AGENCY FOR INTERNATIONAL DEVELOPMENT
WASHINGTON, D. C. 20523

BIBLIOGRAPHIC INPUT SHEET

FOR AID USE ONLY
Batch 31

1. SUBJECT CLASSIFICATION	A. PRIMARY: Agriculture	AE10-0000-0000
	B. SECONDARY: Agricultural economics	

2. TITLE AND SUBTITLE
Accuracy in the simulation of distributed delay processes in models of socio-economic systems

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

4. DOCUMENT DATE	5. NUMBER OF PAGES	6. ARC NUMBER
1974	26p.	ARC

7. REFERENCE ORGANIZATION NAME AND ADDRESS
Mich.State

8. SUPPLEMENTARY NOTES (Sponsoring Organization, Publishers, Availability)
(In Project working paper 74-2)

9. ABSTRACT

10. CONTROL NUMBER PN-AAB-789	11. PRICE OF DOCUMENT
12. DESCRIPTORS Delay circuits Estimating Models Simulation Socioeconomic status	13. PROJECT NUMBER 14. CONTRACT NUMBER CSD-2975 Res. 15. TYPE OF DOCUMENT

AID 590-1 (4-74)

PN-AAB-787
CSD-2975 Rec.

AGRICULTURAL SECTOR ANALYSIS

AND SIMULATION PROJECTS

Contract AID/csd-2975

Accuracy in the Simulation of Distributed Delay Processes in Models of Socioeconomic Systems

Project Working Paper 74-2

Prepared By:

M. H. Abkin
C. Wolf

November 1974
Michigan State University

ACCURACY IN THE SIMULATION OF DISTRIBUTED DELAY PROCESSES IN MODELS OF SOCIOECONOMIC SYSTEMS

M. H. Abkin
C. Wolf

Introduction

Distributed time delays occur frequently in many contexts in socioeconomic systems, and models of such systems have reflected this. Examples include production lags in industrial processes [1], age cohorts in tree and cattle populations [2,3], transportation lags, stimulus response lags, and pollution generation response lags [4]. In many cases, distributed delay models can be used as aggregate representations of micro-processes wherein lag times of individual entities are random variables [5]. For example, in a mail order house the time for an individual order to be processed and shipped may be a random variable. In the aggregate, however, this stochastic discrete-event process can be modeled as a deterministic continuous delay process looking at the flow of orders through the mail order house, where the storage in the delay represents unfilled orders.

One mathematical model of distributed time delays which has proven very useful in socioeconomic systems models is the kth-order differential equation:

$$(1)\quad a_k \frac{d^k y(t)}{dt^k} + a_{k-1} \frac{d^{k-1} y(t)}{dt^{k-1}} + \ldots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = x(t),$$

where:

$$(2)\quad a_i = \frac{k!}{i!(k-i)!} \left(\frac{\tau}{k}\right)^i,$$

i.e., the $(i+1)^{th}$ term in the binomial expansion of $(1+\frac{\tau}{k})^k$; and where $x(t)$ is the input to the delay, $y(t)$ is the output (i.e., lagged value of the input), k is the order of the delay and τ is the time constant of the first-order differential equation resulting when k=1. It can readily be shown [6] that (1) is equivalent to k first-order delays in series (Figure 1), each modeled by:

$$(3) \quad \frac{\tau}{k} \frac{dr_i(t)}{dt} + r_i(t) = r_{i-1}(t), \qquad i=1, \ldots, k$$

where $r_0(t) = x(t)$ and $r_k(t) = y(t)$. Figure 2 shows the zero-state unit impulse response of (1) for different values of k.

When using distributed delays to model aggregated random micro-processes, as discussed above, it has been shown [5,6] that (1) is the correct model to use where random individual lag times follow an Erlang distribution. In this case, τ is the mean of the distribution and τ^2/k is its variance; as k gets large, the distribution approaches a normal distribution with mean τ and zero variance (i.e., a discrete delay), as indicated in Figure 2.

Where the delay is a material flow delay rather than an information delay, it is usually desired to compute storages in the delay (e.g., the number of unfilled orders in the above example). This may be done either by integrating net inflow:

$$(4) \quad \frac{ds(t)}{dt} = x(t) - y(t)$$

or equivalently [6] by

$$(5) \quad s(t) = \frac{\tau}{k} \sum_{i=1}^{k} r_i(t).$$

In simulating such delays, Euler integration is commonly used [1,7] to solve (3), i.e., k first-order difference equations of the form:

