D-Optimal Input Design for FIR-type Nonlinear Systems: A Dispersion-based Approach

Alexander De Cock [a], Michel Gevers [b], Johan Schoukens [a]

[a] *ELEC, Vrije Universiteit Brussel, 1050 Brussel, Belgium*

[b] *ICTEAM, Louvain University, B1348 Louvain la Neuve, Belgium*

Abstract

Optimal input design is an important step of the identification process in order to reduce the model variance. In this work a D-optimal input design method for FIR-type nonlinear systems is presented. The optimization of the determinant of the Fisher matrix is expressed as a convex optimization problem. The optimization is performed using an equivalent dispersion-based criterion. This method is easy to implement and converges monotonically to the optimal solution. Without constraints, the optimal design cannot be realized as a time sequence. By imposing that the design should lie in the subspace described by a symmetric and non-overlapping basis, a realizable design is found. A graph-based method is implemented in order to find a time sequence that realizes this optimal constrained design. These methods are illustrated on a numerical example of which the results are thoroughly discussed.

Key words: Optimal Input, Nonlinear Systems, Convex Optimization, Dispersion Function

1 Introduction

The quality of system models obtained through identification largely depends on the experimental conditions under which the measurement data was obtained. They determine the Fisher Information matrix, of which almost all quality criterion are a function. Therefore, experimental design is an important step in the identification process. One aspect of an experiment that can be optimized is the input signal that is used to excite the system.

For linear dynamic systems and nonlinear static systems the problem of optimal input design is well understood and well covered in literature [12,6,4]. For these systems it has been shown that the input design problem can be written as a convex optimization problem. As a result, a vast set of optimization tools can be used to solve these problems.

For linear dynamic systems with stationary signals operating in open loop, the Fisher Information matrix is an affine function of the input spectrum; with a few exceptions, the optimization is then performed by first computing the optimal input spectrum, and then constructing an input signal that has this optimal input spectrum. A few results are available where the optimization is performed directly with respect to the input signal in time-domain, as will be done in this paper; see e.g. [1].

For nonlinear systems the problem is often non-convex making global optimization more difficult, if not impossible. The main difficulty lies in the fact that the information content of the experiment is not only dependent on the second order moment of the input but also on higher order moments [7]. Finding an optimal input design method for the whole class of nonlinear systems seems therefore intractable. Here we shall present a method that applies to a given class of nonlinear systems and where the design is performed directly with respect to the time-domain input sequence.

1.1 System Class

The class of systems will be restricted to FIR-type nonlinear systems, meaning that the output at time t only depends on the current input and at most $n-1$ previous input samples, where n represents the length of the system memory. Such systems are already covered in [9], where the input design is formulated as a convex optimization with respect to the probability density of subsequences of length n.

A subclass of the finite memory nonlinear systems are

Email addresses: adecock@vub.ac.be (Alexander De Cock), Michel.Gevers@uclouvain.be (Michel Gevers), Johan.Schoukens@vub.ac.be (Johan Schoukens).

Wiener systems consisting of a FIR filter followed by a polynomial nonlinearity. The method presented in this work was already illustrated for this subclass of systems in [2]. Additionally, this subclass was also covered in [5], where the optimal input in the class of random Gaussian signals was obtained.

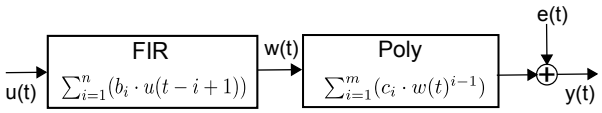

Fig. 1. FIR filter followed by a polynomial nonlinearity

1.2 Method

To transform the optimal input design to a convex optimization problem a special representation of the input is needed. First we shall consider that the input $u(t)$ can only take a finite set of possible values: $\{u_1, \ldots, u_A\}$. Considering the system model, it is clear that the output at time instance t only depends on the values of the input at t and $n-1$ previous instances. Therefore each n-tuple $(u(t-n+1), u(t-n+2), ..., u(t))$, in which each input takes one of the A possible values $\{u_1, \ldots, u_A\}$, is considered to be an elementary design for the system. There are A^n such elementary designs. The tuple frequency vector indicates how many times each one of these A^n tuples is present in the input sequence. The optimization problem will be proven to be convex with respect to the tuple frequency vector.
Instead of considering the general class of optimality criteria, as was done in [9], we restrict ourselves to D-optimal designs (meaning that the determinant of the Fisher Information matrix is maximized). This allows us to produce an algorithm used to perform the optimization. We have opted for a dispersion-based method, which was already successfully applied in the linear case [8]. Advantages of this method are its intuitive interpretation, straightforward implementation and monotonic convergence to the global minimum.
However, the optimal tuple frequency vector may not correspond to a realizable time sequence. To alleviate this problem, constraints need to be incorporated into the optimization problem. In order to do so, the tuple frequency vector will be expressed as a convex combination of a special set of basis vectors, which allows us to restrict the search space to the space of tuple frequency vectors that can be realized as a time signal.
Once a realizable and optimal tuple frequency vector is found, a time sequence satisfying this design needs to be derived. To this end a graph-based method is used. Such a method was already suggested in [9] and later elaborated in [11]. In this work a similar graph-based method is presented for deterministic input sequences.

1.3 Main Contributions

Because the problem statement and solution of this paper show strong similarities with [9], [3] and [11], we want to emphasize the main contributions of this work:

- Generalizing the dispersion-based optimization to nonlinear FIR-type systems.
- Providing a method to add constraints to this dispersion-based method.
- Discussing *deterministic* sequence generation and providing a graph-based generation algorithm.
- Illustrating the presented methods on a numerical example.

1.4 Overview

The remainder of this paper is structured as follows. Section 2 formalizes the optimal input design problem for the considered class of systems. Section 3 shows how the associated optimization problem can be solved based on the dispersion function. In Section 4 the problem of signal generation is considered, and a graph-based method is proposed. Section 5 discusses how the constraints needed for signal generation can be incorporated into the optimization. Section 6 illustrates the previous methods on a numerical example. To conclude, Section 7 will summarize the obtained results.

2 Problem Statement

The goal of this paper is to find a D-optimal input of some given length N for a nonlinear FIR-type system (as defined in Assumption 1) with a known model structure, and disturbed with independent Gaussian output noise. A D-optimal input is an input sequence of length N for which the determinant of the Fisher Information matrix is maximal. For a large number of samples, the average per sample covariance of the estimated parameter vector is proportional to the inverse of the Fisher Information matrix. When the parameters are identified which such an input sequence, the volume of the uncertainty ellipse in the parameter space is minimal [12]. The following assumptions describe this problem formally.

