Integration routines for systems with discontinuities

J. L. Hay, R. E. Crosbie, and R. I. Chaplin

Department of Electrical Engineering, University of Salford, Salford M5 4WT

The digital computer analysis of dynamic systems, described by differential equations, is often complicated when a discontinuity causes the equations to change. This paper discusses the problems introduced by discontinuities and describes subroutines which may be used in conjunction with a general purpose integration routine to aid the modelling of discontinuous systems.

(Received May 1973)

1. Introduction

The dynamic analysis of electrical networks, control systems and other dynamic systems by digital computer is often complicated by the presence of discontinuities. Such systems are usually described by a set of differential equations which change when a discontinuity occurs. These problems are best tackled using specially designed integration routines. This paper discusses the problems of dealing with discontinuities and describes subroutines used in conjunction with a general-purpose integration routine which aid the modelling of discontinuous systems.

2. Location of discontinuities

Most integration routines require the system equations in the form of a first-order set:

$$\frac{dy_i}{dt} = f_i(t, y_1, y_2, \ldots, y_n) \quad i = 1, 2, \ldots, n \tag{1}$$

In a discontinuous system the f_i in (1) change according to the state of the system. Therefore to generalise (1) to permit m different states $S_1, S_2, \ldots, S_m$

$$\frac{dy_i}{dt} = f_{ij}(t, y_1, y_2, \ldots, y_n) \quad i = 1, 2, \ldots, n \qquad j = 1, 2, \ldots, m \tag{2}$$

where the state, S, of the system is determined by a set of discontinuity functions $\phi_k(t, y_1, y_2, \ldots, y_n)$ which are defined such that a discontinuity occurs when one of the conditions $\phi_k = 0$ is satisfied.

The computer program must check the values of ϕ_k repeatedly to determine the state of the system so that the appropriate set of equations from (2) can be substituted. Whatever integration method is used, the values of y_i and ϕ_k will be available only at certain discrete values of t because of the step-by-step nature of digital integration. Therefore, a change of state is detected by noting a change of sign in the value of a discontinuity function and the precise instant at which the discontinuity occurred will not be known.

Subsequent action may take several forms. The simplest approach is to assume that a discontinuity has occurred at the end of the step in which it was detected, equations (2) being changed for the start of the next step. This method introduces a timing error which may seriously affect subsequent results, particularly if more than one discontinuity occurs within a single step. An excessively short step-length may therefore be necessary to locate discontinuities sufficiently precisely.

A second method requires an integration routine which varies the step-length according to an estimate of the local truncation error. The discontinuity functions are checked after each derivative evaluation rather than after each complete integration step, so that if necessary the derivative equations are changed part way through a step. Changing equations in mid-step produces an artificially large error estimate causing the step to be subdivided until the control mechanism selects a much reduced step-length until the discontinuity has been negotiated.

A third method, like the first, requires that the discontinuity functions are evaluated only at the end of each step. If, however, a discontinuity is detected, additional calculations are performed to locate it accurately. The last step is then repeated with a shortened step-length so as to end at the discontinuity and the integration continues with the new equations. The routines, described below, use a sequence of linear interpolations between ends of successive steps to locate the discontinuity to a prescribed accuracy. The method employed detects the discontinuity to a specified accuracy, and provides facilities to process any discontinuity which is simply specified by a discontinuity function alone. O'Regan (1970) gives an interesting alternative to linear interpolation which uses a third-order interpolation to pinpoint the discontinuity, without repeating the integration step. However, the accuracy of discontinuity detection may be questioned and this technique requires the user to provide more information than the simple discontinuity functions.

A completely different approach is to change the variable of integration (Fox, 1962) from time to the appropriate ϕ_k. This method has been found rather unwieldy, when applied to systems with multiple discontinuities and is not well suited to general-purpose routines.

3. Error control of step-length

Error-controlled variation of step-length is a useful feature of some integration routines which is incorporated in the method described in the following section. Organisational problems arise in combining step-length control features and discontinuity location, however, automatic step-length control permits improved computational efficiency. Step-length control is particularly important when solving equations of a discontinuous system because the discontinuity detection procedure demands additional integration steps to be performed. The so-called pseudo-iterative procedures (Sarafyan, 1966) have been found most satisfactory for error estimation. This group of procedures supplies two solutions in each step, of order n and $n - 1$. The difference between these two solutions, which approximates to the nth term in the corresponding Taylor series expansion, is used as an upper bound to the local error (Sarafyan, 1966; Schiesser, 1970; England, 1969; Chai, 1970; Merson, 1957; Crosbie and Hay, 1971).

