0              = [10 -9];
A0                  = 1;
B0                  = theta0;
F0                  = 1;
C0                  = 1;
D0                  = 1;
lambda              = 1; % Variance of measurement noise
Ts                  = 1; % Sampling time
trueSystem          = idpoly(A0,B0,C0,D0,F0,lambda,Ts);
% ----- APPLICATION REGION
VappH               = hessian(Vapp,theta0);
gamma               = 100; % Allowed application degradation
alpha               = 0.95; % Confidence level of degradation
% ----- IDENTIFICATION MODEL
model               = trueSystem;
Nident              = 500;
% ----- OPTIMAL INPUT DESIGN PROBLEM
prob                = oidProblem(model,Nident,'FIR',20);
prob.constraints{1} = oidApplicationConstraint(VappH,gamma,alpha);
optH                = solve(prob,[1 0 0]);


% Application cost where the estimate of theta(2) is important
function V = Vapp(theta)
    theta0 = [10 -9];
    V      = norm(theta(2)-theta0(2),2);
end
```

The four blocks of code are:

- SYSTEM: The true model is set, along with its noise variance and sampling time. See Section 2.2.

- APPLICATION REGION: The application set $\mathscr{E}_{app}(100)$ is defined, along with the confidence level α of the application performance degradation. See Section 2.7.3.

- IDENTIFICATION MODEL: The model to be estimated is defined and the number of samples to use in the identification experiment is given. See Section 2.2.

- OPTIMAL INPUT DESIGN PROBLEM: Functions from MOOSE2 are used to set-up and solve the input design problem. The first line creates an instance (prob) of the optimal input design problem. The input spectrum is set to an FIR with 20 lags. The second line adds the set constraint $\mathscr{E}_{SI}(0.95) \subseteq \mathscr{E}_{app}(100)$ to the problem prob. The third line solves the problem prob with the objective function set to input power, that is $\mathrm{E}\{u^2\}$. The optimal spectral factor optH is given which can be used to realize an input signal. See Section 2.5

2 Using the toolbox

2.1 Installation of MOOSE2

To install MOOSE2, download the toolbox catalog and add it to your MATLAB® path. The MATLAB® version must be MATLAB® 8.3 (R2014a) or newer. The toolbox is based on YALMIP, a toolbox for modeling and optimization in MATLAB® [16]. The YALMIP version must be 20130201 or above. The solver used in YALMIP is SDPT3. Thus, you also need to have YALMIP and SDPT3 installed to be able to use MOOSE2. Other solvers, for example SEDUMI, can be used as well but SDPT3 is used in MOOSE2 by default.

To be able to run some of the examples available in the toolbox, you need to download the MATLAB® package “Adaptive Robust Numerical Differentiation” created by John D’Errico [6]. The package contains the function `hessian()`, which is used to define the ellipsoidal approximation of the application cost is the examples.

2.2 Models

Models of the type `idpoly` are supported by MOOSE2. For a detailed description of the `idpoly` model, we refer to the System Identification Toolbox User’s Guide. The `idpoly` model is defined as

$$\mathscr{M}:\ A(q,\theta)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q,\theta)}{D(q,\theta)}e(t), \tag{1}$$

where A, B, C, D and F are polynomials, and q is the forward shift operator. The noise signal e is a white Gaussian process with covariance matrix Λ. The unknown parameter vector is denoted θ.

To declare $\mathscr{M}$ in MOOSE2 you use the command provided by the System Identification Toolbox [15], that is

```
idpoly(A,B,C,D,F,Lambda,Ts)
```

where `Ts` is the sampling time.

MOOSE2 assumes that the model is fully parameterized. By *fully parametrized model* we mean that all coefficients of the A-, B-, C-, D- and F- polynomials in the model $\mathscr{M}$ are unknown and supposed to be estimated. That is, each coefficient is an element in the parameter vector θ. When this is not the case, you can declare which of the parameters in the model are known and which you need to estimate, that is which are *fixed* and which are *free*. The declaration is done as described in the System Identification Toolbox [15]. For instance the model used in Section 1.3 can be declared as

```
% ----- SYSTEM
theta0                       = [10 -9];
A0                           = 1;
B0                           = theta0;
F0                           = 1;
C0                           = 1;
D0                           = 1;
lambda                       = 1; % Variance of measurement noise
Ts                           = 1; % Sampling time
trueSystem                   = idpoly(A0,B0,C0,D0,F0,lambda,Ts);
trueSystem.Structure.b.Free = [1 0]; % Only estimate theta0(1)
```

if we only want to estimate the first element of θ. That is, the first element of θ is declared *free* and the second element is declared *fixed*.

The definition of the application region has to be modified accordingly as well, since the Hessian no longer is a 2-by-2 matrix but a scalar. For example,

```
% ----- APPLICATION REGION
VappH               = 0.01; % Hessian evaluated at true parameter
gamma               = 100; % Allowed application degradation
alpha               = 0.95; % Confidence level of degradation
```

Note however that, when quality and ellipsoidal quality constraints are implemented, a fully parametrized model has to be considered.

2.3 Objective

There are several objective functions supported by MOOSE2. The first is a weighted sum of the power of the excitation signal r, input signal u and output signal y, respectively. That is,

$$a_r \int_{-\pi}^{\pi} \Phi_r(\omega)\mathrm{d}\omega + a_u \int_{-\pi}^{\pi} \Phi_u(\omega)\mathrm{d}\omega + a_y \int_{-\pi}^{\pi} \Phi_y(\omega)\mathrm{d}\omega,$$

where a_r, a_u and a_y are some nonnegative scalars. The objective function is declared when the `solve`-command is called. The second argument correspond to a vector with coefficients a_r, a_u and a_y as elements, that is

