UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Computer and Hand-calculator Programs to Determine Extension or Contraction of Faulted Marker Planes

P. T. Delaney [1]

M. D. Jackson [2]

Open-file Report 85-107

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards and stratigraphic nomenclature.

[1] U.S. Geological Survey
2255 N. Gemini Dr.
Flagstaff, Arizona, 86001

[2] Department of Earth and Planetary Sciences
Johns Hopkins University
Baltimore, Maryland, 21218

1984

ABSTRACT

Programs are presented which calculate angular relations among fault attitude, marker-plane attitude and slip direction, and determine the extension or contraction of the marker by the fault slip. The programs are written both in FORTRAN and in the reverse-Polish language implemented on Hewlett-Packard type hand calculators. Examples of program use are presented.

INTRODUCTION

It has been shown by Jackson and Delaney (1985) that certain angular relations among a marker plane, a fault and the direction of fault slip provide a measure of the extensional or contractional separation of a marker plane. These angles are: ϕ, the angle between the downward directed normals to the fault and marker planes; θ, the angle from the intersection of the fault and marker planes to the slip direction; and ω, the angle between the direction of maximum contraction and the direction of slip. The first two or the last of these angles can be used to ascertain the extension or contraction of the marker plane in a direction normal to the line of intersection of the fault and marker. This paper presents a program written in FORTRAN for computers, and a program written in a reverse-Polish language for Hewlett-Packard calculators, to calculate these angular relations.

The method of calculating the angle θ from the null to the slip directions used by Jackson and Delaney (1985) can result in divide-by-zero conditions. This situation is treated by calculating the magnitude of θ from the dot-product relation $\cos |\theta| = |\vec{N}\cdot\vec{S}|$, where $\vec{N}$ and $\vec{S}$ are the null and slip vectors, respectively, and the vertical bars denote absolute values of the enclosed quantity. The sign of the angle θ is positive if, looking at the hanging-wall surface of the fault, the angle from the intersection with the marker to the slip direction is drawn in a clockwise direction. This is equivalent to finding whether the direction given by the cross product relation $\vec{N} \times \vec{S}$ is in the same direction (positive θ) or opposite direction (negative θ) as the downwarded directed normal to the fault plane, $\vec{F}$.

THE FORTRAN PROGRAM

The FORTRAN program employs the "structured branching options" (see Meissener and Organick, 1980, p. 480) available in ANSI FORTRAN 77 implementations. The program is thoroughly documented and follows the same notation as used by Jackson and Delaney (1985). The program (called FAULT) calls six subroutines: DDDTDC, which converts dip-direction and dip data to direction-cosine data; TPTDC, which converts trend and plunge data to direction-cosine data; DCTTP, which converts direction-cosine data to trend and plunge data; CROSS, which computes the vector cross product; NORM, which normalizes a vector to unit length; and DOT, which computes the vector dot product. The program prompts the user for input at the keyboard, and displays output on a CRT or line printer. It is therefore assumed that the standard input and output devices are interactive, and denoted by the logical unit numbers 5 and 6, respectively. Listings are given in Appendix A.

It is assumed that no field measurement is more accurate than 1^o; if the angle ϕ is within 1^o of 90^o or the angle θ is within 1^o of 0^o, then the fault slip is identified as "null"-- it neither extended nor contracted the marker. In addition, the program calculates the angle between the slip direction and the fault plane. By definition, this angle should be zero. If it is found to be greater than 1^o, it is used to determine the limits of null faulting.

The example below illustrates the use of the FORTRAN program. User-supplied entries are underlined.

```
RUN FAULT

      MARKER:  Dip Direction and dip (deg) = ?? =
317 20
       FAULT:  Dip Direction and Dip (deg) = ?? =
180 45
 SLICKENLINE:  Trend and Plunge (deg) = ?? =
92 2
      OFFSET:  Hanging Wall UP (U), DOWN (D),
                 or NO (N) apparent offset = ?? =
D

 92.    2. = Trend, Plunge of Hanging Wall SLIP DIRECTION
        0. = Eta, ANGLE of the SLIP VECTOR from the FAULT PLANE
       61. = Phi, ANGLE between MARKER and FAULT PLANES
101.   11. = Trend, Plunge of NULL DIRECTION
      -18. = Theta, ANGLE from NULL to SLIP DIRECTION
       99. = Omega, ANGLE from PURE-CONTRACTION DIRECTION to SLIP DIRECTION

                        EXTENSION FAULT

Another Fault ?? (Y/N)
Y
MARKER PLANE:  Dip Direction and Dip =  317.  20.
       FAULT:  Dip Direction and Dip (deg) = ?? =
180 45
 SLICKENLINE:  Trend and Plunge (deg) = ?? =
92 2
      OFFSET:  Hanging Wall UP (U), DOWN (D),
                 or NO (N) apparent offset = ?? =
U

272.   -2. = Trend, Plunge of Hanging Wall SLIP DIRECTION
        0. = Eta, ANGLE of the SLIP VECTOR from the FAULT PLANE
       61. = Phi, ANGLE between MARKER and FAULT PLANES
101.   11. = Trend, Plunge of NULL DIRECTION
       18. = Theta, ANGLE from NULL to SLIP DIRECTION
       81. = Omega, ANGLE from PURE-CONTRACTION DIRECTION to SLIP DIRECTION
```

