TUGAS AKHIR – RF141501

ANALISIS INVERSI 2D METODE OCCAM UNTUK MEMODELKAN RESISTIVITAS BAWAH PERMUKAAN DATA MAGNETOTELLURIK

SATRIO BUDIRAHARJO
NRP 3712 100 009

Dosen Pembimbing:

Dr. Widya Utama, DEA
Dr. Dwa Desa Warnana
Arif Darmawan S.Si.

JURUSAN TEKNIK GEOFISIKA
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
Surabaya 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – RF141501

ANALYSIS OF 2D OCCAM INVERSION TO RESISTIVITY SUBSURFACE MODELLING IN MAGNETOTELLURIC METHOD

SATRIO BUDIRAHARJO
NRP 3712 100 009

Advisors:

Dr. Widya Utama, DEA
Dr. Dwa Desa Warnana
Arif Darmawan S.Si.

DEPARTMENT OF GEOPHYSICAL ENGINEERING
FACULTY OF CIVIL ENGINEERING AND PLANNING
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

ANALISIS INVERSI 2D METODE OCCAM UNTUK MEMODELKAN RESISTIVITAS BAWAH PERMUKAAN DATA MAGNETOTELLURIK

TUGAS AKHIR

Diajukan Untuk Memenuhi Sebagian Persyaratan
Untuk memperoleh Gelar Sarjana Teknik
Pada
Jurusan Teknik Geofisika
Institut Teknologi Sepuluh Nopember

Oleh:

SATRIO BUDIRAHARJO
NRP. 3712 100 009

Menyetujui:

Dr. Widya Utama, DEA
NIP. 19611024 198803 1001 ( )

Dr. Dwa Desa Warnana M.Si
NIP. 19760123 200003 1001 ( )

Arif Darmawan S.Si
Geophysicist of Elnusa Tbk ( )

Mengetahui,
Ketua Laboratorium Geofisika Eksplorasi Teknik Geofisika
Fakultas Teknik Sipil dan Perencanaan
Institut Teknologi Sepuluh Nopember
Surabaya

Dr. Ayi Syaeful Bahri, S.Si, MT
NIP. 19690906 199702 1001

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini Saya menyatakan bahwa hal-hal yang terdapat dalam keseluruhan penulisan Tugas Akhir ini dengan judul "ANALISIS INVERSI 2D METODE OCCAM UNTUK MEMODELKAN RESISTIVITAS BAWAH PERMUKAAN" adalah benar hasil karya intelektual mandiri dan diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada isi dan daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, maka saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Jakarta, 29 November 2016

[Halaman ini sengaja dikosongkan]

ANALISIS INVERSI 2D METODE OCCAM UNTUK MEMODELKAN RESISTIVITAS BAWAH PERMUKAAN DATA MAGNETOTELLURIK

Nama Mahasiswa	: Satrio Budiraharjo
NRP	: 3712 100 009
Departemen	: Teknik Geofisika ITS
Dosen Pembimbing	: Dr. Widya Utama, DEA
	Dr. Dwa Desa Warnana M.Si
	Arif Darmawan S.Si

ABSTRAK

Program pengolahan data metode Magnetotellurik dikalangan mahasiswa, khususnya inversi 2D sangat jarang ditemukan. Pemodelan struktur resistivitas bawah permukaan menggunakan data Magnetotellurik umumnya saat ini menggunakan inversi 1D yang kemudian dilakukan pseudo-section terhadap resistivitas vertikal untuk menampilkan model dalam bentuk 2D. Untuk medium 2D solusi pemodelannya menjadi lebih kompleks, dikarenakan parameter resistivitas tidak hanya bervariasi terhadap kedalaman namun juga dalam dimensi lateral. Program TGMT2D yang dibuat menggunakan metode inversi Occam yang ditest dengan model sintetik hasil forward modelling, dengan resistivitas lapisan pertama hingga lapisan keempat adalah 50 ohm.m, 1ohm.m, 100 ohm.m dan 500 ohm.m. Dari hasil inversi yang dilakukan didapatkan nilai RMS error 5.15, dengan nilai resistivitas lapisan pertama hingga lapisan keempat adalah 32-79 ohm.m, 1.2-3.9 ohm.m, 2.5-79 Ohm.m, dan 32-200 ohm.m dengan menggunakan *smoothing aspect ratio*.

Kata Kunci : Inversi Occam 2D, Resistivitas, Magnetotellurik

[Halaman ini sengaja dikosongkan]

ANALYSIS OF 2D OCCAM INVERSION TO RESISTIVITY SUBSURFACE MODELLING IN MAGNETOTELLURIC METHOD

Student	: Satrio Budiraharjo
Student ID Number	: 3712 100 009
Department	: Geophysical Engineering ITS
Advisor Lecture	: Dr. Widya Utama, DEA
	Dr. Dwa Desa Warnana M.Si
	Arif Darmawan S.Si

ABSTRACT

Magnetotelluric data processing, especially in 2D inversion is very rare to find in student level. Structural modelling subsurface in magnetotelluric method, generally using 1D modelling inversion which then using pseudo-section with vertical resistivity to built 2D section of subsurface resistivity. In medium 2D, the solution is to be different and make it more complex, it is because resistivity parameter not only different with depth but also affected by latera dimension. TGMT2D program is build by Occam inversion algorithm. To knowing the validity of the TGMT2D program, the model inversion result compared with synthetic model who built before in forward modelling. The resistivity parameter who build is consist of 4 layer, 50 Ohm m in first layer, 1 ohm.m in second layer, 100 Ohm.m in third layer, and 500 Ohm.m in fourth layer. The result of inversion Occam in TGMT2D program showing that 32 – 79 Ohm.m in first layer subsurface, 1.2 – 3.9 Ohm.m in second layer subsurface, 2.5 – 79 Ohm.m in third layer, and 32-200 Ohm.m in last layer. With 5.15 RMS error.

Keywords : Occam inversion 2D, Resistivity, Magnetotelluric method

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kepada Allah SWT karena atas rahmat-Nya laporan Tugas Akhir yang berjudul "ANALISIS INVERSI 2D METODE OCCAM UNTUK MEMODELKAN RESISTIVITAS BAWAH PERMUKAAN" ini dapat terselesaikan.

Pelaksanaan dan penyusunan Laporan Tugas Akhir ini dapat terlaksana dengan baik, tidak terlepas dari bimbingan, bantuan, dan dukungan berbagai pihak. Pada kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Ayah, Ibu, dan semua keluarga berkat dukungan moril maupun materi selama penulis menjalani tugas akhir ini.
2. Bapak Dr. Widya Utama, DEA selaku ketua Departemen Teknik Geofisika ITS dan juga Pembimbing I yang telah memberi bimbingan dan arahan kepada penulis.
3. Bapak Dr. Dwa Desa Warnana selaku pembimbing II di Departemen Teknik Geofisika ITS yang telah memberi bimbingannya.
4. Bapak Arif Darmawan S.Si, selaku pembimbing di PT. Elnusa Tbk yang telah mengajarkan ilmu pemrograman dan metode Magnetotellurik kepada penulis..
5. Seluruh dosen dan staf Jurusan Teknik Geofisika ITS yang telah banyak memberikan ilmu dan membantu secara administrasi selama penulis melakukan studi di Jurusan Teknik Geofisika ITS.
6. Seluruh teman-teman Teknik Geofisika ITS angkatan 2012 atas semangat dan dukungannya.
7. Pak Nefrizal, Pak Irham, PakFirman, Pak Roy, Pak Soleh, Pak Deny, Pak Chandra, Pak Royo, dan Pak Suryadi yang sudah memberi masukan sekaligus bimbingan selama penulis melaksanakan Tugas Akhir di PT. Elnusa Tbk.
8. Semua pihak yang telah membantu yang tidak dapat dituliskan satu per satu.

Penulis menyadari bahwa penulisan dan hasil tugas akhir ini masih banyak kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan. Semoga tugas akhir ini membawa manfaat bagi penulis pribadi maupun bagi pembaca.

Surabaya, 2 Desember 2016

SATRIO BUDIRAHARJO

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Metode Magnetotellurik adalah salah satu metode elektromagnetik pasif yang melibatkan pengukuran fluktuasi medan listrik dan medan magnet alami yang saling tegak lurus dipermukaan bumi yang dapat digunakan untuk mengetahui nilai konduktivitas batuan dibawah permukaan bumi dari kedalaman dangkal hingga puluhan kilometer (Simpson & Bahr, 2005). Metode ini dipublikasikan pertama kali oleh Louis Cagniard pada tahun 1953 (Chave & Weidelt, 2012). Pada perkembangannya, metode Magnetotellurik dapat memodelkan distribusi konduktivitas dalam satu dimensi (1D), dua dimensi (2D), hingga tiga dimensi (3D).

Medium 1D berupa model berlapis horizontal yang variasi nilai resistivitas-nya hanya terhadap kedalaman. Beberapa inversi yang sering digunakan untuk memperoleh parameter resistivitas dari data pada medium 1D ialah inversi Bostick dan inversi Occam (Delgado dkk, 2001). Inversi Bostick merupakan suatu perkiraan yang digunakan untuk memperoleh kurva *apparent resisitivity* berdasarkan penetrasi kedalaman pada medium *halfspace* dengan resistivitas yang sama pada Periode (T), dimana informasi fasa tidak dianggap. Sedangkan, inversi Occam menyelesaikan persamaan non-linier melalui pendekatan linier dengan menambahkan faktor *smoothing* sehingga bentuk penyelesaian metode Occam lebih simpel dibanding metode inversi lain (Constable & Parker, 1987).

Untuk medium 2D solusi pemodelannya menjadi lebih kompleks, hal itu dikarenakan parameter resistivitas tidak hanya bervariasi terhadap kedalaman namun juga dalam dimensi lateral (sumbu y dan sumbu x). Inversi 2D yang umum digunakan di software konvensional (WinGlink) adalah inversi 2D dengan algoritma *nonlinier conjugate gradient* (NLCG), dimana algoritma ini mencari solusi model dengan meminimumkan fungsi objektifnya (Rodi & Mackie, 1998). Namun, software konvensional tersebut merupakan software berbayar yang harganya sangat mahal.

Metode inversi 2D lainnya yang tersedia adalah inversi 2D Occam (Hedlin & Constable, 1990). Inversi occam 2D sebenarnya merupakan pengembangan dari inversi Occam 1D, namun untuk menentukan nilai resistivitas lateral diantara titik pengukuran digunakan pendekatan Elemen Batas (Wannamaker & Stodt, 1985). Untuk masalah

komputasi, inversi Occam2D telah ditulis menggunakan bahasa pemrograman fortran dan dapat diunduh secara gratis untuk kepentingan studi oleh David Myer pada tahum 2006 dan plotting data hasil inversi dikembangkan oleh Kerry Key pada tahun 2005 menggunakan Matlab. Namun, program tersebut belum memiliki *Graphics User Interface* (GUI) dan settingan input data file extension .edi (format data pengolahan Magnetotellurik) untuk menjalankan program tersebut. Hal tersebut menjadi hambatan bagi mahasiswa untuk mempelajari inversi 2D magnetotellurik.

Oleh karena hal itu, penulis pada penelitian Tugas Akhir ini akan membuat GUI yang memudahkan mahasiswa untuk memodelkan penampang 2D resistivitas bawah permukaan hasil inversi 2D Occam.

1.2. Tujuan

Adapun tujuan dari tugas akhir ini yaitu membuat *Graphics User Interface* program inversi 2D Occam menggunakan bahasa pemrograman Python dan melakukan analisis resistivitas penampang bawah permukaan data hasil pemodelan inversi 2D metoda Occam.

1.3. Perumusan Masalah

Adapun perumusan masalah dalam penelitian tugas akhir ini adalah bagaimana mengembangkan *Graphics User Interface* program inversi 2D Occam yang mudah digunakan dan bagaimana analisis resistivitas penampang bawah permukaan hasil pemodelan inversi 2D Occam?

1.4. Batasan Masalah

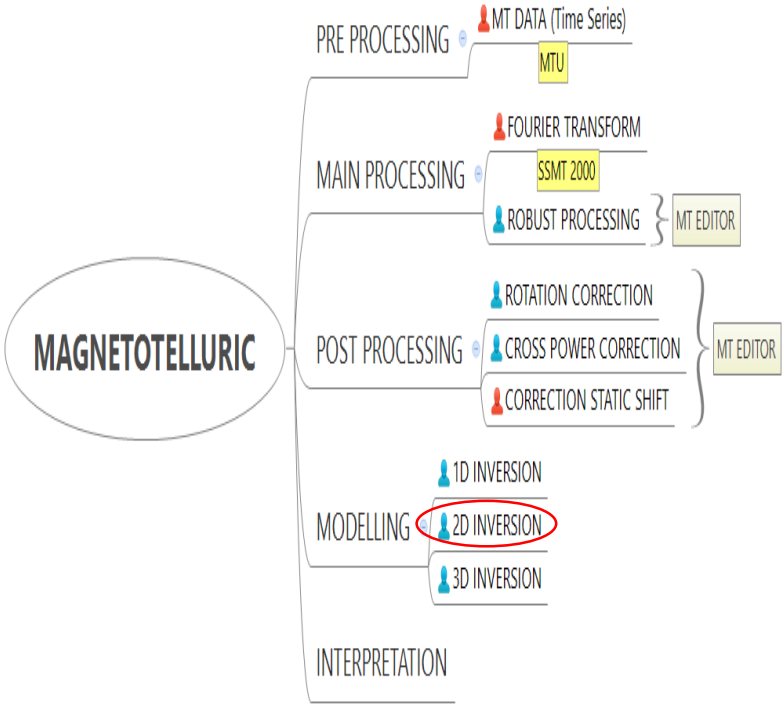
Adapun batasan masalah dalam penelitian tugas akhir ini adalah:

a. Data Magnetotellurik yang digunakan merupakan data sintetik,
b. Data yang didapat merupakan data yang sudah berekstensi .edi,
c. Data Magnetotellurik tidak melalui tahap *static shift correction* dan *rotation correction,*
d. Inversi yang digunakan menggunakan algoritma Occam.

1.5. Manfaat

Manfaat dari penelitian tugas akhir ini, yaitu terciptanya GUI software inversi 2D data Magnetotellurik yang mudah digunakan, sehingga mahasiswa Jurusan Teknik Geofisika dapat menggunakan dan mempelajari pengolahan data inversi 2D Magnetotellurik.

1.6. Roadmap



Gambar 1. 1 Roadmap Penelitian tugas akhir inversi 2D occam

Secara umum, *roadmap* metode pengerjaan data magnetotelurik meliputi tiga tahap, yaitu *pre-processing*, *main-processing*, dan *modelling*.

Penjelasan tiap tahap *roadmap* tersebut ialah sebagai berikut:

Pre-processing : Pada tahap ini, dilakukan seleksi data time series yang berasal dari data lapangan menggunakan program Synchro Time Series Viewer atau sejenisnya.

Main Processing : Di tahap ini terdapat 2 langkah, yang pertama, data time series diubah menjadi domain frekuensi dengan *Fourier Transform* menggunakan program SSMagnetotelluric 2000. Kemudian, dilakukan *robust processing* untuk mengidentifikasi dan menghapus data yang menyimpang oleh noise non-*Gaussian* menggunakan bobot iterative dari residual.

Post Precessing : Pada tahap ini, dilakukan tahap pengolahan mulai dari *correction rotation, cross power selection,* dan *static shift correction* yang secara umum bertujuan untuk menghilangkan kondisi kurva TE dan TM yang terpisah pada jarak tertentu yang diakibatkan oleh faktor distorsi sumber potensial yang terjadi pada kebanyakan metode resistivity, *vertical contact, topografi effec*, dan *near surface effect.* Di tahap ini umumnya dilakukan menggunakan software MT-Editor.

Modelling : Pada tahap ini dipilih model dimensi sesuai kebutuhan target, permodelan 1D biasanya digunakan untuk memberikan gambaran kasar mengenai penyebaran resistivitas bawah permukaan, walaupun dengan batasan tepi yang masih belum akurat dan sempurna. Untuk pemodelan 2D, pemetaan resistivitas bawah permukaan yang dihasilkan lebih akurat, namun memerlukan ketelitian yang cukup tinggi yang dikarenakan besarnya pengaruh efek statik di dalam data dan kejelasan input arah *strike*. Untuk Pemodelan 3D, hasil pemodelan penampang resistivitas yang dihasilkan jauh lebih baik karena dapat menghasilkan daerah target pengeboran dengan lebih akurat, namun untuk melakukan pemodelan ini dibutuhkan komputasi yang sangat kompleks dan spesifikasi hardware yang mumpuni. Area kerja pada penelitian ini terfokus membahas pada pemodelan inversi 2D, yang secara rinci akan dibahas pada bab 3.

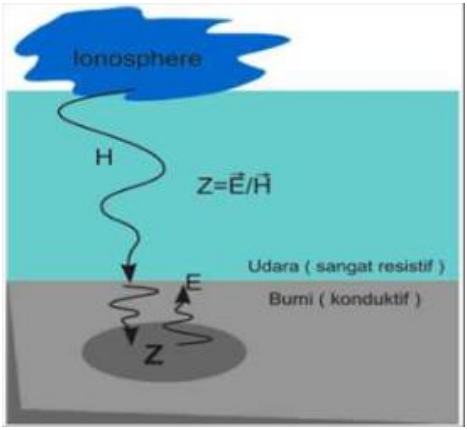
BAB II
TINJAUAN PUSTAKA

2.1. Prinsip Dasar Metode *Magnetotelurik*

Metoda Magnetotellurik merupakan metode elektromagnetik (EM) pasif yang mengukur fluktuasi medan listrik (E) dan medan magnet (H) alami pada arah ortogonal dengan arah permukaan Bumi dengan tujuan untuk menentukan konduktivitas bawah permukaan Bumi dari kedalaman puluhan meter hingga ribuan meter (Simpson dan Bahr, 2005).

Metode Magnetotellurik (magnetotelurik) merupakan salah satu metode geofisika yang dinilai paling baik digunakan dalam eksplorasi panas bumi karena kemampuannya untuk memetakan nilai resistivitas batuan di sistem panas bumi (Oskooi, 2005). Metode magnetotelurik adalah metode elektromagnetik pasif yang melibatkan pengukuran fluktuasi medan listrik dan medan magnet alami yang saling tegak lurus di permukaan Bumi yang dapat digunakan untuk mengetahui nilai konduktivitas batuan di bawah permukaan Bumi dari kedalaman beberapa meter hingga ratusan kilometer (Simpson & Bahr, 2005). Cakupan nilai frekuensi dari medan elektromagnetik alami yang terekam adalah 300 – 0,001 Hz (Daud, 2011).

Konsep gelombang elektromagnetik yang mendasari metode magnetotelurik dapat diwakili oleh Gambar 2.1. Medan elektromagnetik alami (medan elektromagnetik primer) sebagai sumber metode magnetotelurik sampai ke Bumi dengan memiliki variasi terhadap waktu.

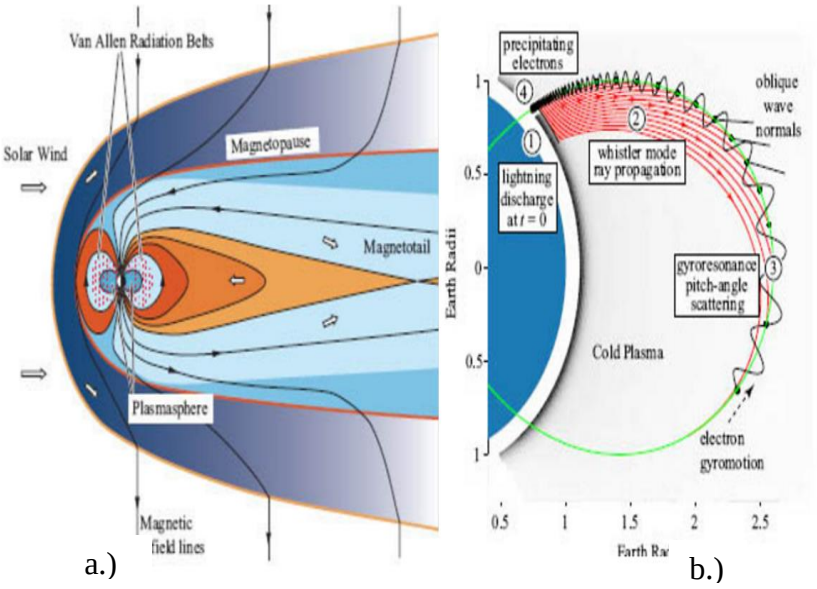


Gambar 2. 1 Prinsip Dasar Metode Magnetotelurik (vozof,1999)

Medan elektromagnetik tersebut menginduksi *Ore body* di bawah permukaan Bumi sehingga menimbulkan *eddy current* (arus telurik) yang menggenerate medan elektromagnetik sekunder. Lalu receiver (RX) yang berada di permukaan menangkap total medan elektromagnetik sebagai penjumlahan dari medan elektromagnetik primer dan medan elektromagnetik sekunder (Daud, 2011).

2.2. Sumber Gelombang *Magnetotelurik*

Bumi memiliki medan magnet yang konstan, namun yang dibutuhkan dalam metode magnetotelurik bukanlah medan magnet yang konstan, melainkan medan magnet yang berubah-ubah terhadap waktu, karena medan magnet yang berubah-ubah terhadap waktu dapat menggenerate medan listrik. Variasi medan elektromagnetik dapat berasal dari petir ataupun interaksi dari *solar wind* dengan lapisan magnetosphere Bumi (Newman et al, 2005).

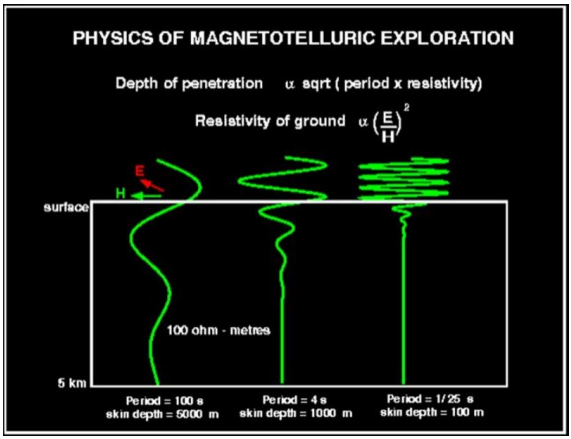


Gambar 2. 2 Dua jenis sumber gelombang magnetotellurik. a) Solar Wind, b) Lightning Discharge (Kaufman, 1981)

Medan elektromagnetik dengan frekuensi lebih dari 1 Hz berasal dari aktivitas meteorologi seperti *lightening discharge* (cahaya petir). Signal petir oleh cahaya dikenal sebagai sferics dan mencakup rentang frekuensi elektromagnetik. Sferic menjalar dalam *waveguide* sebagai gelombang listrik transversal (TE), magnetik transversal (TM), atau gelombang listrik dan magnetik transversal (TEM). Sedangkan interaksi antara *solar wind* dengan lapisan Magnetosphere dan Ionosphere Bumi menghasilkan gelombang electromagnetik dengan frekuensi kurang dari 1 Hz. Solar wind adalah suatu aliran yang kontinu dari plasma, memancarkan sebagian besar proton dan elektron dari Matahari. Pada saat *solar wind* mengenai medan magnet terestrial pada magnetopause, proton dan elektron akan berdefleksi ke arah yang berlawanan sehingga menimbulkan medan listrik (Simpson & Bahr, 2005).

2.3. Penetrasi

Metode magnetotelurik bergantung pada penetrasi medan elektromagnetik yang masuk kedalam Bumi (Green, 2003). Gelombang elektromagnetik dan konduktivitas batuan bumi itu sendiri nantinya akan berpengaruh terhadap penetrasi. Oleh karena domain kedalaman berbanding terbalik dengan domain frekuensi, maka alat yang digunakanpun harus memiliki nilai frekuensi yang spesifik, yaitu yang memiliki frekuensi rendah. Semakin kecil frekuensi dari alat yang digunakan, maka akan semakin dalam penetrasi yang diperoleh (Simpson, 2005). Akibat dari dalamnya penetrasi tersebut, maka konsekuensinya adalah proses perekaman data menjadi semakin lama.



Gambar 2. 3 Hubungan waktu pengukuran yang berhubungan dengan atenuasi Frekuensi terhadap penetrasi kedalaman (wannamaker, 1996)

Besaran *skin depth* digunakan untuk memperkirakan kedalaman penetrasi gelombang elektromagnetik. Adapun *skin depth* dalam metode magnetotelurik memenuhi persamaan berikut ini,

$$\delta = \sqrt{\frac{2\rho}{\omega\mu}} = (\pi f \mu \sigma)^{-1/2} \cong 0.503\sqrt{\frac{\rho}{f}}\,(\text{km}) \qquad (2.1)$$

Dari persamaan 2.1, terlihat bahwa *skin depth* tidak hanya dipengaruhi oleh besarnya frekuensi alat yang kita gunakan, tetapi

factor resistivitas formasi batuan juga turut mempengaruhi. Semakin besar frekuensi alat yang digunakan, maka penetrasi akan semakin dangkal. Namun, ketika frekuensi alat yang digunakan diperkecil, maka penetrasi yang dihasilkan akan semakin dalam. Sementara itu, dengan frekuensi alat yang sama, semakin besar nilai resistivitas formasi batuan yang ada dibawah lapisan bumi maka hasil penetrasi yang diperoleh akan semakin dalam, begitu pula sebaliknya. Besar kecilnya nilai penetrasi bergantung oleh nilai resistivitas batuan, hal itu dikarenakan lapisan yang memiliki nilai resistivitas rendah akan cenderung lebih mudah mengalirkan arus dibandingkan dengan lapisan yang lebih resistif.

