Simulation of Systems, L. Dekker, editor

GASP-V: A UNIVERSAL SIMULATION PACKAGE

François E. Cellier André E. Blitz

Institute for Automatic Control
The Swiss Federal Institute of Technology
Physikstr. 3
CH-8006 Zurich ; Switzerland

An ANSI-FORTRAN-IV written subroutine package capable of handling continuous-time problems (described by mixed ODE's and PDE's) as well as discrete and combined problems is presented. GASP-V is fully upwards compatible with GASP-IV [1,2], but enlarged by:

a) some subroutines out of the program FORSIM-V [3] for easy coding of distributed systems (initial value PDE's of parabolic and/or hyperbolic type) using the method of lines.

b) an effective run-time library for modeling of continuous-time problems (comparable to the one offered e.g. by CSMP-III), but coded in a much more effective way since all discontinous functions are broken up in series of piecewise continuous functions, by utilizing time- and/or state-events to switch over from one branch to the other and by means of that resulting in remarkable reduction of execution time (all "creeping" avoided).

c) a comprehensive selection of different integration algorithms placed at the user's disposal.

1. INTRODUCTION

Simulation packages are normally divided into three classes according to the problems which can be handled by them. These are:

a) simulation of continuous-time systems described by ordinary differential equations (ODE's)

b) simulation of continuous-time systems described by partial differential equations (PDE's)

c) simulation of discrete-time systems described by difference equations mixed with discrete time-events.

Program packages for the treatment of problems out of class (c), e.g. simulation of inventory systems or of queuing situations, exist for approximately 20 years. The most commonly used packages for this purpose are these days: GPSS, SIMULA, SIMSCRIPT and SIMPL/I.

Problems of type (a) are solved by use of digital computers since about 10 to 15 years by utilizing standardized multipurpose packages. Typical examples of such packages are: CSMP-III, CSSL-III, ACSL, DARE-P and MIMIC. These packages are e.g. surveyed in [4,5].

First packages for the numerical solution of problems out of class (b) have been announced about 7 years ago. An optimal package for this purpose, however, has not been developed, since this problem is numerically much more complicated to solve and since there exists in this class of problems a much stronger link between the problem to be solved and the optimal algorithm to be used than in problems of classes (a) or (c). For this reason it is also doubtful whether it will be possible at all to develop a package which really could be called a "general purpose" package for the solution of problems out of class (b). An optimal state, however, is not yet achieved. Development goes on with the increased possibilities of new computer technologies (shorter cycle time, bigger memory). Packages for certain subclasses of (b) are e.g.: LEANS-III and DSS. Surveys are given in [5,6].

It is to be stated that the division of simulation packages into these three classes (a), (b) and (c) results rather from the different simulation techniques used than from the physical world being divided into three classes of dynamic processes. This can easily be shown in the field of traffic control. There one distinguishes between:

a) *macroscopic models* which are entirely continuous and in which the single vehicle is not considered but only the traffic flow with accompanying densities

b) *microscopic models* which are entirely discrete with vehicles entering the considered system, traveling through it in discrete steps e.g. from one corner to the next with queuing situations in front of traffic lights etc.

c) *submicroscopic models* (not often used in this context) which are continuous with specific discontinuities describing the dynamic behaviour of every single vehicle.

The physical process, however, is identical in all three cases. The different simulation techniques have been selected in accordance with the *model of the process* (and not with the process itself). The model of the process has been deve-

loped in a way suited for optimal fitting of given simulation objectives.

There exist simulation objectives which suggest utilization of mixed models. By simulating an elevator system it may be useful to model the arrival time of passengers and their destinations by a discrete model, the motion of the elevators on the other hand continuously. To allow such models to be realized, there have been made first attempts recently to code packages able of handling mixed models which partly belong to one class and partly belong to another, showing all attributes of both classes and in addition some new attributes describing the interactions between them.

There has been announced 1974 a package for the treatment of mixed models out of the classes (a) and (b) called FORSIM-V [3]. Another package GASP-IV [1,2] allows handling of models out of classes (a) and (c).

In this paper a new version of GASP is described, which enables the user to code models out of all three classes. To achieve this goal some of the subroutines from FORSIM-V have been combined with GASP. Besides the GASP-IV package does not place a set of functions (≙ run-time library) as a hysteresis function, a step function etc. at the user's disposal. The reason for this is quite clear. By taking advantage out of the much better possibilities of GASP-IV for the treatment of discrete events a discontinuous function should always be broken up into a set of piecewise continuous functions connected by *time-events* (as in the case of the step function) and/or by *state-events* (as in the case of the hysteresis function). A time-event is a discrete happening taking place in a predetermined instant of the simulation time and is an attribute of class (c). A state-event is a discrete happening taking place in an instant of time which, itself, depends upon the state of the system. It may e.g. take place when a specific state variable crosses a prescribed threshold. State-events, therefore, are new attributes originating from interactions between elements of class (c) and (a). To accomplish time-events a *file handling system* (data-base organization problem) is needed. To handle state-events one needs a mechanism for detecting whether such an event takes place during the current integration step, as well as an iteration procedure to locate it properly.
A hysteresis function should for example be cracked into:

a) a function subprogram for computation of the three continuous branches of the discontinuous function, called from subroutine STATE (for computation of the state derivatives)

b) a function subprogram for detecting whether a state-event (switching from one branch to another) is required during the current integration step, called from subroutine SCOND (for detection of state-events) and

c) a function subprogram for carrying out the actual switching over from one branch to another, called from subroutine EVNTS (for accomplishment of events).

Since all of these subroutines (STATE, SCOND and EVNTS) are user-supplied the user would have to call three different function subprograms to code one single hysteresis function. In GASP-V there has been found a way to code such functions exactly as described above with the exception that the function subprograms (b) and (c) are hided in GASP-V, so that the user can model such functions again the same way as he is used from packages like CSMP-III.

