A NEW FACTORIZATION OF SPECIAL NON-LINEAR DISCRETE SYSTEMS AND ITS APPLICATIONS

Stelios Kotsios
University of Athens, Department of Economics
Pemazoglou 8, 10559 Athens, Greece
e-mail: skotsios@di.uoa.gr

February 22, 2004

Keywords: Algebraic approaches, Factorization methods, Non-linear Systems, Discrete Systems.

Abstract

In this paper a new factorization of discrete non-linear systems is presented. This factorization is based on the star-product, an operation corresponding to the linear connection of systems. A relevant algorithm is developed, based on a theoretical background. Finally, some applications to system analysis, BIBO stability and feedback design are provided.

1 Introduction

A variety of mathematical tools has been used for the analysis and design of control systems. Among these tools algebraic methods were first presented by Kalman [10]. Factorization is a popular algebraic method dealing with
linear as well as non-linear systems. In linear systems the central idea is that of factorizing the transfer matrix as the ratio of two stable rational matrices. This idea was first introduced by Vidyasagar in [22] and subsequent results were the parametric description of stabilizers and robust controllers of a system ([2], [23]).

In non-linear systems, constructions of coprime factorization have been studied of late in the literature ([9], [19]). This setting was applied to systems which can be described in terms of input-output representation, and the parameterization of the system stabilizers was mainly based on the existence of solutions of certain Diophantine equations. Extensions of these results, in which the description of the stabilizers does not need the solution of the Bezout identity, are also available ([21]).

In this paper we are dealing with non-linear discrete systems of the form:

\[ y(t) + \sum a_i y(t-i) + \sum a_{ij} y(t-i) y(t-j) + \cdots + \sum a_{i_1 \ldots i_n} y(t-i_1) \cdots y(t-i_n) = \]

\[ = \sum b_i u(t-i) + \sum b_{ij} u(t-i) u(t-j) + \cdots + \sum b_{i_1 \ldots i_n} u(t-i_1) \cdots u(t-i_n) \] (1)

Our approach is along the traditional path of describing these systems using proper operators ([3], [18], [20]). It is an extension of a method used for linear discrete systems. Specifically, using a simple shift operator, named \( q \), linear discrete systems were described using polynomials of one variable. The common product of these polynomials corresponds to the substitution of one polynomial into another and thus describes the linear connection of linear systems. This set up leads to the solution of problems with special applications to the adaptive control [8].

In our case we use the so-called \( \delta \)-operator. This operator is analogous to the multi \( z \)-transform and to formal powers series ([3], [18]) and permits us to study non-linear discrete systems through polynomials of several variables. By this the model-matching problem was faced and some BIBO criteria were developed for specific discrete non-linear systems, ([13], [14]). Non-linear adaptive controllers were also studied in [11]. Using this operator we can rewrite system (1) in the form \( Ay(t) = Bu(t) \), where \( A \) and \( B \) are polynomials with many variables, named \( \delta \)-polynomials. Among these polynomials we define two product operations: the dot-product, which corresponds to the usual product among polynomials, and the star-product which corresponds to the substitution of one polynomial into another, or, in a more near-to-systems language, to the linear connection of systems.
Coherent with the star-product is the factorization we develop here. The intuitive appeal of this notion is that we can discover the elementary subsystems of which, a polynomial discrete non-linear system consists. This is of prime importance, since after that we can transfer properties of that system to its elementary factors. The problem of complete star-factorization is of extreme difficulty and has not yet been solved. The aim of this paper is to introduce a new algorithm which presents a next step to the approach of this factorization problem. It is an extension of more simple factorizations, already applied to the problem of model matching [14], or to the realization problem [12]. The whole procedure is mainly based on the dot-product factorization problem or in other words, on the factorization of polynomials of many variables according to the usual product. We consider the answer to this problem as given.

Despite the similarity of δ-operator to other algebraic tools, there are some peculiarities. Specifically, the set of δ-polynomials is not a ring, with respect to the star-product, and therefore our approach is completely different to previous ([1],[17]) or current methods ([4], [5], [6], [7]). For instance, in [5] the same symbol δ is used for the description of linear and non-linear dynamics through the notion of difference algebra. All the sets used at [5] are rings, something which is not valid in our case. This makes our method a new tool. The invertibility of systems studied in [5] is reduced in our case to the invertibility of δ-series whilst realization problems are currently under research.

Applications to the system simplification-linearization are presented, where we can see that some non-linear systems are essentially linear. Applications to systems cascade connection are also studied. A BIBO stability result is provided and a simple non-linear model matching problem has been chosen to indicate the applicability of our method to the feedback design. The whole configuration is formal in nature and no internal stability problems were taken into consideration here. This will be the subject of future research together with the applications of our results to robust control problems and complicated realizations. The main aspect of our approach is its total computational orientation, which permits a variety of problems to be examined via the correct software. Throughout this paper N, Z, R will denote the set of natural, rational and real numbers, correspondingly.
2 The $\delta$-operator

The aim of this section is to describe the notion of the $\delta$-operator on which all our further analysis is based. This notion was first introduced in [14] to describe non-linear polynomials. For the sake of completeness we shall briefly review these definitions. For any further details the reader is referred to [14]. In the set of all sequences $N^Z$, we consider the subset $\Delta$ of all finite zero sequences of the form $Z \to N$. We shall use the elements of $\Delta$ as "operators" acting on the sequences $F = \{x : Z \to R \text{ where } x(t) = 0$, for $t < 0\} \subset R^Z$, a set arising from the sampling of continuous functions. Explicitly: given $\delta \in \Delta$ and $x \in F$, we write $\delta = \{l_i\}_{i \in Z}$ where $l_i \neq 0$ only for a finite subset $J \subset Z$ of $i$'s and define: $\delta x(t) = \prod_{i \in J} x^{l_i}(t-i)$. For the sake of fullness we use the symbol $\delta_e$ to denote the "null" element, i.e. $\delta_e = \{0\}$. Obviously $\delta_e \{x(t)\} = 1$. Many times we identify $\delta_e$ with the symbol "1". The quantity $\sum_{i \in Z} l_i$ is called the degree of $\delta$ and it is denoted by $\deg(\delta)$.

Example 2.1 (a) If $\delta = \{l_1, l_2, l_3\}$ where $l_1 = 2, l_2 = 3, l_4 = 7$ then $\delta \ y(t) = y^2(t-1)y^3(t-2)y^7(t-4), \deg(\delta) = 12$.

(b) If $\delta' = \{l_{-2}, l_0, l_6\}$ where $l_{-2} = 2, l_0 = 3, l_6 = 1$ then $\delta' \ y(t) = y^2(t+2)y^3(t)y(t-6), \deg(\delta') = 6$

For each $\delta \in \Delta$ there is a uniquely defined vector; $i = (-k, \ldots, -k \ldots 0, \ldots, 0 \ldots 3, \ldots, 3 \ldots m, \ldots, m)$, known as a multindex [16]. The identification of $\delta$ and $i$ is indicated in the symbol $\delta_1$. In the various calculations we also make use of the notation:

$$
\delta_1 = \delta^{-k} \delta^{-k} \delta^{-k} \delta^{-3} \delta^{-3} \delta_0 \delta_0 \delta_0 \delta_3 \delta_3 \delta_3 \cdots \delta_m \delta_m =
$$

$$
= \delta^{-k} \delta^{-k} \delta^{-3} \delta_0 \delta_0 \delta_3 \delta_3 \cdots \delta_m
$$

It is clear that the degree of $\delta$ coincides with the dimension of its multindex, in other words $\deg(\delta) = \dim(i) = n$. Obviously $\delta_i, i \in Z,$ is nothing else than the well known simple shift operator, i.e. $\delta_i x(t) = x(t-i)$. 

