2004 年度卒業研究レポート

BZ 反応

明治大学 理工学部 数学科
小 林 雄 太

2005 年 2 月 28 日

目 次

第1章　　序

この論文は、桂田先生の研究室の先輩である原野さんと内藤さんのレポート [7]、 [8] を参考に、より良い完成度を目指したものである。

私たちは、卒業研究の中で熱方程式や波動方程式を学んできた。そこで、この研究のまとめとして、熱方程式に似た方程式を持つＢＺ反応についての研究を試みようとしている。

まずはじめに、私たちの身の回りにおいて、生命現象に現れる「形」や「模様」に関心を寄せてみる。すると、シマウマやキリンの模様、心臓の鼓動などリズムとパターンが数多く存在していることに気づく。しかし、リズムやパターンは生物固有のものではない。ＢＺ反応は、リズムとパターン形成の代表的な反応である。

ＢＺ反応は、生命現象と驚くほどの相似性をもつリズムやパターンを示す。そして、今日では、最もよく理解されている非線形化学反応であり、その研究と展開は科学の広い分野に及んでいる。

この研究により、ＢＺ反応の数理モデルを数学的に解析し、理解を深めることを目標とする。

第2章　BZ反応

2.1　BZ反応とは

BZ反応は4種類の化合物を混合して観測される、周期的な酸化還元反応の一種である。酸化剤と還元剤を同一容器で混合すると、通常は瞬時に反応が完了してしまう。しかしBZ反応では、酸化還元反応がゆっくりと周期的に進行する。反応に必要な試薬は、

金属触媒/酸化剤/還元剤/酸

である。

BZ反応の名はBelousov-Zhabotinsky reactionの訳である。これは、この化学反応の発明者と改良者である2人のロシア人科学者の姓に由来している。

1950年代にBelousov(ベローソフ)は、

Ce^{4+}/BrO^{3-}/クエン酸/H_2SO_4

を用いた実験により、セリウムイオンを触媒として硫酸水溶液中におけるクエン酸の酸化還元反応において、クエン酸が臭素酸によって酸化される過程で溶液の色調が黄色と無色に振動的に変化することを発見した。その後、1960年代から70年頃にかけて、ジャボチンスキー(Zhabotinsky)はその反応を調べて、クエン酸をマロン酸におきかえても同様な反応が起こること、またセリウムイオンの代わりにフェロインを触媒として使うと色の変化が赤と青の振動として現れ、より観察しやすいことを見出した。

2.2　空間パターン、時間空間パターンの観測

BZ反応の発見者ベローソフは、空間パターンの存在については言及していない。Turing(チューリング)やI.Prigogine(プリコジン)の研究に刺激されてBZ反応で空間パターンを最初に報告したのは、ジャボチンスキーやH.G.Busse(ブッセ)である。同心円状のパターン(ターゲットパターン)の発見は、1970年になされている。

spiral waves または chemical rotors と呼ばれるらせん波も直後に発見されている。(ジャボチンスキーは1970、71年に、Winfree(ウィンフリー)は1971，72年に報告している。)

第3章　BZ反応のモデル

3.1　FKNメカニズム

標準的なBZ反応の化学組成は

$$Ce^{4+}/BrO_3^-/CH_2(COOH)_2/H_2SO_4$$

からなり、1972年に簡潔な反応機構がR.J.Field(フィールド)、E.Koros(ケレス)、R.M.Noyes(ノイエス)により提案された。

全反応過程

$$2BrO_3^- + 3CH_2(COOH)_2 + 2H^+ \longrightarrow 2BrCH(COOH)_2 + 3CO_2 + 4H_2O \tag{3.1}$$

臭素酸 BrO_3 によるマロン酸 $CH_2(COOH)_2$ の酸化反応である。この反応は３つの主過程から形成されている。

(I) BrO_3^- から Br_2 の生成

$$BrO_3^- + Br^- + 2H^+ \longrightarrow HBrO_2 + HOBr \tag{3.2}$$
$$HB\ rO_2 + Br^- + H^+ \longrightarrow 2HOBr \tag{3.3}$$
$$HOBr + Br^- + H^+ \longrightarrow Br_2 + H_2O \tag{3.4}$$

(II) 自己触媒過程を経由した Br_2 の生成

$$BrO_3^- + HBrO_2 + H^+ \longrightarrow 2BrO_2 + H_2O \tag{3.5}$$
$$BrO_2 + Ce^{3+} + H^+ \longrightarrow HBrO_2 + Ce^{4+} \tag{3.6}$$

この２つの反応をまとめると

$$BrO_3^- + HBrO_2 + 2Ce^{3+} \longrightarrow 2HBrO_2 + 2Ce^{4+} + H_2O \tag{3.7}$$

