
PV-DER Simulation Utility

Release 0.0.1

Sep 15, 2021

Contents:

1	PV-DER package	3
1.1	DER models	3
1.2	DER Simulation	13
2	License	19
3	Indices and tables	21
	Python Module Index	23
	Index	25

This is a reference for all the classes and methods available in PV-DER Simulation Utility.

1.1 DER models

1.1.1 pvder.DER_components_three_phase

Three phase PV-DER components.

```
class pvder.DER_components_three_phase.SolarPVDERThreePhase (events,          con-
                                                              figFile=None,
                                                              **kwargs)
```

Bases: pvder.DER_components.PVModule, pvder.DER_components.SolarPVDER

Class for describing a Solar Photo-voltaic Distributed Energy Resource consisting of panel, converters, and control systems. Attributes: count (int): Number of instances of *SolarPVDERThreePhase*. n_ODE (int): Number of ODE's.

Creates an instance of *SolarPV_DER_ThreePhase*. :param events: An instance of *SimulationEvents*. :type events: *SimulationEvents* :param gridModel: An instance of 'Gridl'(only need to be supplied for stand alone simulation). :type gridModel: *Grid* :param powerRating: A scalar specifying the rated power (VA) of the DER. :type powerRating: float :param VrmsRating: A scalar specifying the rated RMS L-G voltage (V) of the DER. :type VrmsRating: float :param ia0,xa0,ua0: Initial value of inverter states in p.u. of the DER instance. :type ia0,xa0,ua0: complex :param xDC0,xQ0,xPLL0,wte0: Initial value of inverter states in the DER instance. :type xDC0,xQ0,xPLL0,wte0: float :param gridVolatePhaseA,gridVolatePhaseA,gridVolatePhaseA: Initial voltage phasor (V) at PCC - LV side from external program (only need to be supplied if model is not stand alone). :type gridVolatePhaseA,gridVolatePhaseA,gridVolatePhaseA: float :param standAlone: Specify if the DER instance is a stand alone simulation or part of a larger simulation. :type standAlone: bool :param STEADY_STATE_INITIALIZATION: Specify whether states in the DER instance will be initialized to steady state values. :type STEADY_STATE_INITIALIZATION: bool :param allow_unbalanced_m: Allow duty cycles to take on unbalanced values during initialization (default: False). :type allow_unbalanced_m: bool :param derConfig: Configuration parameters that may be supplied from an external program. :type derConfig: dict :param identifier: An identifier that can be used to name the instance (default: None). :type identifier: str

Raises

- `ValueError` – If parameters corresponding to *Sinverter_rated* are not available.

- `ValueError` – If rated DC link voltage is not sufficient.

`Irms_calc()`
Inverter current - RMS

`ODE_model(y, t)`
Derivatives for the equation.

`S_G_calc()`
Power absorbed/produced by grid voltage source.

`S_PCC_calc()`
Power output at PCC LV side

`S_calc()`
Inverter apparent power output

`S_load1_calc()`
Power absorbed by load at PCC LV side.

`Vabrms_calc()`
PCC LV side voltage - line to line RMS

`Vrms_calc()`
PCC LV side voltage - RMS

`Vrms_min_calc()`
PCC LV side voltage - RMS

`Vtabrms_calc()`
Inverter terminal voltage - line to line RMS

`Vtrms_calc()`
Inverter terminal voltage - RMS

`jac_ODE_model(y, t)`
Jacobian for the system of ODE's.

`update_RMS()`
Update RMS voltages.

`update_inverter_frequency(t)`
Update d-q quantities.

`update_inverter_states(ia, xa, ua, ib, xb, ub, ic, xc, uc, Vdc, xDC, xQ, xPLL, wte)`
Update inverter states

`update_iref(t)`
Update reference reference current.

`update_power()`
Update RMS voltages.

`update_voltages()`
Update voltages.

`y0`
List of initial states

1.1.2 `pvder.DER_components_single_phase`

Single phase PV-DER code.


```
class pvder.DER_components_single_phase.SolarPVDERSinglePhase(events, con-  

figFile=None,  

**kwargs)
```

Bases: `pvder.DER_components.PVModule`, `pvder.DER_components.SolarPVDER`

Class for describing a Solar Photo-voltaic Distributed Energy Resource consisting of panel, converters, and control systems.

count

Number of instances of *SolarPVDERSinglePhase*.

Type int

n_ODE

Number of ODE's.

Type int

Creates an instance of *SolarPV_DER_SinglePhase*. Args: *events* (*SimulationEvents*): An instance of *SimulationEvents*. *gridModel* (*Grid*): An instance of *Grid* (only need to be supplied for stand alone simulation). *powerRating* (float): A scalar specifying the rated power (VA) of the DER. *VrmsRating* (float): A scalar specifying the rated RMS L-G voltage (V) of the DER. *ia0,xa0,ua0* (complex): Initial value of inverter states in p.u. of the DER instance. *xDC0,xQ0,xPLL0,wte0* (float): Initial value of inverter states in the DER instance. *gridVolatePhaseA,gridVolatePhaseA,gridVolatePhaseA* (float): Initial voltage phasor (V) at PCC - LV side from external program (only need to be supplied if model is not stand alone). *standAlone* (bool): Specify if the DER instance is a stand alone simulation or part of a larger simulation. *steadyStateInitialization* (bool): Specify whether states in the DER instance will be initialized to steady state values. *allowUnbalancedM* (bool): Allow duty cycles to take on unbalanced values during initialization (default: False). *derConfig* (dict): Configuration parameters that may be supplied from an external program. *identifier* (str): An identifier that can be used to name the instance (default: None).

Raises: *ValueError*: If parameters corresponding to *Sinverter_rated* are not available. *ValueError*: If rated DC link voltage is not sufficient.

Irms_calc()

Inverter current - RMS

ODE_model (*y, t*)

System of ODE's defining the dynamic DER model. :param *y*: Initial conditions for the states.. :type *y*: list of float :param *t*: Simulation time in seconds. :type *t*: float

Returns Derivates for the system of ODE's.

Return type result (list of float)

S_G_calc()

Power absorbed/produced by grid voltage source.

S_PCC_calc()

Power output at PCC LV side

S_calc()

Inverter apparent power output

S_load1_calc()

Power absorbed by load at PCC LV side.

