SUBSYSTEMWISE SIMULATION OF DISCRETE-TIME INTERCONNECTED DYNAMICAL SYSTEMS

by

G. Guardabassi

Memorandum No. UCB/ERL M82/8

16 February 1982

SUBSYSTEMWISE SIMULATION OF DISCRETE-TIME INTERCONNECTED DYNAMICAL SYSTEMS

by

G. Guardabassi

Memorandum No. UCB/ERL M82/8

16 February 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

November 30, 1981

Subsystemwise Simulation of Discrete-Time Interconnected Dynamical Systems

by

G. Guardabassi

Politecnico di Milano-Italy

ABSTRACT

Any structurally well-posed discrete-time interconnected dynamical system S can be simulated subsystemwise on any finite interval of time of length L. That is, the system output can be computed by a finite number Q of simulations of the component subsystems only. In general, $Q \leq L$. If the system is linear and time-invariant, then the number of partial simulations can be made less or equal to the generalized index of the most strongly connected component of S plus two.

Research supported by Office of Naval Research under Contract Nooo 14-80-C-0507 to Electronic Research Lab., University of California, Berkely, and by CNR: Centro di Teoria dei Sistemi, Contr. No. CT80.02086.07.

1. INTRODUCTION

Consider an interconnected system $S = (S_1, S_2, \ldots, S_N; I)$ of the form :

$$S_k : \begin{cases} x_k(t+1) = f_k(x_k(t), u_k(t), t), \\ y_k(t) = g_k(x_k(t), u_k(t), t), \end{cases}$$

$$I : \begin{cases} u_k(t) = \gamma_k(y(t), u_o(t), t), & k = 1,2,\ldots, N, \\ y_o(t) = \sigma(y(t), u_o(t), t), \end{cases}$$

where, for any $k \in \{0,1,2,\ldots, N\}$, $x_k(t) \in R^{n_k}$, $u_k(t) \in R^{m_k}$, $y_k(t) \in R^{p_k}$, $x_o \triangleq |x_1' \; x_2' \; \ldots \; x_N'|'$, $y \triangleq |y_1' \; y_2' \; \ldots \; y_N'|'$ so that $n_o = \sum_{k=1}^{N} n_k$, $y(t) \in R^p$, $p = \sum_{k=1}^{N} p_k$; $t \in T \triangleq \{\bar{t}, \bar{t}+1, \ldots, \bar{t} + L\}$. Thus, $u_o(t)$, $x_o(t)$ and $y_o(t)$ denote the input, the state and the output, respectively, of the overall system S. Note that $y_o(t)$ and $y(t)$ are quite different objects.

Furthermore, suppose that a computing facility is available whose central unit can efficiently handle the "interaction" I plus any subsystem S_k of S, but is not capable to efficiently handle (simulate) the system S as a whole, due to its exceedingly large order n_o. We are interested, then, in the following problem (Subsystemwise Simulation Problem).

Given T, namely $\bar{t}$ and L, $\bar{x} \triangleq x_o(\bar{t})$ and $u_o(.)$ on T, compute $y_o(.)$ on (any right subinterval of) T by handling no more than a single subsystem at a time.

If moving any subsystem in and out the central computing unit is far the most time-expensive operation to be executed, a first conceivable criterion to evaluate the performance of any possible algorithm for the problem above is just the number of such in and out moves needed to completely solve it.

In this paper we present and discuss two kinds of algorithms. The algorithms of the first class, denoted by A, are both general and natural: they attempt to sequentially decompose each elementary transition (time-step) of S into N

elementary transitions of the single subsystems. The algorithms of the second class, denoted by B, are apparently more sophisticated, yet will be proved not to be better, in general, than the ones of the first class inasmuch as the total number of times each subsystem needs to be called in the central computing unit (to simulate it on an interval of time of any length) is adopted as overall performance index. However, if the interconnected system is both linear and time-invariant, then a class B algorithm can be devised which, according to the afore mentioned criterion, is better than any algorithm in class A, whenever the length L of the time interval T minus the generalized index of the most strongly connected component of S is greater than two.

2. CLASS A: STEP-BY-STEP DECOMPOSITION ALGORITHMS

In the present section we shall be concerned with the possibility of decomposing the computation of each elementary transition of S into the sequential computation of the elementary transitions of the subsystems S_k, $k = 1,2,\ldots,N$. In order for this to be a real possibility it is obviously necessary that , for each $t \in T$, an ordering $(S_{j_1}, S_{j_2}, \ldots, S_{j_N})$ of the subsystems exists such that, once the output at time t of the first k-1 subsystems has been computed, all the information needed to compute the output at time t of the k-th subsystem is available. We make this notion more specific by the following definition.

Definition 1

The ordering $(S_{j_1}, S_{j_2}, \ldots, S_{j_N})$ of the subsystems of S is feasible at time $t \in T$ if $\tilde{y}, \tilde{\tilde{y}} \in R^p$ and $\tilde{y}_{j_1} = \tilde{\tilde{y}}_{j_1}$, $\tilde{y}_{j_2} = \tilde{\tilde{y}}_{j_2}, \ldots, \tilde{y}_{j_{k-1}} = \tilde{\tilde{y}}_{j_{k-1}}$ imply

$$g_{j_k}(x_{j_k}, \gamma_{j_k}(\tilde{y}, u_o, t), t) = g_{j_k}(x_{j_k}, \gamma_{j_k}(\tilde{\tilde{y}}, u_o, t), t)$$

for all $k \in \{1,2,\ldots,N\}$ and all $(x_{j_k}, u_o) \in R^{n_{j_k}} \times R^{m_o}$.

A first step-by-step decomposition algorithm can then be quite naturally constructed as follows. The algorithm computes first, sequentially, the output of each subsystem at $\bar{t}$; then, for each $t \in T$, it updates the state of a single subsystem at a time and computes the next value of the output.

Algo A1

Data : $T \sim (\bar{t}, L)$, $\bar{x}$, $\{u_o(t), t \in T\}$.

Step 0 : Set $t = \bar{t}$, $k = 1$, $y = y^+ = 0 \in R^p = R^{p_1} \times R^{p_2} \times \ldots \times R^{p_N}$.

Step 1 : Find an ordering of the subsystems of S which is feasible at time t; rename then the subsystems of S so that the natural ordering $(S_1, S_2, \ldots, S_N)$ is feasible at time t.

Step 2 : Call $S_k = (f_k, g_k; x_k)$. Set $x_k = \bar{x}_k$ and $y_k = g_k(x_k, \gamma_k(y, u_o(t), t), t)$.

