Energy Management System

A thesis submitted in partial fulfillment
of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

Electrical Engineering (Power and Automation)

by

Pranav Verma 2015EE30528
N. Akash 2015EE30522

Under the guidance of
Prof. B K Panigrahi

Department of Electrical Engineering,
Indian Institute of Technology Delhi.
November 2018.

Certificate

This is to certify that the thesis titled **Energy Management System** being submitted by **Pranav Verma and N. Akash** for the award of **Bachelor of Technology** in **Electrical Engineering (Power and Automation)** is a record of bona fide work carried out by them under my guidance and supervision at the **Department of Electrical Engineering**. The work presented in this thesis has not been submitted elsewhere either in part or full, for the award of any other degree or diploma.

B K Panigrahi
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Abstract

With increasing complexities in electrical grids as well as increasing demand for power, it is pertinent that the grid is safely and efficiently maintained. This can be done using an end to end system which can perform an Observability Analysis and State Estimation based on the grid data. This can enable engineers to make better decisions. Such a system has been implemented in a modular, easy to use python package.

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Pranav Verma,
N. Akash

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 SECTION NAME

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

You should cite papers in the following manner: Bayliss et al. [**?**] gave an iterative method for Helmholtz equation etc. Similar work has been done in [**?**, **?**, **?**].

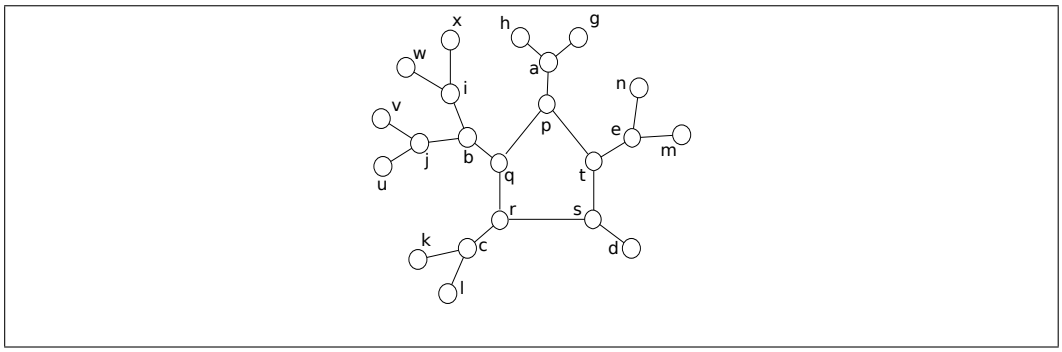

Figure 1.1: Pentagon $pqrst$

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida

item 1	item 2
abcde	5
pqrst	4

Table 1.1: A sample table

mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1.2 SECTION NAME

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 2

CHAPTER NAME

2.1 SECTION NAME

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

2.2 SECTION NAME

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

2.3 SECTION NAME

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 3

Setup

In this chapter we explain the setting up of RTDS model and the code for observability analysis and state estimation

3.1 RTDS

RTDS [**?**] is a popular simulation hardware used to run real time power line simulations. It is commonly used for studies of protective relays, control systems, power hardwares, etc. We use this hardware primarily to simulate a Transmission line and generate and record data for further analysis, such as observability analysis, state estimation, Optimum Power Flow, etc.
RSCAD is a simulation software that is used to create models of power lines, to work on the RTDS Simulator Hardware. For our project we set up a model for a 14 bus power line system, with specifications about lines and buses taken from IEEE 14 Bus power system [**?**]. The model has the following characteristics:

- Number of Buses: 14
- Number of Lines: 20
- Number of Generators: 5

Our network incorporated circuit breakers on each line, their control mechanisms, various GTNET-SKT connections to transmit data, and power measurement units for real and reactive power injections and line flows in the 14 bus network. The final system that we designed looked like Figure **??**.

It had the following key blocks, shown here for reference. (**??** to **??**)

