TEKNIK GEOFISIKA ITS

“Constrained PSO pada Penentuan Lokasi Hiposenter Gempa”

METODE INVERSI

NAMA ANGGOTA KELOMPOK:

ADE SABABURROHMAH	**03411640000010**
YOLANDA MUSTIKA B.	**03411640000012**
YOGIC WAHYU R.	**03411640000023**
RUSBA SAPUTRA	**03411640000026**

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Bagan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Secara istilah, pemodelan geofisika merupakan pencarian nilai minimum dari sebuah fungsi (fungsi misfit, fungsi obyektif) pada suatu ruang yang berdimensi banyak sesuai dengan jumlah parameter model. Solusi yang diberikan dari sebuah model dapat menggambarkan distribusi atau variasi spasial dari sifat fisika bawah permukaan yang dapat digunakan untuk memperkirakan kondisi atau struktur geologinya (Grandis, 2009).

Dalam pemodelan geofisika, terdapat dua jenis pemodelan yang sedang berkembang yaitu pemodelan inversi dengan pendekatan linier dan inversi non-linier dengan pendekatan global. Pemodelan inversi merupakan proses pencarian nilai terkecil (kecocokan) antara data hasil pengamatan (data observasi) dan data hasil perhitungan (data kalkulasi). Pemodelan inversi yang digunakan adalah sebagai optimasi, artinya untuk memenuhi solusi dari sebuah model maka fungsi misfit atau fungsi obyektif harus bernilai minimum. Hubungan yang tidak linier antara parameter model dengan data pengamatan mengakibatkan pemodelan inversi dengan pendekatan linier menjadi kurang memadai. Permasalahan ini dapat diselesaikan dengan motode inversi non-linier dengan pendekatan global, yaitu menggunakan algoritma *Particle Swarm Optimization* (PSO) (Grandis, 2009).

Algoritma *Particle Swarm Optimization* (PSO) adalah metode optimasi yang mengadopsi pola perilaku sekelompok hewan (partikel atau model) untuk mencapai target bersama. Algoritma ini diusulkan oleh Jaberipour dalam menyelesaikan sistem persamaan non-linier yang sebelumnya telah diubah menjadi bentuk permasalahan optimasi, dalam hal ini adalah minimasi. Dengan menggunakan algoritma PSO diharapkan hasil penyelesaian yang diperoleh efisien dengan rata-rata konvergensi yang tinggi (Jaberipour, 2011).

1.2. Rumusan Masalah

Dalam penulisan laporan ini dibahas beberapa permasalahan diantaranya:

1.2.1. Dimanakah lokasi (koordinat) Hiposenter yang diperoleh?
1.2.2. Pada STD berapakah Hiposenter mulai menurun?
1.2.3. Bagaimana pengaruh variasi noise dan STD dengan hasil inversi?
1.2.4. Bagaimana pengaruh *constrain* dengan hasil inversi?

1.3. Tujuan

Adapun tujuan dari penulisan laporan ini antara lain:

1.3.1. Untuk mengetahui lokasi (koordinat) Hiposenter.
1.3.2. Untuk mengetahui titik ke berapa STD Hiposenter mulai menurun.
1.3.3. Untuk mengetahui pengaruh variasi noise dan STD dengan hasil inversi.
1.3.4. Untuk mengetahui pengaruh *constrain* dengan hasil inversi.

BAB II
TINJAUAN PUSTAKA

2.1. Pemodelan ke Depan (*Forward Modelling*)

Pemodelan ke depan (*Forward Modelling*) menyatakan proses perhitungan "data" yang secara teoritis akan teramati di permukaan bumi jika diketahui harga parameter model bawah permukaan tertentu. Perhitungan data teoritis tersebut menggunakan persamaan matematik yang diturunkan dari konsep fisika yang mendasari fenomena yang ditinjau (Grandis, 2009).

Forward Modelling merupakan proses perhitungan data dari hasil teori yang akan teramati di permukaan bumi jika parameter model diketahui. Pada saat melakukan interpretasi, dicari model yang menghasilkan respon yang cocok dan fit dengan data pengamatan atau data lapangan. Sehingga diharapkan kondisi model itu bisa mewakili atau mendekati keadaan sebenarnya. Seringkali istilah forward modelling digunakan untuk proses trial and error. *Trial and error* adalah proses coba-coba atau tebakan untuk memperoleh kesesuaian antara data teoritis dengan data lapangan. Diharapkan dari proses *trial and error* ini diperoleh model yang cocok responnya dengan data (Grandis, 2009).

2.2. Pemodelan Inversi dengan Pendekatan Global

2.2.1. Pencarian Sistematik dan Acak (*Random*)

Untuk mengatasi kelemahan pendekatan linier pada pemodelan inversi non-linier maka digunakan pendekatan global yang menghindari peng¬gunaan turunan fungsi obyektif terhadap setiap parameter model. Untuk itu dilakukan eksplorasi terhadap "ruang model", yaitu ruang berdimensi banyak sesuai dengan jumlah parameter model M, secara lebih intensif. Cara "termudah" meskipun mungkin bukan cara yang efisien adalah dengan pencarian sistematik. Ruang model didefinisikan terlebih dahulu secara dengan menentukan interval harga minimum dan maksimum setiap parameter model. Ruang model didiskretisasi dengan membagi setiap interval tersebut menjadi sejumlah grid sesuai dengan ketelitian yang diinginkan. Jika jumlah grid sama untuk semua parameter model, misalnya N, maka jumlah model yang harus dihitung respons-nya untuk evaluasi fungsi atau fungsi obyektif adalah N. Jika jumlah grid berbeda untuk tiap parameter model, maka jumlah model yang harus dievaluasi adalah $N_1 \times N_2 \times ... \times N_M$ (Grandis, 2009).

2.3. Pemodelan Inversi Sebagai Optimasi

Dalam teori yang dikatakan Hendra Grandis (2009), pemodelan inversi sebagai optimasi dibagi menjadi Algoritma *Markov Chain Monte Carlo* (MCMC), Algoritma Genetika, dan *Particle Swarm Optimization* (PSO). Namun yang akan dibahas di penulisan laporan ini adalah pemodelan inversi dengan pendekatan global menggunakan *Particle Swarm Optimization* (PSO).

2.4. Dasar Teori *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) diperkenalkan oleh Dr. Eberhart dan Dr. Kennedy pada tahun 1995, merupakan algoritma optimasi yang meniru proses yang terjadi dalam kehidupan populasi burung (*flock of bird*) dan ikan (*school of fish*) dalam bertahan hidup. Sejak diperkenalkan pertama kali, algoritma PSO berkembang cukup pesat, baik dari sisi aplikasi

maupun dari sisi pengembangan metode yang digunakan pada algoritma tersebut (Haupt, R.L. & Haupt, S.E. 2004). Oleh sebab hal tersebut, mereka mengategorikan algoritma sebagai bagian dari kehidupan rekayasa/buatan *Artificial Life*. Algoritma ini juga terhubung dengan komputasi evolusioner, algoritma genetik dan pemrograman evolusionari (Jatmiko *et al*. 2010).