Assumption 1 *The considered system is a member of the class of nonlinear FIR-type systems with memory length n and which are differentiable with respect to the parameters of the system. This model class was first defined in [9].*

$$y_0(t,\theta) = G_{NL}(u_0(t), u_0(t-1), .., u_0(t-n+1), \theta) \quad (1)$$

where $u_0(t)$ is the noiseless input, $y_0(t,\theta)$ is the noiseless output, $\theta \in \mathbb{R}^{N_\theta}$ are the parameters of the model. Notice that the output at time t only depends on the current input sample and $n-1$ previous input samples. Additionally it is assumed that the system is identifiable with respect to the parameters, meaning that there exists an input sequence $u(1), \ldots, u(N)$ such that the outputs $\{y_0(t,\theta), t = 1, \ldots, N\}$ and $\{y_0(t,\theta_1), t = 1, \ldots, N\}$ of the corresponding models (1) are identical only if $\theta_1 = \theta$.

Assumption 2 *The class of inputs will be restricted to deterministic time sequences with a length of N samples. The amplitude can only take values from a finite, predefined set of A values:*

$$\forall t: \ u(t) \in \{u_1, ..., u_A\} \tag{2}$$

Remark 1 *No direct constraints are imposed on the total power of the input. However, limiting the maximal amplitude value and fixing the signal length restricts the total power indirectly.*

Assumption 3 *The output $y(t)$ of the true system is obtained as the sum of a noise-free output $y_0(t, \theta_0)$ defined by (1) with a "true" parameter vector θ_0 and some additive independent identically distributed (i.i.d) Gaussian noise $e(t)$. The noise is also independent of the input signal u_0.*

$$u(t) = u_0(t)$$
$$y(t) = y_0(t, \theta_0) + e(t)$$
$$e(t) \sim N(0, \sigma^2)$$

Remark 2 *The actual value of the variance does not play a role because it only scales the Fisher information matrix. This scaling will not alter the optimal input design.*

Assumption 4 *The estimator used to identify the model parameters is unbiased and efficient, which assures that the Cramer-Rao lower bound is reached.*

Assumption 5 *The D-optimality criterion is used, which means that the optimal input sequence u_{opt} of length N corresponds to the sequence for which the determinant of the information matrix M is maximal.*

$$u_{opt} = \arg\max_u(\det(M))$$

Given the noise assumption, M can be computed based on the time domain data as (see [10]):

$$M = \frac{1}{\sigma^2}\left(\frac{\partial y_0}{\partial \theta}\right)^T \left(\frac{\partial y_0}{\partial \theta}\right)$$

where $\frac{\partial y_0}{\partial \theta}$ is a $N \times N_\theta$ matrix containing the partial derivatives of y, and V^T stands for the transpose of V.

Remark 3 *Due to the nonlinear parametrization of the model structure, the determinant remains dependent on the true model parameters. During the computation of the optimal input, it will be assumed that the true parameters of the system, θ_0, are known. While this may seem in contradiction with the final goal of system identification it is a standard assumption in the field of optimal input design [12].*

3 Problem Solution

In order to solve the D-optimal input design problem, as presented in the previous section, three important steps will now be made. First, the concept of n-tuples and the tuple frequency vector are formally introduced. Second, it will be shown that the Fisher information matrix, corresponding to an input design as given by a tuple frequency vector, can be written as a convex combination with respect to the tuple frequencies; this property is the key to solve the optimization problem efficiently. Third and last, it will be shown how the problem can be solved with a dispersion-based method similar to the one used in the linear case, as described in [6,8].

3.1 Elementary Designs

Considering the system model, it is clear that the output at time instance t only depends on the values of the input at t and $n-1$ previous instances. Therefore each n-tuple $(u(t-n+1), u(t-n+2), ..., u(t))$ is considered to be an elementary design for the system. Because the possible amplitude values of the input are discretized, the number of elementary designs is finite.

Definition 1 *An ordered set of n values drawn out of the predefined set $\{u_1, u_2, ...u_A\}$ is called a n-tuple. In total A^n different tuples can be defined. All possible tuples will be stored in a multidimensional cell array called $C \in \{\mathbb{R}^n\}^{A^n}$ where each cell contains a n-tuple:*

$$\mathrm{C}(i_1, i_2, ..., i_n) = (u_{i_1}, u_{i_2}, ..., u_{i_n}) \tag{3}$$
$$\forall i_1, i_2, ...i_n \in \{1, 2, .., A\}$$

Notation 1 *Each index set $(i_1, ..., i_n)$ with $i_1, i_2, ...i_n \in \{1, 2, .., A\}$ can be made to correspond to a unique integer index k defined as follows:*

$$k \triangleq i_1 + \sum_{k=2}^{n}(i_k - 1) \cdot A^{(k-1)} \tag{4}$$

Notice that $(i_n, i_{n-1}, ..., i_1)$ is the representation of k in base A, that k ranges from 1 to A^n, and that (4) defines a one-to-one mapping between k and the index set $(i_1, i_2, ..., i_n)$. Thus, the mapping (4) establishes the equivalence:

$$k \Longleftrightarrow (i_1, i_2, ..., i_n) \tag{5}$$

In future we shall indistinctly use the notation $f(i_1, i_2, ..., i_n)$ or $f(k)$ for a function $f(.)$ of the index set, where k and $(i_1, i_2, ..., i_n)$ are related by (4).

Notation 2 *Let $(i_1, i_2, ..., i_n)$ denote the index set corresponding to k via the mapping (4); then $c(k)$ will denote the n-tuple $(u_{i_1}, u_{i_2}, ..., u_{i_n})$ defined by (3).*

Definition 2 *Given the set of discrete amplitudes* $\{u_1, u_2, ...u_A\}$*, each possible n-tuple is called an elementary design for a FIR-type system with memory length* n*.*

Definition 3 *The Fisher Information matrix* M_k *corresponding to the* k^{th} *tuple will be called the* k^{th} *elementary Fisher matrix:*

$$M_k(i,j) = \frac{1}{\sigma^2} f_{i,j}(c(k), \theta) \tag{6}$$

where the function $f_{i,j}(c(k), \theta)$ *corresponds to the product of the partial derivatives of* y_0 *with respect to the* i^{th} *and* j^{th} *parameter when the input sequence is the n-tuple* $c(k)$ *defined in Notation 2.*

For example, assume that $A = 2$ and $n = 3$ so that there are 8 n-tuples and k ranges from 1 to 8. Then the 5^{th} elementary Fisher Information matrix, say, is defined by

$$M_5(i,j) = \frac{1}{\sigma^2} \left(\frac{\partial y_0}{\partial \theta_i} \right) \left(\frac{\partial y_0}{\partial \theta_j} \right), i, j = 1, 2, 3$$

where these outputs are generated from the elementary input sequence (u_1, u_1, u_2), since $k = 5$ corresponds to the index set $(1, 1, 2)$.

3.2 Fisher Information and frequency vector

Now it will be shown that the Fisher information matrix can be expressed as a convex combination of the elementary Fisher matrices with the tuple frequencies as coefficients. Under assumptions 3 and 5 the information matrix obtained from a data sequence of length N can be written as a sum over the time samples:

$$M(i,j) = \frac{1}{\sigma^2} \sum_{t=1}^{N} f_{i,j}(u(t-n+1), ..., u(t), \theta) \tag{7}$$

Remark 4 *Notice that the first* $n-1$ *terms depend upon samples which are not measured. Their value is set by the initial conditions. It will be assumed that the signal is periodic with period* N*. This allows us to determine the values of these unknown samples.*

Because the functions $f_{i,j}$ only depend on n successive input values, the sum over time in (7) can be rearranged as a sum over all possible tuples:

$$\begin{aligned} M(i,j) &= \frac{1}{\sigma^2} \sum_{k=1}^{A^n} \xi_N(k) \cdot f_{i,j}(c(k), \theta) \\ &= \sum_{k=1}^{A^n} \xi_N(k) M_k(i,j) \end{aligned} \tag{8}$$

The weight $\xi_N(k)$ indicates how often the n-tuple $c(k)$ occurs in the sequence $u(t)$ and is therefore called the tuple frequency.

