Analyzing Piecewise Linear Dynamical Systems

Njal B.O.L. Pettit and Peter E. Wellstead

Piecewise linear (PL) systems are an attractive class of nonlinear systems to study, as they straddle a number of difficult and topical control problems. It is not a new area by any means; for a long time piecewise linear functions have been the standard technique adopted by engineers to represent a range of system nonlinearities, such as dead zones, saturation, relays, and hysteresis. Indeed, stability properties of system components, especially actuators which are piecewise linear, have been studied for decades [1,2]. However, in recent times engineers have started to employ control laws that are piecewise linear in nature. Important examples are rule-based control, gain scheduling, and programmable logic control. Rule-based control in dynamic processes in particular fall into some definitions of what has been termed intelligent control, an area frequently discussed in this magazine [3,4] and often with the view that some analysis technique is needed for these systems. There has also been a recent interest in what has been termed hybrid systems [5]. Although this term has been used for a wide range of systems, from timed finite state automation to complete integrated factory control and scheduling problems, some definitions used would encompass the piecewise linear systems outlined here, for example [6]. In this spirit, the work can be seen as an approach for analyzing classes of hybrid systems. This article presents a method that is felt to provide a possible answer to the problem of analyzing mixed logic/dynamic systems.

The article describes the development of a computational tool for the analysis of PL dynamical systems discussed above. Note that we exclude situations where controllers are based on firm analysis (i.e., sliding mode control), since then they function in a predictable manner. Unfortunately, many PL controllers are developed from ad hoc "intelligent systems" ideas which do not aim or allow the associated dynamic behavior to be predicted. An example of such a system and a strong motivation for this work is the ABS (anti-skid braking) system in a car, where the controller is rule-based and designed using the engineer's knowledge of the system. The only current viable approach to testing such a system is by using extensive simulation and prototype testing, which must be repeated for each of the different car models on which it is installed. Anything that provides insight into the logic and dynamic interaction of such a system would be useful; hence the development of the work in this article. Similarly, systems with programmable logic controllers and gain schedulers also fall into the class of piecewise linear systems, providing further incentive for studying this type of system.

The novel aspect of this work is to take ideas and known results from linear systems, convex set theory, and computational geometry and to synthesize them to create an analysis tool for studying a class of systems that mix logic and dynamics. We choose to develop a computational analysis tool primarily because the traditional theoretical analysis of piecewise linear processes is intractable, except, that is, for certain local dynamic behavior.

A version of this article appeared in the 1994 IEEE 33rd CDC. *The authors are with the Control Systems Centre, Department E.E. & E., University of Manchester Institute of Science and Technology (UMIST), P.O. Box 88, Manchester, M60 1QD, U.K. Email: pettit@csc.umist.ac.uk.*

A Geometric Perspective

The attractions of piecewise linear (PL) systems in control have been recognized for a long time [7,8], and a standard description of such systems is usually used. In particular, in its simplest form a PL system is described as a set of convex polytopes $\mathbf{P_i} \in \Re^n$ each containing some linear system of the form

$$\dot{\underline{x}} = \underline{A}_m \underline{x} + \underline{b}_m, \quad \underline{x} \in \mathbf{P_m} \tag{1}$$

where the $\mathbf{P_i}$ form a partition of $\Re^n$ such that

$$\cup \mathbf{P_i} = \Re^n, \ \mathbf{P_i} \cap \mathbf{P_j} = \varnothing, \ i \neq j \tag{2}$$

The problem has been that the geometric interpretation leads to a complex picture of "boxes" stacked together in state space with each box containing a different linear dynamic system. Any global analysis must somehow identify the behaviors in each box and then link them together to form a global picture of the dynamics. Fig. 1 illustrates this geometric interpretation.

In Fig. 1 a block diagram shows a three-state system with two PL functions—a saturation followed by a relay. In state space, the system will be in three dimensions. One axis will be split by two planes due to the two breakpoints that appear in the saturation. The other will be split by one plane due to the relay, since although the relay has two breakpoints, they occur at the same instance in the input. As a result, the state space will comprise six linear regions. Fig. 1 shows how the PL functions of the system result in switching surfaces in the state space. These surfaces act as the boundaries of the convex polytopes that contain each linear dynamic region. The difficulties presented in analyzing this setup are bound up in the need to manipulate high dimensional convex polytopes and the dynamic systems within them. One analysis technique, that using the phase portrait, fulfills many of the analysis aims. In the phase portrait PL functions can be represented as lines in the plane and trajectories or isoclines plotted to represent the dynamics. The result is a

 0272-1708/95/$04.00©1995IEEE

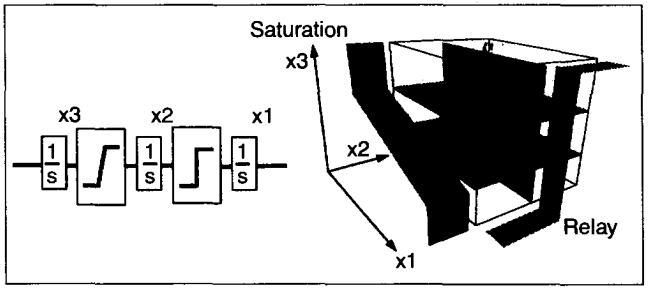


Fig. 1. The geometric interpretation of a piecewise linear system.

graphical plot of the system dynamics that gives global stability information and shows how the dynamic patterns change due to the switching lines and hence PL functions. The big drawback is the limitation of the phase portrait to two states.

The System Stability Graph

In [9] the idea of mapping a piecewise linear system into a connected graph was proposed, the idea being based on the phase portrait. Each convex polytope or region in the state space will have dynamics entering and exiting that region. If the boundaries of every region were partitioned into sections containing only dynamics entering a region (termed an *Nface*) and only dynamics exiting a region (termed an *Xface*) then the boundaries can be characterized into sections of homogeneous dynamic behavior. Each section thus identified is then represented as a node of a graph. The connections between nodes are then characterized by tracking the set of trajectories (or *trajectory bundle*) entering via some Nface and identifying which (if any) Xfaces the trajectory bundle leaves that region. This idea is illustrated in Fig. 2.

