

I. PENDAHULUAN FISIKA SISTEM KOMPLEKS

1.1. Sistem Kompleks

Sistem Kompleks adalah suatu sistem yang terdiri dari bagian-bagian atau agen-agen yang saling berinteraksi disertai kemampuan untuk menghasilkan perilaku kolektif makroskopik yang baru yang manifestasinya merupakan formasi spontan dari struktur temporal, spasial atau fungsional individu-individu tersebut.

Model-model dari sistem-sistem kompleks dapat memetakan keadaan-keadaan nyata yang sangat beragam mulai dari iklim, emisi cahaya koheren dari laser, sistem reaksi-difusi kimia, dinamika pasar modal, jaringan sel biologis, statistik dan prediksi gempa bumi, arus lalu lintas, otak manusia, pertumbuhan populasi, pembentukan opini dalam sistem sosial dan masih banyak lagi.

Semua contoh sistem-sistem kompleks di atas memperlihatkan karakteristik yang sama yaitu:

1. Sistem tersebut terdiri dari sejumlah besar *agen* atau individu atau bagian yang saling berinteraksi.
2. Sistem tersebut memperlihatkan *emergence*, yaitu perilaku kolektif dengan kemampuan pengaturan diri (*self-organization*) yang sulit diprediksi dari pengetahuan kita tentang perilaku agen-agenya.
3. Perilaku baru sistem yang muncul (*emergence*) tidak dihasilkan dari keberadaan suatu pengontrol utama (*central controller*).

1.2. Fisika Sistem Kompleks

Fisika Sistem Kompleks adalah kajian terhadap suatu sistem fisis yang perilakunya mencirikan sistem kompleks seperti misalnya dinamika nonlinier sistem banyak-benda, dinamika kegempaan, deformasi pada granular dan zat amorf, ekonofisik, optika kuantum molekuler, fenomena kolektif zat padat dan lain sebagainya.

Meskipun cakupan dan juga metodologi yang digunakan dalam kajian sistem kompleks bervariasi dan seringkali terkesan *overlap*, kajian fisika sistem kompleks biasanya melibatkan konsep-konsep serta perangkat-perangkat penalaran yang di antaranya adalah: pengaturan diri, dinamika non-linier, sinergetika, turbulensi, sistem dinamik, katastrofi, instabilitas, proses stokastik, *chaos*, jaringan dan *graph*, cellular automata, sistem adaptif, algoritma genetik dan intelegensia komputasi dan lainnya.

1.3. Model

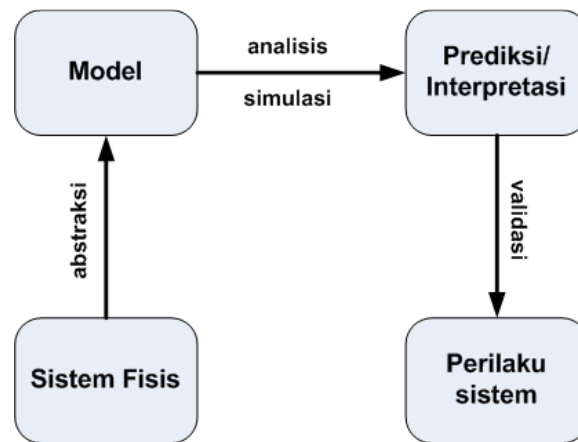
Suatu model adalah representasi matematis dari suatu sistem. Model berfungsi untuk menggambarkan bagaimana variable-variabel dari sistem saling berhubungan untuk mendefinisikan suatu keadaan atau respons. Dalam sistem fisis yang nyata, suatu sistem biasanya ditentukan oleh banyak variabel yang sepertinya penting. Namun demikian tidak seluruhnya dari variable-variabel tersebut yang harus dilibatkan dalam proses pemodelan. Jumlah variabel yang relevan untuk dipertahankan sebagai parameter model hanyalah variabel-variabel yang memiliki peran esensial dalam interpretasi fenomena sistem yang dikaji.

Proses pembuatan model disebut *abstraksi*. Pada konteks ini, “abstrak” adalah lawan kata dari “realistik.” Suatu model abstrak memperlihatkan sedikit kesamaan dengan sistem yang dimodelkannya. Model realistik adalah model yang melibatkan semakin banyak detail (variabel) dan semakin kuat terhubung dengan sistem yang sesungguhnya.

Abstraksi melibatkan pembuatan keputusan yang valid mengenai faktor-faktor yang harus dilibatkan maupun yang harus diabaikan. Model yang lebih realistik tidak selalu yang terbaik. Suatu model bermanfaat jika dapat dianalisa secara matematik dan disimulasikan

dengan perangkat komputasi. Model yang terlalu realistik mungkin akan sulit disimulasikan maupun dianalisis.

Suatu model dianggap berhasil jika ia cukup memenuhi kebutuhan yang dituju. Proses pengecekan apakah model kita cukup baik atau tidak disebut *validasi*. Bentuk validasi yang sangat kuat adalah dengan membuat pengukuran pada sistem fisis aktual dan membandingkannya dengan prediksi dari model. Jika cara di atas tak dapat dilakukan maka masih terdapat cara lain yang lebih lunak yaitu dengan membandingkan berbagai model untuk sistem yang sama. Jika hasilnya inkonsisten, paling tidak kita akan mendapatkan bahwa salah satu salah. Ukuran perbedaan hasil antar model juga dapat memberi petunjuk mengenai reliabilitas prediksi-prediksi yang dibuat.



Gambar 1.1. Ilustrasi singkat proses pemodelan: abstraksi dan validasi

1.4. Tipe-tipe model

Model yang merepresentasikan suatu sistem dapat bersifat *linier*, yaitu bila output (respons, keadaan, data) berbanding lurus dengan input (variabel, parameter model) dan disebut *non-linier* jika output dan input tidak berbanding lurus (misalnya berbanding terbalik, kuadratik, polinomial, eksponensial dan sebagainya). Sistem kompleks lazimnya memperlihatkan non-linieritas yang sangat kuat.

Suatu model bersifat *deterministik* jika semua variabel dan parameter merupakan fungsi variabel bebas ruang dan waktu. Rumus-rumus fisika termasuk dalam kategori model deterministik di mana hubungan antar variabel dinyatakan secara eksplisit dan baku. Model *statistik* adalah ekspresi matematik yang menggambarkan perilaku sistem dari sudut pandang variabel-variabel acaknya dan distribusi probabilitas yang bersesuaian.

Bergantung pada jumlah dimensi ruang, sistem dapat dikategorikan sebagai 0D, 1D, 2D atau 3D. Model 0D tidak memiliki ketergantungan terhadap ruang, misalnya hanya bergantung waktu. Model-model ekologi yang berkaitan dengan populasi spesies biologi dalam suatu lingkungan tertentu lazim bertipe ini. Karena hanya bergantung pada satu variabel bebas yaitu waktu t , maka formulasi analitiknya berkaitan dengan *persamaan differensial biasa* (PDB = *ordinary differential equation* [ODE]). Jika suatu model bergantung pada dua atau lebih variabel bebas maka persoalan analitiknya berkaitan dengan *persamaan differensial parsial* (PDP = *partial differential equation* [PDE]).

Model yang tidak memiliki ketergantungan terhadap waktu disebut model pada *keadaan tunak* atau *steady state* atau stasioner, sedangkan model yang bergantung waktu disebut *non-*

tunak, *unsteady* atau *transient*. Pada sistem yang nyata pendekatan keadaan tunak dapat diterapkan jika proses-proses internal memiliki waktu yang cukup lama untuk menyesuaikan dengan kondisi luar yang konstan. Salah satu syarat keberlakuan pendekatan ini adalah bila proses-proses eksternal atau parameter-parameternya tidak bergantung waktu.

1.5. Sistem dinamik

Kajian sistem kompleks didominasi oleh upaya memahami dinamika atau perilaku sistem tersebut, sehingga sistem yang ditinjau seringkali disebut sistem dinamik. Sistem dinamik adalah sistem yang terbangun dari *ruang fasa* yang elemen-elemennya merepresentasikan keadaan-keadaan yang mungkin dari sistem tersebut; waktu t yang dapat berbentuk kontinyu maupun diskret; serta *kaidah evolusi* (aturan yang memungkinkan kita memprediksi keadaan sistem pada waktu t jika keadaan pada waktu sebelumnya t_0 diketahui). Meskipun pada awalnya sebutan sistem dinamik ditujukan pada persoalan-persoalan persamaan gerak sistem partikel (dalam bentuk PDB maupun PDP), istilah sistem dinamik sering juga dipakai sebagai sinonim dari sistem persamaan non-linier. Contoh-contoh sistem dinamik di antaranya adalah: pertumbuhan populasi spesies biologi, ayunan bandul sederhana, osilator non-linier, model aliran airtanah dan sebagainya.

Latihan 1.1

1. Apakah jam mekanik termasuk sistem kompleks? Jelaskan jawaban Anda.
2. Sebutkan contoh-contoh sistem kompleks dalam ranah fisika, kimia, biologi, sosial, ekonomi, serta sains terpadu. Jelaskan juga mengapa Anda menganggapnya sebagai sistem kompleks.
3. Misalkan Pemkot A ingin menentukan apakah di suatu ruas jalan tertentu pembuatan jalur penyeberangan (*zebracross*) cukup memenuhi syarat-syarat keselamatan para pejalan kakinya. Jika Anda adalah staf yang ditugaskan menentukan jadi tidaknya dibuat *zebracross*, model apakah yang Anda usulkan untuk memandu keputusan Anda?

Petunjuk: sebenarnya faktor yang mempengaruhi keselamatan pejalan kaki di jalan tersebut sangat banyak dan cukup kompleks. Anda harus mencari model sederhana yang menggambarkan syarat dasar keselamatan menyeberang terpenuhi. Manfaatkan asumsi-asumsi berikut:

- Jalan yang dilalui satu arah dan lurus
 - Kecepatan lalu lintas kendaraan konstan
 - Kepadatan lalu lintas kendaraan konstan
 - Kecepatan penyeberang jalan konstan
-



II. SISTEM DINAMIK DAN PERSAMAAN DIFFERENSIAL

2.1. Persamaan Differensial Biasa (PDB)

Persamaan differensial biasa (PDB) adalah persamaan untuk suatu fungsi dengan satu variabel, misalnya $x(t)$, yang melibatkan fungsi tersebut dan turunannya.

$$\dot{x}(t) = \frac{dx}{dt} = 2, \quad \dot{y}(t) = 3t, \quad \dot{z}(t) = \frac{1}{2}z(t) \quad (2.1)$$

adalah contoh-contoh persamaan PDB. *Solusi* suatu persamaan differensial adalah fungsi yang memenuhi persamaan differensial tersebut. Solusi dari ketiga persamaan di atas masing-masing adalah

$$x(t) = 2t + c_1, \quad y(t) = \frac{3}{2}t^2 + c_2, \quad z(t) = c_3 e^{\frac{1}{2}t} \quad (2.2)$$

c_1 , c_2 , dan c_3 adalah konstanta integrasi. Solusi-solusi pada (2.2) disebut sebagai *solusi umum*. Untuk mendapatkan solusi yang unik (khusus) maka nilai konstanta integrasi harus dibuat tetap dengan memasukkan *syarat awal* misalnya $x(0) = 1$, kita dapatkan $c_1 = 1$. Pada contoh pertama $x(t) = 2t + 1$ adalah solusi untuk $x(t) = 2t + c_1$ dengan syarat awal $x(0) = 1$.

Interpretasi PDB yang secara umum berbentuk

$$\dot{x}(t) = f(x(t), t) \quad (2.3)$$

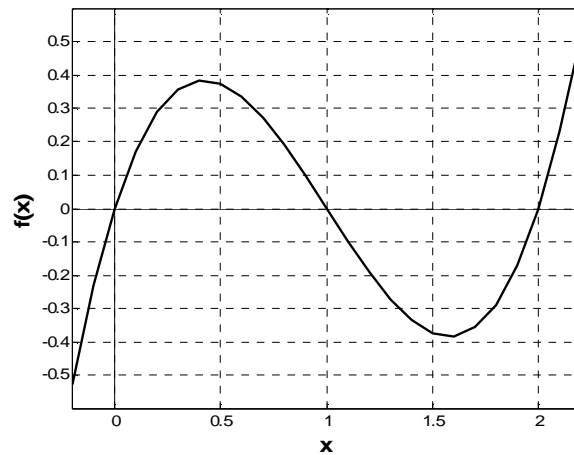
adalah sebagai berikut:

Suku ruas kiri menyatakan *laju perubahan* $x(t)$ terhadap waktu. Ruas kanan menyatakan semua sumber yang menyebabkan perubahan nilai $x(t)$. Ungkapan differensial ini sangat bermanfaat untuk memodelkan proses-proses perubahan suatu sistem terhadap waktu, misalnya laju peluruhan unsur radioaktif, gerak sistem pegas-massa serta laju perkembangbiakan spesies biologis.

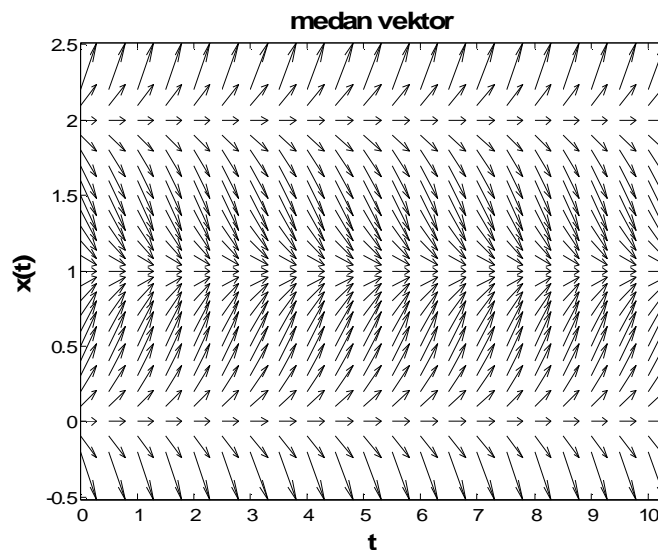
Persamaan (2.3) adalah contoh bentuk differensial scalar. Jika fungsi $f(x, t)$ tidak bergantung waktu maka (2.3) disebut PDB scalar yang *otonom*. Sebagai contoh misalkan kita memiliki persamaan

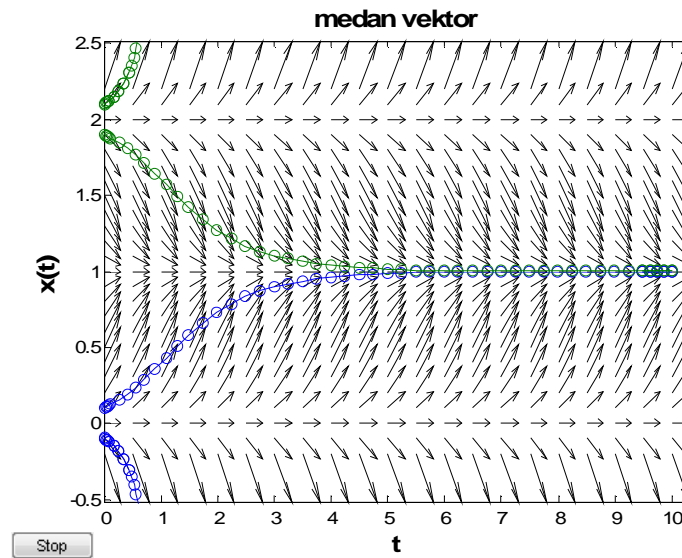
$$\dot{x}(t) = x(1-x)(2-x) \text{ kurangnya diperlihatkan pada Gambar 2.1. Fungsi } f(x) = x(1-x)(2-x) \text{ bernilai nol di } x(t) = 0; x(t) = 1 \text{ dan } x(t) = 2. \text{ Ketiganya merupakan solusi konstan untuk persamaan } \dot{x}(t) = x(1-x)(2-x). \quad (2.3)$$

Ketiga titik di atas disebut titik-titik kesetimbangan atau keadaan tunak dari persamaan (2.3). Jika pada saat $t = 0$ suatu solusi memiliki nilai 0 (atau 1 atau 2) maka nilainya akan tetap demikian untuk semua $t > 0$. Ruas kiri (2.3) menggambarkan perubahan $x(t)$ seiring waktu. Solusi $x(t)$ meningkat selama $f(x) > 0$ yaitu pada interval $(0, 1)$ serta $(2, \infty)$ dan menurun ketika $f(x) < 0$ yaitu interval $(-\infty, 0)$ dan $(1, 2)$. Jika misalnya syarat awal $x(0)$ berada pada interval $(0, 1)$, maka solusi akan meningkat dan konvergen ke nilai 1 untuk $t \rightarrow \infty$. Gambar 2.1 adalah kurva analisis lintasan fasa yang bersesuaian untuk contoh persamaan kita.

Gambar 2.1. Analisis lintasan fasa (*phase-line analysis*) untuk $f(x) = x(1-x)(2-x)$

Medan vektor (*vector field*) dapat dimanfaatkan secara kualitatif untuk menjelaskan perilaku solusi persamaan (2.3) yang perilakunya diperlihatkan pada Gambar 2.2. Medan vektor didapat dengan mengevaluasi kemiringan dari solusi $x(t)$ yaitu $f(x)$ untuk beberapa titik (t, x) dengan mem-plot garis panah yang mengindikasikan kemiringan tersebut dalam diagram (t, x) . Karena solusi-solusi $x(t)$ harus memiliki kemiringan $\dot{x}(t)$ maka kurva-kurva solusi akan berarah tangensial terhadap panah-panah kemiringan ini. Gambar 2.3 memperlihatkan empat contoh solusi tipikal yang dengan jelas menunjukkan bagaimana solusi mengikuti medan vektor. Titik-titik keadaan tunak yaitu 0, 1 dan 2 terlihat horizontal sepanjang t yang berarti bahwa kemiringannya nol ($\dot{x}(t) = 0$). Antara rentang $(0, 2)$, solusi yang tak berawal dari 0, 1, dan 2 akan cenderung menjauhi titik kesetimbangan 0 dan 2 namun akan konvergen ke 1. Dalam kasus ini titik-titik 0 dan 2 disebut titik *kesetimbangan tak-stabil* sedangkan titik 1 disebut titik *kesetimbangan stabil*.

Gambar 2.2. Medan vektor untuk $f(x) = x(1-x)(2-x)$

Gambar 2.3. Solusi tipikal untuk $dx/dt = f(x) = x(1-x)(2-x)$ **Latihan 2.1**

1. Plot ulang Gambar 2.1. Gunakan Matlab, Maple atau *software* lain jika perlu.
2. Gambar 2.2 dibuat dengan *code* yang ditulis menggunakan Matlab:

```
t=linspace(0,10,21);
x=linspace(-0.2,2.2,25);
[T X]=meshgrid(t,x);
fx=inline('x.*(1-x).*(2-x)+0.*t','t','x')
dt=T(1,2)-T(1,1);
DT=dt*ones(size(T));
FX=fx(T,X);
quiver(T,X,DT,FX,'k')
axis tight
xlabel('t','fontsize',14,'fontweight','b')
ylabel('x(t)','fontsize',14,'fontweight','b')
title('medan vektor','fontsize',14,'fontweight','b')
```

Cobalah di-*run* ulang dengan Matlab. Pelajari dan cobalah fungsi ‘quiver’ untuk menggambar medan vektor pada persoalan yang lain.

3. Cobalah memanfaatkan perintah Matlab ‘ode45’ untuk merekonstruksi Gambar 2.3 agar Anda mendapatkan ide yang kuat tentang solusi PDB yang bersangkutan.

Contoh, ketikkan

```
hold on
ode45(fx,[0 10],[0.1 1.8])
```

untuk mendapatkan solusi tipikal dengan $x(0) = 0.1$ dan $x(0) = 1.8$.

Contoh PDB skalar lainnya adalah pertumbuhan eksponensial suatu spesies yang memenuhi hubungan

$$\dot{N} = rN \quad (2.4)$$

yang solusinya berbentuk $N(t) = N_0 e^{rt}$ untuk syarat awal $N(0) = N_0$. Jika $r < 0$ maka fenomena yang teramati adalah peluruhan eksponensial. Tentu saja model ini tidak realistis untuk waktu yang terus meningkat. Karena terbatasnya sumberdaya (makanan), habitat dengan populasi yang besar tentu tidak dapat menyokong daya hidup individu secara terus menerus. Model pertumbuhan eksponensial di atas kemudian dikoreksi menjadi *persamaan logistik* atau yang dikenal juga dengan *persamaan Verhulst*,

$$\dot{N} = rN \left(1 - \frac{N}{K} \right), \quad (2.5)$$

di mana r adalah laju pertumbuhan intrinsic dan K adalah kapasitas pendukung (*carrying capacity*) yang menyatakan jumlah maksimum populasi yang bisa disokong sumberdaya yang tersedia. Jika $K \equiv N$ maka populasi akan mengalami saturasi menuju nilai tertentu dengan laju pertumbuhan nol.

2.2. Sistem Persamaan Differensial

Seringkali kita berhadapan dengan keharusan mengikuti lebih dari satu faktor yang berevolusi terhadap waktu secara bersamaan dan saling berkaitan/berinteraksi. Misalnya sudut simpangan dan kecepatan sudut pada gerak pendulum atau mangsa dan predator pada suatu sistem biologi. Untuk kasus seperti ini, persamaan-persamaan differensialnya membentuk suatu sistem persamaan differensial yang harus dipecahkan secara simultan.

2.2.1 Model Interaksi Dua Populasi

Sistem interaksi dua populasi di bawah ini adalah contoh sistem persamaan differensial biasa:

$$\begin{aligned} \dot{x}(t) &= \alpha x + \beta xy \\ \dot{y}(t) &= \gamma y + \delta xy \end{aligned} \quad (2.6)$$

dengan $x(t)$ dan $y(t)$ mewakili jumlah dua populasi. αx dan γy menggambarkan bagaimana populasi x dan y bertambah atau menyusut terhadap waktu dalam keadaan tidak saling berinteraksi. Suku-suku βxy dan δxy menggambarkan bagaimana interaksi kedua spesies mempengaruhi masing-masing populasi. Berdasarkan tanda koefisien α , β , γ dan δ maka kita dapatkan tiga jenis model interaksi seperti berikut:

Tabel 2.1. Klasifikasi model interaksi dua populasi

α	β	γ	δ	Model interaksi
+	+	+	-	Predator (x) – Mangsa (y)
+	+	-	-	
-	+	+	-	
-	+	-	-	
+	+	+	+	Simbiosis mutualisma
+	+	-	+	
-	+	-	+	
+	-	+	-	Kompetisi
+	-	-	-	
-	-	-	-	

Dapatlah disimpulkan bahwa tanda β dan δ menentukan jenis interaksi kedua jenis populasi.

Dengan memperhatikan persamaan (2.4), (2.5) dan Tabel 2.1 maka tidaklah sulit untuk menyatakan dan memilih tanda koefisien-koefisien untuk **model kompetisi** antara dua populasi N_1 dan N_2

$$\begin{aligned}\dot{N}_1 &= r_1 N_1 \left(1 - \frac{N_1}{K_1}\right) - \frac{\beta_{12}}{K_1} N_1 N_2 \\ \dot{N}_2 &= r_2 N_2 \left(1 - \frac{N_2}{K_2}\right) - \frac{\beta_{21}}{K_2} N_1 N_2\end{aligned}\quad (2.6)$$

2.2.2 Non-Dimensionalisasi Sistem Persamaan Differensial

Beberapa model dapat saja memiliki banyak sekali parameter bebas. Perangkat analisis matematik selalu memungkinkan untuk mereduksi jumlah parameter bebas yang harus dipecahkan tanpa harus kehilangan karakteristik dinamika sistem. Proses reduksi parameter ini disebut *non-dimensionalisasi* yang bermanfaat juga mereduksi kompleksitas model secara signifikan, seperti pada contoh di bawah ini.

Model Predator-Mangsa Lotka-Volterra yang digagas Vito Volterra (1860–1940) dan Alfred James Lotka (1880–1949) menyatakan sistem interaksi antara spesies *predator* (pemangsa) dan *prey* (mangsa) dalam satu ekosistem. Model ini mengasumsikan bahwa jika tidak terdapat predator (P) maka populasi mangsa (M) akan meningkat secara eksponensial. Sedangkan jika tidak ada mangsa, predator akan mengalami kematian karena kelaparan dan populasinya menyusut secara eksponensial. Interaksi antara P dan M akan menyebabkan populasi P bertambah sedangkan M berkurang. Model matematisnya adalah

$$\begin{aligned}\dot{M} &= bM - sMP \\ \dot{P} &= -mP + esMP\end{aligned}\quad (2.7)$$

b adalah laju kelahiran mangsa, m adalah laju kematian predator, sedangkan s adalah efisiensi pencarian mangsa oleh predator dan e adalah efisiensi pengubahan makanan (mangsa) menjadi predator baru (melalui kelahiran predator baru).

