ALGORITHM SUITED FOR THE SOLUTION OF INITIAL VALUE PROBLEMS IN ENGINEERING APPLICATIONS.

François E. Cellier Daniel F. Rufer

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

I) ABSTRACT

It is the scope of this paper to describe the special demands of integration routines for handling initial value problems in engineering applications. Besides the pure integration algorithm all the other activities of the routine which can be called the "interface" between the problem and the integration algorithm have been taken into consideration. The requirements needed for correct processing of discontinuities (time events, state events) and for the proper treatment of memory- and history-functions are discussed in seperate sections. As an example an electro-mechanical system is discussed. It is shown that an algorithm capable to process this small system requires most of the attributes described in this article.

II) INTRODUCTION

A routine for integration of technical processes is described. Models of such processes often involve discontinuities which require special algorithms for proper handling since numerical integration procedures are only defined for differential equations with coefficients which are continuous and have continuous derivatives. One step methods, however, may handle discontinuities if they lie at the end of a step whereas *multistep methods* have to be restarted at each instant of time a discontinuity takes place [1]. Discontinuities may occur at prescribed values of time. In this case they are called *time events*. On the other hand they may be released by a state variable crossing a given level or another state variable. In the latter case they are called *state events*. Time events are easy to process by setting the final time of integration to this event time. State events require an iteration algorithm which evaluates the unknown event time.

The problem can be presented in the following form: Dynamical systems can be described by a set of *state equations* ($\underline{x}\in\mathbb{R}^n$: vector of state variables, $\underline{u}\in\mathbb{R}^m$: vector of control variables)

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u},t) \quad ; \quad \underline{x}(t_i) = \underline{x}_o \qquad (2.1)$$

The integration starts at time t_i and ends at time t_f which may be prescribed or may be the first time where one of the termination conditions -- often referred to as state conditions -- is satisfied:

$$s_j(\underline{x}(t_f),t_f) = 0 \quad ; \quad j\in\{1,\ldots,p\} \qquad (2.2)$$

E.g. these termination conditions may be of the following special types:

$$t - t_f = 0 \qquad (2.3)$$

$$x_k - l_k = 0 \qquad (2.4)$$

A termination condition of type (2.3) describes a time event and, therefore, requires no iteration. (2.4) describes state events and may be iterated by an algorithm using the derivative $\dot{x}_k$ of x_k (e.g. Newton-Raphson algorithm), since $\dot{x}_k$ is available anyhow, whereas the general type (2.2) should be iterated by an algorithm which only requires information on $\underline{x}$ (e.g. Secant algorithm).

Models of technical processes often involve functions with a memory. Functions may be classified into three groups:

$$y(t) = f(\underline{x}(t)) \qquad (2.5)$$

$$y(t) = f(\underline{x}(t-\Delta t),\underline{x}(t-2\Delta t),\ldots,\underline{x}(t-k\Delta t)) \qquad (2.6)$$

$$y(t) = f(\underline{x}(t),\underline{x}(t-\Delta t),\ldots,\underline{x}(t-k\Delta t)) \qquad (2.7)$$

Functions of the type (2.5) are functions which at time t assign to an output variable y a value which only depends on the input $\underline{x}$ at the same time t. No memory is necessary for accomplishing such functions. Functions of the type (2.6) are called *memory-functions*. The output y depends only on values of $\underline{x}$ at past instants of time. Examples of this class of functions are the time-delay and the explicit integration algorithms. Functions of the third class (2.7) are called *history-functions*. The output y depends on values of $\underline{x}$ at past instants of time as well as on the present time t. Examples are the hysteresis function and the implicit integration algorithms [1]. Most of the memory- and history-functions also require special treatment.