Parameter yang diukur dalam survey magnetotelurik adalah medan listrik dan medan magnet diwilayah pengukuran (Daud, 2010). Sementara itu, parameter yang dianalisa dalam metode magnetotelurik adalah *apparent resistivity* dan *phase.* Di dalam teori elektromagnetik, medan listrik selalu tegak lurus terhadap medan magnet. Perbandingan antara medan magnet dan medan listrik dinamakan impedansi. Innpedansi inilah yang menganding informasi mengenai nilai resistivitas medium pada kedalaman tertentu. (Kauffman & Keller, 1981)

2.4. Asumsi Medium 2D

Untuk medium 2D, dimana sistem dibuat dalam koordinat Cartesian dengan sumbu x dan y, terdapat 2 impedansi yang dianalisis. Dalam medium 2D, berlaku $Z_{xx} = Z_{yy} = 0$ dan $Z_{xy} \neq Z_{yx}$

$$Z = \begin{bmatrix} 0 & Zxy \\ Zyx & 0 \end{bmatrix} \tag{2.2}$$

Dengan,

$$Zxy = \frac{E_x}{H_y}$$

$$Zyx = \frac{E_y}{H_x}$$

Z_{xy} dinamakan mode TE dan Z_{yx} dinamakan mode TM. Data TE merupakan data yang terekam ketika komponen medan listrik sejajar dengan *strike* sedangkan data TM merupakan data yang terekam ketika komponen medan magnet sejajar dengan *strike.*

Persamaan mode TE dan TM pada dasarnya diturunkan dari persamaan Maxwell. Adapun penjabarannya adalah sebagai berikut (Simpson, 2005):

$$\nabla x\, B = \sigma\mu E \tag{2.3}$$

$$\nabla x\, E = -\frac{\partial B}{\partial t} = -i\omega B \tag{2.4}$$

Persamaan 2.3 dan 2.4 dijabarkan, menjadi:

$$\frac{\partial B_z}{\partial_y} - \frac{\partial B_y}{\partial_z} = \mu\sigma E_x \; ; \frac{\partial B_x}{\partial_z} - \frac{\partial B_z}{\partial_x} = \mu\sigma E_y \; ; \frac{\partial B_y}{\partial_x} - \frac{\partial B_x}{\partial_y} = \mu\sigma E_z \tag{2.5}$$

$$\frac{\partial E_z}{\partial_y} - \frac{\partial E_y}{\partial_z} = i\omega B_x \; ; \frac{\partial E_x}{\partial_z} - \frac{\partial E_z}{\partial_x} = i\omega B_y \; ; \frac{\partial E_y}{\partial_x} - \frac{\partial E_x}{\partial_y} = i\omega B_z \tag{2.6}$$

Dalam kasus 2D, tidak ada variasi dalam arah x, oleh karena itu, $\frac{\partial}{\partial x} = 0$. Dengan demikian, persamaan 2.5 dan 2.6 disederhanakan menjadi:

$$\frac{\partial B_z}{\partial_y} - \frac{\partial B_y}{\partial_z} = \mu\sigma E_x \; ; \frac{\partial B_x}{\partial_z} = \mu\sigma E_y \; ; -\frac{\partial B_x}{\partial_y} = \mu\sigma E_z \tag{2.7}$$

$$\frac{\partial E_z}{\partial_y} - \frac{\partial E_y}{\partial_z} = -i\omega B_x \; ; \frac{\partial E_x}{\partial_z} = -i\omega B_y \; ; -\frac{\partial E_x}{\partial_y} = -i\omega B_z \tag{28}$$

TM merupakan kondisi dimana medan magnet sejajar dengan *strike* dan medan listrik tegak lurus terhadap *strike.* Jika diasumsikan arah *strike* berada pada arah x, maka medan magnet berada pada arah x dan medan listrik berada pada arah y dan z. Persamaan yang memenuhi adalah :

$$\frac{\partial E_z}{\partial_y} - \frac{\partial E_y}{\partial_z} = -i\omega B_x \; ; \frac{\partial B_x}{\partial_z} = \mu\sigma E_y \; ; -\frac{\partial B_x}{\partial_y} = \mu\sigma E_z \tag{2.9}$$

Dalam kasus 2D, kepastian arah *strike* sangat diperlukan. Ketika arah *strike* diketahui, maka data magnetotelurik selanjutnya dapat dirotasikan terhadap arah *strike* tersebut. Proses perotasian ini dapat dinyatakan kedalam matriks berikut ini.

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{2.10}$$

Sehingga E = RE' dan H = RH'

Tensor impedansi terotasi dirumuskan dengan:

$$Z^* = R \cdot Z \cdot R^T \tag{2.11}$$

Dengan R^T merupakan matriks transpose dari R

Dengan demikian, dalam medium 2D, nilai *apparent resistivity* dapat dirumuskan dalam persamaan berikut.

$$\rho_{xy} = \rho_{TE} = \frac{1}{\omega\mu}\left|Z_{xy}\right|^2 \tag{2.12}$$

$$\rho_{yx} = \rho_{TM} = \frac{1}{\omega\mu}\left|Z_{yx}\right|^2 \tag{2.13}$$

Dengan;

ω =Frekuensi angular

μ =Permeabilitas

Persamaan 2.12 dan persamaan 2.13 menunjukkan nilai *apparent resistivity* pada kondisi TE dan TM secara berturut-turut. (Rodi & Mackie, 2001)

2.5. Algoritma Inversi MT

Kebanyakan proses geofisika dapat diformulasikan dalam bentuk

$$d_i = \int_0^z K_i(z)\, p(z) dz \tag{2.14}$$

Dengan d_i adalah respon data yang terukur, *p(z)* adalah parameter model, yaitu suatu fungsi yang berkaitan dengan parameter fisik yang dikehendaki, dan K_i adalah matriks Kernel. Data kernel adalah data yang menjelaskan hubungan antara data yang diukur dengan parameter model (Constable & De Groot, 1990).

Untuk pendekatan komputasi, persamaan diatas disederhanakan dengan p(z) dz=m dan K_i (z) = $G_{i,}$ sehingga persamaan diatas berubah menjadi:

$$d_i = \Sigma G_{ij} m_j \tag{2.15}$$

Matriks Kernel merupakan solusi untuk menyelesaika permasalahan inversi. Akan tetapi, karena permasalahan magnetotelurik tidak bias didekati dengan solusi diatas, maka

penyelesaiannya tidak dapat dilakukan dengan cara tersebut. Alternatif yang dapat dilakukan untuk menyelesaikan permasalahan inversi magnetotelurik adalah dengna meminimalisir persaman fungsi *unconstraint* yang ada melalui pendekatan iterasi. (Wannamaker & De Lugao, 1996)

Didalam data magnetotelurik, terdapat 2 jenis fungsi utama *unconstraint* yang harus diminimalisir melalui pendekatan iterasi untuk menghasilkan model yang fit dengan data yang ada. Salah satunya adalah fungsi berikut:

$$U(m,\lambda) = \lambda^{-1}\{(d - F[m])^T C_d^{-1}(d - F[m]) - x^2\} + (m - m_0)^T C_m^{-1}(m - m_0) \quad (2.16)$$

Dimana *m* adalah model resistivity dalam dimensi m, m_0 adalah *prior* model, C_m adalah model matriks *Covariance*, *d* adalah data observasi dengan dimensi N, F(m) adalah Forward modeling, Cd adalah data matriks *Covariance*, X^2 adalah level misfit yang diharapkan, dan λ^{-1} adalah Langrange multiplier.

Fungsi yang lain adalah W(m), biasa disebut sebagai fungsi objectif, persamaan fungsinya dapat ditulis sebagai berikut:

$$W(m) = \lambda^{-1}\{(d - F[m])^T C_d^{-1}(d - F[m]) + (m - m_0)^T C_m^{-1}(m - m_0)\} \quad (2.17)$$

Dimana λ= parameter yang mengontrol data eror. Untuk λ yang besar, akan dihasilkan model yang *smooth*. Sementara itu model yang kasar akan dihasilkan oleh λ yang kecil. Terkadang, di beberapa algoritma, nilai λ telah ditetapkan. Akan tetapi, tidak jarang pula algoritmal inversi yang tidak menetapkan nilai λ tetapi justru memulai inversi dengan λ yang besar hingga kemudian nilainya berkurang menjadi kecil. Hal ini dilakukan sebagai langkah untuk mengurangi lamanya waktu komputasi. (Srigutomo, 1997)

Inti dari penyelesaian masalah inversi adalah mencari model yang *smooth* yang berkriteria yaitu, norm yang kecil dan misfit yang masih dapat diterima (dalam artian tidak melebihi toleransi yang ditentukan). Hal ini menunjukan kemiripan dengan data yang ada. Secara teknis, pencarian model dilakukan dengan menjalankan nilai λ yang berbeda yang telah ditetapkan ditiap inversi. Kemudian, tiap running yang sedang berjalan harus diakhiri ketika misfit telah

mencapai level yang diinginkan. Selanjutnya *norm* akan dihitung dan dibandingkan untuk mencari model mana yang memiliki *norm* terkecil. (Oskooi, 2005)

2.6. Occam Inversi

Secara umum sebagian besar permasalahan inversi dalam geofisika adalah inversi non-linier. Meskipun demikian pada beberapa kasus, permasalahan inversi dapat dipilih atau dibuat menjadi linier ataupun non-linier bergantung pada parameterisasi model yang dipilih. Hubungan antara data dengan parameter model secara umum dapat dinyatakan oleh persamaan berikut:

$$d=g\,(m) \tag{2.18}$$

Persamaan tersebut dapat pula digunakan untuk menyatakan hubungan antara data dengan parameter model yang direpresentasikan oleh suatu fungsi non-linier. Dalam hal ini g adalah suatu fungsi pemodelan kedepan (forward modeling) yang merupakan fungsi non-linier dari parameter model. Fungsi g dinyatakan dalam notasi vektor untuk menyatakan adanya komponen yang berasosiasi dengan komponen data.

Misalkan solusi inversi dari persamaan (2.18) adalah model m yang merupakan suatu model awal m_o yang diperturbasi dengan Δm agar diperoleh kecocokan yang lebih baik antara respons model tersebut dengan data:

$$m = m_0 + \Delta m \tag{2.19}$$

$$d = g(m_0 + \Delta m) \tag{2.20}$$

Jika persamaan (2.20) dituliskan kembali dalam bentuk komponennya maka diperoleh:

$$d_i = g_i(m_0^{(j)} + \delta m_j) \tag{2.21}$$

dimana I = 1, 2, … N dan j = 1, 2, … M dengan N dan M masing-masing adalah jumlah data dan jumlah parameter model.

Ekspansi Taylor orde pertama fungsi g(m) disekitar suatu model awal m_o dengan menggunakan noteasi komponen seperti persamaan (2.21) menghasilkan:

$$g_i\left(m_0^{(j)} + \delta m_j\right) \approx g_i\left(m_0^{(j)}\right) + \frac{\partial g_i}{\partial m_j} \left|\partial m_j + O(\partial m_j)\right. \tag{2.22}$$

Dimana $0(\delta m_j)$ adalah suku sisa yang melibatkan turunan orde ke-dua dan orde-orde lebih tinggi. Hasil substitusi persamaan (2.22) dengan mengabaikan suku sisa tersebut adalah sebagai berikut:

$$d = g_i\left(m_0^{(j)}\right) + \frac{\partial g_i}{\partial m_j} \left|\partial m_j\right. \tag{2.23}$$

Suku ke-dua pada ruas kanan persamaan (2.23) adalah komponen turunan parsial fungsi g(m) terhadap suatu elemen parameter model m yang membentuk matriks Jacobi berikut:

$$J_{ij} = \frac{\partial g_i}{\partial m_j} \tag{2.24}$$

Selanjutnya, subtitusi dan pengaturan kembali persamaan (2.18) menghasilkan:

$$d - g(m_0) = \underline{J}_0 \Delta m_0 \; atau \; \Delta d_0 = \underline{J}_0 \Delta m_0 \tag{2.25}$$

dimana $\underline{J}_0$ adalah matriks Jacobi yang dievaluasi pada $m=m_0$, dimana simbol _ menyatakan bahwa bilangan tersebut adalah bilangan kompleks. Dengan menganggap $\Delta d_0 = d - g(m_0)$ maka persamaan (2.25) mirip dengan persamaan yang berlaku pada hubungan linier antara data dengan parameter model, yaitu $d = \underline{G}m$

Dalam hal ini dapat dikatakan bahwa data digantikan oleh perturbasi data dan model menjadi perturbasi model. Sementara itu matriks kernel digantikan oleh matriks Jacobi yang menyatakan sejauh mana data prediksi berubah sebagai akibat dari perubahan atau perturbasi model. Oleh karena itu matriks Jacobi sering pula disebut sebagai matriks sensitivitas (sensitivity matrix).

Kemiripan bentuk persamaan (2.25) dengan persamaan yang menyatakan hubungan linier antara data dengan parameter model $d = \underline{G}m$ mengindikasikan hubungan linier antara $\Delta d_0 = d - g(m_0)$ dengan Δm_0. Berdasarkan analogi, solusi inversi dalam bentuk Δm_0 dari suatu permasalahan yang dapat dinyatakan oleh persamaan (2.25) adalah sebagai berikut:

$$\Delta m_0 = \left[\underline{J}_0^T \underline{J}_0\right]^{-1} \underline{J}_0^T (d - g(m_0)) \tag{2.26}$$

Persamaan (2.22) pada dasarnya menyatakan perturbasi yang diperlukan terhadap suatu model awal m0 agar diperoleh model yang lebih baik, yaitu $m = m_0 + \Delta m$. Respons model m diharapkan lebih fit dengan data.

Mengingat sifat non-linier dari fungsi yang menghubungkan data dengan parameter model (pemodelan kedepan) maka pendekatan orde pertama tersebut tidak dapat langsung menghasilkan model optimum. Oleh karena itu proses perturbasi model dilakukan terhadap model awal m0 secara iteratif menggunakan persamaan (2.26) sampai diperoleh konvergensi menuju solusi optimum.

Untuk memperoleh solusi inversi atau model optimum diperlukan perturbasi secara iteratif suatu model awal m0. Dengan demikian pada iterasi ke-(n+1) perturbasi dilakukan terhadap model hasil iterasi sebelumnya dengan menggunakan persamaan berikut:

$$m_{n+1} = m_n + \left[\underline{J}_{\mathrm{n}}^T \underline{J}_{\mathrm{n}}\right]^{-1} \underline{J}_{\mathrm{n}}^T (d - g(m_{\mathrm{n}})) \tag{2.27}$$

Dalam penyelesaian permasalahan non-liner menggunakan pendekatan linier ada beberapa metode yang dikenal yaitu metode gradient, metode Gauss-Newton, metode Levenberg-Marquardt dan

metode Occam. Metode gradien sangat lamban jika dibandingkan dengan metode Gauss-Newton, terutama jika sudah dekat dengan solusi atau nilai minimum fungsi obyektif. Hal ini disebabkan oleh harga gradien yang semakin kecil. Di lain pihak, metode gradien cukup efektif pada saat awal iterasi dimana metode Gauss-Newton dapat mengalami overshoot. Oleh karena itu kombinasi yang tepat antara kedua metode dapat memperbaiki kinerja tiap metode yang diterapkan secara terpisah. Kombinasi dilakukan dengan menerapkan metode gradien pada saat awal iterasi yaitu saat masih jauh dari solusi, kemudian semakin dekat dengan solusi digunakan metode quasi-Newton.

$$L = \begin{bmatrix} 0 & 0 & \\ -1 & 1 & \\ & -1 & 1 \end{bmatrix}_{nxn} \tag{2.28}$$

Sehingga parameter yang digunakan menjadi (Aster, 2005):

$$\hat{d} = d - g(m_n) + \underline{J}_n m_n \tag{2.29}$$

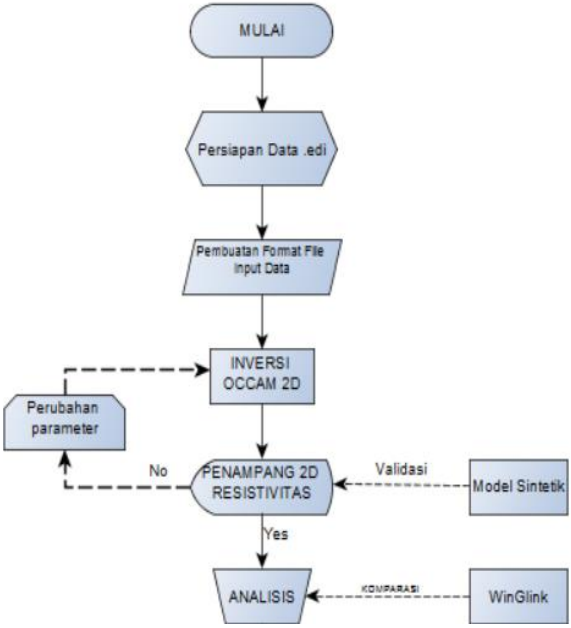
$$m_{n+1} = \left[\underline{J}_n^T \underline{J}_n + \alpha^2 L^T L\right]^{-1} \underline{J}_n^T \hat{d} \tag{2.30}$$

Metode kombinasi tersebut dikenal sebagai metode Levenberg-Marquardt. Sedangkan metode Occam merupakan pengembangan dari metode Levenberg-marquardt dengan menambahkan parameter delta untuk smoothing berdasarkan regulasi tikhonov orde 1 (Constable & De Groot, 1990).

BAB III
METODOLOGI

3.1. Tahapan Penelitian

Tahapan penelitian dalam tugas akhir ini dimulai dengan membuat format file yang digunakan sebagai input data pada program inversi 2D Occam hingga melakukan analisis resistivitas untuk mengetahui struktur bawah permukaan daerah titik stasiun pengukuran. Secara garis besar *workflow* penelitian tugas akhir ini adalah sebagai berikut:



Gambar 3. 1 Workflow penelitian tugas akhir pembuatan software inversi 2D occam

Pada tahap pembuatan format file input data, digunakan bahasa pemrograman Python untuk membuat input data file Startup.txt, file

Data.txt, Model.txt, dan Mesh.txt. Penjelasan mengenai input data tersebut akan dibahas lebih detail pada subbab ini.

Pada tahap inversi, digunakan program inversi 2D Occam menggunakan bahasa pemrograman Fortran (Myer, 2006). Program inversi 2D Occam ini membutuhkan input data 4 format file yang dihasilkan pada tahap sebelumnya untuk dilakukan forward modelling dan iterasi. Output yang dihasilkan pada proses inversi 2D Occam ini ada lah file berekstensi *iter.

Selanjutnya pada tahap plotting nilai hasil inversi, digunakan program matlab (Key, 2005). Hasil plotting ini berupa penampang resistivitas 2D per kedalaman. Penampang resistivitas 2D tersebut kemudian divalidasi oleh model sintetik dengan membandingkan struktur bawah permukaan model sintetik dan data hasil inversi 2D Occam.

Apabila struktur bawah permukaan sudah mendekati model sintetik sebenarnya, maka selanjutnya penampang resistivitas hasil inversi 2D Occam tersebut dibandingkan dengan hasil inversi pada software konvensional, yaitu WinGlink. Perbandingan penampang resistivitas 2D antara inversi Occam dan inversi NLCG (algoritma inversi yang digunakan dalam software WinGlink) bertujuan untuk memaksimalkan parameter input program inversi Occam 2D yang digunakan dalam penelitian tugas akhir ini.

3.2. Persiapan Data

Data yang digunakan merupakan data hasil pembuatan model sintetik yang dilakukan oleh PT. Elnusa Tbk yang berupa file dengan extension *edi dengan jumlah data yang digunakan dalam penelitian sebanyak 20 stasiun titik dengan offset 1000 m untuk tiap stasiun titik pada satu lintasan yang sama.

```
>HEAD
DATAID="MT-181"
ACQBY=""
FILEBY=""
ACQDATE=10/01/16
FILEDATE=10/07/16
PROSPECT="3DFOR"
LOC="3DFOR"
LAT=82:42:46.0087
LONG=117:47:18.1479
ELEV=0
STDVERS="SEG 1.0"
PROGVERS="WINGLINK EDI 1.0.22"
PROGDATE=04/23/02
MAXSECT=999
EMPTY=1.0e+32
```

Gambar 3. 2 Contoh header data file extension .edi

Header pada gambar 3.2 berisi tentang informasi yang berkaitan dengan pengambilan titik pengukuran akuisisi Magnetotellurik. Terdapat parameter-parameter yang ditampilkan dalam satu data stasiun pengukuran dengan extension *edi ini. Parameter tersebut diantaranya adalah Frekuensi, Impedansi Rotasi, Real Impedansi, Imajiner Impedansi, Resistivitas Semu, Fasa, dan Tipper.

3.3. Pembuatan Input Data

Input data merupakan 4 file yang meliputi Startup, Data, Model, dan Mesh. Input data tersebut saling berhubungan, sehingga kesalahan input informasi di satu file saja dapat menyebabkan program inversi tidak berjalan (Program inversi 2D Occam memiliki ketentuan format yang harus diikuti).

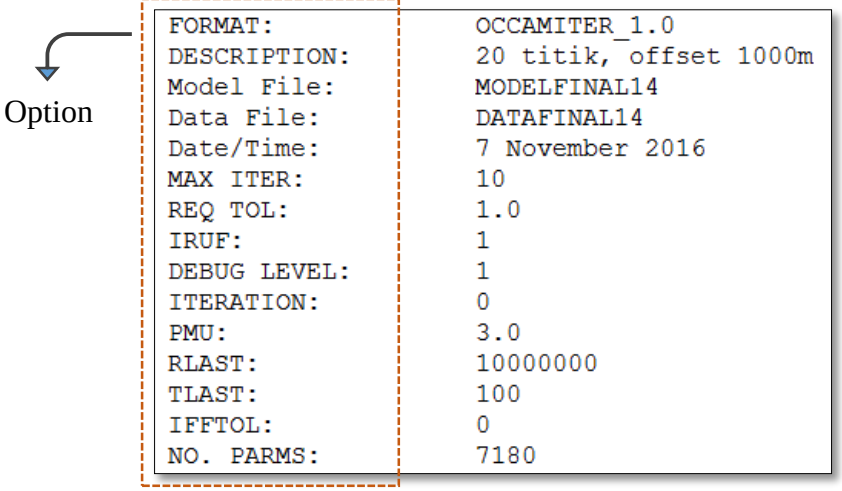
3.3.1. Startup

File ‘startup’ merupakan jenis file yang mengatur parameter dalam melakukan inversi 2D Occam. Pada file ini, *header* tersusun dari tiap line yang memuat informasi *option* dan nilai dari *option*. Perlu dicatat

bahwa perbedaan huruf kapital pada file ini harus sangat diperhatikan. File ‘startup’ terbagi menjadi 2 format file, ‘Startup_1.0’ dan ‘Startup_Flex’, kedua file ini dapat digunakan untuk mengatur parameter dalam melakukan inversi 2D Occam, namun memiliki header dan *option* yang berbeda.

3.3.1.1. Startup_1.0

File ‘Startup_1.0’ terdiri dari 15 line *header* yang memuat informasi *setting* inversi dan informasi dari ‘data file’ dan ‘model file’ yang akan digunakan untuk proses inversi.



Gambar 3. 3 Header file Startup_1.0

Untuk setiap format line diatas harus diberi minimal 18 *space* karakter dari *option* parameter dan tanda titik dua (:). Line ‘format’ memuat informasi nama program yang akan digunakan dalam program inversi 2D Occam. Untuk itu, line ‘format’ secara *default* harus diisi ‘OCCAMITER_1.0’. Line ‘Description’ dapat diisi dengan keterangan tambahan yang memudahkan *user* dalam mengklasifikasikan proses invesi 2D Occam. Line ‘Model File’ memuat informasi tentang nama model file yang digunakan. Line ‘Data File’ memuat informasi tentang nama data file yang digunakan.

