MIDWEST APPLIED SCIENCE CORPORATION • WEST LAFAYETTE, INDIANA

1205 Kent Avenue
McClure Research Park
Phone 743-2161

# FINAL REPORT ON THE DEVELOPMENT OF COMPUTATIONAL TECHNIQUES FOR THE IDENTIFICATION OF LINEAR AND NONLINEAR MECHANICAL SYSTEMS SUBJECT TO RANDOM EXCITATION

Part II

By

Dr. F. Kozin

and

Dr. C. H. Kozin

For

Midwest Applied Science Corp.
McClure Research Park
1205 Kent Avenue
West Lafayette, Indiana

This Research was Conducted by MASC
for the
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland
under
Contract No. NAS5-10106

# A DYNAMICAL MOMENT TECHNIQUE FOR THE IDENTIFICATION OF SYSTEMS

## I. Introduction

The purpose of this report is to describe a new identification technique that appears to be of superior practical significance due to its simplicity both in comprehending and applying the technique.

The technique is based upon relatively straightforward principles of random function analysis, merely involving the estimation of various order moments of the input and output processes.

In a preliminary investigation of the technique for identification of digitally simulated linear systems, the technique produced quite useful approximations to the actual system parameters, with a relatively small number of calculations. Hence, we feel that it warrants further investigation on this basis alone. Even more interesting is that the technique does not appear to be limited to linear systems alone. Since, theoretically, the technique is the same for linear or non-linear systems, that can be described by differential equations with polynomial non-linearities. Of course, it remains to be seen how applicable the technique will be for non-linear systems.

Before describing the technique, we wish to make two points clear relative to this technique and to the general subject of identification of real systems.

First, there exist techniques available for the study of linear systems. The newer techniques may employ cross-correlations or cross-spectral densities for the estimation of the impulse response function or for the frequency response function. These techniques require a random noise source and require the estimation of the functions involved at various time lags or at various frequencies. The techniques that have been used classically employ a simple sinusoidal driver at various frequencies to determine the frequency response of the system. Techniques, such as ours, that require only a parameter estimation are just now emerging. These techniques offer the promise of economy in calculations and time. The establishment of such techniques as practical tools will be of fundamental importance in future complex systems design and development.

References and explanations of the many approaches to this subject may be found in "Identification of Linear and Non-Linear Systems Cross-Correlations Techniques", by Dr. W. Gersch conducted by MASC for NASA-Goddard under NAS5-9741, October 1965.

The second point that must be made concerning the identification of systems is that the methods that have been developed, as well as the method we describe in this report, are methods that will identify the analytical model or simulated model of the actual physical system. Hence, if the analytical or simulated model is not a satisfactory equivalent or approximation to the system, then clearly one is not identifying the real system.

Thus, any identification scheme is only as good as the analytical model that will be used to describe the physical system.

Of course, identification schemes can be used to help provide a better model to the system, if it is found that the original assumption is poor.

With this understanding of the proper role of identification techniques, we can now proceed to describe our approach.

## II. A General Description of an Identification Technique for Linear Systems.

The basic principle of the technique that we propose is that the system undergoing study is time invariant and is being driven by a statistically stationary, non-white noise random process. This rules out flat wide band noise as inputs. We shall discuss the reason for this requirement below.

We shall assume, furthermore, that for all practical purposes the system is in steady state operation. Thus, the output process is a statistically stationary random process. This implies that all of its moments are constant in time.

Now if we write the general time invariant linear system subjected to random input as

$$\dot{\bar{y}}(t) = A\,\bar{y}(t) + \bar{x}(t)\,, \tag{2.1}$$

where

$$\bar{y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix}, \qquad \bar{x}(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, \tag{2.2}$$

then the $\bar{y}$-process is statistically stationary and, assuming that all moments of the $\bar{x}$-process exist, it follows that the moments

$$E\,\{y_1^{r_1}(t)\; y_2^{r_2}(t)\cdots y_n^{r_n}(t)\;\; x_1^{s_1}(t)\cdots x_n^{s_n}(t)\} \qquad (2.3)$$

exist and are constant in time.

Upon multiplying equation (2.1) by $\vec{y}^T$ (the transpose of the state vector), we obtain

$$\dot{\bar{y}}(t)\;\bar{y}(t)^T = A\;\bar{y}(t)\;\bar{y}(t)^T + \bar{x}(t)\;\bar{y}(t)^T. \qquad (2.4)$$

Taking expectations of the equation (2.4) yields

$$E\,\{\dot{\bar{y}}(t)\;\bar{y}(t)^T\} = A\;E\{\bar{y}(t)\;\bar{y}(t)^T\} + E\{\bar{x}(t)\bar{y}(t)^T\} \qquad (2.5)$$

or

$$[E\{\dot{\bar{y}}(t)\bar{y}(t)^T\} - E\{\bar{x}(t)\bar{y}(t)^T\}]\;\;E\{\bar{y}(t)\bar{y}(t)^T\}^{-1} = A \qquad (2.6)$$

assuming $E\{\bar{y}(t)\bar{y}(t)^T\}^{-1}$ exists.

The relation (2.6) presents a general form of the solution of the general problem for the linear system.

It is easily seen that $E\{\bar{y}(t)\bar{y}(t)^T\}$ is symetric and, furthermore, since for any $(\alpha_1, \cdots, \alpha_n)$,

$$E\,\{|\sum_{i=1}^{n} \alpha_i y_i(t)|^2\} \geq 0, \text{ it follows that } E\{\bar{y}(t)\bar{y}(t)^T\}$$

is non-negative definite.

