IMPLEMENTATION OF INTERCONNECTIVE SYSTEMS

Thomas A. Baran *Tarek A. Lahlou*

Digital Signal Processing Group
Massachusetts Institute of Technology

ABSTRACT

This paper proposes a systematic strategy for the automated implementation of mixed constraint- and input-output-based representations of signal processing systems. Examples of the strategy are provided in synthesizing algorithms derived from signal-flow graphs having delay-free loops, as well as in performing automated system inversion. An algorithm that follows the strategy, and which has been deployed online as part of an edX course, is discussed in greater focus. Sensitivity analysis of systems designed using the algorithm is provided.

Index Terms— Algorithm synthesis, behavioral systems theory, inverse systems, signal-flow graphs, computer aided instruction

1. INTRODUCTION

The language used in describing various signal processing systems, e.g. linear and nonlinear signal-flow graphs, may be interpreted as a blend of constraint-based and input-output terminology. For example, in signal-flow graphs for allpass-warped filter structures [1], certain of the elements, such as delays, have a well-specified input-output form, while other of the elements readily form delay-free loops which implicitly represent algebraic constraints.

Techniques for the automated implementation of signal processing systems have traditionally focused, however, on scheduling input-output-based computations to form an overall system implementation, e.g. using callback functions in a "data pull" arrangement or using the techniques described in [2][3]. In certain cases where computation cannot be arranged to form an implementation, alternative techniques identifying various special cases, such as the identification and breaking of delay-free loops, have been used [4].

This paper proposes a systematic strategy for the automated implementation of mixed constraint- and input-output-based representations of signal processing systems, specifically making use of the so-called interconnective representation discussed in [5][6]. We provide examples of the strategy in synthesizing algorithms derived from signal-flow graphs having delay-free loops, as well as in performing automated system inversion. A specific algorithm following this strategy that has been used for automated grading on an edX course [7] is discussed, and sensitivity analysis is provided.

2. SYSTEM REPRESENTATION

The presented class of implementation techniques apply to systems that may be described using a variety of system representations, including e.g., block diagrams and signal-flow graphs. For consistency we formulate the techniques in this paper using the language outlined in this section, which can be readily used to describe systems represented in a variety of alternative forms.

2.1. The behavioral approach

In this paper we take the behavioral approach described in [8] in representing signal processing systems, and in particular in representing systems that are specified as an interconnection of subsystems. The goal in doing this is to facilitate the translation between constraint-based system representations, which may often be used in formulating an initial system description, and input-output representations, which would form the basis for synthesizing an implementation. In the following subsections we outline the key elements of the behavioral approach referred to in this paper.

2.1.1. Definition of behavior

Given a signal processing system or subsystem having one or more external signal- or scalar-valued variables available for interconnection with other systems or subsystems, we refer to the external variables as "terminal variables." It is customary to arrange these variables into a column vector referred to as a "terminal vector," and we will refer to the entire set of possible values of the terminal vector consistent with constraints imposed by a particular system as "the behavior" of the system [8]. With this definition, it is generally possible that there is no explicit indication about whether a given terminal variable is a system input or output.

Written formally, for a particular system R having a total of N terminal variables denoted $x_1, \ldots, x_N$, we write the terminal vector as $\underline{x} = [x_1, \ldots, x_N]^T$. The behavior of R, denoted $\mathcal{S}_R$, is written as

$$\mathcal{S}_R = \{\underline{x} : \underline{x} \text{ is consistent with the constraints imposed by} R\}\,, \quad (1)$$

i.e. $\mathcal{S}_R$ can be obtained by beginning with the set that is equivalent to the domain over which the terminal vector for R is defined, and taking the subset consisting of those terminal vectors that are permitted by R.

2.1.2. The behavior of an interconnection of systems

The interconnection of two systems results in a sharing of the values of their terminal variables, and consequently those values are subject to the constraints of both. From the behavioral viewpoint, we formalize this by saying that an interconnection of two systems corresponds to the intersection of their behaviors imposed by those variables that are shared. For example given two systems denoted P and R having dimensionally-compatible terminal vectors and having respective behaviors written $\mathcal{S}_P$ and $\mathcal{S}_R$, the sharing of their terminal vectors results in an interconnected system PR whose behavior is written $\mathcal{S}_{PR} = \mathcal{S}_P \cap \mathcal{S}_R$.

