STAMP DOCUMENTATION

Josep Arévalo Soler, Èlia Mateu Barriendos, Dionysios Moutevelis, Onur Alican, Marc Cheah Mañé

CITCEA-UPC

A Matlab-based Toolbox for Automatic EMT Modelling and Small-Signal Stability Analysis of Power Systems (STAMP)

The main feature of STAMP is the comprehensive small-signal analysis of large power systems in an EMT framework.

The tool is developed in a MATLAB environment and was chosen for its matrix-oriented programming, robust plotting capabilities, and graphical environment (Simulink).

An EMT non-linear model in Simulink is also generated due to its procedures, streamlining the validation of the produced linear models.

The main features of STAMP are summarized as follows:

1. Small-signal stability analysis for power systems of various sizes and compositions.
2. Generation of EMT, non-linear models based on Matlab-Simulink.
3. A library of linear and non-linear EMT models for power system simulation and analysis.

The general structure of STAMP is depicted in Figure 1. STAMP flowchart.. The following sections describe the practical aspects of its implementation, including input data processing, power flow calculation, and linear and non-linear model construction.

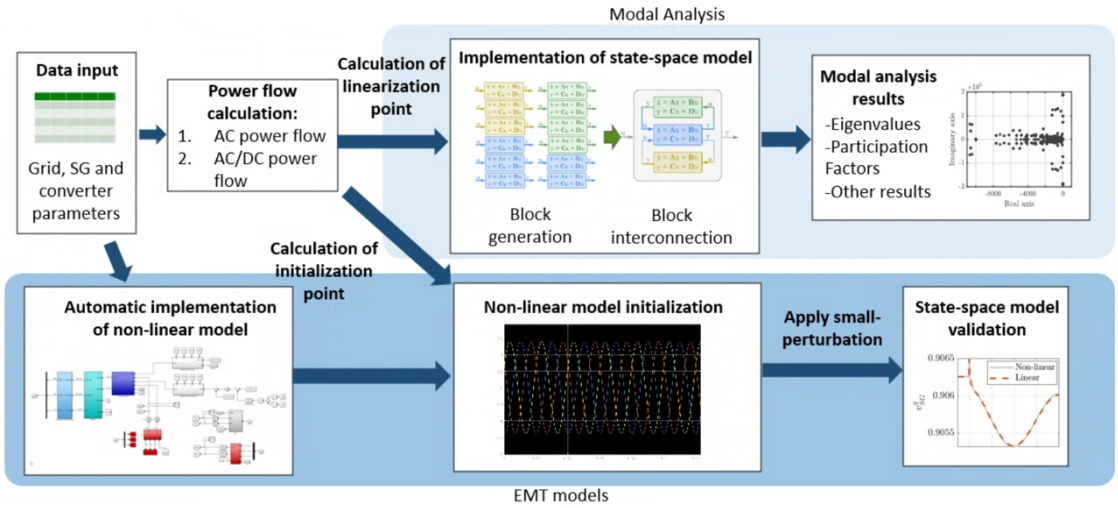


Figure 1. STAMP flowchart.

Table of Contents

Introduction

The stability and dynamics of traditional power systems were dominated by the operation of and interaction between synchronous generators. The most relevant and potentially hazardous for the proper power system operation of these interactions include local and inter-area oscillations between generators, typically manifesting in the subsynchronous timescale, i.e., a timescale way below the network nominal frequency. The electromagnetic dynamics of the generators and the network typically regard higher frequencies and were considered sufficiently damped, thus not threatening the system's stability. For this reason, these dynamics were simplified as algebraic equations in the models used for stability analysis. This modeling approach is the Root Mean Square (RMS) or phasor-domain approach.

For this reason, typical small-signal analysis has traditionally been the most prominent theoretical tool. Therefore, a Root Mean Square (RMS) or phasor-domain approach for modal analysis was sufficient to capture the power system's poorly damped and potentially hazardous oscillations. However, this RMS approach was not sufficient for the analysis of other stability issues in the subsynchronous frequency range (around 10-50 Hz), such as Subsynchronous Resonances (SSR) and Subsynchronous Torsional Interactions (SSTI). These interactions were related to torsional mechanical modes that involved the contribution of the electrical grid and the synchronous generator electrical circuits. As a result, frequency domain analysis methods, such as the complex torque coefficient method or the unity interaction factor (UIF), were presented to understand these interaction phenomena and evaluate their stability.

Integrating power electronics into the power system has triggered new stability phenomena at subsynchronous frequencies, such as Subsynchronous Control Interactions (SSCI), but especially at higher frequencies at harmonic range, i.e., frequencies above 50 Hz. Such harmonic stability issues are mainly due to electromagnetic interactions between the power converter controls, the converter filters, and the electrical grid components, such as lines, transformers, or passive filters. In this context, the electrical dynamics cannot be simplified anymore, and an Electromagnetic (EMT) approach for modal analysis is required to capture such interactions.

Actually, this new approach can represent interactions at all frequency ranges, i.e., covering local and inter-area modes, subsynchronous torsional modes, and harmonic modes. Currently, most commercially available software packages still provide modal analysis tools based on an RMS approach.

The state-space representation facilitates a modular approach to representing and analyzing complex systems. The modularity stems from the superposition property of linear systems, enabling the aggregation of several state-space subsystem blocks into an extensive system. Each subsystem can be arbitrarily defined according to custom specifications, ranging from relatively simple components (e.g., a simple shunt impedance component) to more complicated devices (e.g., power converters with electrical and control sub-systems).

The study of power systems can be categorized into two main branches: large and small studies. Large-signal studies analyze the system behavior under significant disturbances, e.g., faults. On the other hand, small-signal studies analyze the system around its operating point, providing insights into its dynamic stability and control performance around this operating region.

Including High Voltage Direct Current (HVDC) lines further complicates the dynamics of power systems. This introduces additional challenges in modeling and analyzing power system dynamics, as the interaction between HVDC converters and the AC grid must be considered.

Contributions:

1) Modularity -function-based, easy to incorporate new blocks

2) library of predefined state-space elements

3) Non-linear model automatic constructions

4) perfect initialization non-linear model

5) Easy visualization as the non-linear model is implemented in Simulink with its graphic interface (contrary to some other software that uses Python).

User manual for running case studies

This section provides a step-by-step guide for users, including those without prior STAMP experience, to create and run case studies.

This section is organized as follows. Initially, the content within the downloaded folder is outlined. Then, the basic steps for running a new case are explained. If you are okay with the built-in models, no more reading should be needed.

Software and license needed

The tool is intended to be shared as an open code. The minimum required MATLAB version is 2023b.

File structure

The primary folder *Tool* is organized into three sub-folders. The first sub-folder is *00_tool,* which includes all the principal codes needed to perform the EMT small-signal analysis and create and simulate the EMT non-linear models. The second sub-folder is *01_data,* which includes a folder of predefined case studies and the templates needed to create a new case study. The third sub-folder is named *User* and is intended to save data generated from the tool user.

A *.m* file named *main.m* is also inside the main folder. This file is responsible for generating all the case studies. It is the file to run to perform all the studies.

A general view of these sub-folders is depicted in Figure 2.

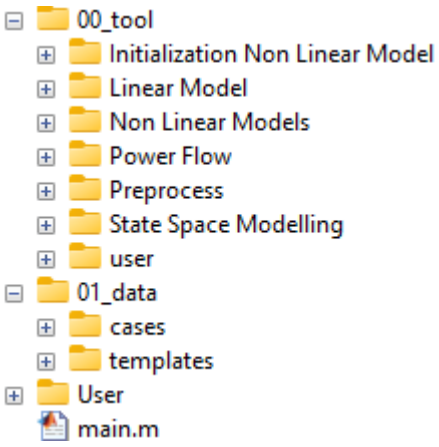


Figure 2. File structure.

Most codes inside both folders are detailed in: Going deep in STAMP. The content behind each sub-folder is detailed next.

00_tool folder

This folder is subdivided into different sub-folders. Each of these sub-folders includes codes to perform different tasks.

Initialization Non Linear Model

The *Initialization Non Linear Model* sub-folder includes built-in codes to initialize the built-in non-linear models. More detailed information on the functions included in this sub-folder can be found in Initialization point calculation.

Linear Model

The *Linear Model* sub-folder includes built-in codes to generate a Simulink model to simulate the linear model automatically. It also includes a sub-folder where these linear models are automatically saved.

Non Linear Models

The *Non Linear Models* sub-folder includes three sub-folders. The first named *layout* includes the built-in functions to generate the non-linear model case study in Simulink automatically. The second named *libraryBlocks* includes the Simulink library of predefined non-linear models. This Simulink library is named *myLibrary.slx.* The third is named *models* and is a folder intended to save the automatically generated non-linear models.

Power Flow

The *Power Flow* sub-folder contains two sub-folders; one contains all MATPOWER needed functions, and the other contains all MATACDC needed functions. This sub-folder also contains built-in functions to adapt the data from the input data generated by the tool to be read for MATPOWER and MATACDC.

Preprocess

The *Preprocess* sub-folder contains built-in functions to adapt the input data to be prepared for use in STAMP. It includes functions to read all the elements' data, such as synchronous generators and voltage source converters.

State Space Modelling

The State Space Modelling sub-folder includes sub-folders for each built-in state-space model. Each sub-folder includes all the functions needed to generate the elements' state-space model. More information on the generation and coding of these state-space models can be found in State-space in-built elements. It also includes a sub-folder named *Analysis,* which includes built-in functions to analyze the linear model generated in the case study. This includes functions to perform eigenvalue and impedance analysis. More information on the output data of these functions can be found in Output data.

01_data

This folder is organized into two sub-folders. One is named *cases* and includes data from predefined case studies. The second one is named *templates* and includes Excel files to generate the user case study.

Input data

The input data is organized in Excel files. Five different Excel files might be needed depending on the case study. These are the following:

1. CASE.xlsx
2. CASE_data_ipc.xlsx
3. CASE_data_sg.xlsx
4. CASE_data_shunt.xlsx
5. CASE_data_vsc.xlsm

Note that all the Excel files have the same naming. In this exemplification, the name of the case study is *CASE.* All the other files should be named accordingly. If the name of the case study is for example *EXAMPLE01* the Excel files have to be named as,

1. EXAMPLE01.xlsx
2. EXAMPLE01_data_ipc.xlsx
3. EXAMPLE01_data_sg.xlsx
4. EXAMPLE01_data_shunt.xlsx
5. EXAMPLE01_data_vsc.xlsm

The description of these files is given next.

Excel 1 (CASE.xlsx)

The Excel file 1 describes the grid topology, loads, generators, power-flow data, and simulation configuration. This Excel file gives a general view of the case study. It has a total of 15 sheets, which are described next.

global

The global sheet imposes the per-unit system used in the case study. As the tool is developed in *qd0*-frame, it also imposes which unit gives the reference angle for the *qd0* angle reference. A general view of this sheet is included in Figure 3. It has a total of 7 columns that need to be filled. The data to be filled is described next.

1. Area: Corresponds to the voltage area. It has to be filled with an integer.
2. SyncArea: Corresponds to the synchronous area. It has to be filled with an integer-
3. Vb_kV: Corresponds to the base voltage for each area in kV.

4. Sb_MVA: Corresponds to the base power for each area in MVA.
5. f_HZ: Corresponds to the base frequency for each synchronous area in Hz.
6. ref_bus: Corresponds to the reference bus for the *qd0*-frame angle. It has to be filled with an integer.
7. ref_element: Corresponds to the element connected to ref_bus that gives the angle of the *qd0*-frame. It can be filled with SG, TH, or GFOR.

	A	B	C	D	E	F	G
1	Area	SyncArea	Vb_kV	Sb_MVA	f_Hz	ref_bus	ref_element
2	1	1	230	100	50	1	SG

Figure 3. Global data sheet.

global_DC

The global_DC sheet imposes the per-unit system used in the DC grid systems. If there is no DC grid in the case study, it can be left empty. A general view of this sheet is included in Figure 4. It has a total of 3 columns that need to be filled. The data to be filled is described next.

1. Area: Corresponds to the voltage area. It has to be filled with an integer.
2. Vb_kV: Corresponds to the base voltage for each area in kV.
3. Sb_MVA: Corresponds to the base power for each area in MVA.

	A	B	C
1	Area	Vb_kV	Sb_MVA
2	1	640	1000
3			

Figure 4. Global_DC data sheet.

AC-NET

The AC-NET sheet imposes the AC grid topology and its line parameters. A general view of this sheet is included in Figure 5. It has a total of 9 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the line. It has to be filled with an integer.
2. Area: Corresponds to the voltage area of each line. It has to be filled with an integer.
3. SyncArea: Corresponds to the synchronous area of each line. It has to be filled with an integer.
4. Bus_from: Corresponds to the starting bus of the line. It has to be filled with an integer.
5. Bus_to: Corresponds to the end bus of the line. It has to be filled with an integer.
6. R: Corresponds to the resistance of this line according to the per unit system defined in Area.
7. X: Corresponds to the impedance of this line according to the per unit system defined in Area.
8. B: Corresponds to the susceptance of this line according to the per unit system defined in Area.
9. State: Corresponds to the status of the line. It can be filled with a 1 or 0. A 1 indicates that the line is connected. A 0 indicates that the line is not connected.

	A	B	C	D	E	F	G	H	I
1	number	Area	SyncArea	bus_from	bus_to	R	X	B	state
2	1	1	1	1	2	0,01	0,085	0,176	1
3	2	1	1	1	3	0,017	0,092	0,158	1
4	3	1	1	2	4	0,032	0,161	0,306	1
5	4	1	1	3	6	0,039	0,17	0,358	1
6	5	1	1	4	5	0,0085	0,072	0,149	1
7	6	1	1	5	6	0,0119	0,1008	0,209	1
8									

Figure 5. AC-NET data sheet.