Vabrms_calc()

PCC LV side voltage - line to lineRMS

Vrms_calc()

PCC LV side voltage - RMS

Vtrms_calc ()
 Inverter terminal voltage -RMS

jac_ODE_model (y, t)
 Jacobian for the system of ODE's. :param y: Initial conditions for the states.. :type y: list of float :param t: Simulation time in seconds. :type t: float

Returns An array containing the elements of the Jacobian.

Return type result (array of float)

update_RMS ()
 Update RMS voltages.

update_inverter_frequency (t)
 Update inverter PLL frequency. :param t: Simulation time in seconds. :type t: float

update_inverter_states (ia, xa, ua, Vdc, xDC, xQ, xPLL, wte)
 Update inverter states :param ia: Inverter phase a current. :type ia: complex :param xa: Inverter controller state. :type xa: complex :param ua: Inverter controller state. :type ua: complex :param Vdc: DC link voltage. :type Vdc: float

update_iref (t)
 Update reference reference current.

update_power ()
 Update RMS voltages.

update_voltages ()
 Update voltages.

y0
 List of initial states

1.1.3 pvder.DER_check_and_initialize

Code for initializing and validating PV-DER model instances.

class pvder.DER_check_and_initialize.**PVDER_SetupUtilities**
 Bases: pvder.grid_components.BaseValues
 Utility class for error checking during model initialization.

attach_grid_model (DER_arguments)
 Attach a grid model to the PV-DER instance. :param grid_model: An instance of *GridModel*.

Returns Description of return value

Return type bool

check_DER_config (DER_config, DER_id)
 Check DER config.

check_circuit_parameters ()
 Method to check whether inverter circuit parameter's are feasible.

check_config_file (config_file, config_dict)
 Check whether DER config file contains necessary fields.

check_jacobian (t=0.0)
 Compare analytical and numerical Jacobian of the ODE model.

check_voltage()
Method to check whether inverter voltage ratings are feasible. :raises: ValueError – If any of the specified voltage ratings is infeasible.

circuit_parameters = {}

controller_gains = {}

creation_message()
Message after PV-DER instance was created.

initialize_DER_model()
Initialize DER ratings.

Parameters DER_arguments (dict) – Key word arguments.

Raises ValueError – If specified parameters corresponding to *parameter_ID* is not available.

initialize_Iac()
Initialize AC side currents.

initialize_Sinverter()
Initialize inverter power rating.

initialize_Vac()
Initialize AC side voltages.

initialize_Vdc()
Initialize DC side voltages.

initialize_basic_options()
Initialize basic options

initialize_basic_specs()
Initialize number of ODEs and phases

initialize_circuit_parameters()
Initialize C, Lf, and Rf parameters.

initialize_controller_gains()
Initialize controller settings.

initialize_derived_quantities()
Initialize quantities other than states.

initialize_grid_measurements (DER_arguments)
Initialize inverter states. :param gridVoltagePhaseA: Value of gridVoltagePhaseA :type gridVoltagePhaseA: complex :param gridVoltagePhaseB: Value of gridVoltagePhaseB :type gridVoltagePhaseB: complex :param gridVoltagePhaseC: Value of gridVoltagePhaseC :type gridVoltagePhaseC: complex :param gridFrequency: Value of gridFrequency :type gridFrequency: float

initialize_inverter_ratings()
Initialize inverter voltage and power ratings.

initialize_jacobian()
Create a Jacobian matrix with zero values.

initialize_states (DER_arguments)
Initialize inverter states.

Parameters

- **ia0 (float)** – Initial current
- **xa0 (float)** – Initial controller state

- `ua0 (float)` – Initial controller state

```
inverter_ratings = {}  
modify_DER_parameters (parameter_ID)  
    Modify the DER parameters to parameters corresponding to the given parameter ID. :param parameter_ID:  
    User specified parameter ID (can be None). :type parameter_ID: str  
  
module_parameters = {}  
power_error_calc (x)  
    Function for power.  
  
solver_spec = {'SLSQP': {'disp': True, 'ftol': 1e-10, 'maxiter': 10000}, 'nelder-mead'  
steady_state_calc ()  
    Find duty cycle and inverter current that minimize steady state error and return steady state values.  
  
steadystate_solver = 'SLSQP'  
steadystate_values = {}  
  
update_DER_config (DER_config, DER_parent_config, DER_arguments, DER_id, DER_parent_id)  
    Update PV-DER design.  
  
update_DER_parameter (DER_config, DER_parent_config, DER_arguments, DER_id,  
                        DER_parent_id, DER_component, DER_parameter)  
    Update DER config using both config file and arguments.
```

1.1.4 pvder.DER_utilities

Code containing utilities used by PV-DER model instances.

```
class pvder.DER_utilities.PVDER_ModelUtilities  
    Bases: pvder.grid_components.BaseValues, pvder.utility_classes.Utilities  
    Utility class for single phase and three phase PV-DER model.  
  
    DO_EXTRA_CALCULATIONS = False  
    MPPT_ENABLE = False  
    MPP_table ()  
        Method to output Vdc reference corresponding to MPP at different insolation levels values.  
    PRINT_INLINE = False  
    Qref_EXTERNAL = False  
    RAMP_ENABLE = False  
    RAMP_FLAG = False  
    S_PCCph_calc (vph, iph)  
        Inverter apparent power output - phase a/b/c  
    VERBOSE = False  
    Vdc_actual  
        Actual DC link voltage. :returns: DC link voltage in Volts. :rtype: float  
    Vdc_ref_counter = 0  
    Vdc_ref_list = []
```

Vdc_ref_ramp (*tstart*, *Vdc_ref_target*)
 Create a ramp signal for voltage reference that ramps at 1 V/s. :param tstart: A scalar specifying start time of ramp in seconds. :type tstart: float :param Vdc_ref_target: A scalar specifying target Vdc reference seconds in volts. :type Vdc_ref_target: float

Vdc_ref_total = 0

add_Vdc_ref (*t*, *Vdc_ref*)
 Add new solar event.

create_parameter_dict (*parameter_ID*)
 Create a DER mode.

del_Vdc_ref = 2.0

del_t_Vdc_ref = 0.5

get_Vdc_ref (*t*)
 Output Vdc reference.

get_default_parameter_ID ()
 Return default parameter ID.

