PENDEKATAN ALGORITMA METAHEURISTIK PADA PENYELESAIAN SISTEM PERSAMAAN NON-LINIER YANG MEMUAT AKAR KOMPLEKS

SKRIPSI

Oleh

Merysa Puspita Sari
NIM 141810101021

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2018

PENDEKATAN ALGORITMA METAHEURISTIK PADA PENYELESAIAN SISTEM PERSAMAAN NON-LINIER YANG MEMUAT AKAR KOMPLEKS

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Program Studi Matematika (S1) dan mencapai gelar Sarjana Sains

oleh

Merysa Puspita Sari
NIM 141810101021

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2018

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ayahanda Bambang Ruchiyanto dan Ibunda Siti Mujayanah tercinta yang senantiasa memberi doa, inspirasi, semangat serta kasih sayangnya untuk saya;
2. Adik-adikku Miranda Rizki Nurbana dan Abdul Aziz Rifaldi yang senantiasa memberi dukungan;
3. Seluruh guru dan dosen sejak taman kanak-kanak hingga perguruan tinggi yang telah memberikan ilmu dan bimbingannya;
4. Almamater Jurusan Matematika FMIPA Universitas Jember, SMA Negeri 5 Jember, SMP Negeri 4 Jember, SD Negeri Kepatihan 3 Jember dan TK Fafitri.

MOTO

However difficult life may seem, there is always something you can do an succeed at.[*)]

[*)] Stephen Hawking

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

nama : Merysa Puspita Sari

NIM : 141810101021

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Pendekatan Algoritma Metaheuristik Pada Penyelesaian Sistem Persamaan Non-linier yang Memuat Akar Kompleks" 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, 15 Juli 2018

Yang menyatakan,

Merysa Puspita Sari
NIM 141810101021

SKRIPSI

PENDEKATAN ALGORITMA METAHEURISTIK PADA PENYELESAIAN SISTEM PERSAMAAN NON-LINIER YANG MEMUAT AKAR KOMPLEKS

Oleh

Merysa Puspita Sari
NIM 141810101021

Pembimbing

Dosen Pembimbing Utama : Ahmad Kamsyakawuni, S.Si., M.Kom.

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

PENGESAHAN

Skripsi yang berjudul "Pendekatan Algoritma Metaheuristik Pada Penyelesaian Sistem Persamaan Non-linier yang Memuat Akar Kompleks" karya Merysa Puspita Sari telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Anggota I,
Ahmad Kamsyakawuni, S.Si., M.Kom. NIP 197211291998021001	Drs. Rusli Hidayat, M.Sc. NIP 196610121993031001
Anggota II,	Anggota III,
Kosala Dwidja Purnomo, S.Si., M.Si. NIP 196908281998021001	Kusbudiono, S.Si., M.Si. NIP 197704302005011001

Mengesahkan
Dekan,

Drs. Sujito, Ph.D
NIP 196102041987111001

RINGKASAN

Penerapan Algoritma Metaheuristik Pada Penyelesaian Sistem Persamaan Non-linier yang Memuat Akar Kompleks; Merysa Puspita Sari, 141810101021; 2018: 65 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Sistem persamaan non-linier adalah kumpulan dari beberapa persamaan non-linier yang dicari solusinya. Mencari solusi dari sistem persamaan non-linier biasanya dengan menggunakan metode analitik, namun ada beberapa kasus kompleks yang tidak dapat diselesaikan secara analitik sehingga dibutuhkan metode-metode baru untuk menyelesaikannya. Salah satu metode yang dapat digunakan untuk menyelesaikan sistem persamaan non-linier yaitu dengan menggunakan algoritma metaheuristik. Algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) merupakan algoritma metaheuristik yang digunakan pada penelitian ini.

Sistem persamaan non-linier yang digunakan pada penelitian ini diambil dari beberapa jurnal serta skripsi rujukan. Sistem persamaan non-linier yang diteliti berupa sistem persamaan non-linier dua variabel, sistem persamaan non-linier tiga variabel dan sistem persamaan non-linier empat variabel. Tujuan dari penelitian ini adalah mencari solusi sistem persamaan non-linier dengan mencari nilai penjumlahan dari $f(x)$ yang paling minimum diantara beberapa kandidat solusi.

Input dalam penelitian ini yaitu sistem persamaan non-linier yang akan diuji serta parameter dari algoritma *Particle Swarm Optimization*, *Firefly Algorithm* dan *Cuckoo Search*. Sistem persamaan non-linier yang menjadi objek permasalahan berupa fungsi polinomial dan fungsi trasenden yang meliputi fungsi logaritma, fungsi trigonometri derajat satu dan fungsi eksponensial. Parameter utama dari algoritma *Particle Swarm Optimization*, *Firefly Algorithm* dan *Cuckoo Search* yaitu Pop, ub (batas atas), dan lb (batas bawah). Parameter untuk algoritma *Particle Swarm Optimization* yaitu $v0$ (kecepatan awal), $c_1 = c_2$,

$Thetamax$ dan $Thetamin$. Parameter untuk *Firefly Algorithm* yaitu $gamma$, $beta0$ dan $alpha$. Parameter untuk algoritma *Cuckoo Search* yaitu $Stepsize$, Pa, serta parameter-parameter pada *Lévy Flights* yang meliputi $alphaLF$, $betaLF$, $gammaLF$ dan $deltaLF$. Sehingga, untuk *output* yang dihasilkan berupa aproksimasi solusi akar kompleks dan nilai fungsi. Solusi sistem persamaan non-linier yang diperoleh dengan menggunakan algoritma *Particle Swarm Optimization*, *Firefly Algorithm* dan *Cuckoo Search*, kemudian dibandingkan hasil akurasinya dengan mencari nilai fungsi $f(x)$ yang lebih mendekati nol.

Pada penelitian yang telah dilakukan dengan menggunakan 5 sistem persamaan non-linier yang digunakan mendapatkan hasil yang baik. Analisis hasil yang dilakukan dengan membandingkan nilai fungsi yang dihasilkan dari masing-masing algoritma. Dimana pada algoritma *Particle Swarm Optimization* didapatkan nilai fungsi lebih mendekati nol dibandingkan dengan algoritma *Firefly Algorithm* dan *Cuckoo Search* akan tetapi *Firefly Algorithm* lebih cepat mendapatkan solusi yang konvergen dibandingkan algoritma *Particle Swarm Optimization* dan *Cuckoo Search*, untuk waktu komputasi yang paling cepat diantara ketiga algoritma tersebut adalah algoritma *Cuckoo Search*. Penyelesaian sistem persamaan non-linier jika dilihat dari penelitian ini lebih baik menggunakan algoritma *Particle Swarm Optimization*, karena kekonvenannya mendapat nilai fungsi yang mendekati nol.

PRAKATA

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan Algoritma Metaheuristik Pada Penyelesaian Sitem Persamaan Non-linier yang Memuat Akar Kompleks". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahua Alam Universitas Jember.

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

1. Ahmad Kamsyakawuni, S.Si., M.Kom., selaku Dosen Pembimbing Utama dan Drs. Rusli Hidayat, M.Sc., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
2. Kosala Dwija Purnomo, S.Si., M.Si., selaku Dosen Penguji I, dan Kusbudiono, S.Si., M.Si., selaku Dosen Penguji II yang telah memberikan kritik serta sarannya terhadap penulisan skripsi ini;
3. Kusbudiono, S.Si., M.Si., selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. seluruh staf pengajar Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahua Alam Universitas Jember yang telah memberikan ilmu serta bimbingannya sehingga penulis dapat menyelesaikan skripsi ini;
5. ayahanda Bambang Ruchiyanto dan ibunda Siti Mujayanah yang senantiasa memberi doa, inspirasi dan semangat demi terselesaikannya skripsi ini;
6. adik-adikku Miranda Rizki Nurbana dan Abdul Aziz Rifaldi yang senantiasa memberikan dukungan dalam pengerjaan skripsi ini;
7. sahabat-sahabat "The Gang" dan "Sipuuut" yang selalu memberikan dorongan motivasi dan nasehat;
8. teman-teman angkatan 2014 (EXTREME) atas keceriaan, canda tawa serta dukungan yang selalu diberikan selama pembelajaran dalam masa perkuliahan;

9. 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, Juli 2018

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Permasalahan yang sering muncul dalam bidang matematika adalah mencari solusi dari sebuah sistem persamaan. Bentuk dari sistem persamaan dibagi menjadi dua, yaitu sistem persamaan linier dan sistem persamaan non-linier. Sistem persamaan non-linier adalah kumpulan dari beberapa persamaan non-linier yang dicari solusinya. Persamaan non-linier adalah persamaan yang memiliki peubah dengan pangkat terkecilnya satu atau berupa fungsi trasenden. Mencari solusi dari sistem persamaan non-linier biasanya dengan menggunakan metode analitik, namun ada beberapa kasus kompleks yang tidak dapat diselesaikan secara analitik sehingga harus menggunakan metode numerik untuk menyelesaikanya.

Metode numerik merupakan metode yang digunakan untuk memformulasikan persoalan matematika sehingga dapat dipecahkan dengan operasi hitung atau aritmatika biasa. Perhitungan secara numerik menawarkan hasil estimasi yang mendekati solusi dari metode analitik dengan galat yang kecil. Solusi dari metode numerik dihasilkan dari proses perhitungan secara berulang hingga tercapai kekonvergenan. Metode yang paling sering digunakan untuk menyelesaikan sistem persamaan non-linier adalah metode *Newton-Raphson*. Metode *Newton-Raphson* merupakan metode yang tergolong cepat dalam menyelesaikan sistem persamaan non-linier, akan tetapi masih memiliki kelemahan yang mana metode ini mengharuskan untuk menghitung fungsi $f(x_n)$ serta penetapan nilai awal x_n yang sulit dan tidak selalu menemukan akar. Kelemahan pada metode tersebut yang telah mendorong para peneliti untuk melakukan penelitian guna mencari metode yang paling efektif untuk menyelesaikan sistem persamaan non-linier, salah satunya adalah dengan menggunakan algoritma metaheuristik.

Metaheuristik merupakan algoritma yang dapat menyelesaikan masalah optimasi kompleks jika diselesaikan dengan algoritma eksak. Menurut Hiller dan Lieberman (2010), metode heuristik adalah metode yang digunakan untuk mencari solusi suatu masalah dimana solusi yang ditemukan merupakan *feasible solution*

yang terbaik. Dalam pencarian solusi yang efisien dan komprehensif, metode heuristik menggunakan mekanisme yang meniru perilaku sosial maupun strategi yang ada di alam. Menurut Madi (2013), algoritma metaheuristik memiliki kecepatan pencarian solusi optimal yang lebih baik dari algoritma heuristik, karena algoritma ini akan selalu berusaha untuk keluar dari solusi *local optima*. Meskipun tidak ada jaminan bahwa solusi yang ditemukan merupakan solusi yang optimal akan tetapi solusi yang ditemukan akan selalu mendekati solusi optimal. Beberapa algoritma metaheuristik diantaranya algoritma *Particle Swarm Optimization* (PSO), *Cat Swarm Optimization* (CSO), *Cockroach Swarm Optimization Algorithm* (CSOA), *Virus Evolutionary Genetic Algorithm* (VEGA), *Firefly Algorithm* (FA), *Cuckoo Search* (CS) dan lain sebagainya.

Baihaki (2016) dalam penelitiannya menggunakan algoritma *Cat Swarm Optimization* (CSO) untuk menyelesaikan sistem persamaan non-linier. Hasil yang didapatkan dari penerapan algoritma CSO untuk menyelesaikan sistem persamaan non-linier memberikan nilai yang konvergen dan mendekati hasil eksak, sehingga dapat disimpulkan bahwa algoritma tersebut dapat mengungguli metode *Newton-Raphson*.

