I-19427 DR·0806-0

**SANDIA REPORT** SAND84—1161 • Unlimited Release • UC—32

Printed February 1985


SAND--84-1161

DE85 006856

# User's Guide for Wilson-Fowler Spline Software SPLPKG, WFCMPR, WFAPPX CADCAM—010

Sharon K. Fletcher

**Prepared by**
**Sandia National Laboratories**
**Albuquerque, New Mexico 87185 and Livermore, California 94550**
**for the United States Department of Energy**
**under Contract DE-AC04-76DP00789**

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

NOTICE
PORTIONS OF THIS REPORT ARE ILLEGIBLE.
It has been reproduced from the best available copy to permit the broadest possible availability.

Distribution
Category UC-32

SAND84-1161
Unlimited Release
Printed February 1985

# User's Guide for Wilson-Fowler Spline Software SPLPKG WFCMPR WFAPPX

CADCAM—010

*Sharon K. Fletcher*
*CAD/CAM Integration Division 2811*
*Sandia National Laboratories*
*Albuquerque, New Mexico 87185*

**DISCLAIMER**

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

## Abstract

The Wilson-Fowler spline is widely used in computer aided manufacturing, but is not available in all commercial CAD/CAM systems. These three programs provide a capability for generating, comparing, and approximating Wilson-Fowler splines. SPLPKG generates a spline passing through given nodes, and computes a piecewise linear approximation to the spline. WFCMPR computes the difference between two splines with common nodes. WFAPPX computes the difference between a spline and a piecewise linear curve. The programs are in Fortran 77 and are machine independent.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

## Table of Contents


| | |
|---|---|
| Introduction | 1 |
| Section I: SPLPKG | 3 |
| Section II: WFCMPR | 22 |
| Section III: WFAPPX | 30 |
| References | 38 |


DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

## Introduction

The three programs described here make up the set of spline software tools described in "Recommended Practices for Spline Usage in CAD/CAM Systems" [1]. They provide a means for generating Wilson-Fowler splines, comparing Wilson-Fowler splines, and evaluating a piecewise linear approximation to a Wilson-Fowler spline.

The Wilson-Fowler spline is widely used in Computer Aided Design and Manufacture. The original algorithm developed by Fowler and Wilson [4] has been incorporated into APT and many commercial CAD/CAM systems. However the implementations do not all produce identical results, and many newer CAD systems use other types of splines. The Wilson-Fowler spline is favored for many applications because it produces a smooth, low curvature fit to planar data points.

Current product definition exchange procedures are not yet fully automated, and spline exchange usually requires reentry of spline nodes into a receiver's CAD/CAM system and recomputation of the spline by that system. Report [1] issues guidelines for minimizing errors and verifying the exchange of spline data. The software tools provided here may be helpful in conjunction with those guidelines.

Program WFCMPR computes the maximum difference between two Wilson-Fowler splines, and may be used to verify the spline recomputed by a receiving system. Program SPLPKG generates a Wilson-Fowler spline passing through given nodes (with given end conditions), and also generates a piecewise linear approximation to that spline within a given tolerance. This program may be used to generate a "desired" spline against which to compare other splines generated by CAD/CAM systems. It may also be used to generate an acceptable approximation to a desired spline in the event that an acceptable spline cannot be generated by the receiving CAD/CAM system. Program WFAPPX computes the maximum difference between a Wilson-Fowler spline and a piecewise linear curve. This may be used to accept or reject a proposed approximation to a desired Wilson-Fowler spline, even if the origin of the approximation is unknown.

One section of this user's guide is devoted to each program. Each section begins with a description of the program's purpose, method, inputs/outputs, and limitations. Following are sample run(s) and data files. Program listings are last within each section.

# Section I
# SPLPKG

# Program SPLPKG

**Purpose:** The program SPLPKG computes a Wilson-Fowler spline and writes an IGES file of points evaluated on the spline and/or a file containing the spline description.

**Language/Libraries:** The program package SPLPKG is written in Fortran 77. It is self-contained, requiring no libraries. Function DPNTLC and subroutine WFEVAL are also used by program WFAPPX, and are copied.

**Inputs** (To be read from a file): Spline nodes:

- Number of spline nodes
- (X,Y) values of nodes.

**Inputs** (By the operator):

- Convergence criterion for computed spline
- Entry and exit angles - may be defaulted
- Maximum deviation of piecewise linear approximation to spline
- Number of points for initial piecewise linear approximation
- Flag indicating whether or not to write points into an IGES file
- IGES level assignment for the points
- Flag indicating whether or not to write spline definition to a file.

**Outputs**

- IGES file containing breakpoints of piecewise linear approximation
- File containing spline definition.

**Method:**

The algorithm for computing the Wilson-Fowler spline from given nodes and end angles was provided by W. R. Melvin [2]. The algorithm for computing the distance from a point (on the spline) to a piecewise linear curve was provided by J. D. Emery [3].

The entry and exit angles for the spline may be defaulted, in which case they are computed by fitting circles to the first (last) three nodes.

A piecewise linear approximation to the spline is computed and the breakpoints of it may optionally be written out in IGES file format. The operator inputs the maximum allowable deviation from the spline, and the initial number of points on the spline to be used for the approximation. The deviation is checked midway between each breakpoint, and the number of points is doubled until the deviation criterion is satisfied.

The spline definition, in terms of local tangents, may optionally be dumped to a file in one of two formats:

- The 5 column format is one line per node containing the X node value, Y node value, start local tangent for the segment that begins with that node, end local tangent for the segment, and length of the segment.
- The linear display format is number of nodes, followed by (X,Y) pairs for the nodes, followed by local tangent pairs for the segments. This is the format needed for input to programs WFAPPX and WFCMPR.

**Limitations:**

The following limitations are explicit in PARAMETER statements at the beginning of the program:

- NNODES (maximum number of spline nodes) = 100
- XSMALL (smallest number used to prevent machine underflow) = 1E-30
- XBIG (largest number used to prevent machine overflow) = 1E+30
- NEPTS (maximum number of evaluation points for linear appx) = 1600

The following limitations are explicit in character statements near the beginning of SPLPKG and its subroutine WRIGES:

- all filename lengths = 12 characters.

**Operational Notes:**

This program may be used to obtain either the definition of a Wilson-Fowler spline, or a piecewise linear approximation to a Wilson-Fowler spline, or both. Appropriate output options are provided. If the spline definition dump is all that is desired, calculation of the linear approximation is incidental, and computation can be minimized by specifying a small number of initial points (eg., 10) and a large deviation (eg., 1E10) for the linear approximation.

```
$ run splpkg
 Enter convergence criterion for spline (eps):
.000001
 Input filename ?
test1
 Calculated entry, exit angles =   323.1301       548.1301
 Enter new values, if desired (>360 TO KEEP DEFAULTS):
500 500
 Enter max allowed deviation for a piecewise
 linear approximation to the spline (delta):
.00001
 Enter initial number of points for the approximation:
200
 Number of interpolated points is         200
 Maximum deviation is  1.1346096E-04
 Number of interpolated points is         399
 Maximum deviation is  2.9065832E-05
 Number of interpolated points is         797
 Maximum deviation is  7.7114091E-06
 Want these points written to an IGES file (Y/N)?
n
 Choose file format for dump of spline definition:
          N = no dump
 (default) T = 5 column table: (x, y, tana, tanb, length)
          L = linear display: n, (x,y), (tana,tanb)
n
FORTRAN STOP
$
```

SPLPKG Sample Run #1

```
$ run splpkg
 Enter convergence criterion for spline (eps):
.000001
 Input filename ?
test1
 Calculated entry, exit angles =   323.1301       548.1301
 Enter new values, if desired (>360 TO KEEP DEFAULTS):
500 500
 Enter max allowed deviation for a piecewise
 linear approximation to the spline (delta):
1e10
 Enter initial number of points for the approximation:
10
 Number of interpolated points is          10
 Maximum deviation is  5.0445277E-02
 Want these points written to an IGES file (Y/N)?
y
 Enter IGES level number:
1
 Enter IGES file name:
iges1
 Choose file format for dump of spline definition:
           N = no dump
 (default) T = 5 column table: (x, y, tana, tanb, length)
           L = linear display: n, (x,y), (tana,tanb)
t
 Enter dump file name:
dump1
FORTRAN STOP
$
```

