3 LIBRARY OF FISH FUNCTIONS

This section contains a library of FISH functions that have been written for general application in FLAC analysis. The functions can be used for various aspects of model generation and solution, including grid generation, plotting, assigning material properties and solution control.

The functions are divided into seven categories:

1. model generation;
2. general utility;
3. plotting;
4. solution control;
5. constitutive model;
6. groundwater analysis; and
7. special purpose.

The functions and their purpose are summarized by category in Tables 3.1 to 3.7. Each function is described individually, and an example application is given in this section. The functions are listed after the tables, in alphabetical order by filename (with the extension “.FIS”).

The FISH function files for the first six categories are FLAC-specific and are contained in the “\FISH\3-LIBRARY” directory. The files in the seventh category will operate with other Itasca programs, such as FLAC3D or UDEC, as well as FLAC. These files are contained in the “\Fishtank” directory.

The general procedure to implement these FISH functions is performed in four steps:

1. Make sure you have enough extra arrays (CONFIG extra=n). n should be equal to or greater than the number of extra arrays noted in Tables 3.1 to 3.7.

2. The FISH file is first called by the FLAC data file with the command

   call filename.fis

   If the selected FISH function requires other FISH functions to operate, as noted in Tables 3.1 to 3.7, these must also be in the working directory.

3. Next, FISH variables, if listed in Tables 3.1 to 3.7, must be set in the data file with the command

   set var1 = value var2 = value ...

   where var1, var2, etc. are the variable names given in Tables 3.1 to 3.7, which must be set to specified values.
If properties are required for *FISH* constitutive models (see Table 3.5), these are supplied with the `PROPERTY` command – i.e.,

```
property prop1=value prop2=value ...
```

for which `prop1`, `prop2`, etc. are property names.

4. Finally, the *FISH* function is invoked by entering the command (or commands) noted in Tables 3.1 to 3.7.

The *FISH* functions interact with *FLAC* in various ways. The user should consult Section 2.5 for a description of the different types of linkages between *FISH* and *FLAC*. It is recommended that users review the tables in this section, and the files in the “*FISH\3-LIBRARY*” directory, for *FISH* functions which may assist them with their *FLAC* analyses or provide a guide to develop their own *FISH* functions.
### Table 3.1  Model generation FISH functions

<table>
<thead>
<tr>
<th>Filename (.FIS)</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables SET before use</th>
<th>Number of Extra Grid Variables (CONFIG (extra)</th>
<th>Other FISH Function Required (.FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEAM</td>
<td>beam</td>
<td>creates structural beam elements around segment of internal boundary</td>
<td>ib ie jb je nprop</td>
<td>1</td>
<td>BOUNG</td>
</tr>
<tr>
<td>DDONUT</td>
<td>ddonut</td>
<td>creates a radial mesh of two holes symmetric about the vertical axis</td>
<td>rmin rmax rtrans rz ratio h_to_w distance</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>DONUT</td>
<td>donut</td>
<td>generates donut-shaped grid</td>
<td>rmin rmul gratio</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>HOLE</td>
<td>hole</td>
<td>generates quarter-symmetry radial mesh with square outer boundaries</td>
<td>rmin rmul gratio</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>QDONUT</td>
<td>qdonut</td>
<td>generates quarter-symmetry donut-shaped grid</td>
<td>rmin rmul gratio</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>REGADD</td>
<td>reg_add</td>
<td>adds a displacement in the x and y directions to gridpoint in a region</td>
<td>x_add y_add i_reg j_reg</td>
<td>1</td>
<td>REGION</td>
</tr>
</tbody>
</table>

**FLAC Version 6.0**
### Table 3.2 General utility FISH functions

<table>
<thead>
<tr>
<th>Filename (.FIS)</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables SET before use</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (.FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUNG</td>
<td>—</td>
<td>finds boundary gridpoints</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>BOUNZ</td>
<td>—</td>
<td>finds boundary zones</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>PRSTRUC</td>
<td>pr_struc</td>
<td>prints selected structural element variables</td>
<td>b_space</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>PS3D</td>
<td>ps3d</td>
<td>computes 3D principal stresses</td>
<td>—</td>
<td>3</td>
<td>—</td>
</tr>
<tr>
<td>REGION</td>
<td>region</td>
<td>sets extra variables for gridpoints inside a region</td>
<td>i_reg, j_reg</td>
<td>1</td>
<td>—</td>
</tr>
</tbody>
</table>
### Table 3.3 Plotting FISH functions

<table>
<thead>
<tr>
<th>Filename (.FIS)</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables SET before use</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (.FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPMAG</td>
<td>disp_mag</td>
<td>calculates displacement magnitude at grid point to generate contour plot</td>
<td>—</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>EXTRAP</td>
<td>extrap_to_gp</td>
<td>extrapolates zone-based field to gridpoints to generate contour plots that extend to model boundaries</td>
<td>gp_avg</td>
<td>4</td>
<td>LUDA</td>
</tr>
<tr>
<td>MCFOS</td>
<td>mc_fos</td>
<td>plots strength/stress ratios for different Mohr-Coulomb materials</td>
<td>—</td>
<td>4</td>
<td>PS3D</td>
</tr>
<tr>
<td>PQ</td>
<td>history qs</td>
<td>calculates stress points p and q to generate a p-q diagram</td>
<td>iv  jv</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>history ps</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>ps</td>
<td>plots phreatic surface</td>
<td>—</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3.4  Solution control FISH functions

<table>
<thead>
<tr>
<th>Filename (.FIS)</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables SET before use</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (.FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVO</td>
<td>servo</td>
<td>control to minimize inertial response to applied conditions</td>
<td>high_unbal, low_unbal, high_vel</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>ZONK</td>
<td>zonk, relax</td>
<td>gradually extracts region of zones to simulate excavation</td>
<td>i1, j1, i2, j2, n_small_steps, n_big_steps</td>
<td>7</td>
<td>—</td>
</tr>
</tbody>
</table>
### Table 3.5 Constitutive model FISH functions

<table>
<thead>
<tr>
<th>Filename (FIS)</th>
<th>Command (Model Name)</th>
<th>Purpose</th>
<th>Properties</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMCLAY</td>
<td>m_camclay</td>
<td>FISH version of modified Cam-clay model</td>
<td>m_g m_k m_kappa m_lambda m_m m_rank m_pi m_pois m_v1 m_v2</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>DRUCKER</td>
<td>m_drucker</td>
<td>FISH version of Drucker-Prager failure model</td>
<td>m_g m_k m_kappa m_m m_rank m_v1 m_v2</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>DY</td>
<td>m_dy</td>
<td>FISH version of double yield model</td>
<td>m_g m_h m_coh m_fric m_dil m_tan</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>ELAS</td>
<td>m_elas</td>
<td>FISH version of elastic-isotropic model</td>
<td>m_g m_h</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>FINN</td>
<td>finn</td>
<td>pore pressure generation model based on Finn approach</td>
<td>ff_g ff_fric ff_tan ff_c2 ff_c4</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>HOEK</td>
<td>supolve</td>
<td>generates a Hoek-Brown failure surface by manipulating the Mohr-Coulomb model</td>
<td>mck* ssi ssr</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>HYP</td>
<td>hyper</td>
<td>elastic hyperbolic law</td>
<td>h_mod yield</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>MOUNCANC</td>
<td>m_duncan</td>
<td>FISH version of Duncan-Chang model</td>
<td>d_bulk d_coh d_gmax d_k d_kb d_bu d_n d_ps d_shear d_ten d_fric</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>MOCR</td>
<td>m_mohr</td>
<td>FISH version of Mohr-Coulomb failure model</td>
<td>m_g m_coh m_dil m_k m_fric m_ten</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>SS</td>
<td>m_ss</td>
<td>FISH version of strain hardening/softening model</td>
<td>m_g m_coh m_dil m_g max m_k m_ten m_stab m_dil</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>SSINT</td>
<td>int_var</td>
<td>adjusts material properties locally along an interface to simulate strain-softening behavior</td>
<td>coh_tab* friu_tab</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>UBI</td>
<td>m_ubi</td>
<td>FISH version of ubiquitous joint model</td>
<td>m_g m_k m_coh m_fric m_dil m_ten m_jcoh m_jtan m_jang</td>
<td>0</td>
<td>—</td>
</tr>
</tbody>
</table>

*Properties for this model are specified with the `SET` command.

Note that if two different FISH constitutive models are used at the same time, there may be a conflict with property names. Properties should be renamed in this case.
Table 3.6  Groundwater analysis FISH functions

<table>
<thead>
<tr>
<th>Filename</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables SET before use</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (.FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMOD5</td>
<td>spup</td>
<td>scales fluid bulk modulus using permeability and zone dimensions to speed convergence to steady state</td>
<td>—</td>
<td>2</td>
<td>—</td>
</tr>
<tr>
<td>ININV</td>
<td>ininv</td>
<td>initializes stresses and pore pressures as a function of depth (no voids)</td>
<td>wth k0x k0z</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>INIV</td>
<td>i_stress</td>
<td>initializes stresses and pore pressures as a function of depth (with voids)</td>
<td>k0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>PS</td>
<td>ps</td>
<td>plots phreatic surface</td>
<td>—</td>
<td>1</td>
<td>—</td>
</tr>
<tr>
<td>QRATIO</td>
<td>hist qratio</td>
<td>calculates relative amount unbalanced flow</td>
<td>—</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>TURBO</td>
<td>—</td>
<td>extrapolates pore pressure change to speed convergence to steady state</td>
<td>—</td>
<td>2</td>
<td>FMOD5</td>
</tr>
</tbody>
</table>
### Table 3.7 Special purpose FISH functions

<table>
<thead>
<tr>
<th>Filename (FIS)</th>
<th>Command</th>
<th>Purpose</th>
<th>Variables Set</th>
<th>Number of Extra Grid Variables (CONFIG extra)</th>
<th>Other FISH Function Required (FIS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DER</td>
<td>derivative</td>
<td>finds the derivative of a table of values</td>
<td>der_in  der_out</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ERFC</td>
<td>erf</td>
<td>finds the error function of e_val</td>
<td>e_val</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>erfc</td>
<td>complementary error function of e_val</td>
<td>e_val</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPINT</td>
<td>exp_int</td>
<td>finds the exponential integral of e_val</td>
<td>e_val</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER</td>
<td>filter</td>
<td>filters acceleration record to remove frequencies above specified level</td>
<td>fc filter_in filter_out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>ffttransform</td>
<td>finds the fast Fourier transform power spectrum of a table of values</td>
<td>fft_in fft_out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FROOT</td>
<td>froot</td>
<td>finds the root of a function bracketed in an interval</td>
<td>c_x1 c_x2 func val</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>integrate</td>
<td>finds the integral of a table of values</td>
<td>int_in int_out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUDA</td>
<td>ludcmp</td>
<td>solves systems of equations using LU—decomposition</td>
<td>lu_nn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lubksb</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODRED</td>
<td>_modred</td>
<td>computes and compares modulus reduction and damping ratio curves</td>
<td>_glab _dlab _gnum _dnum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>number</td>
<td>printing a floating-point number with user-specified precision</td>
<td>given</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPEC</td>
<td>spectrum</td>
<td>finds the response spectrum of an accelerogram</td>
<td>acc_in sv_out pm_min damp sd_out sa_out pmax n_point</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3 - 10 FISH in FLAC

FLAC Version 6.0
Generating a Lined Tunnel Segment

Segments of excavations can be lined with structural beam elements by invoking the FISH routine “BEAM.FIS.” This function creates a series of STRUCT beam commands along the segment of the boundary selected by the user. The function “BOUNG.FIS” is called by “BEAM.FIS” to first identify boundary gridpoints. The user sets the starting gridpoint \((i_b, j_b)\) and ending gridpoint \((i_e, j_e)\) for the beam elements by using the SET command. STRUCT beam commands are generated between all gridpoints along the selected boundary. Note that the grid must be on the left of the direction implied by the starting gridpoint and ending gridpoint for the liner generation. The material property number for the structural elements can be set via SET nprop. The default is \(nprop = 1\). If only the beginning gridpoint is specified, \(i_e\) and \(j_e\) default to \(i_b, j_b\) so that a closed lining will be created. Note that this function only works correctly for internal boundaries.

The example data file “BEAM.DAT” illustrates the use of “BEAM.FIS” to create a lining for a horseshoe-shaped tunnel starting at gridpoint \(i_b = 6, j_b = 6\) and ending at gridpoint \(i_e = 16, j_e = 6\).

Data File “BEAM.DAT”

```
cfg 1
grid 20 20
mod elas
gen arc 10,10 15 10 180
mark i=1,16 j=6
mark j=6,11 i=6
mark j=6,11 i=16
mod null region 6,6
ca beam.fis
set ib=6 jb=6 ie=16 je=6
beam
plot hold beam
return
```
Figure 1  Lined horseshoe-shaped tunnel
Finding Boundary Gridpoints and Zones

It is often useful to identify which gridpoints or zones lie along the external boundary or internal boundaries of a model. This allows the user to perform operations on these gridpoints or zones directly, rather than search the entire grid whenever these entities must be identified. For example, it may be necessary to monitor tunnel closure, or calculate stresses and displacements at the outer boundary. It is also useful to know internal boundary gridpoints to assist with the input of structural element commands (for example, to generate lined tunnel segments – see “BEAM.FIS”).

Two FISH functions, boung and bounz, which identify gridpoints or zones which lie along external or internal boundaries, are available. When “BOUNG.FIS” or “BOUNZ.FIS” is called, the gridpoints or zones that are on a boundary are identified by the integer value 1; otherwise, they are assigned integer 0 in the grid variable \texttt{ex1}. The user may type

\begin{verbatim}
print ex1
\end{verbatim}

to check this assignment. Note that the \texttt{CONFIG extra} command must be specified to have the extra grid variable available.

In the example data file “BOUNG.DAT,” boundary gridpoints are identified using “BOUNG.FIS” and fixed for visual illustration.

