PEMBANGKITAN POHON FRAKTAL TIGA CABANG DENGAN METODE *ITERATED FUNCTION SYSTEM*

SKRIPSI

Oleh

Dita Wahyuningtyas
NIM 161810101025

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

PEMBANGKITAN POHON FRAKTAL TIGA CABANG DENGAN METODE *ITERATED FUNCTION SYSTEM*

SKRIPSI

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

Oleh

Dita Wahyuningtyas
NIM 161810101025

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. kedua orangtua tercinta yaitu Bapak Tejaning Surya Wahyudi dan Ibu Kusluluk Heruningsih yang telah memberikan dukungan, semangat, kasih sayang, perhatian dan pengorbanan yang begitu besar serta doa yang tak pernah putus untuk anak-anaknya;
2. adik tersayang Lanang Risqi Bimantara serta seluruh keluarga yang telah memberikan doa, semangat dan perhatian kepada saya;
3. seluruh dosen dan guru-guru SMAN 3 Jember, SMPN 1 Jember, SDN Kepatihan 05 Jember dan TK Kartika IV-73 Jember yang telah mendidik dan membimbing dengan penuh kesabaran;
4. Almamater tercinta Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
5. teman-teman maisaroh yaitu Ririn, Rudal, dan Pipit yang selalu memberikan dukungan;
6. teman-teman Misdirection 16 HIMATIKA Geokompstat yang selalu memberikan dukungan, bantuan, dan doa;
7. teman-teman pejuang fraktal yaitu Rana, Hafid, dan Rifkatus yang selalu memberikan semangat dan bantuan;
8. sahabat-sahabat tersayang yaitu Diah, Nana, Catrin, dan Lela yang selalu memberikan semangat dan doa.

MOTO

"Diwajibkan atas kamu berperang, padahal berperang itu adalah sesuatu yang kamu benci. Boleh jadi kamu membenci sesuatu, padahal ia amat baik bagimu, dan boleh jadi (pula) kamu menyukai sesuatu, padahal ia amat buruk bagimu; Allah mengetahui, sedang kamu tidak mengetahui"
(QS. Al-Baqarah:216)[*)]

[*)] Departemen Agama Republik Indonesia. 1998. Al Qur'an dan Terjemahannya. Semarang: PT. Kumudasmoro Grafindo.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Dita Wahyuningtyas

NIM : 161810101025

menyatakan dengan sesungguhnya bahwa skripsi yang berjudul "Pembangkitan Pohon Fraktal Tiga Cabang dengan Metode *Iterated Function System*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun, dan bukan karya jiplakan. Saya bertanggungjawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

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

Jember, Januari 2020

Yang menyatakan,

Dita Wahyuningtyas

NIM 161810101025

SKRIPSI

PEMBANGKITAN POHON FRAKTAL TIGA CABANG DENGAN METODE *ITERATED FUNCTION SYSTEM*

Oleh

Dita Wahyuningtyas
NIM 161810101025

Pembimbing

Dosen Pembimbing Utama : Kosala Dwidja Purnomo, S.Si., M.Si
Dosen Pembimbing Anggota : Dr. Firdaus Ubaidillah, S.Si., M.Si

PENGESAHAN

Skripsi berjudul "Pembangkitan Pohon Fraktal Tiga Cabang dengan Metode *Iterated Function System*" karya Dita Wahyuningtyas telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Anggota I,
Kosala Dwidja Purnomo, S.Si., M.Si. NIP. 196908281998021001	Dr. Firdaus Ubaidillah, S.Si., M.Si. NIP. 197006061998031003
Anggota II,	Anggota III,
Bagus Juliyanto, S.Si., M.Si NIP. 198007022003121001	Kusbudiono, S.Si., M.Si NIP. 197704302005011001

Mengesahkan
Dekan,

Drs. Achmad Sjaifullah, M.Sc., Ph.D.
NIP. 195910091986021001

RINGKASAN

Pembangkitan Pohon Fraktal Tiga Cabang dengan Metode *Iterated Function System*; Dita Wahyuningtyas, 161810101025; 2020; 65 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Fraktal merupakan salah satu cabang ilmu matematika yang dapat dibangkitkan pada komputer untuk melihat keindahan visualnya. Fraktal bersifat *self-similarity* atau keserupaan diri yang artinya objek yang dibangun berulang tetapi memiliki ukuran yang lebih kecil dari objek aslinya, sedangkan sifat lain dari fraktal yaitu *infinite detail* yang artinya apabila objek diperbesar maka detail-detailnya akan terlihat dengan jelas. Fraktal dibagi menjadi dua jenis, yaitu himpunan-himpunan fraktal (*fractal sets*) dan fraktal alami (*natural fractal*). Fraktal alami meliputi pohon, daun, pakis, gunung, garis pantai. Salah satu pohon yang dapat dibangkitkan dan divisualisasikan dalam fraktal adalah pohon Pythagoras. Pohon Pythagoras memiliki dua percabangan selanjutnya dikembangkan menjadi tiga percabangan dan disebut dengan pohon fraktal. Pohon fraktal dibangkitkan melalui transformasi affine dilatasi, translasi dan rotasi menggunakan metode *Iterated Function System* (IFS).

Prosedur untuk membangkitkan pohon fraktal adalah sebagai berikut: pertama, ditentukan titik awal *(x,y)* sebanyak jumlah titik sudut dari persegi pada koordinat kartesius. Kedua, ditentukan bentuk dasar percabangan dengan beberapa pemilihan tinggi dan P. Pemilihan tinggi dan P yang diinputkan secara manual atau tinggi dan P yang diatur secara random dalam program GUI Matlab 2015b. Ketiga, pohon fraktal dibangun menggunakan transformasi affine pada IFS. Cabang pertama di sisi l_1 dilakukan transformasi affine pada persegi awal berupa dilatasi dengan titik pusat dilatasi pada titik *K*, dan rotasi sebesar sudut α, titik pusat rotasi di titik *K* dengan arah perputaran berlawanan arah jarum jam. Cabang kedua di sisi l_2 dilakukan transformasi affine pada persegi awal berupa dilatasi dengan titik pusat dilatasi pada titik *K*, selanjutnya dilakukan translasi,

dan rotasi sebesar sudut γ, titik pusat rotasi berada di titik K persegi baru hasil translasi. Cabang ketiga pada sisi l_3 dilakukan transformasi affine pada persegi awal berupa dilatasi dengan titik pusat dilatasi pada titik L, dan rotasi sebesar sudut β, titik pusat rotasi di titik L dengan arah perputaran searah arah jarum jam.

