PERBANDINGAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) DAN ALGORITMA *GLOWWORM SWARM OPTIMIZATION* (GSO) DALAM PENYELESAIAN SISTEM PERSAMAAN NON LINIER

SKRIPSI

Oleh
Ana Ulul Azmi
NIM 141810101025

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2018

PERBANDINGAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) DAN ALGORITMA *GLOWWORM SWARM OPTIMIZATION* (GSO) DALAM PENYELESAIAN SISTEM PERSAMAAN NON LINIER

SKRIPSI

Diajukan Guna Memenuhi Tugas Akhir Kuliah

Oleh

ANA ULUL AZMI

NIM 141810101025

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS JEMBER

2018

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Ibunda Sitiaisyah, Ayahanda Wahyu Kusuma, dan sekeluarga tercinta yang telah memberikan do'a, kasih sayang dan motivasi yang sangat berharga bagi putra tercintanya;
2. Guru-guruku sejak taman kanak – kanak sampai dengan perguruan tinggi;
3. Almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, SMA Negeri 1 Pesanggaran, SMP Negeri 1 Siliragung, SD Negeri 1 Siliragung dan TK Pertiwi 1 Siliragung.

MOTTO

”Barangsiapa bersungguh-sungguh, sesungguhnya kesungguhan itu adalah untuk dirinya sendiri”

(QS. Al- Ankabut[29] : 6)*)

“Allah mencintai pekerja yang apabila bekerja ia menyelesaikannya dengan baik”

(HR. Thabrani)**)

---

*) Kata Bijak Motivasi dari Al-Qur’an

**) Kata-Kata Bijak dari Hadist

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Ana Ulul Azmi

NIM : 141810101022

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Perbandingan Algoritma *Particle Swarm Optimization* dan Algoritma *Glowworm Swarm Optimization* dalam Penyelesaian Sistem Persamaan Non Linier" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, … 2018

Yang menyatakan,

Ana Ulul Azmi

NIM 141810101025

SKRIPSI

PERBANDINGAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* (PSO) DAN ALGORITMA *GLOWWORM SWARM OPTIMIZATION* (GSO) DALAM PENYELESAIAN SISTEM PERSAMAAN NON LINIER

Oleh

Ana Ulul Azmi

NIM 141810101025

Pembimbing

Dosen Pembimbing Utama : Drs. Rusli Hidayat, M.Sc

Dosen Pembimbing Anggota : M. Ziaul Arif, S.Si., M.Sc

PENGESAHAN

Skripsi berjudul "Perbandingan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dalam Penyelesaian Sistem Persamaan Non Linier" telah diuji dan disahkan pada:
hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Anggota I,
Drs. Rusli Hidayat, M.Sc NIP. 196610121993031001	M. Ziaul Arif, S.Si., M.Sc. NIP. 198501112008121002
Anggota II,	Anggota III,
Kusbudiono, S.Si., M.Si NIP. 197704302005011001	Ikhsanul Halikin, S.Pd., M.Si NIP. 198610142014041001

Mengesahkan
Dekan,

Dr. Sujito, Ph. D
NIP. 196102041987111001

RINGKASAN

Perbandingan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dalam Penyelesaian Sistem Persamaan Non Linier, Ana Ulul Azmi, 141810101025; 2018: 54 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam.

Sistem persamaan non linier adalah himpunan dari beberapa persamaan non linier. Umumnya, sistem persamaan non linier sulit diselesaikan menggunakan metode analitik. Terdapat banyak cara untuk mencari solusi sistem persamaan non linier. Solusi sistem persamaan non linier adalah nilai-nilai variabel bebas yang mengakibatkan sistem persamaan bernilai nol. Salah satu metode yang dapat digunakan untuk menyelesaikan sistem persamaan non linier yaitu metode optimasi *metaheuristic*. Algoritma *Particle Swarm Optimization* (PSO) dan algoritma Glowworm Swarm Optimization (GSO) merupakan contoh dari metode optimasi *metaheuristic*.

Penelitian ini dimulai dari menentukan beberapa sistem persamaan non linier dari beberapa artikel yang akan diteliti berupa sistem persamaan non linier dua variabel dan sistem persamaan non linier tiga variabel. Kemudian menentukan nilai fungsi yaitu dengan memilihi yang paling minimum dari hasil penjumlahan absolut sistem persamaan non linier. Kemudian menentukan nilai parameter, parameter utama dari algoritma *Partcle Swarm Optimization* dan algoritma *Glowworm Swarm Optimization* yaitu $npop$, lb(batas atas), dan ub(batas bawah) Parameter untuk algoritma *Partcle Swarm Optimization* yaitu $V0(kecepatan\ awal), c1 = c2, t\_max,$ dan t_min. Parameter untuk algoritma *Glowworm Swarm Optimization* yaitu $l0, r0, rs, gamma, beta, nt, s,$ dan rho. Setelah itu, mencari solusi sistem persamaan non linier menggunakan algoritma *Partcle Swarm Optimization* dan algoritma *Glowworm Swarm Optimization*. Kemudian menguji keakuratan dari algoritma

Partcle Swarm Optimization dan algoritma *Glowworm Swarm Optimization* dengan membandingkan solusi sistem persamaan non linier dari beberapa artikel yang dikutip yang telah diselesaikan menggunakan Metode Newton-Raphson.

Hasil penyelesaian sistem persamaan non linier menggunakan algoritma *Particle Swarm Optimization* menghasilkan solusi yang mendekati eksak. *Particle Swarm Optimization* memberikan solusi konvergen, meskipun untuk mencapai kekonvergenan cukup lama. *Running time* algoritma *Particle Swarm Optimization* dalam menyelesaikan SPNL relatif cepat. Jika nilai awal algoritma *Particle Swarm Optimization* mendekali nilai solusi maka iterasi yang dibutuhkan untuk mencapai nilai fungsi optimum lebih sedikit.

Hasil penyelesaian sistem persamaan non linier menggunakan algoritma *Glowworm Swarm Optimization* tidak menghasilkan solusi yang mendekati eksak. *Glowworm Swarm Optimization* memberikan solusi cepat konvergen dalam penyelesaiaan permasalahan sistem persamaan non linier, hal ini dikarenakan *Glowworm* akan berpindah jika *luciferin* tetangganya ada yang lebih terang, tetapi posisi baru setelah berpindah tidak menjamin solusi menjadi lebih baik. *Running time Glowworm Swarm Optimization* dalam penyelesaian SPNL relatif lambat.

Hasil penyelesaian sistem persamaan non linier menggunakan algoritma *Glowworm Swarm Optimization* lebih cepat konvergen dibandingkan algoritma *Particle Swarm Optimization*. Akan tetapi algoritma *Particle Swarm Optimization* selalu mendapat nilai fungsi yang mendekati solusi eksak, sedangkan algoritma *Glowworm Swarm Optimization* lebih cepat menghasilkan solusi yang konvergen . *Running time Glowworm Swarm Optimization* dalam penyelesaian SPNL relatif lambat jika dibandingkan dengan algoritma *Particle Swarm Optimization*. Solusi dari algoritma *Glowworm Swarm Optimization* dengan algoritma Newton-Raphson dalam penyelesaian sistem persamaan non linier hampir mendekati sama yaitu mendekati nilai solusinya. Penyelesaian sistem persamaan non linier jika dilihat dari penelitian ini lebih baik menggunakan algoritma *Particle Swarm Optimization*, karena kekonvergenannya mendapat nilai fungsi yang mendekati solusi eksak.

PRAKATA

Puji syukur kehadirat Allah SWT. Atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Perbandingan Algoritma *Particle Swarm Optimization* dan Algoritma *Glowworm Swarm Optimization* dalam Penyelesaian Sistem Persamaan Non Linier". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. bapak Drs. Rusli Hidayat, M.Sc. selaku Dosen Pembimbing Utama dan Bapak M. Ziaul Arif, S.Si., M.Sc. selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. bapak Kusbudiono, S.Si., M.Si. selaku Dosen Penguji I dan Bapak Ikhsanul Halikin, S.Pd., M.Si selaku Dosen Penguji II yang telah memberikan kritik serta sarannya terhadap penulisan skripsi ini;
3. bapak Kusbudiono, S.Si., M.Si. selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. seluruh staf pengajar jurusan Matematika Fakultas MIPA Universitas Jember yang telah memberikan ilmu serta bimbingannya sehingga penulis dapat menyelesaikan skripsi ini;
5. ibunda Sitiaisyah dan ayahanda Wahyu Kusuma yang telah memberikan do'a dan motivasi demi terselesaikannya skripsi ini;
6. sahabat-sahabat "Nekad Dolan" atas dukungan dan nasehat yang telah diberikan;
7. teman-teman angkatan 2014 (EXTREME) atas dukungan, keceriaan, dan canda-tawa yang telah diberikan;
8. semua pihak yang tidak dapat disebutkan satu per satu.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, … 2018

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Manusia saat ini didukung oleh teknologi yang modern sehingga manusia harus mengambil sebuah keputusan yang memiliki tingkat kesalahan sangat kecil. Manusia harus merencanakan secara terstruktur untuk menyelesaikan sebuah permasalahan. Permasalahan di dunia nyata dapat diselesaikan dengan menggunakan metode, prosedur, dan strategi pada ilmu matematika. Ilmu matematika dapat menyelesaikan suatu permasalahan menggunakan kajian matematika. Kajian matematika yang ada, salah satunya yaitu sistem persamaan. Sistem persamaan ada dua, yaitu sistem persamaan linier dan sistem persamaan non linier. Permasalahan yang cukup sulit untuk diselesaikan yaitu sistem persamaan non linier, karena merupakan kumpulan dari suatu kalimat matematika terbuka yang variabel berderajat tidak sama dengan satu atau mengandung nilai fungsi nonlinier, seperti log, sin, dan lain sebagainya.