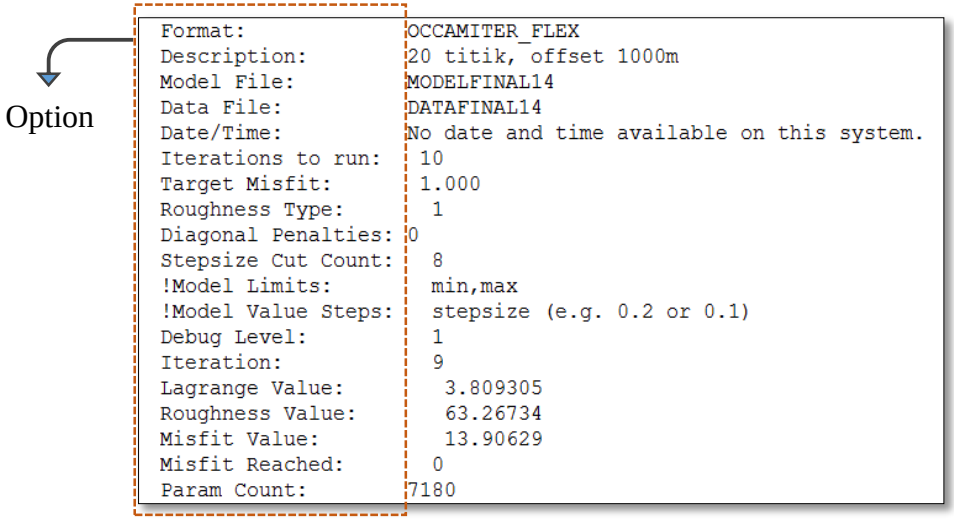
Line ‘MAX ITER’ memuat informasi tentang nilai iterasi maksimal yang digunakan dalam proses inversi 2D Occam. Line ‘Date/time’ memuat keterangan waktu saat melakukan inversi, *option* ini tidak memiliki ketentuan dalam format input tanggal. Line ‘REQ TOL’ memuat informasi toleransi error yang digunakan dalam proses inversi 2D Occam. Line ‘IRUF’ memuat informasi type roughness yang digunakan pada proses inversi 2D.

Line ‘DEBUG LEVEL’ dapat diisi dengan nilai ‘0’, ‘1’, atau ‘2’. Input nilai ‘0’ akan meminimalisir jumlah display ketika inversi sedang berjalan, input ‘1’ memperlihatkan *display* untuk tiap frekuensi saat proses inversi sedang berjalan, input ‘2’ akan mengubah cara kerja dari inversi 2D Occam dalam menemukan model sebenarnya. Jadi semakin tinggi input nilai yang dimasukkan akan berpengaruh kepada waktu dalam melakukan iterasi program inversi 2D Occam.

Line ‘ITERATION’ memuat informasi *start* iterasi yang dilakukan dalam proses inversi. Line ‘PMU’ memuat nilai dari parameter lagrange yang digunakan dalam proses inversi. Line ‘Rlast’ memuat informasi nilai *Roughness* (kekasaran) maksimal dari proses inversi. Line ‘ Tlast’ memuat informasi misfit yang dihasilkan dalam proses inversi. Line ‘IFFTOL’ memuat informasi apakah iterasi terakhir yang dilakukan sudah mencapai target toleransi (RMS misfit) atau belum, ditandai nilai ‘0’ untuk jika belum dan ‘1’ jika sudah. Line ‘No. Parms’ memuat jumlah model block yang terdapat pada model file. Dibawah line ‘No. Parms’ memuat list nilai resistivitas model awal sebanyak input yang dimasukkan pada ‘No. Parms’.

3.3.1.2. Startup_Flex

File ‘Startup_Flex’ pada dasarnya memuat informasi yang sama dengan file ‘Startup_1.0’’ namun memiliki *option* parameter yang lebih lengkap untuk mengatur proses inversi 2D Occam. File ‘Startup_Flex’ terdiri dari 19 line *header* yang memuat informasi setting inversi, hasil inversi, dan juga informasi dari ‘data file’ dan ‘model file’.



Gambar 3. 4 Header file Startup_flex

Terdapat beberapa *option* yang memiliki fungsi dan penulisan yang identik dengan *option* pada file 'Startup_1.0' yang tidak dibahas lagi pada subbab ini. Line 'Roughness type' sama dengan line 'IRUF' yang secara default diberi input nilai '1'.

Line 'Diagonal Penalties' memuat keterangan apakah occam menghitung *roughness penalties* secara vertikal, diberi input '0' jika tidak. Hal tersebut dikarenakan, occam umumnya hanya melakukan perhitungan *penalties* secara horizontal dan vertikal. Line 'Stepsize Cut Count' memuat informasi tentang jumlah maksimal dari proses *fitting smooth* yang dilakukan oleh inversi Occam, karena Occam akan terus melakukan pengulangan hingga *fitting smooth* benar-benar 100% terhadap model dan hal itu membutuhkan waktu yang sangat lama dan terkadang tidak bisa tercapai, input pada line ini akan meng-*cut* proses *fitting smooth* yang dilakukan oleh Occam di tiap iterasi.

Line 'Model Limits' memberi keterangan untuk menampilkan model parameter dalam bentuk resistivitas log10, untuk mengaktifkan line tersebut cukup dengan menghilangkan tanda seru (!) sebelum *option*. Line ' Model Value Steps' memberi pilihan untuk menggunakan diskrit model parameter dalam bentuk pangkat, *option* ini tidak umum digunakan dalam display penampang resistivity sehingga umumnya diberi tanda seru

(!) untuk menonaktifkannya. Untuk *option* yang belum dijelaskan pada subbab startup_flex memiliki kesamaan fungsi yang sama pada *option* di subbab startup_1.0, namun hanya memiliki nama yang berbeda tetapi mudah untuk diidentifikasi.

3.3.2. Data File

File 'Data' merupakan file yang merekam semua parameter yang dibutuhkan oleh proses inversi 2D Occam dalam file berekstensi *edi.

```
FORMAT:            OCCAM2MTDATA_1.0
TITLE:             DATAFINAL14
SITES:             20
MT-181  MT-182  MT-183  MT-184  MT-185
MT-186  MT-187  MT-188  MT-189  MT-190
MT-191  MT-192  MT-193  MT-194  MT-195
MT-196  MT-197  MT-198  MT-199  MT-200
OFFSET (M):
-9500.0 -8500.0 -7500.0 -6500.0 -5500.0
-4500.0 -3500.0 -2500.0 -1500.0 -500.0
500.0   1500.0  2500.0  3500.0  4500.0
5500.0  6500.0  7500.0  8500.0  9500.0
FREQUENCIES:        60
794.328 630.9573  ...    0.001258925      0.001
DATA BLOCKS:        4800
SITE    FREQ    DATA TYPE        DATUM    ERROR
     1     1     1   1.70149     0.01
     1     1     2  45.06573     1.00
     1     1     5   1.70149     0.01
     1     1     6  45.06573     1.00
     1     2     1   1.70141     0.01
     1     2     2  45.05677     1.00
     1     2     5   1.70141     0.01
     1     2     6  45.05676     1.00
     1     3     1   1.70134     0.01
     1     3     2  45.04915     1.00
     1     3     5   1.70134     0.01
     1     3     6  45.04915     1.00
     1     4     1   1.70128     0.01
     1     4     2  45.04268     1.00
     1     4     5   1.70128     0.01
     1     4     6  45.04268     1.00
```

Gambar 3. 5 Format Data File

File 'data' terdiri dari 11 *option* parameter. Untuk line 'FORMAT' secara default diisi dengan nama OCCAM2MTDATA_1.0 yang merupakan nama file 'data' yang diinputkan di program inversi 2D Occam. Line 'TITLE memuat nama dari file data yang akan dikenali oleh file 'startup'. Line 'SITES' memuat keterangan tentang jumlah titik stasiun yang diinput, dan menampilkan nama-nama titik stasiun tersebut

di line berikutnya. Line ‘Offset’ memuat informasi tentang Offset tiap titik stasiun, dan menampilkannnya pada line berikutnya. Line ‘Frekuensi’ memuat list frekuensi titik stasiun pengukuran (Gambar3.6).

```
>FREQ //60
 7.943280e+02  6.309573e+02  5.011873e+02  3.981073e+02  3.162278e+02  2.511887e+02
 1.995262e+02  1.584893e+02  1.258925e+02  1.000000e+02  7.943282e+01  6.309573e+01
 5.011871e+01  3.981071e+01  3.162278e+01  2.511886e+01  1.995262e+01  1.584893e+01
 1.258925e+01  1.000000e+01  7.943283e+00  6.309575e+00  5.011871e+00  3.981072e+00
 3.162278e+00  2.511886e+00  1.995262e+00  1.584893e+00  1.258925e+00  1.000000e+00
 7.943282e-01  6.309573e-01  5.011873e-01  3.981072e-01  3.162278e-01  2.511886e-01
 1.995262e-01  1.584893e-01  1.258925e-01  1.000000e-01  7.943282e-02  6.309573e-02
 5.011871e-02  3.981072e-02  3.162277e-02  2.511886e-02  1.995262e-02  1.584893e-02
 1.258925e-02  1.000000e-02  7.943284e-03  6.309573e-03  5.011873e-03  3.981073e-03
 3.162278e-03  2.511886e-03  1.995262e-03  1.584893e-03  1.258925e-03  1.000000e-03
```

Gambar 3. 6 Nilai frekuensi titik stasiun pengukuran pada file ekstensi *edi

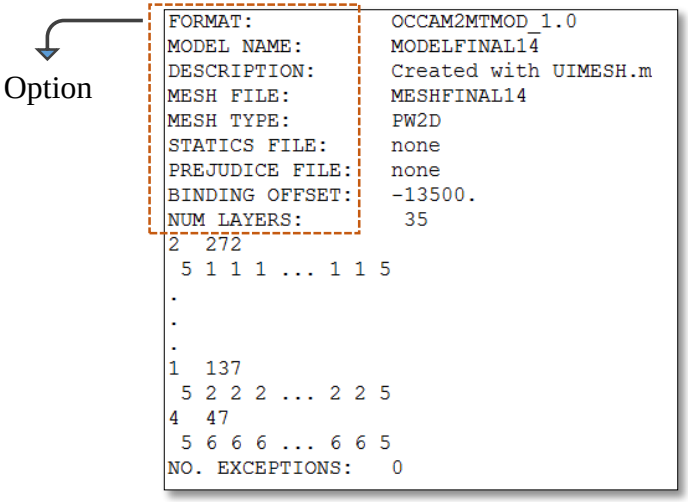
Selanjutnya ialah line ‘data blocks’, line ini memuat keterangan jumlah dari block data untuk semua input titik stasiun. Pada line vertikal ‘data type’ memuat keterangan sebagai berikut:

1 = TE apparent resistivity (log 10)
2 = TE phase
3 = real Hz/Hy
4 = real Hz/Hy
5 = TM apparent resistivity (log 10)
6 = TM phase
7 = TM Ez/Ey tipper
8 = TM Ez/Ey tipper
9 = TE apparent resistivity (linear)
10 = TM apparent resistivity

Data type tersebut dapat dipilih sesuai target inversi 2D yang diinginkan. Line vertikal ‘datum’ memuat nilai dari data type yang diinput. Dan line vertikal ‘error’ memuat nilai error untuk tiap datum.

3.3.3. Model File

File ‘Model’ merupakan file yang berfungsi untuk mendefinisikan model awal yang akan digunakan sebagai input program inversi 2D Occam. Model awal dapat dibentuk berdasarkan pengetahuan mengenai kondisi struktur Geologi bawah permukaan daerah pengukuran metode Magnetotellurik, namun bisa juga membuat model secara default dengan memasukkan parameter-parameter model block.

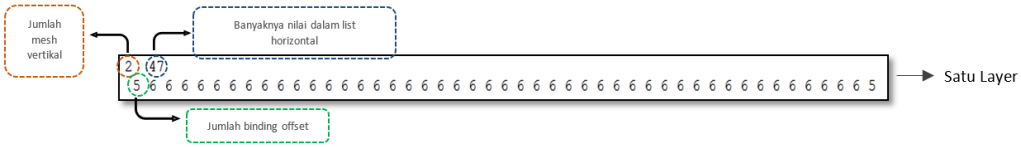


Gambar 3. 7 Header Model File

File 'Model' sangat sensitif terhadap karakter spasi dan penggunaan huruf kapital, sehingga harus berhati-hati dalam input nilai dari *option*. Line 'FORMAT' secara default diisi dengan OCCAM2MTMOD_1.0, sehingga dapat dikenali oleh program inversi 2D Occam.

Line 'MODEL NAME' memuat informasi mengenai nama model file ini. Line 'DESCRIPTION' memuat keterangan tambahan oleh pengguna. Line 'MESH FILE' memuat nama dari file 'mesh' yang dapat diintegrasikan dengan file 'model' tersebut. Line 'MESH TYPE' secara default diisi PW2D.

Line 'Statics File' memuat informasi ada atau tidaknya input file static, file static ini merupakan file *optional* yang berfungsi sebagai data tambahan untuk statik shift. Line 'Prejudice File' memuat informasi ada atau tidaknya file *prejudice*, file ini berfungsi juga sebagai keterangan tambahan untuk *static shift parameter*. Line 'Binding Offset' merupakan input nilai dari penjumlahan offset terjauh dan margin offset. Line 'Num Layers' memuat jumlah layer pada model awal yang dibuat. List nilai dibawah line 'Num layer' memuat informasi mengenai hubungan antara mesh block dan model block.



Gambar 3. 8 Definisi nilai dalam satu layer model

Gambar 3.8 menunjukkan bahwa baris pertama untuk satu layer mendefinisikan jumlah mesh yang dipakai oleh model block dan banyaknya jumlah model block secara horizontal yang mengarah kepada baris kedua. Baris kedua terdiri dari 2 bagian, yaitu model block untuk binding offset dan model block untuk maksimum offset dari semua stasiun.

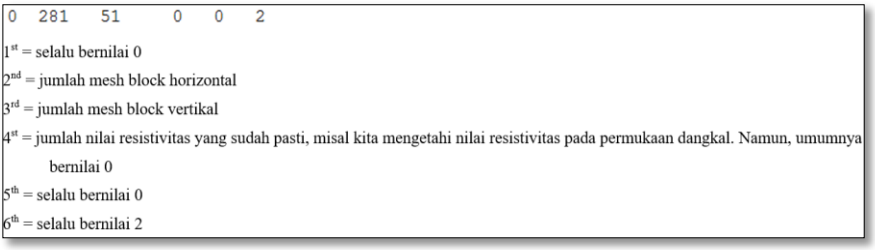
3.3.4. Mesh File

File ‘Mesh’ berfungsi sebagai layout/template dari file ‘model’. File ‘Mesh’ juga sangat sensitif terhadap karakter spasi dan huruf kapital.

```
TEST 14
     0   281    51     0    0    2
248      207      172      144      120      100      100 ....          100      100      120      144      172      207      248
10       11.0     12.100000000000001       13.310000000000002       14.641000000000004       16.105100000000004       17.715610000000005
     0
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
```

Gambar 3. 9 Contoh tampilan Mesh File

File ‘mesh’ tidak memiliki header seperti file yang lainnya, baris pertama berisi deskripsi bebas. Baris kedua memiliki ketentuan sebagai berikut



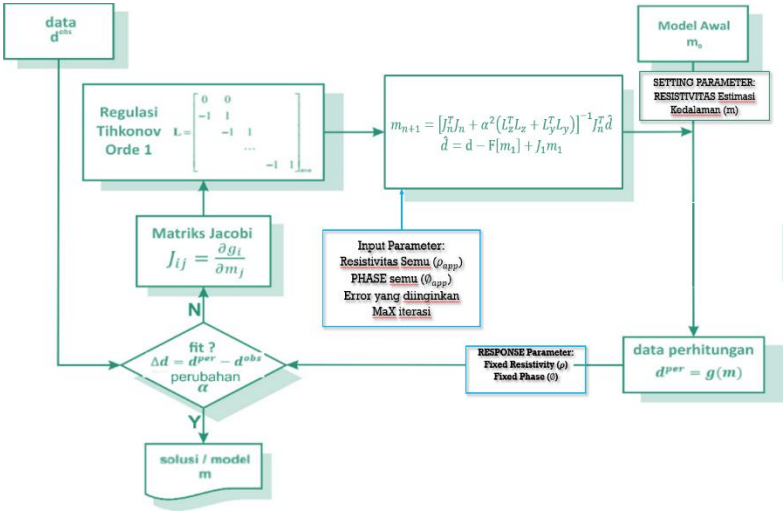
Gambar 3. 10 ketentuan pengisian baris kedua dalam mesh file

Enam angka diatas merupakan karakter spesifik untuk membuat mesh file. Baris ketiga terdiri dari 2 bagian inti, yaitu mesh block untuk binding offset dan mesh block untuk titik stasiun. Mesh block untuk binding offset memiliki deret sesuai faktor skala yang ditentukan, sehingga mesh block yang makin jauh dari titik terluar titik stasiun maka memiliki mesh blok horizontal yang semakin besar.

Mesh block untuk titik stasiun merupakan mesh block yang dibentuk dari pembagian offset tiap stasiun, mesh block untuk titik stasiun sebaiknya memiliki bentuk yang sama untuk semua titik stasiun. Baris keempat berisi list mesh block vertikal. Dan baris kelima '?' menunjukkan mesh block secara horizontal dan vertikal.

3.4. Modelling Inversi 2D Occam

Inversi occam termasuk kedalam penyelesaian inversi non-liner dengan menggunakan pendekatan linier.



Gambar 3. 11 Algoritma penyelesaian inversi non-linier dengan pendekatan linier (Grandis, 2009)

Inversi Occam diawali dengan membangun sebuah model awal (m_0), model awal yang dibuat sebaiknya membentuk setengah bola. Sebelum masuk ke forward modelling maka terlebih dahulu dibuat Matriks Roughness. Kemudian membuat pemodelan kedepan, dari model yang ada tersebut dikalikan dengan matriks Jacobi, dan muncullah data hasil perhitungan pemodelan kedepan. Di tahap ini, dicari margin antara nilai dari parameter data perhitungan dan data pengukuran. Apabila misfit sesuai dengan yang diinginkan maka iterasi akan berhenti. Namun, jika belum maka akan dilakukan pengulangan kembali dengan menggunakan parameter hasil margin tersebut, α. Kemudian dibuat kembali matriks Jacobi sesuai persamaan 2.24, kemudian membuat matriks regulasi roughness hingga masuk ke modifikasi parameter.

BAB IV
ANALISA DAN PEMBAHASAN

4.1. Pembuatan GUI Program

GUI Program inversi 2D metode Occam ini, selanjutnya diberi nama TGMT2D, disusun dengan menggunakan bahasa pemrograman Python dan *QTDesigner* untuk *Guide User Interface*-nya, untuk plotting data hasil inversi ditulis oleh Kerry Key, dan bahasa pemrograman Fortran90 untuk inversi occam 2D oleh David Myer. Penelitian tugas akhir ini, yaitu inversi 2D yang menggunakan algoritma Occam merupakan salah satu dari modul blueprint yang disiapkan untuk pengembangan lebih lanjut. Berikut adalah modul blueprint software yang siap untuk dikembangkan.

Gambar 4. 12 Mainwindow dari blueprint software TGMT

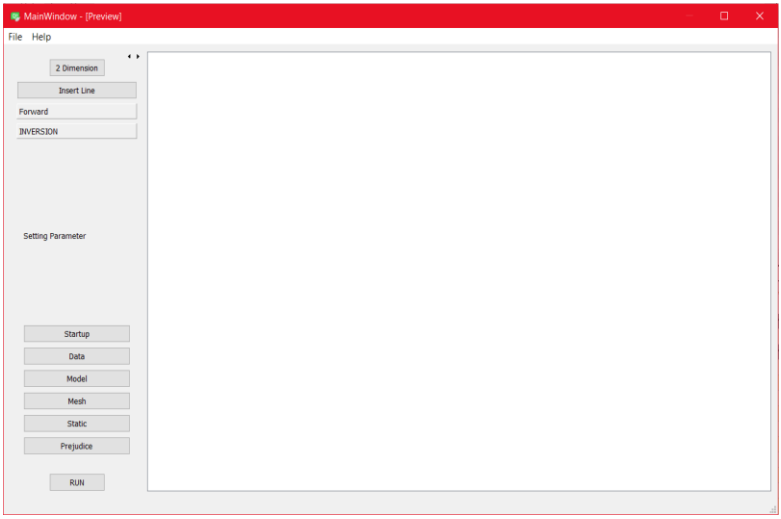
Pada penelitian sebelumnya telah dibuat inversi 1D dengan menggunakan algoritma Occam dan Simulated Annealing, sehingga kedepan dapat di*input* kedalam modul 1DIMENSION pada gambar 3.1 dengan menyesuaikan bahasa pemrograman yang telah dilakukan sebelumnya. Modul MAPS akan menampilkan input titik pengukuran yang telah dilakukan, modul ini umumnya berisi informasi titik stasiun pengukuran Metode Magnetotellurik dan kedepannya memungkinkan

pengguna melihat kondisi terkini dari titik stasiun secara *realtime* yang bisa ditampilkan dalam Google Earth, yang tentunya dapat membantu pemahaman regional sekitar titik stasiun pengukuran.

Pada Modul PSEUDO SECTION akan menampilkan penampang pseudo dari data XY, XX, YY, dan YX pada data hasil pengukuran dengan file extension .edi. Kemudian Modul 2 DIMENSION selanjutnya akan dibahas lebih rinci pada bab ini. Modul 3 DIMENSION memungkinkan pengguna untuk melakukan interpretasi dengan dimensi ruang, hal tersebut tentunya memerlukan algoritma yang lebih kompeks daripada penampang 2D namun sangat membantu interpretasi resisitivitas kondisi bawah permukaan daerah target. Pada Modul Interpretasi dapat memperlihatkan Kolaborasi data, baik itu penampang 1D, 2D, dan 3D hasil inversi maupun jika terdapat informasi data sumur, data geologi, dan data geokimia setempat.

4.1.1. Input Data

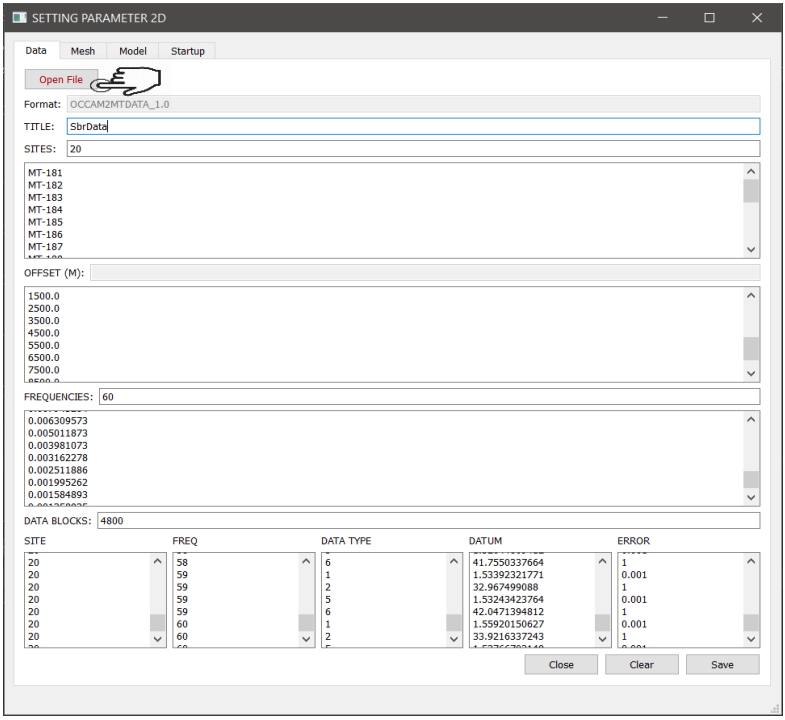
Pada modul 2 Dimension Inversi dibuat model input data yang sesuai dan mudah untuk digunakan. Berikut tampilan subwindow TGMT2D.



Gambar 4. 13 subwindow modul 2D Inversion TGMT2D

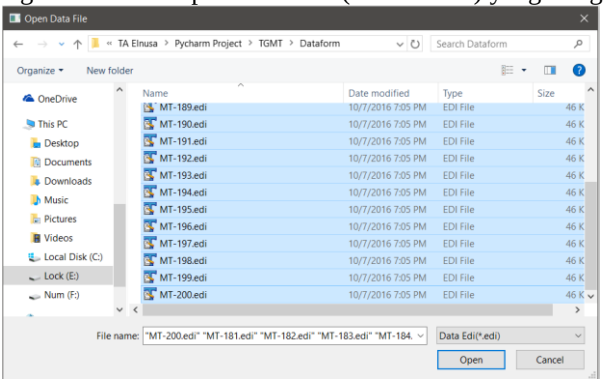
Pada tombol Insert Line, berfungsi untuk membuka data file ekstensi *edi dan memplotnya pada grid penampang X dan Y berdasarkan parameter offset pada tiap stasiun pengukuran. Tombol Forward selanjutnya digunakan untuk menghasilkan data dari model yang telah dibuat, output dari forward modelling ini berupa file ekstensi *edi. Pada tombol Inversi, disediakan tombol Setting Parameter, tombol tersebut akan memunculkan jendela baru berupa 4 (empat) tab yang berisi input paramter Data, Model, Mesh, dan Startup. Tab tersebut masing-masing berfungsi untuk membuat dan menyimpan file ekstensi *txt yang diperlukan untuk menjalankan program inversi 2D Occam.

Input file pertama adalah input Data. Tab Input Data berfungsi untuk mengekstrak variabel Jumlah Titik Stasiun, Offset Titik Stasiun, Frekuensi, Rho XY, Phase XY, Rho YX, Phase YX dan Error.



Gambar 4. 14 Input Data Edi

Untuk membuka file ekstensi Edi, klik tombol ‘Open File’ yang ditandai digambar 3.3 dan pilih file *edi (titik stasiun) yang diinginkan.



