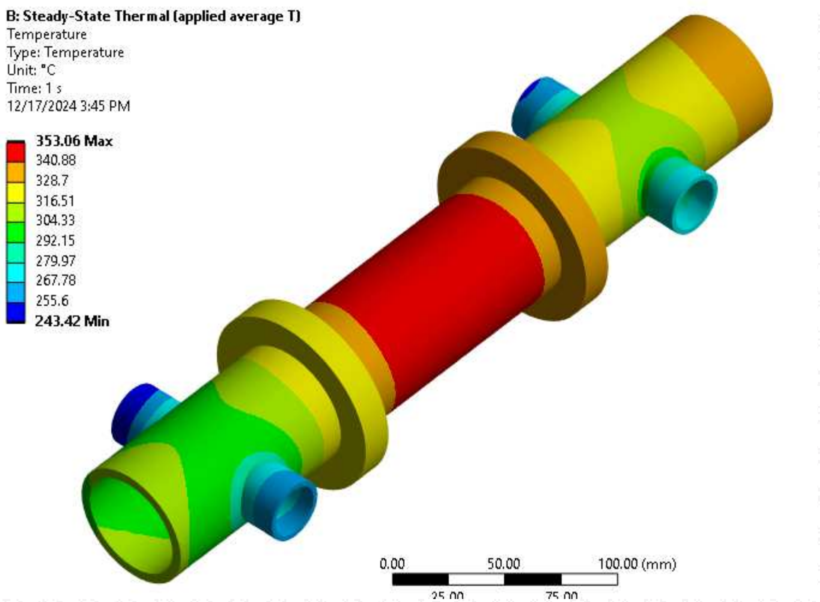
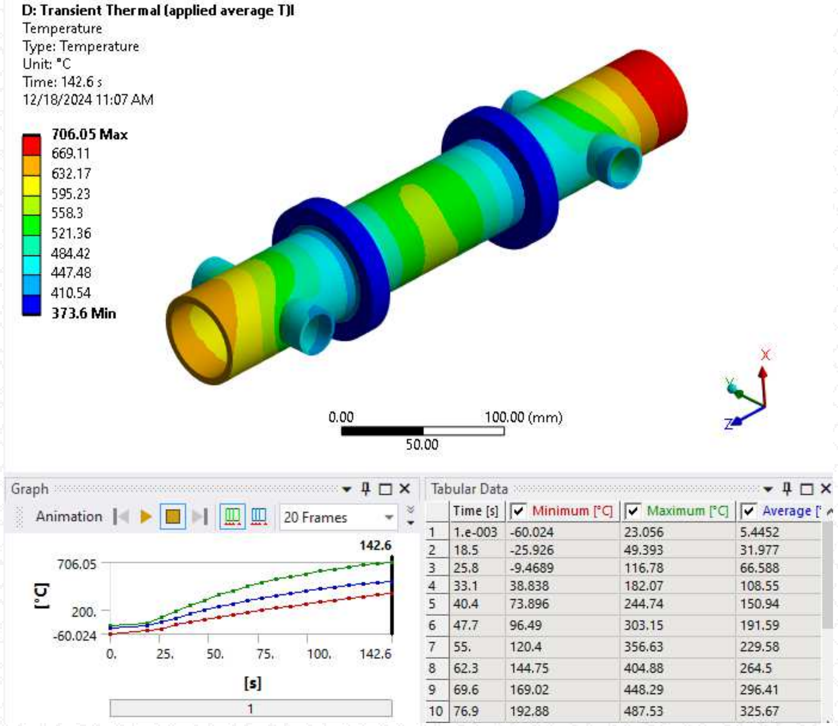
# Applying an Average Temperature in Ansys




| | Time [s] | Minimum [°C] | Maximum [°C] | Average [° |
|---|---|---|---|---|
| 1 | 1.e-003 | -60.024 | 23.056 | 5.4452 |
| 2 | 18.5 | -25.926 | 49.393 | 31.977 |
| 3 | 25.8 | -9.4689 | 116.78 | 66.588 |
| 4 | 33.1 | 38.838 | 182.07 | 108.55 |
| 5 | 40.4 | 73.896 | 244.74 | 150.94 |
| 6 | 47.7 | 96.49 | 303.15 | 191.59 |
| 7 | 55. | 120.4 | 356.63 | 229.58 |
| 8 | 62.3 | 144.75 | 404.88 | 264.5 |
| 9 | 69.6 | 169.02 | 448.29 | 296.41 |
| 10 | 76.9 | 192.88 | 487.53 | 325.67 |

Alex Grishin, PhD
12/20/24

# Background and Motivation

- A PADT customer recently asked (paraphrasing):

  "... I am wondering how to [constrain] an average temperature over a surface? My goal is not to enforce a constant temperature to a surface but to make sure that average temperature is [constrained to a certain value]..."

- But why might someone want to do this?

- We can point to at least a couple of similar scenarios that might warrant application of an average boundary condition:
  - Mapping data from a very coarse experimental data set onto smaller features (i.e. temperature data from contactless temperature measurement devices such as pyrometers)
  - Mapping data from a very coarse numerical model

- In both examples, applying discrete temperatures as averages over regions of coarse resolution would in general provide more conservative fine estimates (local maxima may still exceed the average. And such maxima may exceed those estimated by interpolation)

- public domain wikipedia image

# Background

- We feel that such a well-posed but ‘odd’ question deserves a blog post. Especially because the solution is so simple
- The simplicity of the solution stems from how constraints are applied in a finite element model
- And this deserves a digression of its own. In our previous blog article (in appendix A found in the link), we demonstrated how finite element constraints can be used to model a fixed-fixed beam by coupling two cantilevers tip-to-tip.

$$v_2 - v_3 = 0 \rightarrow [1 \quad 0 \quad -1 \quad 0]\begin{Bmatrix} v_2 \\ \theta_2 \\ v_3 \\ \theta_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

- And:

$$\theta_2 - \theta_3 = 0 \rightarrow [0 \quad 1 \quad 0 \quad -1]\begin{Bmatrix} v_2 \\ \theta_2 \\ v_3 \\ \theta_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

ation Work

- from slide 37 (Appendix A)

- Unfortunately, a full treatment of finite element constraint equations is beyond the scope of this article
- In any case, we can’t do better than the explanation found here and here. Suffice it to say that one starts with equations relating the solution degrees of freedom (in the current context: temperature) and adjusts the system equations accordingly (depending on which method is being used)

# Preliminaries: Defining Constraints in APDL

- The Ansys Mechanical APDL documentation provides a good description of how this works in Ansys (which we won't go into here)
- For this article, we're more interested in how such equations are defined by the user and implemented in the user interface.
- A description of this interface may also be found in the MAPDL documentation (reprinted below. We'll return to this later).

**CE**

**CE**, *NEQN*, *CONST*, *NODE1*, *Lab1*, *C1*, *NODE2*, *Lab2*, *C2*, *NODE3*, *Lab3*, *C3*
**Defines a constraint equation relating degrees of freedom.**

## Notes

Repeat the **CE** command to add additional terms to the same equation. To change only the constant term, repeat the command with no node terms specified. Only the constant term can be changed during solution, and only with the **CECMOD** command.