Sistem persamaan non linear dapat diselesaikan dengan menggunakan metode analitik ataupun metode numerik. Sistem persamaan yang bersifat sederhana akan diselesaikan dengan metode analitik yang memiliki solusi eksak. Sistem persamaan non linier yang seringkali muncul bersifat kompleks, sehingga sulit diselesaikan menggunakan metode analitik, akan tetapi memungkinkan untuk diselesaikan dengan metode numerik. Algoritma metode numerik yaitu solusi yang sangat dibutuhkan untuk menyelesaikan sistem persamaan non linier tersebut.

Algoritma metode numerik yang digunakan untuk menyelesaikan sistem persamaan non linier cukup banyak. Utami (2013) melakukan penelitian tentang menyelesaikan sistem persamaan non linier dengan membandingkan Metode Newton-Raphson dan Metode Jacobian. Kesimpulan yang didapat pada penelitian ini Metode Newton-Raphson lebih baik untuk menyelesaikan sistem persamaan non linier daripada Metode Jacobian. Iterasi untuk menghasilkan hasil yang konvergen pada Metode Newton-Raphson yaitu 5 iterasi, sedangkan jumlah itersai pada Metode

Jacobian sebanyak 58 iterasi. Farida (2016) melakukan penelitian tentang modifikasi Metode Newton Tiga Langkah dalam permasalahan yang sama.

Metode Newton-Raphson, meskipun terbilang baik namun masih saja terdapat beberapa kekurangan. Tetapi metode ini mempunyai kelemahan, antara lain tingginya waktu yang digunakan karena pada metode ini terdapat iterasi yang di dalamnya mengandung matriks Jacobian. Kekurangan Metode Newton-Raphson yang lain yaitu metode ini harus menghitung turunan dari fungsi $f(x_n)$, sedangkan tidak semua fungsi dapat dicari turunannya dengan mudah. *Computational intelligent* adalah solusi yang dapat mengatasi hal tersebut.

Computational intelligent adalah riset penelitian yang digunakan dalam bidang teknik optimasi berdasarkan perhitungan cerdas dari suatu langkah yang terstruktur (Chu et al.,2006). Algoritma yang tergabung dalam *computational intelligent* misalnya *Genetic Algoritma* (GA), *Ant Colony Optimization* (ACO), *Artifical Bee Colony Optimization* (ABCO), *Cat Swarm Optimization* (CSO), *Cockroach Swarm Optimization Algorithm* (CSOA), *Glowworm Swarm Optimization* (GSO), *Particle Swarm Optimization* (PSO). Solusi pada penyelesaian sistem persamaan non linier yang mendekati solusi eksak yaitu menggunakan *computational intelligent* jenis *swarm intelligent. Swarm intelligent* memiliki inspirasi wilayah penelitian dari kawanan hewan atau serangga. Anwar (2016) menggunakan Metode *Zero Crossing* dan Algoritma *Virus Evolutionary Genetic* (VEGA) untuk menyelesaikan permasalahan penyelesaian persamaan non linier. Baihaki (2016) menggunakan Algoritma *Cat Swarm Optimization* (CSO) untuk menyelesaikan sistem persamaan non linier. Prastowo (2016) menggunakan Algoritma *Cockroach Swarm Optimization Algorithm* (CSOA) untuk menyelesaikan permasalahan yang sama.

Metode *Particle Swarm Optimization* (PSO) terinspirasi oleh perilaku sosial dari binatang, seperti sekumpulan burung dalam suatu gerombolan. Kelebihan dari metode *Particle Swarm Optimization* (PSO) yaitu kecepatan dalam menyelesaikan suatu permasalahan optimasi lebih cepat. Algoritma lain yang efektif dalam persoalan

optimasi yaitu *Glowworm Swarm Optimization* (GSO). Algoritma *Glowworm Swarm Optimization* (GSO) merupakan metode baru. Metode *Glowworm Swarm Optimization* (GSO) yaitu metode yang berdasarkan pada perilaku cacing bercahaya. Algoritma *Particle Swarm Optimization* (PSO) dan algoritma Glowworm Swarm Optimization (GSO) merupakan contoh dari metode optimasi *metaheuristic*.

Berdasarkan penjelasan diatas, penulis tertarik untuk melakukan penelitian tentang penerapan dari Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dalam penyelesaian Sistem Persamaan Non linier. Peneliti akan membandingkan galat dan sedikitnya hasil iterasi untuk mendapatkan hasil yang konvergen antara Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO).

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang di atas, maka rumusan masalah dari penelitian ini adalah:

a. bagaimana penyelesaian sistem persamaan non linier dengan Algoritma *Particle Swarm Optimization* (PSO)
b. bagaimana penyelesaian sistem persamaan non linier dengan Algoritma *Glowworm Swarm Optimization* (GSO)
c. bagaimana perbandingan hasil dari penerapan Algoritma *Particle Swarm Optimization* (PSO) dengan Algoritma *Glowworm Swarm Optimization* (GSO) dan perbandingan hasil dari penerapan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dengan metode Newton-Raphson dalam penyelesaian sistem persamaan non linier

1.3 Batasan Masalah

Batasan masalah dari proposal ini agar penerapan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) tidak

meluas maka yang akan diselesaikan yaitu semua sistem persamaan non linier yang memiliki dua variabel dan tiga variabel.

1.4 Tujuan Penelitian

Berdasarkan uraian pada latar belakang dan rumusan masalah di atas, maka tujuan penelitian dari penelitian ini adalah:

a. mengetahui penyelesaian sistem persamaan non linier dengan Algoritma *Particle Swarm Optimization* (PSO)
b. mengetahui penyelesaian sistem persamaan non linier dengan Algoritma *Glowworm Swarm Optimization* (GSO)
d. mengetahui hasil terbaik dari perbandingan hasil dari penerapan Algoritma *Particle Swarm Optimization* (PSO) dengan Algoritma *Glowworm Swarm Optimization* (GSO) dan perbandingan hasil dari penerapan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) dengan metode Newton-Raphson dalam penyelesaian sistem persamaan non linier

1.5 Manfaat Penelitian

Berdasarkan uraian pada latar belakang, rumusan masalah dan tujuan penelitian di atas, maka manfaat dari penelitian ini adalah:

a. menambah wawasan dan referensi tentang Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) sehingga dapat mengembangkan Algoritma lain yang lebih baik untuk menyelesaikan permasalahan sistem persamaan non linier
b. secara khusus dapat memberikan gambaran tentang penyelesaian sistem persamaan non linier dan menjadi pertimbangan dalam penelitian selanjutnya mengenai sistem persamaan non linier

BAB 2. TINJAUAN PUSTAKA

2.1 Metode Numerik

Metode numerik adalah teknik yang digunakan untuk memformulasikan persoalan matematik sehingga dapat dipecahkan dengan operasi perhitungan atau aritmatika biasa, yaitu tambah, kurang, kali dan bagi (Munir, 2003). (Marlena, 2014) menyatakan bahwa metode numerik tidak mengutamakan diperolehnya jawaban yang eksak dari persoalan yang sedang diselesaikan. Penyelesaian yang didapatkan merupakan penyelesaian pendekatan.

Terdapat beberapa metode numerik yang digunakan untuk menyelesaikan sistem persamaan non linier. Metode numerik yang sering digunakan adalah Metode Newton-Raphson. Metode Newton-Raphson adalah salah satu metode yang paling populer untuk menghitung hampiran akar-akar dari sistem persamaan non linier. Sebagai contoh $f(x)$ suatu fungsi diferensiabel pada $[a, b]$. Maka $f(x)$ memiliki kemiringan tertentu dan garis singgung tunggal pada setiap titik di dalam (a, b). Garis singgung di titik $(x_0, f(x_0))$ merupakan pendekatan grafik $f(x)$ didekat titik $(x_0, f(x_0))$. Jadi, pembuat nilai nol garis singgung tersebut merupakan hampiran pembuat nol $f(x)$ (Sahid, 2005).