Step 3 : If $k = N$, compute $y_o = \sigma(y, u_o(t), t)$, set k=1 and go to step 4. Else, set $k = k+1$ and go to step 2.

Step 4 : Find an ordering of the subsystems of S which is feasible at time t+1; rename then the subsystems of S so that the natural ordering $(S_1, S_2, \ldots, S_N)$ is feasible at time t+1.

Step 5 : Call $S_k = (f_k, g_k; x_k)$. Set $x_k = f_k(x_k, \gamma_k(y, u_o(t), t), t)$ and $y_k^+ = g_k(x_k, \gamma_k(y^+, u_o(t+1), t+1), t+1)$.

Step 6 : If $k = N$, go to step 7. Else, set $k = k+1$ and go to step 5.

Step 7 : Set $t = t+1$, $y = y^+$ and compute $y_o = \sigma(y, u_o(t), t)$.

Step 8 : If $t = \bar{t} + L$, stop. Else, set $k = 1$ and go to step 4.

It should be apparent that the critical steps in Algo A1 are step1 and step 4. In other words, we should like to qualify the algorithm by specifying the conditions under which a feasible ordering exists at any $k \in T$. Next, we may want to avoid the reordering of the subsystems of S at each time instant $t \in T$; thus, we should also like to investigate the conditions under which step 4 can be safely dropped out from Algo A1.

In order to perform such an analysis, we need some further definitions and some preliminary results. Most of them refer to the "structure" I of S, namely to some fundamental properties of a time-varying digraph $G_S(t)$ naturally associated with S and called the system graph of S at time t.

We may construct $G_S(t)$ in the following way. First, we associate each variable u_k and y_k, $k = 1,2,\ldots,N$, with as many vertices, called vertex u_k and vertex y_k, respectively. Next, we draw an arc from vertex u_k to vertex y_k, $k = 1,2,\ldots,N$; these arcs will henceforth be called subsystem arcs. Furthermore, we draw an arc from vertex y_j to vertex u_k if and only if u_k does in fact depend upon y_j, i.e. if and only if $\phi_k(.,y,u_o,t)$ is nonconstant for some $(y,u_o) \in R^p \times R^{\bar{m}_o}$, where, for any $z_j \in R^{p_j}$,

$$\phi_k(z_j,y,y_o,t) \triangleq \gamma_k(|y_1' \; y_2' \; \cdots \; y_{j-1}' \; z_j' \; y_{j+1}' \; \cdots \; y_N'|', \; u_o, t) \; ;$$

these arcs will henceforth be called interconnecting arcs. Of course, two or more variables known to be identical at t, because of I, can be associated with the same vertex, which will be arbitrarily given the name of one of them. Finally, the set of vertices and the set of arcs of $G_S(t)$ will be denoted V and $W(t) \subseteq V \times V$, respectively.

Example 1

Consider an extremely simple interconnected system S made up of two subsystems only, i.e. $S = (S_1, S_2; I)$. As for the interconnection I, let $u \triangleq |u_1' \; u_2'|'$ and suppose that

$$u(t) = \Gamma(t)\, y(t) + \gamma_o(u_o(t))$$

$$\Gamma(t) \triangleq \begin{vmatrix} I & I \\ -I & O \end{vmatrix}_{(t-\bar{t})} .$$

It is easy to check that $\Gamma(.)$ is periodic of period 6. So $G_S(.)$ must be periodic as well and its period must be equal to 6 or to one of its integer divisors. In this case, it turns out that $G_S(.)$ is periodic of period 3. For any $i \in \{0,1,2\}$, let $\tau_i = \{t: |t-\bar{t}|_3 = i\}$, where $|.|_3$ means modulo 3. Then, the three possible forms of the system graph of S on τ_o, τ_1 and τ_2 are shown in Fig. 1-a, b and c, respectively.

Definition 2

A subsystem S_k of S is purely dynamic at $t \in T$ if $g_k(x_k,.,t)$ is constant, for all $x_k \in R^{n_k}$. Accordingly, the corresponding subsystem arc of $G_S(t)$ is said to be a purely dynamic subsystem arc.

Definition 3

The reduced system graph of S at time $t \in T$ is the graph $G_{SR}(t) = (V, W_R(t))$ obtained by removing from $G_S(t) = (V, W(t))$ all purely dynamic subsystem arcs.

Definition 4

The system S is structurally well-posed at $t \in T$ if $G_{SR}(t)$ is acyclic (i.e. does not contain any closed path).

Definition 5

The system S is structurally well-posed on $\tau \subseteq T$ if it is structurally well-posed for all $t \in \tau$.

Definition 4 and Definition 5 are essentially motivated by the main result in |1|. In fact, it is easy to proove that if S is structurally well-posed on any left subinterval $\tau \in T$, then it is well-posed on τ, in the sense that the state and output motions of S on τ exist, are unique and depend causally on the input $u_o(.)$ on τ . Structural well-posedness is of course only a sufficient condition for the well-posedness of S; however, it is in fact the weakest condition which ensures well-posedness for all possible specifications of S, namely for all possible specifications of functions f_k, g_k and γ_k, $k = 1,2,\ldots,N$. In this structural sense, it can therefore be said that such a condition is also necessary.

The following two lemmas will be used in the proof of the main result of this section, namely Theorm 1.

Lemma 1.

If S is structurally well-posed at $t \in T$, any ordering of the subsystems of S such that

i) all purely dynamic subsustems precede each subsystem of S which is not purely dynamic,

ii) the ordering of the subsystems which are not purely dynamic fits with

the partial ordering induced on them by $G_{SR}(t)$, is a feasible ordering at time t.

Proof. The proof is by contradiction. If such an ordering $(S_{j_1}, S_{j_2}, \ldots, S_{j_N})$ is not feasible at time t, then in view of Definition 1, there must exist $k \in \{1,2,\ldots,N\}$, $(x_{j_k}, u_o) \in R^{n_{j_k}} \times R^{m_o}$ and $\tilde{y}, \breve{y} \in R^p$ such that

a) $\tilde{y}_{j_1} = \breve{y}_{j_1}$, $\tilde{y}_{j_2} = \breve{y}_{j_2}$, ..., $\tilde{y}_{j_{k-1}} = \breve{y}_{k-1}$,

b) $g_{j_k}(x_{j_k}, \gamma_{j_k}(\tilde{y}, u_o, t), t) \neq g_{j_k}(x_{j_k}, \gamma_{j_k}(\breve{y}, u_o, t), t)$.

