INTERNAL REPORT SDL-88-01

DATA REDUCTION OF THE HALF-INCH SHORTED QUARTZ GAUGE RECORD USING THE CODE INTERP

Z.P. TANG

November 11, 1987

Shock Dynamics Laboratory

Department of Physics

Washington State University

Pullman, WA 99164

(SDL Int. Rpt 88-01)

DATA REDUCTION OF THE HALF-INCH SHORTED QUARTZ GAUGE RECORD USING THE CODE INTERP

Z.P. TANG

Shock Dynamics Laboratory

Department of Physics

Washington State University

November 11, 1987

I. Converting equation for the current records of the half-inch shorted quartz gauge

According to the recent work[1] and the previus work[2-5] on the calibration of the half-inch shorted quartz gauge used in our laboratory, its output response of the current $i(t)$ to a single-step impact stress $\sigma(t)$ can be described as follows:

$$i(t) = \frac{U_s A k(\sigma)}{l}\sigma[1+\alpha(\sigma)(t-t_o)] \tag{1}$$

where U_s is the shock velocity (in $mm/\mu s$) of the x-cut quartz, normally U_s=5.72 $mm/\mu s$ is used, A is the effective area (in mm^2) of the gauge, which is equal to the area of the inner electrode plus half the area of the guard ring as suggested by Hayes and Gupta,[2] l is the thickness of the gauge (in mm), t_o (in μs) is the time when the stress applied. k and α in Eq. (1) are the current coefficient and the ramping coefficient, recpectively. According to the calibration work,[1-5] we have:

$$k = 1.919 + 0.008249\ \sigma \quad (10^{-8} coul/cm^{2}/Kbar) \tag{2}$$

$$\alpha = \frac{i(t)-i_o(t_o)}{i_o(t_o)(t-t_o)} = 0.195 + 0.008236\sigma \ \ (1/\mu s) \tag{3}$$

where σ is in kbar, and $i_o(t_o)$ is the current jump corresponding to the stress jump at time t_o.

How to convert the measured current output profile $i_{meas}(t)$ into the input stress profile for an arbitrary stress history? This problem is important for the shorted quartz gauge. Strictly speaking, it should use nonlinear superposition theory for multiple-step history (an arbitrary wave profile can be considered as a superposition of multiple steps). But it would need more calibration information. However the problem could be much simplified by making such a basic assumption:

The actual current $i_{actl}(t)$ and the current ramping coefficient α at time t only depend on the stress and actual current at time t, and have no history effection, i.e.

$$i_{actl}(t) = \frac{U_s A}{l}\sigma(t)[1.919+0.008249\sigma(t)] \tag{4}$$

for $i_{actl}(t)$ and Eq. (3) for α.