4
Example 2.2  The operators in the previous example are rewritten as follows:

\[ \delta y(t) = \delta_1^2 \delta_2^3 \delta_4^7 y(t) = y^2(t - 1)y^3(t - 2)y^7(t - 4) = \delta_{(1,1,2,2,2,4,4,4,4,4,4,4,4,4,4)} y(t) \]

\[ \delta' y(t) = \delta_{-2}^2 \delta_0^3 \delta_0 y(t) = y^2(t + 2)y^3(t)y(t - 6) = \delta_{(-2,-2,0,0,0,6)} y(t) \]

Obviously \( \dim \{ \delta \} = 12 \) and \( \dim \{ \delta' \} = 6 \)

We recall now, that a set \( I \) of multindices may be ordered in a lexicographical way as follows: Let \( i = (i_1, i_2, \ldots, i_n) \) be a multindex. We define \( \sigma_{1n} = \min \{ i_1, i_2, \ldots, i_n \} \), \( \sigma_{1,n-1} = \min \{ \{ i_1, i_2, \ldots, i_n \} - \sigma_{1n} \}, \ldots, \sigma_{1,n-k} = \min \{ \{ i_1, i_2, \ldots, i_n \} - \{ \sigma_{1n}, \sigma_{1,n-1}, \ldots, \sigma_{1,n-k-1} \} \}, k = 1, \ldots, n - 1. \)

We say now that the multindex \( i \) is "less" than the multindex \( j \), and we write \( i < j \) if either \( \dim \{ i \} < \dim \{ j \} \) or \( \dim \{ i \} = \dim \{ j \} \) and \( \sigma_{1m} < \sigma_{1,m} \), for some positive integer \( m \) : \( 1 \leq m \leq n \). Since for any two multindices we can immediately deduce which multindex is "larger" and which "smaller", the above lexicographical order is well defined. For instance \( (9,9) < (1,2,1,8) < (1,2,2,3) \). Using this notion we can obtain the order:

\[ \delta_0 \preceq \delta_1 \preceq \delta_2 \preceq \cdots \preceq \delta_k \preceq \cdots \preceq \delta_0 \delta_0 \preceq \delta_0 \delta_2 \preceq \cdots \]

\[ \preceq \delta_1 \delta_k \preceq \cdots \delta_1 \delta_2 \preceq \cdots \preceq \delta_0 \delta_0 \delta_0 \delta_2 \delta_2 \delta_2 \delta_2 \preceq \cdots \]

We equip the set \( \Delta \) with two internal operations. Indeed, let \( a = \{ a_i \}, b = \{ b_i \} \in \Delta, i \in Z \), their dot-product, \( a \cdot \mathbf{b} \), is the operator \( c = \{ c_i \} \) such that \( c_i = a_i + b_i, \forall i \in Z \) and their star-product would be defined as \( a * \mathbf{b} = \{ c_i \} = \{ \sum_{n=\infty}^{+\infty} a_n b_{i-n} \} \). The dot-product corresponds to the usual product among sequences whilst the star-product to the substitution of one sequence into another.

Example 2.3  For the above operators we have:

(a) \( \delta \cdot \delta' = \{ c_{-3} a_0 c_1 c_3 c_4 c_6 \} \) where \( c_{-3} = 2, c_0 = 2, c_1 = 3, c_3 = 3, c_4 = 8, \)

\( c_6 = 1 \) and

\[ \delta \cdot \delta' x(t) = x^2(t + 3)x^2(t)x^3(t - 1)x^3(t - 3)x^8(t - 4)x(t - 6) \]

(b) \( \delta \ast \delta' = \{ c_{-3} c_0 c_1 c_3 c_4 c_7 c_9 \} \) where \( c_{-3} = l_1 l_{-3} = 6, c_0 = l_3 l_{-3} = 6, c_1 = l_1 l_0 + l_4 l_{-3} = 22, c_3 = l_3 l_0 = 6, c_4 = l_4 l_0 = 16, c_7 = l_1 l_6 = 3, c_9 = l_3 l_6 = 3, c_{10} = l_4 l_6 = 8 \) and

\[ \delta \ast \delta' x(t) = x^6(t + 2)x^6(t)x^{22}(t - 1)x^6(t - 3)x^{16}(t - 4)x^3(t - 7)x^3(t - 9)x^8(t - 10) \]
There is also an external operation, named addition, defined as follows:
\[(\delta + \hat{\delta})\{u(t)\} = \delta u(t) + \hat{\delta} u(t).\]

In order to give compact formulas for the star and dot-products, we need the following operators among their indices. Given two multindices \(i = (i_1, i_2, \ldots, i_k)\) and \(j = (j_1, j_2, \ldots, j_\lambda)\), the new multindex \(i \oplus j\) is defined just juxtaposing \(j\) after \(i\). Explicitly: \(i \oplus j = (i_1, j_1, i_2, j_2, \ldots, i_k, j_\lambda)\) where \(i_1 \leq j_1 \leq i_2 \leq j_2 \leq \cdots \leq i_k \leq j_\lambda\). We define the pointwise sum \(j + i\) as follows: Let \(j = (j_1, j_2, \ldots, j_m)\) be a multindex then we set: \(j + i = (j_1 + i, j_2 + i, \ldots, j_m + i)\).

For all the above operations the following properties are valid. Their proofs are obvious from the definitions and therefore omitted.

**Proposition 2.1** The following properties are valid: (I) \(\delta \ast \delta' = \delta' \ast \delta\), (II) \(\delta \cdot \delta' = \delta' \cdot \delta\), (III) \(\delta_{i_k} \ast \delta = \delta_{i_k}\), (IV) \(\delta \ast (\delta' \cdot \delta'') = (\delta \ast \delta') \cdot (\delta \ast \delta'')\), (V) \(\delta \cdot (\delta' \ast \delta'') = (\delta \cdot \delta') \ast (\delta \cdot \delta'')\), (VI) \(\delta + \delta' \ast \delta'' = \delta \ast (\delta' \ast \delta'')\), (VII) \(\delta + \delta' = \delta' + \delta''\), (VIII) \(\delta \ast (\delta' + \delta'') = \delta \ast \delta' + \delta \ast \delta''\), (IX) \(\delta_1 \cdot \delta_j = \delta_{i \in j}, \delta_1 \cdot \delta_j = \delta_{j+1} \cdot \delta_{j+2} \cdots \delta_{j+i}\)

**Example 2.4** By means of the last property we calculate the following star-products:

\[
\begin{align*}
(\cdot) & \quad \delta_2^3 \ast \delta_3^3 = \delta_{(2,2,2)} \ast \delta_{(3,3,3)} = \\
& = \delta_{(3+2,3,2,3,3+2)} \cdot \delta_{(3+2,3,2,3,3+2)} \cdot \delta_{(3+2,3,2,3,3+2)} = \\
& = \delta_{(5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5)} = \delta_5^9 \\
(\cdot \cdot \cdot) & \quad \delta_2^3 \ast \delta_1 \delta_2^2 = \delta_{(1,2,2)} \ast \delta_{(1,2,2)} = \\
& = \delta_{(1,2,2,2,2,2,2,2,2)} \cdot \delta_{(1,2,2,2,2,2,2,2,2)} \cdot \delta_{(1,2,2,2,2,2,2,2,2)} = \\
& = \delta_{(3,4,4)} \cdot \delta_{(3,4,4)} \cdot \delta_{(3,4,4)} = \\
& = \delta_{(3,3,3,3,4,4,4,4,4,4,4,4,4)} = \delta_3^3 \delta_4^6
\end{align*}
\]

It is easily now checked that \((\Delta, *)\) is an abelian semigroup with identity \(\delta_0 = \{l_0\}\), where \(\delta_0 x(t) = x(t)\), \((\Delta, \cdot)\) is an abelian group and \((\Delta, +)\) turns into an \(R\)-module. Obviously \((\Delta, +, *)\) is not a ring.
3 The $\delta$-polynomials

For the description of non-linear polynomial discrete systems the notion of $\delta$-polynomials is necessary. They are just an extension of the $\delta$-operators with some special properties. Despite the fact that these $\delta$-polynomials have been studied exhaustively recently, we shall repeat some of these results here, and shall add some new ones.

Expressions of the form $A = \sum_{i=0}^{\infty} \sum_{\mathbf{i} \in \mathbf{I}_n} a_i \delta_1$ are called $\delta$-polynomials, where by $\mathbf{I}_n$ we denote the set of multindices with degree $n$. Obviously $\mathbf{I}_0 = \{\delta_e\}$. For each polynomial $A$ we define $d(A)$ and $m(A)$ as follows:

$d(A) = \min \{ \min (i_1, i_2, \ldots, i_n), i = (i_1, i_2, \ldots, i_n) \in \mathbb{Z}^n \text{ such that } a_i \neq 0, \text{ for } n = 1, 2, \ldots, k \}$, $m(A) = \max \{ \max (i_1, i_2, \ldots, i_n), i = (i_1, i_2, \ldots, i_n) \in \mathbb{Z}^n \text{ such that } a_i \neq 0, \text{ for } n = 1, 2, \ldots, k \}$. We define as degree of $A$ and we denote it by $\deg(A)$ the maximum $\deg(\delta)$ appeared in $A$. The maximum term of a non-linear polynomial $A$, denoted by $\max(A)$, is its largest term, accordingly to the lexicographical order defined in the previous section. For instance, if $A = \delta_0 \delta_0 + \delta_0 \delta_1 + \delta_0 \delta_0 \delta_0$ then $\delta_0 \delta_0 \delta_0$ is the maximum term. If $A = 2 \delta_0 \delta_0 + 3 \delta_0 \delta_0 \delta_1 + 2 \delta_1 \delta_1 \delta_1 + 5 \delta_2 \delta_2 \delta_1$, then the maximum term is $\delta_2 \delta_2 \delta_1$.