Gambar 4. 15 Window pop-up untuk mencari file ekstensi *edi

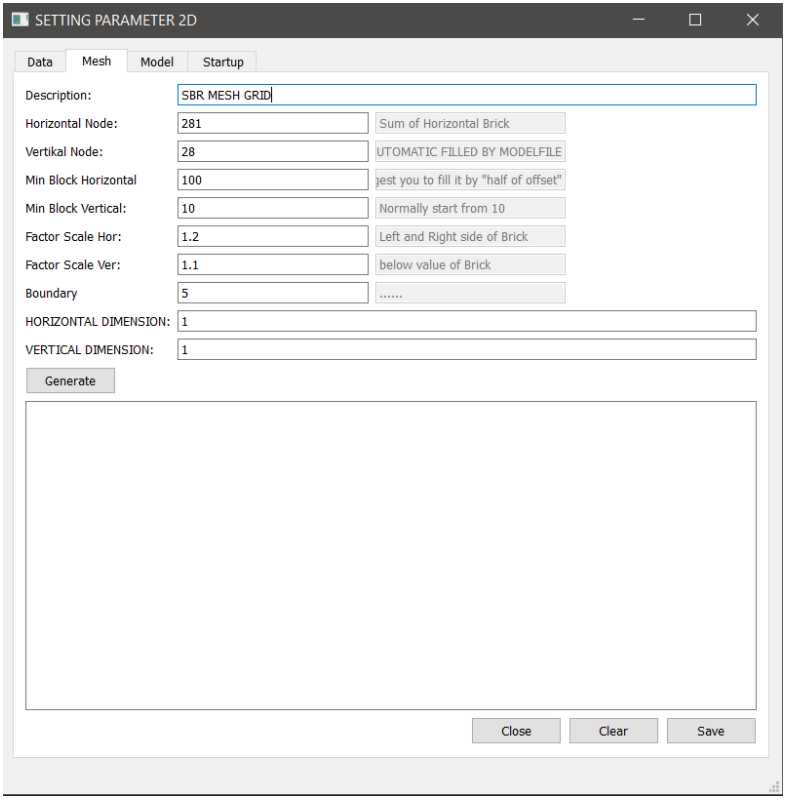
Untuk melakukan list variabel ke form input Data digunakan bahasa pemrograman python (lampiran 2). Pada kolom DATUM, dimana memuat variabel TE apparent resistivity, TM apparent resistivity, TE Phase, dan TM Phase menggunakan persamaan 2.12 dan 2.13 (lampiran 1).

Untuk nilai error pada Kolom ERROR umumnya dapat ditentukan melalui 2(dua) pendekatan, yaitu persentase dari parameter dan list error dari data ekstensi *edi. Pada uji data sintetik di penelitian tugas akhir ini penulis menggunakan pendekatan pertama (menentukan persentase). Dengan nilai 0.001 untuk *apparent resistivity* dan 1 untuk *phase.*

Setelah selesai membuka data file *edi, selanjutnya klik Save untuk menyimpan list variabel kedalam file *txt.

4.1.2. Mesh Grid

Titik stasiun pengukuran Magnetotellurik yang sudah diload pada input Data kemudian dibuat grid sebagai layout parameter resistivitas bawah permukaan untuk keseluruhan offset titik stasiun pengukuran.



Gambar 4. 16 Window pop up mesh grid

Terdapat ketentuan yang harus dipahami untuk melakukan input variabel yang diperlihatkan oleh kolom ditiap pengisian untuk singkatnya dan menekan tombol HELP di bagian luar window Setting Parameter 2D untuk lebih jelasnya.

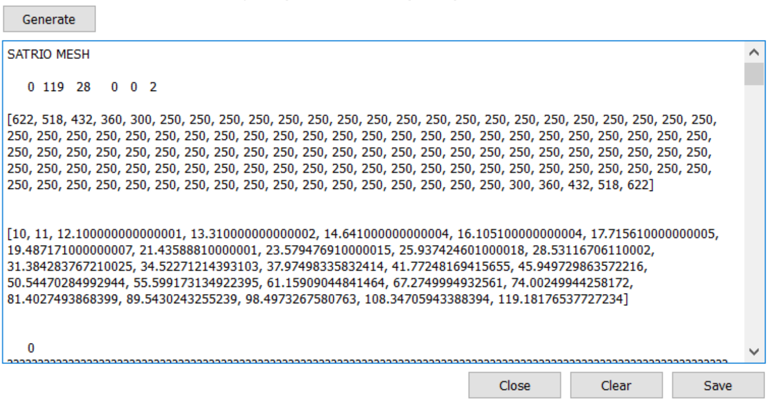
Mesh grid terbagi menjadi 2 elemen penting, yaitu horizontal dan vertikal. Untuk tiap elemen tersebut terdapat variabel jumlah node/grid, lebar/panjang grid, *boundary* dan faktor pengali.

Yang harus diketahui sebelum menginput nilai-nilai tersebut adalah Total Offset (titik stasiun paling kiri hingga titik stasiun paling kanan dalam ordinat X). Pertama, tentukan panjang grid/nodes mesh (untuk horizontal) dengan syarat nilai x(input) lebih kecil dari jarak offset

antar titik (minimal), semakin kecil panjang grid maka semakin banyak mesh yang menyusun grid secara horizontal. Kedua, tentukan lebar grid/nodes (untuk vertikal), tidak ada syarat tertentu dalam menginput nilai lebar grid/nodes ini. Ketiga adalah menentukan faktor skala untuk jumlah boundary, dengan ketentuan bahwa boundary horizontal dimulai dari titik terluar stasiun pengukuran dan untuk boundary vertikal dimulai dengan, a+1 (a = lebar grid/nodes awal). Keempat, tentukan banyaknya grid boundary untuk horizontal.

Dengan memahami dan mengikuti ketentuan diatas maka secara automatis nilai horizontal grid/nodes akan diketahui. Horizontal nodes merupakan jumlah semua grid/nodes dalam offset dan boundary ditambah 1(satu). Selanjutnya ialah memasukkan grid vertikal sesuai keinginan pengguna (tidak lebih dari skin depth frekuensi terendah). Vertikal grid/nodes merupakan jumlah semua grid vertikal ditambah 1(satu).

Horizontal dimension dan Vertikal dimension merupakan nilai awal dari model block yang akan mengisi grid mesh.

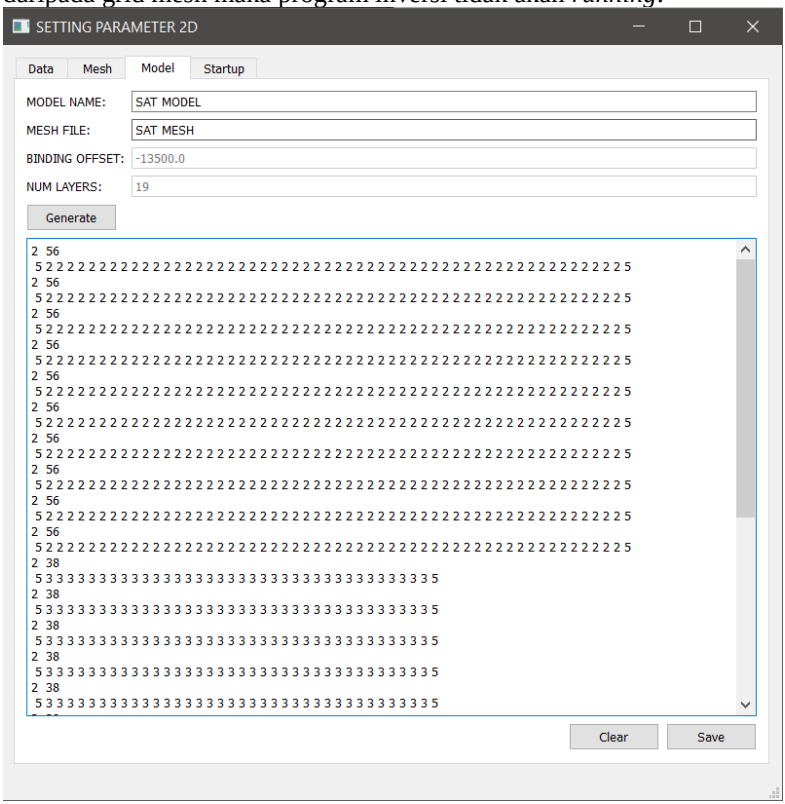


Gambar 4. 17 Tampilan tombol generate untuk mengecek kebenaran input mesh grid

Tombol Generate pada window Mesh akan menampilkan hasil sementara sebelum data mesh disimpan. Penambahan 1(satu) nilai untuk tiap grid horizontal dan vertikal mengikuti format program Inversi Occam 2D.

4.1.3. Model Block

Model block merupakan nilai parameter resistivity yang mengisi layout grid mesh, sehingga apabila model block lebih banyak (jumlah) daripada grid mesh maka program inversi tidak akan *running*.



Gambar 4. 18 Window model block

Input pada window model block hanya memasukkan variabel Model Name dan Mesh File sebelumnya. Nilai Binding Offset akan secara otomatis terisi dari Input Data, yaitu dari banyaknya stasiun yang dipilih, secara default binding offset bernilai 4000m dari offset terluar sumbu X negatif.

Sebelum membahas numlayer, maka kita kembali dahulu memahami horizontal dimension dan vertikal dimension pada input

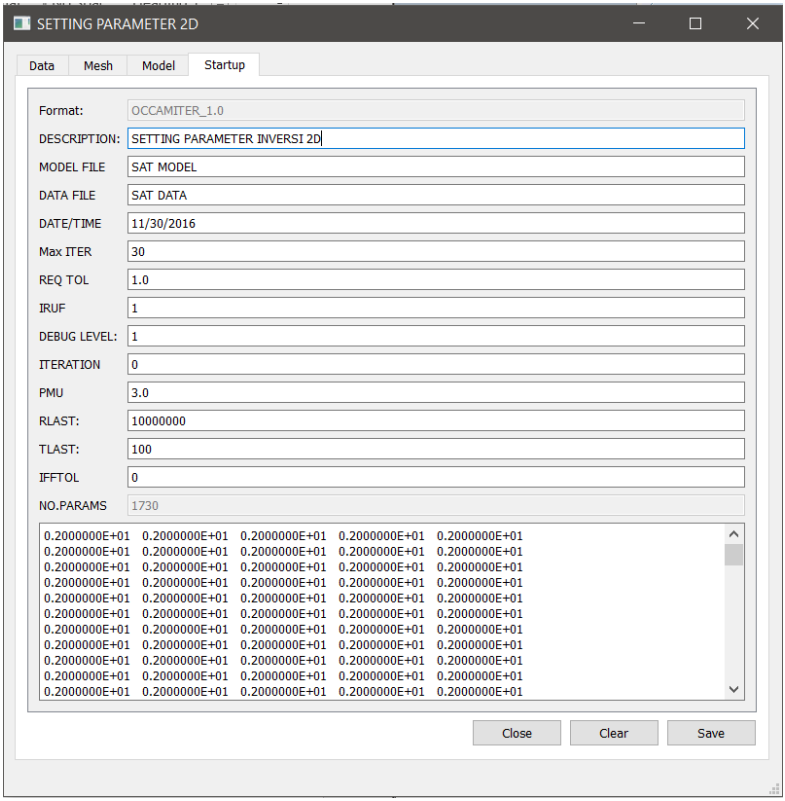
Mesh. Nilai awal yang dimaksud pada pembahasan sebelumnya adalah jumlah model block yang mengisi grid mesh, ketentuannya adalah secara horizontal dan vertikal akan membagi lapisan menjadi 3 layer utama (memiliki nilai horizontal dimension dan vertikal dimension yang sama). Umumnya untuk memudahkan pembagian jumlah grid/nodes minus 1 maka dimulai dengan nilai 1 (1 model block mengisi 1 grid mesh). Kemudian selanjutnya secara otomatis akan memilih nilai yang pembagiannya habis dibagi '0', sehingga menjadi bilangan bulat.

Dari input nilai horizontal dan vertikal dimension maka program akan secara otomatis menginput banyaknya lapisan model awal bawah permukaan, yang erat hubungannya dengan vertikal grid/nodes.

Sama seperti mesh grid, sebelum menyimpan file model sebaiknya koreksi terlebih dahulu dengan menekan tombol generate sehingga akan menampilkan block model untuk tiap layer bawah permukaan.

4.1.4. Setting Inversi di Startup

Window Startup merupakan file yang mengatur parameter inversi Occam 2D. Di window startup ini telah diisi nilai untuk tiap variabel secara default. Namun hal itu tidak memberi batasan kepada pengguna untuk memasukan input nilai sesuai kebutuhannya.



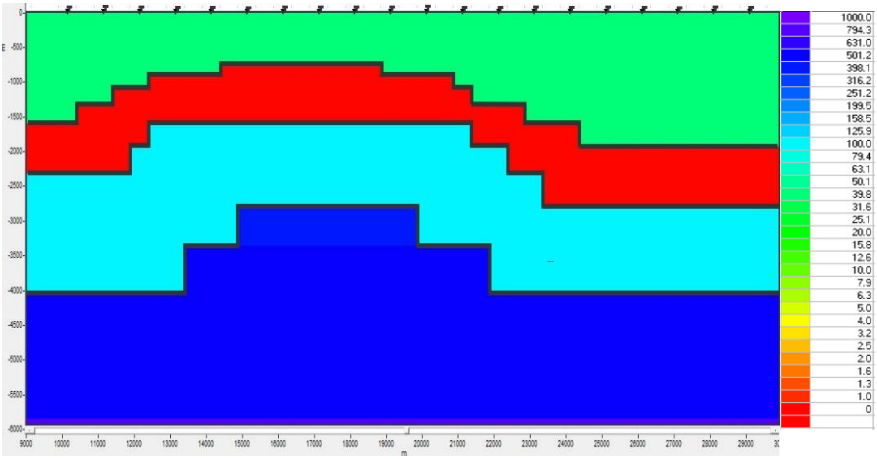
Gambar 4. 19 Window Startup untuk mengatur parameter inversi

Penjelasan mengenai parameter inversi pada gambar 3.8 telah dibahas pada bab sebelumnya, namun beberapa parameter inversi yang mempengaruhi jumlah dan waktu iterasi adalah Maksimum Iterasi (Max Iter), Roughness (IRUF), Debug, Langrange Parameter (PMU), target missfit(Req Tol), dan resistivity model awal (No.Params).

Parameter-parameter yang disebutkan diatas selain resistivity model awal dapat direkayasa sesuai kebutuhan, naum untuk resistivity model awal atau 'No Param' secara otomatis akan terisi saat pengguna menyimpan file model block. No. Params ini adalah banyaknya block model yang mengisi grid mesh, untuk perhitungan awal (iterasi ke-0), maka dimasukkan nilai resistivity ialah Log 10^2.

4.2. Validasi Data Sintetik

Validasi data sintetik merupakan pencocokan data hasil inversi terhadap model sintetik yang dibuat. Validasi ini bertujuan untuk melihat kemiripan lapisan bawah permukaan data hasil inversi 2D metode Occam, dalam parameter resistivitas, terhadap resistivitas model sintetik. Untuk menguji program TGMT2D digunakan dua model sintetik.



Gambar 4. 20 Model 1 yang digunakan untuk mengenerate data

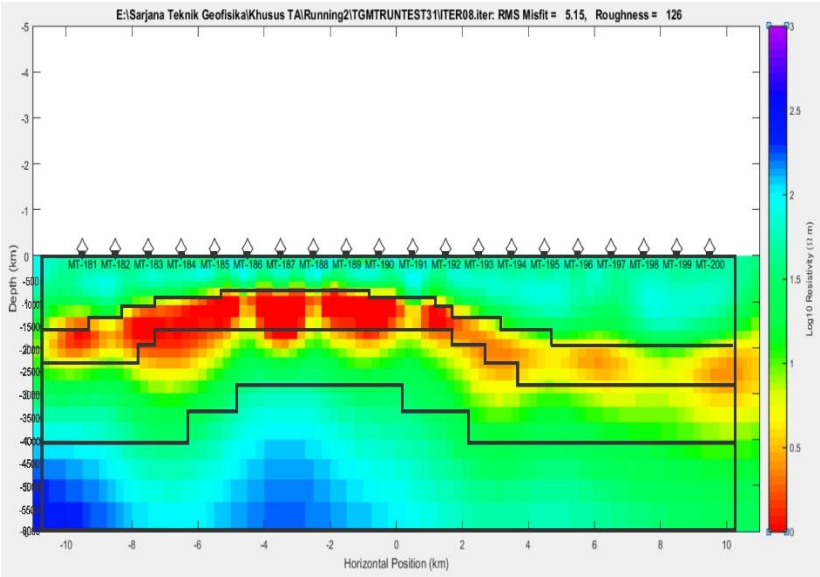
Model sintetik pertama terdiri dari 4 lapisan dengan nilai resisitivitas yang memiliki kontras resistivitas yang besar, lapisan tersebut dibuat merepresentasikan struktur *petroleum system* pada umumnya. Nilai parameter tiap lapisan tersebut ditampilkan pada tabel berikut,

Tabel 4. 1 Parameter Resistivitas Model Sintetik

Lapisan	Kedalaman (m)	Ketebalan (m)	Resistivitas (Ωm)
Hijau	0 – 1500	800 - 1500	50
Merah	1600 – 2400	800	1
Biru Muda	1700 – 4000	1600	100
Biru Tua	3000 – 6000	2000 - 3000	550

Kemudian dilakukan proses inversi 2D menggunakan program TGMT2D dari data sintetik hasil forward modelling pada gambar 4.9.

Validasi hasil inversi 2D metoda Occam terhadap model yang dibuat dilakukan dengan cara melihat kemiripan lapisan (nilai resistivitas) per kedalaman yang ditandai dengan *boldline* pada penampang hasil inversi 2D Occam dan nilai RMS missfit hasil inversinya.



Gambar 4. 21 Penampang 2D hasil Inversi Occam

Pada Gambar 4.10 menunjukan penampang hasil inversi Occam 2D yang di*overlay* dengan batas perlapisan (*boldline*) dari model yang dibuat. Teknik *Overlay* yang dilakukan ialah dengan menyamakan titik pengukuran dan nilai kedalaman tiap titik, sehingga pada gambar 4.10 terlihat bahwa *boldline* model tidak mengisi penampang 2D resistivitas secara sempurna. Hal itu hanya dikarenakan oleh offset yang dimasukan di frame inversi program TGMT2D adalah -11km hingga 11km, sedangkan untuk model yang dibuat frame yang digunakan adalah -10.5km hingga 10.5km.

Colorbar yang ditunjukkan pada gambar 4.9 dan gambar 4.10 memiliki perbedaan, untuk model sintetik (gambar 4.9) menggunakan *range* nilai resistivitas pada umumnya (x Ohm.m), sedangkan untuk penampang resistivitas TGMT2D menggunakan nilai Log10(x.Ohm.m).

Sebelum membahas kevalid-an penampang resistivitas hasil inversi TGMT2D, maka parameter inversi yang digunakan ditampilkan secara rinci pada Tabel 4.2.

Tabel 4. 2 Parameter penampang inversi TGMT2D pada gambar 3.10

Parameter	Nilai	Keterangan
Mesh Grid Horizontal	250m	Mesh grid horizontal yang digunakan ¼ dari jarak tiap stasiun, jarak yang dianggap ideal oleh penulis untuk menghindari roughness yang besar dan nilai resistivitas yang relevan.
Mesh Grid Vertikal	10m	Mesh grid vertikal dimulai dengan ketebalan 10m dan bertambah sebanyak 94 grid secara vertikal dengan faktor pengali 1.1 sehingga kedalaman mesh maksimal adalah sebesar 777 km
Jumlah Layer	63	Untuk merelevankan nilai mesh grid vertikal tersebut maka dibuat 63 layer
Iterasi	8	Iterasi berhenti di 8 dari 20(maksimal iterasi) dikarenakan nilai missfit sudah tidak bisa lebih kecil lagi.
RMS. Missfit	5.15%	Nilai RMS missfit diiterasi ke-8
Parameter Lagrange	3.83	Nilai konstanta lagrange yang digunakan pada iterasi ke-8
Roughness	126.4579	Tingkat kekasaran penampang hasil inversi pada iterasi terakhir

Pada lapisan pertama, yaitu lapisan permukaan yang memiliki nilai resistivitas sebesar 50 Ω.m dan kedalaman 800m di stasiun MT-187 hingga kedalaman 1500 di titik stasiun MT-197, untuk hasil inversi TGMT2D menampilkan batas lapisan yang mirip (kedalaman) dengan hasil overlay di lapisan permukaan ini dengan nilai resistivitas 32 Ω.m - 79 Ω.m ($10^{1.5}$ Ω.m – $10^{1.9}$ Ω.m). Untuk perbedaaan warna yang terjadi dikarenakan perbedaan colorbar untuk tiap nilai resisitivitas di pemodelan

hasil inversi TGMT2D, karena itu pada penelitian tugas akhir ini lebih menekankan pada *range* nilai resistivitasnya.

Pada lapisan kedua, yaitu lapisan berwarna merah dengan nilai resistivitas sebesar 1 Ω.m dan kedalaman 800m – 1600m pada titik stasiun MT-187, untuk hasil inversi TGMT2D menampilkan batas lapisan yang mirip dengan hasil overlay model sintetik, yaitu dengan *range* nilai resistivitas 1.2 Ω.m – 3.9 Ω.m ($10^{0.1}$ Ω.m – $10^{0.6}$ Ω.m). Namun dititik MT-184 dan MT-185 dikedalaman 1600m dan dititik MT-193 dan MT-194 dikedalaman 2000m mengalami *overvalue*, hal tersebut berhubungan dengan kondisi kedua struktur dititik tersebut yang tidak lateral sehingga mengakibatkan perhitungan nilai resistivitas dengan prinsip elemen batas pada metode Occam menjadi melebihi nilai resistivitas sebenarnya dikedalaman tersebut.

Pada lapisan ketiga, yaitu lapisan berwarna biru muda dengan nilai resistivitas sebesar 100 Ω.m dan kedalaman 1700m hingga 2800m di titik stasiun MT-187, menampilkan penampang resistivitas 2D hasil inversi TGMT2D dengan nilai resistivitas sebesar 2.5 Ω.m - 79 Ω.m ($10^{0.4}$ Ω.m – $10^{1.9}$ Ω.m) di *range* kedalaman yang sama. Pada lapisan ini, nilai resistivitas yang dihasilkan jauh lebih rendah resistivitas sebenarnya, hal tersebut dapat dikarenakan kontras resistivitas yang besar dengan lapisan diatasnya, sehingga membuat pembuat resisitivitas dilapisan ini menjadi samar, hal yang dapat menjadi penyebab lainnya adalah besar dari grid mesh yang digunakan, karena grid mesh vertikal yang berbanding lurus terhadap kedalaman maka nilai untuk tiap mesh secara vertikal akan dipengaruhi oleh frekuensi minimum grid dan maksimum grid yang dihitung menggunakan metode elemen batas. Hal tersebut berdampak kepada jumlah data resistivitas yang mengisi lapisan ini tidak sebanyak yang sebenarnya melainkan hasil perhitungan di skin depth minimum grid dan maksimum grid.

Pada lapisan keempat, yaitu lapisan berwarna biru tua dengan nilai resistivitas sebesar 500 Ω.m dan kedalaman 3000m hingga 6000m pada titik MT-18s7, menampilkan penampang hasil inversi TGMT2D dengan nilai sebesar 32 Ω.m -200 Ω.m ($10^{1.5}$ Ω.m – $10^{2.3}$ Ω.m). Lapisan yang cukup kelihatan(memiliki resistivitas >100 Ω.m) berada di kedalaman yang paling dekat dengan permukaan (Titik MT-187), sedangkan untuk titik MT-193 hingga MT-200 memiliki resistivitas yang berbeda cukup jauh dari resistivitas model sintetik. Hal yang perlu diperhatikan adalah skin depth maksimal dari pemodelan ini mencapai kedalaman ratusan kilometer, namun untuk memfokus terhadap data yang

masih relevan di batasi hingga kedalaman 6km. Hal tersebut memungkinkan adanya pergeseran nilai resistivitas akibat perhitungan matematis yang dipengaruhi oleh skin depth di titik minimum mesh grid dan maksimum grid. Untuk meminimumkan pergeseran tersebut, seharusnya dibuat mesh grid yang lateral terhadap kedalaman, namun tentunya akan membutuhkan waktu yang llabih lama dalam proses iterasi. Nilai resistivitas yang cukup relevan pada lapisan ini, yaitu terdapat ditengah dari line offset, hal itu sesuai dengan model awal yang dibuat sehingga sesuai dengan bentuk *half-space*.

Hasil inversi data sintetik menggunakan metode Occam 2D menghasilkan RMS missfit sebesar 5.15%, nilai tersebut dinilai cukup baik untuk mempercayai penampang resistivitas hasil inversi dengan toleransi nilai RMS missfit dibawah 10%. Roughness yang dihasilkan pada iterasi terakhir hasil inversi Occam bernilai 126,457. Tingkat kekasaran tersebut tidak menjadi tolak ukur primer dalam mengukur kevalid-an hasil inversi, namun nilai tersebut lebih menitikberatkan kepada kontras resistivitas yang menyusun tiap model block. Untuk mendapatkan nilai roughness dibawah 10, maka perlu dibuat mesh grid yang seminimal mungkin yang tentunya akan berefek kepada waktu iterasi saat proses inversi.

