Optimal identification experiment design software

Xavier Bombois

Optimal identification experiment design software

X. Bombois (xavier.bombois@ec-lyon.fr)[1,2]

[1]Laboratoire Ampère, Ecole Centrale de Lyon, Université de Lyon, Ecully, France
[2]Centre National de la Recherche Scientifique (CNRS), France

March 19, 2021

1 Introduction

This document gives an overview of an optimal identification experiment design software that we have recently developed. This software is made of a number of m-files that have to be used with Matlab and can be downloaded from the French repository HAL:

https://hal.archives-ouvertes.fr/hal-03157382

In this software, we mostly focus on the so-called *least costly framework* [5] even though we also consider the more classical framework [6]. The functions of this software follow from the results in many contributions (see e.g., [10, 3, 9, 7, 5, 4]).

The LMI optimization problems involved in the different optimal experiment design problems are solved using the **LMI lab of Matlab**. In the directory *Experiment_Design_Software*, you will find another directory *MatlabfunctionsOED* where all the Matlab functions of this software are located. Besides the directory *MatlabfunctionsOED*, you will also find three m-files giving examples and a .mat-file that is needed by one of these examples. These three examples and the important functions in the directory *MatlabfunctionsOED* are discussed in this document[1].

Another software for optimal experiment design has been developed at the KTH and present interesting functionalities (see [1]).

Finally, since it is is the first version of this software and since there are many new functions, some flaws could be still present in some functions. Please, do not hesitate to contact the author with remarks, feedback, ...

2 Identification and model structures

In this software, we consider the identification of a SISO discrete-time system (sample time T_s) with one input $u(t)$ and one output $y(t)$ and described by a stable plant transfer function $G(z, \theta_0)$ and a stable, inversely stable and monic noise transfer function $H(z, \theta_0)$:

$$y(t) = G(z, \theta_0)u(t) + H(z, \theta_0)e(t) \tag{1}$$

where $e(t)$ is a white noise of variance σ_e^2 and where θ_0 is the unknown parameter vector describing the system. This unknown parameter vector will be identified using input-output data collected on the system:

[1]Note that some functions in the directory *MatlabfunctionsOED* are just subfunctions of the major functions. These subfunctions will not be discussed. Note that the subfunction *kroprod* for the Kronecker product has been developed by G. Scorletti.

$Z^N = \{y(t), u(t) | t = 1, ..., N\}$. These data can be collected by applying an excitation signal $r(t)$ to the system either in open loop or in closed loop:

$$OPEN - LOOP\ OPERATION: \quad u(t) = r(t) \tag{2}$$

$$CLOSED - LOOP\ OPERATION: \quad u(t) = r(t) - K(z)y(t) \tag{3}$$

where $K(z)$ is a stabilizing controller. When the system is operated in open loop, the input $u(t)$ is thus equal to $r(t)$ and the output $y(t)$ is given by $y(t) = G(z, \theta_0)r(t) + H(z, \theta_0)e(t)$. When the system is operated in closed loop with the controller $K(z)$, the input $u(t)$ and the output $y(t)$ are given by:

$$u(t) = \overbrace{\frac{1}{1 + K(z)G(z, \theta_0)}}^{=u_r(t)} r(t) - \frac{K(z)}{1 + K(z)G(z, \theta_0)} H(z, \theta_0)e(t) \tag{4}$$

$$y(t) = \underbrace{\frac{G(z, \theta_0)}{1 + K(z)G(z, \theta_0)}}_{=y_r(t)} r(t) + \frac{1}{1 + K(z)G(z, \theta_0)} H(z, \theta_0)e(t) \tag{5}$$

Based on the data set Z^N and a parametrized model structure $\mathcal{M} = \{G(z, \theta),\ H(z, \theta)\}$, we can determine an estimate $\hat{\theta}_N$ of θ_0 using prediction error identification [10]:

$$\hat{\theta}_N = \arg\min_{\theta} \frac{1}{N} \sum_{t=1}^{N} \epsilon^2(t, \theta) \tag{6}$$

$$\epsilon(t, \theta) = H(z, \theta)^{-1} \left(y(t) - G(z, \theta)u(t)\right) \tag{7}$$

If the chosen model structure is rich enough to describe the true system and if the data are informative, $\hat{\theta}_N$ is a consistent estimate of the unknown true parameter vector θ_0 [10]. Moreover, $\hat{\theta}_N$ is also (asymptotically) normally distributed around θ_0 with a covariance matrix P_θ given by:

$$P_\theta = \frac{\sigma_e^2}{N} \left(\psi(t, \theta_0)\psi^T(t, \theta_0)\right)^{-1} \tag{8}$$

where $\psi(t, \theta) = \frac{\partial \epsilon(t, \theta)}{\partial \theta}$. This covariance matrix can be estimated based on $\hat{\theta}_N$ and the data set Z^N:

$$P_\theta \approx \frac{\hat{\sigma}_e^2}{N} \left(\frac{1}{N} \sum_{t=1}^{N} \psi(t, \hat{\theta}_N)\psi^T(t, \hat{\theta}_N) \right)^{-1} \tag{9}$$

with $\hat{\sigma}_e^2 = \frac{1}{N} \sum_{t=1}^{N} \epsilon^2(t, \hat{\theta}_N)$. Moreover, the covariance matrix (8) can be related to the power spectrum $\Phi_r(\omega)$ of the excitation signal $r(t)$ used for the identification. If the data have been collected in open loop, we have:

$$P_\theta^{-1} = \frac{N}{2\pi\sigma_e^2} \int_{-\pi}^{\pi} F_u(e^{j\omega T_s}, \theta_0) F_u^*(e^{j\omega T_s}, \theta_0) \Phi_r(\omega) + F_e(e^{j\omega T_s}, \theta_0) F_e^*(e^{j\omega T_s}, \theta_0) \sigma_e^2 \, d\omega \tag{10}$$

$$F_u(z, \theta) = H^{-1}(z, \theta) \frac{\partial G(z, \theta)}{\partial \theta} \quad F_e(z, \theta) = H^{-1}(z, \theta) \frac{\partial H(z, \theta)}{\partial \theta}$$

If the data have been collected in closed loop with a controller $K(z)$, we have:

$$P_\theta^{-1} = \frac{N}{2\pi\sigma_e^2} \int_{-\pi}^{\pi} F_u(e^{j\omega T_s}, \theta_0) F_u^*(e^{j\omega T_s}, \theta_0) |S(e^{j\omega T_s}, \theta_0)|^2 \Phi_r(\omega) + F_v(e^{j\omega T_s}, \theta_0) F_v^*(e^{j\omega T_s}, \theta_0) \sigma_e^2 \, d\omega \tag{11}$$

where $S(z, \theta) = 1/(1 + K(z)G(z, \theta))$ and $F_v(z, \theta) = F_e(z, \theta) - K(z)S(z, \theta)H(z, \theta)F_u(z, \theta)$ with $F_u(z, \theta)$ and $F_e(z, \theta)$ as defined below (10).