CONTRACTION FAULT

```
Another Fault ?? (Y/N)
N
Another Marker?? (Y/N)
Y
      MARKER:  Dip Direction and dip (deg) = ?? =
135 70
       FAULT:  Dip Direction and Dip (deg) = ?? =
180 45
 SLICKENLINE:  Trend and Plunge (deg) = ?? =
92 2
      OFFSET:  Hanging Wall UP (U), DOWN (D),
                 or NO (N) apparent offset = ?? =
D

 92.    2. = Trend, Plunge of Hanging Wall SLIP DIRECTION
        0. = Eta, ANGLE of the SLIP VECTOR from the FAULT PLANE
       45. = Phi, ANGLE between MARKER and FAULT PLANES
 26. -42. = Trend, Plunge of NULL DIRECTION
206.  42. =                   in Lower Hemisphere
       74. = Theta, ANGLE from NULL to SLIP DIRECTION
       47. = Omega, ANGLE from PURE-CONTRACTION DIRECTION to SLIP DIRECTION
```

CONTRACTION FAULT

```
Another Fault ?? (Y/N)
N
Another Marker?? (Y/N)
Y
      MARKER:  Dip Direction and dip (deg) = ?? =
45 45
       FAULT:  Dip Direction and Dip (deg) = ?? =
225 45
 SLICKENLINE:  Trend and Plunge (deg) = ?? =
225 45
      OFFSET:  Hanging Wall UP (U), DOWN (D),
                 or NO (N) apparent offset = ?? =
D

225.  45. = Trend, Plunge of Hanging Wall SLIP DIRECTION
       0. = Eta, ANGLE of the SLIP VECTOR from the FAULT PLANE
      90. = Phi, ANGLE between MARKER and FAULT PLANES
-45.   0. = Trend, Plunge of NULL DIRECTION
135.   0. =                   in Lower Hemisphere
```

HAND-CALCULATOR PROGRAM

The calculation, written for a Hewlett-Packard 15-C calculator, requires 20 storage registers and about 200 lines of instructions. Parameters stored in the registers are shown in Table 1. A listing is given in Appendix B. For brevity, only the angle ϕ between the downward-directed marker- and fault-plane normals, and the angle θ from the null direction to the slip direction, are calculated. Extension or contraction can be determined from these angles (Table 2). There are five programs and three subroutines. The first program uses the dip direction and dip of the marker plane to calculate and store the direction cosines of the downward-directed normals to that plane. The second program performs the same task for the fault plane. For faults with hanging-wall-down displacements, the third program uses the trend and plunge of the

TABLE 1: Contents of storage registers

REGISTER #	CONTENTS	
0	l^{marker}	l, m, n are cosines of angles that
1	m^{marker}	superscripted vector makes with
2	n^{marker}	the South, East and Up directions.
3	l^{fault}	
4	m^{fault}	
5	n^{fault}	
6	l^{slip}	
7	m^{slip}	
8	n^{slip}	
9	ϕ, angle between downward-directed marker- and fault-normals	
.0	θ, angle from null to slip direction	
.1 - .9	miscellaneous	

Table 2: Extension or Contraction from Angles ϕ and θ

	$0^\circ \leq \phi < 90^\circ$	$\phi = 0^\circ$	$90^\circ < \phi \leq 180^\circ$
$-90^\circ \leq \theta < 0^\circ$	Extend	Null	Contract
$0^\circ = \theta$	Null	Null	Null
$0^\circ < \theta \leq 90^\circ$	Contract	Null	Extend

slip direction to calculate and store the equivalent direction cosines. For faults with hanging-wall-up displacements, the fourth program performs the same calculation as the third. The fifth program calculates, displays and stores the angles ϕ and θ.

