NASA Technical Memorandum 83120

A Computer Program for Estimating the Power-Density Spectrum of Advanced Continuous Simulation Language Generated Time Histories

H. J. Dunn

JUNE 1981

NASA

NASA Technical Memorandum 83120

A Computer Program for Estimating the Power-Density Spectrum of Advanced Continuous Simulation Language Generated Time Histories

H. J. Dunn
Langley Research Center
Hampton, Virginia

National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1981

SUMMARY

A computer program for the calculation of the power-density spectrum (PDS) from a data base generated by an Advanced Continuous Simulation Language (ACSL) simulation program is presented. The program uses an algorithm that employs the fast Fourier transform (FFT) to calculate the PDS of the variable. This is done by first estimating the autocovariance function of the variable and then taking the FFT of the smoothed autocovariance function to obtain the PDS. The PDS program is written so that it is transparent to the ACSL run-time executive (that is, the time history of the recorded variable is replaced with its PDS) and so that the ACSL user can perform a frequency analysis from the time-domain ACSL simulation model. An example of the use of the program is presented that determines the frequency content of the output waveform for a Van der Pol oscillator.

INTRODUCTION

In engineering science, mathematical models are often developed to aid in the understanding of the dynamic behavior of the physical system. These models also are of benefit in examining the results of proposed modifications in the physical system. The engineering models are often developed with the aid of digital computer simulation programs. One of these programs is the Advanced Continuous Simulation Language (ACSL) described in references 1 and 2. The ACSL system is designed for modelling the behavior of continuous systems described by time-dependent, nonlinear differential equations and transfer functions. The inputs into ACSL are in two distinct groups: the first group contains those inputs concerned with defining the model or structure of the system being simulated; the second group contains the sequence of commands that execute this model - i.e., change parameters, start runs, plot data. One of the outputs of the ACSL program can be the time history of specified dynamic variables, sampled at regular intervals and stored on a mass storage device. This record may be compared with the behavior of the physical system to determine the fidelity of the mathematical engineering model.

The accuracy of the engineering model may also be verified by use of power-density spectrum (PDS) analysis. The PDS of the model can be calculated and then compared with an experimentally measured PDS of the physical system. One method used in the past to do this analysis utilizes a numerical quadrature procedure, to calculate the complex Fourier integral, at a number of selected frequencies (see example program number 9 in ref. 1). If a complex system were being analyzed or a large number of frequencies were required to define the PDS, the numerical quadrature method could use a substantial amount of computer resources. This paper describes a computer program which utilizes a fast Fourier transform (FFT) technique to estimate the PDS of the model. Because of the efficiency of the FFT, this program should require less computer resources than other methods. The program is designed to be an extension of the ACSL system so that the PDS analysis can be performed as part of the ACSL sequence of commands that exercise the model. In the remainder of this paper, the PDS algorithm used

in this computer program is described, the computer program's subroutines are discussed, the use of the program is described, and an example of the program execution is presented.

PDS ALGORITHM

In this section, an overview of the algorithm used to calculate the PDS is presented. This section is not intended to be a tutorial on the subject of PDS analysis and computation. For further information on this subject the reader should consult reference 3.

The PDS $S(\omega)$ of the continuous dynamic variable $x(t)$ is defined as the Fourier transform of its autocorrelation function $R(\tau)$, or

$$S(\omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} R(\tau) \, d\tau$$

where the autocorrelation is defined as

$$R(\tau) = E[x(t)x(t+\tau)]$$

and E is the statistical expectation operator. Note that if $x(t)$ is a zero-mean function, i.e., $E[x(t)] = 0$, then $R(\tau)$ is also the autocovariance function. If a sequence of numbers is derived from the continuous variable by periodic sampling, the Fourier transform of the continuous variable can be approximated by

$$S(\omega) \approx TS(\Omega)$$

where $S(\Omega)$ is the discrete Fourier transform (DFT) of the sampled-data autocorrelation function and T is the sampling period. The FFT algorithm is a fast method of computing the DFT at a discrete number of frequencies. The FFT is used by the program discussed in this paper not only to calculate the PDS from the autocovariance sequence, but also to calculate the autocovariance sequence from the input data sequence.

The PDS algorithm used by the computer program described in this paper is suggested by Oppenheim and Schafer in reference 3. An estimate of the PDS is found by taking the FFT of the smoothed truncated autocovariance function of the time history. The truncated autocovariance function is calculated by taking the time history, composed of a data sequence of N data samples, and dividing it

into K data sequences, each of length L/2. The FFT of each of these smaller data sequences, augmented with L/2 zero samples (forming a data sequence of length L), are then used in a formula that uses the property of circular convolution to force the values of the autocovariance function to be correct for the first L/2 data elements. In doing so, the truncated autocovariance function is calculated by using K FFT's of length L and one inverse FFT for a real symmetric data sequence. This autocovariance function is then smoothed with a data window and the PDS is calculated by taking the FTT of the real symmetric data sequence. By using this method, N can be selected as large as necessary to obtain a good estimate of the autocovariance function, while the values of L and the sampling period are chosen to determine the frequency range and resolution of the PDS.

The subroutines that are used to calculate the FFT were obtained from reference 4, sections 1.2 and 1.3. In order to gain maximum efficiency, special purpose FFT algorithms were used for real input sequences and for real, symmetric input sequences. The basic FFT routine, which is common to all the special purpose routines, is a fast radix 8-4-2 algorithm which performs as many base-8 iterations as possible and then performs one base-4 or base-2 iteration, if necessary. Additional information on the FFT subroutines used can be found in reference 4.