SPLPKG Sample Run #2

```
5
5.  0.
4.  1.
3.  4.
1. 5.
0  5.
```

Input File for SPLPKG Sample Runs.

```
INTERPOLATED POINTS FROM WILSON-FOWLER SPLINE PACKAGE                    S      1
,,19HINTERPOLATED POINTS,,6HSPLPKG;                                      G      1
     116       1       1       1       1                                 D      1
     116                       1                       POINT            D      2
     116       2       1       1       1                                 D      3
     116                       1                       POINT            D      4
     116       3       1       1       1                                 D      5
     116                       1                       POINT            D      6
     116       4       1       1       1                                 D      7
     116                       1                       POINT            D      8
     116       5       1       1       1                                 D      9
     116                       1                       POINT            D     10
     116       6       1       1       1                                 D     11
     116                       1                       POINT            D     12
     116       7       1       1       1                                 D     13
     116                       1                       POINT            D     14
     116       8       1       1       1                                 D     15
     116                       1                       POINT            D     16
     116       9       1       1       1                                 D     17
     116                       1                       POINT            D     18
     116      10       1       1       1                                 D     19
     116                       1                       POINT            D     20
116,  0.5000000E+01,  0.0000000E+00,0.0000000;                           1P      1
116,  0.4338405E+01,  0.5660306E+00,0.0000000;                           3P      2
116,  0.3848392E+01,  0.1288788E+01,0.0000000;                           5P      3
116,  0.3594636E+01,  0.2119220E+01,0.0000000;                           7P      4
116,  0.3416715E+01,  0.2974932E+01,0.0000000;                           9P      5
116,  0.3124311E+01,  0.3792482E+01,0.0000000;                          11P      6
116,  0.2538533E+01,  0.4489997E+01,0.0000000;                          13P      7
116,  0.1754998E+01,  0.4863973E+01,0.0000000;                          15P      8
116,  0.8680618E+00,  0.5012667E+01,0.0000000;                          17P      9
116, -0.2384186E-06,  0.5000000E+01,0.0000000;                          19P     10
S      1G      1D     20P     10                                         T      1
```

Output IGES File from SPLPKG Sample Run #2.

| | | | | |
|---|---|---|---|---|
| 5.000000 | 0.0000000E+00 | 0.1428571 | -0.2385142 | 1.414214 |
| 4.000000 | 1.000000 | 0.2336244 | 0.2775339 | 3.162278 |
| 3.000000 | 4.000000 | -0.5655161 | 0.3745124 | 2.236068 |
| 1.000000 | 5.000000 | -0.1056955 | 0.1428573 | 1.000000 |
| 0.0000000E+00 | 5.000000 | 0.1428573 | 0.0000000E+00 | 0.0000000E+00 |

