

PyAnsys: Revolutionizing Engineering Through the Power of Python

Rajesh Meena

Mehul Muradia

Who are we?



Rajesh Meena
Sr. Technology Specialist
Ansys Pune



Mehul Muradia
Technology Specialist
Ansys Pune

- Automation and Customization of Ansys Products
- Customized Workflow Implementations
- Structural Simulation Experts

Agenda



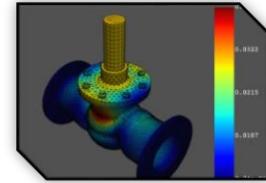
What is
PyAnsys?



Why use it?

```
! -m venv venv  
venv\Scripts\activate  
install ansys-mapdl  
install ansys-dpf-c  
install ansys-dpf-p  
install jupyterlab
```

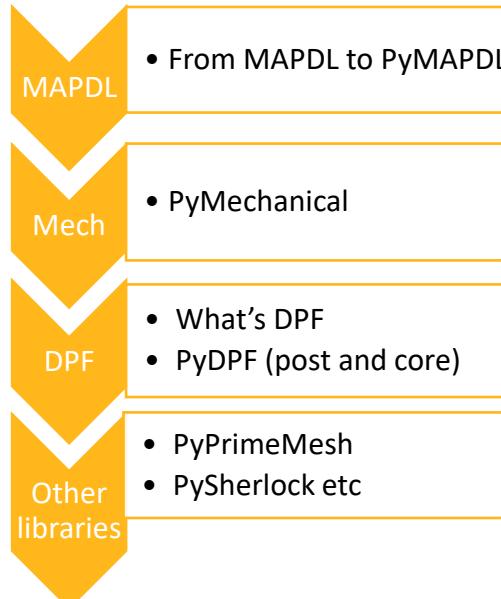
Getting
Started



Libraries
and demos



Final
remarks +
Q&A



What Should be the takeaway?

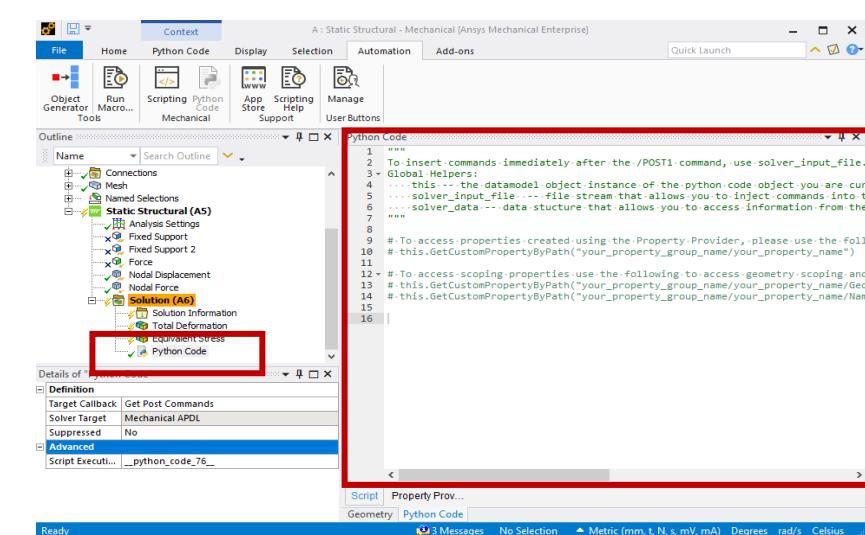
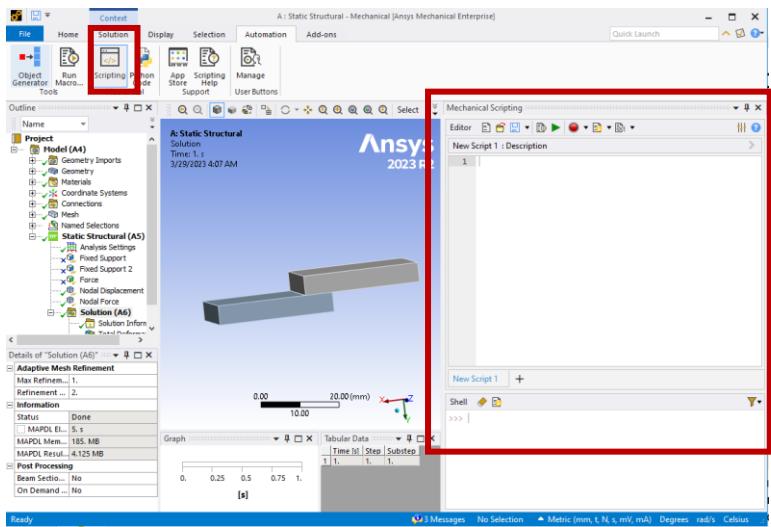
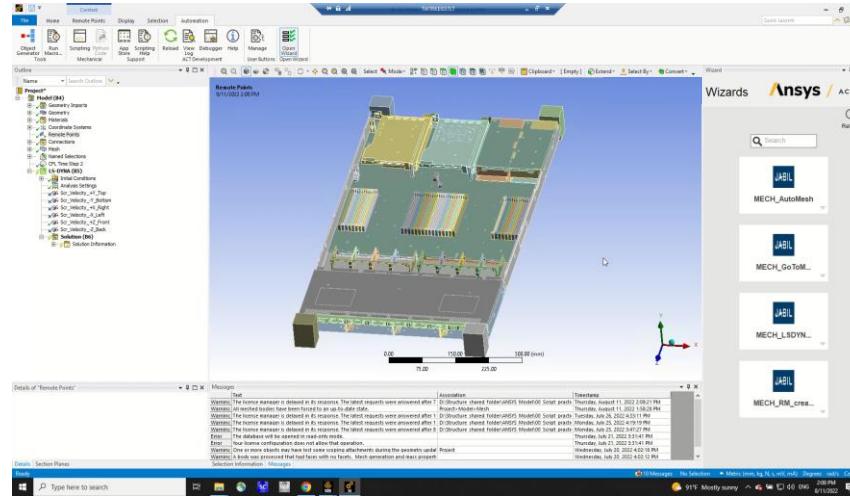
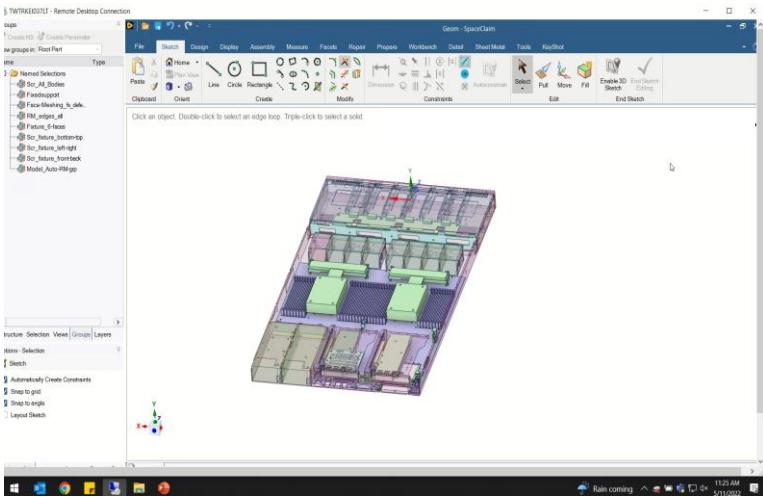
How does PyAnsys makes difference..?

How can I get started **myself**?

Traditional Automation in Ansys (In-Tool)



In-tool Automation



In-tool Automation

- Good for:
 - customizing the software (ACT)
 - Automate repetitive manual operations
- Not optimal for:
 - Multi-tool workflow automation
 - Fully Automated workflow, with maintainability
 - Use of 3rd party tool in the workflow
 - Train AI pair programmer

Simulation Democratization: Facts, Challenges, Solution

Facts

100s

Simulation Tools

1000s

Workflows/Scripts

10s

Programming Languages

Challenges

OS Dependency

Version Management

Full Scale Automation

Maintenance/Upgrade

Interact with Data Packages

Deployment

Solution: Python

Platform Independent

Wider Adoption

Large Community Support

Technology Enabler

Pre-Built Libraries

APIs

What's PyAnsys?



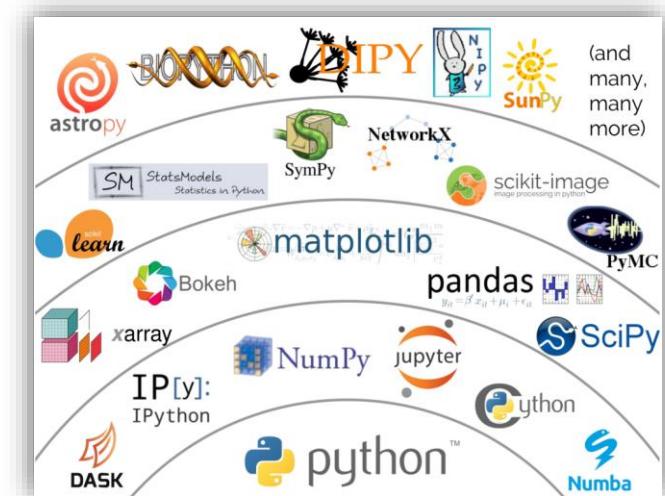
What is PyAnsys?

Set of technologies that allow the user to interface with Ansys products pythonically

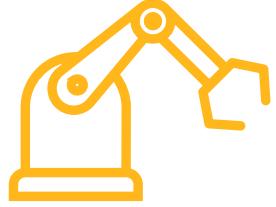


The PyAnsys project is **Ansys's commitment to open-source** where we provide Python libraries that expose Ansys technologies in the Python ecosystem through APIs and interfaces that are **clear, concise, and maintainable**. This allows Ansys customers to do:

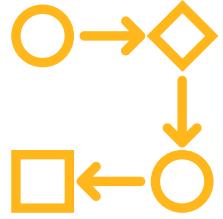
- **Flexible Automation:** Democratize powerful capabilities offered by Ansys through scripting
- **Flexible Distribution :** Connect Ansys and Open-Source technologies in a seamless manner
- **Broader Technology Integration:** Integrate Ansys physics capabilities easily with AI/ML



PyAnsys: Ansys's Commitment to Open Source



Better Automation



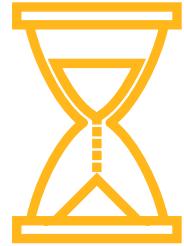
Simple Workflow



Access from Anywhere



Integration with AI/ML



Real Time Post Process

Ansys Technologies in Python Ecosystem

Clear, Concise, Maintainable APIs

Flexible Automation

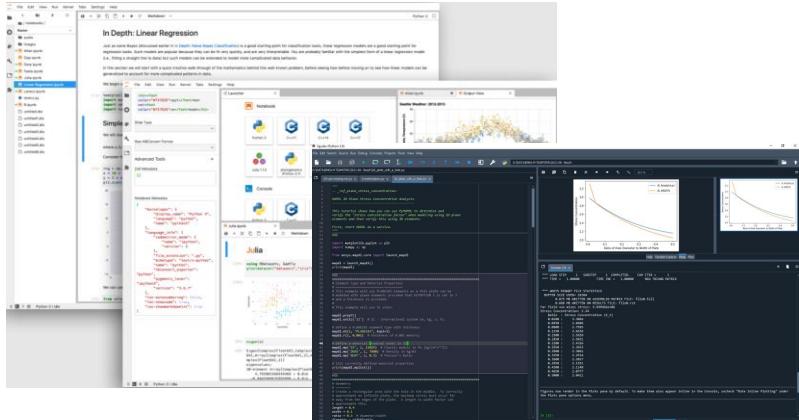
Flexible Distribution

Broader Technology Integration

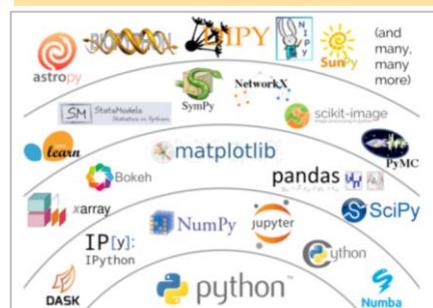
Easier for Deployment

Scripting from within the customer tools – PyAnsys

Any tool from which you can write Python (UI is yours to choose)



Possibility to use other useful Python tools



Customer tools

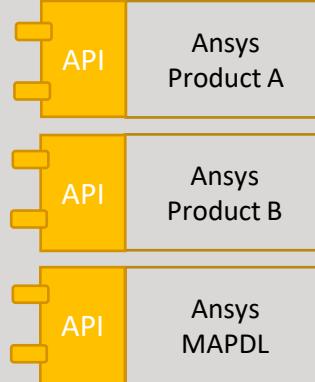


PyAnsys
technology

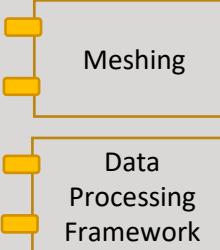


Ansys
Technology

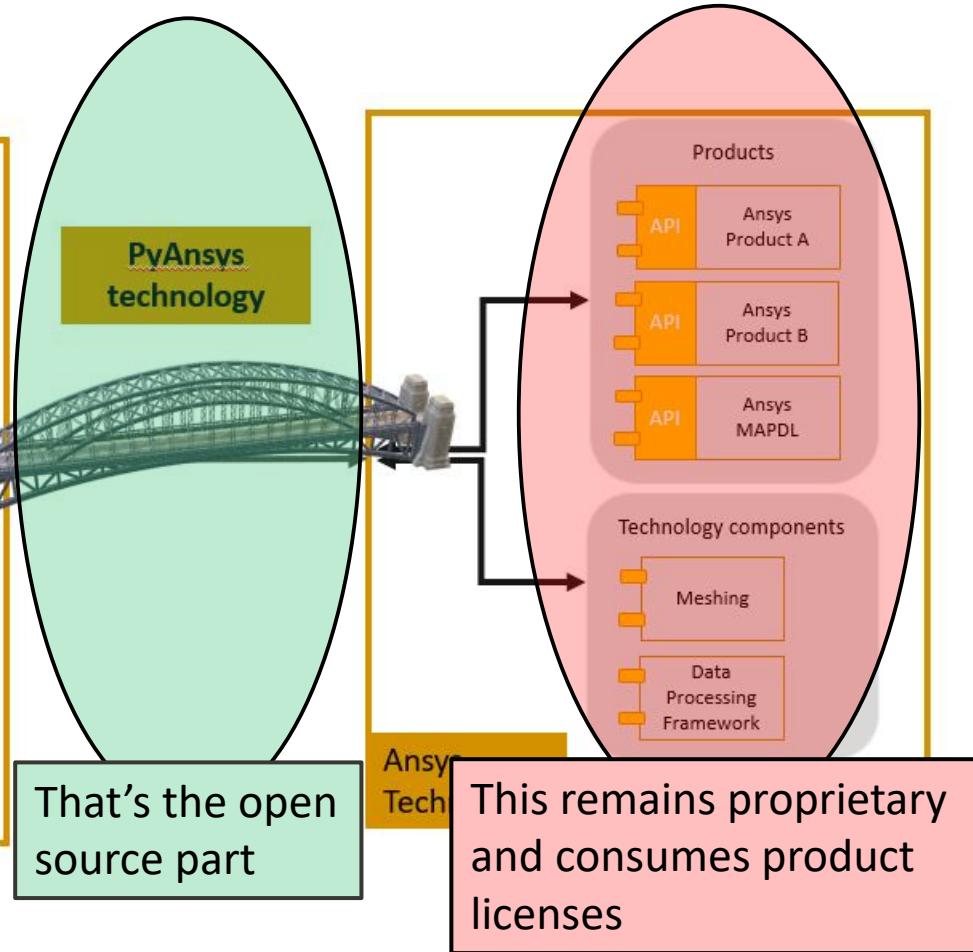
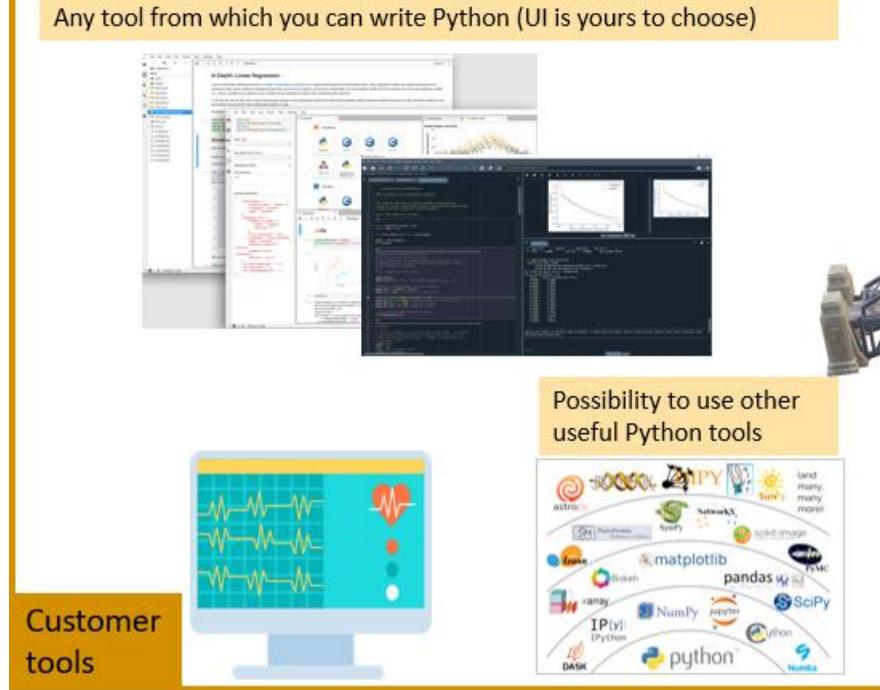
Products



Technology components



Open source ?! What's in it for us?

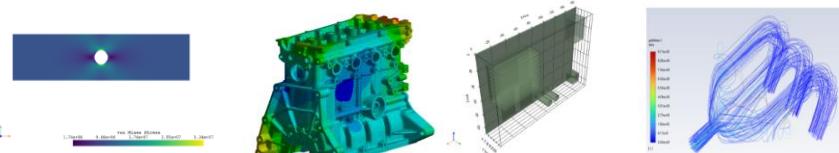


We're just offering a new way of using our products!

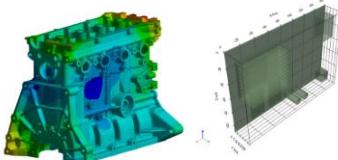
This will allow existing users to use our products differently, effectively and efficiently.

Status of PyAnsys?

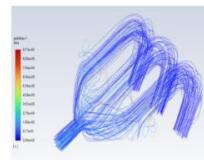
Simulation libraries



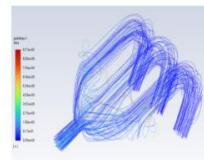
PyMAPDL
Pythonic interface to Ansys
MAPDL (Mechanical APLD)



PyMechanical
Pythonic interface to Ansys
Mechanical

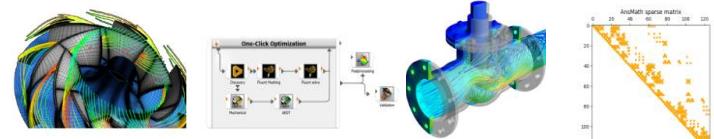


PyAEDT
Pythonic interface to AEDT
(Ansys Electronic Desktop)



PyFluent
Pythonic interface to Ansys
Fluent

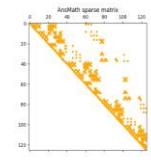
Utility libraries



PyPrimeMesh
Pythonic interface to Ansys
Prime Server, which delivers
core Ansys meshing technology

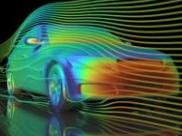
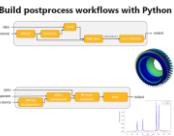
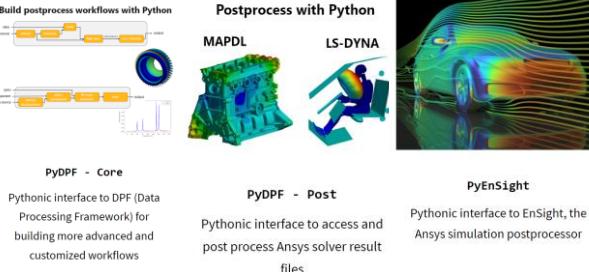
PyOptislang
Pythonic interface to Ansys
Optislang

PySystem Coupling
Pythonic interface to communicate with Ansys
System Coupling
PyAnsys Math
Pythonic interface to PyAnsys
Math libraries



PyDynamicReporting
Pythonic interface to Ansys
Dynamic Reporting for service
and control of its database and
reports

Post-processing libraries



✓ PyMAPDL

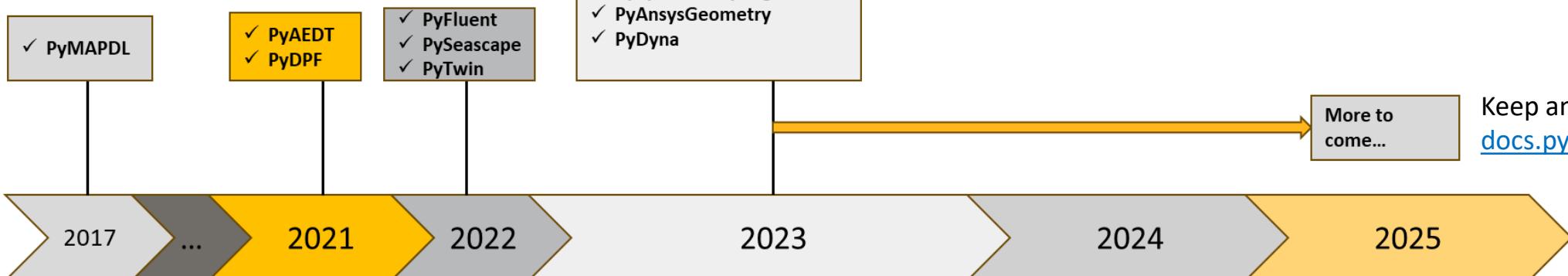
✓ PyAEDT
✓ PyDPF

✓ PyFluent
✓ PySeascape
✓ PyTwin

- ✓ PyEnSight
- ✓ PyDynamicReporting
- ✓ PySherlock
- ✓ Grantami
- ✓ PyMechanical
- ✓ PyOptislang
- ✓ PyMotorCad
- ✓ PySystemCouplings
- ✓ PyAnsysGeometry
- ✓ PyDyna

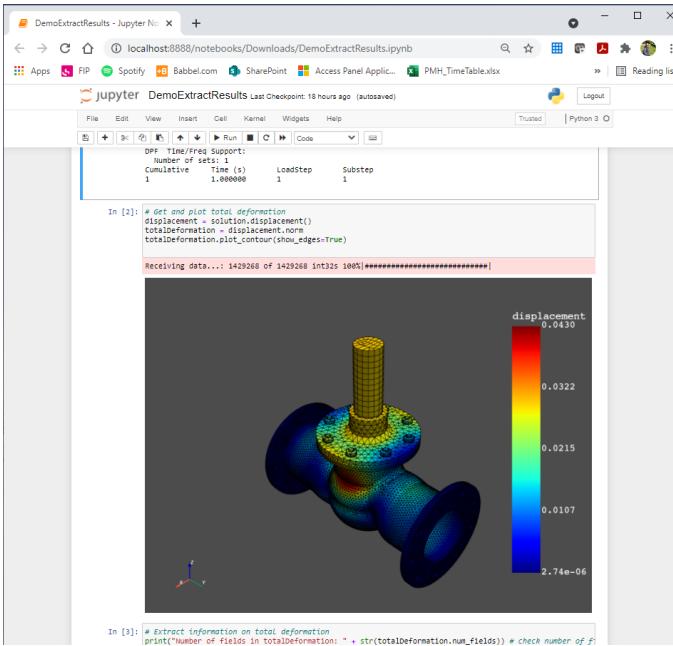
More to come...