get_iref_actual (*t*, *ia_ref_command*, *iaR_ref_previous*, *iaI_ref_previous*)
 Update reference reference current.

get_parameter_dictionary (*parameter_type*, *parameter_ID*, *SHOW_DICTIONARY=True*)
 Return parameter dictionary for specified parameter type and parameter ID. :param parameter_type: Specify type of parameter. :type parameter_type: str :param parameter_ID: Specify parameter ID or 'all'. :type parameter_ID: str :param SHOW_DICTIONARY: Print the dictionary. :type SHOW_DICTIONARY: bool

Returns Parameters and their values

Return type dict

get_ramp_limited_iref (*t*, *ia_ref_command*)
 Update reference reference current.

ia_ref_activepower_control ()
 Phase A current reference for constant Vdc

ia_ref_calc ()
 Phase A current reference

ib_ref_activepower_control ()
 Phase B current reference for constant Vdc

ib_ref_calc ()
 Phase B current reference

ic_ref_activepower_control ()
 Phase C current reference for constant Vdc

ic_ref_calc ()
 Phase C current reference

initialize_parameter_dict (*parameter_ID*, *source_parameter_ID*)
 Initialize a new parameter dictionary with values from an existing parameter dictionary.

iphload1_calc (*vph*)
 Current consumed by load connected at PCC LV side - Phase A/B/C.

load_parameter_dictionary (*file_name*)
 Load parameter dictionary from saved file.

```

m_limit = 1.0

ma
    Phase A duty cycle. :returns: Duty cycle. :rtype: complex

mb
    Phase B duty cycle. :returns: Duty cycle. :rtype: complex

mc
    Phase C duty cycle. :returns: Duty cycle. :rtype: complex

n_steps = 0

ramp_del_t = 0.5

ramp_list = []

reset_reference_counters()
    Reset counter for reference change events.

save_parameter_dictionary(parameter_ID,                                save_format='pickle',
                             SHOW_DICTIONARY=False)
    Save parameter dictionary.

set_Vdc_ref()
    Return the correct Vdc reference voltage.

show_PV_DER_parameters(parameter_type='inverter_ratings')
    Display rated values. :param parameter_type: A string ('in-
    verter_ratings','controller_gains','circuit_parameters') specifying the parameter to be displayed.
    :type parameter_type: str

show_PV_DER_states(quantity='voltage')
    Display values of states in the DER model quantities. Arguments
    quantity: A string ('voltage','current','power','duty cycle') specifying the electrical quantity to
    be displayed.

show_parameter_dictionaries()
    Show all parameter dictionary types and their ID's.

show_parameter_types()
    Show all parameters within all parameter dictionary types.

show_references()
    Print references.

update_Ppv(t)
    Update PV module power output based on solar events and DC link voltage.

update_Qref(t)
    Update reactive power set-point.

update_Vdc_ref(t)
    Update DC link voltage reference.

update_Zload1(t)
    Update load impedance at PCC-LV side.

update_parameter_dict(parameter_ID, parameter_type, parameter_dict)
    Update parameters.

use_frequency_estimate = False

```

```

va_calc()
    PCC - LV side - Phase A

validate_model (PRINT_ERROR=True)
    Compare error between RMS quantities and Phasor quantities.

vb_calc()
    PCC - LV side - Phase B

vc_calc()
    PCC - LV side - Phase C

vta_calc()
    Inverter terminal voltage - Phase A

vtb_calc()
    Inverter terminal voltage - Phase B

vtc_calc()
    Inverter terminal voltage - Phase C

we_calc()
    Calculate inverter frequency from PLL.

wgrid_calc (t)
    Frequency of grid voltage source.

wgrid_estimate (t)
    Estimate frequency from phasor angle.

wgrid_estimate_calc (t, phi_new, phi_previous)
    Estimate frequency.

```

1.1.5 pvder.DER_features

Code for features inside PV-DER model instances.

```

class pvder.DER_features.PVDER_SmartFeatures
    Bases: object

    Class for describing smart inverter inverter features of PV-DER.

    DER_CONNECTED = True

    DER_MOMENTARY_CESSATION = False

    DER_TRIP = False

    DER_disconnect()
        Function to disconnect PV-DER from grid.

    DER_disconnect_logic (t)
        Logic for disconnecting/deenergizing DER from grid.

    DER_reconnect_logic (t)
        Logic used to decide reconnection.

    FRT (t)
        Frequency ride through and trip logic.

    FRT_INSTANTANEOUS_TRIP

    FRT_initialize()
        Initialize LFRT and HFRT settings.

```

HVRT (*t*)
Function to implement HVRT ridethrough and trip logic.

LVRT (*t*)
Function to implement LVRT ridethrough and trip logic.

Q_Volt_VAR_absorb (*Vrms_measured*)
Calculate reactive power for Volt-VAR control.

Q_Volt_VAR_inject (*Vrms_measured*)
Calculate reactive power for Volt-VAR control.

Qlimit_calc ()
Calculate maximum Q reference.

RT_initialize ()
Initialize VRT and FRT settings.

VOLT_VAR_ENABLE = False

VOLT_VAR_FLAG = False

VOLT_WATT_ENABLE = False

VOLT_WATT_FLAG = False

VRT_initialize ()
Initialize LVRT and HVRT settings.

V_threshold_high_limit = 1.5

Volt_VAR_absorb_range (*Vrms_measured, del_V=0.0*)
Check if voltage in volt-VAR operating range.

Volt_VAR_inject_range (*Vrms_measured, del_V=0.0*)
Check if voltage in volt-VAR operating range.

