Univerza v Ljubljani
Fakulteta za strojništvo

EUROfusion

VIVID-DTE

Verification-oriented Interactive VISualisation and Decision support for the Digital Twin Environment

Presentation of Graphical User Interface with IMAS workflows with visualisation actors and tools

Leon Kos, University of Ljubljana, LeCAD laboratory

LeCAD

iter

china eu india japan korea russia usa



Contents

- VIVID-DTE framework objectives and Introduction
- Field line tracing and PFC power deposition with SMITER demo
- ParaVIS module with TIARA and IMAS plugins demo
- SOLPS-ITER GUI - Edge profiles and Adjoint Optimization visualization demo
- Workflows with Computational and Visualization Actors
- Workflow Designer, Uncertainty Quantification Actor generator and Workflow demo
- Pulse Design Simulator Workflow demo

VIVID-DTE framework objectives

Build user-centred visualisation stack for DTE supporting rapid V&V and decision support.

- O1. Unified visual analytics (iWrap/Muscle3-linked 3D/temporal + IMAS-ParaView)
- O2. Workflow Designer with no-code workflow interface
- O3. Operator decision-support dashboards
- O4. Robust V&V toolchain with overlays & provenance
- O5. FAIR/IMAS-aligned interoperability

Univerza v Ljubljani
Fakulteta za strojništvo



VIVID-DTE at glance

- Single executable GUI framework running on Enterprise Linux 9.x and compatible distributions workstations or cluster virtual desktop login nodes.
- GUI is based on SALOME framework with additional modules for IMAS, SMITER PFC, SOLPS-ITER, TIARA, PDS-WF (NOVA, DINA, Scenplint, Transmak) and many Python actors within GUI workflows, workflow designer, actor generator and studies
  Overall, it consists of 160+ C++ and 100+ Python libraries
- Distributed as a compressed AppImage binary. Can be installed locally with:
  - wget http://static.iter.org/imas/assets/smiter/appimage/SMITER-1.6.4-glibc2.34-x86_64.AppImage # around 5.2GB of compressed squashfs filesystem
  - chmod +x SMITER-*.AppImage # is an executable
  - ./SMITER-1.6.4-glibc2.34-x86_64.AppImage # ldd --version
- Documentation available online
  https://static.iter.org/imas/assets/smiter/html/howto/appimage.html



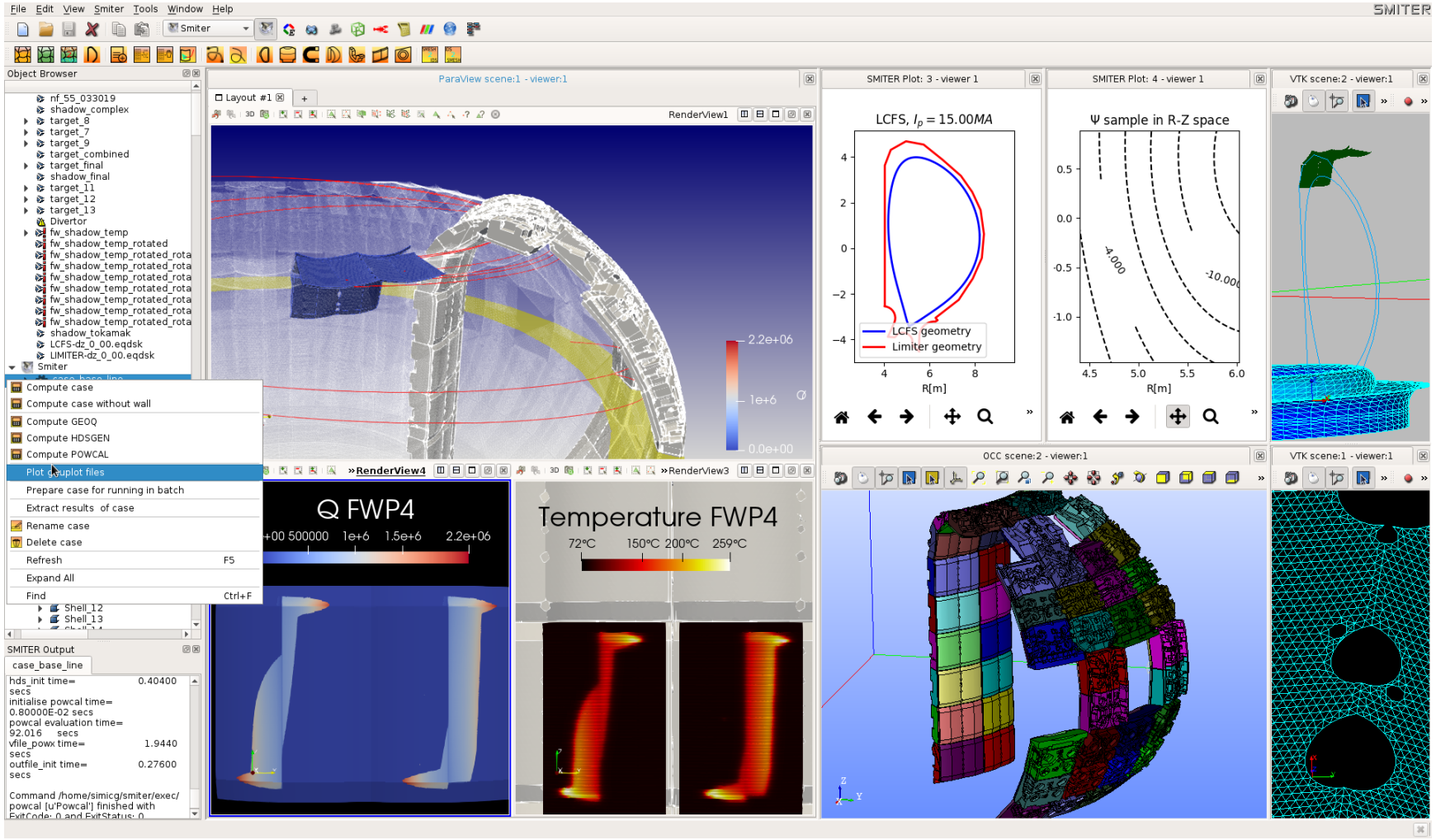
VIVID-DTE is SALOME based tool

- SALOME (https://www.salome-platform.org) is a 3D GUI with many pre/post processing tools (modules) ranging from CAD, meshing to visualisation and UQ tools. SALOME is time consuming to compile from 250+ libraries into a single GUI.
- Includes Parametric (Shaper) and CSG (GEOM) CAD modules
- Meshing with SMESH includes GMSH, NetGen, MeshGems,…
- ParaVIS (ParaView) module now includes IMAS and TIARA
- Several other modules for Data assimilation (ADAO), Job management, semantic analysis (Eficas), Uncertainty Quantification library (OpenTURNS) with Persalys GUI, Field operation and presentation (Fields)

Univerza v Ljubljani
Fakulteta za strojništvo

SMITER for Plasma Facing Components

SMITER is a GUI module (solver) for field line tracing and PFC power deposition used at ITER. Includes SMARDDA and L2G PFC libraries. ELMER FEM for thermal model and RAYSECT, CHERAB, for synthetic diagnostic.

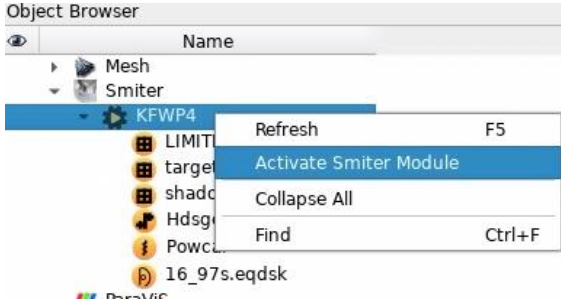


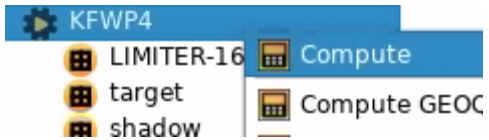
Univerza v Ljubljani
Fakulteta za strojništvo

ITER NF55* benchmark case demo

1. Start and load the NF55 case and prepare CAD STEP data with TMPDIR=/tmp/$USER ./SMITER-1.6.4-glibc2.34-x86_64.AppImage withsession ${HOME}/study/iter-nf55/nf_55_033019_2015.hdf & make -C ${HOME}/study/step
2. In *Object Browser expand Smiter and* right click on KFPW4 case and Activate Smiter module
3. Use File > Preferences > Smiter > MPI Options and check if MPI and Internal MPI are selexte
4. Right click on KFPW4 case again and select Compute
5. ParaVIS module will open and show result after a minute. In Pipeline browser click on the eye of KPFW4_2026 case

For more Help select Help > SMITER GUI Documentation



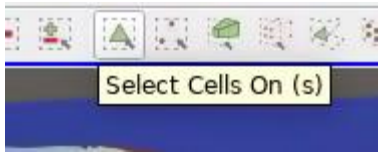


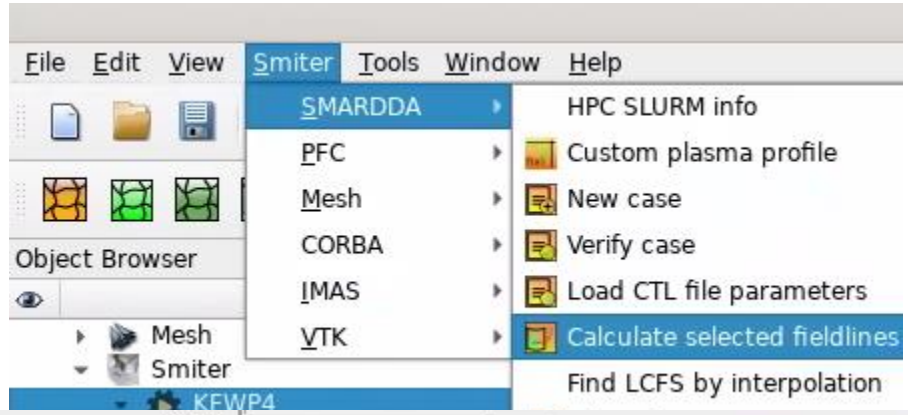


* Martin Kocan et al., Nucl. Fusion. 55 (2015) 033019 (16pp), doi:10.1088/0029-5515/55/3/033019

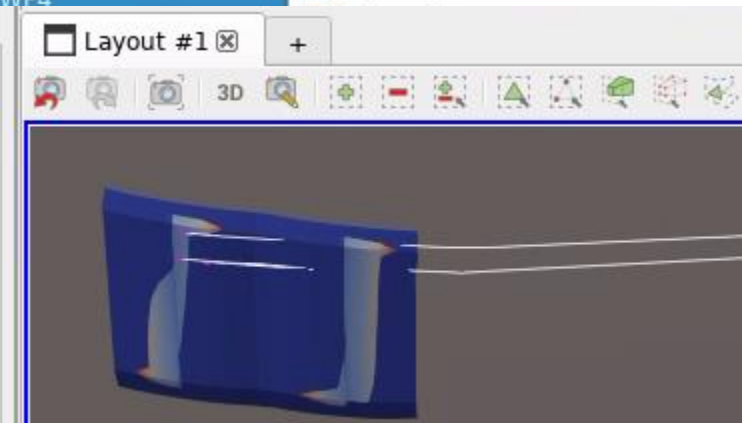
Fieldline tracing visualisation

1. Select Smiter module again while looking at the panel in ParaVIS
2. In Paravis pres key “s” and click on one triangle highlight
3. Press “s” again and with Ctrl-click add another triangle…
4. In Object Browser select case KFPW4
5. Smiter > SMARDDA >
   > Calculate selected fieldlines
6. Select ParaVIS and explore fieldlines using RKF4 integration

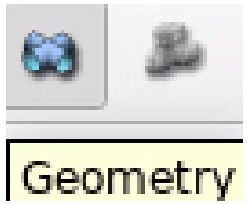




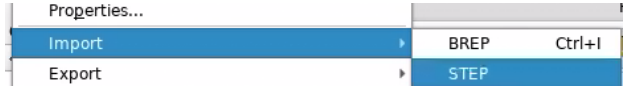




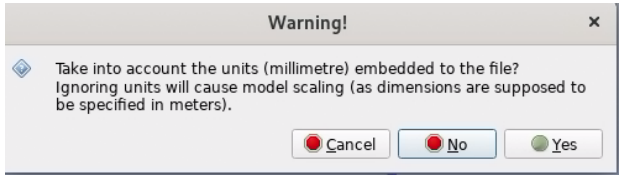
PFC CAD Geometry



1. Select Geometry module and File > Import > STEP



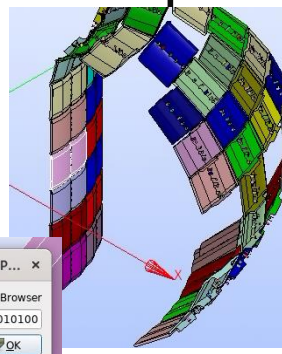
2. Import STEP ~/study/step/ ITER-inner_wall_surfaces_sides_V2.stp
3. Click No to take into account millimeter units



4. Click on Mouse and Fit All to see PFC



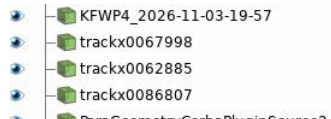
5. Select *tele0F94CB71zyae* compound in Object Browser and in New Entity menu select Explode to Shell Sub-shape type that will result in 435 sub-shapes after clicking Apply and Close and Explode



6. Select *tele0F94CB71zyae* compound again and right-click Auto Color
7. Select *first wall panel 4 (pink) and right-click Show Only*
8. *Select Smiter module* > CORBA > Show GEOM/SMESH and click on the panel again and OK then Cancel



