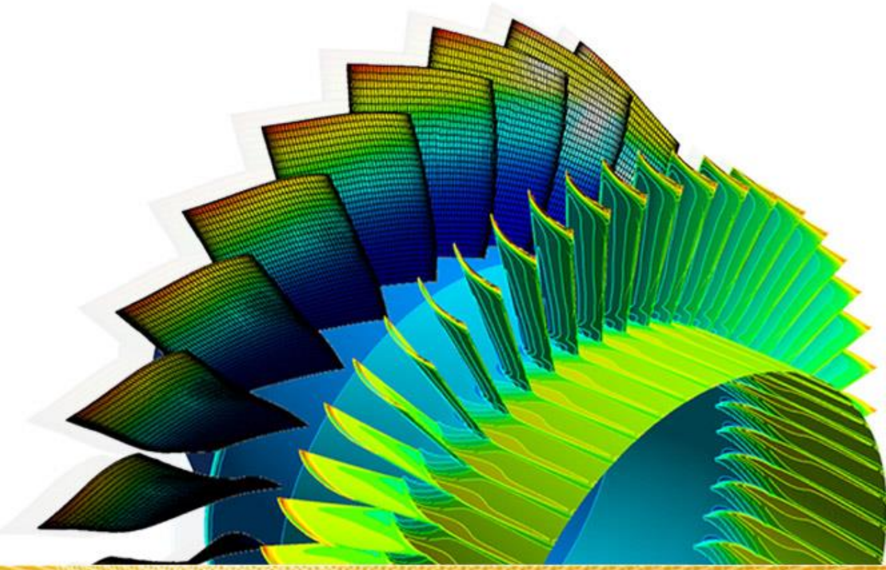
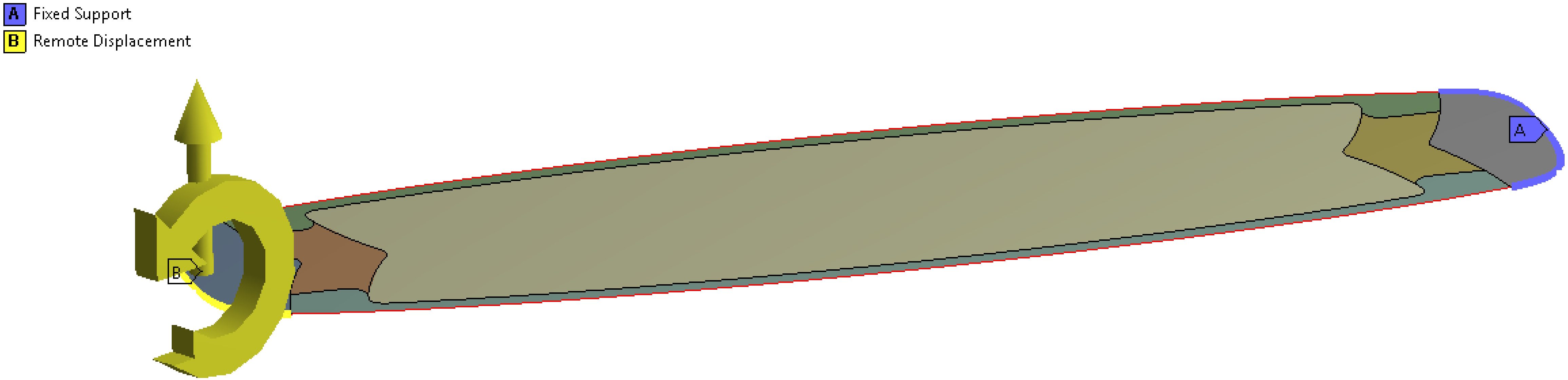
ANSYS Composite PrepPost 19.0