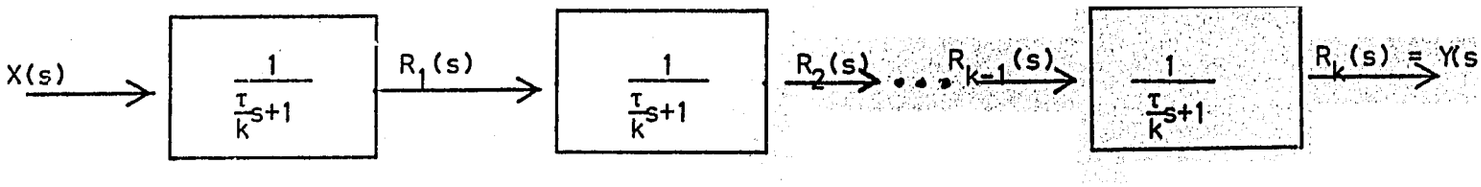

Figure 1

k^{th}-Order Distributed Delay as
k First-Order Delays in Series

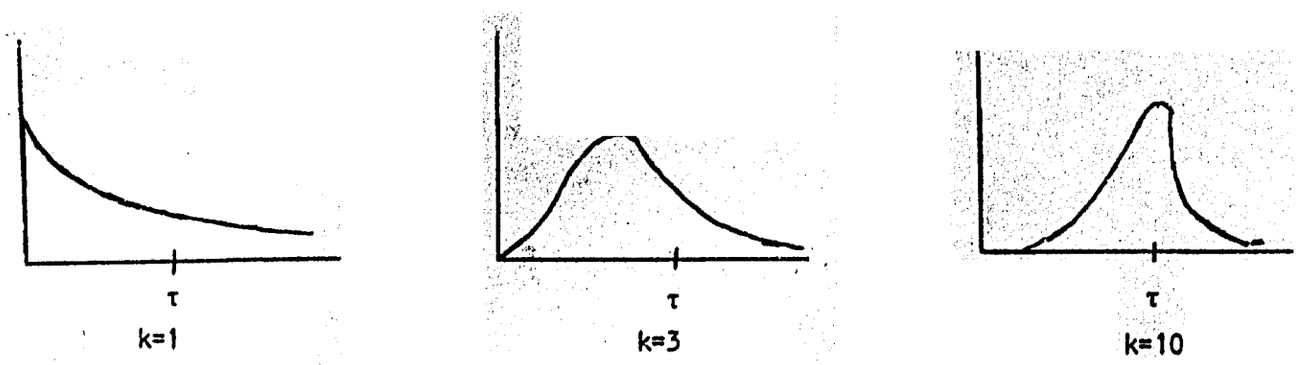

Figure 2

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

(6) $r_i(t) = r_i(t-h) + h\frac{k}{\tau}(r_{i-1}(t-h) - r_i(t-h)), \qquad i=1,2,\ldots,k,$

where h is the integration step size. Likewise, storage may be simulated recursively by:

(7) $s(t) = s(t-h) + h(x(t-h) - y(t-h))$

or by (5) directly.

When simulating with (6) (or any other numerical integration technique) stability considerations place an upper bound on h. Accuracy problems arising from this constraint and alternative ways to overcome them are discussed in the following sections.

Stability Criterion and Problems Posed for Simulation

As shown in [6], looking at the homogeneous part of (6),

(8) $r_i(t) = (1 - h\frac{k}{\tau})\, r_i(t-h),$

and solving recursively, we find that

(9) $r_i(t) = r_i(nh) = (1 - h\frac{k}{\tau})^n\, r_i(0)$

for a given initial condition $r_i(0)$. Solution stability (i.e., for the solution to be bounded as n increases) requires, therefore, that

(10) $|1-h\frac{k}{\tau}| \leq 1$

which implies that:

(11) $0 \leq h \leq 2\frac{\tau}{k}.$

Figure 3 illustrates the numerical results for various values of n as compared to the analytical solution. Obviously, a reasonable simulation requires $h < \frac{\tau}{k}$ (we must of course have $h > 0$); Forrester [1] recommends $h \leq \frac{\tau}{2k}$ as a rule of thumb.

In a large model containing several such delays, the requirement becomes:

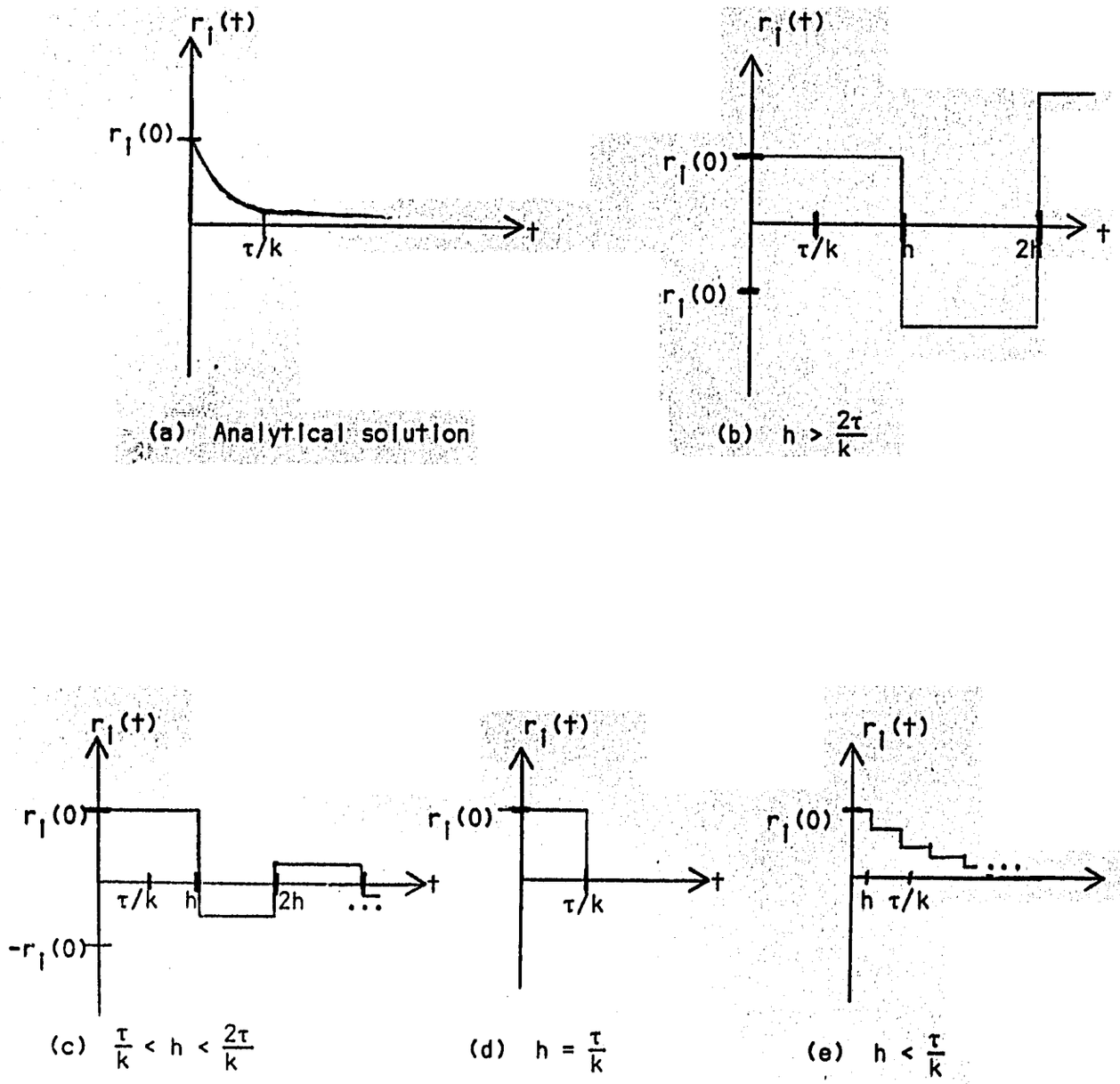

Figure 3

Numerical Solutions Depending
on Step Size h

(12) $h \leq \min_j[\tau_j/2k_j]$

where j indexes the delays in the model. This can often lead to extremely small values for h and thus, for large complex models, expensive simulations. Llewellyn [7] suggests that a value for h be chosen for the model as a whole consistent with acceptable integration error and with computational efficiency and that, for those delays j for which $\tau_j/2k_j < h$, h be divided into subintervals of length $h'_j = h/\alpha_j$, where:

(13) $\alpha_j = [1 + h\frac{2k_j}{\tau_j}]$,

so that $h'_j < \tau_j/2k_j$ and α_j is integer. Notice that when $h < \tau_j/2k_j$, α_j will be one and $h'_j = h$.[1/]

Given α, Llewellyn's subroutine, DELDT (Figure 4),[2/] divides the time interval $(t-h, t]$ into α subintervals of length $h' = h/\alpha$ and solves (6) recursively for each subinterval (Figure 5):

(14) $r'_i(t_j) = r'_i(t_j-h') + h'\frac{k}{\tau}(r'_{i-1}(t_j-h') - r'_i(t'_j-h'))$, $\quad i=1,2,\ldots,h$,

for $j=1,2,\ldots,\alpha$, where $t_j = t-h+jh'$, $r'_i = r_i/\alpha$ and $r'_0 = x/\alpha$. (For illustrative purposes, Figure 5 assumes $\dot{r}_k(v) > 0$ for $v\varepsilon(t-h,t]$.) The input x and intermediate rates r_i are reduced by a factor of α (the r_i are so reduced by virtue of $x(t)$ and $r_i(0)$ being so reduced), and the output is computed as:

[1/] In what follows, we will assume a particular delay j and so will drop the subscript j for convenience.

[2/] In DELDT, RINR = x, ROUTR = y, CROUTR(I) = $r'_i = r_i/\alpha$, DEL = τ, IDT = α, DT = h and K = k.

```
      SUBROUTINE DELDT(RINR,ROUTR,CROUTR,DEL,IDT,DT,K)
      DIMENSION CROUTR(1)
      DEL1 = DEL*FLOAT(IDT)/(FLOAT(K)*DT)
      ROUTR = 0.0
      DO 2 J=1,IDT
      RIN = RINR/FLOAT(IDT)
      DO 1 I=1,K
      ABC = CROUTR(I)
      CROUTR(I) = ABC+(RIN-ABC)/DEL1
    1 RIN = ABC
    2 ROUTR = ROUTR+CROUTR(K)
      RETURN
      END
```

Figure 4

Subroutine DELDT

$$(15)\quad y(t) = \sum_{j=1}^{\alpha} r'_k(t_j) = \frac{1}{\alpha}\sum_{j=1}^{\alpha} r_k(t_j$$

which is obviously (Figure 5) the <u>average</u> output over the interval $(t-h+h', t+h']$ rather than the <u>instantaneous</u> output at time t.

Subroutine DELF (Figure 6),[3/] on the other hand, computes the instantaneous output:

$$(16)\quad y(t) = r_k(t_\alpha) = r_k(t),$$

and (14) is replaced by:

$$(17)\quad r_i(t_j) = r_i(t_j-h') + h'\frac{k}{\tau}(r_{i-1}(t_j-h') - r_i(t_i-h')), \qquad i=1,\ldots,k,$$

for $j=1,2,\ldots,\alpha$.

The difference is readily apparent in Table 1 where the results of DELDT and DELF are compared with the analytical solution for y(t) for the test input $x(t) = 100u(t)$ and with $\tau=3$, $k=3$ and $h=.25$. The analytical solution in this case is:

$$(18)\quad y^*(t) = 100[1 - e^{-t}(.5t^2+t+1)].$$

Although $h < \tau/2k$ and thus $\alpha = 1$ by (13), α was arbitrarily set to 4 for this illustration.[4/] Notice that when $\alpha = 1$ and $h = .0625$, DELDT and DELF give the same results; and those of DELF are the same as for $\alpha = 4$ and $h = .25$, as they should be.

