Power System Dynamics and Control

Assignment 1: State Estimation.

Time:
13 March 2018, 08:15 - 12:00, ETZ E6
20 March 2018, 10:15 - 12:00, ETZ E6

Due on: 3 April, 2018

Assistants:
Dmitry Shchetinin and Nadezhda Davydova
dynamicsandcontrol@eeh.ee.ethz.ch

This document is structured as follows: Section 1 gives an overview over administrative issues. Section 2 describes the tasks you are to complete. Section 3 introduces the test systems. Section 4 gives a brief review of the theory behind the assignment. A more detailed description of the state estimation techniques can be found in the lecture notes and in the literature given in the references.

1 Administrative Issues

Submission Deadline 3 April 2018

Course homepage All files with be available on the course moodle page, https://moodle-app2.let.ethz.ch/course/view.php?id=4257

Evaluation All five exercises in this course are graded in a binary way, i.e. you will receive either a zero or a plus for each exercise. For exercise one, a team that produces the most efficient code for solving the SE problem with the decoupled formulation will receive an **extra** plus. If at the end of the semester you have at least five pluses, an additional 0.25 will be added to your grade on the final exam. If you have several pluses (but less than five) and we think that your performance at the final exam puts you in between two potential grades (for instance, between 5.25 and 5.5), then it is more likely that you will get a higher grade of these two. If you have not completed any of the exercises, your final grade will be solely based on your exam performance.

Groups Please form groups of up to three people. One report per group is sufficient. All group members receive the same grade, i.e., 0, '+', or '++'.

Data Files Download the input data and code files from the moodle page. You will receive:

- *.mat files that contain system data and measurements for two systems: 14-bus system and 1354-bus system
- Function that constructs the admittance matrix (f_Y_bus_compressed_v2017)
- Function that constructs the measurement function h (f_measFunc_h_v2017)
- Function that constructs the sparsity pattern of the measurement Jacobian H (f_meas_indices_and_H_sparsity_pattern_v2017)
- Function that constructs the measurement Jacobian H (f_measJac_H_v2017)
- Two script stubs which you need to complete
  - Script I: SE solution techniques (main_script_student_part1_v2017)
  - Script II: Bad data detection and correction (main_script_student_part2_v2017)
- Stub of a function for solving SE with the Gauss-Newton method which you need to complete (f_SE_NR_algorithm_v2017)

Deliverables Please send a zip-file containing the following files to the email address dynamicsandcontrol@eeh.ee.ethz.ch:

- The completed Matlab scripts and functions with unchanged names and function definitions. Please make sure to include all necessary Matlab files so that we can run your code smoothly.
- A brief report in pdf format, no longer than 6 pages, giving answers to the questions marked 'Question' or 'Report' in the task description.

The archive has to be named `Group_Lastnames.zip`, where `Lastnames` are the last names of all members in the group. E.g., `Group_JovicicMarkovicShchetinin.zip` if Alexandar Jovicic, Uros Markovic and Dmitry Shchetinin were in this group.

Evaluation The grading will be influenced by the correctness of the solution, the readability and efficiency of the code, and the clarity of the report.

Responsible assistant Nadezhda Davydova

Do not hesitate to contact the assistant of the course if there are any questions on this assignment and/or if you are puzzled - *dynamicsandcontrol@eeh.ee.ethz.ch*.

This assignment aims to help you learn, not to test your knowledge!

2 Assignment

2.1 State Estimation

Assume you are a system operator, and you have measurements of your system such as power flows and voltages. Using these measurements, you can compute all your system states. However, your measurements are noisy. To get a robust guess of your system state, you decide to use state estimation techniques. In this task, you will analyze different aspects of solving the state estimation problem by the Gauss-Newton method.

2.1.1 Data Processing

Before solving a WLS (Weighted Least Squares) problem, you need to process the available data. You are provided with a script stub which you should complete with your code (main_script_student_part1_v2017). Additionally, you are provided with functions that compute the system admittance matrix, the measurement function h, the measurement Jacobian H and its sparsity pattern. You also have topology and measurement data, see Section 3.1. Follow these steps:

- *Question:* What dimension do the measurement Jacobian H and the measurement function h have? Explain your answer.
- Compile the measurement vector z using the indices of available measurements in a list of all possible measurements of each type contained in a structure '`ind_meas`'.
- Compile the diagonal matrix of measurements' weights W and a matrix $\widetilde{W}$, whose elements are the square roots of corresponding elements of matrix W, i.e. $W = \widetilde{W} \cdot \widetilde{W}$.