Data File “BOUNG.DAT”

\begin{verbatim}
config ex 1
grid 20 20
model elas
  gen arc 10,10 15 10 180
  mark i=1,16 j=6
  mark j=6,11 i=6
  mark j=6,11 i=16
  model null region 6,6
  ca boung.fis
  def fix_it
    loop ii (1,igp)
      loop jj (1,jgp)
        if ex1(ii,jj) = 1 then
          command
            fix x i=ii j=jj
          command
        end_if
      end_loop
    end_loop
  end
  fix_it
plot hold bou bla fix
return
\end{verbatim}
Figure 1  Boundary gridpoints identification
**Modified Cam-Clay FISH Model**

The file “CAMCLAY.FIS” contains a FISH function which duplicates the built-in modified Cam-clay plasticity model in FLAC\textsuperscript{3D}. The detailed explanation of the model is provided in Section 2.4.7 in Theory and Background. The function is named \texttt{m\_camclay}, and requires that the following parameters be specified with the \texttt{PROPERTY} command:

- \texttt{m\_g} shear modulus
- \texttt{m\_k} maximum elastic bulk modulus
- \texttt{m\_kappa} slope of elastic swelling line, $\kappa$
- \texttt{m\_lambda} slope of normal consolidation line, $\lambda$
- \texttt{m\_m} material constant, $m$
- \texttt{m\_p1} reference pressure, $p_1$
- \texttt{m\_pc} pre-consolidation pressure, $p_c$
- \texttt{m\_poiss} Poisson’s ratio, $\nu$
- \texttt{m\_v1} specific volume at reference pressure on normal consolidation line, $v_\lambda$
- \texttt{m\_v0} initial value of specific volume, $v_0$

These parameters default to zero if not specified. In addition, the user has access to:

- \texttt{m\_e} total volumetric strain
- \texttt{m\_ep} plastic volumetric strain
- \texttt{m\_ind} state indicator:
  - 0 elastic
  - 1 plastic
  - 2 elastic now, but plastic in past
- \texttt{m\_kc} current elastic bulk modulus
- \texttt{m\_p} mean effective pressure
- \texttt{m\_q} deviator stress
- \texttt{m\_v} specific volume
The following data file exercises the FISH Cam-clay model for a normally consolidated material subjected to several load-unload excursions in an isotropic compression test. Note that the initial specific volume must be set to a value consistent with the initial effective pressure and the choice of model parameters before the Cam-clay model can be invoked (see Section 2.4.7.8 in Theory and Background). This is accomplished by entering the FISH command set*v0 before stepping. set*v0 is defined in “CAMCLAY.FIS.”

This test was also performed with the built-in model (see Example 3.43 in the User’s Guide). The results from the FISH model are identical to the built-in model.

Data File “CAMCLAY.DAT”

; Isotropic compression test on Cam-clay sample (drained)
; using CAMCLAY.FIS FISH function
;
config axis
g 1 1
tit
   Isotropic compression test for normally consolidated soil
; --- model properties ---
call camclay.fis
model m_camclay
prop m.g 250. m.k 10000. dens 1
prop m.m 1.02 m.lambda 0.2 m.kappa 0.05
prop m.pc 5. m.pl 1. m.vl 3.32
; --- boundary conditions ---
fix y
fix x
ini sxx -5. syy -5. szz -5.
iniv yvel -0.5e-4 j=2
ini xvel -0.5e-4 i=2
; --- fish functions ---
; ... numerical values for p, q, v ...
def path
   s1 = -syy(1,1)
s2 = -szz(1,1)
s3 = -sxx(1,1)
sp = (s1 + s2 + s3)/3.0
sq = sqrt(((s1-s2)*(s1-s2)+(s2-s3)*(s2-s3)+(s3-s1)*(s3-s1))*0.5)
sqcr= sp*m(1,1)
lnp = ln(sp)
svol = m_v(1,1)
mk = m_kc(1,1)
mk = m_kc(1,1)
end
; ... loading-unloading excursions ...
def trip
loop i (1,5)
  command
    ini yv -0.5e-4 xv -0.5e-4
    step 300
    ini xv mul -.1 yv mul -.1
    step 1000
    ini xv mul -.1 yv mul -.1
    step 1000
  end_command
end_loop
end
; --- histories ---
his nstep 20
his unbal
his path
his sp
his lnp
his sq
his sqcr
his svol
his mk
his mg
his ydisp i=1 j=2
; --- test ---
set v0 ; see CAMCLAY.FIS
trip
; --- results ---
plot his 3 vs -10 hold
plot his 7 vs 4 hold
plot his 8 9 vs -10 hold
save c_cfish.sav
ret
Figure 1  Pressure versus displacement

Figure 2  Specific volume versus ln p
**Figure 3**  Bulk and shear moduli versus displacement
**Two-Hole Radial Mesh**

The *FISH* file “DDONUT.FIS” creates a radial mesh with two holes. The mesh is symmetric about a vertical line through the center of the grid. The following variables are set to define the mesh:

- **distance**: distance between hole centroids
- **h_to_w**: ratio of total height to half the model width
- **ratio**: ratio of zone size change outside the \( r_{trans} \) region
- **rmax**: distance from hole centroid to outer boundary
- **rmin**: internal radius of holes
- **rtrans**: radial distance from hole centroid within which the zone ratio = 1
- **rz**: number of zones within distance \( r_{trans} \)

The data file “DDONUT.DAT” illustrates the use of this *FISH* function. Note that there must be an odd number of zones in the \( i \)-direction, and an even number in the \( j \)-direction.

**Data File “DDONUT.DAT”**

```plaintext
G 81 160
model el
call ddonut.fis
set rmin 0.0142 rmax 0.20 rtrans 0.050 rz 20 ratio 1.05
set h_to_w 2.0 distance 0.142
ddonut
plot hold grid
ret
```

---

FLAC Version 6.0
**Figure 1**  Two-hole model

**Figure 2**  Zoning in the vicinity of holes
Finding the Derivative of a FLAC Table

The FISH file “DER.FIS” integrates the values of a table and returns another table. The input table is specified with the `der_in` argument, and the output table is specified with the `der_out` argument. If the output table already exists, all entries in it will be deleted (see “TABDEL.FIS”) and replaced with new values. If the output table does not exist, one will be created.

The function calculates the slopes between points in the input table, and locates the value midway between the points in the output table. Therefore, there will be \( n-1 \) points in the resulting table if there are \( n \) points in the source table.

Figure 1 shows the result of taking the derivative of a simple cosine wave. Table 1 is the input data, and table 2 is the output. The input data is read using the HISTORY read command followed by a HISTORY write command to copy the data into a table.

Data File “DER.DAT”

```plaintext
new
g r 1 1
title Example of DERIVATIVE FISH function
hist read test01.his
hist write 1 table 1
;
c a tabdel.fis
c a der.fis
;
set der_in 1 der_out 2
derivative
;
plot hold table 1 line 2 line
```
Figure 1  Derivative of a FLAC table (table 1 is a cosine wave; table 2 is the derivative)
Plotting Displacement Magnitude Contours

The user may write a function to calculate special grid variables for plotting. For example, when the function `disp_mag` is invoked, the displacement magnitudes are calculated at all gridpoints in the model and stored in the FISH grid variable `ex_1`. This array can then be plotted with the `PLOT` command. By typing

```
plot ex_1 fill alias ‘displacement magnitude’
```

a filled contour plot will be generated. Note that this function requires that one extra grid variable be designated via the `CONFIG extra` command. The keyword `alias` is added to rename `ex_1` to displacement magnitude in the plot legend. Figure 1 shows the plot.

Data File “DISPMAG.DAT”

```
config extra 1
  call dispmag.fis
  grid 5 5
  model
  prop dens 1000 bulk 2e8 sh 1e8
  set grav 10
  fix x y j 1
  step 100
  disp_mag
  plot hold ex_1 fill alias ‘displacement magnitude’
```
Figure 1    Contours of displacement magnitude
Donut-Shaped Radial Mesh

The FISH file “DONUT.FIS” creates a donut-shaped mesh in which each gridpoint is defined by polar coordinates $\alpha$ and $\rho$. This function is similar to that in “HOLE.FIS,” and the same FISH variables are set with the SET command. The outer boundary is circular though, and the ATTACH command is used to connect the grid into a donut shape.

It should be noted that, at present, the IEB command cannot be used with this mesh because the infinite elastic boundary does not recognize the attached grid.

Data File “DONUT.DAT”

```plaintext
grid 10 40
me
call donut.fis
set rmin=1 rmul=10 gratio=1.1
donut
plot hold grid
return
```

![Figure 1 Donut-shaped mesh](image-url)
Drucker-Prager FISH Model

The file “DRUCKER.FIS” contains a FISH function which duplicates the built-in Drucker-Prager plasticity model. The detailed explanation of the model is provided in Section 2.4.1 in Theory and Background. The function is named m_drucker and requires that the following parameters be specified with the PROPERTY command:

- \( m_g \)  shear modulus
- \( m_k \)  bulk modulus
- \( m_{kshear} \)  material parameter, \( k_\phi \)
- \( m_{qdi} \)  material parameter, \( q_k \)
- \( m_{qvold} \)  material parameter, \( q_\phi \)
- \( m_{ten} \)  tensile strength

These parameters default to zero if not specified.

In addition, the user has access to:

- \( m_{ind} \)  state indicator:
  - 0  elastic
  - 1  plastic shear
  - 2  elastic now, but plastic in past
  - 3  plastic tensile

The following problem compares the FISH model to the built-in Drucker-Prager model. The built-in model is used for zones in the left half of the model; the FISH function is used for zones in the right half.

**Data File “DRUCKER.DAT”**

```plaintext
  g 12 10
  gen 0,0 0,25 30,25 30,0
  model drucker i=1,6
  prop den 2500 bulk 1.19e10 shear 1.1e10 i=1,6
  prop kshear 2.94e6 qvol 1.04 ten 2e6 i=1,6
  call drucker.fis
  model m_ss i=7,12
  prop den 2500 m_k 1.19e10 m_g 1.1e10 i=7,12
  prop m_{kshear} 2.94e6 m_{qvold} 1.04 m_{ten} 2e6 i=7,12
```

FLAC Version 6.0
ini xv 1e-6 i=1
ini xv -1e-6 i=13
fix x y i=1
fix x y i=13

his nstep 100
his unbal
his xdisp i=1 j=1
his sxx i=6 j=1
his sxx i=6 j=5
his sxx i=6 j=10
step 15000
plot hold bou estress disp
save drucker.sav
return

**Figure 1** Comparison of stresses and displacements
Double-Yield FISH Model

The file “DY.FIS” contains a FISH function which duplicates the built-in double-yield plasticity model. A detailed explanation of the model is provided in Section 2.4.6 in Theory and Background. The function is named \texttt{m\_dy} and requires that the following parameters be specified with the \texttt{PROPERTY} command (for reference, corresponding built-in property names are indicated in parentheses):

\begin{verbatim}
  m_coh          cohesion (cohesion)
  m_cpmx         current cap pressure (cap\_pressure)
  m_cptab        number of cap pressure table (cptable)
  m_ctab         number of cohesion table (ctable)
  m_dil          dilation angle (dilation)
  m_dtab         number of dilation table (dtable)
  m_dymul        moduli multiplier (multiplier)
  m_epvol        accumulated plastic volumetric strain (ev\_plas)
  m_fric         friction angle (friction)
  m_ftab         number of friction table (ftable)
  m_g            shear modulus (shear\_mod)
  m_k            bulk modulus (bulk\_mod)
  m_ten          tensile strength (tension)
  m_ttab         number of tension table (ttable)
\end{verbatim}

These parameters default to zero if not specified.

In addition, the user has access to:

\begin{verbatim}
  m_epdev        accumulated plastic shear strain
  m_epten        accumulated plastic tensile strain
\end{verbatim}
m_ind  state indicator:

0  elastic
1  plastic shear and/or volume
2  elastic now, but plastic in past
3  plastic tensile

The following problem compares the FISH model to the built-in double-yield model in a volumetric loading/unloading test. The built-in model is used for the bottom zone, and the FISH function for the top zone, of the model.

Data File “DY.DAT”

title
  Volumetric loading/unloading using dy and m_dy models
config axi
g 1 3

ca dy.fis
mo m_dy j=3
pro m_k 1110e6 m_g 507.7e6 m_pctab 1 m_dymul 10 j=3
pro den 1000 m_coh 1e10 m_ten 1e10 j=3
table 1 0 0 1 1.1e7

mo null j=2

mo dy j=1
pro bu 1110e6 sh 507.7e6 cptable 1 mul 10 j=1
pro den 1000 coh 1e10 ten 1e10 j=1

fix x i 2 j=1,2
fix y
ini yvel -1e-6 j=2
ini xvel -1e-6 i=2 j=1,2

fix x i 2 j=3,4
ini yvel -1e-6 j=4
ini xvel -1e-6 i=2 j=3,4

hist syy i=1 j=1
hist ydis i=1 j=2

hist syy i=1 j=3
hist ydis i=1 j=4
step 1000
init xv mul -.1
init yv mul -.1
step 900
plot hold his -1 -3 cross vs -2
plot hold his -2 -4 cross
ret

Figure 1  Comparison of vertical stress versus displacement for FISH model and DY model
Elastic FISH Model

The file “ELAS.FIS” contains a FISH function which replicates the built-in elastic-isotropic model in FLAC. The detailed explanation of the model is provided in Section 2.3.1 in Theory and Background. The function is named \texttt{m_elas} and requires that the following parameters be specified with the \texttt{PROPERTY} command:

- \texttt{m_g} shear modulus
- \texttt{m_k} bulk modulus

These parameters default to zero if not specified. The user also has access to:

- \texttt{m_dvol} volumetric strain increment (for one sub-zone)
- \texttt{m_vol} accumulated volumetric strain

The following problem compares the FISH model to the built-in elastic-isotropic model. The built-in model is used for zones in the left half of the model; the FISH function is used for zones in the right half of the model.

Data File “ELAS.DAT”

```
g 12 10
gen 0,0 0,25 30,25 30,0

model elas i = 1,6
prop den 2500 bulk 1.19e10 shear 1.1e10 i=1,6

call elas.fis
model m_elas i = 7,12
prop den 2500 m_k 1.19e10 m_g 1.1e10 i=7,12

ini xv 1e-6 i=1
ini xv -1e-6 i=13
fix x y i=1
fix x y i=13
his nstep 100
his unbal
his xdisp i=1 j=1
his sxx i=6 j=5
his sxx i=7 j=5
step 15000
save elas.sav
plot hold bou est disp
return
```
**Figure 1**  Comparison of stresses and displacements
**Error Function and Complementary Error Function**

The file “ERFC.FIS” contains two *FISH* functions that calculate the error function,

\[
\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{0}^{x} e^{-t^2} dt
\]

and complementary error function,

\[
\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt
\]

of a real variable \(x\) using the rational approximation in section 7.1.26 in Abramowitz and Stegun (1970). The error magnitude is less than \(1.5 \times 10^{-7}\). The value of \(x\) is defined by \(\text{e_val}\); the functions \(\text{erf}\) and \(\text{erfc}\) return the corresponding function value. The following data file plots functions \(\text{erf}\) and \(\text{erfc}\) in the interval \([0, 1.5]\).

**Data File “ERFC.DAT”**

```plaintext
new
title 'Error and Complementary Error Functions'
cache erfc.fis suppress
def plot_erf
    local dx = 1.5/20.
    local e_val = -dx
    local ii
    loop ii (1,21)
        e_val = e_val + dx
        xtable(1,ii) = e_val
        ytable(1,ii) = erf(e_val)
        xtable(2,ii) = e_val
        ytable(2,ii) = erfc(e_val)
    end_loop
end
@plot
plot hold table 1 line 2 line alias '1: erf 2:erfc' min 0
ret
```
Reference


---

**Figure 1**  
Error and complementary error functions
Exponential Integral Function

The file “EXPINT.FIS” contains a FISH function which calculates the exponential integral function:

\[ E_1(x) = \int_x^\infty \frac{e^{-t}}{t} dt \]

of a real and positive variable \( x \), using polynomial approximations in sections 5.1.53 and 5.1.54 in Abramowitz and Stegun (1970). The error magnitude is less than \( 2 \times 10^{-7} \) for \( x \leq 1 \), and less than \( 5 \times 10^{-5} \) for \( x > 1 \).