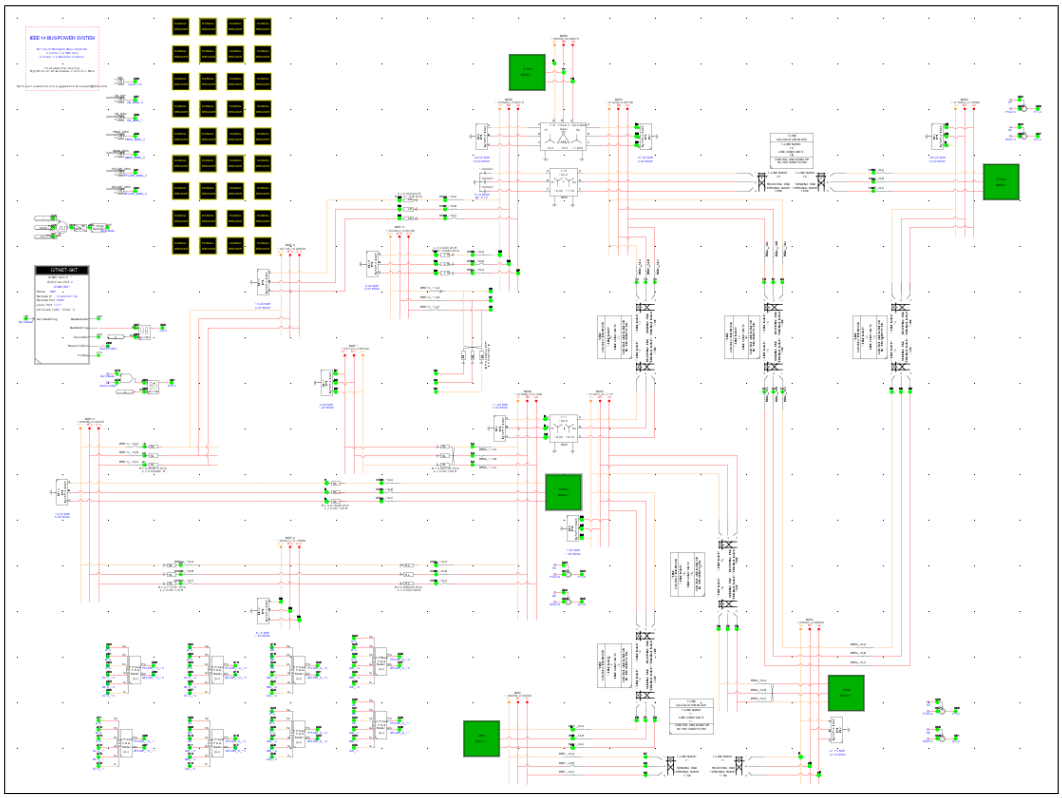

Figure 3.1: IEEE 14 Bus Power Line System Model on RSCAD

The GTNET-SKT block (Figure **??**), which is crucial for sending and receiving data from RTDS to our server, which records and processes data for further analysis of the power system. To function, it needs the following parameters specified:

- All the signals to be measured. In our case, these were the status about all the circuit breakers in the system, the real and complex power injections, real and complex power flows in the lines, and status about availability of each measurement
- The remote IP address and port of the server to which data is to be sent.

GTNET-SKT sends raw data over UDP protocol to the server, and each measurement is just a 32 bit Floating point number or a 32 bit signed Integer. It further places a restriction, that we can send only upto 300 measurements per GTNET-SKT block. For more measurements, we add more blocks, and

server will then have to communicate with RTDS over multiple ports, which would become complicated as more blocks are added. Furthermore, in real life situations, not all data measurements are available, so we also need to check which measurements correspond to which variables. We simulate this condition by toggling the availability of measurements recorded by GTNET-SKT. With all this in mind, we see that while the essential information about which measurement corresponds to which variable is critical, we can not simply transmit the measurement labels with each measurement since we would be essentially doubling the number of variables used for transmission of same number of measurement values.

To counter these points, we developed an efficient scheme to transmit information about labels over GTNET-SKT.

Firstly, we figured that all the topological information about a network (the number of buses, number and connectivity of lines, generator buses etc.), as well as information about the line impedances remain fixed, and that information can be stored in files we will call *casefiles*, since most networks used for power line research are typically standard networks, described in IEEE standards. MATPOWER, a MATLAB based application for calculating load flow of power systems, and PyPower, it's python equivalent, typically work on these standard configurations, and they store the above information about a network in casefiles, in a similar fashion to what we are doing. Specifying the casefile will tell us everything we need to know about the topology of the network, it's number of buses, lines, and circuit breakers, as well as the total real and reactive power injection and flow measurements.

To allow the flexibility to work with a non-standard network configuration, we gave the user an option to create it's own casefile, and we would then use their custom casefiles to process topological information about their network instead.

Second, we created an order in which the measurements from the GTNET-SKT will be transmitted. We will first send all the circuit breaker information, then the real power injections at the buses, then the reactive power injections, then real power line flows, and then reactive power line flows. The order of transmitting data about buses follows the following rule:

- *For two buses with bus numbered* ***a*** *and* ***b****, measurement related to bus*

a** will come before bus **b** if **a** < **b

For measurements related to lines, we follow the following rule:
*Information about a line between buses numbered **a** and **b** (where **a** < **b**), will come before information about a line between buses numbered **c** and **d** (where **c** < **d**) if:*

$$\boldsymbol{a} < \boldsymbol{c},\ \textit{or}\ (\boldsymbol{a} = \boldsymbol{c}\ \textit{and}\ \boldsymbol{b} < \boldsymbol{d})$$

Thirdly, we reasoned that status of availability of each measurement can be encoded in 1 bit each. Therefore, for maximum of n possible measurements needed, we will need exactly n bits, or approx $n/32$ measurement variables to tell whether each measurement is available or not. Coupled with the ordering in which each measurement comes from RTDS, this allows us to know exactly which information is coming, without explicitly sending the labels!
We will call these $n/32$ measurement variables as **status numbers**, since they provide us information about availability status of each measurement.
The order in which RTDS sends information about the system:

- **case_number**, informing us about the topology of the network
- **status number** about the availability of circuit breaker measurements,
- **Circuit breaker measurements**
- **status number** about the availability of real power injections
- **Real power injection measurements**
- **status number** about the availability of reactive power injections

  ...
- **status number** about the availability of reactive power flows
- **Reactive power flow measurements**

3.2 Server side data receiving and processing

We provide a remote IP address and port number to GTNET-SKT, which is the address at which our server is listening to receive the packets of data from RTDS. We wrote a Python Script to act as the server and receive data. This script is present in the Appendix.
Since RTDS just sends packets with 32 bit blocks for each measurement, we need to process the binary message packet to obtain meaningful information. To decode this information, we also need to remember that the information can either be a Floating point value, or an Integer. We had to observe and figure out the fact that for floating point numbers, the GTNET-SKT encodes it in IEEE 754 standard for 32 bit Floating point format, which is that the MSB is the sign bit, then the next 8 bits are for the exponent, and the rest 23 bits are for the mantissa. To get the floating point number from a 32 bit binary string, as given by IEEE 754 standards:

- Let e= decimal value of exponent part of the string
- For base, we note that mantissa represents the fractional part of the number. Say, for example mantissa is given by $0111001\ldots$, then the number, in binary form will be given by $1.0111001\ldots$ Base is the decimal value of this number
- Finally, the floating point number f=base$\cdot 2^{(e-127)}$

If, however, the measurement is an integer value, a simple binary to decimal conversion of that 32 bit binary number will give us the result. We use the integer values mainly for the status number measurements, since encoding the status of each measurement will involve encoding the binary string into an integer number or a float, and the former one is much much easier to do, and much easier to implement for a user on RTDS side, since that is where the status number will come from. Thus, given our expectation of data type of each measurement, we can appropriately decode the binary data into real valued measurements about active and reactive power.

