Transmission and Distribution Systems Co-Simulation Tool (TDcoSim)

User Manual

A project funded by
Office of Electricity
U.S. Department of Energy

TDcoSim Version 2.0.0
October 2021

Office of Electricity program sponsors:
Ali Ghassemian
Dan Ton

PI and lead developer:
Karthikeyan Balasubramaniam

Co-developers:
Sang-il Yim
Siby Jose Plathottam
Kumarsinh Jhala
Zhao, Shijia

Project repository:
https://github.com/tdcosim/TDcoSim

TDcoSim User Guide

Table of Contents

Introduction

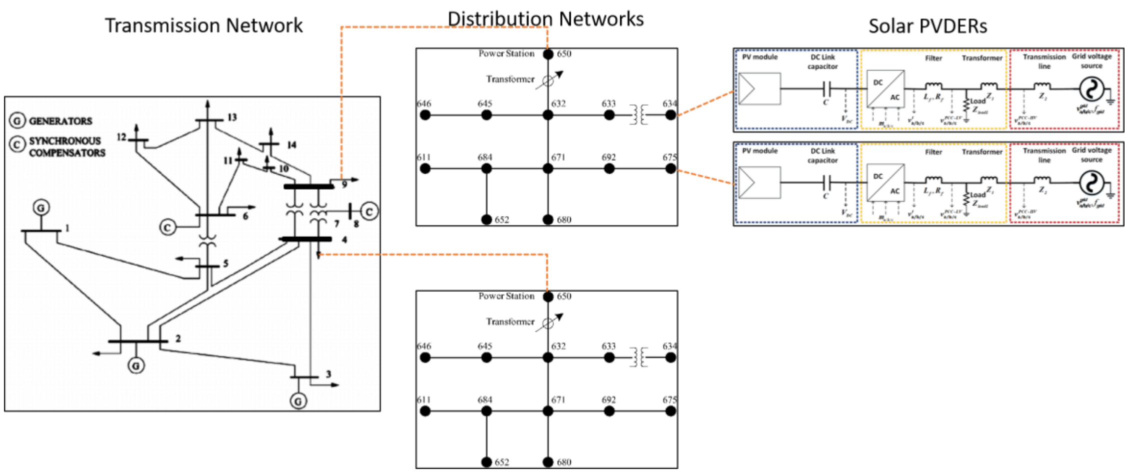
Argonne's transmission and distribution systems co-simulation tool (**TDcoSim**) is a tool to conduct studies that capture short- and long-term interactions between transmission and distribution systems with and without distributed energy resources (DER). It is capable of performing steady-state and dynamic simulations, as well as perfuming analytics on the results from the simulation . Consideration of inverter-based DER dynamics along with its protection and controls are among the most salient features of this tool. Additionally, the tool is capable of efficiently simulating tens of thousands of individual DER models. **TDcoSim** is designed to be used in offline planning, operational, and control studies. Transmission system entities can use the tool to study high-penetration DER scenarios, which will assist in ensuring secure, reliable, and economic grid planning and operations.

This manual intends to introduce users to **TDcoSim**, provide a step-by-step guide to its installation and use, and offer examples of its capabilities as a tool for conducting studies. A list of case studies possible with **TDcoSim** can be found here and a brochure with FAQ related to the project goals can found here.

What is TDcoSim?

TDcoSim is a software tool that can be used to perform static and dynamic co-simulation of transmission and distribution networks as well as DER. Currently, the tool has incorporated dynamic photovoltaic (PV) systems (referred to as PV-DER) for its dynamic simulation capabilities. Dynamic models of other inverter-based distributed generations (DG) (e.g. battery energy storage systems (BESS), wind turbines (WT)) will be included in future versions. Interested users can also integrate their own dynamic DER models into the tool.

Fig. 1. below illustrates the various components that currently can be simulated using **TDcoSim**. It features a representative IEEE 14-bus transmission system, IEEE 13-bus distribution systems, and solar PV-DERs. Please note that Solar PV-DER simulation-utility is used to simulate PV-DER's in dynamic T&D co-simulations. Default OpenDSS standard generator models for PV or other DER are used in steady-state T&D co-simulations.



14-bus transmission, 13-bus distribution network, and Solar PVDER

Fig. 1. Components that can be simulated using TDcoSim.

Note that **TDcoSim** can also simulate advanced models like DER_A, composite load model (CMLD), and complex load mode (CLOD) that available in PSS/E.

How can I use it?

TDcoSim is available as an open source Python package and can be installed at no cost from its GitHub repository . Additionally, the user needs to separately install PSS®E for simulating a transmission network, OpenDSS for simulating distribution networks, and Solar PV-DER simulation-utility for simulating dynamic PV-DERs.

Detailed installation instructions and links to the requisite supporting software can be found here.

What are the inputs?

In order to run a co-simulation using **TDcoSim**, the user needs to provide the following inputs:

- A Transmission system model in a format compatible with PSS/E (required)
- Distribution system models in a format compatible with OpenDSS (required)
- Simulation type - steady-state or dynamic (required)
- DER penetration levels (required for dynamic co-simulation)
- DER ratings and ride through settings (optional)
- Simulation events (optional)
- Presence of ZIP, Composite load model (CMLD), Composite load model (CLOD),
- Presence of aggregate DER model (DER_A) (only available in PSS/E 35)

Detailed description of each input can be found here.

What are the outputs?

TDcoSim provides following outputs from each component of the T&D co-simulation:

- Transmission bus: voltage, frequency, load active and reactive power consumption (if a bus is connected with a load), generator active and reactive power output (if a bus is connected with a generator).
- Distribution feeder node: voltage, active and reactive load, DER active and reactive power output (if a bus is connected with a DER).

Please note that the output comes in an interval of half-a-cycle for dynamic simulations. For steady-state simulations, the output comes at an interval corresponding to the time step of users' choice, which can range from seconds, minutes, hours, to years.

The output format is an Excel spreadsheet.

Types of studies

TDcoSim is intended to be used as a tool for studying static and dynamic impacts of distributed energy resources on the transmission system. Studies that can be conducted with the current version of the software are listed below:

- Steady-state studies
    1. Impact of DER on bulk power system load following or ramping requirements throughout the day and over the course of the seasons.
    2. Analyze voltage profile of both T-system and D-system with varying levels of DER penetrations.
    3. Impact of different DER penetration levels on the voltage stability of BES via continuations power flow analysis.
- Dynamic studies
    1. Impact of DER's tripping/ride-through settings on bulk power system stability (both frequency and voltage) during and post transmission system faults.
    2. Parameterization and performance verification of DER_A and composite load model.
    3. Impact of DER on the small-signal stability of bulk power system

---

Note: Examples of dynamic studies performed with TDcoSim are included in the **Examples** section.

Future development

The studies that can be conducted in the next version of the software are listed below:

- Dynamic studies

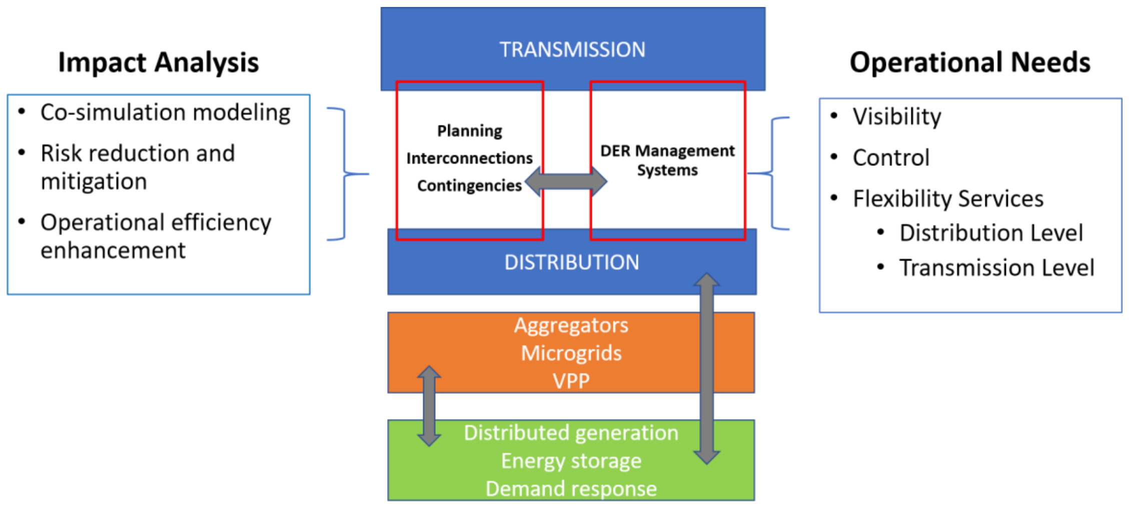
  1. Impact of cloud cover events on conventional synchronous generators and operations of bulk power system.
  2. Impact of DER dynamic reactive power support on bulk power system stability (both frequency and voltage) during and post transmission system faults.

  3. Impact of line outages on bulk power system operations and stability under high-DER-penetration scenarios.
  4. Impact of generator and/or load outages on system frequency under high DER penetration.
  5. Impact of sudden load increase/decrease on the stability of bulk power system.
- Protection studies

  1. Analyze impact of high DER penetration on coordination among distribution-system protection devices and DER protection relays, and other protection schemes such as under voltage load shedding (UVLS) and under frequency load shedding (UFLS) schemes.
  2. Determine (a) appropriate DER frequency and voltage ride-through settings; and (b) distribution system protection devices settings. These settings will help ensure bulk system reliability and also satisfy distribution-system safety requirements (e.g. prevention of unintentional islanding).
  3. Short-circuit analysis.
- T&D operations coordination studies

  1. Aggregation of DER is an effective approach to integrate DER as a dispatchable resource into the planning and operation of distribution and transmission systems. DERMS has emerged as an effective tool sitting between the transmission and distribution operators to manage the aggregation of DERs at the substation and feeder level.
  2. TDcoSim, in conjunction with DERMS, can be used to manage DERs to provide T&D coordinated congestion management, system balancing, frequency and voltage control, and flexibility.

The following figure illustrates how **TDcoSim** can assist in T&D planning and operations coordination for future high-DER-penetration grid.



T & D planning and operationsl

Following capabilities are planned in be added in the future:

- Capability to include other types of DERs such as energy battery storage system.
- Capability to introduce insolation change events.
- Capability to simulate generator tripping, line outages, load increase/decrease.
- Capability to consider distribution system UFLS and UVLS schemes.

Scalability and solution time

The scale of the T & D system (including PV-DERs) to be co-simulated is limited only by the available memory (RAM) in the workstation where **TDcoSim** is installed. The solution time for the dynamic co-simulation depends on the number of distribution feeder instances, the number and type of DER instances, the type of ODE solver being used, and on the number of logical cores available in the workstation.

Continue to Getting Started

Getting started with TDcoSim

In this section, we describe how you can get started with using **TDcoSim** to conduct static or dynamic co-simulation studies for T & D systems with different DER penetration levels and various events.

Setup TDcoSim

Please install the software per installation instructions as the first step (Installation instructions for can be found here). Make sure the system requirements are satisfied (System requirements can be found here).

Invoking TDcoSim from command prompt

Once TDcoSim has been successfully installed, we get access to the *tdcosimapp* from the command line. The *tdcosimapp* is to *TDcoSim* what *kubectl* is to *kubernetes*. The following functionalities are available through the *tdcosimapp*:

1. Describe the location of folders containing logs, example configurations, T system models, D system models etc.: **tdcosim describe**
2. Run test configurations: **tdcosim test**
3. Create configuration template: **tdcosim template –templatePath "/path/to/save/template.json"**
4. Validate user provided configuration, providing helpful hints to troubleshoot issues, if there are any.
5. Run the co-simulation: **tdcosim run -c "/path/to/config.json"**
6. Launch browser based dashboard to analyze the results: **tdcosim dashboard -o "/path/to/df-pickle.pkl"**
7. Provide information/help about any top level declaration used in configuration file: **tdcosim info –configHelp** *configtype* (**psseConfig, openDSSConfig, simulationConfig, outputConfig, logging**)

Note: The **tdcosimapp** can be invoked using the **tdcosim** command from the command line .

Steps involved in a co-simulation

1. Information required by TDcoSim

1. Specify version of PSS/E (either PSS/E 33 or PSS/E 35)

2. Specify parameters for the power system to be analyzed:

   – Transmission system
     - Transmission system model (e.g. IEEE 118 bus system)

        - Supported file formats:.*raw,* .dyr
    - Buses where distribution system models are attached
    - Type of load model (static,ZIP,CLOD,CMLD).
    - Presence of DER_A model
  - Distribution system
    - Distribution system model (e.g. IEEE 123 node feeder)
        - Supported file formats:*.dss

    - Solar PV penetration level (fraction of the distribution system load)
    - Scaling factor for power output from single DER model instance
  - DER parameters (optional)
    - Type of DER model (fast DER or detailed DER)
    - DER voltage and power ratings (e.g. 50 kW, 175 V)
    - DER interconnection standard during voltage anomaly (eg. IEEE 1547 Category II)
    - Settings specific to detailed DER:
        - DER configuration file path
        - DER configuration ID (e.g. '50') -> only for detailed
        - Type of ODE solver

3. Specify whether simulation is static or dynamic.

4. Specify length of simulation (e.g. 5 s).

5. Specify transmission bus fault events (optional).

  - Start and end time of fault (for e.g. 0.5 s, 0.667 s)
  - Bus at which fault occurs and fault impedance value.

