ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - TE141599

STATE ESTIMASI PADA JARINGAN DISTRIBUSI 20 KV SURABAYA UTARA DENGAN METODE *EXTENDED KALMAN FILTER* MENGGUNAKAN BAHASA PEMROGRAMAN *JAVA*

Aisah Nur Laily
NRP 2213106073

Dosen Pembimbing
Prof. Ir. Ontoseno Penangsang, M.Sc., Ph.D.
Prof. Dr. Ir. Adi Soeprijanto,MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

STATE ESTIMATION AT 20 KV DISTRIBUTION NETWORK NORTH SURABAYA WITH EXTENDED KALMAN FILTER METHOD USING JAVA PROGRAMMING LANGUAGE

Aisah Nur Laily
NRP 2213106073

Supervisors
Prof. Ir. Ontoseno Penangsang, M.Sc., Ph.D.
Prof. Dr. Ir. Adi Soeprijanto, MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty Of Industry Technology
Sepuluh Nopember Institute Of Technology
Surabaya 2016

STATE ESTIMASI PADA JARINGAN DISTRIBUSI 20 KV SURABAYA UTARA DENGAN METODE *EXTENDED KALMAN FILTER* MENGGUNAKAN BAHASA PEMROGRAMAN *JAVA*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Tenaga
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D	Prof. Dr. Ir. Adi Soeprijanto, MT.
NIP. 194907151974121001	NIP. 196404051990021001

Surabaya
Januari, 2016

JURUSAN
TEKNIK ELEKTRO

STATE ESTIMASI PADA JARINGAN DISTRIBUSI 20 KV SURABAYA UTARA DENGAN METODE EXTENDED KALMAN FILTER MENGGUNAKAN BAHASA PEMROGRAMAN JAVA

Nama Mahasiswa : Aisah Nur Laily
NRP : 2213106073
Dosen Pembimbing I : Prof. Ir. Ontoseno Penangsang, M.sc, Ph.D
NIP : 19490715 1974121 001
Dosen Pembimbing II : Prof. Dr. Ir. Adi Soeprijanto, MT.
NIP : 19640405 1990021 001

ABSTRAK

Analisis aliran daya digunakan untuk mengetahui parameter aliran daya, khususnya parameter tegangan. Untuk mengetahui parameter-parameter yang ada diperlukan peralatan pengukuran yang ditempatkan pada tiap bus agar dapat dimonitoring secara real time. Permasalahan yang sering terjadi dalam monitoring sistem tenaga yaitu kesalahan pada transduser yang menyebabkan error pengukuran semakin besar. Analisis *state estimation* (SE) digunakan untuk mengatasi permasalahan tersebut. State estimasi berperan untuk monitoring nilai parameter dari keadaan suatu sistem jaringan dan mengurangi *gross error measurement* pada alat ukur. Pada tugas akhir ini akan dikaji state estimasi dengan metode *Extended Kalman Filter (EKF)*. Metode *EKF* ini dapat menghasilkan estimasi tegangan yang mendekati hasil analisis pada *software* ETAP, yaitu sebesar 0.99pu dengan *error* rata-rata terbesar 0.0071%. Metode ini diaplikasikan menggunakan bahasa pemrograman *JAVA* pada jaringan distribusi Surabaya Utara.

Kata kunci : *Extended Kalman Filter*, *error* pengukuran, *JAVA*

STATE ESTIMATION AT 20 KV DISTRIBUTION NETWORK NORTH SURABAYA WITH EXTENDED KALMAN FILTER METHOD USING JAVA PROGRAMMING LANGUAGE

Student Name	**: Aisah Nur Laily**
Id Number	**: 2213106073**
Supervisor I	**: Prof. Ir. Ontoseno Penangsang, M.sc, Ph.D**
Id Number	**: 19490715 1974121 001**
Supervisor II	**: Prof. Dr. Ir. Adi Soeprijanto, MT.**
Id Number	**: 19640405 1990021 001**

ABSTRACT

Power flow analysis is used to determine the parameters of the power flow, in particular voltage parameter. To determine the existing parameters necessary measuring equipment placed on each bus to be monitored in real time. Problems often occur in the power system monitoring is an error in the measurement transducer that causes greater error. Analysis of state estimation (SE) is used to overcome these problems. State estimation of a role for monitoring parameter values of the state of a network system and reduce the gross errors of measurement on the instrument. In this final project will be studied state estimation method Extended Kalman Filter (EKF). EKF method can produce voltages approaching the estimated results of the analysis on ETAP software, which amounted 0.99pu with the greatest error rate 0.0071%. This method was applied using the JAVA programming language in North Surabaya distribution network.

Keyword : *Extended Kalman Filter, measurement error, JAVA*

KATA PENGANTAR

Alhamdulillah puji syukur kehadirat Allah SWT, atas izin-Nya sehingga Tugas Akhir ini dapat terselesaikan. Tidak lupa penyusun menyadari bahwa ini semua juga atas bantuan dari banyak pihak. Oleh karena itu dengan penuh rasa hormat dan rendah hati, penyusun mengucapkan terima kasih kepada :

1. Allah SWT yang senantiasa memberikan rahmat, hidayah, dan petunjuk-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Kepada kedua orang tua dan keluarga yang telah memberikan dukungan dan doa.
3. Kepada Bapak Prof. Ir. Ontoseno Penangsang, M.Sc.,Ph.D dan Bapak Prof. Dr. Ir. Adi Soeprijanto, MT. atas bimbingan selama pengerjaan tugas akhir ini.
4. Seluruh dosen penguji atas arahan dan kritikan yang telah diberikan demi kesempurnaan Tugas Akhir ini.
5. Bapak Dr. Ardyono Priyadi, ST., MT. selaku Ketua Jurusan Teknik Elektro ITS.
6. Seluruh Dosen dan Karyawan di jurusan Teknik Elektro ITS atas fasilitas dan waktu yang diberikan.
7. Teman-teman seperjuangan Niken, Sherly, Santi, Nadya, Uci yang telah berjuang bersama-sama untuk mendapatkan gelar Sarjana.
8. Mas Wira Candra, Mas I Putu Widya Wiranata, Mbak Indri Suryawati yang telah banyak membantu dalam Tugas Akhir ini.
9. Semua teman-teman mahasiswa Lintas Jalur Teknik Elektro ITS angkatan 2013 semester genap bidang studi Teknik Sistem Tenaga.

Akhir kata, segala kritik dan saran sangat saya harapkan untuk pengembangan selanjutnya.

Surabaya, 15 Desember 2015

Penyusun

DAFTAR ISI

HALAMAN

DAFTAR GAMBAR

HALAMAN

DAFTAR TABEL

HALAMAN

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kota Surabaya sebagai pusat pemerintahan provinsi Jawa Timur merupakan kota kedua di Indonesia dengan jumlah penduduk cukup besar, pertumbuhan industri yang pesat dan luas wilayah yang cukup besar. Kota tersebut membutuhkan distribusi tenaga listrik yang cukup besar dan memadai agar kebutuhan pelanggan akan tenaga listrik dapat terpenuhi. Dalam memenuhi kebutuhan tenaga listrik, analisis aliran daya diperlukan untuk mengetahui daya yang tersambung dan daya terpakai oleh para pelanggan.

Analisis aliran daya digunakan untuk mengetahui parameter aliran daya, khususnya parameter tegangan.[1] Untuk mengetahui parameter-parameter yang ada diperlukan peralatan pengukuran yang ditempatkan pada tiap bus. Peralatan pengukuran ini dipasang pada tiap bus agar dapat dimonitoring secara real time. Namun, dalam monitoring sistem tenaga listrik seringkali terdapat beberapa permasalahan. Permasalahan yang sering terjadi dalam monitoring sistem tenaga yaitu kesalahan pada transduser yang menyebabkan *error* pengukuran semakin besar. Di samping itu, transduser juga memiliki *gross error* alat ukur yang mengakibatkan hasil pengukuran yang tidak akurat. Oleh karena itu, State Estimasi diperlukan untuk mengestimasi nilai parameter dari keadaan suatu sistem jaringan, mengetahui nilai yang paling mendekati dengan nilai real pengukuran sehingga *output* yang didapat lebih akurat.

Pada tugas akhir ini dikaji sistem non-linier dengan *EKF* (*Extended Kalman Filter)* yang merupakan pengembangan dari metode Kalman Filter[2], yang dapat mengidentifikasi sistem non-linier. Metode *EKF* diharapkan dapat menghasilkan output yang lebih akurat dan dapat mengetahui *gross error measurement* pada alat ukur.

Bahasa pemrograman yang digunakan pada tugas akhir ini adalah bahasa pemrograman *JAVA*. *JAVA* merupakan bahasa pemrograman komputer yang berbasis *OOP (Object Oriented Programming)* yaitu suatu pendekatan yang memungkinkan suatu kode yang digunakan untuk menyusun program menjadi lebih mudah untuk digunakan kembali, lebih handal, dan lebih mudah dipahami.[3]

1.2 Permasalahan

Permasalahan yang akan dibahas dalam penyelesaian Tugas Akhir ini adalah :

1. Cara menerapkan metode *EKF* pada state estimasi.
2. Cara menentukan parameter state estimasi pada jaringan distribusi Surabaya Utara.
3. Cara merancang & mendesain program *SE* dengan metode *EKF* menggunakan bahasa pemrograman *JAVA*.
4. Cara menerapkan state estimasi dengan metode *EKF* pada jaringan distribusi Surabaya utara.

1.3 Tujuan

Adapun tujuan dari pembuatan Tugas Akhir ini adalah :

1. Mengetahui prinsip *state estimation.*
2. Mengetahui prinsip metode *Extended Kalman Filter(EKF.)*
3. Mengetahui nilai tegangan pada suatu bus dan *error* pengukuran dengan metode *EKF*.
4. Mengetahui bagaimana membangun program *state estimation* dengan bahasa pemrograman *JAVA*.

1.4 Metode Penelitian

Penelitian Tugas Akhir ini dibagi menjadi beberapa tahap :

1. Studi literatur, dilakukan dengan membaca referensi-referensi berkaitan dengan metode EKF dari buku, paper, dan internet.
2. Pengumpulan data beban dan data impedansi saluran pada sistem distribusi 20KV di Surabaya utara.
3. Pemodelan program dilakukan dengan membuat algoritma *Extended Kalman Filter* untuk *state estimation.*
4. Perancangan perangkat lunak untuk memperoleh hasil state estimasi menggunakan bahasa pemrograman *JAVA*.
5. Simulasi dan analisis, melakukan pengujian program kemudian hasilnya dibandingkan dengan *software* ETAP.
6. Penyusunan laporan, melaporkan hasil penelitian tersebut dalam bentuk laporan tugas akhir. Laporan ini berisi metode dan kesimpulan dari penelitian.

1.5 Sistematika Penulisan

Laporan Tugas Akhir ini disusun dalam suatu sistematika sebagai berikut :

Bab 1: Pendahuluan yang membahas mengenai latar belakang, permasalahan, tujuan penelitian, metode penelitian, sistematika penulisan dan relevansi dari Tugas Akhir.

Bab 2: Dasar teori yang merupakan penjelasan teori mengenai sistem distribusi secara umum, konsep state estimasi, metode *Extended Kalman Filter*, serta bahasa pemrograman *JAVA*.

Bab 3: Metodologi yang menjelaskan tentang alur proses pengerjaan tugas akhir beserta data jaringan distribusi 20 KV Surabaya Utara.

Bab 4: Hasil simulasi program State Estimasi *EKF* pada data beban dan data saluran jaringan distribusi 20 KV Surabaya Utara dengan bahasa pemrograman *JAVA*.

Bab 5: Kesimpulan Tugas Akhir yang mengemukakan hasil pembahasan dan saran-saran yang sehubungan dengan pokok-pokok bahasan tugas akhir.

1.6 Relevansi

Hasil yang diperoleh dari Tugas Akhir ini diharapkan dapat memberikan manfaat berikut,

1. Memberikan kontribusi terhadap perkembangan sistem kelistrikan, khususnya pada permasalahan yang berkaitan dengan pengukuran dan monitoring sistem tenaga listrik.
2. Dapat meningkatkan penguasaan Ilmu Pengetahuan dan Teknologi (IPTEK) di bidang Sistem Tenaga bagi pengusul Tugas Akhir.
3. Dapat menjadi referensi bagi mahasiswa lain yang hendak mengambil topik yang serupa sebagai Tugas Akhir.
4. Memberikan kontribusi terhadap instansi/perusahaan kelistrikan, dengan digunakannya *Java Programming* yang dapat digunakan sebagai program aplikasi pada komputer.

Halaman ini sengaja dikosongkan

BAB 2
STATE ESTIMASI
PADA JARINGAN DISTRIBUSI

2.1 Sistem Tenaga Listrik

Energi listrik merupakan kebutuhan yang sangat penting bagi setiap orang. Energi listrik dapat dimanfaatkan untuk berbagai keperluan, seperti penerangan jalan, keperluan rumah tangga, sampai di industri/ pabrik-pabrik dan berbagai bidang lain. Dalam penyediaan energi listrik kita perlu mengetahui sistem tenaga listrik. Sistem tenaga listrik dibagi menjadi 3 bagian utama, yaitu pembangkitan, transmisi dan distribusi yang terintegrasi untuk memenuhi kebutuhan energi listrik untuk semua konsumen energi listrik.

Pada pembangkitan tenaga listrik sumber energi alam diubah menjadi energi mekanis berupa kecepatan atau putaran, kemudian dari energi mekanis diubah menjadi energi listrik oleh generator. Untuk menyalurkan energi listrik dari pusat pembangkit ke pusat beban (Gardu Induk) dibutuhkan saluran transmisi. Daya yang akan dikirim atau disalurkan dari pusat pembangkit ke pusat beban ditransmisikan dengan tegangan tinggi maupun tegangan ekstra tinggi.

Pada pusat beban, seperti GI (Gardu Induk) daya yang disalurkan melalui saluran transmisi tegangan tinggi diubah menjadi tegangan menengah (tegangan distribusi primer), selanjutnya diturunkan lagi menjadi tegangan rendah yang dipakai oleh konsumen energi listrik. Berikut deskripsi penyaluran energi listrik dari sumber ke beban :

Gambar 2.1 Diagram Garis Sistem Tenaga Listrik

2.2 Jaringan Distribusi

2.2.1 Sistem Pendistribusian Tenaga Listrik

Sistem distribusi pada jaringan distribusi tenaga listrik perlu diketahui agar kita mengetahui proses penyaluran tenaga listrik dari pembangkit sampai ke pengguna tenaga listrik (beban). Sistem pendistribusian tenaga listrik dibagi menjadi dua, yaitu :

a. Sistem pendistribusian langsung
   Merupakan sistem distribusi yang tidak melalui saluran transmisi. Sistem penyalurannya dari pusat pembangkit langsung ke pusat beban. Biasanya terletak dekat dengan daerah pelayanan beban.
b. Sistem pendistribusian tidak langsung
   Merupakan sistem distribusi yang melalui saluran transmisi. Biasanya terletak dekat dengan pusat-pusat beban, sehingga memerlukan jaringan transmisi.