If the system equations are linearily independent, we shall expect that $E\{\bar{y}(t)\bar{y}^T(t)\}$ is positive definite and that its inverse exists. Thus, the matrix A is identified by estimation of the correlation matrices in Equation (2.6).

For our applications, we wish to be less general and somewhat more specific. We are assuming that the $\bar{y}$-process is a stationary, mean square differentiable process. Such a process is generated, for example, by passing a stationary mean square continuous process (i.e. a process whose covariance is continuous at the origin) through the time invariant linear system. This is the reason that we are ruling out wide band white noise processes. For, by passing a wide band, essentially white-noise, through the system the mean square differentiable property will cease to hold which, as can be shown, yields a bias to our estimations.

By our stationarity assumptions, it follows that the following moments, for the components of the vector $\bar{y}$-process,

$$E\,\{y_i^n(t)\} \quad \text{and} \quad E\,\{y_i^n(t)y_j^m(t)\} \tag{2.7}$$

exist and are constant in time.

Hence, it follows that

$$\begin{aligned} \frac{d}{dt}\, E\{y_i^n(t)\} &= 0 \\ \frac{d}{dt}\, E\{y_i^n(t)\, y_j^m(t)\} &= 0. \end{aligned} \tag{2.8}$$

Due to our assumption of mean square differentiability, the derivatives in Equation (2.8) can be taken inside the expectations to give

$$\text{a)}\quad E\{y_i^{n-1}(t)\ \dot{y}_i(t)\} = 0 \qquad\qquad (2.9)$$

$$\text{b)}\quad n\,E\{y_i^{n-1}(t)\ \dot{y}_i(t)\ y_j^m(t)\} + m\,E\{y_i^n(t)\ y_j^{m-1}(t)\ \dot{y}_j(t)\} = 0$$

In particular it follows that, for $n=m=1$ in 2.9b), $n=2$ in 2.9a)

$$\text{a)}\quad E\{y_i(t)\ \dot{y}_i(t)\} = 0 \qquad\qquad (2.10)$$

$$\text{b)}\quad E\{y_i(t)\ \dot{y}_j(t)\} + E\{\dot{y}_i(t)\ y_j(t)\} = 0.$$

The first equality in Equation (2.10) states the well-known fact that a stationary process and its derivative are uncorrelated at any given time.

With these last few statements, we can now show how to identify a few specific linear systems.

## III. Examples

In the following examples the x-process is a stationary mean square continuous, non-white, random process and the y-process is the stationary solution.

### Example 1. A First-Order System

Consider the system

$$\dot{y}(t) = -a\, y(t) + x(t) \tag{3.1}$$

Upon multiplying Equation (3.1) by $y(t)$ and taking the expectations we have

$$E\{y(t)\, \dot{y}(t)\} = -a\, E\{y^2(t)\} + E\{y(t)\, x(t)\}. \tag{3.2}$$

But, by (2.10) it follows that

$$a = \frac{E\{y(t)\, x(t)\}}{E\{y^2(t)\}}. \tag{3.3}$$

### Example 2. An Oscillator

Consider the second order system

$$\begin{aligned} &\text{a)} \quad \dot{y}_1(t) = y_2(t) \\ &\text{b)} \quad a\, \dot{y}_2(t) + b\, y_2(t) + c\, y_1(t) = x(t) \end{aligned} \tag{3.4}$$

Upon multiplying 3.4b) by $y_1(t)$, $y_2(t)$, $y_2^2(t)$ respectively and taking the expectations, we obtain

$$\begin{aligned} a\, E\{y_1(t)\, \dot{y}_2(t)\} + b\, E\{y_1(t)\, y_2(t)\} + c\, E\{y_1^2(t)\} &= E\{y_1(t)\, x(t)\} \\ a\, E\{y_2(t)\, \dot{y}_2(t)\} + b\, E\{y_2^2(t)\} + c\, E\{y_1(t)\, y_2(t)\} &= E\{y_2(t)\, x(t)\} \\ a\, E\{y_2^2(t)\, \dot{y}_2(t)\} + b\, E\{y_2^3(t)\} + c\, E\{y_2^2(t)\, y_1(t)\} &= E\{y_2^2(t)\, x(t)\} \end{aligned} \qquad (3.5)$$

But, by equations (2.10) and (3.4a), we have

$$\begin{aligned} E\{y_2(t)\, \dot{y}_2(t)\} &= E\{y_2^2(t)\, \dot{y}_2(t)\} = 0 \\ E\{y_1(t)\, y_2(t)\} &= E\{y_1(t)\, \dot{y}_1(t)\} = 0 \\ E\{y_1(t)\, \dot{y}_2(t)\} &= -E\{\dot{y}_1(t)\, y_2(t)\} = -E\{y_2^2(t)\}. \end{aligned} \qquad (3.6)$$

The equations 3.5) may now be written as

$$\begin{aligned} -a\, E\{y_2^2(t)\} \qquad\qquad + c\, E\{y_1^2(t)\} &= E\{y_1(t)\, x(t)\} \\ b\, E\{y_2^2(t)\} &= E\{y_2(t)\, x(t)\} \\ b\, E\{y_2^3(t)\} + c\, E\{y_2^2(t)\, y_1(t)\} &= E\{y_2^2(t)\, x(t)\} \end{aligned} \qquad (3.7)$$

from which a, b, c can be solved.