This means that: 1) S_{j_k} is not purely dynamic, 2) $u_{j_k}(t)$ does depend upon the output of some subsystem S_{j_h}, $h \geq k$. Hence, $G_{SR}(t)$ must have an interconnecting arc from vertex y_{j_h} to vertex u_{j_k}. Furthermore, in view of (i), S_{j_h} cannot be purely dynamic since S_{j_k} is not purely dynamic and precedes it. Both S_{j_k} and S_{j_h} being not purely dynamic and the existence in $G_{SR}(t)$ of an arc from vertex y_{j_h} to vertex y_{j_k}, with $h \geq k$, either contradicts the assumption that S is structurally well-posed at t, if $h = k$, or condition (ii), if $h > k$.

Corollary

If all the subsystems of S are purely dynamic at $t \in T$, then S is structurally well-posed at t and every ordering of the subsystems is feasible at t.

Lemma 2

If there exists a feasible ordering $(S_{j_1}, S_{j_2}, \ldots, S_{j_N})$ of the subsystems of S at time $t \in T$, then $G_{SR}(t)$ is acyclic.

Proof. Feasibility at time t implies that, for all k = 1,2,..., N, either S_{j_k} is purely dynamic or u_{j_k} does not depend on y_{j_h}, for any $h \in \{k, k+1,\ldots,N\}$. Hence, the input of a subsystem which is not purely dynamic cannot depend upon the output of any not purely dynamic subsystem unless it is a preceding subsystem (in the considered feasible ordering). Since, by definition, any closed path of $G_{SR}(t)$ must alternate between interconnecting and not purely dynamic subsystem arcs, the only possible conclusion is that $G_{SR}(t)$ is acyclic.

Theorem 1

Algo A1 solves the subsystemwise simulation problem if and only if S is structurally well-posed on T.

Proof. It should be apparent, from the preceding discussion, that Algo A1 actually solves the subsystemwise simulation problem if and only if at least one feasible ordering of the subystems of S exists, for each $t \in T$. The if part of the theorem is then an immediate consequence of Lemma 1. As for the only if part, note that if S is not structurally well-posed on T, then there exists $t \in T$ such that $G_{SR}(t)$ is not acyclic. Hence, by Lemma 2, there does not exist any ordering of the sybsystems which is feasible at t.

Admittedly, the determination of a feasible ordering of the subsystems of S, at each $t \in T$, may be a cumbersome task. When a large number of simulations, over a long interval of time (L large), has to be carried out for a given class of systems, it may in fact be rewarding to know a priori whether or not step 4 can be safely removed from Algo A1. As a trivial example, suppose that the interconnected system S is not only structurally well-posed

but also structurally time invariant on T, i.e. that $G_{SR}(.)$ is acyclic and constant on T; then, Algo A1 does obviously work even without step 4. But the converse is not true: in order that the simplified version of Algo A1 based on a fixed ordering of the subsystems may work much less than structural time-invarinace on T is required. In fact, in view of the corollary of Lemma 1, a second such case is readily recognized to occur when all the subsystems of S are purely dinamic on T, no matter of whether $G_{SR}(.)$ is constant on T or not. To be more specific on the above issue we need to introduce some further definitions which, on the other hand, will turn out to be useful in next section too.

Definition 6

The ordering $(S_{j_1}, S_{j_2}, \ldots, S_{j_N})$ of the subsystems of S is feasible on $\tau \subseteq T$ if it is feasible for all $t \in \tau$.

Algo A2

Same as Algo A1 but for step 4, which is dropped out, and step 1, which is modified as follows:

Step 1: Find an ordering of the subsystems of S which is feasible on T; rename then the subsystems of S so that the natural ordering $(S_1, S_2, \ldots, S_N)$ is feasible on T.

Definition 7

For any $\tau \subseteq T$, the graphs $G_S^\tau = (V, W^\tau)$, $W^\tau \triangleq \bigcup_{t\in\tau} W(t)$, and $G_{SR}^\tau = (V, W_R^\tau)$ $W_R^\tau \triangleq \bigcup_{t\in\tau} W_R(t)$, are the cumulative system graph and the cumulative reduced system graph over τ, respectively.

Definition 8

The System S is invariantly structurally well-posed on $\tau \subseteq T$ if the cumulative reduced system graph over τ is acyclic.

Remark 1

If the system S is at once structurally well-posed and structurally invariant on $\tau \subseteq T$, that is $G_{SR}(.)$ is acyclic and constant on τ, or all the subsystems of S are purely dynamic on τ, then S is invariantly structurally well-posed on τ, but the converse is not true.

Therem 2

Algo A2 solves the subsystemwise simulation problem if and only if S is invariantly structurally well-posed.

Proof. It is obvious that Algo A2 solves the subsystemwise simulation problem if and only if there exists an ordering of the subsystems of S which is feasible on T. In view of Lemma 1 and Lemma 2, such an ordering exists if and only if G_{SR}^{τ} is acyclic.

Example 2 .

Consider again the interconnected system of Example 1. A simple thought enables one to recognize that such a system is structurally well-posed on any interval T of length greater than two if and only if S_1 is purely dynamic on $(\tau_o \cup \tau_2) \cap T$ and S_2 is purely dynamic on $(\tau_o \cup \tau_1) \cap T$. The system S is invariantly structurally well-posed on T if and only if both S_1 and S_2 are purely dynamic on T. Hence, looking at S_1 and S_2, separately, it is possible to determine which algorithm is best suited to the problem at hand.

3. CLASS B: A "FUNCTION-SPACE" APPROACH

In Section 2, the strong relationship between structural well-posedness and the possibility of decomposing the computation of each elementary time-transition of S into a sequence of similar computations involving no more than a single subsystem at a time has been demonstrated. However, once a subsystem has been moved into the central computing unit, we might well be ready to leave it run over an interval of time even much longer than one (e.g. L elementary time-transitions), if the prospect is that the total number of subsystem moves needed to solve the overall problem can be significantly reduced. To explore this possibility, let us recast the subsystemwise simulation problem in slightly different terms.