As usual in prediction error identification, we will suppose that the transfer functions $G(z,\theta)$ and $H(z,\theta)$ in the mode structure $\mathcal{M}$ can be written as the ratio of a number of polynomials defined as follows

$$\begin{aligned} B(z,\theta) &= b_0 + b_1 z^{-1} + \cdots + b_{n_b-1} z^{-n_b+1} \\ A(z,\theta) &= 1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a} \\ C(z,\theta) &= 1 + c_1 z^{-1} + \cdots + c_{n_c} z^{-n_c} \\ D(z,\theta) &= 1 + d_1 z^{-1} + \cdots + d_{n_d} z^{-n_d} \\ F(z,\theta) &= 1 + f_1 z^{-1} + \cdots + f_{n_f} z^{-n_f} \end{aligned}$$

where n_a, n_b, n_c, n_d and n_f can be freely chosen by the user. The parameter vector θ is a column vector containing the coefficients of the different polynomials in alphabetical order. If the system is e.g. parametrized with the polynomials A, B and C, the parameter vector θ is defined as:

$$\theta = (a_1,\ a_2,\ ...,a_{n_a},\ b_0,\ ...,b_{n_b-1},\ c_1,...,c_{n_c})^T$$

In practice, we only use a finite number of combinations of these polynomials. This leads to the five types of model structures $\mathcal{M}$ given in Table 1. In this table, the delay n_k of the plant transfer function G can also be freely chosen by the user. A model structure is entirely defined by its type (ARX, ARMAX, OE, FIR or BJ), the delay n_k of the plant transfer function G and the number of parameters present in each of the polynomials defining this model structure. In this software, this will be determined by the input arguments *typemodel* and *vector_order* that are also defined in Table 1.

When *typemodel='arx'*, n_a and n_b in *vector_order* must be larger or equal to one ($n_a \geq 1$ and $n_b \geq 1$). When *typemodel='armax'*, n_a, n_b and n_c in *vector_order* must be larger or equal to one. When *typemodel='oe'*, n_b and n_f in *vector_order* must be larger or equal to one. When *typemodel='fir'*, n_b in *vector_order* must be larger or equal to one, but n_c and n_d may be equal to zero ($n_c \geq 0$ and $n_d \geq 0$). When *typemodel='bj'*, n_b, n_c+n_d and n_f in *vector_order* must be larger or equal to one. Let us give some examples.

Example 1. Let us consider the following system

$$y(t) = G(z,\theta_0)u(t) + H(z,\theta_0)e(t) = \frac{3z^{-1}}{1-0.7z^{-1}}u(t) + \frac{1}{1-0.7z^{-1}}e(t) \tag{12}$$

A full-order model structure for this system can be chosen as an ARX model structure (i.e. *typemodel='arx'*) with $n_a = 1$, $n_b = 1$ and $n_k = 1$ (*vector_order*=[1 1 1]). The parameter vector θ_0 that needs to be identified is given by $\theta_0 = (-0.7, 3)^T$.

Example 2. Let us consider the following system

$$y(t) = G(z,\theta_0)u(t) + H(z,\theta_0)e(t) = \frac{3z^{-1}}{1-0.7z^{-1}}u(t) + \frac{1+0.9z^{-1}+0.2z^{-2}}{1+0.6z^{-1}}e(t) \tag{13}$$

A full-order model structure for this system can be chosen as an BJ model structure (i.e. *typemodel='bj'*) with $n_b = 1$, $n_c = 2$, $n_d = 1$, $n_f = 1$ and $n_k = 1$ (*vector_order*=[1 2 1 1 1]). The parameter vector θ_0 that needs to be identified is given by $\theta_0 = (3, 0.9, 0.2, 0.6, -0.7)^T$.

To distinguish between a system operated in open loop and a system operated in closed loop, we use the input argument *controller*. To select the closed-loop configuration, the input argument *controller* must be chosen equal to the transfer function or the state-space representation of the controller $K(z)$. If $K(z) = 1$, we choose: *controller=tf(1,1,T_s)* where T_s is the sample time. To select the open-loop configuration, the input argument *controller* must be chosen equal to the scalar 0 i.e., *controller=0*.

Remark. Using the Matlab function *determinationGH.m*, you can compute the transfer function $G_i(z) = G(z,\theta_i)$ and $H_i(z) = H(z,\theta_i)$ for any value θ_i of the parameter vector θ. The syntax of this Matlab function is:

[Gi,Hi]=determinationGH(typemodel,vector_order,Ts,theta_i)

Model structure	$G(z,\theta)$	$H(z,\theta)$	*typemodel*	*vector_order*
ARX	$\frac{z^{-n_k}B(z,\theta)}{A(z,\theta)}$	$\frac{1}{A(z,\theta)}$	'arx'	$[n_a, n_b, n_k]$
ARMAX	$\frac{z^{-n_k}B(z,\theta)}{A(z,\theta)}$	$\frac{C(z,\theta)}{A(z,\theta)}$	'armax'	$[n_a, n_b, n_c, n_k]$
OE	$\frac{z^{-n_k}B(z,\theta)}{F(z,\theta)}$	1	'oe'	$[n_b, n_f, n_k]$
FIR	$z^{-n_k}B(z,\theta)$	$\frac{C(z,\theta)}{D(z,\theta)}$	'fir'	$[n_b, n_c, n_d, n_k]$
BJ	$\frac{z^{-n_k}B(z,\theta)}{F(z,\theta)}$	$\frac{C(z,\theta)}{D(z,\theta)}$	'bj'	$[n_b, n_c, n_d, n_f, n_k]$

Table 1: Model structures

where the input arguments *typemodel*, *vector_order*, *Ts* (sampling time T_s) are defined above and where the input argument *theta_i* must be chosen equal to the column vector θ_i. The two output arguments are the transfer functions $G_i(z) = G(z,\theta_i)$ and $H_i(z) = H(z,\theta_i)$ (in their state-space representation).

