Michigan State University

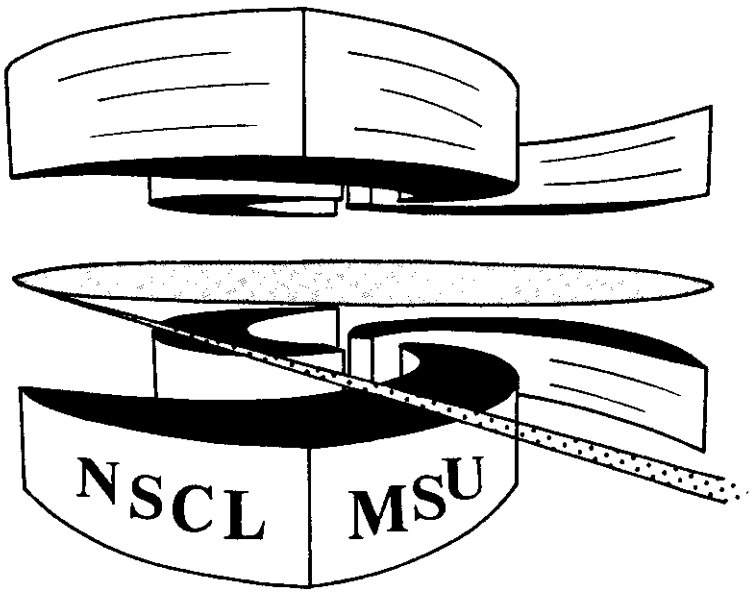
National Superconducting Cyclotron Laboratory

DAFOR

Differential Algebra Precompiler Version 3

Reference Manual

M. BERZ



MSUCL-755

JANUARY 1991

MSUCL-755
January, 1991

DAFOR

Differential Algebra Precompiler Version 3

Reference Manual

M. Berz

Department of Physics and Astronomy
and National Superconducting
Cyclotron Laboratory
Michigan State University
East Lansing, Mi 48824

Abstract

This manual describes the syntax and use of the DA precompiler DAFOR. DAFOR represents an extension to standard FORTRAN 77 which allows the use of a differential algebraic data type. DAFOR is designed to allow rapid conversion of existing programs. It converts the extended FORTRAN code to regular FORTRAN code by expressing all DA operation by calls to subroutines from the DA library DAPRE.

The library allows computation to arbitrary order and for arbitrarily many variables, limited only by computer memory and speed. After the conversion process, every DA variable is represented by an integer pointer to the address in DAPRE's memory management. This technique allows to pass DA variables into subroutines and functions and to put them into common blocks. It is also possible to construct DA-valued functions.

The elementary operations in the library are highly optimized. A sophisticated algorithm is used for multiplication, reducing the bookkeeping overhead to only about 30 % of the computation time necessary for the double precision operations. Furthermore, vanishing entries in the DA vectors do not contribute to computation time, an essential feature since in practice vectors are often very sparse.

1 Initial Preparation

The first step in the conversion of a FORTRAN code to Differential Algebraic computation of derivatives is to identify the independent variables with respect to which to differentiate as well as the number of these variables and the maximum order to which derivatives are to be computed.

If any of the independent variables occur in the main program, it is necessary to artificially turn the main program into a subroutine and generate a new main program which calls this subroutine. Before the subroutine call, the following statement has to be included:

CALL **DAINI**(<order>,< # of variables>)

where the arguments are the maximum occurring order and the number of variables. This call initializes the Differential Algebra package DAPRE.

2 Declarations

In each program segment, all variables that become DA have to be declared to the precompiler and their original FORTRAN declarations have to be removed. All declarations for the precompiler have to be located between the declaration and execution section of the FORTRAN program. All precompiler declarations have to be included between the precompiler commands

***DA B D ;**

which marks the beginning of the DAFOR declaration section, and

***DA E D ;**

which denotes the end of the declaration section. The beginning of the commands, *DA, entails that the commands are ignored by regular FORTRAN and processed by the precompiler.

DA variables occurring in the subroutine are declared with the command

***DA D V DA** <globality> <name> <NO> <NV> [Dim1] [Dim2] ... ;

Here <globality> has to be set to INT or EXT. INT is used for variables that are used for the first time in the momentary program section; EXT is used for variables that have been used in another program section before and are passed into the current section through the argument list of FUNCTION or SUBROUTINE or through COMMON blocks. <name> is the name of variables, and <NO> and <NV> are the order and number of variables of the variable to be declared, not to exceed the maximum order and variable number set with DAINI. [Dim1], Dim2], are up to seven dimensions. Note that at the moment, all arrays have to begin with 1 as their lower index. The command is terminated with a semicolon.

