Geological Survey of Canada

Open File 1627

HFRED : A PROGRAM FOR THE REDUCTION OF MARINE HEAT FLOW DATA ON A MICROCOMPUTER

H. Villinger and E.E. Davis

Pacific Geoscience Centre
Geological Survey of Canada
Sidney, B.C. V8L 4B2
Canada

1987

Canada

HFRED : A PROGRAM FOR THE REDUCTION OF MARINE HEAT FLOW DATA ON A MICROCOMPUTER

H. VILLINGER AND E.E. DAVIS

PACIFIC GEOSCIENCE CENTRE
GEOLOGICAL SURVEY OF CANADA
SIDNEY, B.C. V8L 4B2
CANADA

GSC OPEN FILE REPORT NO. 1627

1987

Table of contents

Abstract

HFRED: A PROGRAM FOR THE REDUCTION OF MARINE HEAT-FLOW DATA ON A MICROCOMPUTER

The program HFRED is a Fortran 77 program developed for use on a microcomputer for the reduction of heat-flow data collected with a heat-flow instrument that utilizes the pulse-probe method of conductivity measurement. The program uses a stable, accurate, and self-consistent iterative algorithm that deals with the non-ideal nature of the penetration and calibrated heat-pulse decays of each sensor and each penetration individually. The report includes commented listings and a 5.25" floppy disk of the program HFRED and of subroutines used, examples of input parameter and data files, a complete example of a reduction of a heat-flow penetration, a listing of the program FTABLE used to generate values of the cylindrical decay function F(alpha,tau), and a manuscript describing the theory of the reduction scheme. The program is provided without warranty of any kind, and the Geological Survey of Canada accepts no responsibility for its performance.

Foreword

The program HFRED reduces heat flow data made with a heat flow instrument that measures thermal conductivities in situ using the pulse probe method (Lister, 1979). The program was designed after the reduction outlines given in Hyndman et al. (1979), Davis et al. (1984), and Davis (1988), and includes an iterative search for the effective origin time both for the frictional and the heat pulse decay. The algorithm used is described briefly in the commentary notes within the program, but a complete description is found in Villinger and Davis (1987), a reprint of which is included with this report.

The program results are :

- undisturbed sediment temperatures vs. depth
- thermal conductivities vs. depth
- interval gradients vs. depth
- interval heat flow vs. depth derived from thermal resistance vs. depth.

The final heat flow has to be calculated manually.

The programming language is Fortran 77.The program was developped on an IBM-PC™ with a math coprocessor installed which is vital for the satisfactory performance of the program. The graphical output can be either done on a plotter or a graphic printer depending on the particular system configuration used.

The program FATAU is used to create the F(alpha,tau) table for a specific alpha value.

HFRED was designed and written by non-professional programmers. We are sure that it can be streamlined and made more user-friendly especially where error trapping is concerned.

Questions regarding details of the code should be send to :

Dr. H. Villinger
Alfred-Wegener-Institut for Polar Research
D-2850 Bremerhaven
W.-Germany
Tel. (0471) 4831-215
Telex 238 695