Finally the maximum term of $A = \delta_0 \delta_0 + \delta_3 \delta_3 \delta_2 + \delta_3 \delta_2 \delta_1$ is $\delta_3 \delta_3 \delta_2$. An expression of the form $\sum_{i \in \mathbb{Z}} a_i \delta_i$ is called a linear polynomial.

Two polynomials $A$ and $A'$ are equal if $A y(t) = A' y(t)$ for each $y(t) \in F$. We define the dot-product as follows $A \cdot B = \sum_i \sum_j a_i b_j \delta_1 \cdot \delta_i$. This dot-product is nothing else than the classical product among polynomials with many variables. Also a star product is defined accordingly to the formula: $A \ast B y(t) = A \circ B y(t) = A(B(y(t)))$. In the the general case where $A, B$ are two $\delta$-polynomials their star-product is given by a formula arising from the straight substitution of $B$ into $A$. This formula as well as the proof of the following proposition, can be found in [14].

**Proposition 3.1** The following equalities hold: (I) $[A + B] \ast C = A \ast C + B \ast C$, (II) $C \ast [A + B] = C \ast A + C \ast B$ iff $C$ linear, (III) $A \ast B \neq B \ast A$, (IV) $(A \ast B) \cdot \Gamma = A \cdot (B \cdot \Gamma)$, (V) $d(A \ast B) = d(A) + d(B)$, (VI) $\deg(A \ast B) = \deg(A) \cdot \deg(B)$.

Two $\delta$-polynomials $A, B$, are called commutative if $A \ast B = B \ast A$. The following proposition provides us with a class of commutative polynomials. Its proof is straightforward.
Proposition 3.2 Let \( A, B \) be two \( \delta \)-polynomials. These are commutative if and only if, either both \( A, B \) are linear or both \( A, B \in \Delta \) or one of them is of the form \( \delta_i, i \in \mathbb{Z} \), i.e. a simple shift operator.

The results will follow combine the dot and star-product. All of them are new in the literature.

Proposition 3.3 If \( \delta = \delta_{i_1}\delta_{i_2} \cdots \delta_{i_n} \in \Delta \) and \( A \) a \( \delta \)-polynomial then the following equality holds: 
\[
\delta * A = (\delta_{i_1} * A) \cdot (\delta_{i_2} * A) \cdots (\delta_{i_n} * A).
\]

Proof: Let \( u(t) \in F \) be a sequence and \( z(t) = Au(t) \). Then from the definition of the star-product as a composition among polynomials we get: 
\[
\delta * Au(t) = \delta z(t) = (\delta_{i_1} z(t)) \cdot (\delta_{i_2} z(t)) \cdots (\delta_{i_n} z(t)) = (\delta_{i_1} * Au(t)) \cdot (\delta_{i_2} * Au(t)) \cdots (\delta_{i_n} * Au(t))u(t)
\]
and the equality has been proved.

\[Q.E.D\]

The above proposition provides us with a very useful tool for various cases.

Proposition 3.4 The following equality holds: 
\[
(A \cdot B) * C = (A * C) \cdot (B * C)
\]

Proof: From the definition of equality among polynomials, we are going to prove that \( (A \cdot B) * Cu(t) = (A * C) \cdot (B * C)u(t) \) for an arbitrary sequence \( u(t) \). Let \( z(t) = Cu(t) \), then 
\[
(A \cdot B) * z(t) = (A \cdot B)z(t) = (Az(t)) \cdot (Bz(t))
\]
Using the definition of star-product we get: 
\[
Az(t) = A * Cu(t)
\]
and therefore the previous equation becomes: 
\[
(A * Cu(t)) \cdot (B * Cu(t)) = (A * C) \cdot (B * C)u(t)
\]
\[Q.E.D\]

The left-hand equality: \( C * (A \cdot B) = (C * A) \cdot (C * B) \) does not in general hold, as we can see in the following example:

Example 3.1 \( \delta_1 + \delta_1^2 \) \[
(\delta_1 + \delta_2) \cdot \delta_3 = (\delta_1 + \delta_2^2) \cdot (\delta_1 \delta_3 + \delta_2 \delta_3) = \delta_2 \delta_4 + \delta_2^2 \delta_4 + \delta_2 \delta_4^2 + 2 \delta_2 \delta_3 \delta_4,
\]
but \( [\delta_1 + \delta_1^2] \cdot (\delta_1 + \delta_2) = \delta_1 + \delta_3 + \delta_2 \delta_3 \cdot (\delta_4 + \delta_4^2) = \delta_2 \delta_4 + \delta_3 \delta_4 + \delta_2 \delta_4^2 + \delta_2^2 \delta_4 + 2 \delta_2 \delta_3 \delta_4 + \delta_2 \delta_4^2 + \delta_2 \delta_4^2 + \delta_3 \delta_4 + \delta_2 \delta_3 \delta_4^2\]
The only exception to this generality is the following:
Proposition 3.5 If \( \delta \in \Delta \) then the following equality holds: \( \delta \ast (A \cdot B) = (\delta \ast A) \cdot (\delta \ast B) \).

Proof: Let us suppose that \( \delta = \delta_1 \delta_2 \cdots \delta_i \) and \( A, B \) \( \delta \)-polynomials. We have:

\[
\delta \ast (A \cdot B) = \delta_1 \ast (A \cdot B) \cdot \delta_2 \ast (A \cdot B) \cdots \delta_i \ast (A \cdot B) \tag{2}
\]

Using propositions (3.2, 3.4) we get \( \delta_1 \ast (A \cdot B) = (A \cdot B) \ast \delta_1 = (A \ast \delta_1) \cdot (B \ast \delta_{i-1}) = (\delta_i \ast A) \cdot (\delta_{i-1} \ast B) \). Thus (2) becomes: \( (\delta_1 \ast A) \cdot (\delta_2 \ast A) \cdots (\delta_i \ast A) \cdot (\delta_1 \ast B) \cdot (\delta_2 \ast B) \cdots (\delta_i \ast B) = (\delta \ast A) \cdot (\delta \ast B) \).

Q.E.D

4 Factorizations

The main task of our efforts is to factorize a \( \delta \)-polynomial according to the star-product. If we can write a \( \delta \)-polynomial \( A \) in the form: \( A = A_1 \ast A_2 \ast \cdots \ast A_k \) then we can say that the polynomial is star-factorized and the terms \( A_1, A_2, \ldots, A_k \) are called star-factors. If the terms \( A_1, A_2, \ldots, A_k \) cannot be star-factorized any more, then we say that we have a maximum star-factorization. Bearing in mind that the star-product corresponds to the composition of maps we can understand the importance of the star-factorization since it will permit us to analyse a non-linear system to its primary factors. The problem of finding an algorithm, which will lead to the maximum factorization of a given polynomial, remains open and it is under investigation. Alternatively some other algorithms can deal with a part of the problem and solve a special aspect of it. A first attempt was made in [14] where an algorithm for the linear factorization of \( \delta \)-polynomials appeared. Let \( A \) be a \( \delta \)-polynomial, with \( d(A) \geq 0 \). We say \( A \) is linearly factorizable (LF) if there is a non-trivial linear \( \delta \)-polynomial \( L, m(L) > 0 \) and a \( \delta \)-polynomial \( D \) such that: \( A = L \ast D \). Not every polynomial is linearly factorizable. The polynomial \( A = \delta_0 \), for instance, is not LF. All the details of linear factorization can be found in [14], and since we shall use them in this paper as well, the essential algorithms are included briefly in the Appendix. In this chapter we shall describe a new factorization, the so-called \( \delta \)-factorization. This is a method based on the linear factorization,
which includes a larger variety of $\delta$-polynomials and permits an advanced form of star-factorization to take place. Therefore the $\delta$-factorization method establishes the next step to the solution of the maximum star-factorization problem.