9. See ParaVIS for CAD geometry

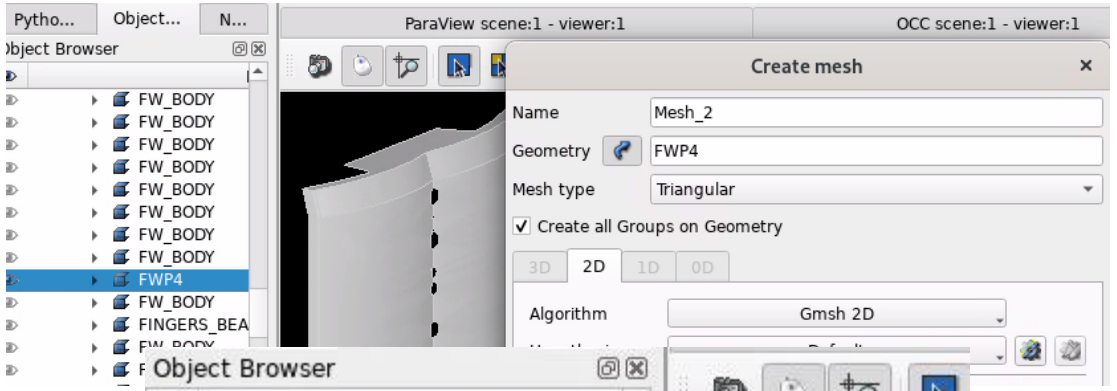


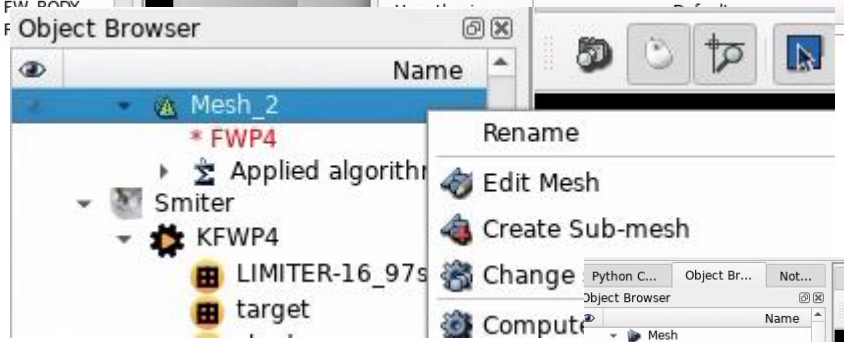


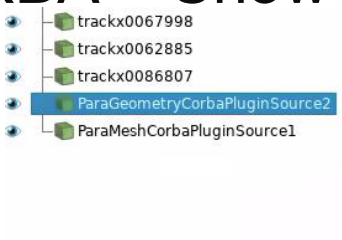
Meshing First Wall Panel

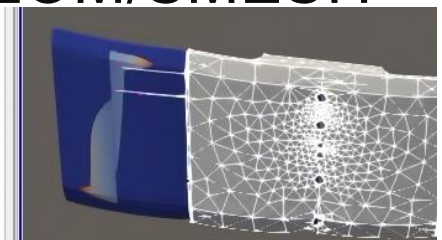


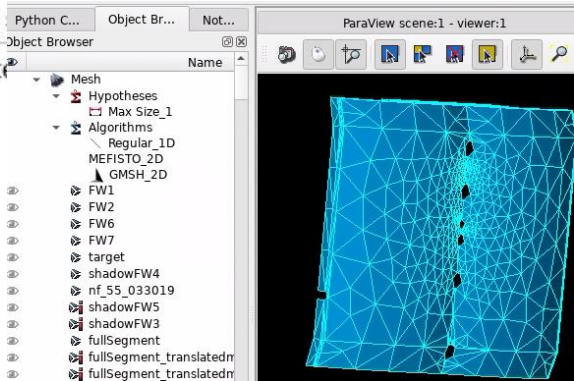
1. In Geometry select FWP4 and Rename it for easier ID
2. Select Mesh Module and Select FWP4 from Geometry
3. Mesh > Create Mesh and
   Gmsh 2D, Triangular
   Apply and Close
4. Expand Mesh in Object Browser and select Mesh_2
5. Right click on Mesh_2 and Compute
6. Hide FWP4 from geometry to see only mesh
7. Use *Smiter module* > CORBA > Show GEOM/SMESH to show Mesh in ParaVIS





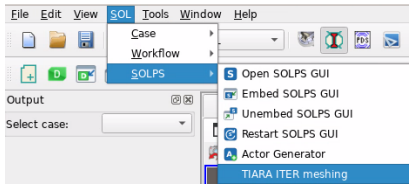




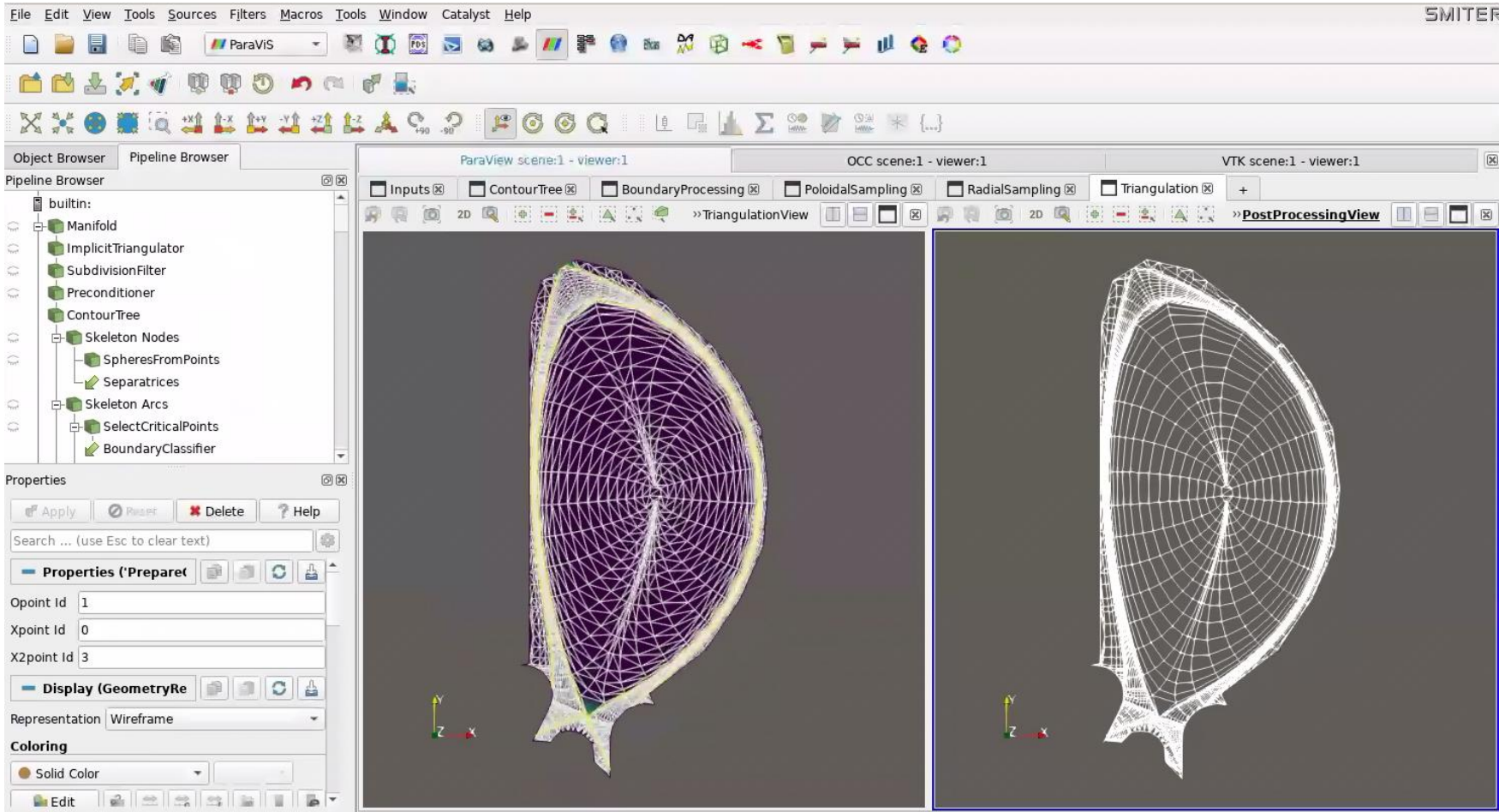


TIARA mesher – ITER case

1. Select module SOL > SOLPS > TIARA ITER meshing



See https://user.iter.org/?uid=89WXN5 for a walkthrough

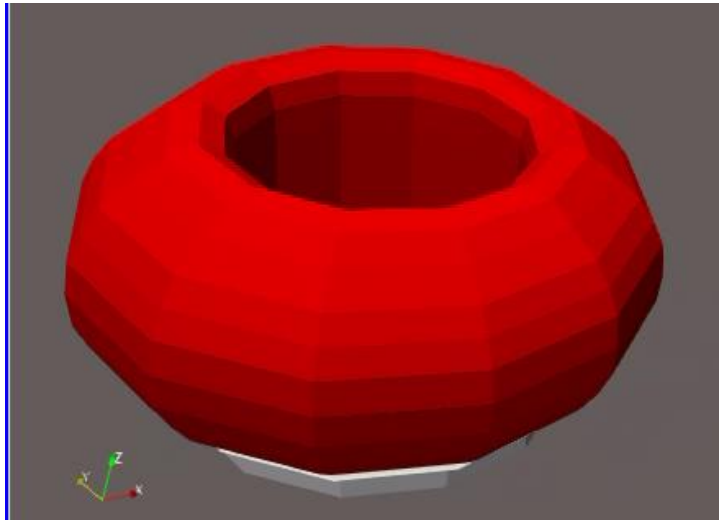


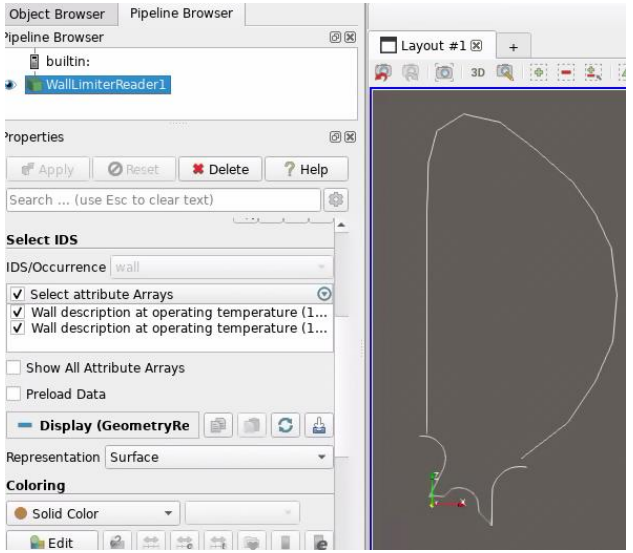

```
SOURCES > TIARA > example.py
1  from tiara_pv import *
2  m = MeshGenerator()
3  load_ITER(m)
4  m.run()
```

IMAS-Paraview - Machine Description

- See https://imas-paraview.readthedocs.io/en/stable/training/training_md.html

1. ParaVIS > Sources > IMAS Tools > Wall Limiter Reader
2. Enter URI (below) and Apply, Select Attribute Arrays and imas:hdf5?path=/work/imas/shared/imasdb/ITER_MACHINE_DESCRIPTION/3/116000/5
3. Click on +Y axis and Zoom All
4. Filters > Miscllanelus > Rotational Extrusion > Apply

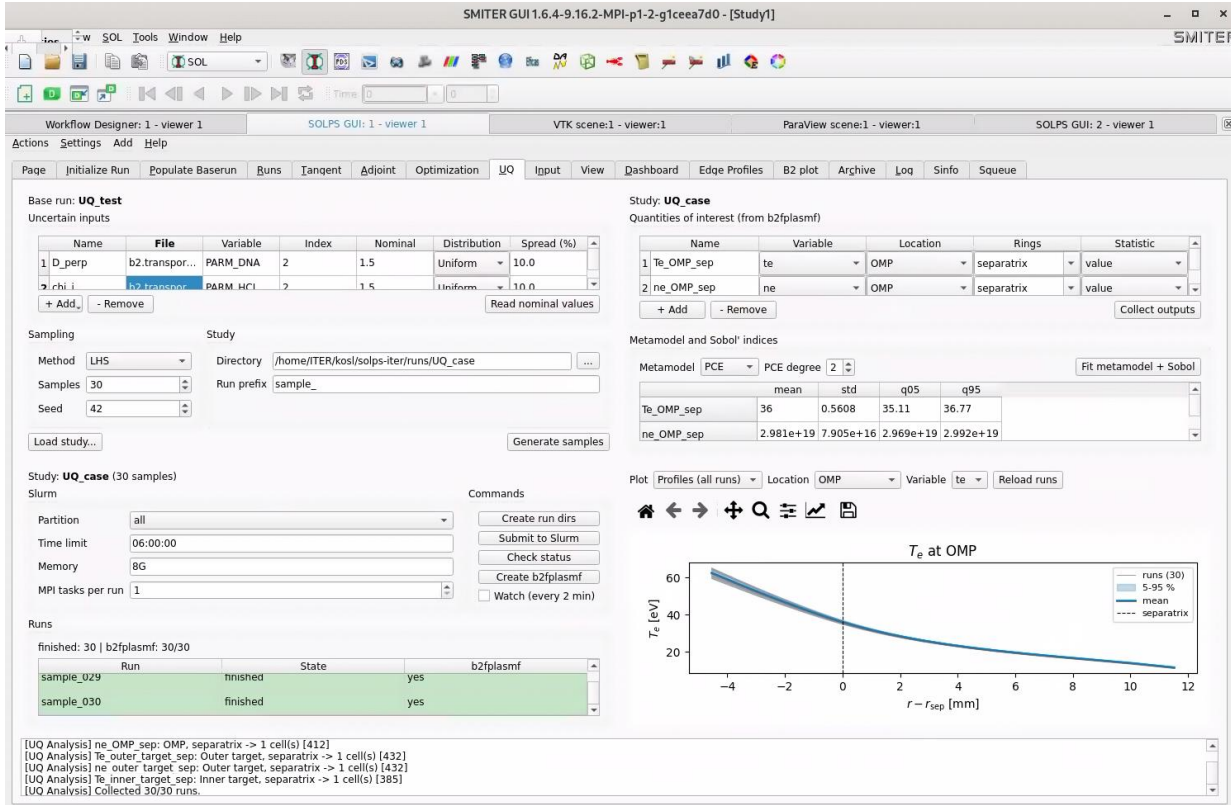




Univerza v Ljubljani
Fakulteta za strojništvo

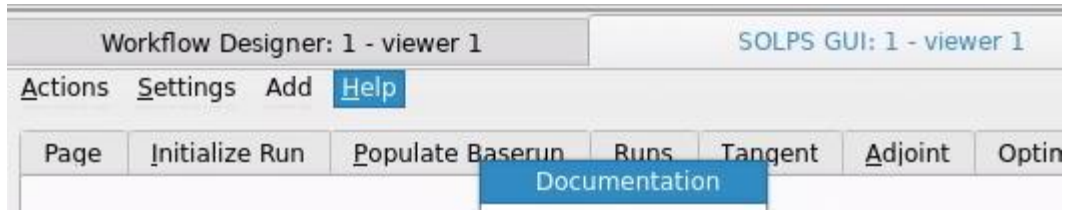
SOLPS Dashboard for robust design

- For automated design and Uncertainty Quantification(UQ) processing we integrated SOLPS-ITER GUI and Workflow Designer inside VIVID-DTE framework, that can provide
  - Workflow design with ”actors” and Actor Generator
  - IMAS coupling
  - Reproducible single study runs
  - Coupling input/output with other “solvers”
  - SOLPS input edit
  - UQ analysis