For any vector function $v(.)$ defined on T and any integer $\lambda \leq L$, let $\underset{\sim}{v}(\lambda) \triangleq |v'(\bar{t})\ v'(\bar{t}+1) \ldots\ v'(\bar{t}+\lambda)|'$. Furthermore, for any $k = 1,2,\ldots,N$, define the local simulation problem $P_k(\lambda)$ as the one of computing $\underset{\sim}{y}_k(\lambda)$, given $x_k(o) = \bar{x}_k$ and $\underset{\sim}{u}_k(\lambda)$. Then, problem $P_k(\lambda)$ may be described by a (readily computable) function $\pi_{k\lambda}$ such that :

$$P_k(\lambda) : \quad \underset{\sim}{y}_k(\lambda) = \pi_{k\lambda}(\bar{x}_k, \underset{\sim}{u}_k(\lambda)) .$$

In a similar way, the interaction and output equations I can be reformulated as follows:

$$\underset{\sim}{u}_k(\lambda) = \gamma_{k\lambda}(\underset{\sim}{y}(\lambda), \underset{\sim}{u}_o(\lambda)), \qquad k = 1,2,\ldots,N,$$
$$\underset{\sim}{y}_o(\lambda) = \sigma_k(\underset{\sim}{y}(\lambda), \underset{\sim}{u}_o(\lambda)) ,$$

where the functions $\gamma_{k\lambda}$, $k = 1,2,\ldots,N$, and σ_λ are trivially induced by γ_k and σ, respectively. With this notation, the original subsystemwise si-

mulation problem has been equivalently reformulated as a composite problem $P_o(\lambda)$, the analysis of which will be the object of the present section.

Following the usual pattern |2|-|4|, in order to find a way to solve $P_o(\lambda)$ by solving (possibly many times) no more than a single subproblem at a time, we look at the interaction graph $G(\lambda)$ associated with $P_o(\lambda)$.

A little thought enables one to recognize that, in this particular case, the interaction graph is nothing but the cumulative system graph over $\tau \triangleq \{\bar{t}, \bar{t}+1, \ldots, \bar{t}+\lambda\}$, with the variables u_k and y_k substituted by $u_k(\lambda)$ and $\underset{\sim}{y}_k(\lambda)$, respectively, for all $k = 1,2,\ldots,N$, and the subproblems $P_k(\lambda)$ simply associated with the subsystem arcs (from vertex $\underset{\sim}{u}_k(\lambda)$ to vertex $\underset{\sim}{y}_k(\lambda)$).

Example 3

Consider once more the interconnected system of Example 1. For any $\bar{t}$ and any $\lambda \geq 2$, the interaction graph $G(\lambda)$ associated with $P_o(\lambda)$, simply obtained by superposition of the three graphs of Fig. 1, is shown in Fig. 2. As an additional example, assume again $S = (S_1, S_2; I)$, where I is defined as in Example 1, but

$$\Gamma(t) \triangleq \begin{vmatrix} 0 & I \\ \Gamma_{21}(t) & \Gamma_{22}(t) \end{vmatrix};$$

$\Gamma_{21}(\cdot)$ and $\Gamma_{22}(\cdot)$ are arbitrarily given functions (not identically zero on τ). The interaction graph associated with $P_o(\lambda)$, is shown in Fig. 3, where the identity $u_1 = y_2$ has been explicitly taken into account.

In both cases (Fig. 2 and Fig. 3), it should be apparent that simulating S subsystemwise over T is exactly equivalent to solving $P_o(L)$ by solving first, possibly more than once but no more than one at a time, the subproblems

$P_1(L)$ and $P_2(L)$ and then combining (if possible) these local solutions to get the global one. Of course, in order for the decomposition above to be really of interest, the computational effort required by the last step should not significantly exceed the one needed to solve the larger subproblem.

Now, if the interaction graph G(L), or equivalently G_S^T, is acyclic, then $P_o(L)$ is said to be a partially or totally sequential problem, according to whether the ordering induced by G(L) on the set of the subproblems is a partial or a total ordering, respectively. Of course, solving a sequential problem $P_o(L)$ by solving one at a time its subproblems is trivial. If, on the contrary, G(L) exhibits some closed paths, by a standard strongly connected components analysis |5|, |6|, we can easily carry $P_o(L)$ into a (partial or total) sequence of composite problems associated with the strongly connected components of G(L). Note that each strongly connected component of G(L) identifies a strongly connected component of S over T. Without any loss of generality, we can therefore assume from now on that G(L) is strongly connected.

Let $\tilde{E}$ be an essential set of G(L), i.e. a set of vertices the removal of which leaves the graph acyclic |7|. By definition, associated with $\tilde{E}$, there is in G_S^T a corresponding essential set E. Denote by z_E a composite vector made up of all variables u_r or y_s, r, s $\varepsilon\{2,3,\ldots,N\}$, associated with vertices of E.

If the essential set is chosen in such a way as to minimize the dimension ν of z_E, then E is an optimum essential set |8|- |14|and the minimum dimension ν^o of z_E is the generalized index of G_S^T. If S consists of single-input single-output subsystems only, then ν^o coincides with the index of G_S^T, as usually defined in the literature. If G_S^T is not strongly connected, let ν_h^o denote the generalized index of its h-th strongly connected component, h=1,2,...,q. Then $\nu^o = \sum_{h=1}^{q} \nu_h^o$.

If $\nu_i^o > \nu_h^o$, for some i, h $\in \{1,2,\ldots,q\}$, we say that the i-th component is more strongly connected than the h-th one. This leads to considering, besides the generalized index ν^o, an interconnectivety index, μ^o, defined as the generalized index of the most strongly connected component of G_S^T, i.e.:

$$\mu^o \triangleq \max_h \nu_h^o \ .$$

Note that, with the adopted notation, the elements of the vector $z_E(L)$ are the same, up to a suitable rearrangement, as the ones of all vectors $\underset{\sim}{u}_r(L)$ or $\underset{\sim}{y}_s(L)$ associated with nodes of G(L) belonging to $\tilde{E}$.

Whenever an oriented graph is used to describe an interconnection of causal objects, the variable associated with each vertex can be thought of as playing two different roles: it is the signal caused by all incoming arcs (output) and also the signal which is carried by all outgoing arcs (input). These two roles may be separated by splitting the node and adding an arc (identity) from the output to the input semivertex. By doing that for all vertices of the essential set E, we can readily see that $P_o(L)$, namely the original subsystemwise simulation problem, can be equivalently reformulated [3] as the combination of a sequential problem $P_E(L)$ and a feedback identity. This feedback canonical reformulation of $P_o(L)$ is shown in Fig. 4, where $\hat{z}_E(L)$ and $z_E(L)$ denote the input and output variables of $P_E(L)$, generated by splitting the vertices of E.

The feedforward composite problem $P_E(L)$ being sequential can easily be solved componentwise, i.e. by keeping in the central computing unit a single subsystem of S at a time. A straightforward algorithm is as follows.

