NASA TECHNICAL MEMORANDUM

NASA TM X-71526

NASA TM X-71526

(NASA-TM-X-71526) FUNCTION GENERATION SUBPROGRAMS FOR USE IN DIGITAL SIMULATIONS (NASA) 15 p HC $4.00

CSCL 09B

G3/08

N74-19838

Unclas

34362

FUNCTION GENERATION SUBPROGRAMS FOR USE IN DIGITAL SIMULATIONS

by Clint E. Hart
Lewis Research Center
Cleveland, Ohio 44135

FUNCTION GENERATION SUBPROGRAMS FOR USE IN DIGITAL SIMULATIONS

By Clint E. Hart

ABSTRACT

Convenient and efficient function generation subprograms have been developed for handling functions of one variable and two types of functions of two variables. These subprograms can easily be used in any digital or hybrid simulation requiring function generation. Use of these programs can often lower overall program execution time.

FUNCTION GENERATION SUBPROGRAMS FOR USE IN DIGITAL SIMULATIONS

By Clint E. Hart

Lewis Research Center

E-7926

SUMMARY

Dynamic models of propulsion systems often contain component performance data in the form of functions of one and two variables. Digital and hybrid simulation of these dynamic models require subprograms to handle function generation. Convenient and efficient function generation subprograms have been developed for handling functions of one variable and two types of functions of two variables. Use of these subprograms in digital or hybrid simulations can often lower the overall program execution time.

INTRODUCTION

Simulation of dynamic models of turbojet or tubofan engines has proved valuable in analyzing their dynamic behavior and in designing engine control systems. Often the model equations contain several functions of one and two variables. The two variable functions are difficult to program accurately for analog computer simulation and require large amounts of analog computer equipment. For these reasons most engine simulations have been programmed for digital computers.

The most commonly used programming language for these engine simulations is FORTRAN. However, some simulations have been programmed in either CSMP or one of the special languages developed

for hybrid computers. When CSMP first became available it had no provision for handling a function of two variables. Also, at that time no suitable function generation subprograms were being retained in our computer system library. Thus, to use CSMP for engine simulations, two variable function generation subprograms had to be developed.

For FORTRAN engine simulations the programmers usually wrote their own subprograms to handle functions of one and two variables. Examination of many of these subprograms showed that they were much slower in execution time than desirable. Fast execution times for these subprograms is essential because in some dynamic engine simulations the function generation subprograms may be called more than 10^5 times per second of problem time.

Therefore, a programming effort was made to develop convenient and efficient one and two variable function generation subprograms.

The following sections will contain discussion of the types of functions usually encountered and descriptions of the function generation subprograms. The application of these function generation subprograms with CSMP or other digital simulation programs will be explained.

PROGRAM DEVELOPMENT

A typical function frequently encountered in engine simulations is a function of one variable $Z = f(X)$, as shown in figure 1. For

the simulation, a set of discrete points is selected to represent this function. Usually linear interpolation is used between points, but higher order interpolation can be used if desired. The X-input values are often called breakpoints when using linear interpolation.

There are two types of functions of two variables that may occur in engine simulations. A common type of function is shown in figure 2. For this type the range of the family of curves is over the same set of values of the X-input variable. Usually, two dimensional linear interpolation is made in directions parallel to the X-input and Z-output axes.

A second type of function of two variables is shown in figure 3. This type, often called a map, differs from the first type in that the range of each curve is over a different set of values of the X-input variable. The two-dimensional linear interpolation mentioned above cannot be used for this map-type function of two variables. Significant errors can occur with this interpolation method. A special method should be used with interpolation in directions not parallel to either the X-input or Z-output axes. An interpolation method of this type will be discussed in a later section.

Description of Subprograms

Consider first a subprogram to handle a function of two variables as shown in figure 2. The data which describes the function consists of lists or tables of X breakpoints and Y curve values and the corresponding Z output values. For this type of function the X

breakpoint values are common for all curves. The X and Y values are listed in order of monotonically increasing value. The Z values are listed in order corresponding to the X values for the first curve (lowest Y value), second curve, etc.

Typically, a comparison table search method is used in function generation subprograms. The X and Y inputs are compared with table values until the proper pairs of table values are found which bracket the inputs. Usually the table search starts at either the low or high end of the table and moves either up or down.

The table search method can be made more efficient, i.e., faster, by using table-search indices. These search indices are used as starting points in the table searches of the X and Y data tables. First, the X and Y inputs are tested to see if they are in the same intervals between pairs of X or Y values that they were in for the preceding entry. If not, the search indices are incremented either up or down until the proper intervals are found. The final values of the search indices are stored for use at the next entry. After the proper table values are determined a two-dimensional linear interpolation is made to obtain the output.

