IASTED

Proceedings of the IASTED International Symposium

MODELLING, IDENTIFICATION AND CONTROL

Grindelwald, Switzerland
February 17-19, 1988

EDITOR: M.H. Hamza

A Publication of
The International Association of Science and Technology for Development - IASTED

ISBN 0-88986-093-9

ACTA PRESS

ANAHEIM * CALGARY * ZURICH

Proceedings of the IASTED International Symposium MODELLING, IDENTIFICATION AND CONTROL - MIC '88, Grindelwald, Switzerland, February 17-19, 1988.

SPONSORS
The International Association of Science and Technology for Development - IASTED
Technical Committee on Modelling and Simulation
Technical Committee on Control

Editor: M.H. Hamza

ACTA PRESS
P.O. Box 2481
Anaheim, CA
U.S.A. 92814

ACTA PRESS
P.O. Box 3243, Stn. "B"
Calgary, Alberta
Canada T2M 4L8

ACTA PRESS
P.O. Box 354
CH-8053 Zurich
Switzerland

ACTA PRESS CODE: 137

TABLE OF CONTENTS

An example of ISDN U–Interface Simulation

Sílvio A. Abrantes

Faculdade de Engenharia da Universidade do Porto /
INESC–Norte
4000 Porto, Portugal

Abstract

A computer simulation of full-duplex 2-wire digital transmission on the subscriber loop using echo-canceling techniques is presented along with some results. The program is written in Pascal and allows flexible interconnections between modules because of its programmable topology and its first-in first-out interfaces.

Keywords : Simulation, Signal Processing, Algorithms, ISDN U-Interface.

1 – Introduction

Digital transmission in the existing telephone network will lead to the **I**ntegrated **S**ervices **D**igital **N**etwork (ISDN) of the near future. Due to its upcoming importance electronics and telecommunications companies all around the world, along with some national PTT's, have been doing research on all the aspects concerned with the reliable implementation of this service.

On the other hand it is well known that computer simulation is a very useful research tool on which our design choices and conclusions can be based.

Reflecting the work of the author, the purpose of this paper is to present a computer simulation of some aspects of the ISDN U-Interface. The transmission is full-duplex using echo-canceling techniques.

2 – A brief description of the ISDN U-Interface

The transmission over the existing telephone subscriber loops is analog. However, today's needs call for reliable high-speed bidirectional digital transmission of voice and data. That is the purpose of ISDN: merging data processing, office automation and telecommunications, it will provide a common basis for information transmission [1]. In this environment a 160 kbit/s rate is required.

To overcome the resulting difficulties of using a medium not intended for bidirectional digital transmission some ingenious processing has to take place, for example in the form of adaptive equalization and echo cancelation, or by choosing an appropriate line code.

In the so-called ISDN basic rate access (two 64 kbit/s B-channels plus one 16 kbit/s D-channel) several interface reference points have been defined. Fig.1 shows two of them:

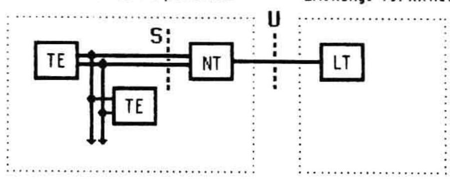


Fig.1 — ISDN S and U reference points.

- The **S** reference point – the **user-network interface** – connects the **T**erminal **E**quipment (TE) to the **N**etwork **T**ermination (NT) and is characterized by 144 kbit/s (2B+D) net user rate (in fact, 192 kbit/s total rate) over four wires .

- The **U** reference point – the **loop interface** – connects the **N**etwork **T**ermination to the **L**ine **T**ermination (LT) in the Exchange or Central Office and is characterized by the same net transmission rate (in fact, a total of 160 kbit/s is required, allowing for maintenance and other control information), this time over two wires.

In the most favoured method of transmission in the U-Interface [2] the 4-wires to 2-wires conversion takes place in a hybrid coupler.

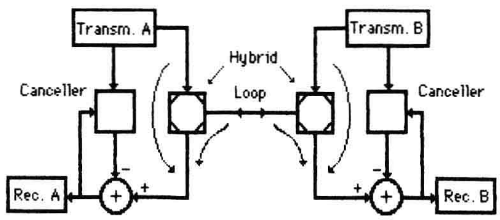


Fig.2 — Hybrid and echo canceler.

Ideally the signal coming from the near-end transmitter A would be delivered to the far-end receiver B but, because of loop mismatches (bridged taps, different loop lengths, etc.), an echo signal produced within the hybrid will flow to the near-end receiver A, disturbing the reception of the far-end signal coming from transmitter B. Thus the need of another device, an echo-canceler, to reduce or even cancel the echo signal. This is accomplished by adaptive signal processing in which the echo canceler tries to estimate the echo path impulse response.

3 – The simulation

Our research on these topics is based on computer simulation. For that matter a simulation program was written in Pascal. Running on a VAX-11/750 minicomputer, it is built of several modules, some of which perform general tasks (e.g., complex numbers manipulation, Fast Fourier Transforms) and others, the most of them, model the blocks of Fig.3. This is the basic, or "backbone", compile-time topology; however, as we shall see, in run-time the program can configure the system differently.

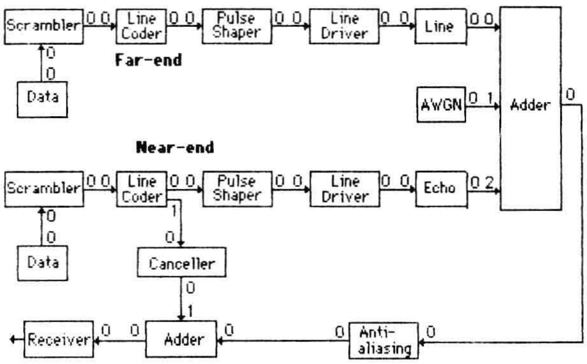


Fig.3 – Block diagram.

3.1 – Description of each block

In the system under study there are two transmitters, one in the near-end and the other in the far-end position. It can be seen that both branches possess a similar architecture.

The data source can generate two kinds of boolean data streams: a pseudo-random sequence or a random sequence. The former is periodic and is generated by a linear feedback shift-register configured according to a suitable polynomial. If this is a primitive polynomial of degree n the generated sequence will have its maximum period, 2^n-1, and is called a maximal-length, or *m*, sequence.

The boolean random sequence is obtained simply assigning the value TRUE (FALSE) to the output of a random number generator (with values between 0 and 1) if it is greater (less) than 0.5.

As the module can generate these two kinds of binary data, the user can compare the echo-canceler performance with near-end and far-end cross-correlated data (PN sequences) with that using less correlated data (random sequences).

The scramblers are based in shift-registers [3], as is usual. Their purpose is to ease timing recovery at the receiver and also to avoid statistical dependence between the near-end and far-end data streams.

The line coding block allows the user to choose and use one among several line codes implemented in the module: NRZ, biphase (WAL1), WAL2, AMI and 2B1Q. Each one has its advantages and drawbacks, so the simulation can help us to decide on the best one for this particular application.

