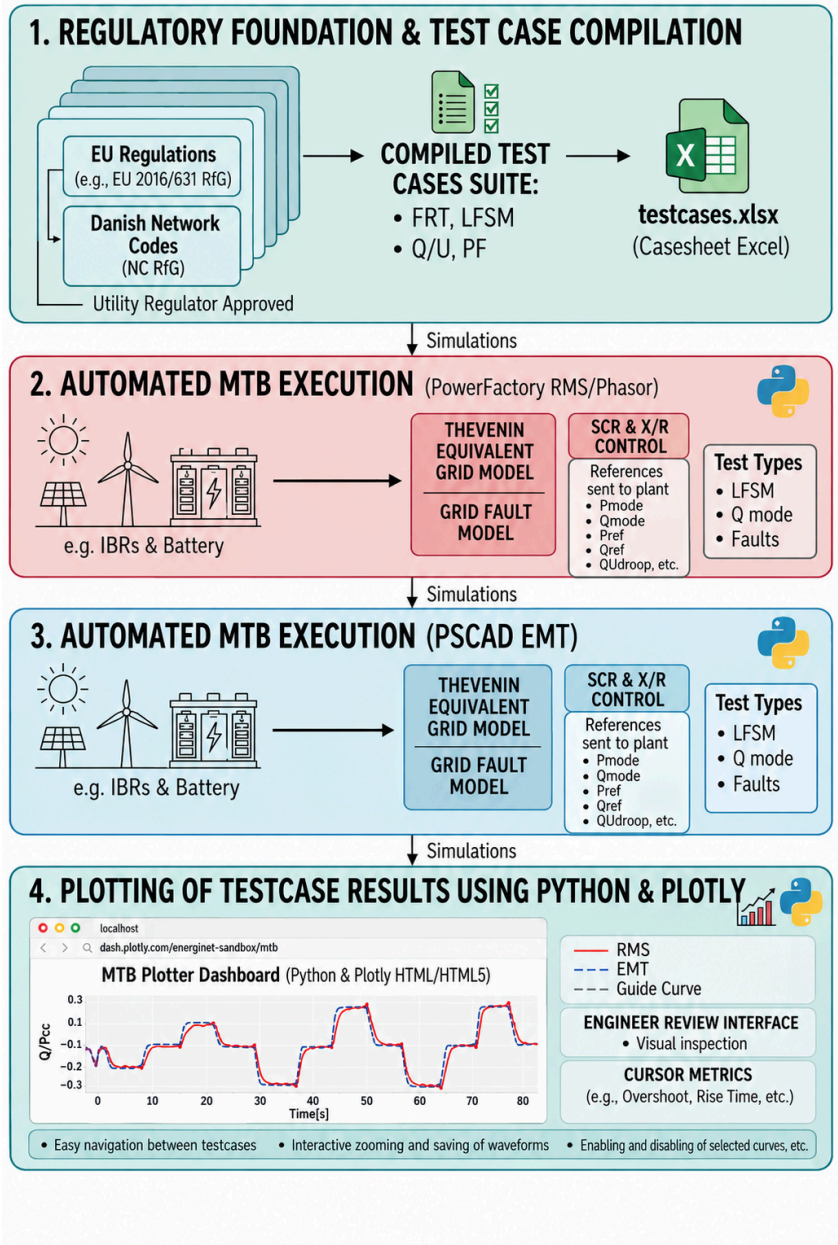
MTB 2.0 Quickstart Guide



ENERGINET

Table of Contents

Welcome to the MTB

- MTB Workflow
- MTB Quickstart Wiki Guides
- MTB Appendices Wiki Guides
- Example Setups
- Related Links
- Contact

1. Quickstart Testcases Workbook Guide

- 1.1 Overview
- 1.2 Settings Sheet
  - 1.2.1 General plant settings
  - 1.2.2 Grid strength settings
  - 1.2.3 Timing and PowerFactory settings
  - 1.2.4 Co-located settings
- 1.3 Case Sheets
  - 1.3.1 Case basic information
  - 1.3.2 Case initial settings
  - 1.3.3 Pmode and Qmode values
  - 1.3.4 Case events
- 1.4 Example Case Setup

2. Quickstart PowerFactory Guide

- 2.1 System Requirements
- 2.2 Preparation
  - 2.2.1 Create an MTB installation folder
  - 2.2.2 Edit the Settings sheet
  - 2.2.3 Edit config.ini
- 2.3 Model Setup in PowerFactory
  - 2.3.1 Import the MTB into the project
  - 2.3.2 Connect the MTB to the plant controller
  - 2.3.3 Configure the MTB composite model
  - 2.3.4 Connect the inverter to the MTB
  - 2.3.5 Configure execute.ComPython
- 2.4 Script Execution Finished
- 2.5 Add Additional Measurements to Result Files
- 2.6 Custom Signal Setup
  - 2.6.1 Custom signal setup through execute.ComPython
  - 2.6.2 Custom signal setup through the MTB frame
- 2.7 Troubleshooting

3. Quickstart PSCAD Guide

3.1 System Requirements
3.1.1 Running from within PSCAD
3.1.2 Running from the command line
3.1.3 Example workspace
3.2 Preparation
3.2.1 Create an MTB installation folder
3.2.2 Edit the Settings sheet
3.2.3 Edit config.ini
3.3 MTB Setup in PSCAD
3.3.1 Import the MTB into the PSCAD workspace
3.3.2 Connect the MTB block to the PoC
3.3.3 Connect MTB controls to the PPC
3.3.4 Connect additional MTB measurements (optional)
3.4 Script Execution
3.4.1 Running all selected cases (Volley mode)
3.4.2 Running individual cases (Manual mode)
3.5 Script Execution Finished
3.6 Troubleshooting
3.6.1 Wrong compiler
3.6.2 UnicodeDecodeError
3.6.3 PSCAD Out of Memory (OOM) error

4. Quickstart Plotter Guide
4.1 System Requirements
4.2 Preparation
4.3 Configuration of config.ini
4.4 Configuration of figureSetup.csv
4.5 Configuration of cursorSetup.csv
4.6 Script Execution
4.7 Plot Result Examples

5. Quickstart Guide Curve Generation
5.1 Introduction
5.2 How Guide Curves Are Selected
5.3 General Guide Helpers
5.3.1 guideLPF
5.3.2 guideDelay
5.4 Dedicated Guide Functions
5.4.1 guidePramp
5.4.2 guidePramp2
5.4.3 guideLFSMRamp
5.4.4 guideLFSM
5.4.5 guideFSM
5.4.6 guideQU
5.4.7 guideQpf
5.4.8 guideFFC

5.4.9 guideSIPS
5.4.10 guideSIPS2

6. Quickstart Cursor Metrics

6.1 Introduction
6.2 Cursor Setup
6.3 Cursor Time Ranges
6.4 Cursor Functions
6.4.1 START
6.4.2 END
6.4.3 DELTA
6.4.4 MIN
6.4.5 MAX
6.4.6 MEAN
6.4.7 GRAD_MIN
6.4.8 GRAD_MAX
6.4.9 GRAD_MEAN
6.4.10 RESPONSE
6.4.11 RISE_FALL
6.4.12 SETTLING
6.4.13 OVERSHOOT
6.4.14 FSM_DROOP
6.4.15 LFSM_DROOP
6.4.16 QU_T1
6.4.17 QU_T2
6.4.18 QU_DROOP
6.4.19 QU*SS*TOL
6.4.20 DELTA_FFC

7. Quickstart Python Utility Scripts

7.1 Script overview
7.2 PowerFactory model check
7.3 Export component data from PowerFactory
7.4 Export DSL checksums from PowerFactory
7.5 Export relay data from PowerFactory
7.6 Compare PowerFactory and PSCAD component data
7.7 Clean unused PSCAD project definitions
7.8 List signals in a `.psout` file
7.9 Recover `.psout` files after a PSCAD OOM crash
7.10 Notes and cautions

Appendix A: Working with .psout files

A.1 Introduction
A.2 PSOUT Reader
A.3 Enerplot
A.4 MTB library functions and scripts

A.4.1 Script examples
A.4.2 Function examples

Appendix B: Changelog

[2.0.0] - 2026-09-07
Added
Changed
Fixed
[1.1.2] - 2025-06-18

Welcome to the MTB

MTB (**M**odel **T**est **B**ench) is a test bench for automating grid compliance simulation studies in both PowerFactory and PSCAD, with external plotting and comparison of RMS and EMT results. MTB is intended to help facility owners and model providers evaluate simulation model behaviour in the context of Danish grid-code requirements and Energinet simulation model requirements.

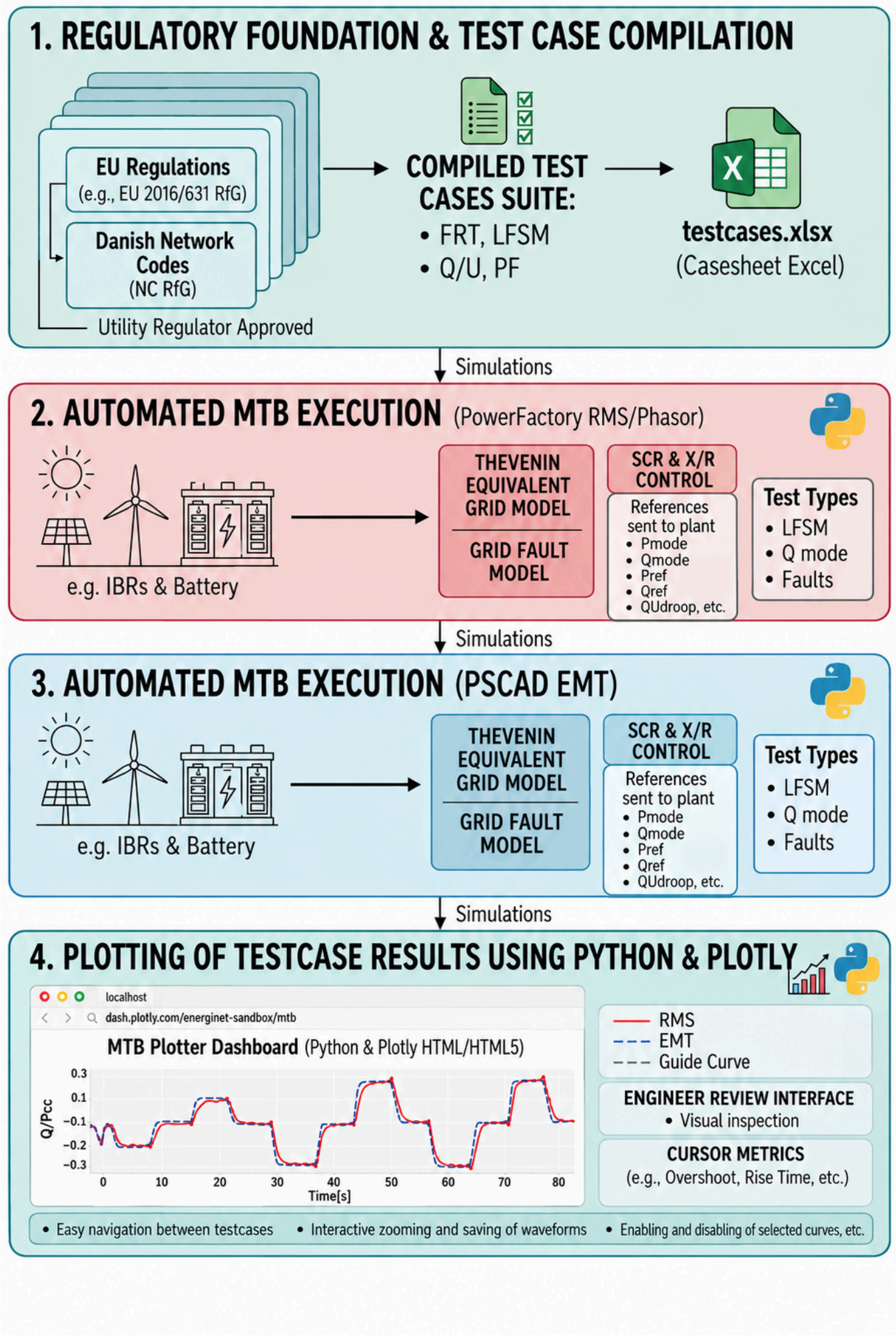
MTB includes predefined case sets for:

- RfG (Requirements for Generators)
- DCC (Demand Connection Code)
- Unit testing
- Co-located generation and demand cases (*Not yet fully configured in MTB 2.0*)
- Custom user-defined cases

MTB Workflow

1. Configure the testcase workbook to test various requirements in e.g. EU 2016/631 with specific relevance to the Danish NC RfG
2. Set up the MTB in PowerFactory and execute the selected simulation cases
3. Set up the MTB in PSCAD and execute the selected simulation cases
4. Plot and compare RMS and EMT results making use of among other things, analytically calculated guide curves, cursor metrics, and visual inspection

> **IMPORTANT**
>
> Using MTB is not a guarantee of model compliance. The plant owner remains responsible for ensuring that models comply with the requirements applicable at any time.


1. REGULATORY FOUNDATION & TEST CASE COMPILATION
EU Regulations
(e.g., EU 2016/631 RfG)
Danish Network Codes
(NC RfG)
Utility Regulator Approved
COMPILED TEST CASES SUITE:
• FRT, LFSM
• Q/U, PF
testcases.xlsx
(Casesheet Excel)
Simulations
2. AUTOMATED MTB EXECUTION (PowerFactory RMS/Phasor)
e.g. IBRs & Battery
THEVENIN EQUIVALENT GRID MODEL
GRID FAULT MODEL
SCR & X/R CONTROL
References sent to plant
• Pmode
• Qmode
• Pref
• Qref
• QUdroop, etc.
Test Types
• LFSM
• Q mode
• Faults
Simulations
3. AUTOMATED MTB EXECUTION (PSCAD EMT)
e.g. IBRs & Battery
THEVENIN EQUIVALENT GRID MODEL
GRID FAULT MODEL
SCR & X/R CONTROL
References sent to plant
• Pmode
• Qmode
• Pref
• Qref
• QUdroop, etc.
Test Types
• LFSM
• Q mode
• Faults
Simulations
4. PLOTTING OF TESTCASE RESULTS USING PYTHON & PLOTLY
localhost
dash.plotly.com/energinet-sandbox/mtb
MTB Plotter Dashboard (Python & Plotly HTML/HTML5)
Q/Pcc
0.3
0.1
−0.1
−0.2
−0.3
0
10
20
30
40
50
60
70
80
Time[s]
RMS
EMT
Guide Curve
ENGINEER REVIEW INTERFACE
• Visual inspection
CURSOR METRICS
(e.g., Overshoot, Rise Time, etc.)
• Easy navigation between testcases
• Interactive zooming and saving of waveforms
• Enabling and disabling of selected curves, etc.

MTB Quickstart Wiki Guides

Section	Page	Purpose
1	Quickstart Testcases Workbook Guide	Configure `testcases.xlsx`, case sheets, settings, and event types.
2	Quickstart PowerFactory Guide	Set up and execute MTB cases in PowerFactory.
3	Quickstart PSCAD Guide	Set up and execute MTB cases in PSCAD.
4	Quickstart Plotter Guide	Configure and run the MTB plotter.
5	Quickstart Guide Curve Generation	Understand analytical guide curves generated by the plotter.
6	Quickstart Cursor Metrics	Configure and interpret cursor metric tables.
7	Quickstart Python Utility Scripts	Use helper scripts for model checks, data extraction, `.psout` inspection, and recovery.

MTB Appendices Wiki Guides

Appendix	Page	Purpose
A	Working with .psout files	Use PSCAD `.psout` files with MHI tools, MTB helper functions, and conversion/listing scripts.

Example Setups

Example PowerFactory and PSCAD setups are available in setup_examples. These examples demonstrate the MTB setup workflow only; they are not representative of compliant plant models.

Related Links

- MTB README
- MTB CHANGELOG
- MTB Wiki
- MTB Releases
- MTB Issues
- Energinet electricity rules and regulations

Contact

For inquiries, please contact the Energinet simulation model team: simuleringsmodeller@energinet.

1. Quickstart Testcases Workbook Guide

1.1 Overview

The `testcases.xlsx` workbook is the model-independent MTB interface used to enter plant settings and define simulation cases. The workbook contains predefined case sets used by Energinet for RfG, DCC, and unit test studies, and it also supports custom cases.

The workbook contains these sheets:

Sheet	Purpose
`Settings`	Model-specific plant data, grid data, default control settings, simulation timing, and execution options.
`Area values`	Requirement values derived from plant-specific settings such as area and connection point.
`RfG cases`	Predefined generation cases.
`DCC cases`	Predefined demand cases.
`Unit cases`	Unit test cases.
`Co-located cases`	Predefined co-located generation/demand cases.
`Custom cases`	User-defined custom cases.
`Event types`	Describes supported event types and how `X1` and `X2` are interpreted.
`RfG cases overview`	Overview of predefined RfG cases.
`DCC cases overview`	Overview of predefined DCC cases.
`Unit cases overview`	Overview of predefined unit cases.
`Co-located cases overview`	Overview of predefined co-located cases (*Not yet fully configured in MTB 2.0*).
`Custom cases overview`	Overview of custom cases.
`datavalidation`	Workbook helper sheet used for data validation lists.

The `Settings` sheet and case sheets are read directly by case_setup.py.

1.2 Settings Sheet

	A	B	C	D
1	**Name**	**Value**	**Unit**	**Comment**
2	**Casegroup**	RfG	-	
3	**Run custom cases**	FALSK	-	*If custom not selected for casegroup.*
4	**Projectname**	Solbakken	-	
5	**Pn**	198,90	MW	
6	**Default P available**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed*
7	**Number of Units**	3	-	
8	**PnG3**	300,00	MW	*Allowed generation capacity in PCC*
9	**PnD3**	100,00	MW	*Allowed consumption capacity in PCC*
10	**Unit A type**	Generation		
11	**Pn Unit A [Generation]**	250,00	MW	*Nominal power (Generation)*
12	**Default P available or SoC Unit A**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
13	**Pn Unit A [Consumption]**	0,00	MW	*Nominal power (Consumption)*
14	**Unit B type**	Energy Storage		
15	**Pn Unit B [Generation]**	100,00	MW	*Nominal power (Generation)*
16	**Default P available or SoC Unit B**	0,50	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
17	**Pn Unit B [Consumption]**	100,00	MW	*Nominal power (Consumption)*
18	**Unit C type**	Demand		
19	**Pn Unit C [Generation]**	0,00	MW	*Nominal power (Generation)*
20	**Default P available or SoC Unit C**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
21	**Pn Unit C [Consumption]**	10,00	MW	*Nominal power (Consumption)*
22	**Unit D type**	Generation		
23	**Pn Unit D [Generation]**	10,00	MW	*Nominal power (Generation)*
24	**Default P available or SoC Unit D**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
25	**Pn Unit D [Consumption]**	0,00	MW	*Nominal power (Consumption)*
26	**Uc**	161,90	kV	*Used for easy setup to make sure simulations, start in normal operation voltage and not 1 pu.*
27	**Un**	152,00	kV	*This will be 1pu in simulations and results*
28	**Area**	DK1		
29	**FSM deadband**	0,05	Hz	*If any*
30	**FSM droop**	2,0	%	
31	**SCR min**	5,0	-	
32	**SCR tuning**	23,7	-	
33	**SCR max**	42,4	-	
34	**Default Q(U) droop**	4,0	%	
35	**X/R SCR min**	8,10	-	
36	**X/R SCR tuning**	16,90	-	
37	**X/R SCR max**	25,60	-	
38	**Main Transformer Grounded**	FALSK	-	*Default grounding setting for the Main Transformer*
39	**R0**	3,92	ohm	*Grounding impedance of the MTB voltage source*
40	**X0**	18,30	ohm	*Grounding impedance of the MTB voltage source*
41	**Default Q mode**	Q(U)	-	*Qmode signal from MTB is constant. Qref mode == 0, Q(U) mode == 1 and Qpf == 2.*
42	**PSCAD Timestep**	5	µs	
43	**PSCAD Initialization time**	3,55	s	*Atleast 1s*
44	**PF flat time**	0,15	s	*Atleast 0,1s*
45	**PF variable step**	SAND	-	
46	**PF enforced sync.**	FALSK	-	
47	**PF force asymmetrical sim.**	FALSK	-	
48	**PF enforce P limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*
49	**PF enforce Q limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*

The `Settings` sheet contains model-specific data used by both PowerFactory and PSCAD execution.

> **CAUTION**
>
> `Projectname` should not contain spaces. The MTB replaces spaces with underscores internally, but avoiding spaces in the workbook keeps result folder and file names predictable.

1.2.1 General plant settings

case_setup.py reads the following general settings:

Setting	Used for
`Casegroup`	Selects which case sheet is used, for example `RfG`, `DCC`, `Unit`, or `Co-located`.
`Run custom cases`	Includes rows from the `Custom cases` sheet when enabled.

Setting	Used for
`Projectname`	Project/result file name prefix. Spaces are converted to underscores in code.
`Pn`	Nominal active power in MW.
`Number of Units`	Number of plant units represented in the model.
`Default P available`	Default available active power used by `Pavail0 = Default`.
`Uc`	Connection-point voltage used in impedance calculations.
`Un`	Nominal voltage. Also used to determine whether DSO-specific guide logic applies.
`Area`	Danish area, normally `DK1` or `DK2`.
`FSM deadband`	Default FSM deadband in Hz, used by guide curves and cursor metrics.
`FSM droop`	Default FSM droop in percent, used by guide curves and cursor metrics.
`Default Q(U) droop`	Default Q(U) droop used by `QUdroop0 = Default`.
`Main Transformer Grounded`	Default main transformer grounding state used by `MtrfrGnd0 = Default`.
`R0`, `X0`	Zero-sequence grid impedance values.
`Default Q mode`	Default reactive power control mode used when a case has `Qmode = Default`.
`Unit A type`	Type label for co-located Unit A.
`Unit B type`	Type label for co-located Unit B.
`Unit C type`	Type label for co-located Unit C.
`Unit D type`	Type label for co-located Unit D.

1.2.2 Grid strength settings

There are three SCR and X/R levels:

Setting	Meaning
`SCR min`, `X/R SCR min`	Minimum short-circuit level.
`SCR tuning`, `X/R SCR tuning`	Intermediate short-circuit level used for tuning cases.
`SCR max`, `X/R SCR max`	Maximum short-circuit level.

In case rows, `SCR0` and `XR0` select the initial short-circuit level and X/R ratio. A negative SCR value is used as a stiff-grid indicator in the predefined cases.

1.2.3 Timing and PowerFactory settings

Setting	Used for
`PSCAD Timestep`	PSCAD simulation time step.

Setting	Used for
`PSCAD Initialization time`	PSCAD initialization period before case time 0.
`PF flat time`	PowerFactory flat-run period before case time 0.
`PF variable step`	PowerFactory variable-step configuration.
`PF enforced sync.`	PowerFactory synchronization setting.
`PF force asymmetrical sim.`	Forces asymmetrical PowerFactory simulation where required.
`PF enforce P limits in LDF`	Enforces active power limits in PowerFactory load flow.
`PF enforce Q limits in LDF`	Enforces reactive power limits in PowerFactory load flow.

`PSCAD Initialization time` and `PF flat time` offset the simulation time before the first event. In the plotter, these offsets are removed so the first event appears at time 0.

1.2.4 Co-located settings

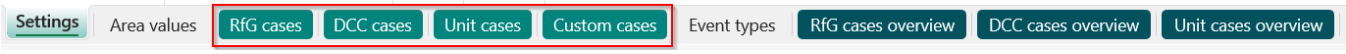
When `Casegroup = Co-located`, case_setup.py also reads co-located unit settings:

Setting	Used for
`PnG3`	Aggregated generation nominal power.
`PnD3`	Aggregated demand nominal power.
`Pn Unit A (Generation)`	Generation nominal power for Unit A.
`Default P available or SoC Unit A`	Default available power or state of charge for Unit A.
`Pn Unit A (Consumption)`	Consumption nominal power for Unit A.
`Pn Unit B (Generation)`	Generation nominal power for Unit B.
`Default P available or SoC Unit B`	Default available power or state of charge for Unit B.
`Pn Unit B (Consumption)`	Consumption nominal power for Unit B.
`Pn Unit C (Generation)`	Generation nominal power for Unit C.
`Default P available or SoC Unit C`	Default available power or state of charge for Unit C.
`Pn Unit C (Consumption)`	Consumption nominal power for Unit C.
`Pn Unit D (Generation)`	Generation nominal power for Unit D.
`Default P available or SoC Unit D`	Default available power or state of charge for Unit D.
`Pn Unit D (Consumption)`	Consumption nominal power for Unit D.

> **NOTE**
>
> Co-located case handling is present in the workbook parser, but some aggregation logic is still pending in the current implementation.

1.3 Case Sheets

The workbook contains predefined case sheets and a custom case sheet:



Each predefined case set also has an overview sheet:

Each case sheet has the same general structure:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Case				Initial Settings										
2	Rank	RMS	EMT	Name	U0	P0	Pavail0	Pmode	Qmode	Qref0	QUdroop0	SCR0	XR0	MtrfrGnd0	Simulationtime
3	1	TRUE	TRUE	RfG_SS_flatrun_Q1	1,065	0,000	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	60,00
4	2	TRUE	TRUE	RfG_SS_flatrun_Q2	1,065	0,500	Default	LFSM	Q	0,200	Default	10,000	10,000	Default	60,00
5	3	TRUE	TRUE	RfG_SS_flatrun_Q3	1,065	0,500	Default	LFSM	Q	-0,200	Default	10,000	10,000	Default	60,00
6	4	TRUE	TRUE	RfG_SS_flatrun_Q4	1,065	1,000	Default	LFSM	Q	0,330	Default	10,000	10,000	Default	60,00
7	5	TRUE	TRUE	RfG_SS_flatrun_Q5	1,065	1,000	Default	LFSM	Q	-0,330	Default	10,000	10,000	Default	60,00
8	6	TRUE	TRUE	RfG_SS_flatrun_Q6	1,065	1,000	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	60,00
9	7	TRUE	TRUE	RfG_SS_flatrun_Uctrl_min	0,968	0,700	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	60,00
10	8	TRUE	TRUE	RfG_SS_flatrun_Uctrl_max	1,118	0,700	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	60,00
11	9	TRUE	TRUE	RfG_SS_flatrun_PFctrl1	1,065	0,700	Default	LFSM	PF	0,949	Default	10,000	10,000	Default	60,00
12	10	TRUE	TRUE	RfG_SS_flatrun_PFctrl2	1,065	0,700	Default	LFSM	PF	-0,949	Default	10,000	10,000	Default	60,00
13	11	TRUE	TRUE	RfG_P_step_up_0.0_0.5	1,065	0,000	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	155,00
14	12	TRUE	TRUE	RfG_P_step_up_0.5_0.7	1,065	0,500	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	80,00
15	13	TRUE	TRUE	RfG_P_step_up_0.7_1.0	1,065	0,700	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	100,00
16	14	TRUE	TRUE	RfG_P_step_down_1.0_0.7	1,065	1,000	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	100,00
17	15	TRUE	TRUE	RfG_P_step_down_0.7_0.5	1,065	0,700	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	80,00
18	16	TRUE	TRUE	RfG_P_step_down_0.5_0.0	1,065	0,500	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	155,00
19	17	TRUE	TRUE	RfG_FSM_step1	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	60,00
20	18	TRUE	TRUE	RfG_FSM_step2	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	60,00
21	19	TRUE	TRUE	RfG_LFSM-OU_step1	1,065	0,700	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	100,00
22	20	TRUE	TRUE	RfG_LFSM-O_ramp	1,065	0,700	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	21,50
23	21	TRUE	TRUE	RfG_LFSM-U_ramp	1,065	0,700	Default	LFSM	Default	0,000	Default	10,000	10,000	Default	21,50
24	22	TRUE	TRUE	RfG_FSM_LFSM-O_ramp	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	26,50
25	23	TRUE	TRUE	RfG_FSM_LFSM-U_ramp	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	26,50
26	24	TRUE	TRUE	RfG_FSM_LFSM-O_step	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	45,00
27	25	TRUE	TRUE	RfG_FSM_LFSM-U_step	1,065	0,700	Default	LFSM+FSM	Default	0,000	Default	10,000	10,000	Default	45,00
28	26	TRUE	TRUE	RfG_LFSM-O_Pavail	1,065	1,000	0,500	LFSM	Default	0,000	Default	10,000	10,000	Default	50,00
29	27	TRUE	TRUE	RfG_LFSM-U_Pavail	1,065	0,500	0,700	LFSM	Default	0,000	Default	10,000	10,000	Default	30,00
30	28	TRUE	TRUE	RfG_U-Q/Pn_Neutral	1,065	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	100,00
31	29	TRUE	TRUE	RfG_U-Q/Pn_Neutral_stab	1,065	1,000	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	100,00
32	30	TRUE	TRUE	RfG_U-Q/Pn_underEX	1,065	1,000	Default	LFSM	Q	-0,330	Default	-1,000	0,000	Default	100,00
33	31	TRUE	TRUE	RfG_U-Q/Pn_overEX	1,065	1,000	Default	LFSM	Q	0,330	Default	-1,000	0,000	Default	100,00
34	32	TRUE	TRUE	RfG_PQ/Pn_Neutral1	1,065	1,000	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	155,00
35	33	TRUE	TRUE	RfG_PQ/Pn_Neutral2	1,065	0,500	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	155,00
36	34	TRUE	TRUE	RfG_PQ/Pn_underEX1	1,065	1,000	Default	LFSM	Q	-0,330	Default	10,000	10,000	Default	155,00
37	35	TRUE	TRUE	RfG_PQ/Pn_underEX2	1,065	0,500	Default	LFSM	Q	-0,330	Default	10,000	10,000	Default	155,00
38	36	TRUE	TRUE	RfG_PQ/Pn_overEX1	1,065	1,000	Default	LFSM	Q	0,330	Default	10,000	10,000	Default	155,00
39	37	TRUE	TRUE	RfG_PQ/Pn_overEX2	1,065	0,500	Default	LFSM	Q	0,330	Default	10,000	10,000	Default	155,00
40	38	TRUE	TRUE	RfG_Ucontrol_Scmin	1,065	1,000	Default	LFSM	Q(U)	1,065	Default	10,000	10,000	Default	85,00
41	39	TRUE	TRUE	RfG_Ucontrol_Sctune	1,065	1,000	Default	LFSM	Q(U)	1,065	Default	20,000	15,000	Default	85,00
42	40	TRUE	TRUE	RfG_Ucontrol_Scmax	1,065	1,000	Default	LFSM	Q(U)	1,065	Default	30,000	20,000	Default	85,00
43	41	TRUE	TRUE	RfG_Ucontrol_FRTsensitivity	1,000	1,000	Default	LFSM	Q(U)	1,000	Default	-1,000	0,000	Default	85,00
44	42	TRUE	TRUE	RfG_Ucontrol_FRTsensitivity-ramp	1,000	1,000	Default	LFSM	Q(U)	1,000	Default	-1,000	0,000	Default	20,00
45	43	TRUE	TRUE	RfG_Uarea_Qreq_1(up)	1,065	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	50,00
46	44	TRUE	TRUE	RfG_Uarea_Qreq_2(up)	1,118	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	50,00
47	45	TRUE	TRUE	RfG_Uarea_Qreq_1(down)	1,065	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	50,00
48	46	TRUE	TRUE	RfG_Uarea_Qreq_2(down)	0,968	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	50,00
49	47	TRUE	TRUE	RfG_Uarea_Qreq_3(down)	0,900	1,000	Default	LFSM	Q	0,000	Default	-1,000	0,000	Default	50,00
50	48	TRUE	TRUE	RfG_Uarea_Q(U)req_1(up)	1,065	1,000	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	50,00
51	49	TRUE	TRUE	RfG_Uarea_Q(U)req_2(up)	1,118	1,000	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	50,00
52	50	TRUE	TRUE	RfG_Uarea_Q(U)req_1(down)	1,065	1,000	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	50,00
53	51	TRUE	TRUE	RfG_Uarea_Q(U)req_2(down)	0,968	1,000	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	50,00
54	52	TRUE	TRUE	RfG_Uarea_Q(U)req_3(down)	0,900	1,000	Default	LFSM	Q(U)	1,065	Default	-1,000	0,000	Default	50,00
55	53	TRUE	TRUE	RfG_Q_control_up	1,065	1,000	Default	LFSM	Q	0,000	Default	10,000	10,000	Default	120,00

1.3.1 Case basic information

Each row represents one case.

Column	Used for
`Rank`	Unique case rank number.
`RMS`	Whether the case is run in PowerFactory.
`EMT`	Whether the case is run in PSCAD.
`Name`	Case name used in generated study cases, result files, and plot titles.

1.3.2 Case initial settings

For normal RfG, DCC, and Unit cases, case_setup.py reads these initial settings:

Column	Used for
`U0`	Initial voltage in pu.
`P0`	Initial active power reference in pu.
`Pavail0`	Initial available active power in pu, or `Default` to use `Default P available`.
`Pmode`	Initial active power control mode.
`Qmode`	Initial reactive power control mode, or `Default` to use `Default Q mode`.
`Qref0`	Initial reactive power, voltage, or power factor reference, depending on `Qmode`.
`QUdroop0`	Initial Q(U) droop in percent, or `Default` to use `Default Q(U) droop`.
`SCR0`	Initial SCR.
`XR0`	Initial X/R ratio.
`MtrfrGnd0`	Initial main transformer grounding state: `Default`, `Grounded`, or ungrounded/not grounded.
`Simulationtime`	Case simulation time excluding initialization/flat time. If blank or 0 for recording cases, it can be derived from recording length.

For co-located cases, the parser reads per-unit values instead of the normal `P0` and `Pavail0` fields:

Column group	Used for
`P0_unitA` to `P0_unitD`	Initial active power per unit.
`Pavail0_unitA` to `Pavail0_unitD`	Initial available power or state of charge per unit.

1.3.3 Pmode and Qmode values

The accepted `Pmode` values are defined by `PMODES` in case_setup.py:

Pmode	MTB value
`No P(f)`	0
`LFSM`	1
`FSM`	2
`LFSM+FSM`	3
`Pmode4`	4
`Pmode5`	5
`Pmode6`	6
`Pmode7`	7

The accepted `Qmode` values are defined by `QMODES`:

Qmode	MTB value
`Q`	0
`Q(U)`	1
`PF`	2
`Qmode3`	3
`Qmode4`	4
`Qmode5`	5
`Qmode6`	6

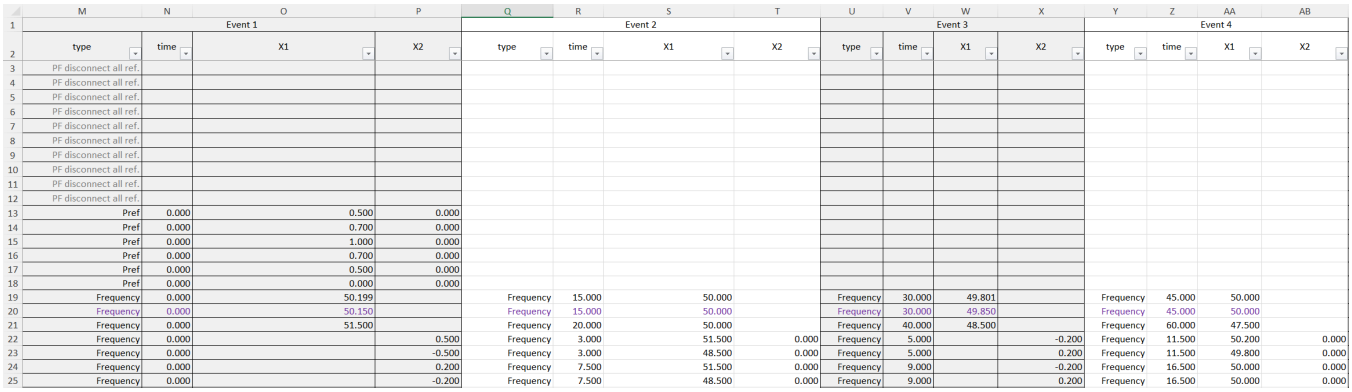
`Qref0` and `Qref` event values are interpreted according to the active Q mode:

Qmode	Meaning of `Qref`
`Q`	Reactive power reference in pu at PoC.
`Q(U)`	Voltage reference in pu at PoC.
`PF`	Power factor reference at PoC.
`Qmode3` to `Qmode6`	Custom Q-mode reference.

1.3.4 Case events

Each case can define a sequence of events. In the current workbook, most sheets include Event 1 through Event 12, while `RfG cases` and `Co-located cases` include Event 1 through Event 13. Every event has four fields:

Field	Used for
`type`	Event type.
`time`	Event time relative to the end of PSCAD initialization time or PF flat time.
`X1`	First event argument. Meaning depends on `type`.
`X2`	Second event argument. Meaning depends on `type`.



	Event 1				Event 2				Event 3				Event 4			
	type	time	X1	X2	type	time	X1	X2	type	time	X1	X2	type	time	X1	X2
3	PF disconnect all ref.															
4	PF disconnect all ref.															
5	PF disconnect all ref.															
6	PF disconnect all ref.															
7	PF disconnect all ref.															
8	PF disconnect all ref.															
9	PF disconnect all ref.															
10	PF disconnect all ref.															
11	PF disconnect all ref.															
12	PF disconnect all ref.															
13	Pref	0.000	0.500	0.000												
14	Pref	0.000	0.700	0.000												
15	Pref	0.000	1.000	0.000												
16	Pref	0.000	0.700	0.000												
17	Pref	0.000	0.500	0.000												
18	Pref	0.000	0.000	0.000												
19	Frequency	0.000	50.199		Frequency	15.000	50.000		Frequency	30.000	49.801		Frequency	45.000	50.000	
20	Frequency	0.000	50.150		Frequency	15.000	50.000		Frequency	30.000	49.850		Frequency	45.000	50.000	
21	Frequency	0.000	51.500		Frequency	20.000	50.000		Frequency	40.000	48.500		Frequency	60.000	47.500	
22	Frequency	0.000		0.500	Frequency	3.000	51.500	0.000	Frequency	5.000		-0.200	Frequency	11.500	50.200	0.000
23	Frequency	0.000		-0.500	Frequency	3.000	48.500	0.000	Frequency	5.000		0.200	Frequency	11.500	49.800	0.000
24	Frequency	0.000		0.200	Frequency	7.500	51.500	0.000	Frequency	9.000		-0.200	Frequency	16.500	50.000	0.000
25	Frequency	0.000		-0.200	Frequency	7.500	48.500	0.000	Frequency	9.000		0.200	Frequency	16.500	50.000	0.000

The current parser supports these event types:

Event type	X1	X2
`Pref`	New active power reference in pu based on `Pn`.	Gradient in 1/s.
`Pavail`	New available active power limit in pu based on `Pn`.	Gradient in 1/s.
`Qref`	New Q, Q(U), PF, or custom Q-mode reference.	Gradient in 1/s.
`QUdroop`	New Q(U) droop in percent.	Ignored; set to 0 in code.
`Voltage`	New Thevenin grid voltage in pu based on `Un`.	Gradient in 1/s.
`dVoltage`	Delta change in Thevenin grid voltage in pu based on `Un`.	Gradient in 1/s.
`Phase`	New Thevenin grid voltage phase in degrees.	Gradient in 1/s.
`Frequency`	New Thevenin grid frequency in Hz.	Gradient in 1/s.
`SCR`	New short-circuit ratio.	New X/R ratio.
`3p fault`	Three phase fault; residual voltage (across fault reactance), base Un.	Fault time in seconds.
`2p-g fault`	Two phase to ground fault; residual voltage (across fault reactance), base Un.	Fault time in seconds.
`2p fault`	Two phase fault; residual voltage (across fault reactance), base Un.	Fault time in seconds.
`1p fault`	One phase fault; residual voltage (across fault reactance), base Un.	Fault time in seconds.
`3p fault (ohm)`, `2p-g fault (ohm)`, `2p fault (ohm)`, `1p fault (ohm)`	fault resistance.	fault reactance / fault time as configured by the fault implementation.
`Clear fault`	Clear any fault.	Ignored.
`Pref recording`	Relative path to measurement file used for `Pref`.	Scaling factor.
`Qref recording`	Relative path to measurement file used for `Qref`.	Scaling factor.
`Voltage recording`	Relative path to measurement file used for Thevenin grid voltage.	Scaling factor.
`Inst. Voltage recording`	Relative path to instantaneous three-phase voltage recording.	Scaling factor.

Event type	X1	X2
`Phase recording`	Relative path to measurement file used for Thevenin grid voltage phase.	Scaling factor.
`Frequency recording`	Relative path to measurement file used for Thevenin grid frequency.	Scaling factor.
`SIPS Generation`	SIPS generation integer command.	Ignored; set to 0 in code.
`SIPS Demand`	SIPS demand integer command.	Ignored; set to 0 in code.
`Signal 1` to `Signal 10`	New custom signal value.	Gradient in 1/s.
`Signal 1 recording` to `Signal 10 recording`	Relative path to measurement file used for the custom signal.	Scaling factor.
`PF disconnect all ref.`	Ignored.	Ignored.
`PF force asymmetrical`	Ignored.	Ignored.

The custom event names in the workbook are written explicitly as `Signal 1`, `Signal 2`, `Signal 3`, `Signal 4`, `Signal 5`, `Signal 6`, `Signal 7`, `Signal 8`, `Signal 9`, and `Signal 10`, with equivalent recording event names `Signal 1 recording`, `Signal 2 recording`, `Signal 3 recording`, `Signal 4 recording`, `Signal 5 recording`, `Signal 6 recording`, `Signal 7 recording`, `Signal 8 recording`, `Signal 9 recording`, and `Signal 10 recording`.

The `Event types` sheet in the workbook contains the user-facing event descriptions:

Name	Description	X1 description	X1 unit	X2 description	X2 unit
Pref	Pref change	setpoint, base Pn	pu	gradient	1/s
Pref_unitA	Pref change (unit A, Co-Located)	setpoint (base PnG3 or PnD3)	(pu)	gradient	1/s
Pref_unitB	Pref change (unit B, Co-Located)	setpoint (base PnG3 or PnD3)	(pu)	gradient	1/s
Pref_unitC	Pref change (unit C, Co-Located)	setpoint (base PnG3 or PnD3)	(pu)	gradient	1/s
Pref_unitD	Pref change (unit D, Co-Located)	setpoint (base PnG3 or PnD3)	(pu)	gradient	1/s
Pavail	Maximum power available change	maximum limit, base Pn	pu	gradient	1/s
Pavail_unitA	Maximum power available change (unit A, Co-Located)	maximum limit (base PnG3 or PnD3)	(pu)	gradient	1/s
Pavail_unitB	Maximum power available change (unit B, Co-Located)	maximum limit (base PnG3 or PnD3)	(pu)	gradient	1/s
Pavail_unitC	Maximum power available change (unit C, Co-Located)	maximum limit (base PnG3 or PnD3)	(pu)	gradient	1/s
Pavail_unitD	Maximum power available change (unit D, Co-Located)	maximum limit (base PnG3 or PnD3)	(pu)	gradient	1/s
Qref	Qref change	setpoint, base Pn	pu	gradient	1/s
Qudroop	Q(U) droop change	Q-U droop	%	-	-
Voltage	Grid voltage change (thevenin)	setpoint, base Un	pu	gradient	1/s
dVoltage	Delta grid voltage change (thevenin)	setpoint, base Un	pu	gradient	1/s
Phase	Grid phase change (thevenin)	setpoint	degree	gradient	degree/s
Frequency	Grid frequency change	setpoint	Hz	gradient	Hz/s
SCR	Grid impedance change	SCR	-	X/R	-
3p fault	Three phase fault	residual voltage (across fault reactance), base Un	pu	fault time	s
2p-g fault	Two phase to ground fault	residual voltage (across fault reactance), base Un	pu	fault time	s
2p fault	Two phase fault	residual voltage (across fault reactance), base Un	pu	fault time	s
1p fault	One phase fault	residual voltage (across fault reactance), base Un	pu	fault time	s
3p fault (ohm)	Three phase fault	fault resistance	ohm	fault reactance	ohm
2p-g fault (ohm)	Two phase to ground fault	fault resistance	ohm	fault reactance	ohm
2p fault (ohm)	Two phase fault	fault resistance	ohm	fault reactance	ohm
1p fault (ohm)	One phase fault	fault resistance	ohm	fault reactance	ohm
Clear fault	Clear any fault	N/A	N/A	N/A	N/A
Pref recording	Use external measurement file for Pref signal. First value hold until flattime or initialtime has elapsed.	Relative path to measurementfile	N/A	Scaling factor	-
Qref recording	Use external measurement file for Qref signal. First value hold until flattime or initialtime has elapsed.	Relative path to measurementfile	N/A	Scaling factor	-
Voltage recording	Use external measurement file for the thevenin voltage signal (RMS). First value hold until flattime or initialtime has elapsed.	Relative path to measurementfile	N/A	Scaling factor	-
Inst. Voltage recording	Only supported for PSCAD. Use external measurement file for the thevenin instantaneous voltages. Time value ignored.	Relative path to measurementfile	N/A	Scaling factor	-
Phase recording	Use external measurement file for the thevenin voltage phase signal. First value hold until flattime or initialtime has elapsed.	Relative path to measurementfile	N/A	Scaling factor	-
Frequency recording	Use external measurement file for the thevenin voltage frequency signal. First value hold until flattime or initialtime has elapsed.	Relative path to measurementfile	N/A	Scaling factor	-
SIPS	System Integrity Protection Scheme (SIPS) real signal The real signal will be converted into binary equivalent signal using the MTB SIPS Decoder blocks in PSCAD and PowerFactory	equivalent binary value	N/A	-	-
Signal 1	Custom signal 1 change (Use t < 0s to initialize)	value	?	gradient	?/s

1.4 Example Case Setup

Case rank 86 is used here as an example. This case simulates a three-phase fault and a change in SCR when the fault is cleared, representing for example a faulty line being disconnected.

	Case				Initial Settings							
Rank	RMS	EMT	Name	U0	P0	Pmode	Qmode	Qref0	SCR0	XR0	Simulationtime
86	TRUE	TRUE	ION_RfG_Fault_3_SCR	1.065	1.000	LFSM	Default	0.000	20.000	15.000	15.00

In this example:

- The case is conditionally active depending on whether the plant is a DSO connection.
- The case is run for both RMS and EMT.
- The case name is `ION_RfG_Fault_3_SCR`.
- `U0` is based on `Uc` from the `Settings` sheet.
- `P0` initializes the active power reference at 1.0 pu.
- `Pmode` initializes LFSM as the active power control mode.
- `Qmode` is set to the default control mode from the `Settings` sheet.
- `Qref0` is set to 0.
- `SCR0` and `XR0` use the tuning values from the `Settings` sheet.
- `Simulationtime` is 15 seconds, excluding the PF flat time or PSCAD initialization time.

Event 1				Event 2			
type	time	X1	X2	type	time	X1	X2
3p fault	0.000	5%	0.150	SCR	0.149	10.000	10.000

The case consists of two events:

1. Event 1 is a `3p fault` at time 0. `X1 = 0.05` gives a 5% residual voltage at PoC, and `X2 = 0.150` gives a 150 ms fault duration.
2. Event 2 is an `SCR` change at time 0.149. `X1` is the new SCR value and `X2` is the new X/R ratio.

The simulation then continues until the configured simulation time has elapsed.

---

2. Quickstart PowerFactory Guide

2.1 System Requirements

The MTB execute_pf.py script executes RMS test cases in DIgSILENT PowerFactory. The current version is developed for PowerFactory 2025 SP4 with Python 3.11.

Install the MTB Python dependencies from the main MTB folder:

```
python -m pip install -r requirements.txt
```

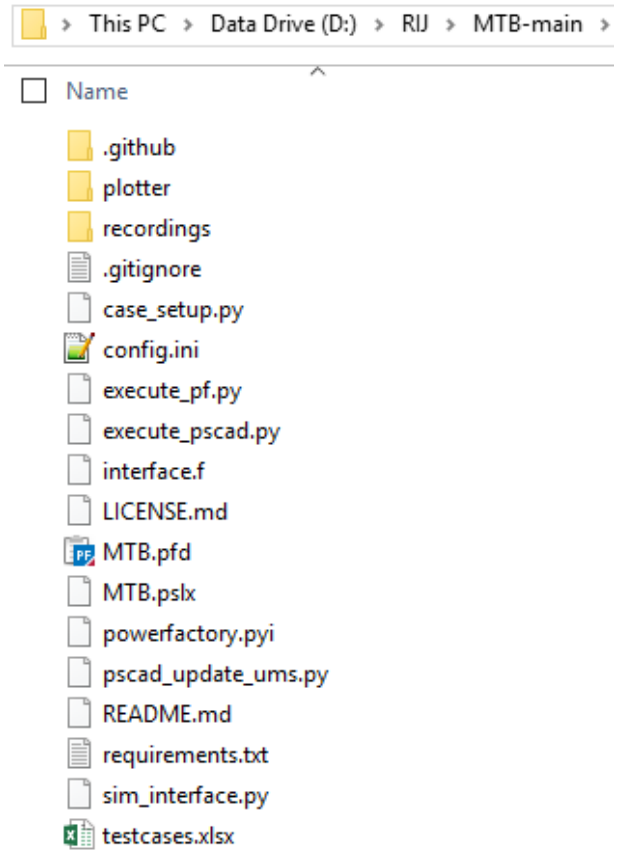
> **NOTE**
>
> An example PowerFactory setup is available in setup*examples/MTB*Setup_Example.pfd. The example demonstrates the MTB setup only; it is not representative of a compliant plant model.

> **IMPORTANT**
>
> The PowerFactory example was tested by an external third party with these changes to the default values in testcases.xlsx: `Pn = 17.00 MW`, `Uc = 50.00 kV`, and `Un = 50.00 kV`.

2.2 Preparation

2.2.1 Create an MTB installation folder

Extract or copy the MTB files into a common folder on the PC.



2.2.2 Edit the Settings sheet

Fill in the model-specific information in the `Settings` sheet in testcases.xlsx. The workbook guide is described in 1. Quickstart Testcases Workbook Guide.

	A	B	C	D
1	**Name**	**Value**	**Unit**	**Comment**
2	**Casegroup**	RfG	-	
3	**Run custom cases**	FALSK	-	*If custom not selected for casegroup.*
4	**Projectname**	Solbakken	-	
5	**Pn**	198,90	MW	
6	**Default P available**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed*
7	**Number of Units**	4		
8	**PnGS**	200,00	MW	*Allowed generation capacity in POC*
9	**PnDS**	-100,00	MW	*Allowed consumption capacity in POC*
10	**Unit A type**	Generation		
11	**Pn Unit A (Generation)**	250,00	MW	*Nominal power (Generation)*
12	**Default P available or SoC Unit A**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
13	**Pn Unit A (Consumption)**	0,00	MW	*Nominal power (Consumption)*
14	**Unit B type**	Energy Storage		
15	**Pn Unit B (Generation)**	100,00	MW	*Nominal power (Generation)*
16	**Default P available or SoC Unit B**	0,50	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
17	**Pn Unit B (Consumption)**	-100,00	MW	*Nominal power (Consumption)*
18	**Unit C type**	Demand		
19	**Pn Unit C (Generation)**	0,00	MW	*Nominal power (Generation)*
20	**Default P available or SoC Unit C**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
21	**Pn Unit C (Consumption)**	-10,00	MW	*Nominal power (Consumption)*
22	**Unit D type**	Generation		
23	**Pn Unit D (Generation)**	10,00	MW	*Nominal power (Generation)*
24	**Default P available or SoC Unit D**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
25	**Pn Unit D (Consumption)**	0,00	MW	*Nominal power (Consumption)*
26	**Uc**	161,90	kV	*Used for easy setup to make sure simulations, start in normal operation voltage and not 1 pu.*
27	**Un**	152,00	kV	*This will be 1pu in simulations and results*
28	**Area**	DK1		
29	**FSM deadband**	0,05	Hz	*If any*
30	**FSM droop**	2,0	%	
31	**SCR min**	5,0	-	
32	**SCR tuning**	23,7	-	
33	**SCR max**	42,4	-	
34	**Default Q(U) droop**	4,0	%	
35	**X/R SCR min**	8,10	-	
36	**X/R SCR tuning**	16,90	-	
37	**X/R SCR max**	25,60	-	
38	**Main Transformer Grounded**	FALSK	-	*Default grounding setting for the Main Transformer*
39	**R0**	3,92	ohm	*Grounding impedance of the MTB voltage source*
40	**X0**	18,30	ohm	*Grounding impedance of the MTB voltage source*
41	**Default Q mode**	Q(U)	-	*Qmode signal from MTB is constant. Qref mode == 0, Q(U) mode == 1 and Qpf == 2.*
42	**PSCAD Timestep**	5	µs	
43	**PSCAD Initialization time**	3,55	s	*Atleast 1s*
44	**PF flat time**	0,15	s	*Atleast 0,1s*
45	**PF variable step**	SAND	-	
46	**PF enforced sync.**	FALSK	-	
47	**PF force asymmetrical sim.**	FALSK	-	
48	**PF enforce P limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*
49	**PF enforce Q limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*

Important MTB 2.x settings include:

- **Default P available**: associated with `mtb_s_pavail_pu`, the `Pavail` event type, and `Pavail0`.
- **Default Q(U) droop**: associated with `mtb_s_qudroop`, the `QUdroop` event type, and `QUdroop0`.
- **Main Transformer Grounded**: associated with `mtb_s_mtrfrgnd` and `MtrfrGnd0`.

