



No this is not how PADT's staff suits up when embarking on a particularly difficult scripting process. We would never wear those colors!

## ACP: Realistic Composite Models Without the Sparkly Costume

By Jason Krantz

You can model almost any structure with ANSYS—as long as it's not carbon fiber. That's a bit of an overstatement, but not by much. Complex composite structures with compound curvature (e.g., monocoque bicycle frames) are a pain to model in Classic, requiring Olympic-level APDL gymnastics (and, in some offices, Olympic-level sequined unitards). However, ANSYS has fixed this scripting and wardrobe nightmare with the release of Ansys Composite PrepPost (ACP).

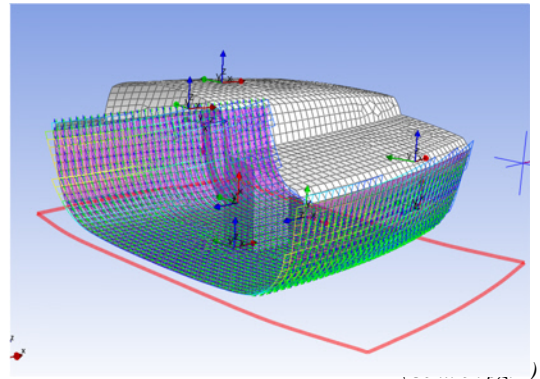
ACP razes several longstanding barriers to using ANSYS for sophisticated composite analysis. ACP makes fiber alignment much, much easier. ACP further adds much-needed pre- and post-processing tools, such as material draping and sophisticated failure criteria.

It's also fully scriptable (using the Python language) which opens up intriguing possibilities for automating aspects of pre- and post-processing.

It is possible to accurately model composites in ANSYS without ACP, but you'll need both extraordinary APDL skills and extraordinary patience. If you had very simple geometry such as a plate, sphere or cylinder, you could skip the APDL. But most analyses involve more complex geometry.

Let's say you're trying to build a model of that carbon bicycle frame. You can use shell elements to mesh the CAD geometry easily enough, but you still need to tell

(Cont. on pg. 2)



### In this Issue...

- 1.....ACP: Realistic Composite Models Without the Sparkly Costume
- 1.....Part 1: Modeling Cracks in ANSYS
- 6.....Going "Over the Top" with Joint Simulation in ANSYS Mechanical
- 6.....Training Schedule
- 8.....Passing Entity Attributes from NX to ANSYS MAPDL
- 10.....About PADT



You do an article on crack modeling, and you have to go with the classic "fat guys on a stool" pictures. No Choice.

By Carlos Shultz

Fracture analysis is a serious field of study, I won't be cracking any more jokes in this article. ANSYS supports several of the common techniques used to analyze cracks. In a typical first analysis, an engineer assumes a flaw shape and size and then uses the stress results from an FEA model as input into NASGRO. If a more detailed simulation is required, an FEA model with a crack included can be created.

There are a few guidelines for using ANSYS to model cracks which will be demonstrated through an example of a steel turbine wheel, shown in Figure 1, which has been assumed to have a crack introduced during the (Cont. on pg. 4)

### Part 1

## Modeling Cracks with ANSYS

(ACP, Cont...)

ANSYS which way the fibers point—that is, which direction to use for the 0° material direction.

ANSYS Mechanical APDL assumes that the 0° direction corresponds with the element coordinate system's X direction. Before ACP, the most direct way to indicate material direction was to rotate each element's coordinate system. Analysts typically used the ESYs command to rotate the element coordinate system into alignment with the laminate's 0° direction.

The process goes like this:

- Select the element you want to orient based on X,Y,Z location
- Select the nodes associated with that element via the NSLE command
- Offset the working plane to the average of those nodes via NWPAVE
- Rotate the WP so that its X direction is aligned with the fiber direction
- Create a local coordinate system aligned with the WP coordinate system (CSWPLA)
- Align the element's coordinate system with the newly-created local CS (ESYS,N)
- Lather, rinse and repeat for every element in your model

It's true that you can simplify these steps a bit for prismatic members such as aircraft wings. Even then, the number of \*do loops and local coordinate systems quickly becomes unwieldy. For compound curvature in three dimensions, each element in your model may have a different material direction; orienting each element's coordinate system can take a long time even when fully automated.

Let's get back to our bicycle frame. Most modern carbon bicycles have irregularly shaped tubes, so aligning your element CSYs with a local cylindrical coordinate system will not save you here. The joints between tubes may each require their own variation on the above script. Essentially, it's impractical to analyze the frame with the APDL method outlined above—such an analysis would take far too long to set up.

Prior to the release of ACP, anyone using ANSYS to model complex composite parts ended up using a third-party tool to indicate material direction. FiberSim is one option, though licenses can run into the six-figure range depending on the options you select. A few other options exist, but they lack the tight integration that ANSYS has been striving for in recent releases.

What ANSYS users really need is an integrated composites pre- and post-processor, and that's exactly what ACP is. The tool introduces a few new concepts, but don't fret—they're really just extensions of existing ANSYS ideas.

Oriented element sets (OES) are a central tool for indicating material direction. (We'll discuss the details shortly). OESes are just groups of elements that use a common set of criteria to describe their material directions. You might think of them as named selections or components.

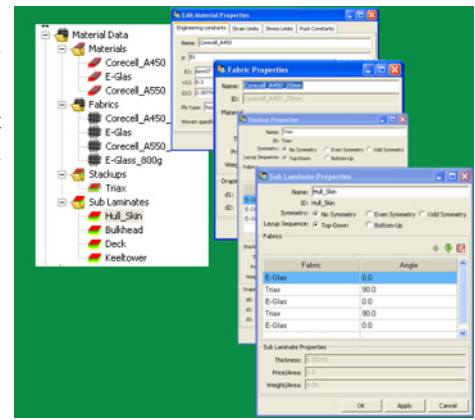
"Rules" are one way to select elements you'd like to include in an OES. They're not so foreign, though; ACP rules are analogous to selection logic in Classic or spheres of influence in Workbench.

We've seen that it's burdensome to indicate material direction in ANSYS Classic. How does ACP make it easier? Essentially, it frees the analyst from the tyranny of element coordinate systems. ACP lets you create arbitrary coordinate systems and then tie them to sets of elements. ACP "wraps" the material using these CSes as a guide. Material direction ends up being a "best fit" as defined by the relevant CSes and various physical constraints (such as draping).

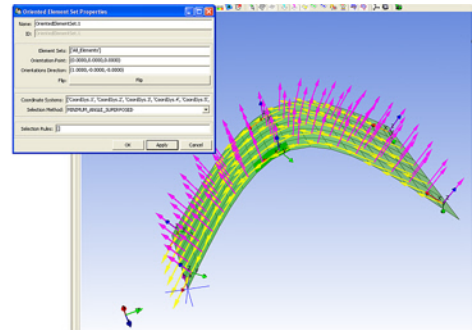
This may sound complicated, but in practice, it's not hard to get your material direction aligned the way you want it. For example, you can apply multiple CSes to a single set of elements. ACP will automatically select the most relevant CS to use for wrapping.

You can then check the material direction ACP chose by plotting the 0° material direction—ACP shows you an arrow for each element. By modifying the relevant CSes or adding new ones, you can tweak the material direction to match your intent. In practice, it's a bit like using Workbench's size and method controls to get the mesh you want.