PROGRAM DESCRIPTION

ACSL is an application oriented system for investigating the dynamic behavior of physical systems described by sets of differential equations. The interface of the PDS program to ACSL is illustrated in figure 1. Initially the model description is read by the ACSL translator. The output of the translator is passed to the run-time executive, which also reads the run-time commands from the user. With the proper commands, the ACSL executive generates a data base, the PREPARE file, consisting of the time histories of the specified dynamic variables. The ACSL executive uses this PREPARE file to generate listings and plots of the simulation. The PDS program of this paper, which is called by the executive, reads the PREPARE file, calculates the PDS's of the variables, and writes them on the PREPARE file. After the program has returned to the ACSL executive, the executive can be directed to list or plot the PDS's.

The structure of the PDS program is given in figure 2. There are four subroutines that were written for this report (PDSMAIN, PDS, WINDOW, INOUT). A listing of each of the routines described in this section can be found in appendix A. These subroutines call the FFT subroutines FFA, FFTSYM, and IFTSYM, which are described in reference 4. The subroutines ZZLINE and ZZPAGE are ACSL subroutines for output formating. The subroutines described in the subsequent sections make use of the imaginary part of a declared complex number being stored sequentially in memory after the real part of the complex number. For this reason, care should be taken with program modifications not to disturb the storage allocations in the calling statement to subroutine PDS. In the sections which follow, each of the programs developed in this report are discussed.

Subroutine PDSMAIN

Usage:

```
CALL PDSMAIN(A,MP,DELTT,LO2,IWIND,IFO2,AMAX,IDIV,LOG)
```

Argument list:

A — Work array with dimensions of AMAX or greater.

MP — Equal to the number of dynamic variables written on the PREPARE file by the run-time executive.

DELTT — The sampling interval at which the data were recorded.

LO2 — The number of data points in the calculated estimate of the autocovariance sequence, L/2. Must be a power of 2.

IWIND — Selects the type of data window to be applied to the autocovariance function:

IWIND	Window type
1	Rectangular, or no window
2	Bartlett
3	Hanning
4	Hamming
5	Blackman

IFO2 — The number of data points in the calculated PDS.

AMAX — The length of the work array A. Must be greater than:

512 + 2(MP-1) + MAXIMUM(A,B)

where

A = (LO2+2)(MP-1)

B = (IFO2+1)(MP-1)+IFO2+2

IDIV — If N is not equal to zero, the Nth dynamic variable is divided into the remaining variables processed by the program. If N is equal to zero, nothing is done.

LOG A logical variable, that if set true; the logarithm of the frequency is placed on the PREPARE file. If LOG is true and IDIV is not equal to zero, then the PDS of each variable, except for the Nth variable, is converted to decibels.

This routine is called the ACSL simulation program. The purpose of this routine is to organize the data storage required by the routine PDS. If the amount of storage required exceeds the value of the variable AMAX, subroutine PDS will not be called. Instead, the storage requirements will be printed and the program will be abnormally terminated. If there is enough storage in array A, subroutine PDS will be called. On return from the routine PDS, the amount of storage required is printed along with the mean and variance for each of the variables processed by subroutine PDS. PDSMAIN will then return normally to the calling program.

Subroutine PDS

Usage:

```
CALL PDS(BUFF,MP,KBUFF,A,X1,X1P1,MPM1,DELTT,XM,X2M,FREK,LP2,IWX,LO2P1,IFO2P1,
   IDIV,RA,RX1,RX1P1,FREKX,LOG)
```

Argument list:

BUFF Data storage buffer used by INOUT with dimensions MP by KBUFF.

MP Number of variables on the PREPARE file.

KBUFF Dimension of BUFF.

A A complex work matrix with dimension MPM1.

X1 A complex work matrix with dimensions LO2P1 by MPM1.

X1P1 A complex work matrix with dimensions LO2P1 by MPM1.

MPM1 MP-1.

DELTT Sampling interval.

XM A vector of length, MPM1, that on return contains the mean of the dynamic variables.

X2M On return contains the variance of the dynamic variables.

FREK A work matrix with dimensions IFO2P1 by MPM1.

LP2 L+2.

IWX The window-select integer, same as IWIND in subroutine PDSMAIN.

LO2P1 L/2+1.

IFO2P1 IFO2+1, where IFO2 is defined by PDSMAIN.

IDIV See subroutine PDSMAIN.

RA Work matrix that must be set equal to array A in the calling program. On returning to the calling program, the scaled autocovariance sequence is returned.

RX1 A work matrix that must be set equal to the array X1 in the calling program.

RX1P1 A work matrix that must be set equal to the array X1P1 in the calling program.

FREKX A work vector with dimension of IFO2P1.

LOG See subroutine PDSMAIN.

Subroutine PDS calculates the algorithm proposed on page 560 of reference 3. The major steps of the routine are as follows:

1. The mean and variance of the variables that are recorded on the PREPARE file are determined and stored.

2. The number of data sequences of length LO2 that exist on the PREPARE file are determined and stored in the variable K. The PREPARE file is then rewound.

3. For each of the K segments, as the data sequence is inputed the mean is subtracted and the data sequence is augmented with LO2 zero-data samples.

4. The FFT for each of the segments is calculated by the routine FFA and combined to form the input to the routine IFTSYM, which calculates the estimate of the autocovariance function.

5. The autocovariance function is normalized by the variance and the selected data window is applied by the routine WINDOW.