2.2 Persamaan Non linier

Persamaan non linier adalah persamaan yang jika digambarkan pada bidang kartesius berbentuk garis tidak lurus. Menurut (Devi, 2011), persamaan non linier adalah persamaan yang melibatkan fungsi transenden dan fungsi non trasenden. Fungsi transenden meliputi fungsi sinus, cosinus, eksponensial, logaritma, dan lainnya, sedangkan fungsi non transenden yaitu seperti fungsi polinomial.

Persoalan mencari solusi persamaan non linier dapat di rumuskan sebagai berikut: S adalah himpunan solusi dari

$$f(x) = 0 \tag{2.1}$$

Jika untuk setiap $s \in S$ sedemikian hingga $f(s)$ samadengan nol. Hingga saat ini sudah banyak metode yang dilakukan untuk menyelesaikan persamaan non linier. Munir (2003) menyatakan bahwa metode tersebut dapat dapat dikelompokkan menjadi dua kelompok, yaitu Metode tertutup (Biseksi dan Regulsi Falsi) dan Metode terbuka (Iterasi Titik Tetap, Newton-Raphson, dan Secant).

2.3 Sistem Persamaan Non linier

Sistem persamaan non linier merupakan sistem persamaan yang mempunyai bentuk persamaan yang kompleks dimana persamaan tersebut tidak dapat dipecahkan secara analitik. Apabila suatu persamaan tidak dapat diselesaikan dengan metode analitik, maka persamaan tersebut dapat diselesaikan menggunakan metode numerik (Munir, 2003).

Sistem persamaan nonlinier adalah kumpulan dari beberapa persamaan nonlinier yang dicari penyelesaiannya. Contoh masalah penyelesaian sistem persamaan non linier 2 variabel dalam matematika diaplikasikan dalam mencari titik potong antara 2 kurva, misalnya kurva parabola ($x^2 - 2x - y - 0.5 = 0$) dan elips ($x^2 + 4y^2 - 4 = 0$). Hingga diperoleh solusi $(-0.2,1)$ dan $(1.9,0.3)$, yang memenuhi 2 kurva tersebut (Devi, 2011). Bentuk umum sistem persamaan non linier dapat ditulis sebagai berikut:

$$\begin{cases} f_1(x_1, x_2, x_3, \dots, x_n) = b_1 \\ f_2(x_1, x_2, x_3, \dots, x_n) = b_1 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ f_n(x_1, x_2, x_3, \dots, x_n) = b_n \end{cases} \tag{2.2}$$

Penyelesaian sistem persamaan (2.2) adalah himpunan nilai $\boldsymbol{x} = \{x_{11}, x_{12}, x_{13}, \dots, x_{1n}, x_{21}, x_{22}, x_{23}, \dots, x_{2n}, \dots, x_{nn}\}$ yang memenuhi seluruh persamaan.

Terdapat beberapa metode yang dapat digunakan untuk menyelesaikan sistem persamaan non linier, diantaranya yaitu Newton-Raphson, biseksi, Regulasi Falsi, PSO, dan CSO. Contoh sistem persamaan non linier yaitu:

$$\begin{cases} f_1(x_1, x_2) = 3x_1 + x_1^3 + x_2 + 1 = 0 \\ f_2(x_1, x_2) = x_1 + 2x_2 + e^{x_2} - 2 = 0 \end{cases} \quad (2.3)$$

Contoh lain dari sistem persamaan non linier, yaitu sebagai berikut (Grailoo, et.al., 2011):

$$\begin{cases} f_1(x_1, x_2) = \cos(2x_1) - \cos(2x_2) - 0.4 = 0 \\ f_2(x_1, x_2) = 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_2) - 1.2 = 0 \end{cases} \quad (2.4)$$

Penyelesaian sistem persamaan non linier terdiri dari himpunan nilai-nilai variable atau peubah yang secara simultan memenuhi semua persamaan tersebut.

2.4 *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) merupakan salah satu Algoritma metaheuristik yang didasarkan pada perilaku sekawanan burung dan ikan. Algortima *Particle Swarm Optimization* (PSO) pertama kali dikembangkan oleh James Kennedy dan Russel Eberhart pada tahun 1995 yang meniru perilaku sosial sekawanan burung atau ikan. Dalam Algortima *Particle Swarm Optimization* (PSO) disebut sebagai patikel. James Kennedy dan Russel Eberhart terinspirasi oleh sekawanan burung atau ikan karena partikel ini berperilaku kelompok. “partikel” dimaksudkan sebagai seekor burung dalam kawanan burung atau seekor ikan dalam kawanan ikan. James dan Russel menggunakan istilah partikel karena pada dasarnya banyak makhluk hidup

yang memiliki kecerdasan seperti burung dan ikan, misalnya belalang, lebah, dan lain sebagainya. *Particle* yang berarti individu sedangkan *Swarm* yang berarti populasi (Santoso, 2011).

Particle Swarm Optimization (PSO) mensimulasikan perilaku dari sekelompok burung atau ikan. Misalnya, sekelompok burung sedang terbang secara acak mencari makanan di sebuah area tetapi hanya ada satu potong makanan pada area tersebut. Semua burung tidak mengetahui dimana lokasi makanan yang sebenarnya, hanya saja mereka dapat mengetahui seberapa jauh letak makanan yang ada pada setiap pencarian. Strategi terbaik yang dapat digunakan untuk mencari makanan adalah dengan cara mengikuti burung yang lokasinya paling deka dengan makanan tersebut. Satu burung menyatakan satu solusi di dalam ruang masalah dan burung tersebut dipresentasikan sebagai "partikel" dalam Algoritma *Particle Swarm Optimization* (PSO). Algoritma *Particle Swarm Optimization* (PSO) menggunakan populasi dari sekumpulan partikel, dimana setiap partikel mewakili sebuah solusi yang mungkin untuk sebuah permasalahan optimasi. Semua partikel memiliki nilai *fitness* yang dievaluasi oleh fungsi *fitness* yang akan dioptimalkan, dan memiliki kecepatan (*velocity*) setiap partikel berpindah dengan kecepatan yang diadaptasi dari daerah

pencarian dan disimpan sebagai posisi terbaik yang pernah dicapai. Sehingga dari skenario kumpulan burung tersebut yang kemudian menggunakannya untuk memecahkan masalah optimasi (Erny, 2013).

Nilai fitness disebut p_{best}. Ketika partikel mengambil semua populasi sebagai tetangga topologinya, posisi terbaik adalah global terbaik yang disebut g_{best}. Melalui p_{best} dan g_{best}, patikel memperbarui diri untuk menghasilkan generasi berikutnya dari kawanan.

Partikel dalam Algoritma *Particle Swarm Optimization* (PSO) merupakan solusi potensial untuk masalah ini, dan kawanan terdiri dari partikel P. Setiap patikel p dapat direpresentasikan melalui vektor n-dimensi: vektor pertama didenisikan sebagai $C_p^t = (C_{p1}^t, C_{p2}^t, \dots, C_{pn}^t)$ dengan $p = (1,2, \dots, p)$ yang menunjukkan posisi

partikel p dalam ruang pencarian di iterasi t. Vektor kedua dituliskan $V_p^t = (v_{p1}^t, v_{p2}^t, \dots, v_{pn}^t)$ yang merepresentasikan kecepatan dengan partikel p bergerak. Vektor ketiga dituliskan $P_{best_p}^t = (p_{best_{p1}}^t, p_{best_{p2}}^t, \dots, p_{best_{pn}}^t)$ menunjukkan posisi terbaik dari partikel ke-p dan vektor terakhir dituliskan $G_{best_p}^t = (g_{best_{p1}}^t, g_{best_{p2}}^t, \dots, g_{best_{pn}}^t)$ yang merepresentasikan posisi terbaik secara global dalam kawanan sampai iterasi ke-t. Kawanan diperbarui oleh persamaan berikut:

$$v_{pn}^{t+1} = \theta v_{pn}^t + k_1 \times rand1(\ )\left(p_{best_{pn}}^t - x_{pn}^t\right) + k_2 \times rand2(\ )\left(g_{best_{pn}}^t - x_{pn}^t\right) \quad (2.5)$$