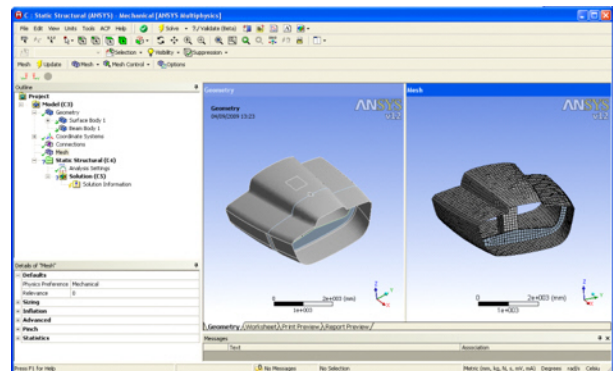
(Cont. on pg. 3)



A hierarchical data structure is used to enter, display and store information on materials,



Oriented Element Sets are used to Specify and View Element Orientation



ACP is Built Directly into ANSYS Workbench and Shares the Same Look and Feel that Users are Comfortable with



(ACP, Cont...)

You can also specify how ACP decides which CS is the most relevant for a particular OES: the CS that creates the smallest angle between the CS Z direction and the element's Z direction, the CS that is physically closest to each element and several variations on these. ACP then wraps the material so that its 0° direction is aligned as closely and realistically as possible with the relevant CS's X direction.

Several Workbench tools help prepare a model for ACP. Coordinate systems defined in Workbench show up in ACP, as do named selections of bodies. (Named selections of surfaces are nodal components in ANSYS Classic, and ACP "sees" element components, not nodal ones).

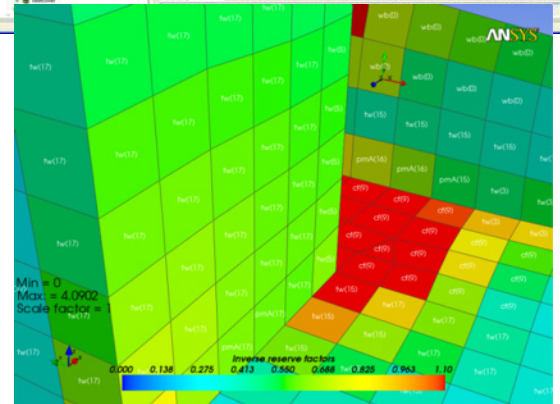
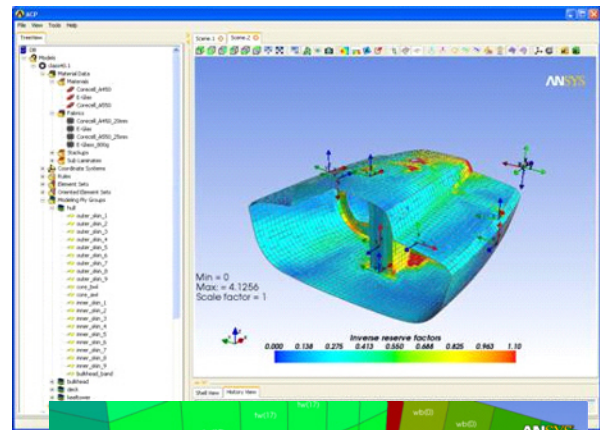
You can slice up your model with DesignModeler, combining the slices into a multibody part. By creating named selections out of one or more sub-bodies, you can divide your model into element sets that show up in ACP. Each multibody part's mesh is contiguous among its sub-bodies, so you don't introduce any discontinuities this way. This is handy not only for indicating fiber alignment but also for defining details like ply drop-off boundaries.

Anyone who's tried to apply wrapping paper to an irregularly shaped gift (say, a rugby ball) has struggled with draping. Essentially, the 2-D paper doesn't always conform to the gift the way you might expect it to. For our bicycle frame, this same effect makes it hard to predict exactly how the fibers of each swatch will be oriented when the swatches are pressed into the frame mold. Since composites tend to be highly anisotropic, that orientation matters a lot.

ACP takes draping into account when calculating material direction; this feature is essential to creating FEA models of composite structures that correspond to real-world finished parts. There's another benefit: ACP can "back out" the 2D swatch shapes required to get the desired 3D fiber orientation. These shapes can be used as templates for cutting real-world swatches for your bicycle frame, saving time and material costs.

FiberSim's version of this feature is well suited to production environments. ACP isn't meant to compete with FiberSim in such situations, but its draping and ply-book generation are more than adequate for design and development purposes.

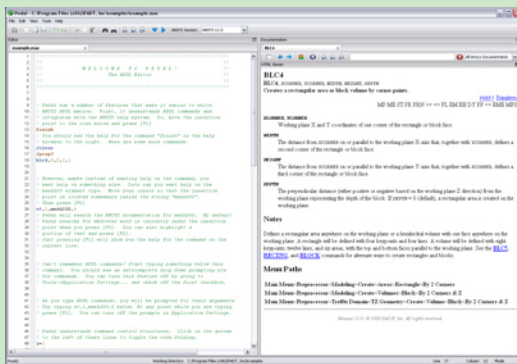
There's not nearly enough room in *The Focus* to give a full description of ACP. But if your work involves laminar composites to any degree, it may well be worth giving ACP a whirl. It makes the ANSYS suite attractive to a whole group of industries that perhaps ignored it in favor of more composite-friendly packages. Best of all, ANSYS composite analysts won't have to wear those embarrassing sparkly unitards. To see a recorded seminar on this from ANSYS, Inc, visit the [Demo Room](#).



Post Processing Includes Failure Criteria and Standard Stresses/Strains per Layer



## The Editor for ANSYS APDL Users



PeDAL is a Windows text editor for ANSYS APDL scripts. It integrates with the ANSYS help system to provide instantaneous help on any one of the 1,000s of ANSYS commands. PeDAL was written by Matt Sutton, an Engineer at PADT, to make his own job easier. Matt has years of experience writing APDL scripts and has long wished for a tool that would provide help for a given command right at his fingertips. Pedal can be purchased for \$49 by pressing on the Buy Pedal button below.

### Key Features

- 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.
- Mouse hover command descriptions.
- Much More...

Download your 30 day trial or learn more details at:

[www.padtinc.com/pedal](http://www.padtinc.com/pedal)



Figure 1: Fictitious Turbine Wheel  
Used in this Example

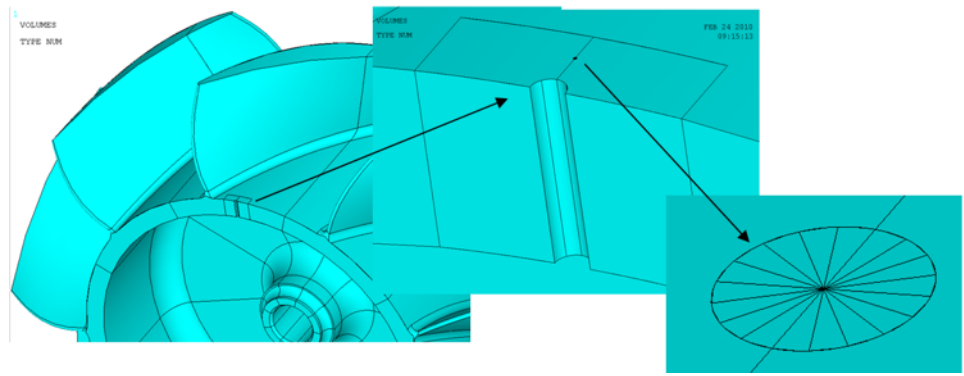


Figure 2: Crack Geometry Created Using Boolean Operations

balancing process. A link to the geometry model (v12 UP20090415) and APDL command files are provided at the end of this article in order to allow the user to reproduce this analysis step by step.