It is possible to have several of these commands in one line and to extend them over many lines. All continuation lines also have to begin with *DA in columns 1 through 6. Note that if the program segment is a DA-valued function, the name of the function has to be declared as a (dimensionless) DA variable.

Besides the DA variables, all REAL, INTEGER, or DOUBLE PRECISION variables occurring in arithmetic expressions in which DA variables occur have to be declared to the precompiler. This is done with the com-

mand

***DA D V** <type> <globality> <name> [Dim1] [Dim2] ... ;

Here <type> identifies the type of the variable. REAL or DOUBLE PRECISION is identified with RE, and INTEGER is identified with IN. The globality again has to be set to INT or EXT. [Dim1],... again are up to seven dimensions. In the case of these scalar variables, the original FORTRAN declaration must not be removed.

Besides the variables occurring in the program section, it is necessary to declare all external functions that occur in an expression that contains DA operations. This is done with the command

***DA D F** <type> <name> <number of arguments> ;

where <type> is the type of the function (RE, IN or DA), <name> is its name and the last entry specifies the number of arguments of the function. Intrinsic functions do not have to be declared; a list of all supported intrinsic functions can be found in section 5.

If the program section under consideration is a function which returns DA types, the name of the function has to be declared using the command

***DA D V** <type> FUN <NO> <NV> ;

Like in any language requiring explicit declaration of variables, it is very important that all DA variables are declared properly. Since FORTRAN does not require explicit declaration, the precompiler offers a facility to help identify all variables that have to be converted to DA. If the precompiler is activated, it flags any line that contains a declared DA variable by putting a comment *DA in columns 73 through 76 of the line. Since any dependent DA variable has to occur at least once together with a previously declared variable in one command, this technique allows to iteratively identify all dependent DA variables.

While the values of dependent DA variables are set automatically through arithmetic expressions, it is necessary to set the values of the independent variables manually. This can be achieved with the command

CALL DAVAR(<name> ,<value>, <number>)

Here <name> is the name of all the independent variables, <value>

a double precision number giving the value at which to differentiate, and <number> is the number of the independent variable. This number serves for identification purposes in the output where it will identify the specific derivative.

3 Assignment Statements

The main feature of the precompiler DAFOR is that it automatically converts assignment statements in which DA variables are computed in terms of others. Any assignment statement containing a DA variable has to be identified for precompilation with a *DA in columns 1 through 3. The statement can extend over several lines, which then also have to be marked with *DA in the first columns. The end of the command has to be denoted with a semicolon.

As pointed out in the previous section, it is important that the assigned variable is properly declared as DA, REAL, DOUBLE PRECISION or INTEGER.

DAFOR can process the full syntax of a FORTRAN assignment statement, including parentheses, arrays and functions. Functions can be either intrinsic or external.

4 Other Commands

FORTRAN commands other than assignments are not processed by DAFOR. In the case of outputting statements like WRITE or PRINT, it is necessary to rewrite the commands. In case it is desirable to output the value of the quantity along with its derivatives with respect to the independent variables, i.e. the full DA variable, the DA subroutine

CALL DAPRI(<name>,<unit>)

can be used. It prints the variable to the requested unit. In case the value has to printed without the derivatives, this value can be obtained using the DOUBLE PRECISION function

DARE(<name>)

which returns the real part of the DA vector.

In case of logical branching statements involving the size of the variable, the function DARE has to be used also. Note, however, that in many cases branching according to the value of a variable introduces a non-differentiable behavior at the branching point, which violates the basic assumption of differentiation of algorithms.

5 Supported Intrinsic Functions

The DA precompiler supports a large number of intrinsic functions occurring in standard FORTRAN. The arguments of the functions can either be real, integer, double precision, or DA. If the argument is DA, the type of the result will be DA, otherwise it will be like in standard FORTRAN. The functions supported by DAFOR and the number of arguments are:

Name	Arguments
ABS	1
ABS	1
ACOS	1
ACOSD	1
AINT	1
ALOG	1
ALOG10	1
AMAX	1
AMAX0	1
AMAX1	1
AMIN	1
AMIN0	1
AMIN1	1
AMOD	2
ANINT	1
ASIN	1
ASIND	1
ATAN	1
ATAN2	2
ATAND	1