2.2.2 Struktur Jaringan Distribusi

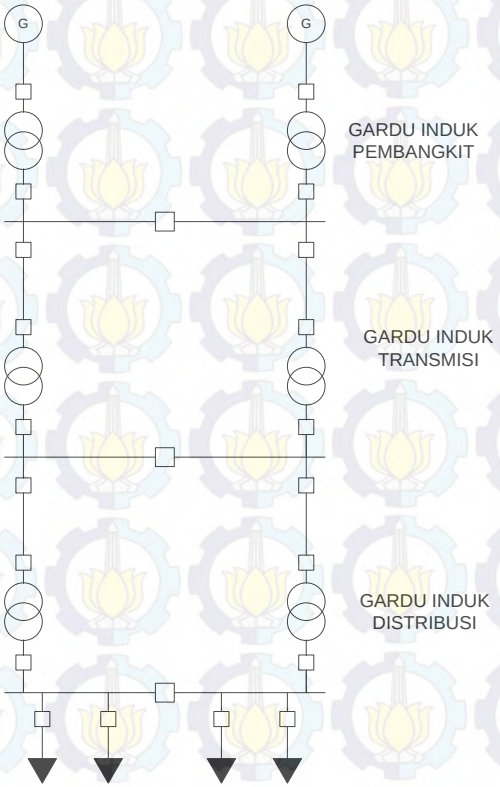
Struktur jaringan distribusi dibagi menjadi beberapa bagian, yaitu :

a. Gardu Induk atau Pusat Pembangkit Tenaga Listrik
   Jika sistem pendistribusian tenaga listrik dilakukan secara langsung, maka pusat pembangkit tenaga listrik menjadi bagian pertama dari sistem distribusi tenaga listrik. Sebagai contohnya adalah PLTD (Pembangkit Listrik Tenaga Diesel) yang terletak di pinggiran kota. Untuk menyalurkannya ke pusat-pusat beban melalui jaringan distribusi primer dan jaringan distribusi sekunder.
   Jika sistem pendistribusian tenaga listriknya secara tidak langsung, maka bagian pertama dari sistem pendistribusian tenaga listrik adalah GI (Gardu Induk). Pada Gardu Induk Tegangan Tinggi diturunkan melalui saluran transmisi, kemudian akan disalurkan melalui jaringan distribusi primer.
b. Jaringan distribusi primer
   Untuk sistem distribusi langsung, jaringan distribusi primer merupakan awal penyaluran tenaga listrik dari Pusat Pembangkit Tenaga Listrik ke konsumen.
   Untuk pendistribusian tidak langsung, jaringan distribusi primer menjadi tahap penyaluran berikutnya setelah GI (Gardu Induk). Jaringan distribusi primer memiliki tegangan sistem sebesar 20 kV. Untuk wilayah kota tegangan di atas 30 kV

tidak diperbolehkan, karena pada tegangan 30 kV akan terjadi korona yang dapat mengganggu frekuensi radio, televisi, dan telekomunikasi.

c. Jaringan distribusi sekunder

Berfungsi merubah tegangan dari jaringan distribusi primer menjadi Tegangan Rendah (TR) yang siap digunakan oleh pelanggan. Bagian ini disebut dengan jaringan distribusi sekunder. Kapasitas trafo yang digunakan pada bagian ini tergantung pada jumlah beban yang akan dilayani dan luas daerah pelayanan beban. Dapat berupa trafo satu fasa maupun trafo tiga fasa.

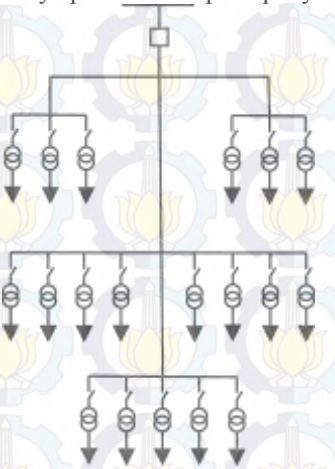


Gambar 2.2 Struktur Jaringan Distribusi

2.2.3 Bentuk Konfigurasi Jaringan Distribusi

a. Sistem distribusi radial

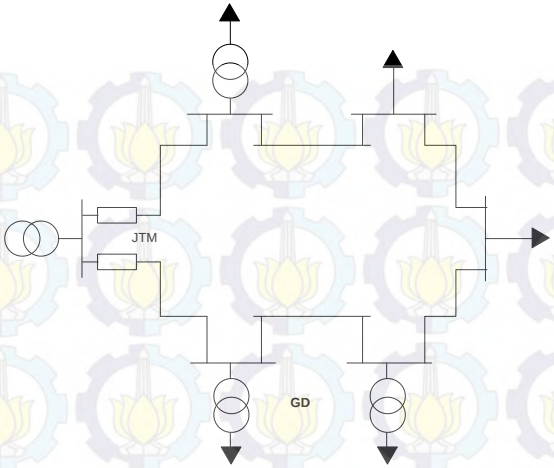
Merupakan jaringan sistem distribusi yang salurannya ditarik secara radial dari satu titik sumber kemudian bercabang menuju ke beban-beban. Tipe radial yang paling banyak diterapkan pada saluran distribusi karena konfigurasinya sederhana dan biaya investasinya murah. Pada jaringan ini arus yang paling besar adalah arus yang paling dekat dengan Gardu Induk. Namun, bentuk jaringan ini kurang handal dalam penyaluran energi listrik karena bila terjadi gangguan pada penyulang akan menyebabkan terjadinya pemadaman pada penyulang tersebut.



Gambar 2.3 Sistem distribusi radial

b. Sistem distribusi Ring / Loop

Merupakan jaringan distribusi yang melingkar (*Ring*), pada saat terjadi gangguan masih bisa disuplai dari bus yang lain. Seperti terlihat pada gambar 2.4, konfigurasi jaringan ini merupakan jaringan distribusi primer, gabungan dari dua tipe jaringan radial dimana ujung kedua jaringan dipasang PMT. Pada keadaan normal tipe ini bekerja secara radial dan pada saat terjadi gangguan PMT dapat dioperasikan sehingga gangguan dapat dilokalisir. Konfigurasi ring/loop ini lebih handal dalam penyaluran tenaga listrik dibandingkan tipe radial, namun biaya investasinya lebih mahal.



Gambar 2.4 Sistem distribusi ring

2.3 Analisis Aliran Daya

Semakin bertambahnya konsumen akan kebutuhan tenaga listrik, maka akan selalu terjadi perubahan beban, perubahan unit-unit pembangkit, dan perubahan saluran. Dengan analisis aliran daya dapat digunakan untuk perencanaan dan pengembangan sistem tenaga listrik di masa depan, serta menentukan operasi terbaik pada jaringan yang sudah ada.

Analisis aliran daya penting untuk diketahui karena dengan analisis aliran daya kita dapat mengetahui informasi tentang beban pada saluran, tegangan di setiap bus, menentukan besarnya daya nyata (real power) dan daya reaktif (reactive power) pada saluran saat sistem beroperasi normal, mengetahui arus yang mengalir pada saluran, dan untuk mengetahui besar rugi-rugi daya pada saluran.

Untuk mendapatkan nilai aliran daya pada masing-masing bus diperlukan data saluran dan data bus. Data saluran terdiri dari data nilai resistansi dan reaktansi penghantar beserta dengan panjangnya. Nilai ini digunakan sebagai pertimbangan untuk menghitung tegangan jatuh yang melewati saluran tersebut. Data bus terdiri dari nilai daya aktif dan reaktif yang mengalir di tiap bus. Semua data saluran dan data bus tersebut diproses untuk mendapatkan nilai tegangan di tiap bus sesuai dengan nilai toleransi yang diinginkan. Proses iterasi analisa aliran daya

yang umum digunakan ada tiga metode yaitu Newton Raphson, Gauss Seidel dan Fast Decoupled[1].

Hasil perhitungan analisis aliran daya dapat digunakan untuk menyelesaikan berbagai masalah yang berhubungan dengan jaringan sistem tenaga, yaitu meliputi hal-hal yang berhubungan dengan operasi jaringan, yaitu :[4]

a. Pengaturan tegangan (voltage regulation), perbaikan factor daya (power factor) jaringan, kapasitas kawat penghantar, termasuk rugi-rugi daya.
b. Perluasan atau pengembangan jaringan, yaitu menentukan lokasi yang tepat untuk penambahan bus beban baru dan unit pembangkit atau gardu induk baru.
c. Perencanaan jaringan, yaitu kondisi jaringan yang diinginkan pada masa mendatang untuk melayani pertumbuhan beban karena kenaikan terhadap kebutuhan tenaga listrik.

2.4 Monitoring Sistem Tenaga Listrik

Suatu sistem tenaga listrik yang kompleks memiliki puluhan dan bahkan sampai ratusan titik bus. Untuk menjaga agar keadaan pada tiap bus dalam kondisi aman, harus dilakukan monitoring agar dapat diketahui kondisi pada tiap bus. Sehingga dapat segera ditangani apabila sistem mengalami *overloading* yang dapat menyebabkan *blackouts*.

Sistem monitoring yang umum digunakan pada sistem tenaga listrik ialah *Supervisory Control And Data Acquisition* (SCADA). Konsep dari state estimasi adalah menggunakan data SCADA untuk menghitung magnitude tegangan dan sudut fasa pada masing-masing bus. Data SCADA ini berupa data pengukuran $PQ_{injection}$, PQ_{flow}, dan magnitude tegangan pada masing-masing bus.[11]

2.4.1 Supervisory Control And Data Acquisition (SCADA)

SCADA adalah singkatan dari Supervisory Control And Data Acquisition merupakan suatu teknologi yang menggabungkan fungsi pengawasan, pengendalian, dan pengambilan data jarak jauh. Suatu sistem SCADA bertujuan untuk mengumpulkan informasi atau data dari lapangan kemudian mengirimkan data tersebut ke sebuah komputer pusat yang akan mengatur dan mengontrol data-data tersebut [11].

Dalam pengoperasiannya, SCADA terbagi atas beberapa komponen penyusun utama, yaitu [11] :

1. Instrumentasi, dapat berupa sensor atau relay kontrol yang terhubung langsung ke peralatan. Instrumentasi ini akan menghasilkan data yang akan dimonitor oleh Remote Telemetry Units (RTU).
2. Remote Station, merupakan sebuah unit yang dilengkapi dengan sistem mandiri seperti sebuah komputer, yang ditempatkan pada lokasi dan tempat-tempat tertentu di lapangan.
3. Jaringan komunikasi (Communications Network), merupakan peralatan yang berfungsi untuk mentransfer data dari Remote Station ke Master Terminal Unit. Peralatan transfer ini dapat berupa jaringan kabel, jaringan radio ataupun jaringan telpon.
4. Unit Master SCADA (Master Terminal Unit – MTU), merupakan komputer yang digunakan sebagai pengolah data pusat dari sistem SCADA. Pengolahan data yaitu berupa monitor dan kontrol. Unit master ini menyediakan HMI (*Human Machine Interface*) bagi pengguna dan secara otomatis mengatur sistem sesuai dengan masukan-masukan (dari sensor) yang diterima.

Pada dasarnya suatu sistem SCADA mempunyai 4 fungsi utama yaitu sebagai berikut [11] :

1. *Telemetering* (TM)
   Telemetering merupakan suatu proses untuk mendapatkan informasi atau data. Data yang didapat merupakan hasil pengukuran dari alat ukur yang dipasang pada suatu peralatan. Misalnya seperti pengukuran tegangan, arus, daya dan lain-lain.
2. *Telesignal* (TS)
   Telesignal adalah suatu proses untuk mendapatkan informasi suatu peralatan tertentu dengan mendeteksi nilai suatu sensor.
3. *Telecontrol* (TC)
   Telecontrol adalah suatu proses kontrol atau kendali terhadap suatu peralatan dengan secara *remote* atau jarak jauh.
4. *Data Communication*
   Data communication adalah suatu proses pengiriman data dari suatu sensor ke komputer ataupun dari suatu komputer ke komputer lain. Kebanyakan sinyal yang dihasilkan dari suatu sensor dan relai kontrol tidak dapat langsung diterjemahkan oleh protokol komunikasi. Oleh karena itu dibutuhkan

Remote Terminal Unit **(RTU)** yang berfungsi menjembatani antara sensor dan jaringan SCADA.

2.5 State Estimasi

Metode State estimasi merupakan sebuah proses untuk menentukan sebuah variabel yang tidak diketahui berdasarkan pengukuran dari sistem. Di desain untuk menghasilkan estimasi atau perkiraan terbaik yang mendekati dengan kondisi real sistem.

Kriteria yang paling sering dan familiar untuk state estimasi yaitu meminimalkan jumlah kuadrat selisih antara estimasi pengukuran dengan pengukuran real (least-squares). Ide dari estimasi least-squares diperkenalkan dan digunakan sejak abad-19. Kemudian pengembangan aplikasinya pada abad 20 di bidang *aerospace* untuk menentukan estimasi lintasan dan posisi pesawat terbang, rudal, dan kendaraan ruang angkasa.[5]

Pada sistem tenaga variabel state estimasi berupa magnitude tegangan dan sudut fasa. Data yang diperlukan untuk estimasi variabel tersebut antara lain data bus dan data saluran. Untuk data bus berupa data daya aktif dan daya reaktif yang mengalir pada bus, sedangkan data saluran berupa data resistansi dan reaktansi antar saluran bus. Dari data-data tersebut diproses dengan metode state estimasi untuk mendapatkan nilai variabel tegangan.

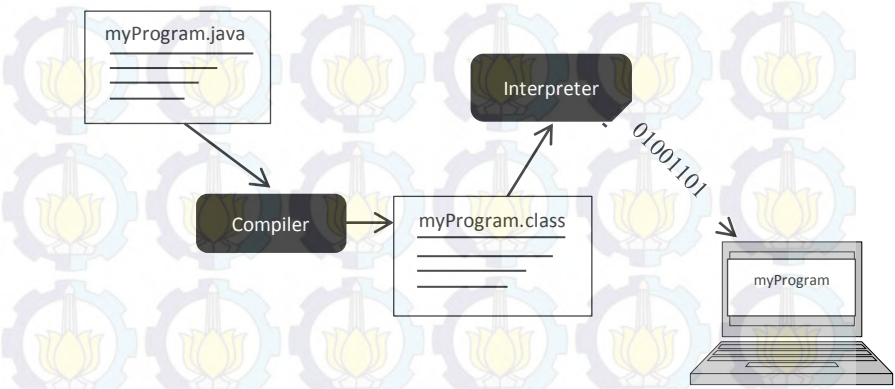
Dengan metode state estimasi kita dapat memperbaiki kesalahan atau error pengukuran yang terjadi agar diperoleh nilai variabel tegangan yang lebih akurat.[5] State estimasi juga dapat membantu operator sistem tenaga dalam mengidentifikasi apabila terjadi kesalahan dalam monitoring aliran daya.

2.6 Bahasa Pemrograman *JAVA*

JAVA dikembangkan oleh Sun Microsystems pada Agustus 1991 dengan nama semula **Oak**. Di buat pertama kali oleh James Gosling. Nama Oak dianggap kurang komersial, kemudian namanya diubah menjadi *JAVA* pada Januari 1995. *JAVA* merupakan bahasa pemrograman komputer yang berbasis *OOP (Object Oriented Programming),* terdiri dari kelas-kelas yang membentuk suatu objek yang dapat digunakan untuk menyusun program menjadi lebih mudah. [7].

JAVA dapat dijalankan pada semua *operating system (OS)*, karena *JAVA* memiliki cara kerja yang berbeda dengan bahasa

pemrograman lain. Gambar 2.6 mendeskripsikan tentang cara kerja *JAVA* :



Gambar 2.5 cara kerja *JAVA*

Kode program ditulis kemudian disimpan dengan nama file yang sama persis dengan nama *class*nya dengan ekstensi(.java) misalnya "coding.java", dari file ini kemudian di compile dengan compiler java(javac) sehingga menghasilkan file dengan ekstensi (.class),misalkan file coding.java tadi setelah di compile akan muncul file baru dengan nama "coding.class". file coding.class ini akan dijalankan oleh *JVM (JAVA Virtual Machine).*

JVM merupakan perangkat lunak yang dikembangkan secara khusus agar tidak tergantung dengan perangkat keras serta sistem operasi tertentu.

Penulisan kode program *JAVA* seperti berikut :

```
Class HelloWorldApp{
        Public static void main(String[] args){
                System.out.println("Hello
World");
                }
}
```

Gambar 2.6 penulisan kode program *JAVA*

Karakteristik *JAVA* antara lain :

- Sederhana (simple)
- Berorientasi objek (Object Oriented)
- Terdistribusi (Distributed)
- Aman
- Portable

Kelebihan bahasa pemrograman *JAVA* [7]:

- Multiplatform

Kelebihan utama dari *JAVA* ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program *JAVA* dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan pada beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis *JAVA* dikerjakan pada *operating system* Linux tetapi dijalankan dengan baik di *Microsoft Windows*. Penyebabnya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs *JAVA*) untuk menginterpretasikan *bytecode* tersebut.

- OOP (Object Oriented Programming)

Pemrograman berorientasi objek.

- *Library* yang lengkap

JAVA terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman *JAVA*) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas *JAVA* yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

- Bergaya C++

memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke *JAVA*. Saat ini pengguna *JAVA* sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke *JAVA*.

Kelebihan lain dari bahasa pemrograman JAVA ini adalah hasil *build programnya* dapat dijadikan *.exe*, sehingga dapat dijalankan menjadi suatu aplikasi yang dapat di-install pada komputer *stand alone*

jika dibandingkan dengan bahasa pemrograman lain yang hanya dapat digunakan untuk simulasi dan membantu perhitungan saja, tidak dapat dijadikan suatu program aplikasi.