3.3 Casefiles

MATPOWER, and by extension, PyPower provides us the casefiles for a lot of different IEEE standard transmission line systems, ranging from 6-bus or 9-bus system to 300 bus systems. If we know the casefile for the system at hand, we can decode the topological information about the system, as well as get useful information like line impedances, and generation costs defined for the generators in the system. Decoding the data received from RTDS, we learn the case_number of the system from the first measurement, and we can use this to get the appropriate case file. All case files contain the following data:

- **Bus Data:** For each bus in the system, tells us about its type (PV,PQ or Slack), baseKV rating, voltage limits, real and reactive power loads at that bus, etc.
- **Line/Branch Data:** For each line/branch, it tells us about the buses it is connected to, its impedances, status, etc.
- **Generator Data:** Tells us which bus each generator is connected to, it's real and reactive power generation, terminal voltage, reactive power limits, AGC participation factors, etc.
- **Generation Cost Data:** For each generator, it has the coefficients *(a,b,c)* of generation cost as a function of real power generated ($Cost = a \cdot P^2 + b \cdot P + c$), and the time it stays shut down.

3.4 Observability Analysis

3.5 State Estimation

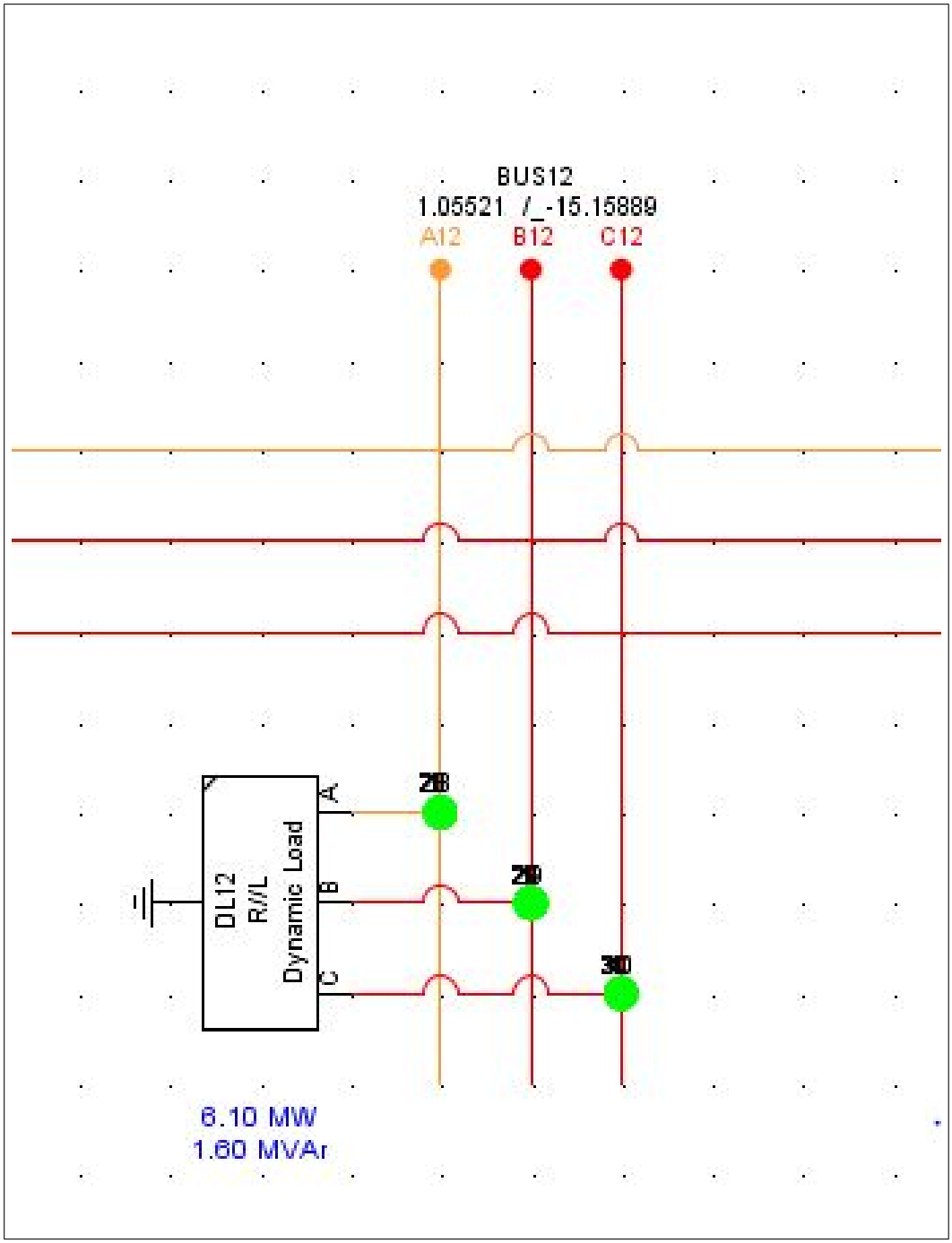

Figure 3.2: Typical Bus in RSCAD

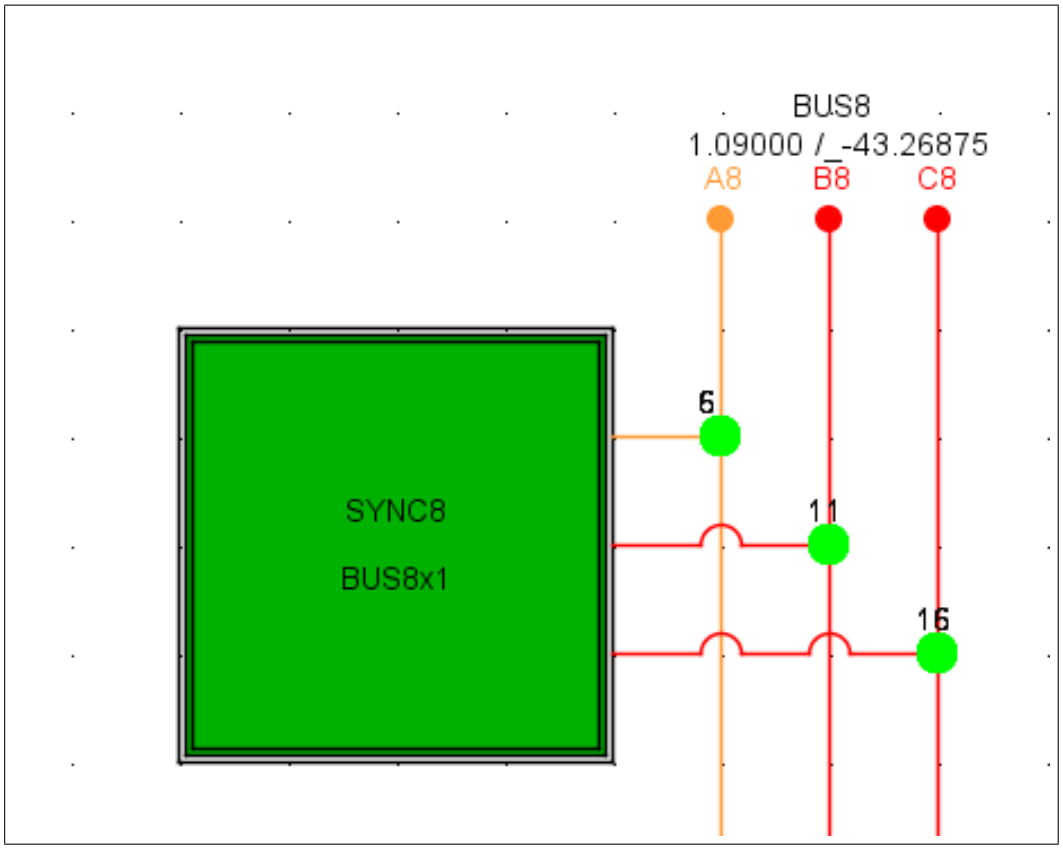

Figure 3.3: Generator Block in RSCAD

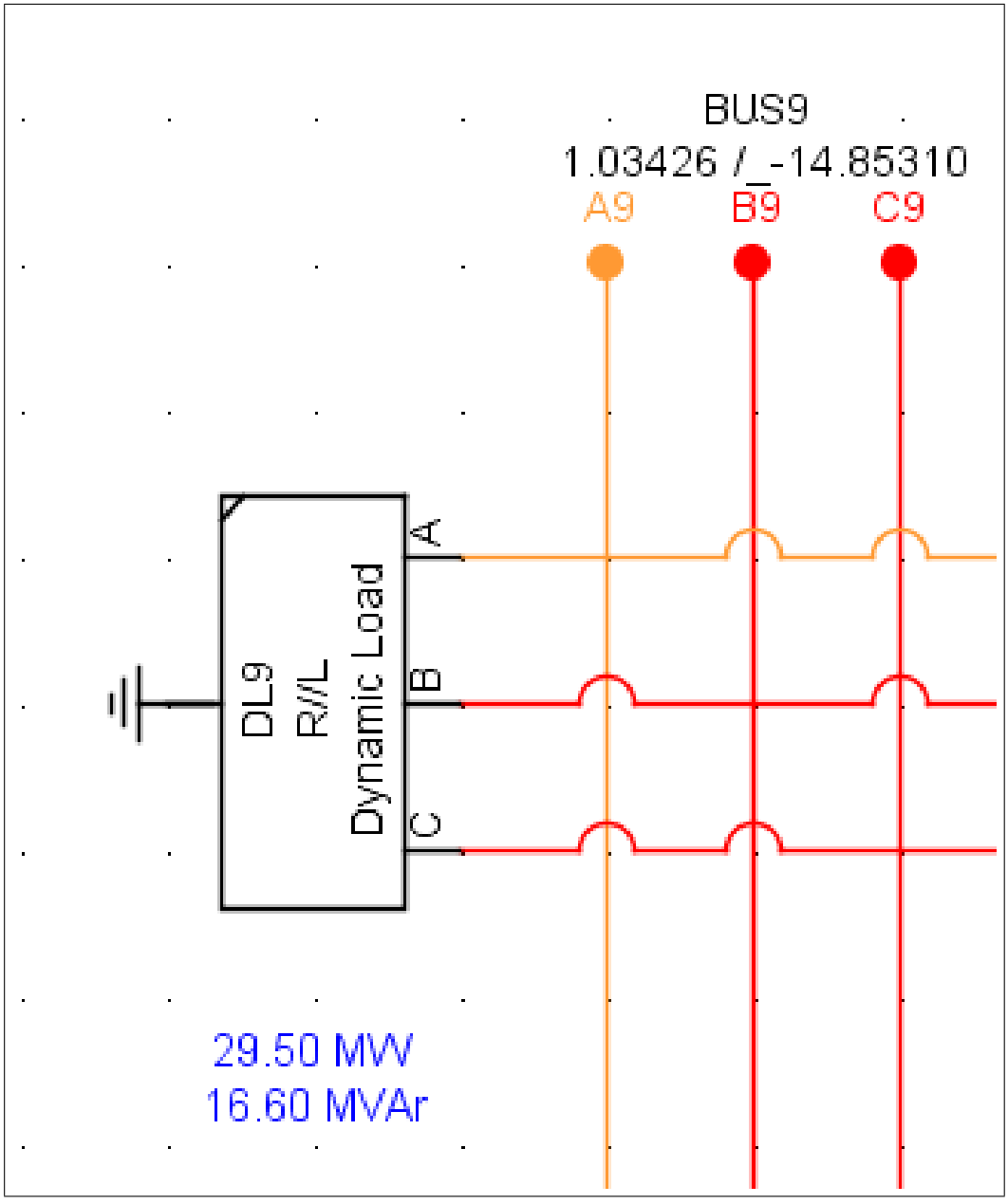

Figure 3.4: PQ load Block in RSCAD

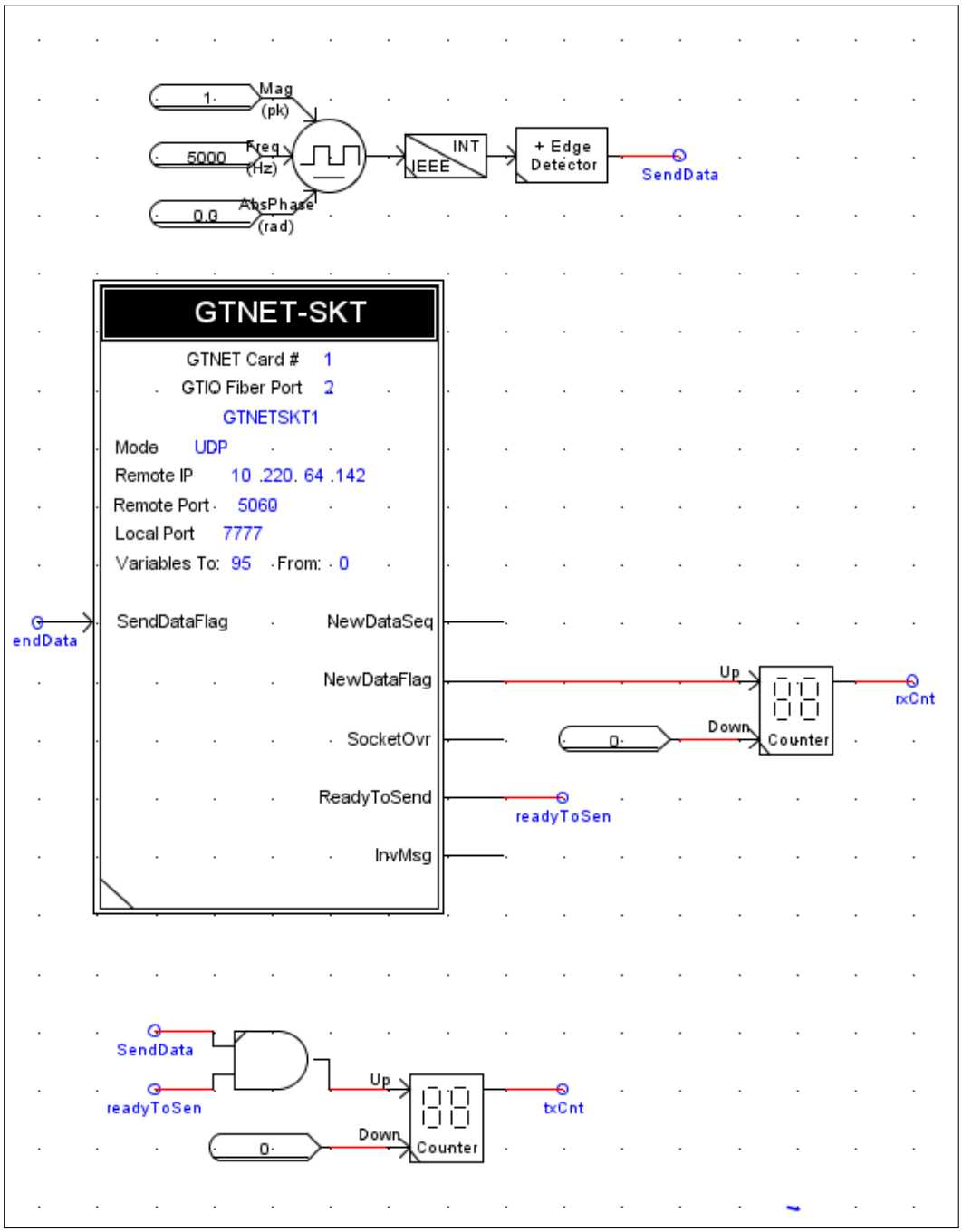

Figure 3.5: GTNET-SKT Block in RSCAD

Chapter 4

Experimental Results

4.1 Transfer of Data from RTDS to server

In Figure **??**, we observe real time data transfer from RTDS via GTNET-SKT to our server, which we built using a python script running on the Desktop. Here the measurements come in the order described in chapter 3 (Real injection then reactive . The value 4294967295 ($2^{32} - 1$) is the status number for the set of measurements. All bits are 1, signifying all measurements are available. In Figure **??**, we observe the availability status of all circuit breakers in the system. The first number is the case_number, and 4 represents 14 bus system. Circuit breakers status number is also 4294967295, signifying all circuit breakers are available. Circuit breaker measurement of 7 represents all 3 phases are ON (3 bits are 1).
The graph is the frequency response curve of the system, in rad/s. The system operates at $376.5/2\pi = 60Hz$ with very little fluctuation.

This shows that we have been able to extract all the available measurements, and identify which measurement belongs to which variable, without using many extra measurement variables.