Linear constraint equations may be used to relate the degrees of freedom of selected nodes in a more general manner than described for nodal coupling [**CP**]. The constraint equation is of the form:

$$\text{Constant} = \sum_{I=1}^{N} (\text{Coefficient}(I) * U(I))$$

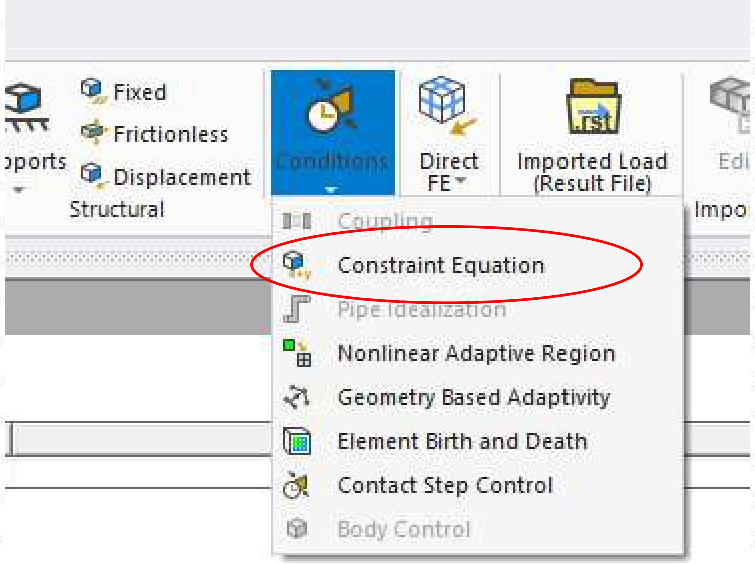
where U(I) is the degree of freedom (displacement, temperature, etc.) of term (I). The following example is a set of two constraint equations, each containing three terms:

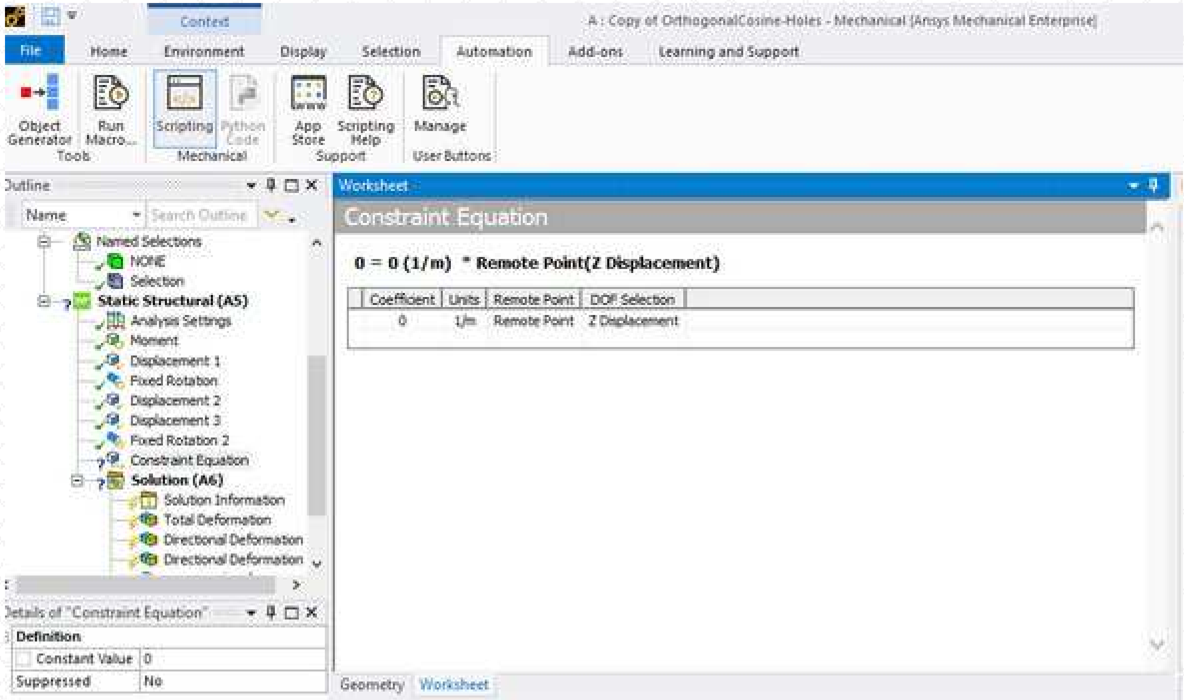
0.0 = 3.0* (1 UX) + 3.0* (4 UX) + (-2.0)* (4 ROTY)

2.0 = 6.0* (2 UX) + 10.0* (4 UY) + 1.0* (3 UZ)

# Preliminaries: Defining Constraints in Mechanical

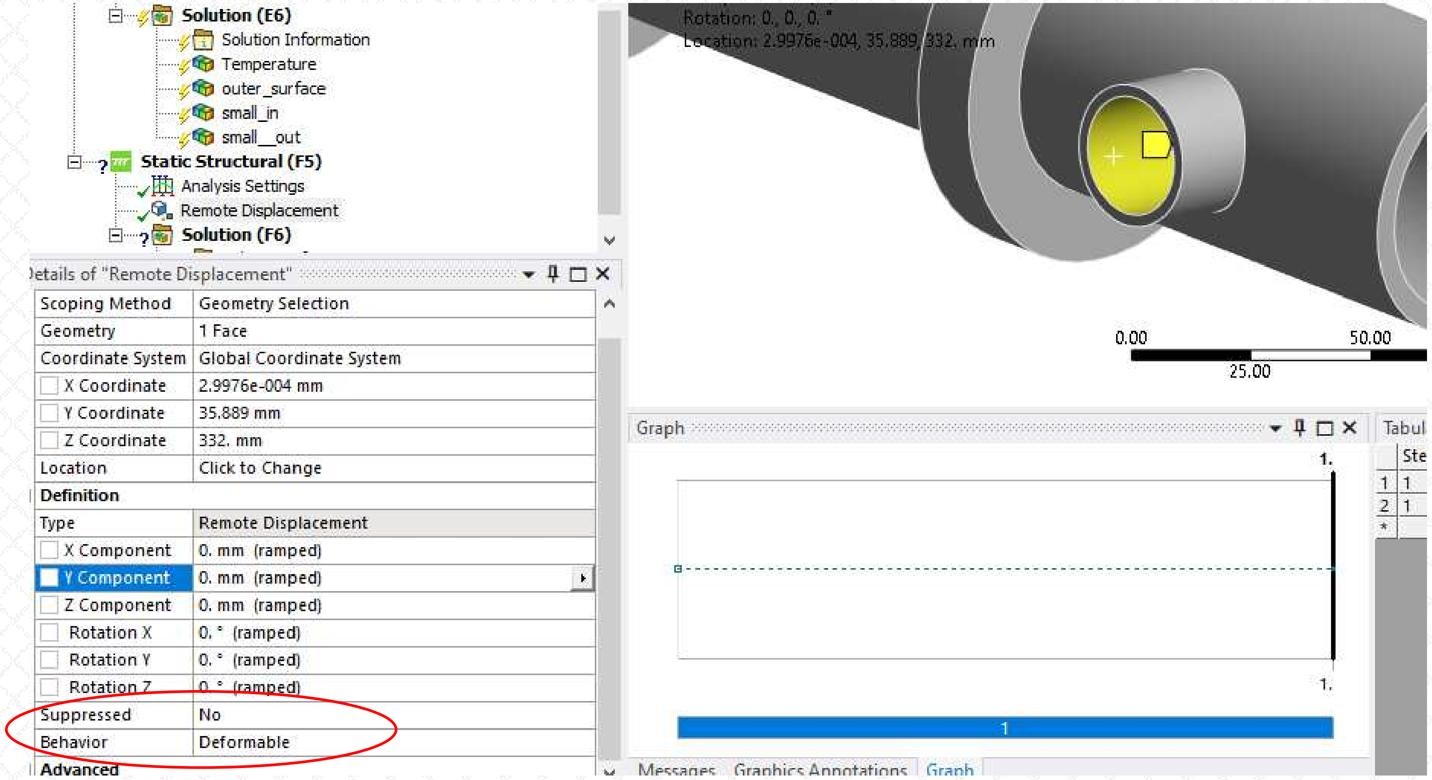
- In Ansys Mechanical, users can only apply constraint equations through remote points (through the ‘Constraint Equation’ object under the Environment tab)
- But this is only available for Static or Transient Structural analyses (and is limited to a specific type of constraint as shown below)
- If this blog post achieves nothing else, we’d like to emphasize to users that these are limitations of Workbench only. A robust interface for constructing constraint equations exists for all analysis types supported by MAPDL in that environment. We’ll demonstrate that in this article
- The previous slide shows the APDL scripting interface for constraint equations (the CE command)