Workshop 10.8 – Scripts in ACP

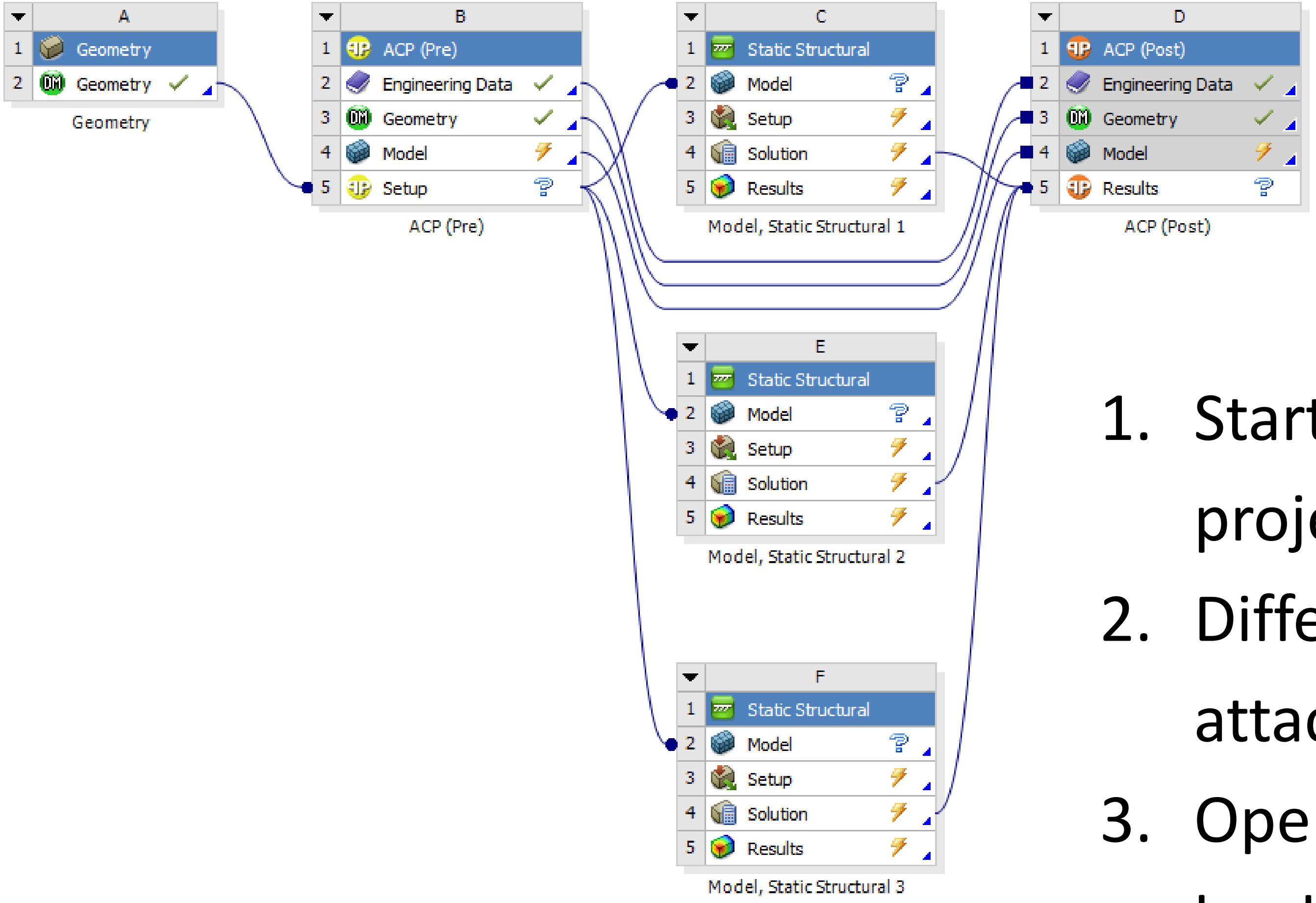


17. Workshop Scripts in ACP

- In this workshop we use scripts to automatically save snapshots of the same model under different loading conditions
- The ACP model is the same built in Workshop 2 (kiteboard), different loading conditions are applied using separate Static Structural toolboxes