Persamaan Lotka-Volterra mengandung empat parameter yaitu b , s , d dan e . Jumlahnya dapat direduksi dengan memperkenalkan besaran tak berdimensi berikut:

$$h = \frac{Hes}{d}, \quad p = \frac{Ps}{b}, \quad \tau = \sqrt{bd}t, \quad \rho = \sqrt{\frac{b}{d}}, \quad \text{maka persamaan (2.7) tereduksi menjadi}$$

$$\begin{aligned}\frac{dh}{d\tau} &= \rho h(1-p) \\ \frac{dp}{d\tau} &= -\frac{1}{\rho} p(1-h)\end{aligned}\quad (2.8)$$

yang hanya memiliki dua parameter yaitu ρ dan h .

2.3. Analisis Bidang-Fasa: Sistem Linier

2.3.1. Sistem Linier Khusus dengan Nilai Eigen Real

Misalkan terdapat suatu sistem linear sederhana berbentuk

$$\begin{aligned}\dot{x}_1 &= \lambda_1 x_1 \\ \dot{x}_2 &= \lambda_2 x_2\end{aligned}\tag{2.9}$$

dengan titik kesetimbangan $(\bar{x}_1, \bar{x}_2) = (0, 0)$. Bila ditulis dalam bentuk matriks maka persamaan di atas menjadi

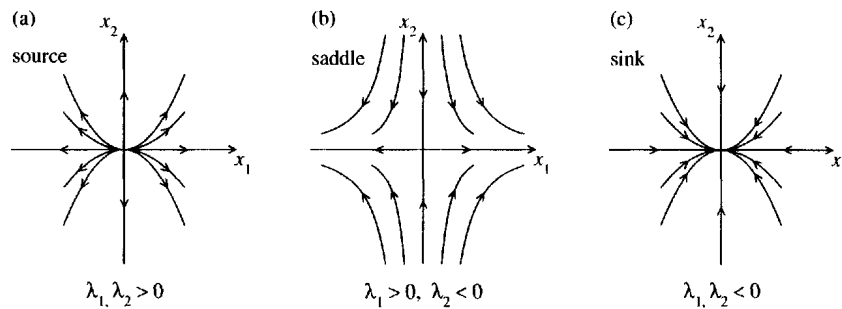
$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.\tag{2.10}$$

λ_1 dan λ_2 adalah nilai eigen dari matriks

$$A = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.\tag{2.11}$$

Solusi dari persamaan (2.9) adalah

$$\begin{aligned}x_1(t) &= x_1(0)e^{\lambda_1 t} \\ x_2(t) &= x_2(0)e^{\lambda_2 t}\end{aligned}\tag{2.12}$$



Gambar 2.4. Potret fasa (phase portrait) kualitatif untuk sistem persamaan (2.9) bergantung pada tanda λ_1 dan λ_2 . (a) $\lambda_1, \lambda_2 > 0$; (b) $\lambda_1 > 0$ dan $\lambda_2 < 0$ (c) $\lambda_1, \lambda_2 < 0$. Pada (a) dan (c) diasumsikan bahwa $\lambda_2 > \lambda_1$.

Plot kurva-kurva parametric $(x_1(t), x_2(t))$ untuk nilai-nilai awal $(x_1(0), x_2(0))$ yang berbeda diperlihatkan pada Gambar 2.4. Terlihat bahwa terdapat tiga potret fasa yang berbeda bergantung tanda λ_1 dan λ_2 .

Kasus (a):

Jika kedua nilai eigen λ_1 dan λ_2 positif, maka semua solusi bersifat divergen dari keadaan tunak atau titik kesetimbangan $(0, 0)$ seperti yang diperlihatkan Gambar 2.4(a). Titik keadaan tunak pada kasus ini disebut *sumber* atau *source* atau *simpul tak-stabil*.

Kasus (b):

Jika nilai eigen memiliki tanda berlawanan, $\lambda_1 > 0$ dan $\lambda_2 < 0$, maka $x_1(t)$ akan meningkat secara eksponensial sedangkan $x_2(t)$ akan meluruh. Semua solusi akan mendekati sumbu x_1 seperti pada Gambar 2.4(b). Titik kesetimbangan $(0, 0)$ disebut titik pelana atau *saddle*.

Kasus (c):

Kedua nilai eigen bertanda negatif. Solusi konvergen menuju keadaan tunak $(0, 0)$ sebagaimana ditunjukkan pada Gambar 2.4 (c). Titik tunaknya disebut *sink* atau *simpul stabil*.

2.3.2. Sistem linier khusus dengan nilai eigen kompleks

Misalkan terdapat suatu sistem persamaan linier

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (2.13)$$

Untuk $\beta \neq 0$, sistem mempunyai titik awal, $(0, 0)$, yang merupakan keadaan tunak. Koefisien

matriks $\mathbf{A} = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}$ mempunyai dua nilai eigen konjugat kompleks yaitu

$$\lambda_1 = \alpha + \beta i \text{ dan } \lambda_2 = \alpha - \beta i \quad (2.14)$$

Persamaan (2.13) mempunyai solusi

$$x^{(1)}(t) = e^{\alpha t} \begin{pmatrix} \cos \beta t \\ -\sin \beta t \end{pmatrix}, \quad x^{(2)}(t) = e^{\alpha t} \begin{pmatrix} \sin \beta t \\ \cos \beta t \end{pmatrix}. \quad (2.15)$$

Prinsip superposisi sistem linier memberikan solusi persamaan (2.13) menjadi berbentuk

$$x(t) = c_1 x^{(1)}(t) + c_2 x^{(2)}(t) = a e^{\alpha t} \begin{pmatrix} \cos(\beta t + \phi) \\ -\sin(\beta t + \phi) \end{pmatrix} \quad (2.17)$$

atau

$$\begin{aligned} x_1(t) &= a e^{\alpha t} \cos(\beta t + \phi), \\ x_2(t) &= -a e^{\alpha t} \sin(\beta t + \phi), \end{aligned} \quad (2.18)$$

di mana α dan β ditentukan oleh nilai awal $(x_1(0), x_2(0))$.

Potret fasa dan trajektori solusi persamaan (2.18) dapat dikelompokkan menjadi tiga kasus berbeda:

Kasus (a):

$\alpha = 0$, kedua nilai eigen bersifat imajiner. Semua solusinya periodik, dan semua trajektori merupakan orbit tertutup yang mengelilingi keadaan tunak $(0, 0)$, seperti yang ditunjukkan Gambar 2.5 (a). Keadaantunak tersebut disebut *center* atau pusat.

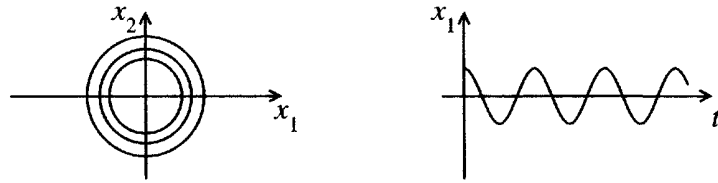
Kasus (b):

$\alpha > 0$, kedua nilai eigen merupakan bilangan real positif. Fungsi eksponensial $e^{\alpha t}$ meningkat untuk $t > 0$. Semua trajektori spiral bergerak dari keadaan tunak $(0, 0)$ seperti ditunjukkan Gambar 2.5 (b). Keadaan tunak ini disebut spiral tak-stabil atau *spiral source*.

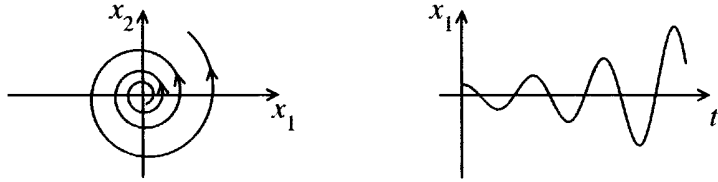
Kasus (c):

$\alpha < 0$, kedua nilai eigen adalah bilangan real negatif. Fungsi eksponensial $e^{\alpha t}$ menurun untuk $t < 0$. Semua trajektori spiral bergerak menuju keadaan tunak $(0, 0)$ seperti pada Gambar 2.5(c). Keadaan tunak ini disebut spiral stabil atau *spiral sink*.

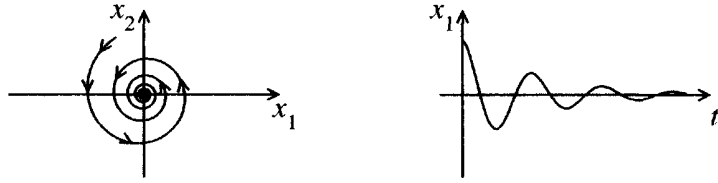
(a) $\alpha = 0$, center



(b) $\alpha > 0$, spiral tak-stabil



(c) $\alpha < 0$, spiral stabil



Gambar 2.5. tiga kasus system bergantung nilai parameter α . (a) $\alpha = 0$; (b) $\alpha > 0$; (c) $\alpha < 0$. Gambar sebelah kiri menunjukan portrait fasa. Gambar sebelah kanan menunjukan trajektori solusi khusus untuk $x_1(t)$. Diasumsikan $\beta > 0$ ketika membuat sketsa (a) - (c), sehingga spiral bergerak searah jarum jam.

2.3.3. Sistem Linier secara Umum

Misalkan Diberikan suatu sistem linier dalam bentuk yang umum,

$$\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}. \quad (2.19)$$

Selanjutnya, bila kita transformasi (x_1, x_2) menjadi (y_1, y_2) seperti di bawah ini,

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{P}^{-1} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad (2.20)$$

di mana \mathbf{P} adalah matriks 2×2 yang dapat balik (*invertible*), maka $y = (y_1, y_2)$ memenuhi sistem persamaan berikut

$$\frac{d}{dt} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{B} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad (2.21)$$

dengan $\mathbf{B} = \mathbf{P}^{-1} \mathbf{A} \mathbf{P}$. Matriks \mathbf{B} serupa dengan \mathbf{A} karena mempunyai nilai eigen yang identik sehingga sistem persamaan (2.19) dan (2.21) mempunyai potret fasa yang sama.

Dari prinsip-prinsip aljabar linier diketahui bahwa jika \mathbf{A} mempunyai dua nilai eigen real yang berbeda λ_1 dan λ_2 serta $\lambda_1 \neq \lambda_2$, maka kita dapat memilih \mathbf{P} sedemikian sehingga

$$\mathbf{B} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}. \quad (2.22)$$

Jika \mathbf{A} memiliki dua nilai eigen konjugat kompleks $\lambda_1 = \lambda_2 = \alpha + \beta i$, maka kita dapat memilih \mathbf{P} sedemikian sehingga

$$\mathbf{B} = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}. \quad (2.23)$$

Dapatlah bisa disimpulkan bahwa potret fasa persamaan (2.19) akan sama dengan sistem persamaan (2.9) atau (2.13).

Latihan 2.1

Soal 1.

Perhatikan sistem linier berikut

$$\begin{aligned} \dot{x} &= 2x - 2y \\ \dot{y} &= 2x - 3y \end{aligned} \quad (2.24)$$

yang dalam notasi matriks dapat ditulis dengan

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 & -2 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} 2 & -2 \\ 2 & -3 \end{pmatrix} \quad (2.25)$$

a. Gunakan $\det(\lambda \mathbf{I} - \mathbf{A}) = 0$ dan $(\lambda \mathbf{I} - \mathbf{A}) \mathbf{v} = 0$ untuk membuktikan bahwa nilai eigen dan vektor eigen untuk \mathbf{A} adalah $\lambda_1 = 1, v_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ dan $\lambda_2 = -2, v_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

b. Gunakan Matlab dan cobalah lakukan perintah a dengan menuliskan:

```
A=[2 -2; 2 -3];
[V, E]=eig(A)
V1=V(:,1)/min(V(:,1));
V2=V(:,2)/min(V(:,2));
V=[V1 V2]
```

c. Kedua nilai eigen dari \mathbf{A} adalah real dan berbeda. Jika digunakan vector eigen v_1 dan v_2 sebagai kolom matriks \mathbf{P} , diperoleh transformasi

$$\mathbf{P} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{P}^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad (2.26)$$

sehingga

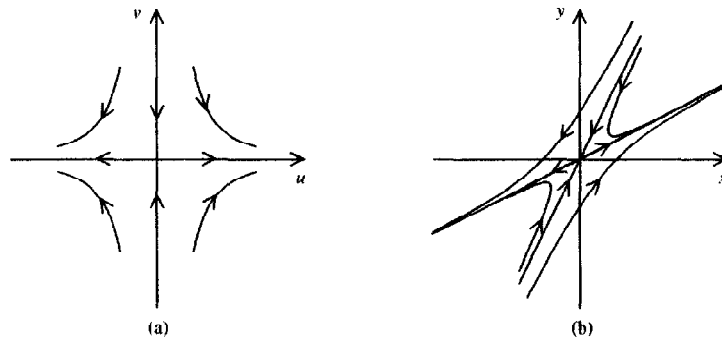
$$\mathbf{B} = \mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \frac{1}{3} \begin{pmatrix} 1 & 0 \\ 0 & -2 \end{pmatrix}. \quad (2.27)$$

Nyatakan secara eksplisit solusi untuk sistem persamaan baru tertransformasi

$$\frac{d}{dt} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \quad (2.28)$$

d. Nyatakanlah solusi persamaan (2.24) secara eksplisit dengan memanfaatkan hubungan

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{P} \begin{pmatrix} u \\ v \end{pmatrix}.$$



Gambar 2.6. Potret fasa dari (a) persamaan (2.8) dan (b) persamaan (2.24).

Potret fasa persamaan (2.28) ditunjukkan oleh gambar 2.6(a), untuk persamaan (2.24)

ditunjukkan oleh gambar 2.6(b). Transformasi \mathbf{P} memetakan arah tak-stabil $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ dari

persamaan (2.28) ke arah tak-stabil $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$ persamaan (2.24). Demikian pula arah stabil $\begin{pmatrix} 0 \\ -2 \end{pmatrix}$

pada persamaan (2.28) dipetakan ke stable direction $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ persamaan (2.24). Potret fasa pada

Gambar 2.6(b) merupakan hasil rotasi dan kompresi dari Gambar 2.9(a).

e. Reproduksi Gambar 2.6(a) dengan menuliskan *code* berikut :

```
% Membuat medan vektor
u=linspace(-5,5,21);
v=linspace(-5,5,21);
[U V]=meshgrid(u,v);
fu=inline('u','u','v');
```

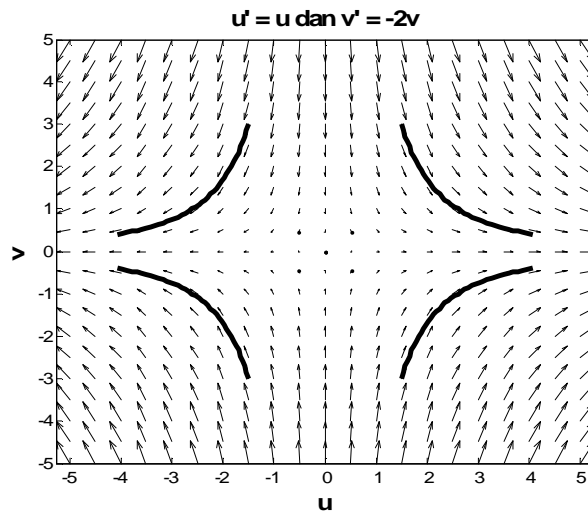
```

fv=inline('-2*v','u','v');
FU=fu(U,V);
FV=fv(U,V);
quiver(U,V,FU,FV,'k')
axis tight
xlabel('u','fontsize',14,'fontweight','b')
ylabel('v','fontsize',14,'fontweight','b')
title('u' = u dan v' = -2v','fontsize',14,'fontweight','b')
hold on

% Sistem persamaan
g = @(t, x) [x(1); -2*x(2)];
% Memplot beberapa solusi untuk syarat awal u(0) dan v(0)
% dalam rentang waktu 0 hingga 1 detik
[t, xa] = ode45(g, [0:1], [1.5 3]);
plot(xa(:,1),xa(:,2),'LineWidth',3,'Color',[0 0 0])
[t, xa] = ode45(g, [0:1], [1.5 -3]);
plot(xa(:,1),xa(:,2),'LineWidth',3,'Color',[0 0 0])
[t, xa] = ode45(g, [0:1], [-1.5 3]);
plot(xa(:,1),xa(:,2),'LineWidth',3,'Color',[0 0 0])
[t, xa] = ode45(g, [0:1], [-1.5 -3]);
plot(xa(:,1),xa(:,2),'LineWidth',3,'Color',[0 0 0])

```

Sehingga didapat gambar berikut :



Gambar 2.7. Hasil reproduksi Gambar 2.6(a)

f. Reproduksi Gambar 2.6(b) dengan memodifikasi *code* di atas.

Contoh 2.1

Misalkan terdapat sistem persamaan

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & -4 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \quad (2.29)$$

Nilai eigen matriks tersebut adalah $\lambda_1 = -1 + 2i$, dan $\lambda_2 = -1 - 2i$. Sedangkan vektor eigennya adalah

$$\zeta_1 = \begin{pmatrix} -4 \\ -2+2i \end{pmatrix} \quad \text{dan} \quad \zeta_2 = \zeta_1^*. \quad (2.30)$$

Jika kita tulis $\zeta_1 = \phi + i\psi$ dengan vektor real ϕ dan ψ serta matriks transformasi $\mathbf{P} = (\phi \psi)$,

$$\mathbf{P} = \begin{pmatrix} -4 & 0 \\ -2 & 2 \end{pmatrix}, \quad (2.31)$$

dengan matriks invers

$$\mathbf{P}^{-1} = \frac{1}{4} \begin{pmatrix} -1 & 0 \\ -1 & 2 \end{pmatrix}. \quad (2.32)$$

Dengan menggunakan transformasi \mathbf{P} , diperoleh matriks,

$$\mathbf{B} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{pmatrix} -1 & 2 \\ -2 & -1 \end{pmatrix}, \quad (2.33)$$

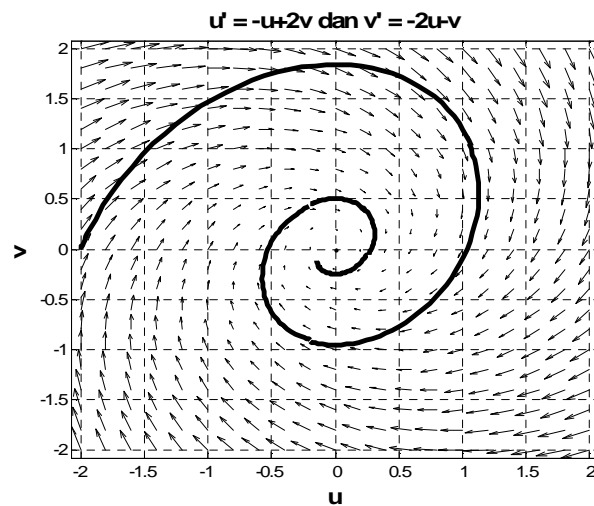
dan sistem tertransformasinya mempunyai bentuk persamaan (2.13):

$$\frac{d}{dt} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -1 & 2 \\ -2 & -1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}. \quad (2.34)$$

Solusi umumnya adalah

$$\begin{pmatrix} u \\ v \end{pmatrix}(t) = e^{-t} \left(c_1 \begin{pmatrix} -\sin(2t) \\ \cos(2t) \end{pmatrix} + c_2 \begin{pmatrix} \cos(2t) \\ \sin(2t) \end{pmatrix} \right). \quad (2.35)$$

Solusi tersebut menggambarkan osilasi di sekitar $(0, 0)$ dengan frekuensi π^{-1} , dengan amplitudo yang menurun secara eksponensial sebesar e^{-t} . Solusi konvergen menuju $(0, 0)$ dan keadaan tunak $(0, 0)$ adalah spiral stabil. Medan vektor dan kurva solusinya diperlihatkan pada Gambar 2.8.



Gambar 2.8. Medan vektor dan solusi tipikal untuk persamaan (2.34)

Dari semua kasus di atas, solusinya hanya konvergen menuju keadaan tunak $(0, 0)$ ketika kedua nilai eigen $\lambda_1, \lambda_2 < 0$ (origin merupakan simpul stabil), atau ketika bagian real nilai eigen $\alpha < 0$ (origin merupakan spiral stabil). Keadaan tunak (titik kesetimbangan) pada kasus ini bersifat stabil asimptotis.

Latihan 2.2

Buatlah program kecil untuk menggambar ulang Gambar 2.8.

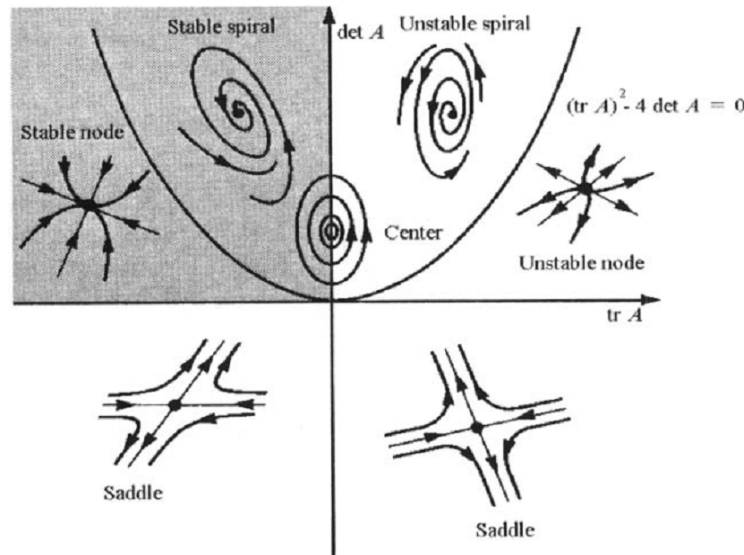
Terlihat bahwa titik-titik kesetimbangan sistem linier dapat diklasifikasikan berdasarkan nilai eigen dari koefisien matriks berikut,

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}. \quad (2.36)$$

Untuk matriks 2×2 , terdapat cara alternative untuk mencari nilai eigen, yaitu dengan memanfaatkan definisi *trace*, $\text{tr } \mathbf{A} = a + d$, dan determinan, $\det \mathbf{A} = ad - bc$. *Trace* juga diketahui sebagai penjumlahan nilai eigen, $\text{tr } \mathbf{A} = \lambda_1 + \lambda_2$, sedangkan determinan merupakan perkaliannya, $\det \mathbf{A} = \lambda_1 \lambda_2$. Sehingga *trace* dan determinan dapat digunakan untuk menghitung nilai eigen.

$$\lambda_{1,2} = \frac{\text{tr } \mathbf{A}}{2} \pm \frac{1}{2} \sqrt{(\text{tr } \mathbf{A})^2 - 4 \det \mathbf{A}}. \quad (2.37)$$

Berdasarkan persamaan (2.37), suatu keadaan tunak akan stabil asimptotis $\text{tr } \mathbf{A} < 0$ (Jika sebaliknya maka paling tidak salah satu bagian real nilai eigennya positif). Jika $\text{tr } \mathbf{A} < 0$ maka diskriminan, $(\text{tr } \mathbf{A})^2 - 4 \det \mathbf{A}$, akan negatif atau lebih kecil daripada $(\text{tr } \mathbf{A})^2$. Sehingga bagian real nilai eigen selalu negatif dan $(0, 0)$ stabil asimptotis. Dapatlah disimpulkan bahwa jika titik kesetimbangan $(0, 0)$ stabil asimptotis, maka semua nilai eigen dari \mathbf{A} memiliki bagian real negatif serta $\det \mathbf{A} = ad - bc > 0$ dan $\text{tr } \mathbf{A} = a + d < 0$.



Gambar 2.9. Karakteristik keadaan tunak sistem persamaan linier (2.19).

Gambar 2.9 adalah ilustrasi dari ringkasan berikut ini:

1. Kasus $\det \mathbf{A} < 0$. Dalam hal ini $(\text{tr } \mathbf{A})^2 - 4 \det \mathbf{A} > 0$. Dari persamaan (2.37), diperoleh satu nilai eigen positif dan negatif, $\lambda_1 > 0$ dan $\lambda_2 < 0$. Sehingga $(0, 0)$ merupakan titik pelana (*saddle point*). Untuk λ_1 solusinya meningkat sebesar $e^{\lambda_1 t}$ dalam arah vektor eigen Φ_1 , sedangkan pada λ_2 solusinya menurun sebesar $e^{\lambda_2 t}$ dalam arah vektor eigen Φ_2 .
2. Kasus $\det \mathbf{A} > 0$, $\text{tr } \mathbf{A} < 0$. Jika $(\text{tr } \mathbf{A})^2 < 4 \det \mathbf{A}$ (di atas parabola dalam Gambar 2.9), maka λ_1, λ_2 merupakan nilai eigen konjugat kompleks dengan bagian real $(\text{tr } \mathbf{A})/2$, dan $(0, 0)$ merupakan spiral stabil. Jika $(\text{tr } \mathbf{A})^2 > 4 \det \mathbf{A}$ (di bawah parabola), maka λ_1, λ_2 merupakan dua bilangan real dengan tanda yang sama dan $(0, 0)$ merupakan simpul stabil.
3. Kasus $\det \mathbf{A} > 0$ dan $\text{tr } \mathbf{A} > 0$. Bergantung pada tanda $(\text{tr } \mathbf{A})^2 - 4 \det \mathbf{A}$, kita akan peroleh spiral tak-stabil simpul tak-stabil.
4. Kasus $\det \mathbf{A} > 0$, $\text{tr } \mathbf{A} = 0$, maka akan diperoleh *center* atau pusat.

