Data Analitics for Smart Grids

MEEC

Lab 3 - State Estimation

Authors:

Rodrigo Contreiras (90183) rodrigo.contreiras@tecnico.ulisboa.pt
Gonçalo Aniceto (96218) goncalo.aniceto@tecnico.ulisboa.pt

2022/2023 – 2nd Semester, P3

Contents

1 Problem description

State estimation is a critical function in power systems that involves estimating the system's state variables, such as voltage magnitude and phase angles, based on available measurements. State estimation is necessary to monitor the performance of the power system and ensure the system's stability and reliability.

Data analytics can be applied to power system state estimation to improve the accuracy and efficiency of the estimation process. Data analytics techniques such as regression can be used to process large volumes of data from various sources, including sensors and smart meters.

2 Objectives

Using the grid's static knowledge about its structure and related component models, let's take a grid for which we have sufficient metering data (z_{meter}) and figure out the state vector x that fully represents the condition in which the grid is being operated.

3 Note on Observability

In discrete LTI systems terminology, the observer model of the system is described by the equations:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases}$$

In this system the scalar $k \in \mathbb{Z}$ is the time epoch. The state vector is x and u y are the control input and the measurement. The matrices A and C dictate system dynamics and current state observability. B and D reflect controller dynamics. Note that during the measurement the measurement controller dynamics are negligible, D, therefore the system can be written in the form $y(k) = Cx(k)$. In state estimation terminology, this equation is often presented as $z = H_x x$.

In general, a system is observable if a state estimation for any instance, from the initial state $s = 0$ to the current state $s = k$, can be obtained from the current observation $y(k)$. The observability matrix, $\mathcal{O}$, maps the state in a given time , $x(s)$, into the resulting output over the time epochs, $y(t)$, $t \in [s, k] \subset \mathbb{Z}$.

The inverse mapping is only possible if $Rank(\mathcal{O}) \geq size(x)$ allowing the left inverse: $x(k) = [\mathcal{O}^*\mathcal{O}]^{-1}\mathcal{O}^* y(k)$. In the context of the grid, we are interested in estimating the current state, $s = k$, $\mathcal{O} = C = H_x$. In the context of state estimation the expression for the estimated state becomes $\hat{x} = [H_x^* H_x]^{-1} H_x^* z$.

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-s-1} \end{bmatrix}$$

4 State Estimation Problems

We used the same "kite" grid for this assignment as we did for the last one. As a result, everything we previously knew about the network structure still holds true and is assumed moving forward.

4.1 Incomplete State Estimation

First, we made the decision to only estimate the state, disregarding pseudo measurements, and to only use the measurements we could access and were being monitored. As a result, we will **ignore V3** estimation and just **take into account complete information about currents I12 and I54 (amplitude and angle)**.

Our z_{meter} vector is composed by the metering information, to which we have information. It's values are the following:

$$z_{meter} = \begin{bmatrix} I_{12} \\ I_{54} \\ V_2 \\ V_5 \end{bmatrix} = \begin{bmatrix} -0.1117607 + 0.04431433j \\ -2.4472 + 0.80435574j \\ 0.87962652 - 0.25388267j \\ 1 + 0j \end{bmatrix}$$

The structural model that lets us represent the physical structure of the grid and lets us calculate the state, it is the following:

$$H_x = \begin{bmatrix} y_{12} & -y_{12} & 0 & 0 \\ 0 & 0 & -y_{45} & y_{45} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 + 10j & 1 - 10j & 0 + 0j & 0 + 0j \\ 0 + 0j & 0 + 0j & 2 - 20j & -2 + 20j \\ 0 + 0j & 1 + 0j & 0 + 0j & 0 + 0j \\ 0 + 0j & 0 + 0j & 0 + 0j & 1 + 0j \end{bmatrix}$$

Finally, the state which we estimate, based on this measurements was the following:

$$x = \begin{bmatrix} V_1 \\ V_2 \\ V_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} 0.88512062 - 0.243256005j \\ 0.87962652 - 0.253882666j \\ 0.94806556 - 0.117166556j \\ 1 - 1.42108547e^{-14}j \end{bmatrix}$$

As it is visible, by taking into account only the monitored variables, we weren't capable of estimating the entire state (V_3 is missing). We need to find a proper way to find more information, capable of making the entire state observable.

4.2 Complete State Estimation

In this section, data on network loads and the current arguments are used as the pseudo-measurements to make the entire state observable. With these new factors taken into account, we can now calculate the final state variable, the V_3, which was previously impossible for us to compute. With this adjustments our z_{meter} vector is now in the form:

$$z_{meter} = \begin{bmatrix} I_{12}\angle-\phi \\ I_{54}\angle-\phi \\ V_2 \\ V_5 \\ |S_1|\angle-\phi \\ |S_2|\angle-\phi \\ |S_3|\angle-\phi \\ |S_4|\angle-\phi \end{bmatrix} = \begin{bmatrix} 0.1142144 - 0.03754046j \\ 2.4472 - 0.80435574j \\ 0.87962652 - 0.25388267j \\ 1 + 0j \\ -0.2128 + 0.06994398j \\ -0.6726 + 0.22107293j \\ -1.4934 + 0.49085684j \\ -0.0684 + 0.02248199j \end{bmatrix}$$

The structural model, in turn also had to be updated, taking consideration the added loads as pseudo-measurements:

$$H_x = \begin{bmatrix} y_{12} & -y_{12} & 0 & 0 & 0 \\ 0 & 0 & 0 & -y_{45} & y_{45} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ y_{12}+y_{13} & -y_{12} & -y_{13} & 0 & 0 \\ -y_{12} & y_{12}+y_{23} & -y_{23} & 0 & 0 \\ -y_{13} & -y_{23} & y_{13}+y_{23}+y_{34} & -y_{34} & 0 \\ 0 & 0 & -y_{34} & y_{34}+y_{45} & -y_{45} \end{bmatrix}$$

We evaluated two distinct models to estimate the state. One in which the pseudo-measurements were given the same importance as the real-time ones and one in which the pseudo-measurements were given less importance by having less weight attached. To see how these two models would respond and which one was more accurate, we ran 100 experiments with a small normally distributed error of $\epsilon \sim \mathcal{N}(0, 0.5)$, over the load measurements. First, we used the following vector to represent the diagonal of the weight matrix:

$$W_z = diag \begin{bmatrix} 1.e^6 & 1.e^6 & 1.e^6 & 1.e^6 & 4 & 4 & 4 & 4 \end{bmatrix}^T$$

After running the simulations, we obtained the following plot:

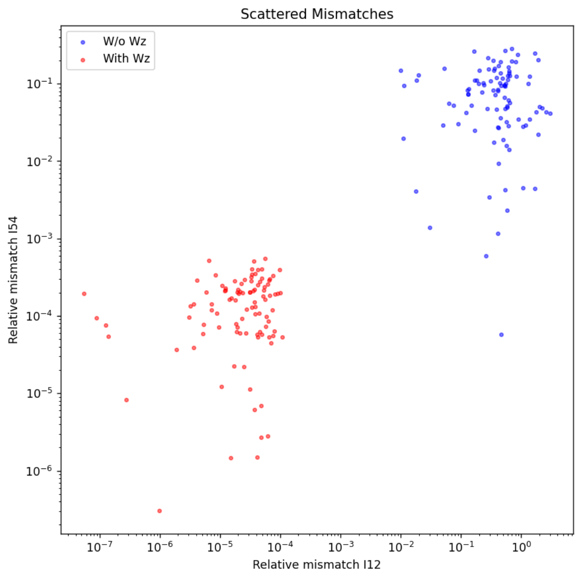


Figure 1: Scattered mismatches between real-time measured variables (I_{12} and I_{24}) and the obtained power-flow results from the estimate state. Red dots depict weighted estimation, while the blue circles were obtained through unweighted estimation

Regarding the results, it is possible to draw the same conclusions as those made in the lecture notes. When calculating the mismatches by using the weights, we obtain values that are negligible for both currents, however if we don't use it the mistakes are too big. If we do not use weights, the load volatility is projected to all state variables, but it is mostly projected to V_3 if we do.

We can therefore conclude that choosing appropriate weights to use both on the real-time measurements and on the pseudo-measurements is essential to having a proper state estimation. Pseudo-measurements are a great way to achieve full state observability. However, they bring added difficulties that need to be addressed in order to reduce the error probability. Note that the assignment of high weights to real measurements and low weights to pseudo-measurements may cause ill-conditioned system. The use of Lagrange multipliers was proposed to handle pseudo-measurements and will be implemented in the section "Modelling pseudo-measurements as equality constraints".

4.3 Chose just two out of four possible pseudo-measurements

This section aims to demonstrate how the results can be influenced by selecting the appropriate pseudo-measurements. Certain measures will directly affect the state values from which we wish to derive an estimate, while others won't. In order to determine which are useful and which are unnecessary, it is crucial to conduct this analysis.