Definition 4 *The vector* $\xi_N(\cdot) \in \mathbb{N}^{A^n}$*, containing the number of times each n-tuple occurs in the input design is called the tuple frequency vector of the design.*

Remark 5 *Given a time sequence* $u(t)$*, the corresponding tuple frequency vector can be obtained by counting the number of times each tuple occurs in the signal* $u(t)$*. The tuples are counted as depicted in Fig.2. Notice that a signal with N samples contains N tuples, due to the assumed periodicity of the signal (see Remark 4).*

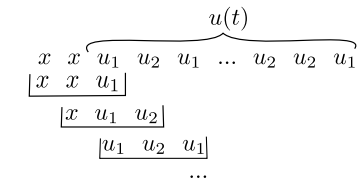

Fig. 2. Example of how the tuples are counted in the case of a FIR-type system with $n = 3$ and two amplitude values $\{u_1, u_2\}$. x represents an unknown sample which should be resolved by the chosen initial conditions.

Definition 5 *The tuple frequency vector divided by the total number* N *of tuples in the design is called the normalized tuple frequency vector* $\xi(\cdot) \in \mathbb{R}_+^{A^n}$*. By construction the normalized tuple frequencies have the properties of convex coefficients, meaning that their values range from 0 to 1 and that their sum is exactly one:*

$$\begin{aligned} &\forall k: \quad \xi(k) = \xi_N(k)/N \\ &\xi(k) \in [0,1] \quad \textit{and} \quad \sum_{k=1}^{A^n} \xi(k) = 1 \end{aligned} \tag{9}$$

Rewriting equation (8) as a matrix equation, and normalizing by the number of tuples N, allows us to rewrite the normalized Fisher information matrix as a convex combination of the elementary matrices with the normalized tuple frequencies as convex coefficients:

$$\frac{M(\xi_N)}{N} = \sum_{k=1}^{A^n} \xi(k).M_k \tag{10}$$

Notice that only the coefficients $\xi(k)$ depend upon the design. The elementary information matrices M_k are independent and can be computed a priori given the amplitude set A and the memory n of the FIR filter.

3.3 Dispersion Function

Computing an optimal experiment consists of finding the vector ξ that maximizes the determinant of the nor-

malized information matrix:

$$\xi_{opt} = \arg\max_{\xi}(det(\frac{M(\xi)}{N}))$$

In the previous subsection it was shown that the normalized Fisher information matrix can be rewritten as a convex combination of known elementary information matrices. We now show that this allows us to use a dispersion-based method in order to perform the optimization.
Instead of solving the optimization problem directly, an equivalent problem is solved where the maximum of an auxiliary function, called the *dispersion function*, is minimized. The dispersion function, also called *response dispersion* was introduced in the experiment design for identification arena in [6].

Definition 6 *With the notations introduced above, the dispersion function $v(.,.)$ is defined as:*

$$v(\xi, k) = \text{trace}(M(\xi)^{-1} \cdot M_k) \tag{11}$$

where $M(\xi)$ is the information matrix computed for the given design ξ, and M_k is the information matrix corresponding to the k^{th} elementary design.

Some useful properties of the dispersion function are [6]:

- The maximal value of the dispersion can never be smaller than the number of independent parameters in the model N_θ.
- For any design ξ, the inner product $\sum_{k=1}^{A^n} v(\xi, k) \cdot \xi(k)$ equals the number of free parameters in the model.
- The dispersion function can also be interpreted as a normalized variance of the estimated model.

Theorem 1 *The following characterizations of an optimal design are equivalent:*

(1) ξ_{opt} *maximizes* $\det(M)$
(2) ξ_{opt} *minimizes* $\max_k v(\xi, k)$
(3) $\max_k v(\xi_{opt}, k) = N_\theta$

where N_θ is the number of independent parameters in the model.
Proof: *see [6] Chapter 6, page 147.*

Theorem 1 states that the design that maximizes the determinant of the Fisher matrix is the same design that minimizes the maximum of the dispersion function. Since a simple and efficient algorithm exists that solves the latter problem, we shall adopt it for the computation of our optimal experiment.

3.4 Optimization Algorithm

In [8] a simple and robust, monotonically converging algorithm is presented, which finds the design that minimizes the maximum of the dispersion function. This algorithm can be summarized in four steps:

(1) Initialize with a uniform design: $\xi(k) = 1/A^n$
(2) Compute the dispersion function $v(\xi, k)$ for the current design using (11)
(3) Update the design in accordance with the dispersion function as follows $\xi_{new}(k) = \frac{v(\xi,k)}{A^n}.\xi_{old}(k)$
(4) Stopping criterion: if $(\max_k v(\xi_{new}, k) - N_\theta)$ is smaller than a predefined threshold, the optimal solution is assumed to be found; else go to step 2.

The stopping criterion is based on the third expression of Theorem 1. The monotonic convergence of the algorithm is proven in [13].

Remark 6 *Notice that once a particular tuple frequency $\xi(k)$ becomes zero for some k, this frequency remains zero for all subsequent iterations. Therefore, the computational speed of the algorithm can be improved by only updating the nonzero frequencies. This avoids unneeded evaluations of the dispersion function.*

4 Signal Generation

The optimal tuple frequency vector $\xi_{N,opt}$ can be interpreted as an experiment of N measurements, where each measurement consists of applying a single tuple to the system and measuring the corresponding output sample. A naive way to perform the optimal design is to concatenate all the tuples contained in $\xi_{N,opt}$ (i.e. all the tuples $c(k)$ for which $\xi_{N,opt}(k) \neq 0$), and only measure the output samples which correspond to these tuples. This means that a signal with nN samples is used at the input, in order to collect N samples at the output. Clearly this is not an efficient approach, since only N out of the nN output samples are used for parameter estimation. It would be better to generate a periodic input sequence with a period length of N samples, containing the N needed tuples. However, not every tuple frequency vector has a corresponding input sequence of length N because the $n-1$ last inputs of tuple $c(k)$ for which $\xi_{opt}(k) \neq 0$ may not correspond to the $n-1$ first inputs of another tuple $c(j)$ for which $\xi_{opt}(j) \neq 0$. In order to derive conditions on the tuple frequency vector which guarantee the existence of at least one time sequence, a sequence generation method will be introduced. This generation method will correspond with a path through the associated graph of the tuple frequency vector.