To reduce the intersymbol interference due to bandwidth limitations in the channel or filters a signal pulse with good spectral characteristics must be considered. One that has found wide use in digital transmission has a roll-off raised cosine spectral characteristic and is implemented in the pulse shaper module.

Each filter in the system (e.g., echo path, line) can be specified by its own sampled impulse response, typed on the keyboard or previously saved in the filter library. This library is partly built of channel models found in the literature [4], [5], [6]. Some classical filters (Butterworth, Chebychev) can also be considered. These filters are modeled as FIR digital filters using a combination of the window and Helms's 4T's methods [7] in a repetitive fashion until there is close agreement between the original and the approximated transfer function.

Additive white gaussian noise with zero mean is also considered.

The echo canceler is an adaptive transversal filter with tap coefficients adjusted by the LMS or the sign algorithms.

3.2 – Interfaces and topology

A distinguishing feature of the program is the interface between modules.

Each module collects its input signals and delivers its output signals in first-in first-out (FIFO) queues [8].

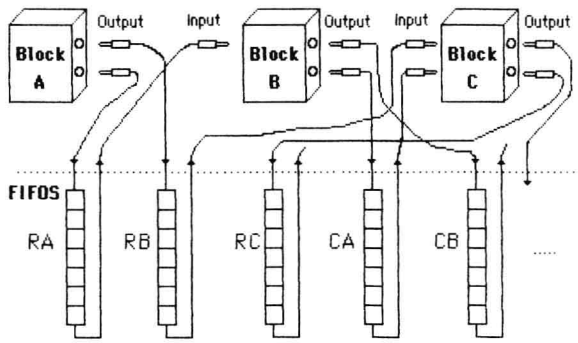


Fig.4 – The interface between blocks.

In Fig.4 we represent the program topology and interface philosophy: FIFO queues behave like information buffers (which they actually are...) provided with "plugs" to be connected to appropriate, coherent "boxes" (i.e., block inputs and outputs).

System configuration is the task of the **Topology Module** and FIFOs manipulations are performed by the **FIFOs Management Module**.

Each module representing a block of the system neither "sees" the other ones nor "knows" which FIFO queue it is attached to – this provides a great deal of flexibility allowing one to add, delete or make modifications in a given module without affecting the others – because all that information is defined in the just cited programmable topology module that manages the interconnections. As a consequence each module can be called several times to perform the same function with different parameters (different filters, for example) or the interconnections can be changed (e.g., scramblers or other suitable modules can be by-passed in run-time).

The block interface specifies the number of inputs and outputs (numbered from zero upwards, see Fig.3) and the corresponding FIFO name. Boolean-, real- and complex-valued FIFOs have been considered.

An example of an interface definition and topology follows (see Fig.3):

Declarations:

```
type
   nomes_fifos = (BA, BB, RA, RB, RC, CA, CB, CC, CD, CE, CF,
         CG, CH, CI, CJ);
   interface_bloco =
       Record
            entrada, saida : array [0..nr_max_fifos - 1] of
                                               nomes_fifos;
            nr_entradas, nr_saidas : integer
       end;    { interface_bloco }
var
   interface_scrambler1,
   interface_codlin1 {...}           : interface_bloco;
```

Executable section:

```
procedure topologia_scrambler1;
begin
    with interface_scrambler1 do      {near-end}
    begin
        entrada [0] := BA;   {input 0}
        saida [0] := BB;     {output 0 }
        nr_entradas := 1;    {number of inputs}
        nr_saidas := 1       {number of outputs}
    end { with }
end; { procedure topologia_scrambler1 }

procedure topologia_codlin1;
begin
    with interface_codlin1 do  {near-end}
    begin
        entrada [0] := BB;   {input 0}
        saida [0] := RA;     {output 0}
        saida [1] := RC;     {output 1}
        nr_entradas := 1;    {number of inputs}
        nr_saidas := 2       {number of outputs}
    end { with }
end; { procedure topologia_codlin1 }
```

In this example the FIFO queue called BB connects the near-end scrambler (**scrambler1**) output 0 to the line coder (**codlin1**) input 0; RA and RC connect the **codlin1** output to the pulseshaper and canceler modules, respectively.

Every major parameter of the system is defined interactively allowing, in an easy and systematic way, to study the performance in terms of the desired variable (be it a line code, a filter, an adaptive algorithm, or whatever one desires).

3.3 – Execution

Fig.5 shows a general overview of the simulation program modules and blocks.

In the outlet the main program calls a definition routine located in the **Definition Module**. Here input data is specified manually by the user or a previously created input data file is read. The simulation duration is set by the user as the number of symbols to be processed.

The main program calls all block modules involved in a simulation run. The calling order is irrelevant because input and output FIFOs are governed by the Topology Module. However, the program listing will be more readable and processing more efficient if a module routine is executed after all modules connected to its input have been called, that is, if the order of execution is the same as the signal flow.

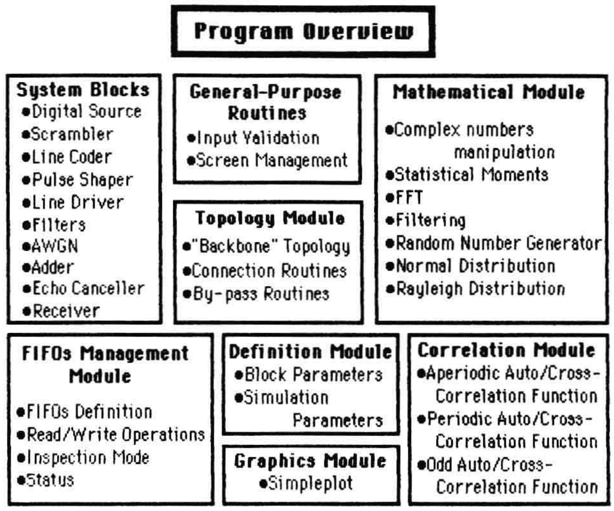


Fig.5 – Overview of program blocks and modules.

Each module routine performs its function if there are enough elements in the input FIFOs to be collected and processed and the output FIFOs have enough empty places to be filled in with processed data.

After all prescribed symbols have been generated by the sources the various input FIFOs gradually become empty and so the corresponding modules gradually stop executing. When every module has stopped executing the main program stops the simulation.

3.4 – Other features

A training period can be considered: the far-end transmitter is "switched off" so the convergence state is more readily reached. After this initial period true, double-talking operation takes place. Now adaptive tracking becomes easier because, in spite of the disturbance on convergence caused by the far-end signal, tap coefficients have nearly converged to their final values during the initial training period.

To improve convergence even more, as canceler operation proceeds a lower and lower step size can be taken.

To fully study and understand the influence of cross-correlated near-end and far-end sequences on the echo canceler performance a separate correlation module was written. It computes the aperiodic, periodic and odd autocorrelation and cross-correlation functions between sequences of arbitrary period.

Processing is done in the time domain except in filtering: the filter input is converted to the frequency domain through FFT and its output is computed by use of the "overlap-and-add" fast convolution method [9].

4 – Results

One can mention some graphical outputs[1] of this program:

- output of each block.
- transfer function and impulse response of each filter
- learning curve and echo return loss of the echo canceler.

Figs. 6-8 are examples of program outputs.