Algo F^2

Data : $T \sim (\bar{t}, L)$, $\bar{x}$, $\underset{\sim}{u}_o(L)$, $\hat{\underset{\sim}{z}}_E(L)$.

Step 0: Set $k = 1$, $\underset{\sim}{u} = 0 \in R^{mxL} = R^{m_1 xL} \times R^{m_2 xL} \times \ldots \times R^{m_N xL}$ and $\underset{\sim}{y} = 0 \in R^{pxL} = R^{p_1 xL} \times R^{p_2 xL} \times \ldots \times R^{p_N xL}$.

Step 1: Rename the subsystems of S so that the natural total ordering $\{S_1, S_2, \ldots, S_N\}$ fits with the partial ordering of the subsystems of S induced by the removal of E in G_S^T.

Step 2: Compute $\underset{\sim}{u}_k(L)$ as a function of $\underset{\sim}{y}$, $\underset{\sim}{u}_o(L)$ and $\hat{\underset{\sim}{z}}_E(L)$. Set $\underset{\sim}{u}_k = \underset{\sim}{u}_k(L)$.

Step 3: Call $S_k = (f_k, g_k; \bar{x}_k)$ and set $\underset{\sim}{y}_k = \pi_{kL}(\underset{\sim}{u}_k, x_k)$

Step 4: If $k = L$, compute $\underset{\sim}{y}_o(L)$ and/or $\underset{\sim}{z}_E(L)$ and stop. Else, set $k = k + 1$ and go to step 2.

A natural way to see whether the extreme simplicity of Algo F^2 can be exploited by carrying the solution of $P_o(L)$ into the possibly repeated solution of $P_E(L)$, at suitably chosen values of the "feedback data" $\hat{\underset{\sim}{z}}_E(L)$, is provided by the so called Feedback Information Tearing Principle |4|. Computationally, it amounts (see Fig. 4) to devise some mechanism to cleverly guess the value of $\hat{\underset{\sim}{z}}_E(L)$ in such a way that, eventually, the corresponding value of $\underset{\sim}{z}_E(L)$ is equal to $\hat{\underset{\sim}{z}}_E(L)$. If $\underset{\sim}{z}_E(L)$ is simply taken as next value for $\hat{\underset{\sim}{z}}_E(L)$, a relaxation algorithm is obtained, for which some convergence condition must in general be provided. As far as the simulation of dynamical systems is concerned, such a technique has been first proposed in |15| under the name of waveform relaxation. Within the context of the present paper, it is expedient to use it inorder to make the subsystemwise simulation (over multisteps time-intervals) of an interconnected system prossible. The extremely simple algorithm is as follows.

Algo B1

Data : $T \sim (\bar{t}, L)$, $\bar{x}$, $\underset{\sim}{u}_o(L)$.

Step 0 : Set $k = 1$ and $\hat{\underset{\sim}{z}}_E(L) = 0$.

Step 1 : Compute $\underset{\sim}{y}_o(L)$ and $\underset{\sim}{z}_E(L)$ by Algo F^2.

Step 2 : If $\underset{\sim}{z}_E(L) = \hat{\underset{\sim}{z}}_E(L)$, stop. Else, set $\hat{\underset{\sim}{z}}_E(L) = \underset{\sim}{z}_E(L)$, $k = k + 1$ and go to step 1.

Theorem 3

If S is structurallv well-posed, then Algo B1 stops in at most L iterations and the last value of $\underset{\sim}{y}_o(L)$ is the solution of $P_o(L)$.

Proof. Denote by S_E the composite dynamical system underneath problem $P_E(L)$. Of course, S_E plus a feedback identity is an equivalent representation of S. If S is structurally well posed, then S_E must be purely dynamic from $\hat{z}_E$ to z_E. This means that, for any $\lambda = 1,2,\ldots, L$, $\underset{\sim}{z}_E(\lambda)$ is uniquely determined by $\bar{x}$, $\underset{\sim}{u}_o(\lambda)$ and $\hat{\underset{\sim}{z}}_E(\lambda-1)$. Hence, after k iterations, the first k values of $\hat{z}_E$ are permanently set to their final value; and, of course, $\underset{\sim}{z}_E(k) = \hat{\underset{\sim}{z}}_E(k)$. Thus, after at most L iterations, the algorithm must stop. But $\underset{\sim}{z}_E(L) = \hat{\underset{\sim}{z}}_E(L)$ implies (Fig. 3) that any solution of $P_E(L)$ is a solution of $P_o(L)$ as well, so the theorem is proved.

Following the proof of Theorem 3, one readily realizes that Algo B1 can be significantly improved by shortening to k the length of the simulation of each subsystem at the k-th iteration. This rules out the remote (probability zero) possibility that the algorithm stops after less than L iterations. Precisely, the modified algorithm is as follows.

Algo B2

Data : $T \sim (\bar{t}, L)$, $\bar{x}$, $\underset{\sim}{u}_o(L)$.

Step 0 : Set $\lambda = 1$, $\hat{\underset{\sim}{z}}_E(L) = 0$.

Step 1 : If $\lambda = L$, go to step 4. Else, go to step 2.

Step 2 : Compute $\underset{\sim}{z}_E(\lambda)$ by Algo F^2.

Step 3 : Set $\hat{\underset{\sim}{z}}_E(\lambda) = \underset{\sim}{z}_E(\lambda)$, $\lambda = \lambda + 1$ and go to step 1.

Step 4 : Compute $\underset{\sim}{y}_o(L)$ by Algo F^2 and stop.

To solve $P_o(L)$, both Algo B1 (with probability 1) and Algo B2 (for sure) require that each subsystem of S is moved L times in and out the central computing unit. Since the length of each simulation is here generally greater than one, even neglecting the extra computational effort needed to find an optimal (or at least a good) essential set of G_S^T, these algorithms cannot prove better than Algo A1 or Algo A2, in any practical situation.

There is, though, at least one interesting special case in which the feedback information tearing principle combined with a loop algorithmic function identification strategy |3|, |4|, when applied to the subsystemwise simulation problem, lead to an algorithm which may compare favorably with all the preceding ones. The case occurs when S is both linear and time-invariant. The remainder of the present section deals with such a case only.

