LUND UNIVERSITY

Real Time Computing I Implementing Linear Filters

Hagander, Per; Källström, Claes; Åström, Karl Johan

1971

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Hagander, P., Källström, C., & Åström, K. J. (1971). *Real Time Computing I Implementing Linear Filters*. (Research Reports TFRT-3031). Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
3

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

REAL TIME COMPUTING I
IMPLEMENTING LINEAR FILTERS.

P. HAGANDER
C. KÄLLSTRÖM
K. J. ÅSTRÖM

REPORT 7106 (B) MAY 1971
LUND INSTITUTE OF TECHNOLOGY
DIVISION OF AUTOMATIC CONTROL

REAL TIME COMPUTING I
IMPLEMENTING LINEAR FILTERS.

P. Hagander
C. Källström
K.J. Åström

ABSTRACT.

When implementing control and filtering algorithms on a small digital computer or with special hardware it is important to know the trade offs between accuracy, sampling interval, computing time and storage. When planning process control systems it is also of interest to know orders of magnitude for storage and computing time for various tasks. This report, which is a first in a series will give some aspects on these problems, it does not contain any new results. It is rather a compilation of a few useful facts.

TABLE OF CONTENTS

1. INTRODUCTION.

The problem of implementing a linear filtering algorithm occurs frequently when a digital computer is used for on-line control. Since a finite dimensional discrete time system has the realization

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad (1)$$

the implementation simply consists of an algorithm for (1). If the system is specified by a continuous time realization there is also the additional problem of choosing a suitable sampling interval so that (1) is a good approximation of the continuous time system.

In practice there are several problems which have to be considered, i.e.

- accuracy required,
- computing time,
- storage requirements,
- programming convenience.

There is a certain amount of arbitrariness since the input-output relation is invariant to linear transformations of x. It turns out that all factors listed above are also influenced by the choice of state variables in (1).

2. STORAGE AND COMPUTING TIME.

It will first be investigated how storage and computing time are influenced by the choice of coordinates in the realization (1).

2.1. The General Case.

Let n_x be the number of states, n_u the number of inputs and n_y the number of outputs. In the general case the model (1) then requires

$$N_1 = n_x(n_x + n_u + n_y) + n_u n_y$$

parameters to represent the elements of the matrices A, B, C and D. To construct an algorithm it is furthermore necessary to have $2n_x + n_u + n_y$ storage cells for the vectors x(t+1), x(t), u(t) and y(t). To implement a code for (1) it is thus necessary to have a data area of

$$N_2 = n_x(n_x + n_u + n_y + 2) + n_u n_y + n_u + n_y$$

cells.

To get a crude estimate of the computing time we can just count the number of additions and multiplications in (1). We find that N_1 additions and N_1 multiplications are required. Notice, however, that this estimate does not include the computations required to evaluate the indices of vectors and matrices.

A FORTRAN subroutine GLISY, which implements (1), is given in appendix. This code is organized in such a

way that savings are obtained when D = 0 or when it is not necessary to compute the output. The subroutine has been compiled on PDP-15 and requires 177 cells. The actual computing time on PDP 15-30 as well as crude estimates computed as $2N_1 \cdot 200$ µs where 200 µs is the average addition and multiplication time are listed in Table 1.

Table 1 - Computing time for GLISY on PDP 15-30.

n_x	n_u	n_y	Remark	Execution time (msec)	Crude estimate (msec)	Execution time/N_1
4	3	2		13.82	16.8	0.33
4	3	2	D = 0	11.49	14.4	
4	3	0		9.23	11.2	
5	1	1		12.4	14.4	0.34
10	1	1		32.1	48.4	0.27
20	1	1		97.6	176.4	0.22
40	1	1		331.9	672.4	0.20
10	5	1		39.8	66.0	0.24
10	1	5		44.0	66.0	0.27
10	5	5		54.6	90.0	0.24
40	3	2		354.8	722.4	0.20

2.2. Systems on Diagonal Form.

If the matrix A has distinct eigenvalues it can always be transformed to diagonal form

$$\begin{aligned} z(t+1) &= z(t) + u(t) \\ y(t) &= z(t) + Du(t) \end{aligned} \qquad (2)$$

where all β_{ij} or all γ_{ij} can be set to arbitrary values. In this case the system can be represented by

$$N_3 = n_x(n_u + n_y + 1) + n_u n_y$$

parameters. To construct an algorithm it is also necessary to have $2n_x + n_u + n_y$ storage cells for $x(t+1)$, $x(t)$, $u(t)$ and $y(t)$. To implement a code for (2) it is thus necessary to have a data area with

$$N_4 = n_x(n_u + n_y + 3) + n_u n_y + n_u + n_y$$

cells.