Example 3. A Coupled Oscillator

We now consider the system

$$
\begin{aligned}
\dot{y}_1(t) &= y_2(t) \\
\dot{y}_2(t) &= -a\ y_2(t) - b\ y_1(t) + c[y_4(t)-y_2(t)] + d[y_3(t)-y_1(t)] \\
\dot{y}_3(t) &= y_4(t) \\
\dot{y}_4(t) &= -c[y_4(t)-y_2(t)] - d[y_3(t)-y_1(t)] + x(t).
\end{aligned}
\tag{3.8}
$$

The second equation of (3.8) is multiplied by $y_1, y_2$ and the resulting equations are averaged. The fourth equation of (3.8) is multiplied by $y_3, y_4$ and the resulting equations are averaged.

On the basis of (2.10) and (3.8), the following four equations will result.

$$
\begin{aligned}
-E\{y_2^2\} &= -b\ E\{y_1^2\} + c\ E\{y_1 y_4\} + d[E\{y_1 y_3\} - E\{y_1^2\}] \\
0 &= -a\ E\{y_2^2\} + c[E\{y_4 y_2\} - E\{y_2^2\}] + d\ E\{y_2 y_3\} \\
E\{y_4^2\} + E\{x y_3\} &= -c\ E\{y_2 y_3\} + d[E\{y_3^2\} - E\{y_1 y_3\}] \\
E\{x y_4\} &= +c[E\{y_4^2\} - E\{y_4 y_2\}] - d\ E\{y_1 y_4\}
\end{aligned}
\tag{3.9}
$$

The set of four linear algebraic equations in four unknowns given by (3.9) are easily solved to determine the parameters (a,b,c,d).

Thus, it is easily established that

$$c = \frac{\begin{vmatrix} E\{y_4^2\} + E\{xy_3\} & E\{y_3^2\} - E\{y_1y_3\} \\ E\{xy_4\} & -E\{y_1y_4\} \end{vmatrix}}{B},$$

$$d = \frac{\begin{vmatrix} -E\{y_3y_2\} & E\{y_4^2\} + E\{xy_3\} \\ E\{y_4^2\} - E\{y_4y_2\} & E\{xy_4\} \end{vmatrix}}{B}$$

where

$$B = \begin{vmatrix} -E\{y_3y_2\} & E\{y_3\} - E\{y_1y_3\} \\ E\{y_4^2\} - E\{y_4y_2\} & -E\{y_1y_4\} \end{vmatrix}$$

Similar expressions yield a,b, as well.

## IV. Numerical Examples

In this section we present the results of parameter identification obtained by digitally simulating each of the examples above.

The systems digitally simulated are

a) $\dot{y}(t) = -y(t) + x(t)$

b) $\begin{cases} \dot{y}_1(t) = y_2(t) \\ \dot{y}_2(t) = -4y_2(t) - 16y_1(t) + x(t) \end{cases}$

c) $\begin{cases} \dot{y}_1(t) = y_2(t) \\ \dot{y}_2(t) = -5y_2(t) - 100y_1(t) + x(t) \end{cases}$ (4.1)

d) $\begin{cases} \dot{y}_1(t) = y_2(t) \\ \dot{y}_2(t) = -3y_2(t) - 9y_1(t) + 4[y_4(t) - y_2(t)] + 16[y_3(t) - y_1(t)] \\ \dot{y}_3(t) = y_4(t) \\ \dot{y}_4(t) = -4[y_4(t) - y_2(t)] - 16[y_3(t) - y_1(t)] + x(t) \end{cases}$

The digital simulation of the input process, the x(t)-process, is a stationary gaussian process generated by taking a moving average of the form

$$x_N = \sum_{i=0}^{K-1} c_i \, v_{N-i} , \qquad (4.2)$$

$N = 0,1,2,3,\cdots,$

where the v's are independent identically distributed gaussian random variables generated by the usual computer routines, and the c's are weights chosen to obtain a specific covariance function.

For our example we chose $K = 10$,

$$c_i = 0.1 \quad \text{for} \quad i = 0, 1, 2 \cdots 9.$$

The moments calculated were simple arithmetic averages taken over the first 1300 output values, the second 1300 output values, and so on. The complete details of the digital simulation are given in Appendix A.

For the system,(4.1a), the first three groups of 1300 data points yield the estimates.

| $E\{y^2\}$ | $E\{xy\}$ | a |
|---|---|---|
| 5.6892 | 6.0682 | 1.0667 |
| 4.4014 | 4.8197 | 1.0950 |
| 3.3332 | 4.7634 | 1.0992 |

FIGURE 1.