The subprogram for a function of one variable is similar to the one described above. Since there is only one curve, no Y values are required, and only one set of Z values. A table search index and one dimensional linear interpolation are also used.

The subprogram for a map-type function of two variables requires a set of X values for each curve. The special interpolation method, which will be described later, requires that the same number of breakpoints must be used to define each curve. The X values are listed in order of monotonically increasing value for the first curve (lowest Y value), second curve, etc. The Z values are listed in order corresponding to the X values. Additional data required for all of the function generation programs are the number of points per curve and in the case of the two-variable subprograms, the number of curves.

Part of the map-type function of two variables of figure 3 is also shown in figure 4. Extra lines and points have been added to explain the interpolation method which is used in the subprogram.

Consider the point A whose location is determined by inputs X_A and Y_A. First, the input Y_A is compared with successive values in the Y table to determine which pair of curves it is between. In this example, it is between curves having values of Y(2) and Y(3). Adjacent pairs of similarly located breakpoints on these curves define quadrilaterals. Through special logic in the subprogram the quadrilateral containing point A can be determined. In this example it is defined by points B, C, D, and E. Knowing the Z and X coordinates of these points, the Z and X coordinates of points F and G can be found by linear interpolation in Y along

lines BC and ED. Then the Z output ZA can be found by linear interpolation in X along line FG.

Another problem which must be considered in developing these subprograms is: what to do when one of the inputs is beyond the range of the table data. If an input is out of range of the tabulated values for the function, the output is calculated by extrapolating beyond the end values of the tables. An error message is printed, but only the first time an input goes out of range. The decision whether or not to extrapolate or print an error message is arbitrary and the subprograms can easily be modified to fit needs of a particular simulation.

FORTRAN listings for two subprograms, FUN2 and MAPFUN, to handle both types of functions of two variables are presented in Appendix A. Also in Appendix A is a listing of a subprogram FUN1 to handle a function of one variable.

Instructions for Using Subprograms

The function data should be stored in one-dimensional arrays. Separate arrays should be used for each set of X input, Y input, and Z output data. For FORTRAN simulations the BLOCK DATA subprogram is convenient to use for entering the function data. Duplicate labelled COMMON statements must be used in the main simulation program and the BLOCK DATA subprogram. For CSMP simulations the TABLE data input option is used to enter the function data.

In addition to arrays for function data, two arrays are required for the search indices. One of these arrays is used to store the X-input search indices and one to store the Y-input search indices for all the functions. Also, an array is needed to store an error indicator which controls printing an error message when an input is out of range of the function data. For FORTRAN simulations these arrays can be initialized by DATA statements. For CSMP simulations a subprogram FUNSET can be called in the **INITIAL** section to initialize these arrays. A listing of this program is presented in Appendix A.

The calling statement for a function of one variable is of the form:

ZOUT = FUN1(N,NXP,X1,Z1,XIN)

N is an integer used to select the table search index for each call. NXP is the number of points per curve. X1 and Z1 are arrays containing the function data. XIN is the input variable and ZOUT is the output variable.

For the regular type function of two variables the calling statement is of the form:

ZOUT = FUN2(N,NXP,NYC,XX1,ZZ1,XIN,YIN).

ZOUT,N,NXP, and XIN are the same as described in the preceding paragraph. NYC is the number of curves. XX1,YY1, and ZZ1 are arrays containing the function data and YIN is the second input variable.

The calling statement for the map-type function of two variables is of the form:

ZOUT = MAPFUN(N,NXP,NYC,XM1,YM1,ZM1,XIN,YIN).

XM1, YM1, ZM1 are arrays containing the function data. All the other arguments are the same as described for the regular function of two variables.

