PyAnsys

Integration von Ansys in Simulation Workflows

Dario Rüegg

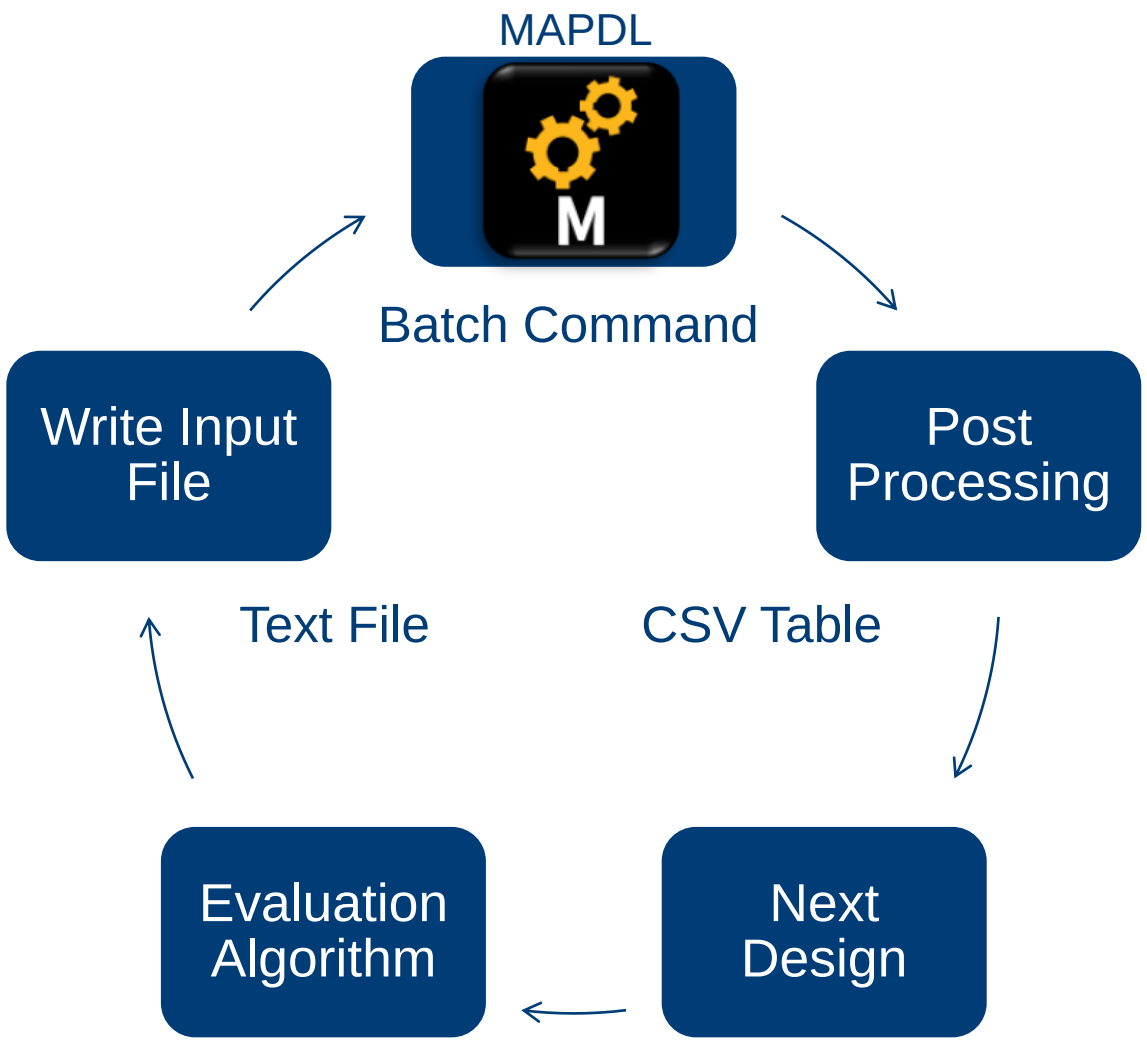
PyAnsys Stand

CADFEM ANSYS SIMULATION CONFERENCE
RAPPERSWIL 2023

CADFEM Ansys / ELITE CHANNEL PARTNER

SIMULATION CONFERENCE

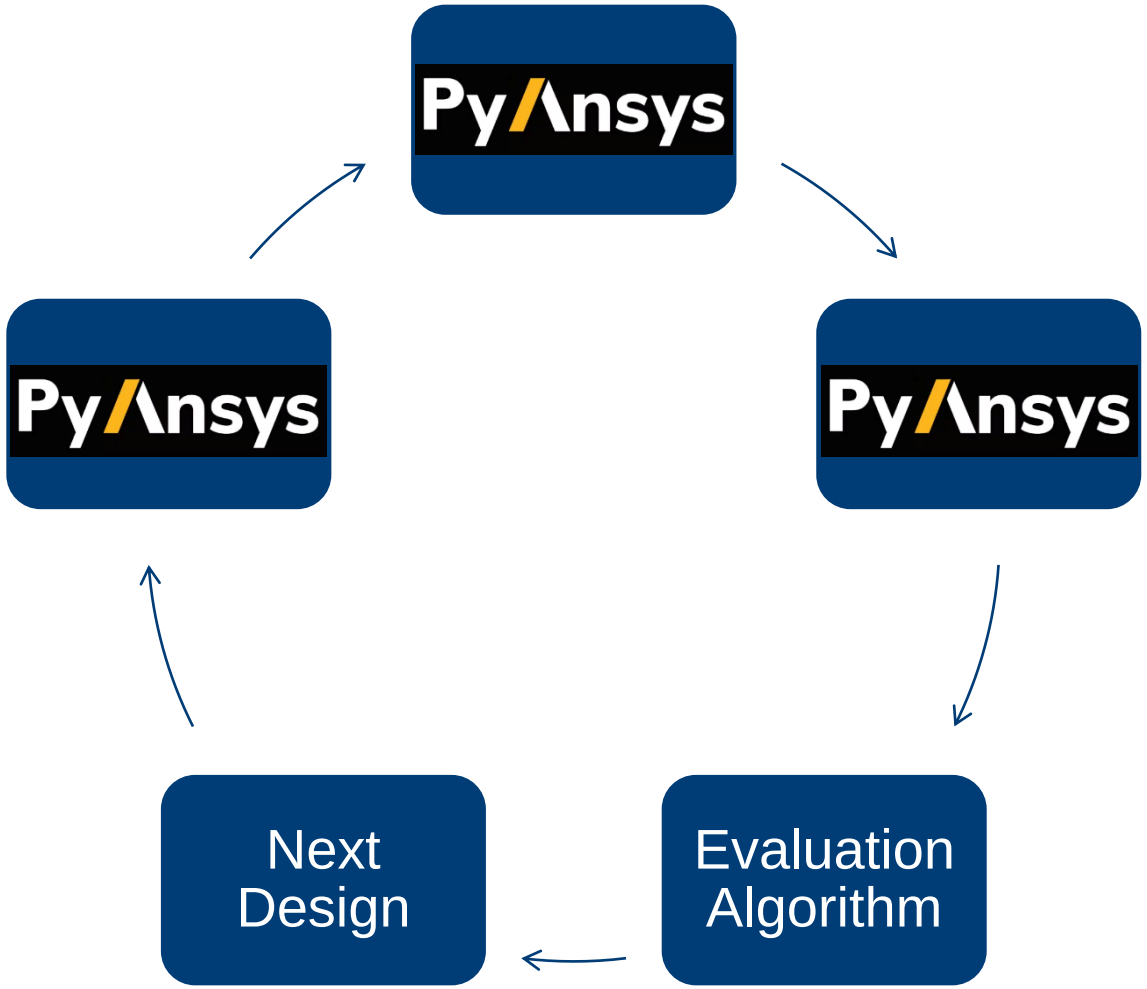
Simulation Workflows



Damals im Studium:

- Python Script steuert Workflow
- Iterativer Loop:
  - Schreiben eines MAPDL input file. Pre-Processing, Meshing, Solve, Post
  - Aufruf MAPDL Solver über batch commands.
  - Warten für Feedback
  - Auslesen Print-Out einzelner Resultate

Simulation Workflows



Damals im Studium:

- Python Script steuert Workflow
- Iterativer Loop:
  - Schreiben eines MAPDL input file. Pre-Processing, Meshing, Solve, Post
  - Aufruf MAPDL Solver über batch commands.
  - Warten für Feedback
  - Auslesen Print-Out einzelner Resultate

Workflow Ansatz heutzutage:

PyAnsys

CADFEM

Dominik Gresch

- **ETH Zurich**
  Bachelor & Master of Science: Physics
  PhD, Condensed Matter and Materials Physics
  Research and Teaching Assistant
  - Algorithms development, library design, high-throughput calculations
  - Programming techniques for scientific simulations, solid state theory, computational quantum physics
- **Experis Switzerland**
  Simulation Engineer – Microsoft Quantum
- **Ansys** – Senior R&D Engineer
  ACP Workflow
  PyAnsys

PyAnsys

Ansys Simulation als programmierbare Bibliothek

Dominik Gresch
Senior R&D Engineer

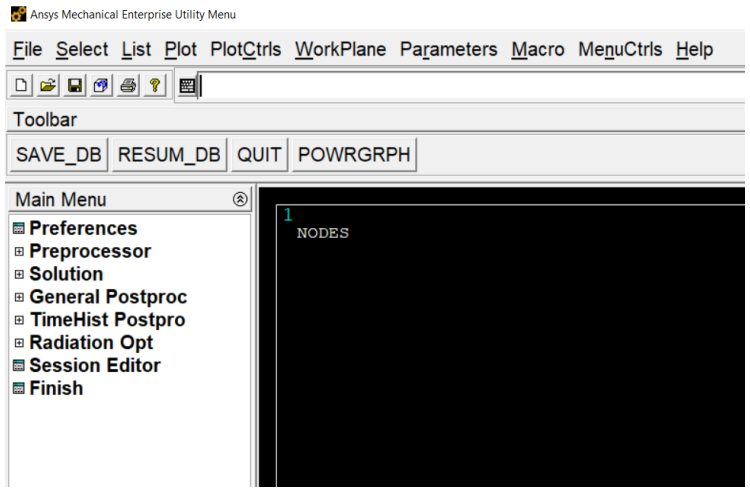
Agenda

- Was ist PyAnsys?
- Erste Schritte + Ressourcen
- Beispiele
  - PyMAPDL
  - PyFluent
  - PyDPF
  - PyPrimeMesh

