Int. J. General Systems, Vol 13, pp. 333–344
Photocopying permitted by license only

QUALITATIVE SIMULATION OF TECHNICAL SYSTEMS USING THE GENERAL SYSTEM PROBLEM SOLVING FRAMEWORK

FRANÇOIS E. CELLIER
Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona 85721, U.S.A.

(*Received 21 January 1987; in final form 13 April 1987*)

In this paper, it is discussed how the General System Problem Solving (GSPS) Framework[1] can be applied to the identification of technical systems for the purpose of a qualitative simulation of such systems. Both advantages and severe shortcomings of this identification technique are demonstrated, and it is discussed under what circumstances this technique may eventually lead to good results. Major emphasis is devoted to the design of multi-layered hierarchical control systems.

INDEX TERMS: Qualitative simulation, General Systems Problem Solving framework, forecasting, inductive modelling, control systems, expert systems, artificial intelligence, human overload.

INTRODUCTION

The design of automatic feedback control systems has truly become a success story. Automatic devices for the control of continuous-time systems are in use everywhere. A modern airliner would not be operational without a multitude of automatic control loops. Controllers for linear single-input/single-output (SISO) systems are in constant use for roughly 50 years. Their design bases primarily on frequency domain techniques (Laplace transform, z-transform). Linear multi-variable (MIMO) systems have been successfully controlled since the sixties of this century when the state-space representation of linear systems in the time domain was introduced. Optimal controllers were developed by solving a Riccati equation to determine optimal coefficients for the state-feedback, and a merge between frequency domain techniques and time domain techniques has been found e.g. through the use of the so-called pole placement technique. A limited number of techniques have also been developed for the synthesis of nonlinear control systems, but these techniques are more specialized, and today's trend goes more into the development of techniques that make nonlinear systems behave like linear systems that we know better how to handle.

While these automatic feedback controllers work far more reliably than any human operator ever could do, they all lack "intelligence". They are very bad at making global assessments, and taking intelligent decisions on the basis of their findings. Although a certain success can be noticed in the design of decentralized controllers for particular types of hierarchical control systems (e.g. for power systems), they all perform properly under well defined and previously foreseen conditions only. No automatic device of today is able to take care of unforeseen emergency situations. This is still strictly the domain of the human operator who is inventive, and who may be able to devise a new control strategy "on-line", a strategy able to function properly under the given emergency conditions. That is:

our automated devices are always very systematic, but rarely ingenious, whereas the human operator is sometimes ingenious, but not necessarily always very systematic.

Where are the shortcomings of this division of labor between the human operator who makes the global decisions, and his auxiliary automatic control devices for the detail work? The amount of information to be processed depends on the complexity of the system to be controlled. It actually grows overproportionally (roughly quadratically) with the number of components of the overall system. As we design more and more complex systems, we are faced with a situation where the density of information to be processed grows rapidly. This causes a serious problem. When an automatic controller is not able to deal with the amount of data to be processed, we simply buy a faster computer. Unfortunately, we cannot "acquire" faster human operators. When the amount of information to be processed grows, the human operator has to rely more and more on his ingenuity, and less and less on systematism. Unfortunately, this is very risky as an essential piece of information may be easily overlooked when hidden under a wealth of detail data. This is called the *human overload* problem.

How is the human overload problem tackled today? One way to increase system safety is by *duplication.* This works with human operators equally well as with hardware components. Let the same data be processed simultaneously by several operators. If one of them overlooks an essential piece of information, the other may still find it. However, such a solution can only increase the system safety, not the capacity of digesting information quickly. If the "speed" of a single operator does not suffice, duplication will not help. *Decentralization* looks like an answer to this problem. The information is distributed among several operators such that the amount of information to be processed by each individual operator is reduced. However, this is not really an answer to the problem. We still need a *central coordinator* who decides which information is to be sent to whom. If this central coordinator is a human, then he will be the bottleneck in the system, and we have not solved our problem at all. On the other hand, if the coordinator is an automatic device, we are again faced with the previous problem in that this device will work properly except in an unforeseen emergency situation, that is: when its proper functioning would be most essential.

Let us repeat what turn out to be the two most common causes of complex system failure:

1) An important piece of information is not available at the right place at the right time. The *Challenger* disaster could probably have been avoided if the information flow would not have been distributed among some persons with technical knowledge but without decision power, and some other persons with decision power but without technical detail knowledge, and a very imperfect information link between those two groups.