DC-NET

The DC-NET sheet imposes the DC grid topology and its line parameters. The model used for the DC lines can be found in EMT DC grid. A general view of this sheet is included in Figure 6. It has a total of 13 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the line. It has to be filled with an integer.
2. Area: Corresponds to the voltage area of each line. It has to be filled with an integer.
3. Bus_from: Corresponds to the starting bus of the line. It has to be filled with an integer.
4. Bus_to: Corresponds to the end bus of the line. It has to be filled with an integer.
5. Ra: Corresponds to the resistance parameter of branch *a* according to the per unit system defined in Area.
6. Rb: Corresponds to the resistance parameter of branch *b* according to the per unit system defined in Area.
7. Rc: Corresponds to the resistance parameter of branch *c* according to the per unit system defined in Area.
8. Xa: Corresponds to the impedance parameter of branch *a* according to the per unit system defined in Area.
9. Xb: Corresponds to the impedance parameter of branch *b* according to the per unit system defined in Area.
10. Xc: Corresponds to the impedance parameter of branch *c* according to the per unit system defined in Area.
11. C: Corresponds to the capacitance parameter of the line according to the per unit system defined in Area.
12. G: Corresponds to the conductance parameter of the line according to the per unit system defined in Area.
13. State: Corresponds to the status of the line. It can be filled with a 1 or 0. A 1 indicates that the line is connected. A 0 indicates that the line is not connected.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	number	Area	bus_from	bus_to	Ra	Rb	Rc	La	Lb	Lc	C	G	state
2	1	1	1	2	0,25	0,1	0,3	0,003	0,001	0,002	0,01	0,1	1
3													

Figure 6. DC-NET data sheet.

Trafo

The *trafo* imposes the transformers and its parameters. The model used for the transformers can be found in Transformers. A general view of this sheet is included in Figure 7. It has a total of 11 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the transformer. It has to be filled with an integer.
2. Area: Corresponds to the voltage area of each transformer. It has to be filled with an integer.
3. SyncArea: Corresponds to the synchronous area of each transformer. It has to be filled with an integer.
4. Bus_from: Corresponds to the starting bus of the transformer. It has to be filled with an integer.
5. Bus_to: Corresponds to the end bus of the transformer. It has to be filled with an integer.
6. R: Corresponds to the transformer resistance parameter according to the per unit system defined in Area.
7. X: Corresponds to the transformer impedance parameter according to the per unit system defined in Area.
8. B: Corresponds to the transformer susceptance parameter according to the per unit system defined in Area.
9. Tap_module: Corresponds to the transformer tap module.
10. Tap_angle: Corresponds to the transformer tap angle.
11. State: Corresponds to the status of the transformer. It can be filled with a 1 or 0. A 1 indicates that the transformer is connected. A 0 indicates that the transformer is not connected.

	A	B	C	D	E	F	G	H	I	J	K
1	number	Area	SyncArea	bus_from	bus_to	R	X	B	tap_module	tap_angle	state
2	1	1	1	1	2	0,001	0,1	0	0	0	1
3											

Figure 7. Transformer data sheet.

TH

The *trafo* imposes the Thévenin equivalent and its parameters. The model used for the Thévenin equivalent can be found in Thévenin. A general view of this sheet is included in Figure 8. It has a total of 14 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the transformer. It has to be filled with an integer.
2. bus: Corresponds to the bus of connection. It has to be filled with an integer.
3. Vn: Corresponds to the Thévenin equivalent base voltage in kV.
4. Sn: Corresponds to the Thévenin equivalent power voltage in MVA.
5. Area: Corresponds voltage area where the Thévenin equivalent is connected to. It has to be filled with an integer.
6. SyncArea: Corresponds to the synchronous area of each Thévenin equivalent. It has to be filled with an integer.
7. R: Corresponds to the equivalent resistance in per unit according to the Thévenin equivalent base voltage and power.
8. X: Corresponds to the equivalent impedance in per unit according to the Thévenin equivalent base voltage and power.
9. State: Corresponds to the status of the Thévenin equivalent. It can be filled with a 1 or 0. A 1 indicates that the Thévenin equivalent is connected. A 0 indicates that the Thévenin equivalent is not connected.

10. P: Corresponds to the active power that the Thévenin equivalent is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
11. Q: Corresponds to the reactive power that the Thévenin equivalent is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
12. V: Corresponds to the voltage at the connection bus of the Thévenin equivalent. It is used for the power flow calculation.
13. Theta: Corresponds to the angle at the connection bus of the Thévenin equivalent. It is used for the power flow calculation.
14. Type: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	number	bus	Vn	Sn	Area	SyncArea	R	X	state	P	Q	V	theta	type
2	1	1	230	100	1	1	0,01	0,1	1	0	0,1	1	0	1
3														

Figure 8. Thévenin equivalent data sheet.

Load

The *load* imposes the loads and its parameters. The model used for the loads can be found in Load. A general view of this sheet is included in Figure 9. It has a total of 10 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the load. It has to be filled with an integer.
2. Area: Corresponds to the voltage area of each load. It has to be filled with an integer.
3. SyncArea: Corresponds to the synchronous area of each load. It has to be filled with an integer.
4. Bus: Corresponds to the bus number where the load is connected to. It has to be filled with an integer.
5. R: Corresponds to the equivalent load resistance in per unit according to the global base defined in the global sheet.
6. X: Corresponds to the equivalent load impedance in per unit according to the global base defined in the global sheet.
7. P: Corresponds to the active power consumed by the load in per unit according to the global base defined in the global sheet.
8. Q: Corresponds to the reactive power consumed by the load in per unit according to the global base defined in the global sheet.
9. Type: Corresponds to the definition type of the load for the power flow calculation. It can be filled with PQ or RX. PQ defines the load as a constant active and reactive power injection. It takes the values from the P and Q columns. RX define the load as a constant impedance. It takes the values from the R and X columns.
10. State: Corresponds to the status of the load. It can be filled with a 1 or 0. A 1 indicates that the load is connected. A 0 indicates that the load is not connected.

	A	B	C	D	E	F	G	H	I	J
1	number	Area	SyncArea	bus	R	X	P	Q	type	state
2	1	1	1	2	0	0	1.25	0.5	PQ	1
3	2	1	1	3	1	0,1	0	0	RX	1
4	3	1	1	5	0	0	1	0.35	PQ	1
5										

Figure 9. Load data sheet.

Shunt

The *shunt* imposes the shunt elements and its parameters. A general view of this sheet is included in Figure 10. It has a total of 8 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the shunt element. It has to be filled with an integer.
2. Bus: Corresponds to the bus number where the shunt is connected to. It has to be filled with an integer.
3. Vn: Corresponds to the base voltage of the shunt element in kV.
4. Sn: Corresponds to the base power of the shunt element in MVA.
5. Area: Corresponds to the voltage area of each shunt element. It has to be filled with an integer.
6. SyncArea: Corresponds to the synchronous area of each shunt element. It has to be filled with an integer.
7. State: Corresponds to the status of the shunt element. It can be filled with a 1 or 0. A 1 indicates that the shunt element is connected. A 0 indicates that the shunt element is not connected.
8. Type: Corresponds to the topology of the shunt element. It can be filled with a RLC or C-type.

	A	B	C	D	E	F	G	H
1	number	bus	Vn	Sn	Area	SyncArea	state	type
2	1	1	230	100	1	1	1	RLC
3	2	2	230	100	1	1	1	C-type
4								

Figure 10. Shunt data sheet.

SG

The *SG* imposes the synchronous generators and its parameters. The models used for the synchronous generators' elements can be found in Synchronous Generator. A general view of this sheet is included in Figure 11, Figure 12 and Figure 13. It has a total of 32 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the synchronous generator. It has to be filled with an integer.
2. Bus: Corresponds to the bus number where the synchronous generator is connected to. It has to be filled with an integer.
3. Vn: Corresponds to the base voltage of the synchronous generator element in kV.
4. Sn: Corresponds to the base power of the synchronous generator element in MVA.
5. Area: Corresponds to the voltage area of each synchronous generator element. It has to be filled with an integer.

6. SyncArea: Corresponds to the synchronous area of each synchronous generator element. It has to be filled with an integer.
7. State: Corresponds to the status of the synchronous generator. It can be filled with a 1 or 0. A 1 indicates that the synchronous generator is connected. A 0 indicates that the synchronous generator is not connected.
8. P: Corresponds to the active power that the synchronous generator is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
9. Q: Corresponds to the reactive power that the synchronous generator is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
10. V: Corresponds to the voltage at the connection bus of the synchronous generator. It is used for the power flow calculation.
11. Theta: Corresponds to the angle at the connection bus of the synchronous generator. It is used for the power flow calculation.
12. Type: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus.
13. Rs: Corresponds to the synchronous generator stator resistance in per unit according to its voltage and power base.
14. Xl: Corresponds to the synchronous generator leakage inductance in per unit according to its voltage and power base.
15. Xd: Corresponds to the synchronous generator d-axis magnetizing inductance in per unit according to its voltage and power base.
16. Xd_tr: Corresponds to the synchronous generator d-axis transient inductance in per unit according to its voltage and power base.
17. Xd_sutr: Corresponds to the synchronous generator d-axis subtransient inductance in per unit according to its voltage and power base.
18. Xq: Corresponds to the synchronous generator q-axis magnetizing inductance in per unit according to its voltage and power base.
19. Xq_tr: Corresponds to the synchronous generator q-axis transient inductance in per unit according to its voltage and power base.
20. Xq_sutr: Corresponds to the synchronous generator q-axis subtransient inductance in per unit according to its voltage and power base.
21. Td0_tr: Corresponds to the synchronous generator d-axis transient open-circuit.
22. Td0_subtr: Corresponds to the synchronous generator d-axis subtransient open-circuit.
23. Tq0_tr: Corresponds to the synchronous generator q-axis transient open-circuit.
24. Tq0_subtr: Corresponds to the synchronous generator q-axis subtransient open-circuit.
25. Rtr: Corresponds to the transformer resistance.
26. Xtr: Corresponds to the transformer impedance.
27. Rsnb: Corresponds to the transformer snubber resistance.
28. Exciter: It allows to choose the exciter model.
29. PSS: It allows to choose the PSS model.
30. Govturb: It allows to choose the governor and turbine model.
31. H: Corresponds to the synchronous generator inertia.
32. D: Corresponds to the synchronous generator damping.

	A	B	C	D	E	F	G	H	I	J	K	L
1	number	bus	Vn	Sn	Area	SyncArea	state	P	Q	V	theta	type
2	1	1	180	310	1	1	1	1.63	0.07	1.025	0	0
3												

Figure 11. SG sheet (part I)

	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	Rs	Xl	Xd	Xd_tr	Xd_subtr	Xq	Xq_tr	Xq_subtr	Tdo_tr	Tdo_subtr	Tqo_tr	Tqo_subtr	Rtr	Xtr	Rsnb
2	0.0025	0.2	1.8	0.3	0.25	1.7	0.55	0.25	8	0.03	0.4	0.05	0.002	0.1	300

Figure 12. SG sheet (part II)

	AB	AC	AD	AE	AF
1	exciter	pss	govturb	H	D
2	AC8C	no	no	6.5	0

Figure 13. SG sheet (part III)

VSC

The *VSC* imposes the two-level voltage source converters and its parameters. The models used for the two-level voltage source converters can be found in Two-level voltage source converter. A general view of this sheet is included in Figure 14. It has a total of 13 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the VSC. It has to be filled with an integer.
2. Bus: Corresponds to the bus number where the VSC is connected to. It has to be filled with an integer.
3. Vn: Corresponds to the base voltage of the VSC in kV.
4. Sn: Corresponds to the base power of the VSC in MVA.
5. Area: Corresponds to the voltage area of each VSC. It has to be filled with an integer.
6. SyncArea: Corresponds to the synchronous area of each VSC. It has to be filled with an integer.
7. State: Corresponds to the status of the VSC. It can be filled with a 1 or 0. A 1 indicates that the VSC is connected. A 0 indicates that the VSC is not connected.
8. P: Corresponds to the active power that the VSC is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
9. Q: Corresponds to the reactive power that the VSC is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
10. V: Corresponds to the voltage at the connection bus of the VSC. It is used for the power flow calculation.
11. Theta: Corresponds to the angle at the connection bus of the VSC. It is used for the power flow calculation.
12. Type: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus.
13. Mode: Corresponds to the control mode associated to each VSC. It can be filled with GFOL, GFOR, STATCOM or WT. GFOL stands for a standardized grid-following control, GFOR stands for a standardized grid-forming control, STATCOM stands for a standardized STATCOM converter and WT stands for a standardized type IV wind turbine model. More detailed information of these model can be found in Two-level voltage source converter.

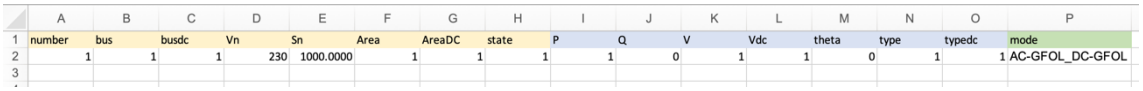
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	number	bus	Vn	Sn	Area	SyncArea	state	P	Q	V	theta	type	mode
2	1	4	230	100	1	1	1	0.85	-0.11	1.025	0	2	GFOR
3	2	6	230	50	1	1	1	0	-0.11	1.025	0	1	GFOL
4													

Figure 14. VSC data sheet.

IPC

The *IPC* imposes the interconnecting power converters and its parameters. The models used for the interconnecting power converters can be found in Modular Multilevel Converter. A general view of this sheet is included in Figure 15. It has a total of 16 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the IPC. It has to be filled with an integer.
2. Bus: Corresponds to the bus number where the IPC is connected to. It has to be filled with an integer.
3. Vn: Corresponds to the base voltage of the IPC in kV.
4. Sn: Corresponds to the base power of the IPC in MVA.
5. Area: Corresponds to the AC voltage area of each IPC. It has to be filled with an integer.
6. AreaDC: Corresponds to the DC voltage area of each IPC. It has to be filled with an integer.
7. SyncArea: Corresponds to the AC synchronous area of each IPC. It has to be filled with an integer.
8. State: Corresponds to the status of the IPC. It can be filled with a 1 or 0. A 1 indicates that the IPC is connected. A 0 indicates that the IPC is not connected.
9. P: Corresponds to the active power that the IPC is injecting to the AC grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
10. Q: Corresponds to the reactive power that the IPC is injecting to the AC grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
11. V: Corresponds to the voltage at the connection bus of the IPC. It is used for the power flow calculation.
12. Vdc: Corresponds to the voltage at the connection bus of the IPC. It is used for the power flow calculation.
13. Theta: Corresponds to the angle at the connection bus of the IPC. It is used for the power flow calculation.
14. Type: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus.
15. Typedc: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus. All on the DC side.
16. Mode: Corresponds to the control mode associated to each IPC. It can be filled with AC-GFOL_DC-GFOL, AC-GFOR_DC-GFOL or AC-GFOL_DC-GFOR. AC-GFOL_DC-GFOL stands for a standardized AC grid-following control and DC grid-following control scheme, AC-GFOR_DC-GFOL stands for a standardized AC grid-forming control and DC grid-following control scheme and AC-GFOL_DC-GFOR stands for a standardized AC grid-following control and DC grid-forming control scheme. More detailed information of these model can be found in Modular Multilevel Converter.



