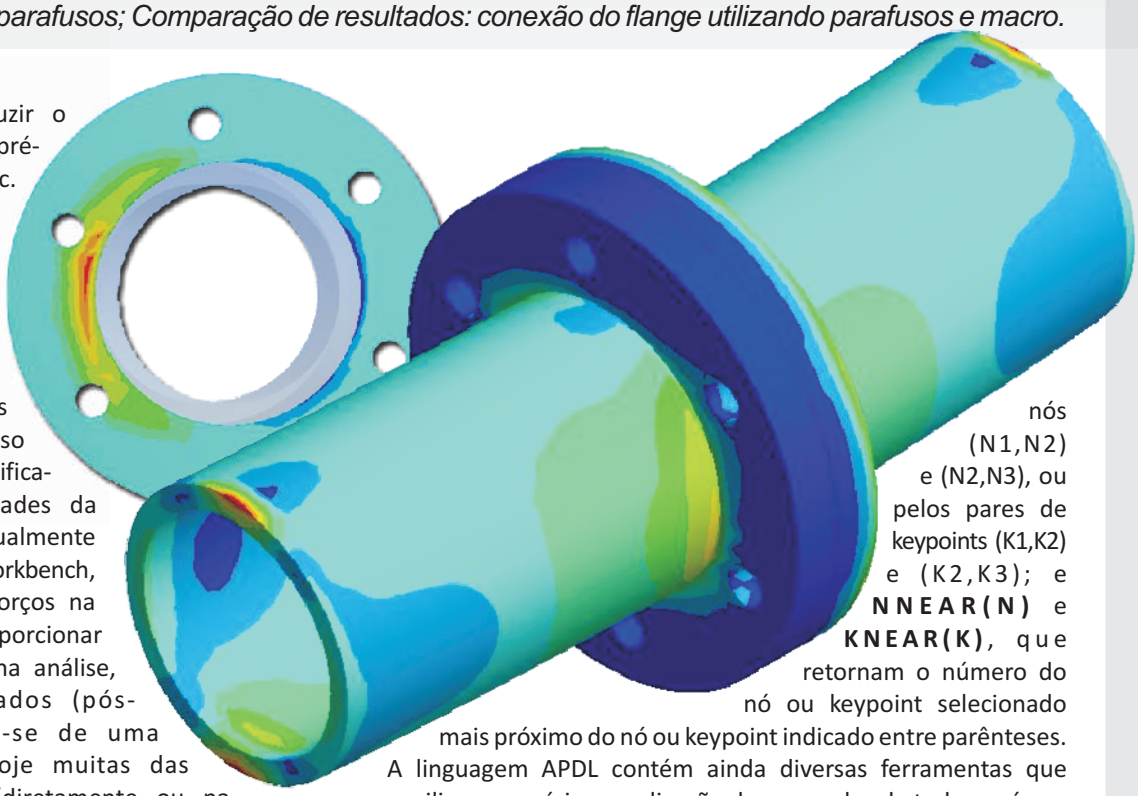
APDL: Linguagem fundamental na modelagem em ANSYS

Uma incursão pelos comandos e funções considerados indispensáveis para o domínio pleno do software; Macro para simulação da ação de parafusos; Comparação de resultados: conexão do flange utilizando parafusos e macro.



Com o objetivo de reduzir o tempo dispendido em pré-processamento, a ANSYS, Inc. tem investido muito no aprimoramento das interfaces de seus softwares, paralelamente ao aperfeiçoamento das rotinas que comandam alguns dos melhores "solvers" de elementos e volumes finitos disponíveis no mercado. No caso do ANSYS (*estrutural*), especificamente, todas as funcionalidades da interface clássica têm sido gradualmente transportadas para a interface Workbench, objetivando economia de esforços na montagem dos modelos e proporcionar mais tempo a ser dedicado na análise, efetivamente, dos resultados (pós-processamento). Tratando-se de uma mudança gradual, ainda hoje muitas das funcionalidades disponíveis (diretamente ou na forma de comandos) na interface clássica não sofreram as adaptações necessárias para figurarem na nova interface. Assim sendo, o conhecimento da linguagem conhecida como APDL (*ANSYS Parametric Design Language*) é um item importante na formação dos usuários do código ANSYS de elementos finitos. Aqui começa, então, a nossa incursão pelos comandos e funções considerados indispensáveis para o domínio pleno do software.