Definition 7 *Given a n-tuple, the right subtuple is defined as the (n-1)-tuple obtained by removing the first element of the original n-tuple. The left subtuple is defined as the (n-1)-tuple obtained by removing the last element of the original n-tuple.*

Definition 8 *The graph associated with a tuple frequency vector ξ_N (see Notation 1) is defined as follows:*

- *The graph contains A^{n-1} nodes, each containing a different (n-1)-tuple*

- *Each n-tuple corresponds to a directed edge connecting its left and right subtuple. The edge starts in the left subtuple and ends in the right.*
- *Each edge has a multiplicity which corresponds to the tuple frequency $\xi_N(i_1, \ldots, i_n)$ of its corresponding n-tuple.*

Example 1 *Consider a FIR-type system with $n = 3$ and two amplitude levels $\{u_1, u_2\}$. In total $2^{(3-1)}$ different tuples of length n-1 can be defined. So the associated graph contains four nodes. Additionally 2^3 different tuples of length n can be defined. This means that the graph contains eight edges. For example, the edge corresponding to the n-tuple $u_1u_1u_2$ connects its left subtuple u_1u_1 with its right subtuple u_1u_2. The multiplicity of the edge connecting u_1u_1 with u_1u_2 equals its tuple frequency $\xi_N(1,1,2)$; it is a scalar weighting. If this reasoning is repeated for every n-tuple, the graph in Fig.3 is obtained.*

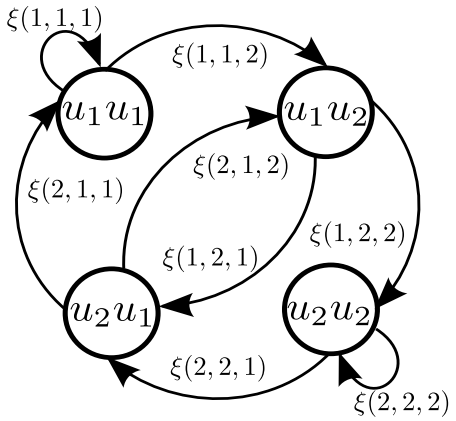

Fig. 3. Associated graph in the case of $n = 3$ and two amplitude values

If there exists a time sequence corresponding to the tuple frequency vector ξ_N, then this sequence can be obtained by the following steps:

(1) Construct the graph associated with the tuple frequency vector ξ_N
(2) Find a path, starting from an arbitrary node, that uses each edge exactly as many times as its multiplicity indicates. (see Appendix A for a path finding algorithm)
(3) Add the amplitude values of the starting node to the start of the sequence
(4) For every edge in the path, add the last amplitude value of the corresponding n-tuple to the end of the sequence.

From the above, it can be concluded that a time sequence exists if there exists a path, starting from an arbitrary node, that uses each edge exactly as many times as its multiplicity indicates. This requires that the following constraints on the tuple frequencies must be satisfied.

Theorem 2 *A periodic time sequence exists that realizes a prescribed tuple frequency vector ξ_N only if that frequency vector satisfies the following conditions:*

$$\forall i_1, i_2, ..., i_{n-1} \in \{1, ..., A\}$$
$$\sum_{j=1}^{A} \xi_N(j, i_1, ..., i_{n-1}) = \sum_{j=1}^{A} \xi_N(i_1, ..., i_{n-1}, j) \tag{12}$$

or equivalently, using the scalar index k rather than the vector index $(i_1, \ldots, i_n)$, as defined in Notation 1:

$$\forall m \in [1, \ldots, A^{n-1}]:$$
$$\sum_{j=1}^{A} \xi_N(j + (m-1)A) = \sum_{j=1}^{A} \xi_N(m + (j-1)A^{n-1}) \tag{13}$$

Proof: *A time sequence has the correct frequency vector ξ_N if a path can be constructed from the associated graph whereby each edge is used exactly as many times as its multiplicity indicates. In order for such a path to exist, the sum of the multiplicities of the outgoing and incoming edges needs to be equal in every node, where each node is defined by a $(n-1)$ vector index $(i_1, \ldots, i_{n-1})$. This is precisely the constraint (12). Now fix a $(n-1)$ tuple $(i_1, \ldots, i_{n-1})$, and let m denote the scalar index for this $(n-1)$ tuple, i.e.*

$$m = i_1 + (i_2 - 1)A + (i_3 - 1)A^2 + \ldots + (i_{n-1} - 1)A^{n-2}$$

We now express the two indices of length n appearing in (12) as a function of m:

$$\begin{aligned}(j, i_1, \ldots, i_{n-1}) &= j + (i_1 - 1)A + (i_2 - 1)A^2 \\ &\quad + \ldots + (i_{n-1} - 1)A^{n-1} \\ &= mA - A + j = j + (m-1)A\end{aligned}$$

With exactly the same procedure one gets

$$(i_1, \ldots, i_{n-1}, j) = m + (j-1)A^{n-1}$$

These conditions guarantee that the number of occurrences of a tuple that ends with a given $(n-1)$ subtuple $(u_{i_1}, u_{i_2}, ..., u_{i_{n-1}})$ in the time sequence equals the number of occurrences of a tuple that starts with $(u_{i_1}, u_{i_2}, ..., u_{i_{n-1}})$.

Remark 7 *In order to illustrate that the equations in (12) are not sufficient conditions for the existence of a realizable time sequence for a given a frequency vector, consider a disconnected graph which satisfies the constraints. In such a graph there is no single path connecting all the nodes, meaning there is also no corresponding time sequence.*

To find a tuple frequency vector that can be realized with a time sequence, these constraints should be taken into account during the optimization.

Remark 8 *In a stochastic framework, the tuple frequency matrix can be considered as mutual discrete probability distribution functions of the n stochastic variables. Imposing that the signal is stationary, will lead to the same constraint as given in (12) [9]. The graph described above can then be seen as a Markov chain used to generate a realization of the tuple frequency matrix.*

5 Constrained Optimization

In the previous section it was shown that a tuple frequency vector can only be realized as a time sequence if it meets the conditions (12). Therefore, these conditions will be imposed during the optimization. This can be done by adding constraints to the optimization problem, but unfortunately the dispersion-based algorithm cannot handle constraints.
Alternatively, the search space of normalized tuple frequency vectors will be restricted to a subspace of normalized tuple frequency vectors that obey (12). This is achieved by expressing a normalized tuple frequency vector as a convex combination of basis vectors that span such a space.

$$\forall k \in \{1, 2, \ldots, A^n\} : \ \xi_\gamma(k) = \sum_{j=1}^{N_b} \gamma(j).\xi_{b_j}(k) \qquad (14)$$

$$\text{with } \gamma(j) \in [0, 1] \text{ and } \sum_{j=1}^{N_b} \gamma(j) = 1$$

where $\xi_{b_j}(k)$ are the mentioned basis vectors, a possible choice of such vectors will be given below in Definition 9, N_b is the number of basis vectors, $\gamma(j)$ are the new convex coefficients with respect to which the optimization will be performed, and ξ_γ is the tuple frequency vector corresponding to the coefficient vector γ.
Remember that the tuple frequency vectors ξ_γ do not only have to obey the constraints expressed in (12) but they must also obey the properties of convex coefficients as stated in (9). These restrictions will translate into special properties of the basis vectors ξ_{b_j}. In the next section a possible choice for the set basis vectors is given.