Piecing together the nodes and connections for each region results in a directed graph that captures the global dynamic patterns of the system. The nodes of the graph represent the PL functions and the directed connections represent the interaction of the PL functions with the system's dynamics. As will be explained in subsequent sections, the realization of this apparently simple idea is not easy.

The Problem Formulation

The problem divides into three main distinct tasks with the link being a consistent and compatible data flow between each task.

1. System Representation. The system model must be translated into a description that contains all the information about the linear regions and switching surfaces that separate them. This information needs to be in some compact form that allows easy generation of any particular linear system and its associated switching boundaries. Explicit storage of every region and its boundaries, together with the relationship between boundaries, leads to problems of data explosion.

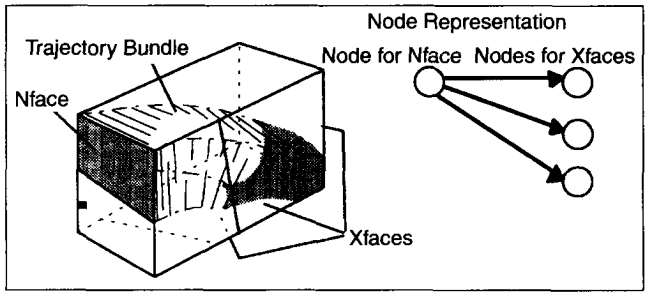


Fig. 2. Forming node connections using trajectory bundles.

2. Node Identification. To identify the nodes, the convex polytopes containing the linear systems must be explicitly formed and the boundaries partitioned into Nfaces and Xfaces. This requires information about the linear system in a region, the switching surfaces bounding that linear system, and the linear systems adjacent to the region of interest. This data should be accessible from the system representation described in the first task to allow continuity.

3. Node Connection. To connect the nodes, once the Nfaces and Xfaces of a region are identified, this information must be combined with the linear system dynamics to allow the trajectory bundles to be formed and tracked. Data continuity is achieved by using the geometric information concerning the partitioned boundaries as a key element in the definition and tracking of the trajectory bundles.

System Representation

The most widely understood system representation method is the block diagram approach. As such the ideal PL system representation would be one that can be directly derived from a block diagram of the system with PL functions. To achieve this, a graphical user interface was selected as the input stage of the analysis process; in this case SIMULINK was adopted. Some work has appeared in the circuit theory literature on representing PL systems in a compact framework ([10-12]) but a modified approach was needed to allow SIMULINK to be used as a front end, although the resulting method has some similarities with [11]. The PL representation devised is essentially composed of three system equations:

$$\left.\begin{aligned} \underline{n} &= \underline{R}^{\underline{i}}\,\underline{m} + \underline{k}^{\underline{i}} \quad (a) \\ \underline{m} &= \underline{F}\underline{x} + \underline{G}\underline{n} + \underline{h}\underline{u} \quad (b) \\ \dot{\underline{x}} &= \underline{J}\underline{x} + \underline{K}\underline{n} + \underline{l}\underline{u} \quad (c) \end{aligned}\right\} \tag{3}$$

where in 3(a), $\underline{n}$ represents a vector of outputs from PL functions, one output for each function, and $\underline{m}$ is the vector of inputs to those PL functions. At present only single-valued PL functions that operate over the entire range of the input are considered (e.g., saturation, dead zones, relays, quantization). Extending the representation to include all key logic type actions and multi-valued PL functions such as hysteresis is still a topic of research. In 3(b), the inputs of the PL functions $\underline{m}$ are defined to comprise a linear combination of states $\underline{x}$, outputs from other PL functions $\underline{n}$, and any system constants u. As PL functions can interact, $\underline{G}$ can be seen as an interaction matrix for the PL functions. The dynamics of the system are embedded in 3(c), a combination of the states $\underline{x}$, PL function outputs $\underline{n}$ and system constant u. The key to extracting each linear system and associated bounding switching surfaces is in the vector $\underline{i}$ in 3(a). Each PL function is composed of a number of linear segments. The vector $\underline{i}$ indicates which linear segments of the different PL functions is "active." Thus $\underline{R}^{\underline{i}}, \underline{k}^{\underline{i}}$ define the active gradients and offsets of the different PL functions. Incrementing or decrementing any element of $\underline{i}$ moves along one PL function to its next segment and hence

moves to an adjacent region. Using this, the explicit equations for a particular region are given by

$$\underline{\gamma}^i = \begin{bmatrix} -\left(\underline{I} - \underline{GR}^i\right)^{-1} \underline{F} \\ \left(\underline{I} - \underline{GR}^i\right)^{-1} \underline{F} \end{bmatrix}$$

$$\underline{c}^i = \begin{bmatrix} -\left(\underline{I} - \underline{GR}^i\right)^{-1} \left(\underline{Gk}^i + \underline{hu}\right) + \underline{BX}^{i+1} \\ \left(\underline{I} - \underline{GR}^i\right)^{-1} \left(\underline{Gk}^i + \underline{hu}\right) - \underline{BX}^i \end{bmatrix}$$

$$\underline{A}^i = \underline{J} + \underline{K}\left[\underline{I} - \underline{R}^i \underline{G}\right]^{-1} \underline{R}^i \underline{F}$$

$$\underline{B}^i = \underline{l} + \underline{K}\left[\underline{I} - \underline{R}^i \underline{G}\right]^{-1} \underline{R}^i \underline{h}$$

$$\underline{Q}^i = \underline{K}\left[\underline{I} - \underline{R}^i \underline{G}\right]^{-1} \tag{4}$$