CONCLUDING REMARKS

For digital simulations requiring function generation, considerable savings in execution time can be made by using the subprograms discussed in this memorandum. Timing tests indicate these subprograms run at least 80% faster than comparable subprograms used in current digital engine simulations. These subprograms can be easily incorporated in digital simulation programs.

```
      FUNCTION FUN1(N,NXP,XX,ZZ,XIN)
      COMMON /FMEMR/ IX(40),JY(40),IERR(40)
      DIMENSION XX(NXP),ZZ(NXP)
      I = IX(N)
C*****TEST FOR X IN PREVIOUS INTERVAL*****
      IF(XIN-XX(I)) 120,200,110
  110 IF(XIN-XX(I+1)) 200,140,140
C*****COUNT DOWN*****
  120 IF(XIN-XX(1)) 160,160,130
  130 I = I-1
      IF(XIN-XX(I)) 130,200,200
C*****COUNT UP*****
  140 IF(XIN-XX(NXP)) 150,170,170
  150 I = I+1
      IF(XIN-XX(I+1)) 200,200,150
  160 I = 1
      GO TO 180
  170 I = NXP-1
  180 IF(IERR(N)) 200,190,190
  190 WRITE(6,400) N,XIN
      IERR(N) = -1
C*****INTERPOLATE FOR ANSWER*****
  200 XFRAC = (XIN-XX(I))/(XX(I+1)-XX(I))
      FUN1 = ZZ(I)+XFRAC*(ZZ(I+1)-ZZ(I))
      IX(N) = I
      RETURN
  400 FORMAT(1H0,12HFUNCTION NO.,I3,20H  INPUT OUT OF RANGE,
     12X,6HXIN = ,G12.4)
      END
```

```
      FUNCTION FUN2(N,NXP,NYC,XX,YY,ZZ,XIN,YIN)
      COMMON /FMEMR/ IX(40),JY(40),IERR(40)
      DIMENSION XX(NXP),YY(NYC),ZZ(NXP,NYC)
      I = IX(N)
      J = JY(N)
C*****TEST FOR X IN PREVIOUS INTERVAL*****
      IF(XIN-XX(I)) 120,200,110
  110 IF(XIN-XX(I+1)) 200,140,140
C*****COUNT DOWN*****
  120 IF(XIN-XX(1)) 160,160,130
  130 I = I-1
      IF(XIN-XX(I)) 130,200,200
C*****COUNT UP*****
  140 IF(XIN-XX(NXP)) 150,170,170
  150 I = I+1
      IF(XIN-XX(I+1)) 200,200,150
  160 I = 1
      GO TO 180
  170 I = NXP-1
```

```
  180 IF(IERR(N)) 200,190,190
  190 WRITE(6,400) N,XIN,YIN
      IERR(N) = -1
C*****TEST FOR Y IN PREVIOUS INTERVAL*****
  200 IF(YIN-YY(J)) 220,300,210
  210 IF(YIN-YY(J+1)) 300,240,240
C*****COUNT DOWN*****
  220 IF(YIN-YY(1)) 260,260,230
  230 J = J-1
      IF(YIN-YY(J)) 230,300,300
C*****COUNT UP*****
  240 IF(YIN-YY(NYC)) 250,270,270
  250 J = J+1
      IF(YIN-YY(J+1)) 300,300,250
  260 J = 1
      GO TO 280
  270 J = NYC-1
  280 IF(IERR(N)) 300,290,290
  290 WRITE(6,400) N,XIN,YIN
      IERR(N) = -1
C*****INTERPOLATE FOR ANSWER*****
  300 XFRAC = (XIN-XX(I))/(XX(I+1)-XX(I))
      P1ZZ = ZZ(I,J)+XFRAC*(ZZ(I+1,J)-ZZ(I,J))
      P2ZZ = ZZ(I,J+1)+XFRAC*(ZZ(I+1,J+1)-ZZ(I,J+1))
      YFRAC = (YIN-YY(J))/(YY(J+1)-YY(J))
      FUN2 = P1ZZ+YFRAC*(P2ZZ-P1ZZ)
      IX(N) = I
      JY(N) = J
      RETURN
  400 FORMAT(1H0,12HFUNCTION NO.,I3,20H INPUTS OUT OF RANGE,
     12X,6HXIN = ,G12.4,2X,6HYIN = ,G12.4)
      END
```

—

```
      REAL FUNCTION MAPFUN(N,NXP,NYC,XVALS,YVALS,ZVALS,XIN,YIN)
      COMMON /FMEMR/ IX(40),JY(40),IERR(40)
      DIMENSION XVALS(NXP,NYC),YVALS(NYC),ZVALS(NXP,NYC)
      I = IX(N)
      J = JY(N)
C*****TEST FOR Y IN PREVIOUS INTERVAL*****
      IF(YIN-YVALS(J)) 120,200,110
  110 IF(YIN-YVALS(J+1)) 200,140,140
C*****COUNT DOWN*****
  120 IF(YIN-YVALS(1)) 160,160,130
  130 J = J-1
      IF(YIN-YVALS(J)) 130,200,200
C*****COUNT UP*****
  140 IF(YIN-YVALS(NYC)) 150,170,170
  150 J = J+1
      IF(YIN-YVALS(J+1)) 200,150,150
  160 J = 1
      GO TO 180
  170 J = NYC-1
```