Dalam *Particle Swarm Optimization* (PSO), kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak disuatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik yaitu posisi dan kecepatan. Setiap partikel bergerak dalam ruang atau *space* tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut. *Particle Swarm Optimization* (PSO) adalah salah satu dari teknik komputasi *evolusioner*, yang mana populasi pada PSO didasarkan pada penelusuran algoritma dan diawali dengan suatu populasi yang random yang disebut dengan *particle*. Berbeda dengan teknik komputasi *evolusioner lainnya*, setiap particle di dalam PSO juga berhubungan dengan suatu *velocity*. Partikel-partikel tersebut bergerak melalui penelusuran ruang dengan *velocity* yang dinamis yang disesuaikan menurut perilaku historisnya. Oleh karena itu, partikel-partikel mempunyai kecenderungan untuk bergerak ke area penelusuran yang lebih baik setelah melewati proses penelusuran. *Particle Swarm Optimization* (PSO) mempunyai kesamaan dengan *genetic algorithm* yang mana dimulai dengan suatu populasi yang random dalam bentuk matriks. Namun PSO tidak memiliki operator evolusi yaitu crossover dan mutasi seperti yang ada pada *genetic algorithm*. Baris pada matriks disebut particle atau dalam *genetic algorithm* sebagai kromosom yang terdiri dari nilai suatu variable. Setiap particle berpindah dari posisinya semula ke posisi yang lebih baik dengan suatu *velocity*.

Pada algoritma PSO vektor *velocity* di *update* untuk masing-masing partikel kemudian menjumlahkan vektor *velocity* tersebut ke posisi *particle*. *Update velocity* dipengaruhi oleh kedua solusi yaitu *global best* yang berhubungan dengan biaya yang paling rendah yang pernah diperoleh dari suatu particle dan solusi *local best* yang berhubungan dengan biaya yang paling rendah pada populasi awal. Jika solusi *local best* mempunyai suatu biaya yang kurang dari biaya solusi global yang ada, maka solusi *local best* menggantikan solusi *global best*. Kesederhanaan algoritma dan performansinya yang baik, menjadikan PSO telah menarik banyak perhatian di kalangan para peneliti dan telah diaplikasikan dalam berbagai persoalan optimisasi. PSO telah populer menjadi optimisasi global dengan sebagian besar permasalahan dapat diselesaikan dengan baik di mana variabel-variabelnya adalah bilangan riil. Menurut Wati (2011), beberapa istilah umum yang biasa digunakan dalam *Particle Swarm Optimization* dapat didefinisikan sebagai berikut:

1. *Swarm* : populasi dari suatu algoritma.
2. Particle : anggota (individu) pada suatu *swarm*. Setiap *particle* merepresentasikan suatu solusi yang potensial pada permasalahan yang diselesaikan. Posisi dari suatu particle adalah ditentukan oleh representasi solusi saat itu.
3. *Pbest* (*Personal best*): posisi *Pbest* suatu *particle* yang menunjukkan posisi *particle* yang dipersiapkan untuk mendapatkan suatu solusi yang terbaik.

4. *Gbest* (*Global best*) : posisi terbaik *particle* pada *swarm* atau posisi terbaik diantara *Pbest* yang ada.
5. *Velocity* (v) : vektor yang menggerakkan proses optimasi yang menentukan arah dimana suatu *particle* diperlukan untuk berpindah (*move*) untuk memperbaiki posisinya semula atau kecepatan yang menggerakkan proses optimasi yang menentukan arah dimana *particle* diperlukan untuk berpindah dan memperbaiki posisinya semula.
6. *Inertia Weight* (θ) : disimbolkan *w*, parameter ini digunakan untuk mengontrol dampak dari adanya *velocity* yang diberikan oleh suatu *particle*.
7. *Learning Rates* (c1 dan c2) : suatu konstanta untuk menilai kemampuan *particle* (c1) dan kemampuan sosial *swarm* (c2) yang menunjukkan bobot dari *particle* terhadap memorinya.

Menurut Chen & Shih (2013) posisi dari tiap partikel dapat dianggap sebagai calon solusi (*candidate solution*) bagi suatu masalah optimisasi. Tiap-tiap partikel diberi suatu fungsi *fitness* merancang sesuai dengan menunjuk masalah yang yang bersesuaian. Ketika masing-masing partikel bergerak ke suatu posisi baru didalam ruang pencarian, itu akan mengingat sebagai *personal best* (*Pbest*). Sebagai tambahan terhadap ingatan informasi sendiri, masing-masing partikel akan juga menukar informasi dengan partikel yang lain dan mengingat *global best* (*Gbest*). Kemudian masing-masing partikel akan meninjau kembali arah dan percepatannya sesuai dengan *Pbest* dan *Gbest* untuk bergerak ke arah yang optimal dan menemukan solusi yang optimal. Dengan keuntungan dari aplikasi yang mudah dan sederhana, lebih sedikit parameter yang diperlukan, dan hasil yang baik, PSO telah diadopsi didalam banyak bidang, seperti TSP, *flowshop*, VRP, *task-resource assignment*, penjadwalan khusus dan lain-lain. Sebab itu PSO telah pula diterapkan dalam membentuk penjadwalan yang optimal untuk *university cources*. Seperti halnya dengan algoritma evolusioner yang lain, algoritma PSO adalah sebuah populasi yang didasarkan penelusuran inisialisasi partikel secara *random* dan adanya interaksi diantara partikel dalam populasi. Di dalam PSO setiap partikel bergerak melalui ruang solusi dan mempunyai kemampuan untuk mengingat posisi terbaik sebelumnya dan dapat bertahan dari generasi ke generasi. Menurut Kennedy & Eberhart (1995) Algoritma PSO dikembangkan dengan berdasarkan pada model berikut:

1. Ketika seekor burung mendekati target atau makanan (atau bisa minimum atau maximum suatu fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawanan tertentu.
2. Burung yang lain akan mengikuti arah menuju ke makanan tetapi tidak secara langsung.
3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewati pada waktu sebelumnya.

Menurut Bai (2010), keuntungan dari algoritma PSO adalah:

1. PSO berdasar pada kecerdasan (*intelligence*). Ini dapat diterapkan ke dalam kedua penggunaaan dalam bidang teknik dan riset ilmiah.
2. PSO tidak hanya overlap dan kalkulasi mutasi. Pencarian dapat dilakukan oleh kecepatan dari partikel. Selama pengembangan beberapa generasi, kebanyakan hanya

partikel yang optimis yang dapat mengirim informasi kepartikel yang lain, dan kecepatan dari pencarian adalah sangat cepat.
3. Perhitungan didalam Algoritma PSO sangat sederhana, menggunakan kemampuan optimisasi yang lebih besar dan dapat diselesaikan dengan mudah.
4. PSO memakai kode/jumlah yang riil, dan itu diputuskan langsung dengan solusi, dan jumlah dimensi tetap sama dengan solusi yang ada.

Lebih lanjut Bai (2010) menjelaskan beberapa kerugian dari Algoritma PSO antara lain:
1. Metode mudah mendapatkan optimal parsial (sebagian), yang mana menyebabkan semakin sedikit ketepatannya untuk peraturan tentang arah dan kecepatan.
2. Metode tidak bisa berkembang dari permasalahan sistem yang tidak terkoordinir, seperti solusi dalam bidang energi dan peraturan yang tidak menentu di dalam bidang energi.