Similarmente a qualquer linguagem de programação (*PASCAL, C++, FORTRAN, etc*), a linguagem APDL possui recursos para declaração de variáveis (***DIM**), execução de laços (***DO** + ****ENDDO***), decisões lógicas (***IF** + ****ENDIF***), atribuição de valores (***SET**), operadores matemáticos *(+ adição, - subtração, * multiplicação, / divisão, ** exponenciação*) e funções paramétricas (*SIN(x), EXP(x), COS(x), TAN(x), LOG(x), SQRT(x), ABS(x), etc.*). Outra potencialidade dessa linguagem é a leitura de valores de variáveis ou parâmetros do modelo, que pode ser feita de duas formas: utilizando o comando ***GET**, que retorna o valor de um dado item e o armazena em uma nova variável, ou utilizando funções conhecidas como *inline get functions*, que podem ser utilizadas diretamente em operações. Dentre as principais funções deste tipo, podemos citar: **NDNEXT (N), ELNEXT(E), KPNEXT(K), LSNEXT(L), ARNEXT(A) e VLNEXT(V),** que retornam o número de nó, elemento, keypoint, linha, área ou volume selecionado consecutivo àquele indicado entre parênteses; **NX(N), NY(N), NZ(N), KX(K), KY(K) e KZ(K),** que retornam a coordenada x, y ou z do nó ou keypoint indicado entre parênteses; **NODE(X,Y,Z)** e **KP(X,Y,Z)**, que retornam o nó ou keypoint mais próximo das coordenadas indicadas entre parênteses; **DISTND (N1,N2)** e **DISTKP (K1,K2)**, que retornam a distância entre os dois nós ou keypoints indicados entre parênteses; **ANGLEN(N1,N2,N3)** e **ANGLEK (K1,K2,K3)**, que retornam o ângulo entre as duas linhas formadas pelos pares de nós (N1,N2) e (N2,N3), ou pelos pares de keypoints (K1,K2) e (K2,K3); e **NNEAR(N)** e **KNEAR(K)**, que retornam o número do nó ou keypoint selecionado mais próximo do nó ou keypoint indicado entre parênteses. A linguagem APDL contém ainda diversas ferramentas que auxiliam o usuário na aplicação de comandos de todas as áreas do software. Como exemplo, destacam-se os comandos que propiciam mudanças na visualização do modelo (**/VIEW**, **/ANGLE**) e aqueles que geram sistemas de coordenadas auxiliares ou modificam os já existentes (**CS**, **CSYS**, **DSYS**, **LOCAL**). Além de todos esses comandos gerais, também estão disponíveis ferramentas voltadas para as três principais áreas do software (pré-processamento, processamento e pós-processamento); é o que veremos com maior detalhe em seguida.

Na área de pré-processamento (**/PREP7**), dispõe-se de comandos para criação e seleção de entidades geométricas (keypoints - **K** e **KSEL**, linhas - **L**, **LFILLT** e **LSEL**, áreas - **A**, **AFILLT, AL** e **ASEL**, volumes - **V**, **VA** e **VSEL**, nós - **N** e **NSEL**, elementos - **E** e **ESEL**), cálculo de distâncias (entre nós - **NDIST**, keypoints - **KDIST**), criação (**CM**) e seleção (**CMSEL**) de componentes, aplicação de carregamentos (em nós - **F, corpos - BF, superfícies - SF**) e condições de contorno (em nós - **D, áreas - DA, keypoints - DK, linhas - DL**). Também é possível definir toda a estratégia de malha, incluindo geração (**ET**) e especificação (**TYPE**) dos tipos de elementos, definição de regiões com maior ou menor refino de malha (em áreas - **AESIZE**, linhas - **LESIZE**, keypoints - **KESIZE**, ou automático - **SMRTSIZE**), ajuste de parâmetros gerais de malha (comandos **MOPT**, **MSHAPE**, **MSHKEY**), informação das propriedades geométricas (**REAL**) e dos materiais (**MAT**) que compõem todos os elementos da estrutura e criação da malha propriamente dita (em áreas - **AMESH**, em volumes - **VMESH**, em linhas - **LMESH**).

Dentro da próxima área, responsável pelo processamento do modelo (**/SOLU**), são definidos o tipo de análise a ser efetuada (**ANTYPE**) e ajustadas as diversas opções de análise (comandos **EQSLV**, **NLGEOM**, **SOLCONTROL**). É também aqui que entram as instruções para configurar o "load step" (conjunto de

carregamentos para o qual uma solução é obtida - comandos **TIME**, **OUTRES**, **OUTPR**) e para efetivamente resolver o modelo montado (**SOLVE**). Caso desejado, é possível ainda definir os carregamentos e condições de contorno nesta área, ao invés de fazê-lo no pré-processamento.