Keep an eye on...
docs.pyansys.com

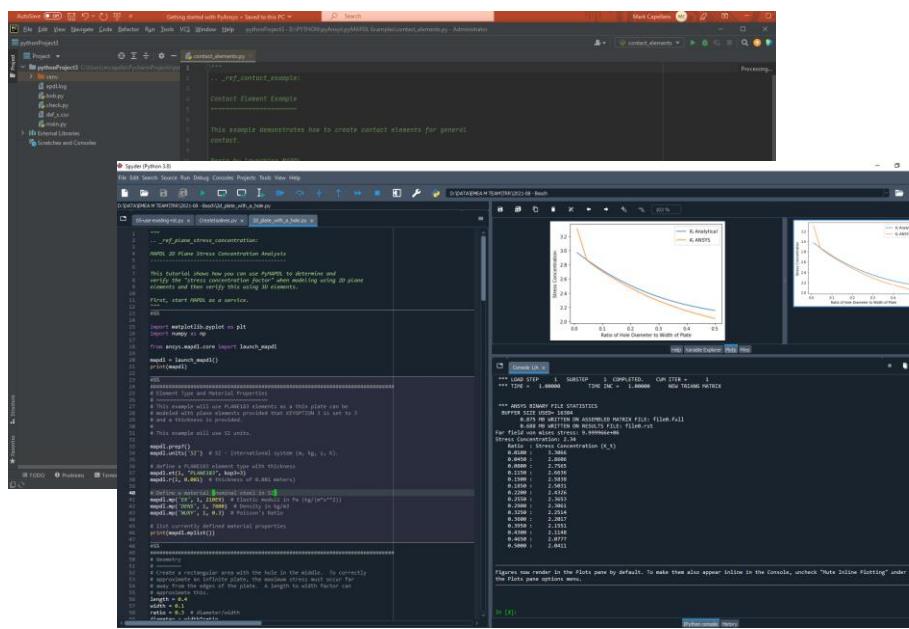


What's the UI like?

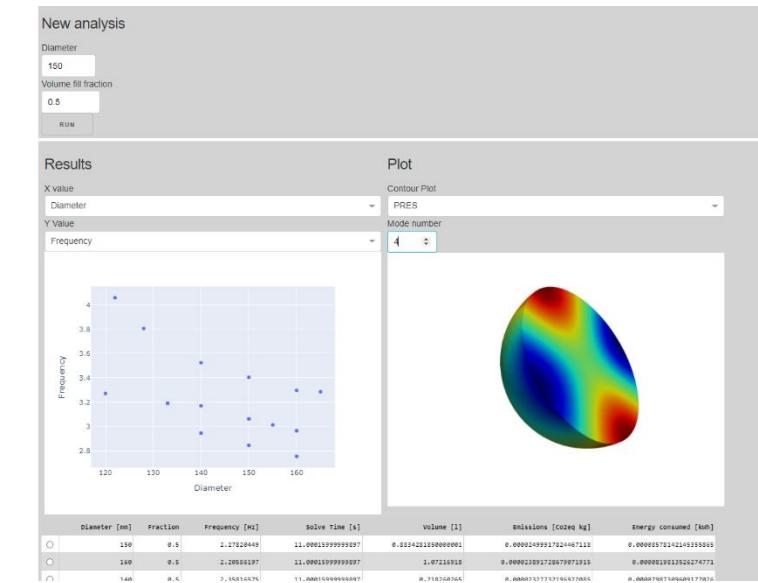
- PyAnsys is a set of tools to interact with Ansys products from a Python interface
 - Any Python interface can be used (even the Windows Command Prompt !)
 - Using an IDE (Integrated Development Environment) is recommended to write and maintain code.
- Many IDEs are available: [Spyder](#), [PyCharm](#), [VS Code](#), [Visual Studio](#),



Code can be written from a web-page thanks to Jupyterlab



An IDE can be used (PyCharm, Spyder, ...)



User can also develop apps by coding both the front end and the back end



What can PyAnsys do for me?

Call Ansys products from external Python environment

This screenshot shows the Spyder Python 3.8 IDE interface. The left pane displays the code for a script named `2d_plate_with_a_hole.py`. The code performs a finite element analysis (FEA) setup for a plate with a hole, calculating stress concentration factors. It uses the PyMAPDL module to define the geometry, material properties, and boundary conditions. The right pane shows the Python console output, which includes usage instructions for DMP-POST and DPF-CORE, and the results of running the script.

```
# # (math) 't' is the thickness of the plate.
# # Experimentally, this is computed by taking the mean of the nodes at
# # the right most side of the plate.
# # We use samean here because mid-side nodes have no stress
# # (see https://ansys.com/doc/api/PyMAPDL/html/_modules/ansys/mapdl/core.html#samean)
# far_field_stress = np.mean(von.mises(maxs))
# print(f"Far field von mises stress: {far_field_stress} MPa")
# stress = far_field_stress * t
# stress_adj = stress / (pi * radius * radius * t)
# stress_adj = far_field_stress * adj
# # The stress concentration is then simply the maximum stress divided
# # by the adjusted far-field stress...
# stress_con = (max_stress/stress_adj)
# print(f"Stress Concentration: {stress_con}")
# print("")

#####
# The above script can be placed within a function to compute the stress concentration
# For a variety of hole diameters.
# For each batch, MAPDL is reset and the geometry is generated from scratch.
# ...

def compute_stress_con(ratio):
    """Compute the stress concentration for plate with a hole Loaded
    with uniform force.
    """
    mapdl.clear("NOSTART")
    mapdl.et(1, "PLANE183")
    mapdl.units("SI") # SI - International system (m, kg, s, K).
    # Define a PLANE183 element type with thickness
    mapdl.et(1, "PLANE183", kopt=3)
    mapdl.t(1, 0.001) # thickness of 0.001 meters
    # Define a material (medium steel)
    mapdl.mp("MATERIAL", 1, "STEEL")
    mapdl.mp("ELEM", 1, 70000) # Modulus of elasticity in Pa (kg/(m*s**2))
    mapdl.mp("POISSON", 1, 0.3) # Poisson's Ratio
    mapdl.endif("MATERIAL", 1)

    # Geometry
    # ...
    # Create a rectangular area with the hole in the middle
    diameter = width/2
    radius = diameter/2
    # Create the rectangle
    rect_num = mapdl.solid(width=length, height=width)
    # Create a circle in the middle of the rectangle
    circ_num = mapdl.cyl(length/2, width/2, radius)
    # Note how pyansys parses the output and returns the area numbers
    # created by each command. This can be used to execute a boolean
```

Complete FE setup, run and solving with PyMAPDL

This screenshot shows the Spyder Python 3.8 IDE interface. The left pane displays the code for a script named `DMPF-POST_and_DPF-CORE_use_existing_RST.py`. The code demonstrates how to use the DMP-POST and DPF-CORE APIs to access and plot data saved in a .rst file. The right pane shows the Python console output, which includes usage instructions for DMP-POST and DPF-CORE, and the results of running the script.

```
## Example of using DMP-POST and DPF-CORE
## This tutorial shows how to use the DMP-POST and DPF-CORE APIs on a same binary result file.
## DMP-POST is used to access and plot data saved in the .rst file.
## DPF-CORE is used to access the data in the .rst, compute new data, and plot it on the FE model.
## ...

#####
# Use DMP-POST to get the solution object
# ...
# The following file is the result of a static analysis computed using
# Ansys Mechanical.
# Here we load the solution
example_path = os.path.join("D:\\PyAnsys\\Examples\\Use_Existing_RST", "use_existing_RST.rst")
solution = post.load_solution(example_path)
# ...
# Check what is the solution object
# ...

print(solution)

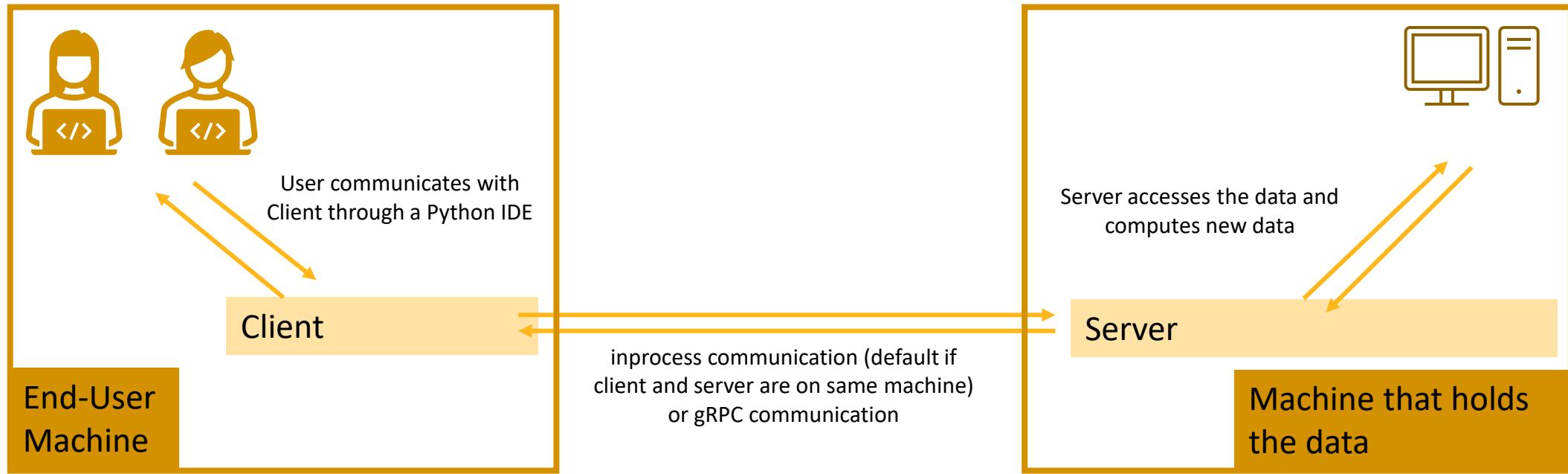
#####
# Get and plot total deformation
# ...
# Extract information on total deformation
# ...
# Print number of fields in total deformation = str(totalDeformation.num_fields)) # check number of fields
totalDeformation = totalDeformation.get_max_data_at_fix(0) # find maximum value
maxTotalDeformation = totalDeformation.get_total_deformation()
minTotalDeformation = totalDeformation.get_min_deformation()
print(f"Max value {maxTotalDeformation} - Min value {minTotalDeformation}) = Total deformation
```

Postprocess a binary result file and create custom results with PyDPF



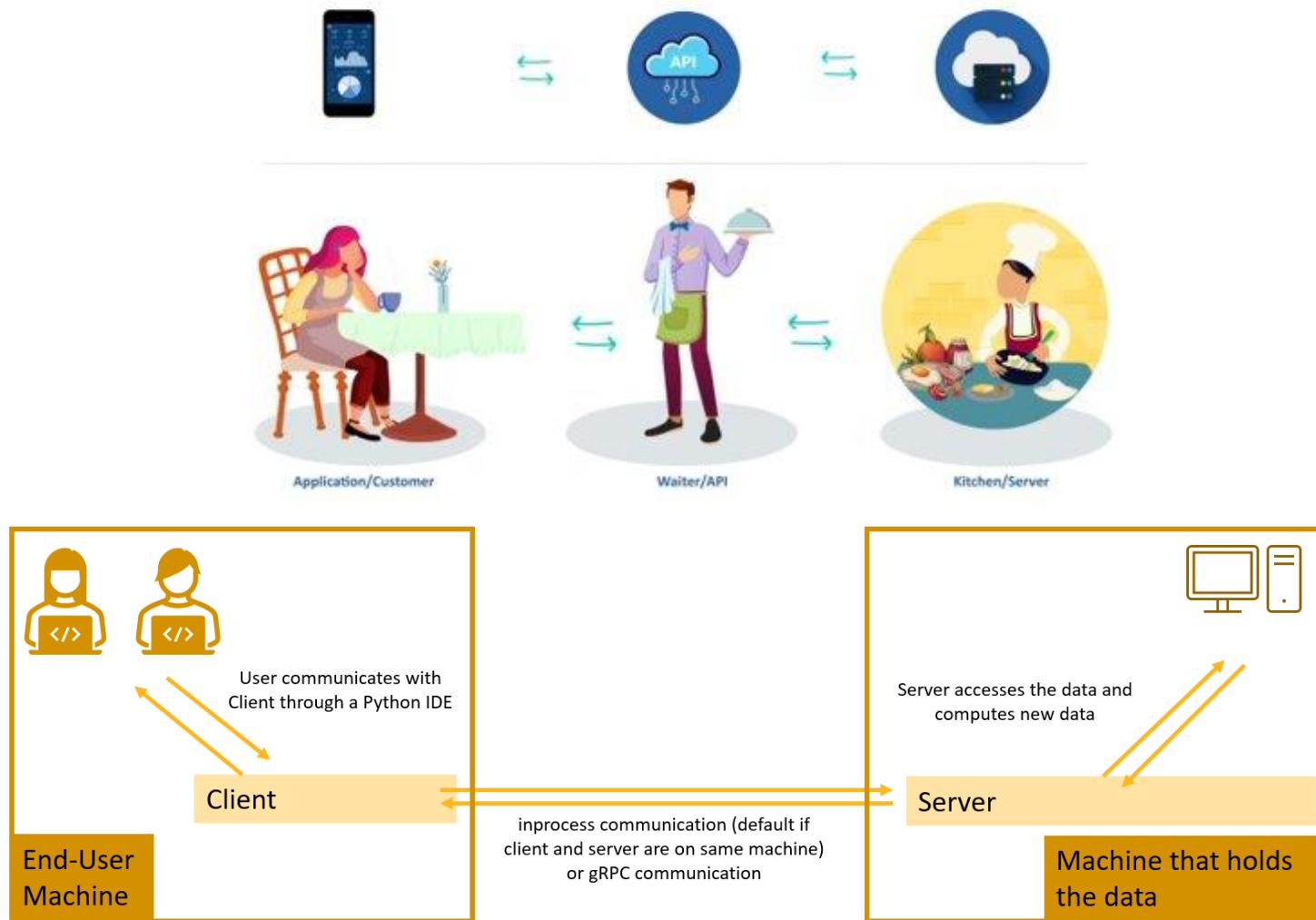
Client/Server architecture

- [PyAnsys project](#) offers several packages that are usually based on a Client/Server architecture. Client and Server can be on the same machine, or on separate/remote machines. Below is a typical configuration:

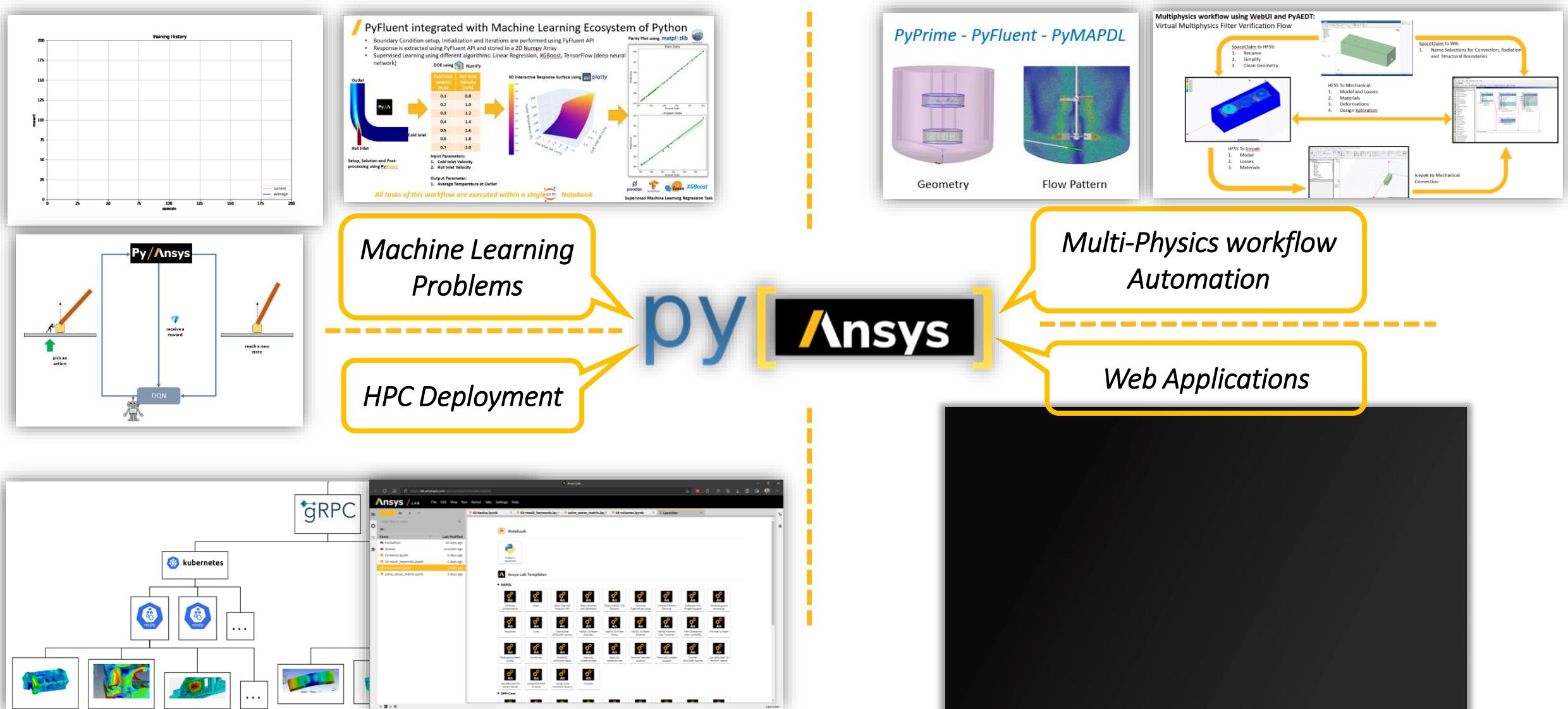


- Where and how to find the files?
 - The Client part comes is provided through Python packages (pip install)
 - The Server part is shipped with the standard Ansys installation

Client/Server architecture

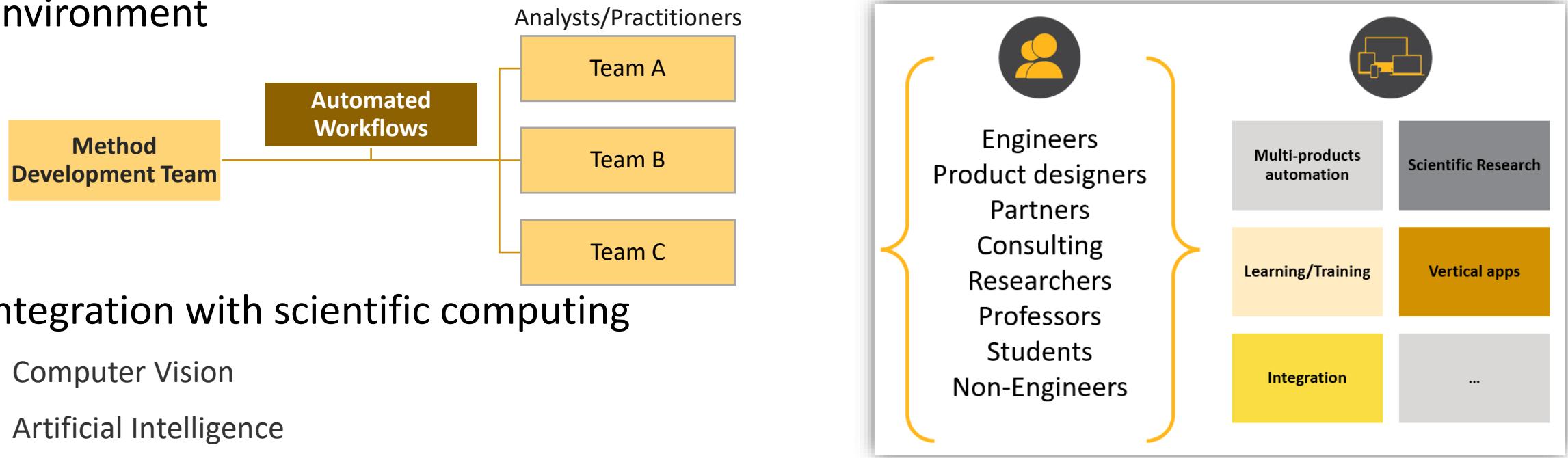


Key workflows where customers are using PyAnsys



Target Personas

- Anyone who wants to **automate** or **integrate** their simulation workflows from outside an Ansys environment



- Integration with scientific computing

- Computer Vision
- Artificial Intelligence
- Machine Learning
- Data processing and visualization
- Optimization

How does PyAnsys Makes Difference?

