AGENCY FOR INTERNATIONAL DEVELOPMENT
WASHINGTON, D. C. 20523

BIBLIOGRAPHIC INPUT SHEET

FOR AID USE ONLY
Batch 63

1. SUBJECT CLASSIFICATION
A. PRIMARY: TEMPORARY
B. SECONDARY:

2. TITLE AND SUBTITLE
Distributed delay routines: DEL,DELS, DELF,DELLF,DELVF,DELLVF

3. AUTHOR(S)
Abkin,M.H.; Wolf,Christopher

4. DOCUMENT DATE: 1976
5. NUMBER OF PAGES: 113p.
6. ARC NUMBER: ARC

7. REFERENCE ORGANIZATION NAME AND ADDRESS
Mich.State

8. SUPPLEMENTARY NOTES (*Sponsoring Organization, Publishers, Availability*)
(In Computer Library for Agr.Systems Simulation. CLASS doc.no.8)

9. ABSTRACT
(LIBRARY & INFO.SCIENCE R&D)

10. CONTROL NUMBER
PN-AAE-013

11. PRICE OF DOCUMENT

12. DESCRIPTORS
Computer programs
Damage assessment
Models
Storage
Time lag

13. PROJECT NUMBER

14. CONTRACT NUMBER
CSD-2975 Res.

15. TYPE OF DOCUMENT

AID 590-1 (4-74)

DISTRIBUTED DELAY ROUTINES:
DEL, DELS, DELF, DELLF, DELVF, DELLVF

Michael H. Abkin

Chris Wolf

COMPUTER LIBRARY FOR AGRICULTURAL SYSTEMS SIMULATION

CLASS Document No. 8

Agricultural Sector Analysis and Simulation Projects
Supported under Contract No. AID/csd-2975
U. S. Agency for International Development

Department of Agricultural Economics
Michigan State University
East Lansing, Michigan

July 1976

TABLE OF CONTENTS

Preface vii

Distributed Delay ..... DEL

Distributed Delay with Storage ..... DELS

Distributed Delay with Finer Time Increment ..... DELF

Distributed Delay with Storage Losses ..... DELLF

Distributed Delay with Variable Time Delay ..... DELVF

Distributed Delay with Storage Losses and Variable Time Delay .... DELLVF

COMPUTER LIBRARY FOR AGRICULTURAL SYSTEMS SIMULATION

The Computer Library for Agricultural Systems Simulation (CLASS) is one of the four major activities of the Agricultural Sector Analysis and Simulation Projects at Michigan State Unviersity under U. S. Agency for International Development Contract AID/csd-2975. The other three major interrelated project activities include theoretical and methodological research, the Development Analysis Study Program, and field activities, primarily in the Republic of Korea.

The project objective is to develop an approach to institutionalizing an analytical capacity for planning, policy formulation, program development, and project implementation for agricultural sector development within the public decision making structure of developing countries. A major component of the analytical capacity is a series of system simulation models tailored to the needs of the individual country. Much of the experience gained from the field activity and the knowledge gained from the theoretical and methodological research added to the present stock of knowledge about building and maintaining analytical capacities for agricultural sector development can be preserved and extended in the training provided through the Development Analysis Study Program and in the stock of model, component, and utility routine computer software documented in the Computer Library for Agricultural Systems Simulation.

In full operation, the Computer Library for Agricultural Systems Simulation (CLASS) acquires, catalogs, maintains and distributes computer programs and associated documentation. These computer programs are of generalized simulation models, components, and routines designed specifically for the analysis of agricultural development problems and processes. In particular, the library sets standards of admissibility for programs and documentation; catalogs and indexes programs and documentation so as to facilitate their retrieval by users seeking a set of programs to be used in a specific problem analysis; and distributes programs and documentation to users.

To enhance the effectiveness of the library, its functions also include identifying and soliciting needed models; actively bringing programs and documentation up to the library's standards; and providing limited consultation in identifying and implementing appropriate library programs for a particular application. A subsidiary function of the library in conjunction with the identification and solicitation of models is to survey and catalog ongoing research in agricultural systems modeling and simulation.

The CLASS document publication series is the main vehicle for informing potential users of the substance of CLASS holdings and activities.

July, 1976

George E. Rossmiller
Director
Agricultural Sector Analysis and
Simulation Projects

PREFACE

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times so that, while entities may enter the delay processes at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc.

This publication provides the set of documentations for six distributed delay routines: DEL, DELS, DELF, DELLF, DELVF, DELLVF. The delay in all routines is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. DEL is the simplest form of the distributed time delay process. The other routines have additional features as summarized in Table 1. Five routines compute storage in the delay by integrating the net flow into the delay. Four routines insure simulation stability by automatically decreasing the integration step size when the time constant is small relative to the order of the delay. Two

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	73
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	106

routines allow for proportional storage losses and/or accretions. Finally, two routines allow the mean lag time of the delay to be varied over the simulation run.

The last three columns of Table 1 show the compile and execution central processor times and core storage required by the distributed delay routines on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

M. H. A.
C. J. W.

CLASS Document

1. Symbolic Name	2. Name	3. Type	4. Version No. and Date
DEL	Distributed Delay	routine	1A 1/11/74

DEL Abstract

DEL simulates a distributed time delay process. It is useful for modeling information lags and for flow delays inherent in such aggregated processes as investment, production, transportation, information dissemination, and demographic processes.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The input to DEL is the unlagged value or flow into the delay, and the output is the lagged value or flow out of the delay.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DEL	Distributed Delay	routine	1A 1/11/74

DEL Documentation

Contents

I. Process Description 1

II. Technical Description 2

A. Mathematical model 2

B. Sample run 5

III. Program Information 9

A. Program description 9

B. Program implementation 9

C. Program lists 10

1. Of sample run executive 11

2. Of component 12

Bibliographic References 13

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DEL simulates this process.

In the technical description which follows, reference is made to the use of DEL to model the gestation stage of cocoa trees, where the input is tree plantings and the output is trees becoming productive.

II. Technical Description

A. Mathematical model

DEL is identical mathematically to the DELAY2 routine of [1], produces the same numerical results as the DELAY routine of [3], and is a generalization of the DELAY3 routine of [4].

The kth-order differential equation modeling a distributed delay can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [5] (Figure 1).

The delay times for individual trees entering the gestation delay follow a kth-order gamma distribution with mean D. k determines the variance σ^2 of the gamma distributed delay times as $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real-world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance--i.e., a discrete delay of length D.

The ith stage of the delay is modeled by the first-order differential equation

$$(1) \quad \frac{D}{k} \frac{dR_i(t)}{dt} + R_i(t) = R_{i+1}(t)$$

which can be solved numerically using Euler's integration approximation by

$$(2) \quad R_i(t) = R_i(t-DT) + DT \cdot \frac{k}{D}[R_{i+1}(t-DT) - R_i(t-DT)]$$

where, for the kth stage of the delay, R_{k+1} = RIN and where

R_i = rate out of the ith stage (trees/year)

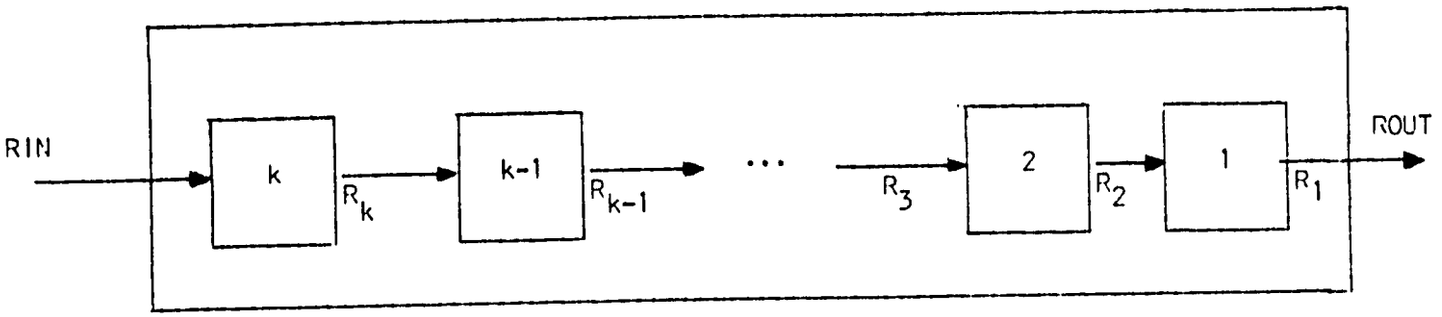

Figure 1

k^{th}-Order Distributed Delay Process

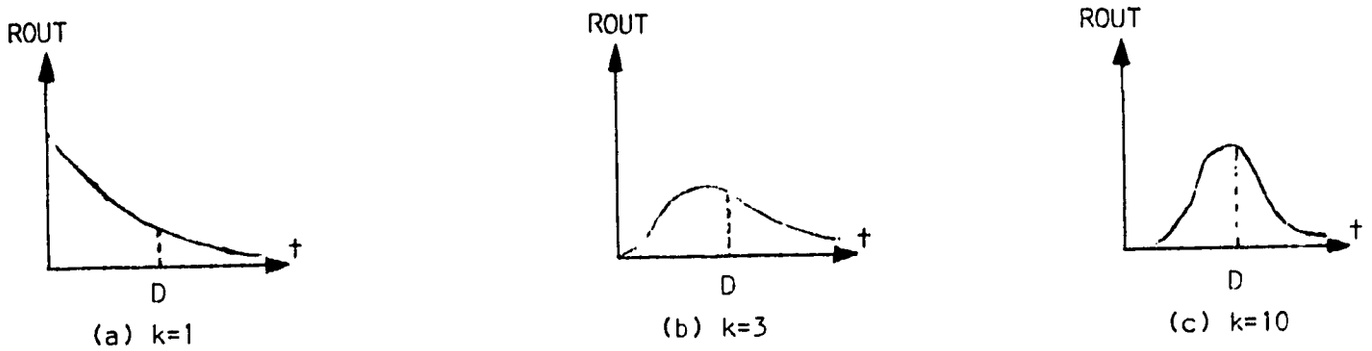

Figure 2

k^{th}-Order Delay Zero-State Unit Impulse Response

RIN = planting rate (Figure 1) (trees/year)

k = the order of the delay process

D = the mean delay time (years)

DT = the integration step size, the simulation time increment (years).

The output of the delay (trees becoming productive--Figure 1) is

(3) $ROUT(t) = R_1(t)$.

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates, R_i, $i = 1, \ldots, k$, should be initialized to

(4) $R_i(0) = S(0)/D, \qquad i = 1, \ldots, k$

where S(0) is the initial storage (i.e., number of trees) in the gestation delay. If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to

(5) $R_i(0) = ROUT(0) = RIN(0), \qquad i = 1, \ldots, k.$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

(6) $S(0) = \frac{D}{k} \sum_{i=1}^{k} R_i(0).$

If a model contains several delays, stability considerations dictate that

(7) $DT < \min_j \left[\frac{2D_j}{k_j}\right]$.

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes

(8) $DT \leq \min_j \left[\frac{D_j}{2k_j}\right]$

where j indexes all the delays in a model. Alternatively, if (8) requires DT to be too small for efficient computation, a larger DT may be used for the model as a whole by using routine DELF in place of DEL for those delays j whose $D_j/2k_j$ ratios require a smaller DT.

DEL is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DEL (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample run

The sample run assumes a third-order, three-year gestation delay (i.e., k = 3 and D = 3). No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter. In addition, the number of trees in the gestation stage at time t (storage) is

(9) $\frac{d\ S(t)}{dt} = RIN(t) - ROUT(t)$

and is simulated by Euler integration (See the list of the sample run executive program.) by:

(10) $S(t) = S(t-DT) + DT\cdot(RIN(t-DT) - ROUT(t-DT))$.

Under these conditions, we can use (1) and (9) to solve analytically for the output and storage of the gestation stage and compare these with the simulated output and storage from (2) and (10). The analytical solution for ROUT and S are:

(11) $ROUT(t) = 100[1-e^{-t}(\frac{1}{2}t^2 + t + 1)]$

(12) $S(t) = 100[3 - e^{-t}(\frac{1}{2}t^2 + 2t + 3)]$.

Results, including the percent deviations of the simulated solutions from the analytical solutions, are tabulated in Figure 3 for a 10-year simulation with DT = .25 years.