In the following, the instructions, input keystrokes and the output are shown for the first and second examples given above. Underlined numbers refer to the values of the parameters for the particular example.

STEP	INSTRUCTIONS	INPUT	KEYSTROKE	DISPLAY
1.	Input marker-plane data			
	a. Marker dip direction (clockwise degrees from North)	317		
			ENTER	
	b. Marker dip (degrees down from horizontal)	20		
	c. Run program A		f A	
2.	Input fault-plane data			
	a. Fault dip direction (clockwise degrees from North)	180		
			ENTER	
	d. Fault dip (degrees down from horizontal)	45		
	c. Run program B		f B	
3.	Input slip-direction data			
	a. Slickenline trend (clockwise degrees from North)	92		
			ENTER	
	b. Slickenline plunge (degrees down from horizontal)	2		
	c. Sense of offset Run program C for hanging wall down displacements		f C	

STEP	INSTRUCTIONS	INPUT	KEYSTROKE	DISPLAY
4.	Calculate ϕ and θ			
	Run program E		f E	
				61 (= ϕ)
				-18 (= θ)

For the same marker-plane and fault-plane attitudes, it is not necessary to re-enter these orientations to enter a different slickenline. Similarily, it is not necessary to re-enter the attitude of the marker for new fault and slickenline attitudes. For hanging-wall-up fault slips program D is run instead of program C:

STEP	INSTRUCTIONS	INPUT	KEYSTROKE	DISPLAY
5.	Input slip-direction data			
	a. Slickenline trend			
	(clockwise degrees from North)	92		
			ENTER	
	b. Slickenline plunge			
	(degrees down from horizontal)	2		
	c. Sense of offset			
	Run program D for			
	hanging wall up displacements		f D	
6.	Calculate ϕ and θ			
	Run program E		f E	
				61 (= ϕ)
				18 (= θ)

REFERENCES

Jackson, M.D., and P.T. Delaney, 1985, Extension and contraction of faulted marker planes: Geology, to be submitted. 16 ms. pages, 5 figures, 2 tables.

Meissner, L.P., and E.I. Organick, 1980, Fortran 77: Featuring Structured Programming, Addison-Wesley, Reading, Mass., 500 p.

APPENDIX A: FORTRAN PROGRAM

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      PROGRAM FAULT
C
C  Input: (1) Marker-plane Dip Direction and Dip
C         (2) Fault-plane Dip Direction and Dip
C         (3) Slickenline Trend and Plunge
C         (4) Sense of Offset
C
C  Output: (1) Slip Direction of Hanging Wall relative to Footwall
C          (2) Angle between Slip Direction and Fault Plane (ideally, this
C                should be equal to zero
C          (3) Angle from Marker Plane to Fault Plane
C          (4) Null Direction
C          (4) Angle from Null to Slip Direction
C          (5) Angle from direction of maximum contraction to Slip Direction
C          (6) Determination as to whether the Fault Extends, Contracts,
C                or has a Null effect on the Marker
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      EXTERNAL DOT, CROSS, NORM, DDDTDC, TPTDC, DCTTP
      REAL*4 MD, MDC, ND, NDC, MNDC
      DIMENSION  MD(2), FD(2), SD(2), ND(2), MDC(3), FDC(3), SDC(3),
     .           NDC(3), FPDC(3), MNDC(3), XDC(3)
      CHARACTER*1 ANS
      COMMON /PIO180/ PI180
      PI180 = 3.141592654E0/180.0E0
C
C  Vectors of Length 2: (1) Dip Direction or Trend in Clockwise
C                           Degrees from North
C  (last letter of      (2) Dip or Plunge in Degrees Down from
C   name is 'D')            Horizontal
C
C  Vectors of Length 3: (1) South-pointed Direction Cosine
C                       (2) East-pointed Direction Cosine
C  (last two letters    (3) Up-pointed Direction Cosine
C   of name are 'DC')
C
C  OUTER LOOP POINT: Marker-plane Orientation
C
 10   WRITE (6,12)
 12   FORMAT (/'       MARKER:  Dip Direction and dip (deg) = ?? = ')
      READ (5,*) MD(1), MD(2)
      IF (MD(2) .LT. 1.0E-1) MD(1) = 0.0E0
      GOTO 14
 20   WRITE (6,22) MD(1), MD(2)
 22   FORMAT (/' MARKER PLANE:  Dip Direction and Dip = ',2(F5.0))
C
C  INNER LOOP POINT: Fault Data
C
 14   WRITE (6,24)
```