- Allows Control of Ansys tool from external script
- Enables use of Cpython libraries
- Clean, Concise and Maintainable
- Open Source and Well-Documented
- Favorable for AI Pair Programming Tools
- Easy to Deploy and Scalable Workflows
- Enables easy integration with AI/ML Framework
- ...Many more!!



How can I install PyAnsys?



How to install PyAnsys?

Follow those three steps:

1. Install Ansys Products from standard installation (Mechanical, Fluent, etc.)
2. Install Python and the IDE (code editor) of your choice
3. Install the PyAnsys packages

More info on each step in the next slides!

Alternative to step 2 and 3: use the [Ansys Python Manager](#)

How to install it? Step 1: Install Ansys Products

- Download Ansys Products from the [Customer Portal](#)
- Using the latest release is recommended to get latest features
- Supported versions:
 - PyMAPDL: Ansys 17.0 and onwards
 - PyDPF: Ansys 2021R1 and onwards

Downloads: Current Release - 2023 R1

Select Release: 2023 R1 Select Operating System: Windows x64

Windows x64 packages are displayed

Select Download Type: Primary Packages

Primary Packages (Commercial & Academic Packages)

Structures	Fluids	Electronics	PrePost	Discovery
Full Package	Full Package	Full Package Motor-CAD	Full Package	Full Package

How to install it? Step 2: Install Python and an IDE

- Python version 3.8 or newer is required. Installation from python.org is recommended.
- Use the IDE you are most comfortable with. Some suggestions:
 - [Spyder](#): Free and open-source scientific environment
 - [PyCharm](#): Offers a free community version and full-fledged professional one
 - [Visual Studio Code](#): Free open-source code editor
 - [JupyterLab](#): Web-based interactive development environment

How to install it? Step 3: Install the PyAnsys packages

- Creating and using a virtual environment (venv) is recommended whenever working on different Python based projects. This allows the dependencies of each project to be isolated from each other.
- Python packages are usually installed through “pip” by opening a command prompt and typing “`pip install package_name`”. PyAnsys packages follow the same procedure:

```
# To install all PyAnsys libraries:  
pip install pyansys
```

```
# To install each library individually:  
pip install ansys-mapdl-core  
pip install ansys-dpf-core  
pip install ansys-dpf-post  
pip install ansys-dpf-composites  
pip install ansys-meshing-prime[all]  
pip install ansys-mechanical-core
```

Note: It is also possible to download the wheels from Github.

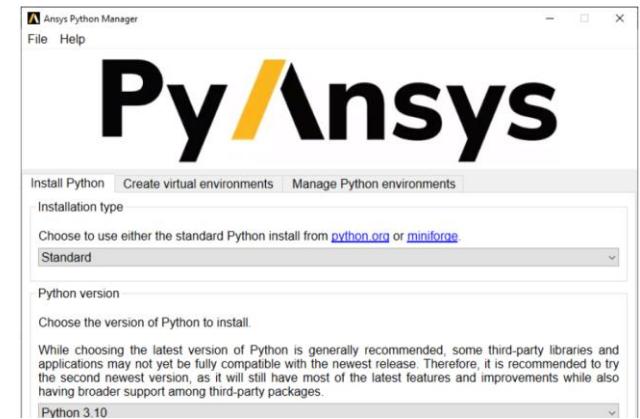


Ansys Python Manager

- Free, open-source tool
- Seamlessly install Python, PyAnsys packages, and common Python libraries
- Facilitates creation and management of Python virtual environments
- Useful links:
 - [GitHub repository](#)
 - [Executable](#)
 - Documentation: <https://installer.docs.pyansys.com/>
 - If you encounter any bugs or have any suggestions for enhancements, [log an issue](#) directly in the GitHub repository.

Ansys Python Manager (QT)

This is a simple cross-platform QT application you can use to install Python and (optional) PyAnsys packages.



How to launch PyAnsys?

- From a Python IDE, import the package you need and launch it

```
>>> from ansys.mapdl.core import launch_mapdl
>>> mapdl = launch_mapdl()
>>> print(mapdl)
```

Product: ANSYS Mechanical Enterprise
MAPDL Version: RELEASE 2021 R1 BUILD 21.0
PyMAPDL Version: Version: 0.57.0

PyMAPDL

```
from ansys.dpf.composites.composite_model import CompositeModel, CompositeScope
from ansys.dpf.composites.constants import FailureOutput
from ansys.dpf.composites.example_helper import get_continuous_fiber_example_files
from ansys.dpf.composites.failure_criteria import (
    CombinedFailureCriterion,
    CoreFailureCriterion,
    MaxStrainCriterion,
    MaxStressCriterion,
    VonMisesCriterion,
)
from ansys.dpf.composites.server_helpers import connect_to_or_start_server
server = connect_to_or_start_server()
composite_files_on_server = get_continuous_fiber_example_files(server, "shell")
```

PyDPF-Composites

```
>>> from ansys.dpf.core import Model
>>> from ansys.dpf.core import examples
>>> model = Model(examples.find_simple_bar())
>>> print(model)
```

PyDPF-Core

```
import ansys.meshing.prime as prime
with prime.launch_prime() as prime_client:
    model = prime_client.model
```

PyPrimeMesh

```
>>> from ansys.dpf import post
>>> from ansys.dpf.post import examples
>>> simulation = post.load_simulation(examples.download_crankshaft())
>>> displacement = simulation.displacement()
>>> print(displacement)
```

PyDPF-Post

```
import os
from ansys.mechanical.core import launch_mechanical

mechanical = launch_mechanical()
```

PyMechanical (remote session and embedded instance)

```
from ansys.mechanical.core import App
app = App()
ns = app.DataModel.Project.Model.AddNamedSelection()
```

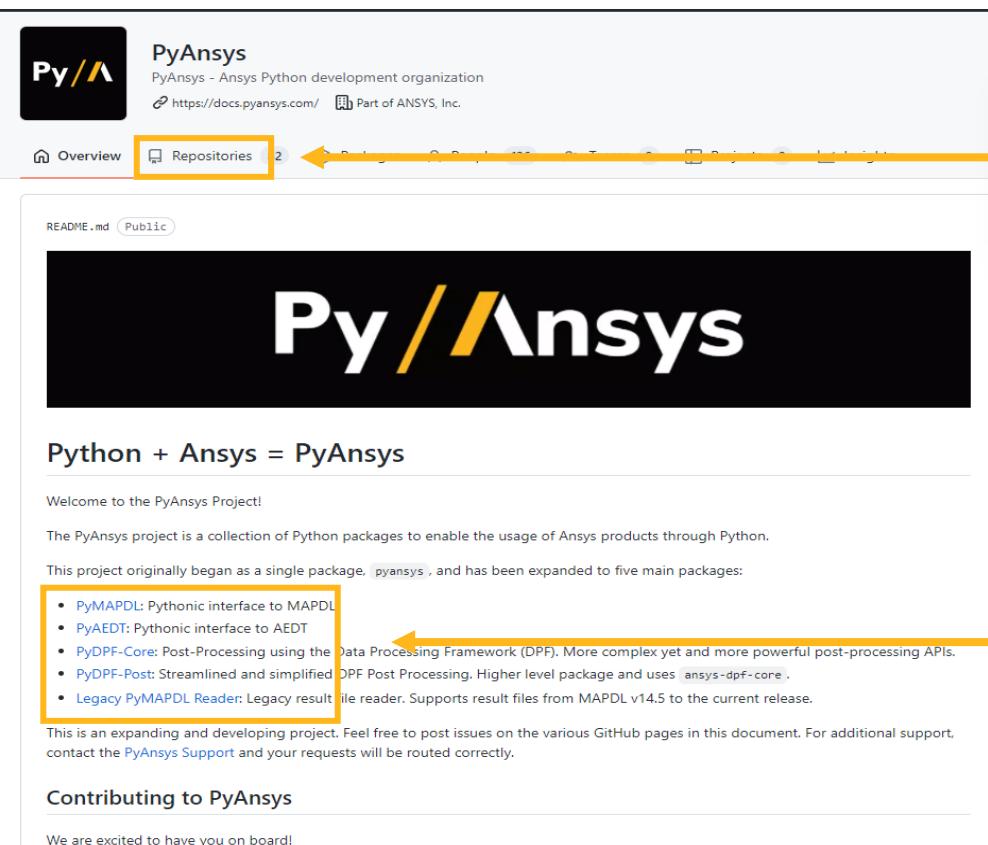


Where are the source code
and documentation?
How do I report a bug?
Where can I find help?

Source code and Documentation

PyAnsys is an open-source project. The source code and documentation are provided on Github.

<https://github.com/ansys>

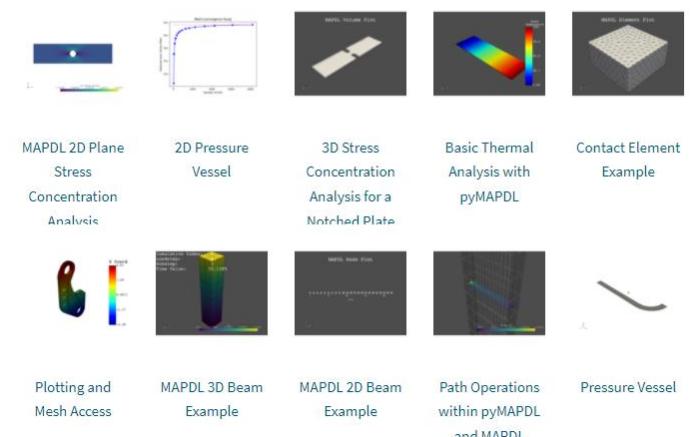


The screenshot shows the GitHub repository page for PyAnsys. The top navigation bar has a yellow arrow pointing to the 'Repositories' tab, which is highlighted with a yellow border. Below the navigation bar, there's a large banner with the text 'Python + Ansys = PyAnsys'. The main content area contains sections for 'README.md' (Public), 'MAPDL 2D Plane Stress Concentration Analysis', '2D Pressure Vessel', '3D Stress Concentration Analysis for a Notched Plate', 'Basic Thermal Analysis with pyMAPDL', 'Contact Element Example', 'MAPDL 3D Beam Example', 'MAPDL 2D Beam Example', 'Path Operations within pyMAPDL and MAPDL', and 'Pressure Vessel'. There are also sections for 'Plotting and Mesh Access', 'Legacy PyMAPDL Reader', 'Contributing to PyAnsys', and 'Getting Started'.

Source Code

Full Examples Using PyMAPDL

These examples demonstrate full examples using the PyMAPDL module.



Documentation

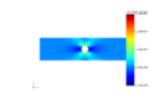
Each package has documentation and many examples



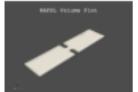
Many examples available

Full Examples Using PyMAPDL

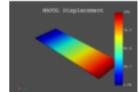
These examples demonstrate full examples using the PyMAPDL module.



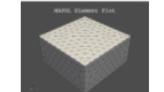
MAPDL 2D Plane Stress Concentration Analysis



3D Stress Concentration Analysis for a Notched Plate



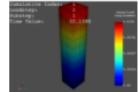
Basic Thermal Analysis with pyMAPDL



Contact Element Example



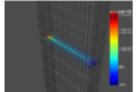
Plotting and Mesh Access



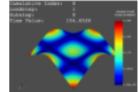
MAPDL 3D Beam Example



MAPDL 2D Beam Example



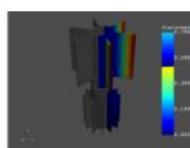
Path Operations within pyMAPDL and MAPDL



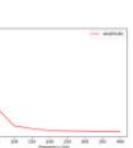
PyVista Mesh Integration

Advanced and Miscellaneous Examples

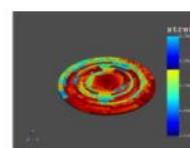
These demos show advanced use cases demonstrating high level of workflow customization



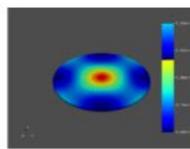
Multi-stage Cyclic Symmetry Use Advanced Customization



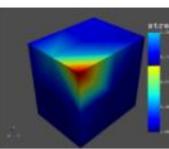
Solve Harmonic Problem (with damping) Using Matrix Inverse



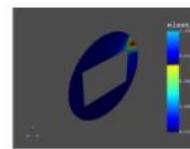
Average Elemental Stress on a given volume



Exchange Data Between Servers



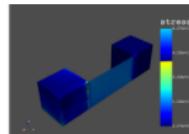
Extrapolation Method for stress result of 3D-element



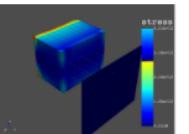
Extrapolation Method for strain result of 2D-element

DPF-Post Analysis Types

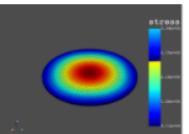
Here are a series of analysis type focused examples with [ansys-dpf-post](#) module.



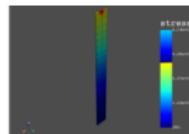
Static Analysis



ANSYS DPF-Post: Modal Analysis



ANSYS DPF-Post: Harmonic Analysis



ANSYS DPF-Post: Transient Analysis

New examples are being added constantly:

- <https://mapdldocs.pyansys.com/>
- <https://dpfdocs.pyansys.com/>
- <https://postdocs.pyansys.com/>

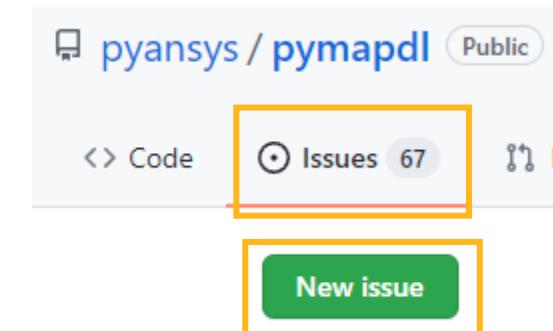
Documentation Demo

Report a bug, Suggest an enhancement, Want to contribute?

This should be done through Github: open an issue from the Github page of each package.

<https://github.com/ansys>

The screenshot shows the GitHub repository page for PyAnsys. The 'Repositories' tab is selected and highlighted with an orange box. The main content area features the PyAnsys logo and a brief description: "Python + Ansys = PyAnsys". Below this, there's a welcome message and a list of five main packages: PyMAPDL, PyAEDT, PyDPF-Core, PyDPF-Post, and Legacy PyMAPDL Reader. At the bottom, there's a "Contributing to PyAnsys" section and a note about having new members on board.



Need help? Use the Ansys Developer Portal and Developer Forum

The screenshot shows the homepage of the Ansys Developer website. At the top, there's a navigation bar with links for Documentation, Knowledge Base, Forum, GitHub, a search bar, and a login button. Below the header, a large banner features the text "Ansys Developer" and "Engineer new solutions to automate, manage, and extend the power of simulation workflows". There are two buttons: "WHAT IS SIMULATION >" and "EXPLORE OUR TOOLS >". The main content area is divided into three sections: "Start Building Here", "Master The Basics", and "A Community For You". Each section has a small icon (a house, a gear, and a person), a title, and a brief description. At the bottom, there are links to "TO THE DOCUMENTATION >", "VISIT THE KNOWLEDGE BASE >", and "BROWSE THE FORUMS >".

<https://developer.ansys.com/>

The screenshot shows a forum page titled "HOME > ENGINEERING SIMULATION". It displays a list of posts with titles, user profiles, and metadata. The first post is about controlling contour in PyDPF plots, the second is about retrieving messages from Mechanical through ACT scripting, the third is about getting results with DPF for specific frequencies, and the fourth is about reading files using Python. Each post includes a reply count, view count, and a timestamp.

how to control contour in PyDPF plot
https://dpf.docs.pyansys.com/examples has quite some useful example but I couldn't find anythin on how to manipulate the contour. For example banded/smooth, number of bands, their colors, min and max ...
Answered ✓ PyDPF 17 views 1 comments 0 reactions Started by Pavel Most recent by Ayush Kumar Jan 22, 2023 Engineering Simulation

Retrieve messages from Mechanical
Through ACT scripting, how can I retrieve the info, warning and error messages in Mechanical ?
Answered ✓ AnsysACT 18 views 3 comments 0 reactions Started by Pennelle Marone-Hitz Most recent by Pennelle Marone-Hitz Jan 20, 2023 Engineering Simulation

How to get results with DPF for a time/frequency not available in my rst file?
For instance, let's imagine that our harmonic analysis contains results for frequency 12Hz and 15Hz. If I want to get a result for 14Hz and 14.5Hz, how can I do it?
Answered ✓ DPF 17 views 1 comments 0 reactions Started by Javier Vique Most recent by Javier Vique Jan 17, 2023 Engineering Simulation

How can I read a file using Python?
How can I read a file using Python?
Answered ✓ IronPython 53 views 6 comments 2 reactions Started by Pennelle Marone-Hitz Most recent by Pennelle Marone-Hitz Jan 12, 2023 Engineering Simulation

<https://discuss.ansys.com/>



Training documents

Some PyAnsys courses or other useful courses are available on the [Ansys Innovation Space](#) and the [Ansys Learning Hub](#)

STRUCTURES

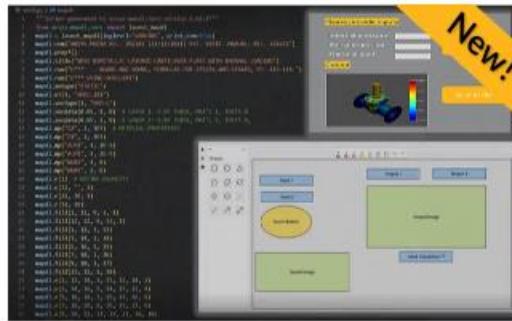


[LEARN SIMULATION](#)

Getting Started with Ansys
PyMAPDL

[LEARN MORE →](#)

STRUCTURES



[LEARN SIMULATION](#)

Developing WebApps for Modeling
and Simulation Using PyAnsys

[LEARN MORE →](#)

PYTHON



[LEARN SIMULATION](#)

Intro to Python

[LEARN MORE →](#)

STRUCTURES



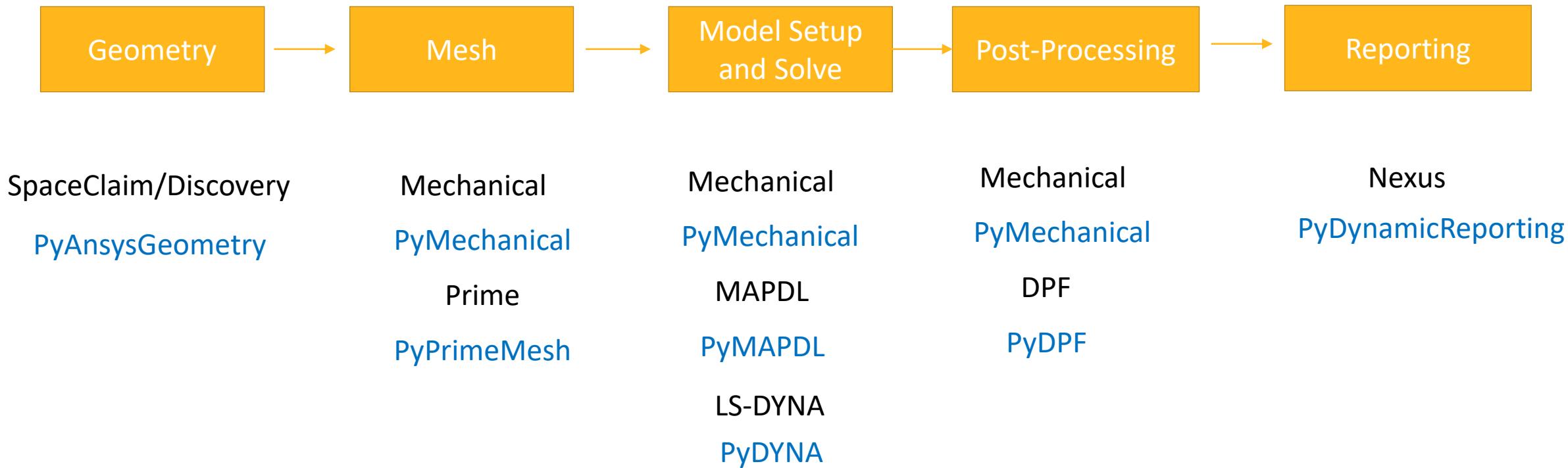
[LEARN SIMULATION](#)