2.2.3 Edit config.ini

Edit config.ini if required.

```
[General]
# Test case sheet path

Casesheet path = .\testcases.xlsx
# Path to export result files (relative to execute.py)
Export folder = export

[Python]
# Optional path to append to the python path
Python path =
```

1 2

.
.
.

3

```
[PowerFactory]
# PowerFactory parallel task automation.
Parallel = True

# As per 2023 SP5 there is a bug in PF that causes QDSL blocks to be ignored in parallel simulations.
# PowerFactory temporary workaround: QDSL controller sometimes fails when not in same grid as calc. relevant statgens.
# The QDSL controller will be copied to the following grid. Disable by setting to empty string:
QDSL copy grid =

# Create MTB_<Casegroup> subfolders for generated study cases and variations.
# Set to True to create and use the folders for cases and variations, or False to keep the current structure.
create_case_folder = False
```

The PowerFactory execution script reads the following settings:

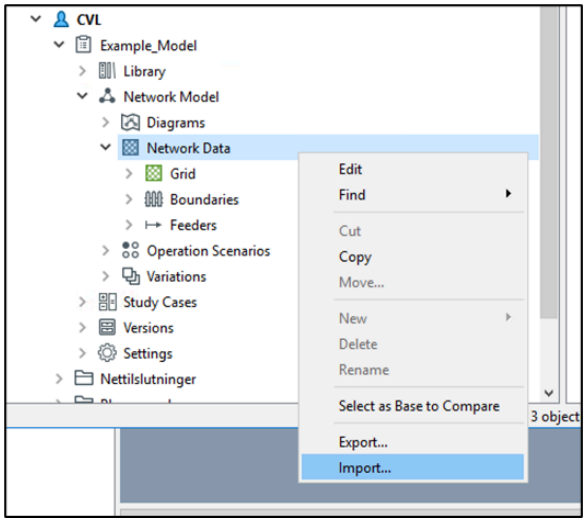
Section	Setting	Default / fallback	Used for
`[General]`	`Casesheet path`	`testcases.xlsx`	Path to the `testcases.xlsx` file used for PowerFactory execution.
`[General]`	`Export folder`	`export`	Folder where PowerFactory CSV result files are exported.
`[Python]`	`Python path`	no fallback	Additional path appended to `sys.path`, typically used to make the PowerFactory Python module available.
`[PowerFactory]`	`Parallel`	`True`	Enables PowerFactory task automation parallel execution.
`[PowerFactory]`	`QDSL copy grid`	blank	Optional workaround grid where `initializer_qdsl` is copied if QDSL execution fails from the MTB grid. Leave blank to disable.
`[PowerFactory]`	`create_case_folder`	`False`	Creates `MTB_<Casegroup>` subfolders for generated study cases and variations when `True`.

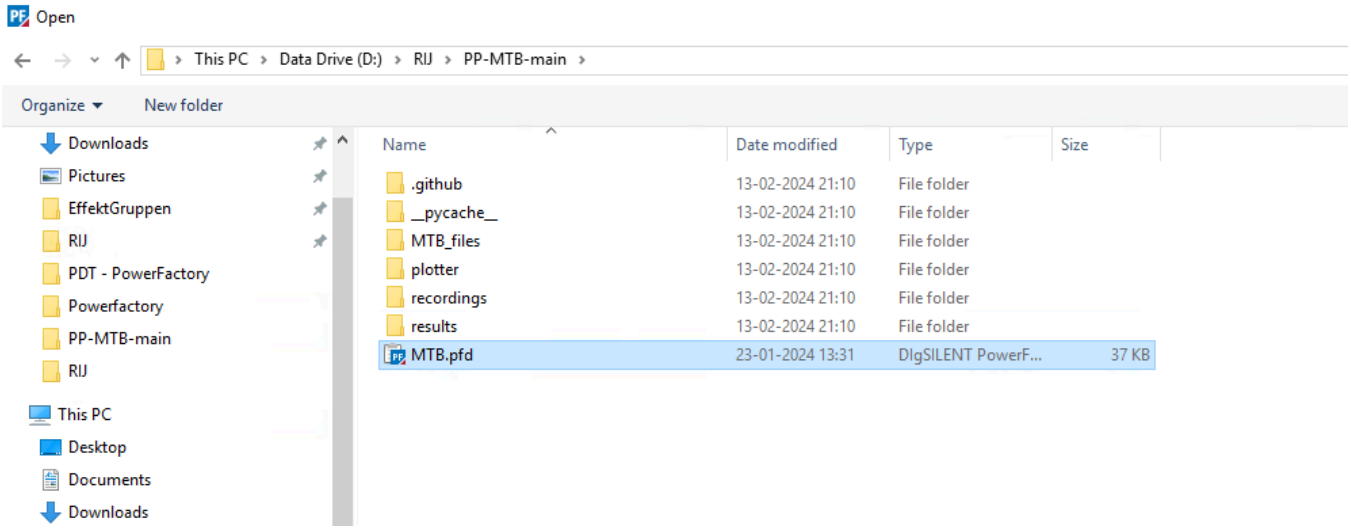
2.3 Model Setup in PowerFactory

2.3.1 Import the MTB into the project

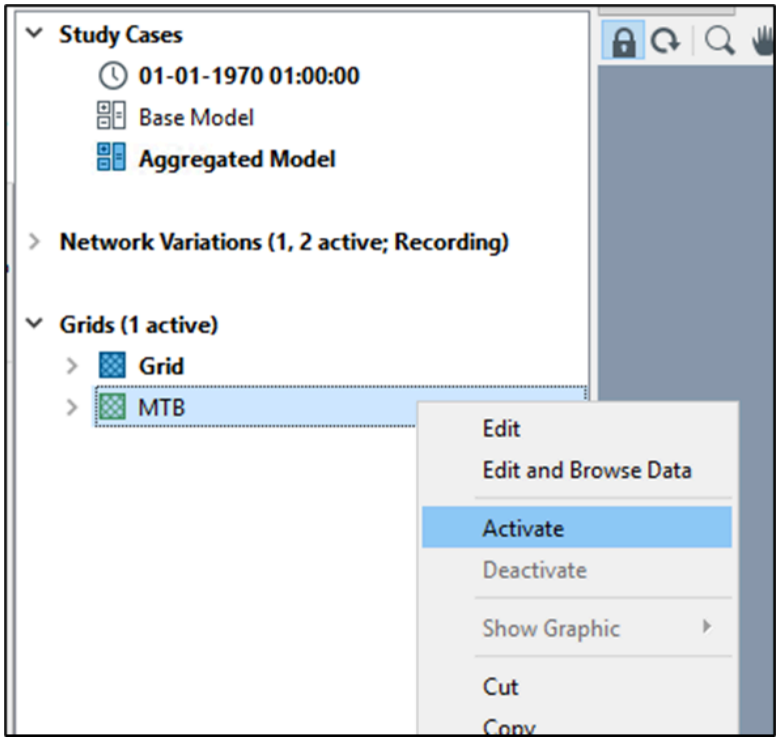
Right-click the **Network Data** folder in the PowerFactory model and import MTB.pfd from the MTB folder. The project must be inactive to import using this method.

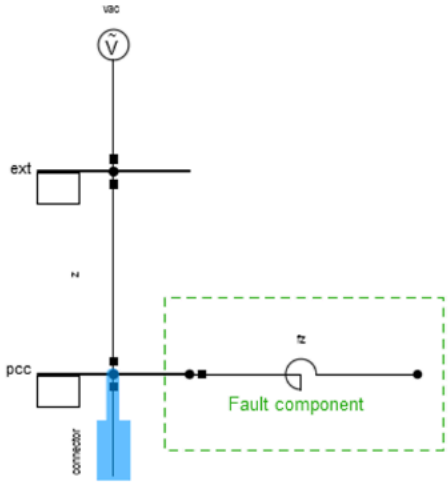
> **IMPORTANT**
>
> The MTB grid naming must remain `MTB`.

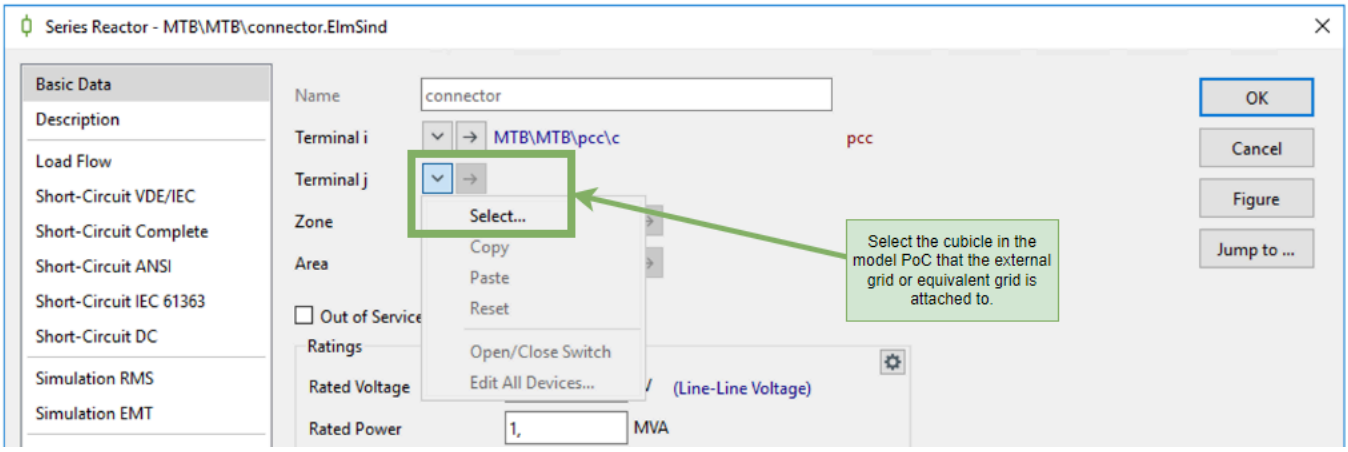


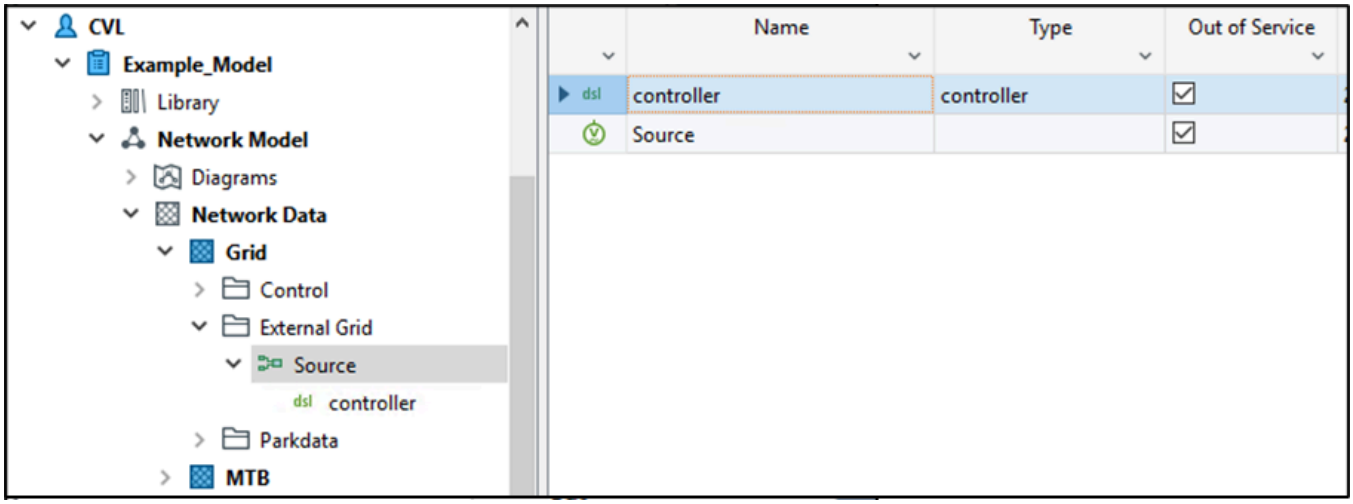


Activate the MTB grid and replace the external grid with MTB. Remember to deactivate the existing external grid and any linked dynamic controls used for the external grid by marking them as **Out of Service**.

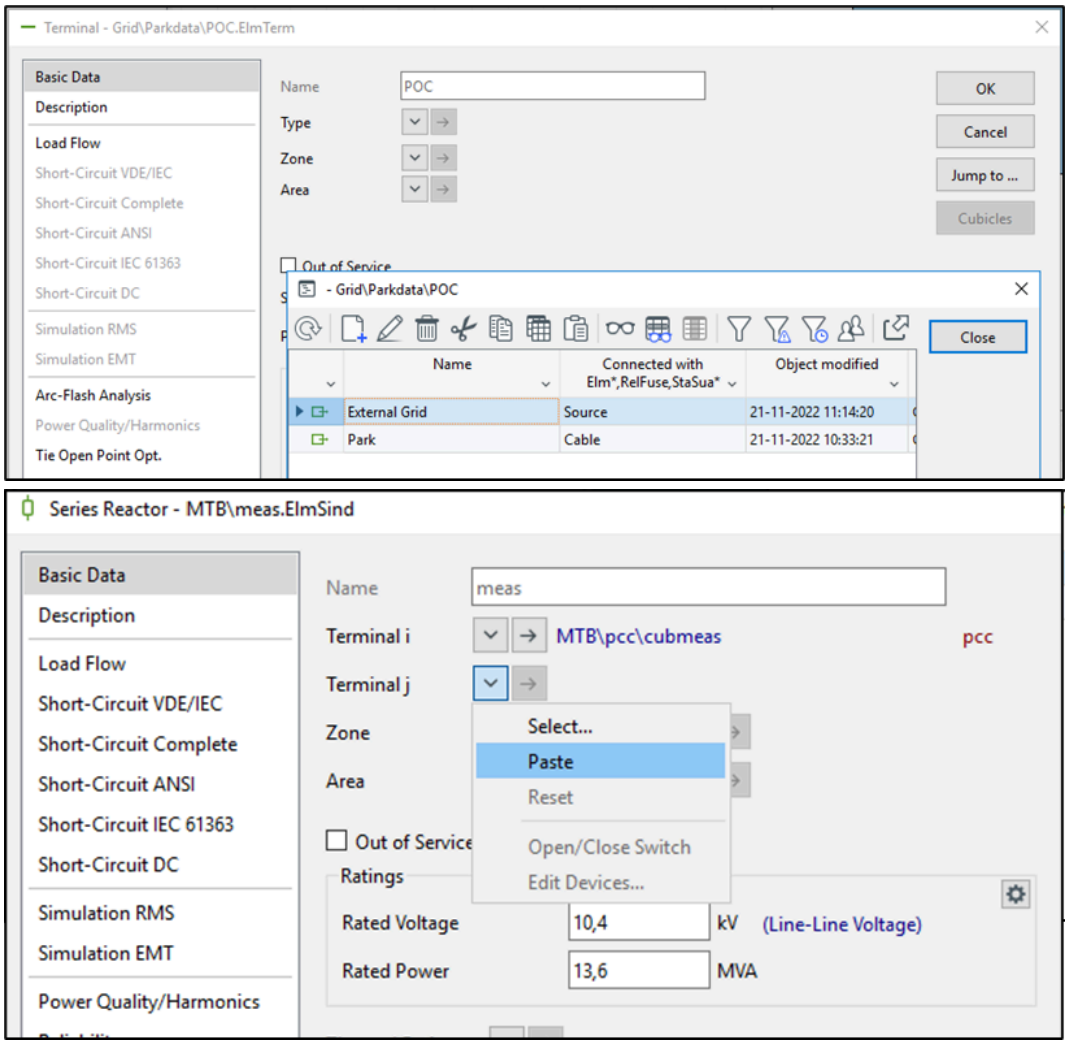

Study Cases
01-01-1970 01:00:00
Base Model
Aggregated Model
Network Variations (1, 2 active; Recording)
Grids (1 active)
Grid
MTB
Edit
Edit and Browse Data
Activate
Deactivate
Show Graphic
Cut
Copy


vac
ext
pcc
connector
Fault component


Series Reactor - MTB\MTB\connector.ElmSind
Basic Data
Description
Load Flow
Short-Circuit VDE/IEC
Short-Circuit Complete
Short-Circuit ANSI
Short-Circuit IEC 61363
Short-Circuit DC
Simulation RMS
Simulation EMT
Name
connector
Terminal i
MTB\MTB\pcc\c
pcc
Terminal j
Zone
Area
Out of Service
Ratings
Rated Voltage
(Line-Line Voltage)
Rated Power
1,
MVA
Select...
Copy
Paste
Reset
Open/Close Switch
Edit All Devices...
Select the cubicle in the model PoC that the external grid or equivalent grid is attached to.
OK
Cancel
Figure
Jump to ...



Connect the cubicle formerly used for the external grid at PoC to `meas.ElmSind`, Terminal j. Copy the cubicle from the busbar used by the external grid and paste it to `meas.ElmSind`.



2.3.2 Connect the MTB to the plant controller

There are two main ways to connect the MTB to the model's Power Plant Controller (PPC):

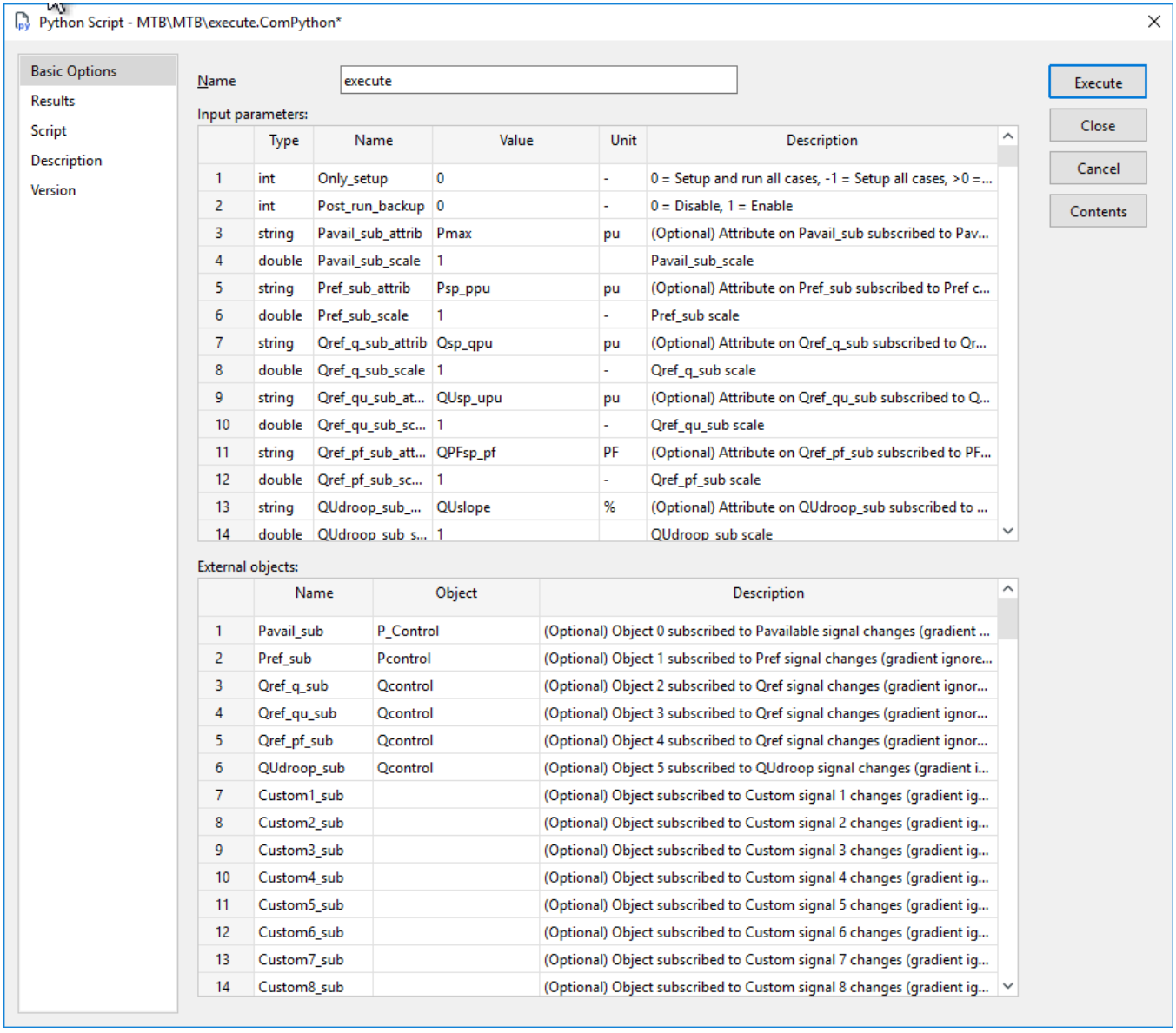
1. Connect references through the `execute.ComPython` script, where reference signals and model parameters are linked through script input parameters and external objects.

2. Assign the PPC P-control and Q-control block types to the MTB frame slots and connect MTB signals graphically through the MTB composite frame.

> **NOTE**
>
> The `execute.ComPython` script also contains relevant execution parameters. `Only_setup` can be used to set up one selected case without running all cases, and `Post_run_backup` creates a project version after the MTB run.

2.3.2.1 MTB connection through execute.ComPython

1. Open `execute.ComPython` and add the active and reactive PPC controller signal names to the relevant rows in the input parameter table.
2. Add the PPM and PPC controller block instances responsible for active and reactive power reference control. Depending on the plant model, this can be one or several blocks.



For the example shown above:

- **Pmax** is the attribute of the PPM control block that controls maximum available active power.

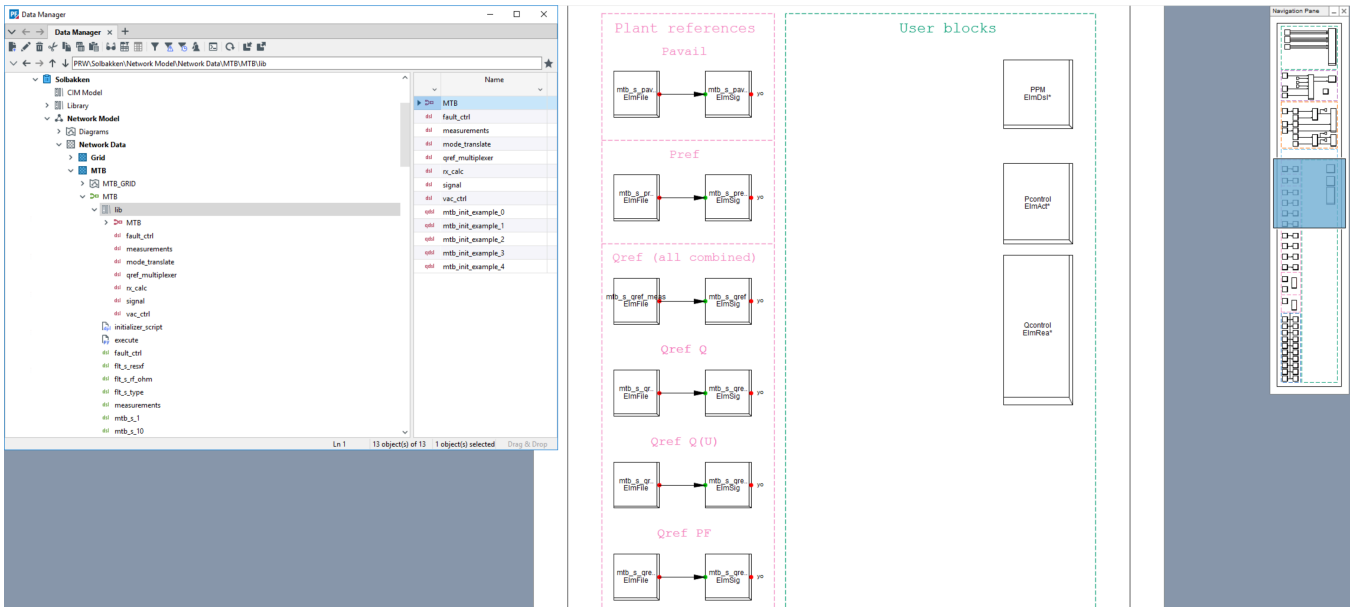
- **Psp_ppu** is the attribute of the active power control block that controls the active power reference in pu.
- **Qsp_qpu** is the attribute of the reactive power control block for Q control mode.
- **QUsp_upu** is the attribute of the reactive power control block for Q(U) voltage reference control.
- **QPFsp_pf** is the attribute of the reactive power control block for PF control mode.
- **QUslope** is the attribute of the reactive power control block for Q(U) droop in percent.

> **IMPORTANT**
>
> When using predefined case sets from testcases.xlsx, the SIPS event interface can only be set up through the MTB frame. This requires the plant SIPS interface to be available as signals. Alternatively, custom signals can be used.

> **IMPORTANT**
>
> Some PPCs use a reactive power base equal to 0.33 times the active power base, in accordance with Danish grid-code requirements. In those cases, the MTB reactive power reference must be scaled by approximately 3.03.

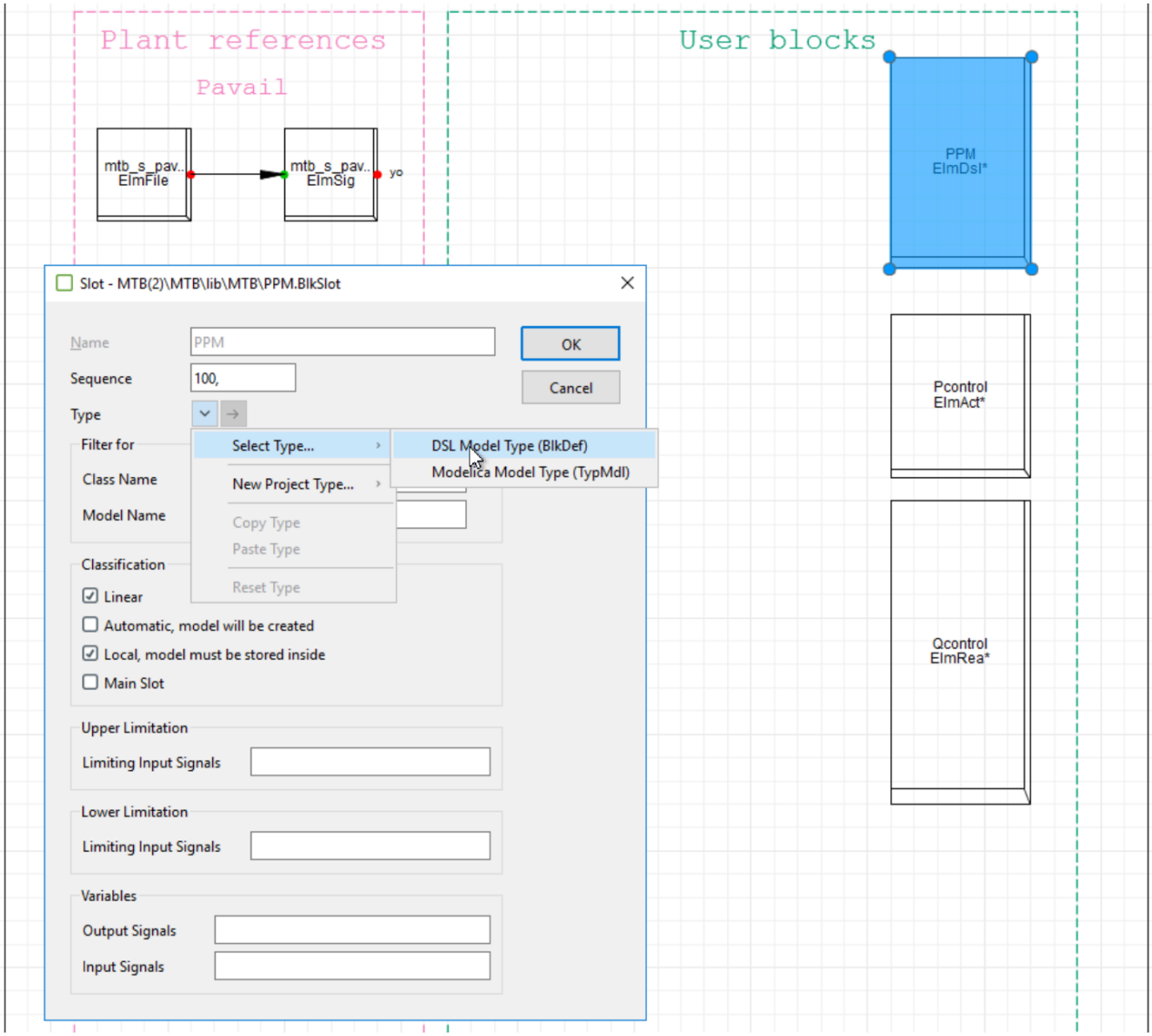
2.3.2.2 MTB connection through the MTB frame

1. Open the MTB composite frame type, for example by right-clicking it and selecting **Mark in Graphic**. Scroll to the **User blocks** area and insert the PPC active and reactive power control block types.



> **IMPORTANT**
>
> This setup method requires the plant references to be modeled as signals, not only as internal parameters in the PPC P-control and Q-control blocks.

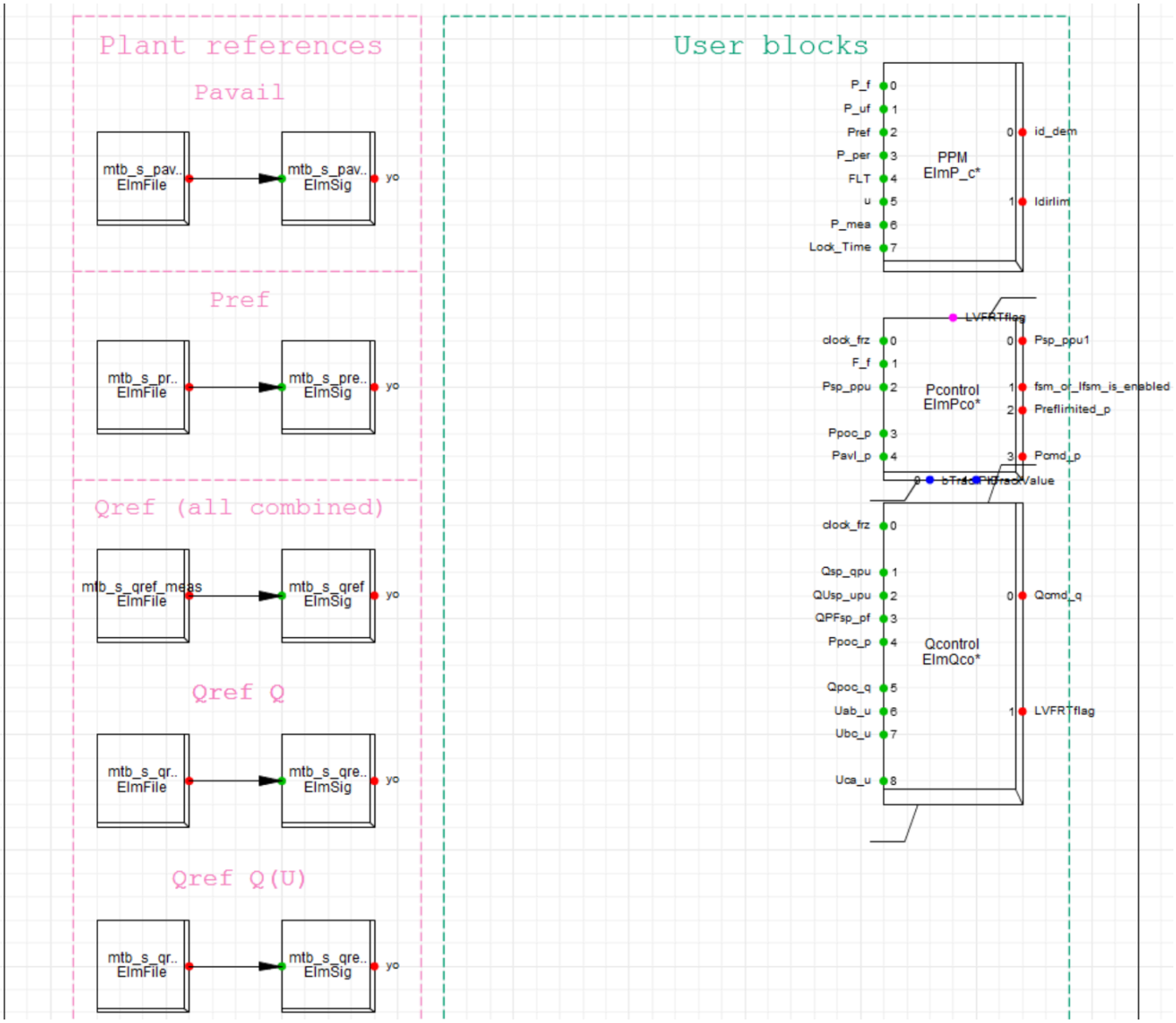
1. Select the DSL model type for the MTB slot by editing the slot and selecting the relevant type from the project library.



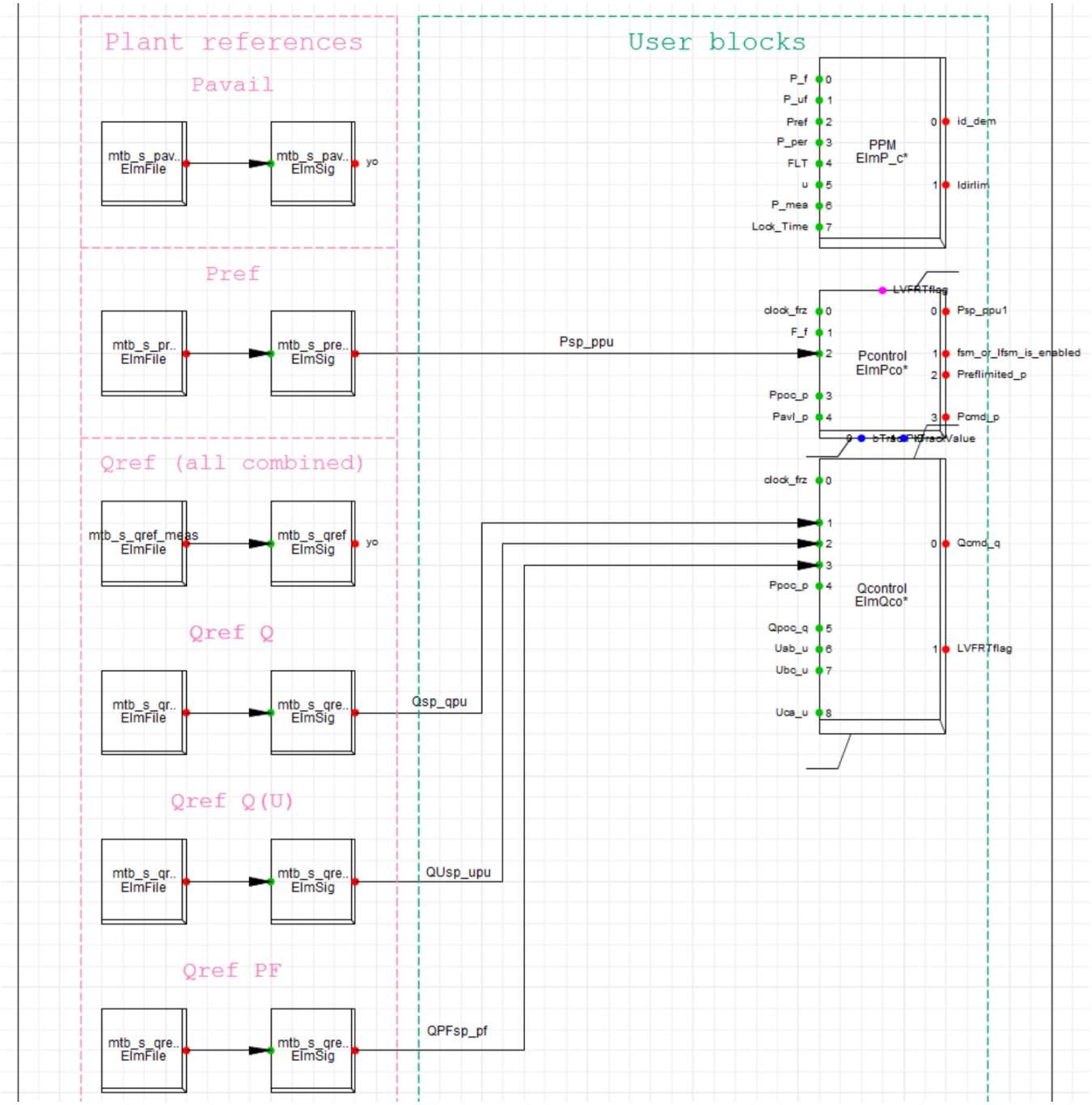
NOTE

DSL instances are coloured green, whereas DSL library types are coloured red.

1. Repeat this setup for the PPM active power controller and the PPC active and reactive power controllers.



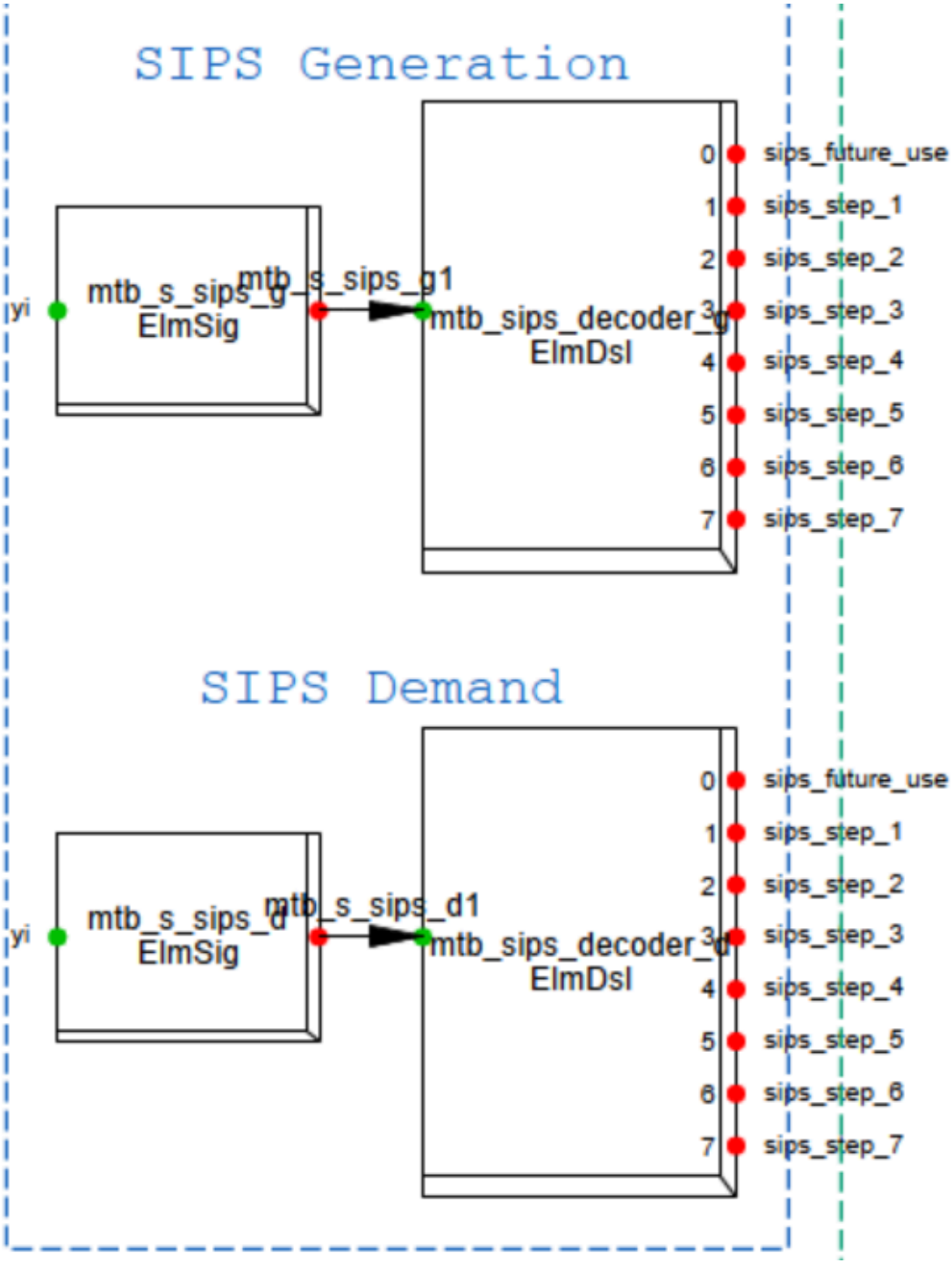
1. Wire the MTB reference signals `Pref`, `Qref`, `Uref`, and `PF` to the corresponding PPC controller inputs. If possible, also wire `Pavail` and `Q(U)droop` as signals.



IMPORTANT

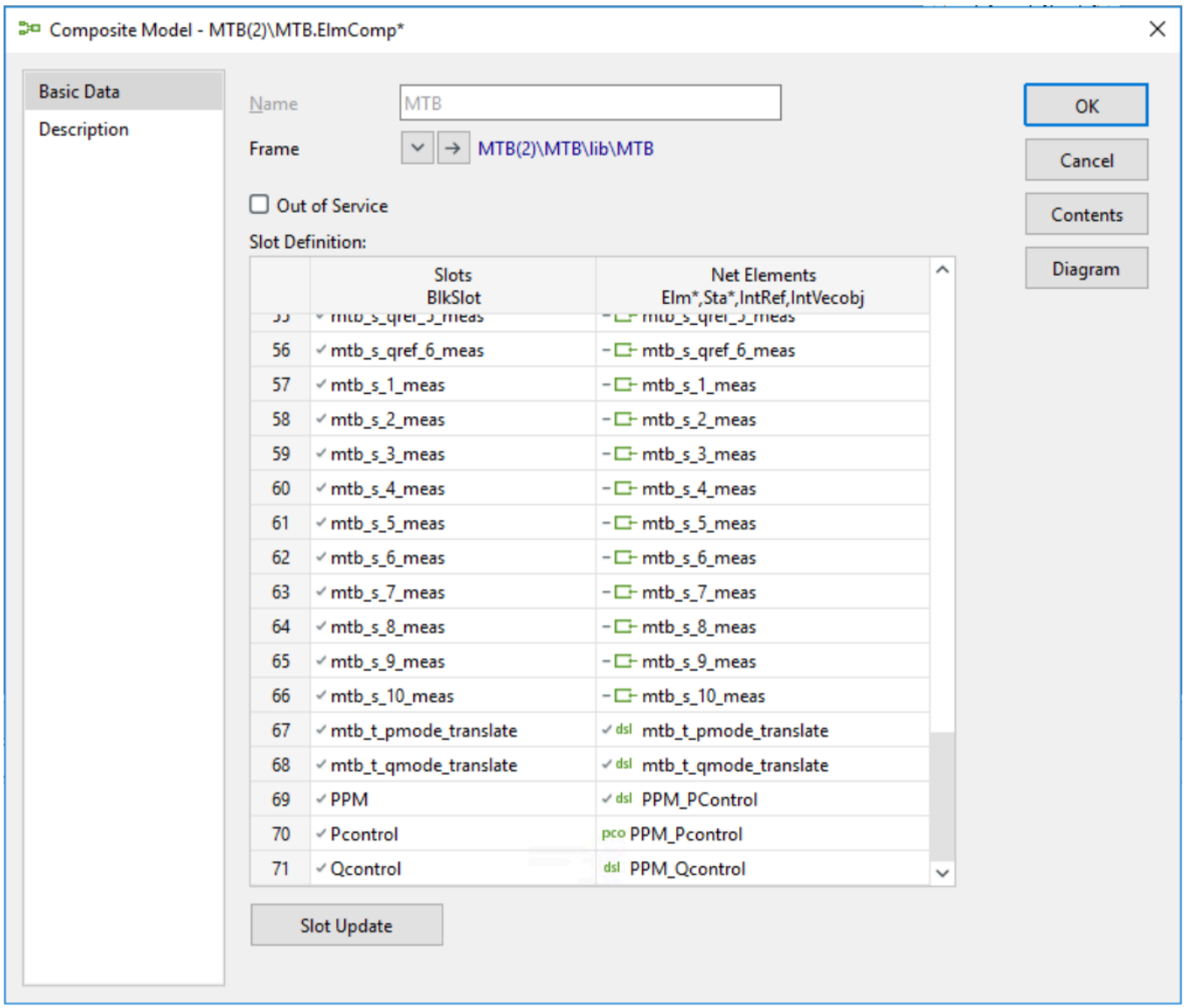
If `Pavail` or `Q(U)droop` are parameters rather than signals, initialize them through `initializer_script.ComDpl` or through `execute.ComPython` parameter subscriptions.

1. The `mtb_sips_decoder_g` and `mtb_sips_decoder_d` blocks provide the binary interface between the `mtb_s_sips_g` and `mtb_s_sips_d` signals and the plant's System Integrity Protection Scheme (SIPS) interface.



2.3.3 Configure the MTB composite model

Add the ElmDsl model instances for the PPM P-control, PPC P-control, and PPC Q-control at the bottom of **Net Elements** in the MTB composite model `MTB.ElmComp`.



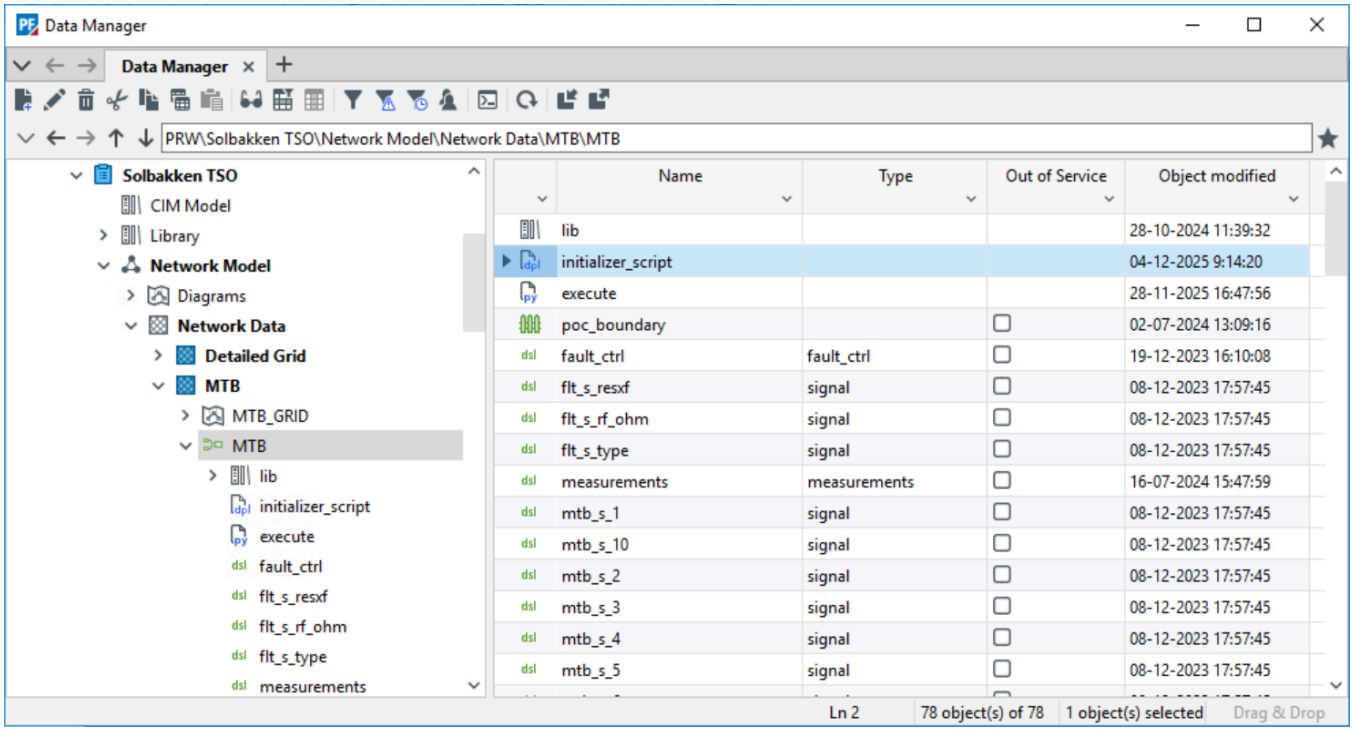
2.3.3.1 Edit initializer_script.ComDpl

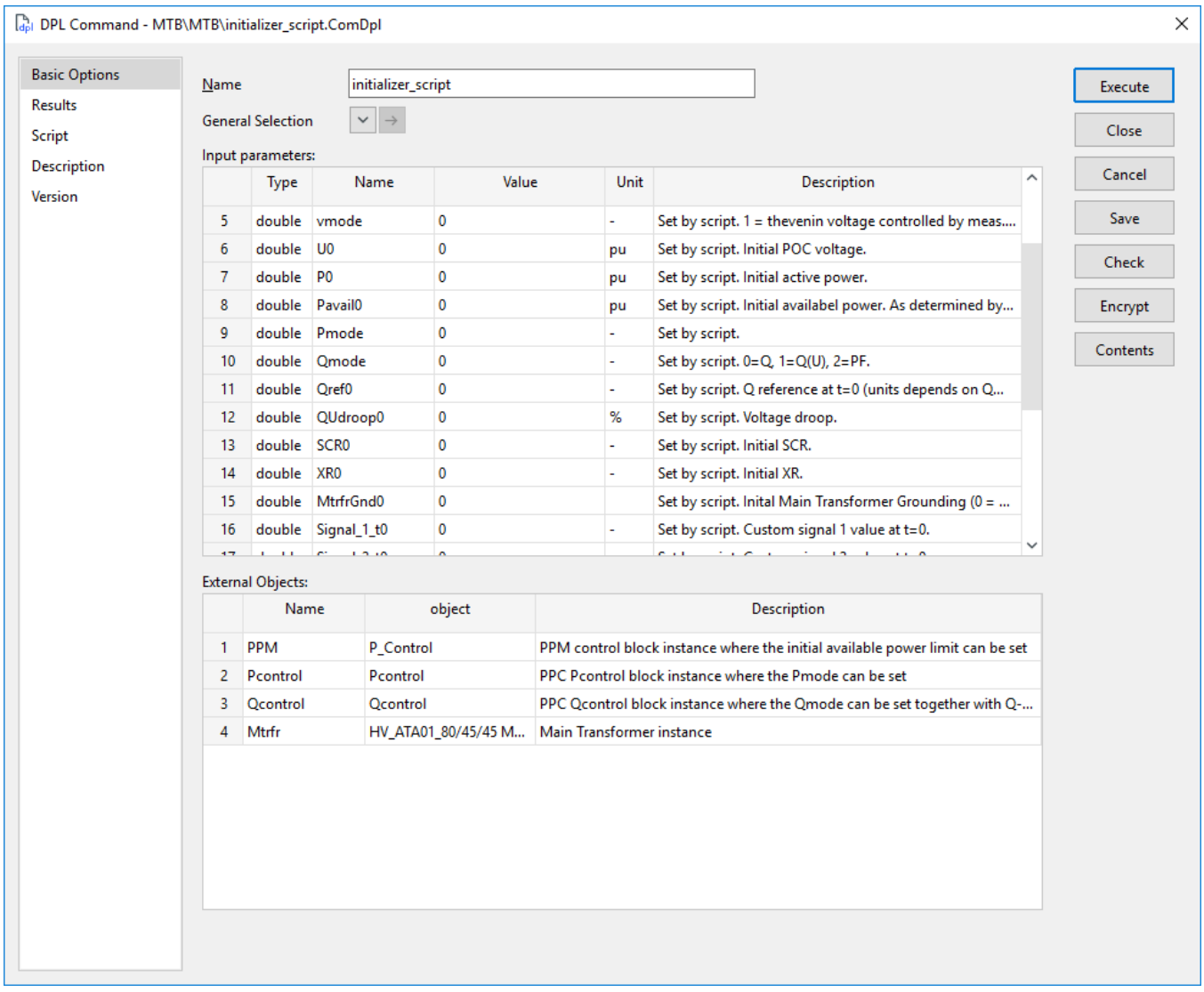
The `initializer_script.ComDpl` script is used to configure initial model parameters and objects for each test case. It is executed for each generated study case when execute_pf.py runs.