4. The integration routine and discontinuity detection process

INT is a general purpose integration routine which may employ any mathematical integration formulae. However, in general a high order method with automatic step-length control is recommended. Sarafyan (1966) and England (1969) each describe a suitable fifth order integration process which has an embedded fourth order solution. The difference between the fifth and fourth order solutions is an indication of the local truncation error and may be used to control the integration step-length.

The discontinuity facilities are introduced by a subroutine INTERP which intercepts the flow of results from INT to the printout instructions, **Fig. 1.** If a discontinuity function has changed sign during the last integration step INTERP forces INT to recompute the last step with a reduced step-length to give a first approximation to the point of discontinuity and this interpolative procedure is repeated until the discontinuity is located to a specified accuracy. CNTRL may then be entered to register the change in the system state. However, if no discontinuity has occurred during the last step INTERP arranges for control to pass to the printout instructions and the subroutine CNTRL is by-passed. INTERP, like the integration subroutine, is a general purpose mathematical subroutine and is not dependent on the system being simulated. Whereas the integration subroutine obtains information of the differential equations representing the current state of the system from the user provided subroutine DERIV, INTERP obtains values of the discontinuity functions from the user provided routine DFUNCT. The relationship between the two mathematical and the three user provided subroutines is shown by Fig. 1, and a simplified flow diagram of INTERP is presented in **Fig. 2.**

Consider as an example a system which has three states:

State ISTATE = 1: if $|y| < z$ then

$$\frac{dy}{dt} = f_1(t, y) = -Ay + \sin(\omega t)$$

ISTATE = 2: if $y \geqslant z$ then

$$\frac{dy}{dt} = f_2(t, y) = -By + \sin(\omega t)$$

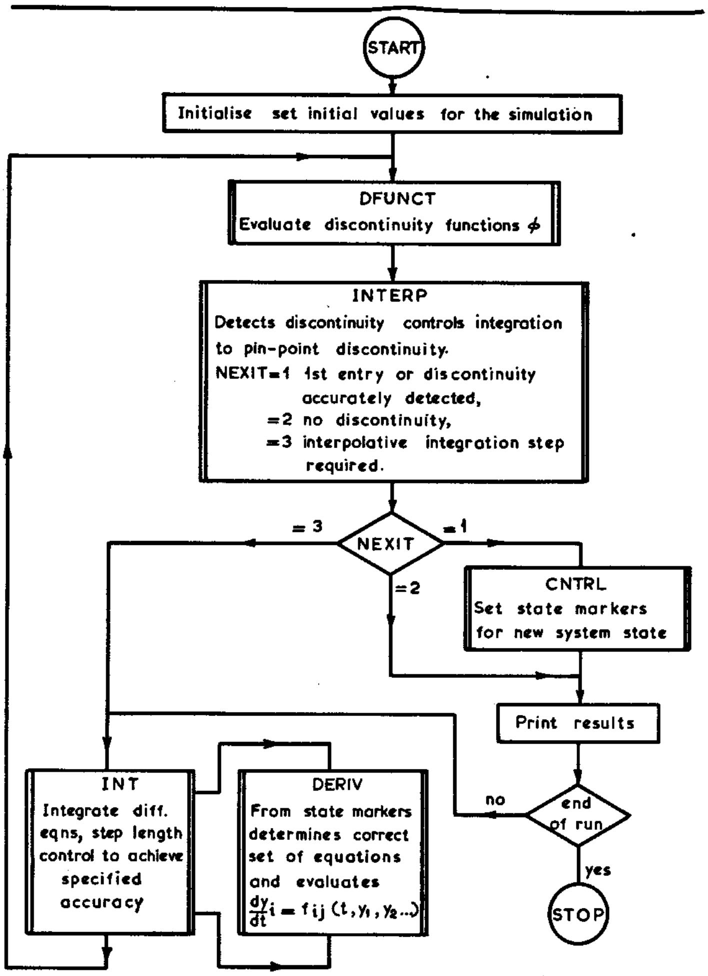

Fig. 1 General flow diagram for complete program

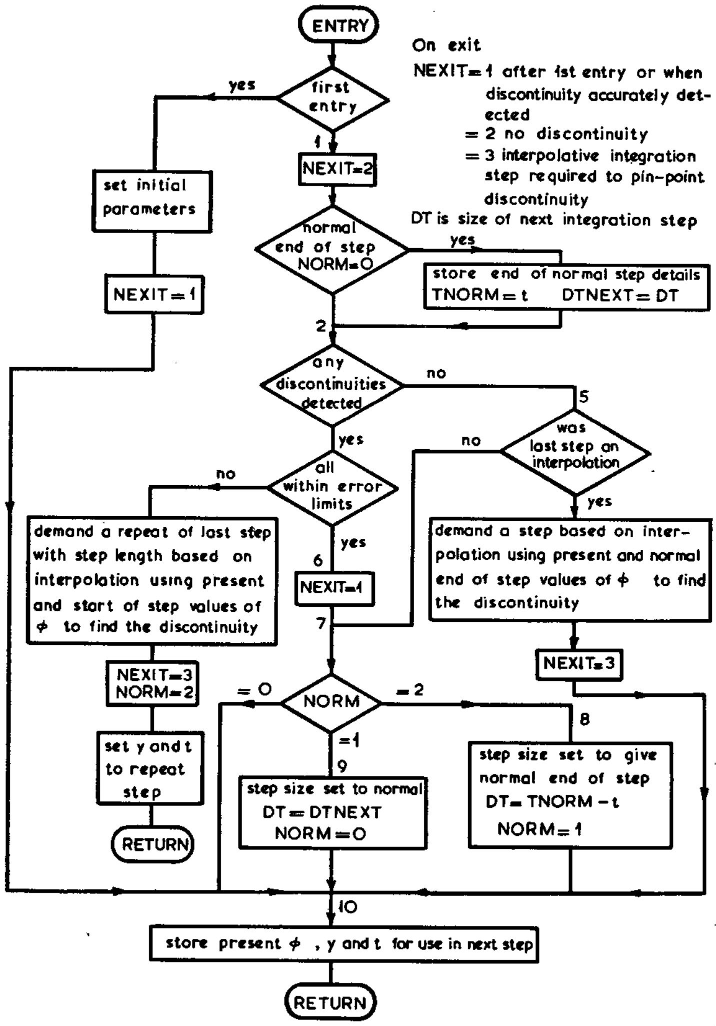

Fig. 2 INTERP flow diagram

ISTATE = 3: if $y \leqslant -z$ then

$$\frac{dy}{dt} = f_3(t, y) = -Cy + \sin(\omega t) \quad (3)$$

The state marker, ISTATE, determines which differential equation is selected in DERIV. The value of the state marker is passed to DERIV from CNTRL in which it is evaluated from the current values of the two discontinuity functions required in this case. These are defined in DFUNCT as:

$$\phi_1 = y - z; \ \phi_2 = -y - z$$

and ISTATE is defined in CNTRL as:

$$\begin{aligned} \text{ISTATE} &= 1 \text{ if } \phi_1 < 0 \text{ and } \phi_2 < 0 \\ \text{ISTATE} &= 2 \text{ if } \phi_1 \geqslant 0 \\ \text{ISTATE} &= 3 \text{ if } \phi_2 \geqslant 0 \end{aligned}$$

The system equations are integrated using the standard integration method provided in subroutine INT until a discontinuity is detected. Let the step in which this occurs range from $t = T$ to $t = T + h$. INTERP now interpolates linearly to obtain a first approximation to the point of discontinuity $t = T + ah (0 < a < 1)$. An integration step is now taken from T to $T + ah$ and the discontinuity functions are again checked. Either the discontinuity is found between T and $T + ah$ or $T + ah$ and $T + h$, in either case an interpolation/integration cycle is repeated employing an interpolation based on values of the discontinuity function at t equal to T and $T + ah$ or $T + ah$ and $T + h$. These operations are repeated until the discontinuity function becomes sufficiently small and has changed sign. The latter ensures that at the point detected as the discontinuity, the discontinuity function has a sign which

