Writing and Compiling a Custom Material Property in ANSYS Mechanical APDL

Eric Miller

Co-Owner

Principal, Simulation and Business Technologies

09/27/2012

PADT, Inc.
Agenda

• Note: This presentation is being recorded

• Introductions
• Background and Requirements
• Compiling & Linking
• The User Material routines
• Simple Example
• Thoughts
Introductions
Upcoming Webinars

• Upcoming Webinars
  – October 11, 2012
    12:00 - 1:00 MST
    An Example of Moving Mesh Modeling of a Valve
  – October 25, 2012
    12:00 - 1:00 MST
    Getting Started with ANSYS Engineering Knowledge Manager (EKM)
  – November/December: Start the ANSYS 14.5 Webinars?

• See upcoming and past webinars at:
  – padtincevents.webex.com
    • Click on ANSYS Webinar Series
About PADT

• We Make Innovation Work
• **PADT is an Engineering Services Company**
  – Mechanical Engineering
  – 18 Years of Growth and Happy Customers
  – 70’ish Employees
• **3 Business Areas**
  – **CAE Sales & Services**
    • Consulting, Training, Sales, Support
  – **Product Development**
  – **Rapid Prototyping & Manufacturing**

Learn More: **www.PADTINC.com**

We Make Innovation Work
Cube HVPC Systems

- Balance between speed and cost
  - **Mini-Cluster**
    96 Cores / 512 GB RAM / 6 TB Disk
    Mobile Rack / UPS / Monitor / Keyboard
    $34,900

- **Compute Server**
  32 Cores / 256 GB RAM / 3 TB Disk
  $14,250

- **Simulation Workstation (Intel)**
  12 Cores / 96 GB RAM / 3 TB Disk
  $11,750

- **Simulation Workstation (AMD)**
  12 Cores / 64 GB RAM / 3 TB Disk
  $6,300

- [www.CUBE-HVPC.com](http://www.CUBE-HVPC.com)
PeDAL – The APDL Editor

- Side-by-side editor and help viewer layout.
- Instant help on any documented APDL command by pressing F1.
- Full syntax highlighting for ANSYS v12 Mechanical APDL.
- Auto-complete drop downs for APDL Commands.
- APDL Command argument hints while typing commands.
- Search ANSYS help phrases and keywords.
- Multiple tabs for the editor and html viewer.
- Full capability web browser built in allows for rich web experience and web searches.
Background and Requirements
ANSYS Mechanical APDL = OPEN

- MAPDL is the most open FEA commercial program on the planet
  - Designed and built that way in the 90’s
- Several ways to access:
  - APDL Macros – command based programming language
  - APDL Math – Access to the matrices in MAPDL
  - User routines – write and link user routines
    - Including utilities needed for routines
  - FORTRAN API
    - No longer fully documented
- You either compile a custom executable or you use external commands
  - We will do custom executable today
  - External commands link dynamic libraries at run time
    - Used for user commands and such
User Routines

• Can be C or FORTRAN
  – But we recommend only FORTRAN
  – Provided user subroutines are all FORTRAN
• Referred to as User Programmable Features, or UPF’s
• Different types:
  – Database access
  – User calculated loading
  – Modify or monitor existing ANSYS elements
  – Create a new element
  – Specify your own material behavior
  – Set up ANSYS to run as a subroutine in another program
• We are covering materials today, but most is applicable to all the other uses
• Note: Some routines won’t work correctly with Parallel
  – Verify parallel on test cases.
What you Need to Know

- **FOTRAN**
  - If you don’t know FOTRAN, you can figure it out, but it will be a lot of debugging.
    - Find a grey haired person to help you.