Intro to Ansys APDL Scripting

[LEARN MORE →](#)

Overview of Modules

A Typical Structural Simulation Workflow

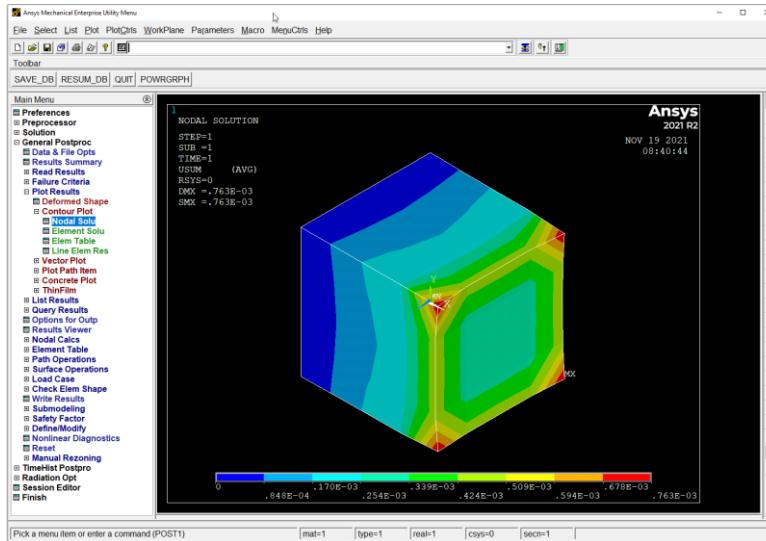


PyMAPDL

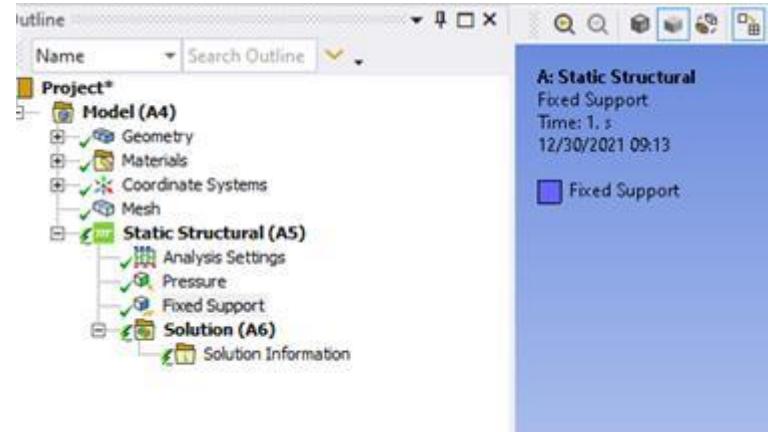


MAPDL = Mechanical Ansys Parametric Design Language

Scripting language used for the Ansys Mechanical solver



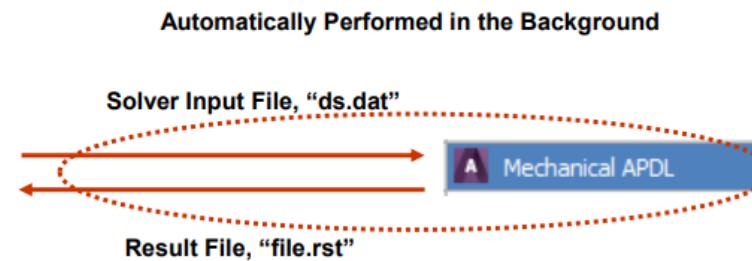
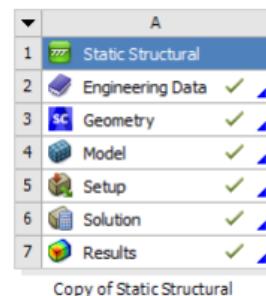
Ansys Classic



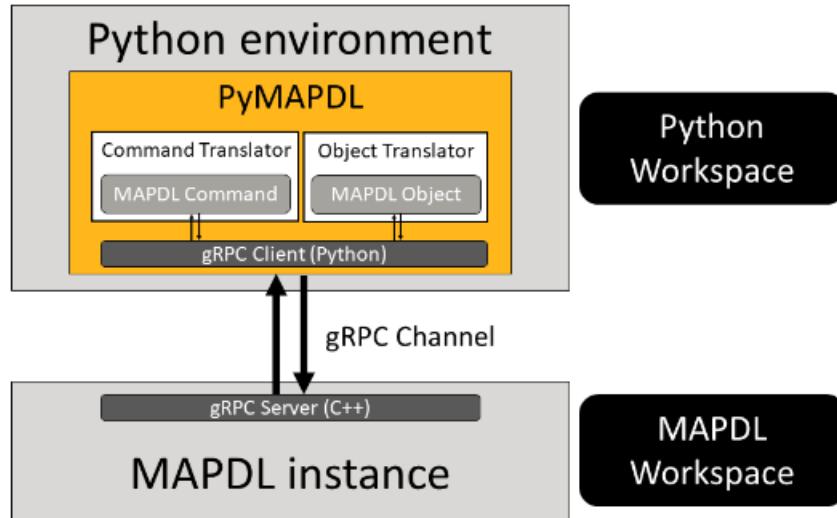
Mechanical

```
/PREP7
x = 1
y = x
z = x
f = 1e9
MP , EX , 1 , 2.1E+14
MP , PRXY , 1 , 0.3
BLOCK , 0 , x , 0 , y , 0 , z
ET , 1 , 186
ESIZE , x/4
VMESH, ALL
NSEL , S , LOC , X , 0 ,
D , ALL , ALL
NSEL , S , LOC , X , x
F,ALL,FX,f
ALLSEL , ALL
/SOL
SOLVE
FINISH
/POST1
PLDISP,2
```

Both rely on MAPDL

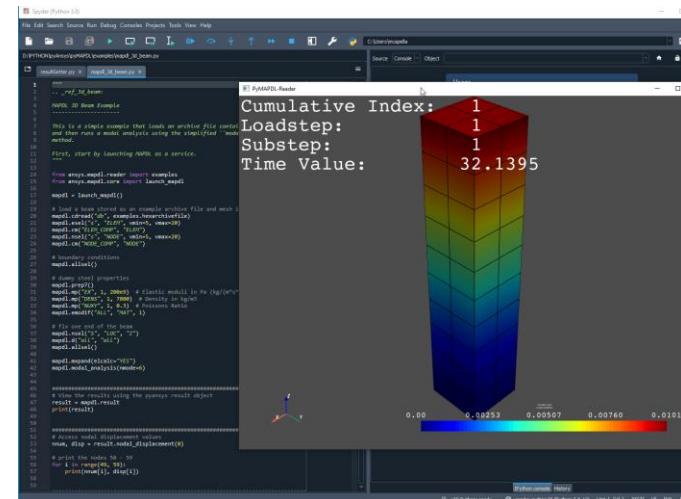


From MAPDL to PyMAPDL



PyMAPDL architecture diagram

- ❑ Use MAPDL in a Python environment
- ❑ Interface of users' choice
- ❑ Python ease of use:
 - *do/*enddo -> for i in data
 - *get -> mapdl.parameters["MY_ARRAY"]
- ❑ PyMAPDL can be combined with any other python packages



PyMAPDL

- With PyMAPDL, it's possible to:
 - Create geometry, mesh, model setup
 - Do interactive plotting
 - Post-process
 - Translate MAPDL scripts to PyMAPDL script

- Either use standard APDL commands through "Run" command, or call MAPDL Pythonically:

```
mapdl.run('/PREP7')
mapdl.run('K, 1, 0, 0, 0')
mapdl.run('K, 2, 1, 0, 0')
mapdl.run('K, 3, 1, 1, 0')
mapdl.run('K, 4, 0, 1, 0')
mapdl.run('L, 1, 2')
mapdl.run('L, 2, 3')
mapdl.run('L, 3, 4')
mapdl.run('L, 4, 1')
mapdl.run('AL, 1, 2, 3, 4')
```

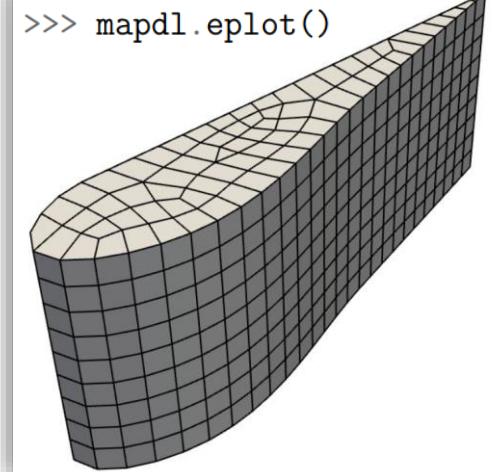
```
mapdl.prep7()
mapdl.k(1, 0, 0, 0)
mapdl.k(2, 1, 0, 0)
mapdl.k(3, 1, 1, 0)
mapdl.k(4, 0, 1, 0)
mapdl.l(1, 2)
mapdl.l(2, 3)
mapdl.l(3, 4)
mapdl.l(4, 1)
mapdl.al(1, 2, 3, 4)
```

PyAnsys Code Example: Meshing

```
>>> mapdl.et(1, 'SOLID186')
>>> mapdl.vsweep('ALL')
>>> mapdl.esize(0.1)
>>> mapdl.mesh
```

ANSYS Mesh

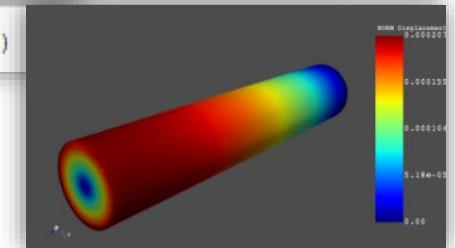
Number of Nodes:	7217
Number of Elements:	2080
Number of Element Types:	2
Number of Node Components:	0



```
POST1:
PRNSOL, U, X
PRINT U NODAL SOLUTION PER NODE
***** POST1 NODAL DEGREE OF FREEDOM LISTING *****
LOAD STEP= 1 SUBSTEP= 1
TIME= 1.0000 LOAD CASE= 0
THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM
NODE UX
1 0.10751E-003
2 0.85914E-004
3 0.57069E-004
4 0.13913E-003
5 0.35621E-004
6 0.52186E-004
7 0.30417E-004
8 0.26110E-004
```

```
>>> mapdl.set(1, 1)
>>> disp_x = mapdl.post_processing.nodal_displacement('X')
array([1.07512979e-04, 8.59137773e-05, 5.70690047e-05, ...,
       5.70333124e-05, 8.58600402e-05, 1.07445726e-04])
```

```
>>> mapdl.post_processing.plot_nodal_displacement('X')
```



Demo PyMAPDL – evaluate stress concentration factor on a 2D plate with a hole

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays a script named `2d_plate_with_a_hole.py`. The code uses PyANSYS to model a 2D plate with a central hole, calculating the stress concentration factor. It includes comments explaining the physics and the use of the far-field stress as a reference. The right side of the interface shows the IPython console output, which includes a help dialog for the `Usage` command and the actual command-line output for Python 3.9.7.

```
# - :math:`t` is the thickness of the plate.
#
# Experimentally, this is computed by taking the mean of the nodes at
# the right-most side of the plate.
#
# We use nanmean here because mid-side nodes have no stress
mask = result.mesh.nodes[:, 0] == length
far_field_stress = np.nanmean(von_mises[mask])
print('Far field von mises stress: %e' % far_field_stress)
# Which almost exactly equals the analytical value of 1000000.0 Pa
#
#####
# Since the expected nominal stress across the cross section of the
# hole will increase as the size of the hole increases, regardless of
# the stress concentration, the stress must be adjusted to arrive at
# the correct stress. This stress is adjusted by the ratio of the
# width over the modified cross section width.
adj = width/(width - diameter)
stress_adj = far_field_stress*adj
#
# The stress concentration is then simply the maximum stress divided
# by the adjusted far-field stress.
stress_con = (max_stress/stress_adj)
print('Stress concentration: %2f' % stress_con)
#
#####
# ~~~~~
# The above script can be placed within a function to compute the stress concentration
# for a variety of hole diameters.
# For each batch, MAPDL is reset and the geometry is generated from scratch.
#
def compute_stress_con(ratio):
    """Compute the stress concentration for plate with a hole loaded
    with a uniaxial force.
    """
    mapdl.clear('NOSTART')
    mapdl.prep7()
    mapdl.units('SI') # SI - International system (m, kg, s, K).
    #
    # Define a PLANE183 element type with thickness
    mapdl.et(1, "PLANE183", kopt=3)
    mapdl.r(1, 0.001) # thickness of 0.001 meters
    #
    # Define a material (nominal steel in SI)
    mapdl.mp('EX', 1, 210e9) # Elastic moduli in Pa (kg/(m*s**2))
    mapdl.mp('DENS', 1, 7800) # Density in kg/m³
    mapdl.mp('NUXY', 1, 0.3) # Poisson's Ratio
    mapdl.emodif('ALL', 'MAT', 1)
    #
    # Geometry
    #
    # Create a rectangular area with the hole in the middle
    diameter = width*ratio
    radius = diameter*0.5
    #
    # create the rectangle
    rect_num = mapdl.blc4(width=length, height=width)
    #
    # create a circle in the middle of the rectangle
    circ_num = mapdl.cyl4(length/2, width/2, radius)
    #
    # Note how pyansys parses the output and returns the area numbers
    # created by each command. This can be used to execute a boolean
```

Demo PyMAPDL – evaluate stress concentration factor on a 2D plate with a hole

```
import matplotlib.pyplot as plt
import numpy as np

from ansys.mapdl.core import launch_mapdl

mapdl = launch_mapdl()
print(mapdl)

#%%
#####
# Element Type and Material Properties
# ~~~~~
# This example will use PLANE183 elements as a thin plate can be
# modeled with plane elements provided that KEYOPTION 3 is set to 3
# and a thickness is provided.
#
# This example will use SI units.

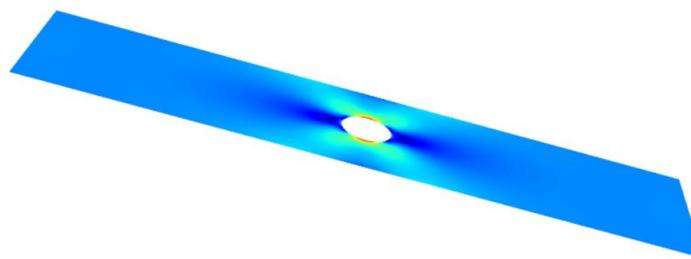
mapdl.prep7()
mapdl.units('SI') # SI - International system (m, kg, s, K).

# define a PLANE183 element type with thickness
mapdl.et(1, "PLANE183", kop3=3)
mapdl.r(1, 0.001) # thickness of 0.001 meters

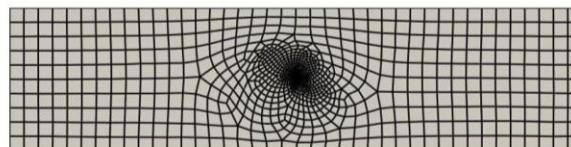
# Define a material (nominal steel in SI)
mapdl.mp('EX', 1, 210E9) # Elastic moduli in Pa (kg/(m*s**2))
mapdl.mp('DENS', 1, 7800) # Density in kg/m3
mapdl.mp('NUXY', 1, 0.3) # Poisson's Ratio

# list currently defined material properties
print(mapdl.mplist())
```

Model definition with PyMAPDL

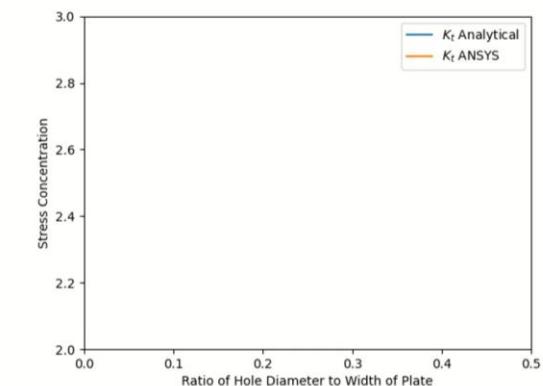


Interactive plotting (mesh, results, etc)



```
Far field von mises stress: 9.999966e+06
Stress Concentration: 2.34
Ratio : Stress Concentration (K_t)
0.0100 : 3.3066
0.0450 : 2.8606
0.0800 : 2.7565
0.1150 : 2.6636
0.1500 : 2.5838
0.1850 : 2.5031
0.2200 : 2.4326
0.2550 : 2.3653
0.2900 : 2.3061
0.3250 : 2.2514
0.3600 : 2.2017
0.3950 : 2.1551
0.4300 : 2.1148
0.4650 : 2.0777
0.5000 : 2.0411
```

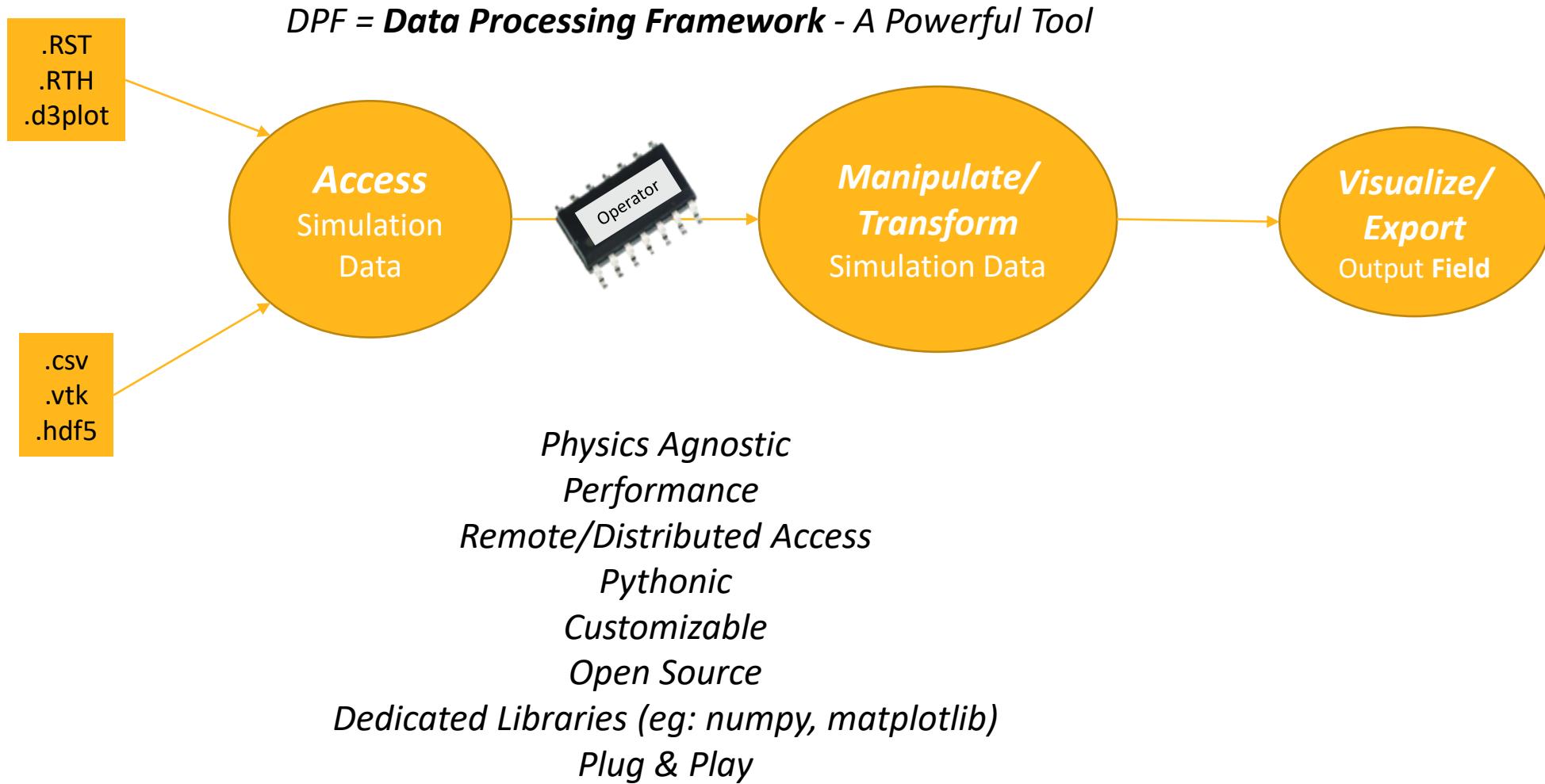
Batch analysis to determine Stress concentration factor (ratio of maximum stress at hole to the far field stress)



PyDPF



What is DPF?