4.3. Komparasi Software Konvensional

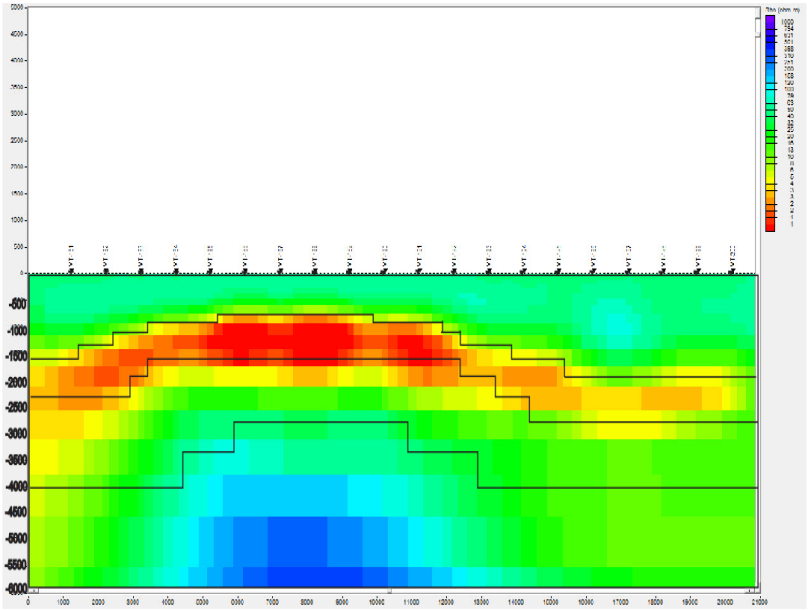
Komparasi dengan software konvensional merupakan perbandingan penampang resistivitas hasil inversi data 2D yang dihasilkan oleh software WinGlink terhadap penampang resistivitas hasil inversi TGMT2D. Penampang resistivitas inversi 2D yang dilakukan pada software WinGlink dilakukan di Workstation PT. Elnusa Tbk. Untuk software WinGlink, metode inversi yang digunakan dalam pengolahan data 2D Magnetotellurik adalah metode *Nonlinier Conjugate Gradient* (NLCG), dan untuk program TGMT2D, metode inversi yang digunakan dalam pengolahan data 2D magnetotellurik adalah metode Occam.

Perbandingan yang dilakukan menggunakan parameter yang dianggap menghasilkan penampang hasil inversi paling baik untuk software WinGlink dan TGMT2D sehingga dapat dianalisa perbedaan model penampang resistivitas 2D yang dihasilkan. Parameter input inversi untuk software WinGlink dan program TGMT2D ditampilkan pada tabel 3.3.

Tabel 4. 3 Input Seting Parameter inversi untuk TGMT2D dan WinGlink

Parameter	TGMT2D	WinGlink	Keterangan
Max Iterasi	30	30	Iterasi akan berhenti di nilai ini jika missfit belum tercapai atau nilai missfit semakin besar
Min. Grid Mesh Horizontal	250	500	*Range* spasi tiap block model
Jumlah Lapisan	63	x	Lapisan model awal yang digunakan

Dari input parameter yang dilakukan pada tabel 4.3 untuk program TGMT2D dan WinGlink menghasilkan penampang resistivitas 2D yang ditampilkan pada gambar 4.10 dan 4.11. Maksimal iterasi yang diatur pada kedua program senilai 30, yang umumnya merupakan nilai yang cukup efektif (bahkan lebih) untuk menghasilkan RMS missfit yang kecil. Minimal grid mesh dengan offset 1000m pada program WinGlink senilai 500m dan 250m untuk TGMT2D, nilai ini berpengaruh terhadap kekasaran (roughness) dari penampang resisitivitas 2D hasil inversi.



Gambar 4. 22 Penampang 2D resistivitas hasil inversi software WinGlink

Penampang 2D resistivitas hasil inversi WinGlink menggunakan data yang sama dengan hasil inversi TGMT2D pada gambar 4.10. Pada lapisan pertama, nilai resistivitas yang dihasilkan dari inversi software WinGlink berada pada *range* 25 Ω.m - 79 Ω.m ($10^{1.39}$ Ω.m-$10^{1.9}$).

Pada lapisan Kedua, dengan nilai resistivitas sangat rendah, *overlay* yang dilakukan terhadap hasil inversi 2D sangat cocok untuk top lapisannya, namun untuk *bottom layer*, nilai resistivitas rendah ini masuk kedalam layer ketiga, terutama di struktur lateral yaitu di titik stasiun MT-187, sedangkan untuk struktur yang cenderung diagonal (MT-183, MT-184, MT-192, MT-193, dan MT-94) software WinGlink cukup mampu membatasi kontras resistivitas yang cukup tinggi antara lapisan. *Range* nilai resistivitas pada lapisan ini adalah 1 Ω.m-5 Ω.m (10^{0} Ω.m-$10^{0.7}$ Ω.m).

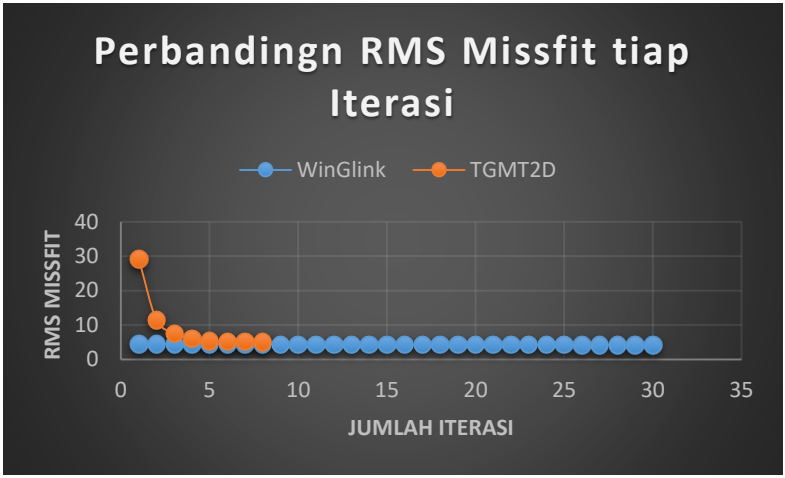
Pada lapisan ketiga, penampang 2D resistivitas hasil inversi software WinGlink bernilai 4 Ω.m-79 Ω.m ($10^{0.6}$ Ω.m-$10^{1.9}$ Ω.m). *Range* nilai di lapisan ini, yang seharusnya 100 Ω.m sulit dimodelkan oleh inversi NLCG maupun Occam, sesuai dengan penjelasan sebelumnya pada penampang inversi TGMT2D.

Pada lapisan keempat, dengan nilai resistivitas 500 Ω.m, hasil inversi yang dihasilkan oleh software WinGlink berada pada *range* nilai 25 Ω.m -251 Ω.m ($10^{1.39}$ Ω.m – $10^{2.4}$). Kenampakan nilai resistivitas di titik stasiun MT-194 hingga MT-200 hampir sama dengan penampang TGMT2D, namun terlihat block penyusun di penampang inversi WinGlink lebih besar.

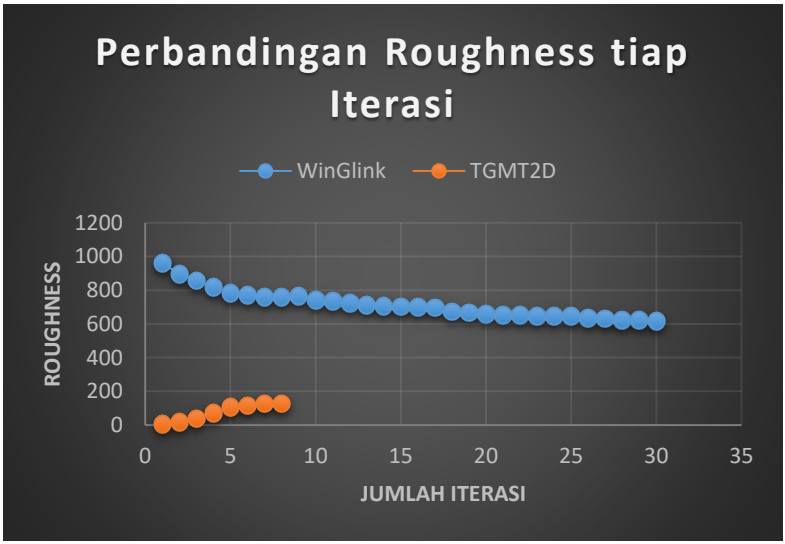
Tabel 4. 4 Range nilai tahanan jenis hasil inversi TGMT2D dan WinGlink pada tiap lapisan

Layer n	**Range Resistivitas (Ohm.m)**	
	TGMT2D	WinGlink
1 (50 Ohm.m)	32-79	25-79
2 (1 Ohm.m)	1.2-3.9	1-5
3 (100 Ohm.m)	2.5-79	4-79
4 (500 Ohm.m)	32-200	25-251

Sebelumnya tabel 4.3 memberi informasi input parameter inversi untuk program TGMT2D dan WinGlink. Dari setting paramter yang diinput maka tiap metode melakukan iterasi berdasarkan algortima NLCG dan OCCAM. Dari masing-masing inversi yang dilakukan didapat nilai RMS error tiap iterasi yang ditampilkan pada 4.12. Untuk Program TGMT2D inversi berhenti di iterasi ke-8, hal tersebut karena model dianggap sudah konvergen sehingga Missfit sudah mencapai perhitungan maksimum (semakin kecil). Untuk program WinGlink, inversi berhenti di iterasi ke-30, hal itu menunjukkan bahwa inversi berhenti karena setting maksimum iterasi yang diberikan, sehingga memungkinkan inversi untuk terus melakukan iterasi namun dengan nilai missfit yang konstan dapat diasumsikan bahwa iterasi ke-30 sudah dapat dipercaya.



Gambar 4.12 Perbandingan missfit program WinGlink dan TGMT2D



Gambar 4.13 Perbandingan roughness TGMT2D terhadap WinGlink

Pada gambar 4.12, menunjukkan nilai missfit maksimum untuk WinGlink adalah 4.2732% dan TGMT2D senilai 5.1093%. Dari masing-

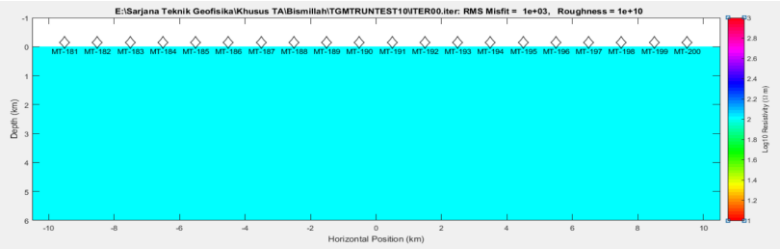
masing nilai tersebut, disepakati telah mencapai batas toleransi, yaitu <10%. Sehingga kedua penampang resistivitas 2D dapat dijadikan acuan awal untuk menentukan struktur bawah permukaan bumi.

Jika dilihat dari tren missfit tiap iterasi, maka untuk program TGMT2D pada iterasi pertama RMS Missfit yang dihasilkan masih tinggi (29.2068%), namun pada iterasi selanjutnya mengalami margin yang sangat besar dibandingkan RMS Missfit sebelumnya hingga mencapai nilai yang konstan dimulai dari iterasi ke-5. Sebaliknya, tren yang dihasilkan oleh WinGlink dari iterasi ke-1 sudah berada range toleransi missfit (4.478%) dan mengalami penurunan RMS missfit yang konstan hingga iterasi ke-30, namun margin yang sangat kecil untuk tiap iterasi. Dari pembahasan tersebut, hal yang dapat mempengaruhi adalah model awal (m_0) yang dimasukan kedalam inversi. Semakin jauh margin model awal terhadap resistivitas sebenarnya maka akan semakin besar RMS Missfit yang dihasilkan, namun karena besarnya margin antara model awal (m_0) dan resistivitas sebenarnya, maka pada iterasi selanjutnya terjadi trun penurunan RMS missfit yang signifikan. Sehingga disimpulkan bahwa penentuan model awal akan berpengaruh terhadap, waktu iterasi & jumlah iterasi.

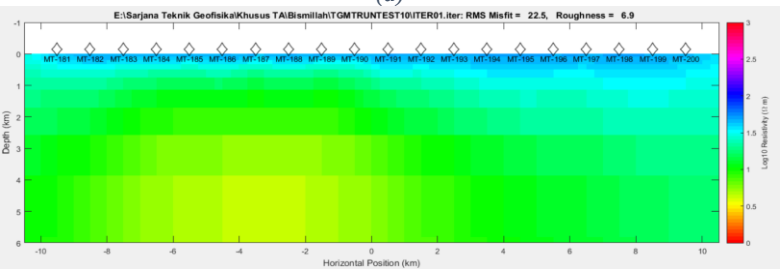
Untuk parameter roughness (tingkat kekasaran penampang hasil inversi), yang ditampilkan pada gambar 4.13, terdapat perbedaan matematik dalam menghitungnya. Untuk inversi Occam, roughness dihitung perkedalaman (nLayer+1), namun untuk inversi NLCG roughness dihitung dari

(Rough = (Nilai Fungsi Objektif – Statistik Chi Square)/TAU)

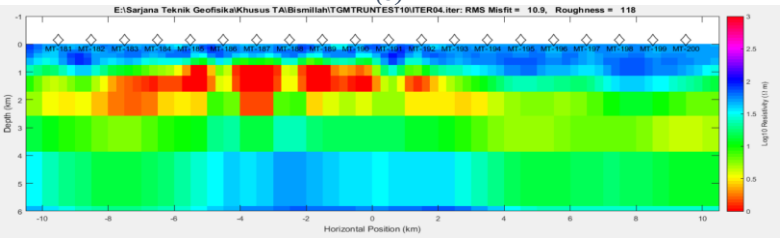
Sehingga terdapat perbedaan tren untuk menentukan nilai roughness untuk setiap nilai iterasi. Tren metode Occam adalah, semakin banya iterasi yang dilakukan , maka akan semakin tinggi roughness nya. Hal tersebut berbanding terbalik dengan metode NLCG, semakin banyak iterasi yang diakukan maka akan semakin smooth penampang resistivitas 2D.



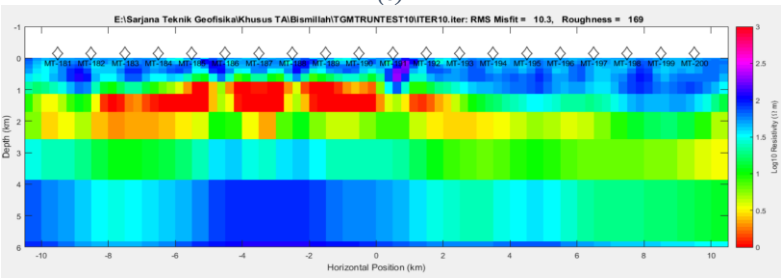
(a)



(b)



(c)



(d)

Gambar 4.14 Tren Roughness dari Metode Occam untuk model awal (a), yaitu iterasi ke-0, input parameter resistivitas sama untuk semua grid mesh. Berlanjut

ke-iterasi ke-1 (b), mulai dilakukan perkalian matematik sehingga model menjadi mendekati nilai sebenarnya, namun memiliki block model yang kasar (c), generate nilai resistivitas semakin mendekati nilai dugaan, namun kontras resistivity membuat roughness semakin besar (d).

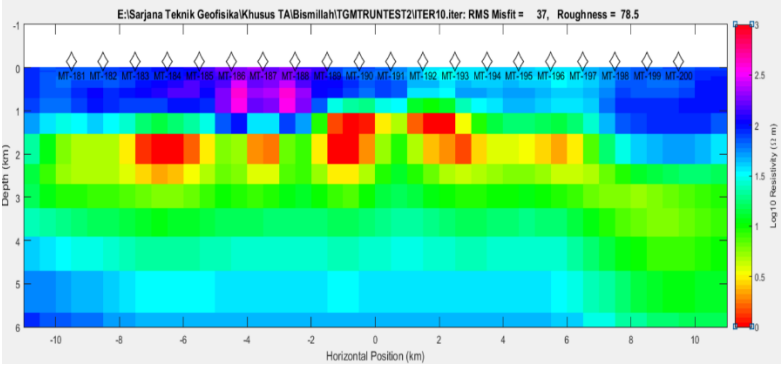
Berkebalikan dengan tren pada Gambar 4.14 dengan niai roughness yang makin besar tiap iterasi, Metode NLCG karena menggunakan prinsip penyelesaian nonlinier, maka nilai resistivitas dugaan dapat mendekati mulai dari iterasi pertama.

4.4. Analisa Parameter Inversi 2D Occam

Analisa pada subbab ini akan membahas mengenai efek perubahan input parameter inversi terhadap pemodelan resistivitas 2D yang dihasilkan. Untuk memudahkan analisis terhadap perubahan parameter, maka perubahan parameter yang lain tidak diikutsertakan pada pemodelan yang sama.

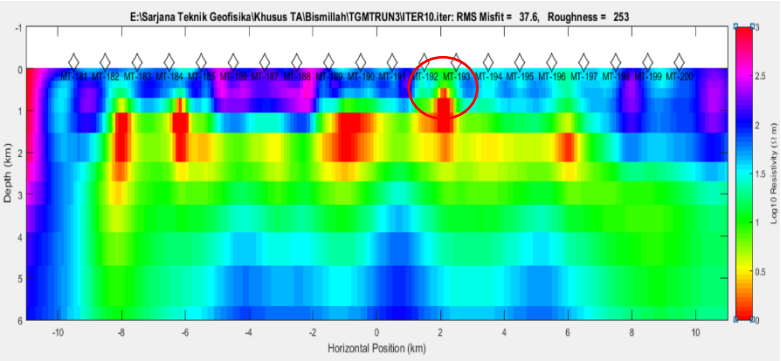
4.4.1. Mesh Horizontal

Dilakukan test parameter mesh horizontal untuk melihat respon pemodelan resistivitas 2D yang dihasilkan oleh program TGMT2D.



Gambar 4. 15 Penampang resistivitas dengan mesh grid 500m

Gambar 4.1 menunjukan pemodelan resistivitas hasil inversi 2D Occam dengan parameter mesh grid horizontal sebesar 500m. Grid ini membagi jarak tiap stasiun pengukuran menjadi 2 grid, sehingga total grid horizontal tiap layer adalah 64 dan untuk boundary (sisi kanan/kiri Titik MT-181 dan MT-200) memakai perkalian faktor skala 1.2



Gambar 4. 16 Penampang resistivitas dengan mesh grid 100m

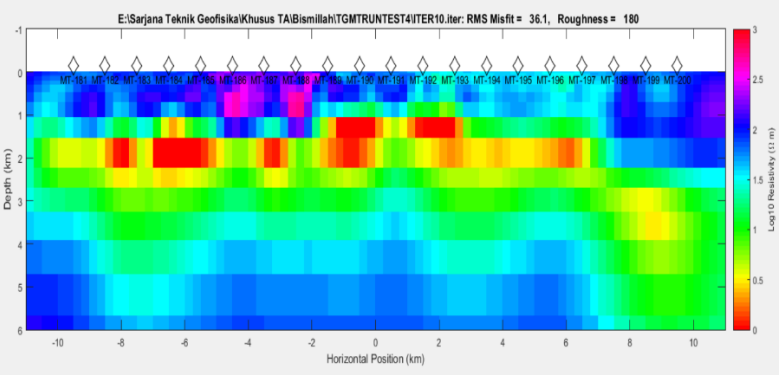
Untuk gambar 4.16 menunjukkan penampang resistivitas dengan lebar grid 100m. Mesh grid ini membagi jarak tiap stasiun pengukuran menjadi 200. Selain parameter Mesh grid horizontal, untuk mempermudah analisa maka parameter yang lain memiliki nilai yang sama.

Penampang resistivitas dengan grid 500m memiliki RMS missfit sebesar 37% dan Roughness 78, dan penampang resistivitas dengan grid 100m memiliki RMS missfit sebesar 37.6% dan Roughness 253. Hal yang menyebabkan nilai roughness yang dihasilkan oleh grid 500m lebih kecil dibandingkan dengan roughness yang dihasilkan oleh grid 100m adalah karena semakin sedikit jumlah block nilai yang mengisi mesh, maka perubahan nilai yang dihasilkan juga semakin sedikit sehingga roughness gambar 4.15 lebih kecil dibandingkan dengan gambar 4.16.

Dengan mesh grid horizontal yang kecil (gambar 4.16), penampang resistivitas yang dihasilkan lebih memiliki representasi nilai resistivitas yang baik. Namun, mesh grid yang terlalu kecil akan mengakibatkan banyaknya persebaran nilai yang sulit ditentukan sehingga menghasilkan resistivitas warna (merah) yang terpisah secara horizontal dan vertikal, seperti ditunjukkan pada lingkaran merah di gambar 4.6. Sehingga mesh horizontal yang paling baik adalah mesh yang bernilai ¼ antara titik stasiun pengukuran, sehingga penampang resistivitas yang dihasilkan memiliki representasi nilai resistivitas yang akurat dan lebih *smooth*.

4.4.2. Mesh Vertikal

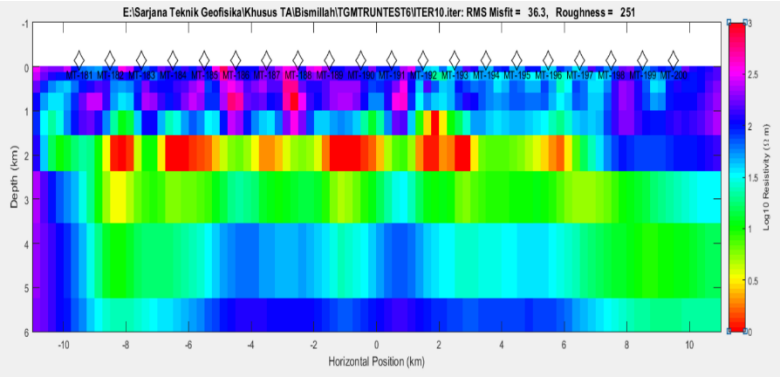
Dilakukan pemodelan resistivitas 2D dengan mengubah parameter mesh verrtikal. Untuk mesh horizontal dipakai horizontal 250m.



Gambar 4. 17 Penampang resistivitas dengan mesh vertikal awal dimulai 10 dengan fs 1.1

Dari gambar 4.17 dan gambar 4.18, nilai missfit yang dihasilkan tidak jauh berbeda, yaitu masing-masing 35.1% dan 36.3%. Dari nilai tersebut terlihat bahwa penampang resistivitas dengan nilai awal mesh grid vertikal yang kecil menghasilkan RMS missfit yang lebih kecil, walaupun dengan range nilai yang tidak terlalu besar.

Untuk nilai roughness, penampang resistivitas dengan mesh grid awal 10 memiliki roughness 180 di iterasi ke-10 dan penampang dengan mesh grid awal 50 memiliki roughness 251. Hal tersebut dikarenakan besarnya nilai awal pada mesh grid vertikal, mengakibatkan nilai resistivitas pada titik ke-x dikedalaman tertentu menjadi sulit untuk ditentukan karena besarnya blok nilai yang mengisi mesh grid secara vertikal. Dari gambar 4.17 dan 4.18 juga terlihat bahwa resistivitas dipermukaan yang dihasilkan berturut-turut didominasi oleh 2 ohm.m dan 2.5 ohm.m (kedalaman 0 – 1000m), dengan nilai sebenarnya nilai pada data sintetik adalah 2.7 ohm.m, sehingga penampang dengan nilai mesh grid vertikal yang kecil diawal, lebih bisa merepresentasikan nilai resistivitas permukaan dengan baik.



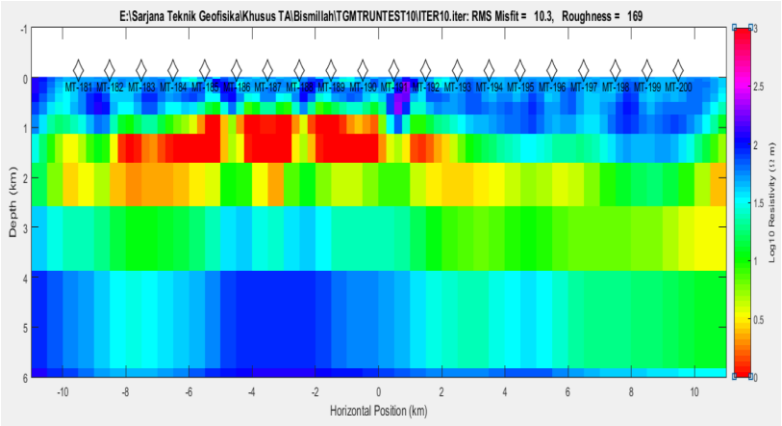
Gambar 4. 18 Penampang resistivitas dengan mesh vertikal awal dimulai dari 50 dan fs 1.1

Sehingga untuk menentukan nilai awal mesh grid secara vertikal adalah dengan memasukkan input yang kecil (dibawah 20m), sehingga nilai resistivitas didekat permukaan dapat direpresentasikan dengan baik.