- **How the ANSYS solver works**
  - The theory guide is a good place to start
  - For the area you are using, you need to know what equations ANSYS uses and how it applies them.
  - Things like substeps, loadsteps, PREP7 vs POST1 vs SOLU, solver types, etc…

- **The math behind the thing you want to model**
  - Know this math inside and out because you probably will have to morph it to fit within how the solver needs it specified.
What you Need

• A full load of ANSYS MAPDL on your machine:
  – C:\Program Files\ANSYS Inc\v140\ansys\customize\user
  – /ansys_inc/v140/ansys/customize/user/
  – Should have full include and user directories

• Read and Write access to the vNNN directory and all sub folders

• The Programmer’s Manual
  – Mechanical APDL
    > Programmer’s Manual

• And…
The Most Important Thing You Will Learn Today……..
YOU MUST USE THE RIGHT COMPILER

• The number one problem we see with user routines is people using the wrong compiler!!!!!
  – It says it everywhere in the help, and still, it is a problem.
  – No maybe, no it kind of works. You must get the right one – Visual studio and compilers

• // Installation and Licensing Documentation // Windows Installation Guide // 2. Platform Details :: 0
  – Bottom of the page:

  **Compiler Requirements for Windows Systems**
  All ANSYS, Inc. products are built and tested using the Visual Studio 2008 SP1 (including the MS C++ compiler) and Intel FORTRAN 11.1 compilers.

• // Installation and Licensing Documentation // Linux Installation Guide // 2. Platform Details
  – Table 2.1

• It sometimes says “or newer”
  – Nope, you need the one listed
Intel Compiler

• ANSYS has been using the Intel compiler for some time
• Start at the Intel website:
• You may have to contact them to make sure you get the right version
  – Be very careful on this, ANSYS usually uses an older version because it is more stable and QA’d
But First!

• Do you really need a UPF?
  – Dig a little deeper into the material models and make sure you can’t use what is already there

• Will your material model work in ANSYS
  – Does it use the proper formulation and approach
  – Does it fit within the element and solve architecture
Some Advice

• Before you get deep into your model get the system working
  – Compiler, ANSYS, environment variables, etc…

• Take the standard usermat.f routine and get it to compile and link.
  – It has the basic TB, BISO model built in as a demo.

• Test it
  – I like to build two beams and run one with a standard BISO and another with the use routine

• Get everything working.

• Then make a small difference to the calculations and make sure you can see it

• Keep the test routine
  – If something stops working, you can go back and verify where you are.
Compiling and Linking Your Routine
Windows vs Linux, USERMAT vs Other

• You can do all of this on both platforms
  – We will cover Windows because it is the most common
  – Linux is very similar, just need to do things slightly different in syntax and such

• Same goes for other UPF’s
  – Method used for USERMAT works for most other routines

• Documentation can be used to see the differences

• Also: we will talk about USERMAT, it works for creep, hyperelasticity and all other user material UPF’s.
Two Ways: Custom Executable and Dynamic

- **Old Way: Custom Executable**
  - Use a supplied script to compile and link a custom ansys.exe
  - Accessed at run time by command line or launcher options
  - **Pros:**
    - Easy to deploy to other machines
    - You know you have a working executable
    - No special setup required, just the options when your run
  - **Cons:**
    - Takes longer to compile, a pain during debug loops
    - Big file to move around

- **Newer Way: Dynamic**
  - Use supplied script or APDL command to link at runtime
  - Can reside anywhere on the solver machine
  - Accessed through environment variable and/or an APDL command
  - **Pros:**
    - Quick compile time, great for debugging
    - You can have multiple versions of your routines, pick at run time
    - Using APDL command, you can actually compile at solve time
  - **Cons:**
    - Less control. DLL’s all over the place
    - Setup for users can be confusing, Environment variables and paths and such
    - If using compile at run, if the compile fails you don’t get real good feedback
Using DLL’s at Run Time

- Decide on a working directory (workdir) and get your usermat.f routine in that directory
  - Don’t change the name!
- Copy ansusershared.bat to workdir from: C:\Program Files\ANSYS Inc\v140\ansys\custom\user\winx64
- Open up the FORTRAN command line window
- CD to workdir
- Run ansusershared.bat
  - Enter the name of the routine you want compiled
  - Enter a blank return to get out of the script
- To use the DLL you made:
  - Set the environment variable: ANS_USER_PATH=workdir
Using DLL’s at Run Time

- Recommended method
- Use different directories for each version of routines and change environment variable to access what you want
- Robust, you either have a DLL or you don’t
Using /UPF

- First, set up some environment variable letting the program know you are going to use /UPF
  - Make sure that your ANSYS executable directory is in your PATH
    - It needs to run a script called findUPF.bat
  - Set ANS_USE_UPF=TRUE
- From your working directory, launch the FORTRAN command line
- Add /upf,userrmat.f to your input command file
- Run ansys in batch mode from the command line shell
- A DLL will be made, reuse it just like when you make the DLL
Using /UPF