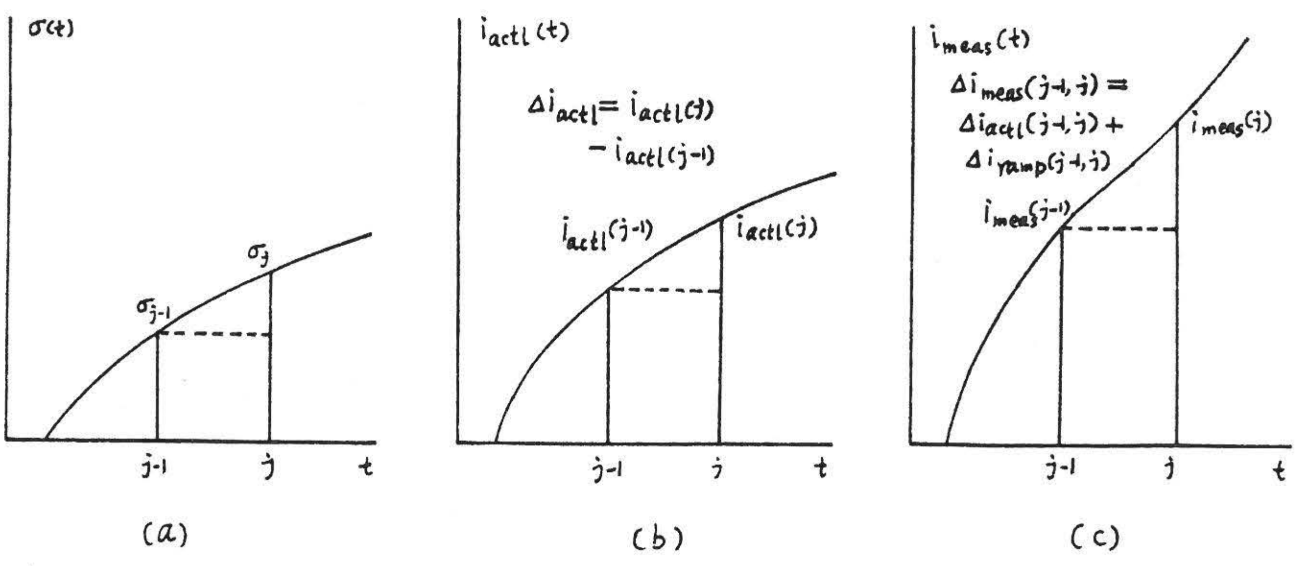


Fig. 1 Data reduction of a quartz gauge.

Based on this assumption, we can convert $i_{meas}(t)$ into $\sigma(t)$. The procedure is illustrated in Fig. 1. From Fig. 1, we have:

$$\Delta i_{meas}(j,j-1) = \Delta i_{actl}(j,j-1) + \Delta i_{ramp}(j,j-1)$$

i.e.

$$i_{meas}(j)-i_{meas}(j-1) = i_{actl}(j)-i_{actl}(j-1)+\overline{\alpha}_{j-1,j}\overline{i}_{actl}(j-1,j)\Delta t \quad (5)$$

where $\overline{\alpha}_{j-1,j}$ and $\overline{i}_{actl}(j-1,j)$ are the average rampimg coefficient and actual current at time interval t_{j-1} and t_j, and we have:

$$\begin{aligned} \overline{i}_{actl}(j-1,j) &= [i_{actl}(j)-i_{actl}(j-1)]/2 \\ \overline{\alpha}_{j-1,j} &= 0.195+0.008236(\sigma_{j-1}+\sigma_j)/2 \\ \Delta t &= t_j-t_{j-1} \end{aligned} \quad (6)$$

Substituting Eqs. (4) and (6) into Eq. (5), it turns out the following equaton:

$$a_1\sigma_j^3 + a_2\sigma_j^2 + a_3\sigma_j + a_4 = 0 \quad (7)$$

where

$$\begin{aligned} a_1 &= 0.25\alpha_2\beta_2c_3\Delta t \\ a_2 &= c_3\left[\beta_2+0.25(2\alpha_1\beta_2+\alpha_2\beta_2\sigma_{j-1}+\alpha_2\beta_1)\Delta t\right] \\ a_3 &= \beta_1c_3+0.25\left[2\alpha_1\beta_1c_3+\alpha_2\beta_1c_3\sigma_{j-1}+\alpha_2i_{actl}(j-1)\Delta t\right] \\ a_4 &= i_{meas}(j-1)-i_{meas}(j)+(0.5\alpha_1\Delta t+0.25\alpha_2\sigma_{j-1}\Delta t-1)i_{actl}(j-1) \end{aligned}$$

where α_1=0.195, α_2=0.008236, β_1=1.92, β_2=0.008249 and $c_3=AU_s/l$. If σ_{j-1} and $i_{actl}(j-1)$ are known, σ_j can be calculated by means of Eq. (7) and then $i_{actl}(j)$ by Eq. (4). Using Eqs. (7) and (4) step by step, a stress history corresponding to the measured current can be calculated.