3 Parametrization of the excitation spectrum

In general, in optimal experiment design, the objective is to optimally design the power spectrum $\Phi_r(\omega)$ of the excitation signal $r(t)$. We will consider here two types of excitation signals [8, 5].

The first type is a multisinus made up of L sinusoids at different frequencies i.e., $r(t) = \sum_{m=1}^{L} A_m cos(\omega_m t T_s + \phi_m)$. Such a multisinus has the following spectrum:

$$\Phi_r(\omega) = \pi \sum_{m=1}^{L} \frac{A_m^2}{2} \ (\delta(\omega - \omega_m) + \delta(\omega + \omega_m)) \tag{14}$$

We observe that the spectrum $\Phi_r(\omega)$ is independent of the phase shifts ϕ_m $(m = 1, ..., L)$. When designing the spectrum of the multisinus, we will suppose that the frequencies ω_m $(m = 1, ..., L)$ are a-priori specified by the user. In other words, only the amplitudes A_m $(m = 1, ..., L)$ will be determined optimally by the optimal experiment design problem. Once these amplitudes determined, a realization of the excitation signal $r(t)$ can be easily constructed by e.g. choosing $\phi_m = 0$ $(m = 1, ..., L)$.

The second type of excitation signal is a unit variance white noise filtered by an arbitrary FIR filter of degree M. Such excitation signal has the following spectrum:

$$\Phi_r(\omega) = c_0 + \sum_{m=1}^{M} c_m \left(e^{-jm\omega T_s} + e^{jm\omega T_s}\right) \geq 0 \ \forall \omega \tag{15}$$

where c_m $(m = 0, ..., M)$ are the auto-correlation coefficients of the excitation signal $r(t)$. It is to be noted that, when M is chosen equal to zero, this parametrization corresponds to a white noise $r(t)$ of variance c_0. The coefficients c_m $(m = 0, ..., M)$ in (15) will be the ones that will be optimally determined by the optimal experiment design problem.

Based on a spectrum of the type (15), a realization of the excitation can be achieved based on spectral factorization [9]. The matlab function *real_filtwhitenoise* in this software allows to derive a realization of N

samples of a signal $r(t)$ having the spectrum (15). This function takes, as first input argument, a column vector of dimension $M+1$ containing the coefficients c_m $(m = 0, ..., M)$ i.e., $(c_0, c_1, ..., c_M)^T$ and, as second input argument, the number N of samples of the desired realization. Consequently, a realization of 1000 samples of the signal $r(t)$ with spectrum $\Phi_r(\omega) = 1+0.1\left(e^{-j\omega T_s} + e^{j\omega T_s}\right)$ can be obtained via the command:

r=real_filtwhitenoise([1;0.1],1000);.

The value of the spectrum (15) at the frequencies in a vector *vecw* can be obtained via the matlab function *analysiswhitenoisespectrum* that takes, as first input argument, the column vector $(c_0, c_1, ..., c_M)^T$ and, as second input argument, the row vector *vecw* while the third argument is the sampling time T_s. Consequently, for $T_s = 1$, the value of the spectrum $\Phi_r(\omega) = 1 + 0.1\left(e^{-j\omega T_s} + e^{j\omega T_s}\right)$ at the frequencies 0.5 and 1.5 can be obtained via the command:

phirw=analysiswhitenoisespectrum([1;0.1],[0.5,1.5],1);.

phirw is then a row vector of dimension 2 given by $(\Phi_r(\omega = 0.5),\ \Phi_r(\omega = 1.5))$.

4 Matlab function *optimalexpdesign_classical*: classical A-optimal or E-optimal experiment design

Syntax.

[PHI,Pthetath,optimizedPtheta_measure]=optimalexpdesign_classical(typeoptimality,typecost,parameter_cost, ...
... typespectrum,parameter_spectrum,ndata,typemodel,vector_order,Ts,controller,theta_init,var_e_i)

This matlab function computes the optimal excitation signal (A-optimal or E-optimal) for the identification of a discrete-time system operated either in open loop or in closed loop. The considered optimal experiment design problem can be summarized as determining, for a given data length N, the spectrum of the excitation signal minimizing a scalar measure of P_θ subject to one or several cost constraints [6, 9, 7]. We will here consider different scalar measures of P_θ and different cost constraints.

The measure of P_θ can be determined via the input argument *typeoptimality*. If we choose *typeoptimality='A'*, we wish to minimize the trace of P_θ (A-optimality). If we choose *typeoptimality='E'*, we wish to minimize the largest eigenvalue of P_θ (E-optimality).

The cost constraint can be determined via the input arguments *typecost* and *parameter_cost*. Let us denote by $\mathcal{P}(x)$ the power of the discrete-time signal $x(t)$ and by $u_r(t)$ (resp. $y_r(t)$) the part of the input signal $u(t)$ (resp. the output signal $y(t)$) induced by the excitation signal. See (4) and (5) for the closed-loop operation. For the open-loop operation, $u_r(t) = u(t) = r(t)$ and $y_r(t) = G(z, \theta_0)r(t)$.

If we choose *typecost='separated'* and *parameter_cost =[β_u,β_y]* for some user-chosen scalar constants β_u and β_y, we have two cost constraints:

$$\mathcal{P}(u_r) < \beta_u \tag{16}$$

$$\mathcal{P}(y_r) < \beta_y \tag{17}$$

If we choose *typecost='sum'* and *parameter_cost =[β,α_u,α_y]* for some user-chosen scalar constants β, α_u and α_y, we have a unique cost constraint:

$$\alpha_u\mathcal{P}(u_r) + \alpha_y\mathcal{P}(y_r) < \beta \tag{18}$$

If we choose *typecost='power_r'* and *parameter_cost =[β]* for some user-chosen scalar constant β, we have the following cost constraint:

$$\mathcal{P}(r) < \beta \tag{19}$$

In other words, we directly constrain the power of the excitation signal $r(t)$.

