The Power and Energy Storage Systems Toolbox PSTess Version 1.0

Ryan T. Elliott[1], Daniel J. Trudnowski[2], Hyungjin Choi[1], and Tam Nguyen[2]

[1]Sandia National Laboratories
[2]Montana Technological University

September 2021

Abstract

This document describes the Power and Energy Storage Systems Toolbox for MATLAB, abbreviated as PSTess. This computing package is a fork of the Power Systems Toolbox (PST) Version 3.0. PST was originally developed at Rensselaer Polytechnic Institute (RPI) and later upgraded by Dr. Graham Rogers at Cherry Tree Scientific Software. While PSTess shares a common lineage with PST Version 3.0, it is a substantially different application. This document supplements the main PST manual by describing the features and models that are unique to PSTess. As the name implies, the main distinguishing characteristic of PSTess is its ability to model inverter-based energy storage systems (ESS). The model that enables this is called `ess.m`, and it serves the dual role of representing ESS operational constraints and the generator/converter interface. As in the WECC REGC_A model, the generator/converter interface is modeled as a controllable current source with the ability to modulate both real and reactive current. The model `ess.m` permits four-quadrant modulation, which allows it to represent a wide variety of inverter-based resources beyond energy storage when paired with an appropriate supplemental control model. Examples include utility-scale photovoltaic (PV) power plants, Type 4 wind plants, and static synchronous compensators (STATCOM). This capability is especially useful for modeling hybrid plants that combine energy storage with renewable resources or FACTS devices.

*Send correspondence to `rtellio@sandia.gov`.

Contents

1 Summary of differences with PST

This section lists the models, features, and miscellaneous changes that differentiate PSTess from PST Version 3.0 [1,2]. Because of structural changes made to the code, such as the reorganization of global variables, models developed for PSTess are not backward compatible with prior versions of PST. The most important example of this is the inverter-based energy storage system (ESS) model, `ess.m`. This model employs a custom flag that allows it to iterate back and forth with the power flow algorithm specified in `nc_load.m`. The purpose of this iteration is to reconcile the current injection provided by the inverter with the network solution. Hence, attempting to port `ess.m` for use with other versions of PST is not recommended. For a detailed changelog of the modifications and additions that went into creating PSTess, see the README.

1.1 Models

The new models introduced in PSTess are summarized in the list below. The key entries are `ess.m`, the inverter-based ESS model, and `ess_sud.m`, a flexible user-defined controller that interfaces with `ess.m`. See Section 2 for additional information about `ess.m` and `ess_sud.m`. In addition to user-defined code, energy storage systems can also interface with built-in PSTess models. As an example, we provide a linear time-varying (LTV) synchronizing torque control model, called `lsc.m`. For additional information about the control strategy implemented in `lsc.m`, see [3].

- `ess.m`: Inverter-based ESS model with four-quadrant modulation capability.
- `ess_sud.m`: User-defined ESS control model.
- `lsc.m`: Linear time-varying synchronizing torque controller (for use with `ess.m`).

The `ess.m` model was primarily designed for utility-scale batteries; however, it can also be used to represent other types of inverter-based resources provided that an appropriate user-defined or built-in control model is provided. If `ess.m` is used to represent resources that lack storage, the energy capacity should be set to a large value to ensure that the state of charge constraints do not bind.

1.2 Features

In addition to new dynamic models, PSTess also expands upon the feature set of PST, as shown in the list below. One of the biggest structural changes is that the global variables employed by the application have been grouped into a common struct `g`. The reason for this change is two-fold: it clearly distinguishes between global and local variables, and it also paves the way for potentially eliminating global variables from the application. The new file `contents_global.m` contains an index showing the organization of the global struct `g` and how to access internal program data. See Section 3 for additional information about the new features included in PSTess.

- Global variables: All global variables have been reorganized into a single struct, called `g`.
- Batch processing: A new mode of operation has been added to modify application-wide behavior when performing multiple simulation (or analysis) runs; see `get_dypar.m`.

- Error statements: All errors and warnings issued by the application have been revised to be more specific, consistent, and useful.
- Dynamic brake insertions: `y_switch.m` has been updated to support three-phase resistive shunt insertions (e.g., for modeling the Chief Joseph Brake); this option is now available as `f_type=8`.
- Linearization routine: `svm_mgen.m` has been redesigned around the central difference method for improved accuracy; the surrounding code has also been modified to better track the time-domain simulation procedure in `s_simu.m`.

1.3 Bug fixes and miscellaneous changes

Throughout the course of developing PSTess we made several changes to coding conventions and patched a few bugs in PST. These are summarized in the list below. MATLAB has evolved considerably since development on PST first began in the early 1990s. These stylistic changes are largely aimed at bringing the codebase in line with contemporary MATLAB conventions. See Section 4 for further details about the patches implemented in PSTess.

The impact of the patch in the HVDC model `inv_lf.m` has not been thoroughly tested. We recommend that the HVDC modeling capabilities available in PSTess be exercised only by advanced users who are comfortable making changes to the source code.

- `exc_st3.m`: The angle variable `theta` was mistakenly indexed by the machine number `n` rather than the bus number `n_bus`; this has been corrected.
- `inv_lf.m`: The angle variable `gamma` was mistakenly reported in radians rather than degrees; this has been corrected for consistency with the rest of the HVDC code, including `rec_lf.m`.
- `tcsc.m`: This model is evidently based on `svc.m` because in the dynamics calculation the state derivative was accidentally called `dB_cv` instead of `dB_tcsc`; this `dB_cv` was subsequently never used, so this bug has been corrected.
- Imaginary numbers: The original `jay = sqrt(-1)` convention has been eliminated in favor of `1j`.
- Dummy variables: Functions no longer return unnecessary dummy variables.
- Unused function inputs: It appears that many functions were based on a common template and had unused inputs as a result; these have been removed to avoid confusion.

2 Models

2.1 Energy storage systems

This section describes the structure of `ess.m` and the corresponding user-defined control model `ess_sud.m`. This framework was designed to represent utility-scale battery-based storage systems. Today, a majority of these systems employ PLL-driven inverters whose controls specify a commanded current injection. We are actively developing models of grid-forming inverters whose controls specify a commanded voltage phasor. The code in `ess.m` specifies a model of

the energy storage device and the generator/converter interface, the latter of which is loosely based on the WECC REGC_A model [4]. On its own, `ess.m` can be used to represent steady-state charging/discharging behavior; however, to implement any type of closed-loop control, a supplemental model is required. Section 2.1.3 discusses the interface that allows other models to connect to `ess.m`.