The formula (2) requires

$$N_a = n_x(n_u + n_y) + n_u n_y$$

additions and

$$N_m = n_x(n_u + n_y + 1) + n_y n_u$$

multiplications.

A comparison with section 2.1 thus shows that it is possible to reduce both storage and computations significantly by transforming (1) to diagonal form if this is possible.

2.3. Companion Forms.

Completely observable systems with one output and completely controllable systems with one input can always be transformed to companion form. In these cases the storage requirements and the crude estimates of computing times are the same as for systems on diagonal form, if β or γ in (2) is set to $(1, 1, \ldots, 1)^T$.

Two subroutines, OBSY (OBservable SYstem) and COSY (COntrollable SYstem), which implements (1) in the observable and controllable cases are given in appendices B and C. The code OBSY requires 131 cells and COSY requires 122 cells. The actual computing times as well as the crude estimates are given in Table 2 and Table 3. Notice that COSY is shorter and considerably faster. For single input single output systems this algorithm should therefore be preferred. The fact that COSY is faster is that the multiplication $\Sigma a_i x_i(t)$ is done by the efficient scalar product algorithm, while in OBSY the computation of $a_i x_i$ all requires separate index evaluation.

Table 2 - Execution time for OBSY on PDP 15-30.

n_x	n_u	Execution time (msec)	Estimated execution time (msec)
4	3	8.72	7.6
40	3	79.17	65.2
5	1	8.63	4.4
10	1	16.67	8.4
20	1	32.55	16.4
40	1	64.32	32.4

Table 3 - Execution time for COSY on PDP 15-30.

n_x	n_y	Execution time (msec)	Estimated execution time (msec)
4	3	6.8	7.6
40	3	32.1	65.2
5	1	3.84	4.4
10	1	5.68	8.4
20	1	9.33	16.4
40	1	16.69	32.4

3. ACCURACY.

The analysis of section 2 indicates that the diagonal and the companion forms have distinct advantages in terms of both storage and computing time. The parameter accuracy required for the different algorithms will now be considered.

It is not obvious how the parameter accuracy should be defined. The ultimate measure is, of course, the degradation of the performance of the control system. This is, however, very difficult to evaluate, and therefore we might instead investigate the accuracy of the transfer function or the time response of the system.

We will first discuss the effect of the parameter accuracy on the stability of the system. This is, of course, quite crude because it corresponds to drastic changes in the system. We will later briefly discuss other measures of accuracy.

3.1. Stability Considerations.

We will first consider a few examples.

Example 1 - Diagonal systems.

Consider first a first order system

$$x(t+1) = ax(t) \qquad 0 < a < 1$$

Let δa denote the absolute error in a. In order to assume stability we must require

$$|\delta a| \leq 1-a$$

If the discrete system is a sampling of a continuous system

$$\dot{x} = -\alpha x \qquad\qquad a = \exp(-\alpha h)$$

it is obvious that the accuracy requirement becomes more and more stringent with shorter sampling interval. Approximately we get

$$|\delta a| \leq \alpha h$$

This simple stability analysis easily extends to the general real valued diagonal system, giving corresponding criteria for all diagonal elements.

Example 2 - Companion form.

Consider a n-th order system in companion form. Let it have equal eigenvalues a. The characteristic equation is

$$(z-a)^n = 0$$

Now assume that the characteristic equation is perturbed by δ then

$$(z-a)^n + \delta = 0$$

The roots of this equation are on a circle around $z = a$ with radius $\sqrt[n]{\delta}$. If we require that no root could possibly fall outside the unit circle we get

$$|\delta| \leq (1-a)^n$$

Hence if $a = 0.99$ and $n = 4$ we require

$$|\delta| \leq 10^{-8}$$

which is rather surprising.

This corresponds to errors only in the last coefficient.