If all the subsystems of S and the interconnection I are linear and time-invariant, the interconnected (but loop-free) system S_E underneath Problem $P_E(L)$ is linear and time-invariant as well. This means that there exist a matrix $M \in R^{(\nu \times L)^2}$ and a vector $z \in R^{(\nu \times L)}$ such that:

$$\underset{\sim}{z}_E(L) = M \hat{\underset{\sim}{z}}_E(L) + z.$$

Of course, both M and z, generally depend upon $\overline{x}$ and $\underset{\sim}{u}_o(L)$ and are a priori unknown. However, note that causality, linearity and time-invariance of S enable to conclude that M must be of the following form :

$$M = \begin{vmatrix} H(o) & O & O & \cdots & O \\ H(1) & H(o) & O & \cdots & O \\ H(2) & H(1) & H(o) & \cdots & O \\ \vdots & \vdots & \vdots & & \vdots \\ H(L) & H(L-1) & H(L-2) & \cdots & H(O) \end{vmatrix}$$

where H(.) is the impulse response matrix of S_E, from the input z_E to the output z_E. Furthermore, S is well-posed if and only if no eigenvalue of H(O) is equal to 1 |16, IV-6|. If S is structurally well-posed then S_E must be purely dynamic, whereby H(O) = O.

We can now solve $P_o(L)$ in two steps as follows. First, we need to identify M and z by a number of experiments. Each experiment consists in actually solving $P_E(L)$ for a given value of its data. As already noticed, this can easily be done subsystemwise, thanks to the loop-free structure of S_E. Of course, the experiments we need to perform amount to computing H(.) on T and z. This, in turn, calls for the computation, by Algo F^2, of ν single-input impulse responses plus $z_E(.)$ on T when $z_E(.) = O$. In summary, M and z can be identified by simulating $\nu + 1$ times on T each subsystem of S (recall that ν is, by definition, the dimension of z_E; since G_S^T is, by assumption, strongly connected, the minimum number of times each subsystem must be moved into the central computing unit to complete the identification process is then $\nu^o + 1$, where ν^o is the generalized index of G_S^T). As a second step, the congruence equation $\underset{\sim}{z}^o(L) = M\underset{\sim}{z}^o(L) + z$ must be solved for $\underset{\sim}{z}^o(L)$, and the corresponding value of

$\underset{\sim}{y}_o(L)$ must be computed. Again, thanks to the special structure of M, this can be done recursively in a very simple way. In fact, let $\underset{\sim}{\bar{z}}(L)$ be the value of $\underset{\sim}{z}_E(L)$ corresponding to $\underset{\sim}{\hat{z}}_E(L) = 0$. Then, the congruent value $\underset{\sim}{z}^o(L)$ of $\underset{\sim}{\hat{z}}_E(L)$ can be computed by repeatedly solving, for different values of b, a set of ν linear algebraic equation of the form Ax=b. Precisely, the recursion is as follows :

$$(I-H(o))\ z^o(\bar{t}) = \bar{z}(t)$$

$$(I-H(o))\ z^o(t) = \bar{z} + \sum_{\lambda=o}^{t-\bar{t}-1} H(t-\bar{t}-\lambda) z^o(\bar{t}+\lambda), \qquad \forall t \in T - \{\bar{t}\}.$$

Recall that I-H(o) is nonsingular if and only if S is well posed.
The simplest way to compute $\underset{\sim}{y}_o(L)$, once $\underset{\sim}{z}^o(L)$ is known, is to make use once more of Algo F^2. Alternatively, one might record the zero-input response and the impulse response matrix from $\hat{z}_E$ to y_o, during the identification process, thus making the direct output computation possible.

All the preceding discussion can be summarized by stating the following theorem, the formal proof of which is obviously omitted.

Theorem 4

If the interconnected system S is well-posed, linear and time-invariant, then the subsystemwise simulation problem $P_o(L)$ can be solved by a two phases procedure of the following type:

Phase 1 : The strongly connected components $\{C_1, C_2, \ldots, C_q\}$ of S are determined, by analyzing the time-invariant system graph G_S.

Phase 2 : The simulation of each strongly connected component C_h of S is carried out sequentially by : a) simulating, at most $\nu_h^o + 1$ times, each subsystem of S belonging to C_h (canonical feedback loop identification); b) computing, by a simple recursion, the congruent

value of the feedback variables; c) computing the output of C_h over T either directly or by one more simulation of its subsystems.

Example 4

Consider the interconnected system S introduced in the second part of Example 3 (Fig. 3). Let $m_1 = p_2 = 2$, $m_2 = p_1 = 1$ and assume that :

i) S is well posed ,

ii) S_1 and S_2 are linear and time-invariant

iii) $\Gamma_{21}(t) = 1$, $\Gamma_{22}(t) = |0 \quad 1|$, $\forall t$,

(iv) $\sigma(y(t), u_o(t), t) = |1 \quad 0 \,|y_2(t)$, $\forall t$.

Then, G_S is strongly connected (see Fig. 3), $\nu^o = 1$ and $E = \{u_2\}$ is an optimum essential set of G_S. The system S_E underneath $P_E(L)$ is shown in Fig. 5. Since $\nu^o = 1$ and an optimum essential set has been found, only two experiments ($\hat{z}_E(.)=0$, $\hat{z}_E(.)=\delta(.)$; $\delta(L) \triangleq |1\ 0\ 0\ .\ .\ .\ 0|'$, $\forall L \geq 1$) are needed to identify S_E. Both $H(.)$ and $\bar{z}(.)$ are scalar functions and $H(0) \neq 1$, in view of (i). Hence, the recursion for the congruent value $z^o(.)$ of the feedback variable is straightforward.

Remark 2

In view of Theorem 4, the maximum number of times each subsystem has to be called in the central computing unit to run a "local" simulation) is $\mu^o + 2$, where μ^o is the interconnectivity index of G_S, namely the generalized index of the most strongly connected component of G_S.

Remark 3

If i) the computational effort needed to simulate a system over a given interval of time is proportional to the square of its order, ii) all the sub-

systems are approximatively of the same order, iii) G_S is strongly connected and iv) $v^o << N$, then the decomposition technique described in Theorem 4 may compare favorably, from a computational point of view, with more standard techniques, even if the computer at hand is large enough as to make the simulation of the overall system S possible, in a single shot.

Remark 4

If the subsystemwise simulation problem has to be solved many times, for different values of $\bar{x}$ and $u_o(.)$, then a substantial saving in the computations involved by the method of Theorem 4 may be achieved by exploiting the fact that H(.) is not going to change; it does not need, then, to be identified more than once (see step (a) of phase 2 in Theorem 4).

4. CONCLUSIONS

The problem of simulating piece-by-piece a large-scale interconnected discrete-time dynamical system has been considered in this paper (subsystemwise simulation problem).

