



This repository Search

Pull requests Issues Gist



zubaexy / MgSpall2

Unwatch 1

Star 0

Fork 0

branch: master

MgSpall2 / strength.f



zubaexy 3 minutes ago MgUMAT as obtained from Jeff on 22 June 2015

0 contributors

256 lines (218 sloc) 8.219 kB

Raw

Blame

History



```

1  C=====
2  C=====
3  c Calculate strength for basal slip
4  C-----
5
6      subroutine calc_str_bas(gambas, gamtw, epsl, temp, hbastype,
7      & hbas1, hbas2, hbas3, hbas4, hbas5, hbas6, qbastw, qbass1, ysbas)
8
9      implicit none
10
11      !input
12      double precision gambas, epsl, gamtw, temp
13      integer hbastype
14      double precision hbas1, hbas2, hbas3, hbas4, hbas5, hbas6
15      double precision qbastw, qbass1
16      dimension gamtw(6)
17
18      !output
19      double precision ysbas
20
21      !util
22      integer n
23      double precision gamtwtot
24
25      gamtwtot = 0.0d+0
26      do n=1,6
27          gamtwtot = gamtwtot + dabs(gamtw(n))
28      end do
29
30      if (hbastype.eq.1) then
31  C-----
32  c hbastype = 1: Power law hardening
33  C-----
34      !sigy = sig0 + B*(gambas + qbastw * gamtwtot + qbass1*epsl)^n
35      !hbas1 = sig0
36      !hbas2 = B
37      !hbas3 = n
38      !hbas4-6 = not used
39      ysbas = hbas1 + hbas2*(dabs(gambas) + qbastw*gamtwtot +
40      & qbass1*dabs(epsl))*hbas3
41      else if (hbastype.eq.2) then
42  C-----
43  c hbastype = 2: Chang and Kochmann
44  C-----
45      !sigy = tau0 + sig0(1-exp(-h*gam/sig0))
46      !hbas1 = tau0
47      !hbas2 = sig0
48      !hbas3 = h
49      !hbas3-6 = not used

```

```

50     ysbas = hbas1+hbas2*(1.0d+0-dexp(-hbas3*dabs(gambas)/hbas2))
51     else
52         print*, 'BASAL STRENGTH TYPE ', hbastype, ' NOT IMPLEMENTED'
53     end if
54
55     return
56 end
57
58
59 C=====
60 C=====
61 c Calculate strength for tensile twinning
62 C-----
63
64     subroutine calc_str_tw(gambas, gamtw, epsl, temp, htwtype,
65 & htw1, htw2, htw3, htw4, htw5, htw6, qtwbas, qtwsl, ystw, twcap,
66 & captw)
67
68     implicit none
69
70     !input
71     double precision gambas, epsl, gamtw, temp
72     integer htwtype
73     double precision htw1, htw2, htw3, htw4, htw5, htw6
74     double precision qtwbas, qtwsl
75     dimension gamtw(6)
76
77     !output
78     double precision ystw(6), twcap
79     logical captw
80
81     !util
82     integer n
83     double precision gamtwtot, PI
84
85     PI = 3.14159265358979d+0
86     gamtwtot = gamtw(1)+gamtw(2)+gamtw(3)+gamtw(4)+gamtw(5)+gamtw(6)
87
88     if (htwtype.eq.1) then
89 C-----
90 c htwtype = 1: Bilinear - Graff et al., IJP 07
91 C-----
92 c if gamwtot < gamthresh
93 c tau^a = tau0 + h0*(gamwtot + qtwbas*gambas + qtwsl*epsl)
94 c else
95 c tau^a = tau0 + h0*( (gamwtot/gamthresh)^(m-1)
96 c + qtwbas*gambas + qtwsl*epsl)
97 c htw1 = tau0
98 c htw2 = h0
99 c htw3 = gamthresh
100 c htw4 = m (power law penalty exponent)
101 c htw5-6 = not used
102
103     if (gamwtot.lt.htw3) then
104         do n=1,6
105             ystw(n) = htw1 + htw2*(qtwbas*gambas + gamwtot
106 & + qtwsl*epsl)
107         end do
108     else
109         do n=1,6
110             ystw(n) = htw1 + htw2*((gamwtot/htw3)**(htw4-1.0d+0)
111 & + qtwbas*gambas + qtwsl*epsl)
112         end do
113
114         twcap = htw3