Model Algortima PSO ini akan disimulasikan dalam ruang dengan dimensi tertentu dengan sejumlah iterasi sehingga di setiap iterasi, posisi partikel akan semakin mengarah ke target yang dituju (minimasi atau maksimasi fungsi). Ini dilakukan hingga maksimum iterasi dicapai atau bisa juga digunakan kriteria penghentian yang lain. Hal ini disebabkan, PSO merupakan algoritma optimasi yang mudah dipahami, cukup sederhana, dan memiliki unjuk kerja yang sudah terbukti handal. Algoritma PSO dapat digunakan pada berbagai masalah optimasi baik kontinyu maupun diskrit, linier maupun nonlinier. PSO memodelkan aktivitas pencarian solusi terbaik dalam suatu ruang solusi sebagai aktivitas terbangnya kelompok partikel dalam suatu ruang solusi tersebut. Dengan demikian, awal penelusuran pada algoritma PSO dilakukan dengan populasi yang *random* (acak) yang disebut dengan partikel dan jika suatu partikel atau seeokor burung menemukan jalan yang tepat atau pendek menuju sumber makanan, maka sisa kelompok yang lain juga akan segera mengikuti jalan tersebut meskipun lokasi mereka jauh di kelompok tersebut. Posisi partikel dalam ruang solusi tersebut merupakan kandidat solusi yang berisi variabel-variabel optimasi. Setiap posisi tersebut akan dikaitkan dengan sebuah nilai yang disebut nilai objektif atau nilai *fitness* yang dihitung berdasarkan fungsi objektif dari masalah optimasi yang akan diselesaikan.

2.4.1. Proses Algoritma PSO

Menurut Chen & Shih (2013) untuk memulai algoritma PSO, kecepatan awal (*velocity*) dan posisi awal (*position*) ditentukan secara *random*. Kemudian proses pengembangannya sebagai berikut:
1. Asumsikan bahwa ukuran kelompok atau kawanan (jumlah partikel) adalah *N*. Kecepatan dan posisi awal pada tiap partikel dalam *N* dimensi ditentukan secara *random* (acak).
2. Hitung kecepatan dari semua partikel. Semua partikel bergerak menuju titik optimal dengan suatu kecepatan. Awalnya semua kecepatan dari partikel diasumsikan sama dengan nol, set iterasi $i = 1$.
3. Nilai *fitness* setiap partikel ditaksir menurut fungsi sasaran (*objective function*) yang ditetapkan. Jika nilai *fitness* setiap partikel pada lokasi saat ini lebih baik dari *Pbest*, maka *Pbest* diatur untuk posisi saat ini.

4. Nilai *fitness* partikel dibandingkan dengan Gbest. Jika Gbest yang terbaik maka Gbest yang di*update*.
5. Persamaan (2.1) dan (2.2) ditunjukkan di bawah ini untuk memperbaharui (*update*) kecepatan (*velocity*) dan posisi (*position*) setiap partikel.

$$V_{id}^{k+1} = w \times V_{id}^{k} + c1 \times rand1 \times (P_{id} - X_{id}) + c2 \times rand2 \times (G_{id} - X_{id}) \quad (2.1)$$

$$X_{id}^{k+1} = X_{id}^{k} + V_{id}^{k+1} \quad (2.2)$$

Dimana:

Vid = komponen kecepatan individu ke *i* pada *d* dimensi
Xid = posisi individu *i* pada *d* dimensi
ω = parameter *inertia weight*
c_1 c_2 = konstanta akselerasi (*learning rate*), nilainya antara 0 sampai 1
rand1, rand2 = parameter random antara 0 sampai 1
Pid = *Pbest* (*local best*) individu *i* pada *d* dimensi
Gid = *Gbest (global best*) pada *d* dimensi

6. Cek apakah solusi yang sekarang sudah konvergen. Jika posisi semua partikel menuju ke satu nilai yang sama, maka ini disebut konvergen. Jika belum konvergen maka langkah 2 diulang dengan memperbarui iterasi $i = i + 1$, dengan cara menghitung nilai baru dari *Pbest,j* dan *Gbest*. Proses iterasi ini dilanjutkan sampai semua partikel menuju ke satu titik solusi yang sama. Biasanya akan ditentukan dengan kriteria penghentian (*stopping criteria*), misalnya jumlah selisih solusi sekarang dengan solusi sebelumnya sudah sangat kecil.
7. Menurut Engelbrecht (2006) ada 2 aspek penting dalam memilih kondisi berhenti yaitu:
   a. Kondisi berhenti tidak menyebabkan PSO *convergent premature* (memusat sebelum waktunya) dimana solusi tidak optimal yang didapat.
   b. Kondisi berhenti harus melindungi dari kondisi *oversampling* pada nilainya, jika kondisi berhenti memerlukan perhitungan yang terus-menerus maka kerumitan dari proses pencarian akan meningkat.

Beberapa kondisi berhenti yang dapat dipakai dalam *Particle Swarm Optimization* menurut Engelbrecht (2006) adalah:

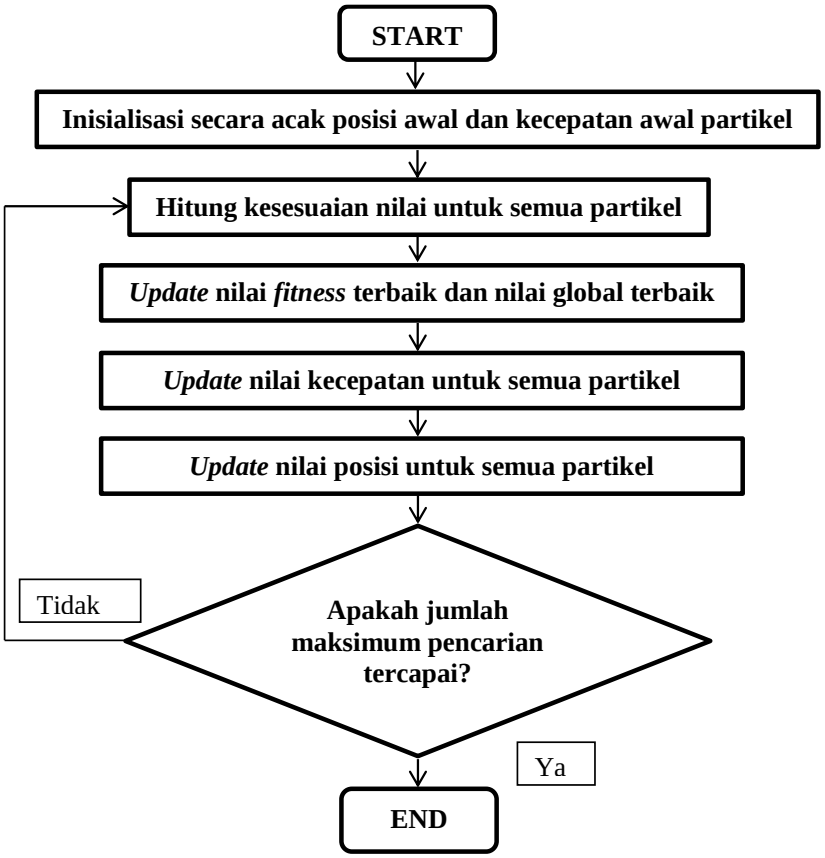
- Berhenti ketika jumlah iterasi telah mencapai jumlah iterasi maksimum yang diperbolehkan, berhenti ketika solusi yang diterima ditemukan, berhenti ketika tidak ada perkembangan setelah beberapa iterasi.