5.1 Non-overlapping Symmetric Basis

The basis vectors that exactly describe the space of realizable tuple frequencies are unknown. Instead, a different set of basis vectors is used, of which the convex combinations describe a subspace of the space of realizable tuple frequencies.

Definition 9 *A set of vectors* $[\xi_{b_1}, \xi_{b_2}, ..., \xi_{b_{N_b}}]$ *in* $\mathbb{R}^{A^n}$ *is called non-overlapping symmetric if they have the following four properties:*

(1) positivity constraint:
$\forall k, \ \forall j : \xi_{b_j}(k) \geq 0$
(2) non-overlapping constraint:
$\forall k, \ \exists! j : \xi_{b_j}(k) > 0$
(3) unity sum constraint:
$\forall j : \sum_{k=1}^{A^n} \xi_{b_j}(k) = 1$
(4) symmetry constraint:
$\forall j, \ \forall i_1, ..., i_n, \ \forall p \in \ \mathrm{Perm}_{1,2,...,n} :$
$\xi_{b_j}(i_1, i_2, ..., i_n) = \xi_{b_j}(i_{p_1}, i_{p_2}, ..., i_{p_n})$

where N_b *indicates the number of basis vectors,* $\mathrm{Perm}_{1,2,...,n}$ *stands for the set of all possible permutations of the symbols* $\{1, 2, ..., n\}$*, and* $\exists! j$ *means 'there exists only one'.*

Remark 9 *The first property guarantees that the tuple frequencies are nonnegative. The second property guarantees that every tuple appears in only one basis vector, thereby making these basis vectors linearly independent. The third property makes sure that the sum of the tuple frequencies is one. The first three properties are needed in order to impose (9) on* ξ_γ*. The fourth property imposes symmetry on the tuple frequency vector. This symmetry constraint implies that the constraint (12) is satisfied.*

Remark 10 *The number of non-overlapping symmetric basis vectors* N_b *equals the number of degrees of freedom in a symmetric tensor of order n and dimensionality A. For example if* $n = 2$*,* $N_b = \frac{A(A+1)}{2}$*, which correponds to the degrees of freedom in a symmetric AxA matrix.*

Example 2 *In the case of* $n = 2$ *and two possible amplitude values (i.e.* $A = 2$*), the number of basis vectors is* $N_b = 3$*. The non-overlapping symmetric basis is given by the three frequency vectors below:*

$$\xi_{b_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \xi_{b_2} = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \\ 0 \end{bmatrix}, \ \xi_{b_3} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

It is easy to check that the four constraints are satisfied. Note that the fourth constraint reduces here to $\xi_{b_j}(1, 2) = \xi_{b_j}(2, 1)$ for $j = 1, 2, 3$ or, equivalently $\xi_{b_j}(2) = \xi_{b_j}(3)$ when the unique index k of (4) is used in lieu of (i_1, i_2).

5.2 Symmetric Elementary Designs

Assuming the use of a non-overlapping symmetric basis, the expression of the normalized information matrix can

be rewritten as a function of the weighting coefficients $\gamma(j)$ by substituting (14) into (10):

$$\frac{M}{N} = \sum_{j=1}^{N_b} \gamma(j) \sum_{k=1}^{A^n} \xi_{b_j}(k) \cdot M_k = \sum_{j=1}^{N_b} \gamma(j) \cdot M_{\gamma(j)} \quad (15)$$

where $M_{\gamma(j)} = \sum_{k=1}^{A^n} \xi_{b_j}(k) \cdot M_k$ are the new elementary information matrices.
During the constrained optimization, the dispersion function will be computed based on these $M_{\gamma(j)}$. We define:

$$v_\gamma(\xi_\gamma, j) \triangleq \text{trace}(M(\xi_\gamma(k))^{-1} \cdot M_{\gamma(j)}) \quad (16)$$

where the subscript γ indicates that v_γ is computed from the elementary designs $M_{\gamma(j)}$.
Notice that the values and dimensions of the dispersion functions $v_\gamma(\xi_\gamma, j)$ and $v(\xi_\gamma, k)$ are different. This is because the dispersion function v_γ evaluates the quality of the input design relative to the considered subspace of input designs described by the elementary designs, while v evaluates the quality relative to the full tuple frequency space.

5.3 Optimization Algorithm

By introducing the new elementary information matrices $M_{\gamma(j)}$ and the dispersion function v_γ, and as a result of the convex properties of the coefficients $\gamma(j)$, the dispersion-based algorithm described in Subsection 3.4 can be reused to find the $\gamma_{opt}(j)$ for which the determinant is maximal. Revisiting the 4-step algorithm leads to:

(1) Initialize with a uniform design: $\gamma(j) = 1/N_b$
(2) Compute the dispersion function $v_\gamma(\xi_\gamma, j)$ for the current design using (16)
(3) Update the design in accordance with the dispersion function as follows $\gamma_{new}(j) = \frac{v_\gamma(\xi_\gamma, j)}{N_b} . \gamma_{old}(j)$
(4) Stopping criterion: if $(\max_j v_\gamma(\xi_{\gamma_{new}}, j) - N_\theta)$ is smaller than a predefined threshold, the optimal solution is assumed to be found; else go to step 2.

6 Numerical Example

The methods above will now be illustrated on the following numerical example which is a member of the subclass discussed in subsection 1.1:

$$y(t) = c_1 w(t)^3 + c_2 w(t) + e(t)$$
$$w(t) = b_1 u(t) + b_2 u(t-1)$$
$$u(t) \in [-1 : 2/9 : 1] \quad \text{and} \quad e(t) \sim N(0,1)$$

with $b = (1; 3)$ and $c = (1; -0.25)$. The value of a_1 will be fixed in order to make the problem identifiable.

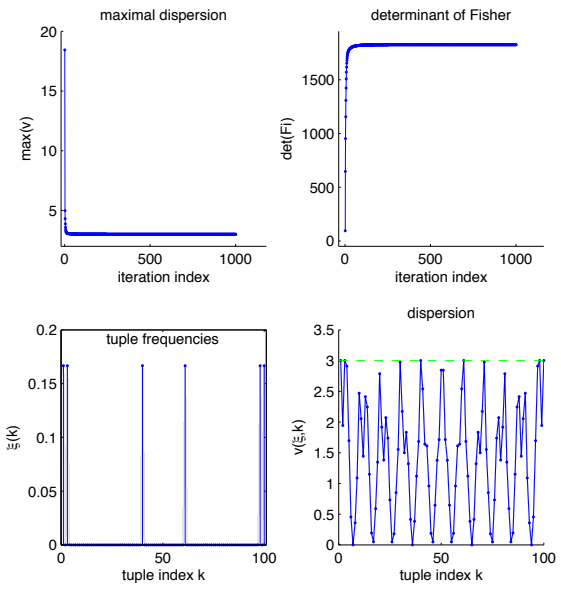

Fig. 4. Results without constraints. Top, left: maximal dispersion as function of the iteration index Top, right: determinant of the normalized information matrix function of the iteration index. Bottom,left: normalized tuple frequencies. Bottom,right: dispersion function of the optimal design.