Output Dump File from SPLPKG Sample Run #2.

```
      PROGRAM SPLPKG
      PARAMETER (NNODES=100, NEPTS=1600)
      PARAMETER (XSMALL=1E-30, XBIG=1E+30)
      PARAMETER (NE02=NEPTS/2)
      CHARACTER*12 F2, F4
      CHARACTER*1 ANS
      DIMENSION TABL(5,NNODES)
      DIMENSION XOUT(NEPTS),YOUT(NEPTS),XTMP(NE02),YTMP(NE02)
      COMMON /MACHIN/ ZERO, BIG
      DATA ZERO, BIG / XSMALL, XBIG /
C
C     PROGRAM TO CALCULATE A WILSON-FOWLER SPLINE THROUGH A
C     SET OF NODES, AND THEN INTERPOLATE ON THE SPLINE TO
C     OBTAIN A SET OF OUTPUT POINTS REPRESENTING A PIECEWISE
C     LINEAR APPROXIMATION TO THE SPLINE.  THE OUTPUT POINTS
C     MAY BE WRITTEN INTO AN IGES FILE.  THE SPLINE DEFINITION
C     IN TERMS OF LOCAL TANGENTS MAY BE WRITTEN TO A FILE.
C
C     THIS PACKAGE OF FORTRAN ROUTINES IS WRITTEN IN FORTRAN77.
C     I/O OCCURS IN:
C          MAIN PROGRAM
C          SUBROUTINE NWF  (3 ERROR MESSAGES NEAR END)
C          SUBROUTINE WRIGES.
C
C     ANY QUESTIONS SHOULD BE REFERED TO:
C           SHARON FLETCHER
C           SANDIA NATL LABORATORIES, DIV 7611
C           (505) 846 - 5506
C
C     TWO END CONDITIONS ARE REQUIRED IN ADDITION TO THE INPUT
C     NODES TO DEFINE THE SPLINE; THESE MAY BE ENTERED AS
C     ENTRY AND EXIT ANGLES (DEGREES) OR MAY BE DEFAULTED, IN
C     WHICH CASE THE PROGRAM CALCULATES THE TANGENTS TO CIRCLES
C     FIT THROUGH THE INITIAL 3 AND LAST 3 POINTS.
C
C     EPSILON IS A CONVERGENCE CRITERION FOR SUBROUTINE NWF --
C     NWF GUARANTEES TO CONVERGE TO THE TRUE (THEORETICAL)
C     SOLUTION TO WITHIN EPS UNITS ALONG THE ENTIRE LENGTH OF
C     THE SPLINE.
C
C     DELTA IS THE MAXIMUM ALLOWED DEVIATION FOR THE PIECEWISE LINEAR
C     APPROXIMATION TO THE SPLINE.  THE SPLINE IS INITIALLY EVALUATED AT
C     NPTS POINTS FOR THE APPROXIMATION.  THEN THE NUMBER OF EVALUATION
C     POINTS IS DOUBLED UNTIL THE APPROXIMATION IS WITHIN DELTA OF
C     THE SPLINE.
C
C     INPUTS:
C        (UNIT 2): N = NUMBER OF POINTS (SPLINE NODES)
C                  TABL(1,I),TABL(2,I) = X & Y COORDINATES OF A POINT
C        (OPERATOR): EPS = SPLINE CONVERGENCE CRITERION
C                    EN = OVERRIDE ENTRY ANGLE IN DEGREES
C                    EX = OVERRIDE EXIT ANGLE IN DEGREES
C                    DELTA = MAX DEVIATION OF LINEAR APPROXIMATION
C                    NPTS = NUMBER OF INITIAL POINTS FOR LINEAR APPX
C                    ANS = IGES FILE WRITE OPTION
C                    LVL = IGES LEVEL ON WHICH TO WRITE POINTS
C                    ANS = DUMP FILE OPTION
C                    F2, F3, F4 = FILENAMES
C
C     OUTPUTS:
C        (UNIT 3): OPTIONAL IGES FILE CONTAINING INTERPOLATED POINTS
C        (UNIT 4): OPTIONAL DUMP FILE CONTAINING SPLINE DEFINITION
C
```

```
C
      PRINT *, ' Enter convergence criterion for spline (eps):'
      READ *, EPS
C
C     READ THE SPLINE NODES INTO TABL.
C
         PRINT *, ' Input filename ? '
         READ (*,200) F2
         OPEN (UNIT = 2, FILE = F2, STATUS = 'OLD')
C
         READ (2,*) N
         DO 10 I=1,N
            READ (2,*) TABL(1,I), TABL(2,I)
10       CONTINUE
C
C     CALCULATE ENTRY AND EXIT ANGLES
C
         CALL ANGLES(N,TABL,ENTRYA,EXITA)
C
C     ALLOW OVERRIDE OF CALCULATED ANGLES
C
         PRINT *, ' Calculated entry, exit angles =', ENTRYA,EXITA
         PRINT *, ' Enter new values, if desired (>360 TO KEEP DEFAULTS):'
         READ *, EN,EX
            IF (EN .LE. 360.) ENTRYA = EN
            IF (EX .LE. 360.) EXITA = EX
C
C     CALCULATE WILSON-FOWLER SPLINE
C
         CALL NWF(N,TABL,ENTRYA,EXITA,EPS)
C
C     OBTAIN PIECEWISE LINEAR APPROXIMATION BY EVALUATING
C     SPLINE AT NPTS POINTS
C
      PRINT *, ' Enter max allowed deviation for a piecewise'
      PRINT *, ' linear approximation to the spline (delta):'
      READ *, DELTA
      PRINT *, ' Enter initial number of points for the approximation:'
      READ *, NPTS
C
C     GET TOTAL CHORD LENGTH
      TOTL = 0.
      DO 40 I=1,N-1
         TOTL = TOTL + TABL(5,I)
40    CONTINUE
C
C     INTERPOLATE AT POINTS EQUALLY SPACED ALONG THE CHORD LENGTH
C
      RINC = TOTL/(NPTS-1)
      DO 50 I=1,NPTS
         ULOCL = (I-1) * RINC
         CALL WFEVAL(N,TABL,ULOCL,XOUT(I),YOUT(I))
50    CONTINUE
C
C     CHECK AT MIDPOINTS
C
         TINC = .5 * RINC
         NINC = 2 * (NPTS-1)
60       DIST = CHEKIT(N,TABL,NPTS,XOUT,YOUT,NINC,TINC,XTMP,YTMP,.TRUE.)
         PRINT *, ' Number of interpolated points is', NPTS
         PRINT *, ' Maximum deviation is', DIST
         IF (DIST .LE. DELTA) GOTO 75
C
```

```
C       INCREASE NPTS (ADD MIDPOINTS) UNTIL APPX IS WITHIN DELTA
C
           DO 72 I=NPTS,1,-1
              J = 2*I-1
              XOUT(J) = XOUT(I)
              YOUT(J) = YOUT(I)
72         CONTINUE
           DO 73 I=1,NPTS-1
              J = 2*I
              XOUT(J) = XTMP(I)
              YOUT(J) = YTMP(I)
73         CONTINUE
           NPTS = 2 * NPTS - 1
           TINC = .5 * TINC
           NINC = 2 * (NPTS - 1)
           IF (NPTS .GT. NE02) THEN
              DIST = CHEKIT(N,TABL,NPTS,XOUT,YOUT,NINC,TINC,XTMP,YTMP,
     +                      .FALSE.)
              PRINT *, ' Number of interpolated points is', NPTS
              PRINT *, ' Maximum deviation is', DIST
              PRINT *, ' *** Warning: Deviation may exceed delta. ***'
              GOTO 75
           ENDIF
        GOTO 60
C
C       (OPTIONALLY) WRITE AN IGES FORMATTED FILE CONTAINING THE
C                    EVALUATED POINTS
C
75      CONTINUE
        PRINT *, ' Want these points written to an IGES file (Y/N)?'
        READ (*,200) ANS
        IF (ANS .EQ. 'Y' .OR. ANS .EQ. 'y') THEN
           PRINT *, ' Enter IGES level number:'
           READ *, LVL
           CALL WRIGES(LVL,NPTS,XOUT,YOUT)
        ENDIF
C
C       (OPTIONALLY) DUMP OUT SPLINE DEFINITION IN TERMS OF LOCAL TANGENTS
C
        PRINT *, ' Choose file format for dump of spline definition:'
        PRINT *, '              N = no dump'
        PRINT *, ' (default) T = 5 column table: (x, y ,tana, tanb, length)'
        PRINT *, '              L = linear display: n, (x,y), (tana,tanb)'
        READ (*,200) ANS
        IF (ANS .NE. 'N' .AND. ANS .NE. 'n') THEN
           PRINT *, ' Enter dump file name:'
           READ (*,200) F4
C(VAX)     OPEN (UNIT=4, FILE = F4, STATUS = 'NEW', CARRIAGECONTROL = 'LIST')
           OPEN (UNIT=4, FILE = F4, STATUS = 'NEW')
           IF (ANS .EQ. 'L' .OR. ANS .EQ. 'l') THEN
              WRITE (4,*) N
              DO 80 I=1,N
                 WRITE (4,*) TABL(1,I), TABL(2,I)
80            CONTINUE
              DO 85 I=1,N-1
                 WRITE (4,*) TABL(3,I), TABL(4,I)
85            CONTINUE
           ELSE
              DO 90 I=1,N
                 WRITE (4,*) (TABL(J,I), J=1,5)
90            ENDDO
           ENDIF
        ENDIF
```

```
      STOP
200   FORMAT(A)
      END
      FUNCTION CHEKIT(N,TABL,NPTS,X,Y,NINC,TINC,XTMP,YTMP,SAVFLG)
      DIMENSION X(1), Y(1), XTMP(1), YTMP(1)
      LOGICAL SAVFLG
C
C     INPUTS:
C        N = NUMBER OF SPLINE NODES
C        TABL = SPLINE DEFINITION TABLE
C        NPTS = NUMBER OF POINTS ON PIECEWISE CURVE
C        X(I), Y(I) = PIECEWISE CURVE POINTS
C        NINC = NUMBER OF INCREMENTS ALONG SPLINE TO CHECK
C        TINC = CHORD LENGTH INCREMENT CORRESPONDING TO NINC