• I’m not a fan of this method
  – Have to run ANSYS from the FORTRAN shell
    • If you set up paths so any routine can compile/link you can run without shell
  – Need compiler on machine you are solving on
  – If your compile fails you are kind of screwed
  – Added because ABAQUS allows for compile at run

• Only works with batch mode
Making a Custom ansys.exe

• Open up the FORTAN command line shell
• You can do this in the custom\user\winx64 directory or in your own directory
  – I prefer your own. If so, copy:
    anscust.bat, ansys.lrf, ansysex.def from custom\user\winx64 to your directory
• Copy your routine to the workdir
• CD to the workdir
• Run anscust.bat
  – It looks for any UPF’s and if it finds them, compiles them
  – You will get an ansys.exe
• To use it, specify the path in the launcher (customization/preferences) or with the –custom <path> switch
Making a Custom ansys.exe

- Takes a while to compile but when it is done, it works.
- No need for environment variables.
- Note: Do not rename the executable, must be ansys.exe
  - Use directories to have different versions
Compiling Recommendations

- Debug using the DLL
- If it is just you, keep using the DLL
- If you deploy to others, when everything is working, make a new ansys.exe and deploy that
- Remember to do everything from the FORTRAN shell
The User Material Routines
The Basics

- Standard FORTRAN, nothing fancy
- Well documented
- Comes with the TB, BISO model
- Contains several subroutines
  - Usermat
    - Doesn’t do much, just figures out dimension of element and calls proper routine:
      - Usermat1d: 1D truss
      - Usermat3d: 3D elements
      - Usermatbm: beam elements
      - Usermatps: plain strain
- Works on current element technology only
  - Does not work with legacy elements
The Basics

- The routine gets called for every integration point in your model that is assigned the material number that is defined by a TB, User
- Stress, Strain, state variables, time increment, strain increment are passed in
- Your routine updates values and passes them back
- Read documentation on math
- Lots of helper routines provided to make your job easier
  - General routines you will need
  - Vector utilities
  - Matrix utilities
Call

- Standard call, all the info that gets passed to the routine is listed
Input Arguments

• Documented in the comments

c input arguments
  ===========
c  matId   (int,sc,i)  material #
c  elemId  (int,sc,i)  element #
c  kDomIntPt (int,sc,i) "k"th domain integration point

c  kLayer  (int,sc,i)  "k"th layer

c  kSectPt (int,sc,i)  "k"th Section point

c  ldstep  (int,sc,i)  load step number

c  isubst  (int,sc,i)  substep number

c  nDirect (int,sc,in)  # of direct components

c  nShear  (int,sc,in)  # of shear components

c  ncomp   (int,sc,in)  nDirect + nShear

c  nstatev (int,sc,i)  Number of state variables

c  nProp   (int,sc,i)  Number of material constants

c  Temp    (dp,sc,in)  temperature at beginning of
c                     time increment

c  dTemp   (dp,sc,in)  temperature increment

c  Time    (dp,sc,in)  time at beginning of increment (t)

c  dTime   (dp,sc,in)  current time increment (dt)

c  Strain  (dp,ar(ncomp),i) Strain at beginning of time increment

c  dStrain (dp,ar(ncomp),i) Strain increment

c  prop    (dp,ar(nprop),i) Material constants defined by TB,USER

c  coords  (dp,ar(3,i)) current coordinates

c  defGrad_t(dp,ar(3,3),i) Deformation gradient at time t

c  defGrad (dp,ar(3,3),i) Deformation gradient at time t+dt
Input/Output

- These go in and out, so be careful.
- Note the VARn is not used right now
- State variables: Important
  - These for your use to pass things back and forth
  - How you supply values that you can change
    - As opposed to properties that don’t change
    - Unique to each integration point
  - Also how you store any specific “result” or “intermediate” values at each integration point that you want to plot or list
  - Very powerful

```
c input output arguments
  c stress (dp,ar(ncomp),io) stress
  c ustatev (dp,ar(nstatev),io) user state variables
  c sedEl (dp,sc,io) elastic work
  c sedPl (dp,sc,io) plastic work
  c epseq (dp,sc,io) equivalent plastic strain
  c epsPl (dp,ar(ncomp),io) plastic strain
  c var? (dp,sc,io) not used, they are reserved arguments for further development
```
Output