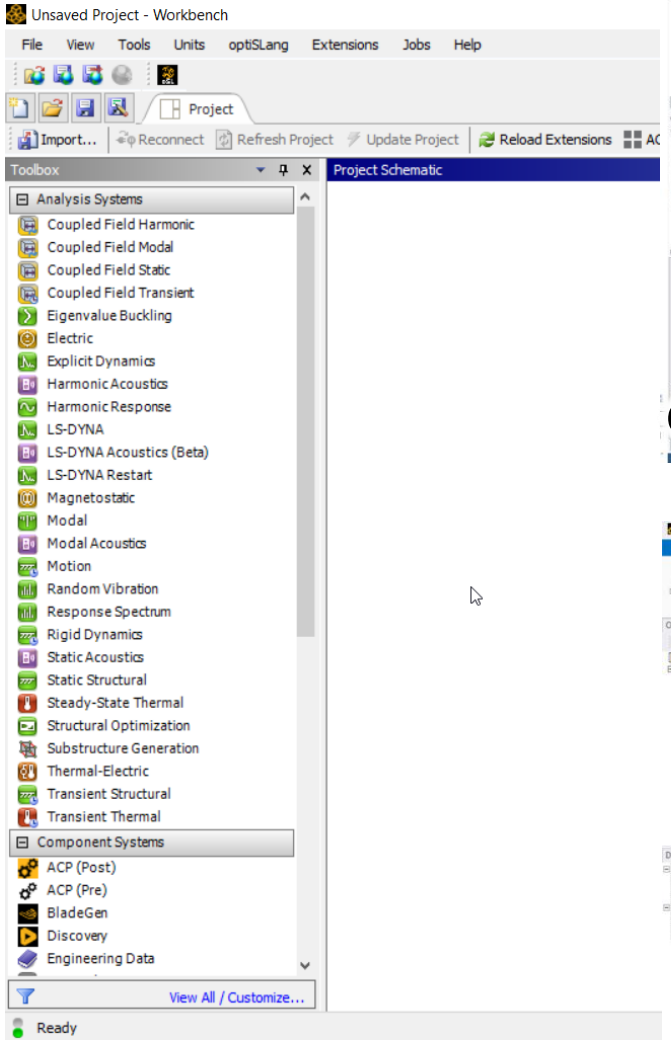
Was ist PyAnsys?

Automatisierung von Ansys Simulationen durch Scripting

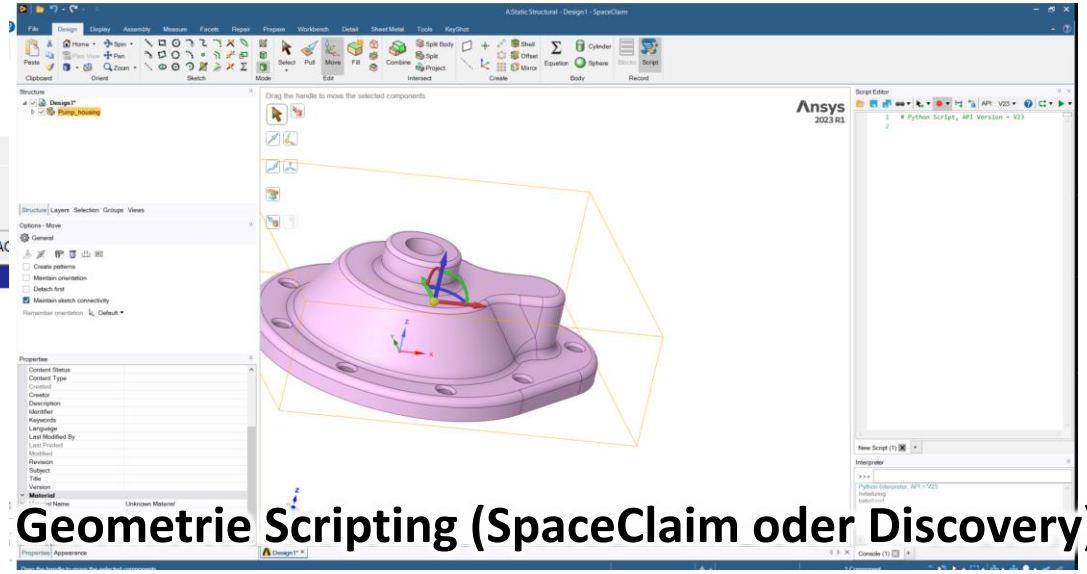
Zuvor: Scripting im Produkt



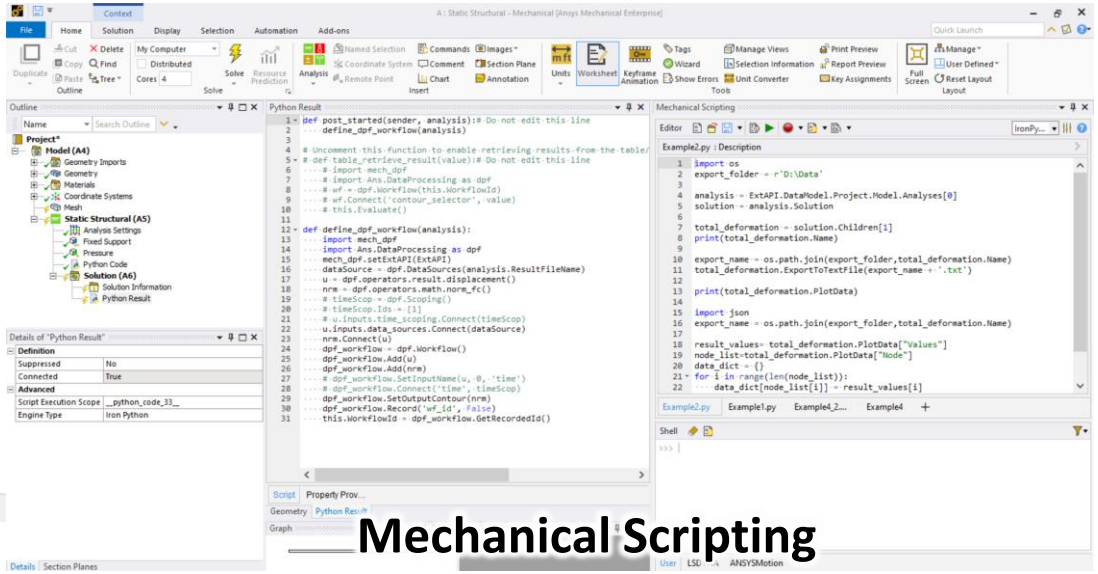
MAPDL Scripting



WB Scripting

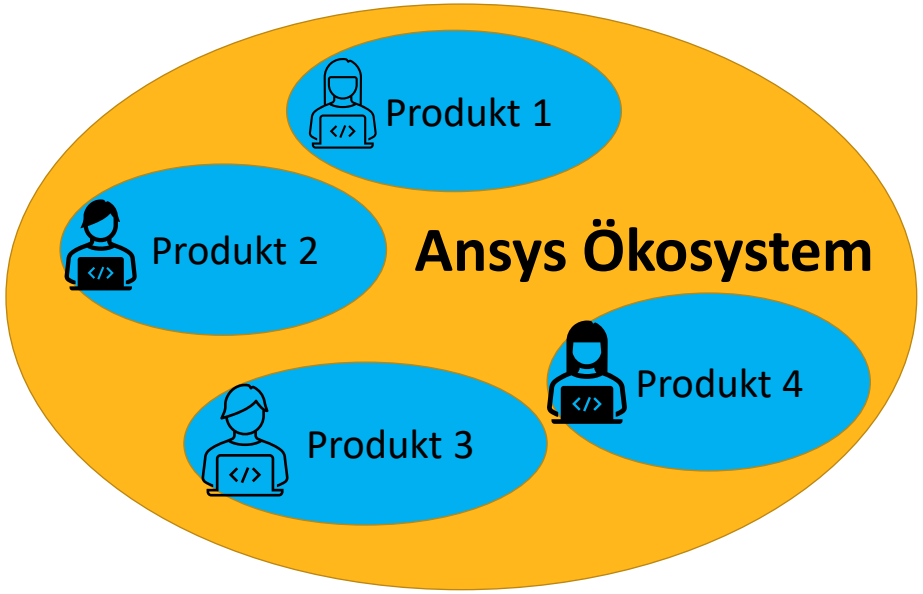


Geometrie Scripting (SpaceClaim oder Discovery)