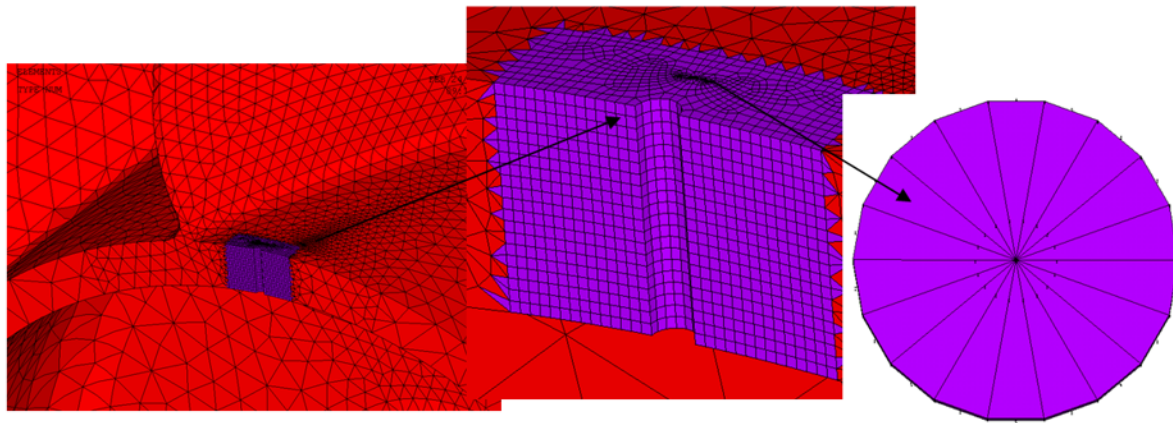


Figure 3: Meshed model, red=solid92, purple=solid95, elements at the crack tip have the midside nodes moved to 1/4 point radius location

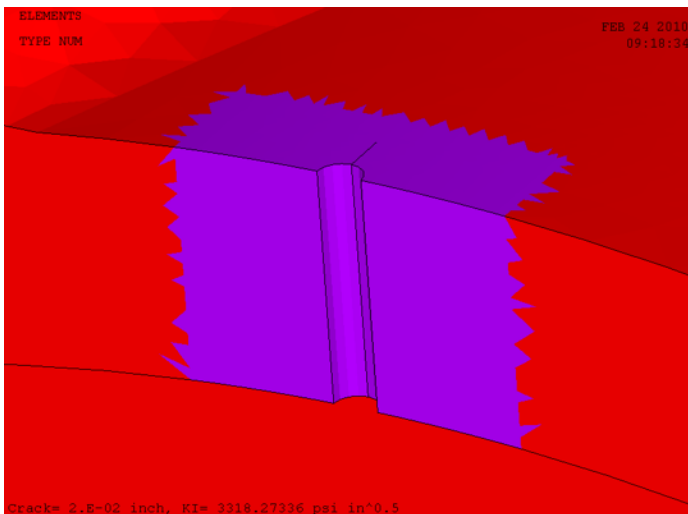


Figure 4: Element plot of the meshed model with /edge,1,1 and /pnum,type,1

Rotating components are often balanced to reduce bearing loads by cutting sacrificial material. A zero thickness crack is introduced starting at the bottom of a balancing cut. Figure 2 shows the booleaned volumes to create a balancing cut and a crack. Note that the crack tip is a series of wedge volumes that will be meshed with wedge elements.

Figure 3 shows the volumes are meshed with a combination of brick 95, wedge 95, tet 92, and pyramid 95 transition elements. Note that the wedge elements at the crack tip need to have the midside nodes moved to the 1/4 radius location. This creates the stress singularity needed to properly model the stress distribution seen in a crack front.

Figure 4 shows the mesh is discontinuous at the crack and so shows up as a line on the surface.

A baseline model was first run without a crack. The S1 result from that run is shown in Figure 5. Maximum S1 shown is equal to 28.8 ksi.

(Cont. on pg. 5)

(Crack Modeling, Cont...)

The model was then run with a 0.020in deep crack. The USUM and S1 result from that run are shown in Figure 6. The USUM plot shows the discontinuity of displacement across the crack. Maximum S1 shown is equal to 140 ksi. Figure 7 shows that the crack opens, as expected, under the applied loading.

Once the model has been solved, the elastic stress intensity factors can be calculated in POST1 using the KCALC command. After creating a path including 5 nodes along the crack face, the command macro KCALC is run. The results of that command are shown in Figure 8. This crack is opening in a mode 1 fashion which is consistent with the results of KI=5280, KII=554, and KIII=345.

The value of KI=5280 psi in<sup>0.5</sup> would then be used to look up the crack opening rate on a da/dN vs dK chart similar to the one shown in Figure 9. The units of dK in the chart are ksi in<sup>0.5</sup> so 5280 psi in<sup>0.5</sup> would predict a da/dN ~ 1e-7 in/cycle. That means that after 1000 cycles of this load, the crack should grow ~ 1e-7 \* 1000 = 1e-4 in. We can add this extension to the crack and rerun the analysis with a crack a=0.020 + 1e-4 = 0.0201in to see if the crack will continue to grow at the same rate.

This new crack length results in a KI=5296 which predicts a slightly higher da/dN. This process of extending the crack, rerunning KCALC, and then extending the crack can be repeated for the full loading history. A successful design will typically be able to withstand the assumed cracks by not accelerating to failure.

### Summary

- Cracks can be introduced to existing models using boolean operations
- Crack parameters including KI, KII, KIII are calculated using KCALC
- KI is used to estimate da/dN which in turn is used to determine updated crack length a\*=a+da
- a\* Model is then used to calculate a new KI

The APDL and geometry files are available here:

[ftp://ftp.padtinc.com/public/thefocus/Focus73\\_Crack\\_Modeling.zip](ftp://ftp.padtinc.com/public/thefocus/Focus73_Crack_Modeling.zip)

Look for Part 2 of this article in next month's Focus.