We'll test this by determining which pair, out of the four possible pseudo-measurements, will produce the best outcome. The performance of each model will be evaluated for the purposes of this analysis using the inverse of the gain matrix. We'll do this by using the diagonal values from this matrix, which correspond to the squared difference between the real state variable and its estimated value. Hence, the model that produces the best final model will be the one with the lowest value. After running the simulation, we obtained the following results:

	W = 1	**W != 1**
S1-S2	4.99505864	3.03047259e-04
S1-S3	**2.36874202**	5.70646774e-05
S1-S4	2.46166175	3.03535344e-04
S2-S3	2.37202619	5.69918312e-05
S2-S4	2.46708931	3.01337082e-04
S3-S4	2.4731319	5.70484640e-05

Table 1: Sum of the inverse gain matrix diagonal values.

By analyzing table 1, it is obvious that the ideal load combination to produce the estimation depends on the weights used. Consequently, it is reasonable to draw the conclusion that the selection of the employed pseudo-measurements for state estimation depends not only on the grid but also on the assumptions we make about the measurement accuracy.

In the cases shown on table 1, when considering that the weights are the same and equal to 1, the best pseudo-measurements would be S1 with S3, while in the case where the measurements are different than 1, the best are S2 with S3. These results were expected since this power injections are the most closely related to the state variable that we wanted to estimate, V_3.

4.4 Adding Measurements

The effect of including additional measurements on state estimate performance will be examined in this section. Two experiments were conducted. First, the measurements, I_{43} and I_{32}, were added to z with the same noise level as the other measurements, $\epsilon \sim \mathcal{N}(0, 0.02)$. For the second experiment, the same measurements were introduced, but with a bigger error, $\epsilon \sim \mathcal{N}(0, 2)$. It can be concluded from the outcomes that expanding z resulted in lower error when this was not done with noisy measurements.

Original	One Added	Two Added
0.0091	0.0059	0.0050

Table 2: Average error, uniform noise

Original	One Added	Two Added
0.0091	0.0217	0.0273

Table 3: Average error, noisy measurements added

When there are more measurement points, there are typically more equations that can be used to solve the state estimation problem, leading to a more overdetermined system. This can help to mitigate the effects of measurement errors, as well as reduce the impact of unobservable variables. However, it is important to note that adding more measurement points also increases the computational complexity of the state estimation problem. In addition, if the measurements are not accurate or if the system is not observable, then adding more measurement points may not improve the accuracy of the state estimation. Therefore, while having more measurement points can generally be beneficial for state estimation, it is important to carefully consider the trade-offs between the benefits of additional measurements and the computational and practical limitations of the system.

4.5 Topology processing

Lets consider that a fault occurs and one line is disconnected, altering the grid topology. In this section, state estimation will be utilized as a technique to access topology changes.

To create the necessary data for the estimation, a linearized power flow with the revised admitance matrix was solved, yielding exact current and voltage values. A slight error that followed a normal distribution was then introduced.

A network model, H_x , will be built for every possible topological variation of the "kite" grid. State estimation will proceed to be done as before, for every H_x. The smallest error, $\sum G_{ii}^{-1}$,will indicate the most correct topology.

This method proved to be problematic. Considering that exact state values aren't available, as is common in a state estimation problem, the inverse of the gain matrix to be computed trought expression **(2)**:

$$R_{\hat{x}}(x) = E\{(x-\hat{x})(x-\hat{x})^T\} \qquad \mathbf{(1)}$$
$$R_{\hat{x}} = (H_x^* W_z H_x)^{-1} \qquad \mathbf{(2)}$$

The fact that the measurement jacobian is constant in linear state estimation leading to a constant error, is concerning. It is not evident, but errors are independent of both the grid state and on the grid topology. To exemplify consider the following two grids:

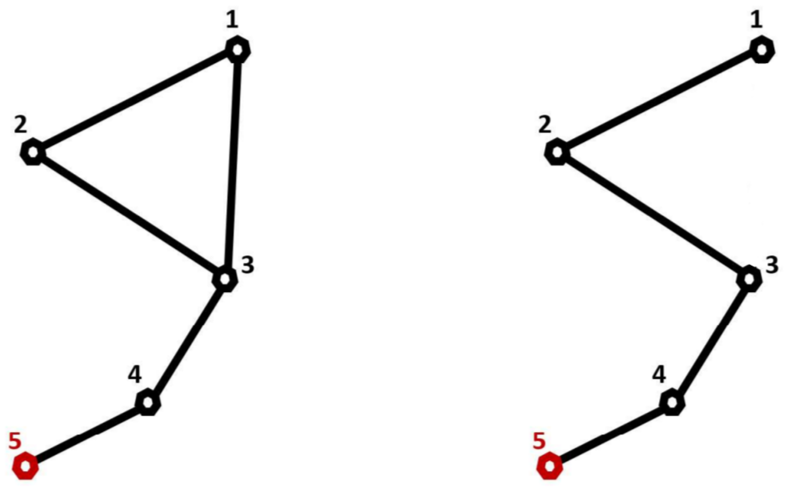


Figure 2: "Kite" grid (left) and radial grid (right)

Considering the usual measurements and pseudomeasuremnts, the measurement jacobian are obtained considering a linear mapping between state and measurement. Note that H_x^{radial} can be derived from H_x^{kite} (see section Complete State estimation) by considering null longitudinal admittance conecting buses 1 and 3, $y_{13} = 0$.

Let's consider that the grid assumes first the "kite" topology and then the radial topology. The calculated error will be the constant, regardless of the reality of the grid. As a result, the "kite" will always be considered the correct topology, leading to an incorrect estimation in the second experiment. Note that this method also doesn't show any sensibility towards the presence of noise in the measurements, ascribing the same error to an estimation done with exact measurements as to one done with erroneous ones.

"kite"	radial
5.5731e-05	1.0638e-04

Table 4: Error, "Kite" Grid

"kite"	radial
5.5731e-05	1.0638e-04

Table 5: Error, Radial Grid

Despite the fact that this strategy did not work for linear SE, it should not be dismissed because it may be applicable to non-linear SE. Note that the measurement jacobian is given by $H_x = \frac{\partial h}{\partial x}$, with h being a column vector of power flow equations relating the measurement variables with state variables. Both injected and transmited power, S_i and S_{ij} are quadratic w.r.t voltage. Therefore the will jacobian depend on the state, $H_x = H_x(x)$ and expression **(2)** may be valid for error analysis.

4.6 Estimation on meshed grids

A meshed grid is clearly more difficult to analyze than a radial grid in terms of power flow. This section will examine whether the same approach applies to state estimation. Consider the suggested "kite" grid, the radial grid obtained by disconnecting buses 1 and 3, and the meshed grid constructed by connecting buses 4 and 2.

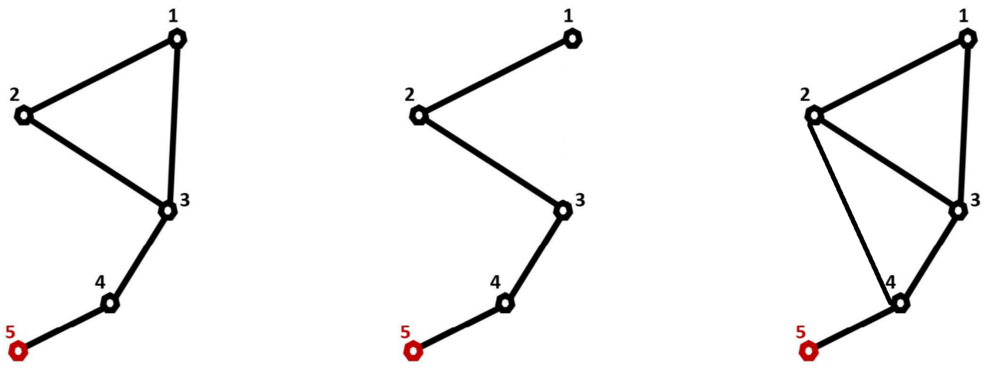


Figure 3: "Kite" grid (left), radial grid (center) and meshed grid (right)

Considering the usual measurements and the admittance $y_{24} = 1 - 10j$, the measurement jacobian for the meshed grid is given by:

$$H_x^{meshed} = \begin{bmatrix} y_{12} & -y_{12} & 0 & 0 & 0 \\ 0 & 0 & 0 & -y_{45} & y_{45} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ y_{12}+y_{13} & -y_{12} & -y_{13} & 0 & 0 \\ -y_{12} & y_{12}+y_{23}+y_{24} & -y_{23} & -y_{24} & 0 \\ -y_{13} & -y_{23} & y_{13}+y_{23}+y_{34} & -y_{34} & 0 \\ 0 & -y_{24} & -y_{34} & y_{34}+y_{45}+y_{24} & -y_{45} \end{bmatrix}$$

To generate the exact state a linear power flow was run. As can be seen, distinct grids had different nodal voltages for the identical load conditions:

$$x_{kite} = \begin{bmatrix} 0.9507 - 0.0450i \\ 0.9558 - 0.0407i \\ 0.9605 - 0.0362i \\ 0.9776 - 0.0197i \\ 1.0000 + 0.0000i \end{bmatrix} \quad x_{radial} = \begin{bmatrix} 0.9216 - 0.0716i \\ 0.9465 - 0.0498i \\ 0.9605 - 0.0362i \\ 0.9776 - 0.0197i \\ 1.0000 + 0.0000i \end{bmatrix} \quad x_{meshed} = \begin{bmatrix} 0.9577 - 0.0378i \\ 0.9657 - 0.0307i \\ 0.9661 - 0.0305i \\ 0.9776 - 0.0197i \\ 1.0000 + 0.0000i \end{bmatrix}$$