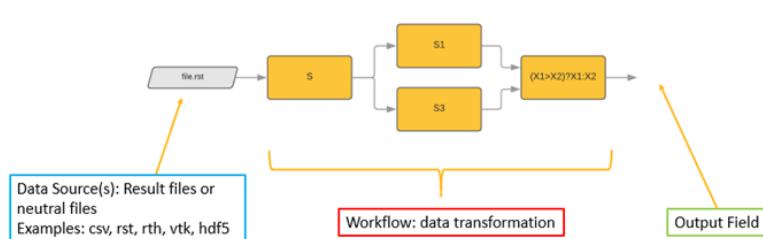


DPF in PyAnsys

3 packages based on DPF in PyAnsys

DPF Core

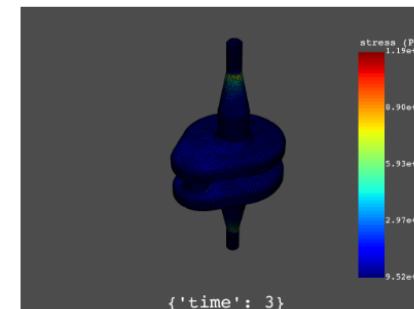
Read and manipulate finite element data with powerful, scalable operators



DPF Post

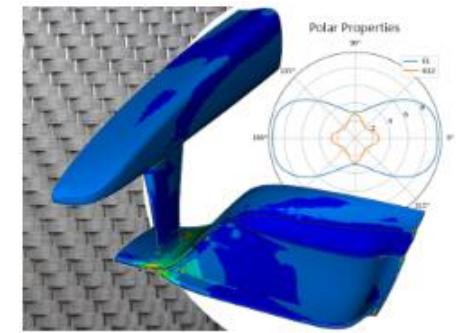
*Simplified post-processing API,
Uses DPF-Core Behind the scenes*

```
>>> stress_eqv = simulation.stress_eqv_von_mises_nodal()  
>>> stress_eqv.plot()
```



DPF Composites

*Additional classes for short fiber and layered composites
(shell and solid elements)*



Training material

- Ansys Learning Hub:
 - Introduction to Ansys Data Processing Framework

Overview

This course teaches the essential skills you will need to perform data processing using Ansys Data Processing Framework. The aim of the course is that you become autonomous in creating user defined workflows for your data processing (specific results post processing for example). DPF offers limitless possibilities for data transformation, learn how you could take benefit from this.

Instructor demonstrations and hands-on workshops cover every aspect you will need to create your own DPF workflows.

The course covers DPF as a stand-alone tool as well as DPF in Mechanical.

This course proposes a complete description of DPF concepts. It also specifies in which situations DPF can be used. It shows the expected outputs when using DPF in Mechanical and / or DPF as a stand-alone tool.

This course illustrates the use of DPF operators and how to chain them to create workflows. It also teaches get the right information for script creation, from the help documentation.

Module 03 covers DPF as a stand-alone tool. **Module 01** to **Module 03** do not require any knowledge about Ansys Mechanical. This learning path is accessible to users from any discipline.

Module 04 covers DPF in Mechanical.

Click here for [Prerequisites and Learning Outcome](#)

The screenshot shows the course landing page with tabs for 'Download Complete Course', 'Enroll in Instructor Led Class', 'Explore Training Material', and 'Upcoming Classes'. Below these are two large icons: 'COURSE MATERIALS EN 2021R2' and 'COURSE DATES'. To the right, there's a 'Folder' section for 'Mod-Agenda' by Meghana Kolathar, created about 1 year ago, with a 'Go To Folder' link. At the bottom right is a 'Go To Events' button.

- PyDPF documentation and examples:
 - PyDPF-Post
 - PyDPF-Core
 - PyDPF-Composites

PyDPF-Post

The Data Processing Framework (DPF) is designed to provide numerical simulation users/engineers with a toolbox for accessing and transforming simulation data.

The Python `ansys.dpf.post` package provides a high level, physics oriented API for postprocessing. Loading a simulation (defined by its result files) allows you to extract simulation metadata as well as results and apply postprocessing operations on it.

This module leverages the PyDPF-Core project's `ansys.dpf.core` package which can be found by visiting [PyDPF-Core GitHub](#). Use `ansys.dpf.core` for building more advanced and customized workflows using Ansys DPF.

Brief demo

Provided you have Ansys 2023 R1 installed, a DPF server starts automatically once you start using PyDPF-Post. Loading a simulation for a MAPDL result file to extract and post-process results:

```
>>> from ansys.dpf import post
>>> # Load a simple post-processing example
>>> simulation = post.load_simulation(examples.download_crankshaft())
>>> displacement = simulation.displacement()
>>> print(displacement)
```

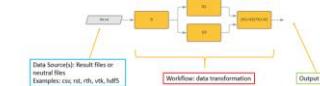
PyDPF-Core

The Data Processing Framework (DPF) provides numerical simulation users and engineers with a toolbox for accessing and transforming simulation data. With DPF, you can perform complex preprocessing or postprocessing of large amounts of simulation data within a simulation workflow.

DPF is an independent, physics-agnostic tool that you can plug into many apps to both data input and data output, including visualization and result plots. It can access data from solver result files and other neutral formats, such as CSV, HDF5, and VTK files.

Using the many DPF operators that are available, you can manipulate and transform this data. You can also chain operators together to create simple or complex data-processing workflows that you can reuse for repeated or future evaluations.

The data in DPF is defined based on physics-agnostic mathematical quantities described in self-sufficient entities called fields. This allows DPF to be a modular and easy-to-use tool with a large range of capabilities.



PyDPF Composites

PyDPF Composites is a Python wrapper for Ansys Composites. It implements classes on top of DPF Composites operators and data accessors for short fiber and layered composites (layered shell and solid elements). This module can be used to postprocess fiber reinforced plastics and layered composites, and to implement custom failure criteria and computation.

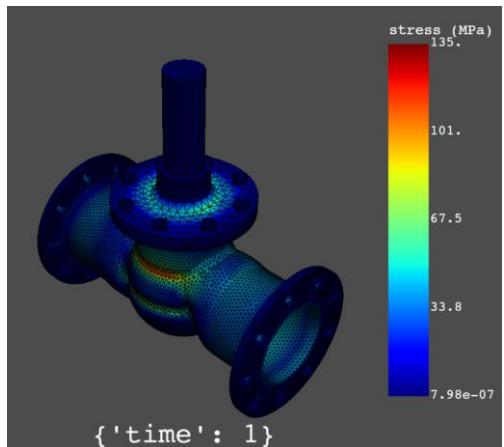
Getting started
Contains installation instructions and a simple example to create a failure plot from a Workbench project.

Examples
Demonstrate the use of PyDPF Composites for various workflows.

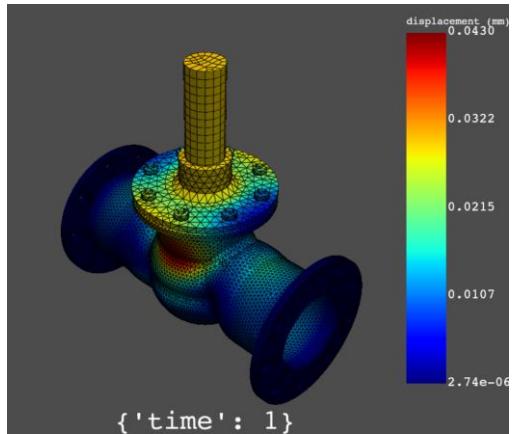
API reference
Describes the public Python classes, methods, and functions.

Contribute
Provides developer installation and usage information.

PyDPF-Post – Simplified API for postprocessing



Max value is: 0.04295543409307043 and min value is: 2.7364042403964693e-06



```
#####
#%%
# Get started
# ~~~~~
from ansys.dpf import post
import os

#####
# Use DPF-POST to get the solution object
# ~~~~~
# The following file is the result of a static analysis computed using
# Ansys Mechanical.
#
# Here we load the solution
example_path = os.path.join('D:\\PyAnsys\\PyAnsys Workshop\\Examples', 'valve_result_file.rst')

simulation = post.load_simulation(example_path)

#####
# Check what is the solution object
# ~~~~~

print(simulation)

#####
# Get and plot total deformation
# ~~~~~

total_deformation = simulation.displacement(norm = True)
total_deformation.plot()

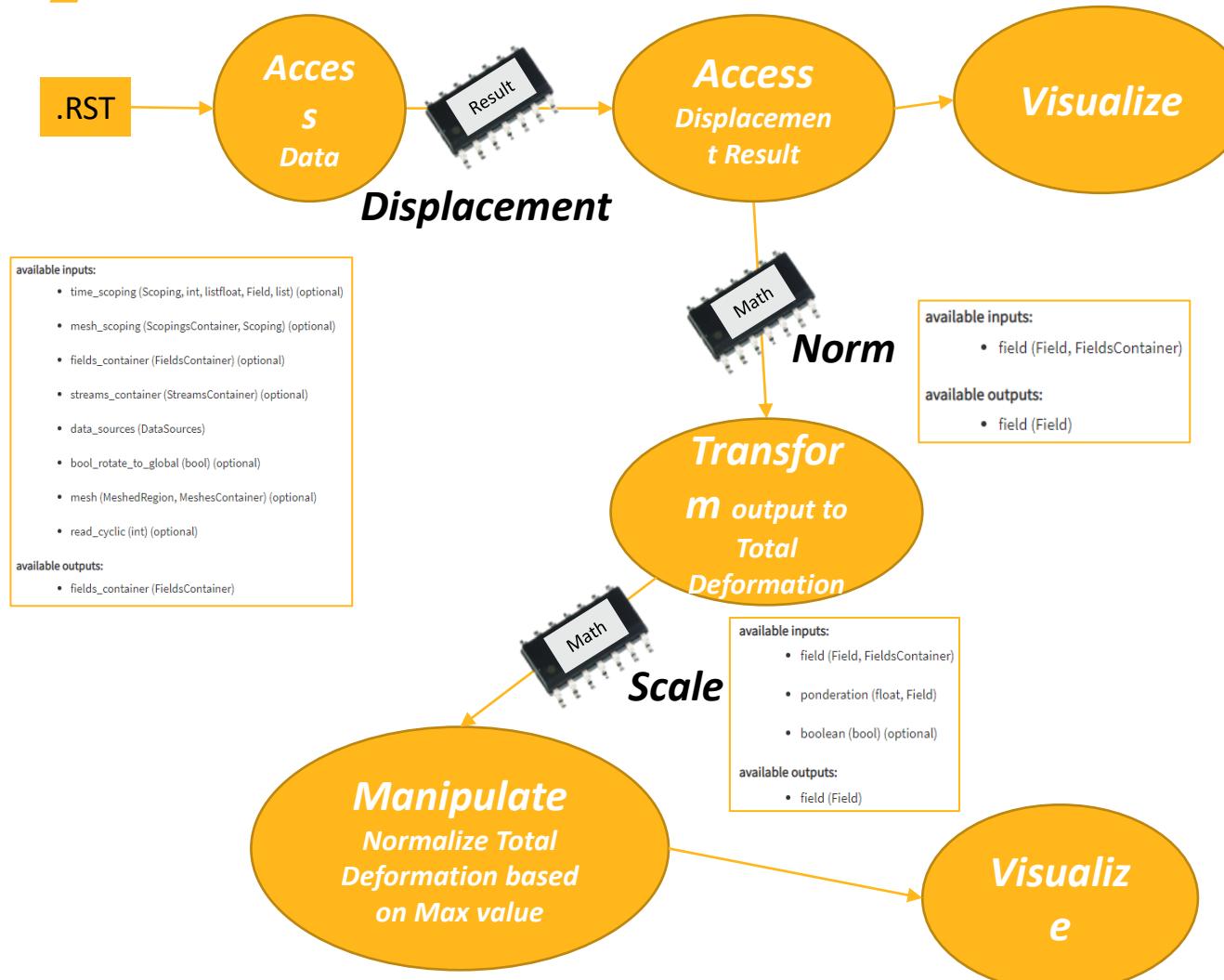
#####
# Extract information on total deformation
# ~~~~~
print(total_deformation)

total_deformation_data = total_deformation.array # extract data
max_total_deformation = total_deformation.array.max() # find maximum value
min_total_deformation = total_deformation.array.min() # find minimum value
print("Max value is: " + str(max_total_deformation) + " and min value is: " + str(min_total_deformation))

#####
#%%
# Get and plot Von Mises stress
# ~~~~~
stress_eqv = simulation.stress_eqv_von_mises_nodal()
stress_eqv.plot()
```



PyDPF-Core



```
from ansys.dpf import core as dpf
import os

example_path = os.path.join(r'D:\PyAnsys\Examples\Use_Existing_Rst', 'use_existing_rst.rst')

model = dpf.Model(example_path)
print(model) # print all info contained in .rst

# print info separately
metadata = model.metadata
print(metadata.result_info)
print(metadata.meshed_region)
print(metadata.time_freq_support)

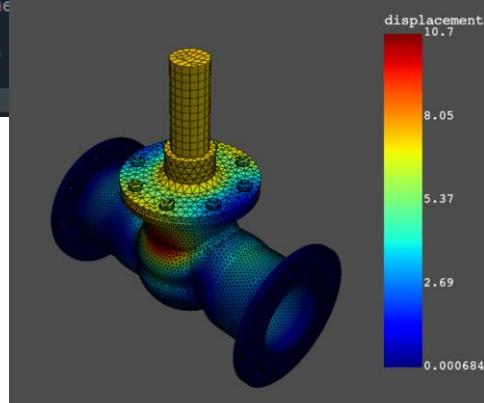
displacement_op = model.results.displacement()
disp_field_container = displacement_op.outputs.fields_container()
print(disp_field_container[0].data)

norm_op = dpf.operators.math.norm()
norm_op.inputs.field.connect(disp_field_container)
total_deformation = norm_op.outputs.field()

scale_op = dpf.Operator('scale') # operator instantiation
scale_op.inputs.field.connect(total_deformation)
scale_value = float(250)
scale_op.inputs.ponderation.connect(scale_value)

my_new_field = scale_op.outputs.field()
print(my_new_field) # Check what information is in the field
print(my_new_field.data) # Check the data in the field

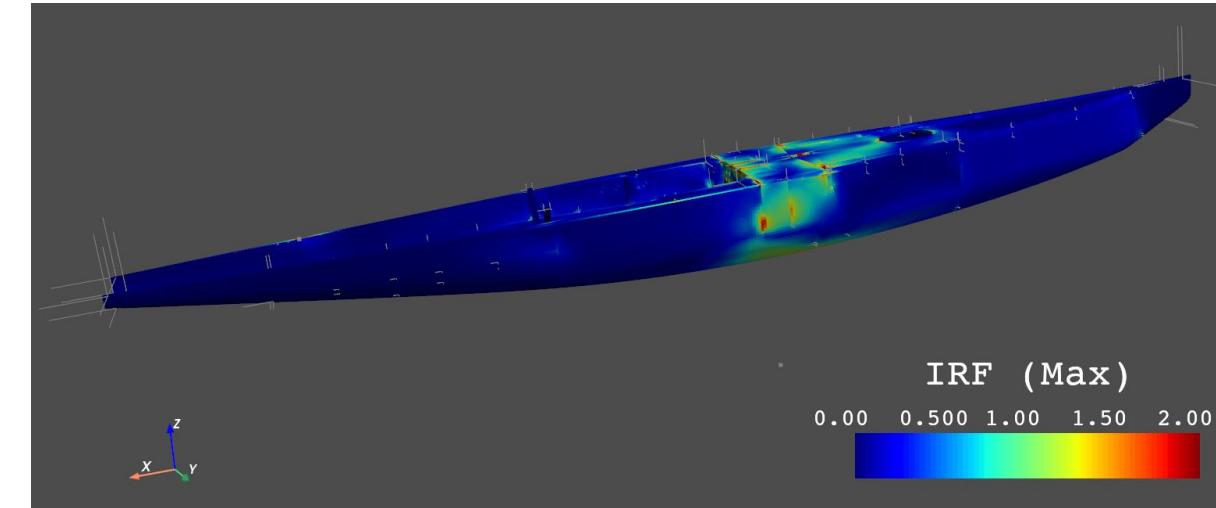
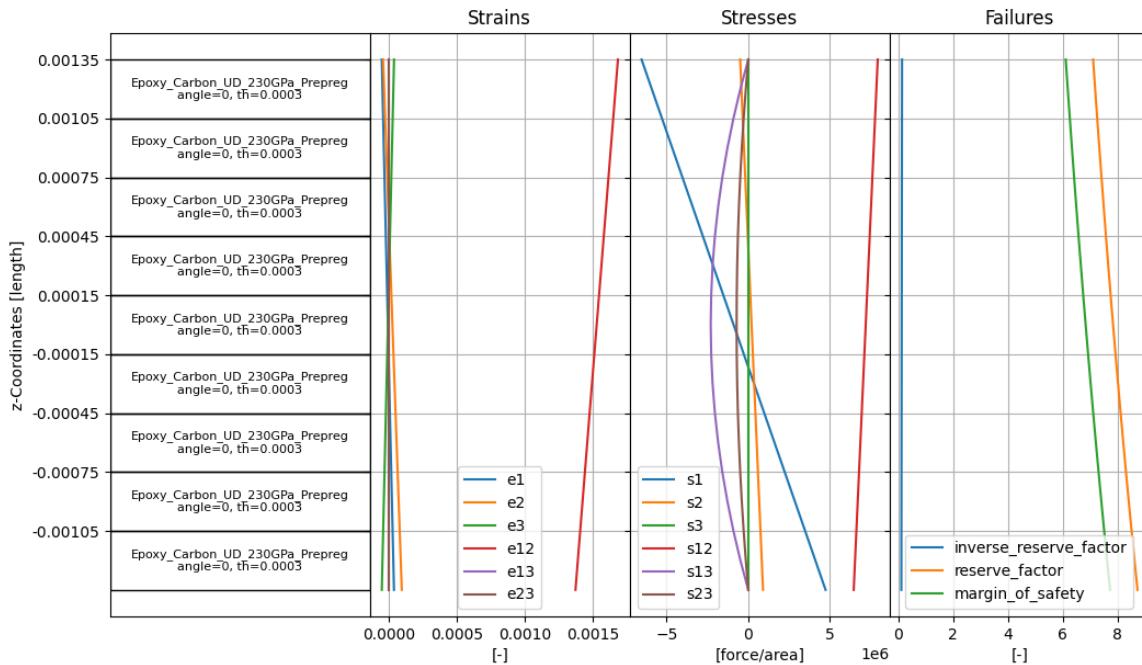
model.metadata.meshed_region.plot(my_new_field)
```



Ansys

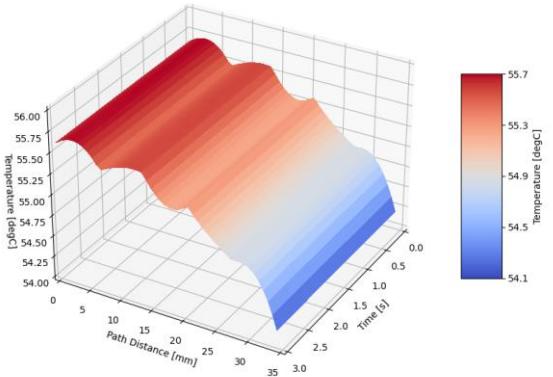
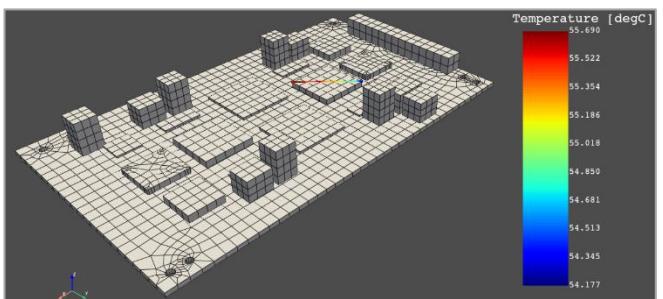
PyDPF-Composites

- Python wrapper for post-processing of composite models
- Additional classes for short fiber and layered composites
- Handles shell and solid elements
- Custom failure criteria
- Sampling points

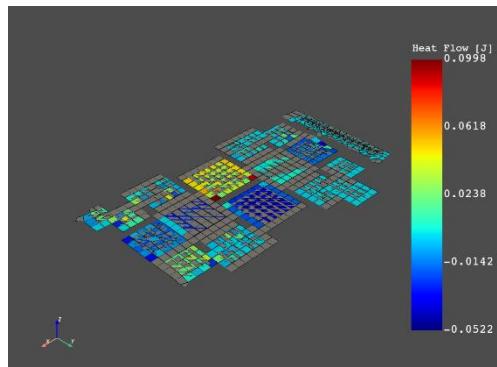
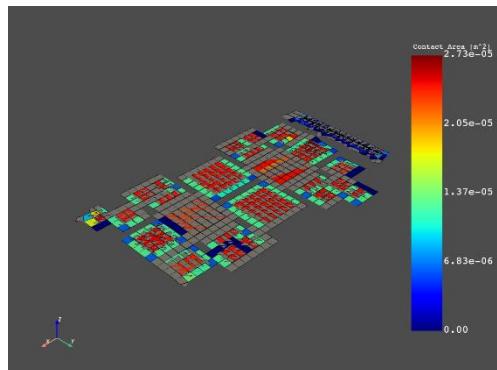


Examples

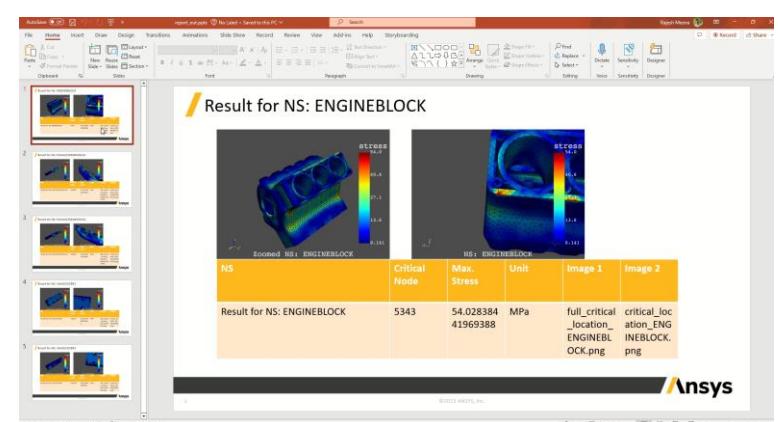
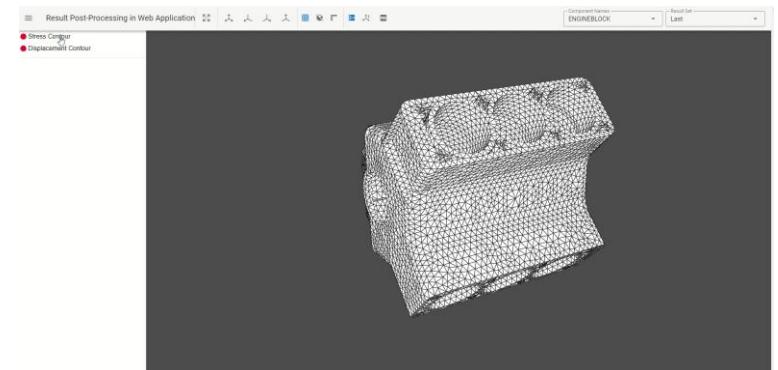
Extracting path and creating 3D Surface plot



