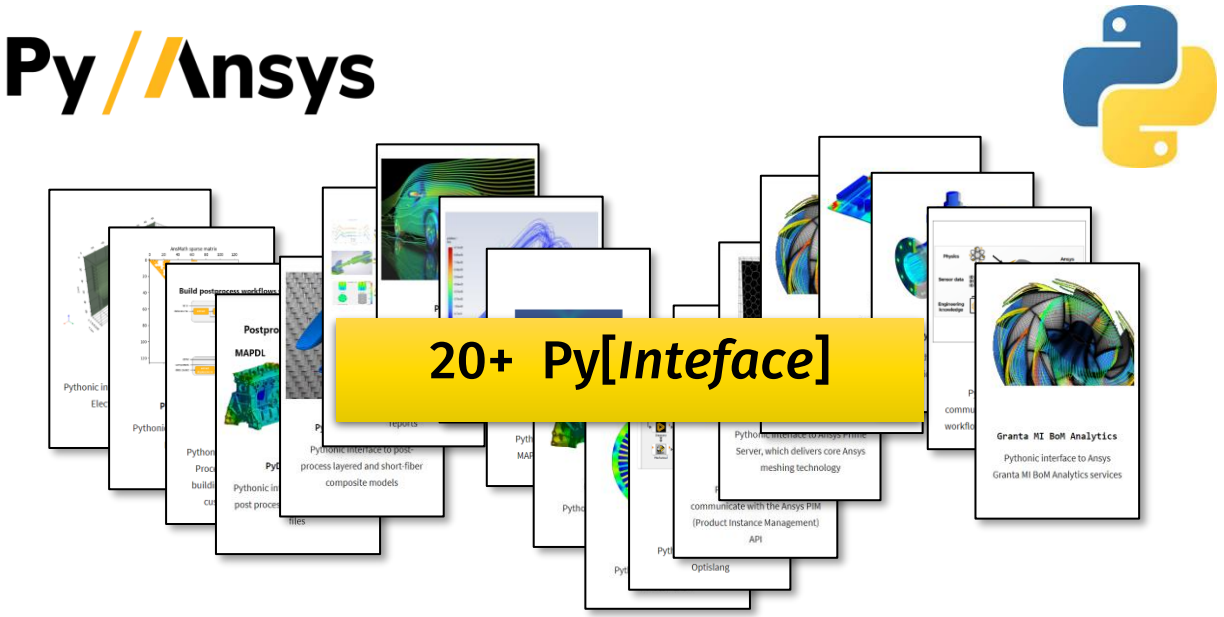
pyVšeCFD

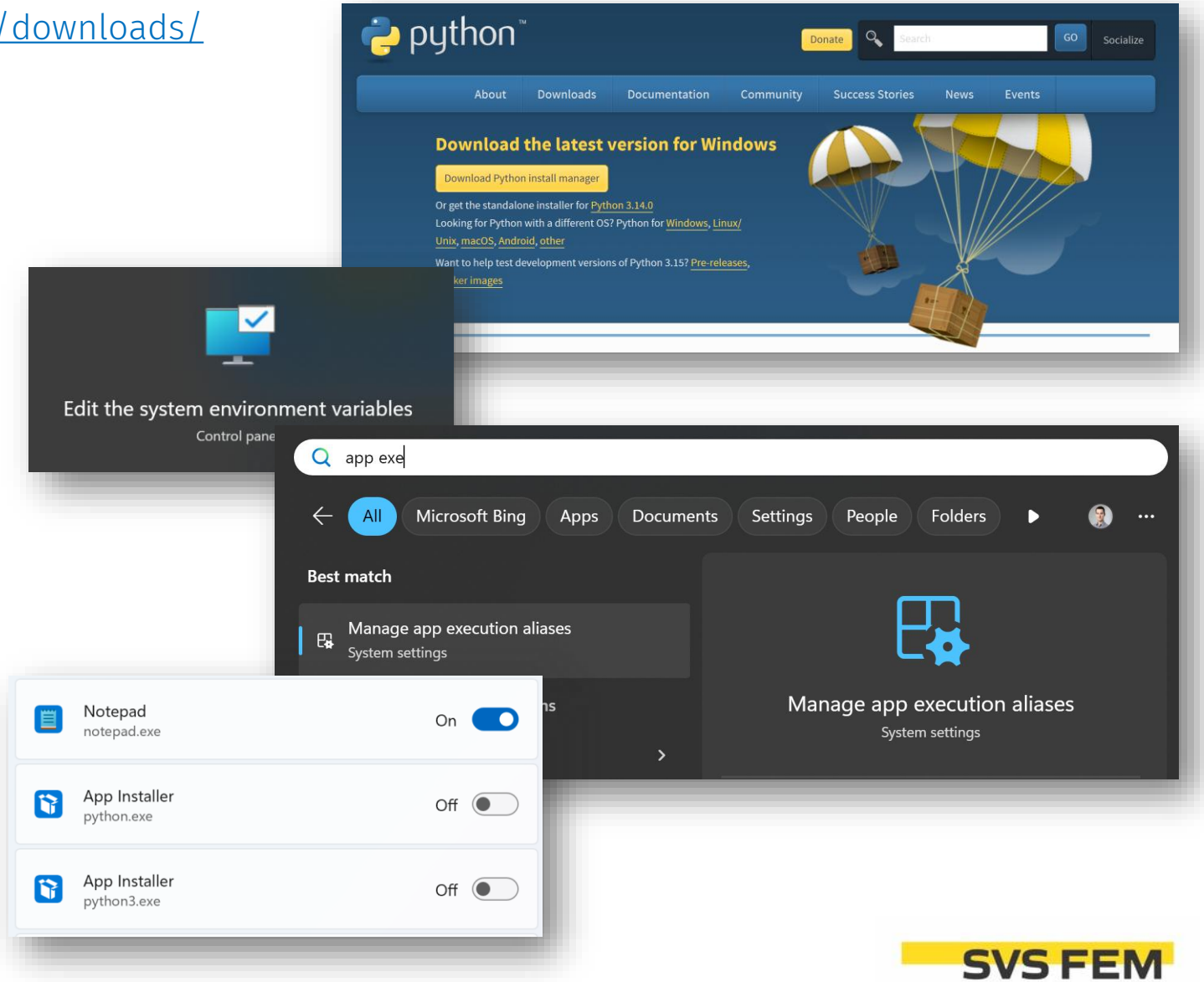
Úvod

- Relevantní balíčky
  - pyGeometry
  - pyTurboGrid
  - pyFluentMeshing
  - pyCFX
  - pyFluent
  - pyOptiSLang



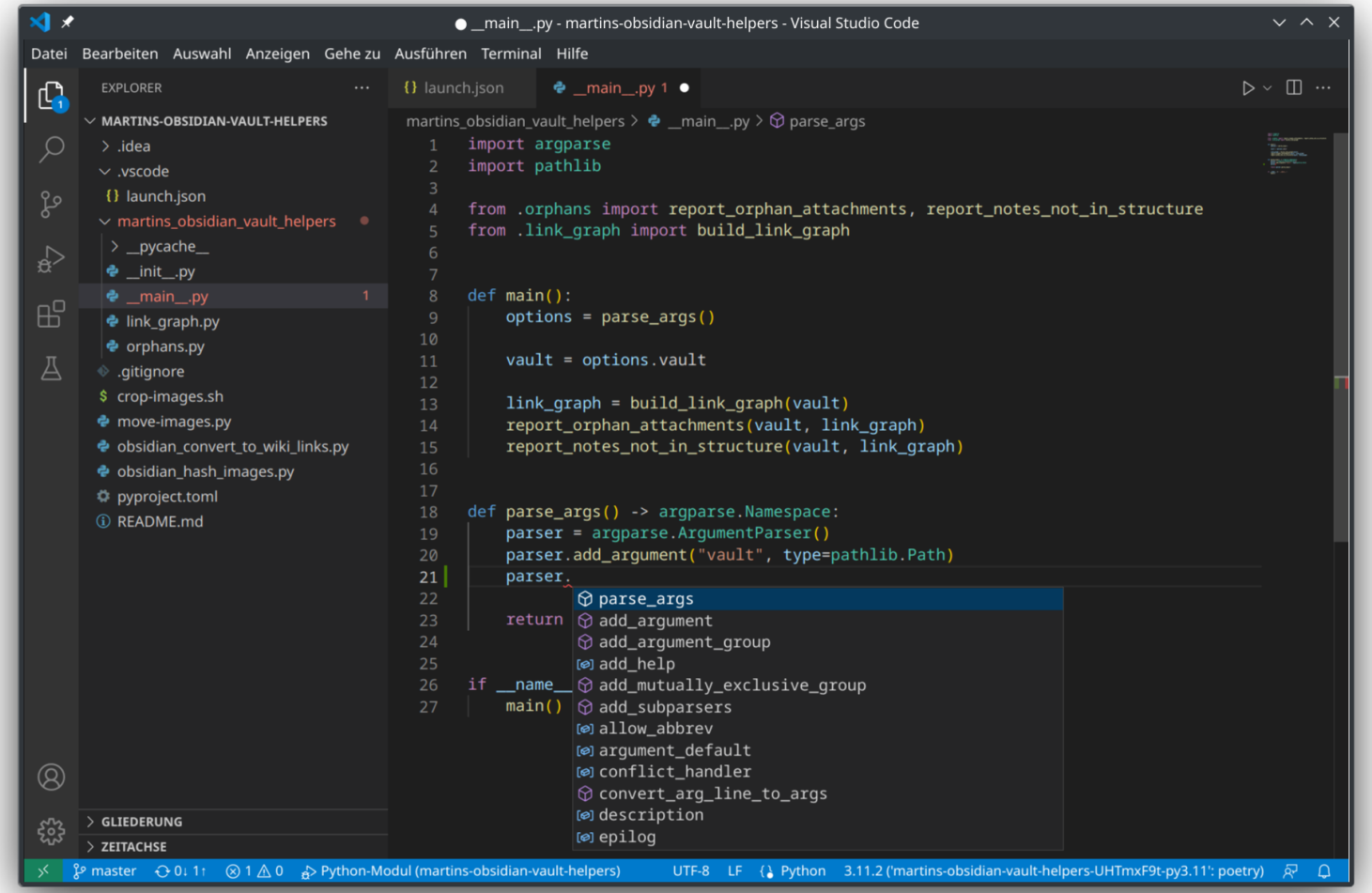
How to Start – Installing Python

1. Download **Python 3x** : https://www.python.org/downloads/
2. **Install Python** (recommended machine-wide)
   1. Disable path length limit
   2. Do not add folders to Path
3. Add Folders **to Path**
   1. C:\Program Files\Python3x
   2. C:\Program Files\Python3x\Scripts
4. Disable app **execution aliases**
   1. Start > Manage app execution aliases
   2. Turn off App Installer
      1. python.exe
      2. python3.exe
5. **Verify installation** in PowerShell
   1. python -V
   2. pip -V



How to Start – Installing PyAnsys Geometry

1. **Open terminal (cmd)** in folder where you want to work
2. **Run**
   1. py –m venv .venv
   2. .venv\Source\activate
   3. pip install ansys.geometry.core
3. Open **code editor** (VS Code)
4. Open **folder** in code editor
5. Create **.py** or **.ipynb** file



Documentation (https://docs.pyansys.com/)

PyAnsys

Home | Getting started | Projects | Open-source contributions | Blog

Search the docs ...