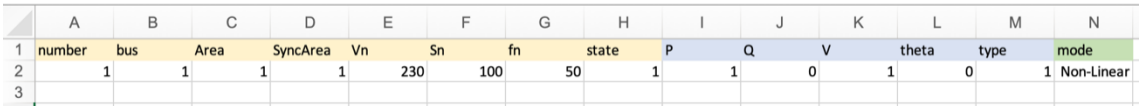
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	number	bus	busdc	Vn	Sn	Area	AreaDC	state	P	Q	V	Vdc	theta	type	typedc	mode
2	1	1	1	230	1000.0000	1	1	1	1	0	1	1	0	1	1	AC-GFOL_DC-GFOL
3																

Figure 15. IPC data sheet.

User

The *User* imposes the user defined elements and its parameters. A general view of this sheet is included in Figure 16. It has a total of 14 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds to the number of the user element. It has to be filled with an integer.
2. Bus: Corresponds to the bus number where the user element is connected to. It has to be filled with an integer.
3. Area: Corresponds to the voltage area of each user element. It has to be filled with an integer.
4. SyncArea: Corresponds to the synchronous area of each user element. It has to be filled with an integer.
5. Vn: Corresponds to the base voltage of the user element in kV.
6. Sn: Corresponds to the base power of the user element in MVA.
7. fn: Corresponds to the base frequency of the user element in Hz.
8. State: Corresponds to the status of the VSC. It can be filled with a 1 or 0. A 1 indicates that the VSC is connected. A 0 indicates that the VSC is not connected.
9. P: Corresponds to the active power that the user element is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
10. Q: Corresponds to the reactive power that the user element is injecting to the grid in per unit according to the global power defined in the global sheet. It is used for the power flow calculation.
11. V: Corresponds to the voltage at the connection bus of the user element. It is used for the power flow calculation.
12. Theta: Corresponds to the angle at the connection bus of the user element. It is used for the power flow calculation.
13. Type: Corresponds to the type of bus defined for the power flow calculation. It can be filled with a 0, 1 or 2. 0 defines the bus as a slack bus. 1 defines the bus as a PQ bus and 2 defines the bus as a PV bus.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	number	bus	Area	SyncArea	Vn	Sn	fn	state	P	Q	V	theta	type	mode
2	1	1	1	1	230	100	50	1	1	0	1	0	1	Non-Linear
3														

Figure 16. User data sheet.

PF

The *PF* sheet imposes the initial point for the power flow calculation solver or it imposes the solution of the power flow. A general view of this sheet is included in Figure 17. The sheet has a total of 5 columns that need to be filled. The data to be filled is described next.

1. Bus: Corresponds to the bus number. It has to be filled with an integer.
2. Area: Corresponds to the voltage area of the bus. It has to be filled with an integer.

3. Vm: Corresponds to the voltage magnitude of the bus in per unit according to the global base.
4. Theta: Corresponds to the angle of the bus in degrees.

	A	B	C	D	E
1	bus	Area	SyncArea	Vm	theta
2	1	1	1	1.02579	0
3	2	1	1	0.99563	-1.772
4	3	1	1	1.01265	-1.4706
5	4	1	1	1.02577	5.9365
6	5	1	1	1.01588	2.9444
7	6	1	1	1.03235	4.1836
8					

Figure 17. PF data sheet.

PF_DC

The *PF_DC* sheet imposes the initial point for the power flow calculation solver or it imposes the solution of the power flow. A general view of this sheet is included in Figure 18. The sheet has a total of 2 columns that need to be filled. The data to be filled is described next.

1. Bus: Corresponds to the bus number. It has to be filled with an integer.
2. Vm: Corresponds to the voltage magnitude of the bus in per unit according to the global base.

	A	B
1	bus	Vm
2	1	1
3	2	1
4		

Figure 18. PF_DC data sheet.

Sim

The sim data sheet imposes different parameters for the non-linear simulation. A general view of this sheet is included in Figure 19. The sheet has a total of 7 columns that need to be filled. The data to be filled is described next.

1. Type: Corresponds to the solver type. It can be set to Discrete or Continuous.
2. Solver: Corresponds to the type of the solver. It can be set to the names that Simulink allows. For example ode1.
3. Tsample_s: Corresponds to the sampling of the output data in seconds.
4. Ts_s: Corresponds to the time step used for the solver in seconds.
5. Tsim_s: Corresponds to the simulation time in seconds.
6. Tstep_s: Corresponds to the set time for a perturbation in seconds.
7. Step_factor: Corresponds to the percentage of perturbation.

	A	B	C	D	E	F	G
1	Type	solver	Tsample_s	Ts_s	Tsim_s	tstep_s	step_factor
2	Discrete	ode1	1.00E-04	1.00E-05	2	0.5	0.01
3							

Figure 19. Sim data sheet.

Excel 2 (CASE_data_ipc.xlsx)

The Excel file 2 describes the specific parameters for the interconnecting power converters. These are described next.

AC-GFOL_DC-GFOL

The *AC-GFOL_DC-GFOL* sheet imposes the specific parameters for the interconnecting power converters in this control mode. A general view of this Excel sheet is depicted in Figure 20. The sheet has a total of 24 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds the IPC number. It should match with the number of the IPC defined in Excel 1. It has to be filled with an integer.
2. Bus: Corresponds the IPC bus number. It should match with the bus of the IPC defined in Excel 1. It has to be filled with an integer.
3. Busdc: Corresponds the IPC dc bus number. It should match with the dc bus number of the IPC defined in Excel 1. It has to be filled with an integer.
4. Rc: Corresponds with the IPC coupling resistance in per unit according with the base of the IPC defined in Excel 1.
5. Xc: Corresponds with the IPC coupling impedance in per unit according with the base of the IPC defined in Excel 1.
6. Ra: Corresponds with the IPC arm resistance in per unit according with the base of the IPC defined in Excel 1.
7. Xa: Corresponds with the IPC arm impedance in per unit according with the base of the IPC defined in Excel 1.
8. kpPLL: Corresponds with the proportional gain of the PLL in per unit according with the base of the IPC defined in Excel 1.
9. kiPLL: Corresponds with the integral gain of the PLL in per unit according with the base of the IPC defined in Excel 1.
10. kpIs: Corresponds with the proportional gain of the inner current loop in per unit according with the base of the IPC defined in Excel 1.
11. kiIs: Corresponds with the integral gain of the inner current loop in per unit according with the base of the IPC defined in Excel 1.
12. kpIsum: Corresponds with the proportional gain of the additive current loop in per unit according with the base of the IPC defined in Excel 1.
13. kiIsum: Corresponds with the integral gain of the additive current loop in per unit according with the base of the IPC defined in Excel 1.
14. kpEt: Corresponds with the proportional gain of the total energy loop in per unit according with the base of the IPC defined in Excel 1.
15. kiEt: Corresponds with the integral gain of the total energy loop in per unit according with the base of the IPC defined in Excel 1.
16. kpPac: Corresponds with the proportional gain of the active power loop in per unit according with the base of the IPC defined in Excel 1.
17. kiPac: Corresponds with the integral gain of the active power loop in per unit according with the base of the IPC defined in Excel 1.
18. kpQ: Corresponds with the proportional gain of the reactive power loop in per unit according with the base of the IPC defined in Excel 1.
19. kiQ: Corresponds with the integral gain of the reactive power loop in per unit according with the base of the IPC defined in Excel 1.
20. kpVac: Corresponds with the proportional gain of the AC voltage loop in per unit according with the base of the IPC defined in Excel 1.
21. tPac: Corresponds with the filter time constant of the active power loop in seconds.
22. tQ: Corresponds with the filter time constant of the reactive power loop in seconds.
23. tVac: Corresponds with the filter time constant of the AC voltage loop in seconds.

24. delay: Corresponds with the modulation delay in seconds.

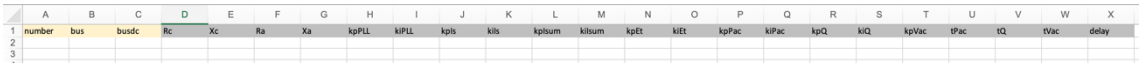


Figure 20. IPC AC-GFOL_DC-GFOL data sheet.

AC-GFOL_DC-FOR

The *AC-GFOL_DC-GFOR* sheet imposes the specific parameters for the interconnecting power converters in this control mode. A general view of this Excel sheet is depicted in Figure 21. The sheet has a total of 24 columns that need to be filled. The data to be filled is described next.

1. Number: Corresponds the IPC number. It should match with the number of the IPC defined in Excel 1. It has to be filled with an integer.
2. Bus: Corresponds the IPC bus number. It should match with the bus of the IPC defined in Excel 1. It has to be filled with an integer.
3. Busdc: Corresponds the IPC dc bus number. It should match with the dc bus number of the IPC defined in Excel 1. It has to be filled with an integer.
4. Rc: Corresponds with the IPC coupling resistance in per unit according with the base of the IPC defined in Excel 1.
5. Xc: Corresponds with the IPC coupling impedance in per unit according with the base of the IPC defined in Excel 1.
6. Ra: Corresponds with the IPC arm resistance in per unit according with the base of the IPC defined in Excel 1.
7. Xa: Corresponds with the IPC arm impedance in per unit according with the base of the IPC defined in Excel 1.
8. kpPLL: Corresponds with the proportional gain of the PLL in per unit according with the base of the IPC defined in Excel 1.
9. kiPLL: Corresponds with the integral gain of the PLL in per unit according with the base of the IPC defined in Excel 1.
10. kpIs: Corresponds with the proportional gain of the inner current loop in per unit according with the base of the IPC defined in Excel 1.
11. kiIs: Corresponds with the integral gain of the inner current loop in per unit according with the base of the IPC defined in Excel 1.
12. kpIsum: Corresponds with the proportional gain of the additive current loop in per unit according with the base of the IPC defined in Excel 1.
13. kiIsum: Corresponds with the integral gain of the additive current loop in per unit according with the base of the IPC defined in Excel 1.
14. kpEt: Corresponds with the proportional gain of the total energy loop in per unit according with the base of the IPC defined in Excel 1.
15. kiEt: Corresponds with the integral gain of the total energy loop in per unit according with the base of the IPC defined in Excel 1.
16. kpVdc: Corresponds with the proportional gain of the DC voltage loop in per unit according with the base of the IPC defined in Excel 1.
17. kiVdc: Corresponds with the integral gain of the DC voltage loop in per unit according with the base of the IPC defined in Excel 1.
18. kpQ: Corresponds with the proportional gain of the reactive power loop in per unit according with the base of the IPC defined in Excel 1.
19. kiQ: Corresponds with the integral gain of the reactive power loop in per unit according with the base of the IPC defined in Excel 1.

20. kpVac: Corresponds with the proportional gain of the AC voltage loop in per unit according with the base of the IPC defined in Excel 1.
21. tPac: Corresponds with the filter time constant of the active power loop in seconds.
22. tQ: Corresponds with the filter time constant of the reactive power loop in seconds.
23. tVac: Corresponds with the filter time constant of the AC voltage loop in seconds.
24. delay: Corresponds with the modulation delay in seconds.

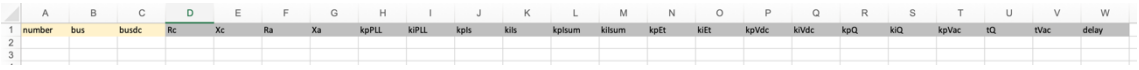


Figure 21. IPC AC-GFOL_DC-GFOR data sheet.

Excel 3 (CASE_data_sg.xlsx)

The Excel file 3 describes the specific parameters for the synchronous machine. These are details the specific parameters for the exciters, power system stabilizers and governor-turbines schemes.

The following exciters are considered:

1. IEEE-AC4A
2. IEEE-ST4B
3. IEEE-ST1
4. IEEE-ST4C
5. IEEE-AC8C

The following power system stabilizers are considered:

1. IEEE-2A

The following governor plus turbine schemes are considered:

2. IEEE-G1
3. Tandem-single-mass
4. Tandem-multi-mass

Excel 4 (CASE_data_shunt.xlsx)

The Excel file 4 describes the specific parameters for the shunt elements. Two shunt elements are considered.

The first one is an RLC filter the parameters are the following:

1. Number: Corresponds to the number of the shunt element.
2. Bus: Corresponds to the bus where the shunt element is connected to.
3. R: Corresponds to the shunt resistance in per unit.
4. L: Corresponds to the shunt inductance in per unit.
5. C: Corresponds to the shunt capacitance in per unit.

The seconds on is a C-type filter. The parameters are the following:

1. Number: Corresponds to the number of the shunt element.
2. Bus: Corresponds to the bus where the shunt element is connected to.
3. C1: Corresponds to the shunt capacitance 1 in per unit.
4. L: Corresponds to the shunt inductance in per unit.

5. C2: Corresponds to the shunt capacitance 2 in per unit.
6. R: Corresponds to the shunt resistance in per unit.

Excel 5 (CASE_data_vsc.xlsx)

The Excel file 5 describes the specific parameters for the VSC converter. This Excel file defines some electrical parameters and control tunning for the different control schemes considered for the VSC. There are included 4 types of schemes. These are the following ones:

1. STATCOM
2. GFOL
3. GFOR
4. WT

Running a case study

The necessary steps for running a case study are introduced next.

1. Define the case study name: For example, CASE. The case study's name should be the Excel file's name.
2. Create and fill the 5 Excel files. Each Excel file has to be named according to the case study name as explained in the Input data. If the system does not have one of the elements defined in the Excel files, creating the corresponding Excel file is unnecessary.
3. Open *main.m.* Set your working directory to that of *main.m*.
4. Configure *main.m* to run the case study. The main file (main.m) requires the setting of the following case definition variables, which are found in the second section, "SET INPUT DATA" of the main.m.
    a. caseName: The name of the case study. Note that the Excel files and simulation models names are automatically generated based on this variable.
    b. path results: The relative path of the folder to store results (figures, files). The default is the 02 results/folder.
    c. fanals: Select the power-flow engine (0: Data from Excel, 1: Fanals, 2: MATACDC).

```
%% SET INPUT DATA

% Case name as in Excel files
    caseName = 'CASE';

% Relative path to the Folder for storing results
    path_results = '02_results\';

% Set power-flow source (0: Excel, 1: Fanals, 2: MATPOWER, 3: MATACDC)
    fanals = 2;

% Flag to indicate if T_case should be used (REE)
    shared_power = 0;
```

Figure 22. Set input data script.