The exact measurement data was then obtained and an normally distributed error was added. The state was estimated and compared to the exact state 100 times. The average error for high and low noise level were:

Kite	Radial	Meshed
0.3170	0.2118	0.3220

Table 6: Average error, $\epsilon \sim \mathcal{N}(0, 0.2)$

Kite	Radial	Meshed
0.0042	0.0041	0.0051

Table 7: Average error, $\epsilon \sim \mathcal{N}(0, 0.002)$

Since in radial grids, power flows in a single direction from the substation to the end-users, it is easier to identify the sources and sinks of power. As a result, state estimation on radial grids can be performed more easily, with smaller error. On the other hand, meshed grids have multiple interconnections creating more complex power flow patterns. Identifying the sources and sinks of power can be more challenging, and power flows may have multiple possible paths through the network, making state estimation more difficult. Furthermore, to increase estimation accuracy on meshed grids, more measurement will be needed , as to capture all the grid's dynamics, which can increase the size of the state estimation problem and require more computationally intensive algorithms.

4.7 Bad data

Lets consider that a one (or more) of the measurements is erroneous. In this section, a method for identifying and eliminating bad data using the residual analysis based on chi-square and normalized residual tests will be evaluated. State estimation will be done as usual, making use of the left-inverse, resulting in:

$$\hat{x} = [H_x^* H_x]^{-1} H_x^* z = \begin{bmatrix} 0.9567 + 0.0409i \\ 0.9567 + 0.0407i \\ 0.9800 + 0.0201i \\ 1.0221 - 0.0198i \\ 0.9997 - 0.0001i \end{bmatrix}$$

The estimated error vector must be computed to validate the estimation. Inspection reveals that several of these estimated errors have fairly substantial values.

$$\hat{e} = z - \hat{z} = z - H_x \hat{x} = \begin{bmatrix} -0.0019 + 0.0000i \\ 0.0014 - 0.0000i \\ -0.0008 - 0.0001i \\ 0.0003 + 0.0001i \\ -0.5650 - 0.7787i \\ -0.3781 - 0.5813i \\ -0.0081 + 0.0083i \\ 0.9526 + 1.3517i \end{bmatrix}$$

Then the estimated quadratic error norm, $\hat{f}$ is computed. This value is to be compared with the chi-squared inverse cumulative probability function, with three degrees of freedom and a confidence interval of 99%. Note that the difference between the number of measurements and the number of state variables yields the number of degrees of freedom.

$$\hat{f} = \hat{e}^T W_z \hat{e} = 16.3851 > \chi^2_{0,01;3} = 11.3449$$

Since the estimated quadratic error norm $\hat{f}$ is bigger, the estimation is invalid due to an erroneous measurement. To identify this measurement, the residuals have to be computed.

$$res_i = \frac{\hat{e_i}}{R'_{ii}} = \begin{bmatrix} -3.4064 + 0.0805i \\ 2.7093 - 0.1926i \\ 0.0901 - 0.0203i \\ 0.0210 - 0.0119i \\ -0.0000 - 0.0000i \\ -0.0000 - 0.0000i \\ -0.0000 + 0.0000i \\ 0.0000 + 0.0000i \end{bmatrix}, \quad R' = W_z^{-1} - H_x(H_x^T W_z H_x)^{-1} H_x^T$$

The residual in relation to the incorrect measurement is the largest in absolute magnitude, showing its invalidity. The state is then re-estimated without taking this measurement into account. Note that new weight and network model matrices have to be computed, W_z^{new} and H_x^{new}. By inspection it can be verified that that estimated errors have decreased.

$$\hat{e}_{new} = z - \hat{z}_{new} = z - H_x^{new} \hat{x}_{new} = \begin{bmatrix} 0.0014 - 0.0000i \\ -0.0009 - 0.0000i \\ 0.0004 + 0.0000i \\ -0.0480 - 0.0620i \\ -0.4257 - 0.5797i \\ 0.1383 + 0.1969i \\ 0.3368 + 0.4447i \end{bmatrix}$$

The new estimated quadratic error norm, $\hat{f}_{new}$ is computed. The estimation is now valid since the estimated quadratic error norm is smaller. If this were not the case, the residual analyses would have to be repeated in order to eliminate one more measurement. Note that the number of degrees of freedom in the chi-squared inverse cumulative probability function decreased to two due to the removed measurement.

$$\hat{f}_{new} = \hat{e}_{new}^T W_z^{new} \hat{e}_{new} = 6.6999 > \chi^2_{0,01;2} = 9.2103$$

4.8 SOP

New electronic devices such as soft open points (SOPs) are posing new challenges to distribution system operators, imposing serious difficulties on the state estimation (SE) process of weakly monitored grids[2]. The SOPs functional model based on current conservation will be utilized to improve estimation in this section.

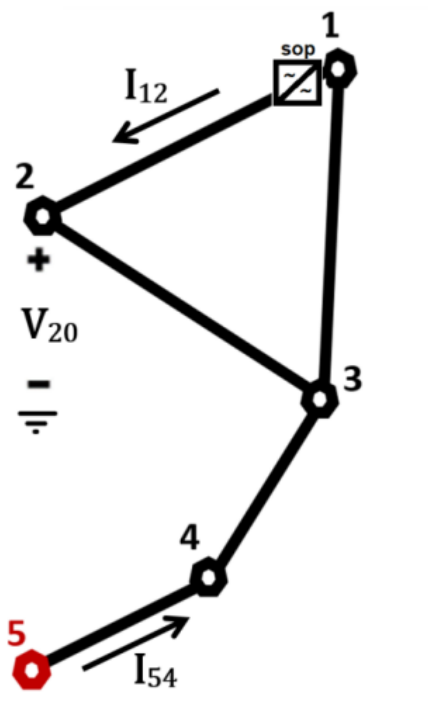


Figure 4: "Kite" grid with SOP

Consider the "kite" grid with a SOP between buses 1 and 2, the first of which may be broken down into 1j and 1i. This decomposition will increase the state vector size. Considering the current conservation approximation on SOP, an equation can be obtained, relating the observed injected current on the $1j$ bus with the state, x, increasing $Rank(H_x)$ to 6, making the system observable:

$$\begin{aligned} I_{1j}^{inj} &= \textstyle\sum_{n=1}^{6} Y(1,n)x(n) \\ I_{1i}^{inj} &= \textstyle\sum_{n=1}^{6} Y(2,n)x(n) \\ I_{1j}^{original} &= I_{1j}^{inj} + I_{1i}^{inj} = [Y(1,:) + Y(2,:)]x(n) \end{aligned}$$

This equation can be used to expand H_x, taking into account available grid admittance data. The current state can be estimated through $\hat{x} = [H_x^* H_x]^{-1} H_x^* z$. Even though observabillity was achieved, there was a slight disparity between the true and estimated states:

$$x = \begin{bmatrix} 0.9843 - 0.1110i \\ 0.8967 - 0.3021i \\ 0.9500 - 0.2000i \\ 0.9622 - 0.1518i \\ 0.9662 - 0.1119i \\ 1.0000 + 0.0000i \end{bmatrix} \quad \hat{x} = \begin{bmatrix} 0.8558 - 0.0818i \\ 0.7628 - 0.3218i \\ 0.8180 - 0.2000i \\ 0.8318 - 0.1414i \\ 1.0048 - 0.1208i \\ 1.0395 + 0.0000i \end{bmatrix}$$

4.9 Modelling pseudo-measurements as equality contraints

Lets consider that load pseudo-measurements are available. In this section these measurements alongside the SOP functional model will utilized to improve estimation.