III) DESCRIPTION OF THE ROUTINES

FORTRAN-IV written subroutines have been developed for the simulation of technical systems. These subroutines properly process all kind of discontinuities (discrete events) and also process memory- and history-functions. This section describes the attributes of the subroutines and how the routines are to be used. The following routines have been coded:

a) Runge-Kutta 5th and 8th order algorithm with step size calculation according to the method of Fehlberg [2,3]. This subroutine is an enlarged version of an earlier written subroutine by J. Waldvogel, to which a sophisticated "interface" for the treatment of engineering problems has been added.

b) Euler method with fixed step size.

c) Gear algorithm (DIFSUB) for integration of stiff systems [2] with variable step size and variable order combined with Adams Predictor-Corrector method of variable step size and variable order.

The system has to be modeled as a set of n state equations in a user supplied subroutine (DER) which for calculation of the derivatives is called several times during each time step. The termination conditions may be coded:

a) in a subroutine (TCOND) which is executed once every time step:

$$s_j = s_j(\underline{x},t) \quad ; \quad j = 1,\ldots,p$$

$$I_j = \begin{cases} 0 & ; \text{Secant algorithm is used for iteration of } s_j=0 \\ k\in[1,n] & ; \text{Newton-Raphson algorithm is used for iteration of } s_j=0 \text{ where } s_j=x_k-l_k \end{cases}$$

$\underline{I}$ is an associated integer vector to specify the iteration procedure to be used (Secant- or Newton-Raphson algorithm). The Newton-Raphson algorithm is only applicable if

$$s_j(\underline{x},t) = x_k - l_k \qquad (3.1)$$

b) in the integration algorithm itself, if only one termination condition has been specified which is of type (3.1). In this case k and l_k can be entered into the integration routine as parameters. The execution will be somewhat faster since the subroutine where normally the termination criteria are specified has not to be called.

c) in the integration algorithm itself for termination conditions of type (2.3) by setting a final time parameter.

For sampling and storing the integrated values $\underline{x}(t)$ a further user supplied subroutine (USER) is called at:

a) time t_i

b) time t_f (prescribed or iterated)

c) each *sampling event* time. A sampling event is a special time event which occurs periodically at times $k \cdot T_s$ (k=1,2,...). T_s normally is referred to as communication interval. Since no discontinuity is associated with this event, integration goes on.

Before calling the integration subroutine (INT) for the first time the user has to call once an initialization routine (INTIC) for initializing the parameters of the integration algorithm.

The program therefore has the following structure:

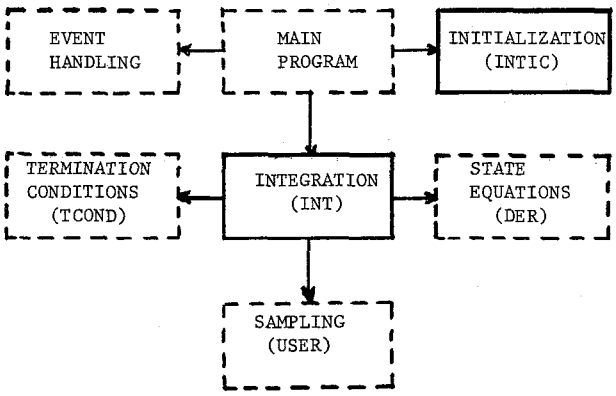

Fig. 3 Program structure

In the main program the initialization routine is called first. Then the integration routine is called performing integration of the system described in the state equation routine from time t_i until the prescribed final time t_f is reached or until one of the termination conditions has been realized. Then the control is given back to the main program which calls an event handling routine to perform the discrete event associated with the realized state condition. Then the integration routine is executed again until the final time is reached.

IV) HANDLING OF STATE EVENTS

Two algorithms have been implemented: The Newton-Raphson algorithm and the "safe" Secant algorithm [4]. The *"safe" Secant algorithm* always keeps one end point of the considered interval beyond and one below the zero line and, therefore, guarantees convergence.