where the linear dynamic equation for that region is given by

$$\dot{\underline{x}} = \underline{A}^i \underline{x} + \underline{B}^i u + \underline{Q}^i \underline{k}^i \tag{5}$$

and the switching hyperplanes that bound the region are defined by

$$\underline{\gamma}^i \text{ and } \underline{c}^i \tag{6}$$

The advantage of this representation is that the complete system description is stored in six matrices and two vectors:

$$\underline{BX}, \underline{BY}, \underline{F}, \underline{G}, \underline{J}, \underline{K}, \underline{h}, \underline{l} \tag{7}$$

where $\underline{BX}$, $\underline{BY}$ are matrices that store the look up tables used to define the PL functions in SIMULINK from which $\underline{R}^i$, $\underline{k}^i$ are derived. The remaining matrices can be found by using the "linmod" function of SIMULINK as a system identifier. By redefining the system input and outputs for linmod, the remaining matrices can be found with two function calls. This method is detailed in [13].

Node Identification

In Fig. 1, the geometric interpretation of a PL system in state space was given. This was essentially comprised of adjacent convex polytopes with linear systems. This interpretation essentially divides the PL system into two parts: (a) a static description built up from the PL functions and forming the convex polytopes; and (b) a dynamic description that includes the linear dynamical systems in each region. This static description can be termed a *geometric model*, since it constitutes a geometric representation of each region that is needed to identify the nodes of the system. To obtain the geometric model of all or part of the PL system, a method of representing the geometry of multiple convex polytopes is needed together with all the relationships between the polytopes. These polytopes must then be manipulated so that their boundaries can be partitioned into Nfaces and Xfaces to identify the nodes.

Computer Representation

The data to build up a computer representation of the geometric model must come from the system representation in (3). This provides the switching hyperplane information together with the linear system dynamics. To solve this, work done in developing data structures for high dimensional (i.e., three dimensions and above) convex polytopes was used as the basic representational technique. This is essentially in the domain of computational geometry with a good survey of the field being given in [14]. The solution developed was to generate a data structure based on the topology of the partitions between the convex polytopes [15]. This means that the relationships between convex polytopes and relationships between the boundaries of any one convex polytope must be identified from the switching hyperplanes of the PL system. This information is then stored as complex set of links between objects representing the boundaries of each region. To illustrate the idea, a tetrahedron is represented in terms of the topology of its boundaries in Fig. 3. The tetrahedron comprises four faces, H1, H2, H3 and H4, which share common edges; for instance, H1 and H2 share edge e2 and H3 and H4 share e6. An alternative to representing the tetrahedron as a fixed geometric object in Euclidean space is to describe it as a set of connections between objects, in this case the objects being faces and edges. In such a way the boundary topology of the tetrahedron can be captured as a set of linked lists of objects as shown in Fig. 3. Since the tetrahedron can be viewed from "inside" and "outside" we get the separate structures as shown in Fig. 3.

Any manipulation of a particular boundary then becomes a well-defined set of operations on the topology to obtain the updated topology. The geometric information is hidden as references to the normals of each switching hyperplane and to the vertices generated at the intersections of the hyperplanes. The key algorithms that allow the relationships between boundaries to be found are convex hull algorithms that can identify convex hulls from vertices and sets of hyperplanes. The main algorithm used here is based on [16].

Boundary Partitioning

As described in the problem formulation, each region must be divided into Nfaces and Xfaces. To see how the Nfaces and

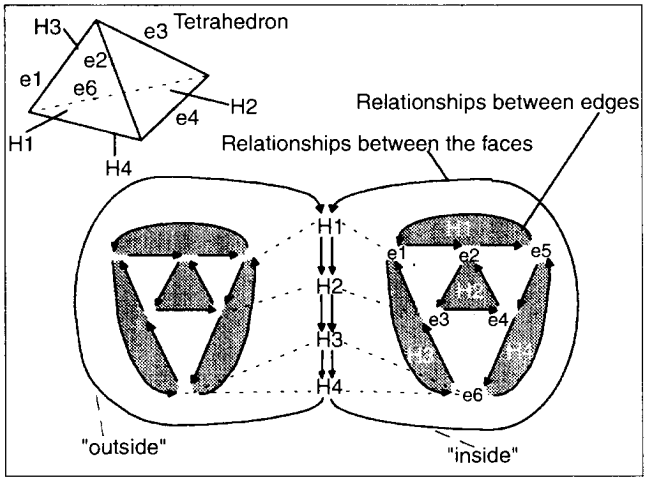


Fig. 3. An example of a geometric object represented as a topology based data structure.

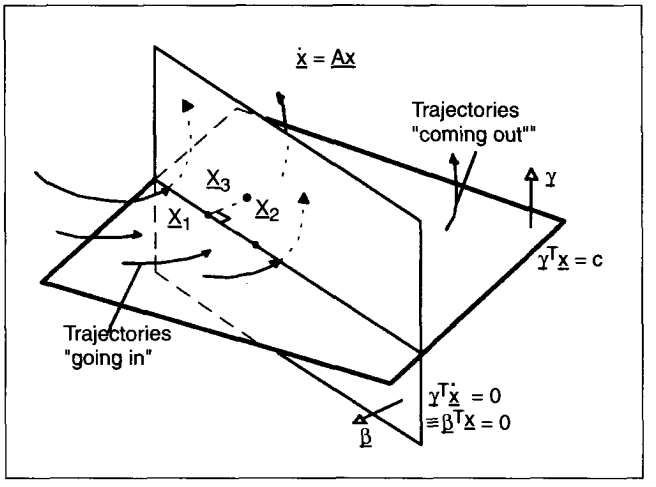


Fig. 4. Trajectories tangent to a hyperplane.