The value of \( x \) is defined by \texttt{e\_val}, and the function \texttt{exp\_int} returns the corresponding value of \( E_1 \). The following data file plots function \( E_1 \) in the interval \([0,1.6]\).

Data File “EXPINT.DAT”

```fish
new
title 'Exponential Integral Function'
ca expint.fis suppress
def plot_e1
  local dx = 1.6/20.
  local e_val = 0.
  local ii
  loop ii (1,20)
    e_val = e_val + dx
    xtable(1,ii) = e_val
    ytable(1,ii) = exp_int(e_val)
  end_loop
end
@plot_e1
plot hold table 1 line min 0
ret
```

Reference

Figure 1  Exponential integral function
Extrapolating a Zone-Based Field to Gridpoints

The contour-plotting logic in FLAC can be applied to either zone-based (e.g., stress) or gridpoint-based fields (e.g., displacement). Because the zone-based values are assumed to be constant over each FLAC zone, the contours generated for such fields do not extend to the model boundaries, whereas the gridpoint-based fields do extend to the model boundaries. The FISH function _extrap_ extrapolates a zone-based field to gridpoints. Thus, a contour of the extrapolated field will extend to the model boundaries.

The function operates upon the values in the _FISH_ grid variable _ex_1, and stores the extrapolated values in the _FISH_ grid variable _ex_2. (The values in _ex_1 are assumed to be zone-based, and the output values in _ex_2 are gridpoint-based.) A filled contour plot of the extrapolated field, along with the model boundaries, can be generated by typing

```
plot ex_2 fill bound
```

Note that this function requires that four extra grid variables be designated via the _CONFIG extra_ command.

Two different procedures are available to perform the extrapolation: a simple averaging and a least-squares fit. In the simple averaging procedure (invoked by setting the variable _gp_avg_ to 1), the value at each gridpoint is taken to be the average of the values from all non-null zones that use the gridpoint. In the least-squares-fit procedure (invoked by setting the variable _gp_avg_ to 0), the value at each gridpoint is found by assuming that the field can be described locally by the bilinear function

\[
f(x, y) = a_0 + a_1 x + a_2 y + a_3 x y
\]

where \(a_i\) are undetermined coefficients, and \(x\) and \(y\) are the global problem coordinates. The \(a_i\) values are found by sampling the function at the centroids of six nearby zones and performing a least-squares-fit. The value at the gridpoint is then found by evaluating the function at the gridpoint location.

Both extrapolation procedures operate in linear time (i.e., the execution time depends linearly on the number of gridpoints). The simple averaging procedure requires less execution time than the least-squares-fit procedure. Also, the simple averaging procedure will always produce fully symmetric contours for symmetric problems, while the least-squares-fit procedure may not. This behavior arises because the six nearby sampling locations are not guaranteed to be placed symmetrically for two symmetric gridpoints. The least-squares-fit procedure is the default, and is recommended for best accuracy.

An example problem demonstrating the use of the _FISH_ function _extrap_ is provided in the data file below. A circular hole in a block of elastic material loaded by gravity, and free to expand laterally at its base, is modeled. Symmetry conditions are employed such that only one-half of the system is modeled; the grid is shown in Figure 1. A contour plot of the vertical stresses, generated using the _PLOT syy_ command, is shown in Figure 2. Note that the contours do not extend to the model boundaries. The results of invoking the _FISH_ function _extrap_ using both of the available
extrapolation procedures in turn are shown in Figures 3 and 4. Both of these plots were generated using the PLOT ex.2 bound command after performing the extrapolation.

**Data File “EXTRAP.DAT”**

```plaintext
; Half of symmetric model of tunnel in elastic material,
; loaded by gravity and free to slide at base.
;
new
config extra=5
;
def fill_ex1_syy
; --- Loop through all zones
loop i (1,izones)
    loop j (1,jzones)
        ex1(i,j) = syy(i,j)
    end_loop
end_loop
end
;
grid 10 20
model elastic
gen circle 0.0 10.0 2.0 ; mark hole
model null region 1,11 ; make hole zones null
prop dens=1000 shear=0.3e8 bulk=1e8
set grav=9.81
fix x i=1
fix y j=1
solve
sclin 1 8.0 0.0 8.0 20.0
sav sag1.sav

Title Demonstration of extrap FISH function (grid and fixity)
plo hold grid fix

title
Demonstration of extrap FISH function (no extrapolation)
plo hold syy bound
;
call extrap.fis
fill_ex1_syy
;
set gp_avg=1
extrap_to_gp

title
Demonstration of extrap FISH function (simple average extrapolation)
plo hold bound ex.2 alias ‘yy-stress contours’
;
```

FLAC Version 6.0
set gp_avg=0
extrap_to_gp
title
Demonstration of extrap FISH function (least-squares-fit extrapolation)
plo hold bound ex_2 alias ‘yy-stress contours’
;
return

Figure 1  Grid and boundary conditions used for example problem
**Figure 2**  Vertical stress contours generated via the PLOT SYY command

**Figure 3**  Vertical stress contours generated after extrapolating the zonal-based values to gridpoints using the simple averaging procedure
**Figure 4** Vertical stress contours generated after extrapolating the zonal-based values to gridpoints using the least-squares-fit procedure.
Finding the Fast Fourier Transform Power Spectrum of a FLAC Table

The FISH file “FFT.FIS” performs a Fast Fourier Transform on a table of data, resulting in a power spectrum that is output to another table. The input table is specified by the first argument, and the output table is specified by the second argument.

There are several definitions for a power spectrum. The one used here is adapted from Press et al. (1992). The power spectrum is a set of $N/2$ real numbers defined as:

$$P_0 = \frac{1}{N^2} \ast (|f_0|^2)$$  \hspace{1cm} (1)

$$P_k = \frac{1}{N^2} \ast [(|f_k|^2 + |f_{N-k}|^2)]$$  \hspace{1cm} (2)

$$P_{N/2} = \frac{1}{N^2} \ast (|f_{N/2}|^2)$$  \hspace{1cm} (3)

where: $N$ is half the number of points in the original data field;

- $P$ is the power spectrum output;
- $f$ is the result of the Fast Fourier Transform of the original data; and
- $k$ varies from 0 to $N/2$.

Note that an array, worka, is used to manipulate the table data. The array dimension (n_point) is defined from the following conditions: (1) to be greater than the number of elements in the input table; and (2) to be a power of 2. (The array dimension need not be declared manually.) The fft algorithm requires input data with a constant timestep. So, a timestep is calculated, and the data is interpolated from the table and stored in the array for processing.

The following example verifies the fft FISH function. The history input is the sum of a sine wave at 1 Hz and an amplitude of 1, a cosine wave at 5 Hz and an amplitude of 2, and a sine wave at 10 Hz and an amplitude of 3. The combined history input is calculated by the FISH function cr.tab. The input is plotted in Figure 1. The power spectrum shown in Figure 2 consists of three sharp peaks at 1, 5 and 10 Hz, with increasing peak values.

Reference

Data File "FFT.DAT"

def cr_tab(num_point, end_time, per1):
    local i = 1
    local p2 = 2.*pi
    loop while i <= num_point
        local xx = end_time*float(i)/float(num_point)
        i = i + 1
        local yy = sin(xx*p2/per1)+2.*cos(5.*xx*p2/per1)+3.*sin(10.*xx*p2/per1)
        table(1,xx) = yy
    end_loop
end
@cr_tab(1024,12,1.0)

; ATTENTION: fft.fis uses a temporary setup to erase table

c a f f t . f i s suppress
@fftransform(1,2)
plot hold table 1 line
plot hold table 2 line
**Figure 1**  Sum of three input waves

**Figure 2**  Power spectrum; power versus frequency in Hz
Filtering Acceleration Records in a FLAC Table

The FISH file “FILTER.FIS” removes frequencies greater than $F_c$ from the signal (e.g., seismic acceleration) records stored in a FLAC table. The ID number of the input table is defined by `filter_in`, and the output table is defined by `filter_out`. If the output table currently exists, it will be deleted and overwritten with the new results. In addition, the cutoff-marking frequency $F_c$ needs to be provided before the function’s run.

The algorithm is implemented from the Butterworth filter (Rabiner and Gold, 1975). “FILTER.DAT” is presented to demonstrate how to use the function. In the example, an acceleration record (“ACC1.HIS”) is read into FLAC and then written to a table (Table 1); the FISH function (“FILTER.FIS”) processes the original data (Table 1) twice, with cutoff frequencies ($F_c$) of 10 Hz and 5 Hz, and the filtered data is saved into two different output tables (Tables 2 and 3) respectively; the efficiency of the algorithm is shown in the plots of three power spectra tables that are obtained by applying Fast Fourier Transform (see “FFT.FIS” and “FFT.DAT”) on the original data and the two filtered data tables.

It should be noted that some other parameters (e.g., `iorder`) are open to reset. Users are encouraged to review the Butterworth filter and modify the function according to the application requirements.

**Data File “FILTER.DAT”**

```plaintext
hist 100 read acc1.his
; original data in table 1
hist write 100 table 1
ca.filter.fis
;
; data with cutoff 10 Hz in table 2
set Fc=10.0
set filter_in = 1
set filter_out = 2
Filter
;
; data with cutoff 5 Hz in table 3
set Fc=5.0
set filter_in = 1
set filter_out = 3
Filter
;
; put power spectra (FFT.FIS) for original data in table 11
def tab_ind
  fft_in = 1
  fft_out = 11
end
tab_ind
ca.fft.fis
fftransform
;```
; put power spectra (FFT.FIS) for data with cutoff 10 Hz in table 12
set fft_in = 2
set fft_out = 12
fftransform
;
; put power spectra (FFT.FIS) for data with cutoff 5 Hz in table 13
set fft_in = 3
set fft_out = 13
fftransform
save Filter.sav

Reference

Pore Pressure Generation FISH Model Based on the Approach by Martin et al. (1975)

The file “FINN.FIS” is a FISH function that models dynamic pore-pressure generation based upon the approaches described by Martin et al. (1975) and Byrne (1991). The detailed explanation of this model is provided in Section 1.4.4.2 in Dynamic Analysis. The function is named finn and requires that the following parameters be specified with the PROPERTY command:

- `ff_c1`: constant $C_1$ in Eqs. (1.92) and (1.93), Section 1 in Dynamic Analysis
- `ff_c2`: constant $C_2$ in Eqs. (1.92) and (1.93), Section 1 in Dynamic Analysis
- `ff_c3`: constant $C_3$ in Eq. (1.92), Section 1 in Dynamic Analysis, and threshold shear strain for Eq. (1.93) in Dynamic Analysis
- `ff_c4`: constant $C_4$ in Eq. (1.92), Section 1 in Dynamic Analysis
- `ff_coh`: cohesion
- `ff_dil`: dilation angle in degrees
- `ff_fric`: friction angle in degrees
- `ff_g`: shear modulus
- `ff_k`: bulk modulus
- `ff_latency`: minimum number of timesteps between reversals
- `ff_switch`: = 0 for Eq. (1.92) in Dynamic Analysis, and 1 for Eq. (1.93) in Dynamic Analysis
- `ff_ten`: tension cutoff

In addition, the following variables may be printed or plotted for information:

- `ff_count`: number of shear strain reversals detected
- `ff_evd`: internal volume strain $\epsilon_{vd}$ of Eq. (1.92), Section 1 in Dynamic Analysis
The following example application compares the “FINN.FIS” model to the Finn model described in Section 1.4.4.2 in *Dynamic Analysis*. The example is the shaking table test given in Example 1.31 in *Dynamic Analysis*. The results are identical to those given for Example 1.31 in *Dynamic Analysis*. The data file for the *FISH* model is given in “FINN.DAT.”

**Data File “FINN.DAT”**

```
conf dyn gw
; shaking table test for liquefaction
call finn.fis

g 1 5
m finn
gen 0 0 0 5 50 5 50 0
fix x y j=1
fix x
set grav 10, flow=off
prop dens 2000 ff_g 2e8 ff_k 3e8
prop ff_fric 35 poros 0.5
water dens 1000 bulk 2e9 tens 1e10
ini pp 5e4 var 0 -5e4
ini syy -1.25e5 var 0 1.25e5
ini sxx -1e5 var 0 1e5 szz -1e5 var 0 1e5
prop ff_latency=50
; ; parameters for Martin formula
;prop ff_switch = 0
;prop ff_c1=0.8 ff_c2=0.79
;prop ff_c3=0.45 ff_c4=0.73
;
; ; parameters for Byrne formula
prop ff_switch = 1
def _setCoeff_Byrne
  ff_c1_ = 8.7*exp(-1.25*ln(n1_60_))
  ff_c2_ = 0.4/ff_c1_
  ff_c3_ = 0.0000
end
set n1_60_ = 7
_setCoeff_Byrne
prop ff_c1=ff_c1_ ff_c2=ff_c2_
prop ff_c3=ff_c3_
;
set ncwrite=50
def sine_wave
  while stepping
    vv = ampl * sin(2.0 * pi * freq * dytime)
    loop j (1,jzones)
      vvv = vv * float(jgp - j) / float(jzones)
```

*FLAC Version 6.0*
loop i (1,igp)
    xvel(i,j) = vvv
end_loop
end_loop
end
def eff_stress
    eff_stress = (sxx(1,2)+syy(1,2)+szz(1,2))/3.0 + pp(1,2)
    settlement = (ydisp(1,jgp)+ydisp(2,jgp))/2.0
end
set dy_damp=rayl 0.05 20.0
dytime
his pp i 1 j 2
eff_stress
his settlement
his nstep 20
set ampl=0.005 freq=5.0
solve dyt=10.0
plot hold his 2 3 vs 1 skip 2
save sh_tab.sav

**Figure 1**  Pore pressure (top) and effective stress (bottom) for shaking table test, using Eq. (1.93) in Dynamic Analysis and the FISH Finn model
References


Scaling Fluid Bulk Modulus to Speed Convergence to Steady State

When only the pore-pressure distribution corresponding to steady-state flow is of interest, a flow-only calculation may be performed. If there are substantial differences in permeability or zone size throughout the grid, the number of cycles needed to reach steady state may be large. The FISH function “FMOD5.FIS” may be used to speed the convergence in these cases.

In “FMOD5.FIS,” the local fluid bulk modulus is scaled for each zone, depending on the zone permeability and geometry. The variable \( fmod \) is set to a value that ensures optimal convergence. The scaling factor is chosen in such a way that the resulting average for \( fmod \) will equal the bulk modulus given in the WATER command. This modulus can therefore be changed by the user to speed up the calculation for free surface problems, as discussed in Section 1.4.2.1 in Fluid-Mechanical Interaction. The following data file illustrates the use of “FMOD5.FIS” to speed up the convergence toward steady state in a problem of flow around a high-permeability lens. See Section 1.10.4.1 in Fluid-Mechanical Interaction for additional discussion on the function “FMOD5.FIS.”

Data File “FMOD5.DAT”