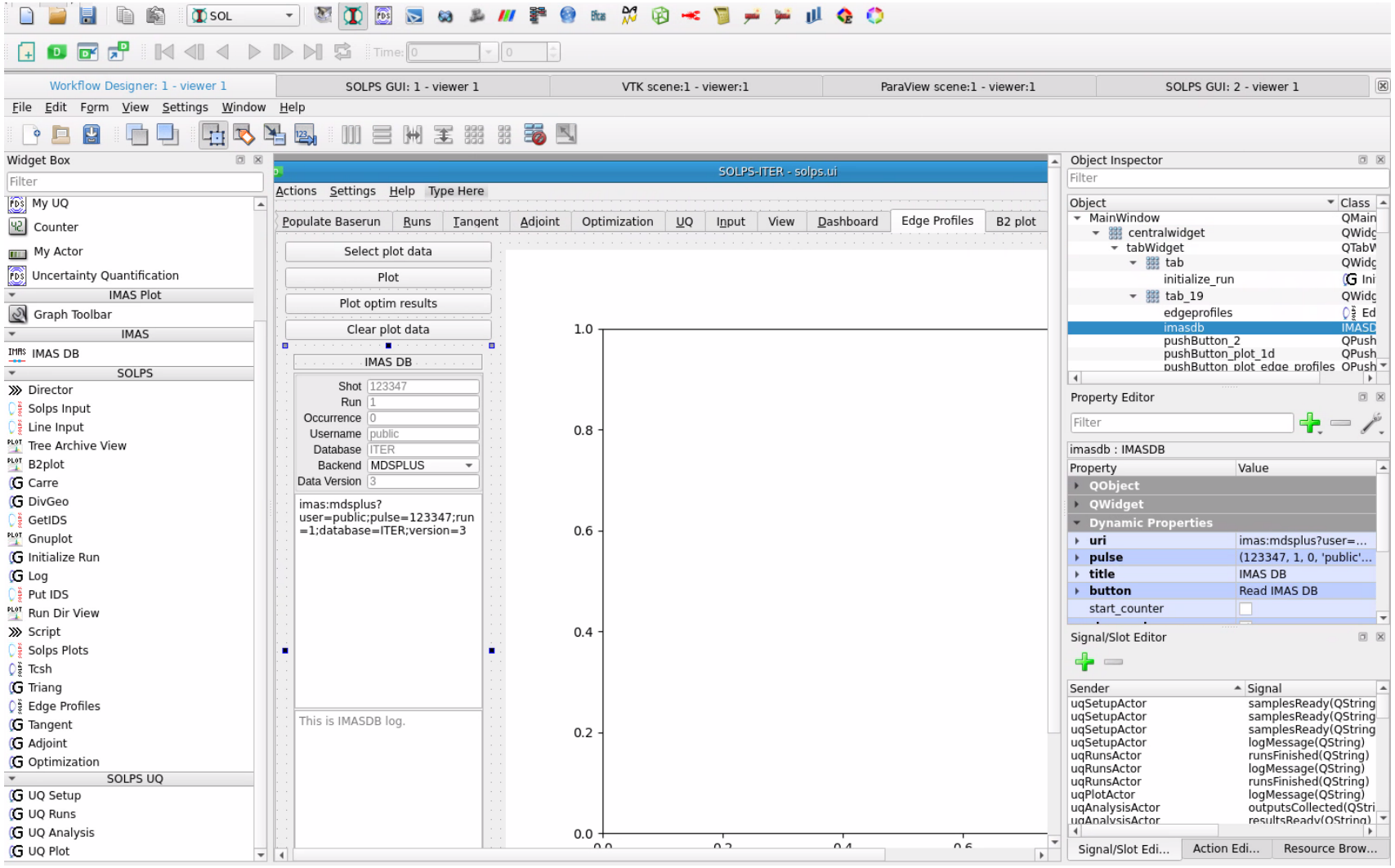
SOLPS-GUI

- SOLPS-GUI is a modular Pyside6-designer based GUI that is built on signal-slot “workflow” principles.
- Besides simple triggers of actions by buttons more complex workflows are possible with passing IDS structures between custom widgets (actors).
- Automation by chaining can be used to create workflows for pre and post processing, including visualisation and steering of runs.
- Help

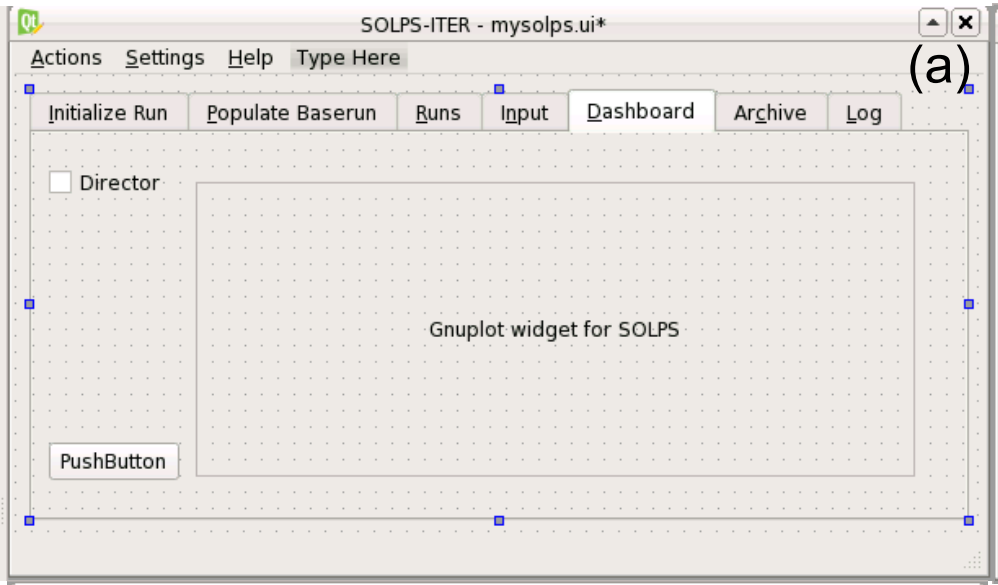


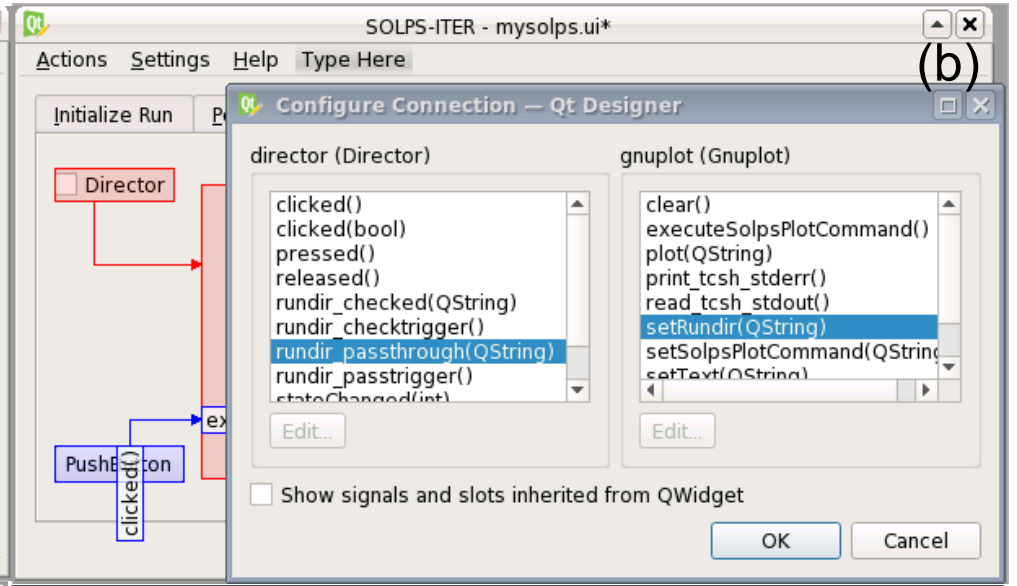
https://static.iter.org/imas/assets/solps-iter/html/gui/index.html

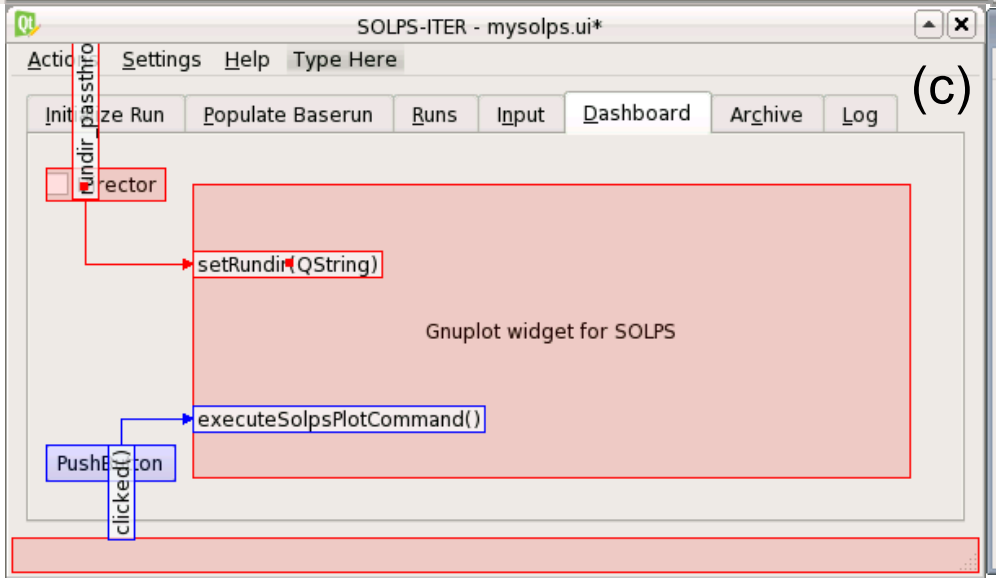
Modular GUI designer for Workflows

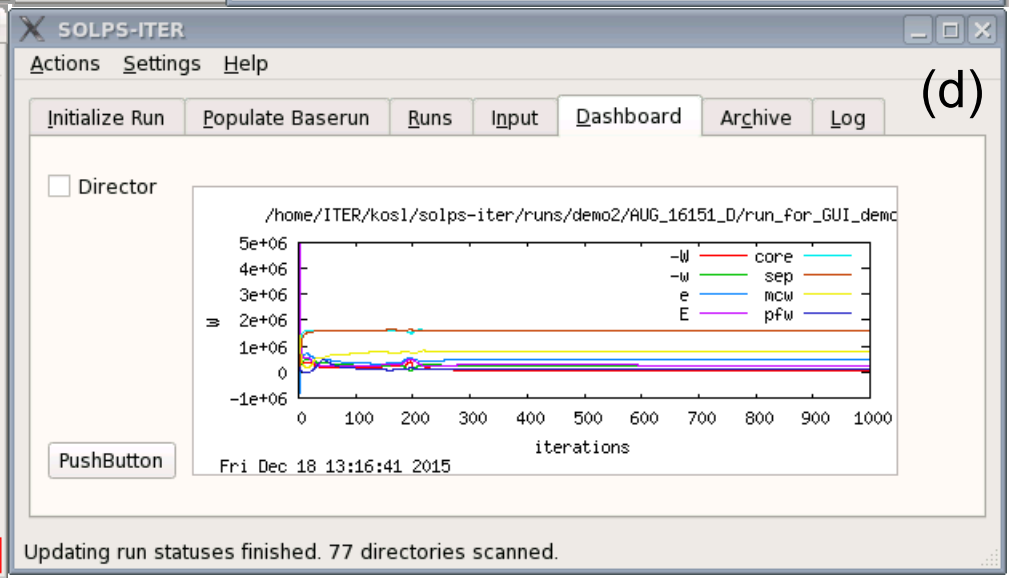


Designer Actors (PySide6 custom widgets)