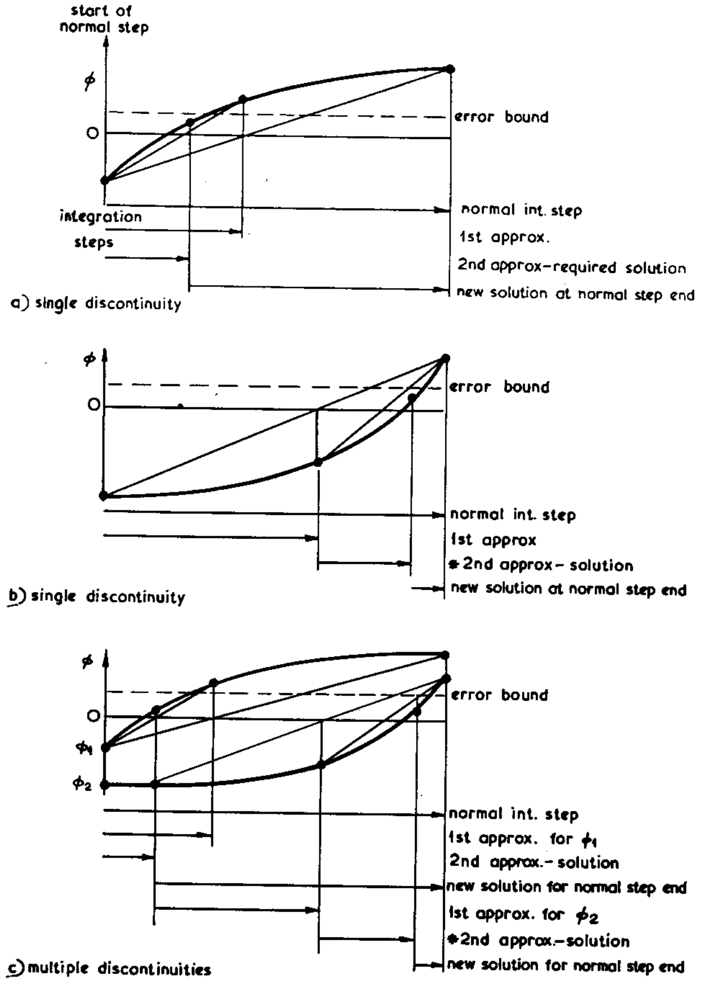

Fig. 3 Integration steps when negotiating discontinuities
*Forward interpolations aim for error bound and not $\phi = O$.

corresponds to the new state of the system. **Fig. 3** indicates the steps which may be necessary, and Fig. 3(*c*) the steps undertaken when two discontinuities occur within a single step.

5. Program detail

The FORTRAN program which simulates the discontinuous system described in the previous section is presented in **Fig. 4.** The program conforms to the flow diagram of Fig. 1, and the definition of program variables is presented in the following table.

List of principal program variables associated with:
1. *Differential equations*

N	number of differential equations.
T	independent variable time.
Y	array of the current values of the dependent variables.
YLAST	values of *Y* at the start of the last integration step.
DY	array for the current values of dY/dT.
A, B, C, W	parameters of the equations defined in (3).