2.4. Sistem Persamaan Non-Linier dan Linierisasi

Misalkan terdapat suatu sistem persamaan non- linier berikut

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, x_2) \\ \dot{x}_2 &= f_2(x_1, x_2),\end{aligned}\tag{2.38}$$

di mana f_1 dan f_2 merupakan fungsi yang differensiabel.

Tiap pasangan (\bar{x}_1, \bar{x}_2) yang sedemikian memberikan $f_1(\bar{x}_1, \bar{x}_2) = f_2(\bar{x}_1, \bar{x}_2) = 0$, merupakan equilibrium atau keadaan tunak.

Untuk sistem linier, solusi dapat konvergen menuju $(0, 0)$, divergen dari $(0, 0)$, atau solusi dapat saling berdekatan seperti dalam kasus *center*.

Untuk mengeneralisir perilaku keadaan tunak untuk sistem dinamik non-linier, perlu diperhatikan beberapa definisi di bawah ini:

- (a) Keadaan tunak (\bar{x}_1, \bar{x}_2) disebut stabil (*Lyapunov stable*) jika solusi dengan syarat awal yang berdekatan dengan keadaan tersebut akan tetap berdekatan. Artinya (\bar{x}_1, \bar{x}_2) stabil jika untuk semua $\varepsilon > 0$, terdapat $\delta > 0$ sehingga solusi-solusi terhadap data awal (x_1^0, x_2^0) dengan $\|(x_1^0, x_2^0) - (\bar{x}_1, \bar{x}_2)\| < \delta$ memenuhi $\|(x_1(t), x_2(t)) - (\bar{x}_1, \bar{x}_2)\| < \varepsilon$ untuk semua waktu $t > 0$. $\|\cdot\|$ menunjukkan norm vektor Euclidean.
- (b) Keadaan tunak (\bar{x}_1, \bar{x}_2) yang tidak stabil disebut tak-stabil (paling tidak salah satu solusinya diverge dari (\bar{x}_1, \bar{x}_2)).
- (c) Keadaan tunak (\bar{x}_1, \bar{x}_2) disebut stabil asimptotis jika (\bar{x}_1, \bar{x}_2) stabil dan semua solusi dekat (\bar{x}_1, \bar{x}_2) konvergen ke (\bar{x}_1, \bar{x}_2) .

Stabilitas keadaan tunak (\bar{x}_1, \bar{x}_2) dapat ditentukan dengan linierisasi persamaan (2.38) yang prosesnya sama dengan linierisasi sistem waktu-diskret.

Jika dianggap

$$\begin{aligned} x_1(t) &= \bar{x}_1 + z_1(t) \\ x_2(t) &= \bar{x}_2 + z_2(t), \end{aligned} \quad (2.39)$$

di mana $z_1(t)$ dan $z_2(t)$ diasumsikan kecil, sehingga dapat dianggap sebagai perturbasi dari keadaan tunak. Jika kita tulis $\bar{x} = (\bar{x}_1, \bar{x}_2)$ dan $z = (z_1, z_2)$, maka ekspansi Taylor $f = (f_1, f_2)$ di sekitar (\bar{x}_1, \bar{x}_2) dapat dinyatakan dengan

$$f(\bar{x} + z) = f(\bar{x}) + Df(\bar{x}) \cdot z + \text{suku berorde lebih tinggi} \quad (2.40)$$

di mana

$$Df(\bar{x}_1, \bar{x}_2) := \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \left(\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{array} \right) \bigg|_{(x_1, x_2) \approx (\bar{x}_1, \bar{x}_2)} \quad (2.41)$$

mengandung turunan parsial dari f yang dievaluasi pada (\bar{x}_1, \bar{x}_2) . Matriks $Df(\bar{x}_1, \bar{x}_2)$ disebut matriks Jacobian dari f pada (\bar{x}_1, \bar{x}_2) .

Ekspansi Taylor persamaan (2.39) dilakukan dengan mengabaikan suku-suku berorde lebih tinggi. Karena $\dot{x}_1 = \frac{d}{dt}(\bar{x}_1 + z_1(t)) = \dot{z}_1$ dan $\dot{x}_2 = \dot{z}_2$ serta $f(\bar{x}) = 0$, maka diperoleh sistem linier yang mengontrol dinamika perturbasi (z_1, z_2) :

$$\frac{d}{dt} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}. \quad (2.42)$$

(\bar{x}_1, \bar{x}_2) disebut hiperbolik jika semua nilai eigen dari Jacobian $Df(\bar{x}_1, \bar{x}_2)$ mempunyai bagian real tak-nol.

2.5. Analisis Kualitatif Model Interaksi Populasi secara Umum

Contoh interaksi dua spesies yang akan dibahas di sini adalah model-model predator-mangsa, simbiosis-mutualisma, serta kompetisi. Dalam semua kasus terlebih dahulu harus ditentukan keadaan-keadaan tunaknya serta linierisasinya.

Persamaan (2.6) dituliskan dalam notasi vektor:

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \end{pmatrix}, \quad (2.43)$$

dengan $f_1(x, y) = \alpha x + \beta xy$ dan $f_2(x, y) = \gamma y + \delta xy$. Untuk menemukan titik-titik pembuat nol arah x (x -nullcline), n_x maka diset $f_1 = 0$ sehingga

$$n_x = \left\{ (x, y) \mid x = 0, \text{ atau } y = -\frac{\alpha}{\beta} \right\}. \quad (2.44)$$

Dengan analogi yang sama maka y -nullcline:

$$n_y = \left\{ (x, y) \mid y = 0, \text{ atau } x = -\frac{\gamma}{\delta} \right\}. \quad (2.45)$$

Keadaan tunak (\bar{x}_1, \bar{x}_2) merupakan titik potong antara x -nullcline dengan y -nullcline serta memenuhi $f_1(\bar{x}, \bar{y}) = 0$ dan $f_2(\bar{x}, \bar{y}) = 0$. Diperoleh dua keadaan tunak yaitu

$$P_1 = (0, 0) \text{ dan } P_2 = \left(-\frac{\gamma}{\delta}, -\frac{\alpha}{\beta} \right). \quad (2.46)$$

Linierisasi dari persamaan (2.43) diberikan oleh

$$\frac{d}{dt} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = Df(\bar{x}, \bar{y}) \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \quad (2.47)$$

dan

$$Df = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} \alpha + \beta y & \beta x \\ \delta y & \gamma + \delta x \end{pmatrix}. \quad (2.49)$$

Matriks Jacobian di atas dievaluasi pada dua keadaan tunak yaitu P_1 dan P_2 . Untuk P_1 diperoleh,

$$Df(0, 0) = \begin{pmatrix} \alpha & 0 \\ 0 & \gamma \end{pmatrix}, \quad (2.50)$$

yang mempunyai dua nilai eigen $\lambda_1 = \alpha$ dan $\lambda_2 = \gamma$. Untuk P_2 , diperoleh,

$$Df\left(-\frac{\gamma}{\delta}, -\frac{\alpha}{\beta}\right) = \begin{pmatrix} 0 & -\frac{\beta\gamma}{\delta} \\ -\frac{\alpha\delta}{\beta} & 0 \end{pmatrix} = \mathbf{A}. \quad (2.51)$$

karena $\text{tr } \mathbf{A} = 0$ dan $\det \mathbf{A} = -\alpha\gamma$, persamaan (2.37) memberikan nilai eigen

$$\lambda_{1/2} = \pm \sqrt{\alpha\gamma}. \quad (2.52)$$

Identifikasi jenis keadaan tunak lebih jauh memerlukan informasi mengenai tanda dari parameter α, β, γ dan δ . Berikut adalah analisis untuk tiga kasus khusus.

Kasus $(-+ + -)$: Model predator-mangsa

Diasumsikan bahwa $\alpha < 0$, $\beta > 0$, $\gamma > 0$ dan $\delta < 0$. Berdasarkan persamaan (2.50) dapat dilihat bahwa satu nilai eigennya adalah negatif ($\lambda_1 = \alpha < 0$) dan nilai eigen yang lainnya adalah positif ($\lambda_2 = \gamma > 0$). Maka $P_1 = (0, 0)$ adalah pelana.

Sebelum berlanjut pada $P_2 = \left(-\frac{\gamma}{\delta}, -\frac{\alpha}{\beta}\right)$, haruslah dipastikan bahwa secara biologis relevan yaitu $-\frac{\gamma}{\delta} > 0$ dan $-\frac{\alpha}{\beta} > 0$. Karena γ, δ dan α, β mempunyai tanda yang berlawanan, jelaslah syarat di atas terpenuhi. Pada persamaan (2.52), perkalian $\alpha\gamma < 0$, sehingga nilai eigennya bilangan imajiner murni, $\lambda_{1/2} = \pm i\sqrt{|\alpha\gamma|}$, sehingga $\left(-\frac{\gamma}{\delta}, -\frac{\alpha}{\beta}\right)$ adalah *center* atau pusat.

Namun demikian, P_2 tidak hiperbolik, sehingga P_2 mungkin pusat, spiral stabil, atau spiral tak-stabil. Informasi yang hilang didapat dengan mengintegralkan

$$\frac{dy}{dx} = \frac{dy/dt}{dx/dt} = \frac{y(\gamma + \delta x)}{x(\alpha + \beta y)} \quad (2.53)$$

melalui teknik separasi variabel yang hasilnya

$h(x, y) = \alpha \ln y + \beta y - (\gamma \ln x + \delta x) = \text{konstanta}$. Potret fasanya mendekati kurva tertutup yang bersesuaian dengan $h(x, y) = \text{konstanta}$ seperti pada Gambar 2.10 (c). P_2 dinyatakan sebagai pusat non-linier. Medan vektor dan potret fasa untuk kasus $(-++-)$ ditunjukkan pada Gambar 2.10. Teramati osilasi predator-mangsa terjadi di antara periode populasi tinggi dan rendah.

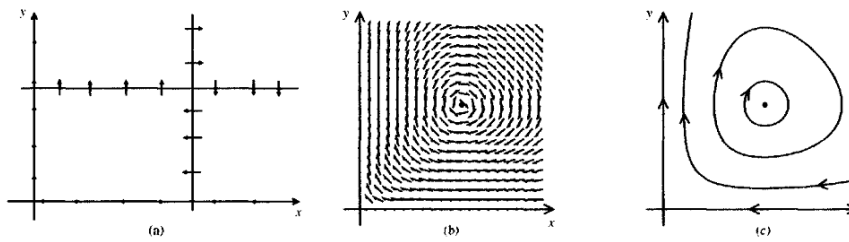
Kasus $(-++-)$: Mutualisme dua spesies yang tidak bisa bertahan sendiri

($\alpha < 0$ dan $\gamma < 0$). Nilai eigen dari $Df(0,0)$ adalah $\alpha < 0$ dan $\gamma < 0$ sehingga $(0, 0)$ merupakan simpul stabil. Karena juga $-\frac{\gamma}{\delta} > 0$ dan $-\frac{\alpha}{\beta} > 0$ maka P_2 relevan secara biologis. Perkalian $\alpha\gamma > 0$, sehingga P_2 merupakan pelana. Medan vektor dan potret fasa diperlihatkan pada Gambar 2.11.

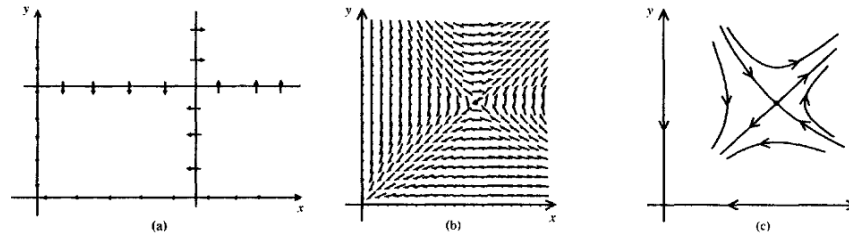
Berdasarkan potret fasa terlihat bahwa jika populasi awal untuk x dan y cukup besar, maka kedua populasi akan tumbuh. Jika salah satu jumlah awalnya terlalu kecil, maka kedua spesies menjadi punah (konvergen ke nol).

Kasus $(+---)$: Model Kompetisi

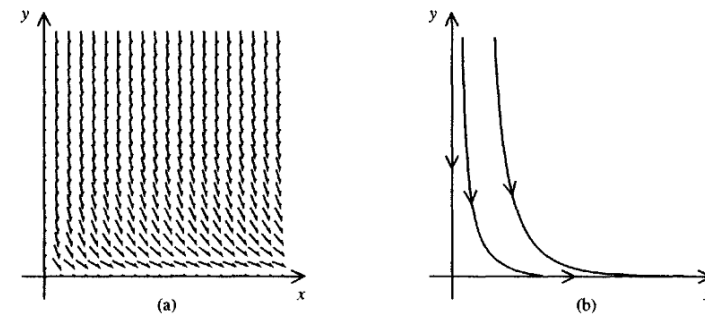
Dalam kasus ini $(0, 0)$ merupakan pelana dan P_2 tidak relevan secara biologis ($-\frac{\gamma}{\delta} < 0$). Medan vektor dan potret fasa untuk kasus ini diperlihatkan pada gambar 2.12. Populasi y akan punah ketika populasi x tumbuh tanpa adanya kompetisi.



Gambar 2.10. (a) Nullclines; (b) direction field; dan (c) phase portrait untuk model dua-spesies, persamaan (2.6) dengan tanda $(-+-)$ (predator-prey).



Gambar 2.11. (a) Nullclines; (b) direction field; dan (c) phase portrait untuk model dua-spesies, persamaan (2.6) dengan tanda $(-+-)$ (mutualisme).



Gambar 2.12. (a) direction field dan (b) phase portrait untuk model dua-spesies, persamaan (2.6) dengan tanda $(+--)$ (kompetisi).



III. CONTOH-CONTOH SISTEM DINAMIK

Untuk mempertajam gagasan tentang sistem dinamik, berikut dibahas beberapa contoh persoalan yang terkait dengan sistem tersebut.

Contoh 3.1

Model Predator-Mangsa.

Diketahui:

- $x(t)$ merupakan populasi mangsa terhadap waktu
- $y(t)$ merupakan populasi pemangsa terhadap waktu
- Asumsi:
 - Bila tidak terdapat pemangsa maka populasi mangsa akan mendekati *carrying capacity*-nya (model pertumbuhan Verhulst) seperti yang dimodelkan dalam persamaan logistik

$$\dot{x} = ax \left(1 - \frac{x}{K}\right),$$
 dimana $a > 0$
 - Pemangsa tak dapat hidup tanpa mangsa, sehingga dapat diambil model peluruhan untuk pemangsa

$$\dot{y} = -dy$$
 - Interaksi pemangsa-mangsa akan mengurangi populasi mangsa, dengan Hukum Aksi Massa maka model pengurangan populasinya adalah $-bxy$ dengan $b > 0$ adalah suatu konstanta.
 - Keberhasilan memangsa tentu saja akan membuat laju pertumbuhan pemangsa meningkat, dengan model laju produksi pemangsa berupa cxy dengan $c > 0$.

Berdasarkan asumsi-asumsi di atas, maka model Predator-Mangsa dapat ditulis sebagai sistem persamaan berikut:

$$\begin{aligned}\dot{x} &= ax \left(1 - \frac{x}{K}\right) - bxy \\ \dot{y} &= cxy - dy\end{aligned}\tag{3.1}$$

Reduksi parameter dilakukan sedemikian agar model di atas menjadi tak berdimensi,

dengan $\tau = at$, $k = \frac{c}{d}K$, $g = \frac{d}{a}$, $u(\tau) = \frac{c}{d}x\left(\frac{\tau}{a}\right)$, $v(\tau) = \frac{b}{a}y\left(\frac{\tau}{a}\right)$ maka akan didapat

$$\begin{aligned}\dot{u} &= u \left(1 - \frac{u}{k}\right) - uv \\ \dot{v} &= g(u - 1)v\end{aligned}\tag{3.2}$$

Kesetimbangan sistem terjadi bila tidak ada laju perubahan populasi mangsa ($\dot{u} = 0$) dan predator ($\dot{v} = 0$), sehingga:

$$u \left(1 - \frac{u}{k}\right) - uv = 0\tag{3.3}$$

dan

$$g(u - 1)v = 0\tag{3.4}$$

Dengan membagi persamaan (3.4) dengan g diperoleh:

$$(u-1)v = 0; u = 1 \text{ dan } v = 0.$$

Dengan mensubstitusikan nilai tersebut ke dalam persamaan (3.4), didapat:

$$u\left(1 - \frac{u}{k}\right) - uv = 0$$

$$u - \frac{u^2}{k} - uv = 0$$

- Untuk $u = 1$

$$u - \frac{u^2}{k} - uv = 0$$

$$1 - \frac{(1)^2}{k} - (1)v = 0$$

$$1 - \frac{1}{k} - v = 0,$$

sehingga diperoleh

$$v = 1 - \frac{1}{k}$$

maka diperoleh titik kesetimbangan (keadaan tunak) $\left(1, 1 - \frac{1}{k}\right)$.

- Untuk $v = 0$

$$u - \frac{u^2}{k} - uv = 0$$

$$u - \frac{(u)^2}{k} - u \cdot 0 = 0$$

$$u - \frac{u^2}{k} = 0$$

$$u\left(1 - \frac{u}{k}\right) = 0,$$

diperoleh

$$u = 0; u = k$$

Maka diperoleh titik $(0,0)$ dan $(k,0)$.

Diperoleh tiga titik kesetimbangan untuk sistem (3.2) yaitu

$$(0,0); (k,0); \text{ dan } \left(1, 1 - \frac{1}{k}\right).$$

Misalkan kita tinjau kasus lebih spesifik di mana $k = 2$ dan $g = 1$ sehingga titik kesetimbangan untuk sistem di atas adalah

$$(0,0); (2,0); \text{ dan } \left(1, \frac{1}{2}\right).$$

Stabilitas titik-titik kesetimbangan dievaluasi dari nilai-nilai eigen yang dihitung dengan menggunakan matrix Jacobian yang bersangkutan.

Matriks Jacobian yang bersesuaian pada suatu titik kesetimbangan dapat dinyatakan dalam

$$\begin{aligned}
 J(u,v) &= \begin{bmatrix} \frac{du}{dt} & \frac{dv}{dt} \\ \frac{du}{dt} & \frac{dv}{dt} \end{bmatrix} \\
 &= \begin{bmatrix} \frac{d \left[u \left(1 - \frac{u}{k} \right) - uv \right]}{du} & \frac{d \left[u \left(1 - \frac{u}{k} \right) - uv \right]}{dv} \\ \frac{d [g(u-1)v]}{du} & \frac{d [g(u-1)v]}{dv} \end{bmatrix} \\
 J(u,v) &= \begin{bmatrix} 1 - \frac{2u}{k} - v & -u \\ gv & g(u-1) \end{bmatrix} \tag{3.5}
 \end{aligned}$$

\Rightarrow Pada titik kesetimbangan $(0,0)$

$$\begin{aligned}
 J(0,0) &= \begin{bmatrix} 1 - \frac{2(0)}{2} - (0) & -(0) \\ (1)(0) & (1)(0-1) \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}
 \end{aligned}$$

Nilai eigen untuk matriks diatas adalah:

$$\begin{vmatrix} 1-\lambda & 0 \\ 0 & -1-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)(-1-\lambda) = 0$$

$$-1 - \lambda + \lambda + \lambda^2 = 0$$

$$\lambda^2 = 1$$

$$\lambda = \pm 1, \text{ pasangan bilangan real berbeda tanda dengan vektor eigen } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Tipe kesetimbangannya adalah titik pelana.

\Rightarrow Pada titik kesetimbangan $(2,0)$

$$J(2,0) = \begin{bmatrix} 1 - \frac{2(2)}{2} - (0) & -(2) \\ (1)(0) & (1)(2-1) \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -2 \\ 0 & 1 \end{bmatrix}$$

Nilai eigen untuk matriks di atas

$$\begin{vmatrix} -1-\lambda & -2 \\ 0 & 1-\lambda \end{vmatrix} = 0$$

$$(-1-\lambda)(1-\lambda) = 0$$

$$-1 + \lambda - \lambda + \lambda^2 = 0$$

$$\lambda^2 = 1$$

$\lambda = \pm 1$, pasangan bilangan real berbeda tanda dengan vektor eigen

$$\begin{bmatrix} 1 & -0.7071 \\ 0 & 0.7071 \end{bmatrix}. \text{ Tipe kesetimbangannya adalah titik pelana.}$$

\Rightarrow Pada titik kesetimbangan $\left(1, \frac{1}{2}\right)$

$$\begin{aligned} J(u, v) &= \begin{bmatrix} 1 - \frac{2(1)}{2} - \left(\frac{1}{2}\right) & -(1) \\ (1)\left(\frac{1}{2}\right) & (1)(1-1) \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{2} & -1 \\ \frac{1}{2} & 0 \end{bmatrix} \end{aligned}$$

Nilai eigen untuk matriks di atas

$$\begin{vmatrix} -\frac{1}{2}-\lambda & -1 \\ \frac{1}{2} & -\lambda \end{vmatrix} = 0$$

$$\left(-\frac{1}{2}-\lambda\right)(-\lambda) - (-1)\left(\frac{1}{2}\right) = 0$$

$$\lambda^2 + \frac{\lambda}{2} + \frac{1}{2} = 0$$

Dengan menggunakan rumus abc , diperoleh

$$\lambda_{1,2} = -\frac{1}{4} \pm \frac{1}{2} \sqrt{\frac{7}{4}}i, \text{ yang berupa pasangan bilangan konjugat kompleks}$$

$$\text{vektor eigennya adalah } \begin{bmatrix} 0.8165 & 0.8165 \\ -0.2041 - 0.5401i & -0.2041 + 0.5401i \end{bmatrix}.$$

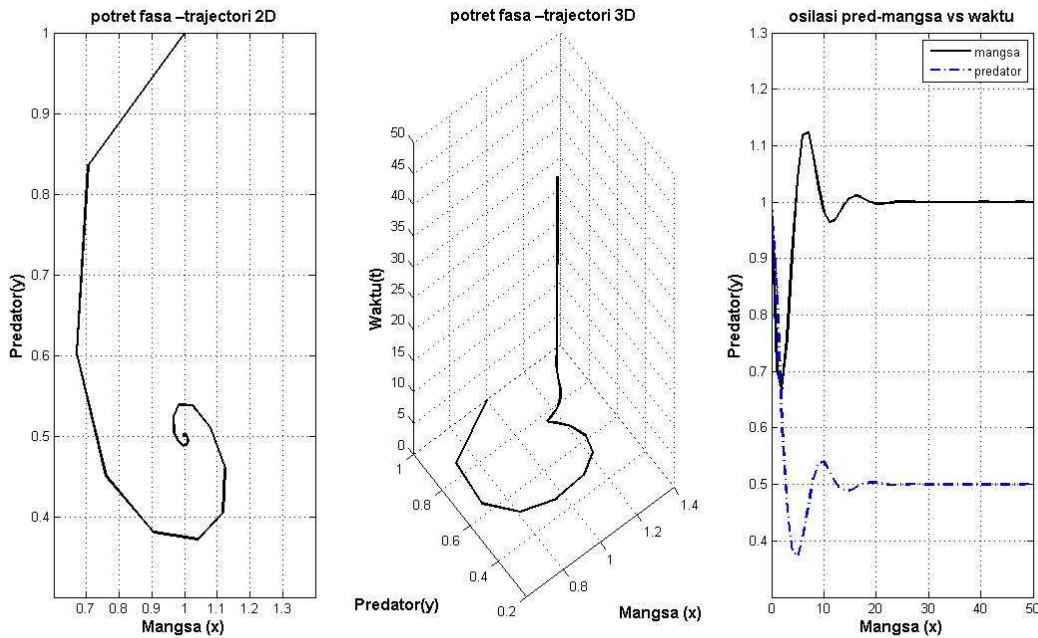
Tipe kesetimbangannya adalah spiral stabil atau *sink spiral*.

Dengan memanfaatkan Matlab maka program sederhana untuk menyelesaikan Contoh 3.1 adalah sebagai berikut:

```
% Contoh 3.1. Lotka-Volterra
clc; clear all; clf
%-----
% Koefisien tak-berdimensi:
%k = 2; g = 1;
% Mencari titik kesetimbangan (keadaan tunak):
[ue,ve]=solve('u*(1-(u/2))-u*v=0','1*(u-1)*v=0','u,v');
% Tulis sebagai pasangan titik:
eq=[ue,ve]
%-----
% Mencari nilai dan vektor eigen
k = 2; g = 1;
m=1;
while m < length(ue)+1
Jac=[1-(2*ue(m)/k)-ve(m) -ue(m);...
      g*ve(m) g*(ue(m)-1)];
[Veig, Eig]=eig(Jac)
m=m+1;
end
%-----
% Mencari solusi persamaan differensial
% x(1)=u; x(2)=v
x=zeros(2,1); % siapkan array untuk u dan v
% Tulis sistem persamaan:
F=@(t, x) [x(1)*(1-(x(1)/k))-x(1)*x(2); g*(x(1)-1)*x(2)];
[t, xa] = ode45(F, [0:50], [1 1]); % t antara 0-50s
                                % u(0)=1; v(0)=1

subplot(1,3,1)
plot(xa(:,1),xa(:,2),'LineWidth',2,'Color',[0 0 0])
grid on
xlabel('Mangsa (x)','fontsize',12,'fontweight','b')
ylabel('Predator(y)','fontsize',12,'fontweight','b')
title('potret fasa –trajectori 2D','fontsize',12,'fontweight','b')
subplot(1,3,2)
plot3(xa(:,1),xa(:,2),t,'LineWidth',2,'Color',[0 0 0])
grid on
xlabel('Mangsa (x)','fontsize',12,'fontweight','b')
ylabel('Predator(y)','fontsize',12,'fontweight','b')
zlabel('Waktu(t)','fontsize',12,'fontweight','b')
title('potret fasa –trajectori 3D','fontsize',12,'fontweight','b')
subplot(1,3,3)
plot(t,xa(:,1), '-k',t,xa(:,2), '-.b', 'LineWidth',2)
legend('mangsa', 'predator')
grid on
xlabel('Mangsa (x)','fontsize',12,'fontweight','b')
ylabel('Predator(y)','fontsize',12,'fontweight','b')
title('osilasi pred-mangsa vs waktu','fontsize',12,'fontweight','b')
```

Gambar 3.1 memperlihatkan potret fasa (trajectori) mangsa-predator serta mangsa-predator-waktu dan osilasi mangsa dan predator terhadap waktu.