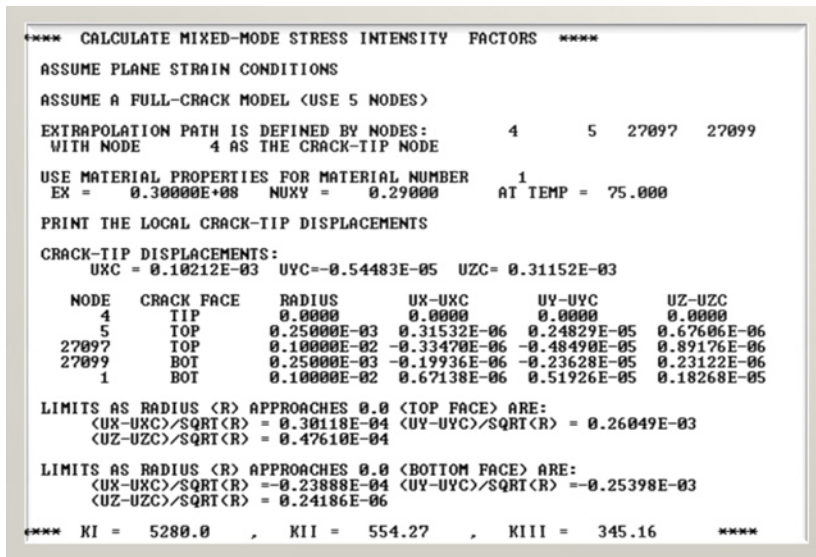


Figure 8: KCALC Output

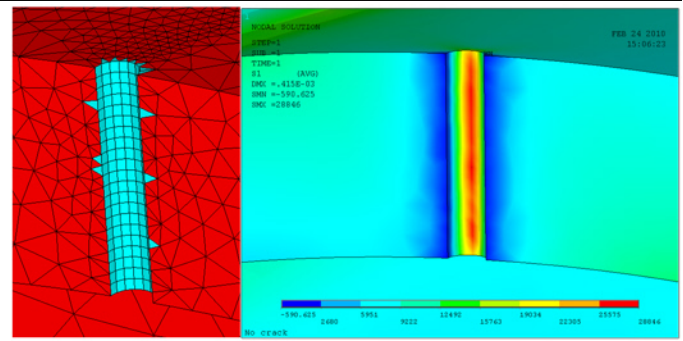


Figure 5: Element and S1 plot of the model without a crack

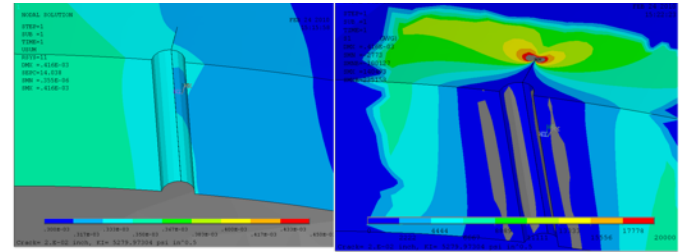


Figure 6: USUM and S1 plots of the model with a 0.020 in crack

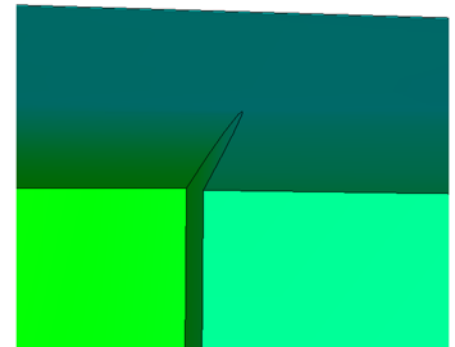


Figure 7: USUM, total displacement shows that

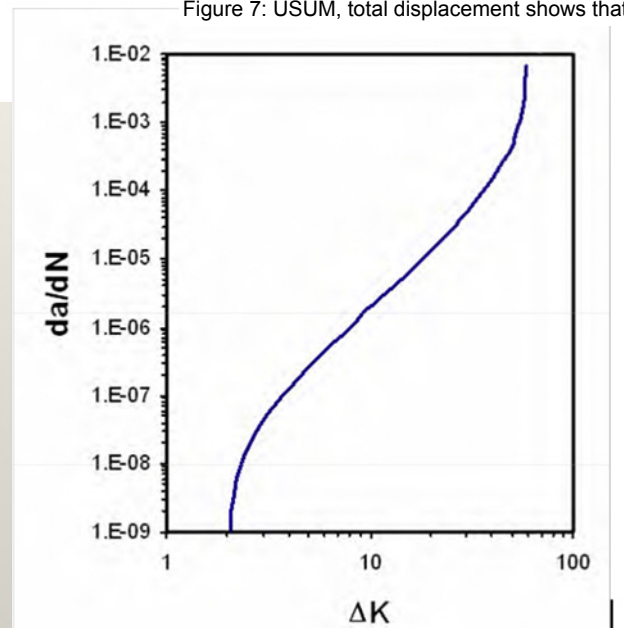


Figure 9. Sample da/dN vs dK from [Fatigue Crack Growth Equations For TC-128B Tank Car Steel](#).

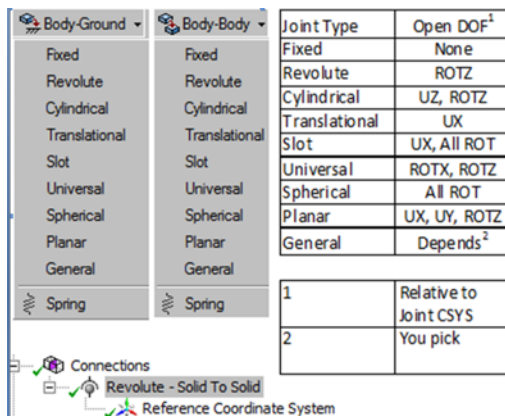




# Going “Over the Top” with Joint Simulation in ANSYS Mechanical

By: Doug Oatis

It's been a while since I've written a Focus article, so considering most of my articles contain some sort of oddball cultural reference, an inspiring night of top-notch cinematic adventures was just what the doctor ordered. As I watched Lincoln Hawk go *over the top* on Bob 'Bull' Hurley, I had an epiphany. The joint capabilities in Workbench would be perfect for analyzing arm wrestling...as well as actual engineering work! In case you don't get the reference, I'm referring to the greatest arm-wrestling movie ever made, “Over the Top”, starring Sylvester Stallone as a trucker who is trying to rebuild his life and reconnect with his son by winning an arm-wrestling tournament where the grand prize is a big rig. They just don't make movies like this anymore...



So back to ANSYS. Simulating joints used to require some creativity and constraint equations to allow parts to slide/rotate/etc... relative to each other. Along the way, the MPC184 element was introduced, which could simulate everything from a welded connection to a spherical joint, depending on the keyoptions and coordinate systems used. The downside to using the MPC184 in the Mechanical APDL (formerly known as ANSYS Classic) was that it was difficult to visualize the joint and confirm that the parts were properly connected without running a solution. Now, I've never finished setting up a model, hit solve, and come back the next day only to realize something was wrong, so this wasn't a big issue for me. However, if you're not like me, odds are you've said a few choice words after realizing a coordinate system was mis-aligned or a keyoption incorrectly set.

Starting with v11 Simulation (now known as Mechanical), you could define joints between flexible or rigid parts. You could easily visualize the joint (Cont. on pg. 7)

## PADT's Training Schedule

| Month   | Start | End  | #   | Title  | Location      |
|---------|-------|------|-----|--|---------------|
| Mar '10 | 3/8   | 3/9  | 203 | ANSYS Mechanical APDL Dynamics                         | Tempe, AZ     |
|         | 3/11  | 3/12 | 501 | ANSYS/LS-DYNA  | Tempe, AZ     |
|         | 3/15  | 3/16 | 604 | Introduction to CFX                                    | Tempe, AZ     |
|         | 3/17  | 3/17 | 112 | Introduction to ANSYS Meshing                          | Tempe, AZ     |
|         | 3/18  | 3/18 | 107 | ANSYS Workbench DesignModeler                          | Tempe, AZ     |
|         | 3/22  | 3/24 | 902 | Multiphysics Simulation for MEMS                       | Tempe, AZ     |
|         | 3/25  | 3/25 | 653 | CFX Turbulence Modeling                                | Tempe, AZ     |
|         | 3/30  | 3/31 | 502 | ANSYS Explicit STR                                     | Tempe, AZ     |
| Apr '10 | 4/6   | 4/15 | 113 | Introduction to ANSYS Workbench Mechanical (Web Class) | Web Based     |
|         | 4/7   | 4/9  | 101 | Introduction to ANSYS (Mechanical APDL), Part I        | Tempe, AZ     |
|         | 4/14  | 4/16 | 401 | ANSYS Mechanical APDL Low Frequency Electromagnetics   | Tempe, AZ     |
|         | 4/19  | 4/20 | 201 | ANSYS Mechanical APDL Basic Structural Nonlinearities  | Tempe, AZ     |
|         | 4/21  | 4/22 | 204 | ANSYS Mechanical APDL Advanced Contact and Fasteners   | Tempe, AZ     |
| May '10 | 5/4   | 5/5  | 103 | Introduction to ANSYS Workbench Mechanical             | Las Vegas, NV |
|         | 5/6   | 5/7  | 100 | Engineering with Finite Element Analysis               | Tempe, AZ     |
|         | 5/10  | 5/10 | 107 | ANSYS Workbench DesignModeler                          | Tempe, AZ     |
|         | 5/11  | 5/12 | 205 | ANSYS Workbench Mechanical Dynamics                    | Tempe, AZ     |
|         | 5/14  | 5/14 | 702 | ANSYS DesignXplorer                                    | Tempe, AZ     |
|         | 5/17  | 5/18 | 207 | ANSYS Workbench Mechanical – Structural Nonlinearities | Tempe, AZ     |
|         | 5/19  | 5/20 | 302 | ANSYS Workbench Simulation 11.0 Heat Transfer          | Tempe, AZ     |
| Jun '10 | 6/1   | 6/2  | 121 | ANSYS Mechanical 12.0 Advanced (Using Command Objects) | Tempe, AZ     |
|         | 6/7   | 6/8  | 201 | ANSYS Mechanical APDL Basic Structural Nonlinearities  | Tempe, AZ     |
|         | 6/9   | 6/10 | 204 | ANSYS Mechanical APDL Advanced Contact and Fasteners   | Tempe, AZ     |
|         | 6/14  | 6/15 | 301 | ANSYS Mechanical APDL Heat Transfer                    | Tempe, AZ     |
|         | 6/22  | 6/23 | 604 | Introduction to CFX                                    | Tempe, AZ     |
|         | 6/24  | 6/24 | 112 | Intro to ANSYS Meshing                                 | Tempe, AZ     |
|         | 6/25  | 6/25 | 107 | ANSYS Workbench DesignModeler                          | Tempe, AZ     |