2.1.1 Input parameters

Table 1 describes the model parameters and the structure of the matrix `ess_con` where they are entered. For parameters that require further information, a letter has been added to the "note" column of the table. See the corresponding list of notes below. As with other PSTess models, the `ess_con` matrix must be specified in the file that contains the algebraic and dynamic data for the case. See `d2asbegp_ess.m` and `d_minniWECC_ess.m` under the `data` subfolder for examples of how to specify `ess_con`. These cases are discussed further in Section 5.

Table 1: Parameter description and structure of `ess_con`

column	description	typical	units	note
1	energy storage system number	-	integer	-
2	bus number	-	integer	-
3	voltage transducer time constant	0.02	s	-
4	use Padé approximation flag	0	binary	a
5	voltage magnitude time delay (Padé)	-	s	-
6	apparent power capacity	-	MVA	-
7	energy capacity	-	MWh	-
8	converter voltage for current limit	0.93	pu	b
9	complex power priority mode	2	integer	c
10	converter allowable power factor	0.00	power factor	d
11	initial state of charge	$[0, 1]$	fraction of capacity	e
12	minimum state of charge	$[0, 1]$	fraction of capacity	e
13	maximum state of charge	$[0, 1]$	fraction of capacity	e
14	converter interface time constant	0.02	s	-
15	active current ramp rate limit	-	pu/s on ess base	-
16	reactive current ramp rate limit	-	pu/s on ess base	-
17	LVPL breakpoint current	1.22	pu on ess base	f
18	LVPL zero crossing voltage	0.50	pu	f
19	LVPL breakpoint voltage	0.90	pu	f
20	charge/discharge only flag	0	integer	g
21	charge/discharge efficiency	$[0, 1]$	% as fraction	e

Notes:

a) The `ess.m` model features an optional time delay approximation applied to the terminal voltage measurement. As described in Section 2.1.3, this feature is implemented using a first-order Padé approximant. It is primarily intended as means of determining the impact of delay on the frequency-domain properties of control systems. We do not recommend using this feature for time-domain simulation because shift registers are better suited for

that application. The flag in column 4 of `ess_con` is interpreted as follows:

`=0` : No time delay approximation
`=1` : First-order Padé approx. applied to terminal voltage measurement

b) The value specified in column 8 of `ess_con` is the minimum per unit voltage at which the inverter is capable of supplying its rated apparent power. This value is used to determine the current capacity of the inverter in the saturation/limiting scheme. A voltage lower than 1.0 pu implies that the power electronics are rated conservatively, i.e., the inverter is oversized. Hence, values larger than 1.0 pu imply that the inverter is not capable of supplying its rated apparent power at nominal voltage, and so should be seldom used. Refer to the definition of the internal parameters stored in `ess_pot` in `ess.m` for details.

c) The inverter saturation/limiting scheme is designed to support three different modes of operation:

`=1` : Real power priority
`=2` : Reactive power priority
`=3` : Proportional power sharing

In mode 1, the inverter allocates all of of its capacity to active current. Hence, reactive modulation is only permitted if the active current command is less than the current capacity. In mode 2, the converse is true and the inverter prioritizes reactive current. Finally, in mode 3, the capacity is shared and the active and reactive current commands are curtailed proportionally upon saturation. See Section 2.2 for further information.

d) The inverter may be constrained to operate within a specified power factor range. If the value in column 10 of `ess_con` is zero, the inverter is allowed to operate at any power factor. When configured this way, it is possible for the inverter to supply reactive power without simultaneously providing real power. At the other extreme, when the value in column 10 is one, the inverter is forced to operate at unity power factor, meaning that it cannot supply reactive power. When the specified power factor is in the interval $(0, 1)$, it imposes symmetric bounds around unity. For example, a value of 0.95 means that the inverter is allowed to operate at any power factor between 0.95 leading and 0.95 lagging.

Care must be taken to ensure that this parameter does not conflict with the steady-state output of the inverter specified in the `bus` matrix. For example, if the inverter supplies reactive power in steady-state but not active power, then the value in column 10 of `ess_con` must be zero. Otherwise, the dynamic model will curtail the reactive current after initialization, causing the operating point to change. See Section 2.1.2 for additional information about how `ess` is integrated into the power flow calculations.

e) In `ess.m`, the state of charge of the ith ESS is modeled as

$$\dot{E}_i(t) = \eta_i p_i^{\text{ch}}(t) - p_i^{\text{dis}}(t)/\eta_i, \tag{1}$$

where $E_i(t)$ is the energy stored at time t, and η_i the charge/discharge efficiency entered into column 21 of `ess_con`. In (1), $p_i^{\text{ch}}(t)$ is the charging amount, and $p_i^{\text{dis}}(t)$ the discharging

amount. At a given time t only one of these quantities may be nonzero, i.e.,

$$p_i^{\text{ch}}(t) = \begin{cases} -p_i(t), & p_i(t) < 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

$$p_i^{\text{dis}}(t) = \begin{cases} p_i(t), & p_i(t) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here $p_i(t)$ is the real power injection from the ith ESS to the grid at time t. We recommend that the efficiency factor η_i be computed such that

$$\eta_i = \sqrt{\eta_i^{\text{rt}}}, \tag{4}$$

where η_i^{rt} is the round trip efficiency. For example, if the round trip efficiency is 81 %, we have $\eta_i^{\text{rt}} = 0.81$ and $\eta_i = 0.90$.

The state of charge is tracked so that it is possible to monitor when the storage device is fully charged or depleted. The model also allows the user to specify operational limits for the state of charge, e.g., to limit depth of discharge. The lower bound on the allowable state of charge range is specified as a fraction of the energy capacity in column 12 of `ess_con`, and the upper bound in column 13. The initial state of charge is entered in column 11. The initial state of charge should fall within the allowable range.

f) Here LVPL stands for *low-voltage power logic*. This feature provides the user with a mechanism to introduce a voltage-dependent limit on the real current provided by the inverter. The parameters in columns 17, 18, and 19 together specify points on a piecewise linear limit characteristic. For further details, see Section 2.2.

g) To facilitate modeling other types of resources besides storage, the `ess.m` model may be prevented from either charging or discharging. For example, an aggregated collection of electric vehicles may provide an ancillary service to the grid by modulating its charging behavior. In that scenario, we would set the value of column 20 to "charge only" mode. Conversely, for utility-scale PV plants that lack energy storage, "discharge only" mode would be appropriate. And for systems that may either charge or discharge, we have "bidirectional modulation" mode. This behavior is controlled by setting the flag in column 20 as follows:

`=0` : Bidirectional modulation
`=1` : Charge only
`=2` : Discharge only

For further information about the saturation logic, see Section 2.2.