For the solution of Eq. (7), we have two methods. One is to use the exact solution of the real root of a cubic equation as follows:

$$\sigma_j = Y - a_2/3a_1 \tag{8}$$

where

$$Y = \left\{ -\frac{q}{2} + \left[\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3 \right]^{\frac{1}{2}} \right\}^{\frac{1}{3}} + \left\{ -\frac{q}{2} - \left[\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3 \right]^{\frac{1}{2}} \right\}^{\frac{1}{3}}$$

$$p = \frac{a_3}{a_1} - \frac{a_2^2}{3a_1^2}$$

$$q = \frac{2a_2^3}{27a_1^3} - \frac{a_2 a_3}{3a_1^2} + \frac{a_4}{a_1}$$

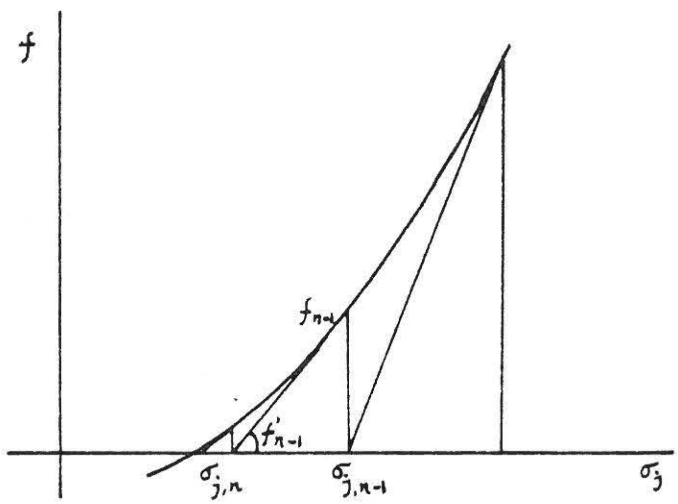


Fig. 2 Newton's iteration method.

The another method is the Newton's iteration method for nonlinear algebra equations, which is shown in Fig. 2. From Fig. 2 we have following equations for the n-th iteration:

$$f_{n-1} = a_1\sigma_{j,n-1}^3 + a_2\sigma_{j,n-1}^2 + a_3\sigma_{j,n-1} + a_4 \tag{9a}$$