- Stuff that is passed out

```markdown
<table>
<thead>
<tr>
<th>output arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keycut (int,sc,o)</td>
<td>loading bisect/cut control</td>
</tr>
<tr>
<td></td>
<td>0 - no bisect/cut</td>
</tr>
<tr>
<td></td>
<td>1 - bisect/cut</td>
</tr>
<tr>
<td></td>
<td>(factor will be determined by solution control)</td>
</tr>
<tr>
<td>dsdcF1 (dp,ar(ncomp,ncomp),o)</td>
<td>material jacobian matrix</td>
</tr>
<tr>
<td>tsstif (dp,ar(2),o)</td>
<td>transverse shear stiffness</td>
</tr>
<tr>
<td></td>
<td>tsstif(1) = Gxz</td>
</tr>
<tr>
<td></td>
<td>tsstif(2) = Gyz</td>
</tr>
<tr>
<td></td>
<td>tsstif(1) is also used to calculate hourglass stiffness</td>
</tr>
<tr>
<td></td>
<td>stiffness, this value must be defined when low order element, such as 181,</td>
</tr>
<tr>
<td></td>
<td>182, 185 with uniform integration is used.</td>
</tr>
<tr>
<td>epsZZ (dp,sc,o)</td>
<td>strain epsZZ for plane stress,</td>
</tr>
<tr>
<td></td>
<td>define it when accounting for thickness change in shell and plane stress states</td>
</tr>
</tbody>
</table>
```
Important details

• Ncomp: Number of terms for each type of element
• Vector orders
• Matrix order

```c
ncomp 6 for 3D (nShear=3)
ncomp 4 for plane strain or axisymmetric (nShear = 1)
ncomp 3 for plane stress (nShear = 1)
ncomp 3 for 3d beam  (nShear = 2)
ncomp 1 for 1D (nShear = 0)

stresses and strains, plastic strain vectors
11, 22, 33, 12, 23, 13 for 3D
11, 22, 33, 12 for plane strain or axisymmetry
11, 22, 12 for plane stress
11, 13, 12 for 3d beam
11 for 1D

material jacobian matrix
3D
dsdeP1 1111 1122 1133 1112 1123 1113 |
dsdeP1 2211 2222 2233 2212 2223 2213 |
dsdeP1 3311 3322 3333 3312 3323 3313 |
dsdeP1 1211 1222 1233 1212 1223 1213 |
dsdeP1 2311 2322 2333 2312 2323 2313 |
dsdeP1 1311 1322 1333 1312 1323 1313 |
plane strain or axisymmetric (11, 22, 33, 12)
dsdeP1 1111 1122 1133 1112 |
dsdeP1 2211 2222 2233 2212 |
dsdeP1 3311 3322 3333 3312 |
dsdeP1 1211 1222 1233 1212 |
plane stress (11, 22, 12)
dsdeP1 1111 1122 1112 |
dsdeP1 2211 2222 2212 |
dsdeP1 1211 1222 1212 |
3d beam (11, 13, 12)
dsdeP1 1111 1113 1112 |
dsdeP1 1311 1313 1312 |
dsdeP1 1211 1213 1212 |
1d
dsdeP1 1111 |
```
Rest of Routine

• Declares types
• Then has and If-then-else to call the proper subroutine for the dimension of the element
  – Just pass everything through
  – They do this so that the logic of the program is not full of if-then-else statements.
• Header info repeats for each subroutine
USERMAT3D

• This is where you would do your own thing
• Simple example for biso is here
  – Get values
  – Calc elastic and plastic slopes
  – Our first helper function: vmove (copies vectors)

```c
keycut = 0
dsigdep = ZERO
pleq_t = ustatev(1)
pleq = pleq_t

*** get Young's modulus and Poisson's ratio, initial yield stress and others
young = prop(1)
posn = prop(2)
sigy0 = prop(3)

*** plastic strain tensor
  call vmove(ustatev(2), epsPl(1), ncomp)

*** calculate plastic slope
  dsigdep = young*prop(4)/(young-prop(4))
twoG = young / (ONE+posn)
threeG = ONEHALF * twoG
elast1=young*posn/((1.0D0+posn)*(1.0D0-TWO*posn))
elast2=HALF*twoG

*** define tsstif(1) since it is used for calculation of hourglass stiffness
tsstif(1) = elast2
```
USERMAT3D