2.4.2. Menentukan Parameter Algoritma PSO

Menurut Hsieh *et al.* (2007) menentukan kombinasi parameter terbaik Algoritma PSO dengan kondisi yang berbeda yaitu:

1. Jumlah partikel (*number of particle*)
   Rangenya adalah 20 – 40, tetapi 10 partikel akan mendapatkan hasil yang lebih baik.
2. Kecepatan maksimum (*maximum velocity*)
   Kecepatan maksimum (v) diatur untuk perpindahan partikel. Jika kecepatan Vid diantara -10, 10. Kemudian kecepatan maksimum adalah pada 20.

3. *Learning factors*
   Learning factor (c1 dan c2) pada umumnya mempunyai nilai 2. Berbeda masalah juga akan berbeda nilainya dan *range*nya adalah 0 - 4.
4. Kondisi berhenti (*stop condition*)
   Termasuk jumlah iterasi maksimum dari Algoritma PSO dan syarat error mencapai minimal. Kondisi berhenti tergantung masalah yang dioptimalkan.
5. *Inertia weight*
   Shi & Eberhart (1998) menemukan Algoritma PSO dengan *Inertia weight* (*w*) dengan *range* antara 0.8 – 1.2.



Gambar 2. 1 Flowchart Algoritma PSO

2.4.3. Algoritma PSO untuk Menyelesaikan Sistem Persamaan Non-Linier

Langkah awal yang dilakukan adalah sistem persamaan non-linier diubah menjadi permasalahan optimasi. Bentuk sistem persamaannya sebagai berikut.

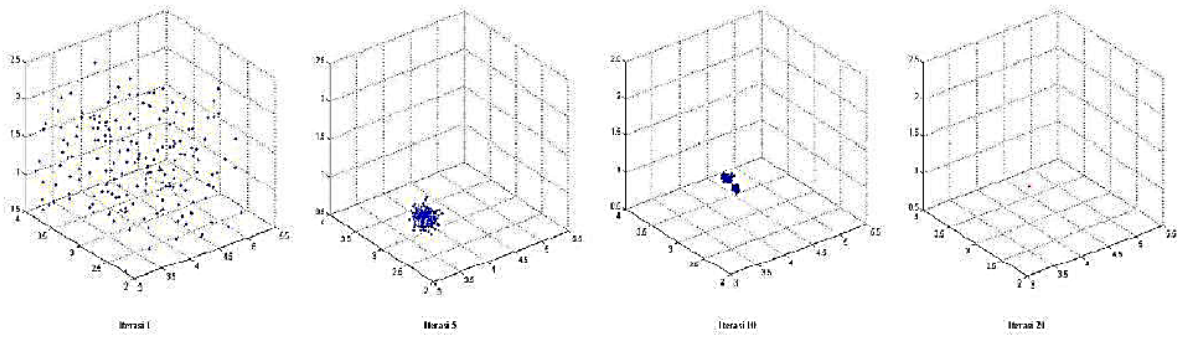
$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ F_m(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \qquad (2.1)$$

Agar dapat menggunakan metode optimasi global, sistem persamaan (2.1) diubah menjadi fungsi kuadratik sebagai berikut.

$$F(x) = \sum_{i=1}^{m} fi^2(x) \qquad (2.2)$$

Untuk menyelesaikan sistem persamaan nonlinier sama dengan meminimalkan fungsi (2.2).

PSO sebagai alat optimasi menyediakan prosedur pencarian berbasis populasi dimana masing-masing individu yang disebut partikel mengubah posisi mereka terhadap waktu. Pada sistem PSO, masing-masing partikel terbang mengitari ruang pencarian multi dimensional (*dimensional search space*) dan menyesuaikan posisinya berdasarkan pengalaman pribadinya dan pengalaman partikel di sebelahnya. Visualisasi pergerakan populasi dalam mencari posisi terbaik ditampilkan pada Gambar 2.2. Dari penjelasan diatas dapat dikatakan bahwa algoritma PSO menggabungkan metode pencarian lokal (*local search*) dengan metode pencarian global (*global search*).



Gambar 2. 2 Visualisasi perubahan posisi populasi saat mencari posisi terbaik

Tiap partikel memiliki posisi $x = (x_1, x_2, ..., x_n)$ dan kecepatan $v = (v_1, v_2, ..., v_n)$ pada ruang pencarian berdimensi N, dimana *i* menyatakan partikel ke-*i* dan N menyatakan dimensi ruang pencarian atau jumlah variabel yang belum diketahui pada sistem persamaan nonlinear. Inisialisasi algoritma PSO dimulai dengan menetapkan posisi awal partikel secara acak (solusi) dan kemudian mencari nilai optimal dengan memperbarui posisinya. Seperti yang telah dijelaskan di atas, setiap iterasi masing-masing partikel memperbarui posisinya mengikuti dua nilai terbaik, yaitu solusi terbaik yang telah didapat oleh masing-masing partikel (*pbest*) dan solusi terbaik pada populasi (*gbest*). Setelah mendapatkan dua nilai terbaik, posisi dan kecepatan partikel diperbarui dengan menggunakan persamaan berikut :

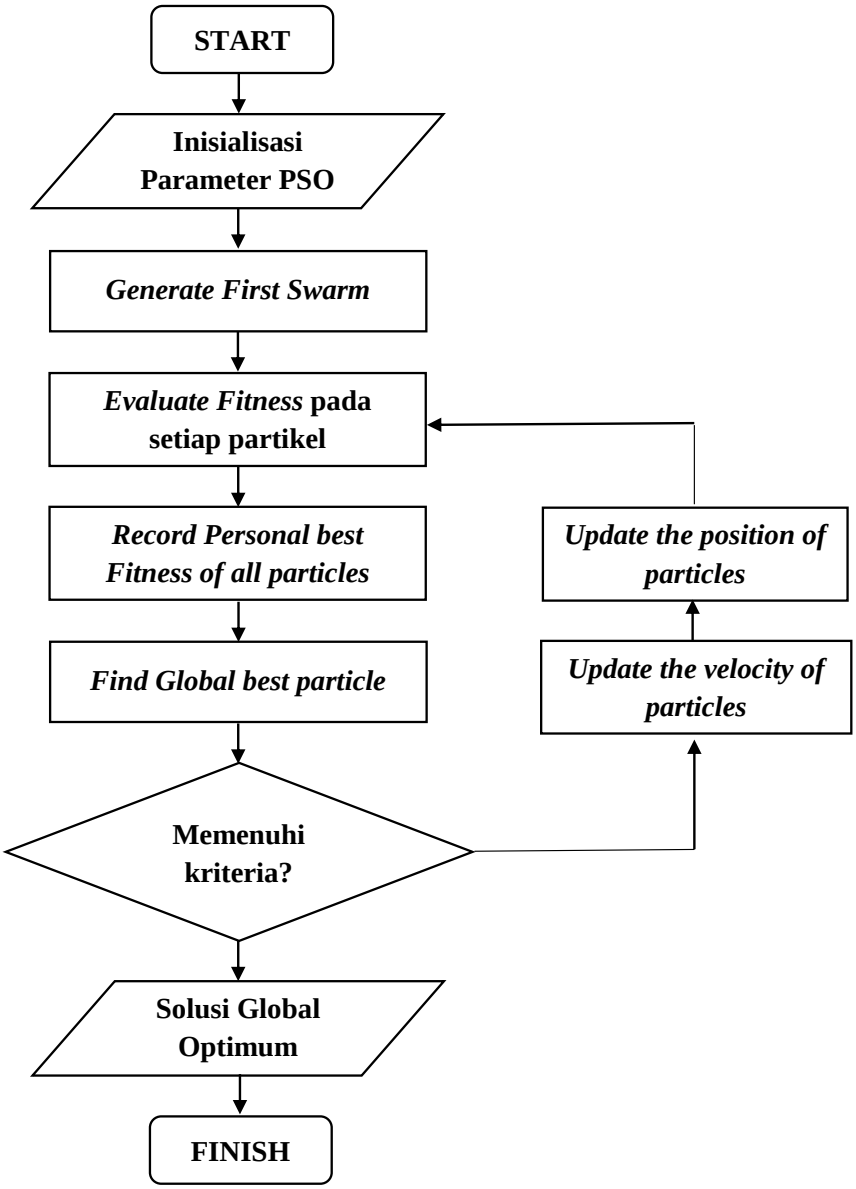
$$v_k^i = wv_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k)$$
$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad (2.3)$$