2.1.2 Task 1. Effects of Sparsity and Linear System Solver

Develop a *state estimation* (SE) program based on the Gauss-Newton algorithm. The solution algorithms should be contained in the Matlab function 'f_SE_NR_algorithm_v2017'. A detailed description of the function's input and outputs can be found in the function stub. For all algorithms, assume that bus 1 is a reference bus and the vector of state variables is $x = [\theta, V]$. Your code should include the following techniques for solving the SE problem:

1. Solve normal equations by inverting matrix G. Note: Matlab may give you a warning to substitute the inverse operation with the backslash. You should ignore this warning.
2. Solve normal equations by the Cholesky factorization of matrix G. For this, you should use a backslash operator '\'. This is generally a recommended way to solve a linear system in Matlab because it enables Matlab to automatically choose a solver that is best suited for solving a particular linear system. In this case, Matlab will choose to use the Cholesky factorization of matrix G.

3. Use the orthogonal factorization method of matrix $\tilde{H} = \widetilde{W}H$. Please use exactly the following expression `[Q, R, e] = qr(`$\tilde{H}$`, 0)`, which produces a so-called economy-size decomposition $\tilde{H}(:,\mathrm{e}) = Q\cdot R$ with a minimum number of fill-in elements. Matrices Q and R here correspond to matrices U and Q_n in Section 4.3. Do not forget to permute back your vector of updates!

4. Use the hybrid method, which produces the orthogonal factorization of matrix $\tilde{H} = \widetilde{W}H$ without returning matrix Q. To reduce the number of fill-in elements, you should first obtain a permutation of $\tilde{H} = \widetilde{W}H$ by using `p=colamd(`$\tilde{H}$`)` and then apply R`=qr(`$\tilde{H}$`(:,p),0)`. Do not forget to permute back your vector of updates!

In all cases, use the flat start for values of state variables x and the 1-norm $\varepsilon = \|\Delta x\|_1 < 1 \times 10^{-5}$ as a stopping criterion. Use the 14-bus system to test that each algorithm converges to a solution (for this, use the data set `GoodMeasurement_14_bus`). Once your code is debugged, do the following steps:

- *Report:* For the 14-bus system, determine the computation time and number of iterations for each algorithm for the cases when the measurement Jacobian is constructed as 1) sparse 2) dense matrix. Parameter '`H_sparse`' controls the way this matrix is formed. Note: you should only count the time spent doing matrix operations, i.e. exclude the time spent forming vector h and matrix H. Present your results in a form of a table similar to Table 1.
- *Report:* Repeat these steps for the 1354-bus system. Present your results in a form of a table similar to Table 1.
- *Question:* Do sparsity and the solution method affect the number of iterations? Why or why not?
- *Question:* How does the sparsity affect the computation time? Is there any relation to the system size? Explain your answer.
- *Question:* What is the fastest solution algorithm? Why do you think that is?

2.1.3 Task 2. Assessing Number of Required Operations and Numerical Stability

For a dense square matrix A, the number of operations required for solving the linear system $Ax = b$ is proportional to n^3, where n is the number of rows/columns of A. For sparse

Table 1: Solution of SE problem by different methods

	Inverse		Cholesky		QR		Hybrid	
	sparse	full	sparse	full	sparse	full	sparse	full
time, s								
N_{iter}								

matrices, efficient solution algorithms have been developed so that the number of required operations becomes proportional to the number of nonzero elements in the matrix. Assess the number of required operations for solving a linear system with 1) matrix inverse, 2) Cholesky decomposition, 3) QR decomposition. For this task, you will use the 1354-bus system to ensure that sparsity effects become noticeable. When constructing the matrices described below, use the flat start for the values of state variables. Follow these steps:

- *Report:* Determine the density factors of matrices H, G and G^{-1}. Recall that density, measured in percent, is the ratio of the number of nonzero elements to the total number of elements in the matrix.
- *Report:* Obtain the Cholesky decomposition of matrix $G = LL^T$ and record the density factor of L. Please use exactly the following expression `[L,p,S] = chol(G)` to ensure that the algorithm will try to minimize the number of fill-in elements.
- *Report:* Obtain the QR decomposition of matrix H and record the density factors of the resulting matrices R and Q. Please use exactly the following expression `[Q, R, e] = qr(H, 0)` to ensure that the algorithm will try to minimize the number of fill-in elements.
- *Question:* Which decomposition leads to the highest number of fill-in elements? Which preserves sparsity the best? How does it correspond to the computation times that you obtained in the previous section?