であり、$HBrO_2$ が自己触媒的に生成されている。

$$2HbrO_2 \longrightarrow BrO_3^- + HOBr + H^+ \tag{3.8}$$

$HBrO_2$ の反応によって HOBr に変化し (3.4) に使われて Br_2 を生成する。

(III) Br^- の生成

$$Br_2 + CH_2(COOH) \longrightarrow BrCH(COOH)_2 + Br^- + H^+ \quad (3.9)$$

$$BrCH(COOH)_2 + 4Ce^{4+} + 2H_2O \longrightarrow 4Ce^{3+} + Br^- + HCOOH + 2CO_2 + 5H^+ \quad (3.10)$$

$$CH_2(COOH)_2 + 6Ce^{4+} + 2H_2O \longrightarrow 6Ce^{3+} + HCOOH + 2CO_2 + 6H^+ \quad (3.11)$$

$$Br_2 + HCOOH \longrightarrow 2Br^- + CO_2 + 2H^+ \quad (3.12)$$

(I)(II)(III) のプロセスが繰り返されることにより、濃度の振動現象が見られることになる。

((I) で Br^- が少なくなる ⇒(I) の進行が停止 ⇒ 残っている $HBrO_2$ が過程 (II) で急激に増加 ⇒ 残っている Br^- を使って Br_2 がさらに生成 ⇒(III) によって Br_2 から Br^- への変換が起こる)

3.2 Oregonater(オレゴネータ)

フィールドとノイエスは、ＦＫＮメカニズムの反応過程から (3.2), (3.3), (3.7), (3.8), (3.10) を基本的なプロセスとして反応式を抜き出し、Oregonater(オレゴネータ) と呼ばれる数理モデルを提唱した。(ノイエスたちがオレゴン大学にいたことがその名前の由来である。)

各成分の濃度を

$$A=[BrO_3^-], \quad B=[BrCH(COOH)_2], \quad (3.13)$$

$$P=[HOBr], \quad X=[HBrO_2], \quad (3.14)$$

$$Y=[Br^-], \quad Z=[Ce^{4+}] \quad (3.15)$$

と定義する。すると、上のプロセスは

$$A + Y \longrightarrow X + P \quad (3.16)$$

$$X + Y \longrightarrow 2P \quad (3.17)$$

$$A + X \longrightarrow 2X + 2Z \quad (3.18)$$

$$2X \longrightarrow A + P \quad (3.19)$$

$$Z + P \longrightarrow hY \quad (3.20)$$

(ただし、h は Ce^{4+} １個を消費するのに伴ってどれだけの Y が生成されるかどうかのパラメーターである。)

と書ける。A と B は系に十分多量にあると仮定して、これらの時間変化は他の成分の時間変化に比べて無視する。５つの反応速度を上から各々 $k_1, k_2, \cdots, k_5$ とおく。

オレゴネータの反応中間体は、X, Y, Z の３種である。以後の反応速度は科学種 X の濃度 $[X]$ を X で表し、それぞれについて速度式を作ると、 X, Y, Z に対して次の方程式を得る。

$$\frac{dX}{ds} = k_1 H^2 AY - k_2 HXY + k_3 HAX - 2k_4 x^2, \quad (3.21)$$

$$\frac{dY}{ds} = -k_1 H^2 AY - k_2 HXY + hk_5 BZ, \quad (3.22)$$

$$\frac{dZ}{ds} = 2k_3 HAX - k_5 BZ. \quad (3.23)$$

(ただし、$k_1, k_2, \cdots, k_5, h, H$ は定数。)

3.3 オレゴネータの無次元化タイソン・バージョン

オレゴネータをコンピュータで処理するために、通常、無次元化という処理を行う。ここでは、タイソンによる無次元化を紹介する。フィールドとノイエスとの違いは次の２つである。１つ目は、$k_jB = 0.02(s^{-1})$ の値である。２つ目は速度定数の値である。タイソンによって実験的に不確定さが残るパラメーターの値に対し、q の値として、$(q = 8 \times 10^{-4}$ "Lo values") と $(q = 4 \times 10^{-4}$ "Hi values") が提案された。現在では、前者が正しいとされる。

オレゴネータのタイソンバージョンでは、無次元化パラメータ

$$X = \frac{X}{X_0}, \qquad X_0 = \frac{k_5HA}{2k_4}, \qquad y = \frac{Y}{Y_0}, \qquad Y_0 = \frac{k_5A}{k_2},$$

$$Z = \frac{Z}{Z_0}, \qquad Z_0 = \frac{(k_5HA)^2}{k_4k_jB}, \qquad t = \frac{s}{s_0}, \qquad s_0 = \frac{1}{k_jB},$$

$$\epsilon = \frac{k_jB}{k_5HA}, \qquad \epsilon' = \frac{2k_4k_jB}{k_2k_5H^2A}, \qquad f = 2h, \qquad q = 2k_3(\frac{k_5}{k_2})(\frac{k_3}{k_5^2})$$

として、オレゴネータを書き下す。

＜３変数版＞

$$\epsilon\frac{dx}{dt} = qy - xy - x(1-x), \tag{3.24}$$

$$\epsilon'\frac{dy}{dt} = -qy - xy + fz, \tag{3.25}$$

$$\frac{dz}{dt} = x - z. \tag{3.26}$$

ただし $\epsilon, \epsilon', q, f$ は正定数とする。

ϵ, ϵ' の値は、反応過程に現れる臭素酸およびマロン酸の濃度が 0.06M, 0.02M の場合では

$$\epsilon' = 2.5 \times 10^{-5}, \epsilon = 10^{-2}$$

ここで $\epsilon' \ll \epsilon$ と考えると次の２変数版を得る。

＜２変数版＞

$$\epsilon\frac{dx}{dt} = x(1-x) - fz\frac{x-q}{x+q}, \tag{3.27}$$

$$\frac{dz}{dt} = x - z. \tag{3.28}$$

3.4 オレゴネーターのキーナー・タイソンバージョン

オレゴネーターを基本に化学反応波を理解するために空間パターンを考える必要がある。（今までは、時間だけに依っていた。）

そこで、ＢＺ反応に伴うパターンダイナミックスを説明するモデルとして次式のキーナー・タイソンバージョンのオレゴネーターがよく知られている。

＜キーナー・タイソンバージョン＞

$u = u(x,t), v = v(x,t), \Delta =$Laplacian

$$u_t = D_u\Delta + \frac{1}{\epsilon}\left(u(1-u) - fv\frac{u-q}{u+q}\right), \tag{3.29}$$