Masing-masing pemilihan tinggi dan P menghasilkan bentuk pohon fraktal yang berbeda-beda. Pemilihan tinggi dan P yang ditentukan secara manual pada program GUI Matlab 2015b menghasilkan pohon fraktal yang simetris dikarenakan peneliti menginputkan secara manual dengan aturan syarat $0 < P_1 < P_2 < 1$ dan $t_1 \leq \sqrt{0{,}78} \approx 0{,}88$ $t_2 \leq \sqrt{0{,}68} \approx 0{,}82$ hingga menghasilkan pohon fraktal yang simetris. Pemilihan tinggi dan P secara random pada program GUI Matlab 2015b pada setiap *running* menghasilkan beberapa bentuk pohon fraktal yang berbeda-beda. Bentuk pohon fraktal yang lebih condong ke arah kanan, ke kiri, atau simetris dipengaruhi oleh besar nilai tinggi dan P.

PRAKATA

Puji syukur ke hadirat Allah SWT, atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Pembangkitan Pohon Fraktal Tiga Cabang dengan Metode *Iterated Function System*". 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. Kosala Dwidja Purnomo, S.Si., M.Si. selaku Dosen Pembimbing Utama dan Dr. Firdaus Ubaidillah, S.Si., M.Si. selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran dan perhatian dalam penulisan skripsi ini;
2. Bagus Juliyanto, S.Si., M.Si. dan Kusbudiono, S.Si., M.Si. selaku Dosen Penguji yang telah memberikan masukan, saran dan kritik yang membangun dalam penyusunan skripsi ini;
3. seluruh dosen dan karyawan Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam;
4. Derby Fiershanty Ramadhany yang telah memberikan dukungan dan bantuan;
5. Teman-teman KKN 001 Jeruk Soksok yaitu Dio, Puguh, Iqbal, Fenry, Gatot, Hajar, Via, A'yun, dan Hendri yang selalu memberikan dukungan;
6. 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, Januari 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Fraktal merupakan salah satu cabang ilmu matematika yang dapat dibangkitkan pada komputer untuk melihat keindahan visualnya. Fraktal diperkenalkan oleh Benoit Mandelbrot pada tahun 1975 yang kini dikenal sebagai bapak fraktal (Beek, 1996). Fraktal berasal dari bahasa latin *"fractus"* yang artinya patah, *"frangere"* yang artinya rusak, dan *"fragmen"* yang artinya tidak teratur (Mandelbrot, 1983). Fraktal bersifat *self-similarity* atau keserupaan diri yang artinya objek yang dibangun berulang tetapi memiliki ukuran yang lebih kecil dari objek aslinya, sedangkan sifat lain dari fraktal yaitu *infinite detail* yang artinya apabila objek diperbesar maka detail-detailnya akan terlihat dengan jelas.

Fraktal dibagi menjadi dua jenis, yaitu himpunan-himpunan fraktal (*fractal sets*) dan fraktal alami (*natural fractal*). Himpunan-himpunan fraktal meliputi segitiga Sierpinski, *Koch Snowflake*, himpunan Mandelbrot dapat dilihat pada Gambar 1.1.

(a) (b) (c)

(a) Koch Snowflake; (b) segitiga Sierpinski; (c) himpunan Mandelbrot

Gambar 1.1 Jenis-jenis fraktal (Sumber: Hasang dan Supardjo, 2012)

Fraktal alami meliputi pohon, daun, pakis, gunung, garis pantai. Salah satu pohon yang dapat dibangkitkan dan divisualisasikan dalam fraktal adalah pohon Pythagoras. Pohon Pythagoras pertama kali dibangun oleh Albert E. Bosman pada tahun 1942. Nama untuk fraktal ini berasal dari fakta bahwa bentuk penggantian secara tradisional digunakan untuk menggambarkan Teorema Pythagoras (Matam, 2012). Gambar 1.2 merupakan gambar pohon Pythagoras.

Gambar 1.2 pohon Pythagoras (Sumber: Matam, 2012)

Jenis-jenis fraktal yang telah diuraikan di atas dapat dibangkitkan menggunakan beberapa metode untuk menghasilkan bentuk fraktal seperti pada Gambar 1.1 dan 1.2. Beberapa metode tersebut yaitu metode *Iterated Function System* (IFS), metode *Iterated Complex Polynomial*, metode *L-system* dan metode-metode lainnya. IFS diperkenalkan oleh Michael Barnsley sekitar tahun 1985. Ramandhani (2012) dalam makalahnya menyebutkan bahwa fraktal yang rumit dan memiliki detail tidak terbatas dapat dibangkitkan dengan metode IFS. Sundusia (2019) membangkitkan pohon Pythagoras dengan metode IFS melalui transformasi affine.

Ramandhani (2012) menyatakan bahwa pembangkitan karpet Sierpinski dengan metode IFS mengalami sekumpulan transformasi (fungsi) pada gambar yang menjadi bentuk awal, baik itu dengan translasi, rotasi, refleksi, maupun pengubahan skala sehingga menghasilkan gambar baru. Setiap gambar-gambar baru ini akan ditransformasi lagi dengan sekumpulan transformasi baik itu dengan translasi, rotasi, refleksi, maupun pengubahan skala sedemikian sehingga setiap transformasi pada gambar akan membentuk sebuah iterasi. Jika transformasi tersebut bersifat kontraktif, yaitu mengakibatkan titik-titik pada gambar menjadi semakin berdekatan, maka gambar tadi pada akhirnya akan konvergen ke suatu bentuk. Sehingga, setelah terjadi iterasi tak berhingga serta transformasinya adalah transformasi kontraktif, maka gambar akan konvergen ke suatu bentuk. Selain itu, Purnomo (2014) dalam penelitiannya melakukan pembangkitkan segitiga Sierpinski dengan memanfaatkan transformasi affine dalam bentuk dilasi dan translasi pada segitiga sama sisi sebagai bentuk dasar,

modifikasi selanjutnya mengganti bentuk dasar segitiga sama sisi dengan benda geometris segiempat. Prasasti (2018) dalam penelitiannya memanfaatkan metode IFS dalam mengembangkan motif anyaman menggunakan beberapa transformasi affine pada gambar benda geometri persegi panjang dan bujur sangkar sebagai bentuk awal. Dalam penelitian lainnya, Sundusia (2019) membangun berbagai bentuk pohon Pythagoras dengan memanfaatkan metode IFS. Pohon Pythagoras dibangkitkan melalui beberapa transformasi affine diantaranya dilatasi dan rotasi pada gambar benda geometri persegi sebagai bentuk awal. Pohon Pythagoras terinspirasi dari teorema Pythagoras yang berbunyi, kuadrat panjang sisi miring suatu segitiga siku-siku adalah sama dengan jumlah kuadrat panjang sisi-sisi yang lain (Rahayu, 2012). Percabangan pada pohon berbentuk segitiga siku-siku, sehingga disebut pohon Pythagoras.