```
solve(prob,[a_r,a_u,a_y])
```

In addition, there exists a possibility to separate the objective function into contributions coming from the excitation and noise signal, respectively. That is,

$$a_r \int_{-\pi}^{\pi} \Phi_r(\omega)\mathrm{d}\omega + a_{u_r} \int_{-\pi}^{\pi} \Phi_{u_r}(\omega)\mathrm{d}\omega + a_{u_e} \int_{-\pi}^{\pi} \Phi_{u_e}(\omega)\mathrm{d}\omega + \\ a_{y_r} \int_{-\pi}^{\pi} \Phi_{y_r}(\omega)\mathrm{d}\omega + a_{y_e} \int_{-\pi}^{\pi} \Phi_{y_e}(\omega)\mathrm{d}\omega,$$

where a_r, a_{u_r}, a_{u_e}, a_{y_r} and a_{y_e} are some nonnegative scalars and Φ_{x_z} is the spectrum contribution to Φ_x from signal z. The objective function is then declared as

```
solve(prob,[a_r,a_u_r,a_u_e,a_y_r,a_y_e])
```

MOOSE2 also supports A-, E- and D-optimality criteria. The objective functions for the three criteria are stated in Table 1. The objective functions are declared in MOOSE2 as

```
solve(prob,'A'),  solve(prob,'E'),  solve(prob,'D'),
```

respectively.

Optimality criterion	Objective function
A	minimize trace$\{P\}$
E	minimize $\lambda_{max}\{P\}$
D	minimize det$\{P\}$

Table 1: Objective function for A-, E- and D-optimality criteria. The matrix P is the covariance matrix of the estimated parameters.

2.4 Decision variables

The decision variable in the input design problem is the spectrum of the excitation signal r used in the identification problem. To get a tractable optimization problem we use finite dimensional parametrization of the spectrum. So, more specifically, the decision variables are the spectrum parameters used to describe the excitation spectrum in the finite dimensional setting. The spectrum type and number of spectrum parameters used are defined by the user as described in Section 2.6.

2.5 Optimization problem

A problem instance of the sort described in Section 1.2 is constructed by any of the following commands:

```
oidProblem(MODEL)
oidProblem(MODEL,N)
oidProblem(MODEL,N,type,M)
oidProblem(MODEL,N,type,M,C,cParam)
```

The arguments are

- `MODEL`: Model of the system, see Section 2.2.
- `N`: Number of samples to use in the identification experiment.
- `type`: Type of the input spectrum, see Section 2.6. (Currently MOOSE2 only supports MA (FIR) spectra.)
- `M`: Number of lags or spectrum parameters, see Section 2.6.
- `C`: Controller for closed-loop design, see Section 2.8.
- `cParam`: Set to `fixed` for a fixed controller or `free` to include the controller parameters in the design, see Section 2.8. (Currently MOOSE2 only supports `fixed` controllers.)

Default values of the arguments are

```
N=1, type='MA', M=20, C=0, cParam='fixed' .
```

2.6 Spectrum

Only one type of input spectrum is supported by MOOSE2 at the moment. It is the MA (FIR) spectrum, see Section 5.3. The spectrum is declared when calling `oidProblem` in the following way:

```
oidProblem(model,Nident,'MA',20)
```

The third argument declares the type of spectrum to use (you can also write `'FIR'`). The fourth argument declares how many lags to use, that is how many spectrum parameters to use. Here, it is set to 20.

2.7 Constraints

There are five different constraints available in MOOSE2. They are, as stated in Section 1.2, spectrum, power, application, quality (weighted trace) and ellipsoidal quality constraints.

The three latter constraints are declared as specific constraint-instances, say `con`. The constraints can then be added to the constraints property of the `oidProblem`-instance `prob` by writing

```
prob.constraints{i} = con
```

where `i` is a nonnegative scalar specifying the *i*th constraint.

In general, all constraints are implemented and solved using either the KYP lemma or a frequency grid, see Section 5.5. However, it is the experience of the authors (and has been noted by others as well) that using the KYP lemma is not numerically robust. For small problems, say single-input-single-output systems with a handful parameters and spectrum coefficients, the KYP implementation works well, but in general it is recommended to use a well-chosen frequency grid instead when evaluating the different constraints. See Section 4, for some related comments.

We describe how to declare the constraints using MOOSE2 in the following.

2.7.1 Spectrum constraints

We can impose upper and lower bounds on the excitation, input and output spectra. That is,

$$\begin{aligned} r_{lb}(\omega) &\leq \Phi_r(\omega) \leq r_{ub}(\omega), \\ u_{lb}(\omega) &\leq \Phi_u(\omega) \leq u_{ub}(\omega), \\ y_{lb}(\omega) &\leq \Phi_y(\omega) \leq y_{ub}(\omega). \end{aligned}$$

Once an instance of `oidProblem` has been created, say `prob`, we can specify the bounds by writing

```
prob.spectrum.signal.ub = signal_ub
prob.spectrum.signal.lb = signal_lb
```

The constraints are imposed on the excitation, input or output signal if `signal` is exchanged with `excitation`, `input` or `output`, respectively. The first line declares a frequency dependent upper bound on the spectrum. The second line declares a frequency dependent lower bound on the spectrum.

The bounds can be given in three different formats. They are state-space or transfer functions, structure arrays or matrices. If

```
prob.spectrum.input.ub = input_ub
```

is declared, with `input_ub` being a linear time invariant model in state-space or transfer function format, the following Hermitian linear matrix inequality is imposed:

$$\Phi_u(\omega) \leq u_{ub}(\omega)u_{ub}(\omega)^* \text{ for all } \omega,$$

where $u_{ub}(\omega) =$ `input_ub`. If `input_ub` instead is defined as the structure array