Name	Arguments
COS	1
COSD	1
COSH	1
COSHD	1
DABS	1
DACOS	1
DACOSD	1
DASIN	1
DASIND	1
DATAN	1
DATAN2	1
DATAND	1
DBLE	1
DCOS	1
DCOSD	1
DCOSH	1
DDIM	2
DEXP	1
DIM	2
DINT	1
DLOG	1
DLOG10	1
DLOG2	1
DMAX1	1
DMIN1	1
DMOD	2
DNINT	1
DPROD	2
DSIGN	2
DSIN	1
DSIND	1
DSINH	1
DSQRT	1
DTAN	1
DTAND	1
DTANH	1
EXP	1

Name	Arguments
FLOAT	1
IABS	1
IDIM	2
IDINT	1
IDNINT	1
IFIX	1
INT	1
ISIGN	2
LOG	1
LOG10	1
LOG2	1
MAX	2
MAX0	2
MAX1	2
MIN	2
MIN0	2
MIN1	2
MOD	2
NINT	1
REAL	1
SIGN	2
SIN	1
SIND	1
SINH	1
SINHD	1
SNGL	1
SQR	1
SQRT	1
TAN	1
TAND	1
TANH	1
TANHD	1

6 Appendix 1: Sample Code Before Precompilation

In this section, we will provide an example of a FORTRAN using DA operations which can be processed using the precompiler DAFOR and which serves as a test for the differential algebra package DAPRE.

```
      PROGRAM TEST
*     ************
*
*     THIS PROGRAM TESTS THE DA PRECOMPILER AND PACKAGE. IT EVALUATES
*     SEVERAL ARITHMETIC DA EXPRESSIONS, ALL OF WHICH ARE
*     IDENTICAL TO ZERO. THE SUM OF ABSOLUTE VALUES OF THE RESULTING DA
*     VECTOR THUS HAS TO VANISH. THESE ABSOLUTE VALUES ARE OUTPUT TO THE
*     SCREEN; SHOULD THERE BE ANY QUANTITY THAT IS NOT ZERO (UP TO ROUNDOFF),
*     THERE IS AN ERROR IN THE DA PACKAGE OR PRECOMPILER.
*
*     THE FILE ALSO SERVES AS AN EXAMPLE FOR THE SYNTAX AND PRACTICAL USE
*     OF DA
*
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON NO,NV
*
  5   CONTINUE
*
      WRITE(*,*)' '
      WRITE(*,*)'     GIVE ORDER, NUMBER OF VARIABLES'
      WRITE(*,*)' '
      READ(*,*) NO,NV
*
      CALL DAINI(NO,NV,1)
*
      CALL ARITEST
*
      STOP
      END
*
      SUBROUTINE ARITEST
*     ******************
*
*     THIS SUBROUTINE PERFORMS SEVERAL ARITHMETICAL CHECKS; IF EVERYTHING
```

```
*     IS OK, THE RESULTING DA VECTOR SHOULD BE IDENTICAL TO ZERO.
*
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
*
      COMMON NO, NV
      DIMENSION R(15)
      INTEGER JJ(20)
      DATA JJ /20*0/
      INTEGER J
*
*DA   B D ; D V DA INT X NO NV ; D V DA INT Y NO NV ;
*DA   D V DA INT Z NO NV ; D V DA INT XYZ NO NV 7 4 ; D V RE INT R 15 ;
*DA   D V IN INT I ; D V RE INT RRR ; D V RE INT C ; D V IN INT J ;
*DA   D V IN INT ITEST ;
*DA   D F RE DARE 1 ; D F DA TTAN 1 ; D F RE FLOAT 1 ;
*DA   E D ;
*
      IRAN = 123454321

      WRITE(*,*) '     GIVE NUMBER OF TESTS'
      READ(*,*) NTEST
*
      DO 500 ITEST=1,NTEST
*
      DO 10 I=1,15