Listing of the program HFRED and the subroutines used

```
C*********************************************************************
C
C
C
C       AUTHOR : H. VILLINGER / E.E. Davis
C
C
C       TITLE :  HFRED
C
C
C
C       DATE : 1984/1985
C
C
C
C
C       PURPOSE
C       -------
C
C       REDUCTION OF HEAT FLOW PENETRATION MEASUREMENTS MADE WITH A
C       VIOLIN-BOW TYPE HEAT PROBE AND IN SITU THERMAL CONDUCTIVITY
C       MEASUREMENTS USING THE PULSE PROBE METHOD.
C
C
C       ALGORITHM
C       ---------
C       THE MAIN PURPOSE OF THE PROGRAM IS TO DETERMINE ITERATIVELY
C       THE EFFECTIVE ORIGIN TIMES FOR THE FRICTIONAL AND THE HEAT
C       PULSE DECAY. THE FINAL HEAT FLOW VALUE IS  N O T  CALCULATED
C       BUT HAS TO BE DETERMINED MANUALLY.
C       THE PROGRAM ITSELF IS NOT EQUIPPED WITH A LOT OF ERROR
C       TRAPPING ROUTINES SO THE USER SHOULD MAKE SURE IN ADVANCE
C       THAT FOR EXAMPLE THE DATA FILE IS SET UP PROPERLY.
C       THE ALGORITHM OF THE PROGRAM IS DESCRIBED IN DETAIL IN :
C
C       VILLINGER, H. AND DAVIS, E.E.  (1987)
C       A NEW REDUCTION ALGORITHM FOR MARINE HEAT-FLOW MEASUREMENTS.
C       JOURNAL OF GEOPHYSICAL RESEARCH (IN PRESS)
C
C       THE READING OF THE FOLLOWING PAPERS IS HIGHLY SUGGESTED
C       BEFORE USING THE PROGRAM :
C
C       HYNDMAN,R.D.,DAVIS,E.E. AND WRIGHT,J.A.(1979)
C       THE MEASUREMENT OF MARINE GEOTHERMAL HEAT FLOW BY A MULTI-
C       PENETRATION PROBE WITH DIGITAL ACOUSTIC TELEMETRY AND IN
C       SITU THERMAL CONDUCTIVITY.
C       MARINE GEOPHYSICAL RESEARCHES,4,181-205
C
C       C.R.B.LISTER (1979)
C       THE PULSE-PROBE METHOD OF CONDUCTIVITY MEASUREMENT.
C       GEOPHYS.J.R.ASTR.SOC.,57,451-461
C
C       DAVIS,E.E.,LISTER,C.R.B. AND SCLATER,J.G. (1984)
C       TOWARDS DETERMINING THE THERMAL STATE OF OLD OCEAN LITHO-
C       SPERE : HEAT FLOW MEASUREMENTS FROM THE BLAKE-BAHAMA OUTER
C       RIDGE,NORTH-WESTERN ATLANTIC.
C       GEOPHYS.J.R.ASTR.SOC.,78,507-545
```

```
C
C
C
C
C       FILE USAGE  (ONLY FOR THIS PROGRAM SEGMENT)
C       ----------------------------------------
C
C       2  = PARAMETER - FILE (INPUT)
C       11 = RESULT DISK FILE (OUTPUT)
C       *  = KEYBOARD/SCREEN
C
C
C       EXPLANATION OF SUBROUTINES USED
C       -------------------------------
C       GETPAR : LOADS THE PARAMETERS FROM A PARAMETER FILE HFRED.PAR
C       PARAM : PRINTS THE PARAMETERS OF HFRED.PAR
C       TABLE : LOADS THE F(ALPHA,TAU)-TABLE USED
C               THE TABLE HAS TO BE GENERATED BY THE USER USING THE
C               PROGRAM FALTAU LISTED ALSO IN THIS REPORT
C       TITLE : READS THE HEADER INFORMATION FROM THE
C               PENETRATION DATA FILE
C       CHANGE : CHANGES PRESET PARAMETERS READ IN FROM
C                 HFRED.PAR
C       READ : READS THE PENETRATION DATA FROM DISK ,CONVERTS THE DATA
C              AND SPLITS THE DATA SET INTO FRICTIONAL DECAY AND HEAT
C              PULSE DECAY.
C       FRIC : CALCULATES INFINITE TIME SEDIMENT TEMPERATURES
C       FOUT : PRINTS THE RESULTS FROM FRIC
C        COR : CORRECTS THE HEAT PULSE DECAY TEMPERATURES FOR RESIDUAL
C              FRICTIONAL HEATING
C       COND : CALCULATES THE IN SITU THERMAL CONDUCTIVITY
C       COUT : PRINTS RESULTS FROM COND
C       PLOTID : ASSIGNS A PLOT-ID NUMBER FOR THE PLOT
C       FINRES : PRINTS THE FINAL RESULTS FROM FRIC AND COND
C       RESULT : PLOTS THE FRICTIONAL AND HEAT PULSE DECAY, TEMPERATURE
C                 AND THERMAL CONDUCTIVITY VS. DEPTH AND THERMAL
C                 RESISTANCE VS. DEPTH
C       ZERO : SETS ALL ELEMENTS IN THE MAIN ARRAYS TO ZERO
C       FMVSD : COMPUTES THE MEAN, VARIANCE AND STANDART DEVIATION
C               OF AN ARRAY (FILE NAME IS MEAN)
C       BDWT : CALCULATES THERMAL RESISTANCES (AUTHORS : MARK WIEDER-
C              SPAN/U.TEXAS AT AUSTIN AND CLIVE LISTER/U.WASHINGTON
C              SEATTLE)
C       RATBUL : CALCULATES A THERMAL DIFFUSIVITY FROM THE THERMAL
C                 CONDUCTIVITY USING THE RELATIONSHIP DESCRIBED
C                 IN HYNDMAN ET AL. 1979 (SEE ABOVE)
C       TEMP : CONVERTS READINGS INTO TEMPERATURES
C       LEAST : CALCULATES A LEAST SQUARE FIT
C       FTABLE : INTERPOLATES FROM THE F(ALPHA,TAU) TABLE USED
C
C
C       VARIABLES
C       ---------
C
C       A DETAILED VARIABLE DESCRIPTION WILL BE GIVEN IN THE SUB-
C       ROUTINES WHERE THEY ARE INTRODUCED. ONLY THE VARIABLES
C       USED IN COMMON BLOCKS WILL BE DESCRIBED HERE.
C
```

```
C
C       COMMON BLOCK DECDAT  ( DECAY DATA )
C
C       TFRIC(J,K) : TIME OF FRICTIONAL DECAY DATA POINT J FOR SENSOR K
C       RFRIC(J,K) : VALUE OF FRICTIONAL DACAY DATA POINT J FOR SENSOR K
C       TPULS(J,K) : TIME OF HEAT PULSE DECAY DATA POINT J FOR SENSOR K
C       RPULS(J,K) : VALUE OF HEAT PULSE DECAY DATA POINT J FOR SENSOR K
C
C
C       COMMON BLOCK PDATA (PLOT DATA)
C
C       TFPLOT(J,K) : TIME OF FRICTIONAL DECAY DATA POINT J FOR SENSOR K
C       RFPLOT(J,K) : VALUE OF FRICTIONAL DACAY DATA POINT J FOR SENSOR K
C       TPPLOT(J,K) : TIME OF HEAT PULSE DECAY DATA POINT J FOR SENSOR K
C       RPPLOT(J,K) : VALUE OF HEAT PULSE DECAY DATA POINT J FOR SENSOR K
C
C
C
C       COMMON BLOCK SENSOR (SENSOR DATA)
C
C       NSENSR : NUMBER OF SENSORS (MAX. 15)
C       TCYCLE : TIME BETWEEN MEASURMENT CYCLES [SEC]
C       DELTAT : TIME BETWEEN MEASUREMENTS AT CONSECUTIVE SENSORS [SEC]
C       A      : RADIUS OF THE SENSOR STRING  [M]
C       DSENSR : LENGTH INCREMENT BETWEEN SENSORS [M]
C       TERROR : ASSUMED ERROR IN THE TEMPERATURE MEASUREMENTS [K]
C       PLEN   : LENGTH OF THE POWER INPUT [SEC]
C       AA,BB,CC : CALIBRATION COEFFICIENTS FOR THE CONVERSION OF
C                  READINGS INTO TEMPERATURES ASSUMING A POLYNOMIAL
C                  OF THIRD DEGREE (SEE FUNCTION TEMP)
C                  THE COEFFICIENTS HAVE TO BE SPECIFIED FOR EACH SENSOR
C       RATCL : COEFFICIENTS FOR THE RELATIONSHIP BETWEEN THE THERMAL
C               CONDUCTIVITY AND THERMAL DIFFUSIVITY
C       F : DATA OF THE F(ALPHA,TAU)-TABLE USED
C       FSTART : START VALUE OF THE F(ALPHA,TAU) TABLE
C       FINCR : INCREMENT OF TAU IN THE USED F(ALPHA,TAU) TABLE
C
C
C       COMMON BLOCK FRICO
C
C       FDELAY(J) : TIME DELAY FOR FRICTIONAL DELAY FOR SENSOR J [SEC]
C       DINCR : TIME INCREMENT USED IN THE FDELAY ITERATION [SEC]
C       IFMAX : MAX. NUMBER OF STEPS
C       TFMIN/TFMAX : MIN. AND MAX. TAU VALUES USED FOR FRICTIONAL DELAY
C
C
C       COMMON BLOCK CONDCO
C
C       KINIT(J) : INITIAL THERMAL CONDUCTIVITY [W/MK] FOR SENSOR J
C       KTOLER : ERROR BOUNDS FOR CONDUCTIVITY ITERATION [W/MK]
C       TSINIT : INITIAL TIME SHIFT [SEC]
C       TSINCR : INITIAL TIME SHIFT INCREMENT [SEC]
C       TPMIN/TPMAX : MIN. AND MAX. TAU VALUES USED FOR THE THERMAL
C                     CONDUCTIVITY DETERMINATION
C       IPMAX : MAX. ITERATION NUMBER
C       PDELAY(J) : TIME DELAY FOR HEAT PULSE DECAY FOR SENSOR J [SEC]
C       POWER : ENERGY INPUT PER LENGTH [JOULE/M]
C
```

```
C
C       COMMON BLOCK OUT
C
C       LUNDAT : UNIT NUMBER FOR DATA FILE
C       LUNOUT : UNIT NUMBER FOR OUTPUT FILE
C       LONG : IF LONG=1 THEN AN EXTENSIVE PRINTOUT IS CREATED WHICH IS HELPFUL
               IN CASES WHERE HFRED DOES NOT CONVERGE.
C       LPLOT : -- NOT USED AT THE MOMENT --
C
C
C       COMMON BLOCK PLOTPA (PLOT PARAMETER)
C
C       FOR DETAILS SEE SUBROUTINE RESULT
C
C*********************************************************************
c
       program hfred
c
       implicit real(k)
       character answ*1,fname*8,cruise*10
        dimension nfric(15),npuls(15),rwater(15),
     $            sedtem(15),estsed(15),sedrms(15),fslope(15),nfused(15),
     $            k(15),ktem(15),pslope(15),krms(15),npused(15),
     $            icond(15),ishift(15),
     $            work1(75),work2(75),time(75),tem(75)
       common/decdat/tfric(75,15),rfric(75,15),
     $               tpuls(75,15),rpuls(75,15)
       common/pdata/tfplot(75,15),rfplot(75,15),
     $              tpplot(75,15),rpplot(75,15)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
       common/plotpa/fp(8),tp(8),cp(8),hp(8),bp(8)
c
c.......
c
       write(*,*)'              HEAT FLOW REDUCTION PROGRAM '
       write(*,*)'              ==========================='
       write(*,*)' '
       write(*,*)'          Pacific Geoscience Centre  October 1984'
c
c.......GETPAR : LOADS THE PARAMETERS FROM THE PARAMETER FILE
c                HFRED.PAR
c
       call getpar(2)
c
c.......TABLE : LOADS THE F(ALPHA,TAU)-TABLE USED
c
       call table
c
c.......TITLE : READS THE HEADER INFORMATION FROM THE PENETRATION
c               DATA FILE AND PRINTS IT
c
2000       call title(istn,ipen,cruise)
```

```
c
c.......PARAM : PRINTS THE PARAMETERS OF HFRED.PAR
c
       call param
c
c.......CHANGE : CHANGES PRESET PARAMETERS READ IN FROM
c                HFRED.PAR
c
       write(*,*)'DO YOU WANT TO CHANGE PARAMETERS  Y / N '
       read(*,10)answ
       if(answ.eq.'Y'.or.answ.eq.'y')   then
         call change
         call param
       endif
c
c.......READ : READS THE PENETRATION DATA FROM DISK ,CONVERTS THE DATA
c              AND SPLITS THE DATA SET INTO FRICTIONAL DECAY AND HEAT
c              PULSE DECAY.
c
           call read(nfric,npuls,rwater,fricon,pulson)
c
c.......FRIC : CALCULATES INFINITE TIME SEDIMENT TEMPERATURES
c
         msum = 0
1000              do       100       i=1,nsensr
                      if(nfric(i).eq.0)   goto 100
                  kappa=ratbul(kinit(i))
                  call fric(i,nfric(i),kappa,work1,work2,
     $                 sedtem(i),estsed(i),sedrms(i),fslope(i),nfused(i))
                   msum = msum + nfused(i)
c
                            do       200       j=1,nfused(i)
                                    tfplot(j,i)=work1(j)
                                     rfplot(j,i)=work2(j)
                                     work1(j) = 0.
                                    work2(j) = 0.
200                            continue
c
100               continue
c
c.......FOUT : PRINTS THE RESULTS FROM FRIC
c
       if(msum.gt.0)  then
             call fout(nfric,nfused,sedtem,estsed,sedrms,fslope,rwater)
       endif
c
c.......
c
              msum = 0
              do       300       i=1,nsensr
c
                     if(npuls(i).eq.0)       then
                             k(i) = kinit(i)
                             goto       300
                     endif
c
c.......COR : CORRECTS THE HEAT PULSE DECAY TEMPERATURES FOR RESIDUAL
c             FRICTIONAL HEATING
```

```
c
              call cor(i,time,tem,npuls,sedtem,fslope,fricon,pulson)
c
c.......COND : CALCULATES THE IN SITU THERMAL CONDUCTIVITY
c
              call cond(i,npuls(i),time,tem,ktem(i),estsed(i),
     $                        npused(i),work1,work2,k(i),krms(i),
     $                        pslope(i),icond(i),ishift(i))
c
c.......
c
                    msum = npused(i) + msum
                       do       400       j=1,npused(i)
                       tpplot(j,i) = work1(j)
                       rpplot(j,i) = work2(j)
                       work1(j) = 0.
                       work2(j) = 0.
400                    continue
c
300                 continue
c
c.......COUT : PRINTS RESULTS FROM COND
c
       if(msum.gt.0)  then
              call cout(npuls,npused,ktem,k,krms,pslope,icond,ishift)
       endif
c
c.......PLOTID : ASSIGNS A PLOT-ID NUMBER FOR THE PLOT
c       FINRES : PRINTS THE FINAL RESULTS FROM FRIC AND COND
c       RESULT : PLOTS THE FRICTIONAL AND HEAT PULSE DECAY, TEMPERATURE
c                AND THERMAL CONDUCTIVITY VS. DEPTH AND THERMAL
c                    RESISTANCE VS. DEPTH
c
4000        write(*,*)' DO YOU WANT TO PLOT THE DATA ?   Y/N'
            read(*,10)answ
               if(answ.eq.'Y'.or.answ.eq.'y')   then
                  call plotid(pid)
              call finres(istn,ipen,sedtem,k,pid,weit,npused)
              call result(nfused,sedtem,fslope,istn,ipen,
     $                        npused,k,ktem,pslope,pid,weit)
                   else
              pid=0.
               call finres(istn,ipen,sedtem,k,pid,weit,npused)
               endif
c
c.......
c
         write(*,*)' SAVE THE RESULTS ON DISKETTE ? Y/N'
         read(*,10)answ
          if(answ.eq.'Y'.or.answ.eq.'y')   then
             idnt = pid
             write(fname,90)istn,ipen,idnt
             open(11,file=fname,status='new')
             write(11,110)istn,ipen,idnt,cruise
             do              800 l=1,nsensr
c
             write(11,120)l,sedtem(l),estsed(l),fdelay(l),fslope(l),
     $                       k(l),krms(l),pdelay(l),pslope(l)
```

```
800          continue
             close(11)
          endif
c
c.......
c
            write(*,*)' ANOTHER ITERATION ?   Y/N'
       read(*,10)answ
       if(answ.eq.'Y'.or.answ.eq.'y')   then
           write(lunout,20)
                do          500            i=1,nsensr
          fdelay(i) = abs(fdelay(i)/2.)
          kinit(i) = k(i)
           pdelay(i) = pdelay(i)-(pdelay(i)*0.3)
500          continue
c
              write(*,*)' DO YOU WANT TO CHANGE PARAMETERS?   Y/N'
             read(*,10)answ
          if(answ.eq.'Y'.or.answ.eq.'y')      call change
c
          call param
          goto 1000
       endif
c
       write(*,*)' ANOTHER PENETRATION ?   Y/N'
       read(*,10)answ
       if(answ.eq.'Y'.or.answ.eq.'y')  then
          write(lunout,20)
c
c.......ZERO : SETS ALL ELEMENTS IN THE MAIN ARRAYS TO ZERO
c
           do       600       i=1,nsensr
             call zero(nfric,npuls,rwater,
     $                  sedtem,estsed,sedrms,
     $                         fslope,nfused,
     $                  k,ktem,pslope,krms,npused,
     $                  icond,ishift,
     $                  work1,work2,time,tem)


600          continue
c
             call getpar(2)
c
              goto 2000
c
       endif
c
10     format(a1)
20     format(1h1)
90     format(i2.2,i3.3,i3.3)
110    format(3i5,1x,1h',a10,1h')
120    format(i2,1x,f8.3,1x,f8.4,1x,f8.2,1x,f8.3,1x,f8.3,1x,f8.4,1x,
     $       f8.2,1x,f8.3)
       stop
       end
```

```
C*********************************************************************
C       SUBROUTINE GETPAR
C
C       SUBROUTINE LOADS THE PARAMETRS FROM A PARAMETER FILE  CALLED
C       HFRED.PAR WHICH HAS TO BE ON THE DEFAULT DRIVE.
C
C       VARIABLES ARE EXPLAINED IN THE MAIN PROGRAM
C
C*********************************************************************
c
       subroutine getpar(lun)
       implicit real(k)
       character para*80,fname*50
c
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
       common/plotpa/fp(8),tp(8),cp(8),hp(8),bp(8)
c
c.......
c
       open( lun, file='hfred.par' )
c
c.......
c
       read( lun,* ) nsensr
       read( lun,* ) tcycle
       read( lun,* ) deltat
       read( lun,* ) a
       read( lun,* ) dsensr
       read( lun,* ) terror
       read( lun,* ) plen
       read( lun,* ) (aa(i),i=1,nsensr)
       read( lun,* ) (bb(i),i=1,nsensr)
       read( lun,* ) (cc(i),i=1,nsensr)
       read( lun,* ) (ratcl(i),i=1,3)
c
       read( lun,* ) (fdelay(i),i=1,nsensr)
       read( lun,* ) ifmax,dincr
       read( lun,* ) tfmin,tfmax
c
       read( lun,* ) (pdelay(i),i=1,nsensr)
       read( lun,* ) (kinit(i),i=1,nsensr)
       read( lun,* ) power
       read( lun,* ) tsinit,tsincr
       read( lun,* ) ipmax,ktoler
       read( lun,* ) tpmin, tpmax
c
       read( lun,* ) (fp(i),i=1,8)
       read( lun,* ) (tp(i),i=1,8)
       read( lun,* ) (cp(i),i=1,8)
       read( lun,* ) (hp(i),i=1,8)
       read( lun,* ) (bp(i),i=1,8)
```

```
c
      read( lun,* ) lundat,lunout,long,lplot
c
      close(lun)
c
c.......filename has to be entered at run-time
c
c
       write(*,*)'ENTER NAME OF OUTPUT-FILE '
       read(*,10)fname
10     format(a50)
c
      open(lunout,file=fname)
      if(fname.eq.'CON'.or.fname.eq.'con')    open(0)
c
      return
      end
```

```
c**********************************************************************
c
c       SUBROUTINE PARAM
c
c
c        SUBROUTINE TO PRINTOUT PARAMETERS OF THE PARAMETER-FILE
c       HFRED.PAR
c
c**********************************************************************
c
       subroutine param
c
       implicit real(k)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     +               ipmax,pdelay(15),power
       common/plotpa/fp(8),tp(8),cp(8),hp(8),bp(8)
       common/out/lundat,lunout,long,lplot
c
       write(lunout,*)
       write(lunout,*)'       PARAMETERS'
       write(lunout,*)'       ==========='
       write(lunout,50)nsensr
       write(lunout,140)tcycle
       write(lunout,150)deltat
       write(lunout,60)a
       write(lunout,70)power
       write(lunout,10)(fdelay(i),i=1,nsensr)
       write(lunout,80)tfmin
       write(lunout,20)(pdelay(i),i=1,nsensr)
       write(lunout,30)(kinit(i),i=1,nsensr)
       write(lunout,40)tsinit,tsincr
       write(lunout,90)ktoler
       write(lunout,110)tpmin
       write(lunout,120)alpha
       write(lunout,130)fincr
c
       write(*,*)
       write(*,*)'       PARAMETERS'
       write(*,*)'       ==========='
       write(*,50)nsensr
       write(*,140)tcycle
       write(*,150)deltat
       write(*,60)a
       write(*,70)power
       write(*,10)(fdelay(i),i=1,nsensr)
       write(*,80)tfmin
       write(*,20)(pdelay(i),i=1,nsensr)
       write(*,30)(kinit(i),i=1,nsensr)
       write(*,40)tsinit,tsincr
       write(*,90)ktoler
       write(*,110)tpmin
       write(*,120)alpha
       write(*,130)fincr
```

```
c
10      format(1h ,5x,'frictional time delays (s)',7f6.1)
20      format(1h ,5x,'pulse time delays (s) ',7f6.1)
30      format(1h ,5x,'initial thermal cond.',7f6.2)
40      format(1h ,5x,'initial time-shift and time-shift increment',
     $         2f8.2)
50      format(1h ,5x,'number of sensors : ',i2)
60      format(1h ,5x,'radius of the sensor string (m) : ',f7.5)
70      format(1h ,5x,'power input per length (J/m) : ',f8.2)
80      format(1h ,5x,'min. tau for frictional decay : ',f6.2)
90      format(1h ,5x,'max. thermal cond. error in COND : ',f8.4)
110      format(1h ,5x,'min. tau for heat pulse decay : ',f6.2)
120      format(1h ,5x,'alpha-value used : ',f6.2)
130      format(1h ,5x,'increment of F(alpha,tau)-table : ',f8.3,/)
140      format(1h ,5x,'cycle time (s) : ',f4.1)
150      format(1h ,5x,'time increment between sensor readings (s) :',
     $         f5.2)
c
      return
      end
```

```
C*********************************************************************
C       SUBROUTINE TABLE
C
C       SUBROUTINE READS PRECALCULATED TAU AND F(ALPHA,TAU) VALUES
C       FROM A FILE CALLED FAT__.TAB. THE NAME OF THE FILE INDICATES
C       WHICH VALUE OF ALPHA WAS USED. FAT__.TAB HAS TO BE ON THE
C       DEFAULT DRIVE.
C
C*********************************************************************
C
       subroutine table
c
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $                aa(15),bb(15),cc(15),
     $                ratcl(3),f(5000),alpha,fstart,fincr
        character faltau*20
c
       write( *,* )
       write( *,20 )
       read( *,10 )faltau
c
       open(79,file=faltau)
c
c.......F(1) = VALUE OF ALPHA
c       F(2) = START VALUE OF TAU
c       F(3) = INCREMENT OF TAU
c
       read(79,*)f(1),f(2),f(3)
       alpha=f(1)
       fstart=f(2)
       fincr=f(3)
c
c.......F(I) FOR I>3 : VALUES OF F(ALPHA,TAU)
c
       read(79,*,err=1000,end=1000)(f(i),i=4,5000)
1000   close(79)
10     format(a20)
20     format(1h ,////,10x,'ENTER NAME OF THE F(ALPHA,TAU)-FILE',/,
     $         15x,'NAME CONVENTIONS ARE AS FOLLOWS',/,
     $         15x,'ALPHA','     FILE-NAME',/,
     $         15x,' 1.8','      FAT18.TAB',/,
     $         15x,' 1.9','      FAT19.TAB',/,
     $         15x,' 2.0','      FAT20.TAB',/,
     $         15x,' 2.1','      FAT21.TAB',/,
     $         15x,' 2.2','      FAT22.TAB',/)
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE TITLE
c
c       SUBROUTINE READS HEADER INFORMATION FROM SPECIFIED PENETRATION
c       DATA FILE AND PRINTS IT.
c
c**********************************************************************
C
       subroutine title( istn,ipen,cruise)
c
       character hfdata*40,cruise*10
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $                 aa(15),bb(15),cc(15),
     $                 ratcl(3),f(5000),alpha,fstart,fincr
        common/out/lundat,lunout,long,lplot
c
       cruise='--n/a--'
c
       write(*,*)'   enter penetration file number'
       read(*,10)hfdata
c
       open( lundat, file=hfdata )
c
       read( lundat,*,err=200,end=100 ) istn, ipen, cruise
200     read( lundat,*,end=100 ) xlat, xlong, depth
        read( lundat,*,err=700, end=100 ) instid, istrng, nsensr
c
700        write( lunout,1010 )
           write( lunout,1020 )
           write( lunout,1025 ) cruise
           write( lunout,1030 ) istn, ipen
           write( lunout,1040 ) instid, istrng
           write( lunout,1050 ) xlat, xlong, depth
           write( lunout,1020 )
           write( lunout,1060 )
c
100       continue
c
10          format(a40)
1010           format(1h1,/,1h ,15x, 50('*') )
1020           format(15x, ' *', 48x, '*' )
1025           format(15x,' *',5x,'Cruise:',a10,26x,'*')
1030           format(15x, ' *',5x,'Station:',i5,8x,'Penetration:',i5,
     $             5x,'*')
1040           format(15x, ' *',5x,'Instrument:',i5,5x,'Sensor:',i5,
     $             10x,'*')
1050           format(15x, ' *',5x,'Latitude:',f8.3,4x,'Longitude:',f9.3
     $                ,3x,'*'/15x, ' *',5x,'Depth(m):',f5.0,29x,'*' )
1060           format(1h ,15x, 50('*') )
       return
       end
```

```
c**********************************************************************
c       SUBROUTINE CHANGE
c
c       SUBROUTINE TO CHANGE PARAMTERS READ IN FROM THE PARAMETER
c       FILE HFRED.PAR
c
c**********************************************************************
c
       subroutine change
c
       implicit real(k)
c
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/plotpa/fp(8),tp(8),cp(8),hp(8),bp(8)
       common/out/lundat,lunout,long,lplot
c
       write(*,*)'       CHANGE OF PARAMETERS'
       write(*,*)'       ====================='
       write(*,*)
       write(*,*)
       write(*,*)' TO LEAVE A PARAMETER UNCHANGED JUST PRESS C'
       write(*,*)' AND <return> '
       write(*,*)
       write(*,*)' FRICTIONAL DECAY PARAMETERS'
       write(*,*)' NUMBER OF SENSORS : ',nsensr
       read(0,*,err=50)idata
       nsensr = idata
       write(*,*)' NEW VALUE : ',nsensr
50      continue
       write(*,*)' FRICTIONAL TIME DELAYS'
       do       100       i=1,nsensr
       write(*,*)' SENSOR : ',i,'  DELAY : ',fdelay(i)
       read(0,*,err=100) data
       fdelay(i)=data
       write(*,*)' NEW VALUE : ',fdelay(i)
100     continue
       write(*,*)' MAXIMAL ITERATION NUMBER :',ifmax
       read(0,*,err=200) idata
       itmax=idata
       write(*,*)'  NEW VALUE : ',ifmax
200       write(*,*)'  TIME INCREMENT  FRIC DECAY ADJUSTMENT :',dincr
       read(0,*,err=300) data
       dincr=data
       write(*,*)'  NEW VALUE : ',dincr
300       write(*,*)' MINIMAL TAU-VALUE : ',tfmin
       read(0,*,err=400)data
       tfmin=data
       write(*,*)'  NEW VALUE : ',tfmin
400       write(*,*)' MAXIMAL TAU-VALUE : ',tfmax
       read(0,*,err=500) data
       tfmax=data
       write(*,*)'  NEW VALUE : ',tfmax
```

```
500       write(*,*)' CONDUCTIVITY REDUCTION PARAMETERS'
      write(*,*)
      write(*,*)' HEAT PULSE TIME DELAYS'
      do       600       i=1,nsensr
      write(*,*)' SENSOR : ',i,'  DELAY : ',pdelay(i)
      read(0,*,err=600) data
      pdelay(i)=data
      write(*,*)'  NEW VALUE : ',pdelay(i)
600      continue
      write(*,*)' INITIAL CONDUCTIVITIES '
      do       700       i=1,nsensr
      write(*,*)' SENSOR : ',i,'  INITIAL CONDUCTIVITY : ',kinit(i)
      read(0,*,err=700) data
      kinit(i)=data
      write(*,*)'  NEW VALUE : ',kinit(i)
700      continue
      write(*,*)' INITIAL TIME-SHIFT FOR COND : ',tsinit
      read(0,*,err=800) data
      tsinit=data
      write(*,*)'  NEW VALUE : ',tsinit
800      write(*,*)' TIME-SHIFT INCREMENT FOR COND : ',tsincr
      read(0,*,err=900) data
      tsincr=data
      write(*,*)'  NEW VALUE : ',tsincr
900       write(*,*)' K-TOLERANCE IN COND : ',ktoler
      read(0,*,err=1000) data
      ktoler=data
      write(*,*)'  NEW VALUE : ',ktoler
1000      write(*,*)' MINIMAL TAU-VALUE : ',tpmin
      read(0,*,err=1100) data
      tpmin=data
      write(*,*)'  NEW VALUE : ',tpmin
1100      write(*,*)' MAXIMAL TAU-VALUE : ',tpmax
      read(0,*,err=1200) data
      tpmax=data
      write(*,*)'  NEW VALUE : ',tpmax
1200   write(*,*)'  ---   END OF CHANGE  ---  '
      return
      end
```

```
c**********************************************************************
c       SUBROUTINE READ
c
c       SUBROUTINE TO READ THE PENETRATION DATA FROM A DATA FILE
c       SPECIFIED IN TITLE
c
c       VARIABLES
c       ---------
c       NFRIC : NUMBER OF DATA POINTS FOR THE FRICTIONAL DECAY
c       NPULS : NUMBER OF DATA POINTS FOR THE HEAT PULSE DECAY
c       RWATER : WATER TEMPERATURES FOR EACH SENSOR
c       FRICON : ORIGIN TIME OF PENETRATION
c       PULSON : ORIGIN TIME OF HEAT PULSE
c
c**********************************************************************
c
       subroutine read( nfric,npuls,rwater,fricon,pulson)
c
       implicit real(k)
       common/decdat/tfric(75,15),rfric(75,15),
     $               tpuls(75,15),rpuls(75,15)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
       dimension rwater(nsensr),work(16),nfric(nsensr),npuls(nsensr)
c
c.......
c
       do 100 i = 1, nsensr
         nfric(i) = 0
         npuls(i) = 0
100       continue
c
c.......INPUT OF TIMES OF PENETRATION AND HEAT PULSE
c
        read( lundat,*,err=2100,end=2100) fricon
2100        read( lundat,*,err=2200,end=2200 ) pulson
2200       fricon=fricon*tcycle
        pulson=pulson*tcycle
c
       if( long .ne. 0 ) write( lunout,1020 ) fricon,pulson
c
c.......INPUT AND CONVERSION OF WATER TEMPERATURES
c
       read( lundat,* ) (rwater(i),i=1,nsensr)
       do 200 i=1, nsensr
         rwater(i) = temp( rwater(i)/1000., i )
200       continue
c
       if( long .ne. 0 ) write( lunout,2010 ) (rwater(i),i=1,nsensr)
c
c.......READ DATA FROM FILE UNTIL EOF/ERROR/ZEROS
c       THE TIME IS EXPECTED TO BE IN THE FIRST COLUMN
c       DATA WITH A VALUE OF 1023 ARE IGNORED AS THEY INDICATE OVERFLOW
```

```
c
300       read( lundat,*,end=500,err=500 ) (work(i),i=1,nsensr+1)
c
       if( work(1) .eq. 0.0 ) goto 500
c
       time=work(1)*tcycle
c
        do 400 i=1,nsensr
         if( work(i+1).eq. 1023 ) then
           goto 400
             else
               if(time.lt.pulson )  then
                    nfric(i)=nfric(i)+1
                    tfric(nfric(i),i)=time-fricon+float(i-1)*deltat
                    rfric(nfric(i),i)=temp(work(i+1)/1000.,i )-rwater(i)
              else
                   npuls(i)=npuls(i)+1
                    tpuls(npuls(i),i)=time-pulson+float(i-1)*deltat
                   rpuls(npuls(i),i)=temp(work(i+1)/1000.,i)-rwater(i)
              endif
             endif
400         continue
c
        goto 300
c
500       continue
c
       close(lundat)
c
c.......
c
       if(long.ne.0)       then
       write(lunout,*)'frictional decay,nfric:',(nfric(i),i=1,nsensr)
       mxfric=nfric(1)
       mxpuls=npuls(1)
       do       600       i=1,nsensr
       if(nfric(i).gt.mxfric)    mxfric=nfric(i)
       if(npuls(i).gt.mxpuls)    mxpuls=npuls(i)
600       continue
       do       700       i=1,mxfric
       write(lunout,3030)i,(tfric(i,l),rfric(i,l),l=1,nsensr)
700       continue
       write(lunout,*)'heat pulse decay,npuls:',(npuls(i),i=1,nsensr)
       do       800       i=1,mxpuls
       write(lunout,3030)i,(tpuls(i,l),rpuls(i,l),l=1,nsensr)
800       continue
       endif
c
c.......
c
1020       format( ' fricon,pulson : ', 2(f6.0,2x))
2010       format( ' water temp = ', 10 (f6.3,1x ))
3030       format(1h ,i2,1x,7(f4.0,1x,f5.3,'|'))
c
       return
       end
```

```
c**********************************************************************
c       SUBROUTINE FRIC
c
c       SUBROUTINE TO CALCULATE INFINITE TIME TEMPERATURE FOR SENSOR i
c
c        VARIABLES
c       ---------
c       I : SENSOR NUMBER
c       NUMB : NUMBER OF DATA POINTS AVAILABLE FOR FRICTIONAL DECAY
c       KAPPA : ASSUMED THERMAL CONDUCTIVITY
c       FTAU : F(ALPHA,TAU) VALUES FOR THE DATA POINTS
c       TEM : TEMPERATURE DATA
C       STEM : SEDIMENT TEMPERATURE
c       EST : 95% CONFIDENCE ERROR LIMITS
c       FRMS : RMS ERROR
c       SLOPE : SLOPE OF THE LINEAR REGRESSION OF F(ALPHA,TAU)
c               VS. TEMPERATURE
c       NUSED : NUMBER OF DATA POINTS USED FOR THE CALCULATION OF STEM
c               WITHIN PRESET TAU-WINDOW
c
c
c**********************************************************************
c
      subroutine fric(i,numb,kappa,ftau,tem,
     $                  stem,est,frms,slope,nused)
c
      implicit real (k)
      common/decdat/tfric(75,15),rfric(75,15),
     $               tpuls(75,15),rpuls(75,15)
      common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
      common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
      common/out/lundat,lunout,long,lplot
      dimension tau(75),ftau(75),tem(75),err(75),
     $           ts(50),rms(50)
c
      nused=0
      rms(1) = 0.05
c
c.......SELECTION OF POINTS IN THE PRESET TIME WINDOW
c       NEGATIVE TEMPERATURES ARE EXCLUDED
c
c       THE USER OF THE ALGORITHM SHOULD BE AWARE OF THE FACT
c       THAT IN THE IDEAL CASE WITH PERFECT DATA THE DATA SET
c       SHOULD BE DECIMATED AT THIS POINT TO EQUALLY SPACED
c       POINTS IN THE 1/TIME-SPACE. FROM OUR EXPERIENCE WITH
c       DIFFERENT HEAT FLOW INSTRUMENTS AND THEIR CHARACTERISTIC
c       NOISE PROBLEMS, HOWEVER, INCLUSION OF ADDITIONAL POINTS MAY BE
c       STATISTICALLY WARRENTED.
c
c
      do       100       j=1,numb
      tau(j) = tfric(j,i)
       tau(j) = (kappa * tau(j))/( a*a )
           if(tau(j).ge.tfmin.
     $     and.tau(j).le.tfmax.
```

```
     $          and.tau(j).ge.fstart.
     $      and.rfric(j,i).gt.0.0)      then
             nused = nused + 1
             tem(nused) = rfric(j,i)
              tau(nused) = tau(j)
              ftau(nused) = ftable(tau(j))
            err(nused) = terror
        endif
100        continue
c
c.......
c
       if(long.ne.0)    then
         write(lunout,*)'       sensor : ',i
         write(lunout,*)'tau-falphatau-temperature,nfused : ',nused
         do       200       l=1,nused,4
         write(lunout,10)(tau(l+j-1),ftau(l+j-1),tem(l+j-1),j=1,4)
10         format(1h ,4(f5.3,1x,f7.5,1x,f5.3,'*'))
200         continue
         write(lunout,*)'l,ts,inf.time temp,slope,rms'
       endif
c
c.......IF DINCR = 0 THEN NO ITERATION
c
       if(dincr.eq.0.)     then
              call least(tau,tem,err,nused,stem,slope,vara,
     $        varb,covar,cor,frms)
              if(long.ne.0)     then
                     write(lunout,*)'  least squares fit'
                     write(lunout,*)'  sed. temp. : ',stem
                     write(lunout,*)'  slope : ',slope
                     write(lunout,*)'  rms : ',frms
                     write(lunout,*)'  var slope : ',vara
                     write(lunout,*)'  var intercept : ',varb
                     write(lunout,*)'  covar slope-intercept : ',covar
              endif
               slope = 0.
               fdelay(i) = 0.
               est = 2.*frms
               return
         endif
c
c.......IF NO DATA POINT WITHIN TAU-WINDOW GO TO MAIN PROGRAM
c
       if(nused.eq.0)      return
c
c.......INCREMENTAL INCREASE OF THE DELAY TIME TS
c
       do       400       l=2,ifmax
c
              ts(l) = fdelay(i) + float(l-1)*dincr
c
                     do       500       j=1,nused
                      time = tau(j) + (ts(l)*kappa)/(a*a)
                      ftau(j) = ftable(time)
500                     continue
c
              call least(ftau,tem,err,nused,stem,slope,vara,
```

```
     $                    varb,covar,cor,rms(l))
c
               if(long.ne.0)    then
                      write(lunout,*)l,ts(l),stem,slope,rms(l)
               endif
c
               drms = rms(l) - rms(l-1)
c
c.......
c
               if(drms.gt.0.)   then
                       fdelay(i) = ts(l) - dincr/2.
                      goto 1000
               endif
c
400               continue
c
       if(l.gt.ifmax)   fdelay(i)=ts(l-1)
c
c.......FINAL CALCULATION OF THE SEDIMENT TEMPERATURE WITH THE
c       DELAY GIVING THE SMALLEST RMS VALUE
c
1000      do       700       j=1,nused
             time = tau(j) + (fdelay(i)*kappa)/(a*a)
             ftau(j) = ftable(time)
700       continue
c
       call least(ftau,tem,err,nused,stem,slope,vara,
     $             varb,covar,cor,frms)
c
        if(long.ne.0)    then
          write(lunout,*)'  least squares fit'
          write(lunout,*)'  var slope : ',vara
          write(lunout,*)'  var intercept : ',varb
          write(lunout,*)'  covar slope-intercept : ',covar
        endif
c
c.......THE 95% CONFIDENCE ERRORS ARE ASSUMED TO BE TWICE THE
c       STANDART DEVIATION OF THE DATA FIT
c
       est=2.*frms
c
       return
       end
```

```
c**********************************************************************
c       SUBROUTINE FOUT
c
c       SUBROUTINE FOUT PRINTS ALL THE RESULTS OF FRIC FOR ALL
c       SENSORS IN A TABULATED FORM
c
c       VARIABLES
c       ---------
c       THE EXPLANATION OF THE VARIABLES CAN BE FOUND IN THE
c       SUBROUTINE FRIC
c
c**********************************************************************
c
       subroutine fout(nfric,nfused,sedtem,estsed,sedrms,fslope,
     $                  rwater)
c
       implicit real(k)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/out/lundat,lunout,long,lplot
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
        dimension sedtem(nsensr),estsed(nsensr),sedrms(nsensr),
     $           fslope(nsensr),rwater(nsensr),
     $           nfric(nsensr),nfused(nsensr)
c
       write(lunout,20)
       write(lunout,30)
       write(lunout,40)
       write(lunout,50)
c
       write(*,20)
       write(*,30)
       write(*,40)
       write(*,50)
c
       do       100       i=1,nsensr
       write(lunout,60)i,nfric(i),nfused(i),sedtem(i),estsed(i),
     $                  fdelay(i),fslope(i),rwater(i)
       write(*,60)i,nfric(i),nfused(i),sedtem(i),estsed(i),
     $                  fdelay(i),fslope(i),rwater(i)
c
       if(i.lt.nsensr)   then
        grad = (sedtem(i) - sedtem(i+1))/dsensr
       grad = 1000.*grad
       write(lunout,70) grad
       write(*,70) grad
       endif
c
100       continue
c
20       format(1h ,/,1h ,32x,'FRICTIONAL DECAY')
30       format(1h ,32x,'===============')
40       format(1h ,5x,'sensor',2x,'no of points',2x,'sed.temp.',3x,
     $         '95% level',3x,'gradient',2x,'delay',2x,'slope',
     $          2x,'water')
50       format(1h ,13x,'total / used',3x,'(deg)',5x,'error (deg)',
```

```
     $        2x,'(mdeg/m)',3x,'(s)',10x,'temp.')
60       format(1h ,5x,i3,8x,i2,' / ',i2,5x,f6.3,7x,f5.4,14x,f6.1,1x,
     $         f6.3,2x,f5.3)
70       format(1h ,52x,f6.1)
         write(lunout,80)
80       format(1h ,//)
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE COR
c
c       SUBROUTINE CORRECTS THE HEAT PULSE DECAY TEMPERATURES FOR
c       RESIDUAL FRICTIONAL HEATING
c
c       VARIABLES
c       ---------
c       VARIABLES ARE EXPLAINED IN FRIC AND HFRED
c
c**********************************************************************
c
       subroutine cor(i,time,tem,npuls,sedtem,fslope,fricon,pulson)
c
       implicit real (k)
       common/decdat/tfric(75,15),rfric(75,15),
     $                tpuls(75,15),rpuls(75,15)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $                aa(15),bb(15),cc(15),
     $                ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
            common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $                ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
       dimension sedtem(nsensr),fslope(nsensr),npuls(nsensr),
     $            time(100),tem(100)
c
       kappa = ratbul(kinit(i))
c
       do       100       j=1,npuls(i)
        tim = tpuls(j,i) + (pulson-fricon)
       tau = (kappa*tim)/(a*a)
       fatau = ftable(tau)
       temps = sedtem(i) + fslope(i)*fatau
        tem(j) = rpuls(j,i) - temps
       time(j) = tpuls(j,i) - (plen/2.)
c
100       continue
c
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE COND
c
c       SUBROUTINE COND CALCULATES IN SITU THERMAL CONDUCTIVITY
c       USING THE DECAY OF A HEAT PULSE (PULSE PROBE METHOD AFTER
c       LISTER (1979)
c
c       DETAILS OF THE USED ALGORITHM ARE EXPLAINED IN :
c
c       HEINER VILLINGER AND E.E.DAVIS (1987)
c       A NEW REDUCTION ALGORITHM FOR MARINE HEAT-FLOW MEASUREMENTS
c       JOURNAL OF GEOPHYSICAL RESEARCH, IN PRESS
c
c       VARIABLES
c       ---------
c       I : SENSOR NUMBER
c       NUMB : NUMBER OF DATA POINTS
c       TIME : TIME
c       TEMP : TEMPERATURE
c       AXE : INFINITE TIME TEMPERATURE
c       EST : 95% ERROR ESTIMATE FOR INFINITE TIME TEMPERATURES
c       NUSED : NUMBER OF POINTS USED
c       FTAU : F(ALPHA,TAU) VALUES FOR THE DATA POINTS
c       TEM : CORRECTED TEMPERATURES
c       K : CALCULATED THERMAL CONDUCTIVITY
c       RMS : STANDART DEVIATION OF THE THERMAL CONDUCTIVITY
c       SLOPE : SLOPE OF TEMPERATURE VS. F(ALPHA,TAU)
c       ICOND : NUMBER OF CONDUCTIVITY ITERATIONS
c       ISHIFT : TOTAL NUMBER OF ITERATIONS
c
c**********************************************************************
c
       subroutine cond(i,numb,time,temp,axe,est,nused,
     $                  ftau,tem,k,rms,slope,icond,ishift)
c
       implicit real (k)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
c
       dimension tau(75),ftau(75),tem(75),err(75),temp(75),
     $           time(75),timinv(75),temc(75),kpoint(75)
c
       nused = 0
       pi=4.*atan(1.)
       kappa = ratbul(kinit(i))
       if(est.eq.0.)   est = terror
c
c.......SELECTION OF DATA POINTS IN THE PRESET TIME-WINDOW
c       NEGATIVE TEMPERATURES ARE EXCLUDED
c
c       THE USER OF THE ALGORITHM SHOULD BE AWARE OF THE FACT
c       THAT IN THE IDEAL CASE WITH PERFECT DATA THE DATA SET
```

```
c       SHOULD BE DECIMATED AT THIS POINT TO EQUALLY SPACED
c       POINTS IN THE 1/TIME-SPACE. FROM OUR EXPERIENCE WITH
c       DIFFERENT HEAT FLOW INSTRUMENTS AND THEIR CHARACTERISTIC
c       NOISE PROBLEMS, HOWEVER, INCLUSION OF ADDITIONAL POINTS MAY BE
c       STATISTICALLY WARRENTED.
c
c
       do       100       j=1,numb
       tau(j) = (kappa * time(j))/( a*a )
       if(tau(j).ge.tpmin.
     $    and.tau(j).le.tpmax.
     $        and.tau(j).ge.fstart.
     $            and.temp(j).gt.0.0)     then
            nused = nused + 1
            time(nused)=time(j)
            tau(nused) = tau(j)
            tem(nused) = temp(j)
            err(nused) = est
       endif
100    continue
c
c.......
c
       if(long.ne.0)   then
         write(lunout,*)
         write(lunout,*)
         write(lunout,*)'       sensor : ',i
         write(lunout,*)'tau-falphatau-temperature,npused : ',nused
         do       200       l=1,nused,6
         write(lunout,30)(tau(l+j-1),tem(l+j-1),j=1,6)
30       format(1h ,6(f5.3,1x,f5.3,'*'))
200      continue
         write(lunout,*)'icond,ishift,ts,k,kslope,axe,rms,cor'
       endif
c
       ts = pdelay(i)
       k = kinit(i)
c
       ishift = 0
       icond = 0
c
       if(nused.eq.0)    return
c
c.......MAIN LOOP FOR THE ITERATIVE DETERMINATION OF THE CONDUCTIVITY
c       AND THE EFFECTIVE TIME ORIGIN
c
       do       300       j=1,ipmax
c
       icond = icond + 1
       kappa = ratbul(k)
c
             do 400   l=1,ipmax
c
             ishift = ishift + 1
             sum = 0.
c
c.......CALCULATION OF KPOINT USING TEMPERATURES
c
```

```
                    do       500       m=1,nused

                    dtime = time(m) + ts
                          if(dtime.lt.1.)  then
                             icond = 99
                             ishift = 99
                          return
                    endif

                    timinv(m) = 1./dtime
                    tau(m) = (kappa*dtime)/(a*a)
                    ftau(m) = ftable(tau(m))
                    temc(m) = tem(m)/(2.*alpha*tau(m)*ftau(m))
                    kpoint(m) = (power*timinv(m))/(4.*pi*temc(m))
                    sum = sum + kpoint(m)

500                 continue
c
              call least(timinv,temc,err,nused,axe,slope,vara,varb,
     $                  covar,cor,rms)
c
c.......CALCULATION OF KSLOPE USING THE SLOPE
c
              kslope=(power)/(4.*pi*slope)
              dk=kslope-(power)/(4.*pi*(slope+vara))
              kper=(dk/kslope)*100.
              kdif=k - kslope
c
c.......IF KDIF IS GREATER THEN KTOLER TS HAS TO BE ADJUSTED ITERATIVELY
c
             if(abs(kdif).ge.ktoler)   then
             if(long.ne.0)    then
             write(lunout,10)icond,ishift,ts,k,kslope,axe,rms,cor
10           format(1h ,2i4,f8.2,2f8.3,f8.3,e12.4,f8.4)
             endif
             else
             if(long.ne.0)    then
             write(lunout,10)icond,ishift,ts,k,kslope,axe,rms,cor
             endif
             goto  1000
             endif
c
c.......START VALUES FOR THE REGULA FALSI (FALSE POSITION)
c
              if(l.eq.1)       then
              start=kdif
              c=ts
              ts=ts+tsincr
              goto       400
              else
c
c.......CALCULATION OF THE TIME STEP
c
              zael=c*kdif-ts*start
              ts=zael/(kdif-start)
              endif
c
400              continue
```

```
c
1000       continue
c
       call fmvsd(kpoint,nused,kmean,kvar,ksigma)
c
       if(abs(k-kmean).le.ktoler)       goto 2000
c
c.......START VALUES FOR THE REGULA FALSI CONDUCTIVITY ITERATION
c
       if(j.eq.1)       then
       startf = axe
       cd = kmean
       k = kmean
       goto       300
       else
c
c.......CALCULATION OF THE CONDUCTIVITY STEP SIZE
c
       zael1 = cd*axe - kmean*startf
       k = zael1 / (axe-startf)
       endif
c
300       continue
c
c.......THIS POINT OF THE PROGRAM IS ONLY REACHED IF
c       K(ASSUMED)=K(SLOPE)=K(POINT)
c
2000       continue
c
       if(long.ne.0)   then
       write(lunout,*)'final values for sensor : ',i
       write(lunout,*)'icond,ishift,ts,k,kslope,kmean,ksigma,axe,rms,co
     $                r'
       write(lunout,20)icond,ishift,ts,k,kslope,kmean,ksigma,
     $                  axe,rms,cor
20     format(1h ,2i4,f7.2,4f8.4,f8.3,e12.4,f8.4)
       endif
c      .
c
c.......FINAL CALCULATION OF THE TEMPERATURE VS. F(ALPHA,TAU)
c       RELATION
c
       k = (kmean+kslope+k)/3.
       pdelay(i) = ts
       kappa = ratbul(kmean)
       do       600       j=1,nused
       tau(j)=((time(j)+ts)*kappa) / (a*a)
       ftau(j) = ftable(tau(j))
600       continue
c
       call least(ftau,tem,err,nused,axe,slope,vara,varb,
     $             covar,cor,rms)
c
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE COUT
c
c       SUBROUTINE PRINTS DETAILED TABLE OF THE  THERMAL CONDUCTIVITY
c       CALCULATIONS
c
c       VARIABLES
c       ---------
c       ALL VARIABLES ARE EXPLAINED IN COND OR HFRED
c
c**********************************************************************
c
       subroutine cout(npuls,npused,ktem,k,krms,pslope,
     $                  icond,ishift)

       implicit real(k)
       character answ*1
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/frico/fdelay(15),dincr,ifmax,tfmin,tfmax
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/out/lundat,lunout,long,lplot
        dimension ktem(nsensr),krms(nsensr),k(nsensr),
     $            pslope(nsensr),ishift(nsensr),
     $            npuls(nsensr),npused(nsensr),icond(nsensr)
c
       write(lunout,20)
       write(lunout,30)
       write(lunout,40)
       write(lunout,50)
c
       write(*,20)
       write(*,30)
       write(*,40)
       write(*,50)
c
c.......
c
       do       100       i=1,nsensr
       write(lunout,60)i,npuls(i),npused(i),ktem(i),k(i),
     $                 krms(i),pdelay(i),pslope(i),icond(i),ishift(i)
       write(*,60)i,npuls(i),npused(i),ktem(i),k(i),
     $                 krms(i),pdelay(i),pslope(i),icond(i),ishift(i)
100       continue
c
       write(lunout,70)
       write(*,70)
c
20       format(1h ,/,1h ,32x,'Heat Pulse Decay')
30       format(1h ,32x,'===============')
40       format(1h ,5x,'sensor',2x,'no of points',2x,'temp. at in-',
     $         2x,'conducti-',5x,'sd',4x,'delay',2x,'slope',2x,
     $          'ci',1x,'ti')
50       format(1h ,13x,'total / used',2x,'finity (deg)',2x,
```

```
     $         'vity(W/mK)',2x,'(W/mK)',3x,'(s)')
60        format(1h ,5x,i3,8x,i2,' / ',i2,7x,f6.4,7x,f5.3,6x,f4.3,3x,f5.1
     $           ,2x,f5.2,2x,i2,1x,i2)
70        format(1h ,/,6x,'Remark : If ci and ti = 0   k = k(initial)',/,
     $           1h ,  6x,'         If ci and ti = 99  time < 0')
c
c.......
c
          write(lunout,80)
80        format(1h ,//)
c
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE PLOTID
c
c       SUBROUTINE ASSIGNS A PLOT NUMBER BY READING A NUMBER FROM
c       A FILE AND ADDING 1.
c       THIS ROUTINE HELPS TO KEEP PRINTOUTS AND PLOTS TOGETHER.
c       IT COULD BY REPLACED BY USING DATE AND TIME ON
c       THE PLOTS AND PRINTOUTS.
c
c**********************************************************************
c
      subroutine plotid(pid)
      open(78,file='plotid.dat')
      read(78,*)pid
      pid=pid+1.
      rewind 78
      write(78,*)pid
      close(78)
      return
      end
```

```
c**********************************************************************
c
c       SUBROUTINE FINRES
c
c       SUBROUTINE PRINTS A TABLE WHICH CONTAINS THE FINAL RESULTS
c       OF THE SEDIMENT TEMPERATURES AND THE THERMAL CONDUCTIVITIES
c       FOR ALL SENSORS.
c       A THERMAL RESISTANCE IS CALCULATED FOR EACH DEPTH WHICH
c       IS USED TO CALCULATE AN INTERVAL HEAT FLOW.
c
c       VARIABLES
c       ---------
c       VARIABLES ARE EXPLAINED ELSEWHERE.
c
c**********************************************************************
c
       subroutine finres(istn,ipen,sedtem,k,pid,weit,npused)
c
       implicit real(k)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common /out/lundat,lunout,long,lplot
       dimension sedtem(nsensr),k(nsensr),bdepth(20),ipuls(15),
     $           npused(nsensr)
c
c.......
       write(*,*)
       write(*,*)'  ENTER WEIGHTING FACTOR FOR BULLARD-DEPTH'
       write(*,*)'     0. : LINEARLY VARYING CONDUCTIVITY'
       write(*,*)'     1. : SHARP VARIATIONS IN CONDUCTIVITY'
       read(*,*)weit
c
c.......
c
       call bdwt (k,nsensr,weit,dsensr,bdepth)
c
       write(lunout,20)istn,ipen
       write(lunout,30)
       write(lunout,40)
       write(lunout,50)
c
       write(*,20)istn,ipen
       write(*,30)
       write(*,40)
       write(*,50)
c
       l=0
c
       do       100       i=1,nsensr
c
       if(npused(i).eq.0)    then
       l=l+1
       ipuls(l) = i
       endif
c
c.......CALCULATION OF GRADIENT AND INTERVAL HEAT FLOW
```

```
c
       depth = (nsensr-i)*dsensr
       if(i.lt.nsensr)     then
       grad = (sedtem(i)-sedtem(i+1))/dsensr
       hf = (sedtem(i)-sedtem(i+1))/(bdepth(i)-bdepth(i+1))
       hf = hf*1000.
       grad = grad*1000.
       endif
c
       write(lunout,60)i,depth,sedtem(i),k(i),bdepth(i)
       write(*,60)i,depth,sedtem(i),k(i),bdepth(i)
       if(i.lt.nsensr)   then
         write(lunout,70) grad,hf
         write(*,70) grad,hf
       endif
c
100       continue
c
c.......
c
       write(lunout,80)weit,pid,(ipuls(j),j=1,l)
       write(*,80)weit,pid,(ipuls(j),j=1,l)
c
c.......
c
20       format(1h1,////,1h ,20x,'HEAT FLOW RESULTS FOR STA ',i2,
     $          ' PEN ',i2)
30       format(1h ,20x,35('='))
40       format(1h ,5x,'sensor',2x,'depth',2x,'sed.temp.',2x,'gradient',
     $         2x,'conductivity',2x,'Bullard-depth',2x,'heat flow')
50       format(1h ,14x,'(m)',5x,'(deg)',4x,'(mdeg/m)',5x,'(W/mk)',
     $         8x,'(m*mK/W)',5x,'(mW/m*m)')
60       format(1h ,5x,i3,6x,f4.2,3x,f6.3,17x,f4.2,11x,f5.3)
70       format(1h ,32x,f6.1,33x,f7.2)
80       format(1h ,/,1h ,5x,'Notes : ',/,
     $        1h ,5x,'(1)  weighting factor for Bullard-depth : ',f4.1
     $        ,/,1h ,5x,'(2) plot id  (0 = no plot) : ',f4.0
     $        ,/,1h ,5x,'(3) assumed conductivities for sensors :',15i3)
c
c.......
c
       return
       end
```

```
c******* ****************************************************************
c
c       SUBROUTINE RESULT
c
c       SUBROUTINE MAKES FOUR PLOTS :
c       1. F(ALPHA,TAU) VS. TEMPERATURE FOR THE FRICTIONAL DECAY
c       2. F(ALPHA,TAU) VS. TEMPERATURE FOR THE HEAT PULSE DECAY
c       3. TEMPERATURE AND THERMAL CONDUCTIVITY VS. DEPTH
c       4. THERMAL RESISTANCE VS. DEPTH (BULLARD PLOT)
c
c       THE PLOT SOFTWARE USED IS PLOT88 MADE BY:
c       PLOTWORKS INC.
c       P.O.BOX 12385
c       LA JOLLA CA 92037-0635
c       (619) 457-5090
c
c       VARIABLES
c       ---------
c       VARIABLES ARE EXPLAINED ELSEWHERE
c
c*********************************************************************
c
        subroutine result(nfused,sedtem,fslope,istn,ipen,
     $                    npused,k,ktem,pslope,pid,weit)
c
       implicit real (k)
       character answ*1
       common/pdata/tfplot(75,15),rfplot(75,15),
     $              tpplot(75,15),rpplot(75,15)
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
       common/condco/kinit(15),ktoler,tsinit,tsincr,tpmin,tpmax,
     $               ipmax,pdelay(15),power
       common/plotpa/fp(8),tp(8),cp(8),hp(8),bp(8)
       dimension nfused(nsensr),sedtem(nsensr),fslope(nsensr),k(nsensr),
     $           depth(20),work(20),bdepth(20),tem(20),
     $           npused(nsensr),ktem(nsensr),pslope(nsensr)
c
c.......ARRAYS FP,HP,TP,CP AND BP CONTAIN THE FOLLOWING PARAMETERS
c       (1) : LENGHT OF X-AXIS
c       (2) : LENGTH OF Y-AXIS
c       (3) : X-COORDINATE OF STATION NUMBER
c       (4) : Y-COORDINATE OF STATION NUMBER
c       (5) : START VALUE OF X-AXIS
c       (6) : START VALUE OF Y-AXIS
c       (7) : INCREMENTS PER INCH ON X-AXIS
c       (8) : INCREMENTS PER INCH ON Y-AXIS
c
c.......SET-UP OF THE PLOTTING DEVICES
c
5000   write(*,*)'  you have four options to plot the results '
       write(*,*)'    (1) plot on the screen  (enter 1)'
       write(*,*)'    (2) plot on the Hp7440A-Plotter (enter 2)'
       write(*,*)'    (3) plot on the printer (enter 3)'
       write(*,*)'    (4) create a plot-file HFRED.PLT (enter 4)'
       read(*,*)iopt
```

```
c
      if(iopt.eq.1)       then
         ioport=99
         model=99
      endif
c
      if(iopt.eq.2)       then
         ioport=9600
         model=80
c
         write(*,*)'  default port is COM1 '
         write(*,*)'  do you want to change to COM2 ?  y/n'
         read(*,10) answ
10       format(a1)
c
           if(answ.eq.'y'.or.answ.eq.'Y')    then
           ioport = 9600
           endif
c
        endif
c
      if(iopt.eq.3)       then
         ioport=0
         write(*,*)'default printer model is IBM Graphics Printer'
         write(*,*)'  do you want to change the model ?  y/n'
         read(*,10) answ
         if(answ.eq.'y'.or.answ.eq.'Y')    then
            write(*,*) 'enter model number (see PLOT88-Manual)'
            read(*,*) model
         else
            model=1
         endif
      endif
c
      if(iopt.eq.4) then
         write(*,*) 'enter model number (see PLOT88-Manual)'
         read(*,*) model
         ioport=10
         open(90,file='a:hfred.plt',form='unformatted')
         endif
c
c........INITIALIZATION OF THE PLOTTING DEVICE
c
       call plots(0,ioport,model)
c
       call window(0.,0.,9.75,7.)
c
       stn=istn
       pen=ipen
c---------------------------------------------
c.......PLOTTING OF FRICTIONAL DECAY
c---------------------------------------------
c
      nsum = 0
      do        800 i=1,nsensr
      nsum = nsum + nfused(i)
800   continue
      if(nsum.eq.0)  goto 1000
```

```
c
      call plot( 1.0, 1.0, -3 )
      call factor(cp(6))
c
      call symbol(fp(3),fp(4),.21,'STN ',0.0,4)
      call number(fp(3)+1.,fp(4),.21,stn,0.0,-1 )
      call number(fp(3)+1.6,fp(4),.21,pen,0.0,-1)
      call number(fp(3)+2.5,fp(4),.15,pid,0.0,-1)
      call axis(0.0,0.0,'F(alpha,tau)',-12,fp(1),0.0,fp(5),fp(7))
      call axis(0.0,0.0,'Temperature (deg c)',19,fp(2),90.,fp(6),fp(8))
c
      do       100       i=1,nsensr
c
      tem(i) =(fp(8)*fp(2)*(nsensr-i+1))/(nsensr+1)
c
      if(nfused(i).eq.0)    goto 100
c
      do       200       j=1,nfused(i)
      x = (tfplot(j,i)-fp(5))/fp(7)
      rfplot(j,i) = rfplot(j,i) - sedtem(i)
      rfplot(j,i) = rfplot(j,i) + tem(i)
      y = (rfplot(j,i)-fp(6))/fp(8)
c
      call plot(x,y,3)
      call plot(x,y,2)
c
200   continue
c
      y = (tem(i)-fp(6))/fp(8)
c
      call plot(0.0,y,3)
c
      x = tfplot(nfused(i),i)-0.2*fp(7)
      y = tem(i)+fslope(i)*x
      x = (x-fp(5))/fp(7)
      y = (y-fp(6))/fp(8)
c
      call plot(x,y,2)
c
      x = tfplot(1,i)+0.2*fp(7)
      y = tem(i)+fslope(i)*x
      x = (x-fp(5))/fp(7)
      y = (y-fp(6))/fp(8)
c
      call plot(x,y,3)
c
      x = fp(1)*fp(7)+fp(5)
      y = tem(i)+fslope(i)*x
      x = (x-fp(5))/fp(7)
      y = (y-fp(6))/fp(8)
c
      call plot(x,y,2)
c
100   continue
c
c-----------------------------------
c.....HEAT PULSE DECAYS.....
c-----------------------------------
```

```
c
      call factor(1.)
      call plot(4.5,0.,-3)
      call factor(cp(6))
c
      nsum = 0
      do        700 i=1,nsensr
      nsum = nsum + npused(i)
700      continue
      if(nsum.eq.0)  then
         call plot(0.,0.,-999)
         goto 3000
      endif
c
c
      call symbol(hp(3),hp(4),.21,'STN ',0.0,4)
      call number(hp(3)+1.,hp(4),.21,stn,0.0,-1 )
      call number(hp(3)+1.6,hp(4),.21,pen,0.0,-1)
      call number(hp(3)+2.5,hp(4),.15,pid,0.0,-1)
      call axis(0.0,0.0,'F(alpha,tau)',-12,hp(1),0.0,hp(5),hp(7))
      call axis(0.0,0.0,'Temperature (deg c)',19,hp(2),90.,hp(6),hp(8))
c
      do       400       i=1,nsensr
c
      if(npused(i).eq.0)   goto 400
c
      do       500       j=1,npused(i)
      rpplot(j,i) = rpplot(j,i) + 0.2*(nsensr-i)
      x = (tpplot(j,i)-hp(5))/hp(7)
      y = (rpplot(j,i)-hp(6))/hp(8)
c
      call plot(x,y,3)
      call plot(x,y,2)
c
500      continue
c
      axe = ktem(i) + (nsensr-i)*0.2
c
      y = (axe-hp(6))/hp(8)
c
      call plot(0.0,y,3)
c
      x = tpplot(npused(i),i)-0.2*hp(7)
      y = axe+pslope(i)*x
      x = (x-hp(5))/hp(7)
      y = (y-hp(6))/hp(8)
c
      call plot(x,y,2)
c
      x = tpplot(1,i)+0.2*hp(7)
      y = axe+pslope(i)*x
      x = (x-hp(5))/hp(7)
      y = (y-hp(6))/hp(8)
c
      call plot(x,y,3)
c
      x = hp(1)*hp(7)+hp(5)
      y = axe+pslope(i)*x
```

```
      x = (x-hp(5))/hp(7)
      y = (y-hp(6))/hp(8)
c
      call plot(x,y,2)
c
400       continue
c
      call plot(0.,0.,-999)
c
c-------------------------------------------------
c.......TEMPERATURE AND CONDUCTIVITY VS. DEPTH
c-------------------------------------------------
3000     continue
c
      call factor(1.)
      call window(0.,0.,9.75,7.)
      call plot(1.0,1.0,-3)
      call factor(cp(6))
c
c
      call symbol(tp(3),tp(4),.21,'STN ',0.0,4)
      call number(tp(3)+1.,tp(4),.21,stn,0.0,-1)
      call number(tp(3)+1.6,tp(4),.21,pen,0.0,-1)
      call number(tp(3)+2.5,tp(4),.15,pid,0.0,-1)
      call axis(0.0,8.0,'Temp (deg c)',12,tp(1),0.0,tp(5),tp(7))
       call axis(0.0,0.0,'Depth (m)' , 9,tp(2),90.,tp(6),tp(8))
c
      l = 0
c
      do       300       i=1,nsensr
      if(nfused(i).eq.0)  goto 300
      l = l + 1
      work(l)=sedtem(i)
      depth(l)=(nsensr-i)*dsensr
300   continue
c
      work(l+1) = tp(5)
      work(l+2) = tp(7)
      depth(l+1) = tp(6)
      depth(l+2) = tp(8)
c
      call line(work,depth,l,1,-1,3)
c
      off = tp(1)-cp(1)
      call plot(off,0.,-3)
c
      call axis(.0,.0,'K (W/mK)',-8,cp(1),.0,cp(5),cp(7))
c
      l = 0
      do       900       i=1,nsensr
      if(npused(i).eq.0)   goto 900
      l=l+1
      work(l)=k(i)
      depth(l) = (nsensr-i)*dsensr
900   continue
c
      work(l+1) = cp(5)
      work(l+2) = cp(7)
```

```
      depth(l+1) = tp(6)
      depth(l+2) = tp(8)
c
      call line(work,depth,l,1,-1,4)
c
c
c-------------------------------------------
c.......THERMAL RESISTANCE VS. DEPTH.........
c-------------------------------------------
c
1000  continue
c
      call plot(-off,0.,-3)
      call factor(1.)
      call plot(-1.,-1.,-3)
      call plot(4.5,0.,-3)
      call plot(1.0,1.0,-3)
      call factor(cp(6))
c
c
      call bdwt(k,nsensr,weit,dsensr,bdepth)
c
      l=0
      do       600     i=1,nsensr
      if(nfused(i).eq.0)   goto 600
      l=l+1
      work(l)=sedtem(i)
      bdepth(l)=bdepth(i)
600   continue
c
      bp(5) = work(l)
      bp(6) = bdepth(l)
      bp(7) = (work(l)-work(l))/bp(1)
      bp(8) = (bdepth(l)-bdepth(l))/bp(2)
      bp(8) = -bp(8)
      work(l+1) = bp(5)
      work(l+2) = bp(7)
      bdepth(l+1) = bp(6)
      bdepth(l+2) = bp(8)
c
      call symbol(bp(3),bp(4),.21,'STN ',0.0,4)
      call number(bp(3)+1.,bp(4),.21,stn,0.0,-1)
      call number(bp(3)+1.6,bp(4),.21,pen,0.0,-1)
      call number(bp(3)+2.5,bp(4),.15,pid,0.0,-1)
      call axis(0.0,8.0,'Temp (deg c)',12,bp(1),0.0,bp(5),bp(7))
      call axis(0.0,0.0,'Bullard - depth (m*m K/W)',25,bp(2),90.,bp(6),
     $           bp(8))
c
      call line(work,bdepth,l,1,-1,4)
c
      call plot(0.,0.,999)
c
      if(ioport.eq.99)       then
         write(*,*) 'do you want a hard copy of the plot ? y/n'
         read(*,10) answ
            if(answ.eq.'y'.or.answ.eq.'Y')    goto 5000
      endif
c
```

```
return
end
```

```
c**********************************************************************
c
c       SUBROUTINE ZERO
c
c       SUBROUTINE SETS ALL THE ELEMENTS OF THE ARRAYS LOADED TO ZERO
c
c**********************************************************************
c
        subroutine zero( nfric,npuls,rwater,
     $                   sedtem,estsed,sedrms,
     $                          fslope,nfused,
     $                   k,ktem,pslope,krms,npused,
     $                   icond,ishift,
     $                   work1,work2,time,tem)
c
       implicit real(k)
c
        dimension nfric(15),npuls(15),rwater(15),
     $            sedtem(15),estsed(15),sedrms(15),
     $                     fslope(15),nfused(15),
     $            k(15),ktem(15),pslope(15),krms(15),npused(15),
     $            icond(15),ishift(15),
     $            work1(75),work2(75),time(75),tem(75)
       common/decdat/tfric(75,15),rfric(75,15),
     $               tpuls(75,15),rpuls(75,15)
       common/pdata/tfplot(75,15),rfplot(75,15),
     $              tpplot(75,15),rpplot(75,15)
c
       do       100       i=1,15
       nfric(i) = 0
       npuls(i) = 0
       rwater(i) = 0.
       sedtem(i) = 0.
       estsed(i) = 0.
       sedrms(i) = 0.
       fslope(i) = 0.
       nfused(i) = 0
       npused(i) = 0
       k(i) = 0.
       ktem(i) = 0.
       pslope(i) = 0.
       krms(i) = 0.
       icond(i) = 0
       ishift(i) = 0
c
       do       200       l=1,75
       tfric(l,i) = 0.
       rfric(l,i) = 0.
       tpuls(l,i) = 0.
       rpuls(l,i) = 0.
       tfplot(l,i) = 0.
       rfplot(l,i) = 0.
       tpplot(l,i) = 0.
       rpplot(l,i) = 0.
       work1(l) = 0.
       work2(l) = 0.
       time(l) = 0.
```

```
        tem(l) = 0.
200       continue
100       continue
        return
        end
c
```

```
c**********************************************************************
c
c       SUBROUTINE BDWT
c
c       ORIGINAL VERSION BY MARK WIEDERSPAN (UTA)
c
c       MODIFIED BY H. VILLINGER DEC 1984
c
c ROUTINE TO CALCULATE THE BULLARD DEPTH WEIGHTINGS. THIS SCHEME WAS
c CONCOCTED BY LISTER TO ACHIEVE THESE ENDS: 1) TO ALLOW THE TWO HALVES
c AROUND AN AVERAGING SENSOR TO BE SEPERATELY CONSIDERED; 2) TO REDUCE
c TO THE SAME WEIGHTING AS THE FULL SPAN WEIGHTING FUNCTION.
c
c IT IS ASSUMED THAT THE PROBE IS OF THE FOLLOWING CONSTRUCTION:
c THERE ARE N (ODD) SENSORS IN A PROBE, N/2+1 POINT SENSORS WHICH
c MEASURE TEMPERATURE IN SOME SMALL REGION, AND N/2 SENSORS WHICH
c AVERAGE TEMPERATURE. THE LATTER ARE COMPRISED OF NINT EQUALLY SPACED
c SENSORS BETWEEN EACH POINT SENSOR. ALL SENSORS ARE IDENTICAL, BUT
c THE AVERAGING 'SENSOR' HAS ELEMENTS CONNECTED IN A SERIES PARALLEL
c COMBINATION TO YIELD THE SAME NOMINAL RESISTANCES AS SINGLE SENSORS.
c
c THIS ROUTINE WILL WEIGHT THE CONTRIBUTIONS OF THE AVERAGING
c SENSORS AND THE POINT SENSORS IN ORDER TO PRODUCE A BULLARD
c DEPTH FOR EACH SENSOR NOMINAL LOCATION. THE FUDGE PARAMETER W IS
c A WAY OF ALTERING THE WEIGHTING FUNCTION. IT WILL BE BETWEEN
c 0 AND 1, THE LATTER APPROPRIATE FOR A MORE SHARPLY CURVING
c CONDUCTIVITY VS. DEPTH RELATION.
c
c TOPHAF AND BOTHAF DO THE WEIGHTING FOR THE TOP AND BOTTOM HALVES
c OF THE AVERAGING SECTION PLUS THE POINT SENSORS AT EACH END
c OF THAT SECTION.
c
c NO ERRORS ARE DERIVED
c
c MAIN MODIFICATION : NO AVERAGING SENSORS INCORPORATED IN THE PROGRAM
c
c**********************************************************************
c
      subroutine bdwt( kmean, n, w, dsensr,
     $              bdepth )
      real kmean(n)
      integer n
      real w
      integer nint
      real dsensr
      real bdepth(n)
c
C THE BOTTOM SENSOR IS 1, THE TOP IS N.
c
      nint = 1
      bdepth(n) = 0.0
c
C BDINC IS THE INCREMENTAL RESISTIVITY ASSOCIATED WITH AN INTERVAL
C THE EVEN ELEMENTS ARE THE AVERAGING SENSORS; ODDS ARE POINT SENSORS
C BDEPTH CONTAINS THE SUM OF THE INCREMENTAL RESISTIVITIES TO A DEPTH
c
      do 100 i = n-1, 1, -1
```

```
          if( mod(i,2) .eq. 0 ) then
              bdinc = tophaf( kmean,i,w,nint )
            else
              bdinc = bothaf( kmean,i,w,nint )
          endif
          bdepth(i) = bdepth(i+1) + bdinc
100        continue
c
C COMPUTE BULLARD DEPTH = INTER SENSOR DISTANCE * SUM OF RESISTIVITIES
c
        do 200 i=1,n
          bdepth(i) = dsensr * bdepth(i)
200        continue
c
        return
        end
c
c
        real function tophaf( kmean, i, w, nint )
        real kmean(1)
        integer i
        real w
        integer nint
c
c
        tophaf = 1.0 / (1.0 + 1.0/nint) *
     $               ( (1.0/nint + w) / kmean(i+1)  +
     $               1.0 / kmean(i)  -
     $               w / kmean(i-1)  )
c
        return
        end
c
c
        real function bothaf( kmean, i, w, nint )
        real kmean(1)
        integer i
        real w
        integer nint
c
c
        bothaf = 1.0 / (1.0 + 1.0/nint) *
     $               ( -w / kmean(i+2)  +
     $               1.0 / kmean(i+1) +
     $               (w + 1.0/nint) / kmean(i)  )
c
        return
        end
```

```
c**********************************************************************
c
c       FUNCTION RATBUL
c
c       FUNCTION OF THE CONDUCTIVITY - DIFFUSIVITY - RELATIONSHIP
c
c       VARIABLE
c       --------
c       K     : THERMAL CONDUCTIVITY
c       KAPPA : THERMAL DIFFUSIVITY
c       RATCL : COEFFICENTS FROM HYNDMAN ET AL. 1979
c               MARINE GEOPYS. RES.,4,181-205
c               OR AT THE USER'S DISCRETION
c
c**********************************************************************
c
       function ratbul(k)
c
       real k,kappa
c
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
c
       kappa = ratcl(1) - ratcl(2) * k + ratcl(3) * k ** 2
       kappa = k / kappa
        ratbul = kappa * 1.0e-6
c
       return
       end
```

```
c**********************************************************************
c
c       FUNCTION TEMP
c
c       FUNCTION CONVERTS READINGS IN TEMPERATURES USING CALIBRATION
c       COEFFICIENTS SUPPLIED IN THE PARAMETER FILE HFRED.PAR
C
c       VARIABLES
c       ---------
c
c       X        : READING
c       L        : SENSOR NUMBER
c       AA,BB,CC : CALIBRATION COEFFICIENTS
c
c**********************************************************************
c
c
       real function temp(x,l)
c
       common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $               aa(15),bb(15),cc(15),
     $               ratcl(3),f(5000),alpha,fstart,fincr
c
        temp=aa(l) * x **2 + bb(l) * x + cc(l)
c
       return
       end
```

```
c**********************************************************************
c
c       SUBROUTINE LEAST
c
c       SUBROUTINE CALCULATES A LINEAR LEAST SQUARE FIT USING THE
c       ASSUMED DATA ERRORS AS WEIGHTS
c
c       VARIABLES
c       ---------
c       X/Y : INPUT DATA
c       ER : ERRORS OF THE Y-VALUES
c       N : NUMBER OF DATA POINTS
c       AXE : INTERCEPT VALUE FOR X=0
c       SLOPE : SLOPE OF THE LINEAR FIT
c       VARA : VARIANCE OF AXE
c       VARB : VARAINCE OF THE SLOPE
c       COVAR : COVARIANCE BETWEEN AXE AND SLOPE
c       COR : CORRELATION  COEFFICIENT
c       RMS : STANDART DEVIATION OF THE FIT
c
c**********************************************************************
c
       subroutine least(x,y,er,n,axe,slope,vara,varb,covar,cor,rms)
c
       double precision a,b,c,d,e,f,det
       dimension x(n),y(n),er(n)
c
       a=0.d0
       b=0.d0
       c=0.d0
       d=0.d0
       e=0.d0
       f=0.d0
c
       do       100       i=1,n
       eror=er(i)**2.d0
       a=a+x(i)/eror
       b=b+1.d0/eror
       c=c+y(i)/eror
       d=d+(x(i)**2.d0)/eror
       e=e+(x(i)*y(i))/eror
       f=f+(y(i)**2.d0)/eror
100       continue
c
       det=b*d-a*a
       slope=sngl((e*b-c*a)/det)
       axe=sngl((d*c-e*a)/det)
       vara=sqrt(sngl(b/det))
       varb=sqrt(sngl(d/det))
       covar=sngl(-a/det)
c
       rms=0.
       do       200       i=1,n
       w=(y(i)-slope*x(i)-axe)**2
       rms=rms+w
200       continue
       rms=sqrt(rms/(n-2))
```

```
c
      det=dble(n)*e-a*c
      b=(dble(n)*d-a*a)*(dble(n)*f-c*c)
      b=det/dsqrt(b)
      cor=sngl(b)
c
      return
      end
```

```
C*************************************************************************
C
C       FUNCTION FTABLE
C
C       FUNCTION FTABLE INTERPOLATES VALUES FROM THE F( , ) TABLE
C       WHICH WAS GENERATED USING THE PROGRAM FATAU
C
C*************************************************************************
C
        function ftable(x)
c
        common/sensor/nsensr,tcycle,deltat,a,dsensr,terror,plen,
     $                aa(15),bb(15),cc(15),
     $                ratcl(3),f(5000),alpha,fstart,fincr
c
        if(x.lt.9.9)    then
        start = (x-fstart) / fincr
        start = aint(start)
        n = ifix(start)
        p = x - (start*fincr + fstart)
        p = p / fincr
        ftable = (1.-p) * f(n+4) + f(n+5)*p
c
        else
c
        ftable = (1. / (2.*alpha*x)) - (1. / (4.*alpha*x*x))
c
        endif
c
c
        return
        end
```

```
C*************************************************************************
C
C       SUBROUTINE FMVST
C
C       SUBROUTINE TO COMPUTE MEAN, VARIANCE AND STD OF AN ARRAY
C
C       AUTHOR : MARK WIEDERSPAN, U. TEXAS, AUSTIN
C
C*************************************************************************
C
        subroutine fmvsd(x,n,xmean,xvar,xsigma)
c
        double precision sum
c
c
        sum = 0.0d0
        do      100     i=1,n
                sum = sum +x(i)
100     continue
c
        xmean = sum / n
c
        sum = 0.0d0
        do      200     i=1,n
        sum = sum + (x(i)-xmean)**2
200     continue
c
        xvar = sum / n
c
        xsigma = dsqrt( sum / (n-1))
c
        return
        end
```