Gambar 3.1. Potret fasa dan kurva osilasi terhadap waktu untuk Contoh 3.1.

Contoh 3.2

Pendulum 360°

Pendulum atau bandul yang dimaksud adalah massa yang diikat oleh batang ringan namun rigid yang salah satu ujungnya dipasang engsel sedemikian sehingga pendulum bisa mengayun dalam bidang 2D hingga sudut-sudut besar bahkan 360° jika diberi kecepatan yang cukup. Analisis keadaan tunak diasumsikan telah Anda pahami, sehingga dalam contoh di bawah ini yang dibahas hanya arti fisis potret fasanya saja.

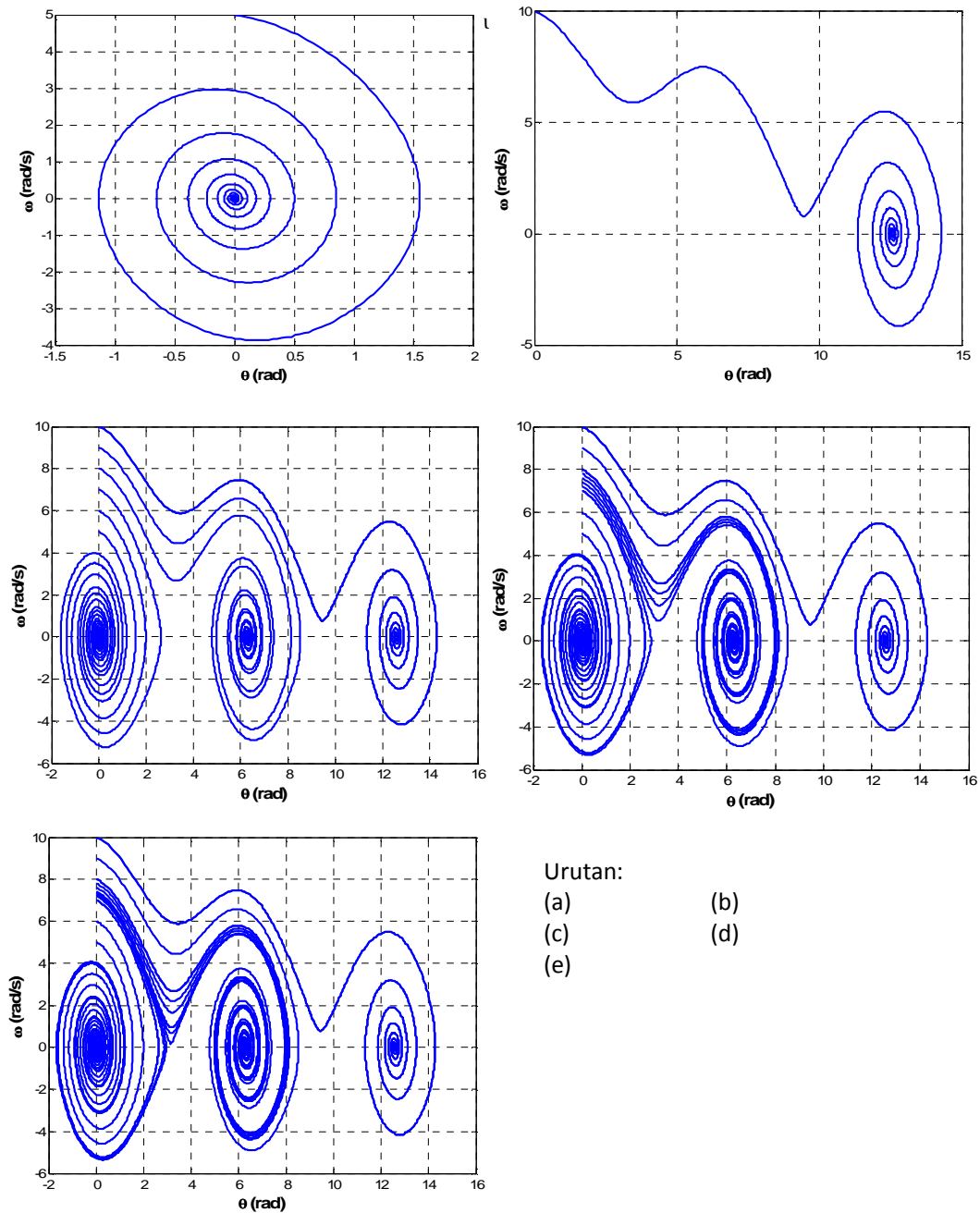
Persamaan Gerak

Kita asumsikan bahwa gaya gesek yang memperlambat pendulum hingga akhirnya berhenti berbanding lurus dengan kecepatan pendulum. Jika panjang batang pendulum 1 m dan massa di ujungnya 1 kg, serta koefisien gesek 0.5 maka persamaan gerakanya adalah sebagai berikut: In that case, the equations of motion for the pendulum are as follows.

$$\begin{aligned}\dot{x}(t) &= y(t) \\ \dot{y}(t) &= -0.5y(t) - 9.81\sin(x(t))\end{aligned}\quad (3.6)$$

di mana t adalah waktu dalam sekon (s), x adalah sudut simpangan pendulum dari vertikal dalam radian dan y adalah kecepatan sudutnya dalam radian per sekon, serta 9.81 adalah percepatan gravitasi dalam meter per sekon kuadrat. Potret fasa untuk solusi dengan posisi awal $x(0) = 0$ dan kecepatan awal $y(0) = 5$ untuk interval waktu $0 < t < 20$ dapat kita buat dengan perintah Matlab berikut:

```
g = @(t, x) [x(2); -0.5*x(2) - 9.81*sin(x(1))];
[t, xa] = ode45(g, [0:0.01:20], [0 5]);
plot(xa(:, 1), xa(:, 2), 'LineWidth', 2)
ylabel('\omega (rad/s)', 'fontsize', 12, 'fontweight', 'b')
xlabel('\theta (rad)', 'fontsize', 12, 'fontweight', 'b')
grid on
```



Gambar 3.2. Potret fasa (trajektori) sistem pendulum.
 (a) Kecepatan awal 5 rad/s; (b) kecepatan awal 10 rad/s; (c) kecepatan awal 5-10 rad/s; (d) kecepatan awal 7-8 rad/s; dan (e) kecepatan awal 7.2-7.4 rad/s

Pada Gambar 3.2 (a). Kurva solusi membentuk spiral menuju $(0, 0)$. Ketikkan `comet(xa(:,1),xa(:,2))` agar Anda yakin bahwa trajektori searah jarum dan konvergen. Sudut pendulum berosilasi bolak-balik hingga ayunannya mengecil dan akhirnya diam pada $t = 20$ s. Kecepatan sudut

juga berosilasi dengan nilai yang maksimum setiap bandul melewati sudut 0 dan berkurang hingga akhirnya diam.

Penambahan kecepatan awal

Bagaimana jika kecepatan awal ditingkatkan hingga 10 rad/s?

```
[t, xa] = ode45(g, [0:0.01:20], [0 10]);
plot(xa(:, 1), xa(:, 2), 'LineWidth', 2)
```

Terlihat pada Gambar 3.2(b) sudut bertambah hingga lebih 14 radian sebelum kurvanya membentuk spiral menuju sekitar (12.5, 0) atau tepatnya menuju (4π , 0), karena 4π radian menyatakan posisi yang sama dengan 0 radian. Pendulum telah mengayun melawati posisi teratas vertikal dan membuat dua lingkaran penuh sebelum mengalami osilasi teredam menuju posisi diam. Mula-mula kecepatan berkurang, namun setelah melewati sudut π nilainya bertambah lagi karena pendulum melewati posisi teratas dan pendulum mendapatkan tambahan momentum. Pendulum masih memiliki momentum yang cukup untuk mengayun membentuk satu lingkaran penuh lagi pada sudut 3π .

Bagaimana memperkirakan kecepatan minimum agar pendulum dapat melingkar penuh?

Pendulum dari posisi diam kemudian diberi kecepatan minimum agar dapat melingkar satu kali.

Tulis pada layar Matlab Anda:

```
hold on
for a = 5:10
[t, xa] = ode45(g, [0:0.01:20], [0 a]);
plot(xa(:, 1), xa(:, 2), 'LineWidth', 2)
end
hold off
```

Terlihat pada Gambar 3.2(c) kecepatan awal 5, 6, dan 7 rad/s tidak cukup untuk membuat sudut mencapai π . Kecepatan awal 8, 9, and 10 rad/s ternyata cukup. Dapatlah kita simpulkan bahwa kecepatan yang diperlukan tentu berada di antara 7 dan 8 rad/s.

Iterasi kedua 7 - 8 rad/s.

```
hold on
for a = 7:0.2:8
[t, xa] = ode45(g, [0:0.01:20], [0 a]);
plot(xa(:, 1), xa(:, 2), 'LineWidth', 2)
end
hold off
```

Perhatikan Gambar 3.2(d). Kecepatan ambang berada di antara 7.2 dan 7.4

Iterasi ketiga 7.2 – 7.4 rad/s.

```
hold on
for a = 7.2:0.05:7.4
[t, xa] = ode45(g, [0:0.01:20], [0 a]);
plot(xa(:, 1), xa(:, 2), 'LineWidth', 2)
end
hold off
```

Pada Gambar 3.2(e) akan Anda lihat bahwa kecepatan minimum yang dicari berada di antara 7.25 dan 7.30 rad/s.

Contoh 3.3

Model aliran lalu-lintas

Sebagai penduduk Kota Bandung, Anda tentu pernah atau bahkan sering terjebak kemacetan lalu-lintas, atau sering melihat mobil-mobil terkumpul dan bergerombol pada satu ruas jalan kemudian mengalami kemacetan tanpa alasan yang jelas. Pada contoh kali ini kita akan mencoba mengkaji perilaku model lalu-lintas tersebut berdasarkan teori aliran lalu lintas yang disebut teori “follow-the-leader.” Model lalu-lintas adalah suatu model yang cukup sederhana namun menunjukkan perilaku yang cukup kompleks. Kita tinjau suatu sistem lalu lintas yang terdiri dari jalan satu jalur dan satu arah membentuk lintasan tertutup (lingkaran) dengan mobil-mobil berada di atasnya (bayangkan Bandung saat *weekend*, ke mana Anda pergi, di situlah terjadi kemacetan!). Setiap pengemudi akan memperlambat atau mempercepat laju kendaraannya berdasarkan laju yang dimilikinya, laju kendaraan yang di depannya dan jarak terhadap kendaraan yang di depannya tersebut. Perlu diketahui bahwa pengemudi sebagai manusia memiliki waktu reaksi yang berhinga atau terbatas (biasanya sekitar satu sekon) untuk memperhatikan keadaan sekitarnya dan untuk menginjak pedal gas atau rem.

Teori *follow-the-leader* dapat dinyatakan seperti berikut

$$\ddot{u}_n(t+T) = \lambda (\dot{u}_{n-1}(t) - \dot{u}_n(t)), \quad (3.7)$$

di mana t adalah waktu, T adalah waktu reaksi, u_n adalah posisi mobil ke- n , dan λ adalah *koeffisien sensitivitas* yang dapat bergantung pada jarak antara dua mobil, $u_{n-1}(t) - u_n(t)$ dan/atau pada laju mobil ke- n , yaitu $\dot{u}_n(t)$. Persamaan di atas menerangkan bahwa seorang pengemudi akan cenderung memperlambat laju bila mobilnya melaju lebih cepat daripada mobil di depannya, atau bila ia sudah dekat dengan mobil di depannya. Kemudian, ia akan cenderung mempercepat bila melaju lebih lambat dibanding mobil di depannya. Sebagai tambahan, pengemudi (terutama pada lalu-lintas yang tak terlalu padat) mungkin cenderung untuk mempercepat atau memperlambat laju bergantung pada laju yang disarankan (misalnya batas laju yang diundangkan DLLAJR). Karena jalannya berupa lingkaran, maka dalam persamaan di atas u_0 diinterpretasi sebagai u_N , dengan N berupa jumlah total mobil.

Model yang paling sederhana adalah odel dengan λ berupa suatu konstanta positif. Pada kasus ini kita memiliki persamaan differensial linier homogen dengan koefisien untuk $\dot{u}_n(t)$ yang konstan. Terdapat solusi keadaan tunak jika semua kecepatan bernilai sama dan konstan (misalnya saat lalu lintas mengalir pada laju yang seragam), akan tetapi di sini kita akan menitikberatkan pada stabilitas aliran atau dengan kata lain mengkaji efek apa yang ditimbulkan oleh adanya perbedaan kecepatan mobil-mobil yang terlibat. Solusi persamaan (3.7) berupa superposisi solusi-solusi berbentuk eksponensial,

$$u_n(t) = \exp(\alpha t) v_n \quad (3.8)$$

di mana v_n dan α adalah konstanta-konstanta kompleks. Sistem akan bersifat tak-stabil jika kecepatan-kecepatannya tak berlimit (selalu ada solusi untuk bagian real α yang positif). Dalam notasi vektor kita tulis ulang

$$\begin{aligned}\dot{\mathbf{v}}(t) &= \exp(\alpha t) \mathbf{v} \\ \ddot{\mathbf{v}}(t+T) &= \alpha \exp(\alpha T) \exp(\alpha t) \mathbf{v}\end{aligned}\quad (3.9)$$

Substitusi kembali ke persamaan (3.7) kita dapatkan persamaan

$$\alpha \exp(\alpha T) \exp(\alpha t) \mathbf{v} = \lambda (S - I) \exp(\alpha t) \mathbf{v} \quad (3.10)$$

dengan

$$S = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \quad (3.11)$$

berupa matriks “pergeseran” yang jika dikalikan pada suatu vektor di sebelah kiri secara siklik akan mem-permutasi elemen-elemen vektor. Faktor $\exp(\alpha t)$ dapat saling dihilangkan di kedua sisi ruas untuk mendapatkan

$$\alpha \exp(\alpha T) \mathbf{v} = \lambda (S - I) \mathbf{v} \quad (3.12)$$

atau

$$\left(S - \left(1 + \frac{\alpha}{\lambda} \exp(\alpha T) \right) I \right) \mathbf{v} \quad (3.14)$$

yang juga berarti bahwa v adalah vektor eigen dengan nilai eigen $1 + \frac{\alpha}{\lambda} \exp(\alpha T)$.

Karena nilai-nilai eigen dari S adalah akar ke- N dari nilai satu yang berinterval sama dalam lingkaran satuan pada bidang kompleks dan saling berdekatan untuk N yang sangat besar, maka kemungkinan terdapat instabilitas jika $1 + \frac{\alpha}{\lambda} \exp(\alpha T)$ memiliki nilai absolut sama

dengan 1 untuk beberapa α dengan bagian real yang positif, yaitu bila $\left(\frac{\alpha T}{\lambda T} \right) e^{\alpha T}$ dapat dinyatakan dalam $\exp(i\theta) - 1$ untuk beberapa αT dengan bagian real positif. Instabilitas dapat terjadi atau tidak bergantung produk λT .

Evaluasi hal di atas dapat dilakukan dengan mem-plot $z \exp(z)$ untuk $z = \alpha T = iy$ yang merupakan bilangan kompleks pada garis kritis $\text{Re } z = 0$ dan membandingkannya dengan kurva-kurva $\lambda T (e^{i\theta} - 1)$ untuk berbagai nilai parameter λT .

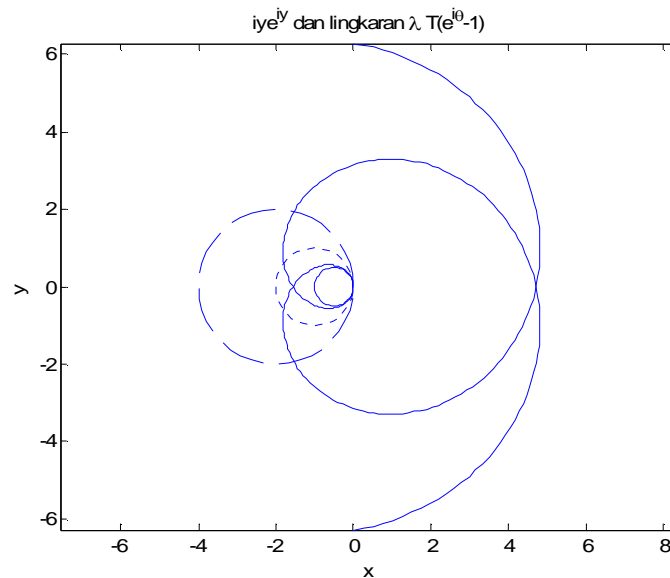
Ketikkan di layar Matlab:

```
syms y;
expand(i*y*(cos(y) + i*sin(y)))
```

```

ezplot(-y*sin(y), y*cos(y), [-2*pi, 2*pi]); hold on
theta = 0:0.05*pi:2*pi;
plot((1/2)*(cos(theta) - 1), (1/2)*sin(theta), '-');
plot(cos(theta) - 1, sin(theta), ':');
plot(2*(cos(theta) - 1), 2*sin(theta), '--');
title('iye^{iy} dan lingkaran \lambda T(e^{i\theta}-1)')
hold off

```



Gambar 3.3. Potret fasa untuk model aliran lalu-lintas.
Kecepatan mobil terhadap jarak relatif mobil

Lingkaran solid yang kecil bersesuaian dengan $\lambda T = 1/2$, merupakan limit stabilitas karena lingkaran ini tidak memotong spiral yang dihasilkan oleh $z \exp(z)$ meskipun bersinggungan dengan spiral. Lingkaran titik-titik dan garis putus bersesuaian $\lambda T = 1$ atau 2 , memotong spiral dan bersesuaian dengan aliran lalu-lintas yang tak-stabil.

Contoh 3.4

Dinamika Populasi (lagi)

Dua model dinamika populasi akan dibahas. Pertama adalah model pertumbuhan dan peluruhan standard yang menyatakan bahwa populasi suatu spesies akan punah, atau dalam waktu yang relatif singkat perilaku pertumbuhan meningkat tak terkontrol. Yang kedua adalah model yang lebih realistic di mana pertumbuhan spesies terkendala ruang yang tersedia, suplai makanan, keberadaan competitor dan predator.

Pertumbuhan dan peluruhan eksponensial

Kita asumsikan spesies memiliki populasi awal P_0 , sedangkan populasi setelah berlalu n satuan waktu adalah P_n . Setiap peningkatan interval waktu, populasi meningkat atau menurun dengan proporsi tetap dari populasi awalnya. Sehingga,

$$P_n = P_{n-1} + rP_n, \quad n \geq 1. \quad (3.15)$$

Konstanta r menyatakan selisih laju kelahiran dengan laju kematian. Populasi bertambah jika r positif, berkurang jika r negatif dan tetap tak berubah jika $r = 0$.

Berikut adalah M.file sederhana untuk menghitung populasi spesies di atas:

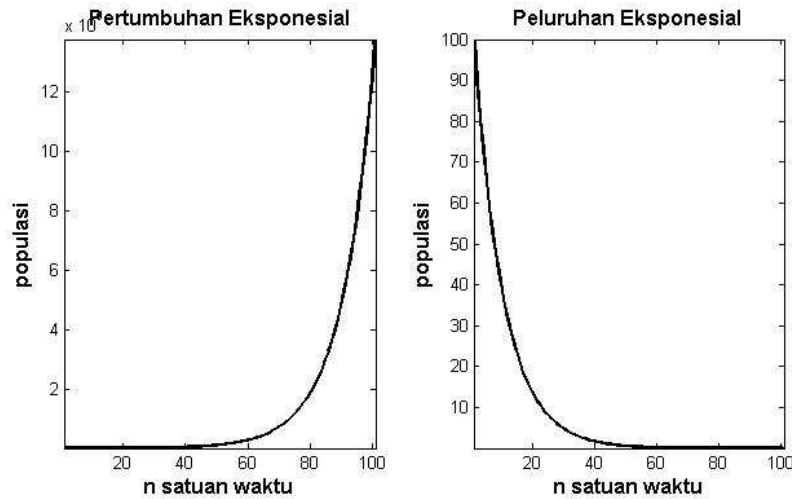
```
function P = jumpop(fun, Pawal, n, r)
% menghitung jumlah populasi seiring waktu
P = zeros(n + 1, 1);
P(1) = Pawal;
for i = 1:n
P(i + 1) = fun(P(i), r);
end
```

Simpanlah sebagai *jumpop.m*.

Program di atas secara iteratif menghitung nilai-nilai deret $a_n = f(a_{n-1})$, $n \geq 1$ dengan sebelumnya menyediakan perumusan untuk fungsi f (bernama 'fun') dan nilai awal a_0 serta parameter r .

Pada layar Matlab (*command prompt*), tulislah

```
r = 0.1; Pawal = 100; fun = inline('x*(1 + r)', 'x', 'r');
P = jumpop(fun, Pawal, 100, r);
subplot(1,2,1)
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
ylabel('populasi','fontsize',12,'fontweight','b')
title('Pertumbuhan Eksponensial','fontsize',12,'fontweight','b')
axis tight
subplot(1,2,2)
r = -0.1; Pawal = 100; fun = inline('x*(1 + r)', 'x', 'r');
P = jumpop(fun, Pawal, 100, r);
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
ylabel('populasi','fontsize',12,'fontweight','b')
title('Peluruhan Eksponensial','fontsize',12,'fontweight','b')
axis tight
```



Gambar 3.4. Pertumbuhan dan peluruhan populasi secara eksponensial.

Model (3.15) dapat dinyatakan dalam $P_n = P_0(1+r)^n$, $n \geq 0$. Model ini tidak menyertakan kendala untuk populasi yang meningkat sehingga tidak realistis (Gambar 3.4). Model berikutnya akan melibatkan pengaruh ruang yang terbatas, suplai makanan, kompetitor maupun predator.

Model Pertumbuhan Logistik

Model sebelumnya mengasumsikan perubahan relatif pada populasi bernilai konstan, yaitu

$$(P_{n+1} - P_n)/P_n = r. \quad (3.16)$$

Jika terdapat suatu suku yang menghambat pertumbuhan, maka persamaan menjadi

$$(P_{n+1} - P_n)/P_n = r - uP_n. \quad (3.17)$$

Bila diasumsikan $u = 1 + r$, maka hubungan rekursi di atas berubah menjadi

$$P_{n+1} = uP_n(1 - P_n), \quad (3.18)$$

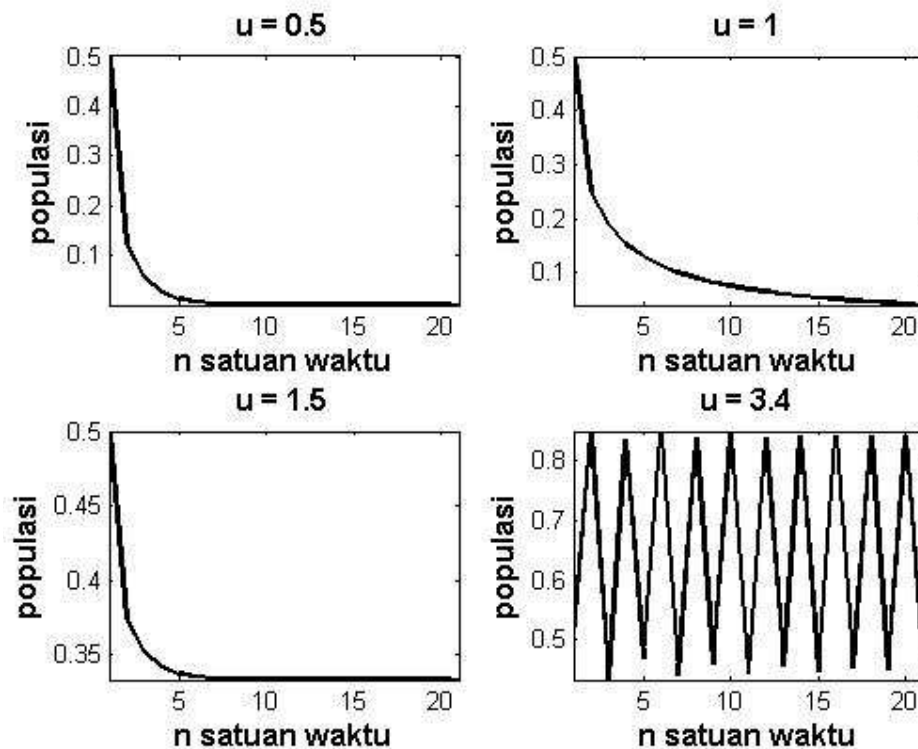
di mana u adalah suatu konstanta positif. Pada model ini, populasi P terkendala agar selalu berada di antara 0 dan 1 dan harus diinterpretasi sebagai persentase populasi maksimum yang mungkin dalam lingkungan persoalan yang dibahas. Ungkapan model yang baru adalah sebagai berikut:

```
clear fun;
fun = inline('u*x*(1 - x)', 'x', 'u');
u = 0.5; Pawal = 0.5; P = jumpop(fun, Pawal, 20, u);
subplot(2,2,1)
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
ylabel('populasi','fontsize',12,'fontweight','b')
title('u = 0.5','fontsize',12,'fontweight','b')
axis tight
subplot(2,2,2)
u = 1; Pawal = 0.5; P = jumpop(fun, Pawal, 20, u);
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
```

```

ylabel('populasi','fontsize',12,'fontweight','b')
title('u = 1','fontsize',12,'fontweight','b')
axis tight
subplot(2,2,3)
u = 1.5; Pawal = 0.5; P = jumpop(fun, Pawal, 20, u);
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
ylabel('populasi','fontsize',12,'fontweight','b')
title('u = 1.5','fontsize',12,'fontweight','b')
axis tight
subplot(2,2,4)
u = 3.4; Pawal = 0.5; P = jumpop(fun, Pawal, 20, u);
plot(P,'LineWidth',2,'Color',[0 0 0])
xlabel('n satuan waktu','fontsize',12,'fontweight','b')
ylabel('populasi','fontsize',12,'fontweight','b')
title('u = 3.4','fontsize',12,'fontweight','b')
axis tight

```



Gambar 3.5. Model pertumbuhan logistik dengan berbagai nilai u .