A non-linear polynomial $A$ is called $\delta$-factorizable if it can be written in the form: $A = \delta \ast D$, where $\delta \in \Delta$, $\deg(\delta) > 1$ and $D$ a non-linear $\delta$-polynomial. The following propositions provide us with some non-\delta-factorizable classes.

**Proposition 4.1** If $\deg(A)$ is a prime number larger than 2, then $A$ cannot be written in the form: $A = \delta \ast A$, with $\deg(A) > 1$.

**Proof:** Since $\deg(A) = \deg(\delta) \cdot \deg(\hat{A})$ and $\deg(\delta) > 1$, the result is straightforward.

$Q.E.D$

**Proposition 4.2** If $A$ contains a linear term, then it cannot be written in the form: $A = \delta \ast D$, where $\delta \in \Delta$, $\deg(\delta) > 1$ and $D$ a non-linear $\delta$-polynomial.

**Proof:** The degree of a linear term is equal to 1, but the product will produce terms with degree more than 1, since $\deg(\delta) > 1$, and therefore the proposition is valid.

$Q.E.D$

The $\delta$-factorization method is based on the two algorithms, named the "$l\delta F$" and the "$\delta F$" algorithm. Both of them are presented in the Appendix. Here we give the main result concerning the output of these algorithms.

**Theorem 4.1** If $\delta, M_1, M_2, \ldots, M_n, A_{\lambda_1}, A_{\lambda_2}, \ldots A_{\lambda_n}$ are the outputs of the algorithm $\delta F$, then $A = \delta \ast [(M_1 \ast A_{\lambda_1}) \cdot (M_2 \ast A_{\lambda_2}) \cdots (M_n \ast A_{\lambda_n})]$, is the $\delta$-factorization upon request.

**Proof:** Supposing that the "$\delta F$" algorithm terminates giving an output. Following its steps we get: $A = (L_1 \ast A_1) \cdot (L_2 \ast A_2) \cdots (L_n \ast A_n)$. Since $A_{\mu_1} = A_{\mu_1} = A_{\mu_1} = \cdots = A_{\mu_1} = A_{\lambda_1}$, $A_{\mu_2} = A_{\mu_2} = A_{\mu_2} = \cdots = A_{\mu_2} = A_{\lambda_2}$, ......., $A_{\mu_n} = A_{\mu_n} = A_{\mu_n} = \cdots = A_{\mu_n} = A_{\lambda_n}$, we get $A = (L_{\mu_1} \ast A_{\lambda_1}) \cdot (L_{\mu_2} \ast A_{\lambda_2}) \cdots (L_{\mu_n} \ast A_{\lambda_n})$ and using proposition
(3.4): \( A = ((L_{\mu_1} \cdot L_{\mu_1'} \cdots L_{\mu_1''}) \ast A_{\lambda_1}) \cdot (L_{\mu_2} \cdot L_{\mu_2'} \cdots L_{\mu_2''}) \ast A_{\lambda_2}) \cdot \cdots \cdot (L_{\mu_n} \cdot L_{\mu_n'} \cdots L_{\mu_n''}) \ast A_{\lambda_n}). \) Following now the steps of the algorithm "\( l\delta F \)" we have successively: 
\[ L_{\mu_1} \cdot L_{\mu_2} \cdots L_{\mu_n} = (\delta_{i_1} \cdot \hat{L}_{\mu_1}) \cdot (\delta_{i_2} \cdot \hat{L}_{\mu_2}) \cdots (\delta_{i_n} \cdot \hat{L}_{\mu_n}), \]
since 
\[ \hat{L}_{\mu_1} = \hat{L}_{\mu_2} = \cdots = \hat{L}_{\mu_n} = M_e \] using proposition (3.3) we get 
\[ L_{\mu_1} \cdot L_{\mu_2} \cdots L_{\mu_n} = (\delta_{i_1} \cdot M_e) \cdot (\delta_{i_2} \cdot M_e) \cdots (\delta_{i_n} \cdot M_e) = (\delta_{i_1} \delta_{i_2} \cdots \delta_{i_n}) \ast M_e = \delta^{(v)} \ast M_e. \] But since \( \delta^{(1)} = \delta^{(2)} = \cdots = \delta^{(v)} = \delta \) we have: 
\[ A = (\delta \ast M_1 \ast A_{\lambda_1}) \cdot (\delta \ast M_2 \ast A_{\lambda_2}) \cdots (\delta \ast M_e \ast A_{\lambda_e}) \] and using proposition (3.5) we get the desired result.

Q.E.D

**Theorem 4.2** Let \( A \) be a \( \delta \)-polynomial and \( A = \delta \ast D \) its \( \delta \)-factorization accordingly to the algorithm "\( \delta F \)". If \( A = \delta' \ast D' \), \( \delta' \in \Delta \), is any other \( \delta \)-factorization of this form, then \( \delta' \preceq \delta \) and therefore the \( \delta \)-factorization \( A = \delta \ast D \) is unique.

**Proof:** The dot-factorization of the step 1 of the algorithm \( \delta F \) is maximum. This means that \( A \) cannot be analysed in more dot-terms than those appeared in this step. Therefore if \( A = \delta \ast D \) is a \( \delta \)-factorization coming out from the algorithm "\( \delta F \)" and \( A = \delta' \ast D' \) any other one then obviously \( \deg(\delta) \geq \deg(\delta') \). Let us now suppose that \( \deg(\delta) = \deg(\delta') \). The operator \( \delta = \delta_{i_1} \cdots \delta_{i_n} \) has produced by means of algorithm \( \delta F \) and therefore the step 1 of algorithm "\( l\delta F \)" has been used. All the operators \( \delta_{i_n} \) appeared there have maximum delay. This means that for the operator \( \delta' \) will have \( \sigma(\delta') < \sigma(\delta) \) and finally \( \delta' \preceq \delta \). In other words the algorithm \( \delta F \) creates the maximum \( \delta \)-factorization. Therefore the \( \delta \)-operator produced by it, is maximum, in the sense that \( \delta' \preceq \delta \).

Concerning the uniqueness now. If \( A = \delta' \ast D' \) is arising as an output of the algorithm \( \delta F \), then \( \delta \succeq \delta' \) and \( \delta' \succeq \delta \) and thus we conclude that \( \delta = \delta' \). This means that \( \delta \ast Du(t) = \delta' \ast D'u(t) \) for any \( u(t) \in F \). Setting \( z(t) = Du(t) \) and \( \dot{z}(t) = D'u(t) \) we get \( \dot{z}(t) = \delta \dot{z}(t) \). It is trivial, using induction and the form of \( \delta \) to prove that \( z(t) = \dot{z}(t) \) and therefore \( Du(t) = D'u(t) \). Since \( u(t) \) arbitrary we conclude that \( D = D' \) and the theorem has been proved.

Q.E.D

**Example 4.1** (i) Let us consider the polynomial \( A = \delta_0 \delta_1 + \delta_0 \delta_1^2 + 2\delta_0 \delta_1 \delta_2 + \delta_0 \delta_1^2 \delta_2 + \delta_0 \delta_1^2 \delta_3 + \delta_0 \delta_1 \delta_2^2 + \delta_0 \delta_1 \delta_2 \delta_3 \). Factorizing accordingly to the dot-product
we get $A = \delta_0 \delta_1 (1 + \delta_1 \delta_2)(1 + \delta_2 \delta_3)$. From the step 2 of $\delta F$ we have: $A_1 = \delta_0 = \delta_0 \circ \delta_0$, $A_2 = \delta_1 = \delta_1 \circ \delta_0$, $A_3 = 1 + \delta_1 + \delta_2 = \delta_1 \circ (1 + \delta_0 + \delta_1)$, $A_4 = 1 + \delta_2 + \delta_3 = \delta_2 \circ (1 + \delta_0 + \delta_1)$. So $L_1 = \delta_0$, $A_1 = \delta_0$, $L_2 = \delta_1$, $A_2 = \delta_0$, $L_3 = \delta_1$, $A_3 = 1 + \delta_0 + \delta_1$, $L_4 = \delta_2$, $A_4 = 1 + \delta_0 + \delta_1$. We observe that $A_{\lambda_1} = A_1 = A_2 = \delta_0$ and $A_{\lambda_2} = A_3 = A_4 = 1 + \delta_0 + \delta_1$.