```
conf gw extra=2
gr 10 10
model elas
prop d 1 s 1 b 1
gen circle 5 5 2.4
prop perm=1e-10 reg=1,1
prop perm=1e-9 reg=5,5
set mech=off
apply pp=0 i=1
apply pp=10 i=11
water bulk 1e9
step 1
call fmod5.fis
solve sratio 0.01
plot hold s1 blue pp i=.5 cyan
return
```
**Figure 1** Streamlines and pressure contours around a high-permeability lens
Root of a Function in an Interval

The file “FROOT.FIS” contains a FISH function that calculates the root of a function \( f(x) \), known to lie in the interval \([a, b]\), using Brent’s method algorithm (Press et al. 1986).

The FISH function `func` must be specified. It returns the value of \( f \) for the argument \( x \) defined by \( c_x \). The interval bounds \( a \) and \( b \) are assigned using \( c_x1 \) and \( c_x2 \), respectively. The root, returned as `froot`, is refined until its accuracy is \( tol \).

The following data file plots the function \( f(x) = \tan x - x \) and marks its root located in the interval \([\frac{\pi}{2}, \frac{3\pi}{2}]\).

Reference


Data File “FR.DAT”

```plaintext
; fr.dat
title
   Root of a function in an interval
cf root.fis
def func
   func = tan(c_x0) - c_x0end
def itis_root
   ; find the root of function func
   ; in the interval ] pi/2, 3pi/2 [ 
   ; with accuracy tol
   c_x1 = 1.01 * pi/2.
c_x2 = 0.99 * 3.*pi/2.
tol = 1.e-4
plot_func
   root = froot
   c_x0 = root
   xtable(2,1) = root
   ytable(2,1) = funcend
plot_func
   ; calculate func at 20 points in the interval and
   ; store the values in table 1
   dx = (c_x2 - c_x1)/20.
c_x0 = c_x1
loop ii (1,20)
   c_x0 = c_x0 + dx
   xtable(1,ii) = c_x0
```

FLAC Version 6.0
JOB TITLE: ROOT OF A FUNCTION IN AN INTERVAL

**FLAC (Version 6.00)**

**LEGEND**
- 7-May-08 16:09
- Step 0
- Table Plot
- Table 2
- Table 1

---

Itasca Consulting Group, Inc.
Minneapolis, Minnesota USA

**Figure 1** Function and its root in an interval
Adapting the Mohr-Coulomb Model to a Hoek-Brown Failure Surface

The FISH routines in “HOEK.FIS” adapt the Mohr-Coulomb model in FLAC to approximate the nonlinear failure surface for a Hoek-Brown material (Hoek and Brown 1982). The Hoek-Brown failure criterion is based on a nonlinear relation between major and minor principal stresses, $\sigma_1$ and $\sigma_3$:

$$\sigma_1 = \sigma_3 - \sqrt{-\sigma_3 \sigma_c m + \sigma_c^2 s}$$

(1)

where $\sigma_c$ is the unconfined compressive strength of the intact rock, $m$ and $s$ are material constants of the rock mass, and compressive stresses are negative (FLAC convention).

For a given value of $\sigma_3$, a tangent to the function (Eq. (1)) will represent an equivalent Mohr-Coulomb yield criterion in the form

$$\sigma_1 = N_\phi \sigma_3 - \sigma_c^M$$

(2)

where $N_\phi = \frac{1+\sin \phi}{1-\sin \phi} = \tan^2 \left(\frac{\phi}{2} + 45^\circ\right)$.

By substitution, $\sigma_c^M$ is

$$\sigma_c^M = \sigma_1 + \sigma_3 N_\phi$$

$$= -\sigma_3 + \sqrt{-\sigma_3 \sigma_c m + \sigma_c^2 s} \cdot -\sigma_3 N_\phi$$

$$= -\sigma_3 \left(1 - N_\phi\right) + \sqrt{-\sigma_3 \sigma_c m + \sigma_c^2 s}$$

(3)

$\sigma_c^M$ is the apparent uniaxial compressive strength of the rock mass for that value of $\sigma_3$.

The tangent to the function (1) is defined by

$$N_\phi(\sigma_3) = \frac{\partial \sigma_1}{\partial \sigma_3} = 1 + \frac{\sigma_c m}{2 \sqrt{-\sigma_3 \cdot \sigma_c m + s \cdot \sigma_c^2}}$$

(4)

The cohesion ($c$) and friction angle ($\phi$) can then be obtained from $N_\phi$ and $\sigma_c^M$. 

---

FLAC Version 6.0
\[
\phi = 2 \tan^{-1} \sqrt{N_\phi} - 90^\circ 
\]

\[
c = \frac{\sigma_c^M}{2\sqrt{N_\phi}} 
\]

The comparison of the Mohr-Coulomb linear approximation to the Hoek-Brown yield surface is shown in the figure. These equivalent \( c \) and \( \phi \) are a good approximation of the nonlinear yield surface for values of the minor principal stress that are close to the given \( \sigma_3 \). The FISH function \text{cfi} \text{ calculates the value of } \( c \) \text{ and } \( \phi \) \text{ for each zone every } \text{ns} \text{ steps. Thus, as } \sigma_3 \text{ changes, the values of } \( c \) \text{ and } \( \phi \) \text{ will also change. It is noted that the instantaneous values of } \( c \) \text{ and } \( \phi \) \text{ calculated in this way closely match those calculated using Hoek’s (1990) expressions based on normal and shear stress.}

Hoek and Brown (1982) also define constants \( m_r \) and \( s_r \) for properties of a broken rock mass. If failure occurs, \( m \) and \( s \) are changed to \( m_r \) and \( s_r \) to represent sudden post-failure response. A progressive strain-softening behavior could be modeled by replacing the Mohr-Coulomb model with the strain-softening model.

The Hoek-Brown parameters, \( \sigma_c, m, s, m_r \) and \( s_r \) are set in “HOEK.FIS” via the variables \text{hb}_\text{sc}, \text{hb}_\text{mmi}, \text{hb}_\text{ssi}, \text{hb}_\text{mmr} \text{ and } \text{hb}_\text{ssr} \text{, respectively, through the SET command. The FISH function } \text{cfi} \text{ is called to update cohesion, friction and tension variables in the Mohr-Coulomb model. The dilation angle may be specified using the variable } \text{hoek}_\text{psi} \text{ (use } \text{hoek}_\text{psi} = \_fi \text{ for an associated flow rule — see example below). Note that, if } \sigma_3 \text{ becomes tensile, the yield surface remains linear with the slope } N_\phi (\sigma_3) \text{ defined at } \sigma_3 = 0.

The user controls the update process by specifying \text{ns} \text{ and } \text{nsup} \text{ through the SET command. } \text{ns} \text{ defines the number of steps taken before } \text{cfi} \text{ is called to update properties. } \text{nsup} \text{ defines the total number of times } \text{cfi} \text{ is to be called. } \text{ns} \times \text{nsup} \text{ corresponds to the total number of steps in the FLAC run. If not specified, the default for } \text{ns} \text{ is 5. This may require variation depending on the nonlinearity of the failure surface.}

A triaxial compression test on a Hoek-Brown material sample is provided below as an example application of this routine. The test is strain-controlled, and an associated flow rule is selected, for the numerical simulation.

References


Figure 1  Linear approximation of the Hoek-Brown failure criterion

Data File “HOEK.DAT”

new
title
  Triaxial test on a Hoek-Brown material
config axi
grid 5 11
gen 0,0 0,2.2 0.5,2.2 0.5,0

model mohr
prop dens=2.0e-3 bulk=666.67 shear=400.0 tension 1e10

fix y j=1
fix y j=12

ini sxx -5 syy -5 szz -5
apply sxx -5 i=6
ini yvel 1.25e-5 var 0 -2.5e-5

his syy i 3 j 6
his sxx i 3 j 6
his szz i 3 j 6
his ydisp i=1 j=12
his xdisp i=6 j=6
set mess off echo off
call hoek.fis
def hoek_psi
    hoek_psi = fi ; associated flow rule
end
;set hb_mmi=1.0 hb_mmr=1.0
;set hb_ssi=0.00753 hb ssr=0.00753
;set hb_sc=50.0
;set nsup=400 ns=10 ; note, FLAC will cycle nsup*ns times
supsolve
plot hold his 1 2 3 vs -4
plot hold bou disp
set ucs=hb_sc hbs=hb ssr hbm=hb_mmr
plot hold fail hoek
return

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Stresses versus top vertical displacement}
\end{figure}
**Figure 3**  Displacement vectors at end of simulation

**Figure 4**  Hoek-Brown failure criterion and final stress state
Quarter-Symmetry Radial Mesh

The *FISH* file “HOLE.FIS” can be called to create a radial mesh in which each gridpoint is defined by polar coordinates *alfa* and *ro*. The variable *rmaxit* is the maximum distance from the center of the grid for each *alfa*. The grid can be adjusted by setting the variables *rmin* (radius of interior hole), *rmul* (number of hole radii to the outer boundary) and *gratio* (grid-spacing ratio) through the *SET* command. The data file “HOLE.DAT” illustrates the use of this *FISH* function. It is quite easy to use “HOLE.FIS” to evaluate the influence of boundary location on the analysis. Only *rmin* and *rmul* need to be changed with the *SET* command.

**Data File “HOLE.DAT”**

```plaintext
grid 10 10
model
call hole.fis
set rmin 1 rmul 10 gratio 1.1
hole
pl grid hold
ret
```

*Figure 1 Quarter-symmetry hole mesh*
Elastic, Hyperbolic Constitutive Model

In order to demonstrate the use of FISH to construct a constitutive model, consider an elastic, nonlinear law of the (hyperbolic) form:

\[ \sigma_d = \frac{\epsilon_1}{E_i + \frac{\epsilon_1}{Y}} \]  

where:
\[ \sigma_d = |\sigma_1 - \sigma_3|; \]
\[ \epsilon_1 = \text{axial strain}; \]
\[ Y = \text{maximum value of } |\sigma_1 - \sigma_3|; \text{ and} \]
\[ E_i = \text{initial Young's modulus (at } \sigma_d = 0). \]

The equation can be differentiated to obtain the slope of the stress/strain curve:

\[ \frac{d\sigma_d}{d\epsilon_1} = E = \frac{E_i (Y - \sigma_d)^2}{Y^2}, \]  

where \( E \) is the tangent Young's modulus.

It can be seen that \( E \) decreases as the stress difference \( \sigma_d \) approaches the yield limit, \( Y \). Since the shear modulus is directly proportional to \( E \) for a constant bulk modulus, the material “fails” in shear as the limit is approached.

Assume that \( E_i \) and \( K \) (the bulk modulus) are given, and that \( K \) remains constant while \( E \) decreases with increasing stress level, \( \sigma_d \). A FISH constitutive model can be written with input properties:

- \texttt{b.mod} \hspace{1cm} \text{K = bulk modulus}
- \texttt{y.initial} \hspace{1cm} \text{E} = \text{initial Young's modulus}
- \texttt{yield} \hspace{1cm} \text{Y = } (\sigma_1 - \sigma_3)_{\text{max}}

These names are used just like built-in property names. The constitutive model is given in “HYPFIS.” The use of special statements like \texttt{f.prop} and local variables like \texttt{zs11} is fully explained in Section 2.8. There are a number of rules to be followed when writing a new constitutive model but, as the example demonstrates, the coding should not be too difficult for a user with some programming experience.

Other (internal) variables are calculated as required. The FISH function \texttt{hyper} is intentionally simple: it does not contain any logic to address unloading behavior, frictional effects and stress-dependent moduli. These could easily be added, but the objective here is to demonstrate how FISH can be employed by the FLAC user to construct a constitutive model.

Once the model has been input (via a data file), it may be used just like any other model. In the following example, the new model \texttt{hyper} is used to obtain the collapse load of a footing (the Prandtl’s wedge problem).
The resulting load/displacement curve is shown in Figure 1, and the velocity field is shown in Figure 2. The response is similar to that produced using the plasticity model \textit{mohr}, but there are differences: (a) the collapse load is only reached asymptotically; and (b) the flow field is much more localized around the footing edge.

\textit{Data File ‘HYP.DAT’}

```plaintext
call hyp.fis ; get user-written law
grid 15 10
model hyper
prop dens=1000  b_mod=1e8
prop  y_initial=2e8  yield=2e5
fix x  i=1
fix x y i=16
fix x y j=1
fix x y i=1,4  j=11
ini yvel -5e-5  i 1,4  j=11
def load ;measure load on platen
  sum = 0.0
  loop i (1,4)
    sum = sum + yforce(i,jgp)
  end_loop
  load = sum / 3.5
  disp = -ydisp(1,jgp)
  anal = (2.0 + pi) * 1e5
end
his load
his anal
his disp
step 10000
plot hold his 1 2 vs 3
plot hold bou vel
```
Figure 1  Load versus displacement — comparison to Prandtl’s wedge solution

Figure 2  Velocity field beneath footing
Initial Equilibrium Stress Distribution in Groundwater Problems for Horizontally Layered Media (No Voids)

In order to establish an initial stress equilibrium in a groundwater problem, several **INITIAL** commands must be issued, bearing in mind the following.

1. Saturation should be zero above the water table, and 1 below it.
2. The pore pressure should be zero above the water table, and have a gradient of $\rho_w g$ below it.
3. The gradient of the total vertical stress should be $g (\rho + \text{sat} \cdot \text{porosity} \cdot \rho_w)$.
4. If $k_{0x}$ is the ratio of effective $\sigma_{xx}$ to effective $\sigma_{yy}$, then the total $\sigma_{xx}$ should be $\sigma_{xx} = k_{0x} (\sigma_{yy} + pp) - pp,$.*
5. If $k_{0z}$ is the ratio of effective $\sigma_{zz}$ to effective $\sigma_{yy}$, then the total $\sigma_{zz}$ should be $\sigma_{zz} = k_{0z} (\sigma_{yy} + pp) - pp$.

This task is very tedious if many different materials are involved or if porosity varies with depth. For horizontally layered materials in which the grid is such that $j$-rows are horizontal, the **FISH** function “ININV.FIS” will perform all the tasks listed above.

The following three parameters must be set:

- $k_{0x}$ ratio of effective $\sigma_{xx}$ to effective $\sigma_{yy}$
- $k_{0z}$ ratio of effective $\sigma_{zz}$ to effective $\sigma_{yy}$
- $w_{gw}$ groundwater table height (it is assumed that the water table is initially horizontal)

The grid must be configured for $gw$.

**NOTE:** “ININV.FIS” requires that the model contains horizontal layers and that no voids exist. Use the **FISH** function “INIV.FIS” for a model containing non-horizontal layers and voids.

* Remember that compressive stresses are negative, and pore pressure is positive, in **FLAC**.
Data File “ININV.DAT”

    config gw ex=5
g 10 10
mo el
pro bulk 3e8 she 1e8 den 2000 por .4
pro den 2300 por .3 j 3 5
pro den 2500 por .2 j 1 2
pro perm 1e-9
water bulk 2e9 den 1000
set g=9.8

set echo off
cp ininv.fis
set wth=8
set k0x=.7
set k0z=.8
ininv
set echo on

fix x i 1
fix x i 11
fix y j 1
step 1
; plot phreatic surface ... assumes that EX_5 is available
def ps
    loop i (1,igp)
        loop j (1,jgp)
            ex_5(i,j) = max(sat(i,j),0.001)
            ex_5(i,j) = min(ex_5(i,j),0.999)
        end_loop
    end_loop
end
ps
; plot yy-effective stress contours interpolated to gridpoints
; (see EXTRAP.FIS)
def fill_ex1_esyy
    loop i (1,izones)
        loop j (1,jzones)
            ex_1(i,j) = syy(i,j) + pp(i,j)
        endloop
    endloop
end

set echo off
cp extrap.fis
fill_ex1_esyy
set gp_avg=1
eextrap_to_gp
set echo on

pl hol ex_2 fi al 'YY-effective stress' ex_5 in=0.5 lm al 'water table'
ret

Figure 1  Initial effective stresses and phreatic surface
Initializing Equilibrium Stress Distribution in Groundwater Problems for Media with Voids

The FISH function “INIV.FIS” initializes stresses and pore pressures as a function of depth, taking into account the presence of voids in the model. It is assumed that line \( j = j_{gp} \) is the free surface, and that stresses depend on the vertical distance below this. Pore pressures are set to vary linearly from a free ground surface; a given free-water surface is not recognized. The grid must be configured for \( gw \).

One input parameter must be set:

\[ k_0 \quad \text{ratio of effective horizontal stress to effective vertical stress} \]

The following example illustrates the initialization of stresses in a model with a surface excavation. Note that there still is an unbalanced force as a result of the excavation. However, the stress state is close to equilibrium.

Data File “INIV.DAT”