So far we have concentrated on analyzing the efficiency of different solution methods. Another important characteristic of these methods is their numerical stability, which depends on both the inherent stability of the method itself and the 'quality' of its typical input data. Here, we will concentrate on the latter. For a linear system, the 'quality' of the input data can be measured by a so-called condition number of the matrix, which is the ratio of the largest singular value of that matrix to the smallest singular value. The higher this number is, the more susceptible to small perturbations in the input data and thus more numerically unstable the solution becomes. If this number is high, the linear system $Ax = b$ is said to be ill-conditioned, which means that small errors in the entries of A and b translate into significant errors in elements of x. Note: condition number is not a property of the solution algorithm, it is purely a characteristic of the linear system itself. We know that the Cholesky factorization works with matrix G, whereas the QR decomposition operates on matrix H. Therefore, we can compare the numerical stability of these two methods by estimating the quality of the input data that they are provided. To do that, use the 1354-bus system and the flat start for the values of state variables. Follow these steps:

- *Report:* Determine the condition number of matrix G. Please use the dense representation of this matrix and function `cond(`G`)`.
- *Report:* It can be shown that the condition number of matrix H is equal to the condition number of matrix R obtained by the QR decomposition of H. Compute the condition number of matrix R. Make sure to use the dense representation of matrix R for computing its condition number.

- *Question:* Which decomposition is more numerically stable?
- *Question:* Based on the results that you have obtained so far, which method would you use for your state estimation program? Why?

2.1.4 Task 3. Comparing Efficiency of Full and Decoupled Formulations

When solving power flow problems, one can use a decoupled formulation of the Newton-Raphson method to speed up the computation process. Not surprisingly, the same idea can be applied for solving the state estimation problem. In this task you will implement a decoupled algorithm that solves the SE problem in a form of the normal equations. You can find the information on this decoupled algorithm on the moodle (SE Chapter Weighted Least Square Estimation, section 2.7). Use the sparse representation of the measurement Jacobian and the flat start for the values of state variables. Follow these steps:

- Include the code for the decoupled method into the function 'f_SE_NR_algorithm_v2017'. A choice of full/decoupled algorithm should be controlled by parameter `H_decoupled`.
- *Report:* First, use the 14-bus system. Record the computation time and the number of iterations required for the decoupled method to converge.
- *Report:* Repeat the process for the 1354-bus system.
- *Question:* Compare the obtained results to the results obtained for the Cholesky factorization in Task 1. What are advantages/disadvantages of using full/decoupled formulation?

Note: Try to make your code for the decoupled formulation as efficient as possible. We will measure the computation time for the 1354-bus system for all groups that submit their code and the group that produced the fastest code will receive an additional plus for the exercise.

2.2 Dealing with Bad Data

In the first part of this exercise, you developed a solution algorithm for the state estimation problem using Gauss-Newton method. In this part, you will work with detecting and correcting bad data for four provided datasets. When writing your code, use the script 'main_script_student_part2_v2017'.

Task 4 – Bad Data Detection

After implementing the state estimation, you are now able to compute the state of your system from your measurements. However, every now and then there are bad measurements. These are the measurements which are not simply affected by noise but are actually incorrect. When relying on such measurements, your WLS-based state estimation may mislead you. Luckily, there are methods to identify bad data in your data set.

Develop a routine that detects and classifies bad data sets based on the residual analysis. Additionally, there may or may not be information from the SCADA system about bad measurements, which in this exercise is represented by a parameter '`meas.warning`'. With this information, you are able to find good data sets, bad data sets where the errors are in redundant measurements, and data sets with an error in a critical measurement. Follow these steps:

- Obtain the state estimation result and perform a residual analysis for all the data sets A, B, C and D.
- *Report:* Plot the residuals of the four data sets.
- *Report:* Based on the residuals and the warning flag, infer which data sets are good, can be corrected, or have an error in a critical measurement. Use threshold value of 4 to detect bad measurements.
- *Question:* What is a critical measurement? What is a redundant measurement?
- *Question:* Propose an additional measurement that adds redundancy to all critical measurements. Elaborate on your choice.