$$v_t = D_v\Delta + u - v. \tag{3.30}$$

u は反応拡散因子（$HBrO_2$）の濃度、v は抑制因子（Fe^{3+}）の濃度、D_u, D_v はそれぞれ u, v の拡散係数、ϵ, f, q は定数とする。

第4章　　オレゴネーターによる数値解析

4.1　キーナー・タイソンバージョン（Neumann 境界条件）

Ω を R^2 の開区間 $(a,b)\times(c,d)$ とする。このとき、Ω におけるキーナー・タイソンバージョンの初期値境界値問題

$$\begin{aligned} u_t &= D_u\Delta u+\frac{1}{\epsilon}\left(u(1-u)-fv\frac{u-q}{u+q}\right) \quad (t>0,(x,y)\in\Omega), && (4.1)\\ v_t &= D_v+u-v \quad (t>0,(x,y)\in\Omega), && (4.2)\\ \frac{\partial u}{\partial n}(x,y,t) &= \frac{\partial v}{\partial n}(x,y,z)=0 \quad (t>0,(x,y)\in\partial\Omega), && (4.3)\\ u(x,y,0) &= u_0(x,y), \quad v(x,y,z)=v_0(x,y) \quad ((x,y)\in\bar{\Omega}). && (4.4)\end{aligned}$$

を考える。ここで、u,v は未知関数とし、D_u,D_v,ϵ,f,q は既知定数、u_0,v_0 は既知関数とする。また、n は境界における単位法線ベクトルを表し、Δ は R^2 における Laplace 作用素 (Laplacian) である:

$$\Delta\stackrel{\mathrm{def}}{=}\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}.$$

領域 Ω を座標軸に平行な格子線で x 軸、y 軸方向にそれぞれ N_x 等分、N_y 等分し、格子点を (x_i,y_j) と表す。すなわち、

$$h_x\stackrel{\mathrm{def}}{=}\frac{1}{N_x},\quad h_y\stackrel{\mathrm{def}}{=}\frac{1}{N_y}$$

として、

$$\begin{aligned} x_i&\stackrel{\mathrm{def}}{=}ih_x \quad (0\le i\le N_x),\\ y_j&\stackrel{\mathrm{def}}{=}jh_y \quad (0\le j\le N_y)\end{aligned}$$

とおく。

次に時間変数 t に関する刻み幅 $\tau(>0)$ を固定し、

$$t_n\stackrel{\mathrm{def}}{=}n\tau \qquad (n=0,1,\cdots)$$

とおく。

以下、我々は格子点 (x_i,y_j) における u,v の値、すなわち

$$\begin{aligned} u^n_{i,j}&\stackrel{\mathrm{def}}{=}u(x_i,y_j,t_n) \quad (0\le i\le N_x;0\le j\le N_y;n=0,1,\cdots),\\ v^n_{i,j}&\stackrel{\mathrm{def}}{=}v(x_i,y_j,t_n) \quad (0\le i\le N_x;0\le j\le N_y;n=0,1,\cdots)\end{aligned}$$

の近似値

$$U_{i,j}^n \quad (0 \le i \le N_x; 0 \le j \le N_y; n = 0, 1, \cdots),$$
$$V_{i,j}^n \quad (0 \le i \le N_x; 0 \le j \le N_y; n = 0, 1, \cdots)$$

を求めることを目標にする。

さて、時間、空間に関する偏微分係数をどのように近似するかを示す。以下では、ADI 法 (alternating direction implicit method, 交互方向陰解法) を用いた差分近似を紹介する。