C        SAVFLG = IF .TRUE., SAVE NEWLY EVALUATED POINTS
C
C     OUTPUTS:
C        CHEKIT = MAXIMUM DEVIATION FOUND
C        XTMP, YTMP = EVALUATED POINTS
C
      CHEKIT = 0.
C     (CHECK ODD MULTIPLES OF THE INCREMENT TINC -- THESE ARE MIDPOINTS)
      DO 50 I=1,NINC,2
         T = I * TINC
         CALL WFEVAL(N,TABL,T,U,S)
         IF (SAVFLG) THEN
            J = (I+1) / 2
            XTMP(J) = U
            YTMP(J) = S
         ENDIF
C     (LIMIT PIECEWISE CURVE TO 3 SEGMENTS AROUND SPLINE POINT BEING CHECKED)
         NK = 4
         K = (I-1) / 2
         IF (K .LT. 1) K = 1
         IF (K+3 .GT. NPTS) NK=3
         DIST = DPNTLC(NK,X(K),Y(K),U,S)
         IF (DIST.GT. CHEKIT) CHEKIT = DIST
50    CONTINUE
      RETURN
      END
      function dpntlc(nk,x,y,u,s)
      dimension x(1), y(1)
      common /machin/ zero, big
c
c     INPUTS:
c        nk = number of points (piecewise linear curve)
c        x(i) = i-th curve point x coordinate
c        y(i) = i-th curve point y coordinate
c        u = x coordinate of measurement point
c        s = y coordinate of measurement point
c
c     OUTPUT:
c        dpntlc = distance from the point (u,s) to the curve
c
c     Finds minimum distance from a point to a piecewise linear
c     curve defined by its breakpoints {x,y}.  All segments of
c     the linear curve are tested.
c
      m = nk - 1
      dpntlc = big
      do 10 i=1,m
         a = sqrt( (x( i )-u)**2 + (y( i )-s)**2 )
         b = sqrt( (x(i+1)-u)**2 + (y(i+1)-s)**2 )
```

```
            if (abs(a*b) .lt. zero) goto 50
         c = sqrt( (x(i+1)-x(i))**2 + (y(i+1)-y(i))**2 )
         cosa = ( b*b + c*c - a*a) / (2.*b*c)
         cosb = ( a*a + c*c - b*b) / (2.*a*c)
         if (cosa*cosb .ge. 0.) then
            e = b * sqrt(abs( 1-cosa**2 ))
         else
            e = aminl(a,b)
         endif
         if (e .lt. dpntlc) dpntlc = e
10       continue
         return
50       dpntlc = 0.
         return
         end
         SUBROUTINE WRIGES(LVL,NPTS,X,Y)
         DIMENSION X(1), Y(1)
         CHARACTER*12 F3
C
C        INPUTS:
C           LVL = IGES LEVEL ON WHICH TO WRITE POINT ENTITIES
C           NPTS = NUMBER OF (X,Y) POINTS
C           X = ARRAY OF X VALUES
C           Y = ARRAY OF Y VALUES
C
C        OUTPUTS:
C           (UNIT 3): IGES FILE
C
C        WRITES AN IGES FILE CONTAINING THE (X,Y) POINTS.
C
         PRINT *, ' Enter IGES file name:'
         READ (*,200) F3
C(VAX)   OPEN (UNIT = 3, FILE = F3, STATUS = 'NEW', CARRIAGECONTROL = 'LIST')
         OPEN (UNIT = 3, FILE = F3, STATUS = 'NEW')
         NPTS2 = NPTS * 2
C
C        WRITE START AND GLOBAL SECTIONS (1 CARD IMAGE EACH)
C
         WRITE (3,100)
         WRITE (3,101)
C
C        WRITE DIRECTORY SECTION (2 CARD IMAGES PER POINT)
C
         DO 10 I=1,NPTS
            I2 = I*2
            I2M1 = I2 - 1
            WRITE (3,102) I, LVL, I2M1
10          WRITE (3,103) I2
C
C        WRITE PARAMETER SECTION (1 CARD IMAGE PER POINT)
C
         DO 20 I=1,NPTS
            I2M1 = I * 2 - 1
20          WRITE (3,104) X(I), Y(I), I2M1, I
C
C        WRITE TERMINATE SECTION
C
         WRITE (3,105) NPTS2, NPTS
         CLOSE (3)
         RETURN
100      FORMAT('INTERPOLATED POINTS FROM WILSON-FOWLER SPLINE PACKAGE',
     +          19X,'S      1')
101      FORMAT(',,19HINTERPOLATED POINTS,,6HSPLPKG;'37X,'G      1')
```

```
102     FORMAT('     116',I8,2(7X,'1'),I8,32X,'D',I7)
103     FORMAT('     116',23X,'1',24X,'   POINT',8X,'D',I7)
104     FORMAT('116,',2(E15.7,','),'0.0000000;',18X,I8,'P',I7)
105     FORMAT('S      1G      1D',I7,'P',I7,40X,'T      1')
200     FORMAT(A)
        END
        SUBROUTINE ANGLES(N,TABL,ENTRYA,EXITA)
        DIMENSION TABL(5,100)
C
C       INPUTS:
C          N = NUMBER OF POINTS (SPLINE NODES)
C          TABL(1,I) = I-TH X COORDINATE
C          TABL(2,I) = I-TH Y COORDINATE
C
C       OUTPUTS:
C          ENTRYA = TANGENT ANGLE (DEGREES) FOR FIRST POINT
C          EXITA =  TANGENT ANGLE (DEGREES) FOR LAST POINT
C
C       TANGENT ANGLES ARE CALCULATED BY FITTING A CIRCLE THROUGH
C       THREE CONSECUTIVE POINTS.
C
        PI = 4. * ATAN(1.)
        DX1 = TABL(1,2) - TABL(1,1)
        DY1 = TABL(2,2) - TABL(2,1)
        DX2 = TABL(1,3) - TABL(1,2)
        DY2 = TABL(2,3) - TABL(2,2)
        RS1 = DX1**2 + DY1**2
        RS2 = DX2**2 + DY2**2
        GA1 = ATAN2(DY1,DX1)
        DEN = RS1 * DX2 + RS2 * DX1
        DEN = SIGN(AMAX1(1E-9,ABS(DEN)),DEN)
        A2 = ATAN((RS1 * DY2 + RS2 * DY1) / DEN)
        A1 = GA1 - (A2 - GA1)
        ENTRYA = A1 * 180./PI
        DXNM1 = TABL(1,N) - TABL(1,N-1)
        DYNM1 = TABL(2,N) - TABL(2,N-1)
        DXNM2 = TABL(1,N-1) - TABL(1,N-2)
        DYNM2 = TABL(2,N-1) - TABL(2,N-2)
        RSNM1 = DXNM1**2 + DYNM1**2
        RSNM2 = DXNM2**2 + DYNM2**2
        GANM1 = ATAN2(DYNM1,DXNM1)
        DEN = RSNM2 * DXNM1 + RSNM1 * DXNM2
        DEN = SIGN(AMAX1(1E-9,ABS(DEN)),DEN)
        ANM1 = ATAN((RSNM2 * DYNM1 + RSNM1 * DYNM2) / DEN)
        AN = GANM1 + PI - (ANM1 - GANM1)
        EXITA = AN * 180./PI
        RETURN
        END
        SUBROUTINE WFEVAL(N,TABL,ULOCL,U,S)
        DIMENSION TABL(5,100)
C
C       INPUTS:
C          N = NUMBER OF POINTS (SPLINE NODES)
C          TABL(1,I) = I-TH POINT X COORDINATE
C          TABL(2,I) = I-TH POINT Y COORDINATE
C          TABL(3,I) = I-TH SEGMENT ENTRY ANGLE TANGENT
C          TABL(4,I) = I-TH SEGMENT EXIT ANGLE TANGENT
C          TABL(5,I) = I-TH SEGMENT LENGTH
C          ULOCL = A VALUE ALONG THE CHORD LENGTH FOR INTERPOLATION
C
C       OUTPUTS:
C          U = GLOBAL COORDINATES X VALUE OF INTERPOLATED POINT
C          S = GLOBAL COORDINATES Y VALUE OF INTERPOLATED POINT
```

```
C
C       INTERPOLATE TO THE SPLINE AT ULOCL IN TERMS OF THE LOCAL
C       COORDINATE SYSTEM, THEN ROTATE THE POINT TO THE GLOBAL
C       COORDINATE SYSTEM.
C
C       (DETERMINE WHICH SEGMENT ULOCL IS IN -- J-TH SEGMENT)
C
        J = 1
5       IF (ULOCL .LT. TABL(5,J) .OR. J .EQ. N-1) GOTO 10
           ULOCL = ULOCL - TABL(5,J)
           J = J + 1
           GOTO 5
10      CONTINUE
C
C       (EVALUATE HERMITE FORM OF CUBIC)
C
        UMT = ULOCL - TABL(5,J)
        SLOCL = (TABL(3,J) * ULOCL * UMT**2 +
     +           TABL(4,J) * ULOCL**2 * UMT) / TABL(5,J)**2
C
C       (ROTATE AND TRANSLATE LOCAL POINT (ULOCL,SLOCL) TO
C        GLOBAL COORDINATE SYSTEM (U,S) )
C
        TH = ATAN2(TABL(2,J+1)-TABL(2,J),TABL(1,J+1)-TABL(1,J))
        U = ULOCL * COS(TH) - SLOCL * SIN(TH) + TABL(1,J)
        S = ULOCL * SIN(TH) + SLOCL * COS(TH) + TABL(2,J)
        RETURN
        END
        SUBROUTINE NWF(N,TABL,ENTRYA,EXITA,EPS)
        REAL Z(150),F(150),R(150),FPR(3,150),TABL(5,1)
        DATA IFLAG/0/,CONV/.0174532925/,C1D3/.333333333333/
C
C       SUBROUTINES NWF, FSPL, TRIDG, AND FUNCTIONS DINV, DFLIP ARE
C       ADAPTED FROM A SUBROUTINE PACKAGE WRITTEN BY W. MELVIN OF
C       LOS ALAMOS NATL LABORATORIES.  REF. REPORT LA9178.
C
C       NWF COMPUTES THE WILSON-FOWLER SPLINE WHICH PASSES THROUGH
C       THE N POINTS FOUND IN THE FIRST 2 COLUMNS OF TABL WITH
C       GIVEN END CONDITIONS, AND PUTS THE RESULTING SEGMENT
C       ENTRY TANGENTS, EXIT TANGENTS, AND SEGMENT LENGTHS
C       IN COLUMNS 3, 4, 5 OF TABL.  THE SPLINE CAN BE EASILY