For the system's (4.1 b,c),the first three groups of 1300 data points yielded (see following page),

| | $E\{y_1^2\}$ | $E\{y_1y_2\}$ | $E\{y_2^2\}$ | $E\{y_1x\}$ | $E\{y_2x\}$ | a | b |
|---|---|---|---|---|---|---|---|
| System (4.1-b) | .0637 | .0004 | .4862 | .5841 | 2.0500 | 4.2172 | 16.8006 |
| | .0710 | .0009 | .5657 | .6340 | 2.3313 | 4.1210 | 17.3080 |
| | .0678 | .0000 | .5450 | .5936 | 2.2500 | 4.1280 | 16.7810 |
| System (4.1-c) | .0029 | .0001 | .0881 | .2098 | .4074 | 4.6190 | 103.72 |
| | .0028 | .0001 | .0773 | .2094 | .3719 | 4.8120 | 103.27 |
| | .0028 | .0001 | .0873 | .2065 | .4084 | 4.673 | 103.74 |

FIGURE 2

For system (4.1d), the correlation matrices and parameter estimates on the first two runs are

| Run I | $y_1$ | $y_2$ | $y_3$ | $y_4$ |
|---|---|---|---|---|
| $y_1$ | .2253 | | | |
| $y_2$ | .0000 | .8223 | | |
| $y_3$ | .3216 | .0839 | .4700 | |
| $y_4$ | -.0837 | 1.1044 | .0004 | 1.6038 |
| x | .1418 | 1.8412 | .5172 | 3.5491 |
| Parameter Estimates | a=3.1790 | b=9.194 | c=4.3044 | d=16.7258 |

| Run II | | | | |
|---|---|---|---|---|
| $y_1$ | .2465 | | | |
| $y_2$ | -.0004 | .9255 | | |
| $y_3$ | .3505 | .0924 | .5105 | |
| $y_4$ | -.0929 | 1.2505 | .0004 | 1.8117 |
| x | .1000 | 2.1328 | .4837 | 3.9482 |
| Parameter Estimates | a=3.1708 | b=9.2560 | c=4.2525 | d=16.8094 |

FIGURE 3

## V. Comments and Conclusions

It is immediately obvious that the estimates obtained in the examples above are quite practical being less than 10%, and in most cases less than 5%, in error. The most significant feature of the estimates above is that their bias is almost always positive. That is, they do not fluctuate about the true value, but in fact they are almost all greater than the true value. We believe that this bias is a function of the nature of the noise input into the system. It can be demonstrated that if the noise input to the system approximates white noise into the system then the bias will increase. Hence, the noise source to the system should be smoothly filtered.

Exactly how the noise source effects the parameter estimates, remains to be studied and understood. It is our belief at this time, however, that a real system should be subjected to noise obtained from various filters to determine the robustness of the resulting system parameter estimates.

The stationarity assumption used, Equation (2.8), appears to be quite realistic in the simulated systems. Upon checking Figures 1, 2, 3, we see that the random functions and their derivatives possess correlation values that are orders of magnitude less than the other cross-correlations.

At this time we can only reiterate our sincere belief that this technique can be developed into one of significant engineering value both for linear and non-linear systems.

## APPENDIX A

This section describes briefly the computer program used to simulate a system of five (or less) first order differential equations and to compute the second order moments to be used for calculating the parameters of the system as described in Section III.

The method of Runge-Kutta integration is used in Subroutine RK3 to simulate a system of first order differential equations which are supplied by Function Subroutines FN1, FN2, FN3, FN4, and FN5. The external excitation to the system is a filtered white Gaussian noise which is generated by Subroutines GENR, GAUSS, and RANDU. The filtering process used in the Subroutine GENR is a weighted sum of NF number of independent noise samples generated by Subroutines GAUSS and RANDU. The second order moments are estimated in Subroutine MOMENT, the method used there is to calculate the sample moments of N samples, that is, sample moment $m_{jk} = \frac{1}{N} \sum_{i=1}^{N} Y_i(j)\, Y_i(k)$.

Autocorrelation of the variables up to 9 lags are calculated in Subroutine AUTCOR in order to check the smoothness of the noise input as compared to system. As mentioned in Section II, the noise must have smooth correlation function in order for the method to be valid.

Different systems can be simulated by using different Function Subroutines FN1, FN2, FN3, FN4 and FN5. Different choices of the number of samples, the input noise level, the filter constants and the sampling interval are controlled by 7 input cards. The description and the formats of the 7 input cards are as follows.

| | Col. No. | Format | Description |
|---|---|---|---|
| 1st Card | 1 | I1 | 0 if the simulated sample points are not to be printed.<br>1 if the simulated sample points are to be printed. |
| | 2 - 3 | I2 | Number of runs desired. |
| | 4 | I1 | 0 if each run starts from zero initial point.<br>1 if success runs are desired. |
| 2nd Card | 1 - 10 | I10 | Any odd integer up to 9 digits for the first entry to Subroutine GAUSS. |
| | 11 - 20 | F10.1 | Standard deviation of the noise. |
| | 21 - 30 | F10.1 | Mean value of the noise. |
| 3rd Card | 1 - 2 | I2 | Number of filter coefficients. |
| 4th Card | 1 - 80 | 16I5.1 | Filter coefficients, as many cards as needed should be used. |
| 5th Card | 1 - 10 | F10.1 | Sample interval for integration. |
| 6th Card | 1 - 2 | I2 | Number of intervals between values of samples stored for calculation. 1 is used for the results in Section IV. |
| 7th Card | 1 - 4 | I4 | Number of sample points, N, desired for each run, $N \leq 1300$. If larger number is desired, the dimension of y in the Main Program has to be changed accordingly. Note that the average of the moments estimated from successive K runs is equivalent to the moments estimated from KxN number of samples. |

The output consists of:

A) The mean value of each variable.

B) The second order moment matrix.

C) The autocorrelation up to 9 lags for each variable.

D) Sample points simulated if desired.