Using the algorithm "l$\delta F$" we analyse the polynomials $L_1 \cdot L_2$ and $L_3 \cdot L_4$ as follows: $L_1 \cdot L_2 = \delta_0 \delta_1 = \delta_0 \delta_1 \circ \delta_0$ and $L_3 \cdot L_4 = \delta_1 \delta_2 = \delta_0 \delta_1 \circ \delta_1$.

Thus $\delta = \delta^{(1)} = \delta^{(2)} = \delta_0 \delta_1$ and $M_1 = \delta_0$, $M_2 = \delta_1$. Finally the requested factorization is: $\delta \cdot (M_1 \circ A_{\lambda_1}) \cdot (M_2 \circ A_{\lambda_2}) = \delta_0 \delta_1 \cdot [(\delta_0 \circ \delta_0) \circ (\delta_1 \circ (1 + \delta_0 + \delta_1))] = \delta_0 \delta_1 \cdot [\delta_0 \circ (1 + \delta_0 + \delta_1)] = \delta_0 \delta_1 \circ (\delta_0 + \delta_0 \delta_1 + \delta_0 \delta_2)$.

(iii) Let us consider the polynomial $A = \delta_1 \delta_2^2 + 2 \delta_1 \delta_2 \delta_3 + \delta_3^2 + 2 \delta_2 \delta_3 + \delta_1 \delta_3^2 + \delta_2 \delta_3^2$. Gradually we get: $A = \delta_2 (\delta_2^2 + 2 \delta_2 \delta_3 + \delta_3^2) + \delta_1 (\delta_2^2 + 2 \delta_2 \delta_3 + \delta_3^2) = (\delta_1 + \delta_2) (\delta_2 + \delta_3) (\delta_2 + \delta_3)$. But, $\delta_1 + \delta_2 = \delta_1 (\delta_0 + \delta_1)$, $\delta_2 + \delta_3 = \delta_2 (\delta_0 + \delta_1)$, $\delta_2 + \delta_3 = \delta_2 (\delta_0 + \delta_1)$, from which we finally get $\delta = \delta_1 \delta_2^2$ and $L = \delta_0 + \delta_1$. Hence, $A = \delta_1 \delta_2^2 \circ (\delta_0 + \delta_1)$ is the required factorization.

(iv) The polynomial $A = \delta_0 \delta_1 + \delta_2 \delta_3$ cannot be factorized accordingly to the dot-product and therefore it cannot be written in the form: $A = \delta \circ D$.

The polynomial $A = \delta_0^2 + \delta_0 \delta_1 + \delta_0 \delta_2 + \delta_1 \delta_2$ despite can be written as $A = (\delta_0 + \delta_1) \cdot (\delta_0 + \delta_2)$ does not "satisfy" step 1 of the subroutine "l$\delta F$", since there is not a common linear factor and therefore it is not finally factorizable.

5 Applications to systems analysis

We are now ready to apply all the above procedures to some systems problems. First we need two definitions.

Definition 5.1 Let $A$ be a $\delta$-polynomial and $\delta_a$, $a = d(A)$, be the term of $A$ with the minimum delay. The polynomial $A$ is called proper if the power of $\delta_a$, in $A$, is equal to one.

For instance $\delta_1 \delta_2 + \delta_2 \delta_3$ is a proper polynomial, besides $\delta_0^2 \delta_1 + \delta_2 \delta_3$ is not. Obviously all the polynomials where their linear part contains the lowest delay term are proper.

Definition 5.2 Let $A, B$ be two polynomials and $A = A_1 \circ A_2 \circ \cdots \circ A_n$, $B = B_1 \circ B_2 \circ \cdots \circ B_m$ two star-factorizations of them. The polynomials $(A_i, B_j)$ are called simplified-among-each-other iff (i) $A_i = B_j$ and (ii) we
can transfer them at the "beginning" of \( A \) and \( B \), i.e. \( A_i \) is commutative with all the polynomials \( A_1, A_2, \ldots, A_{i-1} \) and \( B_j \) is commutative with all the polynomials \( B_1, B_2, \ldots, B_{j-1} \).

If \( A = \delta_0 \delta_1 \delta_3^2 \) and \( B = \delta_0 \delta_1 \delta_3^2 \) then the terms \( \delta_3^2 \) are simplified, whilst the terms \( \delta_0 \delta_1 \) are not, since we cannot commute them with the terms \( \delta_3^2 \).

### 5.1 Linearization

The above mentioned factorizations can be used effectively as a "simplification" method for a given non-linear system. In other words they can permit us to determine systems with a "less" complex structure, which are equivalent to the original ones, i.e. they will give the same output under the same input.

Among these "simpler" systems the linear are the most desirable ones, since their behaviour has been studied exhaustively. Therefore we may ask when a non-linear system of the form (1) is equivalent to a linear one. This is a version of the well known linearization problem. The following results will help us to face this linearization problem gradually.

**Theorem 5.1** The equality \( Ay(t) = Au(t), \forall t \in Z \) implies \( y(t) = u(t) \), provided that \( A \) is proper and that \( u(t), y(t) \) have the same initial conditions.

**Proof:** Since \( A \) is proper we can rewrite the above equality as follows:

\[
y(t)f_1(y(t-1), y(t-2), \ldots, y(t-n)) + f_2 = u(t)f_1(u(t-1), u(t-2), \ldots, u(t-n)) + f_2,
\]

where \( f_1, f_2 \) polynomial functions of the delays of \( y(t) \) and \( u(t) \). Some of \( f_1, f_2 \) can be equal to zero. Let \( y(0), y(1), \ldots, y(n-1), u(0), u(1), \ldots, u(n-1) \) be the initial conditions. Since they are equal we conclude from the above equation that \( y(n) = u(n) \). Working inductively we can prove that \( y(t) = u(t) \) for each \( t \in Z \).

\[Q.E.D\]

The following is the main simplification theorem.

**Theorem 5.2** Let \( A'y(t) = B'u(t) \) be a non-linear system with \( u(t) \) as an input and \( y(t) \) as an output. Let us suppose that \( A' = C' \) and \( B' = C' \)

13
where $C$ a proper $\delta$-polynomial. Then the system $Ay(t) = Bu(t)$ gives the same output with the original one, under the same input and the same initial conditions.

**Proof:** Let $A' y(t) = B' u(t)$ be the original system with the sequence $u(t)$ as an input. Then we have $C * Ay(t) = C * Bu(t)$. Setting $y_1(t) = Ay(t)$, $u_1(t) = Bu(t)$, we get $Cy_1(t) = Cu_1(t)$. But since $C$ is proper, accordingly to theorem (5.1) we take $y_1(t) = u_1(t)$ and thus $Ay(t) = Bu(t)$.

Q.E.D

We have to make clear here that the above equivalence is a rather formal one, since some stability conditions have to be taken into consideration. In this case some BIBO criteria (as those in [13]) are needed in order to guarantee the internal stability of the simulations.

An immediate consequence of the theorem 5.2 is the following corollary about linearization.

**Corollary 5.1** Let $Ay(t) = Bu(t)$ be a non-linear system with $u(t)$ as an input and $y(t)$ as an output. Let us suppose that $A = C * L$, $B = C * M$ where $C$ a proper $\delta$-polynomial and $L, M$ linear ones. Then the original system is linearizable and $Ly(t) = Mu(t)$ is its linearization.

**Example 5.1** Let us consider the non-linear system: $y(t) + 2y(t-1) + y(t-2) + y(t-1)y(t-2) + y(t-1)y(t-3) + y^2(t-2) + y(t-2)y(t-3) = u(t-1) + u(t-2) + u(t-2)u(t-3) or using $\delta$-polynomials: $(\delta_0 + 2\delta_1 + \delta_2 + \delta_1 \delta_2 + \delta_1 \delta_3 + \delta_2^2 + \delta_2 \delta_3)y(t) = (\delta_0 + \delta_2 + \delta_1 \delta_2 + \delta_1 \delta_3 + \delta_2^2 + \delta_2 \delta_3)u(t)$. Using factorization algorithms, we get: $(\delta_0 + 2\delta_1 + \delta_2) = (\delta_0 + \delta_1) * (\delta_0 + \delta_1)$, $(\delta_0 \delta_2 + \delta_1 \delta_3 + \delta_2^2 + \delta_2 \delta_3) = \delta_1 \delta_2 * (\delta_0 + \delta_1)$, $(\delta_0 + 2\delta_1 + \delta_2 + \delta_1 \delta_2 + \delta_1 \delta_3 + \delta_2^2 + \delta_2 \delta_3) = (\delta_0 + \delta_1 + \delta_1 \delta_2) * (\delta_0 + \delta_1)$ and $(\delta_0 + \delta_2 + \delta_2 \delta_3) = (\delta_0 + \delta_1 + \delta_1 \delta_2) * (\delta_1)$. From theorem (5.2) we conclude that the original system is equivalent with the linear system: $y(t) + y(t-1) = u(t-1)$. Relative is the numerical simulation at figure 1, where we present the output of the two systems. We used as input the random function $\text{rnd}(1)$.