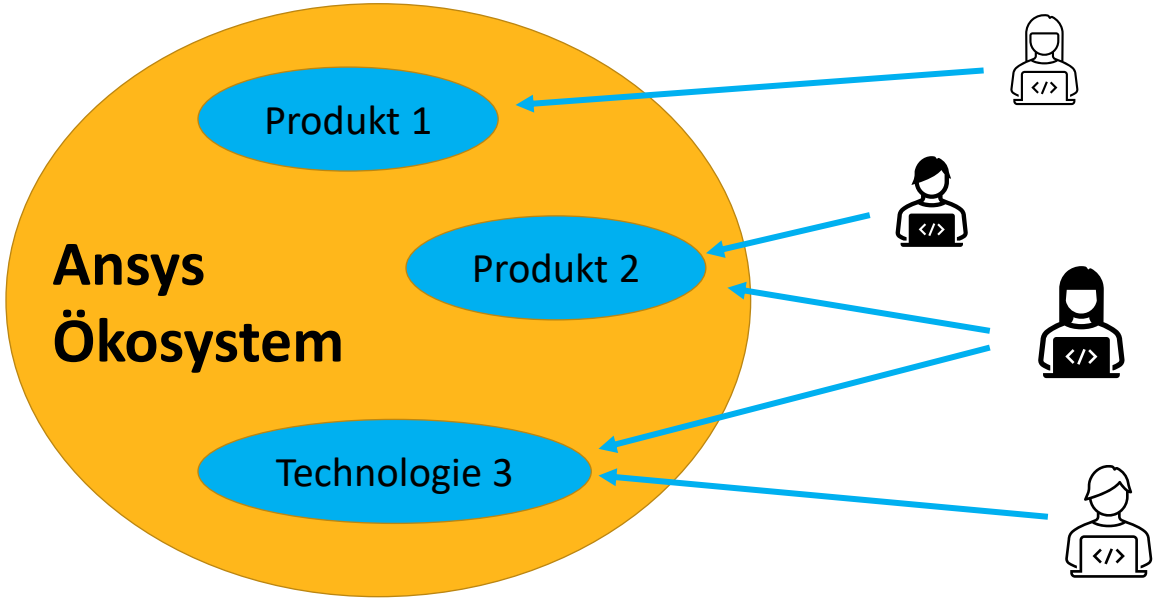


Mechanical Scripting

Zwei Ansätze für Scripting



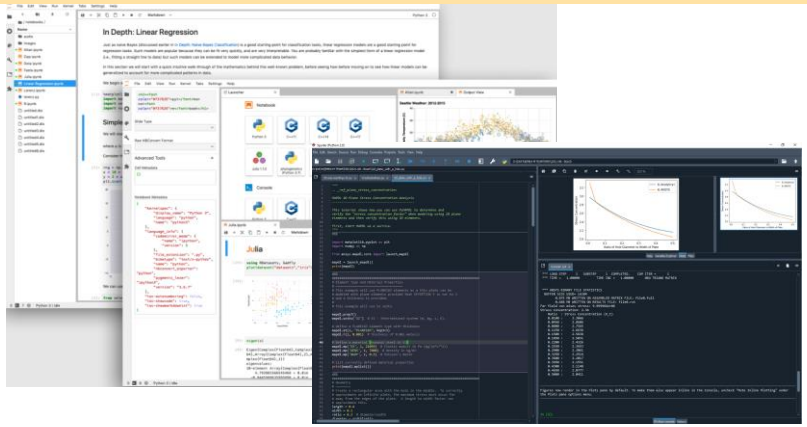
Scripting im Produkt



PyAnsys Scripting

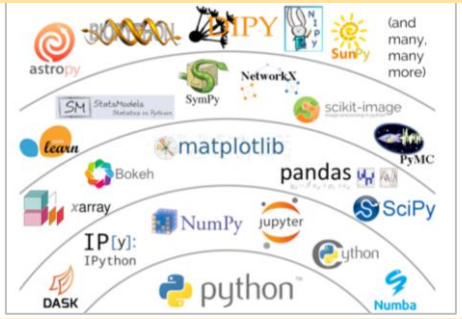
Scripting aus der Kundenumgebung

Benutzer wählt das UI, um Python – Code zu schreiben



Integration in Apps

Kann beliebige Python – Bibliotheken benutzen



Benutzerumgebung

PyAnsys Technologie

Produkte

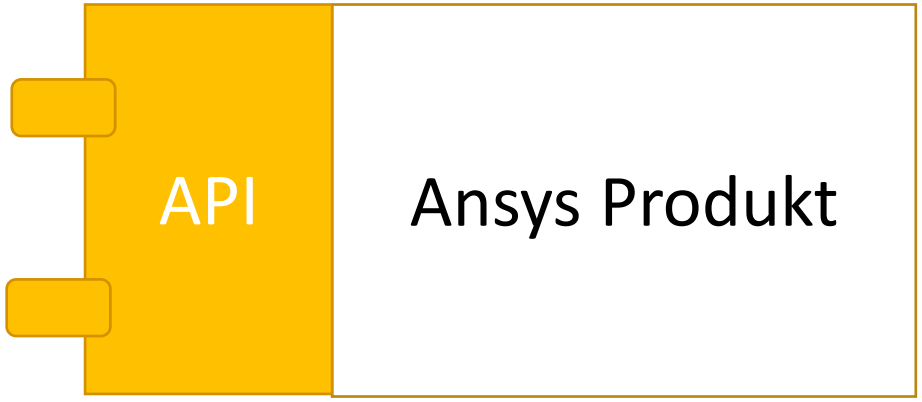
- API — Ansys MAPDL
- API — Ansys Fluent
- API — Ansys Electronic Desktop

Technologien

- API — Meshing
- API — Data Processing Framework

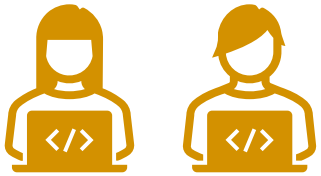
Ansys Technologie

Entkoppelung durch Programmierschnittstellen



Ansys Produkte + Technologien können über Programmierschnittstellen (APIs) benutzt werden

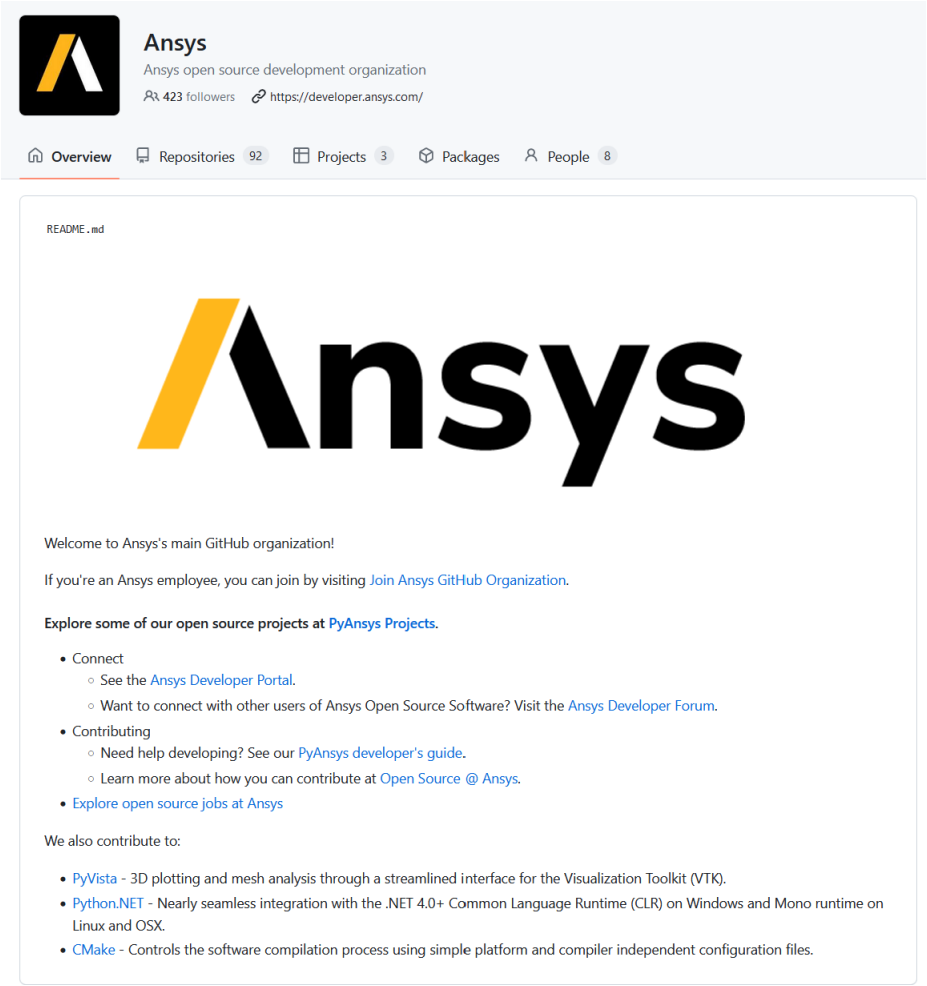
Entkoppelung: Client/Server Architektur



Client:
Benutzeroberfläche

Server:
Berechnet + speichert Daten

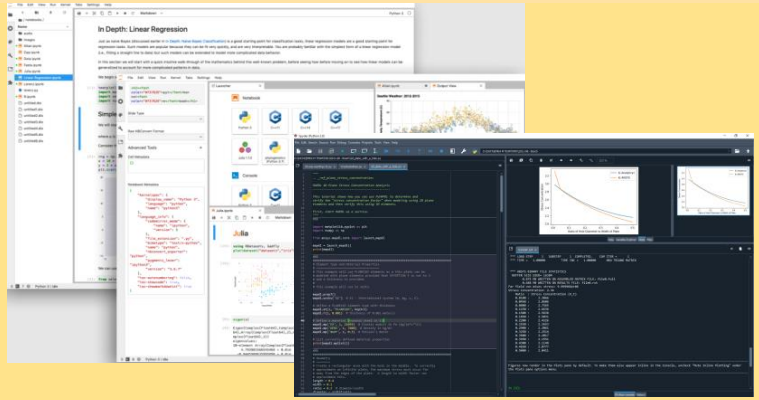
Open Source



- Source code frei verfügbar
- Direkte Interaktion mit Entwicklern über Github
- “Release when ready” - Modell

Was ist Open-Source in PyAnsys?

Benutzer wählt das UI, um Python – Code zu schreiben

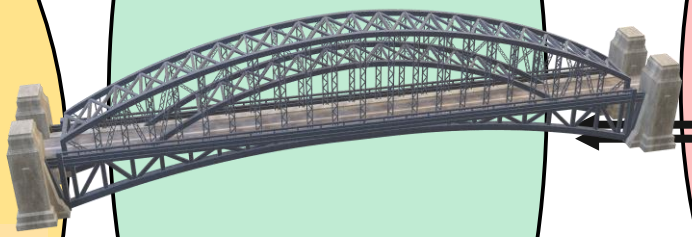


Integration in Apps

Kann beliebige Python – Bibliotheken benutzen

Je nach Anwendung: Benutzer kann auch proprietäre Anwendungen entwickeln

PyAnsys Technologie



Open Source

Produkte

API — Ansys MAPDL

API — Ansys Fluent

API — Ansys Electronic Desktop

Technologien

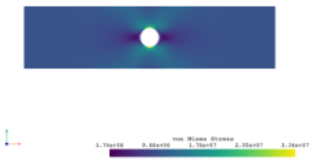
Meshing

Data Processing Framework

Bleibt proprietär, benutzt Produktlizenzen

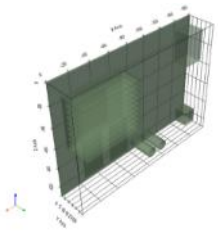
PyAnsys Bibliotheken

Simulationsbibliotheken



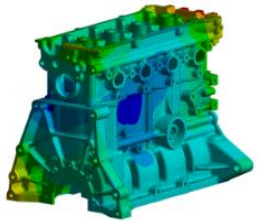
PyMAPDL

Pythonic interface to Ansys MAPDL (Mechanical APDL)



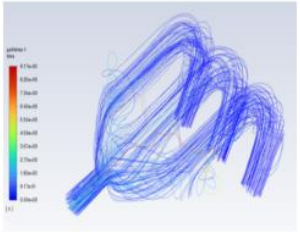
PyAEDT

Pythonic interface to AEDT (Ansys Electronic Desktop)



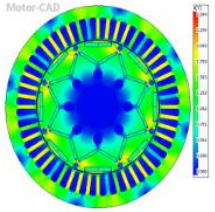
PyMechanical

Pythonic interface to Ansys Mechanical



PyFluent

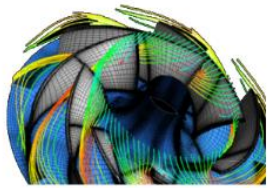
Pythonic interface to Ansys Fluent



PyMotorCAD

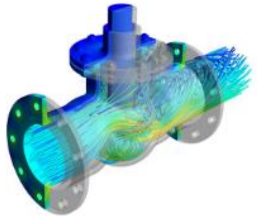
Pythonic interface to Ansys Motor-CAD.