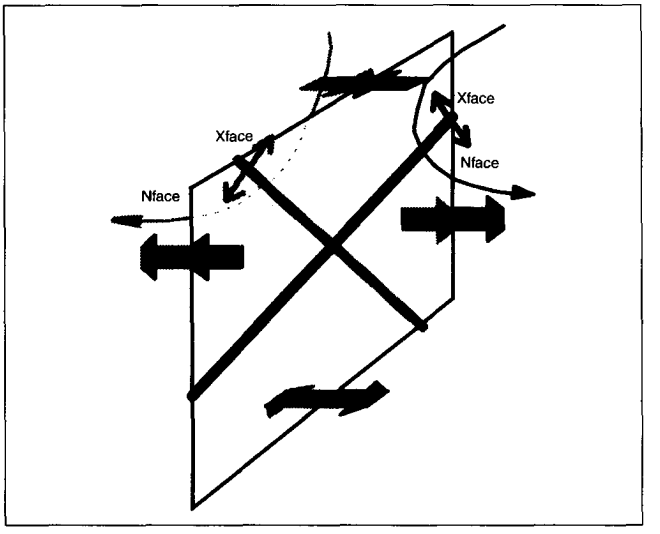


Fig. 5. The maximum number of trajectory patterns across a hyperplane.

Xfaces are defined, consider the situation of a linear system tangent to a boundary shown in Fig. 4.

Let the equation of a hyperplane be $\underline{\gamma}^T \underline{x} = c$ where $\underline{\gamma}$ and $\underline{x}$ are vectors and c is a constant such that $\underline{\gamma}, \underline{x} \in E^d, c \in E$. Let the linear system around the hyperplane be described by $\dot{\underline{x}}(t) = \underline{A}\underline{x}(t)$. Give two points $\underline{x}_1$ and $\underline{x}_2$ (see Fig. 4) that belong to $\underline{\gamma}^T \underline{x} = c$, then $\underline{x}_2$ - $\underline{x}_1$ defines a vector on the hyperplane; therefore,

$$\underline{\gamma}^T (\underline{x}_2 - \underline{x}_1) = c - c = 0 \tag{8}$$

that is $\underline{\gamma}$ is parallel to the hyperplane normal.

If $\underline{x}(t)$ is tangent to the hyperplane then $\dot{\underline{x}}(t)$ must be parallel to the hyperplane, as

$$\underline{\gamma}^T \dot{\underline{x}}(t) = 0$$

which gives

$$\underline{\gamma}^T (\underline{A}\underline{x}(t)) = (\underline{\gamma}^T \underline{A})\underline{x}(t) = 0 \tag{9}$$

Let $\underline{\beta}^T = \underline{\gamma}^T \underline{A}$. The trajectory gradient will thus be parallel to the hyperplane $\underline{\gamma}^T \underline{x} = c$ at a "tangent (hyper)plane" $\underline{\beta}^T \underline{x} = 0$.

Assign to a trajectory "going in" to the hyperplane the inequality $\underline{\gamma}^T \dot{\underline{x}}(t) < 0$. A trajectory "coming out" of the hyperplane will then have the inequality relationship $\underline{\gamma}^T \dot{\underline{x}}(t) > 0$. This is evident by comparing the gradient vector of $\underline{x}(t)$ when on the hyperplane with the normal of that hyperplane. Therefore the trajectories "going in" obey $\underline{\beta}^T \underline{x} < 0$ and the trajectories "coming out" obey $\underline{\beta}^T \underline{x} > 0$. Thus the dynamics define a tangent hyperplane that divides the switching hyperplane in two, one Nface and one Xface. This is simple to prove, with the proof given in [15]. The implication of this is that any hyperplane can be split into at most two sections, one Nface and one Xface. The corollary is that if a different linear system existed either side of the hyperplane, the hyperplane could be split into two sections on each side, resulting in a maximum of four sections across the boundary. This latter case is depicted in Fig. 5.

Thus any boundary of a region can be partitioned into a maximum of four nodes in the final System Stability Graph. As every Nface and Xface is found by partitioning a boundary of a convex polytope (i.e., the boundary will also be convex [17]) by one or more hyperplanes, all Nfaces and Xfaces will themselves be convex.

Node Connection

When the boundaries of a region have been partitioned into Nfaces and Xfaces, the geometric model is updated to accommodate the new information. The next stage is to identify the connections between nodes. The approach used exploits a simple but powerful property of linear systems. The dynamics of a linear system can be tracked from some initial point using the solution to a linear system equation of (1)

$$\underline{x}(t) = e^{\underline{A}(t)} \underline{x}_0 + \int_0^t e^{\underline{A}(\tau)} \underline{b} d\tau \tag{10}$$

Using $\underline{\Phi}(t)$ to represent the matrix exponential and $\underline{k}(t)$ the integral expression gives

$$\underline{x}(t) = \underline{\Phi}(t)\underline{x}_0 + \underline{k}(t) \tag{11}$$

where $\underline{\Phi}(t) = e^{\underline{A}t}$.

Now consider all the trajectories that have an Nface as their initial condition. The linear region the dynamics are entering defines the linear equation to use. The Nface is a convex polytope which can be defined by its vertices. If all points on the Nface are projected forward in time using (11) to some fixed time, then the new set of points will form another convex polytope and the