The prime reason for the simplicity of the program included in this appendix is that it was used mainly to check the moment estimates and establish the overall credibility and practicality of the technique for identifying simple simulated systems.

This program, therefore, must be considered as preliminary in that we have not allowed for an arbitrary number of degrees of freedom and we have not included the final arithmetic calculations of the estimated system parameters.

The studies included within this report are the initial investigations of the identification technique. Future investigations of this technique will include the composition of a comprehensive program that will accommodate linear systems of n-degrees of freedom and will carry the calculations to the final arithmetic stages providing a print out of the parameter estimates for prescribed classes of systems.

```
      DIMENSION Y(6,1300),X(1300),YI(6),FC(20) ,FMON(6),SMON(6),XMON(21)
     1,FX(20),R(10)
      EXTERNAL FN1,FN2,FN3,FN4,FN5,FN6
      READ(5,500) IO,NR,IR
      READ(5,501) IX,S,AM
      READ(5,502) NFC,(FC(I),I=1,NFC)
      READ(5,503) STEP,KS,NV
      KR=NR
      DO 15 I=1,NFC
      CALL GAUSS(IX,S,AM,V)
   15 FX(I)=V
    5 TI=0
      DO 10 I=1,6
   10 YI(I)=0
   17 DO 20 J=1,NV
      CALL GENR(F,FC,NFC,FX,IX,S,AM)
   20 X(J)=F
      CALL RK3(FN1,FN2,FN3,FN4,FN5,FN6,STEP,TI,YI,KS,NV,Y,X)
      DO 25 J=1,6
   25 YI(J)=Y(J,NV)
      TI=TI+NV*KS*STEP
      DO 30 J=1,NV
   30 Y(6,J)=X(J)
      CALL MOMENT(Y,6,NV,1,FMON,SMON,XMON)
      WRITE(6,604)
      WRITE(6,602) (FMON(J),J=1,6)
      DO 40 J=1,6
      CALL LOC(J,1,J1,6,6,1)
      CALL LOC(J,J,JJ,6,6,1)
   40 WRITE(6,602) (XMON(K),K=J1,JJ)
      DO 45 I=1,6
      CALL AUTCOR(Y,I,6,NV,10,R)
   45 WRITE(6,605) (R(J),J=1,10)
      IF(IO) 50,55,50
   50 WRITE(6,601) ((Y(J,I),J=1,6),I=1,NV)
   55 KR=KR-1
      IF(KR) 100,100,60
   60 IF(IR) 17,5,17
  500 FORMAT(I1,I2,I1)
  501 FORMAT(I10,2F10.1)
  502 FORMAT(I2/16F5.1)
  503 FORMAT(F10.1/I2/I4)
  601 FORMAT(7F10.3)
  602 FORMAT(1H0,6E16.8)
  604 FORMAT(1H1,7X,'Y1',16X,'Y2',16X,'Y3',16X,'Y4',16X,'Y5',16X,'X')
  605 FORMAT(1H0,10E11.3)
  100 STOP
      END
```

---

```
      SUBROUTINE AUTCOR(X,NOVAR,NV,NO,KS,R)
      DIMENSION X(1),R(1)
      DO 10 J=1,KS
   10 R(J)=0.0
      NOKS=NO-KS+1
      DO 50 I=1,NOKS
      IJ=(I-1)*NV+NOVAR
      DO 50 K=1,KS
      IK=(I+K-2)*NV+NOVAR
   50 R(K)=R(K)+X(IJ)*X(IK)
      DO 60 K=1,KS
   60 R(K)=R(K)/FLOAT(NOKS)
      RETURN
      END
```

---

```
      FUNCTION FN1(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN1=-Y1+X
      RETURN
      END
```

---

```
      FUNCTION FN2(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN2=Y3
      RETURN
      END
```

---

```
      FUNCTION FN3(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN3=-4.0*Y3-16.0*Y2+X
      RETURN
      END
```

---

```
       FUNCTION FN4(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN4=Y5
      RETURN
      END
```

---

```
      FUNCTION FN5(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN5=-5.0*Y5-100.*Y4+X
      RETURN
      END
```