C       EVALUATED FROM THIS INFORMATION USING ITS HERMITE
C       REPRESENTATION.
C       NEWTON ITERATION IS USED TO SOLVE FOR THE INTERNAL SLOPES;
C       THE ITERATION CONTINUES UNTIL THE COMPUTED SOLUTION IS WITHIN
C       EPS OF THE THEORETICAL SOLUTION ALONG THE ENTIRE LENGTH OF
C       THE CURVE.
C       THE INPUT DATA IS CHECKED FOR VALIDITY AND THE
C       SOLUTION IS CHECKED FOR EXISTENCE AND UNIQUENESS --
C       ERROR MESSAGES ARE OUTPUT.
C
C
        TLM = 0.
        SEPS = SQRT(EPS)*.1
        NM1 = N-1
        IF (NM1 .LT. 2) GOTO 999
        DO 10 I=1,NM1
           DX = TABL(1,I+1) - TABL(1,I)
           DY = TABL(2,I+1) - TABL(2,I)
           IF (I .NE. 1) GOTO 2
           THETA = ENTRYA * CONV
           GOTO 6
2          IF (I .NE. NM1) GOTO 8
```

```
            THETA = EXITA * CONV
6           SINN = SIN(THETA)
            COSN = COS(THETA)
            Z(I) = (SINN * DX - DY * COSN) / (DX * COSN + DY * SINN)
            IF (I .EQ. NM1) Z(N) = Z(I)
8           TL = SQRT(DX * DX + DY * DY)
            DX = DX / TL
            DY = DY / TL
            TABL(5,I) = TL
            TLM = AMAX1(TL,TLM)
            IF (I .EQ. 1) GOTO 9
            DX0 = DX0 + DX
            DY0 = DY0 + DY
            DEN = DX * DX0 + DY * DY0
            IF (DEN .EQ. 0.)  GOTO 999
            TL = (DX0 * DY - DX * DY0) / DEN
            TABL(3,I) = TL
            TABL(4,I) = 1. + TL * TL
            Z(I) = 0.
9           DX0 = DX
            DY0 = DY
10        CONTINUE
          TABL(3,1) = 0.
          TABL(3,N) = 0.
          TABL(5,N) = 0.
          FPR(1,NM1) = 0.
          FPR(3,NM1) = 0.
          TL = TLM * .148148148
          K = 0.
C
15        CALL FSPL(Z,N,TABL,F,FPR)
          DO 20 I=2,NM1
             R(I-1) = -F(I)
20        CONTINUE
C
          CALL TRIDG(FPR(1,2),N-2,R,IFLAG)
          IF (IFLAG .LT. 0) GOTO 999
C
          RNORM = 0.
          DO 30 I=1,N-2
             RNORM = RNORM + ABS(R(I))
30        CONTINUE
C
          DO 40 I=2,NM1
             Z(I) = Z(I) +R(I-1)
40        CONTINUE
          K = K + 1
C
          IF (K .LT. 10 .AND. RNORM .GT. SEPS) GOTO 15

C
          GAMMA = DFLIP(Z,TABL,NM1,SEPS)
          TWOAK = 2. * GAMMA * DINV(FPR(1,2),N-2) * RNORM
          IF (TWOAK .GE. 1.) GOTO 100
          E1 = TWOAK * RNORM
          TK = 0.
          DO 45 I=2,NM1
             TK = AMAX1(TK,TABL(4,I)/(1.-ABS(E1*TABL(3,I)))**2)
45        CONTINUE
          ERMAX = E1 * TL * TK
          IF (K .GT. 20) GOTO 90
          IF (ERMAX .GT. EPS) GOTO 15
C
```

```
50      TABL(3,1) = Z(1)
        DO 60 I=2,N
           DI = TABL(3,I)
           ZI = Z(I)
           T4 = TABL(4,I-1)
           TABL(3,I)   = (ZI - DI) / (1. + ZI * DI)
           TABL(4,I-1) = (ZI + DI) / (1. - ZI * DI)
           IF (I .EQ. 2) GOTO 60
           RI = TABL(5,I-1) / TABL(5,I-2)
           E1 = T4 * (1. + C1D3 / RI) - 2.
           E2 = 2. - T4 * (1. + C1D3 * RI)
           IF (DIM * TABL(3,I-2) .LT. E2 .OR. DIM * TABL(4,I-1) .GT. E1)
     +        GOTO 60
           IM1 = I - 1
           PRINT *, ' UNIQUENESS NOT SATISFIED AT NODE #', IM1
60      DIM = DI
        RETURN
C
90      PRINT *, ' ACCURACY NOT ATTAINED ... MAX ERROR =', ERMAX
        GOTO 50
C
100     IF (K .LT. 10) GOTO 15
        PRINT *, ' ITERATION FAILED TO CONVERGE !!'
        GOTO 50
C
999     PRINT *, ' ERROR IN SPLINE DATA !!!'
        STOP
        END
        SUBROUTINE FSPL(Z,N,TABL,F,FPRIME)
        REAL Z(1),TABL(5,1),F(1),FPRIME(3,1)
        ZDPI = 1.
        ZMDI = Z(1)
        ZDMP = 1. - Z(2) * TABL(3,2)
        ZPDP = Z(2) + TABL(3,2)
        RL = TABL(5,1)
C
        DO 10 I=2,N-1
           DI = TABL(3,I)
           ZDPM = ZDPI
           ZMDM = ZMDI
           ZDMI = ZDMP
           ZPDI = ZPDP
           ZDPI = 1. + Z(I) * DI
           ZMDI = Z(I) - DI
           ZDMP = 1. - Z(I+1) *TABL(3,I+1)
           ZPDP = Z(I+1) + TABL(3,I+1)
           TBI = ZPDP / ZDMP
           TAIM = ZMDM / ZDPM
           RLM = RL
           RL = TABL(5,I)
C
           F(I) = ZDPI * ZDPI * (2. *ZMDI +TBI * ZDPI) * RLM
     +          + ZDMI * ZDMI * (TAIM * ZDMI + 2. * ZPDI) * RL
C
           IF (I .NE. 2) FPRIME(1,I-1)=ZDMI*TABL(4,I-1)*(ZDMI/ZDPM)**2*RL
C
           IF (I .NE. N-1) FPRIME(3,I)=ZDPI*TABL(4,I+1)*(ZDPI/ZDMP)**2*RLM
C
           FPRIME(2,I) = RLM*ZDPI * ((2.+3.*DI*TBI)*ZDPI + 4.*DI*ZMDI)
     +                 + RL*ZDMI * ((2.-3.*DI*TAIM)*ZDMI - 4.*DI*ZPDI)
10      CONTINUE
        RETURN
        END
```

```
      FUNCTION DFLIP(Z,TABL,NM1,EPS)
      REAL TABL(5,1),Z(1)
      RL = TABL(5,1)
      RLP = TABL(5,2)
      TABL(4,NM1+1) = 0.
      EPS2 = 2.*EPS
      GAMMA = 0.
      C2 = 0.
      DI = 0.
      DIP = TABL(3,2)
      V1 = 1. - (Z(2) - EPS2) * DIP
      V2 = 1. - (Z(2) + EPS2) * DIP
      R1P = AMAX1(V1,V2)
      R2 = 1. / AMIN1(V1,V2)
      Z1 = Z(1)
      Z2 = Z1
      Z1P = Z(2) - EPS2
      Z2P = Z(2) + EPS2
      DO 10 I=2,NM1
         RLM = RL
         RL = RLP
         RLP = TABL(5,I+1)
         DIM = DI
         DI = DIP
         DIP = TABL(3,I+1)
         R1 = R1P
         R2M = R2
         Z1M = Z1
         Z2M = Z2
         Z1 = Z1P
         Z2 = Z2P
         Z1P = Z(I+1) - EPS2
         Z2P = Z(I+1) + EPS2
         V1P = 1. - Z1P * DIP
         V2P = 1. - Z2P * DIP
         V1 = 1. + Z1 * DI
         V2 = 1. + Z2 * DI
         R1P = AMAX1(V1P,V2P) / AMIN1(V1,V2)
         R2 = AMAX1(V1,V2) / AMIN1(V1P,V2P)
         C1 = R1P * R1P * RLP * TABL(4,I)
         ALIPP = 4. * C1 * DI * R1P
         CLIPP = 6. * C1 * DIP
         BLIPM = 4. * C2 * DI * R2M
         C2 = R2 * R2 * RLM * TABL(4,I+1)
         DLIP = 6. * C2 * DI
         DI2 = DI * DI
         A1 = (RLM - RL) * (2. - DI2)
         A2 = 3. * (RLM + RL) * DI
         ELIP = DI2 * (12.*(AMAX1(ABS(Z1P+DIP),ABS(Z2P+DIP))*RLM*R2
     +                    +  AMAX1(ABS(Z1M-DIM),ABS(Z2M-DIM))*RL*R1)
     +                    +  8.*AMAX1(ABS(A1+A2*Z1),ABS(A1+A2*Z2)))
         GAMMA = AMAX1(GAMMA,BLIPM+ELIP+ALIPP,DLIP+CLIPP)
10    CONTINUE
      DFLIP = GAMMA
      RETURN
      END
      SUBROUTINE TRIDG(D,M,B,IFLAG)
      DIMENSION D(3,1),B(1)
      DATA E/.0000001/
      IF (M .EQ. 1) GOTO 30
      DO 10 I=1,M-1
         IF (IFLAG .GT. 0) GOTO 5
         IF (ABS(D(2,I)) .LT. E) GOTO 999
```

```
            D(1,I) = -D(1,I) / D(2,I)
            D(2,I+1) = D(2,I+1) + D(1,I)*D(3,I)
5           B(I+1) = B(I+1) + D(1,I)*B(I)
10      CONTINUE
C
        DO 20 I=1,M-1
            J=M-I
            IF (IFLAG .GT. 0) GOTO 15
            IF (ABS(D(2,J+1)) .LT. E) GOTO 999
            D(3,J) = -D(3,J) / D(2,J+1)
15          B(J) = B(J) + D(3,J) * B(J+1)
            B(J+1) = B(J+1) / D(2,J+1)
20      CONTINUE
C
30      B(1) = B(1) / D(2,1)
        RETURN
C
999     IFLAG = -1
        RETURN
        END
        FUNCTION DINV(D,M)
        DIMENSION D(3,1)
        RNORM = 1.
        QSNORM = 1. / ABS(D(2,1))
        IF (M .EQ. 1) GOTO 40
        RCOL = 1.
        QSCOL = QSNORM
        DO 30 I=1,M-1
            QSCOL = QSCOL * ABS(D(3,I)) + 1./ABS(D(2,I+1))
            RCOL = RNORM * ABS(D(1,M-I)) + 1.
            RNORM = AMAX1(RNORM,RCOL)
            QSNORM = AMAX1(QSNORM,QSCOL)
30      CONTINUE
40      DINV = RNORM * QSNORM
        RETURN
        END
```