Volt_VAR_logic (*t*)
Volt-VAR.

check_HFRT_settings ()
Sanity check for HFRT settings.

check_LFRT_settings ()
Sanity check for LFRT settings.

check_RT_config ()
Check whether the config file is good.

check_VRT_settings ()
Sanity check for VRT settings.

check_anomaly ()
Check if voltage anomaly was detected.

disconnect_or_reconnect (*t*)
Check flags and either disconnect or reconnect DER.

f_ref = 60.0

get_Vrms_measured ()
Get Vrms measurement

initialize_Volt_VAR ()
Initialize the Volt-VAR controller settings.


```
DEBUG_CONTROLLERS = False
DEBUG_CURRENTS = False
DEBUG_PLL = False
DEBUG_POWER = False
DEBUG_SIMULATION = False
DEBUG_SOLVER = False
DEBUG_VOLTAGES = False

ODE_model(y, t)
    Combine all derivatives.

check_jac_availability()
    Check if Jacobian matrix is available.

collect_full_trajectory(solution)
    Collect full solution from solver.

collect_last_states()
    Collect states at last time step.

collect_solution(solution, t=None)
    Collect solution.

collect_states(solution)
    Collect states from ode solution.

count = 0

debug_simulation(t)
    Print to terminal for debugging.

get_trajectories()
    Return trajectories as a dictionary.

initialize_y0_t()
    Initialize y0_t.

invert_arrays()
    Inverter arrays before usage by plots.

jac_ODE_model(y, t)
    Combine all derivatives.

jac_list = ['SolarPVDERThreePhase', 'SolarPVDESinglePhase', 'SolarPVDERThreePhaseBalanc

reset_stored_trajectories()
    Reset for plotting.

run_simulation(gridVoltagePhaseA=None, gridVoltagePhaseB=None, gridVoltagePhaseC=None,
               y0=None, t=None)
    Call the ODE solver and collect states.

show_simulation_time()
    Show simulation time.

tInc = 0.001

tStart = 0.0

t_calc()
    Vector of time steps for simulation
```

```

time_series_PCC_HV_side_voltage()
    Calculate time series PCC voltage.

time_series_PCC_LV_side_voltage()
    Calculate time series PCC voltage.

time_series_PLL()
    Calculate time series PLL and d-q quantities.

time_series_Ppv()
    Calculate time series Solar PV power output.

time_series_RMS()
    Calculate time series RMS quantities.

time_series_S()
    Calculate time series apparent power.

time_series_Zload1(tOverride=None)
    Calculate time series load impedance.

time_series_duty_cycle()
    Calculate time series PCC voltage.

time_series_inv_terminal_voltage()
    Calculate time series inverter terminal voltage.

time_series_phase_angle()
    Calculate time series phase angle between grid and inverter voltage.

time_series_power_transfer()
    Calculate time series power transfer between grid and PCC.

time_series_standalone_grid()
    Time series grid voltage and frequency for standalone model.

update_grid_measurements(gridVoltagePhaseA, gridVoltagePhaseB, gridVoltagePhaseC)
    Update grid voltage and frequency in non-standalone model. :param gridVoltagePhaseA: Value of
    gridVoltagePhaseA :type gridVoltagePhaseA: complex :param gridVoltagePhaseB: Value of gridVoltagePhaseB :type gridVoltagePhaseB: complex :param gridVoltagePhaseC: Value of gridVoltagePhaseC :type gridVoltagePhaseC: complex

y0
    Combine all initial conditions from solution.

```

1.2.2 pvder.simulation_events

Manage simulation events.

```

class pvder.simulation_events.SimulationEvents(events_spec=None, SO-
                                                LAR_EVENT_ENABLE=True,
                                                GRID_EVENT_ENABLE=True,
                                                LOAD_EVENT_ENABLE=True, ver-
                                                bosity='INFO', identifier='')

```

Bases: `pvder.utility_classes.Utilities`

Utility class for events.

Creates an instance of *SimulationEvents*. :param SOLAR_EVENT_ENABLE: A boolean to enable solar insolation events. :param GRID_EVENT_ENABLE: A boolean to enable grid voltage or frequency events. :param LOAD_EVENT_ENABLE: A boolean to enable load change events at PCC-LV side.

Tactual_default = 298.15

Zload1_actual_default = (10000000+0j)

add_grid_event (*T*, *Vgrid=1.0*, *Vgrid_angle=0.0*, *fgrid=60.0*)
Add new grid event. :param *T*: A scalar specifying start time of grid event in seconds. :type *T*: float :param *Vgrid*: A scalar specifying grid voltage magnitude in fraction. :type *Vgrid*: float :param *Vgrid_angle*: A scalar specifying grid voltage angle in radians. :type *Vgrid_angle*: float :param *fgrid*: A scalar specifying grid frequency in Hz. :type *fgrid*: float

add_load_event (*T*, *Zload1_actual=(10000000+0j)*)
Add new load event. :param *T*: A scalar specifying start time of load event in seconds. :param *Zload1_actual*: A complex scalar specifying load in ohm.

add_solar_event (*T*, *Sinsol=100.0*, *Tactual=298.15*)
Add new solar event. :param *T*: A scalar specifying start time of solar event in seconds. :type *T*: float :param *Sinsol*: A scalar specifying solar insolation in percentage. :type *Sinsol*: float :param *Tactual*: A scalar specifying module temperature in Kelvin. :type *Tactual*: float

count = 0

create_random_events (*t_event_start*, *t_event_end*, *t_event_step*, *events_type=['insolation', 'voltage']*)
Create random events of specified types.

create_random_insolation_events (*t_event*)
Create random voltage event at specified time.

create_random_voltage_events (*t_event*)
Create random voltage event at specified time.

del_t_event = 0.001

grid_events (*t*)
Generate grid event during simulation at specified time. :param *t*: A scalar specifying the time (s). :type *t*: float

insolation_ramp (*tstart*, *tstop*, *Sinsol_target*, *tstep=0.25*)
Create a ramp signal. :param *tstart*: A scalar specifying start time of solar insolation event in seconds. :param *tstop*: A scalar specifying stop time of solar insolation event in seconds. :param *Sinsol_target*: A scalar specifying target solar insolation at 'tstop' in percentage. :param *tstep*: A scalar specifying time step size.

load_events (*t*)
Generate load event at PCC LV side during simulation. :param *t*: A scalar specifying the time (s). :type *t*: float

override_angle = True

remove_grid_event (*T*)
Remove grid event at 'T'.

remove_load_event (*T=None*, *REMOVE_ALL=False*)
Remove solar event at 'T'

remove_solar_event (*T=None*, *REMOVE_ALL=False*)
Remove solar event at 'T'.

reset_event_counters ()
Reset event counts.

show_events ()
Print all the simulation events.

simulation_events_list

List of all simulation events.

solar_events (*t*)

Generate solar event during simulation at specified time. :param t: A scalar specifying the time (s). :type t: float

update_event_totals ()

Update event counts.

update_events_spec (*events_spec*)

Update the _events_spec dictionary.

voltage_ramp (*tstart, tstop, vg_target, tstep=0.5*)

Create a ramp signal for grid voltage. :param tstart: A scalar specifying start time of grid voltage event in seconds. :param tstop: A scalar specifying stop time of grid voltage event in seconds. :param vg_target: A scalar specifying target grid voltage at 'tstop' in fraction. :param tstep: A scalar specifying time step size (optional).