5. Run *main.m.* We recommend executing the file section by section. This approach allows for easier identification of any mistakes made along the way. By proceeding incrementally, you reduce the likelihood of creating simulation models with errors that might be challenging to trace back. Thus, we are next describing what each section does.
6. Read grid topology data. This section is depicted in Figure 23. It runs three different functions.
    a. Set_file_names: This function generates the file names of the 5 Excel files needed for running the case study.

b. Read_data: This function reads the data stored in the general Excel file and converts and stores it in MATLAB format.
c. Preprocess_data: This function adjusts some of the data taken from the Excel file to be compatible with the tool procedures.

```
%% READ GRID TOPOLOGY DATA

% Create excel files and simulink models names
    run set_file_names.m

% Read excel file and generate tables of grid elements
    run read_data.m

% Clean input data
    run preprocess_data.m
```

Figure 23. Read grid topology data section.

7. Read parameters data: This section reads the data in the device Excel files and converts and stores it in MATLAB format. This section is depicted in Figure 24.

```
%% READ PARAMETERS DATA

% Get parameters of generator units from excel files & compute pu base
    run get_parameters.m
```

Figure 24. Read parameters data section.

8. Power flow: This section runs the power flow, updates the operation point of the generator elements, and computes the reference angle (delta_slk) for the *qd0*-frame. It also gives the possibility to update the state of the elements by setting the switches to 'close' in the function *set_breaker_state*. This section is depicted in Figure 25.

```
%% POWER-FLOW

% Update state of switches (open/close lines)
    set_breaker_state('line',1,'close')

% Get Power-Flow results
    run PF_results.m;

% Update operation point of generator elements
    run update_OP.m

% Compute reference angle (delta_slk)
    run delta_slack_acdc.m
```

Figure 25. Power flow section

9. Generate state-space model: This section generates the state-space systems of all the elements. Its output is *l_blocks{}.* It is an array that stores all the state-space systems. This section also provides a graph plot of the network. The section is depicted in .

```
%% GENERATE STATE-SPACE MODEL

% Generate AC & DC NET State-Space Model
    run generate_NET_with_Qneg.m

% Display system graph
graphPlot = generate_NET_graph(T_trafo, T_DC_NET, T_NET, T_nodes, T_load, T_shunt, T_TH, T_SG, T_VSC,T_IPC, T_user, fanals);

% Generate generator units State-Space Model
    run generate_elements.m
```

Figure 26. Generate state-space model section.

10. Build the full system state-space model (see Figure 27): Select the input and outputs of the model. The user is asked to select the system's input and output variables. There are

two options for doing this (you have to uncomment the one you want and comment the other one):

a. Display and select global input and outputs.
b. Write the names of the variables by hand.

```
%% BUILD FULL SYSTEM STATE-SPACE MODEL

% Display and select all global input & outputs
    run display_io.m
    % or... Write manually the inputs and outputs
    %input = {''};
    %output = {''};
    ss_sys = connect(l_blocks{:}, input, output);
```

Figure 27. Build the full system state-space model section.

11. Running the non-linear model simulation. This section automatically builds the non-linear model and simulates it. This section is depicted in Figure 28.

```
%% NON-LINEAR

    if ~isfile(['00_tool/Non Linear Models/models/' nonlinear '.slx'])
        newSys = 1;
    else
        newSys = 0;
    end

% Initialization

     run NET_initialization.m

% Create simulink nonlinear model

    if newSys
        if fanals==3
            run NET_layout_FORCE_ACDC0.m % create AC/DC simulink nonlinear model
        else
            run NET_layout_FORCE.m % create simulink nonlinear model
        end
    else
        open(nonlinear) % open already existing model
    end

% Avoid redundant initialization
    run dependent_states.m

% Set disturbance
    run param_nonlinear.m

% Simulate
    out_nolin = sim(nonlinear);

    MsgBoxH = findall(0,'Type','figure','Name','Initial state conflict');
    close(MsgBoxH);
```

Figure 28. Non-linear section.

12. Running the linear model. This section automatically builds a linear model and simulates it. It is depicted in Figure 29.

```
%% LINEAR MODEL

% Set Linear Simulation parameters
    run param_linear.m

% Create simulink linear model
    if ~isfile(['00_tool/Linear Model/models/' linear '.slx'])
        generate_linear(ss_sys,linear,tstep_lin,Tsim_lin,delta_u) % create simulink linear model
    else
        open(linear) % open already existing model
    end

% Simulate
    out_lin = sim(linear);
```

Figure 29. Linear model section.

13. Validate state-space model. This final section outputs the comparison of the dynamics of the linear and non-linear models by superimposing them. It is depicted in Figure 30.

```
%% VALIDATE LINEAR MODEL

run validate_linear.m
```

Figure 30. Validate linear section.

14. Performs small-signal analysis. The small-signal analysis can be performed with the generated state-space model names *ss_sys.*

Output data

In this section, some of the functionalities included in STAMP to generate, process and plot outputs and results are explained.

FEIG

The function *FEIG(ss_sys,CCOLOR,marker,plotYes)* takes as input a linear system, defined as a Matlab system structure, and calculates a table with the eigenvalues of its state space matrix. Different options (colour, marker type) are passed also as inputs to the function for the case that an eigenvalue plot in the complex plain is required. The latter is selected with a Boolean input that serves as the final input of the system. The provided table also includes common metrics related with the eigenvalues, like for example the frequencies of the resulting modes as well as their damping. If the linear system is evaluated as a function of a varying parameter, as it typically happens for the case of parameter sweeps, the specified instance of the system, whose eigenvalues are to be evaluated, needs to be passed to the function via an index, eg. ss_sys {1}.

FMODAL

The function *FMODAL(ss_sys)* takes as input a linear system, defined as a Matlab system structure, and calculates a table containing the participation factors of all the state variables. This table is also plotted in a Matlab figure.

FMODAL_REDUCED

The function *FMODAL_REDUCED(ss_sys,modeID)* is similar to FMODAL, however it only plots the participation factors related to specific eigenvalues, which are defined in the vector modeID. The result is a table/plot that is a subset of the one produced by FMODAL. This function is particularly useful for large systems, where maybe only a few critically damped eigenvalues are of interest.

FMODAL_REDUCED_th

The function *FMODAL_REDUCED_th(ss_sys,modeID,th)* is Another variation of the FMODAL function, where only the participation factors larger than a given threshold (input variable "th") are included in the produced table and plotted.

get_sensistivy_eingevalues_figure

The function *get_sensitivy_eigenvalues_figure(list_ss,factorArray)* plots the eigenvalues of a linear system as a parameter of the system changes. The input is a list of linear systems (one instance of the linear system evaluated for each parameter value), together with a vector containing the values of the changing parameter itself.

sensitivity_samePF.m

This script produces a list of linear systems with a single varying parameter. The range of the parameter variation is defined outside of an iterating for loop and then the corresponding system parameters are updated with the corresponding values for each iteration. At the end of the script, the *get_sensitivy_eigenvalues_figure* function plots the eigenvalue locus for the different parameter values. This script should be used for cases where the chosen parameter for the sensitivity analysis does not affect the operating point of the system (e.g., the time constant of a low-pass filter).

sensitivity_recalculatePF.m

This is a variation of the previous script, with the only difference being that the power flow calculation is included within the iterating loop. This script should be used for cases where the chosen parameter for the sensitivity analysis affects the operating point of the system (e.g., for the active power reference of a converter).

plot_impedance_sensitivities_qd

This function plots the impedance/admittance of a linear system in the frequency domain for a varying system parameter. The main input is a list of linear systems, similar to the main input of the *get_sensitivy_eigenvalues_figure* function. The second main input is the factorArray vector, which contains the values of the varying parameter itself. The rest of the inputs are related with evaluation and plotting options and include (in order) the maximum frequency to be evaluated (the minimum is set to zero by default), the frequency evaluation step, the type of the evaluated transfer function (impedance or admittance), an option to additionally plot only the qq impedance/admittance and a scaling parameter for a potential transfer function normalization.

plot_impedance_sensitivities_pn

Similar to the previous function but it evaluates and plots the considered admittance/impedance in the positive-negative sequence domain.

plot_passivity_sensitivities

This function plots the passivity index of a linear system in the frequency domain for a varying system parameter. The inputs and their structure are similar to the previous two functions.

plot_nyquist

This function gets two linear systems as inputs (and not two lists of linear systems), defined as Matlab system structure and plots the Nyquist plots for the two eigenvalues of the open-loop

transfer function. The input matrices need to be of matching dimensions; however, this functionality is typically reserved for 2by2 matrices. The maximum frequency and frequency step are also passed as inputs. The Nyquist plots are plotted on a point-to-point basis. An option to interpolate the points via a *pchip* algorithm is included as an input.

IMPORTANT NOTE:

The above four functions apply for any properly defined linear systems. However, for the standard partition between device (e.g., converter) and grid, some considerations need to be first taken. In order to save memory space, the routines *sensitivity_samePF.m* and *sensitivity_recalculatePF.m* that generate the linear model list only produce the linear model of the fully interconnected, closed loop system. If a system partition is necessary the different linear subsystems should be saved independently, before their interconnection. This is implemented in the above two routines by enabling the line of code where the *l_blocks_list* variable is created. This is a structure that contains all the different linear sub-systems of the model on the device level. Once this list is available, the input-output relation between the desired impedance/admittance can be properly defined in order to calculate the required system partition.

Predefined case studies

The predefined case studies are:

1. IEEE-9
2. IEEE-118

Going deep in STAMP

Power flow results

The power flow calculation can be calculated based in 4 different approaches. The first one considers that the power flow solution is calculated in another software. Then, the results of this solution are just inputted in the excel files in the blue columns. The second approach considered a legacy power flow solver developed by the authors of this tool. However, this has proven to not be robust in big systems. The third approach uses MATPOWER to solve the power flow (the authors encourage to use this approach). The fourth approach used MATACDC to calculate the power flow. This last one must be used when dealing with systems with AC and DC grids.

The results of the power flow are stored in a structure named “results”. This structure and its stored variables are described next. The solution of the power flow can be checked by writing the command “results” in the MATLAB command window as depicted in Figure 31.

```
>> results

results =

  struct with fields:

      bus: [16×8 table]
     load: [1×4 table]
    shunt: [4×4 table]
       th: [1×4 table]
       sg: [0×4 table]
      vsc: [2×4 table]
     user: [0×4 table]
```

Figure 31. Power flow results structure.

As it is shown 7 different categories are displayed which shows the results of the power flow for the different elements. Next all the elements' results are described:

1. bus: Stores the results for each bus. It can be checked by writing "results.bus" at the command window. An example is shown in
    a. bus: gives the bus number.
    b. Area: gives the voltage area number.
    c. SyncArea: gives the synchronous are number.
    d. Vm: gives the voltage magnitude based on line-to-line RMS magnitudes.
    e. Theta: gives the voltage angle in degrees.
    f. Type: gives the type of bus for the power flow calculation
    g. Element_X: gives which elements are connected to this bus.

```
>> results.bus

ans =

  16×8 table
```

bus	Area	SyncArea	Vm	theta	type	Element_1	Element_2
1	1	1	0.995	0	"slack"	"TH 1"	<missing>
2	1	1	1.00102652231757	6.32675243281412	"PQ"	"Load 1"	"Shunt 1"
3	1	1	1.00878208576793	6.84101482636596	"PQ"	<missing>	<missing>
4	1	1	1.01577724178945	7.36494900961563	"PQ"	<missing>	<missing>
5	1	1	1.02200996771815	7.89925525514232	"PQ"	<missing>	<missing>
6	1	1	1.02747915464674	8.44464308573083	"PQ"	<missing>	<missing>
7	1	1	1.03218462373405	9.00183482065511	"PQ"	<missing>	<missing>
8	1	1	1.03612714331543	9.57156904932536	"PQ"	<missing>	<missing>
9	1	1	1.03930844690627	10.1546040670235	"PQ"	<missing>	<missing>
10	1	1	1.04173125220591	10.7517213038152	"PQ"	<missing>	<missing>
11	1	1	1.04339928122539	11.3637287746048	"PQ"	<missing>	<missing>
12	1	1	1.04431728167843	11.99146457555	"PQ"	"Shunt 2"	<missing>
13	2	1	1.05378463038763	17.4827354543374	"PQ"	"Shunt 3"	<missing>
14	2	1	1.05400045460194	17.5505883089768	"PQ"	"Shunt 4"	<missing>
15	2	1	1.06578018743135	18.1561835278952	"PQ"	"VSC 1"	<missing>
16	2	1	1.06598397301312	18.2216250580892	"PQ"	"VSC 2"	<missing>

Figure 32. results.bus example.

2. load: Stores the results for each load. It can be checked by writing "results.load" at the command window. An example is shown in Figure 33.
    a. number: gives the load number.
    b. bus: gives the bus number.
    c. P: gives the active power consumed by the load in per unit system of the global reference.
    d. Q: gives the active power consumed by the load in per unit system of the global reference.

```
>> results.load

ans =

  1×4 table

    number    bus              P                Q
    ______    ___    ____________________    _

      1         2     0.00100205409838322     0
```

Figure 33. results.load example.

3. shunt: Stores the results for each shunt. It can be checked by writing “results.shunt” at the command window. An example is shown in Figure 34.
    a. number: gives the shunt number.
    b. bus: gives the bus number.
    c. P: gives the active power consumed by the shunt in per unit system of the global reference.
    d. Q: gives the active power consumed by the shunt in per unit system of the global reference.

```
>> results.shunt

ans =

  4×4 table

    number    bus              P                      Q
    ______    ___    ____________________    ____________________

      1         2     0.00599756743198288      0.300789587511274
      2        12     0.00652753036406925      0.327368251869171
      3        13     1.20682722741564e-05    -0.00264944901027311
      4        14     9.054691334733e-06      -0.00198786353571408
```

Figure 34. results.shunt example.

4. th: Stores the results for each Thévenin equivalent. It can be checked by writing “results.th” at the command window. An example is shown in Figure 35.
    a. number: gives the Thévenin number.
    b. bus: gives the bus number.
    c. P: gives the active power injected by the Thévenin in per unit system of the global reference.
    d. Q: gives the active power consumed by the Thévenin in per unit system of the global reference.

```
>> results.th

ans =

  1×4 table

    number    bus              P                Q
    ______    ___    ____________________    ____________________

      1         1     -0.936384104389309     0.0115171151615734
```

Figure 35. results.th example.