Hilfsbibliotheken



PyPrimeMesh

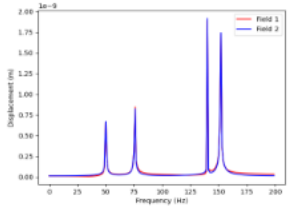
Pythonic interface to Ansys Prime Server, which delivers core Ansys meshing technology



PySystem Coupling

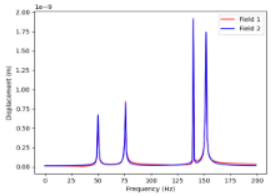
Pythonic interface to communicate with Ansys System Coupling

Postprocessing



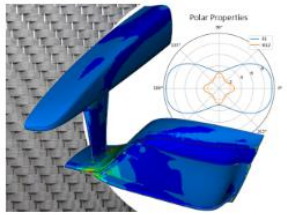
PyDPF-Core

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



PyDPF-Post

Pythonic interface to DPF's postprocessing toolbox for manipulating and transforming simulation data



PyDPF Composites

Pythonic interface to post-process layered and short-fiber composite models

Erste Schritte mit PyAnsys

Wie wird PyAnsys installiert?

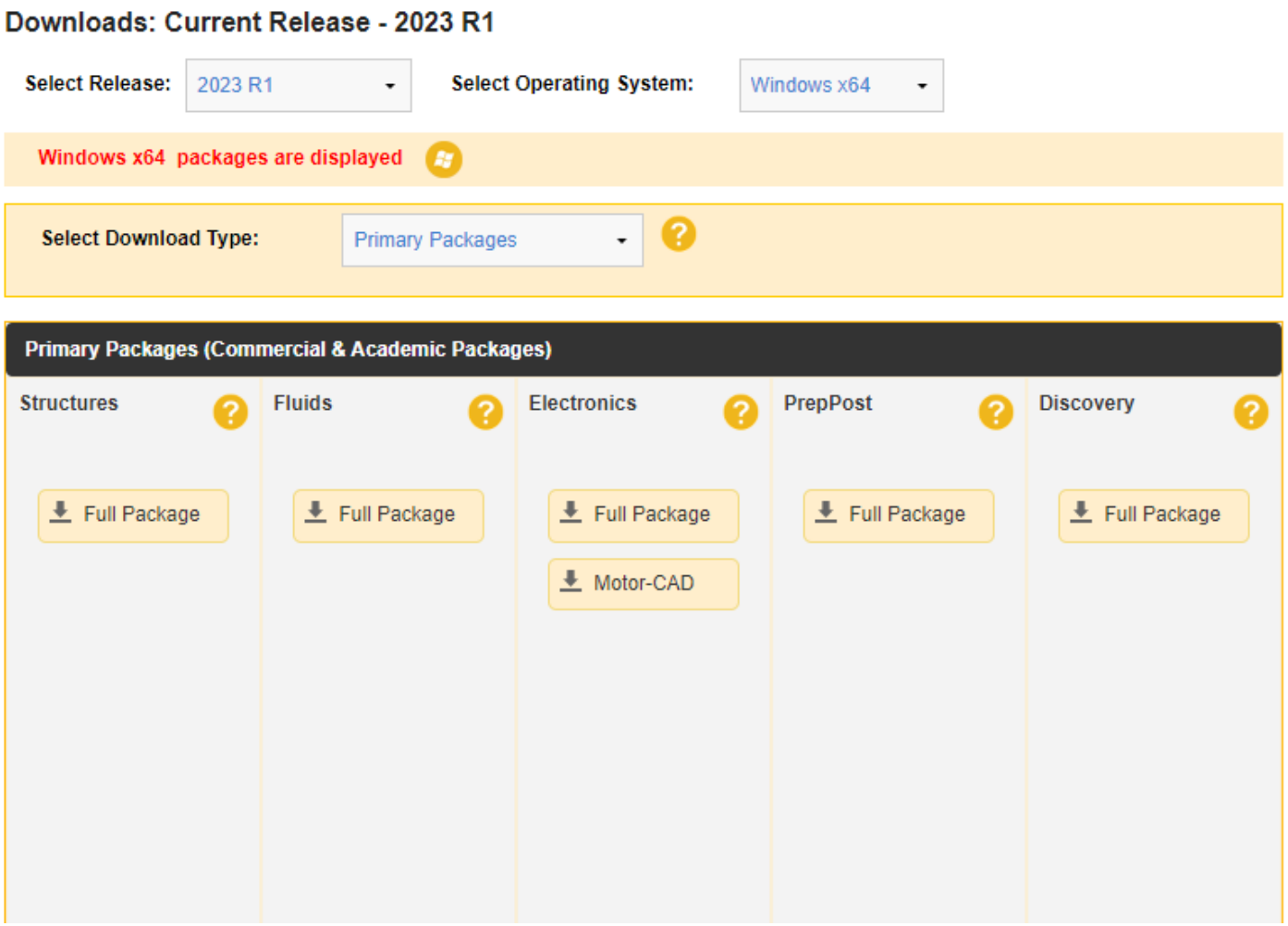
Drei Schritte:

1. Installiere die Ansys Produkte (Mechanical, Fluent, etc.) mit dem regulären Installer
2. Installiere Python und einen Code Editor
3. Installiere die PyAnsys Pakete

Für 2. & 3.: Benutze den Ansys Python Manager

Schritt 1: Installiere Ansys Produkte

Download der Ansys Produkte aus dem Customer Portal



Schritt 2: Installiere Python und einen Code Editor

- Python:
  - Version **3.8** oder neuer benötigt
  - Installation von python.org empfohlen
- Beliebige Code-Editoren können verwendet werden. Zum Beispiel:
  - Spyder: Open Source, Fokus auf wissenschaftliche Anwender
  - PyCharm: Bezahltes Produkt, hat eine gratis «Community Edition»
  - Visual Studio Code: Open Source Editor von Microsoft
  - JupyterLab: Webbasierte interaktive Entwicklungsumgebung

Schritt 3: Installiere die PyAnsys Pakete

- Benutzen von Python «virtual environments» empfohlen
- Python Pakete werden mit dem «pip» Kommando installiert:

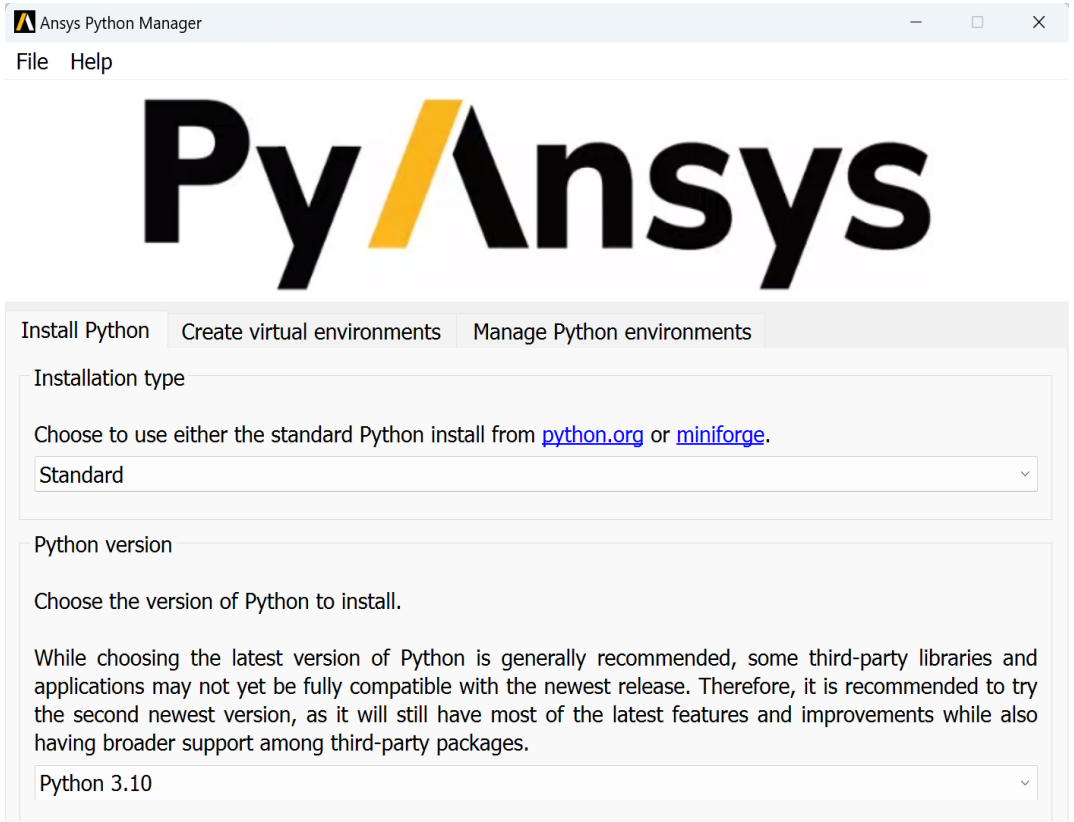
```
# Installiert alle PyAnsys Pakete
pip install pyansys
```

```
# Installiert die Pakete einzeln
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
```

Ansys Python Manager

- Installiert Python, PyAnsys Pakete, und weitere Python Pakete
- Kann “virtual environments” erstellen

https://installer.docs.pyansys.com/



Wie wird PyAnsys gestartet?