1.2.3 pvder.simulation_utilities

Manage simulation results and ODE solver.

```
class pvder.simulation_utilities.SimulationResults (simulation, figure_index=1,  
                                                    PER_UNIT=True, font_size=18,  
                                                    PLOT_TITLE=True, SO-  
                                                    LUTION_TIME=True, ver-  
                                                    bosity='INFO', identifier=None)
```

Bases: pvder.utility_classes.Utilities

Utility class for simulation results.

Creates an instance of *SimulationResults*. Args: simulation: An instance of *GridSimulation*. figure_index: An integer specifying the figure index. font_size: An integer specifying the font size to be used withing plots. PLOT_TITLE: A boolean specifying whether the title will be displayed in plots. verbosity: A string specifying the verbosity level (DEBUG,INFO,WARNING,ERROR).

SAVE_PLOT_JPEG = False

SAVE_PLOT_SVG = False

available_plot_types = ['power', 'active_power', 'active_power_Ppv_Pac_PCC', 'active_p

change_units ()

Change units from per unit to S.I. or vice versa.

compare_with_external (*external_time_values, external_plot_values, external_plot_legends,*
 plot_type='power')

Compare with external simulation.

count = 0

figure_DPI = 1200

group_quantities_for_plotting (*plot_type*)

Plot power from simulation. Args: plot_type (string): The quantities to be plotted.

Raises ValueError – If *plot_type* is not available.

parameters = {'figure': {'height': 8, 'width': 8}}

plot_DER_simulation (*plot_type='power'*)

Plot desired electrical quantity from simulation. Args: *plot_type* (str): Specify type of plot (to see available plot types use `SimulationResults.available_plot_types`).

plot_multiple (*time_values, plot_values, legends, plot_title, y_labels*)

Function to plot multiple time series on same plot.

save_plot (*plot_object, plot_name='results'*)

Save the plots.

class `pvder.simulation_utilities.SimulationUtilities`

Bases: `object`

Utility class for dynamic simulations.

ODE_model_ode (*t, y*)

” Combine all derivatives when using ode method

call_ODE_solver (*derivatives, jacobian, y, t*)

Call the SciPy ODE solver.

call_ode_solver ()

Use the SciPy ode solver.

call_odeint_solver (*derivatives, jacobian, y, t*)

Use the SciPy odeint solver.

check_simulation (*infodict, t*)

Check whether the ODE solver failed at any time step.

initialize_solver (*solver_type, t0=0.0*)

Initialize an integrator.

jac_ODE_model_ode (*t, y*)

Combine all derivatives.

max_steps = 1000

solver_list = ['odeint', 'ode-vode-bdf']

CHAPTER 2

License

Copyright © 2019, UChicago Argonne, LLC All Rights Reserved Photovoltaic Distributed Energy Resource (PV-DER) Simulation Utility Argonne National Laboratory OPEN SOURCE LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

p

- `pvder.DER_check_and_initialize`, [6](#)
- `pvder.DER_components_single_phase`, [4](#)
- `pvder.DER_components_three_phase`, [3](#)
- `pvder.DER_features`, [11](#)
- `pvder.DER_utilities`, [8](#)
- `pvder.dynamic_simulation`, [13](#)
- `pvder.simulation_events`, [15](#)
- `pvder.simulation_utilities`, [17](#)

A

`add_grid_event()` (`pvder.simulation_events.SimulationEvents`
method), 16
`add_load_event()` (`pvder.simulation_events.SimulationEvents`
method), 16
`add_solar_event()`
(`pvder.simulation_events.SimulationEvents`
method), 16
`add_Vdc_ref()` (`pvder.DER_utilities.PVDER_ModelUtilities`
method), 9
`attach_grid_model()`
(`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`available_plot_types`
(`pvder.simulation_utilities.SimulationResults`
attribute), 17

C

`call_ODE_solver()`
(`pvder.simulation_utilities.SimulationUtilities`
method), 18
`call_ode_solver()`
(`pvder.simulation_utilities.SimulationUtilities`
method), 18
`call_odeint_solver()`
(`pvder.simulation_utilities.SimulationUtilities`
method), 18
`change_units()` (`pvder.simulation_utilities.SimulationResults`
method), 17
`check_anomaly()` (`pvder.DER_features.PVDER_SmartFeatures`
method), 12
`check_circuit_parameters()`
(`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`check_config_file()`
(`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`check_DER_config()`
(`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`check_HFRT_settings()`
(`pvder.DER_features.PVDER_SmartFeatures`
method), 12
`check_jac_availability()`
(`pvder.dynamic_simulation.DynamicSimulation`
method), 14
`check_jacobian()` (`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`check_LFRT_settings()`
(`pvder.DER_features.PVDER_SmartFeatures`
method), 12
`check_simulation()`
(`pvder.simulation_utilities.SimulationUtilities`
method), 18
`check_voltage()` (`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
method), 6
`check_VRT_settings()`
(`pvder.DER_features.PVDER_SmartFeatures`
method), 12
`circuit_parameters`
(`pvder.DER_check_and_initialize.PVDER_SetupUtilities`
attribute), 7
`collect_full_trajectory()`
(`pvder.dynamic_simulation.DynamicSimulation`
method), 14
`collect_last_states()`
(`pvder.dynamic_simulation.DynamicSimulation`
method), 14
`collect_solution()`
(`pvder.dynamic_simulation.DynamicSimulation`
method), 14
`collect_states()` (`pvder.dynamic_simulation.DynamicSimulation`
method), 14
`compare_with_external()`
(`pvder.simulation_utilities.SimulationResults`
method), 17