Berdasarkan penelitian yang telah diuraikan di atas, pada penelitian selanjutnya peneliti tertarik untuk memodifikasi percabangan pada pohon Pythagoras. Pohon Pythagoras yang memiliki dua percabangan dikembangkan menjadi tiga percabangan dan selanjutnya disebut dengan istilah pohon fraktal. Pohon fraktal akan dibangkitkan melalui transformasi affine dilatasi, translasi dan rotasi menggunakan metode IFS.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas adalah bagaimana prosedur dan hasil visual pembangkitkan pohon fraktal dengan metode *Iterated Function System*?

1.3 Tujuan

Berdasarkan rumusan masalah di atas, tujuan dari penelitian ini yaitu mendapatkan prosedur dan hasil pembangkitan pohon fraktal dengan metode *Iterated Function System*.

1.4 Manfaat

Manfaat dari penelitian ini adalah memberikan informasi dan mendapatkan beberapa bentuk pohon fraktal yang didapat dari memodifikasi pohon Pythagoras menggunakan *Iterated Function System*.

BAB 2. TINJAUAN PUSTAKA

2.1 Fraktal

Fraktal adalah kumpulan pola-pola geometris baik yang terdapat di alam maupun yang berupa visualisasi model matematis di mana pola tersebut diulang berkali-kali dengan skala yang semakin kecil (Sampurno dan Faryuni, 2016). Fraktal berasal dari bahasa latin *"fractus"* yang artinya patah, *"frangere"* yang artinya rusak, dan *"fragmen"* yang artinya tidak teratur. Ciri khas fraktal adalah memiliki dimensi dalam bentuk pecahan. Keberadaan fraktal terdapat hampir seluruh sudut alam dan sistem matematis (Mandelbrot, 1983).

Berdasarkan sifatnya, fraktal secara umum dapat dikategorikan menjadi tiga bentuk yaitu fraktal *self-similarity*, fraktal acak (stokastik), dan fraktal *self affine*. Fraktal *self-similarity* adalah fraktal yang potongan kecil bagiannya memiliki bentuk yang sama persis dengan bentuk keseluruhannya. Fraktal acak adalah fraktal yang dibangkitkan oleh generator yang bersifat stokastik, sehingga sifat fraktalnya tidaklah *self-similar* secara eksak namun dalam pengertian statistik tetap *self-similar*. Sedangkan fraktal *self affine* adalah fraktal yang memiliki simetri dilatasi anisotropik, artinya bentuk potongan bagian kecilnya menyerupai namun tidak persis sama dengan bentuknya secara keseluruhan (Sampurno dan Faryuni, 2016).

2.2 *Iterated Function System* (IFS)

IFS mengkontruksi fraktal dengan cara mengulang transformasi berkali-kali untuk sebarang pola awal. Pola awal ditransformasi menjadi suatu pola berulang dengan struktur yang sama pada detail tertentu, hal tersebut merupakan karakteristik dasar dari himpunan fraktal (Utomo, 2011). IFS merupakan suatu fungsi iterasi yang terdiri dari sekumpulan transformasi affine yang digunakan untuk membangun suatu objek fraktal. Transformasi affine yaitu transformasi linier yang diikuti dengan dilatasi, rotasi dan translasi (Budhi, 1995).

Definisi 2.2.1 Transformasi Linier

Misalkan V dan W adalah dua ruang vektor dan $f: V \rightarrow W$ adalah sebuah transformasi dari V ke W. Fungsi f dikatakan sebagai transformasi linier (pemetaan linier) apabila memenuhi dua sifat berikut,

a. (sifat kehomogenan) untuk setiap $\alpha \in \mathbb{R}$ dan $\vec{v} \in V$ berlaku $f(\alpha\vec{v}) = \alpha f(\vec{v})$

b. (sifat aditif) untuk setiap $\vec{u}, \vec{v} \in V$ berlaku $f(\vec{u} + \vec{v}) = f(\vec{u}) + f(\vec{v})$

(Anton dan Rorres, 2010)

Definisi 2.2.2 Translasi

Translasi merupakan transformasi yang memetakan titik (x, y) ke (x', y') yaitu bergeser sejauh p satuan searah sumbu x dan q satuan searah sumbu y, sehingga didapatkan persamaan:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} x + p \\ y + q \end{bmatrix} \quad (2.1)$$

(Kusno, 2003)

Definisi 2.2.3 Dilatasi

Dilatasi adalah transformasi yang mengubah ukuran bangun, tetapi tidak mengubah bentuk.

a. Dilatasi dengan pusat *O*(0,0) dan faktor skala *k* ditulis [*O*,*k*]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.2)$$

b. Dilatasi dengan pusat *A(a,b)* dan faktor skala *k* ditulis [*A(a,b)*,*k*]

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix} \begin{bmatrix} x - a \\ y - b \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \quad (2.3)$$

(Anton dan Rorres, 2010)

Definisi 2.2.4 Rotasi

Rotasi adalah suatu perpindahan benda pada gerakan melingkar. Pada dimensi dua, benda akan berputar pada pusat rotasi. Jika $T: R^2 \rightarrow R^2$ adalah suatu transformasi yang memetakan titik (x, y) ke titik (x', y') dan misalkan θ adalah sebuah sudut tetap maka persamaan rotasi melalui pusat $P(a, b)$ dengan arah rotasi berlawanan arah jarum jam adalah,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - a \\ y - b \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \quad (2.4)$$

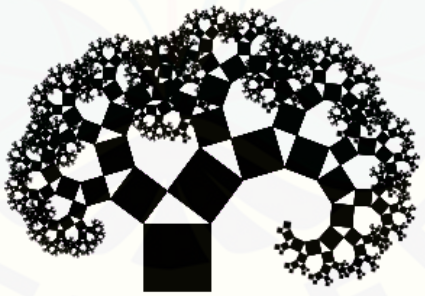
Dan persamaan rotasi dengan arah rotasi searah jarum jam adalah,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x-a \\ y-b \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \quad (2.5)$$