6. The scaled PDS is calculated, by the routine FFTSYM, by taking the FFT of the smoothed autocovariance function.

7. The PDS is corrected for scaling and outputed to the PREPARE file.

Subroutine WINDOW

Usage:

```
CALL WINDOW(A,LP2,MPM1,IWX,FREK,IFO2P1)
```

Argument list:

A	The input autocovariance sequence that is to be attenuated.
LP2	The number of data points in the autocovariance sequence.
MPM1	Number of dynamic variables being transformed.
IWX	The type of window to be used. See subroutine PDSMAIN.
FREK	Output smoothed autocovariance sequence.
IFO2P1	The number of data points in the output sequence.

The purpose of this subroutine is to attenuate the autocovariance sequence with a data window. On returning to the calling program, the array FREK contains the attenuated sequence of the first LO2P1 data elements with the remaining IFO2PI - LO2P1 data elements equal to zero.

Subroutine INOUT

Usage:

```
CALL INOUT(BUFF,N,M,IN,UN,IERR)
```

Argument list:

BUFF	Input data buffer.
N,M	Dimensions of BUFF, N*M must be less than 512.
IN	If equal to 1, the program will read data from the input device; if not equal to 1, the program will write to the output device.
UN	Input/Output device number.
IERR	Positive if I/O error occurred; zero if an end-of-file condition occurred; and negative if no I/O error occurred.

Subroutine INOUT is used to transmit data to and from the ACSL run-time executive. The routine INOUT is used to read and write data to the PREPARE file (unit number 8). The routine will be abnormally terminated if more than 512 words are to be transmitted or if an input/output error occurs. If, at some time in the future, the run-time executive's method of writing data on the PREPARE file is changed, this subroutine must be modified to conform to the new accession method.

PROGRAM OPTIONS AND USAGE

The purpose of this section is to describe several of the options that are available and to give some general guidelines on the use of the program. These guidelines are as follows:

1. Since the PDS subroutine replaces the first variable that is stored on the PREPARE file with the frequency, the independent variable of the simulation (time, T), should be placed as the first variable on the file.

2. The sampling interval is the communication interval (CINT) in the ACSL program. The sampling interval should be small enough so that aliasing does not occur. A general rule is to choose a sampling interval at least equal to the period of a frequency that is 20 to 40 times the highest frequency of interest. This rule is arbitrary because it assumes that the actual PDS is almost zero at this frequency.

3. The length of the data base is chosen so that the estimate of the autocovariance is ergodic. Since this length is dependent on the system and the random process, it must be chosen by trial and error or experience.

4. The frequency resolution of the PDS is determined by the sampling interval (DELTT in the listing) and the number of data points in the autocovariance function estimate (LO2) and is equal to 1./(2.*DELTT*LO2). The variable LO2 must be a power of 2, but since the FFT routine is a radix-8 type, LO2 should be a power of 4 for the optimum use of the FFT routines.

5. The PDS may be calculated at closely spaced frequencies, as is convenient and practical, by using the variable IFO2. The frequency increment between PDS data points is equal to 1./(2.*DELTT*IFO2). A general rule is to make IFO2 at least twice LO2.

6. No matter how large the variable IFO2 is, the frequency resolution of the PDS is still determined by the length of the autocovariance estimate (DELTT*LO2). If in the PDS program, IFO2 is smaller than LO2, LO2 is used instead of IFO2 for the frequency output interval calculation.

7. In the development of the program, the only feasible means for inputting the number of variables that are written on the PREPARE file (MP) to the program was as a passed variable in the calling sequence. If the user does not insure that MP is correct, unpredicted results can occur.

8. In parameter identification, it is useful to compare the magnitude of a frequency response curve to the PDS of one variable divided by the PDS of another variable. This can be done in the PDS program by setting the variable IDIV equal to N, where N represents the PDS of the variable that is to be divided into the other PDS's. No division takes place if IDIV is zero.

9. If the logical variable LOG in the calling sequence is set true, the logarithm of the frequency is written onto the PREPARE file. Also, if IDIV is not equal to zero, the PDS's of variables, except for the PDS of the Nth variable, are converted into decibels.

10. So that the calculated PDS can be easily compared to the magnitude of a transfer function, the square root of the PDS is taken before it is written on the PREPARE file.

EXAMPLE OF PROGRAM USE

An example is discussed in this section which shows the subroutine being used to calculate a PDS of a time history and the ACSL run-time executive plotting the results. The example presented is an ACSL program simulating a Van der Pol oscillator. The equation for the Van der Pol oscillator is similar to that of a free vibration of a spring mass system with viscous damping. However, the damping term of this equation is nonlinear in that it depends on the amplitude of the oscillation. At small amplitude levels, the damping is negative and the amplitude of the oscillation grows. At large amplitude values, the damping is positive and the amplitude diminishes. The combination of these two effects leads to a stable limit cycle. The equation for describing a Van der Pol oscillator can be written in parametric form as

$$\ddot{x} - \lambda(1 - x^2)\dot{x} + x = 0$$

where λ determines the growth rate of the oscillation and x is the state variable that represents the output of the oscillator. A dot over a symbol indicates the derivative with respect to time. In reference 5, Rogers and Connolly give an approximate solution for the fundamental frequency ω_o in radians per second as

$$\omega_o = 1 - \frac{\lambda^2}{8}$$

The amplitude of the fundamental frequency and the first two harmonics are defined as

$$A_0 = 2$$

$$A_1 = 0$$

$$A_2 = \frac{2\lambda}{9}$$

A listing of the ACSL translator input for this problem is given in appendix B. Except for the additional statements needed to specify the calculation of the PDS, the simulation program is identical to the example presented in reference 2.