# Section II
# WFCMPR

## Program WFCMPR

**Purpose:** The program WFCMPR compares two Wilson-Fowler splines with common nodes. It reports the maximum distance between curves and the maximum difference of their tangents (or normals), both computed along the entire length of the splines. Distance is measured perpendicular to segments.

**Language/Libraries:** Program WFCMPR is written in Fortran 77. It is self-contained, requiring no libraries.

**Inputs** (To be read from a file):

Spline nodes (common to both splines):

- Number of spline nodes
- (X,Y) values of nodes.

First spline:

- Start and end local tangents for each segment.

Second spline:

- Start and end local tangents for each segment.

**Outputs** (To the terminal screen):

- Maximum difference between splines.
- Maximum deviation of spline normals.

**Method:**

The algorithms for computing the difference and normal deviation between splines were provided by J. D. Emery [3]. Both use a difference curve, which is the piecewise cubic representing the difference between the two splines in a direction normal to the segment orientation.

One of several equivalent methods of describing a Wilson-Fowler spline is to specify the nodes and the local tangents (ie., endpoint tangent angles for each segment measured with respect to the local segment coordinate system). This form is used because it is easy to obtain from most CAD/CAM systems and it facilitates computation of the difference curve.

Function WFNORM finds the maximum absolute value of the difference curve over all segments. Within each segment, the maximum occurs either at the midpoint or at one of the roots of the derivative of the difference curve.

Function WFDNRM finds the maximum absolute value of the derivative of the difference curve over all segments. Within each segment, the maximum occurs either at one of the endpoints or at the root of the second derivative of the difference curve.

**Limitations:**

The following limitations are explicit in PARAMETER statements at the beginning of the program:

- NNODES (maximum number of spline nodes) = 100
- XSMALL (smallest number used to prevent machine underflow) = 1E-35

The following limitation is explicit in a character declaration statement near the beginning of the program:

- filename length = 12 characters

```
$ run wfcmpr
    -----------> PROGRAM WFCMPR <-----------
 Compares two Wilson_Fowler splines interpolating
          the same set of points.
 Enter data filename:
testcmpr
Maximum difference = 0.28747765463777E-03
Maximum diff of normals (degrees) = 0.49119293689728E-01
FORTRAN STOP
$
```

WFCMPR Sample Run.

```
          5
  5.000000        0.0000000E+00
  4.000000         1.000000
  3.000000         4.000000
  1.000000         5.000000
 0.0000000E+00     5.000000
  0.1428571       -0.2385142
  0.2336244        0.2775339
 -0.5655161        0.3745124
 -0.1056955        0.1428573
  0.142           -0.238
  0.233            0.277
 -0.565            0.374
 -0.105            0.142
```