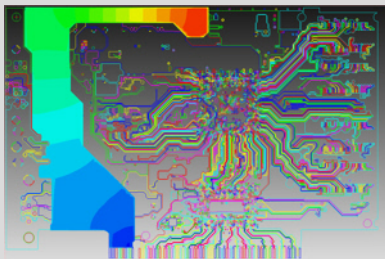
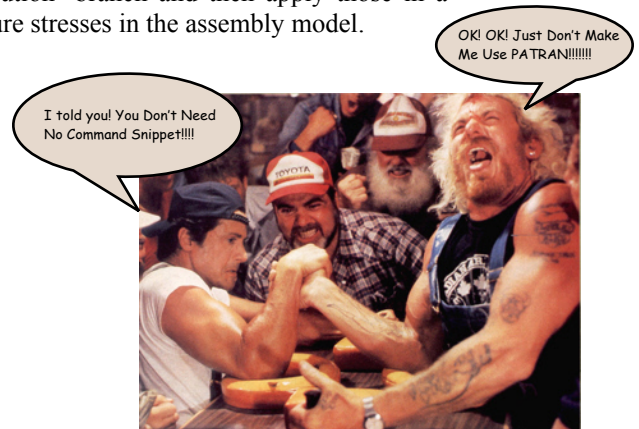
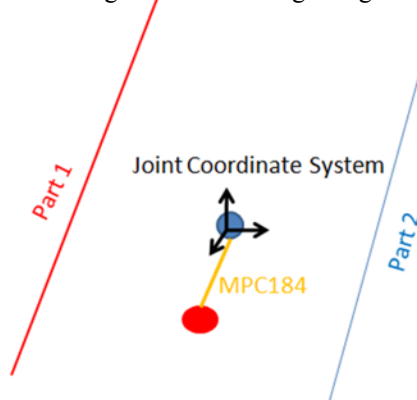
(Joint Simulation, Cont...)

coordinate system, open degrees of freedom, and even test everything without solving! Then R12 came out and allowed you to specify stops and locks, defined relative to the joint coordinate system. To visualize the difference between a stop and lock (I'll avoid the obvious pop-n-lock dance style joke) is that a stop is like a while, while a lock is like a latch that doesn't let go. Once you hit the wall, you can back up, while the latch locks the joint up for the rest of the solution.

There are two different categories of joints. The first is Body-Ground joint. You must have at least one of these if you wish to use the 'Configure' option to test the behavior of your joint system (more on that later). This type of joint provides the equivalent of a 'Fixed Support', only you can open specific degrees of freedom corresponding to different joints. The second type is a Body-Body joint, which connects two parts together. The documentation does a good job explaining what each joint simulates, complete with animations of relative motion, so I won't go into any details of the relative DOFs other than the table shown. It's important to note that all of these DOFs are relative to the joint coordinate system. This is shown underneath the defined joint in the 'Connections' branch, and can be easily realigned just like any user-defined coordinate system in Simulation. After you're done setting up your joint system, you can click on the 'Configure' button and actually slide/rotate/etc your parts around. It's hours of fun for the kids!

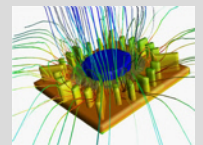
So now you're probably wondering what is actually created in the input deck. I contracted a 4<sup>th</sup> grade class from the local school to draw a picture to help illustrate (okay...I drew it). Constraint-type contacts with pilot nodes are created for each surface tagged in the joint definition. These are indicated by the red/blue lines (contact surface) and red/blue dots (target nodes). To connect the two surfaces together, an MPC 184 element and local coordinate system are created between pilot nodes for the two surfaces. Keyoptions are then set that allow motion between the two target nodes based on the joint coordinate system. You have the option to define either an RBE3 or CERIG type constraint connection between the pilot nodes and corresponding body nodes.

Finally, you're asking why anyone would want to use joints (which requires the purchase of an additional license). The first reason is that simulating these types of connections using traditional contact elements is difficult and expensive. Just try simulating a spherical joint using frictionless contact on a meshed spherical face and you'll understand what I mean. The second reason is that this allows you to do some pseudo sub-modeling. You can pull joint loads in the 'Solution' branch and then apply those in a part-by-part model rather than being forced into using a large mesh to capture stresses in the assembly model.



### PADT Adds Training and Support for ANSYS ICEPAK

If you are involved in the thermal modeling of electronic systems then you probably have heard of ICEPAK. And now you can count on PADT to help you be more productive with ICEPAK. Our support team recently completed certification and we offer support to our ANSYS Sales customers and training to anyone who has ICEPAK. For More Details contact Steve Hendry at 207-333-8780 or e-mail [steve.hendry@padtinc.com](mailto:steve.hendry@padtinc.com).



### 2010 PADT Webinars

This year PADT is making a full switch to web based seminars. We decided to offer two series: a Technical one and a Product focused one. Anyone can attend and we hope to see more Focus readers swelling our numbers.

There is usually room for 50 attendees so check our web-site for more information.

| Technical Series |  | Product Series |  |
|------------------|--|----------------|--|
| 3/25/10          | Modeling Composites with ANSYS                                   | Postponed      | Detailed Fatigue Calculations with nCode and ANSYS |
| 4/22/10          | New, Interesting and Underused Elements in ANSYS Mechanical APDL | 4/6/10         | Maximizing the ROI of your CAE Investment          |
| 5/27/10          | ANSYS Explicit Dynamics Tools                                    | 5/4/10         | Optimization with ANSYS Design Xplorer             |



# Passing Entity Attributes from NX to ANSYS MAPDL

By: Eric Miller

Workbench users have become accustomed to having access to CAD system parameters and named selections. It is one of those very cool capabilities that can really save a large amount of time and effort. What a lot of users do not know is that you can also get entity name and assigned attributes from NX into ANSYS Mechanical APDL (the program formerly and erroneously known as ANSYS Classic). When you execute the ~ugin (File->Import->UG) the program reads in the geometry and creates two files that identify named entities and attributes assigned to them. If you write a macro to read those files, you now have that information transferred into ANSYS for your use. In this article we will discuss one way of doing this and share the macros we developed. Even if you do not use NX with MAPDL, these macros are a good example of reading and parsing a text file, so they are worth a look.

## One Approach: Use Unique Names and Control with Attributes

Every entity in NX has properties, and within those properties the user can define a single name and any number of attributes. But names in NX are different than components in MAPDL: each entity in NX can have only one name, but in MAPDL they can belong to many different components. The way we decided to solve that is to put unique names on the edges, faces and volumes and use the attributes to define components. We do not use the default names because they are very long and cryptic. Once the entities have unique names, we then go in and define things like components, loads and boundary conditions in NX.

On the ANSYS side, we use the standard translator to read in the geometry, and then the macros discussed below to parse the name and attribute files. The contents of those files are stored in arrays that are used to create components or apply the loads or boundary conditions. Again, this is only one of many approaches, but it works well for us.