Uma vez resolvido o modelo, os resultados podem ser acessados através das áreas (**/POST1**), "General Post Processor" e (**/POST26**), "Time History Post Processor". A diferença básica entre as duas está no tipo de resultado apresentado por cada uma: a área **/POST1** é utilizada para acessar resultados da análise em partes do modelo ou no modelo inteiro em um dado instante de tempo ou em uma dada frequência, enquanto a área **/POST26** é indicada para pós-processamento de resultados em regiões específicas do modelo em função do tempo ou da freqüência. As ações de interesse no pós-processamento dividem-se em três tipos: plotagem ou impressão de resultados na própria tela, na forma de distribuição em escala de cores nos elementos (obtidas em **/POST1**, utilizando os comandos **/SHOW** + **INRES** + **PLESOL**, para plotagem de resultados relativos a elementos**, /SHOW** + **INRES** + **PRESOL**, para impressão de resultados relativos a elementos, **/SHOW** + **INRES** + **PLNSOL**, para plotagem de resultados relativos a nós, ou **/SHOW** + **INRES** + **PRNSOL**, para impressão de resultados relativos a nós), geração de gráficos variável x variável dos resultados (obtida em **/POST26**, utilizando os comandos **/SHOW** + **XVAR** + **PLVAR**, para plotagem do gráfico, ou **/SHOW** + **XVAR** + **PRVAR**, para impressão dos dados constituintes do gráfico) e impressão dos resultados em arquivos de texto, para análise em outros *softwares* ou documentação (obtida tanto em **/POST1** quanto em **/POST26**, utilizando os comandos ***GET** + ***CFOPEN** + ***VWRITE** - que podem ser utilizados também em **/PREP7** e **/SOLU**). Caso se deseje armazenar os resultados obtidos em **/POST1**, pode-se ainda utilizar o comando **ETABLE**, que gera uma tabela com valores de variáveis requisitadas pelo usuário para posterior pós-processamento.

Um caso que requer especial atenção é o de análises dinâmicas explícitas, pois alguns parâmetros de todas as áreas devem ser definidos de forma diferente do que foi exposto anteriormente (pré-processamento, processamento e pós-processamento); para tanto, introduz-se uma série de novos comandos, dentre os quais destacam-se: **EDLOAD** (aplicação de carregamentos), **EDMP** (definição de propriedades de materiais) e **EDPART** (configuração de "parts" - grupos de elementos com características similares). Também é necessário definir parâmetros de contato entre as "parts" envolvidas no modelo, o que é feito utilizando-se os comandos **EDCONTACT** e **EDCGEN**, além de outros comandos secundários.

Por fim, um conjunto de comandos APDL particularmente útil diz respeito à operação de vetores e matrizes/tabelas - denominados "arrays" pelo ANSYS. A criação dos "arrays" é feita em dois passos: em primeiro lugar, utilizando o comando ***DIM** citado anteriormente, são definidos o tipo (vetor, tabela) e as dimensões do "array" em questão. Em seguida, são inseridos os valores, o que pode ser feito de diversas formas: inserindo comandos ***SET** para designação individual dos valores; preenchendo vetores (ou colunas, no caso de tabelas), pelo comando ***VFILL**, com valores especificados ou calculados pelo software; lendo os valores de arquivos externos, a partir dos comandos ***VREAD** e ***TREAD**; ou empregando o comando ***VGET**, que armazena os valores obtidos pela lógica do ***GET** diretamente em uma posição de um vetor definido pelo usuário.

Uma vez definidos os "arrays", pode-se lançar mão de diversas ferramentas que auxiliam em sua operação e caracterização, entre as quais incluem-se os comandos ***VSCFUN** (que determina propriedades do "array"), ***VFUN** (que opera uma função em um único vetor), ***VOPER** (que realiza operações entre vetores) e ***VITRP** (que gera um vetor a partir da interpolação de uma tabela). Além da geração de "arrays" para uso interno, também é possível criar arquivos contendo vetores (***VWRITE**) e matrizes (***MWRITE**) com parâmetros do modelo, para posterior utilização / visualização em outros softwares.

Apenas com os comandos expostos até aqui, já é possível implementar melhorias ou aprofundar o estudo da maioria dos problemas cotidianos modelados pela interface Workbench. Ainda assim, é importante ressaltar que abordamos apenas uma pequena parcela das potencialidades da linguagem APDL, o que torna claro que o domínio dessa linguagem é fundamental para o usuário que deseja extrair o máximo da ferramenta ANSYS.

Para ilustrar essa conclusão, é apresentado a seguir um exemplo de uso de uma macro escrita em APDL em um modelo ANSYS Workbench. Essa macro simula a ação de parafusos em um flange, tornando possível evitar a inclusão "física" dos parafusos no modelo; dessa forma, reduz o número de graus de liberdade e elimina o contato entre os parafusos e os furos do flange, reduzindo o custo computacional do modelo. Os resultados apresentam pequena divergência quantitativa daqueles obtidos por modelos tradicionais (com parafusos) numa escala local, ou seja, no nível de pressões de contato e tensões nas imediações, mas representam com extrema fidelidade a ação dos parafusos na junção de diferentes partes da estrutura, que é o que se deseja na maioria dos casos.

Macro para simulação da ação de parafusos

Para a utilização dessa macro, apenas quatro passos são necessários:

1 - Criar "Named Selections" para os furos (selecionando as superfícies do furo pertencentes às duas peças, como mostrado na Figura 1) com os nomes "paraf_1", "paraf_2", "paraf_3", ... , "paraf_n", onde "n" é o número total de parafusos do modelo.

