"the method discussed does not require computation of the state transition matrix"

On Computing the Input to a Discretized Linear System

by

Christian B. San Juan*

Abstract

In some applications, there is the need to compute for the input signals of a linear system knowing the form of the output signal. This is normally true in physiological and mechanical systems, especially in the field of robotics. A simple way to do this is by discretizing the state equation. The resulting equations are not only easy to manipulate but also amenable to computer solutions.

Keywords

State Equation, State Variable, Matrix

Level of Reader

Should know basic matrix operation and state variable formulation of linear dynamic systems.

Introduction

In simulation studies, the output is normally computed given the known configuration and parameters of the system as well as the inputs. However, there are problems where we must determine the input, given the model and its output. Examples of these problems are given below.

1.	Biomechanics	– the voluntary, pathological (spasm) or artificial (electrical excitation) input to a muscle can be calculated from known muscle parameters, joint angle and torque.
2.	Robotics	– from the known motor properties and the desired torque or force between two segments of an industrial manipulator of an electrical or hydraulic activator, the desired input (current or voltage) signal can be computed.
3.	Servomechanism	– knowing the desired position or speed of an elevator, the input signal (current or voltage) needed to control the elevator could be calculated.

A method that would compute for the inputs of a linear system given its model and output is discussed.

*Faculty of Electrical Engineering,College of Engineering,University of the Philippines

The Method

Consider the state equation and output equation given below. These two equations describe a linear continuous system.

$$\dot{x} = Ax + Bu \qquad (1)$$

$$y = Cx + Du \qquad (2)$$

where	A	=	n x n	state coefficient matrix
	B	=	n x m	driving matrix
	C	=	p x n	output coefficient matrix
	D	=	p x m	transfer matrix
	x	=	n x 1	state variable vector
	u	=	m x 1	input vector
	y	=	p x 1	output vector
	$\dot{x}$	=	$\frac{d}{dt}x$	

The solution of equation 1 is found to be

$$x = \phi(t)x(0) + \int_0^t \phi(t-\tau)Bu(\tau)d\tau \qquad (3)$$

with $\phi(t)$[1] known as the state transition matrix.

The output equation is then equal to

$$y = C\phi(t)x(0) + C\int_0^t (t-\tau)Bu(\tau)d\tau + Du \qquad (4)$$

Tadej Bajd[2] uses equation 4, but with D = 0, to compute for the input u(t), knowing the output and initial condition x(0). This method, although quite efficient, is hard to use since it requires the knowledge of the state transition matrix.[3] To circumvent this limitation the discretized form of equations 1 and 2 was used.

Using the definition of the derivative, we get

$$\dot{x}^4 = \frac{x(t = \Delta t) - x(t)}{\Delta t} \qquad (5)$$

---

[1] $\phi(t) = e^{AT}$

$= \mathcal{L}^{-1}\left[(SI - A)^{-1}\right]$ with I = identity matrix

[2] Tadej Bajd, Computing the Input to a Linear Model, "Simulation", Vol. 40, No. 6, June 1983, pp 241-243.

[3] The state transition matrix is quite difficult to evaluate as could be seen from the equation given previously. (Note 1)

[4] Doing this, we are said to be discretizing the state equation 1.

Let $\Delta t = T$ and $t = kT$ where $k = 0, 1, 2, \ldots$ substituting equation 5 to equation 1 will result to

$$x(kT + T) = (I + TA)\,x(kT) + TBu(kT) \tag{6}$$

and for equation 2,

$$y(kT) = Cx(kT) + Du(kT) \tag{7}$$

To simplify our notations, we let $K = kT$, $A = I + TA$ and $B = TB$, hence we have

$$x(K + 1) = Ax(K) + Bu(K) \tag{8}$$
$$y(K) = Cx(K) + Du(K) \tag{9}$$

The solution of equation 8, a 1st order difference equation is

$$x(K) = A^{K}x(0) + \sum_{i=0}^{K-1} A^{K-i-1}Bu(i) \tag{10}$$

Therefore, equation 9 becomes

$$y(K) = CA^{K}x(0) + \sum_{i=0}^{K-1} CA^{K-i-1}Bu(i) + Du(k) \tag{11}$$

To simplify further our discussion, we let $p = m$, i.e., the number of inputs is equal to the number of outputs, hence D is a square matrix.