The initial conditions are such that with $\lambda = 1$, the simulation will oscillate in a stable limit cycle with a fundamental frequency of 0.14 Hz. The subroutine is configured to calculate the autocovariance function with data sequences of 512 data samples, uses a Blackman data window and calculates a PDS having 1024 data points. Also, the program expects to find three variables recorded on the PREPARE file every 0.063 sec. Since this results in the recording of approximately 800 data points on the PREPARE file, there will be only one data sequence used in the algorithm. For this case, this will be sufficient since the autocovariance is a periodic and deterministic function. With the preceding values, the frequency resolution of the PDS will be 0.016 Hz, and the frequency increment between the plotted data points of the PDS will be 0.008 Hz.

A listing of the ACSL run-time executive input can be found in appendix C. The simulation is constructed so that the initial START command generates the data base to be used by PDSMAIN. This data base is displayed by the next two plot commands. Figure 3 is the output generated by the ACSL run-time executive for the first plot command listed in appendix C. This instruction produces a plot of the variable X, the oscillator output, on the y-axis against the variable T, time. A phase-plane plot of the system can be generated by plotting the derivative of x (XD in the simulation), against the variable X. Figure 4 was generated by the second PLOT command in appendix C. When the second START command is issued, the simulation has been reconfigured to call PDSMAIN and return to the executive. The last PLOT instruction produces figure 5, the desired PDS of the output variable, PDSX plotted against frequency W in Hertz. The peak of the first harmonic is not exactly what was predicted in reference 4. The applied window reduced the low-frequency peak. However, the third harmonic peak is in agreement with reference 5.

CONCLUDING REMARKS

A computer program for approximating the PDS from time histories generated and stored by an advanced continuous simulation language (ACSL) simulation program has been presented. The method uses an algorithm that employs the fast Fourier transform (FFT) of the time history to calculate an estimate of the autocovariance function of the time history. The power-density spectrum (PDS) is then calculated by taking the FFT of the windowed autocovariance function. An example was presented that determined the PDS for a Van der Pol oscillator.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
May 15, 1981