2. *Discontinuity functions*

NDF	number of discontinuity functions.
DF	array of the current values of the discontinuity functions ϕ.
DFLAST	values of DF at the start of the last integration step.
ERROR	array of error bounds to which the corresponding discontinuities must be detected.
Z	parameter defining DF defined in (3).

```
      DIMENSION Y(1),DY(1),ERROR(2),DF(2),DFLAST(2),YLAST(1),WORK(11)
      COMMON ISTATE,A,B,C,W
      EXTERNAL DERIV
      WRITE(6,10)
C        DIFF. EQN. DATA
      READ(5,11)A,B,C,W,Y(1),T
      N=1
C        DISCONTINUITY FUNCTION DATA
      NDF=2
      READ(5,11)(ERROR(I),I=1,NDF),Z
C        RUN CONTROL DATA
      READ(5,11)DTMAX,EPS,TF
      DT=DTMAX
      IFIRST=0
C        MAIN PROGRAM LOOP
    1 CALL DFUNCT(DF,Y,T,Z)
      CALL INTERP(NEXIT,T,DT,Y,YLAST,ERROR,DF,DFLAST,NDF,N,IFIRST)
      GOTO(2,3,4),NEXIT
    2 CALL CNTRL(DF,ISTATE)
      WRITE(6,13)ISTATE
    3 X=SIN(W*T)
      WT=W*T*180.0/3.1415927
      WRITE(6,12)T,WT,Y(1),X
      IF(T.GE.TF)STOP
    4 CALL INT(DERIV,Y,DY,T,DT,DTMAX,EPS,N,WORK)
      GOTO 1
   10 FORMAT(1H1,6X,4HTIME,3X,7HWT(DEG),9X,1HY,4X,6HSIN WT//)
   11 FORMAT(7F10.4)
   12 FORMAT(F10.4,F10.1,2F10.4)
   13 FORMAT(16H SYSTEM STATE = ,I1)
      END

      SUBROUTINE DFUNCT(DF,Y,T,Z)
      DIMENSION DF(2),Y(1)
      DF(1)=Y(1)-Z
      DF(2)=-Y(1)-Z
      RETURN
      END

      SUBROUTINE DERIV(Y,DY,T)
      DIMENSION Y(1),DY(1)
      COMMON ISTATE,A,B,C,W
      GOTO(1,2,3),ISTATE
    1 D=A
      GOTO 4
    2 D=B
      GOTO 4
    3 D=C
    4 DY(1)=-D*Y(1)+SIN(W*T)
      RETURN
      END

      SUBROUTINE CNTRL(DF,ISTATE)
      DIMENSION DF(1)
      ISTATE=1
      IF(DF(1).GE.0.0)ISTATE=2
      IF(DF(2).GE.0.0)ISTATE=3
      RETURN
      END

DATA
1.0       0.5       0.2       1.0       0.0       0.7853982
0.0001    0.0001    0.5
3.1415927 0.001     12.5
```

Fig. 4a Program—user provided code

3. *Integration procedure*

DTMAX	maximum value of integration step.
DT	actual value of step which may be adjusted by INT.
EPS	absolute accuracy criterion for an integration step.
WORK	array of working storage used by INT.

4. *Interpolation procedure*

IFIRST	is set to 0 to indicate first entry to INTERP.
NEXIT	is set by INTERP to:

=1 if first use of INTERP, or if a discontinuity has been accurately detected as occurring at the current *T*,
=2 no discontinuity detected,
=3 an integration step is required to locate a discontinuity.

The program listing does not contain an integration subroutine, and it is the authors' intention that a suitable standard subroutine will be incorporated from the library available at the user's computer installation. Any integration process may be used with this program, however, preference should be given to subroutines which comply with the specification presented in the last two sections.

The program running efficiency could be slightly improved by making greater use of FORTRAN COMMON facilities to reduce the time required to communicate arguments during the subroutine calling process. However, this speed improvement is offset by the inconvenience of having a long compulsory list of COMMON variables.

```
      SUBROUTINE INTERP(NEXIT,T,DT,Y,YLAST,ERROR,DF,DFLAST,NDF,N,IFIRST)
      DIMENSION Y(N),YLAST(N),ERROR(NDF),DF(NDF),DFLAST(NDF)
      NDIS=0
      IF(IFIRST.NE.0)GOTO 1
      IFIRST=1
      NEXIT=1
      NORM=0
      IREM=0
      GOTO 10
1     NEXIT=2
      IF(NDF.LE.0)GOTO 10
      IF(NORM.NE.0)GOTO 2
      TNORM=T
      DTNEXT=DT
2     DTPREV=T-TLAST
      DTNEAR=DTPREV
      NEAR=0
      NOTACC=0
      DO 3 I=1,NDF
      U=DFLAST(I)
      V=DF(I)
      IF((U.GE.0.0.AND.V.GE.0.0).OR.(U.LT.0.0.AND.V.LT.0.0))GOTO 3
      ERR=ERROR(I)
      IF(ABS(V).GT.ERR)NOTACC=1
      TEMP=DTPREV*U/(U-V)
      NDIS=NDIS+1
      IF(TEMP.GE.DTNEAR)GOTO 3
      DTNEAR=TEMP
      NEAR=I
3     CONTINUE
      IF(NDIS.EQ.0)GOTO 5
      IF(NOTACC.EQ.0)GOTO 6
      NEXIT=3
      IREM=NEAR
      DT=DTNEAR
      NORM=2
      DFEND=DF(NEAR)
      TEND=T
      T=TLAST
      DO 4 I=1,N
4     Y(I)=YLAST(I)
      RETURN
5     IF(IREM.EQ.0)GOTO 7
      NEXIT=3
      U=DF(IREM)
      V=1.0/(U-DFEND)
      DT=TEND-T
      DT=U*DT*V+ABS(ERROR(IREM)*DT*V)
      GOTO 10
6     IREM=0
      NEXIT=1
7     IF(NORM-1)10,9,8
8     DT=TNORM-T
      NORM=1
      GOTO 10
9     DT=DTNEXT
      NORM=0
10    DO 11 I=1,NDF
11    DFLAST(I)=DF(I)
      DO 12 I=1,N
12    YLAST(I)=Y(I)
      TLAST=T
      RETURN
      END
```