**Example 5.2** Let us examine now the system: $y(t-1) + y^2(t-2) + 2y(t-1)y(t-3) + y(t-2)y(t-3) = u(t-2) + u^2(t-3) or using $\delta$-operators: $(\delta_1 \delta_2 + 2\delta_1 \delta_2 \delta_3 + \delta_2^2 + 2\delta_2 \delta_3 + \delta_1 \delta_3^2 + \delta_2 \delta_3)u(t) = \delta_2 \delta_3^2 u(t)$. Working with the factorizations algorithms we get that:
\begin{tabular}{|c|c|c|}
\hline
Repeats & Non-linear output & Linear output \\
\hline
3 & 0.7516421 & 0.7516421 \\
53 & -1.009715 & -1.009714 \\
103 & -1.74956 & -1.749559 \\
203 & 1.047003 & 1.04699 \\
503 & 1.751679 & 1.75165 \\
703 & 6.320683 & 6.320615 \\
903 & 4.216112 & 4.216075 \\
\hline
\end{tabular}

Figure 1: A simulation with linear terms.

\[\delta_1\delta_2^2 + 2\delta_1\delta_2\delta_3 + \delta_2^3 + 2\delta_2\delta_3 + \delta_1\delta_3^2 + \delta_2\delta_3^2 = \delta_1\delta_2^2 \ast (\delta_0 + \delta_1) \text{ and } \delta_2\delta_3^2 = \delta_1\delta_2^2 \ast \delta_1.\]

Properness of \(\delta_1\delta_2^2\) implies equivalence among the original non-linear system and the linear system \(y(t) + y(t - 1) = u(t - 1)\), in the sense that they give the same output under the same input.

### 5.2 A BIBO Stability Theorem

A BIBO behaviour for a system is the most important one and any result in this approach is always welcome, especially in the case of non-linear discrete systems where the relative literature is still poor.

The factorization method, developed before, endows us with a functional BIBO tool. Specifically, if the polynomial \(A\) of the original system \(y(t) = Ay(t) + Bu(t)\) can be analysed in the form \(A = \delta \ast L\), where \(L\) is linear, then a BIBO result is straightforwardly.

**Theorem 5.3** We have the non-linear system \(y(t) = Ay(t) + Bu(t)\) where \(A, B\) are the non-linear polynomials: \(A = \sum_{i=1} a_i\delta_i\), \(B = \sum_{j=3} b_j\delta_j\), \(d(A) > 0\), \(d(B) > 0\). If the input \(u(t)\) is bounded by the number \(M\), i.e. \(|u(t)| \leq M\), \(M \in \mathbb{R}\) and the following conditions hold:

(i) \(A\) can be written in the form: \(A = \delta \ast L\), where \(\delta = \delta_{i_1}\delta_{i_2}\ldots\delta_{i_m} \in \Delta\) and \(L = \sum_{i=1}^{m} \lambda_i\delta_i\), \(\lambda_i \in \mathbb{R}\) a linear polynomial.

(ii) The initial values of \(y(t)\) lie into a certain interval, i.e. \(|y(t)| \leq K\), \(K \in \mathbb{R}\), \(t = 0, 1, 2, \ldots, m + i_n\).

(iii) \(K^n + M < K\), where \(K = \sum_{i=1}^{m} |\lambda_i|\), \(M = \sum_{j=3} b_jM\).
then the system is BIBO stable in the sense that all the output values are bounded by the value $K$, i.e. $|y(t)| \leq K$, $\forall t \in N$.

**Proof:** Firstly we set $z(t) = \sum_{i=1}^{m} \lambda_i y(t - i)$. By using assumption (i) the system is written in the form:

$$y(t) = \delta_1 \delta_2 \cdots \delta_n \left( \sum_{i=1}^{m} \lambda_i y(t-i) \right) + \sum_{s=1}^{w} \sum_{(j_1,j_2,\ldots,j_s) \in I^s} b_{j_1,j_2,\ldots,j_s} u(t-j_1)u(t-j_2)\cdots u(t-j_s)$$

or

$$y(t) = z(t-i_1)z(t-i_2)\cdots z(t-i_n) + \sum_{s=1}^{w} \sum_{(j_1,j_2,\ldots,j_s) \in I^s} b_{j_1,j_2,\ldots,j_s} u(t-j_1)u(t-j_2)\cdots u(t-j_s)$$

Let us suppose now that we have an input signal bounded by the number $M$, i.e. $|u(t)| \leq M$. For the time instant $\phi = m + i_n + 1$ we get:

$$y(\phi) = z(\phi-i_1)z(\phi-i_2)\cdots z(\phi-i_n) + \sum_{s=1}^{w} \sum_{(j_1,j_2,\ldots,j_s) \in I^s} b_{j_1,j_2,\ldots,j_s} u(\phi-j_1)u(\phi-j_2)\cdots u(\phi-j_s)$$

and therefore

$$|y(\phi)| \leq |z(\phi-i_1)||z(\phi-i_2)|\cdots |z(\phi-i_n)| + M$$  \hspace{1cm} (3)

But $|z(\phi-i_n)| \leq \sum_{i=1}^{m} |\lambda_i||y(t-i-i_n)|$ and by assumption (ii) we get $|z(\phi-i_n)| \leq \sum_{i=1}^{m} |\lambda_i|K = K$. This is valid for all the values of $n$ and thus (3) becomes: $|y(\phi)| \leq KK \cdots K + M = K^n + M$. By means of assumption (iii) we conclude that $y(\phi) < K$. Finally, using induction we can prove that this is valid for every $t \in N$ and therefore the output is bounded and the system is BIBO stable.

Q.E.D

What is hidden in the previous theorem is that if the non-linear polynomial $A$ has a certain structure, i.e. it "contains" a linear polynomial, then some special properties can be obtained. The $\delta$-factorization can reveal structures of this form therefore it is a useful tool for this approach.
Example 5.3 The system \( y(t) = 0.01y(t-2)y(t-3) + 0.02y(t-2)y(t-4) + 0.02y^2(t-3) + 0.04y(t-3)y(t-4) + 0.1u(t-1) + 0.3u(t-1)u(t-4) \) can be analysed in the form: \( y(t) = \delta_1 \delta_2 \ast \left[ 0.1y(t-1) + 0.2y(t-2) \right] + 0.1u(t-1) + 0.3u(t-1)u(t-4) \). It is now easy to verify that the assumptions of the previous theorem hold and therefore the system is BIBO stable with input and output bounds \( M \) and \( K \) equal to 1. Simulation results can show the truth of the above statement.

5.3 The Cascade Connection

An important aspect of the system analysis is the so-called cascade connection where a kind of composition among systems is described. In this subsection we introduce a procedure in order to derive a compact formula for the cascade connection of non-linear systems.

Let us suppose that we have the non-linear system \( Cu = Dv \), with input \( v \) and output \( u \) and the non-linear system \( Ay = Bu \) with input \( u \) and output \( y \), \( A, B, C, D \) non-linear \( \delta \)-polynomials. This configuration is called a linear or cascade connection (See figure 2). The following describes mathematically the whole approach.

Proposition 5.1 If \( B, C \) are commutative then \( C \ast Ay = B \ast Dv \) represents the cascade connection of the two systems.
Proof: Post multiplying with $C$, using the star-product, we have successively: $C \ast Ay = C \ast Bu = B \ast Cu = B \ast Dv$, where we used the fact that $B, C$ are commutative.