APPENDIX A

PROGRAM LISTING

```
      SUBROUTINE PDSMAIN(A,MP,DELTT,LO2,IWIND,IFO2,AMAX,IDIV,LOG)
C
C             PURPOSE     TO CALCULATE THE PDS OF VARIABLES
C                         WRITTEN TO THE PREPARE FILE
C
C
C             ANALYSIS    H.J. DUNN  -  NASA,LARC
C
C             WRITTEN     MARCH, 1980
C
C             ARGUMENT LIST
C
C     A       WORK ARRAY
C     MP      NUMBER OF VARIABLES WRITTEN ON THE PREPARE FILE
C     DELTT   SAMPLING INTERVAL
C     LO2     THE NUMBER OF DATA POINTS OF THE ESTIMATED
C             AUTOCOVARIANCE FUNCTION (MUST BE A POWER OF 2)
C     IWIND   SELECTS THE DATA WINDOW TYPE,
C               IWIND     WINDOW
C               1         RECTANGULAR
C               2         BARTLETT
C               3         HANNING
C               4         HAMMING
C               5         BLACKMAN
C     IFO2    NUMBER OF DATA POINTS OF THE PDS (MUST BE A POWER
C             OF 2)
C     AMAX    LENGTH OF THE WORK ARRAY A
C     IDIV    IF IDIV=N THEN DIVIDE THE PDS OF THE NTH VARIABLE
C             INTO THE PDS OF THE REMAINING VARIABLES ON THE PREPARE
C             FILE
C     LOG     IF LOG IS SET TRUE IN THE CALLING PROGRAM, THE
C             LOGARITHM OF THE FREQUENCY IS PLACED ON THE PREPARE
C             FILE
C
C     IF IDIV.NE.0 AND LOG IS TRUE THE REMAINING VARIABLES ARE CHANGED
C     INTO DECIBELS
C
      DIMENSION A(AMAX)
      INTEGER AMAX
      LOGICAL LOG
      MPM1=MP-1
      L=LO2*2
      LO2P1=LO2+1
      IFO2P1=IFO2+1
```

```
      IF(LO2P1.GT.IFO2P1) IFO2P1 = LO2P1
      LP2=L+2
      KBUFF=512/MP
      N1=1
      N2=N1+KBUFF*MP
      N3=N2+MPM1
      N4=N3+MPM1
      N5=N4+LP2*MPM1
      N61=N5+LP2*MPM1
      N71=N61+LP2*MPM1
      N62=N5+IFO2P1*MPM1
      N72=N62+IFO2P1+1
      N7=AMAX0(N71,N72)
      IF(N7.GT.AMAX) GO TO 100
      CALL PDS(A(N1),MP,KBUFF,A(N4),A(N5),A(N61),MPM1,DELTT,
     1         A(N2),A(N3),A(N5),LP2,IWIND,LO2P1,IFO2P1,IDIV,
     2         A(N4),A(N5),A(N61),A(N62),LOG)
100   CALL ZZPAGE(1,Z0)
      CALL ZZLINE( Z0)
      CALL ZZLINE(4)
      WRITE(6,200) N7
200   FORMAT(///10X,22HSTORAGE NEEDED EQUALS       I8)
      IF(N7.GT.AMAX) STOP 333
      CALL ZZLINE(3)
      WRITE(6,300)
300   FORMAT(//10X,8HVARIABLE,5X,4HMEAN,11X,8HVARIANCE    )
400   FORMAT(10X,I3,5X,G15.4,G15.4)
      DO 500 I=1,MPM1
      CALL ZZLINE(1)
      WRITE(6,400) I,A(N2+I-1),A(N3+I-1)
500   CONTINUE
      RETURN
      END
      SUBROUTINE PDS(BUFF,MP,KBUFF,A,X1,X1P1,MPM1,DELTT,
     1         XM,X2M,FREK,LP2,IWX,LO2P1,IFO2P1,IDIV,
     2         RA,RX1,RX1P1,FREKX,LOG)
      DIMENSION BUFF(MP,KBUFF),X1(LO2P1,MPM1),X1P1(LO2P1,MPM1),
     1         XM(MPM1),X2M(MPM1),FREK(IFO2P1,MPM1),A(LO2P1,MPM1),
     2         RX1(LP2,MPM1),RX1P1(LP2,MPM1),RA(LP2,MPM1),FREKX(1)
      COMPLEX X1,X1P1,A
      LOGICAL LOG
C
C             INITIALIZE VARIABLES
C
      DO 9 I=1,MPM1
      XM(I)=0.
      X2M(I)=0.
9     CONTINUE
      L=LP2-2
      LO2=LO2P1-1
      N=0
      IERR=UNIT(8)
```

```
      REWIND 8
15    CONTINUE
C
C             CALCULATE MEAN AND VARIANCE
C
      CALL INOUT(BUFF,MP,KBUFF,1,8,IERR)
      IF(IERR.EQ.0) GO TO 10
      K=LENGTH(8)/MP
      DO 11 J=1,MPM1
      DO 11 I=1,K
      XM(J)=XM(J)+BUFF(J+1,I)
      X2M(J)=X2M(J)+BUFF(J+1,I)**2
11    CONTINUE
      N=N+K
      GO TO 15
10    CONTINUE
      DO 12 J=1,MPM1
      XM(J)=XM(J)/N
      X2M(J)=X2M(J)/N-XM(J)**2
12    CONTINUE
      IERR=UNIT(8)
      REWIND 8
C
C             K IS THE NUMBER OF RECORDS OF LENGTH L/2 ON FILE 8
C
      K=N/LO2
      N=0
C
C             LOAD X1 AND SET INITIAL A=0
C
25    CONTINUE
      DO 30 I=1,LP2
      IF(I.GT.LO2) GO TO 35
      IF(N.NE.0) GO TO 40
      CALL INOUT(BUFF,MP,KBUFF,1,8,IERR)
      IF(IERR.EQ.0) GO TO 100
      N=LENGTH(8)/MP
      NI=0
40    CONTINUE
      NI=NI+1
      DO 31 J=1,MPM1
      RX1(I,J)=BUFF(J+1,NI)-XM(J)
      RA(I,J)=0.
31    CONTINUE
      IF(NI.EQ.N) N=0
      GO TO 30
35    CONTINUE
C
C             SET RX1(I)=0 FOR I L/2
C
      DO 32 J=1,MPM1
      RX1(I,J)=0.
```

```
      RA(I,J)=0.
32    CONTINUE
30    CONTINUE
      DO 36 I=1,MPM1
      CALL FFA(X1(1,I),L)
36    CONTINUE
C
C             START ALGORITHM
C
      DO 50 I=1,K
      IF(I.EQ.K) GO TO 55
C
C             READ X1P1
C
      DO 51 II=1,L
      IF(II.GT.LO2) GO TO 52
      IF(N.NE.0) GO TO 53
      CALL INOUT(BUFF,MP,KBUFF,1,8,IERR)
      IF(IERR.EQ.0) GO TO 100
      N=LENGTH(8)/MP
      NI=0
53    CONTINUE
      NI=NI+1
      DO 54 J=1,MPM1
      RX1P1(II,J)=BUFF(J+1,NI)-XM(J)
54    CONTINUE
      IF(NI.EQ.N) N=0
      GO TO 51
52    CONTINUE
      DO 61 J=1,MPM1
      RX1P1(II,J)=0.
61    CONTINUE
51    CONTINUE
C
C             CALCULATE FFT
C
      DO 56 J=1,MPM1
      CALL FFA(X1P1(1,J),L)
56    CONTINUE
      GO TO 57
C
C             X1P1(K)=0.
C
55    CONTINUE
      DO 58 J=1,MPM1
      DO 58 II=1,LP2
      RX1P1(II,J)=0.
58    CONTINUE
57    CONTINUE
C
C             CALCULATE A
C
```

```
      DO 60 J=1,MPM1
      DO 60 II=1,LO2P1
      A(II,J)=A(II,J)+X1(II,J)*(CONJG(X1(II,J))+(-1)**(II-1)
     1        *CONJG(X1P1(II,J)))
60    CONTINUE
C
C             X1=X1P1 AND REPEAT MAJOR LOOP
C
      DO 50 J=1,MPM1
      DO 50 II=1,LO2P1
      X1(II,J)=X1P1(II,J)
50    CONTINUE
C
C             SHIFT REAL PART OF A INTO SINGLE REAL ARRAY
C
      DO 82 J=1,MPM1
      DO 82 I=2,LO2P1
      RA(I,J)=REAL(A(I,J))
82    CONTINUE
C
C             TAKE INVERSE FFT OF A
C
      DO 80 J=1,MPM1
      CALL IFTSYM(RA(1,J),L,X1)
80    CONTINUE
C
C             DIVIDE A BY SIMGA**2 AND K*L/2
C             RA IS NOW THE ESTIMATE OF THE AUTOCOVARIANCE FUNCTION
C
      XN=K*LO2
      DO 90 K=1,MPM1
      DO 90 I=1,LO2P1
      RA(I,K)=RA(I,K)/(XN*X2M(K))
90    CONTINUE
C
C             APPLY WINDOW
C
      IFREK=(IFO2P1-1)*2
      CALL WINDOW(RA,LP2,MPM1,IWX,FREK,IFO2P1)
C
C             TAKE FFT TO GET PDS
C
      DO 91 J=1,MPM1
      CALL FFTSYM(FREK(1,J),IFREK,FREKX)
91    CONTINUE
C
C             OUTPUT PDS AND SCALE
C
      IERR=UNIT(8)
      REWIND 8
      DF=1./(DELTT*IFREK)
      XN=DELTT/(8.*ATAN(1.))
```

```
      F=DF/2.
      J=0
      DO 70 I=1,IFO2P1
      J=J+1
      BUFF(1,J)=F
      IF(LOG) BUFF(1,J)=ALOG10(F)
      F=F+DF
      DO 71 K=1,MPM1
      KP1=K+1
      BUFF(KP1,J)=SQRT(ABS(FREK(I,K)*XN*X2M(K)))
C
C             IF IDIV=N DIVIDE F(N,J) INTO F(I,J)
C             FOR I=1 TO MPM1, I.NE.N
C
      IF(IDIV.EQ.K.OR.IDIV.EQ.0) GO TO 71
      BUFF(KP1,J)=BUFF(KP1,J)/BUFF(IDIV,J)
      IF(LOG) BUFF(KP1,J)=20.*ALOG10(BUFF(KP1,J))
71    CONTINUE
      IF(J.NE.KBUFF) GO TO 70
      J=0
      CALL INOUT(BUFF,MP,KBUFF,0,8,IERR)
70    CONTINUE
      IF(J.NE.KBUFF.AND.J.NE.0) CALL INOUT(BUFF,MP,J,0,8,IERR)
      ENDFILE 8
      RETURN
100   CONTINUE
      PRINT 101
101   FORMAT(10X,20HBUFFER ERROR                )
      RETURN
      END
      SUBROUTINE WINDOW(A,LP2,MPM1,IWX,FREK,IFO2P1)
      DIMENSION A(LP2,MPM1),FREK(IFO2P1,MPM1)
      REAL HANN,HAMM
      BART(I)=FLOAT(LO2P1-I+1)/XLO2P1
      HANN(I)=.5*(1.+COS(3.14159*FLOAT(I-1)/XLO2P1))
      HAMM(I)=.54+.46*COS(3.14159*FLOAT(I-1)/XLO2P1)
      BLACK(I)=.42+.5*COS(3.14159*FLOAT(I-1)/XLO2P1)
     1       +.08*COS(6.28319*FLOAT(I-1)/XLO2P1)
C
C              SET FREK=0
C
      LO2P1=LP2/2
      XLO2P1=FLOAT(LO2P1)
      DO 11 J=1,MPM1
      DO 11 I=1,IFO2P1
      FREK(I,J)=0.
11    CONTINUE
      DO 12 I=1,LO2P1
      AMP=1.
      IF(IWX.EQ.2) AMP=BART(I)
      IF(IWX.EQ.3) AMP=HANN(I)
      IF(IWX.EQ.4) AMP=HAMM(I)
```

```
      IF(IWX.EQ.5) AMP=BLACK(I)
      DO 12 J=1,MPM1
      FREK(I,J)=A(I,J)*AMP
12    CONTINUE
      RETURN
      END
      SUBROUTINE INOUT(BUFF,N,M,IN,UN,IERR)
C
C
C             BUFF      INPUT/OUTPUT ARRAY
C
C             IN        =1,  READ DATA
C                       OTHERWISE, WRITE DATA
C
C             UN        UNIT NUMBER
C
C             IERR      NEGATIVE IF NO DATA ERROR
C                       ZERO IF EOF
C
C
      DIMENSION BUFF(512)
      INTEGER UN
      K=N*M
      IF(K.GT.512) STOP 222
      IF(IN.EQ.1) GO TO 1
C             WRITE DATA
      BUFFER OUT (UN,0) (BUFF(1),BUFF(K))
      GO TO 2
1     CONTINUE
C             READ DATA
      BUFFER IN  (UN,0) (BUFF(1),BUFF(K))
2     IERR=UNIT(UN)
      IF(IERR.GT.0) STOP 111
      RETURN
      END
```