• This is where you would do your own thing
• Simple example for biso is here
  – Get values
  – Calc elastic and plastic slopes
  – Our first helper function: vmove (copies vectors)

```c
/*
   keycut  = 0
dsigdep = ZERO
pleq_t  = ustatev(1)
pleq    = pleq_t
*/

*** get Young's modulus and Poisson's ratio, initial yield stress and others
young  = prop(1)
posn    = prop(2)
sigy0  = prop(3)

*** plastic strain tensor
    call vmove(ustatev(2), epsPl(1), ncomp)

*** calculate plastic slope
    dsigdep = young*prop(4)/(young-prop(4))
twoG    = young / (ONE+posn)
threeG  = ONEHALF * twoG
elast1=young*posn/((1.0D0+posn)*(1.0D0-TWO*posn))
elast2=HALF*twoG

*** define tssstif(1) since it is used for calculation of hourglass stiffness
    tssstif(1) = elast2
*/
```
USERMAT3D

- Calculate the elastic stiffness matrix

```c
**calculate elastic stiffness matrix (3d)**

dseEl(1,1)=(elast1+TWO*elast2)*G(1)*G(1)
dseEl(1,2)=elast1*G(1)*G(2)+elast2*TWO*G(4)*G(4)
dseEl(1,3)=elast1*G(1)*G(3)+elast2*TWO*G(5)*G(5)
dseEl(1,4)=elast1*G(1)*G(4)+elast2*TWO*G(1)*G(4)
dseEl(1,5)=elast1*G(1)*G(5)+elast2*TWO*G(1)*G(5)
dseEl(1,6)=elast1*G(1)*G(6)+elast2*TWO*G(4)*G(5)
dseEl(2,2)=(elast1+TWO*elast2)*G(2)*G(2)
dseEl(2,3)=elast1*G(2)*G(3)+elast2*TWO*G(6)*G(6)
dseEl(2,4)=elast1*G(2)*G(4)+elast2*TWO*G(1)*G(4)
dseEl(2,5)=elast1*G(2)*G(5)+elast2*TWO*G(1)*G(5)
dseEl(2,6)=elast1*G(2)*G(6)+elast2*TWO*G(2)*G(6)
dseEl(3,3)=(elast1+TWO*elast2)*G(3)*G(3)
dseEl(3,4)=elast1*G(3)*G(4)+elast2*TWO*G(5)*G(6)
dseEl(3,5)=elast1*G(3)*G(5)+elast2*TWO*G(5)*G(5)
dseEl(3,6)=elast1*G(3)*G(6)+elast2*TWO*G(6)*G(6)
dseEl(4,4)=elast1*G(4)*G(4)+elast2*(G(1)*G(2)+G(4)*G(4))
dseEl(4,5)=elast1*G(4)*G(5)+elast2*(G(1)*G(6)+G(5)*G(4))
dseEl(4,6)=elast1*G(4)*G(6)+elast2*(G(4)*G(6)+G(5)*G(2))
dseEl(5,5)=elast1*G(5)*G(5)+elast2*(G(1)*G(3)+G(5)*G(5))
dseEl(5,6)=elast1*G(5)*G(6)+elast2*(G(4)*G(3)+G(5)*G(6))
dseEl(6,6)=elast1*G(6)*G(6)+elast2*(G(2)*G(3)+G(6)*G(6))
do  i=1,ncomp-1
  do  j=i+1,ncomp
    dseEl(j,i)=dseEl(i,j)
  end do
end do
```
Stop