6. Specify directory for logging, and results.

Note: Frequency ride through will be included in future version.

Note: Other disturbance events like loss of generator, line trip etc. will be included in future version.

---

2. Configure T & D & DER models and simulation scenarios

The power system models and simulation scenarios defined in the previous section can be transferred to TDcoSim using the **TDcoSim config** file (detailed explanations for every entry in the **TDcoSim config** file is provided here).

Defining the TDcoSim config file

The **TDcoSim config** file follows a specific template, and any deviations from the template will result in an error. The most streamlined way to make sure that the template is followed, is generate a sample config file and then manually populate the fields in this file. An example command for generating a sample config file for a dynamic co-simulation is given below:

```
tdcosim template --templatePath config_dynamic_example.json --simType dynamic
```

A detailed description of the configuration template is provided in the Using the configuration template chapter. Note:*** The **config** file can be edited with Notepad++.

Note: The **config** file can be in any folder on the user machine.

---

3. Running the co-simulation

Once the **config** file has been populated with the required entries, the user can start the co-simulation through the run command on command line as shown below:

```
tdcosim run -c "\path\to\config\config_dynamic_example.json"
```

The user will see a progress bar similar to the one shown below:

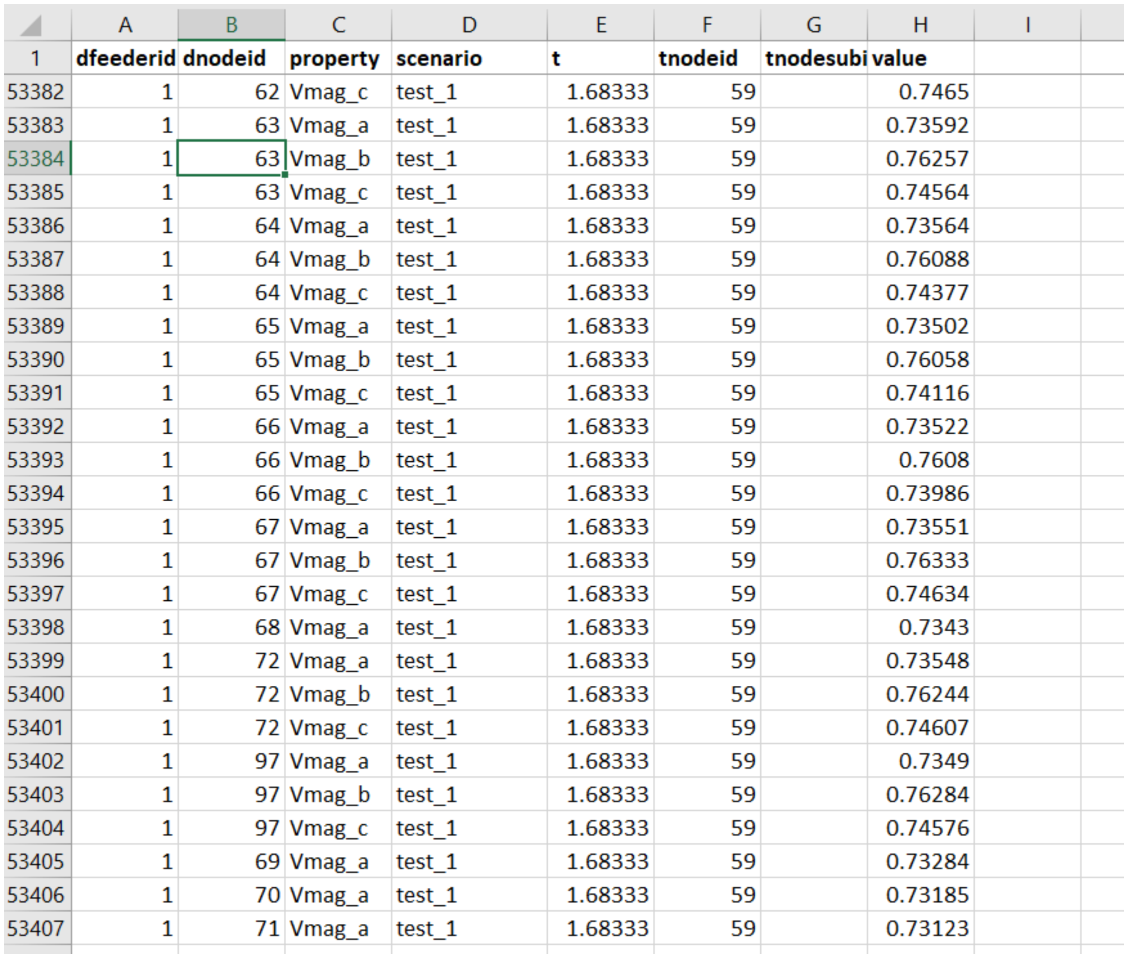
```
             INITIATED ON WED, JUN 02 2021  14:54
Simulation Progress : ====================>
100.04000000000013%(0.5002000000000006s/0.5s)
Solution time: 9.066482067108154
```

Note: Logs generated during co-simulation are can be found in the **install* folder.

---

4. Accessing the results

Outputs (from both transmission and distribution systems) are saved in the following formats within the user specified output folder at the end of the co-simulation: 1. PSS/E channel output file (**.out**) for containing all the simulated quantities from PSS/E. 2. A pickle (**df_pickle.pkl**) file containing the values of co-simulation variables from both PSS/E and OpenDSS. The co-simulated variables are stored as a data frame (as shown in Fig. 1). More information on the fields within the data frame is provided in TDcoSim Data Visualization and Analytics. 3. An **options.jSON** file containing the configuration parameters for the co-simulation. 4. A **psse_progress_output.txt** file containing logging information from PSS/E.



	A	B	C	D	E	F	G	H	I
1	dfeederid	dnodeid	property	scenario	t	tnodeid	tnodesubi	value	
53382	1	62	Vmag_c	test_1	1.68333	59		0.7465	
53383	1	63	Vmag_a	test_1	1.68333	59		0.73592	
53384	1	63	Vmag_b	test_1	1.68333	59		0.76257	
53385	1	63	Vmag_c	test_1	1.68333	59		0.74564	
53386	1	64	Vmag_a	test_1	1.68333	59		0.73564	
53387	1	64	Vmag_b	test_1	1.68333	59		0.76088	
53388	1	64	Vmag_c	test_1	1.68333	59		0.74377	
53389	1	65	Vmag_a	test_1	1.68333	59		0.73502	
53390	1	65	Vmag_b	test_1	1.68333	59		0.76058	
53391	1	65	Vmag_c	test_1	1.68333	59		0.74116	
53392	1	66	Vmag_a	test_1	1.68333	59		0.73522	
53393	1	66	Vmag_b	test_1	1.68333	59		0.7608	
53394	1	66	Vmag_c	test_1	1.68333	59		0.73986	
53395	1	67	Vmag_a	test_1	1.68333	59		0.73551	
53396	1	67	Vmag_b	test_1	1.68333	59		0.76333	
53397	1	67	Vmag_c	test_1	1.68333	59		0.74634	
53398	1	68	Vmag_a	test_1	1.68333	59		0.7343	
53399	1	72	Vmag_a	test_1	1.68333	59		0.73548	
53400	1	72	Vmag_b	test_1	1.68333	59		0.76244	
53401	1	72	Vmag_c	test_1	1.68333	59		0.74607	
53402	1	97	Vmag_a	test_1	1.68333	59		0.7349	
53403	1	97	Vmag_b	test_1	1.68333	59		0.76284	
53404	1	97	Vmag_c	test_1	1.68333	59		0.74576	
53405	1	69	Vmag_a	test_1	1.68333	59		0.73284	
53406	1	70	Vmag_a	test_1	1.68333	59		0.73185	
53407	1	71	Vmag_a	test_1	1.68333	59		0.73123	

report example

Fig. 1. Dataframe generated after Dynamic T&D co-simuation.

Note: The **.pkl**, **.JSON**, and **.out** files will be found in the folder specified by the user through the **output** field in the config file.

Data visualization and analytics

Modules for performing visualization and analytics are available as part of the TDcoSim package. The simplest way to visualize the data is using the **TDcoSim Dashboard** which may be launched using the command shown below.

```
tdcosim dashboard -o "path/to/results.pkl"
```

Detailed intructions on the **TDcoSim Dashboard** and other analytic tools are included in the TDcoSim Data Visualization and Analytics chapter.

Examples

Example **TDcoSim config** files for static and dynamic co-simulation scenarios are available in **install* folder.

Using the configuration template

TDcoSim is a declarative tool, where the user defines specification for the co-simulation. The specifications are given through a JSON configuration file. ## Create Configuration Template Example configuration files which can be used as a template can be generated using the **tdcosimapp**.

```
tdcosim template --templatePath config.json --simType static
```

Let us go over the above command. **template** specifies that the user is interested in using tdcosimapp to create configuration template. Next, **–templatePath config.json** specifies that the user wants to store the created configuration in the current working directory under the name config.json. Finally, the type of configuration is specified using **–simType static**.

Configuration template for a dynamic co-simulation using fast DER:

```
tdcosim template --templatePath config.json --simType dynamic
```

Configuration template for a dynamic co-simulation using detailed DER:

```
tdcosim template --templatePath config.json --simType dynamic_detailed_der
```

Configuration File Help

```
tdcosim template --configHelp outputConfig
```

This will show the following information,

```
outputConfig
------------
Output/results setup

Check:
outputConfig.simID
outputConfig.type
outputConfig.outputDir
outputConfig.outputfilename
outputConfig.scenarioID
```

In case one wants to know more information about outputConfig.type,

```
tdcosim template --configHelp outputConfig.type
```

which will result in,

```
outputConfig.type
-----------------

Type of output. Default is dataframe.
```

Using Data Visualization and Analytics

During the co-simulation, TDcoSim collects and stores a lot of data from the transmission system, distribution system, and DER models. The amount of data may make it challenging to the user to make effective use of it. We have implemented the following three features to improve the user experience with using the co-simulation data: 1. Store data using a TDcoSim DataFrame. 2. Provide a visualization dashboard 3. Provide data analytics module.

TDcoSim DataFrame

TDcoSim uses a custom data frame known as TDcoSim DataFrame to store and manage the data from transmission, distribution, and DER simulations. Not that this data frame is not in a time series format. Each entry in the data frame describes the attributes of a co-simulation quantity. Each entry is represented by a row, and each attribute is represented by a column. The attributes are as follows:

1. *scenario*: It specifies the co-simulation scenario from which the value was generated. Note that this can be specified through the **scenarioID** in the config file.
2. *tnodeid*: It specifies the transmission system bus which was the source of the value.
3. *dnodeid*: It specifies the distribution system node which was the source of the value. Note that this will only have a value if the entry is a distribution system quantity.
4. *t*: It specifies the time stamp of the co-simulation quantity.
5. *property*: It specifies the type of co-simulation quantity. Some of the valid values are: VOLT, ANGL, POWR, VARS
6. *value*: It specifies the numerical value of the co-simulation quantity.

Both the visualization and data analytics features available in TDcoSim uses the TDcoSim DataFrame as the underlying data structure.

Visualization

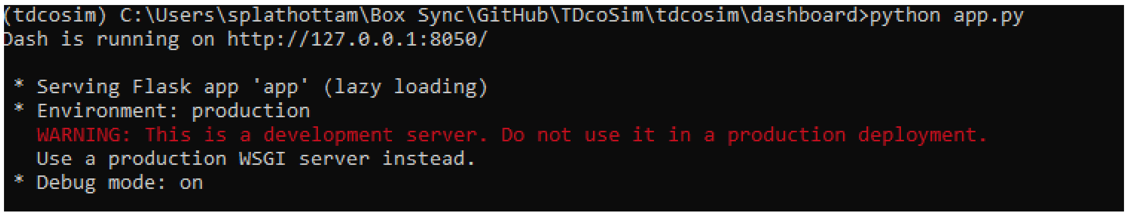
The visualization module provides an simple way to quickly visualize the co-simulation data without having to write code. Visualization is done through a browser based dashboard built using the Dash framework. The plots within the visualization were created using Plotly.