## Setting up the NX Model

The first thing we want to do is assign a unique name to each entity. We do this using Type Filters and the properties dialog, as shown in Figure 1:

- 1: Zoom out so that you can see your entire model
- 2: Go to the Type Select drop down and choose "Edge"
- 3: Use Ctrl-A to select all the edges in the model
- 4: Go to Edit->Properties to bring up the Properties Dialog
- 5: Click on the General Tab
- 6: Enter "edge" for the name and click the Add index to name option, starting at 1.
- 7: Click OK
- 8: Now the edges are name named EDGE1, EDGE2, etc...
- 9: Repeat for Face, using the name surf
- 10: Repeat for Solid Body, using the name volu
- 11: If you created Datum Points, do the same for Points using the name point

Now it is time to assign attributes to those entities. We do this by selecting the entities we want in a component, then bringing up the property dialog again, adding attributes with a title of CM and a string with the name we want the component to have as the value (Figure 2).

For loads and BC's we use a convention of the type of load or BC, followed by a direction. So F\_FX for force in the X direction and DSP\_UY for displacement in the UY direc-

(Cont. on pg. 9)

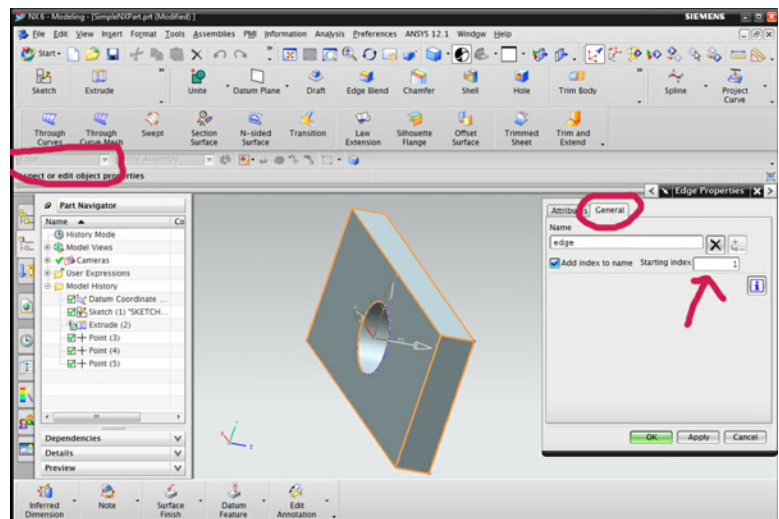


Figure 1: Naming Entities in NX

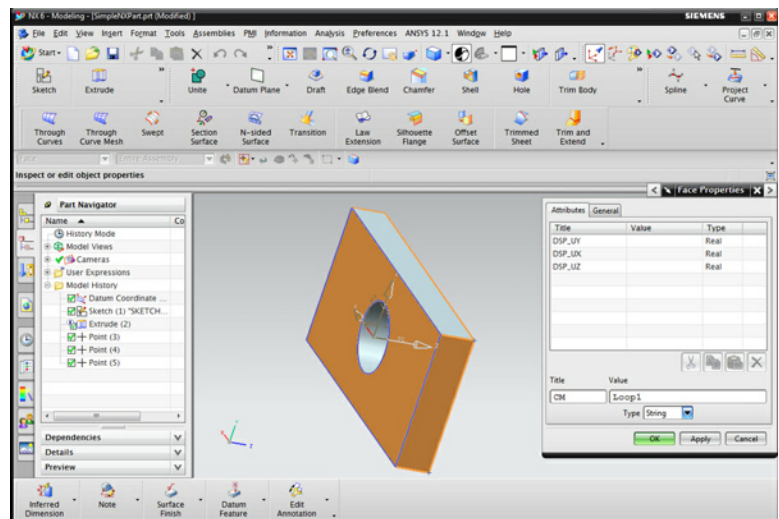


Figure 2: Assigning Attributes in NX



(NX to MAPDL Cont...)

tion. Make sure you that you change the type to real, or things won't work right.

Once you have the attributes assigned, save your file and it is time to import into ANSYS.

### Importing Into MAPDL

The first thing you will want to do is read in the NX part file with ~ugin or File-> Import->UG. One the reading is done, if you look in your working directory you will find two files: ugfilename.tbl and ugefilename.attr, where ugfilename is the root name of your NX part file.

The TBL file contains a list of all the entities in the NX file with names and the ANSYS entity number that was assigned to the entity. So to get this information into MAPDL, you run NXRDATBL.MAC. It reads the file, parses it, and puts the results into two arrays: NXP\_ENT\_S contains the names assigned in NX and NXP\_ENT\_N contains the type of entity in the first column and the ANSYS number for the entity in the second. Take some time to look through the array and see how it works. It is a fairly good example of reading a file with \*SREAD, and parsing it with the string functions available in APDL. The results of the read are displayed to the output window using another macro called, NXLIST,1.

Once the entities names and numbers are stored it is time to read the attributes. This is done with NXRDATTR. It also goes through a text file, parses out the needed information and places it into arrays. In this case it puts it into five arrays:

```
nxp_ent: Is the entity name from NX
nxp_kw: is the parameter name (keyword)
nxp_typ: Stores if it is a real value (1) or a string (0)
nxp_val: stores the parameter value, if it is a real number
nxp_sval: stores the parameter value, if it is a string
```

At the end, the macro lists out the parameters with NXLIST,2.

### Building Components

So, now that everything is inside ANSYS MAPDL, it is time to build the components, so we named the macro that does that NXBLDCM.mac. This macro is pretty complicated and can get a bit confusing with arrays used as indices to arrays and lots of double checking and flag setting. If you really want to understand it, we recommend you diagram all the loops on a sheet of paper. Also, this macro is a good example of using string substitution into commands to reduce the number of if-then-else statements.

What it basically does is:

- Loop through all the parameters
- If it finds one that starts with 'CM' it then checks to see if there are any components that already have that name, and if so, set a flag
- When it finds a match, it either adds the entity to the existing component, or creates a new one.

That is pretty much it. The names of the components are stored in NXP\_CM and the count is stored in NXP\_NCM.

All of the macros are reprinted here and area available, along with a test model, on our ftp site:

[ftp://ftp.padtinc.com/public/thefocus/Focus73\\_NX\\_to\\_MAPDL.zip](ftp://ftp.padtinc.com/public/thefocus/Focus73_NX_to_MAPDL.zip)

### Next Steps

Take some time to review the macros and understand how they work. There are some advanced APDL options in there that you may find useful in other places. Also, feel free to add to or modify the approach we have taken to do things like loads and boundary conditions. If you are real gung-ho you can also write an interface on the NX side using VB to apply the attributes and automate the whole process.

The macros are available for download at <LINK> and are shown on the following pages. Notice how nice the source code looks? Formatted, indented and color coded? That is because we used PeDAL to write these macros and when we got done, we used the Export option to create formatted source code that can be pasted right into any document. Check out pedal at

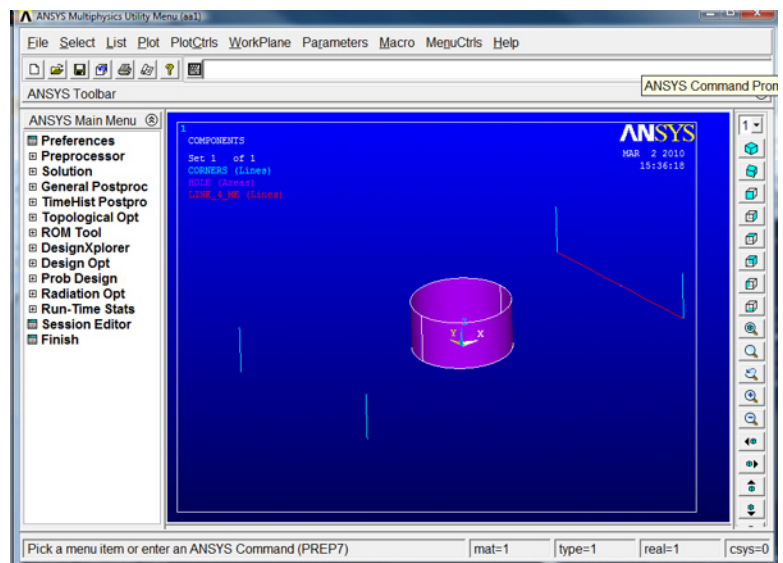


Figure 3: Components in ANSYS

**RNALL.MAC**

```

1: ! rnall.mac: Runs test case for NX Import
2: finish
3: /clear
4: ~UGIN,'simplenxpart','prt',,SOLIDS,1-256,0
5: /prep7
6: save,m1,db
7: resume,m1,db
8: nxrdtbl,'simplenxpart'
9: nxrdatr,'simplenxpart'
10: lnxblcdm