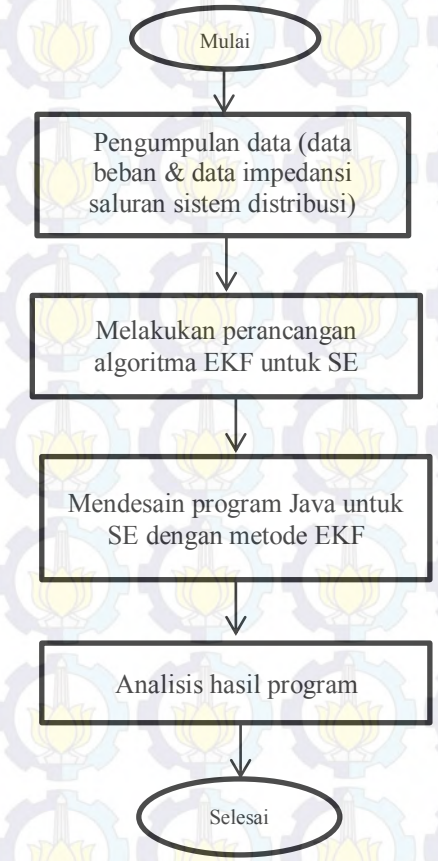
Bahasa pemrograman *JAVA* saat ini sudah dikembangkan di berbagai bidang. Sebagai contohnya, *JAVA* dapat digunakan untuk aplikasi jual-beli online di internet, untuk aplikasi pelelangan saham, aplikasi game pada ponsel, aplikasi *diagnose* penyakit di bidang kesehatan, aplikasi peramalan tagihan listrik, dan lain-lain. Pada tugas akhir ini, *JAVA* digunakan untuk simulasi state estimasi. Simulasi state estimasi ini berupa tampilan *Single Line Diagram (SLD)* dan data hasil state estimasi tegangan dan sudut fasa. Dengan tampilan ini akan lebih memudahkan dalam monitoring sistem tenaga listrik.

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI

3.1 Metodologi Pengerjaan Tugas Akhir

Pada bab 3 ini akan dijelaskan tentang metode pengerjaan tugas akhir. Dijelaskan dengan *flowchart*/diagram alir berikut :



Gambar 3.1 diagram alir metodologi pengerjaan tugas akhir

1. Pengumpulan data

   Melakukan pengambilan data dari jaringan distribusi Surabaya Utara, berupa data beban dan data impedansi saluran yang akan digunakan untuk state estimasi.

2. Melakukan perancangan algoritma

   Membuat diagram alir program sesuai dengan logika program EKF.

3. Mendesain dan menulis kode program *JAVA*.

   Mengimplementasikan diagram alir program ke dalam bahasa pemrograman *JAVA* untuk state estimasi menggunakan data penyulang Kaliasin, penyulang Basuki rahmat, penyulang Ometraco, penyulang Tegalsari dan Tunjungan.

4. Analisis hasil program

   Setelah memperoleh beberapa hasil dari simulasi yang telah dilakukan, dianalisis hasil keakuratan hasil simulasinya, untuk mengetahui apakah metode yang digunakan dapat menghasilkan data yang akurat.

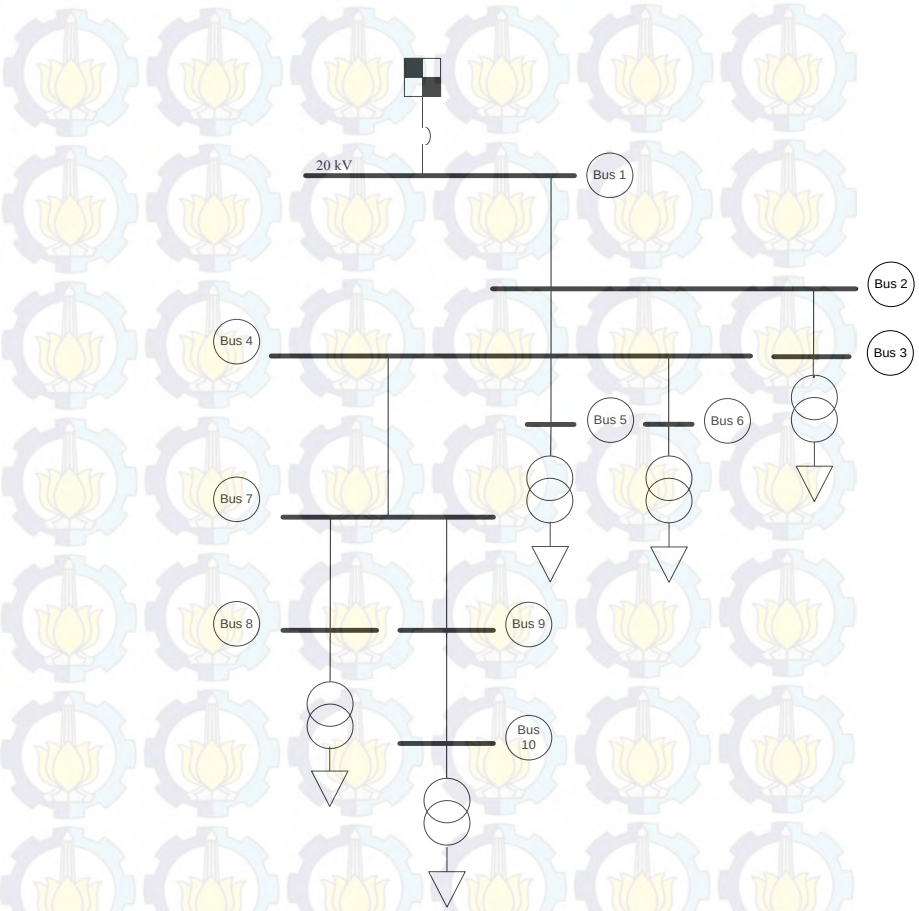
3.1.1 Sistem Distribusi 20KV Surabaya Utara

Pada Tugas Akhir ini, penulis melakukan pengambilan data yang berupa data pembangkitan, data saluran, data pembebanan pada sistem distribusi 20 KV wilayah Surabaya Utara. Data saluran dan data pembebanan inilah yang akan digunakan untuk input program state estimasi pada bahasa pemrograman *JAVA Eclipse* yang berupa resistansi (R), reaktansi (X), daya aktif (P) , dan daya reaktif (Q).

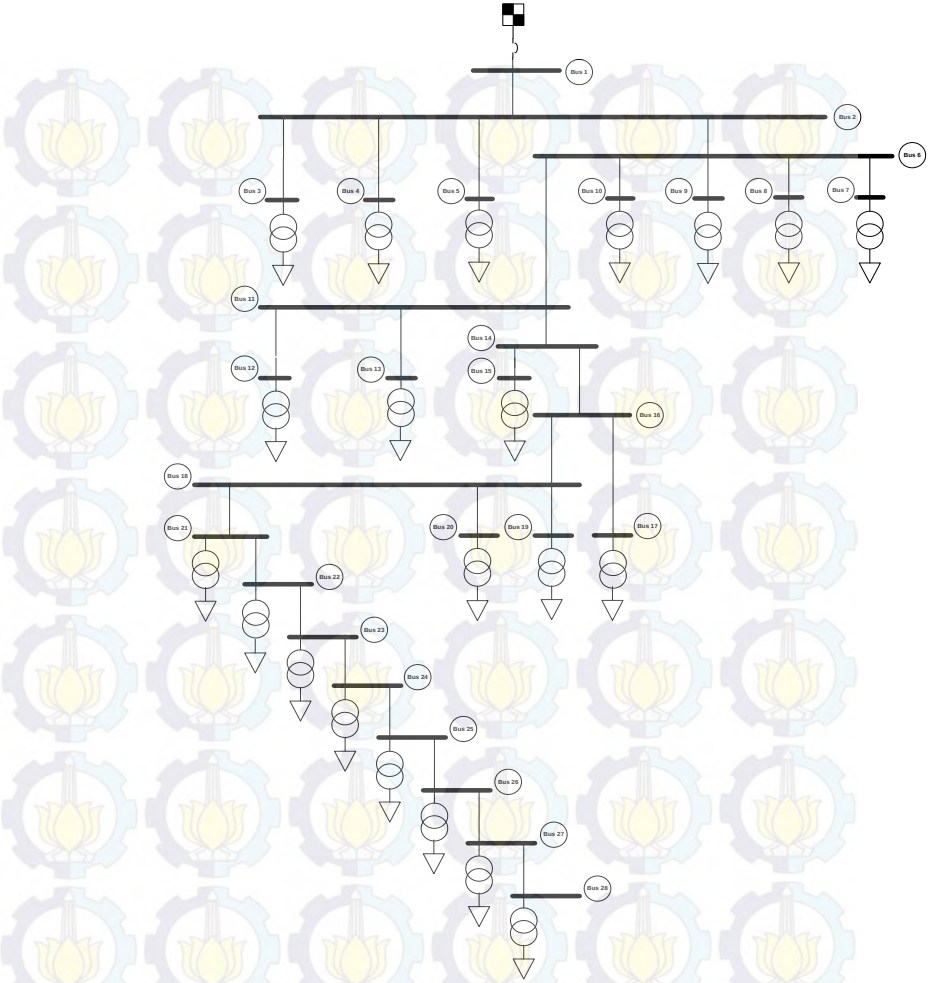
Sistem distribusi Surabaya Utara terdiri dari lima penyulang, yaitu penyulang kaliasin, penyulang basuki rahmat, penyulang ometraco, penyulang tunjungan dan penyulang tegalsari. Masing-masing penyulang memiliki konfigurasi jaringan dan jumlah beban yang berbeda-beda.

Penyulang kaliasin terdiri dari 10 bus dengan pembebanan pada bus 3, 5, 6, 9, dan 10 serta terdapat 10 cabang saluran. Penyulang basuki rahmat terdiri dari 28 bus dengan 28 cabang saluran yang berbeda. Penyulang ometraco terdiri dari 13 bus dengan 13 cabang saluran yang berbeda. Penyulang tunjungan terdiri dari 12 bus dengan pembebanan yang berbeda serta terdapat 12 cabang saluran yang berbeda. Penyulang tegalsari terdiri dari 20 bus dengan pembebanan yang berbeda serta terdapat 20 cabang saluran yang berbeda.

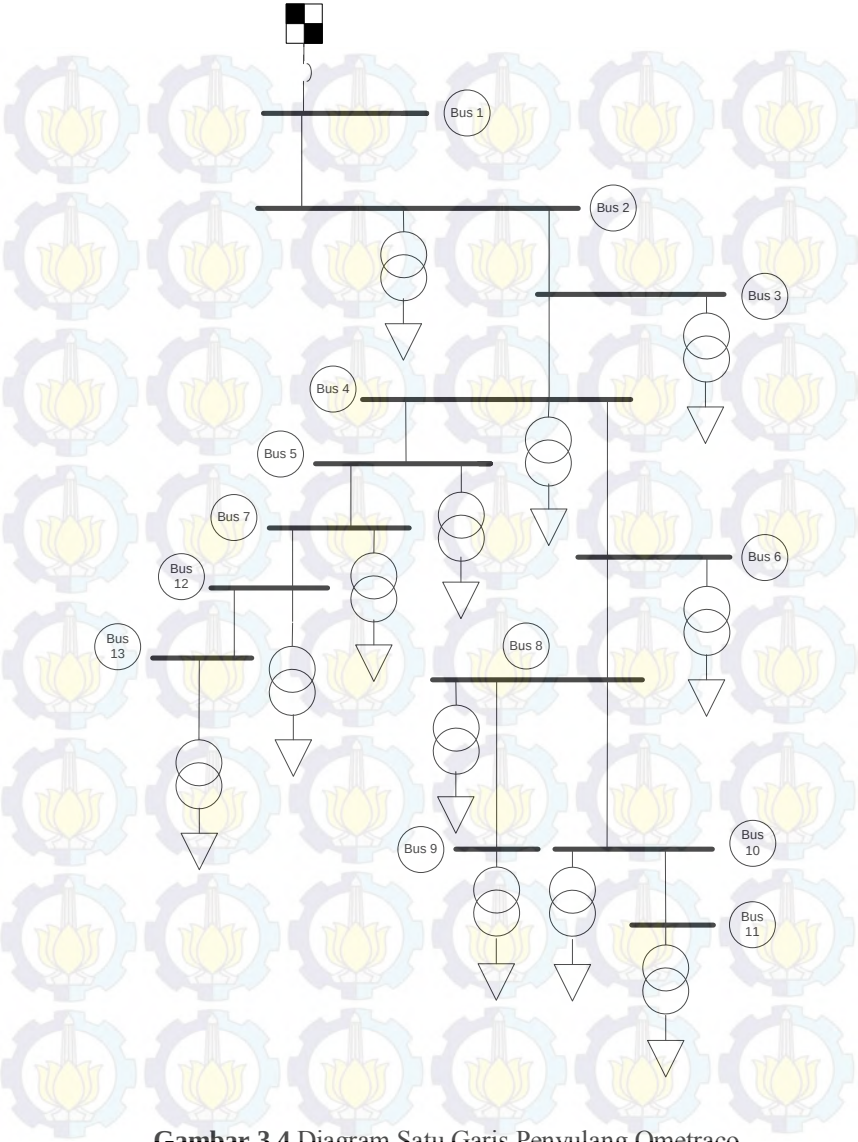
Diagram satu garis untuk penyulang kaliasin, basuki rahmat, ometraco, tunjungan dan tegalsari berturut-turut ditunjukkan pada Gambar 3.2, Gambar 3.3, Gambar 3.4, Gambar 3.5 dan Gambar 3.6.



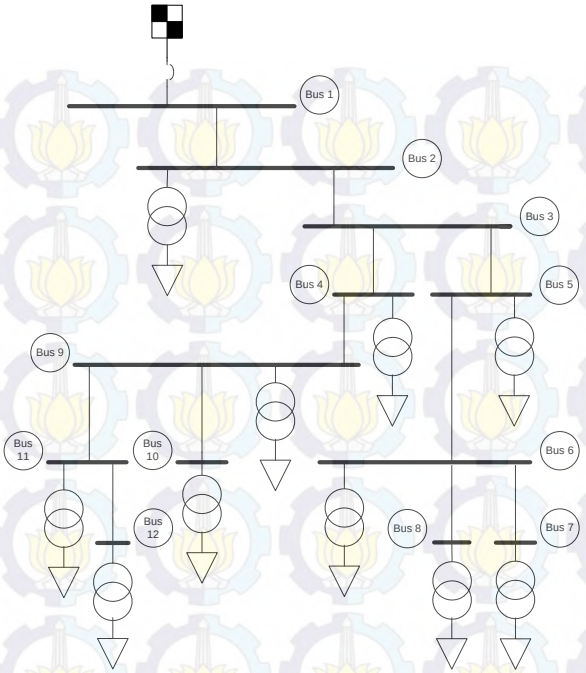
Gambar 3.2 Diagram Satu Garis Penyulang Kaliasin



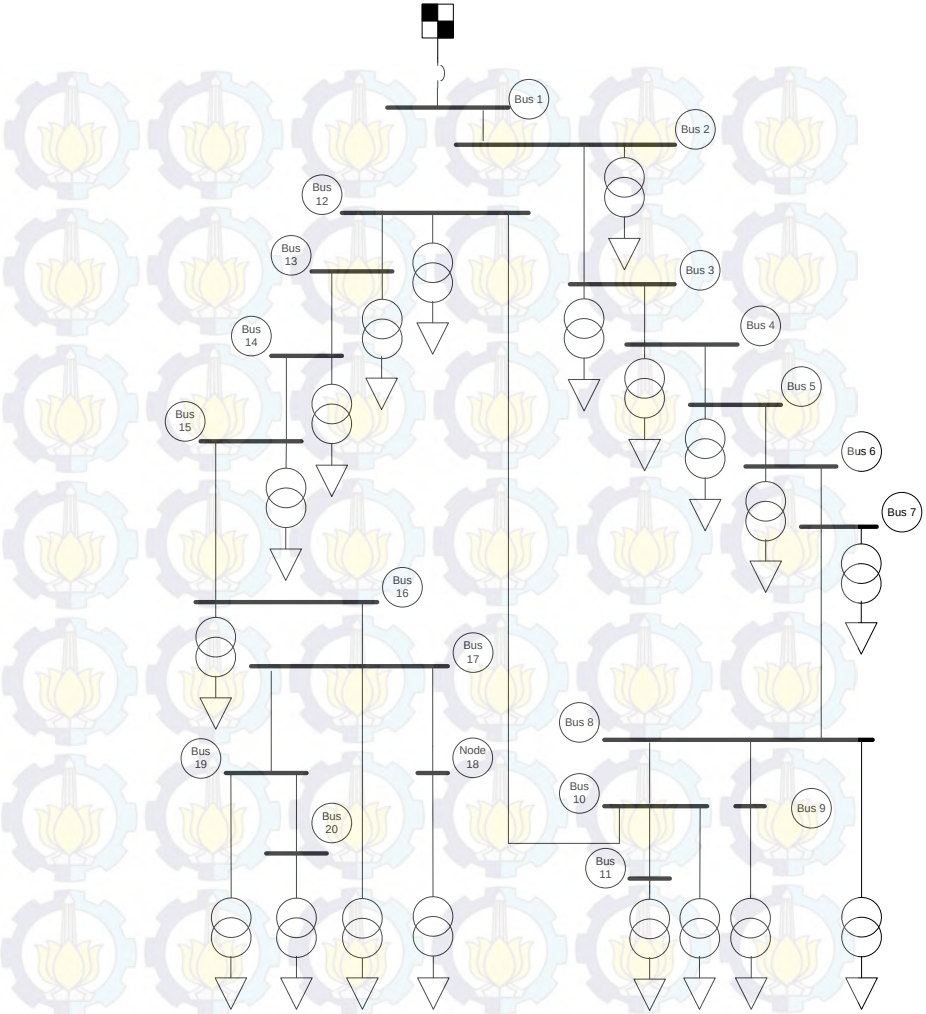
Gambar 3.3 Diagram Satu Garis Penyulang Basuki Rahmat



Gambar 3.4 Diagram Satu Garis Penyulang Ometraco



Gambar 3.5 Diagram Satu Garis Penyulang Tunjungan



Gambar 3.6 Diagram Satu Garis Penyulang Tegalsari

Data beban untuk penyulang kaliasin, basuki rahmat, ometraco, tunjungan dan tegalsari ditunjukkan pada Tabel 3.1, Tabel 3.2, Tabel 3.3, Tabel 3.4, dan Tabel 3.5.