Using the above proposition and the factorization analysis we can get simpler forms of the cascade connection. Indeed if $C_1, C_2, \ldots, C_n, A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_k, D_1, D_2, \ldots, D_\lambda$ are factorizations of $C, A, B, D$, respectively, then we have: $C_1 \ast C_2 \ast \cdots \ast C_n \ast A_1 \ast A_2 \ast \cdots \ast A_m = B_1 \ast B_2 \ast \cdots \ast B_k \ast D_1 \ast D_2 \ast \cdots \ast D_\lambda$. If some of the factors of the left and right part of the above equation, are simplified-among-each-other, as this simplification was defined in (5.2), then we can get another simpler form for the cascade connection. The following example is illustrative:

Example 5.4 We have the systems $u(t) + u(t - 1) = v(t - 1)v(t - 3) + v(t - 1)v(t - 4) + v(t - 2)v(t - 3) + v(t - 2)v(t - 4)$ and $y(t)[y(t - 2) + y(t - 3)] + y(t - 1)y(t - 2) + y(t - 1)y(t - 3) = u(t - 1) + u(t - 2)$. In the cascade connection the output of the first system is used as an input in the latter. Using the factorizations algorithms we get for the polynomials: $A = \delta_0 \delta_2 + \delta_0 \delta_3 + \delta_0 \delta_2 + \delta_1 \delta_3 = \delta_0 \delta_2 * (\delta_0 + \delta_1)$, $B = \delta_1 + \delta_2 = (\delta_0 + \delta_1) * \delta_1$, $C = \delta_0 + \delta_1$, $D = \delta_1 \delta_3 + \delta_1 \delta_4 + \delta_2 \delta_3 + \delta_2 \delta_4 = \delta_0 \delta_2 * (\delta_1 + \delta_2)$. Their cascade connection is given by $C \ast Ay(t) = B \ast Dv(t)$ or $(\delta_0 + \delta_1) \ast \delta_0 \delta_2 * (\delta_0 + \delta_1) y(t) = (\delta_0 + \delta_1) \ast \delta_1 \ast \delta_0 \delta_2 * \delta_1 \ast \delta_0 v(t)$. Since the terms $\delta_0 + \delta_1$, $\delta_0 \delta_2$ can be simplified, we finally get $\delta_0 + \delta_1 = \delta_2 + \delta_3$ and the whole system is equivalent to the linear system: $y(t) + y(t - 1) = v(t - 2) + v(t - 3)$.

In the following simulation (figure 3), we used the function $\text{rand}(1)$ as input. We must note here that different inputs can lead to overflows. In order to avoid that we must ensure in advance the internal stability of the several systems.

5.4 The Feedback Design.

One essential problem in system theory is that of feedback designing. By this we mean the search for proper input so that the closed-loop system will behave like a desired one. In this subsection we study a particular form of this problem for special non-linear discrete systems. Variations of the same problem can be found in [14], [15].
<table>
<thead>
<tr>
<th>Repeats</th>
<th>Non-linear output</th>
<th>Linear output</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-0.4035994</td>
<td>-0.4035994</td>
</tr>
<tr>
<td>1006</td>
<td>-0.2024171</td>
<td>-0.2024042</td>
</tr>
<tr>
<td>1756</td>
<td>-0.04916799</td>
<td>-0.04916316</td>
</tr>
<tr>
<td>2506</td>
<td>-0.1998779</td>
<td>-0.1998839</td>
</tr>
<tr>
<td>3256</td>
<td>-0.4814319</td>
<td>-0.4814172</td>
</tr>
<tr>
<td>4006</td>
<td>0.02503521</td>
<td>0.02505237</td>
</tr>
<tr>
<td>4756</td>
<td>-0.5281621</td>
<td>-0.5282</td>
</tr>
</tbody>
</table>

Figure 3: Simulation of the Cascade connection.

![Feedback Design Diagram](image)

Figure 4: The Feedback Design.

Let us suppose that we have the system \( y(t) = Au(t) \), \( A \) a non-linear \( \delta \)-polynomial. We want to find a feedback connection of the form \( F_1u(t) = F_2y(t) \), \( F_1, F_2 \), \( \delta \)-polynomials, \( d(F_1) < d(F_2) \), so that the closed-loop system will behave like the desired system \( y(t) = A_d y(t) \), (see figure (4)). The following theorem provides us with a method to solve this problem.

**Theorem 5.4** Let \( A, A_d \) be two \( \delta \)-polynomials with \( d(A) < d(A_d) \). Let \( A = A_1 \ast A_2 \ast \cdots \ast A_n \), \( A_d = A_{i_1} \ast A_{i_2} \ast \cdots \ast A_{i_k} \) be two star-factorizations of \( A \) and \( A_d \) respectively. If \( (A_{i_1}, \overline{A}_{j_1}), (A_{i_2}, \overline{A}_{j_2}), \ldots, (A_{i_k}, \overline{A}_{j_k}) \) are the pairs of simplified-among-each-other factors, we set: \( F_1 = A_{i_1} \ast A_{i_2} \ast \cdots \ast A_{i_k}, \lambda_x \neq i_1, i_2, \ldots, i_k, x = 1, \ldots, n, F_2 = \overline{A}_{j_1} \ast \overline{A}_{j_2} \ast \cdots \ast \overline{A}_{j_k}, \mu_x \neq j_1, j_2, \ldots, j_k, \)
\[ x = 1, \ldots, m, \text{ i.e. } F_1, F_2 \text{ consist of the no-simplified terms. Then the feedback law: } F_1 u(t) = F_2 y(t), \text{ applied to the system } y = Au, \text{ will give a closed system which will produce the same output with the desired system } y = A_d y. \]

**Proof:** First we rewrite \( y = Au \) in the form: \( y = A_1 \ast A_2 \ast \cdots \ast A_n u \), post-multiplying with \( A_{\lambda_1} \ast A_{\lambda_2} \ast \cdots \ast A_{\lambda_n} \). Since the factors \( A_1, \ldots, A_n \) are simplified, we can transfer them at the beginning of the product and therefore to have: \( y = A_{i_1} \ast A_{i_2} \ast \cdots \ast A_{i_n} \ast A_{\lambda_1} \ast A_{\lambda_2} \ast \cdots \ast A_{\lambda_n} u(t) = A' F_1 y(t) \). Using the fact that \( F_1 u(t) = F_2 y(t) \) we get \( y = A' F_2 y(t) \). But \( A' = A_{j_1} \ast A_{j_2} \ast \cdots \ast A_{j_k} \) and so \( y(t) = A' \ast F_2 y(t) = A_d y(t) \). Now, the inequality \( d(A') < d(A) \) implies \( d(A') + d(F_1) < d(A') + d(F_2) \) and since \( d(A') = d(A) \) we conclude that \( d(F_1) < d(F_2) \), which ensures the causality of the system.

**Q.E.D**

**Example 5.5** We have the system: \( y(t) = u(t-1)u(t-2) + u(t-1)u(t-3)u(t-4) + u(t-2)u(t-3) + u(t-2)u(t-4)u(t-5) + u(t-3)u(t-4)u(t-5), \) we seek for a feedback law so that the closed-loop system will behave like the system: \( y(t) = y(t-2)y(t-3) + y(t-2)y(t-4) + y(t-3) + y(t-2)y(t-3)y(t-4) + y(t-3)y(t-4) + y(t-4)y(t-5) \). Using the \( \delta \)-operators and the factorization techniques, we get: \( A = \delta_1 \delta_2 + \delta_1 \delta_3 \delta_4 + \delta_2 \delta_3 + \delta_2 \delta_4 \delta_5 + \delta_3 \delta_4 \delta_5 = (\delta_1 + \delta_2) \ast \delta_0 \delta_1 \ast (\delta_0 + \delta_1 \delta_2) \) and \( A_d = (\delta_1 + \delta_2) \ast \delta_0 \delta_1 \ast (\delta_0 + \delta_1) \). The terms \( \delta_1 + \delta_2, \delta_0 \delta_1 \) are simplified and so the feedback law is \( (\delta_0 + \delta_1 \delta_2) u(t) = (\delta_1 + \delta_2) y(t) \) or \( u(t) + u(t-1)u(t-2) = y(t-1) + y(t-2) \). Numerical simulations can easily verify the whole procedure. Indicative is table (5) where we took as initial conditions: \( y(0) = 0.1, \ y(1) = 0.018, \ y(2) = -0.1, \ y(3) = 0.145, \ y(4) = -0.5664878. \)

