Toolbox for Power System Computing - TPSC

Faruk Selimović

August 18, 2023

Next Steps (Instead of Introduction)

- Power flow - check and documentation.
- Device distribution - check and documentations
- Use information if a generator is in-service or out-of-service.

Chapter 1

DATA

1.1 General Data of Power Systems

The power system is represented as a structure within the Matlab software package. When loading a power system, the variable of structural type named "data" is loaded into the workspace. The names and descriptions of the structure's fields are displayed in Table 1.1.

Table 1.1: Fields of the structure used to describe the power system (*data*).

Field	Description
data.nBuses	number of nodes
data.fn	nominal system frequency
data.nBranches	number of branches
data.nGens	number of generators
data.slackNo	number of the slack bus
data.baseMVA	base power
data.bus	matrix whose rows represent groups of information used to describe nodes in the system
data.branch	matrix whose rows represent groups of information used to describe branches in the system.
data.generator	matrix whose rows represent groups of information used to describe generators in the system.
data.gencost	matrix whose rows represent groups of information used to describe generators in the system from an economic perspective

The matrix containing information about the nodes of the corresponding power system consists of fifteen columns. The column number, description, and unit of the quantity are shown in Table 1.2.

Table 1.2: Bus data (*data.bus*).

Column	Description	Unit
1	bus number	
2	bus type (1=PQ, 2=PV, 3=slack)	
3	active power demand	p.u.
4	reactive power demand	p.u.
5	active power of shunt conductance at the voltage 1 [p.u.]	p.u.
6	reactive power of shunt conductance at the voltage 1 [p.u.]	p.u.
7	area number	
8	voltage magnitude	p.u.
9	voltage angle	rad
10	base voltage	kV
11	loss zone	
12	maximum voltage magnitude	p.u.
13	minimum voltage magnitude	p.u.
14	active power generation	p.u.
15	reactive power generation	p.u.

The matrix containing information about the branches of the corresponding power system consists of thirteen columns. The column number, description, and unit of the quantity are shown in Table 1.3.

Table 1.3: Branch data (*data.branch*).

Column	Description	Unit
1	from bus	
2	to bus	
3	resistance	p.u.
4	reactance	p.u.
5	total line susceptance	p.u.
6	long term rating (0=unlimited)	p.u.
7	short term rating (0=unlimited)	p.u.
8	emergency rating (0=unlimited)	p.u.
9	transformer off nominal turns, taps at “from bus”, pi equivalent at “to bus”; value 0 indicates transimission line rather that transformer	p.u.
10	transformer phase shift angle (positive $\Rightarrow$ delay)	rad
11	branch status (0=in service, 1=out of service)	
12	minimum bus voltage angle difference	rad
13	maximum bus voltage angle difference	rad

The matrix containing information about the generators of the corresponding power system consists of ten columns. The column number, description, and unit of the quantity are shown in Table 1.4.

Table 1.4: Generator data (*data.generator*).

Column	Description	Unit
1	bus number	
2	active power output	p.u.
3	reactive power output	p.u.
4	maximum reactive power output	p.u.
5	minimum reactive power output	p.u.
6	regulated voltage magnitude	p.u.
7	base power of machine	p.u.
8	machine status (0=in service, 1=out of service)	
9	maximum active power output	p.u.
10	minimum active power output	p.u.

Several power systems are already available in the directory 'data/power_systems'. To define a new power system, a user has to write a script with the structure and fields as previously defined. In the case that a user wants to use one of the cases available in the Matpower package, the script "data/matpower2tpsc" can be employed. For example, a user needs to place a Matpower case 'caseName.m' in the current working directory or in one of the directories defined in a list of Matlab search paths. After running the script, the power system will be saved in the 'data/power_systems' directory, and the prefix "TPSC" will be added to specify that this structure is part of the TPSC package. However, users do not have to worry about this because the only requirement in every routine is to enter the 'caseName'.