Tabel 3.1 Data Beban Penyulang Kaliasin

No Bus	P(KW)	Q(KVar)
1	0	0
2	0	0
3	74	19
4	0	0
5	58	18
6	94	27
7	0	0
8	64	17
9	0	0
10	312	74

Tabel 3.2 Data Beban Penyulang Basuki Rahmat

No Bus	P(KW)	Q(KVar)
1	0	0
2	0	0
3	279	61
4	29	6
5	39	9
6	0	0
7	342	87
8	601	100
9	66	18
10	54	25
11	0	0
12	25	5

Lanjutan Tabel 3.2 Data Beban Penyulang Basuki Rahmat

No Bus	P(KW)	Q(KVar)
13	455	127
14	0	0
15	12	3
16	0	0
17	317	86
18	0	0
19	67	18
20	108	29
21	83	23
22	146	58
23	129	34
24	78	18
25	97	28
26	92	22
27	40	6
28	38	12

Tabel 3.3 Data Beban Penyulang Ometraco

No Bus	P(KW)	Q(KVar)
1	0	0
2	250	63
3	68	16
4	2	1
5	103	24
6	116	26
7	178	40
8	108	32
9	200	64

Lanjutan Tabel 3.3 Data Beban Penyulang Ometraco

No Bus	P(KW)	Q(KVar)
10	131	33
11	960	285
12	368	76
13	377	115

Tabel 3.4 Data Beban Penyulang Tunjungan

No Bus	P(KW)	Q(KVar)
1	0	0
2	26	7
3	73	23
4	183	43
5	300	94
6	73	87
7	337	106
8	158	96
9	178	58
10	92	21
11	225	59
12	279	74

Tabel 3.5 Data Beban Penyulang Tegalsari

No Bus	P(KW)	Q(KVar)
1	0	0
2	96	32
3	100	19
4	77	23
5	4	0.42
6	33	9

Lanjutan Tabel 3.5 Data Beban Penyulang Tegalsari

No Bus	P(KW)	Q(KVar)
7	77	20
8	31	9
9	61	21
10	25	9
11	319	76
12	63	12
13	68	16
14	10	4
15	96	22
16	61	20
17	88	35
18	6	6
19	157	33
20	25	6

Data lainnya adalah data impedansi saluran. Data impedansi saluran untuk penyulang kaliasin, basuki rahmat, ometraco, tunjungan dan tegalsari berturut-turut ditunjukkan pada ditunjukkan pada Tabel 3.6, Tabel 3.7, Tabel 3.8, Tabel 3.9, dan Tabel 3.10.

Tabel 3.6 Data Impedansi Saluran Penyulang Kaliasin

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
1	1	0.265	0.13	1000
1	2	0.265	0.13	20
2	3	0.265	0.13	250
2	4	0.265	0.13	39
4	5	0.265	0.13	39.8
4	6	0.265	0.13	51.96
4	7	0.265	0.13	230
7	8	0.265	0.13	53
7	9	0.265	0.13	200
9	10	0.265	0.13	1000

Tabel 3.7 Data Impedansi Saluran Penyulang Basuki Rahmat

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
1	2	0.265	0.13	562
2	3	0.265	0.13	100
2	4	0.265	0.13	25
2	5	0.265	0.13	23.5
2	6	0.265	0.13	100
6	7	0.265	0.13	50
6	8	0.265	0.13	50
6	9	0.265	0.13	50
6	10	0.265	0.13	50
6	11	0.265	0.13	100
11	12	0.265	0.13	50
11	13	0.265	0.13	50
11	14	0.265	0.13	300
14	15	0.265	0.13	50
14	16	0.265	0.13	130
16	17	0.265	0.13	50

Lanjutan Tabel 3.7 Data Impedansi Saluran Penyulang Basuki Rahmat

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
16	18	0.265	0.13	63
18	19	0.265	0.13	50
18	20	0.265	0.13	50
18	21	0.265	0.13	100
21	22	0.265	0.13	117.38
22	23	0.265	0.13	25
23	24	0.265	0.13	52
24	25	0.265	0.13	100
25	26	0.265	0.13	31.17
26	27	0.265	0.13	54.84
27	28	0.265	0.13	44.93

Tabel 3.8 Data Impedansi Saluran Penyulang Ometraco

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
1	2	0.265	0.13	377.67
2	3	0.265	0.13	100
3	4	0.265	0.13	18
4	5	0.265	0.13	100
4	6	0.265	0.13	107.9
5	7	0.265	0.13	100
6	8	0.265	0.13	100
8	9	0.265	0.13	100
8	10	0.265	0.13	100
10	11	0.265	0.13	100
7	12	0.265	0.13	100
12	13	0.265	0.13	100

Tabel 3.9 Data Impedansi Saluran Penyulang Tunjungan

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
1	2	0.265	0.13	100
2	3	0.265	0.13	100
3	4	0.265	0.13	100
3	5	0.265	0.13	100
5	6	0.265	0.13	100
6	7	0.265	0.13	100
6	8	0.265	0.13	100
4	9	0.265	0.13	200
9	10	0.265	0.13	100
9	11	0.265	0.13	100
11	12	0.265	0.13	100

Tabel 3.10 Data Impedansi Saluran Penyulang Tegalsari

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
1	2	0.265	0.13	50.55
2	3	0.265	0.13	186.35
3	4	0.265	0.13	81.8
4	5	0.265	0.13	38.22
5	6	0.265	0.13	151.9
6	7	0.265	0.13	48
7	8	0.265	0.13	211
8	9	0.265	0.13	50
8	10	0.265	0.13	205.89
10	11	0.265	0.13	270.64
10	12	0.265	0.13	566.85
12	13	0.265	0.13	249.3
13	14	0.265	0.13	31.46

Lanjutan Tabel 3.10 Data Impedansi Saluran Penyulang Tegalsari

Dari Bus	Ke Bus	R (ohm/km)	X (ohm/km)	Panjang (meter)
14	15	0.265	0.13	242.5
15	16	0.265	0.13	33.4
16	17	0.265	0.13	64.49
17	18	0.265	0.13	100
17	19	0.265	0.13	22.48
19	20	0.265	0.13	33.7

3.1.2 Perancangan Algoritma Extended Kalman Filter

Extended Kalman Filter (EKF) merupakan pengembangan dari metode Kalman Filter. EKF menggabungkan antara sistem dengan vektor pengukuran z_k melalui transformasi deret Taylor. EKF digunakan untuk mengestimasi variabel x_k dari sistem dinamik non-linier. Dengan kata lain, EKF digunakan untuk melinearkan fungsi non-linear menggunakan deret Taylor.

Pemodelan untuk sistem non-linear x_k dari EKF dapat diuraikan pada persamaan 3.1.[6]

$$x_k = f(x_{k-1}) + w_k \tag{3.1}$$

Dimana x_k dan x_{k-1} merupakan vektor state pada waktu k dan k-1, $f(x_{k-1})$ merupakan fungsi non linier yang menggantikan matriks F dan G pada Kalman Filter, w_k merupakan *noise* proses dengan mean 'nol' dan dengan kovarians Q $\sim N(0, Q)$.[6]

Pemodelan untuk pengukuran dari EKF diuraikan pada persamaan 3.2.[6]

$$z_k = \begin{bmatrix} z_{k1} \\ z_{k2} \\ \cdots \\ \cdots \\ z_{km} \end{bmatrix} = \begin{bmatrix} h_1(x_{k1}, x_{k2}, \ldots, x_{kn}) \\ \cdots \\ \cdots \\ \cdots \\ h_m(x_{k1}, x_{k2}, \ldots, x_{kn}) \end{bmatrix} + \begin{bmatrix} v_{k1} \\ v_{k2} \\ \cdots \\ \cdots \\ v_{km} \end{bmatrix} = h(x_k) + v_k \quad (3.2)$$

Dimana nilai z_k adalah vektor pengukuran dengan (m) jumlah sensor pengukuran, $h(x_k)$ adalah fungsi non linier dari pengukuran untuk vektor keadaan *m* dengan v_k adalah *noise* vektor pengukuran, *noise* ini berupa *error* alat ukur. *M* adalah jumlah pengukuran sedangkan *n* adalah variabel state estimasi.

3.1.2.1 Proses Pelinieran *EKF*

EKF bekerja dengan cara melinierkan proses dan pengukuran menggunakan deret Taylor di sekitar mean dan kovarians. Pada persamaan (3.3) dan (3.4) variabel state x_k dan z_k dapat diestimasi *dengan memperhatikan nilai* w_k *dan* v_k [6]:

$$\hat{x}_{\bar{k}} = f(\hat{x}_{k-1}) \quad (3.3)$$

$$\hat{z}_{\bar{k}} = h(\hat{x}_{\bar{k}}) \quad (3.4)$$

$$x_k - \hat{x}_{\bar{k}} = f(x_{k-1}) - f(\hat{x}_{k-1}) + w_k \quad (3.5)$$

Dan,

$$z_k - \hat{z}_{\bar{k}} = h(x_{k-1}) - h(\hat{x}_{\bar{k}}) + v_k \quad (3.6)$$

variabel state x_k dan z_k yang diestimasi *dengan deret Taylor* [6]*:*

$$f(x_{k-1}) = f(\hat{x}_{k-1}) + f'(\hat{x}_{k-1})(x_{k-1} - \hat{x}_{k-1}) + \frac{1}{2} f''(\hat{x}_{k-1})(x_{k-1} - \hat{x}_{k-1})^2 + \cdots + w_k$$

$$h(x_k) = h(\hat{x}_{\bar{k}}) + h'(\hat{x}_{\bar{k}})(x_k - \hat{x}_k) + \frac{1}{2}h''(\hat{x}_{\bar{k}})(x_k - \hat{x}_{\bar{k}})^2 + \cdots + v_k$$

Jika turunan orde ≥ 2 diabaikan, maka persamaan (3.5) dan (3.6) menjadi :

$$x_k - \hat{x}_{\bar{k}} = f'(\hat{x}_{k-1})(x_{k-1} - \hat{x}_{k-1}) + w_k \quad (3.7)$$

$$z_k - \hat{z}_{\bar{k}} = h'(\hat{x}_{\bar{k}})(x_k - \hat{x}_k) + v_k \quad (3.8)$$

Atau

$$x_k = \hat{x}_{\bar{k}} + F_k(x_{k-1} - \hat{x}_{k-1}) + w_k \quad (3.9)$$

$$z_k = \hat{z}_{\bar{k}} + H_k(x_k - \hat{x}_{\bar{k}}) + v_k \quad (3.10)$$

Persamaan (3.9) dan (3.10) merupakan persamaan linier.

Dimana $x_k - \hat{x}_k = e_{\bar{k}}$,sehingga $e_{\bar{k}} = F_k(x_{k-1} - \hat{x}_{k-1}) + w_k$

$e_{\bar{k}}$ merupakan error estimasi priori, untuk mencari nilai kovarians prior $P_{\bar{k}}$, yaitu :

$$P_{\bar{k}} = E[e_{\bar{k}}\, e_{\bar{k}}]^T \quad (3.11)$$

Sehingga di dapatkan :

Estimasi prior dari state : $\hat{x}_{\bar{k}} = f(\hat{x}_{k-1})$ (3.12)

Estimasi kovarians prior : $P_{\bar{k}} = F_k P_{k-1} {F_k}^T + Q$ (3.13)

Setelah estimasi state dan kovarians prior didapatkan, hasil ini dikoreksi kembali pada proses measurement update (proses koreksi).

Dari persamaan (3.10) dapat dirubah menjadi[6] :

$$Z_k = H_k x_k + v_k \quad (3.14)$$

Estimasi state posterior didapatkan dari kombinasi linier dari estimasi state prior $\hat{x}_{\bar{k}}$ dan pengukuran Z_k sebagai berikut[6] :

$$\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - h(\hat{x}_{\bar{k}})) \quad (3.15)$$

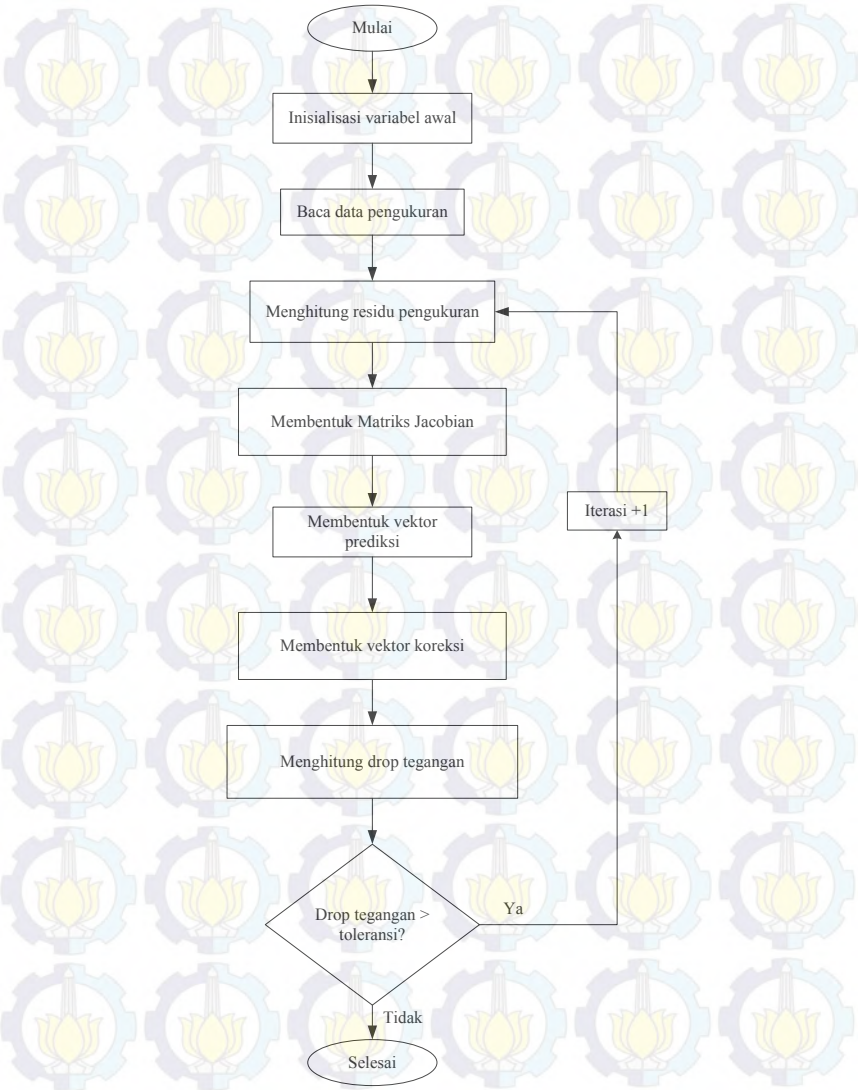
$$K_k = P_{\bar{k}} {H_k}^T (H_k P_{\bar{k}} {H_k}^T + R)^{-1} \quad (3.16)$$

$$P_k = E[e_k e_k]^T$$

$$= P_{\bar{k}} + K_k(HP_{\bar{k}}H^T + R){K_k}^T - K_k H P_{\bar{k}} - (K_k H P_{\bar{k}})^T$$

$$= (\mathrm{I}\text{-}K_k H_k) P_{\bar{k}} \quad (3.17)$$

3.1.2.2 Diagram Alir Program EKF



Gambar 3.7 Diagram alir Extended Kalman Filter

3.1.2.3 Inisialisasi Variabel Awal dan Input Data

Sebelum melakukan iterasi *EKF*, ada beberapa data awal yang harus didefinisikan, diantaranya adalah :

1. Magnitude tegangan semua bus untuk pertama kali didefinisikan bernilai 1 pu dan sudutnya 0^o .
2. Impedansi saluran berupa resistansi (R) dan reaktansi (X). Data ini adalah data *existing* dari plan lima penyulang distribusi Surabaya Utara.
3. Daya aktif dan daya reaktif hasil pengukuran pada bus-bus di bawah bus utama. Daya ini dibagi menjadi dua jenis yaitu, daya injeksi pada bus ($PQ_{injection}$) dan daya yang mengalir pada saluran (PQ_{flow}). Jumlah dari daya yang terukur tergantung pada jumlah sensor yang terpasang. Nilai data ini juga berasal dari data *existing* beban terpasang pada plan lima penyulang distribusi di Surabaya Utara.
4. Toleransi error dari masing-masing alat pengukuran (R). Nilai ini diasumsikan 10^{-4} untuk pengukuran daya injeksi pada bus dan 6.4×10^{-5}.