Typical initialization tasks include:

- Initial available power, `Pavail0`.
- Active power control mode.
- Reactive power control mode.
- Initial Q(U) droop value, `QUdroop0`.
- Initial main transformer grounding state, `MtrfrGnd0`.

Add the PPM P-control block, PPC P-control block, PPC Q-control block, and main transformer object to the **External Objects** table as required.





If the test cases include limited available power, assign the relevant plant parameter, for example `Irradiance` or `Pmax`, to `Pavail0`.

The default MTB P control modes are:

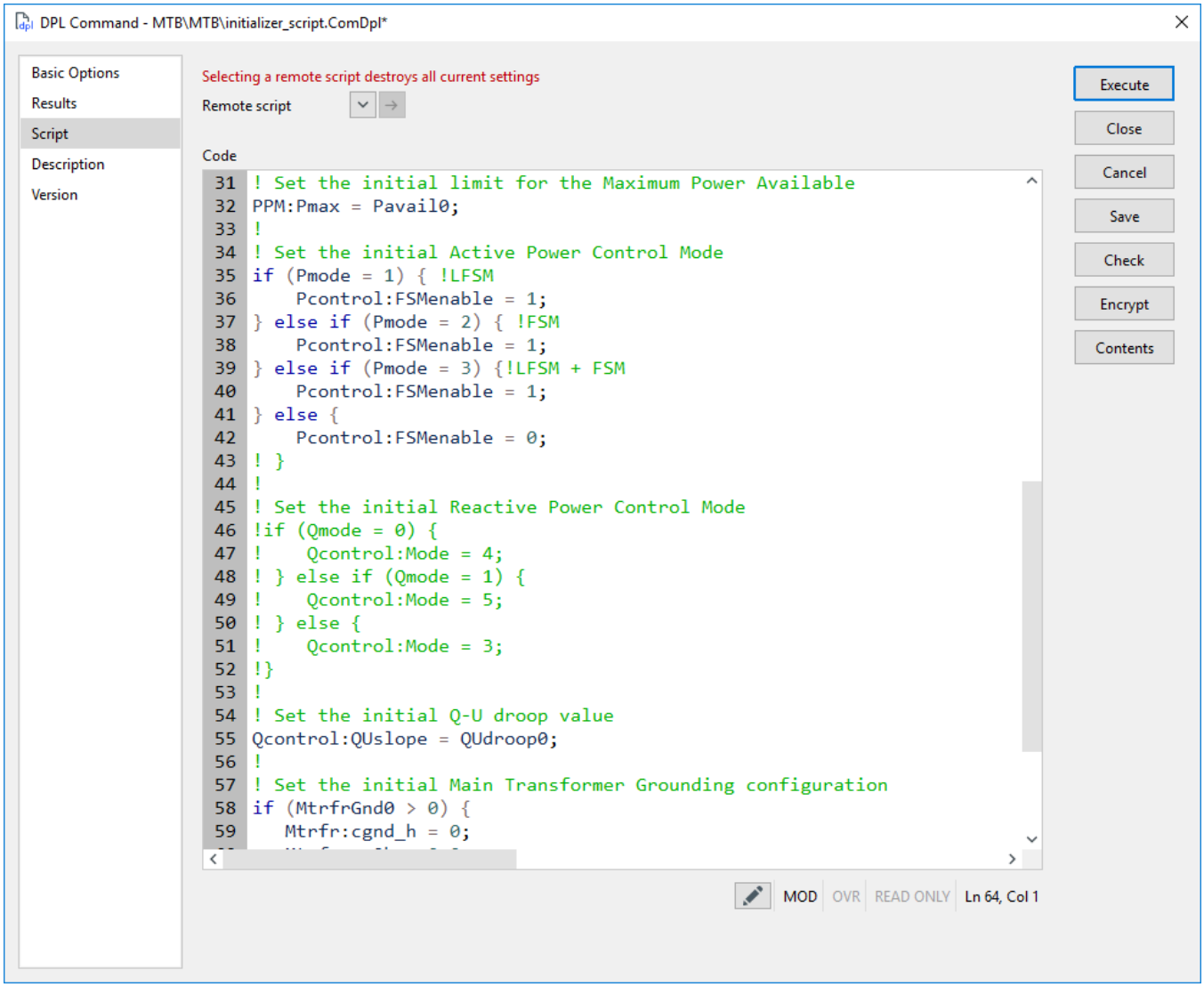
Value	Mode
0	No power-frequency relation, constant power
1	Limited Frequency Sensitivity Mode (LFSM)
2	Frequency Sensitivity Mode (FSM)
3	LFSM + FSM

The default MTB Q control modes are:

Value	Mode
0	Q reference mode
1	Q(U) mode
2	PF / cosphi mode

If the plant uses other integer values for Pmode or Qmode, use `initializer_script.ComDpl` to map the MTB values to the plant-specific control parameters.

If the test cases include changes to Q(U) droop or main transformer grounding, connect the relevant objects and uncomment or adapt the corresponding code in `initializer_script.ComDpl`.

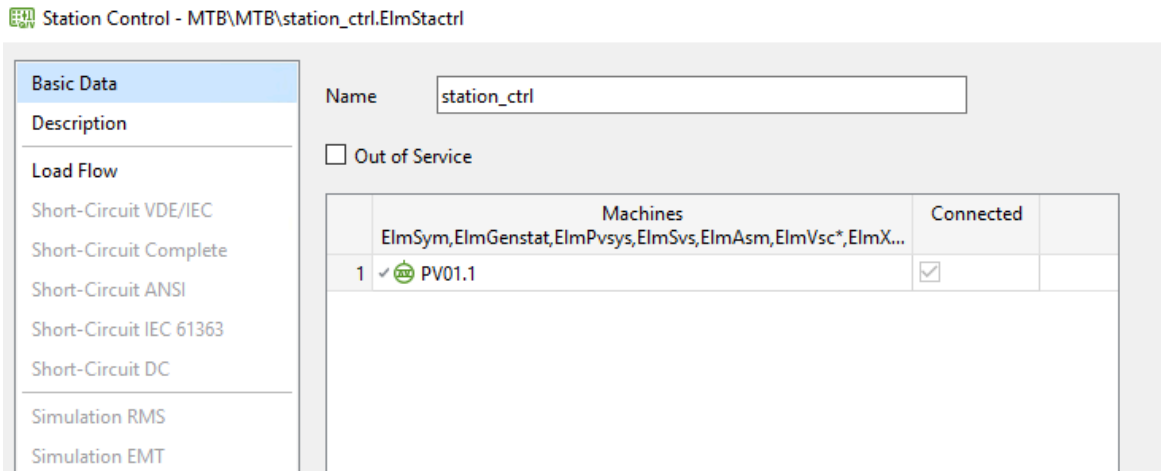


2.3.4 Connect the inverter to the MTB

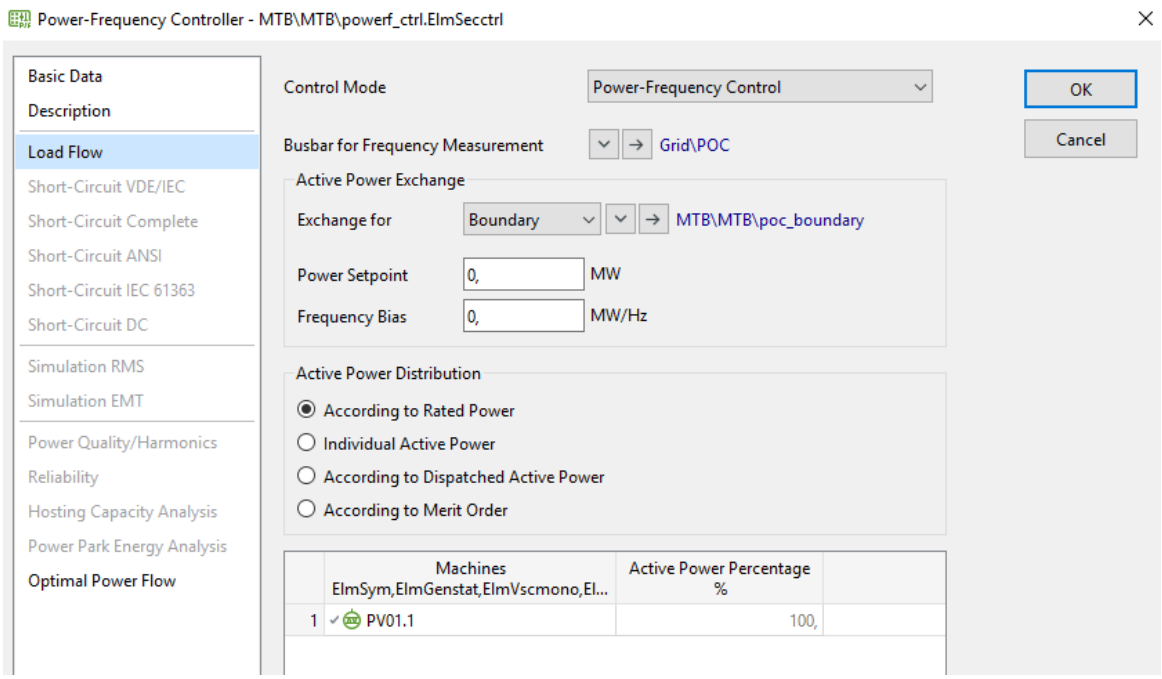
This can be done with the MTB built-in Station and Power Frequency controllers, or with the `initializer_qdsl.ElmQdsl` initializer.

2.3.4.1 Station and Power Frequency controller setup

1. Open the Station controller under the MTB folder and add the inverter `ElmGenstat` to the **Machines** column.

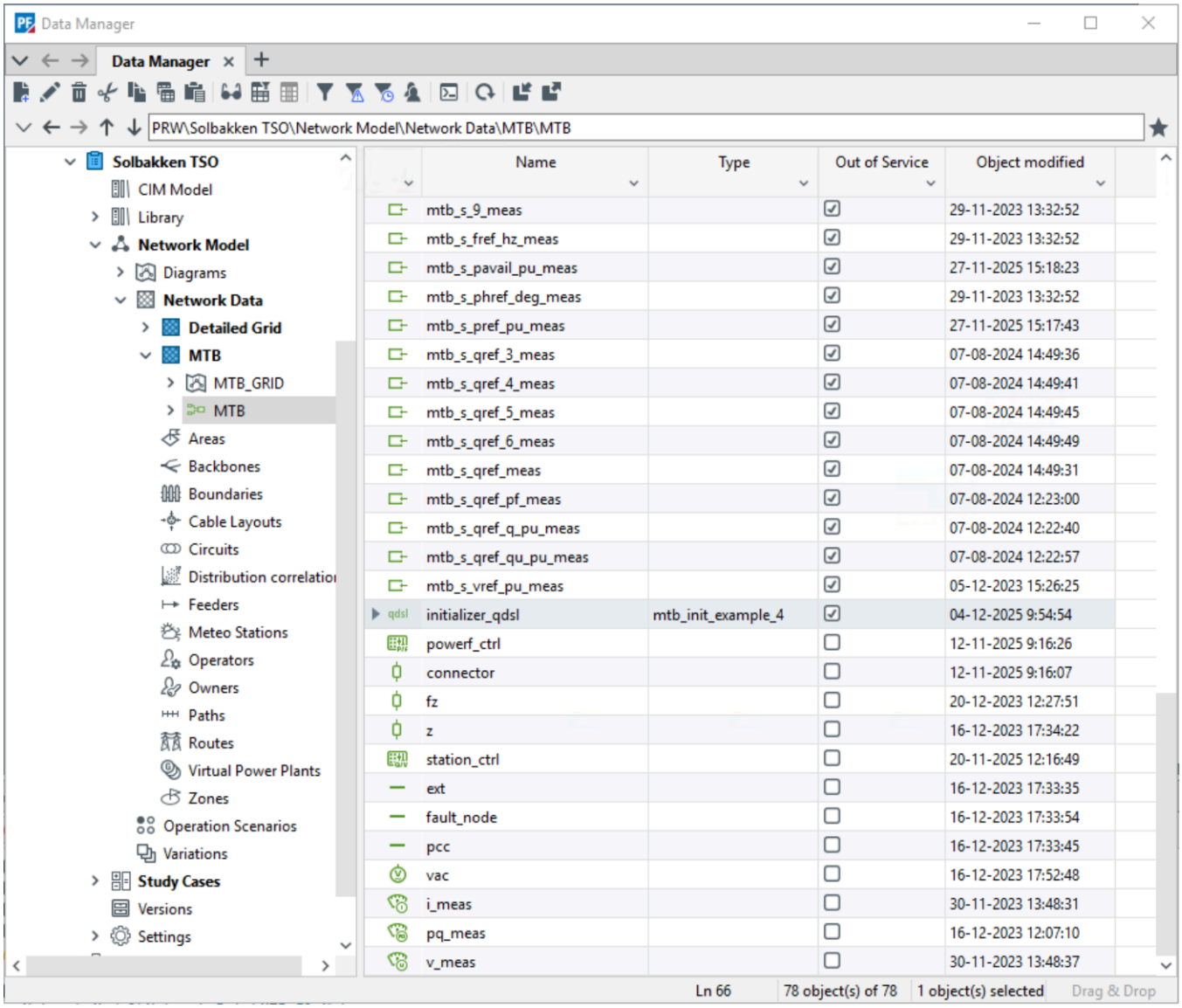


1. Open the Power Frequency controller and add the inverter on the **Load Flow** tab.

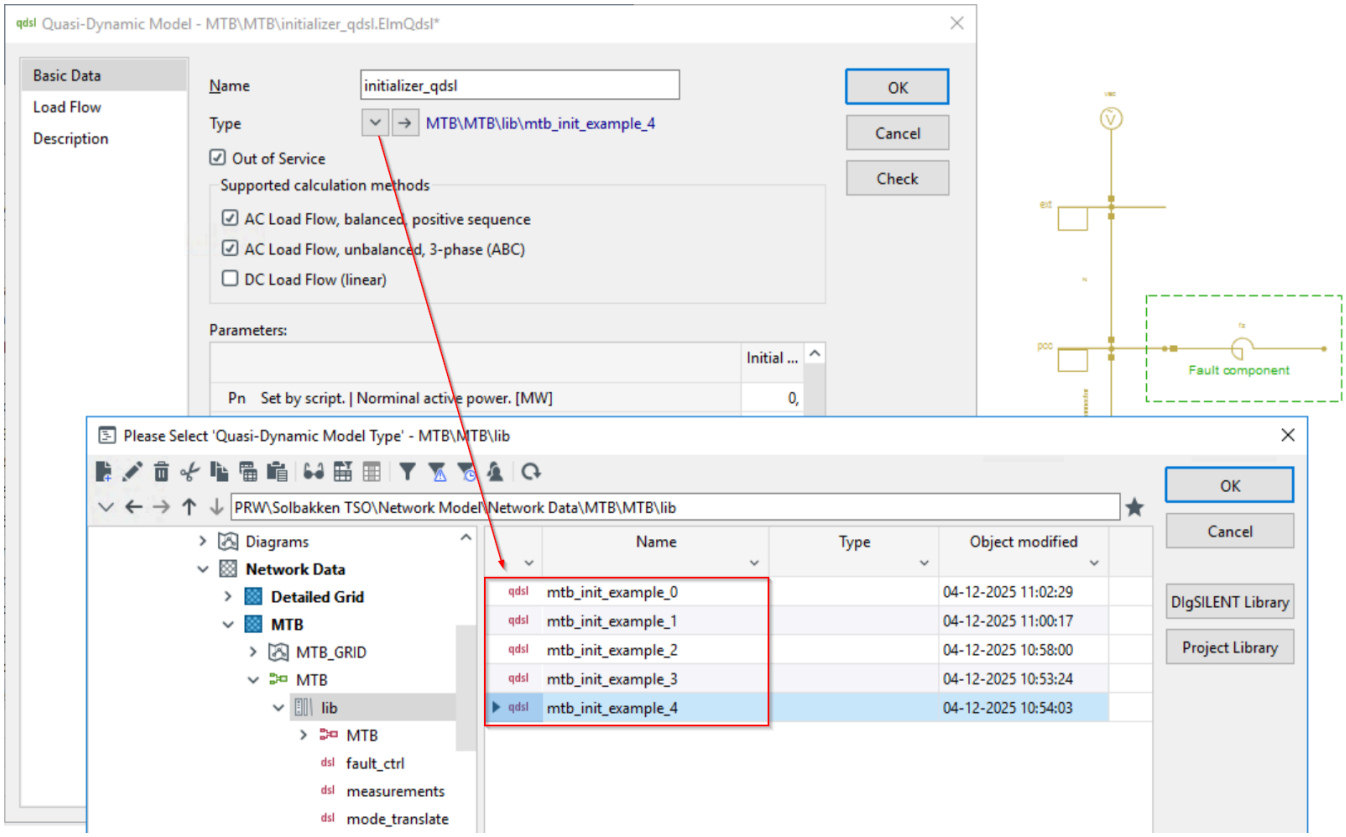


2.3.4.2 QDSL initializer setup

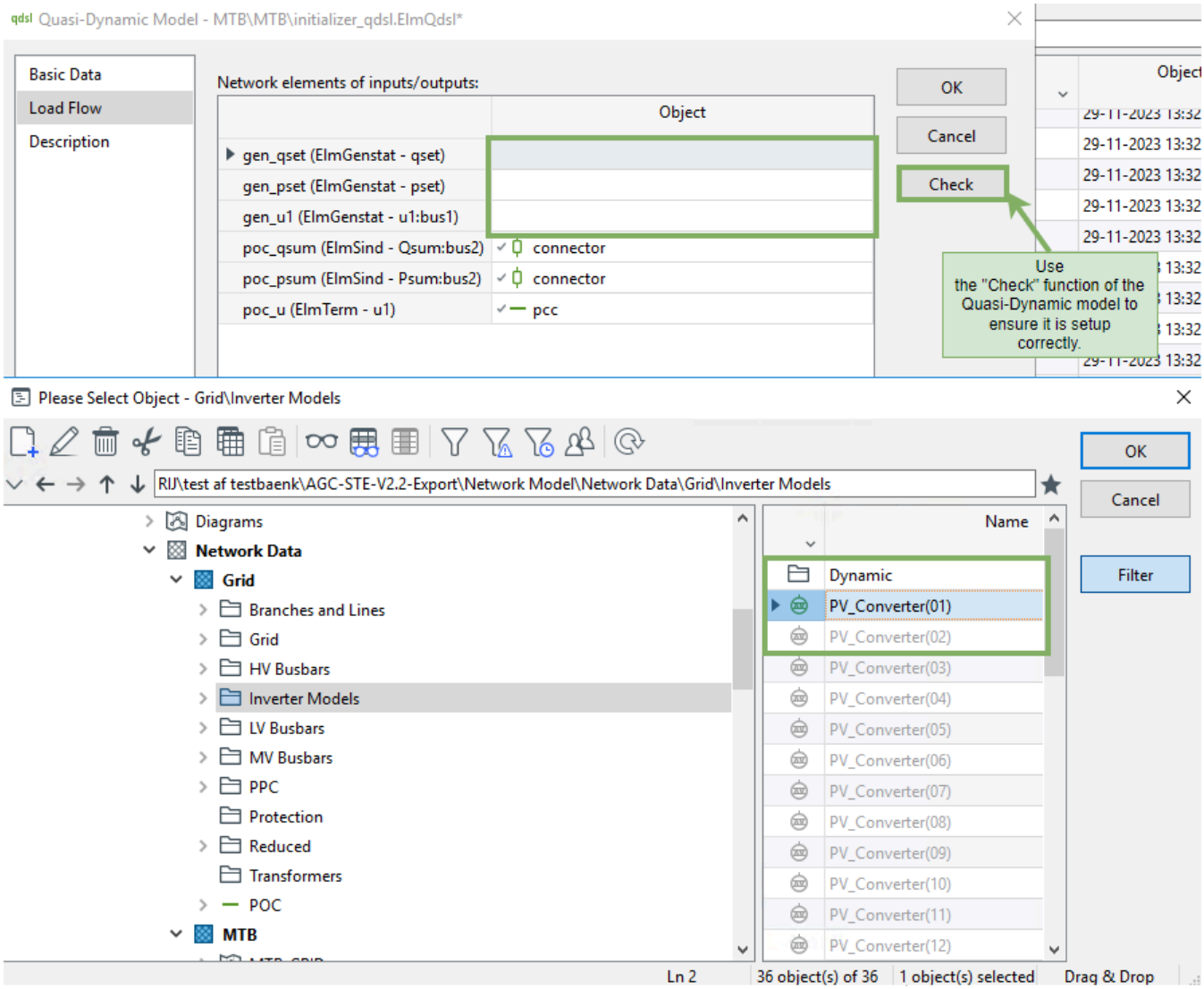
1. Put `initializer_qdsl.ElmQdsl` in service. The QDSL script is located in the MTB composite frame folder and is out of service by default.



1. Choose a QDSL type from the MTB library that fits the plant model and fill in the data that is not set by script. By default, `mtb_init_example_4` is selected and recommended.

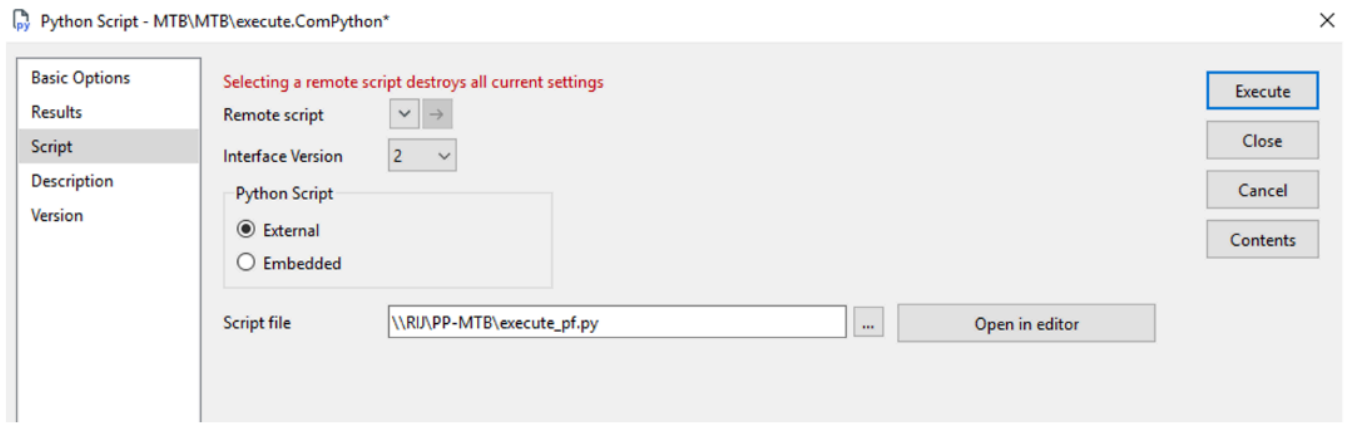


1. Select the load-flow tab of the QDSL block and connect the relevant network elements, for example the plant inverter.

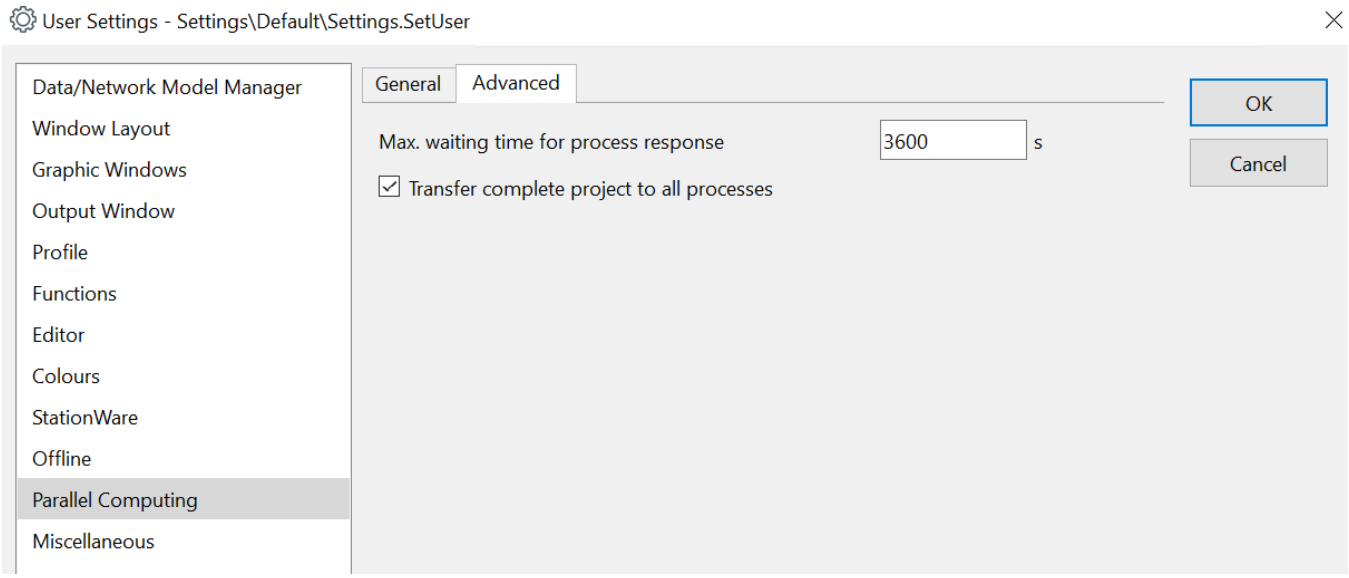


2.3.5 Configure execute.ComPython

Edit `execute.ComPython` and select execute_pf.py in the script tab. The full file path must be defined in **Script file**.



> **NOTE**
>
> If there are issues during plot setup or result export with parallel execution, go to **Tools -> User Settings -> Parallel Computing -> Advanced** and enable **Transfer complete project to all processes**.



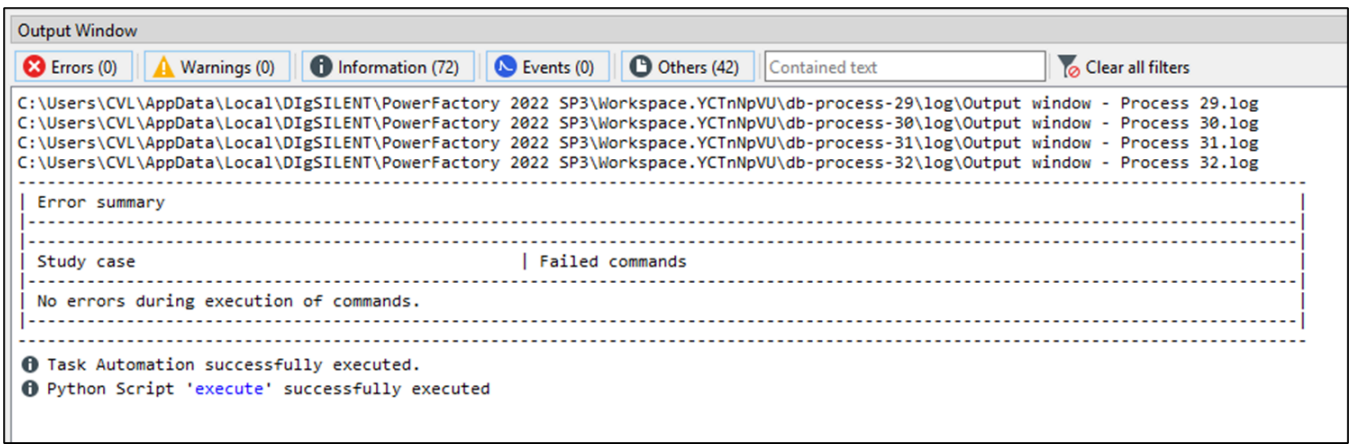
When `execute.ComPython` is executed, execute_pf.py:

1. Creates a pre-run project version named `PRE_MTB_<timestamp>`.
2. Resets project units and consolidates the active study case.
3. Reads plant settings and cases from testcases.xlsx.
4. Creates PowerFactory study cases and variations for all RMS-enabled cases.
5. Applies case-specific MTB signals and events.
6. Executes initial conditions, RMS simulation, and CSV export through `ComTasks`.
7. Sets up standard PowerFactory plots for PowerFactory 2024 and newer.
8. Optionally creates a post-run project version when `Post_run_backup > 0`.

> **NOTE**
>
> When executing the script, active variations are consolidated, meaning changes recorded in the active variations are applied to the Network Data folder.

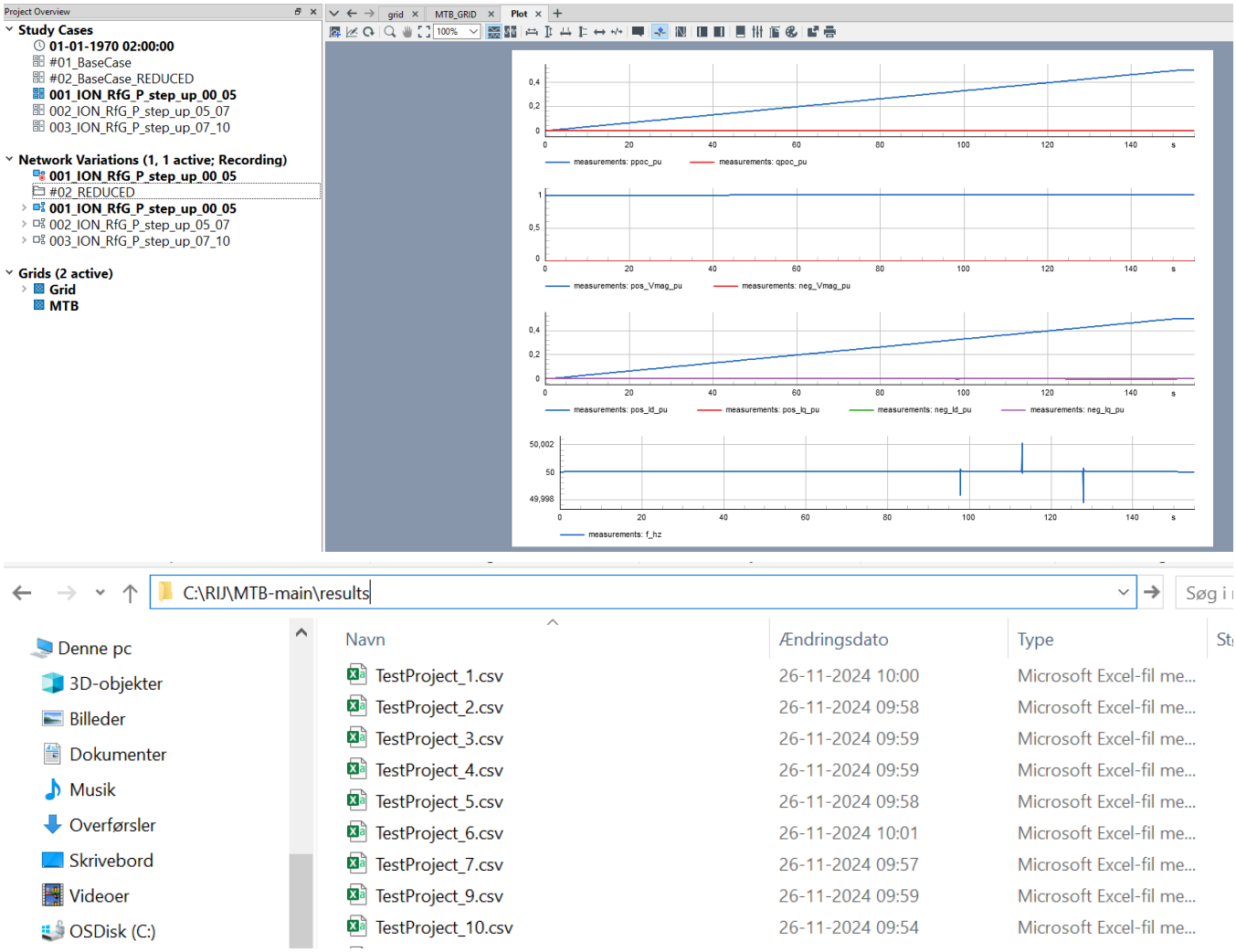
2.4 Script Execution Finished

When execution is finished, the PowerFactory output window should show that the Python script executed successfully.



The study cases enabled for RMS execution in testcases.xlsx should be visible in the project overview. If `create_case_folder = True`, generated study cases and variations are placed under `MTB_<Casegroup>` folders.

Simulation results are exported as CSV files to a timestamped subfolder inside the configured `Export folder`.



The exported `.csv` files can be plotted with the MTB plotter and compared with PSCAD results.

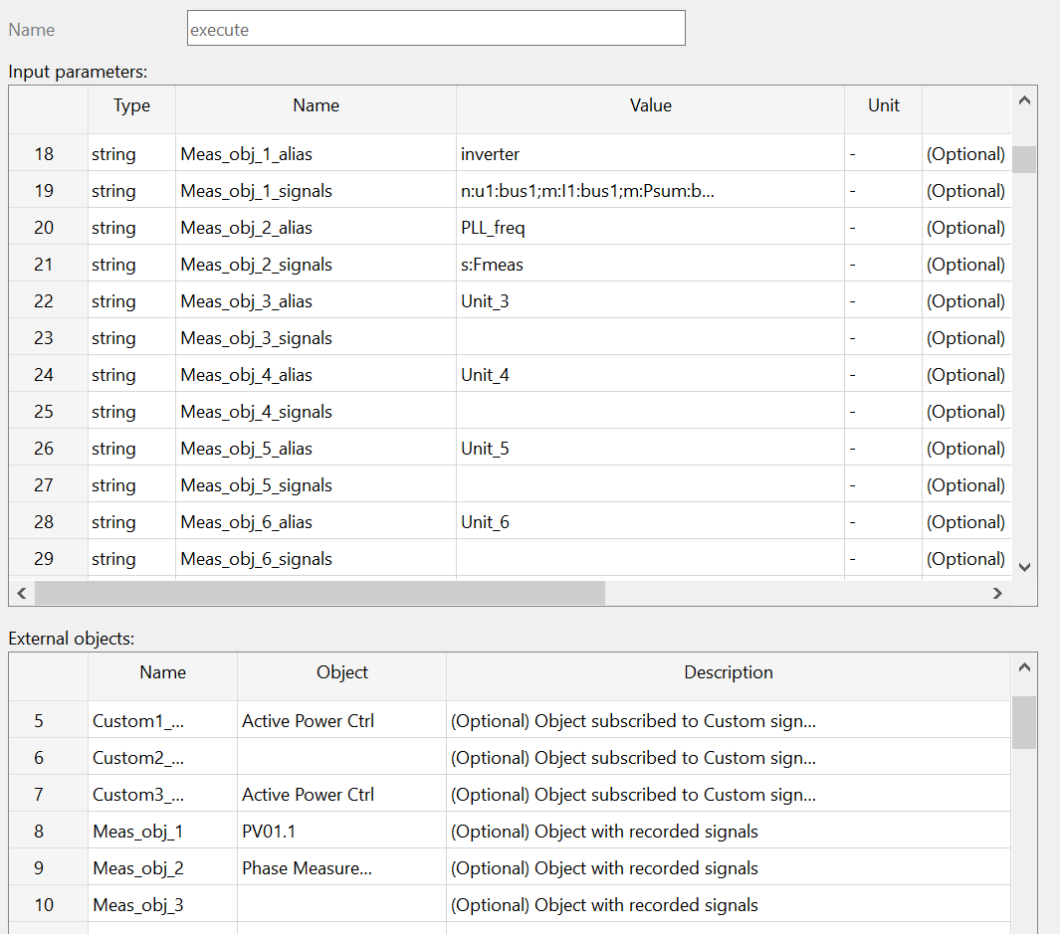
2.5 Add Additional Measurements to Result Files

Additional PowerFactory signals or parameters can be recorded through the `execute.ComPython` input parameters and external objects.

For each measurement object:

1. Add the object to the `Meas_obj_<n>` external object row.
2. Set `Meas_obj_<n>_signals` to a semicolon-separated list of PowerFactory result variables to record.
3. Set `Meas_obj_<n>_alias` to the alias used in the exported result file.

execute_pf.py scans `Meas_obj_1` through `Meas_obj_99` and adds the requested variables to the result file.



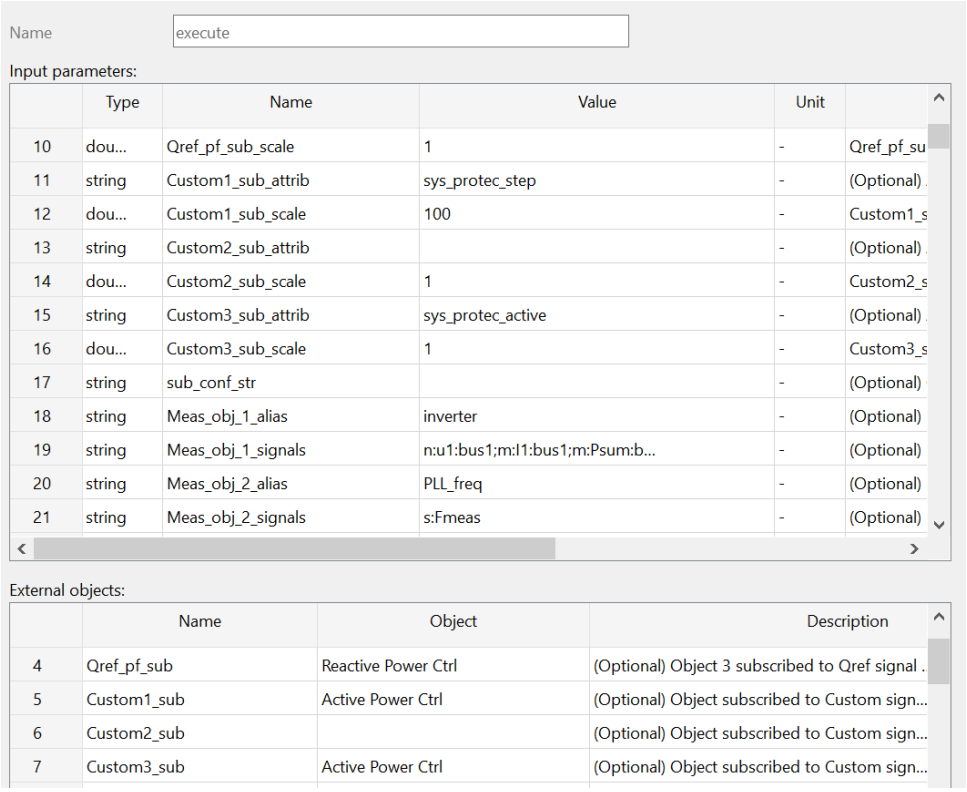
2.6 Custom Signal Setup

The MTB can control additional plant-specific signals or parameters in the PPC or other DSL models. Custom signals can be configured through `execute.ComPython` parameter subscriptions or through the MTB frame.

2.6.1 Custom signal setup through execute.ComPython

Use the custom signal input rows in `execute.ComPython` to subscribe plant parameters to MTB custom signals.

1. Enter the plant attribute to control and the required scaling. MTB custom signals are normally in pu.
2. Connect the object that owns the attribute in the external objects pane.


Name execute

Input parameters:

	Type	Name	Value	Unit	
10	dou...	Qref_pf_sub_scale	1	-	Qref_pf_su
11	string	Custom1_sub_attrib	sys_protec_step	-	(Optional)
12	dou...	Custom1_sub_scale	100	-	Custom1_s
13	string	Custom2_sub_attrib		-	(Optional)
14	dou...	Custom2_sub_scale	1	-	Custom2_s
15	string	Custom3_sub_attrib	sys_protec_active	-	(Optional)
16	dou...	Custom3_sub_scale	1	-	Custom3_s
17	string	sub_conf_str		-	(Optional)
18	string	Meas_obj_1_alias	inverter	-	(Optional)
19	string	Meas_obj_1_signals	n:u1:bus1;m:I1:bus1;m:Psum:b...	-	(Optional)
20	string	Meas_obj_2_alias	PLL_freq	-	(Optional)
21	string	Meas_obj_2_signals	s:Fmeas	-	(Optional)

External objects:

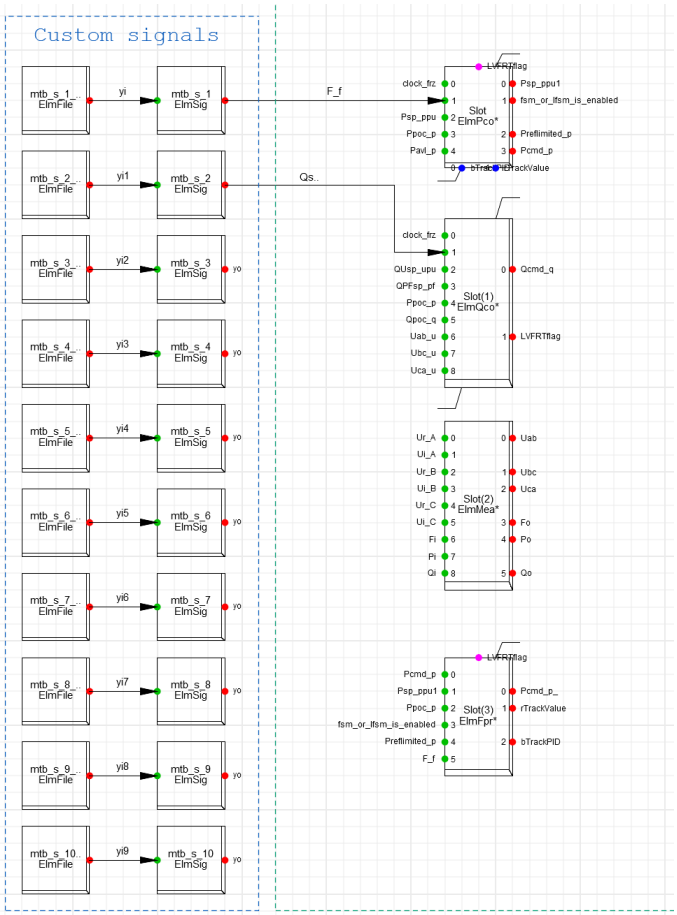
	Name	Object	Description
4	Qref_pf_sub	Reactive Power Ctrl	(Optional) Object 3 subscribed to Qref signal .
5	Custom1_sub	Active Power Ctrl	(Optional) Object subscribed to Custom sign...
6	Custom2_sub		(Optional) Object subscribed to Custom sign...
7	Custom3_sub	Active Power Ctrl	(Optional) Object subscribed to Custom sign...

Internally, execute_pf.py supports custom subscribers for `Custom1` through `Custom10`, connected to `mtb_s_1` through `mtb_s_10`.

2.6.2 Custom signal setup through the MTB frame

From the MTB frame graphic:

1. Copy existing slots from the **User Blocks** section and place them by the custom signal slots, or insert new blocks/signals from the drawing tools. The MTB frame type's graphical freeze mode must be unlocked.
2. Connect the MTB custom signals to the desired signal inputs in the plant control blocks.



2.7 Troubleshooting

- Make sure the P-control and Q-control blocks used for MTB setup are from the PPC, not from the inverter, unless the model intentionally uses inverter-level control.
- If PowerFactory parallel execution fails during plot setup or result export, enable **Transfer complete project to all processes** under **Tools -> User Settings -> Parallel Computing -> Advanced**.
- If QDSL initialization fails because the QDSL controller is not in the same grid as the calculation-relevant static generators, configure `QDSL copy grid` in config.ini or leave it blank to disable the workaround.

---

3. Quickstart PSCAD Guide

3.1 System Requirements

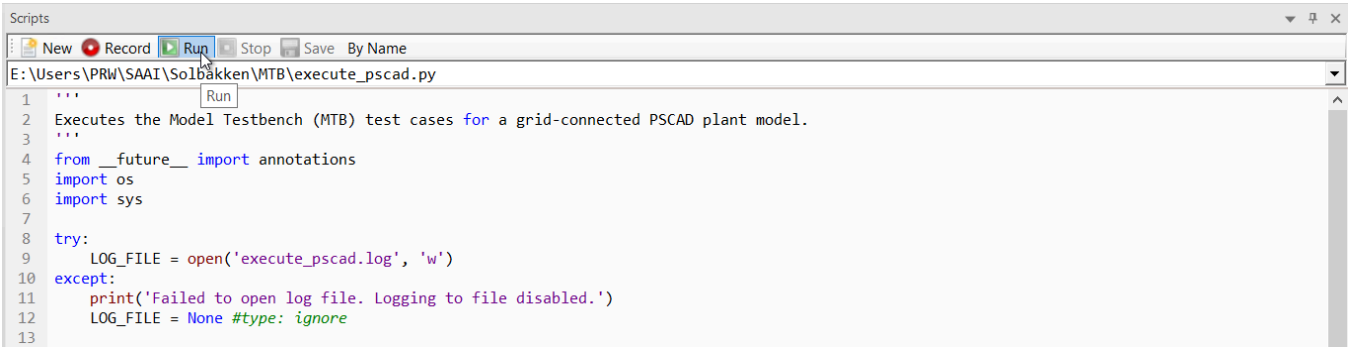
The MTB execute_pscad.py script is used to execute MTB test cases for a PSCAD plant model. It supports two execution modes:

Mode	Python environment	Typical use
Run from within PSCAD	PSCAD embedded Python 3.7.2	Automation tasks that may run slightly faster inside PSCAD
Run from the command line	External Python, for example Python 3.13	Automated execution and batch runs

The required Python packages are listed in requirements.txt in the main MTB folder.

3.1.1 Running from within PSCAD

When running execute_pscad.py from within PSCAD, PSCAD 5.0.2 uses its embedded Python 3.7.2 environment. This is not an interactive execution workflow; it is mainly useful because some PSCAD automation calls may run slightly faster when executed inside PSCAD.



External packages cannot always be installed directly into PSCAD's embedded Python environment. A practical workaround is:

1. Install Python 3.7.x.
2. Install the MTB requirements into that Python 3.7.x installation or virtual environment:

```
<PATH-TO-PYTHON37> -m pip install -r requirements.txt
```

A typical Python path is:

```
C:\Program Files\Python37\python.exe
```

If normal installation is blocked by permissions, install to the user's roaming Python folder:

```
python -m pip install --target C:\Users\
<USERNAME>\AppData\Roaming\Python\Python37\site-packages -r requirements.txt
```

The MTB usually finds the Python path automatically, but an additional path can be added with the `Python path` setting in config.ini, for example:

```
C:\Program Files\Python37\Lib\site-packages
C:\Users\<USERNAME>\AppData\Roaming\Python\Python37\site-packages
```

3.1.2 Running from the command line

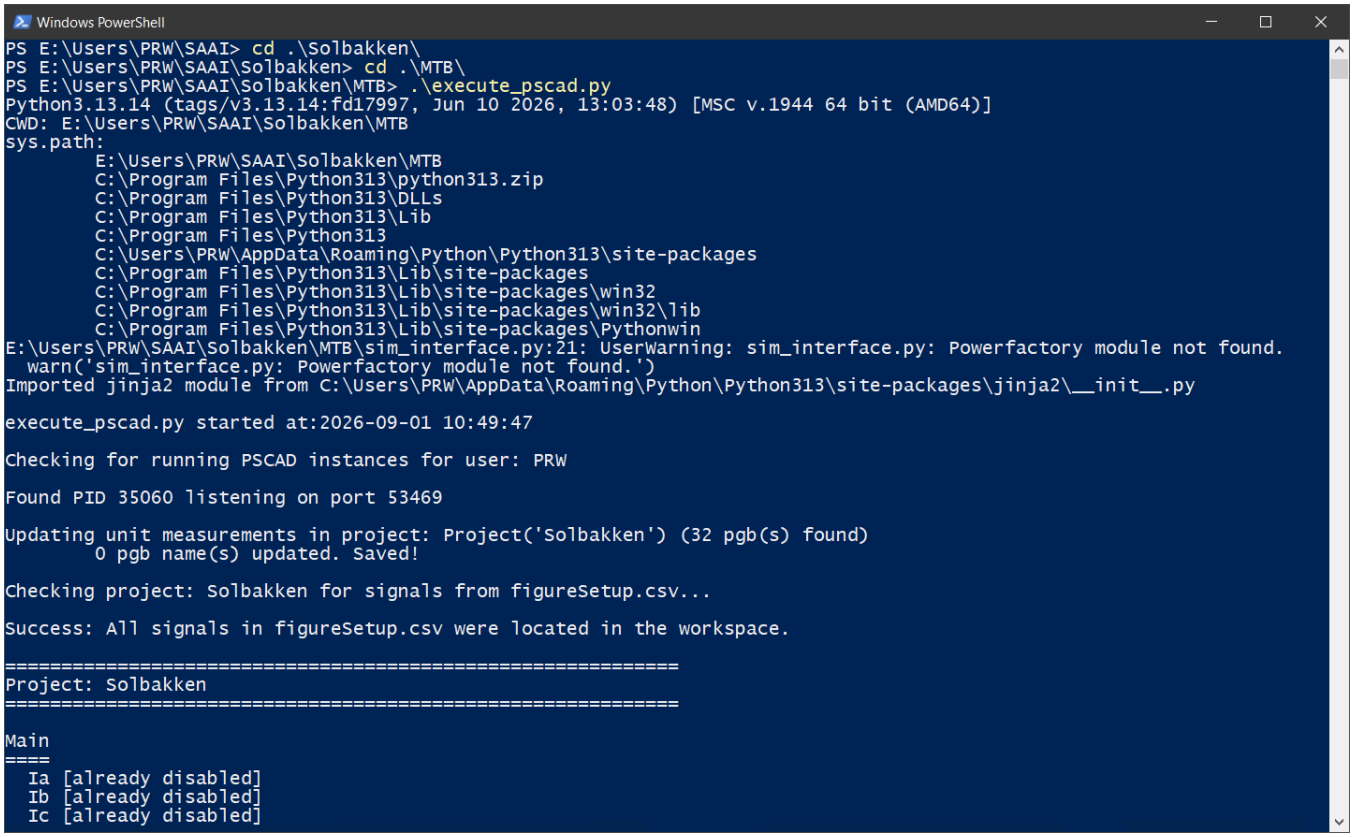
When running execute_pscad.py from the command line, or through batch*execute*pscad.py, the script uses the active external Python installation.