Prastowo (2016) juga menggunakan algoritma metaheuristik dalam penelitiannya untuk menyelesaikan sistem persamaan non-linier. Algoritma metaheuristik yang digunakan adalah *Cockroach Swarm Optimization Algorithm* (CSOA). Hasil dari penelitiannya menunjukkan bahwa error yang didapat dari algoritma tesebut dalam menyelesaikan sistem persamaan non-linier hampir mendekati nol dan lebih akurat jika dibandingkan dengan metode *Newton-Raphson*.

Begitu pula Azmi (2018) yang membandingkan algoritma metaheuristik menyelesaikan sistem persamaan non-linier. Algoritma metaheuristik yang dibandingkan adalah *Particle Swarm Optimization* (PSO) dan *Glowworm Swarm Optimization* (GSO). Kesimpulan yang diperoleh pada penelitiannya menyatakan bahwa algoritma *Particle Swarm Optimization* (PSO) lebih baik jika dibandingkan dengan algoritma *Glowworm Swarm Optimization* (GSO) dalam menyelesaikan sistem persamaan non-linier, karena menghasilkan nilai fungsi yang mendekati solusi eksak.

Ariyaratne (2015) dalam jurnalnya persamaan non-linier yang memiliki akar real maupun akar kompleks dengan menggunakan fungsi trigonometri, trasenden, weierstrass, dan polinom yang memiliki akar tunggal maupun lebih dapat diselesaikan menggunakan algoritma metaheuristik. Algoritma yang digunakan pada penelitiannya yaitu *Modified Firefly Algorithm* (MOD FA) yang kemudian dibandingkan dengan *Genetic Algorithm* (GA). Kesimpulan yang diperoleh dari penelitiannya tersebut menyatakan bahwa algoritma *firefly* yang telah dimodifikasi menunjukkan hasil yang lebih baik dalam hal akurasi dan waktu komputasi dari pada *Genetic Algorithm* (GA).

Dalam penelitiannya, Farikha (2018) menggunakan *Cockroach Swarm Optimization Algorithm* (CSOA) untuk menyelesaikan persamaan non-linier yang memuat akar kompleks. Hasil yang didapat pada penelitiannya tersebut menunjukkan bahwa *Cockroach Swarm Optimization Algorithm* (CSOA) mampu menyelesaikan permasalahan non-linier yang memuat akar kompleks dengan hasil yang lebih konvergen jika dibandingkan dengan metode *Newton-Raphson*.

Berdasarkan uraian diatas, Baihaki (2016), Prastowo (2016), dan Azmi (2018) membahas tentang penyelesaian sistem persamaan non-linier dengan menggunakan algoritma metaheuristik serta Ariyaratne (2015) dan Farikha (2018) yang membahas tentang penyelesaian persamaan non-linier yang memuat akar kompleks dengan menggunakan algoritma metaheuristik. Algoritma *Particle Swarm Optimization* (PSO) memiliki kelebihan karena nilai fungsinya lebih konvergen jika dibandingkan dengan algoritma *Glowworm Swarm Optimization* (GSO). *Firefly Algoritm* (FA) menunjukkan hasil yang baik dalam hal akurasi dan waktu komputasi sedangkan algoritma *Cuckoo Search* (CS) menunjukkan hasil yang baik dalam hal akurasi pada perhitungannya. Sehingga menjadi bahan pertimbangan bagi penulis untuk melakukan penelitian tentang pendekatan algoritma metaheuristik yang meliputi algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) pada penyelesaian sistem persamaan non-linier yang memuat akar kompleks.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan masalah dalam penelitian ini yaitu:

a. Bagaimana penerapan algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) untuk menyelesaian sistem persamaan non-linier yang memuat akar kompleks?
b. Bagaimana perbandingan hasil akurasi dari algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) pada penyelesaian sistem persamaan non-linier yang memuat akar kompleks?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah fungsi yang digunakan merupakan fungsi polinomial dan fungsi trasenden yang meliputi fungsi logaritma, fungsi trigonometri derajat satu dan fungsi eksponensial.

1.4 Tujuan Penelitian

Sesuai dengan rumusan masalah dan latar belakang diatas, maka tujuan pada penelitian ini antara lain:

a. Mengetahui penerapan dari algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) untuk menyelesaian sistem persamaan non-linier yang memuat akar kompleks.
b. Mengetahui perbandingan hasil akurasi dari algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS) pada penyelesaian sistem persamaan non-linier yang memuat akar kompleks.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini antara lain:

a. Menambah wawasan mengenai metode baru dalam menyelesaikan masalah penyelesaian sistem persamaan non-linier yang memuat akar kompleks khususnya menggunakan algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) dan *Cuckoo Search* (CS).

b. Memberi kontribusi terhadap perkembangan pengetahuan dalam bidang pendidikan, khususnya dalam ruang lingkup masalah penyelesaian sistem persamaan non-linier yang memuat akar kompleks.
c. Memberikan motivasi pada peneliti lain untuk melakukan penelitian tentang penyelesaian sistem persamaan non-linier yang memuat akar kompleks dengan menggunakan metode-metode baru yang berbeda.

BAB 2. TINJAUAN PUSTAKA

2.1 Persamaan Non-linier

Persaman non-linier adalah semua persamaan yang bukan merupakan persamaan linier dengan peubah terkecil sama dengan satu atau berupa fungsi trasenden. Beberapa fungsi yang termasuk persamaan non-linier adalah fungsi trasenden seperti $\sin 2x - \sec^{-1} x = 5$ dan fungsi non trasenden yang merupakan polinomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 = 0$, dimana polinom merupakan bentuk suku-suku dengan banyak terhingga yang disusun dari peubah dan konstanta (Basuki, 2004).

Penyelesaian persamaan non-linier adalah penentuan akar-akar persamaan non-linier, dengan menentukan nilai x yang menyebabkan $f(x) = 0$. Akar dari persamaan non-linier dapat diperoleh secara analitik maupun numerik.

2.2 Sistem Persamaan Non-linier

Sistem persamaan non-linier merupakan sistem persamaan yang mempunyai bentuk persamaan yang kompleks, sehingga tidak dapat diselesaikan secara analitik. Apabila suatu persamaan tidak dapat diselesaikan secara analitik, maka persamaan tersebut dapat diselesaikan secara numerik (Munir, 2003).

Sistem persamaan non-linier adalah kumpulan dari beberapa persamaan non-linier yang dicari solusinya. Menurut Devi (2011), sistem persamaan non-linier adalah suatu sistem dengan n buah persamaan non-linier yang harus diselesaikan secara simultan dalam suatu sistem. Sistem persamaan non-linier merupakan kumpulan dari dua atau lebih persamaan non-linier. Bentuk umum dari sistem persamaan non-linier adalah sebagai berikut:

$$f(x) = \begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \tag{2.1}$$

dengan $f_1, f_2, \dots, f_n$ adalah fungsi-fungsi non linier dalam variabel $x_1, x_2, \dots, x_n$. Penyelesaian dari sistem persamaan (2.1) adalah himpunan nilai $x =$

$\{x_{11}, x_{12}, \dots, x_{1n}, x_{21}, x_{22}, \dots, x_{2n}, \dots, x_{nn}\}$ yang memenuhi sistem persamaan diatas.

Salah satu contoh dari sistem persamaan non-linier adalah sebagai berikut (Nasiha, 2008).

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1 + \cos(x_1 x_2) - x_3^2 - 1{,}1 = 0 \\ f_2(x_1, x_2, x_3) &= x_1 - 10x_2 - e^{x_2 x_3} + 0{,}8 \quad = 0 \\ f_3(x_1, x_2, x_3) &= x_1 x_3 + x_2^2 - x_3 - 0{,}3 \qquad = 0 \end{aligned} \tag{2.2}$$

Contoh lain dari sistem persamaan non-linier, yaitu sebagai berikut (Graiioo, 2011).

$$\begin{aligned} f_1(x_1, x_2) &= \cos(2x_1) - \cos(2x_2) - 0{,}4 \qquad = 0 \\ f_2(x_1, x_2) &= 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_2) - 1{,}2 = 0 \end{aligned} \tag{2.3}$$

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

2.3 Sistem Bilangan Kompleks

Himpunan bilangan terbesar di dalam matematika adalah himpunan bilangan kompleks. Himpunan bilangan riil, merupakan himpunan bagian dari himpunan bilangan kompleks. Secara umum bilangan kompleks terdiri dari dua bagian, yaitu bagian riil dan bagian imajiner. Bagian imajiner dicirikan dengan adanya bilangan imajiner yang dinotasikan dengan i dan didefinisikan sebagai $i = \sqrt{-1}$ maka $i^2 = -1$.

Definisi 2.1

Bilangan kompleks adalah pasangan terurut dari dua bilangan riil x dan y, yang dinyatakan oleh (x, y) (Sardi, 2014).

Pernyataan diatas merupakan definisi dari bilangan kompleks. Bilangan kompleks dilambangkan dengan $z = (x, y)$. Bilangan riil x disebut bagian riil dari z, ditulis $Re(z)$. Bilangan riil y disebut bagian imajiner dari z, ditulis $Im(z)$. Beberapa pasangan terurut diidentifikasikan secara khusus, yaitu

$(x, 0) = x$, merupakan bilangan riil x

$(0,1) = i$, dinamakan satuan imajiner

Definisi 2.2

Dua bilangan kompleks $z_1 = (x_1, y_1)$ dan $z_2 = (x_2, y_2)$ dikatakan sama, ditulis $z_1 = z_2$, jika $x_1 = x_2$ dan $y_1 = y_2$. Khususnya $z = (x, y) = (0,0)$ jika dan hanya jika, $x = 0$ dan $y = 0$ (Sardi, 2014).

Himpunan bilangan kompleks diberi notasi $\mathbb{C}$, jadi $\mathbb{C} = \{z | z = x + iy, x \epsilon \mathbb{R}, y \epsilon \mathbb{R}\}$. Jika $Im(z) = 0$ maka bilangan kompleks z menjadi bilangan riil x, sehingga bilangan riil adalah keadaan khusus dari bilangan kompleks, sehingga $\mathbb{R} \subset \mathbb{C}$. Jika $Re(z) = 0$ dan $Im(z) \neq 0$, maka z menjadi iy dan dinamakan bilangan imajiner murni. Bilangan imajiner murni dengan $y = 0$, yakni bilangan i, dinamakan satuan imajiner.

Sistem bilangan kompleks merupakan perluasan dari sistem bilangan riil. Jika solusi dari suatu persamaan tidak dapat dinyatakan dengan bilangan riil. Maka, persamaan tersebut perlu didefinisikan dengan bilangan kompleks. Bilangan kompleks dituliskan sebagai pasangan terurut dua bilangan riil, $z = x + iy$ atau $z = x + yi$. Bilangan riil x disebut juga bagian riil dari bilangan kompleks, dan bilangan riil y disebut bagian imajiner. Antara bagian riil atau bagian imajiner dari bilangan kompleks bisa saja bernilai nol. Jika pada suatu bilangan kompleks, nilai y adalah 0, maka bilangan kompleks tersebut sama dengan bilangan riil x. Nilai r dinamakan modulus atau nilai mutlak dari z, yang dirumuskan sebagai berikut:

$$r = |z| = \sqrt{x^2 + y^2} \tag{2.4}$$

Jika dalam sistem bilangan riil dapat direpresentasikan dalam satu garis lurus, maka bilangan kompleks juga dapat direpresentasikan dalam satu bidang kartesian yang dinamakan bidang kompleks atau bidang Argand. Bilangan kompleks, dapat direpresentasikan dalam bentuk pasangan berurutan (x, y), maka untuk sumbu x adalah sumbu riil dan sumbu y adalah sumbu imajiner untuk setiap bilangan kompleks $z = x + iy$ dalam sistem koordinat kartesius (Humam, 2015).