2) An essential piece of information is available, but it is overlooked because it is hidden under an avalanche of secondary information. Several mid-air collisions have already taken place because there were so many aircrafts to be processed at the same time, that one of them simply got overlooked even though it was perfectly visible on the radar screens.

The most frequent causes of complex system failure are thus related to a breakdown of information flow. What really needs to be done in order to solve this problem is the construction of an automatic device that simulates the behavior

of human operators with all their inventiveness and global assessment capabilities, but without their shortcomings, that is: their speed limitation in information processing. This, of course, is a very ambitious goal, and we are still far from a solution. However, without a solution to this essential problem, we cannot continue to build more and more complex equipment.

The forthcoming *U.S. space station* is a device that is probably about one order of magnitude more complex than any single piece of equipment ever constructed by mankind so far. Yet, it is the declared goal of NASA to make this space station as *autonomously operational* as possible. Thus, the control of the space station will be primarily the duty of a few on-board astronauts rather than that of Mission Control at the Johnson Space Center (JSC), astronauts that cannot even devote their full attention to this task as they have other jobs as well. It is our conviction that this concept carries the seed of disaster, unless it is accompanied by the development of *intelligent automatic controllers* (expert systems) for the individual subsystems, and the development of an intelligent *executive expert system* as a central coordinator. Correct distribution of information even under emergency situations must not be left under the responsibility of a human operator (due to his inherent speed limitations). While final decisions can still be made by the astronauts, it is essential that the individual subsystem controllers are able to identify potential causes of failure reliably and ahead of time, and provide the human decision makers with informative problem reports. Even worse, these devices cannot be programmed statically. The programs must be able to *learn* as the space station is to be operated over a series of years, and it is certain that some components will suddenly fail, and will be replaced by improvised patches, while new components will be added on to the system. Thus, the expert systems including the coordinator must be able to operate under both *system degradation* and *system upgrading.*

CONCEPTS OF INTELLIGENT CONTROLLER DESIGN

Controllers of complex systems are always *multi-layered.* The innermost control loops require the fastest rate of updating, but the least intelligence. The outermost control loops are sampled at the slowest rate, but they operate on the largest amount of data, and they require the most "insight", that is: intelligence.

This concept works quite well, and there is little reason to deviate from this route. The innermost control loops are either classical PID controllers (lead/lag compensators), or state-feedback controllers. Nonlinearities are often controlled by on-line linearization (system parameter identification) and optimization of the controller parameters of the innermost control loop in an encompassing *adaptive control loop.* These two loops together are able to take care of mild nonlinearities, and can handle diverse modes of system operation, as long as all possible operational modes have been foreseen. The adaptive control loop can be updated (that is: sampled) at a slower rate than the underlying innermost control loop, but it exhibits a higher degree of complexity. At the next higher level, a central coordinator can be used to coordinate the behavior of several subsystems consisting of individual (eventually adaptive) controllers. At this level, performance optimization can be already quite difficult, and today's control engineers are usually quite happy if they can guarantee the *overall stability* of the system, that is: while the performance of the individual subsystem controllers is optimized, the overall system performance is not.

That is where we currently stand with respect to the design and practical implementation of automatic feedback control circuitry. All these controllers operate under carefully planned operating conditions, and while the innermost control loops are those least likely to change their behavior, the employed control algorithms are ironically the most *robust* ones. The hierarchically higher controllers deal with a multitude of system components that can either change their behavior or fail entirely. Thus, they should be made extremely robust in order to be able to perform their duties under varying system conditions. Unfortunately, they are not. Hierarchical controllers (central coordinators) are designed in such a way that overall system stability (but not optimal performance) is guaranteed under precisely defined conditions with respect to how many and which components may fail before overall stability is lost.

In particular, it becomes evident that none of the controllers is able to truly learn from modified system behavior. Adaptive control loops are able to "learn" new system parameters, but only under very well defined circumstances, and these parameters are strictly local parameters of a particular subsystem. No one even dreams of on-line optimization of the central coordinators, less the outermost control loop that determines the global strategies (set points), a loop which today always involves the human operator. That is: the one loop that requires screening of the largest amount of individual data points is controlled by the human operator who is great in recognizing global patterns, and who is truly able to learn from varying situations, but who is least capable of reliably viewing data in a systematic manner.