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
0.0	0.00	0.00	0.00	0.00	0.00	0.00
1.0	8.03	5.08	-36.76	97.67	99.61	1.99
2.0	32.33	32.15	-.58	178.20	185.43	4.06
3.0	57.68	60.93	5.64	232.79	241.93	3.93
4.0	76.19	80.29	5.38	265.20	272.94	2.92
5.0	87.53	90.87	3.81	282.82	288.13	1.88
6.0	93.80	96.02	2.36	291.82	295.02	1.10
7.0	97.04	98.34	1.34	296.22	297.98	.60
8.0	98.62	99.33	.71	298.29	299.20	.31
9.0	99.38	99.74	.36	299.24	299.69	.15
10.0	99.72	99.90	.18	299.67	299.88	.07

Figure 3

DEL Sample Run Output

III. Program Information

A. Program description

DEL was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU HUSTLER 2 L239 LSD 32--an extension of CDC Scope 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 56 decimal words
CP compile time: .24 seconds
CP execution time (for sample run): .27 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DEL is structured as a single subroutine, DEL, which simulates a continuous (distributed) delay process. No COMMON blocks are used.

B. Program Implementation

All input is transmitted through the argument list. Input variables DELAY (called D in section II above), DT, and K require values which must remain constant during the simulation of a particular delay; RIN requires a value which may change during a simulation run; R(I) requires only initial values, as they are subsequently recomputed within DEL. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I). See the list of the sample run executive program for an example of the call statement.

For DEL to operate properly, the value of DT must be such that $0 < DT \leq DELAY/2K$, DELAY must be greater than zero, and K must be an integer ≥ 1. When $K = 1$, the delay can be simulated more efficiently by a single statement in the calling program than by this routine:

(13) $ROUT(t) = ROUT(t-DT) + DT \cdot \frac{K}{2 \cdot DELAY} (RIN(t-DT) - ROUT(t-DT))$.

C. Program lists

The computer programs for the sample run and for DEL are listed below:

```
      PROGRAM DELTST(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DELAY / 3. /
      DATA K / 3 /
      DATA DT / .25 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA STRG / 0. /
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        STRG = STRG + DT*(RIN-ROUT)
        CALL DEL(RIN,ROUT,R,DELAY,DT,K)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      AROUT = 100.*(1.-EXP(-T)*(T*T/2.+T+1.))
        ASTRG = 100.*(3.-EXP(-T)*(T*T/2.+2.*T+3.))
        PDOUT = 100.*(ROUT-AROUT)/AROUT
        IF(AROUT .EQ. 0.) PDOUT = 0.
        PDSTRG = 100.*(STRG-ASTRG)/ASTRG
        IF(ASTRG .EQ. 0.) PDSTRG = 0.
        WRITE(2,91) T,AROUT,ROUT,PDOUT,ASTRG,STRG,PDSTRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1      ANALYTICAL  SIMULATED  PER CENT  ANALYTICAL  SIMUL*
     +        *ATED  PER CENT*/* TIME   OUTPUT     OUTPUT    DEVIATION *
     +        *  STORAGE    STORAGE   DEVIATION*/)
91    FORMAT(* *F4.1,F10.2,5F11.2)
      END
```

```
      SUBROUTINE DEL(RIN,ROUT,R,DELAY,DT,K)

      TITLE - DISTRIBUTED DELAY

      VERSION - 1A

      DATE - 1 NOV 1974

      PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS

      INPUT REQUIREMENTS

          VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DELAY, DT, K,
                     R(I), RIN

          VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN
                    -- DELAY, DT, K
          VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN
                    -- RIN
          VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES
                    -- R(I)

      OUTPUT VARIABLES

         VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT

      VARIABLE DEFINITIONS

          DELAY - LENGTH OF DELAY IN TIME UNITS.

          DT - SIMULATION TIME INCREMENT IN TIME UNITS.

          K - ORDER OF DELAY. UNITLESS.

          R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

          RIN - INPUT RATE IN UNITS/UNIT TIME.

          ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

      DIMENSION R(1)
      A = FLOAT(K)*DT/DELAY
      IF(K .EQ. 1) GO TO 20
      KM1 = K - 1
      DO 10 I=1,KM1
        R(I) = R(I) + A*(R(I+1)-R(I))
10    CONTINUE
20    R(K) = R(K) + A*(RIN-R(K))
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald L. Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch. "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria." IEEE Transactions on Systems, Man, and Cybernetics. SMC-2:472-486. Sept., 1972.

[3] Llewellyn, Robert W., FORDYN: An Industrial Dynamics Simulator (Raleigh, North Carolina, 1965).

[4] Forrester, Jay W., Industrial Dynamics (Cambridge: M.I.T. Press, 1961).

[5] Manetsch and Park, op. cit., Chapter 10.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELS	Distributed Delay with Storage	routine	1A 1/11/74

DELS Abstract

DELS simulates a distributed time delay process and computes the storage associated with the flow through the delay. It is useful for modeling flow delays inherent in such aggregated processes as investment, production, transportation, and demographic processes.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The input to DELS is the unlagged value or flow into the delay, and the output is the lagged value or flow out of the delay. In addition, DELS computes storage in the delay by integrating the net flow into the delay.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELS	Distributed Delay with Storage	routine	1A 1/11/74

DELS Documentation

Contents

I. Process Description 1

II. Technical Description 2

A. Mathematical model 2

B. Sample run 7

III. Program Information 9

A. Program description 9

B. Program implementation 9

C. Program lists 10

1. Of sample run executive 11

2. Of component 12

Bibliographic References 13

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times; so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time, production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DELS simulates this process.

In the technical description which follows, reference is made to the use of DELS to model the gestation stage of cocoa trees, where the input is tree plantings and the outputs are trees becoming productive and the number of trees in the gestation stage.

II. Technical Description

A. Mathematical model

Aside from its computation of storage, DELS is identical mathematically to the DELAY2 routine of [1], produces the same numerical results as the DELAY routine of [3], and is a generalization of the DELAY3 routine of [4].

The kth-order differential equation modeling a distributed delay can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [5] (Figure 1).

The delay times for individual trees entering the gestation delay follow a kth-order gamma distribution with mean D. k determines the variance σ^2 of the gamma distributed delay times as $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real-world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance; i.e., a discrete delay of length D.

The ith stage of the delay--where i=1, 2, ..., k--is modeled by the first-order differential equation:

$$(1) \quad \frac{D}{K} \frac{dR_i(t)}{dt} + R_i(t) = R_{i+1}(t)$$

which can be solved numerically using Euler's integration approximation by

$$(2) \quad R_i(t) = R_i(t-DT) + DT \cdot \frac{k}{D}[R_{i+1}(t-DT) - R_i(t-DT)]$$

where, for the kth stage of the delay, R_{k+1} = RIN and where:

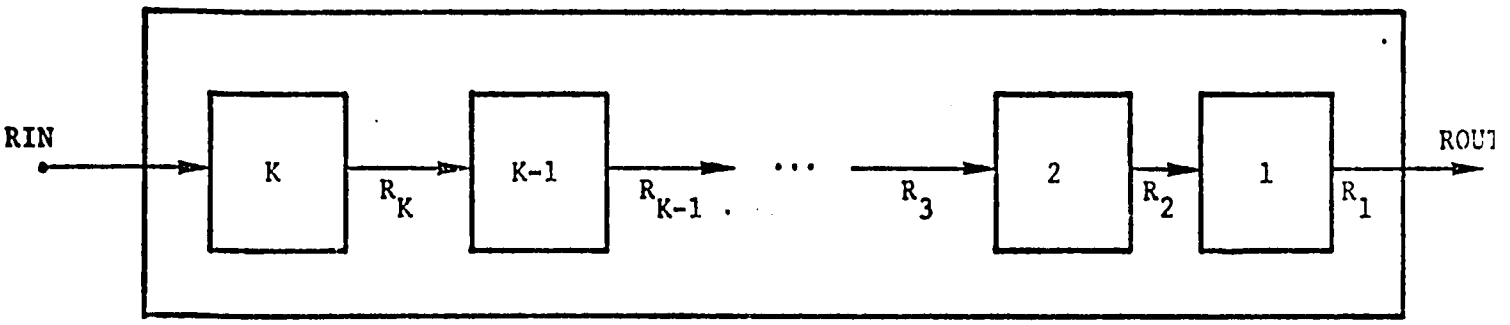

Figure 1

K^{th}-Order Distributed Delay Process

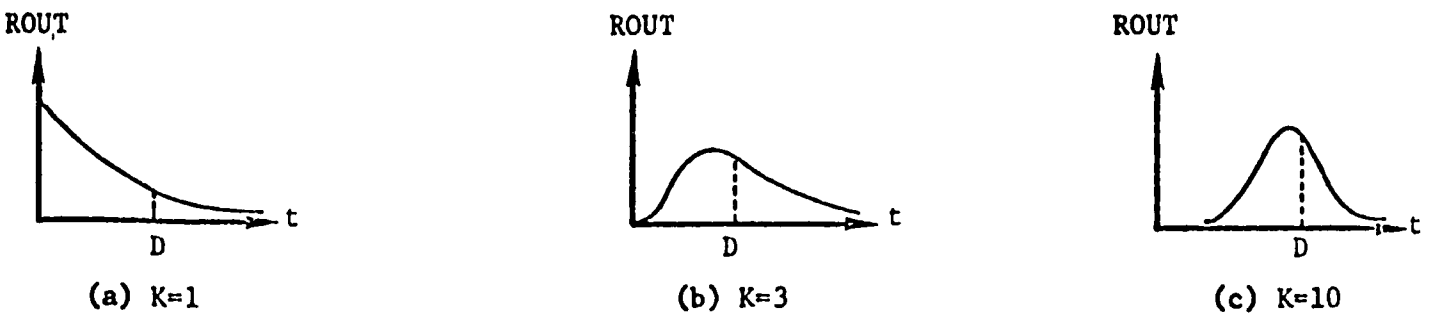

Figure 2

k^{th}-Order Delay Zero-State Unit Impulse Response

R_i = rate out of the ith stage (trees/year)

RIN = planting rate (Figure 1) (trees/year)

k = the order of the delay process

D = the mean delay time (years)

DT = the integration step size, the simulation time increment (years).

The output of the delay (trees becoming productive--Figure 1) is

(3) $ROUT(t) = R_1(t)$.

Finally, the storage in the delay associated with the flow through the delay (the number of trees in the gestation stage) is computed from the differential equation:

(4) $$\frac{dS(t)}{dt} = RIN(t) - ROUT(t)$$

using Euler integration:

(5) $S(t) = S(t-DT) + DT \cdot (RIN(t-DT) - ROUT(t-DT))$.

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates, R_i, i=1, ..., k, should be initialized to:

(6) $R_i(0) = S(0)/D, \qquad i=1, \ldots, k$

where S(0) is the initial number of trees in the gestation delay.

If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to:

(7) $R_i(0) = ROUT(0) = RIN(0), \qquad i=1, \ldots, k.$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

$$(8) \quad S(0) = \frac{D}{k}\sum_{i=1}^{k} R_i(0).$$

If a model contains several delays, stability considerations dictate that:

$$(9) \quad DT < \min_j\left[\frac{D_j}{k_j}\right].$$

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes:

$$(10) \quad DT \leq \min_j\left[\frac{D_j}{2k_j}\right]$$

where j indexes all the delays in a model. Alternatively, if (10) requires DT to be too small for efficient computation, a larger DT may be used for the model as a whole by using routine DELF in place of DELS for those delays whose $D_j/2k_j$ ratios require a smaller DT.

DELS is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DELS (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample Run

The sample run assumes a third-order, three-year gestation delay (i.e., k=3 and D=3). No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter.

Under these conditions, we can use (1) and (4) to solve analytically for the output and storage of the gestation stage and compare these with the simulated output and storage from (2) and (5). The analytical solutions for ROUT and S are:

(11) $ROUT(t) = 100[1 - e^{-t}(\frac{1}{2}t^2 + t + 1)]$

(12) $S(t) = 100[3 - e^{-t}(\frac{1}{2}t^2 + 2t + 3)]$.

Results, including the per cent deviations of the simulated solutions from the analytical solutions, are tabulated in Figure 3 for a 10-year simulation, with DT = .25 years.

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
0.0	0.00	0.00	0.00	0.00	0.00	0.00
1.0	8.03	5.08	-36.76	97.67	99.61	1.99
2.0	32.33	32.15	-.58	178.20	185.43	4.06
3.0	57.68	60.93	5.64	232.79	241.93	3.93
4.0	76.19	80.29	5.38	265.20	272.94	2.92
5.0	87.53	90.87	3.81	282.82	288.13	1.88
6.0	93.80	96.02	2.36	291.82	295.02	1.10
7.0	97.04	98.34	1.34	296.22	297.98	.60
8.0	98.62	99.33	.71	298.29	299.20	.31
9.0	99.38	99.74	.36	299.24	299.69	.15
10.0	99.72	99.90	.18	299.67	299.88	.07

STOP
OK-

Figure 3

DELS Sample Run Output

III. Program Information

A. Program Description

DELS was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU Hustler 2 L239 LSD 32--an extension of CDC Scope 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 62 decimal words
CP compile time: .28 seconds
CP execution time (for sample run): .31 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DELS is structured as a single subroutine, DELS, which simulates a continuous (distributed) delay process and computes storage in the delay. No COMMON blocks are used.

B. Program implementation

All input is transmitted through the argument list. Input variables DELAY (called D in section II above), DT, and K require values which must remain constant during the simulation of a particular delay; STRG and R(I) require only initial values, as they are subsequently recomputed within DELS. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I) and STRG (called S in section II above). See the list of the sample run executive program for an example of the call statement.

For DELS to operate properly, the value of DT must be such that $0 < DT \leq DELAY/2K$, DELAY must be > 0, and K must be an integer ≥ 1. When

K = 1, the delay can be simulated more efficiently by a single statement (in addition to the STRG equation) in the calling program than by this routine:

$$(13) \quad ROUT(t) = ROUT(t-DT) + DT \cdot \frac{K}{2 \cdot DELAY} (RIN(t-DT) - ROUT(t-DT)).$$

C. Program lists

The computer programs for the sample run and for DELS are listed below:

```
      PROGRAM DELTST(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DELAY / 3. /
      DATA K / 3 /
      DATA DT / .25 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA STRG / 0. /
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        CALL DELS(RIN,ROUT,R,STRG,DELAY,DT,K)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      AROUT = 100.*(1.-EXP(-T)*(T*T/2.+T+1.))
        ASTRG = 100.*(3.-EXP(-T)*(T*T/2.+2.*T+3.))
        PDOUT = 100.*(ROUT-AROUT)/AROUT
        IF(AROUT .EQ. 0.) PDOUT = 0.
        PDSTRG = 100.*(STRG-ASTRG)/ASTRG
        IF(ASTRG .EQ. 0.) PDSTRG = 0.
        WRITE(2,91) T,AROUT,ROUT,PDOUT,ASTRG,STRG,PDSTRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1      ANALYTICAL  SIMULATED  PER CENT  ANALYTICAL  SIMUL*
     +        *ATED  PER CENT*/* TIME   OUTPUT     OUTPUT    DEVIATION *
     +        *  STORAGE    STORAGE   DEVIATION*/)
91    FORMAT(* *F4.1,F10.2,5F11.2)
      END
```

```
      SUBROUTINE DELS(RIN,ROUT,R,STRG,DELAY,DT,K)

      TITLE - DISTRIBUTED DELAY WITH STORAGE

      VERSION - 1A

      DATE - 1 NOV 1974

      PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS WITH
                STORAGE

      INPUT REQUIREMENTS

          VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DELAY, DT, K,
                     R(I), RIN, STRG

          VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN
                    -- DELAY, DT, K
          VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN
                    -- RIN
          VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES
                    -- R(I), STRG

      OUTPUT VARIABLES

         VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT,
                    STRG

      VARIABLE DEFINITIONS

          DELAY - LENGTH OF DELAY IN TIME UNITS.

          DT - SIMULATION TIME INCREMENT IN TIME UNITS.

          K - ORDER OF DELAY. UNITLESS.

          R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

          RIN - INPUT RATE IN UNITS/UNIT TIME.

          ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

          STRG - STORAGE IN UNITS.

      DIMENSION R(1)
      STRG = STRG + DT*(RIN-ROUT)
      A = FLOAT(K)*DT/DELAY
      IF(K .EQ.1) GO TO 20
      KM1 = K - 1
      DO 10 I=1,KM1
        R(I) = R(I) + A*(R(I+1)-R(I))
10    CONTINUE
20    R(K) = R(K) + A*(RIN-R(K))
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald L. Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II, Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch, "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria," IEEE Transactions on Systems, Man, and Cybernetics, SMC-2 (Sept., 1972) 472-486.

[3] Llewellyn, Robert W., FORDYN: An Industrial Dynamics Simulator (Raleigh, North Carolina, 1965).

[4] Forrester, Jay W., Industrial Dynamics (Cambridge: M.I.T. Press, 1961).

[5] Manetsch and Park, Op. cit., Chapter 10.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELF	Distributed Delay with Finer Time Increment	routine	1A 1/11/74

DELF Abstract

DELF simulates a distributed time delay process, computes the storage associated with the flow through the delay, and automatically insures simulation stability. It is useful for modeling flow delays inherent in such aggregated processes as investment, production, transportation, and demographic processes.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The input to DELF is the unlagged value or flow into the delay, and the output is the lagged value or flow out of the delay. In addition, DELF computes storage in the delay by integrating the net flow into the delay. DELF insures simulation stability by automatically decreasing the integration step size when the time constant is small relative to the order of the delay.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELF	Distributed Delay with Finer Time Increment	routine	1A 1/11/74

DELF Documentation

Contents

I. Process Description 1

II. Technical Description 2

A. Mathematical model 2

B. Sample run 8

III. Program Information 11

A. Program description 11

B. Program implementation 11

C. Program lists 12

1. Of sample run executive 13

2. Of component 14

Bibliographic References 16

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times; so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time, production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DELF simulates this process.

In the technical description which follows, reference is made to the use of DELF to model the gestation stage of cocoa trees, where the input is tree plantings and the outputs are trees becoming productive and the number of trees in the gestation stage.

II. Technical Description

A. Mathematical model

Aside from its computation of storage, DELF is identical mathematically to the DELAY2 routine of [1], produces the same numerical results as the DELAY routine of [3], and is a generalization of the DELAY3 routine of [4].

The kth-order differential equation modeling a distributed delay can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [5] (Figure 1).

The delay times for individual trees entering the gestation delay follow a kth-order gamma distribution with mean D. k determines the variance σ^2 of the gamma distributed delay times as $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance; i.e., a discrete delay of length D.

The ith stage of the delay--where i=1, 2, ..., k--is modeled by the first-order differential equation:

$$(1) \quad \frac{D}{k}\frac{dR_i(t)}{dt} + R_i(t) = R_{i+1}(t)$$

which can be solved numerically using Euler's integration approximation by

$$(2) \quad R_i(t) = R_i(t-DT) + DT\cdot\frac{k}{D}[R_{i+1}(t-DT) - R_i(t-DT)]$$

where, for the kth stage of the delay, R_{k+1} + RIN and where:

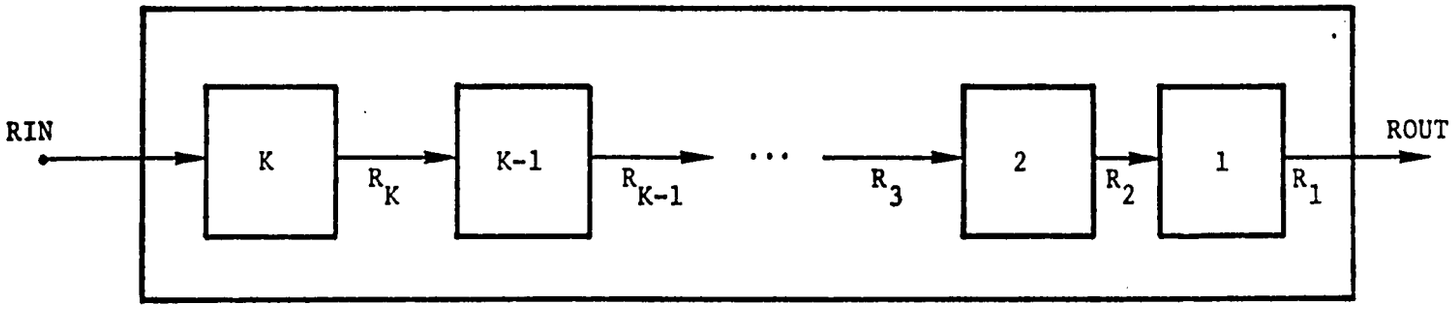

Figure 1

K^{th}-Order Distributed Delay Process

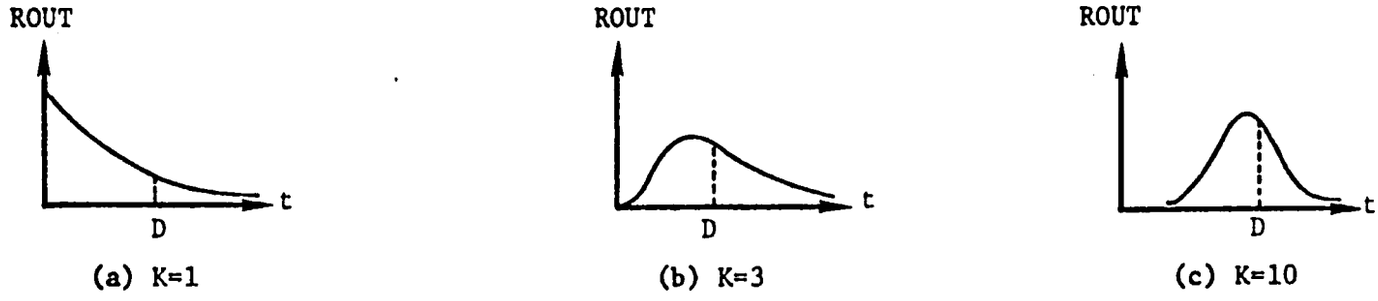

Figure 2

k^{th}-Order Delay Zero-State Unit Impulse Response

R_i = rate out of the ith stage (trees/year)

RIN = planting rate (Figure 1) (trees/year)

k = the order of the delay process

D = the mean delay time (years)

DT = the integration step size, the simulation time increment (years).

DELF does not use (2), however. In order to insure stability, DELF divides the DT time interval into IDT subintervals, where IDT is computed in (12). Thus, integrating over the subintervals of length EDT = DT/IDT, (2) becomes:

(3) $R_i(\tau) = R_i(\tau\text{-EDT}) + \text{EDT}\cdot\frac{k}{D}\cdot[R_{i+1}(\tau\text{-EDT}) - R_i(\tau\text{-EDT})]$

for $\tau = t - \text{DT} + \text{EDT}, \ldots, t - \text{EDT}, t$, and where, for the kth stage of the delay, $R_{k+1}(\tau - \text{EDT}) = \text{RIN}(t - \text{DT})$ for all τ.

The output of the delay (trees becoming productive--Figure 1) is:

(4) $\text{ROUT}(t) = R_1(t)$.

Finally, the storage in the delay associated with the flow through the delay (the number of trees in the gestation stage) is computed from the differential equation:

(5) $\frac{dS(t)}{dt} = \text{RIN}(t) - \text{ROUT}(t)$

using Euler integration:

(6) $S(\tau) = S(\tau\text{-EDT}) + \text{EDT}\cdot[\text{RIN}(t\text{-DT}) - \text{ROUT}(\tau\text{-EDT})]$

for $\tau = t - \text{DT} + \text{EDT}, \ldots, t - \text{EDT}, t$.

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time

zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates, R_i, i=1, ..., k, should be initialized to:

(7) $R_i(0) = S(0)/D, \qquad i=1, \ldots, k$

where S(0) is the initial storage (i.e., number of trees) in the gestation delay. If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to:

(8) $R_i(0) = ROUT(0) = RIN(0), \qquad i=1, \ldots, k.$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

(9) $S(0) = \frac{D}{k}\sum_{i=1}^{k} R_i(0).$

If a model contains several delays, for each delay j stability considerations dictate that:

(10) $\frac{DT}{IDT_j} < \frac{2D_j}{k_j}.$

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes:

(11) $\frac{DT}{IDT_j} \leq \frac{D_j}{2k_j}.$

With DT, D_j, and k_j already specified, DELF computes IDT_j so that (11) holds:

$$(12) \quad IDT_j = MAX\{IDTU_j, [1 + 2 \cdot DT \cdot \frac{k_j}{D_j}]\}$$

where the operation [•] sets IDT_j equal to the largest integer in the value of the expression in brackets and MAX{a,b} sets IDT_j equal to the maximum of a and b. The user may specify an IDT greater than 1 to increase integration accuracy (even if not required for stability) by specifying IDTU greater than 1; i.e., the number of desired subintervals. Thus, DT will have only one subinterval of length DT unless stability requires otherwise.