(Kusno, 2003)

2.3 Pohon Pythagoras

Pohon Pythagoras pertama kali dibangun oleh Albert E. Bosman pada tahun 1942. Albert E. Bosman adalah seorang guru matematika di Belanda. Nama untuk fraktal ini berasal dari fakta bahwa bentuk penggantian secara tradisional digunakan untuk menggambarkan Teorema Pythagoras (Matam, 2012). Sundusia (2019) membangkitkan pohon Pythagoras dengan memodifikasi sudut-sudut pada segitiga. Pohon Pythagoras divariasi dengan pemilihan sudut tetap sebesar $\theta_1 = 45^{\circ}, \theta_2 = 45^{\circ}$. Modifikasi pohon Pythagoras selanjutnya dengan variasi pemilihan sudut beda per iterasi, misal iterasi pertama $\theta_1 = 30^{\circ}, \theta_2 = 60^{\circ}$, iterasi kedua $\theta_1 = 45^{\circ}, \theta_2 = 45^{\circ}$ dan sudut random per iterasi, dimana pemilihan sudut periterasinya random. Gambar 2.1 merupakan gambar pohon Pythagoras dengan variasi sudut tetap sebesar $\theta_1 = 60^{o}, \theta_2 = 30^{o}$



Gambar 2.1 Pohon Pythagoras dengan sudut $\theta_1 = 60^{o}, \theta_2 = 30^{o}$ (Sumber: Matam, 2012)

Hakim dkk. (tanpa tahun) pada penelitiannya menyebutkan langkah-langkah pembentukan pohon Pythagoras secara heuristik sebagai berikut:

a. gambar sebuah persegi,
b. letakkan atau tempelkan segitiga siku-siku pada salah satu sisi persegi tersebut sepanjang sisi miring dari segitiga siku-siku yang ditempelkan,

c. letakkan atau tempelkan dua buah persegi pada sisi-sisi segitiga siku-siku yang lain, selain sisi miring yang telah digunakan,
d. letakkan atau tempelkan dua buah segitiga siku-siku,
e. letakkan atau tempelkan empat buah persegi,
f. letakkan atau tempelkan empat buah segitiga siku-siku,
g. letakkan atau tempelkan delapan buah persegi, dan seterusnya.

Langkah-langkah pembentukan pohon Pythagoras diilustrasikan pada Gambar 2.2 sebagai berikut:

Gambar 2.2 Langkah-langkah pembentukan pohon Pythagoras
(Sumber: Hakim, tanpa tahun)

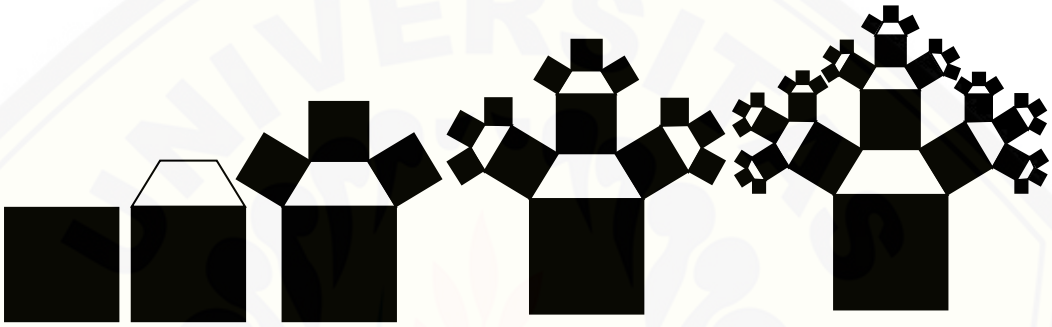
2.4 Pengembangan Pohon Pythagoras

Sundusia (2019) dalam penelitiannya membangkitkan pohon Pythagoras menggunakan transformasi affine dengan metode IFS. Pohon Pythagoras tersebut dimodifikasi dengan memvariasi pemilihan sudut pada segitiga siku-siku. Terdapat beberapa pemilihan sudut, diantara sudut tetap dengan besar sudut $\theta_1 = 45^{\circ}, \theta_2 = 45^{\circ}$, sudut beda per iterasi, dan sudut random per iterasi.

Hakim dkk. (tanpa tahun) menyebutkan bahwa pohon Pythagoras adalah sebuah fraktal yang dibangun secara berulang dari segitiga siku-siku dengan persegi yang dipasang pada masing-masing sisi. Dengan pendapat tersebut, dapat dilihat bahwa pohon Pythagoras memiliki dua percabangan. Peneliti selanjutnya memodifikasi percabangan pada pohon Pythagoras menjadi tiga percabangan yang disebut dengan istilah pohon fraktal. Percabangan ditentukan dengan menghitung panjang ketiga sisi terlebih dahulu, setelah mengetahui panjang sisi-sisi tersebut maka bentuk dasar dari percabangan akan terbentuk.

Langkah-langkah pembentukan pohon fraktal diilustrasikan sebagai pada Gambar 2.3:

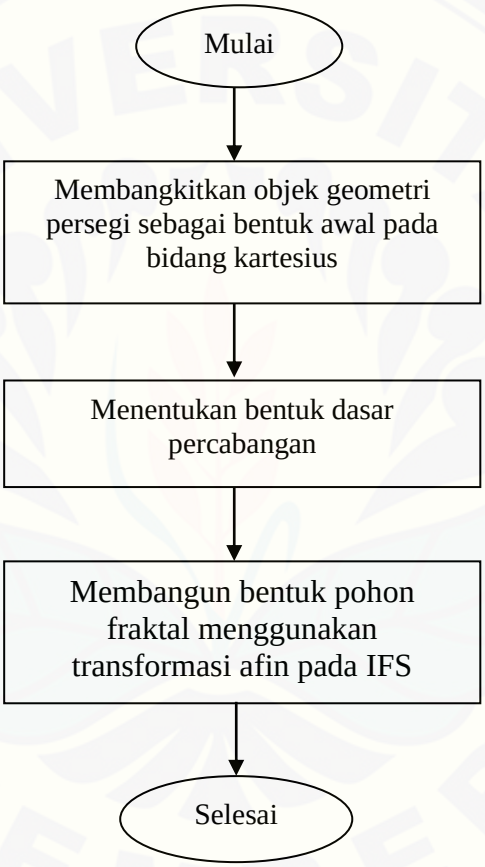
1. menggambar sebuah persegi,
2. menentukan percabangan dengan menghitung panjang ketiga sisi yang selanjutnya akan menjadi bentuk dasar dari percabangan,
3. membangun tiga bentuk persegi yang kemudian ditempelkan pada masing-masing sisi pada bentuk dasar percabangan,
4. dan seterusnya.