2.1.2 Power flow

This section describes how ESSs are managed in power flow calculations. Each ESS must be located on its own bus, i.e., not co-located with any other sources or sinks. In steady-state power flow calculations, the buses at which ESSs are located are treated as load buses, i.e., PQ buses. Thus, the power flow algorithm respects the reactive power output of the ESS rather than the steady-state voltage magnitude.

Because ESSs are treated as loads in power flow, the values in the `bus` matrix corresponding to real and reactive power generation (columns 4 and 5) must be zero; however, the values specified in the load columns (6 and 7) may be nonzero. If the load specified in the `bus` matrix is nonzero at an ESS bus, then that value is interpreted as the steady-state apparent power withdrawal of the ESS. For example, if the value listed in column 6 of the `bus` matrix is 0.01, then the ESS located at that bus initially charges at a rate of 1 MW (with a 100 MVA system base). Any modulation command sent to the ESS then adjusts the charging behavior around that 1 MW initial value.

Finally, injections from the ESSs are treated as constant current in the network solution algorithm in `nc_load.m`. This function handles what are referred to as *non-conforming* loads, meaning loads that are not constant impedance. Loads that behave this way necessitate a nonlinear solution procedure to determine the boundary current injections between the sources/sinks and the network. In the case of `ess.m`, we implemented a custom flag that allows the Newton-Raphson algorithm in `nc_load.m` to iterate with `ess.m`. To begin the procedure, the `ess` model specifies the inverter current injections based on an initial vector of terminal voltages. The `nc_load.m` bus then updates the terminal voltages using the increment computed in the Newton-Raphson algorithm. These two processes then iterate to determine the final `ess` current injections.

2.1.3 Dynamics

Here, we describe the dynamics of the `ess.m` model. Recall that in order to implement closed-loop control, a supplemental model is required that sends real and reactive power commands to the ESS(s). Fig. 1 provides a simplified block diagram of the `ess` model where P_{ref} and Q_{ref} are setpoints determined by the steady-state power flow solution. The signals p_s and q_s are the per unit real and reactive power modulation commands (on the system MVA base), respectively.

The converter interface model depicted in Fig. 1 neglects the PLL dynamics. Hence, the model effectively assumes that the inverter is able to instantaneously track the voltage angle reference from the grid. Future iterations of the `ess` framework will explore the capability to explicitly model the PLL dynamics.

Table 2 lists the dynamic states of the model. All of these states have been integrated into

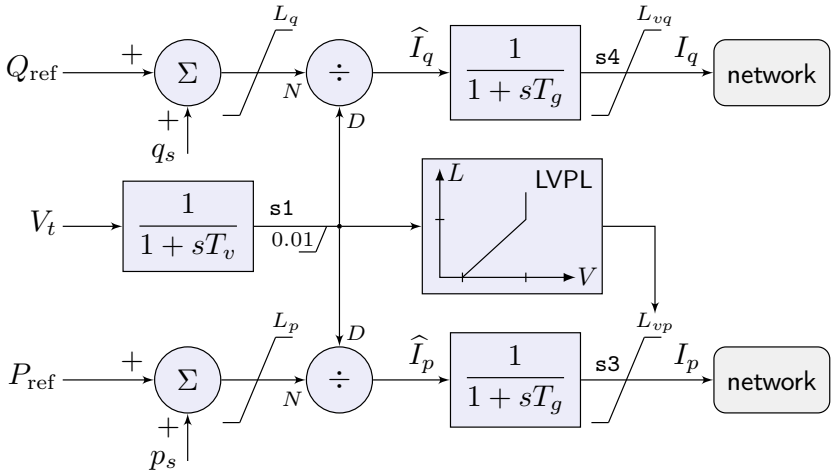

Figure 1: High-level block diagram of `ess.m`.

Table 2: State variables

variable	description
`ess1`	transducer for the local voltage magnitude
`ess2`	Padé approx. for the local voltage magnitude
`ess3`	active current converter interface
`ess4`	reactive current converter interface
`ess5`	state of charge integrator

the PSTess linearization routine, meaning that it is possible to perform linear analysis with the `ess` dynamics in the loop. In Fig. 1, the voltage transducer is modeled as a first-order lowpass filter, where the output is `ess1`. Similarly, the active and reactive converter interfaces are also modeled as first-order lowpass filters, where the outputs are `ess3` and `ess4`, respectively. Each of these filters may be bypassed by setting the time constant equal to zero; however, the parameter values should reflect the characteristics of the physical components being modeled.

We have developed three ways to interface with `ess.m`, i.e., to supply p_s and q_s in Fig. 1. Each method involves specifying a complex-valued command signal where the real part is interpreted as p_s, and the imaginary part as q_s.

- User-defined model: Implement a custom control strategy in `ess_sud.m`; the complex-valued output is specified as `g.ess.ess_dsig`.
- Built-in PSTess model: Implement a built-in model and specify the complex power order as `g.ess.ess_scmd`; see `lsc.m` for an example.
- `mess_sig.m`: Modulate the output of one or more ESSs by sending a complex-valued probing signal or test pulse via `g.ess.ess_sig` (not for closed-loop control).

When interfacing with the `ess` models, care should be taken to ensure that the mapping between control models and actuators is implemented as expected, i.e., that the appropriate controller is "talking" to the appropriate actuator. See `ess_indx.m` for an example of how to manage this mapping. And do not attempt to use `mess_sig.m` to implement closed-loop control strategies; it is intended for injecting pulses and probing signals only.

There are two states not depicted in Fig. 1, `ess2` and `ess5`. The Padé approximation for the terminal voltage measurement is optional and intended for research purposes. We recommend that it be used only for frequency-domain analysis because shift registers are better suited for modeling time delays in simulation. Fig. 2 shows the first-order Padé approximation where $\widetilde{V}_t$ is the output of the terminal voltage transducer, and the time delay is $\tau_d = 2A_\tau$.

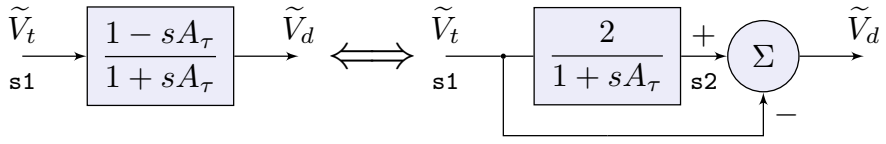

Figure 2: `ess2`, Padé approximation.