Task 5 – Bad Data Correction

Under certain circumstances, bad measurements can be corrected. Based on the results of Task 4, develop a routine that, if possible, removes bad measurements and computes the correct state estimate. Use the results from Task 4 as input for this task. Follow these steps:

- Implement a routine that removes a detected bad measurement from the data and runs the state estimation again, if necessary.
- *Question:* What happens if a critical measurement is removed from the data set?
- *Report:* Compare the state estimation results before and after the correction of a faulty data set with a correct data set. Plot voltage magnitudes and angles before and after data correction.

In the end, the system operator should know whether the data set is good or at least correctable, and if so the operator should have a good estimate of the system state.

3 System Description

Two test systems are used in this exercise: 14 bus system and 1354 bus system. The 14 bus system is depicted in Figure 1. It consists of two voltage levels: buses 1 to 5 are operated at 132 kV and buses 6 to 14 at 33 kV. It has 11 loads that are covered by two generators. The power from the generators is delivered to the loads through the transmission grid (Table 2), which is formed by 14 buses, 15 lines and 3 transformers. One of the transformers actually

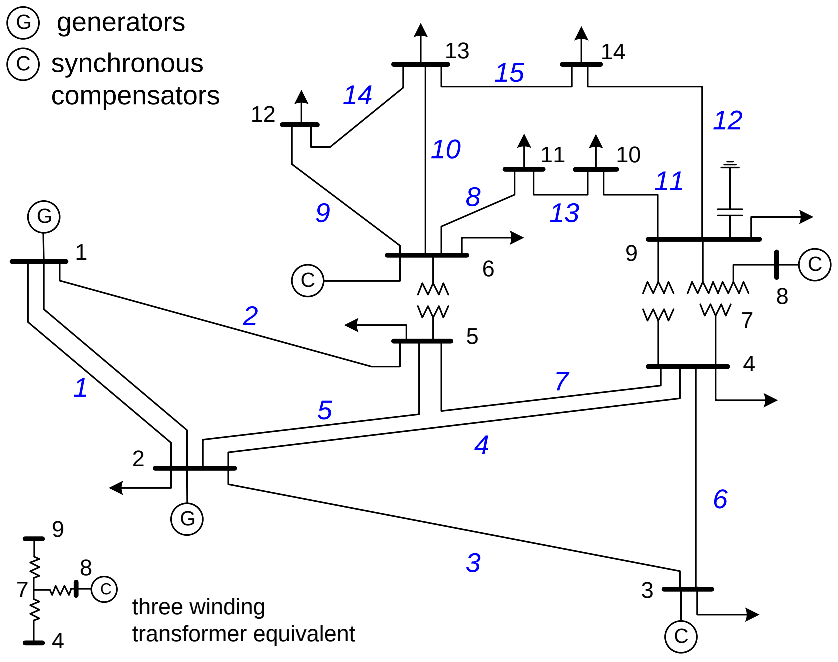
G generators
C synchronous compensators
three winding transformer equivalent
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Figure 1: Single line diagram of IEEE 14-bus system.

has three connections, and is modeled as three individual transformers connected by bus 7. A shunt capacitor with constant susceptance of $b^{\mathrm{sh}} = 0.19$ p.u. is installed at bus 9 to support the voltage. Reactive power is provided by synchronous compensators at buses 3, 6 and 8.

Topology information for each system is stored in the Matlab structure `topo`. Fields `nBus` and `busNumbers` hold the number of buses and their numbers, `busShuntB`, `busShuntG`, `lineShuntB` and `lineShuntG` hold real and imaginary parts of bus and line shunt admittances, and `lineR` and `lineX` – line impedance. `lineTapRatio` and `linePhaseShift` have the tap ratios and phase shifts of transformers, and `FromBusRef` and `ToBusRef` are the start and end nodes of lines.