---

```
      FUNCTION FN6(T,Y1,Y2,Y3,Y4,Y5,Y6,X)
      FN6=0.
```

```
      END
C
C     ..................................................................
C
C        SUBROUTINE RK3
C
C        PURPOSE
C           INTEGRATES A SYSTEM OF SIX FIRST ORDER DIFFERENTIAL
C           EQUATIONS AND PRODUCES A TABLE OF INTEGRATED VALUES
C
C        USAGE
C           CALL RK3(FN1,FN2,FN3,FN4,FN5,FN6,H,XI,YI,K,N,VAL,F)
C
C        DESCRIPTION OF PARAMETERS
C           FN1-FN6 -SIX USER-SUPPLIED FUNCTION SUBPROGRAMS GIVING
C                    DY/DX AS A FUNCTION OF (X,Y1,Y2,Y3,Y4,Y5,Y6)
C           H       -STEP SIZE
C           XI      -INITIAL VALUE OF X
C           YI      -VECTOR OF LENGTH SIX CONTAINING THE INITIAL VALUES
C                    FOR Y1,Y2,Y3,Y4,Y5,Y6
C           K       -THE DESIRED NUMBER OF STEPS OF SIZE H BETWEEN
C                    VALUES OF THE INTEGRALS STORED IN VAL
C           N       -THE NUMBER OF VALUES TO BE STORED IN VAL.  THE
C                    FINAL VALUE OF X WILL BE XI&(N*K*H).
C           VAL     -RESULTANT SIX BY N MATRIX CONTAINING THE INTEGRATED
C                    VALUES OF THE SIX EQUATIONS
C             F       - INPUT TO THE SYSTEM, ARRAY OF N
C
C        REMARKS
C           NONE
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           FN1,FN2,FN3,FN4,FN5,FN6 AS NOTED ABOVE
C           CALLING PROGRAM MUST HAVE FORTRAN EXTERNAL STATEMENT
C           CONTAINING NAMES OF FUNCTION SUBPROGRAMS LISTED IN CALL TO
C           RK3
C
C        METHOD
C           EXTENSION OF FOURTH ORDER RUNGE-KUTTA INTEGRATION ON A
C           RECURSIVE BASIS AS SHOWN IN F.B. HILDEBRAND, 'INTRODUCTION
C           TO NUMERICAL ANALYSIS', MCGRAW-HILL, NEW YORK, 1956
C
C     ..................................................................
C
      SUBROUTINE RK3(FN1,FN2,FN3,FN4,FN5,FN6,H,XI,YI,K,N,VAL,F)
      DIMENSION YI(6),Y(6),S1(6),S2(6),S3(6),S4(6),VAL(1) ,F(1)
C
C        ...............................................................
C
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C        C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C        STATEMENT WHICH FOLLOWS.
C
C     DOUBLE PRECISION H,XI,YI,VAL,Y,S1,S2,S3,S4,H2,X,T,XA,A1,A2,A3,
C    1                 A4,A5,A6,FN1,FN2,FN3,FN4,FN5,FN6
C
C        THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
C        APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
C        ROUTINE.
C
C        USER FUNCTION SUBPROGRAMS FN1-FN6 MUST BE IN DOUBLE PRECISION
C
C        ...............................................................
C
      H2=H/2.
      X=XI
      DO 10 I=1,6
   10 Y(I)=YI(I)
      DO 70 LL=1,N
      L=(LL-1)*6
      DO 69 JJ=1,K
C
C        COMPUTE K SUB 0
C
      DO 30 I=1,6
      GO TO (21,22,23,24,25,26),I
   21 T=FN1(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
      GO TO 30
   22 T=FN2(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
      GO TO 30
   23 T=FN3(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
      GO TO 30
   24 T=FN4(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
      GO TO 30
   25 T=FN5(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
      GO TO 30
   26 T=FN6(X,Y(1),Y(2),Y(3),Y(4),Y(5),Y(6),F(LL))
```

```
   30 S1(I)=H*1
      XA=X&H2
      A1=Y(1)&S1(1)/2.
      A2=Y(2)&S1(2)/2.
      A3=Y(3)&S1(3)/2.
      A4=Y(4)&S1(4)/2.
      A5=Y(5)&S1(5)/2.
      A6=Y(6)&S1(6)/2.
C
C        COMPUTE K SUB 1
C
      DO 40 I=1,6
      GO TO (31,32,33,34,35,36),I
   31 T=FN1(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 40
   32 T=FN2(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 40
   33 T=FN3(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 40
   34 T=FN4(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 40
   35 T=FN5(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 40
   36 T=FN6(XA,A1,A2,A3,A4,A5,A6,F(LL))
   40 S2(I)=H*T
      A1=Y(1)&S2(1)/2.
      A2=Y(2)&S2(2)/2.
      A3=Y(3)&S2(3)/2.
      A4=Y(4)&S2(4)/2.
      A5=Y(5)&S2(5)/2.
      A6=Y(6)&S2(6)/2.
C
C        COMPUTE K SUB 2
C
      DO 50 I=1,6
      GO TO (41,42,43,44,45,46),I
   41 T=FN1(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 50
   42 T=FN2(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 50
   43 T=FN3(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 50
   44 T=FN4(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 50
   45 T=FN5(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 50
   46 T=FN6(XA,A1,A2,A3,A4,A5,A6,F(LL))
   50 S3(I)=H*T
      XA=X&H
      A1=Y(1)&S3(1)
      A2=Y(2)&S3(2)
      A3=Y(3)&S3(3)
      A4=Y(4)&S3(4)
      A5=Y(5)&S3(5)
      A6=Y(6)&S3(6)
C
C        COMPUTE K SUB 3
C
      DO 65 I=1,6
      GO TO (51,52,53,54,55,56),I
   51 T=FN1(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 65
   52 T=FN2(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 65
   53 T=FN3(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 65
   54 T=FN4(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 65
   55 T=FN5(XA,A1,A2,A3,A4,A5,A6,F(LL))
      GO TO 65
   56 T=FN6(XA,A1,A2,A3,A4,A5,A6,F(LL))
   65 S4(I)=H*T
      X=X&H
C