a) Setup of the actors

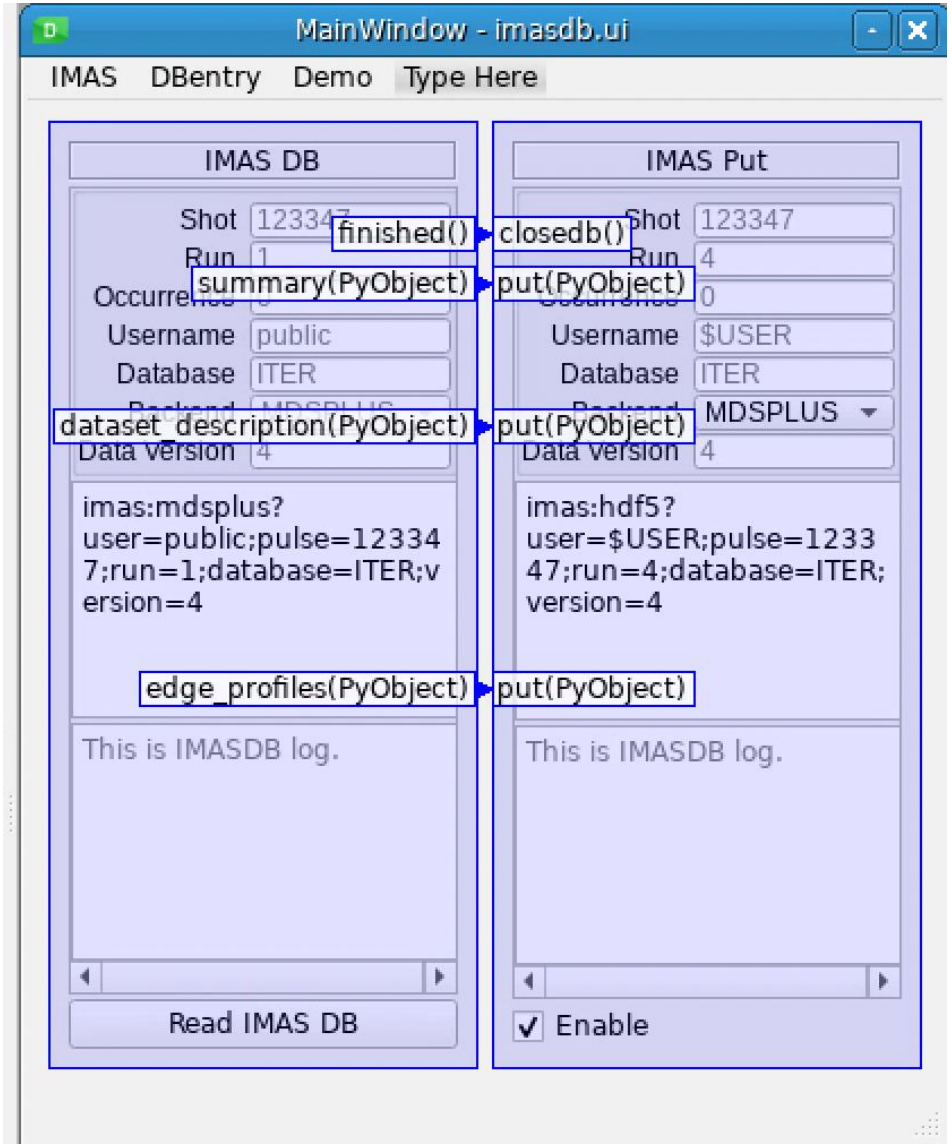
b) Director triggers the property setup

c) Signal-Slot Editor

d) Run of the plot

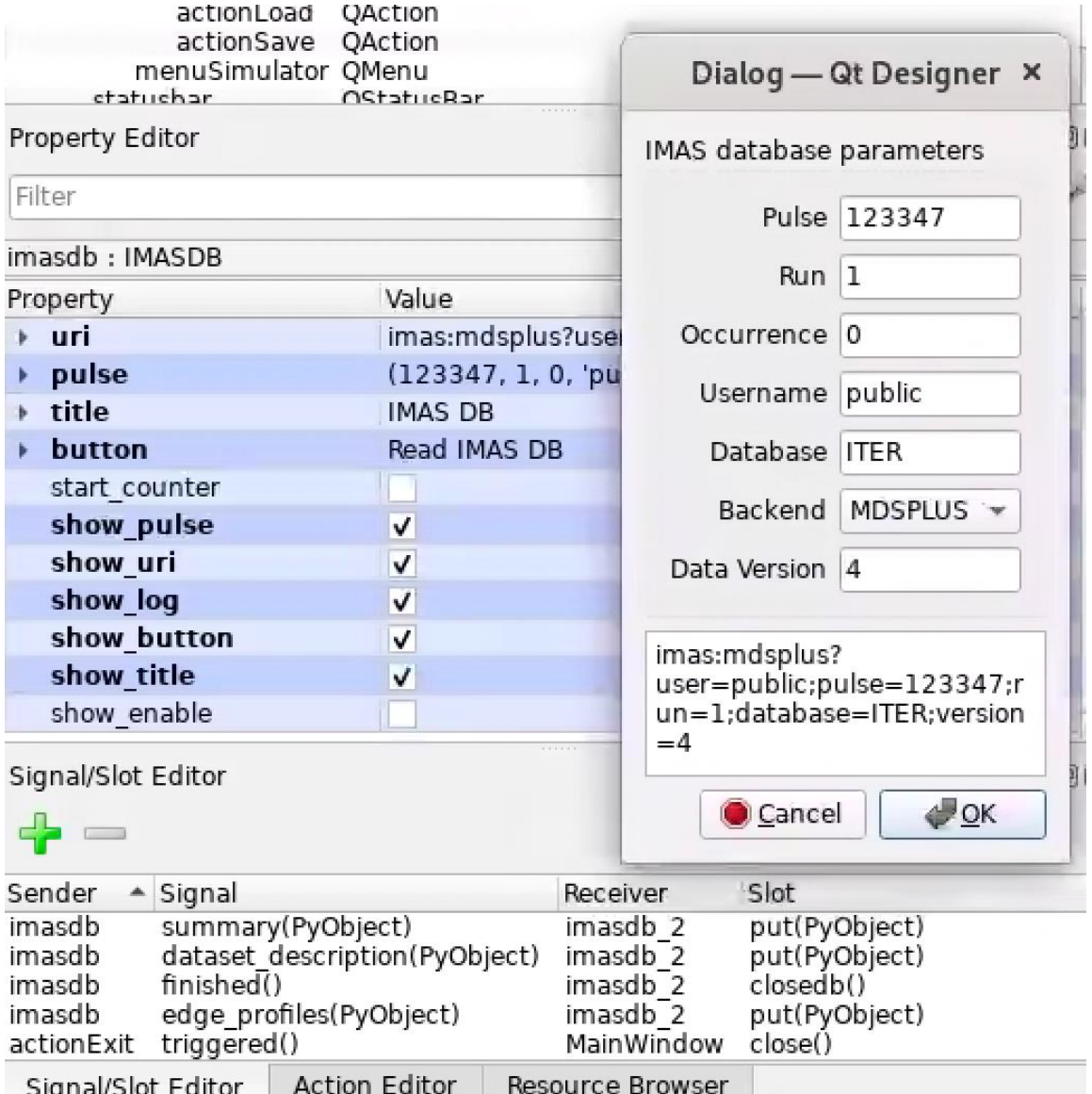
IMAS Database Actor (IMAS DB)

- Example shows IDS copy workflow from *imasdb* public into user‘s *imasdb_2*
- Only selected IDSs are copied by sending IDS object reference signal
- Put slot is using Python multiprocessing Queue() to achieve internal asynchronous writes while other actors can run.
- Get is using Qrunnable Worker within global QThreadPool



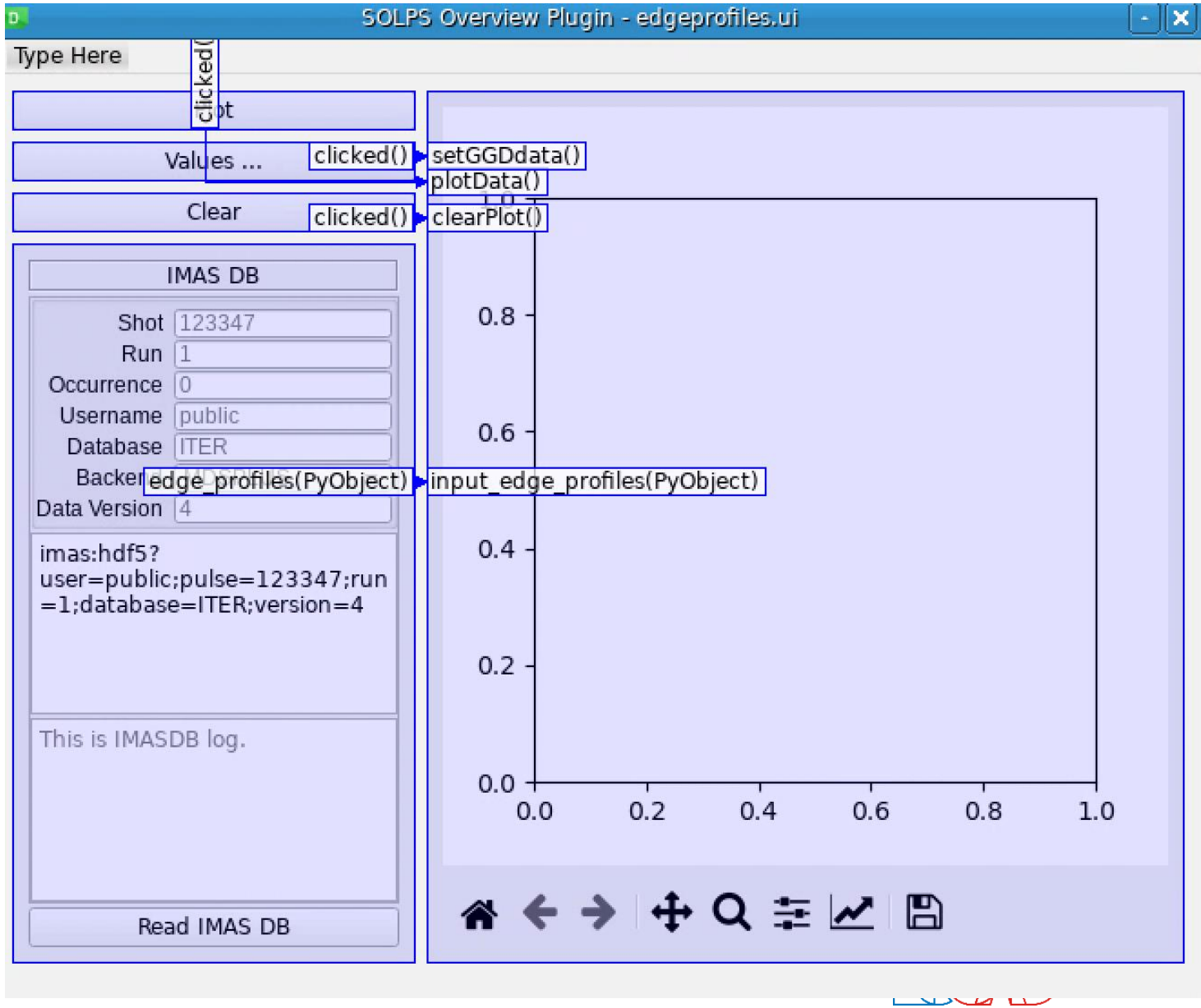
Dynamic properties for design and run

- Dynamic properties are added at the design time as variables that can be used by the custom widget to configure itself.
- Complex properties such as URI can be designed with Designer dialogs



Edge profiles

- Example edge profiles actor that allows selection of plot values and subgrids through dialog.
- IMAS DB “signals” edge_profiles IDS structure to edge_profiles actor that plots selection.

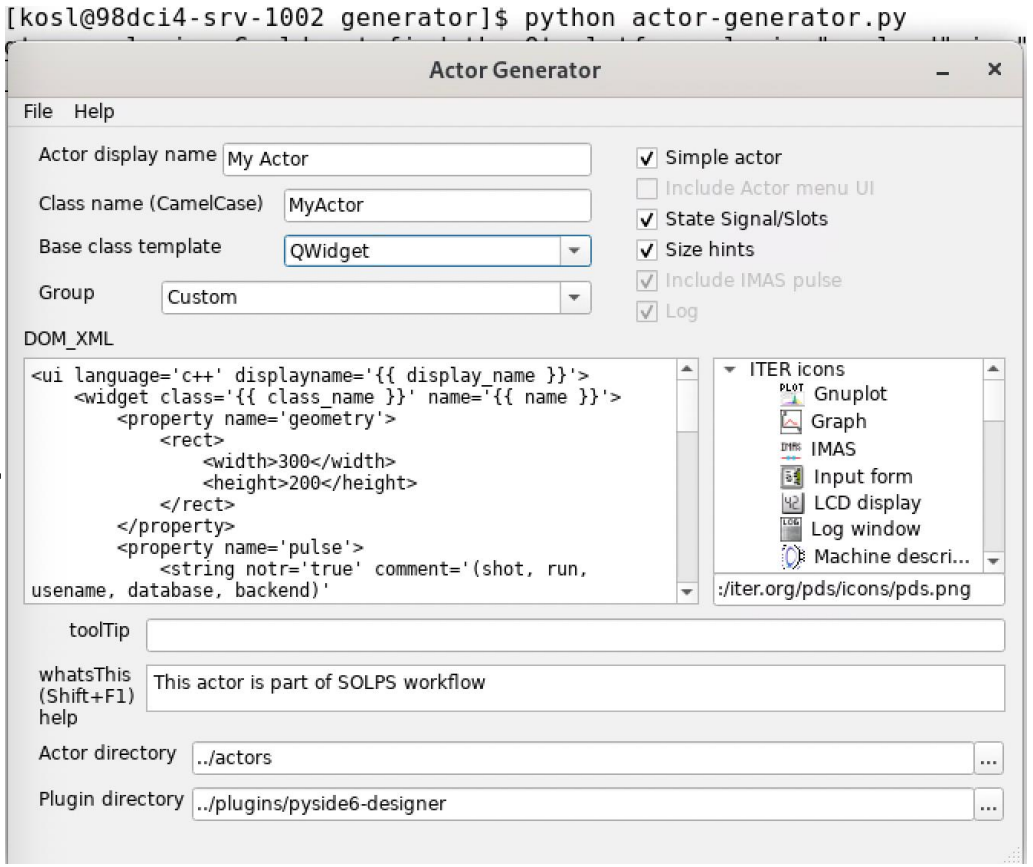


Actor generator for Workflow Designer

Newly developed GUI provides consistent generation of Qt Designer Actor (custom widget) template to