3.1 Measurements

The measurement devices installed in the system provide information on active and reactive power flows, voltage magnitudes, and active and reactive power injections. Measurement data is provided in the form of MatLab structures. Each structure contains several fields with measurements. The fields are `BusV` for the voltage measurements, `BusInjP` and `BusInjQ` for the active and reactive power injections, respectively, and `LineP_FT`, `LineP_TF`, `LineQ_FT` and `LineQ_TF` for the active and reactive power flows. Here, FT stands for 'from-to' and TF for 'to-from'. The node measurements are sorted by node index, and the line measurements are ordered according to the from-bus and the to-bus indices (which slightly differs from the order in Table 2). The value NaN means that a measurement is not available. Voltage phase angles are not measured as this is a technically difficult task. Recently, phase measurement units (PMUs) relying on GPS time stamps have become more widely available but they are

Table 2: Parameters of the branches, all values in [p.u.].

Asset	From	To	Resistance	Reactance	Shunt Susceptance	Tap ratio
Lines	1	2	0.019	0.059	0.053	1.000
	1	5	0.054	0.223	0.049	1.000
	2	3	0.047	0.198	0.044	1.000
	2	4	0.058	0.176	0.034	1.000
	2	5	0.057	0.174	0.035	1.000
	3	4	0.067	0.171	0.013	1.000
	4	5	0.013	0.042	0.000	1.000
	6	11	0.095	0.199	0.000	1.000
	6	12	0.123	0.256	0.000	1.000
	6	13	0.066	0.130	0.000	1.000
	9	10	0.032	0.085	0.000	1.000
	9	14	0.127	0.270	0.000	1.000
	10	11	0.082	0.192	0.000	1.000
	12	13	0.221	0.200	0.000	1.000
	13	14	0.171	0.348	0.000	1.000
Transformer	4	7	0.000	0.209	0.000	0.978
	4	9	0.000	0.556	0.000	0.969
	5	6	0.000	0.252	0.000	0.932
	7	8	0.000	0.176	0.000	1.000
	7	9	0.000	0.110	0.000	1.000

still not standard equipment.

The standard deviation σ of all measurements is also provided in the measurement data, with fields named `std_*`, where * is the name of the measurement.

4 Additional Material

4.1 Theoretical background

The objective of State Estimation is to find the most probable system state x given a set of observations z. The WLS estimator will minimize the following objective function $J(x)$:

$$J(x) = [z - h(x)]^T \cdot W \cdot [z - h(x)]. \tag{1}$$

where $h(x)$ is a vector that maps system states into the space of measurements. At the minimum, the first order optimality conditions will have to be satisfied:

$$g(x) = \frac{\partial J(x)}{\partial x} = -H^T(x) \cdot W \cdot [z - h(x)] = 0, \tag{2}$$

where $H(x) = \frac{\partial h(x)}{\partial x}$ is the measurement Jacobian.

Expanding the non-linear function $g(x)$ into its Taylor series around the state vector x^k, and neglecting second and higher order terms yields:

$$g(x^k) + \frac{\partial g(x^k)}{\partial x}(x - x^k) = 0, \tag{3}$$

where $\frac{\partial g(x^k)}{\partial x}$ is called a gain matrix $G(x^k) \approx H^T(x^k) \cdot W \cdot H(x^k)$.

This leads to the iterative solution scheme known as the Gauss-Newton method, with the updates of $\Delta x^{k+1} = x^{k+1} - x^k$ for the next iteration $k+1$ obtained by solving the equation:

$$G(x^k)\Delta x^{k+1} = -g(x^k). \tag{4}$$

Substituting $g(x^k)$ using (2) the following equations are obtained:

$$G(x^k)\Delta x^{k+1} = H^T(x^k) \cdot W \cdot [z - h(x^k)]. \tag{5}$$

These equations are known as the normal equations.

4.2 SE based on normal equations

1. Choose initial value of x^k (usually "flat start")
2. Update iteration number $k = k + 1$
3. Compute $H(x^k)$, $h(x^k)$
4. Compute matrix $G(x^k) = H^T(x^k) \cdot W \cdot H(x^k)$
5. Compute right hand side $g(x^k) = -H^T(x^k) \cdot W \cdot [z - h(x^k)]$
6. Solve $G(x^k)\Delta x = -g(x^k)$ by either inverting matrix $G(x^k)$ or decomposing $G(x^k) = LL^T$ (Cholesky factorization) and solving for Δx^k using backward and forward substitutions
7. Test convergence: $\max|\Delta x^k| \leq \varepsilon$?
   if no, $x^{k+1} = x^k + \Delta x^k$ and go to step 2.