Case I, D = 0

Expanding equation II gives us the following equations.

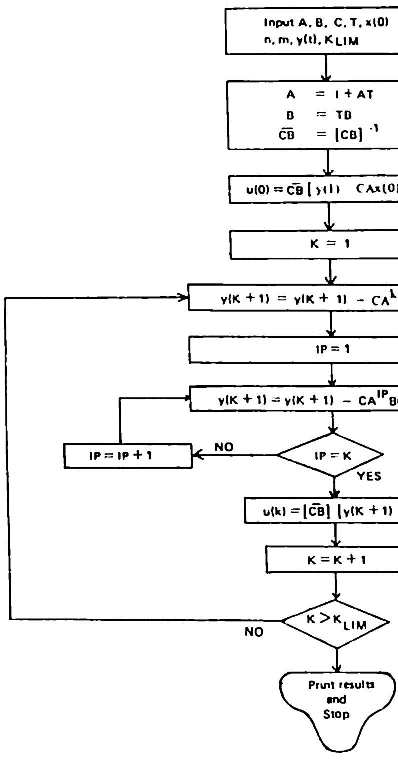
$$\begin{aligned} y(0) &= Cx(0) \\ y(1) &= CAx(0) + CBu(0) \\ y(2) &= CA^{2}x(0) + C[ABu(0) + Bu(1)] \\ &\bullet \\ &\bullet \\ &\bullet \\ y(K+1) &= CA^{K+1}x(0) + C[A^{K}Bu(0) \\ &\quad + A^{K-1}Bu(1) + \ldots + Bu(K)] \end{aligned} \tag{12}$$

From 12, we could solve for the input variable

$$u(0) = [CB]^{-1}\{y(1) - CAx(0)\}$$

$$
\begin{aligned}
u(1) &= [\,CB\,]^{-1}\ \{y(2) - CA^2x(0) - CABu(0)\,\} \\
u(2) &= [\,CB\,]^{-1}\ \{y(3) - CA^3x(0) - CA^2Bu(0) \\
&\qquad - CABu(0) \\
&\ \ \vdots \\
u(K) &= [\,CB\,]^{-1}\ \{y(K+1) - CA^{K+1}x(0) \\
&\qquad - CA^KBu(0) - CA^{K-1}Bu(1) \\
&\qquad - \ldots - CABu(K-1)\,\}
\end{aligned}
\qquad (13)
$$

Flow Chart For Case I



Input A, B, C, T, x(0) n, m, y(t), K LIM
A = I + AT
B = TB
C̄B = [CB]^-1
u(0) = C̄B [y(1) CAx(0)]
K = 1
y(K + 1) = y(K + 1) - CA^(K+1) x(0)
IP = 1
y(K + 1) = y(K + 1) - CA^IP Bu (K - IP)
IP = K
NO
YES
IP = IP + 1
u(k) = [C̄B] [y(K + 1)]
K = K + 1
K > K LIM
NO
Print results and Stop