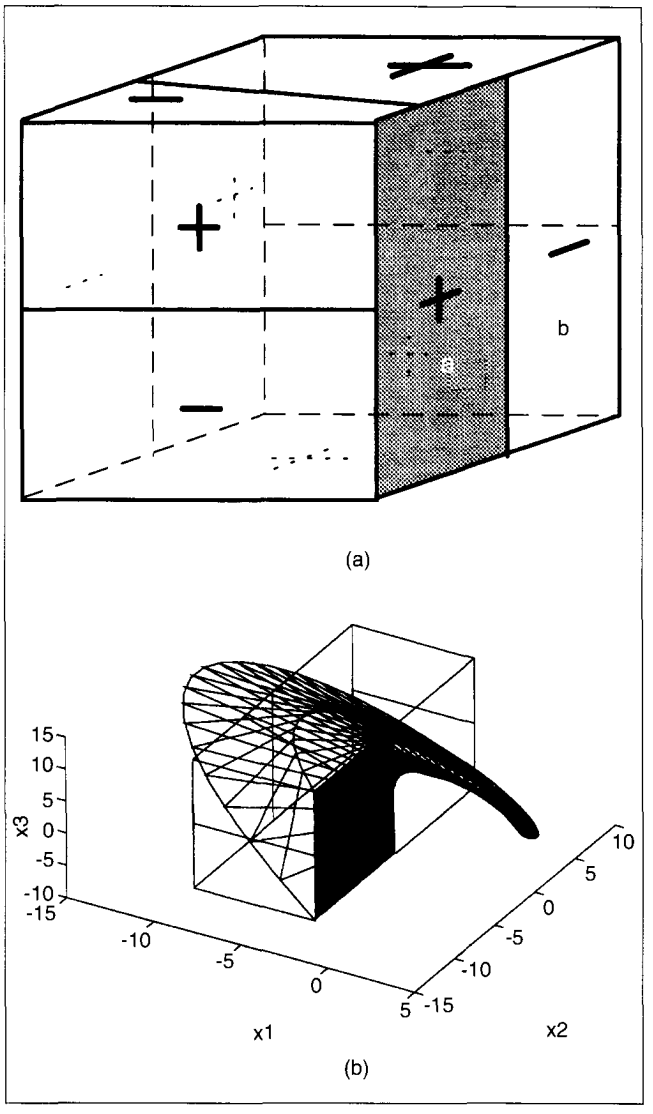


Fig. 6. (a) An example of a geometric model of a region. (b) Tracking a trajectory bundle through the region.

vertices defining this new convex polytope will be the projected vertices of the original Nface. This preservation of convexity is inherent in the linear system and is easy to show [18]. The advantage of the property is that only the vertices of an Nface need be tracked in order to follow where the trajectories from an Nface go. To illustrate this, Fig. 6(a) shows a visualization of a geometric model of one region. The "+" indicates Nfaces and "-" Xfaces. The trajectory bundle defined by the shaded Nface "a" is then tracked by projecting its vertices and forming the projected Nface at discrete time steps using the projected vertices. This is illustrated in Fig. 6(b), which shows the strong tendency for the dynamics from "a" to exit the region via the Xface "b", hence node "a" connects to node "b".

The Global System Stability Graph

Once the dynamics of each region have been mapped, each region will have its own connected graph of the dynamics in that region. The last step in the analysis procedure is to form a global

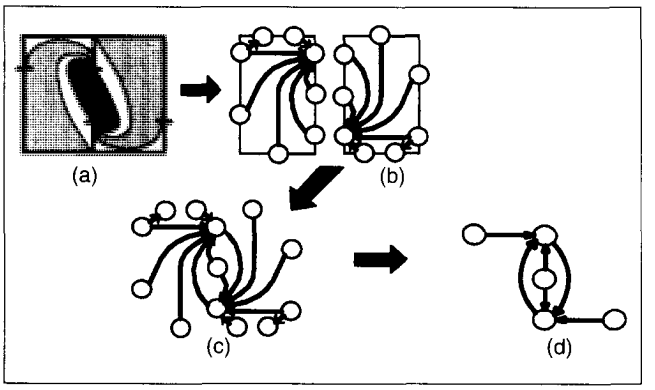


Fig. 7. Forming a global node graph. (a) The state space of a system. (b) The results of analyzing each region. (c) The complete system stability graph. (d) Simplifying the graph.

picture of the dynamics. This is done by connecting the graphs for each region together to form a global directed graph. To show this, Fig. 7 gives a stylized example of the analysis progression. Fig. 7(a) illustrates a two-state system where two linear regions are separated by a switch. The main patterns of dynamic movement in the system are shown with the shaded sets of trajectories. Fig. 7(b) illustrates how dividing up the boundaries according to tangent behavior generates the nodes in each region. The projection of the dynamics through the region are shown by the connections between nodes in the figure. In 7(c) the resulting graphs of the two regions are merged to give the global stability graph of the system. Finally, Fig. 7(d) demonstrates some graph simplification to clarify the representation of the system dynamics.

Bringing It All Together

The intractability of stability analysis of PL systems by pencil-and-paper analysis have led us to a computational solution which is based upon a system stability graph that reveals the stability patterns of the system. Note here that the fact that limit cycles can be identified as well as convergence points is important, since many PL control systems are designed to limit cycle (e.g., automotive anti-skid controllers).

We have described the sequence of techniques and procedures required to build the system stability graph for a piecewise linear system. The research sequence was not random, but specifically designed to yield the operational components of a computer-aided control system design (CACSD) tool. The overall operation of such a tool can be outlined given the principles behind the main sections of the analysis method, as discussed in this article. The CACSD tool requires a number of different routines, but it is envisioned that MATLAB and SIMULINK will provide a suitable front end to the system. This front end was selected because of its status as a de facto standard for control engineers.

Stage 1: Initial System Description. SIMULINK is used as a well-understood visual means of inputting a model of the system. This is already partly achieved, as outlined in the "System Representation" section. MATLAB is then used to manipulate the SIMULINK description to derive an analytic description of the PL system. An added advantage of this is that the SIMULINK model can be used to simulate parts of the system

that show up as unusual in the systems stability graph after the analysis.

Stage 2: Computer Representation and Node Identification. This stage requires involved data structures and data structure manipulation. As such it is outside the realm of MATLAB and must be done in specialist code. C++ provides the best environment, as it is ideally suited to the types of data structures used. Data is picked up from files stored by MATLAB that contain the description of the PL system. The geometric model is then generated, either in its entirety or partially depending on the size of the system being analyzed. Information from this is accessed via a C or C++ interface with MATLAB.