Product families

- AI 1
- Acoustics 1
- CAD 2
- Connect 2
- Digital Mission Engineering 1
- Digital Twin 1
- Electronics 11
- Embedded Software 1
- Fluids 10
- Materials 1
- Meshing 2
- Multiphysics 5
- Optics 2
- Platform 5
- Semiconductors 1
- Structures 12

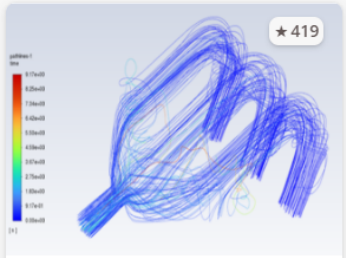
Show less

★487

PyMAPDL

Structures | Flagship

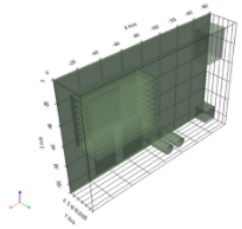
A Python client library for Ansys MAPDL (Mechanical APDL)



PyFluent

Fluids | Meshing | Flagship | Visualization

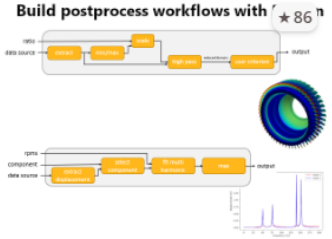
Pythonic interface to Ansys Fluent



PyAEDT

Electronics | Flagship

Pythonic interface to Ansys Electronics Desktop (AEDT)



PyDPF - Core

Structures | Multiphysics | Fluids | Post-processing | Flagship

Pythonic interface to the Data Processing Framework (DPF) for building advanced and customized workflows

★81

PyAnsys Geometry

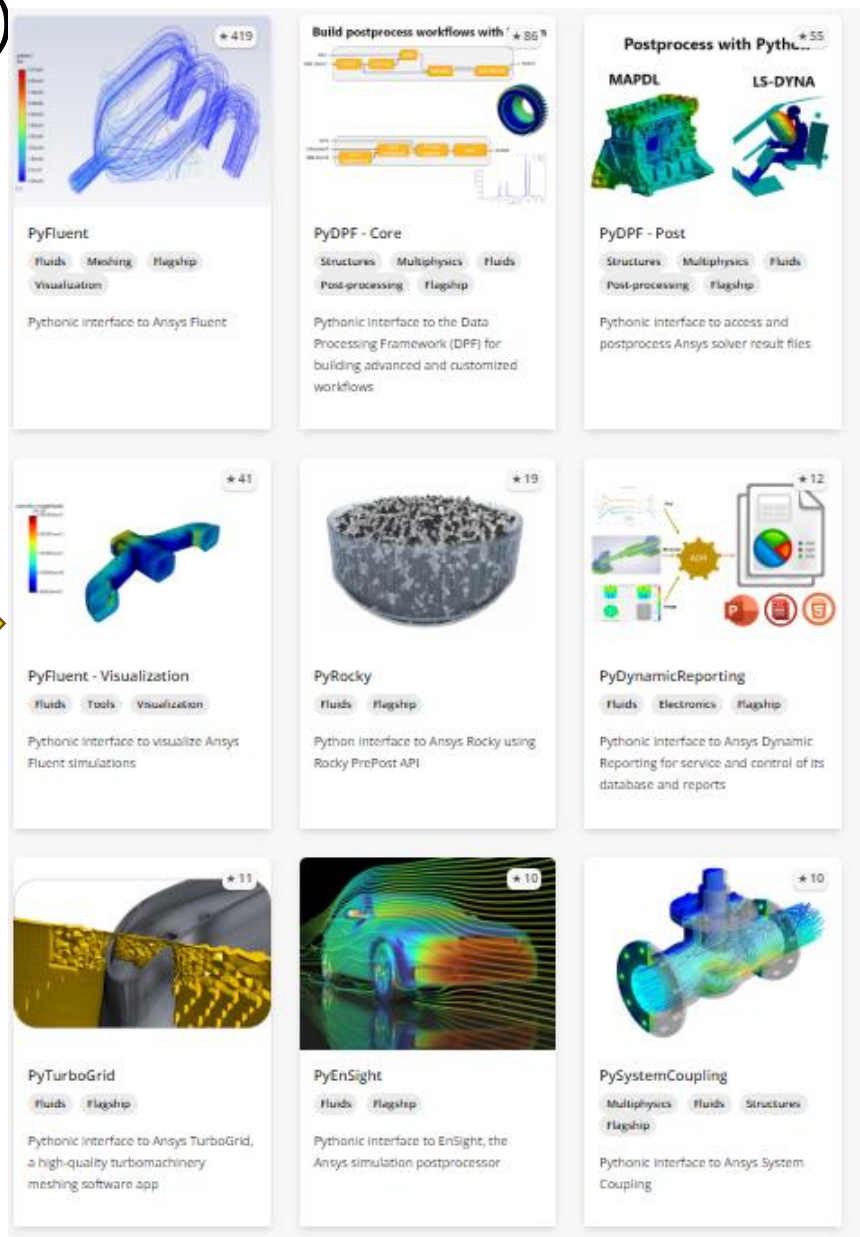
CAD | Flagship

Pythonic interface to the Ansys Geometry service

PyDYNA

Structures | Flagship

Pythonic interface to build the Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver, and postprocess its results