```
input_ub=struct('B',freqResp,'w',freq,'type','lmi'),
```

the constraint becomes the Hermitian linear matrix inequality

$$\Phi_u(\omega) \leq u_{ub}(\omega),$$

where $u_{ub}(\omega) =$ `freqResp` and $\omega =$ `freq`. The inequality is imposed element-wise if the structure field `type` is left empty or not set to `lmi`. The real and imaginary part of each element are treated as separate inequalities. If `input_ub` is defined as a matrix, the constraint is a Hermitian linear matrix inequality with a constant upper bound with respect to ω,

$$\Phi_u(\omega) \leq u_{ub} \text{ for all } \omega.$$

2.7.2 Power constraints

We can impose lower and upper bounds on all signal powers. That is,

$$r_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_r(\omega)\mathrm{d}\omega \leq r_{ub}^{pow},$$
$$u_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_u(\omega)\mathrm{d}\omega \leq u_{ub}^{pow},$$
$$y_{lb}^{pow} \leq \int_{-\pi}^{\pi} \Phi_y(\omega)\mathrm{d}\omega \leq y_{ub}^{pow}.$$

We specify the bounds by writing

```
prob.spectrum.signal.power.ub = signal_pow_ub
prob.spectrum.signal.power.lb = signal_pow_lb
```

As for spectrum constraints, the bounds are imposed on the excitation, input or output signal if `signal` is exchanged with `excitation`, `input` or `output`, respectively.

2.7.3 Application constraints

The application constraint is defined using a function $V_{app}(\theta) : \mathbb{R}^n \mapsto \mathbb{R}$ (the application cost) and scalars γ_a and α. The constraint is

$$V_{app}(\theta) \leq 1/\gamma_a \text{ with probability } \alpha, \tag{2}$$

see Section 5.5.1. Since (2) might not be a convex constraint, we approximate it with a convex constraint using the scenario approach or ellipsoidal approximation, see Section 5.5.1.

The application constraints are declared using the command

```
con=oidApplicationConstraint(VappH,gamma_a,alpha).
```

The first argument is either a fat matrix with all the scenarios collected column-wise and the last row of each column is the value of the application cost evaluated at the corresponding scenario or the Hessian of the application cost evaluated at the true parameters θ^0. The former case uses the scenario approach and the latter case uses the ellipsoidal approximation. The second argument is the bound γ_a and the third argument is the probability α.

Note that MOOSE2 uses a transfer function model of the type `idpoly` in the input design. Meaning, the first argument `VappH` must be formed in accordance to the parameter ordering of `idpoly`, see the System Identification Toolbox [15]. Consequently, if `VappH` has been calculated based on a state-space model, a suitable transformation has to be made before it is used in MOOSE2.

For details regarding the application constraints, see, for example, [1, Ch. B.2.2], [11, Ch. 2.3, 2.6, 3.3 and 3.4] and the references therein.

2.7.4 Quality constraints

The quality constraints are weighted trace constraints given on the form

$$\operatorname{trace}(W(\omega)P) \leq \gamma_q \text{ for all } \omega, \tag{3}$$

where

$$W(\omega) = V(\omega)V(\omega)^* \text{ for all } \omega,$$

and P is the covariance matrix of the estimated parameters, see Section 5.2.

Two quality constraints that can be represented in the form of (3) are

$$\left\| \left| \frac{T(\omega)}{G_0(\omega)} \right|^2 \frac{\mathrm{d}G^*(\omega,\theta_0)}{\mathrm{d}\theta} P \frac{\mathrm{d}G(\omega,\theta_0)}{\mathrm{d}\theta} \right\|_2 \leq \gamma_q, \tag{4}$$

$$\left\| \left| \frac{T(\omega)}{G_0(\omega)} \right|^2 \frac{\mathrm{d}G^*(\omega,\theta_0)}{\mathrm{d}\theta} P \frac{\mathrm{d}G(\omega,\theta_0)}{\mathrm{d}\theta} \right\|_\infty \leq \gamma_q, \tag{5}$$

If the quality constraint (3) is on the form (4) or (5), we can declare it directly in MOOSE2. We declare it directly in the sense that we only need to define $T(\omega)$ and not $V(\omega)$. However, this direct declaration is only supported for SISO systems.

The quality constraint can be defined using the commands

```
con = oidQualityConstraint(V)
con = oidQualityConstraint(V,gamma_q)
con = oidQualityConstraint(V,gamma_q,wSamp)
con = oidQualityConstraint(T,gamma_q,wSamp,norm)
```

The first line defines constraint (3) with $\gamma_q = 1$. The second line defines constraint (3) with an arbitrary $\gamma_q \geq 0$. Both commands can be applied to MIMO systems. The third line gives the option to change how the infinite constraint (3) is transformed to a finite dimensional constraint. The argument `wSamp` can be a vector of frequency points where (3) is to be evaluated or the string `'KYP'` which uses the KYP lemma for evaluating (3). For details on how to apply the KYP lemma to quality constraints, see [9, Ch. 3.6.2]. By default, a uniform frequency grid on $[0,\pi[$ is used with 20 frequency points. This is found to be much more numerically robust than using KYP lemma. The fourth line defines constraint (4) if `norm` is set to `2` and constraint (5) if `norm` is set to `inf`. If declaring constraint (4), the argument `wSamp` is obsolete and can be left empty (`[]`) since neither KYP lemma nor a frequency grid is used in the evaluation. MOOSE2 instead calls the MATLAB®-function `covar` to evaluate the left hand side of the inequality in (4).

Note that MOOSE2 only allows for fully parametrized models when evaluating quality constraints.

For a brief theoretical summary of the quality constraints described here see Section 2.7.4. For a complete background see [9, Ch. 3.6] and the references therein.

2.7.5 Ellipsoidal quality constraints

The ellipsoidal quality constraints are given on the form

$$F(\omega,\eta) \leq \gamma_e \text{ for all } \omega \text{ and } \eta \in \mathscr{E}_{SI}(\alpha), \tag{6}$$

where

$$F(\omega,\eta) = \frac{(W_n G(\omega,\eta)+X_n)^* \mathscr{Y}_n (W_n G(\omega,\eta)+X_n) + \mathscr{K}_n}{(W_d G(\omega,\eta)+X_d)^* \mathscr{Y}_d (W_d G(\omega,\eta)+X_d) + \mathscr{K}_d}.$$

The inequality (6) has to be satisfied in the ellipsoid $\mathscr{E}_{SI}(\alpha)$ for all ω. The ellipsoidal quality constraints are only supported for SISO systems with polynomial $A(q,\eta)$ and $F(q,\eta)$. The parameter vector η consists of the parameters present in the transfer function from the input to output, $G(\omega,\eta)$. That is, the parameters in the noise model are not included.

The ellipsoidal quality constraint is declared using the commands