4.3 Derivation of the orthogonal SE method

It has been noticed that normal equations are prone to numerical instabilities. Therefore, other solution methods have to be sought to efficiently solve the linear system of equations at each step of Gauss-Newton iteration method. One such method is based on the QR decomposition of the measurement Jacobian.

Normal equation formulation:

$$G \cdot \Delta x = H^T W \left[z - h(x) \right] \tag{3.1}$$

Recalling that $G = H^T W H$, (3.1) can be easily transformed to:

$$\boxed{\tilde{H}^T \tilde{H} \cdot \Delta x = \tilde{H}^T \Delta \tilde{z}}, \tag{3.2}$$

where

- $\tilde{H} = W^{1/2} H$
- $\Delta \tilde{z} = W^{1/2} \Delta z$.

Orthogonal factorization method avoids inversion of the G matrix in (3.1) by using special decomposition of $\tilde{H}$ in two matrices $Q_n U$:

$$\tilde{H} = QR = \underbrace{\left[Q_n \ Q_0 \right]}_{Q} \cdot \underbrace{\begin{bmatrix} U \\ 0 \end{bmatrix}}_{R} = Q_n U, \tag{3.3}$$

where Q is orthogonal matrix, meaning $Q^T = Q^{-1}$ and R is trapezoidal, meaning that U is upper triangular matrix and the rest are zero.

We add in (3.2) multiplier QQ^T which is equal to unity matrix I and, thus, does not disturb equality:

$$\tilde{H}^T \underbrace{QQ^T}_{=I} \tilde{H} \cdot \Delta x = \tilde{H}^T \Delta \tilde{z}, \tag{3.4}$$

then,

$$\underbrace{\tilde{H}^T Q}_{R^T} \underbrace{Q^T \tilde{H}}_{R} \cdot \Delta x = \tilde{H}^T \Delta \tilde{z}. \tag{3.5}$$

Employing decomposition (3.3) the right side of the equation becomes:

$$R^T R \cdot \Delta x = R^T Q^T \Delta \tilde{z}. \tag{3.6}$$

Eliminating zero elements in R, the equation becomes:

$$U^T U \cdot \Delta x = U^T Q_n{}^T \Delta \tilde{z}. \tag{3.7}$$

Finally, we can eliminate U^T from both parts of equation:

$$\boxed{U \cdot \Delta x = Q_n{}^T \Delta \tilde{z}}. \tag{3.8}$$

This equation is the basic one for the orthogonal method application. U is upper triangular matrix and once $Q_n{}^T \Delta \tilde{z}$ is computed, Δx can be obtained by back substitution.

4.4 Hybrid Method

Comparing (3.2) and (3.7), it can be concluded that matrix U in the QR factorization is the same as that of the Cholesky factorization of G. Based on this observation, a hybrid scheme can be devised as follows:

1. Obtain U by orthogonal transformations on $\tilde{H}$. There is no need to keep track or save the components of Q.
2. Compute the independent vector $\Delta z_h = \tilde{H}^T \Delta \tilde{z}$.
3. Obtain Δx by solving the system $U^T U \Delta x = \Delta z_h$.

Hence, the normal equations are solved at step 3 but U is obtained by orthogonal transformations on H rather than by Cholesky factorization of G.

4.5 Structure of the measurement Jacobian

The measurement Jacobian contains partial derivatives of the measurement function. Its structure corresponds to the arrangement of measurements in measurement function and measurement vector and for this exercise is as follows:

$$H^{\text{full}} = \begin{bmatrix} 0 & \frac{\partial V_{mag}}{\partial V} \\ \frac{\partial P_{\text{inj}}}{\partial \theta} & \frac{\partial P_{\text{inj}}}{\partial V} \\ \frac{\partial Q_{\text{inj}}}{\partial \theta} & \frac{\partial Q_{\text{inj}}}{\partial V} \\ \frac{\partial P_{\text{flow}}}{\partial \theta} & \frac{\partial P_{\text{flow}}}{\partial V} \\ \frac{\partial Q_{\text{flow}}}{\partial \theta} & \frac{\partial Q_{\text{flow}}}{\partial V} \end{bmatrix} \qquad H^{\text{decoupled}} = \begin{bmatrix} 0 & \frac{\partial V_{mag}}{\partial V} \\ \frac{\partial P_{\text{inj}}}{\partial \theta} & 0 \\ 0 & \frac{\partial Q_{\text{inj}}}{\partial V} \\ \frac{\partial P_{\text{flow}}}{\partial \theta} & 0 \\ 0 & \frac{\partial Q_{\text{flow}}}{\partial V} \end{bmatrix} \tag{14}$$