SVS FEM

Documentation – Cheat sheets

PyFluent cheat sheet

Version: 0.38.1

Quick Start

```
import ansys.fluent.core as pyfluent
from ansys.units import VariableCatalog as VC
from ansys.fluent.core import (
    ScalarFieldDataRequest,
    VectorFieldDataRequest,
    SurfaceFieldDataRequest,
    PathlinesFieldDataRequest,
    SurfaceDataType,
    )
from ansys.fluent.core.solver import *
```

Session Management

Launch sessions

```
solver = pyfluent.Solver.from_install()
meshing = pyfluent.Meshing.from_install()

# Connect to existing session
session = pyfluent.Solver.from_connection(ip,
    port, password)
```

File I/O

```
ReadCase(settings_source=solver)(
  file_name="<case_file>")
ReadCaseData(settings_source=solver)(
  file_name="<data_file>")

WriteCase(settings_source=solver)(
  file_name="<output_case_file>")
WriteCaseData(settings_source=solver)(
  file_name="<output_case_file>")
```

Meshing Workflow

Watertight geometry workflow

```
watertight = meshing.watertight()

import_geom = watertight.import_geometry
import_geom.file_name = "<geometry_file>"
import_geom.length_unit = "mm"
import_geom()

surf_mesh = watertight.create_surface_mesh
surf_mesh.cfd_surface_mesh_controls.max_size = 0.3
surf_mesh()

watertight.describe_geometry.setup_type = "fluid"
watertight.describe_geometry()

volume_mesh = watertight.create_volume_mesh_wtm
volume_mesh.volume_fill = "poly-hexcore"
volume_mesh()

solver = meshing.switch_to_solver()
```

Physics Setup

Boundary conditions

```
inlet = VelocityInlet(solver, name="cold-inlet")
inlet.momentum.velocity_magnitude = 0.4
turbulence = inlet.turbulence
turbulence = (turbulence.turbulence_specification.
  INTENSITY_AND_HYDRAULIC_DIAMETER)
turbulence.turbulent_intensity = 0.05
turbulence.hydraulic_diameter = "4 [in]"
inlet.thermal.temperature = 293.15
```

Materials

```
materials = Materials(solver)
air_copy =
    materials.fluid.make_a_copy(from_="air",
    to="air-copied")
materials.fluid["air-copied"].viscosity = 1.81e-05

# Create new material
my_solid = materials.solid.create("my-solid")
my_solid.density.value = 2650
my_solid.thermal_conductivity.value = 7.6
```

Cell zone conditions

```
cell_zones = CellZoneConditions(solver)
cell_zones.fluid["elbow-fluid"].general.material =
    "air-copied"
```

Models

```
models = Models(solver)
models.energy.enabled = True
models.energy.viscous_dissipation = True

viscous = Viscous(solver)
viscous.model = viscous.model.K_EPSILON
k_epsilon_model = viscous.k_epsilon_model
k_epsilon_model = k_epsilon_model.REALIZABLE

rad = Radiation(solver)
rad.model = rad.model.MONTE_CARLO
rad.solve_frequency.number_of_histories = 1e7
species = Species(solver).model.option
species = species.SPECIES_TRANSPORT
```

Solution

```
soln = Solution(solver)
methods = soln.methods
methods.p_v_coupling.flow_scheme = "Coupled"
grad_scheme =
    methods.spatial_discretization.gradient_scheme
grad_scheme = grad_scheme.GREEN_GAUSS_NODE_BASED
```

Report definitions

```
rep_defs = ReportDefinitions(solver)
surf_rep = rep_defs.surface["outlet-temp-avg"] = {}
soln_report_type =
    rep_defs.surface["outlet-temp-avg"].report_type
soln_report_type = soln_report_type.SURFACE_AREA
rep_defs.surface["outlet-temp-avg"].field =
    VC.TEMPERATURE
```

Initialize and solve

```
initialization = Initialization(solver)
init_type = initialization.initialization_type
init_type = init_type.HYBRID
Initialize(solver)()

run_calc = RunCalculation(solver)
run_calc.parameters.iter_count = 100
Iterate(solver)()

# Check convergence
Monitor(solver).residual.plot()
```

Post-Processing

Field data access

```
# Ensure the session is initialized before
# requesting field data

field_data = solver.fields.field_data

abs_press_req = ScalarFieldDataRequest(
    field_name=VC.ABSOLUTE_PRESSURE,
    surfaces=["cold-inlet"],)

abs_pressure =
    field_data.get_field_data(abs_press_req)

vel_req = VectorFieldDataRequest(
    field_name=VC.VELOCITY,
```

PyOptiSLang cheat sheet

Version: 1.4.1

Basic optiSLang Operations

Create a new optiSLang instance

PyOptiSLang automatically detects the newest standard installation paths and by default spawns a new optiSLang instance locally. When used this way, optiSLang creates a new project in a temporary directory. A new instance can be created using either the context manager syntax (recommended):

```
# Create an Optislang instance using context
    manager
from ansys.optislang.core import Optislang
with Optislang() as osl:
    print(osl)
```

Or directly:

```
# Create an Optislang instance directly
osl = Optislang()
print(osl)
osl.dispose()
```

In this case, the instance must be disposed of when it is no longer needed.