The authors wish to thank Analog Devices, Bose Corporation, and Texas Instruments for their support of innovative research at MIT and within the Digital Signal Processing Group.

2.1.3. The behavior of a map

The behavior of a system represented as a map is obtained by forming a terminal vector from its input and output variables, and determining the set of all such vectors that result by applying the map to all elements in the domain over which it is defined. We formally write the definition for the behavior of a generally nonlinear and time-varying map M, coupling an input signal $c[n] \in \mathcal{C}$ to an output signal $d[n] \in \mathcal{D}$, as

$$\mathcal{S}_M = \left\{ \begin{bmatrix} c[n] \\ M(c[n]) \end{bmatrix} : c[n] \in \mathcal{C} \right\}. \tag{2}$$

With the definition in Eq. 2 we have for simplicity ordered inputs before outputs in the terminal vector, and alternative orderings may also generally be used. In particular as was discussed in [5][6], an invertible map is behaviorally-equivalent to its inverse when the input and output terminal variables are appropriately exchanged.

Referring to Eq. 2, if M is a linear map realized as multiplication by a matrix G, and if $\mathcal{C}$ is a vector space, then we can write

$$\mathcal{S}_G = \left\{ \begin{bmatrix} \underline{c} \\ G\underline{c} \end{bmatrix} : \underline{c} \in \mathcal{C} \right\}. \tag{3}$$

We conclude from Eq. 3 that the behavior associated with a linear map is a vector space.

2.1.4. The behavior of an interconnection of linear, memoryless maps

An interconnection of linear, memoryless maps is a vector space. This follows from the previous discussion, in particular from the point that an interconnection of systems corresponds to an intersection of behaviors, that the behavior of a linear map is a vector space, and that in general an intersection of vector spaces is itself a vector space.

2.2. Interconnective system representation

Critical to the class of implementation methods discussed in this paper is the form of system representation referred to in [5][6], which is specifically referred to as "interconnective." The goal in using an interconnective representation is to facilitate the separation of the behaviors of the subsystems in an overall system from the relationships that couple them together, in particular so that the techniques discussed in this paper can directly utilize these coupling relationships.

In particular in interconnective form, a system is viewed as having two parts: constitutive relations (CRs), e.g. a set of possibly nonlinear and time-varying subsystems that are allowed to have memory, and a memoryless, linear interconnecting system (LI) to which the subsystems and overall system input and output are connected.

3. GENERAL IMPLEMENTATION STRATEGY

The general strategy for implementation pertains to a system represented in interconnective form, which for example can be readily obtained from a signal-flow representation by appropriately labeling sub-elements as either being CRs or as being part of the LI. We assume that given any input-output configuration for which the CRs are known to be computable, we have access to functions for computing them. I.e. given a CR represented as an invertible map coupling two variables, it is generally possible that a forward function and and inverse function would be available.

The following steps summarize the general strategy for obtaining an implementation from a system represented as mentioned in an interconnective form:

1. Determine the behavior of the LI, which will be a finite-dimensional vector space.
2. Select an input-output configuration for the LI that allows for individual computability of the CRs, as well as for computability of the LI.
3. Given the input-output configuration selected in the previous step, determine a linear map having the behavior of the LI.
4. Determine an order of computation for the CRs that will be used to schedule how the CRs pass data through the LI. If no such order can be found, return to Step 2 and select a different input-output configuration.

We comment upfront that by following this sequence, any system for which an implementation would have been found using [2][3] would also have an implementation found using this strategy.