```

**NXRDTBL.MAC**

```

1: !-----
2: !      PADT                                Phoenix Analysis & Design
3: !                                           Technologies
4: !
5: !=====
6: !      * (480) 813-4884 * 1-800-293-PADT * (480) 813-4807 F *
7: !      * www.PADTINC.com * info@padtinc.com *
8: !=====
9: !      File: nxrdtbl.mac
10: !      File Type: Macro
11: !      Description: Reads NX table file (*.tbl) which contains the assigned
12: !      names of entities. It places the Entity name and ANSYS entity number into
13: !      arrays for use elsewhere.
14: !
15: !      Input:
16: !                  arg1: The filename root for the filename.tbl file
17: !
18: !      Output:
19: !          npx_ent_s(n): An array that stores the entity names
20: !          npx_ent_n(n,1): First column in array, stores the type of entity
21: !                      1 = KP, 2 = Line, 3 = Area, 4 = Volume
22: !          npx_ent_n(n,1): Second column in array, stores the ANSYS number
23: !                      for the entity
24: !          nxpentnum: Number of entities defined
25: !
26: !          NXP KP, NXPLN, NXPAR, NXPVL: Constants that associate number with
27: !          entity type
28: !
29: !+++++
30: /nopr
31: pname_ = arg1
32:
33: npx_ent_s=
34: npx_ent_n=
35: NXP KP = 1
36: NXPLN = 2
37: NXPAR = 3
38: NXPVL = 4
39:
40: *dim,tbls_,string
41: *dim,objtyp_,string,35
42: *dim,npx_ent_s,string,35,100
43: *dim,npx_ent_n,array,100,2
44:
45: *sread,tbls_(1),pname_,tbl
46: *get,nmvals_,parm,tbls_,dim,2
47:
48: nxpentnum = 0

```

Continued...

## NXRD\_TBL.MAC, continued.

```

49:  *do,i_,1,nmvals
50:    *if,substr(tbls_(1,i_),1,1),ne,'!',then
51:      *if,strlen(tbls_(1,i_)),gt,1,then
52:        nxpentnum = nxpentnum + 1
53:        nxp_ent_s(1,nxpentnum) = substr(tbls_(1,i_),1,36)
54:        objtyp_(1) = substr(tbls_(1,i_),37,25)
55:        nxp_ent_n(nxpentnum,2) = valchr(substr(tbls_(1,i_),61,10))
56:
57:        *if,objtyp_(1),eq,'KP',then
58:          nxp_ent_n(nxpentnum,1) = NXPKP
59:        *endif
60:        *if,objtyp_(1),eq,'LINE',then
61:          nxp_ent_n(nxpentnum,1) = NXPLN
62:        *endif
63:        *if,objtyp_(1),eq,'AREA',then
64:          nxp_ent_n(nxpentnum,1) = NXPAR
65:        *endif
66:        *if,objtyp_(1),eq,'VOLUME',then
67:          nxp_ent_n(nxpentnum,1) = NXPVL
68:        *endif
69:      *endif
70:    *endif
71:  *enddo
72:  nxlist,1
73:  *DEL,,PRM_
74:  /go

```

## NXRDATTR.MAC

```

1:  !-----
2:  !      PADT                                Phoenix Analysis & Design
3:  !                                           Technologies
4:  !
5:  !=====
6:  !      * (480) 813-4884 * 1-800-293-PADT * (480) 813-4807 F *
7:  !      * www.PADTINC.com * info@padtinc.com *
8:  !=====
9:  !      File: nxrdattr.mac
10: !      File Type:      Macro
11: !
12: !      Description:
13: !      Reads an attribute file created by ~ugin. (*.attr) It places the values
14: !      into arrays for use by any user elsewhere.
15: !      nxp_cm(n) contains the component name that the attribute applies to
16: !      nxp_kw(n) the keyword that identifies the parameter (parameter name)
17: !      nxp_typ(n) the type of variable (1 = string, 0 = real)
18: !      nxp_val(n) the real value for the parameter, 0 if type = string
19: !      nxp_sval(n) the string value for the parameter, blank if type = real
20: !
21: !      Input:
22: !      arg1: The filename root for the filename.attr file
23: !
24: !      Output:
25: !      nxp_ent(n): contains the name of the entity that the
26: !                  attribute applies to
27: !      nxp_kw(n): the keyword that identifies the parameter (parameter name)
28: !      nxp_typ(n): the type of variable (1 = string, 0 = real)
29: !      nxp_val(n): the real value for the parameter, 0 if type = string
30: !      nxp_sval(n): the string value for the parameter, blank if type = real
31: !      nxpnum: the number of attributes created
32: !
33: !+++++

```

Continued...



**NXRDATTR.MAC, continued.**

```

34: pname_ = arg1
35:
36: nxp_ent=
37: nxp_cm_typ=
38: nxp_kw=
39: nxp_typ=
40: nxp_val=
41: nxp_sval=
42:
43: *dim,atrics_,string
44: *dim,objtyp_,string,35
45: *dim,nxp_ent,string,35,1000
46: *dim,nxp_kw,string,35,1000
47: *dim,nxp_typ,,1000
48: *dim,nxp_val,,1000
49: *dim,nxp_sval,string,35,1000
50:
51: *sread,atrics_(1),pname_,attr
52:
53: *get,nmvals_,parm,atrics_,dim,2
54:
55: nxpnum = 0
56: *do,i_,1,nmvals_
57:     *if,substr(atrics_(1,i_),1,1),ne,'!',then
58:         *if,strlen(atrics_(1,i_)),gt,1,then
59:             nxpnum = nxpnum+1
60:             nxp_ent(1,nxpnum) = substr(atrics_(1,i_),1,36)
61:             nxp_kw(1,nxpnum) = substr(atrics_(1,i_),37,24)
62:             objtyp_(1) = substr(atrics_(1,i_),61,6)
63:             *if,objtyp_(1),eq,'(str)',then
64:                 nxp_typ(nxpnum) = 1
65:                 nxp_sval(1,nxpnum) = substr(atrics_(1,i_),68,20)
66:             *else
67:                 nxp_typ(nxpnum) = 0
68:                 nxp_val(nxpnum) = valchr(substr(atrics_(1,i_),68,20))
69:             *endif
70:         *endif
71:     *endif
72: *enddo
73: nxlist,2
74: *DEL,,PRM_

```

**NXLIST.MAC**

```

1:  !-----
2:  !   PADT                               Phoenix Analysis & Design
3:  !                                     Technologies
4:  !
5:  !=====
6:  !           * (480) 813-4884 * 1-800-293-PADT * (480) 813-4807 F *
7:  !           * www.PADTINC.com * info@padtinc.com *
8:  !=====
9:  !           File: nxlist.mac
10: !           File Type:   Macro
11: !
12: !   Description:
13: !       Lists out the entity names (arg1 = 1) or attribute information imported
14: !       from an NX file read (~ugin). The value are calculated by
15: !       nxrdtbl.mac and nxrdattr.mac
16: !
17: !       Input:
18: !           arg1: Option to print Entity Information (1) or Attribute Information (2)
19: !