```plaintext
config gw ex=4
g 10 10
mo e
pro bulk 3e8 she 1e8 den 2000 por .4
pro den 2300 por .3 j 3 5
pro den 2500 por .2 j 1 2
pro perm 1e-9
mo null i=1,3 j=8,10
water bulk 2e9 den 1000
set g=9.8
c a invfis
set k0=0.7
i_stress
fix x i 1
fix x i 11
fix y j 1
hist unbal
set flow off
step 1
plot bou estr hold
save invsav
solve
plot bou estress hold
ret
```

FLAC Version 6.0
Figure 1  Effective stresses at step 1

Figure 2  Effective stresses at equilibrium
Finding the Integral of a FLAC Table

The FISH file “INT.FIS” integrates the values of a table, and returns another table. The ID number of the input table is the first argument, and the output table is the second argument. If the new output table already exists, all data points in it will be deleted and overwritten with the new values. If the output table does not exist, one is created.

The function integrates using the trapezoidal rule, with the same number of points in the result tables as are in the source tables. The function assumes an integration constant of zero.

The figure shows the result of a simple cosine wave integration. Table 1 is the input data, and table 2 is the output data. The input data is read using the `cr_tab` function to create the data and copy it into a table.

Data File “INT.DAT”

```
new
title Example of INTEGRATE FISH function'
grid 1,1
;
def cr_tab
    local val = pi * 6.e-3
    local ii
    loop ii (1,1000)
        local xx = float(ii-1) * val
        xtable(1,ii) = xx
        ytable(1,ii) = cos(xx)
    end_loop
end
@cr_tab
;
ca int.fis suppress
;
@integrate(1,2)
plot hold table 1 line 2 line
```
Figure 1  Simple cosine wave integration
Matrix Inversion via LU-Decomposition

A pair of FISH functions, `ludcmp` and `lubksb`, can be used to solve the system of equations

\[ Ax = b \] (1)

where \( A \) is a given square matrix of size \( n \), \( b \) is a given vector of size \( n \), and \( x \) is the desired solution vector of size \( n \). The FISH function `ludcmp` performs an LU-decomposition on the matrix \( A \). If \( A \) is found to be singular, then the FLAC error-handling mechanism is invoked, and an error message is printed. The FISH function `lubksb` performs the back-substitution operation to produce the solution vector \( x \). Both of these functions implement the algorithm described in Press et al. (1986).

An example problem demonstrating the use of the two FISH functions is provided in the data file below. The size of the matrix is passed as an argument to `make_random_data`.

Reference


Data File “LUDA.DAT”

```fis
new ; Test LU-decomposition FISH functions on random matrices.
;
new
title Test LU-decomposition FISH functions on random matrices.
call luda.fis suppress ; support functions for LU-decomposition
;
def make_random_data(lu.nn)
  local aa = get_array(lu.nn,lu.nn)
  local indx = get_array(lu.nn)
  local bb = get_array(lu.nn)
  local xx = get_array(lu.nn)
  --- fill aa for test
  local i
  local j
  loop i (1,lu.nn)
    loop j (1,lu.nn)
      aa(i,j) = urand
    end_loop
  end_loop
  --- set "unknowns"
  loop i (1,lu.nn)
    xx(i) = float(i)
end_def
```
xtable(1,i) = float(i)
ytable(1,i) = xx(i)
end_loop

;--- multiply by matrix to get r.h.s.
loop i (1,lu_nn)
  local sum = 0.0
  loop j (1,lu_nn)
    sum = sum + aa(i,j) * xx(j)
  end_loop
  bb(i) = sum

@make_random_data(8); Righthand-sides ...

ludcmp
lubksb
look

;   compare ‘unknowns’
plot hold table 1 line 2 alias ‘1:as set 2:as calculated’
return

---

Figure 1 Comparison of “unknowns”

---
Strength/Stress Ratios for Different Mohr-Coulomb Materials

The command `PLOT mohr` produces a contour plot of strength/stress ratio for zones in a FLAC model based upon a Mohr-Coulomb failure criterion. The corresponding strength properties are specified using the commands `SET pltc`, `SET pltf` and `SET pltt`, and only one set of properties is accepted for the whole FLAC model. The FISH function `mcfos` plays the same role as the `PLOT mohr` command, but allows for the use of different sets of strength properties. In this function, the factor of safety is calculated based on direct internal access of the strength properties as specified using the `PROPERTY` command (these properties are assumed to be available).

The grid should be configured for at least four extra arrays (`CONFIG extra=4`) when using `mcfos`. The function is called after the model is brought to equilibrium. The FISH function “PS3D.FIS” is used to compute the three principal stresses for each zone and to store these values in the extra arrays, `ex_1`, `ex_2` and `ex_3`. The strength/stress ratios are then calculated using (Eq. (3.31) in Section 3 in the User’s Guide), stored in the extra array `ex_4`, and a contour plot is displayed.

The example data file “MCFOS.DAT” illustrates the use of “MCFOS.FIS” to display contours of the strength/stress ratio in a compression test on a Mohr-Coulomb sample containing an inclusion.

Data File “MCFOS.DAT”

```plaintext
new
title
  Compression test on Mohr-Coulomb material with inclusion
config axi ex 4
grid 3 5
; --- model and properties ---
model mohr
prop den 2000 bulk 1.2e10 she 1.1e10 fric 44 ten 2e5
prop coh 2.72e5
prop coh 1e5 i=1,2 j=1,2 ; assign a low cohesion to the inclusion
; --- boundary conditions ---
fix y j=1
fix y j=6
ini yv -1e-7 j=6
; --- histories ---
hist syy i=1 j=1
hist syy i=1 j=4
hist ydisp i=1 j=6
step 1000
ini yv 0 j=6
solve sratio 1e-3
; --- factor of safety ---
ca mcfos.fis
plot hold bou plas
ret
```
JOB TITLE: STRENGTH/STRESS RATIO FOR DIFFERENT MC MATERIALS

**FLAC (Version 6.00)**

**Legend**
- 30-Jun-08 10:05
- step 1035
- 
- Strength/stress ratio
  - 1.00E+00
  - 2.00E+00
- Contour interval = 1.00E+00

**Figure 1** Strength/stress ratio contours

**FLAC (Version 6.00)**

**Legend**
- 30-Jun-08 10:05
- step 1035
- 
- Boundary plot
  - 0
  - 1E 0
- Plasticity Indicator
  - X elastic, at yield in past

**Figure 2** Plasticity indicators

---

**FLAC Version 6.0**
Hyperbolic Duncan-Chang Constitutive Model

The file “MDUNCAN.FIS” contains a FISH version of the Duncan constitutive model (Duncan et al. 1980). The inelastic model formulation is similar to that implemented in the finite element code SOILSTRUCT (Filz et al. 1990). Also see “HYP.FIS,” for the description of an elastic, hyperbolic constitutive model.

The Duncan hyperbolic soil model is based on the incremental form of Hooke’s law:

\[
\begin{align*}
\Delta \sigma_{11} &= \alpha_1 \Delta \epsilon_{11} + \alpha_2 (\Delta \epsilon_{22} + \Delta \epsilon_{33}) \\
\Delta \sigma_{22} &= \alpha_1 \Delta \epsilon_{22} + \alpha_2 (\Delta \epsilon_{11} + \Delta \epsilon_{33}) \\
\Delta \sigma_{33} &= \alpha_1 \Delta \epsilon_{33} + \alpha_2 (\Delta \epsilon_{11} + \Delta \epsilon_{22}) \\
\Delta \sigma_{12} &= 2G \Delta \epsilon_{12}
\end{align*}
\]

where \( \alpha_1 = K + 4/3G \) and \( \alpha_2 = K - 2/3G \), \( K \) is the bulk modulus and \( G \) is the shear modulus. In these equations, \( G = 3KE/(9K - E) \) and Young’s modulus, \( E \), is a nonlinear function of stresses that varies depending on primary loading or unloading/reloading conditions.

In this version of the model, loading conditions are defined based on the maximum value of a stress state function, \( S_S \), expressed in terms of stress level \( S_I \) and minimum compressive stress \( -\sigma_3 \). (By convention, tensile stresses are positive, and principal stresses are ordered such that \( \sigma_1 \leq \sigma_2 \leq \sigma_3 \).) The stress level is defined in the framework of the Mohr-Coulomb criterion by the relation

\[
S_I = \frac{\sigma_1 - \sigma_3}{(\sigma_1 - \sigma_3) f}
\]

where \( (\sigma_1 - \sigma_3) f \) is the stress difference at failure, given in terms of cohesion, \( c \), and friction, \( \phi \), by

\[
(\sigma_1 - \sigma_3) f = \sigma_3 (N_\phi - 1) - 2c \sqrt{N_\phi}
\]

where \( N_\phi = (1 + \sin \phi)/(1 - \sin \phi) \).

In turn, the stress state function has the form

\[
S_S = S_I \left( \frac{-\sigma_3 + c/\tan \phi}{P_a} \right)^{1/4}
\]

where \( P_a \) is atmospheric pressure, and the frictional term is omitted for \( \phi = 0 \).
Primary loading is assumed to take place when the current value of the stress state function in a zone is greater than the maximum value reached at that location in the past. Unloading/reloading conditions are assumed to hold otherwise. For primary loading, Young’s modulus has the form

$$E = \left[1 - R_f S_l\right]^2 K_{pl} P_a \left(\frac{\max(-\sigma_3, 0)}{P_a}\right)^n$$

(5)

where the failure ratio $R_f$, the modulus exponent $n$, and the modulus number $K_{pl}$ are three model parameters. (The value of $R_f$ is always smaller than unity, and varies from 0.5 to 0.9 for most soils.) For unloading/reloading, $E$ obeys the law

$$E = K_{ur} P_a \left(\frac{\max(-\sigma_3, 0)}{P_a}\right)^n$$

(6)

and the parameter $K_{ur}$ is the unloading/reloading modulus number. ($K_{ur}$ is always larger than $K_{pl}$.)

There are two options in the model. In the first one, Poisson’s ratio, $\nu$, is given and assumed to remain constant. Current values of tangent bulk and shear moduli are then evaluated from $E$ and $\nu$, using the relations

$$K = \frac{E}{3(1-2\nu)}$$

$$G = \frac{E}{2(1+\nu)}$$

(7)

In the second option, the tangent bulk modulus, $K$, varies with $\sigma_3$, according to the law

$$K = K_b P_a \left(\frac{\max(-\sigma_3, 0)}{P_a}\right)^m$$

(8)

where the parameter $K_b$ is the bulk modulus number and $m$ is the bulk modulus exponent. (For most soils, $m$ varies between 0 and 1.) In the FISH constitutive model, values of $K$ are restricted to the interval $[\frac{E}{3}, 17E]$; this corresponds to variations of Poisson’s ratio between 0 and 0.49.
The constitutive model function is named \texttt{m_duncan} and requires that the following parameters be specified with the \texttt{PROPERTY} command:

- \texttt{d_bulk} initial tangent bulk modulus (also gives access to current value)
- \texttt{d_coh} cohesion, $c$
- \texttt{d_fric} friction angle, $\phi$ (degree)
- \texttt{d_gmax} maximum shear modulus in simulation (used for stability)
- \texttt{d_k} modulus number, $K_p$
- \texttt{d_kb} bulk modulus number, $K_b$ (0 for constant $\nu$)
- \texttt{d_kmax} maximum bulk modulus in simulation (used for stability)
- \texttt{d_ku} unloading/reloading modulus number, $K_{ur}$
- \texttt{d_m} bulk modulus exponent, $m$ (not needed if $d_kb = 0$)
- \texttt{d_n} modulus exponent, $n$
- \texttt{d_nu} constant Poisson's ratio value ($d_kb$ must be zero)
- \texttt{d_pa} atmospheric pressure, $P_a$
- \texttt{d_rf} failure ratio, $R_f$
- \texttt{d_shear} initial tangent shear modulus (also gives access to current value)
- \texttt{d_sssmax} maximum past value of the stress state function

These parameters default to zero if not specified.

A \texttt{FISH} function, \texttt{ini_duncan}, is provided in “MDUNCAN.FIS” to automatically initialize the values of tangent bulk and shear moduli and stress state function based on the initial stress state. The initial Young’s modulus is then calculated using the unloading/reloading formula. The maximum values of bulk and shear modulus are also initialized by the function \texttt{ini_duncan}, based on the value of the variable \texttt{d_ms3}. This variable should be set to an estimate of the maximum value of $-\sigma_3$ reached during the simulation.

In the following example, the model \texttt{m_duncan} is used to model the results of a triaxial test involving several unloading/reloading excursions. The resulting load/displacement curve is shown in Figure 1.
Data File “MDUNCAN.DAT”

; mduncan.dat
config axi
s 5 20
g 0 0 0 1 .25 1 .25 0
c a mduncan.fis
; --- model and properties ---
mo m_duncan
prop den .00202
prop d_pa=1.0584 d_k=700 d_n=0.37 d_rf=0.8 d_ku=1820
prop d_kb=280 d_m=0.19
prop d_coh=0.31 d_fric=33
; --- initialisation ---
ini sxx -2 syy -2 szz -2
set d_ms3=2.
ini_duncan
; --- boundary conditions ---
fix y j=1
fix y j=21
ini yv -0.5e-6 j=21
ini yv 0.5e-6 j=1
apply pres 2 i=6
; --- fish functions ---
def s1_s3
  .area=pi*x(igp,jgp)^2
  sum=yforce(1,jgp)*x(2,jgp)*0.25
  loop i (2,igp)
    sum=sum+yforce(i,jgp)*x(i,jgp)
  end loop
  sigmav=2*pi*sum/_area
  s1_s3=sigmav-2.0
  s1_s3_ult=7.4076
  trans_s1_s3=ev/(sigmav-2.0)
end
def ev
  ev=(ydisp(3,1)-ydisp(3,21))/(y(3,21)-y(3,1))
end
; --- histories ---
hist nstep 20
hist s1_s3
hist s1_s3_ult
hist trans_s1_s3
hist ev
hist unbal
hist syy i=3 j=10
; --- test ---
step 4000
save dun0.sav
ini yvel mul -0.5
step 800
save dun1.sav
ini yvel mul -1
step 2000
save dun2.sav
ini yvel mul -1
step 800
save dun3.sav
ini yvel mul -1
step 2000
save dun4.sav
ini yvel mul -1
step 800
save dun5.sav
ini yvel mul -1
step 2000
save dun6.sav
plot hold his 1 2 vs 4
save dun.sav
;
in yvel mul 2.
step 150000
save duna.sav

References


**Figure 1**  Stress-strain curve for hyperbolic Duncan-Chang model
Computing and Comparing Modulus Reduction and Damping Ratio Curves

The FISH file “MODRED.FIS” computes the modulus reduction factor and damping ratio in a one-zone model at each shear strain level stored in an input table, and stores the calculated values to two output tables. The ID number of the input modulus reduction and damping ratio tables, which are usually calibrated in laboratory tests, are defined by \_glab (modulus reduction factor versus shear strain magnitude) and \_dlab (damping ratio versus shear strain magnitude). The output tables are specified by \_gnum (numerically calculated modulus reduction factor versus shear strain magnitude) and \_dnum (numerically calculated damping ratio versus shear strain magnitude).

Figures 1 and 2 compare the modulus reduction and damping ratio curves predicted by hysteretic damping model \textbf{sig3} (see Section 1.4.3.5 in Dynamic Analysis) to the laboratory measurements in “MODRED.DAT.”

Data File “MODRED.DAT”