By using pure continuous simulation packages even large systems may be simulated in a very attractive way as long as there are no discontinuous functions involved. As soon as there are, serious numerical troubles arise. By using integration algorithms with fixed step size the accuracy of the results will be very bad, in many cases there will result even an overflow of one state variable or another. By using variable step integration algorithms the program normally starts "creeping". This results from the step size being reduced to very small values each time a discrete event takes place which is a very inefficient way of locating state-events and an even criminal way of looking up time-events! By use of GASP-V all of these numerical problems may be avoided. GASP-V is, therefore, recommended for simulation of all systems with discontinuous functions forming part of the model even if there are no further discrete elements involved and even if most people do call such models "continuous" which for our opinion is wrong . These models are mixed models of the classes (a) and (c).

In GASP-IV the integration algorithm (Runge-Kutta algorithm of fourth order with variable step size) has been coded directly within the central subroutine GASP. This concept is not very flexible since no alternative integration algorithm can easily be accessed. For GASP-V the subroutine GASP has, therefore, been entirely rewritten in a way such that the user may select the integration algorithm to be utilized. This is specially important in connection with the simulation of PDE's as will be shown further down.

2. PARTIAL DIFFERENTIAL EQUATIONS [7]

For the treatment of PDE's the FORSIM-V program has been selected, mainly because FORSIM-V does not use the concept of *master equations* which makes it more flexible for use in problems modeled by sets of coupled PDE's and mixed problems

out of the classes (a) and (b).

Since GASP-V bases upon the FORSIM-V package for handling of PDE's, we will describe first the essential attributes of this package. FORSIM-V is designed mainly for the solution of parabolic and/or hyperbolic initial value problems in one space dimension within a limited interval D(x) = [0,L] using only one derivative in the time domain t and up to second derivatives in the space dimension x.

$$\frac{\partial \underline{u}(x,t)}{\partial t} = \underline{f}\left\{\underline{u}(x,t), \frac{\partial \underline{u}(x,t)}{\partial x}, \frac{\partial^2 \underline{u}(x,t)}{\partial^2 x}, x, t\right\} \qquad (1)$$

$$t \in [0,\infty)$$
$$x \in [0,L]$$

and:
$$B_1(t) \cdot \frac{\partial \underline{u}(x,t)}{\partial x} + B_2(t) \cdot \underline{u}(x,t) = B_3(t)$$

or:
$$\frac{\partial \underline{u}(x,t)}{\partial t} = B_3(t) \qquad (2)$$

$$t \in [0,\infty)$$
$$x = \{0\}, \{L\}$$

where: $\dim\{\underline{u}\} = n$

The first equation describes the system and the second its boundary conditions.

Higher derivatives in the space dimension may be present as well but require additional statements for coding (subroutines PUPX and PUPXX). PDE's containing higher derivatives in the time domain may be reduced to sets of PDE's with only one derivative in the time domain. Multidimensional problems (more than one space variable) may also be coded. In such cases the upsetting of the boundary conditions may give sometimes troubles if the definition domain $D(\underline{x})$ is other than rectangular. Examples for coding of such problems are given in [3].

FORSIM-V solves the simulation problem by using the *method of lines*, in which the space dimension is discretized whereas the time-axis remains "continuous". Each equation in eq.(1) is reduced to a set of NNDIV coupled first order ODE's:

$$\begin{aligned} &\underline{u}(x,t) \rightarrow U(t) \\ &\underline{x} \rightarrow \underline{\xi} = \{1,2,\ldots,\text{NNDIV}\} \text{ (indices)} \\ &\dim\{\underline{u}\} = n \\ &\dim\{U\} = n \times \text{NNDIV} \end{aligned} \qquad (3)$$

leading to:

$$U_t(t) = F\{U(t), U_x(t), U_{xx}(t), \underline{\xi}, t\} \qquad (4)$$

The above defined problem is now reduced to two new problems:

a) computation of the space derivatives U_x and U_{xx} out of given values for U
b) computation of U out of given values for U_t.

For the numerical solution of both problems there exist well known algorithms.

FORSIM-V consists of:

a) a FORTRAN-IV written main program with subroutines for a couple of integration algorithms to which the user has to supply a subroutine (UPDATE) for the computation of the state derivatives according to eq.(4)
b) a FORTRAN-IV written subprogram library containing subroutines for:
   α) computation of the space derivatives
   β) output subroutines
   γ) function subprograms for comfortable modeling of often used functional blocks (hysteresis, step etc.) referred to as run-time library

The package is written to be used on CDC 6000-series installations, and since it does not follow ANSI standards, a remarkable effort would be necessary to adjust it to other installations.

From the above described architecture it can be seen that FORSIM-V is entirely procedural (no preprocessing). It is used to integrate sets of initial value ODE's over time. The subroutines (α) are not accessed by FORSIM itself but are used, if required, in the user-supplied subroutine UPDATE only.

This concept allows to use the subroutines (α) almost as they are also in a GASP program. Since GASP-V follows entirely ANSI standards they have, however, been recoded before they were added to the GASP library. Slight adjustments have been necessary for:

a) adjustment of the COMMON-blocks
b) adjustment for input of FORSIM variables according to GASP-philosophy which is different from FORSIM input
c) initialization of system variables.

Finally the following program structure is realized: The subroutine GASP (of GASP-V) computes the state variables out of their derivatives by numerical integration. The user-supplied subroutine STATE computes the state derivatives according to eq.(4) and calles the FORSIM-subroutines PARSET and PARFIN (and indirectly further FORSIM-subroutines: BOUND, PUPX, PUPXX) for computation of the space derivatives. This is illustrated in fig.1.

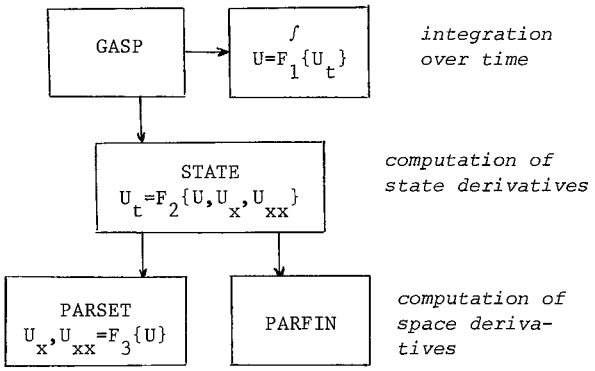

fig.1 Program structure for the simulation of distributed systems by use of GASP-V

Advantages of the program structure shown above are:

a) full upwards compatibility with GASP-IV is guaranteed

b) beside of the few statements for data input and initialization of FORSIM variables the requirement for core memory remains constant if a GASP-IV program is computed by use of GASP-V

c) the execution time for a GASP-IV program running under GASP-V is almost unchanged.