# Preliminaries: RBE3 Constraints

- However, there is one type of ‘average’ constraint that CAN be applied in the Workbench Mechanical interface, but again only in static or transient structural analysis
- A brief exploration of this functionaltiy will help readers understand the solution we will offer shortly:
- Whenever one applies a remote load or displacement with the ‘Behavior’ set to ‘Deformable’ (the default), the load is applied in an average sense to the scoped geometry



# Preliminaries: RBE3 Constraints

- If we open the corresponding ds.dat file and search for ‘RBE3’, we find the following lines of APDL which create the constraint
- This type of constraint is implemented in Ansys as a node-to-surface contact pair, which is essentially just bonded contact with an RBE3 constraint applied (instead of rigidly connecting two surfaces, they are related through the constraint equation. Within the context of contact elements, Ansys refers to these as MPC constraints)

```
/com,*********** Create Remote Point "Internal Remote Point" ***********
! -------- Remote Point Used by "Remote Displacement" --------
*set,tid,3
*set,cid,2
et,cid,174
et,tid,170
keyo,tid,2,1              ! Don't fix the pilot node
keyo,tid,4,111111
keyo,cid,12,5             ! Bonded Contact
keyo,cid,4,1              ! Deformable RBE3 style load
keyo,cid,2,2              ! MPC style contact
eblock,10,,,57
(15i9)
    10083        2        2        2        0      311      312     1453     1453      45
    10084        2        2        2        0      312      313     1454     1454      45
```

- So, what exactly is it doing? What IS an RBE3 constraint?

# RBE3 Constraints

- The APDL interface offers an RBE3 command
- As shown below, it ties a single node (this would correspond to the remote point in Mechanical) to a set of 'slave' nodes (the scoped geometry in Mechanical) with constraint equations that calculate the average nodal values (multiplied by an optional weighting factor which defaults to 1).

**RBE3**

**RBE3**, *Master*, *DOF*, *Slaves*, *Wtfact*

**Distributes the force/moment applied at the master node to a set of slave nodes, taking into account the geometry of the slave nodes as well as weighting factors.**

PREP7: Constraint Equations

Compatible Products: – | Pro | Premium | Enterprise | Ent PP | Ent Solver | –

**RBE3** creates constraint equations such that the motion of the master is the average of the slaves. For the rotations, a least-squares approach is used to define the "average rotation" at the master from the translations of the slaves. If the slave nodes are colinear, then one of the master rotations that is parallel to the colinear direction can not be determined in terms of the translations of the slave nodes. Therefore, the associated moment component on the master node in that direction can not be transmitted. When this case occurs, a warning message is issued and the constraint equations created by **RBE3** are ignored.

- the term 'RBEx' comes from NASTRAN
- It stands for 'Rigid-Body-Element
- There are two types: RBE2 (rigid constraints) and RBE3 (average constraints)

- But here again, Ansys has restricted the applicable degrees of freedom with those of static and transient structural analyses

*DOF*

Refers to the master node degrees of freedom to be used in constraint equations. Valid labels are: UX, UY, UZ, ROTX, ROTY, ROTZ, UXYZ, RXYZ, ALL

- No temperature DOFs (?)

# Constructing A Thermal RBE3-Type Constraint

- At this point, we know we need something like an RBE3 constraint (but not using the RBE3 command because that doesn't support temperature DoFs). Reviewing the CE command's options, it is apparent that we don't need to constrain save nodes to a master. We just need to define an equation that involves all the nodes
- So we'll construct our own equivalent average (RBE3-type) constraint using the robust, universally appicable CE command (with no restrictions on degrees of freedom)
- Fortunately, the APDL interface makes it quite clear how this can be done

**CE**, *NEQN*, *CONST*, *NODE1*, *Lab1*, *C1*, *NODE2*, *Lab2*, *C2*, *NODE3*, *Lab3*, *C3*

**Defines a constraint equation relating degrees of freedom.**

PREP7: Constraint Equations

$$\text{Constant} = \sum_{I=1}^{N} \left(\text{Coefficient}(I) * U(I)\right)$$

- With this single command, users can connect up to three degrees of freedom on a single line of code
- The command may simply be repeated to add additional degrees of freedom

*NEQN*

Set equation reference number:

**n** — Arbitrary set number.

**HIGH** — The highest defined constraint equation number. This option is especially useful when adding nodes to an existing set.

**NEXT** — The highest defined constraint equation number plus one. This option automatically numbers coupled sets so that existing sets are not modified.

The default value is HIGH.

*CONST*

Constant term of equation.

*NODE1*

Node for first term of equation. If -*NODE1*, this term is deleted from the equation.

*Lab1*

Degree of freedom label for first term of equation. Structural labels: UX, UY, or UZ (displacements); ROTX, ROTY, or ROTZ (rotations, in radians). Thermal labels: TEMP, TBOT, TE2, TE3, . . ., TTOP (temperature). Electric labels: VOLT (voltage). Magnetic labels: MAG (scalar magnetic potential); AZ (vector magnetic potential). Diffusion label: CONC (concentration).

*C1*

Coefficient for first node term of equation. If zero, this term is ignored.

*NODE2, Lab2, C2*

Node, label, and coefficient for second term.

*NODE3, Lab3, C3*

Node, label, and coefficient for third term.

# Constructing A Thermal RBE3-Type Constraint

$$\text{Constant} = \sum_{I=1}^{N}(\text{Coefficient}(I)*U(I)) \Rightarrow \bar{T} = \sum_{i=1}^{n}\frac{1}{n}T(i)$$

- To apply an average temperature, $\bar{T}$ we would do this...

- Since we can repeat the CE command to add as many degrees of freedom as we want, a compact way of doing this for arbitrarily many nodes is to add one degree of freedom at a time within a loop.
- The syntax looks like this:

```
*do,i,1,n
   ce,cenum,aveT,nd,temp,1/n
*enddo
```