まず、ADI 法により近似して作った差分方程式は

$$\begin{aligned}
\frac{U_{i,j}^{n+\frac{1}{2}} - U_{i,j}^n}{\tau/2} &= D_u \left(\frac{U_{i+1,j}^{n+\frac{1}{2}} - 2U_{i,j}^{n+\frac{1}{2}} + U_{i-1,j}^{n+\frac{1}{2}}}{h_x^2} + \frac{U_{i,j+1}^n - 2U_{i,j}^n + U_{i,j-1}^n}{h_y^2} \right) \\
&+ \frac{1}{\epsilon} \left(U_{i,j}^n (1 - U_{i,j}^n) - f V_{i,j}^n \frac{U_{i,j}^n - q}{U_{i,j}^n + q} \right), \\
\frac{V_{i,j}^{n+\frac{1}{2}} - V_{i,j}^n}{\tau/2} &= D_v \left(\frac{V_{i+1,j}^{n+\frac{1}{2}} - 2V_{i,j}^{n+\frac{1}{2}} + V_{i-1,j}^{n+\frac{1}{2}}}{h_x^2} + \frac{V_{i,j+1}^n - 2V_{i,j}^n + V_{i,j-1}^n}{h_y^2} \right) \\
&+ U_{i,j}^n - V_{i,j}^n \quad , \\
\frac{U_{i,j}^{n+1} - U_{i,j}^{n+\frac{1}{2}}}{\tau/2} &= D_u \left(\frac{U_{i+1,j} n + \frac{1}{2} - 2U_{i,j}^{n+\frac{1}{2}} + U_{i,j} n + \frac{1}{2}}{h_x^2} + \frac{U_{i,j+1}^{n+1} - 2U_{i,j}^{n+1} + U_{i,j-1}^{n+1}}{h_y^2} \right) \\
&+ \frac{1}{\epsilon} \left(U_{i,j}^{n+\frac{1}{2}} (1 - U_{i,j}^{n+\frac{1}{2}}) - f V_{i,j}^{n+\frac{1}{2}} \frac{U_{i,j} n + \frac{1}{2} - q}{U_{i,j}^{n+\frac{1}{2}} + q} \right), \\
\frac{V_{i,j}^{n+1} - V_{i,j}^{n+\frac{1}{2}}}{\tau/2} &= D_v \left(\frac{V_{i+1,j}^{n+\frac{1}{2}} - 2V_{i,j}^{n+\frac{1}{2}} + V_{i-1,j}^{n+\frac{1}{2}}}{h_x^2} + \frac{V_{i,j+1}^{n+1} - 2V_{i,j}^{n+1} + V_{i,j-1}^{n+1}}{h_y^2} \right) \\
&+ U_{i,j}^{n+\frac{1}{2}} - V_{i,j} n + \frac{1}{2}
\end{aligned}$$

である。　ここで、

$$\lambda_{x,u} \stackrel{\text{def}}{=} \frac{D_u \tau}{2h_x^2}, \quad \lambda_{y,u} \stackrel{\text{def}}{=} \frac{D_u \tau}{2h_y^2}, \quad \lambda_{x,v} \stackrel{\text{def}}{=} \frac{D_v \tau}{2h_x^2}, \quad \lambda_{y,v} \stackrel{\text{def}}{=} \frac{D_v \tau}{2h_y^2},$$

$$R_u(u, v) \stackrel{\text{def}}{=} u(1 - u) - f v \frac{u - q}{u + q}, \quad R_v(u, v) \stackrel{\text{def}}{=} u - v$$

として、まず n から $n + \frac{1}{2}$ 段へは、

$$\begin{aligned}
&(1 + 2\lambda_{x,u}) U_{i,j}^{n+\frac{1}{2}} - \lambda_{x,u} (U_{i+1,j}^{n+\frac{1}{2}} + U_{i-1,j} n + \frac{1}{2}) \\
&\quad = (1 - 2\lambda_{y,u}) U_{i,j}^n + \lambda_{y,u} (U_{i,j+1}^n + U_{i,j-1}^n) + \frac{\tau}{2} R_u (U_{i,j}^n, V_{i,j}^n) \quad \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix}, \quad (4.5)
\end{aligned}$$

$$(1+2\lambda_{x,v})V_{i,j}^{n+\frac{1}{2}} - \lambda_{x,v}(V_{i+1,j}^{n+\frac{1}{2}} + V_{i-1,j}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,v})V_{i,j}^{n} + \lambda_{y,v}(V_{i,j+1}^{n} + V_{i,j-1}^{n}) + \frac{\tau}{2}R_v(U_{i,j}^{n} + V_{i,j}^{n}) \quad \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix}. \quad (4.6)$$

次に $n+\frac{1}{2}$ から $n+1$ 段へは、

$$(1+2\lambda_{y,u})U_{i,j}^{n+1} - \lambda_{y,u}(U_{i,j+1}^{n+1} + U_{i,j-1}^{n+1})$$
$$= (1-2\lambda_{x,u})U_{i,j}^{n+\frac{1}{2}} + \lambda_{x,u}(U_{i+1,j}^{n+\frac{1}{2}} + U_{i-1,j}^{n+\frac{1}{2}}) + \frac{\tau}{2}R_u(U_{i,j}^{n+\frac{1}{2}}, V_{i,j}^{n+\frac{1}{2}}) \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix}, \quad (4.7)$$

$$(1+2\lambda_{y,v})V_{i,j}^{n+1} - \lambda_{y,v}(V_{i,j+1}^{n+1} + V_{i,j-1}^{n+1})$$
$$= (1-2\lambda_{x,v})V_{i,j}^{n+\frac{1}{2}} + \lambda_{x,v}(U_{i+1,j}^{n+\frac{1}{2}} + V_{i-1,j}^{n+\frac{1}{2}}) + \frac{\tau}{2}R_v(U_{i,j}^{n+\frac{1}{2}}, V_{i,j}^{n+\frac{1}{2}}) \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix} \quad (4.8)$$

となる。

さて、(4.5) に $i=0$ を代入すると $U_{-1,j}^{n+\frac{1}{2}}$ が出てくるが、これには、(4.3) の $\frac{\partial u}{\partial n}(x,y,t)=0$ を 1 階中心差分近似して、

$$-\frac{U_{1,j}^{n+\frac{1}{2}} - U_{-1,j}^{n+\frac{1}{2}}}{2h_x} = 0 \quad (4.9)$$

によって出てくる、$U_{-1,j}^{n+\frac{1}{2}}$ で消去すればよい。他にも同様に境界には 1 階中心差分近似を用いることによって整理できる。ただし 1 階中心差分近似する際に法線ベクトル n の方向に注意する必要がある。