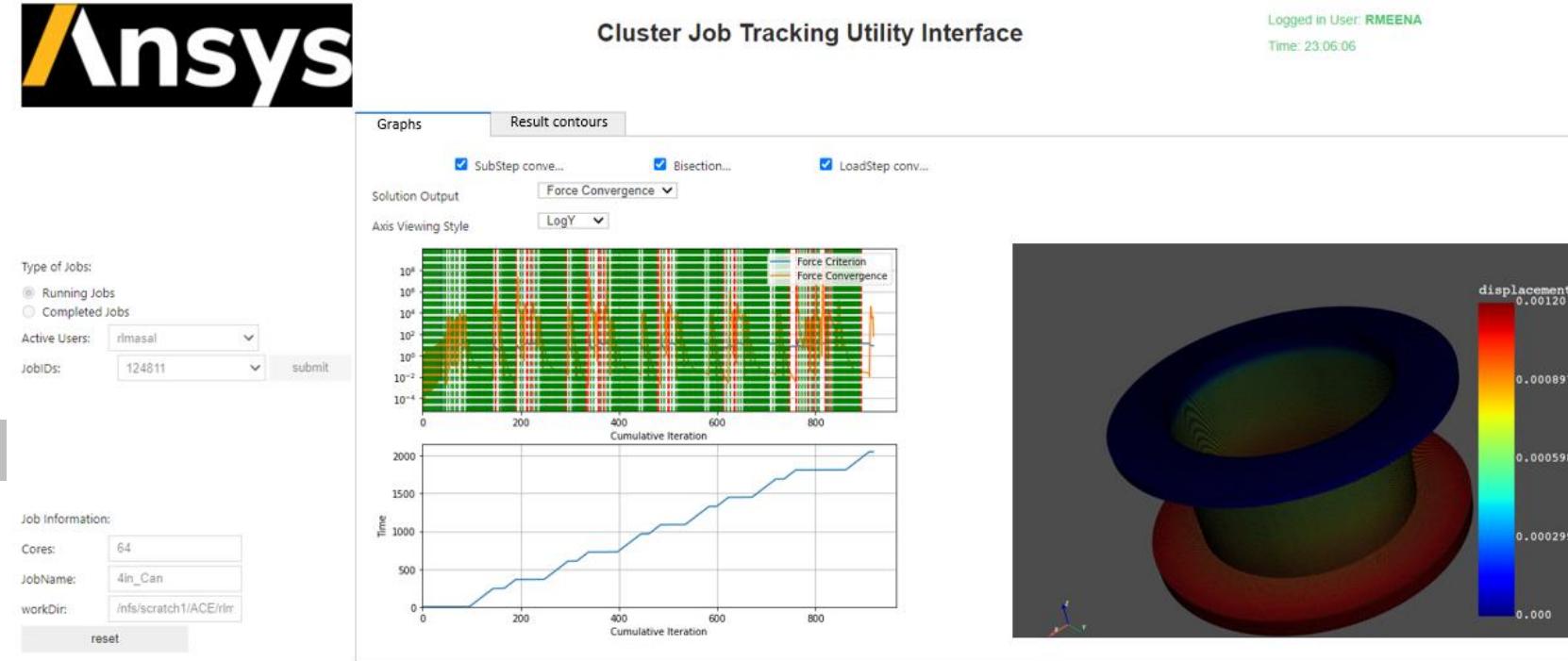
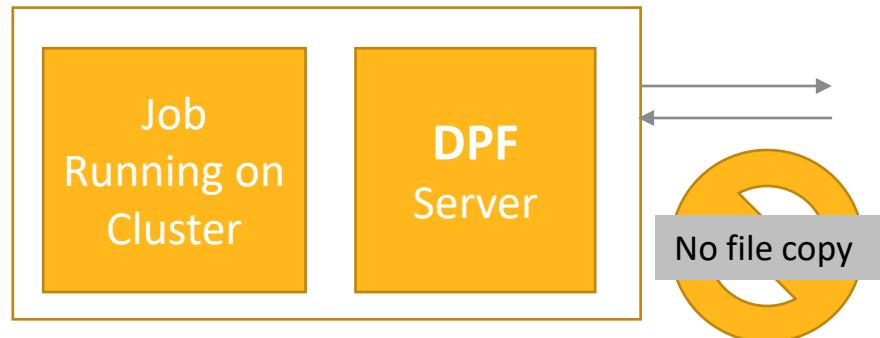
Contact Area and heat flow plotting



Web App and Automatic Report Generation



Cluster Job Tracking WebApp using pyAnsys



Tasks on Server/Cluster

Realtime tracking on client side

- ✓ Uses pyAnsys APIs at backend
- ✓ Realtime tracking
 - ✓ Contacts, convergence graphs
 - ✓ Result contours

Possibilities: More diagnostic features; Can be available on smartphone



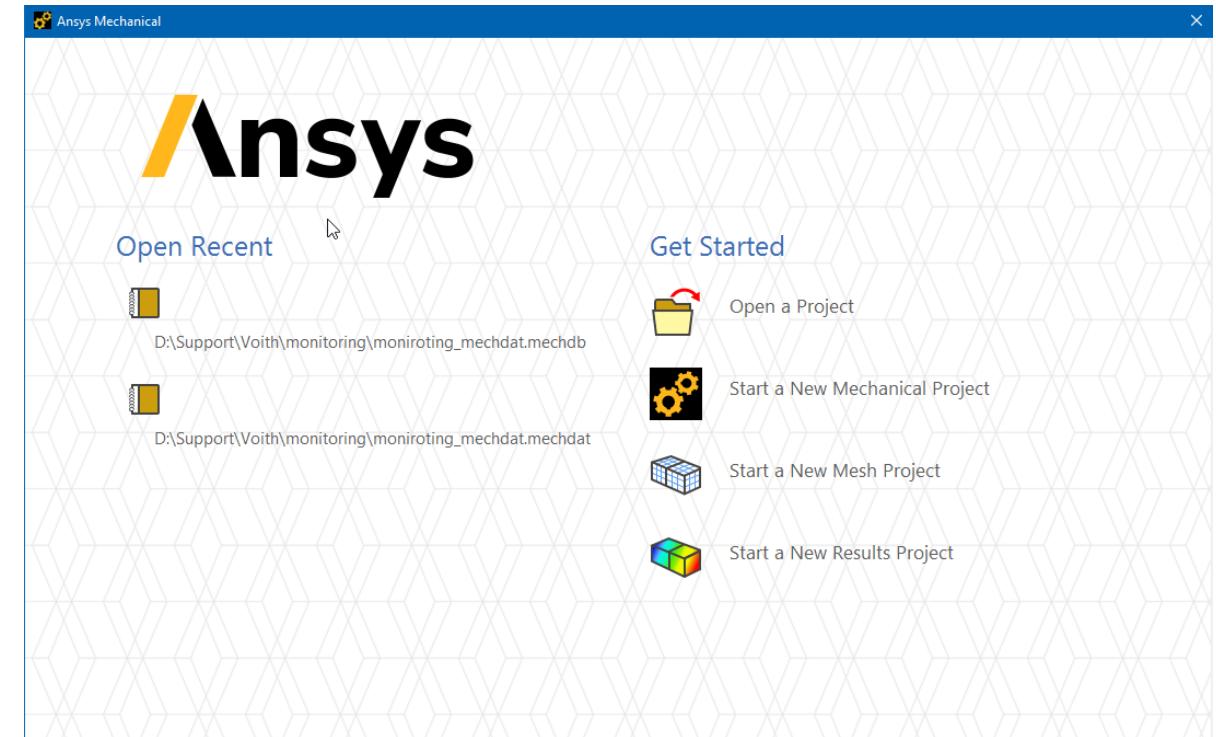
PyMechanical



What is Mechanical Standalone?

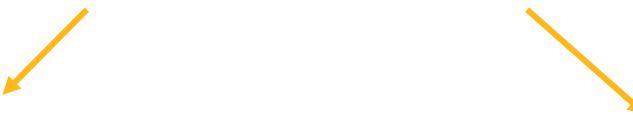
Standalone Mechanical

- Mechanical without dependencies to the project page: Geometry, External Data, Engineering Data, Data transfer and multiphysics links ...
- Allows users to access and automate Mechanical directly without having to script WB



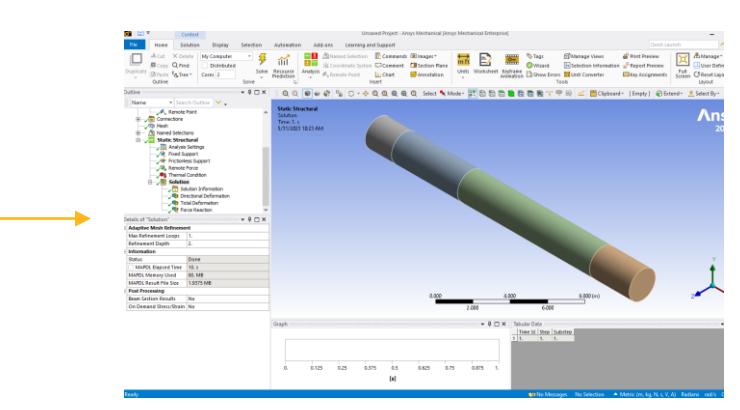
PyMechanical: Access Mechanical through Python

ansys-mechanical-core



Remote session

- Based on gRPC:
 - Server = Mechanical
 - Client = PyMechanical
- Send commands as strings, or as .py scripts

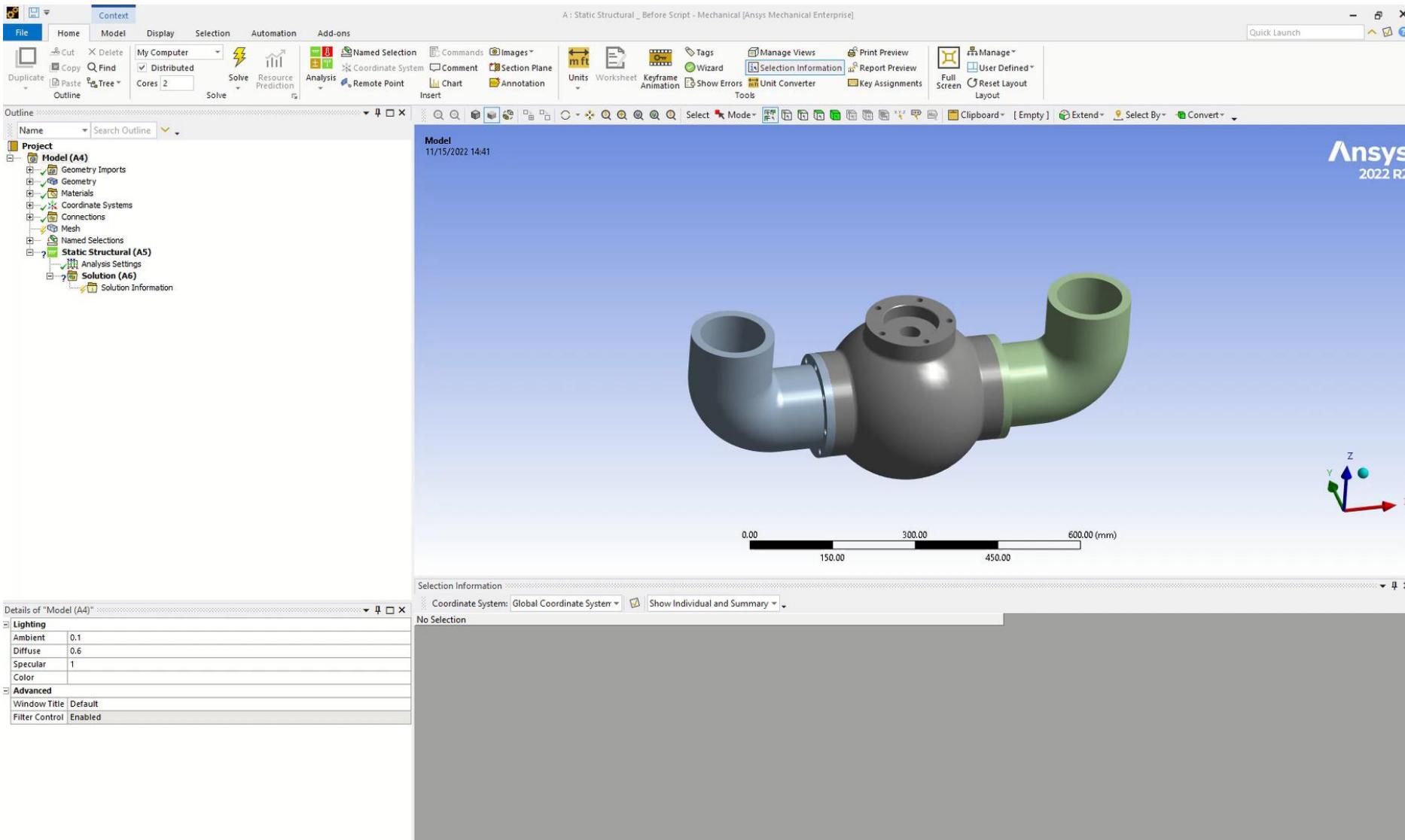


Embedded instance

- Based on [Python.NET](#)
- Mechanical object directly loaded into Python memory using Python.NET
- Send commands as if in the Mechanical console

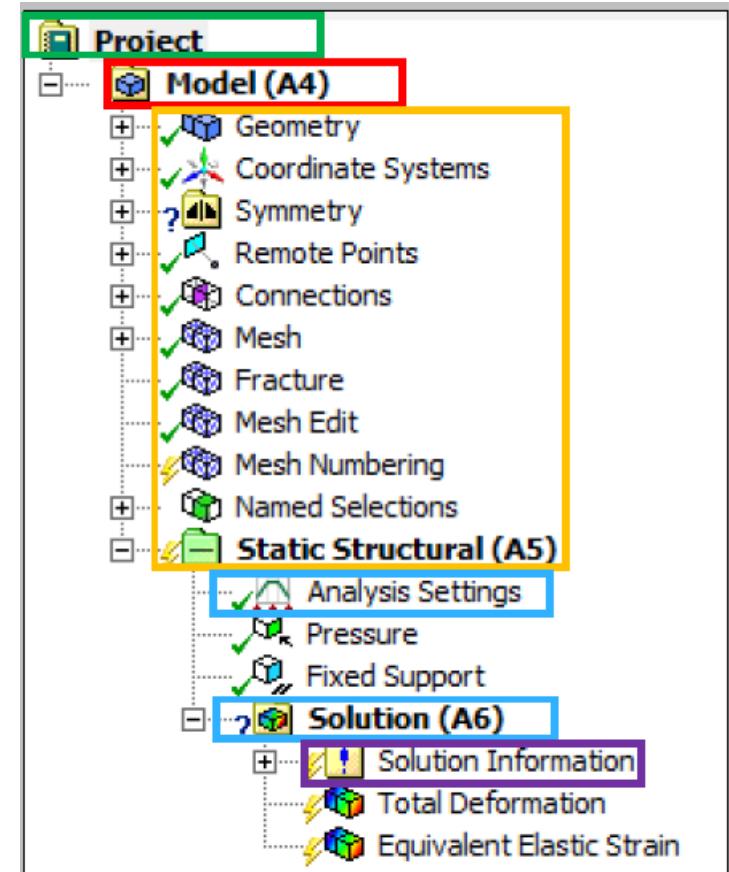
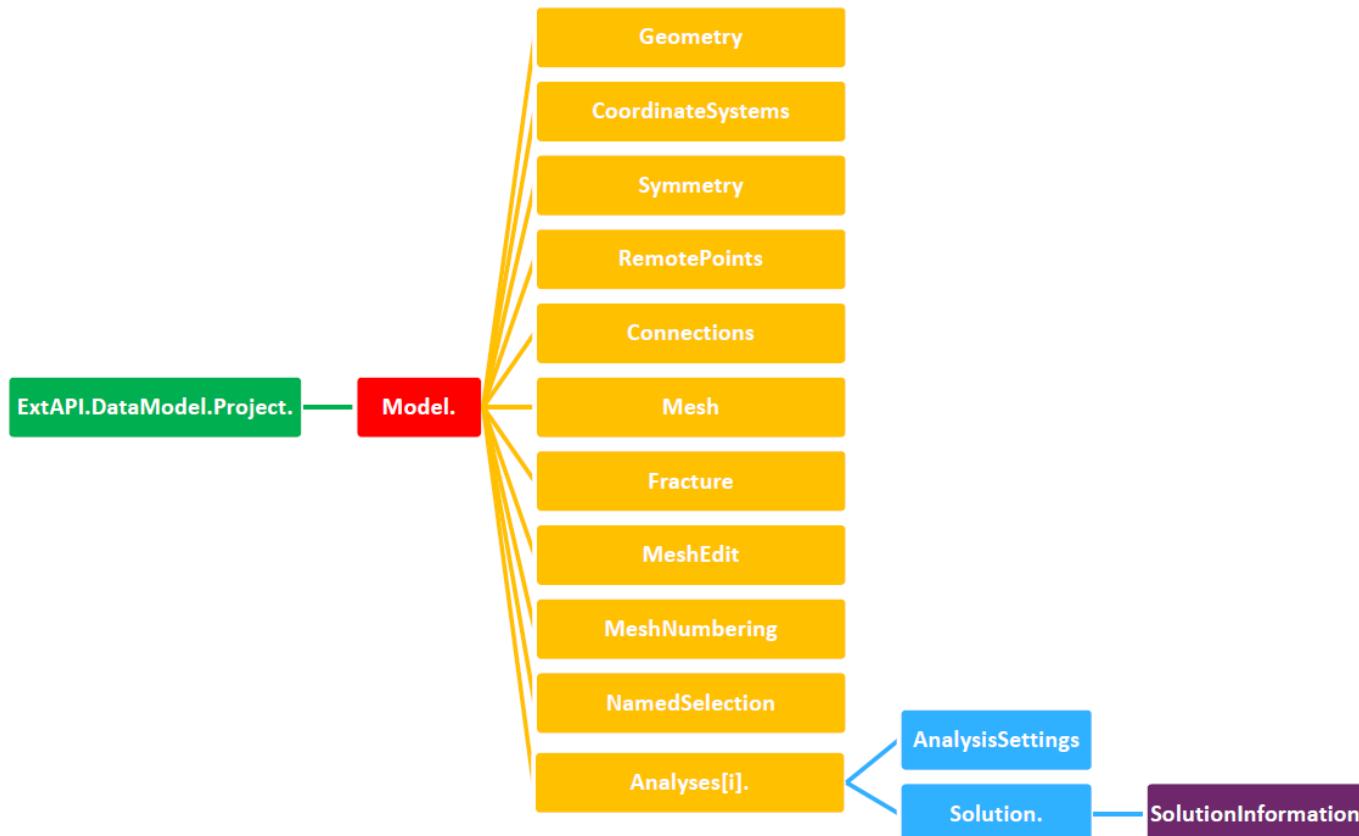
```
20 import json
21
22 from ansys.mechanical.core import embedding as app
23 try:
24     from ansys.mechanical.core import global_variables
25 except:
26     # No embedding - this import breaks test collection
27     global_variables = {}
28
29 embedded_app = app.App(version=232)
30
31 globals().update(global_variables(embedded_app))
32
33 from Ansys.Mechanical.DataModel.Enums import *
34 from Ansys.ACT.Interfaces.Common import *
35 # Wrapper functions to shorten overall script
36 # Function to retrieve geometric body from name
37 def GetGeoBodyByName(searchName):
38     geo = ExtAPI.DataModel.Project.Model.Geometry
39     for part in geo.Children:
40         for body in part.Children:
41             if body.Name == searchName:
42                 return body.GetGeoBody()
```