Example of the parameter file HFRED.PAR

```
7                                       no of sensors
1.                                      length of a cycle in sec
1.0                                     time increment
0.00397                                 radius of the sensor
0.3700                                  distance between sensors
0.004                                   assumed temperature error
28.125                                  length of heat pulse
0.,0.,0.,0.,0.,0.,0.                    aa  |coefficients for
1.9,1.9,1.9,1.9,1.9,1.9,1.9             bb  >function TEMP
0.,0.,0.,0.,0.,0.,0.                    cc  |
5.79 3.67 1.016                         kappa=function(k)-coeff.
10. 10. 10. 10. 10. 10. 10.             frictional time delays
20 -2.0                                 ifmax   time increment
2.0  10.                                min. and max. tau-values
10. 10. 10. 10. 10. 10. -30.            pulse time delays
1.00 1.00 1.00 1.00 1.00 1.00 1.00      initial conductivities
582.00                                  power per length
10. 1.                                  init. time shift & incr.
10 0.002                                ipmax k-tolerance
2.0  10.                                min. and max. tau-values
6.0 8.0 3.0 0.1 0.0 0.0 0.03 0.20       fp  |
5.0 8.0 1.5 0.1 0.0 2.4 0.30 -0.300     tp  | plot parameters
3.0 0.0 0.0 0.0 0.8 0.7 0.1 -0.300      cp  | description see
6.0 8.0 3.0 0.1 0.0 0.0 0.03 0.20       hp  | result.for
5.0 8.0 1.5 0.1 0.0 3.0 0.7 -0.500      bp  |
3 6 0 0                                 lundat,lunout,long,lplot
```

