SAND2014-2551P

Sandia National Laboratories

Operated for the U.S. Department of Energy's
National Nuclear Security Administration
by **Sandia Corporation**

Albuquerque, New Mexico 87185-0747

date: March 25, 2014

to: File

from: Ernest Hardin, MS 0747 (06224)

subject: FISH program file for the Munson-Dawson multi-mechanism model of salt creep

FISH is a scripting language built into the FLAC code (Fast Langrangian Analysis of Continua), available from Itasca Consulting Group, Minneapolis, MN. Sandia holds a licensed copy of FLAC 7.0.411 (S/N 213-001-0683-16559). The FISH program file described here implements the Munson-Dawson (MD) constitutive model for creep of intact salt (Munson et al. 1989). A similar implementation was described by Jove-Colon et al. (2012). This program file was written entirely by E. Hardin, and submitted for Sandia Review & Approval as a step toward releasing it to members of the U.S.-German collaboration on repositories in salt (Hansen et al. 2013).

The script is shown in Table 1. Lines 1 to 208 comprise the MD constitutive model, and can be used with any FLAC process that invokes the CONFIG CREEP command. It follows the instructions for constitutive models described in the FLAC documentation. It is a non-recursive model except for the internal variable `mdiv` that is recursive over all time steps, and tracks a total strain measure used to control transition from transient to steady-state creep, on both loading and unloading.

The remainder of the script (lines 209 to 297) is a demonstration problem. It starts by declaring values for a set of parameters, all of which have names beginning with "`md.`" It then creates a 2-D grid with a semi-circular opening representing an underground drift (Figure 1), using a gridding method developed in Example 2.18 of the FLAC 7.0 Theory and Background manual. It assigns the constitutive model, sets the displacement boundary conditions, and loads the model domain from the top boundary. A number of histories are established to output displacement of the top, bottom and side of the drift opening. Finally, it sets the parameters for automatic time step adjustment, then executes the time stepping, and outputs a final history. An example output (x-displacement) is also shown in Figure 1.

Pursuant to review and approval of this memo, it will be available for unclassified, unlimited release.

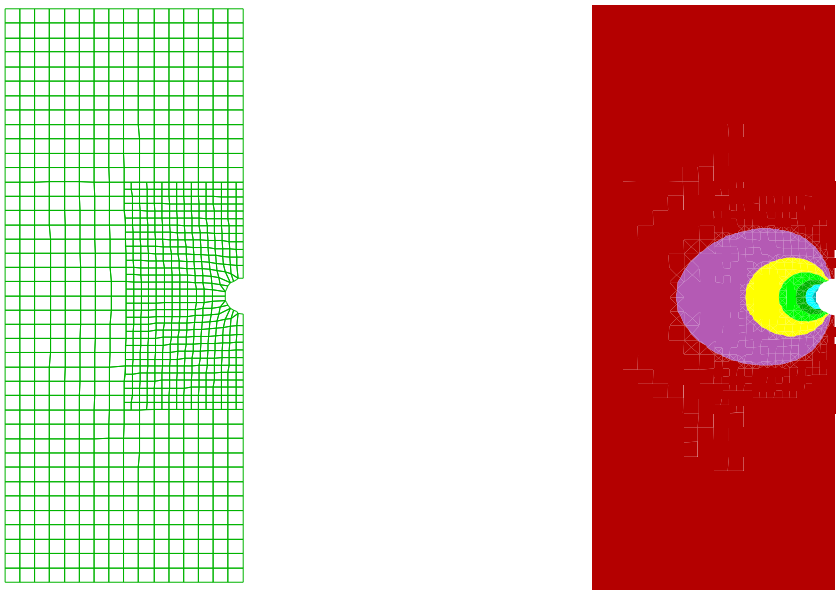

Figure 1. Demonstration problem results showing deformed model grid (left) and contoured x-displacement output (right) after approximately 30,000 time steps corresponding to 1.6×10^{9} seconds (~50 years).