2.4 Algoritma *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) adalah algoritma *swarm intelligence* yang berdasarkan pada populasi, PSO ditemukan oleh Kennedy dan Eberhart pada tahun 1995. Algoritma PSO terinspirasi oleh perilaku sosial dari kawanan ikan atau sekelompok burung camar yang terbang bersama-sama untuk mencari makan atau sarang. Algoritma PSO dapat dengan mudah mencapai titik global maupun optimal karena kemudahan dalam penerapannya untuk memecahkan masalah dan memiliki performa yang konsisten. Algoritma PSO terbukti merupakan algoritma yang baik dan efektif untuk menyelesaikan permasalahan optimasi (Dhanasaputra, 2010).

Particle Swarm Optimization (PSO) mensimulasikan perilaku sosial dari sekelompok burung atau ikan. Misalnya, seekor burung sedang terbang secara acak mencari makan di sebuah area dimana hanya terdapat 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 makan adalah dengan cara mengikuti burung yang lokasinya paling dekat dengan makanan tersebut. Satu burung menyatakan satu solusi di dalam ruang masalah yang direpresentasikan sebagai "partikel" dalam algoritma PSO. Algoritma PSO menggunakan populasi dari sekumpulan partikel, dimana setiap partikel mewakili sebuah solusi yang mungkin untuk sebuah permasalah optimasi. Semua partikel memiliki nilai *fitness* yang dievaluasi oleh fungsi *fitness* yang akan dioptimalkan dan kecepatan (*velocity*) yang diadaptasi dari area pencarian untuk berpindah dan disimpan sebagai posisi terbaik yang pernah dicapai. Sehingga dari perilaku kumpulan burung tersebut yang kemudian digunakan untuk memcahkan masalah optimasi (Erny, 2013).

Pada sistem PSO, masing-masing partikel terbang mengitari ruang pencarian multi dimensional (*multidimensional search space*) dan menyesuaikan posisinya berdasarkan pengalaman pribadinya dan pengalaman partikel di sebelahnya dalam satu populasi. Tiap partikel memiliki posisi $z_i = [z_{i1}, z_{i2}, \dots, z_{iN}]$ dan kecepatan $v_i = [v_{i1}, v_{i2}, \dots, v_{iN}]$ 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 non-linier. Inisialisasi algoritma PSO dimulai dengan menetapkan posisi awal partikel secara acak (solusi) dan kemudian mencari nilai optimal dengan memperbarui posisinya. Pada 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:

$$v_i^{t+1} = \theta v_i^t + c_1 r_1 (pbest_i^t - z_i^t) + c_2 r_2 (gbest_i^t - z_i^t) \quad (2.5)$$

$$z_i^{t+1} = z_i^t + v_i^{t+1} \quad (2.6)$$

Dimana v_i^t adalah kecepatan partikel ke-i pada iterasi ke-k, dan z_i^t adalah solusi (posisi) partikel ke-i pada iterasi ke-t. c_1, c_2 adalah konstanta positif, pada umumnya nilai $c_1 = c_2 = 2$ dan r_1, r_2 adalah dua variabel acak yang berdistribusi *uniform* antara 0 sampai 1. Pada persamaan di atas θ adalah koefisien berat inersia yang menunjukkan pengaruh perubahan kecepatan dari vektor lama ke vektor yang baru (Rosita, 2012).

θ merupakan koefisien berat inersia yang digunakan untuk mengurangi kecepatan. Biasanya nilai θ dibuat sedemikian hingga semakin banyak iterasi yang didapatkan, maka semakin mengecil kecepatan partikel. Nilai ini bervariasi secara linier dalam rentang $0{,}4 - 0{,}9$. Nilai koefisien berat inersia yang tinggi menambah porsi pencarian global (*global exploration*), sedangkan nilai yang rendah lebih menekankan pencarian lokal (*local search*). Agar tidak terlalu menitikberatkan pada salah satu pencarian dan tetap mencari area pencarian yang baru dalam ruang berdimensi tertentu, maka perlu dicari nilai koefisien berat inersia (θ) yang secara imbang untuk menjaga pencarian global dan lokal. Agar dapat mecapai hal tersebut dan mempercepat proses konvergensi, suatu koefisien berat inersia yang mengecil nilainya dengan bertambahnya iterasi dapat dicari dengan rumusan sebagai berikut:

$$\theta = \theta_{max} - i \frac{\theta_{max} - \theta_{min}}{i_{max}} \quad (2.7)$$

Dimana θ_{max} dan θ_{min} masing-masing adalah nilai awal dan nilai akhir koefisien berat inersia, i_{max} adalah jumlah iterasi maksimum yang digunakan dan i adalah iterasi. Biasanya menggunakan nilai $\theta_{max} = 0{,}9$ dan $\theta_{min} = 0{,}4$ (Azmi, 2018).

c_1 dan c_2 adalah parameter, r_1 dan r_2 adalah bilangan acak yang berdistribusi seragam pada selang $[0,1]$. Koefisien c_1 dan c_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 daerah sasaran. Pada umumnya nilai $c_1 = c_2 = 2$, pemberian nilai yang berbeda pada c_1 dan c_2 biasanya mengarah pada peningkatan kerja.

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

Setiap partikel bergerak dalam ruang pencarian dengan kecepatan sesuai dengan solusi terbaik individu dan solusi terbaik kelompok sebelumnya. Kecepatan v_i^{t+1}, pada persamaan (2.5) terdiri dari tiga bagian yaitu bagian pertama adalah v_i^t menunjukkan kecepatan partikel sebelumnya. Bagian kedua $c_1 r_1$ menunjukkan proses penyelesaian dari pengalaman individu. Bagian ketiga $c_2 r_2$ menunjukkan proses penyelesaian dari pengalaman individu 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* dapat dijabarkan sebagai berikut:

a. Inisialisasi parameter Algoritma *Particle Swarm Optimization* yang meliputi posisi awal (z_i), kecepatan awal partikel (v_i), dengan $i = 1,2,\dots,S$ dan S adalah ukuran *swarm* dan iterasi max.

b. Evaluasi nilai fungsi tujuan untuk setiap partikel $\left(f(z_i)\right)$ dan nilai *fitness* yang dirumuskan sebagai berikut:

$$fitness = \frac{1}{1 + f(z_i)}$$

c. Tentukan $pbest$ awal dan $gbest$ awal.

d. Perbarui kecepatan dengan persamaan:

$$v_i^{t+1} = \theta v_i^t + c_1 r_1 (pbest_i^t - z_i^t) + c_2 r_2 (gbest_i^t - z_i^t)$$

dimana:

1) $pbest$ adalah individu dengan nilai partikel terbaik dari masing-masing individu.

2) $gbest$ adalah individu dengan nilai partikel terbaik dari semua individu dalam *swarm*.

3) θ adalah koefisien berat inersia, dengan rumusan:

$$\theta = \theta_{max} - i\frac{\theta_{max} - \theta_{min}}{i_{max}}$$

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

4) c_1 dan c_2 adalah konstanta positif.

5) r_1 dan r_2 adalah bilangan acak diantara 0 dan 1.

e. Perbarui posisi individu baru dengan persamaan:

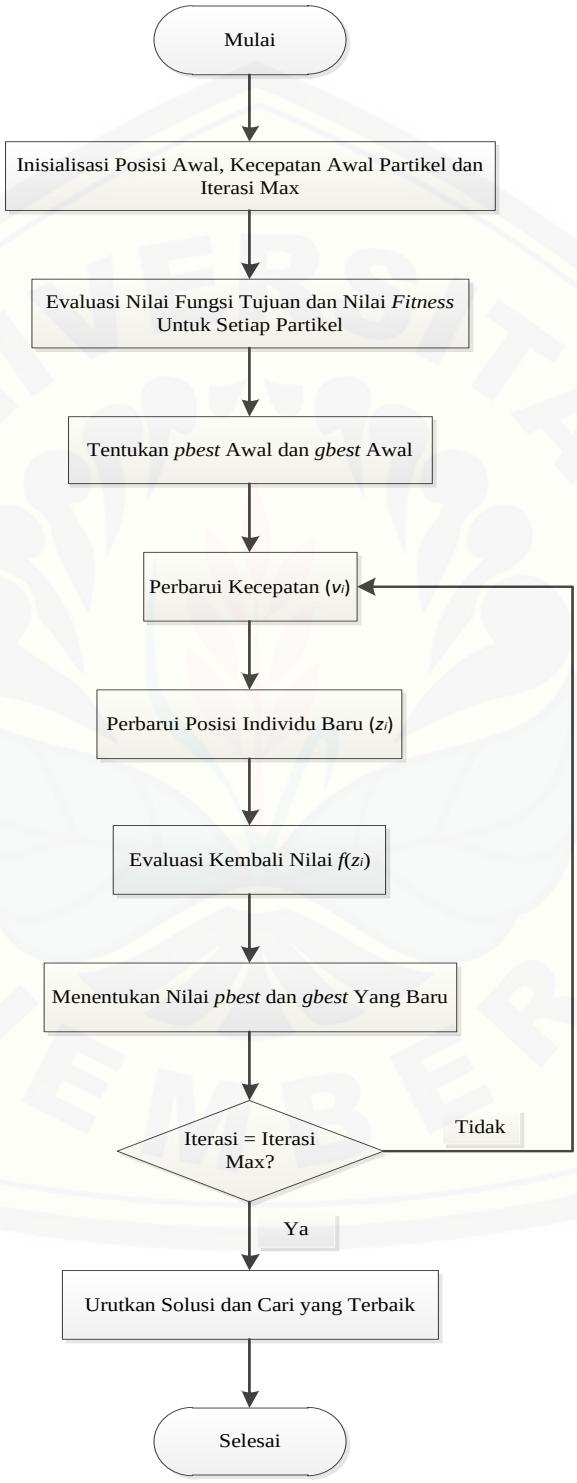
$$z_i^{t+1} = z_i^t + v_i^{t+1}$$

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

$$gbest = \begin{cases} pbest_i, fitness(pbest_i) < fitness(gbest) \\ gbest, fitness(pbest_i) \geq fitness(gbest) \end{cases}$$

g. Melakukan proses diatas sampai batas iterasi telah dipenuhi.

Langkah-langkah pada algoritma *Particle Swarm Optimization* (PSO) dapat diilustrasikan pada *flowchart* di bawah ini:



Gambar 2.1 *Flowchart* Algoritma *Particle Swarm Optimization*

2.5 *Firefly Algorithm* (FA)

Firefly Algorithm (FA) pertama kali dikembangkan oleh Xin-She Yang pada akhir tahun 2007 sampai pada tahun 2008 di Cambridge University, yang didasarkan pada pola berkedip dan perilaku kunang-kunang. Menurut Yang (2010), FA menggunakan tiga aturan yang dianggap ideal, yaitu:

a. Kunang-kunang bersifat unisex, sehingga satu kunang-kunang dapat tertarik dengan kunang-kunang lainnya tanpa melihat jenis kelaminnya.
b. Ketertarikan kunang-kunang akan sebanding dengan tingkat kecerahan kunang-kunang tersebut. Dengan ketentuan bahwa semakin jauh jarak antar kunang-kunang, maka tingkat kecerahan kunang-kunang akan menurun atau menghilang. Jadi untuk setiap kunang-kunang yang berkedip, kunang-kunang yang kurang terang (redup) akan mendekati kunang-kunang yang lebih terang. Jika dari kedua kunang-kunang tidak ada yang lebih terang maka kunang-kunang akan bergerak secara acak.
c. Kecerahan pada kunang-kunang akan ditentukan oleh fungsi tujuan dari masalah yang diberikan.