5. sg: Stores the results for each synchronous generator. It can be checked by writing “results.sg” at the command window.
    a. number: gives the synchronous generator number.
    b. bus: gives the bus number.
    c. P: gives the active power injected by the synchronous generator in per unit system of the global reference.

   d. Q: gives the active power consumed by the synchronous generator in per unit system of the global reference.
6. vsc: Stores the results for each two-level voltage source converter. It can be checked by writing “results.vsc” at the command window.
   a. number: gives the two-level voltage source converter number.
   b. bus: gives the bus number.
   c. P: gives the active power injected by the two-level voltage source converter in per unit system of the global reference.
   d. Q: gives the active power consumed by the two-level voltage source converter in per unit system of the global reference.
7. user: Stores the results for each user element. It can be checked by writing “results.user” at the command window.
   a. number: gives the user element number.
   b. bus: gives the bus number.
   c. P: gives the active power injected by the user element in per unit system of the global reference.
   d. Q: gives the active power consumed by the user element in per unit system of the global reference.

State-space in-built elements

The state-space representation facilitates a modular approach to representing and analyzing complex power systems. The modularity stems from the superposition property of linear systems, enabling the different small state-space system blocks to be aggregated to represent a bigger system. This property is also actually the other way around; a larger power system can be partitioned into smaller blocks representing small sub-systems as proposed in this tool.

The modularity property makes the system linearization manageable, enabling the linearization of each subsystem individually and thus increasing flexibility and scalability. In addition, this modular approach facilitates the process of coding and automatizing the system construction, minimizing the margin for human error. Moreover, each linearized subsystem can be part of a library of state-space models that can be reused, reducing redundancy and simplifying the process of building large, complex systems, as is the case in the proposed tool.

Each subsystem can be arbitrarily defined based on custom preferences and specifications. For example, each sub-system can represent specific components such as RL (Resistor-Inductor) filters or PI (Proportional-Integral) controllers. However, the size of each state-space representation is not limited to such detailed components or control structures. Each subsystem can also represent more complex systems, such as complete power converters (i.e., combining control algorithms and electrical systems).

Figure 36 shows an example of the process followed by this tool for building a full state-space representation of a power system model from individual state-space blocks of each electrical component. Linear algebra methods achieve the integration of these subsystems into a complete system. The arrangement between subsystems is based on the inputs and outputs of each subsystem. If the output of one subsystem corresponds to the input of another subsystem, these two subsystems can be interconnected. This subsystem combination creates a meshed structure of state-space subsystems that represents the linear dynamic behavior of the full system.

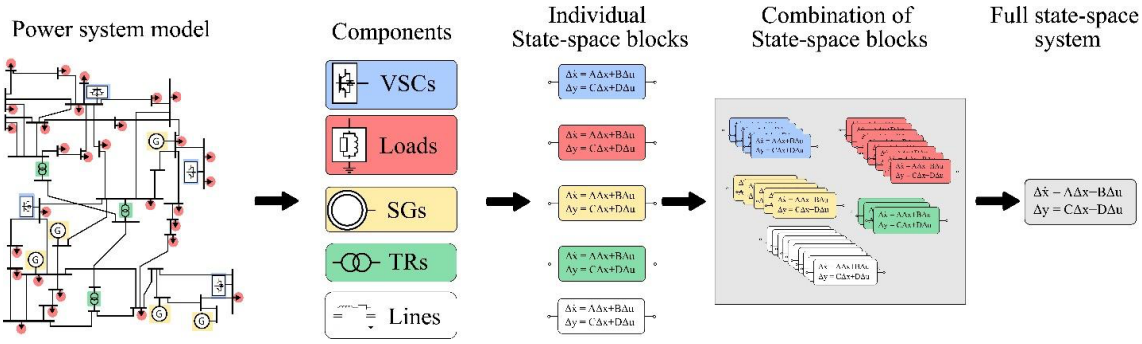


Figure 36. Process to generate the full state-space system.

The elements considered in this tool are presented in Simulink non-linear in-built elements. The state-space generation of these elements is done by means of different functions, which are called from a main function. These are the synchronous generators, the voltage source converters, the interconnecting power converters, and the shunt elements. The following functions generate the state-space system of each of these elements. These functions can be found inside the folder *00_tool/State Space Modelling/.*

1. generate_SG_pu(l_blocks, T_SG, lp_SG, T_global, num_slk, element_slk, REF_w): This function returns the state-space system of the specified synchronous generator. It needs the following inputs:
    a. l_blocks: It is an array list of state-space systems. It can be given empty.
    b. T_SG: It is a table with the data read from Excel file number 3.
    c. lp_SG: It is an array with the linearization point for each synchronous generator.
    d. T_global: It is a table with the global base's data.
    e. num_slk: It is a list with the number of the slack bus.
    f. Element_slk: It is a list with a pointer for the defined slack element.
    g. REF_w: It is a list with the name of the angle reference for the general *qd0*-frame rotation.
2. generate_VSC_pu_with_functions(l_blocks, T_VSC, lp_VSC, T_global, num_slk, element_slk, REF_w): This function returns the state-space system of the specified voltage source converter. It needs the following inputs:
    a. l_blocks: It is an array list of state-space systems. It can be given empty.
    b. T_VSC: It is a table with the data read from Excel file number 5.
    c. lp_VSC: It is an array with the linearization point for each voltage source converter.
    d. T_global: It is a table with the global base's data.
    e. num_slk: It is a list with the number of the slack bus.
    f. Element_slk: It is a list with a pointer for the defined slack element.
    g. REF_w: It is a list with the name of the angle reference for the general *qd0*-frame rotation.
3. generate_IPC_pu_with_functions(l_blocks, T_IPC, lp_IPC, T_global, num_slk, element_slk, REF_w): This function returns the state-space system of the specified interconnecting power converter. It needs the following inputs:
    a. l_blocks: It is an array list of state-space systems. It can be given empty.
    b. T_IPC: It is a table with the data read from Excel file number 2.
    c. lp_IPC: It is an array with the linearization point for each interconnecting power converter.
    d. T_global: It is a table with the global base's data.

e. num_slk: It is a list with the number of the slack bus.
f. Element_slk: It is a list with a pointer for the defined slack element.
g. REF_w: It is a list with the name of the angle reference for the general *qd0*-frame rotation.

4. generate_SHUNT_pu_with_functions(l_blocks, T_shunt, T_global): This function returns the state-space system of the specified shunt element. It needs the following inputs:
   a. l_blocks: It is an array list of state-space systems. It can be given empty.
   b. T_shunt: It is a table with the data read from Excel file number 4.
   c. T_global: It is a table with the global base's data.

Simulink non-linear in-built elements

The Simulink non-linear library includes the elements modeled in state space but in its non-linear form. This library is named *myLibrary.slx* and it can be found inside the folder *00_tool/Non Linear Models/libraryBlocks/*. A general view of the library is depicted in Figure 37. Next, the elements included in the library are described. All the elements are created based on a mask. Each mask requires different parameters to be inputted. All these are described next.

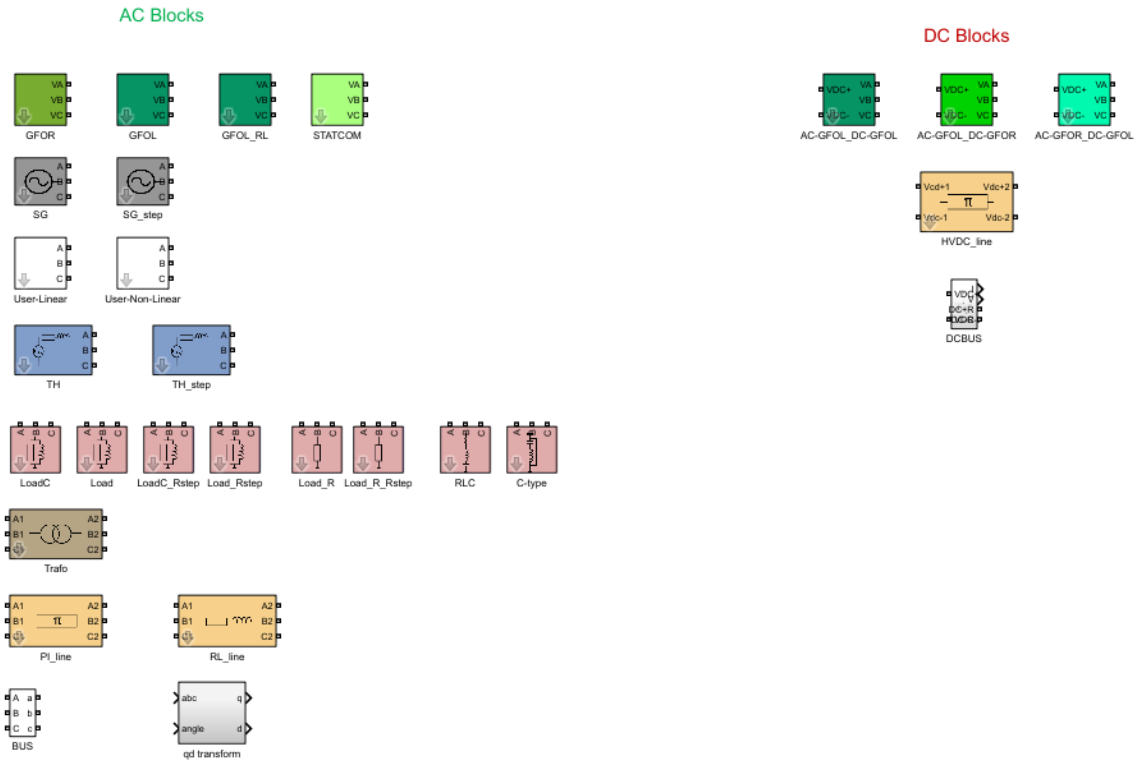


Figure 37. Library of non-linear elements.

PI-Line

The mask of the PI-line on the Simulink library is depicted in Figure 38. The parameters that the mask asks for are depicted in Figure 39. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_line: Table of line parameters.
2. Init_line: Array of initialization values.
3. Num: Line number.

4. Init_cap: vector with the capacitors to initialize.
5. Init_ind: vector with the inductance to initialize.

The scheme model is depicted in Figure 40.

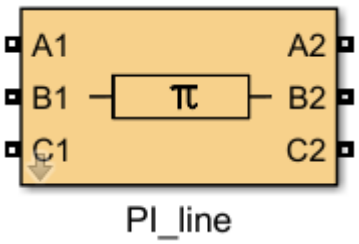


Figure 38. PI-line mask.

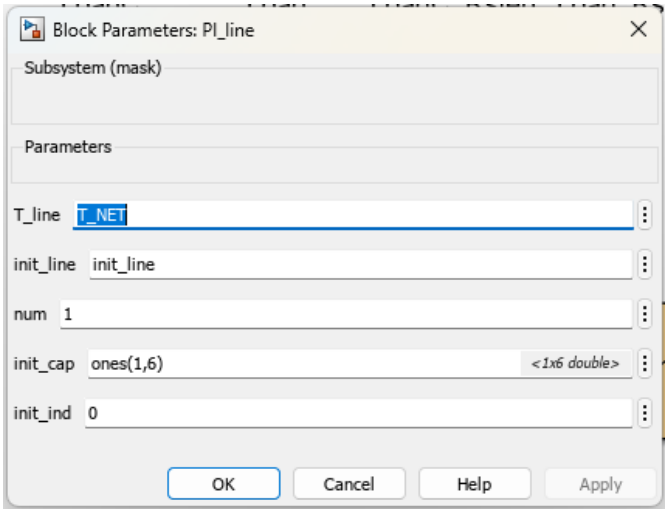


Figure 39. PI-line mask parameters.

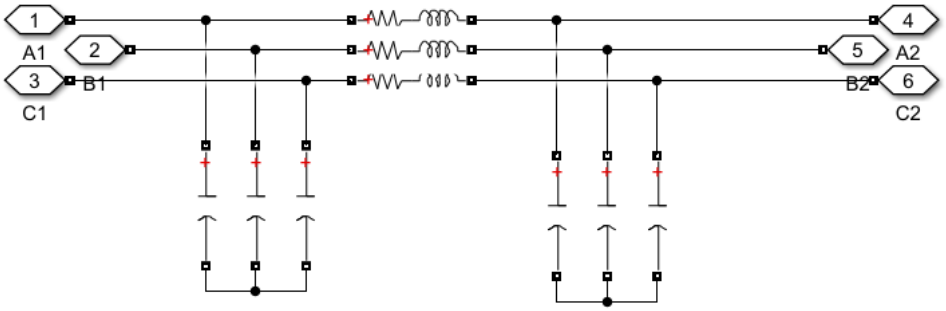


Figure 40. PI-line scheme.

EMT DC grid

The mask of the HVDC line on the Simulink library is depicted in Figure 41. The parameters that the mask asks for are depicted in Figure 42. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_DC_line: Table with the DC line parameters.

2. Init_DC_line: Array with the initialization variables.
3. Num: HVDC line number.

The scheme model is depicted in Figure 43.

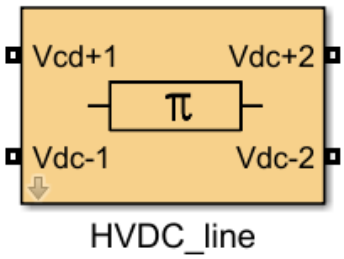


Figure 41. HVDC line mask.

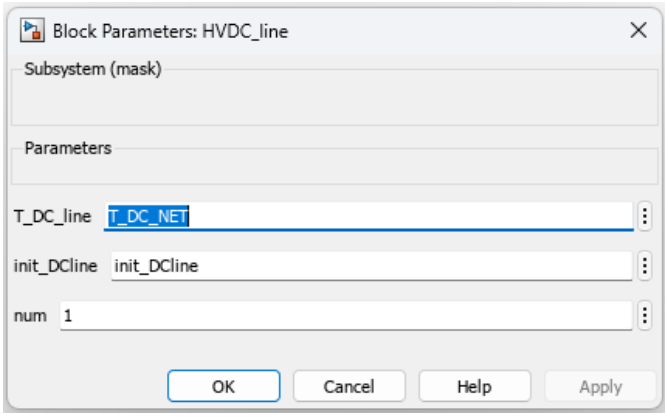


Figure 42. HVDC line mask paramters.