The state of charge integrator partially determines the final bounds on the active current, I_p. Charging and discharging losses are modeled as described in Section 2.1.1. Fig. 3 shows a

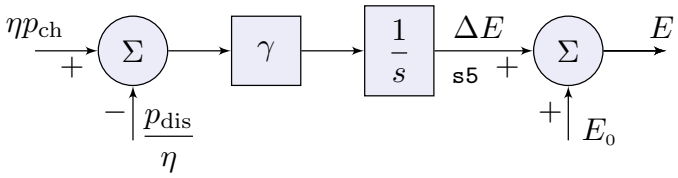

Figure 3: `ess5`, state of charge tracking.

diagram of the implementation in PSTess, where γ is a conversion factor to translate between unit bases, and η is the one-way efficiency factor from (4). The state `ess5` is the deviation in the state of charge away from its initial value.

2.2 Saturation logic

The model `ess.m` includes not only the ESS and generator/converter dynamics, but also saturation logic for representing how the system behaves when approaching its current limits. The inverters integrated with battery-based ESSs are current limited, meaning that they may not be able to supply their rated power at off-nominal voltages. To compensate for this limitation, some inverters are oversized which allows them to provide their rated power over a wider range of voltages. Fig. 4 shows the operating characteristics for an oversized inverter, where the shaded region indicates the allowable power factor range.

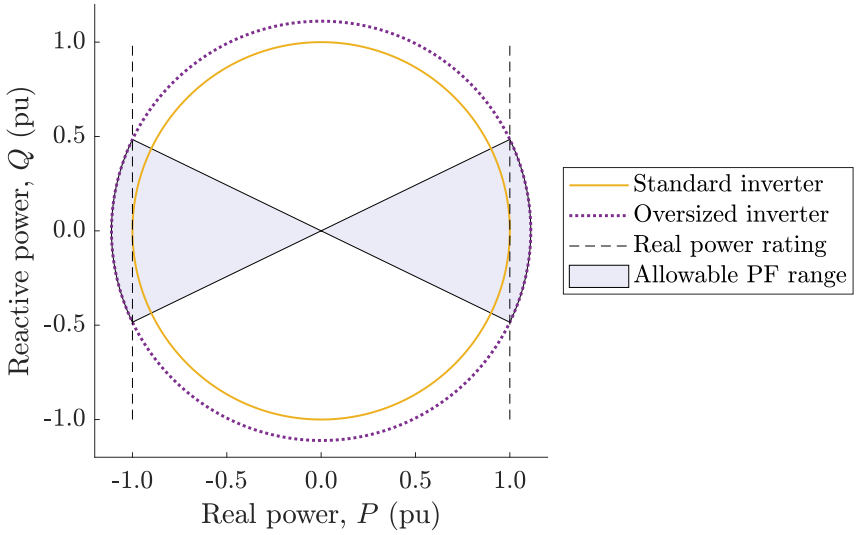

Figure 4: Operating characteristics for an oversized inverter.

2.2.1 Complex power priority

The saturation characteristics of `ess.m` also depend on the complex power priority mode (column 9 of `ess_con`). Some operating entities require inverter-based generators to run in reactive power priority mode [5]. This ensures that inverter-based generators will contribute to post-disturbance voltage regulation. The snippet of code from `ess_sat.m` below shows how the priority mode affects the allocation of available current capacity. Here `g.ess.ess_pot(:,3)` is the current

capacity of the inverter on the system MVA base (see `ess.m`). In the code excerpt, `p_inj` is the sum of the steady-state real power injection and the commanded power specified by the supplemental control model. Likewise for `q_inj` with reactive power.

Current capacity allocation

```
% pre-saturation complex current
ip_inj = p_inj./max(v_term,lbnd);
iq_inj = q_inj./max(v_term,lbnd);
i_inj = ip_inj + 1j*iq_inj;

% checking for current limit violations
mask = (abs(i_inj) > g.ess.ess_pot(:,3));
if any(mask)
    ip_sign = sign(ip_inj);                         % storing injection signs
    iq_sign = sign(iq_inj);

    ip_inj(mask) = abs(ip_inj(mask));               % one-quadrant perspective
    iq_inj(mask) = abs(iq_inj(mask));

    ip_lim = g.ess.ess_pot(:,3);                    % initialize the limits
    iq_lim = g.ess.ess_pot(:,3);

    p_mask = mask & (g.ess.ess_con(:,9) == 1);      % active priority
    q_mask = mask & (g.ess.ess_con(:,9) == 2);      % reactive priority
    pq_mask = mask & (g.ess.ess_con(:,9) == 3);     % proportional scaling

    % active priority
    ip_inj(p_mask) = min(ip_inj(p_mask),g.ess.ess_pot(p_mask,3));
    iq_lim(p_mask) = sqrt(g.ess.ess_pot(p_mask,3).^2 - ip_inj(p_mask).^2);
    iq_inj(p_mask) = min(iq_inj(p_mask),iq_lim(p_mask));

    % reactive priority
    iq_inj(q_mask) = min(iq_inj(q_mask),g.ess.ess_pot(q_mask,3));
    ip_lim(q_mask) = sqrt(g.ess.ess_pot(q_mask,3).^2 - iq_inj(q_mask).^2);
    ip_inj(q_mask) = min(ip_inj(q_mask),ip_lim(q_mask));

    % proportional scaling
    pq_scale = g.ess.ess_pot(pq_mask,3)./abs(i_inj(pq_mask));
    ip_inj(pq_mask) = pq_scale.*ip_inj(pq_mask);
    iq_inj(pq_mask) = pq_scale.*iq_inj(pq_mask);

    % accounting for positive and negative injections
    ip_inj(mask) = ip_sign(mask).*abs(ip_inj(mask));
    iq_inj(mask) = iq_sign(mask).*abs(iq_inj(mask));
end
```

2.2.2 Low-voltage power logic

The *low-voltage power logic* (LVPL) block shown in Fig. 1 allows the user to implement a voltage-dependent limit on the active current injection. This serves multiple purposes, for instance, accounting for the fact that the inverter cannot inject power into a faulted bus. The limit characteristic is piecewise linear and has three segments. An example is provided in Fig. 5. The final active current limit is determined by the LVPL logic, the state of charge constraints,

and the current capacity allocation of the inverter. The snippet of code below shows the LVPL implementation in `ess_sat.m`. The variable names used in Fig. 5 follow the MATLAB code. To disable this feature, the user may enter parameters in columns 17, 18, and 19, of `ess_con` that produce an LVPL limit curve that exceeds the current capacity of the inverter for all voltages.