Mechanical scripting demo: complete model setup



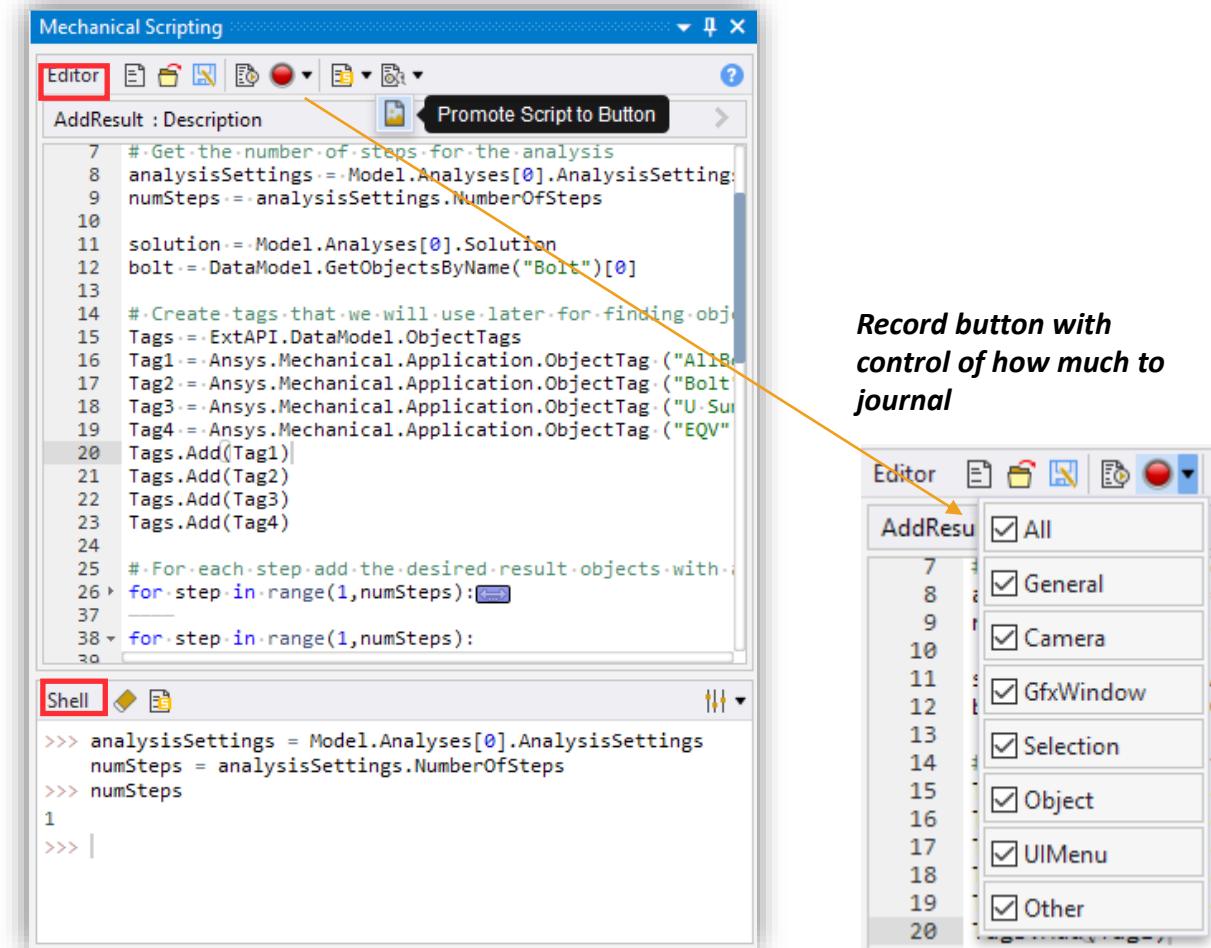
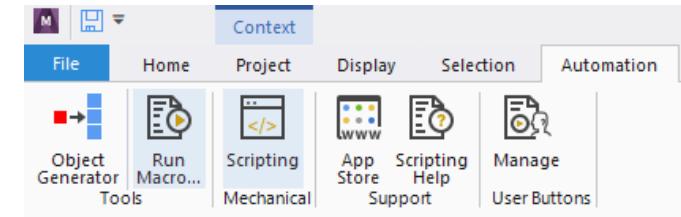
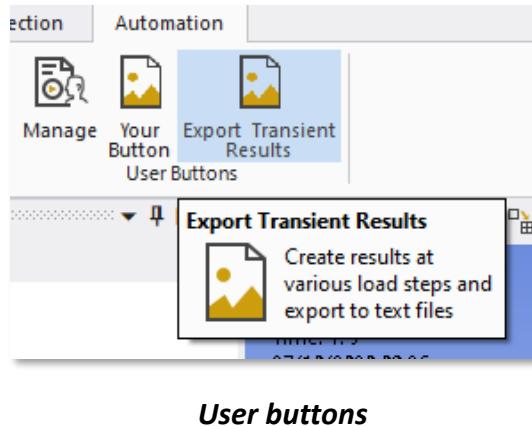
Mechanical Automation API

```
# Reference model  
model = ExtAPI.DataModel.Project.Model
```



Mechanical Automation Scripting

- Well documented Python APIs with scripting guide and quick start examples
- Shell and multi-line editor to build and debug scripts quickly
- Supports autocomplete and Intellisense
- Ability to easily promote scripts to user buttons
- Recording



PyMechanical demo for embedded and remote versions

This examples illustrates how to setup the same model, once with embedded instance and once with remote version.

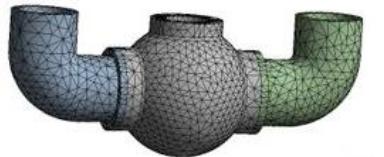
Embedded instance

```
import os
from ansys.mechanical.core import App, global_variables

app = App(version=232)
globals().update(global_variables(app)) # update global variables

# Add static analysis
analysis = Model.AddStaticStructuralAnalysis()
```

Type commands as if in the Mechanical console

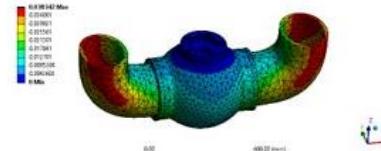


Remote session

```
import os
from ansys.mechanical.core import launch_mechanical

mechanical = launch_mechanical(batch=True, loglevel="DEBUG")
print(mechanical)
# Check working directory
server_project_directory = mechanical.run_python_script("ExtAPI.DataMode")
print(server_project_directory)
# Run mechanical automation script
mechanical_script = os.path.join(work_dir, 'mesh_script.py')
result = mechanical.run_python_script_from_file(mechanical_script)
```

Send Mechanical commands as strings or as second Python script



- Import CAD geometry
- Define material assignment
- Define mesh settings
- Define boundary conditions
- Solve
- Post-process
- Export results to text files

Example

The image shows a dual-screen setup. On the left, a Jupyter Notebook interface is displayed in a browser window titled "Remote_Mechanical - Jupyter Notebook". The notebook contains four code cells:

- ACT app**:
In [2]:

```
import os
from ansys.mechanical.core import launch_mechanical
if "AWP_ROOT231" in os.environ.keys():
    exe_path = os.path.join(os.environ["AWP_ROOT231"], "aisol/bin/winx64/AnsysWBUE")
else:
    raise Exception("For PyAnsys workshop Ansys 2023R1 is needed. Please install")
```
- Launching Mechanical Session**:
In [3]:

```
mechanical = launch_mechanical(exec_file=exe_path,batch=False, cleanup_on_exit=False)
```
- Opening An Existing Project**:
In []:

```
project_open_command= """
DataModel.Project.Open(r"D:/Workshop/PyAnsys/Workshop Getting Started with PyAns
"""

mechanical.run_python_script(project_open_command)
```
- Adding ACT Object and scoping properties**:
In []:

```
act_control_command = """
ext = ExtAPI.ExtensionManager.GetExtensionByName("clampLoad")
analysis = Model.Analyses[0]
load_obj = analysis.CreateLoadObject("clampLoad", ext)
ns = DataModel.GetObjectsByName("clamp")[0]
load_obj.Properties["Geometry"].Value = ns
load_obj.NotifyChange()
str(load_obj.Properties["Step"].Value)
"""

step_number = mechanical.run_python_script(act_control_command)
```

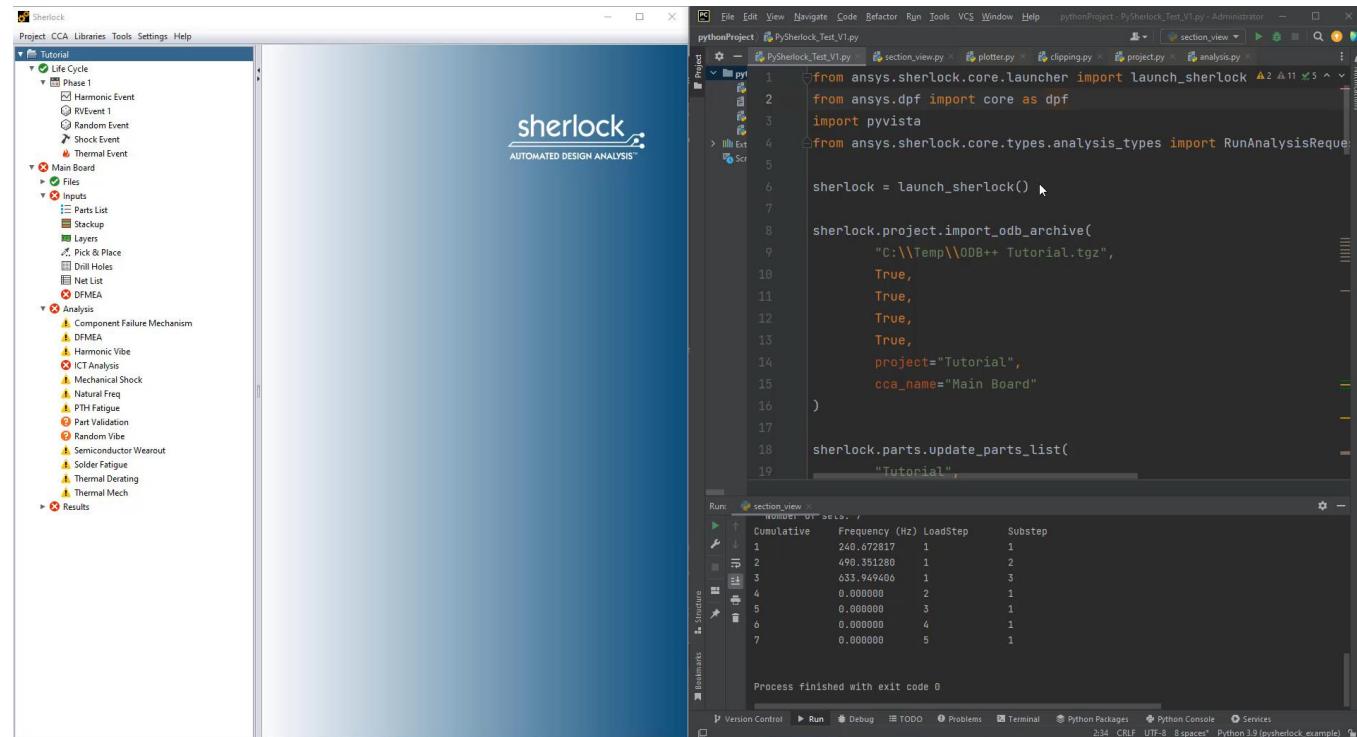
On the right, the Ansys Mechanical interface (version 2023 R1) is open. The "Outline" panel shows a project structure with "Project*", "Model", "Geometry Imports", "Geometry", "Materials", "Coordinate Systems", and "Mesh". The "Geometry" panel displays a 3D coordinate system with X, Y, and Z axes. A "Details of 'Geometry'" dialog is open, showing settings for "Definition" (Source, Type: Millimeters, Element Control: Program Controlled, Display Style: Body Color), "Bounding Box", and "Properties" (Statistics, Update Options, Basic Geometry Options, Advanced Geometry Options). The "Graphics Annotations" panel is also visible.

PySherlock

Ansys

PySherlock

- Communicate With Sherlock via CPython
 - Update parts
 - Add vibration events profiles
 - Run analysis
 - Generate report
 - Plot results using pyvista and PyDPF



The screenshot shows the PyCharm IDE interface. On the left is the Sherlock CCA graphical user interface, displaying a project structure for 'Tutorial' with sections like Life Cycle, Main Board, Inputs, Analysis, and Results. On the right is the Python code editor with a script named 'PySherlock_Test_V1.py'. The code uses the Sherlock API to launch the application, import core components, update part lists, and run specific analyses. A terminal window at the bottom shows the command 'section_view' and its output, which includes a table of frequency data.

Number of sets	Cumulative	Frequency (Hz)	LoadStep	Substep
1	240.672817	1	1	
2	490.351280	1	2	
3	633.949408	1	3	
4	0.000000	2	1	
5	0.000000	3	1	
6	0.000000	4	1	
7	0.000000	5	1	

PyPrimeMesh

Ansys

PyPrimeMesh

- PyPrimeMesh lets you use Ansys core meshing technology, embedded across Ansys flagship tools, directly from within your python environment.
- In 2023R1 with PyPrimeMesh, it's possible to:
 - Generate surface and volume mesh of various types using parallel capabilities
 - Apply a range of sizing controls to control mesh distribution
 - Create complex workflows and automation
 - Use wrapping methods to extract and mesh regions
 - Modify connectivity of topology and mesh
 - Import and facet CAD
 - Export to various Ansys solvers

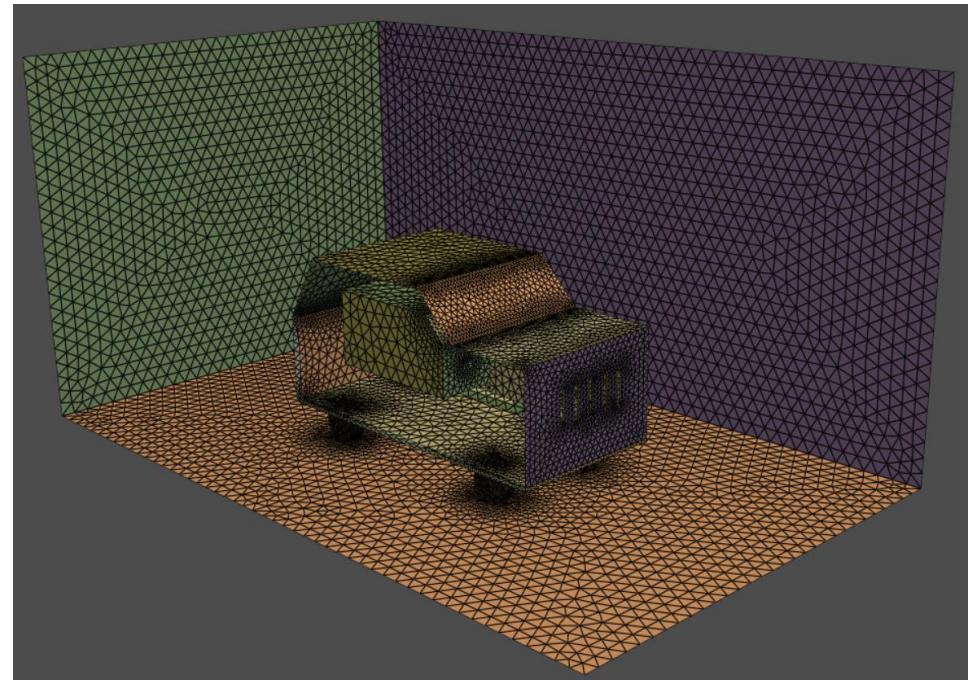
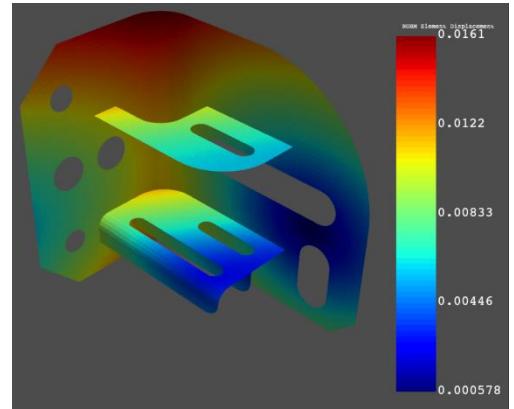
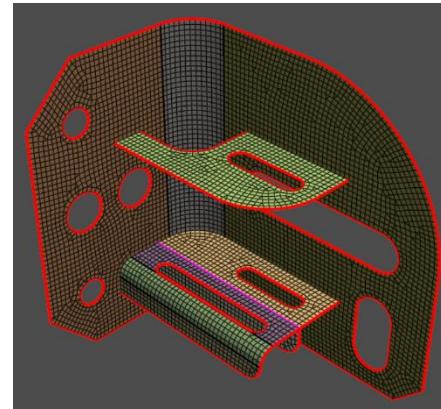
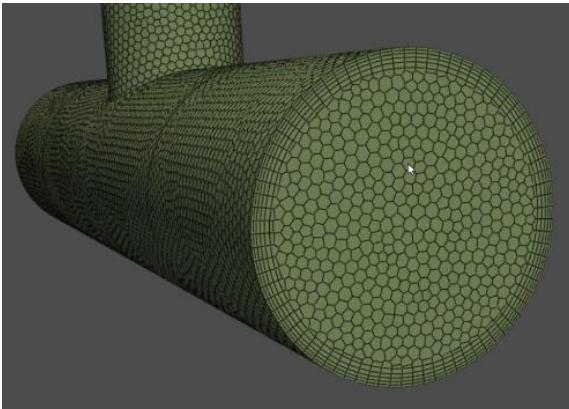
```
14 # read a file
15 file_io = prime.FileIO(model)
16 file_io.  
    ⌂ read_pmdat
    ⌂ append_mesh
    ⌂ export_boundary_fitted_spline_kfile
    ⌂ export_fluent_case
    ⌂ export_fluent_meshing_mesh
    ⌂ export_lsdyna_keyword_file
    ⌂ export_mapdl_cdb
    ⌂ import_fluent_case_beta
    ⌂ import_fluent_meshing_mesh_beta
    ⌂ import_fluent_meshing_meshes_beta
    ⌂ import_lsdyna_keyword_file
    ⌂ import_mapdl_cdb
```

```
3
4
5
6     import logging
7     import ansys.meshing.prime
8
9     prime.launch_prime()
10
11     client = prime.Client()
12     model = client.model
13
14     # read a file
15     file_io = prime.FileIO()
16     file_io.read_pmdat()  
    ⌂ (file_name) -> Any
    Function that reads PRIME's database file.
    Read PRIME's database file from disk. PRIME's database files have
    pmdat extension. Unicode paths are not currently supported by this
    API.
  
Parameters
    file_name : str
        Path to file on disk.
  
Returns
```

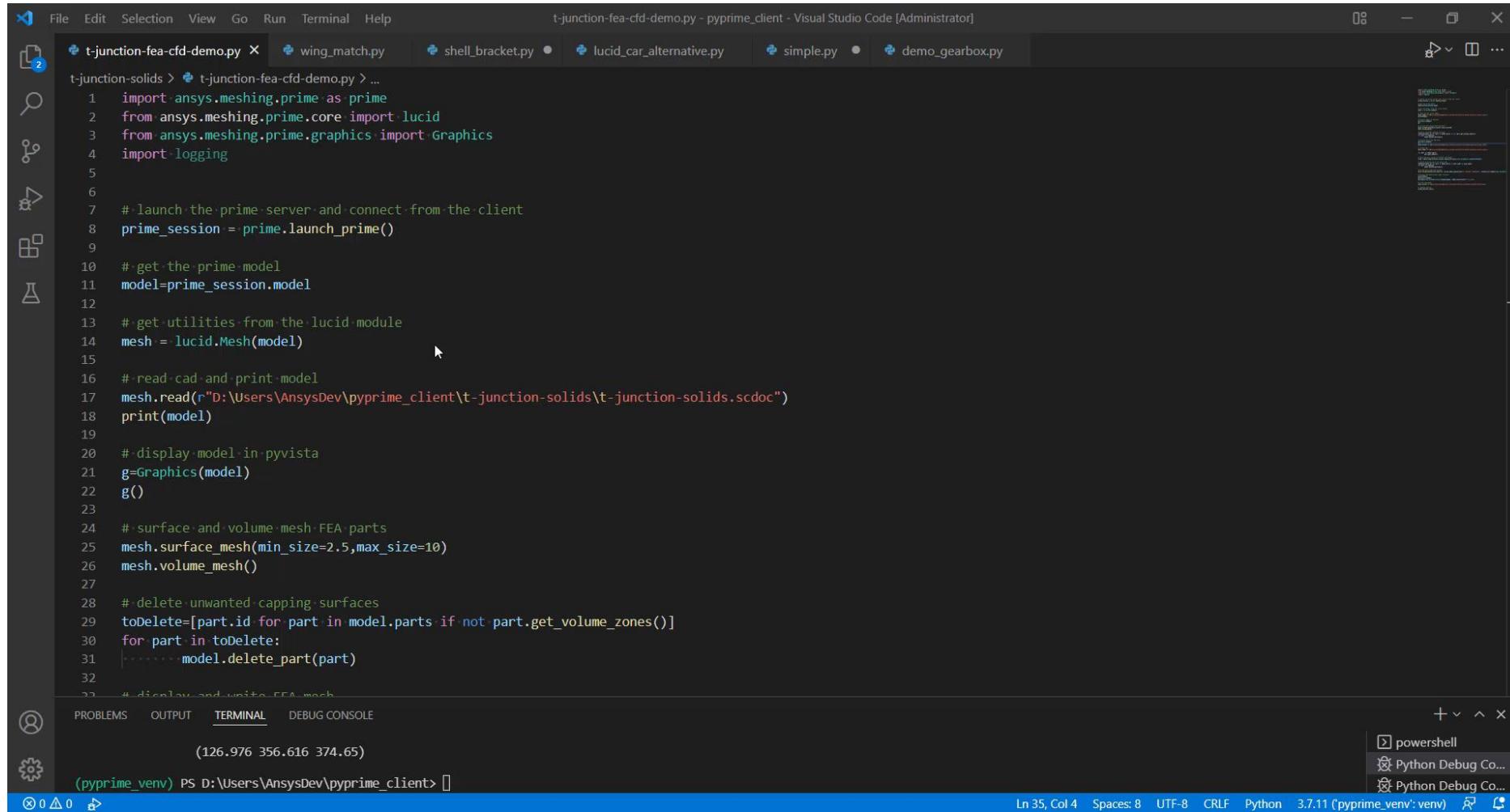
PyPrimeMesh Examples

Some examples:

- Wrapping
- Poly mesh with inflation
- Quad shells
- Connecting topology
- ...