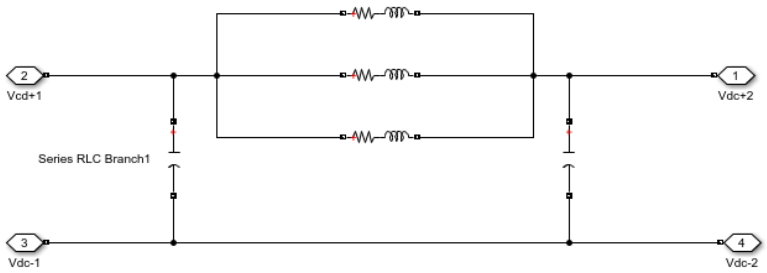


Figure 43. HVDC line scheme.

Transformers

The mask of the transformers on the Simulink library is depicted in Figure 44. The parameters that the mask asks for are depicted in Figure 45. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_trafo: Table with the DC line parameters.
2. Init_trafo: Array with the initialization variables.
3. Num: Transformer number.

The scheme model is depicted in Figure 46.

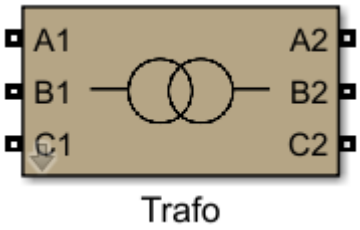


Figure 44. Transformer mask.

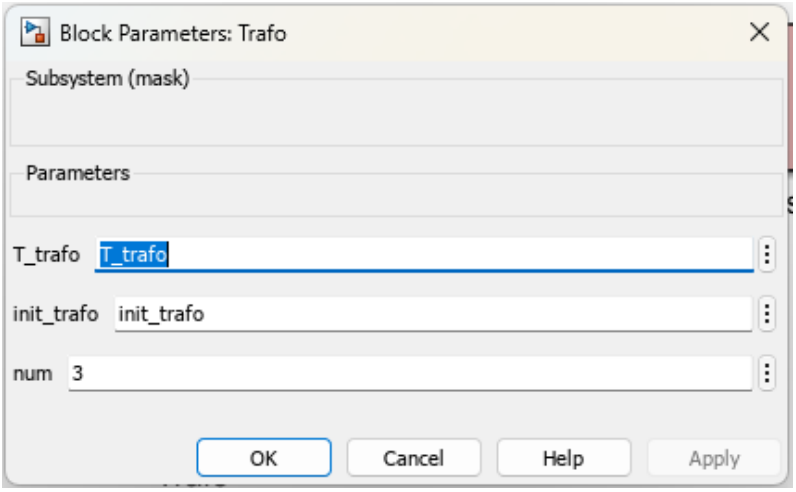


Figure 45. Transformer mask parameters.

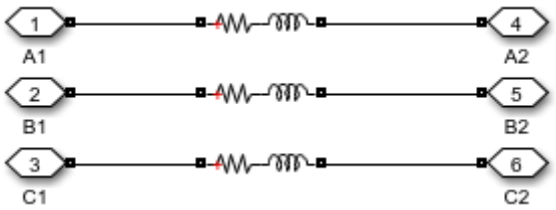


Figure 46. Transformer scheme.

Thévenin

The mask of the Thévenin equivalent on the Simulink library is depicted in Figure 47. The parameters that the mask asks for are depicted in Figure 48. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_TH: Table with the Thévenin paramters.
2. Init_TH: Array with the initialization variables.
3. Num: Thévenin number.

The scheme model is depicted in Figure 49.

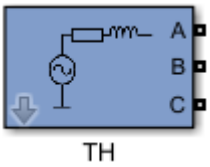


Figure 47. Thévenin scheme.

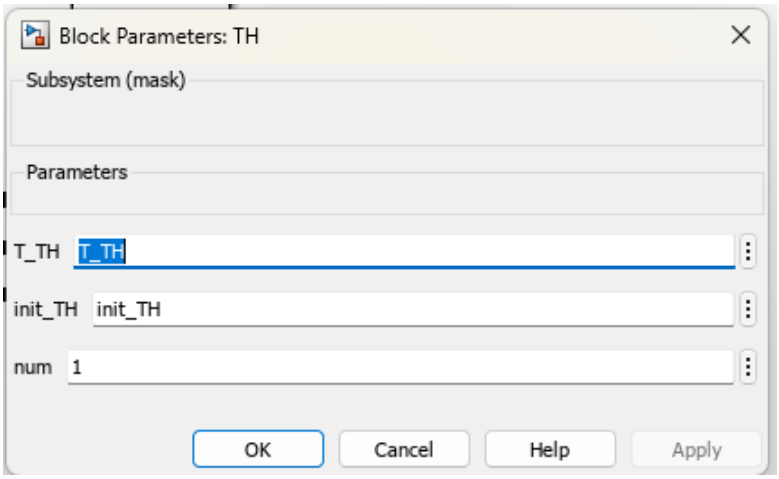


Figure 48. Thévenin mask parameters.

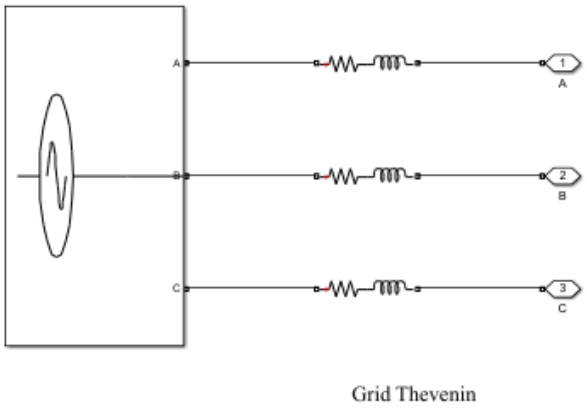


Figure 49. Thévenin scheme.

Load C

The mask of the Load C on the Simulink library is depicted in Figure 50. The parameters that the mask asks for are depicted in Figure 51. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_load: Table with the load parameters.
2. Init_load: Array with the initialization variables.
3. Num: Load number.

The scheme model is depicted in Figure 52.

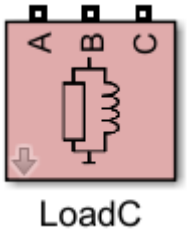


Figure 50. Load C mask.

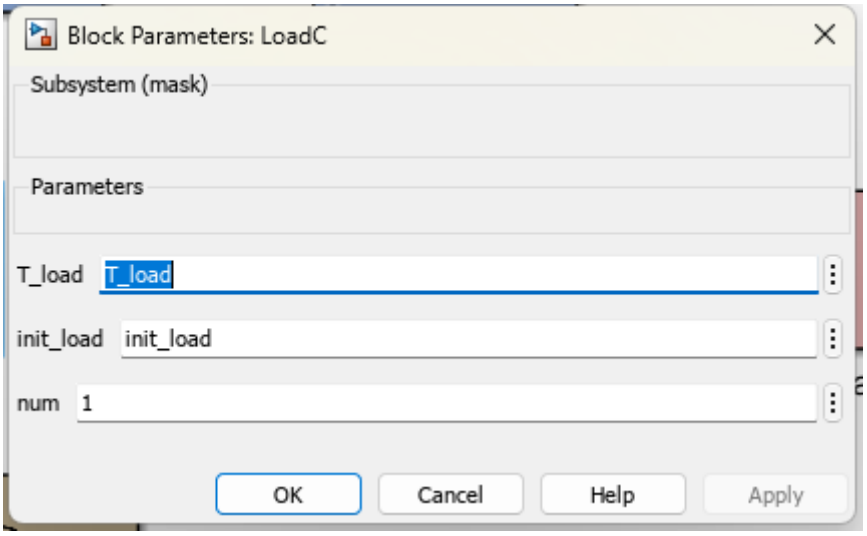


Figure 51. Load C mask parameters.

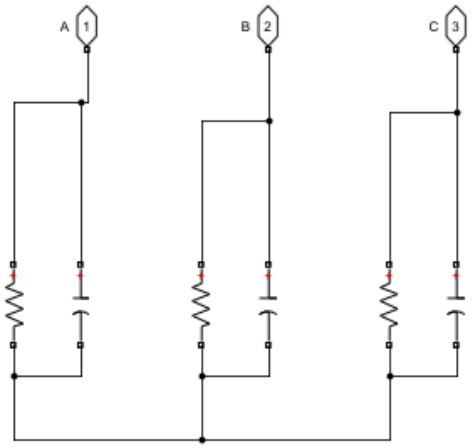


Figure 52. Load C scheme.

Load

The mask of the Load on the Simulink library is depicted in Figure 53. The parameters that the mask asks for are depicted in Figure 54. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_load: Table with the load parameters.
2. Init_load: Array with the initialization variables.
3. Num: Load number.

The scheme model is depicted in Figure 55.

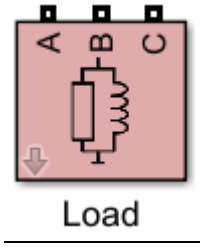


Figure 53. Load mask.

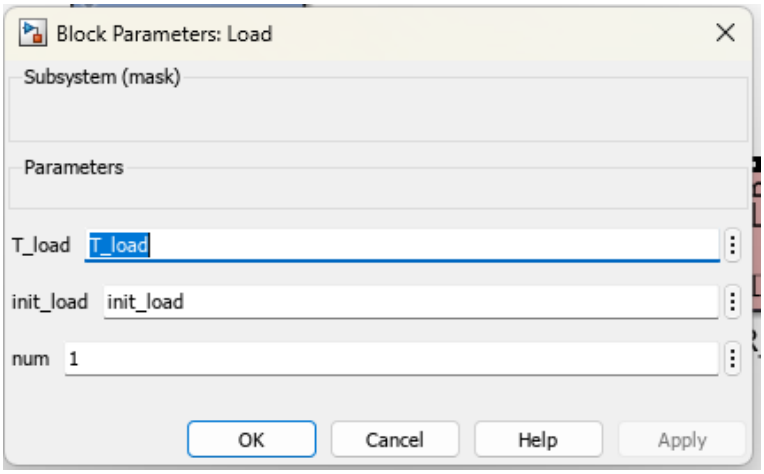


Figure 54. Load mask parameters.

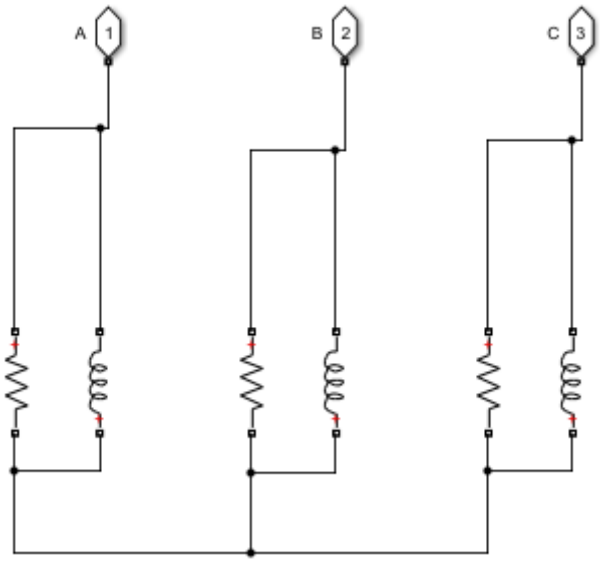


Figure 55. Load scheme.

Load R

The mask of the Load R on the Simulink library is depicted in Figure 56. The parameters that the mask asks for are depicted in Figure 57. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_load: Table with the load parameters.
2. Num: Load number.

The scheme model is depicted in Figure 58.

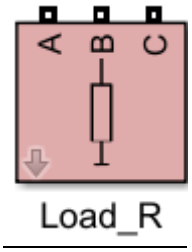


Figure 56. Load R mask.

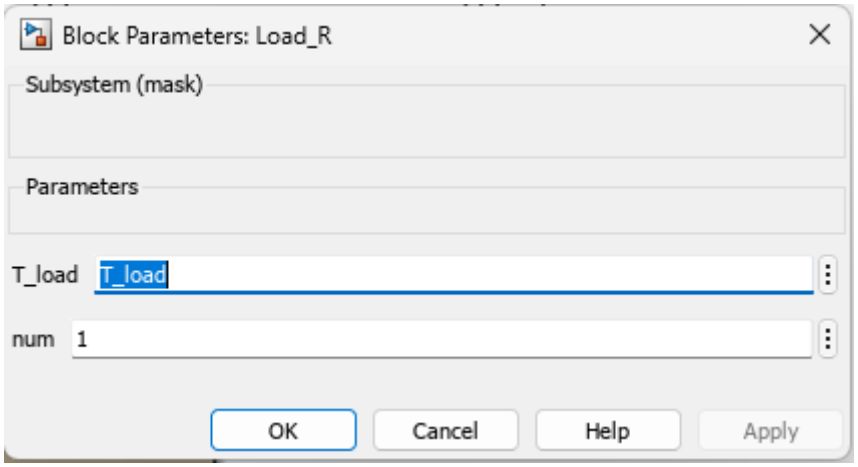


Figure 57. Load R mask parameters.

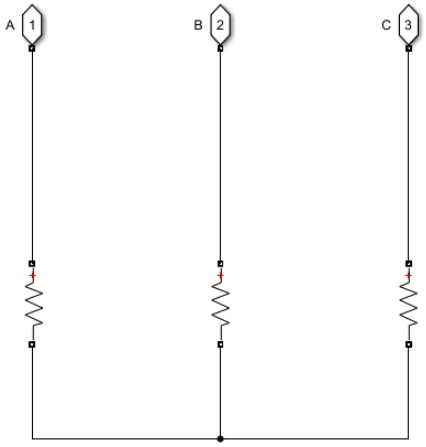


Figure 58. Load R scheme.

RLC

The mask of the shunt element RLC on the Simulink library is depicted in Figure 59. The parameters that the mask asks for are depicted in Figure 60. The tool automatically sets these parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_shunt: Table with the shunt parameters.

2. Init_shunt: Array with the initialization variables.
3. Num: Shunt element number.

The scheme model is depicted in Figure 61.

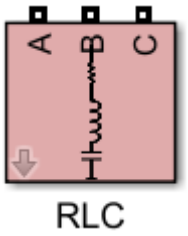


Figure 59. RLC mask.