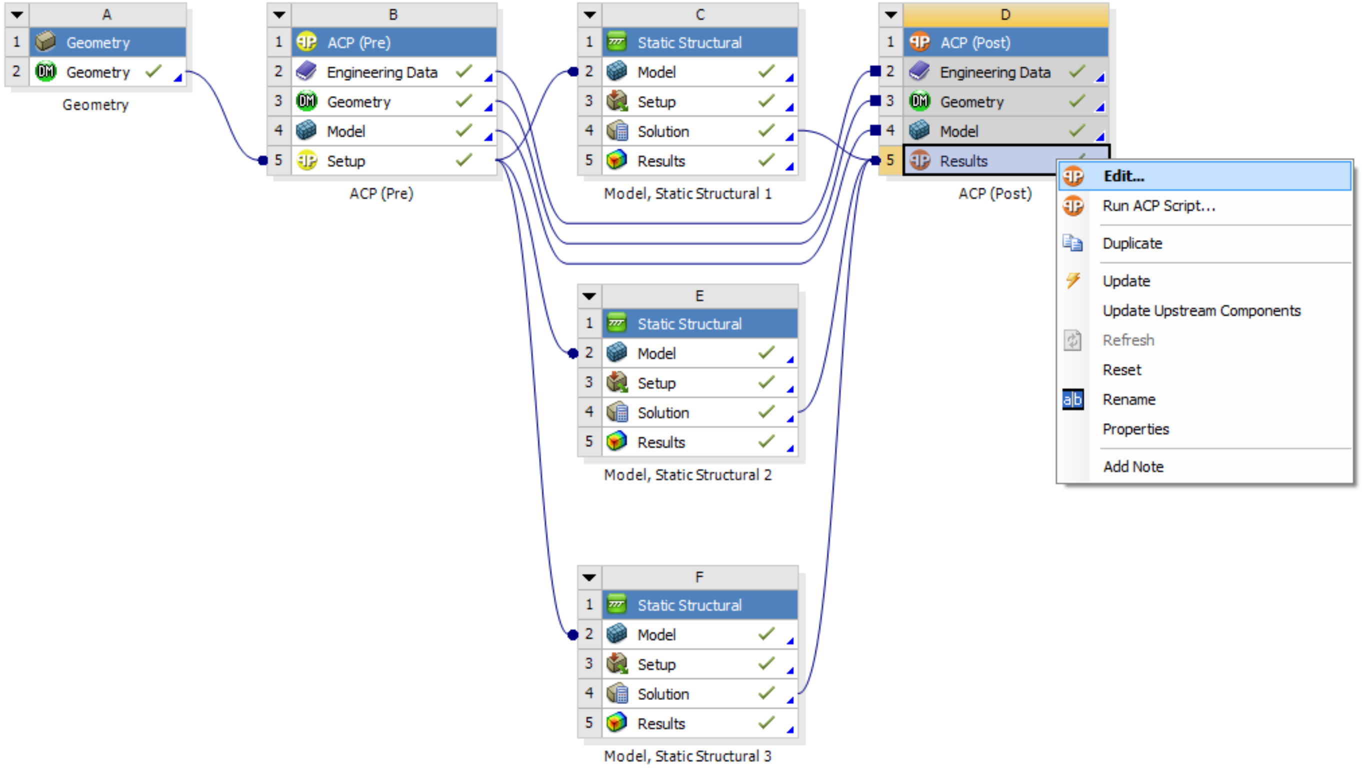


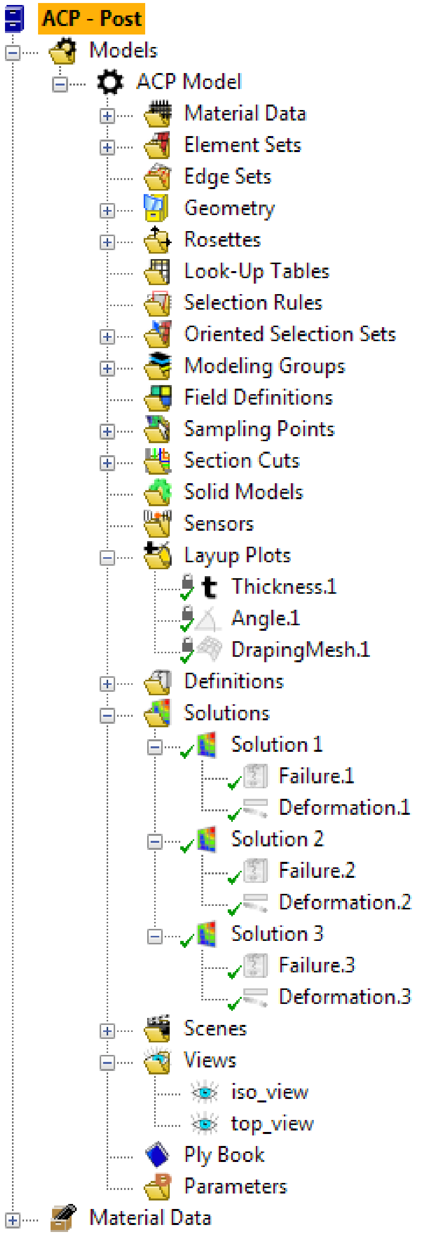
17. Workshop Scripts in ACP



1. Start ANSYS Workbench and update the project
2. Different Static Structural toolboxes are attached to the same ACP (Pre) and (Post)
3. Open Mechanical and review the different loading conditions applied to the kiteboard