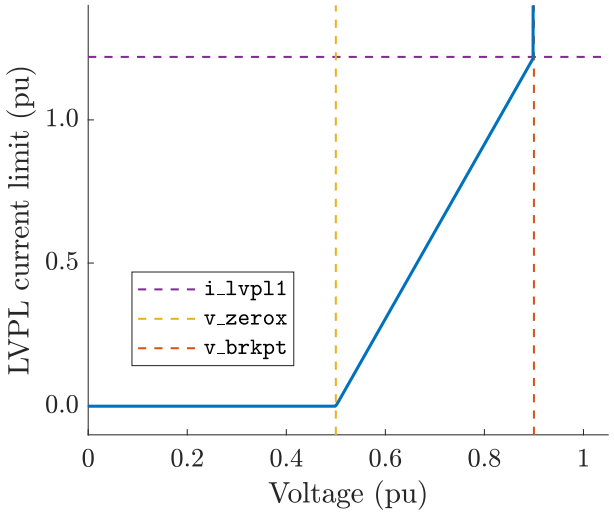

Figure 5: Low-voltage power logic characteristic.

LVPL implementation

```
% i_lvpl1 -- breakpoint current (pu on syst. base)
% v_zerox -- zero crossing volt (pu)
% v_break -- breakpoint voltage (pu)
i_lvpl1 = g.ess.ess_con(:,17).*g.ess.ess_pot(:,1);
v_zerox = g.ess.ess_con(:,18);
v_break = g.ess.ess_con(:,19);

a_slope = i_lvpl1./(v_break - v_zerox);
b_inter = -i_lvpl1.*v_zerox./(v_break - v_zerox);

ip_lvpl_ub = i_lvpl1;

mask = (v_term < v_break);                          % linear roll-off
if any(mask)
    ip_lvpl_ub(mask) = a_slope(mask).*max(v_term(mask),0) + b_inter(mask);

    zero_mask = (v_term < v_zerox);                 % zero after this point
    if any(zero_mask)
        ip_lvpl_ub(zero_mask) = 0.0;
    end

    ip_lvpl_lb = -ip_lvpl_ub;                       % symmetric limits

    ip_inj(mask) = max(ip_inj(mask),ip_lvpl_lb(mask));
    ip_inj(mask) = min(ip_inj(mask),ip_lvpl_ub(mask));
end
```

2.3 User-defined energy storage controls

The protocol for interacting with `ess.m` using a user-defined model provides flexibility for rapidly prototyping various control strategies. When defining or redefining `ess_sud.m` it is not necessary to make any modifications or additions to the simulation routine `s_simu.m`, provided that the implementation rules are followed. In contrast, when developing a new built-in PSTess model, such as `lsc.m`, the simulation routine must be modified to properly initialize and integrate the states of the model. See `ess_sud.m` in the `pstess` folder for an example that also serves as a coding template. User-defined controls are not included in the linearization routine in `svm_mgen.m`.

The implementation rules for using `ess_sud.m` are as follows:

- All `ess_sud` input data and model parameters must be defined in the base case data file using the matrix `essud_con`. This matrix may contain any number of parameters.
  - The rows of this matrix correspond to the number of `ess_sud` instances.
  - The columns correspond to the number of parameters employed by `ess_sud`.
  - The first column of `essud_con` must contain the `ess_sud` controller index number.
  - The second column of `essud_con` must specify the bus number (external) to which the ESS linked to the controller instance is connected.
  - No ESS may receive commands from more than one `ess_sud` instance.
- All state variables and state derivatives must be organized within the struct `x`. This struct may contain any number of states and/or algebraic storage variables.
  - For example, state 1 is accessible as `x.s{1}`, state 2 as `x.s{2}`, and so on.
  - Each state variable `x.s{n}` is a two-dimensional array where the rows correspond to the user-defined controllers, and the columns correspond to the time indexes. So, `x.s{1}(:,1)` is the vector of initial values for state 1.
  - Similarly, the state derivatives are accessible as `x.ds{1}`, state 2 as `x.ds{2}`, and so on, with dimensions that match those of the states.
  - In addition to the states and state derivatives, `x` may also contain algebraic storage variables, available as `x.v{1}`, `x.v{2}`, etc.
  - By definition algebraic storage variables are not sent to the numerical integration routine, so do not use `x.v` to implement dynamic states.
- The `ess_sud.m` model must update the variable `g.ess.ess_dsig` in order to interact with `ess.m`.
  - The rows of `g.ess.ess_dsig` correspond to the `ess` instances, which may be different from the number `ess_sud` controller instances.
  - Care should be taken to ensure that the `ess_sud` model updates the appropriate entries of `g.ess.ess_dsig`.
  - In `ess_indx.m`, we compute `g.ess.dessud_idx` which is the set of ESS indexes connected to `ess_sud.m` controllers.
  - Thus, to update the apparent power command at time `k`, one could set the value of `g.ess.ess_dsig(g.ess.dessud_idx,k)`.
  - This update should take place in the dynamics calculation, i.e., when `flag == 2`.

3 Features

This section describes the new features available in PSTess. The reorganization of the global variables and the new batch processing mode change certain aspects of the user experience, so we encourage users with prior PST exposure to read those sections carefully.

3.1 Global variables

PST Version 3.0, as with prior versions, relies upon hundreds of global variables, and in some cases it is not clear from context which variables are global and which are local. To improve readability and efficiency in PSTess we have reorganized all the global variables into a common struct `g`. In addition to clearly differentiating between global and local variables, this change paves the way to potentially eliminate global variables in the future. To clearly match variables with physical components, they have been grouped under fields within the struct `g`. For example, the field `g.mac` contains all synchronous machine variables, `g.exc` all exciter variables, and so on. Table 3 provides a partial mapping between the old and new variable names. For a comprehensive list of global variables and field names, see `contents_global.m`. See also the new `import_var.m`, which imports data into the global struct `g`.

Table 3: Global variable structure examples

old variable	description	group	new variable
`n_mac`	number of generators	`mac`	`g.mac.n_mac`
`mac_con`	dynamic model parameters	`mac`	`g.mac.mac_con`
`mac_pot`	internal `mac` constants	`mac`	`g.mac.mac_pot`
⋮	⋮	⋮	⋮
`n_exc`	number of exciters	`exc`	`g.exc.n_exc`
`exc_con`	dynamic model parameters	`exc`	`g.exc.exc_con`
`exc_pot`	internal `exc` constants	`exc`	`g.exc.exc_pot`

3.2 Batch processing

It is often desirable to run a batch of computations one after the other, uninterrupted. By default, PST Version 3.0 prompts the user to select data files, insert system base values, and so on. These queries can be helpful for new users, but they also complicate the task of batch processing. To address this problem, we have implemented a new application-wide batch processing mode that bypasses all queries. This new mode also manages the information that gets retained by the application between each simulation run or system linearization. The application behavior is controlled using the variable `dypar.batch_mode`. When this variable is `true`, PSTess will run in batch mode. When it is `false`, PSTess will revert to the default user interaction settings. The value of `dypar.batch_mode` should be set in `get_dypar.m`.

```
% control the batch processing mode from line 21 of get_dypar.m
dypar.batch_mode = true;  % may be either true or false
```