Connect to existing optiSLang

To connect to an already running optiSLang instance, use the `local_server_id` or `host` and `port` arguments.

```
# Connect to an existing optiSLang instance
from ansys.optislang.core import Optislang
osl = Optislang(host="127.0.0.1", port=5310)
print(osl)
osl.dispose()
```

If optiSLang was started with the `shutdown_on_finished` argument set to `False`, it won't shut down automatically. To shut down manually, the `shutdown()` command must be called prior to disposing the instance.

Start optiSLang in GUI mode

By default, optiSLang is started in batch mode. To start it in GUI mode, set the `batch` argument to `False`.

```
# Start optiSLang in GUI mode
from ansys.optislang.core import Optislang
with Optislang(batch=False) as osl:
    print(osl)
```

Find available optiSLang installations

To use a specific optiSLang installation, use the `executable` argument during initialization. Convenience functionality that provides an ordered dictionary sorted by version is also available:

```
# Find and use a specific optiSLang installation
from ansys.optislang.core import Optislang
from ansys.optislang.core.utils import
    find_all_osl_exec

osl_execs = find_all_osl_exec()
latest_exec = next(iter(osl_execs.values()))[0]
with Optislang(executable=latest_exec) as osl:
    print(osl)
```

Project Management

Load a project

The path to the project can be specified either on initialization:

```
# Load project during initialization
from ansys.optislang.core import Optislang
from pathlib import Path

path = Path(r"C:\path\to\project.opf")
with Optislang(project_path=path) as osl:
    print(osl)
```

or after initialization:

```
# Load project after initialization
with Optislang() as osl:
    osl.application.open(path)
    print(osl)
```

Start project execution

```
# Start project execution
from ansys.optislang.core import Optislang
from pathlib import Path

path = Path(r"C:\path\to\project.opf")
with Optislang(project_path=path) as osl:
    osl.application.project.start()
```

Save projects

The project can be saved using the following commands: `save()`, `save_as()` and `save_copy()`. All these commands are provided by the `Application` class.

```
# Save project with a new name
from ansys.optislang.core import Optislang
from pathlib import Path

new_path = Path().cwd() / "project.opf"
with Optislang() as osl:
    osl.application.save_as(new_path)
```

Obtain basic project information

Print the basic project information and nodes at the top level:

```
# Get basic project information
from ansys.optislang.core import Optislang
from pathlib import Path

path = Path(r"C:\path\to\project.opf")
with Optislang(project_path=path) as osl:
    project = osl.application.project
    print(project.get_name())
    print(project.get_status())
    rs = project.root_system
    for node in rs.get_nodes():
        print(
            f"Node: {node.get_name()}, " +
            f"Type: {node.type}"
        )
```

Design Evaluation

Evaluate design

This functionality is implemented for projects that have a parametric on top level.

Figure 1: Evaluate design

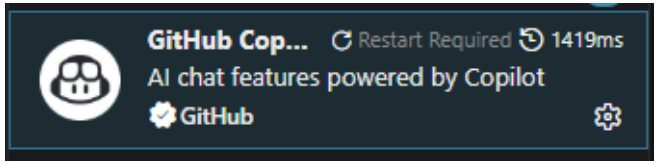
To evaluate a design, query the root system for the reference design, modify parameters as needed and evaluate them. Please note that only the last evaluated design is stored in the optiSLang database.

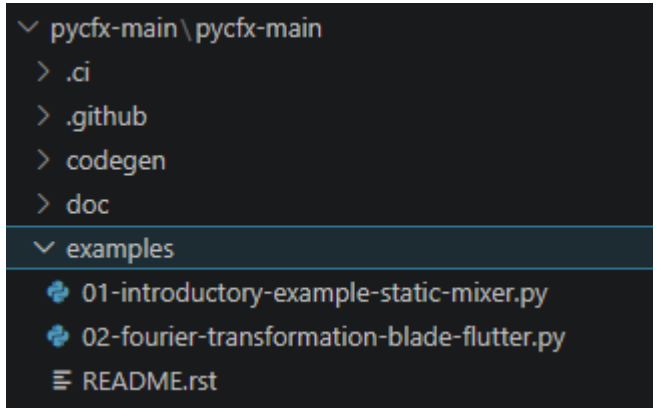
```
# Evaluate a design with modified parameters
from ansys.optislang.core import Optislang
from pathlib import Path

path = Path(r"C:\path\to\project.opf")
with Optislang(project_path=path) as osl:
    rs = osl.application.project.root_system
    design = rs.get_reference_design()
    design.set_parameter_value(
        name="Parameter1",
        value=10
    )
    out_design = rs.evaluate_design(design)
    print([resp.value for resp in
        out_design.responses])
```

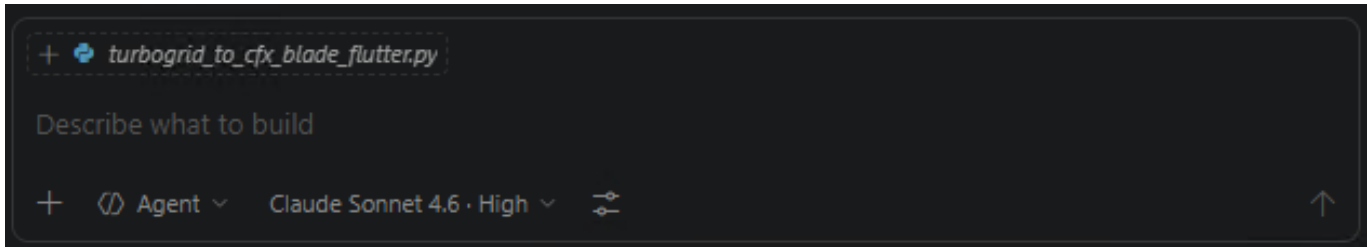
Designs can also be created from scratch:

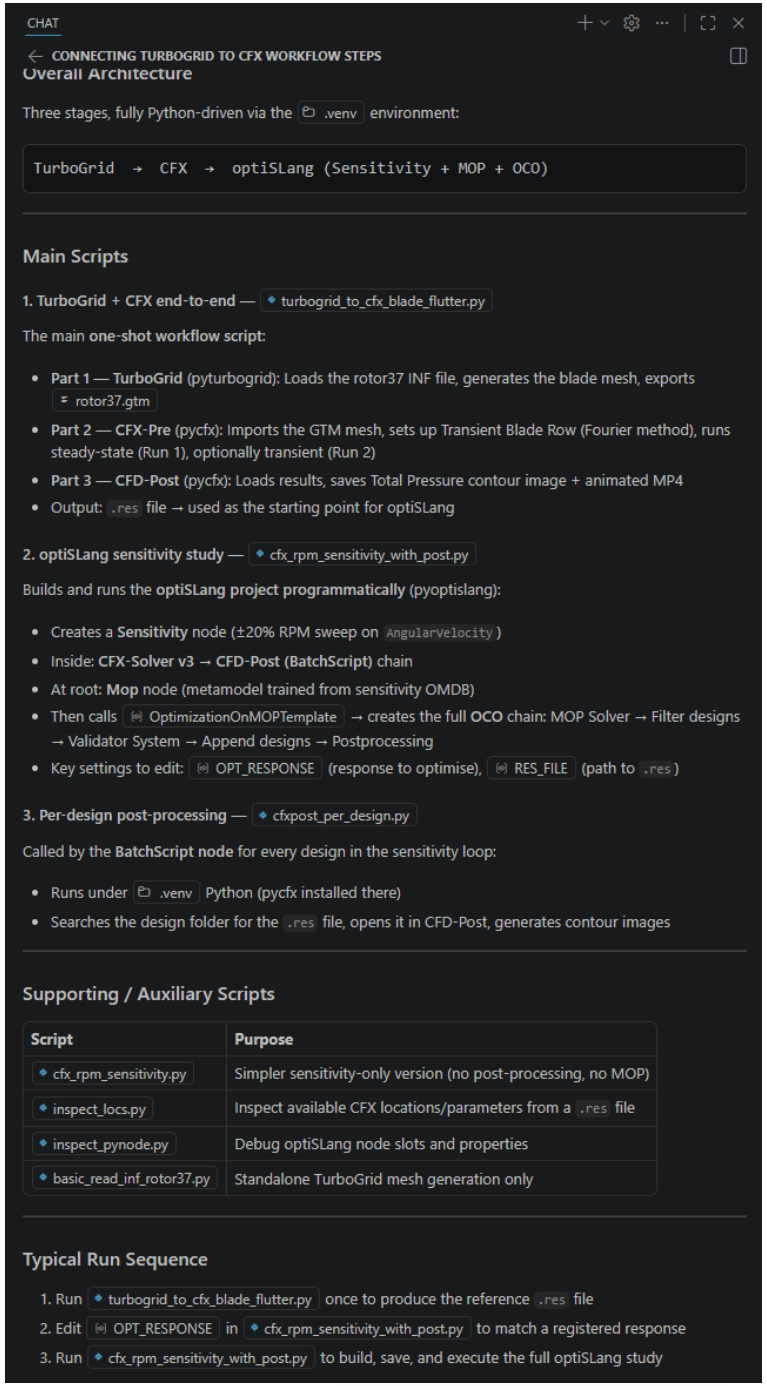
Co-pilot

- Odkazování na příklady v dokumentaci
- Odkazování na předchozí hotové skripty
- Rychlé vyhledávání správných funkcí
- Tvorba dokumentace