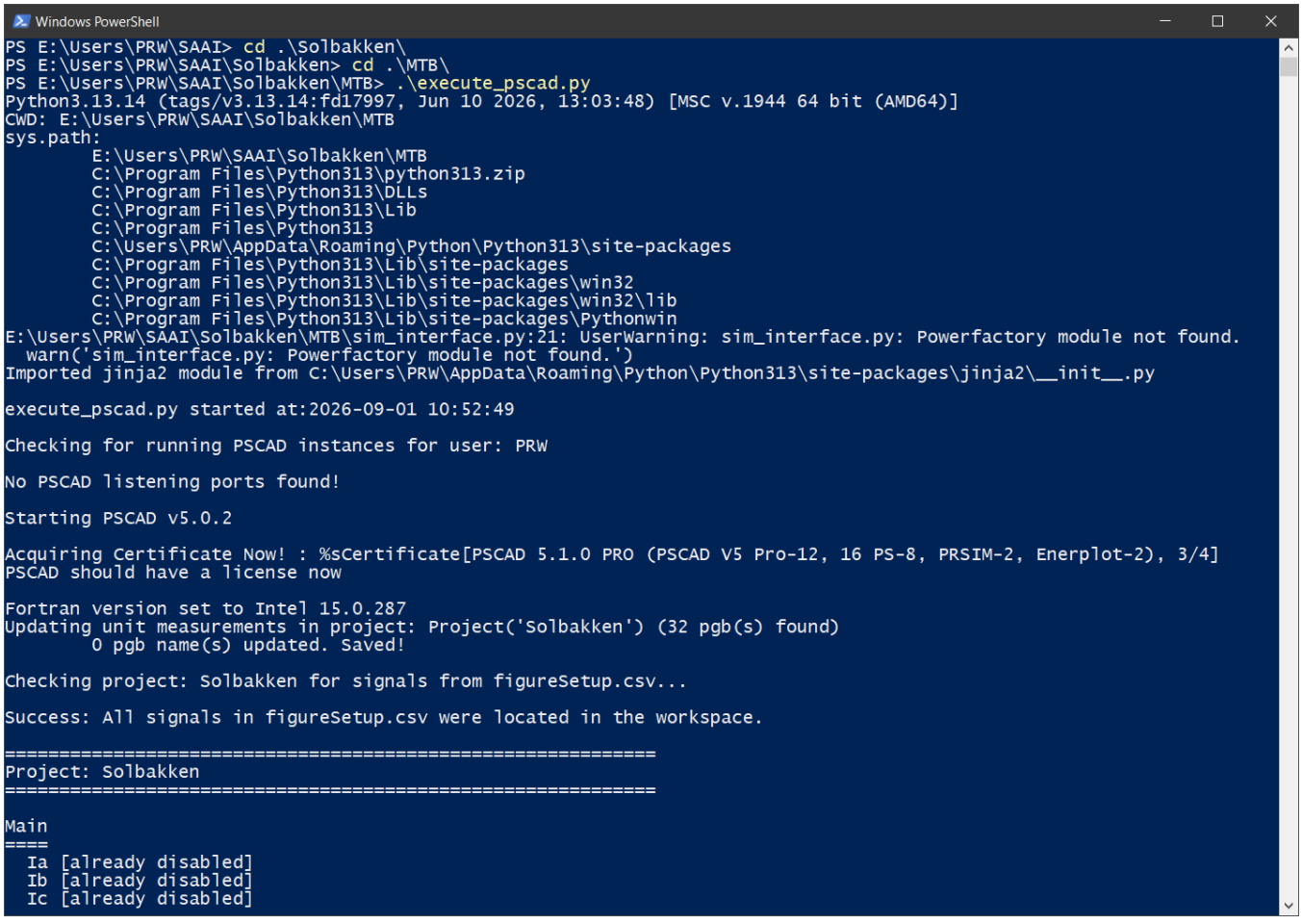
Install the MTB requirements for that Python installation:

```
python -m pip install -r requirements.txt
```

There are two command-line scenarios:

Scenario	Behaviour
PSCAD is already running	`execute_pscad.py` searches for a `Pscad.exe` process owned by the current Windows user and connects to one of its listening ports.
PSCAD is not already running	`execute_pscad.py` launches PSCAD 5.0.2, acquires a certificate license that meets the configured `Volley`, loads the configured workspace, and sets the configured Fortran compiler.


Windows PowerShell
PS E:\Users\PRW\SAAI> cd .\Solbakken\
PS E:\Users\PRW\SAAI\Solbakken> cd .\MTB\
PS E:\Users\PRW\SAAI\Solbakken\MTB> .\execute_pscad.py
Python3.13.14 (tags/v3.13.14:fd17997, Jun 10 2026, 13:03:48) [MSC v.1944 64 bit (AMD64)]
CWD: E:\Users\PRW\SAAI\Solbakken\MTB
sys.path:
E:\Users\PRW\SAAI\Solbakken\MTB
C:\Program Files\Python313\python313.zip
C:\Program Files\Python313\DLLs
C:\Program Files\Python313\Lib
C:\Program Files\Python313
C:\Users\PRW\AppData\Roaming\Python\Python313\site-packages
C:\Program Files\Python313\Lib\site-packages
C:\Program Files\Python313\Lib\site-packages\win32
C:\Program Files\Python313\Lib\site-packages\win32\lib
C:\Program Files\Python313\Lib\site-packages\Pythonwin
E:\Users\PRW\SAAI\Solbakken\MTB\sim_interface.py:21: UserWarning: sim_interface.py: Powerfactory module not found.
warn('sim_interface.py: Powerfactory module not found.')
Imported jinja2 module from C:\Users\PRW\AppData\Roaming\Python\Python313\site-packages\jinja2__init__.py
execute_pscad.py started at:2026-09-01 10:49:47
Checking for running PSCAD instances for user: PRW
Found PID 35060 listening on port 53469
Updating unit measurements in project: Project('Solbakken') (32 pgb(s) found)
0 pgb name(s) updated. Saved!
Checking project: Solbakken for signals from figureSetup.csv...
Success: All signals in figureSetup.csv were located in the workspace.
Project: Solbakken
Main
Ia [already disabled]
Ib [already disabled]
Ic [already disabled]


Windows PowerShell
PS E:\Users\PRW\SAAI> cd .\Solbakken\
PS E:\Users\PRW\SAAI\Solbakken> cd .\MTB\
PS E:\Users\PRW\SAAI\Solbakken\MTB> .\execute_pscad.py
Python3.13.14 (tags/v3.13.14:fd17997, Jun 10 2026, 13:03:48) [MSC v.1944 64 bit (AMD64)]
CWD: E:\Users\PRW\SAAI\Solbakken\MTB
sys.path:
E:\Users\PRW\SAAI\Solbakken\MTB
C:\Program Files\Python313\python313.zip
C:\Program Files\Python313\DLLs
C:\Program Files\Python313\Lib
C:\Program Files\Python313
C:\Users\PRW\AppData\Roaming\Python\Python313\site-packages
C:\Program Files\Python313\Lib\site-packages
C:\Program Files\Python313\Lib\site-packages\win32
C:\Program Files\Python313\Lib\site-packages\win32\lib
C:\Program Files\Python313\Lib\site-packages\Pythonwin
E:\Users\PRW\SAAI\Solbakken\MTB\sim_interface.py:21: UserWarning: sim_interface.py: Powerfactory module not found.
warn('sim_interface.py: Powerfactory module not found.')
Imported jinja2 module from C:\Users\PRW\AppData\Roaming\Python\Python313\site-packages\jinja2__init__.py
execute_pscad.py started at:2026-09-01 10:52:49
Checking for running PSCAD instances for user: PRW
No PSCAD listening ports found!
Starting PSCAD v5.0.2
Acquiring Certificate Now! : %sCertificate[PSCAD 5.1.0 PRO (PSCAD V5 Pro-12, 16 PS-8, PRSIM-2, Enerplot-2), 3/4]
PSCAD should have a license now
Fortran version set to Intel 15.0.287
Updating unit measurements in project: Project('Solbakken') (32 pgb(s) found)
0 pgb name(s) updated. Saved!
Checking project: Solbakken for signals from figureSetup.csv...
Success: All signals in figureSetup.csv were located in the workspace.
Project: Solbakken
Main
Ia [already disabled]
Ib [already disabled]
Ic [already disabled]

> **IMPORTANT**
>
> Command-line execution that launches PSCAD requires certificate licensing. If no existing PSCAD instance is found, the script uses the first available certificate that meets the `Volley` requirement in config.ini.

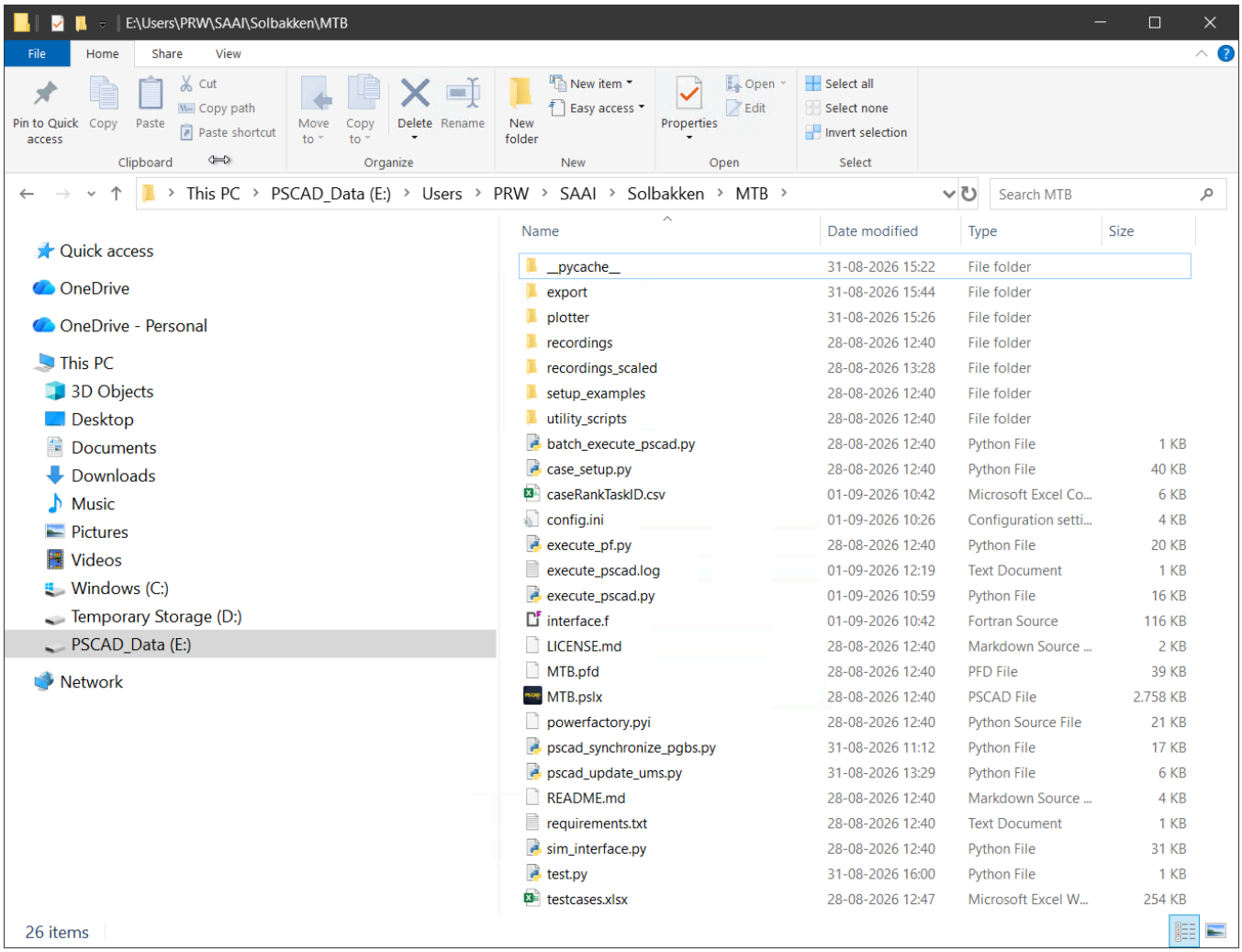
3.1.3 Example workspace

An example setup is available in setup_examples, including SimpleSolarFarm.pscx and MTB*Setup*Example.pswx. The example demonstrates the MTB setup only; it is not representative of a compliant plant model.

3.2 Preparation

3.2.1 Create an MTB installation folder

Create an MTB subfolder in the PSCAD workspace folder, where the `.pswx` file is usually located, and copy or extract the MTB files into that folder.



3.2.2 Edit the Settings sheet

Fill in the model-specific information in the `Settings` sheet in testcases.xlsx. The workbook guide is described in 1. Quickstart Testcases Workbook Guide.

	A	B	C	D
1	**Name**	**Value**	**Unit**	**Comment**
2	**Casegroup**	RfG	-	
3	**Run custom cases**	FALSK	-	*If custom not selected for casegroup.*
4	**Projectname**	Solbakken	-	
5	**Pn**	198,90	MW	
6	**Default P available**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed*
7	**Number of Units**	4		
8	**PnGS**	100,00	MW	*Allowed generation capacity in POC*
9	**PnDS**	100,00	MW	*Allowed consumption capacity in POC*
10	**Unit A type**	Generation		
11	**Pn Unit A [Generation]**	250,00	MW	*Nominal power (Generation)*
12	**Default P available or SoC Unit A**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
13	**Pn Unit A [Consumption]**	0,00	MW	*Nominal power (Consumption)*
14	**Unit B type**	Energy Storage		
15	**Pn Unit B [Generation]**	150,00	MW	*Nominal power (Generation)*
16	**Default P available or SoC Unit B**	0,50	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
17	**Pn Unit B [Consumption]**	100,00	MW	*Nominal power (Consumption)*
18	**Unit C type**	Demand		
19	**Pn Unit C [Generation]**	0,00	MW	*Nominal power (Generation)*
20	**Default P available or SoC Unit C**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
21	**Pn Unit C [Consumption]**	-10,00	MW	*Nominal power (Consumption)*
22	**Unit D type**	Generation		
23	**Pn Unit D [Generation]**	10,00	MW	*Nominal power (Generation)*
24	**Default P available or SoC Unit D**	1,00	pu	*Used to limit the available power due to e.g. reduced irradiance or wind speed or state of charge on battery*
25	**Pn Unit D [Consumption]**	0,00	MW	*Nominal power (Consumption)*
26	**Uc**	161,90	kV	*Used for easy setup to make sure simulations, start in normal operation voltage and not 1 pu.*
27	**Un**	152,00	kV	*This will be 1pu in simulations and results*
28	**Area**	DK1		
29	**FSM deadband**	0,05	Hz	*If any*
30	**FSM droop**	2,0	%	
31	**SCR min**	5,0	-	
32	**SCR tuning**	23,7	-	
33	**SCR max**	42,4	-	
34	**Default Q(U) droop**	4,0	%	
35	**X/R SCR min**	8,10	-	
36	**X/R SCR tuning**	16,90	-	
37	**X/R SCR max**	25,60	-	
38	**Main Transformer Grounded**	FALSK	-	*Default grounding setting for the Main Transformer*
39	**R0**	3,92	ohm	*Grounding impedance of the MTB voltage source*
40	**X0**	18,30	ohm	*Grounding impedance of the MTB voltage source*
41	**Default Q mode**	Q(U)	-	*Qmode signal from MTB is constant. Qref mode == 0, Q(U) mode == 1 and Qpf == 2.*
42	**PSCAD Timestep**	5	µs	
43	**PSCAD Initialization time**	3,55	s	*Atleast 1s*
44	**PF flat time**	0,15	s	*Atleast 0,1s*
45	**PF variable step**	SAND	-	
46	**PF enforced sync.**	FALSK	-	
47	**PF force asymmetrical sim.**	FALSK	-	
48	**PF enforce P limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*
49	**PF enforce Q limits in LDF**	SAND	-	*Can be neccesary to turn off when QDSL is utilized for initialization.*

Important MTB 2.x settings include:

- **Default P available**: associated with `mtb_s_pavail_pu`, the `Pavail` event type, and `Pavail0`.
- **Default Q(U) droop**: associated with `mtb_s_qudroop`, the `QUdroop` event type, and `QUdroop0`.
- **Main Transformer Grounded**: associated with `mtb_s_mtrfrgnd` and `MtrfrGnd0`.

3.2.3 Edit config.ini

Edit config.ini if required.

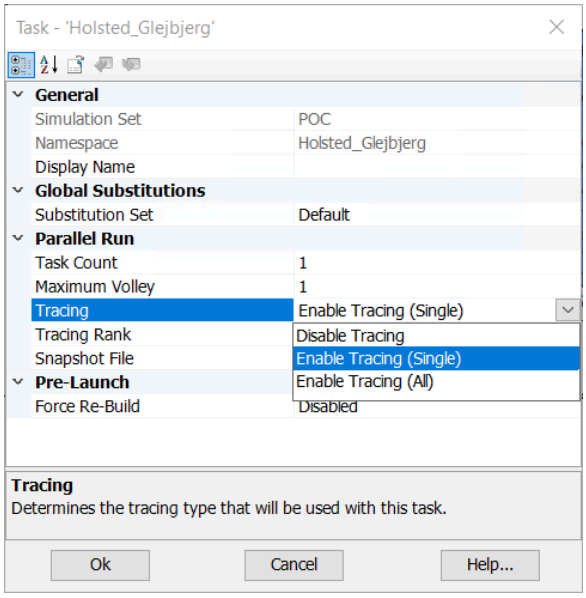
```
[General]
# Test case sheet path

Casesheet path = .\testcases.xlsx
# Path to export result files (relative to execute.py)
Export folder = export

[Python]
# Optional path to append to the python path
Python path =

[PSCAD]
# Volley size is the number of simulations to be run in parallel (up to a maximum of 64, license dependent).
# This should always be at least 1 less than the number of processors available.
Volley = 8

# "Tracing = False", corrisponds to "Affinity Type" 0, which means that none of the parallel simulation instance will
# send simulation data back to display in graphs during the simulation run.
# This can improve performance, but the user won't be able to see the live state of the simulation in the GUI until it finishes.
# "Tracing = True", corrisponds to "Affinity Type" 2 ('ALL'), which means that all the parallel simulation instance will
# send simulation data back to display in graphs during the simulation run.
# This can significantly increase overhead and reduce performance, especially for large simulations, but it allows the user to
# see the current state of the simulation in real time.
# This can be useful for monitoring and debugging, but may also increase overhead and reduce performance.
Tracing = False

# With "State animation = False", the schematic remains static.
# With "State animation = True", as the simulation runs, the PSCAD GUI constantly updates the schematic to show the current
# state of the power system.
State animation = False

# With "Only in use channels = False", the simulation engine calculates and prepares every single signal and component
# state in the entire project.
# With "Only in use channels = True", the engine only processes and buffers data for signals that are explicitly connected
# to an Output Channel or a Plot.
Only in use channels = True

# To reduce the size of the .psout files and reduce the possibility  of an Out of Memory (OoM) error in PSCAD when running
# a lot of test cases, it is recommended to only enable # the PGBs of the signals specified in the Plotter's figureSetup.csv file
Disable all unused PGBs = True

# Use legacy Unit measurement signal naming convention, e.g. <unit_name>\<alias>default_unit_measurement_signal_name,
# instead of the new convention, e.g. <unit_name>\default_unit_measurement_signal_name
# Because .psout files are stored with a signal path name, e.g. Unit, Unit_1, Unit_2, etc, a unique alias is no longer required
# to distinguish between signals with the same name in different measurement units.
Use legacy Unit measurement signal naming = True

# Fortran version when running "execute_pscad.py" from the command line, e.g. Intel 12.1.371, Intel 15.0.148,
# Intel 15.0.148 (64-bit), Intel 15.0.287, Intel 15.0.287 (64-bit), etc.
# REQUIRED only if no PSCAD instance owned by the user is already running (if connectPSCAD() finds none,
# startPSCAD() will launches a new instance using this compiler version).
# If an existing instance is found and reused, this setting is ignored.
Fortran version =

# Workspace path when running "execute_pscad.py" from the command line
# REQUIRED only if no PSCAD instance owned by the user is already running – the newly launched instance
# will load this workspace. If an existing instance is reused instead, whatever workspace is
# already loaded there is used, and this setting is ignored.
Workspace = ..\Solbakke.pswx
```

The PSCAD execution script reads the following settings:

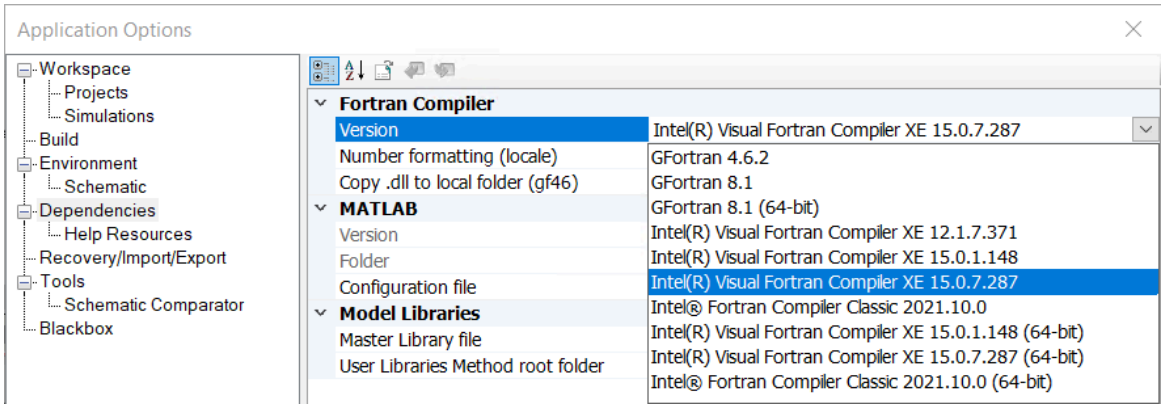
Section	Setting	Default / fallback	Used for
`[General]`	`Casesheet path`	`testcases.xlsx`	Path to the `testcases.xlsx` file used for PSCAD execution.
`[General]`	`Export folder`	`export`	Folder where `.psout` result files are moved after execution.
`[Python]`	`Python path`	blank	Optional path appended to `sys.path`, often used for PSCAD embedded Python package access.

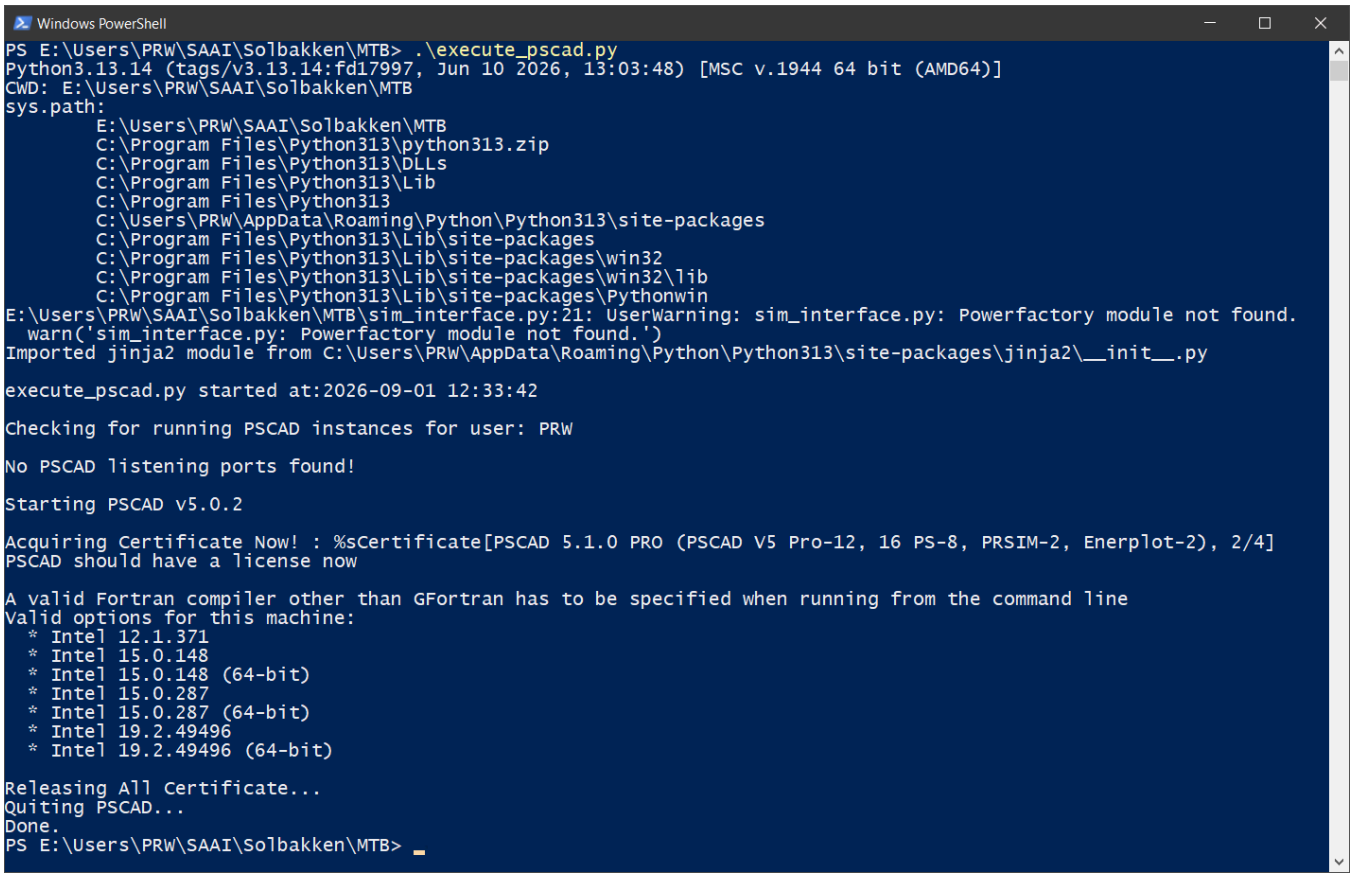
Section	Setting	Default / fallback	Used for
`[PSCAD]`	`Volley`	`16` in code, `8` in the default config	Number of parallel PSCAD simulation tasks.
`[PSCAD]`	`Tracing`	`False`	Sets PSCAD simulation-set affinity type. `False` disables tracing; `True` enables tracing for all workers.
`[PSCAD]`	`State animation`	`False`	Enables or disables schematic animation during simulation.
`[PSCAD]`	`Only in use channels`	`True`	Only processes channels explicitly connected to output channels or plots.
`[PSCAD]`	`Disable all unused PGBs`	`True`	Synchronizes Plotter Group Blocks with `plotter/figureSetup.csv` and disables unused PGBs.
`[PSCAD]`	`Use legacy Unit measurement signal naming`	`True`	Uses the legacy `<Unit_name>\<alias>_<signal_name>` naming style for Unit Measurement signals.
`[PSCAD]`	`Fortran version`	no fallback	Required when the script launches PSCAD from the command line. Ignored when connecting to an existing PSCAD instance.
`[PSCAD]`	`Workspace`	no fallback	Workspace loaded when the script launches PSCAD from the command line. Ignored when connecting to an existing PSCAD instance.

> **NOTE**
>
> `Tracing = False` corresponds to Disable Tracing. `Tracing = True` corresponds to Enable Tracing (All). Tracing is useful for debugging but can significantly slow down large parallel simulations.

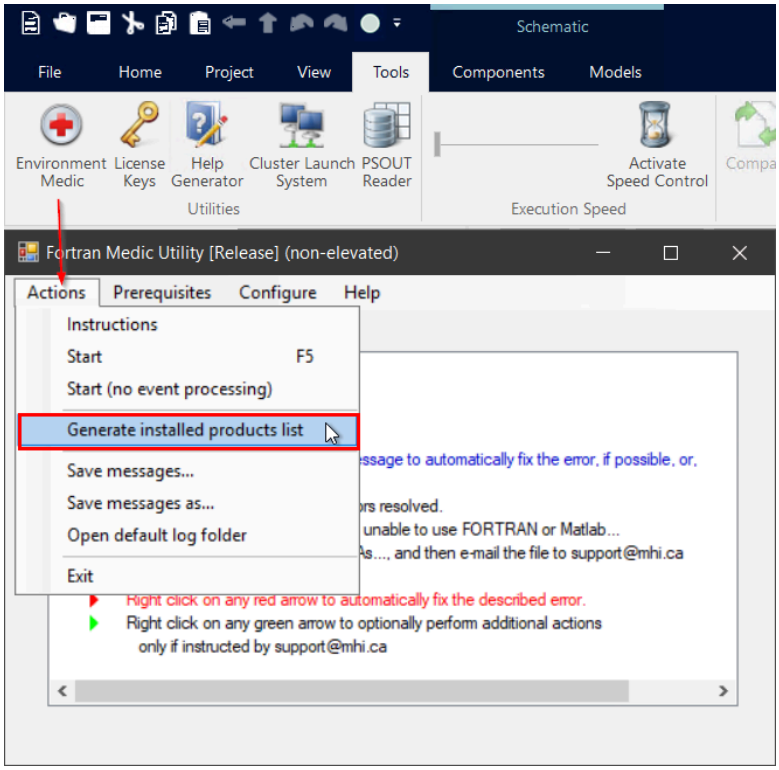


> **NOTE**
>
> The Fortran compiler name shown in the PSCAD GUI may not exactly match the value that must be written in config.ini. When running without an existing PSCAD instance, an empty or invalid `Fortran version` makes the script print the valid Intel compiler names available on the machine.





It may also be necessary to run the PSCAD **Environment Medic** and update the Fortran compiler list by selecting **Generate installed product list**, especially after a PSCAD server update.



> **IMPORTANT**
>
> Energinet's simulation model requirements currently specify that PSCAD models should use Intel Fortran 12 or 15.

3.3 MTB Setup in PSCAD

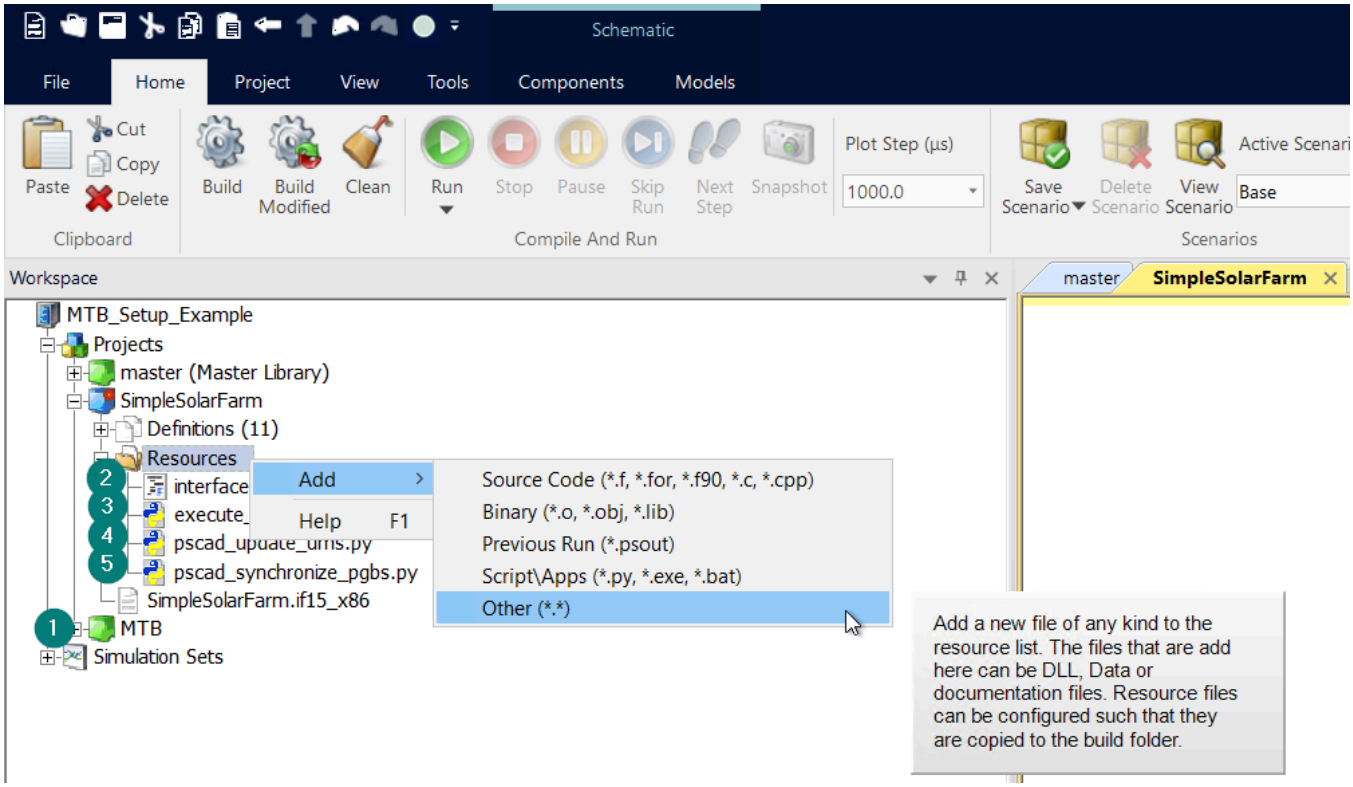
3.3.1 Import the MTB into the PSCAD workspace

Open a workspace with all required models, libraries, ETRAN libraries, and resources. In the example workspace, the model under test is `SimpleSolarFarm` in MTB*Setup*Example.pswx.

Add the following resources:

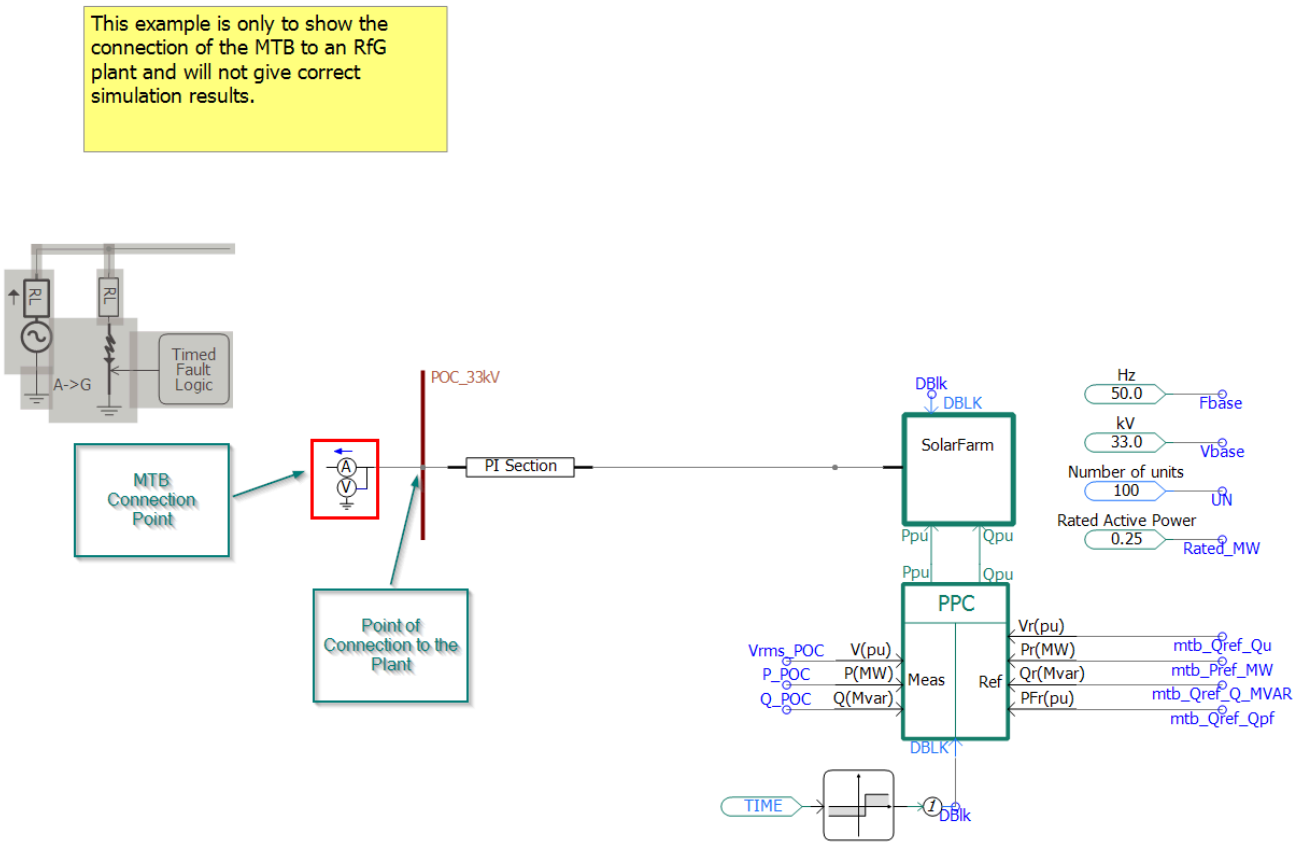
1. MTB.pslx, added as an existing project.
2. interface.f, added as source code.
3. execute_pscad.py, added as a script/app resource.
4. Optional pscad*update*ums.py, used for Unit Measurement signal naming.
5. Optional pscad*synchronize*pgbs.py, used to inspect or synchronize Plotter Group Blocks.

All resources can also be added in one go by right-clicking the Resources folder and selecting **Add -> Other (.)**.

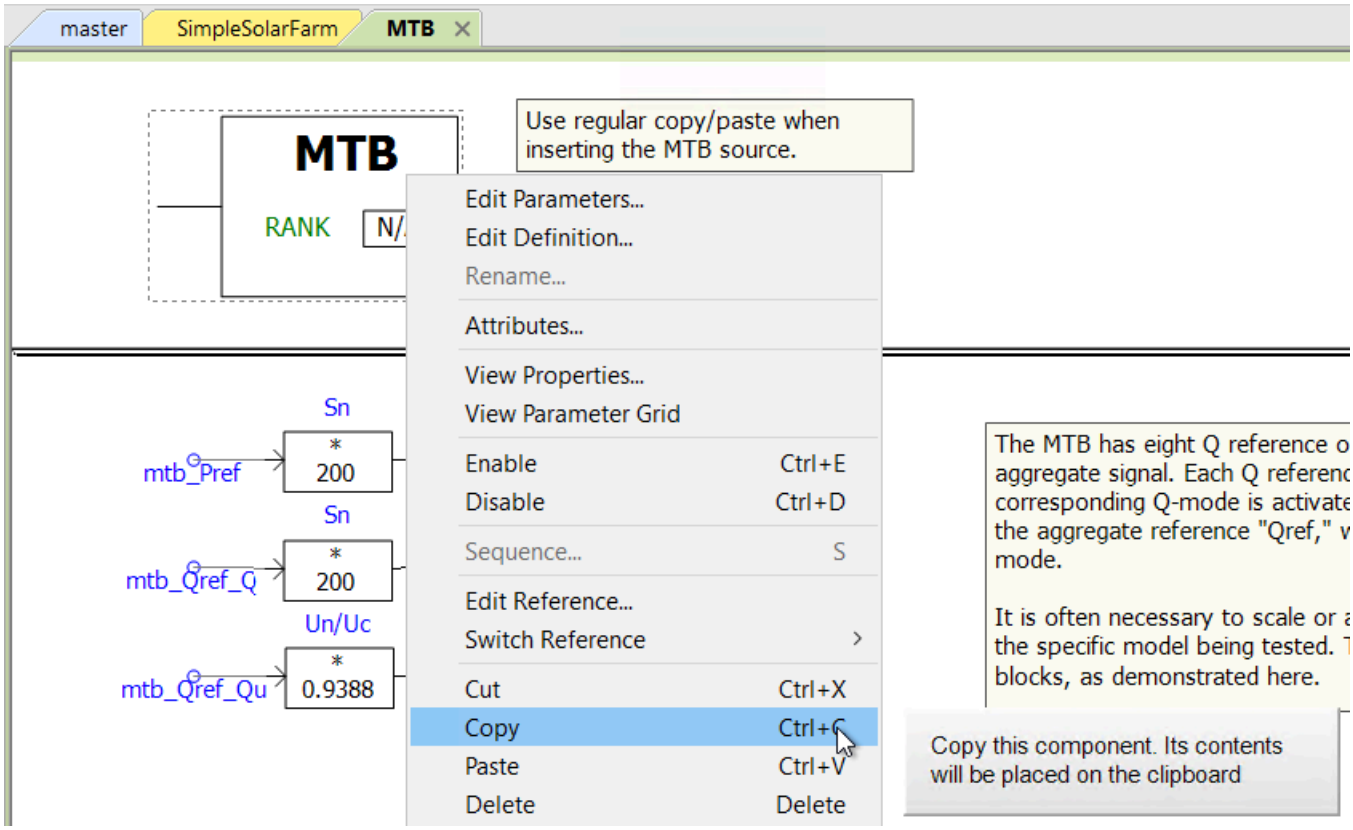


3.3.2 Connect the MTB block to the PoC

1. Disconnect and disable the power grid from the PoC in the model. Keep any PoC measurements that are already present.

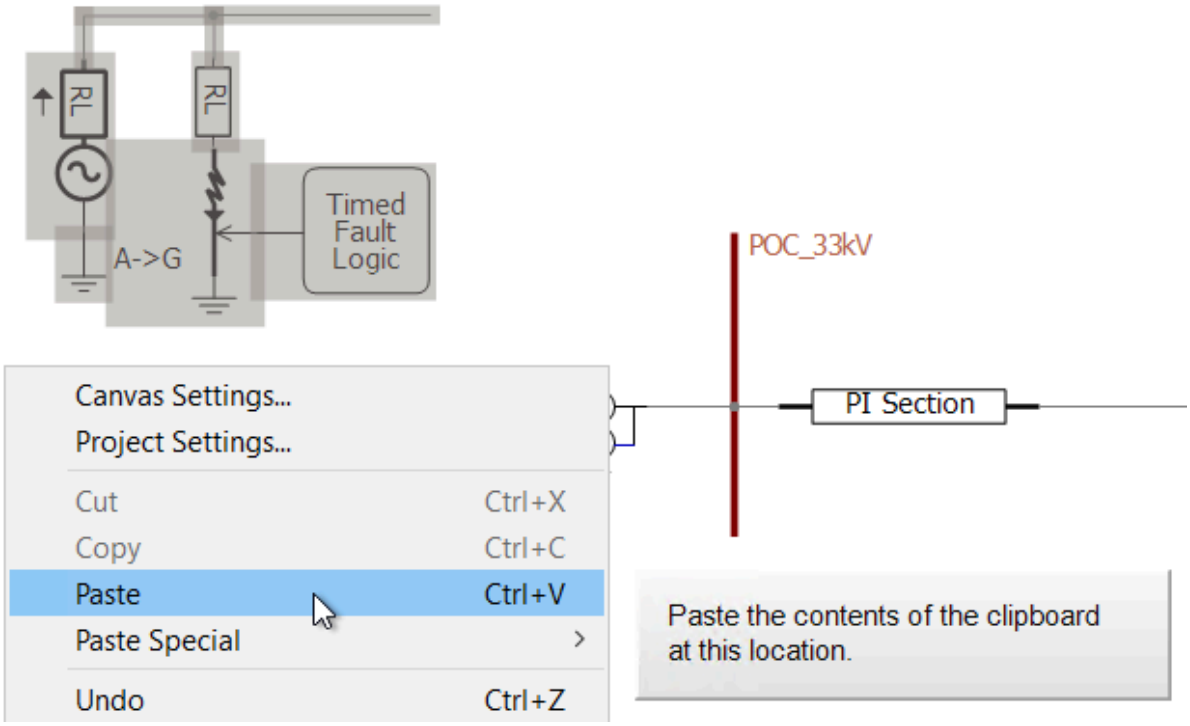


1. Copy the MTB block from the MTB project.

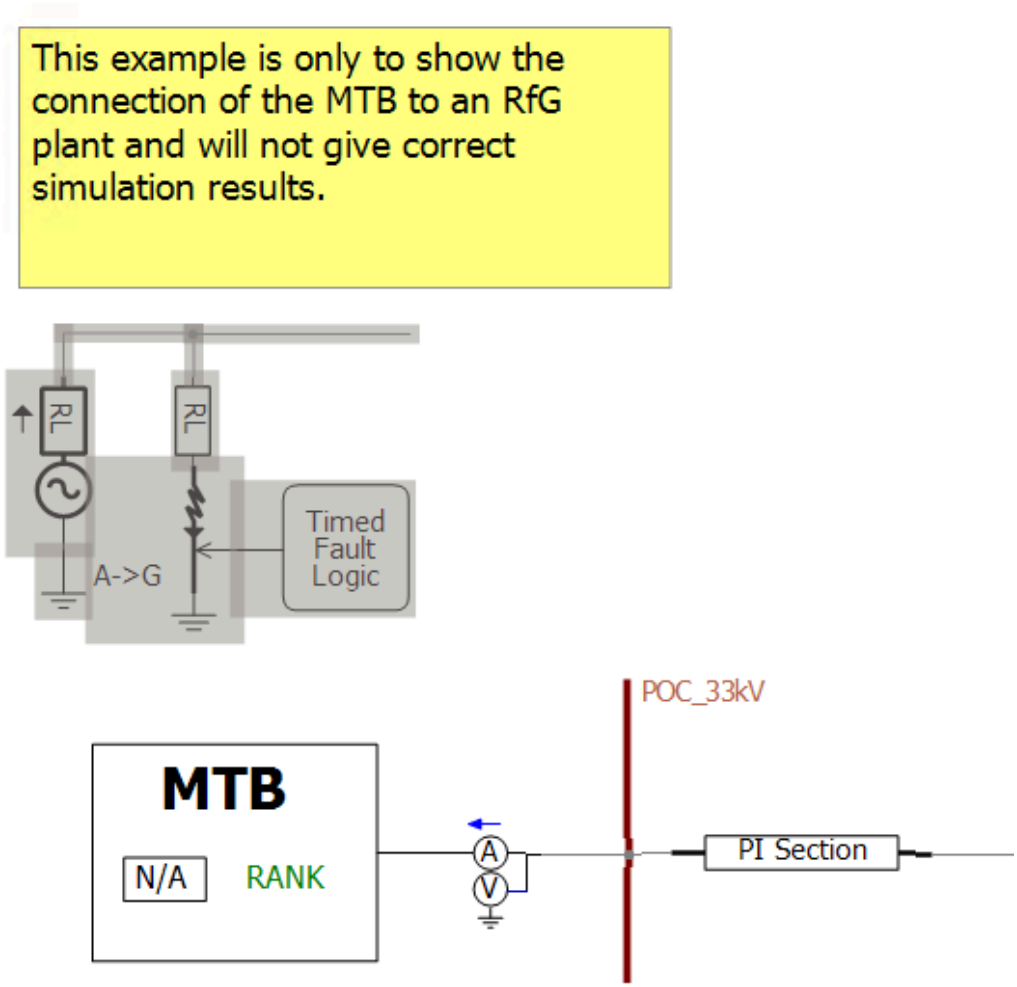


1. Paste the MTB block into the plant model and connect it to the PoC.

This example is only to show the connection of the MTB to an RfG plant and will not give correct simulation results.



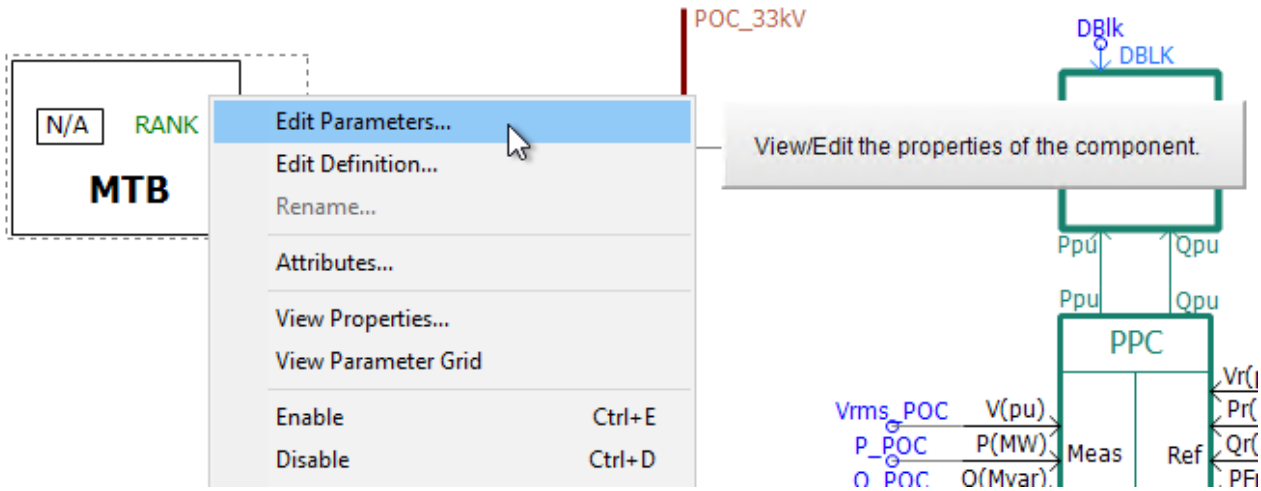
1. Adjust the orientation of the MTB block if required.

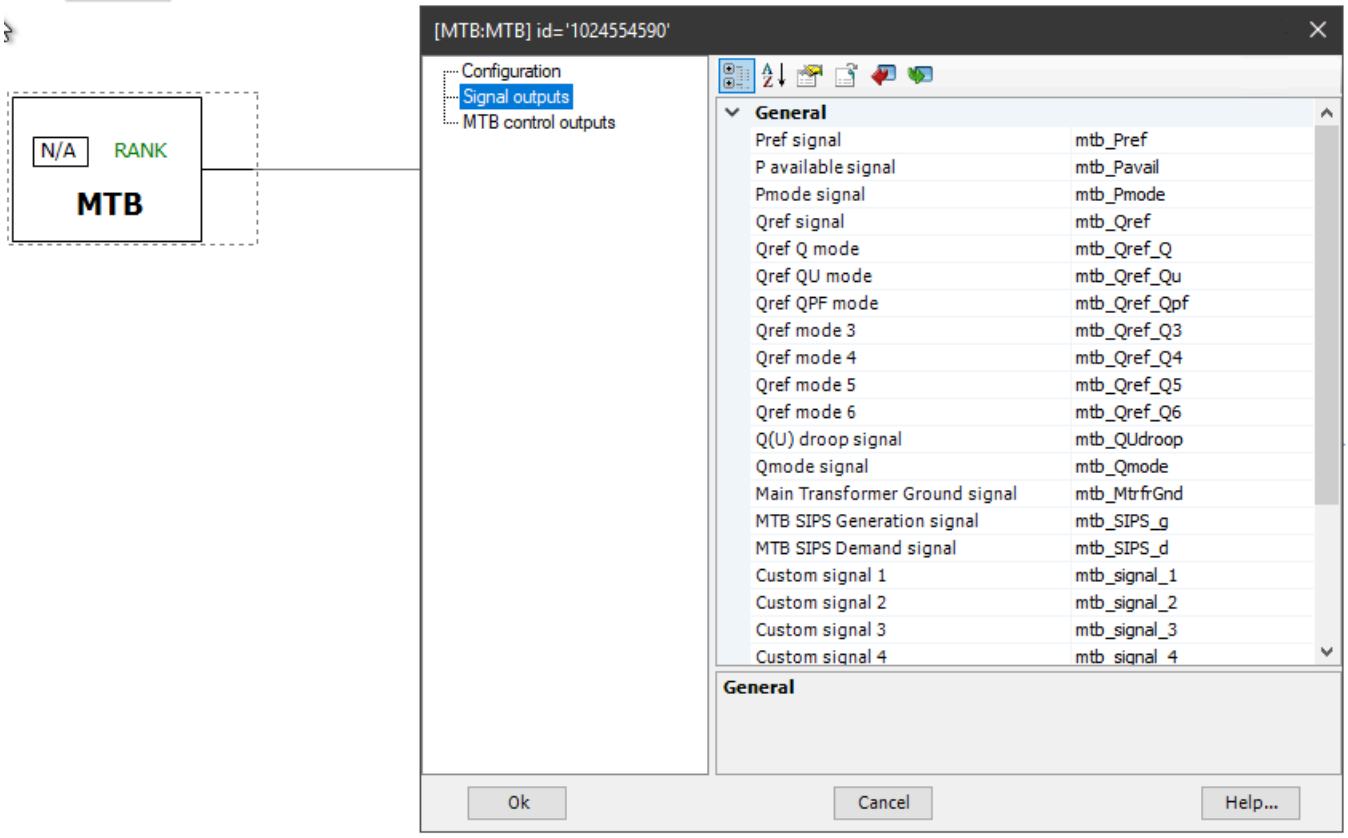


3.3.3 Connect MTB controls to the PPC

3.3.3.1 MTB output signals

Right-click the MTB block, select **Edit Parameters...**, and open **Signal outputs** to view the output signals.





The main output signals are:

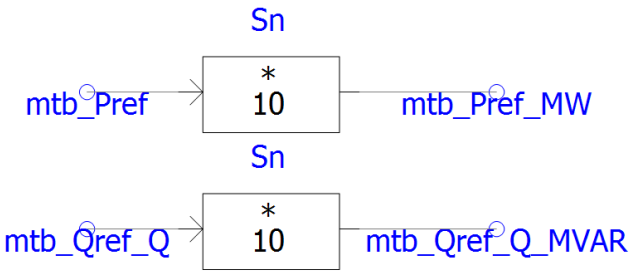
Signal	Purpose
`Pref`	Active power reference in pu using `Pn` from the `Settings` sheet.
`Pavail`	Available active power limit in pu.
`Qref`	Shared reactive power, voltage, or power factor reference depending on Q mode.
`Qref_Q`	Reactive power reference when Q mode is active.
`Qref_Qu`	Voltage reference when Q(U) mode is active.
`Qref_Qpf`	Power factor reference when PF mode is active.
`Qref mode 3-6`	Custom Q-mode outputs.
`Q(U) droop`	Q(U) droop value in percent.
`Pmode`	Active power mode: 0 = FSM/LFSM disabled, 1 = LFSM, 2 = FSM, 3 = FSM + LFSM.
`Qmode`	Reactive power mode: 0 = Q, 1 = Q(U), 2 = PF.
`Main Transformer Ground`	Main transformer grounding signal.
`mtb_SIPS_g`	Integer SIPS signal for generation facilities.
`mtb_SIPS_d`	Integer SIPS signal for demand facilities.