When modeling delays in physical flows, however, storage in the delay --computed by (4) or (5)--may be of more concern than the output rate, in which case DELDT and, to a lesser extent, DELF both have problems. An

---

[3/] In DELF, RIN = x, ROUT = y, R(I) = r_{k-i+1}, DEL = τ, IDT = α, DT = h and K = k.

[4/] Setting $\alpha > 0$ even though $h < \tau/2k$ may be desirable in certain cases where greater integration accuracy is required.

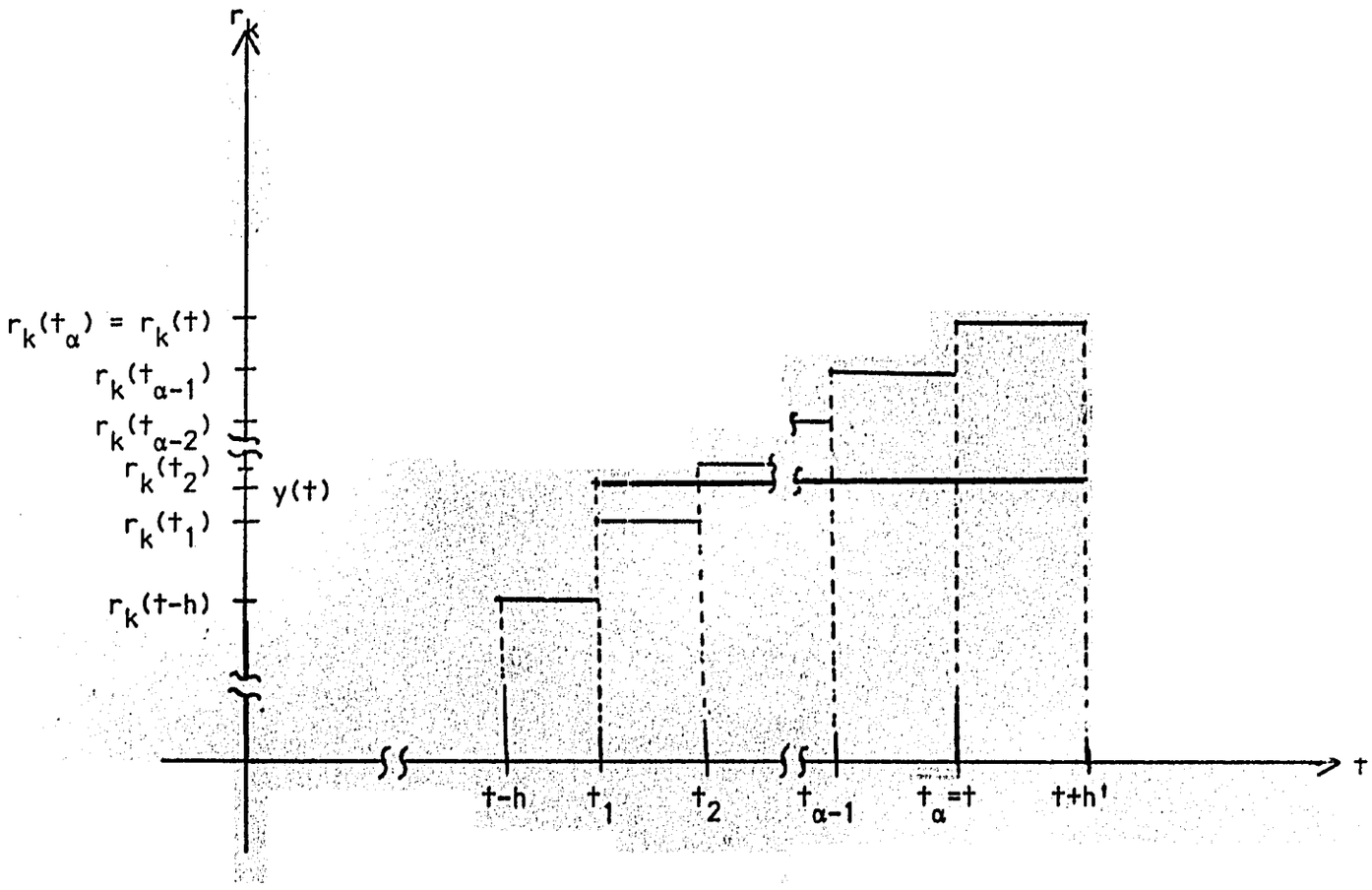

Figure 5

Subintervals Created by DELDT and Output $y(t)$

When $\dot{r}_k(v) > 0$, $v \varepsilon (t-h,t]$

```
      SUBROUTINE DELF(RIN,ROUT,R,DEL,IDT,DT,K)
      DIMENSION R(1)
      A = FLOAT(K)*DT/(DEL*FLOAT(IDT))
      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)-R(I))
   10 CONTINUE
   15 R(K) = R(K)+A*(RIN-R(K))
   20 CONTINUE
      ROUT=R(1)
      RETURN
      END
```