The amplitude set contains 10 uniformly spaced values between -1 and 1 (A=10). Because the system has a memory length of two (n=2), 100 different tuples should be considered.

6.1 Unconstrained optimization

First, the unconstrained optimization is performed. This means that the normalized tuple frequencies $\xi(k)$ are optimized with the dispersion-based optimization method described in subsection 3.4. The results for 1000 iterations are plotted in Fig.4. In each iteration step the dispersion function, normalized tuple frequency vector and determinant of the normalized information matrix are computed and stored.
The top plots represent the evolution of the maximum dispersion and of the determinant of the Fisher information matrix as a function of the iterations. They show that the method successfully lowers the maximal value of the dispersion and at the same time increases the determinant of the information matrix. Note that in the end the maximal dispersion reaches the value of 3, which corresponds to the number of free parameters. This indicates that the obtained solution is optimal (see Theorem 1).
The optimal normalized tuple frequency vector at the end of the iterations is depicted in the bottom left plot; it only contains 6 entries that are different from zero. This means that the optimal design is such that only 6 out of the 100 possible tuples are used to excite the system. Each of them has the same frequency value. The

tuple index	tuple	frequency
1	[-1;-1]	1/6
3	[-1;-10/18]	1/6
40	[-1/3,1]	1/6
61	[1/3;-1]	1/6
98	[1;10/18]	1/6
100	[1;1]	1/6

Table 1
Table containing the tuple index, tuples and normalized tuple frequencies of the optimal unconstrained design ξ_{opt}.

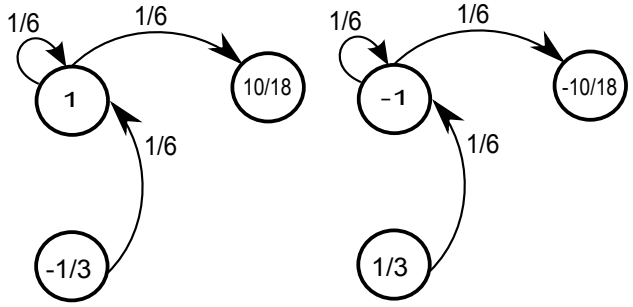

Fig. 5. Associated graph of the optimal unconstrained design.

bottom right plot represents the value of the dispersion function for each of the 100 tuples.
Table 1 shows the tuples of the optimal unconstrained design and their corresponding normalized frequencies, while Fig. 5 represents the associated graph. Two things should be noted. First, the design does not obey the constraints (12). As a result, the tuple frequency vector cannot be realized as a time sequence. Second, the graph is disconnected, which means that not every node can be reached from any other node.

6.2 Evolution of the Determinant

Now let us have a closer look at the evolution of the determinant in Fig.4. For the first 10 to 20 iterations there is a rapid increase in the determinant value. During these iterations, the number of different tuples is reduced drastically. After this rapid change, the evolution of the determinant value becomes stable. Only the frequencies of the remaining tuples are changed but the selection of tuples stays the same. From this observation it can be concluded that selecting the optimal subset of tuples is more important for the quality of the design than finding the optimal frequencies of the selected tuples.

6.3 Constrained optimization

Next, the optimization is performed using the non-overlapping symmetric basis vectors obeying the constraints of Definition 9. This means that the optimization is performed with respect to the coefficients $\gamma(j)$ of

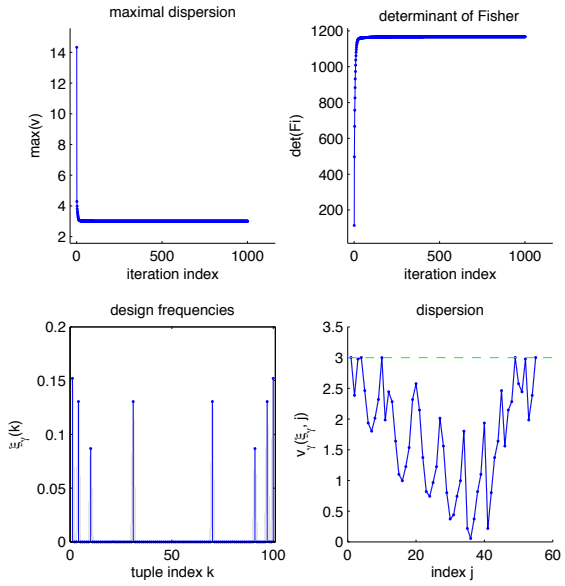

Fig. 6. Results with constraints. Top, left: maximal dispersion. Top, right: determinant of the normalized information matrix. Bottom,left: normalized tuple frequencies of the optimal design. Bottom,right: dispersion function at the end of the iterations for each of the 55 tuples.

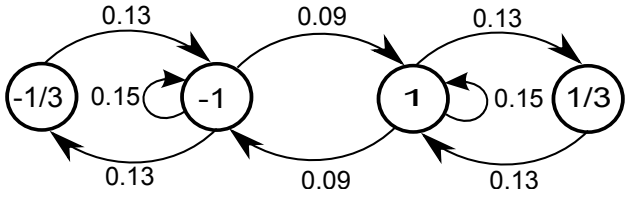

Fig. 7. Associated graph of the optimal constrained design.

(15) using the elementary information matrices $M_{\gamma(j)}$. Again 1000 iterations are taken which leads to the plots in Fig.6. Due to the symmetry constraint, only $N_b = 55$ coefficients need to be considered; hence $\gamma(j)$ in the bottom right plot ranges from 1 to 55. The determinant increases monotonically while the maximal value of the dispersion $\max(v_\gamma(\gamma_{opt}, j))$ is driven to its minimal value of 3. This indicates that the final design is optimal in its subspace of constrained designs.
Table 2 shows the normalized tuple frequencies of the optimal constrained design which contains eight different tuples with different frequencies. As expected the design is symmetric, meaning that the tuples $[u_1, u_2]$ and $[u_2, u_1]$ have the same frequency. This allows to concatenate the tuples without the need of unwanted transition tuples. The same conclusion can be made from the associated graph in Fig. 7.

tuple index	tuple	frequency
1	[-1;-1]	0.15
4	[-1;-1/3]	0.13
10	[-1,1]	0.09
31	[-1/3;-1]	0.13
70	[1/3;1]	0.13
91	[1;-1]	0.09
97	[1;1/3]	0.13
100	[1;1]	0.15

Table 2
Table containing the tuple index, tuples and normalized tuple frequencies of the optimal constrained design $\xi_{\gamma_{opt}}$.

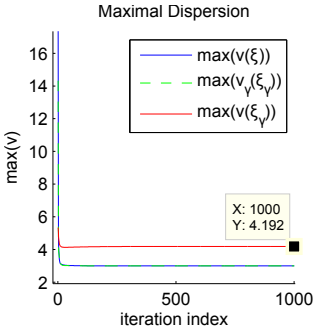

Fig. 8. Maximal dispersion of the normalized information matrix in the constrained case, computed for different elementary designs.