また、初期条件 (4.4) に対応して、

$$U_{i,j}^{0} = u_0(x_i, y_j), \quad V_{i,j}^{0} = v_0(x_i, y_j) \quad \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix} \quad (4.10)$$

となる。

以上をまとめると、3 重添字を持つ未知数列 $\{U_{i,j}^{n}; 0 \le i \le N_x; 0 \le j \le N_y; n = 0, 1, \cdots\}$ を定めるための次のような差分方程式を得る。

まず、n から $n+\frac{1}{2}$ 段へは、

1. 長方形の内部

$$(1+2\lambda_{x,u})U_{i,j}^{n+\frac{1}{2}} - \lambda_{x,u}(U_{i+1,j}\ n + \frac{1}{2} + U_{i-1,j}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,u})U_{i,j}^{n} + \lambda_{y,u}(U_{i,j+1}n + U_{i,j-1}^{n}) + \frac{\tau}{2}R_u(U_{i,j}^{n}, V_{i,j}^{n}) \quad \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix}, \quad (4.11)$$

$$(1+2\lambda_{x,v})V_{i,j}^{n+\frac{1}{2}} - \lambda_{x,v}(V_{i+1,j}^{n+\frac{1}{2}} + V_{i-1,j}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,v})V_{i,j}^{n} + \lambda_{y,v}(V_{i,j+1}^{n} + V_{i,j-1}^{n}) + \frac{\tau}{2}R_v(U_{i,j}^{n}, V_{i,j}^{n}) \quad \begin{pmatrix} 1 \le i \le N_x - 1 \\ 1 \le j \le N_y - 1 \end{pmatrix}. \quad (4.12)$$

2. 長方形の左辺

$$(1+2\lambda_{x,u})U_{0,j}^{n+\frac{1}{2}} - 2\lambda_{x,u}U_{1,j}^{n+\frac{1}{2}}$$
$$= (1-2\lambda_{x,v})U_{0,j}^{n} + \lambda_{y,u}(U_{0,j+1}^{n} + U_{0,j-1}^{n}) + \frac{\tau}{2}R_u(U_{0,j}^{n}, V_{0,j}^{n}) \quad (1 \le j \le N_y - 1), \quad (4.13)$$

$$(1+2\lambda_{x,v})V_{0,j}^{n+\frac{1}{2}} - 2\lambda_{x,v}V_{1,j}^{n+\frac{1}{2}}$$
$$= (1-2\lambda_{y,v})V_{0,j}^{n} + \lambda_{y,v}(V_{0,j+1}^{n} + V_{0,j-1}^{n}) + \frac{\tau}{2}R_v(U_{i,j}^{n}, V_{i,j}^{n}) \quad (1 \le j \le N_y - 1). \quad (4.14)$$

3. 長方形の右辺

$$(1+2\lambda_{x,u})U_{N_x,j}^{n+\frac{1}{2}} - 2\lambda_{x,u}U_{N_x-1,j}^{n+\frac{1}{2}}$$
$$= (1-2\lambda_{y,u})U_{N_x,j}^{n} + \lambda_{y,u}(U_{N_x,j+1}^{n} + U_{N_x,j-1}^{n}) + \frac{\tau}{2}R_u(U_{N_x,j}^{n}, V_{N_x,j}) \quad (1 \le j \le N_y - 1), \quad (4.15)$$

$$(1+2\lambda_{x,v})V_{N_x,j}^{n+\frac{1}{2}} - 2\lambda_{x,v}V_{N_x-1,j}^{n+\frac{1}{2}}$$
$$= (1-2\lambda_{y,v})V_{N_x,j}^{n} + \lambda_{y,v}(V_{N_x,j+1}^{n} + V_{N_x,j-1}^{n}) + \frac{\tau}{2}R_v(U_{N_x,j}^{n}, V_{N_X,j}^{n}) \quad (1 \le j \le N_y - 1). \quad (4.16)$$

4. 長方形の下辺

$$(1+2\lambda_{x,u})U_{i,0}^{n+\frac{1}{2}} - \lambda_{x,u}(U_{i+1,0}^{n+\frac{1}{2}} + U_{i-1,0}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,u})U_{i,0}^{n} + 2\lambda_{y,u}U_{i,1}^{n} + \frac{\tau}{2}R_u(U_{i,0}^{n}, V_{i,0}^{n}) \quad (1 \le i \le N_x - 1), \quad (4.17)$$

$$(1+2\lambda_{x,v})V_{i,0}^{n+\frac{1}{2}} - \lambda_{x,v}(V_{i+1,0}^{n+\frac{1}{2}} + V_{i-1,0}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,v})V_{i,0}^{n} + 2\lambda_{y,v}V_{i,1}^{n} + \frac{\tau}{2}R_v(U_{i,0}^{n}, V_{i,0}^{n}) \quad (1 \le i \le N_x - 1). \quad (4.18)$$