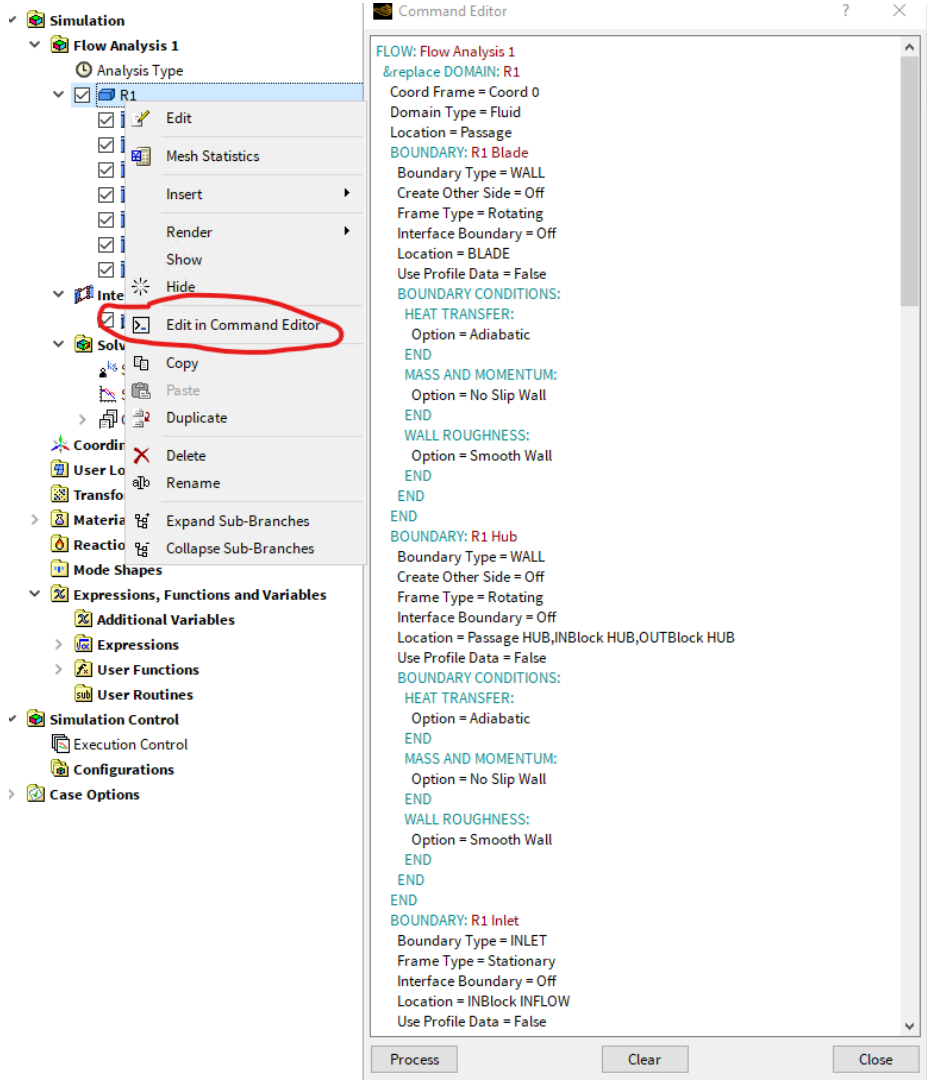




pyCFX

```
class UIMode(CFXEnum):
    """Provides the supported user interface mode of CFX."""

    NO_GUI = ("batch",)
    HIDDEN_GUI = ("",)
    # TODO: Need to handle 'show gui' option in CFX engine
    GUI = ("",)

    def _default(self):
        return self.HIDDEN_GUI
```

- Chybí GUI (UIMode)
  - Příkaz GUI vede na HIDDEN_GUI
- Execute_ccl (lze exportovat z existujícího casu)
  - pypre.execute_ccl(mesh_transformation_ccl)
- Špatný příkaz vede na výpis erroru, nikoliv zamrznutí celé aplikace.
- Solution je nezávislý.
  - Kód dál pokračuje až po pysolve_ini.solution.wait_for_run()
- Validace před spuštěním
  - physics_messages = pypre.setup.get_physics_messages(severity="All")
- Dědění session mezi pre,solver,post
  - pysolve_ini = pycfx.Solver.from_session(pypre)
- Proměnná pro složku, kde běží výpočet
  - pysolve_ini.solution._run_directory



pyTurboGrid

- Taky chybí GUI -> turbogrid = launch_turbogrid() -> args_list.append("-py")
- Většina sizingů přes set_obj_param (
  - turbogrid.set_global_size_factor(str(TG_GLOBAL_SIZE_FACTOR))
  - turbogrid.set_obj_param(object="/MESH DATA", param_val_pairs=f"ATM Proportional BL Factor Base = 2“)
- turbogrid.unsuspend(object="/TOPOLOGY SET")
- Save mesh/state
  - turbogrid.save_mesh(filename=mesh_file_name)
  - turbogrid.save_state(filename=state_file_name)
- Rozdíl před a po změnách
  - state_before = turbogrid.get_state()
  - turbogrid.set_global_size_factor("0.8")
  - turbogrid.unsuspend(object="/TOPOLOGY SET")
  - state_after = turbogrid.get_state()
- turbogrid.send_ccl("""
  MESH DATA:
  ATM Proportional BL Factor Base = 4.5
  END """)

```
Between Boundary Layers End Ratio = 1 -> 2
Between Boundary Layers Number Of Elements = 6 -> 4
Global Size Factor = 1.0 -> 0.8
Hub Boundary Layer Expansion Rate = 2.07071 -> 2.21374
Hub Boundary Layer Number Of Elements = 8 -> 7
Hub Vertex Distance = 0.0028308 [cm] -> 0.00542988 [cm]
Hub Vertex Offset = 0.000431097 -> 0.000826904
Hub Vertex YPlus = 20.8068 -> 39.9104
Number Of Constant Spanwise Blade Elements = 5 -> 3
Number Of Elements = 102753 -> 53006
Number Of Inlet Elements = 12 -> 11
Number Of Outlet Elements = 15 -> 13
Number Of Spanwise Blade Elements = 21 -> 17
Number Of Vertices = 113256 -> 59778
Shroud Boundary Layer Expansion Rate = 1.89062 -> 1.97616
Shroud Boundary Layer Number Of Elements = 8 -> 7
Shroud Vertex Distance = 0.0028308 [cm] -> 0.00542988 [cm]
Shroud Vertex Offset = 0.000431097 -> 0.000826904
Shroud Vertex YPlus = 20.8068 -> 39.9104
Blade Vertex Distance = 0.00283082 [cm] -> 0.0054299 [cm]
Blade Vertex Offset = 0.00283082 -> 0.0054299
Blade Vertex YPlus = 20.8069 -> 39.9105
Actual Edge Count = 21 -> 17
Target BL Element Count = 21 -> 17
Target First Element Offset = 0.00283082 [cm] -> 0.0054299 [cm]
Target First Element YPlus = 20.8069 -> 39.9105
```