While DELF accurately simulates a distributed delay (e.g., the gestation stage of a cocoa tree population in this case), non-trivial inaccuracies relating to non-conservation of flow arise when $IDT_j > 1$ and when the output $ROUT_j(t)$ (equation (4)) is used elsewhere in a larger model; e.g., integrated or used as the input rate to another delay. In this example, the problem arises when the rate of trees leaving the gestation stage is used as the input to a second delay representing trees with increasing yields up to full maturity. For a full discussion of the problem and alternative ways to modify DELF to handle it, see [6].

DELF is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DELF (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample Run

The sample run assumes a third-order, three-year gestation delay (i.e., k=3 and D=3). No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter.

Under these conditions, we can use (1) and (5) to solve analytically for the output and storage of the gestation stage and compare these with the simulated output and storage from (2) and (6). The analytical solutions for ROUT and S are:

(13) $ROUT(t) = 100[1 - e^{-t}(\frac{1}{2}t^2 + t + 1)]$

(14) $S(t) = 100[3 - e^{-t}(\frac{1}{2}t^2 + 2t + 3)]$.

Results from two sample runs are tabulated in Figure 3. This includes the per cent deviation of the simulated solution from the analytical solution. Each run is for 10 years, with DT = .25 years. IDTU has a value of 1 for the first run and a value of 4 for the second run. This

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
.0	.00	.00	.00	.00	.00	.00
1.0	8.03	5.08	-36.76	97.67	99.61	1.99
2.0	32.33	32.15	-.58	178.20	185.43	4.06
3.0	57.68	60.93	5.64	232.79	241.93	3.93
4.0	76.19	80.29	5.38	265.20	272.94	2.92
5.0	87.53	90.87	3.81	282.82	288.13	1.88
6.0	93.80	96.02	2.36	291.82	295.02	1.10
7.0	97.04	98.34	1.34	296.22	297.98	.60
8.0	98.62	99.33	.71	298.29	299.20	.31
9.0	99.38	99.74	.36	299.24	299.69	.15
10.0	99.72	99.90	.18	299.67	299.88	.07

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
.0	.00	.00	.00	.00	.00	.00
1.0	8.03	7.42	-7.59	97.67	98.22	.57
2.0	32.33	32.32	-.03	178.20	179.92	.96
3.0	57.68	58.41	1.26	232.7[illegible]	234.93	.92
4.0	76.19	77.13	1.23	265.20	267.06	.70
5.0	87.53	88.34	.91	282.82	284.14	.47
6.0	93.80	94.36	.59	291.82	292.65	.28
7.0	97.04	97.38	.35	296.22	296.69	.16
8.0	98.62	98.82	.20	298.29	298.55	.09
9.0	99.38	99.48	.11	299.24	299.37	.04
10.0	99.72	99.78	.05	299.67	299.74	.02

Figure 3

DELF Sample Run Output

demonstrates the increased accuracy that can be obtained by subdividing the time interval.

III. Program Information

A. Program Description

DELF was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU HUSTLER 2 L239 LSD 32--an extension of CDC SCOPE 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 93 decimal words
CP compile time: .40 seconds
CP execution time (for sample run): .44 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DELF is structured as a single subroutine, DELF, which simulates a continuous (distributed delay process and computes storage in the delay. No COMMON blocks are used.

B. Program implementation

All input is transmitted through the argument list. Input variables DEL (called D in section II above), DT, and require values which must remain constant during the simulation of a particular delay; STRG and R(I) require only initial values, as they are subsequently recomputed within DELF. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I) and STRG (called S in section II above). See the list of the sample run executive program for an example of the call statement.

For DELF to operate properly, DT and DEL must be > 0 and K must be an integer $\geq$ 1. In addition, IDTU should equal 1 unless greater integration accuracy is desired, in which case IDTU will be an integer > 1.