```
 24     FORMAT (/'         FAULT:  Dip Direction and Dip (deg) = ?? = ')
        READ (5,*) FD(1), FD(2)
        IF (FD(2) .LT. 1.0E-1) FD(1) = 0.0E0
        WRITE (6,26)
 26     FORMAT (/'  SLICKENLINE:  Trend and Plunge (deg) = ?? = ')
        READ (5,*) SD(1), SD(2)
        WRITE (6,28)
 28     FORMAT (/'        OFFSET:  Hanging Wall UP (U), DOWN (D) = ?? = ')
        READ (5,30) ANS
 30     FORMAT (A1)
C
C  Combine Slickenline and Sense-of-offset to get Slip Direction
C
        IF ((ANS .EQ. 'U') .OR. (ANS .EQ. 'u')) THEN
          SD(1) = 180.0E0 + SD(1)
          IF (SD(1) .GE. 360.0E0) SD(1) = SD(1) - 360.0E0
          SD(2) = -SD(2)
        ENDIF
        WRITE (6,32) SD(1), SD(2)
 32     FORMAT (//2(F5.0),
     .          ' = Trend, Plunge of Hanging Wall SLIP DIRECTION')
C
C  Dip Direction and Dip, and Trend and Plunge, To Direction Cosines
C
        CALL DDDTDC(MD,MDC)
        CALL DDDTDC(FD,FDC)
        CALL TPTDC(SD,SDC)
C
C  Calculate and write the Angle between Slip Vector and Fault Plane.
C  This angle indicates the accuracy of the field measurements.  It is
C  assumed that no measurement is better than 1 degree, or this angle,
C  whichever is greater.  Null Faulting arises if the appropriate
C  angles are within 1 degree or Phi degrees, whichever is greater, of
C  the true Null directions.
C
        CALL DOT(SDC,FDC,ETA)
        ETA = 90.0E0 - ACOS(ABS(ETA))/PI180
        IF (ETA .GT. 1.0E0) THEN
          DLIMIT = ETA
        ELSE
          DLIMIT = 1.0E0
        ENDIF
        WRITE (6,34) ETA
 34     FORMAT (/5X,F5.0,
     .    ' = Eta, ANGLE of the SLIP VECTOR from the FAULT PLANE')
C
C  Calculate and write Angle from Marker Plane to Fault Plane
C
        CALL DOT(MDC,FDC,PHI)
        PHI = ACOS(PHI)/PI180
        WRITE (6,36) PHI
 36     FORMAT (/5X,F5.0,' = Phi, ANGLE between MARKER and FAULT PLANES')
C
```

```
C  Calculate and write Null direction
C
      CALL CROSS(FDC,MDC,NDC)
      CALL NORM(NDC)
      CALL DCTTP(NDC,ND)
      WRITE (6,38) ND(1), ND(2)
 38   FORMAT (/2(F5.0),' = Trend, Plunge of NULL DIRECTION')
      IF (ND(2) .LT. 0.0E0) THEN
        ND(1) = ND(1) + 180.0E0
        ND(2) = -ND(2)
        IF (ND(1) .GE. 360.0E0) ND(1) = ND(1) - 360.0E0
        WRITE (6,40) ND(1), ND(2)
 40     FORMAT (2(F5.0),' =               in Lower Hemisphere')
      ENDIF
C
C  Calculate and write Theta Angle
C  The method used below is more cumbersome, but more robust, than that
C    described in the paper--there s no possibility for divide-by-zero
C    errors.  The two methods are entirely equivalent.
C
      CALL DOT(NDC,SDC,THETA)
      THETA = ACOS(ABS(THETA))/PI180
      IF (THETA .GT. DLIMIT) THEN
        CALL CROSS(NDC,SDC,FPDC)
        CALL NORM(FPDC)
        CALL DOT(FDC,FPDC,TEST)
        IF (TEST .LT. 0.0E0) THETA = -THETA
      ENDIF
      WRITE (6,50) THETA
 50   FORMAT (/5X,F5.0,
     .          ' = Theta, ANGLE from NULL to SLIP DIRECTION')
C
C  Calculate and write Omega angle
C
      CALL CROSS(MDC,NDC,XDC)
      CALL NORM(XDC)
      CALL DOT(SDC,XDC,OMEGA)
      OMEGA = ACOS(OMEGA)/PI180
      WRITE (6,60) OMEGA
 60   FORMAT (/5X,F5.0,' = Omega, ANGLE from PURE-CONTRACTION ',
     .    'DIRECTION to SLIP DIRECTION')
C
C  Calculate and write Type of Fault
C
C  If Phi < 90 deg, 0 deg < Theta <  90 deg -- Extension Fault
C                   0 deg > Theta > -90 deg -- Contraction Fault
C  If Phi > 90 deg, 0 deg > Theta >  90 deg -- Extension Fault
C                   0 deg < Theta < -90 deg -- Contraction Fault
C
C  If Phi = 90 deg or Theta = 0 deg -- Null Fault
C
      IF (PHI .LT. 90.0E0-DLIMIT) THEN
        IF (THETA .LT. -DLIMIT) THEN
          WRITE (6,70)
```