Berikut merupakan beberapa istilah yang digunakan dalam *Firefly Algorithm* (FA) dan definisinya menurut Yang (2010):

2.5.1 Intesitas Cahaya

Dalam algoritma FA, terdapat dua masalah penting yaitu variasi intensitas cahaya dan perumusan *atractiveness*. Kecerahan pada kunang-kunang akan ditentukan oleh fungsi tujuan dan *atractiveness* sebanding dengan kecerahan, dengan demikian untuk setiap dua kunang-kunang yang berkedip, kunang-kunang dengan cahaya yang kurang terang akan bergerak ke arah kunang-kunang yang cahaya lebih cerah.

Intesitas cahaya pada kunang-kunang dipengaruhi oleh fungsi tujuan. Tingkat intensitas cahaya untuk masalah meminimumkan sebuah kunang-kunang x dapat dilihat sebagai $I(x) = \frac{1}{1+f(x)}$. Nilai $I(x)$ merupakan tingkat intensitas cahaya pada

kunang-kunang x yang berbanding terbalik terhadap solusi fungsi tujuan permasalahan yang akan dicari $f(x)$.

Atractiveness (β) bernilai relatif, karena intensitas cahaya harus dilihat dan dinilai oleh kunang-kunang lain. Dengan demikian, hasil penilaian akan berbeda tergantung dari jarak antara kunang-kunang yang satu dengan yang lainnya r_{ij}. Selain itu, intensitas cahaya akan menurun dari sumbernya dikarenakan terserap oleh media, misalnya udara. Sehingga dapat ditentukan *atractiveness* (β) dengan jarak r sebagai berikut:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.8}$$

Dengan β_0 adalah daya tarik saat tidak ada jarak antara kunang-kunang ($r = 0$) dan $\gamma \in [0{,}100]$ adalah koefisien penyerapan cahaya.

2.5.2 *Distance*

Distance atau jarak antara kunang-kunang i dan j pada posisi z_i dan z_j masing-masing adalah jarak kartesian yang dirumuskan sebagai berikut:

$$r_{ij} = \| z_i - z_j \| = \sqrt{\sum_{t=1}^{n} \left(z_i^t - z_j^t\right)^2} \tag{2.9}$$

Dengan z_i^t adalah komponen ke-t dari z_i pada *firefly* i dan z_j^t adalah komponen ke-t dari z_j pada *firefly* j.

2.5.3 *Movement*

Movement adalah pergerakan yang dilakukan *firefly* i karena ketertarikan terhadap *firefly* lain j, yang intensitas cahayanya lebih terang. Dengan adanya *movement*, maka posisi *firefly* atau solusi dari *firefly* tersebut akan berubah sesuai rumus berikut:

$$z_i^{t+1} = z_i^t + \beta\left(z_j^t - z_i^t\right) + \alpha\left(rand - \frac{1}{2}\right) \tag{2.10}$$

Dengan suku pertama merupakan posisi lama dari *firefly*, suku kedua terjadi karena ketertarikan, suku ketiga adalah pergerakan random *firefly* dengan α adalah koefisien parameter random dan $rand$ adalah bilangan random pada interval $[0{,}1]$.

Pada sebagian besar implementasi *Firefly Algorithm* menggunakan $\beta_0 = 1, \alpha \in [0,1]$ dan $\gamma \in [0,100]$ (Ariyaratne, 2015).

2.5.4 Proses *Firefly Algorithm* (FA)

Menurut Yang (2010), *Firefly Algorithm* dijalankan dengan cara sebagai berikut:

a. Inisialisasi parameter *Firefly Algorithm* yang meliputi posisi awal (z_i), γ, β_0, serta MaxGen.
b. Membangkitkan secara random populasi awal sebanyak n *firefly*.
c. Menghitung intensitas cahaya tiap *firefly* $I(x)$ berdasarkan nilai fungsi tujuan $f(x)$.
d. Membandingkan intensitas cahaya tiap *firefly* dengan *firefly* lainnya. Apabila terdapat *firefly* yang intensitas cahayanya lebih besar, akan diperbarui pergerakan *firefly*.
e. Perbarui pergerakan *firefly* menggunakan persamaan *movement* sebagai berikut:

$$z_i^{t+1} = z_i^t + \beta e^{-\gamma r^2}\left(z_j^t - z_i^t\right) + \alpha\left(rand - \frac{1}{2}\right)$$

dimana:

1) β adalah koefisien *atractiveness*, dengan rumusan:

$$\beta = \beta_0 e^{-\gamma r^2}$$

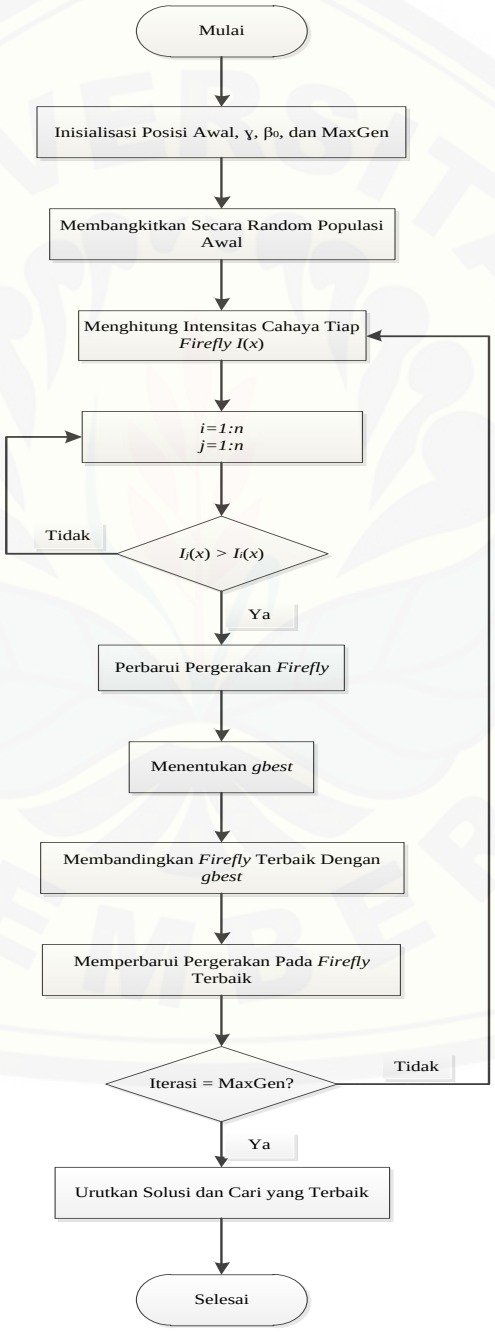
β_0 adalah daya tarik saat tidak ada jarak antara kunang-kunang, γ adalah koefisien penyerapan cahaya dan r adalah jarak antara kunang-kunang i dan j.

2) α koefisien parameter random.
3) $rand$ adalah bilangan acak diantara 0 dan 1.

f. Menentukan *g-best*. Pada iterasi pertama, *firefly* terbaik (*firefly* dengan intensitas cahaya terbesar) adalah *g-best*.
g. Membandingkan *firefly* terbaik pada tiap iterasi dengan *g-best* yang telah diperoleh. Apabila intensitas cahaya *firefly* terbaik saat itu lebih besar daripada *g-best* maka *firefly* tersebut menjadi *g-best*.

h. Melakukan *movement* dengan persamaan (2.10) pada *firefly* terbaik dan menggabungkannya dengan *firefly* yang lain untuk menjadi populasi awal pada iterasi selanjutnya.
i. Melakukan proses diatas sampai batas iterasi telah dipenuhi.

Langkah-langkah pada algoritma *Firefly Algorithm* (FA) dapat diilustrasikan pada *flowchart* di bawah ini:



Gambar 2.2 *Flowchart Firefly Algorithm*

2.6 Algoritma *Cuckoo Search* (CS)

Cuckoo search merupakan algoritma metaheuristik yang dikembangkan oleh Xin-she Yang dan Suash Deb pada tahun 2009. Algoritma ini terinspirasi oleh sifat parasit dari beberapa spesies *cuckoo* yang meletakkan telurnya di sarang burung inang lainnya (dari spesies lain). Beberapa burung inang dapat terlibat konflik langsung dengan *cuckoo* yang mengganggu. Misalnya, jika seekor burung tuan rumah menemukan telur bukan milik mereka sendiri, maka ia akan membuang telur asing tersebut atau hanya meninggalkan sarangnya dan membangun sarang baru di tempat lain. Beberapa spesies *cuckoo* seperti *the New World brood-parasit Tapera* telah berevolusi sedimikian rupa sehingga warna dan pola telurnya meniru beberapa spesies inang yang terpilih (Payne, 2005).

Yang dan Deb (2009) memformulasikan *Cuckoo Search* (CS) dengan memanfaatkan *Lévy Flights* yang merupakan pengembangan dari *random walk*. *Lévy Flights* merupakan *random walk* yang terdistribusi oleh *Lévy*. *Lévy Flights* terbukti dapat memetakan lalat buah dalam mencari makan. Lalat buah dalam mencari makan akan berkonsentrasi pada satu titik kemudian jika lalat buah tersebut merasa makanan disana sudah habis, maka akan mencari ke tempat lain. pada penelitian tesebut juga dijelaskan bahwa *Lévy Flights* membantu algoritma CS dalam pencarian karena langkah pencariannya yang semakin lebar. Algoritma CS yang menggunakan *Lévy Flights* memiliki ketepatan yang lebih baik dari algoritma optimasi yang lain dalam menentukan titik optimum.

2.6.1 Perilaku Burung *Cuckoo*

Yang dan Deb (2013) menjelaskan lebih lanjut mengenai perilaku unik burung *cuckoo* sebagai burung yang memiliki perilaku parasit. Burung *cuckoo* memiliki strategi reproduksi yang unik, dimana burung *cuckoo* akan menaruh telurnya disarang burung lain, antara burung yang memiliki jenis yang berbeda maupun pada sarang burung *cuckoo* lainnya. Setiap burung *cuckoo* hanya menaruh satu butir telurnya di sarang burung lain yang dipilih secara acak untuk meningkatkan kemungkinan menetaskan telurnya tersebut. Mungkin saja terjadi konflik antara burung *cuckoo* dengan burung asli, sehingga pada saat setelah burung *cuckoo*

meletakkan telurnya pada sarang lain, burung inang akan membuang telur burung *cuckoo* tersebut atau meninggalkan sarangnya dan membuat sarang yang baru. Sarang burung yang menghasilkan generasi *cuckoo* terbaik akan melanjutkan proses ke generasi berikutnya. Setiap pergantian generasi, jumlah dari pemilik sarang burung asli akan diatur kembali dan pemilik asli sarang ini mempunyai peluang untuk mengenali telur burung *cuckoo* yang ditaruh di sarangnya. Jika terjadi kasus anak burung *cuckoo* pendatang lebih dahulu menetas, maka telur burung yang belum menetas akan dibuang dari sarang agar anak burung *cuckoo* pendatang tersebut mendapat lebih banyak makanan. Dalam hal ini, bila pemilik asli sarang menemukan telur burung *cuckoo*, maka pemilik tersebut dapat memilih untuk membuang telur tersebut ataupun meninggalkan sarangnya.

2.6.2 Karakteristik Algoritma *Cuckoo Search* (CS)

Yang dan Deb dalam penelitiannya pada tahun 2009 mengungkapkan bahwa terdapat beberapa aturan yang harus dipenuhi dalam penggunaan algoritma ini, diantaranya:

a. Setiap burung meletakkan suatu telur pada satu waktu dan kemudian membuang telur pada sarang yang dipilih acak.
b. Sarang burung yang dianggap paling baik akan dilanjutkan untuk generasi selanjutnya.
c. Jumlah sarang burung dalam satu koloni adalah tetap.
d. Peluang untuk mengenali telur burung *cuckoo* (P_a) yang ditaruh di sarang pemilik aslinya adalah 0 sampai 1.