Figure 6

Subroutine DELF

Table 1

Comparison of DELDT and DELF Outputs

For Step Input $x(t) = 100u(t)$

Time t 1	Analytical Output $y^*(t)$ 2	Percent Deviation: $100(y(t) - y^*(t))/y^*(t)$			
		$\alpha = 4$, $h = .25$		$\alpha = 1$, $h = .0625$	
		DELDT 3	DELF 4	DELDT 5	DELF 6
0	0	0	0	0	0
1	8.03	-27.23	-7.59	-7.59	-7.59
2	32.33	-8.12	-.03	-.03	-.03
3	57.68	-2.56	1.26	1.26	1.26
4	76.19	-.64	1.23	1.23	1.23
5	87.53	-.00	.91	.91	.91
6	93.80	.15	.59	.59	.59
7	97.04	.15	.35	.35	.35
8	98.62	.10	.20	.20	.20
9	99.38	.06	.11	.11	.11
10	99.72	.03	.05	.05	.05
11	99.88	.02	.03	.03	.03
12	99.95	.01	.01	.01	.01
13	99.98	.00	.01	.01	.01
14	99.99	.00	.00	.00	.00
15	100.00	.00	.00	.00	.00

important feature of this distributed delay model, (1) or (3) and its simulation model (6), is that it conserves flow, i.e., storage is neither gained nor lost in the course of the delay and (4) is true. If the stability criterion (12) requires $\alpha > 1$, however, simulation by (14) with DELDT or (17) with DELF does not conserve flow; i.e., while (4) and (5) are true, the numerical solution (7) cannot be used (unless it is executed before the IF statement in Figure 6), its use leading to a not insignificant steady-state error in the case of the above test input (Table 2). Simulating with (5) directly, however, gives very good results (Table 2), results which are identical to the case where $\alpha = 1$ and $h = .0625$ for both (5) and (7). The analytical solution for storage in Table 2 is:

$$(19) \quad s^*(t) = 100[3-e^{-t}(.5t^2+2t+3)].$$

Although this storage computation problem can be solved either by using (5) or by using (7) with h replaced by h' (i.e., by placing it in the outer DO-loop and before the inner DO-loop in Figures 4 and 6), storage will not be conserved in a larger system--of which the delay in question may be only a small part--when the output y(t) is used to represent storage leaving the delay. For example, if the delay represents the processing and shipping of orders through a mail order house--where x(t) is the rate orders are received, y(t) is the rate orders are shipped and s(t) is unfilled orders--the number of orders shipped is the time integral of y(t). In another example, the delay may be one link of a chain of delays modeling production stages of cocoa trees [2]--e.g., the gestation stage, a stage of increasing yields, maturity, etc.--

Table 2

Comparison of DELDT and DELF Storages

For Step Input $x(t) = 100u(t)$

Time t 1	Analytical Storage $s^*(t)$ 2	Percent Deviation[a]: $100\ (s(t) - s^*(t))/s^*(t)$			
		By Euler Integration[b]		By Intermediate Rates[c]	
		DELDT 3	DELF 4	DELDT 5	DELF 6
0	0	0	0	0	0
1	97.67	1.59	1.19	.57	.57
2	178.20	4.00	2.59	.96	.96
3	232.79	5.39	3.22	.92	.92
4	265.20	6.02	3.40	.70	.70
5	282.82	6.25	3.38	.47	.47
6	291.82	6.31	3.31	.28	.28
7	296.22	6.31	3.24	.16	.16
8	298.29	6.29	3.19	.09	.09
9	299.24	6.27	3.16	.04	.04
10	299.67	6.26	3.14	.02	.02
11	299.86	6.26	3.13	.01	.01
12	299.94	6.25	3.13	.01	.01
13	299.97	6.25	3.13	.00	.00
14	299.99	6.25	3.13	.00	.00
15	300.00	6.25	3.13	.00	.00

[a] For $\alpha = 4$, $h = .25$

[b] $s(t) = s(t-h) + h(x(t-h) - y(t-h))$

[c] $s(t) = \frac{\tau}{k} \sum_{i=1}^{k} r_i(t)$

where the output $y_1(t)$ of the first delay is the input $x_2(t)$ to the second delay.

In both cases--either where $y(\cdot)$ is to be integrated or where it is the input to another delay (r_0 in (6))--the nature of Euler integration requires $y(\cdot)$ to be constant over the interval $[t-h, t)$. When $\alpha > 1$ and $h' \neq h$, $y(\cdot)$ is constant only over $[t-h, t-h+h')$. Table 3 shows the results when the delay output of the test case of Table 1 is the input to another delay for which $\tau=3$ and $k=2$. The analytical solutions for output and storage of the second delay in this case are:

$$y^*(t) = 100[1-2e^{-t}(t^2+14t+68) - 9e^{-\frac{2}{3}t}(2t-15)] \tag{20}$$

$$s^*(t) = 300[1-e^{-t}(.5t^2+10t+55) - 9e^{-\frac{2}{3}t}(t-6)]. \tag{21}$$

otice that for DELF in the second delay of the chain, $s(t)$ computed by 5) (columns 7 and 11) and $y(t)$ (columns 5 and 10) are <u>not</u> the same when =4 and h=.25 as when $\alpha=1$ and h=.0625, whereas they are the same for the first delay of the chain (Table 1, columns 4 and 6). For this test input, nevertheless, DELF results are an improvement over DELDT results.

The next section discusses alternative ways to address this problem

Alternative Solutions

Several approaches to correcting for the non-conservation of flow when $\alpha > 1$ were tested, and two of the more promising ones and their shortcomings are reported in this section.