Untuk pertama kali, data impedansi (Z) harus dibentuk seperti pada Persamaan 3.1.

$$Z = R + jX \tag{3.18}$$

dimana :

$R = resistansi\ saluran$

$X = reaktansi\ saluran$

Kemudian dari data impedansi di atas dapat dibentuk data admitansi (Y) sesuai persamaan 3.19.[12]

$$Y = \frac{1}{Z} = \frac{1}{R+jX} \tag{3.19}$$

Data admitansi di atas dapat dipisahkan berdasarkan jenis komponennya sesuai Persamaan 3.3.[12]

$$Y = G + jB \tag{3.20}$$

dimana :

$$G = komponen\ real\ (konduktansi)$$

$$B = komponen\ imajinener\ (suseptansi)$$

Data admitansi di atas digunakan untuk membentuk matriks admitansi (*Ybus*). Matriks admitansi adalah matriks dengan ukuran *n* baris dan *n* kolom. Dimana *n* adalah jumlah bus pada sistem. Setiap bus pada sistem tenaga pasti terhubung dengan bus lainnya. Sehingga matriks admitansi dapat didefinisikan seperti pada Persamaan 3.21.[13]

$$Ybus = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & \dots & Y_{1n} \\ Y_{21} & \dots & \dots & \dots & Y_{2n} \\ Y_{31} & \dots & \dots & \dots & Y_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ Y_{n1} & Y_{n2} & Y_{n3} & \dots & Y_{nn} \end{bmatrix} \tag{3.21}$$

Untuk dapat membentuk persamaan di atas dibutukan Persamaan 3.22 di bawah ini.[13]

$$Yij = \begin{cases} Y_{ii} + \sum_{i \neq j} Y_{ij} & untuk\ i = j \\ -Y_{ij} & untuk\ i \neq j \end{cases} \tag{3.22}$$

Salah satu komponen matriks di atas adalah Y_{12} . Variabel tersebut maksudnya adalah nilai admitansi pada saluran 1-2. Sedangkan komponen matriks dengan baris dan kolom yang sama seperti Y_{11}, Y_{22} hingga Y_{nn} disebut sebagai *self admittance*.[13] Apabila nilai matriks dijumlahkan dengan semua komponen matriks pada baris atau kolomnya, nilainya harus saling menghilangkan.

Kovarians untuk noise proses (Q) merupakan perkalian antara Matriks identitas dari sm dengan kuadrat dari q (standar proses, q = 0.1. sm merupakan ukuran matriks. Inisialisasi awal untuk kovarians P_0 berupa matriks identitas dari sm. Misalkan jumlah busnya ada 2, nilai sm(ukuran matriksnya = nbus x 2-1, sm = 3, ukuran matriksnya 3x3. Maka dapat ditentukan[6] :

$$Q = 0.1^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q = \begin{matrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{matrix}$$

$$P_0 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Data lain yang dibutuhkan sebelum melakukan iterasi adalah data hasil pengukuran (z) dan toleransi error alat pengukuran yang digunakan (R). Kedua data ini direpresentasikan dalam matriks 1 kolom dan beberapa baris yang terdiri dari pengukuran tegangan pada bus utama, pengukuran daya injeksi pada bus dan pengukuran daya pada saluran. Matriks z dan matriks R berturut-turut ditunjukkan pada Persamaan 3.23 dan Persamaan 3.24 [8].

$$z = \begin{bmatrix} V_{mag} \\ P_{inj} \\ Q_{inj} \\ P_{flow} \\ Q_{flow} \end{bmatrix} \tag{3.23}$$

$$R = \begin{bmatrix} RV_{mag} \\ RP_{inj} \\ RQ_{inj} \\ RP_{flow} \\ RQ_{flow} \end{bmatrix} \quad (3.24)$$

Setelah semua data awal didefinisikan, selanjutnya masuk pada proses iterasi EKF. Iterasi ini bertujuan untuk mencari selisih tegangan (ΔV) sesuai toleransi yang diinginkan antara iterasi sekarang dan iterasi sebelumnya.

3.1.2.4 Iterasi EKF

Pada iterasi pertama, dibentuk matriks *h* yang merupakan fungsi non-linier dari data pengukuran. Matriks ini dibentuk dari lima komponen utama dari data pengukuran itu sendiri. Matriks ini direpresentasikan pada Persamaan 3.25.

$$h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} \quad (3.25)$$

Masing-masing komponen matriks *h* adalah h_1, h_2, h_3, h_4 dan h_5 yang berturut-turut ditunjukkan pada Persamaan 3.26, 3.27, 3.28, 3.29 dan 3.30 [10].

$$h_1 = Vmag_1 = 1 \quad (3.26)$$

$$h_2 = Pinj_{\,i} = \sum_{k=1}^{N} |V_i||V_k| \,(G_{ik}\ cos(\theta_i - \theta_k) + B_{ik}\ sin(\theta_i - \theta_k)) \quad (3.27)$$

$$h_3 = Qinj_{\ i} = \sum_{k=1}^{N} |V_i||V_k| (G_{ik}\ sin(\theta_i - \theta_k) - B_{ik}\ cos(\theta_i - \theta_k)) \quad (3.28)$$

$$h_4 = Pflow_{\ ik} = |V_i||V_k|(G_{ik}\ cos(\theta_i - \theta_k) + B_{ik}\ sin(\theta_i - \theta_k)) - |V_i||V_i|G_{ik} \quad (3.29)$$

$$h_5 = Qflow_{\ ik} = |V_i||V_k|(G_{ik}\ sin\ (\theta_i - \theta_k) - B_{ik}cos(\theta_i - \theta_k)) + |V_i||V_i|B_{ik} \quad (3.30)$$

dimana :

$Vmag_1$ = *magnitude tegangan bus utama*

$Pinj_{\ i}$ = *injeksi daya aktif bus ke i*

$Qinj_{\ i}$ = *injeksi daya reaktif bus ke i*

$Pflow_{\ ik}$ = *aliran daya aktif dari bus i ke bus k*

$Qflow_{\ ik}$ = *aliran daya reaktif dari bus i ke bus k*

N = *jumlah bus*

G_{ik} = *konduktansi antara bus i dan bus k*

B_{ik} = *suseptansi antara bus i dan bus k*

V_i = *magnitude tegangan pada bus i*

θ_i = *sudut tegangan pada bus i*

V_k = *magnitude tegangan pada bus k*

θ_k = *sudut tegangan pada bus k*

Matriks *h* ini digunakan untuk mencari residu pengukuran dengan cara mengurangkannya ke data asli pengukuran (*z*). Fungsi residual dapat dicari menggunakan Persamaan 3.31.

$$r = z - h \tag{3.31}$$

Variabel lain yang dibutuhkan untuk proses iterasi adalah matriks Jacobian (*H*). Matriks jacobian adalah matriks semua turunan parsial orde pertama dari suatu nilai fungsi vektor. Dalam Tugas Akhir ini, matriks jacobian yang digunakan berasal dari data jenis pengukuran yang dibaca yang kemudian diturunkan satu orede berdasarkan tegangan dan sudutnya. Komponen matriks jacobian dalam Tugas Akhir ini ditunjukkan pada Persamaan 3.32 [2].

$$H = \begin{bmatrix} \frac{\partial V}{\partial \theta} & \frac{\partial V}{\partial V} \\ \frac{\partial P_{inj}}{\partial \theta} & \frac{\partial P_{inj}}{\partial V} \\ \frac{\partial Q_{inj}}{\partial \theta} & \frac{\partial Q_{inj}}{\partial V} \\ \frac{\partial P_{flow}}{\partial \theta} & \frac{\partial P_{flow}}{\partial V} \\ \frac{\partial Q_{flow}}{\partial \theta} & \frac{\partial Q_{flow}}{\partial V} \end{bmatrix} \tag{3.32}$$

Setelah nilai matriks jacobian didapat, membentuk vector prediksi dengan persamaan (3.33).[6]

$$P_{\bar{k}} = ([P_0] + [Q]) + H^{\mathrm{T}} \tag{3.33}$$

Kemudian membentuk vector koreksi dengan cara mencari K_k Kalman Gain, menghitung kovarian koreksi (P_k), mencari State koreksi x_k, $x_k = \Delta E$ pada persamaan 3.34, 3.35, 3.36.[6]

$$K_k = P_{\bar{k}}.[(H.P_{\bar{k}}) + R_i]^{-1} \tag{3.34}$$

$$P_k = P_0 - [K_k.{P_{\bar{k}}}^T] \tag{3.35}$$

$$x_k = \Delta E$$

$$\Delta E = K_k.r \tag{3.36}$$

Apabila terjadi *drop* tegangan, maka nilai dari persamaan 3.36

akan negatif. Selanjutnya nilai tersebut ditambahkan dengan nilai magnitude tegangan awal seperti pada persamaan 3.37.

$$E = E + \Delta E \tag{3.37}$$

Untuk menghitung error pengukuran dapat dicari dengan persamaan 3.38 dan 3.39 [10]:

$$\varepsilon_{v\_est} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(v_{i\_est} - v_{i_{true}})}{v_{i_{true}}}\right| \text{x } 100\% \tag{3.38}$$

$$\varepsilon_{\theta\_est} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(\theta_{i\_est} - \theta_{i_{true}})}{\theta_{i_{true}}}\right| \text{x } 100\% \tag{3.39}$$

Dimana $v_{i_{true}}$ dan $\theta_{i_{true}}$ merupakan nilai magnitude tegangan dan sudut fasa yang sebenarnya dari data pengukuran ETAP. v_{i_est} dan θ_{i_est} adalah tegangan hasil estimasi dan sudut fasa hasil estimasi. ε_{v_est}merupakan error perbandingan antara nilai magnitude tegangan yang sebenarnya dengan nilai magnitude tegangan hasil estimasi. ε_{θ_est} merupakan error perbandingan antara sudut fasa yang sebenarnya dengan sudut fasa hasil estimasi.

Halaman ini sengaja dikosongkan

BAB 4
SIMULASI DAN ANALISIS

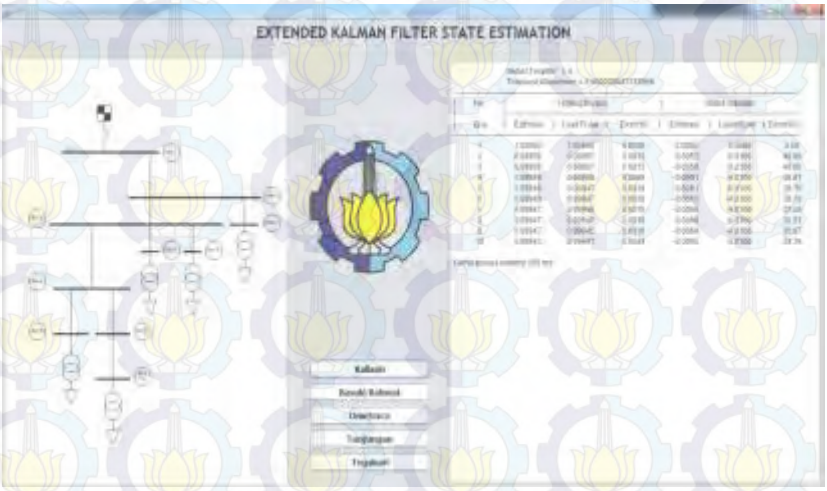
4.1 Simulasi State Estimation

Simulasi state estimation dilakukan dengan cara melakukan running program *extended kalman filter state estimation* dengan data saluran dan data bus dari lima penyulang di jaringan distribusi 20KV Surabaya Utara. Output dari simulasi state estimation adalah data magnitude tegangan dan sudut fasa dari masing-masing bus.

Pada tugas akhir ini pengambilan data dilakukan dengan kondisi semua sensor terpasang. Data-data tersebut selanjutnya divalidasi dengan data hasil aliran daya program ETAP agar dapat diketahui nilai errornya. Dari hasil running program dapat diketahui error rata-rata terbesar dari ke 5 penyulang Surabaya Utara sebesar 0.0071%.

4.1.1 Simulasi Penyulang Kaliasin

Hasil simulasi state estimasi pada penyulang kaliasin dengan metode EKF dapat dilihat pada Gambar 4.1



Gambar 4.1 Running program state estimasi penyulang kaliasin

Pada gambar 4.1 merupakan tampilan hasil simulasi state estimasi untuk penyulang kaliasin. Pada frame tersebut terdapat 5 tombol, diantaranya tombol Kaliasin, tombol Basuki Rahmat,tombol Ometraco, tombol Tunjungan, dan tombol Tegalsari. Apabila tombol Kaliasin ditekan, maka akan muncul hasil estimasi tegangan & sudut fasa pada sisi kanan, dan tampilan Single Line Diagram penyulang Kaliasin pada sisi kiri.

Tabel 4.1 Hasil Simulasi State Estimation Penyulang Kaliasin dengan metode EKF

No. Bus	Tegangan			Sudut		
	Estimasi (pu)	Load Flow (pu)	Error (%)	Estimasi (^{o})	Load Flow (^{o})	Error (%)
1	1.00000	1.00000	0.0000	0.0000	0.0000	0.0000
2	0.99958	0.99957	0.0012	-0.0050	-0.0100	49.98
3	0.99958	0.99957	0.0012	-0.0050	-0.0100	49.95
4	0.99949	0.99948	0.0009	-0.0061	-0.0100	38.81
5	0.99949	0.99947	0.0018	-0.0061	-0.0100	38.70
6	0.99949	0.99947	0.0018	-0.0063	-0.0100	38.70
7	0.99947	0.99946	0.0015	-0.0064	-0.0100	37.06
8	0.99947	0.99945	0.0018	-0.0064	-0.0100	36.21
9	0.99947	0.99945	0.0016	-0.0064	-0.0100	35.87
10	0.99945	0.99941	0.0044	-0.0066	-0.0100	34.38

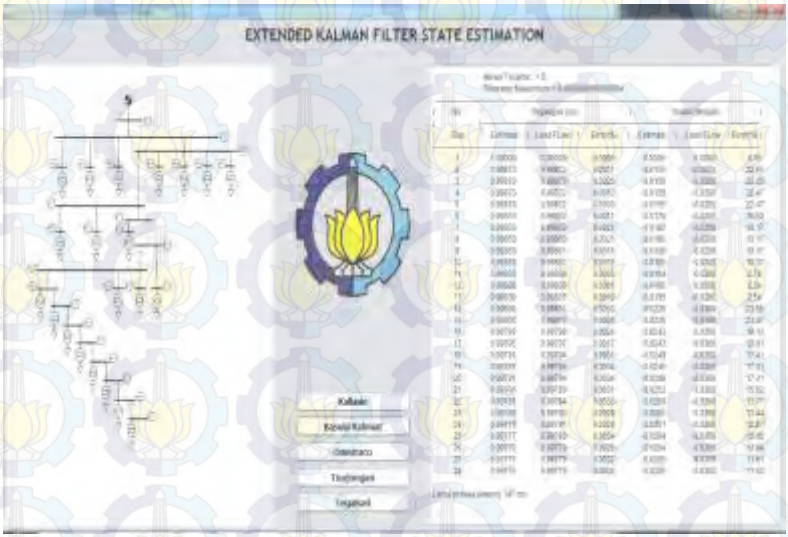
Tabel 4.1 di atas merupakan hasil estimasi tegangan & sudut fasa penyulang kaliasin. Hasil yang ada pada tabel ini memperjelas tampilan gambar 4.1 pada sisi kanan. Dari tabel di atas, didapatkan tegangan estimasi rata-rata untuk penyulang kaliasin bernilai 0.999549 pu. Sedangkan untuk tegangan terkecil bernilai 0.99945 pu. Tegangan paling besar selain magnitude tegangan pada bus utama bernilai 0.99958 pu. Hasil yang didapatkan ini tidak undervoltage, karena nilai magnitude tegangan berkisar pada nilai 0.99 pu.