Gambar 2.3 Langkah-langkah pembentukan pohon fraktal

BAB 3. METODE PENELITIAN

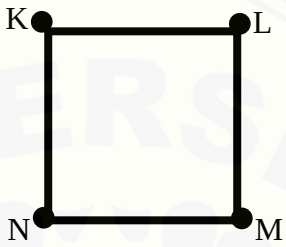
Pada bab ini akan dibahas mengenai langkah-langkah dalam menyelesaikan penelitian Pembangkitan Pohon Fraktal Tiga Cabang dengan Metode *Iterated Function System*. Langkah-langkah penelitian digambarkan dengan diagram alir pada Gambar 3.1



Gambar 3.1 Skema Metode Penelitian

3.1 Membangkitkan Objek Geometri Persegi sebagai Bentuk Awal pada Koordinat Kartesius

Objek geometri yang dibangkitkan sebagai bentuk awal dari pohon fraktal yaitu persegi seperti pada Gambar 3.2. Langkah pertama dengan menentukan posisi awal atau menentukan titik awal *(x,y)* sebanyak jumlah titik sudut dari persegi pada koordinat kartesius.

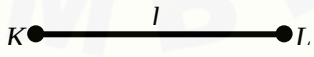


Gambar 3.2 Persegi sebagai bentuk awal

3.2 Menentukan Bentuk Dasar Percabangan

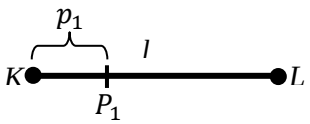
Bentuk dasar percabangan didapatkan setelah membangkitkan objek awal berupa persegi. Bentuk dasar percabangan dari pohon fraktal dihasilkan dari penentuan tinggi (*t*) dan P yang diinputkan. Adapun pemilihan tinggi dan P untuk mendapatkan bentuk-bentuk pohon fraktal tiga cabang, yaitu tinggi dan P yang diinputkan secara manual dengan beberapa aturan syarat atau tinggi dan P yang diatur secara random. Prosedur mendapatkan bentuk dasar percabangan adalah sebagai berikut:

a. Titik *K* dan *L* didapatkan dari persegi awal, *l* adalah jarak antara titik *K* dan *L* seperti pada Gambar 3.3



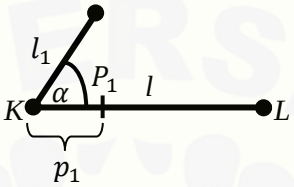
Gambar 3.3 Jarak antara titik *K* dan *L*

b. Posisi P_1 ditentukan untuk mengatur panjang p_1. Jadi, p_1 adalah panjang yang diperoleh dari titik *K* ke P_1. Ilustrasinya pada Gambar 3.4



Gambar 3.4 Panjang p_1 dan P_1

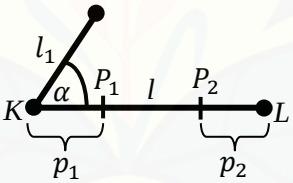
c. Panjang l_1 didapat dari pembagian p_1 dengan $\sin(90-\alpha)$, letak sudut α dan panjang l_1 seperti pada Gambar 3.5



$$l_1 = \frac{p_1}{\sin(90-\alpha)}$$

Gambar 3.5 Letak sudut α dan l_1

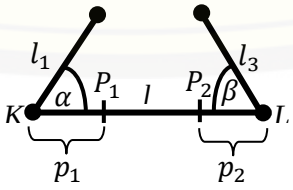
d. Posisi P_2 ditentukan untuk mengatur panjang p_2. Jadi, p_2 adalah panjang yang diperoleh dari P_2 ke titik L. Ilustrasinya pada Gambar 3.6



$$p_2 = l - P_2$$

Gambar 3.6 Panjang p_2 dan P_2

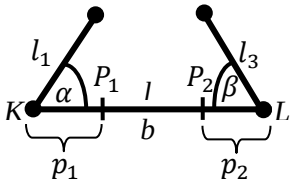
e. Panjang l_3 didapat dari pembagian p_2 dengan $\sin(90-\beta)$, letak sudut β dan l_3 seperti pada Gambar 3.7



$$l_3 = \frac{p_2}{\sin(90-\beta)}$$

Gambar 3.7 Letak sudut β dan l_3

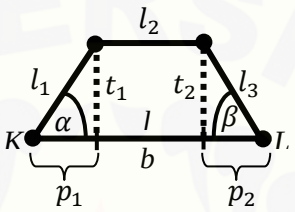
f. Panjang b didapat dari selisih l dengan penjumlahan dari p_1 dan p_2, b adalah panjang dari P_1 ke P_2. Ilustrasinya seperti pada Gambar 3.8



$$b = l - (p_1 + p_2)$$

Gambar 3.8 Ilustrasi b

g. Panjang l_2 didapat dari akar penjumlahan b^2 dengan t_3, dimana t_3 didapat dari selisih t_1 dan t_2. Panjang l_2, t_1, t_2 seperti pada Gambar 3.9



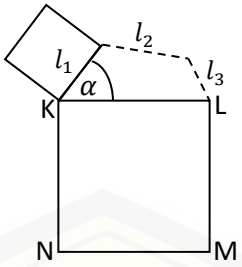
$$l_2 = \sqrt{b^2 + t_3}$$
$$t_3 = |t_1 - t_2|$$