5. 長方形の上辺

$$(1+2\lambda_{x,u})U_{i,N_y}^{n+\frac{1}{2}} - \lambda_{x,u}(U_{i+1,N_y}^{n+\frac{1}{2}} + U_{i-1,N_y}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,u})U_{i,N_y}^{n} + 2\lambda_{y,u}U_{i,N_y-1}^{n} + \frac{\tau}{2}R_u(U_{i,N_y}^{n}, V_{i,N_y}^{n}) \quad (1 \le i \le N_x - 1), \quad (4.19)$$

$$(1+2\lambda_{x,v})V_{i,N_y}^{n+\frac{1}{2}} - \lambda_{x,v}(V_{i+1,N_y}^{n+\frac{1}{2}} + V_{i-1,N_y}^{n+\frac{1}{2}})$$
$$= (1-2\lambda_{y,v})V_{i,N_y}^{n} + 2\lambda_{y,v}V_{i,N_y-1}^{n} + \frac{\tau}{2}R_v(U_{i,N_y}^{n}, V_{i,N_y}^{n}) \quad (1 \le i \le N_x - 1). \quad (4.20)$$

6. 長方形の４隅

$$(1+2\lambda_{x,u})U_{0,0}^{n+\frac{1}{2}} - 2\lambda_{x,u}(U_{1,0}^{n+\frac{1}{2}}) = (1-2\lambda_{y,u})U_{0,0}^{n} + 2\lambda_{y,u}U_{0,1}^{n} + \frac{\tau}{2}R_u(U_{0,0}^{n}, V_{0,0}^{n}), \tag{4.21}$$

$$(1+2\lambda_{x,v})V_{0,0}^{n+\frac{1}{2}} - 2\lambda_{x,v}V_{1,0}^{n+\frac{1}{2}} = (1-2\lambda_{y,v})V_{0,0}^{n} + 2\lambda_{y,v}V_{0,1}^{n} + \frac{\tau}{2}R_u(U_{0,0}^{n}, V_{0,0}^{n}), \tag{4.22}$$

$$(1+2\lambda_{x,u})U_{N_x,0}^{n+\frac{1}{2}} - 2\lambda_{x,u}U_{N_x-1,0}^{n+\frac{1}{2}} = (1-2\lambda_{y,u})U_{N_x,0}^{n} + 2\lambda_{y,u}U_{N_x,1}^{n} + \frac{\tau}{2}R_u(U_{N_x,0}^{n}, V_{N_x,0}^{n}), \tag{4.23}$$

$$(1+2\lambda_{x,v})V_{N_x,0}^{n+\frac{1}{2}} - 2\lambda_{x,v}(V_{N_x-1,0}^{n+\frac{1}{2}}) = (1-2\lambda_{y,v})V_{N_x,0}^{n} + 2\lambda_{y,v}V_{N_x,1}^{n} + \frac{\tau}{2}R_v(U_{N_x,0}^{n}, V_{N_x,0}^{n}), \tag{4.24}$$

$$(1+2\lambda_{x,u})U_{0,N_y}^{n+\frac{1}{2}} - 2\lambda_{x,u}(U_{1,N_y}^{n+\frac{1}{2}}) = (1-2\lambda_{y,u})U_{0,N_y}^{n} + 2\lambda_{y,u}U_{0,N_y-1}^{n} + \frac{\tau}{2}R_u(U_{0,N_y}^{n}, V_{0,N_y}^{n}), \tag{4.25}$$

$$(1+2\lambda_{x,v})V_{0,N_y}^{n+\frac{1}{2}} - 2\lambda_{x,v}(V_{1,N_y}^{n+\frac{1}{2}}) = (1-2\lambda_{y,v})V_{0,N_y}^{n} + 2\lambda_{y,v}V_{0,N_y-1}^{n} + \frac{\tau}{2}R_v(U_{0,N_y}^{n}, V_{0,N_y}^{n}), \tag{4.26}$$

$$\begin{aligned}(1+2\lambda_{x,u})U_{N_x,N_y}^{n+\frac{1}{2}} &- 2\lambda_{x,u}(U_{N_x-1,N_y}^{n+\frac{1}{2}}) \\ &= (1-2\lambda_{y,u})U_{N_x,N_y}^{n} + 2\lambda_{y,u}U_{N_x,N_y-1}^{n} + \frac{\tau}{2}R_u(U_{N_x,N_y}^{n}, V_{N_x,N_y}^{n}),\end{aligned} \tag{4.27}$$

$$\begin{aligned}(1+2\lambda_{x,v})V_{N_x,N_y}^{n+\frac{1}{2}} &- 2\lambda_{x,v}(V_{N_x-1,N_y}^{n+\frac{1}{2}}) \\ &= (1-2\lambda_{y,v})V_{N_x,N_y}^{n} + 2\lambda_{y,v}V_{N_x,N_y-1}^{n} + \frac{\tau}{2}R_v(U_{N_x,N_y}^{n}, V_{N_x,N_y}^{n}).\end{aligned} \tag{4.28}$$

第5章　　非線形自励系

5.1　　自励系

定義　　連立微分方程式

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x})$$

において、右辺が t を含まないとき、すなわち

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}) \qquad (*)$$

の形であるとき、**自励系 (autonomous system)** という。

ここからは、初期値問題の解の一意性が成り立ち、かつ、$-\infty < t < \infty$ で解が存在すると仮定する。

定理　　自励系では解軌道のグラフは初期時刻には依存しない。すなわち、$\boldsymbol{x}(t_1) = \boldsymbol{x}_0$ となる解を $\boldsymbol{x}_1(t), \boldsymbol{x}_2(t) = \boldsymbol{x}_0$ となる解を $\boldsymbol{x}_2(t)$ とし、それぞれの時刻 t_1, t_2 から出発して、過去へも未来へも、解が存在する限りその解軌道を延長したとき、二つの解軌道は $\boldsymbol{R}^n$ 上の集合としては同じものになる。