```
  con =oidEllipsoidalQualityConstraint(Kd,Kn,Wd,Wn,Xd,...
        Xn,Yd,Yn,gamma_e,alpha)
  con =oidEllipsoidalQualityConstraint(Kd,Kn,Wd,Wn,Xd,...
        Xn,Yd,Yn,gamma_e,alpha,wSamp)
```

The first eight arguments define $F(\omega,\eta)$. The ninth argument `gamma_e` is the upper bound on $F(\omega,\eta)$ and the tenth argument is the probability α, see (6). The argument `wSamp` can be a vector of frequency points where (6) is to be evaluated or the string `'KYP'` which uses the KYP lemma for evaluating (6). For details on how to apply the KYP lemma to quality constraints, see [9, Ch. 3.7.2]. The constraint is evaluated using a frequency grid. By default, a uniform frequency grid on $[0,\pi]$ is used with 20 frequency points.

Note that MOOSE2 only allows for fully parametrized models when evaluating ellipsoidal quality constraints.

For a short theoretical background of the ellipsoidal quality constraints described in this section see Section 5.5.3. For a rigorous background see [9, Ch. 3.7] and the references therein.

2.8 Controller

The controller needs to be specified as a linear time invariant transfer function. Currently, MOOSE2 only accepts controllers with *fixed* parameters. That is, parameters that are not to be included as decision variables in the input design problem.

2.9 Spectral factorization

Once a MOOSE2 problem has been solved you can get the stable minimum phase spectral factor of the optimal excitation spectrum. The spectral factor can be obtained by calling

```
optH = solve(prob,[1,0,0])
```

Here, `optH` is the spectral factor.

2.10 Solution

The outputs of the MATLAB®-command `solve` contains specific information about your problem and its solution. The following command extracts all outputs of `solve`:

```
[optH,info,iF,optVal,signalPow,c,con] =...
                                       solve(prob,[1,0,0])
```

Here, `optH` is the spectral factor, `info` is a structure array containing the information given by YALMIP about the problem and the solution, `iF` is the information matrix, `optVal` is the objective value, `signalPow` is a structure array containing the power of all signals, `c` are the spectrum parameters and `con` are the constraints evaluated at the optimal solution.

2.11 Solver

The default solver used in MOOSE2 is SDPT3. However, the user can specify all sdp-settings available, including a different solver. This is done in accordance with the `yalmip`-manual, by defining a structure array `opts` with the desired setting and calling `solve` in the following way:

```
optH = solve(prob,[1,0,0],opts)
```

2.12 Help functions

Currently only one *help function* is included in MOOSE2 and it is the MATLAB®-function `ellipse()`. The function plots an ellipse and its center point given a matrix, a point and the desired color of the plot. The function is used when displaying the results from some of the examples given in the toolbox.

3 Implementation

The implementation of MOOSE2 uses the object-oriented programming capabilities of MATLAB®. The structure of the implementation is presented in Figure 2. The design is built around a predefined set of interfaces for the necessary components of the optimal input design problem.

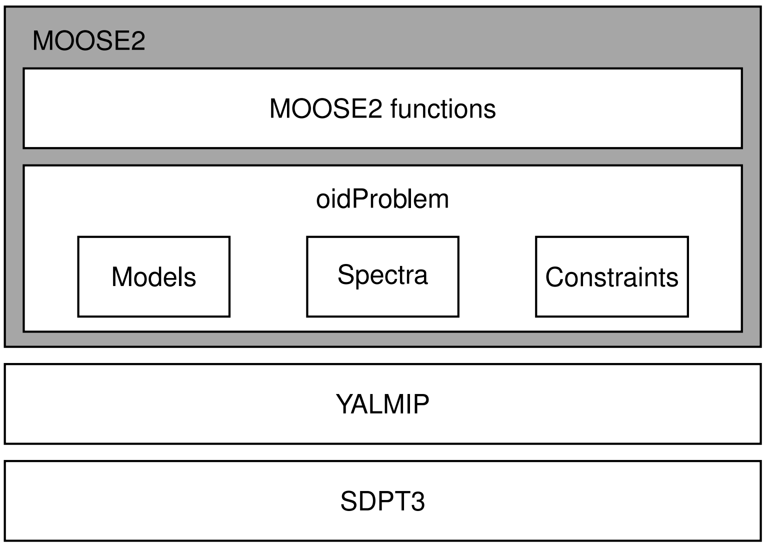

Figure 2: The structure of the MOOSE2 implementation. The user interacts with the top layer through functions. The lower layers of MOOSE2 are used to define and store the optimal input design problem. MOOSE2 relies on YALMIP and SDPT3 to solve the optimization problem.

User interaction with MOOSE2 is done through a set of functions.

The central object is the class `oidProblem` where the input design problem is stored. Every instance of `oidProblem` contains a model-, spectrum-, and one or more constraint-instances.

Abstract classes are used for the model, spectrum and constraint classes to define interfaces. This allows for easy implementation of new models, spectra and constraint classes.

There is no optimization implemented in MOOSE2. Instead the toolbox relies on external SDP solvers for solving the defined optimization problem. MOOSE2 is based on YALMIP, a toolbox for modeling and optimization in MATLAB® [16]. The default solver used in YALMIP is set to SDPT3 [18].

4 Future work

To further develop MOOSE2, we have defined three main areas where future work is needed. These areas are

1. numerical stability of implementation,
2. controller design,
3. multiple choices of input spectra.

The first area concerns numerical stability of the implementation. Based on the authors experiences, the solution found in MOOSE2 using a frequency grid can be quite sensitive in terms of the grid chosen. Currently, there is no method implemented in MOOSE2 that aids the user in choosing a frequency grid in an intelligent way nor is there a warning installed to make the user aware of when numerical issues can occur.

In addition, the authors have also experienced numerical stability issues when using the KYP lemma. The linear matrix inequalities that occur when applying the KYP lemma to the problems in MOOSE2 tend to be quite large even for small system sizes. Consequently, solving these linear matrix inequalities become the bottle neck of solving the optimal input design problem. Also, we encountered numerical issues when solving them. One remedy to the problems related to the KYP lemma might be to reformulate the problem structure in an intelligent way and send it directly to the solver (`sdpt3`) or to actually tailor-make a specific solver for the considered problems. Another idea is to solve the problem for an adaptive frequency grid in a sequential manner instead. However, currently, the author's advice is to use a well-chosen frequency grid when encountering problem with the KYP lemma.

The second area concerns controller design. That is, to enable the design of the controller parameters as well as the spectrum parameters in MOOSE2.

The third area concerns multiple choices of input spectra. Currently we only allow MA (FIR) spectra. We would like to extend this to autoregressive, Laguerre and multisine parametrization.

5 Theoretical background

We give a short theoretical background to concepts used in MOOSE2.

The objective with application-oriented input design is to deliver a model that, when used in the intended application, results in an acceptable performance. This is achieved by constructing a particular input signal to be used in the system identification experiment. The obtained model is highly dependent on the input signal used. Thus, by designing the input we are in a way designing the model obtained from the identification experiment.

5.1 Dynamic system and model

We consider a multivariate discrete time linear time invariant dynamic system. The dynamic system can be approximated by a parametrized model. Given a structure, the model response is expressed as

$$\mathscr{M}(\theta): \quad y(t) = G(q,\theta)u(t) + v(t), \tag{7a}$$
$$v(t) = H(q,\theta)e(t), \tag{7b}$$

where $y(t)$ is the output signal, $u(t)$ is the input signal, $e(t)$ is the white Gaussian noise signal and $v(t)$ is the filtered noise signal. The transfer functions G and H are parameterized by $\theta \in \mathbb{R}^n$ and q is the forward-shift operator.

The true system is assumed to be parametrized by the same structure as the model. Thus, there exist parameters θ^0 such that the true output response of the system can be written as

$$\begin{aligned}\mathscr{S}: \quad y(t) &= G(q,\theta^0)u(t) + v_0(t),\\ v_0(t) &= H(q,\theta^0)e(t).\end{aligned}$$

The unknown parameters are θ, the true values of the parameters are θ^0 and the estimated parameters based on N observations are $\hat{\theta}_N$. The observations consist of observed output and input signal sequences, $Z^N = \{y(t),u(t)\}_{t=1}^N$.

5.2 Prediction error method

The prediction error method (PEM) [14] is a method of identifying the unknown parameters θ in the model (7). The parameter estimates are found by minimizing a criterion function of the prediction error with respect to θ. The prediction error is defined as the difference between the output of the true system and the output predicted by the model. Based on the model structure (7), the one-step-ahead predicted output of the system is

$$\hat{y}(t|\theta) = H^{-1}(q,\theta)G(q,\theta)u(t) + \left[I - H^{-1}(q,\theta)\right]y(t).$$

Consequently, the one-step-ahead prediction error becomes

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta) = H^{-1}(q,\theta)\left[y(t) - G(q,\theta)u(t)\right].$$