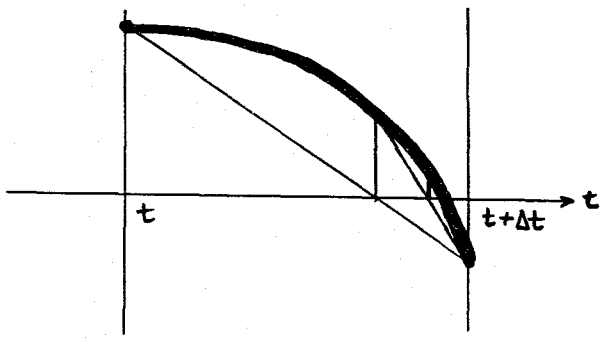

Fig. 4.1 Secant algorithm

The increment Δt never changes sign. If more than one termination condition s_i changes sign during one time step, Δt is reduced according to the following rule in order to determine the significant termination condition, that is the termination condition which is realized first:

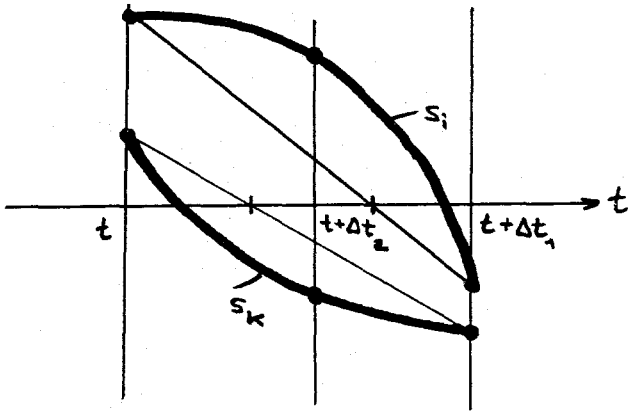

Fig. 4.2 Step size reduction

The points of the active termination conditions (in this case s_i and s_k) at times t and (t+Δt) are connected by straight lines. The arithmetic mean value of all the crossings with the time axis forms the new end point of the next time step. The procedure is repeated if still more than one state condition is active in the reduced time interval. The algorithm described here is a generalization of the Secant algorithm as can be seen from Fig. 4.2.

The *Newton-Raphson algorithm* does not guarantee convergence. If it diverges when started at time (t+Δt) the implemented algorithm will try to obtain convergence by starting at time t. If this also fails the program automatically switches to the Secant rule for postiteration. This is shown in Fig. 4.3.

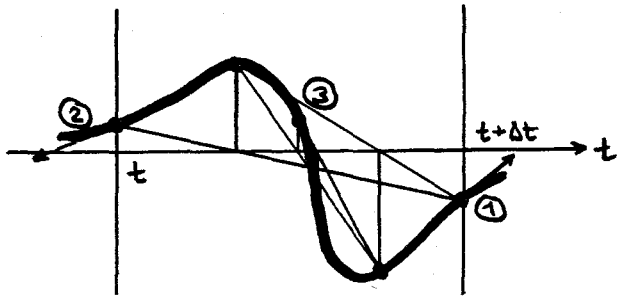

Fig. 4.3 Postiteration

The Newton-Raphson algorithm shows divergence if applied at time (t+Δt) (as shown in Fig. 4.3: point 1), since the solution is known to lie in the time interval [t,t+Δt]. It also shows divergence applied at time t (Fig. 4.3: point 2). Since no convergence can be obtained by utilizing Newton-Raphson the program switches to the Secant algorithm (Fig. 4.3: point 3) which guarantees for convergence.

The coding of termination conditions has already been explained in the last section. It has been shown that in general the Secant rule has to be utilized, since the Newton-raphson algorithm is only applicable for termination conditions of the specific type (2.4). Such termination conditions, however, are rather common since other classes of termination conditions may often be transformed into this type as shown below.