1.1.1 Extended Verison with Measurement Devices

This is a structure that extends the structure described in the previous subsection with data about measurement devices installed in a power system. Structure *data* is now enriched with two new fields:

data.pmu data about phasor measurement units installed in the power system, and it is presented in detail in Table 1.5.

data.scada data about conventional SCADA measurement devices, and it is presented in detail in Table 1.6.

Table 1.5: PMUs data (*data.pmu*).

Column	Description	Unit
1	bus number	
2	number of current channels, -1 for all incident branches	
3	magnitude measurement standard deviation	%
4	phase angle measurement standard deviation	rad
5	frequency measurement standard deviation	Hz
6	RoCoF measurement standard deviation	Hz/s
7	Reporting frequency	Hz

Table 1.6: SCADA data (*data.scada*).

Column	Description	Unit
1	bus number	
2	measurement type (1-active power flow, 2-reactive power flow, 3-active power injection, 4-rective power injection, 5 - branch current magnitude, 6- bus voltage manitude)	
3	line number (type 1, 2 and 6) or bus number (type 3, 4 and 5)[1]	
4	standard deviation	p.u.
5	Reporting frequency	Hz

Similarly to power system data (structured as 'data'), users can input information about power system measurement devices while adhering to the format outlined in Tables 1.5 and 1.6.

On the other hand, a recommended approach involves utilizing the 'run_device_distribution.m' script. This script offers various options for configuring measurement devices, considering their type, standard deviation, and reporting frequency (important for tracking and dynamic modes). Detailed information are available in the Chapter about routines.

1.2 Measurement Data

Measurements from PMUs are separeted in two different tables: synchrophasor measurements 1.7, and frequency and rocof measurements 1.8.

[1]Line number corresponds to a row number of the given line in *data.branches* matrix. When a measurement device is on a "to" side of a line, its number is given with a minus sign.

Table 1.7: PMU measurements (*measurements.synpmu*).

Column	Description	Unit
1	time index	
2	bus number	
3	phasor measurement type (1-current flow, 2-current injection, 3-voltage)	
4	line number (type 1) or bus number (type 2 or 3)	
5	magnitude measurement value	p.u.
6	phase angle measurement value	rad
7	magnitude exact value	p.u.
8	phase angle exact value	rad

Measurements from a SCADA system are collected in a matrix whose columns are described in Table 2.2.

Table 1.8: PMU measurements (*measurements.fpmu*).

Column	Description	Unit
1	time index	
2	bus number	
3	frequency measurement	Hz
4	frequency exact value	Hz
5	rate of change of frequency (rocof) measurement	Hz/s
6	rate of change of frequency (rocof) exact value	Hz/s

Table 1.9: SCADA measurements (*measurements.scada*).

Column	Description	Unit
1	time index	
2	bus number	
3	measurement type	
4	line number (type 1, 2 and 6) or bus number (type 3, 4 and 5)	
5	measurement value	p.u.
6	exact value	p.u.

Chapter 2

Routines

2.1 Power Flows

2.1.1 Basics

Power flows, also known as load flow analysis, are fundamental calculations in electrical power systems engineering. They involve solving a set of equations to determine the steady-state operating conditions of an electrical network. The goal of power flow computations is to find the voltages, currents, and power flows at various nodes and branches of the network under given load and generation conditions.

Key aspects of power flow computations include:

1. **Voltage Magnitudes and Angles**: Power flow calculations determine the magnitude and phase angle of voltages at each node in the network. These values indicate the electrical potential and phase relationship at different points in the system.

2. **Active and Reactive Power Flows**: Power flow computations calculate the active (real) and reactive power flows in each transmission line, transformer, and generator. This information is essential for maintaining the proper balance between generation and consumption in the system.

3. **Load Balancing**: Power flow analysis ensures that the supply of power from generators matches the demand from loads, resulting in a balanced system. Deviations from this balance can lead to voltage instability and other issues.

4. **Voltage Stability Assessment**: Power flow computations help assess voltage stability by identifying voltage limits that should not be exceeded to prevent voltage collapse.

5. **Contingency Analysis**: Power flow analysis can be extended to evaluate the impact of potential network failures, such as line outages or equipment failures, to understand their effects on power flows and system stability.