Untuk estimasi sudutnya bernilai besar karena data pembanding yang digunakan untuk validasi pada ETAP merupakan angka pembulatan hanya dua angka di belakang koma, sedangkan data hasil

simulasi *JAVA* merupakan pembulatan hingga empat angka di belakang koma sehingga berpengaruh pada komputasinya di program.

4.1.2 Simulasi Penyulang Basuki Rahmat

Hasil simulasi state estimasi pada penyulang basuki rahmat dengan metode EKF dapat dilihat pada Gambar 4.2.



Gambar 4.2 Running program state estimasi penyulang basuki rahmat

Pada gambar 4.2 merupakan tampilan hasil simulasi state estimasi untuk penyulang Basuki Rahmat. Pada frame tersebut terdapat 5 tombol, diantaranya tombol Kaliasin, tombol Basuki Rahmat,tombol Ometraco, tombol Tunjungan, dan tombol Tegalsari. Apabila tombol Basuki Rahmat ditekan, maka akan muncul hasil estimasi tegangan & sudut fasa pada sisi kanan, dan tampilan Single Line Diagram penyulang Basuki Rahmat pada sisi kiri.

Tabel 4.2 Hasil Simulasi State Estimation Penyulang Basuki Rahmat dengan metode EKF

No. Bus	Tegangan			Sudut		
	Estimasi (pu)	Load Flow (pu)	Error (%)	Estimasi (°)	Load Flow (°)	Error (%)
1	1.00000	1.00000	0.0000	0.0000	0.0000	0.0000
2	0.99873	0.99872	0.0011	-0.0155	-0.0200	22.56
3	0.99873	0.99870	0.0026	-0.0156	-0.0200	22.20
4	0.99873	0.99872	0.0010	-0.0155	-0.0200	22.47
5	0.99873	0.99872	0.0010	-0.0155	-0.0200	22.47
6	0.99853	0.99852	0.0011	-0.0179	-0.0200	10.52
7	0.99853	0.99850	0.0026	-0.0180	-0.0200	10.17
8	0.99853	0.99850	0.0026	-0.0180	-0.0200	10.17
9	0.99853	0.99851	0.0016	-0.0180	-0.0200	10.17
10	0.99853	0.99851	0.0016	-0.0180	-0.0200	10.17
11	0.99839	0.99839	0.0003	-0.0194	-0.0200	2.78
12	0.99839	0.99839	0.0001	-0.0195	-0.0200	2.54
13	0.99839	0.99837	0.0019	-0.0195	-0.0200	2.54
14	0.99808	0.99811	0.0033	-0.0230	-0.0300	23.55
15	0.99807	0.99810	0.0025	-0.0243	-0.0300	23.47
16	0.99796	0.99798	0.0024	-0.0243	-0.0300	19.13
17	0.99795	0.99797	0.0017	-0.0248	-0.0300	19.01
18	0.99791	0.99794	0.0031	-0.0248	-0.0300	17.41
19	0.99791	0.99794	0.0034	-0.0248	-0.0300	17.31
20	0.99791	0.99794	0.0034	-0.0254	-0.0300	17.31
21	0.99786	0.99789	0.0031	-0.0253	-0.0300	15.62
22	0.99781	0.99784	0.0030	-0.0259	-0.0300	13.77
23	0.99780	0.99783	0.0029	-0.0260	-0.0300	13.44
24	0.99779	0.99781	0.0024	-0.0261	-0.0300	12.87
25	0.99777	0.99780	0.0034	-0.0264	-0.0300	12.05
26	0.99776	0.99779	0.0028	-0.0264	-0.0300	11.84
27	0.99776	0.99779	0.0032	-0.0265	-0.0300	11.61
28	0.99776	0.99779	0.0034	-0.0265	-0.0300	11.52

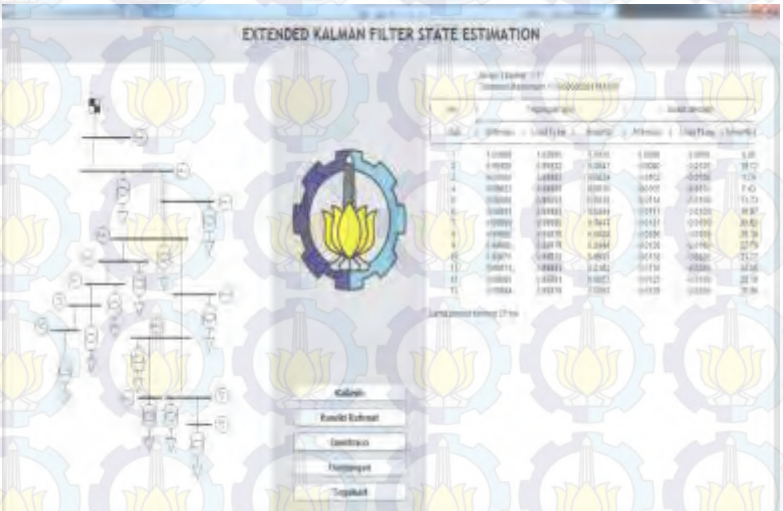
Tabel 4.2 di atas merupakan hasil estimasi tegangan & sudut fasa penyulang Basuki Rahmat. Hasil yang ada pada tabel ini memperjelas

tampilan gambar 4.2 pada sisi kanan. Dari tabel di atas, didapatkan tegangan estimasi rata-rata untuk penyulang Basuki Rahmat bernilai 0.998244 pu. Sedangkan untuk tegangan terkecil bernilai 0.99776 pu. Tegangan paling besar selain magnitude tegangan pada bus utama bernilai 0.99873 pu. Hasil yang didapatkan ini tidak undervoltage, karena nilai magnitude tegangan berkisar pada nilai 0.99 pu.

Untuk estimasi sudutnya bernilai besar karena data pembanding yang digunakan untuk validasi pada ETAP merupakan angka pembulatan hanya dua angka di belakang koma, sedangkan data hasil simulasi *JAVA* merupakan pembulatan hingga empat angka di belakang koma sehingga berpengaruh pada komputasinya di program.

4.1.3 Simulasi Penyulang Ometraco

Hasil simulasi state estimasi pada penyulang ometraco dengan metode EKF dapat dilihat pada Gambar 4.3.



Gambar 4.3 Running program state estimasi penyulang Ometraco

Pada gambar 4.3 merupakan tampilan hasil simulasi state estimasi untuk penyulang Ometraco. Pada frame tersebut terdapat 5 tombol, diantaranya tombol Kaliasin, tombol Basuki Rahmat,tombol Ometraco, tombol Tunjungan, dan tombol Tegalsari. Apabila tombol Ometraco ditekan, maka akan muncul hasil estimasi tegangan & sudut fasa pada sisi kanan, dan tampilan Single Line Diagram penyulang Ometraco pada sisi kiri.

Tabel 4.3 Hasil Simulasi State Estimasi Penyulang Ometraco dengan metode EKF

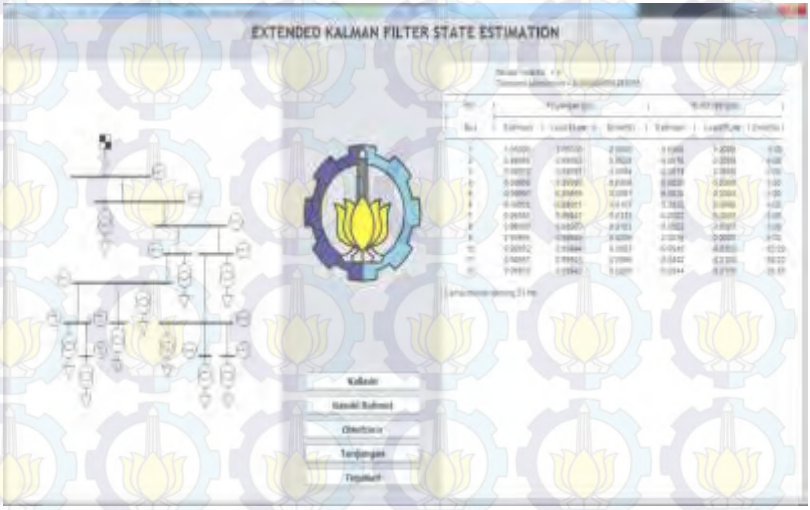
No. Bus	Tegangan			Sudut		
	Estimasi (pu)	Load Flow (pu)	Error (%)	Estimasi (^{o})	Load Flow (^{o})	Error (%)
1	1.00000	1.00000	0.0000	0.0000	0.0000	0.00
2	0.99926	0.99922	0.0041	-0.0080	-0.0100	19.72
3	0.99906	0.99903	0.0034	-0.0102	-0.0100	1.74
4	0.99903	0.99900	0.0030	-0.0105	-0.0100	5.43
5	0.99896	0.99893	0.0030	-0.0114	-0.0100	13.73
6	0.99891	0.99888	0.0034	-0.0117	-0.0100	16.87
7	0.99890	0.99886	0.0043	-0.0121	-0.0100	20.52
8	0.99882	0.99878	0.0038	-0.0126	-0.0100	26.39
9	0.99880	0.99876	0.0044	-0.0128	-0.0100	27.79
10	0.99876	0.99870	0.0056	-0.0132	-0.0200	33.77
11	0.99873	0.99863	0.0102	-0.0135	-0.0200	32.55
12	0.99886	0.99881	0.0053	-0.0125	-0.0100	25.18
13	0.99884	0.99878	0.0060	-0.0128	-0.0200	35.99

Tabel 4.3 di atas merupakan hasil estimasi tegangan & sudut fasa penyulang Basuki Rahmat. Hasil yang ada pada tabel ini memperjelas tampilan gambar 4.3 pada sisi kanan. Dari tabel di atas, didapatkan tegangan estimasi rata-rata untuk penyulang Ometraco bernilai 0.998995 pu. Sedangkan untuk tegangan terkecil bernilai 0.99873 pu. Tegangan paling besar selain magnitude tegangan pada bus utama bernilai 0.99926 pu. Hasil yang didapatkan ini tidak undervoltage, karena nilai magnitude tegangan berkisar pada nilai 0.99 pu.

Untuk estimasi sudutnya bernilai besar karena data pembanding yang digunakan untuk validasi pada ETAP merupakan angka pembulatan hanya dua angka di belakang koma, sedangkan data hasil simulasi *JAVA* merupakan pembulatan hingga empat angka di belakang koma sehingga berpengaruh pada komputasinya di program.

4.1.4 Simulasi Penyulang Tunjungan

Hasil simulasi state estimasi pada penyulang tunjungan dengan metode EKF dapat dilihat pada Gambar 4.4.



Gambar 4.4 Hasil Simulasi State Estimasi Penyulang Tunjungan dengan metode EKF

Pada gambar 4.4 ini merupakan tampilan hasil simulasi state estimasi untuk penyulang Tunjungan. Pada frame tersebut terdapat 5 tombol, diantaranya tombol Kaliasin, tombol Basuki Rahmat,tombol Ometraco, tombol Tunjungan, dan tombol Tegalsari. Apabila tombol Tunjungan ditekan, maka akan muncul hasil estimasi tegangan & sudut fasa penyulang Tunjungan pada sisi kanan, dan tampilan Single Line Diagram penyulang Tunjungan pada sisi kiri.

Tabel 4.4 Hasil Simulasi State Estimasi Penyulang Tunjungan dengan metode EKF

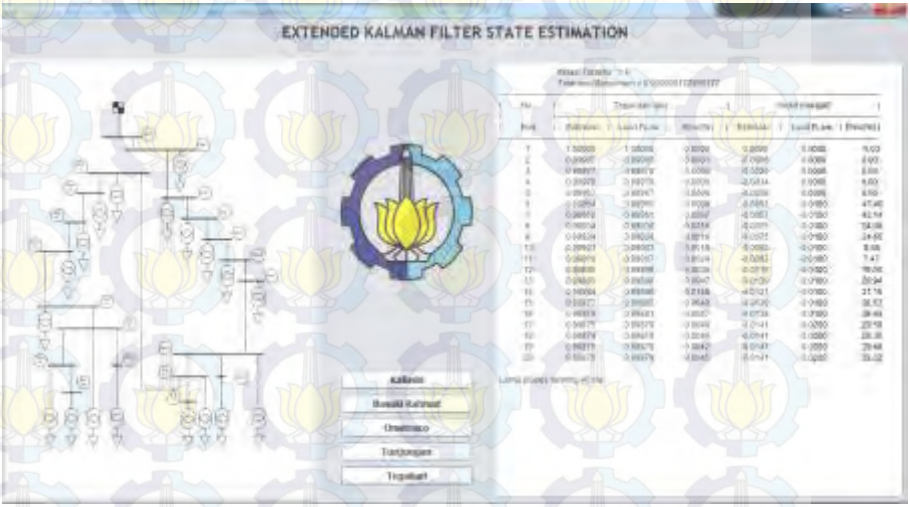
No. Bus	Tegangan			Sudut		
	Estimasi (pu)	Load Flow (pu)	Error (%)	Estimasi (°)	Load Flow (°)	Error (%)
1	1.00000	1.00000	0.0000	0.0000	0.0000	0.0000
2	0.99986	0.99983	0.0029	-0.0010	0.0000	0.0000
3	0.99972	0.99967	0.0054	-0.0019	0.0000	0.0000
4	0.99966	0.99960	0.0058	-0.0026	0.0000	0.0000
5	0.99967	0.99958	0.0087	-0.0020	0.0000	0.0000
6	0.99962	0.99951	0.0107	-0.0022	0.0000	0.0000
7	0.99960	0.99947	0.0133	-0.0022	0.0000	0.0000
8	0.99960	0.99950	0.0103	-0.0022	0.0000	0.0000
9	0.99955	0.99949	0.0056	-0.0038	0.0000	0.0000
10	0.99953	0.99944	0.0093	-0.0040	-0.0100	60.29
11	0.99951	0.99945	0.0065	-0.0042	-0.0100	58.22
12	0.99950	0.99943	0.0069	-0.0044	-0.0100	56.45

Tabel 4.4 di atas merupakan hasil estimasi tegangan & sudut fasa penyulang Tunjungan. Hasil yang ada pada tabel ini memperjelas tampilan gambar 4.4 pada sisi kanan. Dari tabel di atas, didapatkan tegangan estimasi rata-rata untuk penyulang Tunjungan bernilai 0.999652 pu. Sedangkan untuk tegangan terkecil bernilai 0.99950 pu. Tegangan paling besar selain magnitude tegangan pada bus utama bernilai 0.99986 pu. Hasil yang didapatkan ini tidak undervoltage, karena nilai magnitude tegangan berkisar pada nilai 0.99 pu.

Untuk estimasi sudutnya bernilai besar karena data pembanding yang digunakan untuk validasi pada ETAP merupakan angka pembulatan hanya dua angka di belakang koma, sedangkan data hasil simulasi *JAVA* merupakan pembulatan hingga empat angka di belakang koma sehingga berpengaruh pada komputasinya di program.

4.1.5 Simulasi Penyulang Tegalsari

Hasil simulasi state estimasi pada penyulang tegalsari dengan metode EKF dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil Simulasi State Estimasi Penyulang Tegalsari dengan metode EKF

Pada gambar 4.5 ini merupakan tampilan hasil simulasi state estimasi untuk penyulang Tegalsari. Pada frame tersebut terdapat 5 tombol, diantaranya tombol Kaliasin, tombol Basuki Rahmat,tombol Ometraco, tombol Tunjungan, dan tombol Tegalsari. Apabila tombol Tegalsari ditekan, maka akan muncul hasil estimasi tegangan,sudut fasa, beserta error pengukuran penyulang Tegalsari pada sisi kanan, dan tampilan Single Line Diagram penyulang Tegalsari pada sisi kiri.