Considering the coefficients of the characteristic equation one at a time the effect on the root spread can be examined, for instance the effect on the coefficient in the middle.

Example 3.

Let $n = 2m$ and perturb the m-th coefficient with δ

$$(z-a)^{2m} - \delta z^m = 0$$

$$(z-a)^2 = \varepsilon z$$

where

$$\varepsilon = \varepsilon_1 + \varepsilon_2 \cdot i$$

might be complex

$$\varepsilon = |\delta|^{1/m} \cdot e^{i2\pi/\frac{k}{m}} \qquad k = 0, \ldots, m-1$$

Make the substitution $z = -\frac{v+1}{v-1}$ and set $v = i\omega$ to obtain the limit of stability.

$$(+ 2a + \varepsilon + a^2 + 1)v^2 + 2(1 - a^2)v +$$

$$+ (1 - 2a - \varepsilon + a^2) = 0$$

$$\begin{cases} \varepsilon_1(1+\omega^2) = -(a+1)^2\omega^2 + (a-1)^2 & \text{(Real part)} \\ \varepsilon_2(1+\omega^2) = -2(a^2-1)\omega & \text{(Imaginary part)} \end{cases}$$

=>

=>

$$|\varepsilon|^2 = \varepsilon_1^2 + \varepsilon_2^2 = \left| \frac{(a+1)^2\omega^2 + (a-1)^2}{1+\omega^2} \right|^2$$

$$\frac{\partial|\varepsilon|}{\partial\omega^2} = \frac{4a}{(1+\omega^2)^2}$$

thus $\omega = 0$, $\varepsilon_2 = 0$, means most dangerous case and we must require

$|\varepsilon| < (1-a)^2$ and

$|\delta| < (1-a)^n$

just as in the previous case!

However, when implementing digital filters usually all the coefficients are perturbed at the same time. A natural way to restrict the errors in each coefficient would be

$$|\delta| < \frac{1}{n}(1-a)^n \qquad (3)$$

to assure stability in the worst case.

The described effects are, however, highly nonlinear and the different errors tend to eliminate each other, so even for the worst case the above criteria is too conservative.

3.2. Floating Point Arithmetic.

Realisation of digital filters on computers is usually done in floating point arithmetic.

When doing so the introduced errors are almost equal in all stored coefficients, if regarded as relative errors. This figure δ_{rel} is dependent on the word length of the computer, and is for PDP-15 c. 10^{-7}. This means by example 1

$$\left(\frac{1-a}{a} \right) > \delta_{rel} \qquad (4)$$

in the diagonal case, by example 2

$$\left(\frac{1-a}{a} \right)^n > \delta_{rel} \qquad (5)$$

for the last coefficient of a companion form, and by example 3

$$\frac{(1-a)^n}{a^{n/2}} \cdot \frac{1}{\binom{n}{n/2}} > \delta_{rel} \qquad (6)$$

for the coefficient in the middle of a companion form.

Again we see that too fast sampling leads to considerable simulation problems!

Even a moderate sampling interval is difficult when the system order is high and the system has equal or almost equal eigenvalues.

If $n = 6$ and $\delta_{rel} = 10^{-7}$ we must require at least

$$1 - a > \sqrt[6]{20 \cdot 10^{-7} \cdot a^3} = 0.1 \quad \text{or} \quad a < 0.9$$

only to assure stability. The "summation" effect from many errors is then neglected and would require even further restrictions.

Jordan form:

We have found that digital filters on companion form might be dangerous. The Jordan form seems to be the best alternative, since no diagonal matrix is similar to a companion matrix with equal eigenvalues. The stability requirements on a Jordan matrix are easily verified to be the same as for a diagonal matrix.

Example.