4.2 Observability Analysis and State Estimation

After collecting data, we perform observability analysis. If the system is found to be observable, we perform state estimation. Here, for the above example, the system performs state estimation since all measurements are available. In fact, since all the measurements are available, and are exact

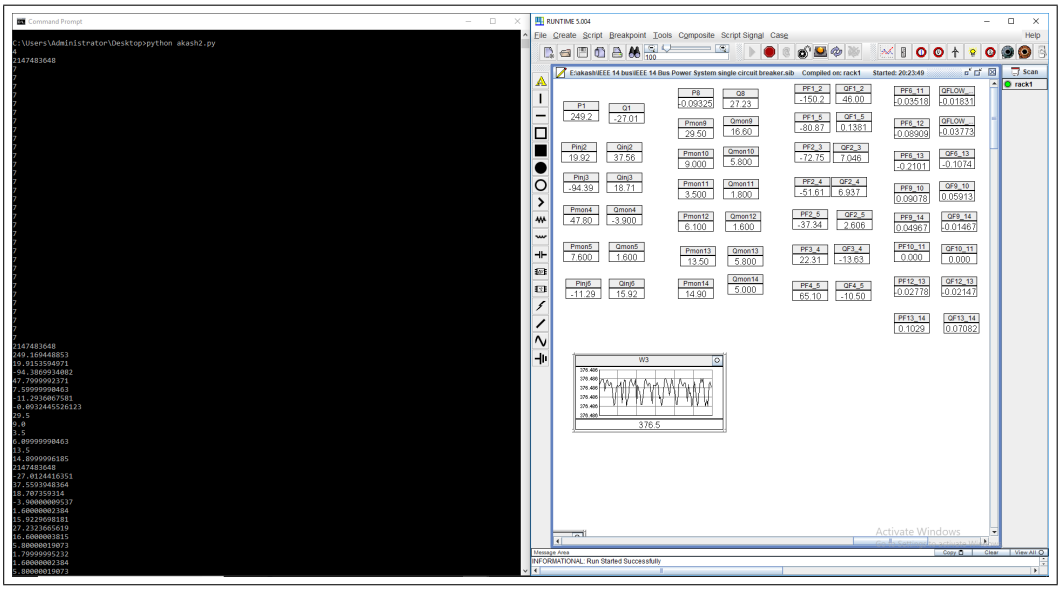

Figure 4.1: Transfer of real and reactive power injections and flows from RTDS (right) to server (left), Observing circuit breaker status

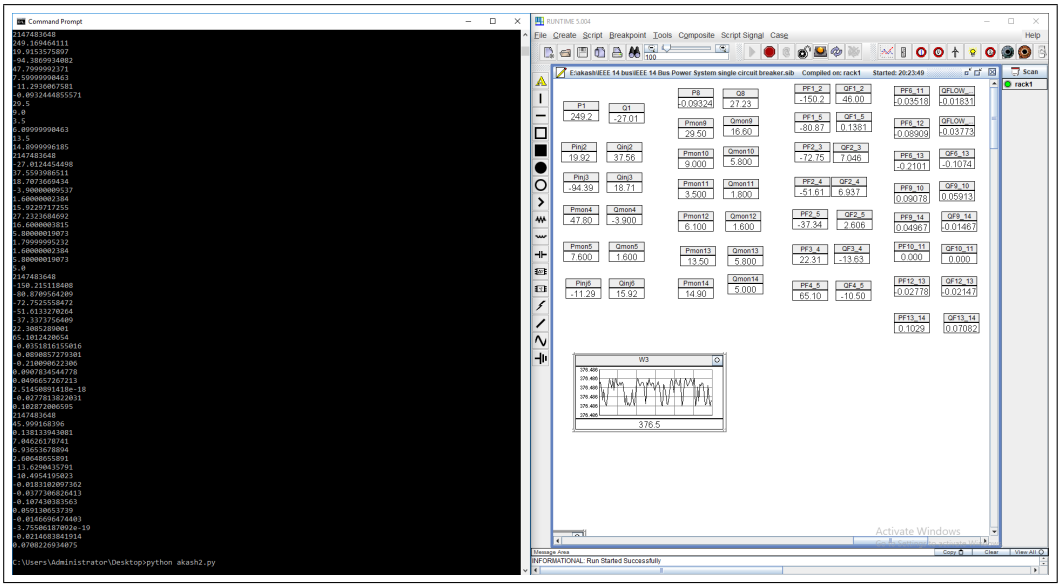

Figure 4.2: Transfer of real and reactive power injections and flows from RTDS (right) to server (left)

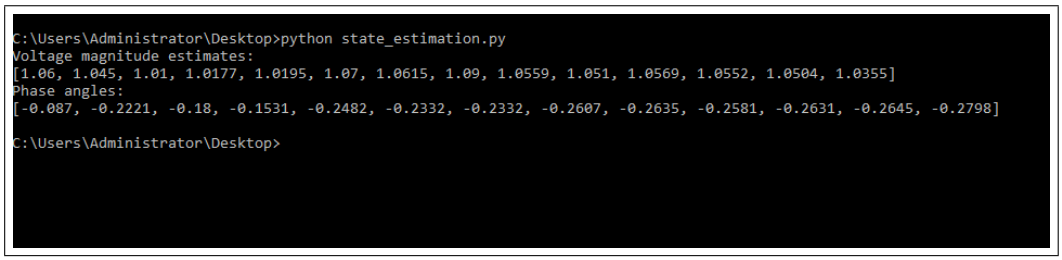

Figure 4.3: Values of Bus voltages and angles for IEEE 14 Bus system

measurements, the results obtained from state estimation are very close to the ones obtained from load flow solutions (See values at [**?**]).

To create an unobservable system, we modify the availability of line flow and bus injection datas, and try finding out the state estimates. We removed the real and reactive power flow measurements in line 6-11 and line 10-11, as well as real and reactive power injection at buses 6,10 and 11. That way, bus 11 is unobservable.