Aturan terakhir dapat didekati dengan parameter P_a untuk menentukan solusi terburuk dari n sarang yang akan diganti dengan sarang baru secara acak. Pada masalah maksimasi, untuk mempermudah dalam pengaplikasiannya maka dapat digunakan representasi sederhana bahwa setiap telur dalam sarang menggambarkan sebuah solusi, dan telurnya merepresentasikan sebuah solusi baru. Tujuannya adalah untuk menggunakan solusi baru yang berpotensi lebih baik untuk menggantikan solusi pada sarang. Kemudian telur-telur pada sarang tersebut akan diseleksi dan dievolusikan dengan membuang telur-telur yang dianggap kurang

baik. Pada beberapa keadaan, sarang induk asli bisa saja memiliki dua telur, sehingga dengan kata lain suatu sarang dapat berisikan lebih dari satu solusi. Namun untuk menyederhanakan permasalahan, suatu sarang hanya dapat menyimpan satu solusi saja (Yang, 2010).

2.6.3 Mekanisme Algoritma *Cuckoo Search* (CS)

Untuk menghasilkan generasi baru $z^{(t+1)}$, maka akan digunakan proses pengacakan langkah dengan *Lévy Flights* yang dapat dilihat sebagai berikut:

$$z_i^{(t+1)} = z_i^t + s \oplus Lévy\ (\alpha, \beta, \gamma, \delta) \tag{2.11}$$

Dimana $z_i^{(t+1)}$ merupakan solusi baru, z_i^t adalah solusi lama, s merupakan ukuran langkah yang dikaitkan dengan tingkatan masalah yang dikerjakan dan tanda $\oplus$ yang memiliki arti kali atau dikalikan. *Lévy Flights* atau yang disebut *Lévy Stable Distribution* pada matlab memiliki beberapa parameter, antara lain (Arti, 2017):

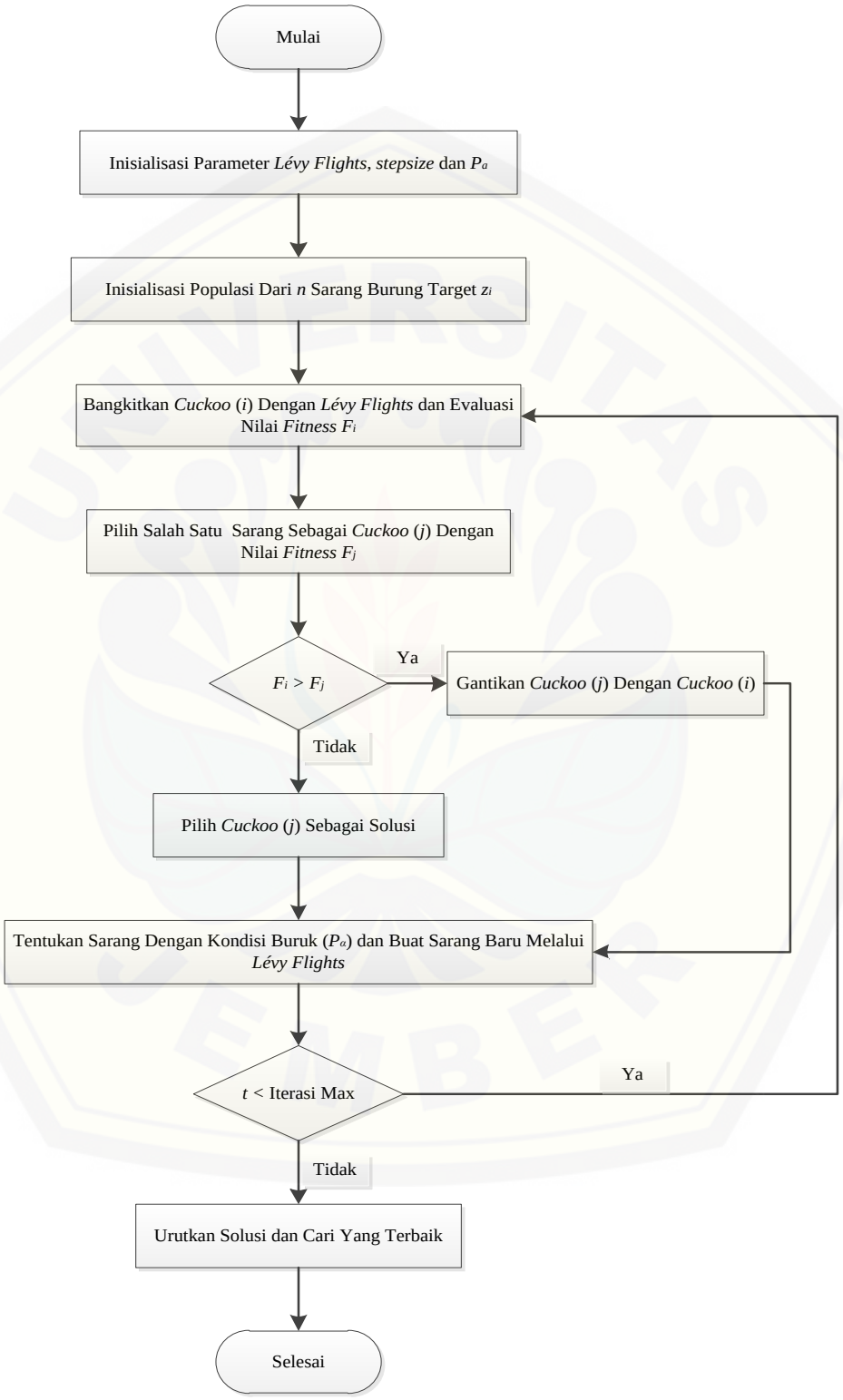
a. Alpha (α), merupakan parameter bentuk pertama dengan selang $0 < \alpha \leq 2$.
b. Betha (β), merupakan parameter bentuk kedua dengan selang $-1 \leq \beta \leq 1$.
c. Gamma (γ), merupakan parameter skala yang memiliki selang $0 < \gamma < \infty$.
d. Delta (δ), merupakan parameter lokasi yang memiliki selang $-\infty < \delta < \infty$.

Langkah-langkah menjalankan algoritma *Cuckoo Search* (CS) adalah sebagai berikut:

a. Inisialisasi parameter *Lévy Flights*. Kemudian inisialisasi jumlah populasi dengan menggunakan nilai vektor acak, nilai *stepsize* yang akan digunakan, serta nilai parameter peluang (P_a).
b. Pembangkitan solusi awal
   1) Bangkitkan z_i.
   2) Hitung nilai *fitness*.
   3) Solusi dengan *fitness* terbesar ditetapkan sebagai *Best-So-Far*.
c. Bangkitkan *cuckoo* (i) secara acak dengan *Lévy Flights* berdasarkan *Best-So-Far* dan evaluasi nilai *fitness* F_i.

d. Pilih salah satu sarang sebagai *cuckoo* (j) secara acak dengan nilai fungsi *fitness* F_j.
e. Jika dipenuhi $F_i > F_j$ gantikan cuckoo j dengan cuckoo i.
f. Tentukan solusi terburuk dengan menggunakan parameter peluang (P_a) dan digantikan dengan solusi baru menggunakan *random walk* dari *Lévy Flights.*
g. Simpan sarang terbaik yang berhasil lolos. Jika lebih baik dari *Best-So-Far*, maka akan menggantikan *Best-So-Far*.
h. Jika generasi *cuckoo* (t) mencapai generasi maksimal, maka *cuckoo* dengan *fitness* terbesar adalah solusi akhir.

Langkah-langkah pada algoritma *Cuckoo Search* (CS) dapat diilustrasikan pada *flowchart* di bawah ini:



Gambar 2.3 *Flowchart* Algoritma *Cuckoo Search*

BAB 3. METODE PENELITIAN

Pada penelitian ini, penulis akan melakukan pendekatan algoritma metaheuristik untuk menyelesaikan sistem persamaan non-linier yang memuat akar komples. Algoritma metaheuristik yang digunakan meliputi algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA), dan *Cuckoo Search* (CS). Langkah-langkah pada penelitian ini adalah sebagai berikut:

a. Studi Literatur

   Studi literatur yang dilakukan pada penelitian ini dilakukan dengan mempelajari berbagai referensi, misalnya melalui media internet, jurnal, maupun buku-buku yang berhubungan dengan sistem persamaan non-linier terutama permasalahan yang memuat akar kompleks serta pendekatan algoritma metaheuristik yang meliputi algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA), dan *Cuckoo Search* (CS).

b. Menentukan Masalah

   Masalah yang akan diteliti adalah penyelesaian sistem persamaan non-linier yang memuat akar kompleks. Fungsi yang digunakan disesuaikan dengan batasan masalah pada penelitian ini. Fungsi-fungsi tersebut merujuk pada jurnal, skripsi, maupun penelitian yang telah dilakukan sebelumnya.

c. Mengolah Data dengan Pendekatan Algoritma Metaheuristik

   Permasalahan yang menjadi objek pada penelitian ini akan diolah dengan menggunakan algoritma metaheuristik yang meliputi algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA), dan *Cuckoo Search* (CS). Adapun langkah-langkah yang harus dilakukan adalah sebagai berikut:

   1) Langkah-langkah dari algoritma *Particle Swarm Optimization* (PSO) adalah sebagai berikut:
      a) Inisialisasi parameter algoritma *Particle Swarm Optimization* (PSO).
      b) Evaluasi nilai fungsi tujuan dan nilai fitness untuk setiap partikel.
      c) Menentukan $pbest$ awal dan $gbest$ awal.
      d) Perbarui kecepatan partikel.
      e) Perbarui posisi individu baru.

f) Menentukan $pbest$ dan $gbest$ yang baru.
g) Berhenti jika iterasi maksimal telah terpenuhi.
h) Mengurutkan dan mencari solusi terbaik.

2) Langkah-langkah dari *Firefly Algorithm* (FA) adalah sebagai berikut:
a) Inisialisasi parameter *Firefly Algorithm* (FA).
b) Membangkitkan secara random populasi awal.
c) Menghitung intensitas cahaya tiap *firefly* $I(x)$.
d) Jika $I_j(x) > I_i(x)$, maka perbarui pergerakan *firefly*.
e) Menentukan $gbest$ dan membandingkan *firefly* terbaik dengan $gbest$.
f) Berhenti jika iterasi maksimal telah terpenuhi.
g) Mengurutkan dan mencari solusi terbaik.

3) Langkah-langkah dari algoritma *Cuckoo Search* (CS) adalah sebagai berikut:
a) Inisialisasi parameter algoritma *Cuckoo Search* (CS).
b) Membangkitkan *cuckoo* (i) dengan *Lévy Flights* dan evaluasi nilai *fitness* F_i.
c) Memilih salah satu sarang sebagai *cuckoo* (j) dengan nilai *fitness* F_j.
d) Jika $F_i > F_j$, maka gantikan *cuckoo* (j)dengan *cuckoo* (i).
e) Menentukan sarang dengan kondisi buruk (P_α)dan membuat sarang baru melalui *Lévy Flights*.
f) Berhenti jika iterasi maksimal telah terpenuhi.
g) Mengurutkan dan mencari solusi terbaik.

d. Membuat Program

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

e. Simulasi dan Implementasi dari Pendekatan Algoritma Metaheuristik

Penyelesaian suatu sistem persamaan non-linier dengan menerapkan algoritma metaheuristik yang meliputi algoritma *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA), dan *Cuckoo Search* (CS) kedalam bentuk program dan membandingkannya, sehingga diperoleh suatu solusi penyelesaian yang dianggap terbaik. Langkah-langkah yang diperlukan adalah sebagai berikut:

1) *Input*

Peneliti menggunakan *input* berupa sistem persamaan non-linier yang akan dicari penyelesaiannya, parameter-parameter yang akan digunakan disesuaikan dengan algoritma yang akan diterapkan.