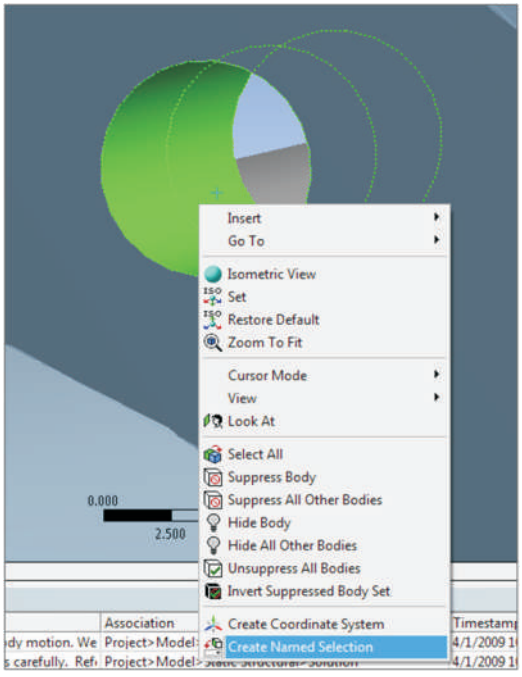


Figura 1: Criação das "Named Selections" necessárias para a inicialização da macro.

2 - Gerar o submenu "Commands" dentro do menu raiz correspondente ao tipo de análise que está sendo modelada. A Figura 2 mostra o procedimento a ser seguido: deve-se clicar com o botão direito em "Static Structural" e então selecionar "Insert --> Commands";

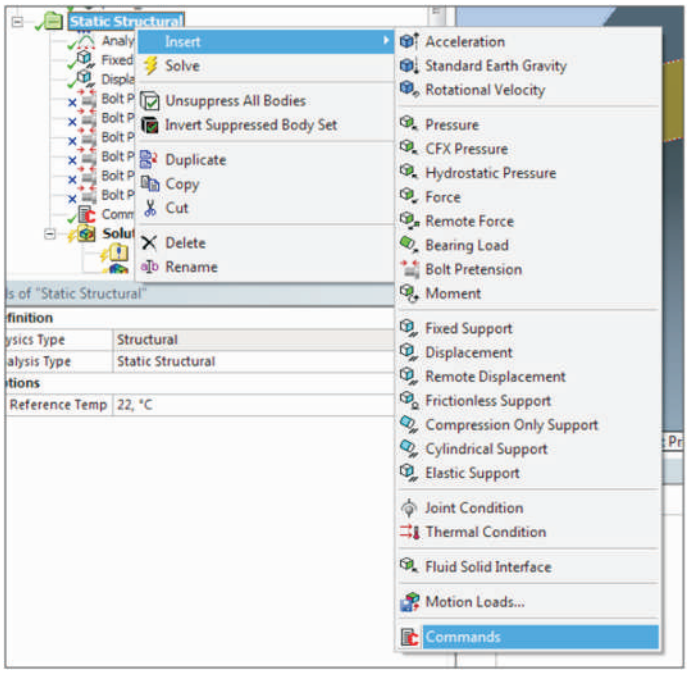


Figura 2: Geração do submenu "Commands", no qual a macro deve ser inserida.

3 - Copiar e colar a macro dentro da tela de diálogo em branco "Commands"; e

4 - Ajustar os valores de "Nparaf" e "forca" no início da macro, como indicado na Figura 3. “Forca” corresponde à força de aperto dos parafusos, cuja magnitude deverá ser correspondente ao sistema de unidades em vigor. Nparaf corresponde ao número de parafusos que está sendo criado através da macro. Esse número deverá coincidir com a quantidade de “named selections” do tipo “nparaf_1, nparaf_2, n_paraf_3, ... , n_paraf_#”.

```
!:::::::::::::::::::::::::::::
!Dados de entrada
!:::::::::::::::::::::::::::::
Nparaf=6
forca=10000
```

Figura 3: Parâmetros iniciais do modelo a serem ajustados na macro.

Procedendo dessa forma, ao se resolver o modelo ("Solution --> Solve") ele já levará em conta a presença dos parafusos "virtuais" dentro dos furos. Os passos anteriores descrevem como a macro funciona, de maneira prática. Agora veremos como ela funciona nos seus detalhes técnicos.

Inicialmente, é acionado o pré-processamento do software (comando /PREP7) e são definidas opções de visualização do modelo (comandos /NUMBER,1 e /PNUM,TYPE,1) que apenas são úteis em caso de análise do modelo na interface clássica. Em seguida, são inicializadas as variáveis de entrada do modelo, "Nparaf" e "forca", as quais, como foi exposto anteriormente, devem ser definidas pelo usuário no próprio submenu "Commands":

```
!:::::::::::::::::::::::::::::
!Parâmetros iniciais
!:::::::::::::::::::::::::::::

/PREP7
/NUMBER,1
/PNUM,TYPE,1

!:::::::::::::::::::::::::::::
!Dados de entrada
!:::::::::::::::::::::::::::::
Nparaf=6
forca=10000
```