Example of a heat flow penetration data file

```
c       THIS DATA SET IS SUPPLIED WITH THE SOURCE CODE OF HFRED
c       TO ALLOW CHECKS OF THE PROGRAM AFTER IMPLEMENTATION.
c
c       THE ROUTINES TO READ THAT DATA FILE ARE TITLE AND READ.
c       THEY CAN BE EASILY CHANGED AT THE USER'S DISCRETION.
c       MAXIMUM NUMBER OF TEMPERATURE CHANNELS : 15
c       MAXIMUM NUMBER OF DATA POINTS FOR FRICTIONAL OR HEAT
c       PULSE DECAY : 75.
c
c       THE USER OF THE ALGORITHM SHOULD BE AWARE OF THE FACT
c       THAT IN THE IDEAL CASE WITH PERFECT DATA THE DATA SET
c       SHOULD BE DECIMATED AT THIS POINT TO EQUALLY SPACED
c       POINTS IN THE 1/TIME-SPACE. FROM OUR EXPERIENCE WITH
c       DIFFERENT HEAT FLOW INSTRUMENTS AND THEIR CHARACTERISTIC
c       NOISE PROBLEMS, HOWEVER, INCLUSION OF ADDITIONAL POINTS MAY BE
c       STATISTICALLY WARRENTED.
```