Virtual measurements can be separated from other measurements and modelled as explicit constraints. Modelling the virtual information on stationarity, can be done expressing control setpoints as $u(s) \approx Ex$ with $E = (B^*B)^{-1}B^*(IA)$. Considering SOP current conservation and bus 1 load pseudo-measurement, we can add one more constraint to improve observability:

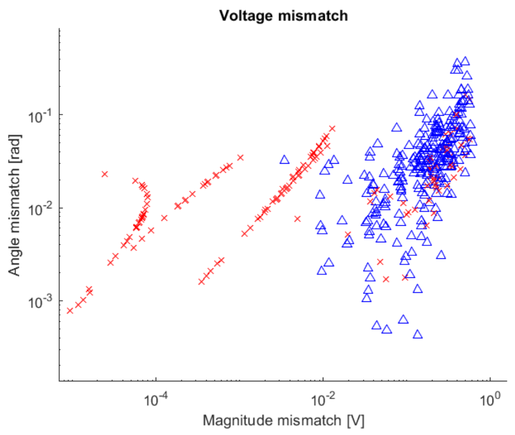
$$I_1^{orig} = I_{1i}(s) + I_{1j}(s) = I_{1i}(k) + I_{1j}(k) \forall k < s.$$

To add this constraint to the model, it has be expressed on the voltages, ie $I_1^{orig} = (Y_1 + Y_2)x$. In matrix form, equality constraints can be expressed as: $Ex = u(s)$, with:

$$E = \begin{bmatrix} (B_2^* B_2)^{-1} B_2^* B_1 F \\ \sum Y_{1,i} + Y_{2,i} \end{bmatrix}$$

Considering that the measurement errors are normally distributed, they can be modelled as $z_i = z_i^{original} + \mathcal{N}(\mu_i, \sigma_i^2)$, considering null values for the means, μ_i, as to respect the Gauss-Markov Theorem. The influence of these inaccuracies on estimations may be observed in the displayed current states as well as the graph that aggregated the errors of several states.

$$x = \begin{bmatrix} 0.9843 - 0.1110i \\ 0.8967 - 0.3021i \\ 0.9500 - 0.2000i \\ 0.9622 - 0.1518i \\ 0.9662 - 0.1119i \\ 1.0000 + 0.0000i \end{bmatrix} \quad \hat{x}^{OLS} = \begin{bmatrix} 0.8558 - 0.0818i \\ 0.7628 - 0.3218i \\ 0.8180 - 0.2000i \\ 0.8318 - 0.1414i \\ 1.0048 - 0.1208i \\ 1.0395 + 0.0000i \end{bmatrix} \quad \hat{x}^{OLS}_{Eq.Cons} = \begin{bmatrix} 0.9864 - 0.0901i \\ 0.8926 - 0.3438i \\ 0.9500 - 0.2000i \\ 0.9622 - 0.1518i \\ 0.9865 - 0.1170i \\ 1.0208 + 0.0000i \end{bmatrix}$$



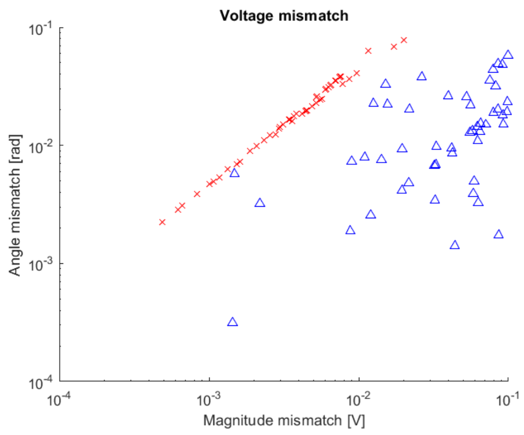


Figure 5: Voltage mismatch comparison

When comparing the state mismatch, it becomes clear that equality-constrained formulation was much more accurate. In red we have the mismatches between the exact and Eq. Constrained OLS which not only had smaller values (in general) but also followed trendlines, therefore being more predictable. In blue, we have the mismatches between the exact and OLS which not only had bigger values (in general) but also were more scattered.

In order to improve our estimation even further, advantage was taken of the fact that different measurements may be associated with different errors. This is often the case as current RMS magnitude meters aren't as accurate as PMUs which use GPS synchronization to estimate voltage magnitude and phase angle. By introducing a weight matrix, we can give more credit to the accurate measurements. Considering that the measurement errors are uncorrelated, the weight matrix, W_z, is a diagonal matrix whose inputs are the inverse of the variance, ie $W_z = diag(\frac{1}{\sigma_i^2})$. This optimization problem can be solved by the method of Lagrange multipliers. The lagrangian can be expressed as follows:

$$\mathcal{L}(x,\lambda) = \tfrac{1}{2}(z - H_x)^* W_z (z - H_x) - \lambda^T (Ex - u(s))$$

The first-order optimal conditions can be written as:

$$\frac{\partial\mathcal{L}(x,\lambda)}{\partial x} = H_x^* W_z (z\text{-}H_x x) - E^*\lambda E = 0$$
$$\frac{\partial\mathcal{L}(x,\lambda)}{\partial \lambda} = Ex + u(s) = 0$$

As a consequence of the linearity, the solution of the first-order optimal conditions (which guarantee the optimum) can be obtained by solving the following linear system:

$$\begin{bmatrix} H_x^* W_z H_x & -E^* \\ E & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} H_x^* W_z z \\ u(s) \end{bmatrix}$$

Considering $\sigma^2 = 0.25$ for the voltage measurements and $\sigma^2 = 0.5$ for the current measurements the following mismatches were obtained:

$$x = \begin{bmatrix} 0.9843 - 0.1110i \\ 0.8967 - 0.3021i \\ 0.9500 - 0.2000i \\ 0.9622 - 0.1518i \\ 0.9662 - 0.1119i \\ 1.0000 + 0.0000i \end{bmatrix} \quad \hat{x}^{OLS}_{Eq.C} = \begin{bmatrix} 0.9864 - 0.0901i \\ 0.8926 - 0.3438i \\ 0.9500 - 0.2000i \\ 0.9622 - 0.1518i \\ 0.9865 - 0.1170i \\ 1.0208 + 0.0000i \end{bmatrix} \quad \hat{x}^{WLS}_{Eq.C} = \begin{bmatrix} 0.9834 - 0.1216i \\ 0.8989 - 0.2808i \\ 0.9500 - 0.2000i \\ 0.9623 - 0.1519i \\ 0.9675 - 0.0997i \\ 1.0000 + 0.0000i \end{bmatrix}$$

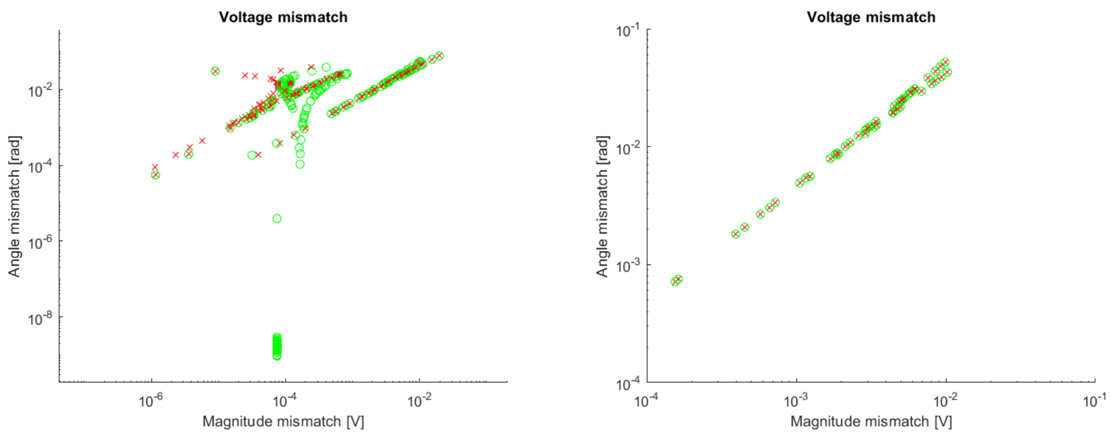


Figure 6: Voltage mismatch comparison

Equality constrained OLS had minor mismatches as seen in the trend lines in red. The equality limited WLS (green) exhibited a marginally smaller magnitude and angle mismatch and followed similar trend lines. As a result, the equality restricted WLS state estimation will perform better for the proposed problem and for minor measurement error discrepancies. Note that when large error differences are present, large weighting factors are used leading to poor results. In the next table, the average mismatch for 500 estimated states reflect the improvements discussed above:

OLS	Eq.Cons OLS	Eq. Cons WLS
0.18136	0.00037	0.00035

Table 8: Average voltage magnitude mismatch

OLS	Eq.Cons OLS	Eq. Cons WLS
0.04525	0.00120	0.00072

Table 9: Average voltage angle mismatch

5 Conclusion

From the experiments done, the main findings to be retained are the following:

- Some measurements provide more information than others because they are closely related to the state variable to be estimated.
- Estimation is easier with more measurements, as long as the added measurements aren't noisy.
- Topology processing in linear SE can't be done with the second expression for inverse gain matrix.
- Meshed grids have more power flow dynamics, making SE harder.
- Residual analysis based on chi-square and normalized residual tests is a valid approach to identifying bad data.
- The formulation of SOP device operating principles on their port injections, while disregarding voltage magnitudes from the DC side and voltage magnitude and angle from the AC side, can assist in retaining observability.
- SOP control setpoints (Pseudo-measurements) can be modeled as equality constraints, constituting an improvement on SE accuracy, even if setpoints are not exactly met.