Aus einem Python Skript / Notebook / Kommandozeile:
PyAnsys Bibliothek importieren + ausführen

```
>>> 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.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

```
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

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

mechanical = launch_mechanical()
```

```
from ansys.mechanical.core import App

app = App()
ns = app.DataModel.Project.Model.AddNamedSelection()
```

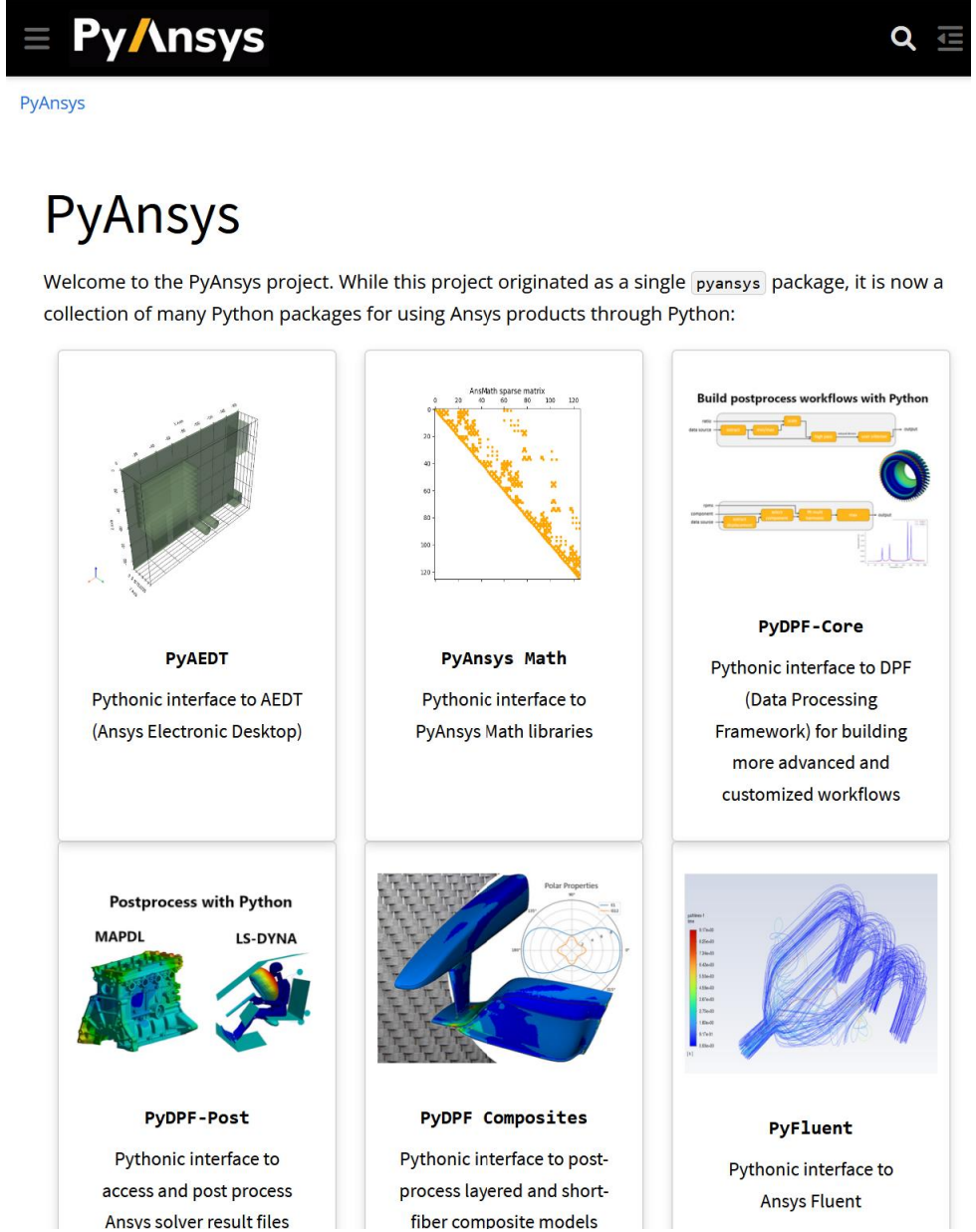
PyMechanical (remote session and embedded instance)

Wo sind Source Code und Dokumentation?
Wie melde ich einen Bug?
Wo finde ich Hilfe?

Ansys

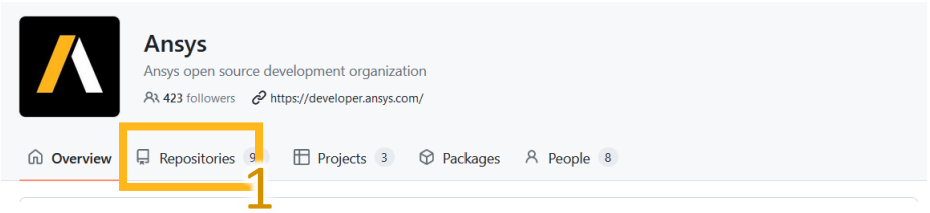
Dokumentation

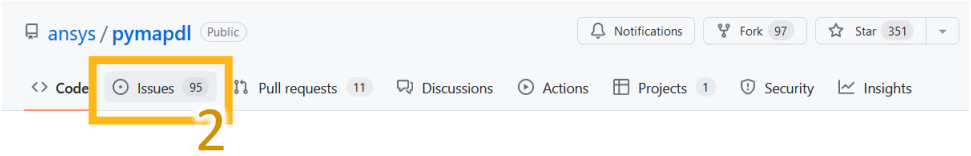
- Frei zugänglich unter **https://docs.pyansys.com**
- Dokumentation + Beispiele für einzelne Pakete:
  - PyMAPDL: https://mapdl.docs.pyansys.com
  - PyFluent: https://fluent.docs.pyansys.com
  - PyDPF Core: https://dpf.docs.pyansys.com
  - PyMechanical: https://mechanical.docs.pyansys.com
  - …

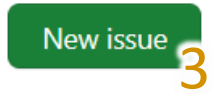


Source Code

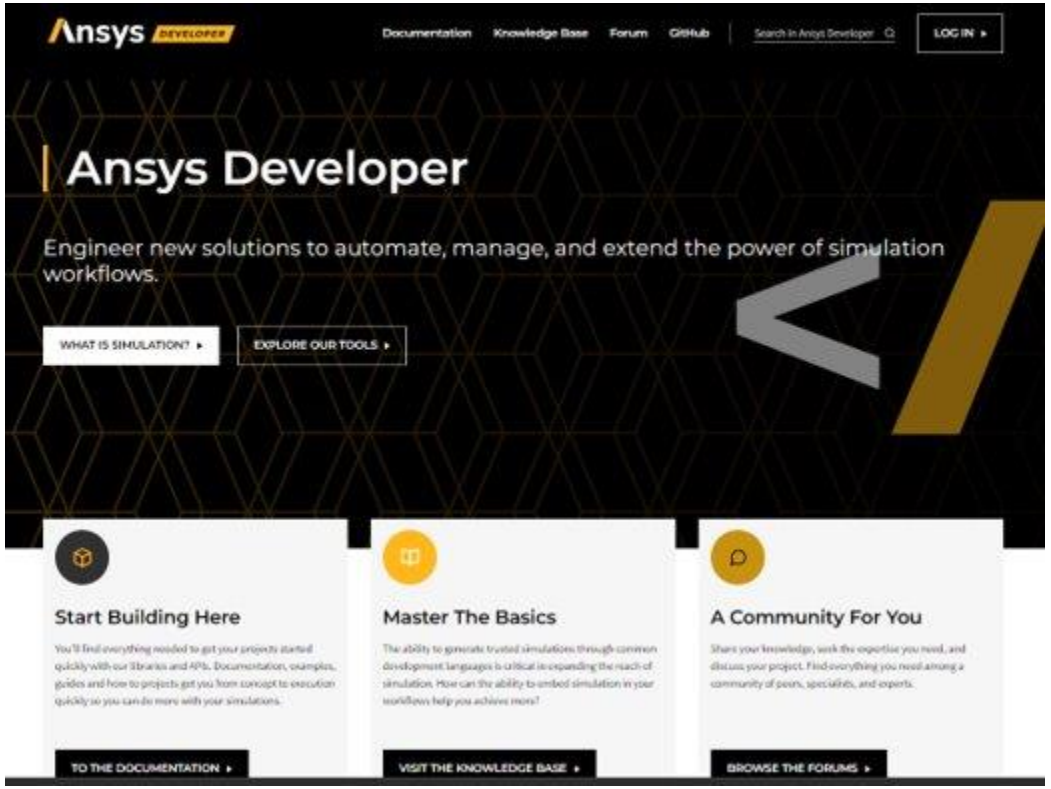
- Verfügbar auf **Github:** https://github.com/ansys
- Einzelne Pakete in verschiedenen **Repositories**
- Benutze **Issues** für:
  - Bugs
  - Feature Requests
  - Fragen an die Entwickler
- Benutze **Pull Requests** für Code - Beiträge

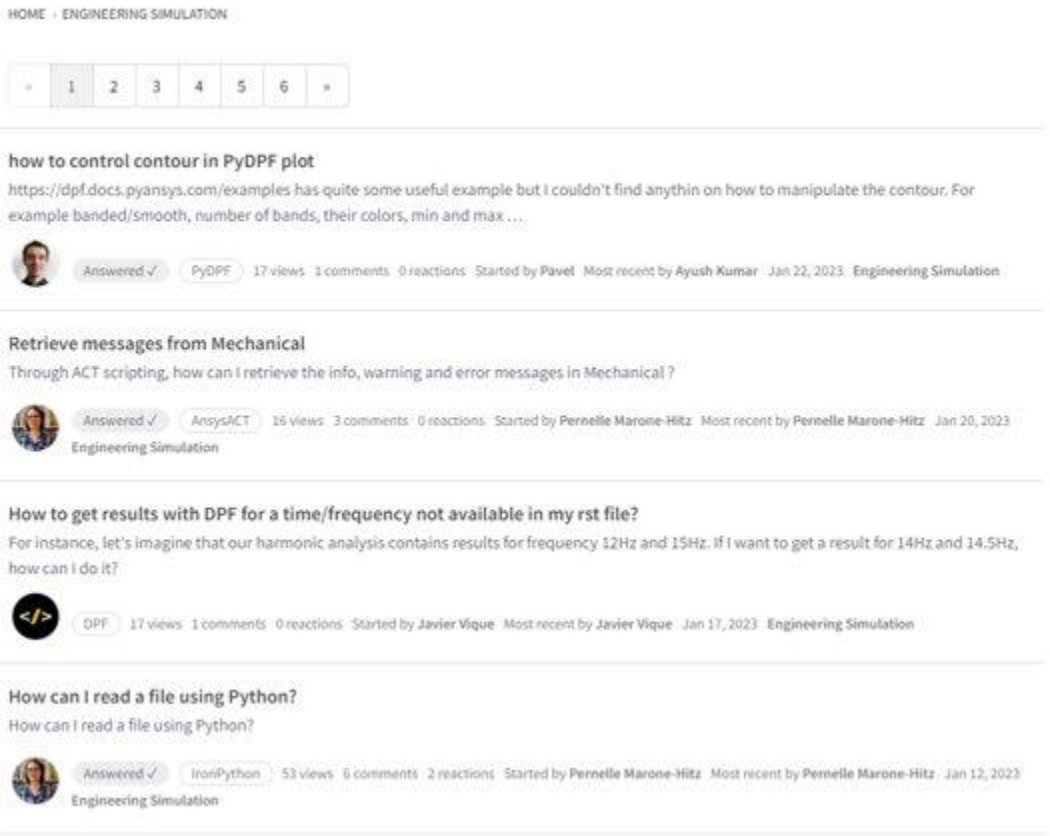






Ansys Developer Portal + Developer Forum





Weitere Informationen, Tipps + Tricks

https://developer.ansys.com/

Austausch mit anderen Kunden / ACE

https://discuss.ansys.com/

Trainingskurse

PyAnsys Kurse sind verfügbar auf Ansys Innovation Space und dem 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 ⟶

Kürzliche Veröffentlichungen von CADFEM

- PyAnsys - Python Interface & Ansys (Webinar)

PYANSYS - PYTHON-SCHNITTSTELLE ZUR INTERAKTION MIT ANSYS-PRODUKTEN

Erleben Sie, wie Sie die Möglichkeiten des Scriptings in Ansys mit PyAnsys nutzen können.

Webinar | On Demand | Deutsch

LINK ANFORDERN

- Ansys 2023 R1: Automatisierung und Digitalisierung im Engineering (Webinar)

ANSYS 2023 R1 – CADFEM TIPPS - AUTOMATISIERUNG UND DIGITALISIERUNG IM ENGINEERING

Lösungen aus dem neuesten Ansys Release für gesteigerte Effizienz und Nachvollziehbarkeit in der Produktentwicklung

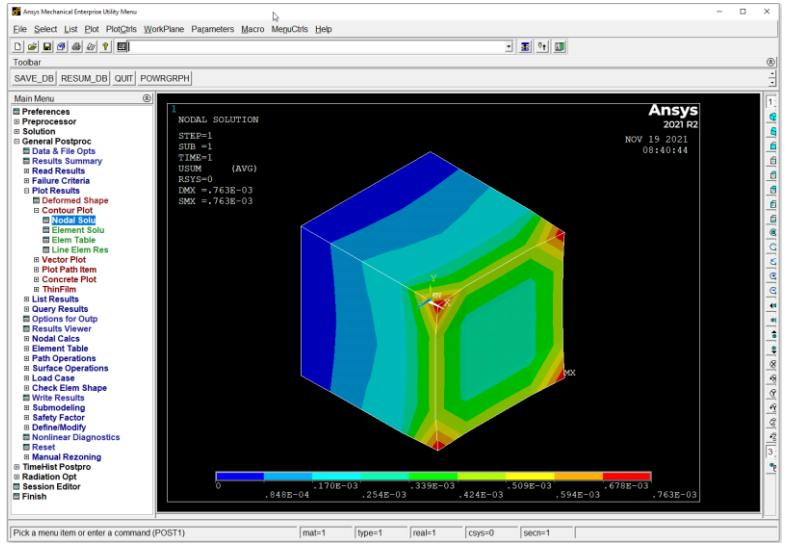
Webinar | On Demand | Deutsch

AUFZEICHNUNG ANFORDERN

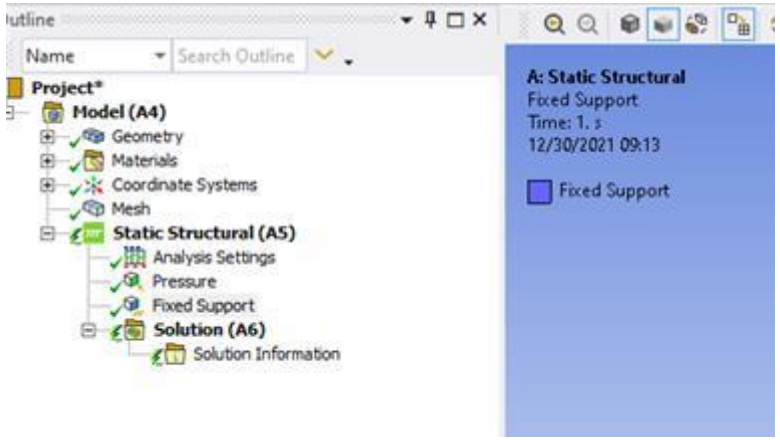
PyMAPDL

MAPDL = Mechanical Ansys Parametric Design Language

Scripting-Sprache des Ansys Mechanical Solvers



Ansys Classic

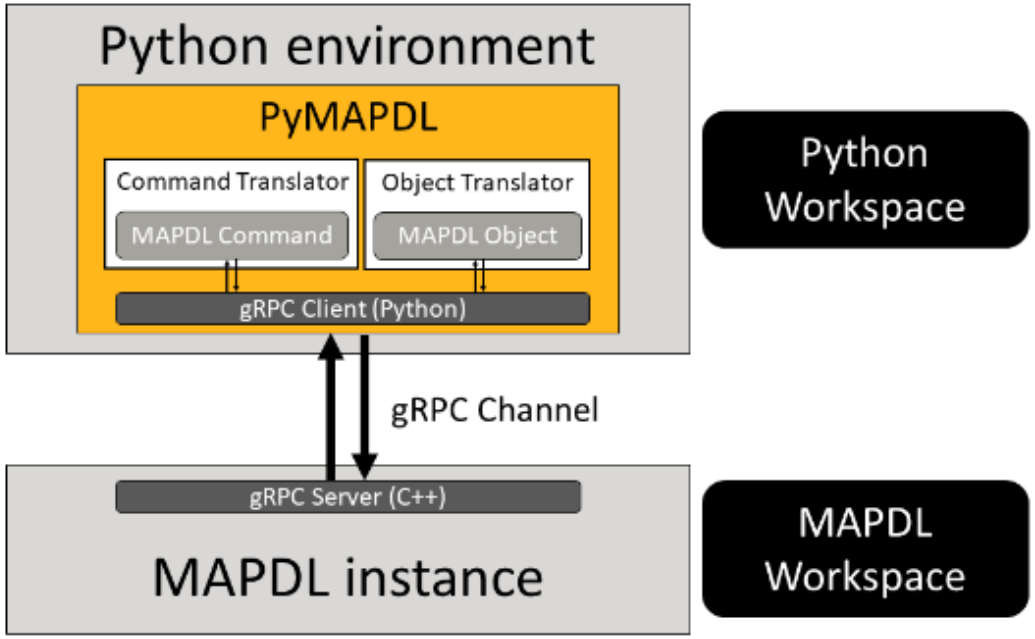


Mechanical