```plaintext
new
conf dy
gri 1 1
model elastic
prop density 2000 bulk 1e8 shear 3e7
fix x y
ini hyst sig3 1.014 -0.4792 -1.249
;
call modred.fis
; input soil’s dynamic properties measured in laboratory tests
; modulus reduction curve
table 101 0.0001,1 0.0003,1 0.001,0.99 0.003,0.96 0.01,0.85 0.03,0.64 &
  0.1,0.37 0.3,0.18 1,0.08 3,0.05 10,0.035
;
damping ratio curve
table 102 0.0001,0.24 0.0003,0.42 0.001,0.8 0.003,1.4 0.01,2.8 0.03,5.1 &
  0.1,9.8 0.3,15.5 1,21 3,25 10,28
;
def _settab
 ; laboratory data
  _glab = 101
  _dlab = 102
 ; numerical (FLAC) results
  _gnum = 111
  _dnum = 112
end
_settab
_modred
;
; convert x-set data of lab data in log
set tab_no _glab
log10_tabx
set tab_no _dlab
```
log10_tabx
;
label table _gnum
FLAC
label table _glab
Laboratory
title
   Modulus reduction curve - lab vs FLAC
plot hold table _glab both _gnum both
;
label table _dnum
FLAC
label table _dlab
Laboratory
title
   Damping ratio curve - lab vs FLAC
plot hold table _dlab both _dnum both
save modred.sav

Figure 1  Modulus reduction curves from the FLAC computation and the laboratory test
**FIGURE 2**  Damping ratio curves from the FLAC computation and the laboratory test.
Mohr-Coulomb FISH Model

The file “MOHR.FIS” contains a FISH function which replicates the built-in Mohr-Coulomb plasticity model in FLAC. A detailed explanation of the model is provided in Section 2.4.2 in Theory and Background. The function is named m_mohr and requires that the following parameters be specified with the PROPERTY command:

- **m_coh**: cohesion
- **m_dil**: dilation angle
- **m_fric**: friction angle
- **m_g**: shear modulus
- **m_k**: bulk modulus
- **m_ten**: tensile strength

These parameters default to zero if not specified. The user also has access to

- **m_ind**: state indicator:
  - 0: elastic
  - 1: plastic shear
  - 2: elastic now, but plastic in past
  - 3: plastic tensile

The following problem compares the FISH model to the built-in Mohr-Coulomb model. The built-in model is used for zones in the left half of the model. The FISH function is used for zones in the right half of the model.

**Data File “MOHR.DAT”**

```plaintext
   g 12 10
   gen 0,0 0,25 30,25 30,0

   model mohr i = 1,6
   prop den 2500 bulk 1.19e10 shear 1.1e10 i=1,6
   prop coh 2.72e6 fric 44 ten 2e6 i=1,6

   call mohr.fis
   model m_mohr i = 7,12
   prop den 2500 m_k 1.19e10 m_g 1.1e10 i=7,12
   prop m_coh 2.72e6 m_fric 44 m_ten 2e6 i=7,12

   ini xv 1e-6 i=1
```
MOHR.FIS - 2

ini xv -1e-6 i=13
fix x y i=1
fix x y i=13
his nstep 100
his unbal
his xdisp i=1 j=1
his sxx i=6 j=5
his sxx i=7 j=5
step 15000
save mohr.sav
plot hol bou est disp
return

Figure 1  Comparison of stresses and displacements
Printing a Floating-Point Number with User-Specified Precision

The *FISH* file “NUMBER.FIS” is used to output a floating point number with a user-specified precision up to a maximum precision limit of 16 digits. Normally a *FISH* function outputs the result of a floating-point operation with a precision limit of 5 digits.

The algorithm used to extract digits up to the precision required is outlined:

(a.) Convert number to a float value (type cast).

(b.) Determine “k,” the exponent of the resulting number.

\[ k = \begin{cases} 
\lfloor \log(\text{number}) - 1 \rfloor & \text{if } \log(\text{number}) < 0.0 \\
\lfloor \log(\text{number}) \rfloor & \text{otherwise} 
\end{cases} \]

where \( \lfloor .. \rfloor \) denotes the integral value of number within brackets.

(c.) Extract each significant digit up to a precision of precision limit + 1 and store it in an array.

After all digits are found up to a precision limit of “precision limit + 1,” the last digit is rounded off depending on whether its value is greater than or equal to 5. Thus the result will be a floating-point number with user-defined precision stored as a string. The digits are then output as a string value with an exponent of “k.”

Note: If the user does not specify a precision limit, the *FISH* function assumes a precision limit of 7 digits.

The data file “NUMTEST.DAT” illustrates how to print numbers with a user-specified precision limit of 10 digits. In this example, 20 numbers are generated randomly and their values are output with 10 digits of precision.

*Data File “NUMTEST.DAT”*

```plaintext
; Exercise the Number functions
call number.fis
set digits=10
def qqq
  loop n (1,20)
    power = int((urand - 0.5) * 40.0)
    Given = urand^power
    oo = out(' input = '+string(Given)+' output = '+Number)
  endLoop
end
qqq
```

*FLAC Version 6.0*
P-Q Stress Diagram

Often, the user may wish to print or plot problem variables that are not directly accessible through the FLAC HISTORY command. It is quite simple for the user to write a FISH function which will calculate the desired variable directly in FLAC.

The data file “PQ.DAT” illustrates the use of FISH to calculate the stress point \( p,q \) and plot a \( p-q \) diagram via the HISTORY and PLOT commands. The generalized stress components \( p \) and \( q \) are expressed in terms of principal stresses,

\[
p = -\frac{1}{3}(\sigma'_1 + \sigma'_2 + \sigma'_3)
\]
\[
q = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_1 - \sigma_3)^2}
\] 

(1)

Note that \( p \) is an effective pressure, defined in terms of the effective principal stresses.

Data File “PQ.DAT”

```
g 5 5
mo el
    call pq.fis
    set iv=3 jv=3
    pro bulk 2e8 she 1e8 den 2000
    fix x y j 1
    fix x i 1
    fix x i 6
    ini yv -1e-2 j 6
    fix y j 6
    hist qs
    hist ps
    step 100
    ini yv 0 j 6
    ini xv -1e-2 i 6
    step 300
    plot hold his 1 vs 2
    ret
```
Figure 1 p-q plot
Printing Selected Structural-Element Variables

The user can select specific variables to be printed from a FLAC model even though they may not be directly available from the PRINT command. This can be done by accessing the data structure in FLAC directly, as described in Section 4. For example, the user may wish to print maximum and minimum stresses associated with structural elements for the case of beams installed on a regular spacing in the out-of-plane direction. The structural element variables stored by FLAC are scaled axial, shear forces and moments. In order to determine the actual forces and moments in the beams, the FLAC forces and moments must be multiplied by the spacing. Actual axial stresses are then derived using actual moment of inertia and area of the beam cross-section. (See Section 1.9.4 in Structural Elements for further discussion on scaling a 2D FLAC model to simulate a 3D problem with regularly spaced structural elements.)

The FISH function “PRSTRUC.FIS” calculates the actual extreme values of axial stresses at the midpoint of beams, assuming a regular spacing defined by b_space, and a cross-sectional height specified by b_height. (Note that the beam formulation in FLAC assumes a linear variation of moment along the beam element.) The function calls the file “STR.FIN” to access values in the offsets associated with the structural-element data structure. The file is contained in the “\FISH\4-ProgramGuide” directory. The actual minimum and maximum axial stresses for each beam element are then printed in a list.

The following data file illustrates the application of PRSTRUC.FIS:

Data File “PRSTRUC.DAT”

```
grid 10,10
m e
prop d=2500 b=3e8 s=2e8
fix y j=1
m n i=5,6 j=5,6
fix x i=1
apply press 1e6 from 1,11 to 11,11
ini sxx -1e6 syy -1e6
fix x i=11
struct beam begin grid 5,5 end grid 5,6
struct beam begin grid 5,6 end grid 5,7
struct beam begin grid 5,7 end grid 6,7
struct beam begin grid 6,7 end grid 7,7
struct beam begin grid 7,7 end grid 7,6
struct beam begin grid 7,6 end grid 7,5
struct beam begin grid 7,5 end grid 6,5
struct beam begin grid 6,5 end grid 5,5
struct prop 1 a 0.25 e 1e9 i 1e-3
hist unbal
hist ydis i 5 j 9
step 500
;```

FLAC Version 6.0
PRSTRUC.FIS - 2

FISH in FLAC

```plaintext
set echo off
call prstruc.fis
set echo on
set b_space 5 b_height=0.5
set log prstruc.log
set log on
prstruc
set log off
```

The printout from this example is shown:

<table>
<thead>
<tr>
<th>ID</th>
<th>F-axial</th>
<th>Str-min</th>
<th>Str-max</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.0921E+06</td>
<td>-4.5272E+06</td>
<td>-4.2100E+06</td>
</tr>
<tr>
<td>7</td>
<td>1.0921E+06</td>
<td>-4.5269E+06</td>
<td>-4.2098E+06</td>
</tr>
<tr>
<td>6</td>
<td>1.3361E+06</td>
<td>-5.4849E+06</td>
<td>-5.2035E+06</td>
</tr>
<tr>
<td>5</td>
<td>1.3492E+06</td>
<td>-5.6054E+06</td>
<td>-5.1883E+06</td>
</tr>
<tr>
<td>4</td>
<td>1.0793E+06</td>
<td>-4.5077E+06</td>
<td>-4.1264E+06</td>
</tr>
<tr>
<td>3</td>
<td>1.0791E+06</td>
<td>-4.5072E+06</td>
<td>-4.1255E+06</td>
</tr>
<tr>
<td>2</td>
<td>1.3495E+06</td>
<td>-5.6065E+06</td>
<td>-5.1893E+06</td>
</tr>
<tr>
<td>1</td>
<td>1.3358E+06</td>
<td>-5.4858E+06</td>
<td>-5.2026E+06</td>
</tr>
</tbody>
</table>

FLAC Version 6.0
Plotting the Phreatic Surface for Groundwater Problem

The FISH function “PS.FIS” can be used to plot a single contour which locates the phreatic surface for a groundwater analysis. The function locates the boundary between saturated and unsaturated gridpoints. The function should be used at steady-state flow and for a grid configured for gw mode.

For example, data file “PS.DAT” restores the save file “H2A.SAV,” created for the steady-state flow problem described in Section 10 in the Verifications volume, and plots the phreatic surface. The result is shown in Figure 1. (Note that CONFIG extra 1 must first be added to data file “FREESURFACE.DAT” in Section 10 in the Verifications volume to generate this plot.)

Data File “PS.DAT”

rest h2a.sav ; restore save file from Verification Problem 10
call ps.fis
ps
plot bound flow ex_1 int 0.5 alias ‘phreatic surface’ lmag hold

Figure 1 Phreatic surface for Section 10 in the Verifications volume
**Computing the 3D Principal Stresses**

Keywords $\text{sig1}$ and $\text{sig2}$ (utilized in printing and plotting) correspond to the two-dimensional major and minor principal stresses, respectively. As stated in Section 1 in the Command Reference, they refer to stresses in the $xy$-plane only. However, the out-of-plane stress $szz$ may be the major or minor principal stress if the full three-dimensional stress tensor is considered.

The FISH function $\text{ps3d}$ computes the three principal stresses, taking into account the out-of-plane stress, and places them in the first three extra arrays, as follows:

- $\text{ex}_1$: major principal stress (most negative; most compressive)
- $\text{ex}_2$: intermediate principal stress
- $\text{ex}_3$: minor principal stress (least negative; least compressive)

When using $\text{ps3d}$, the grid should be configured for at least three extra arrays ($\text{CONFIG extra} = 3$). The word $\text{ps3d}$ should be entered as a command before using the results in the extra arrays. For example,

```fortran
ps3d plot boun ex_1 zon fill alias ‘major principal stress’
```

will plot filled contours of the major (3D) principal stress.