Using the visualization dashboard

Before the dashboard can be used the location of the filecontaining the co-simulation results (in the TDcoSim DataFrame format) should be known. The dashboard can be launched using the following command on the command line interface. Note that **results.pkl** should be replaced with the pickle file containing the TDcoSim DataFrame.

```
tdcosim dashboard -o "path/to/results.pkl"
```

This will result in the following output:



```
(tdcosim) C:\Users\splathottam\Box Sync\GitHub\TDcoSim\tdcosim\dashboard>python app.py
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

report example

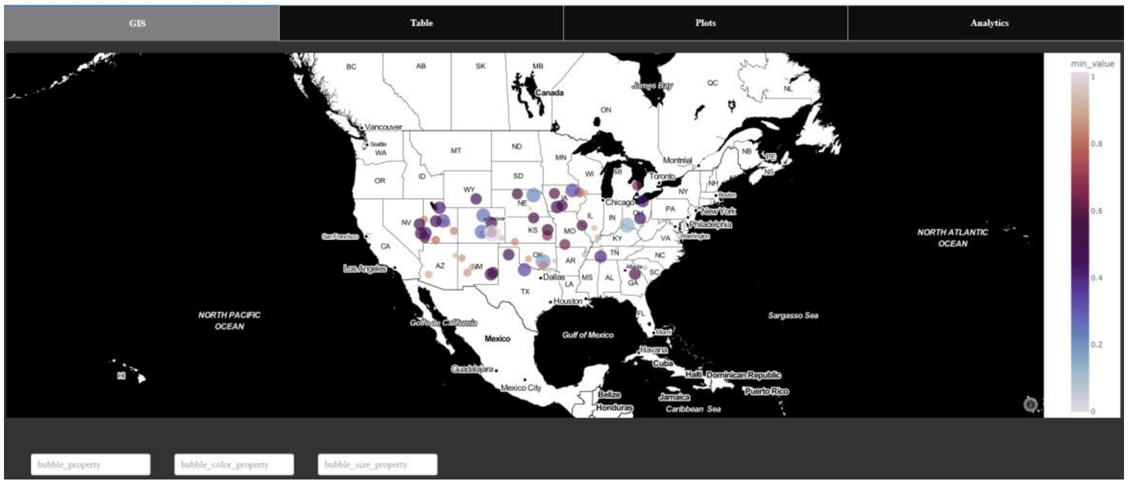
Fig. 1. Starting TDcoSim dashboard.

Copy and paste the web address (**http:/127.0.0.1:8050** in Fig. 1) into your browser. The dashboard will load after a few seconds. There are four tabs on the dashboard corresponding to four visualization capabilities, each of will be explained below.

Note: Using the dashboard only requires TDcoSim to be installed. It does not require PSS/E or OpenDSS to be installed. So co-simulation results that were generated in another machine can be copied to a machine without PSS/E or OpenDSS and then visualized.

GIS

This visualization overlays the all the nodes contained in the T system on an geographical map as bubble plots as shown in Fig. 2. The position of the bubbles are determined by latitude and longitude coordinates corresponding to each node in the T system.



report example

Fig. 2. GIS visualization on TDcoSim dashboard.

The information on the GIS can be further customized using the following fields:

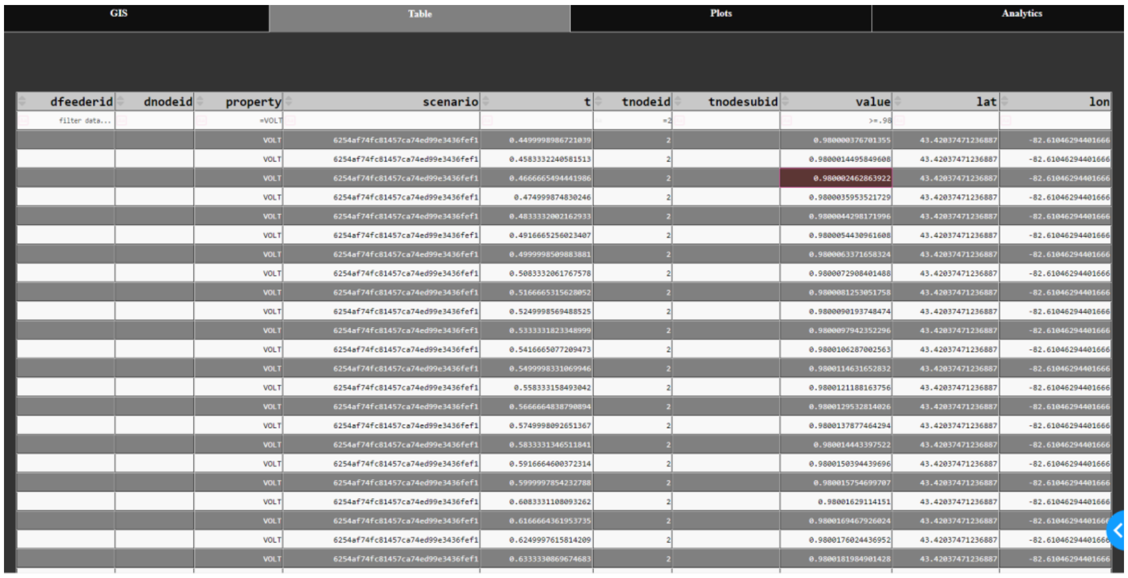
1. *bubble_property:* This selects the type of co-simulation variable that is being visualized by the bubbles. E.g. voltage, angle, speed.

2. *bubble_color_property:* This specifies the statistical property of the selected co-simulation variable that determines the color of the bubble. Note that the color map is included on the right. E.g. min,max, standard deviation.
3. *bubble_size_property:* This specifies the statistical property of the selected co-simulation variable that determines the size of the bubble. E.g. min,max, standard deviation.

Note: It is the responsibility of the user to supply the correct latitude and longitude coordinates corresponding to each T node. If no coordinates are supplied TDcoSim dashboard will assign random coordinates automatically.

Table

The Table visualization tab displays the entire data frame as an interactive Table as shown in Fig. 3. The table has column wise filtering capability using logical operators. For e.g. in Fig. 3. **property** column was used to filter out all the voltage values, the **tnodeid** column was used to filter out the values belonging to T node 2, and finally the **value** column was used to filter out the voltage values greater than or equal to 0.98.



dfeederid	dnodeid	property	scenario	t	tnodeid	tnodesubid	value	lat	lon
filter data...		=VOLT			=2		>=.98		
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4499998986721039	2		0.980000376701355	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4583332240581513	2		0.9800014495849608	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4666665494441986	2		0.980002462863922	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.474999874830246	2		0.9800035953521729	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4833332002162933	2		0.9800044298171996	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4916665256023407	2		0.9800054430961608	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.4999998509883881	2		0.9800063371658324	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5083332061767578	2		0.9800072908401488	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5166665315628052	2		0.9800081253051758	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5249998569488525	2		0.9800090193748474	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5333331823348999	2		0.9800097942352296	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5416665077209473	2		0.9800106287002563	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5499998331069946	2		0.9800114631652832	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.558333158493042	2		0.9800121188163756	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5666664838790894	2		0.9800129532814026	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5749998092651367	2		0.9800137877464294	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5833331346511841	2		0.980014443397522	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5916664600372314	2		0.9800150394439696	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.5999997854232788	2		0.980015754699707	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.6083331108093262	2		0.98001629114151	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.6166664361953735	2		0.9800169467926024	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.6249997615814209	2		0.9800176024436952	43.42037471236887	-82.61046294401666
		VOLT	6254af74fc81457ca74ed99e3436fef1	0.6333330869674683	2		0.9800181984901428	43.42037471236887	-82.61046294401666

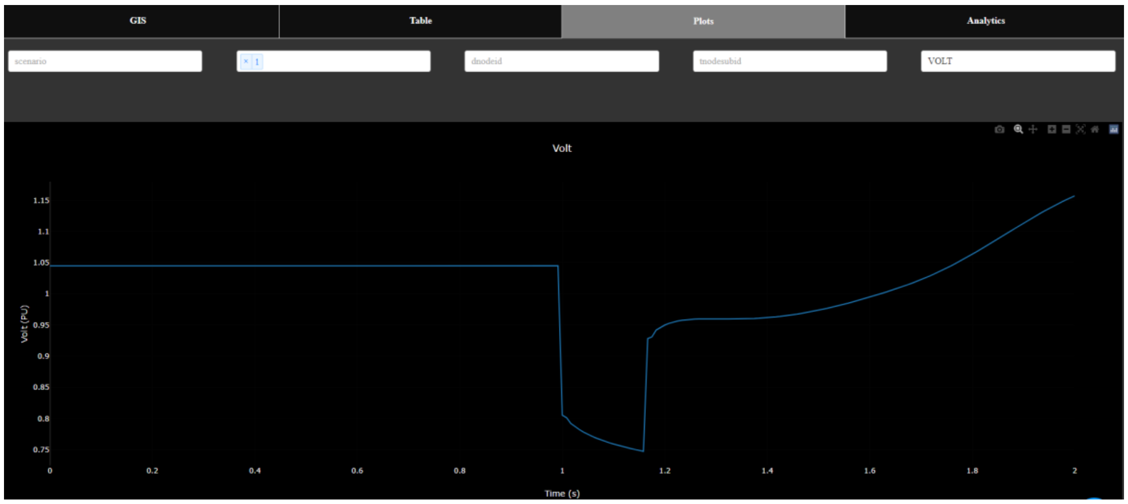
report example

Fig. 3. Interactive table on TDcoSim dashboard.

Plots

The Plots visualization tab allows the user to visualize any co-simulation variable as interactive time-series plots as shown in Fig. 4. The fields provided correspond to the attributes of the TDcoSim DataFrame and desired quantities can be plotted by appropriately plotting the fields. The plots may also be downloaded as .PNG images.

Note: All the quantities being plotted should have the same property. For e.g. you can't have a plot with voltage and angle from one node.



report example

Fig. 4. Time series plot on TDcoSim dashboard.

Data Analytics

The data analytics module provides a set of useful methods for extracting useful information from the co-simulation data. This will enable the user to perform analytics without having to write code for it. ### Using the data analytics The methods within the module be accessed using the Analytics tab on the TDcoSim dashboard. For users who want to use the module within their own Python scripts, it can be imported using:

```
from tdcosim.data_analytics import DataAnalytics
da=DataAnalytics()
```

All the methods take the TDcoSim DataFrame as input. The most useful methods available within the module are described below:

Entries below needs to be reveiwed before publishing

compute_stability_time: Determines whether the co-simulation variables within the data frame reach steady state, and the time taken to reach steady state after a disturbance event has occurred. The DataFrame has at least two columns: 1) value and 2) t.

- *Syntex:* stability_time, comment= compute_stability_time(df, error_threshold)
- *Inputs:*
  - *df:* DataFrame with at least two columns: 1) value, and 2) t.

    - *error_threshold:* Error threshold for stability time calculations. Maximum allowed signal deviation after stability time
- *Outputs:*
    - *stability_time:* Time it takes for the signal to stabilize after disturbance
    - *comment:* Message if the system is stable or not
    - *max_deviation:* Max signal amplitude deviation after stability

lag_finder: Calculate lag/delay between DataFrames df1 and df2 of the same length. Negative outputs show that signal df2.value lags behind df1.value and the positive output shows that df1.value lags behind df2.value.

- *Syntex:* delay = lag_finder(df1, df2)
- *Inputs:*
    - *df1:* DataFrame 1 with at least one column: value
    - *df2:* DataFrame 2 with at least one column: value
- *Outputs:*
    - *delay:* Delay between signals df1.value and df2.value

compute_mean_square_error: Calculate the mean square error between DataFrames df1 and df2 of the same length.

- *Syntex:* MSE = compute_mean_square_error (df1, df2)
- *Inputs:*
    - *df1:* DataFrame 1 with at least one column: value
    - *df2:* DataFrame 2 with at least one column: value
- *Outputs:*
    - *MSE:* Mean square error between signal df1.value and df2.value

shift_array: Shifts array by n bit. Fills extra bits at the end of the vector by a copy of the last bit of the array.

- *Syntex:* x = shift_array(y, n)
- *Inputs:*
    - *y:* Signal (type: np.array)
    - *n:* Number of bits to shift the signal. Positive values of n shifts signal y to the right and negative values shift signals to the left.
- *Outputs:*
    - *x:* Shifted signal

instances_of_violation: Calculates the number of instances where the value of data frame violates upper and lower bounds set by minValue and maxValue.

- *Syntex:* n = instances_of_violation(df,maxValue,minValue)
- *Inputs:*
    - *df:* DataFrame with at least one property: df.value
    - *maxValue:* Upper threshold
    - *minValue:* Lower threshold

- *Outputs:*
    - *n:* Number of instances when 'df.value' is outside the bounds of [minValue, maxValue]

exculde_value: Filter the given data frame based on >=toValue and <=fromValue conditions. For == condition use the same value for fromValue and toValue.

- *Syntex:* excludedDF = exculde_value(df,fromValue,toValue)
- *Inputs:*
    - *df:* DataFrame with at least one property: df.value
    - *fromValue:* Upper threshold
    - *toValue:* Lower threshold
- *Outputs:*
    - *excludedDF:* Filtered data frame

compare_signals: Compare and plot two signals in data frame df1 and df2 of the same length. Returns result in terms of stability time of both signals as well as lag and mean square error between them.

- *Syntex:* lag,MSE,Stability_time_1,Stability_time_2 = compare_signals(thisBusId1,thisBusId2,df1,df2,error_threshold,show_results)
- *Inputs:*
    - *thisBusId1:* Bus ID of data frame 1
    - *thisBusId2:* Bus ID of data frame 2
    - *df1:* DataFrame 1 with at least one property: df1.value
    - *df2:* DataFrame 2 with at least one property: df2.value
    - *error_threshold:* Error threshold for mean square error and stability time calculations.
    - *show_results:* Set to 1 to show the plot.
- *Outputs:*
    - *lag:* Delay between signals 1 and 2
    - *MSE:* Mean square error between signals 1 and 2
    - *Stability_time_1:* Time it takes for the signal 2 to stabilize after disturbance
    - *Stability_time_2:* Time it takes for the signal 1 to stabilize after disturbance

plot_vt_filt_fast_der: Plot the voltage signal at given transmission node and DERs in the connected distribution system.

- *Syntex:* plot_vt_filt_fast_der(df,tnodeid,legendDistNode,showPlot)
- *Inputs:*
    - *df:* Data Frame
    - *tnodeid:* Transmission node ID
    - *legendDistNode:* Set equal to True to show legend for distribution node id (Optional: if not specified set to False)
    - *showPlot:* Set equal to True to show the plot (Optional: if not specified set to False)

plot_t_vmag: Plot the voltage signal at given transmission node. If no transmission node id is given, it will plot voltage at all transmission nodes.

- *Syntex:* plot_t_vmag(df,tnodeid,excludeNodes)
- *Inputs:*
  - *df:* Data Frame
  - *tnodeid:* Transmission node ID (Optional: if not specified set to none)
  - *excludeNodes:* Nodes to be excluded from the plot (Optional: if not specified set to none)

plot_omega: Plot time vs rotor speed at all transmission nodes except specified.

- *Syntex:* plot_omega(df,excludeNodes=None)
- *Inputs:*
  - *df:* Data Frame
  - *excludeNodes:* Nodes to be excluded from the plot (Optional: if not specified set to none)

plot_distribution_der_data: Plots the time series plots for active and reactive power output from each DER.

- *Syntex:* plot_distribution_der_data(df,tnodeid=None,plotDerTotal=True)
- *Inputs:*
  - *df:* Data Frame
  - *tnodeid:* Transmission node ID
  - *plotDerTotal:* Set true to plot total P and Q of distribution DERs

show_plot: Generate plot with time as x-axis and ylable as y-axis

- *Syntex:* show_plot(ylabel,title)
- *Inputs:*
  - *ylabel:* Y-axis label
  - *title:* Plot title

clear_plot: Clear the plot

- *Syntex:* clear_plot()

Continue to Understanding the config file

Understanding the config file

The **config** file is the primary user interface for the TDcoSim package. Before starting the operation of TDcoSim tool and running test cases, it is necessary to understand how the configuration file can be used to setup the simulation conditions. This chapter describes the options within the configuration file and aids the user in setting up a simulation.

Config file options

The config file can be divided into three sections. The purpose of each option in every section is explained below:

Logging configuration

1. **level (int):** The logging level which can be either 10 (debug), 20 (info), 30 (warning) or 40 (error). Note that all log files will be saved in the **logs** folder.
2. **saveSubprocessOutErr (boolean):** Enable logging of error output corresponding to a T node in seperate ***.err** file.

PSSE configuration

1. **psseConfig (dict):** configuration for the transmission system.
   - *installLocation (string)* : full path for PSS/E transmission system python library location. If there isn't a specific location, system will look up the default installation path of PSS/E 33 (e.g. "C:/Program Files/PTI/PSSE35/35.0/PSSPY27").
   - *rawFilePath (string)* : full path for the PSS/E transmission system loadflow case file (e.g. "C:/folder/data/TNetworks/118bus/case118.raw").
   - *dyrFilePath (string)*: full path for the PSS/E transmission system dynamic case file (e.g. "C:/project_folder/data/TNetworks/118bus/case118.dyr").

OpenDSS + DER configuration

The default feeder configuration is **defaultFeederConfig**, which automatically assigns the same feeder to all the transmission system buses unless otherwise configured using **manualFeederConfig**.

1. **DEROdeSolver (string):** If fast DER models, specify **fast_der** since they always use the forward euler solver implimented in TDcoSim. If detailed DER models are being used, specify the solver (either **scipy** or **diffeqpy**).

2. **DEROdeMethod (string):** It is **not applicable** for fast DER models. It is used to specify the solver method for detailed DER models. The available options are **bdf** and **adams** for **scipy**, **CVODE_BDF** and **TRBDF2**for **diffeqpy**.

3. **openDSSConfig (dict):** Configuration for distribution feeders. The user can choose a default feeder configuration through defaultFeederConfig option or specify individual feeder for each transmission bus through manualFeederConfig option.

- ***defaultFeederConfig (dict):*** Default feeder configuration that assigns identical distribution feeders to all the transmission buses.
    - *filePath (string):* Specifies the path for the OpenDSS File (e.g. "C:/project_folder/data/DNetworks/123bus/case123ZIP.dss").
    - *solarFlag (Boolean):* Specifies presence or absence of PV-DERs in a feeder.
    - *solarPenetration (float):* Specifies the total rated capacity of PV-DERs as a percentage of the total feeder load in dynamic co-simulation (e.g. 0.1).
- ***manualFeederConfig:*** Manually specify the distribution system configuration at the desired transmission bus.
    - *nodes (list of dict):* Specifies the configuration of the distribution system and DERs.
        - *nodenumber (integer)*: Specifies the transmission bus to which the distribution system will be connected.
        - *filePath (string)*: Specifies the path for the OpenDSS File containing the distribution system model. (e.g. "C:/project_folder/data/DNetworks/123bus/case123ZIP.dss")
        - *solarFlag (bool):* Specifies presence or absence of PV-DERs in the distribution system.
        - *solarPenetration (float)*: Specifies the total rated capacity of PV-DERs as a percentage of the total feeder load in dynamic co-simulation (e.g. 0.1). It will only be used if *DERSetting* is *default*.
        - *fractionAggregatedLoad (dict):* It specifies the type of load model used to model the load at the T node. The available options are: "cmld", "clod"
        - *DERFilePath (string)*: This is only applicable for detailed DER models. Specifies the path for the JSON file containing the parameters and settings defining the DER model. Note that a single file may contain settings for any number of DER models.
        - *initializeWithActual (bool)*: This is only applicable if detailed DER models are being used in the feeder. If ***true***, the actual power output from the DER model instance will be used when setting up T+D co-simulation. If ***false***, the rated power output given in DER settings will be used.
        - *DERSetting (string)*: Specifies DER configuration at each node. If ***PVPlacement***, the DER at each node will use a unique set of DER settings. If ***default***, DERs at all nodes will use the same DER settings.

- *DERModelType (string)*: This is only applicable if detailed DER models are being used in the feeder. Specifies the type of detailed DER model to be used in each node of the particular feeder. Valid options are: "ThreePhaseUnbalanced","ThreePhaseBalanced","ThreePhase UnbalancedConstantVdc", "ThreePhaseUnbalancedNumba"
- *DERParameters (dict):* Specifies the configuration of PV-DERs to be used in the distribution system. If *PVPlacement* **is provided**, the DER at each node will need a separate set of DER settings. If *default* **is provided**, DERs at all nodes will use the same DER settings.
    - *default:* DER settings that will be used if *DERSetting* is *default*. Note that these settings are optional if *DERSetting* is *PVPlacement*.
        - *pvderScale (float):* Specifies the scaling factor with which to multiply the DER power output from any given node. A higher value of *pvderScale* for similar *solarPenetration* will result in lower number of DER model instances.
        - *pref:* This is only applicable if fast DER models are being used in the feeder. Specifies the rated active power output of the fast DER model.
        - *qref:* This is only applicable if fast DER models are being used in the feeder. Specifies the rated reactive power output of the fast DER model.
        - *interconnectionStandard:* This is only applicable if fast DER models are being used in the feeder. It specified the interconnection settings that must be used during voltage anomalies. The in
        - *derId (string):* This is only applicable if detailed DER models are being used in the feeder. The key word corresponding to DER model that is available in DER config file. The interconnection settings for detailed DER are also specified through the *derId*. Description of all the parameters that can be specified through the *derId* may be found here. Note that an exception will be thrown if a matching *derId* is not found in the DER config file.
        - *powerRating (float):* This is only applicable if detailed DER models are being used in the feeder. Specifies the rated power of DER in kVA.

Note that value specified here will override the rated power given in the DER config file.

- *VrmsRating (float):* This is only applicable if detailed DER models are being used in the feeder. Specifies the rated RMS voltage (L-G) of the DER in Volts. The tool automatically adds a transformer to connect the DER to the distribution system. Note that value specified here will override the rated voltage given in the DER config file.

- *steadyStateInitialization (bool):* This is only applicable if detailed DER models are being used in the feeder. Specifies whether the states in PV-DER model is to be initialized with steady state values before simulation is started.

- *PVPlacement (dict):* Specifies the distribution system node and the details of the DER to be connected to that node. Note that these settings are optional if *DERSetting* is *default*.

  - *node (string):* Any three phase node in the OpenDSS model. Note that the keyword ***node*** must be replaced with the actual node name.
    - *derId (string):* This is only applicable if detailed DER models are being used in the feeder. The key word corresponding to DER model that is available in DER config file. Note that an exception will be thrown if a matching *derId* is not found in the DER config file.
    - *powerRating (float):* This is only applicable if detailed DER models are being used in the feeder. Specifies the rated power of DER in kVA. Note that value specified here will override the rated power given in the DER config file.
    - *pvderScale (float):* This is only applicable if detailed DER models are being used in the feeder. Specifies the scaling factor with

which to multiply the DER power output from any given node.

- *VrmsRating (float):* This is only applicable if detailed DER models are being used in the feeder. Specifies the rated RMS voltage (L-G) of the DER in Volts. The tool automatically adds a transformer to connect the DER to the distribution system. Note that value specified here will override the rated voltage given in the DER config file.

Note: Please check sections 6.4.1 and 6.4.2 in IEEE 1547-2018 for more information on voltage ride-through and trip settings.

Simulation configuration

1. **simulationConfig (dict):** Configuration options for the simulation.
   - ***simType (string):*** A string specifying whether the simulation is **'static'** or **'dynamic'**.
   - ***defaultLoadType***: The default load model that is attached to the T node's that do not have distribution system feeder models connected to it. Available options are: "cmld", "clod"
   - ***dynamicConfig (dict):*** Configuration options for **'dynamic'** simulation.
     - *events (dict):* Specifies the various events in the dynamic simulation listed sequentially.
       - *time (float):* Specifies the time at which an event occurs.
       - *type (string):* Specifies the type of an event (**'simEnd'**, **'faultOn'** or **'faultOff'**).
       - *faultBus (integer):* Specifies the transmission bus at which the fault occurs.
       - faultImpedance (list of floats): Specifies the impedance of the fault.
   - ***staticConfig (dict):*** Configuration options for **'static'** simulation.
     - *loadShape (list of floats):* Load served by the T+D system at each time interval (e.g. [0.81,0.75,0.72,..])
   - ***protocol (string):*** Specifies the nature of coupling between Transmission system and Distribution System (valid options:**'loose_coupling'**,**'tight_coupling'**).
   - ***memoryThreshold (float):*** Specifies the threshold before the co-simulation results collected in memory (RAM) are written to disk. This should be

adjusted by the user depending upon the system memory to avoid crashes due to out of memory errors. Note:*** Tight coupling protocol is only available for static co-simulation in current version.

Note: For static co-simulation, solar penetration needs to be implemented through the **.dss** file. Option to add solar shape through config file will be implemented in next version.

Output configuration

1. **outputConfig (dict):** Configuration options for the data generated during simulation..
   - ***simID***: The main identifier string for the co-simulation. It will also be used to name the folder which will store the co-simulation results..
   - ***scenarioID***: An additional identifier string for the co-simulation. It will be included in the dataframe containing the co-simulation results.
   - ***outputDir (string):*** Main folder within which the results generated during co-simulation would be stored. There will be sub-folders corresponding to the **simID** within this folder.
   - ***outputfilename (string):*** File name given to the **.out** file containing the result from PSS?E.
   - ***type (string):*** Format in which the results generated during simulation would be stored. Available options are: **dataframe**, **csv**.

config.json example

The example below shows the **config** file for a T & D **dynamic** co-simulation with a IEEE 118 bus transmission system having IEEE 123 node test feeder connected to all the buses for a simulation lasting for **1.0** seconds, with **0 %** PV-DER penetration.

```
{
    "cosimHome":"C:\\Users\\username\\Documents\\TDcoSim\\CoSimulator",
    "logging": {
         "level": 20,
         "saveSubprocessOutErr": true},
    "psseConfig":{

"rawFilePath":"C:\\Users\\username\\Documents\\TDcoSim\\SampleData\\TNetworks
\\118bus\\case118.raw",

"dyrFilePath":"C:\\Users\\username\\Documents\\TDcoSim\\SampleData\\TNetworks
\\118bus\\case118.dyr"
    },
    "openDSSConfig":{"DEROdeSolver":"fast_der",
        "defaultFeederConfig":{
```