We have also to specify the type of spectrum we wish to design via the input arguments *typespectrum* and *parameter_spectrum*. If we choose *typespectrum='whitenoise'* and *parameter_spectrum =[M]* where $M \geq 0$ is an user-chosen scalar constant, we consider the parametrization (15) with $M+1$ decision variables c_0, c_1,..., c_M. If we choose *typespectrum='multisinus'* and we choose *parameter_spectrum* equal to an user-chosen column vector $(\omega_1, \omega_2, ..., \omega_L)^T$ containing L different frequencies , we consider the parametrization (14) with L decision variables A_1, A_2,..., A_L.

The remaining input arguments are:

- the input argument *ndata* which must be chosen equal to the duration N of the to-be-designed experiment
- the input arguments describing the to-be-identified system i.e. *typemodel*, *vector_order*, *Ts* (sampling time) and *controller*. See Section 2.
- The input argument *theta_init* which has to be chosen equal to a first estimate θ_{init} of θ_0.
- The input argument *var_e_i* which has to be chosen equal to a first estimate $\sigma^2_{e,init}$ of σ^2_e.

These last input arguments (i.e., θ_{init} and $\sigma^2_{e,init}$) are necessary to evaluate the quantity P_θ (and also in some cases the quantity $\mathcal{J}$) that depend(s) on the unknown quantities θ_0 and σ^2_e.

The first output argument of this matlab function is a column vector *PHI* containing the optimal coefficients of the spectrum. If *typespectrum='whitenoise'* and *parameter_spectrum =[M]*, this column vector *PHI* is equal to $(c_{0,opt}, c_{1,opt}, ..., c_{M,opt})^T$. If *typespectrum='multisinus'* and *parameter_spectrum*=$(\omega_1, \omega_2, ..., \omega_L)^T$, this column vector *PHI* is equal to $(A_{1,opt}, A_{2,opt}, ..., A_{M,opt})^T$.

The second output argument *Pthetath* is the covariance matrix $P_{\theta,opt}$ that is computed with the optimal spectrum $\Phi_{r,opt}$ and with the estimates θ_{init}, $\sigma^2_{e,init}$ for the unknown quantities θ_0 and σ^2_e. For this purpose, we use the expression (10) in the open-loop case and the expression (11) in the closed-loop case.

The third output argument *optimizedPtheta_measure* is the optimal value of the measure of P_θ that has been minimized. If *typeoptimality='A'*, this output argument is equal to the trace of $P_{\theta,opt}$. If *typeoptimality='E'*, this output argument is equal to the largest eigenvalue of $P_{\theta,opt}$.

Example. An example of the use of the Matlab function *optimalexpdesign_classical* is given in the m-file:

example_classicaldesign.m.

In this example, we consider the BJ system:

$$y(t) = G(z,\theta_0)u(t) + H(z,\theta_0)e(t) = \frac{3z^{-1}}{1-0.7z^{-1}}u(t) + (1-0.9z^{-1})\ e(t) \tag{20}$$

and we suppose for simplicity that the initial estimates for $\theta_0 = (3, -0.9, -0.7)^T$ and $\sigma^2_e = 1$ are the true values of these quantities.

5 Matlab function *LCexpdesign_simple*: simple least costly experiment design

Syntax.

[PHI,Pthetath,cost]=LCexpdesign_simple(typecost,parameter_cost,typeaccuracy,parameter_accuracy,...
... typespectrum,parameter_spectrum,ndata,typemodel,vector_order,Ts,controller,theta_init,var_e_i)

This matlab function computes the optimal excitation signal for the identification of a discrete-time system operated either in open loop or in closed loop. The considered optimal experiment design problem can be summarized as determining, for a given data length N, the spectrum of the excitation signal minimizing the cost $\mathcal{J}$ of the identification experiment subject to one or several (simple) accuracy constraints [4]. We will here consider different definitions for the cost of an identification experiment and different accuracy constraints.

The cost $\mathcal{J}$ of the identification experiment will be specified via the input arguments *typecost* and *parameter_cost*. If we choose *typecost='sum'* and *parameter_cost =[α_u,α_y]* for some user-chosen scalar constants α_u and α_y, the cost $\mathcal{J}$ is defined as:

$$\mathcal{J} = \alpha_u \mathcal{P}(u_r) + \alpha_y \mathcal{P}(y_r) \tag{21}$$

In other words, the cost is defined as a linear combination of the power of y_r and u_r (the parts of the output and input signals induced by the excitation signal $r(t)$).

If we choose *typecost='power_r'* and *parameter_cost =[]*, $\mathcal{J}$ is defined as $\mathcal{J} = \mathcal{P}(r)$. In other words, $\mathcal{J}$ is defined as the power of the excitation signal $r(t)$.

The considered accuracy constraint(s) can be specified via the input argument *typeaccuracy* and *parameter_accuracy*. If we choose *typeaccuracy='Radm'* and *parameter_accuracy* equal to a positive-definite matrix $R_{adm} = R_{adm}^T$, the accuracy constraint is given by:

$$P_\theta^{-1} \geq R_{adm} \tag{22}$$

The two other types of accuracy constraints pertain to bound the modeling error $|G(e^{j\omega T_s}, \hat{\theta}_N) - G(e^{j\omega T_s}, \theta_0)|$ between the identified model and the true system in the frequency domain. Using a first order-approximation [4], a measure of this modeling error can be given by the following quantities as a function of the covariance matrix P_θ of $\hat{\theta}_N$:

$$r_1(\omega) = \sqrt{\lambda_{max}\left(\mathcal{T}(e^{j\omega T_s}, \theta_0)\ P_\theta\ \mathcal{T}^T(e^{j\omega T_s}, \theta_0)\right)} \tag{23}$$

$$r_2(\omega) = \sqrt{\Lambda(e^{j\omega T_s}, \theta_0)\ P_\theta\ \Lambda^*(e^{j\omega T_s}, \theta_0)} \tag{24}$$