Given a system

$$\left.\begin{aligned}\dot{x}_i &= f_i(\underline{x},\underline{u},t)\\ x_i(t_i) &= x_{o_i}\end{aligned}\right\}; \; i = 1,\ldots,n \qquad \underline{x}=(x_1,\ldots,x_n)'$$

and a set of termination conditions

$$\begin{aligned}s_j &= s_j(\underline{x},t)\\ I_j &= 0 \qquad ; \; j = 1,\ldots,p\end{aligned}$$

where

$$\frac{d}{dt}(s_j(\underline{x},t))$$

exists.

This problem can be transformed to:

$$\begin{aligned}\dot{x}_i &= f_i(\underline{x},\underline{u},t) \quad ; \; i=1,\ldots,n\\ \dot{x}_{n+j} &= \frac{d}{dt}(s_j(\underline{x},t)) \quad ; \; j=1,\ldots,p\\ x_i(t_i) &= x_{o_i} \quad ; \; i=1,\ldots,n \qquad \underline{x}=(x_1,\ldots,x_{n+p})'\\ x_{n+j}(t_i) &= s_j(\underline{x}(t_i),t_i) \; ; \; j=1,\ldots,p\end{aligned}$$

with the termination conditions

$$\left.\begin{aligned}s_j &= x_{n+j}\\ I_j &= n+j\end{aligned}\right\}; \; j = 1,\ldots,p$$

The order of the system is increased by p which slows down the speed of integration but accelerates the speed of iteration, since now the Newton-Raphson algorithm can be used. This may pay out in cases where a system of high order involves many switching activities.

V) HANDLING OF MEMORY- AND HISTORY-FUNCTIONS

The difficulties arising if memory and/or history functions form part of the system to be simulated are demonstrated by a short example.

Let us consider a mechanical system with a gear with backlash characteristics. The position of the driving part is ϑ_1 (state variable) whereas the position of the driven part is ϑ_2. At time $t = t_i$ both positions ϑ_1 and ϑ_2 are assumed to be equal to zero:

$$\begin{aligned}\vartheta_1(t_i) &= 0\\ \vartheta_2(t_i) &= 0\end{aligned}$$

with the meaning that ϑ_2 is in the middle of the backlash zone. At time t_i the motor starts to rotate ($\vartheta_1(t)>0 \leftarrow (t>t_i)$). This will not effect ϑ_2 until $\vartheta_1=\vartheta_1^*$ where the motor becomes loaded. Fig. 5.1 illustrates the interaction between ϑ_1 and ϑ_2. At this instant ($\vartheta_1=\vartheta_1^*$) the first state event takes place since

a) the model changes (the load becomes involved)

b) the state variable ϑ_1 has no continuous derivatives.

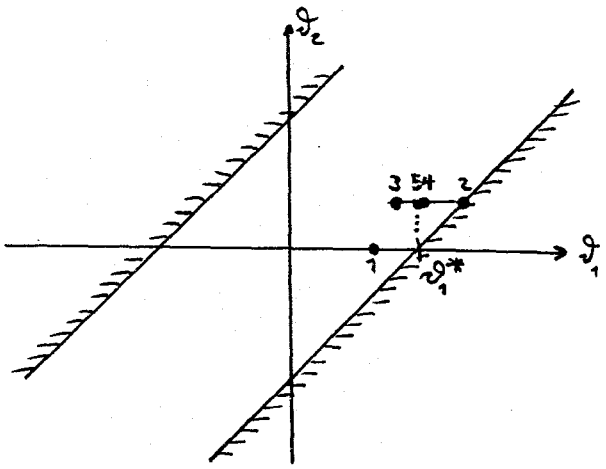

Fig. 5.1 Iteration of a system with backlash

The termination condition can be formulated as:

$$\begin{aligned}s_1 &= \vartheta_1 - \vartheta_1^*\\ I_1 &= 1\end{aligned}$$

In the example shown in Fig. 5.1 the termination condition s_1 changes sign during the second time step. The Newton-Raphson algorithm is activated and changes the direction of the integration. It can easily be seen that the algorithm converges (ϑ_1 obtains the value ϑ_1^*). However, the other variable ϑ_2 obtains the wrong value (the correct value would be zero). In this specific example it would help to split up the problem into three models, one for motor disengaged and two for motor engaged with rotation positive and negative respectively. In this case the use of history functions could be omitted. In general one has to use the Secant algorithm for iteration as soon as there are memory- and/or history-functions involved. By use of the Secant rule ϑ_2 would have obtained the right value in the above example, since the direction of the integration is never changed.