(証明)　$\boldsymbol{x}_1(t)$ も $\boldsymbol{x}_2(t - t_1 + t_2)$ もともに $(*)$ の解で、$\boldsymbol{x}_1(t) = \boldsymbol{x}_2(t - t_1 + t_2) = \boldsymbol{x}_0$ であるから、解の一意性により、$\boldsymbol{x}_1(t) = \boldsymbol{x}_2(t - t_1 + t_2)$。したがって、$\boldsymbol{x}_2(t)$ の解軌道は $\boldsymbol{x}_1(t)$ の解軌道をいつかは必ず通るし、その逆も成り立つ。よって両者は一致する。

系　相空間 $\boldsymbol{R}^n$ において、任意の１点 $\boldsymbol{x}_0$ を通るとき、$\boldsymbol{x}_0$ を通る解軌道はただ１つしかない。

5.2　　平衡点

定義　$(*)$ に対し、$\boldsymbol{x}_t \equiv \boldsymbol{x}_0$(定数ベクトル) の形の解が存在するとき、相空間での点 $\boldsymbol{x}_0$ を微分方程式 $(*)$ の平衡点（または危点）という。

この定義からただちに次の定義を得る。

定理 $\boldsymbol{R}^n$ の点 $\boldsymbol{x}_0$ が $(*)$ の平衡点であるための必要十分条件は

$$\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{0}$$

(証明)

$\Rightarrow$)　$\boldsymbol{x}(t) = \boldsymbol{x}_0$ が平衡点であるから

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{0}$$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}_0) \quad \text{だから} \quad \boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{0}$$

$\Rightarrow$)　$\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{0}$ をみたす点 $\boldsymbol{x}_0$ をとる。

定数ベクトル関数 $\boldsymbol{x}(t) = \boldsymbol{x}_0$ を考えると明かに $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$ をみたす。

＜平衡点の近傍での解の振る舞い＞

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ とおくと、A の固有値は次の４通り。

(i) 異なる２つの実数　　(ii) 重解で対角化可能

(iii) 重解で対角化不可能　(iv) 異なる２つの複素数

固有値を λ_1, λ_2 とおく。$(\lambda_1 < \lambda_2)$

(i) $\lambda_1, \lambda_2 > 0$ のとき、不安定結節点 ・ 湧点

$\lambda_1, \lambda_2 < 0$ のとき、安定結節点 ・ 沈点

$\lambda_1 < 0 < \lambda_2$ のとき、鞍点

(ii) $\lambda_1 > 0$ のとき、不安定結節点

$\lambda_1 < 0$ のとき、安定結節点

(iii) (ii) と同様

(iv) $\lambda_1 = \mu + i\nu, \lambda_2 = \mu - i\nu$ とすると、

$\mu > 0$ のとき、安定渦状点

$\mu < 0$ のとき、不安定渦状点

$\mu = 0$ のとき、渦心点

定義

(i) 平衡点 $\boldsymbol{x}_0$ が**安定平衡点**であるとは、

$\forall \epsilon > 0$ に対し、$\exists \delta > 0$ があって、

$$\| \boldsymbol{x}_1 - \boldsymbol{x}_0 \| < \delta$$

をみたす任意の初期値 $\boldsymbol{x}(0) = \boldsymbol{x}_1$ に対する解 $\boldsymbol{x}(t)$ は $0 \leq t < +\infty$ においてつねに

$$\| \boldsymbol{x}(t) - \boldsymbol{x}_0 \| < \epsilon$$

となることである。

(ii) 平衡点 $\boldsymbol{x}_0$ が**漸近安定な平衡点**であるとは、

$\boldsymbol{x}_0$ が安定平衡点かつ $\exists\delta_1 > 0$ があって、$\| \boldsymbol{x}(0) - \boldsymbol{x}_0 \| < \delta_1$をみたす解 $\boldsymbol{x}(t)$ は

つねに、

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}_0$$

となることである。

(iii) 安定でない平衡点を**不安定平衡点**という。

＜安定性＞　安定性を調べるためには、線形近似の方法を用いる。

5.3　線形近似

＊2 次元

$$\begin{cases} x' = f(x,y) \\ y' = g(x,y) \end{cases}$$

f, g はC ^{1} 級とし、(x_0, y_0) を平衡点とする。

すなわち、

$$f(x_0, y_0) = g(x_0, y_0) = 0.$$

点 (x_0, y_0) のまわりにテイラー展開すると、

$$f(x,y) = f(x_0,y_0) + f_x(x_0,y_0)(x-x_0) + f_y(x_0,y_0)(y-y_0) + o(\sqrt{|x-x_0|^2+|y-y_0|^2}) \quad (x,y)\to(x_0,y_0),$$

$$g(x,y) = g(x_0,y_0) + g_x(x_0,y_0)(x-x_0) + g_y(x_0,y_0)(y-y_0) + o(\sqrt{|x-x_0|^2+|y-y_0|^2}) \quad (x,y)\to(x_0,y_0).$$

したがって、

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f_x(x_0,y_0) & f_y(x_0,y_0) \\ g_x(x_0,y_0) & g_y(x_0,y_0) \end{bmatrix} \begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix} + \begin{bmatrix} o(\sqrt{|x-x_0|^2+|y-y_0|^2}) \\ o(\sqrt{|x-x_0|^2+|y-y_0|^2}) \end{bmatrix}$$