Gambar 3.5 adalah plot rapat populasi terhadap waktu untuk nilai konstanta pertumbuhan logistik u yang berbeda. Pada kasus pertama kita hitung rapat populasi untuk 20 interval waktu dengan mengasumsikan konstanta pertumbuhan logistik yang tetap yaitu $u = 0.5$ serta populasi awal 50% (dibuat sama untuk semua kasus. Populasi terlihat menyusut menuju nol. Ketika $u = 1$, populasi juga menyusut namun dengan laju yang lebih lambat. Berikutnya ketika $u = 1.5$, populasi terlihat stabil pada nilai 33.3%. Fenomena rapat populasi yang berhasil jelas terlihat ketika $u = 3.4$ dengan nilai-nilai yang berada di antara 45% dan 84%. Hal ini menunjukkan fenomena model dinamika populasi logistik. Model ini telah menarik perhatian para ilmuwan matematika-biologi serta sistem kompleks selama lebih dari 150

tahun. Dasar-dasar analisisnya pertama kali dikembangkan oleh Verhulst, seorang matematikawan Belgia.

Beberapa catatan penting tentang model logistik ini diberikan di bawah ini:

- **Konstanta pertumbuhan logistik tidak boleh bernilai lebih besar dari 4**

Model akan bekerja secara realistis jika output rapat populasi berada di antara 0 dan 1. Parabola $ux(1-x)$, $0 \leq x \leq 1$ bernilai maksimum jika $x = 1/2$ dengan nilai $u/4$. Agar nilainya berada di antara 0 dan 1, syarat $u \leq 4$ harus terpenuhi. Perhatikan apa yang terjadi jika $u = 4.5$,

```
P = jumpop(fun,0.5,10,4.5)
```

```
P =
```

```
1.0e+173 *
```

```
0.0000
```

```
0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-0.0000
```

```
-8.3151
```

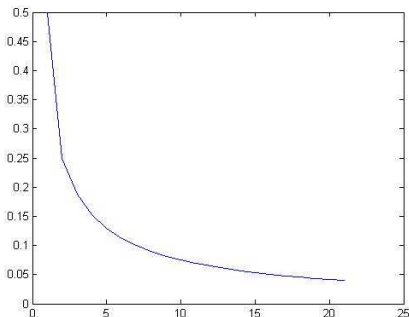
- **Jika $0 \leq u \leq 1$, rapat populasi akan cenderung menuju nol berapapun nilai awalnya**

```
P = jumpop(fun,0.5,100,0.8); P(101)
```

```
ans =
```

```
2.4205e-011
```

```
P = jumpop(fun,0.5,20,1); plot(P)
```



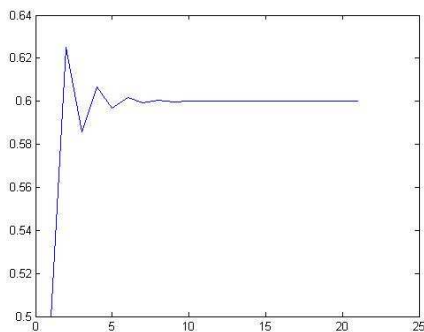
- Jika $1 \leq u \leq 3$, populasi akan stabil pada rapat $1 - \frac{1}{u}$ untuk semua nilai awal bukan nol

Sebelumnya, ketika $u = 1.5$, nilai $1 - 1/u = 1/3$. Jika $u = 2, 2.5$ dan 3 maka $1 - 1/u = 0.5, 0.6$ dan 0.66 .

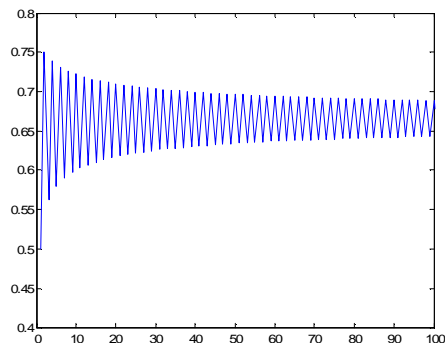
```
P = jumpop(fun,0.25,100,2);P(101)
ans =
    0.5000
```

```
P = jumpop(fun,0.75,100,2);P(101)
ans =
    0.5000
```

```
P = jumpop(fun,0.5,20,2.5); plot(P)
```



```
P = jumpop(fun,0.5,100,3); plot(P);axis([0 100 0.4 0.8])
```



- Jika $3 \leq u \leq 3.56994$, maka akan teramati siklus periodik

Terdapat deretan

$$u_0 = 3 < u_1 = 1 + \sqrt{6} < u_2 < u_3 < \dots < 4$$

sedemikian sehingga antara u_0 dan u_1 terdapat siklus dengan periode 2; antara u_1 dan u_2 terdapat siklus dengan periode 4; secara umum antara u_k dan u_{k+1} terdapat siklus dengan periode 2^{k+1} . Untuk k yang kecil berlaku aproksimasi $u_{k+1} \approx 1 + \sqrt{3 + u_k}$ sehingga

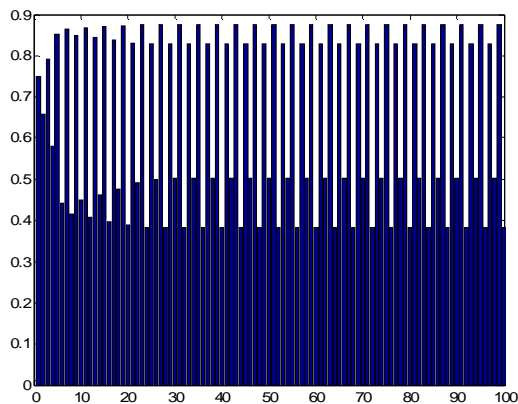
$$u_1 = 1 + \sqrt{6}$$

u1 =
3.4495

u2approx = 1 + sqrt(3 + u1)
u2approx =
3.5396

Hal di atas menerangkan perilaku osilasi dalam contoh sebelumnya untuk $u_0 < u = 3.4 < u_1$ (dalam contoh di sini $u_1 < u = 3.5 < u_2$).

P = jumpop(fun,0.75,100,3.5); bar(P);axis([0 100 0 0.9])

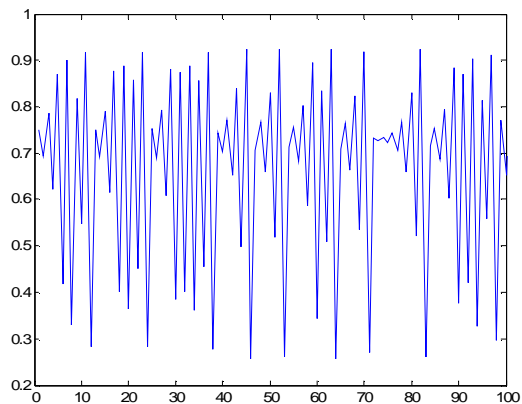


Perhatikan periodisasi yang dihasilkan.

- Terdapat suatu nilai setelah nilai-nilai sebelumnya namun masih $u < 4$, di mana terjadi fenomena 'chaos'

Dapat dibuktikan bahwa terdapat u_k yang nilainya menuju u_∞ . Nilai u_∞ seringkali disebut *parameter Feigenbaum*, nilainya sekitar 3.56994. Cobalah lihat apa yang terjadi jika u berada di antara parameter Feigenbaum dan 4.

P = jumpop(fun,0.75,100,3.7); plot(P);axis([0 100 0.2 1])



Contoh di atas adalah suatu fenomena yang oleh matematikawan disebut “chaotic”. Fenomena di atas bukanlah fenomena acak karena deretan yang dihasilkan dibuat oleh prosedur matematika yang deterministik atau rigid. Namun hasilnya terwujud dalam pola yang tak jelas/tak beraturan. Fenomena *chaos* tidaklah dapat diprediksi, namun dengan bantuan metode-metode modern (termasuk analisis komputer), beberapa polanya dapat dikenali. Misalnya, gambar terakhir mengisyaratkan kemungkinan siklus tak-stabil periodik dan fenomena lainnya.

Membuat Model Populasi Logistik dengan Simulink

Simulink adalah paket aksesoris Matlab berbentuk library-library modular yang dapat dimanfaatkan dalam pemodelan sistem dinamik maupun rekayasa-keteknikan.

Ketikkan di Matlab *command prompt*

`simulink`

Klik

File: Model: New

Buatlah diagram sistem di bawah ini:

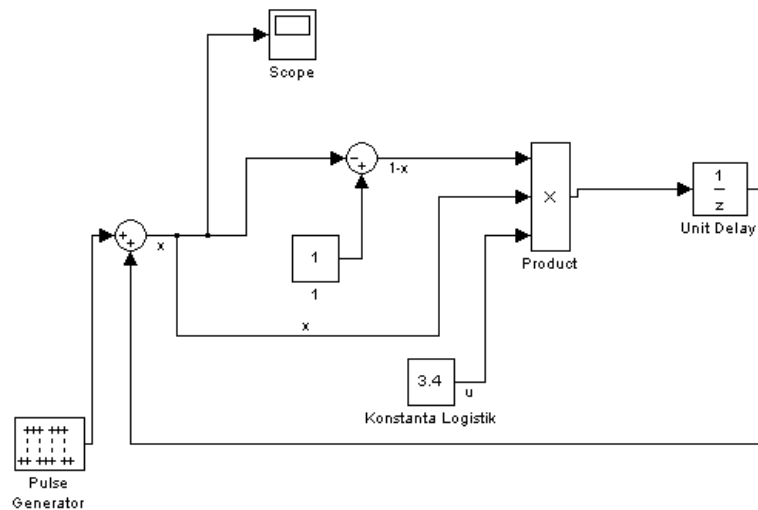
Parameter untuk *Pulse Generator*:

Pulse type : sample based

Amplitude: 0.5

Period: 200

Pulse width: 1

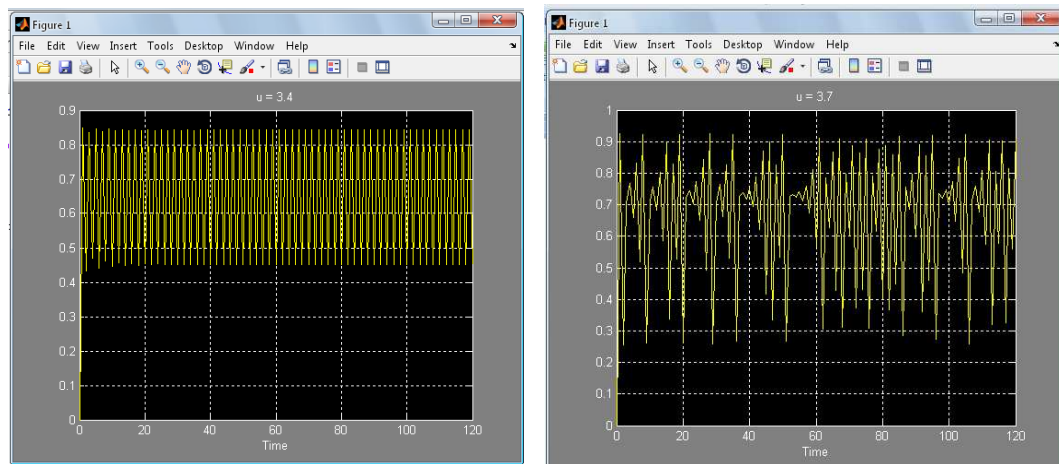


Gambar 3.6. Blok sistem pertumbuhan logistik

```
[t, x] = sim('popjum', [0 120]);
simplot(t, x); title('u = 3.4')
```

Ubah konstanta logistik dari 3.4 ke 3.7

```
[t, x] = sim('popjum', [0 120]);
simplot(t, x); title('u = 3.7')
```



Gambar 3.7. Osilasi rapat populasi logistik untuk $u = 3.4$ (periodik) dan $u = 3.7$ (chaotic)



IV. ANALISIS DERET-WAKTU NON-LINIER

Proses-proses alami pada sistem Bumi sering memperlihatkan perilaku yang kompleks dan *chaotic*. Metode-metode berbasis teknik-teknik linier seringkali memberikan hasil yang tidak memuaskan. Dalam beberapa dekade terakhir, teknik-teknik baru untuk analisis data non-linier yang diturunkan dari teori *chaos* telah menjadi semakin populer. Sebagai contoh, perilaku non-linier dapat dianalisis dengan mendefinisikan hukum penskalaan dan dimensi fraktal proses-proses alami. Namun demikian, kebanyakan metode analisis non-linier memerlukan serial data yang panjang maupun konstan. Syarat ini tidak selalu dapat dipenuhi dalam sains kebumihian. Sementara hampir semua metode analisis data non-linier dapat bekerja dengan baik pada data sintetik, metode-metode ini seringkali gagal menjelaskan perilaku non-linier pada data yang nyata.

4.1. Potret Fasa

Di sebut juga **potret ruang fasa** (*phase space portrait*).

Dalam dekade terakhir, *recurrence plot* sebagai metode baru untuk analisis data non-linier telah menjadi populer untuk keperluan sains maupun rekayasa. *Recurrence* yang arti harfiahnya adalah pengulangan atau kemunculan kembali, merupakan adalah perilaku fundamental dari sistem-sistem dinamik yang dissipatif. Meski sistem-sistem di atas dapat mengalami divergensi eksponensial bila mengalami gangguan-gangguan kecil, namun setelah sekian waktu sistem akan kembali ke keadaan yang secara sebarang dekat dengan keadaannya semula dan seterusnya mengalami evolusi yang sama. *Recurrence plot* mampu memvisualisasi perilaku *recurrence* dari sistem-sistem dinamik. Metode ini sekarang telah diterima luas sebagai perangkat analisis non-linier dari himpunan data yang pendek dan non-stasioner.

Titik awal hampir semua analisis data non-linier adalah penyusunan potret fasa dari suatu sistem. Sebetulnya hal ini telah kita lakukan pada bab-bab terdahulu saat kita membahas perilaku sistem dinamik yang direpresentasikan dalam bentuk persamaan differensial. Keadaan suatu sistem dapat digambarkan oleh variabel-variabel keadaannya, misalnya $x_1(t)$, $x_2(t)$, ..., $x_d(t)$. Misalkan terdapat dua variabel keadaan yaitu *temperatur* dan *tekanan* untuk melukiskan keadaan termodinamika mantel Bumi sebagai suatu sistem kompleks. Keadaan ke- d suatu variabel pada waktu t membentuk suatu vektor dalam ruang berdimensi d yang disebut ruang fasa. Keadaan suatu sistem biasanya berubah terhadap waktu. Vektor pada ruang fasa karenanya menyatakan evolusi temporal atau dengan kata lain dinamika sistem tersebut. Lintasan atau trajektori variabel keadaan memberikan semua informasi penting tentang sifat dinamik sistem. Demikian pula sistem-sistem periodik maupun *chaotic* masing-masing memiliki potret fasa karakteristik.

Observasi proses alami pada kenyataannya seringkali tidak menghasilkan semua variabel keadaan yang mungkin. Baik karena tidak diketahui, ataupun karena tidak dapat diukur. Namun demikian, karena komponen-komponen sistem saling terkopel, kita dapat merekonstruksi trajektori ruang fasa dari observasi tunggal u_i :

$$x_i = \left(u_i, u_{i+\tau}, \dots, u_{i+(m-1)\tau} \right)^T \quad (4.1)$$

di mana m adalah dimensi yang membatasi sistem dan τ adalah delay waktu (berbasis indeks; delay waktu real $\tau = \Delta t$). Rekonstruksi ruang fasa ini disebut *time delay embedding*. Rekonstruksi ruang fasa tidaklah secara eksak sama dengan ruang fasa orisinalnya, namun sifat-sifat topologinya masih dapat dipertahankan jika dimensi m cukup besar. Pada

prakteknya dimensi m harus lebih besar dari dimensi fasa, atau tepatnya $m > 2d$, sehingga trajektori yang terekonstruksi cukup memadai untuk analisis data yang berkaitan.

Sebagai contoh kita akan bahas potret fasa dari sebuah osilator harmonik (misal pendulum tak teredam). Pertama-tama kita nyatakan vektor posisi y_1 dan vektor kecepatan y_2 .

Pada layar Matlab, tulislah

```
x = 0 : pi/10 : 3*pi;
```

```
y1 = sin(x);
```

```
y2 = cos(x);
```

Buatlah potret fasa

```
subplot(1,2,1)
```

```
plot(y1,y2)
```

```
xlabel('y_1'), ylabel('y_2')
```

```
title('Signal Periodik','fontsize',12,'fontweight','b')
```

Potret fasa berupa lingkaran, mengisyaratkan suatu *recurrence* yang eksak untuk setiap keadaan setiap satu siklus (Gambar 4.1(a)). Dengan menggunakan *time delay embedding* kita dapat merekonstruksi potret fasa ini dengan hanya menggunakan satu jenis data, misalnya vektor kecepatan serta nilai delay 5 yang berkaitan dengan seperempat periode pendulum kita

```
subplot(1,2,2)
```

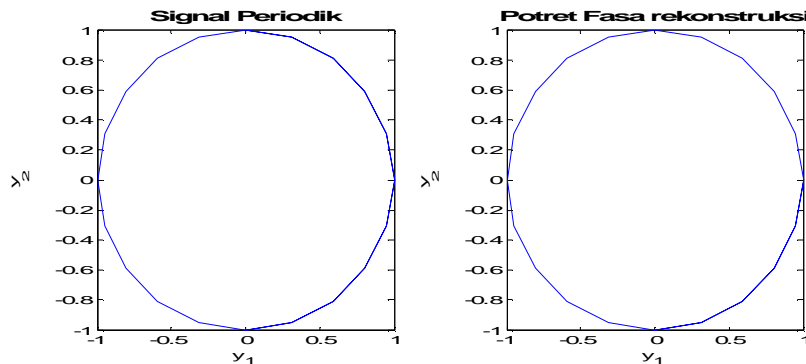
```
t = 5;
```

```
plot(y2(1:end-t), y2(1+t:end))
```

```
xlabel('y_1'), ylabel('y_2')
```

```
xlabel('y_1'), ylabel('y_2')
```

```
title('Potret Fasa rekonstruksi','fontsize',12,'fontweight','b')
```



Gambar 4.1. (a) Potret fasa original, (b) potret fasa rekonstruksi *time delay embedded*

Seperti kita lihat pada Gambar 4.1(b), potret fasa rekonstruksi hampir sama dengan potret fasa aslinya.

Berikutnya kita akan terapkan prinsip di atas pada persoalan yang lebih kompleks, yaitu suatu sistem non-linier tipikal yang kita sebut *sistem Lorenz* yang pertama kali digagas oleh Edward Lorenz tahun 1963. Sistem ini adalah sistem tiga dimensi untuk menjelaskan turbulensi di atmosfer dengan tiga variabel keadaan: dua bentuk distribusi temperature dan satu distribusi kecepatan.

Ketika ia sedang mengkaji pola-pola cuaca, Lorenz menyadari bahwa cuaca seringkali berubah tidak menurut pola yang diprediksi sebelumnya. Analisis ini berdasar pada suatu model cuaca sederhana. Ia menemukan fenomena di mana perubahan-perubahan awal yang kecil dapat menyebabkan perubahan dramatis pola-pola cuaca yang divergen. Perilaku ini sering disebut sebagai *efek kupu-kupu*. Secara matematis, sistem Lorenz direpresentasikan oleh tiga persamaan differensial non-linier untuk tiga variabel yang telah disebutkan.

$$\begin{aligned}\frac{dx}{dt} &= s(y(t) - x(t)), \\ \frac{dy}{dt} &= -x(t)z(t) + rx(t) - y(t), \\ \frac{dz}{dt} &= x(t)y(t) - bz(t).\end{aligned}\tag{4.2}$$

Integrasi persamaan differensial di atas dilakukan untuk menghitung triplet xyz dari *Lorenz attractor*. *Lorenz attractor* adalah orbit keadaan-keadaan fasa yang khas dan berpindah di sekitar suatu titik tetap. s , r dan b adalah parameter-parameter sistem yang mengontrol perilaku *chaotic*. Dalam kasus ini, kita gunakan $s = 10$, $r = 28$ dan $b = 8/3$. Waktu *delay* bernilai $dt = 0.01$. Nilai awalnya adalah $x_1 = 8$, $x_2 = 9$ and $x_3 = 25$ (nilai awal ini dapat diganti oleh nilai yang lain).

```
dt = .01;
s = 10;
r = 28;
b = 8/3;
x1 = 8; x2 = 9; x3 = 25;
for i = 1 : 5000
    x1 = x1 + (-s*x1*dt) + (s*x2*dt);
    x2 = x2 + (r*x1*dt) - (x2*dt) - (x3*x1*dt);
    x3 = x3 + (-b*x3*dt) + (x1*x2*dt);
    x(i,:) = [x1 x2 x3];
end

t = 0.01 : 0.01 : 50;
subplot(1,2,1)
plot(t, x(:,1))
ylabel('Temperatur','fontsize',12,'fontweight','b')
xlabel('Waktu','fontsize',12,'fontweight','b')
```

```
Berikutnya kita plot potret fasa ketiga parameter dalam sistem Lorenz,
subplot(2,2,2)
plot3(x(:,1),x(:,2),x(:,3)), grid, view(70,30)
xlabel('x_1','fontsize',12,'fontweight','b')
ylabel('x_2','fontsize',12,'fontweight','b')
zlabel('x_3','fontsize',12,'fontweight','b')
title('Potret Fasa: Lorenz Attractor','fontsize',12,'fontweight','b')
```

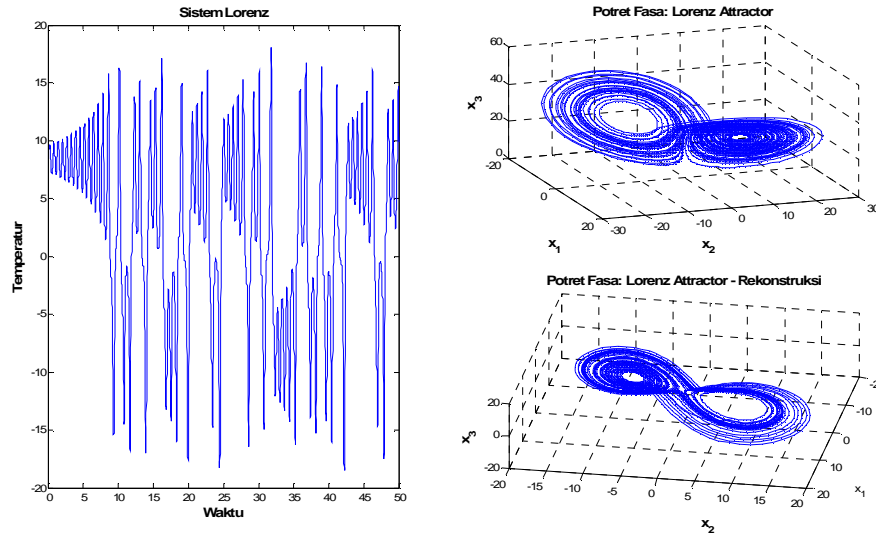
Trajektori sistem Lorenz tidak mengikuti lintasan yang sama, tetapi berulang sangat dekat pada keadaan sebelumnya. Jika kita ikuti dua segmen trajektori yang berdekatan, kita akan lihat bahwa keduanya akan menuju daerah-daerah yang berbeda dalam ruang fasa seiring waktu. Trajektori jelas mengitari suatu titik tetap hingga setelah periode acak tertentu, trajektori akan mengitari titik yang lain. Sifat-sifat ini adalah sifat tipikal sistem *chaotic*.