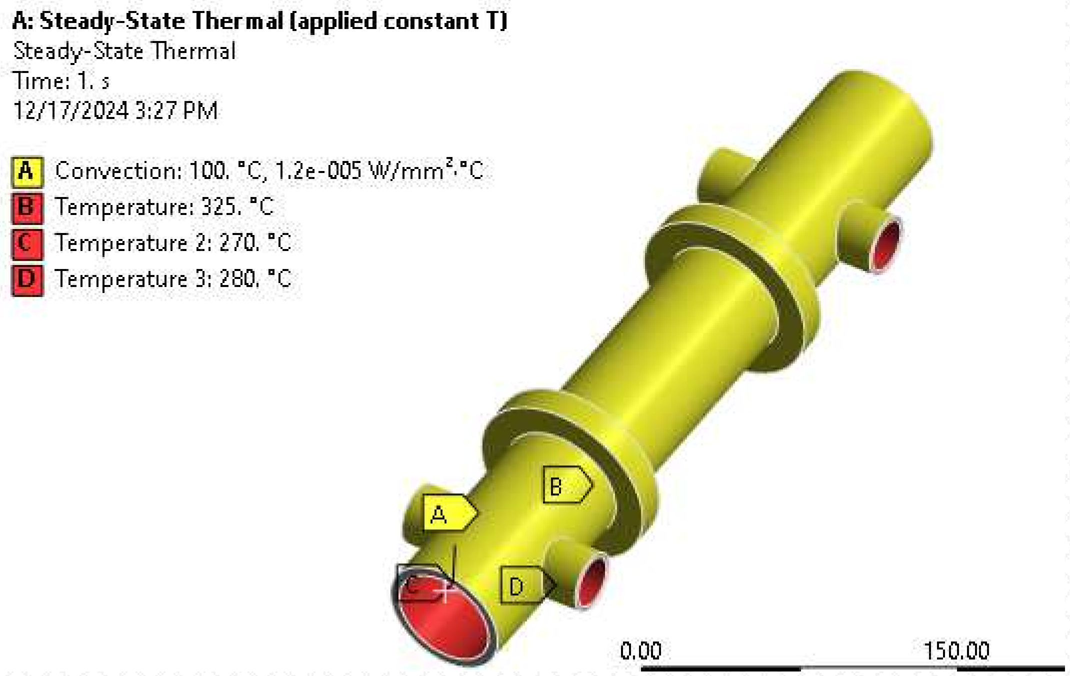
**CE**, *NEQN, CONST, NODE1, Lab1, C1,*

  - n: the number of nodes to be constrained
  - cenum: the number ID of this constraint equation
  - aveT: the target average temperature
  - temp: (temperature DOF label)
  - 1/n: the coefficient to be applied to each degree of freedom to enforce an arithmetic mean

- And that’s all there is to it. Let’s see how to implement this in Mechanical as a command object

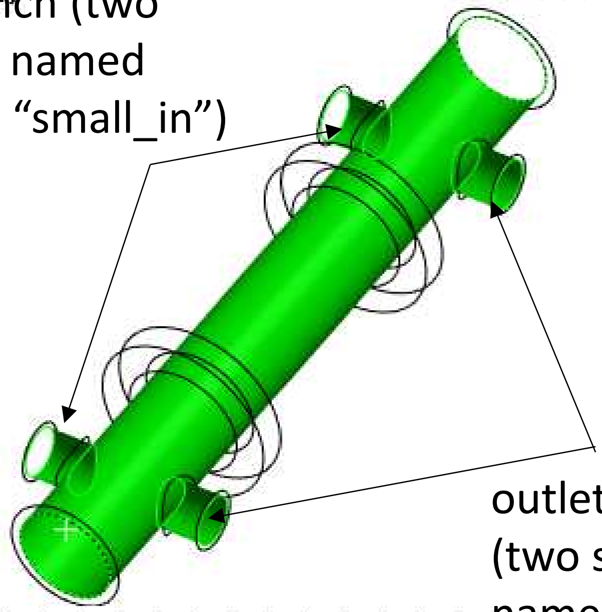
# Example 1: Steady-State Heat Transfer

- Our example geometry consists of a steel pipe with two intersecting branch flows




main pipe (one surface: named selection “osurf”)

inlet branch (two surfaces: named selection “small_in”)



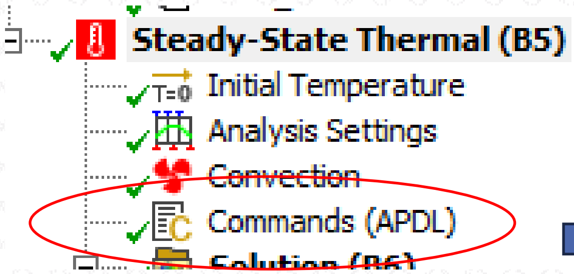
outlet branch (two surfaces: named selection “small_out”)

- Inlet branch: 270°C
- outlet branch: 280°C
- main branch; 325°C

- All outer surfaces have convection coefficient of 1.2e-5 W/mm^2 °C (12 W/m^2 °C) @ 100°C ambient (colored yellow above left)

# Example 1: Steady-State Heat Transfer

- After defining the convection coefficient in the usual way, we insert an APDL Commands object to apply the average temperatures
- The full code is summarized below

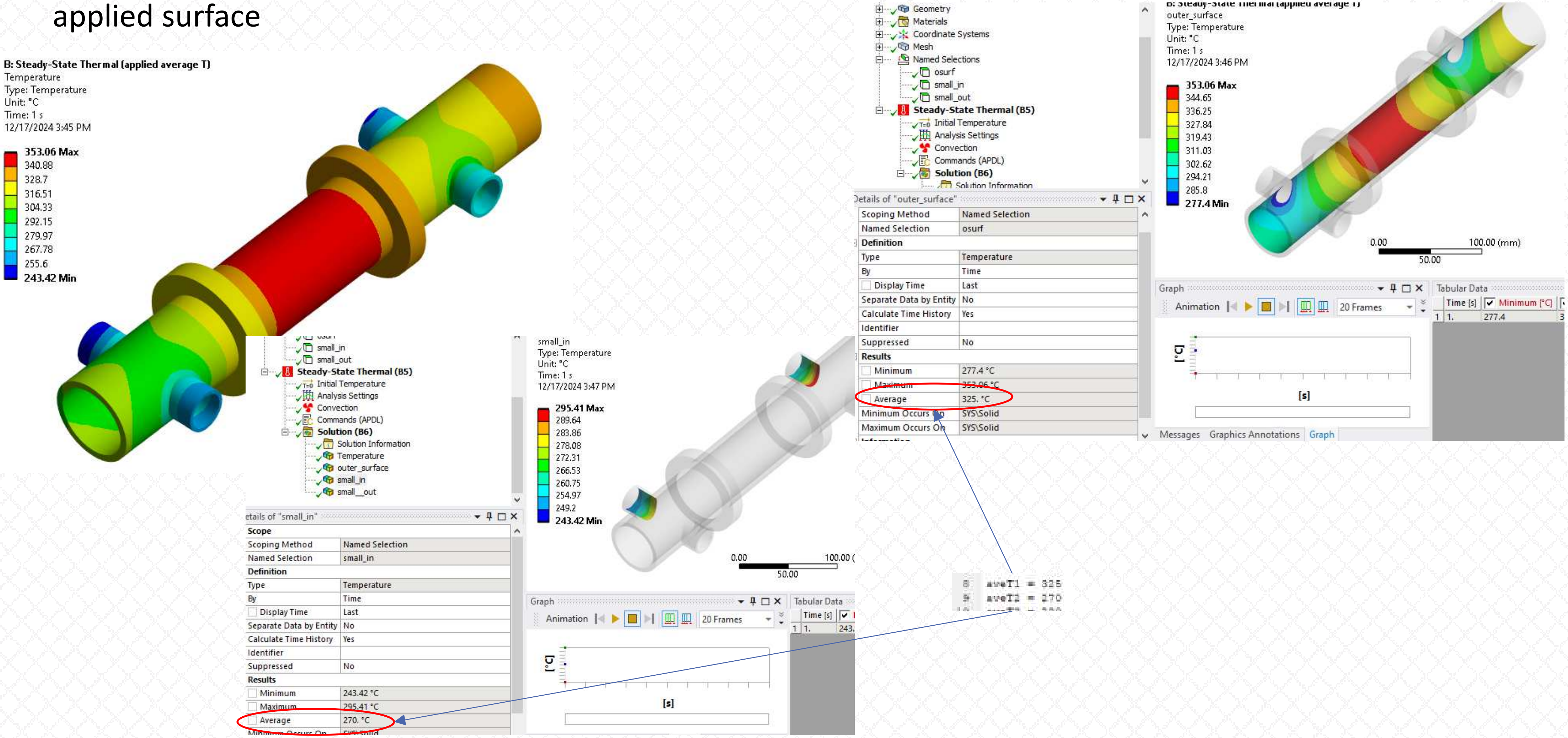



```
aveT1 = 325
aveT2 = 270
aveT3 = 280

/PREP7
*get,cenmax,ce,,max      !get the maximum CE number already defined
cmsel,s,osurf    !select outer surface
*get,nn,node,,count
nd=ndnext(0)     !start node number counter
*do,i,1,nn
   ce,cenmax+1,aveT1,nd,temp,1/nn     !apply aveT1 to osurf
   nd=ndnext(nd)     !increment node counter
*ENDDO

cmsel,s,small_in
*get,nn,node,,count
nd=ndnext(0)
*do,i,1,nn
   ce,cenmax+2,aveT2,nd,temp,1/nn     !apply AveT3 to small_in
   nd=ndnext(nd)
*ENDDO

cmsel,s,small_out
*get,nn,node,,count
nd=ndnext(0)
*do,i,1,nn
   ce,cenmax+3,aveT3,nd,temp,1/nn     !apply AveT2 to small_out
   nd=ndnext(nd)
*ENDDO

allsel,
/SOLU
```