Gambar 3.9 Panjang l_2, t_1, t_2

3.3 Membangun Bentuk Pohon Fraktal Menggunakan Transformasi Afinne pada IFS

3.3.1 Cabang Pertama di sisi l_1

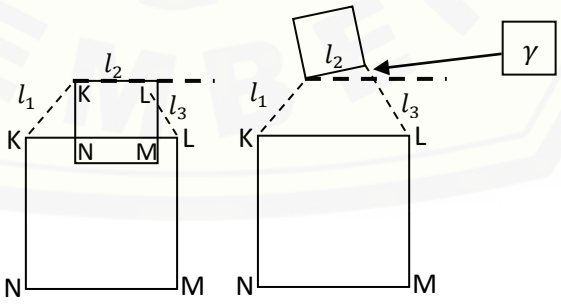
Cabang pertama didapat dari persegi awal yang diduplikasi. Selanjutnya, pada hasil duplikasi diterapkan beberapa transformasi afin yaitu operasi dilatasi menggunakan Persamaan (2.3) yang ditentukan oleh titik pusat dilatasi dan faktor skala dilatasi. Faktor skala dilatasi (k_1) didapat dari pembagian l_1 dengan l. Titik pusat dilatasi pada titik K. Hasil dari dilatasi kemudian dilakukan operasi rotasi sebesar sudut α dengan menentukan arah perputaran berlawanan jarum jam menggunakan Persamaan (2.4) dan titik pusat rotasi berada di titik K. Operasi dilatasi dan rotasi ini diterapkan secara berulang-ulang pada persegi hingga membentuk pohon fraktal. Gambar 3.10 merupakan ilustrasi percabangan di l_1



Gambar 3.10 Ilustrasi percabangan di l_1

3.3.2 Cabang Kedua di sisi l_2

Cabang kedua didapat dari persegi awal yang diduplikasi. Selanjutnya, pada hasil duplikasi diterapkan beberapa transformasi afin yaitu operasi dilatasi menggunakan Persamaan (2.3) yang ditentukan oleh titik pusat dilatasi dan faktor skala dilatasi. Faktor skala dilatasi (k_2) didapat dari pembagian l_2 dengan l. Titik pusat dilatasi pada titik *K*. Hasil dari dilatasi kemudian dilakukan operasi translasi menggunakan Persamaan (2.1). Hasil dari translasi kemudian dilakukan operasi rotasi sebesar sudut γ dengan menentukan arah perputaran berlawanan jarum jam menggunakan Persamaan (2.4) atau searah perputaran jarum jam menggunakan Persamaan (2.5) dan titik pusat rotasi berada di titik *K* persegi baru hasil translasi. Gambar 3.11 merupakan ilustrasi percabangan di l_2. Operasi dilatasi, translasi dan rotasi ini diterapkan secara berulang-ulang pada persegi hingga membentuk pohon fraktal.

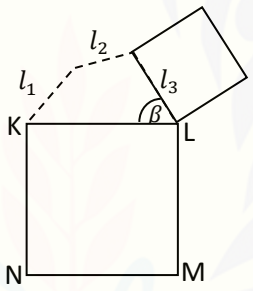


(a) Hasil translasi (b) Hasil rotasi

Gambar 3.11 Ilustrasi percabangan di l_2

3.3.3 Cabang Ketiga di sisi l_3

Cabang ketiga didapat dari persegi awal yang diduplikasi. Selanjutnya, pada hasil duplikasi diterapkan beberapa transformasi afin yaitu operasi dilatasi menggunakan Persamaan (2.3) yang ditentukan oleh titik pusat dilatasi dan faktor skala dilatasi. Faktor skala dilatasi (k_3) didapat dari pembagian l_3 dengan l. Titik pusat dilatasi pada titik L. Hasil dari dilatasi kemudian dilakukan operasi rotasi sebesar sudut β dengan menentukan arah perputaran searah jarum jam menggunakan Persamaan (2.5) dan titik pusat rotasi berada di titik L. Gambar 3.12 merupakan ilustrasi percabangan di l_3. Operasi dilatasi dan rotasi ini diterapkan secara berulang-ulang pada persegi hingga membentuk pohon fraktal.



Gambar 3.12 Ilustrasi percabangan di l_3

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapat dari penelitian ini adalah mendapatkan prosedur dan beberapa hasil pembangkitan pohon fraktal dengan metode *Iterated Function System*, dengan cara:

a. menentukan posisi awal persegi sebagai bentuk awal pada koordinat kartesius,
b. mendapatkan bentuk dasar percabangan,
c. melakukan transformasi affine dengan:
    1. persegi awal diduplikasi, didilatasi, lalu dirotasi di titik pusat K dengan besar sudut α dan arah perputaran berlawanan arah jarum jam,
    2. persegi awal diduplikasi, didilatasi, ditranslasi, dan dirotasi di titik pusat K dengan besar sudut γ dan arah perputaran berlawanan arah jarum jam,
    3. persegi awal diduplikasi, didilatasi, dan dirotasi di titik pusat L dengan besar sudut β dan arah perputaran searah jarum jam.

5.2 Saran

Skripsi ini meneliti tentang pembangkitan pohon fraktal dengan metode IFS. Penelitian selanjutnya, diharapkan dapat mengembangkan bentuk awal selain persegi agar lebih bervariatif dan menggunakan metode atau teknik yang lain agar menghasilkan pohon fraktal yang tidak berpotongan. Dan juga, untuk penelitian selanjutnya diharapkan dapat membangkitkan visualisasi pohon dimana bentuknya menyerupai bentuk aslinya, misalnya membangkitkan pohon beringin.

DAFTAR PUSTAKA

Anton, H dan Rorres, C. 2010. *Elementary Linear Algebra 10th Edition*. Jakarta: Erlangga.

Beek, A. 1996. *Iterated Function System Optimization*. dissertations.ub.rug.nl / FILES / faculties / science / 1996 / m.m.lankhorst / c7.pdf.

Budhi, W.S. 1995. *Aljabar Linear*. Jakarta: Gramedia Pustaka Utama.

Hakim, L., Suprabowo, A., dan Asy'ari, M.H. *Menggambar Fraktal Dengan Teknik Heuristik.* Bandung: Institut Teknologi Bandung.

Hasang, S., Suparjo, S. 2012. Geometri Fraktal dalam Rancangan Arsitektur. *Media Marasan* 9(1):111-124.

Kusno. 2003. *Geometri Rancang Bangun Studi Surfas Putar Transformasi Titik dan Proyeksi*. Jember: Fakultas MIPA Universitas Jember.

Mandelbrot, B. 1983. *The Fractal Geometry of Nature*. New York: W.H.Freeman and Company.

Matam, S. 2012. *Phytagoras Trees*. https://www.codewalk.com/2012/01/pythagoras-trees.html. [29 Mei 2019]

Rahayu, I. 2012. Hitung Ukuran Sudut Poligon Dengan Bantuan Pembagian Bidang dan Duplikasi Poligon Sebangun Serta Aproksimasi Luasan Poligon Dengan Bantuan Kesebangunan Segitiga. Tesis. Jember: Jurusan Matematika FMIPA Universitas Jember.