O próximo passo é a inicialização dos tipos de elementos a serem utilizados no modelo (comando **ET**) com suas respectivas "key options" (comando **KEYOPT** - opções que ajustam alguns parâmetros do tipo de elemento em questão). Nesse ponto, utilizamos pela primeira vez o comando ***GET**, extremamente importante em qualquer programação em APDL, que irá aparecer de forma recorrente neste código. Aqui, ele é utilizado para retornar o maior número de "element type" definido até então, permitindo que se defina dois novos números para os "element types" a serem gerados pelos comandos **ET**, evitando colisão numérica com os demais tipos já presentes no modelo. Também é importante ressaltar o uso do comando **ALLSEL**, pelo qual todas as entidades do modelo são selecionadas para que o ***GET** faça sua busca:

```
!:::::::::::::::::::::::::::::
!Inicialização dos tipos de elementos
!:::::::::::::::::::::::::::::

allsel,all
*get,model_maxet,ETYP,0,num,max
ET,model_maxet+1,MPC184
KEYOPT,model_maxet+1,1,1
KEYOPT,model_maxet+1,2,1
ET,model_maxet+2,BEAM4
```

A linha seguinte representa o primeiro uso de outro comando fundamental em APDL: o *DO, que permite a execução de laços dentro do modelo. Esse primeiro laço será também o principal nessa macro, tendo como função selecionar um parafuso (entre 1 e "Nparaf") e para ele desenvolver toda a cadeia de comandos subseqüentes. No restante das linhas abaixo, destacam-se ainda os comandos CSYS (aqui utilizado para habilitar o sistema de coordenadas global), CMSEL (que seleciona todos os nós do parafuso em foco no momento) e ESLN (que seleciona todos os elementos que tenham vínculo com os nós do parafuso):

```
!:::::::::::::::::::::::::::::
!Início da rotina propriamente dita e *do que varre todos os parafusos do
!modelo
!:::::::::::::::::::::::::::::

*do,cont,1,Nparaf,1

allsel,all
csys,0
*get,maxnode,node,0,num,max
mastop1 = maxnode + 1
mastop2 = mastop1+1
CMSEL,S,paraf_%cont%,node
ESLN,S,0,all
```

Pela forma como selecionamos as "Named Selections" no Workbench, não é possível diferenciar os nós que pertencem aos furos de cada uma das peças que estão sendo unidas. Essa escolha foi feita para facilitar o trabalho do usuário da macro, mas torna necessário inserir no código algumas linhas responsáveis por fazer a distinção das peças da união. Isso é feito no bloco exibido a seguir, a partir de um laço gerado por um comando ***DO.** Esse laço é regulado por dois parâmetros captados por ***GET**s - no caso, os números mínimo e máximo de

elemento do parafuso na seleção atual - e se aplica apenas aos elementos selecionados anteriormente (comandos **CMSEL** + **ESLN**). Dentro deste laço, são utilizados três comandos ***IF**: o primeiro apenas impede que se leia o element type de um número de elemento que não esteja na seleção (o que retornaria um valor de zero e geraria problemas na leitura do ***GET** subsequente), o segundo exclui os "element types" entre 169 e 178, para impedir que elementos de contato entrem na lógica de separação, e o terceiro, contido dentro do segundo, é responsável por reger toda a lógica da distinção dos furos, levando em conta o "element type" de cada elemento (obtido utilizando o ***GET** presente dentro do primeiro ***IF**):

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Distinção dos furos das peças em contato
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

*get,el_max,elem,0,num,max
*get,el_min,elem,0,num,min
maxet=0
minet=0

*do,varr,el_min,el_max,1

        *get,el_typ,elem,varr,attr,type
        *if,el_typ,ne,0,then
                *get,et_name,etyp,el_typ,attr,enam
        *endif
        *if,et_name,lt,169,or,et_name,gt,178,then
            *if,el_typ,gt,maxet,then
                    minet=maxet
                    maxet=el_typ
                *elseif,el_typ,gt,minet,and,el_typ,lt,maxet
                minet=el_typ

                *endif
        *endif
*enddo
```

Feita a separação entre as superfícies (no caso, reduzidas a nós e elementos) de cada furo, podemos fazer o cálculo do centroide de cada uma das entidades. Iniciando pelo que chamaremos de "primeiro furo" de cada parafuso (escolhido como sendo aquele cujos elementos apresentam número de element type maior do que o do outro furo do par), notamos nas linhas abaixo a presença de um comando **CSYS** responsável por habilitar o sistema de coordenadas global, seguido de três comandos (**ESEL**, **NSLE** e **CMSEL**) responsáveis por selecionar os nós do primeiro furo. Essa seleção é salva em um componente, pelo comando **CM**, para facilitar sua posterior utilização. Após esse ponto, 8 comandos ***GET** são apresentados, responsáveis por obter o maior e menor número de nó da atual seleção e as coordenadas máxima e mínima em X, Y e Z. Enquanto os números de nó apenas serão utilizados mais adiante, as coordenadas obtidas são utilizadas logo em seguida, no cálculo das posições X, Y e Z do centroide, as quais são referência para a geração do nó "mastop1", o centroide propriamente dito, por um comando **N**:

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Cálculo do centroide do primeiro furo
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

csys,0

esel,s,type,,maxet
nsle,s,all
cmsel,r,paraf_%cont%,node
cm,furo_1,node

*get,maior1,node,0,num,max
*get,menor1,node,0,num,min
*get,xmin1,node,0,mnloc,x
*get,xmax1,node,0,mxloc,x
*get,ymin1,node,0,mnloc,y
*get,ymax1,node,0,mxloc,y
*get,zmin1,node,0,mnloc,z
*get,zmax1,node,0,mxloc,z

CentX1 = (xmax1+xmin1)/2
CentY1 = (ymax1+ymin1)/2
CentZ1 = (zmax1+zmin1)/2

n,mastop1,centx1,centy1,centz1
```