- We have created named selections out of each applied surface in order to select them in APDL
  - main branch: "osurf"
  - inlet branch: "small_in"
  - outlet branch: "small_out"

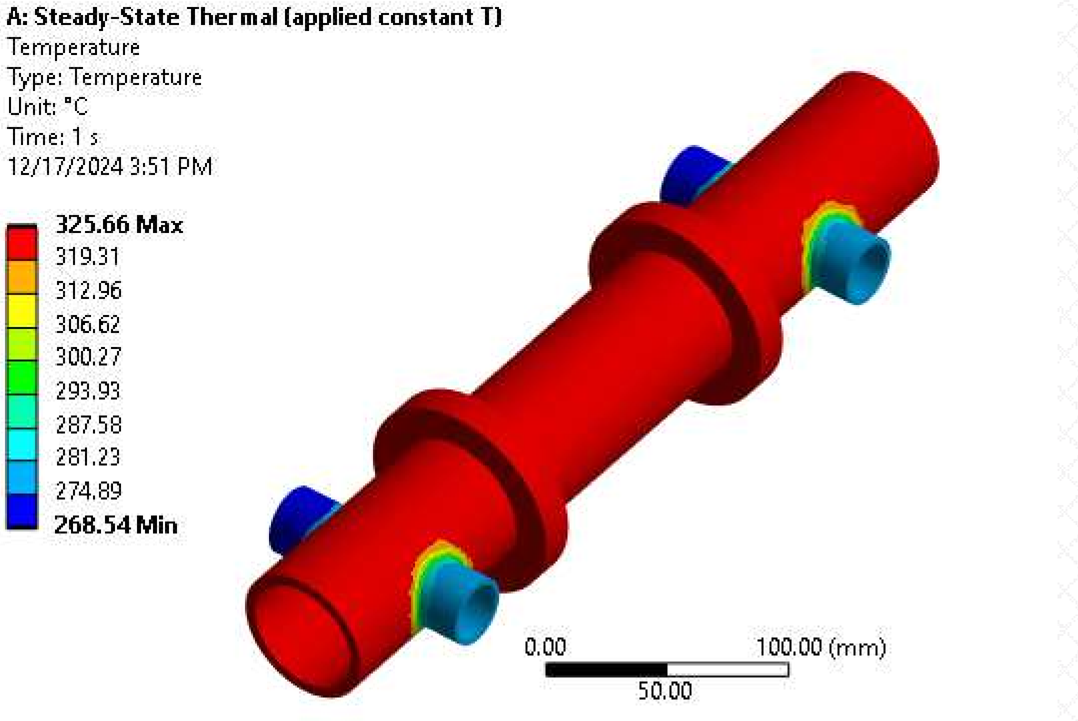
# Example 1: Steady-State Heat Transfer

- We can check that the average temperaure is indeed being applied by reviewing the ‘Average’ field of the output contour plot (below) for each applied surface

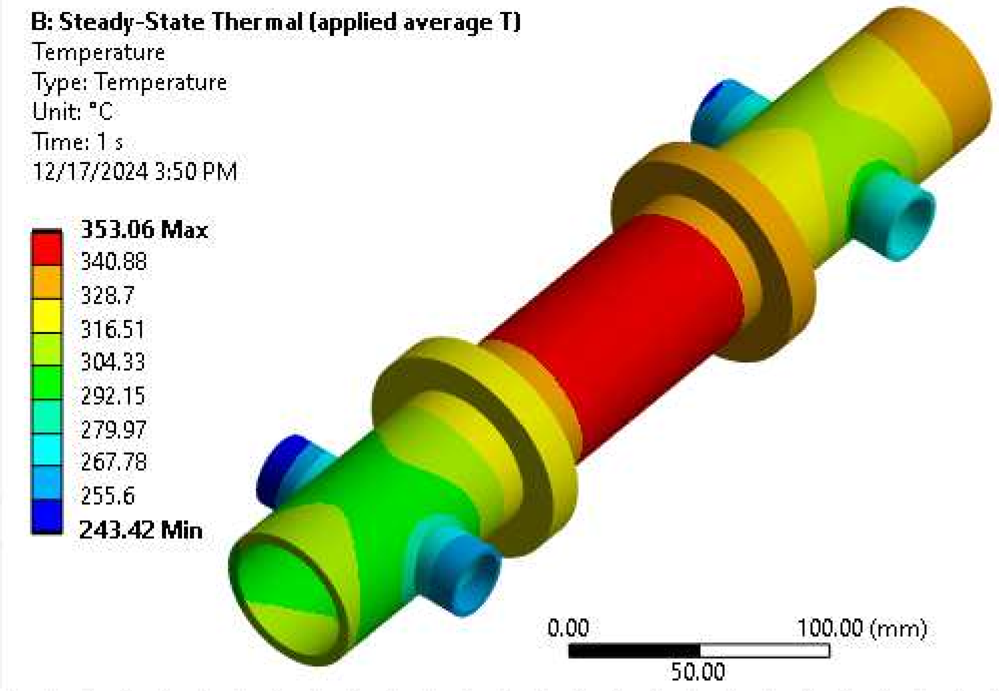


# Example 1: Steady-State Heat Transfer

- It's also instructive to compare the applied average temperatures (right) to what we would get if we applied constant temperatures (left)




Constant temperatures enforced




Average temperatures enforced

# Example 2: Transient Heat Transfer

- Ok. So far, so good. But what if we want to apply average temperatures that change over time?
- This is a little more difficult, because constraint equations don’t support tabular loads (we’d have to ask Ansys why), but it can still be done with a load-stepping approach and the ‘cecmod’ command (below)
- The idea is that we have to modify each constraint equation over every required load step to reflect the updated average temperatures over time (notice this is ALL that the cecmod command will allow us to modify)

**CECMOD**

**CECMOD**, *NEQN*, *CONST*

**Modifies the constant term of a constraint equation during solution.**

SOLUTION: Load Step Options

Compatible Products: – | Pro | Premium | Enterprise | Ent PP | Ent Solver | –

***NEQN***

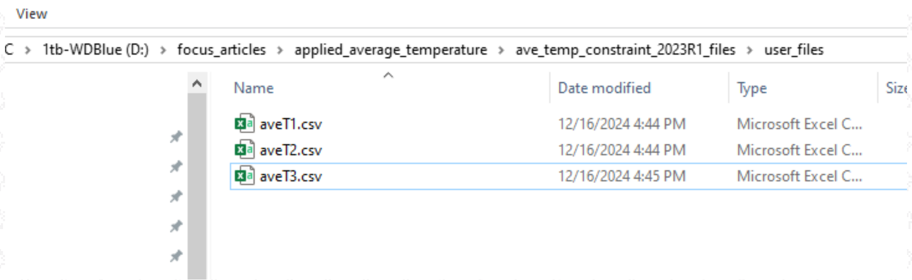
Reference number of constraint equation.

***CONST***

New value of the constant term of equation.