• At this point, if the inputs and outputs sound confusing you need to back up and understand ANSYS non-linear solving and how their elements work
  – Theory manual
• Book that was used by ANSYS
• Calculate the stresses
• Note use of `get_ElmData` to get element call
  – Documented as part of `usermat` documentation
  – Used to get info that is not passed in
• Get yield…
• Next section checks for yield
  – If no, use a goto (yes, a goto!) to skip plastic stuff
• Do plastic calcs

```fortran
! Material Jcobian matrix
IF (qE1.LT.sQtiny) THEN
  cconf1 = ZERO
ELSE
  cconf1 = threeG * dpleq / qE1
END IF
CON2 = threeG/(threeG+dsigdep) - cconf1
CON2 = TWOTHIRD * CON2
DO I=1,NCOMP
  DO J=1,NCOMP
    JMB(J,1) = ZERO
  END DO
END DO
DO I=1,NDIRECT
  DO J=1,NDIRECT
    JMB(J,1) = -THIRD
  END DO
END DO
JMB(1,1) = ONE
END DO
DO I=NDIRECT+1,NCOMP
  JMB(1,1) = HALF
END DO
DO I=1,NCOMP
  DO J=1,NCOMP
    dsdcEl(I,J) = ddsdcEl(I,J) - twoG
      * ( CON2 * dfds(I) * dfds(J) + cconf1 * JMB(1,J) )
  END DO
END DO
END DO
GOTO 600
```

```fortran
C *** check for yielding
  IF (SIGY .LE. ZERO.cr.fratio .LE. -SMALL) GO TO 500
C
  SIGY_T = SIGY
  threeOv2qEl = ONEHALF / qEl
C *** compute derivative of the yield function
  DO I=1, NCOMP
    dfds(I) = threeOv2qEl * sigDev(i)
  END DO
  ONEOV3G = ONE / threeG
  qEl0v3G = qEl * ONEOV3G
C *** initial guess of incremental equivalent plastic strain
  DPLEQ = qEl0v3G - SIGY * ONEOV3G
  PLEQ = PLEQ_T + DPLEQ
  SIGY = SIGY0 + DSIGNDEP * PLEQ

*** update stresses
  DO I = 1, NCOMP
    STRESS(I) = SIGELP(I) - TWOTHIRD * (qEl-SIGY) * dfds(I)
  END DO

*** update plastic strains
  DO I = 1, NDIRECT
    EPSPL(I) = EPSPL(I) + DFDS(I) * DPLEQ
  END DO
  DO I = NDIRECT + 1, NCOMP
    EPSPL(I) = EPSPL(I) + TWO * DFDS(I) * DPLEQ
  END DO
  EPSSEQ = PLEQ

*** Update state variables
  USTATEV(1) = PLEQ
  DO I=1,NCOMP
    USTATEV(I+1) = EPSPL(I)
  END DO

*** Update plastic work
  SEDPL = SEDPL + HALF * (SIGY_T+SIGY)*DPLEQ
```
USERMAT3D

- Clean up and get out
  - Note the 500-600 elastic portion
- Thoughts
  - Simple calcs, yours will probably be much more complex
    - But steps are the same
    - Gather your properties
    - Branch if needed to for different equations
    - Figure out strain/stress
    - Return the info
  - Didn’t use a lot of calls to other routines
  - Remember it gets called for every integration point
    - You need to be efficient

```c
500 continue

C *** Update stress in case of elastic/unloading
   do i=1,ncomp
      stress(i) = sigElp(i)
   end do

600 continue
   sedEl1 = ZERO
   DO i = 1, ncomp
      sedEl1 = sedEl1 + stress(i)*(Strain(i)+dStrain(i)-epsPl(i))
   END DO
   sedEl    = sedEl1 * HALF
   ustatev(nStatev) = sigy

C
   return
end
```
USERMAT

• Restrictions
  – Current-technology elements only
  – If you want to plot state variables, you need to use /graph, full
  – Not enough hooks in/out for incompressible materials
    • Special routine (UserHyper) for that

• Only one usermat per model
  – There is a way around this, use one of your material properties as a flag to access different models
  – Check the flag then call a subroutine for the proper material
TB

• TB, User, *Mat, NTEMPS, NPTS*
  – Mat is material number
  – NTEMPS is number of temperature points you will provide properties at
  – NPTS, number of property values
    
    | tb, user | 1, 2, 4 |
    | tbtemp | 1.0 |
    | tbdata | 1, 19e5, 0.3, 1e3, 100 |
    | tbtemp | 2.0 |
    | tbdata | 1, 21e5, 0.3, 2e3, 100 |