  10  R(I) = DABRAN(IRAN)
      I = 7
*
*     THE TWO FOLLOWING COMMAND FILL THE DA VECTORS X AND Y WITH
*     RANDOM ENTRIES; THEY ARE USUALLY NOT REQUIRED IN A PROGRAM
*
      CALL DARAN(X,1.D0)
      CALL DARAN(Y,.5D0)
*
      J = 1
*DA   XYZ(22-21,3/3) = X ;
*DA   XYZ(2,2*J) = Y ;
*
*
*DA   Z = (X+Y)**2-X**2.D0-2.D0*X*Y-Y*Y ;
*
*     THE ROUTINE DAABS COMPUTES THE SUM OF THE ABSOLUTE VALUES OF THE
*     COMPONENTS IN A DA VECTOR; IT IS ALSO USUALLY NOT REQUIRED IN A
*     PROGRAM
```

```
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  1: ',C
*
*DA   Z = (R(I)/66/X) * (X/R(I)*66) - 1.D0 + EXP(X) - EXP(1.D0)**X ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  2: ',C
*
*DA   Z = 1.D0 - SQR(X+Y) / (X+Y) / (X+Y) ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  3: ',C
*
*DA   Z = SIN(30.D0)*COS(40.D0)-2.D0 - SIN(30.D0)*COS(40.D0)+2.D0 ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  4: ',C
*
*DA   Z = X ;
*DA   Z = Z - X ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  5: ',C
*
*DA   Z = 9 - X - 9 + X + R(I)*X*R(I) - R(I)*R(I)*X ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  6: ',C
*
*DA   Z = SIN(X)*SIN(X) + SQR(COS(X)) - R(I)/R(I) ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  7: ',C
*
*DA   Z = LOG(EXP(X)) - X ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  8: ',C
*
*DA   Z = (LOG(SQR(SQRT(EXP(((X+Y)))))))  - X - Y  ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST  9: ',C
```

```
*
*DA   Z =  SIN (X)  -  COS( 2*ATAN(R(I)/R(I)) - X) ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST 10: ',C
*
*DA   Z = X - R(I) - XYZ(17-16,SIN(12.D0)/SIN(12.D0)) + R(I) ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST 11: ',C
*
*DA   Z = Y - XYZ(2,1+1) ;
*
      CALL DAABS(Z,C)
      WRITE(*,*)'     C FOR TEST 12: ',C
*
*     THE FOLLOWING SUBROUTINE ALLOWS ACCESS TO A SPECIFIC COMPONENT OF
*     A DA VECTOR. IT IS USUALLY NOT REQUIRED IN A PROGRAM.
*
      CALL DAPEK(X,JJ,RRR)

*DA   C = RRR - DARE(X) ;
*
      WRITE(*,*)'     C FOR TEST 13: ',C
*
  500 CONTINUE
*
      RETURN
      END
*
      FUNCTION TTAN(X)
*     ****************
*
*     THIS IS A DA FUNCTION THAT COMPUTES THE TANGENT OF A DA
*     QUANTITY X AS SIN(X)/COS(X)
*
      COMMON NO,NV
*
*DA   B D ; D V DA EXT X NO NV ; D V DA FUN TTAN NO NV ;
*DA   E D ;
*
*DA   TTAN = SIN(X) / COS (X) ;
*
      RETURN
```

```
      END
* D
```