dimana v_i^k adalah kecepatan partikel ke *i* pada iterasi ke *k*, dan x_i^k adalah solusi (posisi) partikel ke *i* pada iterasi ke *k*. c_1, c_2 adalah konstanta positif, dan r_1, r_2 adalah dua variabel acak terdistribusi *uniform* antara 0 sampai 1. Pada persamaan di atas, *w* adalah bobot inersi yang menunjukkan pengaruh perubahan kecepatan dari vektor lama ke vektor yang baru.

BAB III
METODOLOGI

3.1. Metodologi Algoritma PSO Secara Umum

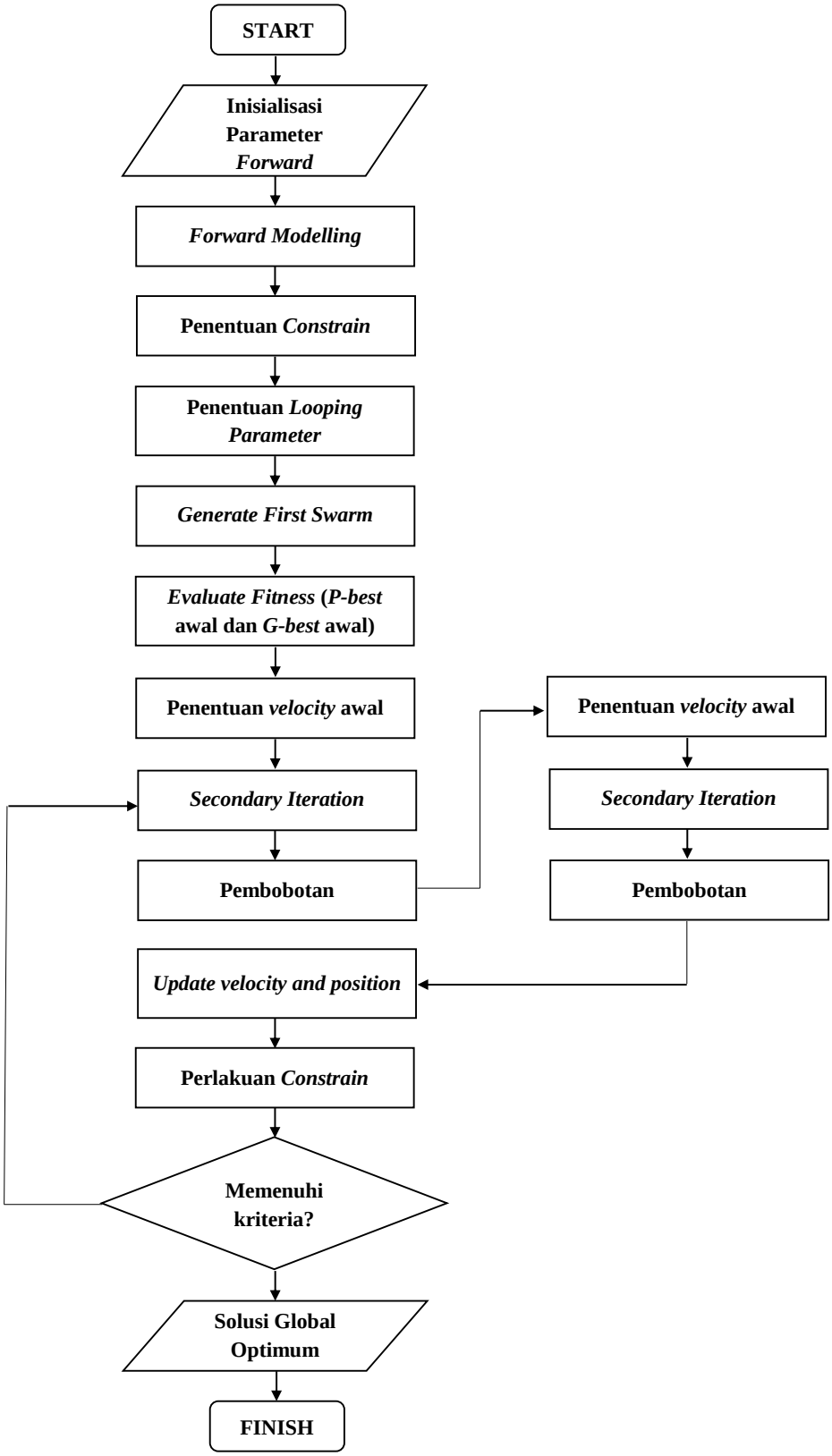
Secara umum metode inversi *Partikel Swarm Optimation* (PSO) dilakukan dengan menggunakan metodologi sebagai berikut:



Bagan 1. Metodologi Inversi Algoritma PSO Secara Umum

3.2. Metodologi Algoritma PSO Secara Khusus

Adapun metodologi yang kita gunakan dalam mencari hiposenter dengan menggunakan Inversi dengan pendekatan global dengan menggunakan PSO dalam *script* ini adalah sebagai berikut:



Bagan 2. Metodologi Inversi PSO dalam Pencarian Lokasi Hiposenter Gempa

3.3. Cara Kerja Metode PSO

Untuk melakukan pemodelan inversi PSO pada *script* ini, kita menggunakan perekaman waktu, dimana perekaman waktu akan merekam seberapa lama *script* ini berjalan dari awal pemporosesan hingga menghasilkan solusi global yang paling optimum. Adapun perekaman waktu ini bertujuan untuk mengetahui tingkat efisiensi dari proses *running* yang akan dibandingkan dengan inversi non-linear dengan pendekatan linear. Dalam *script* ini kita juga membangkitkan nilai error yang terdistribusi secara random dengan range ± 10%. *Swarm* yang digunakan terdiri dari 100 partikel untuk setiap *swarm* (*P-base*) dengan menggunakan batas *constrain* bawahnya [195 395 995] dan batas *constrain* atasnya [205 405 1005]. Selain itu, kita juga menggunakan fungsi pembobotan yang bernilai 0.4 sebagai pembobotan minimum dan 0.9 nilai pembobotan maksimumnya. Hasil nilai *P-base* yang terbaik akan dipilih sebagai nilai *G-base*. Dalam *script* ini terdapat dua jenis iterasi, yaitu iterasi primer dan iterasi sekunder. Pada proses iterasi primer dilakukan sebanyak 30 kali iterasi. Sedangkan pada iterasi sekunder terdapat adanya pembobotan ulang, *update velocity,* dan posisi, serta *handling* atau perlakuan dimana menganggap nilai *swarm* dan *G-base* berada dalam nilai *constrain.* Dalam melakukan pemodelan inversi PSO ini juga digunakan dua jenis variasi, yaitu: variasi *noise* dan variasi *constrain*. Dimana variasi noisenya adalah: 0%, 5%, dan 10%. Sedangkan variasi *constrain* yang digunakan yaitu *constrain* 5 dan 50.