APPENDIX B

ACSL TRANSLATOR INPUT

```
PROGRAM VAN DER POL'S EQUATION
    "----------PROGRAM CONSTANTS"
    CONSTANT LA       = 1.0       , TSTOP    = 50.0        ...
            ,PI       = 3.14159
    "----------STATE INITIAL CONDITIONS"
    CONSTANT XIC      =-1.5340    , XDIC     = .76552
    "----------DEFINE COMMUNICATIONS INTERVAL"
    CINTERVAL   CINT  =  0.063
    "----------DEFINE CONSTANTS FOR PDS"
    ARRAY A(10000)
    INTEGER AMAX,MP,LO2,WINDOW,FREQCY,DIV
    LOGICAL LOG,SKIPP,SKIPT
    CONSTANT AMAX     = 10000     , MP       = 3           ...
            ,LO2      = 512       , WINDOW   = 5           ...
            ,FREQCY   = 1024      , DIV      = 0           ...
            ,W        = 0.        , PDSX     = 0.          ...
            ,PDSXD    = 0.        , LOG      = .FALSE.     ...
            ,SKIPP    = .TRUE.    , SKIPT    = .FALSE.
INITIAL
    IF( SKIPT ) GO TO TERM
END$ "OF INITIAL"
DYNAMIC
DERIVATIVE
    "----------VAN DER POL'S EQUATION"
    XD         = INTEG(LA*(1.0-X**2)*XD-X,XDIC)
    X          = INTEG(XD,XIC)
END$ "OF DERIVATIVE SECTION"
    TERMT(T.GT.TSTOP)
END$ "OF DYNAMIC SECTION"
TERMINAL
TERM..CONTINUE
    IF( .NOT. SKIPP ) ...
    CALL PDSMAIN(A,MP,CINT,LO2,WINDOW,FREQCY,AMAX,DIV,LOG)
END$ "OF TERMINAL"
END$ "OF PROGRAM"
```