Examples illustrating the use of this strategy are depicted in Figs. 1-2. Fig. 1 in particular depicts its use in realizing a recursive system involving allpass elements, e.g. as may be desired with the applications identified in [1]. Fig. 1 also indicates the effectiveness of the strategy in synthesizing computable implementations from signal-flow representations having delay-free loops, and in this sense represents an automated way to obtain a solution to the class of problems discussed in, e.g. [4]. Fig. 2 depicts the use of the general strategy in obtaining two behaviorally-equivalent implementations for the nonlinear system discussed in [5][9], indicating the structure of its implementation both as a forward and an inverse map. The general strategy may thereby be viewed as a blueprint for the automated inversion of systems traditionally performed using graph-based techniques, e.g. [10][11][5].

4. EXAMPLE ALGORITHM

In this section we discuss an example algorithm adhering to the general implementation strategy described in Section 3 by introducing a "constraint form" encoding of the LI in order to describe the behavior of the LI as a finite-dimensional vector space. A matrix whose behavior equals the behavior of the LI is then generated from this representation. We then discuss a method by which a precedence map or scheduling of the linear constraints and CRs is determined as well as a method for determining clock domains compatible with a synchronous implementation of any multirate relations. In generating the ordered sequence of functions, we rely upon the availability of an input-output configuration of the system described in interconnective form such that all CRs have functional realizations.

4.1. Determining the behavior of the LI

Define a behavior-generating matrix B for a fixed LI with behavior $\mathcal{S}_{LI}$ such that the nullspace of B consists of all terminal vectors consistent with the linear, time-invariant and memoryless constraints of the signal-flow system, i.e.,

$$B\underline{x} = \underline{0}, \qquad \forall \underline{x} \in \mathcal{S}_{LI}. \tag{4}$$

A behavior-generating martrix B satisfying Eq. 4 can be populated using the following straightforward procedure: for each constraint in the LI, encode one or multiple rows of B where the coefficients for the row(s) correspond to the coefficients of the constraint written in constraint form, i.e., as a homogeneous linear equation. Figure 3

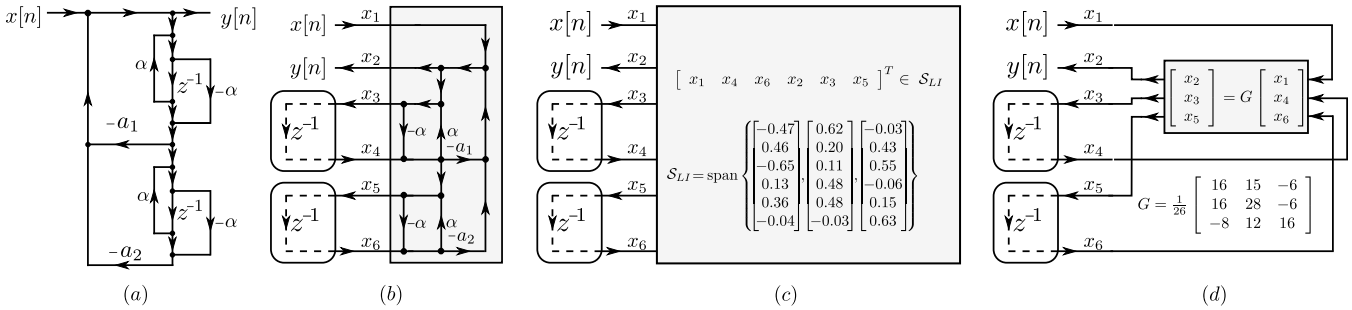

Fig. 1. Example illustrating the presented general strategy for obtaining system implementations, shown for a second-order all-pole system in direct form, with the delays having been replaced by first-order allpass elements. (a) Original system, containing delay-free loops. (b) System represented in interconnective form. (c) System indicating the computed behavior of the LI, computed for a system where $a_1 = -1$ $a_2 = 0.5$, and $\alpha = 0.5$. (d) Computable system involving a matrix G having the behavior of the LI depicted in (c).