6. **Generator Dispatch**: Power flow calculations assist in determining the optimal settings for generator output to minimize losses and improve overall system efficiency.

7. **Optimal Power Flow (OPF)**: Advanced versions of power flow analysis, such as optimal power flow, optimize the generator outputs and other control variables to meet various objectives while satisfying network constraints.

Power flow computations are critical for planning, designing, and operating power systems effectively and reliably. They help ensure that the system operates within acceptable limits, avoids overloads, maintains voltage stability, and meets load demands while efficiently utilizing available generation resources.

2.1.2 Run Settings

To run a power flow analysis in the TPSC package a user is provided with the function:

```
[ results ] = run_power_flows(pfsettings, data);
```

Function arguments are structures: 'pfsettings' whose fileds are user-defined and used to set the computations, and 'data', which presents power system data from Table 1.1. The function's output 'results' is already described in Chapter 1.

'pfsettings' contains the following fields:

domain	a field used to determain whether the computations will be done in 'complex' or 'real' domain. In power flow analysis, state variables are complex nodal voltages. The classical approach involves presenting complex numbers in either their polar or rectangular forms and conducting computations in the real domain. A recently introduced alternative is to utilize Wirtinger calculus and perform calculations with complex numbers.
method	used to specify the method for conducting power flow computations.
start	used to select whether the inital values of state variables will be set to $1\angle 0$[p.u.], or they will be assigned predefiend values (columns 8 and 9 from data.bus).
maxNumberOfIter	used to select the maximum number of iterations that can be conducted before convergence. If this limit is reached, TPSC notifies a user that the power flow analysis did not converge.
eps	tolerance value when checking convergence.
postprocess	used to determine whether postprocessing of power flow results needs to be performed (=1) or not (=0).

Currently available power flows solvers are shown in Table 2.1.

Table 2.1: Power flow solvers.

No.	domain	method	description
1	'complex'	'cnr_pf'	...

TPSC provides a specialized script for conducting power flow anaysis in the root directory of package called 'power_flows.m'. Users input the name of the desired power system (which must be stored in 'src\power_systems'), and they can customize computation specifics using the 'pfsettings' structure.

In addition, TPCS implements the function:

```
results_pf(data, pfsettings, results);
```

This function is used to display the results of power flow computations. It is interactive, allowing a user to decide what information to print to the terminal.

Table 2.2: Scada measurements.

measurement type	notation	maximum number
active power flow	Pij	$2 \cdot data.nBranches$
reactive power flow	Qij	$2 \cdot data.nBranches$
active power injection	Pi	$data.nBuses$
reactive power injection	Qi	$data.nBuses$
current magnitude	Iij	$2 \cdot data.nBranches$
voltage magnitude	Vi	$data.nBuses$

2.2 Device Distribution

2.2.1 Basics

To be done!

2.2.2 Run Settings

To distribute measurement devices across a power system, TPSC provides users with a function with the following calling syntax:

```
data = distribute_devices(name, ddsettings);
```

However, it is highly recommended to use the script 'run_device_distribution.m,' which includes the function call and all necessary settings.

Two parameters to be inserted on the beginning are (example below):

name	a sequence of characters that denotes the corresponding power system. It is important to emphasize that a prerequisite for running this function is that the suitable power system has already been entered and stored. If the case is generated from the Matpower base (using 'matpower2tpsc.m'), the name of the case will be exactly the same.
vrs	enables a user to generates multiple different measurement scenarios on the same power system case. It is recommenden to label versions with capital letters of the alphabet.

```
name = 'case118';
vrs = 'A';
```

TPSC is equipped with many different possibilities when it comes to distributing measurement devices. A user specifies the features by means of a structure called 'ddsettings'. This routine present the initial point when it comes to investigating the state estimation applications.

data.scada The set of SCADA measurements consists of six different electrical quantities. Types, labels and the maximum possible number that can be present in a power system can be found in Table 2.2.