where $\Lambda^T(z,\theta) = \frac{\partial G(z,\theta)}{\partial \theta}$ and $\mathcal{T}(e^{j\omega T_s}, \theta_0) = \begin{pmatrix} Re(\Lambda(e^{j\omega T_s}, \theta_0)) \\ Im(\Lambda(e^{j\omega T_s}, \theta_0)) \end{pmatrix}$ ($Re(.)$ and $Im(.)$ denotes the real and imaginary part) and where $\lambda_{max}(A)$ is the largest eigenvalue of the matrix A. Note that, up to a first order approximation, $r_2(\omega)$ is in fact equal to the square root of $var(G(e^{j\omega T_s}, \hat{\theta}_N)) \triangleq E|G(e^{j\omega T_s}, \hat{\theta}_N) - G(e^{j\omega T_s}, \theta_0)|^2$ (E is the expectation operator) [10] and note also that $\mathcal{T}(e^{j\omega T_s}, \theta_0)\ P_\theta\ \mathcal{T}^T(e^{j\omega T_s}, \theta_0)$ (in the expression of $r_1(\omega)$) is the covariance matrix of the vector $\begin{pmatrix} Re(G(e^{j\omega T_s}, \hat{\theta}_N)) \\ Im(G(e^{j\omega T_s}, \hat{\theta}_N)) \end{pmatrix}$ (see [2, 4] for more details).

Using $r_1(\omega)$ or $r_2(\omega)$, we can consider the following accuracy constraints:

$$r_1(\omega) \leq r_{adm}(\omega) \quad \forall \omega \in \Omega \tag{25}$$

$$r_2(\omega) \leq r_{adm}(\omega) \quad \forall \omega \in \Omega \tag{26}$$

where $r_{adm}(\omega)$ is an user-chosen frequency function and $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ is an user-chosen set of n frequencies covering the frequency domain. If we choose *typeaccuracy='r1'* and *parameter_accuracy* equal to a matrix of dimension $n \times 2$ given by:

$$\begin{pmatrix} \omega_1 & r_{adm}(\omega_1) \\ \omega_2 & r_{adm}(\omega_2) \\ ... & ... \\ \omega_n & r_{adm}(\omega_n) \end{pmatrix}, \tag{27}$$

the accuracy constraint is given by (25). If we choose *typeaccuracy='r2'* and *parameter_accuracy* equal to the matrix (27), the accuracy constraint is given by (26).

Like in the previous section, we have also to specify the type of spectrum we wish to design via the input arguments *typespectrum* and *parameter_spectrum*. If we choose *typespectrum='whitenoise'* and *parameter_spectrum =[M]* where $M \geq 0$ is an user-chosen scalar constant, we consider the parametrization (15) with $M+1$ decision variables c_0, c_1,..., c_M. If we choose *typespectrum='multisinus'* and we choose *parameter_spectrum* equal to an user-chosen column vector $(\omega_1, \omega_2, ..., \omega_L)^T$ containing L different frequencies , we consider the parametrization (14) with L decision variables A_1, A_2,..., A_L.

The remaining input arguments are:

- the input argument *ndata* which must be chosen equal to the duration N of the to-be-designed experiment
- the input arguments describing the to-be-identified system i.e. *typemodel*, *vector_order*, *Ts* (sampling time) and *controller*. See Section 2.
- The input argument *theta_init* which has to be chosen equal to a first estimate θ_{init} of θ_0.
- The input argument *var_e_i* which has to be chosen equal to a first estimate $\sigma^2_{e,init}$ of σ^2_e.

These last input arguments (i.e., θ_{init} and $\sigma^2_{e,init}$) are necessary to evaluate the quantity P_θ (and also in some cases the quantity $\mathcal{J}$) that depend(s) on the unknown quantities θ_0 and σ^2_e.

The first output argument of this matlab function is a column vector *PHI* containing the optimal coefficients of the spectrum. If *typespectrum='whitenoise'* and *parameter_spectrum =[M]*, this column vector *PHI* is equal to $(c_{0,opt}, c_{1,opt}, ..., c_{M,opt})^T$. If *typespectrum='multisinus'* and *parameter_spectrum*=$(\omega_1, \omega_2, ..., \omega_L)^T$, this column vector *PHI* is equal to $(A_{1,opt}, A_{2,opt}, ..., A_{M,opt})^T$.

The second output argument *Pthetath* is the covariance matrix $P_{\theta,opt}$ that is computed with the optimal spectrum $\Phi_{r,opt}$ and with the estimates θ_{init}, $\sigma^2_{e,init}$ for the unknown quantities θ_0 and σ^2_e. For this purpose, we use the expression (10) in the open-loop case and the expression (11) in the closed-loop case.

The third output argument *cost* is the optimal value $\mathcal{J}_{opt}$ of the cost $\mathcal{J}$. When necessary, this optimal value is computed with the estimate θ_{init} for θ_0.

Example. An example of the use of the Matlab function *LCexpdesign_simple* is given in the m-file:

example_simpleleastcostlydesign.m.

In this example, we consider the same BJ system as in [4]:

$$y(t) = G(z,\theta_0)u(t) + H(z,\theta_0)e(t) = \frac{3.6z^{-1}}{1-0.7z^{-1}}u(t) + (1-0.9z^{-1})\ e(t) \tag{28}$$

and we suppose for simplicity that the initial estimates for $\theta_0 = (3.6, -0.9, -0.7)^T$ and $\sigma^2_e = 1$ are the true values of these quantities. For the accuracy constraint (25), the frequency function $r_{adm}(\omega)$ is chosen as

$$r_{adm}(\omega) = \frac{0.01\ |G(e^{j\omega T_s}, \theta_0)|}{2.45} \tag{29}$$

This bound on the measure of the modeling error is indeed the one considered in[2] Section 5.1 of [4]. Note that the results given by this m-file are slightly different than the ones obtained in Section 5.1 of [4] since, in Section 5.1 of [4], $\theta_{init} \neq \theta_0$.

[2]Indeed, in Section 5.1 of [4], the accuracy constraint is $2.45\ r_1(\omega) \leq 0.01\ |G(e^{j\omega T_s}, \theta_0)|$.

6 Matlab function *LCexpdesign_Id4C*: least costly experiment design for control

Syntax.