To illustrate that the crude estimates given above are useful we will give a numerical example. Consider a fourth order system with the pulse transfer function

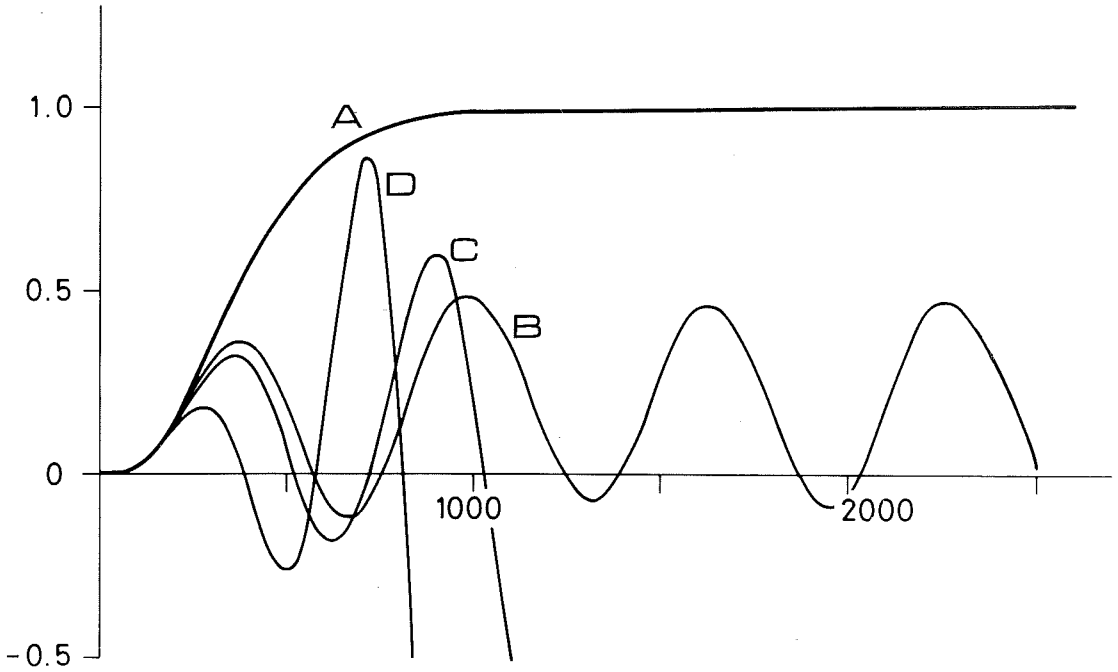
$$H(z) = \frac{10^{-8}}{(z-0.99)^4} \qquad (7)$$

This system has the Jordan form realization.

$$x(t+1) = \begin{bmatrix} 0.99 & 1.00 & 0 & 0 \\ 0 & 0.99 & 1.00 & 0 \\ 0 & 0 & 0.99 & 1.00 \\ 0 & 0 & 0 & 0.99 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 10^{-8} \end{bmatrix} u(t) \qquad (8)$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x(t)$$

The step response of the system computed in floating point arithmetic on PDP 15-30 using this realization is shown in Fig. 1. (Curve A)



The controllable canonical form of the system is

$$x(t+1) = \begin{bmatrix} 3.96 & -5.8806 & 3.881196 & -0.96059601 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot$$

$$\cdot\, x(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \qquad (9)$$

$$y(t) = [\, 0 \quad 0 \quad 0 \quad 10^{-8} \,]\, x$$

The step response of this realization is shown in Fig. 1. (Curve B)

The realization above is stable but its step response is vastly different from that of Curve A.

Now perturb the a_{14} coefficient with one unit in the eighth place:

$$a_{14} = -0.96059602 \quad \text{(instead of } -0.96059601\text{)}$$

and we get the step response in Fig. 1 (Curve C). The response is clearly unstable.

If the coefficient a_{13} is perturbed with one unit in the eighth place (to $a_{13} = 3.8811959$). The realization then becomes unstable. The step response is shown in Fig. 1. (Curve D)

The results obtained above are well in accordance with the estimates given previously. The relative accuracy of a floating point number on PDP 15-30 is $\delta_{rel} \approx 2^{-26} = 2 \cdot 10^{-8}$. Hence

$$\delta_{rel} \approx (1-0.99)^4$$

which was the stability limit found according to relation (5).

3.3. Other Criterias.

The companion form is unsuitable also when the eigenvalues are almost equal. The A matrix

$$\begin{bmatrix} a & 1 \\ 0 & b \end{bmatrix} \qquad (10)$$

is better. Although a transformation to diagonal form from the companion form is possible it is ill-conditioned for almost equal eigenvalues.

The simple stability calculations above give no information of these difficulties. The whole system must be considered or at least the uncontrolled system.