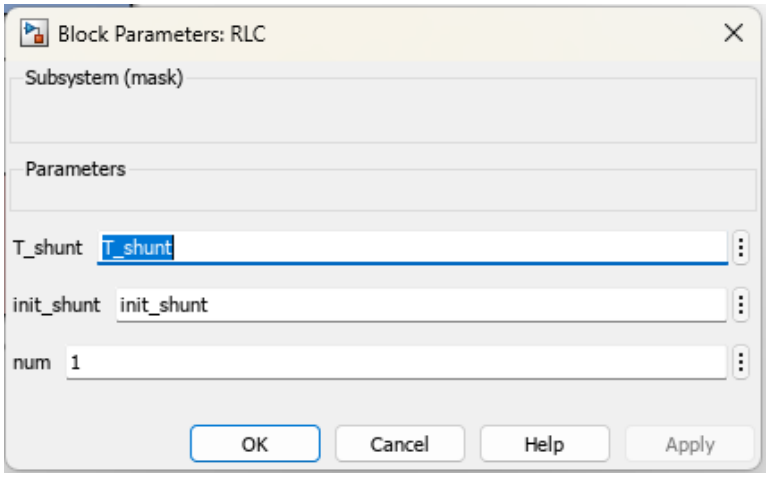


Figure 60. RLC mask parameters.

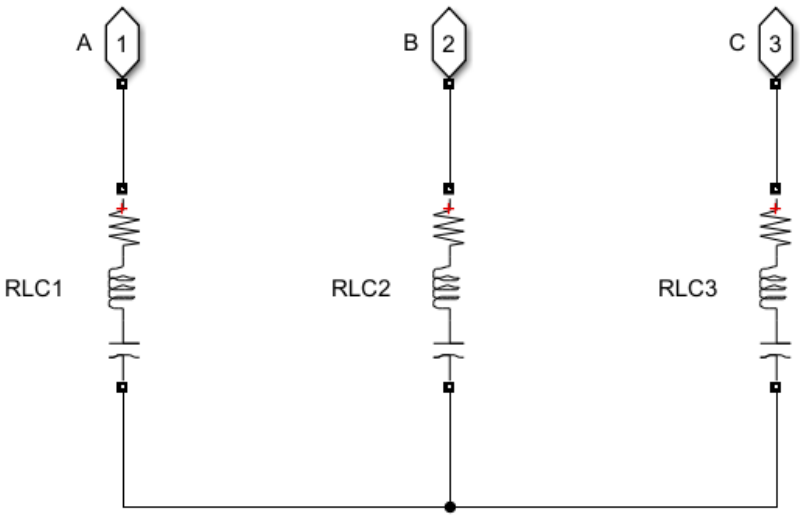


Figure 61. RLC scheme.

C-Type

The mask of the shunt element RLC on the Simulink library is depicted in Figure 50. The parameters that the mask asks for are depicted in Figure 51. The tool automatically sets these

parameters; the user must not change this. This is just displayed for information. The parameters that the mask asks are the following:

1. T_shunt: Table with the shunt parameters.
2. Init_shunt: Array with the initialization variables.
3. Num: Shunt element number.

The scheme model is depicted in Figure 52.

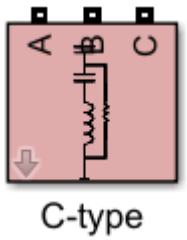


Figure 62. C-type mask.

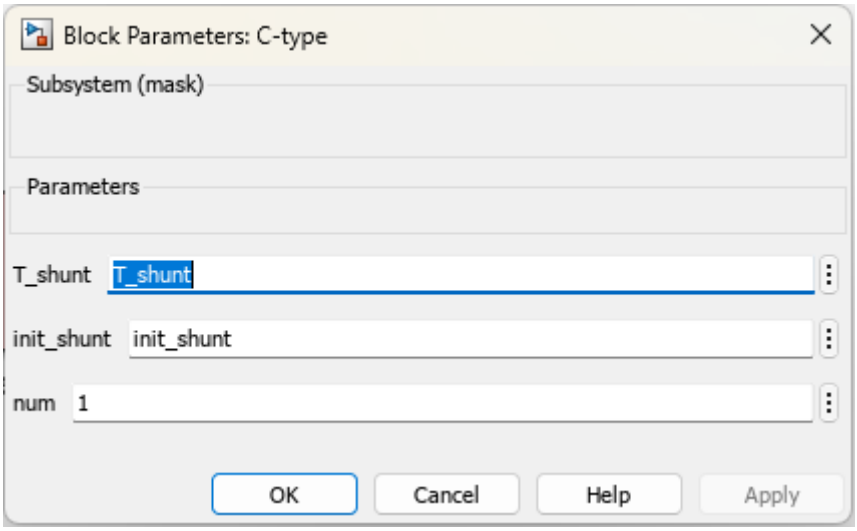


Figure 63. C-type mask parameters.

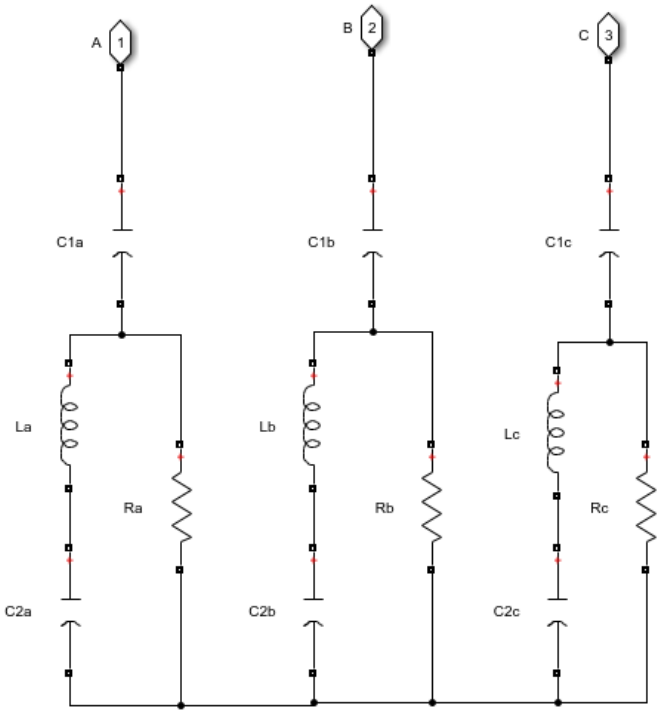


Figure 64. C-type scheme.

Synchronous Generator

In this section, the synchronous generator model is explained. The main component of the synchronous generator is depicted in Figure 65 and described as follows:

1. Electrical machine: It is the fundamental block of the synchronous machine and corresponds to the stator and rotor windings (including field and damper windings).
2. Exciter: The exciter is responsible for regulating the terminal voltage. It controls the voltage applied to the field winding in order to modify the field current.
3. Governor and turbine: The role of the governor is to provide the necessary mechanical power reference to the turbine by measuring the speed of the generator. Then the turbine provides the mechanical power to the synchronous generator shaft. The shaft can be represented with a single-mass or a multi-mass model.

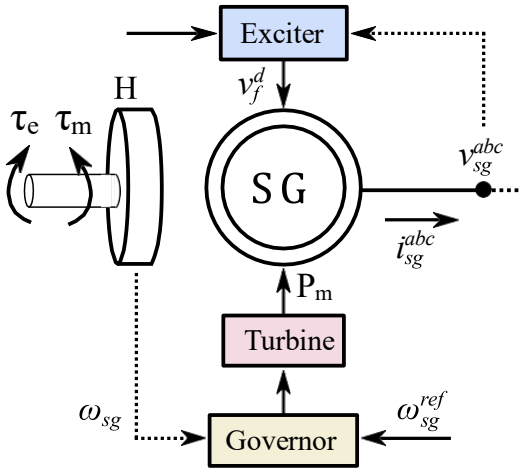


Figure 65. Scheme of the synchronous generator model.

The equivalent circuit for the machine electrical and mechanical model can be seen in Figure 66. A round rotor machine is considered, including the stator winding (q and d axis), the field winding (d-axis), and the damper windings (two in the q axis and one in the d axis).

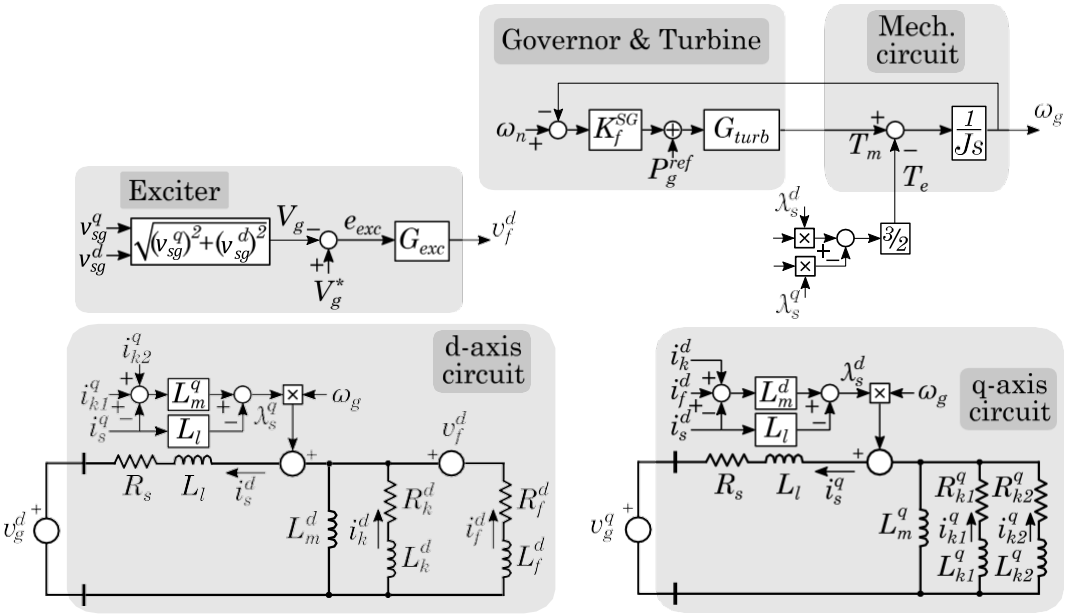


Figure 66. Synchronous machine electrical and mechanical model.

Two-level voltage source converter

In this section, the two-level voltage source converter model is explained. The converter is modelled following the average model. No commutation block is added. Thus, the converter applied voltage can be modelled as three controlled voltage source where the applied voltage comes from the output of the converter control. The converter is equipped with an LCL filter. Different control schemes can be applied. For now, STAMP considers 4 different controls:

1. Grid-following: This control considers a current loop with cascaded active and reactive power controls. The converter is synchronized with the grid by means of a phase-locked loop.
2. Grid-forming: This control considers a current loop with cascaded voltage vector control. The converter is synchronized with the grid by means of a P-f droop.
3. STATCOM: This control scheme is similar to the grid-following one but incorporates the DC side dynamics. It also includes the possibility to add a power oscillation damping controller.
4. WT: This control models a wind turbine model.

Modular Multilevel Converter

In this section, the modular multilevel converter model is explained. The converter is modelled following an average model. No commutation block is added. Moreover, it is considered that the MMC is controlled following a compensated modulation. Therefore, the model can be split between the AC and DC sides, which can be represented as it is shown in Figure 67. A part from that six, condensers are used to simulate the energy stored in each converter arm.

Different control strategies can be applied to the MMC. In this case, three different control strategies are considered:

1. AC-GFOL DC-GFOL: This control strategy includes a current control with a cascaded active and reactive power controls. The DC side regulates the MMC internal energy. A PLL is used to synchronize the converter with the grid.
2. AC-GFOL DC-GFOL: This control strategy includes a current control with a cascaded DC voltage control and reactive power control. The DC side regulates the MMC internal energy. A PLL is used to synchronize the converter with the grid.
3. AC-GFOR DC-GFOL: This control strategy includes a current control with a cascaded vector voltage control. The DC side regulates the MMC internal energy. A P-f droop is used to synchronize the converter with the grid.

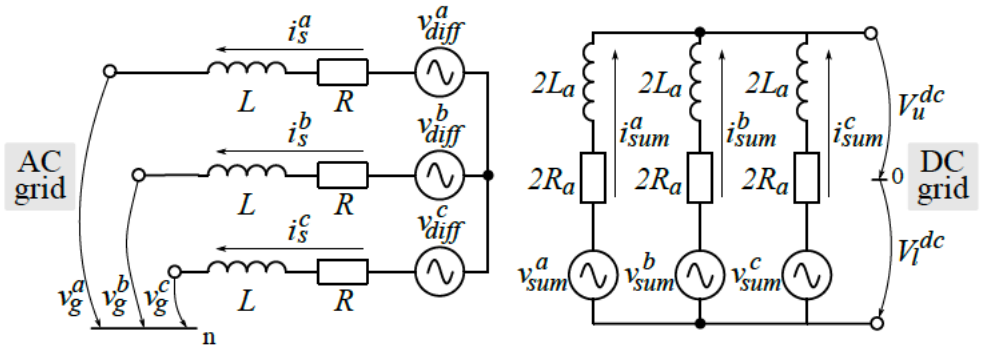


Figure 67. MMC electrical scheme.

Linearization point calculation

The linearization point calculation is a crucial step in computing the state-space model. For each element, a function is created to calculate the linearization point. It calculates the variables necessary to perform the state-space system of elements that are not linear and have to be linearized. Thus, this is not needed for the elements that are already linear. From the previously presented elements (see Simulink non-linear in-built elements), the elements that need this linearization are: the synchronous generator, the two-level voltage source converters, and the modular multilevel converters.

Synchronous Generator

The function in charge of calculating the linearization point variables for the Synchronous generator state-space model is:

Lp_sg = generate_linearization_point_SG(T_SG, T_global, delta_slk_ll)

It requires the following inputs:

1. T_SG: Table with the synchronous generator parameters.
2. T_global: Table with the base global parameters.
3. Delta_slk_ll: Vector with the delta_slk angles values.

The function gives as output an array named lp_sg which stores the following linearization point variables:

1. Isq0: q-component stator windings current in machine per unit system.
2. Isd0: d-component stator winding currents in machine per unit system.