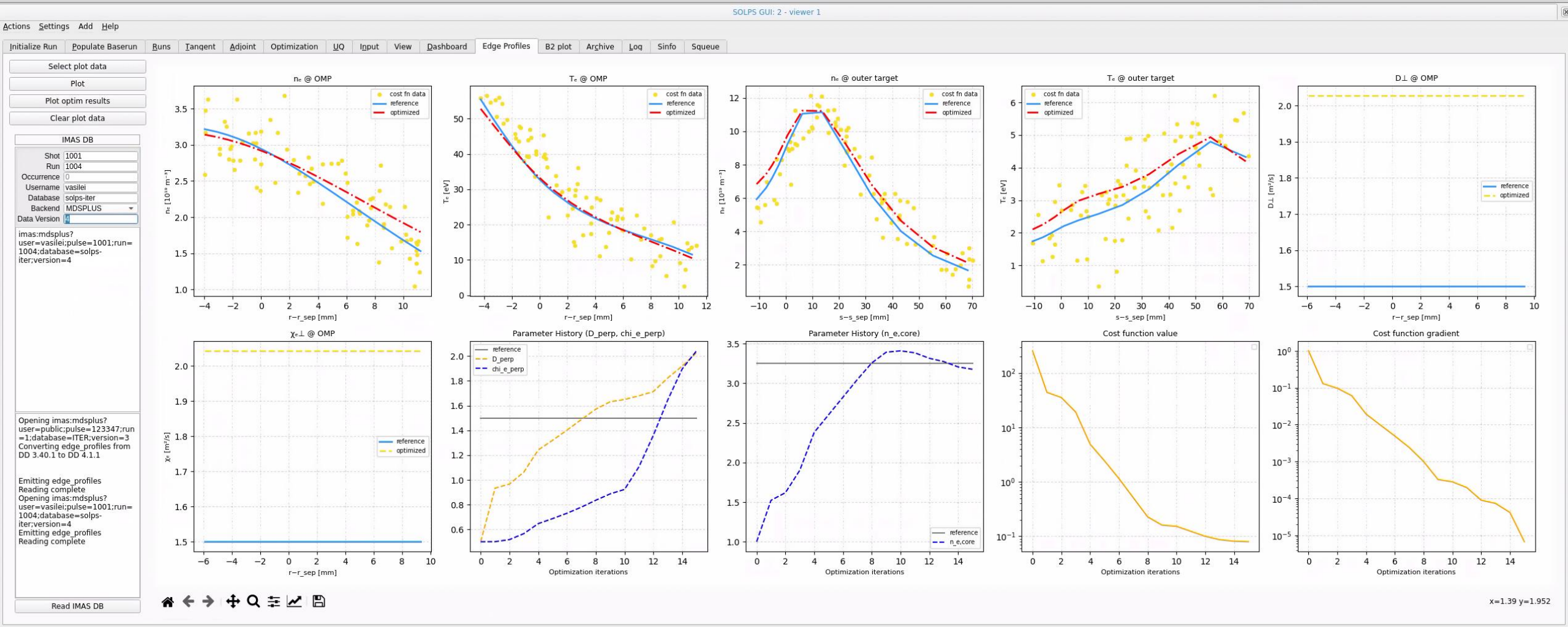
1. Register plugin under Group
2. Generate optional Dialog menu for setting parameters including DOM XML
3. Icon, toolTip and whatIsThis
4. Optional code for signal/slots

Files generated:
~/Documents/actors/actor.py and
~/Documents/designer/
{registeractor.py, actorplugin.py, actor-menu.ui, actor.ui}



Plot Algorithmic Differentiation Optimization

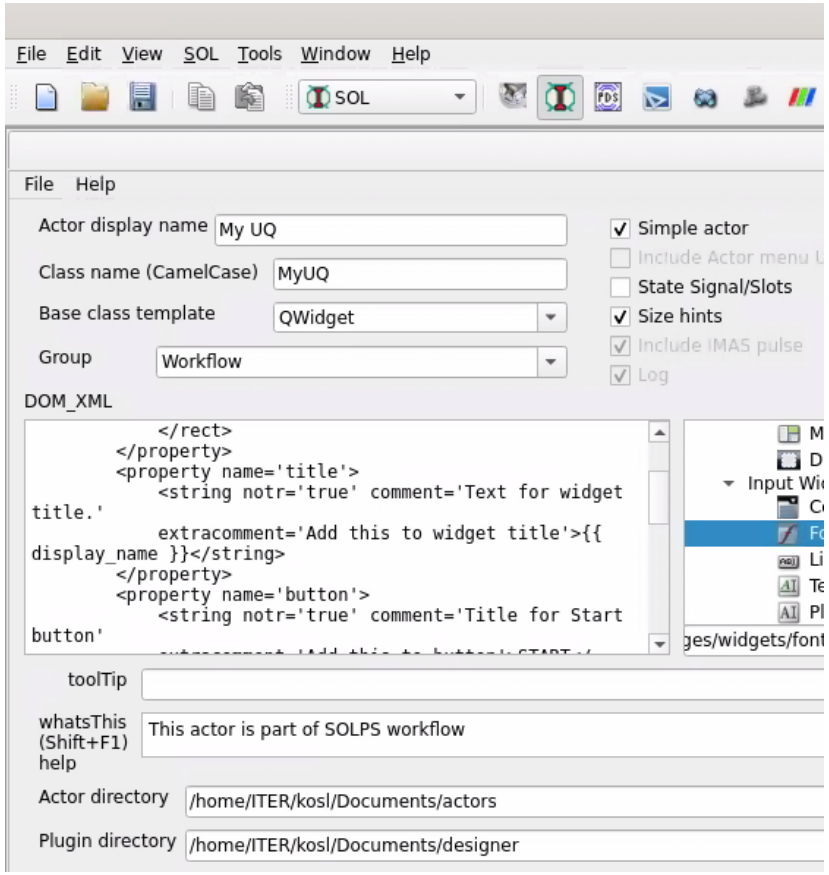
- Enter IDS as shown on the image and press Plot Optim results



Uncertainty Quantification Actor demo

- Start ./SMITER-1.6.4-glibc2.34-x86_64.AppImage
  Select module SOL > SOLPS > Actor Generator and do the following settings

1. Type My UQ in Actor display name
2. Select or type group group as Workflow
3. Add description under toolTip
4. Optionally remove pulse XML section from XML_DOM since we will node be using it.
5. Select an icon in the list of icons
6. For Actor Directory select ~/Documents/actors
   For Designer directory set ~/Documents/designer
7. File > Save Actor

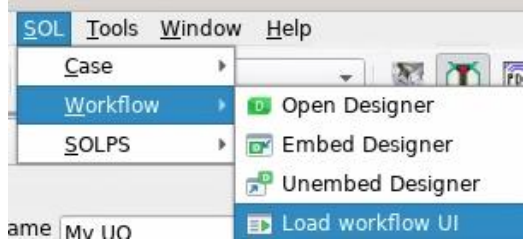


UQ Actor samples generator demo

1. Edit ~/Documents/actors/myuq.py
2. Replace compute() slot with uniform distributions of 3 SOLPS transport parameters that vary 10% from nominal values. Log prints out 30 samples with Latin Hypercube Sampling (near-random)
3. File > Preferences > SOL > Workflow Load UI and select ~/Documents/designer/myuq.ui and then Apply + Close
4. SOL>Workflow > Load workflow UI
5. Press Start or Compute to get samples parameters

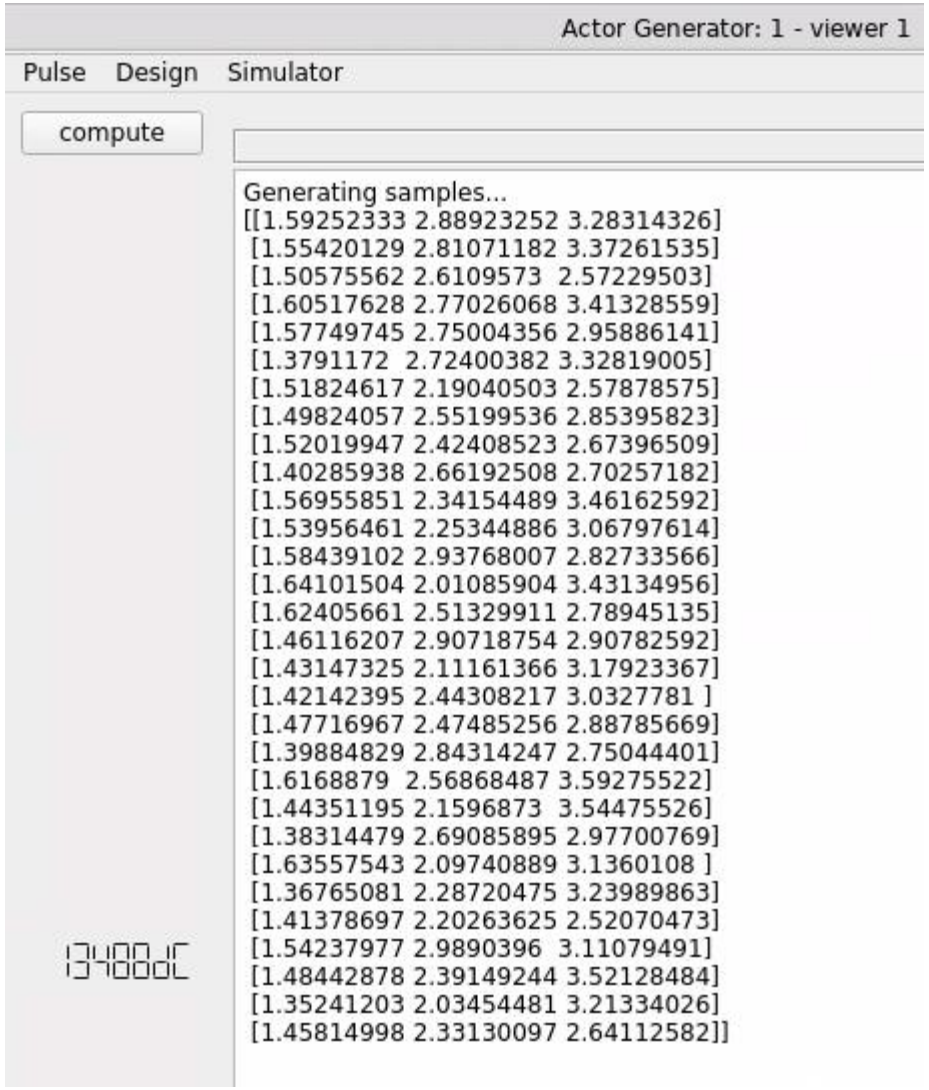
```
@Slot()
def compute(self):
    self._running_state = 1
    self._log.appendPlainText("Generating samples...")
    try:
        import openturns
    except ImportError:
        QMessageBox.critical(self, 'UQ', 'OpenTURNS is not available in this Python.')
        return
    import numpy

    nominal = 1.5
    spread = 0.1*nominal
    marginal1 = openturns.Uniform(nominal-spread, nominal+spread)
    marginal2 = openturns.Uniform(2.0, 3.0)
    marginal3 = openturns.Uniform(2.5, 3.6)
    marginals = [marginal1, marginal2, marginal3]
    distribution =  openturns.JointDistribution(marginals)
    distribution.setDescription(["parm_dna", "parm_hci", "parm_hce"])
    size = 30
    experiment = openturns.LHSExperiment(distribution, size)
    samples = numpy.array(experiment.generate())
    self._log.appendPlainText(str(samples))
    self.finished.emit()
    self._running_state = 2
```



Generating UQ samples

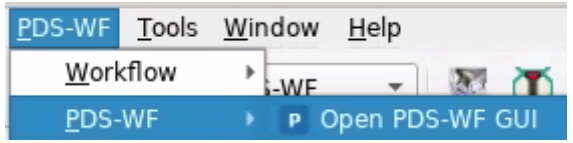
- Add this Actor to solps.ui tab page and run within SOLPS-GUI
- Extend UQ actor with runDir Slot
- Extend UQ actor with creating subdirectories and duplicating required SOLPS input files
- Modify b2.transport.parameters in each sample directory by replacing particle diffusion coefficient, ion heat diffusivity, and electron heat diffusivity (["parm_dna", "parm_hci", "parm_hce"])
- Add batch submission and collective UQ analysis of the results as shown in UQ page