When using batch processing, the user should specify the working data file in `get_path.m`. A helper function `set_path.m` is available for changing the working directory or data file

programmatically. This function works by overwriting the text file that defines the function `get_dypar.m`. Thus, its changes are durable and not affected by `clear` calls. Below is an example that demonstrates how to use `set_path()`. If the `get_dypar.m` function is ever deleted or botched, the header of `set_path.m` contains its default code.

```
% change both the data file and working directory (file location)
set_path('my_data_file.m','C:\Users\joebloggs\workdir\data\')

% change the data file without changing the directory
set_path('my_other_data_file.m')
```

3.3 Error statements

In PSTess, the error and warning statements issued by the program have been modified to be more specific, consistent, and useful. Each error message now provides the name of the file that threw the error, the error message, and the associated component or bus numbers (if applicable). The snippet of code from `exc_st3.m` shown below illustrates the basic form of the new message convention. In this case, if the field voltage of any generator exceeds its upper bound upon initialization, an error is thrown.

```
max_lim = find(g.exc.Efd(g.exc.st3_idx,1) ...
               > g.exc.exc_con(g.exc.st3_idx,18));

if ~isempty(max_lim)
    n_error = g.mac.mac_int(g.exc.exc_con(max_lim,2));
    estr = '\nexc_st3: Efd exceeds maximum in initialization at ';
    estr = [estr, 'machine index %0.0f.'];
    error(sprintf(estr,n_error));
end
```

The message below shows the error thrown when the field voltage check fails for generators 1 and 3.

```
Error using exc_st3 (line 173)

exc_st3: Efd exceeds maximum in initialization at machine index 1.
exc_st3: Efd exceeds maximum in initialization at machine index 3.
```

All other error messages are implemented in a similar way. These changes improve consistency and facilitate debugging.

3.4 Dynamic brake insertions

In order to simulate a particular type of contingency, it must be described as a sequence of actions in `y_switch.m`. In PSTess, the set of available contingencies has been expanded to include dynamic braking resistor insertions. The braking resistor is modeled as a three-phase resistive shunt. This option is useful for modeling the Chief Joseph Brake in the Western Interconnection, which is used during annual testing to excite the oscillatory modes. Table 4 lists the available contingencies in PSTess specified as switching options in `y_switch.m`.

Table 4: Switching options

number	description
0	three-phase bolted fault
1	single line to ground fault
2	double line to ground fault
3	line to line fault
4	loss of line with no fault
5	loss of load at bus
6	no action
7	three-phase bolted fault without loss of line
8	three-phase resistive shunt

Next we provide an example specification for `sw_con` that implements a brake insertion at bus 1. The entry in column 4 specifies the per unit resistance of the brake. In the code below, the resistance is set to 0.1 pu, which means the conductance is 10 pu. When the system apparent power base is 100 MVA, this translates to a 1 GW load. Here, the dynamic braking resistor is inserted for 0.5 s, then removed to inject a test pulse into the system. We recommend that users exercise sound engineering judgment when choosing the brake resistance and insertion time to prevent the system from going unstable (unless that is the desired result).

```
% example brake insertion specification
sw_con = [...
    0.0       0  0    0  0  0  my_Ts;   % sets initial time step
    1.0       1  0  0.1  0  8  my_Ts;   % 8=brake insertion
    1.5       0  0    0  0  0  my_Ts;   % remove the brake
    1.5+1/60  0  0    0  0  0  my_Ts;   %
   20.0       0  0    0  0  0  my_Ts];  % end simulation
```

3.5 Linearization routine

The linearization routine specified in `svm_mgen.m` has been redesigned around the central difference method for improved accuracy. Recall that the primary purpose of `svm_mgen.m` is to build a state-space representation of a power system model around an operating point. Here we provide a high-level overview of how the linearization routine has evolved.

In prior versions of PST, the forward difference method was used to estimate the entries of the Jacobian matrices. Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denote a nonlinear vector field

$$\dot{x}(t) = f(x(t), u(t)), \tag{5}$$

where $x(t) \in \mathbb{R}^n$ is the system state at time t and $u(t) \in \mathbb{R}^m$ the input. Using the forward difference method to estimate the entry in the ith row, jth column of the system matrix yields

$$\left.\frac{\partial f_i}{\partial x_j}\right|_{x_0,u_0} \approx \frac{f_i(x_0 + \Delta x_j e_j, u_0) - f_i(x_0, u_0)}{\Delta x_j}, \tag{6}$$

where f_i is the ith component of the vector field, Δx_j the perturbation in the jth state, and e_j a unit vector whose jth entry is one. Here $\{x_0, u_0\}$ denotes the operating point around which the system is linearized.

In many cases, the performance of the central and forward difference methods is similar; however, the central difference method has a higher order of accuracy, offers advantages in estimating phase response, and can help identify numerical problems in cases where the forward and backward differences disagree. For the example given above, the central difference method takes the form

$$\left.\frac{\partial f_i}{\partial x_j}\right|_{x_0,u_0} \approx \frac{f_i(x_0 + \Delta x_j e_j, u_0) - f_i(x_0 - \Delta x_j e_j, u_0)}{2\Delta x_j}. \tag{7}$$

When performing linear analysis, we strongly recommend that the simulation output of the linearized and original models be compared in the time domain. This is the most reliable way to ensure that the state-space model is representative of the original nonlinear system. For further details on the linearization routine, see `svm_mgen.m` and the functions and scripts called therein.

4 Bug fixes

This section provides details about the PST bugs that were patched while developing PSTess. Because of the changes to the coding conventions discussed in Section 1, some of the patches look slightly different from the original code.