[PHI,Pthetath,cost]=LCexpdesign_simple(typecost,parameter_cost,frequency_weighting,K_init,chi,...
... typespectrum,parameter_spectrum,ndata,typemodel,vector_order,Ts,controller,theta_init,var_e_i)

This matlab function computes the optimal excitation signal for the identification of a discrete-time system operated either in open loop or in closed loop. The considered optimal experiment design problem can be summarized as determining, for a given data length N, the spectrum of the excitation signal minimizing the cost $\mathcal{J}$ of the identification experiment while ensuring that the uncertainty of the identified model is sufficient small for the controller designed with the identified model to achieve satisfactory performance with the unknown true system. We consider more particularly the framework introduced in [5].

The cost of the experiment will be defined as in the previous section. Let us define more precisely the accuracy constraint considered in [5]. For this purpose, note that, since the identified parameter vector is (asymptotically) normally distributed around θ_0 with covariance P_θ (see Section 2), the unknown true parameter vector θ_0 will lie, modulo a probability of $\eta\%$, in the following uncertainty ellipsoid:

$$U = \{\theta \in \mathbf{R}^k \mid (\theta - \hat{\theta}_N)^T P_\theta^{-1} (\theta - \hat{\theta}_N) < \chi\} \tag{30}$$

where χ is such that $Pr(\chi^2(k) < \chi) = \eta$ (k is the dimension of θ and $\chi^2(k)$ is the χ-squared distribution with k degrees of freedom). Suppose that a controller $\hat{K}(z)$ has been designed with the identified model $G(z, \hat{\theta}_N)$. The performance $F(\hat{K}, \theta, \omega)$ of the loop $[\hat{K}(z)\ G(z,\theta)]$ made up of the designed controller $\hat{K}(z)$ and an arbitrary system $G(z,\theta)$ is defined at the frequency ω as:

$$F(\hat{K}, \theta, \omega) = \sigma_{max}\left(\begin{pmatrix} W_{l,1}(e^{j\omega T_s}) & 0 \\ 0 & W_{l,2}(e^{j\omega T_s}) \end{pmatrix} R(\hat{K}, \theta, e^{j\omega T_s}) \begin{pmatrix} W_{r,1}(e^{j\omega T_s}) & 0 \\ 0 & W_{r,2}(e^{j\omega T_s}) \end{pmatrix}\right) \tag{31}$$

$$R(\hat{K}, \theta, z) = \begin{pmatrix} \frac{\hat{K}(z)G(z,\theta)}{1+\hat{K}(z)G(z,\theta)} & \frac{G(z,\theta)}{1+\hat{K}(z)G(z,\theta)} \\ \frac{\hat{K}(z)}{1+\hat{K}(z)G(z,\theta)} & \frac{1}{1+\hat{K}(z)G(z,\theta)} \end{pmatrix} \tag{32}$$

where $\sigma_{max}(A)$ is the largest singular value of the matrix A and where $W_{l,1}(z)$, $W_{l,2}(z)$, $W_{r,1}(z)$ and $W_{r,2}(z)$ are user-chosen weighting functions. In this H_∞-framework, the loop $[\hat{K}(z)\ G(z,\theta)]$ will be deemed to have satisfactory performance if

$$F(\hat{K}, \theta, \omega) < 1 \quad \forall \omega \tag{33}$$

It is to be noted that (33) implies that, for $i = 1, 2$ and $j = 1, 2$,

$$|R_{ij}(\hat{K}, \theta, e^{j\omega T_s})| < \frac{1}{|W_{l,i}(e^{j\omega T_s}) W_{r,j}(e^{j\omega T_s})|} \quad \forall \omega \tag{34}$$

where R_{ij} is the (i,j)-entry of the matrix R.

Since θ_0 lies in U, the loop $[\hat{K}(z)\ G(z,\theta_0)]$ will be deemed to have satisfactory performance if, for all ω,

$$F(\hat{K}, \theta, \omega) < 1 \quad \forall\ \theta \in U \tag{35}$$

The goal of the least costly experiment design for control is to minimize the cost $\mathcal{J}$ of the experiment in such a way that the size P_θ of the uncertainty set U is small enough to guarantee that (35) holds at each frequency ω. Since the center $\hat{\theta}_N$ of U and $\hat{K}$ are both unknown before the identification experiment, $\hat{\theta}_N$ will be replaced by an available initial estimate θ_{init} of θ_0 and $\hat{K}$ will be replaced by the controller K_{init} designed using $G(z, \theta_{init})$. Moreover, the constraint (35) will only be verified at a finite grid Ω of the frequency range.

In the matlab function, the cost $\mathcal{J}$ of the identification experiment will be specified via the input arguments *typecost* and *parameter_cost*. If we choose *typecost='sum'* and *parameter_cost =[α_u,α_y]* for some user-chosen scalar constants α_u and α_y, the cost $\mathcal{J}$ is defined as:

$$\mathcal{J} = \alpha_u \mathcal{P}(u_r) + \alpha_y \mathcal{P}(y_r) \tag{36}$$

In other words, the cost is defined as a linear combination of the power of y_r and u_r (the parts of the output and input signals induced by the excitation signal $r(t)$.

If we choose *typecost='power_r'* and *parameter_cost =[]*, $\mathcal{J}$ is defined as $\mathcal{J} = \mathcal{P}(r)$. In other words, $\mathcal{J}$ is defined as the power of the excitation signal $r(t)$.

In order to define the accuracy constraint, we need to define a number of input arguments *frequency_weighting*, *K_init* and *chi*. The first input argument is *frequency_weighting*. If we suppose that the frequency grid Ω defined above contains n frequencies (i.e. $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$), the input argument *frequency_weighting* will have to be chosen equal to the following matrix of dimension $n \times 5$:

$$\begin{pmatrix} \omega_1 & W_{l,1}(e^{j\omega_1 T_s}) & W_{l,2}(e^{j\omega_1 T_s}) & W_{r,1}(e^{j\omega_1 T_s}) & W_{r,2}(e^{j\omega_1 T_s}) \\ \omega_2 & W_{l,1}(e^{j\omega_2 T_s}) & W_{l,2}(e^{j\omega_2 T_s}) & W_{r,1}(e^{j\omega_2 T_s}) & W_{r,2}(e^{j\omega_2 T_s}) \\ ... & ... & ... & ... & ... \\ \omega_n & W_{l,1}(e^{j\omega_n T_s}) & W_{l,2}(e^{j\omega_n T_s}) & W_{r,1}(e^{j\omega_n T_s}) & W_{r,2}(e^{j\omega_n T_s}) \end{pmatrix} \tag{37}$$

Some of the weighting functions can be chosen equal to zero. For example, if you wish to only constrain the sensitivity function (i.e. R_{22}), you specify $W_{l,1}(z) = W_{r,1}(z) = 0$. The input argument *K_init* must be chosen equal to the initial estimate K_{init} of the controller $\hat{K}(z)$ i.e., the controller that is designed using $G(z, \theta_{init})$. The input argument *chi* must be chosen equal to the scalar χ in U (see (30)).