```
          ELSEIF (THETA .GT. DLIMIT) THEN
            WRITE (6,72)
          ELSE
            WRITE (6,74)
          ENDIF
        ELSEIF (PHI .GT. 90.0E0+DLIMIT) THEN
          IF (THETA .LT. -DLIMIT) THEN
            WRITE (6,72)
          ELSEIF (THETA .GT. DLIMIT) THEN
            WRITE (6,70)
          ELSE
            WRITE (6,74)
          ENDIF
        ELSE
          WRITE (6,76)
        ENDIF
 70     FORMAT (/'                        EXTENSION FAULT'/)
 72     FORMAT (/'                      CONTRACTION FAULT'/)
 74     FORMAT (/'  Slip is in Null Direction: NULL FAULT'/)
 76     FORMAT (/'  Fault is normal to Marker: NULL FAULT'/)
C
        WRITE (6,90)
 90     FORMAT (/'  Another Fault ?? (Y/N) ')
        READ (5,30) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) GOTO 20
C
        WRITE (6,92)
 92     FORMAT (/'  Another Marker?? (Y/N) ')
        READ (5,30) ANS
        IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) GOTO 10
C
        STOP
        END




C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
        SUBROUTINE DDDTDC(D,DC)
C
C  Napier's rules for spherical triangles to convert from Dip Direction
C    and Dip To Direction Cosines for a downward pointed plane normal
C
        IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
        DIMENSION D(2), DC(3)
        COMMON /PIO180/ PI180
C
        DC(1) = SIN(PI180*D(2))*COS(PI180*D(1))
        DC(2) = -SIN(PI180*D(2))*SIN(PI180*D(1))
        DC(3) = -COS(PI180*D(2))
C
        RETURN
        END
```

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      SUBROUTINE TPTDC(D,DC)
C
C  Napier's rules for spherical triangles to convert from Trend and
C    Plunge To Direction Cosines
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      DIMENSION D(2), DC(3)
      COMMON /PIO180/ PI180
C
      DC(1) = -COS(PI180*D(1))*COS(PI180*D(2))
      DC(2) = COS(PI180*D(2))*SIN(PI180*D(1))
      DC(3) = -SIN(PI180*D(2))
C
      RETURN
      END
```

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      SUBROUTINE DCTTP(DC,D)
C
C  Napier's rules for spherical triangles to convert from Direction
C    Cosines for a downward pointed plane normal To Trend and Plunge
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      DIMENSION D(2), DC(3)
      COMMON /PIO180/ PI180
C
      D(2) = ASIN(-DC(3))/PI180
      A = COS(PI180*D(2))
      B = ASIN(DC(2)/A)/PI180
      IF (B .GT. 0.0E0) THEN
        D(1) = ACOS(-DC(1)/A)/PI180
      ELSE
        D(1) = -ACOS(-DC(1)/A)/PI180
      ENDIF
C
      RETURN
      END
```

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      SUBROUTINE CROSS(A,B,Z)
C
C  vector CROSS product
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      DIMENSION A(3), B(3), Z(3)
C
      Z(1) = A(2)*B(3) - B(2)*A(3)
      Z(2) = -A(1)*B(3) + B(1)*A(3)
      Z(3) = A(1)*B(2) - B(1)*A(2)
C
      RETURN
      END
```