$$f'_{n-1} = 3a_1\sigma_{j,n-1}^2 + 2a_2\sigma_{j,n-1} + a_3 \tag{9b}$$

and

$$\sigma_{j,n} = \sigma_{j,n-1} - f_{n-1}/f'_{n-1} \tag{10}$$

When

$$|\sigma_{j,n}-\sigma_{j,n-1}|\leq\epsilon \tag{11}$$

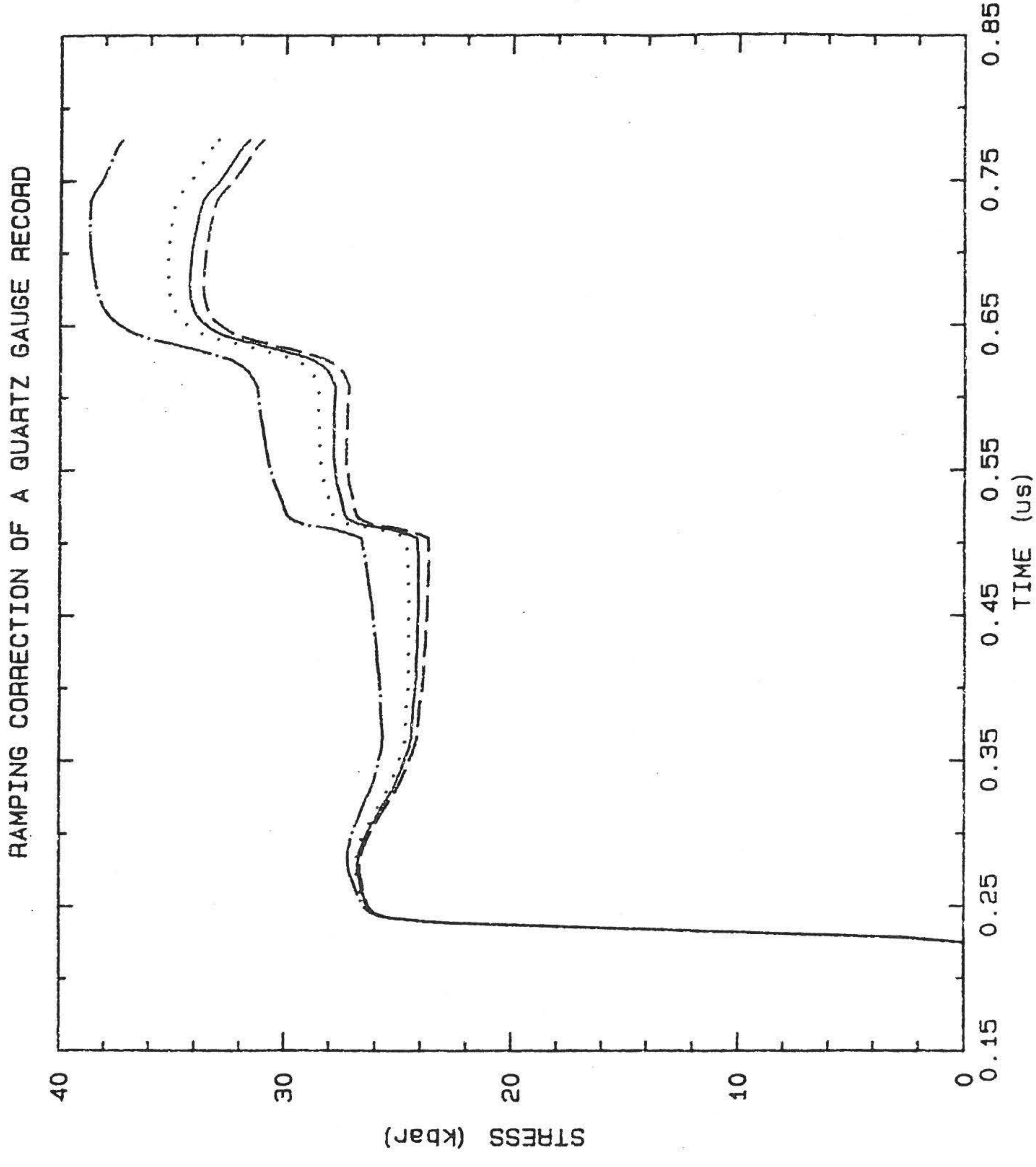
$\sigma_{j,n}$ is considered as the root of Eq. (7). ϵ in Eq. (11) is the control accuracy for the calculation.

II. Revision of the code Interp and the calculation sample

According to the calibration results described above, we have revised the shorted quartz gauge section in the code Interp (main program inter.r). Code Interp is a computer program used for reducing the measured data of the various kinds of the transducers, such as shorted quartz gauge, shunted quartz gauge and electro-magnetic velocity gauge, etc., in our laboratory. The calculation practice shows that the Newton's method is better than the exact solution method. A few times of iteration can reach the accuracy (10^{-4}) we wanted. Hence, we applied Newton's method in the code. Also, the code (main program) was changed a little bit to fit the new computer HP 9000 operation system. All the files in the code Interp are written by rational Fortran. In the Appendix, only the source file of the main program interp.r is shown. Other files are not revised and not shown in this report. It needs to say that the operation procedure of using the new Interp is same as the old one. The only difference is that when you ask for ramping correction, the new Interp uses the Eq. (7) described above and the old one uses the constant ramping coefficient (α=0.26 when $\sigma\leq15kbar$, and α=0.41 when $\sigma>15kbar$).

In Fig. 3 we show the results of an experiment in which a cadmium sulphide single crystal was impacted on a thin piece of a z-cut sapphire backed by a quartz gauge.[6] The impact stress in the CdS crystal exceeded the phase transition stress and, hence, a time-dependent response was expected in the CdS crystal. The stress between sapphire and quartz gauge was about 25 kbar for the first step (the stress relaxation phenomenon can be observed from the record). In Fig. 3, the dotdashed line is converted without ramping correction, the solid line is the result using this method described above. For comparison, it also shows the reduction results using a constant α.

Fig. 3. Typical stress-time profile at the impact surface of a shocked CdS single crystal. The dotdashed line — without the ramping correction; solid line — with ramping correction using Eq. (7); the dotted line and dashed line — using $\alpha = 0.32$ & 0.48 respectively.