7 Appendix 2: Sample Code After Precompilation

```
      PROGRAM TEST
*     ************
*
*     THIS PROGRAM TESTS THE DA PRECOMPILER AND PACKAGE. IT EVALUATES
*     SEVERAL ARITHMETIC DA EXPRESSIONS, ALL OF WHICH ARE
*     IDENTICAL TO ZERO. THE SUM OF ABSOLUTE VALUES OF THE RESULTING DA
*     VECTOR THUS HAS TO VANISH. THESE ABSOLUTE VALUES ARE OUTPUT TO THE
*     SCREEN; SHOULD THERE BE ANY QUANTITY THAT IS NOT ZERO (UP TO ROUNDOFF),
*     THERE IS AN ERROR IN THE DA PACKAGE OR PRECOMPILER.
*
*     THE FILE ALSO SERVES AS AN EXAMPLE FOR THE SYNTAX AND PRACTICAL USE
*     OF DA
*
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON NO,NV
*
  5   CONTINUE
*
      WRITE(*,*)' '
      WRITE(*,*)'     GIVE ORDER, NUMBER OF VARIABLES'
      WRITE(*,*)' '
      READ(*,*) NO,NV
*
      CALL DAINI(NO,NV,1)
*
      CALL ARITEST
*
      STOP
      END
*
      SUBROUTINE ARITEST
*     ******************
```

```
*
*     THIS SUBROUTINE PERFORMS SEVERAL ARITHMETICAL CHECKS; IF EVERYTHING
*     IS OK, THE RESULTING DA VECTOR SHOULD BE IDENTICAL TO ZERO.
*
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
*
      COMMON NO, NV
      DIMENSION R(15)
      INTEGER JJ(20)
      DATA JJ /20*0/
      INTEGER J
*
*DA   B D ; D V DA INT X NO NV ; D V DA INT Y NO NV ;
*DA   D V DA INT Z NO NV ; D V DA INT XYZ NO NV 7 4 ; D V RE INT R 15 ;
*DA   D V IN INT I ; D V RE INT RRR ; D V RE INT C ; D V IN INT J ;
*DA   D V IN INT ITEST ;
*DA   D F RE DARE 1 ; D F DA TTAN 1 ; D F RE FLOAT 1 ;
*DA   E D ;
*DA {
      INTEGER X
      INTEGER Y
      INTEGER Z
      INTEGER XYZ     (7,4)
      INTEGER LFOX0, LFOX1, ISCRDA, ISCRRI,IDA0
      DOUBLE PRECISION RSCRRI
      COMMON /DASCR/ ISCRDA(100),RSCRRI(100),ISCRRI(100),IDA0
      SAVE X
      SAVE Y
      SAVE Z
      SAVE XYZ
      SAVE LFOX0, LFOX1
      DATA LFOX0   / 0 /
      IF(LFOX0.EQ.0) THEN
         LFOX0 = 1
         CALL DAKEY('FOX V2.1')
         CALL DAALL(X       ,1,'X         ',NO,NV)
         CALL DAALL(Y       ,1,'Y         ',NO,NV)
         CALL DAALL(Z       ,1,'Z         ',NO,NV)
         CALL DAALL(XYZ     ,1*(7)*(4),'XYZ       ',NO,NV)
      ENDIF
      IDAA = IDA0
*DA }
*
      IRAN = 123454321
```

```
      WRITE(*,*) '     GIVE NUMBER OF TESTS'
      READ(*,*) NTEST
*
      DO 500 ITEST=1,NTEST
*
      DO 10 I=1,15
   10 R(I) = DABRAN(IRAN)
      I = 7
*
*     THE TWO FOLLOWING COMMAND FILL THE DA VECTORS X AND Y WITH
*     RANDOM ENTRIES; THEY ARE USUALLY NOT REQUIRED IN A PROGRAM
*
      CALL DARAN(X,1.D0)                                                *DA
      CALL DARAN(Y,.5D0)                                                *DA
*
      J = 1
*DA   XYZ(22-21,3/3) = X ;                                              *DA
      ISCRRI(  1+IDAA) = (3          ) / (3          )
      ISCRRI(  2+IDAA) = (22         ) - (21         )
      CALL DACOP(X          ,XYZ        (ISCRRI(  2+IDAA),ISCRRI(  1+IDA
     *A)))
*DA   XYZ(2,2*J) = Y ;                                                 *DA
      ISCRRI(  1+IDAA) = (2          ) * J
      CALL DACOP(Y          ,XYZ        ((2),ISCRRI(  1+IDAA)))
*
*
*DA   Z = (X+Y)**2-X**2.D0-2.D0*X*Y-Y*Y ;                              *DA
      CALL DAADD(X          ,Y          ,ISCRDA(  1+IDAA))
      CALL DAEXC(ISCRDA(  1+IDAA),1.D0*(2          ),ISCRDA(  2+IDAA))