Case II, $\quad D \neq 0$

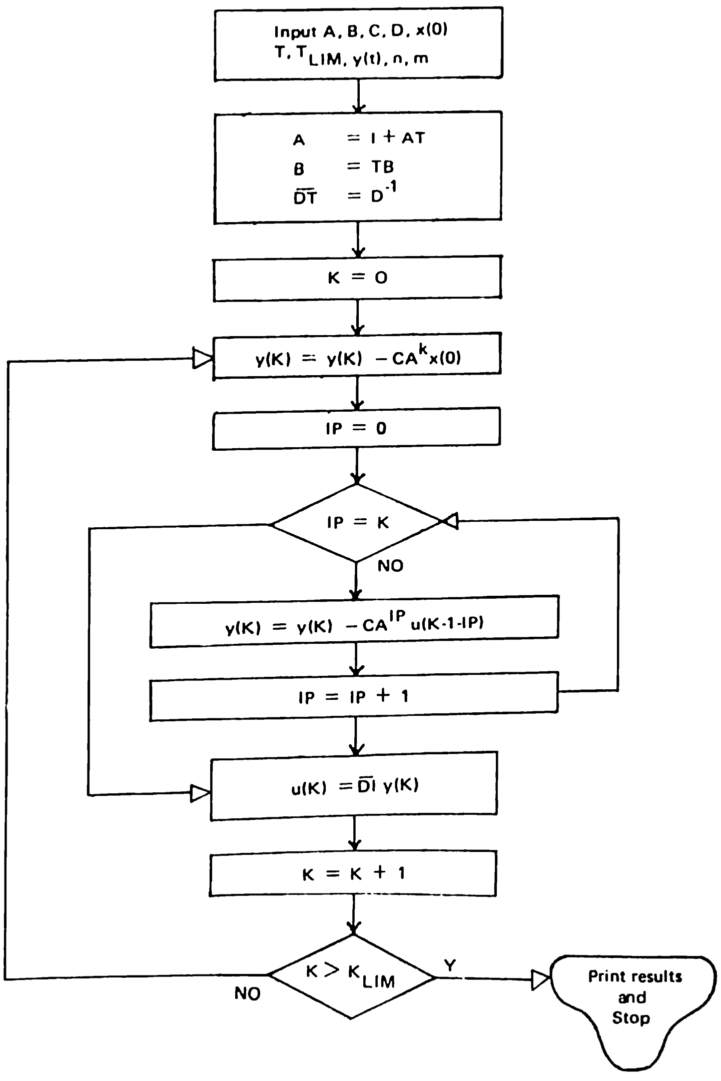
Expanding equation 11 gives the following equations.

$$
\begin{aligned}
y(0) &= Cx(0) + Du(0) \\
y(1) &= CAx(0) + CBu(0) + Du(1) \\
y(2) &= CA^2x(0) + CABu(0) + CBu(1) + Du(2) \\
&\ \ \vdots \\
y(K) &= CA^Kx(0) + CA^{K-1}Bu(0) + CA^{K-2}Bu(1) \\
&\qquad + \cdots + CBu(K-1) + Du(K)
\end{aligned}
\qquad (14)
$$

Solving the input variables from equation 14

$$u(0) = D^{-1} [y(0) - Cx(0)]$$
$$u(1) = D^{-1} [y(1) - CAx(0) \quad CBu(0)]$$
$$u(2) = D^{-1} [y(2) - CA^2x(0) - CABu(0) - CBu(1)]$$
$$\vdots$$
$$u(K) = D^{-1} [y(K) - CA^Kx(0) \quad CA^{K-1}Bu(0) - CA^{K-2}Bu(1) - \ldots - CBu(K-1)]$$

Flow Chart for Case II



Input A, B, C, D, x(0) T, T$_{LIM}$, y(t), n, m
A = I + AT
B = TB
$\overline{DT}$ = D^{-1}
K = 0
y(K) = y(K) − CA^{k}x(0)
IP = 0
IP = K
NO
y(K) = y(K) − CA^{IP} u(K-1-IP)
IP = IP + 1
u(K) = $\overline{DI}$ y(K)
K = K + 1
K > K$_{LIM}$
NO
Y
Print results and Stop