The dotteded line in Fig. 6 is the result using a constant α=0.32 (i.e. σ=15$kbar$),and the dashed line α=0.48(i.e. σ=35$kbar$). Although the errors of α of the dotted and dashed lines are as high as 20% compared to 25 kbar stress step, the errors of the converted stress are only 0.3%, 1.3% and 2.7% at Δt=0.05μs, Δt=0.18μs and Δt=0.45μs respectively. According the analysis in Reference 1, the effect of the error in α on the revised current history is much less than the error in α itself. One could imagine that the error caused by the uncertainty of α between the actual stress history and the solid line reduced by means of the present method would be quite small. It means that the shorted gauge can be applied for measuring the stress history at later times with enough accuracy, if suitable data reduction method is used.

ACKNOWLEDGMENT

Professor Y.M. Gupta is sincerely thanked for his helpful discussion and advice.

REFERENCES

1. Z.P. Tang, Y.M. Gupta and P.M. Bellamy, Impact response of the half-inch shorted quartz gauge to 40 kilobar, to be published.
2. D.B. Hayes and Y.M. Gupta, Rev. Sci. Instr. 45, 1554 (1974).
3. J.J. Dick, G.E. Duvall and J.E. Vorthman, J. Appl. Phys., 47(9), 3987, (1976).
4. G.E. Duvall, "Impact response of the shorted quartz gauge to 40 KB", Internal Report 73-03, Shock Dynamics Laboratory, Washington State University (1973), unpublished.
5. K.S. Tunison, private communication.
6. Z.P. Tang and Y.M. Gupta, to be published.