Table 1. FISH script for Munson-Dawson multi-mechanisn constitutive model for intact salt (lines 1 to 208), and demonstration program.

```
1     ;
2     ; FISH user-defined constitutive model:
3     ; multi-mechanism intact salt model, steady-state part only
4     ;
5     def mdtransint
6      constitutive_model 198
7     ;
8      f_prop mdevs mdtvs mddevs
9      f_prop mddens mdiv sum_mdiv
10     f_prop mdepstar
11    ;
12     case_of mode
13    ;
14    ; --- initialization ---
15    ;
16      case 1
17    ;
18    ; intact salt model initialization
19    ;
20       mdepstar = 0.
21       mdiv = 0.
22       sum_mdiv = 0.
23    ;
24    ; --- running section --- intact salt model
25    ;
26      case 2
27    ;
28    ;  trap an initial elastic (crtdel <= 0.) calculation
```

```
29    ;
30       if crtdel <= 0. then
31    ;
32        mdetvol = zde11 + zde22 + zde33
33        mdetd11 = zde11 - mdetvol/3.
34        mdetd12 = zde12
35        mdetd22 = zde22 - mdetvol/3.
36        mdetd33 = zde33 - mdetvol/3.
37    ;
38        zs11 = zs11 + 2.*mdshear*mdetd11 + mdbulk*mdetvol
39        zs12 = zs12 + 2.*mdshear*mdetd12
40        zs22 = zs22 + 2.*mdshear*mdetd22 + mdbulk*mdetvol
41        zs33 = zs33 + 2.*mdshear*mdetd33 + mdbulk*mdetvol
42    ;
43       else
44    ;
45    ; stress invariants (plane strain)
46    ;
47        mdii1 = zs11 + zs22 + zs33
48        mdmean = mdii1/3.
49        mdii2 = zs11*zs22 + zs22*zs33 + zs11*zs33
50        mdii2 = mdii2 - zs12^2
51        mdii3 = zs11*zs22*zs33 - zs33*zs12^2
52        mdjj2 = mdii1*mdii1/3. - mdii2
53        mdjj3 = (2./27.)*mdii1^3 - mdii1*mdii2/3. + mdii3
54    ;
55    ; Lode angle (if J2 = 0 then J3 = Lode angle = 0)
56    ;
57        if mdjj2 = 0 then
58         mdlode = 0.
59         mdseff = 0.
60    ;
61         mds11 = zs11 - mdmean
62         mds12 = zs12
63         mds22 = zs22 - mdmean
64         mds33 = zs33 - mdmean
65    ;
66         mdt11 = 0.
67         mdt12 = 0.
68         mdt22 = 0.
69         mdt33 = 0.
70    ;
71         mdecx1 = 0.
72         mdecx2 = 0.
73         mdseff = 0.
74         mdff = 1.
75        else
76         mdarg = -3.*1.73205081*mdjj3/(2.*mdjj2*sqrt(mdjj2))
77         if abs(mdarg) < (1.-mdepsilon) then
78          mdlode = asin(mdarg)/3.
79          mdseff = 2*sqrt(mdjj2)*mdcosp
80         else
81          mdlode = sgn(mdarg)*3.14159265358979/3.
82          mdseff = sqrt(3*mdjj2)
83         endif
84    ;
85    ; update current stresses zs11, zs22, zs33, zs12
86    ; first calculate term coefficients for flow function
87    ;
88         mdcosp = cos(mdlode)
```

```
89          mdcos3p = cos(3.*mdlode)
90          mdecx1 = cos(2.*mdlode)/(mdcos3p*sqrt(mdjj2))
91          mdecx2 = 1.73205081*sin(mdlode)/(mdcos3p*mdjj2)
92      ;
93      ; then deviator tensor and t-tensor
94      ;
95          mds11 = zs11 - mdmean
96          mds12 = zs12
97          mds22 = zs22 - mdmean
98          mds33 = zs33 - mdmean
99      ;
100         mdt11 = mds11^2 + mds12^2 - 2.*mdjj2/3.
101         mdt12 = mds11*mds12 + mds12*mds22
102         mdt22 = mds12^2 + mds22^2 - 2*mdjj2/3.
103         mdt33 = mds33^2 - 2.*mdjj2/3.
104     ;
105     ; find transient strain limit and scaling function F
106     ;
107         mdepstar = mdkk0*exp(mdc*mdtemp)*((mdseff/mdshear)^mdm)
108         if mdiv < mdepstar then
109          mdworkhard = mdalphaw+mdbetaw*log(mdseff/mdshear)
110          mdff = exp(mdworkhard*(1.-(mdiv/mdepstar))^2)
111         endif
112         if mdiv = mdepstar then
113          mdff = 1.
114         endif
115         if mdiv > mdepstar then
116          mdff = exp(-mdrecovery*(1.-(mdiv/mdepstar))^2)
117         endif
118     ;
119       endif
120     ;
121     ; calculate multi-mechanism creep rates
122     ;
123       mdeps1dot = mdaa1*((mdseff/mdshear)^mdn1)*exp(-mdqq1rr/mdtemp)
124       mdeps2dot = mdaa2*((mdseff/mdshear)^mdn2)*exp(-mdqq2rr/mdtemp)
125       if mdseff >= mdsig0 then
126        mdeps3dot = mdbb1*exp(-mdqq1rr/mdtemp)+mdbb2*exp(-mdqq2rr/mdtemp)
127        mdeps3dot = mdeps3dot*sinh(mdq*(mdseff-mdsig0)/mdshear)
128       else
129        mdeps3dot = 0.
130       endif
131       mdepsdot = mdeps1dot + mdeps2dot + mdeps3dot
132     ;
133     ; total creep strain rates
134     ;
135       mdecr11 = mdff*mdepsdot*(mdecx1*mds11 + mdecx2*mdt11)