APPENDIX C

ACSL RUN-TIME EXECUTIVE INPUT

```
PREPAR T,X,XD
SET CALPLT=.T.
START
PLOT "XHI"=TSTOP,X,"LO"=-5.0,"HI"=5.0
PLOT "XAXIS"=X,"XLO"=-5.0,"XHI"=5.0,  ...
     XD,"HI"=5.0,"LO"=-5.0
SET SKIPT=.T.,SKIPP=.F.
START
PREPAR "CLEAR",W,PDSX,PDSXD
PLOT "XAXIS"=W,"XLO"=0.0,"XHI"=0.72,  ...
     PDSX,"HI"=3.0,"LO"=0.
END
```

REFERENCES

1. Advanced Continuous Simulation Language (ACSL) — User Guide/Reference Manual. Mitchell and Gauthier, Assoc., Inc., c.1975.

2. Mitchell, Edward E. L.; and Gauthier, Joseph S.: Advanced Continuous Simulation Language (ACSL). Simulation, vol. 26, no. 3, Mar. 1976, pp. 72-78.

3. Oppenheim, Alan V.; and Schafer, Ronald W.: Digital Signal Processing. Prentice-Hall, Inc., c.1975, pp. 556-562.

4. Programs for Digital Signal Processing. IEEE Press, c.1979.

5. Rogers, A. E.; and Connolly, T. W.: Analog Computation in Engineering Design. McGraw-Hill Book Co., Inc., 1960, pp. 146-149.

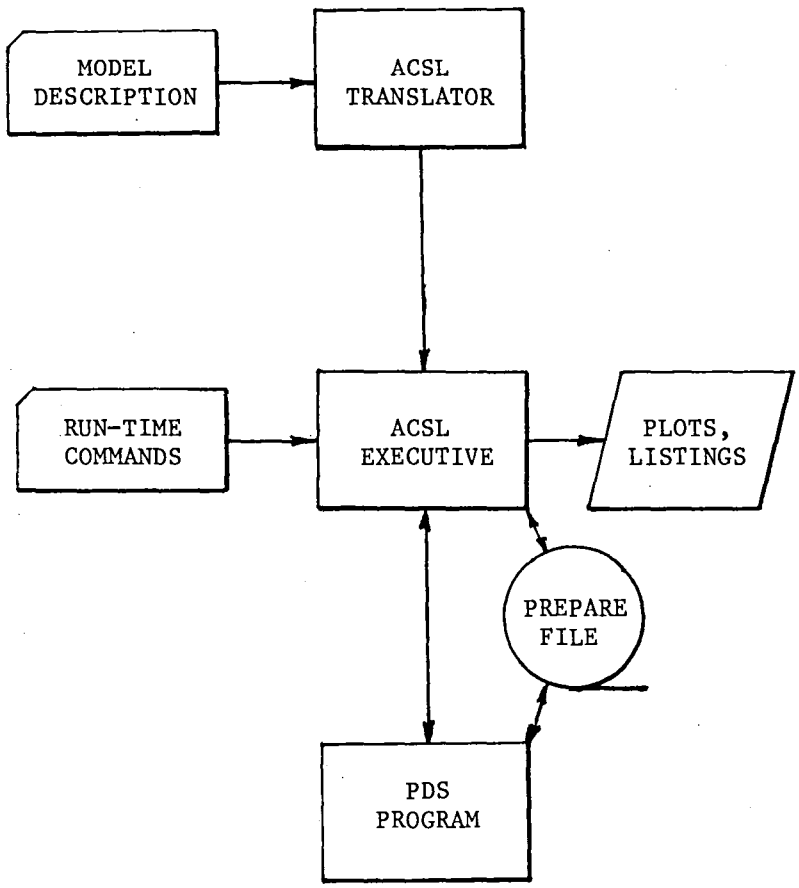

Figure 1.- ACSL and PDS program interface.

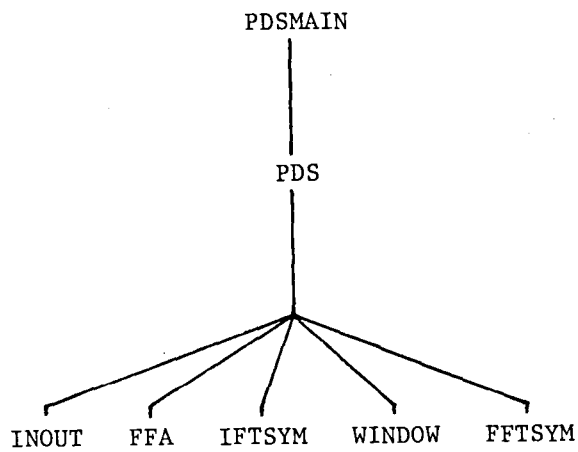

Figure 2.- PDS program structure.