- To specify the types and quantity of SCADA measurements, a user utilizes a field called 'scadaset'. To include all possible SCADA measurements, the 'scadaset' should only have one entry: "complete". To individually define the number of measurements for a specific type, a user needs to input pairs of type labels and corresponding numbers. This configuration has two options concerning how the device count is entered. The first option is to input the exact number of devices. This approach requires that the first element of 'scadaset' is the string "num". The second option involves specifying a percentage of devices based on the maximum possible number. In this case, the first element of 'scadaset' must be the string "perc". In essence, if a user wishes to input the number of devices per measurement type, the first element of 'scadaset' must be either 'num' or 'perc'. In both cases, positions of the measurement devices are randomly chosen. The following examples illustrate how to use the 'scadaset' field and describe the expected outcomes.

```
ddsettings.scadaset = [ "complete" ] % deploys all possible measurements
ddsettings.scadaset = [ "num", "Pij", 150, "Qij", 150, "Vi", 200 ]; % deploys
    150 devices for measuring both active and reactive power flow, and 200
    devices for measuring voltage magnitude
ddsettings.scadaset = [ "perc", "Pi", 40, "Qi", 40, "Iij", 60, "complete" ]; %
    deploys 40% of both active and reactive power injection measurement devices
    , 60% of current flow magnitude measurement devices, and all possible
    measurement devices (100%) for the remaining measurement types.
```

- Standard deviations of measurement devices are configured using a field called 'scadasd'. This field follows a somewhat similar behavior to the previous one. The standard deviations can be set for all devices using the identifier "complete", or individually for specific measurement types using the labels from Table 2.2. TPSC offers two options for inputting standard deviations: using a single fixed number (identified as "fixed"), or specifying a range between two numbers from which the value will be randomly chosen (identified as "rand"). One of these options (identifiers) must be the initial entry in the field. The standard deviation for all SCADA measurements is given in per units.

```
ddsettings.scadasd = [ "fixed", "Pij", 0.01, "Vi", 0.005 ]; % assigns a
    standard deviation value of 0.01 to active power flow measurements and a
    value of 0.05 to active power injection measurements
ddsettings.scadasd = [ "rand", "Qij", 0.01, 0.02, "Vi", 0.005, 0.1, "complete"
    0.01, 0.03 ]; % assigns random values of standard deviation in the range
    between 0.01 and 0.02 for reactive power flow measurements, between 0.005
    and 0.1 for voltage magnitude measurements, and between 0.01 and 0.03 to
    measurement devices of the remaining types
```

Standard deviation must be attached to all devices which are deployed using 'scadaset', otherwise, the program will crash.

- The reporting frequency, denoted as the number of measurements conducted in one second (with a unit of $s^{-1} = Hz$), for devices is established using a field called 'scadafreq'. The identifiers remain consistent: "complete" or the labels of measurement types.

```
ddsettings.scadafreq = [ "Pij", 0.2, "Vi", 1 ]; % assigns a reporting frequency
     value of 0.2 [Hz] to active power flow measurement devices and a value of
    1 [Hz] to voltage magnitude measurement devices
ddsettings.scadafreq = [ "Iij", 0.5, "complete", 2 ];  % assigns a reporting
    frequency value of 0.5 [Hz] to current magnitude measurement devices and a
    value of 2 [Hz] to measurement devices of the remaining types
```

Although the reporting frequency is not important for static state estimation, it must be defined for all device from 'scadaset', otherwise, the program will crash.

data.pmu As previously mentioned, PMUs measure voltage and current synchrophasors, as well as frequency and rate of change of frequency at the connected bus. Given the fact that power systems are sparsely connected, i.e. each bus has only a limited number of neighbors irrespective of the system size, channel limits on PMUs may be assumed to be sufficient to monitor as many signals as needed at a given bus. Assuming that a PMU is placed at bus k, the following quantities are usally assumed to be available: voltage phasor at bus k, current phasors along all lines/branches incident to bus k. Hence, TPSC does not provide an option for a user to specify the number of measurements from a specific category (such as nodal voltage magnitude/phase or branch current magnitude/phase). Instead, a user only specifies the number of PMUs deployed in the power system.