controller_gains (pvder.DER_check_and_initialize.PVDER_SetupUtilities attribute), 7	connect_logic() (pvder.DER_features.PVDER_SmartFeatures method), 11
count (pvder.DER_components_single_phase.SolarPVDER_SinglePhase attribute), 5	count (pvder.DER_features.PVDER_SmartFeatures method), 11
count (pvder.dynamic_simulation.DynamicSimulation attribute), 14	DER_MOMENTARY_CESSATION
count (pvder.simulation_events.SimulationEvents attribute), 16	count (pvder.DER_features.PVDER_SmartFeatures attribute), 11
count (pvder.simulation_utilities.SimulationResults attribute), 17	DER_reconnect_logic() (pvder.DER_features.PVDER_SmartFeatures method), 11
create_parameter_dict() (pvder.DER_utilities.PVDER_ModelUtilities method), 9	DER_TRIP (pvder.DER_features.PVDER_SmartFeatures attribute), 11
create_random_events() (pvder.simulation_events.SimulationEvents method), 16	disconnect_or_reconnect() (pvder.DER_features.PVDER_SmartFeatures method), 12
create_random_insolation_events() (pvder.simulation_events.SimulationEvents method), 16	DO_EXTRA_CALCULATIONS (pvder.DER_utilities.PVDER_ModelUtilities attribute), 8
create_random_voltage_events() (pvder.simulation_events.SimulationEvents method), 16	DynamicSimulation (class in pvder.dynamic_simulation), 13
creation_message() (pvder.DER_check_and_initialize.PVDER_SetupUtilities method), 7	
D	
DEBUG_CONTROLLERS (pvder.dynamic_simulation.DynamicSimulation attribute), 13	figure_DPI (pvder.simulation_utilities.SimulationResults attribute), 17
DEBUG_CURRENTS (pvder.dynamic_simulation.DynamicSimulation attribute), 14	FRT() (pvder.DER_features.PVDER_SmartFeatures method), 11
DEBUG_PLL (pvder.dynamic_simulation.DynamicSimulation attribute), 14	FRT_initialize() (pvder.DER_features.PVDER_SmartFeatures method), 11
DEBUG_POWER (pvder.dynamic_simulation.DynamicSimulation attribute), 14	FRT_INSTANTANEOUS_TRIP (pvder.DER_features.PVDER_SmartFeatures attribute), 11
DEBUG_SIMULATION (pvder.dynamic_simulation.DynamicSimulation attribute), 14	
debug_simulation() (pvder.dynamic_simulation.DynamicSimulation method), 14	get_default_parameter_ID() (pvder.DER_utilities.PVDER_ModelUtilities method), 9
DEBUG_SOLVER (pvder.dynamic_simulation.DynamicSimulation attribute), 14	get_iref_actual() (pvder.DER_utilities.PVDER_ModelUtilities method), 9
DEBUG_VOLTAGES (pvder.dynamic_simulation.DynamicSimulation attribute), 14	get_parameter_dictionary() (pvder.DER_utilities.PVDER_ModelUtilities method), 9
del_t_event (pvder.simulation_events.SimulationEvents attribute), 16	get_ramp_limited_iref() (pvder.DER_utilities.PVDER_ModelUtilities method), 9
del_t_Vdc_ref (pvder.DER_utilities.PVDER_ModelUtilities attribute), 9	get_trajectories() (pvder.dynamic_simulation.DynamicSimulation method), 14
del_Vdc_ref (pvder.DER_utilities.PVDER_ModelUtilities attribute), 9	get_Vdc_ref() (pvder.DER_utilities.PVDER_ModelUtilities method), 9
DER_CONNECTED (pvder.DER_features.PVDER_SmartFeatures attribute), 11	

F

G

`get_Vrms_measured()`
 (*pvder.DER_features.PVDER_SmartFeatures* *initialize_jacobian()*
 method), 12 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 7
`grid_events()` (*pvder.simulation_events.SimulationEvents* *initialize_parameter_dict()*
 method), 16 (*pvder.DER_utilities.PVDER_ModelUtilities*
 method), 9
`group_quantities_for_plotting()`
 (*pvder.simulation_utilities.SimulationResults* *initialize_Sinverter()*
 method), 17 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 7
H
`HVRT()` (*pvder.DER_features.PVDER_SmartFeatures* *initialize_solver()*
 method), 12 (*pvder.simulation_utilities.SimulationUtilities*
 method), 18
I
`ia_ref_activepower_control()`
 (*pvder.DER_utilities.PVDER_ModelUtilities* *initialize_states()*
 method), 9 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 7
`ia_ref_calc()` (*pvder.DER_utilities.PVDER_ModelUtilities* *initialize_Vac()* (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 9 *method*), 7
`ib_ref_activepower_control()`
 (*pvder.DER_utilities.PVDER_ModelUtilities* *initialize_Vdc()* (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 9 *method*), 7
`ib_ref_calc()` (*pvder.DER_utilities.PVDER_ModelUtilities* *initialize_Volt_VAR()*
 method), 9 (*pvder.DER_features.PVDER_SmartFeatures*
 method), 12
`ic_ref_activepower_control()`
 (*pvder.DER_utilities.PVDER_ModelUtilities* *initialize_y0_t()*
 method), 9 (*pvder.dynamic_simulation.DynamicSimulation*
 method), 14
`ic_ref_calc()` (*pvder.DER_utilities.PVDER_ModelUtilities* *insolation_ramp()*
 method), 9 (*pvder.simulation_events.SimulationEvents*
 method), 16
`initialize_basic_options()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *invert_arrays()* (*pvder.dynamic_simulation.DynamicSimulation*
 method), 7 *method*), 14
`initialize_basic_specs()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *inverter_ratings* (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*
 method), 7 *attribute*), 8
`initialize_circuit_parameters()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *inphoad1_calc()* (*pvder.DER_utilities.PVDER_ModelUtilities*
 method), 7 *method*), 9
`initialize_controller_gains()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *Irms_calc()* (*pvder.DER_components_single_phase.SolarPVDERSinglePhase*
 method), 7 *method*), 5
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *Irms_calc()* (*pvder.DER_components_three_phase.SolarPVDERThreePhase*
 method), 7 *method*), 4
J
`initialize_DER_model()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *jac_list* (*pvder.dynamic_simulation.DynamicSimulation*
 method), 7 *attribute*), 14
`initialize_derived_quantities()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *jac_ODE_model()* (*pvder.DER_components_single_phase.SolarPVDERSinglePhase*
 method), 7 *method*), 6
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *jac_ODE_model()* (*pvder.DER_components_three_phase.SolarPVDERThreePhase*
 method), 7 *method*), 4
`initialize_grid_measurements()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *jac_ODE_model()* (*pvder.dynamic_simulation.DynamicSimulation*
 method), 7 *method*), 14
`initialize_Iac()` (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* *jac_ODE_model_ode()*
 method), 7 (*pvder.simulation_utilities.SimulationUtilities*
 method), 18
`initialize_inverter_ratings()`
 (*pvder.DER_check_and_initialize.PVDER_SetupUtilities*