BAB IV
PEMBAHASAN

4.1. Penjelasan Script

Tabel 4. 1 Penjelasan *script* yang digunakan dalam Metode PSO

Script	Keterangan

`tic`
`clear all`
`clc`
`close`
	Pemasukan timer untuk menghitung seberapa lama pemrosesan dan efektivitasnya. Pembersihan data dari pengaruh data sebelumnya.

`A = [ 300 700 1000 200 3500 1800 2000 50];  %x`
`B = [ 1200 200 450 600 100 100 1000 200 ];  %y`
`C = rand(size(A)).*1000;`
`%z`
`v_p = 10.9;`
 	
`e =0;`
`%Persen Error Maksimum`
`k =e*randn(1,1)/100;`
`x_hipo = 200;y_hipo=400;z_hipo=1000;to=0;     %Asumsi Letak Pusat Sebenarnya`
	Insialisasi parameter *forward modelling* serta penambahan noise dan letak hiposenter sintetis

`t_obs = zeros(length(A),1);`
`for i=1:length(A)`
`t_obs(i) = (to+(sqrt(((x_hipo-A(i))^2+(y_hipo-B(i))^2+(z_hipo-C(i))^2))/v_p))+...`
`(k*(to+(sqrt(((x_hipo-A(i))^2+(y_hipo-B(i))^2+(z_hipo-C(i))^2))/v_p))); %Data Sintetis`
`end`
	Penghitungan nilai t_obs dengan melakukan *forward modelling*

`LB=[195 395 995] ;     %Batas Bawah x,y,z` `UB=[205 405 1005];     %Batas Atas x,y,z`	Penentuan *constrain*

`m=3;                % Jumlah variabel dalam satu partikel`
`n=100;              % Ukuran Swarm`
`wmax=0.9;          % Konstanta Pembobotan Maksimum`
`wmin=0.4;          % Konstanta Pembobotan Minimum`
`c1=2;               % Konstanta perubahan individu/personal-cognitive`
`c2=c1;              % Konstanta perubahan global/social-cognitive`
`h = length(A);     % Dimensi vektor koordinat x stasiun`
`maxite=1000;       % NEST LOOP/SWARM CALCULATION`
`maxrun=10;         % Main LOOP`
	Penentuan parameter PSO *looping*

for run=1:maxrun%PEMBANGKITAN SWARM
for i=1:n
for j=1:m
x0(i,j)=round(LB(j)+rand()*(UB(j)-LB(j))); %Data dugaan hiposenter secara random
end
end
 	
x=x0; % initial swarm
v=zeros(size(x0)); % initial velocity
 	
for i=1:n
for c = 1:h
t_cal(c,1) =(to+(sqrt(((x(i,1)-A(c))^2+(x(i,2)-B(c))^2+(x(i,3)-C(c))^2))/v_p));
end
f0(i,1) = std(abs(t_obs-t_cal(:,1)));
%Standar Deviasi hiposenter
end
[fmin0,index0]=min(f0);
pbest=x0; % initial pbest
gbest=x0(index0,:); % initial gbest
	Proses pembangkitan dan *generate first swarm*, serta mengevaluasi *fitness* yang berupa nilai p-base awal dan g-base awal. P-base merupakan kumpulan dari partikel-partikel awal (*swarm*). Sedangkan g-base merupakan nilai p-base awal yang terbaik. Selain itu kita juga menentukan nilai *velocity* awal dan posisi awal (sama seperti p-base).

ite=1;
tolerance=1;
while ite<=maxite && tolerance>10^-12
w=wmax-(wmax-wmin)*ite/maxite; % update inertial weight
% pso velocity updates
for i=1:n
for j=1:m
v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))+c2*rand()*(gbest(1,j)-x(i,j));
end
end
% pso position update
for i=1:n
for j=1:m
x(i,j)=x(i,j)+v(i,j);
end
end
% handling boundary violations
for i=1:n
for j=1:m
if x(i,j)<LB(j)
x(i,j)=LB(j);
elseif x(i,j)>UB(j)
x(i,j)=UB(j);
end
end
end
	Pada tahap ini dilakukan proses iterasi sekunder, yaitu iterasi didalam iterasi dengan toleransi iterasi 1. Pada tahapan ini, dilakukan adanya pembobotan, *update velocity*, dan posisi, serta *handling* atau perlakuan dimana menganggap nilai *swarm* dan g-base berada dalam nilai *constrain*.

for i=1:n
for c = 1:h
t_cal_update(c,1) =(to+(sqrt(((x(i,1)-A(c))^2+(x(i,2)-B(c))^2+(x(i,3)-C(c))^2))/v_p));
end
f(i,1) = std(abs(t_obs-t_cal_update(:,1)));
%Standar Deviasi hiposenter
end
% updating pbest and fitness
for i=1:n
if f(i,1)<f0(i,1)
pbest(i,:)=x(i,:);
f0(i,1)=f(i,1);
end
end
	Kemudian dilakukan *evaluating fitness* yang berupa nilai p-base dan g-base yang telah di-*update*. Dalam tahapan ini kita menentukan nilai dari solusi global optimum, jika dalam tahap ini tidak ditemukan solusi yang optimum maka akan kembali lagi dilakukan iterasi sekunder. Hasil solusi global yang optimum berupa nilai g-base yaitu koordinat gempa yang paling mendekati dengan koordinat sebenarnya.

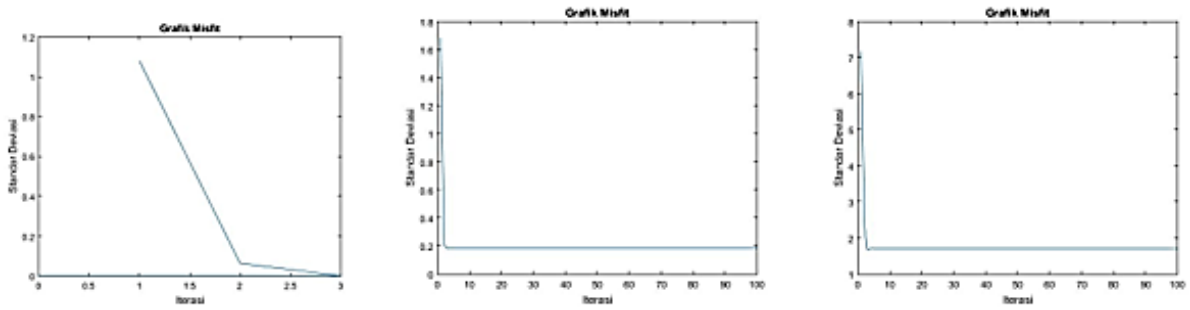
figure(1)
if run < maxrun;
plot3(gbest(1),gbest(2),(-1)*gbest(3),'r.','Markersize',15)
hold on
else
plot3(gbest(1),gbest(2),(-1)*gbest(3),'g.','Markersize',15)
hold on
end
grid on
title ('Update Lokasi Hiposenter')
legend ('Hiposenter')
xlabel ('Koordinat X(m)')
ylabel ('Koordinat Y(m)')
zlabel ('Kedalaman (m)')
 	
figure(2)
plot(ffmin(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
 	
figure(3)
plot3(gbest(1),gbest(2),-1*gbest(3),'g.','Markersize',30)
hold on
plot3 (A,B,C,'bv')
hold on
grid on
title ('LOKASI FINAL HIPOSENTER')
legend ('Hiposenter','Stasiun')
xlabel ('Koordinat X(m)')
ylabel ('Koordinat Y(m)')
zlabel ('Kedalaman (m)')
	Plotting Hiposenter yang telah di-*update*