- The number of PMUs and the number of available current channels per PMU are defined by a field called 'pmuset'. A user can use the identifier "complete" to deploy PMUs on all buses in the power system. Alternatively, the number of PMUs can be determined by entering a single number preceding the identifier "num," representing the desired number of PMUs. Another option is to enter the percentage of buses equipped with a PMU using the identifier "perc". In most cases, PMU has the ability to measure current flow on all incident branches, and this is a default option. Alternatively, TPSC allows a user to set an upper limit to the number of current channels of PMUs. This number is entered after the identifier "currCh".

```
ddsettings.pmuset = [ "num", 120 ]; % deploys the PMUs on 120 randomly chosen buses in the power system, and each PMU measures nodal voltage phasor and current flow phasor on all incident branches
ddsettings.pmuset = [ "perc", 55, "currCh", 4 ];  % deploys the PMUs on 55% randomly chosen buses in the power system, and each PMU measures nodal voltage phasor and current flow phasor on up to four incident branches
```

- The standard deviation of PMUs is set up using a field called 'pmusd'. It should be defined for all types of quantities that a PMU measures independently: "magnitude" in percentages, "phase" in degrees, "frequency" in mHz, and "rocof" in Hz/s. Similar to the SCADA set, TPSC allows a user to choose it with a single number ("fixed") or a random number from a provided range ("rand").

```
ddsettings.pmusd = [  "fixed", "magnitude", 0.01, "phase", 0.2, "frequency", 5, "rocof",  0.4];
 % the standard deviation for magnitude measurements is 0.01%, for angle measurements is 0.2 degrees, for frequency measurements is 5 mHz, and for rate of change of frequency (rocof) is 0.4 Hz/s.
ddsettings.pmusd = [  "rand", "magnitude", 0.01, 0.02, "phase", 0.2, 0.3,  "frequency", 5, 5, "rocof",  0.4, 0.4 ];  % the standard deviation for magnitude measurements is a randomly selected number from the range between 0.01% and 0.02%, for angle measurements it is from the range between 0.2 and 0.3 degrees; for frequency measurements, it is 5 mHz, and for the rate of change of frequency (rocof), it is 0.4 Hz/s.
```

- In order to specify the repoting frequency of PMUs, a user should use a field called 'pmufreq'. To set that all PMUs has the equal reporting frequency a user is supposed to use the following syntax:

```
ddsettings.pmufreq = [ "complete", 25 ]; % all pmus have the reporting frequency equal to 25 Hz
```

The substitute approach is to specify the percentage of devices included in a set of one of the four standard reporting frequencies according to the latest IEEE standard for PMUs: 100, 50, 25 and 10 [Hz].

```
ddsettings.pmufreq = [ 50, 20, 10, 20 ]; % 50% of PMUs report 100 times per
    second, 20% report 50 times per second, 10% report 25 times per second, and
     the remaining 20% report 10 times per second
```

The dependable performance of these features is anticipated only when input parameters fall within permissible ranges; otherwise, the program may crash.

2.3 Measurements Generator

2.3.1 Basics

All state estimation algorithms require measurements to operate. In practice, these measurements are collected using telecom infrastructure and provided as inputs to various algorithms within the Energy Management System. When modeling and analyzing power systems, different tools are employed to simulate this behavior and generate measurements. The TPSC generates the measurements by running power flows and perturbing the true quantities with random Gaussian noise. Primarily, state estimation algorithms are categorized into static and dynamic types. Static state estimation algorithms estimate the states based on a set of measurement values and the known system topology. They operate at a single time instant, independently of the past or future. The states are associated with different measurement values through the use of algebraic equations. On the other hand, the dynamic state estimation algorithms are tracking the state of the system on a specific time interval and usally uses a combination of differential and algebraic equations.

2.3.2 Run Settings

To generate measurements for a specific power system equipped with measurement devices, the TPSC provides users with the function:

```
[ measurements ] = generatemeasurements(mgsettings, pfsettings, data);
```

Users are recommended to use the script 'run_measurement_generator.m' which contains the easy to adjust example on how to set all required paramteres.