LISTING OF PROGRAM ALGOI/BAS: A routine used in computing the input of a discretized linear system

```
10 DIM A(2,2),B(2,2),C(2,2),Y(2),CB(2,2),U(2),UA(2,100)
20 READ A(1,1),A(1,2),A(2.1),A(2,2)
30 DATA -3,-1,1,0
40 READ B(1,1),B(1,2),B(2,1),B(2,2)
50 DATA 1,0,0,1
60 READ C(1,1),C(1,2),C(2,1),C(2.2)
70 DATA 3,0,0,1
80 READ CB(1,1),CB(1,2),CB(2,1),CB(2,2)
90 DATA 333.3333,0,0,1000
100 READ T:DATA 0.001
110 DEF FNY1(K,T)=(9/2)*(1-EXP(-2*K*T))-6*(1-EXP(-K*T))
120 DEF FNY2(K,T)=3*(1-EXP(-K*T))-(1-EXP(-2*K*T))
130 FOR I=1 TO 2
140 FOR J=1 TO 2
150 IF I=J THEN A(I,J)=1.0 + A(I,J)*T :GOTO 170
160 A(I,J)=A(I,J)*T
170 B(I,J)=T*B(I,J)
180 NEXT J
190 NEXT I
200 FOR K=0 TO 20
210 GOSUB 360
220 IF K=0 THEN 230ELSE 300
230 GOSUB 710
240 FOR I=1 TO 2
250 FOR J=1 TO 2
260 U(I)=CB(I,J)*Y(J) +U(I)
270 NEXT J
280 NEXT I
290 GOSUB 610:GOTO 310
300 GOSUB 390
310 NEXT K
320 FOR K=0 TO 20
330 LPRINT FNY1(K,T),FNY2(K,T),UA(1,K),UA(2.K)
340 NEXT K
350 END
370 Y(1)=FNY1(K+1,T): Y(2)=FNY2(K+1,T)
380 RETURN
390 IP=1
400 IF IP=1 THEN GOSUB 650:GOTO 410ELSE GOSUB 720
410 GOSUB 620
420 FOR I=1 TO 2:FOR J=1 TO 2:FOR JJ=1 TO 2
430 RD(I,J)=RD(I,J)+C(I,JJ)*RT(JJ,J)
440 NEXT JJ:NEXT J: NEXT I
450 GOSUB 680
460 FOR I=1 TO 2:FOR J=1 TO 2·FOR JJ=1 TO 2
470 RP(I,J)=RP(I,J)+RD(I,JJ)*B(JJ,J)
480 NEXT JJ:NEXT J:NEXT I
490 GOSUB 710
500 FOR I=1 TO 2: FOR J=1 TO 2
510 U(I)=RP(I,J)*UA(I,K-IP)+U(I)
520 NEXT J: NEXT I
530  Y(1)=Y(1)-U(1):Y(2)=Y(2)-U(2)
540 IF IP=K THEN 550ELSE IP=IP+1 : GOTO 400
550 GOSUB 710
560 FOR I=1 TO 2: FOR J=1 TO 2
570 U(I)=U(I)+CB(I,J)*Y(J)
580 NEXT J:NEXT I
590 GOSUB 610
600 RETURN
610 UA(1,K)=U(1):UA(2,K)=U(2):RETURN
620 FOR I=1 TO 2:FOR J=1 TO 2
630 RD(I,J)=0
640 NEXT J: NEXT I: RETURN
650 FOR I=1 TO 2:FOR J=1 TO 2
660 RT(I,J)=A(I,J)
670 NEXT J: NEXT I: RETURN
680 FOR I=1 TO 2:FOR J=1 TO 2
690 RP(I,J)=0
700 NEXT J: NEXT I: RETURN
710 U(1)=0:U(2)=0:RETURN
720 GOSUB 620
730 FOR I=1 TO 2:FOR J=1 TO 2:FORJJ=1 TO 2
740 RD(I,J)=RD(I,J)+RT(I,JJ)*A(JJ,J)
750 NEXT JJ: NEXT J: NEXT I: RETURN
760 FOR I=1 TO 2:FOR J=1 TO 2
770 RT(I,J)=RD(I,J)
780 NEXT J: NEXT I
790 RETURN
```