Besides updating of the memories should only be done at successfully accomplished time steps and not during intermediate computations, since

a) steps may be refused due to intolerable integration errors (in case of variable step size methods)

b) some of the higher order algorithms do not increase the time monotonously for the intermediate computations. This is illustrated in Fig. 5.2 for the case of the Runge-Kutta method of 8th order.

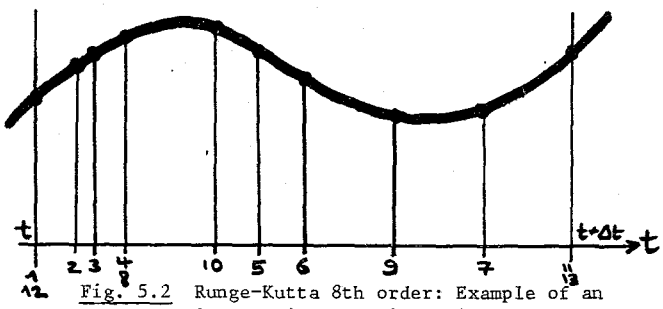

Fig. 5.2 Runge-Kutta 8th order: Example of an integration algorithm with non monotonic increase of time

For this purpose a variable (KEEP) has been made available in the integration routine. It has the following meaning:

$$\text{KEEP} = \begin{cases} -1: & \text{The time has been set back for repeating the last step with smaller step size } \Delta t \\ 0: & \text{intermediate computation of the state equations} \\ 1: & \text{a step has been successfully completed} \end{cases}$$

Memory- and history-functions must only be updated for computations with KEEP = 1.

VI) NUMERICAL EXAMPLE [5]

We consider a cart which can be moved on a 4 meter long, horizontal rail. By means of a DC motor and a steel transmission tape the vehicle can be accelerated in both directions (Fig. 6.1).

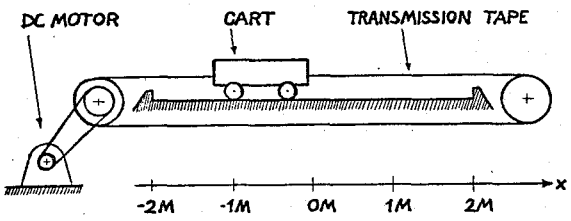

FIG. 6.1 ELECTRO MECHANICAL SYSTEM

This electro-mechanical system can be described by the following equations:

$$m \cdot \ddot{x}(t) = F(t) - F_f(\dot{x}(t), F(t)) \qquad (6.1)$$

$$F(t) = c \cdot i(t)$$

where

m = Equivalent mass of the cart

x = Position

F = Force due to DC motor

F_f = Force due to friction

i = Motor current

c = Torque constant

The friction force F_f is a discontinuous function of the velocity $\dot{x}$ and of the motor force F. It can be characterized by three parameters F_1, F_2 and F_3:

$$F_f(\dot{x}(t), F(t)) = \begin{cases} F_1 + F_2 \cdot \dot{x}(t) & \text{if } (\dot{x}(t) > 0) \text{ or } (\dot{x}(t) = 0 \text{ and } F(t) > F_3) \\ F(t) & \text{if } (\dot{x}(t) = 0 \text{ and } |F(t)| \le F_3 \\ -F_1 + F_2 \cdot \dot{x}(t) & \text{if } (\dot{x}(t) < 0) \text{ or } (\dot{x}(t) = 0 \text{ and } F(t) < -F_3) \end{cases} \qquad (6.2)$$

In Fig. 6.2 the graph of the friction force F_f as a function of the velocity $\dot{x}$ is presented.