Signal	Purpose
`Custom signal 1-10`	Custom signal outputs driven by `Signal 1`, `Signal 2`, etc. events in testcases.xlsx.

3.3.3.2 Convert input signals to the PPC

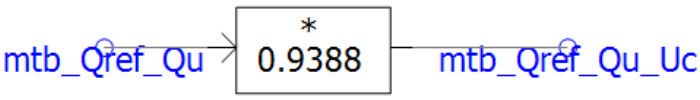
Pref and Qref_Q

If the PPC expects active and reactive power inputs in MW and Mvar, scale the pu references by the plant rating. In the example plant, rated at 10 MW, `mtb_Pref` and `mtb_Qref_Q` are multiplied by 10.



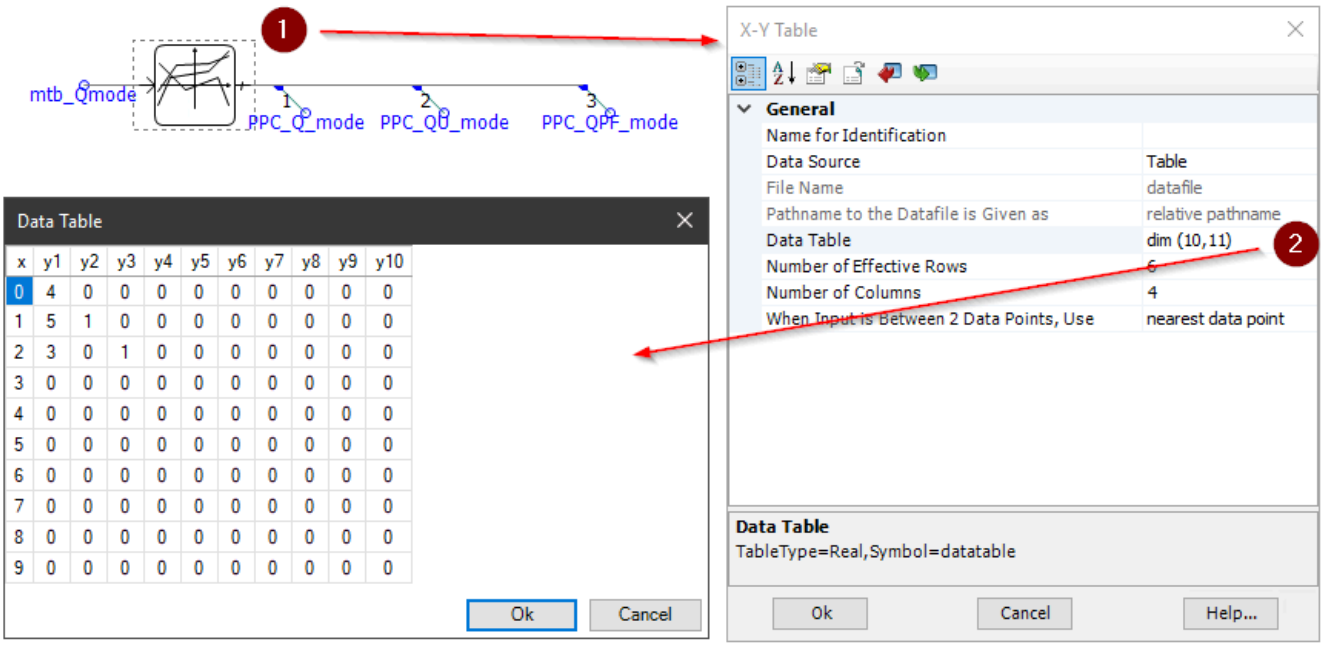
Qref_Qu

If the PPC uses a different voltage base than the MTB, scale the `Qref_Qu` signal accordingly. For example, converting from 152 kV to 161.9 kV gives a factor of `152 / 161.9 = 0.9388`.



Pmode and Qmode

If the PPC expects different integer values for P modes or Q modes, use an X-Y table to map MTB output values to PPC input values.

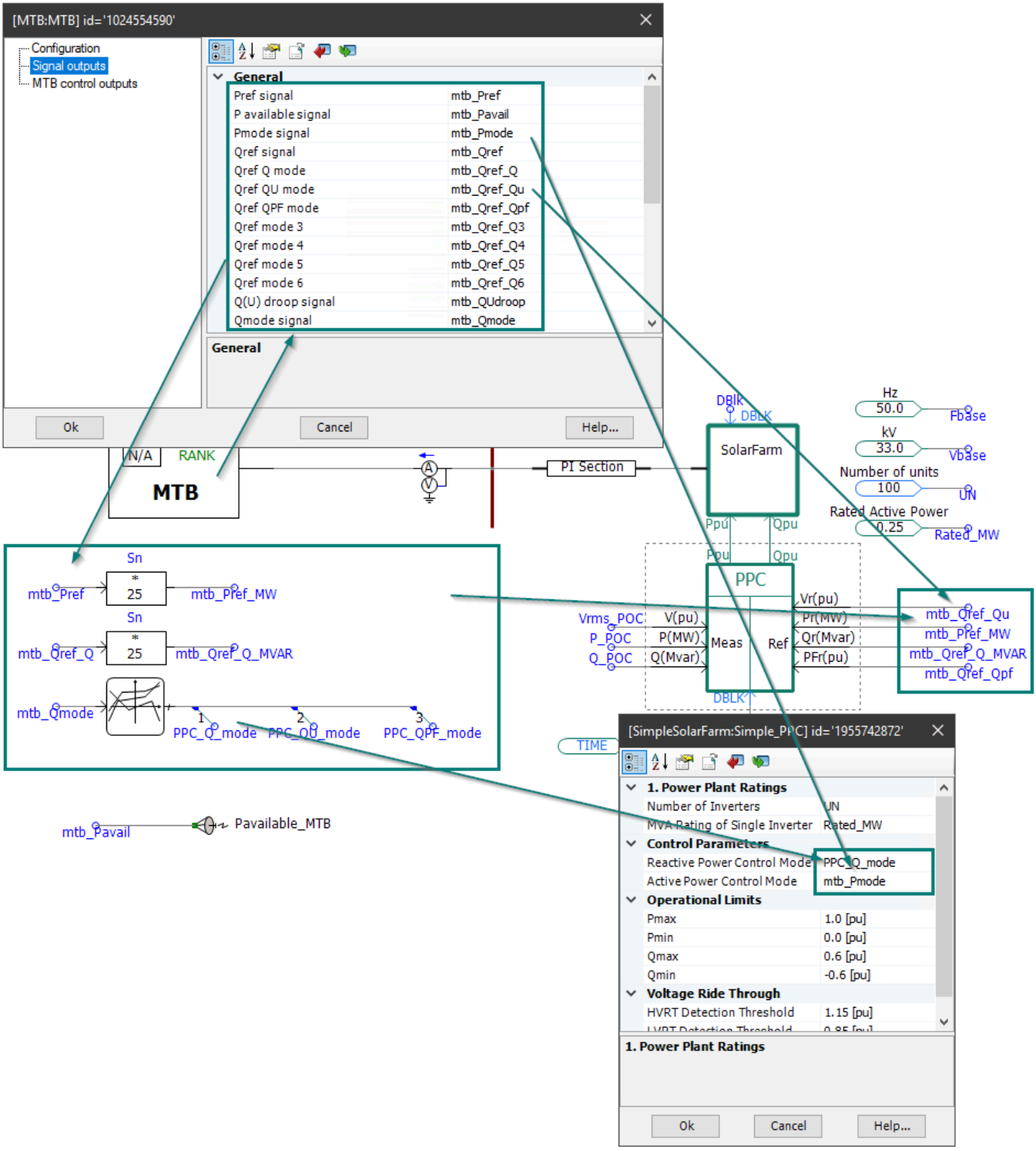


Example Qmode mapping:

Qmode	MTB output	Output tap 1	Output tap 2	Output tap 3
Q	0	4	0	0
U	1	5	1	0
PF	2	3	0	1

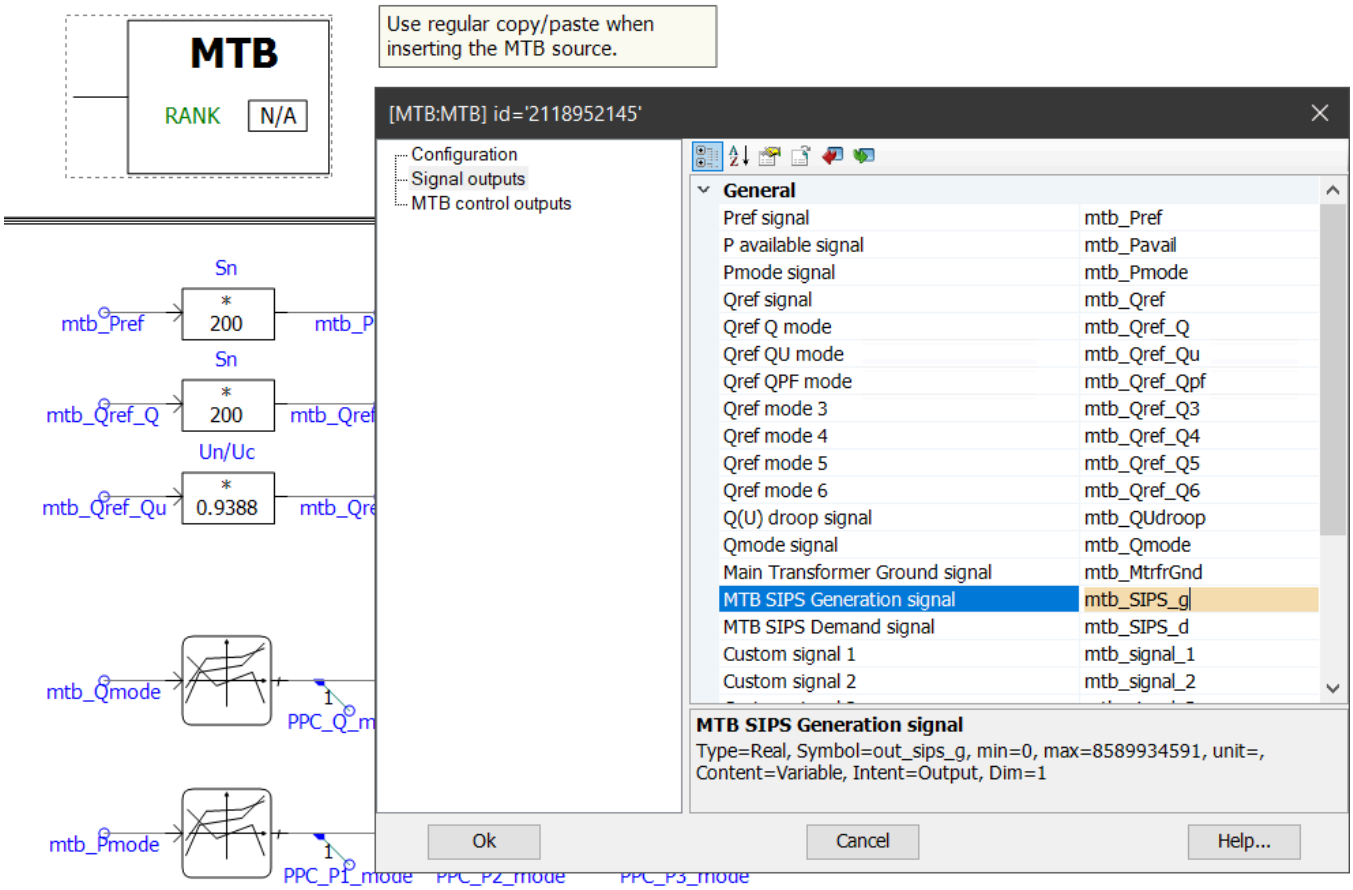
3.3.3.3 Connect signals to the PPC

The exact signal routing is model-specific. In the example, P, Q, Q(U), and PF references are connected on the main canvas, while Qmode and Pmode are connected inside the PPC block.



System protection (SIPS) signals

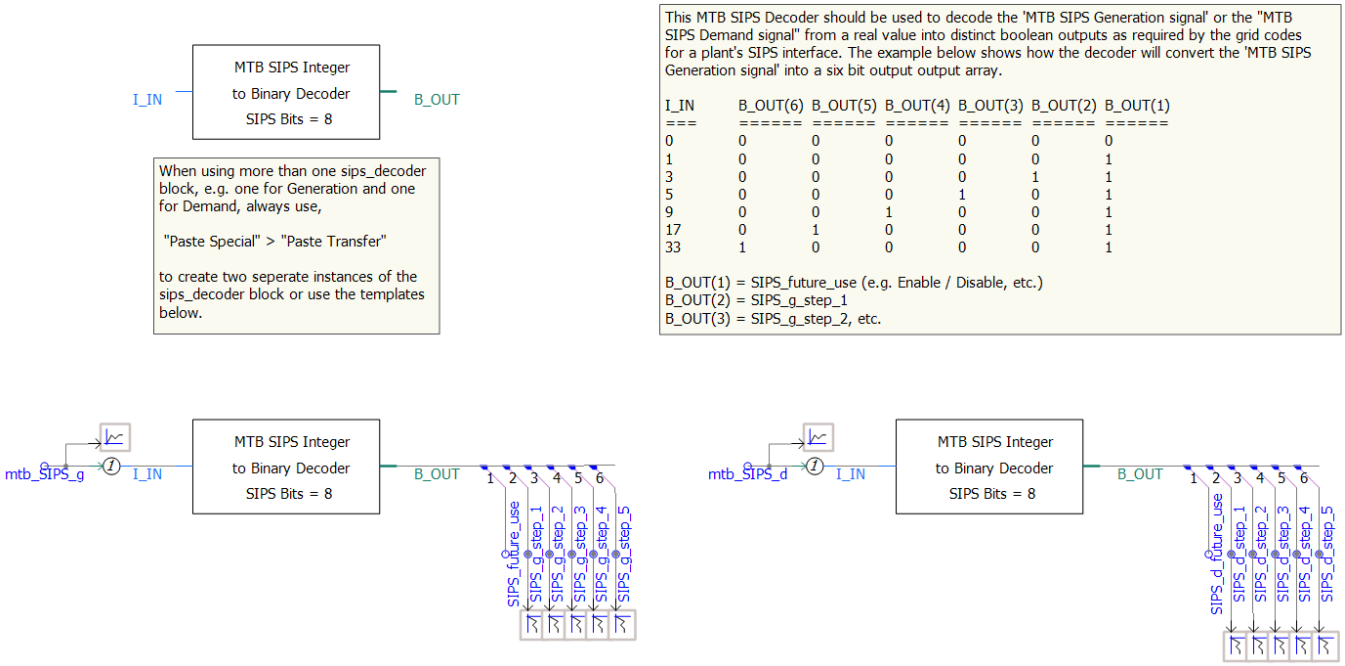
For some plants, a System Integrity Protection Scheme (SIPS) interface is required. MTB supports this through `mtb_SIPS_g` for generation facilities and `mtb_SIPS_d` for demand facilities.



The MTB includes an **MTB SIPS Integer to Binary Decoder** block that decodes `mtb_SIPS_x` into `SIPS_x_step_y` binary signals. The default setup provides five SIPS steps.

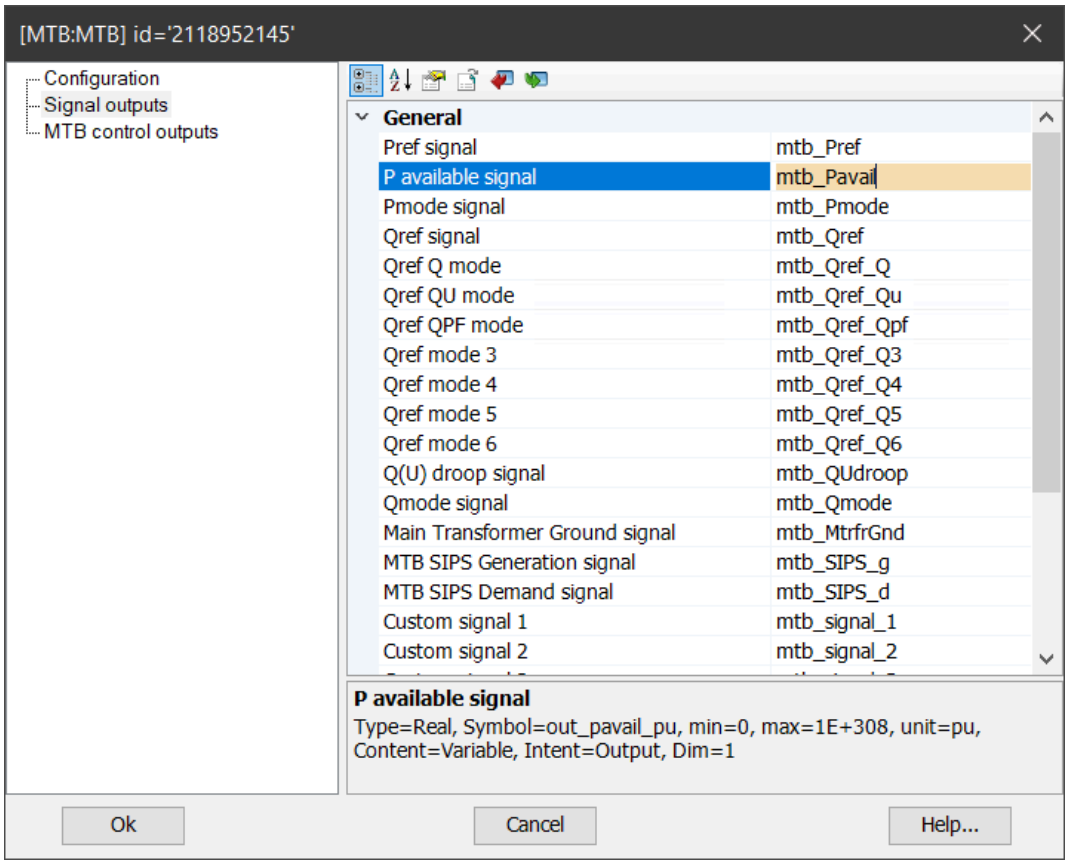
> The facility must be able to set at least five different configurable regulation steps.
>
> The following control ranges are recommended as default values:
>
> 1. Up to 70% of rated power
> 2. Up to 50% of rated power
> 3. Up to 40% of rated power
> 4. Up to 25% of rated power
> 5. Up to 0% of rated power, i.e. the facility has shut down.

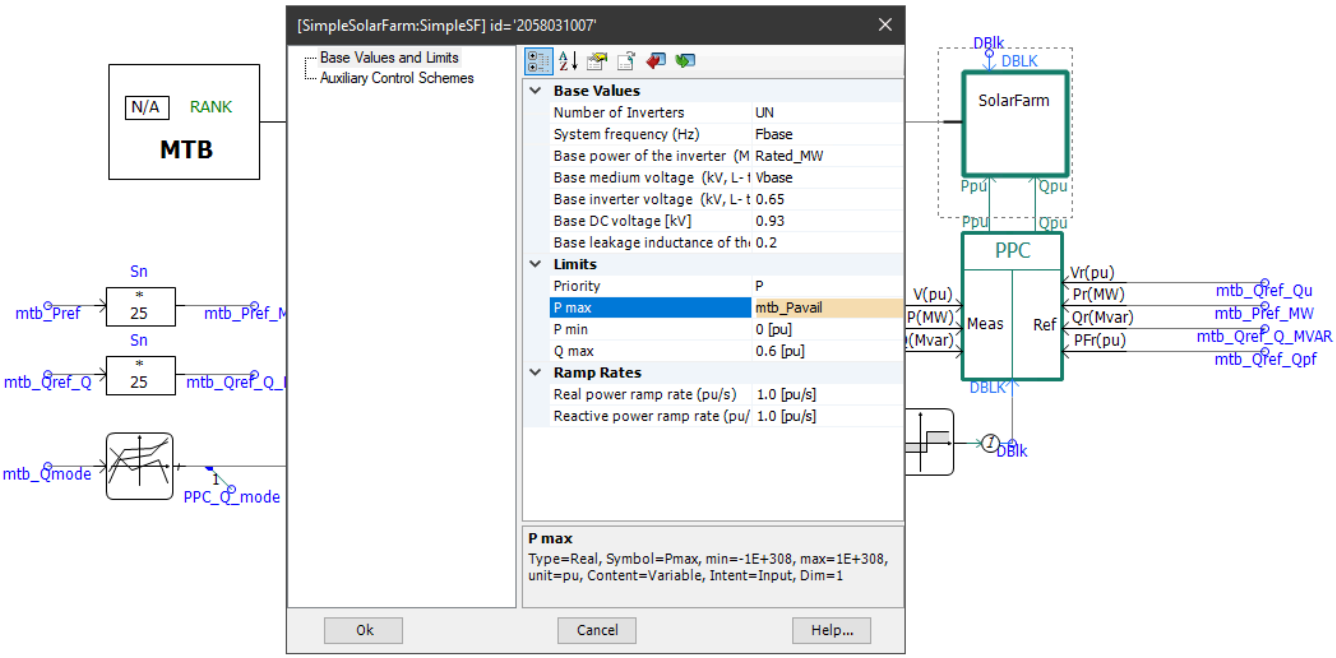
The MTB provides a generic decoder instance and two preconfigured instances with the same signal names used in the plotter figureSetup.csv for generation (RfG) and demand (DCC) cases.



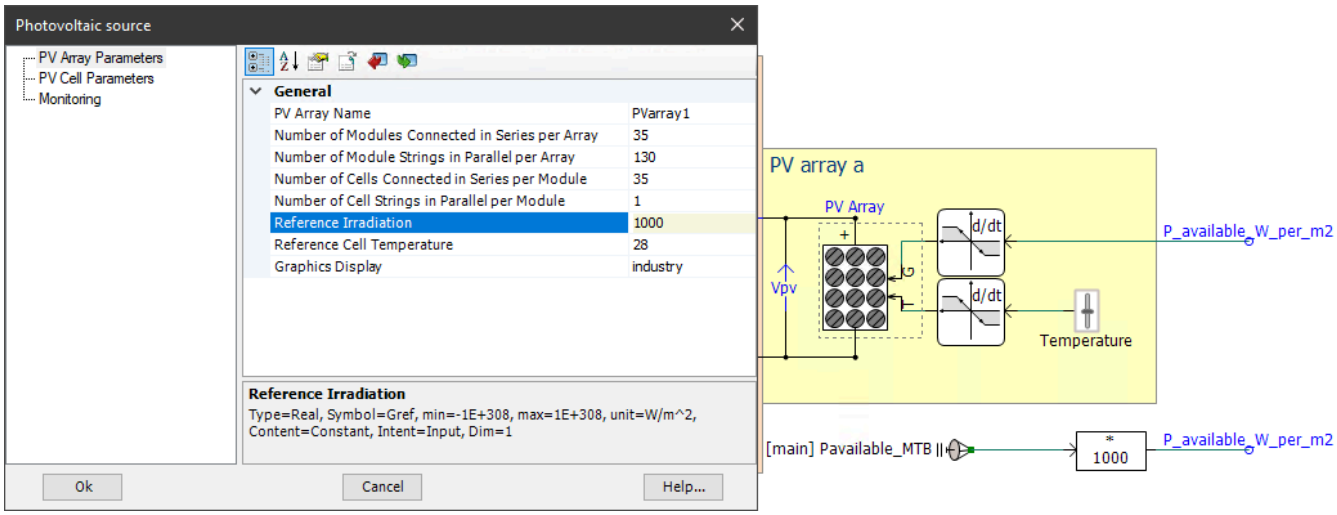
Available power signals

Some MTB cases test changes in available power for solar and wind power plants. From MTB 2.0, a dedicated `mtb_Pavail` signal is provided in pu and can be used to set the upper active power limit.



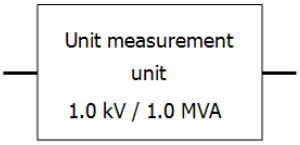


A Wireless Radio Link can also be used to send the `Pavail` signal inside the solar plant and directly control irradiation in W/m².



3.3.4 Connect additional MTB measurements (optional)

The MTB block measures at PoC. Additional measurement points can be added with the **Unit measurement** block from the MTB project.

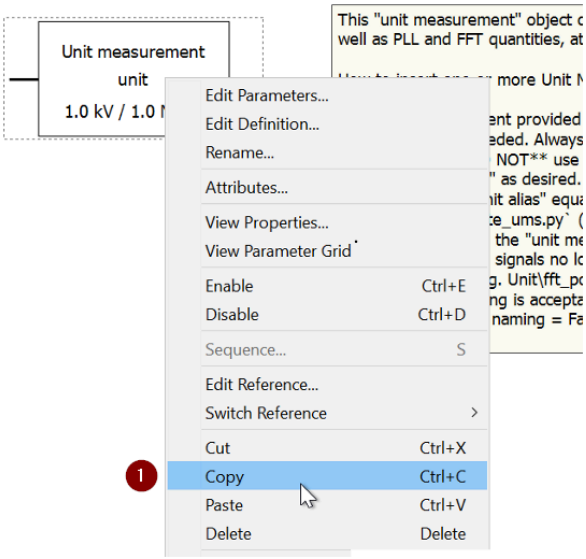


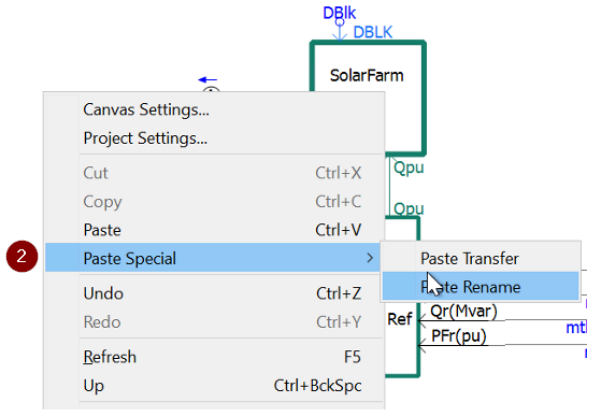
This "unit measurement" object can be used to measure voltage, current, active and reactive power, as well as PLL and FFT quantities, at arbitrary locations.

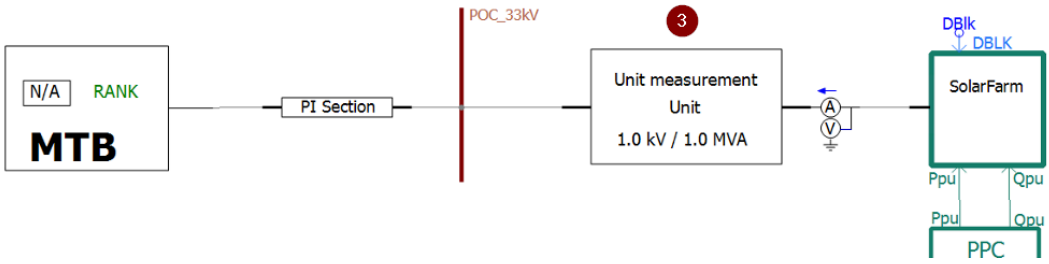
How to insert one or more Unit Measurements block:

1. Copy the component provided above.
2. Paste it where needed. Always use "Paste Special" > "Paste Transfer" when inserting a new unit measurement. **DO NOT** use the standard paste function, as this may cause issues.
3. Set the "Unit alias" as desired.
4. By default the "Unit alias" equals the instance name, i.e. Unit, Unit_1, Unit_2, etc.
5. Run `pscad_update_ums.py` (this script is also triggered by `execute_pscad.py`) to prefix the "Unit alias" to all signals in the "unit measurement" object to create unique signal names
6. For .psout output, signals no longer require unique names as the signals are referenced by the signal path name, e.g. Unit\fft_pos_Vmag_pu, Unit_1\fft_pos_Vmag_pu and Unit_2\fft_pos_Vmag_pu
7. If this signal naming is acceptable, legacy signal naming can be disabled, by setting "Use legacy Unit measurement signal naming = False" in the config.ini file

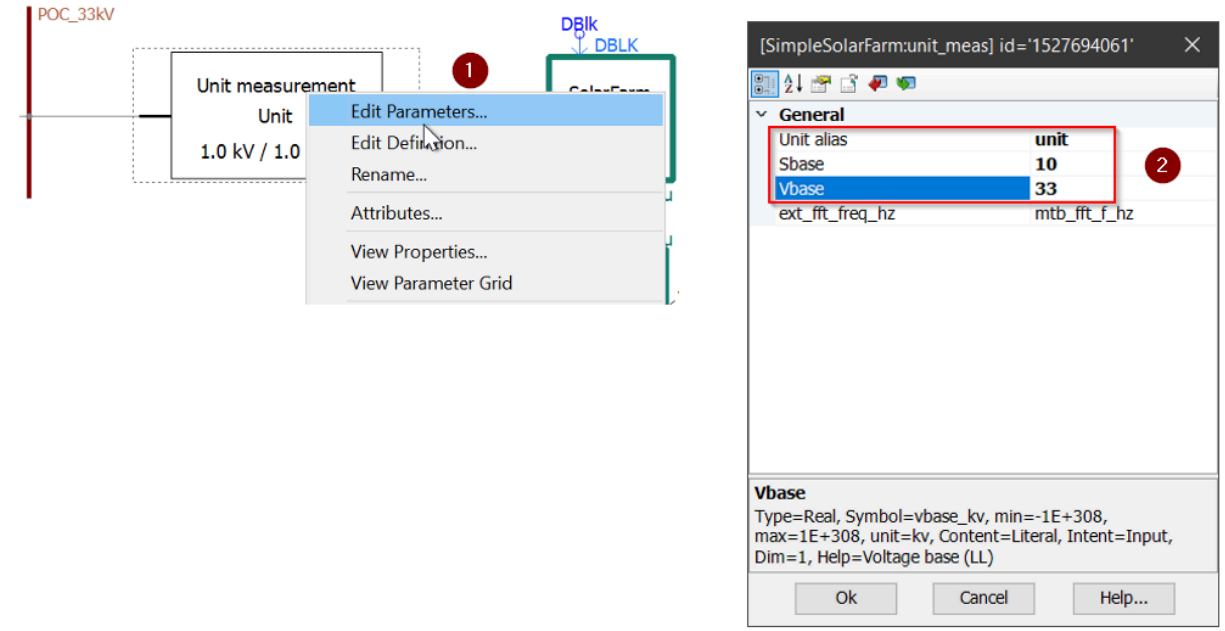
Copy and paste the block into the model and connect it to the point of interest. Use **Paste Special -> Paste Transfer** when pasting the block.

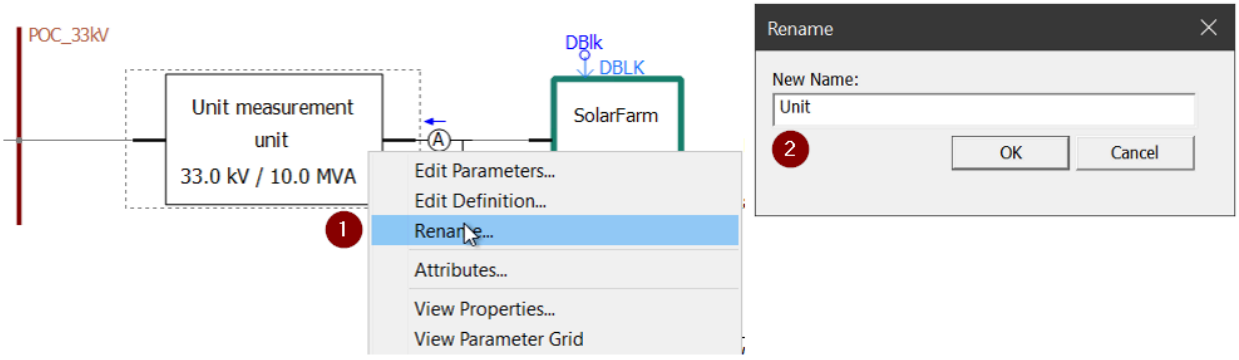






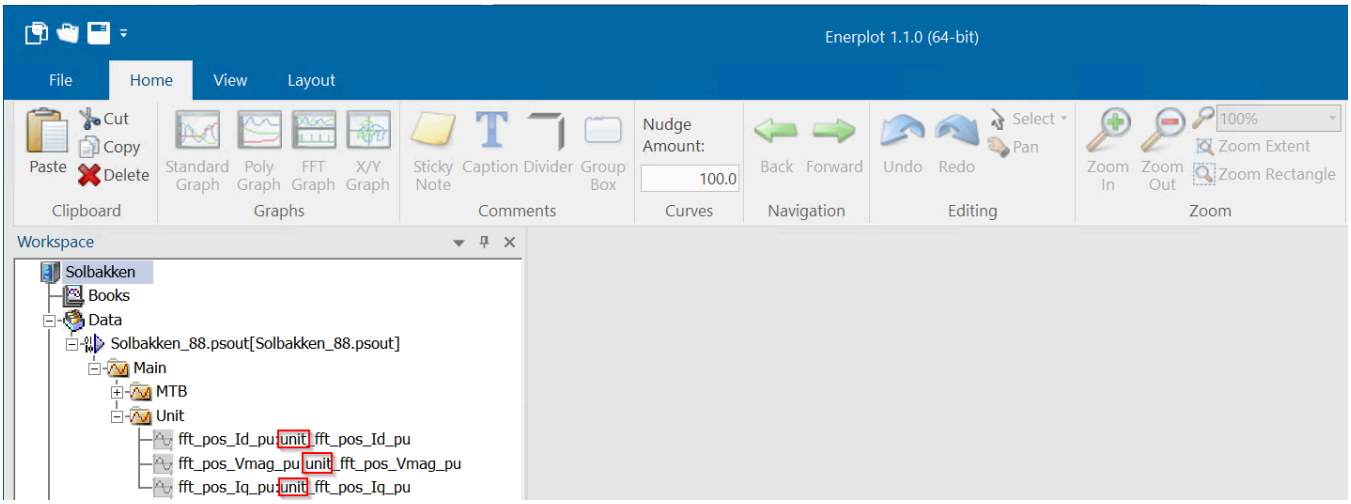
Set the voltage base and apparent power base by editing the block parameters. The `Unit alias` parameter controls the legacy signal-name prefix.

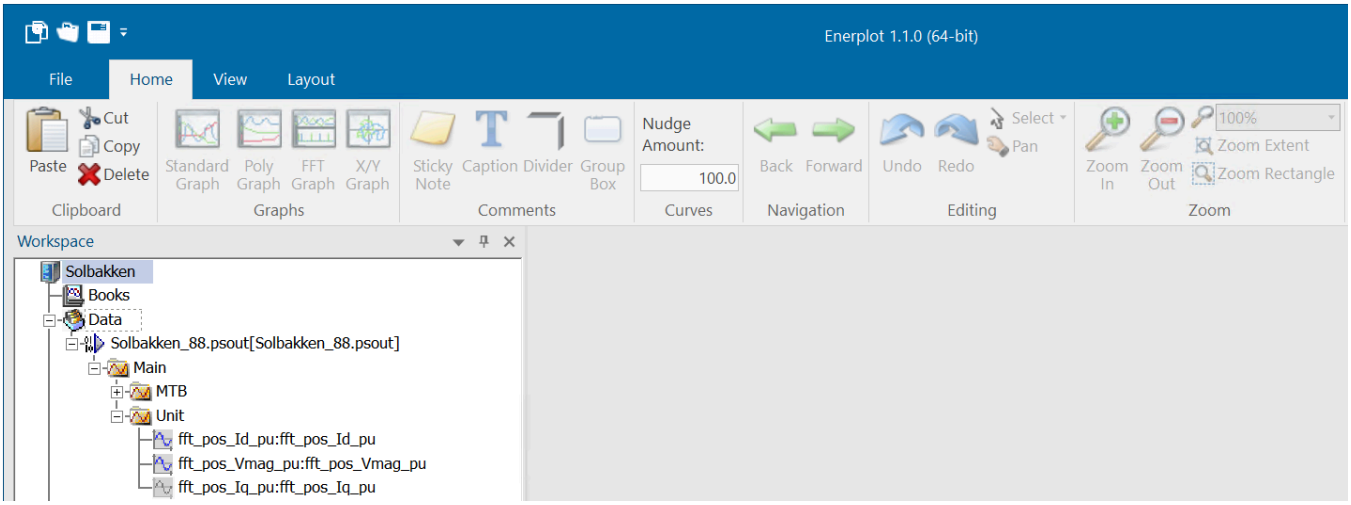




If `Use legacy Unit measurement signal naming = True`, Unit Measurement signals are named with both the Unit Measurement block name and alias. If it is `False`, the alias is ignored because `.psout` signal paths already include the Unit Measurement block instance name.

```
36    # Use legacy Unit measurement signal naming convention, e.g. <unit_name>\<alias>default_unit_measurement_signal_name,
37    # instead of the new convention, e.g. <unit_name>\default_unit_measurement_signal_name
38    # Because .psout files are stored with a signal path name, e.g. Unit, Unit_1, Unit_2, etc, a unique alias is no longer required
39    # to distinguish between signals with the same name in different measurement units.
40    Use legacy Unit measurement signal naming = True
```

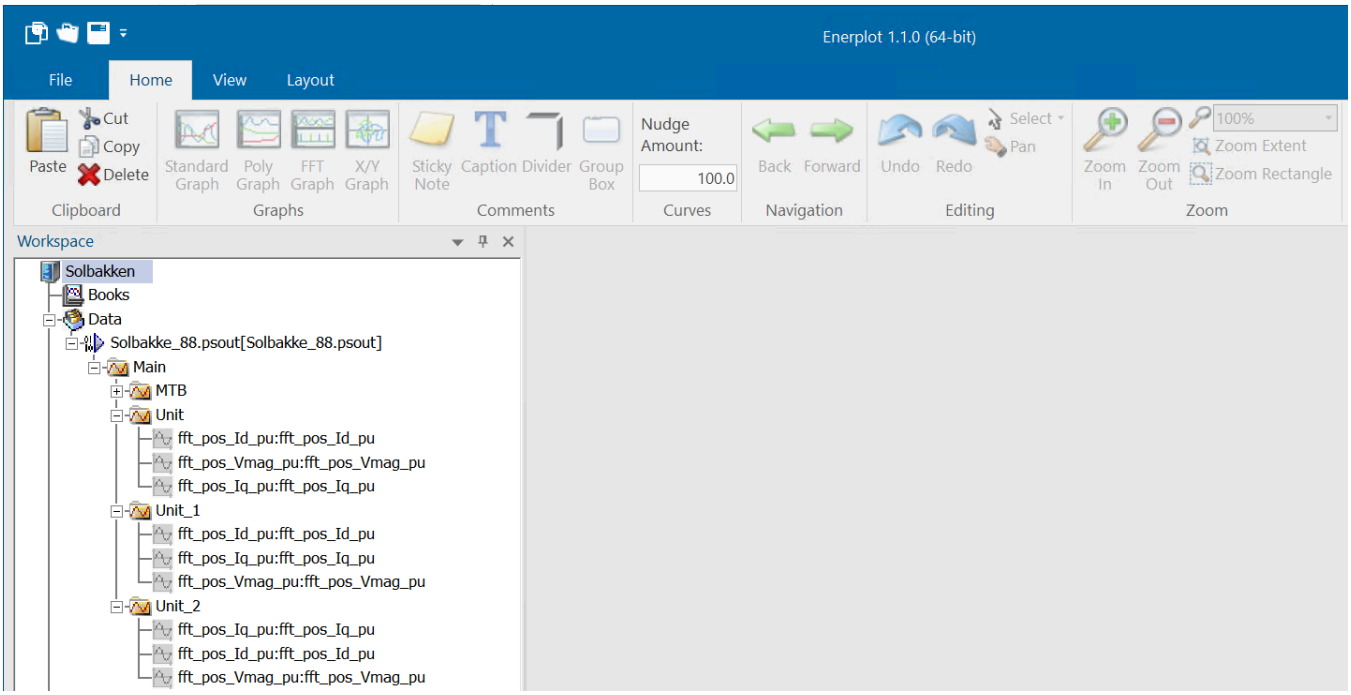




> **IMPORTANT**
>
> If `Disable all unused PGBs = True`, update the plotter figureSetup.csv so any required Unit Measurement signals are listed. Otherwise those PGBs can be disabled before simulation.

```
33    # To reduce the size of the .psout files and reduce the possibility  of an Out of Memory (OoM) error in PSCAD when running
34    # a lot of test cases, it is recommended to only enable # the PGBs of the signals specified in the Plotter's figureSetup.csv file
35    Disable all unused PGBs = True
```

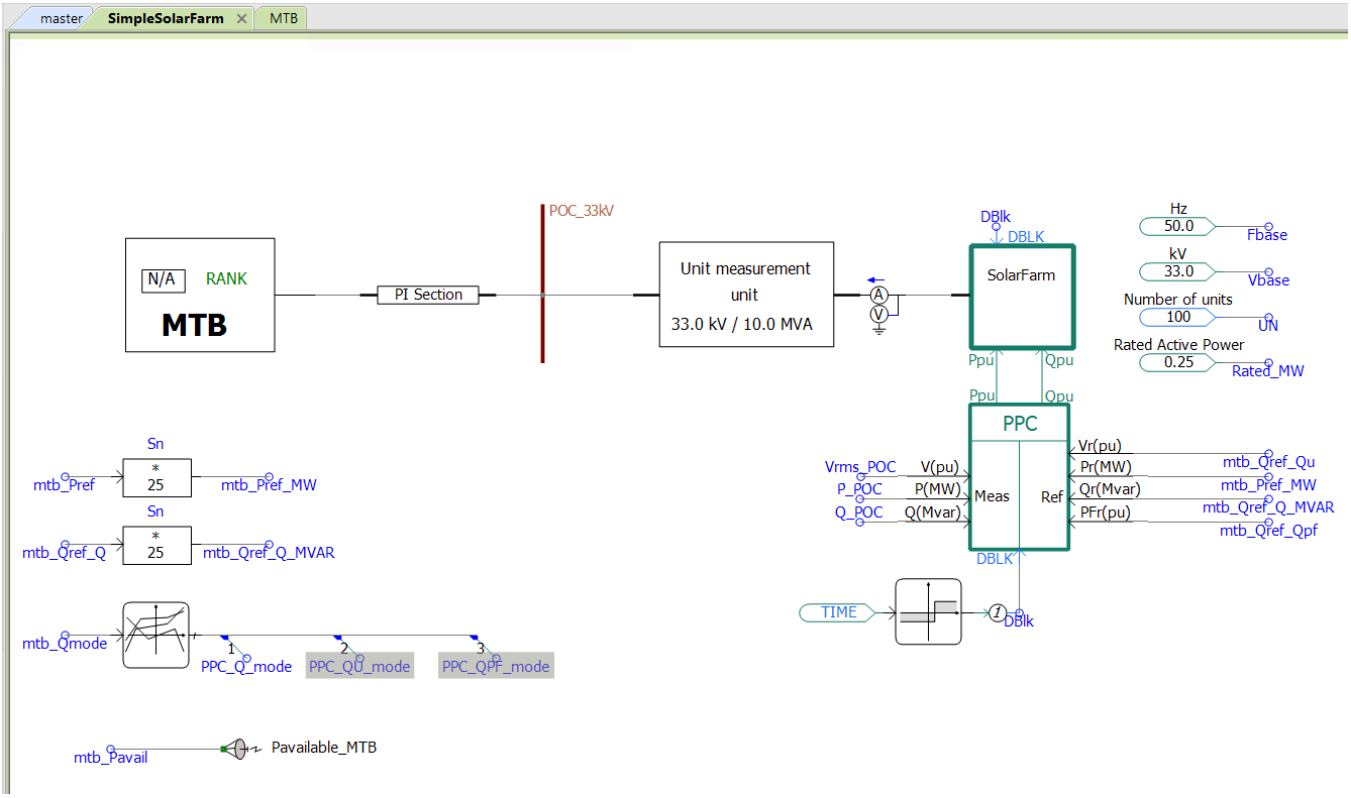
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	figure	title	units	emt_signal_1	emt_signal_2	emt_signal_3	rms_signal_1	rms_signal_2	rms_signal_3	down_sam	gradient_tl	include_in	exclude_in_case	
2	1	Vpp	pu	MTB\meas_Vab_pu	MTB\meas_Vbc_pu	MTB\meas_Vca_pu	meas\s:Vab_pu	meas\s:Vbc_pu	meas\s:Vca_pu	gradient	0.5			
3	2	Vpg	pu	MTB\meas_Vag_pu	MTB\meas_Vbg_pu	MTB\meas_Vcg_pu	meas\s:Vag_pu	meas\s:Vbg_pu	meas\s:Vcg_pu	gradient	0.5			
4	3	Vseq	pu	MTB\fft_pos_Vmag_pu	MTB\fft_neg_Vmag_pu		meas\s:pos_Vmag_pu	meas\s:neg_Vmag_pu		gradient	0.5			
5	4	Itotal	pu	MTB\meas_Ia_pu	MTB\meas_Ib_pu	MTB\meas_Ic_pu	meas\s:Ia_pu	meas\s:Ib_pu	meas\s:Ic_pu	gradient	0.5			
6	5	Iactive	pu	MTB\fft_pos_Id_pu	MTB\fft_neg_Id_pu		meas\s:pos_Id_pu	meas\s:neg_Id_pu		gradient	0.5			
7	6	Ireactive	pu	MTB\fft_pos_Iq_pu	MTB\fft_neg_Iq_pu		meas\s:pos_Iq_pu	meas\s:neg_Iq_pu		gradient	0.5			
8	7	Ppoc	pu	MTB\P_pu_PoC	MTB\mtb_s_pref_pu		meas\s:ppoc_pu			gradient	0.5			
9	8	Qpoc	pu	MTB\Q_pu_PoC	MTB\mtb_s_qref		meas\s:qpoc_pu			gradient	0.5			
10	9	F	Hz	MTB\pll_f_hz			meas\s:f_hz			gradient	0.5			
11	10	Id_pll	pu	MTB\pll_pos_Id_pu	MTB\pll_neg_Id_pu					gradient	0.5			
12	11	Iq_pll	pu	MTB\pll_pos_Iq_pu	MTB\pll_neg_Iq_pu					gradient	0.5			
13	12	Unit	pu	Unit\fft_pos_Id_pu	Unit\fft_pos_Iq_pu	Unit\fft_pos_Vmag_pu	Unit_1\m:i1P:bus1 in p.u.	Unit_1\m:i1Q:bus1 in p.u.	Unit_1\m:u1:bus1 in p.u.	gradient	0.5			
14	12	Unit	pu	Unit_1\fft_pos_Id_pu	Unit_1\fft_pos_Iq_pu	Unit_1\fft_pos_Vmag_pu	Unit_2\m:i1P:bus1 in p.u.	Unit_2\m:i1Q:bus1 in p.u.	Unit_2\m:u1:bus1 in p.u.	gradient	0.5			
15	12	Unit	pu	Unit_2\fft_pos_Id_pu	Unit_2\fft_pos_Iq_pu	Unit_2\fft_pos_Vmag_pu	Unit_3\m:i1P:bus1 in p.u.	Unit_3\m:i1Q:bus1 in p.u.	Unit_3\m:u1:bus1 in p.u.	gradient	0.5			
16	13	Vpg (Inst.)	kV	MTB\meas_Vag_kV	MTB\meas_Vbg_kV	MTB\meas_Vcg_kV				gradient	0.5	1,2,3,4,5,6,7,8,9,10,98		
17	14	Iline (Inst.)	kA	MTB\meas_Ia_kA	MTB\meas_Ib_kA	MTB\meas_Ic_kA				gradient	0.5	1,2,3,4,5,6,7,8,9,10,98		
18														



3.4 Script Execution

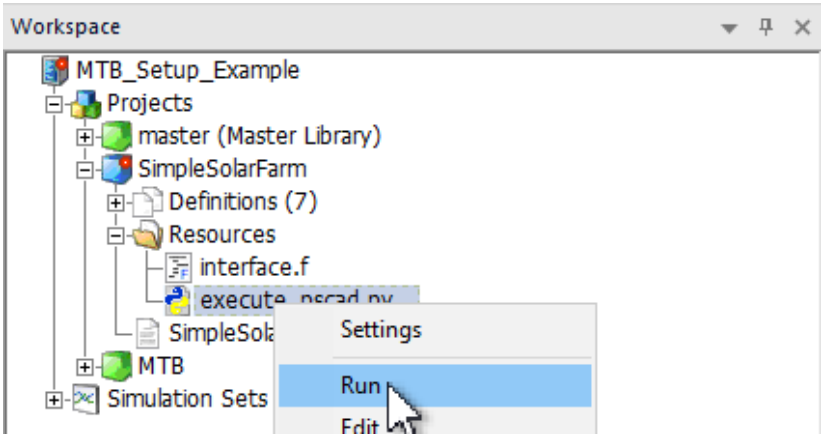
3.4.1 Running all selected cases (Volley mode)

The MTB can run selected cases in parallel. The usable volley count depends on the CPU and the available PSCAD license. The default config.ini sets `Volley = 8`.

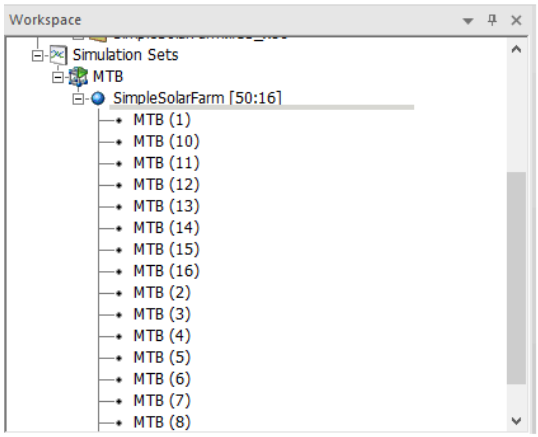


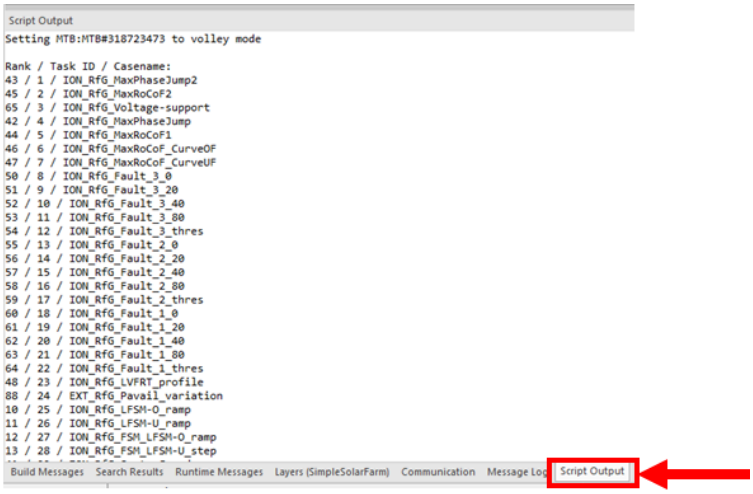

```
11    [PSCAD]
12    # Volley size is the number of simulations to be run in parallel (up to a maximum of 64, license dependent).
13    # This should always be at least 1 less than the number of processors available.
14    Volley = 8
```

To run the simulation from PSCAD, right-click `execute_pscad.py` and select **Run**.



Progress can be followed under **Simulation Sets -> MTB -> <model name></model>**. The PSCAD task number does not necessarily match the case rank from testcases.xlsx, because MTB orders tasks to reduce total simulation time. The script prints the Rank, Task ID, and Case name in the PSCAD Script Output window, and also writes `caseRankTaskID.csv` for OOM recovery.