To illustrate the abilities of GASP-V let us consider as an example the central heating of a building. The building is modeled by a stick of length one. The left side of it represents the center of the building where the heating takes place, the right side of it represents the walls. The temperature distribution in the building is modeled using the diffusion equation eq.(5):

$$\frac{\partial u}{\partial t} = 0.5 \cdot \frac{\partial^2 u}{\partial x^2} \qquad (5)$$

The heating of the central room is modeled by a first order ODE of the following form:

$$u = 30.0 \cdot z$$

$$\text{where: } \dot{z} = 4.0 \cdot (SW - z) \qquad (6)$$

$$\text{and: } SW = \begin{cases} 1.0 & ; \text{ heating "on"} \\ 0.0 & ; \text{ heating "off"} \end{cases}$$

At night time (between 7 p.m. and 7 a.m.) the heating is always "off". During day time the heating is set "on", when the temperature of the walls falls below 19.5°C and is set "off" as soon as the temperature of the walls raises above 22.5°C. The temperature is said to start at 6 a.m. with 0.0°C across the whole building. The model of the building is shown in fig.2.

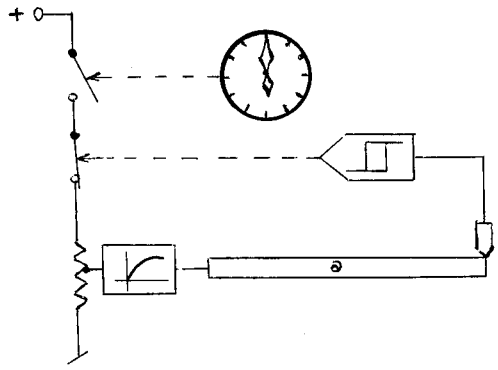

fig.2 Model of a central heating of a building

This model suffers from being far away from the physical reality, but has been chosen intentionally since it is the simplest model still containing all possible simulation elements (PDE's, ODE's, time-events and state-events).

The main program assigns (as usually in GASP-IV) the logical unit number 5 to the input device and the logical unit number 6 to the output device and finally makes a call to subroutine GASP.

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6
     1=OUTPUT,TAPE1)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      NCRDR = 5
      NPRNT = 6
      CALL GASP
      STOP
      END
```

Subroutine INTLC is used to initialize the non-GASP variables and to record the first time-event scheduled for one hour simulation time (≙ 7 a.m.) in the event file with event code 1 (≙ ATRIB(2)) corresponding to switching from night to day activity.

```
      SUBROUTINE INTLC
      DIMENSION  U(11)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
```

```
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /GCOM7/ NNDIF,NNDIV,XXL,XXU,AAUX,AAU
     1XX,NNEQX,DDX,XX(300)
      COMMON /GCOM8/ BL(3,10),BU(3,10),NL(10),NU(
     110)
      COMMON /UCOM1/ MSET,MMAX,STATEV,TIMEV,SW
      EQUIVALENCE (SS(1),U(1)),(SS(12),Z)
      LOGICAL AAUX
C
C*****SCHEDULING OF FIRST TIME-EVENT AT TIME
C*****T=1.0 WITH EVENT CODE 1
C
      ATRIB(1) = 1.0
      ATRIB(2) = 1.0
      CALL FILEM (1)
C
C*****UPSETTING OF BOUNDARY CONDITIONS AND
C*****INITIAL CONDITIONS FOR THE PDE
C
      BL(1,1) = 0.0
      BL(2,1) = 1.0
      BL(3,1) = 0.0
      NL(1) = 0
      NU(1) = -2
      MSET = 1
      MMAX = 11
      DO 1000  I=1,NNDIV
      U(I) = 0.0
 1000 CONTINUE
      AAUX = .FALSE.
C
C*****SETTING OF INITIAL CONDITIONS FOR THE ODE
C*****AND SETTING OF FLAGS (≙ SWITCHES)
C
      Z = 0.0
      TIMEV = 0.0
      STATEV = 1.0
      RETURN
      END
```

In subroutine STATE the state derivatives are computed for both the ODE and the PDE.

```
      SUBROUTINE STATE
      DIMENSION  U(11),UT(11),UX(11),UXX(11)
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /GCOM7/ NNDIF,NNDIV,XXL,XXU,AAUX,AAU
     1XX,NNEQX,DDX,XX(300)
      COMMON /GCOM8/ BL(3,10),BU(3,10),NL(10),NU(
     110)
      COMMON /UCOM1/ MSET,MMAX,STATEV,TIMEV,SW
      EQUIVALENCE (SS(1),U(1)),(SS(12),Z),(DD(1),
     1UT(1)),(DD(12),ZT)
C
C*****COMPUTATION OF THE SWITCH SW
C
      SW = TIMEV*STATEV
C
C*****COMPUTATION OF THE STATE DERIVATIVE FOR
C*****THE ODE
C
      ZT = 4.0*(SW - Z)
C
C*****COMPUTATION OF THE BOUNDARY CONDITION AT
C*****THE LEFT END OF THE PDE
C
      BL(3,1) = 30.0*Z
C
C*****COMPUTATION OF THE STATE DERIVATIVES FOR
C*****THE PDE
C
      CALL PARSET (MSET,MMAX,U,UT,UX,UXX)
      DO 1000  I=1,NNDIV
      UT(I) = 0.5*UXX(I)
 1000 CONTINUE
      CALL PARFIN (MSET,MMAX,U,UT,UX,UXX)
      RETURN
      END
```

Subroutine SCOND is used for detection of the two state conditions:

a) the temperature of the walls falls below 19.5°C

b) the temperature of the walls raises above 22.5°C