```
/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
```

Benutzen beide MAPDL

Von MAPDL zu PyMAPDL



PyMAPDL architecture diagram

- Direkte Übersetzung der MAPDL Kommandos + Objekte
- Moderne Programmiersprache:
  - `for i in data` anstatt `*do/*enddo`
  - `mapdl.parameters["MY_ARRAY"]` anstatt `*get`
- Kann mit anderen Python – Paketen kombiniert werden

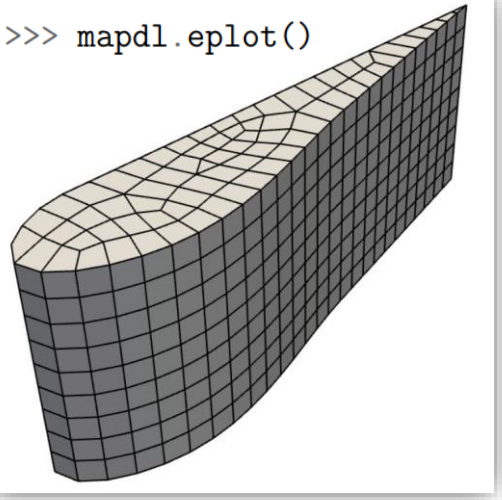
PyMAPDL

- PyMAPDL kann:
  - Setup von Geometrie, Mesh, Modell
  - Interaktives Plotting
  - Postprocessing
  - MAPDL Skipte zu PyMAPDL konvertieren

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
```

```
>>> mapdl.eplot()
```



- MAPDL Kommandos entweder direct mit “run" ausführen, oder als Python Funktionen

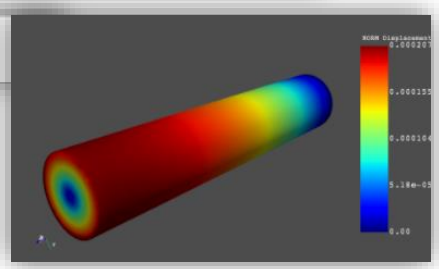
```
mapdl.run('/PREP7')
mapdl.run('K, 1, 0, 0, 0')
mapdl.run('K, 2, 1, 0, 0')
mapdl.run('K, 3, 1, 1, 0')
mapdl.run('K, 4, 0, 1, 0')
mapdl.run('L, 1, 2')
mapdl.run('L, 2, 3')
mapdl.run('L, 3, 4')
mapdl.run('L, 4, 1')
mapdl.run('AL, 1, 2, 3, 4')
```

```
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)
```