Input File for WFCMPR Sample Run.

```
      program wfcmpr
      parameter (nnodes=100, xsmall=1E-35)
      dimension px(nnodes), py(nnodes), dtana(nnodes), dtanb(nnodes)
      character*12 fname
      data rad2dg / 57.29577951 /
      data zero / xsmall /
      common /machin/ zero
c
c     Program to compare two Wilson-Fowler splines, based on an
c     algorithm by J. D. Emery, Bendix Kansas City.  Requires
c     as input the spline nodes and local tangents of each spline
c     for each segment.  Returns (1) the maximum difference between
c     the two splines measured normal to each segment, and (2) the
c     maximum difference of the tangents (or normals) to the splines
c     along the entire length.
c
c     This program is written in Fortran77.  All input/output
c     occurs in the main program.
c
c     Any questions should be refered to:
c           Sharon Fletcher, Division 7611
c           Sandia National Laboratories
c           (505) 846-5506
c
c     INPUTS:
c     (unit 2): np = number of spline nodes
c               {px(i), py(i)} = i-th node;
c                                one pair per line; i=1,np
c               {ta1(j), tb1(j)} = spline 1 j-th segment local tangents;
c                                  one pair per line; j=1,np-1
c               {ta2(j), tb2(j)} = spline 2 j-th segment local tangents;
c                                  one pair per line; j=1,np-1
c
c     OUTPUTS:
c     (terminal): an = max difference between splines
c                 dn = max deviation of spline normals
c
c
      print *, '    ----------> PROGRAM WFCMPR <----------    '
      print *, ' Compares two Wilson-Fowler splines interpolating'
      print *, '          the same set of points.'
      print *, ' Enter data filenmame:'
      read (*,101) fname
      open (unit=2, file=fname, status='old')
c
c     Read spline nodes.
c
      read (2,*) np
      nseg = np - 1
      do 10 i=1,np
         read (2,*) px(i), py(i)
10    continue
c
c     Read local tangents for both splines; need only keep
c     difference between them.
c
      do 20 i=1,nseg
         read (2,*) dtana(i), dtanb(i)
20    continue
      do 30 i=1,nseg
         read (2,*) ta2, tb2
         dtana(i) = dtana(i) - ta2
         dtanb(i) = dtanb(i) - tb2
```

```
30      continue
c
        an = wfnorm(px, py, dtana, dtanb, np)
        dn = wfdnrm(px, py, dtana, dtanb, np)
        dn = atan(dn) * rad2dg
        print 100, an, dn
        close(unit=2)
        stop
100     format(' Maximum difference =',G22.14,/,
     +         ' Max diff of normals (degrees) =',G22.14)
101     format(A)
        end
        function wfnorm(px,py,a,b,n)
        dimension px(1), py(1), a(1), b(1)
        common /machin/ zero
        f(x) = x * (x * (x * alpha + beta) + gamma)
c
c       INPUTS:
c          px(i) = i-th spline node x-coordinate
c          py(i) = i-th spline node y-coordinate
c          a(i)  = i-th spline segment starting local tangent
c          b(i)  = i-th spline segment ending local tangent
c          n     = number of nodes
c
c       OUTPUT:
c          wfnorm = norm of curve (maximum absolute value)
c
c       Calculates norm of the wf-curve described by {px, py, a, b}.
c       Note that the main program has actually passed a
c       difference curve, ie., the difference between two splines.
c
        wfnorm = 0.
        nseg = n-1
        do 10 i=1,nseg
           al = sqrt( (px(i+1)-px(i))**2 + (py(i+1)-py(i))**2 )
           alpha = (a(i) + b(i)) / (al*al)
           beta = - (2. * a(i) + b(i)) / al
           gamma = a(i)
           d = sqrt( beta * beta - 3. * alpha * gamma)
           ul = al / 2.
           u2 = ul
           if (abs(alpha) .gt. zero) then
              u = (- beta + d) / (3. * alpha)
                 if ( (u .ge. zero) .and. (u .le. al) ) ul = u
              u = (- beta - d) / (3. * alpha)
                 if ( (u .ge. zero) .and. (u. le. al) ) u2 = u
           endif
           wfnorm = amax1( wfnorm, abs(f(ul)), abs(f(u2)) )
10      continue
        return
        end
        function wfdnrm(px,py,a,b,n)
        dimension px(1), py(1), a(1), b(1)
        common /machin/ zero
        df(x) = x * (x * 3. * alpha + 2. * beta) + gamma
c
c       INPUTS:
c          px(i) = i-th spline node x-coordinate
c          py(i) = i-th spline node y-coordinate
c          a(i)  = i-th spline segment starting local tangent
c          b(i)  = i-th spline segment ending local tangent
c          n     = number of nodes
c
```

```
c       OUTPUT:
c          wfdnrm = norm of derivative of curve (maximum absolute
c                   value of tangent)
c
c       Calculates norm of the derivative of the wf-curve described
c       by {px, py, a, b}.  Note that the main program has actually
c       passed a difference curve, the difference between two splines.
c
        wfdnrm = 0.
        nseg = n-1
        do 10 i=1,nseg
           al = sqrt( (px(i+1)-px(i))**2 + (py(i+1)-py(i))**2 )
           alpha = (a(i) + b(i)) / (al*al)
           beta = - (2. * a(i) + b(i)) / al
           gamma = a(i)
           rho = 0.
           if (abs(alpha) .gt. zero) then
              u = - beta / (3. * alpha)
                 if ((u .gt. zero) .and. (u .lt. al))  rho = abs( df(u) )
           endif
           wfdnrm = amax1( wfdnrm, rho, abs(df(0.)), abs(df(al)) )
10      continue
        return
        end
```

# Section III
# WFAPPX

# Program WFAPPX

**Purpose:** The program WFAPPX compares a piecewise linear curve to a Wilson-Fowler spline. It reports the maximum deviation between these two curves, and the parameter value on the spline where it occurs.

**Language/Libraries:** Program WFAPPX is written in Fortran 77. It is self-contained, requiring no libraries. Function DPNTLC and subroutine WFEVAL are also used by program SPLPKG, and are copied.

**Inputs** (To be read from a file):

Wilson-Fowler spline definition:

- Number of nodes
- (X,Y) values of nodes
- Start and end local tangents for each segment.

Piecewise linear curve:

- Number of breakpoints
- (X,Y) values of breakpoints.

**Outputs** (To the terminal screen):

- Maximum difference between spline and piecewise linear curve.
- Spline parameter value (chord length) where maximum difference occurs.

**Method:**

The spline definition is requested to be input in terms of local tangents (ie., endpoint tangent angles with respect to the local coordinate system for each segment) because this is form is easily obtained from most CAD/CAM systems.

The algorithm for computing the distance from a point (on the spline) to a piecewise linear curve was provided by J. D. Emery [3].

WFAPPX computes this distance for each of NEVAL points on the spline, and reports the maximum of these. The points are equally spaced along the total chord length of the spline.

**Limitations:**

The following limitations are explicit in PARAMETER statements at the beginning of the program:

- NNODES (maximum number of spline nodes) = 100
- NBKPNT (maximum number of linear curve breakpoints) = 1000
- NEVAL (maximum number of evaluation points along the spline) = 1000
- XSMALL (smallest number used to prevent machine underflow) = 1E-35
- XBIG (largest number used to prevent machine overflow) = 1E+35

The following limitation is explicit in a character declaration statement near the beginning of the program:

- filename length = 12 characters

**Operational Notes:**

This program does not require that any of the breakpoints of the linear curve actually fall on the spline. Thus at each point of comparison, the entire linear approximating curve must be examined. This is very computation intensive, and will take significant time for a large number of breakpoints.

```
$ run wfappx
    ----------> PROGRAM WFAPPX <----------
 Compares a Wilson-Fowler spline and an approximating
            piecewise linear curve.
 Enter data filename:
testappx
 Maximum deviation is    5.0452072E-02
 at parameter value of    0.2017610
FORTRAN STOP
$
```

WFAPPX Sample Run.

```
         5
  5.000000        0.000000
  4.000000        1.000000
  3.000000        4.000000
  1.000000        5.000000
  0.000000        5.000000
 0.1428571       -0.2385142
 0.2336244        0.2775339
-0.5655161        0.3745124
-0.1056955        0.1428573
        10
 0.5000000E+01  0.0000000E+00
 0.4338405E+01  0.5660306E+00
 0.3848392E+01  0.1288788E+01
 0.3594636E+01  0.2119220E+01
 0.3416715E+01  0.2974932E+01
 0.3124311E+01  0.3792482E+01
 0.2538533E+01  0.4489997E+01
 0.1754998E+01  0.4863973E+01
 0.8680618E+00  0.5012667E+01
-0.2384186E-06  0.5000000E+01
```