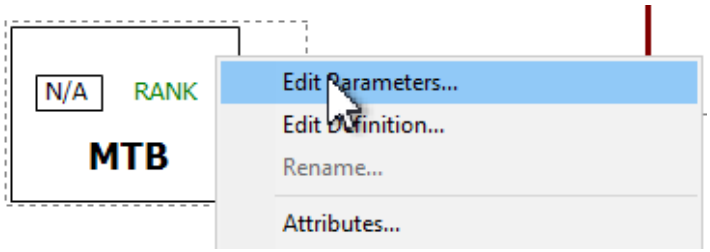


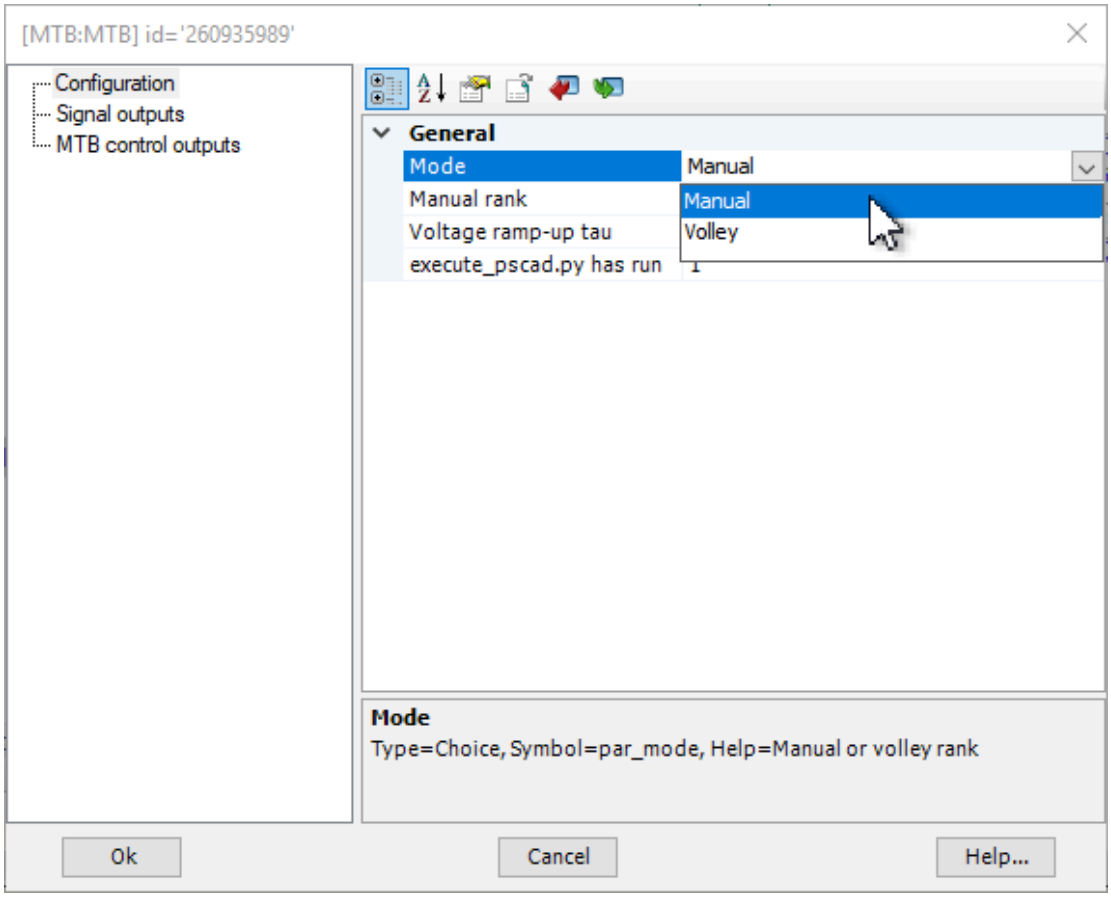


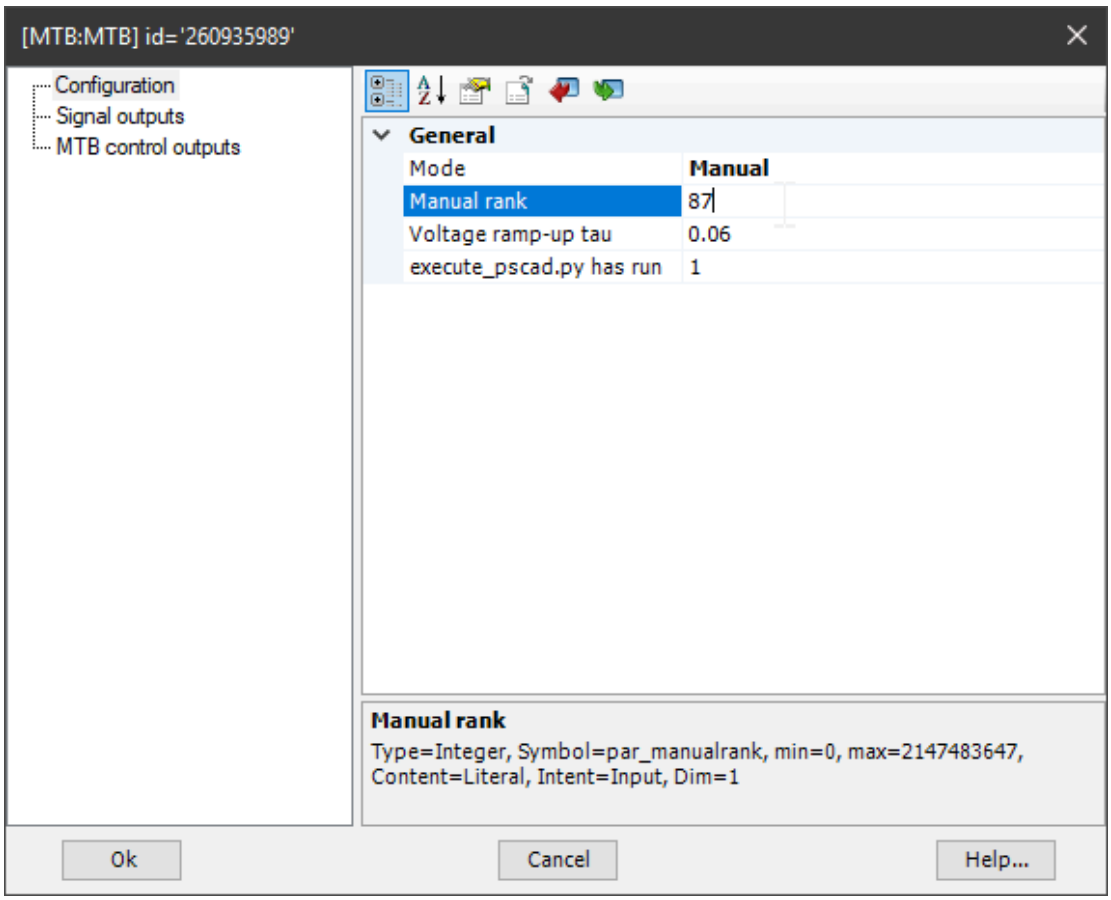
3.4.2 Running individual cases (Manual mode)

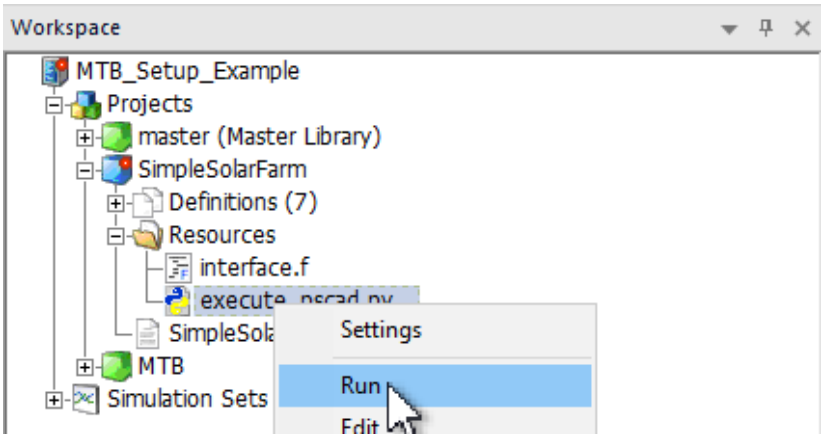
To run a single rank:

1. Right-click the MTB block and select **Edit Parameters...**.
2. Change `Mode` to `Manual`.
3. Select the desired `Manual rank` from testcases.xlsx.
4. Run `execute_pscad.py`.




[MTB:MTB] id='260935989'
Configuration
Signal outputs
MTB control outputs
General
Mode
Manual
Manual rank
Voltage ramp-up tau
execute_pscad.py has run
Manual
Volley
Mode
Type=Choice, Symbol=par_mode, Help=Manual or volley rank
Ok
Cancel
Help...


[MTB:MTB] id='260935989'
Configuration
Signal outputs
MTB control outputs
General
Mode
Manual
Manual rank
87
Voltage ramp-up tau
0.06
execute_pscad.py has run
1
Manual rank
Type=Integer, Symbol=par_manualrank, min=0, max=2147483647,
Content=Literal, Intent=Input, Dim=1
Ok
Cancel
Help...



The `.psout` file for the selected case is saved in a timestamped MTB subfolder inside the configured export folder.

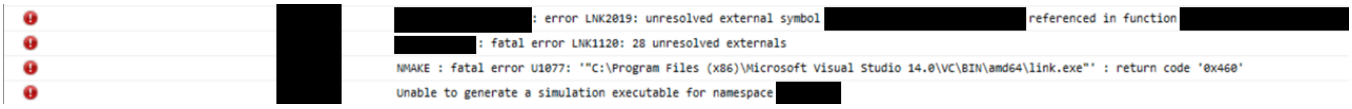
3.5 Script Execution Finished

When the script finishes, result files are available in the configured export folder, by default `export`. The MTB creates `.psout` files that can be plotted with plotter.py and compared with PowerFactory results. See 4. Quickstart Plotter Guide.

3.6 Troubleshooting

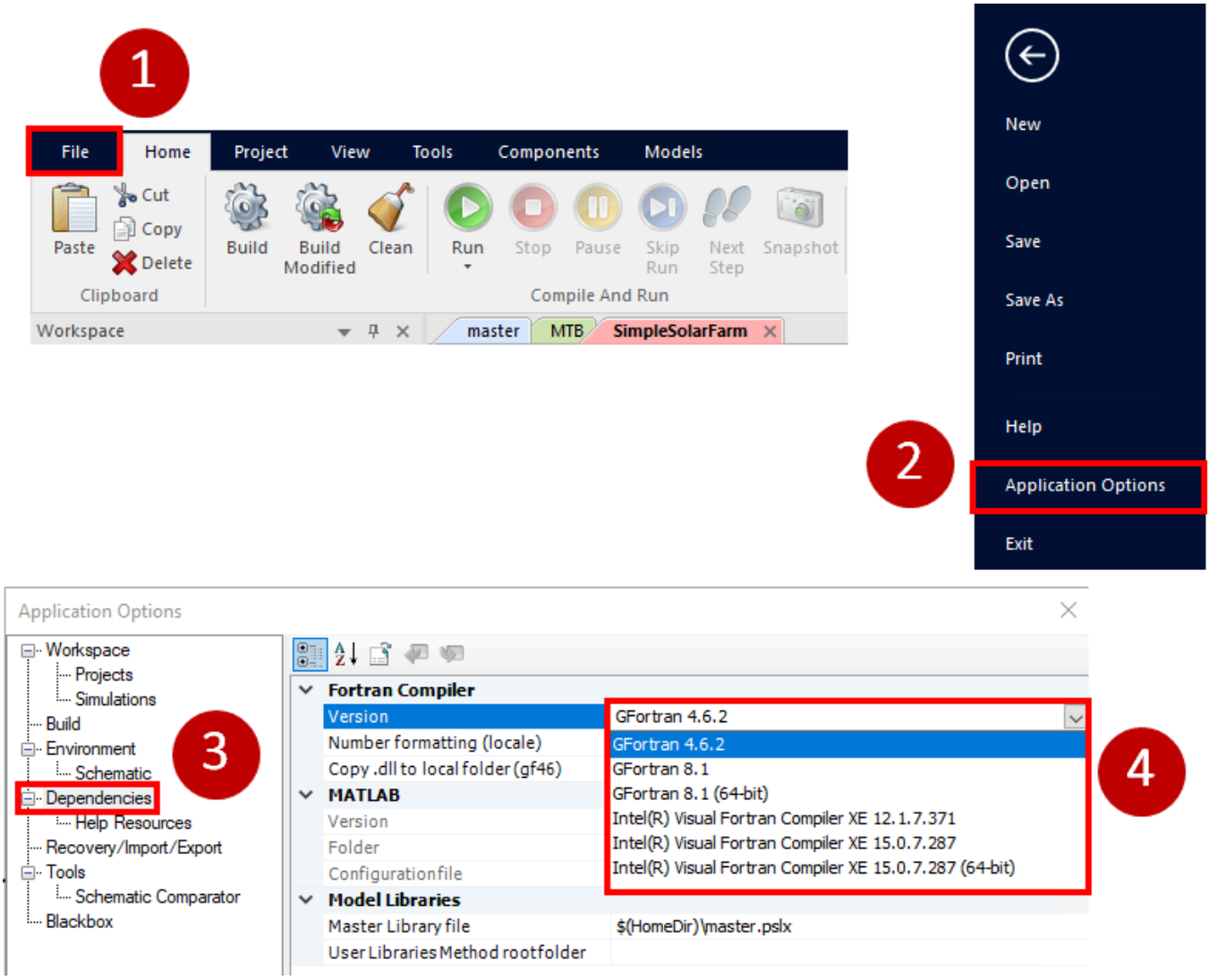
3.6.1 Wrong compiler

Running PSCAD with the wrong compiler can produce errors like these:




```
SimpleSolarFarm  Error: Element in REAL(8) array constructor at (1) is INTEGER(4)
SimpleSolarFarm  Error: Element in REAL(8) array constructor at (1) is INTEGER(4)
SimpleSolarFarm  Error: Element in REAL(8) array constructor at (1) is INTEGER(4)
SimpleSolarFarm  Error: Element in REAL(8) array constructor at (1) is INTEGER(4)
SimpleSolarFarm  Error: Element in REAL(8) array constructor at (1) is INTEGER(4)
```

To change compiler in PSCAD, go to **File -> Application Options -> Dependencies** and select the required Intel Fortran compiler.



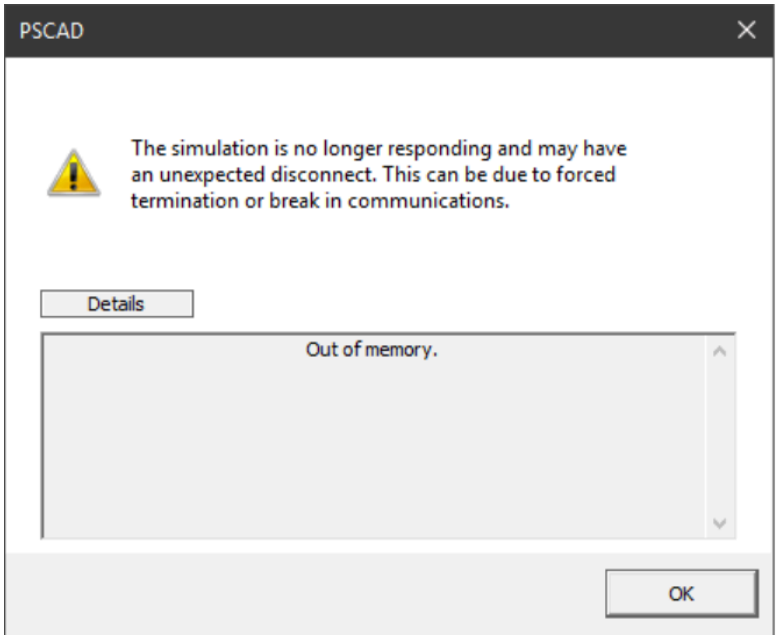
3.6.2 UnicodeDecodeError

If the PSCAD project name or folder path contains Danish letters such as `æ`, `ø`, or `å`, execute_pscad.py may fail with a `UnicodeDecodeError`. Rename the project or folder path to use ASCII characters.

```
Script Output
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc5 in position 30: invalid continuation byte
```

3.6.3 PSCAD Out of Memory (OOM) error

Large PSCAD projects with many signals and many test cases can run into PSCAD out-of-memory errors when generating `.psout` files.



Before running the volley, execute_pscad.py writes `caseRankTaskID.csv` with the mapping between case rank, PSCAD task ID, and case name. If PSCAD crashes, this mapping can be used with recover*psout*files.py to rename recovered `.psout` files from task IDs back to case ranks.

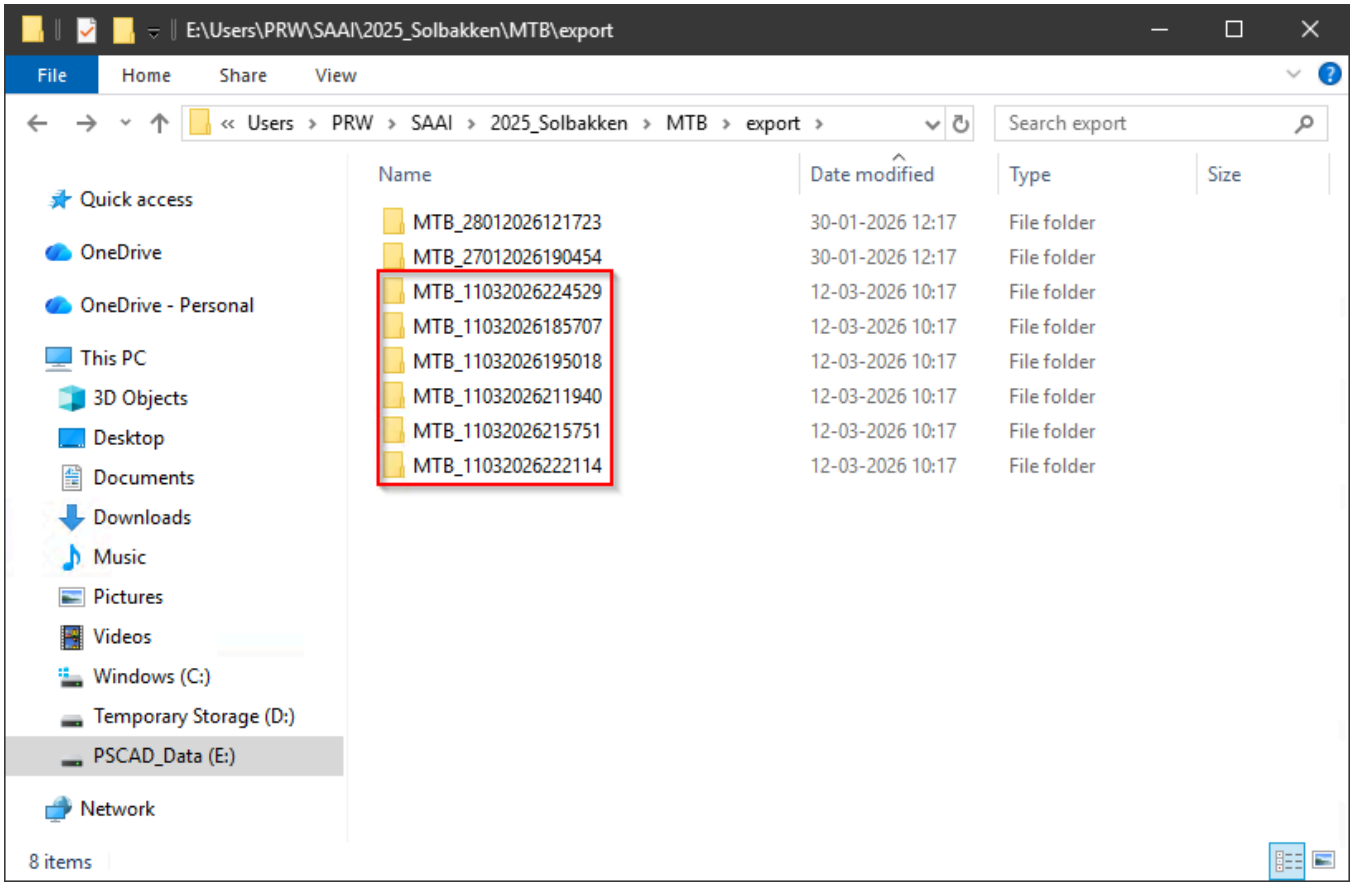
Two workarounds are commonly used when OOM errors occur regularly.

3.6.3.1 Batch execution with smaller test case sets

Use batch*execute*pscad.py to run several smaller `testcases.xlsx` files in sequence:

```
test_case_paths = ['testcases1.xlsx', # e.g. RfG Ranks: 1..44
                   'testcases2.xlsx', # e.g. RfG Ranks: 45..88
                   'testcases3.xlsx', # e.g. RfG Ranks: 89..132
                   'testcases4.xlsx', # e.g. RfG Ranks: 133..176
                   'testcases5.xlsx', # e.g. Custom Ranks: 3001..3044
                   'testcases6.xlsx'] # e.g. Custom Ranks: 3045..3087
```

The output is stored in separate timestamped MTB folders.



3.6.3.2 Disable unused PGBs

Set `Disable all unused PGBs = True` in config.ini to disable Plotter Group Blocks that are not needed for plotting. execute_pscad.py then calls functions from pscad*synchronize*pgbs.py:

Function	Purpose
`getSignalsFromFigureSetup`	Reads EMT signals fromfigureSetup.csv.
`validateFigureSetupAgainstWorkspace`	Verifies that requested signals exist in the PSCAD workspace.
`synchronizePGBsInProject`	Enables required PGBs and disables PGBs that are not required.

The script can also be run from within PSCAD to inspect which signals will be kept or disabled before applying synchronization.

```python
figureSetup = r'.\plotter\figureSetup.csv' # Or set to None to just print status of all PGBs without modifying anything
SYNC = False   # Do a dry run first to review changes before actually applying them with SYNC=True
VERBOSE = True # Print detailed status of each PGB and summary counts at the end of each project

if figureSetup:
    # 1. Load the "Keep" list from CSV
    keep_signals = getSignalsFromFigureSetup(figureSetup)

    # 2. Validate the list against the whole Workspace
    # This catches signal typos regardless of which project they are in.
    missing_total = validateFigureSetupAgainstWorkspace(pscad, keep_signals)

    if not missing_total:
        # 3. Only if the CSV is 100% correct, proceed to modify projects
        case_names = [p['name'] for p in pscad.projects() if p['type'] == 'Case']

        for case_name in case_names:
            proj = pscad.project(case_name)
            # This function (from previous step) now handles one project at a time
            synchronizePGBsInProject(proj, keep_signals, sync=SYNC, verbose=VERBOSE)
    else:
        print("\nAborting: Please fix the signal names in figureSetup.csv before proceeding.")

else:
    # Just print status of all PGBs
    print('No figureSetup.csv file provided. Printing PGB status for all Case projects...')
    case_names = [p['name'] for p in pscad.projects() if p['type'] == 'Case']
    for case_name in case_names:
        proj = pscad.project(case_name)
        printPGBStatus(proj)
```

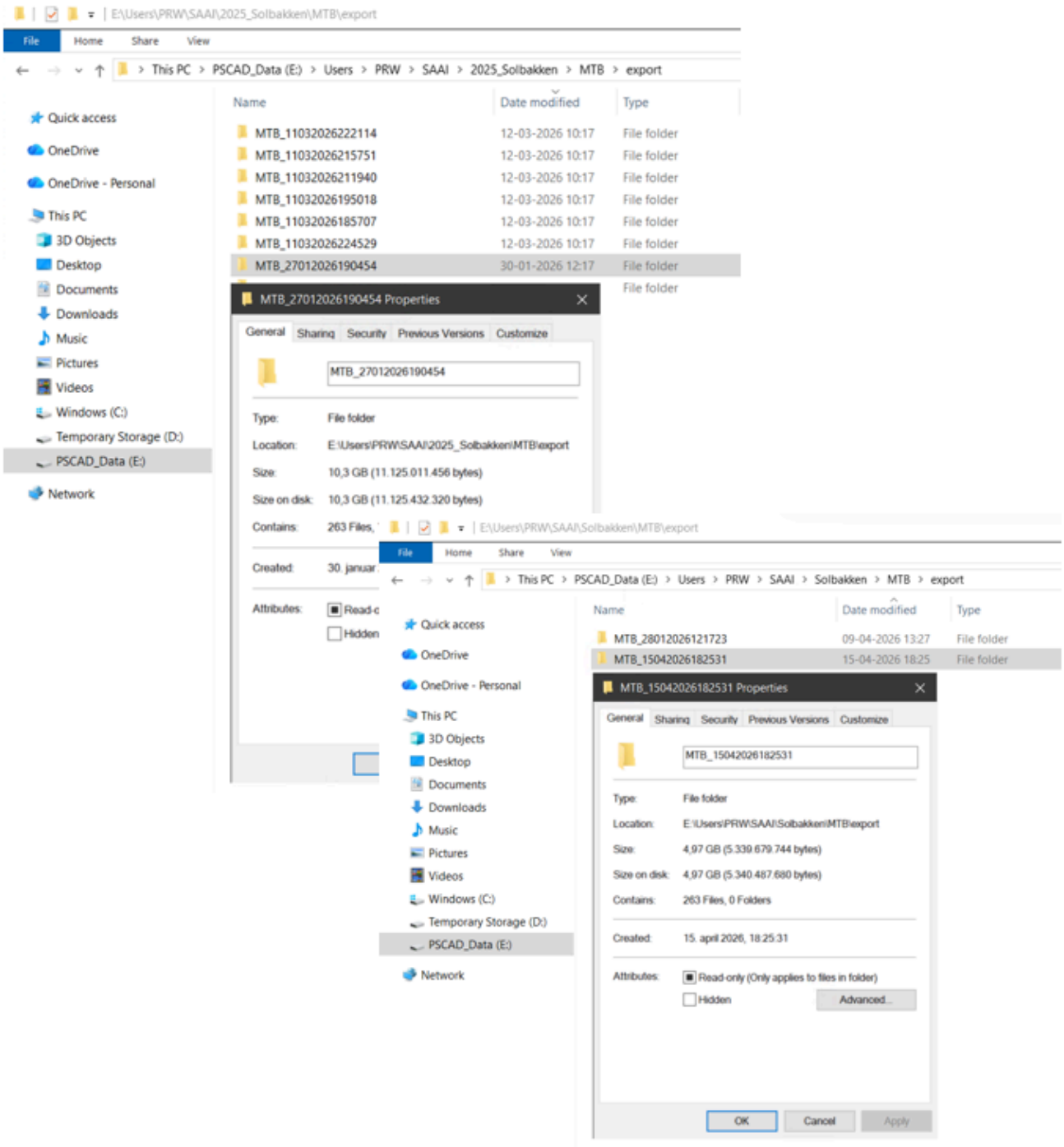
Example output:

```
Success: All signals in figureSetup.csv were located in the workspace.

============================================================
Project: Solbakken
============================================================

Main\PPC\Pcontrol
======================
  Pref [KEEP]
  Pmeas [already disabled]
  Perror [WILL DISABLE]
  Pi [already disabled]
  Pp [already disabled]
  Ppi [WILL DISABLE]
  Pout [KEEP]

============================================================
Total enabled : 132
Total disabled: 88
Will keep     : 10
Will disable  : 123

Project Solbakken DRY RUN: Would disable 123, and enable 1.
============================================================
```

3.6.3.3 Comparing the two OOM workarounds

A test on a TSO plant showed the following trade-off:

Workaround	Advantage	Trade-off
`batch_execute_pscad.py`	All plant signals remain available	Requires splitting `testcases.xlsx`; total result size was 10.3 GB in the test
`Disable all unused PGBs = True`	Can use `testcases.xlsx` as-is; result size was 4.97 GB in the test	Only plant signals listed in `figureSetup.csv` are available for post-processing



4. Quickstart Plotter Guide

4.1 System Requirements

The MTB plotter is located in the plotter folder and is run with Python. The plotter has its own dependency file, plotter/requirements.txt.

Install the plotter dependencies from the `plotter` folder:

```
cd plotter
python -m pip install -r requirements.txt
```

The plotter has been developed and tested with recent Python versions. PSCAD `.psout` support requires the `mhi.psout` package, and static image export requires Plotly/Kaleido.

4.2 Preparation

The MTB plotter creates comparative plots from RMS and EMT simulation results. It can generate HTML reports, image files, analytical guide curves for selected cases, and cursor metric tables.

The most important plotter files are:

File	Purpose
plotter.py	Main plotter script
config.ini	Plotter configuration and simulation result paths
figureSetup.csv	Defines which figures and signals should be plotted
cursorSetup.csv	Defines cursor metrics and cursor time intervals

Before running the plotter, configure at least config.ini. Optionally edit figureSetup.csv and cursorSetup.csv.

4.3 Configuration of config.ini

The plotter configuration file controls output folders, output formats, guide curves, cursor tables, multiprocessing, the testcase spreadsheet, and the result folders to scan.

Example configuration:

```
[config]
resultsDir = results
genHTML = True
genImage = True
genGuide = False
genCursorHTML = True
genCursorPDF = False
imageFormat = png
htmlColumns = 2
imageColumns = 3
htmlCursorColumns = 2
processes = 8
testcaseSheet = ..\testcases.xlsx

[Simulation data paths]
EMT = ..\export\MTB_10122025135456
RMS = ..\export\MTB_23122025140547
```

The current `[config]` parameters are:

Parameter	Type	Default	Description
`resultsDir`	string	`results`	Folder where generated `.html`, image, and cursor output files are stored. The folder is created if it does not exist.
`genHTML`	boolean	`True`	Generate HTML report files.
`genImage`	boolean	`True`	Generate static image files.
`genGuide`	boolean	`False`	Generate analytical guide curves for selected cases. See5. Generation of Guide Curves.
`genCursorHTML`	boolean	`True`	Include cursor metric tables in the HTML output. See6. Cursor Metrics.
`genCursorPDF`	boolean	`False`	Generate separate PDF files containing cursor metric tables.
`imageFormat`	string	`png`	Static image format, for example `png`, `jpg`, or `svg`, depending on Plotly/Kaleido support.
`htmlColumns`	integer	`2`	Number of columns used for HTML figure layout. Must be greater than 0 when `genHTML = True`.
`imageColumns`	integer	`3`	Number of columns used for static image layout. Must be greater than 0 when `genImage = True`.
`htmlCursorColumns`	integer	`2`	Number of columns used for cursor tables in HTML output.
`processes`	integer	`8`	Number of concurrent Python processes used to generate result files. Must be greater than 0. Use `1` for sequential execution and easier debugging.

Parameter	Type	Default	Description
`testcaseSheet`	string	`..\testcases.xlsx`	Path to the `testcases.xlsx` file used to generate the simulation results. This is mandatory for MTB 2.x.

> **NOTE**
>
> If `htmlColumns > 3`, the HTML output is created as a subplot with a common legend instead of separate figures with individual legends.

> **IMPORTANT**
>
> From MTB 2.0, `testcaseSheet` must point to the same `testcases.xlsx` file that was used to generate the PowerFactory and PSCAD simulation results. The plotter uses it for case names, guide curves, and cursor-related context.

The `[Simulation data paths]` section defines the RMS and EMT result folders that should be plotted:

```
[Simulation data paths]
EMT = ..\export\MTB_10122025135456
RMS = ..\export\MTB_23122025140547
```

Each entry is read as:

Left side	Right side
Legend/group name shown in the plot output	Path to a folder containing result files

> **TIP**
>
> Add more lines under `[Simulation data paths]` to compare multiple model versions, for example `RMS_V04`, `RMS_V05`, `EMT_V04`, and `EMT_V05`.

The plotter automatically categorizes result files found in the configured paths:

Type	Enum	File pattern / detection
`RMS`	0	`.csv` file where the second line starts with `"b:tnow in s"`
`EMT_INF`	1	`.inf` file where the first line starts with `PGB(1)`; associated `.csv` data files are expected in the same folder
`EMT_PSOUT`	2	`.psout` file
`EMT_CSV`	3	`.csv` file where the first line starts with `time;`
`EMT_ZIP`	4	`.zip`, `.gz`, `.bz2`, or `.xz` compressed EMT CSV output

> **NOTE**
>
> The plotter now determines the MTB path in `.psout` files automatically by searching for configured MTB signals. A separate `psoutPathMTB` setting is no longer required.

4.4 Configuration of figureSetup.csv

The figure setup file, figureSetup.csv, defines the plot figures and the RMS/EMT signals shown in each figure. A maximum of three EMT signals and three RMS signals can be specified per result file for each figure.

Column	Description
`figure`	Figure number. Currently used as an identifier in the setup file.
`title`	Figure title used in the plot output.
`units`	Units displayed on the y-axis.
`emt_signal_1`	First EMT signal to plot.
`emt_signal_2`	Second EMT signal to plot.
`emt_signal_3`	Third EMT signal to plot.
`rms_signal_1`	First RMS signal to plot.
`rms_signal_2`	Second RMS signal to plot.
`rms_signal_3`	Third RMS signal to plot.
`down_sampling_method`	Down-sampling method: `gradient`, `amount`, or `no_down_sampling`.
`gradient_threshold`	Threshold used when `down_sampling_method = gradient`.
`include_in_case`	Comma-separated list of ranks where this figure should be included. If blank, the figure is global.
`exclude_in_case`	Comma-separated list of ranks where this figure should be excluded.

The plotter first builds a list of global figures where `include_in_case` is blank. For each rank mentioned in `include_in_case` or `exclude_in_case`, it starts with those global figures, adds specifically included figures, and removes specifically excluded figures.

> **TIP**
>
> If several RMS and EMT result folders are configured, a figure can quickly contain many traces. In that case, consider plotting fewer signals per figure to keep the legend readable.

> **WARNING**
>
> If `down_sampling_method = no_down_sampling`, generated HTML files can become large and slow to load.

> **NOTE**
>
> The CSV file uses Danish regional settings, so `;` must be used as the field separator. If the file is saved with comma delimiters, `plotter.py` will not read it correctly.

The default figureSetup.csv can be edited in Microsoft Excel:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	figure	title	units	emt_signal_1	emt_signal_2	emt_signal_3	rms_signal_1	rms_signal_2	rms_signal_3	down_samp	gradien	include_in_case	exclude_in_case
2	1	Vpp	pu	MTB\meas_Vab_pu	MTB\meas_Vbc_pu	MTB\meas_Vca_pu	meas\s:Vab_pu	meas\s:Vbc_pu	meas\s:Vca_pu	gradient	0.5		
3	2	Vpg	pu	MTB\meas_Vag_pu	MTB\meas_Vbg_pu	MTB\meas_Vcg_pu	meas\s:Vag_pu	meas\s:Vbg_pu	meas\s:Vcg_pu	gradient	0.5		
4	3	Vseq	pu	MTB\fft_pos_Vmag_pu	MTB\fft_neg_Vmag_pu		meas\s:pos_Vmag_pu	meas\s:neg_Vmag_pu		gradient	0.5		
5	4	Itotal	pu	MTB\meas_Ia_pu	MTB\meas_Ib_pu	MTB\meas_Ic_pu	meas\s:Ia_pu	meas\s:Ib_pu	meas\s:Ic_pu	gradient	0.5		
6	5	Iactive	pu	MTB\fft_pos_Id_pu	MTB\fft_neg_Id_pu		meas\s:pos_Id_pu	meas\s:neg_Id_pu		gradient	0.5		
7	6	Ireactive	pu	MTB\fft_pos_Iq_pu	MTB\fft_neg_Iq_pu		meas\s:pos_Iq_pu	meas\s:neg_Iq_pu		gradient	0.5		
8	7	Ppoc	pu	MTB\P_pu_PoC	MTB\mtb_s_pref_pu		meas\s:ppoc_pu			gradient	0.5		
9	8	Qpoc	pu	MTB\Q_pu_PoC	MTB\mtb_s_qref		meas\s:qpoc_pu			gradient	0.5		
10	9	F	Hz	MTB\pll_f_hz			meas\s:f_hz			gradient	0.5		
11	10	Id_pll	pu	MTB\pll_pos_Id_pu	MTB\pll_neg_Id_pu					gradient	0.5		
12	11	Iq_pll	pu	MTB\pll_pos_Iq_pu	MTB\pll_neg_Iq_pu					gradient	0.5		
13	12	Unit	pu	Unit\unit_fft_pos_Id_pu	Unit\unit_fft_pos_Iq_pu	Unit\unit_fft_pos_Vr	Unit_1\m:i1P:bus1 in p.u.	Unit_1\m:i1Q:bus1 in p.u.	Unit_1\m:u1:bus1 in p.u.	gradient	0.5		
14	13	Vpg (Inst.)	kV	MTB\meas_Vag_kV	MTB\meas_Vbg_kV	MTB\meas_Vcg_kV				gradient	0.5	1,2,3,4,5,6,7,8,9,10,98	
15	14	Iline (Inst.)	kA	MTB\meas_Ia_kA	MTB\meas_Ib_kA	MTB\meas_Ic_kA				gradient	0.5	1,2,3,4,5,6,7,8,9,10,98	
16	15	SIPS (G1)	Integer/Binary	mtb_SIPS_g	SIPS_g_step_1	SIPS_g_step_2	MTB\MTB\mtb_s_sips_g.El	MTB\MTB\mtb_sips_decod	MTB\MTB\mtb_sips_deco	gradient	0.5	57,58,59,177,178	
17	16	SIPS (G2)	Binary	SIPS_g_step_3	SIPS_g_step_4	SIPS_g_step_5	MTB\MTB\mtb_sips_decoc	MTB\MTB\mtb_sips_decod	MTB\MTB\mtb_sips_deco	gradient	0.5	57,58,59,177,178	
18	17	SIPS (D1)	Integer/Binary	mtb_SIPS_d	SIPS_d_step_1	SIPS_d_step_2	MTB\MTB\mtb_s_sips_d.El	MTB\MTB\mtb_sips_decod	MTB\MTB\mtb_sips_deco	gradient	0.5	1043,1044,1087,1088	
19	18	SIPS (D2)	Binary	SIPS_d_step_3	SIPS_d_step_4	SIPS_d_step_5	MTB\MTB\mtb_sips_decoc	MTB\MTB\mtb_sips_decod	MTB\MTB\mtb_sips_deco	gradient	0.5	1043,1044,1087,1088	

The default figure setup includes signal selections for:

- Phase-phase voltages at PoC
- Phase-ground voltages at PoC
- Positive- and negative-sequence voltages at PoC
- Total currents at PoC
- Id and Iq currents at PoC
- Active and reactive power at PoC
- Frequency at PoC
- Id and Iq currents measured by the PLL
- Id and Iq currents and positive-sequence voltage at terminals
- Direct measurements at generating-unit terminals using the Unit block in PSCAD and PowerFactory
- Instantaneous voltage and current from the EMT model

4.5 Configuration of cursorSetup.csv

The cursor setup file, cursorSetup.csv, defines cursor metric tables for selected ranks. It is only used when `genCursorHTML = True` or `genCursorPDF = True`.

The detailed cursor metric setup is described in 6. Cursor Metrics.

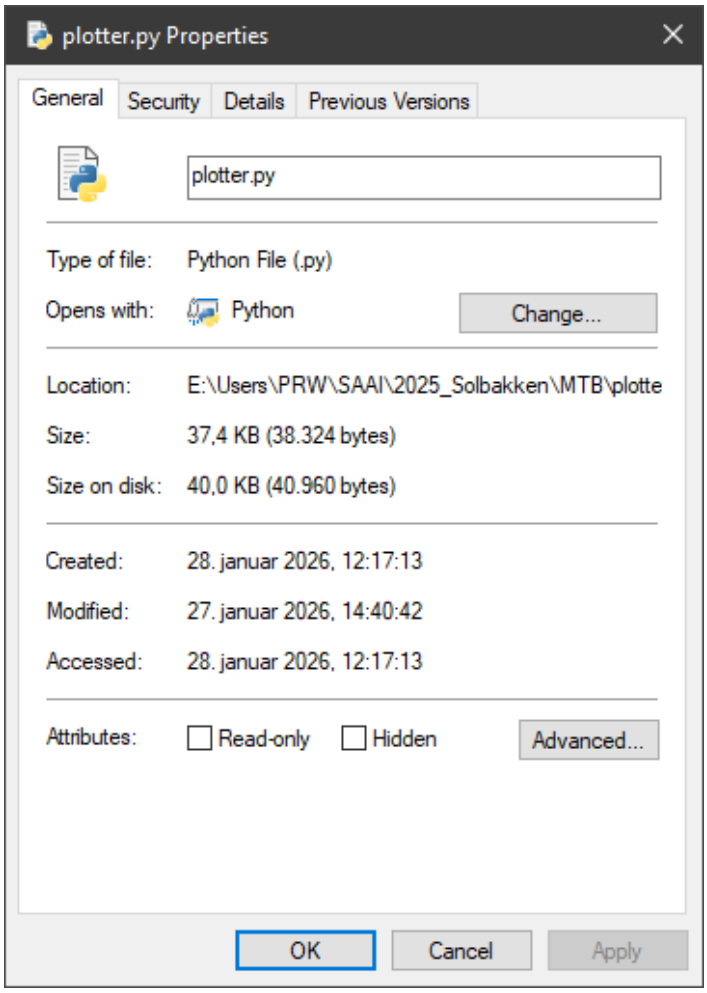
4.6 Script Execution

Run the plotter from the `plotter` folder after config.ini, and optionally figureSetup.csv and cursorSetup.csv, have been configured:

```
cd plotter
python plotter.py
```

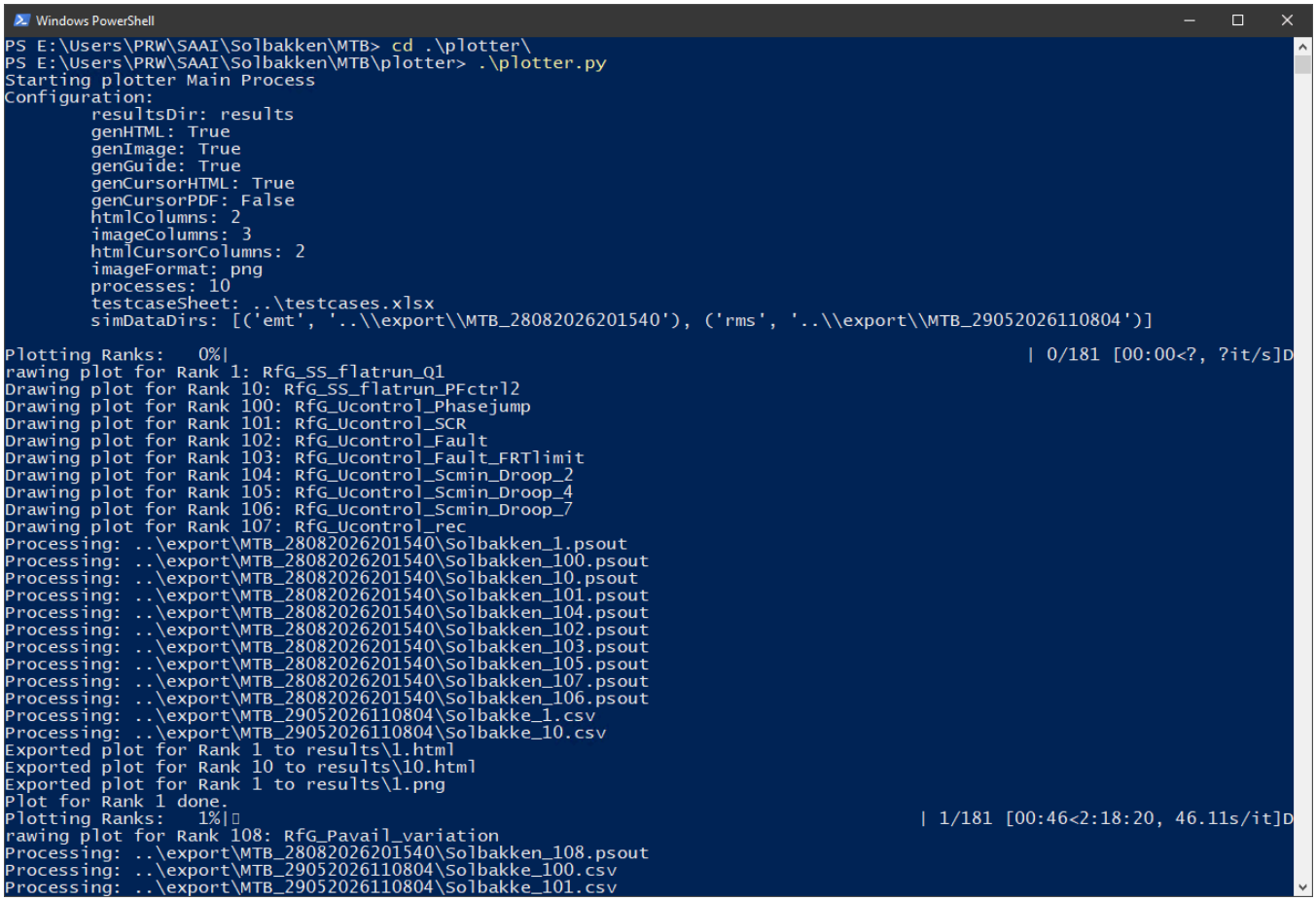
The generated HTML, image, and cursor result files are written to the folder defined by `resultsDir` in config.ini.

If the `.py` extension is associated with a Python interpreter, double-clicking `plotter.py` opens a new console window and starts the script:

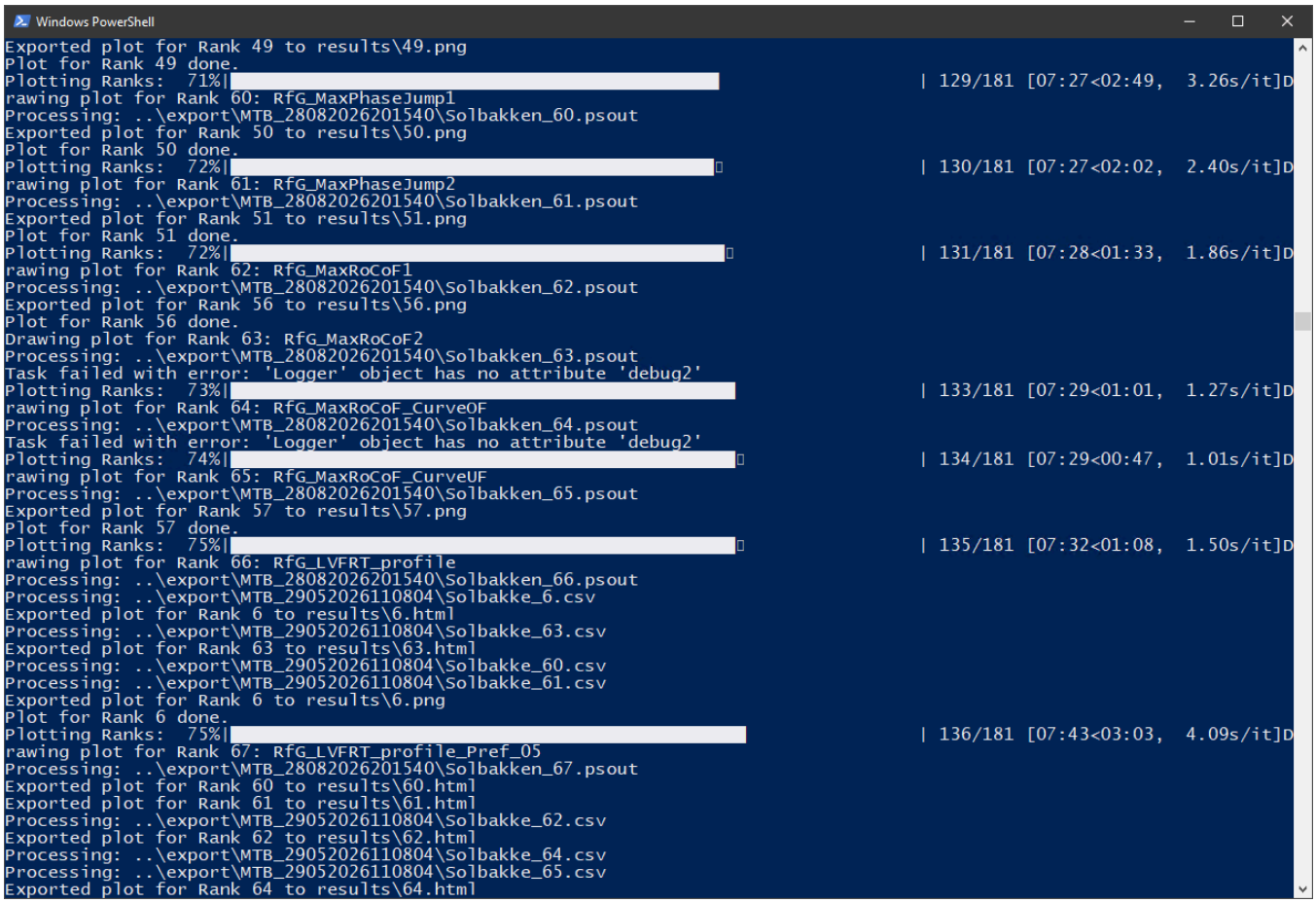


> **NOTE**
>
> The console window may close automatically when the script finishes. Running `python plotter.py` from an already-open console is usually better, especially when debugging.

The script first loads and prints the configuration, then scans the configured simulation data paths and maps recognized result files by rank. Depending on `processes`, the plotter either processes ranks sequentially or distributes them across multiple Python processes.



When `processes > 1`, a progress bar is displayed:



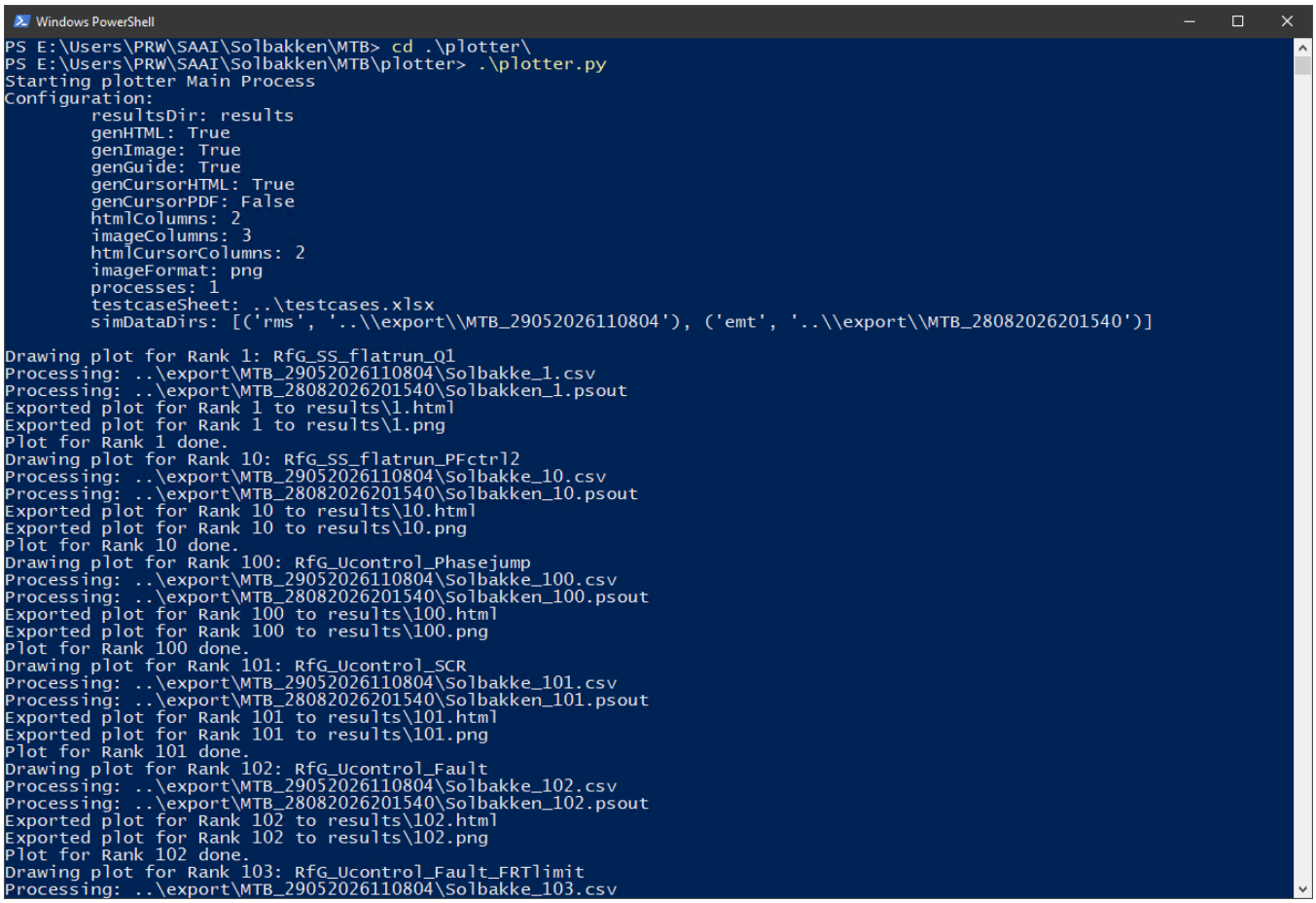
Next to the progress bar, output similar to the following may be displayed:

```
136/181 [07:43<03:03,  4.09s/it]
```

This means:

- `136/181`: 136 out of 181 ranks have completed.
- `07:43`: elapsed time.
- `03:03`: estimated remaining time.
- `4.09s/it`: time used for the latest completed iteration.

When `processes = 1`, plots are generated sequentially. This is the preferred mode for debugging because errors can be associated with the rank currently being processed.