4.2. Pembahasan

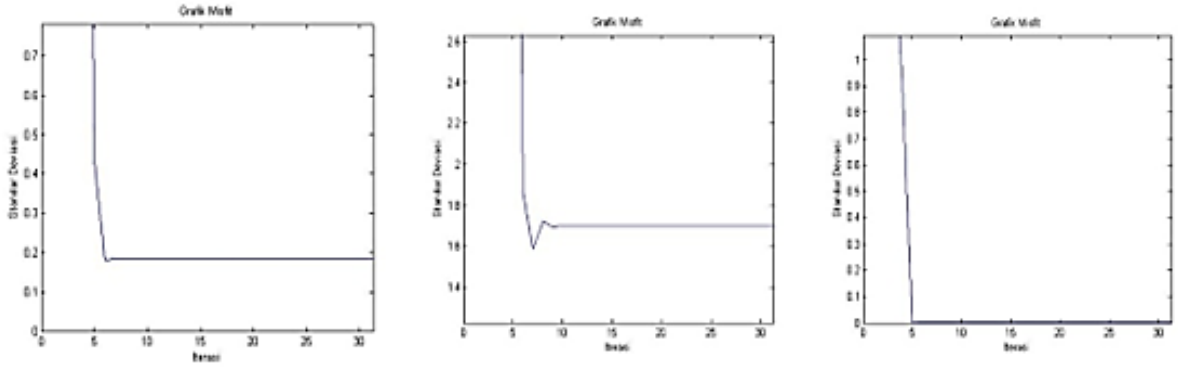
Pada inversi ini, dilakukan perbandingan antara inversi non linier dengan pendekatan linear dengan inversi pedekatan global *Particle Swarm Optimation* (PSO). Inversi dengan pendekatan linier sangat bergantung pada sifat linearitas data yang akan diinversikan. Berapa dan apa saja parameter yang terlibat didalamnya akan memengaruhi proses liniearisasi proses inversi tersebut. Hal terpenting lainnya adalah inversi ini sangat mengandalkan data " a priori" untuk mengoptimalkan proses konvergensi/*Error*. Oleh karena itu, biasanya nilai inversi semacam ini akan jatuh pada titik optimum lokal tanpa bisa melanjutkan ke titik global meskipun nilai *stopping criterion* atau *error* maksimum belum tercapai. Dengan kata lain, ada batasan maksimum nilai optimal yang dapat dihasilkan dari proses inversi pendekatan linier. Oleh karena itu dilakukanlah proses inversi dengan pendekatan global, salah satunya dengan *Particle Swarm Optimazion* (PSO). Inversi dengan pendekatan global PSO ini dilakukan dengan tidak memperdulikan seberapa banyak iterasi dan *error maksimum* yang ditentukan dengan harapan dapat mencapai solusi global yang paling optimum. Satu-satunya harapan adalah pada kesesuaian *constrain* yang dapat menggiring inversi ini menemukan global minimum. Pada eksekusi di MATLAB, digunakan variasi noise untuk melihat seberapa berpengaruhnya noise terhadap proses inversi tersebut. Nilai yang digunakan berupa distribusi *noise* maksimum sebesar 0%, 5%, dan 10%. Hasilnya pada inversi dengan pendekatan global PSO dapat diamati bahwa jika semakin besar nilai noise, maka nilai inversi yang dihasilkan akan semakin menjauhi nilai yang sebenarnya, namun nilai konvergensi yang dihasilkan masih terbilang rendah. Sedangkan pada inversi dengan pendekatan linear menghasilkan pemodelan dengan tingkat ketidaksesuaian yang lebih besar dan nilai konvergensi yang dihasilkan tidak stabil daripada PSO. Hal ini dapat terlihat dan diamati pada tabel berikut:

Tabel 4. 2 Perbandingan Hasil Inversi dengan Pendekatan Linear dan Global PSO

HASIL								
METODE INVERSI	VARIASI		Konvergensi	Time Run(s)	Koordinat Inversi			Ketidaksesuaian
	NOISE	Constraint/" A Priori"			X	Y	Z	
PSO	0	5	0.00128	31.63662	199.938	399.9303	999.9919	0.05
	5	5	2.7751	13.25	195	396.6125	995	4.46
	10	5	13.9659	7.6869	205	395	1005	5.00
	0	50	0.011	38.327	199.962	399.9145	999.7021	0.14
	5	50	3.75	31.9118	200.94	400.4133	999.7673	0.53
	10	50	3.3332	14.998	250	384	1005	23.67
Pendekatan Linier	0	[1000,300,150]	0.0025	1.1385	199.99	400	1000	0.00
	5	[1000,300,150]	0.88	1.8555	232.315	433.49	1009.164	24.99
	10	[1000,300,150]	9.62	1.3604	902.867	874.51	343.2296	611.38
	0	[198,395,1050]	0.000004	1.2565	199.9990	399.9990	1000.0000	0.00
	5	[198,395,1050]	2.4	1.0826	79.1805	348.9221	973.7400	66.05
	10	[198,395,1050]	0.6	1.4866	287.7234	440.1614	1056.6700	61.52



Gambar 4. 1 Grafik Missfit dengan Pendekatan Linier (Kiri) berupa STD Noise 5%, STD Noise 10% (Tengah), dan STD No Noise (Kanan)

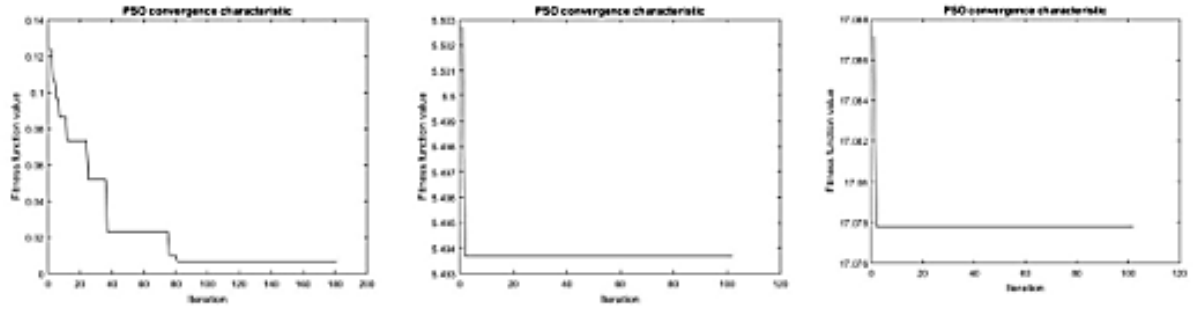


Gambar 4. 2 Grafik Missfit dengan Pendekatan Global PSO berupa STD Noise 5% (Kiri), STD Noise 10% (Tengah), dan STD No Noise (Kanan)