      CALL DAEXC(X          ,1.D0*(2.D0       ),ISCRDA(  3+IDAA))
      CALL DACMU(X          ,1.D0*(2.D0       ),ISCRDA(  4+IDAA))
      CALL DAMUL(ISCRDA(  4+IDAA),Y          ,ISCRDA(  5+IDAA))
      CALL DAMUL(Y          ,Y          ,ISCRDA(  6+IDAA))
      CALL DASUB(ISCRDA(  2+IDAA),ISCRDA(  3+IDAA),ISCRDA(  7+IDAA))
      CALL DASUB(ISCRDA(  7+IDAA),ISCRDA(  5+IDAA),ISCRDA(  8+IDAA))
      CALL DASUB(ISCRDA(  8+IDAA),ISCRDA(  6+IDAA),ISCRDA(  9+IDAA))
      CALL DACOP(ISCRDA(  9+IDAA),Z          )
*
*     THE ROUTINE DAABS COMPUTES THE SUM OF THE ABSOLUTE VALUES OF THE
*     COMPONENTS IN A DA VECTOR; IT IS ALSO USUALLY NOT REQUIRED IN A
*     PROGRAM
*
      CALL DAABS(Z,C)                                                   *DA
```

```
      WRITE(*,*)'     C FOR TEST  1: ',C
*
*DA   Z = (R(I)/66/X) * (X/R(I)*66) - 1.D0 + EXP(X) - EXP(1.D0)**X ;     *DA
      RSCRRI(  1+IDAA) = R          (I          )
      RSCRRI(  2+IDAA) = R          (I          )
      RSCRRI(  3+IDAA) = RSCRRI(  1+IDAA) / (66         )
      CALL DADIC(X          ,1.D0*RSCRRI(  3+IDAA),ISCRDA(  4+IDAA))
      CALL DACDI(X          ,1.D0*RSCRRI(  2+IDAA),ISCRDA(  5+IDAA))
      CALL DACMU(ISCRDA(  5+IDAA),1.D0*(66         ),ISCRDA(  6+IDAA))
      CALL DAFUN('EXP   ',X          ,ISCRDA(  7+IDAA))
      RSCRRI(  8+IDAA) = EXP   ((1.D0       ))
      CALL DACEX(X          ,1.D0*RSCRRI(  8+IDAA),ISCRDA(  9+IDAA))
      CALL DAMUL(ISCRDA(  4+IDAA),ISCRDA(  6+IDAA),ISCRDA( 10+IDAA))
      CALL DACSU(ISCRDA( 10+IDAA),1.D0*(1.D0       ),ISCRDA( 11+IDAA))
      CALL DAADD(ISCRDA( 11+IDAA),ISCRDA(  7+IDAA),ISCRDA( 12+IDAA))
      CALL DASUB(ISCRDA( 12+IDAA),ISCRDA(  9+IDAA),ISCRDA( 13+IDAA))
      CALL DACOP(ISCRDA( 13+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                   *DA
      WRITE(*,*)'     C FOR TEST  2: ',C
*
*DA   Z = 1.D0 - SQR(X+Y) / (X+Y) / (X+Y) ;                            *DA
      CALL DAADD(X          ,Y          ,ISCRDA(  1+IDAA))
      CALL DAADD(X          ,Y          ,ISCRDA(  2+IDAA))
      CALL DAADD(X          ,Y          ,ISCRDA(  3+IDAA))
      CALL DAFUN('SQR   ',ISCRDA(  1+IDAA),ISCRDA(  4+IDAA))
      CALL DADIV(ISCRDA(  4+IDAA),ISCRDA(  2+IDAA),ISCRDA(  5+IDAA))
      CALL DADIV(ISCRDA(  5+IDAA),ISCRDA(  3+IDAA),ISCRDA(  6+IDAA))
      CALL DASUC(ISCRDA(  6+IDAA),1.D0*(1.D0       ),ISCRDA(  7+IDAA))
      CALL DACOP(ISCRDA(  7+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                   *DA
      WRITE(*,*)'     C FOR TEST  3: ',C
*
*DA   Z = SIN(30.D0)*COS(40.D0)-2.D0 - SIN(30.D0)*COS(40.D0)+2.D0 ;     *DA
      RSCRRI(  1+IDAA) = SIN   ((30.D0      ))
      RSCRRI(  2+IDAA) = COS   ((40.D0      ))
      RSCRRI(  3+IDAA) = SIN   ((30.D0      ))
      RSCRRI(  4+IDAA) = COS   ((40.D0      ))
      RSCRRI(  5+IDAA) = RSCRRI(  1+IDAA) * RSCRRI(  2+IDAA)
      RSCRRI(  6+IDAA) = RSCRRI(  3+IDAA) * RSCRRI(  4+IDAA)
      RSCRRI(  7+IDAA) = RSCRRI(  5+IDAA) - (2.D0       )
      RSCRRI(  8+IDAA) = RSCRRI(  7+IDAA) - RSCRRI(  6+IDAA)
      RSCRRI(  9+IDAA) = RSCRRI(  8+IDAA) + (2.D0       )
```

```
      RSCRRI(100) = RSCRRI(  9+IDAA)
      CALL DACON(Z          ,RSCRRI(100))
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  4: ',C
*
*DA   Z = X ;                                                            *DA