	5		1	' PGC84-4'				Header information
	48		128		2400			describing location
	1		1	7				etc.
	3440							see TITLE for
	3960							details
	780	749	769	727	705	714	729	Water temperatures
3450	1399	1269	1199	1094	1995	853	799	start frictional
3460	1571	1407	1305	1171	1031	892	821	decay
3470	1651	1475	1362	1211	1063	917	836	
3480	1689	1509	1394	1234	1083	933	847	col #1 : time in sec
3490	1709	1529	1413	1248	1096	945	855	
3500	1721	1540	1426	1258	1104	954	861	col #2-8 : temperature
3510	1727	1548	1435	1265	1110	962	865	sensors 1 - 7
3520	1733	1554	1441	1270	1115	967	868	
3530	1735	1558	1446	1274	1118	972	871	
3540	1738	1562	1450	1277	1121	976	873	
3550	1739	1564	1453	1280	1123	980	874	
3560	1741	1566	1455	1282	1125	982	876	
3570	1742	1568	1457	1285	1127	985	877	
3580	1743	1570	1459	1286	1129	988	878	
3590	1743	1571	1461	1288	1130	990	879	
3600	1744	1573	1462	1289	1131	991	880	
3610	1745	1573	1463	1290	1132	994	881	
3620	1745	1574	1465	1291	1133	995	881	
3630	1745	1575	1465	1292	1133	996	882	
3640	1745	1576	1466	1293	1134	998	883	
3650	1746	1577	1467	1294	1135	999	883	
3660	1746	1577	1468	1294	1135	1000	884	
3670	1746	1578	1469	1295	1136	1002	885	
3680	1746	1579	1469	1296	1136	1003	885	
3690	1747	1579	1470	1296	1137	1004	886	
3700	1747	1579	1470	1297	1137	1004	886	
3710	1747	1580	1471	1297	1137	1005	887	
3720	1747	1580	1471	1297	1138	1006	887	
3730	1747	1581	1471	1298	1138	1007	887	
3740	1747	1581	1472	1298	1138	1008	888	

3750	1747	1581	1472	1298	1138	1008	889	
3760	1747	1581	1473	1298	1139	1009	889	
3770	1747	1581	1473	1299	1139	1010	889	
3780	1747	1582	1473	1300	1139	1010	890	
3790	1748	1582	1473	1300	1140	1011	890	
3800	1748	1582	1474	1300	1140	1011	890	
3810	1748	1582	1474	1300	1140	1012	891	
3820	1748	1583	1474	1300	1140	1012	891	
3830	1748	1583	1474	1300	1140	1012	891	
3840	1748	1583	1475	1300	1141	1013	891	
3850	1748	1583	1475	1301	1141	1013	892	
3860	1748	1583	1475	1301	1141	1013	892	
3870	1748	1583	1475	1301	1141	1014	892	
3880	1748	1583	1475	1301	1141	1014	893	
3890	1748	1584	1476	1301	1141	1015	893	
3900	1748	1584	1476	1301	1141	1015	893	
3910	1748	1584	1476	1301	1141	1015	893	
3920	1748	1584	1476	1301	1141	1015	894	
3930	1748	1584	1476	1301	1141	1015	894	
3940	1748	1584	1476	1302	1141	1016	894	
4020	2500	2263	2227	1888	1736	1650	1023	Start heat pulse
4030	2283	2083	2012	1748	1585	1498	1023	decay
4040	2159	1980	1890	1665	1498	1405	1791	
4050	2083	1914	1816	1610	1441	1345	1562	
4060	2034	1870	1765	1571	1402	1302	1412	
4070	1998	1838	1730	1541	1373	1270	1308	
4080	1971	1813	1702	1518	1351	1245	1236	
4090	1949	1792	1681	1500	1333	1225	1184	
4100	1932	1775	1664	1484	1317	1208	1146	
4110	1917	1761	1650	1470	1305	1194	1118	
4120	1905	1749	1638	1459	1294	1182	1096	
4130	1895	1739	1628	1449	1284	1172	1078	
4140	1885	1730	1618	1440	1276	1162	1064	
4150	1877	1722	1610	1433	1268	1154	1052	
4160	1870	1714	1604	1426	1262	1147	1042	
4170	1863	1708	1597	1420	1256	1140	1033	
4180	1858	1702	1591	1414	1250	1134	1026	
4190	1852	1697	1586	1409	1245	1129	1020	
4200	1847	1692	1581	1404	1241	1124	1013	
4210	1843	1687	1577	1400	1237	1120	1008	
4220	1839	1684	1573	1396	1234	1116	1004	
4230	1835	1680	1569	1393	1230	1112	999	
4240	1832	1677	1566	1389	1227	1109	995	
4250	1829	1673	1563	1386	1224	1106	992	
4260	1826	1670	1560	1383	1221	1103	988	
4270	1823	1668	1557	1380	1219	1100	985	
4280	1821	1665	1555	1378	1216	1098	982	
4290	1818	1662	1553	1376	1214	1096	980	
4300	1816	1661	1550	1374	1212	1093	978	
4310	1814	1658	1548	1372	1210	1091	975	
4320	1812	1656	1546	1369	1208	1089	973	
4330	1810	1654	1544	1368	1206	1087	971	
4340	1808	1652	1542	1366	1205	1085	969	
4350	1806	1650	1541	1364	1203	1084	967	
4360	1805	1649	1539	1362	1201	1082	965	
4370	1803	1647	1537	1361	1200	1081	964	
4380	1802	1646	1536	1360	1199	1079	962	
4390	1800	1644	1535	1358	1197	1078	961	

4400 1799 1643 1533 1357 1196 1077 959

Complete example of a reduction of a heat flow penetration

```
**************************************************
*                                                *
*     Cruise: PGC84-4                            *
*     Station:    5        Penetration:    1     *
*     Instrument:    1     Sensor:    1          *
*     Latitude:  48.000    Longitude:  128.000   *
*     Depth(m):2400.                             *
*                                                *
**************************************************
```

PARAMETERS
==========

number of sensors : 7
cycle time (s) : 1.0
time increment between sensor readings (s) : 1.00
radius of the sensor string (m) : .00397
power input per length (J/m) : 582.00
frictional time delays (s) 10.0 10.0 10.0 10.0 10.0 10.0 10.0
min. tau for frictional decay : 2.00
pulse time delays (s) 10.0 10.0 10.0 10.0 10.0 10.0 -30.0
initial thermal cond. 1.00 1.00 1.00 1.00 1.00 1.00 1.00
initial time-shift and time-shift increment 10.00 1.00
max. thermal cond. error in COND : .0020
min. tau for heat pulse decay : 2.00
alpha-value used : 2.00
increment of F(alpha,tau)-table : .010

FRICTIONAL DECAY
================

sensor	no of points total / used	sed.temp. (deg)	95% level error (deg)	gradient (mdeg/m)	delay (s)	slope	water temp.
1	50 / 40	1.845	.0017	667.6	-28.0	-.179	1.482
2	50 / 40	1.598	.0013	651.2	-28.0	-.429	1.423
3	50 / 40	1.357	.0012	682.8	-25.0	-.527	1.461
4	50 / 40	1.104	.0016	714.9	-28.0	-.475	1.381
5	50 / 40	.840	.0013	647.7	-28.0	-.403	1.339
6	50 / 39	.600	.0017	742.1	7.0	-1.154	1.357
7	50 / 39	.325	.0027		7.0	-.617	1.385

Heat Pulse Decay
================

sensor	no of points total / used	temp. at infinity (deg)	conductivity(W/mK)	sd (W/mK)	delay (s)	slope	ci	ti
1	39 / 33	-.0002	1.172	.001	-32.7	4.086	3	15
2	39 / 33	.0001	1.021	.001	-26.2	3.789	3	13
3	39 / 33	.0001	1.070	.001	-29.1	3.884	3	13
4	39 / 34	.0000	1.093	.001	-27.8	3.933	3	13
5	39 / 34	.0001	1.101	.001	-25.7	3.941	3	12
6	39 / 34	.0000	1.055	.001	-32.7	3.857	3	14
7	37 / 34	.0018	1.066	.010	-67.0	3.810	10	23

Remark : If ci and ti = 0 k = k(initial)

HEAT FLOW RESULTS FOR STA 5 PEN 1

sensor	depth (m)	sed.temp. (deg)	gradient (mdeg/m)	conductivity (W/mk)	Bullard-depth (m*mK/W)	heat flow (mW/m*m)
1	2.22	1.845		1.17	2.064	
			667.6			728.81
2	1.85	1.598		1.02	1.725	
			651.2			680.79
3	1.48	1.357		1.07	1.371	
			682.8			738.72
4	1.11	1.104		1.09	1.029	
			714.9			784.56
5	.74	.840		1.10	.692	
			647.7			698.07
6	.37	.600		1.06	.349	
			742.1			786.90
7	.00	.325		1.07	.000	

Notes :
(1) weighting factor for Bullard-depth : .0
(2) plot id (0 = no plot) : 0.
(3) assumed conductivities for sensors :

PARAMETERS
==========

number of sensors : 7
cycle time (s) : 1.0
time increment between sensor readings (s) : 1.00
radius of the sensor string (m) : .00397
power input per length (J/m) : 582.00
frictional time delays (s) 10.0 10.0 10.0 10.0 10.0 10.0 10.0
min. tau for frictional decay : 2.00
pulse time delays (s) -22.9 -18.3 -20.3 -19.5 -18.0 -22.9 -46.9
initial thermal cond. 1.17 1.02 1.07 1.09 1.10 1.06 1.07
initial time-shift and time-shift increment 10.00 1.00
max. thermal cond. error in COND : .0020
min. tau for heat pulse decay : 2.00
alpha-value used : 2.00
increment of F(alpha,tau)-table : .010

FRICTIONAL DECAY
================

sensor	no of points total / used	sed.temp. (deg)	95% level error (deg)	gradient (mdeg/m)	delay (s)	slope	water temp.
1	50 / 31	1.846	.0022	671.7	-28.0	-.251	1.482
2	50 / 38	1.598	.0013	652.5	-28.0	-.441	1.423
3	50 / 36	1.356	.0012	681.4	-25.0	-.567	1.461
4	50 / 34	1.104	.0013	715.2	-28.0	-.532	1.381
5	50 / 34	.840	.0012	649.6	-28.0	-.454	1.339
6	50 / 37	.599	.0019	744.4	7.0	-1.224	1.357
7	50 / 36	.324	.0026		7.0	-.638	1.385

Heat Pulse Decay
================

sensor	no of points total / used	temp. at infinity (deg)	conductivity(W/mK)	sd (W/mK)	delay (s)	slope	ci	ti
1	39 / 31	-.0002	1.181	.001	-32.8	4.099	3	9
2	39 / 34	-.0002	1.022	.001	-26.5	3.795	1	4
3	39 / 34	-.0001	1.070	.001	-29.6	3.887	1	4
4	39 / 35	-.0003	1.093	.001	-27.7	3.940	1	4
5	39 / 34	-.0003	1.101	.001	-25.9	3.951	1	4
6	39 / 34	.0001	1.052	.001	-32.6	3.848	2	5
7	37 / 35	.0011	1.064	.011	-68.4	3.811	1	10

Remark : If ci and ti = 0 k = k(initial)

HEAT FLOW RESULTS FOR STA 5 PEN 1

sensor	depth (m)	sed.temp. (deg)	gradient (mdeg/m)	conductivity (W/mk)	Bullard-depth (m*mK/W)	heat flow (mW/m*m)
1	2.22	1.846		1.18	2.064	
			671.7			735.87
2	1.85	1.598		1.02	1.727	
			652.5			681.98
3	1.48	1.356		1.07	1.373	
			681.4			737.05
4	1.11	1.104		1.09	1.031	
			715.2			784.88
5	.74	.840		1.10	.693	
			649.6			699.13
6	.37	.599		1.05	.350	
			744.4			787.75
7	.00	.324		1.06	.000	

Notes :
(1) weighting factor for Bullard-depth : .0
(2) plot id (0 = no plot) : 0.
(3) assumed conductivities for sensors :

Listing of the program FTABLE and the subroutines used

```
C*******************************************************************************
C
C       PROGRAM TO CALCULATE AN F( , ) TABLE WITH A   VALUE WHICH
C       HAS TO CHOSEN AT RUN TIME AND WRITES THE TABLE TO A FILE.
C
C       THE FILE NAME WILL BE ASSIGNED AT RUNTIME AND SHOULD BE
C       CHOSEN IN THE FOLLOWING WAY : F<VALUE OF ALPHA>.TAB.
C
C
C       REFERENCE
C       ---------
C       JAEGER, J.C. (1956)
C       CONDUCTION OF HEAT IN AN INFINITE REGION BOUNDED INTERNALLY BY A
C       CIRCULAR CYLINDER OF A PERFECT CONDUCTOR.
C       AUSTRAL.J.PHYSICS,9,167-179.
C
C       VARIABLES
C       --------
C       ALPHA : VALUE OF ALPHA
C       START : START VALUE OF
C       DTAU : INCREMENT IN
C
C       THE SUBROUTINES WERE ALL INHERITED FROM LINDA MEINKE, WHO WROTE
C       THEM AT ABOUT 1980 AT THE MIT, BOSTON (MASS.).
C       WE THEREFORE DO NOT COMMENT THEM IN DETAIL BUT DESCRIBE ONLY
C       BRIEFLY WHAT THEY ARE DOING.
C
C       FATEV : SELECTS THE INTEGRATION BOUNDS USED FOR A SPECIFIC
C                 VALUE
C       DQG12 : NUMERICAL INTEGRATION
C       BES : CALCULATION OF BESSEL FUNCTIONS
C
C*******************************************************************************

      program ftable

      implicit double precision (a-h,o-z)
      real ftau(1000),al,dtau,start
      common /one/alpha,a,tau,pi

      write(0,*)'enter alpha, min.tau value and tau-increment'
      read(0,*)al,start,dtau
      a=1.d0

      open(3,file=' ',status='new')

      write(3,*)al,start,dtau

      pi=4.d0*datan(1.d0)
      alpha=dble(al)
      dt=dble(dtau)
      tau=dble(start)-dt
      l=0

      do       100       i=1,1000
      tau=tau+dt
      call fatev(v)
```

```
      ftau(i)=sngl(v)
      if(tau.gt.10.d0)   goto 200
      l=l+1
      write(0,*)l,tau,ftau(i)
100      continue

200      write(3,*)(ftau(i),i=1,l)

      stop
      end
```

```
      SUBROUTINE  FATEV(F)
C
      DOUBLE PRECISION ZINF,PI,ALPHA,TAU,F1,F,F2,GAM,F3,F4,A,
     1 XK(7),AA(7,3),BB(7,3),CC(7,3),QQ,DQG12,TODEL
      COMMON/ONE/ALPHA,A,TAU,PI,XK,AA,BB,CC,QQ,ISTRNG,INSTR,RR,RL,STN,
     1 PEN,TODEL
C
C-----------------------------------------------------------------------
C
C
      F1=4.D0*ALPHA/(PI*PI)
      IF(TAU.GT.3.0D0)GO TO 5
        ZINF=4.D0
        F=F1*DQG12(0.D0,ZINF)
        GOTO 100
C
    5 IF(TAU.GT.6.D0)GO TO 10
        ZINF=2.5D0
        F=F1*DQG12(0.D0,ZINF)
        GOTO 100
C
   10 IF(TAU.GT.10.D0)GO TO 15
        ZINF=2.5D0
        F=F1*DQG12(0.D0,ZINF)
        GOTO 100
C
   15 F2=1.0D0/(2.0D0*ALPHA*TAU)
        GAM=DEXP(.577215665D0)
        F3=DLOG(4.0D0*TAU/GAM)
        F4=(2.0D0+(ALPHA-2.0D0)*F3)*(F2*F2)
        F=F2-F4
C
100   CONTINUE
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION DQG12(XL,XU)
C     XL - DOUBLE PRECISION LOWER BOUND OF THE INTERVAL
C     XU - DOUBLE PRECISION UPPER BOUND OF THE INTERVAL
C     FAT - EXTERNAL DOUBLE PRECISION FUNCTION USED
C     Y - THE RESULTING DOUBLE PRECISION INTEGRAL VALUE
C     METHOD
C
C     EVALUATION IS DONE BY MEANS OF 12 POINT GAUSS QUADRATURE
C     FORMULA, WHICH INTEGRATES POLYNOMIALS UP TO DEGREE 23
C     EXACTLY.  SEE SCIENTIFIC SUBROUTINE PACKAGE.
C     FROM V.I. KRYLOV, APPROXIMATE CALCULATION OF INTEGRALS,
C     MACMILLAN, NEW YORK/LONDON, 1962, PP.100-111 AND 337 - 340
      DOUBLE PRECISION XL,XU,Y,A,B,C,FAT
C
      A=.5D0*(XU+XL)
      B=XU-XL
      C=.4907803171233596 3D0*B
      Y=.2358766819325591 4D-1*(FAT(A+C)+FAT(A-C))
      C=.4520586281852374 3D0*B
      Y=Y+.5346966299765921 5D-1*(FAT(A+C)+FAT(A-C))
      C=.3849513370971523 4D0*B
      Y=Y+.8003916427167311D-1*(FAT(A+C)+FAT(A-C))
      C=.2936589771433087 2D0*B
      Y=Y+.1015837133615329 6D0*(FAT(A+C)+FAT(A-C))
      C=.1839157494990901 0D0*B
      Y=Y+.1167462682691774 0D0*(FAT(A+C)+FAT(A-C))
      C=.6261670425573445 8D-1*B
      DQG12=B*(Y+.1245735229067013 9D0*(FAT(A+C)+FAT(A-C)))
      RETURN
      END
```

```
      SUBROUTINE BES(NO,X,T,Y)
C
      DOUBLE PRECISION T(200),Y(200),X,TOPI,F,SUM,TYO,SYO,S,XN
C
C---------------------------------------------------------------------
C
C
 107  FORMAT('Negative order not accepted in bessel function routine.')
      KODE=0
      IOUT=6
      TOPI=0.6366197734D0
      KLAM=1
      KO=NO+1
      IF(X.NE.0.D0)GO TO 6
      GO TO 4
 5    WRITE(IOUT,107)
    4 STOP
 6    IF(NO.LT.0)GO TO 5
 7    IF(KODE.EQ.0)GO TO 9
 8    KLAM=KLAM+1
 9    JO=2*IFIX(SNGL(X))
      MO=NO
      IF((MO-JO) .GE.0)GO TO 12
 11   MO=JO
 12   MO=MO+11
      T(MO)=0.D0
      LUB=MO-1
      T(LUB)=1.0D-35
      GO TO (23,51),KLAM
 23   F=DBLE(FLOAT(2*LUB))
      MO=MO-3
      I2=MO
 24   F=F-2.D0
      T(I2+1)=F/X*T(I2+2)-T(I2+3)
      IF(I2)25,26,25
 25   I2=I2-1
      GO TO 24
 26   SUM=T(1)
      DO 40 J=3,MO,2
        SUM=SUM+2.D0*T(J)
40    CONTINUE
      F=1.D0/SUM
      DO 50 J=1,MO
        T(J)=T(J)*F
50    CONTINUE
      TYO=TOPI*T(1)*(DLOG(.5D0*X)+0.577215665D0)
      MOO2=MO/2
      SYO=0.D0
      S=-1.D0
      DO 101 K=1,MOO2
        XN=DBLE(FLOAT(K))
        KK=2*K+1
        SYO=SYO+S*T(KK)/XN
        S=-S
101   CONTINUE
      Y(1)=TYO-2.D0*TOPI*SYO
      Y(2)=(T(2)*Y(1)-TOPI/X)/T(1)
```

```
      DO 102 K=3,KO
        Y(K)=2.DO*DBLE(FLOAT(K-2))/X*Y(K-1)-Y(K-2)
102   CONTINUE
      RETURN
C
 51   F=DBLE(FLOAT(2*LUB-2))
      MO=MO-3
      I2=MO
 511  T(I2+1)=F/X*T(I2+2)+T(I2+3)
      IF(I2.EQ.0)GO TO 53
 52   I2=I2-1
      F=F-2.DO
      GO TO 511
C
 53   SUM=T(1)
      DO 70 J=2,MO
        SUM=SUM+2.DO*T(J)
70    CONTINUE
      F=1.DO/SUM*DEXP(X)
      DO 80 J=1,KO
        T(J)=T(J)*F
80    CONTINUE
C
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION FAT(X)
C
      DOUBLE PRECISION X,ALPHA,T1,T2,TN,TAU,BJ(200),A,BY(200),
     1 PI,XK(7),AA(7,3),BB(7,3),CC(7,3),QQ,RR,RL,TODEL
      COMMON/ONE/ALPHA,A,TAU,PI,XK,AA,BB,CC,QQ,ISTRNG,INSTR,RR,RL,STN,
     1 PEN,TODEL
C
C-----------------------------------------------------------------------
C
C
      IF( X.EQ.0.DO )GO TO 100
          CALL BES(2,X,BJ,BY)
          TN=DEXP(-TAU*X*X)
          T1=X*BY(1)-ALPHA*BY(2)
          T2=X*BJ(1)-ALPHA*BJ(2)
          FAT=TN/(X*(T1*T1+T2*T2))
          GOTO 200
100     CONTINUE
          FAT=0.DO
200   CONTINUE
      END
```

REFERENCES

Davis, E.E. (1988)
Oceanic Heat-Flow Density, in R. Haenel, L. Rybach, and L. Stegena (eds.), Handbook of Terrestrial Heat-Flow Density Determination, D. Reidel Publishing Co., Dordrecht, 223-260.