The criterion function to be minimized is denoted $V_N(\theta, Z^N)$. The estimates are defined as

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta, Z^N). \tag{8}$$

In MOOSE2 the criterion function is set to the quadratic criterion. That is,

$$V_N(\theta, Z^N) = \frac{1}{2N} \sum_{t=1}^{N} \varepsilon(t,\theta)^T \Lambda^{-1} \varepsilon(t,\theta). \tag{9}$$

It holds under mild conditions that the estimated parameters converge to the true values with probability one as the number of observations tends to infinity [14]. We also have, under the same conditions, that the sequence of random variables

$$N(\hat{\theta}_N - \theta^0)^T V_N''(\theta^0, Z^N)(\hat{\theta}_N - \theta^0)$$

converges in distribution to the χ^2-distribution with n degrees of freedom [14]. Thus, for a sufficiently large N, the estimates $\hat{\theta}_N$ are with a probability α contained inside the ellipsoid

$$\mathscr{E}_{SI}(\alpha) = \left\{ \theta \,\middle|\, (\theta - \theta^0)^{\mathrm{T}} V_N''(\theta^0, Z^N)(\theta - \theta^0) \le \frac{\chi^2_\alpha(n)}{N} \right\}, \tag{10}$$

where $\chi^2_\alpha(n)$ is the α-percentile of the χ^2-distribution with n degrees of freedom. We call $\mathscr{E}_{SI}$ the system identification set.

The Hessian of the quadratic criterion function in (9) is

$$V_N''(\theta^0, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \hat{y}'(t|\theta^0) \Lambda^{-1} \hat{y}'(t|\theta^0)^T. \tag{11}$$

We can rewrite expression (11) in the frequency domain using Parseval's relation. We get different expressions depending on if we perform open- or closed-loop identification. Open-loop identification is when there is no feedback control of the system during the identification experiment, that is, $F_y = 0$ in Figure 1. Closed-loop identification is when there is feedback control ($F_y \neq 0$ in Figure 1). MOOSE2 handles both cases. However, in closed-loop identification the parameters of the controller are assumed to be *fixed*. We have

$$\begin{aligned} V_N''(\theta^0, Z^N) = & \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_r(e^{i\omega}, F_y(e^{i\omega})), \theta^0)(\Lambda^{-1} \otimes \Phi_r(e^{i\omega}))\Gamma_r^*(e^{i\omega}), F_y(e^{i\omega}), \theta^0) d\omega + \\ & \frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_e(e^{i\omega}, F_y(e^{i\omega}), \theta^0)(\Lambda^{-1} \otimes \Lambda(e^{i\omega}))\Gamma_e^*(e^{-i\omega}, F_y(e^{i\omega}), \theta^0) d\omega, \end{aligned} \tag{12a}$$

where

$$\Gamma_r = \begin{bmatrix} vec[F_r^1] \\ \vdots \\ vec[F_r^n] \end{bmatrix}, \; \Gamma_e = \begin{bmatrix} vec[F_e^1] \\ \vdots \\ vec[F_e^n] \end{bmatrix}, \tag{12b}$$

$$F_r^i = H^{-1}(\theta^0) \frac{dG(\theta)}{d\theta_i} \left(I + F_y G(\theta^0)\right)^{-1}, \tag{12c}$$

$$F_e^i = H^{-1}(\theta^0) \left(\frac{dH(\theta)}{d\theta_i} - \frac{dG(\theta)}{d\theta_i} \left(I + F_y G(\theta^0)\right)^{-1} F_y H(\theta^0) \right), \text{ for all } i = 1 \ldots n. \tag{12d}$$

Here $\otimes$ is the Kronecker product, θ_i denotes the ith component of the vector θ and $vec[X]$ denotes a row vector containing the rows of the matrix X stacked on top of each other. For details, see [2]. From expression (12), we can see that the Hessian is an affine function of the excitation spectrum $\Phi_r(\omega)$. With some abuse of notation and language, we will in the rest of text only consider the input signal u and input spectrum $\Phi_u(\omega)$. If the system set-up is in closed loop these entities should be exchanged by excitation signal r and excitation spectrum $\Phi_r(\omega)$.

Thus, we can directly affect the shape of the system identification set and, consequently, the estimates by designing a particular spectrum of the input signal.

The system identification set in (10) can be expressed using the Fisher information matrix instead of the Hessian. The information matrix is defined as

$$\mathbf{I}_F = \frac{1}{N}\sum_{t=1}^{N} \hat{y}'(t|\theta^0)\Lambda^{-1}\hat{y}'(t|\theta^0)^T, \tag{13}$$

[14]. Thus, we can write

$$\mathscr{E}_{SI}(\alpha) = \left\{\theta \;\middle|\; (\theta-\theta^0)^{\mathrm{T}}\mathbf{I}_F(\theta^0)(\theta-\theta^0) \le \frac{\chi^2_\alpha(n)}{N}\right\}.$$

PEM can also be used with state space formulations of the system and model. For details, see [12].

The covariance matrix of the estimated parameters is defined as $P = N\mathbf{I}_F^{-1}$.

5.3 Spectrum of signals

We saw in the previous section that the spectrum of the input signal used in the identification experiment affects the estimates. Thus, input design can be performed in terms of its frequency characteristics by choosing the spectrum of the signal. The spectral density of a stationary signal u can be written as

$$\Phi_u(\omega) = \sum_{k=-\infty}^{\infty} c_k \mathscr{B}_k(e^{i\omega}), \tag{14}$$

where the scalar basis functions $\{\mathscr{B}_k(e^{i\omega})\}_{k=0}^{\infty}$ are proper, stable, and rational such that $\mathscr{B}_{-k}(e^{i\omega}) = \mathscr{B}_k(e^{-i\omega})$ and the real coefficients $c_{-k} = c_k^T$. MOOSE2 currently only handles spectra that are shaped as an FIR filter. That is, the basis functions are exponentials, $\mathscr{B}_k(e^{i\omega}) = e^{-i\omega k}$. Consequently, the coefficients become the autocovariance sequence of the input signal. That is, $c_k = \mathrm{E}\left\{u(t)u(t-k)^T\right\}$, see for example [17].

Some optimal input design problems can be formulated as convex optimization problems with decision variables c_k. The design is then a matter of finding the coefficients c_k. There are two main difficulties with choosing them. First, the spectral density of a stationary process is a non-negative entity. Therefore, the coefficients must be chosen such that

$$\Phi_u(\omega) \succeq 0, \text{ for all } \omega, \tag{15}$$

for (14) to define a spectral density. Second, the constraint (15) is infinite dimensional making it computationally impractical to work with. To simplify the problem, we consider the partial expansion