Stage 3: Node Identification. This is a local analysis problem and as such local data can be passed from the geometric model to MATLAB; and this information is then used to project individual trajectory bundles. More sophisticated integration routines are needed than those included with MATLAB, so some additional "mex" files are needed. The results of each node connection test can be passed back to the geometric model for storage.

Stage 4: The System Stability Graph. When all connections are found, the geometric model can pass an adjacency matrix back to MATLAB that describes the resulting SSG. This is then available for graphical analysis, something to which MATLAB is well suited. Recently a prototype graph analysis package has been developed that links a number of graph analysis routines together using MATLAB's GUI tools and provides a convenient graphics-driven analysis environment for directed, connected graphs. The graph analysis has been extended to allow information such as weights on each graph connection, so that connections can be allocated quantitative measures of importance in the dynamics of the system. This weight information can then be used as part of the graph analysis procedures.

To illustrate the analysis and how it evolves, a simple example is presented. This example is a three-state system with relay which generates chaotic dynamic behavior. The system is taken from [19] and is chosen since the dynamics are interesting, but the system is still simple enough to allow visualization of simulation in three dimensions. This allows comparison of the analysis results with simulation. The system model is described in SIMULINK, as shown in Fig. 8. The block "NL1" is a relay switching between ±1 at zero input.

From the graphic model, the matrices outlined in Equation (7) that represent the system are found as

$$\underline{BX} = [-10 \quad 0 \quad 0 \quad 10],\ \underline{BY} = [-1 \quad -1 \quad 1 \quad 1]$$

$$\underline{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ \underline{L} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},\ \underline{K} = \begin{bmatrix} 0 \\ 1 \\ 100 \end{bmatrix} \underline{G} = 0,\ \underline{H} = 0$$

$$\underline{J} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0.05 & 0 \\ 100 & 0 & -100 \end{bmatrix}$$

These are used by the computer to identify the regions and their associated linear system equations. The boundaries of the region can then be partitioned into Nfaces and Xfaces. Fig. 9(a) shows a visualization of the geometric model the computer constructs of the system. The central shaded boundary is due to the relay and is shown as being split into three partitions due to tangent trajectories. The other boundaries are "global" boundaries to limit the area of interest. Fig. 9(b) superimposes some simulated trajectories onto the geometric model, showing the chaotic region of attraction.

Once the projections are completed, a connected graph is generated. Fig. 10(a) shows the graph as seen in the prototype

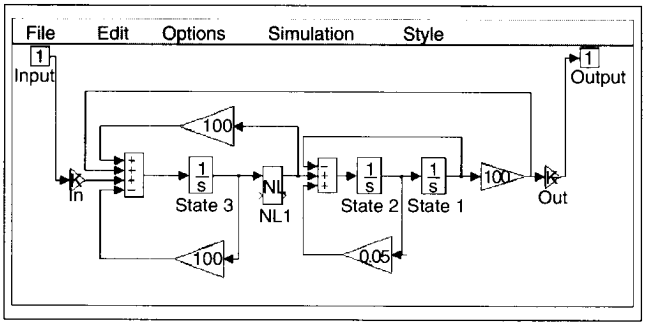


Fig. 8. A simple system with chaotic behavior.

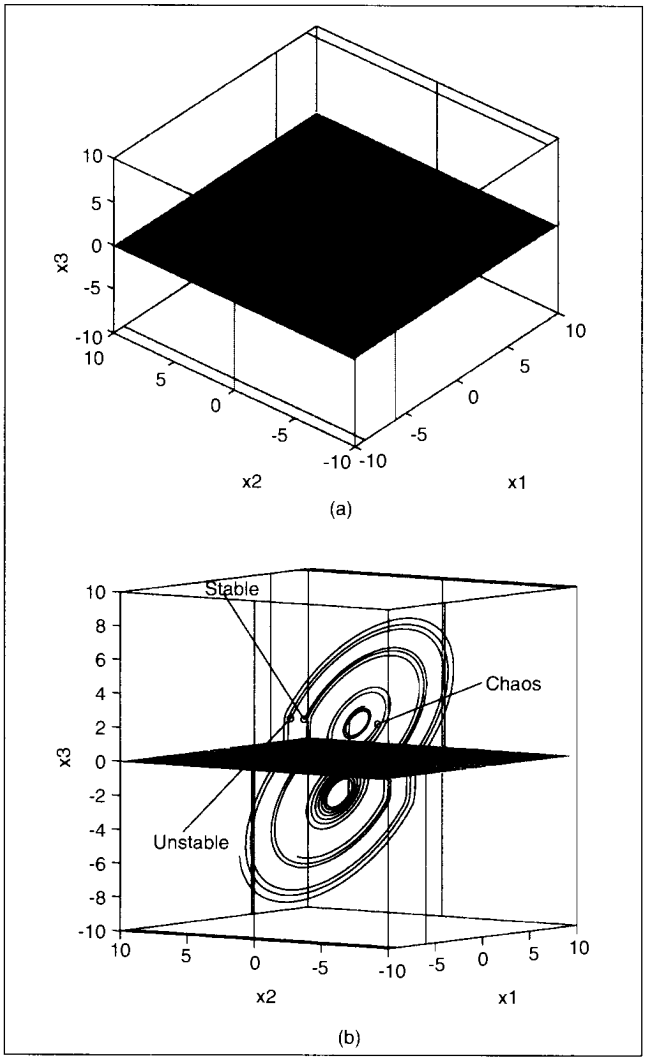


Fig. 9. (a) The geometric model generated from the chaotic system. (b) Some simulated dynamics.