The system under consideration has the input variable i(t) and the state variables x(t),$\dot{x}$(t). It is of second order and discontinuous (Fig. 6.3).

For given initial conditions x(0), $\dot{x}$(0) and a given input function i(t) (continuous or discontinuous) we shall simulate this system over the time interval $0 \le t \le t_f$. We want to know the values of x(t) and $\dot{x}$(t) at prescribed sampling points $k \cdot T_s$ (k=0,1,2,...).

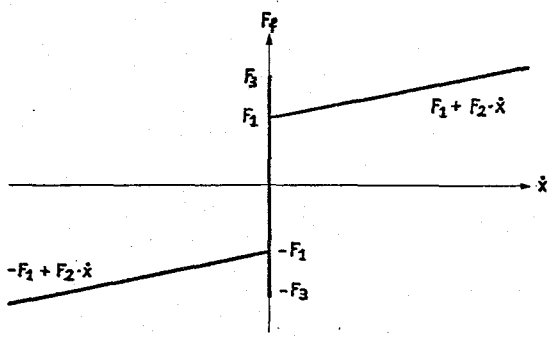

FIG. 6.2 FRICTION FORCE

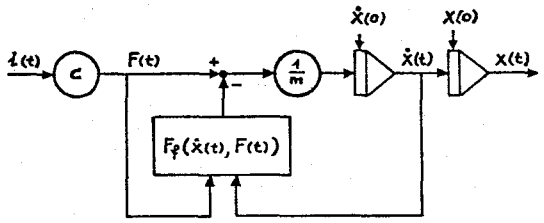

FIG. 6.3 STRUCTURE OF THE SYSTEM

Special care must be taken for proper consideration of the discontinuities in the differential equation (6.1). To solve such a simulation problem by using any numerical integration algorithm one has to describe the process by a sequence of continuous models. For our cart we obtain from (6.2) the following three models which consecutively can be integrated:

Model 1: Cart moving forward:

$$m \cdot \ddot{x}(t) = c \cdot i(t) - F_1 - F_2 \cdot \dot{x}(t) \; ; \; t_a \le t \le t_b$$

This model has to be selected if $\dot{x}(t_a) > 0$ or if $\dot{x}(t_a) = 0$ and $F(t_a) > F_3$. It has to be utilized until a time $t = t_b$ where one of the following events first occurs:

Event A: $\dot{x}(t_b) = 0$

Event B: $t_b = t_f$

Event C: $t_b = t_j$; j=1,2,... Time where i(t) is discontinuous

Model 2: Cart moving backward:

$$m \cdot \ddot{x}(t) = c \cdot i(t) + F_1 - F_2 \cdot \dot{x}(t) \; ; \; t_a \le t \le t_b$$

This model is selected if $\dot{x}(t_a) < 0$ or if $\dot{x}(t_a) = 0$ and $F(t_a) < -F_3$. It must be utilized until a time $t = t_b$ where one of the events A, B or C first occurs.

Model 3: Cart not moving:

$$\ddot{x}(t) = 0 \; ; \; t_a \le t \le t_b$$

This model has to be selected if $\dot{x}(t_a) = 0$ and $|F(t_a)| \le F_3$. It must be utilized until a time $t = t_b$ where one of the following events first occurs:

Event D: $|F(t_b)| > F_3$

Event B: $t_b = t_f$

The events A and D are state events, the events B and C

are time events. In case of a state event taking place an iterative algorithm has to be activated in order to determine the exact time t_b where the integration of the model must be stopped (e.g. Newton-Raphson or Secant method). For this reason it is most important that each of the three continuous models is also defined for $t>t_b$, that means outside of the range it can physically be interpreted.

The structure of the FORTRAN-IV program used to simulate the motion of the cart is shown in Fig. 6.4.