# Example 2: Transient Heat Transfer

- The average temperatures on each of the pipe IDs is defined in three separate comma-delimited (csv) files as shown below
- We store these in the Workbench project user_files folder for convenience (this is a file path which Workbench knows about and will travel with the project).




| time (s) | AveT1 (C) |
|---|---|
| 0 | -2.18329 |
| 18.5 | 38.69604 |
| 25.8 | 83.9732 |
| 33.1 | 132.2438 |
| 40.4 | 179.4773 |
| 47.7 | 223.8253 |
| 55 | 264.5705 |
| 62.3 | 301.4408 |
| 69.6 | 334.6947 |
| 76.9 | 364.84 |
| 84.2 | 392.294 |
| 91.5 | 417.356 |
| 98.8 | 440.251 |
| 106.1 | 461.1641 |
| 113.4 | 480.2589 |
| 120.7 | 497.687 |
| 128 | 513.5879 |
| 135.3 | 528.158 |
| 142.6 | 541.497 |

| time (s) | AveT2 (C) |
|---|---|
| 0 | -4 |
| 18.5 | 0 |
| 25.8 | 33.9732 |
| 33.1 | 82.24377 |
| 40.4 | 129.4773 |
| 47.7 | 173.8253 |
| 55 | 214.5705 |
| 62.3 | 251.4408 |
| 69.6 | 284.6947 |
| 76.9 | 314.84 |
| 84.2 | 342.294 |
| 91.5 | 367.356 |
| 98.8 | 390.251 |
| 106.1 | 411.1641 |
| 113.4 | 430.2589 |
| 120.7 | 447.687 |
| 128 | 463.5879 |
| 135.3 | 478.158 |
| 142.6 | 491.497 |

| time (s) | AveT3 (C) |
|---|---|
| 0 | -4 |
| 18.5 | 0 |
| 25.8 | 23.9732 |
| 33.1 | 72.24377 |
| 40.4 | 119.4773 |
| 47.7 | 163.8253 |
| 55 | 204.5705 |
| 62.3 | 241.4408 |
| 69.6 | 274.6947 |
| 76.9 | 304.84 |
| 84.2 | 332.294 |
| 91.5 | 357.356 |
| 98.8 | 380.251 |
| 106.1 | 401.1641 |
| 113.4 | 420.2589 |
| 120.7 | 437.687 |
| 128 | 453.5879 |
| 135.3 | 468.158 |
| 142.6 | 481.497 |

# Example 2: Transient Heat Transfer

- The new code is summarized below
- lines 8 - 27: Read in the temperature data and store in tables

```
 8 ! read average temperatures from file in user_files location
 9 fname1 = 'aveT1'  !file name (careful. There's a 32 char limit)
10 fname2 = 'aveT2'
11 fname3 = 'aveT3'
12 *dim,fpath1,string,248   !path string array to file (strings longer than 32 char n
13 *dim,fpath2,string,248
14 *dim,fpath3,string,248
15 !create path string by appending fname to user_files location (comes from WB)
16 fpath1(1) = strcat(_wb_userfiles_dir(1),fname1)
17 fpath2(1) = strcat(_wb_userfiles_dir(1),fname2)
18 fpath3(1) = strcat(_wb_userfiles_dir(1),fname3)
19 !get the number of lines in the file
20 /inquire,numlines,lines,fpath1(1),csv
21 !define Ave temperature table for N discrete times between 0 and end-time
22 *dim,aveT1,table,numlines-1,,,time
23 *dim,aveT2,table,numlines-1,,,time
24 *dim,aveT3,table,numlines-1,,,time
25 *tread,aveT1,fpath1(1),csv,,1  !read in the file (skip the first line)
26 *tread,aveT2,fpath1(1),csv,,1
27 *tread,aveT3,fpath1(1),csv,,1
```

- lines 29 – 62: Define a ‘time’ array to define the load-step times to apply and initialized the constraint equations at t=0s as done for the steady-state case

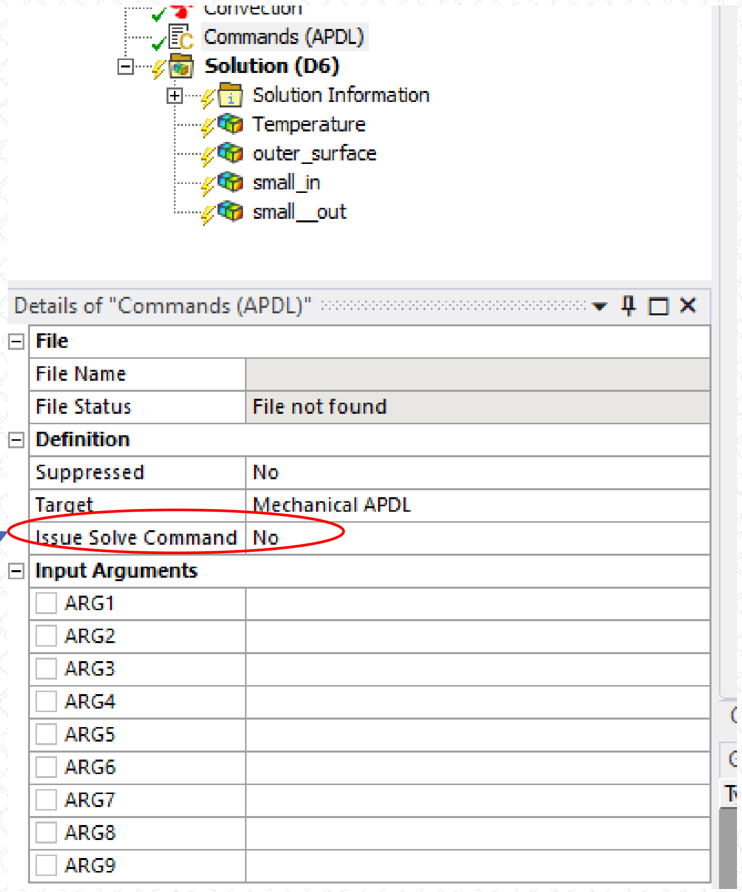
```
29 !define an array to hold the three time points corresponding to discrete times above
30 !do this by copying the 0 (time) column of any of the three tables...
31 *dim,ttim,array,numlines-1
32 *vfun,ttim(1),copy,aveT1(1,0)
33
34 /PREP7
35 !iniitalize ave temperatures at first load step
36
37 !define constraint equations as usual
38 cmsel,s,osurf
39 *get,nn,node,,count
40 nd = ndnext(0)
41 *do,i,1,nn
42    ce,100,aveT1(0),nd,temp,1/nn
43    nd= ndnext(nd)
44 *ENDDO
45
46 cmsel,s,small_out
47 *get,nn,node,,count
48 nd=ndnext(0)
49 *do,i,1,nn
50    ce,101,aveT3(0),nd,temp,1/nn     !apply AveT2 to small_out
51    nd=ndnext(nd)
52 *ENDDO
53
54 cmsel,s,small_in
55 *get,nn,node,,count
56 nd=ndnext(0)
57 *do,i,1,nn
58    ce,102,aveT2(0),nd,temp,1/nn     !apply AveT3 to small_in
59    nd=ndnext(nd)
60 *ENDDO
61 allsel,
62 /SOLU
```

# Example 2: Transient Heat Transfer

- lines 64 - 84: loop through the number of load steps (in the case, the same as the number of time points defined in the tables, but these can differ)
- At each load step i, issue the 'cecmod' command to apply the new average temperaure at time(i) and 'solve'