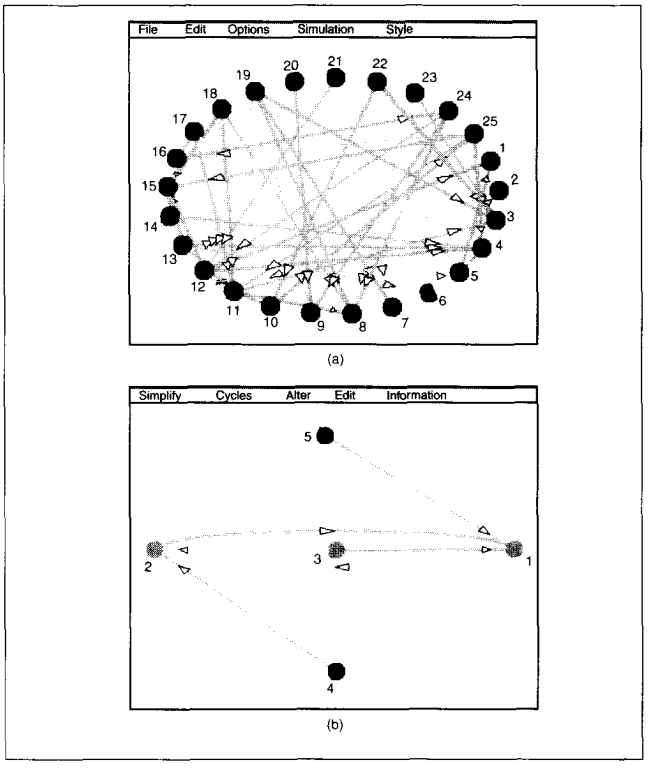


Fig. 10. (a) The complete system stability graph of the chaotic system. (b) A simplified graph.

graph analysis program. Most of the nodes are related to the global boundaries and confuse the picture. Simple graph manipulation allows only the nodes related to the relay and singular points to be visualized. This result is shown in Fig. 10(b). This latter figure shows a cycle around two of the relay nodes, with the third relay node feeding into the cycle. The singular points also feed into the cycle. If the nodes related to the global boundaries were examined, it would be seen that both the singular points and the three relay nodes have connections to the global boundaries. Although this stability graph does not explicitly state there is chaos, the essential dynamic behavior is captured and the combination of cyclic behavior in the dynamics with unstable singular points suggests some kind of complex dynamic behavior such as chaos. Fig. 10(b) can be compared to Fig. 9(b) to show the graph capturing the simulated dynamics of the system.

The software used here is a prototype aimed at proving the viability of the stages for the CACSD tool outlined and misses many features needed for the full package. Some analysis of the software has been carried out and reported in [20]. Summarizing the software work to date, the approach for the software used here is limited to systems with around five states and associated piecewise linear functions given a standard workstation such as a Sun Sparc 2. However, as outlined in [20], it should be possible to reduce memory requirements by some orders of magnitude, allowing systems of 20 or more states to be analyzed. The penalty paid by increased computation time is uncertain at present.

What the SSG Gives Us

The whole analysis procedure can be viewed as a mapping from piecewise linear system to system stability graph. What does this SSG give us that was not available at the start? By bundling trajectories in the system and representing these bundles as simple connections, the global dynamics can be reduced to a finite set of connections. The alternative being exhaustive simulation, selecting a large number of initial conditions from an infinite number of possibilities. The PL functions within the system have been mapped to a finite number of nodes within a connected graph. The connections and nodes relate directly to the system dynamics and their interaction with PL functions such as logic rules. This relationship between the continuous and discrete system elements is almost impossible to track in a simulation, much less make sense of. Finally the resulting SSG is essentially a connected graph, and as such, even if the number of nodes and connections becomes too large to allow effective visual interpretation of the SSG, the graph can be analyzed for patterns and critical paths using well-tested graph theoretic algorithms that have long been in the literature (e.g., [21]). The relationship between nodes and PL system structures means that patterns found in the graph can be related directly to those elements of the system that generate that particular dynamic pattern.

Conclusion

The aim of this work has been to take ideas from linear theory, convex set theory, and computational geometry, and from them synthesize an analysis tool for dynamic systems with piecewise linear functions. This mixing of ideas from different disciplines was felt to be an effective way of approaching the analysis of a class of systems that is not within the scope of more traditional nonlinear systems analysis techniques. The work so far has been to develop the algorithms and computational techniques needed to achieve the different stages of the analysis and show their effectiveness. Future research is directed toward solving computational and memory issues related to the analysis process, as well as to widening the class of nonlinearities and hence systems that can be dealt with.

References

[1] R.E. Kalman, "Analysis and Design Principles of Second and Higher Order Saturating Servomechanisms," *Trans. AIEE Part II: Applications and Industry*, vol. 74, pp. 294-310, 1955.

[2] I. Flügge-Lotz, *Discontinuous and Optimal Control*, New York: McGraw-Hill, 1968.

[3] H. Stephanou, guest editor for a special issue on intelligent control, *IEEE Control Systems Magazine*, vol. 11, no. 4, June 1991.

[4] P. Antsaklis (chair), "Defining Intelligent Control," report of the Task Force on Intelligent Control, *IEEE Control Systems Magazine*, vol. 14, no. 3, pp. 4-5 and 58-66, June 1994.

[5] R.L. Grossman, A. Nerode, A.P. Ravn, H. Richel, eds., *Hybrid Systems*, Lecture Notes in Computer Science, 736, Springer Verlag, 1993.

[6] B. Lennartson, B. Egardt, M. Tittus, "Hybrid Systems in Process Control," *Proc. IEEE 33rd CDC*, Orlando, FL, pp. 3587-3592, 1994.

[7] E.D. Sontag, "Nonlinear Regulation: The Piecewise Linear Approach," *IEEE Trans. Autom. Control*, vol. 26, pp. 346-358, 1981.

[8] S.P. Banks and S.A. Kathur, "Structure and Control of Piecewise-Linear Systems," *Int. J. Control*, vol. 50, pp. 667-686, 1989.