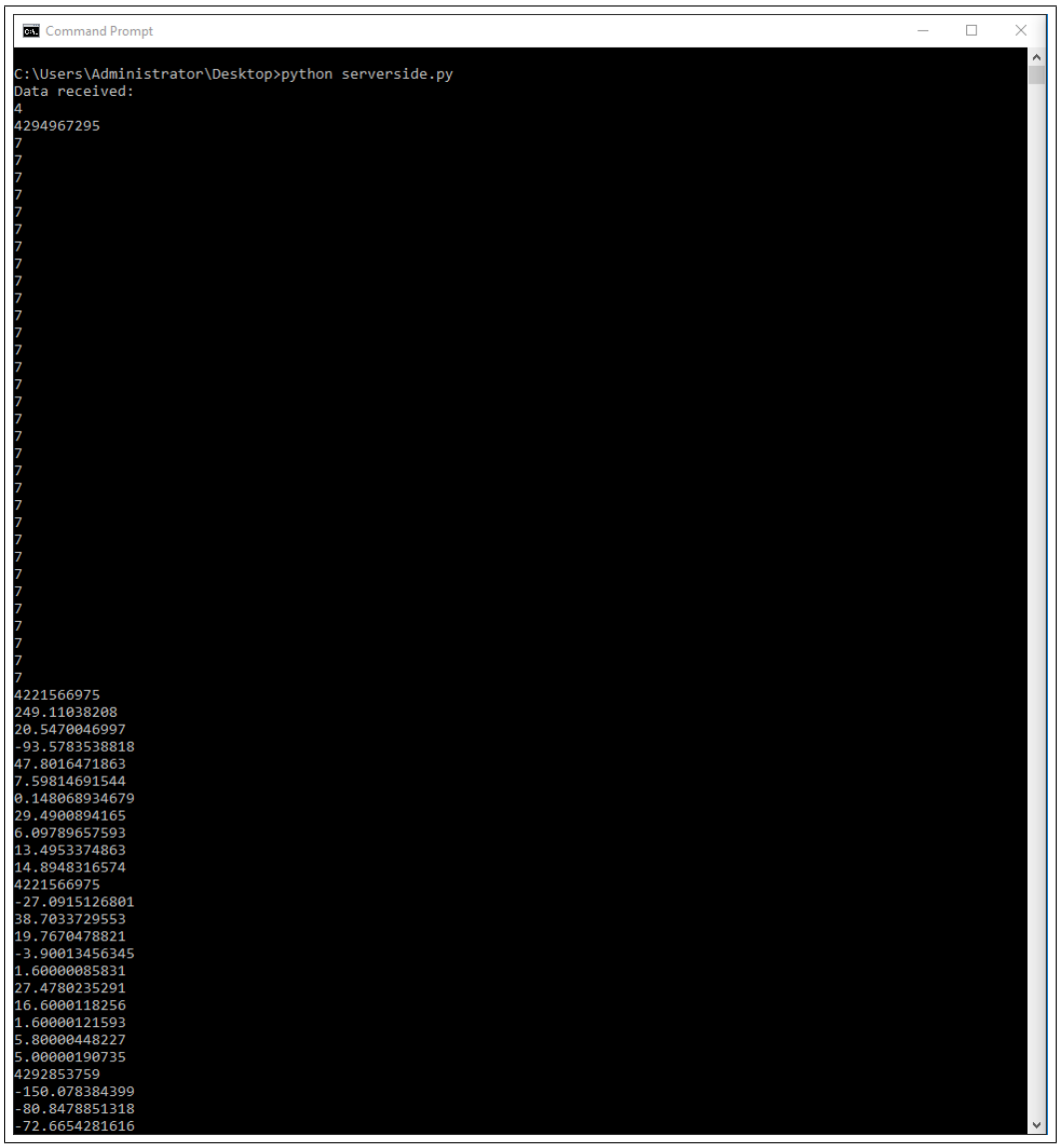

Figure 4.4: Removing measurements of flows at lines 6-11 and 10-11 and injections at 6,10,11. Part 1