**Data File “PS3D.DAT”**

```plaintext
config extra 3
grid 5 5
model
prop dens 1000 bulk 2e8 sh 1e8
set grav 10
fix x y j 1
step 100
call ps3d.fis
ps3d
plot hold ex_1 zon fill alias ‘major principal stress’
```

FLAC Version 6.0
JOB TITLE: COMPUTING 3D PRINCIPAL STRESSES

FLAC (Version 6.00)

LEGEND
7-May-08 16:09
step 100
-8.330E-01 < x < 5.833E+00
-8.330E-01 < y < 5.833E+00

major principal stress
-4.50E+04
-4.00E+04
-3.50E+04
-3.00E+04
-2.50E+04
-2.00E+04
-1.50E+04
-1.00E+04
-5.00E+03
0.00E+00

Contour interval= 5.00E+03

Figure 1  Major principal stress contours
Quarter-Symmetry Donut-Shaped Mesh

A quarter-symmetry section of the donut-shaped mesh can be generated with “QDONUT.FIS.” This is identical to that for “DONUT.FIS,” except that only one-quarter of the grid is generated. The same variables \( rmin, rmul \) and \( gratio \) are set as with the grids generated in “HOLE.FIS” and “DONUT.FIS.”

Data File “QDONUT.DAT”

\[
g 10 10 \\
m o e l \\
c all qdonut.fis \\
se t rmin 1 rmul 10 gratio 1.1 \\
qdonut \\
pl grid hold \\
ret
\]

**Figure 1** Quarter-symmetry donut-shaped mesh
**Tracking Unbalanced Flow**

Most groundwater problems have an initial transient period in which the flow entering the model differs from the flow leaving the model. The FISH function `qratio` evaluates total `inflow` and `outflow` at fixed pore pressure boundaries of the model, and calculates the ratio between the unbalanced flow `qbalance = inflow - outflow`, and the average flow `(inflow + outflow)/2`. `qratio` should be identical to the internal variable `sratio`, provided all flow exchanges occur at fixed pore pressure boundaries (no applied discharge or wells).

`inflow`, `outflow` and `qbalance` are flow rates for a unit model thickness (with dimension of $[L^3/T]$, cubic meters per second, for example). `qratio` is a dimensionless ratio that ranges from 0 to 2. When steady-state flow is reached, `qratio = 0`, provided the model does not contain applied discharges or wells.

The data file “QRATIO.DAT” illustrates the use of the variables `inflow` and `outflow` to check detection of steady-state flow in a problem of flow through an embankment. See Section 1.8.4.3 in Fluid-Mechanical Interaction for an additional example application.

**Data File “QRATIO.DAT”**

```plaintext
title
Flow through an embankment
config gw extra 1
g 16 8
def ini_h2
  h1 = 4.
  h2 = 1.
  b1 = 8.
  ck = 1e-10
  rw = 1e3
  gr = 10.
end
ini_h2
gen 0 0 0 h1 b1 h1 b1 0
mo el
; --- Properties ---
prop por .3 perm=ck den 2000
water den=rw bulk 1e3
; --- Initial conditions ---
ini sat 0
; --- Boundary conditions ---
ini pp 4e4 var 0 -4e4 i 1
ini pp 1e4 var 0 -1e4 i 17 j 1 3
fix pp i 1
fix pp i 17
ini sat 1 i 1
ini sat 1 i 17 j 1 3
; --- Settings ---
```

**FLAC Version 6.0**
set mech off g=gr
; --- Fish functions ---
ca qratio.fis
; --- Histories ---
hist nstep 50
hist sratio
hist qratio
hist inflow
hist outflow
; --- Step ---
solve sratio 5.e-3
; --- View plots ---
plot hold his 3 4
ret

![HISTORY PLOT](image)

**Figure 1 Evolution of inflow and outflow**
Translation of Grid Region

In Section 4.4 in Theory and Background, the upper region of a grid is translated downward by using a set of \texttt{INI y add} commands. The \texttt{FISH} file “REGADD.FIS” can be used to translate a region in the model automatically by setting the following variables:

\[
\begin{align*}
\text{i\_reg} & : i\text{-index of gridpoint inside region} \\
\text{j\_reg} & : j\text{-index of gridpoint inside region} \\
\text{x\_add} & : x\text{-direction addition to gridpoints within the region} \\
\text{y\_add} & : y\text{-direction addition to gridpoints within the region}
\end{align*}
\]

The function “REGION.FIS” is called to first identify the gridpoints with the region.

The example below is the same as that in Section 4.4 in Theory and Background.

Data File “REGADD.DAT”

```
config extra 1
grid 5 20
model elas
  gen line 0.0,3.0 5.0,14.0
  gen line 0.0,5.0 5.0,16.0
model null region 1,5
ini x 5.0 y 14.0 i 6 j 14
ini x 0.0 y 5.0 i=1 j 8
set echo off
call region.fis
call regadd.fis
set echo on
plot hold grid mark
set i_reg 3 j_reg 18 x_add 0.0 y_add -2.0
reg_add
inter 1 as from 1,4 to 6,14 bs from 1,8 to 6,17
mark j 1,4
mark j 17,21
gen adj
plot hold grid
ret
```
Figure 1    Grid before translation

Figure 2    Grid after translation
Finding Gridpoint Variables inside a Region

The FISH file “REGION.FIS” can be used to find all gridpoints located within a region of the grid. This function is similar to the action of the region keyword, except that the operation is performed on gridpoints and variables associated with gridpoints. See Section 1.1.3 in the Command Reference.

The example data file “REGION.DAT” illustrates the use of “REGION.FIS” to assign saturation values at gridpoints within a selected region. Gridpoints with zero saturation are marked and plotted for the purpose of verification.

Another application translates all gridpoints in a specified region. See the FISH library file “REGADD.FIS.”

Note that the CONFIG extra command must be specified to have one extra variable available. The extra gridpoint variable ex.1 is set to integer value 1 if the gridpoint is inside the region identified by the variables i_reg and j_reg.

Data File “REGION.DAT”

```plaintext
conf gw extra=1
gr 20 20
mod elas
prop d 1 s 1 b 1
gen circle 10 10 5
;
set echo off
call region.fis
set echo on
;
set i_reg=1 j_reg=1
region
def ini_sat
  loop i (1,igp)
    loop j (1,jgp)
      if ex.1(i,j) # 1 then
        sat(i,j) = 0
      end_if
    end_loop
  end_loop
end
ini_sat
; --- mark nodes with zero saturation ---
def mark_to_check
  loop ii (1,igp)
    loop jj (1,jgp)
      if sat(ii,jj) = 0 then
        command
        mark i=ii j=jj
      end_if
    end_loop
  end_loop
end
```

FLAC Version 6.0
end_command
end_if
end_loop
end_loop
end
mark_to_check
plot hold grid mark
ret

Figure 1  Gridpoints with zero saturation
Servo Control

A servo-control function is used to minimize the influence of inertial effects on the response of the model. The FISH file “SERVO.FIS” shows how the applied vertical velocities can be adjusted as a function of the maximum unbalanced force in the model. By preventing the unbalanced force from getting too high (i.e., controlling the inertial effects), the user has better control over model behavior.

The control is specified by setting the upper limit for unbalanced force, high_unbal, and lower limit, low_unbal, with the SET command. The loading velocity is also controlled by specifying an upper limit (high_vel). A command is not issued for this function because it is automatically invoked at every calculation step through the WHILE_STEPPI NG FISH command.

This function is demonstrated for the problem of a triaxial compression test of a strain-softening material (data file “SERVO.DAT”). The stress-strain response of the specimen indicates a weakening of the material after the peak strength is reached. The servo-control of the applied velocity allows for an analysis with minimal inertial effects.

Note that FISH functions are built into the data file to calculate the average vertical stress, sigmav, and average vertical strain, ev, in order to generate the stress-strain plot shown in Figure 1.

The servo-control function will need to be modified for different types of loading.

Data File “SERVO.DAT”

; Triaxial test of strain-softening material
; with controlled velocity
title
    Triaxial test of strain-softening material
config axi
g 5 10
mo ss
call servo.fis
fix y j 1
fix y j 11
ini yvel -2.5e-5 j 11
ini yvel 2.5e-5 j 1
pro den 2500 bulk 2e8 she 1e8 co 2e6 fric 45 ten 1e6 dil 10
pro ftab 1 ctab 2 dtab 3
table 1 0 45 .05 42 .1 40 1 40
table 2 0 2e6 .05 1e6 .1 5e5 1 5e5
table 3 0 10 .05 3 .1 0
app pres 1e6 i 6
ini sxx -1e6 syy -1e6 szz -1e6
def sigmav
    sum=0.0
    loop i (1,igp)
        sum=sum+yforce(i,jgp)
end_loop
    sigmav=sum/(x(igp, jgp)-x(1, jgp))
end

def ev
    ev=(ydisp(3,1)-ydisp(3,11))/(y(3,11)-y(3,1))
end
hist sigmav
hist ev
hist yvi1j1
hist unbal
set high_unbal=5e4
set low_unbal=2e4
set high_vel=1e-4
step 6000
save servo.sav
plot hold his 1 vs 2
plot hold his 4
plot hold his 3
ret

Figure 1 Axial stress versus axial strain for a triaxial test with strain-softening material with controlled velocity
Figure 2  Unbalanced force history for a triaxial test with strain-softening material with controlled velocity

Figure 3  Vertical velocity history for a triaxial test with strain-softening material with controlled velocity
Finding the Response Spectrum of an Acceleration History in a FLAC Table

The FISH file “SPEC.FIS” finds the displacement response spectrum, the pseudo-velocity response spectrum and the pseudo-acceleration response spectrum of an input acceleration stored in a FLAC table. The ID number of the input table is defined by the `acc_in` argument, and the three output tables are identified by the `sd_out`, `sv_out` and `sa_out` arguments. If any of the output tables currently exists, it will be deleted and overwritten by the new results.

The damping constant for the response analysis is specified by the `dmp` argument. The calculation is only approximate for damped responses; the higher `dmp` is, the less accurate the response.

The range of periods over which the spectrum is calculated by the `pmin` and `pmax` arguments, and the number of points in the output tables, are defined by the `n_point` argument.

This routine can take considerable time to execute. If $N_i$ is the number of input points and $N_p$ is the number of points in the output, then the number of calculations increases as $N_p \times N_i \times \log(N_i)$. This formulation tends to give somewhat distorted results for periods approaching zero. However, improving the accuracy for small periods increases the calculation time.

The algorithm was adapted from Craig (1981). As an example of its use, a simple sine wave was input into a FLAC table as an input acceleration. The function was then executed from a period of 0.5 to 2, with 50 points, in the output tables (see “SPEC.DAT”). Figure 1 shows the input acceleration generated; a sine wave with a period of 1.0. Figures 2 through 4 show the various response spectrums generated, displaying the expected peaks at a period of 1.0.

Reference

Data File “SPEC.DAT”

new
def cr_tab(num_point,end_time,perl)
    local i = 0
    local p2 = 2.*pi
    loop while i <= num_point
        local xx = end_time*float(i)/float(num_point)
        i = i+1
        local yy = sin(xx*p2/perl)
        table(1,xx) = yy
    end_loop
end

@cr_tab(250,3.0,1.0)

ca spec.fis suppress
@spectra(0.0,0.5,2.0,1,2,3,4,50);

; table 1 name ‘Input Acceleration’
table 2 name ‘Displacement Response’
table 3 name ‘Velocity Response’
table 4 name ‘Acceleration Response’
plot hold table 1 line
plot hold table 2 line
plot hold table 3 line
plot hold table 4 line
**FLAC (Version 6.00)**

**LEGEND**

7-May-08 16:09
step 0
Table Plot
Input acceleration

Itasca Consulting Group, Inc.
Minneapolis, Minnesota USA

**Figure 1** Input acceleration

**Figure 2** Displacement response spectrum

---

FLAC Version 6.0
Figure 3  Pseudo-velocity response spectrum

Figure 4  Pseudo-acceleration response spectrum
Strain-Softening FISH Model

The file “SS.FIS” is a FISH function which duplicates the built-in strain-hardening/strain-softening plasticity model in FLAC. The detailed explanation of the model is provided in Section 2.4.4 in Theory and Background. The function is named `m_ss` and requires that the following parameters be specified with the PROPERTY command:

- `m_coh`    cohesion
- `m_ctab`   table number relating cohesion to plastic shear strain
- `m_dil`    dilation angle
- `m_dtab`   table number relating dilation angle to plastic shear strain
- `m_fric`   friction angle
- `m_ftab`   table number relating friction angle to plastic shear strain
- `m_g`      shear modulus
- `m_k`      bulk modulus
- `m_ten`    tensile strength
- `m_ttab`   table number relating tensile strength to plastic tensile strain

These parameters default to zero if not specified. The user also has access to:

- `m_epdev`  shear-hardening parameter
- `m_epten`  tensile-hardening parameter
- `m_ind`    state indicator:
  - 0 elastic
  - 1 plastic shear
  - 2 elastic now, but plastic in past
  - 3 plastic tensile

The following problem compares the FISH model to the built-in strain-softening model. The built-in model is used for zones in the left half of the model. The FISH function is used for zones in the right half.
**Data File “SS.DAT”**

```
g 12 10
gen 0,0 0,25 30,25 30,0

model ss i=1,6
prop den 2500 bulk 1.19e10 shear 1.1e10 i=1,6
prop coh 2.72e6 fric 44 dil 0 ten 2e6 i=1,6
prop ctab 1 ftab 2 i=1,6

  tab 1 0,2.72e6 1e-4,2e6 2e-4,1.5e6 3e-4,1.03e6 1,1.03e6
  tab 2 0,44 1e-4,42 2e-4,40 3e-4,38 1,38

  call ss.fis
  model m_ss i=7,12
  prop den 2500 m_k 1.19e10 m_g 1.1e10 i=7,12
  prop m_coh 2.72e6 m_fric 44 m_dil 0 m_ten 2e6 i=7,12
  prop m_ctab 1 m_ftab 2 i=7,12

  ini xv 1e-6 i=1
  ini xv -1e-6 i=13
  fix x y i=1
  fix x y i=13

  his nstep 100
  his unbal
  his xdisp i=1 j=1
  his sxx i=6 j=1
  his sxx i=6 j=5
  his sxx i=6 j=10
  step 15000
  save ss.sav
  plot hold bou est disp
  return
```
Figure 1  Comparison of stresses and displacements
Shear Displacement-Softening Cables

*FISH* can be used to adjust material properties locally along a cable. For example, the shear resistance along a cable, defined by *sbond*, can be made to decrease as a function of the relative shear displacement. A table relating the shear resistance (i.e., bond strength) to shear displacement can be input to define this relation. The cable element properties are accessed directly in *FLAC*’s data structure. (Refer to Section 4 for a program guide to the elements of the data structure.)

The structural element logic in *FLAC* allows for only one set of properties per cable element. Therefore, it is necessary to describe a cable as a set of separate, connected cable elements. This is done with *FISH* function *stru_set* in the example data file “SSCAB.DAT.” *stru_set* is problem-dependent; for this example, a vertical ground anchor is created and will be pulled at a constant velocity from a rock mass. The anchor is represented by six separate cables; see Figure 1.

