Introduction to Python for simulation

April, 2025

Presented by Florian Jacta – Product Specialist

The Goals

- Start with 3D Data Visualization in Python
- Solve missing information in object
- Compute drag for object

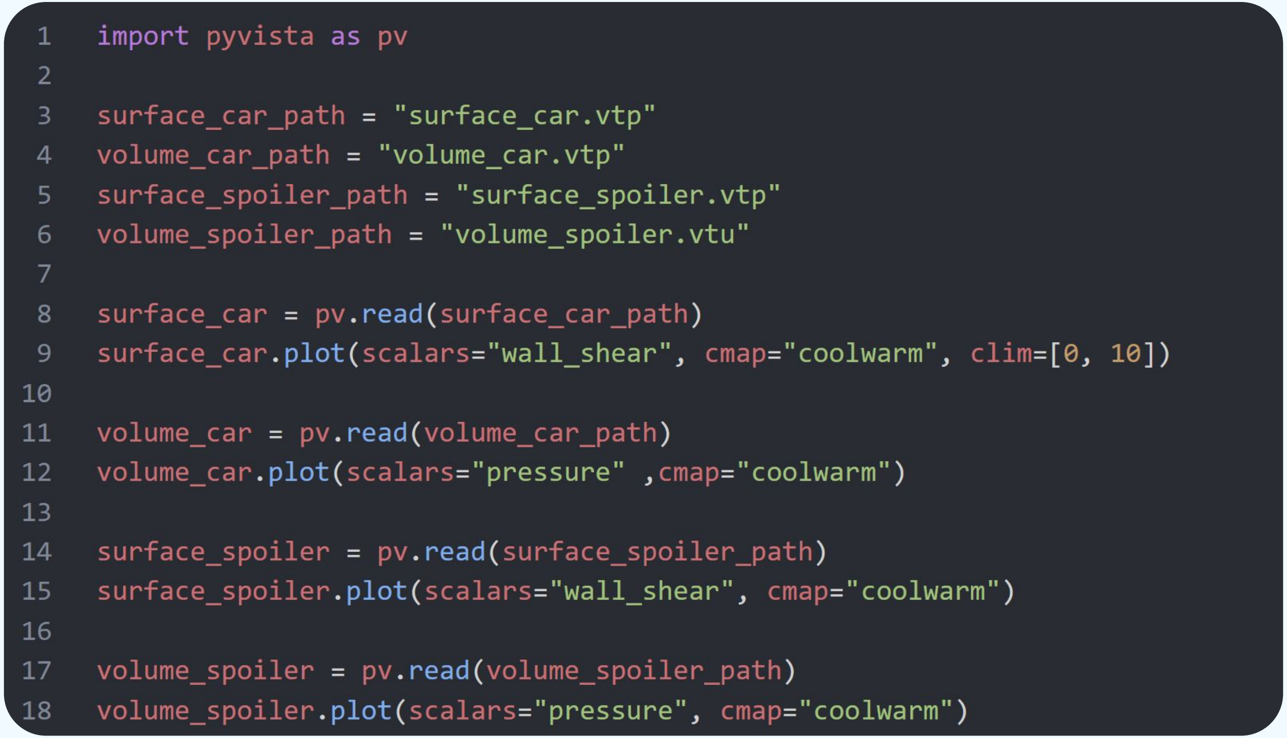
PyVista - What it is

- **Python** package built on VTK for 3D scientific visualization
- Natively supports file formats like **.vtp, .vtu, .vtk**
- Enables interactive **3D plots** for volume and surface meshes
- Integrates with **NumPy** for numerical analysis

PyVista Functions

- **Read**: Import data from formats like .vtp, .vtu, .stl, .ply
- **Transform/Analyze**: Translate, rotate, and scale meshes, compute distances, areas…
- **Filter**: Smooth meshes, extract surfaces, compute normals
- **Plot:** Create 2D/3D visualizations of structured and unstructured data

Data Visualization



```python
import pyvista as pv

surface_car_path = "surface_car.vtp"
volume_car_path = "volume_car.vtp"
surface_spoiler_path = "surface_spoiler.vtp"
volume_spoiler_path = "volume_spoiler.vtu"

surface_car = pv.read(surface_car_path)
surface_car.plot(scalars="wall_shear", cmap="coolwarm", clim=[0, 10])

volume_car = pv.read(volume_car_path)
volume_car.plot(scalars="pressure" ,cmap="coolwarm")

surface_spoiler = pv.read(surface_spoiler_path)
surface_spoiler.plot(scalars="wall_shear", cmap="coolwarm")

volume_spoiler = pv.read(volume_spoiler_path)
volume_spoiler.plot(scalars="pressure", cmap="coolwarm")
```

Steps:

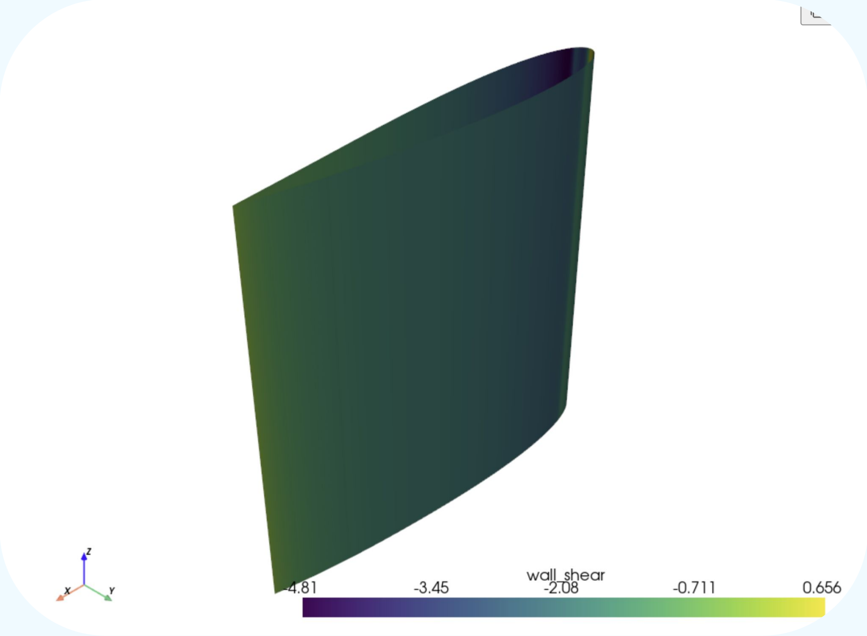
- Install PyVista Package
- Import it
- Read the different files
- Plot the files according to existing scalars

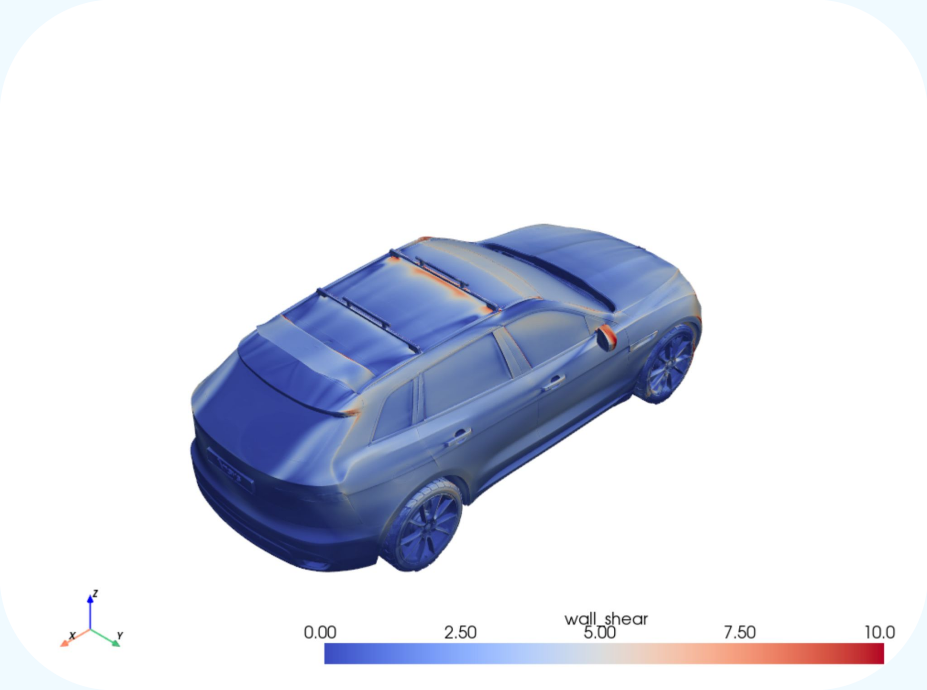
You can check which scalars are present in the object with:

volume_car.array_names

Note: SimAI proposes data visualization out of the box.

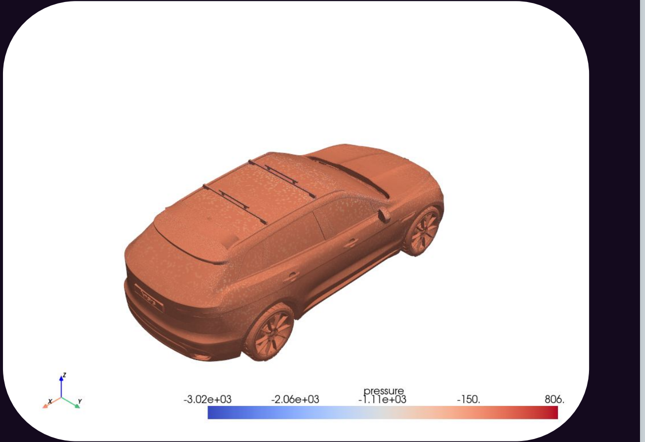
The results


z
x
y
wall_shear
-4.81
-3.45
-2.08
-0.711
0.656

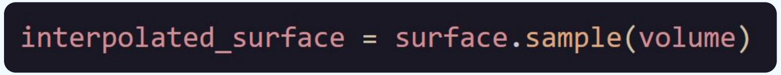

z
x
y
wall_shear
0.00
2.50
5.00
7.50
10.0

Objective

Compute drag using surface values.



- Surface does not contain the *Pressure* data but volume does.
- Use PyVista’s `*.sample()*` to interpolate nearest volume data to the surface.



```
interpolated_surface = surface.sample(volume)
```

- We can use either all scalars from the interpolated surface or a part of it.

Now that we have the pressure thanks to the interpolated surface, we can use this formula:

$$\text{Drag} = \int_S \left(\text{pressure} + \text{wall\_shear}\right)\, dS$$

We can approximate the drag from our data:

$$\text{Drag} \approx \sum_{i=1}^{N} \left(\text{pressure}_i + \text{wall\_shear}_i\right) \cdot \text{Area}_i$$

Check the Github and see more on SimAI

- ✓ Fast and reliable data-driven insights
- ✓ Intuitive and simple workflow
- ✓ Flexible deployment options
- ✓ Expansive design space exploration