$$C_{pn}^{t+1} = C_{pn}^t + v_{pn}^{t+1} \quad (2.6)$$

θ merupakan nilai bobot inersia untuk mengurangi kecepatan. Biasanya nilai θ dibuat sedemikian hingga semakin meningkat iterasi yang dilalui, semakin mengecil kecepatan partikel. Nilai ini bervariasi secara linier dalam rentang 0.9 hingga 0.4. Nilai bobot inersia yang tinggi menambah porsi pencarian global (global exploration), sedangkan nilai yang rendah lebih menekankan pencarian lokal (local search). Untuk tidak terlalu menitikberatkan pada salah satu bagian dan tetap mencari area pencarian yang baru dalam ruang berdimensi tertentu, maka perlu dicari nilai bobot inersia (θ) yang secara imbang menjaga pencarian global dan lokal. Untuk mencapai itu dan mempercepat konvergensi, suatu bobot inersia yang mengecil nilainya dengan bertambahnya iterasi digunakan dengan formula:

$$\theta = \theta_{max} - Iterasi \frac{\theta_{max} - \theta_{min}}{Maksimum\ Iterasi} \quad (2.7)$$

dimana θ_{max} dan θ_{min} masing-masing adalah nilai awal dan nilai akhir bobot inersia, i_{max} adalah jumlah iterasi maksimum yang digunakan dan i adalah iterasi yang sekarang. Biasanya digunakan nilai θ_{max} =0,9 dan θ_{max} =0,4. k_1 dan k_2 adalah

parameter, $rand1(\ )$ dan $rand1(\ )$ adalah bilangan acak yang berdistribusi seragam dalam selang [0,1]. Koefisien k_1 dan k_2 adalah konstanta positif untuk mengontrol seberapa jauh sebuah partikel akan bergerak dalam iterasi tunggal. Nilai-nilai yang rendah memungkinkan partikel untuk menjelajah jauh dari daerah sasaran sebelum menarik kembali, sementara nilai-nilai yang tinggi mengakibatkan gerakan tiba-tiba terhadap atau masa lalu daerah sasaran. Biasanya adalah kedua nilai 2.0, meskipun pemberian nilai berbeda untuk k_1 dan k_2 kadang-kadang mengarah kepeningkatan kinerja.

Konstanta v_{max} digunakan untuk membatasi kecepatan dari partikel v_{pn}^t dan meningkatkan resolusi pencarian. Ketika v_{max} besar, kecepatan partikel juga besar, hal ini adalah kondusif untuk pencarian global, yang mungkin terbang melalui solusi optimal. Ketika v_{max} kecil, kecepatan partikel juga kecil, hal itu mengarah ke pencarian terbaik diwilayah tertentu, tetapi mudah untuk jatuh ke optimum lokal. Secara singkat, efisiensi pencarian tergantung pada v_{max}.

Setiap partikel bergerak dalam ruang pencarian dengan kecepatan sesuai dengan solusi terbaik sendiri dan solusi terbaik kelompok sebelumnya. Kecepatan v_{pn}^{t+1}, pada persamaan (2.7) terdiri dari tiga bagian yaitu bagian pertama adalah v_{pn}^t menunjukkan kecepatan partikel sebelumnya. Bagian kedua $k_1 \times rand1(\ )$ menunjukkan proses penyelesaian dari pengalaman individu. Bagian ketiga $k_2 \times rand1(\ )$ menunjukkan proses penyelesaian dari pengalaman orang lain, yang mewakili berbagai informasi dan kerjasama sosial antar partikel. Keseimbangan antara bagian-bagian ini menentukan kinerja Algoritma *Particle Swarm Optimization* (PSO) (Xu, 2015).

Secara garis besar, Algoritma *Particle Swarm Optimization* (PSO) dapat dijabarkan sebagai berikut:

a. Inisialisasi posisi awal (C_i) dan kecepatan awal partikel (V_i), dengan $i = 1,2, \dots, S$ dan S adalah ukuran *swarm*.
b. Evaluasi nilai fungsi tujuan untuk setiap partikel $(f(C_i))$.

c. Tentukan p_{best} awal dan g_{best} awal.

d. Update kecepatan dengan persamaan

$$V_{il} = \theta V_{il} + k_1 r_1 (p_{best} - C_{il}) + k_2 r_2 (g_{best} - C_{il}) \quad (2.8)$$

Dengan:

1) p_{best} adalah individu dengan nilai partikel terbaik dari masing-masing individu.
2) g_{best} adalah individu dengan nilai partikel terbaik dari semua individu dalam swarm.
3) θ adalah koefisien berat inersia, dengan rumus:

$$\theta = \theta_{max} - Iterasi \frac{\theta_{max} - \theta_{min}}{Maksimum\ Iterasi} \quad (2.9)$$

θ_{max} dan θ_{min} adalah nilai maksimum dan nilai minimum dari θ.

4) k_1 dan k_2 adalah koefisien akselerasi yang berupa konstanta positif.
5) r_1 dan r_2 adalah bilangan random antara 0 dan 1.

e. Update posisi individu baru dengan persamaan :

$$C'_{il} = C_{il} + V_{il} \quad (2.10)$$

f. Evaluasi kembali $f(C_i)$, jika $f(C_i) \leq f(p_{ibest})$ maka $p_{ibest} = C_i$, setelah mendapatkan p_{ibest} baru, maka didapatkan $f(p_{ibest})$ baru.

$$g_{best} = \begin{cases} p_{ibest}, f(p_{ibest}) < f(g_{best}) \\ g_{best}, f(p_{ibest}) \geq f(g_{best}) \end{cases} \quad (2.11)$$

g. Jika iterasi sudah maksimum maka Algoritma berhenti, jika tidak maka kembali ke langkah (d).

2.5 *Glowworm Swarm Optomization* (GSO)

Algoritma *Glowworm Swarm Optomization* (GSO) merupakan pengembangan dari Algoritma *Ant Colony Optimization* (ACO) dan karakter kunang-kunang atau cacing bercahaya. Setiap kunang-kunang atau agen memiliki jarak penglihatan untuk mengetahui keberadaan agen lain atau tetangga, yang disebut *local decision range*. *Local decision range* tergantung dari jumlah tetangga, ketika jumlah tetangga terlalu sedikit maka *local decision range* membesar untuk menemukan lebih banyak tetangga, sebaliknya *local decision range* akan mengecil apabila jumlah tetangga terlalu banyak. Nilai *luciferin* setiap agen dikaitkan dengan nilai fungsi objektif, sedangkan parameter nilai fungsi objektif dikaitkan dengan posisi dari setiap agen. Setiap agen selalu mengubah arah gerak sesuai dengan posisi tetangga yang dipilih dan hanya satu tetangga yang dipilih oleh setiap agen, yaitu tetangga yang memiliki nilai *luciferin* yang tertinggi diantara seluruh tetangga dari agen tersebut. Pada akhirnya, sebagian besar agen akan berkumpul di beberapa lokasi (Krishnanand *et al*., 2008).

Pengkodean adalah kunci untuk menyelesaikan masalah pada *Glowworm Swarm Optomization* (GSO) untuk masalah optimasi. Prosesnya sebagai berikut : $\boldsymbol{x}_i^t$ adalah lokasi dari *Glowworm* i pada iterasi t dimana $\boldsymbol{x}_i^t = [x_{i1}^t, x_{i2}^t, \dots, x_{id}^t]$ (d adalah banyaknya dimensi) dan $\boldsymbol{y}_i^t$ adalah solusi *Glowworm* dengan $\boldsymbol{y}_i^t = [y_{i1}^t, y_{i2}^t, \dots, y_{id}^t]$ yang elemennya tergantung pada solusi dari masalah (Gong *et al*., 2011).

Langkah-langkah Algoritma *Glowworm Swarm Optomization* (GSO) adalah sebagai berikut:

a. Inisialisasi parameter-parameter *Glowworm Swarm Optomization* (GSO) antara lain: banyaknya *Glowworm* (n), nilai awal *luciferin* ($l_0 > 0$), nilai awal *adaptive decision-range* ($r_d > 0$), jarak antar *Glowworm* ($r_s > 0$), tingkat *update* nilai

luciferin ($\gamma > 0$), tingkat *update dynamic decision-range* ($\beta > 0$), batas ketetanggaan ($n_t > 0$), *step-size* ($s > 0$), tingkat disiplin nilai *luciferin* ($0 < \rho < 1$), iterasi (t), nilai maksimum iterasi. Bangkitkan *Glowworm* sebanyak n secara random.

b. Hitung nilai *fitness*.

$$fitness_i = \begin{cases} f(\boldsymbol{y}_i), & \text{untuk kasus maksimasi} \\ \frac{1}{f(\boldsymbol{y}_i)}, & \text{untuk kasus minimasi} \end{cases} \quad (2.12)$$

c. Perbarui nilai *luciferin*, nilai *local decision-range*, dan lokasi *Glowworm* sebagai berikut:

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma(fitness_i) \quad (2.13)$$

$$N_i(t) = \{j : \|X_j - X_i\| < r_d^i(t); l_i(t) < l_j(t)\} \quad (2.14)$$

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \quad (2.15)$$

Setiap *Glowworm* i memilih satu tetangga j dengan maksimum kemungkinan $p_{ij}(t)$ dan berpindah ke tetangga j

$$X_i(t+1) = X_i(t) + s\left(\frac{X_j(t) - X_i(t)}{|X_j(t) - X_i(t)|}\right) \quad (2.16)$$

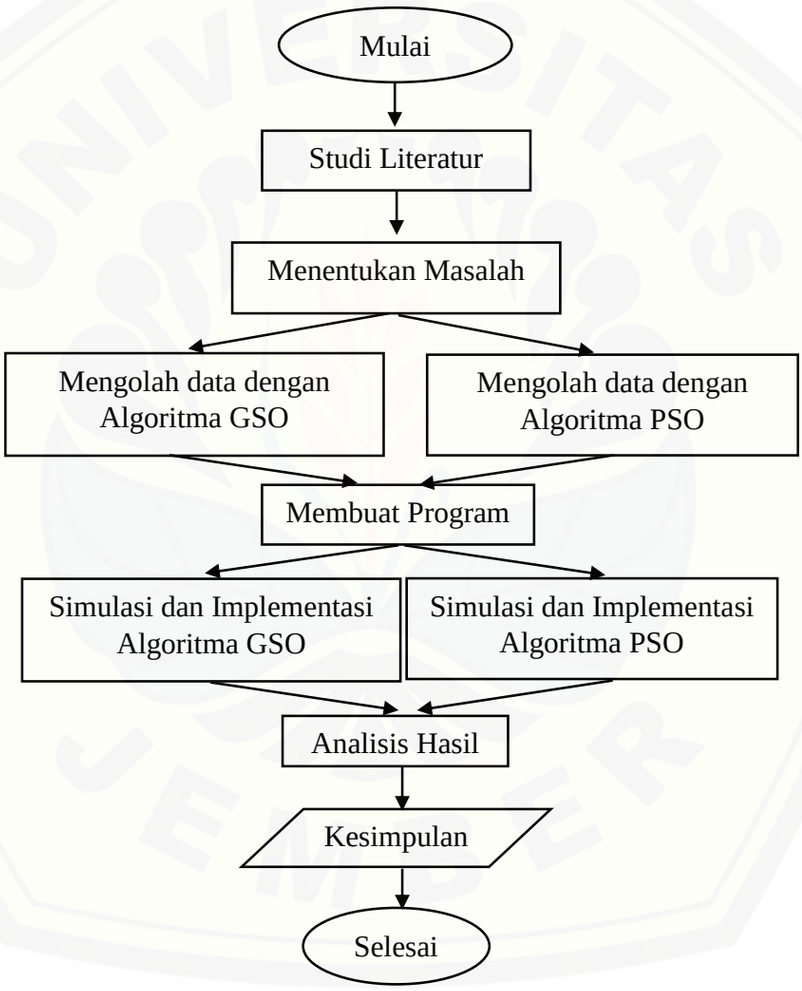
$$r_d^i(t+1) = min\left\{r_s, max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\right\} \quad (2.17)$$

d. Jika nilai maksimum iterasi terpenuhi maka iterasi dihentikan, jika belum maka kembali ke langkah b.

BAB 3. METODE PENELITIAN

Penelitian ini membandingkan Algoritma *Particle Swarm Optimization* (PSO) dan Algoritma *Glowworm Swarm Optimization* (GSO) pada beberapa sistem persamaan non linier dari beberapa sumber referensi yang dirujuk.

Langkah-langkah penelitian yang akan dilakukan untuk menyelesaikan sistem persamaan non linier digambarkan dalam skema pada gambar 3.1



Gambar 3.1 Skema Metode Penelitian

Penjelasan skema langkah penelitian pada Gambar 3.1 untuk memperoleh hasil yang diinginkan sebagai berikut:

a. Studi literatur

Studi literatur yang dilakukan pada penelitian ini adalah mempelajari mengenai Algoritma *Particle Swarm Optomization* (PSO), Algoritma *Glowworm Swarm Optomization* (GSO), dan sistem persamaan non linier.

b. Menentukan masalah

Masalah yang akan diteliti dalam penelitian kali ini yaitu tentang semua sistem persamaan non linier yang memiliki dua variabel dan tiga variabel.

c. Menerapkan Algoritma *Particle Swarm Optomization* (PSO) dan Algoritma *Glowworm Swarm Optomization* (GSO) untuk menyelesaikan permasalahan sistem persamaan non linier.

Peneliti akan menentukan *input* berupa suatu sistem persamaan yang akan dicari penyelesaiannya, kriteria pemberhentian, dan parameter yang dipakai yaitu:

1) Menyetarakan fungsi SPNL

Untuk menyelesaikan sistem persamaan berbasis pendekatan/hampiran baru, sistem (p) setara dengan:

$$(p) = \begin{cases} f_1(x_1, x_2, x_3, \dots, x_n) = 0 \\ f_2(x_1, x_2, x_3, \dots, x_n) = 0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ f_n(x_1, x_2, x_3, \dots, x_n) = 0 \end{cases} \Leftrightarrow (p') = \begin{cases} abs\left(f_1(x_1, x_2, x_3, \dots, x_n)\right) + \\ abs\left(f_2(x_1, x_2, x_3, \dots, x_n)\right) + \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ +abs\left(f_n(x_1, x_2, x_3, \dots, x_n)\right) \end{cases} = 0$$

Absolut $f(x)$ dicari seminimalis mungkin mendekati nol. Untuk inisialisasi, beberapa titik awal dihasilkan berdasarkan masalah domain dari definisi. Setiap solusi adalah vector yang panjangnya sama dengan jumlah variable untuk sistem persamaan yang dipertimbangkan.

2) Kriteria pemberhentian dan parameter mengacu pada Algoritma *Particle Swarm Optomization* (PSO) dengan langkah-langkah pada subab 2.4 dan menerapkan Algoritma *Glowworm Swarm Optomization* (GSO) dengan langkah-langkah pada subab 2.5 dengan memperhatikan bentuk masalah dan solusi pada subab 2.2 dan dibandingkan dengan Metode Newton-Raphson.

Sedangkan *output* berupa nilai *fitness* terbaik, solusi, dan grafik kekonvergenan.

d. Membuat program

Program yang dipakai untuk penelitian ini yaitu Matlab R2015b, dengan menuliskan *script* ke dalam program tersebut.

e. Simulasi dan implementasi Algoritma *Particle Swarm Optomization* (PSO) dan Algoritma *Glowworm Swarm Optomization* (GSO)

Penyelesaian suatu sistem persamaan non linier dengan membandingkan program Algoritma *Particle Swarm Optomization* (PSO) dan Algoritma *Glowworm Swarm Optomization* (GSO) dengan metode Newton-Raphson yang telah dibuat, sehingga diperoleh suatu solusi penyelesaian yang dianggap terbaik.

f. Analisis hasil

Peneliti akan menganalisis dari hasil keluaran program. Peneliti juga akan membandingkan hasil dari Algoritma *Particle Swarm Optomization* (PSO), Algoritma *Glowworm Swarm Optomization* (GSO) dan metode Newton-Raphson dengan menjawab rumusan masalah yang ditulis oleh peneliti pada subab 1.2.

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada Bab 4, didapatkan kesimpulan sebagai berikut:

a. Penyelesaian 6 sistem persamaan non liner dengan ini menggunakan algoritma *Particle Swarm Optimization* menghasilkan nilai fungsi yang mendekati solusi eksak. *Particle Swarm Optimization* memberikan solusi konvergen yang mendekati solusi eksak, meskipun untuk mencapai kekonvergenan cukup lama. *Running time* algoritma *Particle Swarm Optimization* dalam penyelesaian sistem persamaan non linier relatif cepat. Jika nilai awal algoritma *Particle Swarm Optimization* mendekati nilai solusi maka iterasi yang dibutuhkan untuk mencapai nilai fungsi yang mendekati solusi eksak.

b. Penyelesaian 6 sistem persamaan non liner dengan ini menggunakan algoritma *Glowworm Swarm Optimization* tidak menghasilkan nilai fungsi yang mendekati solusi eksak. *Glowworm Swarm Optimization* memberikan solusi yang cepat konvergen untuk penyelesaian permasalahan sistem persamaan non linier, hal ini dikarenakan *Glowworm* akan berpindah jika *luciferin* tetangganya ada yang lebih terang, tetapi posisi baru setelah berpindah tidak menjamin solusi menjadi lebih baik. *Running time Glowworm Swarm Optimization* dalam penyelesaian SPNL relatif lambat.

c. Berdasarkan penyelesaian 6 SPNL, algoritma *Glowworm Swarm Optimization* lebih cepat konvergen dibandingkan algoritma *Particle Swarm Optimization*. Akan tetapi algoritma *Particle Swarm Optimization* selalu mendapat nilai fungsi yang mendekati solusi eksak daripada algoritma *Glowworm Swarm Optimization*. *Running time Glowworm Swarm Optimization* dalam penyelesaian SPNL relatif lambat jika dibandingkan dengan algoritma *Particle Swarm Optimization*. Solusi dari algoritma *Glowworm Swarm Optimization* dengan algoritma Newton-Raphson dalam penyelesaian sistem persamaan non linier

hampir mendekati sama yaitu mendekati nilai solusinya. Penyelesaian sistem persamaan non linier jika dilihat dari penelitian ini lebih baik menggunakan algoritma *Particle Swarm Optimization*, karena kekonvergenannya mendapat nilai fungsi yang mendekati solusi eksak.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, disarankan kepada peneliti selanjutnya untuk menerapkan algoritma *Particle Swarm Optimization* dan membandingkan dengan algoritma yang lain dalam penyelesaian sistem persamaan non-linier, misalnya *An Improved Cockroach Swarm Optimization*, *Stochastic Constriction Cockroach Swarm Optimization*, *Modified Cockroach Swarm Optimization*, *Binary Cockroach Swarm Optimization*, dan lainnya.