—

```
  180 IF(IERR(N))200,190,190
  190 WRITE(6,400)N,XIN,YIN
      IERR(N) = -1
C*****CALCULATE XLO AND XHI*****
  200 YFRAC = (YIN-YVALS(J))/(YVALS(J+1)-YVALS(J))
      XLO = XVALS(I,J)+YFRAC*(XVALS(I,J+1)-XVALS(I,J))
      XHI = XVALS(I+1,J)+YFRAC*(XVALS(I+1,J+1)-XVALS(I+1,J))
C*****TEST FOR X BETWEEN XLO AND XHI*****
      IF(XIN-XLO) 220,300,210
  210 IF(XIN-XHI) 300,240,240
C*****COUNT DOWN*****
  220 XMIN = XVALS(1,J)+YFRAC*(XVALS(1,J+1)-XVALS(1,J))
      IF(XIN-XMIN) 260,260,230
  230 I = I-1
      XHI = XLO
      XLO = XVALS(I,J)+YFRAC*(XVALS(I,J+1)-XVALS(I,J))
      IF(XIN-XLO) 230,300,300
C*****COUNT UP*****
  240 XMAX = XVALS(NXP,J)+YFRAC*(XVALS(NXP,J+1)-XVALS(NXP,J))
      IF(XIN-XMAX) 250,270,270
  250 I = I+1
      XLO = XHI
      XHI = XVALS(I+1,J)+YFRAC*(XVALS(I+1,J+1)-XVALS(I+1,J))
      IF(XIN-XHI) 300,250,250
  260 I = 1
      GO TO 280
  270 I = NXP-1
  280 IF(IERR(N))300,290,290
  290 WRITE(6,400)N,XIN,YIN
      IERR(N) = -1
C*****INTERPOLATE FOR ANSWER*****
  300 XFRAC = (XIN-XLO)/(XHI-XLO)
      ZL = ZVALS(I,J)+YFRAC*(ZVALS(I,J+1)-ZVALS(I,J))
      ZR = ZVALS(I+1,J)+YFRAC*(ZVALS(I+1,J+1)-ZVALS(I+1,J))
      MAPFUN = ZL+XFRAC*(ZR-ZL)
      IX(N) = I
      JY(N) = J
      RETURN
  400 FORMAT(1H0,8H MAP NO.,I3,20H INPUTS OUT OF RANGE,
     12X,6HXIN = ,G12.4,2X,6HYIN = ,G12.4)
      END
```

```
      FUNCTION FUNSET(N)
      COMMON /FMEMR/ IX(40),JY(40),IERR(40)
      DO 10 K=1,N
      IX(K) = 1
      JY(K) = 1
      IERR(K) = 1
   10 CONTINUE
      FUNSET = 1.0
      RETURN
      END
```