```
"filePath":["CC:\\Users\\username\\Documents\\TDcoSim\\SampleData\\DNetworks\
\123Bus\\case123ZIP.dss"],
            "solarFlag":0,
            "solarPenetration":0.0,
            "DERParameters": {
                        "default": {
                            "pvderScale": 1,
                            "pref":10,
                            "qref":0,
                            "sbase":10
                        }
                             },
                    },
    "simulationConfig":{
        "simType":"dynamic",
        "dynamicConfig":{
            "events":{
                "1":{
                    "type":"simEnd",
                    "time":1.0
                    }
                     }
        },
            },
"memoryThreshold": 20480.0,
"protocol":"loose_coupling",
"outputConfig": {
   "simID": "test",
   "type": "dataframe",
   "outputDir": "..\\output",
   "outputfilename": "test",
   "scenarioID": "0percent"
}
}
```

T&D co-simulation with PSS/E and OpenDSS

This section provides a brief overview of T & D co-simulation with PSSE and OpenDSS as implemented in TDcoSim.

Advantages

The state-of-the-art approach of studying DER impact on the bulk power system entails developing aggregate positive-sequence load and DER model for distribution systems and then applying these models in conventional positive-sequence transmission system simulation software such as PSSE, PSLF, etc.

TDcoSim offers higher degree of fidelity since there is no need to parameterize the aggregate load and DER models for distribution system, which are only approximate representation of actual distribution system behaviors,

TDcoSim also offers time savings by running an integrated T&D co-simulation compared to the approaches of running separate simulations for T&D systems and manually combining the results.

Assumptions in current software version

1. It is assumed that all the distribution feeders connected to the transmission system load bus have the same characteristics. Hence only one distribution feeder is simulated in an OpenDSS instance. The power output from the single feeder is then multiplied with a scaling factor (calculated automatically by TDcoSim) so that we can match the rated load at the transmission bus with the total load from all the feeders connected to the bus.
2. No Sub-station is explicity added to the distribution network model by TDcoSim to interface the T bus with the distribution feeder. However if the user provided distribution network model comes with a sub-station, then it is used.

T&D interface

The transmission system simulator (TSS) and distribution system simulator (DSS) are separate programs with their own solution methods. **TDcoSim** is responsible for exchanging data and synchronizing their runs.

Data exchange

The TSS, PSSE uses positive sequence quantities while the DSS, OpenDSS uses phase quantities. Hence it is necessary to convert positive sequence quantities to equivalent phase quantities and vice-versa. After PSSE completes a solution, it outputs the sequence voltages at the T&D interface. Then (1) is applied to convert the sequence voltages at the boundary bus to phase voltages. Using the phase voltages at the boundary bus, the DSS completes a solution and outputs the phase current injection at the boundary bus, which are expressed in (2).

\$ \$

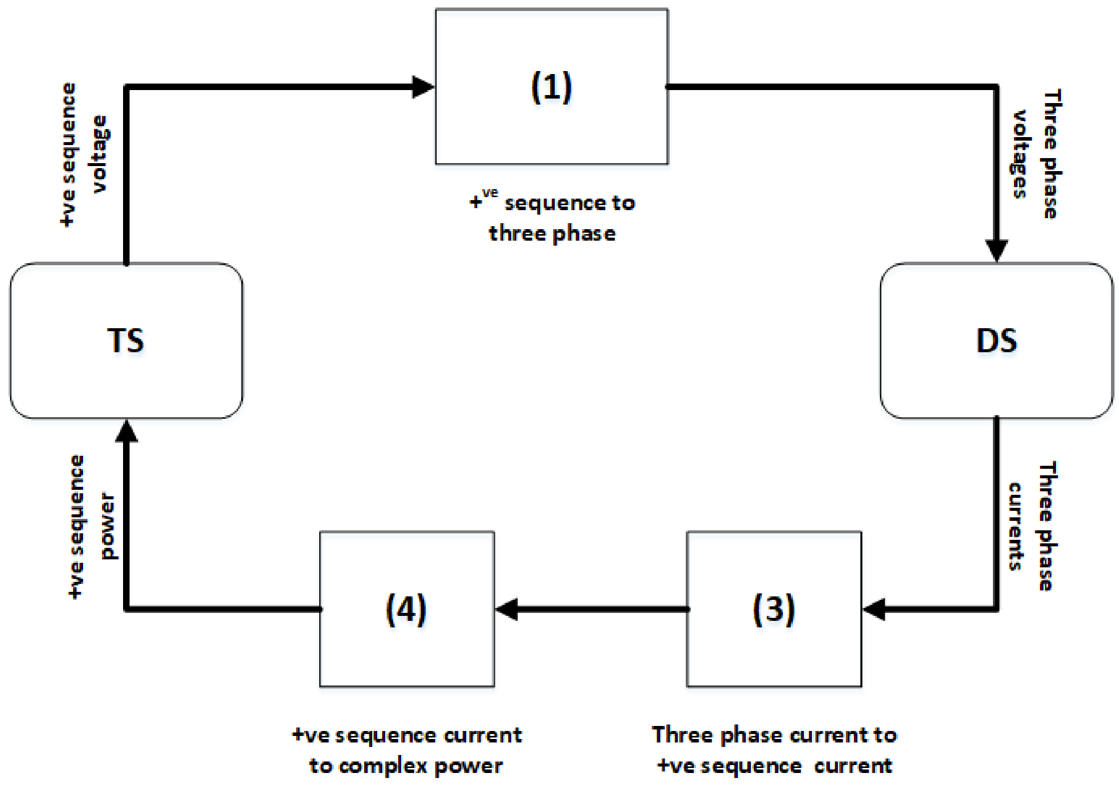
$$ \begin{equation} I_{DS}= \left[\begin{array} {r} \bar{I}_{DS,a}\\\bar{I}_{DS,b}\\\bar{I}_{DS,c} \end{array}\right]\tag{2} \end{equation} $$

The phase current injection at the boundary bus is converted to sequence quantities using (3). The current injection at the boundary bus is then used to obtain complex power injection at the boundary bus using (4).

$$ \begin{equation} \left[\begin{array} {r} \bar{I}_{DS,0}\\\bar{I}_{DS,+}\\\bar{I}_{DS,-} \end{array}\right]= \frac{1}{3}\left[\begin{array} {rrr} 1&&1&&1\\1&&a&&a^2\\1&&a^2&&a \end{array}\right]\left[\begin{array} {r} \bar{I}_{DS,a}\\\bar{I}_{DS,b}\\\bar{I}_{DS,c} \end{array}\right]\tag{3} \end{equation} $$

$$ \begin{equation} S_{TS,+}=3.\bar{V}_{TS,+}.\bar{I}^*_{TS,+}\tag{4} \end{equation} $$

The obtained value of S_TS,+ is used as the total power requirement for the said load bus at the transmission system. It is worth mentioning that the equivalent load that is replaced with the distribution system simulator is modeled as a constant power load with respect to the transmission system simulator. The data exchange across the T&D interface is illustrated in Fig. 1 where the loosely-coupled synchronization protocol is utilized (The synchronization protocols will be introduced in the next section).


(1)
+ve sequence voltage
Three phase voltages
+ve sequence to three phase
TS
DS
+ve sequence power
Three phase currents
(4)
(3)
+ve sequence current to complex power
Three phase current to +ve sequence current

loosely coupled protocol

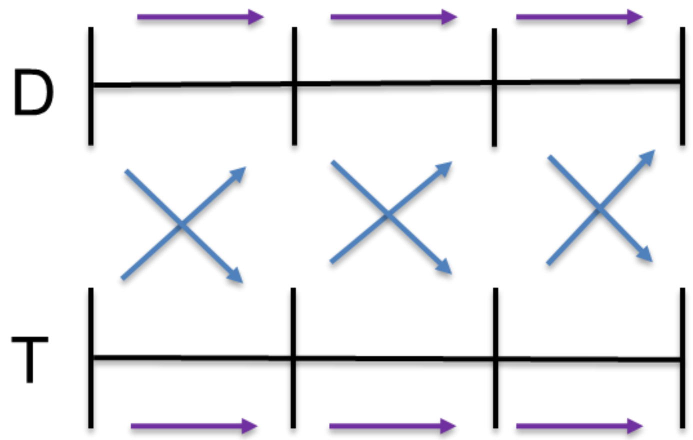
Fig. 1. Loosely couple protocol for data exchange across T&D interface.

Synchronization

Both static and dynamic co-simulations starts with an initialization for both PSSE and OpenDSS software. The T&D interface contains sockets that enable simulators to communicate and exchange data. Specifically, the TSS and the DSS are synchronized through two protocols, namely, loosely-coupled and tightly-coupled protocols.

Loosely-coupled protocol

The loosely-coupled protocol is illustrated in Fig. 2. There is a one-step lag in information exchange between transmission and distribution, but less computation is required.



loosely coupled protocol

Fig. 2. Loosely-couple protocol.

Steady-state T&D co-simulation process

The data exchange protocol for steady-state co-simulation in the current version of the software may be tightly-coupled or loosely-coupled. The initialization steps are:

1. Input case files (both PSSE and OpenDSS).
2. Run power flow in PSSE.
3. Get voltage (p.u.) at load buses from PSSE.
4. Set OpenDSS VSource voltage to be equal to the voltage from the respective load bus.
5. Run power flow in OpenDSS, get P and Q requirement for DNetworks.
6. Scale P & Q using scaling factor and set as input to T bus.

The scaling factor (see assumptions) is calculated by dividing the total load at a transmission system load bus by the aggregated load of one distribution feeder.

At every time step (one half-cycle), the steps 2 to 6 are repeated untill the end of simulation.

Dynamic T&D co-simulation process

The data exchange protocol for dynamic co-simulation in the current version of the software is loosely coupled. The initialization steps are:

1. Input case files (both PSSE and OpenDSS).
2. Run power flow in PSSE.
3. Get voltage (p.u.) at load buses from PSSE.
4. Set OpenDSS VSource voltage to be equal to the voltage from the respective load bus.
5. Run power flow in OpenDSS, get P and Q requirement for DNetworks.
6. Compute difference of P & Q between TNetworkBus and DNetwork.
7. Calculate compensating shunt value using $P_{shunt} + j * Q_{shunt} = V_{pu}^2 * (YP_{shunt} - YQ_{shunt})$.
8. Add YP_{shunt} and YQ_{shunt} as fixed compensating shunt in PSSE.

At every time step (one half-cycle), the steps 2 to 8 are repeated until the user specified simulation end time is reached.

Capability and Limitations

TDcoSim uses PSS®E as transmission system simulator and OpenDSS as Distribution system simulator. In general, **TDcoSim** can be used for analysis that is supported by PSS®E and OpenDSS. On the other end of the spectrum **TDcoSim** is limited by the limitations of PSS®E and OpenDSS. PSS®E is a positive sequence simulator, as a result, studies that involve unbalanced faults on the transmission side cannot be studied in detail.

Types of Analysis

- Quasi-static time series (QSTS)
  - Solves power flow for both transmission and distribution system
  - Uses a tightly-coupled scheme to interface T&D systems
  - Use of tightly-coupled scheme ensures that the obtained boundary solutions are stable. In other words the solution will be the same (within a user defined tolerance) if one were to solve both the T&D system together as a single large system.
  - Loadshapes can be provided in the configuration file to model the change in load over time. It is assumed that all the loads in a given distribution feeder follow the same loadshape.
- Dynamic simulation
  - Unlike the QSTS, which solves a set of algebraic equations on both the T and the D side, the dynamic simulation solves differential algebraic equations (DAEs) on the T-side and D-side
  - OpenDSS solves power flow for the distribution system

- When distributed solar generation is modeled, the dynamic model of PVDER is used. Thus, the combination of OpenDSS and PVDER makes the D-side equations DAEs.
- Depending on the configuration multiple pvder instances will be used
- Balanced faults on the transmission side can be modeled
- Cloud cover event – particularly relevant for system studies involving large solar farms
- Uses a loosely-coupled

Assumptions on Reduction in System Inertia

TDcoSim was specifically designed to study distributed energy resource (DER) integration. Hence, it is designed to support two different viewpoints, * The user can defined the system in detail using existing study models, or * The user can setup future scenarios with varying levels of solar penetration

TDcoSim makes certain assumptions in the latter case. As an example, let us say the user wants to study 10 and 20 percent solar penetration scenarios. In both the cases, the user will provide the same T and D systems. This means certain amount of conventional generation from the transmission system needs to be reduced in order to accomodate the increased solar generation. The realistic way to accomplish this would be to find out the units committed for the given scenario, run an optimal power flow to find the generator set points, then start the simulation at that operating point. However, the data required to do that is not typically available and/or not provided by the user. **TDcoSim** overcomes this problem by making the following assumptions,

- Reduce generation of each unit by solar penetration value. For example, if solar penetration is 10 percent then reduce each generator output by 10 percent.
- Reduce the inertia constant of each generator the same way

TDcoSim advanced usage

Importing TDcoSim

TDcoSim can be imported and used like a normal Python module. Note that package name is in lower case.