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      SUBROUTINE NORM(Z)
C
C  NORMalize a vector to unit length
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      EXTERNAL DOT
      DIMENSION Z(3)
C
      CALL DOT(Z,Z,ZMAG)
      ZMAG = SQRT(ZMAG)
      Z(1) = Z(1)/ZMAG
      Z(2) = Z(2)/ZMAG
      Z(3) = Z(3)/ZMAG
C
      RETURN
      END
```

```
C 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
C
      SUBROUTINE DOT(A,B,Z)
C
C  vector DOT product
C
      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*2 (I-N)
      DIMENSION A(3), B(3)
C
      Z = A(1)*B(1) + A(2)*B(2) + A(3)*B(3)
C
      RETURN
      END
```

APPENDIX B: CALCULATOR PROGRAM

STEP	KEY ENTRY	KEY CODE	COMMENTS
1	f lbl a	42 21 11	Program A: direction cosines, l, m and n, of downward pointed marker-plane normal, M, calculated from dip direction and dip.
2	gsb 1	32 1	Go to subroutine 1
3	sto 2	44 2	Store n^M in register 2
4	r↓	33	Roll n^M off stack
5	sto 1	44 1	Store m^M in register 1
6	r↓	33	Roll m^M off stack
7	sto 0	44 0	Store l^M in register 0
8	g rtn	43 32	Return -- end of program
9	f lbl b	42 21 12	Program B: direction cosines, l, m and n, of downward pointed fault-plane normal, F, calculated from dip direction and dip.
10	gsb 1	32 1	Go to subroutine 1
11	sto 5	44 5	Store n^F in register 5
12	r↓	33	Roll n^F off stack
13	sto 4	44 4	Store m^F in register 4
14	r↓	33	Roll m^F off stack
15	sto 3	44 3	Store l^F in register 3
16	g rtn	43 32	Return -- end of program
17	f lbl d	42 21 14	Program D: for hanging-wall-up faults, reverse the slip direction entered as trend, T, and plunge, P
18	chs	16	Change sign : P = -P
19	x↔y	34	Reverse contents of x and y stack registers
20	1	1	1
21	8	8	8
22	0	0	0
23	+	40	$T = 180^o + T$
24	x↔y	34	Reverse contents of x and y stack registers

STEP	KEY ENTRY	KEY CODE	COMMENTS
25	f fbl c	42 21 13	Program C: direction cosines, l, m and n, of slip direction, S, calculated from trend and plunge
26	gsb 2	32 2	Go to subroutine 2
27	sto 8	44 8	Store n^S in register 8
28	r↓	33	Roll n^S off stack
29	sto 7	44 7	Store m^S in register 7
30	r↓	33	Roll m^S off stack
31	sto 6	44 6	Store l^S in register 6
32	g rtn	43 32	Return -- end of program

STEP	KEY ENTRY	KEY CODE	COMMENTS
33	f lbl e	42 21 15	Program E: calculate ϕ and θ
			<u>Step 1: calculate $\phi = \cos^{-1}(M \cdot F)$</u>
34	rcl 0	45 0	Recall l^M to stack
35	rcl 3	45 3	Recall l^F to stack
36	x	20	$l^M \times l^F$
37	rcl 1	45 1	Recall m^M to stack
38	rcl 4	45 4	Recall m^F to stack
39	x	20	$m^M \times m^F$
40	+	40	$(l^M \times l^F) + (m^M \times m^F)$
41	rcl 2	45 2	Recall n^M to stack
42	rcl 5	45 5	Recall n^F to stack
43	x	20	$n^M \times n^F$
44	+	40	$\cos \phi = (l^M \times l^F) + (m^M \times m^F) + (n^M \times n^F) = M \cdot F$
45	g cos⁻1	43 24	$\phi = \cos^{-1}(M \cdot F)$
46	sto 9	44 9	Store ϕ in register 9
47	f pse	42 31	pause to display ϕ
48	f pse	42 31	pause to display ϕ
			<u>Step 2: calculate N = F x M</u>
49	rcl 2	45 2	Recall n^M to stack ...