```
      SUBROUTINE SCOND
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      LFLAG(1) = KROSS (11,0,0.0,19.5,-1,0.5)
      LFLAG(2) = KROSS (11,0,0.0,22.5,+1,0.5)
      RETURN
      END
```

In subroutine EVNTS the required discrete events are carried out. The event code (IX) has the following meaning:

IX=1: Switching from night to day activity

IX=2: Switching from day to night activity

IX=3: State-event code (both state-events have the same event code and are distinguished by use of the LFLAG value)

```
      SUBROUTINE EVNTS (IX)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /UCOM1/ MSET,MMAX,STATEV,TIMEV,SW
C
C*****BRANCH TO APPROPRIATE EVENT
C
      GO TO (1,2,3), IX
C
C*****EVENT CODE 1: SET TIMEV TO BUSY AND
C*****SCHEDULE NEXT TIME EVENT FOR T=TNOW+12.0
```

```
C
    1 ATRIB(1) = ATRIB(1) + 12.0
      ATRIB(2) = 2.0
      CALL FILEM (1)
      TIMEV = 1.0
      GO TO 3
C
C*****EVENT CODE 2: SET TIMEV TO IDLE AND
C*****SCHEDULE NEXT TIME EVENT FOR T=TNOW+12.0
C
    2 ATRIB(1) = ATRIB(1) + 12.0
      ATRIB(2) = 1.0
      CALL FILEM (1)
      TIMEV = 0.0
C
C*****EVENT CODE 3: CHECK WHETHER A STATE-EVENT
C*****TYPE 1 OR TYPE 2 HAPPENED AND SET STATEV
C*****TO BUSY OR IDLE ACCORDINGLY.
C
    3 CONTINUE
      IF (LFLAG(1).NE.-1)  GO TO 4
      STATEV = 1.0
      RETURN
    4 IF (LFLAG(2).NE.+1)  RETURN
      STATEV = 0.0
      RETURN
      END
```

Subroutine SSAVE is used to store data for output documentation by calling the GASP-IV subroutine GPLOT and by calling the FORSIM-V subroutine RITER.

```
      SUBROUTINE SSAVE
      DIMENSION  U(11),PLOT(4)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /GCOM7/ NNDIF,NNDIV,XXL,XXU,AAUX,AAU
     1XX,NNEQX,DDX,XX(300)
      COMMON /UCOM1/ MSET,MMAX,STATEV,TIMEV,SW
      EQUIVALENCE (SS(1),U(1))
C
C*****PRODUCE PRINT PLOT
C
      PLOT(1) = SS(1)
      PLOT(2) = SS(11)
      PLOT(3) = 30.0*SW
      PLOT(4) = SS(12)
      CALL GPLOT (PLOT,TNOW,1)
C
C*****PRINT THE PDE
C
      DATA  COOR/4HCOOR/,UU/4H U  /
      CALL RITER (XX,COOR)
      CALL RITER (U,UU)
      RETURN
      END
```

By using GASP-IV to realize this example the user would have to code himself the algorithms for the computation of the space derivatives and for the upsetting of the boundary conditions which is done automatically in GASP-V by use of the subroutines PARSET and PARFIN. By using FORSIM-V the user would find it difficult to code the events in a proper way. To the best of the authors' knowledge GASP-V is the first package capable of handling such models in an easy and attractive form both from the side of coding and from the side of numerics.

3. FUNCTIONAL BLOCKS WITH DISCONTINUITIES

As it has been stated already in the introduction to this paper proper handling of functional blocks containing discontinuities requires breaking up the discontinuous models into a set of continuous models (branches). For illustration let us consider a hysteresis function (back-lash) as it is given in fig.3.

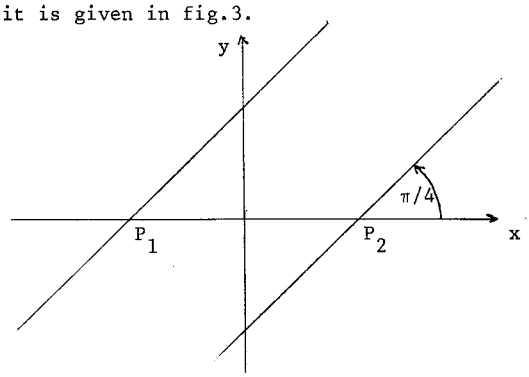

fig.3 Graph of a back-lash function

This function should be broken up into the two branches shown in fig.3 plus a horizontal branch which can be placed anywhere depending on the *history* of the function. We, therefore, have to distinguish between three different models:

Model 1:

$$y = y_L \tag{7}$$

where y_L means the last computated value for y. This model has to be selected for

$$y > x-P_2 \quad \cup \quad y < x-P_1 \tag{8}$$

and is valid until:

Event A: x crosses $y+P_2$ in positive direction
→ switching over to model 2.

Event B: x crosses $y+P_1$ in negative direction
→ switching over to model 3.

Model 2:

$$y = x - P_2 \tag{9}$$

This model has to be used when:

$$x \geq y_L + P_2 \qquad (10)$$

and is valid until:

Event C: $x < x_L$
→ switching over to model 1.

Model 3:

$$y = x - P_1 \qquad (11)$$

This model has to be utilized when:

$$x \leq y_L + P_1 \qquad (12)$$

and is valid until:

Event D: $x > x_L$
→ switching over to model 1.

Another example of proper spliting up of such functional blocks (for the description of a friction force) is given in [8].

It has been found a way to enable the user to code a hysteresis function (and most of the other functional blocks offered by a package like CSMP-III) in a very simple form when using GASP-V:

```
Y = HSTRSS (IX,YIC,P1,P2,TOL,ID)        (13)
```

where: IX describes the input function:

$$x = \begin{cases} SS(IX) & ; IX>0 \text{ (state variable)} \\ DD(-IX) & ; IX<0 \text{ (state deriv.)} \end{cases}$$

YIC initial condition for y

P1,P2 as described above P_1, P_2

TOL required accuracy for locating state events

ID identifier which is unique to each call

This utilization is syntactically almost equivalent to the use in CSMP-III. The TOL parameter is new (meaningless in CSMP-III, since no state-events exist). The ID parameter is necessary in CSMP-III as well, but there it is automatically added during preprocessing.

To show how this goal could be achieved, let us consider now the following system:

$$\dot{x}_1 = x_2 \qquad x_1(0) = 0$$
$$\dot{x}_2 = -x_1 \qquad x_2(0) = 1$$
$$y_1 = \text{HSTRSS } (1,0.0,-0.5,0.5,10^{-3},1)$$
$$y_2 = \text{HSTRSS } (2,0.5,-0.5,0.5,10^{-3},2)$$
$$y_3 = \text{STEP } (5.0, 3)$$

The system described by x_1 and x_2 results in a sin- and a cos-function. y_1 and y_2 are the output functions of a back-lash with sin-input. y_3 is equal to zero until t=5 and then jumps to one.

Below the appropriate STATE-subroutine for GASP-V is given:

```
      SUBROUTINE STATE
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      DD(1) = SS(2)
      DD(2) = -SS(1)
      SS(3) = HSTRSS (1,0.0,-0.5,0.5,1.E-3,1)
      SS(4) = HSTRSS (2,0.5,-0.5,0.5,1.E-3,2)
      SS(5) = STEP (5.0,3)
      RETURN
      END
```

For illustration how this goal could be achieved, below the GASP-IV listing of the HSTRSS and STEP function and of the subroutines SCOND and EVNTS are given:

```
      FUNCTION HSTRSS (IX,YIC,P1,P2,TOL,ID)
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /GCOM12/ XXMEM(200),IISTRT(50),IITYP
     1E(50),IIDENT(50),IITRY,IIATR,IIPNT,IIUPD,TN
     2OWL
      DIMENSION  IIMEM(200)
      EQUIVALENCE (XXMEM(1),IIMEM(1))
      IF (IIPNT) 10000,2,1
    1 DO 1000  J=1,IIPNT
      IF (IIDENT(J).NE.ID)  GO TO 1000
      I = IISTRT(J)
      GO TO 5
 1000 CONTINUE
    2 IIPNT = IIPNT + 1
      IITYPE(IIPNT) = 1
      IISTRT(IIPNT) = IIUPD
      I = IIUPD
      IIUPD = IIUPD + 11
      IIDENT(IIPNT) = ID
      XXMEM(I+2) = YIC
      IIMEM(I+8) = 1
      IF (IX) 3,10001,4
    3 I1 = -IX
      XXMEM(I+6) = DD(I1)
      XXMEM(I+7) = DD(I1)
      GO TO 5
    4 XXMEM(I+6) = SS(IX)
      XXMEM(I+7) = SS(IX)
    5 IIMEM(I+1) = IX
      XXMEM(I+3) = P1
      XXMEM(I+4) = P2
      XXMEM(I+5) = TOL
      IM = IIMEM(I+8)
      IF (IM-2) 6,7,10
    6 XXMEM(I) = XXMEM(I+2)
```

```
      GO TO 13
    7 IF (IX) 8,10001,9
    8 I1 = -IX
      XXMEM(I) = DD(I1) - P2
      GO TO 13
    9 XXMEM(I) = SS(IX) - P2
      GO TO 13
   10 IF (IX) 11,10001,12
   11 I1 = -IX
      XXMEM(I) = DD(I1) - P1
      GO TO 13
   12 XXMEM(I) = SS(IX) - P1
   13 HSTRSS = XXMEM(I)
      RETURN
10000 CALL ERROR (901)
      RETURN
10001 CALL ERROR (902)
      RETURN
      END

      FUNCTION STEP (P,ID)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM12/ XXMEM(200),IISTRT(50),IITYP
     1E(50),IIDENT(50),IITRY,IIATR,IIPNT,IIUPD,TN
     2OWL
      DIMENSION  IIMEM(200)
      EQUIVALENCE (XXMEM(1),IIMEM(1))
      IF (IIPNT) 10000,2,1
    1 DO 1000  J=1,IIPNT
      IF (IIDENT(J).NE.ID)  GO TO 1000
      I = IISTRT(J)
      GO TO 4
 1000 CONTINUE
    2 IIPNT = IIPNT + 1
      IITYPE(IIPNT) = 2
      IISTRT(IIPNT) = IIUPD
      I = IIUPD
      IIUPD = IIUPD + 1
      IIDENT(IIPNT) = ID
      IITRY = IITRY + 1
      IIATR = MAXO (IIATR,3)
      IF (P.GT.TTBEG)  GO TO 3
      XXMEM(I) = 1.0
      GO TO 4
    3 XXMEM(I) = 0.0
      ATRIB(1) = P
      ATRIB(2) = 2.
      ATRIB(3) = FLOAT (I) + 0.5
      CALL FILEM (1)
    4 STEP = XXMEM(I)
      RETURN
10000 CALL ERROR (901)
      RETURN
      END