136       mdecr12 = mdff*mdepsdot*(mdecx1*mds12 + mdecx2*mdt12)
137       mdecr22 = mdff*mdepsdot*(mdecx1*mds22 + mdecx2*mdt22)
138       mdecr33 = mdff*mdepsdot*(mdecx1*mds33 + mdscx2*mdt33)
139     ;
140     ; decompose creep strain rate into vol. and dev. parts
141     ;
142       mdecrmean = (mdecr11 + mdecr22 + mdecr33)/3.
143       mdecrd11 = mdecr11 - mdecrmean
144       mdecrd12 = mdecr12
145       mdecrd22 = mdecr22 - mdecrmean
146       mdecrd33 = mdecr33 - mdecrmean
147     ;
148     ; decompose total strain increments into vol. and dev. parts
```

```
149   ;
150       mdetmean = (zde11 + zde22 + zde33)/3.
151       mdetd11 = zde11 - mdetmean
152       mdetd12 = zde12
153       mdetd22 = zde22 - mdetmean
154       mdetd33 = zde33 - mdetmean
155   ;
156   ; new total strain rates corrected for creep strain rates
157   ;
158       mderv = mdetmean/crtdel - mdecrmean
159       mderd11 = mdetd11/crtdel - mdecrd11
160       mderd12 = mdetd12/crtdel - mdecrd12
161       mderd22 = mdetd22/crtdel - mdecrd22
162       mderd33 = mdetd33/crtdel - mdecrd33
163   ;
164   ; then Cauchy stress rates
165   ;
166       mdsr11 = 2.*mdshear*mderd11 + mdbulk*mderv*3
167       mdsr12 = 2.*mdshear*mderd12
168       mdsr22 = 2.*mdshear*mderd22 + mdbulk*mderv*3
169       mdsr33 = 2.*mdshear*mderd33 + mdbulk*mderv*3
170   ;
171   ; and new stresses
172   ;
173       zs11 = zs11 + crtdel*mdsr11
174       zs12 = zs12 + crtdel*mdsr12
175       zs22 = zs22 + crtdel*mdsr22
176       zs33 = zs33 + crtdel*mdsr33
177   ;
178   ; update internal variable
179   ;
180       mdivrate = (mdff-1.)*mdepsdot
181       sum_mdiv = sum_mdiv + crtdel*mdivrate
182   ;
183   ; subzone accumulation logic for accumulated quantities
184   ; update internal variable
185   ;
186       if zsub > 0. then
187        mdiv = mdiv + sum_mdiv/zsub
188        sum_mdiv = 0.
189       endif
190      endif
191   ;
192   ; --- max modulus ---
193   ;
194     case 3
195      dummy = out('case 3')
196      cm_max = mdbulk + 4.*mdshear/3.
197      sm_max = mdshear
198   ;
199   ; --- thermal stresses ---
200   ;
201     case 4
202      dummy = out('case 4')
203   ;  ztsa = ztea*mdbulk
204   ;  ztsb = zteb*mdbulk
205   ;  ztsc = ztec*mdbulk
206   ;  ztsd = zted*mdbulk
207     end_case
208    end ; end of mdtransint fish model
```

```
;
; demonstration problem for mdtransint model (ventilation drift)
; initialize properties for creep model
; don't use property statements for these--
; unless they are to be set interactively in FLAC
;
set mdshear 1.24e10
set mdbulk 2.07e10
set mdepsilon 0.000001
;
set mdtemp 313.15 ; uniform rock temperature
set mdaa1 8.386e22
set mdn1 5.5
set mdqq1rr 12580
set mdaa2 9.672e12
set mdn2 5.
set mdqq2rr 5032
set mdbb1 6.086e6
set mdbb2 3.034e-2
set mdq 5335
set mdsig0 20.57e6
set mdkk0 6.275e5
set mdc 9.198e-3
set mdm 3.
set mdalphaw -17.37
set mdbetaw -7.738
set mdrecovery 0.58
;
; set up demonstration grid, models and BCs
; see FLAC 7.0 manuals for theory and syntax
; grid comes from Example 2.18 of the Theory and Background manual
;
config creep
grid 33 40
mod elas  ; to see the grid
prop density 2140 shear 1.24e10 bulk 2.07e10
set gravity = 9.81
set large
;
; make holes in big grid for inserts
;
mod null i=17
mod null i=17,33 j=33,40
mod null i=9,16 j=13,28
;
; now insert separated blocks with squeezing,
; using grid points
;
gen 8.0,12.0 8.0,28.0 16.0,28.0 16.0,12.0 i=18,34 j=1,33
;
; now attach blocks, also using grid points
; start with the "long" way around, then do the direct
; mapping comes from previous block position, bounded by null
;
attach as      from 9,13 to 17,13 bs      from 18,1 to 34,1
attach as      from 9,13 to 9,29 bs      from 18,1 to 18,33
attach as      from 9,29 to 17,29 bs      from 18,33 to 34,33
;
gen circ 16.0 20.0 1.88
mod null reg 33,16
```

```
;
fix x i=1
fix x i=17
fix x i=34 j=1,33
fix y j=1 i=1,17
;
apply syy -14e6 from 1,41 to 17,41
;
solve
;
mod mdtransint reg i=2 j=2
mod mdtransint reg i=19 j=2
set creeptime 0.0
history 1 crtime
history 2 unbalanced
history 3 ydisp i=34, j=21
history 4 ydisp i=34, j=13
history 5 xdisp i=30, j=17
;
set mindt=1.0E-6
set maxdt=2.0E5
set fobl=5000.
set fobu=50000.
set lmul=2.
set umul=0.5
set crdt=auto
;
step 28000
history write 3 4 5 vs 1 begin 3000 end 28000 skip 100
```