C. Program lists

The computer programs for the sample run and for DELF are listed below:

```
      PROGRAM DELTST(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DELAY / 3. /
      DATA K / 3 /
      DATA DT,IDTU / .25, 1 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA STRG / 0. /
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        CALL DELF(RIN,ROUT,R,STRG,DELAY,DT,IDTU,K)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      AROUT = 100.*(1.-EXP(-T)*(T*T/2.+T+1.))
        ASTRG = 100.*(3.-EXP(-T)*(T*T/2.+2.*T+3.))
        PDOUT = 100.*(ROUT-AROUT)/AROUT
        IF(AROUT .EQ. 0.) PDOUT = 0.
        PDSTRG = 100.*(STRG-ASTRG)/ASTRG
        IF(ASTRG .EQ. 0.) PDSTRG = 0.
        WRITE(2,91) T,AROUT,ROUT,PDOUT,ASTRG,STRG,PDSTRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1      ANALYTICAL  SIMULATED  PER CENT  ANALYTICAL  SIMUL*
     +        *ATED  PER CENT*/* TIME   OUTPUT     OUTPUT    DEVIATION *
     +        *  STORAGE    STORAGE   DEVIATION*/)
91    FORMAT(* *F4.1,F10.2,5F11.2)
      END
```

```
SUBROUTINE DELF(RIN,ROUT,R,STRG,DEL,DT,IDTU,K)

TITLE - DISTRIBUTED DELAY WITH FINER TIME INCREMENT

VERSION - 1A

DATE - 1 NOV 1974

PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS WITH
          FINER TIME INCREMENT

INPUT REQUIREMENTS

    VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DEL, DT,
                IDTU, K, R(I), RIN, STRG

    VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN
              -- DEL, DT, IDTU, K
    VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN
              -- RIN
    VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES
              -- R(I), STRG

OUTPUT VARIABLES

   VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT,
              STRG

VARIABLE DEFINITIONS

    DEL - LENGTH OF DELAY IN TIME UNITS.

    DT - SIMULATION TIME INCREMENT IN TIME UNITS.

    IDT - NUMBER OF SUBDIVISIONS OF DT REQUIRED. UNITLESS.

    IDTU - USER-SPECIFIED NUMBER OF SUBDIVISIONS OF DT. UNITLESS.

    K - ORDER OF DELAY. UNITLESS.

    R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

    RIN - INPUT RATE IN UNITS/UNIT TIME.

    ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

    STRG - STORAGE IN UNITS.

DIMENSION R(1)
FK = FLOAT(K)
IDT = 1. + 2.*FK*DT/DEL
IF(IDT .LT. IDTU) IDT = IDTU
```

```
      FIDT = FLOAT(IDT)
      A = FK*DT/(DEL*FIDT)
      KM1 = K-1
      DO 20 J=1,IDT
        STRG = STRG + DT*(RIN-R(1))/FIDT
        IF(K .EQ. 1) GO TO 15
        DO 10 I=1,KM1
          R(I) = R(I) + A*(R(I+1)-R(I))
10      CONTINUE
15      R(K) = R(K) + A*(RIN-R(K))
20    CONTINUE
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald L. Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II, Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch, "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria," IEEE Transactions on Systems, Man, and Cybernetics, SMC-2 (Sept., 1972) 472-486.

[3] Llewellyn, Robert W., FORDYN: An Industrial Dynamics Simulator (Raleigh, North Carolina, 1965).

[4] Forrester, Jay W., Industrial Dynamics (Cambridge: M.I.T. Press, 1961).

[5] Manetsch and Park, Op. cit., Chapter 10.

[6] Abkin, M. H. and C. Wolf, "Accuracy in the Simulation of Distributed Delay Processes in Models of Socioeconomic Systems," Project Working Paper 74-2. (East Lansing: Michigan State University, November 1974.)

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELLF	Distributed Delay with Storage Losses	routine	1A 1/11/74

DELLF Abstract

DELLF simulates a distributed time delay process with storage losses (or additions), computes the storage associated with the flow through the delay, and automatically insures simulation stability. It is useful for modeling flow delays inherent in such aggregated processes as capital formation, transportation, and demographic processes. In these cases, storage losses could refer to disinvestment, spoilage, and deaths or migration, respectively.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The inputs to DELLF are the unlagged value, or flow into the delay, and the proportional storage loss rate. Outputs are the lagged value, or flow out of the delay, and the storage resulting from integration of net flow into the delay. Finally, DELLF insures simulation stability by automatically decreasing the integration step size when the time constant is small relative to the order of the delay.

CLASS Document

1. Symbolic Name	2. Name	3. Type	4. Version No. and Date
DELLF	Distributed Delay with Storage Losses	routine	1A 1/11/74

DELLF Documentation

Contents

I. Process Description 1

II. Technical Description 2

A. Mathematical model 2

B. Sample run 7

III. Program Information 11

A. Program description 11

B. Program implementation 11

C. Program lists 12

1. Of sample run executive 13

2. Of component 14

Bibliographic References 16

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DELLF simulates this process.

It is frequently the case that storage losses or additions occur in the course of the delay process. Notable examples include population migrations, tree removals, deaths, production losses, distribution losses, etc. DELLF is specifically designed to incorporate this factor in the delay process.

In the technical description which follows, reference is made to the use of DELLF to model the gestation stage of cocoa trees, where the inputs are tree plantings and failure rates, and the outputs are trees becoming productive and the number of trees in the gestation stage.

II. Technical Description

A. Mathematical model

The kth-order differential equation modeling a distributed delay with storage losses (or additions) can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [3] (Figure 1). The delay times for individual trees entering the gestation delay (assuming no losses) follow a kth-order gamma distribution with mean D. k determines the variance σ of the gamma distribution with mean D. $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real-world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance--i.e., a discrete delay of length D.

For the ith stage of the delay--where i = 1, 2, ..., k--Q_i is the storage (number of trees) and [3]

$$(1) \quad Q_i(t) = \frac{D}{k} R_i(t)$$

where:

R_i = rate out of the ith stage (trees/year)

D = the mean delay time (years)

k = the order of the delay process.

The rate of change of Q_i is the net flow into the ith stage:

$$(2) \quad \frac{dQ_i(t)}{dt} = R_{i+1}(t) - R_i(t) - L_i(t)$$

where R_{k+1} = RIN, the planting rate into the delay (figure 1), and L_i is the

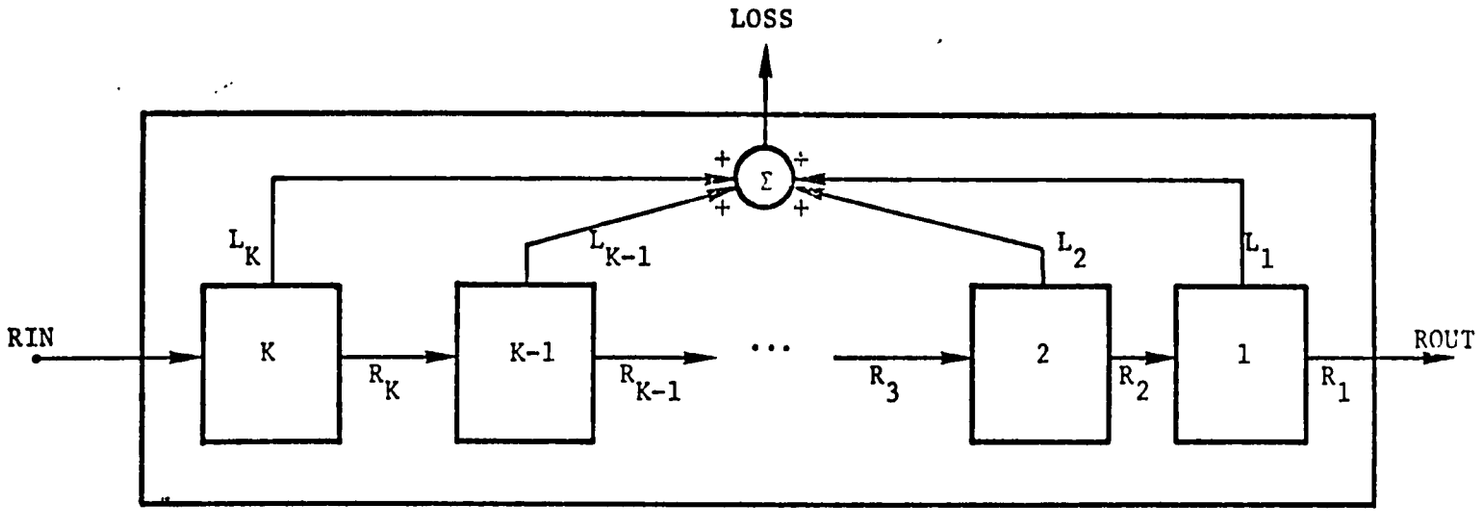

Figure 1

K^{th}-Order Distributed Delay Process with Storage Losses

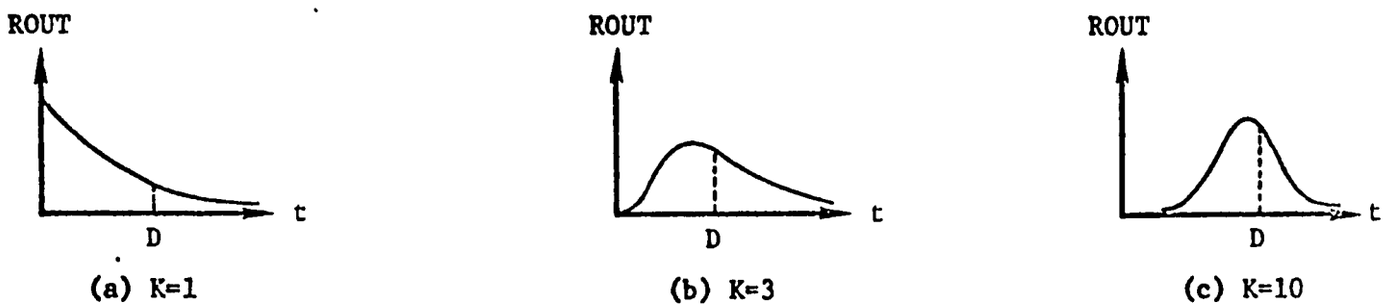

Figure 2

K^{th}-Order Delay Zero-State Unit Impulse Response

storage loss rate from the ith stage. If $L_i(t) = PLR(t) \cdot Q_i(t)$ where PLR is the proportional loss rate for the total storage in the whole delay (i.e., the losses are apportioned proportionately to each stage), then:

$$(3) \quad \frac{dQ_i(t)}{dt} = R_{i+1}(t) - R_i(t) - PLR(t) \cdot Q_i(t)$$

$$= R_{i+1}(t) - R_i(t) \cdot (1 + \frac{D}{k} \cdot PLR(t)).$$

But from (1), $\frac{dQ_i(t)}{dt} = \frac{D}{k} \frac{dR_i(t)}{dt}$ also, so:

$$(4) \quad \frac{D}{k} \frac{dR_i(t)}{dt} + R_i(t) \cdot (1 + \frac{D}{k} \cdot PLR(t)) = R_{i+1}(t)$$

is the first-order differential equation which models the ith stage of a kth-order delay with storage losses. This can be solved numerically using Euler's integration approximation by:

$$(5) \quad R_i(t) = R_i(t-DT) + DT \cdot \frac{k}{D} \cdot [R_{i+1}(t-DT) - R_i(t-DT) \cdot (1 + \frac{D}{k} \cdot PLR(t-DT))]$$

where DT is the integration step size, the simulation time increment. DELLF does not use (5), however. In order to insure stability, DELLF divides the DT time interval into IDT subintervals, where IDT = IDT(t) is computed in (15). Thus, integrating over the subintervals of length EDT = DT/IDT(t), (5) becomes:

$$(6) \quad R_i(\tau) = R_i(\tau-EDT) + EDT \cdot \frac{k}{D} \cdot [R_{i+1}(\tau-EDT) - R_i(\tau-EDT) \cdot (1 + \frac{D}{k} \cdot PLR(t-DT))]$$

for $\tau = t - DT + EDT, \ldots, t - EDT, t$, and where, for the kth stage of the delay, $R_{k+1}(\tau-EDT) = RIN(t-DT)$ for all τ.

The output of the delay (trees becoming productive--Figure 1) is

(7) $ROUT(t) = R_1(t)$

Finally, the storage in the delay associated with the flow through the delay (the number of trees in the gestation stage) is computed from the differential equation:

(8) $$\frac{dS(t)}{dt} = \sum_{i=1}^{k} \frac{dQ_i(t)}{dt}$$

$$= RIN(t) - ROUT(t) - PLR(t) \cdot S(t)$$

using Euler integration:

(9) $S(\tau) = S(\tau-EDT) + EDT \cdot [RIN(t-DT) - ROUT(\tau-EDT) - PLR(t-DT) \cdot S(\tau-EDT)]$

for $\tau = t - DT + EDT, \ldots, t - EDT, t$.

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates R_i, i=1, ..., k, should be initialized to:

(10) $R_i(0) = S(0)/D, \qquad i=1, \ldots, k.$

where S(0) is the initial storage (i.e., number of trees) in the gestation delay. If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to:

(11) $R_i(0) = ROUT(0) = RIN(0), \qquad i=1, \ldots, k.$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

$$(12) \quad S(0) = \frac{D}{k}\sum_{i=1}^{k} R_i(0).$$

If a model contains several delays with losses, for each delay j stability considerations (assuming PLR(t) = PLR, a constant) dictate that:

$$(13) \quad \frac{DT}{IDT_j(t)} < \frac{D_j}{k_j + PLR_j(t-DT)\cdot D_j}.$$

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes:

$$(14) \quad \frac{DT}{IDT_j(t)} \leq \tfrac{1}{2}\cdot\frac{D_j}{k_j + PLR_j(t-DT)\cdot D_j}.$$

With DT, PLR, D, and k already known, DELLF computes IDT_j so that (14) holds:

$$(15) \quad IDT_j(t) = MAX\{IDTU_j, [1 + 2\cdot DT\cdot(PLR_j(t-DT) + \frac{k_j}{D_j})]\}$$

where the operation [•] sets IDT_j equal to the largest integer in the value of the expression in brackets and MAX(a, b) sets IDT_j equal to the larger of a and b. The user may specify an IDT greater than 1 to increase integration accuracy (even if not required for stability) by specifying IDTU greater than 1, i.e., the derived number of subintervals.

While DELLF accurately simulates a distributed delay (e.g., the gestation stage of a cocoa tree population in this case), non-trivial inaccuracies relating to non-conservation of flow arise when $IDT_j > 1$

and when the output $ROUT_j(t)$ (equation (4)) is used elsewhere in a larger model; e.g., integrated or used as the input rate to another delay. In this example, the problem arises when the rate of trees leaving the gestation stage is used as the input to a second delay representing trees with increasing yields up to full maturity. For a full discussion of the problem and alternative ways to modify DELLF to handle it, see [4].

DELLF is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DELLF (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample Run

The sample run assumes a third-order, three-year gestation delay (i.e., k=3 and D=3) with a constant failure rate PLR(t) = 0.1/year. No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter.

Under these conditions, we can use (4) and (8) to solve analytically for the output and storage of the gestation stage and compare these with

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

the simulated output and storage from (5) and (9). The analytical solutions for ROUT and S are:

$$(16) \quad ROUT(t) = \frac{100}{(1 + PLR)^3} \left[1 - A \left(\frac{t^2}{2} (1 + PLR)^2 + t(1 + PLR) + 1 \right) \right]$$

$$(17) \quad S(t) = \frac{100}{(1 + PLR)} (1 - A) + \frac{100}{(1 + PLR)^2} (1 - A) - \frac{100At}{(HPLR)} + ROUT(t).$$

Results, including the per cent deviations of the simulated solutions from the analytical solutions, are tabulated in Figure 3 for a 10-year simulation, with DT = .25 years.

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
0.0	0.00	0.00	0.00	0.00	0.00	0.00
1.0	7.48	4.96	-33.69	93.00	95.92	3.14
2.0	28.35	29.89	1.92	162.52	170.05	4.63
3.0	48.13	51.29	6.57	205.22	213.25	3.91
4.0	61.22	64.48	5.32	228.18	234.10	2.59
5.0	68.49	70.79	3.35	239.48	243.05	1.49
6.0	72.13	73.47	1.86	244.70	246.61	.78
7.0	73.83	74.52	.94	247.01	247.95	.38
8.0	74.58	74.92	.45	248.00	248.43	.17
9.0	74.91	75.06	.20	248.41	248.60	.08
10.0	75.04	75.11	.09	248.58	248.66	.03

Figure 3

DELLF Sample Run Output

III. Program Information

A. Program description

DELLF was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU HUSTLER 2 L239 LSD 32--an extension of CDC SCOPE 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 103 decimal words
CP compile time: .47 seconds
CP execution time (for sample run): .49 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DELLF is structured as a single subroutine, DELLF, which simulates a continuous (distributed) delay process with storage losses and/or additions and computer storage in the delay. No COMMON blocks are used.

B. Program implementation

All input is transmitted through the argument list. Input variables DEL (called D in section II above), DT, IDTU and K require values which must remain constant during the simulation of a particular delay; STRG and R(I) require only initial values, as they are subsequently recomputed within DELLF. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I) and STRG (called S in section II above). See the list of the sample run executive program for an example of the call statement.

For DELLF to operate properly, DT and DEL must be > 0, and K and IDTU must be integers $\geq$ 1.

C. Program lists

The computer programs for the sample run and for DELLF are listed below:

```
      PROGRAM DELTST(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DELAY / 3. /
      DATA K / 3 /
      DATA PLR / .1 /
      DATA DT,IDTU / .25, 1 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA STRG / 0. /
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        CALL DELLF(RIN,ROUT,R,STRG,PLR,DELAY,DT,IDTU,k)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      EXP1 = EXP(-(1.+PLR)*T)
        AROUT = 100.*(1.-EXP1*(((1.+PLR)*T)**2/2.+(1.+PLR)*T+1.))/
     +          (1.+PLR)**3
        ASTRG = 100.*(1.-EXP1)/(1.+PLR) + 100.*(1.-EXP1)/(1.+PLR)**2 -
     +          100.*T*EXP1/(1.+PLR) + AROUT
        PDOUT = 100.*(ROUT-AROUT)/AROUT
        IF(AROUT .EQ. 0.) PDOUT = 0.
        PDSTRG = 100.*(STRG-ASTRG)/ASTRG
        IF(ASTRG .EQ. 0.) PDSTRG = 0.
        WRITE(2,91) T,AROUT,ROUT,PDOUT,ASTRG,STRG,PDSTRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1      ANALYTICAL  SIMULATED  PER CENT  ANALYTICAL  SIMUL*
     +        *ATED  PER CENT*/* TIME   OUTPUT     OUTPUT    DEVIATION *
     +        *  STORAGE    STORAGE   DEVIATION*/)
91    FORMAT(* *F4.1,F10.2,5F11.2)
      END
```

```
SUBROUTINE DELLF(RIN,ROUT,R,STRG,PLR,DEL,DT,IDTU,K)

TITLE - DISTRIBUTED DELAY WITH STORAGE LOSSES

VERSION - 1A

DATE - 1 NOV 1974

PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS WITH
          STORAGE LOSSES

INPUT REQUIREMENTS

    VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DEL, DT,
              IDTU, K, PLR, R(I), RIN, STRG

    VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN
              -- DEL, DT, IDTU, K
    VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN
              -- PLR, RIN
    VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES
              -- R(I), STRG

OUTPUT VARIABLES

    VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT,
              STRG

VARIABLE DEFINITIONS

    DEL - LENGTH OF DELAY IN TIME UNITS.

    DT - SIMULATION TIME INCREMENT IN TIME UNITS.

    IDT - NUMBER OF SUBDIVISIONS OF DT REQUIRED. UNITLESS.

    IDTU - USER-SPECIFIED NUMBER OF SUBDIVISIONS OF DT. UNITLESS.

    K - ORDER OF DELAY. UNITLESS.

    PLR - PROPORTIONATE LOSS RATE IN PROPORTION/UNIT TIME.

    R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

    RIN - INPUT RATE IN UNITS/UNIT TIME.

    ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

    STRG - STORAGE IN UNITS.
```

```
      DIMENSION R(1)
      FK = FLOAT(K)
      IDT = 1. + 2.*DT*(FK/DEL + PLR)
      IF(IDT .LT. IDTU) IDT = IDTU
      FIDT = FLOAT(IDT)
      A = FK*DT/(DEL*FIDT)
      B = 1.+PLR*DEL/FK
      KM1 = K-1
      DO 20 J=1,IDT
        STRG = STRG + DT*(RIN-R(1)-PLR*STRG)/FIDT
        IF(K .EQ. 1) GO TO 15
        DO 10 I=1,KM1
          R(I) = R(I) + A*(R(I+1)-B*R(I))
10      CONTINUE
15      R(K) = R(K) + A*(RIN-B*R(K))
20    CONTINUE
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald L. Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II, Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch, "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria," IEEE Transactions on Systems, Man, and Cybernetics, SMC-2 (Sept., 1972) 472-486.