3. Ikd0: d-component damper winding in machine per unit system.
4. Ikq10: q-component damper winding in machine per unit system.
5. Ikq20: q-component damper winding in machine per unit system.
6. Vq0: q-component voltage after the transformer in grid per unit system.
7. Vd0: d-component voltage after the transformer in grid per unit system.
8. Vsg_q0: q-component voltage after the transformer in machine per unit system.
9. Vsg_d0: d-component voltage after the transformer in machine per unit system.
10. Vq_bus0: q-component grid voltage in machine per unit system.
11. Vd_bus0: d-component grid voltage after the transformer in machine per unit system.
12. W0_pu: per unit frequency.
13. W0: frequency
14. Etheta0: machine angle deviation respect to grid angle.
15. Pm0: mechanical power.
16. Efd0: field voltage.
17. Ifd0: d-component excitation field current in machine per unit system.

Two-level voltage source converter

The function in charge of calculating the linearization point variables for the two-level voltage source converter state-space model is:

Lp_VSC = generate_linearization_point_VSC(T_VSC, T_global, delta_slk_ll)

It requires the following inputs:

1. T_VSC: Table with the two-level voltage source converter parameters.
2. T_global: Table with the base global parameters.
3. Delta_slk_ll: Vector with the delta_slk angles values.

The function gives as output an array named lp_VSC which stores the following linearization point variables:

1. Ig_q0: q-component grid current in grid angle base.
2. Ig_d0: d-component grid current in grid angle base.
3. Ig_qc0: q-component grid current in the converter angle base.
4. Ig_dc0: d-component grid current in the converter angle base.
5. Is_q0: q-component filter current in the grid angle base.
6. Is_d0: d-component filter current in the grid angle base.
7. Is_qc0: q-component filter current in the converter angle base.
8. Is_dc0: d-component filter current in the converter angle base.
9. U_q0: q-component filter voltage in the grid angle base.
10. U_d0: d-component filter voltage in the grid angle base.
11. U_qc0: q-component filter voltage in the converter angle base.
12. U_dc0: d-component filter voltage in the converter angle base.
13. Ucap_q0: q-component capacitor filter voltage in the grid angle base.
14. Ucap_d0: d-component capacitor filter voltage in the grid angle base.
15. Vc_qc0: q-component converter applied voltage in the converter angle base.
16. Vc_dc0: d-component converter applied voltage in the converter angle base.
17. Vg_q0: q-component grid voltage in the grid angle base.
18. Vg_d0: d-component grid voltage in the grid angle base.
19. W0_pu: grid frequency in the per unit.

20. W0: grid frequency in real values.
21. Etheta0: converter angle deviation respect to grid angle.

Modular Multilevel Converter

The function in charge of calculating the linearization point variables for the modular multilevel converter state-space model is:

Lp_ipc = generate_linearization_point_IPC(T_IPC, T_global, delta_slk_ll)

It requires the following inputs:

1. T_IPC: Table with the modular multilevel converter parameters.
2. T_global: Table with the base global parameters.
3. Delta_slk_ll: Vector with the delta_slk angles values.

The function gives as output an array named lp_IPC which stores the following linearization point variables:

1. Vnq0: q-component grid voltage in the grid angle base.
2. Vnd0: d-component grid voltage in the grid angle base.
3. Vnq0_c: q-component grid voltage in the converter angle base.
4. Vnd0_c: d-component grid-voltage in the converter angle base.
5. Idiffq0: q-component AC filter current in the grid angle base.
6. Idiffd0: d-component AC filter current in the grid angle base.
7. Idiffq0_c: q-component AC filter current in the converter angle base.
8. Idiffd0_c: d-component AC filter current in the converter angle base.
9. Vdiffq0: q-component AC voltage applied by the converter in grid angle base.
10. Vdiffd0: d-component AC voltage applied by the converter in grid angle base.
11. Vdiffq0_c: q-component AC voltage applied by the converter in converter angle base.
12. Vdiffd0_c: d-component AC voltage applied by the converter in converter angle base.
13. Isum: Sum circuit current.
14. Vsum: Sum circuit voltage.
15. Vdc: DC voltage at converter terminals.
16. Theta: converter angle deviation respect to grid angle.

Initialization point calculation

Electromagnetic transient simulation models can be very time-consuming. The simulation time depends on a lot of factors. However, the main ones are the solver used, the solver step size, and the system size. The nature of the system usually imposes these factors, and therefore, these are difficult to change as it is possible to lose model accuracy if these are changed.
In the majority of the studies, the event that is to be studied is within a few seconds. However, if the model is not properly initialized, the initialization can take longer than the event itself. Moreover, in some scenarios, the initialization can lead the system to instability. Thus, the proper initialization of the model is highly important and can save time.
This section documents the functions used to initialize each of the elements presented in Simulink non-linear in-built elements.

PI-section line

The function in charge of calculating the initialization point variables for the pi-section line is:

Ini_line = generate_initialization_line(T_NET, results)

It requires the following inputs:

1. T_NET: Table with the line parameters.
2. results: structure with the power flow solution.

The function gives as output an array named *ini_line* which stores the following initialization point variables:

1. Va1: a-component bus 1 voltage.
2. Vb1: b-component bus 1 voltage.
3. Vc1: c-component bus 1 voltage.
4. Va2: a-component bus 2 voltage.
5. Vb2: b-component bus 2 voltage.
6. Vc2: c-component bus 2 voltage.
7. Ia: a-component current flowing from bus 1 to bus 2.
8. Ib: b-component current flowing from bus 1 to bus 2.
9. Ic: c-component current flowing from bus 1 to bus 2.

HVDC line

The function in charge of calculating the initialization point variables for the HVDC line is:

Ini_DCline = generate_initialization_DCline(T_NET, results)

It requires the following inputs:

1. T_DC_NET: Table with the DC line parameters.
2. results: structure with the power flow solution.

The function gives as output an array named *ini_DCline* which stores the following initialization point variables:

1. V1: bus 1 voltage.
2. V2: bus 2 voltage.
3. Ia: branch a current.
4. Ib: branch b current.
5. Ic: branch c current.

Load

The function in charge of calculating the initialization point variables for the loads is:

Ini_load = generate_initialization_load(T_Load, results)

It requires the following inputs:

1. T_Load: Table with the load parameters.
2. results: structure with the power flow solution.

The function gives as output an array named *ini_load* which stores the following initialization point variables:

1. Va: a-component voltage at the connection bus.
2. Vb: b-component voltage at the connection bus.
3. Vc: c-component voltage at the connection bus.

4. Ia: a-component current absorbed by the load.
5. Ib: b-component current absorbed by the load.
6. Ic: c-component current absorbed by the load.

Shunt elements

The function in charge of calculating the initialization point variables for the shunt elements is:

Ini_shunt = generate_initialization_Shunt(T_shunt, results)

It requires the following inputs:

1. T_shunt: Table with the shunt element parameters.
2. results: structure with the power flow solution.

The function gives as output an array named *ini_shunt* which stores the following initialization point variables depending on the type of shunt element:

1. RLC:
    a. iL_a: a-component inductance current absorbed by the shunt element.
    b. iL_b: b-component inductance current absorbed by the shunt element.
    c. iL_c: c-component inductance current absorbed by the shunt element.
    d. vC_a: a-component capacitor voltage.
    e. vC_b: b-component capacitor voltage.
    f. vC_c: c-component capacitor voltage.
2. C-type:
    a. iL_a: a-component inductance current absorbed by the shunt element.
    b. iL_b: b-component inductance current absorbed by the shunt element.
    c. iL_c: c-component inductance current absorbed by the shunt element.
    d. vC1_a: a-component capacitor 1 voltage.
    e. vC1_b: b-component capacitor 1 voltage.
    f. vC1_c: c-component capacitor 1 voltage.
    g. vC2_a: a-component capacitor 2 voltage.
    h. vC2_b: b-component capacitor 2 voltage.
    i. vC2_c: c-component capacitor 2 voltage.

Synchronous Generator

The function in charge of calculating the initialization point variables for the synchronous generator is:

[Init_SG initMachineBlocks] = generate_initialization_SG(T_SG, results)

It requires the following inputs:

1. T_SG: Table with the synchronous generator parameters.
2. results: structure with the power flow solution.

The function gives as output two arrays:

1. init_SG: stores the initialization values outside the synchronous machine. This includes the exciter, governor, turbine values and some electrical values outside the machine.
2. initMachineBlocks: stores the initialization values to initialize the machine.

Two-level voltage source converter

The function in charge of calculating the initialization point variables for the voltage source converter is:

Init_VSC = generate_initialization_VSC(T_VSC, results)

It requires the following inputs:

1. T_VSC: Table with the voltage source converter parameters.
2. Results: structure with the power flow solution.

The function gives as output an array named *init_VSC* which stores the necessary variables to initialize the converter. This includes electrical variables and control states. The electrical variables are common to all the converter control modes. However, the control states depend on the control mode. Next, the electrical variables are specified.

1. Udiff_mag: converter applied voltage magnitude.
2. Angle_vdiff: converter applied voltage angle.
3. Vdiffq = q-componenet converter applied voltage in converter angle reference frame.
4. Vdiffd = d-componenet converter applied voltage in converter angle reference frame.
5. Isa: a-component filter current flowing outside the converter.
6. Isb: b-component filter current flowing outside the converter.
7. Isc: c-component filter current flowing outside the converter.
8. Idiffq: q-component filter current flowing outside the converter in converter angle reference frame.
9. Idiffd: d-component filter current flowing outside the converter in converter angle reference frame.
10. Iga: a-component grid current flowing outside the converter.
11. Igb: b-component grid current flowing outside the converter.
12. Igc: c-component grid current flowing outside the converter.
13. Uq: q-component voltage after the transformer in converter angle reference frame.
14. Ud: d-component voltage after the transformer in converter angle reference frame.
15. Igq: q-component grid current flowing outside the converter in angle reference frame.
16. Igd: d-component grid current flowing outside the converter in angle reference frame.
17. Ucap_a: a-component filter capacitor voltage.
18. Ucap_b: b-component filter capacitor voltage.
19. Ucap_c: c-component filter capacitor voltage

Modular Multilevel Converter

The function in charge of calculating the initialization point variables for the modular multilevel converter is:

ini_mmc = generate_initialization_MMCs(T_MMC, results)

It requires the following inputs:

1. T_MMC: Table with the modular multilevel converter parameters.
2. Results: structure with the power flow solution.

The function gives as output an array named *init_MMC* which stores the necessary variables to initialize the converter. This includes electrical variables and control states. The electrical

variables are common to all the converter control modes. However, the control states depend on the control mode. Next, the electrical variables are specified.

1. Udiff_mag: AC voltage magnitude applied by the converter.
2. Angle_vdiff: AC voltage angle applied by the converter.
3. Vdiffq: q-component ac voltage applied by the converter in the converter angle reference frame.
4. Vdiffd: d-component ac voltage applied by the converter in the converter angle reference frame.
5. Isa: a-component of the AC filter current flowing outside the converter.
6. Isb: b-component of the AC filter current flowing outside the converter.
7. Isc: c-component of the AC filter current flowing outside the converter.
8. Idiffq: q-component of the AC filter current flowing outside the converter in the converter angle reference frame.
9. Idiffd: d-component of the AC filter current flowing outside the converter in the converter angle reference frame.
10. Uq0: q-component AC grid voltage in the converter angle reference frame.
11. Ud0: d-component AC grid voltage in the converter angle reference frame.
12. Isum0_DC: converter inner current.
13. Vdc: DC voltage at the converter terminal.
14. Usuma: a-component of the applied voltage at the DC side.
15. Usumb: b-component of the applied voltage at the DC side.
16. Usumc: c-component of the applied voltage at the DC side.

User manual to incorporate user-made elements

This section describes the necessary steps to incorporate a new element to STAMP. The final incorporation to STAMP will be done by the coordinators of this project, i.e., the authors of this manual. The incorporation of the new element can be done after pulling up a request. Next the necessary steps are presented. In this STAMP version this is only considered for AC elements, i.e., it is not considered for elements that interconnect AC and DC grids such as interconnecting power converters.

Non-linear model

The user must define a new mask-element in the library with the name “user”. This mask must have 3 different input variables:

1. T_user: A table containing the necessary parameters for the new element. This table is read inside the mask an all the element parameters are taken from here. This includes electrical parameters and control parameters.
2. Init_user: An array containing the necessary data to initialize the new element. This is read inside the mask to initialize all the elements. This includes electrical parameters and control states.
3. Num: the number of the new element.

Initialization point

The user must define a new function named “initialization_user” which will output “Init_user” the array that will be inputted to the non-linear model mask. This function should have the following structure:

Init_user = initialization_user(T_user, results)

The input of this function must be:

1. T_user: A table containing the necessary parameters for the new element. The same table included in the non-linear model mask.
2. Results. An array containing the results of the power flow data (See Power flow results).

The output of the function must be:

1. Init_user: An array containing the necessary data to initialize the new element. The same array included in the non-linear model mask.

Linearization point

The user must define a new function named “linearization_user” which will output “lp_user” the array that will be inputted to the calculation of the state-space system model. This function should have the following structure:

lp_user = linearization_user(T_user, results)

The input of this function must be:

1. T_user: A table containing the necessary parameters for the new element. The same table included in the non-linear model mask.
2. Results. An array containing the results of the power flow data (See Power flow results).

The output of the function must be:

2. lp_user: An array containing the necessary variables to linearize the new element.

State-space model

The user must define a new function named “genereate_user” which will output the state-space model of the new element. This function should have the following structure:

l_blocks = generate_user(l_blocks, T_user, lp_user, T_global, REF_w)

The input of this function must be:

1. l_blocks: an array of small-signal elements. This will store the new element at the end of this array.
3. T_user: A table containing the necessary parameters for the new element. The same table included in the non-linear model mask.
2. lp_user: The output of the function “linearization_user”.
3. T_global: A table containing the base of the grid.
4. REF_w: Is the name of the angle reference of the global qd0-frame (if needed).

The output of this function must be:

1. l_blocks: an array of small-signal elements. With the new element stored at the end of this array.

When defining the inputs and outputs of the new small-signal model the following considerations must be taken into account.

1. Inputs naming of the state-space model:
    a. At least three inputs must be considered:
        i. NET.vnXq: This is the q-component voltage of the grid in the point of connection where X is the bus number.
        ii. NET.vnXc: This is the c-component voltage of the grid in the point of connection where X is the bus number.
        iii. REF_w: This is the name of the frequency that is considered as the grid global frequency. Is used to change the qd0-frame from the grid reference to the element reference (if needed).
2. Outputs naming of the state-space model:
    a. At least two outputs must be considered:
        i. USERY.iq: This is the q-component current of the new element flowing to the grid in the point of connection where Y is the new element number.
        ii. USERY.iq: This is the d-component current of the new element flowing to the grid in the point of connection where Y is the new element number.