Mechanical / CFD - T-section demo clean CAD & wrapping mesh processes



The screenshot shows a Visual Studio Code interface with several tabs open at the top, including "t-junctionfea-cfd-demo.py", "wing_match.py", "shell_bracket.py", "lucid_car_alternative.py", "simple.py", and "demo_gearbox.py". The main editor area contains the following Python script:

```
File Edit Selection View Go Run Terminal Help t-junctionfea-cfd-demo.py - pyprime_client - Visual Studio Code [Administrator] t-junction-solids > t-junctionfea-cfd-demo.py > ... 1 import ansys.meshing.prime as prime 2 from ansys.meshing.prime.core import lucid 3 from ansys.meshing.prime.graphics import Graphics 4 import logging 5 6 # launch the prime server and connect from the client 7 prime_session = prime.launch_prime() 8 9 # get the prime model 10 model=prime_session.model 11 12 # get utilities from the lucid module 13 mesh = lucid.Mesh(model) 14 15 # read cad and print model 16 mesh.read("D:\\Users\\AnsysDev\\pyprime_client\\t-junction-solids\\t-junction-solids.scdoc") 17 print(model) 18 19 # display model in pyvista 20 g=Graphics(model) 21 g() 22 23 # surface and volume mesh FEA parts 24 mesh.surface_mesh(min_size=2.5,max_size=10) 25 mesh.volume_mesh() 26 27 # delete unwanted capping surfaces 28 toDelete=[part.id for part in model.parts if not part.get_volume_zones()] 29 for part in toDelete: 30     model.delete_part(part) 31 32 # display and unite FEA mesh 33 34 # PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE (126.976 356.616 374.65) (pyprime_venv) PS D:\\Users\\AnsysDev\\pyprime_client> [] @ 0 △ 0 🔍 0 8 PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE + powershell Python Debug Co... Python Debug Co... Ln 35, Col 4 Spaces: 8 UTF-8 CRLF Python 3.7.11 ('pyprime_venv': venv) 🔍 🔍
```

Input:
scdoc geometry

Output:
Mechanical cdb
+
Fluent msh

Reading -> meshing
-> writing using
high level api's



PyDyna



Ansys

PyDYNA is Public !!!



PyAnsys » PyDYNA documentation 0.4.2

PyDYNA documentation 0.4.2

PyDYNA



Overview

PyDYNA is a Pythonic package for providing a more convenient and complete way to build an Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver, and finally postprocess the results.

PyDYNA contains two submodules, `ansys.dyna.core.pre` and `ansys.dyna.core.solver`

- `pre`: This module provides highly abstracted APIs for creating and setting up DYNA input decks. There are many classes supported, namely, DynaMech, DynalGA, DynalCFD, DynaSALE, DynaEM, DynaNVH, DynaMaterial, DynalSPH, DynalCFD and DynaAirbag. Each of these classes can be used to generate LS-DYNA keywords. Since these classes have high-level abstraction, each function call generates groups of keywords needed to define an input in LS-DYNA.
- `solver`: This API provides features to interact directly with the Ansys LS-DYNA solver. LS-DYNA is primarily a batch solver with very limited interactive capabilities, the `solver` service provides a way to push input files to the LS-DYNA solver, monitor the state of the running job, change the value of a load curve and finally retrieve result files back from the server

Once you have results, you can use the Ansys Data Processing Framework (DPF), which is designed to provide numerical simulation users and engineers with a toolbox for accessing and transforming simulation data. DPF can access data from Ansys solver files and from several files with neutral formats, including CSV, HDF5, and VTK. Using DPF's various operators, you can manipulate and transform this data.

≡ On this page
PyDYNA documentation 0.4.2
PyDYNA
Overview
Documentation and issues
License

Edit on GitHub
Show Source

<https://dyna.docs.pyansys.com/version/stable/>

<https://github.com/ansys/pydyna>

DYNA input deck generation

Connect to dyna.pre server

```
hostname = 'localhost'  
drop_test = DynaSolution(hostname)  
fns = []  
path = sys.path[0]+os.sep  
fns.append(path + "main.k")  
fns.append(path + "nodes_elements.k")  
drop_test.open_files(fns)
```



CONTROL and DATABASE Cards

```
drop = DynaMech()  
drop_test.add(drop)  
drop_test.set_termination(0.01)  
drop.set_timestep(timestep_size_for_mass_scaled=-4e-9)  
drop_test.set_output_database(glstat=1.e-7,matsum=1.e-7,rwforc=1.e-7)
```



MATERIAL Cards

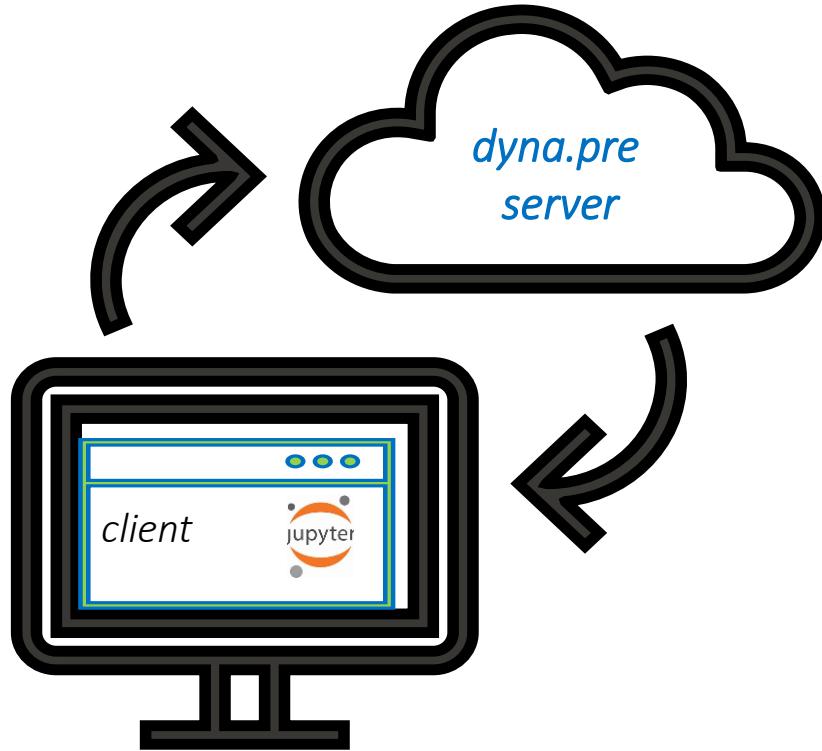
```
mat_1 = MatElastic(mass_density=1.3e-9,young_modulus=4100.,poisson_ratio=0.29  
mat_2 = MatElastic(mass_density=1.57e-9,young_modulus=25000.,poisson_ratio=0.:  
mat_3 = MatElastic(mass_density=1.0e-9, young_modulus=3000., poisson_ratio=0.:  
mat_4 = MatPiecewiseLinearPlasticity(mass_density=8.30e-09,young_modulus=1100.  
                                         poisson_ratio=0.34,yield_stress=280.  
                                         tangent_modulus=1150.)
```



Assign Material and Section to Parts

```
Bottom_Top = SolidPart(1)  
Bottom_Top.set_material(mat_1)  
Bottom_Top.set_element_formulation(SolidFormulation.CONSTANT_STRESS_SOLID_ELEI  
drop.parts.add(Bottom_Top)
```

DYNA input deck generation



RIGIDWALL, INITIAL VELOCITY and GRAVITY

```
rigidwall = RigidwallPlanar(Point(-1.8581,-32.039,2.904), Point(-2.0,-32.039,:  
    coulomb_friction_coefficient=0.1)  
  
drop.add(rigidwall)  
  
saset = range(1, 25)  
velocity = drop.initialconditions.create_velocity(PartSet(saset), velocity=Velocity(0.0, 0.0, 0.0))  
  
g = Gravity(dir=GravityOption.DIR_X, load=Curve(x=[0.005, 1e18], y=[1, 1]))  
drop.add(g)
```

TIED CONTACT between Part Sets

```
contact = Contact(category=ContactCategory.SHELL_EDGE_TO_SURFACE_CONTACT  
                   , type=ContactType.TIED)  
surf1 = ContactSurface(PartSet([5]))  
surf2 = ContactSurface(PartSet([9,4]))  
contact.set_slave_surface(surf1)  
contact.set_master_surface(surf2)  
drop.add(contact)
```

Save the file

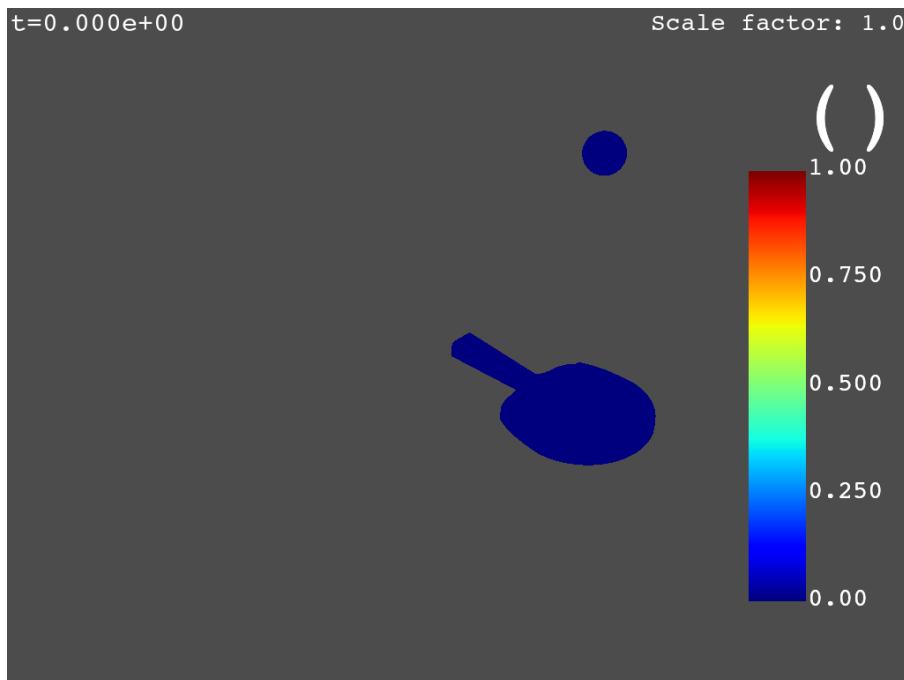
```
drop_test.save_file()
```

Easily deployable containers or standalone servers

Interactive Solver

```
# my docker files are under
#D:\PYDYNA_BETA_V.0.1\pyDyna\src\ansys\dyna\core\solver
import ansys.dyna.core.solver as solver

dyna = solver.DynaSolver('localhost','5000')
dyna.push(r'D:\pyDyna_Examples\DROP_BINOUT\tt.k')
dyna.start(4)
dyna.run('i=racket_inp.k ncpu=4 jobid=tt')
```



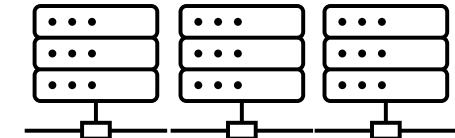
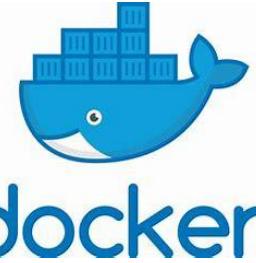
```
while t_step < Tend:
    dyna.pause()
    (c,t) = dyna.time()
    c1000_1,v1000_1 = dyna.node(100000)
    c2000_1,v2000_1 = dyna.node(200000)
    c3000_1,v3000_1 = dyna.node(300000)
    cx1000_1 = c1000_1[0]
    cz1000_1 = c1000_1[2]
    cx2000_1 = c2000_1[0]
    cz2000_1 = c2000_1[2]
    cx61_1 = c3000_1[0]
    cz61_1 = c3000_1[2]
    vx1000_1 = v1000_1[0]
    vz1000_1 = v1000_1[2]
    xpos10_1=vx1000_1

    ang_ball=(math.atan(vx1000_1/vz1000_1))
    vx_plate=cx2000_1-cx61_1
    vz_plate=cz2000_1-cz61_1

    ang_plate=(math.atan(vx_plate/vz_plate))
    diff_ang=ang_ball-ang_plate
    distz=cz1000_1-cz61_1
    t_imp=abs(distz/vz1000_1)
    rot_vel=diff_ang/t_imp

    if cz1000_1 < 60:
        dyna.setlc(30,rot_vel)
    else:
        dyna.setlc(30,no_vel)

    dyna.setlc(20,xpos10_1)
    t_step = t_step + step_size
    t_sol += [t_step]
    n_step+=1
    try:
        dyna.resume(time=t_step)
    except:
        print('LSDYNA run terminated')
```



MPP DYN

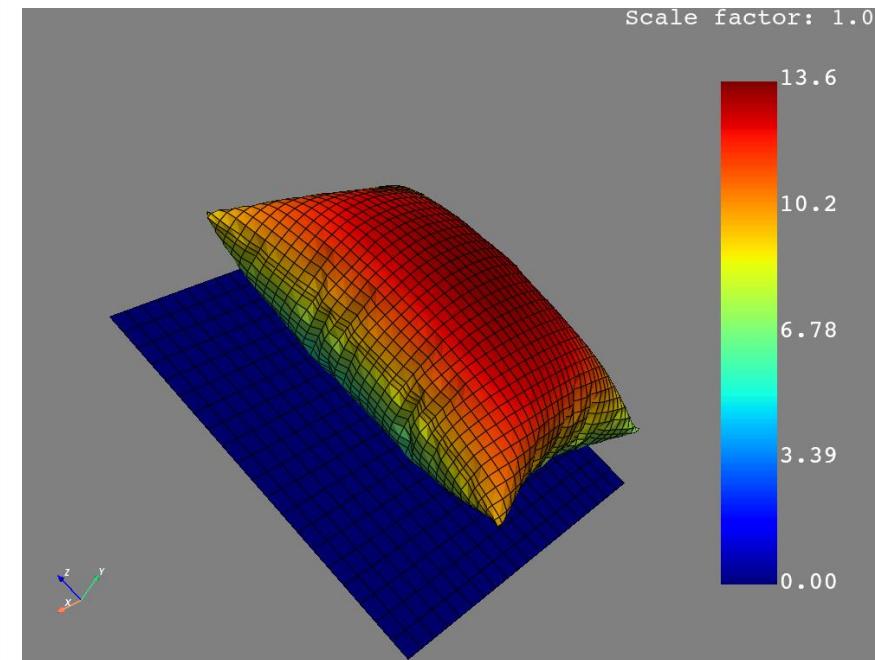
Read the data from the server and post process

Plot the displacement of select parts

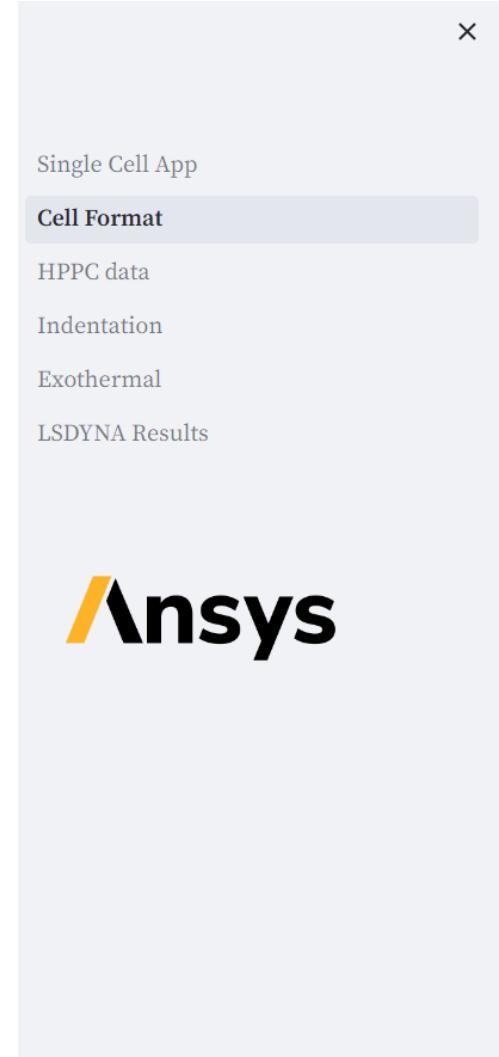
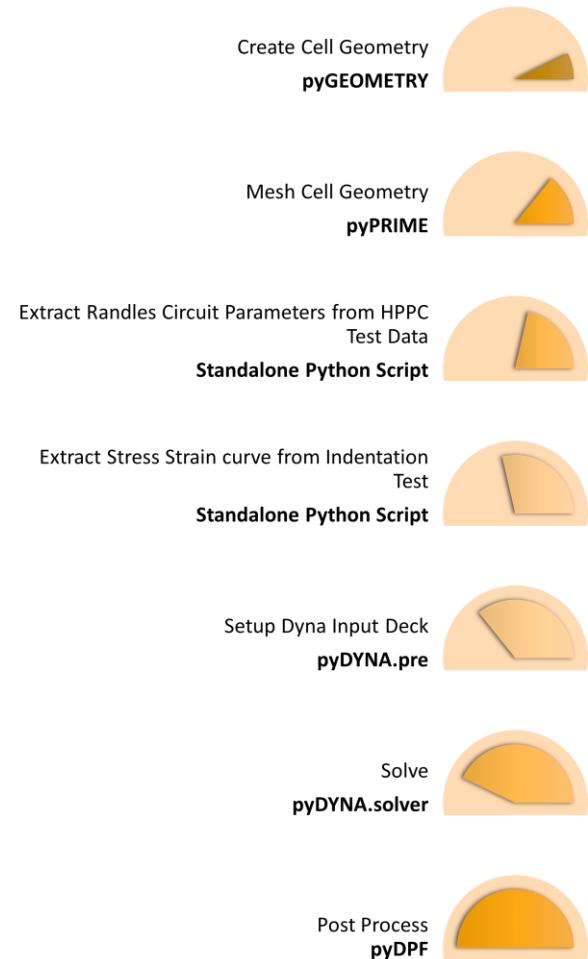
```
model = dpf.Model(data_sources=ds,server=server)
print(model)

part_id = [6,9]

stre = dpf.operators.result.stress(time_scoping=[11])
stre.inputs.data_sources.connect(ds)
stre.inputs.requested_location.connect("Nodal")
stre.inputs.time_scoping.connect(range(12))
stre.connect(25, part_id)
fieldsStr = stre.outputs.fields_container()
shell_layer_extract = dpf.operators.utility.change_shell_layers()
shell_layer_extract.inputs.fields_container.connect(fieldsStr)
shell_layer_extract.inputs.e_shell_layer.connect(0)
fields_top = shell_layer_extract.outputs.fields_container_as_fields_container()
field = fields_top.get_field({'time':11})
field.plot(notebook=False)
```



End to End - Battery Characterization App



Ansys

Cell Format Selector

Choose the format of cell being characterized

Select the Cell Format

Cylindrical Pouch Prismatic

width

length

thickness

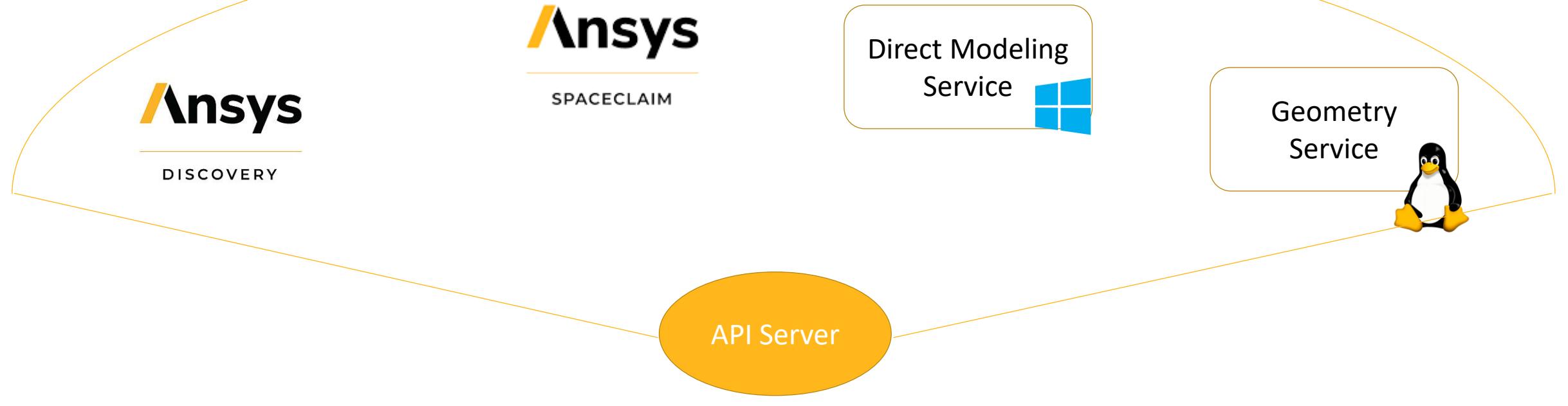
height

Submit

PyAnsysGeometry

Ansys

PyAnsysGeometry



Py // Ansys

Client-side visualization

- Tessellation allows to visualize geometries on client-side

```
[5]: # Create a new design
design = modeler.create_design("AdvancedFeatures_SpurGear")

# Extrude your sketch
dummy_gear = design.extrude_sketch("SpurGear", sketch, Distance(200, UNITS.mm))

# Let's see how it looks like
design.plot()
```