Davis, E.E., Lister, C.R.B., and Sclater, J.G. (1984)
Towards determining the thermal state of old ocean lithospere : heat flow measurements from the Blake-Bahama outer ridge,north-western Atlantic.
Geophys. J. R. Astr. Soc. ,78, 507-545.

Hyndman, R.D., Davis, E.E., and Wright, J.A. (1979)
The measurement of marine geothermal heat flow by a multi-penetration probe with digital acoustic telemetry and in situ thermal conductivity.
Marine Geophysical Researches, 4, 181-205.

Lister, C.R.B. (1979)
The pulse-probe method of conductivity measurement.
Geophys.J.R.Astr.Soc., 57, 451-461.

Villinger, H., and Davis, E.E. (1987)
A new reduction algorithm for marine heat flow measurements.
J. Geophys. Res., 92, 12846-12856.

JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 92, NO. B12, PAGES 12,846–12,856, NOVEMBER 10, 1987

A New Reduction Algorithm for Marine Heat Flow Measurements

HEINER VILLINGER[1] AND EARL E. DAVIS

Pacific Geoscience Centre, Geological Survey of Canada, Sidney, British Columbia

Multiple penetration heat flow surveys that employ "violin bow" multithermistor array instruments generate large quantities of high-quality data. To cope with the problems associated with the reduction of these data, an algorithm has been developed which can be used to reduce the data automatically on a microcomputer. The algorithm is designed for use with a pulsed line source method of conductivity measurement, although it could be modified easily for use with the continuous line source method. The multiply iterative algorithm deals, on a sensor by sensor basis, with errors associated with nonideal probe construction and sediment entry, factors that affect the determination of both in situ temperatures and thermal conductivities. Numerical tests of the algorithm show that it is accurate and stable and fast enough for post-real-time reduction of data at sea. Practical tests on high-quality data from the western Pacific show that accuracies are limited by the instrumental resolution; in these cases, the reduction algorithm provided determinations of undisturbed in situ temperatures to better than 1 mK and of in situ conductivities to within 1%.

INTRODUCTION

Most contemporary marine heat flow studies have a common requirement for high accuracy. Obvious examples include those carried out to determine better the characteristic relationship between heat flow and lithospheric age (e.g., *Davis et al.*, 1984; *Louden et al.*, 1987], to estimate seafloor age from heat flow in the absence of other more directly applicable information [e.g., *Langseth et al.*, 1980; *Tamaki*, 1985], and to establish the anomalous heat flow associated with major hot spots [e.g., *Von Herzen et al.*, 1982; *Detrick et al.*, 1986; *Courtney and White*, 1986]. Others include studies in ridge crest and back arc environments, where heat flow variability can provide information about the nature of hydrothermal circulation in the crust beneath the mantle of sediment through which the heat flow is determined [e.g., *Green et al.*, 1981; E. E. Davis and H. Villinger, manuscript in preparation, 1987], and where the determination of any nonconductive component of heat flow may provide information about the rate of pore fluid flux through the sediments [e.g., *Anderson et al.*, 1979; *Langseth and Herman*, 1981]. Similarly, in sediment-accretionary environments, heat flow can be of great value in the description of the way in which sediments are accreted and pore fluids expelled during deformation [e.g., *Yamano et al.*, 1984].

In all cases, the heat flow must be well characterized, both in plan and with depth. This creates several requirements for the heat flow instrumentation used: (1) Multiple penetrations must be made to characterize local heat flow variations adequately or to verify that none are present and thus that the values measured are regionally representative, (2) thermal conductivity must be measured in situ, for it is rare that both spatial and depth variations in conductivity are small enough for values to be estimated reliably from cored samples, and (3) measurements of in situ temperatures and conductivities must be made in sufficient detail to characterize the conductivity-depth structure and ultimately any depth variations of heat flow that may be present.

Developments in heat flow instrumentation in the past decade have responded well to these needs. Multiple penetration stations were made possible with the instrument design of R. Von Herzen (first described by *Von Herzen and Anderson* [1972]) which used outrigger sensors on a solid strength member to enable many penetrations to be made without mechanical damage to the instrument. A low-resolution acoustic link provided enough data in real time to permit the operator to monitor the status of the instrument during extended stations. A significant improvement to this design was provided by C. Lister (described by *Hyndman et al.* [1978]), who developed a successful high-resolution acoustic telemetry link in order to eliminate the constraint on station duration from data storage capacity and, more importantly, incorporated the solid strength-member design with a single sensor tube which measures both in situ temperatures and thermal conductivities with a closely spaced thermistor array. Many instruments now exist which have followed this "violin bow" design [e.g., *Hyndman et al.*, 1979; *Davis et al.*, 1984; *Hutchison*, 1983], and although differences exist in the details of their design (e.g., thermistor excitation, data acquisition and storage, and telemetry), the problem of the reduction of the massive quantity of data produced by these instruments can be treated in a common way. To deal with this "problem" efficiently yet accurately, we have developed a complete heat flow reduction algorithm that deals with all significant and commonly encountered sources of error and permits in situ temperatures and conductivities to be calculated automatically.

BACKGROUND

The conductive heat flow through the seafloor is normally determined as the product of the vertical temperature gradient and the thermal conductivity measured with gravity-driven probes or corers in deep-sea sediments. Two methods are commonly used to measure thermal conductivity of deep-sea sediments in situ and in the laboratory. One is a continuously powered line source method [*Von Herzen and Maxwell*, 1959] where the temperature rise of a cylindrical probe is used to calculate the thermal conductivity. The other one is the pulsed line source method [*Lister*, 1979] in which the decay of a calibrated heat pulse is used. Details of the theoretical background of both methods can be found in the work by *Blackwell* [1954] and *Jaeger* [1956]. Historical, constructional, and

[1]Now at Alfred-Wegener-Institut für Polarforschung, Bremerhaven, Federal Republic of Germany.

Paper number 7B6038.
0148-0227/87/007B-6038$05.00

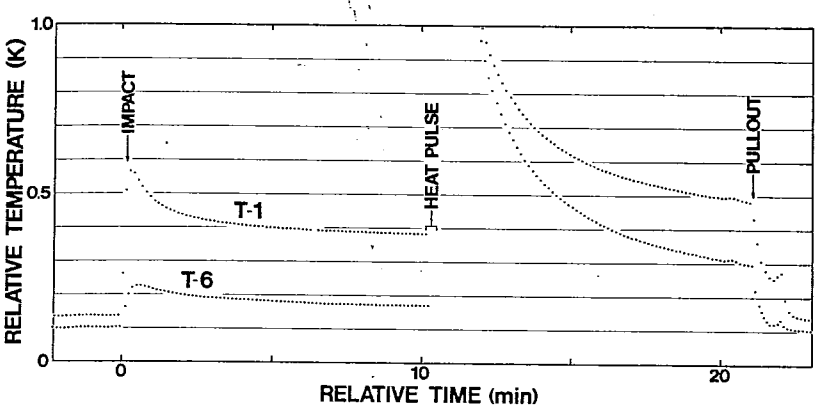

Fig. 1. Record of temperature versus time for a "typical" heat flow measurement. Data for only two sensors are shown for clarity.

operational reasons have led to the exclusive use of the pulsed line source method with violin bow heat flow probes. The lower power required by the pulsed line source method is advantageous for multipenetration heat flow surveys, since with a given battery capacity it allows twice the number of in situ thermal conductivity measurements that is possible with a continuous line source [*Hyndman et al.*, 1979]. With this technique evolving as a standard, we restrict our discussion to problems associated with the reduction of measurements using a violin bow heat flow probe with in situ heat pulse thermal conductivity measurement, although much of the discussion that follows can be applied directly to the reduction of impulse decays obtained with Bullard probes, outrigger sensors, and needle probes and with minor modification to the reduction of continuous line source data as well.

An example of a temperature-time record used for the determination of heat flow is shown in Figure 1. In sequence are (1) the time period prior to final descent and penetration, during which the probe is held some tens of meters above the seafloor and "zero-gradient" reference temperatures are recorded, (2) the frictional heating and subsequent thermal decay following the entry of the probe into the sediments (penetration decay), and (3) the thermal decay following the calibrated heat pulse (heat pulse decay). In brief, the data are reduced in the following way. First, the penetration decay is fitted in a least squares sense with a straight line after its time axis is transformed to the cylindrical decay function $F(\alpha, \tau)$ [*Bullard*, 1954] (see following discussion), with the origin at the time when the impact is first observed. The transformed data are extrapolated to infinite time in order to establish the temperature rise at a given thermistor above the "zero-gradient" reading and to times during the heat pulse decay in order to remove residuals that remain from the penetration disturbance. After these residuals are removed, a conductivity is calculated using the temperature rises above the extrapolated infinite time temperatures measured during the decay following the heat pulse supplied to the sensor string. Use of the cylindrical decay function in both cases requires knowledge of the sediment diffusivity, which is in turn related to the sediment conductivity. Initially, an estimate of the diffusivity is used, and after the conductivity is determined from the heat pulse decay, a second iteration of the reduction is made using a new value of diffusivity determined from the first-computed conductivity and an empirical relationship between diffusivity and conductivity given by *Hyndman et al.* [1979].

THEORY

The theory for the time-dependent temperature distribution within an infinitely long cylinder subject to a known thermal pulse is presented by *Carslaw and Jaeger* [1959]. Their solution can be applied to the determination of the undisturbed sediment temperature from heat flow probe penetration decays [*Bullard*, 1954] as well as to the in situ measurement of thermal conductivity [*Lister*, 1979]. Assumptions are made that the probe is initially isothermal at a temperature different from that of the sediment and that it is a perfect thermal conductor of known heat capacity per length.

The temperature of the cylinder at a time t is given by

$$T(t) = T_0 F(\alpha, \tau) \tag{1}$$

where

$$F(\alpha, \tau) = \frac{4\alpha}{\pi^2} \int_0^\infty \frac{\exp(-\tau u^2)\, du}{u\, \Delta u} \tag{2}$$

and

$$\Delta u = [uJ_0(u) - \alpha J_1(u)]^2 + [uY_0(u) - \alpha Y_1(u)]^2 \tag{3}$$

J_n and Y_n are Bessel functions of the order n of the first and second kinds, $\tau = (\kappa t)/a^2$ defines the time constant of the cylinder of radius a in a sediment with a thermal diffusivity κ, and α is twice the ratio of the heat capacity of the sediment to that of the probe material. T_0 is the initial temperature of the cylinder above ambient temperature.

For large τ ($\tau > 10$), $F(\alpha, \tau)$ can be approximated by

$$F(\alpha, \tau) = 1/2\alpha\tau \tag{4}$$

This relationship is best when $\alpha = 2$. Since T_0 can be expressed as the total heat input Q divided by the thermal capacity S of the probe and the asymptotic expression for (1) can be written as

$$T(t) = Q/4\pi kt \tag{5}$$

for large t. Thus if the heat input is known, either the slope of the measured temperature rise of the cylinder versus $1/t$ or the temperatures at any time during the decay allows us to calculate the thermal conductivity of the sediment:

$$k_{\text{slope}} = \frac{Q}{4\pi} \left[\frac{dT(t)}{d(1/t)}\right]^{-1} \tag{6}$$

or

$$k_{\text{point}} = \frac{Q}{4\pi t T(t)} \tag{7}$$

Relationship (6) or (7) gives a correct value for k only if the large time approximation is valid, i.e., for $\tau > 10$. One way to use (6) or (7) for earlier times is by correcting the temperatures $T(t)$ in such a way that the relationships (6) and (7) are valid for all times [*Lister*, 1979; *Hyndman et al.*, 1979]:

$$T_c(t) = T(t)\, C(\alpha, \tau) \tag{8}$$

with

$$C(\alpha, \tau) = \frac{1}{2\alpha\tau F(\alpha, \tau)} \tag{9}$$

Using (1) with (8) and (9) gives

$$T_c(t) = \frac{T_0}{2\alpha\tau} \tag{10}$$

and replacing T_0 with the equivalent heat input yields

$$T_c(t) = \frac{Q}{4\pi k t}$$

which is valid for all times and can be used to calculate the thermal conductivity of the sediment as discussed before.

The calculation of $C(\alpha, \tau)$ requires the knowledge of κ and α. Simple calculations show that for typical probe constructions α is close to 2 and varies only by about 10% over the thermal conductivity range of deep-sea sediments [*Lister*, 1979; *Hyndman et al.*, 1979]. The thermal diffusivity is obtained from an empirical relationship between thermal conductivity and thermal diffusivity for deep-sea sediments [*Hyndman et al.*, 1979] that was based on the conductivity-water content relationship of *Ratcliff* [1960] and a mean grain specific heat appropriate for red clay and carbonate [*Bullard*, 1954]:

$$\kappa = k/(5.79 - 3.67k + 1.016k^2)10^{-6}\ \text{m}^2\ \text{s}^{-1} \tag{11}$$

This relationship has been confirmed in several environments [e.g., *Hyndman et al.*, 1979; *Davis et al.*, 1984].

With an assumed thermal conductivity k_{ass}, κ can be calculated using (11), and under the assumption of $\alpha = 2$ all the needed parameters for $C(\alpha, t)$ are obtained. Thus $T_c(t)$ can be calculated (equations (9) and (2)), and either of the relationships (6) or (7) can be used to compute the conductivity.

UNCERTAINTIES IN THE ORIGIN TIME AND HENCE A NEW REDUCTION ALGORITHM

Unfortunately, this reduction scheme cannot be used without consideration of another factor that arises from the nonideal nature of heat flow probes. The problem was recognized by *Lister* [1979], who in testing the pulse probe technique noted that "... some phenomenon other than the probe-sediment thermal decay is affecting the results and obvious possibilities are the thermal time constants associated with the internal structure of the sensor string itself." He fitted an exponential function to the nonideal component of the heat pulse decay. In an earlier discussion of probe behavior, *Pratt* [1969] noted that "... the various effects of contact resistance and of temperature sensor position are normally found empirically" and could be accounted for by using a constant time shift of the origin time. This behavior has been observed subsequently by others who have employed the pulse probe measurement technique [*Hyndman et al.*, 1979; *Hutchison*, 1983; *Davis et al.*, 1984].

There are two possible causes of this nonideal behavior: one associated with the nonideal aspects of the probe construction and one associated with the physical disturbance of the sediments due to the penetration of the probe. The internal construction of the sensor string introduces thermal barriers between temperature sensors and the probe's outer surface as well as between internal heater wires and the surface. The filling of the sensor tube with mineral oil helps to reduce these thermal barriers but cannot overcome them completely. These barriers cause a finite thermal response time of the probe and thus a delay both between the time that the heat pulse is delivered by the probe and the time that the heat is received by the surrounding sediment as well as in the times that thermal signals from the sediments appear at the sensors. The magnitude of the composite delay associated with the heat pulse has been obtained by carefully calibrating the sensor string in the laboratory in material with known thermal conductivity. *Hyndman et al.* [1979] and *Mojesky* [1981] found delays that were consistent from test to test with a given sensor but that ranged from about 15 to 30 s depending on the sensor string and on the sensor in a given sensor string used. Our and other's [*Hutchison*, 1983] observations on a number of penetration decays and in situ thermal conductivity measurements with the pulse probe technique suggest that the effective time origin also changes from penetration to penetration. This observation suggests that in the process of penetration the sensor string disturbs the sediment and creates a thin, low thermal conductivity layer at the outside of the sensor tube. This may be caused either by the distortion of the sediments in the direct vicinity of the tube or by a thin layer of water left in the wake of the penetration. If such disturbances are common, then we would expect that (1) a shift of the effective origin time should be present in the penetration as well the heat pulse decay, (2) the distortion of the sediment should decrease with depth, and as a result, so should the shift in the effective origin time, and (3) the shift of the effective origin time should vary with the mechanical characteristics of the sediment. Thus a time delay must be determined individually for each thermistor and each penetration.

To account for the contact resistance and the finite response time of the probe, we have designed ad hoc but physically reasonable procedures to calculate the effective origin times for both the penetration and heat pulse decays. In the case of the penetration decay the appropriate origin time is problematic in two ways. In addition to the unknown delay associated with the various internal and external contact resistances is the uncertain nature of the penetration itself. Penetration in not instantaneous, and the sensor sampling interval is typically too large (e.g., 10 s) for the penetration disturbance to be well characterized. Any inaccuracy in the choice of the penetration time should be reflected in a nonlinearity of the decay temperatures versus $F(\alpha, \tau)$. In the reduction algorithm described here, we use the first indication of the penetration in the temperature record only as the initially assigned origin time. The origin time for each sensor is then shifted by a preset time increment until the standard deviation for the least squares linear fit is minimized. The extrapolated temperature (i.e., for $F(\alpha, \tau) = 0$ at $t = \infty$) is then the undisturbed sediment

temperature at the depth of the sensor. The slope of the penetration decay versus $F(\alpha, \tau)$ is then used to extrapolate the temperature decay to times of the heat pulse decay in order to remove the residual temperature remaining from probe penetration. The success of this technique is apparent in Figure 2*a*, where the scatter in decay values about the best fit line is plotted as a function of effective origin time delay. The minimum in this example occurs at a time substantially later than the time when penetration is first sensed, and the scatter of temperatures about the best fit (0.63 mK) is less than the instrumental resolution (1 mK).

The heat pulse decay appears to behave in much the same way (Figure 2*b*), and a similar criterion is used for determining its effective origin time, although in this case, an additional constraint is applied in the following way. The corrected temperatures of the heat pulse decay T_c versus $1/t$ allow the calculation of a k_{slope} (cf. equation (6)). On the other hand, (7) provides a way of using individual corrected temperatures T_c to determine thermal conductivities k_{point} which should be independent of time. The assumed thermal conductivity k_{ass} is the sought true conductivity of the sediment only if

$$k_{ass} = k_{slope} = k_{point} \tag{12}$$

If (12) is not met within preset error bounds, k_{ass} and the effective origin time are varied, and a new $C(\alpha, \tau)$ function and new T_c values are calculated. Once the condition of (12) is met, the iteration will have converged to a value of k_{ass} where the penetration and heat pulse decays give the same infinite time temperature. In practice, this also occurs where the standard deviation of the linear fit of T_c versus $1/t$ is a minimum.

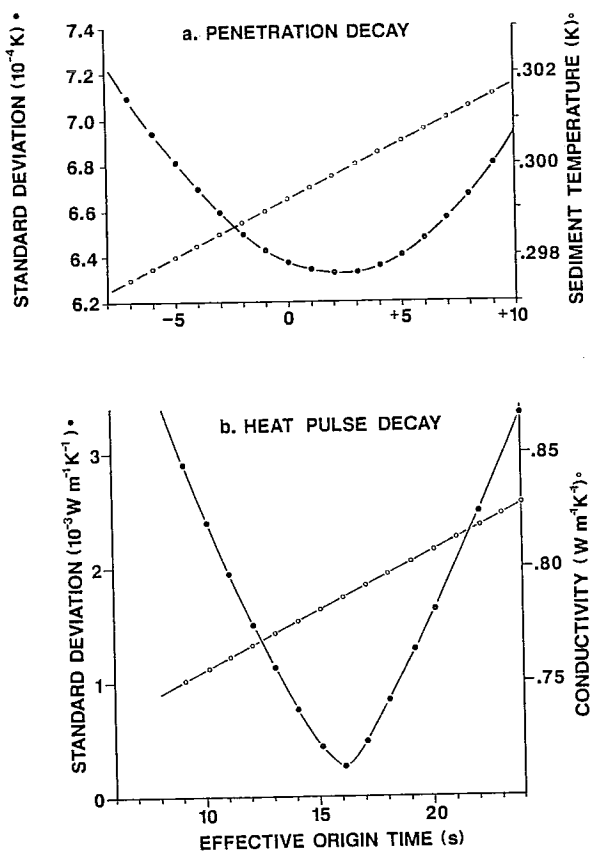

Fig. 2. (*a*) Standard deviation of temperature residuals of the best linear fit to penetration decay temperature versus $F(\alpha, \tau)$ for various values of assumed origin time (solid circles). Open circles show the sediment temperature calculated using the respective origin times. (*b*) Similar plot for the heat pulse decay.