Kerugian lain yang diperoleh saat memilih mesh grid vertikal awal yang besar adalah susahnya merepresentasikan nilai lapisan tipis yang mungkin merupakan target yang bisa membantu interpretasi struktur bawah permukaan didaerah penelitian.

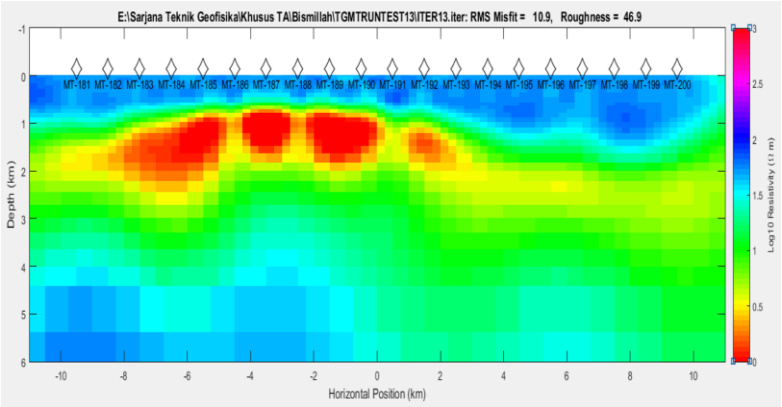
4.4.3. *Factor Scale* Vertikal

Factor scale vertikal merupakan konstanta pengali yang menentukan nilai mesh grid di n+1, sehingga semakin besar input konstanta *factor scale* maka lebar mesh grid yang dihasikan akan semakin besar dan akan berpengaruh kepada waktu iterasi, maksimum kedalaman, RMS missfit dan roughness.



Gambar 4. 19 Penampang resistivitas dengan faktor scale terhadap mesh grid vertikal senilai 1.5

Gambar 4.19 dan gambar 4.20 menunjukkan penampang resistivitas dengan faktor scale 1.5 dan 1.1. Dari gambar 4.19 memiliki nilai RMS missfit 10.3 dengan roughness 169 dan gambar 4.20 memiliki nilai RMS missfit 10.9 dengan roughness 46.9.



Gambar 4. 20 Penampang resistivitas dengan faktor scale terhadap mesh grid vertikal senilai 1.1

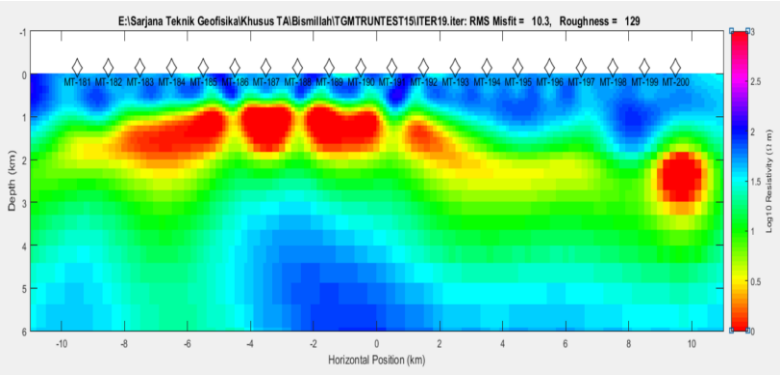
Dilihat dari nilai missfit pada iterasi ke-10 untuk gambar 4.19 dan iterasi ke-13 untuk gambar 4.20, dimana RMS missfit pada gambar 4.19 lebih kecil dibandingkan pada gambar 4.20 menunjukan bahwa RMS missfit dihitung berdasarkan kecocokan data hasil iterasi dengan data sebelumnya, sehingga tidak bisa dijadikan acuan utama dalam menentukan kesesuain data hasil inversi dengan data sebenarnya.

Rougness yang dihasilkan oleh pemodelan *factor scale* yang lebih besar juga menyebabkan roughnes yang besar dan begitu juga sebaliknya. Hal tersebut dikarenakan besarnya margin antara satu titik perubahan resistivitas dengan titik yang lain, sehingga gradasi nilai resistivitas yang dihasilkan sangat tidak *smoth.*

Factor scale yang baik umumnya memiliki nilai yang tidak terlalu besar, sehingga nilai-nilai resistivitas yang sebenarnya masih dapat direpresentasikan dengna baik.

4.4.4. Debug Level

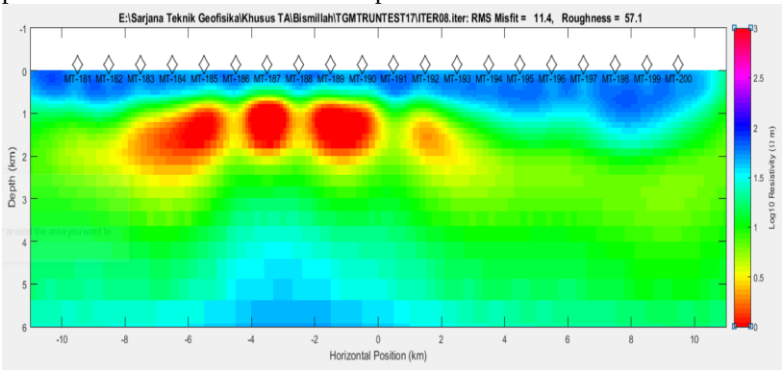
Debug level merupakan cara program dalam menampilkan proses iterasi yang sedang berlangsung dan memaksimalkan hasil ditiap iterasi dengan cara mencari nilai terkecil dari selisih forward dan model yang dihasilkan.



Gambar 4. 21 Penampang resistivitas dengan setting debug level 2

Gambar 4.21 dan gambar 4.22 menampilkan penampang resistivitas dengan setting debug level 2 dan 1. Bentuk struktur resistivitas yang dihasilkan menunjukkan bahwa settingan debug level 2 memberikan representasikan lapisan merah (10ohm.m) dengan cukup baik secara

lateral, dimana struktur resistivitas rendah tersebut dapat direpresentasikan hingga titik stasiun MT-120. Walaupun terjadi penurunan nilai resistivitas rendah pada titik MT-199.



Gambar 4. 22 Penampang resistivitas dengan nilai debug level 1

Dari perubahan debug level ternyata juga mempengaruhi nilai RMS missfit dan roughness yang dihasilkan. RMS missfit untuk penampang resistivitas debug 2 memiliki nilai 10.3% dan roughness 129. RMS missfit untuk penampang resistivitas debug 1 memiliki nilai 11.4% dan roughness 57.1.

Perbedaan nilai misfit dan nilai roughness tersebut disebabkan oleh cara kerja debug level dalam melakukan iterasi, dimana debug level 2 melakukan maksimalisasi terhadap jumah iterasi yang dilakukan.

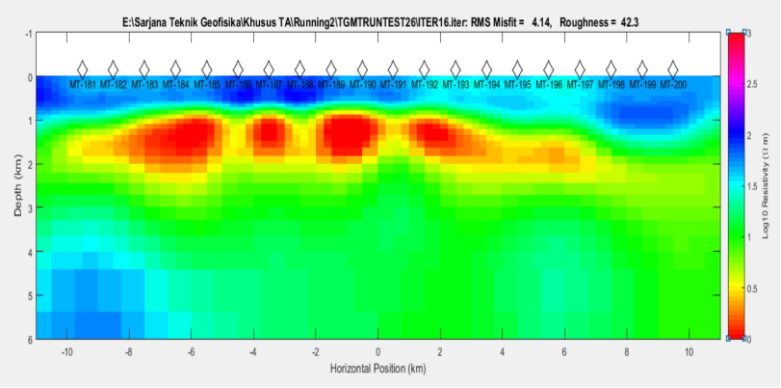
Jika dicermati terdapat perbedaan struktur resistivitas di kedalaman 3000m kebawah. Untuk model sintetik, nilai resistivitas sebenarnya beradi di nilai 2.6 ohm meter (Log 10), dan untuk penampang resistivitas debug level 1 dan 2 terlihat bahwa debug 2 bisa merepresentasikan batas perlapisan (biru) dengan lebih baik dibandingkan dengan stting debug level 1.

Namun diperlukan waktu itersi yang lebih lama untuk mencapai maksimum iterasi jika melalukan setting debug level 2, bahkan mencapai 2 kali lipat waktu yang dibutuhkan untuk menyelesaikan proses iterasi.

4.4.5. Input Data Error

Input error merupakan input error yang dimasukkan di list Data File, ada dua cara dalam menentukan nilai input error ini pada umumnya.

Yang pertama adalah dengan memasukkan nilai error Rhoxx, Rhoxy, Rhoyx, dan Rhoyy dari hasil pengolahan data yang telah dilakukan hingga file berekstensi edi. Alternatif lain adalah dengan melakukan input manual dengan memasukkan nilai error yang minim.



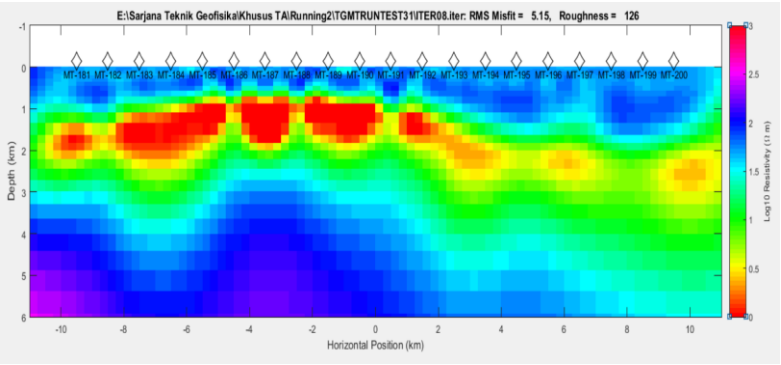
Gambar 4. 23 Input error Rho 0.01 dan Phase 1

Dalam memasukkan nilai error untuk log10 x nilai resistivitas, terdapat sedikit perbedaan. Perhitungan yang dilakukan adalah sebagai berikut:

$$\frac{d(\log\_10(x))}{dx} = \frac{1}{[x\ln(10)]}$$

Jadi untuk 10% error, maka nilai yang dimasukan adalah 0.0434.

Karena output plotting dari TGMT2D adalah log10 x nilai resistivitas, maka dilakukan konversi tersebut.



Gambar 4.24 input error Rho 0.02 dan phase 2

Dari gambar 4.23 dan 4.24 terlihat perbedaan struktur resistivitas yang sangat jauh. Pada gambar 4.23, resistivitas tinggi dikedalaman 3000m tidak dapat direpresentasikan dengan baik, bahkan hanya direpresentasikan $10^{1.3}$ Ohm.m dari yang sebenarnya adlah $10^{2.7}$ Ohm.m. Untuk gambar 4.24, dimana input yang dimasukan adalah 0.02(resistivitas semu) dan 2 (phase) cukup baik dalam merepresentasikan nilai resisivitas model sintetik tersebut.

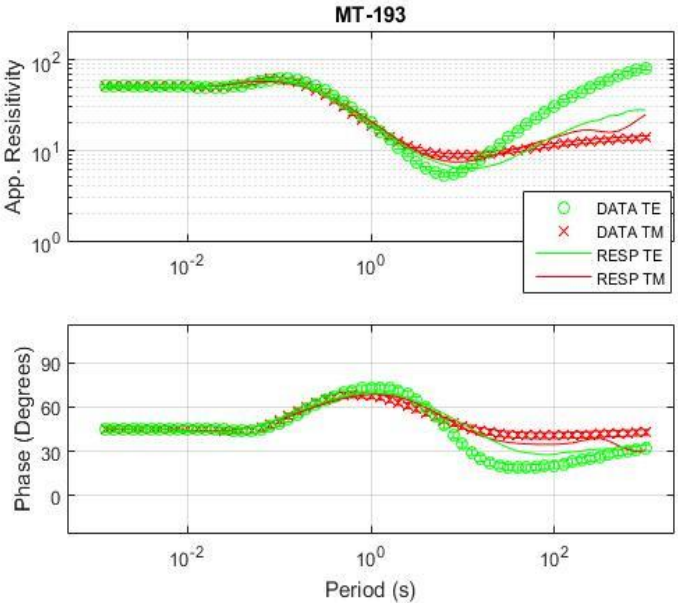
Nilai RMS missfit dan roughness juga berbeda pada hasil permodelannya. Untuk gambar 4.23 missfit dan roughness yang dihasilkan berturut-turut adalah 4.14 dan 42.3. Untuk gambar 4.24 nilai RMS missfit dan roughnessnya adalah 5.15 dan 126.

Dari hal diatas, bahwa sebelum memasukkan nilai error, apabila perhitungan output dari plotting menggunakan format log10xNilai resistivitas, maka perlu dilakukan perubahan nilai terlebih dahulu sehingga model yang dihasilkan dapat merepresentasikan model sebenarnya. Semakin kecil nilai error yang dimasukkan maka akan semakin kecil nilai RMS missfit nya namun belum tentu dapat merepresentasikan struktur resistivitas dengan baik.

4.5. Analisis Kurva Resistivitas dan Fasa

Analisis kurva resistivitas semu dan fase dilakukan untuk melihat tren nilai *apparent resistivity* dan *phase* terhadap perioda.

Pada pembahasan kurva Resistivitas semu dan phase hanya diambil satu contoh random, yaitu Titik MT-193. Dari gambar terlihat nilai resistivitas pada perioda awal berada di apparent resistivity tinggi (orde 2) yang menunjukkan lapisan di permukaan dengan nilai dari model data sintetik adalah 50 Ohm.m. Kemudian nilai resistivitas semu bergerak turun yang merepresentasikan lapisan apparent resistivity rendah (<10Ohm.m). Kemudian kurva resistivitas semu naik keatas yang merepresentasikan lapisan konduktif secara kontras.



Gambar 4. 25 Kurva resistivitas semu dan phase Titik stasiun MT-193

Umumnya target utama dalam pengolahan data metode magnetotellurik adalah nilai resistivitas yang berada di bawah 10^1 yang merupakan representatif dari lapisan penudung.

Data TE dan TM yang ditunjukkan dalamm kurva diasumsikan merupakan data yang telah dirotasi terhadap sumbu strike. Response merupakan nilai resistivitas semu dan fase setelah dilakukan inversi 2D metode Occam.

Karena data TE dan TM dirotasi terhadap sumbu strike, maka seharusnya kurva data TE dan TM saling berimpit, hal tersebut terlihat setelah dilakukan inversi terhadap data TE dan TM. Sehingga resistivitas semu dikedalaman 4000m dapat diinterpretasi dengan baik.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Kesimpulan yang diperoleh dari penelitian Tugas Akhir ini adalah sebagai berikut;

1. Dari percobaan yang dilakukan terhadap model data sintetik, untuk mendapatkan penampang resistivitas hasil inversi 2D Occam yang baik sangat ditentukan oleh model awal yang dibuat, parameter inversi, dan data error.
2. Untuk menentukan nilai minimum RMS missfit dan minimum roughness, inversi Occam tidak bisa membuktikan nilai global minimum, test yang dilakukan didata sintetik mengindikasikan solusi yang konvergen terjadi saat model awal dibuat dengan bentuk *half-space* dan pengaruh *faktor smoothing.*
3. Parameter yang sangat berpengaruh terhadap kecocokan data dan model adalah besarnya mesh horizontal, besarnya mesh vertikal, faktor *scale*, dan data error.
4. Nilai resistivitas dari hasil inversi program TGMT2D, lapisan pertama 32-79 ohm.m, lapisan kedua 1.2-3.9 ohm.m, lapisan ketiga 2.5-79 Ohm.m, dan lapisan keempat dengan 32-200 ohm.m dengan RMS error 5.15 dan Roughness 126.

5.2. Saran

1. Perlu dilakukan pengembangan terhadap program, terutama untuk masalah pembuatan model awal secara default tetapi menghasilkan RMS missfit dan Roughness yang minimum.
2. Perlu dilakukan pengembangan terhadap fleksibilitas input data file ekstensi *edi.
3. Perlu dilakukan pengembangan program sesuai blueprint yang telah dibuat, terutama dalam menyempurnakan inversi 2D dan forward modelling.

[Halaman Ini Sengaja Dikosongkan]

DAFTAR PUSTAKA

Coggon, J. H., 1971, *Electromagnetic and electrical Modeling by The Finite element Methods, Geophysics, 36, 132-155.*

Daud, Yunus. 2010. Diktat Kuliah : *Metode* Magnetotellurik *(magnetotelurik).* Laboratorium Geofisika, FMIPA Universitas Indonesia.

Green, Alisa Marie, 2003, *Magnetotelluric crustal studies in Kenai, Alaska, Golden Colorado.*

Kauffman, A dan G. V. Keller. 1981, *The Magnetotelurik Sounding Method,* Elsevier, Amsterdam.

Oskooi, B. 2005. *1D Interpretation of The* Magnetotellurik *Data from Travale Geothermal Field in Italy, Jurnal of The Earth & Space Physics.* Vol. 32, No. 2, 2006.

Rodi, William and Mackie, Randall L., 2001, *Nonlinier Conjugate Gradient Algorithm for 2D Magnetotelluric Inversion,* Geophysics Vol. 66, No. 1; page 174-187

Srigutomo, W., 1997, *Pemodelan Elektromagnetik 2D Menggunakan Metode Elemen Hingga Untuk Sumber Alami dan Sumber Arus Garis, Tesis Magister,* Jurusan Fisika ITB.

Simpson F. dan Bahr K., 2005, "*Practical Magnetotelurik",* Cambridge University

DeGroot-H and Constable, "*Occam inversion to generate smooth, two dimension models from magnetotellurik data".* Geophysics, vol 55, no 12 P.1613-1624, 1990

De Lugao, P.P., P.Wannamaker, 1996. Calculating the two-Dimensional magnetotelluric Jacobian in finite elements using reciprocity. Geophys, J Int., 127 806-810,.

Wannamaker, P.E., J.A.Stodt, and L.Rijo, 1995. A stable finete element solution for two dimensional Magnetotellurik Modeling, Geophysical Journal of the Royal Astronomical Society.