*FISH* function *bond_s* (in file “SSCAB.FIS”) evaluates the relative shear displacement at the cable nodes every calculation step, and varies the bond strength as a function of accumulated relative shear displacement. The variation is prescribed by Table 1 in the data file. In this example, the initial bond strength is 250 MPa, and decreases to 125 MPa after a relative shear displacement of 10 mm.

The file “STR.FIN,” called at the beginning of “SSCAB.FIS,” contains the names and values of the offsets on the cable data structure. This file is contained in the “\FISH\4-ProgramGuide” directory.

Histories are taken of the axial force that develops in the anchor and the relative shear displacement at selected points along the anchor. An additional *FISH* function, *check_sbond*, is given to monitor the change in the bond strength at different locations along the anchor. Figure 2 plots the shear force that develops along the anchor at the end of the calculation. Figure 3 plots the accumulated relative shear displacement at three cable nodes, and Figure 4 plots the change in bond strength at the same three nodes, versus the vertical displacement of the top cable node (node 7). Note that the bond strength decreases at nodes 4 and 5 along the anchor, but remains constant at the end of the anchor (node 1).

**Data File “SSCAB.DAT”**

```
title
    Ground anchor pull test with softening bond strength
g 4 6
mo el
    pro bulk 5e9 she 3e9 den 2000
def stru_set
    xi = 2.
    yi = 1.
    xf = 2.
    yf = 7.
nseg = 6
dy = (yf-yi)/float(nseg)
syi = yi
loop ii (1,nseg)
    syf = syi + dy
```
command
   stru pro ii e 1.e15 yield 1e20 a 1e-4 sbond 0.25e9 &
   kbond 1.e11 sfrc 0 perim 1
   stru cable beg xi syi end xf syf seg 1 prop ii
end_command
   syi = syf
end_loop
end
call sscab.fis
def find_id
   ip=imem(str_pnt+$ksnode)
loop while ip # 0
   case_of imem(ip+$kndid)
      case 7
      ipn7 = ip
      case 5
      ipn5 = ip
      case 4
      ipn4 = ip
      case 1
      ipn1 = ip
   end_case
   ip = imem(ip)
end_loop
end
def check sbond
   while_stepping
      pip1 = imem(ipn1+$kndtad)
      pip4 = imem(ipn4+$kndtad)
      pip5 = imem(ipn5+$kndtad)
      pip7 = imem(ipn7+$kndtad)
      bs1 = fmem(pip1+$ktypsb)
      bs4 = fmem(pip4+$ktypsb)
      bs5 = fmem(pip5+$ktypsb)
      bs7 = fmem(pip7+$ktypsb)
end
; --- install structural elements ---
stru_set
find_id
; --- table 1: sbond as a function of relative shear displacement ---
table 1 0 0.25e9 0.25e-2 0.25e9 1e-2 0.125e9 4e-2 0.125e9
; --- boundary condition ---
stru node 7 fix y ini yvel 1e-5
fix x y
; --- histories ---
hist ns 1
hist elem 6 axial
hist node 7 ydisp
hist node 1 sdisp
hist node 4 sdisp
hist node 5 sdisp
hist node 7 sdisp
hist bs1
hist bs4
hist bs5
hist bs7
hist unbal
; --- pull test ---
step 3000
; --- plots ---
plot hold bou cable struct num
plot hold bou cable bl struct cs_sforce struc sbond
;print struc cable
;print struc prop cable
plot hold his -1 vs 2
plot hold his 3 4 5 vs 2
plot hold his 7 8 9 vs 2
**Figure 1**  Ground anchor defined by six connected cables

**Figure 2**  Shear forces along the anchor
**Figure 3** Histories of relative shear displacement at three cable nodes

**Figure 4** Histories of bond shear strength at three cable nodes
Strain-Softening Interface

*FISH* can be used to adjust material properties locally along an interface. For example, strain-softening behavior can be prescribed along an interface to represent local weakening of a sliding fault plane. The interface properties are accessed directly in *FLAC*’s data structure. (Refer to Section 4 for a program guide to the elements of the data structure.)

The interface logic in *FLAC* allows for only one set of properties per interface. Therefore, it is necessary to describe an interface plane by a set of separate interfaces along the plane. This is done with the *FISH* function `create.int` in the example data file “SSINT.DAT.” `create.int` is problem-dependent; for this example, a slope is bounded by inclined and horizontal weakness planes. The two planes are represented by 11 interfaces. Each interface has a separate face prescribed for `aside`, but the same face for `bside`. The inclined weakness plane is created with five `INTERFACE` commands, and the horizontal plane with six `INTERFACE` commands. For all 11 interfaces, the `bside` begins at `i = 1`, `j = 2` and ends at `i = 25`, `j = 2`; see Figure 1.

The *FISH* function `int.var` (in file “SSINT.FIS”) evaluates the relative shear displacement along all the interfaces at every calculation step, and varies the cohesion and friction as a function of accumulated relative shear displacement. The variations are prescribed by tables that are set by `coh.tab` and `fri.tab`. In this example, the initial friction angle is 10° and decreases to 5° after a relative shear displacement of 2 mm. The variation is prescribed in table #2 (`fri.tab = 2`).

The file “INT.FIN,” called at the beginning of “SSINT.FIS,” contains the names and values of the offsets on the interface data structure. This file is contained in the “\FISH\4-ProgramGuide” directory.

An additional *FISH* function, `check.int`, is given to print the current accumulated relative shear displacement along each interface, and store the current friction angle of each interface in the table `out.tab`. Figure 2 shows the friction angle of the interfaces after 2000 steps. Interfaces 9, 10 and 11 have reached the residual value of 5°. This is consistent with the slope failure indicated by the displacement vector plot in Figure 3.

Data File “SSINT.DAT”

```
grid 24 7
mo el j 1
mo mo i 3 8 j 3 7
gen -3.464 6 -1.887 6 1 1 0 0 i 1 11 j 1 2
gen ss 11 1 11 0 i 11 25 j 1 2
gen 1 1 -1.30945357 1 i 39 j 38
gen line 1 1 -1.887 6
fix x y j 1
pro bulk 2e9 she 1e9 den 2500 j 1
pro bulk 1e9 she 5e8 den 2500 fric 45 i 3 8 j 3 7
set g 10
def create_int
;left side
loop jbeg (3,7)
```
jend=jbeg+1
ifn=8-jbeg
command
  int ifn aside from 3 jbeg to 3 jend bside from 1 2 to 25 2
  int ifn kn 1e9 ks 1e8 fric 10
end_command
end_loop
;bottom
loop ibeg (3,8)
  iend=ibeg+1
  ifn=3+ibeg
  command
  int ifn aside from ibeg 3 to iend 3 bside from 1 2 to 25 2
  int ifn kn 1e9 ks 1e8 fric 10
end_command
end_loop
end
create_int
ini syy -1e5 var 0 1e5 j37
ini szz -1e5 var 0 1e5 j37
hist unbal
hist xdisp i 8 j3
table 1 0 0 10 0
table 2 0 10 1e-3 7 2e-3 5 10 5
ca ssint.fis
set coh_tab=1 fri_tab=2 out_tab=100
step 2000
check_int
plot hold bou if
plot hold table 100 both min 0
plot hold grid dis
ret
**Figure 1** Interfaces defining planes of weaknes

**Figure 2** Friction angles along interfaces
Figure 3  Slope failure indicated by displacement vectors
Extrapolating Pore Pressure Change to Speed Convergence to Steady State

When only the pore-pressure distribution corresponding to steady-state flow is of interest, a flow-only calculation may be performed. If there are substantial differences in permeability or zone size throughout the grid, the number of cycles needed to reach steady state may be large. The FISH function “TURBO.FIS” may be used in conjunction with “FMOD5.FIS” to speed the convergence in these cases. “TURBO.FIS” is provided for experimental use only.

FISH function “TURBO.FIS” periodically makes an estimate of the changing slopes of the pore pressure histories at all gridpoints and extrapolates them to an estimated steady state, using the fact that the convergence must be of logarithmic form. The following data file illustrates the use of “TURBO.FIS” to speed up the convergence towards steady-state in a problem of flow around a high-permeability lens. See Section 1.10.4.2 in Fluid-Mechanical Interaction for additional discussion on the function “TURBO.FIS.”

This scheme should be used with great caution and discontinued if the results look at all strange.

Data File “TURBO.DAT”

```
conf gw extra=2
gr 20 20
mod elas
prop d 1 s 1 b 1
gen circle 10 10 5
prop perm=1e-10 reg=1,1
prop perm=1e-9  reg=10,10
set mech=off
apply pp=0  i=1
apply pp=10  i=21
water bulk 1e9
step 1
call fmod5.fis
call turbo.fis
solve sratio 0.01
no_turbo
solve sratio 0.01
plot hold s1 blue pp i=.5 cyan
ret
```
**Figure 1** Streamlines and pressure contours around a high-permeability lens
Ubiquitous-Joint FISH Model

The file “UBI.FIS” is a FISH function which duplicates the built-in ubiquitous-joint model in FLAC. The detailed explanation of the model is provided in Section 2.4.3 in Theory and Background. The function is named m_ubi, and requires that the following parameters be specified with the PROPERTY command:

- m_coh: cohesion
- m_dil: dilation angle
- m_fric: friction angle
- m_g: shear modulus
- m_jang: joint angle (measured counterclockwise from x-axis)
- m_jcoh: joint cohesion
- m_jfric: joint friction angle
- m_jten: joint tension limit
- m_k: bulk modulus
- m_ten: tensile strength

These parameters default to zero if not specified. The user also has access to

- m_ind: state indicator:
  0: elastic
  1: plastic shear
  2: elastic now, but plastic in past
  3: plastic tensile
  6: joint plastic shear
  7: joint elastic now, but plastic in past
  8: joint plastic tensile
The following problem (see Section 7 in the Verifications volume) compares the FISH model to the built-in ubiquitous-joint model. The built-in model is used for the upper half of the grid, and the FISH function is used for the lower half. The comparison of model results to the analytical solution is given in Figure 1.

Data File “UBI.DAT”

```
title
Compressive Strength of a Jointed Sample. (Built-in + FISH UBI)
g 6 10
gen -3 -5 -3 5 3 5 3 -5
call ubi.fis
;
def sigmav
  sum = 0.0
  loop i (1,igp)
    sum = sum + yforce(i,jgp)
  end_loop
  sigmav = sum / (x(igp,jgp)-x(1,jgp))
end
;
def ve
  ve = (ydisp(3,1)-ydisp(3,11))/(y(3,11)-y(3,1))
end
;
def anal
  mc= 2e3
  mfi = 40.0*degrad
  jc = 1e3
  jfi= 30.0*degrad
  sm = 2.0*mc*cos(mfi)/(1.0-sin(mfi))
  sj = 2.0*jc/((1.0-tan(jfi)*tan(30.0*degrad))*sin(2.0*30.0*degrad))
  anal=min(sj,sm)
end

def servo
while stepping
  if unbal > high_unbal then
    loop i (1,igp)
      yvel(i,jgp) = yvel(i,jgp)*0.975
      if abs(yvel(i,jgp)) > high_vel then
        yvel(i,jgp) = sgn(yvel(i,jgp))*high_vel
      end_if
      yvel(i,1) = yvel(i,1)*0.975
      if abs(yvel(i,1)) > high_vel then
        yvel(i,1) = sgn(yvel(i,1))*high_vel
      end_if
    end_loop
```

FLAC Version 6.0
end_if

if unbal < low_unbal then
  loop i (1,igp)
    yvel(i,jgp) = yvel(i,jgp)*1.025
    if abs(yvel(i,jgp)) > high_vel then
      yvel(i,jgp) = sgn(yvel(i,jgp))*high_vel
    end_if
  end_loop
end_if

set high_unbal 20 low_unbal 10 high_vel 1.0e-4
mo ubi i=1,3
prop den 2000 bulk 1e8 she 7e7 fric 40 co 2e3 ten 2400 i=1,3
prop jco 1e3 jfric 30 jang 60.0 jten 2000 i=1,3
mo m ubi i=4,6
prop den 2000 m_k 1e8 m_g 7e7 m_fric 40 m_coh 2e3 m_ten 2400 i=4,6
prop m_jcoh 1e3 m_jfric 30 m_jang 60.0 m_jten 2000 i=4,6
fix y j=1
fix y j=11
ini yvel -1e-7 j=11
ini yvel 1e-7 j=1
his unbal
his sigmav
his anal
his ve
his yv i 1 j 1
step 3000
print sigmav
print anal
save ubi.sav
Figure 1  Stress-strain curve for joint angle = 30°

Figure 2  Comparison of stresses and displacements
Gradual Unloading of Void Regions

The FISH function “ZONK.FIS” creates a void within a model and slowly relaxes the forces around the void region. This facility is useful for simulating a gradual excavation in elasto-plastic material. The influence of transients on material failure is minimized; the solution is more “static.” The following variables must be specified with the SET command:

\[ i_1, i_2 \quad \text{bounding} \quad i \text{-numbers of zones to be extracted} \quad (i_2 \geq i_1) \]

\[ j_1, j_2 \quad \text{bounding} \quad j \text{-numbers of zones to be extracted} \quad (j_2 \geq j_1) \]

\[ n_{\text{big\_steps}} \quad \text{number of reductions in applied forces} \]

\[ n_{\text{small\_steps}} \quad \text{number of FLAC steps within each force reduction step} \]

The applied forces are calculated for the boundary of the extracted region by specifying the FISH function \texttt{zonk}. The forces are then relaxed by specifying the function \texttt{relax}. The functions \texttt{zonk} and \texttt{relax} make use of extra arrays 6 and 7; the \texttt{CONFIG extra} command must be specified to have those extra grid variables available.

The example data file “ZONK.DAT” illustrates the use of “ZONK.FIS” to simulate a gradual excavation in Mohr-Coulomb material.

Data File “ZONK.DAT”

```plaintext
title
Hole in a Mohr-Coulomb medium
config extra=7
g 10 10
mo ss
prop shear=2.8e9 bulk=3.9e9 dens=2500 coh=3.45e6 fric=30 dil=0 ten=1e10

gen 0 0 0 10 10 5 0
; --- boundary conditions ---
fix y j 1
fix x i 1
app sxx=-30e6 i 11
app syy=-40e6 j 11
; --- initial conditions ---
ini sxx=-30e6 syy=-40e6 szz=-30e6
; --- histories ---
hist unbal
hist xd i 3 j 6
hist xv i 3 j 6
hist sxx i 3 j 6
hist syy i 3 j 6
hist szz i 3 j 6
hist sxy i 3 j 6
; --- gradual excavation ---
```

FLAC Version 6.0
call zonk.fis
set i1=1 i2=2 j1=5 j2=6
set n_big_steps=10 n_small_steps=100
zonk
relax

solve sratio 1.e-3
save zonk.sav
plot hold grid plas his
plot hold his -4 vs -2
ret

Figure 1  Excavation with plastic zone and history point
Figure 2  Horizontal stress vs displacement at history point