APPENDIX

The Main Program: interp.r

```
include interp.h
#
#
#       Main Interp program
#
        integer i, limit, nrec, ntcal, nvcal, gauge
        integer*2 line(MAXBUFF), temp(MAXBUFF), btime(LENTIME)
        integer*2 time(LENTIME), header(MAXBUFF)
        real*8  vcal(MAXCAL), tcal(MAXCAL), trec(MAXREC), vrec(MAXREC)
        real*8  tmprec(MAXREC), tmp2rec(MAXREC), tmp3rec(MAXREC)
        real*8  tdiv, vdiv, num, v0, v1, c3, smax, alpha, tmp, thick, area
        real*8 fact, dvr
        logical suces, suces2, success, ramping
#
#       string constants
        btime(1)=ichar('T')
        btime(2)=ichar('I')
        btime(3)=ichar('M')
        btime(4)=ichar('E')
        time(1)=ichar('t')
        time(2)=ichar('i')
        time(3)=ichar('m')
        time(4)=ichar('e')
#
#       Main Program
#       Attach external files to logical unit numbers
#
#   MODIFIED FOR HP 9000
#       call ioinit(.false.,.false.,.false.,'FORT',.false.)
        open(UNIT=FORT01, STATUS="UNKNOWN", FILE=".interp01")
        open(UNIT=FORT02, STATUS="UNKNOWN", FILE=".interp02")
#
        write(STDOUT,330); 330 format('Select one of the following Gauges')
        write(STDOUT,331); 331 format('-----------------------------------')
        write(STDOUT,332); 332 format('(0) ******** No Gauge ********')
        write(STDOUT,333); 333 format('(1) - RP Gauge')
        write(STDOUT,334); 334 format('(2) - Short Quartz Guage')
        write(STDOUT,335); 335 format('(3) - EMV Guage')
        write(STDOUT,336); 336 format('(4) - Shunted Quartz Guage (lin)')
        write(STDOUT,337); 337 format('(5) - Shorted Quartz Guage (lin)')
#       write(STDOUT,800); 800 format(' *0')
        call readline(STDIN,line)
#       write(STDOUT,801); 801 format(' *1')
        i = 1
        call atof(line,i,num)
#       write(STDOUT,802); 802 format(' *2')
        gauge = idint(num)
        call readline(FORT01,line)
#       write(STDOUT,803); 803 format(' *3')
        while(line(1) != EOF) {
            write(STDOUT,210); 210 format('Calibration records titles:')
            call writeline(STDOUT,line)
#       write(STDOUT,804); 804 format(' *4')
            #find limit of string
            for (limit = MAXBUFF; line(limit) == BLANK; limit = limit - 1);
           #find out which calibration record is first
            call strsrch(time,LENTIME,line,limit,suces)
#       write(STDOUT,805); 805 format(' *5')
            call strsrch(btime,LENTIME,line,limit,suces2)
#       write(STDOUT,806); 806 format(' *6')
            success = suces .or. suces2
            if (success) {
#               time is first
                        call calinput(tcal,ntcal,line,FORT01,success)
                call writeline(STDOUT,line)
                call calinput(vcal,nvcal,line,FORT01,!success)
```

```
            }
            else {
#               voltage is first
                call calinput(vcal,nvcal,line,FORT01,success)
                call writeline(STDOUT,line)
                call calinput(tcal,ntcal,line,FORT01,!success)
            }
            do i = 1, MAXBUFF
                header(i) = line(i)
           #write out header on top of record
            write(STDOUT,201); 201 format('Record Title:')
            call writeline(STDOUT,line)
           #read in main record
            call inrecord(trec,vrec,nrec,line,FORT01)
            write(STDOUT,203)
            203 format('Timing mark interval in microseconds ?')
            call readline(STDIN,temp)
            i = 1
            call atof(temp,i,tdiv)
            write(STDOUT,204)
            204 format('Voltage mark interval in volts or amps ?')
            call readline(STDIN,temp)
            i = 1
            call atof(temp,i,vdiv)
           #interpolate record
            call interpolate(trec,nrec,tcal,ntcal,tdiv,0d0)
            call interpolate(vrec,nrec,vcal,nvcal,vdiv,0d0)
           #do action appropriate to gauge selected
            switch (gauge) {

                case EMVGUAGE:
                    call reduceit(trec,vrec,nrec)

                case RPGUAGE:
                    write(STDOUT,340); 340 format('RP Gauge :')
100                 write(STDOUT,341); 341 format('Input V0, V1')
                    call readline(STDIN,temp)
                    i = 1
                    call atof(temp,i,v0)
                    if (i < 0) go to 100
200                 call atof(temp,i,v1)
                    if (i > 0) go to 300
                    write(STDOUT,350); 350 format('Input V1, again please')
                    call readline(STDIN,temp)
                    i = 1
                    go to 200
300                 continue
                    do i=1, nrec
                        vrec(i) = (vrec(i) - v1) / v0

               case SQGUAGE:
                    write(STDOUT,380); 380 format('Short Quartz Guage')
                    write(STDOUT,382); 382 format('Input Area (mm^2)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,area)
                    write(STDOUT,383); 383 format('Input Thickness (mm)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,thick)
                    write(STDOUT,384)
384                 format('Do you wish to do ramping ?')
                    call readline(STDIN,temp)
                    ramping=.false.
                    if (temp(1) == SY || temp(1) == LY) {
                        ramping=.true.
                    }
#                   smax=vrec(1)
```

```
#                  do i=2,nrec {
#                      if (vrec(i) > smax) smax=vrec(i)
#                  }
#                  c3=(smax*thick*1.0e4)/(area*5.72)
#                  smax=(sqrt(C2*C2+4*C1*c3)-C2)/(2*C1)
#                  a1=0.1055
#                  a2=0.5123
#                  a3=-0.3685
#                  alpha=a1+a2*trec(i)+a3*(trec(i)**2)
#                  if (smax > 15) alpha=.41
#
#       I CHANGED TO alpha=0.195+0.008236*po, WHERE po IS STRESS OF
#         HEAVISIDE STEP, AND k(po)=1.919+0.008249*po, WHERE k(po)
#         IS CURRENT COEFFICIENT, SO C2=1.919, AND C1=0.008249,
#         BECAUSE C1*po**2+C2*po-C3=0. C1 AND C2 ARE DEFINED IN
#         FILE interp.h
#
#       ASSUMING CURRENT COEFFICIENT k AND RAMPING COEFFICIENT alpha ONLY
#         DEPEND ON THE STRESS AT THAT TIME, THEN WE CAN CONVERT THE CURRENT
#         RECORDS INTO THE STRESS HISTORY. (STRICTLY SPEAKING, IT SHOULD USE
#         NONLINEAR DECONVOLUTION METHOD TO DO SO, BUT IT NEEDS MORE CALI-
#         BRATION INFORMATION.)
#
#         write(STDOUT,386); 386 format(' Stress Plateau po= ?Kbar')
#         read (STDIN,387) po; 387 format(e14.6)
#         alpha=0.195+0.008236*po
#
#                  if (ramping) {
#                      tmp=vrec(1)
#                      do i=2,nrec {
#                          dvr=vrec(i)-tmp
#                          tmp=vrec(i)
#                          fact=alpha*(trec(i)-trec(i-1))/2.0
#                          vrec(i)=(dvr+(1-fact)*vrec(i-1))/(1+fact)
#                      }
#                  }
#         THIS IS THE TEXT AFTER CHANGING:
#           WHERE rc1--i record (j-1)
#                 rc2--i record (j)
#                 tr1--i true (j-1)
#                 vrec(j)--stress (j) after converting.
#

                  c3=(area*5.72)/(thick*1.0e4)
                  a1=0.195
                  a2=0.008236
                  b1=1.919
                  b2=0.008249
                  if(ramping) {
                    rc1=vrec(1)
                    tr1=vrec(1)
                    vrec(1)=(sqrt(b1*b1+4.*b2*tr1/c3)-b1)/(2.*b2)
                  do j=2,nrec {
#                 write(STDOUT,1510)vrec(j-1),vrec(j)
#1510             format(' vrec(j-1),vrec(j)',2e14.5)
                  rc2=vrec(j)
                  dt=trec(j)-trec(j-1)
                  A=0.25*a2*b2*c3*dt
                  B=c3*(b2+0.25*(2.*a1*b2+b2*a2*vrec(j-1)+b1*a2)*dt)
         C=b1*c3+0.5*b1*c3*dt*(a1+0.5*a2*vrec(j-1))+0.25*a2*dt*tr1
                  D=rc1-rc2+(0.5*a1*dt+0.25*a2*dt*vrec(j-1)-1.)*tr1
#    THE ROOT FORMULA OF A CUBIC EQUATION:
#                 p=C/A-B*B/(3.*A*A)
#                 q=2.*(B/(3.*A))**3-B*C/(3.*A*A)+D/A
#                 xx=(0.5*q)**2+(p/3.)**3
#                 w=sqrt(abs(xx))
#                 if((w-q/2.) < 0.) goto 1001
```

```
#                   y1=(w-q/2.)**(1./3.)
#                   goto 1002
#1001               y1=-(q/2.-w)**(1./3.)
#1002               if((-w-q/2.) < 0.) goto 1005
#                   y2=(-w-q/2.)**(1./3.)
#                   goto 1010
#1005               y2=-(w+q/2.)**(1./3.)
#1010               rc1=vrec(j)
#                   vrec(j)=y1+y2-B/3./A
#                   write(STDOUT,1500) A,B,C,D,rc1,tr1
#1500               format(' ABCDrc1tr1',6e11.3)
#                   write(STDOUT,1501) p,q,y1,y2,vrec(j)
#1501               format('pqy1y2vrec(j)',5e12.4)
#                   write(STDOUT,1502)vrec(j-1),rc2,xx,dt
#1502               format(' vrec(j-1),rc2,xx,dt',4e12.4)
#
#     USING NEWTON'S FORMULA TO CALCULATE THE ROOT OF A CUBIC EQUATION:
#
                    x1=0.
                    do k=1,50 {
                    f=A*x1**3+B*x1*x1+C*x1+D
                    f1=3.*A*x1*x1+2.*B*x1+C
                    x2=x1-f/f1
#                   write(STDOUT,1505)k,x2,f
#1505               format(' k,x2,f',i3,2e14.5)
                    if(abs(x2-x1) < 1.e-4) goto 1012
                    x1=x2
                    }
 1012               rc1=vrec(j)
                    vrec(j)=x2
                    tr1=c3*vrec(j)*(b1+b2*vrec(j))
                    }