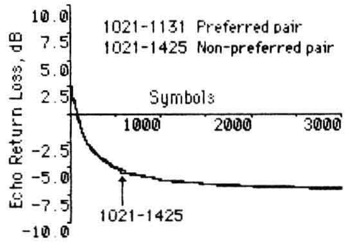


Fig.6 – Echo return loss for two different source data pairs.

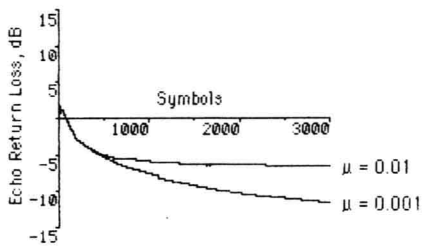


Fig.7 – Echo return Loss for two values of the step size.

Fig.6 shows two curves of the evolution of the echo return loss of a 16-tap echo canceler adjusted by the LMS algorithm, with a fixed step size: one curve corresponds to a preferred pair (1021-1131, in octal) of maximal-length PN near-end and far-end data streams with period 511, the other to a non-preferred pair of the same period (1021-1425). A preferred pair is one whose even cross-correlation function is three-valued. In certain applications it is the most convenient pair, however here it seems it doesn't matter if it is a preferred pair or not – in fact, both curves are undistinguishable. We expect to get some further results in the near future.

Fig.7 shows the echo return loss when different step sizes are taken, considering the same echo canceler and maximal-length PN sequences with period 8191.

Fig. 8 shows the influence of the step size value on the convergence of the same canceler's reference tap (taken as tap *8).

[1]The program uses the SIMPLEPLOT graphics package developed by Bradford University Software Services, Ltd., U.K.

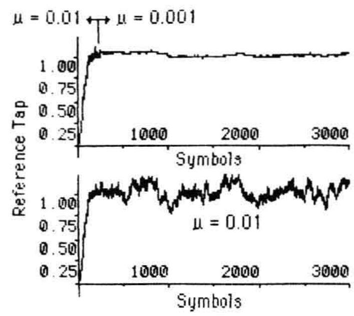


Fig.8 – The influence of the step size on the reference tap convergence.

5 – Conclusions

A programmable topology allowing different block configurations and the use of FIFOs to interconnect modules are the distinguishing features of the simulation program just described. They make the program highly flexible and rather user-friendly.

The program has proven to be a valuable tool in assessing U-interface performance. It is being used at INESC-Norte by other researchers in this field. Research is going on and further results are expected.

References

[1] **Wong, C.L. and Wood, R.**, "Implementation of ISDN", *Telesis*, Vol.13, 3, 1986, pp. 5-13.

[2] **Van Gerwen, P.J., Verhoeckx, N.A.M. and Claasen, T.A.C.M.**, "Design Considerations for a 144 kbit/s Digital Transmission Unit for the Local Telephone Network", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, No. 2, March 1984, pp. 314-323.

[3] **Lin, D.W. and Wu, L.**, "Conditioning of data sequences for 144 kb/s digital subscriber loop transmission", *Proc. International Conference on Communications*, Chicago, 1985, pp. 160-164.

[4] **Proakis, J.G.**, *Digital Communications*, McGraw-Hill, 1983.

[5] **Clark, A.P.**, *Equalizers for digital modems*, Pentech Press, 1985.

[6] **Bonnet, M. and Macchi, O.**, "Choix d'un algorithme en précision finie, pour annuleur d'écho", *Annales des Télécommunications*, 38, nº7-8, 1983, pp. 305-329.

[7] **Helms, H.D.**, "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters", *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-15, No.2, June 1967, pp. 85-90.

[8] **Messerschmitt, D.G.**, "A Tool for Structured Functional Simulation", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, No. 1, January 1984, pp. 137-147.

[9] **Stockham, T.G., Jr.**, "High-speed Convolution and Correlation with Applications to Digital Filtering", in Gold and Rader, *Digital Processing of Signals*, McGraw-Hill, 1969.

"UPSIDE-DOWN" HYBRID SIMULATION

David W. Pessen
Assoc. Professor, Dept. of Mechanical Engineering
TECHNION - Israel Institute of Technology, Haifa, Israel

ABSTRACT

In usual hybrid simulations of control systems, an analog computer simulates the process, and a digital computer the control system. This paper describes a case where an opposite, or "upside-down" method is followed: The process steady-state response is calculated on-line on a digital computer, while the analog computer simulates the control system or instrumentation. The method is used to simulate water mixing junctions, which can be used to extend the available water supply by diluting brackish with higher-quality water.

KEY WORDS: Hybrid simulation, water supply, mixing, diluting, brackish water.

1. INTRODUCTION

In typical hybrid simulations, it is customary to simulate the process on an analog computer, and the control system on the digital computer. This is done because process dynamics lend themselves well to analog simulation, whereas digital simulation of the control system permits trying out various sophisticated control algorithms.

There are processes, such as mixing systems, whose time lags are negligibly small as compared to the dynamic response of measuring element and final control element. However, the dynamics of mixing processes generally includes dead time (i.e., transportation lag). Such mixing processes can be very non-linear, so that it becomes inconvenient to simulate the process on an analog computer, whereas a digital computer can easily calculate the static response of the process. The control system, on the other hand, may consist of conventional PI- or PID control, so that digital simulation of the controller provides no particular advantage.

2. DESCRIPTION OF MIXING JUNCTION

An example of such a system is a mixing junction intended to dilute cheap but brackish water with more expensive low-salinity water, in order to obtain a mixture suitable for irrigation or even domestic consumption. This technique provides an inexpensive method of extending the available water supply in a given region, and is especially effective in arid or semi-arid regions where plentiful supplies of brackish water may be available.

To make the mixing process more efficient, it is advisable to control both salinity and outlet pressure of the mixing junction, shown in Fig. 1. The two water flow rates are Q_1 and Q_2, their salinities s_1 and s_2, and the inlet line pressures P_1 and P_2 respectively. Beyond the mixing point, we get a flow rate Q_1+Q_2, salinity s_3 and outlet pressure P_3. This outlet pressure is measured by a pressure probe and transmitter, and controlled by a pressure controller setting a control valve in inlet line 2. Control of outlet pressure is important, because the water may be utilized in an irrigation system using sprinklers or drip-irrigation units, which perform best under controlled-pressure conditions.