References

1)**Lecture Notes: Chap 2.3 State Estimation**

2)**State estimation in weakly monitored active distribution networks: Modeling soft open points functional behavior-Diogo M.V.P. Ferreira, Pedro M.S. Carvalho**

2)**Smart Grid Fundamentals Lecture Notes**

Appendix

A Incomplete State Estimation, Complete State Estimation and Chose just two out of four possible pseudo-measurements Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

networkFactor = 100  # To change the characteristics of the network (Y)
cosPhi = 0.95  # Value of teta
m = 100  # Number of Iterations
sig = 0.5  # Noise factor

# Import data (From Excel file)

Info = np.array(pd.read_excel(r'\Lab3\DASG_Prob2_new.xlsx', sheet_name='Info', header=None))
# Information about the slack bus
SlackBus = Info[0, 1]
print("Slack Bus: ", SlackBus, "\n")

# Network Information
Net_Info = np.array(pd.read_excel(r'\Lab3\DASG_Prob2_new.xlsx', sheet_name='Y_Data'))
print("Lines information (Admitances)\n", Net_Info, "\n")

# Power Information (train)
Power_Info = np.array(pd.read_excel(r'\Lab3\DASG_Prob2_new.xlsx', sheet_name='Load(t,Bus)'))
Power_Info = np.delete(Power_Info, [0], 1)
print("Power consumption information (time, Bus) - (Train)\n", Power_Info, "\n")

time = Power_Info.shape[0]
P = np.dot(-Power_Info, np.exp(complex(0, 1) * np.arccos(cosPhi)))
I = np.conj(P[2, :])

# Determine the number of Bus
nBus = max(np.max(Net_Info[:, 0]), np.max(Net_Info[:, 1]))

# Create the variable number of lines and the admitance matrix (Y)
nLines = Net_Info.shape[0]

Y = np.zeros((nBus, nBus), dtype=complex)

# Complete the Y matrix nad update the number of lines
for i in range(Net_Info.shape[0]):
    y_aux = Net_Info[i, 2].replace(",", ".")
    y_aux = y_aux.replace("i", "j")
    Y[Net_Info[i, 0] - 1, Net_Info[i, 0] - 1] = Y[Net_Info[i, 0] - 1, Net_Info[i, 0] - 1]
    + complex(y_aux) * networkFactor
    Y[Net_Info[i, 1] - 1, Net_Info[i, 1] - 1] = Y[Net_Info[i, 1] - 1, Net_Info[i, 1] - 1]
    + complex(y_aux) * networkFactor
    Y[Net_Info[i, 0] - 1, Net_Info[i, 1] - 1] = Y[Net_Info[i, 0] - 1, Net_Info[i, 1] - 1]
    - complex(y_aux) * networkFactor
    Y[Net_Info[i, 1] - 1, Net_Info[i, 0] - 1] = Y[Net_Info[i, 1] - 1, Net_Info[i, 0] - 1]
    - complex(y_aux) * networkFactor

# Remove the slack bus from the admitance matrix
Yl = np.delete(Y, np.s_[SlackBus - 1], axis=0)
Yl = np.delete(Yl, np.s_[SlackBus - 1], axis=1)

# Conductance Matrix
G = Yl.real

# Susceptance Matrix
B = Yl.imag

print("The admitance matrix Y is:\n", Y, "\n")
print("The conductance matrix G is\n", G, "\n")
print("The susceptance matrix B is\n", B, "\n")

# State Estimation considering complete information about currents (amplitude and angle)
and not estimating V3 (Page 39)

# Matrix Creation
b0s = np.zeros(4, dtype=complex)
A0s = np.zeros((4, 4), dtype=complex)
v0s = np.zeros(5, dtype=complex)

# Voltage Computing (Reference)
v0s[0:4] = 1 + np.dot(np.linalg.inv(Yl), I)
v0s[4] = 1

# Measurement values z
b0s[0] = Y[0, 1] * (v0s[0] - v0s[1])
b0s[1] = Y[3, 4] * (v0s[4] - v0s[3])
b0s[2] = v0s[1]
b0s[3] = v0s[4]
b0s = np.transpose(b0s)

# Matrix Hx
A0s[0, 0] = Y[0, 1]
A0s[0, 1] = -A0s[0, 0]
A0s[1, 2] = -Y[3, 4]
A0s[1, 3] = Y[3, 4]
A0s[2, 1] = 1
A0s[3, 3] = 1
print("H")
print(A0s)
# State Variables (x) - These variables are the estimated voltages (V1; V2; V4; V5)

x = np.dot(np.linalg.inv(np.dot(np.transpose(A0s), A0s)), np.dot(np.transpose(A0s), b0s))
x = x.reshape(4, 1)

# State Estimation considering RMS information of currents I12 and I54 and estimating V

# Matrix Creation
b0 = np.zeros(8, dtype=complex)
b1 = np.zeros(8, dtype=complex)
b2 = np.zeros(8, dtype=complex)
A = np.zeros((8, 5), dtype=complex)
v = np.zeros(5, dtype=complex)

# Voltage Computing (Reference)
v[0:4] = 1 + np.dot(np.linalg.inv(Yl), I)
v[4] = 1

# Measurement values z (In that case, we are computing the currents and voltages but we
can also measure them).
b0[0] = np.dot(np.absolute(np.dot(-Y[0, 1], (v[0] - v[1]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b0[1] = np.dot(np.absolute(np.dot(-Y[3, 4], (1 - v[3]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b0[2] = v[1]
b0[3] = 1
b0[4:8] = I  # In that case, we are also including the vectors of aparente power

# Matrix Hx (Page 41)
A[0, 0] = -Y[0, 1]
A[0, 1] = Y[0, 1]
A[1, 3] = Y[3, 4]
A[1, 4] = -Y[3, 4]
A[2, 1] = 1
A[3, 4] = 1
A[4, 0] = -Y[0, 1] - Y[0, 2]
A[4, 1] = Y[0, 1]
A[4, 2] = Y[0, 2]
A[5, 0] = Y[0, 1]
A[5, 1] = -Y[0, 1] - Y[1, 2]
A[5, 2] = Y[1, 2]
A[6, 0] = Y[0, 2]
A[6, 1] = Y[1, 2]
A[6, 2] = -Y[0, 2] - Y[1, 2] - Y[2, 3]
A[6, 3] = Y[2, 3]
A[7, 2] = Y[2, 3]
A[7, 3] = -Y[2, 3] - Y[3, 4]
A[7, 4] = Y[3, 4]
```

```
# 2º a) - First, the weight of pseudo-measurements (Matrix W) are not considered

# Noise to be added to peseudo-measurements
e = np.random.normal(0.0, 1.0, size=(4, m)) * sig

# Estimation not considering the noise

x = np.dot(np.linalg.inv(np.dot(np.transpose(A), A)), np.dot(np.transpose(A), b0))

# Matrix creation
sx = np.zeros(5)
rms = np.zeros((5, m))
ei12a = np.zeros(m)
ei54a = np.zeros(m)
xerro = np.zeros((5, m), dtype=complex)

b1[0] = np.dot(np.absolute(np.dot(-Y[0, 1], (v[0] - v[1]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b1[1] = np.dot(np.absolute(np.dot(-Y[3, 4], (1 - v[3]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b1[2] = v[1]
b1[3] = 1

for i in range(m):
    # Introduce error in the measurements (Matrix z)
    b1[4:8] = I + e[:, i]

    # Estimate the voltages based on measurements with the errors
    xerro[:, i] = np.dot(np.linalg.inv(np.dot(np.transpose(A), A)), np.dot(np.transpose(A), b1
    # xerro[:, i] = xerro.reshape(5, 1)
    # print('\n Com erro \n',xerro)

    # Accumulated value of estimations
    sx = xerro[:, i] + sx

    # Errors in the voltages
    rms[:, i] = np.sqrt(np.dot((xerro[:, i] - x), np.conjugate(xerro[:, i] - x)))
    # x0 is the initial estimation not considering the noise
    # x is the estimation considering the noise

    # Relative current errors (To be used in the graphics)
    ei12a[i] = np.divide(
        np.absolute(np.dot(Y[0, 1], (xerro[0, i] - xerro[1, i])))
        - np.absolute(np.dot(Y[0, 1], (v[0] - v[1]))),
        np.absolute(np.dot(Y[0, 1], (v[0] - v[1]))))
    ei54a[i] = np.divide(np.absolute(np.dot(Y[3, 4], (1 - xerro[3, i])))
    - np.absolute(np.dot(Y[3, 4], (1 - v[3]))),
        np.absolute(np.dot(Y[3, 4], (1 - v[3]))))
```

```
# Average Voltage Estimation
x_avg = sx / m

# Average Voltage RMS Error
ee = np.transpose(np.matrix, sum(np.transpose(rms))) / m
print('\n Average voltage RMS error \n', ee)

print('\n Errors in the voltages \n', rms)

# 2º b) - Second, the weight of pseudo-measurements (Matrix W) are considered

# The vector of errors is the same used in the first scenario

# Matriz of the Weights (W)
W = np.zeros((8, 8))
xerro2 = np.zeros((5, m), dtype=complex)
sx2 = np.zeros(5)
ei12a2 = np.zeros(m)
ei54a2 = np.zeros(m)
rms2 = np.zeros((5, m))
np.fill_diagonal(W[0:4], 0.001 ** -2)
np.fill_diagonal(W[4:8, 4:8], sig ** -2)