Example

$$\begin{cases} x(t+1) = \begin{bmatrix} a & 1 \\ 0 & b \end{bmatrix} x(t) \qquad x(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ y(t) = [\,1 \quad 0\,] \end{cases}$$

which is easily represented on diagonal form:

$$z(t+1) = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} z(t) \qquad z(0) = \begin{bmatrix} -1/(a-b) \\ 1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & \frac{1}{a-b} \end{bmatrix} z$$

Thus an error in the two diagonal elements will be multiplied by a factor $\frac{1}{a-b}$ in the diagonal form, which for almost equal eigenvalues spoils every solution.

Of course, there are systems which are well suited for representation in diagonal form. Originally uncoupled modes are for instance easily modelled in diagonal form. We have a sum of first order pulse transfer functions. The difficulties do not occur until we try to model a signal as the difference between two almost equal signals.

These signals should be modelled with the modified Jordan matrix (10).

3.4. Complex Eigenvalues.

Oscillating systems and oscillating subsystems are more difficult to model. Naturally we try to circumvent complex arithmetic and storage. The smallest subsystem to consider must be second order containing two complex conjugate poles, say

$\alpha \pm i\beta$

Then the forms

$$A_1 = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}, \quad A_2 = \begin{bmatrix} \alpha & -\beta^2 \\ 1 & \alpha \end{bmatrix}, \quad A_3 = \begin{bmatrix} 2\alpha & -(\alpha^2+\beta^2) \\ 1 & 0 \end{bmatrix}$$

are the most common alternatives.

The differences between A_1 and A_2 cannot be seen only from the simple stability reasoning, and are mainly due to different B and C magnifying the errors in different ways. Especially when β is small there is the question whether the form should be almost a diagonal form or almost a Jordan form.

The difference between A_2 and A_3 is small considering the stability; the change in the absolute value of the poles are almost the same. A_2, however, reproduces the oscillator frequence much better.

3.5. Comparison with the Continuous Case.

It has been shown that large sampled systems are difficult to represent on the companion form. The stability is easily affected.

For continuous system the companion form is a better alternative, still not very good, however. The parameters are disturbed and the roots are changed in the same way, but the stability criteria is quite different and the roots have another meaning!

It can be shown that relative errors less than one in the last coefficient never cause instability. The relative error in the worst coefficient can be as large as 10^{-3} for $n = 20$, equal eigenvalues, and the system is still stable.

A sampled system with the same state variables as a continuous system in companion form is more insensitive than the corresponding sampled system in companion form. The advantage of few coefficients other than one or zero is, however, lost.