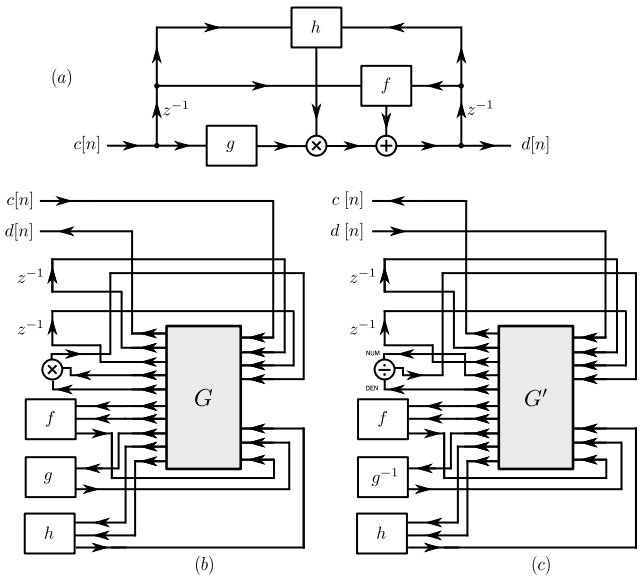

Fig. 2. Example illustrating the use of the presented automated strategy in obtaining two behaviorally-equivalent implementations of an original system (a), with (b) representing its implementation as a forward map and (c) representing its implementation as an inverse map.

depicts three example signal-flow elements along with their corresponding constraint form equations. Then, by definition we have that null $\{B\} = \mathcal{S}_{LI}$.

4.2. Realizing LI as a matrix

For a given behavior-generating matrix B, we next define a related matrix X which satisfies the following two properties: (i) the range of X is equal to the behavior of the LI, and (ii) the rows of X are ordered such that a partitioning yields an upper block X_1 corresponding to the N_i input variables and a lower block X_2 corresponding to the N_o output variables as described by the input-output configuration. The matrix X may be assembled by permuting the rows of the singular vectors of B corresponding to zero singular values.

An important question at this stage deals with the existance of a realization of the LI for the given input-output configuration as a matrix. Consider the description of the behavior of the LI, i.e. $\mathcal{S}_{LI}$,

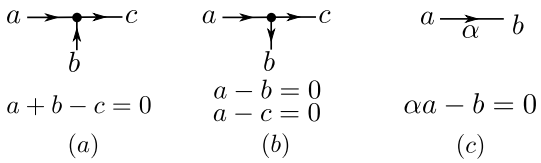

Fig. 3. Example signal-flow elements and constraint form equations for (a) a summation node, (b) a distribute node, and (c) a constant-coefficient multiplier.

written in terms of X_1 and X_2 as

$$\mathcal{S}_{LI} = \text{range} \left\{ \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \right\}. \tag{5}$$

It is straightforward to show from this equation that for the realization of the LI as a matrix G to exist it is both necessary and sufficient for the associated block matrix X_1 to be invertible. It then immediately follows that when the realization exists it is given by

$$G = X_2 X_1^{-1}. \tag{6}$$

4.3. Determining precedence relations

The issue of computability for a signal-flow system in this paper is closely associated with the existance of delay free loops within the system. In Section 4.2 we realized the behavior of the LI as a matrix, i.e. with no such loops, thus a sufficient condition for scheduling of the total system is the absence of delay free loops from the CRs [2][3]. A procedure for identifying a precedence map or schedule for executing the linear constraints, i.e., rows of G, and constitutive relations is now described.

We next make use of the input-output configuration in defining a sequence of precedence vectors $\underline{\mathrm{p}}^{(\ell)}$ where each output or end-loop variable is represented by a coordiante of $\underline{\mathrm{p}}^{(\ell)}$, i.e., $\underline{\mathrm{p}}^{(\ell)}$ has a dimensionality equal to the number of CR inputs and overall system outputs. The term end-loop variable is suggestive of the fact that these variables are needed to proceed to the next iteration of the runloop under design. Let $D^{(LI)}$ and $D^{(CR)}$ respectively denote binary dependency matrices where $D^{(LI)} = 1$ for non-zero entries of G and $D^{(CR)}$ similarly encodes which linear interconnection outputs are needed for the computation of each CR and overall system output. The sequence of precedence vectors are then populated by iterating

$$\underline{\mathrm{p}}^{(\ell)} = \left(D^{(LI)} D^{(CR)} \right)^T \underline{\mathrm{p}}^{(\ell-1)}, \quad l = 1, \ldots, L \tag{7}$$