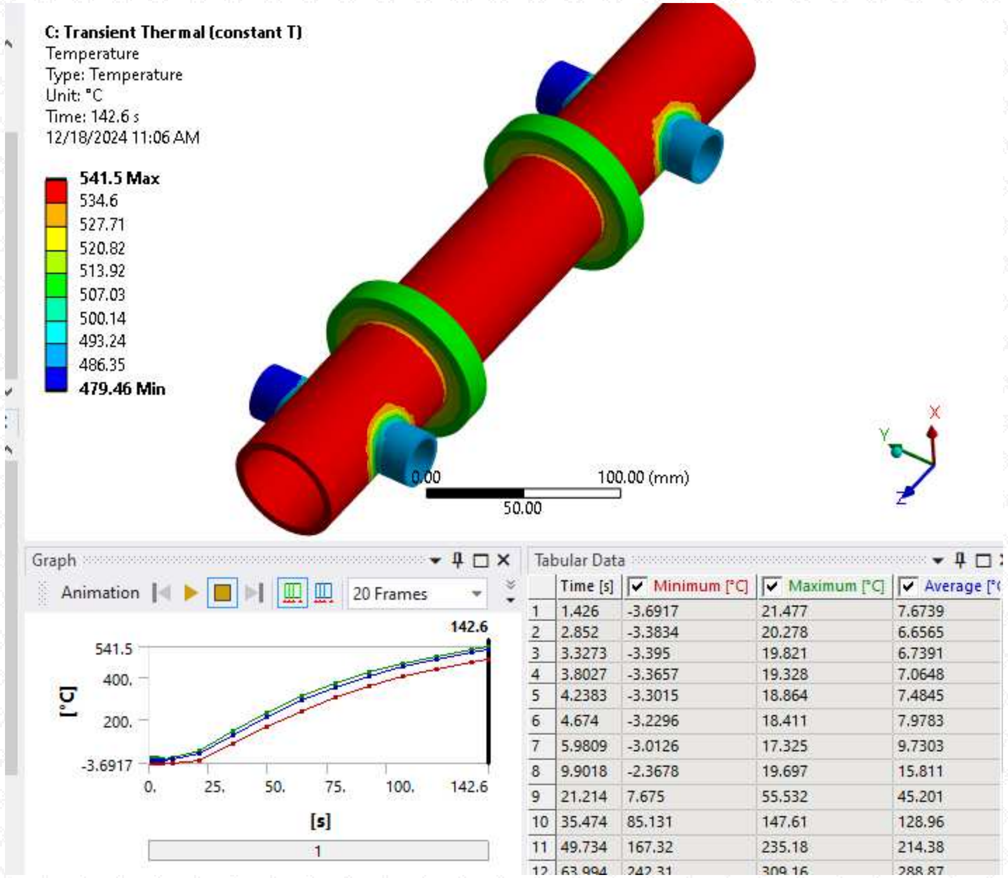
```
64  !specify number of load steps. Note: this can be different from table and array
65  !break points
66  numsteps = numlines-1
67
68  !perform load step solutions with APDL (set 'Issue Solve Command' to 'No' in WB)
69  !this allows us to just use one WB load step. Hopefully, making management a little eaier.
70  !Note that the first load step reduntantly redefines aveTemp at 60 s. This is just
71  !to make the code easier to read and follow.
72  *do,i,1,numsteps
73     *if,i,eq,1,then
74        *if,ttim(1),eq,0,then
75           ttim(1)=ttim(1)+0.001  !can't start an analysis at t=0
76        *endif
77     *endif
78     time,ttim(i)
79     !aveTemp = aveT1(ttim(i))
80     cecmod,100,aveT1(ttim(i))
81     cecmod,101,aveT3(ttim(i))
82     cecmod,102,aveT2(ttim(i))
83     solve
84  *ENDDO
```

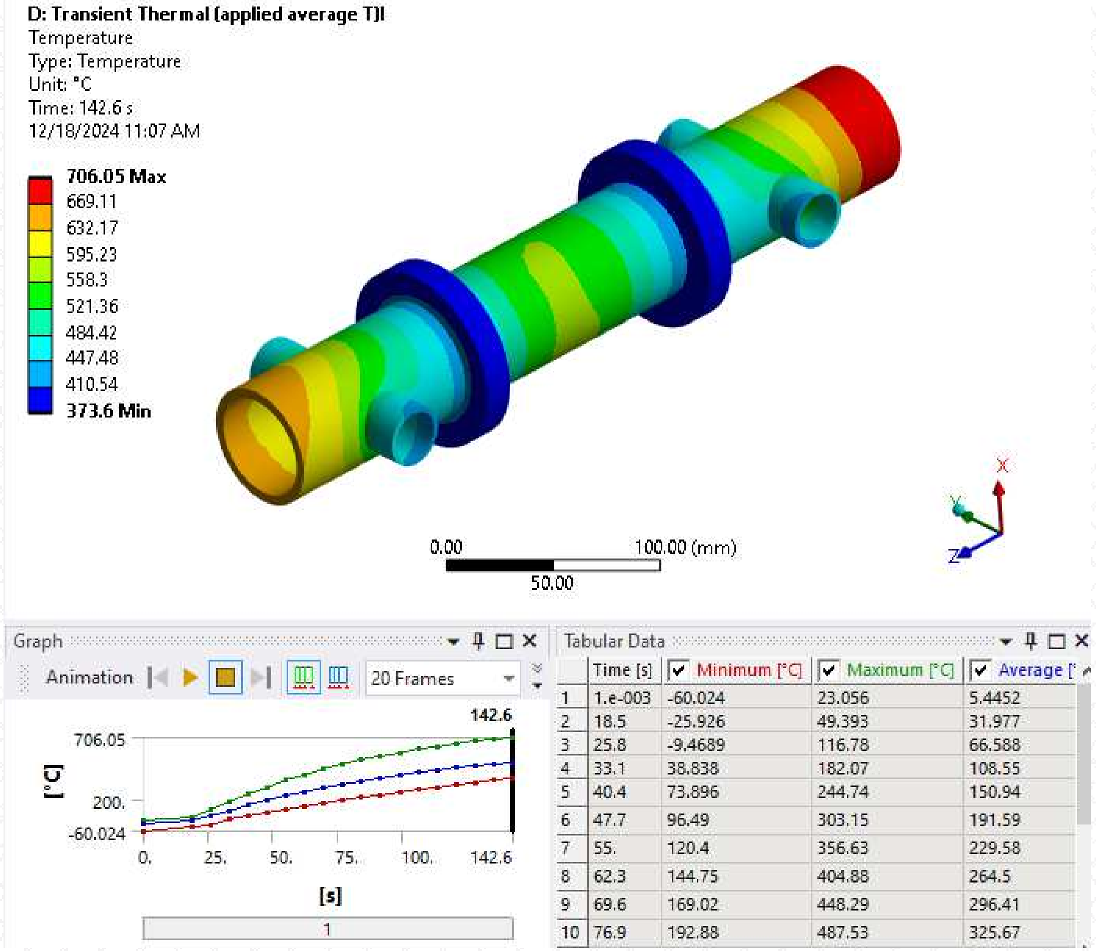
- Important NOTE: Turn the 'Issue Solve Command' to 'No' because we're doing the 'solve' in the macro
- If you don't do this, Workbench will solve the model again (you'll get a redundant additional solution)



# Example 2: Transient Heat Transfer

- And review the results. Once again, we review the difference between applying the average over time (right) vs temperature over time (left)...