• TB, State, *Mat,,NPTS*
  – Specifies the material and number of state variables you will use
  – NPTS max is 1000, yes, 1000
  – Plot/list with ETABLE, ESOL
Simple Example
Modified Slightly from the Help

- Two elements, pull on them
- One is TB,BISO, the other TB,USER
- Files will be on The Focus Blog tomorrow
- Modified usermat.f
  - Scale yield by 0.75
Using the User Mat

- Mat2 is the user mat
- Same properties, just a different table
1: Modify usermat.f

- Make sure ANS_USER_PATH is pointing to my user directory
- Copy to my working directory
- Edit and in usermat3d subroutine change sigy0 line to:
  - sigy0 = prop(3)*.75
- Save file
- Launch FORTRAN command line shell
- ansusershared.bat
- Run ANSYS with demo input file as input
- Check output: BISO and USER stresses and strains are different
Results

- Note that it tells us we are using a user mat

<table>
<thead>
<tr>
<th>TIME</th>
<th>1 S X</th>
<th>2 S X</th>
<th>1 S Y</th>
<th>2 S Y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SX_BISO</td>
<td>SX_USER</td>
<td>SY_BISO</td>
<td>SY_USER</td>
</tr>
<tr>
<td>0.10000</td>
<td>-0.188102E-02</td>
<td>-0.197699E-02</td>
<td>1509.45</td>
<td>1134.47</td>
</tr>
<tr>
<td>0.28750</td>
<td>-0.110968</td>
<td>-0.110968</td>
<td>1525.07</td>
<td>1158.90</td>
</tr>
<tr>
<td>0.45625</td>
<td>-0.814415</td>
<td>-0.814415</td>
<td>1536.67</td>
<td>1161.69</td>
</tr>
<tr>
<td>0.66204</td>
<td>-1.73150</td>
<td>-1.73150</td>
<td>1548.95</td>
<td>1173.97</td>
</tr>
<tr>
<td>0.89592</td>
<td>-1.86235</td>
<td>-1.86235</td>
<td>1561.97</td>
<td>1186.99</td>
</tr>
<tr>
<td>1.0000</td>
<td>-0.176949E-01</td>
<td>-0.176949E-01</td>
<td>1569.16</td>
<td>1194.18</td>
</tr>
</tbody>
</table>

Note - This ANSYS version was linked by Licensee
Let's try it live…
Thoughts
Parallel

• Things get tricky with parallel
• You can get it to work
• Compare parallel and non-parallel on all hardware options
  – Make sure they match
• For shared memory parallel:
• All UPF ‘s are supported in parallel
• But don’t use Common Block variables.
  – Each core may have a different value.
  – You don’t want to set them different on each core
  – You can usually read them if they are not something that is changed by a solve
  – But don’t write to them
Convert UPF into ANSYS

- ANSYS does convert customer/university supplied material UPF’s into the solver

- A few things needed:
  - More than just one user out there wants it, need to show need
  - You have published test results/and or theoretical papers to verify your accuracy
  - The model is free of all legal claims
  - You have time to work with ANSYS development to work out any issues and help with testing

- Contact your support provider
  - If they can't help, contact me.
Hints

• Use state variable to set flag for first time used, write something to output that says “HEY, I’m BEING USED!”
  – Maybe even give more info on the routine
  – Use iout = wrinqr(2) to get output unit

```
if(ustatev(9) .eq. 0) then
  iout = wrinqr(2)
  ustatev(9) = 1.0
  write (iout,2000)
2000 format ('**** ERM3 CALL TO ANSYS, INC USERMAT 
  end if
```

• User erhandler() to send out notes, warnings, errors
  – Could use it rather than write above
• User /UNDO to write a *.db file at ansy point
• Crawl, Walk, Run
Thank You…

- PADT Enjoys doing these webinars…
- Please consider us as your partner
- ANSYS Related
  - Training, Mentoring
  - Consulting Services
  - Customization
  - Sales (if in AZ, NM, CO, UT, NV)
- Stratasys 3D Printers and Systems
- CUBE HVPC Systems
- Product Development
  - High-end engineering with practical, real world application
- Rapid Prototyping
  - SLA, SLS, FDM, PolyJet, CNC, Soft Tooling, Injection Molding
- Help us by letting us Help you