Like in the previous sections, we have also to specify the type of spectrum we wish to design via the input arguments *typespectrum* and *parameter_spectrum*. If we choose *typespectrum='whitenoise'* and *parameter_spectrum =[M]* where $M \geq 0$ is an user-chosen scalar constant, we consider the parametrization (15) with $M + 1$ decision variables c_0, c_1,..., c_M. If we choose *typespectrum='multisinus'* and we choose *parameter_spectrum* equal to an user-chosen column vector $(\omega_1, \omega_2, ..., \omega_L)^T$ containing L different frequencies , we consider the parametrization (14) with L decision variables A_1, A_2,..., A_L.

The remaining input arguments are:

- the input argument *ndata* which must be chosen equal to the duration N of the to-be-designed experiment
- the input arguments describing the to-be-identified system i.e. *typemodel*, *vector_order*, *Ts* (sampling time) and *controller*. See Section 2. Recall that the input argument *controller* (if it is nonzero) corresponds to the controller that is present in the loop when doing the experiment. This controller may thus be different from the controller given in the input argument *K_init*.
- The input argument *theta_init* which has to be chosen equal to a first estimate θ_{init} of θ_0. This is also the estimate with which the controller given in the input argument *K_init* has been designed.
- The input argument *var_e_i* which has to be chosen equal to a first estimate $\sigma^2_{e,init}$ of σ^2_e.

These last input arguments (i.e., θ_{init} and $\sigma^2_{e,init}$) are necessary to evaluate the quantity P_θ (and also in some cases the quantity $\mathcal{J}$) that depend(s) on the unknown quantities θ_0 and σ^2_e.

The first output argument of this matlab function is a column vector *PHI* containing the optimal coefficients of the spectrum. If *typespectrum='whitenoise'* and *parameter_spectrum =[M]*, this column vector *PHI* is equal to $(c_{0,opt}, c_{1,opt}, ..., c_{M,opt})^T$. If *typespectrum='multisinus'* and *parameter_spectrum*=$(\omega_1, \omega_2, ..., \omega_L)^T$, this column vector *PHI* is equal to $(A_{1,opt}, A_{2,opt}, ..., A_{M,opt})^T$.

The second output argument *Pthetath* is the covariance matrix $P_{\theta,opt}$ that is computed with the optimal spectrum $\Phi_{r,opt}$ and with the estimates θ_{init}, $\sigma^2_{e,init}$ for the unknown quantities θ_0 and σ^2_e. For this purpose, we use the expression (10) in the open-loop case and the expression (11) in the closed-loop case.

The third output argument *cost* is the optimal value $\mathcal{J}_{opt}$ of the cost $\mathcal{J}$. When necessary, this optimal value is computed with the estimate θ_{init} for θ_0.

To verify the results given by the Matlab function, we also provide the Matlab function *robustnessanalysis.m* that allows to perform the robustness analysis.

Syntax.

[nominal_perf,worstcase_perf,mu]=robustnessanalysis(typemodel,vector_order,Ts,theta_hat,K_hat,chi,Ptheta,vec_w)

In a nutshell, based on the identified model $G(z,\hat{\theta}_N)$, a controller $\hat{K}(z)$ designed with this identified model and the uncertainty set U obtained after an identification experiment, this function provides the worst case performance $\mathcal{R}_{WC,ij}$ for each of the closed-loop transfer functions and for all frequencies ω in an user-chosen frequency grid. For $i = 1, 2$ and $j = 1, 2$ and for a given frequency ω, $\mathcal{R}_{WC,ij}$ is defined as:

$$\mathcal{R}_{WC,ij}(\omega) = \sup_{\theta \in U} |R_{ij}(\hat{K}, \theta, e^{j\omega T_s})| \tag{38}$$

where R_{ij} is the (i,j)-entry of the matrix R (see (32). We can then easily verify whether (34) holds. The worst case performance (38) is computed using the tools in [3]. Note that, prior to that, it is verified[3] whether the loop $[\hat{K}(z)\ G(z,\theta)]$ is stable for all $\theta \in U$. This is also done using the tools in [3] via the computation of a frequency function $\mu(\omega)$. If $\mu(\omega)$ is smaller than one at all frequencies, $[\hat{K}(z)\ G(z,\theta)]$ is stable for all $\theta \in U$.

The Matlab function *robustnessanalysis.m* takes as first input arguments *typemodel vector_order*, *Ts* and *theta_hat* that allows to define $G(z,\hat{\theta}_N)$ (*theta_hat* has to be chosen equel to $\hat{\theta}_N$). The next input argument *K_hat* has to be chosen equal to the controller $\hat{K}(z)$. Finally, the input arguments *chi* and *Ptheta* must be respectively chosen equal to the scalar χ and the covariance matrix P_θ defining U (see (30)). Finally, the input argument *vec_w* is a **column** vector containing the frequencies at which $\mathcal{R}_{WC,ij}$ must be computed.