```

```

115         if (gamwtot.ge.twcap) then
116             captw = .true.
117         else
118             captw = .false.
119         end if
120     end if
121
122     else if (htwtype.eq.2) then
123 c-----
124 c   htwtype = 2: tangent hardening with taylor approx for twin, no
125 c               other latent hardening from other defm modes
126 c-----
127 c   tau^a = tau0 + h0*tan(pi*gamwtot/(2.0*frac*gamthresh))
128 c   htw1 = tau0
129 c   htw2 = h0
130 c   htw3 = gamthresh
131 c   htw4 = frac (0<=frac<=1) shifts cutoff max gamthresh
132 c   htw5-6 = not used
133 c   twcap = htw3*htw4
134 c   do n=1,6
135 c       ystw(n) = htw1 + htw2*dtan(gamwtot*PI/(2.0d+0*htw3))
136 c   end do
137 c   if (gamwtot.ge.twcap) then
138 c       captw = .true.
139 c   else
140 c       captw = .false.
141 c   end if
142
143     else if (htwtype.eq.3) then
144 c-----
145 c   htwtype = 3: Chang and Kochmann
146 c-----
147 c   htw1 = tau0
148 c   htw2 = h1
149 c   htw3 = h2
150 c   htw4 = gamthresh
151 c   htw5 = frac (0<=frac<=1) shifts cutoff max gamthresh
152 c   htw6 = not used
153 c   twcap = htw4*htw5
154 c   do n=1,6
155 c       ystw(n)=htw1+(htw2*gamtw(n)+htw3*(gamwtot-gamtw(n)))
156 c   & / htw4
157 c   end do
158 c   if (gamwtot.ge.twcap) then
159 c       captw = .true.
160 c   else
161 c       captw = .false.
162 c   end if
163 c   else
164 c       print*, 'TWIN STRENGTH TYPE ', htwtype, ' NOT IMPLEMENTED'
165 c   end if
166
167     return
168 end
169
170 c=====
171 c=====
172 c Calculate strength for non-basal slip
173 c-----
174
175     subroutine calc_str_sl(bis, gambslip, gamtw, epsl, epdsl, temp,
176 & tempmelt, hsltype, hsl1, hsl2, hsl3, hsl4, hsl5, hsl6, hsl7, hsl8,
177 & qslbas, qsltw, yssl, dyysdepd)
178
179     implicit none

```

```

180
181     !input
182     logical bis
183     double precision gambslip, epsl, epdsl, gamtw, temp, tempmelt
184     integer hsltype
185     double precision hsl1, hsl2, hsl3, hsl4, hsl5, hsl6, hsl7, hsl8
186     double precision qslbas, qsltw
187     dimension gamtw(6)
188
189     !output - yield strength, and deriv of ys wrt strain rate
190     double precision yssl, dyysdepd
191
192     !util
193     double precision thom
194
195     c IF MELTED, SET YSSL = 0, OTHERWISE CALC THOM
196     if (temp.ge.tempmelt) then
197         yssl = 0.0d+0
198     else
199         thom = (temp-293d+0)/(tempmelt-293d+0)
200         if (hsltype.eq.1) then
201             c-----
202             c hsltype = 1: Johnson-Cook, no latent from other defm modes
203             c uses a min epdsl (str rate) of props(6)
204             c-----
205             c sigy = (A+B*epsl**n)(1+C*ln(epdsl))(1-((t-t0)/(tmelt-t0))**m)
206             c hsl1 = A
207             c hsl2 = B
208             c hsl3 = n
209             c hsl4 = C
210             c hsl5 = m
211             c hsl6-7 = not used
212             c hsl8 = cutoff strain rate
213
214             if (epdsl.le.hsl8) then
215                 epdsl = hsl8
216             end if
217             yssl=(hsl1+hsl2*dabs(epsl)**hsl3)*(1.0d+0+hsl4*dlog(epdsl))
218             & *(1.0d+0-thom**hsl5)
219
220             if (bis) return
221
222             dyysdepd=hsl4*(1.0d+0-thom**hsl5)*(hsl1+hsl2*dabs(epsl)**hsl3)
223             dyysdepd = dyysdepd / epdsl
224
225             else if (hsltype.eq.2) then
226                 c-----
227                 c hsltype = 2: Chang and Kochmann
228                 c-----
229                 c sigy = (tau0+sig0*(1-exp(-h1*eps/sig0))+h2*eps)*(edot/edot0)**m
230                 c hsl1 = tau0
231                 c hsl2 = sig0
232                 c hsl3 = epd0
233                 c hsl4 = m
234                 c hsl5 = h1
235                 c hsl6 = h2
236                 c hsl7 = not used
237                 c hsl8 = cutoff strain rate
238
239                 if (epdsl.le.hsl8) then
240                     epdsl = hsl8
241                 end if
242                 yssl=(hsl1+hsl2*(1.0d+0-dexp(-hsl5*epsl/hsl2))+hsl6*epsl)*
243                 & (epdsl/hsl3)**hsl4
244
245                 if (bis) return

```

```
246         dyysdepd = hs14/hs13*(epds1/hs13)**(hs14-1.0d+0)*
247     &      (hs11+hs12*(1.0d+0-dexp(-hs15*eps1/hs12))+hs16*eps1)
248     else
249         print*, 'NB SLIP STRENGTH TYPE ', hsltype, ' NOT IMPLEMENTED'
250     end if
251
252 end if
253
254 return
255 end
```