Pulse Design Simulator Workflow demo

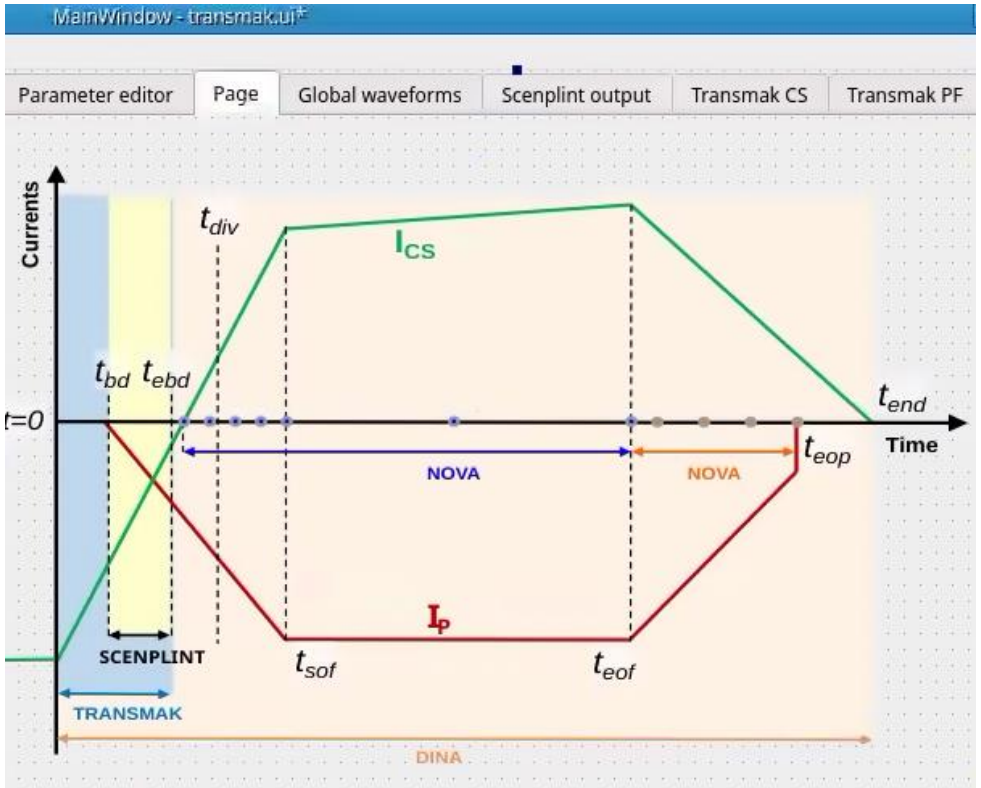
1. Select PDS-WF module > Workflow > Open Designer
2. Under File>Preferences>PDS-WF UI File see read-only path to mounted workflow. For example /tmp/.mount_SMITERKiK2uX/PDS_WF/designer/ScenTranNovaDina.ui and replace it with /tmp/.mount_SMITERKiK2uX/PDS_WF/designer/transmak.ui Copy this path with Ctrl+C and Apply and Close Preferences
3. Open this transmak.ui within PDS-WF Designer and explore Workflow signals with F4/escape. IMAS DB setting and plot setups
4. Run the workflow with PDS-WF>PDS-WF->Open PDS-WF GUI
5. Select Scenario 1, Click Load and Start

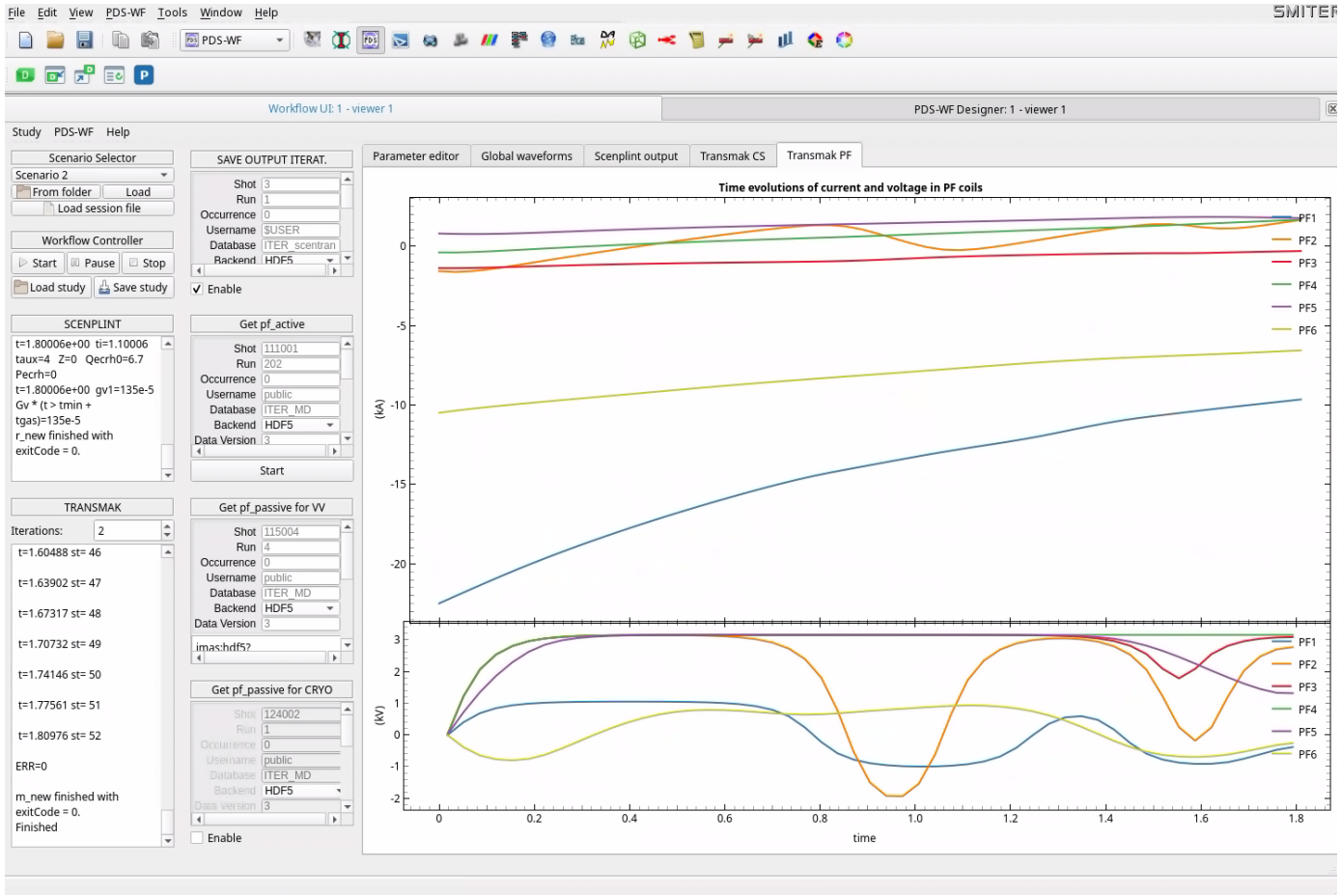
Help > Open Documentation for a complete introduction



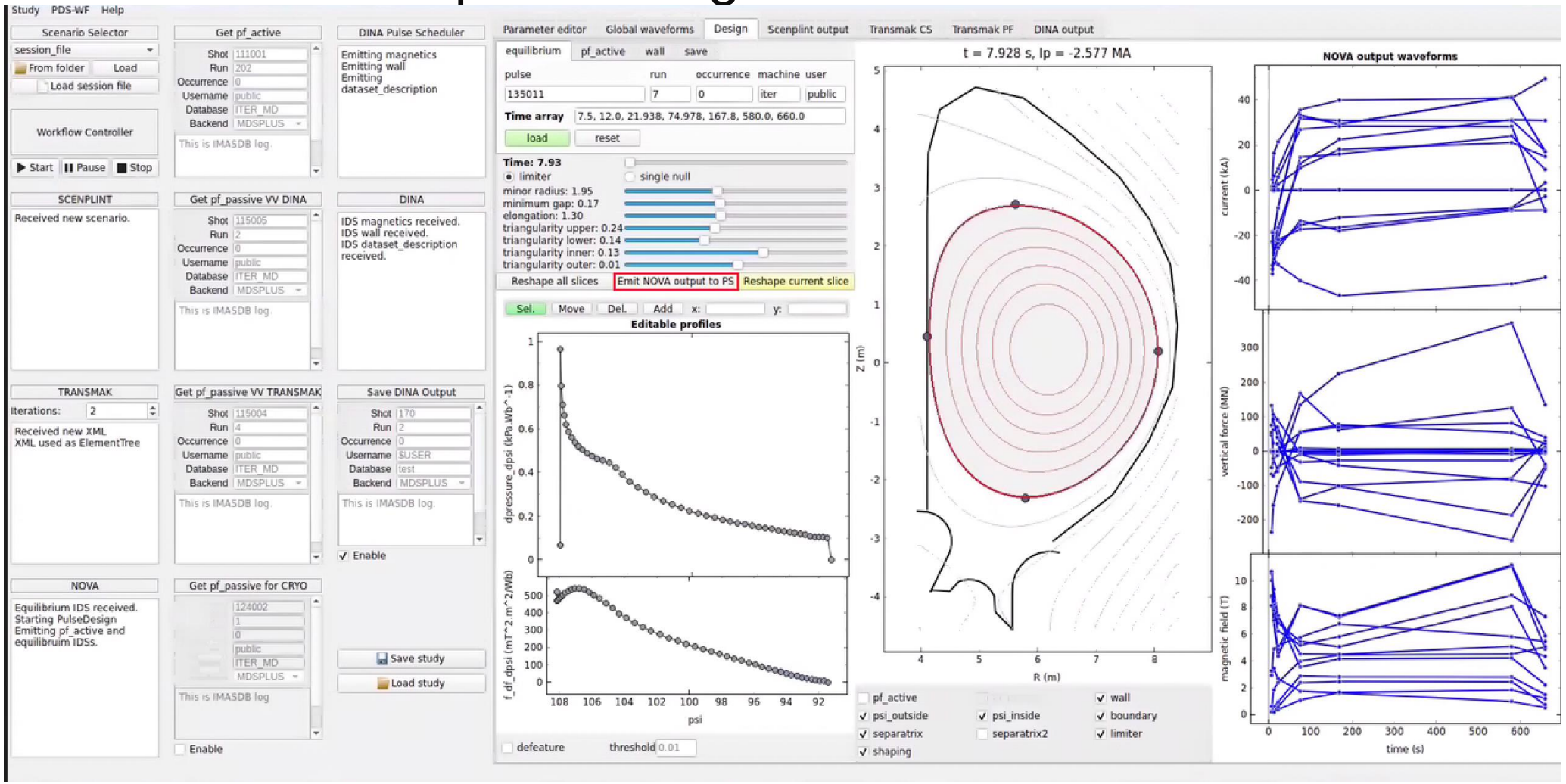
Scenplint Transmak Workflow results

- Add SVG Viewer page with buttons for intuitive parameter editing and plotting as shown below





Complete PDS-WF using Scenplint-Transmak-DINA-NOVA codes with pulse design editors





Conclusion

- VIVID-DTE framework combines reproducible workflows and provides data provenance
- IMAS integration within SOL and PDS workflow demonstrators are extensible for future shareable actors and DTE modules
- Future UDA integration and reproducibility
- Muscle3 actors in workflows and DTE demonstrators
- IPlotlib configurator for more visualizations



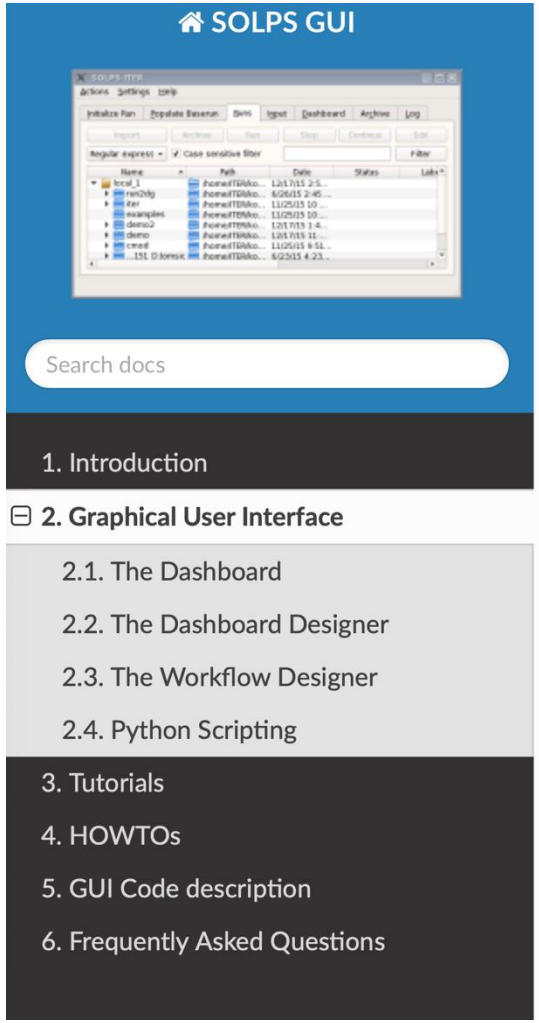
Backup slides

DSO Science Meeting - Visualization Tools for IMAS Applications (4 November 2026) · Indico