Ramandhani, M.R. 2012. *Penggunaan Sistem Fungsi Iterasi untuk Membangkitkan Fraktal Beserta Aplikasinya*. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2012-2013/Makalah2012/Makalah-IF2091-2012-052.pdf [17 Agustus 2019]

Prasasti, I. 2018. Pemanfaatan Metode *Iterated Function System* Dalam Pengembangan Motif Anyaman. Skripsi. Jember: Jurusan Matematika FMIPA Universitas Jember.

Purnomo, K. D. 2014. Pembangkitan Segitiga Sierpinski Dengan Transformasi Affine. *Prosiding Seminar Nasional Matematika*. Jember: Jurusan Matematika FMIPA Universitas Jember. Hal:365-375.

Sampurno, J., Faryuni, I.D. 2016. *Metode Analisis Fraktal*. Yogyakarta: Deepublish.

Sundusia, J. K. 2019. Pembangkitan Fraktal Pohon Pythagoras Menggunakan *Iterated Function System*. Skripsi. Jember: Jurusan Matematika FMIPA Universitas Jember.

Utomo, B. 2011. Fraktal dan Invers Fraktal. *Jurnal Matematika* 2(1):28-37.

LAMPIRAN

4.1 Script Tampilan GUIDE

```
function varargout = GUIFraktal(varargin)
% GUIFRAKTAL MATLAB code for GUIFraktal.fig
%      GUIFRAKTAL, by itself, creates a new GUIFRAKTAL or raises
the existing
%      singleton*.
%
%      H = GUIFRAKTAL returns the handle to a new GUIFRAKTAL or
the handle to
%      the existing singleton*.
%
%      GUIFRAKTAL('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in GUIFRAKTAL.M with the given
input arguments.
%
%      GUIFRAKTAL('Property','Value',...) creates a new GUIFRAKTAL
or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before GUIFraktal_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUIFraktal_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 GUIFraktal

% Last Modified by GUIDE v2.5 25-Nov-2019 09:01:45

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUIFraktal_OpeningFcn, ...
                   'gui_OutputFcn',  @GUIFraktal_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 GUIFraktal is made visible.
function GUIFraktal_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 GUIFraktal (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);
clc;
movegui(gcf,'center');
set(handles.radiobutton1,'value',1);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',0);
set(handles.edit1,'enable','on','string','');
set(handles.edit2,'enable','on','string','');
set(handles.edit3,'enable','on','string','');
set(handles.edit4,'enable','on','string','');
set(handles.edit5,'string','');
set(handles.popupmenu1,'value',1);
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);
% UIWAIT makes GUIFraktal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUIFraktal_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;


% --- 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;
movegui(gcf,'center');
set(handles.radiobutton1,'value',1);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',0);
set(handles.edit1,'enable','on','string','');
set(handles.edit2,'enable','on','string','');
set(handles.edit3,'enable','on','string','');
set(handles.edit4,'enable','on','string','');
set(handles.edit5,'string','');
set(handles.popupmenu1,'value',1);
cla(handles.axes1,'reset');
set(handles.axes1,'XTick',[],'YTick',[]);

% --- 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)
clc;
cla(handles.axes1,'reset');
axes(handles.axes1);
set(gca,'XTick',[],'YTick',[]);

%
rb1 = get(handles.radiobutton1,'value');
rb2 = get(handles.radiobutton2,'value');
rb3 = get(handles.radiobutton3,'value');
iterasi = str2num(get(handles.edit5,'string'));
color=get(handles.popupmenu1,'value');
if ~isempty(iterasi)
    Coor=[0 1 1 0;1 1 0 0];
    if color > 10
        clr = [];
    elseif color == 10
        clr = rand(1,3);
    else
        fixcolor=[0 0 0;.4 .4 .4;1 0 0;0 1 0;0 0 1;0 1 1;1 0 1;1 1
0;1 1 1];
        clr=fixcolor(color,:);
    end
    if rb1 == 1 % P dan t ditetapkan
        P1 = str2num(get(handles.edit1,'string'));
        P2 = str2num(get(handles.edit2,'string'));
        t1 = str2num(get(handles.edit3,'string'));
        t2 = str2num(get(handles.edit4,'string'));
        if ~isempty(P1) && ~isempty(P2) && ~isempty(t1) &&
~isempty(t2)
            P = [P1 P2];
            t1s = t2 / (P2) * P1;
            t2s = t1 / (1 - P1) * (1 - P2);
            if t1 >= t1s && t2 >= t2s
```

```
                alpha = atand(t1 / P1);
                beta = atand(t2 / (1 - P2));
                theta = [alpha beta];
                IFS_Pohon(Coor, theta, P, iterasi, clr);
            else
                wrndlg('Segiempat non konveks');
            end
        end
    elseif rb2 == 1 % P dan t random awal
        P = rand(1,2)*0.8+0.1;
        while P(1) >= P(2)
            P = rand(1,2)*0.8+0.1;
        end
        P1 = P(1); P2 = P(2);
        LB1 = 0; UB1 = sqrt(0.8 - P1^2);
        t1 = rand * (UB1 - LB1) + LB1;
        LB2 = 0; UB2 = sqrt(0.8 - (1 - P2)^2);
        t2 = rand * (UB2 - LB2) + LB2;
        t1s = t2 / (P2) * P1;
        t2s = t1 / (1 - P1) * (1 - P2);
        while t1 < 1.2*t1s || t2 < 1.2*t2s
            P = rand(1,2)*0.8+0.1;
            while P(1) >= P(2)
                P = rand(1,2)*0.8+0.1;
            end
            P1 = P(1); P2 = P(2);
            LB1 = 0; UB1 = sqrt(0.8 - P1^2);
            t1 = rand * (UB1 - LB1) + LB1;
            LB2 = 0; UB2 = sqrt(0.8 - (1 - P2)^2);
            t2 = rand * (UB2 - LB2) + LB2;
            t1s = t2 / (P2) * P1;
            t2s = t1 / (1 - P1) * (1 - P2);
        end
        alpha = atand(t1 / P1);
        beta = atand(t2 / (1 - P2));
        theta = [alpha beta];
        IFS_Pohon(Coor, theta, P, iterasi, clr);
    elseif rb3 == 1 % P dan t random per cabang
        IFS_Pohon(Coor, [], [], iterasi, clr);
    end
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, 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)
clc;
frame=getframe(handles.axes1);
image=frame2im(frame);
[name_file,name_path] = uiputfile( ...
        {'*.jpg','File Type JPEG (*.jpg)';
        '*.bmp','File Type BITMAP (*.bmp)';
        '*.tif','File Type TIF (*.tif)';
```

```
        '*.png','File Type PNG (*.png)';
        '*.jpg;*.bmp;*.tif;*.png','Files of type
(*.jpg,*.bmp,*.tif,*.png)'},...
        'Save As Image');
if name_file~=0
    imwrite(image,fullfile(name_path,name_file));
    msgbox('Save complete');
end


function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5 as text
%        str2double(get(hObject,'String')) returns contents of
edit5 as a double


% --- Executes during object creation, after setting all
properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit4 as text
%        str2double(get(hObject,'String')) returns contents of
edit4 as a double


% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of
edit3 as a double


% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double
```

```
% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double


% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.radiobutton1,'value',1);
set(handles.radiobutton2,'value',0);
```

```
set(handles.radiobutton3,'value',0);
set(handles.edit1,'enable','on','string','');
set(handles.edit2,'enable','on','string','');
set(handles.edit3,'enable','on','string','');
set(handles.edit4,'enable','on','string','');
% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',1);
set(handles.radiobutton3,'value',0);
set(handles.edit1,'enable','inactive','string','random');
set(handles.edit2,'enable','inactive','string','random');
set(handles.edit3,'enable','inactive','string','random');
set(handles.edit4,'enable','inactive','string','random');
% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.radiobutton1,'value',0);
set(handles.radiobutton2,'value',0);
set(handles.radiobutton3,'value',1);
set(handles.edit1,'enable','inactive','string','random');
set(handles.edit2,'enable','inactive','string','random');
set(handles.edit3,'enable','inactive','string','random');
set(handles.edit4,'enable','inactive','string','random');
% Hint: get(hObject,'Value') returns toggle state of radiobutton3


% --- 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
```