DAFTAR PUSTAKA

Anwar, Zainul. 2016. Penerapan Gabungan Metode Zero Crossing dan Virus Evolutionary Genetic Algorithm (VEGA) pada Penyelesaian Persamaan Non linier. *Skripsi*. Jember: Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Baihaki, N.A. 2016. Penerapan Algoritma Cat Swarm Optimization pada Penyelesaian Sistem Persamaan Non linier. *Skripsi*. Jember: Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Chapra, S. C. & Canale, R. P. 1988. *Metode Numerik*. Jakarta: Erlangga.

Chu, S. C., Tsai, P. W., & Pan, J. S. 2006. "Computational Intelligence Based on the Behavior of Cats". *International Journal of innovative Computing, Information and Control.* 3 (1): 163-173

Devi, F. M. 2011. Penyelesaian Sistem Persamaan Non linier dengan Metode Jaringan Syaraf Tiruan Hopfield. *Skripsi*. Jakarta: Universitas Islam Negeri Syarif Hidayatullah.

Erny, 2013. Optimasi Pola Penyusunan Barang Dalam Peti Kemas Menggunakan Algoritma Particle Swarm Optimization. *Skripsi*. Makasar : UNHAS

Farida, Y. 2016. Modifikasi Metode Newton Tiga Langkah dalam Penyelesaian Sistem Persamaan Non linier. *Skripsi*. Jember: Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Gong, Q., Zhou, Y, & Yang, Y. 2011. Artificial Glowworm Swarm Optimization Algoritma For Solving 0-1 Knapsack Problem. *Journal of Advanced Materials Research.* 143-144: 166-171.

Graiioo, Mahdieh. 2011. Solving Sistem of Nonlinear Equations with an Improved Particle Swarm Optimization. *International Journal of Conference on Computer Science and Information Technology*. 2011: 578-582.

Krishnanand KN, Ghose D. 2008. Theorical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations. *Journal of Robotics and Autonomous Sistem*. 56 (7): 549-569.

Marlena, L. 2014. Perbandingan Solusi Sistem Persamaan Non linier Metode Titik Tetap Dan Metode Gauss-Seidel. *Skripsi.* Riau : Universitas Islam Negeri Sultan Syarif Kasim

Munir, R. 2003. *Metode Numerik*. Jakarta: Erlangga.

Nasiha, K. 2008. Penyelesaian Sistem Persamaan Tak Linier dengan Metode Newton-Raphson. *Skripsi.* Malang: Universitas Islam Negeri Malang.

Ozel, M. 2010. A New Decomposition Method for Solving System of Nonlinear Equations. *Journal of Mathematical and Computational Applications*. 15 (1): 89-95.

Prastowo, F.K. 2016. Penerapan Algoritma Cockroach Swarm Optimization Algorithm pada Penyelesaian Sistem Persamaan Non linier. *Skripsi.* Jember: Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Sahid. 2005. *Pengantar Komputasi Numerik dengan MATLAB*. Yogyakarta: ANDI.

Santoso, B. 2011. *Particle Swarm Optimization*. Surabaya : ITS

Singh, S. 2013. A System of Nonlinear Equations with Singular Jacobian. *International Journal of Innovative Research in Science, Engineering and Technology*. 2 (7): 2650-2653.

Utami, N. N. R., Widana, I. N., dan Asih, N. M. 2013. Perbandingan Solusi Sistem Persamaan Non linier Menggunakan Metode Newton-Raphson dan Metode Jacobian. *E-Jurnal Matematika*. 2 (2): 11-17.

Xu,S.H, Liu,J.P, Zhang,F.H, Wang,L., dan Sun,L.J. 2015. *A Combination of Genetic Algorithm and Particle Swarm Optimization for Vehicle Routing Problem With Time Windows*. Swithzerland : Licensee MDPI, B|asel, Switzerland.