The question thus raises whether it is feasible to at least second the human decision maker by providing an automatic device that simulates the human decision making process, but without the severe capacity limitations of the human operator.

A SYSTEM THEORETIC APPROACH TO QUALITATIVE SIMULATION

Let us try to identify how a human operator handles unforeseen emergency situations. For that purpose, let us envisage the pilot who, after a non-destructive mid-air collision had lost most of the controls except for the thrust that went into the engines of his aircraft. He soon learned that he actually was able to control both the altitude of the aircraft and the pitch angle by the amount of thrust that he used. By carefully controlling this one input variable left to him, he was able to successfully execute an emergency landing, and thereby save the life of his passengers. How did the pilot manage this situation? The sequence of operations can be described as follows:

1) Identification of the problem, and determination of the controls (inputs) that were still at his disposal. For that purpose, he used a *mental model* of the effects of the different controls, and varied the controls carefully one at a time to see whether the predicted effects would still take place.
2) Identification of a subset of characteristics (state variables) that were related to his problem.
3) Determination of the manner in which the available controls did influence the related characteristics. For that purpose, a careful experimentation with

the available controls was now executed. On the basis of this experimentation, the pilot was able to identify a more global behavioral pattern of the related characteristics under the influence of the available controls (on-line identification of a forecasting model).

4) Determination of an appropriate strategy to use the available controls in an optimal manner to achieve the desired characteristics (that is: state trajectories).

It turns out that this "recipe" is much more general. It is not at all restricted to the above scenario. Thus, our automated device should actually follow the same pattern, that is: the overall task can be subdivided into the four individual subtasks outlined above.

The first subtask is not difficult. Detailed flight simulators can be built that simulate the behavior of the aircraft under correct operation very accurately. With today's technology, such simulators can easily be carried on board, in particular, as the accuracy is not essential for our task, and as there is no necessity for visual feedback. A technological data base can provide information about the tolerable range of experimentation. The flight simulator can be used to:

1) determine that something went wrong (by comparing predicted system behavior with observed system behavior), and thereafter to;
2) project the expected influence of exerting individual controls for comparison with the actually experienced system responses.

Thus, the flight simulator can be used to duplicate the pilot's mental model, and it can perform this task even better than the mental model would ever be able to.

The second subtask is not too difficult either as it contains primarily static information. The technological data base can provide us with the necessary information at all times.

The third subtask is a difficult one. We are meanwhile confronted with an *unknown system.* The flight simulator is hardly of any use to provide us with much information about its behavior. In order to "learn" the behavioral pattern of this unknown device, we need to extract data from it. For that purpose, we probably need to "shake" it (unless it shakes all on its own!), that is: voluntarily exert the available controls in order to retrieve information about the altered system's behavior. To optimally excite all frequencies, we will probably use a random exertion pattern for the available controls. This process is, of course, very risky as the tolerable exertions were determined for the original (that is: correctly functional and thus *known*) system, and not for the meanwhile deteriorated (that is: erratically functional and thus *unknown*) system. Moreover, once we have determined appropriate data, we still cannot rely on the structural information available in the flight simulator. Thus, a parameter identification will not suffice. We must identify the structure of the unknown process together with its parameters. The aim of the model is to provide us with an as accurate forecasting capability as possible. We suggest that an *optimal mask analysis*[2] might just be the right tool for that purpose.

The fourth subtask is again a little simpler. Once, we have gained confidence in a forecasting model, we can parameterize the available controls, e.g. by slicing the time axis into segments, and by requesting that each control be kept constant (at a yet unknown level) during each time slice. These unknown levels (there are *nc*ns*

such unknown levels where *nc* equals the number of available controls, and *ns* equals the number of time slices) determine the unknown parameters to be optimized in order to obtain an optimal fit between the desired flight trajectory, and the flight trajectory obtained by the forecasting model.

OPTIMAL MASK ANALYSIS AS A TOOL FOR THE IDENTIFICATION OF UNKNOWN TECHNICAL SYSTEMS

As we have discussed, the third subtask is the most difficult among the four. It was suggested that a technique (optimal mask analysis[2]) originating from the GSPS[1] framework might be the most appropriate handle for this task. In the remainder of this paper, we are going to analyse this approach in greater detail.