L

load_events() (pvder.simulation_events.SimulationEvents
method), 16

load_parameter_dictionary()
(pvder.DER_utilities.PVDER_ModelUtilities
method), 9

LVRT() (pvder.DER_features.PVDER_SmartFeatures
method), 12

M

m_limit (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 9

ma (pvder.DER_utilities.PVDER_ModelUtilities at-
tribute), 10

max_steps (pvder.simulation_utilities.SimulationUtilities
attribute), 18

mb (pvder.DER_utilities.PVDER_ModelUtilities at-
tribute), 10

mc (pvder.DER_utilities.PVDER_ModelUtilities at-
tribute), 10

modify_DER_parameters()
(pvder.DER_check_and_initialize.PVDER_SetupUtilities
method), 8

module_parameters
(pvder.DER_check_and_initialize.PVDER_SetupUtilities
attribute), 8

MPP_table() (pvder.DER_utilities.PVDER_ModelUtilities
method), 8

MPPT_ENABLE (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 8

N

n_ODE (pvder.DER_components_single_phase.SolarPVDERSinglePhase
attribute), 5

n_steps (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 10

O

ODE_model() (pvder.DER_components_single_phase.SolarPVDERSinglePhase
method), 5

ODE_model() (pvder.DER_components_three_phase.SolarPVDERThreePhase
method), 4

ODE_model() (pvder.dynamic_simulation.DynamicSimulation
method), 14

ODE_model_ode() (pvder.simulation_utilities.SimulationUtilities
method), 18

override_angle (pvder.simulation_events.SimulationEvents
attribute), 16

P

parameters (pvder.simulation_utilities.SimulationResults
attribute), 17

plot_DER_simulation()
(pvder.simulation_utilities.SimulationResults
method), 17

plot_multiple() (pvder.simulation_utilities.SimulationResults
method), 18

power_error_calc()
(pvder.DER_check_and_initialize.PVDER_SetupUtilities
method), 8

print_event() (pvder.DER_features.PVDER_SmartFeatures
method), 13

PRINT_INLINE (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 8

print_LVRT_events()
(pvder.DER_features.PVDER_SmartFeatures
method), 12

print_reconnect_events()
(pvder.DER_features.PVDER_SmartFeatures
method), 13

print_VRT_events()
(pvder.DER_features.PVDER_SmartFeatures
method), 13

pvder.DER_check_and_initialize (module), 6

pvder.DER_components_single_phase (mod-
ule), 4

pvder.DER_components_three_phase (mod-
ule), 3

pvder.DER_features (module), 11

pvder.DER_utilities (module), 8

pvder.dynamic_simulation (module), 13

pvder.simulation_events (module), 15

pvder.simulation_utilities (module), 17

PVDER_ModelUtilities (class in
pvder.DER_utilities), 8

PVDER_SetupUtilities (class in
pvder.DER_check_and_initialize), 6

PVDER_SmartFeatures (class in
pvder.DER_features), 11

Q

Q_volt_VAR_absorb()
(pvder.DER_features.PVDER_SmartFeatures
method), 12

Q_volt_VAR_inject()
(pvder.DER_features.PVDER_SmartFeatures
method), 12

qlimit_calc() (pvder.DER_features.PVDER_SmartFeatures
method), 12

Qref_EXTERNAL (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 8

R

ramp_del_t (pvder.DER_utilities.PVDER_ModelUtilities
attribute), 10

RAMP_ENABLE (*pvder.DER_utilities.PVDER_ModelUtilities* attribute), 8
 RAMP_FLAG (*pvder.DER_utilities.PVDER_ModelUtilities* attribute), 8
 ramp_list (*pvder.DER_utilities.PVDER_ModelUtilities* attribute), 10
 remove_grid_event () (*pvder.simulation_events.SimulationEvents* method), 16
 remove_load_event () (*pvder.simulation_events.SimulationEvents* method), 16
 remove_solar_event () (*pvder.simulation_events.SimulationEvents* method), 16
 reset_event_counters () (*pvder.simulation_events.SimulationEvents* method), 16
 reset_reference_counters () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 reset_stored_trajectories () (*pvder.dynamic_simulation.DynamicSimulation* method), 14
 RT_initialize () (*pvder.DER_features.PVDER_SmartFeatures* method), 12
 run_simulation () (*pvder.dynamic_simulation.DynamicSimulation* method), 14

S
 S_calc () (*pvder.DER_components_single_phase.SolarPVDERSinglePhase* method), 5
 S_calc () (*pvder.DER_components_three_phase.SolarPVDERThreePhase* method), 4
 S_G_calc () (*pvder.DER_components_single_phase.SolarPVDERSinglePhase* method), 5
 S_G_calc () (*pvder.DER_components_three_phase.SolarPVDERThreePhase* method), 4
 S_load1_calc () (*pvder.DER_components_single_phase.SolarPVDERSinglePhase* method), 5
 S_load1_calc () (*pvder.DER_components_three_phase.SolarPVDERThreePhase* method), 4
 S_PCC_calc () (*pvder.DER_components_single_phase.SolarPVDERSinglePhase* method), 5
 S_PCC_calc () (*pvder.DER_components_three_phase.SolarPVDERThreePhase* method), 4
 S_PCCph_calc () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 8
 save_parameter_dictionary () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 save_plot () (*pvder.simulation_utilities.SimulationResults* method), 18
 SAVE_PLOT_JPEG (*pvder.simulation_utilities.SimulationResults* attribute), 17
 SAVE_PLOT_SVG (*pvder.simulation_utilities.SimulationResults* attribute), 17
 set_Vdc_ref () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_events () (*pvder.simulation_events.SimulationEvents* method), 16
 show_parameter_dictionaries () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_parameter_types () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_PV_DER_parameters () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_PV_DER_states () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_references () (*pvder.DER_utilities.PVDER_ModelUtilities* method), 10
 show_RT_flags () (*pvder.DER_features.PVDER_SmartFeatures* method), 13
 show_RT_settings () (*pvder.DER_features.PVDER_SmartFeatures* method), 13
 show_simulation_time () (*pvder.dynamic_simulation.DynamicSimulation* method), 14
 simulation_events_list (*pvder.simulation_events.SimulationEvents* attribute), 16
 SPVDERSinglePhase (class in *pvder.simulation_events*), 15
 SPVDERThreePhase (class in *pvder.simulation_utilities*), 17
 SPVDERUtilities (class in *pvder.simulation_utilities*), 18
 SPVDERThreePhase (class in *pvder.DER_components_single_phase*), 4
 SPVDERThreePhase (class in *pvder.DER_components_three_phase*), 3
 solver_list (*pvder.simulation_utilities.SimulationUtilities* attribute), 18
 solver_spec (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* attribute), 8
 steady_state_calc () (*pvder.DER_check_and_initialize.PVDER_SetupUtilities* method), 8
 steadystate_solver