LAMPIRAN

```
function varargout = GUI_SPNL(varargin)
% GUI_SPNL MATLAB code for GUI_SPNL.fig
%      GUI_SPNL, by itself, creates a new GUI_SPNL or raises the
existing
%      singleton*.
%
%      H = GUI_SPNL returns the handle to a new GUI_SPNL or the
handle to
%      the existing singleton*.
%
%      GUI_SPNL('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_SPNL.M with the given input
arguments.
%
%      GUI_SPNL('Property','Value',...) creates a new GUI_SPNL or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_SPNL_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_SPNL_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_SPNL

% Last Modified by GUIDE v2.5 18-Jan-2018 22:29:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_SPNL_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_SPNL_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI_SPNL is made visible.
function GUI_SPNL_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_SPNL (see VARARGIN)

% Choose default command line output for GUI_SPNL
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
movegui(gcf,'center');
set(handles.listbox1,'string','   ','value',1,'UserData',[]);
axes(handles.axes1);
cla(handles.axes1,'reset');
xlim([0 500]);ylim([-1 5]);
% UIWAIT makes GUI_SPNL wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_SPNL_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function var1_Callback(hObject, eventdata, handles)
% hObject    handle to var1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of var1 as text
%        str2double(get(hObject,'String')) returns contents of var1
as a double
```

```
% --- Executes during object creation, after setting all properties.
function var1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to var1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function var2_Callback(hObject, eventdata, handles)
% hObject    handle to var2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of var2 as text
%        str2double(get(hObject,'String')) returns contents of var2
as a double


% --- Executes during object creation, after setting all properties.
function var2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to var2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function func1_Callback(hObject, eventdata, handles)
% hObject    handle to func1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of func1 as text
%        str2double(get(hObject,'String')) returns contents of func1
as a double


% --- Executes during object creation, after setting all properties.
function func1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to func1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function func2_Callback(hObject, eventdata, handles)
% hObject    handle to func2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of func2 as text
%        str2double(get(hObject,'String')) returns contents of func2
as a double


% --- Executes during object creation, after setting all properties.
function func2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to func2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editpop_Callback(hObject, eventdata, handles)
% hObject    handle to editpop (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editpop as text
%        str2double(get(hObject,'String')) returns contents of
editpop as a double


% --- Executes during object creation, after setting all properties.
function editpop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editpop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editlb_Callback(hObject, eventdata, handles)
% hObject    handle to editlb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editlb as text
%        str2double(get(hObject,'String')) returns contents of
editlb as a double


% --- Executes during object creation, after setting all properties.
function editlb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editlb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editub_Callback(hObject, eventdata, handles)
% hObject    handle to editub (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editub as text
%        str2double(get(hObject,'String')) returns contents of
editub as a double


% --- Executes during object creation, after setting all properties.
function editub_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editub (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edititer_Callback(hObject, eventdata, handles)
% hObject    handle to edititer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edititer as text
%        str2double(get(hObject,'String')) returns contents of
edititer as a double


% --- Executes during object creation, after setting all properties.
function edititer_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edititer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function editv0_Callback(hObject, eventdata, handles)
% hObject    handle to editv0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editv0 as text
%        str2double(get(hObject,'String')) returns contents of
editv0 as a double


% --- Executes during object creation, after setting all properties.
function editv0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editv0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editc1_Callback(hObject, eventdata, handles)
% hObject    handle to editc1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editc1 as text
%        str2double(get(hObject,'String')) returns contents of
editc1 as a double


% --- Executes during object creation, after setting all properties.
function editc1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editc1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editc2_Callback(hObject, eventdata, handles)
% hObject    handle to editc2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editc2 as text
%        str2double(get(hObject,'String')) returns contents of
editc2 as a double


% --- Executes during object creation, after setting all properties.
function editc2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editc2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edittmax_Callback(hObject, eventdata, handles)
% hObject    handle to edittmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edittmax as text
%        str2double(get(hObject,'String')) returns contents of
edittmax as a double


% --- Executes during object creation, after setting all properties.
function edittmax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edittmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edittmin_Callback(hObject, eventdata, handles)
% hObject    handle to edittmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edittmin as text
%        str2double(get(hObject,'String')) returns contents of
edittmin as a double


% --- Executes during object creation, after setting all properties.
function edittmin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edittmin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editl0_Callback(hObject, eventdata, handles)
% hObject    handle to editl0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editl0 as text
%        str2double(get(hObject,'String')) returns contents of
editl0 as a double


% --- Executes during object creation, after setting all properties.
function editl0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editl0 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editr0_Callback(hObject, eventdata, handles)
% hObject    handle to editr0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editr0 as text
%        str2double(get(hObject,'String')) returns contents of
editr0 as a double


% --- Executes during object creation, after setting all properties.
function editr0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editr0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editrs_Callback(hObject, eventdata, handles)
% hObject    handle to editrs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editrs as text
%        str2double(get(hObject,'String')) returns contents of
editrs as a double
```

```
% --- Executes during object creation, after setting all properties.
function editrs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editrs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editgamma_Callback(hObject, eventdata, handles)
% hObject    handle to editgamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editgamma as text
%        str2double(get(hObject,'String')) returns contents of
editgamma as a double


% --- Executes during object creation, after setting all properties.
function editgamma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editgamma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editbeta_Callback(hObject, eventdata, handles)
% hObject    handle to editbeta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editbeta as text
```

```
%        str2double(get(hObject,'String')) returns contents of
editbeta as a double


% --- Executes during object creation, after setting all properties.
function editbeta_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editbeta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editnt_Callback(hObject, eventdata, handles)
% hObject    handle to editnt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editnt as text
%        str2double(get(hObject,'String')) returns contents of
editnt as a double



% --- Executes during object creation, after setting all properties.
function editnt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editnt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edits_Callback(hObject, eventdata, handles)
% hObject    handle to edits (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edits as text
%        str2double(get(hObject,'String')) returns contents of edits
as a double


% --- Executes during object creation, after setting all properties.
function edits_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edits (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editrho_Callback(hObject, eventdata, handles)
% hObject    handle to editrho (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editrho as text
%        str2double(get(hObject,'String')) returns contents of
editrho as a double


% --- Executes during object creation, after setting all properties.
function editrho_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editrho (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in listbox1.
```

```
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox1


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
format long;

%Parameter
pop=str2num(get(handles.editpop,'string'));
lb=str2num(get(handles.editlb,'string'));
ub=str2num(get(handles.editub,'string'));
max_iter=str2num(get(handles.edititer,'string'));
%PSO
V0=str2num(get(handles.editv0,'string'));
c1=str2num(get(handles.editc1,'string'));
c2=str2num(get(handles.editc2,'string'));
thetamin=str2num(get(handles.edittmin,'string'));
thetamax=str2num(get(handles.edittmax,'string'));
%GSO
l0=str2num(get(handles.editl0,'string'));
r0=str2num(get(handles.editr0,'string'));
rs=str2num(get(handles.editrs,'string'));
gamma=str2num(get(handles.editgamma,'string'));
```

```
beta=str2num(get(handles.editbeta,'string'));
nt=str2num(get(handles.editnt,'string'));
s=str2num(get(handles.edits,'string'));
rho=str2num(get(handles.editrho,'string'));

waktu1=0;
waktu2=0;
guidata(hObject, handles);
axes(handles.axes1);

num_var=get(handles.popupmenu1,'value');
%SPNL
if num_var==2
    var1=get(handles.var1,'string');