Meski gangguan-gangguan kecil menyebabkan sistem mengalami divergensi eksponensial, sistem akan kembali ke aproksimasi keadaan sebelumnya melalui lintasan yang hampir sama.

Sekarang kita akan mencoba merekonstruksi potret fasa dengan hanya menggunakan satu variabel keadaan dan *delay* senilai 6.

```
subplot(2,2,4)
tau = 6;
plot3(x(1:end-2*tau,1),x(1+tau:end-tau,1),x(1+2*tau:end,1))
grid, view([100 60])
xlabel('x_1'), ylabel('x_2'), zlabel('x_3')
ylabel('x_2','fontsize',12,'fontweight','b')
zlabel('x_3','fontsize',12,'fontweight','b')
title('Potret Fasa: Lorenz Attractor - Rekonstruksi','fontsize',12,'fontweight','b')
```

Potret fasa yang hampir sama dengan dua ‘telinga’ yang tipikal terlihat (Gambar 4.2). Sifat karakteristik sistem *chaotic* juga terlihat pada rekonstruksi di atas.



Gambar 4.2. Sistem Lorenz

Waktu *delay* dan *embedding dimension* harus sedemikian dipilih dengan analisis data sebelumnya. *Delay* dapat diestimasi dengan bantuan fungsi autokovarians atau autokorelasi. Sebagai contoh, untuk osilasi periodik,

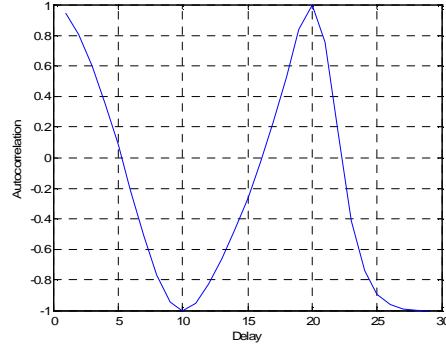
```
x = 0 : pi/10 : 3*pi;
y1 = sin(x);
```

kita hitung dan plot fungsi autokorelasi

```
for i = 1 : length(y1) - 2
r = corrccoef(y1(1:end-i),y1(1+i:end));
C(i) = r(1,2);
end
plot(C)
```

xlabel('Delay'), ylabel('Autocorrelation')
grid on

Delay kita pilih sedemikian sehingga nilai fungsi autokorelasinya bernilai nol pertama kali yang dalam kasus kita adalah 5. Estimasi *embedding dimension* dilakukan dengan metode *tetangga terdekat* atau yang lebih sederhana yaitu *recurrence plot* yang akan dibahas dalam sub-bab berikut.



Embedding dimension secara bertahap ditambah sampai mayoritas garis-garis diagonal sejajar dengan garis matriks identitas.

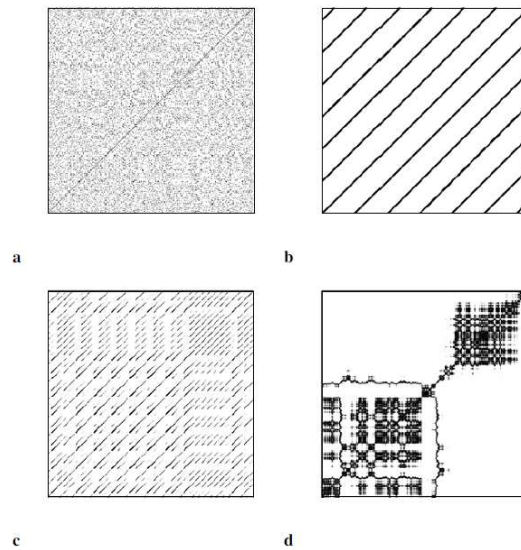
Trajektori ruang fasa atau rekonstruksinya menjadi dasar beberapa ukuran karakteristik dalam analisis data non-linier, seperti *eksponen Lyapunov*, entropi atau dimensi *Rényi*. Trajektori ruang fasa dan rekonstruksinya juga diperlukan pada saat rekonstruksi *recurrence plot*.

4.2. Recurrence Plot

Trajektori ruang fasa suatu sistem dinamik yang memiliki lebih dari tiga dimensi cukup sulit untuk divisualisasi. *Recurrence plot* merupakan suatu cara untuk menganalisis sistem-sistem berdimensi tinggi. Plot ini dapat digunakan untuk mendeteksi transisi antara berbagai rezim yang berbeda ataupun interrelasi beberapa sistem sehingga repetisi keadaan-keadaan dalam ruang fasa dapat dinyatakan dalam plot dua dimensi.

$$R_{i,j} = \begin{cases} 0 & \|x_i - x_j\| > \varepsilon \\ 1 & \|x_i - x_j\| \leq \varepsilon \end{cases} \quad (4.3)$$

Jika jarak antara dua keadaan ke- i dan ke- j pada trajektori lebih kecil dari suatu ambang tertentu ε , nilai matriks *recurrence* adalah satu. Jika bukan, maka nilainya adalah nol. Sehingga analisis lengkap akan membentuk pasangan test untuk semua keadaan. Untuk N buah keadaan kita hitung N^2 test. *Recurrence plot* karenanya adalah display dua dimensi dari suatu matriks $N \times N$ di mana piksel hitam menunjukkan $R_{ij} = 1$ sedangkan piksel putih menyatakan $R_{ij} = 0$ disertai sistem koordinat yang mewakili dua sumbu waktu. *Recurrence plot* ini dapat membantu untuk menemukan karakterisasi paling dasar dari suatu set data sistem dinamik atau untuk menemukan transisi serta interrelasi sistem yang bersangkutan (Gambar 4.3).



Gambar 4.3. Beberapa *recurrence plot* yang mewakili perilaku dinamik yang tipikal: (a) data stasioner tak-terkorelasi (white noise), (b) osilasi periodik, (c) data *chaotic* dan (d) data non-stasioner dengan perubahan-perubahan yang tajam.

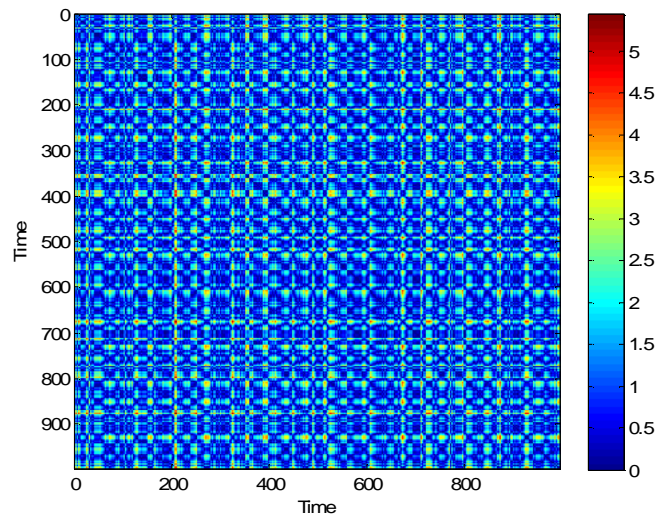
Misalkan kita memiliki suatu data isotop oksigen dari *calcareous algae* (foraminifera) pada sedimen laut dalam dengan periodisitas 100, 40 dan 20 ribu tahun plus ikutan noise gaussiannya (data tersebut kita dapatkan dengan membuat data sintetik).

```
clear
t = 0 : 3 : 996; % Pertama, kita buat interval waktu
x1 = 0.5*sin(2*pi*t/100) + ...
1.0*sin(2*pi*t/40) + ...
0.5*sin(2*pi*t/20); % Buatlah datanya
series1 = x1;
series1 = series1 + 0.5*randn(size(series1)); % Tambahkan noise gaussian
```

Kita awali analisis kita dengan mengasumsikan ruang fasa berdimensi satu. Kalkulasi jarak antara semua titik pada trajektori ruang fasa menghasilkan matriks jarak **S** (Gambar 4.4).

```
N = length(series1);
S = zeros(N, N);
for i = 1 : N,
    S(:,i) = abs(repmat(series1(i), N, 1) - series1(:));
end
```

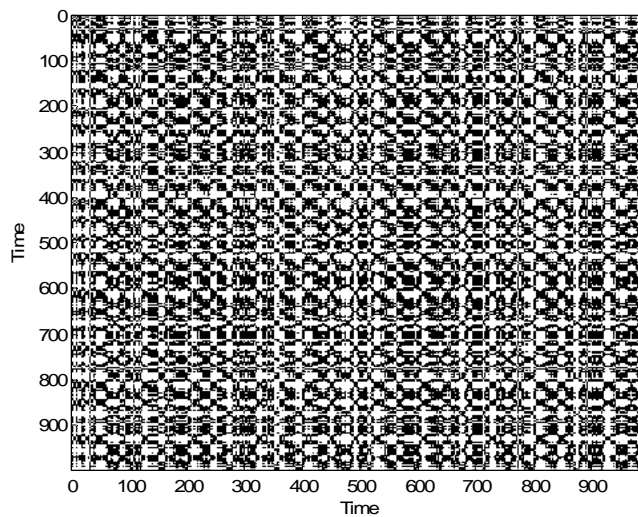
```
imagesc(t,t,S) % Berikutnya kita plot matriks jarak S:
colorbar
xlabel('Time'), ylabel('Time')
```



Gambar 4.4. Visualisasi matriks jarak dari data sintetik memberikan ukuran kuantitatif untuk jarak antar keadaan pada waktu-waktu tertentu.

Jika kita terapkan ambang $\varepsilon = 1$ pada \mathbf{S} maka kita dapatkan piksel hitam-putih seperti pada Gambar 4.5.

```
imagesc(t,t,S<1)
colormap([1 1 1;0 0 0])
xlabel('Time'), ylabel('Time')
```



Gambar 4.5. *Recurrence plot* yang diturunkan dari matriks \mathbf{S} pada Gambar 4.4 setelah menerapkan nilai ambang $\varepsilon = 1$.

Gambar 4.4.dan 4.5 keduanya memperlihatkan pola-pola periodik. Jarak antara pola-pola periodik tersebut memperlihatkan siklus-siklus yang dimiliki deret waktu yang bersangkutan. Struktur periodik yang signifikan memiliki periode 200 dan 100 ribu tahun. Periode 200 ribu tahun adalah yang paling signifikan karena superposisi siklus 100 dan 40 ribu tahun (keduanya pembagi siklus 200 ribu tahun). Terlihat juga sub-struktur kecil yang memiliki ukuran 40 dan 20 ribu tahun.

Berikutnya kita beranjak pada penerapan *recurrence plot* pada sistem Lorenz. Kita buat terlebih dahulu data triplet xyz sintetik:

```
clear
dt = .01;
s = 10;
r = 28;
b = 8/3;
x1 = 8; x2 = 9; x3 = 25;

for i = 1 : 5000
    x1 = x1 + (-s*x1*dt) + (s*x2*dt);
    x2 = x2 + (r*x1*dt) - (x2*dt) - (x3*x1*dt);
    x3 = x3 + (-b*x3*dt) + (x1*x2*dt);
    x(i,:) = [x1 x2 x3];
end
```

Kita pilih komponen pertama dari sistem ini yang di-sample ulang dan rekonstruksi trajektori ruang fasa dilakukan dengan menggunakan *embedding dimension* $m = 3$ dan $\tau = 2$ yang bersesuaian dengan *delay* 0.17 sekon.

```
t = 0.01 : 0.05 : 50;
y = x(1:5:5000,1);
m = 3; tau = 2;
N = length(y);
N2 = N - tau*(m - 1);
```

Data original semula memiliki panjang 5000 yang setelah di-resampling panjangnya menjadi 1000 atau 50 sekon, namun karena metode *delay* waktu, trajektori ruang fasa terrekonstruksi memiliki panjang 996. Trajektori dibuat dengan:

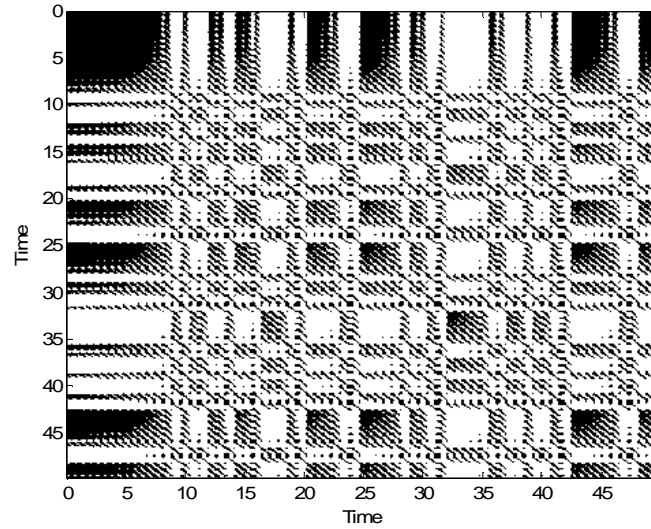
```
for mi = 1:m
    xe(:,mi) = y([1:N2] + tau*(mi-1));
end
```

Untuk mempercepat test pasangan antara tiap titik pada trajektori maka dilakukan transformasi vektor trajektori ke dalam dua buah vektor tets:

```
x1 = repmat(xe,N2,1);
x2 = reshape(repmat(xe(:),1,N2)',N2*N2,m);
```

Hitung *recurrence plot* menggunakan Euclidean norm:

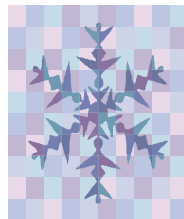
```
S = sqrt(sum((x1 - x2).^ 2,2 ));
S = reshape(S,N2,N2);
imagesc(t(1:N2),t(1:N2),S<10)
colormap([1 1 1;0 0 0])
xlabel('Time'), ylabel('Time')
```

Gambar 4.6. *Recurrence plot* untuk sistem Lorenz dengan ambang $\varepsilon = 10$. Daerah dengan konsentrasi garis-garis diagonal mengindikasikan orbit-orbit periodik tak-stabil, sebagai petunjuk tipikal sistem *chaotic*.

Plot yang dihasilkan (Gambar 4.6) memperlihatkan banyaknya garis-garis diagonal pendek. Garis-garis tersebut mengindikasikan *epoch*, di mana trajektori berjalan sejajar terhadap sekuen trajektori sebelumnya ataupun sesudahnya. Keadaan dan dinamika trajektori berarti bersifat sama pada waktu tersebut. Jarak antar garis-garis diagonal ini menyatakan periode atau siklus yang berbeda dan tidak konstan.

Struktur *recurrence plot* dapat juga digambarkan dengan besaran-besaran kuantitatif berdasarkan panjang garis-garis diagonal dan vertikal yang dapat digunakan untuk menelusuri transisi tersembunyi dalam suatu proses. Ekstensi bivariat dan multivariat dari plot ini lebih jauh memfasilitasi analisis test korelasi dan sinkronisasi korelasi non-linier.



V. CELLULAR AUTOMATA (CA)

Cellular Automata (atau dalam bentuk tunggalnya disebut *cellular automaton*) adalah model-model matematika (hasil penyederhanaan) untuk interaksi-interaksi spasial-temporal suatu entitas (fisis, biologis, kimiawi, sosial, ekonomi dan lainnya). Keadaan entitas tersebut (yang diimplementasikan dalam sel maupun grid) pada waktu yang akan datang ditentukan oleh keadaan intrinsiknya pada saat sekarang dan interaksinya dengan sel-sel tetangganya.

Cellular automata dikarakterisasi oleh transisi fasa yang dapat menghasilkan pola-pola kompleks suatu sistem terutama dari perilaku *self-organization* sistem tersebut. Teknik CA ini telah banyak diterapkan untuk memahami berbagai sistem seperti prediksi zona-zona aktif di sekitar gunung berapi, prediksi penjaralan kebakaran hutan, pemodelan aliran airtanah, pemodelan perkembangbiakan spesies biologi, prediksi urbanisasi daerah perkotaan dan lain sebagainya.

5.1. Pemodelan Cellular Automata

Game of Life

Cellular automata merupakan system dinamik yang diskrit dengan ruang yang terbagi ke dalam sel spasial beraturan dan progress waktu dalam langkah diskrit. Masing-masing sel dalam system mempunyai suatu batasan keadaan. Keadaan sel di-*update* berdasarkan aturan-aturan lokal, yaitu keadaan sel pada suatu waktu bergantung pada keadaan sel itu sendiri dan keadaan sel tetangga pada waktu sebelumnya (Wolfram 1984).

Aplikasi cellular automata yang pertama kali adalah “Game of Life” dari John Conway (Gardner 1972). “Life” dikonstruksi sebagai grid 2-dimensi dengan dua keadaan sel yang mungkin serta delapan sel tetangganya. Dua keadaan sel yang mungkin tersebut adalah *hidup* atau *mati*. Delapan sel tetangganya berada dalam arah Timur, Barat, Utara, Selatan, Baratdaya, Tenggara, Timurlaut dan Baratlaut dari sel yang bersangkutan. Jenis tetangga berjumlah delapan ini disebut sebagai *Moore Neighbourhood*. Jenis lainnya yaitu tetangga berjumlah empat disebut *von Neumann Neighbourhood*.

Dalam “Game of Life” Conway, generasi sebuah sel dapat berstatus *survive*, mati atau lahir berdasarkan aturan berikut:

- *Survive*, sel yang hidup dengan dua atau tiga sel tetangga yang hidup, maka pada generasi selanjutnya sel tersebut akan survive.
- Mati, sel yang hidup dengan kurang dari dua atau lebih dari tiga sel tetangga yang hidup, maka sel tersebut akan mati karena isolasi dan *overcrowding*.
- Lahir, sel yang mati dengan tiga sel tetangganya yang hidup, maka sel tersebut akan menjadi hidup pada generasi selanjutnya.

Dengan menggunakan aturan sederhana tersebut, model dapat mensimulasi struktur yang sangat kompleks, yaitu dinamika keadaan sel mati, survive atau lahir pada generasi selanjutnya.

“Game of Life” Conway selalu memikat perhatian para peneliti terutama untuk mengeksplor kompleksitas cellular automata yang muncul dari aturan sederhana. Pada 1983, Stephen Wolfram mempublikasikan paper pertamanya yang secara sistematis menginvestigasi cellular automata 1-dimensi paling sederhana yang disebut *elementary cellular automata* (Wolfram 2002, 1994, 1983). Tiap sel dalam *elementary cellular automata* tersebut hanya mempunyai dua nilai atau keadaan yang mungkin yaitu 0 dan 1, dan aturan transisi hanya

bergantung pada nilai tetangga terdekat. Berdasarkan sifat aturan sederhana pada kompleksitas sistem ini, Wolfram menduga bahwa kompleksitas di alam mungkin disebabkan oleh mekanisme yang mirip (Wolfram 1994).

Studi cellular automata terus berkembang terutama pada era 1990an, di mana teknologi komputer sudah sangat mendukung, sehingga studi teori kompleksitas, *self-organization*, dan *chaos* telah berkembang sangat pesat. Berdasarkan prinsip kompleksitas, struktur kompleks dapat dihasilkan oleh aturan yang sangat sederhana. Oleh karena itu, teknik cellular automata ini dapat digunakan untuk mengeksplorasi perilaku dinamik dan evolusi suatu sistem. Cellular automata dikonstruksi dari berbagai komponen yang identik, sederhana, tapi bersama-sama menunjukkan karakteristik atau sifat kompleksnya.

Menurut Levy (1992), cellular automata adalah mesin yang beroperasi sendiri yang *“processes information, proceeding logically, inexorably performing its next action after applying data received from outside itself in light of instructions programmed within itself”*.

Dalam system cellular automata, ruang terbagi ke dalam sel-sel beraturan. Keadaan sel ditentukan oleh sel itu sendiri dan keadaan sel tetangganya pada waktu sebelumnya melalui aturan transisi lokal. Keadaan semua sel di-*update* secara serempak. Sifat keseluruhan sistem ditentukan oleh efek-efek kombinasi sistem dalam langkah waktu diskrit.

5.2. Lima Elemen Dasar Cellular Automata

Berdasarkan definisi sebelumnya, cellular automata terdiri dari lima elemen dasar:

- Sel**, yang merupakan satuan spasial dasar dalam ruang cellular. Sel cellular automata disusun dalam pembagian spasial. Grid dua-dimensi dari sel merupakan bentuk umum cellular automata yang digunakan untuk memodelkan evolusi dinamik suatu sistem. pertumbuhan. Ruang sel dapat juga dibagi ke dalam susunan lain, seperti bentuk sarang lebah atau bahkan tiga-dimensi.
- Keadaan**, mendefinisikan atribut suatu sistem. Pada suatu waktu, tiap sel hanya mempunyai satu keadaan. Keadaan tersebut merepresentasikan satu sifat sel.
- Tetangga**, sekumpulan sel yang berinteraksi dengan sel yang dikaji. Dalam ruang dua-dimensi, terdapat dua jenis tetangga, yaitu *von Neumann* yang terdiri dari empat sel tetangga, dan *Moore* yang terdiri dari delapan sel tetangga.
- Aturan transisi**, yang menentukan bagaimana keadaan suatu sel dapat berubah, sebagai suatu respon dari keadaan sel itu sendiri dan keadaan tetangganya. Aturan transisi merupakan komponen kunci dari cellular automata. Aturan tersebut menentukan proses bagaimana suatu sistem dimodelkan dan merupakan dasar keberhasilan pemodelan yang baik. Untuk cellular automata yang ketat, aturan transisinya seragam dan berlaku serempak pada seluruh sel dalam sistem tersebut.
- Waktu**, yang menspesifikasi dimensi temporal dimana cellular automata berada. Berdasarkan definisi cellular automata, keadaan semua sel di-*update* secara simultan dalam setiap iterasi sepanjang waktu.

5.3. Representasi Matematis Cellular Automata

Misalkan $S_{x_{i,j}}^t$ adalah keadaan sel $x_{i,j}$ pada lokasi ke- (i, j) dan waktu t dan berada dalam rentang nilai-nilai keadaan yang berhingga dalam ruang cellular. Jika $S_{x_{i,j}}^{t+1}$ menyatakan keadaan sel pada waktu $t + 1$, maka:

$$S_{x_{i,j}}^{t+1} = f\left(S_{x_{i,j}}^t, S_{\Omega_{x_{i,j}}}^t\right) \quad (5.1)$$

di mana $\Omega_{x_{ij}}$ merupakan semua sel tetangga dari sel $x_{i,j}$, $S_{\Omega_{x_{ij}}}^t$ merupakan keadaan sel-sel $\Omega_{x_{ij}}$ pada waktu t dan f sebagai fungsi yang menunjukkan aturan transisi.

Jika sel itu sendiri dianggap sebagai anggota dari tetangganya, maka persamaan (5.1) dapat ditulis sebagai :

$$S_{x_{i,j}}^{t+1} = f\left(S_{\Omega_{x_{i,j}}}^t\right). \quad (5.2)$$

Persamaan (5.2) dapat diekspresikan dalam bentuk verbal yang menggambarkan prinsip cellular automata, seperti berikut:

IF sesuatu terjadi pada lingkungan tetangga suatu sel
THEN sesuatu yang lain terjadi pada sel tersebut.

Model cellular automata selalu terdiri dari pernyataan “IF-THEN” sebagai implikasi aturan transisi khusus. Sebagai contoh, untuk model “Game of Life” diekspresikan menjadi tiga pernyataan “IF-THEN”:

IF	terdapat dua atau tiga sel hidup dalam <i>Moore Neighbourhood</i> dari suatu sel hidup,
THEN	sel akan bertahan hidup pada generasi berikutnya;
IF	terdapat kurang dari dua atau lebih dari tiga sel hidup dalam <i>Moore Neighbourhood</i> dari suatu sel hidup,
THEN	sel hidup tersebut akan mati pada generasi selanjutnya;
IF	tepat ada tiga sel hidup dalam <i>Moore Neighbourhood</i> dari suatu sel mati,
THEN	sel mati tersebut akan menjadi hidup pada generasi selanjutnya.

5.4. Contoh Cellular Automata

Cellular automata mampu men-*generate* karakteristik sistem yang cukup kompleks, bahkan untuk cellular automata satu-dimensi paling sederhana yang hanya mempunyai nilai biner keadaan. Hal ini mengindikasikan bahwa model sederhana seperti cellular automata berpotensi untuk mereproduksi fenomena kompleks.

Cellular automata juga memiliki karakteristik sistem terbuka yang mampu menjalankan *self-organization*. Pada proses ini, sistem meningkatkan pengorganisasian internal kompleksitasnya tanpa dipandu atau dikontrol sumber-sumber luar. Sistem dengan kemampuan *self-organising* ini lazimnya memperlihatkan sifat-sifat *emergence* di mana fenomena turbulensi maupun *chaos* mampu menghasilkan struktur dan pola yang unik dan koheren sebagai deskripsi dinamik sistem yang ditinjau.