For that purpose, let us consider the simple state-space model:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -3 & -4 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

We are going to simulate this system in CTRL-C[3] using a random number generator for the input signal. Thereafter, we shall use SAPS-II[2] to recode the simulation results (which represent our "measurement data"). Then, we want to use an optimal mask analysis to determine a forecasting model. Finally, we shall use this forecasting model to predict the behavior of the system under various input conditions. We can easily verify whether our forecasts are "correct" by simply comparing the results from the *qualitative simulation* obtained by use of the SAPS-II forecasting module with the (recoded) results from the *quantitative simulation* performed by use of the CTRL-C simulation module.

Thus, we start by executing the following CTRL-C code:

```
A=[0,1,0;0,0,1;-2,-3,-4];
B=[0;0;1];
C=EYE(3);
D=[0;0;0];
T=0:0.1:10;
U=ROUND(RAND(T));
Y=SIMU(A,B,C,D,U,T);
```

A, *B*, *C*, and *D* are the four system matrices of the linear continuous-time system:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

which, in our example, has one input and three outputs. *T* is the time base. It is a row vector of length 101 containing the values [0.0.0.1 0.2 ... 9.9 10.0]. *RAND* (*T*)

generates a random vector of same dimensions as T where each element is uniformly distributed between 0 and 1. *ROUND*( ) rounds the values of the argument to the next integer, making U a stochastic binary trajectory over the time base T with values arbitrating between 0 and 1. The *SIMU*( ) operator finally simulates the linear system over the time base T using U as an input, and sampling Y over the same time base. Y is a matrix with 3 rows (representing the three output variables), and 101 columns (representing the sampling points).

Y looks almost like a raw data matrix. However, according to our SAPS-II[2] methodology, trajectories are columns rather than rows, thus:

$$\text{DATA} = [U'\ Y'];$$

will give us exactly what we need. There was no need to store the time base itself, but we decided to store the input stream as the first column of the raw data matrix.

Next, the data need to be recoded in order to obtain a finite state space. We decided to use three levels for each of the variables which are to be equidistantly spaced between minimum and maximum of each variable. This can be coded as:

$$\text{RAW} = \text{RECODE(DATA,'EQUI',0,0:2);}$$

Now, we can perform an optimal mask analysis on the recoded raw data matrix. Let us start by using the first state variable as output, and allow all state variables one time step back and two time steps back to be potential inputs together with the physical input variable at all three time instants. Thus, the mask candidate matrix can be coded as:

$$\text{MCN1} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \end{bmatrix};$$

The optimal mask analysis is performed by the command:

$$\text{MSK1} = \text{OPTMASK(RAW,MCN1);}$$

and reveals the following pattern:

$$\text{MSK1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix};$$

Thus, the first variable seems only to depend on its own past. Exactly the same results can be found for the second and the third variable. Why this result? Looking at the eigenvalues of the system matrix A, we find that they are located at -3.2695 and at $-0.3652 \pm 0.6916i$. Thus, the slowest time constant of this system is at 2.7380. However, the depth of the mask was chosen to be 0.2 only. Consequently over such a short time span, the variables can all be represented by straight lines, and thus, depend on their own past only.

Had we chosen the following time base instead:

$$T=0:20:2000;$$

we would have determined that:

$$\text{MSK1}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{bmatrix};$$

Thus, the first variable would have depended on the physical input only (and the same would have been true for the second and the third variable). Again, this result is quite obvious. In this case, the depth of the mask is 40, and even during the next recording, the system has almost reached its new steady state. (Remember that the stochastic input changes its value at sampling times only.) Obviously, the steady state values of our state variables depend on the physical input only.

What can be learned from these experiments? The optimal mask contains information not only about the *system* alone, but also about the input stream, that is: the *experiment* performed on our system. It is our experience that the best results are obtained if the depth of the mask is chosen to roughly cover the slowest time constant of the system that is to be captured. In our example we chose a depth of 3, thus:

$$T=0:1.5:150;$$

We ran the optimal mask analysis with the following three mask candidates:

$$\text{MCN1}=\begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \end{bmatrix};\quad \text{MCN2}=\begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & 1 & 0 \end{bmatrix};$$

$$\text{MCN3}=\begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix};$$

and found the following three optimal masks to forecast the three state variables:

$$\text{MSK1}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \end{bmatrix};\quad \text{MSK2}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & -2 & 0 & 0 \\ -3 & 0 & 1 & 0 \end{bmatrix};$$

$$\text{MSK3}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -2 & 0 & 0 & 1 \end{bmatrix};$$