The first approach is based on the recognition that for Euler integration the output rate $y(t)$ is not just the instantaneous output at time t but also is considered the constant output over the interval $[t, t+h)$,

Table 3

Comparison of DELDT and DELF Used as the

Second Delay in a Chain

Time t 1	Anal. Outp. y*(t) 2	Anal. Strg. s*(t) 3	Percent Deviation of Simulated From Analytical							
			α = 4, h = .25				α = 1, h = .0625			
			Output		Storage[a/]		Storage[b/]		Output	Storage[c/]
			DELDT 4	DELF 5	DELDT 6	DELF 7	DELDT 8	DELF 9	(DELDT and DELF) 10	11
0	0	0	0	0	0	0	0	0	0	0
1	.18	2.30	-85.69	-60.08	-66.36	-49.52	-66.20	-49.31	-38.38	-23.44
2	2.91	20.60	-51.73	-27.68	-31.38	-19.95	-30.20	-19.10	-13.15	-6.83
3	11.28	59.43	-30.63	-13.83	-16.45	-9.34	-14.08	-7.90	-4.73	-1.83
4	24.91	109.26	-18.42	-7.07	-9.02	-4.51	-5.61	-2.60	-1.23	.07
5	40.88	158.81	-11.13	-3.52	-4.99	-2.10	-.76	.18	.27	.77
6	56.25	201.10	-6.69	-1.63	-2.72	-.87	2.13	1.67	.85	.95
7	69.25	233.71	-3.96	-.62	-1.44	-.26	3.87	2.47	.98	.89
8	79.28	257.12	-2.30	-.13	-.72	.02	4.91	2.89	.91	.74
9	86.52	273.05	-1.29	.09	-.34	.13	5.52	3.09	.75	.57
10	91.48	283.46	-.71	.17	-.14	.15	5.86	3.17	.58	.42
11	94.74	290.05	-.37	.17	-.04	.13	6.06	3.19	.42	.29
12	96.82	294.11	-.18	.15	.00	.10	6.16	3.19	.30	.20
13	98.11	296.56	-.09	.11	.02	.07	6.21	3.18	.20	.13
14	98.89	298.02	-.04	.08	.02	.05	6.24	3.16	.13	.08
15	99.36	298.87	-.01	.05	.01	.03	6.25	3.15	.09	.05

[a/] $s(t) = \frac{\tau}{h} \sum_{i=1}^{h} r_i(t)$

[b/] $s(t) = s(t-h) + h(x(t-h) - y(t-h))$

[c/] Both methods gave identical results for s(t).

i.e., $y(t)$ is a step function. However, when $\alpha > 1$, $y(t)$ is constant only over the interval $[t, t+h')$, where $h' = h/\alpha$. To conserve flow, then, whether integrating the output or feeding it into another delay, we want not $y(t)$ but $\bar{y}(t)$, where $\bar{y}(t)$ is the <u>average</u> output over the interval $[t, t+h)$,

$$(22) \quad \bar{y}(t) = \frac{1}{\alpha} \sum_{j=0}^{\alpha-1} y(t_j),$$

where $t_j = t + jh'$ and $y(t_j) = r_k(t_j)$ in (17).

Unfortunately, at time t we cannot know the output for subintervals in $[t, t+h)$; we only know it for subintervals in the previous period, $[t-h, t)$; i.e., we only know $\bar{y}(t-h)$. Notice that (22) is very similar to (15), the output equation for DELDT. Where DELDT is in error is in interpreting the result as $y(t)$ rather than $\bar{y}(t-h+h')$, i.e., as the instantaneous output at t to be used as the average over $[t, t+h)$ rather than as the average over $[t-h+h', t+h')$. (This was seen above and in Figure 5.) Running the summation index from 0 to $\alpha-1$ instead of from 1 to α (and defining $t_j = t-h+jh'$) gives us $\bar{y}(t-h)$.

The problem with this approach is that, at least for the delay in question, the normal order of simulation time sequencing has to be reversed, the level equation (7) would have to <u>follow</u> execution of the delay rather than precede it, because $\bar{y}(t-h)$ (which would replace $y(t-h)$ in (7)) is determined in the delay at time t rather than at time $t-h$. Similarly, a chain of such delays would have to be executed in reverse order, i.e., a delay whose output feeds into the next would be executed before rather than after the next.

There would be very real difficulties in interfacing parts of a model having this reverse time sequencing with other parts having normal sequencing, not to mention the increased opportunity for error. Furthermore, $\bar{y}(t)$ is unknown at time t even though it may be needed. For example, in a chain of n delays representing n age cohorts in a population, the output of the last cohort $\bar{y}_n(t)$ may be considered deaths,

$$(23) \quad D(t) = \sum_{i=1}^{n} \mu_i s_i(t) + \bar{y}_n(t),$$

where D is individuals/year dying from the population, μ_i is the proportion dying per year from the ith cohort, and s_i is population in the ith cohort.

Nevertheless, a routine such as DELF1 (Figure 7) incorporating this procedure, when it can appropriately and safely be used, does maintain conservation of flow in, for example, a chain of delays when $\alpha > 1$; i.e., storage computed by (7) is equivalent to that computed by (5) with no steady-state error. Table 4 shows the results of simulating the earlier example with DELF1--that is, two delays in sequence, h=.25, α=4, $\tau_1=k_1=\tau_2=3$, $k_2=2$, $x_1(t) = 100u(t)$, $x_2(t) = \bar{y}_1(t)$ (instead of $y_1(t)$)--as compared to the analytical solutions of (18)-(21). Only one storage column is shown since results are identical for either method. For the first delay of the chain, results are identical to those of DELF for both output (Table 1, column 4) and storage computed by (5) (Table 2, column 6). Note also that, in DELF1 (Figure 7), ROUT = $\bar{y}(t-h) = y(t-h)$ for α=1, as should be the case. In this case (α=1), $\bar{y}(t) = y(t) = r_k(t) = R(1)$ is also known.