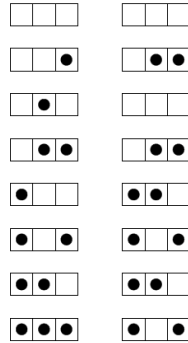
5.4.1. Cellular Automata 1-D

Misalkan kita memiliki suatu rantai sel biologi satu dimensi seperti di bawah:



Meski ukuran rantai sel dapat menuju tak berhingga, kita akan asumsikan bahwa ukuran rantai berhingga karena domain pemodelan sudah tentu berhingga. Setiap sel dapat berada dalam satu dari dua keadaan, *hidup* atau *mati* yang direpresentasi oleh nilai 1 dan 0.

Untuk setiap sel yang sedang ditinjau, tetangganya adalah sel di sisi kiri dan kanan yang langsung bersinggungan. Berikut adalah representasi grafis dari aturan-aturan yang akan menentukan keadaan sel-sel:



Dalam bentuk *array*, aturan di atas ditabelkan sebagai berikut:

Keadaan tetangga sekarang	Keadaan tetangga waktu berikutnya	Keadaan sel waktu berikutnya
0 0 0	0 0 0	0
0 0 1	0 1 1	1
0 1 0	0 0 0	0
0 1 1	0 1 1	1
1 0 0	1 1 0	1
1 0 1	1 0 1	0
1 1 0	1 1 0	1
1 1 1	1 0 1	0

Perhatikan bahwa suatu sel tetap hidup atau menjadi hidup jika hanya satu tetangganya juga hidup. Jika tak ada yang hidup maka sel akan mati karena terisolasi, atau jika kedua tetangganya hidup maka sel akan mati karena populasi berlebih. Misalkan kita tertarik untuk melihat keadaan sel setelah 50 generasi. Jika kita urut secara visual keadaan untuk 50 level waktu tersebut maka grafik yang dihasilkan membentuk pola segitiga yang dikenal dengan *Sierpinski triangle* atau *Sierpinski gasket* yang merupakan contoh sederhana suatu *fractal* (Gambar 5.1).

```

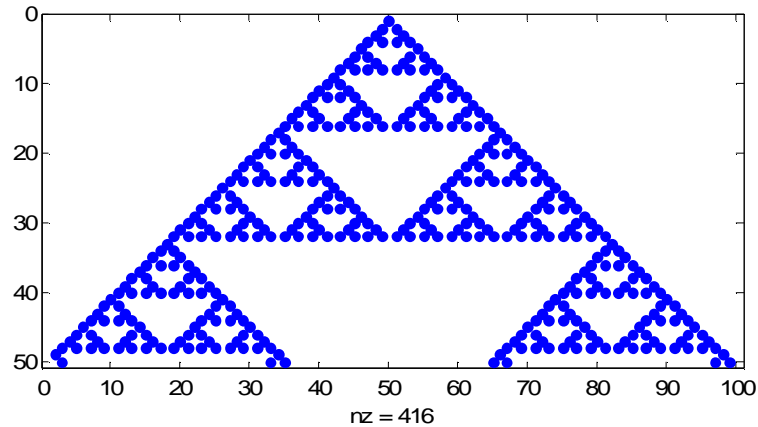
a=zeros(1,100); % siapkan domain model berupa rantai sel panjang 100, sekarang
newa=zeros(1,100); % siapkan yang akan datang
g=1;
max=50; % maksimum iterasi waktu
a(50)=1; % salah satu hidup
B(1,:)=a;
while (g<max),
for i=2:99,
if (a(i-1) + a(i+1))==1,
newa(i)=1; % aturan transisi
else
newa(i)=0; % aturan transisi
end
end
g=g+1;
a=newa;

```

```

B(g,:)=a;
end
spy(B)      % telusuri evolusinya

```



Gambar 5.1. Pola evolusi sistem rantai sel biologi satu dimensi.

5.4.2. Cellular Automata 2-D

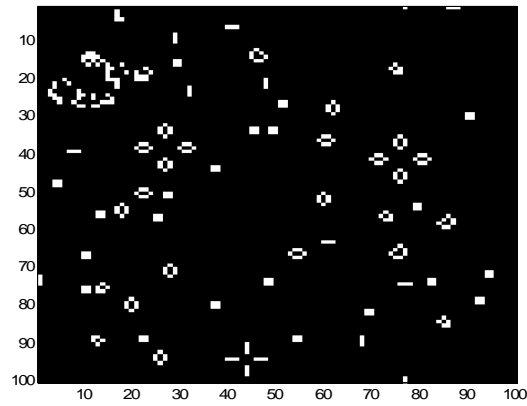
Berikut adalah *code* Matlab yang cukup sederhana namun sangat informatif .

```

% Game of Life
ngrid=100; % menyiapkan ukuran domain pemodelan
SEL=int8(rand(ngrid,ngrid)); % grid membentuk sistem sel berukuran 100x100
                                % dengan keadaan awal dibuat secara random
atas=[2:ngrid 1]; bawah=[ngrid 1:ngrid-1]; %domain pemodelan dibuat periodik: bulat
colormap(gray(2));
for i=1:1000 % iterasi waktu sekian kali
    tetangga=SEL(atas,:)+SEL(bawah,:)+SEL(:,atas)+SEL(:,bawah)+...
    SEL(atas,atas)+SEL(atas,bawah)+SEL(bawah,atas)+SEL(bawah,bawah);
    SEL = tetangga==3 | SEL & tetangga==2; % aturan transisi
    image(SEL*2); pause(0.01); % visualisasi
end

```

Coba fahami aturan transisi CA di atas, jalankan program di atas dan ubah-ubahlah parameter yang ada serta amati efeknya.



Gambar 5.2. Cellular Automata pada keadaan tertentu dari contoh kasus 2-D







VI. FRACTAL

6.1. Pendahuluan Fractal

Banyak struktur spasial di alam ini dihasilkan oleh *penyusunan-diri* (*self-assembly*) dari sejumlah besar komponen-komponen yang identik. Proses penyusunan-diri ini biasanya dinyatakan dalam aturan-aturan yang sederhana yang kita sebut sebagai prinsip-prinsip pengorganisasian. Dua buah prinsip yang paling sederhana adalah prinsip keteraturan (*regularity*) dan prinsip keacakan (*randomness*).

Keteraturan struktur spasial terlihat jika komponen-komponen struktur tersebut tersusun dalam suatu pola yang periodik maupun kuasi-periodik, seperti misalnya barisan tentara pada parade hari ulang tahun TNI serta susunan kristal-kristal pada material. Contoh keacakan misalnya pada struktur pada gas ataupun bulu-bulu binatang. Di antara dua pola ekstrim ini, terdapat suatu pola yang memenuhi prinsip *kemiripan-diri* (*self-similarity*) yang pada akhirnya menghasilkan struktur *self-similar* yang disebut *fractal*.

Pada sistem fractal, jika sebagian dari sistem diperbesar ke segala arah dengan proporsi pembesaran yang sama, maka bagian sistem itu akan menyerupai keseluruhan sistem. Suatu fractal biasanya memiliki dimensi fraksional. Konsep ini biasanya dicontohkan oleh gasket Sierpinski (SG) atau segitiga Sierpinski. Untuk membuat SG, langkah pertama ($n = 0$) adalah menetapkan segitiga samasisi dengan panjang sisi sama dengan satu (disebut *inisiator*). Langkah berikutnya ($n = 1$) adalah menghilangkan segitiga samasisi terbalik pada bagian tengah segitiga sebelumnya (akan terdapat 3 segitiga samasisi hitam, disebut *generator*). Pada $n = 2$, lakukan hal yang sama pada ketiga segitiga hitam sebelumnya. Seterusnya, ulangi prosedur pengurangan segitiga tengah terbalik ini (yang hasilnya disebut prefactal) hingga $n = \infty$ (tentu saja secara abstraksi di kepala). Himpunan segitiga yang terbentuk pada $n = \infty$ disebut gasket Sierpinski. Perhatikanlah bahwa setiap bagian SG akan memiliki bentuk yang sama seperti keseluruhannya (Gambar 6.1). Dapatlah disimpulkan bahwa SG merupakan fractal *self-similar*.

prefractal	n	ϵ	N_ϵ
	0	1	1
	1	2^{-1}	3
	2	2^{-2}	3^2
	3	2^{-3}	3^3
\vdots	\vdots	\vdots	\vdots
\vdots	\vdots	\vdots	\vdots
\vdots	n	2^{-n}	3^n

Gambar 6.1. Konstruksi dan penentuan dimensi fractal Gasket Sierpinski.

Dimensi fractal D dari suatu objek diberikan oleh hubungan

$$N_\epsilon \sim \epsilon^{-D} \quad (6.1)$$

dengan N_ε berupa jumlah minimal objek-objek lebih kecil identik (masing-masing memiliki ukuran linier ε). Simbol \sim (tilde) dapat diartikan “proporsional jika $\varepsilon \rightarrow 0$.” Dimensi D pada persamaan di atas disebut juga dimensi kotak (*box dimension*) yang nilainya ekuivalen dengan

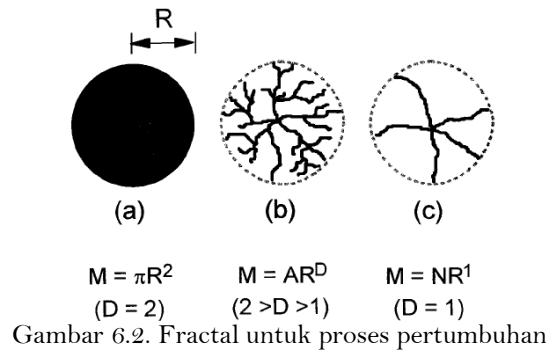
$$D = \lim_{\varepsilon \rightarrow 0} \left(\frac{\log N_\varepsilon}{\log (1/\varepsilon)} \right), \quad (6.2)$$

sebagai contoh pada SG, untuk $\varepsilon = 1/2^n$ maka $N_\varepsilon = 3^n$ sehingga $D = \log 3 / \log 2 \approx 1.585$.

Pada kasus di mana fractal dibangun oleh suatu proses pertumbuhan, maka kita dapat menentukan dimensinya melalui hubungan

$$M \sim R^D \quad (6.3)$$

di mana M adalah massa suatu objek ketika dimensi liniernya sama dengan R . Simbol \sim berarti “proporsional jika $R \rightarrow \infty$.” Biasanya dimensi yang didapat dengan menggunakan (6.1) akan sama dengan (6.3). Ilustrasi fractal proses pertumbuhan diperlihatkan pada Gambar 6.2,



pada gambar di atas M adalah massa objek yang nilainya sebanding dengan luas lingkaran hitam, R adalah jari-jari yang merupakan ukuran linier objek. Besaran A pada (b) adalah suatu konstanta; N adalah jumlah garis pada (c). Pada (a), objek yang tumbuh adalah lingkaran penuh; pada (b), objek yang tumbuh adalah “pohon” fractal bercabang; pada (c) objek yang tumbuh adalah pohon fractal sederhana tak bercabang. Dapat juga disimpulkan dari ilustrasi di atas bahwa “percabangan” adalah komponen utama pembentuk pohon fractal.

Konsep fractal pertama kali diperkenalkan pada awal tahun 1980an oleh Benoit Mandelbrot Melalui bukunya *The Fractal Geometry of Nature* (1982). Contoh-contoh fractal sangatlah luas meliputi agregat dan koloid; pepohonan; batuan; awan; pegunungan; galaksi; polimer; retakan; pasar modal; dan masih banyak lainnya.

Bila untuk menyerupai objek keseluruhan itu ternyata diperlukan magnifikasi sebagian objek dengan nilai magnifikasi yang berbeda untuk arah-arrah yang berbeda, maka objek tersebut termasuk dalam *self-affine* fractal. Contoh fractal jenis ini adalah bidang-bidang antarmuka serta permukaan kasar suatu material. Fractal dapat pula berbentuk multifractal, yaitu terdiri dari kumpulan banyak fractal.

Kunci pemahaman fractal terletak pada perilaku hukum pangkatnya, seperti yang tercermin pada (6.1) dan (6.3). Ungkapan

$$y = Ax^a \quad (6.4)$$

ekivalen dengan

$$y(\lambda x) = \lambda^a y(x), \text{ untuk semua } \lambda > x \quad (6.5)$$

Karena λ bernilai sebarang maka jika kita pilih $\lambda = 1/x$ maka (6.5) tereduksi menjadi $y(x) = y(1)x^a$ yang merupakan (6.4) dengan $A = y(1)$. Dengan demikian kedua persamaan di atas ekuivalen satu sama lain.

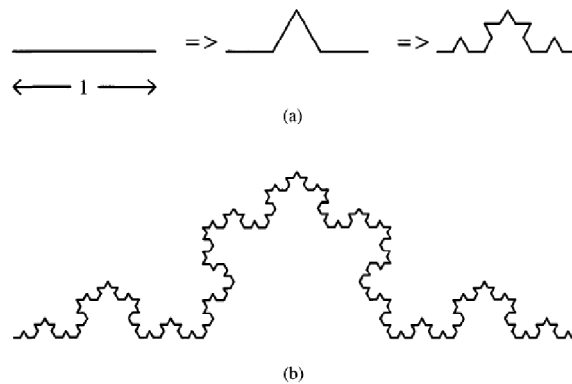
Dari sudut pandang matematik, setiap fungsi $y(x)$ yang memenuhi (6.5) disebut fungsi homogen. Setiap fungsi homogen akan bersifat invarian terhadap perubahan skala. Jika skala ukuran x diubah sehingga $x \rightarrow x' (\equiv \lambda x)$, maka fungsi yang baru $y'(x') [\equiv y(x)]$ akan memiliki bentuk yang sama dengan fungsi $y(x)$ yang sebelumnya. Hal ini dapat dilihat melalui hubungan $y(x) = \lambda^{-a} y(x')$ dari (6.5) sehingga $y'(x') \sim y(x)$.

Suatu sistem bersifat invarian skala berarti jika sebagian dari sistem itu diperbesar hingga skala sistem awal atau aslinya maka keduanya akan terlihat sama satu dan lainnya. Dapatlah dikatakan bahwa sistem aslinya tidak memiliki skala intrinsic. Suatu sistem yang bersifat invarian skala maka akan bersifat *self-similar* dan sebaliknya.

Kita simpulkan bahwa *self-similarity*, hukum pangkat spasial dan invarian skala adalah tiga cara untuk menyatakan bahwa suatu sistem tidak memiliki skala panjang karakteristik. Jika konsep ini kita perluas, maka suatu sistem yang tidak memiliki skala waktu karakteristik maka sistem tersebut dapat dinyatakan dalam hukum pangkat temporal (misalnya noise $1/f$ yang sering hadir di alam). Perlu diingat pula bahwa hukum pangkat berbentuk non-linier dan hanya linier jika eksponennya bernilai satu.

6.2. Fractal Deterministik

Fractal deterministik adalah struktur atau sistem geometri ideal yang memiliki sifat bahwa bagian dari struktur tersebut sama dengan keseluruhannya (*self-similarity*). Untuk lebih memahami konsep fractal kita lihat beberapa contoh berikut.

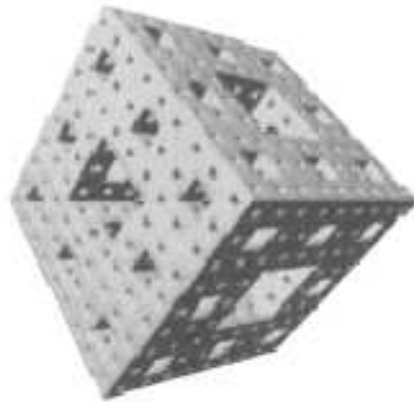


Gambar 6.3. Kurva Koch. (a) Penyusunan kurva: inisiator berupa sebuah segmen; generator tersusun dengan mengganti $1/3$ segmen bagian tengah dengan dua buah $1/3$ segmen; seterusnya berulang. (b) Kurva Koch setelah empat iterasi.

Gambar 6.3 memperlihatkan sistem fractal berupa kurva Koch dan konstruksinya. Inisiatornya berupa satu satuan segmen, generatornya tersusun dengan mengganti $1/3$ segmen bagian tengah dengan dua buah $1/3$ segmen membentuk tenda, proses ini terus berulang sampai tak hingga. Meskipun limit panjangnya dapat menjadi tak berhingga, tetap saja kurva tersebut terkendala untuk berada pada suatu bidang berhingga. Dapat diambil kesimpulan bahwa Kurva Koch akan “lebih padat” dibanding kurva biasa yang berdimensi D

$= 1$ namun “lebih jarang” dari benda objek berdimensi dua karena luasnya sama dengan nol. Dimensi untuk kurva Koch tersebut haruslah $1 < D < 2$. Objek-objek teratur seperti garis, bujursangkar atau kubus yang berdimensi D jika diperbesar dengan faktor pembesaran b akan cocok menempati objek yang sudah diperbesarnya sebanyak b^D kali. Pada kurva Koch, suatu segmen jika diperbesar dengan faktor 3 akan cocok dengan 4 buah segmen aslinya. Dimensi fractal-nya diberikan oleh $3^D = 4$ sehingga $D = \log 4 / \log 3 \approx 1.262$.

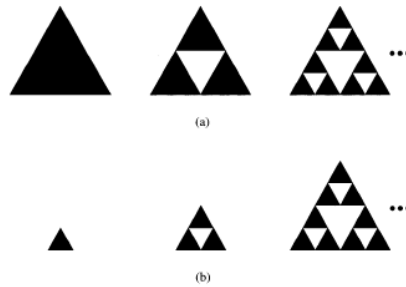
Kembali ke contoh SG, sebagian dari SG (berupa sebuah segitiga samasisi), jika diperbesar 2 kali akan menempati segitiga aslinya sebanyak $(4 - 1)$ kali, sehingga $2^D = 3$ atau $D = \log 3 / \log 2 \approx 1.585$.



Gambar 6.4. Sponge Sierpinski setelah tiga kali iterasi. Inisiatornya berupa satu satuan kubus, generasinya tersusun dari $3 \times 3 \times 3$ kubus yang lebih kecil, 7 satuan kubus lebih kecil tadi dihilangkan (1 di tengah dan 6 tetangga terdekatnya dihilangkan) dan seterusnya diulang (ben-Avraham and Havlin, 2004).

Sponge Sierpinski (sponge Menger) adalah sistem fractal yang terbentuk dari sebuah kubus yang dibagi menjadi $3 \times 3 \times 3 = 27$ kubus kecil. Kubus kecil di tengah dan enam buah tetangga terdekatnya dibuang, proses serupa diulang pada 20 kubus kecil yang tersisa dan seterusnya hingga tak berhingga iterasi. Limit volumenya akan mendekati nol namun luasnya tak berhingga ($2 < D < 3$). Dimensi fractalnya didapat dari $3^D = 20$ atau $D = \log 20 / \log 3 \approx 2.727$.

Secara umum, semua kisi-kisi fractal deterministik didapat dengan cara yang sama dengan contoh-contoh di atas. Konstruksi fractal dimulai dari suatu genus yang disebut inisiator, kemudian diterapkan operasi rekursif iteratif tak berhingga yang disebut generator. Generator dapat digolongkan menjadi dua tipe (Gambar 6.5). Pada tipe pertama, inisiator diganti dengan replika-replika lebih kecil dan fractal dibangun ‘ke dalam’ (*inward*) membentuk skala-skala yang lebih kecil. Fractal tipe ini memiliki panjang skala ambang atas berupa inisiator tetapi tidak memiliki skala mikroskopik karakteristik. Generator tipe kedua dibangun dengan membuat replika inisiator yang kemudian ditempatkan pada objek lebih besar dan fractal dibangun ‘ke luar’ (*outward*). Fractal ini akan memiliki panjang ambang bawah namun tidak memiliki skala makroskopik karakteristik. Suatu fractal ideal seyogyanya merupakan kombinasi dua tipe di atas (tidak memiliki panjang ambang). Namun demikian, perlu disadari bahwa objek-objek yang nyata serta fractal yang dibangun dengan komputer tentu saja memiliki ambang-ambang ini yang menyatakan ukuran struktur fractal serta satuan-satuan elementernya.



Gambar 6.5. Generator SG: Inward generator (a) dan outward generator (b).

6.3. Sifat-sifat Fractal

Seperti telah dibahas pada bagian pendahuluan, sifat terpenting dari fractal adalah *self-similarity* yang berarti keterjagaan simetri di bawah operasi dilatasi. Jika kita perhatikan kurva Koch (Gambar 6.3) yang sering disebut dengan *snowflake* (lihat 6.3 Latihan), pada bagian tengahnya terdapat sosok mirip boneka salju. Pada sisi sebelah kiri dan kanan boneka salju tersebut terdapat masing-masing boneka salju lagi, hanya kali ini lebih kecil dengan faktor $1/3$. Di sebelah kiri dan kanan boneka-boneka salju kecil ini terdapat pula masing-masing boneka salju yang lebih kecil dan seterusnya. Jika kita lihat kurva Koch pada suatu perbesaran tertentu atau sebarang maka kita akan melihat motif yang sama berulang dan seterusnya. Kita akan melihat fenomena yang sama untuk SG seperti telah dibahas sebelumnya.

Perbedaan utama antara ruang Euclidian biasa dengan geometri fractal terletak pada hal berikut: ruang Euclidian biasa bersifat simetri di bawah operasi translasi, sedangkan geometri fractal melanggar aturan simetri ini, bahkan melahirkan konsep simetri yang baru yaitu invarian skala (invarian terhadap operasi dilatasi) seperti telah dibahas pada bagian pendahuluan. Inilah yang dimaksud dengan *self-similarity*, yang membuat konsep fractal bermanfaat pada kajian-kajian transisi fasa di mana keadaan suatu sistem pada titik transisi kritisnya dikarakterisasi oleh simetri dilatasi.

Deskripsi objek-objek fractal tidak sepenuhnya lengkap hanya dengan konsep dimensi fractal D . Karakteristik penting fractal yang lain adalah ramifikasi atau percabangan. Suatu fractal memiliki percabangan berhingga (*finitely ramified*) jika setiap bagian yang terikat pada fractal tersebut dapat diisolasi dengan memotong sejumlah ikatan berhingga. Dengan demikian tidaklah sulit untuk memahami bahwa gasket Sierpinski dan kurva Koch termasuk *finitely ramified* sedangkan sponge Sierpinski bersifat *infinitely ramified*.

Karakteristik lainnya sering juga diketengahkan untuk mengetahui sifat-sifat khusus dari fractal. "Lakunaritas" (*lacunarity*) digunakan untuk menganalisis derajat homogenitas suatu fractal dan bersifat invarian translasi. "Dimensi spektral" adalah eksponen yang digunakan untuk penskalaan rapat keadaan terkait operator Laplace dalam suatu fractal. Eksponen "lintasan terpendek" digunakan untuk mengkarakterisasi panjang lintasan terpendek yang menghubungkan dua buah titik dalam fractal.

6.4. Fractal Acak

Fractal tidaklah harus secara kaku selalu bersifat deterministik. Unsur-unsur stokastik dapat pula dilibatkan ke dalam generatornya. Contoh sederhananya adalah karpet Sierpinski (Gambar 6.6) yang merupakan satu permukaan sponge Sierpinski. Gbr. 6.6(a) dibuat oleh inisiator berukuran satu satuan bujursangkar yang kemudian dibagi ke dalam 3×3 bujursangkar kemudian bujursangkar bagian tengah dihilangkan. Jika bujursangkar yang dihilangkan bukan selalu yang tengah tetapi dipilih secara acak maka akan dihasilkan Gbr

6.6(b). Kedua gambar terlihat berhubungan namun Gbr 6.6(b) tidaklah *self-similar* secara eksak namun dalam pengertian statistik tetap *self-similar*. Terlihat bahwa distribusi total daerah lubang (warna putih) sama pada semua skala panjang sub-bujursangkar yang sama. Terlihat juga bahwa massa objek (daerah hitam) meningkat dengan faktor 8 ketika ruang terdilatasi dengan faktor 3. Oleh karenanya dimensi fractal $D = \log 8 / \log 3$, sama dengan dimensi fractal deterministiknya. Secara umum, massa M dari suatu fractal acak yang mengalami perbesaran dengan faktor b akan berbentuk

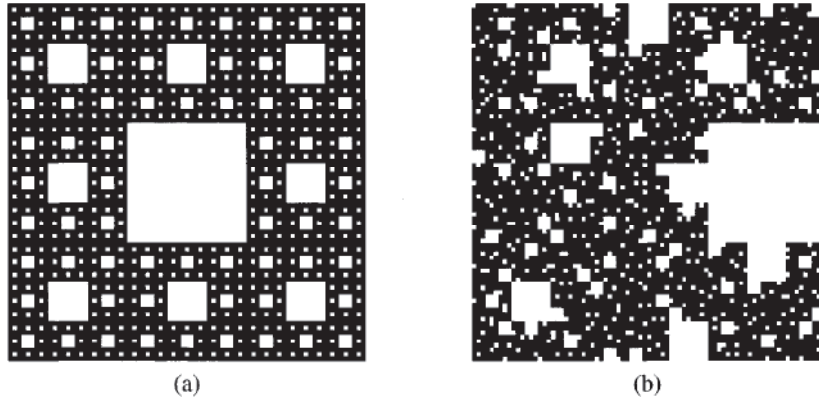
$$M(bR) = b^D M(R) \quad (6.6)$$

yang sama dengan bentuk deterministiknya dan tak lain adalah (6.5) yang solusinya berbentuk

$$M(R) = AR^D \quad (6.7)$$

yang tak lain adalah (6.4).