[9] R. Wilson-Jones, N.B. Pettit, and P.E. Wellstead, "An Analysis Tool For Piecewise Linear Dynamic Systems," *IEE Colloquium on "Nonlinear Con-*

trol Using Structural Knowledge and System Models," Digest No. 1993/105, IEE, Savoy Place, London, WC2R 0BL, 1993.

[10] L.O. Chua, "Section-Wise Piecewise Linear Functions: Canonical Representation, Properties and Applications," *Proceedings of the IEEE*, vol. 65, pp. 915-929, 1977.

[11] W.M.G. van Bokhoven, *Piecewise Linear Modelling and Analysis*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1981.

[12] J.T.J. van Eijndhoven, "Piecewise Linear Analysis," Chap. 3., *Analog Circuits: Computer Aided Analysis and Diagnosis* (T. Ozawa, ed.), New York: Marcel Dekker Inc., 1988.

[13] V. Besson, N.B.O.L. Pettit, P.E. Wellstead, "Representing Piecewise Linear Systems for Analysis and Simulation," *Proc. Third IEEE Conf. on Control App.*, Glasgow, U.K., pp. 1815-1820, Aug. 23-26, 1994.

[14] D.P. Dobkin, "Computational Geometry and Computer Graphics," *Proceedings of the IEEE*, vol. 80, pp. 1400-1411, 1992.

[15] N.B. Pettit, P.E. Wellstead, "Piecewise-Linear Systems with Logic Control: A State-Space Representation," *Proc. Second ECC, Groningen, Netherlands*, pp. 1581-1586, July 1993.

[16] D. Chand, S. Kapur, "An Algorithm for Convex Polytopes," *Journal of the Association of Computing Machinery*, vol. 17, pp 78-86, 1970.

[17] P. McMullen, G.C. Shepard, *Convex Polytopes and the Upper Bound Conjecture*, London Mathematical Society Lecture Note Series 3, C.U.P., 1971.

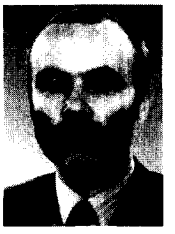
[18] N.B.O.L. Pettit, P.E. Wellstead, "A Graphical Analysis Method for Piecewise Linear Systems," *Proc. IEEE 33rd CDC*, Orlando, FL, pp. 1122-1127, 1994.

[19] P.A. Cook, "Simple Feedback Systems with Chaotic Behavior," *Systems & Control Letters*, vol. 6, pp. 223-227, 1985.

[20] N.B.O.L. Pettit and P.E. Wellstead, "Designing a Computation Environment for the Analysis of Piecewise Linear Systems," preprints of IFAC Nonlinear Control Design Symposium (NOLCOS '95), Lake Tahoe, CA, USA, pp. 947-952, June 1995.

[21] F. Harary, R.Z. Norman, D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs*, New York, John Wiley & Sons, 1965.

Njal B.O.L. Pettit received the B.Eng degree with diploma in electronic engineering in 1990 from the University of Hull, England. He then attended the Control Systems Centre, UMIST, where he received his Ph.D. in 1993. He is currently employed as a research associate at the Control Systems Centre. His research interests are in rule-based and logic control, hybrid systems, and related nonlinear systems theory.

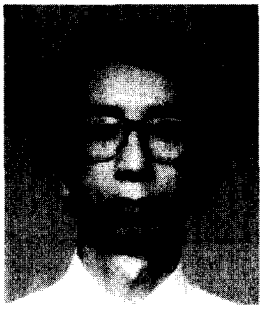


Peter E. Wellstead received the B.Sc. degree in electrical engineering from Hatfield College of Technology, England, and the M.Sc., Ph.D., and D.Sc. degrees from the School of Engineering, University of Warwick, England. He has worked for Marconi Instruments Ltd., first as an apprentice and subsequently as a design engineer. From 1970 to 1972 he was a Technical Fellow at the European Centre for Nuclear Physics, working on real-time control and image processing. He is currently Lucas Professor in Control Engineering at the Control Systems Centre, UMIST, where his teaching includes system modeling and the development of novel laboratory equipment. His current research interests are in adaptive systems and automotive control systems.

Correction

1995 IEEE Fellows

Editor's Note: The following was inadvertently omitted from the June 1995 issue of the Magazine, where the 1995 IEEE Fellows were featured.



Peter B. Luh
University of Connecticut
For contributions to the development of near-optimal and efficient manufacturing scheduling methodologies.

Peter B. Luh received his B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, Republic of China, in 1973, his M.S. degree in Aeronautics and Astronautics Engineering from M.I.T., Cambridge, MA, in 1977, and his Ph.D. degree in Applied Mathematics from Harvard University, Cambridge, MA, in 1980. Since 1980 he has been with the University of Connecticut, and currently is a Professor in the Department of Electrical and Systems Engineering and Director of the Production Systems and Information Technology Program within the Advanced Technology Center for Precision Manufacturing of the State of Connecticut. He is interested in planning and scheduling of manufacturing systems, and schedule and transaction optimization for power systems. Dr. Luh is an Editor for the *IEEE Transactions on Robotics and Automation* (1995-), was a Technical and Associate Editor for the same Transactions (1990-94), and an Associate Editor for *IEEE Transactions on Automatic Control* (1989-91). He is also an Associate Editor for the *International Journal of Intelligent Control and Systems*, and a member of Administrative Committee for IEEE Robotics and Automation Society (1992-1997). He won the Best Paper Award of the 1987 Joint Command and Control Research Symposium, and received Award of Appreciation at the 1993 East of California Asian American Studies Conference. Dr. Luh is a member of the Connecticut Academy of Science and Engineering, a Senior Member of the Society of Manufacturing Engineers, and an Associate Member of CIRP (the International Institution for Production Engineering Research).