C        COMPUTE NEW VALUES OF INTEGRALS
C
      DO 69 I=1,6
   69 Y(I)=Y(I)&(S1(I)&2.*S2(I)&2.*S3(I)&S4(I))/6.
      DO 70 I=1,6
      II=I&L
   70 VAL(II)=Y(I)
      RETURN
      END
      SUBROUTINE GENR(F,FC,NFC,FX,IX,S,AM)
      DIMENSION FC(1),FX(1)
      L=NFC-1
      DO 10 J=1,L
```

```
   10 FX(K+1)=FX(K)
      CALL GAUSS(IX,S,AM,V)
      FX(1)=V
      F=0
      DO 20 I=1,NFC
   20 F=F+FC(I)*FX(I)
      RETURN
      END
C                                                                       GAUSS001
C     ..................................................................GAUSS002
C                                                                       GAUSS003
C        SUBROUTINE GAUSS                                               GAUSS004
C                                                                       GAUSS005
C        PURPOSE                                                        GAUSS006
C           COMPUTES A NORMALLY DISTRIBUTED RANDOM NUMBER WITH A GIVEN  GAUSS007
C           MEAN AND STANDARD DEVIATION                                 GAUSS008
C                                                                       GAUSS009
C        USAGE                                                          GAUSS010
C           CALL GAUSS(IX,S,AM,V)                                       GAUSS011
C                                                                       GAUSS012
C        DESCRIPTION OF PARAMETERS                                      GAUSS013
C           IX -IX MUST CONTAIN AN ODD INTEGER NUMBER WITH NINE OR      GAUSS014
C               LESS DIGITS ON THE FIRST ENTRY TO GAUSS. THEREAFTER     GAUSS015
C               IT WILL CONTAIN A UNIFORMLY DISTRIBUTED INTEGER RANDOM  GAUSS016
C               NUMBER GENERATED BY THE SUBROUTINE FOR USE ON THE NEXT  GAUSS017
C               ENTRY TO THE SUBROUTINE.                                GAUSS018
C           S  -THE DESIRED STANDARD DEVIATION OF THE NORMAL            GAUSS019
C               DISTRIBUTION.                                           GAUSS020
C           AM -THE DESIRED MEAN OF THE NORMAL DISTRIBUTION             GAUSS021
C           V  -THE VALUE OF THE COMPUTED NORMAL RANDOM VARIABLE        GAUSS022
C                                                                       GAUSS023
C        REMARKS                                                        GAUSS024
C           THIS SUBROUTINE USES RANDU WHICH IS MACHINE SPECIFIC        GAUSS025
C                                                                       GAUSS026
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                  GAUSS027
C           RANDU                                                       GAUSS028
C                                                                       GAUSS029
C        METHOD                                                         GAUSS030
C           USES 12 UNIFORM RANDOM NUMBERS TO COMPUTE NORMAL RANDOM     GAUSS031
C           NUMBERS BY CENTRAL LIMIT THEOREM. THE RESULT IS THEN        GAUSS032
C           ADJUSTED TO MATCH THE GIVEN MEAN AND STANDARD DEVIATION.    GAUSS033
C           THE UNIFORM RANDOM NUMBERS COMPUTED WITHIN THE SUBROUTINE   GAUSS034
C           ARE FOUND BY THE POWER RESIDUE METHOD.                      GAUSS035
C                                                                       GAUSS036
C     ..................................................................GAUSS037
C                                                                       GAUSS038
      SUBROUTINE GAUSS(IX,S,AM,V)                                       GAUSS039
      A=0.0                                                             GAUSS040
      DO 50 I=1,12                                                      GAUSS041
      CALL RANDU(IX,IY,Y)                                               GAUSS042
      IX=IY                                                             GAUSS043
   50 A=A&Y                                                             GAUSS044
      V=(A-6.0)*S&AM                                                    GAUSS045
      RETURN                                                            GAUSS046
      END                                                               GAUSS047
C                                                                       RANDU001
C     ..................................................................RANDU002
C                                                                       RANDU003
C        SUBROUTINE RANDU                                               RANDU004
C                                                                       RANDU005
C        PURPOSE                                                        RANDU006
C           COMPUTES UNIFORMLY DISTRIBUTED RANDOM REAL NUMBERS BETWEEN  RANDU007
C           0 AND 1.0 AND RANDOM INTEGERS BETWEEN ZERO AND              RANDU008
C           2**31. EACH ENTRY USES AS INPUT AN INTEGER RANDOM NUMBER    RANDU009
C           AND PRODUCES A NEW INTEGER AND REAL RANDOM NUMBER.          RANDU010
C                                                                       RANDU011