The first parameter to be specified is a power system with distributed measurement devices for which the measurements will be generated. This includes entering the name and version of power system (function argument 'data'). Further, as the power flows are in the heart of the measurement genertion, users have to enter settings for power flow analysis using the variable 'pfsettings'.

Users define the behavior of the measurement generator using a variable called 'mgsettings'. Initially, users specify whether to compute measurements for static or dynamic state estimation by setting a field named 'mode' to either 'static' or 'tracking'. If the selection is 'static,' no other parameters need to be set. On the other hand, the 'tracking' mode is equipped with some additional features. First, a user needs to select a time interval duration for generating measurements.

Quasi-dynamics are supported in terms of changes in load and frequency. The frequency curve over the time interval is configured using a field named 'fdynamics'. To choose a constant frequency, users need to input the string “const”. The second argument is optional and represents the frequency value. If it is not entered, the frequency is equal to its nominal value To simulate random changes in frequency, users should enter the string “random” followed by the step size (e.g., 0.01, i.e., 10 [mHz]) . The third option is to select on of two predefied frequency curves which

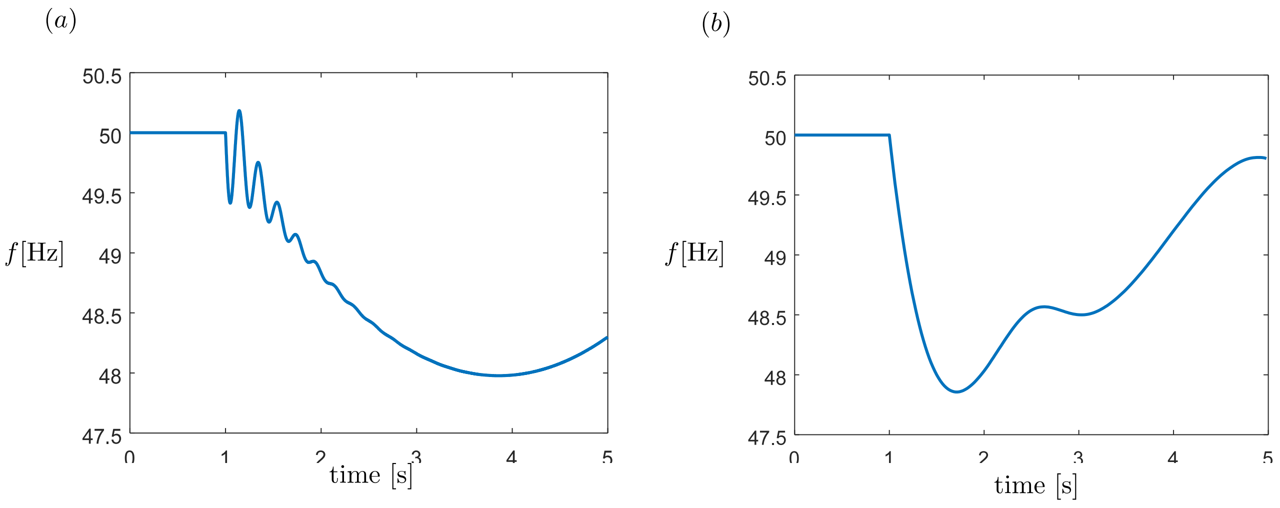

Figure 2.1: Predefined frequency curves: a) “UD1”, b) “UD2”.

are shown in Fig 2.1.This can be done by entering the string “UD1” or “UD2”, along with a value that determines the minimum frequency value reached during the interval. In Fig 2.1, this value is 48 [Hz]. Examples for setting this field are shown below.

```
mgsettings.fdynamics = [ "const", 49.95 ]; % constant frequency equal to 49.95 Hz
mgsettings.fdynamics = [ "random", 0.005 ]; % random change in frequency with step 0.005 Hz
mgsettings.fdynamics = [ "UD1", 49.5 ]; % frequency curve shown in Fig 2.1 a) with the minimum value 49.5 Hz
```

2.4 Static State Estimation

2.5 Dynamic State Estimation