Still, we do not know how large the influence of the particular input stream really

is. For that reason, we repeated the experiment with a second random stream. Thereby, we obtained the following optimal masks:

$$\text{MSK}=1\begin{bmatrix}0&0&0&\overline{0}\\-1&0&0&-2\\0&1&0&0\end{bmatrix};\quad \text{MSK2}=\begin{bmatrix}0&0&0&\overline{0}\\-1&-2&0&0\\-3&0&1&0\end{bmatrix};$$

$$\text{MSK3}=\begin{bmatrix}0&0&0&\overline{0}\\-1&0&0&0\\-2&0&0&1\end{bmatrix};$$

Only the first mask is slightly different. However, looking into the mask history, we found that, also using the first random stream, the newly found mask has a quality just slightly smaller than the optimal mask chosen during that analysis. Thus, we chose to have more confidence in the new mask.

However, we still don't know whether the same masks would also work when totally different input streams were chosen. Therefore, we repeated the experiment again, this time with:

$$U=\text{ROUND}(2*\text{RAND}(T)-\text{ONES}(T));$$

which creates a trinary random input pattern that arbitrates between -1, 0, and $+1$. With this input sequence, we obtained the following optimal masks:

$$\text{MSK1}=\begin{bmatrix}0&0&0&\overline{0}\\-1&0&0&-2\\0&1&0&0\end{bmatrix};\quad \text{MSK}=\begin{bmatrix}\overline{0}&0&0&0\\0&0&0&-1\\0&0&1&0\end{bmatrix};$$

$$\text{MSK3}=\begin{bmatrix}0&0&0&\overline{0}\\-1&0&0&0\\-2&0&0&1\end{bmatrix};$$

This time, the second mask is different, but again, the formerly found mask appears in the mask history with a quality just slightly inferior to the optimal mask chosen. Thus, we decided to stick to the formerly found mask.

Finally, we repeated the experiment a fourth time with the following non-random input pattern:

$$U=\text{SIN}(T/5);$$

This time, the following optimal masks were found:

$$\text{MSK1}=\begin{bmatrix}0&0&0&\overline{0}\\-1&0&0&0\\0&1&0&0\end{bmatrix};\quad \text{MSK2}=\begin{bmatrix}\overline{0}&0&0&0\\0&0&-1&0\\0&0&1&0\end{bmatrix};$$

$$\text{MSK3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

Now, all three masks are different. With respect to the first mask, we again found the previously determined optimal mask in the mask history matrix. The other two masks are probably not good at all. Our explanation for this discrepancy is that the sine wave signal chosen simply did not "shake" the system sufficiently well to extract appropriate information for the mask identification process. The random patterns chosen before were much better suited to excite the system at various frequencies.

FORECASTING BEHAVIOR WITH THE OPTIMAL MASK

After we had gained sufficient confidence in the quality of the obtained optimal masks, we decided to use them on all four data streams for forecasting purposes. To this end, we extracted the first 90 recordings of each data stream to be our new "raw data" streams, and used the optimal mask to forecast the behavior over 11 additional steps. In this way, we were able to compare the predicted (that is: qualitatively simulated) results with the previously obtained "measured" (that is quantitatively simulated) results.

```
//[frcst]=FRC(raw,input,m1,m2,m3);
//
//Forecast behavior of linear four variable system
r=raw;
[row,col]=SIZE(raw);
[n,m]=SIZE(input);
FOR i=1:n,...
      in=input(i);...
      fc=[in,0,0,0];...
      fcc=[r;fc];...
      ff1=FORECAST(fcc,m1,row+i-1,0);...
      ff2=FORECAST(fcc,m2,row+i-1,0);...
      ff3=FORECAST(fcc,m3,row+i-1,0);...
      ff=[in,ff1(row+i,2),ff2(row+i,3),ff3(row+i,4)];...
      r=[r;ff];...
  END
frcst=r;
RETURN
```

Figure 1 SAPS-II function to forecast the linear system's behavior.