where each of the partial derivative denotes $m \times n$ matrix, m is the number of measurements and the n number of state variables θ or V. H^{full} denotes the full measurement Jacobian. $H^{\text{decoupled}}$ is the decoupled measurement Jacobian, analogous to the decoupled Power Flow.

4.6 Bad Data Identification

Measurements may contain random errors due to various reasons: the finite accuracy of the measurement devices, telecommunications, unexpected interference. Bad data is a measurement that contains large error. Some of the errors, such as negative voltage magnitudes can be easily identified by pre-screening. However, the ability to filter out biased measurements should be expected from SE if sufficient redundancy exists. The nature of this filter depends on the specific estimation method.

For the WLS estimation method, detection and identification of the bad data can be performed by the analysis of obtained residuals values.

4.6.1 Residual sensitivity matrix

The linearized measurement equation is:

$$\Delta z = H\Delta x + e \tag{15}$$

Note that measurement residuals may still be correlated even when the measurement errors are not. The WLS estimator for the linearized model is:

$$\Delta\hat{x} = G^{-1}H^T W\Delta z \tag{16}$$

and the estimated value of $\Delta\hat{z}$, also called the *hat* matrix, is:

$$\Delta\hat{z} = H\Delta\hat{x} = HG^{-1}H^T W\Delta z = K\Delta z. \tag{17}$$

The matrix K has some interesting features: a diagonal element being relatively high relative to off-diagonal elements in the row implies that local redundancy around corresponding meter is poor.

It can be proven that the measurement residuals and the measurement covariance can be expressed as:

$$\begin{aligned} r &= (I-K)e = Se, & (18)\\ Cov(r) &= \Omega = SW^{-1} & (19)\end{aligned}$$

where S is the residual sensitivity matrix, representing the sensitivity of the measurement residual to measurement errors.

The residual covariance matrix has important properties and can be used for the identification of the bad data:

- Ω is a real and symmetric matrix
- $\Omega_{ij}^2 \leq \Omega_{ii}\Omega_{jj}$
- $\Omega_{ij} \leq \frac{\Omega_{ii}+\Omega_{jj}}{2}$
- if $\Omega_{ij} \geq \epsilon$ then measurement i and j are said to be strongly interacting. The threshold ϵ depends on the network and the measurement topology, as well as desired level of selectivity among measurements

4.6.2 Use of Normalized residuals for Bad Data Detection

The residue can be normalized by dividing its absolute value by the corresponding diagonal entry in the residual covariance matrix Ω, resulting in r_i^N :

$$r_i^N = \frac{|r_i|}{\sqrt{\Omega_{ii}}} = \frac{|r_i|}{\sqrt{W_{ii}^{-1}S_{ii}}} \tag{20}$$

If there is no bad data, measurement residuals are relatively small and do not exceed some threshold (which can be determined from the experience and knowledge of the measurement system accuracy). The measurement residual of a critical measurement will always be zero; only redundant measurements may have nonzero measurements residuals.

When the analysis of residuals indicates presence of bad data, the largest normalized residue corresponds to the bad measurement. The condition for such detection of the bad measurement is that it is single and non-critical (redundant). This bad measurement can be deleted from the set of available measurements and the SE problem can be solved again.

References

[1] Fernando L. Alvarado; "Solving Power Flow Problems with a Matlab Implementation of the Power System Applications Data Dictionary", *Proceedings of the 32nd Hawaii International Conference on System Sciences,* 1999

[2] Ali Abur, Antonio Gómez Expósito "Power System State Estimation: Theory and implementation"", *Marcel Dekker, Inc.* 2004

[3] A.Monticelli "State Estimation in Electric Power Systems: A Generalised Approach", *Kluwer's Power Electronics and Power Systems Series*, 1999

[4] Göran Andersson, "Modelling and Analysis of Electric Power Systems" Lecture 227-0526-00, ITET ETH Zürich