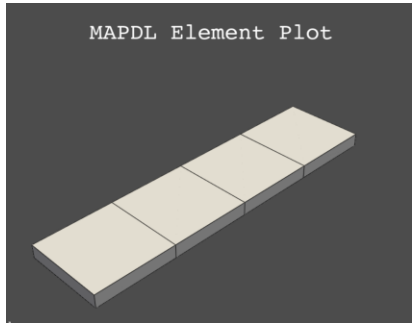
```
POST1:
PRNSOL, U, X

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
```

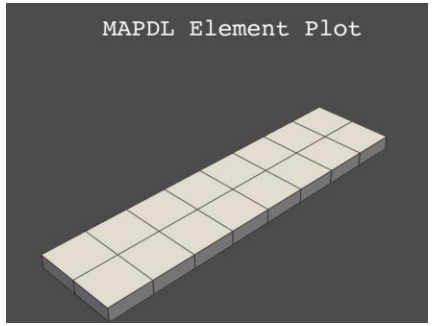
```
>>> mapdl.set(1, 1)
>>> disp_x = mapdl.post_processing.nodal_displacement('X')
array([1.07512979e-04, 8.59137773e-05, 5.70690047e-05, ...,
       5.70333124e-05, 8.58600402e-05, 1.07445726e-04])
```

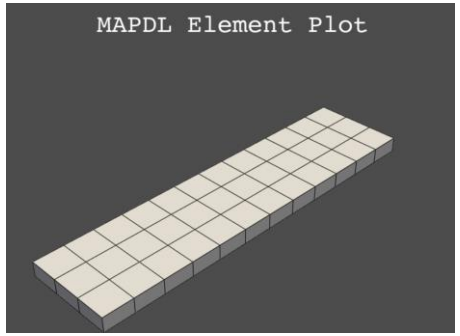
```
>>> mapdl.post_processing.plot_nodal_displacement('X')
```

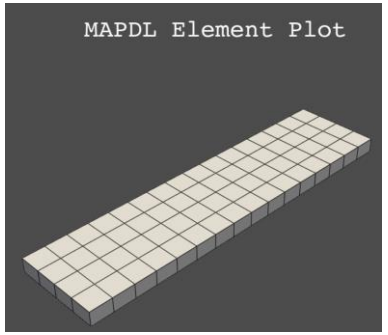


PyMAPDL Beispiel: Konvergenzanalyse

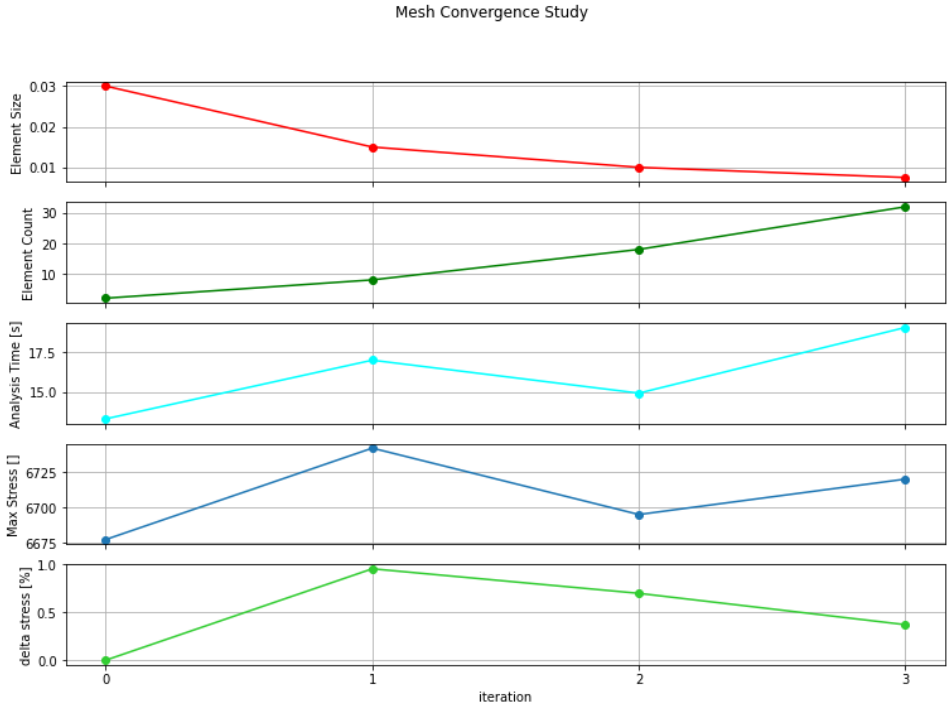


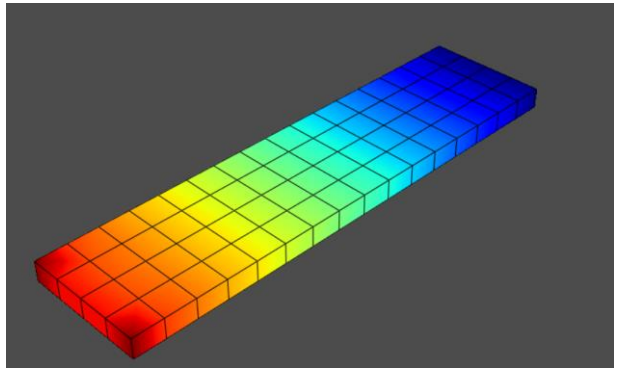







```
Plots will open during the analysis.  Close plots to proceed!
my current element size is 0.03
Close plot to proceed!
Maximum von Mises stress with element size 0.03 is 6677.26
Time for the analysis = 2.0963 seconds.
My delta Stress = 98.50238034783781
my current element size is 0.015
Close plot to proceed!
Maximum von Mises stress with element size 0.015 is 6741.6
Time for the analysis = 1.3067 seconds.
My delta Stress = 0.954292355950986
my current element size is 0.01
Close plot to proceed!
Maximum von Mises stress with element size 0.01 is 6694.87
Time for the analysis = 1.2089 seconds.
My delta Stress = 0.6980179883995382
my current element size is 0.0075
Close plot to proceed!
Maximum von Mises stress with element size 0.0075 is 6719.78
Time for the analysis = 1.2511 seconds.
My delta Stress = 0.3708004504676314
Mesh size shows convergence based on allowable delta.
```





PyFluent

PyFluent

PyFluent ist in 3 Pakete aufgeteilt:

- ansys-fluent-core (https://github.com/ansys/pyfluent)
  - Fluent Meshing, Solver und Postprocessing
- ansys-fluent-parametric (https://github.com/ansys/pyfluent-parametric)
  - Fluent parametrisierte Workflows
- ansys-fluent-visualization (https://github.com/ansys/pyfluent-visualization)
  - Postprocessing und Visualisierung mit PyVista and Matplotlib

PyFluent Core

Zugriff auf die Fluent TUI Hierarchie:

```
solver.tui.define.models.unsteady_second_order("yes")
solver.tui.define.models.energy(viscous_dissipation=True,
                                pressure_work=False, …)
```

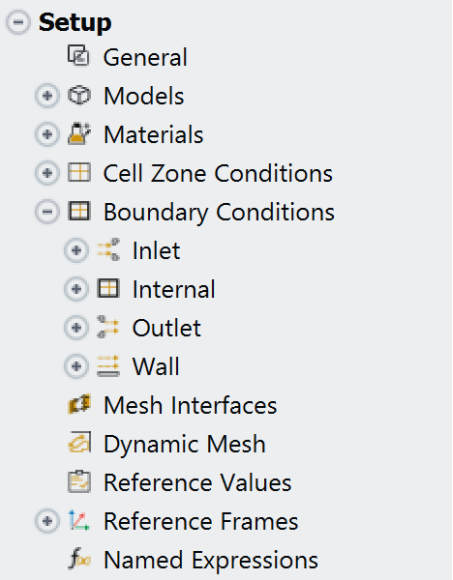
- Komplette TUI Funktionalität ist verfügbar

```
/define/models>
ablation?
acoustics/
addon-module
battery-model
cht/
crevice-model?
dpm/
energy?
eulerian-wallfilm/
heat-exchanger/
multiphase/
potential-and-li-ion-battery?
```

…und Solver Settings Objects:

```
pyfluent.setup.models.energy.enabled = True
pyfluent.setup.boundary_conditions.velocity_inlet['inlet2'].vmag = {
    'option' : 'constant or expression',
    'constant' : 1.2 }
```

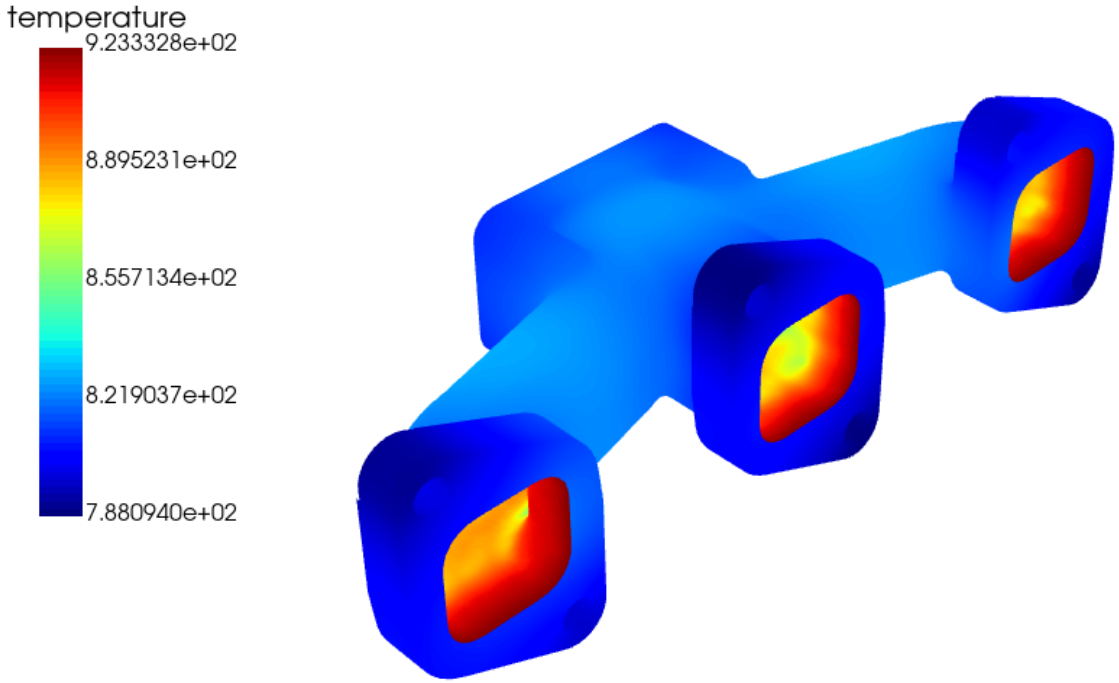
- In aktiver Entwicklung
- Konsitent mit der Fluent Outline View