```

**Continued...**

**NXLIST.MAC, Continued.**

```

20: !+++++
21: /nopr
22: *if,arg1,eq,1,then
23:   *dim,entnam_,string,10,4
24:   entnam_(1,NXPKP) = 'Keypoint'
25:   entnam_(1,NXPLN) = 'Line'
26:   entnam_(1,NXPAR) = 'Area'
27:   entnam_(1,NXPVL) = 'Volume'
28:   *msg,,
29:   -----
30:   *msg,,
31:   Entities from NX File
32:   *msg,,
33:   -----
34:   *msg,,
35:   _____Entity_____Type_____Number
36:   *do,i_,1,nxpentnum
37:     *msg,,nxp_ent_s(1,i_),entnam_(1,nxp_ent_n(i_,1)),nxp_ent_n(i_,2)
38:     %10s %10s %8d
39:   *enddo
40: *endif
41: *if,arg1,eq,2,then
42:   *msg,,
43:   -----
44:   *msg,,
45:   Parameters from NX File
46:   *msg,,
47:   -----
48:   *msg,,
49:   _____Entity_____Keyword_Type_Value_____String Value
50:   *msg,,
51:
52:   *do,i_,1,nxpnum
53:     *msg,,nxp_ent(1,i_),nxp_kw(1,i_),nxp_typ(i_),nxp_val(i_),nxp_sval(1,i_)
54:     %15s %15s %d %16.9g %10s
55:   *enddo
56:   *msg,,
57:   -----
58: *endif
59: *DEL,,PRM_
60: /go

```

**NXBLCM.MAC**

```

1: !-----
2: !   PADT                               Phoenix Analysis & Design
3: !                                     Technologies
4: !
5: !=====
6: !           * (480) 813-4884 * 1-800-293-PADT * (480) 813-4807 F *
7: !           * www.PADTINC.com * info@padtinc.com *
8: !=====
9: !           File: nxblbcm.mac
10: !           File Type:   Macro
11: !
12: !   Description:
13: !       Looks through an array of attributes from NX and locates any that start
14: !       with "CM" and builds components out of them.
15: !
16: !       Input:
17: !           The entity and attribute arrays from NXRDTBL.mac and NXRDATTR.mac must
18: !           exist.
19: !           No Args.

```

**Continued...**

## NXBLDCM.MAC, Continued.

```

20: !
21: !      Output:
22: !      Components are created
23: !      nxp_cm(n): An array containig the names of the created components
24: !      nxp_ncm(n): An array containing the number of entities in
25: !                  the component
26: !      nxcmnum: The number of components created
27: !
28: !+++++
29: /nopr
30: nxp_cm=
31: nxp_ncm=
32: *dim,entnam_,string,10,4
33: *dim,entsel_,string,10,4
34: *dim,nxp_cm,string,10,1000
35: *dim,nxp_ncm,,1000
36:
37: entnam_(1,NXPKP) = 'KP'
38: entnam_(1,NXPLN) = 'LINE'
39: entnam_(1,NXPAR) = 'AREA'
40: entnam_(1,NXPVL) = 'VOLU'
41: entsel_(1,NXPKP) = 'ksel'
42: entsel_(1,NXPLN) = 'lssel'
43: entsel_(1,NXPAR) = 'asel'
44: entsel_(1,NXPVL) = 'vsel'
45:
46: nxcmnum = 0
47: *do,i_,1,nxpnum
48:     *if,substr(nxp_kw(1,i_),1,2),eq,'CM',then
49:         ifound_=0
50:         *do,j_,1,nxcmnum
51:             *if,nxp_cm(1,j_),eq,nxp_sval(1,i_),then
52:                 ifound_ = 1
53:             *endif
54:         *enddo
55:
56:         *do,j_,1,nxpentnum
57:             *if,(nxp_ent_s(1,j_)),eq,nxp_ent(1,i_),then
58:                 *msg,,j_,nxp_ent_s(1,j_),ifound_
59:                 %d: %s %d
60:                 %entssel_(1,nxp_ent_n(j_,1))%,u,,all
61:                 *if,ifound_,eq,0,then
62:                     nxcmnum = nxcmnum+1
63:                     nxp_cm(1,nxcmnum) = nxp_sval(1,i_)
64:                     nxp_ncm(nxcmnum) = nxp_ncm(nxcmnum)+1
65:                     ifound_ = 1
66:                 *else
67:                     cmsel,s,nxp_sval(1,i_)
68:                 *endif
69:                 %entssel_(1,nxp_ent_n(j_,1))%,a,,nxp_ent_n(j_,2)
70:                 cm,nxp_sval(1,i_),entnam_(1,nxp_ent_n(j_,1))
71:             *endif
72:         *enddo
73:     *endif
74: *enddo
75: cmsel,all
76: allsel
77: cmlist,all
78: cmplot
79: *DEL,,PRM_
80: /go

```

Still enjoying this gorgeous source code? Yours can look just as good and you can have imbedded help and argument prompting. Try PeDAL at: [www.padtinc.com/pedal](http://www.padtinc.com/pedal)





# Phoenix Analysis & Design Technologies

## ABOUT PADT...

In the past we have finished up The Focus with a page we called "Shameless Advertising..." The truth was that the page was really only advertising PADT, and PADT related things. So, instead of doing advertising we thought we would just dedicate the final page to explaining who PADT is, what we do and how we can hopefully help you. And, to make sure you read it, we will try and stick something funny in. Want to know more? Call Stephen Hendry at 207-333-8780 or e-mail [steve.hendry@padtinc.com](mailto:steve.hendry@padtinc.com).



### Rapid Prototyping

Did you know that PADT, Inc. is one of the most respected Rapid Prototyping service bureaus in the world? For 16 years, since the founding of the company, PADT's RP group has been providing high quality prototype parts to customers around the world.

At that time the RP market has changed, and most small service bureaus have long since been gobbled up by larger firms or just plain gone out of business. But PADT has held their own by providing a high level of technical expertise and customer focus - similar to what has made the company a leader in the ANSYS world.

On site we have machines that use the three leading technologies: Stereolithography, Fused Deposition Modeling, and Selective Laser Sintering. We also have a CNC mill and lathe, hand finishing tools, and a full paint booth so we can create that perfect part that customers expect. The same team also provides soft tooling (rubber molding) and injection molding for when you need to move beyond prototypes. Next time you need a prototype, please consider having PADT provide a quote.

Visit the rapid manufacturing part of our web site ([www.padtinc.com/rm](http://www.padtinc.com/rm)) or e-mail [rp@padtinc.com](mailto:rp@padtinc.com) to learn more.



### PADT on the Web

[www.PADTINC.com](http://www.PADTINC.com) PADT's main website  
[www.PADTMedical.com](http://www.PADTMedical.com) Medical device development  
[www.DimensionSCA.com](http://www.DimensionSCA.com) A machine that PADT makes  
[www.PADTMarket.com](http://www.PADTMarket.com) A place to buy 3D Printers & Supplies  
[www.XANSYS.org](http://www.XANSYS.org) ANSYS User forum



### Join us on Facebook!

Search for PADT, Inc. and become a fan!

### Need ANSYS Help?

PADT can help in many different ways, here are a few:

- We hold training here or at your facility [<link>](#)
- Leverage our APDL knowledge with the APDL Guide [<link>](#)
- Consider one-on-one support through mentoring, a great way to get a quick start on something new [<link>](#)
- Attend a PADT Webinar [<link>](#)

### Humor:

Here are some Murphy's Laws of Engineering (see the whole list at : [www.badchickens.com/laws.htm](http://www.badchickens.com/laws.htm))

- Any wire or tube cut to length will be too short.
- After any machine or instrument has been fully assembled, extra components will be found on the bench.
- Any error that can creep in, will. It will be in the direction that will do the most damage.
- All constants are variable.
- In any given miscalculation, the fault will never be placed if more than one person is involved.
- Dimensions will always be expressed in the least usable terms.
- The probability of a dimension being omitted is directly proportional to its importance.
- It is impossible to make anything foolproof because fools are so ingenious.