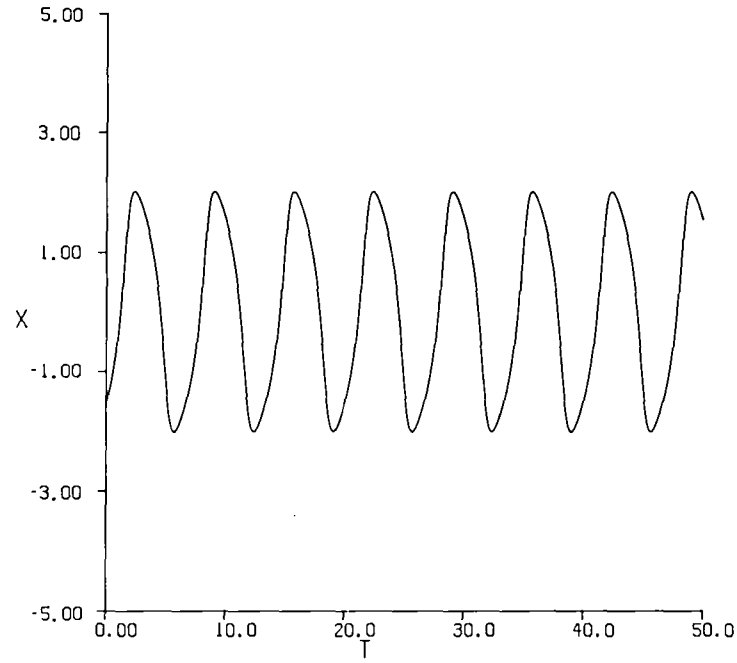

Figure 3.- ACSL plot of Van der Pol oscillator output.

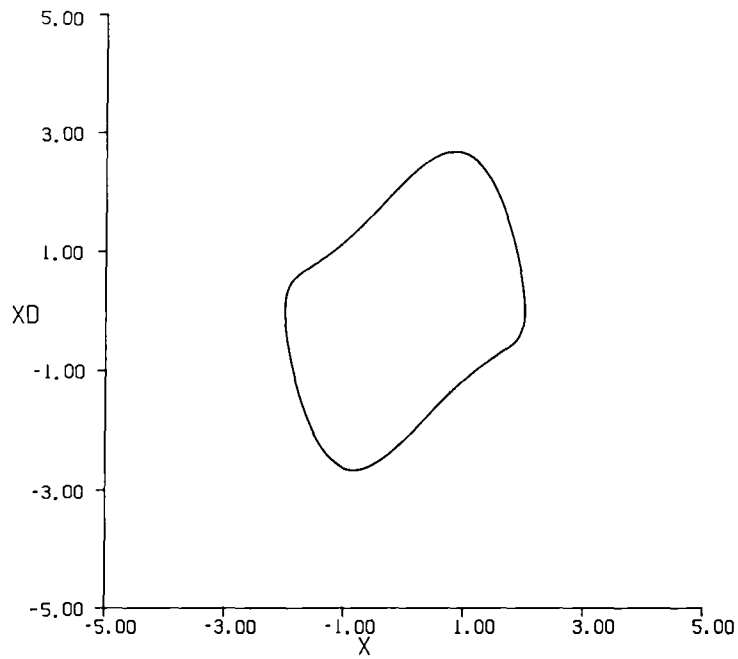

Figure 4.- ACSL plot of phase plane for oscillator.

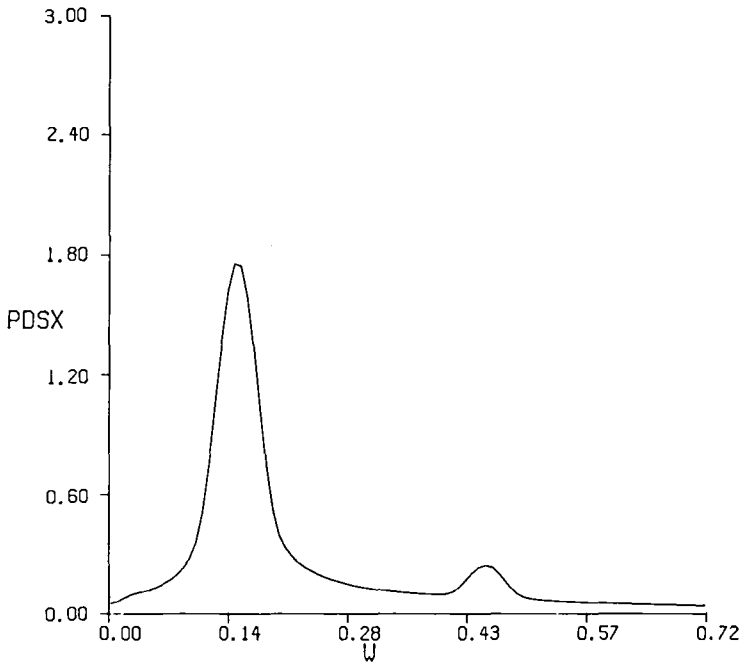

Figure 5.- PDS of Van der Pol oscillator.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-83120		

4. Title and Subtitle	5. Report Date
A COMPUTER PROGRAM FOR ESTIMATING THE POWER-DENSITY SPECTRUM OF ADVANCED CONTINUOUS SIMULATION LANGUAGE GENERATED TIME HISTORIES	June 1981
	6. Performing Organization Code: 505-34-33-05
7. Author(s): H. J. Dunn	8. Performing Organization Report No.: L-14217
	10. Work Unit No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A computer program for performing frequency analysis of time-history data is presented. The program uses circular convolution and the fast Fourier transform to calculate power-density spectrum (PDS) of time-history data. The program interfaces with the Advanced Continuous Simulation Language (ACSL) so that a frequency analysis may be performed on ACSL generated simulation variables. An example of the calculation of the PDS of a Van der Pol oscillator is presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Power-density spectrum Fast Fourier transform Advanced continuous simulation language	Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	24	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1981

End of Document