                    goto 1011
                    }
                    do i=1,nrec {
                        c3=(vrec(i)*thick*1.0e4)/(area*5.72)
                        vrec(i)=(sqrt(C2*C2+4*C1*c3)-C2)/(2*C1)
                    }
 1011               continue

               case SHUNTGUAGE:
                    write(STDOUT,480); 480 format('Shunted Quartz Guage')
                    write(STDOUT,482); 482 format('Input Area (mm^2)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,area)
                    write(STDOUT,483); 483 format('Input Thickness (mm)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,thick)
                    write(STDOUT,484)
484                 format('Do you wish to do ramping ?')
                    call readline(STDIN,temp)
                    ramping=.false.
                    if (temp(1) == SY || temp(1) == LY) {
                        ramping=.true.
                    }
                    alpha=0.1548
                    if (ramping) {
                        tmp=vrec(1)
                        do i=2,nrec {
                            dvr=vrec(i)-tmp
                            tmp=vrec(i)
                            fact=alpha*(trec(i)-trec(i-1))/2.0
                            vrec(i)=(dvr+(1-fact)*vrec(i-1))/(1+fact)
                        }
                    }
                    cc1=5.5484e-6*area
```

```
                    cc2=1.144e-3*area
                    cc3=cc2*cc2
                    do i=1,nrec {
                        vrec(i)=(sqrt(cc3+4.0*cc1*vrec(i)*thick)-
                                cc2)/(2.0*cc1)
                    }

                case SHORTEDGUAGE:
#
#
#                       For Further information on the following data reduction te
#                       see G. Sutherland's thesis.
#
                    write(STDOUT,490); 490 format('Shorted Quartz Guage')
                    write(STDOUT,492); 492 format('Input Area (mm^2)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,area)
                    write(STDOUT,493); 493 format('Input Thickness (mm)')
                    call readline(STDIN,temp)
                    i=1;call atof(temp,i,thick)
                    write(STDOUT,494)
494                 format('Do you wish to do ramping ?')
                    call readline(STDIN,temp)
                    ramping=.false.
                    if (temp(1) == SY || temp(1) == LY) {
                            ramping=.true.
                    }
#                       make a copy of vrec
                        tmp = vrec(1)
                        do i=1,nrec {
                                vrec(i) = vrec(i) - tmp
                                tmprec(i)=vrec(i)
                                tmp2rec(i)=vrec(i)
                        }
                    if (ramping) {
#                               apply shunted gauge formula to give pressures from
#                               and scale shunted gauge results to adjust for shor
                                call trans1(tmprec,nrec,area,thick)
#                               find ramping from based upon experimental results
                                call trans2(tmp2rec,tmprec,trec,nrec)
                                do i=1,nrec {
                                        tmp3rec(i) = tmp2rec(i)
                                }
                                call trans1(tmp2rec,nrec,area,thick)
#                               calculate ramping correction values
                                call trans3(vrec,tmp2rec,tmp3rec,trec,nrec)
#                               find final corrected pressures
                                call trans1(vrec,nrec,area,thick)
                        }
                        else {
                                call trans1(vrec,nrec,area,thick)
                        }

                default :
#                       do nothing
            }
                write(STDOUT,101)
101             format('Do you want to adjust x data so that x0 = 0.0 ? ')
                call readline(STDIN,temp)
                if (temp(1) == SY || temp(1) == LY) {
                        do i=2,nrec {
                                trec(i)=trec(i)-trec(1)
                        }
                        trec(1)=0.0
                }
                write(STDOUT,102)
102             format('Do you want to adjust y data so that y0 = 0.0 ? ')
```

```
            call readline(STDIN,temp)
            if (temp(1) == SY || temp(1) == LY) {
                    do i=2, nrec {
                            vrec(i)=vrec(i)-vrec(1)
                    }
                    vrec(1)=0.0
            }
        #print out interpolated values
         call outrecord(trec,vrec,nrec,header,FORT02)
    }
#
    end
```