If \( A_d \) is a linear system and \( A_d = A_{d_1} \ast A_{d_2} \ast \cdots \ast A_{d_k} \) its star-factorization, then all of \( A_{d_i} \) must be linear. It can be easily now checked that in order the above method to be applied here, two things must valid: firstly \( A \) must be linear factorizable i.e. \( A = L \ast A \) and secondly \( L A_d \). For instance, if \( A = \delta_0 + \delta_1 + \delta_2 + \delta_3 = (\delta_0 + \delta_1) \ast (\delta_0 + \delta_2) \) and \( A_d = \delta_1 + 2\delta_2 + \delta_3 = (\delta_0 + \delta_1) \ast (\delta_2 + \delta_3) \) then \( (\delta_0 + \delta_3) u(t) = (\delta_1 + \delta_2) y(t) \) or \( u(t) + u(t-1)u(t-2) = y(t-1) + y(t-2) \) is the feedback law upon request. We have to make clear once more that our method is formal in nature and we are not interested in internal stability problems. In the case of linear systems, problems can be caused by the
<table>
<thead>
<tr>
<th>Repeats</th>
<th>Non-linear output</th>
<th>Desired output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.145</td>
<td>0.145</td>
</tr>
<tr>
<td>13</td>
<td>0.05257983</td>
<td>0.05257984</td>
</tr>
<tr>
<td>23</td>
<td>0.127039</td>
<td>0.1270394</td>
</tr>
<tr>
<td>33</td>
<td>0.1277664</td>
<td>0.1277668</td>
</tr>
<tr>
<td>43</td>
<td>0.1242766</td>
<td>0.1242797</td>
</tr>
<tr>
<td>53</td>
<td>0.1210505</td>
<td>0.1210731</td>
</tr>
<tr>
<td>63</td>
<td>0.09875184</td>
<td>0.09889082</td>
</tr>
<tr>
<td>73</td>
<td>0.02186507</td>
<td>0.02208837</td>
</tr>
<tr>
<td>83</td>
<td>0.000001258</td>
<td>0.000001337</td>
</tr>
<tr>
<td>93</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 5: Simulation of the Feedback connection.

cancellation of unstable internal modes. Therefore we have to be careful with the various applications since such problems might occur.

6 Concluding Remarks

In this paper we worked with a special factorization of non-linear discrete systems. After developing the relative theory, we presented an algorithm, which permits certain non-linear discrete systems to be expressed in a simpler form. A simple non-linear model-matching problem was also solved, as an application of our method to feedback design, and a BIBO stability result was presented. Extension of our method to include systems with cross-products, or systems described through Volterra series are currently under research. Robust control applications are also being investigated.

7 Appendix

**Definition 7.1** Consider two ordered operators $S$, $S'$ of the form $S = a_{i_1i_2...i_n}$ $\delta_{i_1}\delta_{i_2}...\delta_{i_n}$, $a_{i_1i_2...i_n} \neq 0$, $i_1 \leq i_2 \leq i_3 \leq ... \leq i_n$ and $S' = a'_{i_1'i_2'...i_n}$ $\delta'_{i_1'}\delta'_{i_2'}...\delta'_{i_n'}$, $i_1' \leq i_2' \leq ... \leq i_n'$ such that $\text{deg}(S) = \text{deg}(S') = n$ and $i_1 = \text{d}(S) < \text{d}(S') =$
\(i'_1\). We say that \(S'\) is reducible with respect to \(S\) if there is an integer \(\lambda \geq 0\) such that: 
\((i_1 + \lambda, i_2 + \lambda, \ldots, i_n + \lambda) = (i'_1, i'_2, \ldots, i'_n)\).

**Definition 7.2** \(S'\) is irreducible with respect to \(S\) if it is not reducible.

**Definition 7.3** A polynomial \(R\) is called irreducible with respect to the polynomial \(D\) iff: either \(R\) contains no terms of degree equal to the maximum degree of \(D\), each term of \(R\) with degree equal to the degree of the maximum term \(S\) of \(D\) is irreducible with respect to \(S\).

**Algorithm LD**

**Input data:** Two \(\delta\)-polynomials \(A\) and \(D\).

Let \(S = a_{i_1 i_2 \ldots i_n} \delta_{i_1} \delta_{i_2} \cdots \delta_{i_n}\) be the maximum degree term of \(D\).

**STEP 1:** IF \(A\) is irreducible with respect \(D\) THEN stop ELSE goto step 2.

**STEP 2:** \(A\) contains at least one term \(S' = a_{i_1 i_2 \ldots i_n} \delta_{i_1}' \delta_{i_2}' \cdots \delta_{i_n}'\) such that \(S' = \frac{a_{i_1 i_2 \ldots i_n}}{a_{i_1 i_2 \ldots i_n}} \delta_{i_1} \delta_{i_2} \cdots \delta_{i_n} * S\). We calculate \(R = A - \frac{a_{i_1 i_2 \ldots i_n}}{a_{i_1 i_2 \ldots i_n}} \delta_{i_1} \delta_{i_2} \cdots \delta_{i_n} * D\).

**STEP 3:** Set \(A = R\) and goto step 1.

END.

**Algorithm LF**

**Input Data:** A non-linear \(\delta\)-polynomial \(A\).

**STEP 1:** Factorize \(A = a\delta_{i_1} * A\), \(a = d(A)\). IF \(\hat{A}\) contains only terms of the form \(a_{0 i_2 \ldots i_n} \delta_{i_2} \delta_{i_2} \cdots \delta_{i_n}\) THEN stop ELSE goto to step 2.

**STEP 2:** Let \(D_0\) be the polynomial consisting of all terms of \(\hat{A}\) of the form \(a_{0 i_2 \ldots i_n} \delta_{i_2} \delta_{i_2} \cdots \delta_{i_n}\). Applying the algorithm LD we find a linear polynomial \(L_0\) and a non-linear polynomial \(R_1\) such that: \(A = L_0 * D_0 + R_1\), obviously \(d(L_0) = 0\).

**STEP 3:** Goto step 1, setting \(R_1\) instead of \(A\).

END

**Algorithm prod(A)**

We suppose that there exists an algorithm denoted by \(prod(A)\), which factorizes a non-linear polynomial \(A\), accordingly to the dot-product.

**Algorithm \(l\delta F\)**

**Input data:** The linear polynomials \(L_1, L_2, \ldots, L_n\).
Write: \( L_1 = \delta_1 \star \hat{L}_1, \hat{i}_1 = d(L_1), \ldots, L_n = \delta_n \star \hat{L}_n, \hat{i}_n = d(L_n) \). **IF** \( \hat{L}_1 = \hat{L}_2 = \cdots = \hat{L}_n = L \) **THEN** give as output the \( \delta = \delta_1 \delta_2 \cdots \delta_n \) operator and the linear polynomial \( L \) **ELSE** no result, stop.

**Algorithm \( \delta F \).**

**Input data:** A \( \delta \)-polynomial \( A \).

**STEP 1:** Using the algorithm \( prod(A) \) we get: \( A = \overline{A}_1 \cdot \overline{A}_2 \cdot \overline{A}_3 \cdots \overline{A}_n \)

**STEP 2:** Using algorithm \( LF \) we write each factor in the form: \( \overline{A}_n = L_n \ast A_n, \) \( d(L_n) \geq 0, L_n \) linear. Some of \( A_n \) can be linear, including the operator \( \delta_0 \) as well.

**STEP 3:** We group the above expressions accordingly to the common factors \( A_i \). So \( IF \) \( A_{\mu_1} = A_{\mu_2} = A_{\mu_3} = \cdots = A_{\mu_k} = A_{\lambda_1}, \ldots, A_{\mu_l} = A_{\mu_m} = A_{\delta} \) \( = A_{\mu_n} = \cdots = A_{\mu_s} = A_{\lambda_n} \) **THEN** we form the groups \( (L_{\mu_1} \ast A_{\lambda_1}, \ldots, L_{\mu_k} \ast A_{\lambda_k}), \ldots, (L_{\mu_l} \ast A_{\lambda_l}, \ldots, L_{\mu_m} \ast A_{\lambda_m}) \) **ELSE** stop.

**STEP 4:** **REPEAT** for \( \phi = 1 \) **UNTIL** \( \phi = \nu \):

Using algorithm "\( \delta \circ \bar{F} \)" , with inputs \( L_{\mu_1}, L_{\mu_2}, \ldots, L_{\mu_s} \) we get: \( \delta^{(\phi)} \) and \( M_\phi \), where \( M_\phi \) linear and \( \delta^{(\phi)} \in \Delta \).

**END**

**STEP 5:** **IF** \( \delta^{(1)} = \delta^{(2)} = \cdots = \delta^{(\nu)} = \delta \) **THEN** give as output the quantities: \( \delta, M_1, M_2, \ldots, M_\nu, A_{\lambda_1}, A_{\lambda_2}, \ldots A_{\lambda_n} \) **ELSE** stop.

**END**

**References**


[11] S.Kotsios and N.Kalouptsidis, ” Adaptive Control for a certain Class of non-linear discrete systems”. It has been submitted to AUTOMATIC.