The CTRL-C function, shown in Figure 1, performs the forecasting process for one data stream. Applying this procedure to the first set of data, we obtained the following comparison between "measured" and predicted state variables:

$$\text{MEAS} = \begin{bmatrix} \overline{2} & 2 & 1 & \overline{1} \\ 2 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 2 & 1 & 0 & 2 \\ 2 & 1 & 2 & 1 \\ \underline{2} & 2 & 2 & \underline{1} \end{bmatrix} \qquad \text{PRED} = \begin{bmatrix} \overline{2} & 2 & 1 & \overline{1} \\ 2 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 2 & 1 & 0 & 2 \\ 2 & 1 & 2 & 1 \\ \underline{2} & 2 & 2 & \underline{1} \end{bmatrix}$$

It turns out that not a single prediction was incorrect. Obviously, the recoding process had not destroyed too much of the vital information about the system, and the random pattern chosen to "shake" the system, was a very good idea indeed.

Applying the same masks to the second data stream, we found a few differences, that is: "incorrect" predictions. In the following table, these forecasts have been underlined:

$$\text{MEAS} = \begin{bmatrix} \overline{2} & 1 & 1 & \overline{2} \\ 2 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 2 & 1 & 0 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & \underline{2} \end{bmatrix} \qquad \text{PRED} = \begin{bmatrix} \overline{2} & 1 & 1 & \overline{2} \\ 2 & 1 & 2 & 1 \\ 0 & 2 & 1 & 0 \\ 2 & 1 & 0 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & \underline{0} \\ 0 & \underline{1} & 0 & 1 \\ 2 & 0 & 1 & \underline{2} \end{bmatrix}$$

The third data stream was the one that worked least satisfactorily. So far, we did not find any convincing reason why this has happened:

$$\text{MEAS} = \begin{bmatrix} \overline{1} & 1 & 0 & \overline{1} \\ 2 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ \underline{1} & 0 & 2 & \underline{1} \end{bmatrix} \qquad \text{PRED} = \begin{bmatrix} \overline{1} & \underline{0} & \underline{1} & \overline{2} \\ 2 & \underline{0} & 2 & \underline{1} \\ 0 & \underline{2} & 2 & 0 \\ 2 & \underline{0} & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 1 \\ 0 & 2 & \underline{1} & 0 \\ 0 & \underline{0} & 0 & 1 \\ 1 & 0 & 1 & 2 \\ \underline{1} & 0 & 2 & \underline{1} \end{bmatrix}$$

Finally, we tried our optimal mask on the sine wave input, and obtained:

$$\text{MEAS} = \begin{bmatrix} \overline{2} & 2 & 1 & \overline{0} \\ 2 & 2 & 1 & 0 \\ 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ \underline{0} & 0 & 1 & \underline{2} \end{bmatrix} \qquad \text{PRED} = \begin{bmatrix} \overline{2} & 2 & 1 & \overline{0} \\ 2 & 2 & 1 & 0 \\ 2 & 2 & \underline{1} & 0 \\ 2 & 2 & \underline{1} & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & \underline{1} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ \underline{0} & 0 & \underline{0} & \underline{1} \end{bmatrix}$$

that is, although the sine wave was not good for identification purposes, this does not mean that the previously identified optimal mask cannot be used to predict the behavior of the system when a sine wave is used as the input signal.

We consider these results rather convincing. They show that the methodology of optimal mask identification is a sound procedure to predict the behavior of technical systems. No *a priori* knowledge about the system structure was needed for that purpose. The identification and forecasting processes based solely upon the available measured data.

CONCLUSIONS

In this paper, it was shown that the GSPS framework provides us with a technique to identify unknown technical processes successfully for the purpose of behavior forecasting. This identification process requires no *a priori* knowledge about the system structure, and can thus fairly well be automated. Unfortunately, we must be able to shake the system up in order to obtain appropriate data for the identification. This is done best by applying a random pattern to all inputs. For systems that do not stand "shaking", the proposed methodology may not work equally well.

REFERENCES

1. G. J. Klir, *Architecture of Systems Problem Solving*. Plenum Press, New York, 1985.
2. F. E. Cellier and D. W. Yandell, "SAPS-II: A new implementation of the systems approach problem solver." *Inter. J. General Systems*, **13**, 4, 1987, pp. 307–322.
3. Systems Control Technology, Inc., *CTRL-C. A Language for the Computer-Aided Design of Multivariable Control Systems*. Systems Control Technology, Inc., Palo Alto, CA 94304, 1984.