Fig. 4b Program—standard subroutine INTERP

TIME	WT(DEG)	Y	SIN WT
SYSTEM STATE = 1			
0.7854	45.0	0.0000	0.7071
1.1781	67.5	0.2706	0.9239
SYSTEM STATE = 2			
1.5708	90.0	0.5000	1.0000
1.9635	112.5	0.7579	0.9239
2.3562	135.0	0.9161	0.7071
2.7489	157.5	0.9477	0.3827
3.1416	180.0	0.8456	-0.0000
3.5343	202.5	0.6235	-0.3827
SYSTEM STATE = 1			
3.7015	212.1	0.4999	-0.5311
4.3197	247.5	-0.0864	-0.9239
4.7124	270.0	-0.3756	-1.0000
SYSTEM STATE = 3			
4.9382	282.9	-0.5000	-0.9746
5.1051	292.5	-0.6398	-0.9239
5.4978	315.0	-0.9029	-0.7071
5.8905	337.5	-1.0424	-0.3827
6.2832	360.0	-1.0359	0.0000
6.6759	382.5	-0.8835	0.3827
7.0686	405.0	-0.6075	0.7071
SYSTEM STATE = 1			
7.1936	412.2	-0.5000	0.7897
7.8540	450.0	0.1962	1.0000
8.2467	472.5	0.4482	0.9239
SYSTEM STATE = 2			
8.3693	479.5	0.5000	0.8701
8.6394	495.0	0.6369	0.7071
9.0321	517.5	0.7183	0.3827
9.4248	540.0	0.6571	-0.0000
SYSTEM STATE = 1			
9.7654	559.5	0.4999	-0.3340
9.8175	562.5	0.4563	-0.3827
10.6029	607.5	-0.1860	-0.9239
10.9956	630.0	-0.4429	-1.0000
SYSTEM STATE = 3			
11.1043	636.2	-0.5000	-0.9941
11.3883	652.5	-0.7389	-0.9239
11.7810	675.0	-0.9946	-0.7071
12.1737	697.5	-1.1271	-0.3827
12.5664	720.0	-1.1142	0.0000

Fig. 5 Results from program of fig. 4

The printout resulting from running the program for the specified data is shown in **Fig. 5.**

6. Conclusions

The software has been successfully tested on a number of problems with various types of discontinuity including multiple discontinuities occurring within one integration step. Work is proceeding on the improvement of interpolation techniques and a library of standard subroutines defining common types of discontinuity is under development.

References

CHAI, A. S. (1970). 'Comment on the Runge-Kutta-Merson algorithm', *Simulation*, Vol. 15, pp. 89-91, Aug. 1970.

CROSBIE, R. E., and HAY, J. L. (1971). 'Variable step-length integration routines', *Simulation*, Vol. 17, pp. 206-212, Nov. 1971.

ENGLAND, R. (1969). 'Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations', *The Computer Journal*, Vol. 12, pp. 166-170, May 1969.

FOX, L. (1962). *Numerical Solution of ordinary and partial differential equations*, Pergamon Press, London, 1962.

MERSON, R. H. (1957). 'An operational method for the study of integration processes', *Proc. Symposium on Data Processing*, Weapons Research Establishment, Salisbury, S. Australia, 1957.

O'REGAN, P. G. (1970). 'Step size adjustment at discontinuities for fourth order Runge-Kutta methods', *The Computer Journal*, Vol. 13, pp. 401-404, Nov. 1970.

SARAFYAN, D. (1966). *Error estimation for Runge-Kutta methods through pseudo-iterative formulas*, Tech. Report No. 14, Louisiana State University, New Orleans, 1966.

SCHIESSER, W. E. (1970). 'Technical Comment', *Simulation*, Vol. 14, pp. 94, Feb. 1970.