**Input File for WFAPPX Sample Run.**

```
      program wfappx
      parameter (nnodes=100, nbkpnt=1000, neval=1000)
      parameter (xsmall=1E-35, xbig=1E+35)
      dimension tabl(5,nnodes), x(nbkpnt), y(nbkpnt)
      character*12 fname
      common /machin/ zero, big
      data zero, big / xsmall, xbig /
c
c     Program to find the maximum deviation between a Wilson-Fowler
c     spline and a piecewise linear curve, based on an algorithm by
c     J. D. Emery, Bendix Kansas City.
c     Measurement is made at 1000 points along the spline.
c     The spline definition is input as a set of nodes and local
c     tangents for each segment.  The piecewise linear curve is
c     input as a set of coordinate points.  Output is the maximum
c     deviation and the spline parameter value where it occurs.
c
c     This program is written in Fortran77.  All input/output
c     occurs in the main program.
c
c     Any questions should be refered to:
c             Sharon Fletcher, Division 7611
c             Sandia National Laboratories
c             (505) 846-5506
c
c     INPUTS:
c     (unit 2): np = number of spline nodes
c               {tabl(1,i), tabl(2,i)} = i-th node;
c                                 one pair per line; i=1,np
c               {tabl(3,j), tabl(4,j)} = j-th segment local tangents;
c                                  one pair per line; j=1,np-1
c               nk = number of points on piecewise curve
c               {x(k), y(k)}     = k-th point on piecewise curve;
c                                  one pair per line; k=1,nk
c
c     OUTPUTS:
c     (terminal): dex = max difference between curves
c                 tex = parameter value where max diff occurs
c
c
c
      print *, '    -----------> PROGRAM WFAPPX <-----------'
      print *, ' Compares a Wilson-Fowler spline and an approximating'
      print *, '              piecewise linear curve.'
      print *, ' Enter data filename:'
      read (*,100) fname
      open (unit=2, file=fname, status='old')
c
c     Read spline nodes.
c
      read (2,*) np
      nseg = np - 1
      do 10 i=1,np
         read (2,*) tabl(1,i), tabl(2,i)
10    continue
c
c     Read local tangents for the spline.
c     Compute segment lengths and total length.
c
      totl = 0.
      do 20 i=1,nseg
         read (2,*) tabl(3,i), tabl(4,i)
         tabl(5,i) = sqrt((tabl(1,i)-tabl(1,i+1))**2 +
```

```
     +                        (tabl(2,i)-tabl(2,i+1))**2)
           totl = totl + tabl(5,i)
20       continue
c
c        Read the approximating piecewise linear curve.
c
         read (2,*) nk
         do 30 i=1,nk
            read (2,*) x(i), y(i)
30       continue
c
c        Evaluate the spline at each of NEVAL points (=1000); find
c        distance from spline point to approximating piecewise curve;
c        keep maximum distance.
c
         dex = 0.
         tex = 0.
         ninc = neval - 1
         tinc = totl / float(ninc)
         do 50 i=0,ninc
            t = i * tinc
            call wfeval(np,tabl,t,u,s)
            dist = dpntlc(nk,x,y,u,s)
            if (dist .gt. dex) then
               dex = dist
               tex = t
            endif
50       continue
c
         print *, ' Maximum deviation is ', dex
         print *, ' at parameter value of ', tex
         close (unit=2)
         stop
100      format(A)
         end
         function dpntlc(nk,x,y,u,s)
         dimension x(1), y(1)
         common /machin/ zero, big
c
c        INPUTS:
c           nk = number of points (piecewise linear curve)
c           x(i) = i-th curve point x coordinate
c           y(i) = i-th curve point y coordinate
c           u = x coordinate of measurement point
c           s = y coordinate of measurement point
c
c        OUTPUT:
c           dpntlc = distance from the point (u,s) to the curve
c
c        Finds minimum distance from a point to a piecewise linear
c        curve defined by its breakpoints {x,y}.  All segments of
c        the linear curve are tested.
c
         m = nk - 1
         dpntlc = big
         do 10 i=1,m
            a = sqrt( (x( i )-u)**2 + (y( i )-s)**2 )
            b = sqrt( (x(i+1)-u)**2 + (y(i+1)-s)**2 )
               if (abs(a*b) .lt. zero) goto 50
            c = sqrt( (x(i+1)-x(i))**2 + (y(i+1)-y(i))**2 )
            cosa = ( b*b + c*c - a*a) / (2.*b*c)
            cosb = ( a*a + c*c - b*b) / (2.*a*c)
            if (cosa*cosb .ge. 0.) then
```

```
            e = b * sqrt(abs( 1-cosa**2 ))
         else
            e = aminl(a,b)
         endif
         if (e .lt. dpntlc) dpntlc = e
10    continue
      return
50    dpntlc = 0.
      return
      end
      SUBROUTINE WFEVAL(N,TABL,ULOCL,U,S)
      DIMENSION TABL(5,100)
C
C     INPUTS:
C        N = NUMBER OF POINTS (SPLINE NODES)
C        TABL(1,I) = I-TH POINT X COORDINATE
C        TABL(2,I) = I-TH POINT Y COORDINATE
C        TABL(3,I) = I-TH SEGMENT ENTRY ANGLE TANGENT
C        TABL(4,I) = I-TH SEGMENT EXIT ANGLE TANGENT
C        TABL(5,I) = I-TH SEGMENT LENGTH
C        ULOCL = A VALUE ALONG THE CHORD LENGTH FOR INTERPOLATION
C
C     OUTPUTS:
C        U = GLOBAL COORDINATES X VALUE OF INTERPOLATED POINT
C        S = GLOBAL COORDINATES Y VALUE OF INTERPOLATED POINT
C
C     INTERPOLATE TO THE SPLINE AT ULOCL IN TERMS OF THE LOCAL
C     COORDINATE SYSTEM, THEN ROTATE THE POINT TO THE GLOBAL
C     COORDINATE SYSTEM.
C
C     (DETERMINE WHICH SEGMENT ULOCL IS IN -- J-TH SEGMENT)
C
      J = 1
5     IF (ULOCL .LT. TABL(5,J) .OR. J .EQ. N-1) GOTO 10
         ULOCL = ULOCL - TABL(5,J)
         J = J + 1
         GOTO 5
10    CONTINUE
C
C     (EVALUATE HERMITE FORM OF CUBIC)
C
      UMT = ULOCL - TABL(5,J)
      SLOCL = (TABL(3,J) * ULOCL * UMT**2 +
     +           TABL(4,J) * ULOCL**2 * UMT) / TABL(5,J)**2
C
C     (ROTATE AND TRANSLATE LOCAL POINT (ULOCL,SLOCL) TO
C      GLOBAL COORDINATE SYSTEM (U,S) )
C
      TH = ATAN2(TABL(2,J+1)-TABL(2,J),TABL(1,J+1)-TABL(1,J))
      U = ULOCL * COS(TH) - SLOCL * SIN(TH) + TABL(1,J)
      S = ULOCL * SIN(TH) + SLOCL * COS(TH) + TABL(2,J)
      RETURN
      END
```

References

**1 Fletcher, S. K.**

Recommended Practices for Spline Usage in CAD/CAM Systems CADCAM-007, SAND84-0142, Sandia National Laboratories, Albuquerque, NM, April 1984.

**2 Melvin, W. R.**

Error Analysis and Uniqueness Properties of the Wilson-Fowler Spline, LA-9178, Los Alamos National Laboratory, Los Alamos, NM, Aug 1982.

**3 Emery, J. D.**

Personal communications. Bendix Kansas City, Kansas City, KS, 1983.

**4 Fowler, A. H. and C. W. Wilson**

Cubic Spline, A Curve Fitting Routine, Y-12 Plant Report Y-1400 (Revision 1), Union Carbide Corporation, Oak Ridge, TN, Jun 1966.

**Distribution:**

The Bendix Corporation
Kansas City Division
P.O. Box 1159
Kansas City, MO 64141
Attn: Charlie Mentesana, 2D39
Attn: James Emery, MC45
Attn: Charles R. Miller, MC45

General Electric Company
Neutron Devices Department
P.O. Box 2908
Largo, FL 34294
Attn: Virginia McCauley

Mason & Hanger
Pantex Plant
P.O. Box 30020
Amarillo, TX 79177
Attn: Dean Carpenter

Monsanto Research Corporation
Mound Laboratory
P.O. Box 32
Miamisburg, OH 45342
Attn: David Michaels

Rockwell International
Atomics International Division
Rocky Flats Plant
P.O. Box 464
Golden, CO 80401
Attn: Jack Doyle, T881B
Attn: Maryann Gaug, T881B
Attn: M. F. Schweitzer, B750
Attn: Leroy Mellecker

Martin Marietta Energy Systems
Y-12 Plant
P.O. Box Y
Oak Ridge, TN 37831
Attn: Al Stephens, Bldg 9103 MS 3
Attn: Jim Snyder, Bldg 9111 MS 2
Attn: C. W. Wilson
Attn: Robert Easterday

Lawrence Livermore National Lab
P.O. Box 808
Livermore, CA 94550
Attn: Stan Trost, L-125
Attn: John Martin, L-125
Attn: Fred Fritsch, L-316

Los Alamos National Laboratory
Attn: Ray Elliot, B272
Attn: Ron Dolan, C931

DOE/AL
Attn: H. T. Season, Jr.

**Distribution:**

Sandia Internal:

2800 H. W. Schmitt
2810 D. W. Doak
2811 J. F. Jones, Jr.
2811 S. K. Fletcher (30)
2812 G. R. Urish
2813 T. M. Schultheis
2814 P. A. Erickson
3141 C. Ostrander (5)
3151 W. L. Garner (3)
3154-3 C. H. Dalin (28)
For DOE/TIC
8024 M. A. Pound