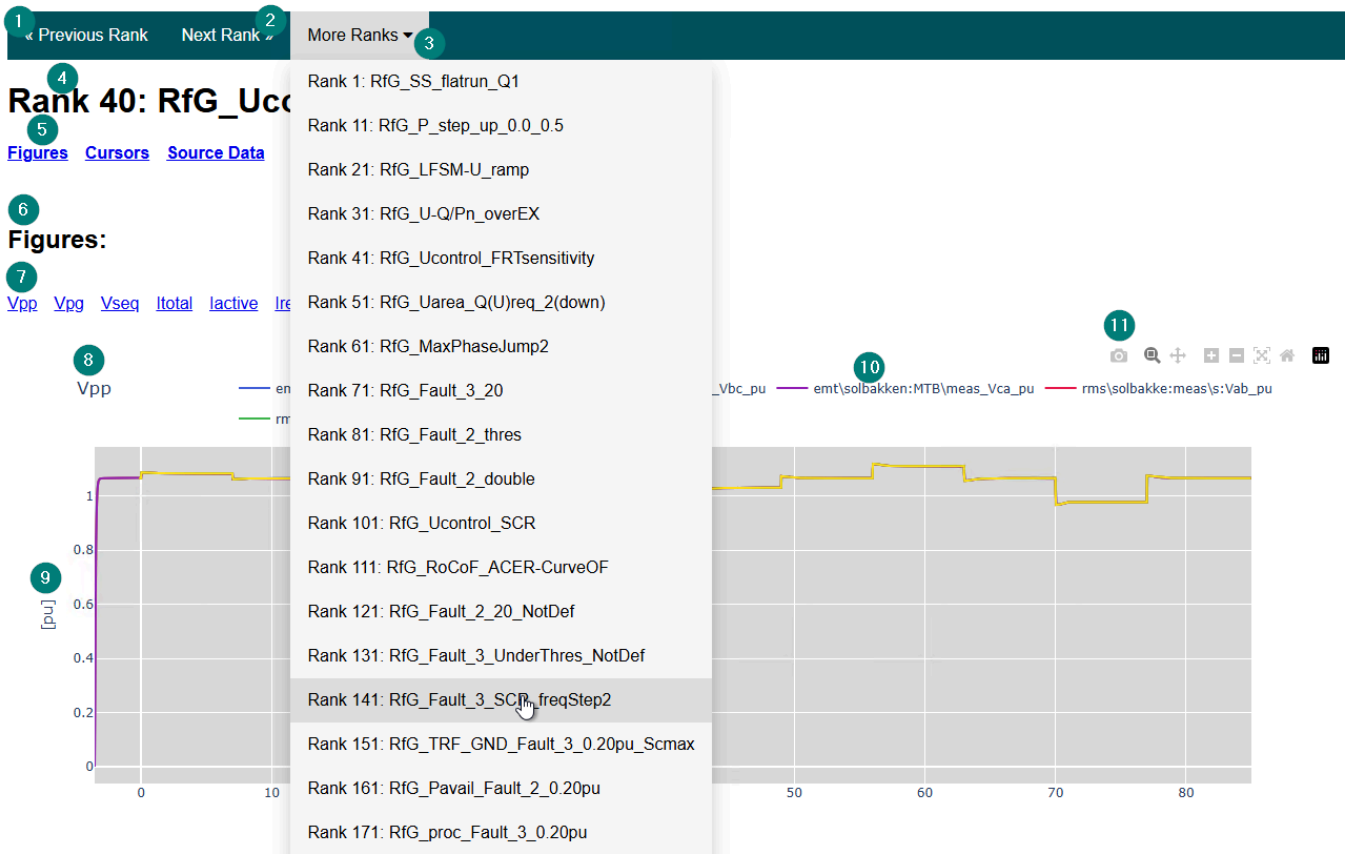


> **CAUTION**
>
> With `genGuide = True`, this might take a while, so only run with `processes = 1` on a small subset of *.csv and* .psout files

4.7 Plot Result Examples

Below, the plot results for Case/Rank 40 are shown with `htmlColumns = 2`,

A zoomed-in view is shown below with red numbers highlighting key parts of the HTML output.

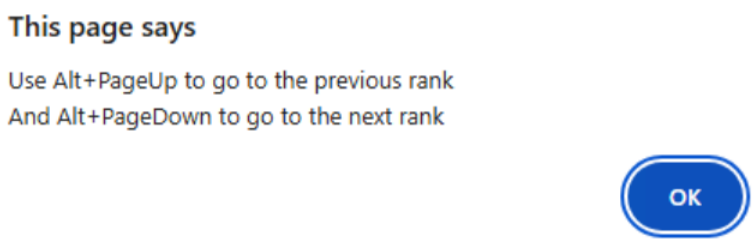


1. **Previous Rank** - navigates to the previous rank in the study, with wraparound.
2. **Next Rank** - navigates to the next rank in the study, with wraparound.
3. **More Ranks** - drop-down list for faster navigation between ranks.
4. **Rank Number and Title** - shows the rank number and case title from testcases.xlsx.
5. **Subsection Hyperlinks** - links to Figures, Cursors, and Source Data.

6. **Figures subsection** - contains the result plot figures.
7. **Plot Figure Hyperlinks** - links to each plot figure on the page.
8. **Plot Figure Title** - taken from figureSetup.csv.
9. **Plot Figure Units** - taken from figureSetup.csv.
10. **Plot Figure Legend** - lists the plotted signals. Click a legend item to hide or show the corresponding trace.
11. **Plotly Buttons** - standard Plotly controls for downloading, zooming, panning, autoscaling, and resetting axes.

The HTML reports also support keyboard shortcuts:

- `Alt + Page Up` : navigate to previous rank.
- `Alt + Page Down` : navigate to next rank.
- `Alt + H` : display help message.



---

5. Quickstart Guide Curve Generation

5.1 Introduction

From MTB 2.0, plotter.py can generate analytical guide curves for selected test cases. Guide curves are enabled in the plotter's config.ini file:

```
genGuide = True
```

> **CAUTION**
>
> Guide curve generation analytically calculates expected output waveforms from MTB-generated references and PSCAD simulation results stored in the `.psout` file. This requires additional post-processing and can add considerable plotting time, depending on CPU speed, result file size, and the number of parallel processes used.

The guide curves are intended as practical reference overlays. Low-pass filtering and delays are used to make the calculated responses look more like realistic controller responses, but the guide curves are still approximations and should not be treated as an exact model of a plant controller.

5.2 How Guide Curves Are Selected

Guide curves are generated by guide_functions.py. The main entry point is `genGuideResults`, which decides which guide signals to calculate based on the case name, Q control mode, test settings, and available PSCAD result signals.

Case or setting condition	Guide signal(s)	Main helper function(s)
Case name contains `P_step` or `PQ/Pn`, but not `Pavail`	`P_pu_PoC_Ramp`	`guidePramp`
Case name contains `Pref-change`, `Pavail_step`, or `Pavail_variation`	`P_pu_PoC_Ramp`	`guidePramp2`
Case name contains `FSM`, `RoCoF`, `Freq`, or `freqStep`	`P_pu_LFSM_FFR`, `P_pu_LFSM_Ramp`, and sometimes `P_pu_LFSM_Ramp_2s`	`guideLFSM`, `guideFSM`, `guideLFSMRamp`, `guideDelay`, `guideLPF`
Case name contains `SIPS`	`P_pu_PoC` and `P_pu_PoC_1s`	`guideSIPS`, `guideSIPS2`, `guideLPF`
Case is not SIPS, Fault, or LVFRT	`P_pu_PoC`	Direct copy of `MTB\mtb_s_pref_pu`
Qmode is `Q`, or Qmode is `Default` and default Q mode is `Q`	`Q_pu_Q_Ctrl`	`guideLPF`

Case or setting condition	Guide signal(s)	Main helper function(s)
Qmode is `Q(U)`, or Qmode is `Default` and default Q mode is `Q(U)`	`Q_pu_QU_Ctrl`	`guideQU`, `guideLPF`
Qmode is `PF`, or Qmode is `Default` and default Q mode is `PF`	`Q_pu_Qpf_Ctrl`	`guideQpf`, `guideLPF`
Case name contains `FRT`, `Fault`, or `support`	`Iq_pu_FFC`	`guideFFC`, `guideLPF`

5.3 General Guide Helpers

5.3.1 guideLPF

Applies a simple first-order low-pass filter to a guide signal. The helper is used to give analytically calculated guide curves a more realistic response shape.

The filter is configured from a cut-off frequency `fc` and sampling frequency `fs`.

5.3.2 guideDelay

Applies a fixed signal delay of `Td` seconds using the simulation sampling time `Ts`. Samples before the delayed signal becomes available are held at the initial value.

This helper is used for delayed frequency response and delayed SIPS guide curves.

5.4 Dedicated Guide Functions

5.4.1 guidePramp

Calculates the guide curve for the maximum rate of change of active power output, `P_pu_PoC_Ramp`, after a step in active power reference. It is used for cases where the case name contains `P_step` or `PQ/Pn`, but not `Pavail`.

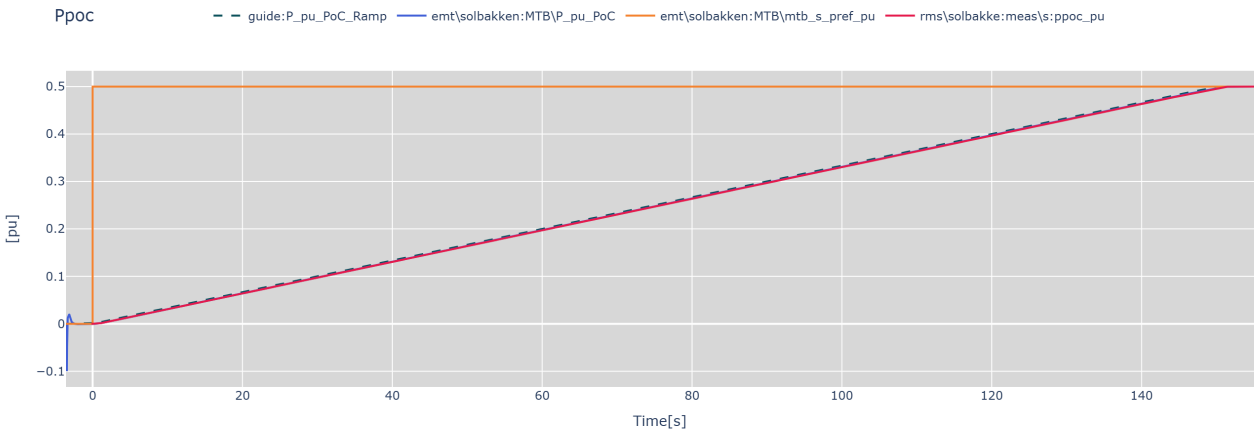
The ramp rate is limited to:

$$ m = \min(0.2, 60/P_n) $$

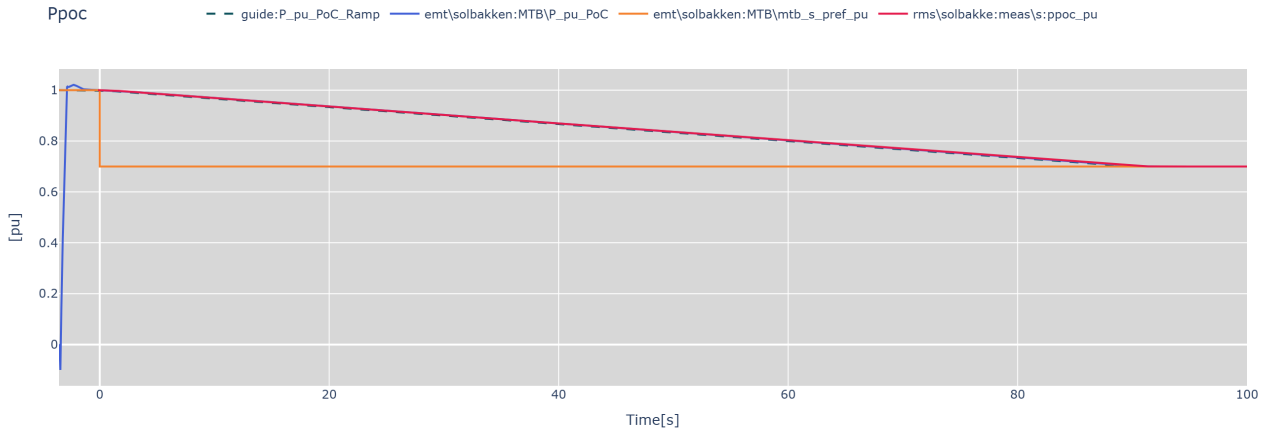
where m is in pu/min before conversion to pu/s, and P_n is the nominal plant power in MW.

This guide is based on a linear approximation of active power ramping for power-generating modules according to RfG (EU) 2016/631, Article 15.6 (e), and the Energinet NC RfG requirements.

Example: Rank 11: RfG*P*step*up*0.0_0.5, Figure Ppoc for upward active power regulation.



Example: Rank 14: RfG*P*step*down*1.0_0.7, Figure Ppoc for downward active power regulation.

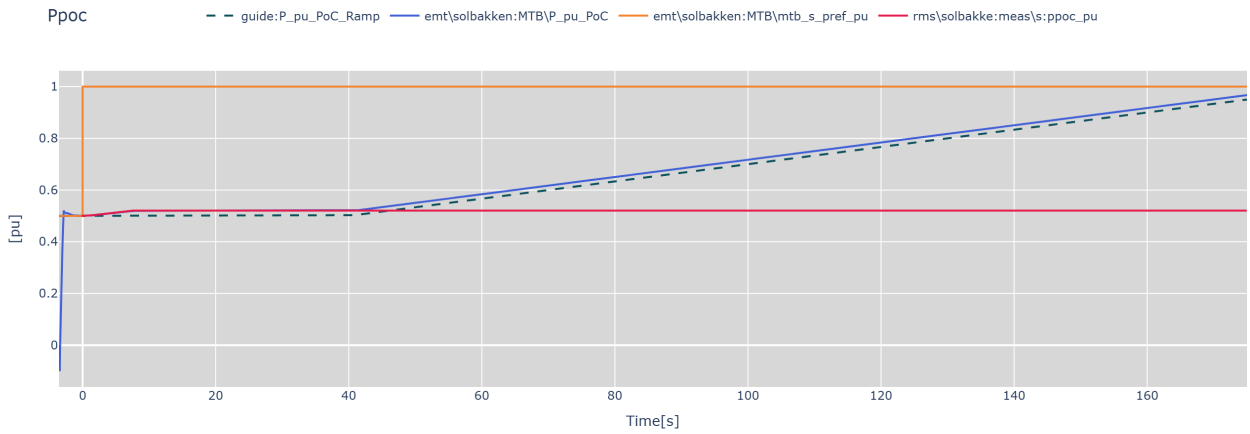


5.4.2 guidePramp2

Calculates `P_pu_PoC_Ramp` using a sample-by-sample difference equation instead of the simple linear approximation used by `guidePramp`. It is used for cases where the case name contains `Pref-change`, `Pavail_step`, or `Pavail_variation`.

The active power reference is clipped to the available active power signal, `MTB\mtb_s_pavail_pu`, before ramping. This makes the function suitable for cases where available power changes dynamically, for example due to reduced irradiation or wind speed.

Example: Rank 109: RfG*Pavail*step*P*step*up*05_10, Figure Ppoc



5.4.3 guideLFSMRamp

Calculates the ramp-limited LFSM active power guide signal, `P_pu_LFSM_Ramp`. It combines the LFSM/FSM active power response with the normal active power ramp-rate limit when frequency returns close to the nominal frequency.

The function uses a hysteresis band around 50 Hz:

Threshold	Meaning
`fLower = 0.020 Hz`	Below this deviation from 50 Hz, ramping is active
`fUpper = 0.040 Hz`	Above this deviation from 50 Hz, LFSM is active

When ramping is active, the output follows the active power reference at the same ramp-rate limit used by `guidePramp2`. When LFSM is active, the output is calculated by `guideLFSM`.

5.4.4 guideLFSM

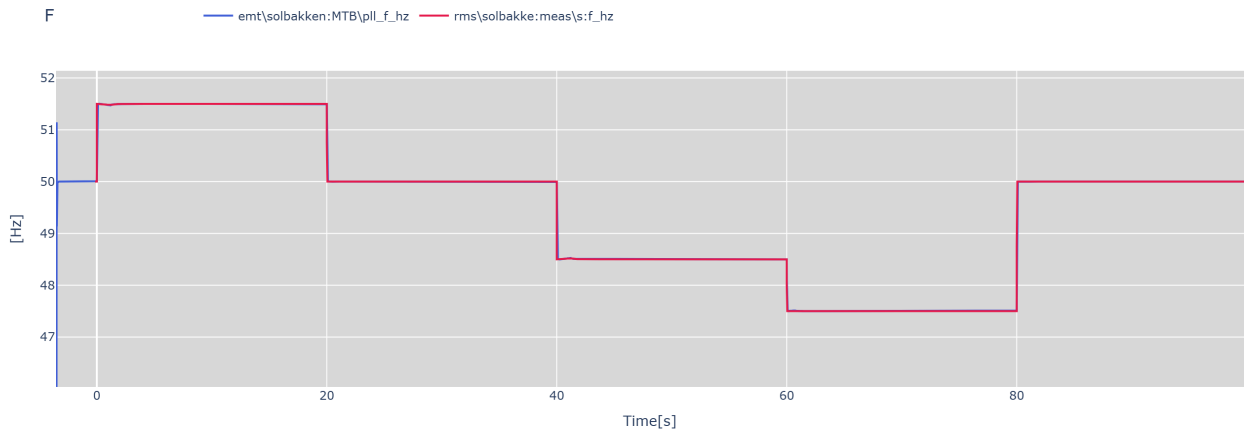
Calculates the LFSM-O or LFSM-U guide value for active power based on the measured frequency `pll_f_hz`.

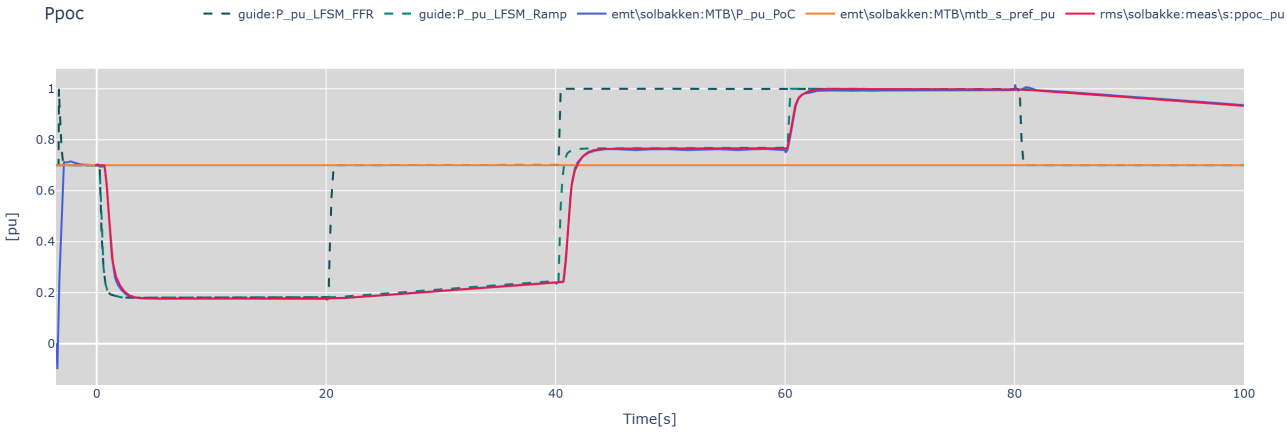
The DK1 and DK2 LFSM thresholds, see Energinet NC RfG are selected from the test settings:

Area	LFSM-O threshold	LFSM-U threshold	Droop
DK1	50.2 Hz	49.8 Hz	5%
DK2	50.5 Hz	49.5 Hz	4%

If FSM is enabled through `Pmode == LFSM+FSM`, `guideLFSM` first calls `guideFSM` and then applies the LFSM limit.

Example: Rank 19: RfG*LFSM-OU*step1, Figure Ppoc





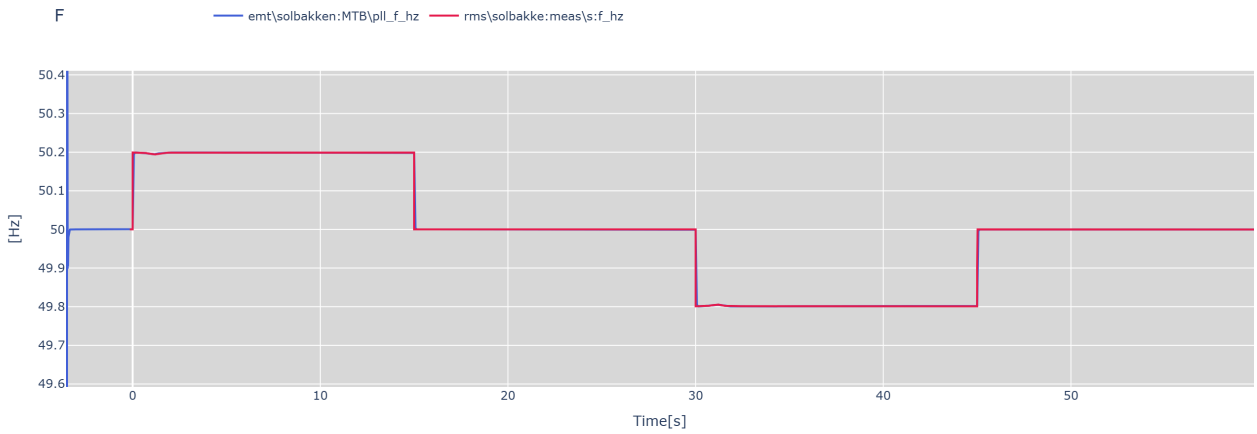
> **NOTE**
>
> LFSM cases can produce both `P_pu_LFSM_FFR` for fast-acting LFSM controllers and `P_pu_LFSM_Ramp` for slower ramp-limited response when the system frequency returns to 50 Hz.

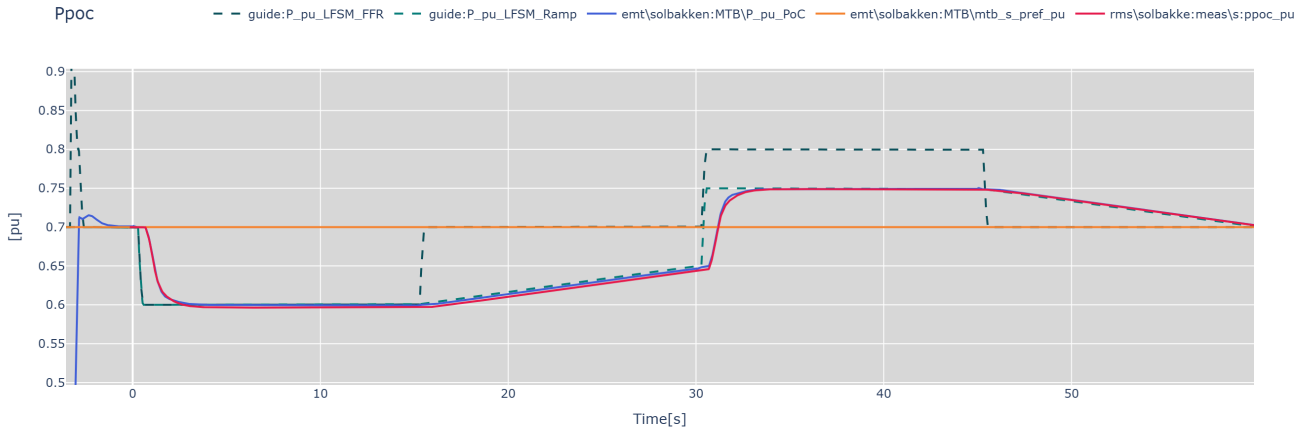
5.4.5 guideFSM

Calculates the FSM droop response used by `guideLFSM` when `Pmode == LFSM+FSM`. The FSM droop and deadband are read from the test settings.

The function applies the configured FSM droop around 50 Hz and clamps the result to +/-10% of the reference active power.

Example: Rank 17: ION*RfG*FSM_step1-Plot-Ppoc





5.4.6 guideQU

Calculates the guide reactive power required for voltage control mode, `Q_pu_QU_Inst`, before low-pass filtering to `Q_pu_QU_Ctrl`. It is used when `Qmode == Q(U)`, or when `Qmode == Default` and the default Q mode is `Q(U)`.

The voltage error is calculated as:

$$ \Delta U = U_{\text{ref}} - U_{\text{pos}} $$

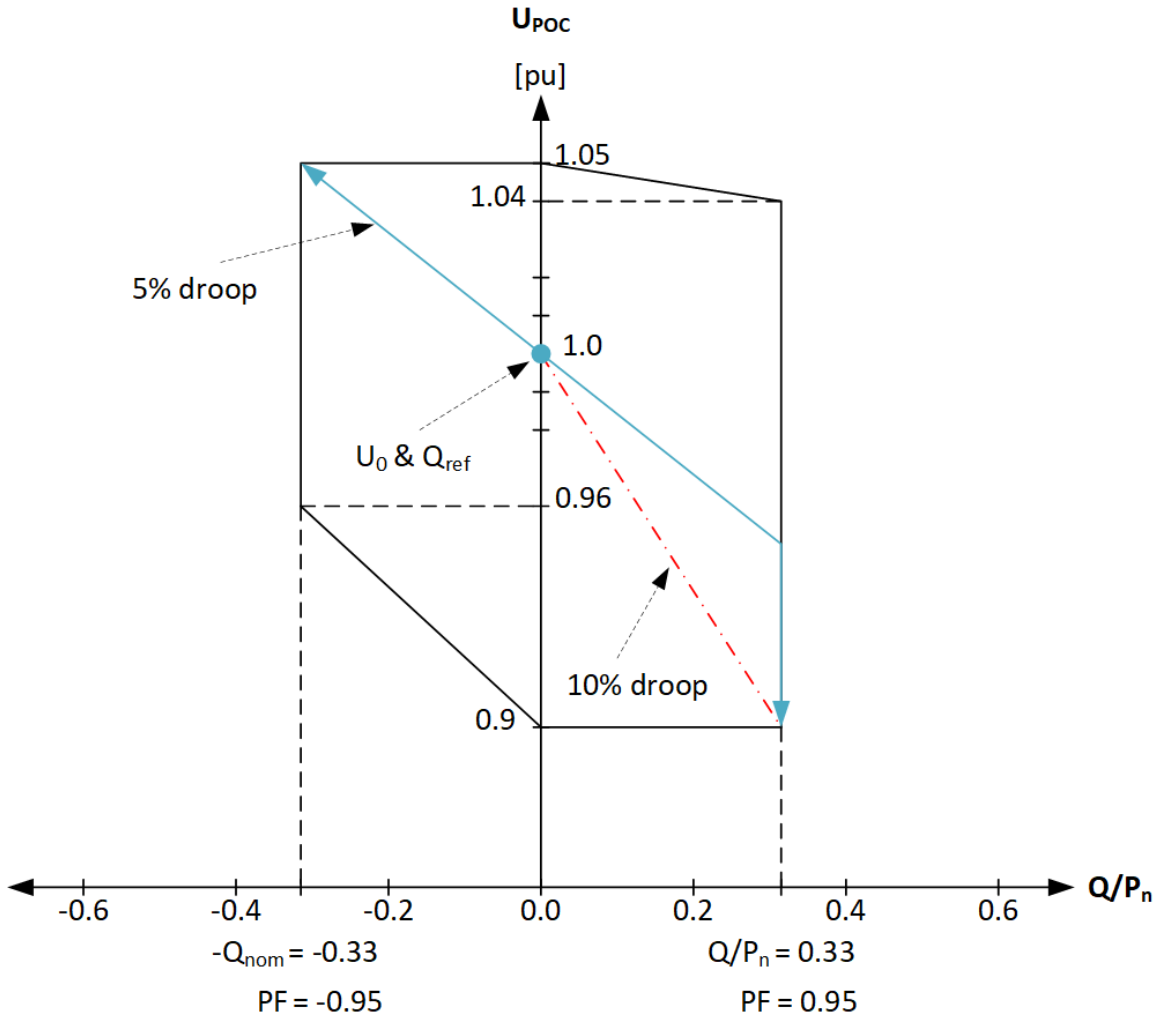
The required reactive power change is calculated from the Q(U) droop value s:

$$ \Delta Q = \frac{100 \cdot \Delta U}{U_{\text{ref}} \cdot \frac{Q_{\text{nom}}}{s}} $$

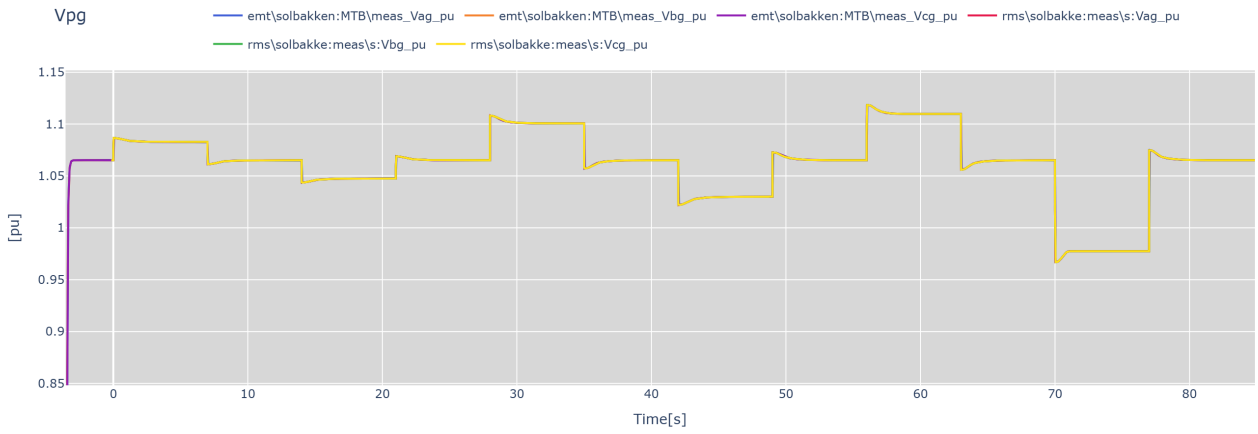
The output is limited to the nominal reactive power range:

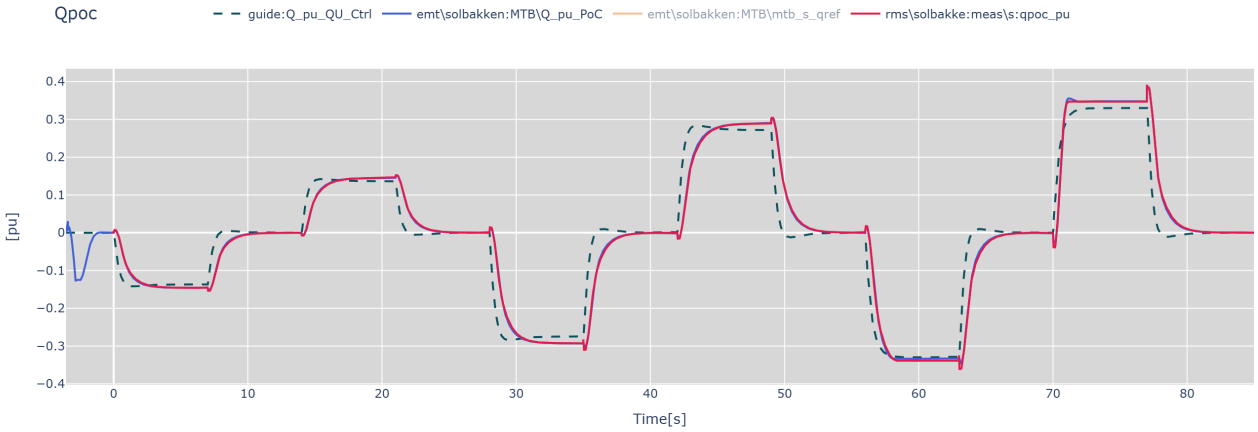
$$ Q_{\text{new}} = \begin{cases} +Q_{\text{nom}} & \text{if } Q_{\text{ref}} + \Delta Q > Q_{\text{nom}} \\ -Q_{\text{nom}} & \text{if } Q_{\text{ref}} + \Delta Q < -Q_{\text{nom}} \\ Q_{\text{ref}} + \Delta Q & \text{otherwise} \end{cases} $$

> **NOTE**
>
> In the current guide implementation, the positive-sequence voltage is measured at the PoC, the initial reactive power reference is $Q_{\text{ref}} = 0.00$ pu, and $Q_{\text{nom}} = 0.33$ pu.

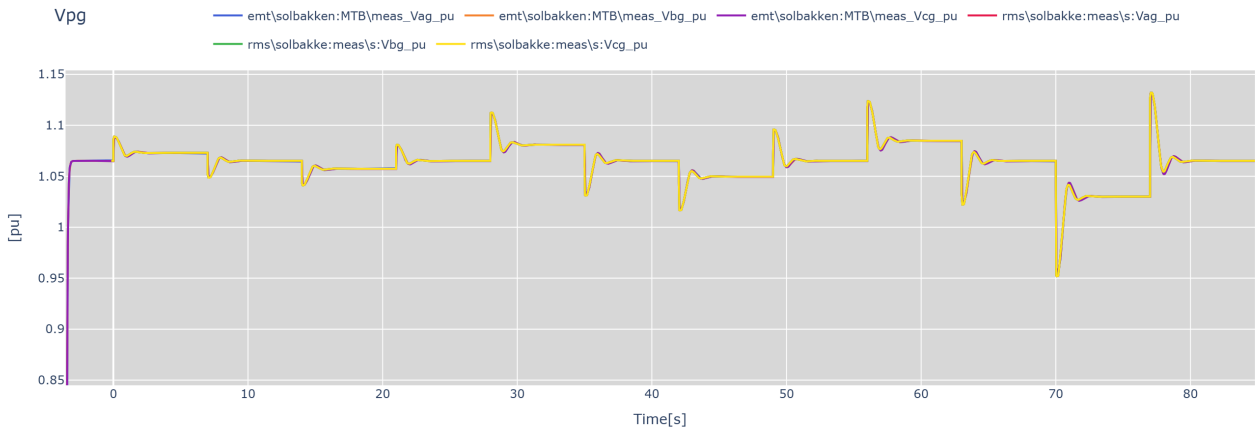


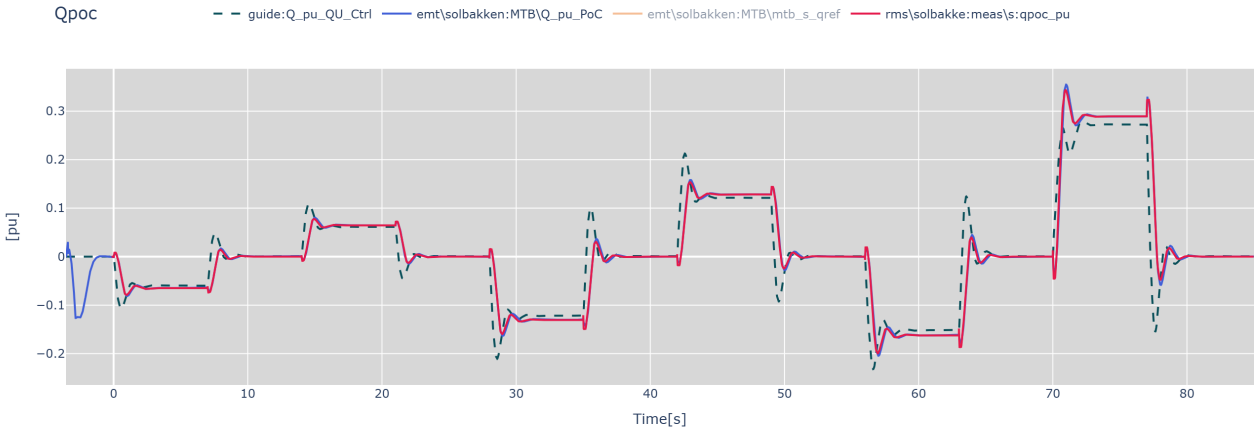
Example: Rank 40: RfG*Ucontrol*Scmax





Example: Rank 38: RfG*Ucontrol*Scmin





> **NOTE**
>
> The guide output above has large overshoots and should not be treated as an ideal guide response. It uses a hard-coded low-pass filter and is not tuned for a specific minimum SCR value. Response overshoot should be limited to less than 20%. In the plots above it is interesting to note that although the equivalent Thévening voltage of the grid is stepped, it is difficult to observe at the PoC due to the fact that the injection of reactive power has an immediate supporting effect on the PoC voltage. The PoC voltage is thus observed as have pulses, exacerbating the reactive power overshoot.

5.4.7 guideQpf

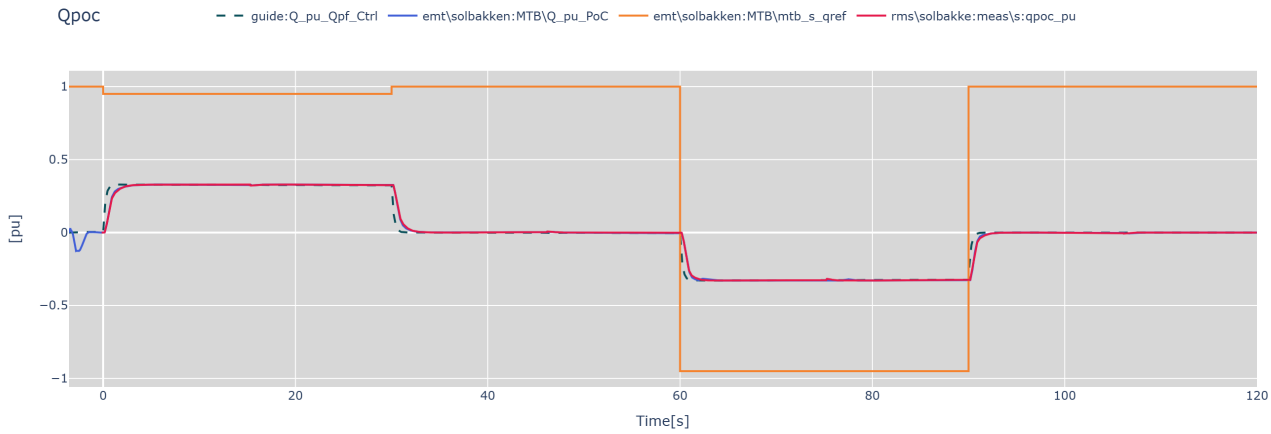
Calculates the guide reactive power output for power factor control mode, `Q_pu_Qpf_Inst`, before low-pass filtering to `Q_pu_Qpf_Ctrl`. It is used when `Qmode == PF`, or when `Qmode == Default` and the default Q mode is `PF`.

The function calculates reactive power from active power and the power factor reference:

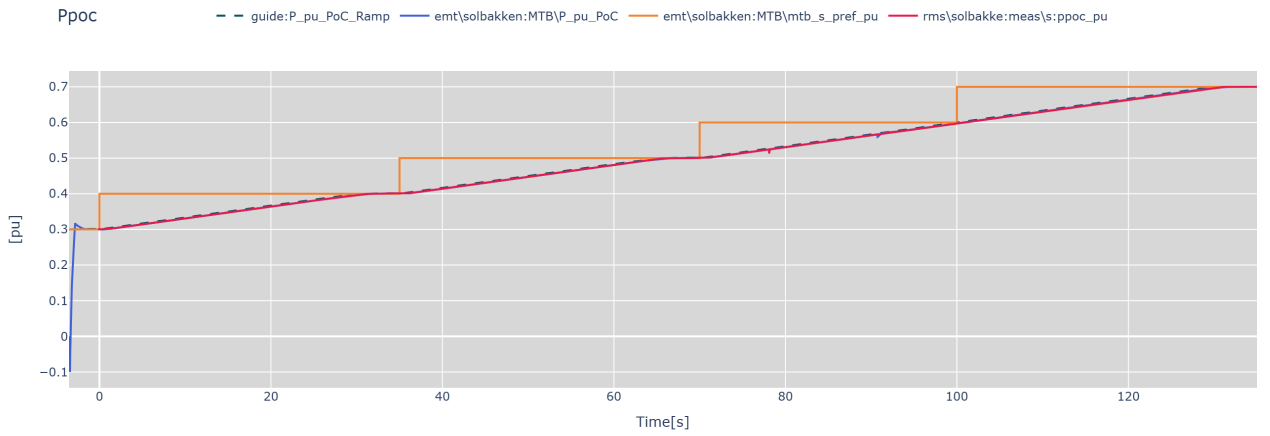
$$ Q = P \cdot \tan(\arccos(PF_{\text{ref}})) $$

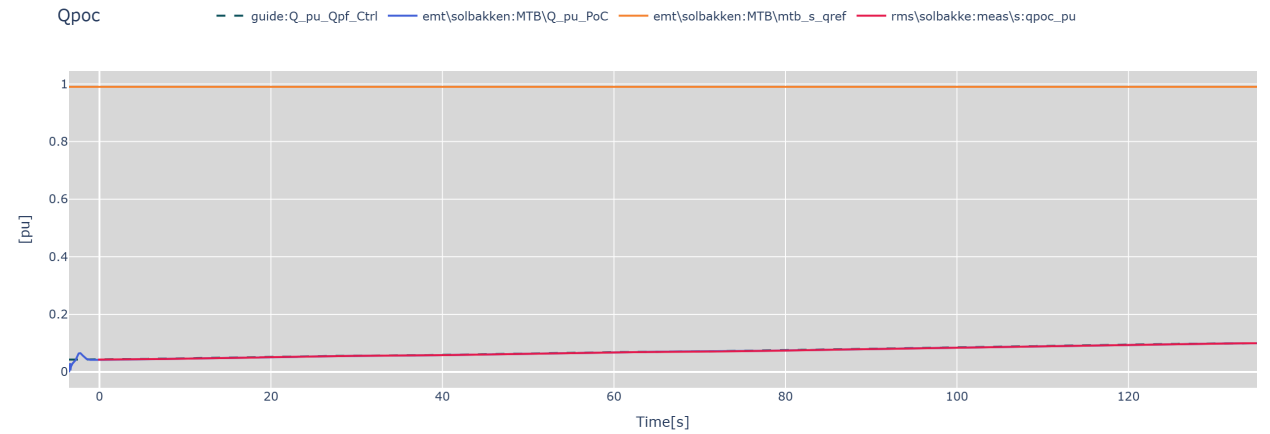
The result is clipped to +/-0.33 pu.

Example: Rank 55: RfG*Qpf*PFref-change



Example: Rank 56: RfG*Qpf*Pref-change





5.4.8 guideFFC

Calculates the fast fault current (FFC) guide contribution, `Iq_pu_FFC`, based on the positive-sequence voltage magnitude `MTB\fft_pos_Vmag_pu` and the pre-fault reactive current `Iq0`.

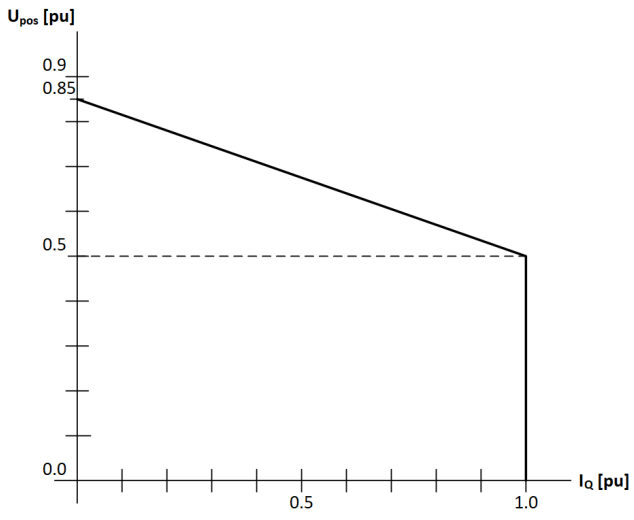
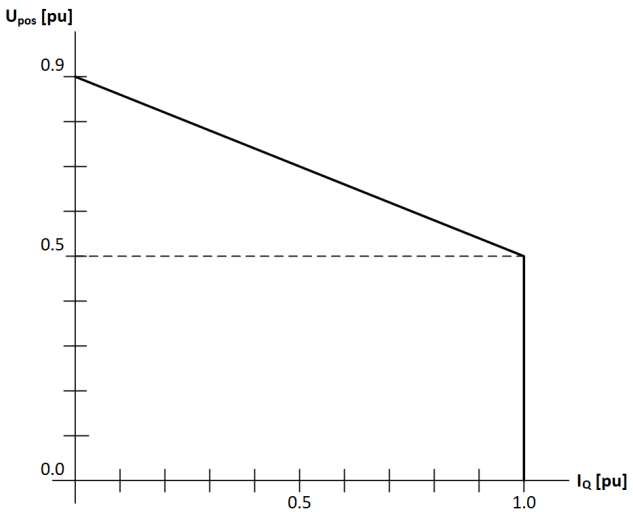
The FFC voltage threshold depends on the area (DK1 or DK2) and whether the plant is connected to a DSO network, see Energinet NC RfG:

Condition	FRT threshold	Slope m	Offset c
DK2 or DSO	0.90 pu	1 / 0.4	2.25
DK1 TSO	0.85 pu	1 / 0.35	2.42857

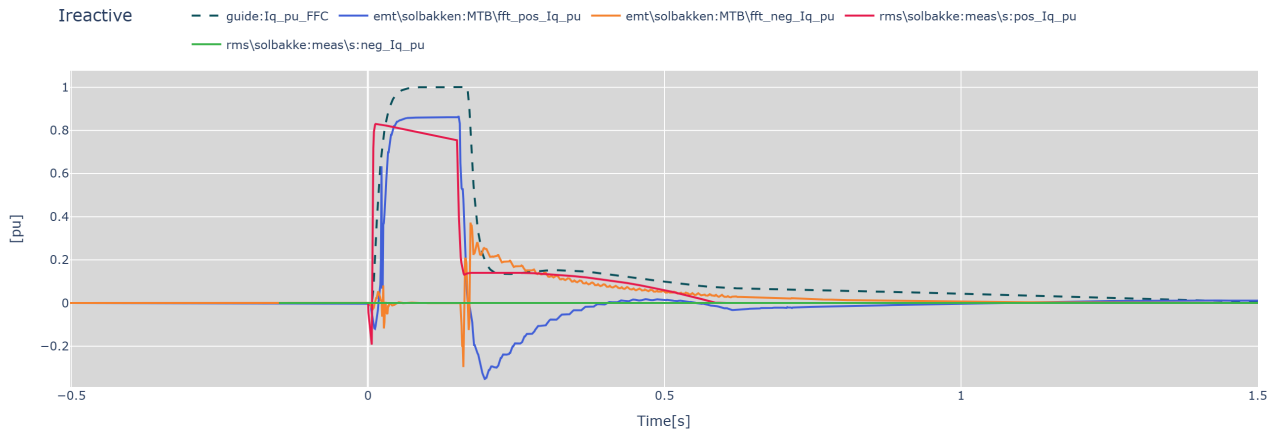
If the positive-sequence voltage is below the FRT threshold and above 0.5 pu, the additional reactive current is calculated as:

$$ I_{q,FFC} = -m \cdot U_{pos} + c + I_{q0} $$

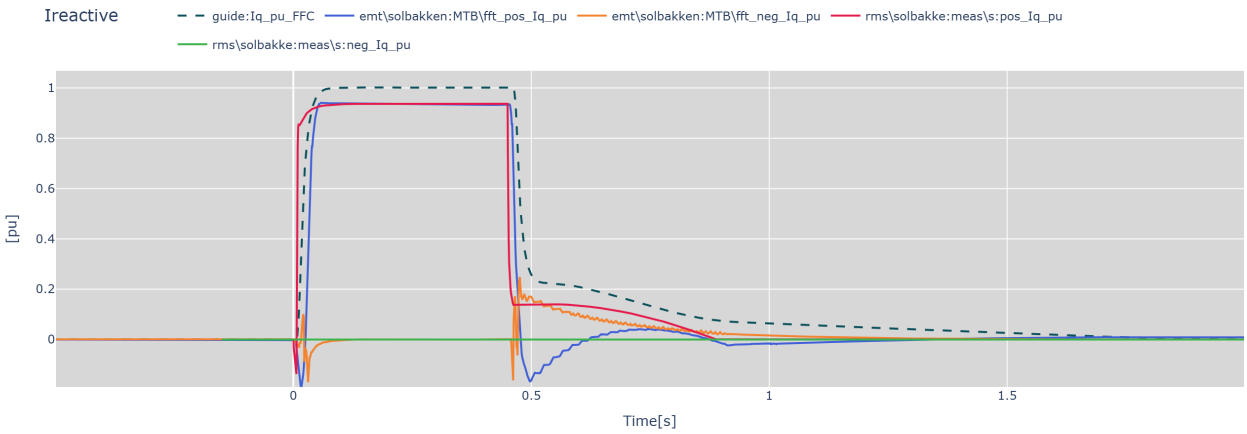
If `Upos <= 0.5`, the FFC contribution is limited to `1.0 + Iq0`.

DK1	DK2
	

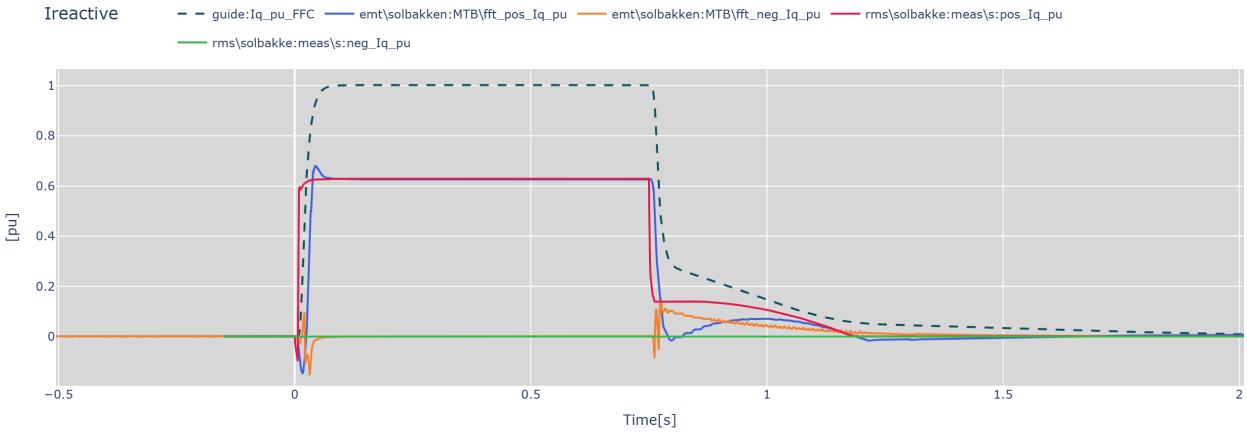
Example: Rank 70: RfG*Fault*3_0



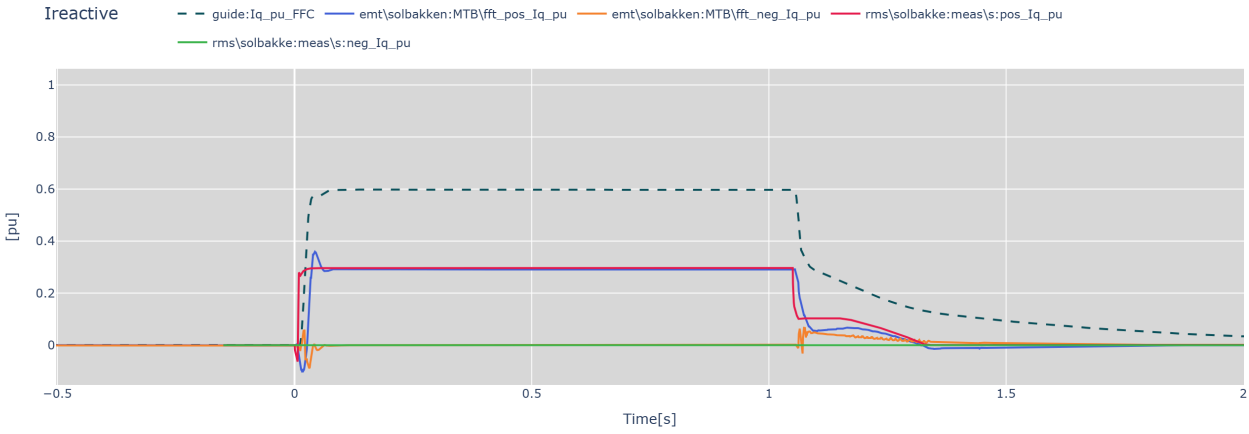
Example: Rank 71: RfG*Fault*3_20



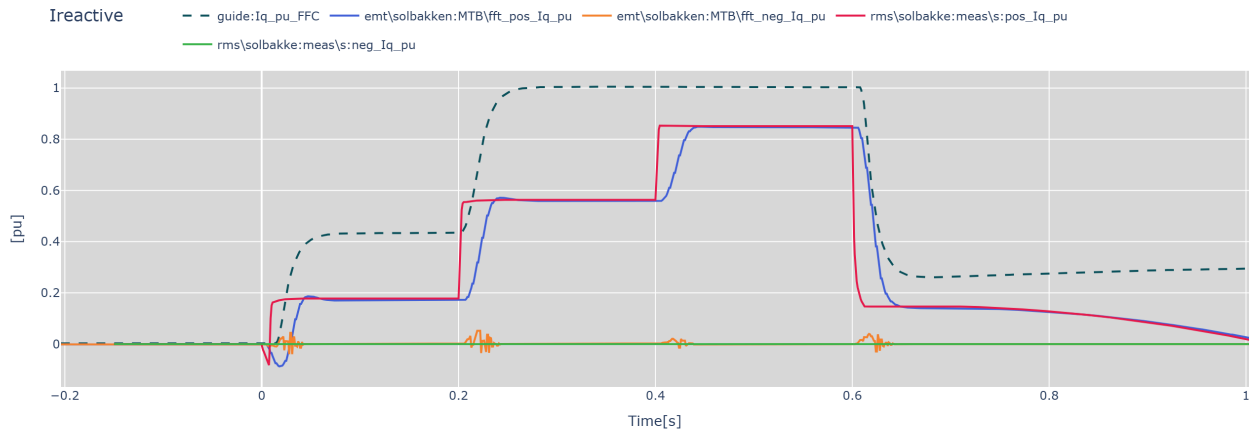
Example: Rank 72: RfG*Fault*3_40



Example: Rank 73: RfG*Fault*3_60



Example: Rank 88: RfG_Voltage-support



5.4.9 guideSIPS

Decodes the MTB SIPS generation bitfield signal, `MTB\mtb_s_sips_g`, and returns the commanded active power ceiling. The result is used as `P_pu_PoC_SIPS_ref` before a low-pass filter creates the plotted `P_pu_PoC` guide curve.