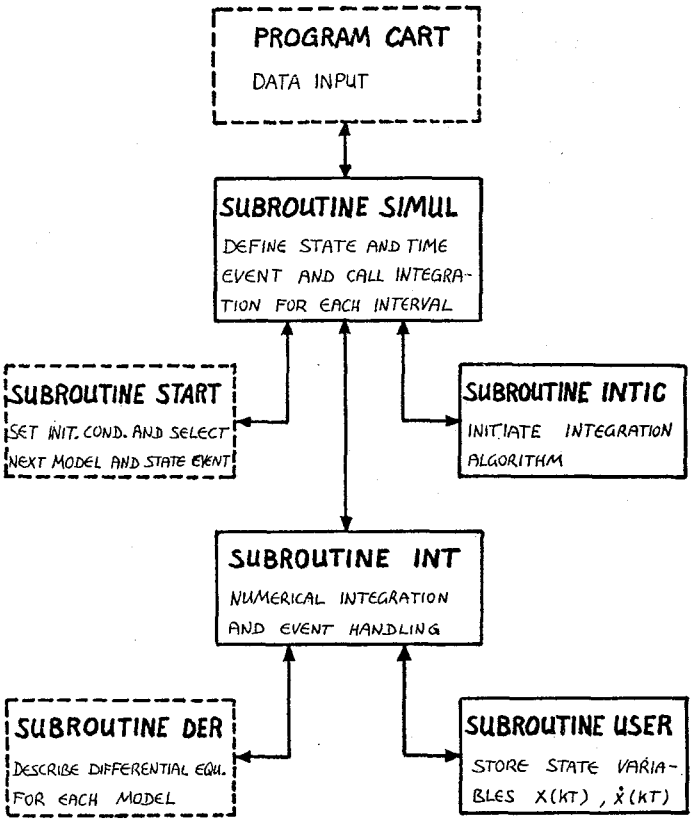

FIG. 6.4 STRUCTURE OF SIMULATION PROGRAM

In the main program numerical values of the input function i(t) and some simulation parameters are read in. The subroutine SIMUL controls and calls the integration of the different models over consecutive time intervals: First the numerical integration subroutine INT (which performs the integration connected with an event handling logic) is initiated by calling the subroutine INTIC. Then the subroutine START is called, where the initial conditions x(0) and $\dot{x}(0)$ are set and where the number of the model to be used during the first time interval and the appropriate state events (A or D) are evaluated. The subroutine SIMUL then determines the next time event (B or C) and calls the integration routine INT which for several times calls the user supplied subroutines DER (containing the differential equations for the three models) and USER (storing of data). If one of the defined time or state events occurs the integration is stopped and control is given back to the subroutine SIMUL. Then again the subroutine START is called to detect the appropriate model number and the event types for the next time interval.

With the following numerical values:

$x(0) = 0$ [m]

$\dot{x}(0) = 0$ [m/s]

$m = 0.64$ [kg]

$F_1 = 0.75$ [kg·m/s^2]

$F_2 = 0.28$ [kg/s]

$F_3 = 0.83$ [kg·m/s^2]

$c = 0.08$ [kg·m/(s^2·A)]

and with the input function i(t) shown in Fig. 6.5 a simulation run was performed. The resulting state variables x(t) and $\dot{x}(t)$ are also presented in Fig. 6.5.