Univerza v Ljubljani
Fakulteta za strojništvo

SOLPS-GUI Documentation

https://static.iter.org/imas/assets/solps-iter/html/gui/index.html



/ 2. Graphical User Interface View page source

2. Graphical User Interface

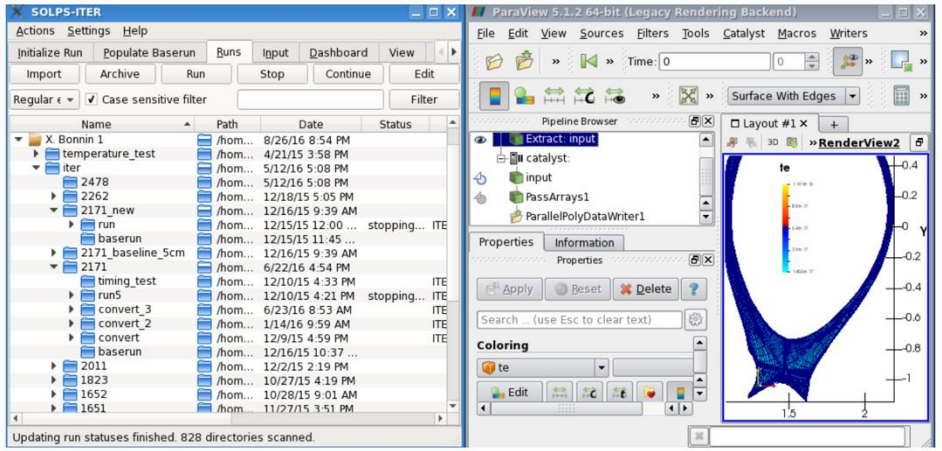


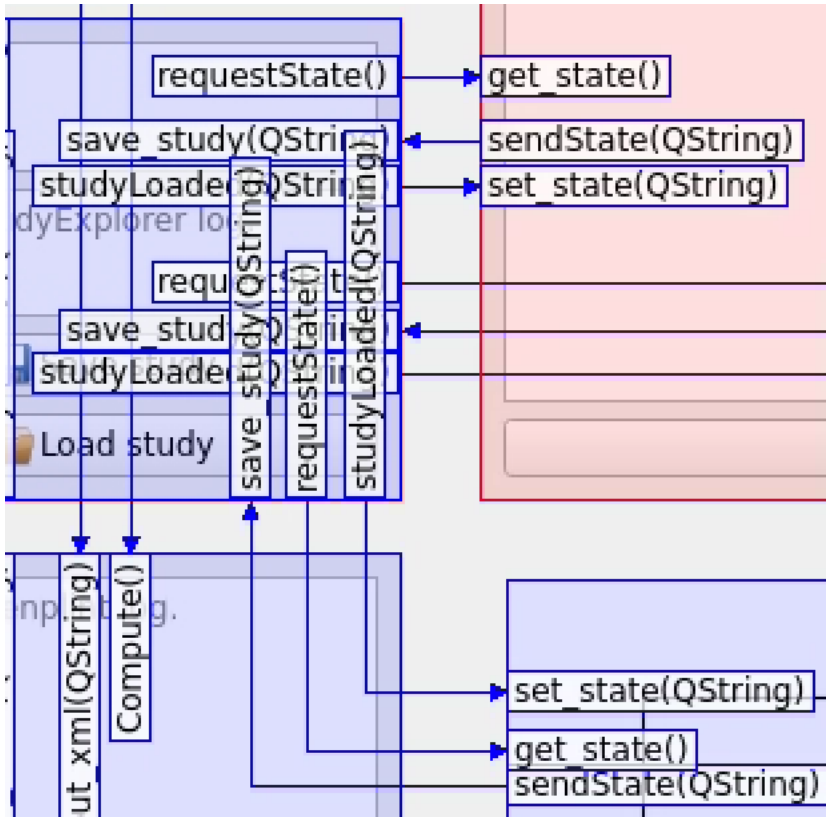
Fig. 2.1 *SOLPS-ITER GUI showing Runs view (left) for controlling simulations and ParaView Catalyst In Situ instrumentation of the SOLPS-ITER code (right).*

In addition to the established SOLPS-ITER code suite a new SOLPS-ITER GUI is being developed in support of users that can now easily work with the codes for preparing and controlling the *runs* on the cluster. SOLPS-

Actor persistence

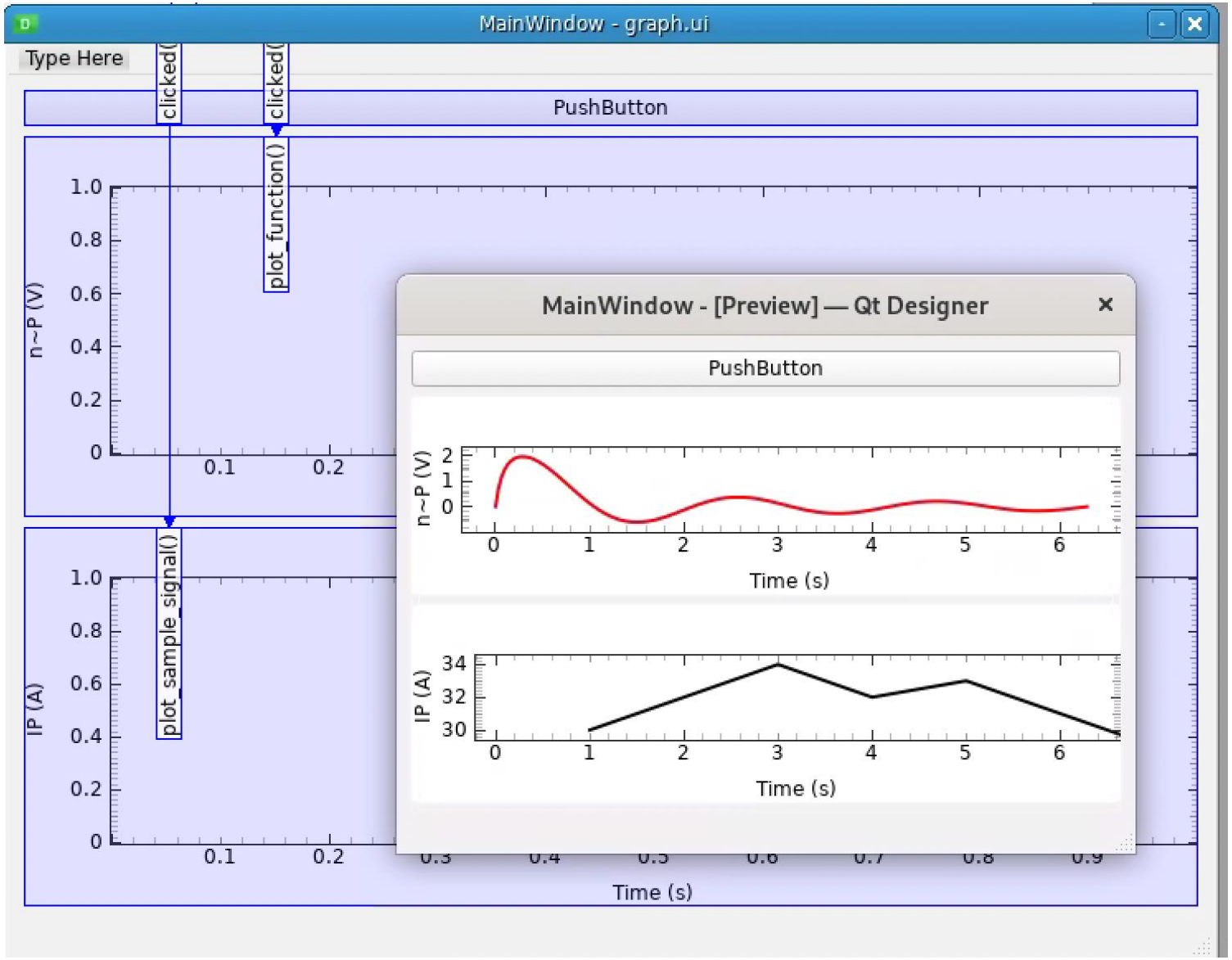
- Each actor can contain its internal state from which it can restart to produce the same results as achieved when running workflow.
- Connection to Study Explorer:

1. Request state -> get state
2. Send state -> save study
3. Study loaded -> set state



Graph (based on PyQtGraph)

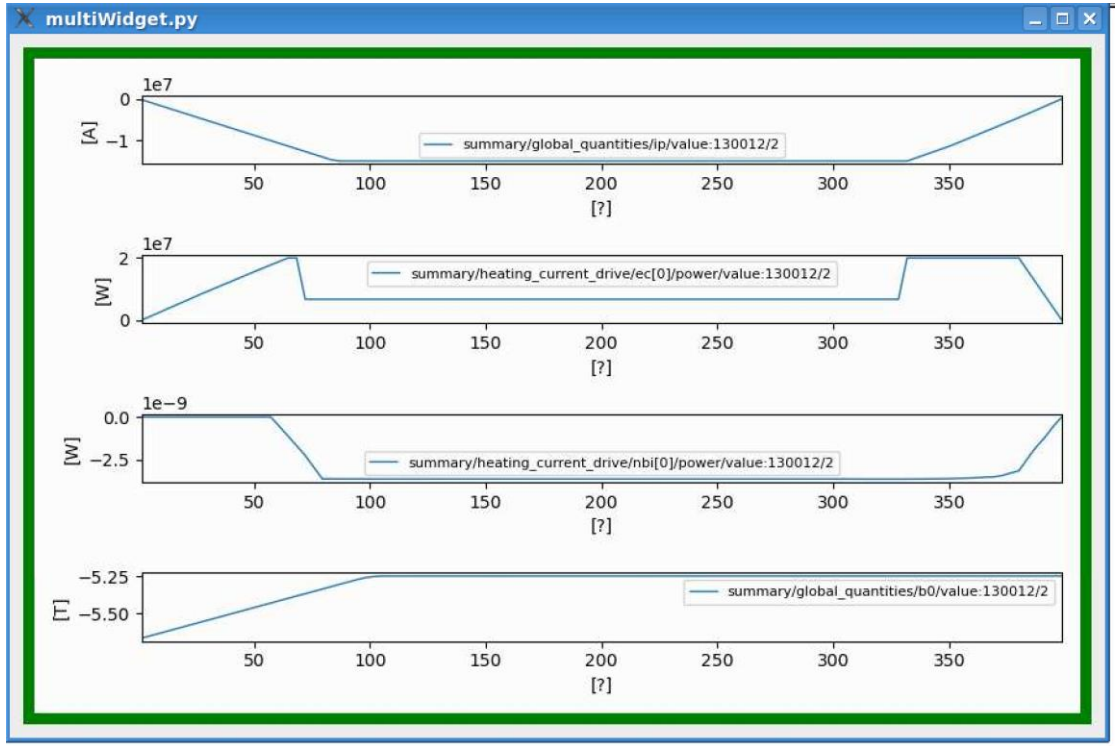
- Can receive IDS and plots it
- Can plot general input signal or filename such as gnuplot
- Multiple graphs can synchronize time axis and ruler.



Univerza v Ljubljani
Fakulteta za strojništvo

ITER plotting library based actor for signal plots

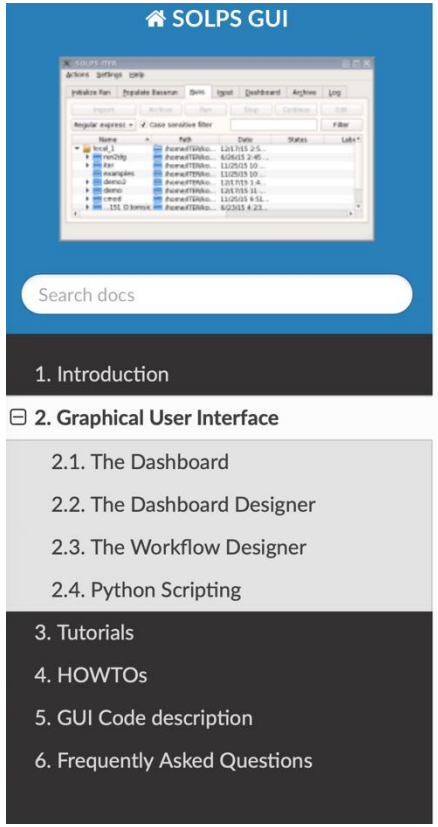
- Ported iplotlib and MINT to PySide6
- Prototype Iplot actor for use in pyside6-designer



Univerza v Ljubljani
Fakulteta za strojništvo

Tutorial for COMPASS #16515 Algorithmic Differentiation

https://static.iter.org/imas/assets/solps-iter/html/tutorial/runs.html#compass-case-for-algorithmic-differentiation-optimization



2. Graphical User Interface View page source

2. Graphical User Interface

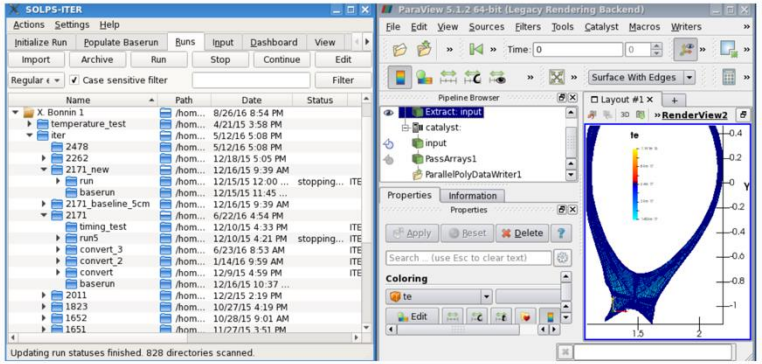
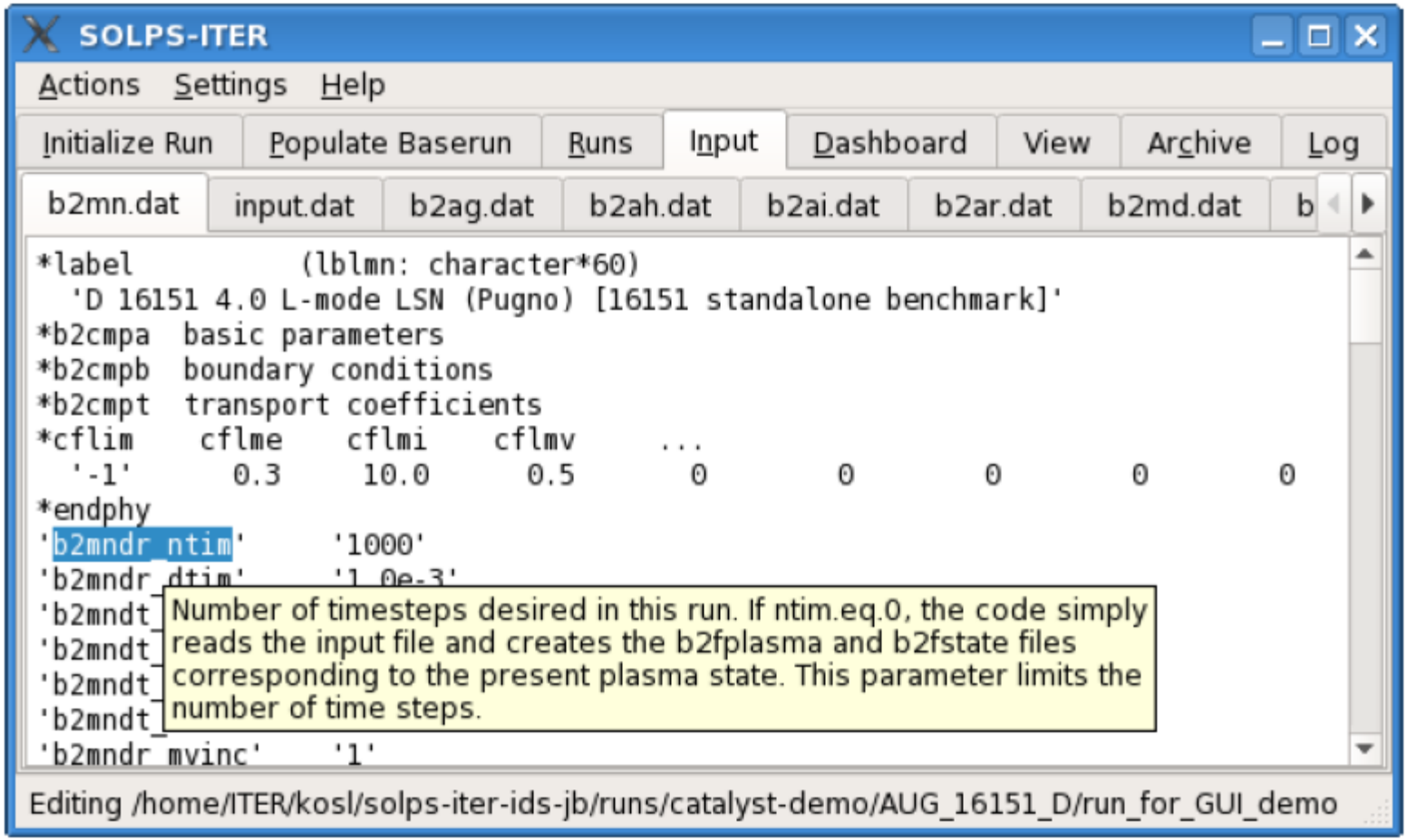