pyOptiSlang

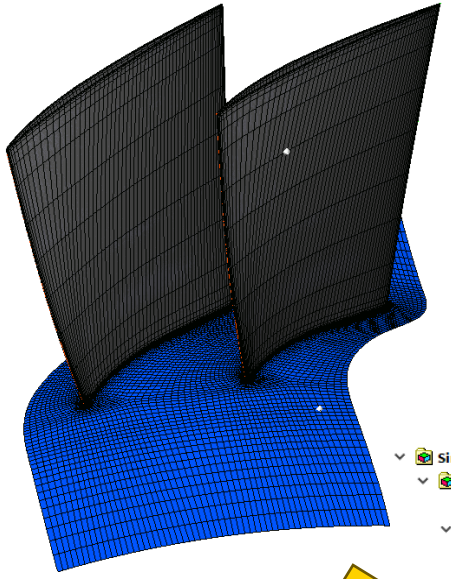
- Automatizované vytvoření systému a následná parametrická studie
- OptiSLang už takto řeší hodně integrací přes Python
  - C:\Program Files\ANSYS Inc\v261\optiSLang\scripting\integrations\

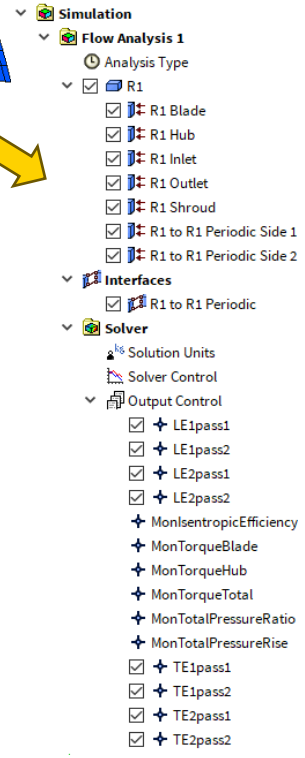
- CFX_Solver_v3.load() načte .res file a automaticky načte expressiony a monitory jako parametry
- OptimizationOnMOPTemplate odpovídá procesu přes GUI wizard
- DesignFlow.RECEIVE_SEND propojuje jednotlivé bloky dohromady

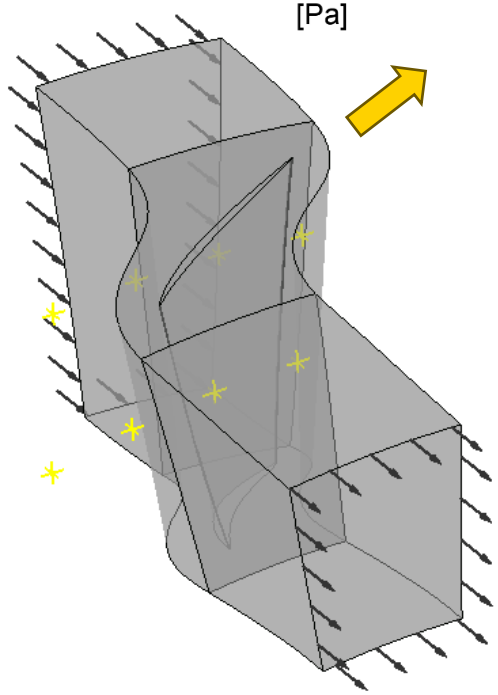
- Taky bez GUI
- Nejdou bodový pointy jako výstupní proměnná – „“ , které se pak špatně naparsujou

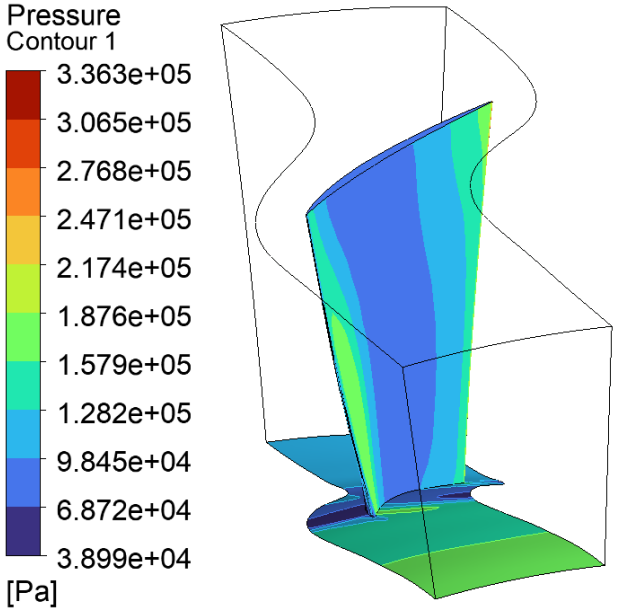
Ukázka

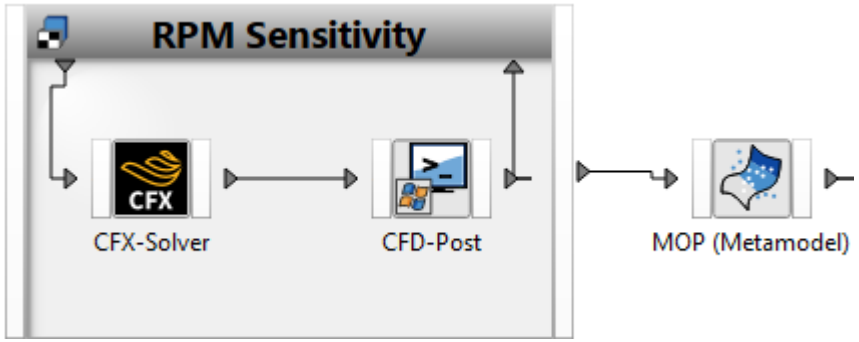
- BladeGen
- hub
- profile
- shroud











Díky za pozornost
a zůstaňme ve spojení