4.2 Script IFS

```
function IFS_Pohon(Coor, theta, P, iter, color)
if ~isempty(color)
    clr = color;
else
    clr = rand(1,3);
end
patch(Coor(1,:),Coor(2,:),clr);
axis image
set(gca,'XTick',[],'YTick',[]);
pause(0.1);
if iter > 0
    if ~isempty(theta) && ~isempty(P)
        alpha = theta(1);
        beta = theta(2);
        P1 = P(1);
        P2 = P(2);
    else
        P = rand(1,2)*0.8+0.1;
        while P(1) >= P(2)
            P = rand(1,2)*0.8+0.1;
        end
        P1 = P(1); P2 = P(2);
        LB1 = 0; UB1 = sqrt(0.8 - P1^2);
        t1 = rand * (UB1 - LB1) + LB1;
        LB2 = 0; UB2 = sqrt(0.8 - (1 - P2)^2);
        t2 = rand * (UB2 - LB2) + LB2;
        t1s = t2 / (P2) * P1;
        t2s = t1 / (1 - P1) * (1 - P2);
        while t1 < 1.2*t1s || t2 < 1.2*t2s
            P = rand(1,2)*0.8+0.1;
            while P(1) >= P(2)
                P = rand(1,2)*0.8+0.1;
            end
            P1 = P(1); P2 = P(2);
            LB1 = 0; UB1 = sqrt(0.8 - P1^2);
            t1 = rand * (UB1 - LB1) + LB1;
            LB2 = 0; UB2 = sqrt(0.8 - (1 - P2)^2);
            t2 = rand * (UB2 - LB2) + LB2;
```

```
            t1s = t2 / (P2) * P1;
            t2s = t1 / (1 - P1) * (1 - P2);
        end
        alpha = atand(t1 / P1);
        beta = atand(t2 / (1 - P2));
        theta = []; P = [];
    end
    L = sqrt((Coor(1,1) - Coor(1,2))^2 + (Coor(2,1) -
Coor(2,2))^2);
    La = L * P1;
    Lc = L * (1-P2);
    L1 = La / sind(90-alpha);
    L3 = Lc / sind(90-beta);
    t1 = sqrt(L1^2 - La^2);
    t2 = sqrt(L3^2 - Lc^2);
    t3 = abs(t1 - t2);
    Lb = L - (La + Lc);
    L2 = sqrt(Lb^2 + t3^2);
    gamma = asind(t3 / L2);
    %
    k1 = L1 / L;
    k2 = L2 / L;
    k3 = L3 / L;
    %kiri
    A = Coor(:,1); % titik tumpu
    Coor1a = k1 * eye(2) * (Coor - repmat(A,1,4)) + repmat(A,1,4);
% dilatasi
    A = Coor1a(:,1);
    Coor1b = [cosd(90 + alpha)  -sind(90 + alpha);sind(90 + alpha)
cosd(90 + alpha)] * (Coor1a - repmat(A,1,4)) + repmat(A,1,4); %
rotasi
    Coor1c = Coor1b(:,[2 3 4 1]);
    IFS_Pohon(Coor1c, theta, P, iter - 1, color);
    %atas
    B = Coor(:,1); % titik tumpu
    Coor2a = k2 * eye(2) * (Coor - repmat(B,1,4)) + repmat(B,1,4);
% dilatasi
    B = Coor2a(:,1);
    if t1 >= t2
        Coor2b = [cosd(90 - gamma)  -sind(90 - gamma);sind(90 -
gamma) cosd(90 - gamma)] * (Coor2a - repmat(B,1,4)) +
repmat(B,1,4); % rotasi
    else
        Coor2b = [cosd(90 + gamma)  -sind(90 + gamma);sind(90 +
gamma) cosd(90 + gamma)] * (Coor2a - repmat(B,1,4)) +
repmat(B,1,4); % rotasi
    end
    B = Coor2b(:,1);
    B2 = Coor1b(:,end);
    Coor2c = Coor2b + repmat(B2 - B,1,4); % translasi
    Coor2d = Coor2c(:,[2 3 4 1]);
    IFS_Pohon(Coor2d, theta, P, iter - 1, color);
    %kanan
    C = Coor(:,2); % titik tumpu
    Coor3a = k3 * eye(2) * (Coor - repmat(C,1,4)) + repmat(C,1,4);
% dilatasi
```

```
    C = Coor3a(:,2);
    Coor3b = [cosd(90 + beta) sind(90 + beta); -sind(90 + beta)
cosd(90 + beta)] * (Coor3a - repmat(C,1,4)) + repmat(C,1,4); %
rotasi
    Coor3c = Coor3b(:,[4 1 2 3]);
    IFS_Pohon(Coor3c, theta, P, iter - 1, color);
end
```