$$\Phi_u(\omega) = \sum_{k=-(m-1)}^{m-1} c_k \mathscr{B}_k(e^{i\omega}). \tag{16}$$

Hence, only the first m coefficients of (14) are used to define the spectrum. Two approaches for choosing the coefficients c_k are partial correlation parameterization [10] and finite dimensional parameterization. MOOSE2 uses the latter approach.

5.3.1 Finite dimensional parameterization

Finite dimensional parameterization requires that $\{c_k\}_{k=0}^{m-1}$ is chosen such that (16) is a spectrum. It means that condition (15) must hold for the truncated sum (16). This can be achieved in various ways, the most frequently used technique is an application of the positive real lemma which springs from the Kalman-Yakubovich-Popov (KYP) lemma.

Lemma 1: If $\{A,\ B,\ C,\ D\}$ is a controllable state-space realization of $\Phi_r^+(\omega) = \sum_{k=0}^{m-1} c_k \mathscr{B}_k(e^{i\omega})$. Then there exists a matrix $Q = Q^T$ such that

$$K(Q,\{A,B,C,D\}) \triangleq \begin{bmatrix} Q - A^T QA & -A^T QB \\ -B^T QA & -B^T QB \end{bmatrix} + \begin{bmatrix} 0 & C^T \\ C & D + D^T \end{bmatrix} \succeq 0, \qquad (17)$$

if and only if $\Phi_u(\omega) = \sum_{k=-(m-1)}^{m-1} c_k \mathscr{B}_k(e^{i\omega}) \geq 0,$ for all ω.

Thus, the necessary and sufficient condition for (15) to hold for the truncated sequence is the matrix inequality (17), assuming a matrix Q exists. The matrix inequality becomes a linear matrix inequality (LMI) in c_k and Q if the only matrices that are linearly dependent on the coefficients c_k are C and D.

5.4 Input generation

When the input spectrum is found, a corresponding time realization of the signal has to be generated. The realization is then used to excite the system in the identification experiment. One possible input generation is to let the input signal be white Gaussian noise filtered through a transfer function matrix. The matrix is chosen such that the obtained signal has the required spectrum. The matrix design is a problem of minimum phase spectral factorization, as such, it has many known solutions, see for example [17]. MOOSE2 does not provide tools for input generation.

5.5 Input design constraints

Three different classes of input design constraints are discussed. They are related to the application and quality of the model.

5.5.1 Application constraints

The input signal needs to be designed with the intended application of the model in mind. To enable this, a measure of how well the model performs is defined. The degradation in performance due to a mismatch between the model and the system is specified by an application cost function. The cost emphasizes an important performance quality of the system. Examples of such qualities are the sensitivity function and the closed-loop output response. The cost function is denoted $V_{app}(\theta)$. The minimal value of V_{app} is equal to zero and is achieved when the true parameters are used in the function. If $V_{app}(\theta)$ is twice differentiable in a neighborhood of θ^0, these conditions are equivalent to the constraints $V_{app}(\theta^0) = 0$, $V'_{app}(\theta^0) = 0$ and $V''_{app}(\theta^0) \succeq 0$.

An increased value of the application cost reflects an increased degradation in performance. The maximal allowed degradation is defined by

$$V_{app}(\theta) \leq \frac{1}{\gamma}, \tag{18}$$

where γ is a positive scalar. The parameters fulfilling inequality (18) are called acceptable parameters and they belong to the application set. The set is defined as

$$\Theta_{app}(\gamma) = \left\{ \theta \;|V_{app}(\theta) \leq \frac{1}{\gamma} \right\}.$$

The concept of using application sets comes from [8] and [3].

We want the estimated parameters to give a model with acceptable performance. Thus, one part of the objective of optimal input design is to guarantee, with high probability, that the estimated parameters are acceptable parameters. This condition is equivalent to requiring that

$$\mathscr{E}_{SI}(\alpha) \subseteq \Theta_{app}(\gamma), \tag{19}$$

for specific values of α and γ. The constraint (19) is not necessarily convex, consequently it needs to be approximated by one. Two methods of doing this are the scenario approach and the ellipsoidal approximation. MOOSE2 supports both methods.

Scenario approach The requirement that the system identification set lies inside the application set is relaxed. It is enough that a finite number of the estimated parameters are contained inside Θ_{app}. These parameters are chosen from Θ_{app}, preferably close to its boundary, according to a given probability distribution. It is shown in [5] that if

$$(\theta_i - \theta^0)^{\mathrm{T}} \mathbf{I}_F (\theta_i - \theta^0) \geq \frac{\chi^2_\alpha(n)\gamma}{N} V_{app}(\theta_i) \text{ for } i = 1 \dots M < \infty,$$

where $\theta_i \in \Theta_{app}$, then $\mathscr{E}_{SI}$ lies inside Θ_{app} with a high probability. For more details, see for example [13].

Ellipsoidal approximation The application cost is approximated by its second order Taylor expansion centered around the true parameters. The corresponding application set becomes an ellipsoidal region. Thus, $\Theta_{app} \approx \mathscr{E}_{app}$ for θ close to θ^0, where

$$\mathscr{E}_{app}(\gamma) = \{\theta \mid (\theta - \theta^0)^{\mathrm{T}} V''_{app}(\theta^0)(\theta - \theta^0) \leq 1/\gamma\}. \tag{20}$$

It is shown in [8] that $\mathscr{E}_{SI}$ lies inside $\mathscr{E}_{app}$ if and only if $\mathbf{I}_F \succeq \chi^2_\alpha(n)\gamma V''_{app}(\theta^0)/N$. Thus, constraint (19) can be approximated as

$$\mathbf{I}_F \succeq \chi^2_\alpha(n)\gamma V''_{app}(\theta^0)/N. \tag{21}$$

For more details, see for example [8].

5.5.2 Quality constraints

One, of several, performance criteria of input design is the weighted trace constraint

$$\operatorname{trace}(W(\omega)P) \leq \gamma_q \text{ for all } \omega,$$

where