50	sto .9	44 .9	... and store in register .9
51	rcl 1	45 1	Recall l^M to stack ...
52	sto .8	44 .8	... and store in register .8
53	rcl 0	45 0	Recall l^M to stack ...
54	sto .7	44 .7	... and store in register .7

STEP	KEY ENTRY	KEY CODE	COMMENTS
55	rcl 5	45 5	Recall n^F to stack
56	rcl 4	45 4	Recall m^F to stack
57	rcl 3	45 3	Recall 1^F to stack
58	gsb 3	32 3	Go to subroutine 3
59	sto .3	44 .3	Store n^N in register .3
60	r↓	33	Roll n^N off stack
61	sto .2	44 .2	Store m^N in register .2
62	r↓	33	Store Roll m^N off stack
63	sto .1	44 .1	Store 1^N in register .1
			<u>Step 3: calculate $\|\theta\| = \cos^{-1}(\|N \cdot S\|)$</u>
64	rcl 6	45 6	Recall 1^S to stack
65	x	20	$1^N \times 1^S$
66	rcl .2	45 .2	Recall m^N to stack
67	rcl 7	45 7	Recall m^S to stack
68	x	20	$m^N \times m^S$
69	+	40	$(1^N \times 1^S) + (m^N \times m^S)$
70	rcl .3	45 .3	Recall n^N to stack
71	rcl 8	45 8	Recall n^S to stack
72	x	20	$n^N \times n^S$
73	+	40	$\cos\theta = (1^N \times 1^S) + (m^N \times m^S) + (n^N \times n^S) = M \cdot F$
74	g test 2	43 30 2	If $\cos\theta < 0$
75	chs	16	Change sign: $\cos\theta = -\cos\theta$
76	g cos^1	43 24	$\|\theta\| = \cos^{-1}(\|N \cdot S\|)$
77	sto .0	44 .0	Store $\|\theta\|$ in register .0
			<u>Step 4: If $\|\theta\| < 1^o$, then null fault</u>
78	1	1	1
79	-	30	$\|\theta\| - 1$
80	g test 1	43 30 1	If $\|\theta\| - 1 > 0^o$
81	gto 4	22 4	Go to label 4
82	rcl .0	45 .0	Recall $\|\theta\|$ to the stack
83	g rtn	43 32	Return — end of program
			<u>Step 5: find sign of θ using $F \cdot (N \times S)$</u>
84	f lbl 4	42 21 4	Label 4
85	rcl 8	45 8	Recall n^S to stack ...

STEP	KEY ENTRY	KEY CODE	COMMENTS
86	sto .9	44 .9	... and store in register .9
87	rcl 7	45 7	Recall m^S to stack ...
88	sto .8	44 .8	... and store in register .8
89	rcl 6	45 6	Recall 1^S to stack ...
90	sto .7	44 .7	...and store in register .7
91	rcl .3	45 .3	Recall n^N to stack
92	rcl .2	45 .2	Recall m^N to stack
93	rcl .1	45 .1	Recall 1^N to stack
94	gsb 3	32 3	Go to subroutine 3
95	rcl 5	45 5	Recall n^F to stack
96	x	20	n^{NxS} x n^F
97	x↔y	34	Reverse contents of x and y stack registers
98	rcl 4	45 5	Recall m^F to stack
99	x	20	m^{NxS} x m^F
100	+	40	(n^{NxS} x n^F) + (m^{NxS} x m^F)
101	x↔y	34	Reverse contents of x and y stack registers
102	rcl 3	45 3	Recall 1^F to stack
103	x	20	1^{NxS} x 1^F
104	+	40	F•(NxS) = (n^{NxS} x n^F) + (m^{NxS} x m^F) + (1^{NxS} x 1^F)
105	g test 1	43 30 1	If F•(NxS) > 0
106	gto 5	22 5	Go to label 5
107	rcl .0	45 .0	Recall θ to stack
108	chs	16	Change sign: θ = −θ
109	sto .0	44 .0	Store θ in register .0
110	g rtn	43 32	Return -- end of program
111	f lbl 5	42 21 5	Label 5
112	rcl .0	45 .0	Recall θ to stack
113	g rtn	43 32	Return -- end of program
114	f lbl 1	42 21 1	Subroutine 1: calculate direction cosines, 1, m, and n, from dip direction, DD, and dip, D
115	sto .9	44 .9	Store D in register .9
116	sin	23	sin D
117	x↔y	34	Reverse contents of x and y stack registers

STEP	KEY ENTRY	KEY CODE	COMMENTS
118	sto .8	44 .8	Store DD in register .8
119	cos	24	cos DD