6.4 Computing the Dispersion

At first sight it seems contradictory that the optimal unconstrained and constrained design have the same maximum dispersion but different determinant values. However, the dispersion of the constrained design $v_\gamma(\xi_\gamma, j)$ and the dispersion of the unconstrained design $v(\xi, k)$ can not be compared directly, because they are based on different elementary designs. In order to make a comparison possible, the dispersion of the constrained solution $\xi_{\gamma,opt}$ is computed with respect to the elementary Fisher matrices M_k.
The results are plotted in Fig.8. From the left plot it is clear that the dispersion function $v(\xi_\gamma)$ is larger than $v_\gamma(\xi_\gamma)$ and $v(\xi)$. This indicates that the constrained solution is not optimal in the full tuple frequency space. This is in accordance with the observation that the determinant of the constrained design $\xi_{\gamma_{opt}}$ is smaller than the determinant of the unconstrained design ξ_{opt}. See Table 6.6 for the exact determinant values.

6.5 Signal Generation

Both designs will now be translated into a time sequence containing 100 tuples. During this process three approximations are considered.

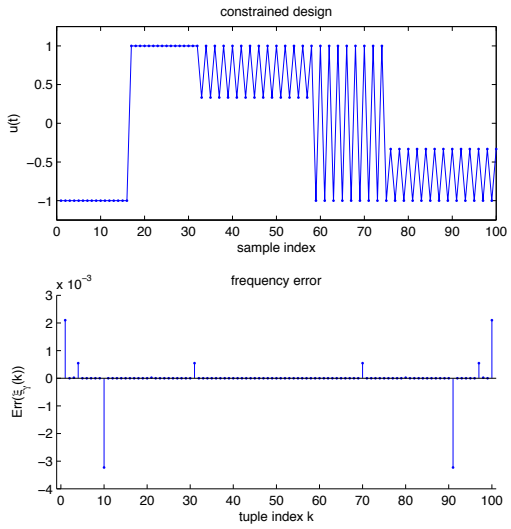

Fig. 9. Input sequence with constraints. Top: time domain signal; Bottom: difference in tuple frequency between optimal design and time sequence.

- After denormalization, the values of ξ_N are rounded to the nearest natural number.
- The unconstrained design needs additional transient tuples because condition (12) is violated.
- The constrained design needs additional transient tuples when the graph is disconnected.

All these approximations lead to a decrease in the determinant of M and alter the total tuple count, making the resulting time sequences suboptimal.
First, the time sequence for the constrained design is generated. This can be done with the path finding algorithm discussed in Appendix A. The resulting input sequence can be found in the top plot of Fig 9. The bottom plot depicts the difference in frequency between the optimal normalized tuple frequency vector ξ_{opt} and the generated time sequence. All errors are smaller than 10^{-2}, meaning only rounding errors are present.
The algorithm of Appendix A can not be used for the unconstrained design because the design does not satisfy the constraints (12). Therefore, the design will be slightly altered in order to find a time sequence which realizes the unconstrained design as well as possible. The graph of the altered design can be found in Fig. 10. Notice that the graph is no longer disconnected and satisfies the conditions in (12).
Now the path finding algorithm from Appendix A can be applied. The results can be found in Fig.11. The generated sequence contains 104 samples due to roundoff errors. If a sequence of exactly 100 samples is needed some $[1, 1]$ and $[-1, -1]$ tuples could be removed. The positive frequency errors reflect the change in tuple frequency from the original tuples. The negative errors reflect the addition of the dotted arrows to the design (see Fig.10).

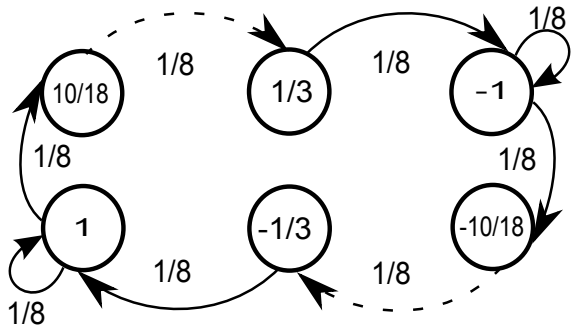

Fig. 10. Modified graph from the unconstrained design. Dotted arrows were added and the tuple frequencies were renormalized.

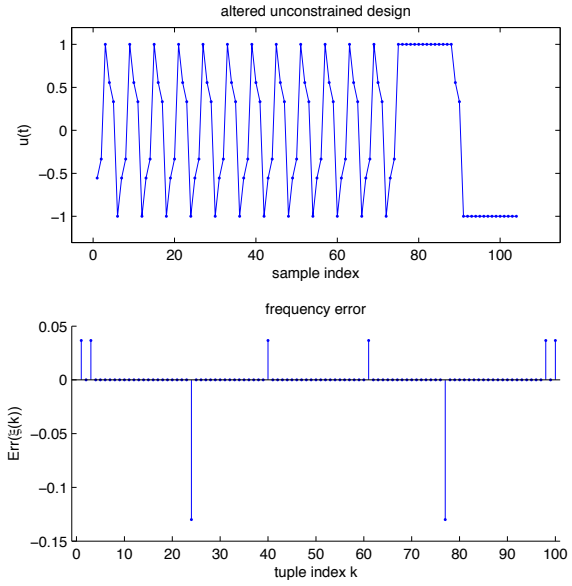

Fig. 11. Input sequence obtained from altered unconstrained design. Top: time domain signal; Bottom: difference in tuple frequency between optimal design and time sequence for the denormalized frequency vector.

6.6 Comparing Designs

Table 3 summarizes the performance of all previous designs. In order to remove the influence of the signal length, the Fisher matrix is computed based on the normalized tuple frequencies. As a reference for comparison, the maximum determinant out of 1000 randomly generated signals is also added.
Both the constrained and unconstrained designs perform two orders of magnitude better than the best randomly generated design. Out of the optimized designs, the unconstrained design has the highest determinant value. When this design is altered and realized as a time sequence, a decrease in the determinant can be observed, but the resulting sequence still outperforms the constrained design. Notice that there is no difference in determinant value between the constrained design and its corresponding time sequence.
From these observations it can be concluded that it is meaningful to compute both the constrained and unconstrained design. If the unconstrained design can easily be altered to a realizable design, without too much loss in performance, it should be preferred. If not, the constrained design presents a valuable alternative, because it can always be realized using the path finding algorithm in Appendix A.

solution type	det(Fi)
max random signal	9.51e+01
unconstrained design	1.83e+03
unconstrained sequence	1.37e+03
constrained design	1.17e+03
constrained sequence	1.17e+03

Table 3
Normalized determinants of all considered designs and there corresponding time sequences

7 Conclusion

In this work a solution to the problem of D-optimal input design for nonlinear FIR-type systems with an input taking a finite set of possible values has been presented. By expressing the optimization problem with respect to the tuple frequency vector, instead of the time sequence the problem became convex.
This convex problem was solved with an optimization scheme based on the dispersion function. However, it turned out that additional constraints are needed in order to guarantee that the optimal design can be realized as a time sequence. By imposing that the solution should lie in the subspace described by a symmetric and non-overlapping basis, a realizable solution was obtained that is optimal in its subspace of constrained solutions.
In order to find a time sequence that realizes this optimal constrained design, the associated graph was introduced. It was shown that a path, using all edges in the graph as many times as their multiplicity indicates, corresponds to a time sequence that realizes the design.
The methods presented in this paper were applied on a simple numerical example. Comparing the realization of the constrained design with the realizations of a random and unconstrained design showed that the determinant was highest for the unconstrained design.
However, it can not be guaranteed that this design is realizable without a significant loss in determinant value. Therefore, the constrained design is proposed as an attractive alternative, because it can always be realized.