17. Workshop Scripts in ACP





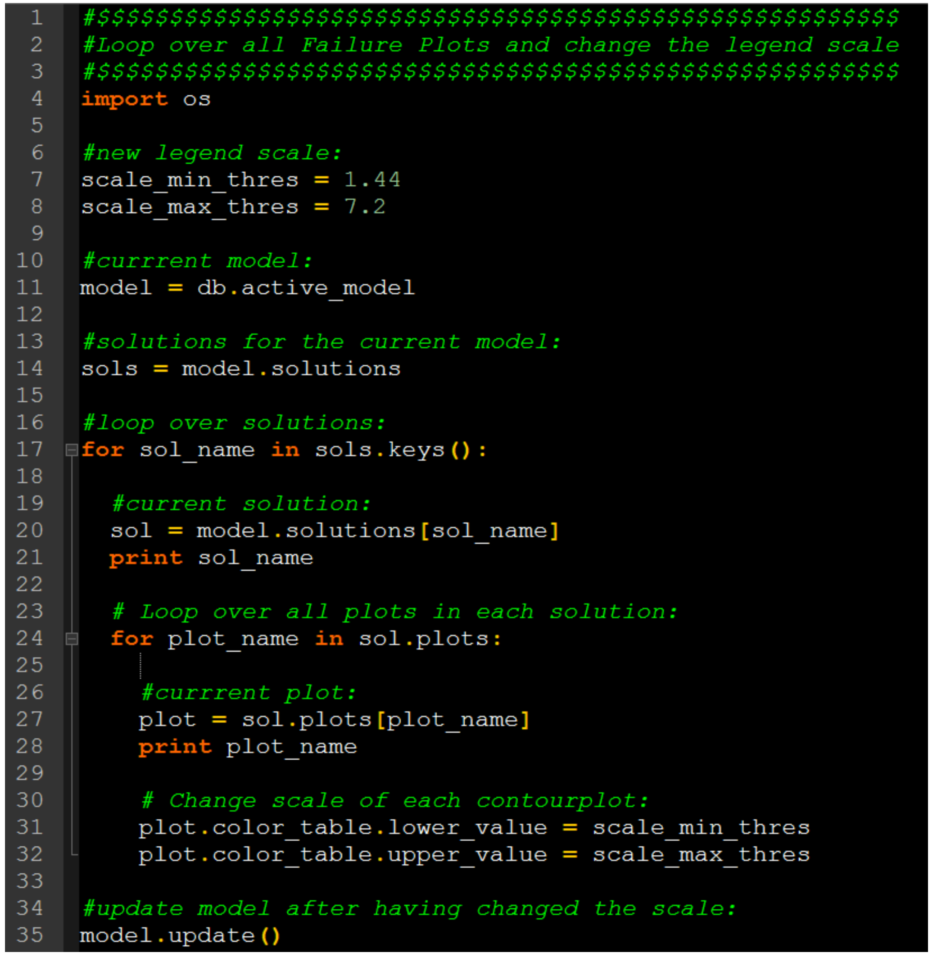
1. Edit *Results* of ACP (Post)
2. Each solution already contains a defined failure and deformation plot. A thickness plot is present among the Layup Plots.
3. Two different views are pre-defined, “iso_view” and “top_view”