References

Hansen, F.D., K. Kulmann, W. Steininger and E. Biurrun 2013.*Proceedings of 3rd US/German Workshop on Salt Repository Research, Design and Operation.* Sandia National Laboratories, Albuquerque, NM. SAND2013-1231P.

Jove-Colon, C.F., J.A. Greathouse, S. Teich-McGoldrick, R.T. Cygan, T. Hadgu, J.E. Bean, M.J. Martinez, P.L. Hopkins, J.G. Argüello, F.D. Hansen, F.A. Caporuscio, M. Cheshire, S.S. Levy, M.K. McCarney, H.R. Greenberg, T.J. Wolery, M. Sutton, J. Rutqvist, C.I. Steefel, J. Birkholzer, H.-H. Liu, J.A. Davis, R. Tinnacher, I. Bourg, M. Holmboe and J. Galindez 2012. *Evaluation of Generic EBS Design Concepts and Process Models: Implications to EBS Design Optimization*. U.S. Department of Energy, Office of Used Nuclear Fuel Disposition. FCRD-USED-2012-000140.

Munson, D.E., Fossum, A.F., and Senseny, P.E. 1989. *Advances in Resolution of Discrepancies between Predicted and Measured WIPP In-situ Room Closures*. Sandia National Laboratories, Albuquerque, NM. SAND88-2948.