A mesma receita seguida para a geração do centroide do primeiro furo é adotada para o segundo furo. Apenas requer atenção a forma como é selecionado este furo (escolhido como sendo aquele de menor "element type") e a definição de variáveis com nomes diferentes dos adotados no caso anterior. O nó gerado para o centróide, de forma análoga ao que se viu para o primeiro furo, tem o nome de "mastop2":

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Cálculo do centroide do segundo furo
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

esel,s,type,,minet
nsle,s,all
cmsel,r,paraf_%cont%,node
cm,furo_2,node

*get,maior2,node,0,num,max
*get,menor2,node,0,num,min
*get,xmin2,node,0,mnloc,x
*get,xmax2,node,0,mxloc,x
*get,ymin2,node,0,mnloc,y
*get,ymax2,node,0,mxloc,y
*get,zmin2,node,0,mnloc,z
*get,zmax2,node,0,mxloc,z

CentX2 = (xmax2+xmin2)/2
CentY2 = (ymax2+ymin2)/2
CentZ2 = (zmax2+zmin2)/2

n,mastop2,centx2,centy2,centz2
```

Nesse ponto do modelo, os dois centroides do furo foram criados. Mais adiante, será feita a ligação de cada centroide com os nós de seu respectivo furo, e dos centroides entre si. Essa última conexão requer atenção especial, pois ela deve simular o corpo do parafuso que desejamos reproduzir, sendo necessário que propriedades como a área da seção transversal e os momentos de inércia do parafuso sejam definidas para o elemento que conecta os dois centroides. O próximo bloco de comandos, que dividiremos em três para facilitar o acompanhamento, faz justamente o cálculo dessas propriedades. Na primeira parte, exibida abaixo, observamos a seleção dos nós da circunferência de uma das faces circulares extremas do "parafuso" (ou seja, da seleção "paraf_x", x=1,2,...,Nparaf), no caso aquela localizada na extremidade do furo 2, e a geração de um novo número de nó para acomodar o centroide dessa circunferência. Ainda nesse bloco, nos deparamos com um comando novo, **LOCAL**, cuja função é gerar um novo sistema de coordenadas local (no contexto, um do tipo esférico com centro no centróide do furo 1). Graças a esse comando, podemos utilizar o ***GET** de uma forma bastante peculiar, como apresentado abaixo:

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Cálculo da área da seção transversal e momentos de inércia do
!parafuso
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

mastop_sup1=mastop2+1

CMSEL,S,paraf_%cont%, node
local,11,2,centx1,centy1,centz1

*get,max_dist,node,0,mxloc,x

nsel,r,loc,x,max_dist

*get,total_sup1,node,0,count
*get,maior_sup1,node,0,num,max
*get,menor_sup1,node,0,num,min

SomaX_sup1=0
SomaY_sup1=0
SomaZ_sup1=0
```

Agora que já utilizamos o sistema de coordenadas 11 (local) para o que era necessário, voltamos ao sistema de

coordenadas global utilizando o comando **CSYS**. O próximo passo é calcular o centroide da circunferência selecionada acima; isso é feito calculando a soma das coordenadas X, Y e Z de todos os nós da seleção, utilizando um ***DO**, e então dividindo cada uma das três somas pelo total de nós da seleção. Com as coordenadas do centroide em mãos, geramos um nó que o representa a partir do comando **N**. As quatro linhas restantes apenas reposicionam o centroide no plano da circunferência, evitando que malhas pouco refinadas interfiram na posição de centroide nesse plano:

```
csys,0
nodo_sup1=menor_sup1

*do,lasso,1,total_sup1

    SomaX_sup1 = SomaX_sup1 + NX(nodo_sup1)
    SomaY_sup1 = SomaY_sup1 + NY(nodo_sup1)
    SomaZ_sup1 = SomaZ_sup1 + NZ(nodo_sup1)
    nodo_sup1=ndnext(nodo_sup1)

*enddo

Centx_sup1 = Somax_sup1 / total_sup1
CentY_sup1 = SomaY_sup1 / total_sup1
CentZ_sup1 = SomaZ_sup1 / total_sup1

n,mastop_sup1,centx_sup1,centy_sup1,centz_sup1

cs,13,0,mastop_sup1,maior_sup1,menor_sup1

*get,centx,node,mastop2,loc,x
*get,centy,node,mastop2,loc,y

n,mastop_sup1_,centx,centy,0
```