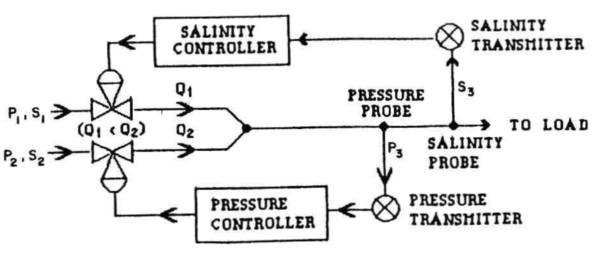


Fig. 1: Schematic diagram of mixing junction

The outlet salinity s_3 is measured by a salinity probe (usually based on conductivity effects) and transmitter, connected to the salinity controller. This sets a second control valve in inlet line 1. The flow ratio $r = Q_1/Q_2$ depends, of course, on the inlet and outlet salinities, and on the set-point of outlet salinity s_3. The material-balance equation is

$$s_1Q_1 + s_2Q_2 = s_3Q_3 = s_3Q_1 + s_3Q_2 \tag{1}$$

giving the following relation for flow ratio r:

$$r = \frac{Q_1}{Q_2} = \frac{s_3 - s_2}{s_1 - s_3} \tag{2}$$

The main problem in the operation of this system is interaction between the two control loops. This can degradate the system's dynamic response if proper design precautions are not taken. Indeed, the main purpose of the hybrid simulation to be described here was to study the effect of this interaction or coupling between the two loops. To minimize coupling, the salinity control valve should be inserted in the line having the lower flow rate, as shown in Fig. 1, where it is indicated that $Q_1 < Q_2$. Thus, a small change in Q_1 will effect mainly salinity, with little effect on outlet pressure. For $Q_1 > Q_2$, the salinity controller should be connected to the valve in line 2, and the pressure controller to line 1, since otherwise the coupling becomes even worse. Coupling is strongest in the $Q_1 = Q_2$ region. If the flow ratio r varies widely between values less than 1 to greater than 1, strong coupling cannot be avoided in parts of the operating range.

3. STEADY-STATE ANALYSIS OF MIXING JUNCTION

For the purpose of the steady-state analysis, the various flow resistances appearing in the mixing junction are shown schematically in Fig. 2. Here, $R_{L1,L2}$ are the

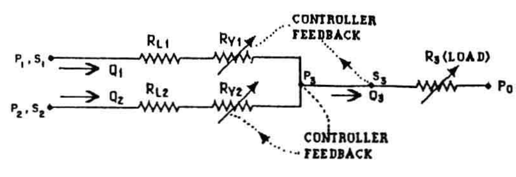


Fig. 2: Flow resistances appearing in mixing junction

hydraulic flow resistances of the inlet lines 1,2 (excluding the control valves), while $R_{V1,V2}$ represent flow resistances of the two control valves, which vary depending on the respective controller outputs. Finally, R_3 is the variable flow resistance of the load; for instance, an irrigation network. This is liable to vary widely, as local valves supplying water to various irrigated plots are opened or closed, and indeed, this represents the most serious disturbance to the control system.

Turbulent flow Q through a flow restriction is related to the pressure drop ΔP by the relation

$$Q = K\sqrt{\Delta P} \qquad (3)$$

where K is termed the "flow admittance" of the restrictions, related to flow resistance R by the relation

$$K = 1/\sqrt{R} \qquad (4)$$

In the case of control valves, we write

$$Q = mC\sqrt{\Delta P} \qquad (5)$$

where m = valve setting (m = 0 to 1)
and C = valve coefficient (supplied by manufacturer).

To simulate the system's steady-state response, we must derive equations giving outlet salinity s_3 and pressure P_3 as function of inlet conditions and controller settings m_1 and m_2. Outlet salinity s_3 is obtained directly from Equ. (1) as

$$s_3 = \frac{s_1Q_1 + s_2Q_2}{Q_1 + Q_2} \qquad (6)$$

To obtain P_3, we apply Equs. (3) to (5) to each inlet and outlet line. This gives flow resistances R, S and T for the three lines respectively, where

$$R = \frac{1}{K_1^2} + \frac{1}{m_1^2C_1^2} \qquad (7)$$

$$S = \frac{1}{K_2^2} \quad \frac{1}{m_2^2C_2^2} \qquad (8)$$

$$T = \frac{1}{K_3^2} \qquad (9)$$

The final solution is given by

$$P_3 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \qquad (10)$$

where $A = (RS + ST + RT)^2 - 4RST^2$ (11)

$B = 2P_1ST(RT-ST-RS) + 2P_2RT(ST-RT-RS)$ (12)

$C = (STP_1 - RTP_2)^2$ (13)

For the derivation of these relations, see Refs. 1 and 2.

4. HYBRID SIMULATION OF THE MIXING JUNCTION

Because of the strong non-linearity of the above-derived equations, it is impractical to simulate these on an analog computer, and an APPLE 2e computer was programmed in BASIC to calculate outlet salinity s_3 and pressure P_3 according to Equs. (6) and (10) respectively. On the other hand, since both controllers are conventional PI-controllers, there is no particular advantage in simulating them digitally. On the contrary, using an analog computer, the controller settings can easily be changed on-line using potentiometers, and the resulting effect noted immediately. The mixing junction itself, working with incompressible fluid, has practically no time delays, except for the dead time (transportation lag) caused by the distance at which the salinity probe is placed downstream from the junction point. This distance was assumed to be 10 m, to assure thorough mixing of the two streams before the measurement is made. However, this dead time is not constant, but equals

$$\text{Dead time } T_d = \text{distance/flow velocity} \qquad (14)$$

and thus varies radically depending on outlet load K_3.

While it is possible to simulate dynamically varying dead time digitally, this requires special equipment that was not available to the author. Fortunately, the dead time of the mixing junction appears only at the junction outlet, so that it is completely isolated from all steady-state effects. We are therefore justified in lumping this dead time together with all other time lags of the system, such as those of the measuring elements, controllers, and control valves, and all these time-dependent effects are easily simulated on the analog computer.

Dead time can be simulated on analog computers using the well-known Padé circuit. Fig. 3 shows a second-order Padé circuit which has been modified so as to minimize the required number of potentiometers. In order to vary the dead time dynamically, so as to be inversely proportional to flow rate Q_3, the two potentiometers in Fig. 3 are replaced by electronic multipliers, as shown in Fig. 4, with the x-inputs of the multipliers being proportional to flow rate. This dead-time circuit is then used in the complete analog-simulation circuit shown in Fig. 5. The remainder of that circuit is conventional, and should require no detailed explanation. Diode limiters are employed at the two output lines, to make sure that the output voltages do not exceed 5 v (which is the operating range of the A/D converters).

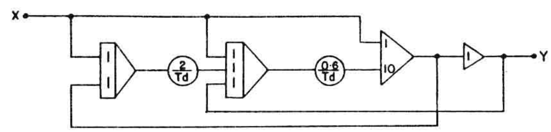


Fig. 3: Second-order Padé circuit using only two potentiometers

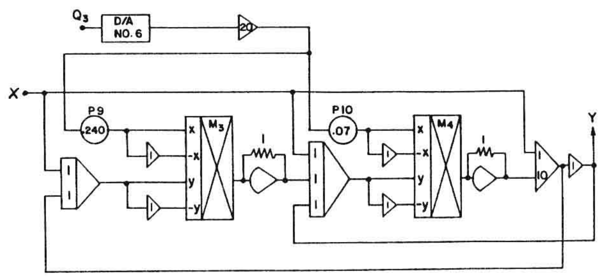


Fig. 4: Second-order Padé circuit with dynamically variable dead time

As interface between the analog and digital computers, an interface card having a number of 8-bit A/D and D/A channels was employed. These channels are utilized as follows:

A/D Channel No. 0: Inlet pressure P_2 (set manually by a potentiometer on the analog computer).

A/D Channel No. 1: Load admittance K_3 (set manually by a potentiometer on the analog computer).

A/D Channel No. 2: Salinity control-valve setting m_1 (set by salinity controller).