Two basic lines of attack have been considered, according as whether a decomposition is sought at each time step or over the entire simulation interval. In each case, specific algorithms have been described and discussed. In particular, their convergence to the exact solution after a finite number of iterations has been proved.

In the paper, enphasis has been placed on the possibility of serializing

a large problem by recasting it as a sequence of smaller subproblems. It has to be stressed however that the proposed analysis is also suited to deal in a quite natural way with some of the basic issues raised by the possibility of speeding up the simulation process by parallel computing. Such a possibility does in fact occur whenever the analysis leads to an ordering of the subproblems, by the data-solution precedence relation, which is a partial ordering in a proper sense.

In particular, in Theorem 4 the partial ordering of the strongly connected components is uniquely determined by the system structure, but the partial ordering of the subsystems within each strongly connected component strongly depends on the choice of the essential set E. Hence, whenever the possibility of using parallel computing is considered, it might be sensible to look for an essential set which is optimum with respect to a more complex performance index, incorporating not only the amount of information carried by the feedback identity but also some index of the degree of parallelism associated with E.

Finally, it is worth mentioning that the same circle of ideas underlying Theorem 4 can easily be used to construct Class A algorithms to solve, in a finite number of iterations, the subsystemwise simulation problem for any linear well-posed interconnected system S, even if it is not structurally well-posed in the sense of Section 2.

Acknowledgment . Thanks are due to Alberto Sangiovanni-Vincentelli for stimulating discussions on the subject of section 3 and to Pravin Varaiya for useful comments on a draft of the manuscript.

REFERENCES

|1| M. Vidyasagar: "On the Well-Posedness of Large-Scale Interconnected Systems", IEEE Trans. on Automatic Control, Vol. AC-25, No.3, June 1980, pp. 413-421.

|2| G. Guardabassi: "Information Structures in Problem Analysis", Proc. Int. Conf. on Information Sciences and Systems, D.G. Lainiotis and N.S. Tzannes Eds., Hamisphere Publ. Corp., Washington D.C. 1977.

|3| G. Guardabassi : "Large-Scale Problem Analysis and Decomposition Theory", J. of The Franklin Institute, Vol. 306, No. 1, July 1978, pp. 41-62.

|4| G. Guardabassi: "Composite Problem Analysis", to appear in Large-Scale Systems, 1982

|5| A.K. Kevorkian, J. Snoek: "Decomposition in Large-Scale Systems: Theory and Application of Structural Analysis in Partitioning, Disjointing and Constructing Hyerarchical Systems", in "Decomposition of Large-Scale Systems", D.M. Himmelblau Ed., North-Holland, Amsterdam 1973.

|6| R. Tarjan: "Depth-First Search and Linear Graph Algorithms", SIAM J. Comp., Vol. 1, 1972, pp. 146-160.

|7| L.P.A.Robichaud, M. Boisvert, J. Robert: Signal Flow Graphs and Applications, Prentice-Hall, Englewood Cliffs, N.J., 1962.

|8| A. Lempel, I. Cederbaum: "Minimum Feedback Arc and Vertex Sets of a Directed Graph", IEEE Trans. Circuit Theory, Vol. CT-13, pp. 339-403, 1966.

|9| L. Divieti, A. Grasselli: "On the Determination of Minimum Feedback Arc and Vertex Sets", IEEE Trans. Circuit Theory, Vol. CT-15, pp.86-88, 1968.

|10| G. Guardabassi: "A Note on Minimal Essential Sets", IEEE Trans. Circuit Theory, Vol. CT-18, pp.557-560, 1971.

|11| M. Diaz, J.P. Richard, M. Courvoisier: "A Note on Minimal and Quasi-Minimal Essential Sets in Complex Directed Graphs", IEEE Trans. Circuit Theory, Vol. VT-19, pp. 512-513, 1972.

|12| G. Guardabassi: "An Indirect Method for Minimal Essential Sets", IEEE Trans. Circuits and Systems, Vol. CAS-21, pp. 14-17, 1974.

|13| L.K. Cheung, E.S. Kuh: "The Bordered Triangular Matrix and Minimum Essential Sets of a Digraph", IEEE Trans. Circuit and Systems, Vol. CAS-21, pp. 633-639, 1974.

|14| G. Guardabassi, A. Sangiovanni-Vincentelli: "A Two-levels Algorithm for Tearing", IEEE Trans. Circuits and Systems, Vol. CAS-23, pp.783-791, 1976.

|15| E. Lelarasmee, A.E. Ruehli, A.L. Sangiovanni-Vincentelli: "The Waveform Relaxation Method for Time-Domain Analysis of Large-Scale Integrated Circuits", University of california, Berkeley, Electronics Research Laboratory, ERL M81/75, September 1981.

|16| C.A. Desoer, M. Vidyasagar: "Feedback Systems: Input-Output Properties", New York: Academic, 1975.

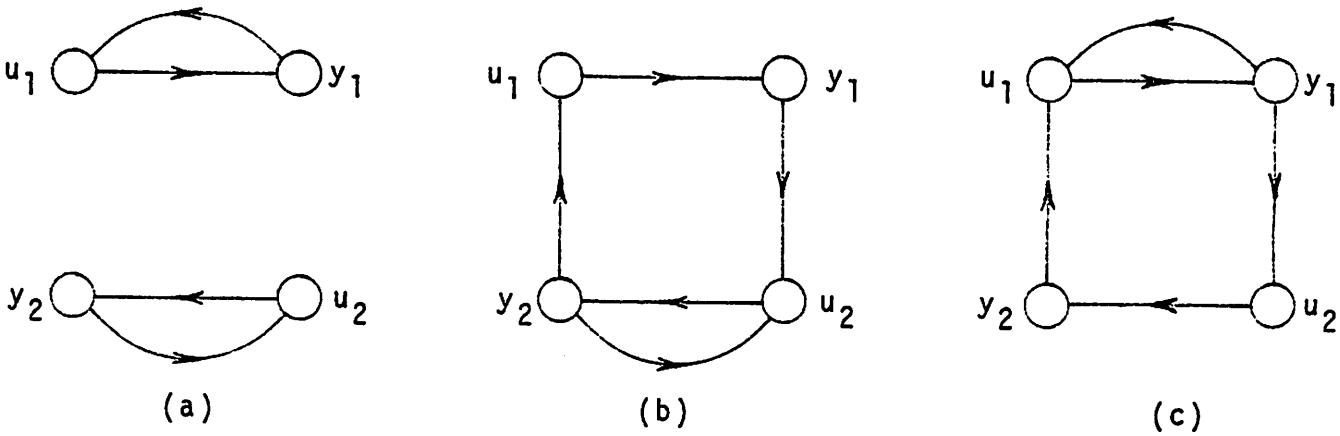
u_1
y_1
y_2
u_2
(a)
u_1
y_1
y_2
u_2
(b)
u_1
y_1
y_2
u_2
(c)