# Estimation not considering the noise, but considering the weight
x2b = np.dot(np.linalg.inv(np.dot(np.dot(np.transpose(A), W), A)),
np.dot(np.dot(np.transpose(A), W), b0))

b2[0] = np.dot(np.absolute(np.dot(-Y[0, 1], (v[0] - v[1]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b2[1] = np.dot(np.absolute(np.dot(-Y[3, 4], (1 - v[3]))), np.exp(complex(0, -1)
* np.arccos(cosPhi)))
b2[2] = v[1]
b2[3] = 1

for i in range(m):
    # Introduce error in the measurements (Matrix z)
    b2[4:8] = I + e[:, i]
    # Estimate the voltages based on measurements with the errors
    xerro2[:, i] = np.dot(np.linalg.inv(np.dot(np.dot(np.transpose(A), W), A)), np.dot(np.dot(

    # Accumulated value of estimations
    sx2 = xerro2[:, i] + sx2
    # Errors in the voltages
    rms2[:, i] = np.sqrt(np.dot((xerro2[:, i] - x2b), np.conjugate(xerro2[:, i] - x2b)))

    # Relative current errors (To be used in the graphics)
    ei12a2[i] = np.divide(
        np.absolute(np.dot(Y[0, 1], (xerro2[0, i] - xerro2[1, i])))
        - np.absolute(np.dot(Y[0, 1], (v[0] - v[1]))),
        np.absolute(np.dot(Y[0, 1], (v[0] - v[1]))))
    ei54a2[i] = np.divide(np.absolute(np.dot(Y[3, 4], (1 - xerro2[3, i])))
    - np.absolute(np.dot(Y[3, 4], (1 - v[3]))),
    np.absolute(np.dot(Y[3, 4], (1 - v[3]))))

# Average Voltage Estimation
x_avg2 = sx2 / m
print('\n X_avg \n', x_avg2)

# Average Voltage RMS Error
ee2 = np.transpose(np.matrix, sum(np.transpose(rms2))) / m
print('\n Average voltage RMS error \n', ee2)

print('\n Errors in the voltages \n', rms2)

# Plot Results
fig = plt.figure()
plt.title("Scattered Mismatches")

# Labeled the axis
plt.xlabel("Relative mismatch I12")
plt.ylabel("Relative mismatch I54")

# Used this to plot the scattered dots
plt.scatter(abs(ei12a), abs(ei54a), c='b', marker='.', linewidths=1, label='W/o Wz',
alpha=0.5)
plt.scatter(abs(ei12a2), abs(ei54a2), c='r', marker='.', linewidths=1, label='With Wz',
alpha=0.5)
plt.legend()

# Used Logarithimic scale
plt.yscale('log')
plt.xscale('log')

# We set the size of the plot
fig.set_figwidth(8)
fig.set_figheight(8)
plt.show()
```

B Adding Measurement Matlab Code

```
  %% Grid Matrices
%Kite
Y1 =     [ 3-30*j -1+10*j -2+20*j  0  0
         -1+10*j  4-30*j -3+20*j  0  0
         -2+20*j -3+20*j  8-60*j -3+20*j  0
          0  0 -3+20*j  5-40*j -2+20*j
          0  0  0 -2+20*j  2-20*j];

Z1 = Y1(1:4,1:4)^-1;

%% Mesurement Jacobian
%Admitances
y12 = 1-1i*10;
y13 = 2*y12;
y45 = 2*y12;
y23 = 3-1i*20;
y34 = y23;
y24 = 1-1i*10;

%Original
Hx_1 =  [y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];

%One added
Hx_2 =  [0 0 -y34 y34 0
        y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];

%Two added
Hx_3 =  [0 -y23 y23 0 0
         0 0 -y34 y34 0
        y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];
```

```
%% Determine acuracy
erro = zeros(100,3);

for kk = 1:100

%Load data (injected currents)
Load = [0.3320 0.0640 0.0840 0.1200].';

for k = 1:4
    Load(k) = -Load(k)*exp(1i*.95);
end

%Linear PF - Kite
state_exact1 = 1-Z1*Load;
state_exact1(5) = 1;


%Exact mesurements - Kite
z_exact1 = [y12*(state_exact1(1) - state_exact1(2))
           y45*(state_exact1(5) - state_exact1(4))
           state_exact1(2)
           state_exact1(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

%Exact mesurements - 1 added
z_exact2 = [y34*(state_exact1(4) - state_exact1(3))
            y12*(state_exact1(1) - state_exact1(2))
           y45*(state_exact1(5) - state_exact1(4))
           state_exact1(2)
           state_exact1(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

%Exact mesurements - 2 added
z_exact3 = [y23*(state_exact1(3) - state_exact1(2))
            y34*(state_exact1(4) - state_exact1(3))
            y12*(state_exact1(1) - state_exact1(2))
            y45*(state_exact1(5) - state_exact1(4))
            state_exact1(2)
            state_exact1(5)
            -Load(1)
            -Load(2)
            -Load(3)
            -Load(4)];

%Introduce error
for k = 1:8
    z1(k,1) = z_exact1(k) + normrnd(0,0.02);
    z2(k+1,1) = z_exact2(k+1) + normrnd(0,0.02);
    z3(k+2,1) = z_exact3(k+2) + normrnd(0,0.02);
end

%Introduce error to added measurments
z2(1,1) = z_exact2(1) + normrnd(0,2);
z3(1,1) = z_exact3(1) + normrnd(0,2);
z3(2,1) = z_exact3(2) + normrnd(0,2);


%% State Estimation
%Weight Matrix
Wz = diag([10^6 10^6 10^6 10^6 4 4 4 4]);
Wz_2 = diag([10^6 10^6 10^6 10^6 10^6 4 4 4 4]);
Wz_3 = diag([10^6 10^6 10^6 10^6 10^6 10^6 4 4 4 4]);

%Original
x_est1 = (Hx_1'* Wz * Hx_1)^-1*Hx_1'* Wz *z1;


%Added 1
x_est2 = (Hx_2'* Wz_2 * Hx_2)^-1*Hx_2'* Wz_2 *z2;


%Added 2
x_est3 = (Hx_3'* Wz_3 * Hx_3)^-1*Hx_3'* Wz_3 *z3;

%% Compute Rx matrices
%Original
Rx1 = (state_exact1-x_est1)*(state_exact1-x_est1)';

%Added 1
Rx2 = (state_exact1-x_est2)*(state_exact1-x_est2)';

%Added 2
Rx3 = (state_exact1-x_est3)*(state_exact1-x_est3)';



for k = 1:5
    erro(kk,1) = erro(kk,1) + abs(Rx1(k,k));
    erro(kk,2) = erro(kk,2) + abs(Rx2(k,k));
    erro(kk,3) = erro(kk,3) + abs(Rx3(k,k));
end
```

```
end


orig = 0;
onemes = 0;
twomes = 0;

for kk = 1:100
    orig = orig + erro(kk,1);
    onemes = onemes + erro(kk,2);
    twomes = twomes + erro(kk,3);
end

orig = orig/100;
onemes = onemes/100;
twomes = twomes/100;

% Plot
% figure()
% ep=1:100;
% plot(ep,erro)
% xlabel("Iteration")
% ylabel("Error magnitude")
% axis([1 100 0 0.4])
% legend({'Kite','Radial','Meshed'},'Location','northwest')
```

C Topology Processing and Bad Data Matlab Code

```
  %% Grid Matrices
%Original
Y =     [ 3-30*j -1+10*j -2+20*j  0  0
         -1+10*j  4-30*j -3+20*j  0  0
         -2+20*j -3+20*j  8-60*j -3+20*j  0
          0  0 -3+20*j  5-40*j -2+20*j
          0  0  0 -2+20*j  2-20*j];

%line 1-3 offline
Y2 =    [ 1-10*j -1+10*j 0  0  0
         -1+10*j  4-30*j -3+20*j  0  0
          0 -3+20*j  6-40*j -3+20*j  0
          0  0 -3+20*j  5-40*j -2+20*j
          0  0  0 -2+20*j  2-20*j];

Z1 = Y(1:4,1:4)^-1;
Z2 = Y2(1:4,1:4)^-1;

%% Mesurement Jacobian
```

```
%Admitances
y12 = 1-1i*10;
y13 = 2*y12;
y45 = 2*y12;
y23 = 3-1i*20;
y34 = y23;

%Original
Hx =  [y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];

%line 1-2 offline
y12 = 0;

Hx_12 =  [y12 -y12 0 0 0
           0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

y12 = 1-1i*10;

%line 1-3 offline
y13 = 0;

Hx_13 =  [y12 -y12 0 0 0
           0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

y13 = 2*y12;

%line 4-5 offline
y45 = 0;

Hx_45 =  [y12 -y12 0 0 0
```

```
           0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

y45 = 2*y12;

%line 2-3 offline
y23 = 0;

Hx_23 =  [y12 -y12 0 0 0
           0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

y23 = 3-1i*20;

%line 3-4 offline
y23 = 0;

Hx_34 =  [y12 -y12 0 0 0
           0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

y34 = 3-1i*20;


%% Generate Data
%Load data (injected currents)
Load = [0.3320 0.0640 0.0840 0.1200].';

for k = 1:4
    Load(k) = -Load(k)*exp(1i*.95);
end

%Linear PF
state_exact = 1-Z2*Load;
```

```
state_exact(5) = 1;