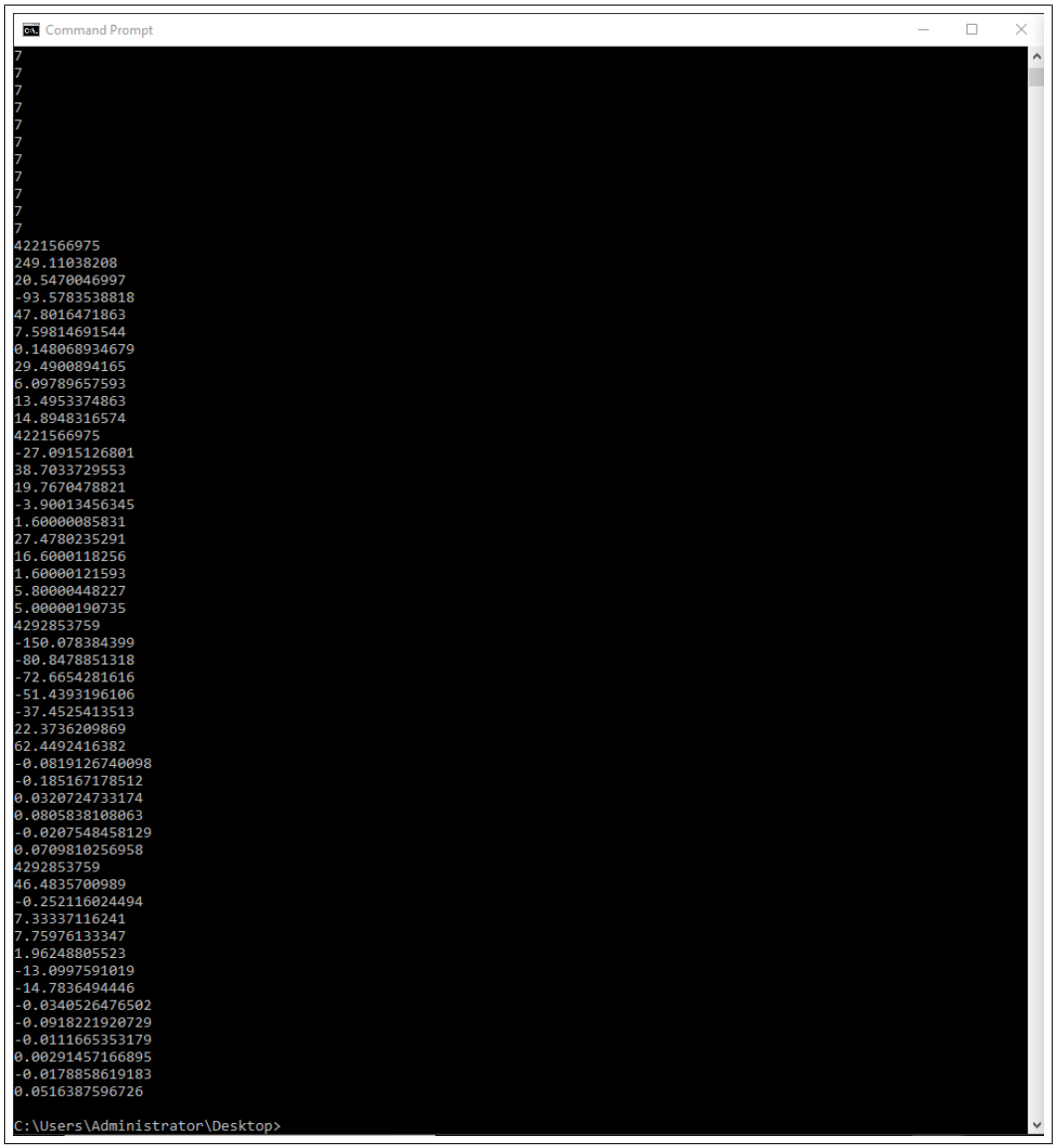

Figure 4.5: Removing measurements of flows at lines 6-11 and 10-11 and injections at 6,10,11. Part 2

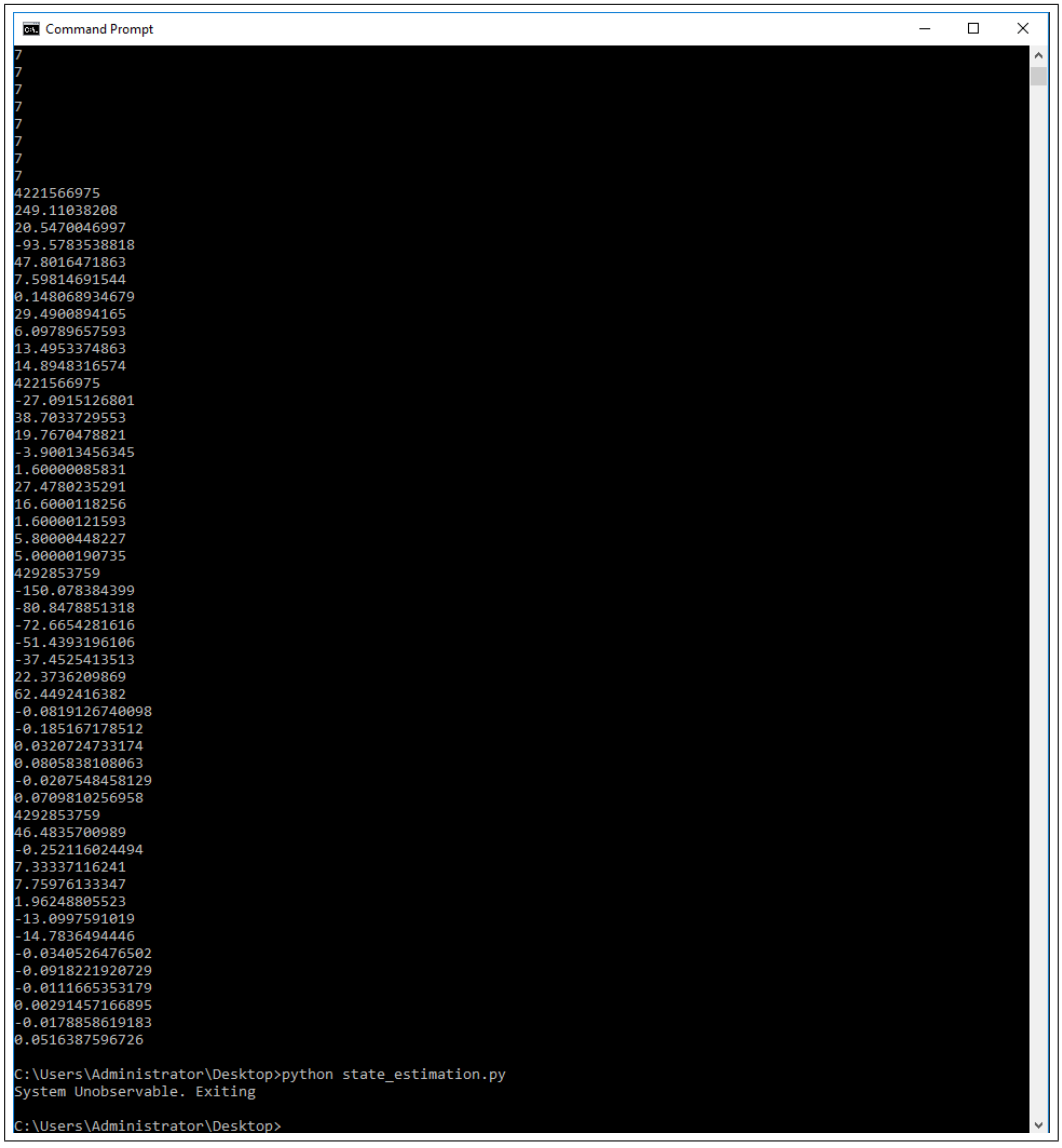

Figure 4.6: Running state estimation with less than neccessary measurements

Chapter 5

Conclusion

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.