Fig. 2.1 *SOLPS-ITER GUI showing Runs view (left) for controlling simulations and ParaView Catalyst In Situ instrumentation of the SOLPS-ITER code (right).*

In addition to the established SOLPS-ITER code suite a new SOLPS-ITER GUI is being developed in support of users that can now easily work with the codes for preparing and controlling the *runs* on the cluster. SOLPS-

Editing input files and inline documentation

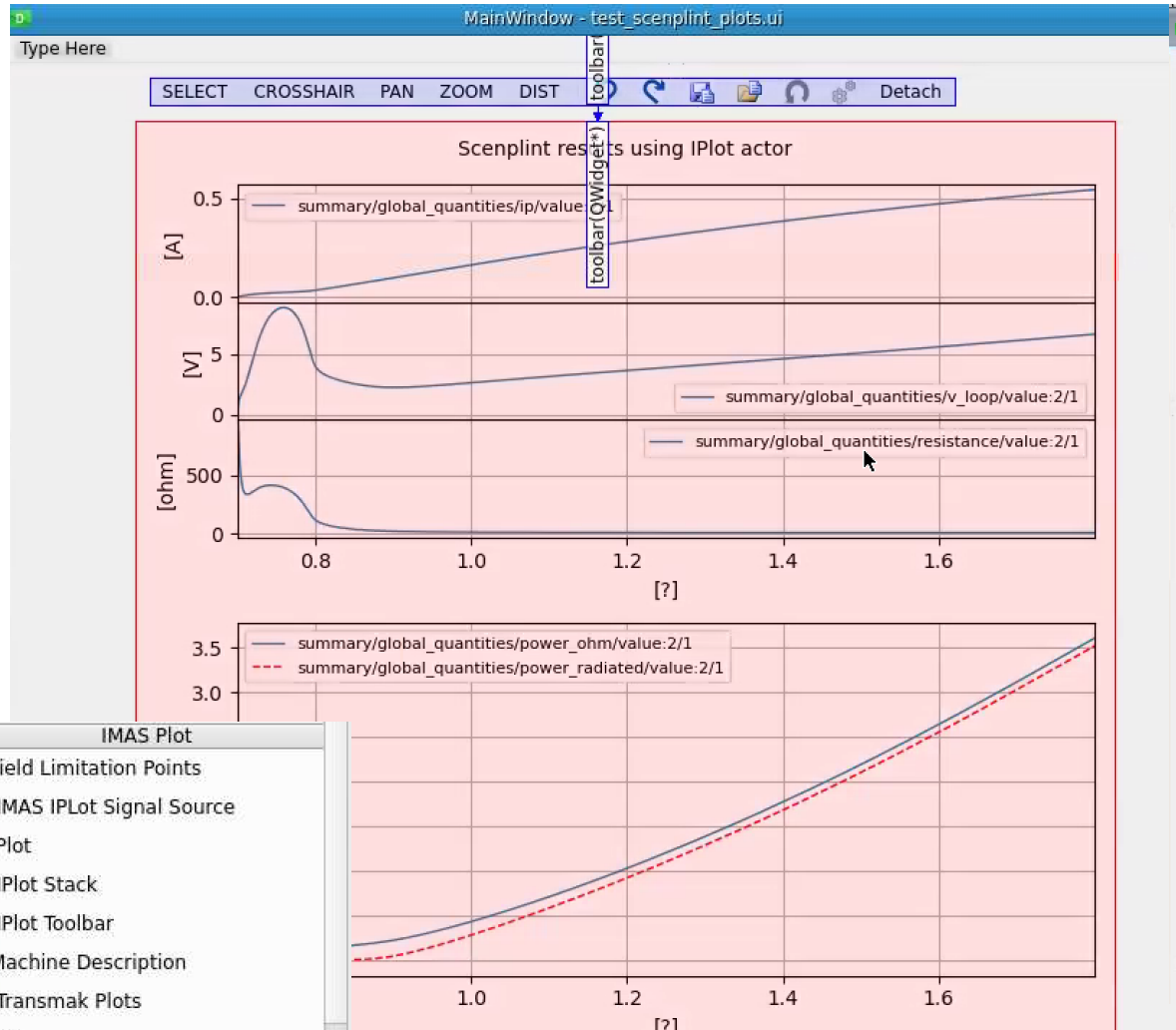


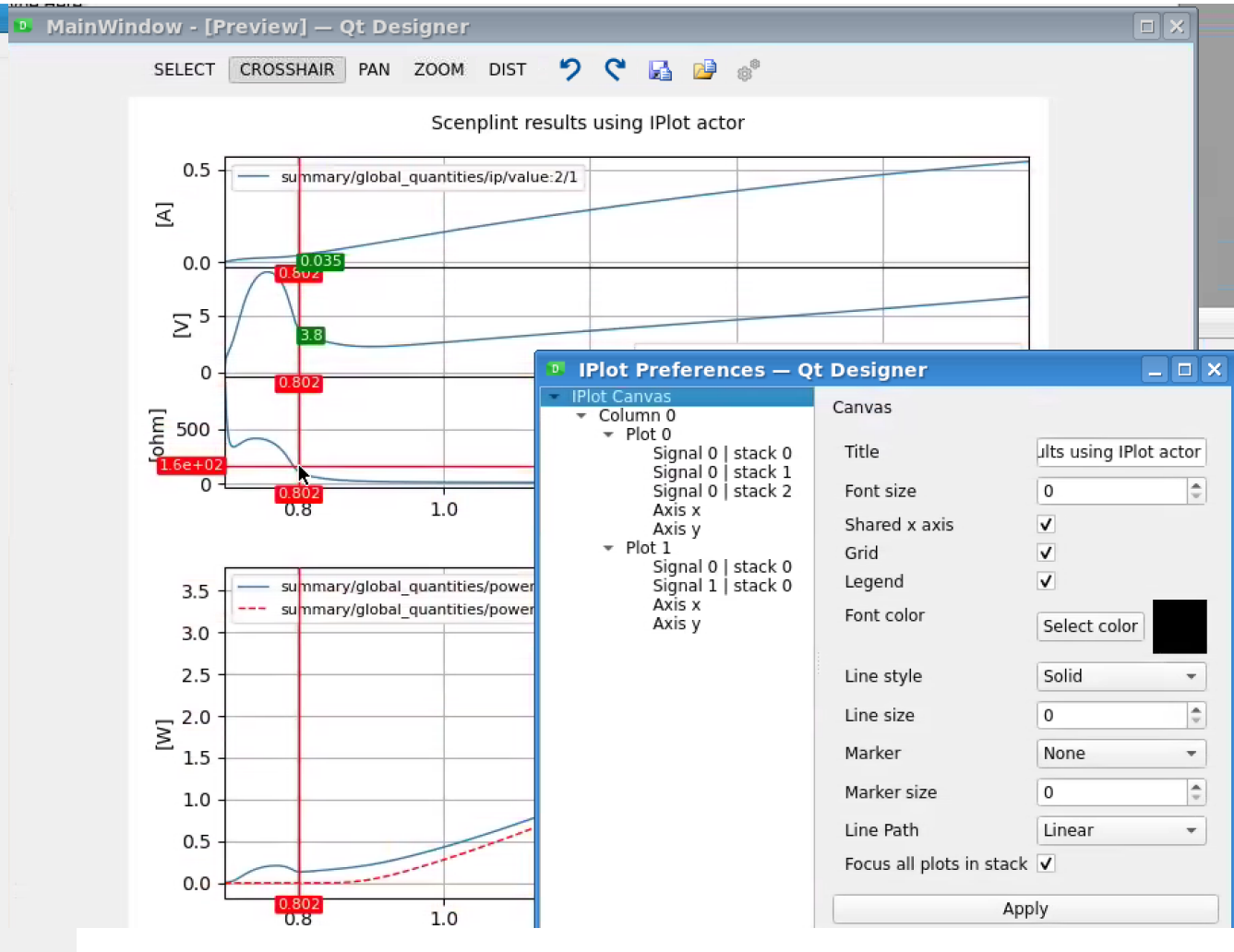
Input switches documentation provided by means of an XML file, re-used for source code and manual build-up

Univerza v Ljubljani
Fakulteta za strojništvo

IPlot actor with toolbar and preferences

IPlot based on Iplotlib with Matplotlib can be used to prepare plot (stacked too).

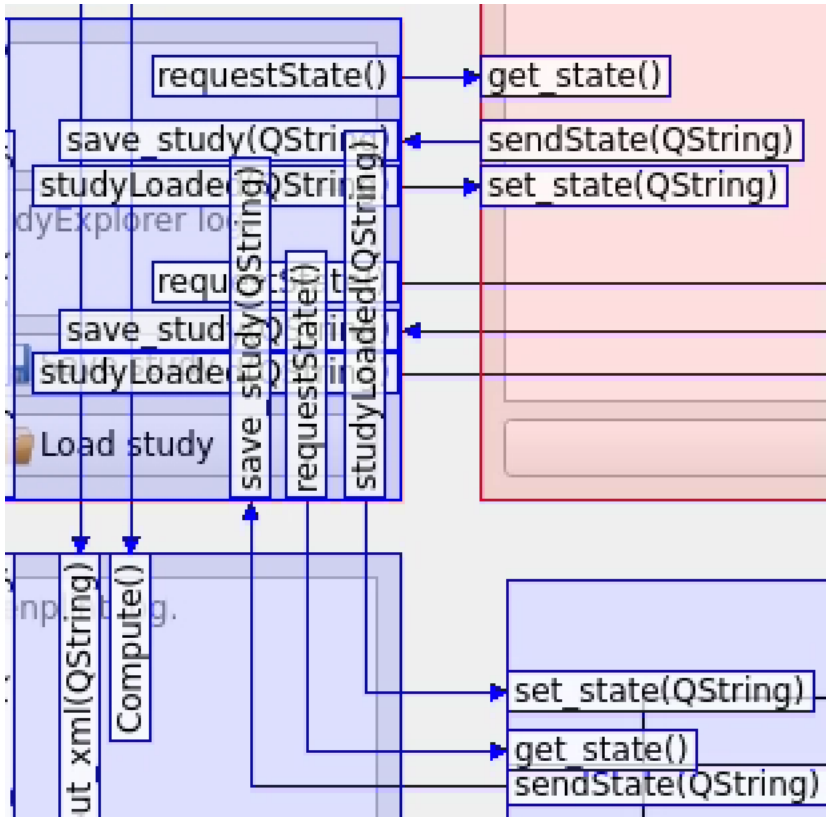




Actor persistence

- Each actor can contain its internal state from which it can restart to produce the same results as achieved when running workflow.
- Connection to Study Explorer:

1. Request state -> get state
2. Send state -> save study
3. Study loaded -> set state





UQ with OpenTURNS

- Treatment of Uncertainties, Risks aNd Statistics
- https://openturns.github.io
- Python interface
- Actively developed

```
import openturns as ot
import persalys

Study_0 = persalys.Study('Study_0')
persalys.Study.Add(Study_0)

# variables
dist_z0 = ot.Uniform(100, 150)
z0 = persalys.Input('z0', 100, dist_z0, '')
dist_v0 = ot.Normal(55, 10)
v0 = persalys.Input('v0', 55, dist_v0, '')
dist_m = ot.Normal(80, 8)
m = persalys.Input('m', 80, dist_m, '')
dist_c = ot.Uniform(0, 30)
c = persalys.Input('c', 16, dist_c, '')
z = persalys.Output('z', '')
z2 = persalys.Output('z2', 'fake output')
inputs = [z0, v0, m, c]
outputs = [z, z2]

# mesh model
meshModel = persalys.GridMeshModel(ot.Interval(0.

# Python model
code = 'from math import exp\n\ndef _exec(z0,v0,m
n    z = [max(z0 + vinf * t_i[0] + tau * (v0 - vi
PhysicalModel_1 = persalys.PythonFieldModel('Phys
Study_0.add(PhysicalModel_1)
```

Persalys GUI within SALOME

https://persalys.github.io/

Allows study creation and analysis