4.1 Exciter models

In line 151 of `exc_st3.m` in PST Version 3.0, the angle variable `theta` was mistakenly indexed by the machine number vector `n` rather than the bus number vector `n_bus`. This led to the value of `vep` being based on the incorrect voltage angles.

```
% bug: line 151 of exc_st3.m in PST v3.0
vep = eterm(n,1).*exp(jay*theta(n,1)).*(exc_pot(st3_idx,1) ...
                                        + jay*exc_pot(st3_idx,2));
```

The section of `exc_st3.m` where this bug has been patched in PSTess is shown below.

```
% patch: line 183 of exc_st3.m in PSTess v1.0
vep = g.mac.eterm(n,1).*exp(1j*g.bus.theta(n_bus,1)) ...
      .*(g.exc.exc_pot(g.exc.st3_idx,1) ...
         + 1j*g.exc.exc_pot(g.exc.st3_idx,2));
```

4.2 HVDC models

In line 157 of `inv_lf.m` in PST Version 3.0, the angle variable `gamma` was mistakenly reported in radians rather than degrees. This is a problem because in line 102 of `dc_lf.m`, the angle `gamma` is multiplied by a factor of `pi/180` in an attempt to convert from degrees to radians. However, this conversion is incorrect because `gamma` was originally defined in radians.

```
% bug: line 157 of inv_lf.m in PST v3.0
gamma = atan2(sgamma,cgamma);  % the output of atan2 is in rad

% downstream of bug: line 79 of dc_lf.m in PST v3.0
[gamma] = inv_lf(mode,idc,Vdc_max,Vdc_min, ...
                 ga_min,ga_max,Vdo(i_idx),Rc(i_idx));

% downstream of bug: line 102 of dc_lf.m in PST v3.0
Vdo(i_idx) = (Vdc(i_idx)+Rc(i_idx).*idc)./cos(gamma*pi/180);
```

The section of `inv_lf.m` where this bug has been patched in PSTess is shown below.

```
% patch: line 202 of inv_lf.m in PSTess v1.0
gamma = atan2(sgamma,cgamma)*180/pi;  % converting to degrees
```

4.3 TCSC models

The model `tcsc.m` was evidently based on `svc.m` because in the dynamics calculation the state derivative was accidentally called `dB_cv` instead of `dB_tcsc`. This `dB_cv` assigned in `tcsc.m` was subsequently never used. This bug has been corrected.

```
% bug: line 58 of tcsc.m in PST v3.0
dB_cv(i,k) = (-B_tcsc(i,k)+tcsc_con(i,4)*err)/tcsc_con(i,5);
```

The section of `tcsc.m` where this bug has been patched in PSTess is shown below.

```
% patch: line 44 of tcsc.m in PSTess v1.0
g.tcsc.dB_tcsc(i,k) = (-g.tcsc.B_tcsc(i,k) ...
                       + g.tcsc.tcsc_con(i,4)*err) ...
                      /g.tcsc.tcsc_con(i,5);
```

5 Example cases

To help users get started with PSTess we have provided two base cases containing energy storage: a modified version of the Klein-Rogers-Kundur 2-area test case and a reduced-order model of the Western Interconnection called the *miniWECC* [6, 7]. Both cases are contained in the `data` subfolder; the first is called `d2asbegp_ess.m` and the second `d_minniWECC_ess.m`. In each case, the ESSs are located near the synchronous machines. Each ESS is rated at 0.5C, meaning that it would take one hour to charge (or discharge) 50 % of its energy capacity at rated power. The power rating is set to 1.5 % of the corresponding machine MVA base. These cases are intended as a testbed for research. To exercise the capabilities of PSTess, they have been configured to utilize the synchronizing torque control strategy described in [3]. We encourage users to develop and test their own energy storage controls.

We have performed tests with the Kundur 2-area and miniWECC models to compare the performance of PSTess against PST Version 3.0 (with the patches described in Section 4). The simulated trajectories produced with the two software packages agreed to numerical precision; however, we caution that some features, such as the HVDC and TCSC models, have not yet been tested. Thus, as discussed in Section 4, we recommend that these program features be exercised only by advanced users who are comfortable making changes to the source code.

5.1 Klein-Rogers-Kundur 2-area system with energy storage

Here we briefly highlight the key features of PSTess using the `d2asbegp_ess.m` base case. Fig. 6 shows a one-line diagram of the modified test case. In addition to the deployment of energy storage, the synchronous generation has also been redistributed so that G1 and G3 each make up 5 % of the system capacity, and G2 and G4 each 45 %. To disturb the system, we simulate a trip of G3 which is roughly equivalent to the loss of a large nuclear plant in the Western

Interconnection. We evaluate three different scenarios: open-loop operation, control via `lsc.m`, and control via `ess_sud.m`. In this case, we have set up `ess_sud.m` to execute the same control strategy as `lsc.m`. The ESSs in the system automatically adjust their charging/discharging behavior in response to the trip. This control strategy modulates active power only, so the reactive power output of each ESS is held at zero.

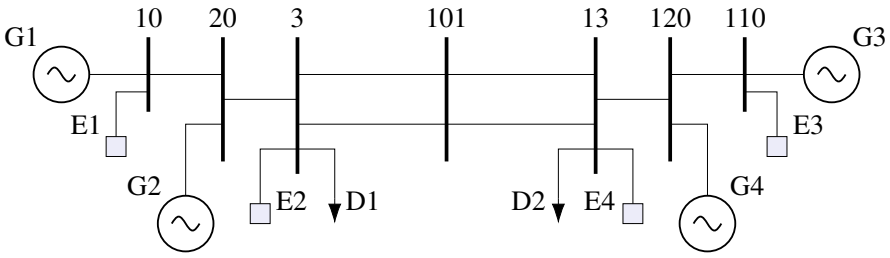

Figure 6: 2-area system with energy storage.

Fig. 7 shows the speed deviation of generator G4 in response to the trip of its neighbor G3. We observe that the nadir and settling frequency are effectively unchanged between open- and closed-loop operation. This is expected based on the design of the control strategy, which employs wide-area feedback and washout filters to avoid responding to steady-state changes. Fig. 8 shows the response of ESS E4 to the loss of generation. Over the first swing of the transient, E4 injects real power into the system aiming to keep G4 in synchronism with the remaining generators. As time elapses, the output of E4 and the other ESSs goes to zero. As expected, the results of the simulations conducted using `lsc.m` and `ess_sud.m` are identical. Recall that the user is free to implement any strategy they choose in `ess_sud.m`. This exercise demonstrates that it is possible to achieve equivalent performance with a user-defined model and a built-in PSTess model. In contrast to built-in models, user-defined models may be implemented without making modifications to the time-domain simulation routine `s_simu.m`. At present, user-defined models are not incorporated in the linearization routine specified in `svm_mgen.m`.

Fig. 9 shows the rotor angle difference between generators G2 and G4. This combination of states provides valuable information about how well the remaining generators are able to

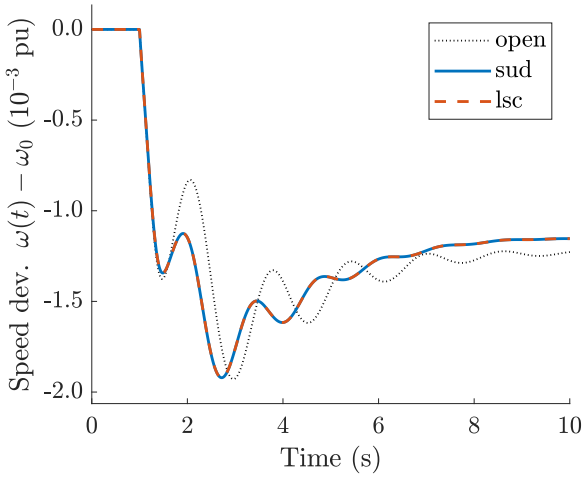

Figure 7: Speed deviation of generator G4.