# Example 2: Transient Heat Transfer

- And again check that the average temperature for each region matches the input...

| Time (s) | osurf (aveT1) | small_in (aveT2) | small_out (aveT2) |
|---|---|---|---|
| Time [s] | Average [°C] | Average [°C] | Average [°C] |
| 1.e-003 | -2.1811 | -3.9998 | -3.9998 |
| 18.5 | 38.696 | -2.7491e-008 | 1.3678e-008 |
| 25.8 | 83.973 | 33.973 | 23.973 |
| 33.1 | 132.24 | 82.244 | 72.244 |
| 40.4 | 179.48 | 129.48 | 119.48 |
| 47.7 | 223.83 | 173.83 | 163.83 |
| 55. | 264.57 | 214.57 | 204.57 |
| 62.3 | 301.44 | 251.44 | 241.44 |
| 69.6 | 334.69 | 284.69 | 274.69 |
| 76.9 | 364.84 | 314.84 | 304.84 |
| 84.2 | 392.29 | 342.29 | 332.29 |
| 91.5 | 417.36 | 367.36 | 357.36 |
| 98.8 | 440.25 | 390.25 | 380.25 |
| 106.1 | 461.16 | 411.16 | 401.16 |
| 113.4 | 480.26 | 430.26 | 420.26 |
| 120.7 | 497.69 | 447.69 | 437.69 |
| 128. | 513.59 | 463.59 | 453.59 |
| 135.3 | 528.16 | 478.16 | 468.16 |
| 142.6 | 541.5 | 491.5 | 481.5 |

- Average summary from Mechanical temperature contour results for the three regions (osurf, small_in, small_out)

aveT1.csv

| time (s) | AveT1 (C) |
|---|---|
| 0 | -2.18329 |
| 18.5 | 38.69604 |
| 25.8 | 83.9732 |
| 33.1 | 132.2438 |
| 40.4 | 179.4773 |
| 47.7 | 223.8253 |
| 55 | 264.5705 |
| 62.3 | 301.4408 |
| 69.6 | 334.6947 |
| 76.9 | 364.84 |
| 84.2 | 392.294 |
| 91.5 | 417.356 |
| 98.8 | 440.251 |
| 106.1 | 461.1641 |
| 113.4 | 480.2589 |
| 120.7 | 497.687 |
| 128 | 513.5879 |
| 135.3 | 528.158 |
| 142.6 | 541.497 |

aveT2.csv

| time (s) | AveT2 (C) |
|---|---|
| 0 | -4 |
| 18.5 | 0 |
| 25.8 | 33.9732 |
| 33.1 | 82.24377 |
| 40.4 | 129.4773 |
| 47.7 | 173.8253 |
| 55 | 214.5705 |
| 62.3 | 251.4408 |
| 69.6 | 284.6947 |
| 76.9 | 314.84 |
| 84.2 | 342.294 |
| 91.5 | 367.356 |
| 98.8 | 390.251 |
| 106.1 | 411.1641 |
| 113.4 | 430.2589 |
| 120.7 | 447.687 |
| 128 | 463.5879 |
| 135.3 | 478.158 |
| 142.6 | 491.497 |

aveT3.csv

| time (s) | AveT3 (C) |
|---|---|
| 0 | -4 |
| 18.5 | 0 |
| 25.8 | 23.9732 |
| 33.1 | 72.24377 |
| 40.4 | 119.4773 |
| 47.7 | 163.8253 |
| 55 | 204.5705 |
| 62.3 | 241.4408 |
| 69.6 | 274.6947 |
| 76.9 | 304.84 |
| 84.2 | 332.294 |
| 91.5 | 357.356 |
| 98.8 | 380.251 |
| 106.1 | 401.1641 |
| 113.4 | 420.2589 |
| 120.7 | 437.687 |
| 128 | 453.5879 |
| 135.3 | 468.158 |
| 142.6 | 481.497 |

- Average Temperature Input files

# Closing Thoughts: Why?

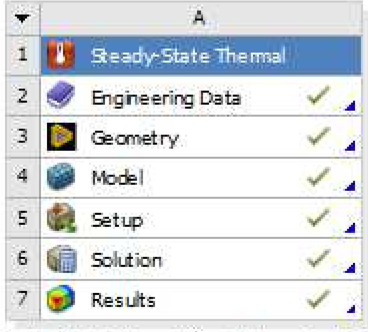
- We've demonstrated that applying an average temperature constraint in a finite element model does indeed "work". But why?
- Some engineers object to the fact that prescribing an average does not prescibe a unique temperature distribution (there's an infinte set of *different* temperature distributions which satisfy a given average)
- If that's true (and it is), then *which* average does this solution prescribe exactly?
- Put another way: what can we say (if anything) about the resulting temperature distribution which satisfies our average?

- What we can say is that the resulting temperature distribution which satisfies the average temperature constraint is the only one which does so while also satisfying equlibrium (i.e. also satisfies all other loads and boundary conditions in the model). No other distribution can do this (even if it satisfies the average constraint). That makes it unique and is 'why' this works
- In other words, it is the optimal (minimizes the thermal energy) solution for the problem as posed.
- In fact, if a user *wants* a different temperature distribution (for a given average), they can simply impose different coefficients in the constraint equation (they can 'weight' the 1/n coefficient which we held constant in order to impose an arithmetic mean)

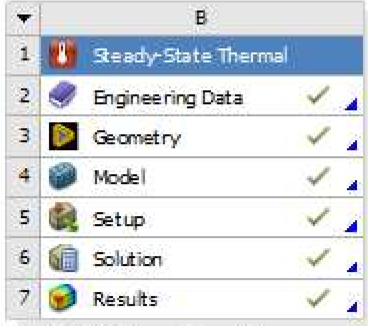
# Workbench Project Description

- We're including the Ansys 2023R1 archive containing the two example cases along with this blog post
- The Project contains the four analysis systems shown below. Systems B and D contain the steady-state and transient applied average temperature cases, respectively (examples 1 and 2), while systems A and B contain the steady-state and transient constant temperature cases we compared them two (slides 14 and 19

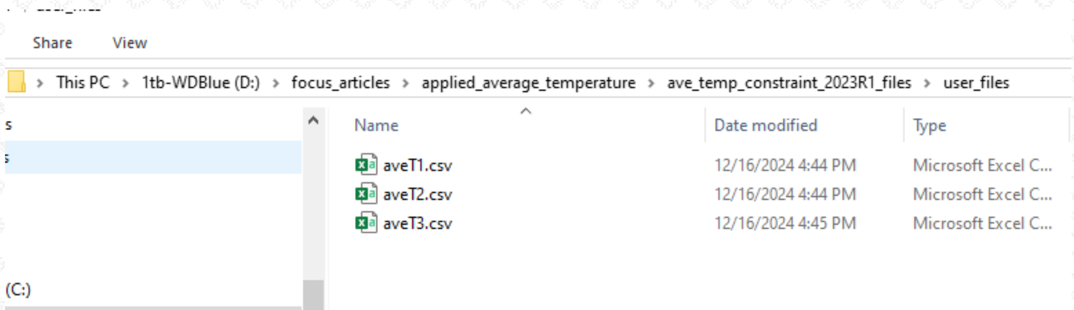
- Comma-delimited (csv) files containig each region's average temprature over time are stored in the user_files folder
- The APDL command objects of slides 12 and 17 can be found in the tree outlines of systems B and D

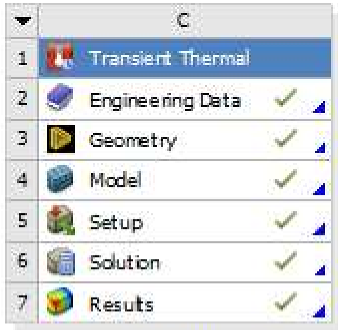


Steady-State Thermal (applied constant T)

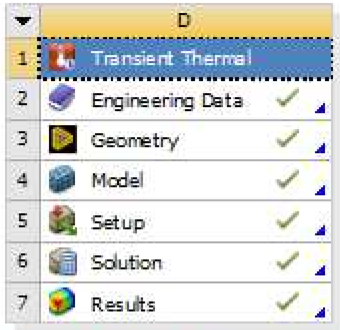


Steady-State Thermal (applied average T)



Transient Thermal (constant T)



Transient Thermal (applied average T)l

# Summary

- In this article, we looked at how to apply an average temperature constraint in Ansys
- We applied the constraint using the APDL 'CE' command in a command object
- We did this for both steady-state thermal and transient thermal environments
- On slide 21, we explain why this works and in fact produces a unique solution (contrary to what is often assumed)