Beispiel: PyFluent Visualization

- Visualisierung mit PyVista: Open Source 3D Visualisierung auf der Basis von VTK

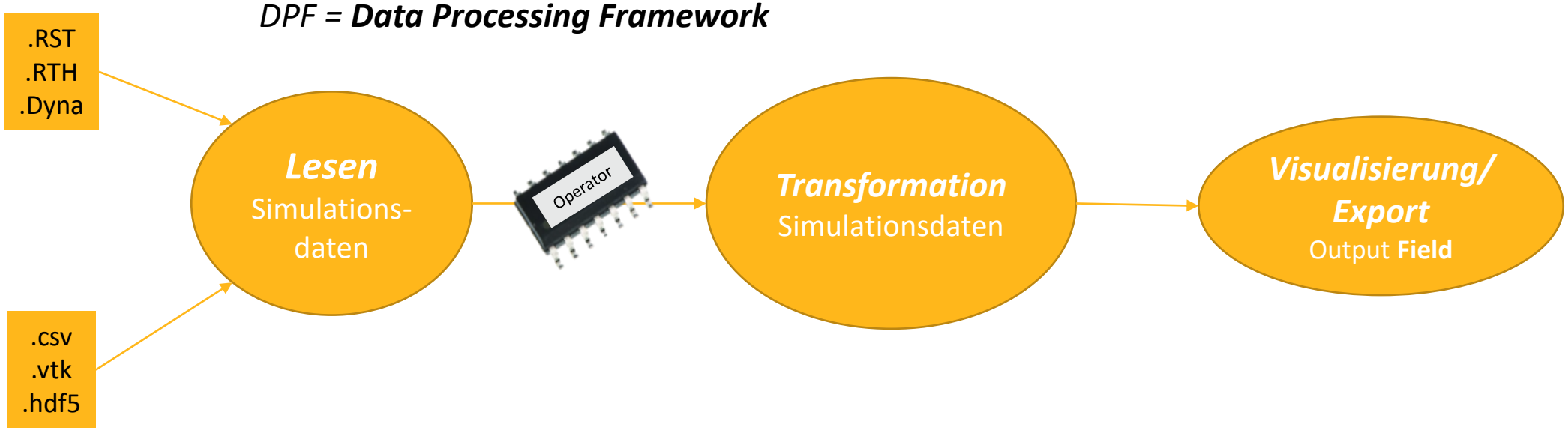
```
from ansys.fluent.visualization.pyvista import Graphics
session = pyfluent.launch_fluent(precision="double", processor_count=2)
graphics = Graphics(session=session)

temperature_contour =
graphics.Contours["contour-temperature-manifold"]
temperature_contour.field = "temperature"
temperature_contour.surfaces_list = [
    "in1",
    "in2",
    "in3",
    "out1",
    "solid_up:1",
    "solid_up:1:830",
]
temperature_contour.display("window-5")
```



PyDPF

Was ist DPF?

*DPF = **Data Processing Framework***

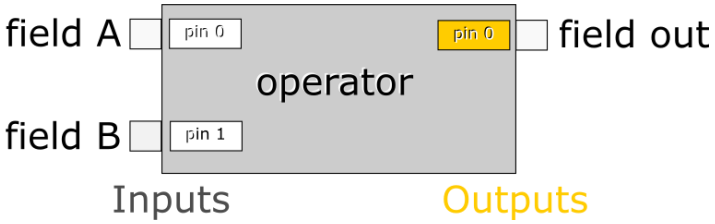


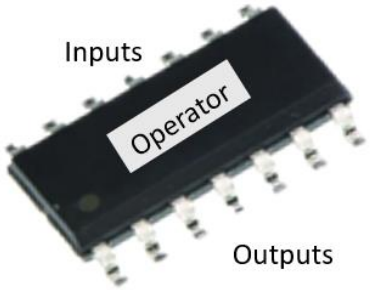
Postprocessing Framework:

- Physik – agnostisch
- Remote / verteilter Zugriff
- Erweiterbar

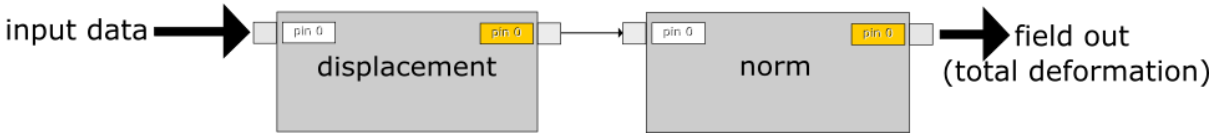
DPF Datenmodell – Operator

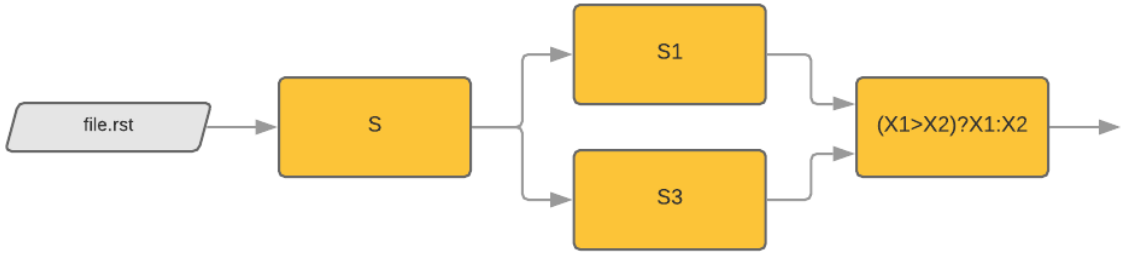
- Operatoren können Daten lesen / schreiben / transformieren





- Komplexe Workflows entstehen durch Zusammenfügen von Operatoren



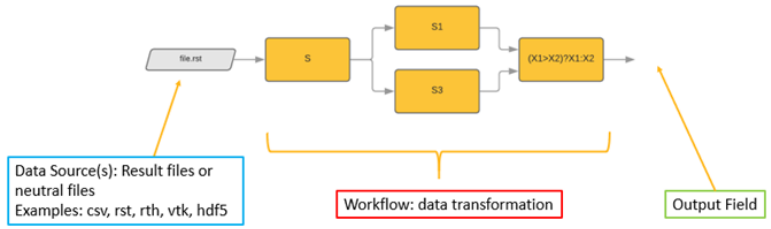


DPF in PyAnsys

3 Pakete basierend auf DPF in PyAnsys

DPF Core

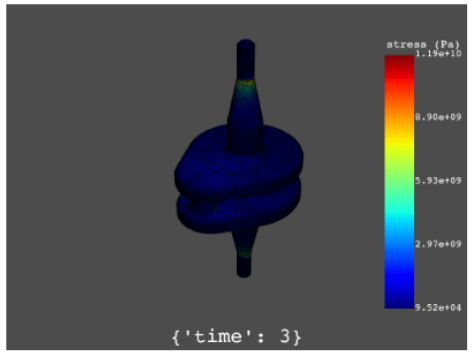
Lesen und Manipulation von FE Daten mit skalierbaren Operatoren



DPF Post

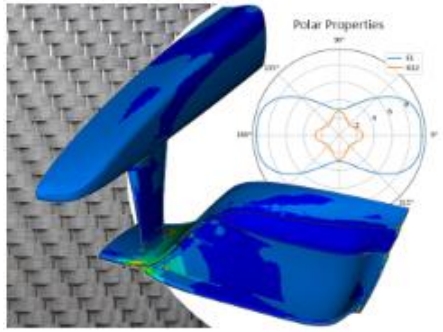
Direkte post-processing API

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



DPF Composites

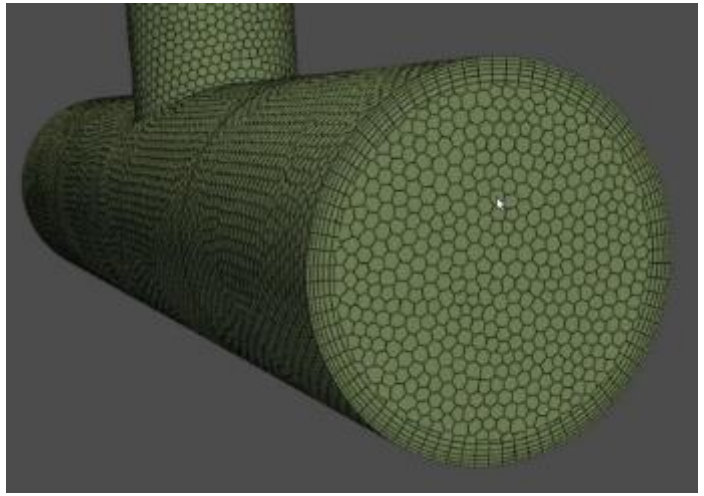
Post-processing von Kurzfaser- und geschichteten Verbundwerkstoffen



PyPrimeMesh

PyPrimeMesh

- Ansys Meshing Technologie verfügbar in Python
- Möglichkeiten:
  - Erstellen von Oberflächen- und Volumennetzen
  - Grössensteuerung anwenden, um die Netzverteilung zu kontrollieren
  - Komplexe Workflows / Automatisierung
  - Extrahieren und Vernetzung von Regionen mittels Wrapping - Methoden
  - Konnektivität der Topologie und des Netzes ändern
  - Import von CAD
  - Export zu verschiedenen Ansys Solvern



Zusammenfassung

- Ansys Produkte + Technologien verfügbar in Python
- Kann mit anderen Bibliotheken verknüpft werden
- Open Source: direkte Interaktion mit Entwicklern über Github

Ansys