TABLE I. Listing of the Computed Input

Y1(K,T)	Y2(K,T)	UA(1,K)	UA(2,K)	K
0	0	99799	1.00052	0
2.99397E-03	1.00052E-03	.997929	1.00054	1
5.97578E-03	2.00206E-03	.994055	1.00143	2
8.94597E-03	3.00449E-03	.990165	1.00232	3
.0119045	4.00782E-03	.986067	1.00346	4
.0148509	5.01227E-03	.982086	1.00441	5
.0177853	6.01769E-03	.978077	1.00548	6
.0207077	7.02417E-03	.974187	1.00637	7
.0236186	8.03155E-03	.970297	1.00727	8
.0265178	9.03982E-03	.966288	1.00834	9
.029405	.0100492	.962517	1.00905	10
.0322809	.0110592	.958389	1.0103	11
.0351445	.0120705	.954587	1.01114	12
.0379968	.0130827	.950787	1.01197	13
.0408377	.0140957	.946779	1.01304	14
.0436666	.0151097	.942978	1.01388	15
.0464842	.0161246	.939059	1.01489	16
.0492901	.0171405	.935347	1.01566	17
.0520848	.0181572	.931339	1.01673	18
.0548676	.0191749	.927627	1.01751	19
.0576393	.0201935	.923917	1.01828	20

NOTE:	UA(1,t)	=	Actual input no. 1 = 1.0
	UA(2,t)	=	Actual input no. 2 = 1.0
	Y1(t)	=	Actual output no. 1 = 4.5 { 1-exp(-2t)} -6{1-exp(-t) }
	Y2(t)	=	Actual output no. 2 = 3.0 { 1-exp(-t) } – { 1-exp(-2t) }
	T	=	Sampling period = 0.001

The given example is for case I, D = 0.

Example:

A circuit satisfying Case II is given in Figure 1.

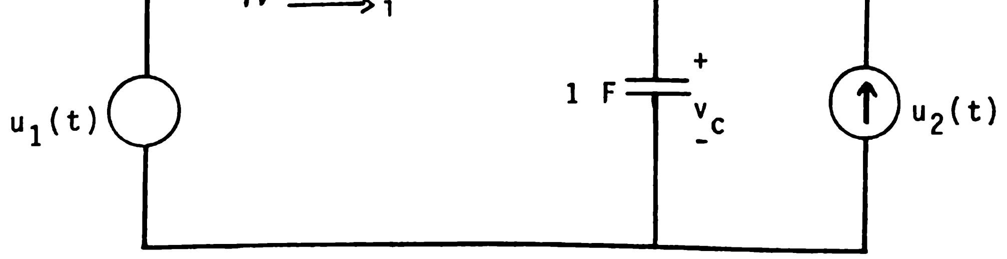


Fig. 1. Example for Case II.

The equations describing the circuit are:

$$\begin{bmatrix} \frac{di}{dt} \\ \frac{dv_c}{dt} \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ v_c \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ v_c \end{bmatrix} = \begin{bmatrix} 4.5(1-e^{-2t})-3(1-e^{-t}) \\ 3(1-e^{-t}) - (1-e^{-2t}) \end{bmatrix}$$

The program ALGOI/BAS, the routine used to compute for the given output is shown. Also Table I, lists the values of the computed inputs for $K = 0, 1, 2, \ldots 20$. The actual value of the inputs are $u_1(t) = 1$ and $u_2(t) = 1$.

Conclusion

The method discussed does not require computation of the state transition matrix. Computing for the state transition matrix, especially for large n, is time consuming. The method is also not limited to cases where $D = 0$. The limitation of the method is the need to store the previous values of the input variable in order to compute for the present considered input variable. However, curve fitting algorithms could be used after several samples of the input variables are obtained.

Acknowledgment

I sincerely thank Prof. Miguel T. Escoto, Jr., and Dr. Francisco Viray for the help and encouragement that they gave me in writing this paper. I also thank Ms. Medy Santiago for allowing me to use the Office of the College Secretary's computer.

References

Bajd, T., *"Computing The Input To A Linear Model"*, Simulation Vol. 40, No. 6, June 1983, pages 241 – 243.

Kuo, B., *Automatic Control System*, 4th Edition, Prentice-Hall, Inc., New Jersey, 1982, pages 272-279.