    var2=get(handles.var2,'string');
    f1=inline(get(handles.func1,'string'),var1,var2);
    f2=inline(get(handles.func2,'string'),var1,var2);
elseif num_var==3
    var1=get(handles.var1,'string');
    var2=get(handles.var2,'string');
    var3=get(handles.var3,'string');
    f1=inline(get(handles.func1,'string'),var1,var2,var3);
    f2=inline(get(handles.func2,'string'),var1,var2,var3);
    f3=inline(get(handles.func3,'string'),var1,var2,var3);
end
if num_var==1
    errordlg('Banyak variabel belum dipilih');
else
%pembangkitan sol awal
for i=1:pop
    Sol(i,:)=rand(1,num_var)*(ub-lb)+lb;
    if num_var==2
        Fit(i)=abs(f1(Sol(i,1),Sol(i,2)))+...
            abs(f2(Sol(i,1),Sol(i,2)));
    else
        Fit(i)=abs(f1(Sol(i,1),Sol(i,2),Sol(i,3)))+...
            abs(f2(Sol(i,1),Sol(i,2),Sol(i,3)))+...
            abs(f3(Sol(i,1),Sol(i,2),Sol(i,3)));
    end
end

%Algoritma PSO
SolPSO=Sol;
FitPSO=Fit;
V=ones(pop,num_var)*V0;
%pbest & gbest
bs=find(FitPSO==min(FitPSO));
Pbest=SolPSO;
Pbest_fit=FitPSO;
Gbest=SolPSO(bs(1),:);
Gbest_fit=min(FitPSO);
```

```
omega=thetamax;

%Algoritma GSO
l=ones(pop,1)*l0;
r=ones(pop,1)*r0;
SolGSO=Sol;
FitGSO=Fit;
FitnGSO=1./(1+Fit);
SolGSOaksen=Sol;
FitGSOaksen=Fit;
Fitnessaksen=FitnGSO;
bestsol=find(FitGSO==min(FitGSO));
solusi=SolGSO(bestsol(1),:);

best_fitPSO(1)=Gbest_fit;
best_fitGSO(1)=min(FitGSO);
tkonvergenPSO=0;
tkonvergenGSO=0;

if num_var==2
    hasil=sprintf('%50s %100s','Solusi PSO','Solusi GSO');
    hasil={char(hasil);sprintf('%5s %15s %25s %35s %40s %25s
%30s','t',var1,var2,'fitness',var1,var2,'fitness')};
else
    hasil=sprintf('%75s %150s','Solusi PSO','Solusi GSO');
    hasil={char(hasil);sprintf('%5s %15s %25s %25s %35s %40s %25s
%25s %30s','t',var1,var2,var3,'fitness',var1,var2,var3,'fitness')};
end
for t=1:max_iter
    %PSO
    if Gbest_fit~=0
        tic;
        for i=1:pop
            V(i,:)=omega*V(i,:)+c1*rand(1,num_var).*(Pbest(i,:)-
SolPSO(i,:))+c2*rand(1,num_var).*(Gbest-SolPSO(i,:));
            SolPSO(i,:)=SolPSO(i,:)+V(i,:);
            if num_var==2
                FitPSO_n(i)=abs(f1(SolPSO(i,1),SolPSO(i,2)))+...
                    abs(f2(SolPSO(i,1),SolPSO(i,2)));
            else

FitPSO_n(i)=abs(f1(SolPSO(i,1),SolPSO(i,2),SolPSO(i,3)))+...
                    abs(f2(SolPSO(i,1),SolPSO(i,2),SolPSO(i,3)))+...
                    abs(f3(SolPSO(i,1),SolPSO(i,2),SolPSO(i,3)));
            end
            %update pbest
            if FitPSO_n(i)<Pbest_fit(i)
                Pbest(i,:)=SolPSO(i,:);
                Pbest_fit(i)=FitPSO_n(i);
            end
```

```
        end
        %update gbest
        if min(Pbest_fit)<Gbest_fit
            bs=find(Pbest_fit==min(Pbest_fit));
            Gbest=Pbest(bs(1),:);
            Gbest_fit=min(Pbest_fit);
        end
        %update omega
        omega=omega-(thetamax-thetamin)/max_iter;
        best_fitPSO(t+1)=Gbest_fit;
        if best_fitPSO(t+1)~=best_fitPSO(t)
            tkonvergenPSO=t;
        end
        waktu1=waktu1+toc;
    end

    %GSO
    if min(best_fitGSO)~=0
        tic;
        %persamaan 2.2
        for i=1:pop
%           l(i)=(1-rho)*l0+gamma*FitnGSO(i);
            l(i)=(1-rho)*l(i)+gamma*FitnGSO(i);
        end
        SolGSOaksen=SolGSO;
        FitGSOaksen=FitGSO;
        Fitnessaksen=FitnGSO;
        for i=1:pop
            Ni=[];%himpunan tetangga lebih baik
            pi=[];%peluang perpindahan
            vNi(i)=0;
            %persamaan 2.3
            for j=1:pop
                if sqrt(sum((SolGSO(j,:)-SolGSO(i,:)).^2))<r(i) &&
l(i)<l(j)
%                if abs(l(j)-l(i))<r(i) && l(i)<l(j)
                    Ni(length(Ni)+1)=j;
                end
            end
            %persamaan 2.4
            if ~isempty(Ni)
                vNi(i)=1;
                jml=sum(l(Ni)-l(i));
                for j=1:length(Ni)
                    pi(j)=(l(Ni(j))-l(i))/jml;
                end
                %persamaan 2.5
                jmax=find(pi==max(pi));
%
SolGSOaksen(i,:)=SolGSO(i,:)+s*((SolGSO(jmax(1),:)-
SolGSO(i,:)));%/sqrt(sum((SolGSO(jmax(1),:)-SolGSO(i,:)).^2)));
```

```
                if sqrt(sum((SolGSO(Ni(jmax(1)),:)-
SolGSO(i,:)).^2))==0
                    SolGSOaksen(i,:)=SolGSO(i,:);
                else

SolGSOaksen(i,:)=SolGSO(i,:)+s*((SolGSO(Ni(jmax(1)),:)-
SolGSO(i,:))/sqrt(sum((SolGSO(Ni(jmax(1)),:)-SolGSO(i,:)).^2)));
                end
                if num_var==2

FitGSOaksen(i)=abs(f1(SolGSOaksen(i,1),SolGSOaksen(i,2)))+...
                        abs(f2(SolGSOaksen(i,1),SolGSOaksen(i,2)));
                else

FitGSOaksen(i)=abs(f1(SolGSOaksen(i,1),SolGSOaksen(i,2),SolGSOaksen(
i,3)))+...

abs(f2(SolGSOaksen(i,1),SolGSOaksen(i,2),SolGSOaksen(i,3)))+...

abs(f3(SolGSOaksen(i,1),SolGSOaksen(i,2),SolGSOaksen(i,3)));
                end
                Fitnessaksen(i)=1/(1+FitGSOaksen(i));%hitung fitness
                %persamaan 2.6
                r(i)=min(rs,max(0,r(i)+beta*(nt-length(Ni))));
            end
        end
        SolGSO=SolGSOaksen;
        FitGSO=FitGSOaksen;
        FitnGSO=Fitnessaksen;
        bestsol=find(FitGSO==min(FitGSO));
        solusi=SolGSO(bestsol(1),:);
        best_fitGSO(t+1)=min(FitGSO);
        if best_fitGSO(t+1)~=best_fitGSO(t)
            tkonvergenGSO=t;
        end
        waktu2=waktu2+toc;
    end

    %tampilkan solusi
    tic;
    if num_var==2
        if Gbest_fit==0 && min(best_fitGSO)~=0
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %15d
%30.10f %15.10f %25.20f',...

t,Gbest(1),Gbest(2),Gbest_fit,solusi(1),solusi(2),min(FitGSO))};
        elseif Gbest_fit~=0 && min(best_fitGSO)==0
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %25.20f
%20.10f %15.10f %15d',...

t,Gbest(1),Gbest(2),Gbest_fit,solusi(1),solusi(2),min(FitGSO))};
        else
```

```
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %25.20f
%20.10f %15.10f %25.20f',...

t,Gbest(1),Gbest(2),Gbest_fit,solusi(1),solusi(2),min(FitGSO))};
        end
    else
        if Gbest_fit==0 && min(best_fitGSO)~=0
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %15.10f
%15d %30.10f %15.10f %15.10f %25.20f',...

t,Gbest(1),Gbest(2),Gbest(3),Gbest_fit,solusi(1),solusi(2),solusi(3)
,min(FitGSO))};
        elseif Gbest_fit~=0 && min(best_fitGSO)==0
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %15.10f
%25.20f %20.10f %15.10f %15.10f %15d',...

t,Gbest(1),Gbest(2),Gbest(3),Gbest_fit,solusi(1),solusi(2),solusi(3)
,min(FitGSO))};
        else
            hasil={char(hasil);sprintf('%5d %15.10f %15.10f %15.10f
%25.20f %20.10f %15.10f %15.10f %25.20f',...

t,Gbest(1),Gbest(2),Gbest(3),Gbest_fit,solusi(1),solusi(2),solusi(3)
,min(FitGSO))};
        end
    end
    set(handles.listbox1,'string',char(hasil),'value',t+2);
    %plot
    plot(0:length(best_fitPSO)-1,best_fitPSO,'r:','LineWidth',2);
    hold on
    plot(0:length(best_fitGSO)-1,best_fitGSO,'b','LineWidth',2);

plot(tkonvergenPSO,best_fitPSO(tkonvergenPSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);

plot(tkonvergenGSO,best_fitGSO(tkonvergenGSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);
    hold off
    ylim([-0.1 max([max(best_fitPSO) max(best_fitGSO)])]);
    pause(0.000001);
    ttoc=toc/2;
    waktu1=waktu1+ttoc;
    waktu2=waktu2+ttoc;
    if Gbest_fit==0 && min(best_fitGSO)==0
        break;
    end
end
plot(0:length(best_fitPSO)-1,best_fitPSO,'r','LineWidth',2);
hold on
plot(0:length(best_fitGSO)-1,best_fitGSO,'b:','LineWidth',2);
legend('PSO','GSO');
```

```
plot(tkonvergenPSO,best_fitPSO(tkonvergenPSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);
plot(tkonvergenGSO,best_fitGSO(tkonvergenGSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);
xlabel('Iterasi');
ylabel('Nilai Fungsi');
hold off
ylim([-0.1 max([max(best_fitPSO) max(best_fitGSO)])]);
set(handles.textrt1,'string',num2str(waktu1));
set(handles.textrt2,'string',num2str(waktu2));
%
set(handles.listbox1,'UserData',{best_fitPSO,best_fitGSO,tkonvergenP
SO,tkonvergenGSO});
end



function var3_Callback(hObject, eventdata, handles)
% hObject    handle to var3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of var3 as text
%        str2double(get(hObject,'String')) returns contents of var3
as a double


% --- Executes during object creation, after setting all properties.
function var3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to var3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu1 contents as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function func3_Callback(hObject, eventdata, handles)
% hObject    handle to func3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of func3 as text
%        str2double(get(hObject,'String')) returns contents of func3
as a double


% --- Executes during object creation, after setting all properties.
function func3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to func3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[FileName,FilePath] = uiputfile('*.jpg','Save Plot As');
if FileName~=0
    fit=get(handles.listbox1,'UserData');
    best_fitPSO=fit{1};
    best_fitGSO=fit{2};
    tkonvergenPSO=fit{3};
    tkonvergenGSO=fit{4};
    figure
    plot(0:length(best_fitPSO)-1,best_fitPSO,'r','LineWidth',2);
    hold on
    plot(0:length(best_fitGSO)-1,best_fitGSO,'b:','LineWidth',2);
    legend('PSO','GSO');

plot(tkonvergenPSO,best_fitPSO(tkonvergenPSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);

plot(tkonvergenGSO,best_fitGSO(tkonvergenGSO+1),'Marker','s','Marker
EdgeColor','k','MarkerFaceColor','y','MarkerSize',5);
    ylim([-0.1 max([max(best_fitPSO) max(best_fitGSO)])]);
    xlabel('Iterasi');
    ylabel('Nilai Fungsi');
    saveas(gcf,fullfile(FilePath,FileName));
    close(gcf);
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, ~, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.var1,'string','x');
set(handles.var2,'string','y');
set(handles.var3,'string','z');
set(handles.popupmenu1,'value',1);
set(handles.func1,'string','');
set(handles.func2,'string','');
set(handles.func3,'string','');
set(handles.editpop,'string','');
set(handles.editlb,'string','');
set(handles.editub,'string','');
set(handles.edititer,'string','');
set(handles.editv0,'string','');
set(handles.editc1,'string','');
set(handles.editc2,'string','');
set(handles.edittmin,'string','');
set(handles.edittmax,'string','');
set(handles.editl0,'string','');
set(handles.editr0,'string','');
```

```
set(handles.editrs,'string','');
set(handles.editgamma,'string','');
set(handles.editbeta,'string','');
set(handles.editnt,'string','');
set(handles.edits,'string','');
set(handles.editrho,'string','');
set(handles.textrt1,'string','');
set(handles.textrt2,'string','');
set(handles.listbox1,'string','    ','value',1,'UserData',[]);
guidata(hObject, handles);
axes(handles.axes1);
cla(handles.axes1,'reset');
xlim([0 500]);ylim([-1 5]);


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: place code in OpeningFcn to populate axes1


% --- Executes during object creation, after setting all properties.
function uipanel2_CreateFcn(~, eventdata, handles)
% hObject    handle to uipanel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called


% ------------------------------------------------------------------
--
function uipanel2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to uipanel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when uipanel2 is resized.
function uipanel2_SizeChangedFcn(hObject, eventdata, handles)
% hObject    handle to uipanel2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```