$$W(\omega) = V(\omega)V(\omega)^* \text{ for all } \omega,$$

where P is the inverse of the Fisher's information matrix $\mathbf{I}_F$. An example of a quality constraint is given in Example 5.1.

Example 5.1 *Consider the classical robust stability condition*

$$\|\Delta(\omega,\theta)\|_\infty < 1, \tag{22}$$

with

$$\Delta(\omega,\theta) = T(\omega)\frac{G(\omega,\theta^0) - G(\omega,\theta)}{G(\omega,\theta)},$$

and $T(\omega)$ equal to the complementary sensitivity function. A way of limiting the discrepancy obtained due to model mismatch is to limit the variance of $\Delta(\omega,\theta)$. The variance constraint can be expressed as

$$\|Var\,\Delta(\omega,\theta)\|_\infty < 1,$$

with

$$Var\,\Delta(\omega,\theta) \approx |T(\omega)/G_0(\omega)|^2 \frac{dG^*(i\omega,\theta_0)}{d\theta} P \frac{dG(\omega,\theta_0)}{d\theta},$$

which in turn can be formulated as a weighted trace constraint. We simply set $W(\omega)$ equal to

$$|T(\omega)/G_0(\omega)|^2 \frac{dG(i\omega,\theta_0)}{d\theta}\frac{dG^*(\omega,\theta_0)}{d\theta},$$

see [9, Ex. 3.12].

In MOOSE2, quality constraints are evaluated either by sampling the frequency range or by applying the KYP lemma. For more details, see [9, Ch. 3.6].

5.5.3 Ellipsoidal quality constraints

In ellipsoidal quality constraints, we require that the quality constraints are fulfilled for models within the system identification set $\mathscr{E}_{SI}$. That is,

$$F(\omega,\eta) \le \gamma_e \text{ for all } \omega \text{ and } \eta \in \mathscr{E}_{SI}(\alpha), \tag{23}$$

where

$$F(\omega,\eta) = \frac{(W_nG(\omega,\eta)+X_n)^*\mathscr{Y}_n(W_nG(\omega,\eta)+X_n)+\mathscr{K}_n}{(W_dG(\omega,\eta)+X_d)^*\mathscr{Y}_d(W_dG(\omega,\eta)+X_d)+\mathscr{K}_d}.$$

The parameter vector η consists of the parameters only present in the transfer function from the input to output, $G(\omega,\eta)$. That is, the parameters in the noise model are not included. An example of ellipsoidal quality constraints is given in Example 5.2.

Example 5.2 *Consider once again the classical robust stability criterion* (22) *in Example 5.1. Instead of approximating the criterion with a variance constraint, we require it to be fulfilled within the system identification set. That is*

$$\max_{\omega,\theta\in\mathscr{E}_{SI}} |\Delta|^2 \le \gamma_e. \tag{24}$$

To capture (24) *within the given framework, we simply set* $W_n(\omega) = W_d(\omega) = 1$*,* $X_n = -G(\theta^0)$*,* $\mathscr{Y}_n = T^*T$ *and* $\mathscr{K}_n = \mathscr{K}_d = X_d = 0$ *which gives*

$$F(\omega, \eta) = \frac{|G(\eta) - G(\eta^0)|^2 |T|^2}{|G(\eta)|^2} = |\Delta|^2,$$

see [9, Ex. 3.13].

In MOOSE2, ellipsoidal quality constraints are evaluated either by sampling the frequency range or by applying the KYP lemma. For more details, see [9, Ch. 3.7].

5.6 Optimal input design problem

We want the estimated parameters to give a model with acceptable performance in terms of application and quality. Let f_{cost} denote the cost related to the experiment. The complete objective of optimal input design can then be stated as the optimization problem

$$\begin{aligned}
\underset{c_k}{\text{minimize}}\ & f_{cost}(c_k), \\
\text{subject to}\ & \text{spectrum constraints}, \\
& \text{power constraints}, \\
& \text{application constraints}, \\
& \text{quality constraints}, \\
& \text{ellipsoidal quality constraints}.
\end{aligned}$$

The optimization problem can be approximated by a convex formulation using scenario approach, ellipsoidal approximation, sampled frequency range and the KYP lemma. The benefit is that a convex optimization problem can be solved accurately and efficiently [4].

References

[1] M. Annergren. ADMM for l_1 regularized optimization problems and applications oriented input design for MPC, 2012.

[2] M. Barenthin Syberg. *Complexity Issues, Validation and Input Design for Control in System Identification*. PhD thesis, KTH Royal Institute of Technology, December 2008. TRITA-EE 2008:055.

[3] X. Bombois, G. Scorletti, M. Gevers, P. M. J. Van Der Hof, and R. Hildebrand. Least costly identification experiment for control. *Automatica*, 42:1651–1662, 2006.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

[5] G. C. Calafiore and M. C. Campi. The Scenario Approach to Robust Control Design. *IEEE Transactions on Automatic Control*, 51(5):742–753, May 2006.

[6] J. D'Errico. Adaptive robust numerical differentiation. http://www.mathworks.com/matlabcentral/fileexchange/13490-adaptive-robust-numerical-differentiation, June 2011.

[7] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, April 2011.

[8] H. Hjalmarsson. System identification of complex and structured systems. *European Journal of Control*, 15(34):275 – 310, 2009.

[9] H. Jansson. *Experiment design with applications in identification for control*. PhD thesis, KTH Royal Institute of Technology, 2004.

[10] H. Jansson and H. Hjalmarsson. Input Design via LMIs Admitting Frequency-wise Model Specifications in Confidence Regions. *IEEE Transactions on Automatic Control*, 50(10):1534–1549, 2005.

[11] C. A. Larsson. Toward applications oriented optimal input design with focus on model predictive control, 2011.

[12] C. A. Larsson. Toward applications oriented optimal input design with focus on model predictive control. September 2011. Licentiate Thesis.

[13] C. A. Larsson, C. R. Rojas, and H. Hjalmarsson. MPC oriented experiment design. In *Proceedings of the 18th IFAC World Congress*, Milano., August 2011.

[14] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, New Jersey, 2nd edition, 1999.

[15] L. Ljung. System identification toolbox: User's guide. The MathWorks, Inc., 2010.

[16] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, 2004.

[17] T. Söderström. *Discrete-Time Stochastic Systems: Estimation and Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[18] K. C. Toh, M.J. Todd, R.H. Tütüncü, and R. H. Tutuncu. SDPT3- a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1998.