C        USAGE                                                          RANDU012
C           CALL RANDU(IX,IY,YFL)                                       RANDU013
C                                                                       RANDU014
C        DESCRIPTION OF PARAMETERS                                      RANDU015
C           IX - FOR THE FIRST ENTRY THIS MUST CONTAIN ANY ODD INTEGER  RANDU016
C                NUMBER WITH NINE OR LESS DIGITS. AFTER THE FIRST ENTRY,RANDU017
C                IX SHOULD BE THE PREVIOUS VALUE OF IY COMPUTED BY THIS RANDU018
C                SUBROUTINE.                                            RANDU019
C           IY - A RESULTANT INTEGER RANDOM NUMBER REQUIRED FOR THE NEXTRANDU020
C                ENTRY TO THIS SUBROUTINE. THE RANGE OF THIS NUMBER IS  RANDU021
C                BETWEEN ZERO AND 2**31                                 RANDU022
C           YFL- THE RESULTANT UNIFORMLY DISTRIBUTED, FLOATING POINT,   RANDU023
C                RANDOM NUMBER IN THE RANGE 0 TO 1.0                    RANDU024
C                                                                       RANDU025
C        REMARKS                                                        RANDU026
C           THIS SUBROUTINE IS SPECIFIC TO SYSTEM/360                   RANDU027
C           THIS SUBROUTINE WILL PRODUCE 2**29 TERMS                    RANDU028
C           BEFORE REPEATING                                            RANDU029
C                                                                       RANDU030
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                  RANDU031
C           NONE
```

```
C        METHOD
C           POWER RESIDUE METHOD DISCUSSED IN IBM MANUAL C20-8011,
C           RANDOM NUMBER GENERATION AND TESTING
C
C     ..................................................................
C
      SUBROUTINE RANDU(IX,IY,YFL)
      IY=IX*65539
      IF(IY)5,6,6
    5 IY=IY&2147483647&1
    6 YFL=IY
      YFL=YFL*.4656613E-9
      RETURN
      END
      SUBROUTINE MOMENT(A,NV,NO,IXM,FMON,SMON,XMON)
C        A-  DATA MATRIX,NV BY NO
C        NV-  NO OF VARIABLES
C        NO-   NO OF OBSERVATIONS
C        IXM-   INDEX FOR CROSS MOMENTS
C                      IXM=0 DO NOT WANT CROSS MOMENTS
C                 IXM=1 WANT CROSS MOMENTS
C        FMON- FIRST MOMENTS ARRAY OF NV
C           SMON- SECOND MOMENTS ,ARRAY OF NV
C         CROSS MOMENTS , ARRAY OF (NV*(NV+1))/2
      DIMENSION A(1),FMON(1),SMON(1),XMON(1)
C     INITIALIZATION
C
      DO 10 J=1,NV
      FMON(J)=0
   10 SMON(J)=0
      DO 20 J=1,NV
      DO 20 L=1,J
      CALL LOC(J,L,K,NV,NV,1)
   20 XMON(K)=0
C
C     CALCULATE FIRST MOMENTS
C
      DO 30 I=1,NO
      DO 30 J=1,NV
      IJ=(I-1)*NV+J
   30 FMON(J)=FMON(J)+A(IJ)
      DO 40 J=1,NV
   40 FMON(J)=FMON(J)/NO
C
C     CALCULATE SECOND MOMENTS
C
      IF(IXM) 60,45,60
   45 DO 50 I=1,NO
      DO 50 J=1,NV
      IJ=(I-1)*NV+J
   50 SMON(J)=SMON(J)+A(IJ)*A(IJ)
      DO 55 J=1,NV
   55 SMON(J)=SMON(J)/NO
      GO TO 100
C
C     CALCULATE CROSS MOMENTS
C
   60 DO 70 I=1,NO
      DO 70 J=1,NV
      DO 70 L=1,J
      CALL LOC(J,L,K,NV,NV,1)
      IJ=(I-1)*NV+J
      IL=(I-1)*NV+L
   70 XMON(K)=A(IJ)*A(IL)+XMON(K)
      KN=(NV*(NV+1))/2
      DO 75 K=1,KN
   75 XMON(K)=XMON(K)/NO
  100 RETURN
      END
```

```
C
C     ..................................................................
C
C        SUBROUTINE LOC
C
C        PURPOSE
C           COMPUTE A VECTOR SUBSCRIPT FOR AN ELEMENT IN A MATRIX OF
C           SPECIFIED STORAGE MODE
C
C        USAGE
C           CALL LOC (I,J,IR,N,M,MS)
C
C        DESCRIPTION OF PARAMETERS
C           I   - ROW NUMBER OF ELEMENT
C           J   - COLUMN NUMBER  OF ELEMENT
C           IR  - RESULTANT VECTOR SUBSCRIPT
C           N   - NUMBER OF ROWS IN MATRIX
C           M   - NUMBER OF COLUMNS IN MATRIX
C           MS  - ONE DIGIT NUMBER FOR STORAGE MODE OF MATRIX
C                  0 - GENERAL
C                  1 - SYMMETRIC
C                  2 - DIAGONAL
C
C        REMARKS
C           NONE
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           MS=0   SUBSCRIPT IS COMPUTED FOR A MATRIX WITH N*M ELEMENTS
C                  IN STORAGE (GENERAL MATRIX)
C           MS=1   SUBSCRIPT IS COMPUTED FOR A MATRIX WITH N*(N&1)/2 IN
C                  STORAGE (UPPER TRIANGLE OF SYMMETRIC MATRIX). IF
C                  ELEMENT IS IN LOWER TRIANGULAR PORTION, SUBSCRIPT IS
C                  CORRESPONDING ELEMENT IN UPPER TRIANGLE.
C           MS=2   SUBSCRIPT IS COMPUTED FOR A MATRIX WITH N ELEMENTS
C                  IN STORAGE (DIAGONAL ELEMENTS OF DIAGONAL MATRIX).
C                  IF ELEMENT IS NOT ON DIAGONAL (AND THEREFORE NOT IN
C                  STORAGE), IR IS SET TO ZERO.
C
C     ..................................................................
C
      SUBROUTINE LOC(I,J,IR,N,M,MS)
C
      IX=I
      JX=J
      IF(MS-1) 10,20,30
   10 IRX=N*(JX-1)&IX
      GO TO 36
   20 IF(IX-JX) 22,24,24
   22 IRX=IX&(JX*JX-JX)/2
      GO TO 36
   24 IRX=JX&(IX*IX-IX)/2
      GO TO 36
   30 IRX=0
      IF(IX-JX) 36,32,36
   32 IRX=IX
   36 IR=IRX
      RETURN
      END
```