A/D Channel No. 3: Pressure control-valve setting m_2 (set by pressure controller).

D/A Channel No. 4: Outlet salinity s_3 (calculated by digital computer).

D/A Channel No. 5: Outlet pressure P_3 (calculated by digital computer).

D/A Channel No. 6: Outlet flow rate Q_3 (calculated by digital computer).

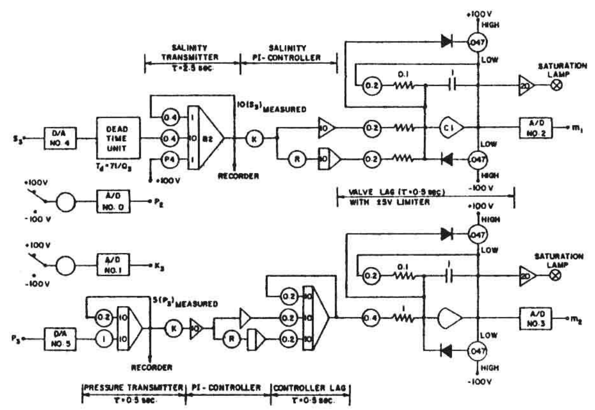


Fig. 5: Analog simulation circuit

When running the digital program, inlet pressure P_1 is set at the reference value 100, and the computer asks operator to specify inlet salinities s_1 and s_2, inlet admittances K_1 and K_2, and the two valve coefficients C_1 and C_2. Furthermore, the computer itself obtains the values of inlet pressure P_2 and load admittance K_3, as set on the analog-computer potentiometers, and also the control-valve settings m_1 and m_2, by means of PEEK commands. The calculated values of P_3, s_3 and Q_3 are then sent back to the analog computer by means of POKE commands, and an audible signal given to signal completion of the calculation (and thus give an indication of the sampling interval) whereupon the cycle is repeated. This sampling interval, determined by the time required for the digital computer to complete one calculation, turned out to be 0.7 sec. This is sufficiently short, considering that the recovery time of the whole system is in the order of 30 to 60 sec.

A typical recording of the simulation results is presented in Fig. 6, and shows response of outlet salinity and pressure to step changes in load admittance.

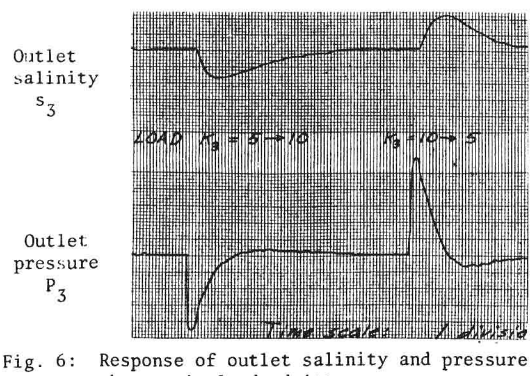


Fig. 6: Response of outlet salinity and pressure to step changes in load admittance. (Time scale: 1 division = 1.2 sec.)

As can be seen from Fig. 6, increasing the load admittance K_3 from 5 to 10 (i.e., drawing more water from the mixing junction), produces an immediate drop in outlet pressure. As a result, the pressure controller increases m_2 and thus the opening of the pressure-control valve. Since this increases the flow Q_2 (i.e., of the water with the lower salinity), outlet salinity begins to drop, but this only makes itself felt after a few seconds, due to the dead time and the measuring lag in the salinity loop. The outlet salinity drops somewhat, and then begins to increase again. This transient in the salinity loop is thus caused by the coupling between the two control loops. The system has returned to steady-state after less than a minute. While this may appear slow, it is perfectly acceptable for water mixing junctions.

One of the conclusions of the simulation results was that control valves with equal-percentage (i.e., logarithmic) characteristics should be used for both control loops, rather than linear valves, since they partially compensate for the non-linearity of the process. The simulation also shows that the system becomes less stable for lower values of load K_3. This can be explained by the fact that, as less water is drawn, the outlet flow velocity decreases, which increases the dead time, as shown in Equ. (14).

The hybrid simulation system described here has proven itself a valuable tool in the design of water mixing junctions. It permits modification of design parameters so as to reduce the coupling effect to a reasonable amount. The effect of changing any of the system's parameters is immediately visible.

Ref. 1: "Computer Simulation of Steady-State Flows in Mixing Junctions", D. Pessen, Research Report TME-416, Department of Mechanical Engineering, Technion, Haifa, Israel.

Ref. 2: "Hybrid Computer Simulation of Type A-30-P Mixing Junction", D. Pessen, Research Report TME-422, Department of Mechanical Engineering, Technion, Haifa, Israel.

MAISTRO

Modular Advanced Interactive Simulation system
To Real Time Operation.

Lars Peter Jensen
Dept. of Process Control, Inst. of Electronic Systems
Aalborg University, Frederik Bajersvej 7
9220 Aalborg Ø, Denmark

ABSTRACT

This paper introduces the simulation tool **MAISTRO**, which has been developed especially to facilitate Control Engineers simulating dynamic processes in real time.

MAISTRO offers an advanced environment for a simulation model, including input/output facilities, man/machine communication, graphical representation of the results on a screen and plots of these results for documentation purposes. The system is programmed in MODULA-2 and implemented on a PC-AT compatible.

KEY WORDS

Real time, Simulation, Control Systems.

INTRODUCTION

When a Control Engineer is developing a control system it is often necessary to simulate the process which is to be controlled. There are two reasons for this:

- Simulation makes it easier to develop a suitable model of a dynamic process and to document it.
- A simulation model can be used to test different controllers under equal conditions. There is no risk of damaging the process, and tests can be carried out at any time.

Simulation is usually carried out on micro-computers or PC's, and a new program is often written for each simulation model. A large part of these programs comprises environments for the model, e.g. the input/output facilities and the graphical representation of the results. Thus the simulation model is only a small part of the total simulation program, and it would save the Control Engineer much time if this was the only section to be rewritten when simulating a new process.

Several simulation languages and tools have been developed to accomplish this goal. Common to them is that they are all off line systems in the sense that you have to prepare the simulation, run it and look at the results afterwards. This means that real time simulations, where the simulator interacts with the surroundings (e.g. a controller) via I/O facilities (e.g. ADC and DAC) are not possible. Furthermore the user can't interfere during the simulation, meaning that if something is to be changed an entirely new simulation must be run.

As a consequense of this, a real time simulation system has been developed at the Department of Process Control, Aalborg University. The system is called **MAISTRO** (modular advanced interactive simulation system to real time operation) and this paper "demonstrates" the present state of **MAISTRO** - a functional execution part - by way of example: Simulating a combine harvester and a ground speed controller for this machine, to ensure a constant feedrate. Before the example the ideas and facilities of **MAISTRO** will be presented.

FACILITIES OF MAISTRO

Based on the discussions in the introduction **MAISTRO** was developed to offer the following facilities:

- The simulation model may be hierarchic and modular. In this way a decomposed model of the process, which is often the result of the modelling phase, can be used directly.
- Given an unchanged module interface, it is possible to replace the contents of a module with a better model without the necessity of a new compilation of the entire program.
- The simulation may be run in real time.
- During the simulation the operator can interfere interactively, changing parameter values.
- The simulator gives opportunity for communication with the surroundings through A/D- and D/A-Converters in real time, making it possible to test external controllers or apply input data from a process.