Errors associated with inappropriate origin times can be large (see Figure 3), and as other aspects of the reduction algorithm are relatively straightforward, the remaining discussion focuses primarily on the automatic determination of the effective origin time for the penetration and the heat pulse decays. The complete reduction scheme has been implemented as a Fortran program designed for a microcomputer, and a complete commented version of the program is given by *Villinger and Davis* [1987].

The reduction procedure for the heat pulse decay is illustrated for a single sensor in Figure 3. All the calculations are done in the corrected temperature versus 1/time space using mainly the equations (6), (7), and (10). The procedure starts with an assumed value for k (k_{ass}) and an initial estimate of the effective origin time which is shifted by a nominal delay from the center of the heat pulse (Figure 3*a*, line 0). The effective origin time is then varied until k_{slope}, the inverse of the slope of the linear least square fit of the corrected temperatures versus 1/time, is equal to k_{ass} (line I). At this stage, k_{point} values and a resultant k_{point} (average) are computed. If k_{point} (average) and k_{ass} differ (i.e., if the k_{slope} fit does not pass through the origin at infinite time), the assumed conductivity k_{ass} is changed and the search for equality of k_{ass} and k_{slope} starts again. The iteration is allowed to terminate if

$$k_{ass} = k_{slope} = k_{point}(\text{average}) \tag{12'}$$

which is the case for line III in Figure 3*a*.

To go from I to II or from II to III in Figure 3*a*, the effective origin time has to be iteratively changed. Figure 3*b* illustrates this procedure. We start out with point I-1 and change the effective origin time by a small amount. A new k_{slope} is calculated (point I-2) that is farther from or closer to k_{ass} than the original value. With the trend known, a third origin time is determined and used to calculate a third k_{slope} (point I-3). The size of the iteration steps is calculated with the method of false position (Regula Falsi). The iteration continues until point I-4 is reached for which the Δk is within a preset error bound. We then compute k_{point}(average), change k_{ass} to be equal to k_{point}(average), and come, with the same effective origin time, to point II-1. The same procedure is then repeated. At the point III-3 the assumed thermal conductivity, the slope conductivity, and the mean of the point thermal conductivities are equal, and therefore the true value for the sediment thermal conductivity is found. Although linearity of the decay is not used explicitly as a constraint in determining the best origin time as is the case with the penetration decays, the final solution (point III-3) usually coincides with a minimal standard deviation of the residuals when fitting T_c versus $1/t$ as expected (Figures 3*c* and 2*b*).

The two iteration loops explained above are required to find the true thermal conductivity of the sediment. In the "inner loop" the assumed thermal conductivity is kept fixed and the time shift is iteratively changed until

$$k_{ass} = k_{slope}$$

within a preset error bound k_{toler}, i.e., until

$$\Delta k = |k_{ass} - k_{slope}| \leq k_{toler}$$

In the "outer loop," k_{ass} is varied until the intercept temper-

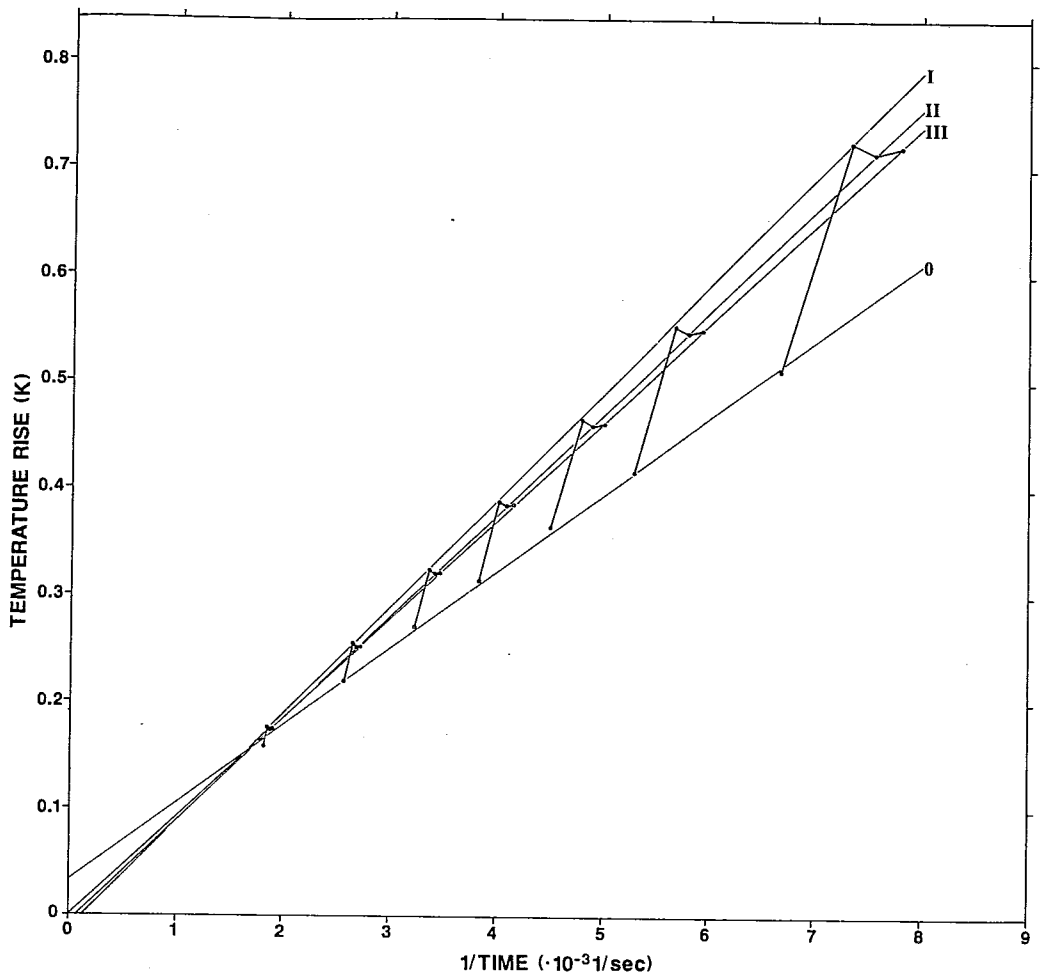

Fig. 3*a*. Example of temperature versus 1/time of a heat pulse decay at various stages of the reduction process. Points along the best fitting line 0 are measured temperature-time points. Points along I and II are corrected temperatures T_c plotted against $1/t$ after a time shift has been applied to make $k_{ass} = k_{slope}$. The line through the final points of the final iteration, III, passes through the origin and satisfies the equality $k_{ass} = k_{slope} = k_{point}$(average).

ature for $t = \infty$ is zero (cf. Figure 3*a*), which occurs when

$$k_{point}(\text{average}) = k_{slope}$$

to the same level of agreement k_{toler}.

In the complete heat flow reduction algorithm, this outer loop includes the penetration decay calculation, as the penetration decay fit and thus the decay residuals and the equilibrium temperatures are affected by the assumed conductivities. Experience shows that after two cycles the results remain unchanged and vary only at the noise level of the input data.

Tests for Sensitivity to Reduction Parameters

A number of numerical tests have been made to determine the sensitivity of the algorithm to the reduction parameters. These are (1) the error bound k_{toler} in the iteration loops, (2) the chosen value value of α, and (3) the initially assumed thermal conductivity k_{ass}.

The error bound k_{toler} controls both the "inner" and "outer" iteration loops and thus the total number of iterations since it is the criterion for their termination. It is therefore necessary to investigate the influence of this value on the final k. Using a number of penetrations, k was calculated for various values of k_{toler} which increased from 0.0001 to 0.005 W m^{-1} K^{-1}. The resulting conductivities were compared to the results obtained with $k_{toler} = 0.0001$ W m^{-1} K^{-1}; differences are plotted as frequency histograms in Figure 4. A value of k_{toler} of about 0.001 W m^{-1} K^{-1} appears to be an optimal value, in that it guarantees correct results with a minimum of iterations.

The ratio of the thermal capacities of the sediment and the probe varies with changing sediment properties. Under the assumption of (11) this results in

$$\alpha = \frac{2\pi a^2(5.79 - 3.67k + 1.016k^2)}{S} \tag{13}$$

with $S = 2\pi a^2 \rho C$ (probe) being the thermal capacity per unit length of the probe itself. Ideally, this variation should be accounted for. In practice, however, the effect is small. Values of ρC have been estimated by *Lister* [1979] and *Hyndman et al.* [1979] for typical probe constructions. Using $\rho C = 3.3 \times 10^6$ J m^{-3} K^{-1}, (13) can be calculated for arbitrary α or k values. The resulting function is shown in Figure 5*a* (circled points). We have reduced several penetrations using varying values of α ranging from 1.8 to 2.2. Results are also shown in Figure 5*a*. Over this range of conductivity, the error introduced by assuming $\alpha = 2$ is $\leq 1\%$. With such insensitivity of k to α, it is adequate to use a fixed value of 2.0 for most sediments. If extreme conductivities are encountered, a correc-

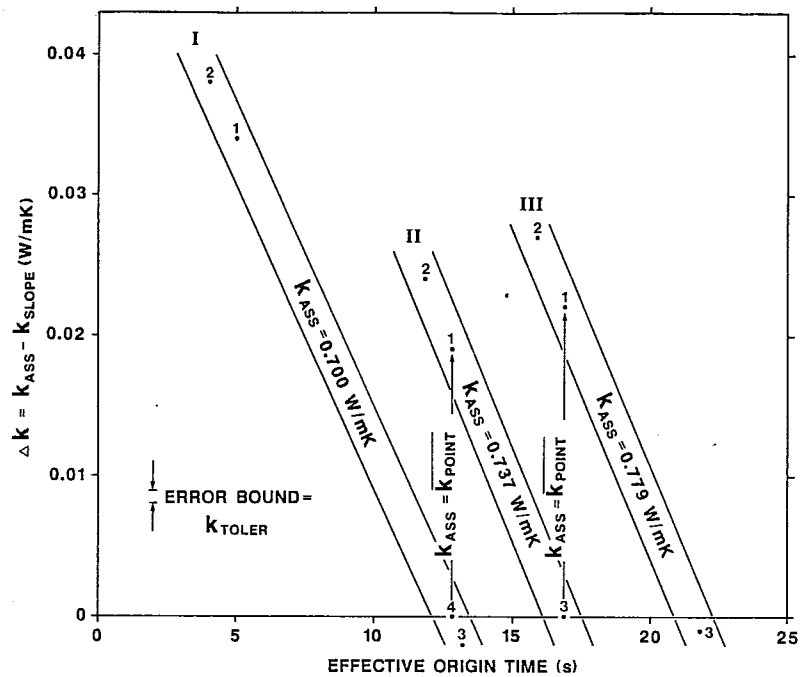

Fig. 3*b*. The difference between k_{ass} and k_{slope} as a function of effective origin time and assumed thermal conductivity k_{ass}. The path of the iteration process can be traced from point to point, with the Roman and Arabic numerals corresponding to the outer and inner loops of the reduction algorithm.

tion factor can be used to compensate for the use of an incorrect α value (Figure 5*b*).

Sensitivity to the choice of the initially used k_{ass} is demonstrated in Figure 6. Clearly the choice is not crucial for the reduction algorithm as long as it is within reasonable limits of deep-sea sediment thermal conductivities. The same test of the reduction algorithm was performed by using model input data calculated with a known thermal conductivity. Resulting calculated k values were always within 0.2% of the assumed model thermal conductivity.

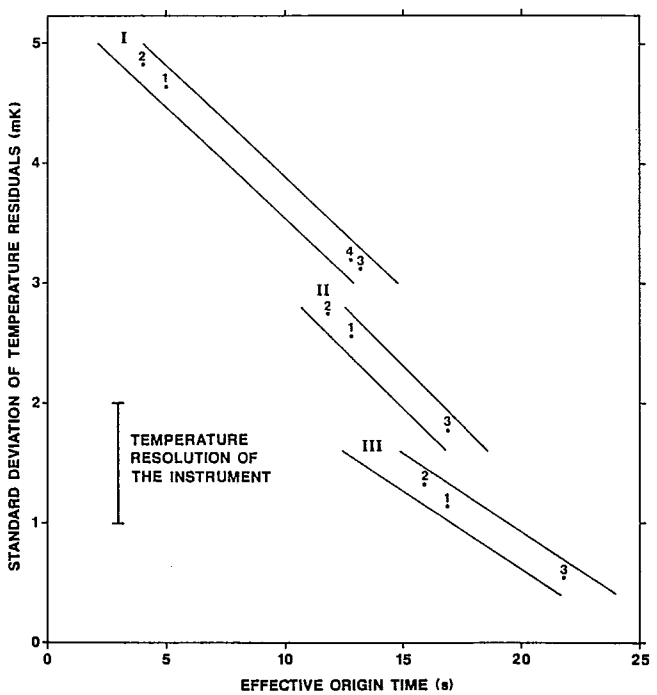

Fig. 3*c*. The standard deviation of the residuals of the linear fit of T_c versus 1/time (cf. Figure 3*a*) for various stages of the reduction algorithm. Correspondence of the best fit with the final solution at $k_{ass} = k_{slope} = k_{point}$ (average) (point III-3) provides an excellent internal consistency check of the algorithm.

The use of a finite length heat pulse has been vindicated on theoretical grounds by *Lister* [1979] and in practice through calibration of probes having a variety of diameters and pulse lengths [*Hyndman et al.*, 1979; *Mojesky*, 1981; H. Villinger, personal communication, 1987]. In an extreme case the decay of a 1-mm needle probe having a pulse length of 10 s conformed to theory in spite of the ratio of the probes thermal time constant to the pulse length being only about 2:1.

PRACTICAL TESTS OF THE ALGORITHM

We have used the described algorithm for the reduction of about 100 heat flow measurements made during a cruise in the western Pacific (TGT-178; for details see C. R. B. Lister et al. (manuscript in preparation, 1987)). The probe used was designed by C. R. B. Lister from the University of Washington (Seattle) and is described in detail by *Davis et al.* [1984]. This first application of the reduction algorithm was an excellent performance test for the program as the data quality is particularly and consistently high. The results allow us to examine our assumptions and perceptions about the causes of nonideal behavior of the frictional and heat pulse decays.

Figure 7 shows a typical example of the reduction of a penetration decay. A number of things are noteworthy: The best fit of temperatures versus $F(\alpha, t)$ occurs for an assumed origin time that is significantly later than the moment of penetration defined by the first observed temperature rise (see Figure 2*a*). In this case the origin time that provides the best fit is fairly well defined, but this definition is dependent on the slope of the temperature versus $F(\alpha, \tau)$ decay. Lack of convergence may occur for low slopes; these are dealt with in the

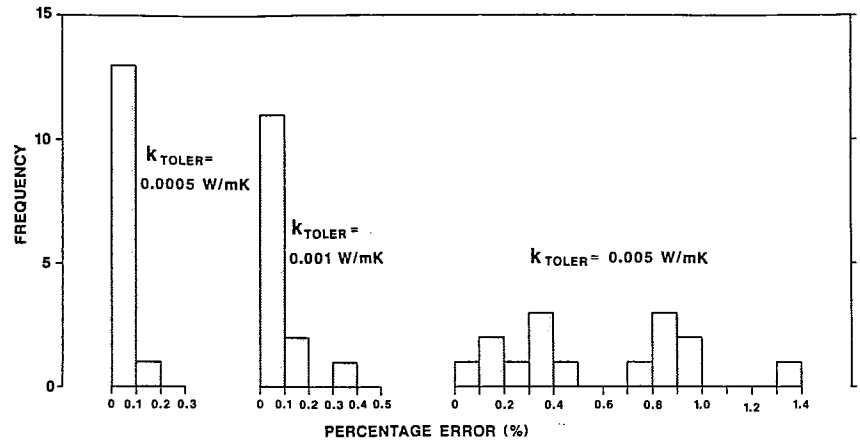

Fig. 4. Frequency distribution of percentage errors due to a particular choice of k_{toler}.

reduction scheme by limiting the possible range of the time shift. The sediment temperature at the estimated infinite time intercept is relatively insensitive to the choice of the origin time, particularly at low slope, as are the residuals removed from the heat pulse decay. Including this step in the reduction process does, however, improve the ultimate accuracy of both the gradient and the conductivity determinations. It also provides a flag for the recognition of disturbed penetration decays, and it is interesting to note that including the adjustment of penetration time increases the convergence speed of the heat pulse decay calculations.

The effective origin times for the penetration decays of all sensors are generally between +10 and −10 s (Figure 8*a*). The scatter in effective origin times is quite large and probably due to the very broad minima determined for many of the decays which have low slopes. Variation of origin time with depth is consistent with the nature of penetration: bottom sensors first. There is not obvious explanation for the abrupt change in effective origin time between sensor 3 and 4.

Figure 8*b* shows the effective origin times for the heat pulse decays as a function of sensor (and depth). The reduction algorithm gives very consistent effective origin times for each sensor. They do not vary from penetration to penetration by more than 15% for each of the sensors 1 to 6. The magnitude of the effective origin time calculated with the program agrees very well with other estimates for probes of similar construction [*Hyndman et al.*, 1979; *Mojesky*, 1981; *Hutchison*, 1983; *Davis et al.*, 1984]. There is a substantial variation with depth, however; the average origin time delays increase from about 15 s (sensor 1) to about 25 s (sensor 6). This systematic variation is believed to be due to the increase in the amount of sediment disturbance up the probe from sensor 1 to sensor 7, along with the changing probe properties along the length of the probe due to the greater number of wires and perhaps

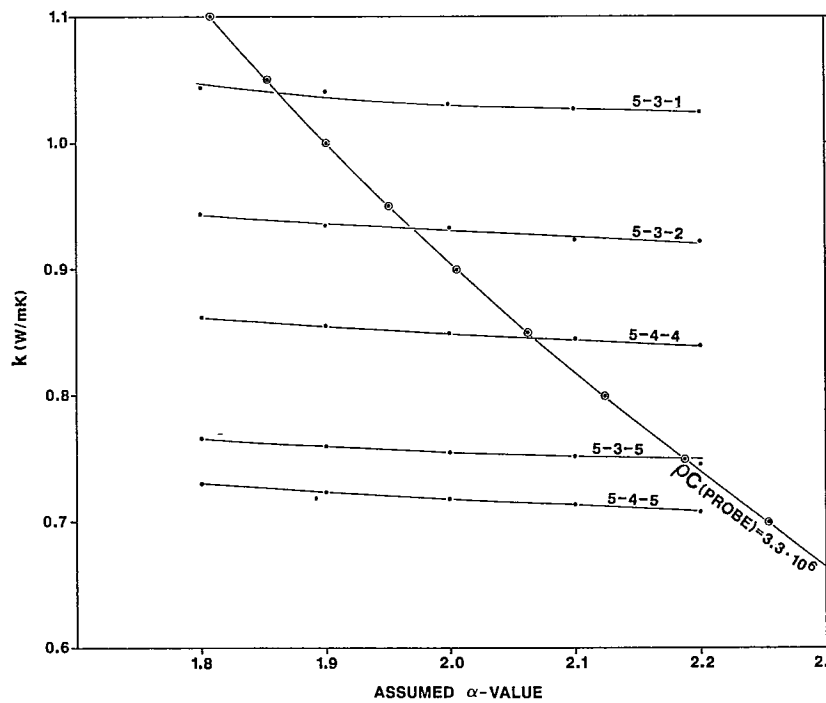

Fig. 5*a*. The influence of the α value used in the reduction algorithm on the final thermal conductivity shown for various penetrations and sensors of station 5, TGT-178. The dashed curve was calculated using equation (13) and under the assumption of a thermal capacity of the probe of 3.3×10^6 J m^{-3} K^{-1}.