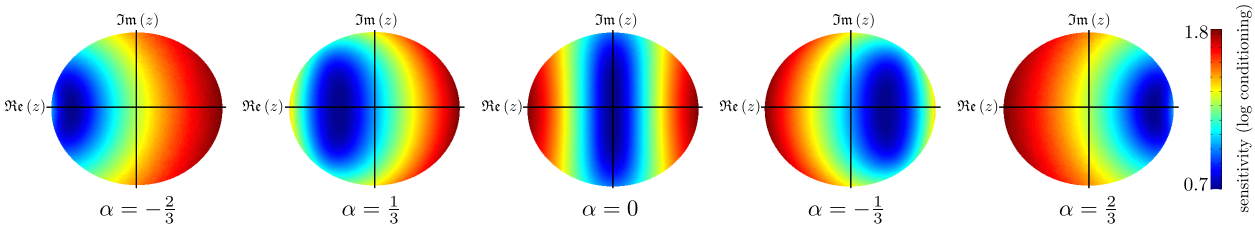

Fig. 5. An illustration of the numerical sensitivity of the LI matrix G for the second-order allpass warped system in Figure 1 computed as the log of the modified condition number for allpass parameters $\alpha = 0, \pm\frac{1}{3}, \pm\frac{2}{3}$. For each allpass parameter, G is generated and analyzed for all complex conjugate pole pairs located within the unit circle of the z-plane.

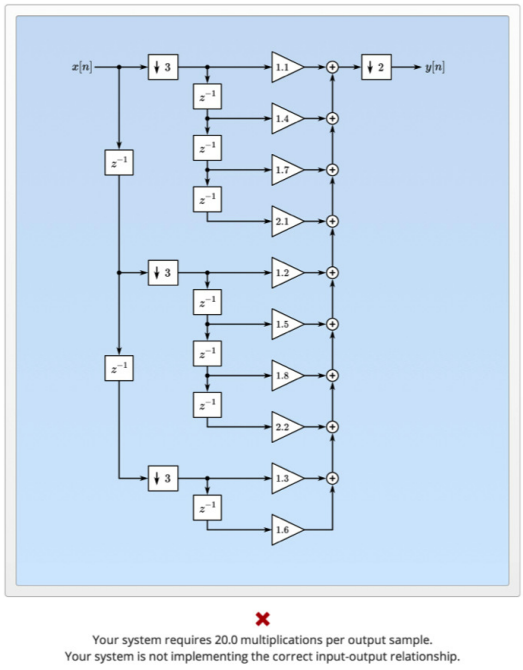

Fig. 4. User interface for an example problem on the edX course [7] utilizing the algorithm in Section 4 in evaluating student input.

where $\underline{\mathrm{p}}^{(0)}$ is initialized to 1 for each overall system output and L is the degree of the nilpotent matrix $D^{(LI)}D^{(CR)}$. The runloop is populated in reverse by ordering the CRs and preceeding them with the necessary linear equations or rows of G such that the functional dependencies of the CRs are satisfied. The final ordering of the CRs corresponds to the end-loop variable coordinates first non-zero value appearing in $\underline{\mathrm{p}}^{(\ell)}$ for increasing ℓ.

4.4. Computing clock domains

In computing clock domains, we associate with each interconnection terminal variable x_k a clock rate r_k. Then, collect a constraint for each x_i, x_j described by the dependency matrices $D^{(LI)}$ and $D^{(CR)}$ written as

$$r_i r_j^{-1} = \mu \tag{8}$$

where $\mu > 0$ describes the rate relationship between x_i and x_j, i.e. x_i operates at μ times the rate of x_j. Rewriting this set of constraints with $\hat{r}_k = \log r_k$ yields a system of linear equations of the form

$$\hat{r}_i - \hat{r}_j = \log \mu \tag{9}$$

where any non-trivial solution uniquely specifies a choice of rates r_k. Conditions for the existance of a synchronous implementation of the signal-flow system is then twofold. First, the right hand side of Eq. 9 must lie in the range of the associated system. This is equivalent to having no contradictory constraints, e.g., a delay free loop through an expander. Second, all rates must be rational, i.e. each element r_k may be made integer by proper normalization. When a system consists soley of integer rate conversion elements, e.g., decimators and expanders, the second condition will always be satisfied. The rates obtained are then used to keep track of which iterations particular functions are executed on.