[3] Manetsch and Park, Op. cit., Chapter 10.

[4] Abkin, M. H. and C. Wolf, "Accuracy in the Simulation of Distributed Delay Processes in Models of Socioeconomic Systems," Project Working Paper 74-2. (East Lansing: Michigan State University, November 1974.)

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELVF	Distributed Delay with Variable Delay Time	routine	1A 1/11/74

DELVF Abstract

DELVF simulates a distributed time delay process with variable delay time, computes the storage associated with the flow through the delay, and automatically insures simulation stability. It is useful for modeling flow delays inherent in such aggregated processes as capital formation, transportation and demographic processes. In these cases, delay times could vary in response to, for example, changing technologies and disease control, respectively.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The inputs to DELVF are the unlagged value, or flow into the delay, and the current value of the mean lag time. Outputs are the lagged value, or flow out of the delay, and the storage resulting from integration of net flow into the storage resulting from integration of net flow into the delay. Finally, DELVF insures simulation stability by automatically decreasing the integration step size when the time constant is small relative to the order of the delay.

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELVF	Distributed Delay with Variable Delay Time	routine	1A 1/11/74

DELVF Documentation

Contents

I. Process Description 1

II. Technical Description 3

A. Mathematical model 3

B. Sample run 9

III. Program Information 12

A. Program description 12

B. Program implementation 12

C. Program lists 13

1. Of sample run executive 14

2. Of component 15

Bibliographic References 17

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times; so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DELVF simulates this process.

It is frequently the case that the length of the delay varies over time as a function of other variables in the system. For example, delays associated with the diffusion of innovations may be functions of the profitability of the innovations and/or of the communication infrastructure. Or production delays may be functions of inventory, order backlogs, overtime wage rates and/or machine reliability. DELVF is specifically designed [3] to incorporate variable delays in the delay process.

In the technical description which follows, reference is made to the use of DELVF to model the gestation stage of cocoa trees, where the inputs are tree plantings and gestation time, and the outputs are trees becoming productive and the number of trees in the gestation stage.

II. Technical Description

A. Mathematical model

The kth-order differential equation modeling a distributed delay with variable delay time can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [3] (Figure 1). The delay times for individual trees entering the gestation delay (assuming constant delay time) follow a kth-order gamma distribution with mean D. k determines the variance σ^2 of the gamma distributed delay times as $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real-world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance--i.e., a discrete delay of length D. Aside fro the computation of storage and the determination of the integration step size consistent with stability, DELVF is identical mathematically with the VDEL routine of [3].

For the ith stage of the delay--where i=1, 2, ..., k--Q_i is the storage (number of trees) and [3]:

$$(1) \quad Q_i(t) = \frac{D(t)}{k} R_i(t)$$

where

R_i = rate out of the ith stage (trees/year)

D = the mean delay time (years)

k = the order of the delay process.

The rate of change of Q_i is the net flow into the ith stage:

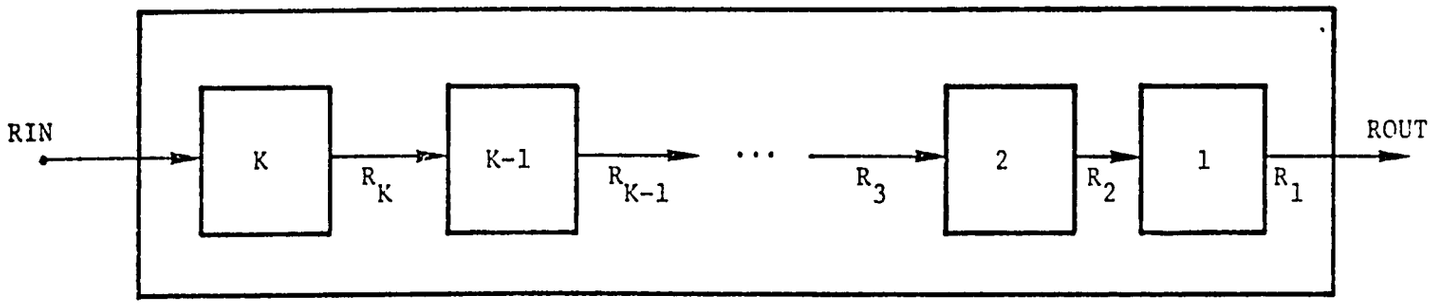

Figure 1

K^{th}-Order Distributed Delay Process

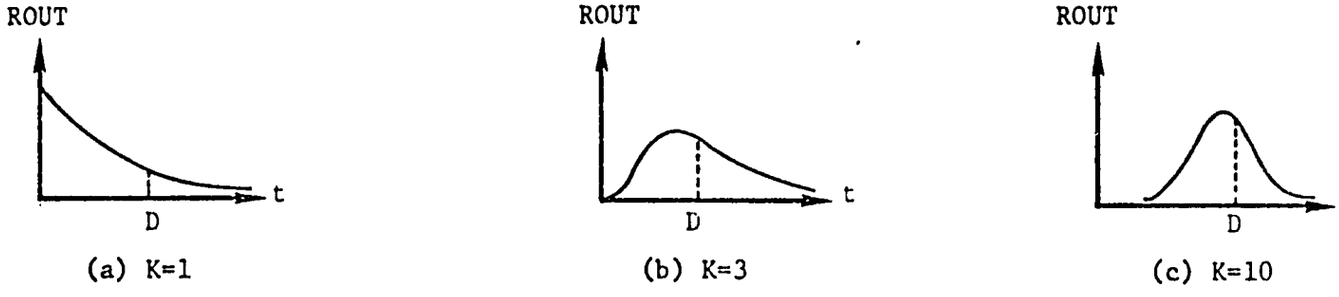

Figure 2

K^{th}-Order Delay Zero-State Unit Impulse Response

(2) $$\frac{dQ_i(t)}{dt} = R_{i+1}(t) - R_i(t)$$

where R_{k+1} = RIN, the planting rate into the delay (Figure 1). But from (1):

(3) $$\frac{dQ_i(t)}{dt} = \frac{1}{k}\left(D(t)\cdot\frac{dR_i(t)}{dt} + R_i(t)\cdot\frac{dD(t)}{dt}\right).$$

Thus, upon substituting and rearranging terms, the first-order differential equation modeling the ith stage of a kth-order delay with variable delay time becomes:

(4) $$\frac{D(t)}{k}\cdot\frac{dR_i(t)}{dt} + \left(1 + \frac{1}{k}\cdot\frac{dD(t)}{dt}\right)\cdot R_i(t) = R_{i+1}(t)$$

which can be solved numerically using Euler's integration approximation and taking $\frac{dD(t)}{dt} \approx \frac{D(t+DT) - D(t)}{DT}$

(5) $$R_i(t) = R_i(t-DT) + DT\cdot\frac{k}{D(t-DT)}\cdot[R_{i+1}(t-DT) - R_i(t-DT)\cdot(1+DD(t-DT))]$$

where $DD(t-DT) = \frac{1}{k}\cdot\frac{D(t) - D(t-DT)}{DT}$ and where DT is the integration step size, the simulation time increment. DELVF does not use (5), however. In order to insure stability, DELVF divides the DT time interval into IDT subintervals, where IDT = IDT(t) is computed in (14). Thus, integrating over the subintervals of length EDT = DT/IDT(t), (5) becomes:

(6) $$R_i(\tau) = R_i(\tau-EDT) + EDT\cdot\frac{k}{D(t-DT)}[R_{i+1}(\tau-EDT) - R_i(\tau-EDT)\cdot(1+DD(t-DT))]$$

for τ = t-DT + EDT, ..., t-EDT, t, and where, for the kth stage of the delay, $R_{k+1}(\tau-EDT)$ = RIN(t-DT) for all τ.

The output of the delay (trees becoming productive--Figure 1) is

(7) $ROUT(t) = R_1(t)$.

Finally, the storage in the delay associated with the flow through the delay (the number of trees in the gestation stage) is computed [3] as:

$$(8) \quad S(t) = \frac{D(t)}{k} \sum_{i=1}^{k} R_i(t).$$

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates R_i, i=1, ..., k, should be initialized to:

$$(9) \quad R_i(0) = S(0)/D, \qquad i=1, \ldots, k.$$

where S(0) is the initial storage (i.e., number of trees) in the gestation delay. If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to:

$$(10) \quad R_i(0) = ROUT(0) = RIN(0), \qquad i=1, \ldots, k.$$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

$$(11) \quad S(0) = \frac{D}{k} \sum_{i=1}^{k} R_i(0).$$

If a model contains several delays with variable delay times, for each delay j stability considerations (assuming D(t) ≐ D, a constant) dictate that:

$$(12) \quad \frac{DT}{TDT_j(t)} < \frac{2D_j(t-DT)}{k_j(1+DD(t-DT))} .$$

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes:

$$(13) \quad \frac{DT}{IDT_j(t)} \leq \frac{D_j(t-DT)}{2 \cdot k_j \cdot (1+DD(t-DT))}$$

where $DD(t) \approx \frac{1}{k} \frac{dD(t)}{dt}$ (see (5)). With DT, D, DD and k already known, DELVF computes IDT_j so that (13) holds:

$$(14) \quad IDT_j(t) = MAX\{IDTU_j, [1 + 2 \cdot DT \cdot \frac{k_j}{D_j(t-DT)} (1 + DD_j(t-DT))]\}$$

where the operation [•] sets IDT_j equal to the largest integer in the value of the expression in brackets; and MAX{a, b} sets IDT_j equal to the larger of a and b. The user may specify an IDT greater than 1 to increase integration accuracy (even if not required for stability) by specifying IDTU greater than 1; i.e., the desired number of subintervals.

While DELVF accurately simulates a distributed delay (e.g., the gestation stage of a cocoa tree population in this case), non-trivial inaccuracies relating to non-conservation of flow arise when $IDT_j > 1$ and when the output $ROUT_j(t)$ (equation (4)) is used elsewhere in a larger model; e.g., integrated or used as the input rate to another delay. In this example, the problem arises when the rate of trees leaving the gestation stage is used as the input to a second delay representing trees with increasing yields up to full maturity. For a full discussion of the problem and alternative ways to modify DELVF to handle it, see [4].

DELVF is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DELVF (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample Run

The sample run assumes a third-order, three-year gestation delay (i.e., k=3 and D=3) with a constant rate of change of the delay time $\frac{dD(t)}{dt} = .1$ years/year. No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter.

Under these conditions, we can use (4) and (8) to solve analytically for the output and storage of the gestation stage and compare these with the simulated output and storage from (5) and (8). The analytical solutions for ROUT and S are:

$$(15) \quad ROUT(t) = \frac{2700}{C^3}\left[1 - A^{-B} - BA^{-B}\ln A\left(1 + \frac{B}{2}\ln A\right)\right]$$

$$(16) \quad S(t) = A\left\{\frac{300}{C}\left[\left(1 - A^{-B}\right)\left(1 + \frac{3}{C}\right) - (B - 1)A^{-B}\ln A\right] + ROUT(t)\right\}$$

where $A = \frac{3 + \beta t}{3}$, $B = \frac{3 + \beta}{\beta}$, $C = \beta B = 3 + \beta$ and $\beta = \frac{dD(t)}{dt}$.

Results, including the per cent deviations of the simulated solutions from the analytical solutions, are tabulated in Figure 3 for a 10-year simulation, with DT = .25 years.