%Faulty line
y13 = 0;

z_exact = [y12*(state_exact(1) - state_exact(2))
           y45*(state_exact(5) - state_exact(4))
           state_exact(2)
           state_exact(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

y13 = 2*y12;

%Introduce error
for k = 1:8
    z(k,1) = z_exact(k) + normrnd(0,0.2);
end

%% State Estimation
%Weight Matrix
Wz = diag([10^6 10^6 10^6 10^6 4 4 4 4]);

%Original
x_est = (Hx'* Wz * Hx)^-1*Hx'* Wz *z;

%line 1-2 offline
x_est_12 = (Hx_12'* Wz *Hx_12)^-1*Hx_12'* Wz *z;

%line 1-3 offline
x_est_13 = (Hx_13'* Wz *Hx_13)^-1*Hx_13'* Wz *z;

%line 4-5 offline
x_est_45 = (Hx_45'* Wz *Hx_45)^-1*Hx_45'* Wz *z;

%line 2-3 offline
x_est_23 = (Hx_23'* Wz *Hx_23)^-1*Hx_23'* Wz *z;

%line 3-4 offline
x_est_34 = (Hx_34'* Wz *Hx_34)^-1*Hx_34'* Wz *z;


%% Compute Rx matrices
Rx = (state_exact-x_est)*(state_exact-x_est)';
Rx_12 = (state_exact-x_est_12)*(state_exact-x_est_12)';
Rx_13 = (state_exact-x_est_13)*(state_exact-x_est_13)';
Rx_45 = (state_exact-x_est_45)*(state_exact-x_est_45)';
```

```
Rx_23 = (state_exact-x_est_23)*(state_exact-x_est_23)';
Rx_34 = (state_exact-x_est_34)*(state_exact-x_est_34)';

% Rx = (Hx'*Wz*Hx)^-1;
% Rx_12 = (Hx_12'*Wz*Hx_12)^-1;
% Rx_13 = (Hx_13'*Wz*Hx_13)^-1;
% Rx_45 = (Hx_45'*Wz*Hx_45)^-1;
% Rx_23 = (Hx_23'*Wz*Hx_23)^-1;
% Rx_34 = (Hx_34'*Wz*Hx_34)^-1;

erro = 0;
erro_12 = 0;
erro_13 = 0;
erro_45 = 0;
erro_23 = 0;
erro_34 = 0;

for k = 1:5
    erro = erro + abs(Rx(k,k));
    erro_12 = erro_12 + (Rx_12(k,k));
    erro_13 = erro_13 + (Rx_13(k,k));
    erro_45 = erro_45 + (Rx_45(k,k));
    erro_23 = erro_23 + (Rx_23(k,k));
    erro_34 = erro_34 + (Rx_34(k,k));
end



%% Faulty meter

%Load data (injected currents)
Load = [0.3320 0.0640 0.0840 0.1200].';

for k = 1:4
    Load(k) = -Load(k)*exp(1i*.95);
end

%Linear PF
state_exact = 1-Z1*Load;
state_exact(5) = 1;


z_exact = [y12*(state_exact(1) - state_exact(2))
           y45*(state_exact(5) - state_exact(4))
           state_exact(2)
           state_exact(5)
           Load(1)
           Load(2)
           Load(3)
           Load(4)];

%Error I = 0
z_exact(1) = 0;

%Introduce error
for k = 1:8
    z(k,1) = z_exact(k) + normrnd(0,0.001);
end

%Estimate State
x_est = (Hx.'* Wz * Hx)^-1*Hx.'* Wz *z;

%Estimated error vector
err_est = z_exact - Hx*x_est;

%Estimated quadratic error norm
f = err_est' * Wz * err_est;

%Residuals
R_linha = Wz^-1 - Hx*(Hx'*Wz*Hx)^-1*Hx';

for k = 1:8
    residuals(k,1) = abs(err_est(k)/R_linha(k,k));
end

%Remove measuremnt
for k = 1:7
z_new(k,1) = z(k+1);
z_exact_new(k,1) = z_exact(k+1);
end

%New weight Matrix
Wz_new = diag([10^6 10^6 10^6 4 4 4 4]);

%Exclude measuremnt
Hx_new =  [0 0 0 y45 -y45
           0 1 0 0 0
           0 0 0 0 1
           y12+y13 -y12 -y13 0 0
           -y12 y12+y23 -y23 0 0
           -y13 -y23 y13+y23+y34 -y34 0
           0 0 -y34 y34+y45 -y45];

%Re-estimation
x_est_new = (Hx_new.'* Wz_new * Hx_new)^-1*Hx_new.'* Wz_new *z_new;

%Estimated error vector
err_est_new = z_exact_new - Hx_new*x_est_new;
```

```
%Estimated quadratic error norm
f_new = err_est_new' * Wz_new * err_est_new;
```

D Estimation on meshed grids Matlab Code

```
 %% Grid Matrices
%Kite
Y1 =      [ 3-30*j -1+10*j -2+20*j  0  0
         -1+10*j  4-30*j -3+20*j  0  0
         -2+20*j -3+20*j  8-60*j -3+20*j  0
          0  0 -3+20*j  5-40*j -2+20*j
          0  0  0 -2+20*j  2-20*j];

%Radial
Y2 =     [ 1-10*j -1+10*j 0  0  0
         -1+10*j  4-30*j -3+20*j  0  0
          0 -3+20*j  6-40*j -3+20*j  0
          0  0 -3+20*j  5-40*j -2+20*j
          0  0  0 -2+20*j  2-20*j];

%Meshed
y24 = 1-1i*10;

Y3 =      [ 3-30*j -1+10*j -2+20*j  0  0
         -1+10*j  4-30*j+y24 -3+20*j  -y24  0
         -2+20*j -3+20*j  8-60*j -3+20*j  0
          0  -y24 -3+20*j  5-40*j+y24 -2+20*j
          0  0  0 -2+20*j  2-20*j];

Z1 = Y1(1:4,1:4)^-1;
Z2 = Y2(1:4,1:4)^-1;
Z3 = Y3(1:4,1:4)^-1;

%% Mesurement Jacobian
%Admitances
y12 = 1-1i*10;
y13 = 2*y12;
y45 = 2*y12;
y23 = 3-1i*20;
y34 = y23;
y24 = 1-1i*10;

%Kite
Hx_1 =  [y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];

%Radial
y13 = 0;

Hx_2 =  [y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23 -y23 0 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 0 -y34 y34+y45 -y45];

y13 = 2*y12;

%Meshed
Hx_3 =  [y12 -y12 0 0 0
       0 0 0 y45 -y45
       0 1 0 0 0
       0 0 0 0 1
       y12+y13 -y12 -y13 0 0
       -y12 y12+y23+y24 -y23 -y24 0
       -y13 -y23 y13+y23+y34 -y34 0
       0 -y24 -y34 y34+y45+y24 -y45];

%% Determine acuracy
erro = zeros(100,3);

for kk = 1:100

%Load data (injected currents)
Load = [0.3320 0.0640 0.0840 0.1200].';

for k = 1:4
    Load(k) = -Load(k)*exp(1i*.95);
end

%Linear PF - Kite
state_exact1 = 1-Z1*Load;
state_exact1(5) = 1;

%Linear PF - Radial
state_exact2 = 1-Z2*Load;
state_exact2(5) = 1;
```

```
%Linear PF - Meshed
state_exact3 = 1-Z3*Load;
state_exact3(5) = 1;

%Exact mesurements - Kite
z_exact1 = [y12*(state_exact1(1) - state_exact1(2))
           y45*(state_exact1(5) - state_exact1(4))
           state_exact1(2)
           state_exact1(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

%Exact mesurements - Radial
z_exact2 = [y12*(state_exact2(1) - state_exact2(2))
           y45*(state_exact2(5) - state_exact2(4))
           state_exact2(2)
           state_exact2(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

%Exact mesurements - Meshed
z_exact3 = [y12*(state_exact3(1) - state_exact3(2))
           y45*(state_exact3(5) - state_exact3(4))
           state_exact3(2)
           state_exact3(5)
           -Load(1)
           -Load(2)
           -Load(3)
           -Load(4)];

%Introduce error
for k = 1:8
    z1(k,1) = z_exact1(k) + normrnd(0,0.002);
    z2(k,1) = z_exact2(k) + normrnd(0,0.002);
    z3(k,1) = z_exact3(k) + normrnd(0,0.002);
end


%% State Estimation
%Weight Matrix
Wz = diag([10^6 10^6 10^6 10^6 4 4 4 4]);

%Kite
x_est1 = (Hx_1'* Wz * Hx_1)^-1*Hx_1'* Wz *z1;
```

```
%Radial
x_est2 = (Hx_2'* Wz * Hx_2)^-1*Hx_2'* Wz *z2;


%Meshed
x_est3 = (Hx_3'* Wz * Hx_3)^-1*Hx_3'* Wz *z3;

%% Compute Rx matrices
%Kite
Rx1 = (state_exact1-x_est1)*(state_exact1-x_est1)';

%Radial
Rx2 = (state_exact2-x_est2)*(state_exact2-x_est2)';