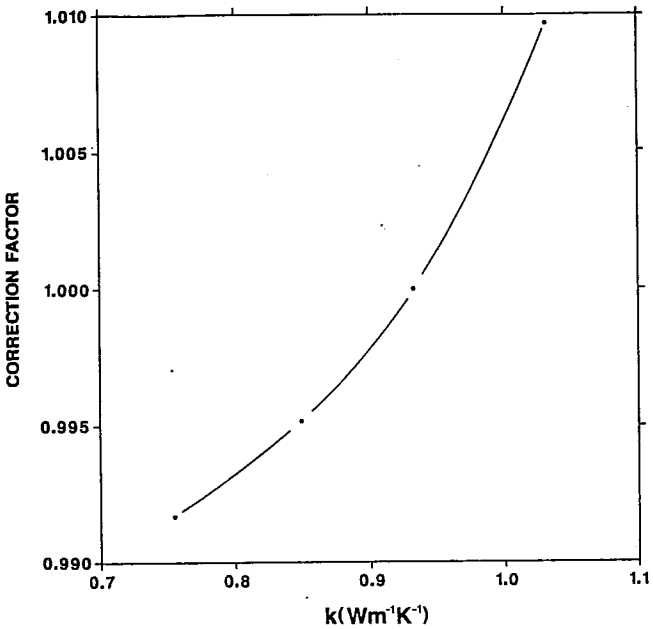

Fig. 5*b*. Conductivity correction factor to account for the error in assuming $\alpha = 2$, calculated using results from Figure 5*a* and plotted as a function of thermal conductivity.

incomplete oil fill in the upper portion of the sensor string. Results from different sites demonstrate that the effective origin time is not resolvably dependent on the thermal conductivity (see Figure 9). This does not imply lack of dependence of time delay on mechanical properties of the sediment, however, since in this case the variability of conductivity is as much due to variations in carbonate content as it is to variations in porosity (C. R. B. Lister et al., manuscript in preparation, 1987). The data for sensor 7 are in many cases more disturbed due to occasional superpenetration of the weight stand. Results for the uppermost sensor should therefore be regarded with caution.

A very good consistency check for the algorithm is provided by the equation

$$T(t) = T_0 F(\alpha, \tau) \tag{14}$$

where in the case of an ideal probe T_0 is the initial temperature at $t = 0$. In theory this value should be constant for a specific probe

$$T_0 = Q/S = \text{const} \tag{15}$$

as Q and S depend only on probe parameters. For the probe used, S is estimated as 3.3×10^6 J m^{-3} K^{-1}, and Q is measured to be 906 J m^{-1}. These values would yield a T_0 of 3.86 K. T_0 can also be calculated from the data since

$$T_0 = \frac{\alpha Q}{[(2\pi\, a^2) * (5.79 - 3.67k + 1.016k^2]} \times 10^{-6} \tag{16}$$

If alpha varies, T_0 should remain constant as it does not depend on sediment properties. However, in our reduction scheme alpha is kept fixed for varying values of k in (14), and as a result, T_0 effectively varies with changing k according to (16). This function and all the determined T_0 values are shown in Figure 10. The agreement is extremely good. Scatter might be due to scatter in Q and/or general "reduction noise."

A final test of the reduction algorithm consisted of a comparison of heat flow reductions of data collected with a variety of other instruments. The violin bow instruments used at Pacific Geoscience Centre (Sidney, British Columbia), at Memorial University (St. John's, Newfoundland), and at Dalhousie University (Halifax, Nova Scotia) all have different internal probe constructions, temperature resolutions, and heat pulse strength. Reduction of data sets from all of these probes confirmed the agorithm's performance and suggested that the re-

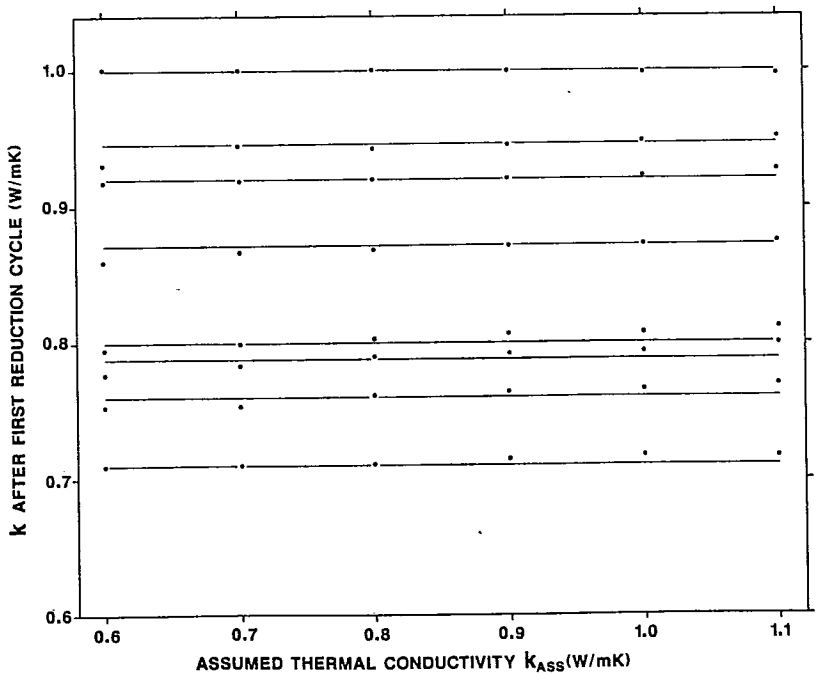

Fig. 6. The effect of the initially assumed thermal conductivity k_{ass} on the results after the first reduction cycle, determined for measurements in sediments of the western Pacific (cruise TGT-178) having a wide range of thermal conductivities. Final values of conductivity calculated after a second iteration are shown as horizontal lines.

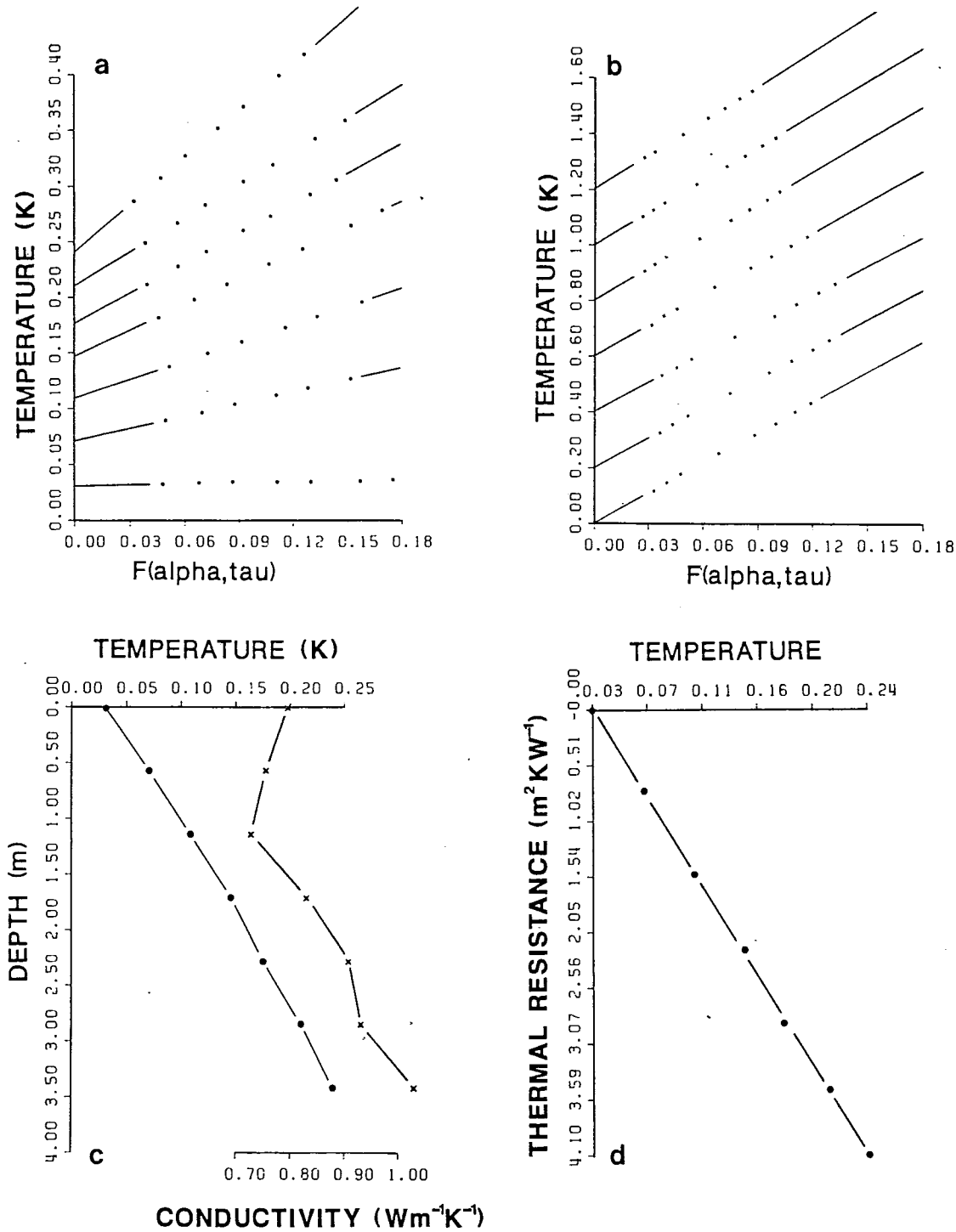

Fig. 7. Results of a typical heat flow penetration, showing the plots of the (*a*) penetration and (*b*) heat pulse decays after two iterations of the reduction algorithm, and the resultant (*c*) temperature and conductivity versus depth and (*d*) temperature versus thermal resistance.

duction of the TGT-178 data set described in this paper is typical.

SUMMARY

A new scheme has been presented for the reduction of marine heat flow data. The scheme has been developed specifically for instruments employing the pulsed line source method of in situ conductivity measurement, although most of the problems dealt with in the reduction algorithm are common to those encountered using the continuous line source method as well.

The theory used to describe the temperature rise or decay following probe penetration and the temperature decay following a calibrated heat pulse is that for a perfectly conductive cylinder imbedded in a medium of uniform thermal conductivity. However, the theory is not entirely suitable for several reasons: (1) Typical probe constructions are not ideal. Low conductivity paths are present between the line source of

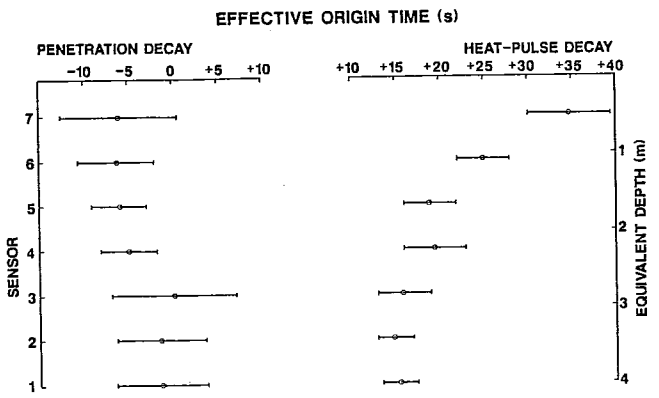

Fig. 8. The effective origin time of the penetration and heat pulse decays plotted against sensor depth for all penetrations of cruise TGT-178. The standard deviations for each sensor are indicated.

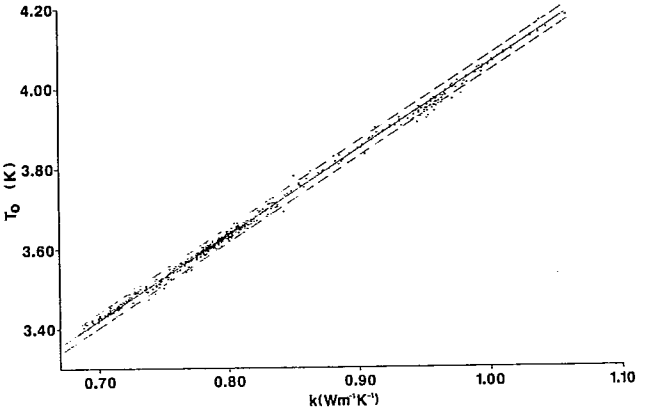

Fig. 10. The slope T_0 as a function of thermal conductivity for all penetrations and sensors of TGT-178. The solid line was calculated with (13). The dashed lines indicate a 10% error margin.

heat and the steel wall of the probe, and between the probe wall and the thermistor sensors. (2) Penetration of the probe into the seafloor disturbs the sediments, producing a thermal barrier between the probe and the undisturbed sediments. (3) Probe penetration is not instantaneous. In the reduction scheme, these nonideal conditions are assumed to produce a composite time delay between the times that the heat is generated and the times the thermal signal in the sediments is sensed. Selection of the appropriate time delay is made on a sensor by sensor basis for each penetration. The effective origin time of penetration is determined by requiring that the penetration decays follow a linear $F(\alpha, \tau)$ relationship as predicted by ideal probe theory. A similar criterion is applied to the heat pulse decay, although in this case, two independent constraints are available: (1) the decay of temperature must be linear in an $F(\alpha, \tau)$ sense, and (2) the infinite time ($F(\alpha, \tau) = 0$) intercept must be equal to the equilibrium temperature determined by the extrapolation of the penetration decay.

The use of an effective origin time delay in the analysis of line source data is physically reasonable, but nevertheless ad hoc. However, numerical simulations of a nonideal probe [*Hyndman et al.*, 1979] produced comparable time shifts to those found in our data reductions. Furthermore, tests of the reduction algorithm on a particularly high-quality data set produced the following results: (1) The standard deviation of residuals between temperatures and best fitting (delayed) $F(\alpha, \tau)$ lines are always at or below the level of instrumental resolution (1 mK), for both penetration and heat pulse decays. These residual minima are well defined and single-valued (Figure 2). (2) In the case of the heat pluse decays, the residual minima occur at an effective origin time that also satisfies the criterion that the infinite-time intercept is equal to that determined from the frictional decay (Figure 3). (3) The best fitting time delays yield initial temperature rises (predicted by the slope of the $F(\alpha, \tau)$ heat pulse decay curve) that are independent of conductivity, as required by theory. Finally, the use of a time delay has been checked by full instrumental and data reduction calibration in materials of known or otherwise determined conductivity [*Hyndman et al.*, 1979; *Mojesky*, 1981; *Davis*, 1987].

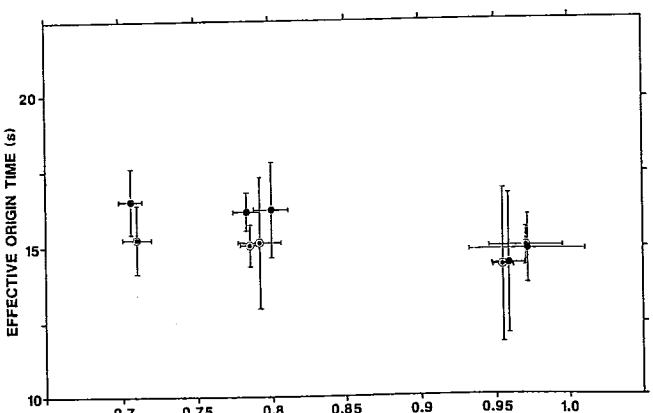

Fig. 9. The effective origin time delay of the heat pulse decay as a function of conductivity for approximately 100 measurements at four sites in the western Pacific (cruise TGT-178). Only values determined using the lowermost two sensors (solid and circled points) are shown to avoid the sensitivity of the origin time to sensor depth (see Figure 8). Conductivities at each site are extremely uniform.

Accounting for nonideal probe behavior is necessary to obtain accurate in situ temperatures and conductivities. For the examples shown in Figures 3*a* and 3*b*, 10-s errors in the penetration or heat pulse origin times result in errors of 3 mK and 0.05 W m^{-1} K^{-1} in temperature and conductivity, respectively. Such errors would lead to substantial (10–20%) errors in interval heat flow values computed for typical thermistor arrays used in marine studies. With the use of the algorithm described here, and full instrument calibration, one can now determine the undisturbed sediment temperature to better than 1 mK and the in situ thermal conductivity to within 1%. This ability to measure marine heat flow to a higher resolution than was previously possible has important consequences for future studies of heat and fluid transfer through the lithosphere, crust, and sediments of ocean ridges, basins, and margins.

Acknowledgments. The authors thank J. Wright (Memorial University) and K. Louden (Dalhousie University) for the use of their data for testing the reduction algorithm. The primary data set used for the tests were collected on cruise 178 of the T. G. Thompson (University of Washington) for which C. Lister was chief scientist. H. Villinger was supported by an NSERC research fellowship. Helpful comments were provided by D. Chapman. Geological Survey of Canada contribution 21487.

REFERENCES

Anderson, R. N., M. A. Hobart, and M. G. Langseth, Geothermal convection through oceanic crust and sediments in the Indian Ocean, *Science*, *204*, 828–832, 1979.

Blackwell, J. H., A transient flow method for determination of ther-

mal constants of insulating materials in bulk, *J. Appl. Phys., 25*, 137–144, 1954.

Bullard, E. C., Heat flow in South Africa, *Proc. R. Soc. London, Ser. A, 173*, 474–502, 1939.

Bullard, E. C., The flow of heat through the floor of the Atlantic Ocean, *Proc. R. Soc. London, Ser. A, 222*, 408–429, 1954.

Carslaw, H. S., and J. C. Jaeger, *Conduction of Heat in Solids*, 2nd ed., Oxford University Press, London, 1959.

Courtney, R. C., and R. S. White, Anomalous heat flow and geoid across the Cape Verde Rise: Evidence for dynamic support from a thermal plume in the mantle, *Geophys. J. R. Astron. Soc., 87*, 815–867, 1986.

Davis, E. E., Determining heat flow through marine sediments, in *Handbook for Terrestrial Heat-Flow Density Determination*, edited by R. Heaenel, L. Ryback, and L. Stegena, D. Reidel, Hingham, Mass., in press, 1987.

Davis, E. E., C. R. B. Lister, and J. G. Sclater, Towards determining the thermal state of old ocean lithosphere: Heat-flow measurements from the Blake-Bahama outer ridge, northwestern Atlantic, *Geophys. J. R. Astron. Soc., 78*, 507–545, 1984.

Detrick, R. S., R. P. Von Herzen, B. Parsons, D. Sandwell, and M. Doherty, Heat flow observations on the Bermuda Rise and thermal models of mid-plate swells, *J. Geophys. Res., 91*, 3701–3723, 1986.

Green, K. E., R. P. Von Herzen, and D. L. Williams, The Galapagos spreading center at 86°W: A detailed geothermal field study, *J. Geophys. Res., 86*, 979–986, 1981.

Hutchison, I., Heat flow studies in the Gulf of Oman and western Mediterranean, Ph.D. thesis, Univ. of Cambridge, Cambridge, 1983.

Hyndman, R. D., G. C. Rogers, M. N. Bone, C. R. B. Lister, U. S. Wade, D. L. Barrett, E. E. Davis, T. Lewis, S. Lynch, and D. Seemann, Geophysical measurements in the region of the Explorer ridge off western Canada, *Can. J. Earth Sci., 15*, 1508–1525, 1978.

Hyndman, R. D., E. E. Davis, and J. A. Wright, The measurement of marine geothermal heat flow by a multipenetration probe with digital acoustic telemetry and in situ thermal conductivity, *Mar. Geophys. Res., 4*, 181–205, 1979.

Jaeger, J. C., Conduction of heat in an infinite region bounded internally by a circular cylinder of a perfect conductor, *Aust. J. Phys., 9*, 167–179, 1956.

Langseth, M. G., and B. M. Herman, Heat transfer in the oceanic crust of the Brazil Basin, *J. Geophys. Res., 86*, 10,805–10,819, 1981.

Langseth, M. G., M. A. Hobart, and K. Horai, Heat flow in the Bering Sea, *J. Geophys. Res., 85*, 3740–3750, 1980.

Lister, C. R. B., The pulse-probe method of conductivity measurement, *Geophys. J. R. Astron. Soc., 57*, 451–461, 1979.

Louden, K. E., D. O. Wallace, and R. C. Courtney, Heat flow and depth versus age for the Mesozoic N.W. Atlantic Ocean: Results from the Sohm abyssal plain and implications for the Bermuda Rise, *Earth Planet. Sci. Lett.*, in press, 1987.

Mojesky, T., Rapid in situ thermal conductivity measurements for the study of geothermal heat flow, M.S. thesis, Univ. of Wash., Seattle, 1981.

Pratt, A. W., Heat transmission in low conductivity materials, in *Thermal Conductivity*, edited by R. P. Tye, pp. 301–402, Academic, Orlando, Fla., 1969.

Ratcliff, E. H., Thermal conductivities of ocean sediments, *J. Geophys. Res., 65*, 1535–1541, 1960.

Tamaki, K., Age of the Japan Sea, *J. Geogr., 94*, 14–29, 1985.

Villinger, H., and E. E. Davis, HFRED: A program for the reduction of marine heat-flow data on a microcomputer, *Geol. Sur. Can Open File Rep., 1627*, 1987.

Von Herzen, R. P., and R. N. Anderson, Implications of heat flow and bottom water temperature in the eastern equatorial Pacific, *Geophys. J. R. Astron. Soc., 26*, 427–458, 1972.

Von Herzen, R. P., and A. E. Maxwell, The measurement of thermal conductivity of deep-sea sediments by a needle probe method, *J. Geophys. Res., 64*, 1557–1563, 1959.

Von Herzen, R. P., R. S. Detrick, S. T. Crough, D. Epp, and U. Fehn, Thermal origin of the Hawaiian swell: Heat flow evidence and thermal models, *J. Geophys. Res., 87*, 6711–6723, 1982.

Yamano, M.,S. Honda, and S. Uyeda, Nankai Trough: A hot trench?, *Mar. Geophys. Res., 6*, 187–203, 1984.

E. E. Davis, Pacific Geoscience Centre, Energy, Mines and Resources, 9860 West Saanich Road, P.O. Box 6000, Sidney, B.C., Canada V8L 4B2.

H. Villinger, Alfred-Wegener-Institut für Polarforschung, D-2850 Bremerhaven, Federal Republic of Germany.

(Received March 17, 1987;
revised July 15, 1987;
accepted August 10, 1987.)