Lampiran I (GUI TGMT 2D)

```
import sys
from PyQt4.uic import loadUiType
from PyQt4 import QtGui
from PyQt4.QtGui import (QPushButton, QLineEdit,
QPlainTextEdit, QTextEdit)
import matplotlib.pyplot as plt
import math
import numpy as np

import rumusEdi

#.................................................ANY
LIST.......#
#..................LIST MESH
#....List SaveMeF
listlefthora = []
listrighthor = []
listtengah = []
minvertikal = []
firstbelowbrick = []
gonomi = []
listbelowver = []
horizontalmesh = []
vertikalmesh = []
nlayer_1 = []
nlayer_2 = []
nlayer_3 = []
nlayer_sisa = []
#...List GenerateMeF
listlefthorag = []
listrighthorg = []
listtengahg = []
minvertikalg = []
firstbelowbrickg = []
gonomig = []
listbelowverg = []
horizontalmeshg = []
vertikalmeshg = []

#................LIST MODEL
#...List SaveMoF
baristengahhorizontal_1 = []
baristengahhorizontal_2 = []
baristengahhorizontal_3 = []
barishorizontal_1 = []
barishorizontal_2 = []
barishorizontal_3 = []
```

```
isiboundary = []
sumhor_1 = []
sumhor_2 = []
sumhor_3 = []
sumhor_4 = []
jumlahparamx = []
sumlayer = []
nverlayer_1 = []
nverlayer_2 = []
nverlayer_3 = []
vernod_1 = []
vernod_2 = []
vernod_3 = []
vertikalnodes = []
descriplayer = []
coro =[]

#..........LIST DATA FORM
listSite = []
listFreq = []
listDataType = []
listDatum = []
listError = []


Ui_MainWindow, QMainWindow = loadUiType('startmodeldata.ui')

class tabparameter(QMainWindow,Ui_MainWindow):
    def __init__(self, ):
        super(tabparameter,self).__init__()
        self.setupUi(self)

        #....tabStartup

self.SaveButtonstartup.clicked.connect(self.save_startup)

self.ClearButtonstartup.clicked.connect(self.clear_startup)

        #....tabMesh
        self.tombolsave_2.clicked.connect(self.save_MeF)
        self.tombolclear_2.clicked.connect(self.clear_MeF)
        self.tombolkeluar_2.clicked.connect(self.close_MeF)
        self.tombolGen_2.clicked.connect(self.generate_MeF)

        #....tabModel
        self.TombolmodGen.clicked.connect(self.generate_MoF)

self.ClearButtonmodel.clicked.connect(self.clear_MoF)
        self.SaveButtonmodel.clicked.connect(self.save_MoF)
```

```
        #....tabDATA
        self.Openfile.clicked.connect(self.open_data)
        self.SaveButtondata.clicked.connect(self.save_data)

self.Clearbuttondata.clicked.connect(self.clear_startup)

self.CloseButtondata.clicked.connect(self.close_windows)

#
.............................................................
.............................DEF TAB DATA
    def open_data(self):
        bukafile = QtGui.QFileDialog.getOpenFileNames(self,
"Open Data File", "", "Data Edi(*.edi)")
        Freq, Perioda, RHOxy, RHOyx, PHASEyx, PHASExy =
rumusEdi.FunOpenEdi(bukafile)
        self.writeToForm(Freq, Perioda, RHOxy, RHOyx,
PHASEyx, PHASExy, bukafile)

    def writeToForm(self, Freq, Perioda, RHOxy, RHOyx,
PHASEyx, PHASExy, bukafile):
        # ---------------------------------------------
-------------------------Form Sites
        for i in range(len(bukafile)):
            tmp = (bukafile[i])
            tmp = tmp.split("\\")
            n = len(tmp) - 1
            tmp = tmp[n].split('.')
            self.textSites.append(str(tmp[0]))  # detail
sites
        self.lineSites.setText(str(len(bukafile)))  # jumlah
sites

        k = 0

        a = len(bukafile) - 1
        a = a / 2
        b = -(a * 1000) - 4000
        self.linebinding.setText(str(b))
        for nu in range(len(bukafile)):
            amp = nu * 1000 - (a * 1000)
            self.textOffset.append(str(amp))  # Kolom isi
offset (margin offset 1km)


        ump = (Freq[0])
        for item in ump:
            self.textFreq.append(str(item))  # kolom list
frekuensi
```

```
        for u in range(len(Freq)):
            self.lineFreq.setText(str(len(Freq[u])))  # line
jumlah frekuensi

        for g in range(len(bukafile)):
            for h in range(len(Freq[g])):
                typeTERho = 1
                typeTEPhase = 2
                typeTMRho = 5
                typeTMPhase = 6
                datatype = [typeTERho, typeTEPhase,
typeTMRho, typeTMPhase]

self.textvertdatum.append(str(np.log10(RHOyx[g][h])))
                listDatum.append(np.log10(RHOyx[g][h]))

self.textvertdatum.append(str(PHASEyx[g][h]))
                listDatum.append(PHASEyx[g][h])

self.textvertdatum.append(str(np.log10(RHOxy[g][h])))
                listDatum.append(np.log10(RHOxy[g][h]))

self.textvertdatum.append(str(PHASExy[g][h]))
                listDatum.append(PHASExy[g][h])
                for i in datatype:
                    self.textvertsite.append(str(g + 1))  #
kolom titik pengukuran
                    listSite.append(g + 1)
                    self.textvertfreq.append(str(h + 1))  #
kolom pengulangan nilai pfrekuensi
                    listFreq.append(h + 1)
                    self.textvertdatatype.append(str(i))  #
kolom type data
                    listDataType.append(i)
                    self.textverterror.append(str('%1.2f' %
(0.02)))  # kolom error, sementara 0.1
                    listError.append(0.02)
                    self.textverterror.append(str(2))
                    listError.append(2)
                    k += 1

                    self.linedatblock.setText(str(k))
        self.dataBlock = k

    def save_data(self):
        name = QtGui.QFileDialog.getSaveFileName(self,
'Simpan File')
        file = open(name, 'w')

        textlineFormat = self.lineFormat.text()
```

```
        file.writelines('FORMAT:            ' +
textlineFormat + '\n')

        textlinetitle = self.Linetitle.text()
        file.writelines('TITLE:             ' + textlinetitle
+ '\n')

        textlinesite = self.lineSites.text()
        file.writelines('SITES:             ' + textlinesite
+ '\n')

        texteditsites = self.textSites.toPlainText()
        file.writelines(texteditsites + '\n')

        textlineoffset = self.lineoffset.text()
        file.writelines('OFFSET (M):        ' +
textlineoffset + '\n')

        texteditoffset = self.textOffset.toPlainText()
        file.writelines(texteditoffset + '\n')

        textlinefreq = self.lineFreq.text()
        file.writelines('FREQUENCIES:       ' + textlinefreq
+ '\n')

        texteditfreq = self.textFreq.toPlainText()
        file.writelines(texteditfreq + '\n')

        textlinedatblock = self.linedatblock.text()
        file.writelines('DATA BLOCKS:       ' +
textlinedatblock + '\n')

        file.writelines(('SITE' + '\t' + 'FREQ' + '\t' +
'DATA TYPE' + '\t' + 'DATUM' + '\t' + 'ERROR') + '\n')

        for i in range(self.dataBlock):
            file.writelines('%6i' '%6i' '%6i' '%10.5f'
'%10.4f' % (
            listSite[i], listFreq[i], listDataType[i],
listDatum[i], listError[i]))

            file.writelines('\n')

        file.close()

    def clear_startup(self):
        self.Linetitle.clear()
        self.lineSites.clear()
        self.textSites.clear()
        self.textOffset.clear()
```

```
        self.lineFreq.clear()
        self.textFreq.clear()
        self.linedatblock.clear()
        self.textvertsite.clear()
        self.textvertfreq.clear()
        self.textvertdatatype.clear()
        self.textvertdatum.clear()
        self.textverterror.clear()
        # self.paraminput.clear()

    def close_windows(self):
        main.destroy()


#.............................................................
.................................DEF TAB MESH
    def save_MeF(self):
        name = QtGui.QFileDialog.getSaveFileName(self,
'Simpan File')
        file = open(name, 'w')

        deskripsi = self.linedeskripsi_2.text()
        hornodes = int(self.linehorizontalnode_2.text())
        vernodes = int(self.lineverticalnode_2.text())
        minblockhor = int(self.lineminblockhor_2.text())
        minblockver = int(self.lineminblockver_2.text())
        facH = float(self.linefactorhor_2.text())
        facV = float(self.linefactorver_2.text())
        boundary = int(self.lineboundary_2.text())
        textverdimensi =
int(self.lineverticaldimension.text())

        minvertikal.append(int(minblockver))
        sisaantiboundary = int((vernodes) - 2)
        midpoint = int((hornodes - 1) - (2 * boundary))  #
...cari pengulangan titik tengah

        #-----------------------------------------------------
------------------------------MEMBUAT NUMLAYER DARI
VERNODES

#.....................................................DEFIN
ISI PEMBAGIAN GRID VERNODES
        vernodes_1 = int((vernodes-1)/2)
        vernodes_2 = int((vernodes-1)/3)
        vernodes_3 = (vernodes-1) - (vernodes_1 +
vernodes_2)

        textverdimensi_2 = textverdimensi + 1
        textverdimensi_3 = textverdimensi_2 + 1
```

```
#.....................................................DEFIN
ISI PEMBAGIAN NUM LAYER
        layer_1 = int(vernodes_1/textverdimensi)
        nlayer_1.append(str(layer_1))
        layer_2 = int(vernodes_2/textverdimensi_2)
        nlayer_2.append(str(layer_2))
        layer_3 = int(vernodes_3/textverdimensi_3)
        nlayer_3.append(str(layer_3))

        #...layer sisa(menyiasati pembagian yang tidak habis
dibagi 0
        layer_4 = (vernodes_1-(layer_1*textverdimensi))
        layer_5 = (vernodes_2-(layer_2*textverdimensi_2))
        layer_6 = (vernodes_3-(layer_3*textverdimensi_3))
        layer_sisa = (layer_4 + layer_5 + layer_6)
        nlayer_sisa.append(layer_sisa)


        banyaklayer = layer_1+layer_2+layer_3+1
        print(banyaklayer)
        numlayer =
self.linenumlayer.setText(str(banyaklayer))

        # ------------------------------------------------
------------------------------------------------------
HORIZONTAL
        # ------------------------------------------------
------------------------------------------------------
boundary

        for i in range(boundary):
            if i == 0:
                rightbrick = (facH) * (minblockhor)
                listrighthor.append(int(rightbrick))  #
kanan
            else:
                rightbrick = (facH) * (rightbrick)
                listrighthor.append(int(rightbrick))  #
kanan

            listlefthora = sorted(listrighthor,
reverse=True)  # kiri

            # ------------------------------------------
---------------------- -middle

        for i in range(midpoint):
            yui = minblockhor
```

```
            listtengah.append(int(yui))

            # ------------------------------------------------
------------------------- VERTICAL

        for i in range(sisaantiboundary):  # ...mau mencari
mesh vertikal(factor scale Vertikal)
            if i == 0:
                belowbrick = (facV) * (minblockver)
                firstbelowbrick.append(int(belowbrick))
            else:
                belowbrick = (facV) * (belowbrick)
                listbelowver.append(float(belowbrick))

        # <------Batas Vertikal Mesh------->

        horizontalmesh.extend(listlefthora + listtengah +
listrighthor)  # ....Horizontal Mesh

        vertikalmesh.extend(minvertikal + firstbelowbrick +
listbelowver)  # ......Vertikal Mesh

        file.writelines(deskripsi + '\n')
        file.writelines('%6i' '%5s' '%5s' '%6i' '%4i' '%4i'
% (0, hornodes, vernodes, 0, 0, 2) + '\n')

        for i in range(len(horizontalmesh)):
            file.write(str(horizontalmesh[i]) + '\t')
        file.writelines('\n')

        for i in range(len(vertikalmesh)):
            file.write(str(vertikalmesh[i]) + '\t')
        file.writelines('\n')

        file.writelines('%6i' % (0) + '\n')

        kolom = ('?')

        for i in range((vernodes - 1) * 4):
            file.writelines(kolom + '')
            for j in range(hornodes - 2):
                file.writelines(kolom + '')
            file.writelines('\n')


        file.close()

    def generate_MeF(self):

        deskripsi = self.linedeskripsi_2.text()
```

```
        hornodes = int(self.linehorizontalnode_2.text())
        vernodes = int(self.lineverticalnode_2.text())
        minblockhor = int(self.lineminblockhor_2.text())
        minblockver = int(self.lineminblockver_2.text())
        facH = float(self.linefactorhor_2.text())
        facV = float(self.linefactorver_2.text())
        boundary = int(self.lineboundary_2.text())

        minvertikalg.append(int(minblockver))
        sisaantiboundary = int((vernodes) - 2)
        midpoint = int((hornodes - 1) - (2 * boundary))  #
...cari pengulangan titik tengah

        # ------------------------------------------------
------------------------------- HOROZONTAL
        # ------------------------------------------------
-------------------------------- boundary

        for i in range(boundary):
            if i == 0:
                rightbrick = (facH) * (minblockhor)
                listrighthorg.append(int(rightbrick))  #
kanan
            else:
                rightbrick = (facH) * (rightbrick)
                listrighthorg.append(int(rightbrick))  #
kanan

            listlefthorag = sorted(listrighthorg,
reverse=True)  # kiri

            # ------------------------------------------------
---------------------------------- -middle

        for i in range(midpoint):
            yoi = minblockhor
            listtengahg.append(int(yoi))

            # ------------------------------------------------
---------------------------------- VERTICAL

        for i in range(sisaantiboundary):  # ...mau mencari
mesh vertikal(factor scale Vertikal)
            if i == 0:
                belowbrick = (facV) * (minblockver)
                firstbelowbrickg.append(int(belowbrick))
            else:
                belowbrick = (facV) * (belowbrick)
                listbelowverg.append(float(belowbrick))
```

```
        # <------Batas Vertikal Mesh------->

        horizontalmeshg.extend(listlefthorag + listtengahg +
listrighthorg)  # ....Horizontal Mesh

        vertikalmeshg.extend(minvertikalg + firstbelowbrickg
+ listbelowverg)  # ......Vertikal Mesh

        self.linegenerate_2.append(str(deskripsi) + '\n')
        self.linegenerate_2.append('%6i' '%5s' '%5s' '%6i'
'%4i' '%4i' % (0, hornodes, vernodes, 0, 0, 2) + '\n')

        self.linegenerate_2.append(str(horizontalmeshg) +
'\t')
        self.linegenerate_2.append('\n')

        self.linegenerate_2.append(str(vertikalmeshg) +
'\t')
        self.linegenerate_2.append('\n')
        self.linegenerate_2.append('%6i' % (0) + '\n')
        kolom = ('?')

        for i in range((vernodes - 1) * 4):
            self.linegenerate_2.append('%1s' % (kolom) + '')
            for j in range(hornodes - 2):
                self.linegenerate_2.append('%1s' % (kolom) +
'')
            self.linegenerate_2.append('\n')

    def clear_MeF(self):
        self.linedeskripsi_2.clear()
        self.linehorizontalnode_2.clear()
        self.lineverticalnode_2.clear()
        self.lineminblockhor_2.clear()
        self.lineminblockver_2.clear()
        self.linefactorhor_2.clear()
        self.linefactorver_2.clear()
        self.lineboundary_2.clear()

    def close_MeF(self):
        main.destroy()

#.............................................................
.................................DEF TAB MODEL
    def save_MoF(self):
        name = QtGui.QFileDialog.getSaveFileName(self,
'Simpan File')
        file = open(name, 'w')

        textmodelname = self.linemodel.text()
```

```
        textmesh = self.linemesh.text()
        textbinding = self.linebinding.text()
        textnumlayer = int(float(self.linenumlayer.text()))
        texthordimensi =
int(self.linehorizontaldimension.text())
        textverdimensi =
int(self.lineverticaldimension.text())

        vernodes = int(self.lineverticalnode_2.text())
        hornodes = int(self.linehorizontalnode_2.text())
        boundary = int(self.lineboundary_2.text())

        # ......input formula

        isiboundary.append(boundary) #boundary
        sumhorizontal = ((hornodes - 1) - (boundary * 2) +
2)

        #......Cari solusi agar pembagian habis dibagi nol
'0', sehingga tidak ada float.

        for i in range(int((sumhorizontal - 2) /
texthordimensi)):
            baristengahhorizontal_1.append(texthordimensi)

        for i in range(int((sumhorizontal - 2) /
(texthordimensi + 1))):
            baristengahhorizontal_2.append(texthordimensi +
1)

        for i in range(int((sumhorizontal - 2) /
(texthordimensi + 2))):
            baristengahhorizontal_3.append(texthordimensi +
2)

        barishorizontal_1.extend(isiboundary +
baristengahhorizontal_1 + isiboundary)
        sumhorizontal_1 = len(barishorizontal_1)  # jumlah
variabel dalam list horizontal layer 1
        barishorizontal_2.extend(isiboundary +
baristengahhorizontal_2 + isiboundary)
        sumhorizontal_2 = len(barishorizontal_2)  # jumlah
variabel dalam list horizontal layer 2
        barishorizontal_3.extend(isiboundary +
baristengahhorizontal_3 + isiboundary)
        sumhorizontal_3 = len(barishorizontal_3)  # jumlah
variabel dalam list horizontal layer 3


        # ------------------------------------------------
```

```
--------------------------------MEMBUAT NUMLAYER DARI
VERNODES
        #
.....................................................DEFINI
SI PEMBAGIAN GRID VERNODES
        vernodes_1 = int((vernodes - 1) / 2)
        vernodes_2 = int((vernodes - 1) / 3)
        vernodes_3 = (vernodes - 1) - (vernodes_1 +
vernodes_2)

        textverdimensi_2 = textverdimensi + 1
        textverdimensi_3 = textverdimensi_2 + 1

        #
.....................................................DEFINI
SI PEMBAGIAN NUM LAYER
        layer_1 = int(vernodes_1 / textverdimensi)
        layer_2 = int(vernodes_2 / textverdimensi_2)
        layer_3 = int(vernodes_3 / textverdimensi_3)

        # ...layer sisa(menyiasati pembagian yang tidak
habis dibagi 0
        layer_4 = (vernodes_1 - (layer_1 * textverdimensi))
        layer_5 = (vernodes_2 - (layer_2 *
textverdimensi_2))
        layer_6 = (vernodes_3 - (layer_3 *
textverdimensi_3))
        layer_sisa = (layer_4 + layer_5 + layer_6)
        print (layer_sisa)

        numlayer_1 = layer_1
        numlayer_2 = layer_2
        numlayer_3 = layer_3
        numlayer_4 = layer_sisa



        # ......Write to Paper
        file.writelines('FORMAT:           OCCAM2MTMOD_1.0'
+ '\n')
        file.writelines('MODEL NAME:       ' + textmodelname
+ '\n')
        file.writelines('DESCRIPTION:      Created with
UIMESH.m' + '\n')
        file.writelines('MESH FILE:        ' + textmesh +
'\n')
        file.writelines('MESH TYPE:        PW2D' + '\n')
        file.writelines('STATICS FILE:     none' + '\n')
        file.writelines('PREJUDICE FILE:   none' + '\n')
        file.writelines('BINDING OFFSET:   ' + textbinding +
```

```
'\n')
        file.writelines('NUM LAYERS:       ' +
str(textnumlayer) + '\n')


        for i in range(int(numlayer_1)):
            file.writelines('%1i' '%4i' % (textverdimensi,
sumhorizontal_1) + '\n')
            vernod_1.append(textverdimensi)
            sumhor_1.append(sumhorizontal_1)
            for j in range(len(barishorizontal_1)):
                file.writelines('%2s' %
(barishorizontal_1[j]) + '')
            file.writelines('\n')
        for i in range(int(numlayer_2)):
            file.writelines('%1i' '%4i' % (textverdimensi_2,
sumhorizontal_2) + '\n')
            vernod_2.append(textverdimensi_2)
            sumhor_2.append(sumhorizontal_2)
            for k in range(len(barishorizontal_2)):
                file.writelines('%2s' %
(barishorizontal_2[k]) + '')
            file.writelines('\n')
        for i in range(int(numlayer_3)):
            file.writelines('%1i' '%4i' % (textverdimensi_3,
sumhorizontal_3) + '\n')
            vernod_3.append(textverdimensi_3)
            sumhor_3.append(sumhorizontal_3)
            for l in range(len(barishorizontal_3)):
                file.writelines('%2s' %
(barishorizontal_3[l]) + '')
            file.writelines('\n')
        for i in range(1):
            file.writelines('%1i' '%4i' % (layer_sisa,
sumhorizontal_3) + '\n')
            sumhor_4.append(sumhorizontal_3)
            for m in range(len(barishorizontal_3)):
                file.writelines('%2s' %
(barishorizontal_3[m]) + '')
            file.writelines('\n')
        file.writelines('NO. EXCEPTIONS:   0')

        jumlahparam = sum(sumhor_1 + sumhor_2 + sumhor_3 +
sumhor_4)  # untuk param di startup
        jumlahparamx.append(jumlahparam)
        self.lineiteration.setText(str(jumlahparam))


        rhoawal = ('0.2000000E+01   0.2000000E+01
0.2000000E+01   0.2000000E+01   0.2000000E+01')
```

```
        barisparam = int(jumlahparam/5)
        barisparam_sisa = jumlahparam - barisparam
        totalbarisparam = barisparam
        for i in range(int(totalbarisparam)):
            self.paraminput.append(rhoawal)
            file.writelines('\t')

        file.close()

    def generate_MoF(self):

        textmodelname = self.linemodel.text()
        textmesh = self.linemesh.text()
        textbinding = self.linebinding.text()
        textnumlayer = int(float(self.linenumlayer.text()))
        # textlayeri = self.linelayeri.text()
        texthordimensi =
int(self.linehorizontaldimension.text())
        textverdimensi =
int(self.lineverticaldimension.text())

        hornodes = int(self.linehorizontalnode.text())
        boundary = int(self.lineboundary.text())

        # ......input formula

        isiboundary.append(boundary)
        sumhorizontal = ((hornodes - 1) - (boundary * 2) +
2)

        for i in range(int((sumhorizontal - 2) /
texthordimensi)):
            baristengahhorizontal_1.append(texthordimensi)

        for i in range(int((sumhorizontal - 2) /
(texthordimensi + 1))):
            baristengahhorizontal_2.append(texthordimensi +
1)

        for i in range(int((sumhorizontal - 2) /
(texthordimensi + 2))):
            baristengahhorizontal_3.append(texthordimensi +
2)

        barishorizontal_1.extend(isiboundary +
baristengahhorizontal_1 + isiboundary)
        sumhorizontal_1 = len(barishorizontal_1)  # jumlah
variabel dalam list horizontal layer 1
        barishorizontal_2.extend(isiboundary +
baristengahhorizontal_2 + isiboundary)
```

```
        sumhorizontal_2 = len(barishorizontal_2)  # jumlah
variabel dalam list horizontal layer 2
        textverdimensi_2 = textverdimensi + 1
        barishorizontal_3.extend(isiboundary +
baristengahhorizontal_3 + isiboundary)
        sumhorizontal_3 = len(barishorizontal_3)  # jumlah
variabel dalam list horizontal layer 3
        textverdimensi_3 = textverdimensi + 2

        numlayer_1 = int(textnumlayer / 3)
        numlayer_2 = int(textnumlayer / 3)

        numlayer_3 = textnumlayer - (numlayer_1 +
numlayer_2)

    def clear_MoF(self):
        self.linegenerateMoF.clear()
        self.linebinding.clear()
        self.linenumlayer.clear()

#.............................................................
.....................................DEF TAB STARTUP
    def save_startup(self, item):
        name = QtGui.QFileDialog.getSaveFileName(self,
'Simpan File')
        file = open(name, 'w')

        textlineEditFormat = self.lineEditFormat.text()
        print(textlineEditFormat)
        file.writelines('FORMAT:            ' +
textlineEditFormat + '\n')

        textlinedes = self.linedesc.text()
        print(textlinedes)
        file.writelines('DESCRIPTION:       ' + textlinedes
+ '\n')

        textlinemod = self.linemodelfile.text()
        print(textlinemod)
        file.writelines('Model File:        ' + textlinemod
+ '\n')

        textlinedata = self.linedatafile.text()
        print(textlinedata)
        file.writelines('Data File:         ' +
textlinedata + '\n')

        textlinetanggal = self.linetanggal.text()
        print(textlinetanggal)
        file.writelines('Date/Time:         ' +
```

```
textlinetanggal + '\n')

        textlineiterasi = self.lineiterationtorun.text()
        print(textlineiterasi)
        file.writelines('MAX ITER:           ' +
textlineiterasi + '\n')

        textlinemisfit = self.linetargetmissfit.text()
        print(textlinemisfit)
        file.writelines('REQ TOL:            ' +
textlinemisfit + '\n')

        textlineroughness = self.lineroughness.text()
        print(textlineroughness)
        file.writelines('IRUF:               ' +
textlineroughness + '\n')

        textlinediapenalties =
self.linediagonalpenalties.text()
        print(textlinediapenalties)
        file.writelines('DEBUG LEVEL:        ' +
textlinediapenalties + '\n')

        textlinemodifyrough =
self.linemodifyroughness.text()
        print(textlinemodifyrough)
        file.writelines('ITERATION:          ' +
textlinemodifyrough + '\n')

        textlinestepcutcount = self.linestepsize.text()
        print(textlinestepcutcount)
        file.writelines('PMU:                ' +
textlinestepcutcount + '\n')

        textlinemodlimit = self.linemodellimits.text()
        file.writelines('RLAST:              ' +
textlinemodlimit + '\n')

        textlinemodvaluestep = self.linemodelvalues.text()
        file.writelines('TLAST:              ' +
textlinemodvaluestep + '\n')

        textlinedebug = self.linedebuglevel.text()
        file.writelines('IFFTOL:             ' +
textlinedebug + '\n')

        textlineiter = self.lineiteration.text()
        file.writelines('NO. PARMS:          ' +
textlineiter + '\n')
```

```
        rhoawal = ('0.2000000E+01   0.2000000E+01
0.2000000E+01   0.2000000E+01   0.2000000E+01')
        barisparam = int(textlineiter/5)

        for i in range(int(barisparam)):
            file.writelines(rhoawal)
        file.writelines('\t')


        file.close()

    def clear_startup(self):
        self.linedesc.clear()
        self.linemodelfile.clear()
        self.linedatafile.clear()
        self.linetanggal.clear()
        self.lineiterationtorun.clear()
        self.linetargetmissfit.clear()
        self.lineroughness.clear()
        self.linediagonalpenalties.clear()
        self.linemodifyroughness.clear()
        self.linestepsize.clear()
        self.linemodellimits.clear()
        self.linemodelvalues.clear()
        self.linedebuglevel.clear()
        self.lineiteration.clear()
        self.paraminput.clear()



if __name__ == '__main__':
    app = QtGui.QApplication(sys.argv)
    import sys
    main = tabparameter()
    main.show()
    sys.exit(app.exec_())
```

Lampiran II (List ekstensi Edi)

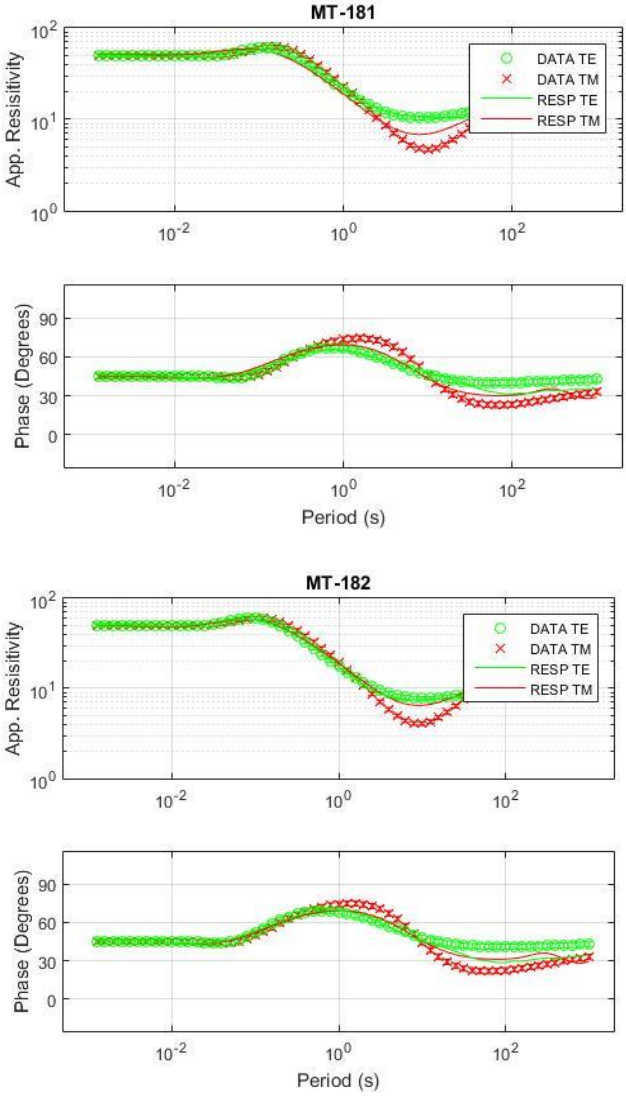
```
import matplotlib.pyplot as plt
import math
import numpy as np



def FunOpenEdi(nameEdiFiles):
    LF = nameEdiFiles

    Freq = []
    Impedanceszxyr = []
    Impedanceszxyi = []
    Impedanceszyxr = []
    Impedanceszyxi = []

    Perioda = []
    RHOxy = []
    RHOyx = []
    PHASExy = []
    PHASEyx = []

    pi = math.pi
    miu = (1.2566*(10**-7))
      # index titik

    for n in range(0, len(LF)):
        openEdi = open(LF[n], 'r')
        while True:
            readEdi = openEdi.readline()
            if readEdi == '>END':
                break
    # ----------------------------------------------
----------------------------------------------------
------------FREQUENCES
            if readEdi == '>FREQ //60\n':
                Freq.append([])
                Perioda.append([])
                for afreq in range(0, 10):
                    readEdi = openEdi.readline()
                    freqmp = readEdi.split()
```

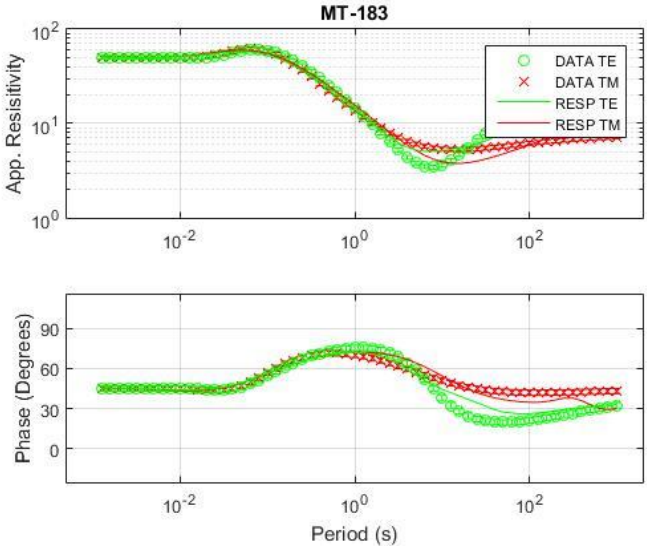
```
                    if readEdi == '>!****IMPEDANCE
ROTATION ANGLES****!':
                        break
                    for i in range(0, len(freqmp)):

Freq[n].append(float(freqmp[i]))

Perioda[n].append(1/float(freqmp[i]))

    ##-----------------------------------------------
-----------------------------------------
Impedansi XY Real
            if readEdi == '>ZXYR ROT=ZROT //60\n':
                Impedanceszxyr.append([])
                for i4 in range(0, 10):
                    readEdi = openEdi.readline()
                    zxyr = readEdi.split()
                    for i in range(0, len(zxyr)):

Impedanceszxyr[n].append(float(zxyr[i]))
    #
    ##-----------------------------------------------
-----------------------------------------
Impedansi XY Imaji
            if readEdi == '>ZXYI ROT=ZROT //60\n':
                Impedanceszxyi.append([])
                for i5 in range(0, 10):
                    readEdi = openEdi.readline()
                    zxyi = readEdi.split()
                    for i in range(0, len(zxyi)):

Impedanceszxyi[n].append(float(zxyi[i]))

            # if readEdi == '>RHOXY.ERR ROT=RHOROT
//60\n':


    ##-----------------------------------------------
-----------------------------------------
Impedansi YX Real
            if readEdi == '>ZYXR ROT=ZROT //60\n':
                Impedanceszyxr.append([])
```