MAISTRO is implemented on a generally available computer that is cheap and has a high performance - a PC-AT compatible. The programming language MODULA-2 is chosen to get modular programs, making it is easy for others to make further developments to the system. The development of an easy-to-use man/machine interface is emphasized. The hardware consists of a display (screen), a keyboard and a printer and the software solution is based on window techniques and menus.

STRUCTURE OF MAISTRO

A system with this kind of facilities might be characterized as a real time operating system and **MAISTRO** is considered a simulation operating system. Like other operating systems it has different parts that handles different user tasks. This is illustrated in fig. 1.

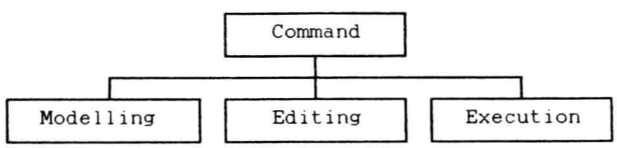


Fig. 1 Main structure of **MAISTRO**.

The names of the different parts equals the working phases for the user and the demands for each part is to facilitate the user in this particular working phase. This means that the demands is given by a specification of the working phases:

- Command is the main part of **MAISTRO**. In this phase the user decides which of the other phases to enter.
- Modelling is the phase where the behaviour of the basic modules (parts) of the decomposed model is described either in a simple modelling language or as difference equations written in MODULA-2.
- Editing is the phase where the basic modules are compound into more complex modules, ending up with one module containing the entire hierarchic model of the dynamic process. In this phase the basic modules are visualized on the screen as blocks with certain inputs and outputs. The connection of these inputs and outputs between different blocks is chosen by the operator by means of the curser keys.
- Execution is the phase where the simulation conditions: external input, output and time conditions are chosen. Simulation is carried out and the operator is able to change the conditions interactively. Data from the simulation may be manipulated (e.g. zoom) and copied for documentation purposes. The man/machine communication is based completely on windows and menus, with the possibility of presenting simulation data graphically in four different windows at a time.

All the parts of **MAISTRO** and the interfaces between them are completely described but at present only the execution part is implemented.

THE EXECUTION PART (AN EXAMPLE)

It is obvious that a simulation model of the dynamic process must exist as a program before it can be executed. This means that the operator must have passed the Modelling and the Editing part before entering the Execution part. Since it is the only part existing at the time we have to write a simulation program ourselves. The structure of this program must observe the rules for the interface between the editor and the executor, yet it is still simple to cunstruct a simulator, while not having to write the execution program. Further explanations about the model and the execution phase will be given by an example.

Simulation model for a combine harvester

The system yet to be simulated in **MAISTRO** is a closed loop control system containing a combine harvester an automatic feedrate controller for this machine. The simulation model is presented in fig. 2.

As shown in fig. 2 the model is both modular and hierarchic, which is one of the advantages of **MAISTRO**. There are 6 basic modules describing the dynamics of: the test fields, the summation point in a feed back control system, the controllers, the driving system of the combine, the header and the feeder. The 3 last modules describe the combine, so they are compounded into a complex module called combine. The combine and the 3 first basic moduls are compound, into the final closed loop control system. The interaction between the different modules in a complex module are handled by a co-ordinator algorithm.

In the present state of **MAISTRO** the co-ordinator only connects inputs and outputs at each sampling point, but we plan to use the ideas of Modular Integration /C.B. Brosilow, et.al./. The input and output that pass by the outer complex module are called external input/output and before an execution starts they must be connected to an ADC/DAC, a file or declared as constants or functions.

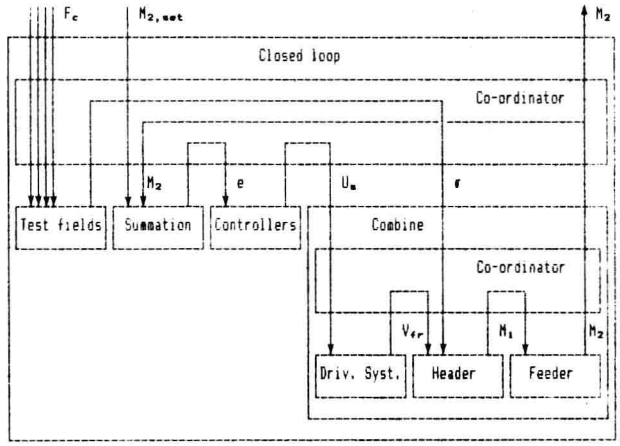


Fig. 2 Simulation model for a control system for a combine harvester.

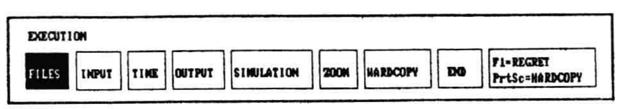
The aim of the control system is to adjust the ground speed (V_{fr}) of the combine so that the throughput at the feeder house (the feedrate M_2) is kept at the set point value ($M_{2,set}$) even with crop density (σ) variations. With this simulation model different controllers are tested on a pure simulation basis under identical test conditions described in the module "Test fields", where different test conditions may be chosen by the inputs F_c. The simulations are used to choose between different controller algorithms. When the best and most robust controller is found the control module may be replaced by an external controller communicating in real time with the simulated combine through ADC and DAC. In this way the real implemented controller can be tested before it gets control over a real combine.

The dynamics of the basic modules of fig. 2 are described as difference equations. The simulation model is programmed in Modula-2. The program is then compiled into executable code which observes proper interface conditions and thereby is able to communicate with the executor.

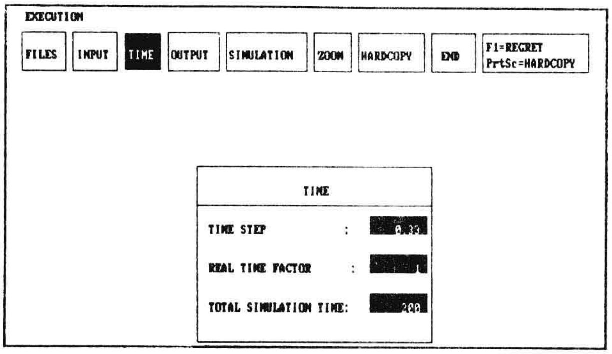
Demonstration of the execution phase

To demonstrate the execution phase several hardcopies of the screen dialogue will be presented and commented. Because they are parts of a dialogue between the operator and the system the hardcopies will not have a figure no. nor figure captions. The curser keys are used to move the highlighted section and the highlighted menu is chosen by pressing the enter key.