Acknowledgements

This work was supported in part by the Fund for Scientific Research (FWO-Vlaanderen), by the Flemish

Government (Methusalem), by the Belgian Government through the Inter university Poles of Attraction (IAP VII) Program, and the ERC Advanced Grant SNLSID.

References

[1] Brian L. Cooley and Jay H. Lee. Control-relevant experiment design for multivariable systems described by expansions in orthonormal bases. *Automatica*, 37(2):273–281, February 2001.

[2] A. De Cock, M. Gevers, and J. Schoukens. A preliminary study on optimal input design for nonlinear systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 4931–4936, Dec 2013.

[3] M. Forgione, X. Bombois, P.M.J. Van den Hof, and H. Hjalmarsson. Experiment design for parameter estimation in nonlinear systems based on multilevel excitation. 2014. Submitted to 2014 European Control Conference.

[4] M. Gevers, X. Bombois, R. Hildebrand, and G. Solari. Optimal experiment design for open and closed-loop system identification. *Communications in Information and Systems*, 11(3):197–224, 2011.

[5] M. Gevers, M. Caenepeel, and J. Schoukens. Experiment design for the identification of a simple Wiener system. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 7333–7338, Dec 2012.

[6] G.C Goodwin and R.L. Payne. *Dynamic System Identification: Experiment Design and Data Analysis*, volume 136 of *Mathematics in Science and Engineering*. Academic Press, 1977.

[7] H. Hjalmarsson, J. Martensson, and B. Ninness. Optimal input design for identification of non-linear systems: Learning from the linear case. In *American Control Conference, 2007. ACC '07*, pages 1572–1576, July 2007.

[8] J.Schoukens and R.Pintelon. *Identification of Linear Systems: A Practical Guideline to Accurate Modeling*. Pergamon Press, 1991.

[9] C.A. Larsson, H. Hjalmarsson, and C.R. Rojas. On optimal input design for nonlinear fir-type systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7220–7225, Dec 2010.

[10] R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. John Wiley & Sons, 2004.

[11] P.E. Valenzuela, C.R. Rojas, and H. Hjalmarsson. Optimal input design for non-linear dynamic systems: A graph theory approach. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 5740–5745, Dec 2013.

[12] V.V.Federov. *Theory of Optimal Experiments*. Academic Press, 1972.

[13] Y. Yu. Monotonic convergence of a general algorithm for computing optimal designs. *The Annals of Statistics*, 38:1593–1606, 2010.

A Appendix: Path Finding

In order to generate an optimal input sequence that realizes the optimal tuple frequency vector, which meets condition (12), a path needs to be found in its associated graph. This path has to use all edges in the graph exactly as often as their multiplicity indicates.

A.1 Basic Concepts

Before the path finding algorithm can be explained, some concepts of graph theory need to be introduced.

Definition 10 *A path in the associated graph is an ordered set of nodes $\{n_1, n_2, ..n_k\}$, in which each successive pair of nodes $\{n_i, n_{i+1}\}$ is connected by an edge starting in n_i and ending in n_{i+1}. Each edge can not be used more than its multiplicity indicates. The first node of the path n_1 is called the starting node, the last node of the path n_k is called the end node. k is called the length of the path.*

Definition 11 *A cycle in the associated graph is a path for which the start and end node are the same. An open cycle is the path that we get by omitting the end node from a cycle. Two cycles are distinct if there is no cyclic permutation that maps their corresponding open cycles.*

Definition 12 *An elementary cycle in the associated graph is a cycle where all nodes are unique with the exception of the start and end node. Two elementary cycles are overlapping if they contain at least one mutual node.*

Theorem 3 *If the tuple frequency vectors satisfy equation (12), then every edge is part of an elementary cycle.*

PROOF: *First, let us eliminate all elementary cycles from the graph. This results in a graph with the same nodes, but which only contains the edges that are not part of an elementary cycle. Note that this remaining graph still has to obey the constraints (12).*
The theorem is now proved by contradiction. Assume that there is at least one node which contains an outgoing and incoming edge that is not part of an elementary cycle. It should be noted that it is not possible that there is a node that contains only one edge which is not part of an elementary cycle because this would violate (12).
These edges should be connected to another node. Connecting the edges to the same node is not possible because this would create an elementary cycle containing one node; therefore the edges would not be present. It must be concluded that the edges are connected to two other nodes.
However, these connections imply that these other nodes also have a pair of edges which are not part of an elementary cycle due to (12). As a result each connected edge leads to a new edge that needs to be connected. Note that these edges can not connect with a previously encountered node because doing so introduces an elementary cycle which is in contradiction with the removal of all elementary cycles in the graph.
Because the number of nodes in the graph is finite, the above process of connecting edges will lead to edges that can no longer be connected to any node. This is in contradiction with the definition of a graph, meaning that the assumption that there are at least two edges which are not part of an elementary cycle is false.

A.2 Path Finding Algorithm

Based on the theorem above it can be concluded that the wanted path consists of only elementary cycles. As a result the following path finding algorithm is proposed. First, all elementary cycles in each node are computed. This can be done with a simple backtracking algorithm. Duplicated cycles are omitted and the remaining distinct elementary cycles are sorted by length.
Next, all elementary cycles are combined into one path by inserting the smaller elementary cycles into the largest elementary cycle. This combined path defines the optimal input sequence.
A special case may occur where not all elementary cycles can be inserted into one path. In this case the associated graph is disconnected and no optimal input sequence exists. However, by finding a path for each subgraph and putting these paths in succession, a suboptimal path can be found with minimum number of unwanted transition tuples.
Pseudo code describing the algorithm can be found below. The variable *eC* is the list of elementary cycles and *path* is the current path. In the case of a disconnected graph, *path* will contain the paths for each subgraph.

Algorithm 1 Path Finding

```
for k=1:length(node) do
  eC = [eC, computeElementaryCycles(node(k))];
end for

eC = removeDuplicates(eC)
eC = sortOnLength(eC)
nrSubgraph = 1
path(nrSubgraph ) = eC(1)
eC = remove(eC(1),eC)

while not isempty(eC) do
  while eC was changed do
    for k=1:length(eC) do
      if insertable(eC(k),path(nrSubgraph)) then
        path(i)=Insert(eC(k),path(nrSubgraph))
        eC=remove(eC(k),eC)
      end if
    end for
  end while

  if not isempty(eC) then
    nrSubgraph = nrSubgraph +1
  end if

end while
```