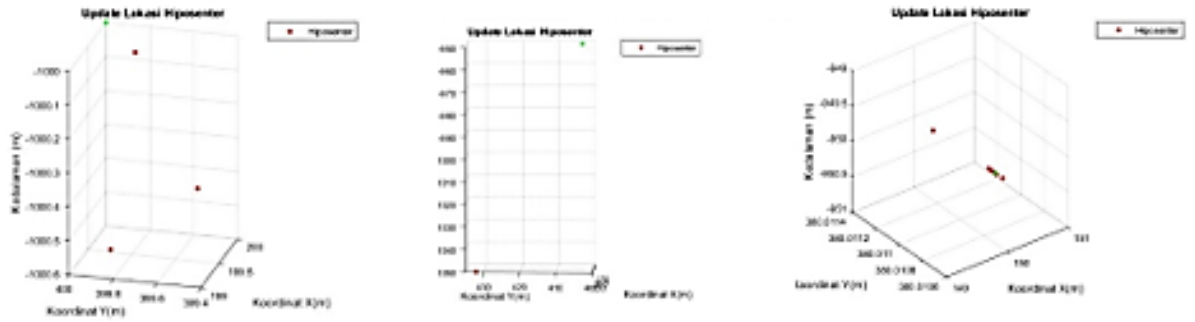
Inversi PSO kali ini divariasikan pada nilai noise yang diterapkan pada data *forward* dan *constrain* pada pembangkitan *swarm* dan *handling constrain*. Dimana *handling constrain* ini akan menormalisasi agar nilai global base tersebut tetap berada di dalam range batas bawah dan batas atas *constrain* yang ditentukan. Pada variasi noise, dipilih beberapa distribusi nilai noise maksimum antara lain 0%, 5%, dan 10%. Pada nilai noise 0%, ternyata proses konvergensi berlangsung lebih lama daripada yang lainnya. Hal ini dikarenakan evaluasi nilai, g-best, dan p-best bisa dioptimalkan lebih oleh sistem MATLAB sehingga *stopping criterion* dapat mendekati nol. Sedangkan pada nilai noise lebih tinggi seperti 5% dan 10% mengakibatkan proses konvergensi menjadi lebih singkat akibat didapatkannya nilai maksimum yang bisa di-*fitting* dengan data *forward* oleh MATLAB dengan ketidaksesuaian yang lebih besar.

Selain itu, pada inversi pendekatan global dengan PSO digunakan juga variasi *constrainst*, dibuat dengan tujuan untuk mengetahui pengaruh dari distribusi *swarm* yang dibangkitkan terhadap proses inversi PSO. Nilai variasi *constraints* yang digunakan kali ini yaitu ± 5 dan ± 50 terhadap koordinat sintetis hiposenter. Pembangkitan *swarm* menggunakan konsep "*randn*" yang menginisialisasi bilangan yang terdistribusi acak dari −1 sampai 1. Pada *constraint* ± 5, proses konvergensi lebih cepat karena diakibatkan "*search area*" lebih mendekati solusi sebenarnya ditambah pembangkitan *swarm* yang terdiri atas 100 partikel koordinat acak terbatas dengan distribusi semakin memadati *search area* yang sempit akibat *constraints* yang kecil. Oleh karena itu proses konvergensi lebih maksimal dan nilai hiposenter yang didapatkan sesuai dengan dugaan awal atau *forward*. Tentunya penerapan *constraints* ini dapat berhasil dan optimal jika batas *constrain* mengenai dugaan hiposenter mendekati aslinya.

Teknik ini mirip seperti yang dipakai pada metode inversi non linier dengan pendekatan linier. Namun perbedaannya terletak pada hasil pendekatan solusi yang paling optimal. Dimana PSO sebagai metode inversi *Global Optimization* yang menghasilkan nilai yang paling mendekati atau paling optimal dengan atau tanpa data "a priori" pada kasus non-linier atau kompleks seperti pusat gempa bumi. Hasil dari solusi global optimum dari pemodelan PSO dengan variasi *constrains* dapat dilihat dan diamati pada gambar berikut.



Gambar 4. 3 STD *Constrain* 5 Noise 0% (Kiri), Noise 5% (Tengah), dan Noise 10% (Kanan)



Gambar 4. 4 *Update* Hiposenter *Constrain* 50, dengan Noise 0% (Kiri), Noise 5% (Tengah), dan Noise 10% (Kanan)

BAB V

KESIMPULAN

Pada proses inversi kali ini didapatkan hasil dari metode PSO pada kasus hiposenter sebagai berikut:

1. Koordinat hiposenter yang didapatkan saat noise 0% adalah [199.9883, 399.9303, 999.9919].
2. Standar deviasi hiposenter menurun berundak dan mulai landai pada titik 1,5.
3. Semakin besar *noise* pada data maka standar deviasi akan naik dan hasil inversi tidak mencerminkan nilai yang mendekati.
4. Semakin kecil atau dekat *constraints*/" a priori" yang digunakan maka proses inversi akan lebih cepat dan menghasilkan koordinat yang sangat mendekati aslinya.

DAFTAR PUSTAKA

Bai, Q., 2010, *Analysis of Particle Swarm Optimization Algorithm, Computer and Information Science*, vol. 3, no. 1, pp.180–184.

C. K. Mohan. E. Ozcan. *Particle Swarm Optimization Homepage*. Available: http://www.cis.syr.edu/~mohan/pso/ (diakses pada 20 Desember 2018 pukul 10.48 WIB)

Dewi, Kadek. 2010. *Distribution Hypocenter Vulcanic Earthquake At Semeru Mountain Lumajang East Java*. Malang :Universitas Negeri Malang.

Eberhart, R.C., Shi, Y., *Comparing Inertia Weight and Constriction Factors in Particle Swarm Optimization*, Proceding of the 2000 Congress on Evolutionary Computation, Vol. 1, ,pp. 84-88 , 2000.

Grandis, H. (2009). Pengantar Pemodelan Inversi Geofisika. Bandung: Himpunan Ahli Geofisika Indonesia (HAGI).

Hajihassani. 2017. *Applications of Particle Swarm Optimization in Geotechnical Engineering: A Comprehensive Review*. Iran : Urmia University.

Haupt, R. L., & Haupt, S. E. 2004. *Particle Genetic Algorithms*. New Jersey: John Wiley & Sons.

Jaberipour Majid, Khorram Esmaile, Karimi Behrooz, "*Particle swarm algorithm for solving systems of nonlinear equations*," Elsevier Computers and Mathematics with Applications ScienceDirect (2011) 566-576.

Kennedy, and R. Eberhart. 1995. "*Particle Swarm Optimization*," IEEE Conference on Neural Networks, pp. 1942-1948, (Perth, Australia), Piscataway, NJ, IV.

Peksen,Ertan. 2011. Application of particle swarm optimization on self-potential data. Turkey: Kacaeli University.