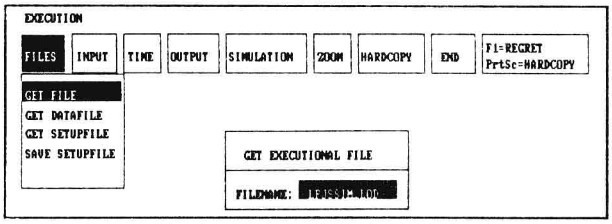
When the execution part is invoked the following picture will occupy the top of the screen:



The simulation program is to be a function of the sampling period. This is only possible if the time conditions are chosen before the simulation program is fetched, so we choose the time menu, and a dialogue box appears:

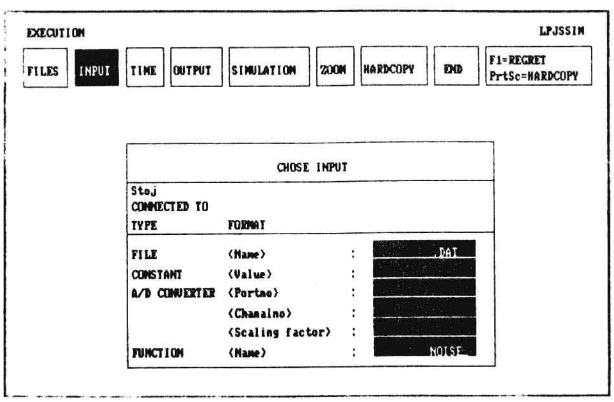


The wanted information must be filled in. The time step determines the step between each sample on the simulation time axis. Time scaling is possible using the real time factor. A real time factor eq. to 1 secures a true real time simulation, while a real time factor eq. to zero makes the computer simulate as fast as possible. The total simulation time determines the simulated time. When the time conditions are chosen the simulation program can be fetched. This is done as a part of the file menu:



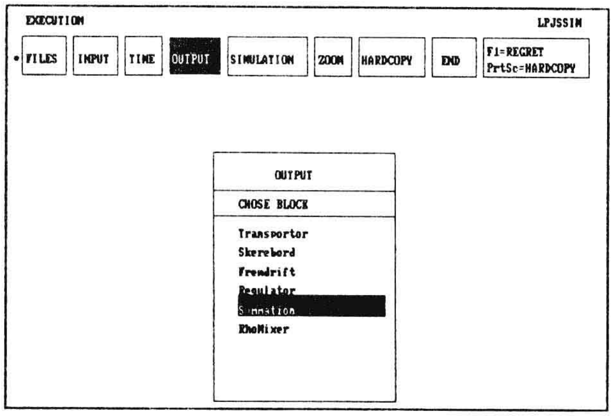
As it is seen the menu "fetch file" opens a dialogue box where the name of the file (the simulation program) has to be be written. The other menu's in file menu make it possible to fetch an input data file and fetch or save a file containing a total set up for a simulation. If a lot of similar simulations is to be run it will be convenient to save the set up.

The next step is to choose the inputs. If the input menu is chosen, another menu with a list of the external input appears. It is possible to scroll down the list and choose one input at a time. When a choice is made a dialogue box appears and the operator has to connect the external input to one of the possibilities. This is shown in the next screen picture:

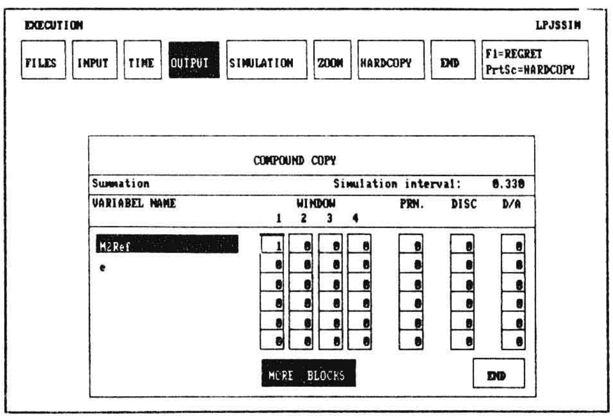


In the example we are about to connect one of the inputs that determine the test conditions. It is connected to a function called noise that creates Gaussian white noise that is added to the crop density (σ) to simulate realistic variations. Other possibilities are data from a file, a constant, or a value read from an ADC.

When all inputs are connected we choose the output menu to select outputs to be saved and/or presented:



All the basic modules are listed and the operator can scroll down the list and choose one block at a time. When this is done the following screen picture appears:

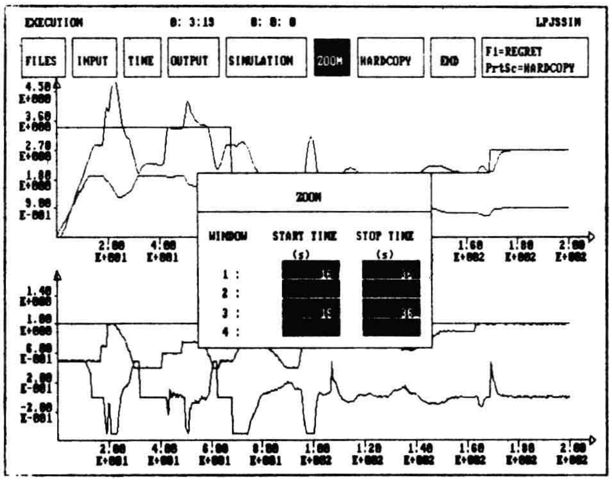


The example shows the summation point where 2 outputs are listed. This is because we want to be able to save the external inputs as well as the outputs, so they are listed as outputs in the blocks where they appear (see fig. 2). The screen can be parted in up to 4 windows and each variable can be presented in one or more window(s), with the exception that there can only be 4 variables in each window. Other possibilities are to have the variable printed out in figures, saved on a disc or transferred to a DAC.

Another facility of the output menu is a possibility to scale the y axis of the plots before simulation. If this is not done a default value is used to scale the axis.

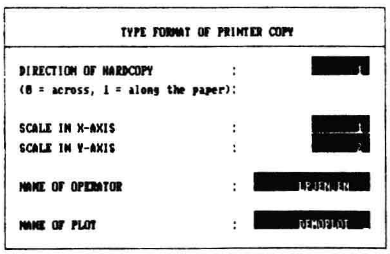
Now we are ready to choose the simulation menu where we can start, stop or continue a simulation. If start is chosen the simulation program will be executed with the chosen set up. During the simulation the chosen output curves are updated at each sampling point and it is possible for the operator to interfere and change the input or output conditions as the execution menu remains active.

The zoom menu might also be used during simulation but it has major advantage after the simulation time has gone. Then it is possible to zoom among the stored output data to see interesting details clearly. An example of the zoom menu is shown in the next screen picture:



In the dialogue box the operator can choose the passages to be zoomed in each window. In this example only window 1 and 3 are chosen. This will produce only 2 curves but they will use the whole screen, splitting it horizontally into 2.

All curves shown graphically on the screen can be hardcopied for documentation purposes. This is done by the hardcopy menu, which produces the following dialogue box:



In this box the operator can make some choice about the direction and scaling of the plot, and it is possible to name the plot. When the hardcopy is made the main menu line will be replaced by these informations and date and time for the simulation. When all interesting plots have been made the operator can use the end menu to close down the execution part and thereby **MAISTRO**.

CONCLUSION

As it is seen in the demonstration of the execution part, **MAISTRO** is an easy-to-handle simulation tool which enables the Control Engineer to run many simulations quickly and easily (e.g. I have made a very large simulation series with 30 runs and 110 plots in less than one week). A major advantage is that **MAISTRO** is able to interact with the surroundings in real time so that, e.g. controllers implemented on another computer can be realistically tested on a simulator before they are used to control a real process.