17. Workshop Scripts in ACP

Change Legend Scale

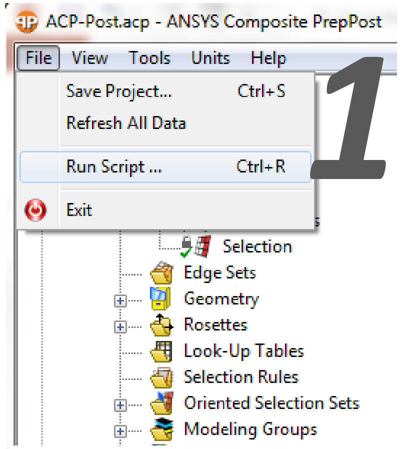
1

acp_change_plot_scales.py

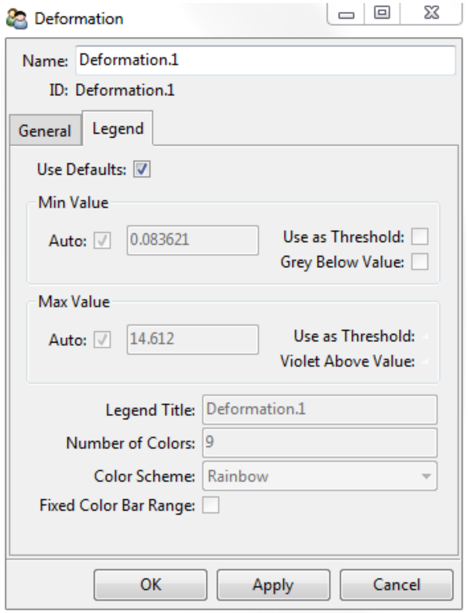


```
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
#Loop over all Failure Plots and change the legend scale
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
import os

#new legend scale:
scale_min_thres = 1.44
scale_max_thres = 7.2

#currrent model:
model = db.active_model

#solutions for the current model:
sols = model.solutions

#loop over solutions:
for sol_name in sols.keys():

  #current solution:
  sol = model.solutions[sol_name]
  print sol_name

  # Loop over all plots in each solution:
  for plot_name in sol.plots:

    #currrent plot:
    plot = sol.plots[plot_name]
    print plot_name

    # Change scale of each contourplot:
    plot.color_table.lower_value = scale_min_thres
    plot.color_table.upper_value = scale_max_thres

#update model after having changed the scale:
model.update()
```

1. Run the python script to change the scale range for the plots and check the effects in ACP



2. Only failure plots are updated since deformation default behavior is set on ‘Auto’



Name: Deformation.1

ID: Deformation.1

Use Defaults: ☑

Min Value

Auto: ☑ 0.083621 Use as Threshold: □ Grey Below Value: □

Max Value

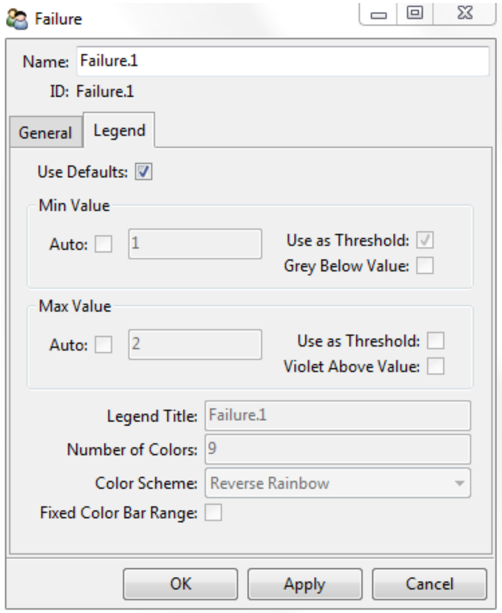
Auto: ☑ 14.612 Use as Threshold: Violet Above Value:

Legend Title: Deformation.1

Number of Colors: 9

Color Scheme: Rainbow

Fixed Color Bar Range: □



Name: Failure.1

ID: Failure.1

Use Defaults: ☑

Min Value

Auto: □ 1 Use as Threshold: ☑ Grey Below Value: □

Max Value

Auto: □ 2 Use as Threshold: □ Violet Above Value: □

Legend Title: Failure.1

Number of Colors: 9

Color Scheme: Reverse Rainbow

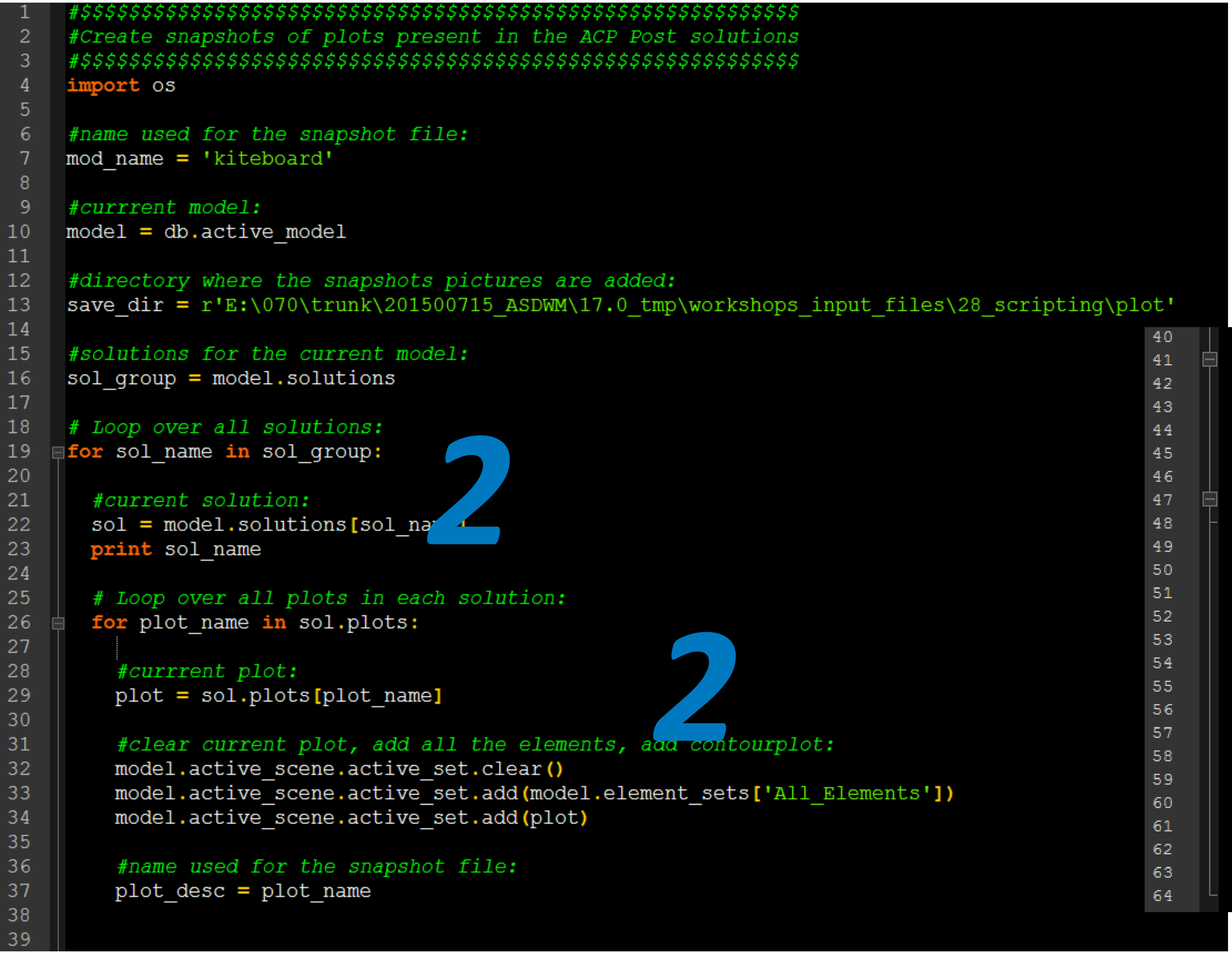
Fixed Color Bar Range: □

17. Workshop Scripts in ACP

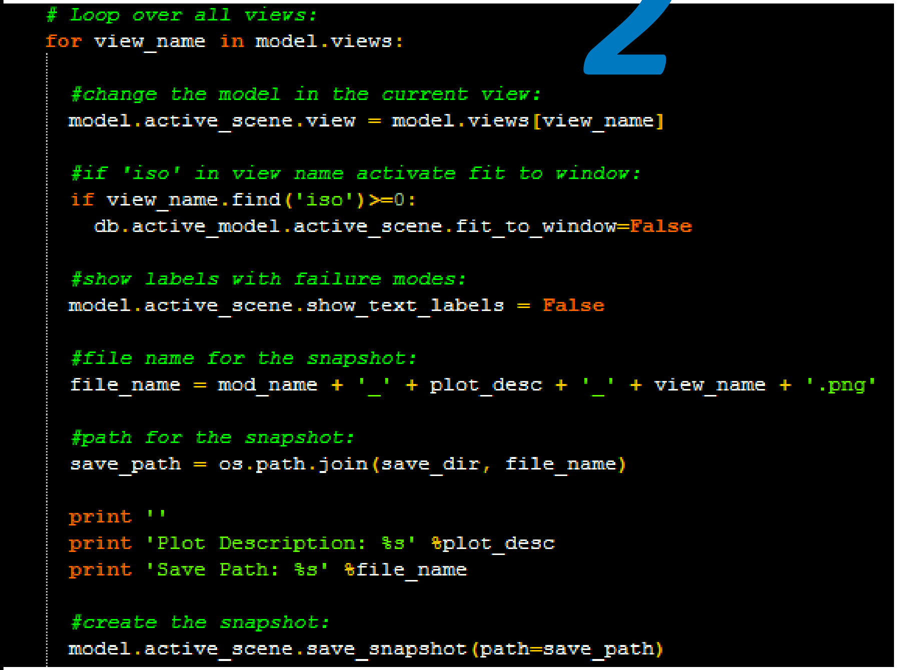
Take Snapshots

acp_create_snapshots.py

1. Run the python script to take the snapshots
2. The scripts has 3 nested loops over available solutions, plots and then views)



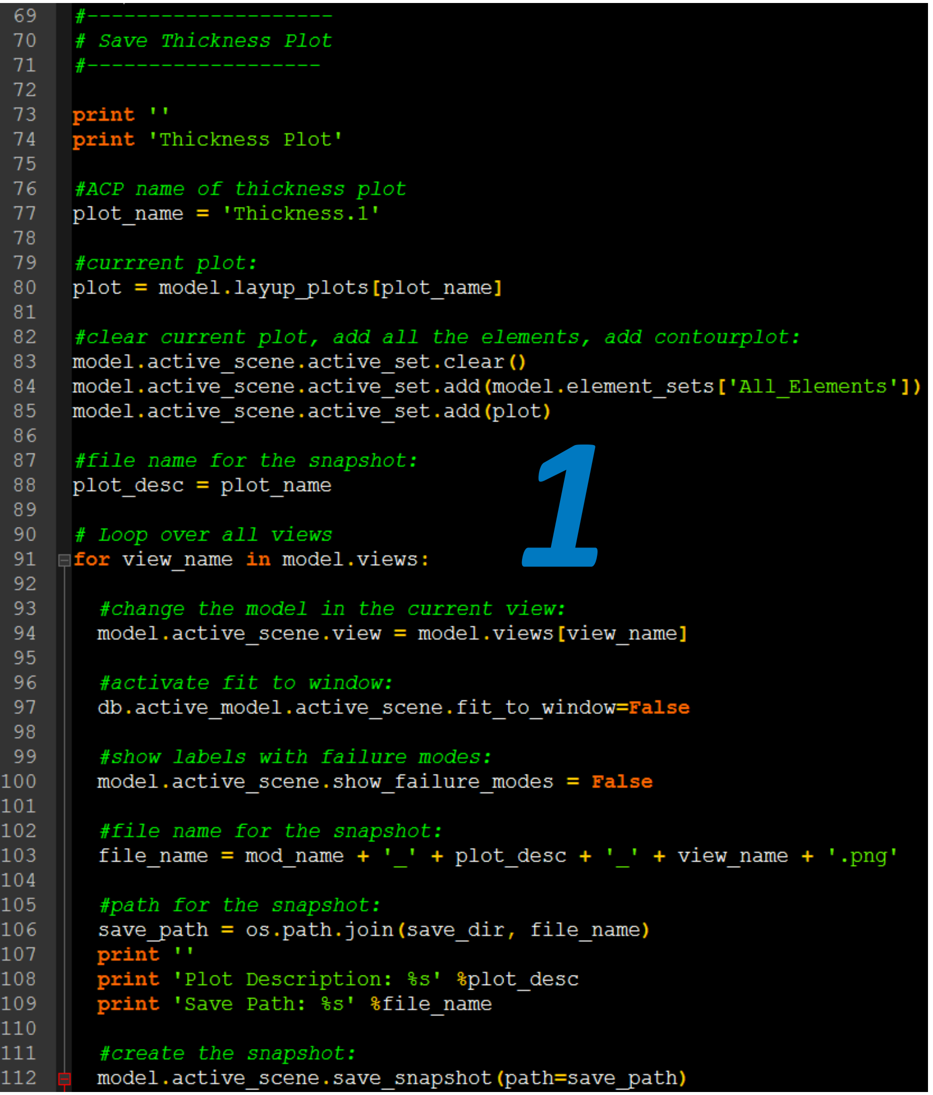
```
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
#Create snapshots of plots present in the ACP Post solutions
#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
import os

#name used for the snapshot file:
mod_name = 'kiteboard'

#currrent model:
model = db.active_model

#directory where the snapshots pictures are added:
save_dir = r'E:\070\trunk\201500715_ASDWM\17.0_tmp\workshops_input_files\28_scripting\plot'

#solutions for the current model:
sol_group = model.solutions

# Loop over all solutions:
for sol_name in sol_group:

  #current solution:
  sol = model.solutions[sol_name]
  print sol_name

  # Loop over all plots in each solution:
  for plot_name in sol.plots:

    #currrent plot:
    plot = sol.plots[plot_name]

    #clear current plot, add all the elements, add contourplot:
    model.active_scene.active_set.clear()
    model.active_scene.active_set.add(model.element_sets['All_Elements'])
    model.active_scene.active_set.add(plot)

    #name used for the snapshot file:
    plot_desc = plot_name