TIME	ANALYTICAL OUTPUT	SIMULATED OUTPUT	PER CENT DEVIATION	ANALYTICAL STORAGE	SIMULATED STORAGE	PER CENT DEVIATION
.0	.00	.00	.00	.00	.00	.00
1.0	7.55	4.87	-35.54	97.78	100.43	2.71
2.0	29.32	29.38	.21	179.90	187.85	4.42
3.0	51.35	54.44	6.01	239.18	248.97	4.09
4.0	67.41	71.25	5.69	279.26	287.79	3.06
5.0	77.53	80.78	4.19	306.36	312.63	2.05
6.0	83.43	85.76	2.78	325.60	329.78	1.28
7.0	86.73	88.25	1.75	340.35	342.98	.77
8.0	88.54	89.47	1.06	352.62	354.22	.45
9.0	89.51	90.07	.62	363.55	364.50	.26
10.0	90.03	90.35	.36	373.76	374.31	.15

Figure 3

DELVF Sample Run Output

III. Program Information

A. Program description

DELVF was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU HUSTLER 2 L239 LSD 32--an extension of CDC SCOPE 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 104 decimal words
CP compile time: .51 seconds
CP execution time (for sample run): .55 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DELVF is structured as a single subroutine, DELVF, which simulates a continuous (distributed) delay process with variable delay time and computer storage in the delay. No COMMON blocks are used.

B. Program implementation

All input is transmitted through the argument list. Input variables DT, IDTU and K require values which must remain constant during the simulation of a particular delay; DELP and R(I) require only initial values, as they are subsequently recomputed within DELVF. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I) and STRG (called S in section II above). See the list of the sample run executive program for an example of the call statement.

For DELVF to operate properly, DT and DEL(t) must be > 0 for all t, and K and IDTU must be integers $\geq$ 1.

C. Program lists

The computer programs for the sample run and for DELVF are listed below:

```
      PROGRAM DELTIT(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DELAY,DELAYP / 3., 3. /
      DATA R / 3 /
      DATA E    .1 /
      DATA DT,IDTU / .25, 1 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA ITRG / 0.
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        DELAY = DELAY + E*DT
        CALL DELVF(RIN,ROUT,R,ITRG,DELAY,DELAYP,DT,IDTU,K)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      A = (3.+E*T)/3.
        B = (E+3.)/E
        C = E+3.
        AROUT = 3700.*C**3*(1.-A**(-B)-B*A**(-B)*ALOG(A)*(1.+B/2.*
     +          ALOG(A)))
        AITRG = A*(300./C*((1.-A**(-B))*(1.+3./C)-(B-1.)*A**(-B)*
     +          ALOG(A))+AROUT)
        PDOUT = 100.*(ROUT-AROUT)/AROUT
        IF(AROUT .EQ. 0.) PDOUT = 0.
        PDITRG = 100.*(ITRG-AITRG)/AITRG
        IF(AITRG .EQ. 0.) PDITRG = 0.
        WRITE(2,91) T,AROUT,ROUT,PDOUT,AITRG,ITRG,PDITRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1      ANALYTICAL  SIMULATED  PER CENT  ANALYTICAL  SIMUL*
     +        *ATED  PER CENT*/* TIME   OUTPUT     OUTPUT    DEVIATION *
     +        *  STORAGE    STORAGE   DEVIATION*/)
91    FORMAT(* *F4.1,F10.2,5F11.2)
      END
```

SUBROUTINE DELVF(RIN,ROUT,R,STRG,DEL,DELP,DT,IDTU,K)

TITLE - DISTRIBUTED DELAY WITH VARIABLE DELAY TIME

VERSION - 1A

DATE - 1 NOV 1974

PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS WITH
VARIABLE DELAY TIME

INPUT REQUIREMENTS

VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DEL, DELP,
DT, IDTU, K, R(I), RIN, STRG

VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN
-- DT, IDTU, K
VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN
-- DEL, RIN
VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES
-- DELP, R(I), STRG

OUTPUT VARIABLES

VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT,
STRG

VARIABLE DEFINITIONS

DEL - CURRENT LENGTH OF DELAY IN TIME UNITS.

DELP - PREVIOUS LENGTH OF DELAY IN TIME UNITS.

DT - SIMULATION TIME INCREMENT IN TIME UNITS.

IDT - NUMBER OF SUBDIVISIONS OF DT REQUIRED. UNITLESS.

IDTU - USER-SPECIFIED NUMBER OF SUBDIVISIONS OF DT. UNITLESS.

K - ORDER OF DELAY. UNITLESS.

R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

RIN - INPUT RATE IN UNITS/UNIT TIME.

ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

STRG - STORAGE IN UNITS.

```
      DIMENSION R(1)
      FK = FLOAT(K)
      B = 1. + (DEL-DELP)/(DT*FK)
      IDT = 1. + 2.*B*DT*FK/DELP
      IF(IDT .LT. IDTU) IDT = IDTU
      A = FK*DT/(DELP*FLOAT(IDT))
      DELP = DEL
      KM1 = K-1
      DO 20 J=1,IDT
        IF(K .EQ. 1) GO TO 15
        DO 10 I=1,KM1
          R(I) = R(I) + A*(R(I+1)-B*R(I))
10      CONTINUE
15      R(K) = R(K) + A*(RIN-B*R(K))
20    CONTINUE
      STRG = 0.
      DO 30 I=1,K
      STRG = STRG + R(I)*DEL/FK
30    CONTINUE
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II, Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch, "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria," IEEE Transactions on Systems, Man, and Cybernetics, SMC-2 (Sept., 1972) 472-486.

[3] Manetsch and Park, Op. cit., Chapter 10.

[4] Abkin, M. H. and C. Wolf, "Accuracy in the Simulation of Distributed Delay Processes in Models of Socioeconomic Systems," Project Working Paper 74-2. (East Lansing: Michigan State University, November 1974.)

CLASS Document

1. *Symbolic Name*	2. *Name*	3. *Type*	4. *Version No. and Date*
DELLVF	Distributed Delay with Storage Losses and Variable Time Delay	routine	1A 1/11/74

DELLVF Abstract

DELLVF simulates a distributed time delay process with storage losses (or additions) and variable delay times, computes the storage associated with flow through the delay, and automatically insures simulation stability. It is useful for modeling flow delays inherent in such aggregated processes as capital promotion, transportation and demographic processes. In these cases, storage losses could refer to disinvestment, spoilage and deaths or migrations, respectively; and delay times could vary in response to, for example, changing technologies and disease control, respectively.

The delay is modeled by a kth-order differential equation, which is also an aggregate representation of the case where individual lag times of entities in the aggregate are random variables following an Erlang distribution. The inputs to DELLVF are the unlagged value, or flow into the delay, the proportional storage loss rate and the current value of the mean lag time. Outputs are the lagged value, or flow out of the delay, and the storage resulting from integration of net flow into the delay. Finally, DELLVF insures simulation stability by automatically decreasing the integration step size when the time constant is small relative to the order of the delay.

CLASS Document

1. Symbolic Name	2. Name	3. Type	4. Version No. and Date
DELLVF	Distributed Delay with Storage Losses and Variable Time Delay	routine	1A 1/11/74

DELLVF Documentation

Contents

I. Process Description 1

II. Technical Description 3

A. Mathematical model 3

B. Sample run 7

III. Program Information 11

A. Program description 11

B. Program implementation 11

C. Program lists 12

1. Of sample run executive 13

2. Of component 14

Bibliographic References 16

I. Process Description

There are many situations in socioeconomic processes where time delays are distributed rather than discrete. That is, for aggregate flows individual entities in the aggregate have different lag times so that, while entities may enter the delay process at the same time, the output flow will be distributed over time. This will be particularly true when the individual lag times of entities are random variables [1]. For example, a tree population may be modeled as a series of age cohorts representing stages of productivity. Modeling the gestation stage, for instance, as a distributed delay says that for trees planted at the same time production will begin at different times for different trees--some sooner, some later--depending on genetic and local environmental conditions obtaining for each tree; and similarly for passage through the other productivity stages defined [2]. Similar examples abound in other processes, such as investment, transportation, diffusion of information, etc. DELLVF simulates this process.

It is frequently the case that storage losses or additions occur in the course of the delay process and that the length of the delay varies over time as a function of other variables in the system. Notable examples of situations in which storage losses or additions may occur include population migrations, tree removals, deaths, production losses, distribution losses, etc. Variable delay times may occur, for example, in production processes as functions of inventory, order backlogs, overtime wage rates and/or machine reliability. Similarly, delays associated with the diffusion of innovations may be functions of the profitability of the innovations

and/or of the communication infrastructure. DELLVF is specifically designed to incorporate storage losses and variable delay times in the delay process.

In the technical description which follows, reference is made to the use of DELLV to model the gestation stage of cocoa trees, where the inputs are tree plantings, failure rates and gestation time, and the outputs are trees becoming productive and the number of trees in the gestation stage.

II. Technical Description

A. Mathematical model

The kth-order differential equation and variable delay time can be shown to be equivalent to k first-order differential equations modeling a k-stage delay process, where each stage is a first-order delay [3] (Figure 1). The delay times for individual trees entering the gestation delay (assuming no losses and constant delay time) follow a kth-order gamma distribution with mean D. k determines the variance σ^2 of the gamma distributed delay times as $\sigma^2 = D^2/k$. In certain cases k and D may be estimated from real world or experimental data. As k gets large and goes to infinity (Figure 2), the distribution degenerates to a normal distribution with mean D and zero variance--i.e., a discrete delay of length D.

For the ith stage of the delay--where i=1, 2, ..., k--Q_i is the storage (number of trees) and [3]

$$(1) \quad Q_i(t) = \frac{D(t)}{k} R_i(t)$$

where

R_i = rate out of the ith stage (trees/year)

D = the mean delay time (years)

k = the order of the delay process.

The rate of change of Q_i is the net flow into the ith stage:

$$(2) \quad \frac{dQ_i(t)}{dt} = R_{i+1}(t) - R_i(t) - L_i(t)$$

where R_{k+1} = RIN, the planting rate into the delay (Figure 1), and L_i is the storage loss rate from the ith stage. If $L_i(t) = PLR(t) \cdot Q_i(t)$ where PLR is the proportional loss rate for the total storage in the whole delay (i.e., the losses are apportioned proportionately to each stage), then

$$(3) \quad \frac{dQ_i(t)}{dt} = R_{i+1}(t) - R_i(t) - PLR(t) \cdot Q_i(t)$$

$$= R_{i+1}(t) - R_i(t) \cdot (1 + \frac{D(t)}{k} \cdot PLR(t)).$$

But from (1)

$$(4) \quad \frac{dQ_i(t)}{dt} = \frac{1}{k}(D(t) \cdot \frac{dR_i(t)}{dt} + R_i(t) \cdot \frac{dD(t)}{dt} .$$

Thus, upon substituting and rearranging terms, the first-order differential equation modeling the ith stage of a kth-order delay with storage losses and variable delay time becomes

$$(5) \quad \frac{D(t)}{k} \cdot \frac{dR_i(t)}{dt} + (1 + \frac{1}{k} \cdot \frac{dD(t)}{dt} + \frac{D(t)}{k} \cdot PLR(t)) \cdot R_i(t) = R_{i+1}(t)$$

which can be solved numerically using Euler's integration approximation and taking $\frac{dD(t)}{dt} \approx \frac{D(t+DT) - D(t)}{DT}$ by

$$(6) \quad R_i(t) = R_i(t-DT) + DT \cdot \frac{k}{D(t-DT)} \cdot [R_{i+1}(t-DT) - R_i(t-DT) \cdot (1 + DD(t-DT)$$

$$+ \frac{D(t-DT)}{k} \cdot PLR(t-DT))]$$

where $DD(t-DT) = \frac{1}{k} \cdot \frac{D(t) - D(t-DT)}{DT}$ and where DT is the integration step size, the simulation time increment. DELLVF does not use (6), however. In order to insure stability, DELLVF divides the DT time interval into IDT

subintervals, where IDT = IDT(t) is computed in (15). Thus, integrating over the subintervals of length EDT = DT/IDT, (6) becomes

$$(7)\quad R_i(\tau) = R_i(\tau-EDT) + EDT\cdot\frac{k}{D(t-DT)}[R_{i+1}(\tau-EDT) - R_i(\tau-EDT)\cdot(1+DD(t-DT) + \frac{D(t-DT)}{k}\cdot PLR(t-DT))]$$

for $\tau = t - DT + EDT, \ldots, t - EDT, t$, and where, for the kth stage of the delay, $R_{k+1}(\tau-EDT) = RIN(t-DT)$ for all τ.

The output of the delay (trees becoming productive--Figure 1) is

$$(8)\quad ROUT(t) = R_1(t).$$

Finally, the storage in the delay associated with the flow through the delay (the number of trees in the gestation stage) is computed [3] as

$$(9)\quad S(t) = \frac{D(t)}{k}\sum_{i=1}^{k} R_i(t).$$

If the user can assume steady-state initial conditions--i.e., that the planting rate RIN has been constant for a long time prior to time zero (implying that the first k-1 derivatives of ROUT(t) are zero at t=0)--then the intermediate rates R_i, i=1, ..., k, should be initialized to:

$$(10)\quad R_i(0) = S(0)/D, \qquad i=1, \ldots, k.$$

where S(0) is the initial storage (i.e., number of trees) in the gestation delay. If, instead of the initial storage, the initial output or input to the delay is known, R_i should be initialized to:

$$(11)\quad R_i(0) = ROUT(0) = RIN(0), \qquad i=1, \ldots, k.$$

If, on the other hand, steady-state initial conditions are not to be assumed, equation (2) may be used to establish other initial conditions for R_i that yield non-zero initial values for the derivatives of ROUT. In any case, the initial storage and the rates R_i must satisfy the relationship:

$$(12) \quad S(0) = \frac{D}{k}\sum_{i=1}^{k} R_i(0).$$

If a model contains several delays with losses and variable delay times, for each delay j stability considerations (assuming PLR(t) = PLR, a constant) dictate that:

$$(13) \quad \frac{DT}{IDT_j(t)} < \frac{2D_j(t-DT)}{k_j(1+DD_j(t-DT)) + PLR_j(t-DT)\cdot D_j(t-DT)} .$$

For reasonable simulation accuracy and to include a margin of safety, the requirement becomes:

$$(14) \quad \frac{DT}{IDT_j(t)} \leq \frac{1}{2}\frac{D_j(t-DT)}{k_j(1+DD_j(t-DT)) + PLR_j(t-DT)\cdot D_j(t-DT)}$$

where $DD(t) \approx \frac{1}{k}\frac{dD(t)}{dt}$ (see (6)). With DT, D, DD, PLR, and k already known, DELLVF computes IDT_j, so that (14) holds:

$$(15) \quad IDT_j(t) = MAX\{IDTU_j, [1 + \frac{2\cdot DT}{D_j(t-DT)}(k_j(1 + DD_j(t-DT)) + PLR_j(t-DT)\cdot D_j(t-DT))]\}$$

where the operation [•] sets IDT_j equal to the largest integer in the value of the expression in brackets and MAX{a, b} sets IDT_j equal to the larger of a and b. The user may specify an IDT greater than 1 to increase

integration accuracy (even if not required for stability) by specifying IDTU greater than 1, i.e., the desired number of subintervals.

While DELLVF accurately simulates a distributed delay (e.g., the gestation stage of a cocoa tree population in this case), non-trivial inaccuracies relating to non-conservation of flow arise when $IDT_j > 1$ and when the output $ROUT_j(t)$ (equation (4)) is used elsewhere in a larger model; e.g., integrated or used as the input rate to another delay. In this example, the problem arises when the rate of trees leaving the gestation stage is used as the input to a second delay representing trees with increasing yields up to full maturity. For a full discussion of the problem and alternative ways to modify DELLVF to handle it, see [4].

DELLVF is one of a set of six distributed delay routines (Table 1) which cover situations where storage is computed automatically within the routine, the delay times, D_j, may vary over time, storage losses and/or accretions may occur in the course of the delay, and/or finer time cycles are required for stability assurances. The last three columns of Table 1 show the compile and execution central processor times and core storage required by DELLVF (and the other distributed delay routines for comparison) on Michigan State University's CDC 6500 computer with its FORTRAN Extended (Version 3.0) compiler and operating under HUSTLER 2, MSU's version of CDC's SCOPE 3.2 operating system.

B. Sample run

The sample run assumes a third-order, three-year gestation delay (i.e., k=3 and D=3) with a constant failure rate PLR(t) = 0.1/year and

Table 1

CLASS Distributed Delay Routines

Routines	Distributed delay process	With storage computed automatically	With stability and non-negative flow assurance	With storage losses and accretions	With variable delay times	Compile time (central processor)--seconds	Execution time (central processor)--milliseconds	Core storage--words (decimal)
DEL	X					.24	.27	56
DELS	X	X				.28	.31	62
DELF	X	X	X			.40	.44	93
DELLF	X	X	X	X		.47	.49	103
DELVF	X	X	X		X	.51	.55	104
DELLVF	X	X	X	X	X	.54	.57	108

constant rate of change of the delay time $\frac{dD(t)}{dt} = 0.1$ years/year. No trees existed in the gestation stage prior to time zero (i.e., $S(0) = R_1(0) = R_2(0) = R_3(0) = 0$), and the planting rate RIN is assumed to jump from 0 to 100 trees/year at time zero and to remain constant thereafter.

Results are tabulated in Figure 3 for a 10-year simulation, with DT = .25 years.

TIME	OUTPUT RATE	STORAGE
.0	.00	.00
1.0	4.76	96.68
2.0	26.38	172.16
3.0	45.70	219.05
4.0	56.85	245.72
5.0	62.12	261.40
6.0	64.25	271.77
7.0	64.89	279.73
8.0	64.86	286.53
9.0	64.54	292.92
10.0	64.11	299.01

III. Program Information

A. Program description

DELLVF was programmed and tested in the following operating environment:

Machine: CDC 6500
Operating system: MSU HUSTLER 2 L239 LSD 32--an extension of CDC SCOPE 3.2
Compiler: CDC FTN V3.0-P357
Core requirements: 108 decimal words
CP compile time: .54 seconds
CP execution time (for sample run): .57 milliseconds
Tapes: none
Direct access files: none
FORTRAN library subprograms used: FLOAT

DELLVF is structured as a single subroutine, DELLVF, which simulates a continuous (distributed) delay process with storage losses and/or additions and with variable delay time. It also computes storage in the delay. No COMMON blocks are used.

B. Program implementation

All input is transmitted through the argument list. Input variables DT, IDTU and K require values which must remain constant during the simulation of a particular delay; DELP and R(I) require only initial values, as they are subsequently recomputed within DELLVF. R must be dimensioned to at least K in the calling program. All output is transmitted through the argument list. The output variables are ROUT and the updated values of R(I) and STRG (called S in section II above). See the list of the sample run executive program for an example of the call statement.

For DELLVF to operate properly, DT and DEL(t) must be > 0 for all t, and K and IDTU must be integers $\geq$ 1.

C. Program lists

The computer programs for the sample run and for DELLVF are listed below:

```
      PROGRAM DELTST(OUTPUT,TAPE2=OUTPUT)
      DIMENSION R(3)
      DATA DT,IDTU / .25, 1 /
      DATA PLR / .1 /
      DATA E / .1 /
      DATA DELAY,DELAYP / 3., 3. /
      DATA K / 3 /
      DATA RIN,R,ROUT / 100., 4*0. /
      DATA STRG / 0. /
      WRITE(2,90)
      T = -DT
      DO 30 I=1,99999
        T = T + DT
        IF(T .EQ. 0.) GO TO 15
        DELAY = DELAY + E*DT
        CALL DELLVF(RIN,ROUT,R,STRG,PLR,DELAY,DELAYP,DT,IDTU,K)
        IF(AMOD(T,1.) .NE. 0.) GO TO 30
15      WRITE(2,91) T,ROUT,STRG
        IF(T .GE. 10.) STOP
30    CONTINUE
90    FORMAT(*1TIME      OUTPUT RATE        STORAGE*)
91    FORMAT(* *F4.1,F14.2,F16.2)
      END
```

```
SUBROUTINE DELLVF(RIN,ROUT,R,STRG,PLR,DEL,DELP,DT,IDTU,K)
```

TITLE - DISTRIBUTED DELAY WITH VARIABLE DELAY TIME AND STORAGE LOSSES

VERSION - 1A

DATE - 1 NOV 1974

PURPOSE - SIMULATION OF A DISTRIBUTED TIME DELAY PROCESS WITH VARIABLE DELAY TIME AND STORAGE LOSSES

INPUT REQUIREMENTS

VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- DEL, DELP, DT, IDTU, K, PLR, R(I), RIN, STRG

VARIABLES REQUIRING VALUES WHICH WILL NOT CHANGE DURING A RUN -- DT, IDTU, K
VARIABLES REQUIRING VALUES WHICH MAY CHANGE DURING A RUN -- DEL, PLR, RIN
VARIABLES COMPUTED BY THIS ROUTINE REQUIRING INITIAL VALUES -- DELP, R(I), STRG

OUTPUT VARIABLES

VALUES TRANSMITTED THROUGH THE ARGUMENT LIST -- R(I), ROUT, STRG

VARIABLE DEFINITIONS

DEL - CURRENT LENGTH OF DELAY IN TIME UNITS.

DELP - PREVIOUS LENGTH OF DELAY IN TIME UNITS.

DT - SIMULATION TIME INCREMENT IN TIME UNITS.

IDT - NUMBER OF SUBDIVISIONS OF DT REQUIRED. UNITLESS.

IDTU - USER-SPECIFIED NUMBER OF SUBDIVISIONS OF DT. UNITLESS.

K - ORDER OF DELAY. UNITLESS.

PLR - PROPORTIONATE LOSS RATE IN PROPORTION/UNIT TIME.

R(I) - INTERMEDIATE RATES IN UNITS/UNIT TIME.(STAGE OF DELAY)

RIN - INPUT RATE IN UNITS/UNIT TIME.

ROUT - OUTPUT RATE IN UNITS/UNIT TIME.

STRG - STORAGE IN UNITS.

```
      DIMENSION R(1)
      FK = FLOAT(K)
      B = 1. + (DEL-DELP)/(FK*DT) + PLR*DELP/FK
      IDT = 1. + 2.*B*DT*FK/DELP
      IF(IDT .LT. IDTU) IDT = IDTU
      A = FK*DT/(DELP*FLOAT(IDT))
      DELP = DEL
      KM1 = K-1
      DO 20 J=1,IDT
        IF(K .EQ. 1) GO TO 15
        DO 10 I=1,KM1
          R(I) = R(I) + A*(R(I+1)-B*R(I))
10      CONTINUE
15      R(K) = R(K) + A*(RIN-B*R(K))
20    CONTINUE
      STRG = 0.
      DO 30 I=1,K
        STRG = STRG + R(I)*DEL/FK
30    CONTINUE
      ROUT = R(1)
      RETURN
      END
```

References

[1] Manetsch, Thomas J. and Gerald Park, Systems Analysis and Simulation with Applications to Economic and Social Systems, Part II, Preliminary Edition (East Lansing: Division of Engineering Research, Michigan State University, 1973), Chapter 12.

[2] Abkin, Michael H. and Thomas J. Manetsch, "A Development Planning-Oriented Simulation Model of the Agricultural Economy of Southern Nigeria," IEEE Transactions on Systems, Man, and Cybernetics, SMC-2 (Sept., 1972) 472-486.

[3] Manetsch and Park, Op. cit., Chapter 10.

[4] Abkin, M. H. and C. Wolf, "Accuracy in the Simulation of Distributed Delay Processes in Models of Socioeconomic Systems," Project Working Paper 74-2. (East Lansing: Michigan State University, November 1974.)

CLASS Documents

Number	Title
1	"Computer Library for Agricultural Systems Simulation: A Progress Report," Michael H. Abkin and Tom W. Carroll. (July 1976)
2	"Software Standards Manual." (July 1976)
3	"Policy Analysis Language, Version 2.3, Programmer's Guide for CDC Cyber Computers," Claudia S. Winer and Chris Wolf. (July 1976)
4	"Policy Analysis Language, Version 2.3, Reference Manual," Claudia S. Winer anc Chris Wolf. (July 1976).
5	"Policy Analysis Language, Version 2.3, Guide to Using a PAL Program," Claudia S. Winer and Chris Wolf. (July 1976)
6	"DEMOGC: Demography with Distributed Age Cohorts," Michael H. Abkin and Chris Wolf. (July 1976)
7	"DEMOGD: Demography with Discrete Age Cohorts," Michael H. Abkin and Chris Wolf. (July 1976)
8	"Distributed Delay Routines: DEL, DELS, DELF, DELLF, DELVF, DELLVF," Michael H. Abkin and Chris Wolf. (July 1976)
9	"Table Functions: TABEL, TABEX, TABUL, TABUX," Michael H. Abkin, Chris Wolf, and Tom W. Carroll. (July 1976)
10	"AGACC: Accounting Routine for the Agricultural Sector," Dennis Pervis and Chris Wolf. (July 1976)
11	"User's Guide for the Beef Cattle Enterprise Simulation Model," Michael R. Jaske. (July 1976)
12	"User's Guide tc SYSOPT: An Interactive System Optimization Computer Program," Marcus Buchner. (August 1976)