If *vec_w* is equal to $(\omega_1, \omega_2, ..., \omega_n)^T$, the first output argument *nominal_perf* represents the nominal performance i.e. the one of the loop $[\hat{K}(z)\ G(z,\hat{\theta}_N)]$ and is given by the following matrix:

$$\begin{pmatrix} \omega_1 & |R_{11}(\hat{K},\hat{\theta}_N,e^{j\omega_1 T_s})| & |R_{21}(\hat{K},\hat{\theta}_N,e^{j\omega_1 T_s})| & |R_{12}(\hat{K},\hat{\theta}_N,e^{j\omega_1 T_s})| & |R_{22}(\hat{K},\hat{\theta}_N,e^{j\omega_1 T_s})| \\ \omega_2 & |R_{11}(\hat{K},\hat{\theta}_N,e^{j\omega_2 T_s})| & |R_{21}(\hat{K},\hat{\theta}_N,e^{j\omega_2 T_s})| & |R_{12}(\hat{K},\hat{\theta}_N,e^{j\omega_2 T_s})| & |R_{22}(\hat{K},\hat{\theta}_N,e^{j\omega_2 T_s})| \\ ... & ... & ... & ... & ... \\ \omega_n & |R_{11}(\hat{K},\hat{\theta}_N,e^{j\omega_n T_s})| & |R_{21}(\hat{K},\hat{\theta}_N,e^{j\omega_n T_s})| & |R_{12}(\hat{K},\hat{\theta}_N,e^{j\omega_n T_s})| & |R_{22}(\hat{K},\hat{\theta}_N,e^{j\omega_n T_s})| \end{pmatrix} \tag{39}$$

The second output argument *worstcase_perf* represents the worst case performance and is given by the following matrix:

$$\begin{pmatrix} \omega_1 & \mathcal{R}_{WC,11}(\omega_1) & \mathcal{R}_{WC,21}(\omega_1) & \mathcal{R}_{WC,12}(\omega_1) & \mathcal{R}_{WC,22}(\omega_1) \\ \omega_2 & \mathcal{R}_{WC,11}(\omega_2) & \mathcal{R}_{WC,21}(\omega_2) & \mathcal{R}_{WC,12}(\omega_2) & \mathcal{R}_{WC,22}(\omega_2) \\ ... & ... & ... & ... & ... \\ \omega_n & \mathcal{R}_{WC,11}(\omega_n) & \mathcal{R}_{WC,21}(\omega_n) & \mathcal{R}_{WC,12}(\omega_n) & \mathcal{R}_{WC,22}(\omega_n) \end{pmatrix} \tag{40}$$

Finally, the third output argument *mu* represents the robust stability analysis and is given by the following matrix:

$$\begin{pmatrix} \omega_1 & \mu(\omega_1) \\ \omega_2 & \mu(\omega_2) \\ ... & ... \\ \omega_n & \mu(\omega_n) \end{pmatrix} \tag{41}$$

[3]If it is not the case, $\mathcal{R}_{WC,ij}(\omega)$ will be infinite at one frequency ω.

Example. An example of the use of the Matlab function *LCexpdesign_Id4C.m* and *robustnessanalysis.m* is given in the m-file:

example_leastcostlydesignforcontrol.m.

In this example, we consider the same ARX system as in [5]:

$$y(t) = G(z,\theta_0)u(t) + H(z,\theta_0)e(t)$$

$$\begin{array}{l} G(z,\theta_0) = \frac{z^{-3}(0.10276+0.18123z^{-1})}{1-1.99185z^{-1}+2.20265z^{-2}-1.84083z^{-3}+0.89413z^{-4}} \\ H(z,\theta_0) = \frac{1}{1-1.99185z^{-1}+2.20265z^{-2}-1.84083z^{-3}+0.89413z^{-4}} \end{array}$$

and we suppose for simplicity that the initial estimates for $\theta_0 = (-1.99185, 2.20265, -1.84083, 0.89413, 0.10276, 0.18123)^T$ and $\sigma_e^2 = 0.5$ are the true values of these quantities. Like in [5], we put a constraint on the entry R_{22} of the matrix R (see (32)), but we here also put a constraint on the entry R_{12}:

$$W_{l,1}(z) = 0.1 \quad W_{l,2}(z) = \frac{0.5165 - 0.4632z^{-1}}{1 - 0.999455z^{-1}} \quad W_{r,1}(z) = 0 \quad W_{r,2}(z) = 1$$

After the experiment design achieved with *LCexpdesign_Id4C.m*, we use *robustnessanalysis.m* to verify whether $\mathcal{R}_{WC,12}(\omega)$ and $\mathcal{R}_{WC,22}(\omega)$ satisfy the imposed constraints:

$$\mathcal{R}_{WC,12}(\omega) < \underbrace{\frac{1}{|W_{l,1}(e^{j\omega T_s})W_{r,2}(e^{j\omega T_s})|}}_{=10} \quad \textit{and} \quad \mathcal{R}_{WC,22}(\omega) < \underbrace{\frac{1}{|W_{l,2}(e^{j\omega T_s})W_{r,2}(e^{j\omega T_s})|}}_{=\frac{1}{|W_{l,2}(e^{j\omega T_s})|}}$$

For simplicity, the worst case performance is computed with $\hat{K} = K_{init}$ and $\hat{\theta}_N = \theta_{init} = \theta_0$. The initial controller K_{init} can be found in *initialcontroller.mat*.

7 Acknowledgment

We thank Federico Morelli for his careful re-reading of the software and his suggestions.

References

[1] M. Annergren and C. Larsson. MOOSE: model based optimal input design toolbox for matlab. Technical report, Royal Institute of Technology (KTH), Sweden, 2011.

[2] X. Bombois, B. Anderson, and M. Gevers. Quantification of frequency domain error bounds with guaranteed confidence level in prediction error identification. *Systems & control letters*, 54(5):471–482, 2005.

[3] X. Bombois, M. Gevers, G. Scorletti, and B.D.O. Anderson. Robustness analysis tools for an uncertainty set obtained by prediction error identification. *Automatica*, 37(10):1629–1636, 2001.

[4] X. Bombois and G. Scorletti. Design of least costly identification experiments : The main philosophy accompanied by illustrative examples. *Journal Européen des Systèmes Automatisés (JESA)*, 46(6-7):587–610, 2012, Available on the HAL repository hal-00756344.

[5] X. Bombois, G. Scorletti, M. Gevers, P.M.J. Van den Hof, and R. Hildebrand. Least costly identification experiment for control. *Automatica*, 42(10):1651–1662, 2006.

[6] G. Goodwin and R. Payne. *Dynamic system identification: Experiment design and data analysis*. New York, Academic Press, Inc., 1977.

[7] H. Jansson. *Experiment design with Applications in Identification and Control*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.

[8] 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, October 2005.

[9] K. Lindqvist. *On experiment design in identification of smooth linear systems*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.

[10] L. Ljung. *System Identification: Theory for the User, 2nd Edition*. Prentice-Hall, Englewood Cliffs, NJ, 1999.