4. CONCLUSIONS.

We can conclude that the choice of coordinates in the state space representation of a dynamic system will have a drastic influence on computing time, storage and accuracy. Diagonal forms and companion forms all require short computing and small storage. A general linear system of order 40 with one input and one output will for example require an execution time of 332 msec, while a representation on an observable companion form requires 64 msec, and a controllable companion form 17 msec. The difference between the observable and controllable companion form representations is due to the fact that the controllable representation can be coded more efficiently by using the scalar product subroutine. The companion form is, however, often extremely sensitive to parameter variations. This is particularly the case when the system has equal eigenvalues which are close to the unit circle. A situation like this will typically arise when a continuous time system is sampled with a short sampling interval. These facts have been illustrated by examples. Other important numerical aspects are not discussed in this report. It is for example most often best to choose as many parameters equal to natural numbers, which are represented exactly in the computer.

```
      SUBROUTINE GLISY(A,B,C,D,X,U,Y,NX,NU,NY,NOD)
C
C     COMPUTES THE NEW STATE X(T+1) AND THE NEW OUTPUT Y(T) FROM
C     X(T) AND U(T) FOR THE SYSTEM
C             X(T+1)=A*X(T)+B*U(T)
C             Y(T)=C*X(T)+D*U(T)
C
C     AUTHOR KJ ASTROM 1970-10-20
C
C     A- MATRIX OF ORDER NX*NX
C     B- MATRIX OF ORDER NX*NU
C     C- MATRIX OF ORDER NY*NX
C     D- MATRIX OF ORDER NY*NU
C     X- STATE VECTOR X(T) OF DIMENSION NX, RETURNED AS X(T+1)
C     U- INPUT VECTOR U(T) OF DIMENSION NU
C     Y- OUTPUT VECTOR Y(T) OF DIMENSION NY
C     NX-NUMBER OF STATES (MAX 64 SEE SECOND DIMENSION STATEMENT)
C     NU-NUMBER OF INPUTS (NO MAX)
C     NY-NUMBER OF OUTPUTS (NO MAX). PUT NY=0 IF COMPUTATION OF Y(T)
C        SHOULD BE SKIPPED.
C     NOD-PUT NOD=0 IF D=0 ELSE NOD=1
C
C     XO-DUMMY VECTOR OF DIMENSION NX CONTAINING THE
C        THE NEW STATE VECTOR X(T+1). STORED IN DUM8 OF
C        THE COMMON BLOCK /SLASK/. THE FIRST 896 CELLS OF /SLASK/
C        ARE NOT USED.
C
C     SUBROUTINES REQUIRED
C             NONE
C
      DIMENSION A(1,1),B(1,1),C(1,1),D(1,1),X(1),U(1),Y(1)
      DIMENSION XO(64),DUMY(448)
C
      COMMON /SLASK/ DUMY,XO
C
```

```
      SUBROUTINE COSY(A,C,D,X,U,Y,NX,NY)
C
C     COMPUTES THE NEW STATE X(T+1) AND THE NEW OUTPUT Y(T) FROM
C     X(T) AND U(T) FOR A SINGLE INPUT DYNAMICAL SYSTEM ON THE
C     CONTROLLABLE CANONICAL FORM
C
C            -A(1) -A(2) ... -A(NX-1) -A(NX)        1
C              1     0    ...     0        0         0
C     X(T+1)=  .     .            .        .  X(T)+  . U(T)
C              .     .            .        .         .
C              0     0    ...     1        0         0
C
C          C(1,1) C(1,2) ... C(1,NX)        D(1)
C            .      .          .              .
C     Y(T)=  .      .          .     X(T)+    .    U(T)
C            .      .          .              .
C          C(NY,1)C(NY,2)...C(NY,NX)        D(NY)
C
C     AUTHOR KJ ASTROM 1970-10-20
C
C     A- VECTOR OF DIMENSION NX
C     C- MATRIX OF ORDER NY*NX
C     D- VECTOR OF DIMENSION NY
C     X- STATE VECTOR OF DIMENSION NX, RETURNED CONTAINING X(T+1)
C     U- SCALAR INPUT
C     Y- OUTPUT VECTOR OF DIMENSION NY
C     NX-NUMBER OF STATES (NO MAX)
C     NY-NUMBER OF OUTPUTS (NO MAX)
C
C     SUBROUTINE REQUIRED
C           NONE
C
      DIMENSION A(1),C(1,1),D(1),X(1),Y(1)
C
```

```
      SUBROUTINE OBSY(A,B,D,X,U,Y,NX,NU)
C
C     COMPUTES THE NEW STATE X(T+1) AND THE NEW OUTPUT Y(T) FROM
C     X(T) AND U(T) FOR A SINGLE OUTPUT LINEAR SYSTEM ON THE
C     OBSERVABLE CANONICAL FORM
C
C              -A(1)      1     0   ...   0          B(1,1) ... B(1,NU)
C              -A(2)      0     1   ...   0            .            .
C                .        .     .         .            .            .
C     X(T+1)=    .        .     .         . X(T)+      .            .    U(T)
C                .        .     .         .            .            .
C              -A(NX-1)   0     0   ...   1            .            .
C              -A(NX)     0     0   ...   0          B(NX,1)... B(NX,NU)
C
C     Y(T)  =  1          0     0   ...   0 X(T) + D(1) ... D(NU) U(T)
C
C     AUTHOR KJ ASTROM 1970-10-20
C
C     A- VECTOR OF DIMENSION NX
C     B- MATRIX OF DIMENSION NX*NU
C     D- VECTOR OF DIMENSION NU
C     X- STATE VECTOR OF DIMENSION NX, RETURNED CONTAINING X(T+1)
C     U- INPUT VECTOR OF DIMENSION NU
C     Y- SCALAR OUTPUT
C     NX-NUMBER OF STATES (NO MAX)
C     NU-NUMBER OF INPUTS (NO MAX)
C
C     SUBROUTINES REQUIRED
C             NONE
C
      DIMENSION A(1),B(1,1),D(1),X(1),U(1)
C
```