Fractal acak sangat bermanfaat dalam memodelkan fenomena-fenomena alam. Yakinkah Anda bahwa bentuk permukaan sponge yang sesungguhnya yang sering digunakan mencuci piring lebih mendekati Gbr 6.6(b) dibanding Gbr 6.6(a)? Pembentukan generator kurva Koch dengan unsur stokastik kemungkinan juga dapat menyediakan deskripsi yang baik mengenai garis pantai Pulau Kalimantan. Mandelbrot serta para perintis lainnya telah berjasa menginformasikan kepada kita bahwa fenomena-fenomena alam lebih dekat kepada konsep fractal dibanding sebaliknya.



Gambar 6.6. Karpet Sierpinski. (a) Deterministik generator, (b) stokastik generator di mana bujursangkar yang dihilangkan bukan yang berada di tengah namun dipilih secara acak.

Dimensi fractal dari suatu fractal acak biasanya didapat secara numerik, baik dari perbesaran massa dengan ukuran linier (6.6) yang disebut juga algoritma *sand-box* maupun dari algoritma *box-counting*. Pada algoritma pertama kita menghitung massa yang berada dalam radius R di sekitar titik yang dimiliki fractal. Dimensi fractal didapat dengan perata-rataan berbagai titik dan dengan menggunakan (6.7). Pada algoritma kedua, ruang pembentuk fractal dibagi ke dalam grid-grid yang membentuk sel-sel dengan ukuran linier ε . Jumlah sel (kotak) yang mengandung bagian fractal (N_ε) kemudian dihitung. Prosedur ini diulangi untuk semua kotak dan kemudian D dihitung dari hubungan (6.1) dan (6.2).

Penskalaan massa dapat juga dianalisis melalui fungsi korelasi densitas-densitas:

$$c(\mathbf{r}) = \frac{1}{V} \sum_{\mathbf{r}'} \rho(\mathbf{r}') \rho(\mathbf{r}' + \mathbf{r}) \quad (6.8)$$

di mana $\rho(\mathbf{r}') = 1$ jika \mathbf{r}' merupakan bagian dari fractal dan 0 jika bukan. $V = \sum_{\mathbf{r}'} \rho(\mathbf{r}')$

adalah faktor normalisasi. Dengan demikian, $c(\mathbf{r})$ adalah rapat massa di sekitar \mathbf{r} dari suatu titik sebarang di dalam fractal. Untuk fractal yang isotropik dan *self-similar*, rapat massa akan berbentuk

$$c(\mathbf{r}) = c(r) \sim r^{-\alpha}. \quad (6.9)$$

Jika fractal berada dalam ruang berdimensi d , massanya dalam ukuran linier R adalah

$$M(R) \sim \int_0^R c(r) d^d r \sim R^{d-\alpha}, \quad (6.10)$$

karena $M(R) \sim R^D$, maka

$$\alpha = d - D. \quad (6.11)$$

Pengukuran D melibatkan digitalisasi data eksperimental yang tersedia. Dalam kasus di mana sistem yang dikaji melibatkan fenomena hamburan, D lebih mudah didapat langsung dari eksperimen hamburan. Intensitas hamburan proporsional terhadap faktor struktur $S(\mathbf{q})$ yang merupakan transform Fourier dari rapat massa $c(r)$. Untuk fractal acak isotropis

$$S(\mathbf{q}) = S(q) \sim q^{-D}. \quad (6.12)$$

Untuk fractal-fractal alami, kebergantungan hukum pangkat massa terhadap jarak hanya berlaku dalam skala-skala panjang ambang β_- dan β_+ dan (6.12) hanya berlaku untuk $1/\beta_+ < q < 1/\beta_-$.

6.5. Fractal *Self-Affine*

Seperti telah dibahas pada bagian pendahuluan, objek fractal dapat memiliki simetri dilatasi anisotropik atau bersifat *self-affine*. Gambar 6.7 adalah contoh fractal jenis ini. Inisiatornya berupa satu satuan bujursangkar. Generatornya terdiri dari pembagian inisiator ke dalam $b_1 \times b_2$ sel-sel persegi panjang di mana hanya n sel disisakan sedangkan $(b_1 b_2 - n)$ sel dibuang. Pada contoh kita $b_1 = 3$, $b_2 = 2$ dan $n = 3$. Objek ini mengalami perbesaran melalui dilatasi sepanjang arah horizontal (x) dengan faktor 3 dan arah vertikal (y) dengan faktor 2.

Berapakah dimensi fractal untuk fractal yang *self-affine*? Jika digunakan algoritma *box-counting* dan (6.1) maka akan terlihat D bervariasi sebagai fungsi skala panjang. Pada satu sisi, pada saat ukuran kotak mengecil hingga mendekati nol maka

$$D_{\text{lokal}} = \lim_{\varepsilon \rightarrow 0} \left(\frac{\log N_\varepsilon}{\log(1/\varepsilon)} \right) = \frac{\log(nb_1/b_2)}{\log b_1} \quad (6.13)$$

yang merupakan dimensi fractal lokal. Pada sisi yang lain, untuk kotak yang tumbuh membesar

$$D_{\text{global}} = \lim_{\varepsilon \rightarrow \infty} \left(\frac{\log N_\varepsilon}{\log(1/\varepsilon)} \right) = \frac{\log(nb_2/b_1)}{\log b_2} \quad (6.14)$$

yang merupakan dimensi fractal global ketika skala panjang memiliki nilai terbesar.

Cara lain adalah dengan meninjau skala anisotropik pada masing-masing arah yang berbeda,

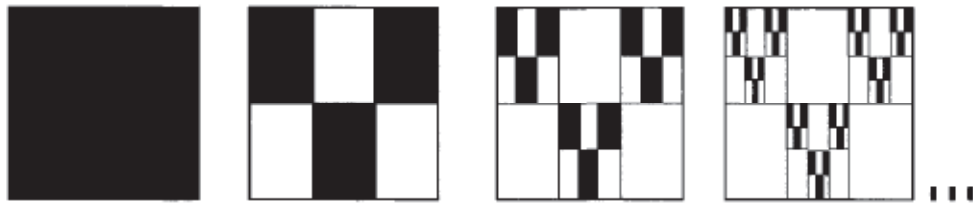
$$M\left(b^{1/D_x} R_x, b^{1/D_y} R_y\right) = b M\left(R_x, R_y\right). \quad (6.15)$$

Untuk contoh pada Gambar 6.7,

$$M\left(b_1 R_x, b_2 R_y\right) = n M\left(R_x, R_y\right) \quad (6.16)$$

sehingga $D_x = \log n / \log b_1$ dan $D_y = \log n / \log b_2$.

Fractal yang bersifat acak dan *affine* banyak ditemui di alam. Lanskap pegunungan, bidang antarmuka dan permukaan merupakan fractal jenis ini. Kajian tentang pertumbuhan permukaan juga banyak memanfaatkan konsep fractal ini.



Gambar 6.7. Contoh fractal *self-affine*

6.6. Latihan

- Gunakan kamera (hp) Anda, carilah fenomena di sekitar yang memenuhi prinsip fractal, baik artifisial maupun alami (paling sedikit 5 foto).
- Carilah pola retakan pada tanah atau dinding yang menurut Anda memenuhi konsep fractal (foto jika perlu). Hitunglah dimensi fractalnya.
- Berikut adalah beberapa fungsi untuk mengenerate fractal yang ditulis oleh Jonas Lundgren (2010) dan Moses Boone (sponge.m, 2010) dalam Matlab. Untuk setiap fungsi, deskripsikan aturan pembuatan fractal tersebut beserta algoritmanya.

a. `sierpinski.m`

```
function z = sierpinski(n)
%SIERPINSKI Sierpinski Cross Curve
% Z = SIERPINSKI(N) is a closed curve in the complex plane
% with 4^(N+1)+1 points. N is a nonnegative integer.
%
% % Example
% plot(sierpinski(4)), axis equal

% Author: jonas.lundgren@saabgroup.com, 2010.

% Constants
a = 1 + 1i;
b = 1 - 1i;
c = 2 - sqrt(2);

% Generate point sequence
z = c;
for k = 1:n
```

```

w = 1i*z;
z = [z+b; w+b; a-w; z+a]/2;
end

```

```

% Close cross
z = [z; 1i*z; -z; -1i*z; z(1)];

```

b. arrowhead.m

```

function z = arrowhead(n)
%ARROWHEAD Sierpinski Arrowhead Curve
% Z = ARROWHEAD(N) is a continuous curve in the complex plane
% with 3^N+1 points. N is a nonnegative integer.
%
% % Example
% plot(arrowhead(7))

```

```

% Author: jonas.lundgren@saabgroup.com, 2010.

```

```

% Constants
a = (1 + sqrt(-3))/2;
b = (1 - sqrt(-3))/2;

```

```

% Generate point sequence
z = 0;
for k = 1:n
    w = conj(z);
    z = [a*w; z+a; b*w+a+1]/2;
end

```

```

% Add endpoint
z = [z; 1];

```

c. dragon.m

```

function z = dragon(n)
%DRAGON Dragon Curve
% Z = DRAGON(N) is a continuous curve in the complex plane
% with 2^(N+1) points. N is a nonnegative integer.
%
% % Example
% z = dragon(12);
% figure(1), plot(z), axis equal
% figure(2), plot(reshape(z,[],4)), axis equal

```

```

% Author: jonas.lundgren@saabgroup.com, 2010.

```

```

% Constants
a = (1 + 1i)/2;
b = (1 - 1i)/2;
c = sqrt(1/2);

```

```

% Generate point sequence
z = [1-c; c];
for k = 1:n

```



```

w = z(end:-1:1);
z = [a*z; 1-b*w];
end

```

d. flowsnake.m

```

function z = flowsnake(n)
%FLOWSNAKE Gosper Flowsnake Curve
% Z = FLOWSNAKE(N) is a continuous curve in the complex plane
% with 7^N+1 points. N is a nonnegative integer.
%
% % Example
% plot(flowsnake(4)), axis equal

% Author: jonas.lundgren@saabgroup.com, 2010.

% Constants
a = (1 + sqrt(-3))/2;
b = (1 - sqrt(-3))/2;
c = [1; a; -b; -1; -a; b];

% Segment angles (divided by pi/3)
u = 0;
for k = 1:n
    v = u(end:-1:1);
    u = [u; v+1; v+3; u+2; u; u; v-1];
end
u = mod(u,6);

% Points
z = cumsum(c(u+1));
z = [0; z/7^(n/2)];

```

e. snowflake.m

```

function z = snowflake(n,a)
%SNOWFLAKE Koch Snowflake Curve
% Z = SNOWFLAKE(N,A) is a closed curve in the complex plane
% with 3*2^N+1 points. N is a nonnegative integer and A is a
% complex number with |A| < 1 and |1-A| < 1.
% Default is A = 1/2 + i*sqrt(3)/6.
%
% % Examples
% plot(snowflake(10)), axis equal
% plot(snowflake(10,0.45+0.35i)), axis equal

% Author: jonas.lundgren@saabgroup.com, 2010.

if nargin < 1, n = 0; end
if nargin < 2, a = 1/2 + sqrt(-3)/6; end

% Constants
b = 1 - a;
c = 1/2 + sqrt(-3)/2;
d = 1 - c;

```

```
% Generate point sequence
```

```
z = 1;
for k = 1:n
    z = conj(z);
    z = [a*z; b*z+a];
end
```

```
% Close snowflake
```

```
z = [0; z; 1-c*z; 1-c-d*z];
```

f. sponge.m

```
% This function generates the Sierpinski Sponge.
```

```
% n=level of iterations
```

```
function Sponge(n)
```

```
if (n==0)
```

```
    vertices=[0 0 0;1 0 0;1 1 0;0 1 0;0 0 1;1 0 1;1 1 1;0 1 1];
    faces=[1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
    patch('Vertices',vertices,'Faces',faces,'FaceVertexCData',hsv(6),...
        'FaceColor','flat')
    good_axis
```

```
else
```

```
    levelcontrol=10^n;
    L=(levelcontrol/(3^n));
    l=ceil(L);
    carp(0,0,0,levelcontrol,0,0,levelcontrol,levelcontrol,0,0,...
        levelcontrol,0,0,0,levelcontrol,levelcontrol,0,levelcontrol,...
        levelcontrol,levelcontrol,levelcontrol,0,levelcontrol,...
        levelcontrol,l)
    good_axis;
```

```
end
```

```
%-----
```

```
function carp(x1,y1,z1,x4,y4,z4,x52,y52,z52,x49,y49,z49,x13,y13,z13,x16,y16,...
    z16,x64,y64,z64,x61,y61,z61,limit)
```

```
if(abs(x1-x4)>limit|abs(x16-x4)>limit|abs(x16-x13)>limit|...
    abs(x13-x1)>limit|abs(x1-x49)>limit|abs(x61-x49)>limit|...
    abs(x61-x13)>limit|abs(x64-x16)>limit|abs(x64-x52)>limit|...
    abs(x52-x4)>limit|abs(x64-x61)>limit|abs(x52-x49)>limit|...
    abs(y1-y4)>limit|abs(y16-y4)>limit|abs(y16-y13)>limit|...
    abs(y13-y1)>limit|abs(y1-y49)>limit|abs(y61-y49)>limit|...
    abs(y61-y13)>limit|abs(y64-y16)>limit|abs(y64-y52)>limit|...
    abs(y52-y4)>limit|abs(y64-y61)>limit|abs(y52-y49)>limit|...
    abs(z1-z4)>limit|abs(z16-z4)>limit|abs(z16-z13)>limit|...
    abs(z13-z1)>limit|abs(z1-z49)>limit|abs(z61-z49)>limit|...
    abs(z61-z13)>limit|abs(z64-z16)>limit|abs(z64-z52)>limit|...
    abs(z52-z4)>limit|abs(z64-z61)>limit|abs(z52-z49)>limit)
```

```
a=abs((x4-x1)/3);
b=abs((y49-y1)/3);
c=abs((z13-z1)/3);
```

```
x2=x1+a; y2=y1; z2=z1;
x3=x1+2*a; y3=y1; z3=z1;
```

```

x5=x1;   y5=y1;   z5=z1+c;
x6=x1+a; y6=y1;   z6=z1+c;
x7=x1+2*a; y7=y1;   z7=z1+c;
x8=x4;   y8=y1;   z8=z1+c;
x9=x1;   y9=y1;   z9=z1+2*c;
x10=x1+a; y10=y1;   z10=z1+2*c;
x11=x1+2*a; y11=y1;   z11=z1+2*c;
x12=x4;   y12=y1;   z12=z1+2*c;
x14=x1+a; y14=y1;   z14=z13;
x15=x1+2*a; y15=y1;   z15=z13;
x17=x1;   y17=y1+b; z17=z1;
x18=x1+a; y18=y1+b; z18=z1;
x19=x1+2*a; y19=y1+b; z19=z1;
x20=x4;   y20=y1+b; z20=z1;
x21=x1;   y21=y1+b; z21=z1+c;
x22=x1+a; y22=y1+b; z22=z1+c;
x23=x1+2*a; y23=y1+b; z23=z1+c;
x24=x4;   y24=y1+b; z24=z1+c;
x25=x1;   y25=y1+b; z25=z1+2*c;
x26=x1+a; y26=y1+b; z26=z1+2*c;
x27=x1+2*a; y27=y1+b; z27=z1+2*c;
x28=x4;   y28=y1+b; z28=z1+2*c;
x29=x1;   y29=y1+b; z29=z13;
x30=x1+a; y30=y1+b; z30=z13;
x31=x1+2*a; y31=y1+b; z31=z13;
x32=x4;   y32=y1+b; z32=z13;
x33=x1;   y33=y1+2*b; z33=z1;
x34=x1+a; y34=y1+2*b; z34=z1;
x35=x1+2*a; y35=y1+2*b; z35=z1;
x36=x4;   y36=y1+2*b; z36=z1;
x37=x1;   y37=y1+2*b; z37=z1+c;
x38=x1+a; y38=y1+2*b; z38=z1+c;
x39=x1+2*a; y39=y1+2*b; z39=z1+c;
x40=x4;   y40=y1+2*b; z40=z1+c;
x41=x1;   y41=y1+2*b; z41=z1+2*c;
x42=x1+a; y42=y1+2*b; z42=z1+2*c;
x43=x1+2*a; y43=y1+2*b; z43=z1+2*c;
x44=x4;   y44=y1+2*b; z44=z1+2*c;
x45=x1;   y45=y1+2*b; z45=z13;
x46=x1+a; y46=y1+2*b; z46=z13;
x47=x1+2*a; y47=y1+2*b; z47=z13;
x48=x4;   y48=y1+2*b; z48=z13;
x50=x1+a; y50=y49; z50=z1;
x51=x1+2*a; y51=y49; z51=z1;
x53=x1;   y53=y49; z53=z1+c;
x54=x1+a; y54=y49; z54=z1+c;
x55=x1+2*a; y55=y49; z55=z1+c;
x56=x4;   y56=y49; z56=z1+c;
x57=x1;   y57=y49; z57=z1+2*c;
x58=x1+a; y58=y49; z58=z1+2*c;
x59=x1+2*a; y59=y49; z59=z1+2*c;
x60=x4;   y60=y49; z60=z1+2*c;
x62=x1+a; y62=y49; z62=z13;

```

```

x63=x1+2*a; y63=y49; z63=z13;

    carp(x1,y1,z1,x2,y2,z2,x18,y18,z18,x17,y17,z17,x5,y5,z5,x6,y6,z6,...
x22,y22,z22,x21,y21,z21,limit);
    carp(x2,y2,z2,x3,y3,z3,x19,y19,z19,x18,y18,z18,x6,y6,z6,x7,y7,z7,...
x23,y23,z23,x22,y22,z22,limit);
    carp(x3,y3,z3,x4,y4,z4,x20,y20,z20,x19,y19,z19,x7,y7,z7,x8,y8,z8,...
x24,y24,z24,x23,y23,z23,limit);
    carp(x17,y17,z17,x18,y18,z18,x34,y34,z34,x33,y33,z33,x21,y21,z21,...
x22,y22,z22,x38,y38,z38,x37,y37,z37,limit);
    carp(x19,y19,z19,x20,y20,z20,x36,y36,z36,x35,y35,z35,x23,y23,z23,...
x24,y24,z24,x40,y40,z40,x39,y39,z39,limit);
    carp(x33,y33,z33,x34,y34,z34,x50,y50,z50,x49,y49,z49,x37,y37,z37,...
x38,y38,z38,x54,y54,z54,x53,y53,z53,limit);
    carp(x34,y34,z34,x35,y35,z35,x51,y51,z51,x50,y50,z50,x38,y38,z38,...
x39,y39,z39,x55,y55,z55,x54,y54,z54,limit);
    carp(x35,y35,z35,x36,y36,z36,x52,y52,z52,x51,y51,z51,x39,y39,z39,...
x40,y40,z40,x56,y56,z56,x55,y55,z55,limit);
    carp(x5,y5,z5,x6,y6,z6,x22,y22,z22,x21,y21,z21,x9,y9,z9,x10,y10,...
x10,x26,y26,z26,x25,y25,z25,limit);
    carp(x7,y7,z7,x8,y8,z8,x24,y24,z24,x23,y23,z23,x11,y11,z11,x12,...
y12,z12,x28,y28,z28,x27,y27,z27,limit);
    carp(x37,y37,z37,x38,y38,z38,x54,y54,z54,x53,y53,z53,x41,y41,z41,...
x42,y42,z42,x58,y58,z58,x57,y57,z57,limit);
    carp(x39,y39,z39,x40,y40,z40,x56,y56,z56,x55,y55,z55,x43,y43,z43,...
x44,y44,z44,x60,y60,z60,x59,y59,z59,limit);
    carp(x9,y9,z9,x10,y10,z10,x26,y26,z26,x25,y25,z25,x13,y13,z13,x14,...
y14,z14,x30,y30,z30,x29,y29,z29,limit);
    carp(x10,y10,z10,x11,y11,z11,x27,y27,z27,x26,y26,z26,x14,y14,z14,...
x15,y15,z15,x31,y31,z31,x30,y30,z30,limit);
    carp(x11,y11,z11,x12,y12,z12,x28,y28,z28,x27,y27,z27,x15,y15,z15,...
x16,y16,z16,x32,y32,z32,x31,y31,z31,limit);
    carp(x25,y25,z25,x26,y26,z26,x42,y42,z42,x41,y41,z41,x29,y29,z29,...
x30,y30,z30,x46,y46,z46,x45,y45,z45,limit);
    carp(x27,y27,z27,x28,y28,z28,x44,y44,z44,x43,y43,z43,x31,y31,z31,...
x32,y32,z32,x48,y48,z48,x47,y47,z47,limit);
    carp(x41,y41,z41,x42,y42,z42,x58,y58,z58,x57,y57,z57,x45,y45,z45,...
x46,y46,z46,x62,y62,z62,x61,y61,z61,limit);
    carp(x42,y42,z42,x43,y43,z43,x59,y59,z59,x58,y58,z58,x46,y46,z46,...
x47,y47,z47,x63,y63,z63,x62,y62,z62,limit);
    carp(x43,y43,z43,x44,y44,z44,x60,y60,z60,x59,y59,z59,x47,y47,z47,...
x48,y48,z48,x64,y64,z64,x63,y63,z63,limit);
else
    fillcub(x1,y1,z1,x4,y4,z4,x52,y52,z52,x49,y49,z49,x13,y13,z13,x16,...
y16,z16,x64,y64,z64,x61,y61,z61);
end
%-----
function fillcub(a1,b1,c1,a2,b2,c2,a3,b3,c3,a4,b4,c4,a5,b5,c5,a6,b6,c6,a7,...
b7,c7,a8,b8,c8)
verticesA=[a1,b1,c1;a2,b2,c2;a3,b3,c3;a4,b4,c4;a5,b5,c5;a6,b6,c6;...
a7,b7,c7;a8,b8,c8];
faces=[1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
patch('Vertices',verticesA,'Faces',faces,'FaceVertexCData',...

```

```
hsv(6), 'FaceColor', 'flat');  
hold on;  
%-----  
function good_axis  
axis equal  
view(3)  
set(gca, 'Visible', 'off')
```



DAFTAR PUSTAKA

1. Badii, R., and Politi, A. *Complexity: Hierarchical structures and scaling in physics*. Cambridge: Cambridge University Press, 2003.
2. Berry, J., and Houston, K. *Mathematical Modeling*. London: Edward Arnold, 1995.
3. Boccara, N. *Modeling Complex Systems*. New York: Springer-Verlag 2004.
4. Boone, M. <http://www.mathworks.com/matlabcentral/fileexchange/3524-sierpinski-sponge>, 2010
5. Cooper, J. *A MATLAB Companion for Multi Variable Calculus*. San Diego: Academic Press, 2001.
6. de Vries, G., Hillen, T., Lewis, M., Muller, J., and Schonfisch. *A course in mathematical biology: quantitative modeling with mathematical and computational methods*. Philadelphia: SIAM, 2006.
7. Downey, A. B. *Physical Modeling in MATLAB*. Free Software Foundation, 2007.
8. Gardner, M. 1972. The fantastic combinations of John Conway's new solitaire game 'Life'. *Scientific American* 233: 120–23.
9. Gebbers, Robin, and Marwan, Norbert. *Matlab® Recipes for Earth Sciences*. Springer-Verlag Berlin Heidelberg 2007.
10. Haliday, D., Resnick, R., and Walker, J. *Fundamental of Physics (Extended)*. John Wiley & Sons (Asia), 2008.
11. Holzbecher, E. *Environmental Modeling using MATLAB*. Berlin: Springer-Verlag 2007.
12. Ivancevic, V. G., and Ivancevic, T. *Understanding Complex Systems*. Berlin: Springer-Verlag 2008
13. Lam, Lui. (Editor). *Introduction to Nonlinear Physics*. New York: Springer-Verlag 2003.
14. Lam, Lui. *Nonlinear Physics for Beginners*. Singapore: World Scientific Publishing, 1998.
15. Levy, S. *Artificial Life*. New York: Vintage Books, 1992.
16. Liu, Y. *Modeling Urban development with geographical information systems and cellular automata*. Boca Raton (Florida): CRC Press, 2009.
17. Lynch, S. *Dynamical Systems with Applications using Mathematica*. Birkhauser, Boston, 2007.
18. Lundgren, J. <http://www.mathworks.com/matlabcentral/fileexchange/27577-fractal-curves>, 2010
19. Mandelbrot, B. *The Fractal Geometry of Nature*, Freeman, New York, 1982.
20. Schiff, J. L. *Cellular Automata*. New Jersey: John Wiley & Sons, 2008.
21. Steeb, Willy-Hans, Hardy, Y., and Stoop, R. *The Nonlinear Workbook (3rd ed.)*. Singapore: World Scientific Publishing, 2005.
22. Wolfram, S. 1983. Statistical mechanics of cellular automata. *Reviews of Modern Physics* 55:601–44.
23. Wolfram, S. 1984. Cellular automata as models of complexity. *Nature*. 311: 419–24.
24. Wolfram, S. 1994. *Cellular automata and complexity: collected papers*. Reading, MA: Addison-Wesley.
25. Wolfram, S. 2002. *A New Kind of Science*. Champaign, Illinois: Wolfram Media.
26. <http://www.exoleta.com/code/life>