4.5. Online implementation

The algorithm described in this section has been implemented in Python and integrated with the online grading system for the edX course, *6.341x: Discrete-Time Signal Processing* [7]. A screen capture of the user interface for an example problem utilizing the algorithm for student evaluation is depicted in Fig. 4. Referring to this figure, the student has been prompted to design a polyphase implementation for a rate-conversion system, and the algorithm in this section has been used to provide feedback to the student that the particular system entered did not exhibit the expected input-output relationship, obtained by applying test signals to the resulting runloop implementation. Still referring to Fig. 4, the grading system has additionally provided feedback about the computational cost of the system, computed using the principles discussed in Subsection 4.4.

5. SENSITIVITY ANALYSIS

In traditional representations of signal processing systems, i.e. linear and nonlinear signal-flow graphs, finite-precision effects such as coefficient quantizaion are typically evaluated using various sensitivity theorems and are discussed, e.g. in [12]. A canonical tool which measures perturbation sensitivity of linear systems with respect to a number of perturbation sources is the relative condition number, i.e., the ratio of extremal singular values [13]. In this paper we use a modified condition number taken as ratio of largest to smallest non-zero singular values. A justification for this modification follows from the fact that the number of inputs and outputs to the LI are generally not equal.

For a given signal-flow system, understanding sensitivity is straightforwad when no delay free loops are present and less so in the general case. Consider the previous example of all-pass filter composition from Fig. 1. A well known design strategy in this context selects an allpass factor α in order to place the pole locations of the composed filter in a region of the z-plane with desireable sensitivity properties. Consistent with these results, Figure 5 depicts the log of the modified condition number of the LI matrix G assuming the poles are chosen in complex conjugate pairs within the unit circle for several values of α. The optimal regions of the z-plane for a fixed allpass factor using this metric is consistent with those expected from conformal mapping theory [1].

6. REFERENCES

[1] P.A Regalia, S.K. Mitra, and P.P. Vaidyanathan, "The digital all-pass filter: a versatile signal processing building block," *Proceedings of the IEEE*, vol. 76, no. 1, pp. 19–37, Jan 1988.

[2] R. E. Crochiere, *Digital network theory and its application to the analysis and design of digital filters*, Ph.D. thesis, Massachusetts Institute of Technology, 1974.

[3] R. E. Crochiere and A. V. Oppenheim, "Analysis of linear digital networks," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 581–595, 1975.

[4] A. Härmä, "Implementation of recursive filters having delay free loops," in *IEEE Proc. Int. Conf. Acoust., Speech and Signal Process.*, 1998, pp. 1261–1264.

[5] T. A. Baran and A. V. Oppenheim, "Inversion of nonlinear and time-varying systems," in *2011 IEEE Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE)*. IEEE, 2011, pp. 283–288.

[6] T. A. Baran, *Conservation in Signal Processing Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2012.

[7] A. V. Oppenheim and T. A. Baran, *6.341x Discrete-Time Signal Processing*, on edX, Spring 2015.

[8] J. C. Willems, "The behavioral approach to open and interconnected systems," *IEEE Control Systems Magazine*, vol. 27, no. 6, pp. 46–99, 2007.

[9] A. Carini, G.L. Sicuranza, and V.J. Mathews, "On the inversion of certain nonlinear systems," *IEEE Signal Processing Letters*, vol. 4, no. 12, pp. 334–336, 1997.

[10] S. Y. Kung, "Inverse systems and an inversion rule," in *Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications, 1977 IEEE Conference on*, Dec 1977, pp. 771–776.

[11] S. J. Mason and H. J. Zimmerman, *Electronic Circuits, Signals, and Systems*, Wiley, 1960.

[12] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.

[13] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, June 1997.