      SUBROUTINE SCOND
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW
     1,ISEES,LFLAG(50),NFLAG,NNEQD,NNEQS,NNEQT,SS
     2(100),SSL(100),TTNEX
      COMMON /GCOM12/ XXMEM(200),IISTRT(50),IITYP
     1E(50),IIDENT(50),IITRY,IIATR,IIPNT,IIUPD,TN
     2OWL
      DIMENSION  IIMEM(200)
      EQUIVALENCE (XXMEM(1),IIMEM(1))
      IF (NNEQT.GE.100)  GO TO 10002
      IF (IIPNT.EQ.0)  RETURN
      DO 1000  J=1,IIPNT
      I = IISTRT(J)
      IT = IITYPE(J)
      GO TO (1,1000), IT
    1 IM = IIMEM(I+8)
      IF (IM-2) 2,3,4
    2 IIMEM(I+9) = KROSS (IIMEM(I+1),0,0.0,XXMEM(
     1I+2)+XXMEM(I+4),+1,XXMEM(I+5))
      IIMEM(I+10) = KROSS (IIMEM(I+1),0,0.0,XXMEM
     1(I+2)+XXMEM(I+3),-1,XXMEM(I+5))
      IX = IIMEM(I+1)
      GO TO 1000
    3 SS(100) = XXMEM(I+6)
      SSL(100) = XXMEM(I+7)
      IIMEM(I+9) = KROSS (IIMEM(I+1),100,1.0,0.0,
     1-1,XXMEM(I+5))
      IX = IIMEM(I+1)
      GO TO 1000
    4 SS(100) = XXMEM(I+6)
      SSL(100) = XXMEM(I+7)
      IIMEM(I+9) = KROSS (IIMEM(I+1),100,1.0,0.0,
     1+1,XXMEM(I+5))
      IX = IIMEM(I+1)
 1000 CONTINUE
      IF ((ISEES.NE.0).OR.(TNOW.LE.TNOWL)) RETURN
      TNOWL = TNOW
      DO 1001  J=1,IIPNT
      I = IISTRT(J)
      IT = IITYPE(J)
      GO TO (5,1001), IT
    5 XXMEM(I+2) = XXMEM(I)
      XXMEM(I+7) = XXMEM(I+6)
      IX = IIMEM(I+1)
      IF (IX) 6,10001,7
    6 I1 = -IX
      XXMEM(I+6) = DD(I1)
      GO TO 1001
    7 XXMEM(I+6) = SS(IX)
 1001 CONTINUE
      RETURN
10001 CALL ERROR (902)
      RETURN
10002 CALL ERROR (903)
      RETURN
      END

      SUBROUTINE EVNTS (IX)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100)
     1,MLE(100),MSTOP,NCRDR,NNAPO,NNAPT,NNATR,NNF
     2IL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TT
     3BEG,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM12/ XXMEM(200),IISTRT(50),IITYP
     1E(50),IIDENT(50),IITRY,IIATR,IIPNT,IIUPD,TN
```

```
      20WL
       DIMENSION  IIMEM(200)
       EQUIVALENCE (XXMEM(1),IIMEM(1))
       TNOWL = TNOW
       GO TO (1,7), IX
    1  DO 1000  J=1,IIPNT
       I = IISTRT(J)
       IT = IITYPE(J)
       GO TO (2,1000), IT
    2  IM = IIMEM(I+8)
       IF (IM-2) 3,5,6
    3  IF (IIMEM(I+9).NE.1)  GO TO 4
       IIMEM(I+9) = 0
       IIMEM(I+8) = 2
       GO TO 1000
    4  IF (IIMEM(I+10).NE.-1)  GO TO 1000
       IIMEM(I+10) = 0
       IIMEM(I+8) = 3
       GO TO 1000
    5  IF (IIMEM(I+9).NE.-1)  GO TO 1000
       IIMEM(I+9) = 0
       IIMEM(I+8) = 1
       GO TO 1000
    6  IF (IIMEM(I+9).NE.1)  GO TO 1000
       IIMEM(I+9) = 0
       IIMEM(I+8) = 1
 1000  CONTINUE
       RETURN
    7  I = IFIX (ATRIB(3))
       XXMEM(I) = 1.0
       RETURN
       END
```

In GASP-IV the user would have to code all the subprograms coded above. In GASP-V there exists in parallel with the user-supplied subroutine SCOND a system-supplied subroutine GCOND in which state conditions resulting from the use of such functional blocks are evaluated. Time-events are recorded by use of the ordinary file handling system of GASP-IV but showing negative event codes (not used in GASP-IV). If a time-event with negative event code is realized or if a state-event is detected in GCOND the system-supplied subroutine GEVNT is called (instead of EVNTS) for accomplishment of the event. In GEVNT all event handling resulting from the use of such functional blocks is coded.

By introducing this feature both core memory requirement and execution time of a GASP-IV program raise by not more than 5 ÷ 10 % (subroutines GCOND and GEVNT). This can be avoided by simply not loading the two subroutines. Another possibility would be to keep both subroutines GASP under two different names (e.g. GASP and GASP5) in the library. GASP-IV programs may then be executed by calling GASP, whereas GASP-V programs require a call to subroutine GASP5 instead.

4. THE INTEGRATION

In GASP-IV a Runge-Kutta algorithm of fourth order with step size computation is coded directly in subroutine GASP. For this reason serious modifications are required if another algorithm should be used instead.

It can be assumed that in most cases at least some discrete events will take place in every program, since for pure continuous simulation problems other packages are easier to apply and are more efficient than GASP for equivalent results. It can be said in general that for integration of systems in which discontinuities exist, always *one step algorithms* should be used, since *multi-step algorithms* always have to be restarted when a discrete event is realized. For most applications, therefore, the Runge-Kutta algorithm will be almost optimal and will give satisfactory results. The Runge-Kutta algorithm will then be no more optimal when a system is very *stiff*.

Let us consider now a PDE of form eq.(4). For the k-th u(t) called $u_k(t)$ it can be written:

$$u_k(t) = 0.5 \cdot \{u_{k-1}(t) + u_{k+1}(t)\} + o(\Delta x) \quad (14)$$

For Δx being sufficiently small the above equation describes an almost linear relationship between neighbouring functions, which implies that the eigenvalues of this system will be wide spread. A system of ODE's resulting from the discretization of a PDE will, therefore, almost by definition form a stiff system.

For this reason the most commonly used algorithm in FORSIM-V is the Hindmarsh-implementation of the Gear algorithm [9,10]. This algorithm, however, is a multistep algorithm and, therefore, not optimally well suited for proper handling of discontinuities.

According to the authors' opinion the best suited "classical" algorithm in this case would be the *tangent rule*, which is an implicit one step algorithm, with a bit of interpolation for raising the order. Such an algorithm is described in [11, 12]. It is, however, very well possible that an even more optimal algorithm for this purpose still could be developed.

To facilitate the implementation of new algorithms the subroutine GASP is entirely rewritten to allow easy implementation of new algorithms. Implementation of a Gear-algorithm and of a tangent rule has been considered but not yet realized.

5. NUMERICAL RESULTS AND EFFICIENCY CONSIDERATIONS

To discuss the efficiency of GASP-V let us come back once more to the heat diffusion problem. Fig.4 shows the GASP-V echo check of the input data for this problem. The meaning of all variables is identical to GASP-IV (cf. [1,2]) except for NNPDE denoting the number of PDE's in the system, NNDIF being the number of points used for the computation of the spatial derivatives, NNDIV corresponding to the number of spatial divisions, NNX indicating whether equal or unequal divisions have been selected, XXL pointing to the lower bound in space and XXU fixing the upper bound correspondingly. Fig.5 shows a part of the graphical output showing the temperature at the lower and at the upper bound versus time. At initialization the temperature in the whole building is equal to the outside temperature ($0^{\circ}C$). After one hour the left side temperature is controlled to reach $30^{\circ}C$. This strategy continues until the temperature at the upper border raises to $22.5^{\circ}C$. Then the control low is changed to cool the building down again to $0^{\circ}C$ (bang-bang control) until the temperature at the upper bound falls below $19.5^{\circ}C$. This bang-bang control continues until 13 hours of the simulation time have passed. Then the control low is changed again to cool the building down to $0^{\circ}C$, independently from the temperature at the upper limit in space.

To execute this problem the CDC 6500 computer needed 65000 octal words of core memory and 66.7 sec. of CPU-time (translation included), corresponding to 43.-- sfr. of total costs for the job. A comparison with other programs for this problem was not possible, since no other program as far as it is known to us is capable of handling the problem in a comparable way.

As an example of handling discontinuities a power electronic convertor circuit has been evaluated. In this example a current is controlled to follow a sinus-wave in a chopped manner over one period. The original program coded in CSMP-6000 costed about 800.-- sfr. to be executed on our CDC 6500 installation. A corresponding GASP-IV program costed 45.-- sfr. and an equivalent GASP-V program costed 32.-- sfr. both computed on the same installation. The two GASP programs are much more accurate since the discontinuities are located properly. The GASP-V program was somewhat faster than the GASP-IV program due to different algorithms used for the integration, the error estimation, the step size control and the iteration of state events. The much higher costs of the CSMP program result from the effect of creeping described above. All other known parallel packages would show similar effects for the same reason.

It is clear that a parallel language could be developed in which the user has the possibility to define discrete events. The preprocessor of this package would have to translate the parallel code into a target language which is similar to GASP. Such a package, however, does not exist yet.