2) *Output*

Output yang dihasilkan dalam penelitian ini yaitu solusi sistem persamaan non-linier yang memuat akar kompleks. Pemaparan *output* berupa nilai fungsi terbaik dan grafik kekonvergenan. Nilai fungsi dirumuskan dengan $f(x) = |f_1(x_1, x_2, \dots, x_n)| + |f_2(x_1, x_2, \dots, x_n)| + \cdots + |f_n(x_1, x_2, \dots, x_n)|$, dimana nilai fungsi $f(x)$ diperoleh dengan mensubsitusikan setiap variabel pada individu ke fungsi dalam bentuk $z = x + iy$. Nilai x adalah elemen bilangan riil dan y adalah elemen bilangan imajiner. Sedangkan nilai modulus z diperoleh dari rumusan $|z| = \sqrt{x^2 + y^2}$.

f. Analisis Hasil

Peneliti akan melakukan analisis terhadap *output* yang dihasilkan oleh program. Pendekatan algoritma metaheuristik yang dilakukan dikatakan baik, jika *output* yang dihasilkan berupa nilai fungsi yang mendekati nol.

g. Kesimpulan

Hasil dari ketiga algoritma tersebut akan ditarik kesimpulan untuk menentukan solusi yang lebih baik.

Gambar 3.1 Skema Metode Penelitian

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada bab sebelumnya, maka didapatkan kesimpulan sebagai berikut:

a. Penerapan algoritma *Particle Swarm Optimization*, *Firefly Algorithm* dan *Cuckoo Search* dalam menyelesaikan lima SPNL yang meliputi fungsi polinomial, fungsi logaritma, fungsi trigonometri derajat satu dan fungsi ekponensial dapat dilihat pada Tabel 4.1.
b. Perbandingan hasil akurasi pada algoritma *Particle Swarm Optimization*, *Firefly Algorithm* dan *Cuckoo Search* dalam menyelesaikan lima SPNL dapat dilihat dari perbedaan nilai fungsi yang diperoleh. Algoritma *Particle Swarm Optimization* dianggap memiliki hasil akurasi yang paling baik dibandingkan dengan algoritma *Firefly Algorithm* dan *Cuckoo Search* karena nilai fungsinya yang semakin mendekati nol yang ditunjukkan pada Tabel 4.1 dan Tabel 4.2.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, disarankan kepada peneliti selanjutnya untuk menerapkan algoritma *Particle Swarm Optimization* untuk dibandingkan dengan algoritma lainya ataupun melakukan modifikasi dan mengaplikasikannya pada persamaan non-linier ataupun sistem persamaan non-linier yang memuat akar kompleks.

DAFTAR PUSTAKA

Ariyaratne, M. K. A., T. G. I. Fernando, dan S. Weerakoon. 2015. A modified firefly algorithm to solve univariate nonlinear equations with complex roots. *International Conference on Advances in ICT for Emerging Regions*: 160-167.

Arti, R. A. J. 2017. Penerapan Algoritma Cuckoo Search Dalam Permasalahan Penjadwalan Flowshop. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Azmi, A. U. 2018. Perbandingan Algoritma Particle Swarm Optimization (PSO) dan Algoritma Glowworm Swarm Optimization (GSO) Dalam Penyelesaian Sistem Persamaan Non Linier. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Baihaki, N. A. 2016. Penerapan Algoritma Cat Swarm Optimization (CSO) Pada Penyelesaian Sistem Persamaan Non-linier. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Basuki, A. dan N. Ramadijanti. 2004. *Metode Numerik Sebagai Algoritma Komputasi*. Surabaya: PENS-ITS.

Chen, Z., X. Qiu, S. Lin, dan B. Chen. 2017. The iterative methods with higher order convergence for solving a system of nonlinear equations. *Journal of Nonlinear Sciences and Applications*: 3834-3842.

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

Dhanasaputra, N., dan B. Santosa. 2010. Pengembangan algoritma cat swarm optimization (cso) untuk klasifikasi. *Jurnal ITS*: 3.

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

Farikha, E. F. 2018. Penerapan Cockroach Swarm Optimization Algorithm (CSOA) Pada Penyelesaian Persamaan Polinomial Yang Memuat Akar Kompleks. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Graiioo, M. 2011. Solving sistem of nonlinear equations with an improved particle swarm optimization. *International Journal of Conference on Computer Science and Information Technology*: 578-582.

Hiller, F. S., dan G. J. Lieberman. 2010. *Introduction to Operation Research*. New York: McGraw Hill International.

Humam, N. 2015. Bilangan Kompleks. *Makalah Kuliah Umum*. Depok: Universitas Gunadarma Sistem Komputer. 8 Juni.

Madi, M., D. Markovi, dan M. Radovanovi. 2013. Comparison of meta-heuristic algorithms for solving machining optimization problem. *Mechanical Engineering*. 11(1): 29-44.

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 decompotition method for solving system of nonliniear equations. *Journal of Mathematical and Computational Applications*. 15(1): 89-95.

Payne, R. B., M. D. Sorenson, dan K. Klitz. 2005. *The Cuckoos*. England: Oxford University Press.

Prastowo, F. K. 2016. Penerapan Cockroach Swarm Optimization Algorithm (CSOA) Pada Penyelesaian Sistem Persamaan Nonlinier. *Skripsi*. Jember: Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Rosita, A., Y. Purwananto, dan R. Soelaiman. 2012. Implementasi algoritma particle swarm untuk menyelesaikan sistem persamaan nonlinear. *Jurnal Teknik ITS*. 1: 2301-9271.

Sardi, H. 2014. *Modul 1 Sistem Bilangan Kompleks*. Jakarta: Universitas Terbuka.

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., I. N. Widana, dan N. M. Asih. 2013. Perbandingan solusi sistem persamaan nonlinear menggunakan metode newton-raphson dan metode jacobian. *E-Jurnal Matematika*. 2(2): 11-17.

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

Yang, X. S., dan S. Deb. 2009. Cuckoo search via lévy flights. *Proc of World Congress on Nature and Biologically Inspired Computing*: 210-214.

Yang, X. S., dan S. Deb. 2013. Multiobjective cuckoo search for design optimization. *Computers and Operations Research*. 40: 1616-1624.

Yang, X. S. 2010. *Nature-Inspire Metaheuristic Algorithm*. Edisi 2. United Kingdom: Luniver Press.

LAMPIRAN

Lampiran 1.

```
% Update handles structure
guidata(hObject, handles);
clc;
movegui(gcf,'center');
set(handles.popupmenu1,'value',1);
set(handles.text3,'string','(x, y)');
set(handles.edit1,'string','','style','edit');
set(handles.edit2,'string','','style','edit');
set(handles.edit3,'string','','style','text');
set(handles.edit4,'string','','style','text');
set(handles.edit5,'string','');
set(handles.edit6,'string','');
set(handles.edit7,'string','');
set(handles.edit8,'string','');
set(handles.edit9,'string','');
set(handles.edit10,'string','');
set(handles.edit11,'string','');
set(handles.edit12,'string','');
set(handles.edit13,'string','');
set(handles.edit14,'string','');
set(handles.edit15,'string','');
set(handles.edit16,'string','');
set(handles.edit17,'string','');
set(handles.edit18,'string','');
set(handles.edit19,'string','');
set(handles.edit20,'string','');
set(handles.edit21,'string','');
set(handles.edit22,'string','');
cla(handles.axes1,'reset');
axes(handles.axes1);
set(handles.axes1,'XLim',[0 1000],'YLim',[-0.2 1]);
grid on
xlabel('Iterasi');ylabel('Nilai Fungsi');
set(handles.listbox1,'string',sprintf('%5s  %20s  %40s  %40s  %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
set(handles.listbox1,'userdata',[]);
set(handles.listbox2,'userdata',[]);
set(handles.listbox3,'userdata',[]);
```

```
set(handles.text27,'string','0 detik');
set(handles.text29,'string','0');
set(handles.text31,'string','0 detik');
set(handles.text33,'string','0');
set(handles.text35,'string','0 detik');
set(handles.text37,'string','0');
```

Seleksi Banyak Variabel

```
set(handles.edit1,'string','');
set(handles.edit2,'string','');
set(handles.edit3,'string','');
set(handles.edit4,'string','');
num_var=get(handles.popupmenu1,'value')+1;
if num_var==2
    set(handles.text3,'string','(x, y)');
    set(handles.edit1,'style','edit');
    set(handles.edit2,'style','edit');
    set(handles.edit3,'style','text');
    set(handles.edit4,'style','text');
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
elseif num_var==3
    set(handles.text3,'string','(x, y, z)');
    set(handles.edit1,'style','edit');
    set(handles.edit2,'style','edit');
    set(handles.edit3,'style','edit');
    set(handles.edit4,'style','text');
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
elseif num_var==4
    set(handles.text3,'string','(x1, x2, x3, x4)');
```

```
    set(handles.edit1,'style','edit');
    set(handles.edit2,'style','edit');
    set(handles.edit3,'style','edit');
    set(handles.edit4,'style','edit');
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
end
cla(handles.axes1,'reset');
axes(handles.axes1);
set(handles.axes1,'XLim',[0 1000],'YLim',[-0.2 1]);
grid on
xlabel('Iterasi');ylabel('Nilai Fungsi');
set(handles.listbox1,'userdata',[]);
set(handles.listbox2,'userdata',[]);
set(handles.listbox3,'userdata',[]);
set(handles.text27,'string','0 detik');
set(handles.text29,'string','0');
set(handles.text31,'string','0 detik');
set(handles.text33,'string','0');
set(handles.text35,'string','0 detik');
set(handles.text37,'string','0');
```