```



```
    # Loop over all views:
    for view_name in model.views:

      #change the model in the current view:
      model.active_scene.view = model.views[view_name]

      #if 'iso' in view name activate fit to window:
      if view_name.find('iso')>=0:
        db.active_model.active_scene.fit_to_window=False

      #show labels with failure modes:
      model.active_scene.show_text_labels = False

      #file name for the snapshot:
      file_name = mod_name + '_' + plot_desc + '_' + view_name + '.png'

      #path for the snapshot:
      save_path = os.path.join(save_dir, file_name)

      print ''
      print 'Plot Description: %s' %plot_desc
      print 'Save Path: %s' %file_name

      #create the snapshot:
      model.active_scene.save_snapshot(path=save_path)
```

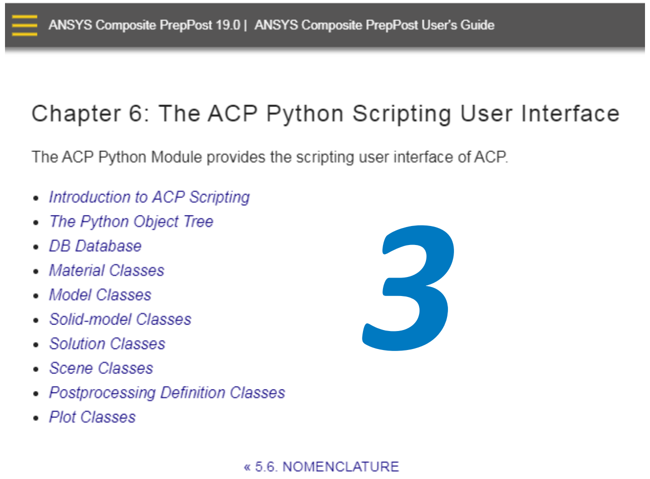
17. Workshop Scripts in ACP

acp_create_snapshots.py



```
#--------------------
# Save Thickness Plot
#-------------------

print ''
print 'Thickness Plot'

#ACP name of thickness plot
plot_name = 'Thickness.1'

#currrent plot:
plot = model.layup_plots[plot_name]

#clear current plot, add all the elements, add contourplot:
model.active_scene.active_set.clear()
model.active_scene.active_set.add(model.element_sets['All_Elements'])
model.active_scene.active_set.add(plot)

#file name for the snapshot:
plot_desc = plot_name

# Loop over all views
for view_name in model.views:

  #change the model in the current view:
  model.active_scene.view = model.views[view_name]

  #activate fit to window:
  db.active_model.active_scene.fit_to_window=False

  #show labels with failure modes:
  model.active_scene.show_failure_modes = False

  #file name for the snapshot:
  file_name = mod_name + '_' + plot_desc + '_' + view_name + '.png'

  #path for the snapshot:
  save_path = os.path.join(save_dir, file_name)
  print ''
  print 'Plot Description: %s' %plot_desc
  print 'Save Path: %s' %file_name

  #create the snapshot:
  model.active_scene.save_snapshot(path=save_path)
```

1. The scripts then take a snapshot of the thickness plot for each of the defined views
2. Browse through the given python scripts to understand all the commands
3. For more on ACP scripting capabilities check the ANSYS help:



ANSYS Composite PrepPost 19.0 | ANSYS Composite PrepPost User's Guide

Chapter 6: The ACP Python Scripting User Interface

The ACP Python Module provides the scripting user interface of ACP.

- Introduction to ACP Scripting
- The Python Object Tree
- DB Database
- Material Classes
- Model Classes
- Solid-model Classes
- Solution Classes
- Scene Classes
- Postprocessing Definition Classes
- Plot Classes

« 5.6. NOMENCLATURE