```
                    SIMULATION PROJECT NUMBER   3  BY      BLITZ

                    DATE 11/ 20/ 1975       RUN NUMBER    1 OF    1
                    LLSUP=000000000000000   GASP V  VERSION 300KT75

NNCLT=     0    NNSTA=     0    NNHIS=     0    NNPRM=     0    NNPLT=     1    NNSTR=     1    NNTRY=     5
NNATR=     2    NNFIL=     1    NNSET=    20    NNEQD=    12    NNEQS=     2    NFLAG=     2    NNPDE=     1

NNDIF=     3    NNDIV=    11    NNX  =    -0
XXL  =    -0.                   XXU  =      .1000E+01

GPLOT NO.  1    LLABP=TIME       IITAP=     2    NNVAR=     4    LLPLT=     2          DTPLT=      .2500E+00
  VAR NO.  1        A=TEMPLEFT   LLPLO=    -0.   LLPHI=    -0   PPLO =    -0.         PPHI =     -0.
  VAR NO.  2        B=TEMPRIGT   LLPLO=    -0    LLPHI=    -0   PPLO =    -0.         PPHI =     -0.
  VAR NO.  3        C=TEMPSORC   LLPLO=    -0    LLPHI=    -0   PPLO =    -0.         PPHI =     -0.
  VAR NO.  4        D=SWITCH     LLPLO=    -0    LLPHI=    -0   PPLO =    -0.         PPHI =     -0.

KKRNK= (   1)

IINN = (   1)

IIEVT=     3    LLERR=     0    AAERR=      .1000E-01          RRERR=      .1000E-01
DTMIN=      .1000E-01           DTMAX=      .2500E+00          DTSAV=      .2500E+00