Tombol Proses

```
clc;
set(handles.listbox1,'userdata',[]);
set(handles.listbox2,'userdata',[]);
set(handles.listbox3,'userdata',[]);
set(handles.text27,'string','0 detik');
set(handles.text29,'string','0');
set(handles.text31,'string','0 detik');
set(handles.text33,'string','0');
set(handles.text35,'string','0 detik');
set(handles.text37,'string','0');
pause(0.1);
format long;
tic;
num_var=get(handles.popupmenu1,'value')+1;

if num_var==2
    f1=inline(get(handles.edit1,'string'),'x','y'); %input fungsi
```

```
    f2=inline(get(handles.edit2,'string'),'x','y'); %input fungsi
elseif num_var==3
    f1=inline(get(handles.edit1,'string'),'x','y','z');      %input
fungsi
    f2=inline(get(handles.edit2,'string'),'x','y','z');      %input
fungsi
    f3=inline(get(handles.edit3,'string'),'x','y','z');      %input
fungsi
elseif num_var==4
    f1=inline(get(handles.edit1,'string'),'x1','x2','x3','x4');
%input fungsi
    f2=inline(get(handles.edit2,'string'),'x1','x2','x3','x4');
%input fungsi
    f3=inline(get(handles.edit3,'string'),'x1','x2','x3','x4');
%input fungsi
    f4=inline(get(handles.edit4,'string'),'x1','x2','x3','x4');
%input fungsi
end

%Parameter
Pop=str2num(get(handles.edit5,'string'));
Lb=str2num(get(handles.edit6,'string'));
Ub=str2num(get(handles.edit7,'string'));
Max_iter=str2num(get(handles.edit8,'string'));
%PSO
V0=str2num(get(handles.edit9,'string'));
c1=str2num(get(handles.edit10,'string'));
c2=str2num(get(handles.edit11,'string'));
Thetamax=str2num(get(handles.edit12,'string'));
Thetamin=str2num(get(handles.edit13,'string'));
%FA
gamma=str2num(get(handles.edit14,'string'));
beta0=str2num(get(handles.edit15,'string'));
alpha=str2num(get(handles.edit16,'string'));
%CS
stepsize=str2num(get(handles.edit17,'string'));
Pa=str2num(get(handles.edit18,'string'));
alphaLF=str2num(get(handles.edit19,'string'));
betaLF=str2num(get(handles.edit20,'string'));
gammaLF=str2num(get(handles.edit21,'string'));
deltaLF=str2num(get(handles.edit22,'string'));

tic;
%Pembangkitan populasi awal
for i=1:Pop
    Populasi(i,:)=rand(1,2*num_var)*(Ub-Lb)+Lb;
    %Evaluasi
    if num_var==2

Fungsi(i)=abs(f1(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Popu
lasi(i,4)*1i))+...

abs(f2(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i));
    elseif num_var==3
```

```
Fungsi(i)=abs(f1(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Popu
lasi(i,4)*1i,Populasi(i,5)+Populasi(i,6)*1i))+...

abs(f2(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i,Populasi(i,5)+Populasi(i,6)*1i))+...

abs(f3(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i,Populasi(i,5)+Populasi(i,6)*1i));
    elseif num_var==4

Fungsi(i)=abs(f1(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Popu
lasi(i,4)*1i,Populasi(i,5)+Populasi(i,6)*1i,Populasi(i,7)+Populasi
(i,8)*1i))+...

abs(f2(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i,Populasi(i,5)+Populasi(i,6)*1i,Populasi(i,7)+Populasi(i,8)*1i))
+...

abs(f3(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i,Populasi(i,5)+Populasi(i,6)*1i,Populasi(i,7)+Populasi(i,8)*1i))
+...

abs(f4(Populasi(i,1)+Populasi(i,2)*1i,Populasi(i,3)+Populasi(i,4)*
1i,Populasi(i,5)+Populasi(i,6)*1i,Populasi(i,7)+Populasi(i,8)*1i))
;
    end
    Fitness(i)=1/(Fungsi(i)+1);
end
waktu=toc;
waktuPSO=waktu;
waktuFA=waktu;
waktuCS=waktu;


%PSO
tic;
PSO=Populasi;
FungsiPSO=Fungsi;
FitnessPSO=Fitness;

%Kecepatan Awal
V=ones(Pop,2*num_var)*V0;

Omega=Thetamax;

%Pbest & Gbest
Pbest=PSO;
FungsiPbest=FungsiPSO;
FitnessPbest=FitnessPSO;
best=find(FitnessPbest==max(FitnessPbest));
Gbest=Pbest(best(1),:);
FungsiGbest=FungsiPbest(best(1));
FitnessGbest=FitnessPbest(best(1));
```

```
KonvergenPSO(1)=FungsiGbest;
ikonPSO=0;
waktuPSO=waktuPSO+toc;

%FA
tic;
FA=Populasi;
FungsiFA=Fungsi;
FitnessFA=Fitness;
beta=ones(1,Pop)*beta0;
best=find(FitnessFA==max(FitnessFA));
FAbest=FA(best(1),:);
FungsiFAbest=FungsiFA(best(1));
FitnessFAbest=FitnessFA(best(1));
KonvergenFA(1)=FungsiFAbest;
ikonFA=0;
waktuFA=waktuFA+toc;

%CS
tic;
CS=Populasi;
FungsiCS=Fungsi;
FitnessCS=Fitness;
best=find(FitnessCS==max(FitnessCS));
bestsf=CS(best(1),:);
Fungsibestsf=FungsiCS(best(1));
Fitnessbestsf=FitnessCS(best(1));
KonvergenCS(1)=Fungsibestsf;
ikonCS=0;
waktuCS=waktuCS+toc;

solusi1=Gbest;
solusi2=FAbest;
solusi3=bestsf;
if num_var==2
    prnt1={sprintf('%5s %20s %40s %40s %40s %42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai Fungsi');
        sprintf('%5d     %20.16f     %24.16f     %24.16f     %24.16f
%28.16e',0,solusi1(1),solusi1(2),solusi1(3),solusi1(4),KonvergenPS
O(1))};
    prnt2={sprintf('%5s %20s %40s %40s %40s %42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai Fungsi');
        sprintf('%5d     %20.16f     %24.16f     %24.16f     %24.16f
%28.16e',0,solusi2(1),solusi2(2),solusi2(3),solusi2(4),KonvergenFA
(1))};
    prnt3={sprintf('%5s %20s %40s %40s %40s %42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai Fungsi');
        sprintf('%5d     %20.16f     %24.16f     %24.16f     %24.16f
%28.16e',0,solusi3(1),solusi3(2),solusi3(3),solusi3(4),KonvergenCS
(1))};
elseif num_var==3
    prnt1={sprintf('%5s %20s %40s %40s %40s %40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi');
```

```
        sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %28.16e',...

0,solusi1(1),solusi1(2),solusi1(3),solusi1(4),solusi1(5),solusi1(6
),KonvergenPSO(1))};
    prnt2={sprintf('%5s %20s %40s %40s %40s %40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi');
        sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %28.16e',...

0,solusi2(1),solusi2(2),solusi2(3),solusi2(4),solusi2(5),solusi2(6
),KonvergenFA(1))};
    prnt3={sprintf('%5s %20s %40s %40s %40s %40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi');
         sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %28.16e',...

0,solusi3(1),solusi3(2),solusi3(3),solusi3(4),solusi3(5),solusi3(6
),KonvergenCS(1))};
elseif num_var==4
    prnt1={sprintf('%5s  %20s  %40s  %40s  %40s  %40s  %40s  %40s  %40s
%42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi');
        sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %24.16f %24.16f %28.16e',...

0,solusi1(1),solusi1(2),solusi1(3),solusi1(4),solusi1(5),solusi1(6
),solusi1(7),solusi1(8),KonvergenPSO(1))};
    prnt2={sprintf('%5s  %20s  %40s  %40s  %40s  %40s  %40s  %40s  %40s
%42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi');
        sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %24.16f %24.16f %28.16e',...

0,solusi2(1),solusi2(2),solusi2(3),solusi2(4),solusi2(5),solusi2(6
),solusi2(7),solusi2(8),KonvergenFA(1))};
    prnt3={sprintf('%5s  %20s  %40s  %40s  %40s  %40s  %40s  %40s  %40s
%42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi');
        sprintf('%5d  %20.16f  %24.16f  %24.16f  %24.16f  %24.16f
%24.16f %24.16f %24.16f %28.16e',...

0,solusi3(1),solusi3(2),solusi3(3),solusi3(4),solusi3(5),solusi3(6
),solusi3(7),solusi3(8),KonvergenCS(1))};
end
set(handles.listbox1,'string',char(prnt1),'value',2);
```

```
set(handles.listbox2,'string',char(prnt2),'value',2);
set(handles.listbox3,'string',char(prnt3),'value',2);

%Iterasi
for iter=1:Max_iter
    %PSO
    tic;
    for i=1:Pop
        V(i,:)=V(i,:)*Omega+c1*rand(1,2*num_var).*(Pbest(i,:)-
PSO(i,:))+c2*rand(1,2*num_var).*(Gbest-PSO(i,:));          %update
kecepatan
        PSO(i,:)=PSO(i,:)+V(i,:); %update posisi
        %Evaluasi
        if num_var==2

FungsiPSO(i)=abs(f1(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i))+..
.

abs(f2(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i));
        elseif num_var==3

FungsiPSO(i)=abs(f1(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(
i,5)+PSO(i,6)*1i))+...

abs(f2(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(i,5)+PSO(i,6)
*1i))+...

abs(f3(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(i,5)+PSO(i,6)
*1i));
        elseif num_var==4

FungsiPSO(i)=abs(f1(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(
i,5)+PSO(i,6)*1i,PSO(i,7)+PSO(i,8)*1i))+...

abs(f2(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(i,5)+PSO(i,6)
*1i,PSO(i,7)+PSO(i,8)*1i))+...

abs(f3(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(i,5)+PSO(i,6)
*1i,PSO(i,7)+PSO(i,8)*1i))+...

abs(f4(PSO(i,1)+PSO(i,2)*1i,PSO(i,3)+PSO(i,4)*1i,PSO(i,5)+PSO(i,6)
*1i,PSO(i,7)+PSO(i,8)*1i));
        end
        FitnessPSO(i)=1/(FungsiPSO(i)+1);
        %update Pbest
        if FitnessPSO(i)>FitnessPbest(i)
            Pbest(i,:)=PSO(i,:);
            FungsiPbest(i)=FungsiPSO(i);
            FitnessPbest(i)=FitnessPSO(i);
        end
    end
    %update Gbest
    best=find(FitnessPbest==max(FitnessPbest));
    Gbest=Pbest(best(1),:);
    FungsiGbest=FungsiPbest(best(1));
    FitnessGbest=FitnessPbest(best(1));
```

```
    %update Omega
    Omega=Omega-(Thetamax-Thetamin)/Max_iter;

    KonvergenPSO(iter+1)=FungsiGbest;
    if KonvergenPSO(iter+1)~=KonvergenPSO(iter)
        ikonPSO=iter;
    end
    waktuPSO=waktuPSO+toc;

    %FA
    tic;
    for i=1:Pop
        for j=1:Pop
            if FitnessFA(j)>FitnessFA(i)
                FA(i,:)=FA(i,:)+beta(i)*(FA(j,:)-
FA(i,:))+alpha*(rand(1,2*num_var)-0.5);
                beta(i)=beta(i)*exp(-gamma*(sum((FA(j,:)-
FA(i,:)).^2)));
                %Evaluasi
                if num_var==2

FungsiFA(i)=abs(f1(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i))+...

abs(f2(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i));
                elseif num_var==3

FungsiFA(i)=abs(f1(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+F
A(i,6)*1i))+...

abs(f2(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+FA(i,6)*1i))+
...

abs(f3(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+FA(i,6)*1i));
                elseif num_var==4

FungsiFA(i)=abs(f1(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+F
A(i,6)*1i,FA(i,7)+FA(i,8)*1i))+...

abs(f2(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+FA(i,6)*1i,FA
(i,7)+FA(i,8)*1i))+...

abs(f3(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+FA(i,6)*1i,FA
(i,7)+FA(i,8)*1i))+...

abs(f4(FA(i,1)+FA(i,2)*1i,FA(i,3)+FA(i,4)*1i,FA(i,5)+FA(i,6)*1i,FA
(i,7)+FA(i,8)*1i));
                end
                FitnessFA(i)=1/(FungsiFA(i)+1);
            end
        end
    end
    best=find(FitnessFA==max(FitnessFA));
    FAbest=FA(best(1),:);
    FungsiFAbest=FungsiFA(best(1));
    FitnessFAbest=FitnessFA(best(1));
```

```
    KonvergenFA(iter+1)=FungsiFAbest;
    if KonvergenFA(iter+1)~=KonvergenFA(iter)
        ikonFA=iter;
    end
    waktuFA=waktuFA+toc;

    %CS
    tic;

Cuckoo=bestsf+stepsize*levystblrnd(alphaLF,betaLF,gammaLF,deltaLF,
[1,2*num_var]);
    %Evaluasi
    if num_var==2

FungsiCuckoo=abs(f1(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i)
)+...

abs(f2(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i));
    elseif num_var==3

FungsiCuckoo=abs(f1(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,
Cuckoo(5)+Cuckoo(6)*1i))+...

abs(f2(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,Cuckoo(5)+Cuc
koo(6)*1i))+...

abs(f3(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,Cuckoo(5)+Cuc
koo(6)*1i));
    elseif num_var==4

FungsiCuckoo=abs(f1(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,
Cuckoo(5)+Cuckoo(6)*1i,Cuckoo(7)+Cuckoo(8)*1i))+...

abs(f2(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,Cuckoo(5)+Cuc
koo(6)*1i,Cuckoo(7)+Cuckoo(8)*1i))+...

abs(f3(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,Cuckoo(5)+Cuc
koo(6)*1i,Cuckoo(7)+Cuckoo(8)*1i))+...

abs(f4(Cuckoo(1)+Cuckoo(2)*1i,Cuckoo(3)+Cuckoo(4)*1i,Cuckoo(5)+Cuc
koo(6)*1i,Cuckoo(7)+Cuckoo(8)*1i));
    end
    FitnessCuckoo=1/(FungsiCuckoo+1);
    j=ceil(rand*Pop);
    if FitnessCuckoo>FitnessCS(j)
        CS(j,:)=Cuckoo;
        FungsiCS(j)=FungsiCuckoo;
        FitnessCS(j)=FitnessCuckoo;
    end
    [FitnessCSurut,indek]=sort(FitnessCS,'descend');
    CS=CS(indek,:);
    FungsiCS=FungsiCS(indek);
    FitnessCS=FitnessCS(indek);
    Worst=fix(Pa*Pop);
    for i=1:Worst
```

```
        CS(Pop-i+1,:)=CS(Pop-
i+1,:)+stepsize*levystblrnd(alphaLF,betaLF,gammaLF,deltaLF,[1,2*nu
m_var]);
        %Evaluasi
        if num_var==2
            FungsiCS(Pop-i+1)=abs(f1(CS(Pop-i+1,1)+CS(Pop-
i+1,2)*1i,CS(Pop-i+1,3)+CS(Pop-i+1,4)*1i))+...
                abs(f2(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i));
        elseif num_var==3
            FungsiCS(Pop-i+1)=abs(f1(CS(Pop-i+1,1)+CS(Pop-
i+1,2)*1i,CS(Pop-i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-
i+1,6)*1i))+...
                abs(f2(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-i+1,6)*1i))+...
                abs(f3(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-i+1,6)*1i));
        elseif num_var==4
            FungsiCS(Pop-i+1)=abs(f1(CS(Pop-i+1,1)+CS(Pop-
i+1,2)*1i,CS(Pop-i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-
i+1,6)*1i,CS(Pop-i+1,7)+CS(Pop-i+1,8)*1i))+...
                abs(f2(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-i+1,6)*1i,CS(Pop-
i+1,7)+CS(Pop-i+1,8)*1i))+...
                abs(f3(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-i+1,6)*1i,CS(Pop-
i+1,7)+CS(Pop-i+1,8)*1i))+...
                abs(f4(CS(Pop-i+1,1)+CS(Pop-i+1,2)*1i,CS(Pop-
i+1,3)+CS(Pop-i+1,4)*1i,CS(Pop-i+1,5)+CS(Pop-i+1,6)*1i,CS(Pop-
i+1,7)+CS(Pop-i+1,8)*1i));
        end
        FitnessCS(Pop-i+1)=1/(FungsiCS(Pop-i+1)+1);
    end
    if max(FitnessCS)>Fitnessbestsf
        best=find(FitnessCS==max(FitnessCS));
        bestsf=CS(best(1),:);
        Fungsibestsf=FungsiCS(best(1));
        Fitnessbestsf=FitnessCS(best(1));
    end

    KonvergenCS(iter+1)=Fungsibestsf;
    if KonvergenCS(iter+1)~=KonvergenCS(iter)
        ikonCS=iter;
    end
    waktuCS=waktuCS+toc;

    %plot kekonvergenan
    axes(handles.axes1);
    plot(0:iter,KonvergenPSO,'r','LineWidth',2);
    hold on
    plot(0:iter,KonvergenFA,'g--','LineWidth',2);
    plot(0:iter,KonvergenCS,'b:','LineWidth',2);

line(ikonPSO,KonvergenPSO(ikonPSO+1),'Marker','s','Markersize',8,'
Markerfacecolor','r');
```

```
line(ikonFA,KonvergenFA(ikonFA+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','g');

line(ikonCS,KonvergenCS(ikonCS+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','b');
    hold off
    grid on
    set(handles.axes1,'Ylim',[-KonvergenPSO(1)/8
KonvergenPSO(1)]);
    %
    solusi1=Gbest;
    solusi2=FAbest;
    solusi3=bestsf;
    if num_var==2
        prnt1={get(handles.listbox1,'string');
            sprintf('%5d    %20.16f    %24.16f    %24.16f    %24.16f
%28.16e',iter,solusi1(1),solusi1(2),solusi1(3),solusi1(4),Konverge
nPSO(iter+1))};
        prnt2={get(handles.listbox2,'string');
            sprintf('%5d    %20.16f    %24.16f    %24.16f    %24.16f
%28.16e',iter,solusi2(1),solusi2(2),solusi2(3),solusi2(4),Konverge
nFA(iter+1))};
        prnt3={get(handles.listbox3,'string');
            sprintf('%5d    %20.16f    %24.16f    %24.16f    %24.16f
%28.16e',iter,solusi3(1),solusi3(2),solusi3(3),solusi3(4),Konverge
nCS(iter+1))};
    elseif num_var==3
        prnt1={get(handles.listbox1,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %28.16e',...

iter,solusi1(1),solusi1(2),solusi1(3),solusi1(4),solusi1(5),solusi
1(6),KonvergenPSO(iter+1))};
        prnt2={get(handles.listbox2,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %28.16e',...

iter,solusi2(1),solusi2(2),solusi2(3),solusi2(4),solusi2(5),solusi
2(6),KonvergenFA(iter+1))};
        prnt3={get(handles.listbox3,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %28.16e',...

iter,solusi3(1),solusi3(2),solusi3(3),solusi3(4),solusi3(5),solusi
3(6),KonvergenCS(iter+1))};
    elseif num_var==4
        prnt1={get(handles.listbox1,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %24.16f %24.16f %28.16e',...

iter,solusi1(1),solusi1(2),solusi1(3),solusi1(4),solusi1(5),solusi
1(6),solusi1(7),solusi1(8),KonvergenPSO(iter+1))};
        prnt2={get(handles.listbox2,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %24.16f %24.16f %28.16e',...
```

```
iter,solusi2(1),solusi2(2),solusi2(3),solusi2(4),solusi2(5),solusi
2(6),solusi2(7),solusi2(8),KonvergenFA(iter+1))};
        prnt3={get(handles.listbox3,'string');
            sprintf('%5d %20.16f %24.16f %24.16f %24.16f %24.16f
%24.16f %24.16f %24.16f %28.16e',...

iter,solusi3(1),solusi3(2),solusi3(3),solusi3(4),solusi3(5),solusi
3(6),solusi3(7),solusi3(8),KonvergenCS(iter+1))};
    end
    set(handles.listbox1,'string',char(prnt1),'value',iter+2);
    set(handles.listbox2,'string',char(prnt2),'value',iter+2);
    set(handles.listbox3,'string',char(prnt3),'value',iter+2);
    pause(0.01);
end
axes(handles.axes1);
plot(0:iter,KonvergenPSO,'r','LineWidth',2);
hold on
plot(0:iter,KonvergenFA,'g--','LineWidth',2);
plot(0:iter,KonvergenCS,'b:','LineWidth',2);
legend('PSO','FA','CS');
line(ikonPSO,KonvergenPSO(ikonPSO+1),'Marker','s','Markersize',8,'
Markerfacecolor','r');
line(ikonFA,KonvergenFA(ikonFA+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','g');
line(ikonCS,KonvergenCS(ikonCS+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','b');
hold off
grid on
set(handles.axes1,'Ylim',[-KonvergenPSO(1)/8 KonvergenPSO(1)]);
set(handles.listbox1,'userdata',{ikonPSO,KonvergenPSO});
set(handles.listbox2,'userdata',{ikonFA,KonvergenFA});
set(handles.listbox3,'userdata',{ikonCS,KonvergenCS});
set(handles.text27,'string',[num2str(waktuPSO) ' detik']);
set(handles.text29,'string',num2str(ikonPSO));
set(handles.text31,'string',[num2str(waktuFA) ' detik']);
set(handles.text33,'string',num2str(ikonFA));
set(handles.text35,'string',[num2str(waktuCS) ' detik']);
set(handles.text37,'string',num2str(ikonCS));
```