Tabel 4.5 Hasil Simulasi State Estimasi Penyulang Tegalsari dengan metode EKF

No. Bus	Tegangan			Sudut		
	Estimasi (pu)	Load Flow (pu)	Error (%)	Estimasi (°)	Load Flow (°)	Error (%)
1	1.00000	1.00000	0.0000	0.0000	0.0000	0.00
2	0.99995	0.99995	0.0001	-0.0006	0.0000	0.00
3	0.99977	0.99978	0.0005	-0.0026	0.0000	0.00
4	0.99970	0.99970	0.0000	-0.0034	0.0000	0.00
5	0.99967	0.99967	0.0004	-0.0038	0.0000	0.00
6	0.99954	0.99955	0.0009	-0.0053	-0.0100	47.46
7	0.99950	0.99951	0.0007	-0.0057	-0.0100	43.14
8	0.99934	0.99936	0.0007	-0.0057	-0.0100	24.90
9	0.99934	0.99936	0.0016	-0.0075	-0.0100	24.65
10	0.99921	0.99923	0.0018	-0.0090	-0.0100	9.68
11	0.99919	0.99917	0.0024	-0.0093	-0.0100	7.47
12	0.99895	0.99899	0.0038	-0.0119	-0.0100	19.05
13	0.99885	0.99890	0.0047	-0.0130	-0.0100	29.94
14	0.99884	0.99899	0.0148	-0.0131	-0.0100	31.15
15	0.99877	0.99882	0.0048	-0.0139	-0.0100	38.53
16	0.99876	0.99881	0.0047	-0.0139	-0.0100	39.44
17	0.99875	0.99879	0.0040	-0.0141	-0.0200	29.59
18	0.99874	0.99879	0.0046	-0.0141	-0.0200	29.36
19	0.99875	0.99879	0.0042	-0.0141	-0.0200	29.48
20	0.99875	0.99879	0.0045	-0.0141	-0.0200	29.32

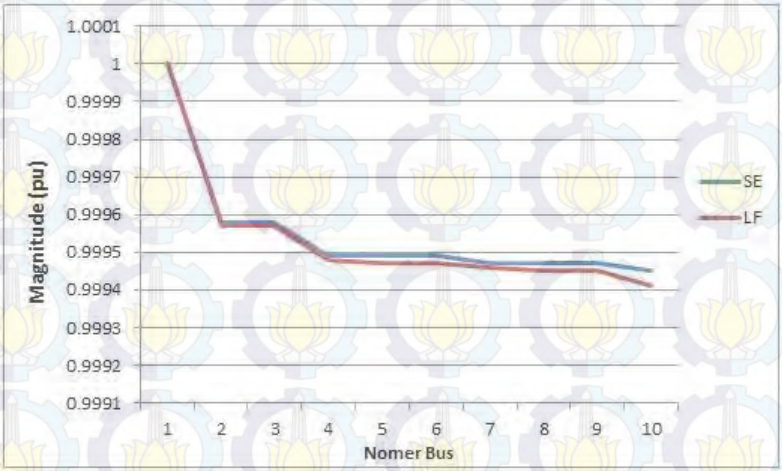
Tabel 4.5 di atas merupakan hasil estimasi tegangan & sudut fasa penyulang Tegalsari. Hasil yang ada pada tabel ini memperjelas tampilan gambar 4.5 pada sisi kanan. Dari tabel di atas, didapatkan tegangan estimasi rata-rata untuk penyulang Tegalsari bernilai 0.999219 pu. Sedangkan untuk tegangan terkecil bernilai 0.99874 pu. Tegangan paling besar selain magnitude tegangan pada bus utama bernilai 0.99995 pu. Hasil yang didapatkan ini tidak undervoltage, karena nilai magnitude tegangan berkisar pada nilai 0.99 pu.

Untuk estimasi sudutnya bernilai besar karena data pembanding yang digunakan untuk validasi pada ETAP merupakan angka pembulatan hanya dua angka di belakang koma, sedangkan data hasil simulasi *JAVA* merupakan pembulatan hingga empat angka di belakang koma sehingga berpengaruh pada komputasinya di program.

4.2 Analisa Grafik State Estimasi

Berdasarkan data hasil simulasi state estimation dari lima penyulang di Surabaya Utara didapatkan hasil estimasi tegangan hampir mendekati hasil analisis pada *software* ETAP. *Error* magnitude tegangan rata-rata berkisar antara 0.0016 – 0.0071%.

4.2.1 Penyulang Kaliasin



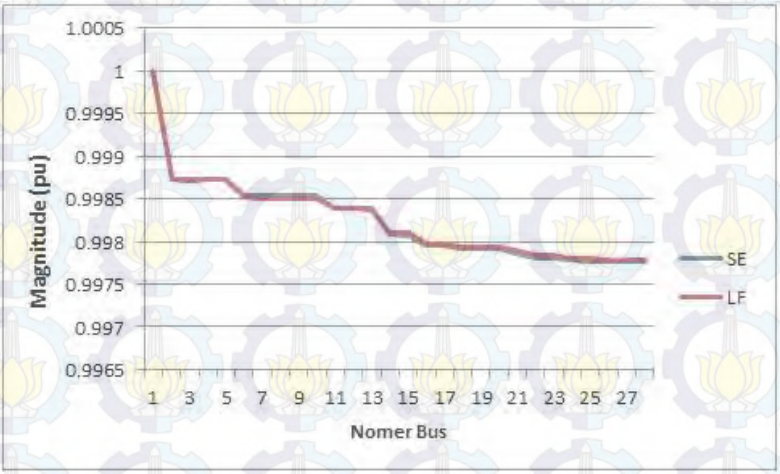
Gambar 4.6 Grafik analisis hasil SE penyulang kaliasin

Gambar 4.6 di atas merupakan grafik/kurva Magnitude vs Nomer Bus hasil state estimasi penyulang kaliasin. Pada sumbu X menyatakan urutan Bus dari penyulang kaliasin, terdiri dari bus nomer 1 sampai dengan bus nomer 10. Sumbu Y menyatakan Magnitude tegangan dalam [pu]. Untuk kurva yang berwarna biru merupakan representasi

magnitude tegangan hasil estimasi dengan metode *EKF*. Kurva yang berwarna merah merupakan hasil *Loadflow* dari *software* ETAP.

Dapat dilihat bahwa error yang paling kecil menunjukkan hasil estimasi yang lebih akurat. Error terkecil pada penyulang kaliasin bernilai 0.0009%, error terbesar bernilai 0.0044%, dan error rata-rata bernilai 0.0016%. Hasil ini masih relevan karena masih pada batas error yang diperbolehkan, yaitu 5%.

4.2.2 Penyulang Basuki Rahmat



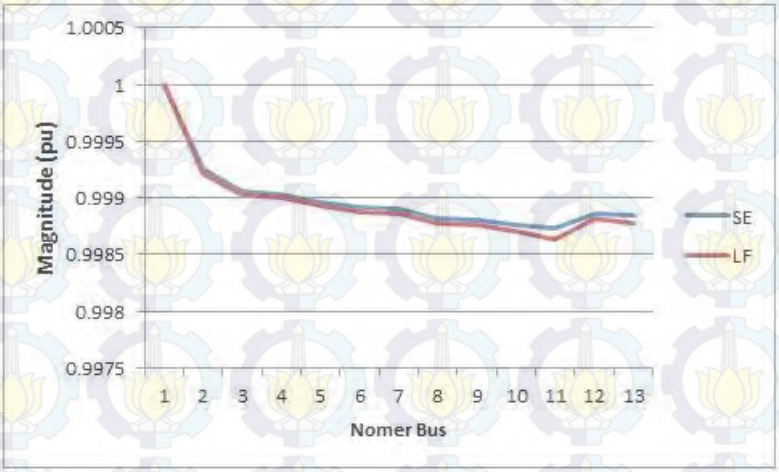
Gambar 4.7 Grafik analisis hasil SE penyulang basuki rahmat

Gambar 4.7 di atas merupakan grafik/kurva Magnitude vs Nomer Bus hasil state estimasi penyulang Basuki rahmat. Pada sumbu X menyatakan urutan Bus dari penyulang Basuki rahmat, terdiri dari bus nomer 1 sampai dengan bus nomer 28. Sumbu Y menyatakan Magnitude tegangan dalam [pu]. Untuk kurva yang berwarna biru merupakan representasi magnitude tegangan hasil estimasi dengan metode *EKF*. Kurva yang berwarna merah merupakan hasil *Loadflow* dari *software* ETAP.

Dapat dilihat bahwa error yang paling kecil menunjukkan hasil estimasi yang lebih akurat. Error terkecil pada penyulang basuki rahmat bernilai 0.0001%, error terbesar bernilai 0.0034%, dan error rata-rata

bernilai 0.0022%. Hasil ini masih relevan karena masih pada batas error yang diperbolehkan, yaitu 5%.

4.2.3 Penyulang Ometraco

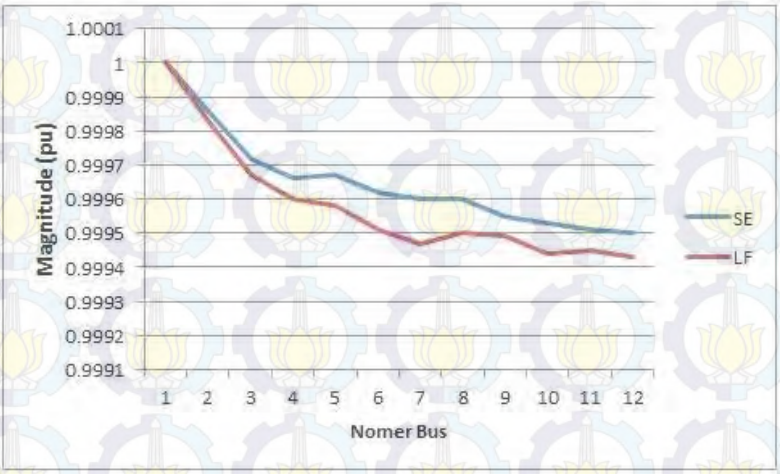


Gambar 4.8 Grafik analisis hasil SE penyulang ometraco

Gambar 4.8 di atas merupakan grafik/kurva Magnitude vs Nomer Bus hasil state estimasi penyulang Ometraco. Pada sumbu X menyatakan urutan Bus dari penyulang Ometraco, terdiri dari bus nomer 1 sampai dengan bus nomer 13. Sumbu Y menyatakan Magnitude tegangan dalam [pu]. Untuk kurva yang berwarna biru merupakan representasi magnitude tegangan hasil estimasi dengan metode *EKF*. Kurva yang berwarna merah merupakan hasil *Loadflow* dari *software* ETAP.

Dapat dilihat bahwa error yang paling kecil menunjukkan hasil estimasi yang lebih akurat. Error terkecil pada penyulang ometraco bernilai 0.0030%, error terbesar bernilai 0.0102%, dan error rata-rata bernilai 0.0043%. Hasil ini masih relevan karena masih pada batas error yang diperbolehkan, yaitu 5%.

4.2.4 Penyulang Tunjungan

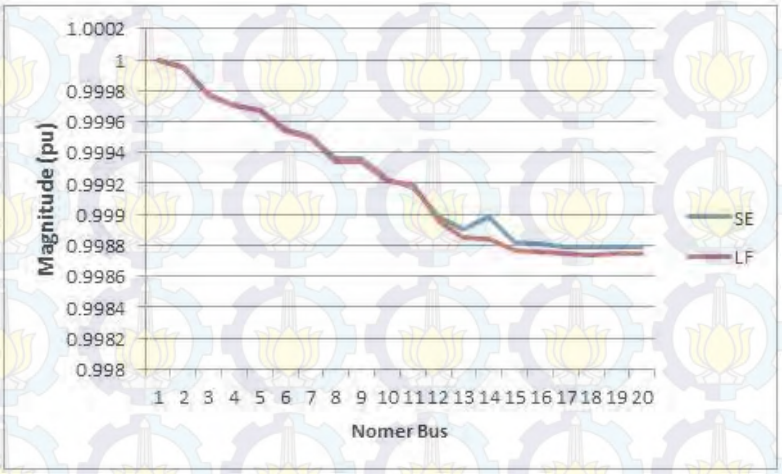


Gambar 4.9 Grafik analisis hasil SE penyulang tunjungan

Gambar 4.9 di atas merupakan grafik/kurva Magnitude vs Nomer Bus hasil state estimasi penyulang Tunjungan. Pada sumbu X menyatakan urutan Bus dari penyulang Tunjungan, terdiri dari bus nomer 1 sampai dengan bus nomer 12. Sumbu Y menyatakan Magnitude tegangan dalam [pu]. Untuk kurva yang berwarna biru merupakan representasi magnitude tegangan hasil estimasi dengan metode *EKF*. Kurva yang berwarna merah merupakan hasil *Loadflow* dari *software* ETAP.

Dapat dilihat bahwa error yang paling kecil menunjukkan hasil estimasi yang lebih akurat. Error terkecil pada penyulang tunjungan bernilai 0.0029%, error terbesar bernilai 0.0133%, dan error rata-rata bernilai 0.0071%. Hasil ini masih relevan karena masih pada batas error yang diperbolehkan, yaitu 5%.

4.2.5 Penyulang Tegalsari



Gambar 4.10 Grafik analisis hasil SE penyulang tegalsari

Gambar 4.10 di atas merupakan grafik/kurva Magnitude vs Nomer Bus hasil state estimasi penyulang Tegalsari. Pada sumbu X menyatakan urutan Bus dari penyulang Tegalsari, terdiri dari bus nomer 1 sampai dengan bus nomer 20. Sumbu Y menyatakan Magnitude tegangan dalam [pu]. Untuk kurva yang berwarna biru merupakan representasi magnitude tegangan hasil estimasi dengan metode *EKF*. Kurva yang berwarna merah merupakan hasil *Loadflow* dari *software* ETAP.

Dapat dilihat bahwa error yang paling kecil menunjukkan hasil estimasi yang lebih akurat. Error terkecil pada penyulang Tegalsari ini bernilai 0.0001%, error terbesar bernilai 0.0148%, dan error rata-rata bernilai 0.0030%. Hasil ini masih relevan karena masih pada batas error yang diperbolehkan, yaitu 5%.

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN & SARAN

5.1 Kesimpulan

Dari hasil pembahasan pada Tugas Akhir ini dapat diambil beberapa kesimpulan, diantaranya:

1. Hasil simulasi *EKF* state estimation dan *Loadflow* dari ETAP menunjukkan bahwa kondisi tegangan pada sistem distribusi 20 KV Surabaya Utara tidak *undervoltage* (normal), yaitu nilai magnitude tegangan berkisar pada nilai 0.99 pu.
2. Penerapan metode *EKF* dapat mengetahui *gross error measurement* peralatan, yaitu untuk error rata-rata bernilai 0.0016% sampai dengan 0.0071%.
3. Penggunaan bahasa pemrograman *JAVA* lebih memudahkan pemrogram atau *programmer* dalam membangun program karena kelebihan *JAVA* sebagai bahasa pemrograman berorientasi objek *(Object Oriented Programming /OOP)*.

5.2 Saran

Adapun beberapa saran dari penulis yang mungkin berguna untuk kedepannya, sebagai berikut :

1. Untuk penelitian ke depan, diharapkan dapat dikembangkan dengan strategi peletakan sensor pengukuran agar sistem lebih handal dan penggunaan sensor dapat dikurangi.
2. Extended Kalman Filter state estimation menggunakan bahasa pemrograman *JAVA* ini kedepannya mungkin dapat dikembangkan dengan *GIS* (*Geographical Information System*) agar dapat mengetahui hasil state estimasi secara *real time*.

DAFTAR PUSTAKA

[1] Penangsang, Ontoseno.*"Analisis Aliran Daya"*, ITS Press, Surabaya,2012.

[2] Gelagaev, Ratmir and Vermayen, Pieter."State Estimation in Distribution Grids", IEEE, 2008.

[3] Kadir, Abdul."Algoritma & pemrograman menggunakan Java", Penerbit ANDI, Yogyakarta, 2012.

[4] Saadat, Hadi., 1999, Power System Analysis, McGraw-Hill Book Co., Singapura

[5] J. Wood, Allen and Wollenberg, Bruce F. *"Power Generation and Operation Control (Third Edition)"*, John Wiley & Sons Inc, United States of America, 1996.