The second approach to maintaining the conservation of flow when $\alpha > 1$ takes as its point of departure the first approach, i.e., the computation at time t of $\bar{y}(t-h)$. Namely, an attempt is made to <u>project</u> what $\bar{y}(t)$ will be over the period [t, t+h) based on past values of $\bar{y}(\cdot)$.

```
      SUBROUTINE DELF1(RIN,ROUT,R,DEL,IDT,DT,K)
      DIMENSION R(1)
      A = FLOAT(K)*DT/(DEL*FLOAT(IDT))
      KM1 = K-1
      ROUT = R(1)
      DO 20 J = 1,IDT
      IF(K.EQ.1) GO TO 15
      DO 10 I = 1,K
      R(I) = R(I) + A*(R(I+1) - R(I))
   10 CONTINUE
   15 R(K) = R(K) + A*(RIN-R(K))
      ROUT = ROUT+R(1)
   20 CONTINUE
      ROUT = (ROUT-R(1))/FLOAT(IDT)
      RETURN
      END
```

Figure 7

Subroutine DELF1

Table 4

Results Using DELF1 to Simulate a Two-Delay Chain, Compared to the Analytical Solutions

Time t 1	Percent Deviation of Simulated From Analytical			
	First Delay		Second Delay	
	Output $y_1(t)$ 2	Storage $s_1(t)$ 3	Output $y_2(t)$ 4	Storage $s_2(t)$ 5
0	0	0	0	0
1	-7.59	.57	-32.15	-23.55
2	-.03	.96	-12.04	-6.95
3	1.26	.92	-4.46	-1.92
4	1.23	.70	-1.18	.00
5	.91	.47	.26	.72
6	.59	.28	.83	.91
7	.35	.16	.96	.86
8	.20	.09	.89	.72
9	.11	.04	.74	.56
10	.05	.02	.57	.41
11	.03	.01	.42	.29
12	.01	.01	.29	.20
13	.01	.00	.20	.13
14	.00	.00	.13	.08
15	.00	.00	.09	.05

This approach, then, assuming a good estimate of $\bar{y}(t)$, eliminates the need to reverse the computational sequence.

First- and second-order Lagrange interpolations were tested as estimates $\bar{y}_e(t)$ of $\bar{y}(t)$. (Other estimators tested, including the 2nd-order Taylor series expansion, led to steady-state errors in the computation of storage.) The estimators are:

(24) $\bar{y}_e(t) = 2\bar{y}(t-h) - \bar{y}(t-2h)$

for the first-order interpolation and

(25) $\bar{y}_e(t) = 3[\bar{y}(t-h) - \bar{y}(t-2h)] + \bar{y}(t-3h)$

for the second-order interpolation.

This approach is further refined by correcting $\bar{y}_e(t)$ for the error on the previous estimation, i.e., by adding $\bar{y}(t-h) - \bar{y}_e(t-h)$ to (24) and (25). This can be done since at time t $\bar{y}(t-h)$ is known. The estimators become:

(26) $\bar{y}_e(t) = 3\bar{y}(t-h) - \bar{y}(t-2h) - \bar{y}_e(t-h)$

and

(27) $\bar{y}_e(t) = 4\bar{y}(t-h) - 3\bar{y}(t-2h) + \bar{y}(t-3h) - \bar{y}_e(t-h)$.

Estimator (27), programmed in Figure 8, gave considerably better results than did (26) and so is shown in Table 5 compared to the analytical solutions for the above example. For the first delay in the chain, the instantaneous output $y_1(t)$ (the projected average output $\bar{y}_e(t)$ is not shown) and storage computed by (5) are identical to earlier results. For the second delay, DELF2 gives as good approximations to the

```
      SUBROUTINE DELF2(RIN,ROUT,ROUTAV,R,DEL,IDT,DT,K)
      DIMENSION R(1), ROUTAV(2)
      A = FLOAT(K)*DT/(DEL*FLOAT(IDT))
      KM1 = K-1
      ROAV = R(1)
      DO 20  J = 1,IDT
      IF(K.EQ.1) GO TO 15
      DO 10  I = 1,K
      R(I) = R(I) + A*(R(I+1) - R(I))
   10 CONTINUE
   15 R(K) = R(K) + A*(RIN - R(K))
      ROAV = ROAV + R(1)
   20 CONTINUE
      ROAV = (ROAV - R(1))/FLOAT(IDT)
      ROUT = 4.*ROAV - 3.*ROUTAV(1) + ROUTAV(2) - ROUT
      ROUTAV(2) = ROUTAV(1)
      ROUTAV(1) = ROAV
      IF(IDT.EQ.1)ROUT = R(1)
      RETURN
      END
```

Figure 8

Subroutine DELF2

Table 5

Results Using DELF2 to Simulate

a Two-Delay Chain, Compared

to the Analytical Solutions

Time t 1	Percent Deviation of Simulated From Analytical					
	First Delay			Second Delay		
	Output $y_1(t)$ 2	Storage $s_1(t)$ a 3	Storage $s_1(t)$ b 4	Output $y_2(t)$ 5	Storage $s_2(t)$ a 6	Storage $s_2(t)$ b 7
0	0	0	0	0	0	0
1	-7.59	.57	.75	-48.15	-30.60	-30.31
2	-.03	.96	.97	-12.97	-6.82	-6.63
3	1.26	.92	.89	-4.35	-1.71	-1.63
4	1.23	.70	.67	-1.07	.10	.13
5	.91	.47	.44	.32	.76	.76
6	.59	.28	.27	.85	.93	.91
7	.35	.16	.15	.96	.87	.85
8	.20	.09	.08	.89	.72	.71
9	.11	.04	.04	.73	.56	.55
10	.05	.02	.01	.57	.41	.40
11	.03	.01	.00	.41	.29	.28
12	.01	.01	-.00	.29	.20	.19
13	.01	.00	-.00	.20	.13	.13
14	.00	.00	-.01	.13	.09	.08
15	.00	.00	-.01	.08	.06	.05

a--$s(t) = \frac{\tau}{k} \sum_{l=1}^{k} r_l(t)$

b--$s(t) = s(t-h) + h(x(t-h) - \bar{y}_e(t-h))$

output and storage, for the most part, as does DELF1 (Table 4, columns 4 and5) and much better ones than does DELF (Table 3, columns 5 and 7).