The default SIPS step mapping as per Energinet NC RfG are:

SIPS bit	Active power ceiling
0	Reserved
1	0.70 pu
2	0.50 pu
3	0.40 pu
4	0.25 pu
5	0.00 pu

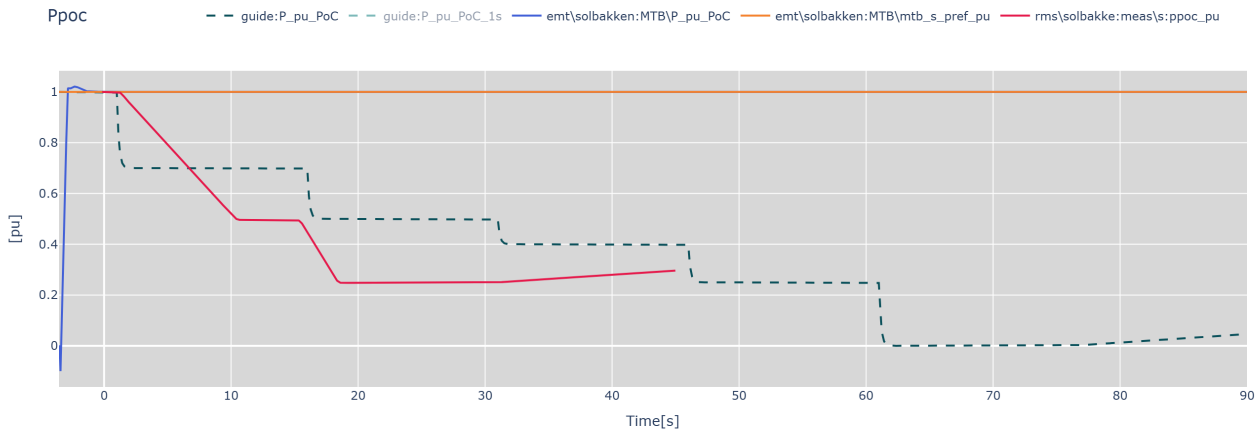
If multiple SIPS steps are active at the same time, the function applies the strongest reduction, i.e. the lowest active power ceiling. The active power ceiling is also limited by the current active power reference, `MTB\mtb_s_pref_pu`.

For the fast SIPS guide curve:

1. `guideSIPS` decodes the SIPS bitfield and applies the active power ceiling.
2. SIPS reductions are applied immediately.
3. SIPS release is ramped up using the same active power ramp-rate limit as the active power ramping guide: `min(0.2 pu/min, 60 MW/min)`.
4. `genGuideResults` filters the resulting reference with an approximately 1 Hz low-pass filter and plots it as `P_pu_PoC`.

This guide curve represents a fast response with no intentional delay.

Example: Rank 58: RfG*SIPS*QU



> **NOTE**
>
> For the example above, the EMT result limits are wrong and fails at 45 s, whilst there are no RMS output.

5.4.10 guideSIPS2

Calculates the delayed SIPS guide curve, plotted as `P_pu_PoC_1s`. It uses the same SIPS bitfield decoding and step mapping as `guideSIPS`, but changes the dynamic response for reductions.

By default, `guideSIPS2` applies:

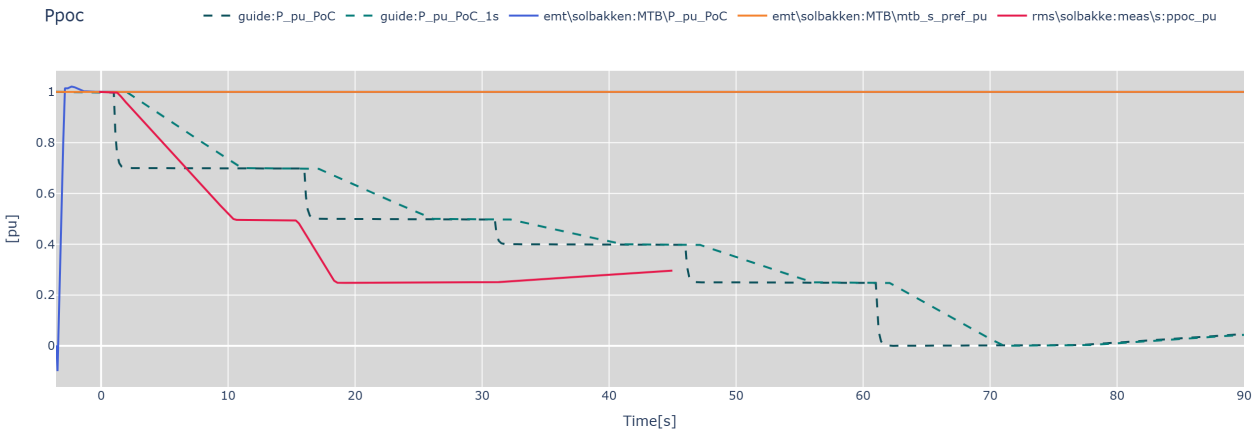
Parameter	Default	Meaning
`Td`	1.0 s	Delay before responding to SIPS reductions
`t_ramp`	10.0 s	Total target response time used for the delayed down-ramp calculation

The delayed SIPS response is calculated as follows:

1. Decode the SIPS bitfield and determine the active power ceiling.
2. Limit the ceiling by `MTB\mtb_s_pref_pu`.
3. Delay the resulting target by `Td` seconds using `guideDelay`.
4. When a new SIPS reduction appears, calculate a linear down-ramp rate so the response reaches the new target over `t_ramp - Td` seconds.
5. Continue the down-ramp until the delayed target is reached.
6. When SIPS is released, ramp active power back up using `min(0.2 pu/min, 60 MW/min)`.

`guideSIPS2` is useful for comparing model behaviour against a delayed response requirement where the response should not be instantaneous, but should ramp linearly after the delay has elapsed.

Example: Rank 58: RfG*SIPS*QU (*shown both the fast and delayed and linear ramped response*)



> **NOTE**
>
> For the example above, the EMT SIPS limits are wrong and the simulations fails at 45 s. Furthermore, there are no RMS output for this case.

6. Quickstart Cursor Metrics

6.1 Introduction

From MTB 2.0, cursor metrics can be used to calculate selected values from RMS and EMT result signals over defined cursor time intervals. The available cursor metrics are:

- START returns the signal value at the start cursor time.
- END returns the signal value at the end cursor time.
- DELTA returns the difference between the signal value at the end cursor time and the signal value at the start cursor time.
- MIN returns the minimum value of the signal between the start and end cursor times.
- MAX returns the maximum value of the signal between the start and end cursor times.
- MEAN returns the mean value of the signal between the start and end cursor times.
- GRAD_MIN returns the minimum value of the signal's numpy.gradient between the start and end cursor times.
- GRAD_MAX returns the maximum value of the signal's numpy.gradient between the start and end cursor times.
- GRAD_MEAN returns the mean value of the signal's numpy.gradient between the start and end cursor times.
- RESPONSE returns the response delay time from the start cursor time until the signal has reached 10% of the DELTA value.
- RISE_FALL returns the rise or fall time by measuring the time from 10% to 90% of the DELTA value.
- SETTLING returns the settling time from the start cursor time until the signal remains within the tolerance band, default 2%, around the END value.
- OVERSHOOT returns the peak overshoot percentage and an estimated damping ratio.
- FSM_DROOP returns the FSM droop calculated from active power and frequency.
- LFSM_DROOP returns the LFSM droop calculated from active power and frequency.
- QU_T1 returns the Q(U) t1 response time as RESPONSE plus RISE_FALL.
- QU_T2 returns the Q(U) t2 settling time using a 2% tolerance band.
- QU_DROOP returns the Q(U) droop calculated from reactive power and voltage.
- QUSSTOL returns the Q(U) steady-state tolerance compared with the required reactive power change.
- DELTA_FFC returns the required fast fault current contribution and the difference to the measured current response.

6.2 Cursor Setup

The cursors to be used for each Case/Rank are configured using the cursorSetup.csv file.

Column Name	Description
title	A descriptive name that is used for the cursor metric table
rank	The case rank number for which the cursor metric table will be generated
*cursor*options_	A comma-separated list of cursor functions to be applied
*emt*signals_	A comma-separated list of the EMT signals to be used by the cursor functions
*rms*signals_	A comma-separated list of the RMS signals to be used by the cursor functions

Column Name	Description
*time*ranges_	A comma-separated list of the cursor start and end times, entered in pairs

Most cursor functions only require one signal. The exceptions are FSM_DROOP and LFSM_DROOP, which require active power and frequency signals, for example **MTB\P*pu*PoC** and **MTB\pll*f*hz** for EMT, and **meas\s:ppoc*pu*** *and* ***meas\s:f*hz** for RMS. QU_DROOP and QUSSTOL require reactive power and voltage signals, for example **MTB\Q*pu*PoC** and **MTB\meas*Vag*pu** for EMT, and **meas\s:qpoc*pu*** *and* ***meas\s:Vag*pu** for RMS. DELTA_FFC requires reactive current and positive-sequence voltage signals.

6.3 Cursor Time Ranges

The cursor time ranges define the start and end time of each cursor pair, for example:

- **0.0** - the cursor functions are applied to the time range from 0.0 s to the end of the simulated signal.
- **0.0, 10.0** - the cursor functions are applied to the time range from 0.0 s to 10.0 s.
- **0.0, 10.0, 10.0** - the cursor functions are applied to the time ranges from 0.0 s to 10.0 s and from 10.0 s to the end of the simulated signal.
- **0.0, 10.0, 10.0, 20.0** - the cursor functions are applied to the time ranges from 0.0 s to 10.0 s and from 10.0 s to 20.0 s.
- **0.0, 10.0, 10.0, 20.0, 20.0** - the cursor functions are applied to the time ranges from 0.0 s to 10.0 s, from 10.0 s to 20.0 s, and from 20.0 s to the end of the simulated signal.

The cursor time ranges can overlap, but each explicit pair must obey that end time > start time, for example:

- **0.0, 10.0, 5.0, 15.0** - the cursor functions are applied to the time ranges from 0.0 s to 10.0 s and from 5.0 s to 15.0 s.

6.4 Cursor Functions

The cursor functions are defined in cursor_functions.py. The numbering below follows the `CursorType` enum order in cursor_type.py.

6.4.1 START

Returns the signal value at the start of the cursor interval.

6.4.2 END

Returns the signal value at the end of the cursor interval.

6.4.3 DELTA

Returns the difference between END and START.

e.g. **Rank*11: ION*RfG*P*step*up*0 0_0 5**

Active power

Cursor time intervals	emt\Solbakke:MTB\P_pu_PoC	rms\Solbakke:meas\s:ppoc_pu
0.0 s : 150.0 s	Start: y(0.000) = -0.000	Start: y(0.000) = -0.000
0.0 s : 150.0 s	End: y(0.497) = 0.497	End: y(0.497) = 0.497
0.0 s : 150.0 s	Delta: Δy = 0.497	Delta: Δy = 0.497
0.0 s : 150.0 s	Grad (min): -0.103 pu/min	Grad (min): -0.000 pu/min
0.0 s : 150.0 s	Grad (max): 1.395 pu/min	Grad (max): 0.990 pu/min
0.0 s : 150.0 s	Grad (mean): 0.199 pu/min	Grad (mean): 0.199 pu/min

6.4.4 MIN

Returns the minimum signal value in the cursor interval and the time at which it occurs.

6.4.5 MAX

Returns the maximum signal value in the cursor interval and the time at which it occurs.

6.4.6 MEAN

Returns the mean signal value in the cursor interval.

e.g. **Rank 7: PRESSflatrun_Uctrl1**

e.g. **Rank 7: PRESSflatrun_Uctrl1**

6.4.7 GRAD_MIN

Returns the minimum signal gradient in the cursor interval. The result is reported in pu/min.

6.4.8 GRAD_MAX

Returns the maximum signal gradient in the cursor interval. The result is reported in pu/min.

6.4.9 GRAD_MEAN

Returns the mean signal gradient in the cursor interval. The result is reported in pu/min.

e.g. **Rank*11: ION*RfG*P*step*up*0 0_0 5**

Active power

Cursor time intervals	emt\Solbakke:MTB\P_pu_PoC	rms\Solbakke:meas\s:ppoc_pu
0.0 s : 150.0 s	Start: y(0.000) = -0.000	Start: y(0.000) = -0.000
0.0 s : 150.0 s	End: y(0.497) = 0.497	End: y(0.497) = 0.497
0.0 s : 150.0 s	Delta: Δy = 0.497	Delta: Δy = 0.497
0.0 s : 150.0 s	Grad (min): -0.103 pu/min	Grad (min): -0.000 pu/min
0.0 s : 150.0 s	Grad (max): 1.395 pu/min	Grad (max): 0.990 pu/min
0.0 s : 150.0 s	Grad (mean): 0.199 pu/min	Grad (mean): 0.199 pu/min

6.4.10 RESPONSE

Returns the response delay time from the start of the cursor interval until the signal has reached 10% of the DELTA value.

6.4.11 RISE_FALL

Returns the rise or fall time from 10% to 90% of the DELTA value.

e.g. **Rank 17: ION*RfG*FSM_step1**

6.4.12 SETTLING

Returns the settling time from the start of the cursor interval until the signal remains within the tolerance band around the END value. The default tolerance is 2% of DELTA.

6.4.13 OVERSHOOT

Returns the peak overshoot percentage and an estimated second-order damping ratio.

e.g. **Rank 38: ION*RfG*UcontrolScmin-Cursor-Reactivepower**

Reactive power

Cursor time intervals	emt\Solbakke:MTB\Q_pu_PoC	rms\Solbakke:meas\s:qpoc_pu
0.0 s : 7.0 s	Overshoot: 16.76 % (ζ ≈ 0.494)	Overshoot: 15.28 % (ζ ≈ 0.513)
7.0 s : 14.0 s	Overshoot: 18.08 % (ζ ≈ 0.478)	Overshoot: 16.12 % (ζ ≈ 0.502)
14.0 s : 21.0 s	Overshoot: 13.32 % (ζ ≈ 0.540)	Overshoot: 11.38 % (ζ ≈ 0.569)
21.0 s : 28.0 s	Overshoot: 12.08 % (ζ ≈ 0.558)	Overshoot: 12.24 % (ζ ≈ 0.556)
28.0 s : 35.0 s	Overshoot: 16.89 % (ζ ≈ 0.493)	Overshoot: 17.01 % (ζ ≈ 0.491)
35.0 s : 42.0 s	Overshoot: 21.45 % (ζ ≈ 0.440)	Overshoot: 19.18 % (ζ ≈ 0.465)
42.0 s : 49.0 s	Overshoot: 10.34 % (ζ ≈ 0.586)	Overshoot: 8.77 % (ζ ≈ 0.612)
49.0 s : 56.0 s	Overshoot: 9.36 % (ζ ≈ 0.602)	Overshoot: 9.30 % (ζ ≈ 0.603)
56.0 s : 63.0 s	Overshoot: 17.54 % (ζ ≈ 0.485)	Overshoot: 17.01 % (ζ ≈ 0.491)
63.0 s : 70.0 s	Overshoot: 21.45 % (ζ ≈ 0.440)	Overshoot: 19.18 % (ζ ≈ 0.465)
70.0 s : 77.0 s	Overshoot: 0.00 % (ζ ≈ 1.000)	Overshoot: 0.00 % (ζ ≈ 1.000)
77.0 s : ..	Overshoot: 13.62 % (ζ ≈ 0.536)	Overshoot: 7.36 % (ζ ≈ 0.639)
0.0 s : 7.0 s	Settling time: 2.927 s	Settling time: 2.813 s
7.0 s : 14.0 s	Settling time: 2.923 s	Settling time: 2.807 s
14.0 s : 21.0 s	Settling time: 2.922 s	Settling time: 1.849 s
21.0 s : 28.0 s	Settling time: 2.921 s	Settling time: 2.907 s
28.0 s : 35.0 s	Settling time: 3.022 s	Settling time: 2.930 s
35.0 s : 42.0 s	Settling time: 2.930 s	Settling time: 2.809 s
42.0 s : 49.0 s	Settling time: 2.915 s	Settling time: 2.033 s
49.0 s : 56.0 s	Settling time: 2.922 s	Settling time: 2.909 s
56.0 s : 63.0 s	Settling time: 3.025 s	Settling time: 2.931 s
63.0 s : 70.0 s	Settling time: 2.930 s	Settling time: 2.809 s
70.0 s : 77.0 s	Settling time: 3.725 s	Settling time: 3.611 s
77.0 s : ..	Settling time: 3.025 s	Settling time: 1.940 s

6.4.14 FSM_DROOP

Returns the FSM droop value calculated from active power and frequency. The FSM deadband from the test case settings is included in the calculation.

6.4.15 LFSM_DROOP

Returns the LFSM droop value calculated from active power and frequency. The DK1/DK2 LFSM threshold is selected from the test case settings.

6.4.16 QU_T1

Returns the Q(U) t1 response time, calculated as RESPONSE + RISE_FALL. This allows the metric to be used directly with RfG Article 21.3 (d)(iv) from NC RfG - Nationale krav.

21	3	d	iv	following a step change in voltage, the power park module shall be capable of achieving 90 % of the change in reactive power output within a time t_1 to be specified by the relevant system operator in the range of 1 to 5 seconds, and must settle at the value specified by the slope within a time t_2 to be specified by the relevant system operator in the range of 5 to 60 seconds, with a steady-state reactive tolerance no greater than 5 % of the maximum reactive power. The relevant system operator shall specify the time specifications;	t1: 1-5 sec: t1 = 1 sek. t2: 5-60 sec: t2 = 5 sek.

6.4.17 QU_T2

Returns the Q(U) t2 settling time, calculated using SETTLING with a 2% tolerance band. This allows the metric to be used directly with RfG Article 21.3 (d)(iv) from NC RfG - Nationale krav.

21	3	d	iv	following a step change in voltage, the power park module shall be capable of achieving 90 % of the change in reactive power output within a time t_1 to be specified by the relevant system operator in the range of 1 to 5 seconds, and must settle at the value specified by the slope within a time t_2 to be specified by the relevant system operator in the range of 5 to 60 seconds, with a steady-state reactive tolerance no greater than 5 % of the maximum reactive power. The relevant system operator shall specify the time specifications;	t1: 1-5 sec: t1 = 1 sek. t2: 5-60 sec: t2 = 5 sek.

6.4.18 QU_DROOP

Returns the Q(U) droop calculated from the change in reactive power and the change in voltage over the cursor interval.

e.g. **Rank 38: ION*RfG*Ucontrol*Scmin-Cursor-Reactive*power**

6.4.19 QU*SS*TOL

Returns the Q(U) steady-state tolerance as the deviation between the measured reactive power change and the required reactive power change, expressed as a percentage of Qnom.

6.4.20 DELTA_FFC

Returns the required fast fault current contribution based on the positive-sequence voltage and compares it with the measured change in reactive current over the cursor interval.

7. Quickstart Python Utility Scripts

The `utility_scripts` folder contains helper scripts for model checking, project cleanup, data extraction, PSCAD/PowerFactory comparison, and recovery of PSCAD result files. These scripts are not part of the normal MTB execution workflow, but they are useful when preparing models, checking consistency between PowerFactory and PSCAD, or troubleshooting simulation output.

Most scripts are intended for one of three execution contexts:

Script type	Run from	Typical purpose
PowerFactory scripts	Inside DIgSILENT PowerFactory as a Python/ComPython script	Extract data or run model checks
PSCAD scripts	From a Python environment with `mhi.pscad`, or inside the PSCAD scripting environment	Inspect or clean PSCAD projects
Standalone command-line scripts	From PowerShell or a terminal	Inspect `.psout` files or recover output files

7.1 Script overview

Script	Wrapper / related file	Main use	Execution context
`check_powerfactory_model.py`	`Check PowerFactory Model.pfd`	Checks that a PowerFactory model can run balanced/unbalanced load flow, calculate initial conditions, and run RMS simulations	PowerFactory
`get_component_data_from_powerfactory.py`	`Get Component Data.pfd`	Exports cable and transformer data from PowerFactory to Excel	PowerFactory
`get_dsl_checksums_from_powerfactory.py`	`Get DSL Checksums.pfd`	Exports checksums for encrypted DSL model types	PowerFactory
`get_relay_data_from_powerfactory.py`	`Get Relay Data.pfd`	Exports relay and protection settings from PowerFactory to Excel	PowerFactory

Script	Wrapper / related file	Main use	Execution context
`compare_component_data_with_pscad.py`	Uses the workbook from `Get Component Data.pfd`	Compares PowerFactory cable/transformer data with the corresponding PSCAD component parameters	PSCAD / Python with `mhi.pscad`
`clean_project_definitions.py`		Removes unused component definitions from a PSCAD project	PSCAD / Python with `mhi.pscad`
`list_psout_signals.py`		Lists available signals in a PSCAD `.psout` file	Command line
`recover_psout_files.py`		Recovers and renames `.psout` files after a PSCAD out-of-memory crash	Command line

7.2 PowerFactory model check

`check_powerfactory_model.py` is used to verify that the active PowerFactory project can run the basic study steps required by the MTB workflow. The script is delivered with the PowerFactory wrapper `Check PowerFactory Model.pfd`.

The script performs the following checks:

1. Compiles all relevant dynamic model types. This is not yet an Energinet requirement.
2. Checks that state variable derivatives are less than the tolerance for the initial conditions.
3. Calculates a balanced load flow and flat run.
4. Calculates an unbalanced load flow and flat run.
5. Checks fixed time step runs.
6. Checks variable time step runs.

The script expects to be run inside PowerFactory and uses attributes from the current PowerFactory script object:

Attribute	Purpose
`MaximumError`	Tolerance used when checking state variable derivatives
`ModelTypes`	Dynamic model types to compile
`ForceRebuild`	Whether model types should be force rebuilt
`DisplayCompilerMessages`	Compiler message output level

7.3 Export component data from PowerFactory

`get_component_data_from_powerfactory.py` exports selected network component data from PowerFactory to an Excel workbook. The script is delivered with the PowerFactory wrapper `Get Component Data.pfd`.

This is the first step in the PowerFactory/PSCAD component comparison workflow. It reads cable and transformer data from the RMS model, calculates equivalent parameters in PSCAD format, and writes the results to an Excel workbook.

The exported workbook contains:

Sheet	Content
`PowerFactory Cable Data`	Cable names, number of parallel lines, length, positive- and zero-sequence impedance/admittance data, and PSCAD-equivalent values
`PowerFactory ElmTr2 Data`	Two-winding transformer names, type, vector group, ratings, voltages, short-circuit data, losses, and calculated PSCAD-equivalent values
`PowerFactory ElmTr3 Data`	Three-winding transformer data, if any three-winding transformers exist in the project

The script is run from PowerFactory and expects the current script object to define:

Attribute	Purpose
`excel_output_path`	Full path to the Excel file to create

This exported workbook is used as input to `compare_component_data_with_pscad.py`.

7.4 Export DSL checksums from PowerFactory

`get_dsl_checksums_from_powerfactory.py` exports checksums for encrypted DSL model types in the PowerFactory project. The script is delivered with the PowerFactory wrapper `Get DSL Checksums.pfd`.

It scans all `BlkDef` objects and records model types containing the encrypted model marker:

```
001! Encrypted model; Editing not possible.
```

The output Excel workbook contains one sheet:

Sheet	Content
`DSL Encryption Data`	DSL model type name and checksum

The script is run from PowerFactory and expects the current script object to define:

Attribute	Purpose
`excel_output_path`	Full path to the Excel file to create

7.5 Export relay data from PowerFactory

`get_relay_data_from_powerfactory.py` exports relay and protection setting information from PowerFactory to Excel. The script is delivered with the PowerFactory wrapper `Get Relay Data.pfd`.

The output workbook makes it easier to compare the PowerFactory protection settings with the corresponding protection settings in PSCAD.

It scans `ElmRelay` objects and creates:

Sheet	Content
`Relay Data`	Overview of relays, application type, out-of-service status, and relevant relay slots
One sheet per relay	Detailed protection functions, IEC symbol, ANSI number, out-of-service status, relay type, tripping direction, characteristic, threshold value, and time value

The script filters out common auxiliary slots such as voltage transformers, current transformers, measurement blocks, PLLs, logic blocks, clocks, and filters.

The script is run from PowerFactory and expects the current script object to define:

Attribute	Purpose
`excel_output_path`	Full path to the Excel file to create

7.6 Compare PowerFactory and PSCAD component data

`compare_component_data_with_pscad.py` compares cable and transformer data exported from PowerFactory with the corresponding components in a PSCAD project.

The script reads the Excel workbook created by `get_component_data_from_powerfactory.py`, then opens the configured PSCAD project and extracts cable and transformer data from matching EMT model components. It writes the PSCAD values to the same workbook and adds comparative sheets showing whether the PSCAD values match the equivalent values calculated from PowerFactory.

At the top of the script, configure:

```
pscad_project_name = 'Solbakken'
excel_path = r'E:\Users\<username>\Solbakken_PowerFactory_Component_Data.xlsx'
SET_PARAMS = False
```

Setting	Purpose
`pscad_project_name`	Name of the PSCAD project to inspect
`excel_path`	Path to the PowerFactory component data workbook
`SET_PARAMS`	If `True`, PSCAD cable and transformer parameters are overwritten with values derived from PowerFactory. Keep this `False` unless parameter synchronization is intended.

The script supports comparison of:

Component type	Notes
Cables	Compares length and positive-/zero-sequence series and shunt values
Two-winding transformers	Supports `db_xfmr_3p2w` and `xfmr-3p2w` PSCAD component definitions
Three-winding transformers	Supports `xfmr-3p3w2` if three-winding transformer data exists in the Excel workbook

The script appends comparative sheets to the same Excel workbook:

Sheet	Content
`Comparative Cable Data`	PowerFactory values, PSCAD values, and percentage differences
`Comparative ElmTr2 Data`	PowerFactory two-winding transformer values, PSCAD values, and percentage differences
`Comparative ElmTr3 Data`	PowerFactory three-winding transformer values, PSCAD values, and percentage differences, if applicable

7.7 Clean unused PSCAD project definitions

`clean_project_definitions.py` deletes unused component definitions from a PSCAD project.

At the top of the script, configure the PSCAD project name:

```
project = pscad.project("Solbakken")
```

The script repeatedly:

1. Finds definitions with zero instances.
2. Deletes them.
3. Saves the project.
4. Repeats until no unused definitions remain.

Use this script with care. It modifies and saves the PSCAD project.

7.8 List signals in a `.psout` file

`list_psout_signals.py` lists and organizes signals from a PSCAD `.psout` file. It is useful when configuring `figureSetup.csv` or troubleshooting missing signal names.

Example:

```
python utility_scripts/list_psout_signals.py "export\MTB_Example\Solbakken_1.psout"
```

Include generic multimeter signals:

```
python utility_scripts/list_psout_signals.py "export\MTB_Example\Solbakken_1.psout" --multimeters
```

Arguments:

Argument	Required	Purpose
`path`	Yes	Path to the `.psout` file
`-m`, `--multimeters`	No	Include generic multimeter signals. By default, non-MTB multimeter signals are hidden.

The output is grouped into:

Group	Meaning
`MAIN CANVAS SIGNALS`	Signals found directly on the PSCAD main canvas
`SUB-MODULE SIGNALS`	Signals found inside submodules

7.9 Recover `.psout` files after a PSCAD OOM crash

`recover_psout_files.py` recovers `.psout` files from the PSCAD build folder after a crash, such as an out-of-memory crash during batch execution.

The script uses the `caseRankTaskID.csv` mapping generated by the MTB PSCAD execution workflow to rename task-ID-based `.psout` files back to case-rank-based filenames.

Example:

```
python utility_scripts/recover_psout_files.py `
  --buildPath "C:\Path\To\PSCAD\Build\Folder" `
  --exportPath "..\export" `
  --csvPath "..\caseRankTaskID.csv" `
  --projectName "Solbakken"
```

Arguments:

Argument	Required	Default	Purpose
`-b`, `--buildPath`	Yes		Path to the `.psout` files that should be recovered
`-e`, `--exportPath`	No	`..\export`	Root export folder where recovered files are moved
`-n`, `--projectName`	No		Optional project-name filter. If supplied, only matching `.psout` files are recovered
`-c`, `--csvPath`	No	`..\caseRankTaskID.csv`	CSV file mapping PSCAD task IDs to MTB case ranks

Recovered files are moved to a timestamped folder named like:

```
OOM_YYYYMMDDHHMMSS
```

Inside that folder, files are renamed to the normal MTB result naming pattern:

```
<ProjectName>_<CaseRank>.psout
```

7.10 Notes and cautions

- The PowerFactory scripts must be run in a PowerFactory environment where the `powerfactory` Python module is available.
- The PSCAD scripts require `mhi.pscad` and, for `.psout` inspection, `mhi.psout`.
- Some scripts contain project-specific defaults such as `Solbakken` or local Excel paths. Review these values before running the scripts on another project.
- `compare_component_data_with_pscad.py` can modify PSCAD component parameters when `SET_PARAMS = True`.
- `clean_project_definitions.py` modifies and saves the PSCAD project.
- Always keep a backup or use version control before running scripts that modify PowerFactory or PSCAD projects.

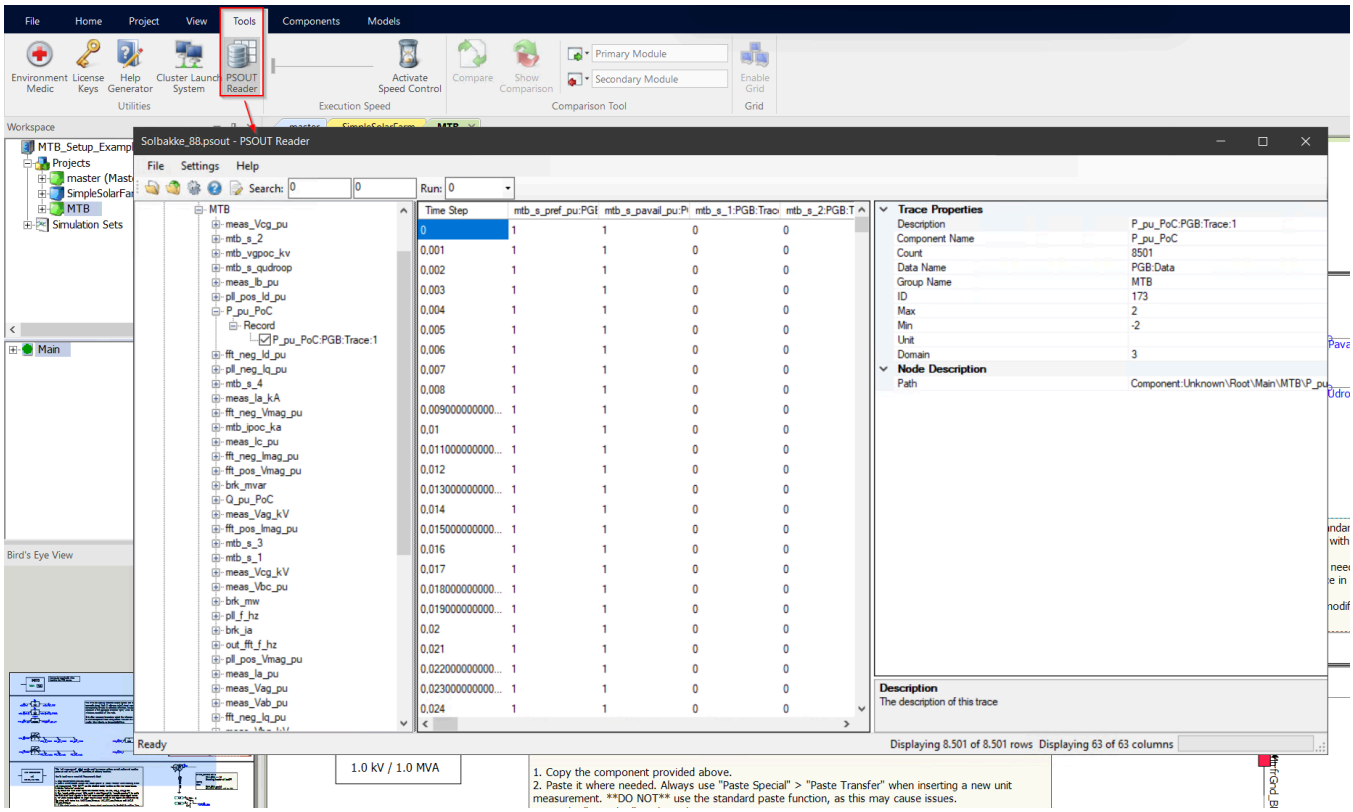
A. Working with .psout files

A.1 Introduction

The .psout format is a modern, proprietary binary format introduced in PSCAD V5 for storing EMTDC simulation data. It replaces legacy text-based `.out` files, offering significantly smaller file sizes, faster read/write speeds, and the ability to store multiple hierarchical simulation runs, including sequential and parallel results, in a single file.

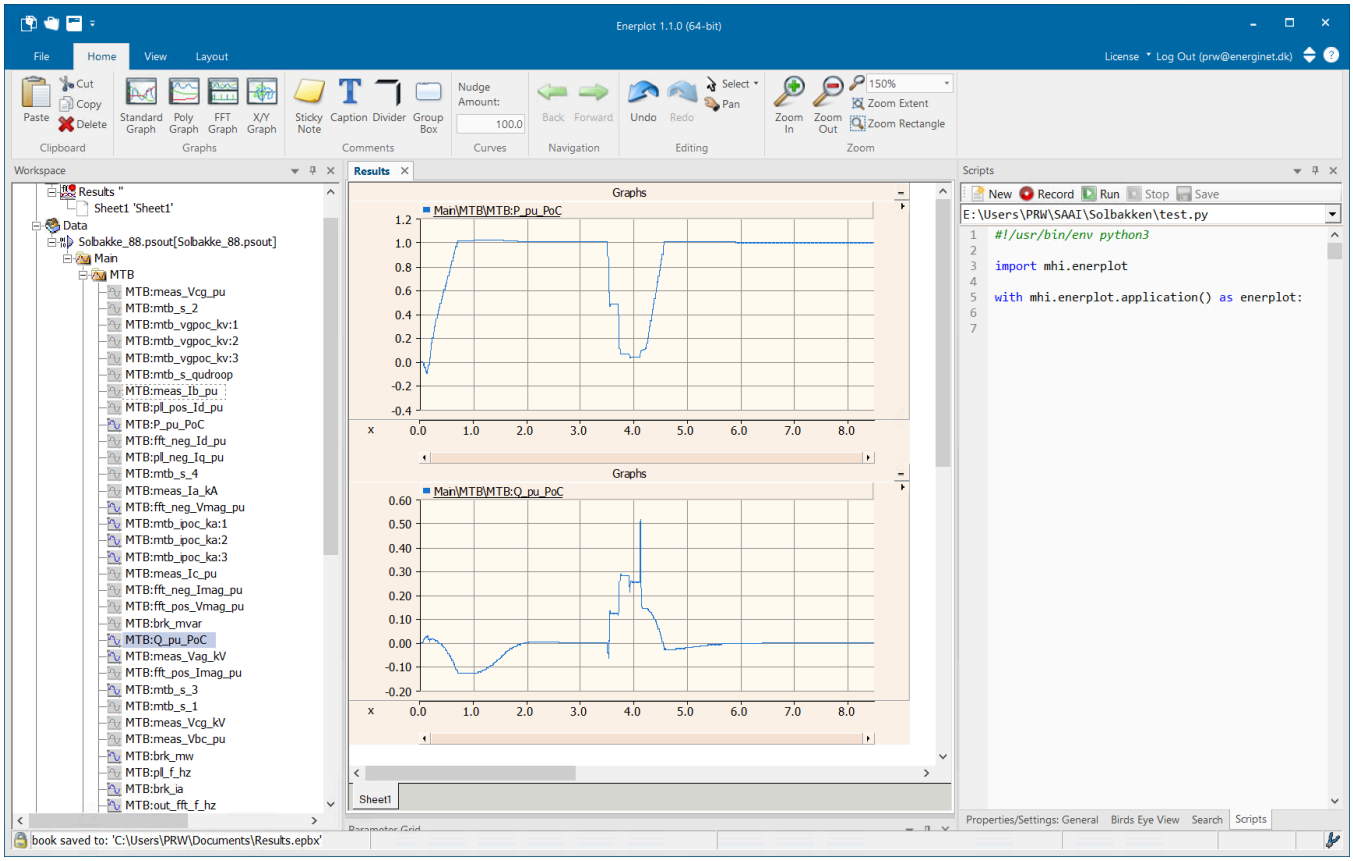
A.2 PSOUT Reader

`.psout` files can be viewed from within PSCAD using the PSOUT Reader.



A.3 Enerplot

MHI also provides Enerplot, which can be used for plotting `.psout` files and for post-processing data using the Python mhi.enerplot library.



A.4 MTB library functions and scripts

The MTB plotter uses the lower-level mhi-psout Python library through a small set of higher-level helper functions in process_psout.py. These helpers make it easier to extract and plot `.psout` signals.

These library functions are:

- **getPsoutSignal**: used by **getPsoutSignals** to obtain the time-domain values and the values of a signal or signal array.
- **getPsoutSignals**: extracts a list of signals as a Pandas DataFrame.
- **findPsoutSignalPath**: finds the relative signal path name or names for a specified signal name.

Two scripts are also provided:

- psout*to*csv: extracts all signals listed in figureSetup.csv from `.psout` files and converts them to human-readable `.csv` format.
- list*psout*signals: lists all signals and their relative signal paths in a `.psout` file.

A.4.1 Script examples

- For more information on using psout*to*csv, call the script from the command line with `-h` or `--help`, e.g.

```
E:\Users\PRW\SAAI\Solbakke_TSO\MTB\plotter>py psout_to_csv.py -h
usage: psout_to_csv [-h] [-p [PSOUTFOLDER]] [-o [OUTPUTROOTFOLDER]] [-f
[FIGURESETUPPATH]] [-c [COMPRESSIONTYPE]] [-q]
                    [-v]

Convert .psout to .csv with optional compression if required
```

```
options:
  -h, --help            show this help message and exit
  -p, --psoutFolder [PSOUTFOLDER]
                        the folder where the .psout files are located
  -o, --outputRootFolder [OUTPUTROOTFOLDER]
                        the output root folder where the date-time stamped folder will
be created and the [compressed]
                        .csv files will be saved
  -f, --figureSetupPath [FIGURESETUPPATH]
                        the path to the figureSetup.csv file
  -c, --compressionType [COMPRESSIONTYPE]
                        the output compression type e.g. .zip, .bz2, .gz or .xz
  -q, --quiet           run quietly
  -v, --version         show program's version number and exit
```

- To list all signals and their relative signal paths in a `.psout` file using list*psout*signals,

```
E:\Users\PRW\SAAI\Solbakke_TSO\MTB>py utility_scripts\list_psout_signals.py
export\MTB_27042026160858\Solbakke_88.psout

Signal Hierarchy for: Solbakke_88.psout
======================================================================

[ MAIN CANVAS SIGNALS ]
  ...
  mtb_GroundTRF1

[ SUB-MODULE SIGNALS ]
  ...
  MTB\fft_neg_Id_pu
  MTB\fft_neg_Imag_pu
  MTB\fft_neg_Iq_pu
  MTB\fft_neg_Vmag_pu
  MTB\fft_pos_Id_pu
  MTB\fft_pos_Imag_pu
  MTB\fft_pos_Iq_pu
  MTB\fft_pos_Vmag_pu
  MTB\init_brk
  MTB\meas_Ia_kA
  MTB\meas_Ia_pu
  MTB\meas_Ib_kA
  MTB\meas_Ib_pu
  MTB\meas_Ic_kA
  MTB\meas_Ic_pu
  MTB\meas_Vab_pu
  MTB\meas_Vag_kV
  MTB\meas_Vag_pu
  MTB\meas_Vbc_pu
  ...
```

A.4.2 Function examples

- If the relative signal path of the signals in a `.psout` file is known, the **getPsoutSignals** function can be used to extract a list of signals from the `.psout` file into a Pandas DataFrame, e.g.

```
In [1]: getPsoutSignals('..\\export\\MTB_27042026160858\\Solbakke_88.psout',
['MTB\meas_Ia_pu', 'MTB\meas_Ib_pu', 'MTB\meas_Ic_pu'])
Out[1]:
       time  MTB\meas_Ia_pu  MTB\meas_Ib_pu  MTB\meas_Ic_pu
0     0.000        0.000000        0.000000        0.000000
1     0.001        0.000267        0.000734        0.000512
2     0.002        0.000632        0.000996        0.000524
3     0.003        0.001027        0.001116        0.000651
4     0.004        0.001383        0.001169        0.000919
    ...             ...             ...             ...
8496  8.496        1.000203        0.999704        0.999979
8497  8.497        1.000204        0.999704        0.999979
8498  8.498        1.000204        0.999704        0.999979
8499  8.499        1.000204        0.999704        0.999978
8500  8.500        1.000204        0.999704        0.999978

[8501 rows x 4 columns]
```

- The plotter.py script uses the **findPsoutSignalPath** function to check the location of the MTB instance in case it is not placed directly on the `Main` canvas, e.g.

```
In [2]:
findPsoutSignalPath('..\\export\\MTB_27042026160858\\HVK_Agg_1.psout','mtb_s_pavail_pu'
)
Out[2]: ['External_Grid\\MTB']
```

As there can only be one MTB per project, MTB signals in figureSetup.csv need only be specified as __MTB\mtb*spavail**pu. If the MTB is not located directly on the `Main` canvas, all MTB signals are automatically mapped to the correct relative path, e.g.** *External*Grid\MTB\mtb*spavail_pu__ as per the above example.

- All signal names are not necessarily unique, i.e. for Unit measurement blocks with **Use legacy Unit measurement signal naming = False** in config.ini, these signals have to be specified with their correct relative path signal names in figureSetup.csv in order to correctly specify which signal is meant to be plotted, e.g.

```
In [3]:
findPsoutSignalPath('..\\export\\MTB_27042026160858\\Solbakke_88.psout','fft_pos_Id_pu'
)
Out[3]: ['Unit', 'MTB', 'Unit_2', 'Unit_1']
```

Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

[2.0.0] - 2026-09-07

Added

SIPS (System Protection Scheme)

- Dedicated MTB SIPS signal (`mtb_s_sips`) and Integer-to-Binary decoder block in PowerFactory and PSCAD (#302)
- Separate generation and demand SIPS signals: `mtb_SIPS_g` (`out_sips_g`) and `mtb_SIPS_d` (`out_sips_d`), replacing the single `mtb_SIPS` signal (#310)
- `SIPS_future_use` bit for future arming/enable logic, replacing `SIPS_notused` (#311)
- `guideSIPS` function in the plotter for active power ceiling decoding from SIPS binary output (#300, #317)
- Ramping logic for SIPS release and reduction in `guideSIPS` (#325)
- Delayed SIPS ramp-down logic via `guideSIPS2` with configurable delay and linear down-ramp (#328, #329)
- SIPS demand test cases (1043, 1044, 1087, 1088) added to `figureSetup.csv` and `testcases.xlsx` (#317)
- SIPS figures with y-labels and PGB templates for generation and demand in `figureSetup.csv` (#302, #317)

New Signals and Parameters

- Q(U) droop signal (`QuDroop`) and `Qudroop0` initial setting for reactive power control (#202)
- Available power signal (`Pavail`) and `Pavail0` initial setting (#202)
- Main Transformer Grounding parameter (`MtrfrGnd`) with `MtrfrGnd0` initialization and output signal (#202)

Plotter

- Multiprocessing support in the plotter for significantly faster report generation (#229, #230, #231, #232, #233, #234, #235, #238)
- New cursor metrics for evaluating simulation results (#188, #190, #191, #192, #194, #195, #197, #200)
- Ideal reference waveforms overlaid on figures (#188–#200)
- Multiple columns with legends in HTML reports (#179)
- `psoutPathMTB` configuration option for when the MTB block is not on the PSCAD main canvas (#264)
- Automatic determination of the MTB location in `.psout` files (#270)
- `findPsoutSignalPath` now returns all matching signal paths instead of only the first (#293)
- FRT state handling with voltage hysteresis and updated Iq0 calculation in `genGuideResults`

PSCAD

- Support for running PSCAD as an external client with configurable Fortran compiler and workspace settings via `config.ini` (#239)
- `batch_execute_pscad.py` script for batch execution of PSCAD simulations to work around out-of-memory (OOM) issues (#243)
- PSCAD OOM recovery: `recover_psout_files.py` script to rename and recover `.psout` files after a PSCAD crash (#211, #213)
- CSV export of case rank and task ID for OOM recovery tracking (#211, #213)
- MTB can now be placed in a PSCAD subfolder rather than on the main canvas (#203)
- PGB (Plotter Group Block) synchronization with `figureSetup.csv` via `pscad_synchronize_pgbs.py`, including automatic disabling of unused signals (#273, #274)
- PSCAD tracing and state animation disabled by default; only in-use channels enabled (#263)

PowerFactory

- Option to organise study cases and variations into a dedicated subfolder in PowerFactory (#319)

Utility Scripts

- `list_psout_signals.py`: list all signals contained in a `.psout` file (#279)
- `clean_project_definitions.py`: clean up project definitions in a PowerFactory project (#297)
- `convert_psout_to_csv.py` (moved from root and renamed from `psout_to_csv.py`) (#283, #284)

Co-location

- Initial framework for co-location case handling in `Case` and `PlantSettings` classes and `testcases.xlsx` (#300)

Changed

Naming and Terminology

- SIPS event types renamed for clarity: `SIPS` → `SIPS Generation` (RfG cases) and `SIPS Demand` (DCC cases) (#310, #317)
- Case names updated from `SystemGuard` / `SystemProtect` to `SIPS` for consistency (#202)
- Q(U) rise/fall time parameter renamed to **Q(U) response time** (#253)
- Unit measurement signal naming simplified: `$Unit$` replaced with `Unit` in PSCAD (#287)
- `psout_to_csv.py` moved to `utility_scripts/` and renamed to `convert_psout_to_csv.py` (#283, #284)
- `psoutRecovery` script renamed to `recover_psout_files.py` for consistency with naming conventions (#272)

PowerFactory

- Updated to **PowerFactory 2025 SP4** (`execute_pf.py`) (#244)
- PowerFactory `.pfd` file exported in **2025 format** (#303)

PSCAD

- Improved port handling to filter PSCAD connections based on the current user, preventing port conflicts in multi-user environments (#330)

- Optimized XML parsing in `_parseProjectXML` for improved performance and reduced latency when resolving PGB signal paths (#334)
- Enhanced `updateUMs` function for improved performance and flexibility; verbose output disabled by default (#334)
- Updated Fortran compiler version list in `config.ini` for clarity and accuracy (#335)
- Updated PSCAD setup example to reflect recent MTB changes (#337)

Requirements

- `requirements.txt` split into separate files for MTB and the plotter (#305, #306)
- Added `mhi.pscad` to requirements (#241)
- Relaxed version constraints for `plotly`, `kaleido`, and `tsdownsample` to allow newer releases

Plotter

- Kaleido compatibility updated to support both the older fast version and the newer, slower version (#266, #323, #324)
- Removed deprecated `engine` parameter from `write_image` calls (#331)
- `psoutPathMTB` configuration removed from `read_configs` (now determined automatically) (#270, #271)

Configuration

- `config.ini` documentation updated for clarity and accuracy (#336)
- `config.ini` updated with examples for running PSCAD as an external client, configuring Fortran, and default GitHub project settings (#239, #246)

Documentation

- Reworked the README to provide a clearer MTB overview, workflow, documentation links, requirements, release notes, contribution guidance, and contact details (#339, #340, #341)
- Added a workflow illustration and an annotated HTML report example to the README (#339, #340, #341)
- Corrected the `.bz2` compression extension in the `psout_to_csv.py` help text (#338)

Fixed

PSCAD / psout Processing

- `process_psout.py` fixed to correctly handle multi-dimensional PSCAD signal arrays (#214, #215)
- Fixed crash when `psoutPathMTB` is empty; missing signals in `.psout` are now ignored without exiting (#265)
- Fixed automatic PSCAD certificate acquisition to ensure the volley requirement is met (#240)
- Fixed `execute_pscad.py` to not crash when no Fortran version is specified (#258)
- Fixed `pscad_synchronize_pgbs.py` to execute correctly from the MTB folder (#281)
- Fixed `pscad_synchronize_pgbs.py` failure when the project was not saved before script execution (#288)
- Fixed PGB `mtb_s_pref_pu` that was accidentally disabled (#289)

PowerFactory

- Fixed garbage output in MTB `execute.ComPython` (line 218) (#207)

- Fixed Check PowerFactory Model script for state variable derivatives (#208)
- Fixed Main Transformer Grounding (`MtrfrGnd`) condition check in setup function (#262)
- Fixed logic for DCC cases where `Pavail0` and `MtrfrGnd0` are not used in the test case sheet (#236)

Plotter / HTML Reports

- Fixed first HTML plot cell being too wide before the browser window is resized (#186)
- Fixed image plots broken by the introduction of multiple-column HTML layout (#185)
- Fixed duplicate signal names in `process_psout.py` (#183)
- Fixed default Unit Measurement EMT signal name (#182)
- Fixed bug with non-legacy naming in `guide_functions` library (#299)
- Fixed `psout_to_csv.py` that was broken after `process_psout.py` was updated (#290)
- Fixed HTML syntax error: missing closing tag on the `mtb.css` stylesheet link (#327)
- Fixed array conversion to use `np.array` for compatibility with pandas 2.0+ (#332)
- Suppressed spurious warnings from `choreographer.utils._tmpfile` logger (#333)

Utility Scripts

- Fixed `compare_component_data_with_pscad.py` to ignore geometric models (#189)
- Fixed `get_dsl_checksums.py` minor bug (#180)
- Fixed `sys.exit(1)` call that incorrectly used `os.exit(1)` (#296)

General

- Fixed Python 3.7.x compatibility issue caused by `pypdf` (#267)
- Fixed `Pavail0` assignment logic in the setup function (#202)
- Fixed bug in the test case sheet for custom cases (#242)

[1.1.2] - 2025-06-18

See repository history for earlier release notes.

For further questions or help, please check if the README or the Quickstart Guides contains the answer.

Otherwise, please contact the Energinet simulation model team: simuleringsmodeller@energinet.dk