```
import tdcosim
```

Using TDcoSim within your script

The basic steps to write your own co-simulation program are as follows:

1. Setup desired T+D or T+D+DER system by making necessary entries in the **config** file.

2. Import necessary classes.

   ```
   from tdcosim.report import generateReport
   from tdcosim.global_data import GlobalData
   from tdcosim.procedure.procedure import Procedure
   ```

3. Read the **config** file and initialize the T&D system.

   ```
   GlobalData.set_config('config.json')
   GlobalData.set_TDdata()
   ```

4. Create a procedure object for the simulation and call *simuate()* method.

   ```
   procedure = Procedure()
   procedure.simulate()
   ```

5. Generate report after *simuate()* exits.

   ```
   generateReport(tdcosim.GlobalData,fname='report.xlsx')
   ```

Continue to Software details

Software details

Features

The software has the following features.

1. Capable of launching sub-processes for individual feeders. The parallelization helps improves the scalability of the software.
2. Capable of configuring each feeder with different DER penetration levels, ratings, and voltage ride through settings (compatible with IEEE Std 1547-2018).
3. Capable of introducing fault events during simulation.
4. Captures and reports data from both transmission and distribution system for the entirety of the co-simulation.

Main components

TDcoSim comprise interface modules and a synchronization module. There are separate interface modules with sockets and interaction protocols for both the T & D and D & DER co-simulations.

1. T&D interface: A Python program that exchanges and iterates information (voltages, currents, and powers) between T&D simulators through synchronization protocols (loosely or tightly-coupled). The Python-based T&D interface is easy to use and adds minimal overhead.

2. D&DER interface: A Python program that exchanges information between the distribution system simulator and the dynamic DER model.

3. Configuration file: It is the main user interface where the user can configure a T+D or T+D+DER co-simulation.

External components

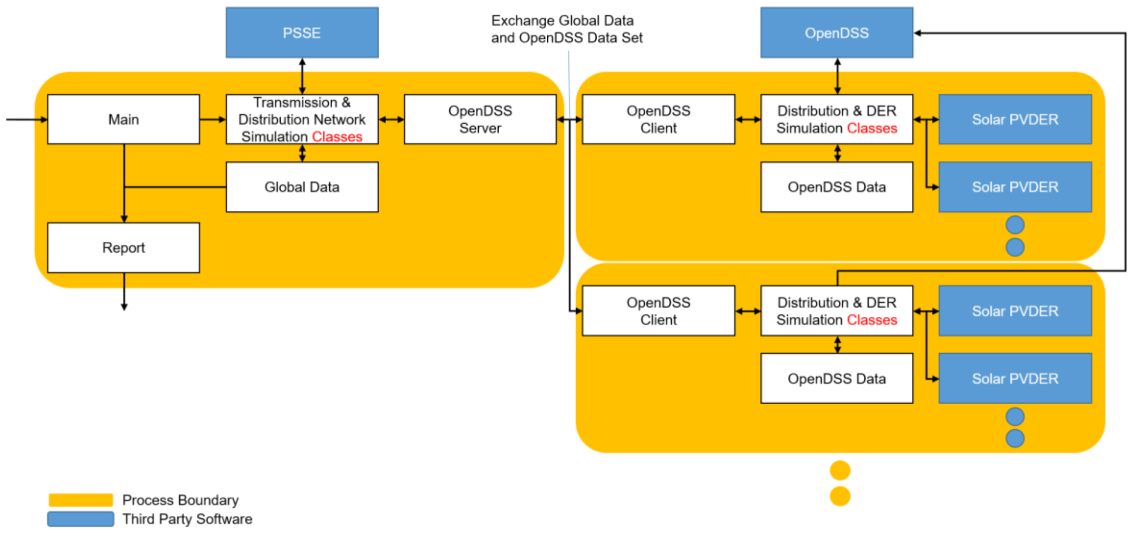
1.PSS®E It is an off-the-shelf positive-sequence transmission system simulator.

2.OpenDSS It is a three-phase unbalanced distribution system simulator.

3.Solar PV-DER simulation utility It is Python utility than can simulate dynamics of grid connected solar PV-DER systems. It uses dynamic phasor models and has single and three-phase PV-DERs.

Software architecture

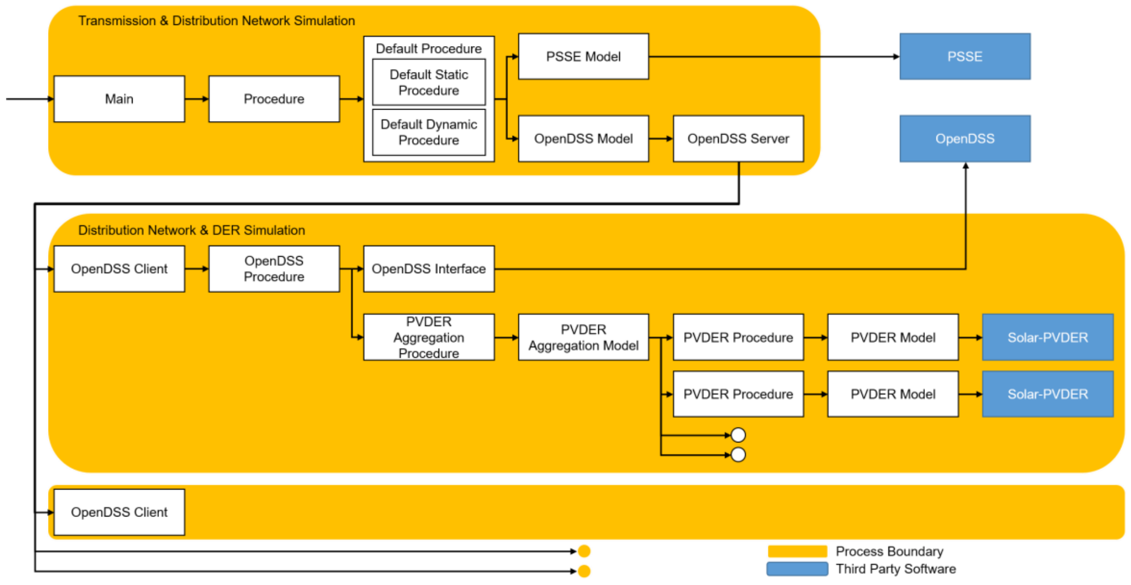
A schematic showing the software architecture of the TDcoSim package is shown in the Fig. 1.



highlevel software architecture

Fig. 1. Highlevel software architecture

TDcoSim runs with multiple processes. The main process runs the transmission network simulation with PSSE and generates a report of the simulation. Each sub-processes runs the distribution network simulation with OpenDSS and PV-DER. The tool uses the TCP sockets to exchange the data between main and sub-processes. The detailed simulation architecture is shown in the Fig.2.



detail simulation architecture

Fig. 2. Detailed simulation architecture

The simulation is managed by procedures for each model. The procedures define the simulation orders between multiple simulation objects. Each model represent a single simulation object that represents a specific component of T&D system. The procedures have hierarchical one-to-many relationships. The simulation type procedures are connected to multiple OpenDSS procedures via OpenDSS model, and the OpenDSS procedures are connected to multiple PVDER procedures via PVDER Aggregation model.

Sequence of operations

The sequence of operations within the program is listed below:

1. Open TCP server sockets
2. Create a transmission network object
3. Create transmission network bus objects
4. Create distribution network objects
5. Create sub-processes
6. Establish TCP connection
7. Connect to OpenDSS via COM-Interface
8. Run the distribution network simulation

Performance Guide

TDcoSim has the capability to scale from a single distribution feeder with a single DER to thousands of distribution feeders with a hundreds of DER. Due to difference in computing effort and memory footprint, the solution time increases.

In this guide we provide suggestions on reducing solution time for a given co-simulation project.

Solution time

The total solution time is comprised of three parts: ***Tsolution = Tinitialize + Tco-sim + Tpost-process*** Where, *Tinitialize* is the time take to setup the co-simulation, *Tco-sim* is the time taken for co-simulation time stepping, and *Tpost-process* is the time take for post-processing the simulation time. Not that each solution time component is comprised of multiple tasks.

- Tinitialize: Here individual components co-simulation is initialized.
  - Read JSON configuration files
  - Launch sub-processes
  - Setup DER model integrators (for dynamic simulation)
  - Initialize co-simulation to steady state
- Tco-sim: Here we do time stepping for individual component.
  - PSS/E model:
    - Solve power flow for transmission system model.
    - Perform integration for dynamic models with a time step of one half-cycle (1/120 s).
    - Write to *.out* file
  - OpenDSS model: Solve non-linear algebraic problem
  - PV-DER model:
    - Fast-DER: Perform integration with a fixed time step.
    - Dynamic Phasor: Perform integration with variable time step
  - Write to disk if memory threshold is exceeded.
- Tpost-process: Here we post process the output and write them to user defined file type from memory.

Solution time contributors

The solution time may also be expressed as a function of the specific T&D+DER co-simulation model. **Tsolution = f(nBus) + f(nFeeder) + nDERFeeder*f(nDERperFeeder)**

Solution time benchmarks on test systems

The following benchmark times were obtained for the following scenario using workstation with the recommended specifications mentioned in System requirements:

T system: IEEE 118 bus system

D feeder: IEEE 123 node feeder

Simulation time: 20.0 s

Events: Three phase fault from 0.2 s to 0.3 s

Number of T nodes with feeders and DER: 97

DER Model type	DER solver type	PV penetration (%)	DER models	Solution time/Simulation time/DER (s/s/DER)
ThreePhaseBalanced - 50 kW	DiffEqPy (CVODE_BDF)	25	1746	0.047
ThreePhaseUnBalanced - 50 kW	DiffEqPy (CVODE_BDF)	25	1746	0.054
Fast DER - 50kW	Forward Euler	25	1746	**0.038**

Example 1: DER Tripping and DER riding through fault (detailed DER)

In this example, TDcosim is used to test the following scenarios on the same T+D+DER system. * DERs instantaneously tripping during a voltage anomaly caused by a T bus fault. * Config file * DERs riding through a voltage anomaly caused by a T bus fault * Config file

Co-simulation setup

1. **T system:** 118 bus system

```
"psseConfig": {
               "rawFilePath": "data\\transmission\\118bus\\case118.raw",
               "dyrFilePath": "data\\transmission\\118bus\\case118.dyr"
               }
```

2. **D + DER system:** 123 node feeder connected to bus 1 of 118 bus system.
   - DER penetration: 10% of distribution system load
   - DER model type: Detailed DER model
     - DER model sub-type:Three Phase Unbalanced

```
"manualFeederConfig":{
                        "nodes": [
                            {
                                "nodenumber": 1,
                                "filePath":
["\\SampleData\\DNetworks\\123Bus\\case123ZIP.dss"],
                                "solarFlag":1,
                                "DERModelType": "ThreePhaseUnbalanced",
                                "solarPenetration": 0.1,
                                "initializeWithActual": true,
                                "DERFilePath":
"config\\detailed_der_default.json",
                                "DERParameters":{
                                "default":{
                                     "pvderScale": 1,
                                     "steadyStateInitialization": true,
                                     "derId": "50_instant_trip"
                                          }
                                               }
                            }
                        ]
                }
```

3. **Simulation configuration:** A 3 phase fault is applied in bus 5 of the T-system. The simulation configuration to apply fault on bus 5 is shown below:

```
"simulationConfig":{
        "protocol": "loose_coupling",
        "simType":"dynamic",
        "dynamicConfig":{
            "events":{
                "1":{
```

```
                                "time":0.5,
                                "type":"faultOn",
                                "faultBus":5,
                                "faultImpedance":[0.0,-100000]
                                },
                            "2":{
                                "time":0.6,
                                "type":"faultOff",
                                "faultBus":5
                                },
                            "3":{
                                "time":1.0,
                                "type":"simEnd"
                                }
                                 }}
```

Scenario A: Instantaneous trip

The DER trip setting used for this case is shown in Figure A below. The full configuration file is

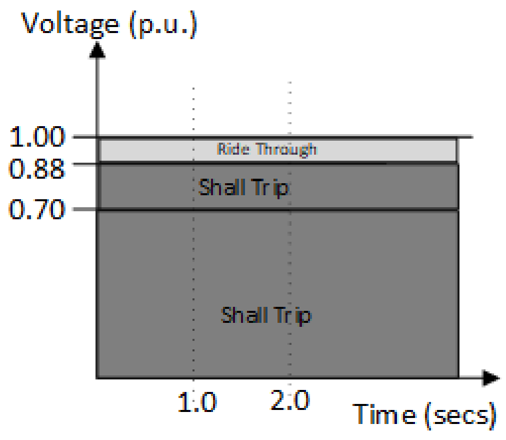


Figure A: DER operational settings curve for the instantaneous trip settings. 4a. **DER Interconnection standard:** TRIP instantaneously on voltage anomaly (i.e. voltage below level "0" threshold).

```
   {
    "50_instant_trip":{"parent_config":"50",
                   "LVRT":{"config_id":"LVRT_instant_trip"}
                   },
   "LVRT_instant_trip":{"config":{"1":{"V_threshold":0.88,
                                       "t_threshold":0.001,
                                       "t_min_ridethrough":0.001,
                                       "mode":"momentary_cessation"
                                      }
                                 }
                      }
   }
```

Scenario B: Ride through

The DER trip setting used for this case is shown in Figure B below.

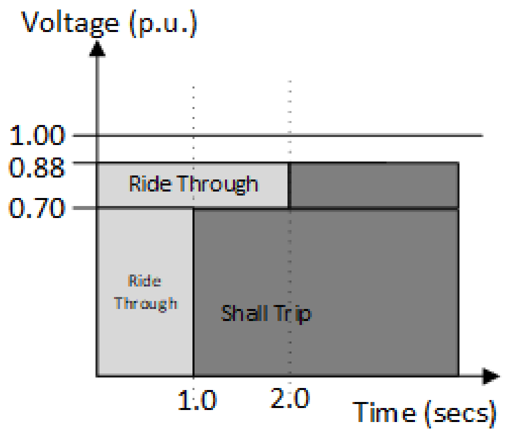


Figure B: DER operational settings curve for the DER ride through settings.

4b. **DER Interconnection standard:** Ride through during the voltage anomaly (i.e. voltage below level "0" threshold).