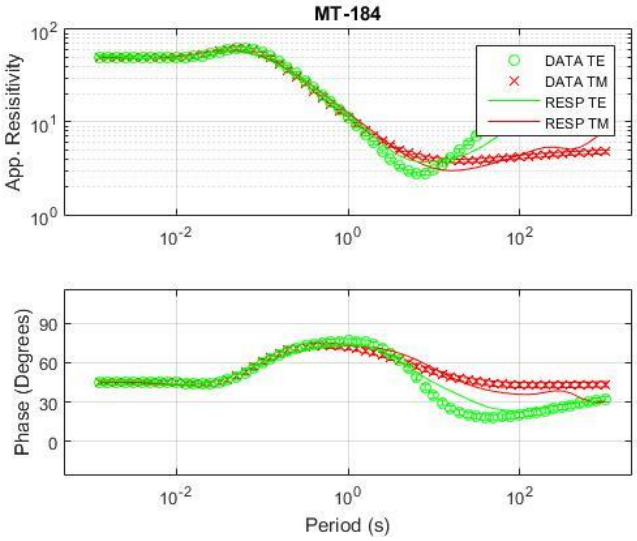
```
                for i7 in range(0, 10):
                    readEdi = openEdi.readline()
                    zyxr = readEdi.split()
                    for i in range(0, len(zyxr)):

Impedanceszyxr[n].append(float(zyxr[i]))

    ##-----------------------------------------
-----------------------------------------
Impedansi YX Imaji
            if readEdi == '>ZYXI ROT=ZROT //60\n':
                Impedanceszyxi.append([])
                for i8 in range(0, 10):
                    readEdi = openEdi.readline()
                    zyxi = readEdi.split()
                    for i in range(0, len(zyxi)):

Impedanceszyxi[n].append(float(zyxi[i]))




#.................................................
.........................................Appare
nt Resisivity
    for t in range(0, len(LF)):  # titik 0-3
        RHOxy.append([])
        RHOyx.append([])
        PHASExy.append([])
        PHASEyx.append([])
        for i in range(0, len(Perioda[t])):  #
komponen 0-60

RHOxy[t].append((0.2*float(Perioda[t][i]))*(np.abs(
complex((Impedanceszxyr[t][i]),
(Impedanceszxyi[t][i])))**2)

RHOyx[t].append((0.2*float(Perioda[t][i]))*(np.abs(
complex((Impedanceszyxr[t][i]),
(Impedanceszyxi[t][i])))**2)
```

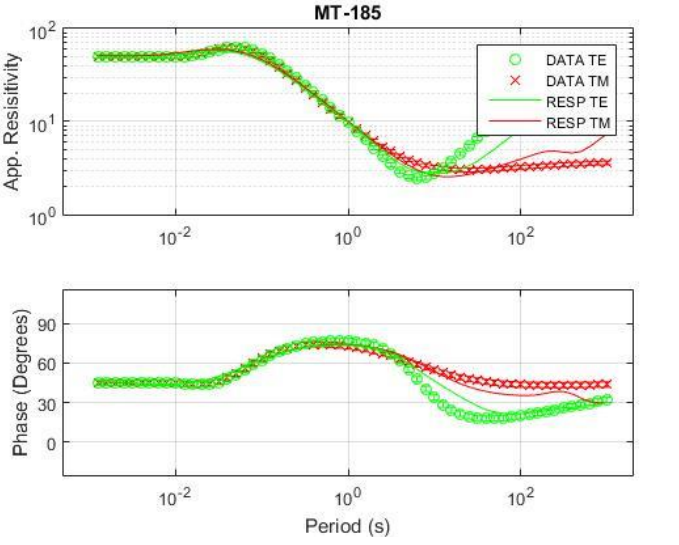
```
PHASExy[t].append(np.degrees(np.arctan(((float(Impe
danceszxyi[t][i]) /
float(Impedanceszxyr[t][i]))))))

PHASEyx[t].append(np.degrees(np.arctan(((float(Impe
danceszyxi[t][i]) /
float(Impedanceszyxr[t][i]))))))

        plt.figure(t)
        plt.subplot(211)
        plt.loglog(Perioda[t], RHOxy[t])
        plt.loglog(Perioda[t], RHOyx[t])
        plt.subplot(212)
        plt.semilogx(Perioda[t], PHASExy[t])
        plt.semilogx(Perioda[t], PHASEyx[t])
    #
    plt.show()

    return Freq, Perioda, RHOxy, RHOyx, PHASEyx,
PHASExy
```

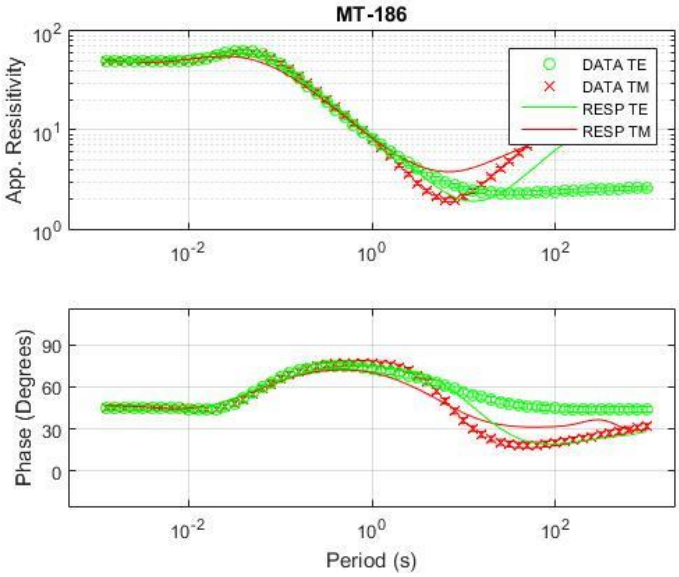
Lampiran II (Kurva Resistivitas Tiap Titid Stasiun)

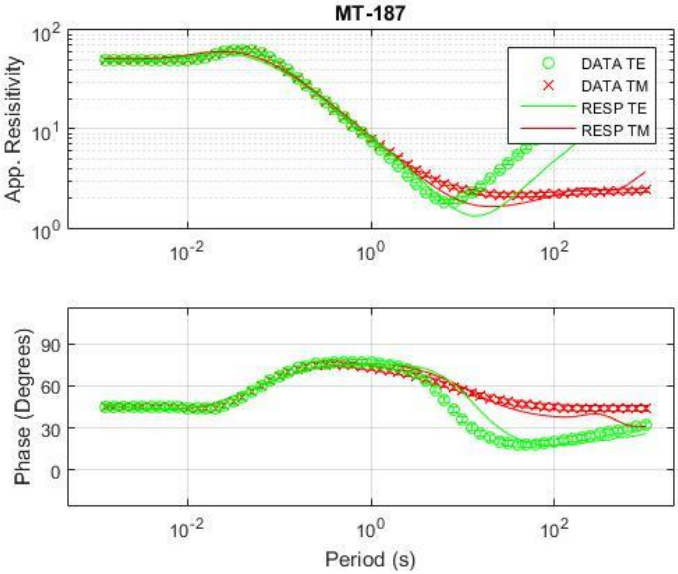


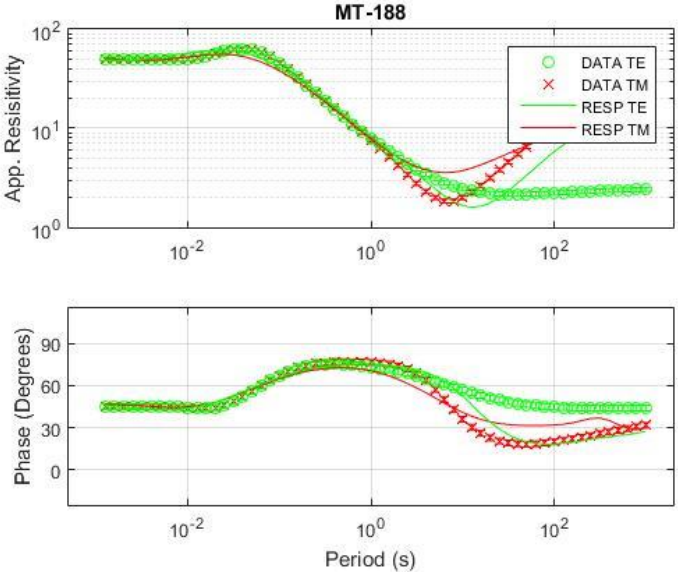

MT-183
App. Resistivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

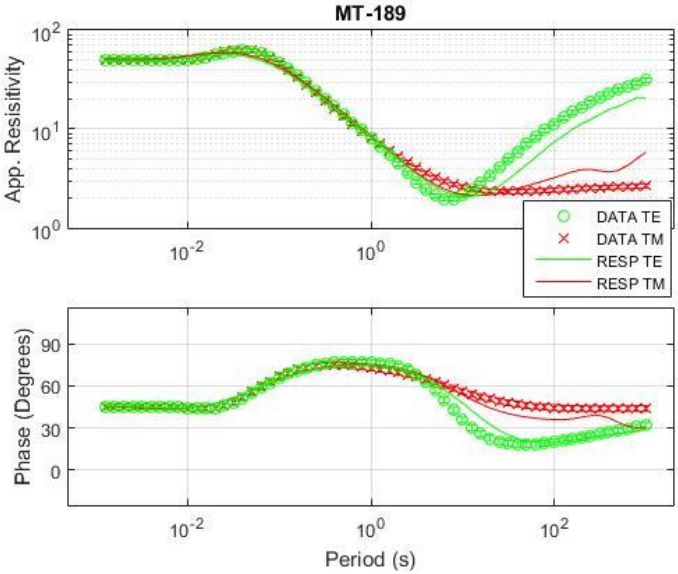

MT-184
App. Resistivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

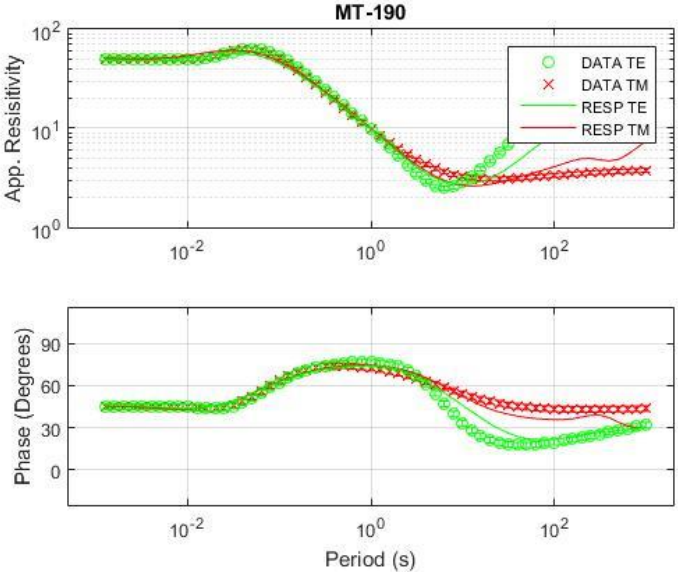

MT-185
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

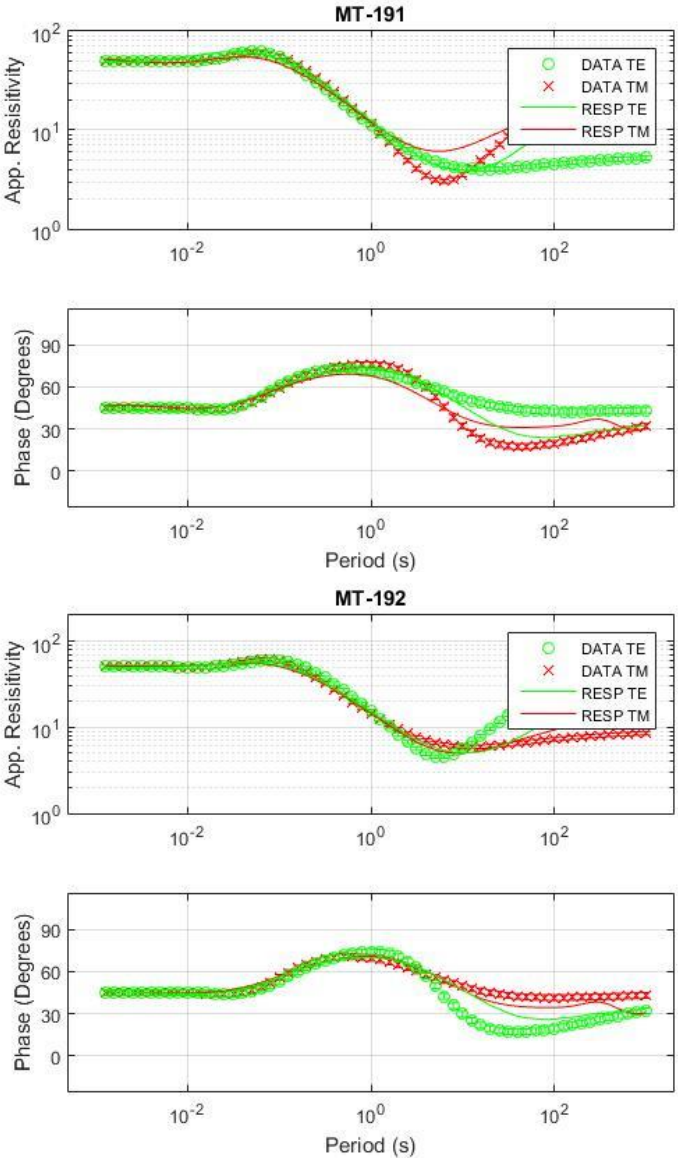

MT-186
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

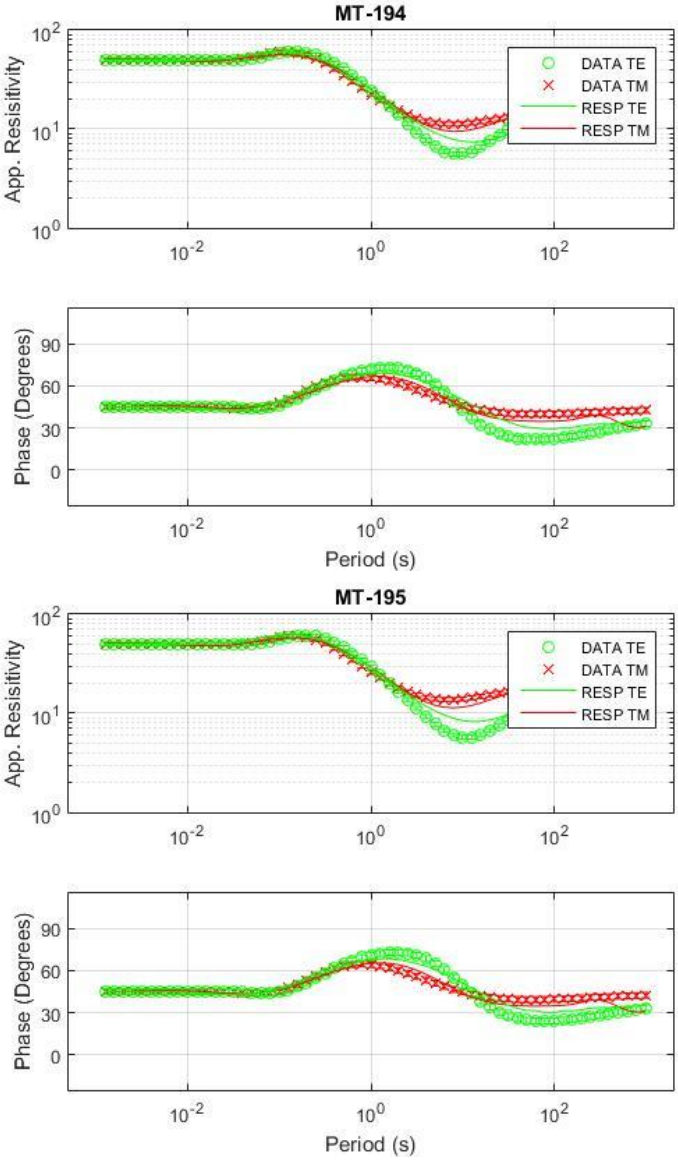

MT-187
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

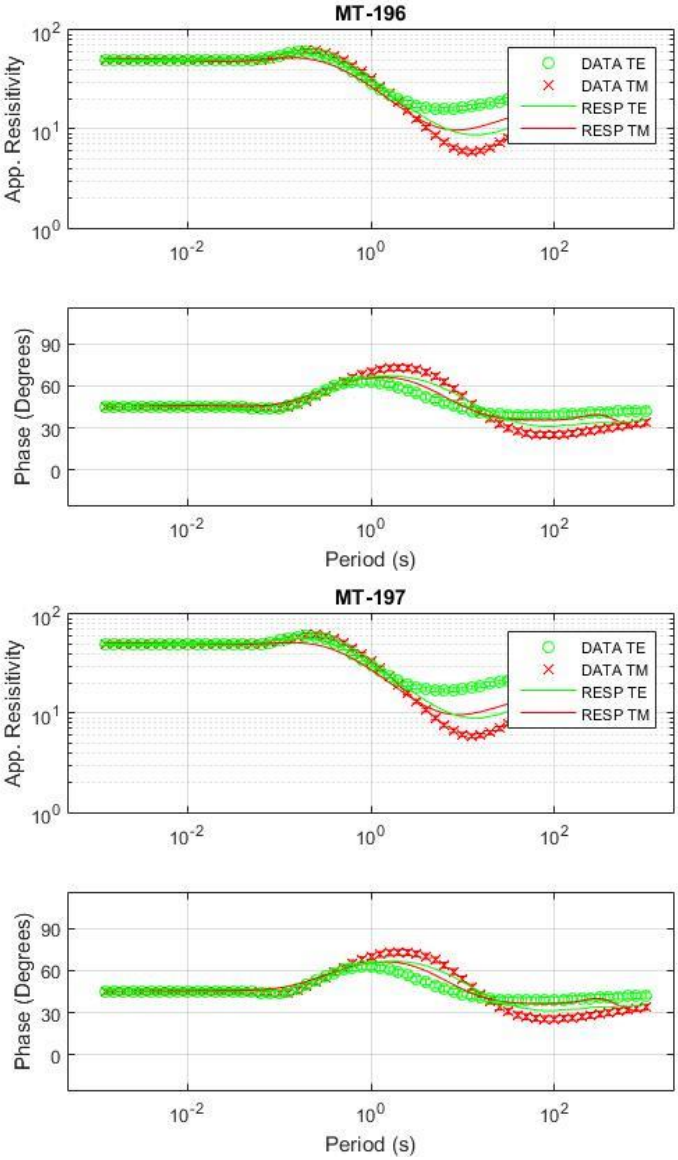

MT-188
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)

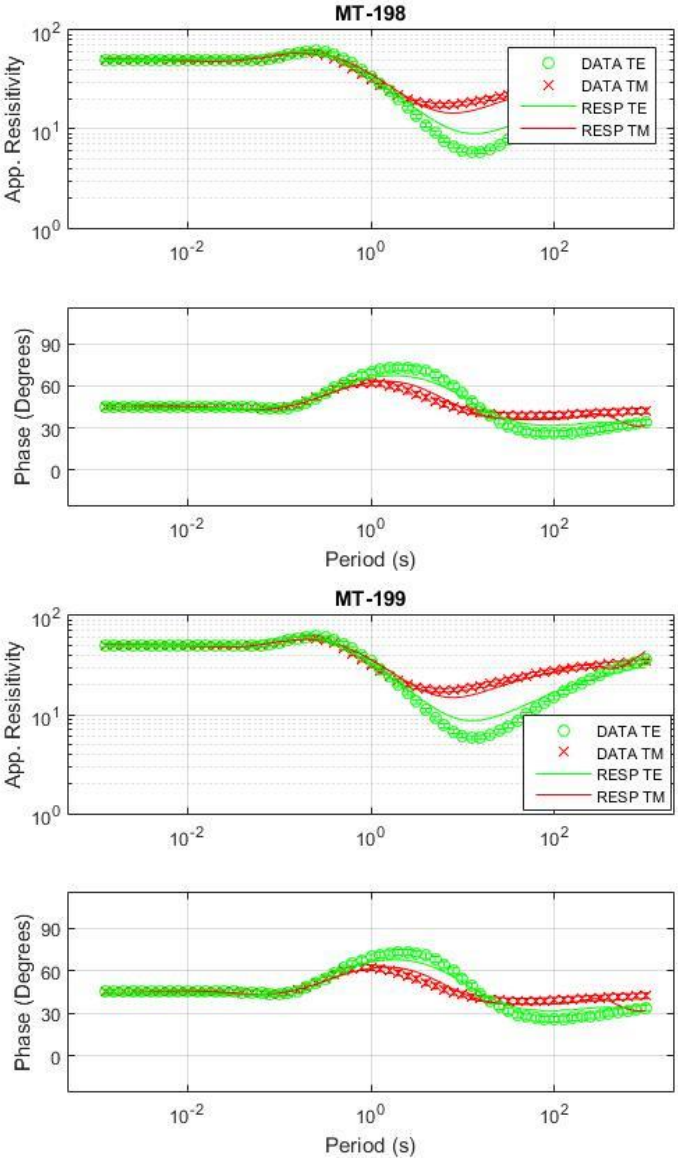

MT-189
App. Resisitivity
10^2
10^1
10^0
10^{-2}
10^0
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
90
60
30
0
10^{-2}
10^0
10^2
Period (s)

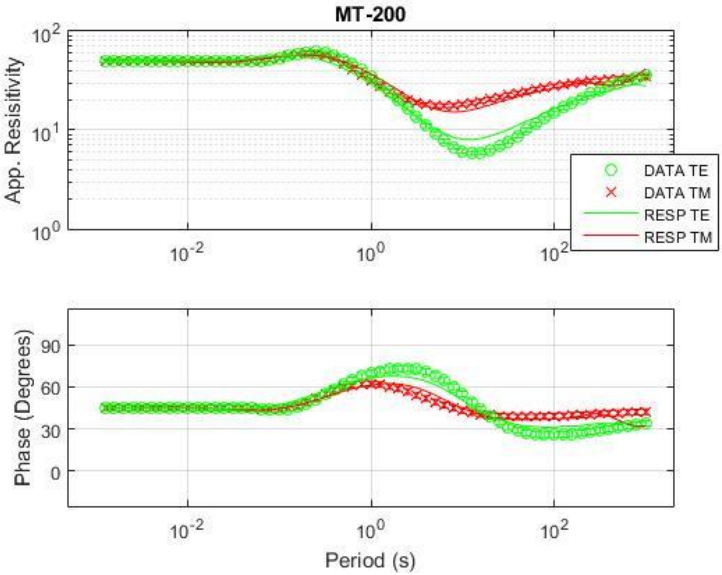

MT-190
App. Resisitivity
10^2
10^1
10^0
10^{-2}
10^0
10^2
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
90
60
30
0
10^{-2}
10^0
10^2
Period (s)


MT-191
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)
MT-192
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)


MT-194
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)
MT-195
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)


MT-196
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)
MT-197
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)


MT-198
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)
MT-199
App. Resisitivity
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
Period (s)


MT-200
App. Resisitivity
10^2
10^1
10^0
10^{-2}
10^0
10^2
DATA TE
DATA TM
RESP TE
RESP TM
Phase (Degrees)
90
60
30
0
10^{-2}
10^0
10^2
Period (s)

PROFIL PENULIS

Satrio Budiraharjo lahir di Kendari, 27 Maret 1995 dari pasangan Bapak Sujono dan Ibu Mariati. Penulis merupakan anak kedua dari 3 bersaudara. Pendidikan formal penulis dimulai di SD 2 Baruga (), kemudian melanjutkan sekolah di SMP 1 Kendari(2006-2008), melanjutkan sekolah di SMP 13 Malang (2008-2009), kemudian SMA 3 MALANG (2009-2012), terakhir Penulis melanjutkan pendidikan di Jurusan Teknik Geofisika, Institut Teknologi Sepuluh Nopember Surabaya. Selama Menjadi mahasiswa di ITS, penulis aktif dalam kegiatan organisasi, diantaranya anggota tim Sepakbola ITS perioda 2012/2013, anggota tim Catur ITS periode 2012/2013, staff Dewan Perwakilan Mahasiswa Teknik Geofisika ITS (DPM HMTG ITS) perioda 2015-2016, staf Hubungan Luar Himpunan Mahasiswa Teknik Geofisika ITS (HMTG ITS) perioda 2014/2015. Selain itu penulis juga pernah menjadi panitia, seperti sie. Sponsorship PETROLIDA SPE SC ITS 2013. Penulis juga memiliki pengalaman melaksanakan kerja praktek di BOB Bumi Siak Pusako – Pertamina dalam interpretasi data seismik dan tugas akhir di PT. Elnusa Tbk dalam analisis inversi 2D metode Occam. Jika ingin berdiskusi lebih jauh mengenai Tugas Akhir ini, dapat menghubungi email: budiraharjo.satrio@gmail.com