120	x	20	1 = cos DD x sin D
121	rcl .9	45 .9	Recall D to stack
122	sin	23	sin D
123	rcl .8	45 .8	Recall DD to stack
124	sin	23	sin DD
125	x	20	sin D x sin DD
126	chs	16	Change sign: m = -sin D x sin DD
127	rcl .9	45 .9	Recall D to stack
128	cos	24	cos D
129	chs	16	Change sign: n = cos D
130	g rtn	43 32	Return -- end of subroutine
131	f lbl 2	42 21 2	Subroutine 2: Calculate direction cosines, l, m and n, from trend, T, and plunge, P
132	sto .9	44 .9	Store P in register .9
133	cos	24	cos P
134	x↔y	34	Reverse contents of x and y stack registers
135	sto .8	44 .8	Store T in register .8
136	cos	24	cos T
137	x	20	cos T x cos P
138	chs	16	Change sign: l = -cos T x cos P
139	rcl .9	45 .9	Recall P to stack
140	cos	24	cos P
141	rcl .8	45 .8	Recall T to stack
142	sin	23	sin T
143	x	20	m = sin T x cos P
144	rcl .9	45 .9	Recall P to stack
145	sin	23	sin P
146	chs	16	Change sign: n = -sin P
147	g rtn	43 32	Return -- end of subroutine

STEP	KEY ENTRY	KEY CODE	COMMENTS
148	f lbl 3	42 21 3	Subroutine 3: calculate and normalize to unit length the cross product A x B
			<u>Step 1: C = A X B</u>
149	sto .4	44 .4	Store l^A in register .4
150	r↓	33	Roll l^A off stack
151	sto .5	44 .5	Store m^A in register .5
152	r↓	33	Roll m^A off stack
153	sto .6	44 .6	Store n^A in register .6
154	rcl .5	45 .5	Recall m^A to stack
155	rcl .9	45 .9	Recall n^B to stack
156	x	20	$m^A \times n^B$
157	rcl .8	45 .8	Recall m^B to stack
158	rcl .6	45 .6	Recall n^A to stack
159	x	20	$n^A \times m^B$
160	-	30	$l^C = (n^A \times m^B) - (m^A \times n^B)$
161	sto .1	44 .1	Store l^C in register .1
162	rcl .7	45 .7	Recall l^B to stack
163	rcl .6	45 .6	Recall n^A to stack
164	x	20	$n^A \times l^B$
165	rcl .4	45 .4	Recall l^A to stack
166	rcl .9	45 .9	Recall n^B to stack
167	x	20	$n^B \times l^A$
168	-	30	$m^C = (n^B \times l^A) - (n^A \times l^B)$
169	sto .2	44 .2	Store m^C in register .2
170	rcl .4	45 .4	Recall l^A to stack
171	rcl .8	45 .8	Recall m^B to stack
172	x	20	$m^B \times l^A$
173	rcl .7	45 .7	Recall l^B to stack
174	rcl .5	45 .5	Recall m^A to stack
175	x	20	$m^A \times l^B$
176	-	30	$n^C = (m^A \times l^B) - (m^B - l^A)$
177	sto .3	44 .3	Store n^C in register .3
			<u>Step 2: $D = C/\sqrt{C \cdot C}$</u>
178	g x^2	43 11	$(n^C)^2$

STEP	KEY ENTRY	KEY CODE	COMMENTS
179	rcl .2	45 .2	Recall m^C to stack
180	g x^2	43 11	$(m^C)^2$
181	+	40	$(m^C)^2 + (n^C)^2$
182	rcl .1	45 .1	Recall 1^C to stack
183	g x^2	43 11	$(1^C)^2$
184	+	40	$C \cdot C = (1^C)^2 + (m^C)^2 + (n^C)^2$
185	$\sqrt{\ }$	11	$\sqrt{C \cdot C}$
186	sto .9	44 .9	Store $\sqrt{C \cdot C}$ in register .9
187	rcl .1	45 .1	Recall 1^C to stack
188	rcl .9	45 .9	Recall $\sqrt{C \cdot C}$ to stack
189	÷	10	$1^D = 1^C/\sqrt{C \cdot C}$
190	rcl .2	45 .2	Recall m^C to stack
191	rcl .9	45 .9	Recall $\sqrt{C \cdot C}$ to stack
192	÷	10	$m^D = m^C/\sqrt{C \cdot C}$
193	rcl .3	45 .3	Recall n^C to stack
194	rcl .9	45 .9	Recall $\sqrt{C \cdot C}$ to stack
195	÷	10	$n^D = n^C/\sqrt{C \cdot C}$
196	g rtn	43 32	Return -- end of subroutine