MSTOP=     1    JJCLR=     0    JJBEG=     1    IICRD=     0    TTBEG=     0.          TTFIN=      .3600E+02
JJFIL=     1
IISED=    -0
PARTIAL DIFFERENTIAL EQUATION  1 WILL BE DISCRETISED USING  3 POINT FORMULAE AND 11 EQU. SPATIAL DIVISIONS
```

fig.4 GASP-V echo check of input data for the heat diffusion problem

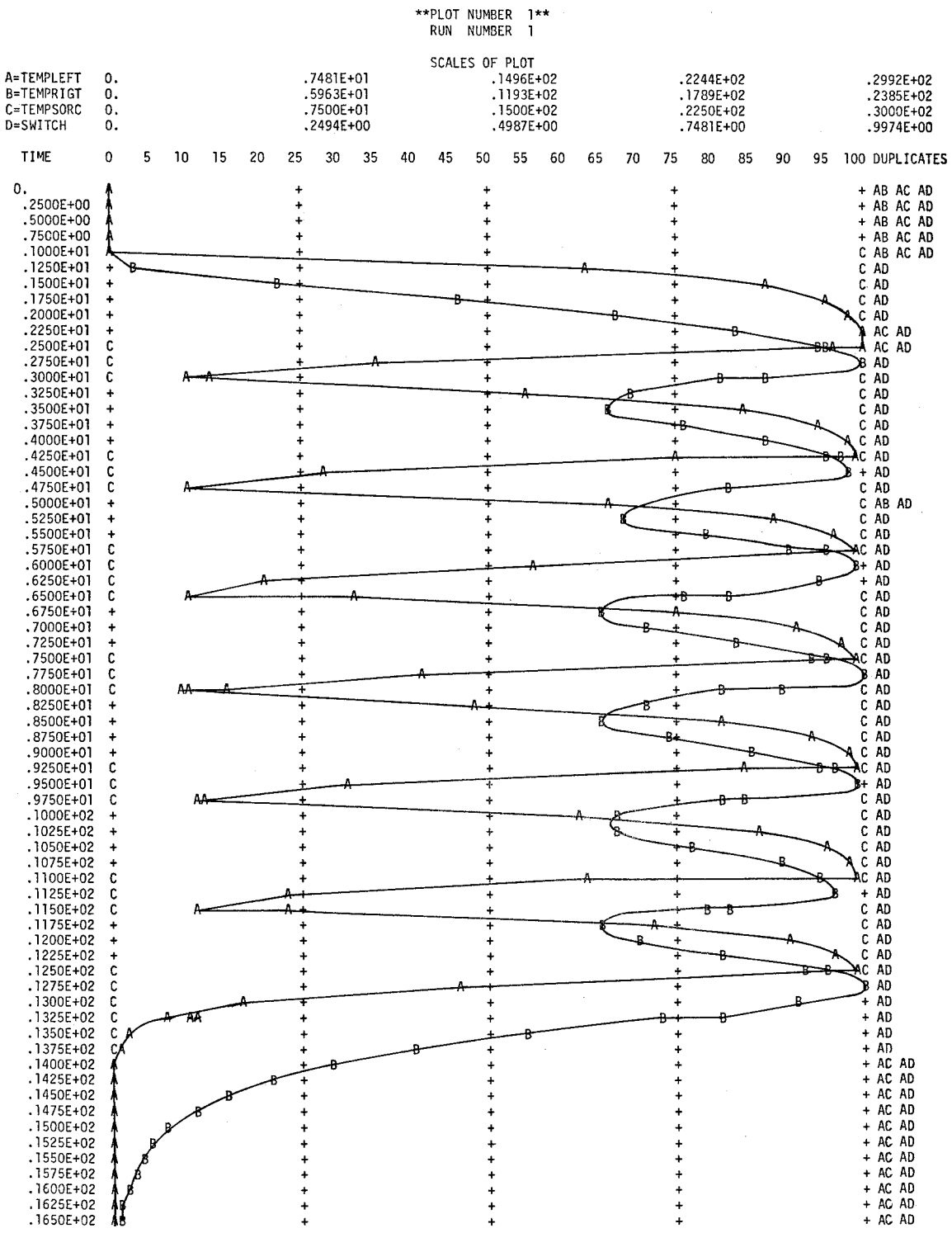

fig.5 Graphical output of the heat diffusion problem

ACKNOWLEDGMENT

The authors would like to state that GASP-V is not a replacement for GASP-IV, but only an enlargement of it with the aim of making it more attractive for use in continuous simulation studies. As a bulb is a quite useless thing if there is no lamp around to use it in, GASP-V would not exist without the tremendous and most valuable work of Prof.A.Pritsker (School of Industrial Engineering, Purdue University, Lafayette Ind., USA) who developed the GASP-IV system with its marvellous architecture and documentation and of Dr.M.Carver (Atomic Energy of Canada Ltd., Chalk River, Ontario, Canada) who developed the FORSIM-V program. Their work is highly appreciated. The GASP-V manual -- which is not yet available -- will contain only those features of GASP-V not described in [1]. The documentation of GASP-IV is so clear that there is no use in repeating something which anyway could not be improved. Potential users of GASP-V are required first to order the GASP-IV package from Prof.Pritsker (available at a nominal cost) and then to obtain from us a replacement for the subroutines GASP and DATIN together with a set of additional subroutines for the new features described in this paper. These programs are available for a nominal cost, too.

Beside that we wish to express our gratitude towards Prof.Dr.M.Mansour who is heading our institute at the Swiss Federal Institute of Technology Zurich and who gave us the opportunity to carry out the research described in this paper. Likewise we wish to thank AGIE Ltd., Losone/TI Switzerland for their support of this work.

REFERENCES

[1] A.A.B.Pritsker: *The GASP-IV Simulation Language*. John Wiley 1974

[2] A.A.B.Pritsker: *The GASP-IV User's Manual*. Pritsker & Associates Inc., 1201 Wiley Drive West Lafayette, Ind. 47906 USA 1974

[3] M.B.Carver: *FORSIM: A FORTRAN Package for the Automated Solution of Coupled Partial and/or Ordinary Differential Equation Systems. User's Manual*. Atomic Energy of Canada Ltd., Chalk River Nuclear Laboratories, Chalk River, Ontario, Canada November 1974

[4] F.E.Cellier: *Continuous-System Simulation by Use of Digital Computers: A State-of-the-Art Survey and Prospectives for Development*. Proceedings: Simulation'75, Zurich, Switzerland 1975

[5] R.N.Nilsen, Karplus W.J.: *Continuous-System Simulation Languages: A State-of-the-Art Survey*. Annales de l'Association Internationale pour le Calcul Analogique, Nr.1, January 1974

[6] M.B.Carver: *Simulation Packages for the Solution of Partial Differential Equation Systems*. Proceedings: Simulation'75, Zurich Switzerland 1975

[7] A.E.Blitz: *Entwicklung eines universell verwendbaren Simulationspaketes* (german). Diplomarbeit (9-th term) WS 1975/76. Internal Report: AIE 8263, Institute for Automatic Control, The Swiss Federal Institute of Technology Zurich, Physikstr. 3, CH-8006 Zurich, Switzerland.

[8] F.E.Cellier, Rufer D.F.: *Algorithm Suited for the Solution of Initial Value Problems in Engineering Applications*. Proceedings: Simulation'75, Zurich, Switzerland 1975.

[9] C.W.Gear: *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall Series in Automatic Computation 1971.

[10] A.C.Hindmarsh, Gear C.W.: *Ordinary Differential Equation System Solver*. Lawrence Livermore Laboratory. Report: UCID 30001 Rev.2, August 1972.

[11] B.Lindberg: *IMPEX - A Program for Solution of Systems of Stiff Differential Equations*. The Royal Institute of Technology Stockholm Sweden. Report: NA 72.50 1972.

[12] B.Lindberg: *IMPEX 2 - A Procedure for Solution of Systems of Stiff Differential Equations*. The Royal Institute of Technology Stockholm Sweden. Report: TRITA-NA-7303 1973.