```
{
   "50_ridethrough":{"parent_config":"50",
                     "LVRT":{"config_id":"LVRT_ridethrough"}
                    },
   "LVRT_ridethrough":{"config":{"1":{"V_threshold":0.88,
                                      "t_threshold":10.0,
                                      "t_min_ridethrough":20.0,
                                      "mode":"mandatory_operation"
                                     }
                                }
                      }
}
```

Results

Figure 1 compares the active power component of the load observed in the T-bus for the three cases considered. It can be observed that case C, without DER on the distribution starts off with higher initial net load. Case A and Case B has a lower initial net load due to the DER connected in the distribution system masking the portion of total load in the system. Here net load is defined as the difference of the total load in the distribution system and the DER connected in the distribution system.

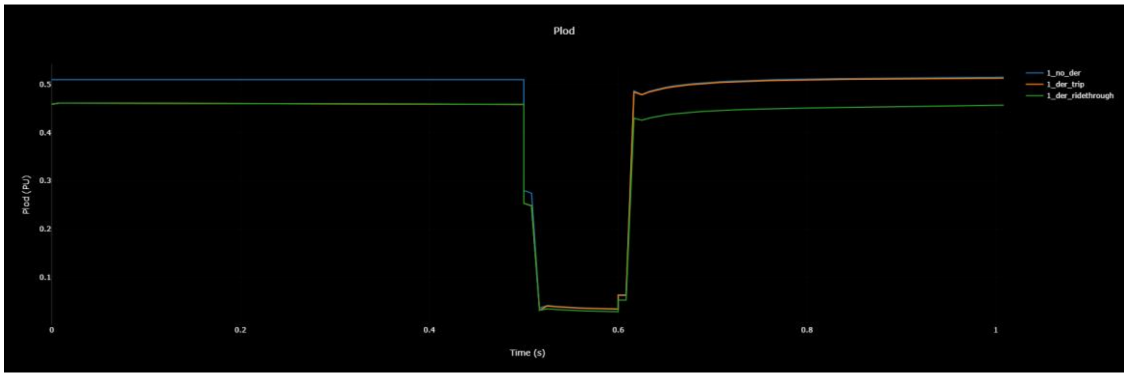


Figure 1: Active component of load as observed at the T-bus for the cases considered. (A): 10% DER penetration with DER TRIP Settings, (B): 10% DER penetration with DER RT Settings and (C) 0% DER penetration.

For the DER trip case, Case A, it can be observed that the net load observed in the bus increases to a value equal to the case without any DERs in the system, case C, which is an expected response of the system as net load in the T-bus reverts back to the total load as DER in the distribution system trips. A similar response can be observed for the reactive power component of the net load in the system as shown in Figure 2, which shows that the net reactive power equals the total reactive power as when DER trips, the system reverts back to the operational condition before DER connection in the system.

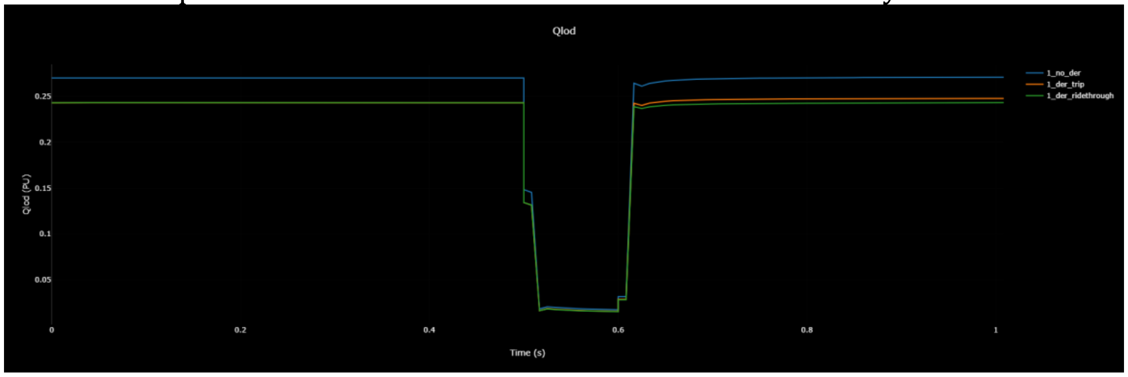


Figure 2: Reactive component of load as observed at the T-bus for the cases considered. (A): 10% DER penetration with DER TRIP Settings, (B): 10% DER penetration with DER RT Settings and (C) 0% DER penetration.

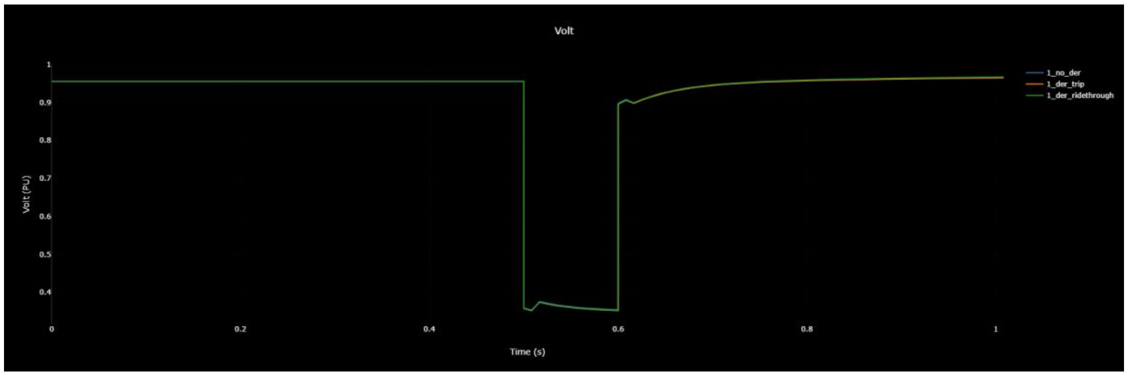


Figure 3: T-bus 1 voltage comparison for the cases considered. (A): 10% DER penetration with

DER TRIP Settings, (B): 10% DER penetration with DER RT Settings and (C) 0% DER penetration.

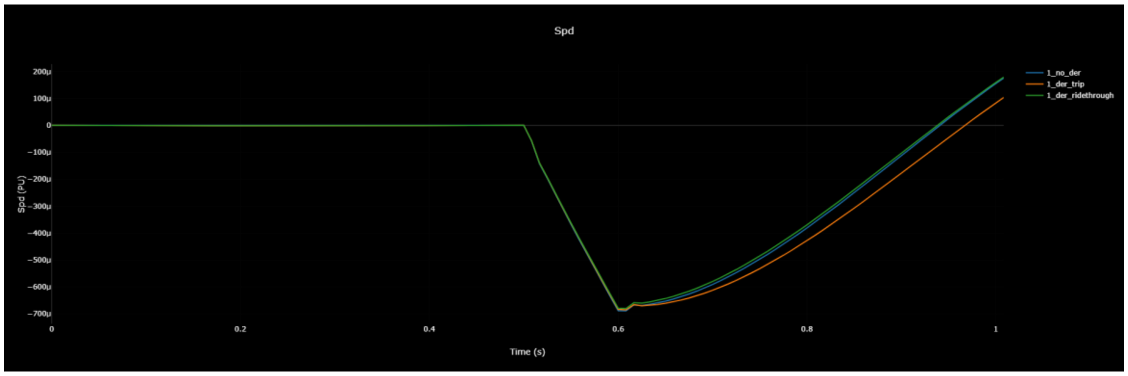


Figure 4: Generator 1 Speed Comparison for the different cases considered.

Figure 3 shows the transmission bus voltage for bus 1 for the three cases considered. It can be observed that the voltage at bus 1 is same for all the cases considered. This is because for this case bus 1, where distribution system is connected, also had a synchronous generator connected to it which was regulating the bus voltage. Figure 4 shows the generator rotor frequency for the cases considered. It can be observed that the frequency nadir following system fault close to the fault location is lower for the case with DER trip. More tests with more distribution system and DERs should be performed to properly study the impact of DERs on system frequency response.

Example 2: DER penetration and steady state initialization (detailed DER and fast DER)

In this test study, different penetration level of DERs within one distribution system connected to a transmission bus is studied. The purpose of this study is to test the ability of the tool to properly initialize the states of all the dynamic components of the system. Without any disturbance introduced in the system through changes in operating points or faults, it is expected that the responses of the various components in the system be a flat profile if the state of the various components are properly initialized i.e. all the variables should have a constant value throughout the duration of simulation.

Co-simulation setup

1. **T system:** 118 bus system

```
"psseConfig": {
               "rawFilePath": "data\\transmission\\118bus\\case118.raw",
               "dyrFilePath": "data\\transmission\\118bus\\case118.dyr"
               }
```

2. **D + DER system:** 123 node feeder connected to bus 1 of 118 bus system.
   - DER penetration: Varied from 0 % (0.0) to 50 % (0.5) of distribution system load in 10 % increments.
   - DER rated capacity: 50 kVA 2a. **DER model type:** Detailed DER model

```
"manualFeederConfig":{
                        "nodes": [
                            {
                                "nodenumber": 1,
                                "filePath":
["\\SampleData\\DNetworks\\123Bus\\case123ZIP.dss"],
                                "solarFlag":1,
                                "DERModelType": "ThreePhaseUnbalanced",
                                "solarPenetration": 0.1,
                                "initializeWithActual": true,
                                "DERFilePath":
"config\\detailed_der_default.json",
                                "DERParameters":{
                                "default":{
                                     "pvderScale": 1,
                                     "steadyStateInitialization": true,
                                     "derId": "50"
                                          }
                                               }
                            }
                        ]
                    }
```

2b. **DER model type:** Fast DER model

```
"manualFeederConfig": {
         "nodes": [
            {
               "solarFlag": 1,
               "solarPenetration": 0.1,
               "DERParameters": {
                  "default": {
                     "qref": 0,
                     "pvderScale": 1,
                     "pref": 50,
                     "sbase": 50
                  }
               },
               "nodenumber": 1,
               "filePath": ["data\\distribution\\123Bus\\case123ZIP.dss"
               ]
            }
         ]
      }
```

3. **Simulation configuration:** The co-simulation is run for 10 s and no faults are applied on the T side buses:

```
"simulationConfig":{
        "protocol": "loose_coupling",
        "simType":"dynamic",
        "dynamicConfig":{
            "events":{
                "1":{
                    "time":10.0,
                    "type":"simEnd"
                    }
                     }}
```

Results

Figures 1-3 shows the flat start performance of the TDcosim tool for different penetration level of DER. It can be observed that with the DER added into the test system, TDcosim tool takes couple of secs to reach the steady state active power consumption at the DER connected bus. Also, note that the settling value of the net active power is slightly below to the calculated net load power based on DER penetration. One of the reason for this is that the addition of DER within the distribution feeder doesn't amount to an exact amount of net load drop within the distribution feeder. This depends on the various factors like power output of the DER, location of DER, nature of the loads modelled and so on. We are working to have the net load initialized properly.

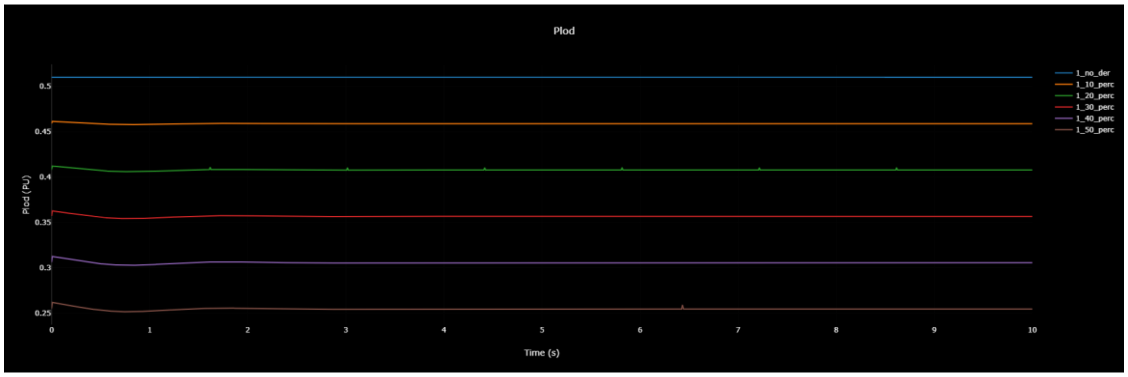


Figure 1: Load observed in bus 1 for the different cases considered (Green: No DER, Red: 10% DER, Cyan: 20% DER, Black: 30% DER, Blue: 40% DER).

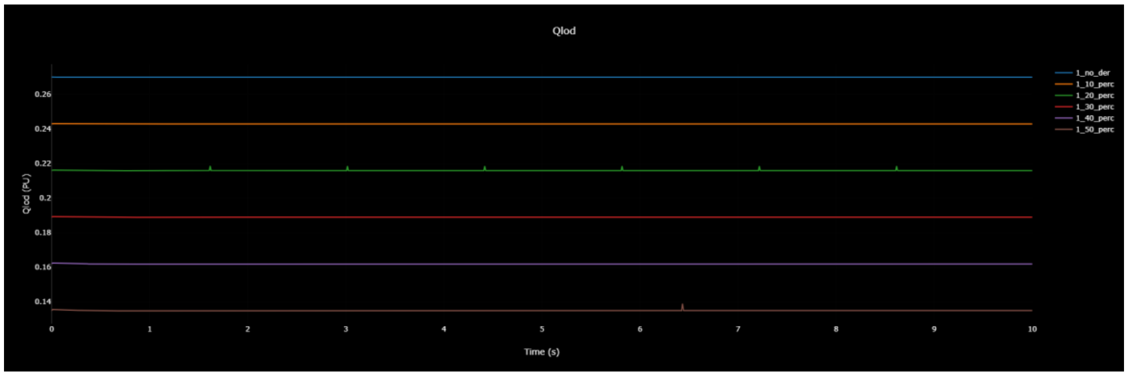


Figure 2: Reactive power observed in bus 1 for the different cases considered. (Green: No DER, Red: 10% DER, Cyan: 20% DER, Black: 30% DER, Blue: 40% DER)

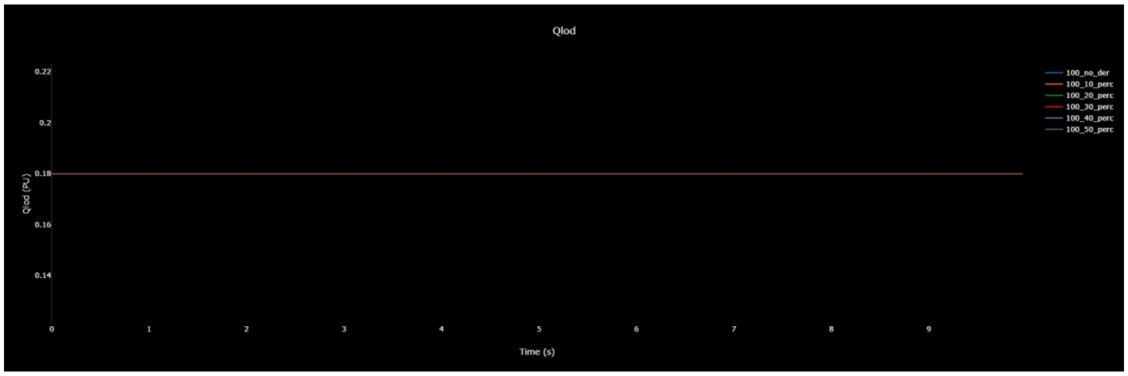


Figure 3: Reactive power observed in bus 100 for the different cases considered. (Green: No DER, Red: 10% DER, Cyan: 20% DER, Black: 30% DER, Blue: 40% DER)

Figure 2 shows the reactive power observed in bus 1 for the different cases considered. It can be observed that the reactive power consumed at the DS & DER connected bus decreases as DER penetration increases. Figure 3 shows no change in the reactive power for a random bus (bus 100).

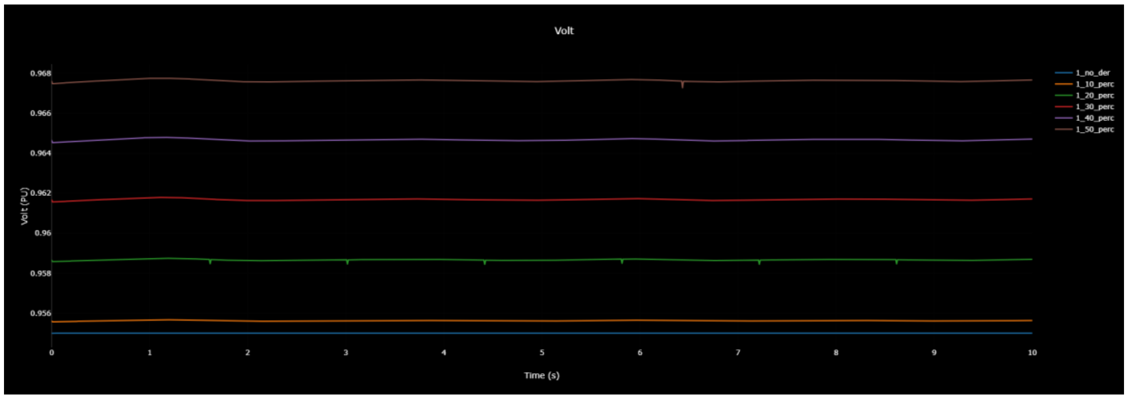


Figure 4: Voltage profile observed in bus 1 for the different cases considered. (Green: No DER, Red: 10% DER, Cyan: 20% DER, Black: 30% DER, Blue: 40% DER)

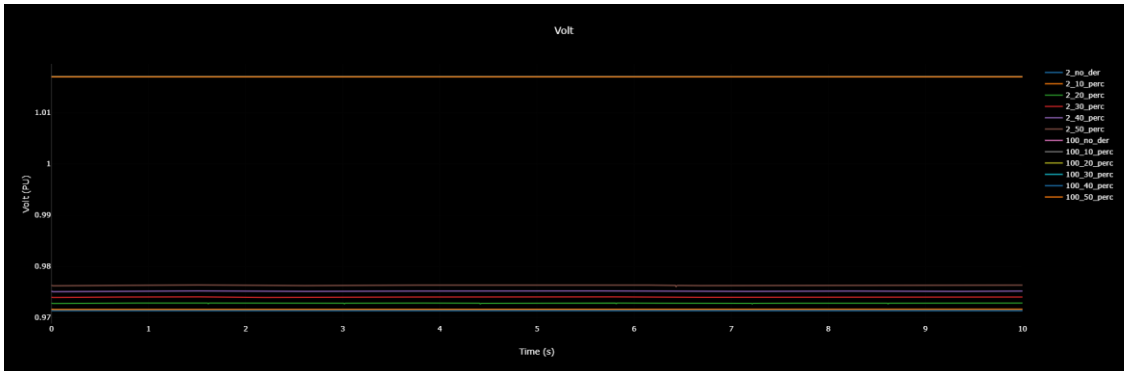


Figure 5: Voltage profile observed in bus 2 and bus 100 for the different cases considered. (Green: No DER, Red: 10% DER, Cyan: 20% DER, Black: 30% DER, Blue: 40% DER)

Figure 4 shows the voltage plot for bus 1 for the different cases considered. It can be observed that the with the DER added in the system, the bus voltage settles at a higher steady state voltage and it takes almost 4 secs for the system to reach a steady state. Similar settling time were observed in the buses nearby the DER connected buses as shown in Figure 5, but not observed in electrically distant buses as shown in Figure 5.

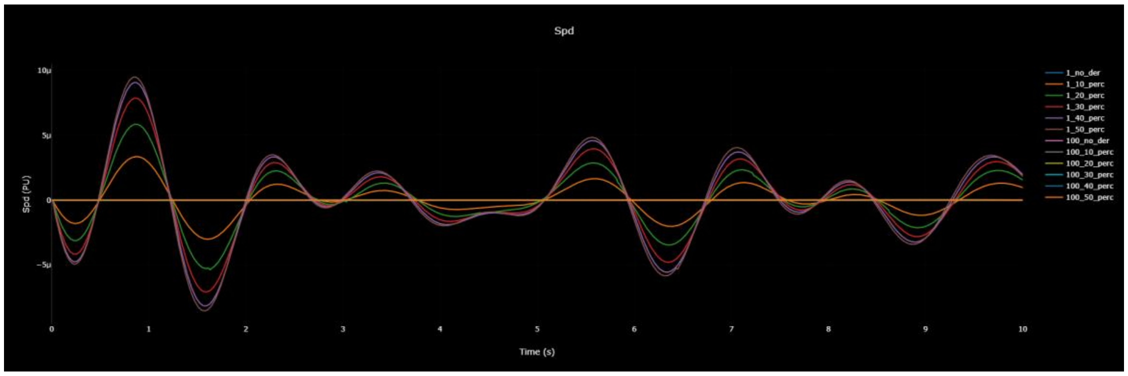


Figure 6: Frequency plot for the generators at bus 1 and bus 100 for different levels of DER penetration.

Figure 6 shows the generator speed profile for different levels of DER penetration at bus 1 and bus 100. It can be observed that the system frequency is initialized at exactly 60 Hz and stays constant at 60 Hz throughout the simulation. It can be observed that the generator speed has few oscillations that dies down slowly over time. The oscillations are due to the mismatch in the calculation of initial conditions for load and generation.

Please note that the tool takes few simulation seconds for the system to reach a steady state solution for the dynamic cases when using detailed DER. The developers are working on the initialization of system dynamic states so as to obtain a steady state solution from time t=0+. So for the current version of the tool, to study the system dynamics change of operating points and disturbances are applied only after the system reaches a certain steady state threshold i.e. at least 0.5 seconds.

Installation

The stable release of TDcoSim can be installed from PyPI. The version under development can be installed from GitHub. The commands to be executed in the Windows command line interface for both options are included below:

You can install tdcosim by running the following command on command line.

```
pip install git+https://github.com/tdcosim/TDcoSim.git@master
```

In the event you do not have git installed on your machine, you can alternatively run the following from command line.

```
pip install https://github.com/tdcosim/TDcoSim/archive/master.zip
```

After installation open a new command prompt and run the following to set psse path,

```
tdcosim setconfig -p "path\to\psse_installation"
```

For example, something similar to:

```
tdcosim setconfig -p "C:\Program Files\PTI\PSSE35\35.0\PSSPY37"
```

Note: Git needs to installed (incase it is not already available) before TDcoSim can be installed.

Post Installation

Get info about tdcosim

```
tdcosim describe
```

Test if tdcosim example simulations (dynamics and QSTS) are working,

```
tdcosim test
```

Dependencies:

The packages listed below must be installed separately:

Required

- Python version = 2.7.5 for PSS®E, version = 33.3.0, version >= 3.5 for PSS®E, version = 35.0.0
- Power system simulator: PSS®E, version = 33.3.0 or PSS®E, version = 35.0.0
- Distribution system simulator: OpenDSS, version >= 8.6.1.1
- DER simulator: Solar PV-DER simulation utility, version >= 0.5.1

- Python packages for basic functionalities: SciPy, Numpy, Matlplotlib, Pywin32, XlsxWriter, Psutil
- Python packages for visualization: Dash, Plotly

Note: Either demo (limited to 50 buses) or full version of PSS/E can be used. OpenDSS is an open source software and can be installed for free.

Note: PSS/E 35.0.0 and above support will support both 32 and 64 bit versions of Python. ***

Optional

- For using the high performance ODE solver diffeqpy, version >= 1.1.0
  - Install tdcosim with the diffeqpy flag as shown below.

    ```
    pip install tdcosim[diffeqpy]
    ```

  - Download and install Julia interpreter: Julia, version >= 1.5
  - Add Julia to system PATH environment variables as shown here (Only for Windows OS).

Note: If co-simulations involving more than 10 detailed DER models are needed to be run, it is recommended to select *diffeqpy* as the ODE solver.

Note: All Python packages can be installed with *pip* (e.g. *pip install scipy*)

System Requirements

Minimum requirements:

- OS: Windows 10
- Processor: Intel Core i7, 2.6 GHz, 2 cores
- Memory (RAM): 16.0 GB

Recommended requirements:

- OS: Windows
- Processor: Intel Core i9, 4.4 GHz, 8 cores
- Memory (RAM): 64.0 GB

Note: Capability to run on a cluster will be added in the future.

Note: Solution time is influenced by the number of logical cores in CPU.

References

1. A combined transmission and distribution system co-simulation framework for assessing the impact of Volt/VAR control on transmission system
2. Load Model Parameter Estimation by Transmission-Distribution Co-Simulation
3. Dynamic Modeling of Solar PV Systems for Distribution System Stability Analysis

Fast DER and Detailed DER parameters

Detailed DER and Fast DER are two model types available to the user (through the **config** file). Detailed DER will use a Dynamic Phasor models while the Fast DER will use Dynamic current inject models. Since the modeling approaches are different, both the model use different set of parameters.

Fast DER

The Fast DER is based on the DER_A model developed by EPRI and made available through commercial simulation software like PSS/E and PSLF. Note that the user need to populate the Fast DER parameters. They are automatically populated from the default values which can be accessed here.

- **Circuit parameters**
  - *xf (float):* Voltage source reactance.
- **Reference values**
  - *Pref (float):* Active power reference value.
  - *Qref (float):* Reactive power reference value.
  - *vref (float):* Voltage magnitude reference setpoint.
  - *vref_ang (float):* Voltage angle reference setpoint.
- **Time constants**
  - *Trv (float):* Voltage transducer time constant.
  - *Tfq (float):* Frequency transducer time constant.
  - *Tg (float):* Current (active and reactive) dynamics time constant.

  - *Tpord (float):* Active power dynamics time constant.
  - *Tiq (float):* Reactive current command time constant.
  - *Tid (float):* Active current command time constant.
- **Thresholds**
  - *Qmax (float):* Maximum reactive power setpoint.
  - *Qmax (float):* Maximum reactive power setpoint.
  - *Qmin (float):* Minimum reactive power setpoint.
  - *Pmax (float):* Maximum active power setpoint.
  - *Pmin (float):* Minimum active power setpoint.
  - *Imax (float):* Maximum allowed converter current.
  - *Id_max (float):* Maximum allowed active current.
  - *Id_min (float):* Minimum allowed active current.
  - *Id_min (float):* Minimum allowed active current.
  - *db_dw_up/db_dw_down (float):* Dead band in active power - frequency control.
  - *db_v_up/db_v_down (float):* Dead band in voltage - reactive power contol.
- **Controller gains**

- *Kp (float):* **Need to be added**
- *Ki (float):* **Need to be added**
- *Kpp/Kip (float):* Active power controller propotional/integral gain.
- *Kpq/Kiq (float):* Reactive power controller propotional/integral gain.
- *Kfv (float):* **Need to be added**

Detailed DER

The Detailed DER is based on the Dynamic Phasor modell developed by ANL. It is available through the pvder Python package. The model parameters for detailed DER are described here.

Comparing DER parameters

Parameter	Fast DER	Detailed DER
Rated power	Pmax	Srated, Np, Ns
Rated voltage	vref	Vrmsrated
Rated current	Imax	Ioverload
Current controller gains		Kp_GCC,Ki_GCC
Active power controller gains	Kpp/Kip	Kp_P,Ki_P,Kp_DC,Ki_DC
Reactive power controller gains	Kpq/Kiq	Kp_Q,Ki_Q

Composite model parameters

If the composite model is connected to a T node, it is parameterized using the default values found [here] (https://github.com/tdcosim/TDcoSim/blob/v2_test/config/composite_load_model_rating.json). Note that the default values can be edited by the user.

DER_A model parameters

If the DER_A model is connected to a T node, it is parameterized using the default values found [here] (https://github.com/tdcosim/TDcoSim/blob/v2_test/config/dera_rating.json). Note that the default values can be edited by the user.