MAISTRO is developed especially to facilitate Control Engineers in the forementioned way. This has made it necessary to reduce the complexity of the processes to be simulated, so that e.g. power plants can't be simulated. But there are still many processes that might be simulated in **MAISTRO**, and for a Control Engineer simplification might be considered as an advantage, as it has made **MAISTRO** very easy to use.

About two man years have been invested in the previous work which is already a helpful tool, but it is of course a limitation that the Modelling and the Editing parts are not yet implemented. This will be the next step and will probably take another two man years. Minor details of the execution part are also not yet functioning but they soon will.

In the demonstration only a few comments are made on the co-ordinator, because in the present state of **MAISTRO** it only connects inputs and outputs at each sampling point. In this way only basic blocks without input that depends on output from a "later" block to the same sampling point can be simulated. This is no problem for the combine model, where there are no feed backs without time delay, and this will also be the case for many other processes. Anyhow we plan to remove this constraint using Modular integration as described in /C.B. Brosilow, et.al./.

Another limitation is the computer power, which determines the minimum sampling period for real time simulation. For the simulated process this is 0.05 sec which is satisfactory.

As mentioned earlier **MAISTRO** has only been used on one application, but when it is fully implemented it can be used to simulate a huge class of minor processes (e.g. a drum boiler or a combine) in real time.

ACKNOWLEDGEMENTS

The projekt is sponsered by the Danish Technical Scientific Research Counsil and it is a co-operation between the Department of Process Control at Aalborg University and Dronningborg Machinefactory. Both DTSRC and DM is greatefully acknowledged.

REFERENCES

C.B. Brosilow, et.al., "Modular integration methods for simulation of large scale dynamic systems," paper presented at the International Seminar on Modern Methodes in Dynamic Simulation of Industrial Processes, Trondheim Norway, May 1985.

Christensen, T.B., Kvorning, S., Koch, P.I., "MAISTRO," Final report for the M.Sc. from the System Construction line, Aalborg University, Fr. Bajersvej 7, DK-9000 Aalborg, Denmark.

Jensen, L. P., "Development, implementation and field test of an automatic feedrate controller for combine harvesters," Puplication from: Institute of Electronic Systems, Aalborg University, Fr. Bajersvej 7, DK-9000 Aalborg. R 87-18.

Modula-2/86 Users manual, Logitech Inc.

Halo functional description manual, Media Cybernetics Inc.

A Simulation Model for Road Traffic Networks[1]

C.ARENA, D.DE LUCA CARDILLO and A.MORO

Dipartimento Statistico - Universita' di Firenze

Abstract. This paper deals with a micro-simulation model of the traffic flowing on a road network of multi-lane suburban highways. The representations we used to describe the physical structure of the road network, the vehicle characteristics and the driver behaviors are emphasized. The most significant simulation procedures, related to the advancement and the lane changing of vehicles, are evidenced. The main feature of the model is its modularity which allows a flexible use in traffic planning and road designing. An application is presented.

1. INTRODUCTION

It does not seem necessary, nowadays, to stress the importance of a sound and comprehensive simulation model for dealing with traffic problems, in particular, when it is required to evaluate the effectiveness of a road network.

We were dealing with a network of multi-lane suburban highways. In such a case a microscopic simulation model suggested itself as the most flexible tool to allow the effect of alternative changes of traffic environment to be assessed.

In what follows, we present several modules for the simulation of the main traffic situations which can be present on such type of network. The results can concern both the modifications to be made to an existing network as well as the control of the features of a design one.

Our first step is to represent the traffic system, i.e. the geometry of the network, the characteristics of vehicles, the driver behaviours and the traffic scenario (see Section 2). Section 3 is devoted to an exposition of the simulation model, while Section 4 deals with an application. Finally, Section 5 offers some conclusive remarks.

2. MODEL FOR TRAFFIC SYSTEM

Our traffic system model is organized into the following steps:
- the coding of the physical elements of the road network;
- the determination of the parameters defining the vehicle characteristics and the driver behaviors;
- the representation of the traffic scenario.

2.1 The scheme of the road network

Let us call *nodal points* the road sections corresponding to which we have either:
- the entry of the vehicles into the network (*generating points*); or
- the exit of the vehicles from the network (*terminating points*); or
- the choice of two or more paths (*decision points*).

A *road segment* is the part of the network between two consecutive nodal points.

Each road segment is divided into one or more *subsegments*, which are bounded by sections with a discontinuity (confluences, bottlenecks, etc.).

Each subsegment has one or more *lanes* and it is homogeneous with respect to some characteristics (number of lanes, maximum speed, etc.).

A *path* is any sequence of road segments connecting a generating point to a terminating point. Corresponding to each terminating point, the model assumes a *terminating storage* whose dimensions are determined on the basis of the characteristics of the zone immediately following the terminating point.

The nodal points are progressively numerated following the algorithm [7]. Accordingly, a path is identified by an increasing sequence of numbers corresponding to its nodal points.

Each element of the network is characterized by a code and by one or more parameters: in particular, we associate
- to each path the number N_1 of segments which form it;
- to each road segment the number N_2 of subsegments which form it;
- to each subsegment its length L and the number n of its lanes;
- to each lane the maximum running speed v_i (where i denotes the number labeling the lane which assumes increasing values starting from lane 1);
- to each terminating storage the maximum speed with which vehicles can enter there.

2.2 Characterization of vehicles and of driver behaviors

The model we built up assumes the volumes of traffic to be expressed in terms of passenger-car equivalents. Therefore, it is necessary to characterize the standard vehicle through the following parameters:
- its length l;
- its maximum acceleration γ_{max}, as function of the speed;
- its maximum deceleration γ_{min}.

The model takes into account the driver behavior through two more parameters:
- the *safety distance* (which each vehicle keeps with respect to the preceding one) defined as

$$d_{saf}(v_0,v_1) = \begin{cases} a\,(v_1 t_d + l + b) & \text{if } v_0 \geq v_1 \\ 2l & \text{if } v_0 < v_1 \end{cases}$$

where
v_0 = current speed of the vehicle under consideration;
v_1 = current speed of the preceding vehicle;
t_d = driver reaction time;

$$a = \left(1 - \frac{v_0 - v_1}{2\alpha_0}\right)^{-1}; \quad b = \frac{(v_0 - v_1)^2\, t_d}{2\alpha_0};$$

α_0 = experimental constant with the dimensions of speed (connected with the "sensitivity" of the driver [6]);
- the *maximum desired speed* (v_{des}) assigned on the basis of a probability distribution which takes into account both the variety of vehicles circulating on the roads and the "driving styles".

2.3 Definition of the traffic scenario

For a correct representation of the traffic scenario which forms the main part of the input of our simulation model , we must assign to each generating point:
- the entering volume Q in vehicles per hour [vph];
- the probability law of the interarrival times (headways);
- the percent sharing of vehicles among the lanes;

[1]Research supported partly by a M.P.I. "40%" grant on the program "Stochastic calculus and stochastic dynamical systems" and partly by a research grant from the firms "La Fondiaria" and "A.G.I.P.".