      CALL DACOP(X          ,Z          )
*DA   Z = Z - X ;                                                        *DA
      CALL DASUB(Z          ,X          ,ISCRDA(  1+IDAA))
      CALL DACOP(ISCRDA(  1+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  5: ',C
*
*DA   Z = 9 - X - 9 + X + R(I)*X*R(I) - R(I)*R(I)*X ;                    *DA
      RSCRRI(  1+IDAA) = R          (I          )
      RSCRRI(  2+IDAA) = R          (I          )
      RSCRRI(  3+IDAA) = R          (I          )
      RSCRRI(  4+IDAA) = R          (I          )
      CALL DACMU(X          ,1.D0*RSCRRI(  1+IDAA),ISCRDA(  5+IDAA))
      CALL DACMU(ISCRDA(  5+IDAA),1.D0*RSCRRI(  2+IDAA),ISCRDA(  6+IDAA)
     *)
      RSCRRI(  7+IDAA) = RSCRRI(  3+IDAA) * RSCRRI(  4+IDAA)
      CALL DACMU(X          ,1.D0*RSCRRI(  7+IDAA),ISCRDA(  8+IDAA))
      CALL DASUC(X          ,1.D0*(9          ),ISCRDA(  9+IDAA))
      CALL DACSU(ISCRDA(  9+IDAA),1.D0*(9          ),ISCRDA( 10+IDAA))
      CALL DAADD(ISCRDA( 10+IDAA),X          ,ISCRDA( 11+IDAA))
      CALL DAADD(ISCRDA( 11+IDAA),ISCRDA(  6+IDAA),ISCRDA( 12+IDAA))
      CALL DASUB(ISCRDA( 12+IDAA),ISCRDA(  8+IDAA),ISCRDA( 13+IDAA))
      CALL DACOP(ISCRDA( 13+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  6: ',C
*
*DA   Z = SIN(X)*SIN(X) + SQR(COS(X)) - R(I)/R(I) ;                      *DA
      CALL DAFUN('COS   ',X          ,ISCRDA(  1+IDAA))
      CALL DAFUN('SIN   ',X          ,ISCRDA(  2+IDAA))
      CALL DAFUN('SIN   ',X          ,ISCRDA(  3+IDAA))
      CALL DAFUN('SQR   ',ISCRDA(  1+IDAA),ISCRDA(  4+IDAA))
      RSCRRI(  5+IDAA) = R          (I          )
      RSCRRI(  6+IDAA) = R          (I          )
      CALL DAMUL(ISCRDA(  2+IDAA),ISCRDA(  3+IDAA),ISCRDA(  7+IDAA))
      RSCRRI(  8+IDAA) = RSCRRI(  5+IDAA) / RSCRRI(  6+IDAA)
```

```
      CALL DAADD(ISCRDA(  7+IDAA),ISCRDA(  4+IDAA),ISCRDA(  9+IDAA))
      CALL DACSU(ISCRDA(  9+IDAA),1.D0*RSCRRI(  8+IDAA),ISCRDA( 10+IDAA)
     *)
      CALL DACOP(ISCRDA( 10+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  7: ',C
*
*DA   Z = LOG(EXP(X)) - X ;                                              *DA
      CALL DAFUN('EXP   ',X          ,ISCRDA(  1+IDAA))
      CALL DAFUN('LOG   ',ISCRDA(  1+IDAA),ISCRDA(  2+IDAA))
      CALL DASUB(ISCRDA(  2+IDAA),X          ,ISCRDA(  3+IDAA))
      CALL DACOP(ISCRDA(  3+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  8: ',C
*
*DA   Z = (LOG(SQR(SQRT(EXP(((X+Y)))))))  - X - Y  ;                     *DA
      CALL DAADD(X          ,Y          ,ISCRDA(  1+IDAA))
      CALL DAFUN('EXP   ',ISCRDA(  1+IDAA),ISCRDA(  2+IDAA))
      CALL DAFUN('SQRT  ',ISCRDA(  2+IDAA),ISCRDA(  3+IDAA))
      CALL DAFUN('SQR   ',ISCRDA(  3+IDAA),ISCRDA(  4+IDAA))
      CALL DAFUN('LOG   ',ISCRDA(  4+IDAA),ISCRDA(  5+IDAA))
      CALL DASUB(ISCRDA(  5+IDAA),X          ,ISCRDA(  6+IDAA))
      CALL DASUB(ISCRDA(  6+IDAA),Y          ,ISCRDA(  7+IDAA))
      CALL DACOP(ISCRDA(  7+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST  9: ',C
*
*DA   Z =  SIN (X)  -  COS( 2*ATAN(R(I)/R(I)) - X) ;                     *DA
      RSCRRI(  1+IDAA) = R          (I          )
      RSCRRI(  2+IDAA) = R          (I          )
      RSCRRI(  3+IDAA) = RSCRRI(  1+IDAA) / RSCRRI(  2+IDAA)
      RSCRRI(  4+IDAA) = ATAN  (RSCRRI(  3+IDAA))