(pvder.DER_check_and_initialize.PVDER_SetupUtilities method), 15
 attribute), 8 tInc (pvder.dynamic_simulation.DynamicSimulation attribute), 14
 steadystate_values (pvder.DER_check_and_initialize.PVDER_SetupUtilities method), 14
 attribute), 8

T

t_calc() (pvder.dynamic_simulation.DynamicSimulation method), 14
 t_disconnect_low_limit (pvder.DER_features.PVDER_SmartFeatures attribute), 13
 t_reconnect_low_limit (pvder.DER_features.PVDER_SmartFeatures attribute), 13
 t_threshold_high_limit (pvder.DER_features.PVDER_SmartFeatures attribute), 13
 Tactual_default (pvder.simulation_events.SimulationEvents attribute), 15
 time_series_duty_cycle() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_inv_terminal_voltage() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_PCC_HV_side_voltage() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_PCC_LV_side_voltage() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_phase_angle() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_PLL() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_power_transfer() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_Ppv() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_RMS() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_S() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_standalone_grid() (pvder.dynamic_simulation.DynamicSimulation method), 15
 time_series_Zload1() (pvder.dynamic_simulation.DynamicSimulation method), 15

U

update_DER_config() (pvder.DER_check_and_initialize.PVDER_SetupUtilities method), 8
 update_DER_parameter() (pvder.DER_check_and_initialize.PVDER_SetupUtilities method), 8
 update_event_totals() (pvder.simulation_events.SimulationEvents method), 17
 update_events_spec() (pvder.simulation_events.SimulationEvents method), 17
 update_grid_measurements() (pvder.dynamic_simulation.DynamicSimulation method), 15
 update_inverter_frequency() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6
 update_inverter_frequency() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_inverter_states() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6
 update_inverter_states() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_iref() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6
 update_iref() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_parameter_dict() (pvder.DER_utilities.PVDER_ModelUtilities method), 10
 update_power() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6
 update_power() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_Ppv() (pvder.DER_utilities.PVDER_ModelUtilities method), 10
 update_Qref() (pvder.DER_utilities.PVDER_ModelUtilities method), 10
 update_ridethrough_flags() (pvder.DER_features.PVDER_SmartFeatures method), 13
 update_RMS() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6

update_RMS() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_RT_config() (pvder.DER_features.PVDER_SmartFeatures attribute), 12
 update_RT_config_old() (pvder.DER_features.PVDER_SmartFeatures method), 12
 update_Vdc_ref() (pvder.DER_utilities.PVDER_ModelUtilities method), 12
 update_voltages() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 6
 update_voltages() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 update_Zload1() (pvder.DER_utilities.PVDER_ModelUtilities method), 10
 use_frequency_estimate (pvder.DER_utilities.PVDER_ModelUtilities attribute), 10

V

V_threshold_high_limit (pvder.DER_features.PVDER_SmartFeatures attribute), 12
 va_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 10
 Vabrms_calc() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 5
 Vabrms_calc() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 validate_model() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 vb_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 vc_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 Vdc_actual (pvder.DER_utilities.PVDER_ModelUtilities attribute), 8
 Vdc_ref_counter (pvder.DER_utilities.PVDER_ModelUtilities attribute), 8
 Vdc_ref_list (pvder.DER_utilities.PVDER_ModelUtilities attribute), 8
 Vdc_ref_ramp() (pvder.DER_utilities.PVDER_ModelUtilities method), 8
 Vdc_ref_total (pvder.DER_utilities.PVDER_ModelUtilities attribute), 9
 VERBOSE (pvder.DER_utilities.PVDER_ModelUtilities attribute), 8
 Volt_VAR_absorb_range() (pvder.DER_features.PVDER_SmartFeatures method), 12

VOLT_VAR_FLAG (pvder.DER_features.PVDER_SmartFeatures attribute), 12
 Volt_VAR_inject_range() (pvder.DER_features.PVDER_SmartFeatures method), 12
 Volt_VAR_logic() (pvder.DER_features.PVDER_SmartFeatures method), 12
 VOLT_WATT_ENABLE (pvder.DER_features.PVDER_SmartFeatures attribute), 12
 VRT_ENABLE_FLAG (pvder.DER_features.PVDER_SmartFeatures attribute), 12
 voltage_ramp() (pvder.simulation_events.SimulationEvents method), 17
 Vrms_calc() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 5
 Vrms_calc() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 Vrms_min_calc() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 VRT_initialize() (pvder.DER_features.PVDER_SmartFeatures method), 12
 vta_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 Vtabrms_calc() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4
 vtb_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 vtc_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 Vtrms_calc() (pvder.DER_components_single_phase.SolarPVDERSinglePhase method), 5
 Vtrms_calc() (pvder.DER_components_three_phase.SolarPVDERThreePhase method), 4

W

we_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 wgrid_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 wgrid_estimate() (pvder.DER_utilities.PVDER_ModelUtilities method), 11
 wgrid_estimate_calc() (pvder.DER_utilities.PVDER_ModelUtilities method), 11

Y

y0 (pvder.DER_components_single_phase.SolarPVDERSinglePhase attribute), 6
 y0 (pvder.DER_components_three_phase.SolarPVDERThreePhase attribute), 4
 y0 (pvder.dynamic_simulation.DynamicSimulation attribute), 15

Z

`Zload1_actual_default`
(*pvder.simulation_events.SimulationEvents*
attribute), [16](#)