Com as implementações desenvolvidas nos dois blocos anteriores, podemos finalmente calcular a área e o momento de inércia polar da seção, obtidos a partir do raio calculado como a distância entre o centroide "mastop_sup1" e um ponto qualquer da circunferência citada acima. Outros destaques neste bloco são a criação de um sistema de coordenadas esférico utilizando o comando **CS**, que será utilizado posteriormente para impedir a ligação de nós envolvidos no contato entre as peças com os centróides de seus furos, e a obtenção do material de uma das peças em contato, para que seja utilizado também como material do parafuso - o que é informado ao sistema pelo comando **MAT**:

```
cs,13+%cont%,2,mastop_sup1,maior_sup1,menor_sup1

raio=distnd(maior_sup1,mastop_sup1)
pi=3.141592
areast=pi*raio**2
itrans=(1/4)*pi*raio**4

nsel,s,node,,maior_sup1
esln,s,0,all
*get,elem_maior_sup1,elem,0,num,max
*get,mat_paraf_%cont%,elem,elem_maior_sup1,attr,mat
mat,mat_paraf_%cont%
```

Passamos agora para a ligação dos nós dos furos com seus respectivos centroides. Começando pelo primeiro furo, observamos em primeiro lugar a seleção dos nós do furo 1 (e a exclusão dos nós da região de contato, como abordado anteriormente). Em seguida, um comando ***GET** faz a contagem do total de nós selecionados, que irá limitar o laço ***DO** de geração dos elementos das conexões furo-centroide (o comando **E** é utilizado para efetivamente gerar esses elementos).

Finalmente, é importante ressaltar a ordem para o uso do tipo de elemento MPC184, definido no começo da macro, por meio do comando **TYPE**:

```
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Ligação dos nós do primeiro furo com seu centroide
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


csys,13+%cont%

cmsel,s,furo_1,node
*get,x1_cont,node,0,mnloc,x
nsel,u,loc,z,z1_cont
*get,total1b,node,0,count
type,model_maxet+1
nodo1=menor1

*do,lasso,1,total1b,1

   e,nodo1,mastop1
   nodo1=ndnext(nodo1)

*enddo
```

Novamente, a mesma receita seguida para a ligação furo-centróide do primeiro furo é adotada para o segundo furo, apenas mudando as variáveis envolvidas:

```
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Ligação dos nós do segundo furo com seu centroide
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


cmsel,s,furo_2,node
*get,x2_cont,node,0,mxloc,x
nsel,u,loc,z,z2_cont
*get,total2b,node,0,count
type,model_maxet+1
nodo2=menor2

*do,lasso,1,total2b,1

    e,nodo2,mastop2
    nodo2=ndnext(nodo2)

*enddo
```

De modo análogo ao que foi desenvolvido para a ligação furo-centroide, iremos agora conectar os centróides entre si. Destaque para os comandos **REAL** e **R**, utilizados na definição e utilização das "real constants" (definição de parâmetros geométricos complementares) desse tipo de elemento:

```
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Ligação entre os centroides e definição das constantes reais do elemento
!de ligação
!:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::


*get,maxrc,RCON,0,num,max


nsel,s,node,,mastop1
nsel,a,node,,mastop2

real,maxrc+1

type,model_maxet+2
r,maxrc+1,areast,itrans,itrans,raio,raio

e,mastop2,mastop1
```

Finalizando o ***DO** que varre todos os parafusos do modelo, temos o bloco abaixo, que aplica pré-carga aos parafusos. Destaques para o uso do comando ***GET,** auxiliando na definição dos vetores mastop2 - mastop1 e mastop1- mastop2, para os comandos **NANG** e **NROTAT** (que alteram o sistema de coordenadas nodal dos nós selecionados) e para o uso do comando **F** para aplicação da força nos centroides:

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Aplicação das forças nos nós de controle
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

nsel,s,node,,mastop1
nsel,a,node,,mastop2
nsel,a,node,,mastop_sup1

csys,0

*get,x_mastop1,node,mastop1,loc,x
*get,y_mastop1,node,mastop1,loc,y
*get,z_mastop1,node,mastop1,loc,z
*get,x_mastop2,node,mastop2,loc,x
*get,y_mastop2,node,mastop2,loc,y
*get,z_mastop2,node,mastop2,loc,z

x12=x_mastop2-x_mastop1
y12=y_mastop2-y_mastop1
z12=z_mastop2-z_mastop1

nang,mastop1,,,,1,1,-((x12+y12)/z12),x12,y12,z12

x21=x_mastop1-x_mastop2
y21=y_mastop1-y_mastop2
z21=z_mastop1-z_mastop2

nang,mastop2,,,,1,1,-((x21+y21)/z21),x21,y21,z21

f,mastop1,fz,forca*((z_mastop2-z_mastop1)/abs(z_mastop1-
z_mastop2))

f,mastop2,fz,forca*((z_mastop1-z_mastop2)/abs(z_mastop1-
z_mastop2))

csys,0
allsel,all
nrotat,all
```

Finalmente, chegamos ao encerramento da macro, com a conclusão do ***DO** principal e a inicialização do módulo de processamento pelo comando **/SOLU**:

```
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
!Final do *do que varre os parafusos e encerramento da macro
!::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

*enddo
finish
/SOLU
```

Comparação de resultados: conexão do flange utilizando parafusos e macro

As principais vantagens de se utilizar a macro que simula a ação dos parafusos, ao invés de efetivamente modelá-los, já foram apresentadas anteriormente: redução de custo computacional e maior facilidade na modelagem do problema. Entretanto, antes de aceitar essa opção para uso na prática, é fundamental validar a macro, ou seja, comparar os resultados obtidos utilizando as linhas de comando daqueles provenientes dos modelos que incluem "fisicamente" os parafusos. Dessa forma, as linhas a seguir apresentam uma comparação entre os resultados obtidos pelos dois métodos em uma situação de flexão de um flange que conecta dois tubos.

As duas figuras a seguir exibem a distribuição de tensões equivalentes de von Mises no conjunto em questão; a Figura 4 mostra os resultados obtidos utilizando a macro, enquanto a Figura 5 apresenta os resultados utilizando parafusos na modelagem. Fica claro que, a menos de pequenas discrepâncias em uma região do tubo próxima ao flange, a distribuição de tensões na estrutura é muito semelhante entre os casos, o que se comprova pela concentração de tensões nas regiões extremas dos tubos e pelo desenho das curvas de níveis de tensão nas faces do flange e no restante das faces cilíndricas dos tubos. Logo, pode-se concluir pelas imagens que a macro consegue simular com sucesso a ação dos parafusos na junção do flange em um nível mais geral.

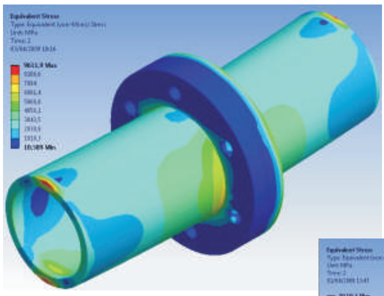


Figura 4: Tensão equivalente de von Mises na simulação utilizando a macro.

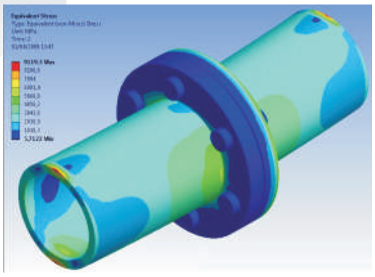


Figura 5: Tensão equivalente de von Mises na simulação utilizando parafusos.

Assim, passamos agora à análise das pressões de contato no flange, resultantes da ação dos parafusos em uma escala mais local.Neste nível de análise, começamos a perceber algumas diferenças nos resultados. De fato, nota-se elevada concentração de pressões de contato no entorno de dois furos no caso que utiliza a macro (Figura 6), que chamaremos de furos 1 e 2, o que não ocorre no caso em que se consideram os parafusos (Figura7). Isso ocorre porque a forma como os dois casos transmitem a pré-carga dos parafusos para o flange é distinta: quando se utiliza parafusos na modelagem do problema, é a cabeça de cada uma dessas peças quem pressiona as faces do flange, proporcionando a fixação. No caso da macro, isso é feito de outra forma: os próprios nós que compõem os furos são carregados na direção axial do furo, comprimindo o flange. Fora dessa região, porém, os resultados são bastante compatíveis, com destaque para as distribuições de pressões na região compreendida entre os furos 1 e 2 citados anteriormente e na região exatamente oposta, em azul mais escuro nas figuras, praticamente idênticas nos dois casos.

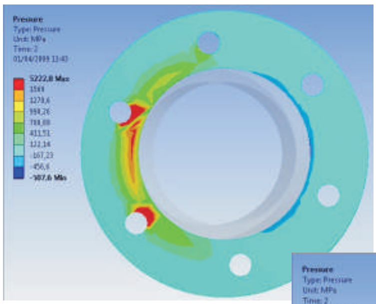


Figura 6: Pressão de contato na simulação utilizando a macro.

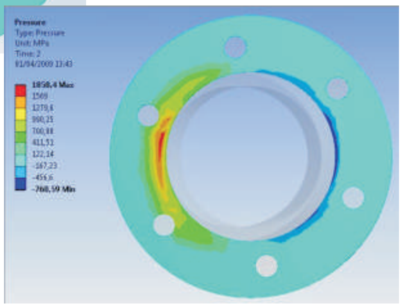


Figura 7: Pressão de contato na simulação utilizando parafusos.

A análise conjunta dos resultados obtidos nas escalas geral e local do problema permite concluir que o uso da macro desenvolvida é válido na simulação de junções por parafusos, apresentando resultados coerentes com aqueles observados em modelos mais tradicionais.