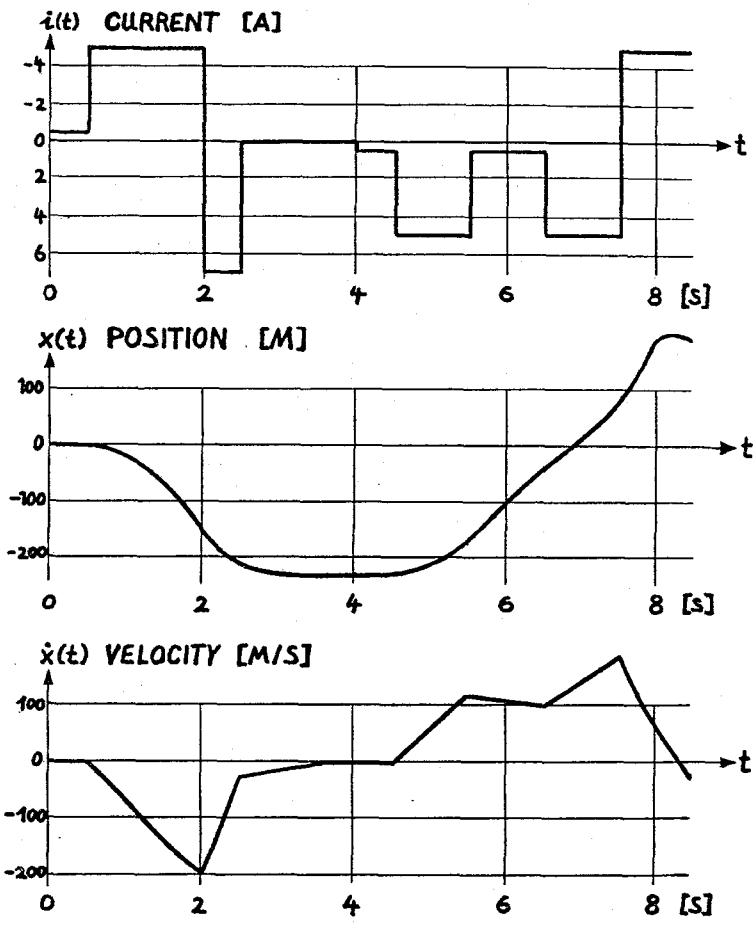

FIG. 6.5 RESULTS OF CART SIMULATION

Using a Runge-Kutta algorithm of 8th order with a relative tolerance of 10^{-6} the simulation required 0.5 seconds on a CDC 6500 computer. In Fig. 6.6 a list of all state and time events occuring during the simulation is given.

VII) ACKNOWLEDGEMENT

We wish to express our gratitude towards Prof.Dr.M.Mansour, the head of our institute at the Swiss Federal Institute of Technology Zurich, who gave us the opportunity to carry out the research work described in this article. Likewise we wish to express our thanks to AGIE Ltd. for their support of this work.

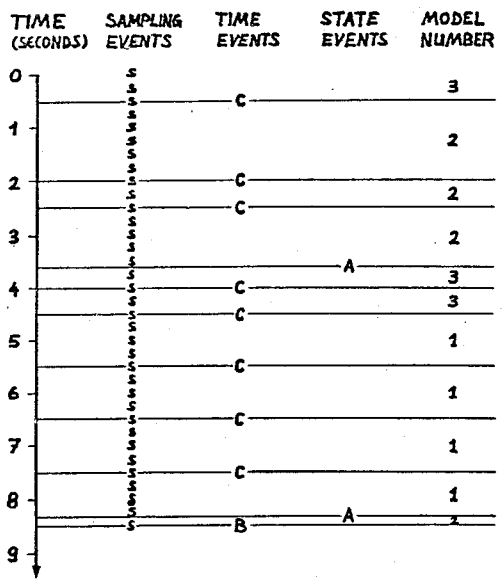

FIG. 6.6 EVENTS DURING SIMULATION

VIII) REFERENCES

[1] F.E.Cellier: Continuous-System Simulation by Use of Digital Computers: A State-of-the-Art Survey and Prospectives for Development. *Proc. SIMULATION'75*

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

[3] E.Fehlberg: Report: *NASA TR R-287*

[4] A.Björk, G.Dahlquist: Numerische Methoden. *Oldenburg Verlag* München 1972

[5] G.Lekkas, H.G.Erzinger: Aufstellen eines Pendels. Semester work. *Institute for Automatic Control, Swiss Federal Institute of Technology Zurich* 1975