[6] Welch, Greg and Gary Bishop,"An introduction to the Kalman Filter",Department of Computer Science, University of North Carolina at Chapel Hill, April 2004.

[7] Kadir, Abdul."Dasar Pemrograman Java 2", Penerbit ANDI, 2005.

[8] Wolley, NC and Milanovic, JV. "Estimating the Voltage Unbalanced Factor using Distribution System State Estimation", IEEE, 2011.

[9] Gelagaev, Ratmir and Vermayen, Pieter. "State Estimation in Distribution Grids", *IEEE,* 2008.

[10] Sideig A. Dowi, Amar Ibrahim Hamza, Dynamic State Forecasting in Electric Power Networks Journal of Power and Energy Engineering, 2014.

[11] Setiyo Wibowo, Dimas dan Taufik Yuliadi, Arif. "*Pengembangan Modul Praktikum Sistem Monitoring Mikro Scada Tahap II*", Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember, Surabaya, 2013.

[12] Admittance < URL : http://en.wikipedia.org/wiki/Admittance > , Mei, 2013.

[13] Nodal Admittance Matrix
< URL : http://en.wikipedia.org/wiki/Admittance > , Mei, 2013.

Halaman ini sengaja dikosongkan

LAMPIRAN

```
package gui;

import Jama.Matrix;
import
org.apache.commons.math3.compl
ex.ComplexFormat;
/**
  * @author Aisah Nur
Laily_2213106073
  * TA 2015/2016
  * LJ Genap 2013
 */
public void prosesHitung(int
dataKe){
double MVAbase = 10, Vbase =
20;
double Zbase = Math.pow(Vbase,
2) / MVAbase;
//------------LINE
for(int i=0; i<Linedata.length;
i++){
 for(int j=2; j<Linedata[0].length;
j++) {
Linedata[i][j] =
Linedata[i][j]/Zbase;
        }
}
//------------BUS
            for(int i=1;
i<Busdata.length; i++){
Busdata[i][2] = (Busdata[i][2] /
1000) / MVAbase;
            }
//------------INISIALISASI UMUM
double fb[][] = new
double[Linedata.length][1];
for(int i=0; i<Linedata.length;
i++)
fb[i][0] = Linedata[i][0];
double tb[][] = new
double[Linedata.length][1];
for(int i=0; i<Linedata.length;
i++)
tb[i][0] = Linedata[i][1];
double res[][] = new
double[Linedata.length][1];
for(int i=0; i<Linedata.length;
i++)
res[i][0] = Linedata[i][2];
double reak[][] = new
double[Linedata.length][1];
for(int i=0; i<Linedata.length;
i++)
reak[i][0] = Linedata[i][3];
NumberFormat nfImp =
NumberFormat.getInstance();
            NumberFormat nfY =
NumberFormat.getInstance();
nfImp.setMinimumFractionDigits(
4);
nfImp.setMaximumFractionDigits(
4);
nfY.setMinimumFractionDigits(1);
nfY.setMaximumFractionDigits(4)
;
ComplexFormat format = new
ComplexFormat(nfImp);
ComplexFormat format2 = new
ComplexFormat(nfY);
org.apache.commons.math3.compl
ex.Complex imp[] = new
```

```
org.apache.commons.math3.compl
ex.Complex[Linedata.length];
org.apache.commons.math3.compl
ex.Complex y[] = new
org.apache.commons.math3.compl
ex.Complex[Linedata.length];
org.apache.commons.math3.compl
ex.Complex penyebut = new
org.apache.commons.math3.compl
ex.Complex(1, 0);
for(int i=0; i<Linedata.length;
i++)    {
imp[i] = new
org.apache.commons.math3.compl
ex.Complex(res[i][0], reak[i][0]);
            y[i] =
penyebut.divide(imp[i]);
            }
double maxFb = getMax2(fb);
double maxTb = getMax2(tb);
double nbus; int nbranch =
fb.length;
if(maxFb > maxTb) nbus = maxFb;
else nbus = maxTb;
org.apache.commons.math3.compl
ex.Complex ybus[][] = new
org.apache.commons.math3.compl
ex.Complex[(int)nbus][(int)nbus];
for(int i=0; i<ybus.length; i++)
for(int j=0; j<ybus[0].length; j++)
ybus[i][j]=new
org.apache.commons.math3.compl
ex.Complex(0, 0);
//Membentuk Matrik Admitansi
for(int i=0; i<nbranch; i++){
    ybus[(int)fb[i][0]-
    1][(int)tb[i][0]-1] =
    ybus[(int)fb[i][0]-
    1][(int)tb[i][0]-1].subtract(y[i]);
    ybus[(int)tb[i][0]-
    1][(int)fb[i][0]-1] =
    ybus[(int)fb[i][0]-
    1][(int)tb[i][0]-1];
                        }
//Membentuk Diagonal Elements
for(int i=0; i<nbus; i++)
for(int j=0; j<nbranch; j++)
if(fb[j][0] == i+1)
ybus[i][i] = ybus[i][i].add(y[j]);
else if(tb[j][0] == i+1)
ybus[i][i] = ybus[i][i].add(y[j]);
//----------Inisialisasi EKF
double type[][] = new
double[Busdata.length][1];
for(int i=0; i<type.length; i++)
type[i][0] = Busdata[i][1];
double z[][] = new
double[Busdata.length][1];
for(int i=0; i<z.length; i++)
z[i][0] = Busdata[i][2];
double fbus[][] = new
double[Busdata.length][1];
for(int i=0; i<fbus.length; i++)
 fbus[i][0] = Busdata[i][3];
double tbus[][] = new
double[Busdata.length][1];
for(int i=0; i<tbus.length; i++)
tbus[i][0] = Busdata[i][4];
double Ri[][] = new
double[Busdata.length][Busdata.le
ngth];
for(int i=0; i<Ri.length; i++)
   for(int j=0; j<Ri.length; j++)
        if(i==j) Ri[i][j] =
Busdata[i][5];
        else Ri[i][j] = 0;
double V[][] = new
double[(int)nbus][1];
```

```
double del[][] = new
double[(int)nbus][1];
for(int i=0; i<(int)nbus; i++)
{ V[i][0] = 1; del[i][0] = 0; }
double E[][] = new
double[(del.length-1) +
V.length][1];
for(int i=0; i<E.length; i++){
        if(i<del.length-1)
        E[i][0] = del[i+1][0];
        else
    E[i][0] = V[i-(V.length-1)][0];
}
double G[][] = new
double[ybus.length][ybus[0].lengt
h];
double B[][] = new
double[ybus.length][ybus[0].lengt
h];
for(int i=0; i<ybus.length; i++){
        for(int j=0;
j<ybus[0].length; j++){
        G[i][j] =
ybus[i][j].getReal();
        B[i][j] =
ybus[i][j].getImaginary();
        }
}
//double q= 0.1;(std of process)
int sm = (int)(nbus*2-1);//size mat
double [][] Q1 =
Matriks.identity(sm);
double Q2 = 0.0001;   //q^2
double[][] Q =
Matriks.multiply(Q2, Q1);
double [][] P =
Matriks.identity(sm); //cov process
int nvi=0, npi=0, nqi=0, npf=0,
nqf=0;
for(int i=0; i<type.length; i++){
        if(type[i][0] == 1)
        nvi++;
        else if(type[i][0] == 2)
        npi++;
        else if(type[i][0] == 3)
        nqi++;
        else if(type[i][0] == 4)
        npf++;
        else if(type[i][0] == 5)
        nqf++;
}
int vi[][] = new int[nvi][1];
int ppi[][] = new int[npi][1];
int qi[][] = new int[nqi][1];
int pf[][] = new int[npf][1];
int qf[][] = new int[nqf][1];
int viItr=0, ppiItr=0, qiItr=0,
pfItr=0, qfItr=0;
for(int i=0; i<type.length; i++){
        if(type[i][0] == 1) {
        vi[viItr][0] = i+1;
        viItr++;
        }
        else if(type[i][0] == 2) {
        ppi[ppiItr][0] = i+1;
        ppiItr++;
}
        else if(type[i][0] == 3) {
        qi[qiItr][0] = i+1;
        qiItr++;
        }
        else if(type[i][0] == 4) {
        pf[pfItr][0] = i+1;
        pfItr++;
        }
        else if(type[i][0] == 5){
        qf[qfItr][0] = i+1;
        qfItr++;
```

```
            }
}
//ITERASI EKF
int iter=0; double tol=5;
while(tol > 1e-4) {
double[][] h1 = new
double[(int)fbus[vi[0][0]-
1][0]][1];
double[][] h2 = new
double[npi][1];
double[][] h3 = new
double[nqi][1];
double[][] h4 = new
double[npf][1];
double[][] h5 = new
double[nqf][1];
for(int i=0; i<h1.length; i++)
  for(int j=0; j<h1[0].length; j++)
          h1[i][j] = V[i][j];
for(int i=0; i<h2.length; i++)
   for(int j=0; j<h2[0].length; j++)
            h2[i][j] = 0;
for(int i=0; i<h3.length; i++)
   for(int j=0; j<h3[0].length; j++)
            h3[i][j] = 0;
for(int i=0; i<h4.length; i++)
   for(int j=0; j<h4[0].length; j++)
            h4[i][j] = 0;
for(int i=0; i<h5.length; i++)
    for(int j=0; j<h5[0].length; j++)
            h5[i][j] = 0;
for(int i=0; i<npi; i++){
int m = (int)fbus[ (int)ppi[i][0]-1
][0];
for(int k=0; k<nbus; k++){
          h2[i][0] = h2[i][0] + V[m-
1][0] * V[k][0] * ( G[m-1][k] *
Math.cos(del[m-1][0]-del[k][0]) +
B[m-1][k] * Math.sin(del[m-1][0]-
del[k][0]) );
  }
}
for(int i=0; i<nqi; i++){
  int m = (int)fbus[ (int)qi[i][0]-1
][0];
for(int k=0; k<nbus; k++){
          h3[i][0] = h3[i][0] + V[m-
1][0] * V[k][0] * ( G[m-1][k] *
Math.sin(del[m-1][0]-del[k][0]) -
B[m-1][k] * Math.cos(del[m-1][0]-
del[k][0]) );
          }}
int sizeH = h1.length + h2.length +
h3.length + h4.length + h5.length;
double h[][] = new
double[sizeH][1];
for(int i=0; i<h.length; i++){
for(int j=0; j<h[0].length; j++){
          if(i < h1.length)
          h[i][j] = h1[i][j];
else if(i < (h1.length+h2.length))
h[i][j] = h2[i - h1.length][j];
else if(i <
(h1.length+h2.length+h3.length))
h[i][j] = h3[i -
(h1.length+h2.length)][j];
else if(i <
(h1.length+h2.length+h3.length+h
4.length))
          h[i][j] = h4[i -
          (h1.length+h2.length+h3.l
          ength)][j];
          else if(i <
          (h1.length+h2.length+h3.l
          ength+h4.length+h5.lengt
          h))
```

```
            h[i][j] = h5[i -
            (h1.length+h2.length+h3.l
            ength+h4.length)][j];
            }
    }
double r[][] = new
double[z.length][1];
for(int i=0; i<h.length; i++){
            for(int j=0; j<h[0].length;
j++){
            r[i][j] = z[i][j] - h[i][j];
            }
}
// Matriks Jacobian
//=========H11
double[][] H11 = new
double[(int)nvi][(int)nbus-1];
for(int i=0; i<H11.length; i++)
    for(int j=0; j<H11[0].length;
j++)
            H11[i][j] = 0;
int sizeHbar = H11.length +
H21.length + H31.length +
H41.length + H51.length;
int sizeHcol = H11[0].length +
H12[0].length;
double H[][] = new
double[sizeHbar][sizeHcol];
for(int i=0; i<H.length; i++){
for(int j=0; j<H[0].length; j++){
            if(i < H11.length) {
              if(j < H11[0].length)
              H[i][j] = H11[i][j];
            else
            H[i][j] = H12[i][j-
H11[0].length];
            }
            else if(i < H11.length +
H21.length){
if(j < H11[0].length)
H[i][j] = H21[i-H11.length][j];
else
H[i][j] = H22[i-H11.length][j-
H11[0].length];
} }
double Ht[][] = new
double[H[0].length][H.length];
Ht = transposeMatrix(H);
Matrix aaa = new Matrix(Ri);
double RiInv[][] =
aaa.inverse().getArray();
//========State Vector
P= Matriks.add(P, Q);
double[][] P12 = Matriks.add(P,
Ht);
double[][] K1 =
Matriks.multiply(H, P12);
double[][] K2 = Matriks.add(K1,
Ri);
Matrix K12 = new Matrix(K2);
double K2Inv[][] =
K12.inverse().getArray();
double[][] K =
Matriks.multiply(P12,K2Inv);
double[][] dE =
Matriks.multiply(K,r);
//====Menambahkan E dgn dE
E = Matriks.add(E, dE);
for(int i=1; i<del.length; i++){
for(int j=0; j<del[0].length; j++){
            del[i][j] = E[i-1][j];
            }
}
//========V = E(nbus:end)
for(int i=0; i<nbus; i++){
for(int j=0; j<E[0].length; j++){
V[i][j] = E[i+Linedata.length][j];
            }
```

```
}
iter = iter+1;
tol = getMax3(dE);
double P12t[][] = new
double[P12[0].length][P12.length]
;
P12t = transposeMatrix(P12);
double [][] P1 =
Matriks.multiply(P12t,K);
P = Matriks.subtract(P, P1);
System.out.printf("%.5f\n",tol);
}          //end separator while
double a = 0;
a = 1 - V[0][0];
if(V[0][0] < 1){
  V[0][0] = 1;
for(int i=1; i<nbus; i++)
V[i][0] = V[i][0]+a;
}
double b = 0;
b = V[0][0] - 1;
if(V[0][0] > 1){
V[0][0] = 1;
for(int i=1; i<nbus; i++)
V[i][0] = V[i][0]-b;
}
//Magnitude dan sudutnya
double Del[][] = new
double[del.length][del[0].length];
for(int i=0; i<Del.length; i++){
for(int j=0; j<Del[0].length; j++){
     Del[i][j] = 180 / Math.PI *
del[i][j];
}  }
double E2[][] = new
double[V.length][V[0].length+Del
[0].length];
for(int i=0; i<E2.length; i++){
for(int j=0; j<E2[0].length; j++){
          if(j<V[0].length)
          E2[i][j] = V[i][j];
          else
          E2[i][j] = Del[i][j-
V[0].length];
          }
}
//=======Menghitung error
double VEtap[][] = new
double[ETAPdata.length][1];
double DelEtap[][] = new
double[ETAPdata.length][1];
for(int i=0; i<VEtap.length; i++){
for(int j=0; j<VEtap[0].length;
j++){
VEtap[i][j] = ETAPdata[i][1];
      }   }
for(int i=0; i<DelEtap.length;
i++){
for(int j=0; j<DelEtap[0].length;
j++){
DelEtap[i][j] = ETAPdata[i][2];
          }}
double ErrV[] = new
double[ETAPdata.length];
double ErrDel[] = new
double[ETAPdata.length];
for(int m=0; m<ETAPdata.length;
m++){
ErrV[m] = Math.abs(V[m][0]-
VEtap[m][0]) / VEtap[m][0] * 100;
if(Double.isNaN(ErrV[m]))
ErrV[m] = 0;
ErrDel[m] = Math.abs(
(Del[m][0]-DelEtap[m][0]) /
DelEtap[m][0] * 100 );
if(Double.isNaN(ErrDel[m]))
ErrDel[m] = 0;
}
```

BIODATA PENULIS

Penulis bernama lengkap Aisah Nur Laily lahir di Pati, 26 Oktober 1991. Merupakan anak pertama dari dua bersaudara. Penulis menyelesaikan sekolah menengah atas dari SMK Telkom Shandhy Putra Purwokerto pada tahun 2009. Pada tahun yang sama diterima di Politeknik Negeri Semarang jurusan Teknik Elektro. Tahun 2012 penulis menyelesaikan program Diploma 3. Kemudian pada tahun 2012 penulis diterima di salah satu Perusahaan Manufaktur Asing, PT.Jideco Indonesia sebagai *Engineering Staff*. Pada tahun 2013 penulis mengundurkan diri dari PMA tersebut, kemudian melanjutkan pendidikan untuk jenjang sarjana. Pendidikan sarjana ditempuh di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Elektro bidang studi sistem tenaga.