Tombol Reset

```
clc;
movegui(gcf,'center');
set(handles.edit1,'string','','style','edit');
set(handles.edit2,'string','','style','edit');
set(handles.edit3,'string','','style','text');
set(handles.edit4,'string','','style','text');
set(handles.edit5,'string','');
set(handles.edit6,'string','');
set(handles.edit7,'string','');
set(handles.edit8,'string','');
set(handles.edit9,'string','');
set(handles.edit10,'string','');
```

```
set(handles.edit11,'string','');
set(handles.edit12,'string','');
set(handles.edit13,'string','');
set(handles.edit14,'string','');
set(handles.edit15,'string','');
set(handles.edit16,'string','');
set(handles.edit17,'string','');
set(handles.edit18,'string','');
set(handles.edit19,'string','');
set(handles.edit20,'string','');
set(handles.edit21,'string','');
set(handles.edit22,'string','');
cla(handles.axes1,'reset');
axes(handles.axes1);
set(handles.axes1,'XLim',[0 1000],'YLim',[-0.2 1]);
grid on
xlabel('Iterasi');ylabel('Nilai Fungsi');
num_var=get(handles.popupmenu1,'value')+1;
if num_var==2
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%42s',...
        'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Nilai
Fungsi'),'value',1);
elseif num_var==3
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %42s',...

'Iter','Re(x)','Im(x)','Re(y)','Im(y)','Re(z)','Im(z)','Nilai
Fungsi'),'value',1);
elseif num_var==4
    set(handles.listbox1,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
    set(handles.listbox2,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...
```

```
'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
    set(handles.listbox3,'string',sprintf('%5s %20s %40s %40s %40s
%40s %40s %40s %40s %42s',...

'Iter','Re(x1)','Im(x1)','Re(x2)','Im(x2)','Re(x3)','Im(x3)','Re(x
4)','Im(x4)','Nilai Fungsi'),'value',1);
end
set(handles.listbox1,'userdata',[]);
set(handles.listbox2,'userdata',[]);
set(handles.listbox3,'userdata',[]);
set(handles.text27,'string','0 detik');
set(handles.text29,'string','0');
set(handles.text31,'string','0 detik');
set(handles.text33,'string','0');
set(handles.text35,'string','0 detik');
set(handles.text37,'string','0');
```

Tombol Simpan Grafik

```
data1=get(handles.listbox1,'userdata');
data2=get(handles.listbox2,'userdata');
data3=get(handles.listbox3,'userdata');
if ~isempty(data1) && ~isempty(data2) && ~isempty(data3)
    [FileName,FilePath] = uiputfile('*.jpg','Save Plot As');
    if FileName~=0
        ikonPSO=data1{1};
        KonvergenPSO=data1{2};
        ikonFA=data2{1};
        KonvergenFA=data2{2};
        ikonCS=data3{1};
        KonvergenCS=data3{2};
        figure
        plot(0:length(KonvergenPSO)-
1,KonvergenPSO,'r','LineWidth',2);
        hold on
        plot(0:length(KonvergenFA)-1,KonvergenFA,'g--
','LineWidth',2);
        plot(0:length(KonvergenCS)-
1,KonvergenCS,'b:','LineWidth',2);
        legend('PSO','FA','CS');

line(ikonPSO,KonvergenPSO(ikonPSO+1),'Marker','s','Markersize',8,'
Markerfacecolor','r');

line(ikonFA,KonvergenFA(ikonFA+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','g');

line(ikonCS,KonvergenCS(ikonCS+1),'Marker','s','Markersize',8,'Mar
kerfacecolor','b');
        hold off
        grid on
        xlabel('Iterasi');ylabel('Nilai Fungsi');
```

```
        ylim([-KonvergenPSO(1)/8 KonvergenPSO(1)]);
        saveas(gcf,fullfile(FilePath,FileName));
        close(gcf);
    end
end
```

Lampiran 2.