As for maintaining conservation of flow, comparing columns 3 and 4 of Table 5 with columns 6 and 4 (respectively) of Table 2, and columns 6 and 7 of Table 5 with columns 7 and 8 of Table 3, DELF2 clearly removes the discrepancy (well within reasonable tolerances considering overall simulation accuracy) between the two methods of computing storage when $\alpha > 1$. In particular, there is no steady-state error when using Euler integration with DELF2 as there is with DELF and DELDT.

There are two major shortcomings with DELF2, however. First, the projected average output does not degenerate to the instantaneous output when $\alpha = 1$, as it does in DELF1. Hence the logic to do this is included at the end of the program in Figure 8.

Secondly and more seriously, although the estimator (27) works well for this example, in general it is sensitive to the input function $x(\cdot)$ and the parameters h, τ and k. The error in the 2nd-order Lagrange interpolation of (27) is:

$$(28) \quad E\bar{y}_e(t) = h^3 \frac{d^3\bar{y}(z)}{dz^3} \text{ evaluated at } z=t,$$

which, from (22) and (3), becomes for $k > 3$

$$(29) \quad E\bar{y}_e(t) = \frac{h^3}{\alpha} \sum_{j=0}^{\alpha-1} \frac{d^3y(z)}{dz^3} \text{ evaluated at } z=t$$

$$= \frac{1}{\alpha}(h\frac{k}{\tau})^3 \sum_{j=0}^{\alpha-1} [r_{k-3}(t_j) - 3r_{k-2}(t_j) + 3r_{k-1}(t_j) - r_k(t_j)],$$

where $t_j = t+jh'$ and $y(t_j) = r_k(t_j)$. The input $x(\cdot)$ and its derivatives appear directly in (29) when $k \leq 3$ and indirectly for any k in the relative values of $r_i(\cdot)$, $i=1,2,\ldots,k$, so that a rapidly changing input will increase

the error in $\bar{y}_e(t)$. In addition, short high-order delays will have greater error in $\bar{y}_e(t)$ because of the factor $(hk/\tau)^3$. The estimation is good in this example because k=3, τ=3 and x(t) = 100u(t) for the first delay in the chain, and k=2, τ=3 and x(t) is smooth (equation (18)) for the second delay.

Complications for Nonconservative and Variable Delays

If h is chosen to satisfy (12) for all delays j in a model, in spite of the inefficiencies created, none of the above problems will occur. However, when (3) is generalized to include cases of variable delay times $\tau(t)$ and storage losses, following (12) for all delays won't work because the stability criterion for h, i.e., its upper bound, becomes a time variable!

The generalization of (3) is:

$$(30)\quad \frac{dr_i(t)}{dt} = \frac{k}{\tau(t)}\left[r_{i-1}(t) - r_i(t)\left(1+\frac{\tau(t)}{k}\lambda(t) + \frac{1}{k}\frac{d\tau(t)}{dt}\right)\right]$$

for i=1,2,...,k and where $\lambda(t)$ is the proportional storage loss rate per unit time from the delay. In this case, the stability criterion for Euler integration is:

$$(31)\quad 0 < h < \min_j\left[\frac{1}{2}\cdot\frac{\tau_j(t)}{\dot{\tau}_j(t) + \lambda_j(t)\tau_j(t) + k_j}\right]$$

Note that (31) degenerates to (12) when $\dot{\tau}_j(t) = 0$ and $\lambda_j(t) = 0$. This implies that $h_j' = h_j'(t)$ and $\alpha_j = \alpha_j(t)$.

Summary and Conclusion

A kth-order distributed delay model was described, and its application to aggregations of random delays of discrete micro processes as well as to continuous flow delays was discussed. The stability criterion associated

with the use of numerical integration techniques and the limitation it places on the integration step size were presented along with two subroutines, DELDT [7] and DELF, which have been proposed to deal with that limitation. DELF is proposed in order to simplify and clear up inconsistencies and misinterpretations in DELDT. Both routines fail to conserve flow under certain conditions, however. This problem was outlined, and two approaches to overcoming it were presented. Both approaches have shortcomings, however, and the user is cautioned to use them only where appropriate and where the error is acceptable.

References

[1] Forrester, Jay W. Industrial Dynamics. Cambridge: MIT Press, 1961.

[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 (September 1972), 472-486.

[3] Posada, Alvaro. "A Simulation Analysis of Policies for the Northern Colombia Beef Cattle Industry." (unpublished Ph.D. dissertation Michigan State University, 1974.

[4] Meadows, Donnella H., et al. The Limits to Growth. New York: Universe Books, 1972.

[5] Manetsch, Thomas J. "Transfer Function Representation of the Aggregate Behavior of a Class of Economic Process." IEEE Transactions on Automatic Control. (October, 1966), pp. 694-698.

[6] Manetsch, Thomas J. and Gerald Park. System Analysis and Simulation with Applications to Economic and Social Systems. Vol. II. East Lansing: Department of Electrical Engineering and System Science, Michigan State University, August, 1974.

[7] Liwellyn, Robert. FORDYN--An Industrial Dynamics Simulator. Raleigh, North Carolina: North Carolina State University, 1965.