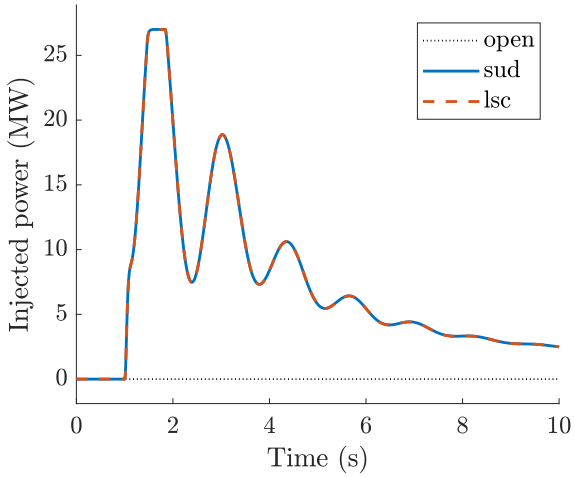

Figure 8: Real power injected by ESS E4.

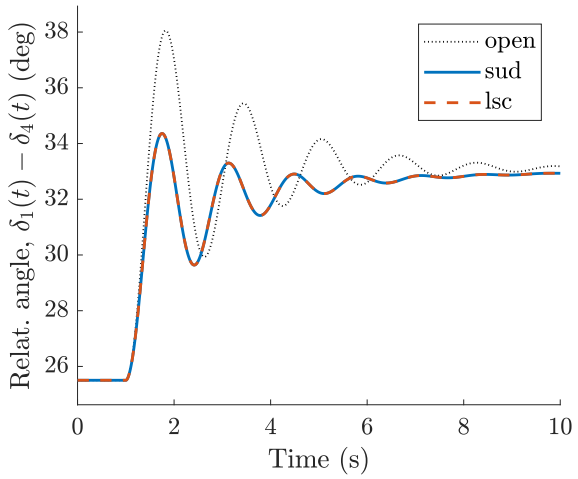

Figure 9: Relative angle between G2 and G4.

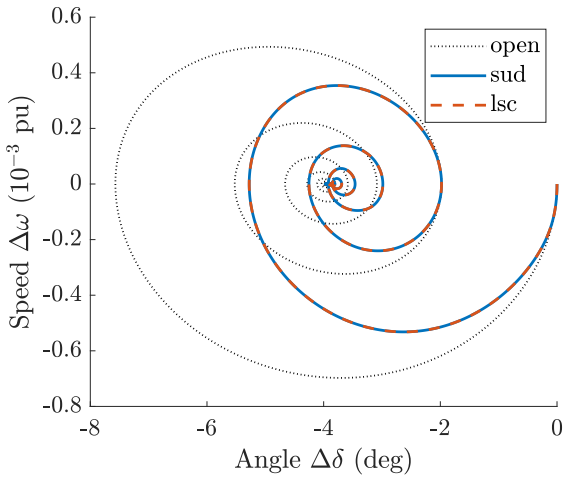

Figure 10: G4 center-of-inertia phase plane.

maintain synchronism following the disturbance. In closed loop, the peak angle excursion is reduced by roughly 33 % in relation to the open-loop response. Furthermore, the oscillatory component of the response increases in frequency, which is a direct byproduct of the synchronizing torque produced by the ESSs. Fig. 10 shows the phase plane response of generator G4 in the center-of-inertia reference frame, where the contour begins at the origin. The key takeaway is that the control strategy is effective in reducing both angle and speed deviations, which means that it helps to hold G4 in synchronism with the remaining machines. Finally, we observe that the post-disturbance equilibrium is effectively the same between the open- and closed-loop cases. The objective of this control strategy is not to change the post-disturbance equilibrium; rather, it is to ensure that the system is able to navigate to it safely.

6 Closing remarks

This document describes the Power and Energy Storage Systems Toolbox, Version 1.0. Its primary purpose is to provide an open source tool for simulation and dynamic analysis of power systems containing high penetrations of energy storage and/or inverter-based renewable resources. The control services provided by these resources will play an increasingly important role in maintaining the stability of the grid in the coming decades.

This toolbox owes a great deal to the original developers of PST: Drs. Joe Chow, Kwok Cheung, and Graham Rogers. We thank them for their contributions and for their generosity in releasing PST under the MIT license, enabling others to build upon the foundation they created.

The `ess.m` modeling framework implemented in PSTess Version 1.0 is designed to represent battery-based energy storage systems coupled to the grid via PLL-driven inverters. The model development has been informed by, and will continue to track, efforts conducted in the WECC Renewable Energy Modeling Working Group [4]. Over time, we intend to continually refine the energy storage device model and to introduce support for inverters whose controls specify a commanded voltage phasor at their terminals.

References

[1] J. Chow and K. Cheung, "A toolbox for power system dynamics and control engineering education and research," *IEEE Trans. Power Syst.*, vol. 7, no. 4, pp. 1559–1564, 1992.

[2] J. Chow, "Power System Toolbox Version 3.0 manual." Available at `https://www.ecse.rpi.edu/~chowj/PSTMan.pdf`.

[3] R. T. Elliott, P. Arabshahi, and D. S. Kirschen, "Stabilizing transient disturbances with utility-scale inverter-based resources," *IET Gen. Tran. Dist.*, vol. 14, pp. 6534–6544, Dec. 2020.

[4] R. Elliott, A. Ellis, P. Pourbeik, J. Sanchez-Gasca, J. Senthil, and J. Weber, "Generic photovoltaic system models for WECC–A status report," in *IEEE Power Energy Soc. Gen. Meeting*, pp. 1–5, 2015.

[5] CAISO, "Inverter-based interconnection requirements (ER19-1153)." Available at `http://www.caiso.com/Documents/`.

[6] M. Klein, G. Rogers, and P. Kundur, "A fundamental study of inter-area oscillations in power systems," *IEEE Trans. Power Syst.*, vol. 6, no. 3, pp. 914–921, 1991.

[7] D. Trudnowski, D. Kosterev, and J. Undrill, "PDCI damping control analysis for the western North American power system," in *IEEE Power Energy Soc. Gen. Meeting*, pp. 1–5, 2013.