      RSCRRI(  5+IDAA) = (2          ) * RSCRRI(  4+IDAA)
      CALL DASUC(X          ,1.D0*RSCRRI(  5+IDAA),ISCRDA(  6+IDAA))
      CALL DAFUN('SIN   ',X          ,ISCRDA(  7+IDAA))
      CALL DAFUN('COS   ',ISCRDA(  6+IDAA),ISCRDA(  8+IDAA))
      CALL DASUB(ISCRDA(  7+IDAA),ISCRDA(  8+IDAA),ISCRDA(  9+IDAA))
      CALL DACOP(ISCRDA(  9+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                    *DA
      WRITE(*,*)'     C FOR TEST 10: ',C
```

```
*
*DA   Z = X - R(I) - XYZ(17-16,SIN(12.D0)/SIN(12.D0)) + R(I) ;          *DA
      RSCRRI(  1+IDAA) = SIN   ((12.D0      ))
      RSCRRI(  2+IDAA) = SIN   ((12.D0      ))
      RSCRRI(  3+IDAA) = RSCRRI(  1+IDAA) / RSCRRI(  2+IDAA)
      ISCRRI(  4+IDAA) = (17         ) - (16         )
      RSCRRI(  5+IDAA) = R          (I          )
      CALL DACOP(XYZ        (ISCRRI(  4+IDAA),RSCRRI(  3+IDAA)),ISCRDA(
     * 6+IDAA))
      RSCRRI(  7+IDAA) = R          (I          )
      CALL DACSU(X          ,1.D0*RSCRRI(  5+IDAA),ISCRDA(  8+IDAA))
      CALL DASUB(ISCRDA(  8+IDAA),ISCRDA(  6+IDAA),ISCRDA(  9+IDAA))
      CALL DACAD(ISCRDA(  9+IDAA),1.D0*RSCRRI(  7+IDAA),ISCRDA( 10+IDAA)
     *)
      CALL DACOP(ISCRDA( 10+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                   *DA
      WRITE(*,*)'     C FOR TEST 11: ',C
*
*DA   Z = Y - XYZ(2,1+1) ;                                              *DA
      ISCRRI(  1+IDAA) = (1          ) + (1          )
      CALL DACOP(XYZ        ((2),ISCRRI(  1+IDAA)),ISCRDA(  2+IDAA))
      CALL DASUB(Y          ,ISCRDA(  2+IDAA),ISCRDA(  3+IDAA))
      CALL DACOP(ISCRDA(  3+IDAA),Z          )
*
      CALL DAABS(Z,C)                                                   *DA
      WRITE(*,*)'     C FOR TEST 12: ',C
*
*     THE FOLLOWING SUBROUTINE ALLOWS ACCESS TO A SPECIFIC COMPONENT OF
*     A DA VECTOR. IT IS USUALLY NOT REQUIRED IN A PROGRAM.
*
      CALL DAPEK(X,JJ,RRR)                                              *DA

*DA   C = RRR - DARE(X) ;                                               *DA
      IDAO = IDAA +     1
      RSCRRI(  1+IDAA) = DARE       (X          )
      IDAO = IDAA -     1
      RSCRRI(  2+IDAA) = RRR         - RSCRRI(  1+IDAA)
      C           = RSCRRI(  2+IDAA)
*
      WRITE(*,*)'     C FOR TEST 13: ',C
*
  500 CONTINUE
*
```

```
      RETURN
      END
*
      FUNCTION TTAN(X)
*     ****************
*
*     THIS IS A DA FUNCTION THAT COMPUTES THE TANGENT OF A DA
*     QUANTITY X AS SIN(X)/COS(X)
*
      COMMON NO,NV
*
*DA   B D ; D V DA EXT X NO NV ; D V DA FUN TTAN NO NV ;
*DA   E D ;
*DA {
      INTEGER X
      INTEGER TTAN
      INTEGER LFOX0, LFOX1, ISCRDA, ISCRRI,IDA0
      DOUBLE PRECISION RSCRRI
      COMMON /DASCR/ ISCRDA(100),RSCRRI(100),ISCRRI(100),IDA0
      SAVE LFOX0, LFOX1
      DATA LFOX0   / 0 /
      IF(LFOX0.EQ.0) THEN
         LFOX0 = 1
         CALL DAKEY('FOX V2.1')
         CALL DAALL(LFOX1   ,1,'LFOX1     ',NO,NV)
      ENDIF
      IDAA = IDA0
      TTAN     = LFOX1
*DA }
*
*DA   TTAN = SIN(X) / COS (X) ;                                          *DA
      CALL DAFUN('SIN   ',X          ,ISCRDA(  1+IDAA))
      CALL DAFUN('COS   ',X          ,ISCRDA(  2+IDAA))
      CALL DADIV(ISCRDA(  1+IDAA),ISCRDA(  2+IDAA),ISCRDA(  3+IDAA))
      CALL DACOP(ISCRDA(  3+IDAA),TTAN       )
*
      RETURN
      END
```