Figure 1 - Function of one variable

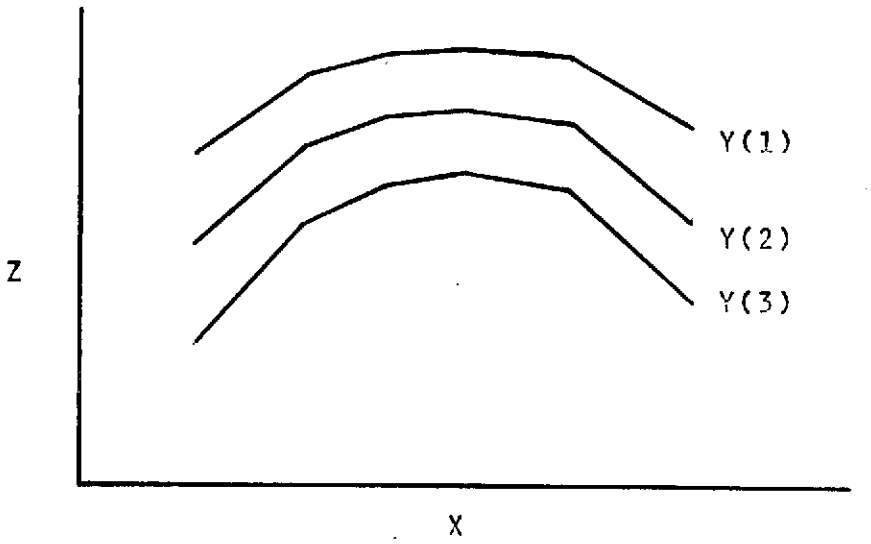

Figure 2 - Regular type function of two variables

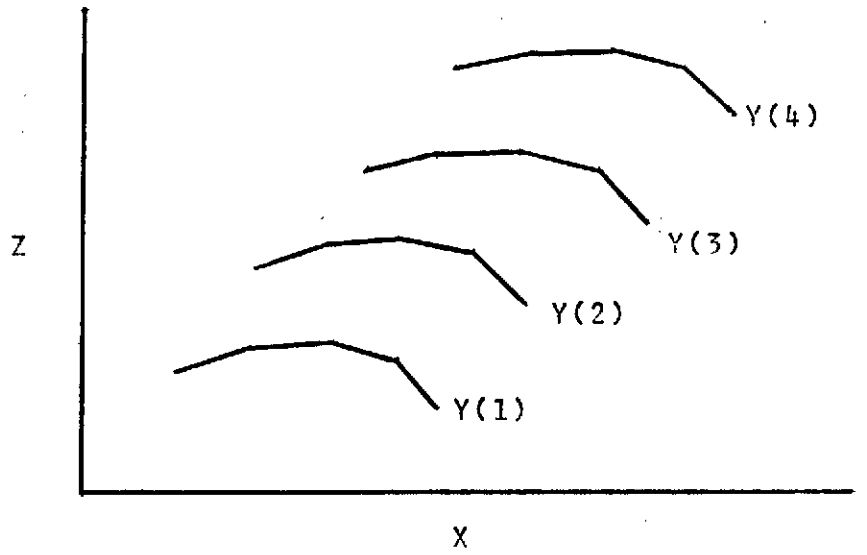

Figure 3 - Map-type function of two variables

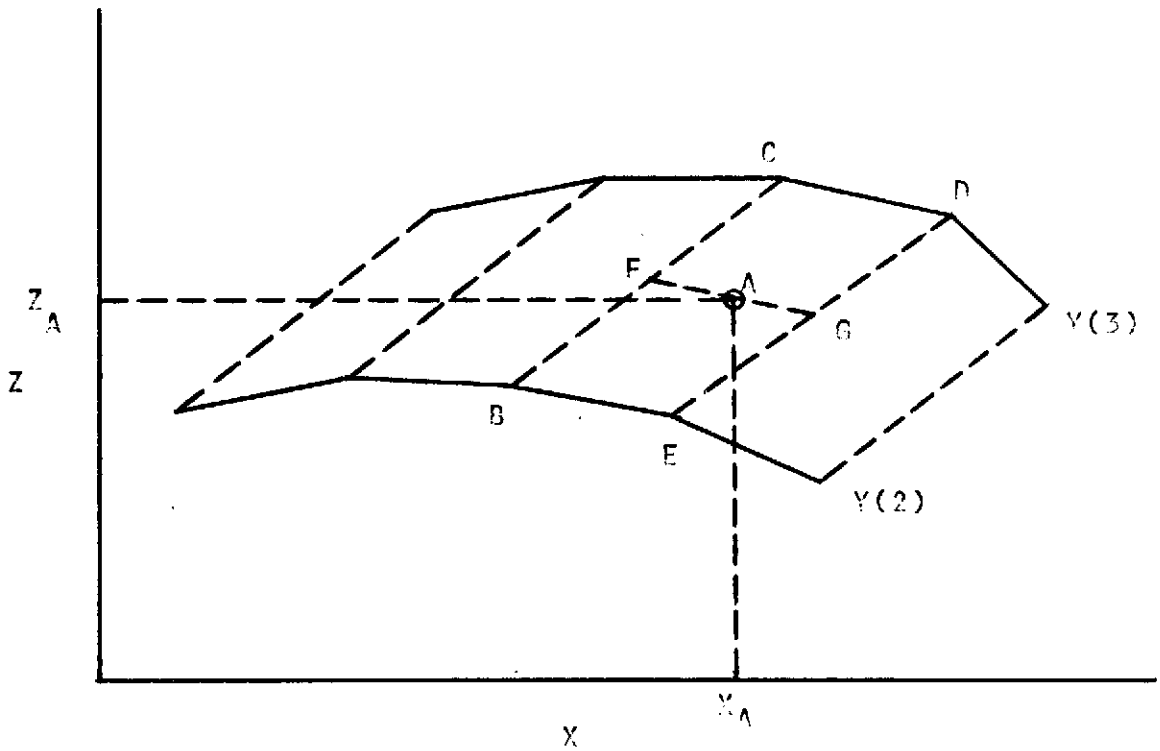

Figure 4 - Details of interpolation method for map-type function of two variables