Fig. 1

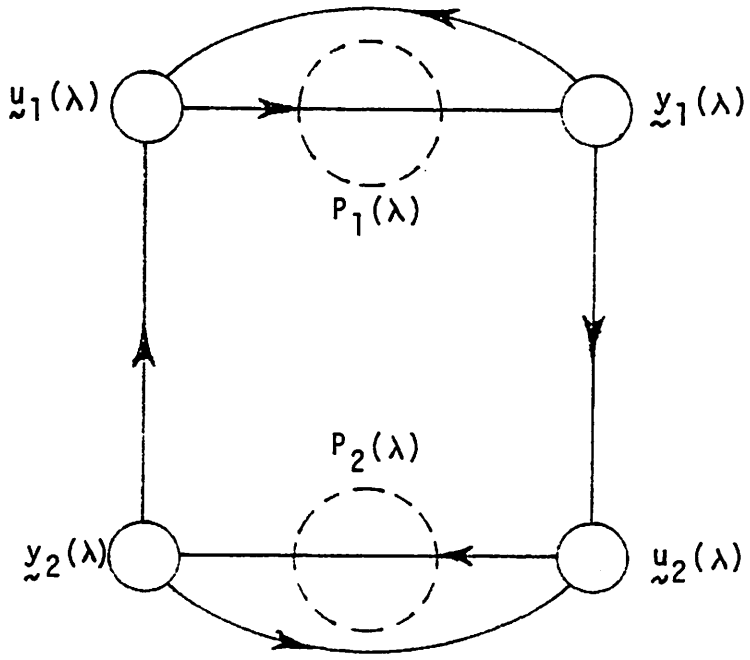
$\underset{\sim}{u}_1(\lambda)$
$\underset{\sim}{y}_1(\lambda)$
$P_1(\lambda)$
$P_2(\lambda)$
$\underset{\sim}{y}_2(\lambda)$
$\underset{\sim}{u}_2(\lambda)$

Fig. 2

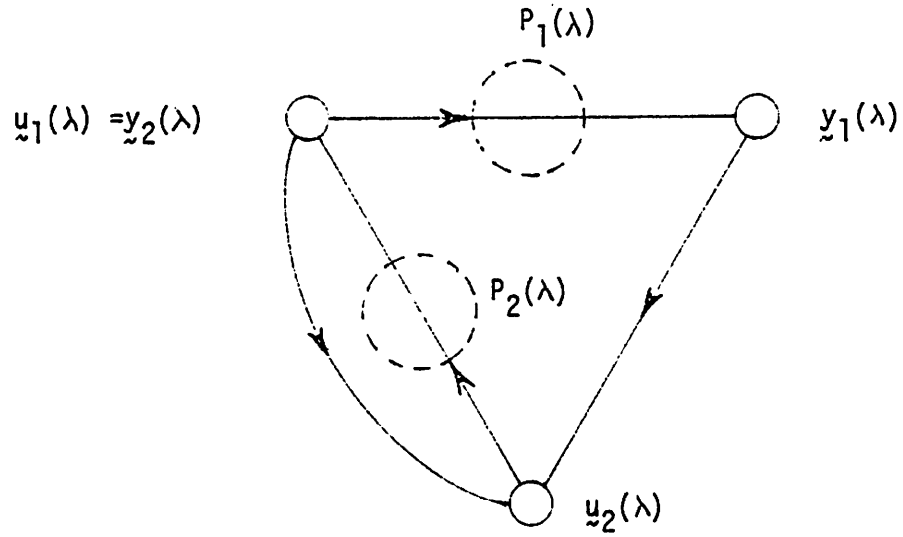

Fig. 3

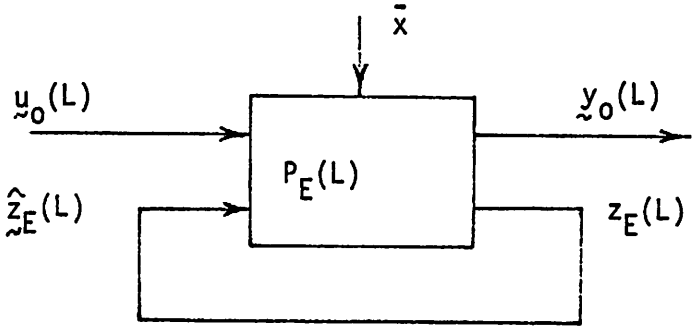

Fig. 4

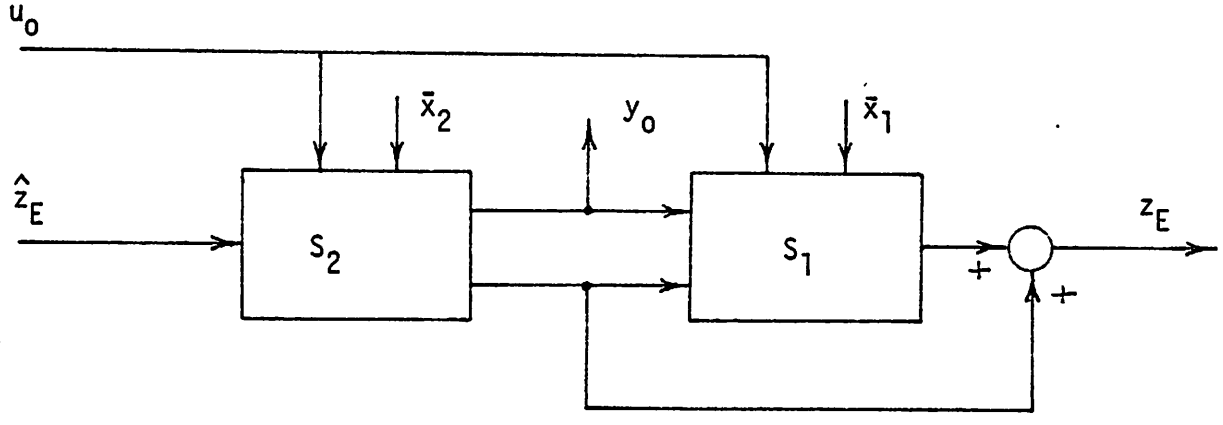

Fig. 5