%Meshed
Rx3 = (state_exact3-x_est3)*(state_exact3-x_est3)';



for k = 1:5
    erro(kk,1) = erro(kk,1) + abs(Rx1(k,k));
    erro(kk,2) = erro(kk,2) + abs(Rx2(k,k));
    erro(kk,3) = erro(kk,3) + abs(Rx3(k,k));
end

end


kite = 0;
rad = 0;
mes = 0;

for kk = 1:100
    kite = kite + erro(kk,1);
    rad = rad + erro(kk,2);
    mes = mes + erro(kk,3);
end

kite = kite/100;
rad = rad/100;
mes = mes/100;

% Plot
% figure()
% ep=1:100;
% plot(ep,erro)
% xlabel("Iteration")
% ylabel("Error magnitude")
% axis([1 100 0 0.4])
% legend({'Kite','Radial','Meshed'},'Location','northwest')
```

E SOP and Modelling pseudo-measurements as equality contraints Matlab Code

```
 %Exact state
x_exact =  [0.984339790360455 - 0.110985225979811i 0.896732673267327
- 0.302073267326733i 0.950000000000000 - 0.200000000000000i 0.962224938875306
- 0.151833740831296i 0.966243564356436 - 0.111864356435646i
1.00000000000000 + 0.00000000000000i].';


%Admitances
y12 = 1-1i*10;
y13 = 2*y12;
y23 = 3-1i*20;
y34 = y23;
y45 = 2*y12;


%Admitance matrix
Y = [y13 0 0 -y13 0 0;
    0 y12 -y12 0 0 0;
    0 -y12 y12+y23 -y23 0 0;
    -y13 0 -y23 y13+y23+y34 -y34 0;
    0 0 0 -y34 y34+y45 -y45;
    0 0 0 0 -y45 y45];


%Impedance
Z = Y(1:5,1:5)^-1;


%Droops
m1 = 0.5;
m2 = 0.5*Z(3,3)^-1;


%Observability matrices
C=[0 y12 -y12 0 0 0
   0 0 0 0 -y45 y45
   0 0 1 0 0 0
   0 0 0 0 0 1];
```

```
C2 = [0 0 0 0 0 1
     0 0 1 0 0 0
     Y(1,:)+Y(2,:)
     0 0 y23 -y23 0 0
     0 y12 -y12 0 0 0
     0 0 0 0 -y45 y45];

%Weight matrix
W = [4 0 0 0
     0 4 0 0
     0 0 10^6 0
     0 0 0 10^6];


%Dynamics
B1F= -Z*[0 0 m1*y23 -m1*y23 0
         0 0 -m1*y23 m1*y23 0
         0 0 -m2 0 0
         0 0 0 0 0
         0 0 0 0 0];


B2 = Z*[-m1 0
         m1 0
         0 m2
         0 0
         0 0];


%L matrix
aux_maxtrix_1 = (B2'*B2)^-1 *B2'*B1F;  %To help build E

E=zeros(3,5);


for k = 1:5
E(3,k) = Y(1,k)+Y(2,k);
end

for k = 1:5
E(1,k) = aux_maxtrix_1(1,k);
E(2,k) = aux_maxtrix_1(2,k);
end

aux_matrix_2 = C'*C;   %To help with L 6x6
aux_matrix_3 = -E';   %To help with L 5x3
aux_matrix_4 = C'*W*C;   %To help with L2 6x6

L = zeros (9,9);
```

```
L2 = zeros (9,9);

for k=1:6
    for m=1:6
        L(k,m) = aux_matrix_2(k,m);
        L2(k,m) = aux_matrix_4(k,m);
    end
end

for k=7:9
    for m=1:5
        L(k,m) = E(k-6,m);
        L2(k,m) = E(k-6,m);

    end
end

for k=1:5
    for m=7:9
        L(k,m) = aux_matrix_3(k,m-6);
        L2(k,m) = aux_matrix_3(k,m-6);

    end
end

%% Plot voltage mismatch OLS and EQ constrained
for kk=1:50

    %Measurement
    V50 = 1;
    I12 = -1.0740 + 1i*0.4306;
    I54 = 2.3048 - 1i*0.4514;
    I1j = -0.2128 + 1i*0.0700;


    %Setpoints
    V20 = 0.9500 - 1i*0.2000;
    I23 = -1.0000 + 1i*0.1000;

    y=[I12+normrnd(0,0.25) I54+normrnd(0,0.25)
    V20+normrnd(0,0.25) V50+normrnd(0,0.25)].';
    u=[I23 V20 I1j].';


    y2 = [V50+normrnd(0,0.25) V20+normrnd(0,0.25)
    I1j+normrnd(0,0.25) I23+normrnd(0,0.25)
    I12+normrnd(0,0.25) I54+normrnd(0,0.25)].';
    %Estimate state and penalties Eq cons
    aux_vector = C'*y;
    x_and_lambda = (L'*L)^-1*L'* [aux_vector(1)
    aux_vector(2) aux_vector(3) aux_vector(4) aux_vector(5)
    aux_vector(6) u(1) u(2) u(3)].';



    %Estiamted state simple ols
    aux_ols = (C2'*C2)^-1*C2'*y2;

    for k=1:6
        x_ols(k,kk) = aux_ols(k,1);
    end


    for k=1:6
        x_EqCon(k,kk) = x_and_lambda(k,1);
    end


    %Mismatches
    for k=1:6
        angle_mismatch_ols(k,kk) = abs(angle(x_ols(k,kk))-angle(x_exact(k,1)));
        magnitude_mismatch_ols(k,kk) = abs(abs(x_ols(k,kk))-abs(x_exact(k,1)));

        angle_mismatch_EqCon(k,kk) = abs(angle(x_EqCon(k,kk))-angle(x_exact(k,1)));
        magnitude_mismatch_EqCon(k,kk) = abs(abs(x_EqCon(k,kk))-abs(x_exact(k,1)));
    end
end


%Plot results
figure()%---------------------Elemento a elemento do estado

for m = 1:kk
for k = 1:5
    scatter(magnitude_mismatch_ols(k,m),angle_mismatch_ols(k,m),"^","blue");
    hold on;
    scatter(magnitude_mismatch_EqCon(k,m),angle_mismatch_EqCon(k,m),"x","red");
    hold on;
end
end


set(gca,'xscale','log');
set(gca,'yscale','log');
xlim([10^-4 10^-1]);
```

```
ylim([10^-4 10^-1]);
ylabel('Angle mismatch [rad]');
xlabel('Magnitude mismatch [V]');
title('Voltage mismatch');



%% Plot voltage mismatch EQ constrained OLS and EQ constrained WLS
for kk=1:50

    %Measurement
    V50 = 1;
    I12 = -1.0740 + 1i*0.4306;
    I54 = 2.3048 - 1i*0.4514;
    I1j = -0.2128 + 1i*0.0700;


    %Setpoints
    V20 = 0.9500 - 1i*0.2000;
    I23 = -1.0000 + 1i*0.1000;

    y=[I12+normrnd(0,0.25) I54+normrnd(0,0.25)
    V20+normrnd(0,10^-6) V50+normrnd(0,10^-6)].';
    u=[I23 V20 I1j].';



    %Estimate state and penalties Eq cons OLS
    aux_vector = C'*y;
    x_and_lambda = (L'*L)^-1*L'* [aux_vector(1) aux_vector(2) aux_vector(3)
    aux_vector(4) aux_vector(5) aux_vector(6) u(1) u(2) u(3)].';

    for k=1:6
        x_EqCon(k,kk) = x_and_lambda(k,1);
    end

    %Estimate state and penalties Eq cons WLS
    aux_vector2 = C'*W*y;
    x_and_lambda2 = (L2'*L2)^-1*L2'* [aux_vector2(1) aux_vector2(2)
    aux_vector2(3) aux_vector2(4) aux_vector2(5) aux_vector2(6) u(1) u(2) u(3)].';

    for k=1:6
        x_EqCon_W(k,kk) = x_and_lambda2(k,1);
    end


    %Mismatches
    for k=1:6
        angle_mismatch_EqCon_W(k,kk) = abs(angle(x_EqCon_W(k,kk))-angle(x_exact(k,1)));
        magnitude_mismatch_EqCon_W(k,kk) = abs(abs(x_EqCon_W(k,kk))-abs(x_exact(k,1)));

        angle_mismatch_EqCon(k,kk) = abs(angle(x_EqCon(k,kk))-angle(x_exact(k,1)));
        magnitude_mismatch_EqCon(k,kk) = abs(abs(x_EqCon(k,kk))-abs(x_exact(k,1)));
    end
end



%Plot results
figure()%---------------------Elemento a elemento do estado

for m = 1:kk
for k = 1:5
    scatter(magnitude_mismatch_EqCon(k,m),angle_mismatch_EqCon(k,m),"x","red");
    hold on;
    scatter(magnitude_mismatch_EqCon_W(k,m),angle_mismatch_EqCon_W(k,m),"o","green");
    hold on;
end
end



set(gca,'xscale','log');
set(gca,'yscale','log');
xlim([10^-4 10^-1]);
ylim([10^-4 10^-1]);
ylabel('Angle mismatch [rad]');
xlabel('Magnitude mismatch [V]');
title('Voltage mismatch');
```