$A = \begin{bmatrix} f_x(x_0,y_0) & f_y(x_0,y_0) \\ g_x(x_0,y_0) & g_y(x_0,y_0) \end{bmatrix}$ とする。A を平衡点 (x_0, y_0) での**線形化行列**という。

$\begin{cases} u = x - x_0 \\ v = y - y_0 \end{cases}$ とおくと、

$$\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} o(\sqrt{|u|^2+|v|^2}) \\ o(\sqrt{|u|^2+|v|^2}) \end{bmatrix} \qquad ((u,v)\to(0,0))$$

＊ n 次元

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \qquad \boldsymbol{x} = (x_1, x_2, \cdots, x_n)$$

$\boldsymbol{f}(\boldsymbol{x})$ を点 $\boldsymbol{x}_0$ のまわりでテイラー展開すると

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= \boldsymbol{f}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{g}(\boldsymbol{x}), \\ \boldsymbol{g}(\boldsymbol{x}) &= o(|\boldsymbol{x} - \boldsymbol{x}_0|) \qquad ((\boldsymbol{x} \to \boldsymbol{x}_0)). \end{aligned}$$

ただし、$\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ は、$\boldsymbol{f}(\boldsymbol{x})$ の各項を $x_1, x_2, \cdots, x_n$ で微分したヤコビ行列

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}), & \cdots, & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{x}), & \cdots, & \frac{\partial f_n}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}$$

で、$A = \dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\boldsymbol{x})$ とおく。

$\boldsymbol{x}_0$ が $\boldsymbol{f}$ の平衡点 $\iff$ $\boldsymbol{f}(\boldsymbol{x}_0) = \boldsymbol{0}$ より、

$$\boldsymbol{f}(\boldsymbol{x}) = A(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{g}(\boldsymbol{x}).$$

ここで座標の平行移動 $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{x}_0$ を行うと、

$$\begin{cases} \frac{\partial y}{\partial t} = A \cdot \boldsymbol{y} + \boldsymbol{h}(\boldsymbol{y}) \\ \boldsymbol{h}(\boldsymbol{y}) = o(|\boldsymbol{y}|) \quad (\boldsymbol{y} \to \boldsymbol{0}) \end{cases}$$

である。

定理　線形化行列 A が安定な行列であれば、平衡点は漸近安定である。

定理　線形化行列 A の固有値の中に実部が正のものがあれば。平衡点は不安定である。

ともに証明は省略する。詳しくは、笠原 [4] を見よ。

第6章　　オレゴネータの数値解析

6.1　　タイソンバージョン(２変数版)

$$\frac{dx}{dt} = \frac{1}{\epsilon}\left(x(1-x) - fz\frac{x-q}{x+q}\right) \equiv F(x,z), \tag{6.1}$$

$$\frac{dz}{dt} = x - z \equiv H(x,z). \tag{6.2}$$

平衡点をもとめる。

$$\frac{1}{\epsilon}\left(x(1-x) - fz\frac{x-q}{x+q}\right) = 0, \tag{6.3}$$

$$x - z = 0 \tag{6.4}$$

を整理すると、

$$fz = \frac{x(1-x)(x+q)}{x-q}, \tag{6.5}$$

$$z = x \tag{6.6}$$

となる。

ここから z を消去して、

$$(x+q)x(1-q) - fx(x-q) = 0$$

これは３次方程式だが、0が根であることがわかるから、他の２根は２次方程式

$$(x+q)(1-q) - f(x-q) = 0$$

を解いて求められる。整理すると、

$$x^2 + (f+q-1)x - q(1+f) = 0.$$

$f > 0$ であるから、根は正と負１つずつ存在する。今、$x > 0$ と考えているため、正の根のみ考える。正の根は

$$x = \frac{1-f-q+\sqrt{(f+q-1)^2+4q(1+f)}}{2}.$$

よって、(6.1), (6.2) の平衡点は (x,x) である。

線形安定性解析により、平衡点を用いて安定性の判定を行う。(6.1), (6.2) から線形化行列をもとめる。

$$\frac{\partial F}{\partial x} = \frac{1}{\epsilon}\frac{-2x^3+(1-4q)x^2+2q(1-q)x+q(q-2fz)}{(x+q)^2},$$
$$\frac{\partial F}{\partial z} = \frac{1}{\epsilon}\frac{f(x-q)}{x+q},$$
$$\frac{\partial H}{\partial x} = 1, \qquad \frac{\partial H}{\partial x} = -1$$

から、線形化行列 A を

$$A = \begin{bmatrix} \frac{1}{\epsilon}\frac{-2x^3+(1-4q)x^2+2q(1-q)x+q(q-2fz)}{(x+q)^2} & \frac{1}{\epsilon}\frac{f(x-q)}{x+q} \\ 1 & -1 \end{bmatrix} \stackrel{\text{def.}}{=} \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

とすると、固有方程式は

$$\lambda^2 - (a+d)\lambda + (ad-bc) = 0.$$

固有値を λ_1, λ_2 とすると、

平衡点 (x, x) での安定性について

・λ_1, λ_2 の実部の少なくとも一方が正 $\Rightarrow$ 不安定

・λ_1, λ_2 がともに負 $\Rightarrow$ 安定

渦の有無について

・λ_1, λ_2 が虚数 $\Rightarrow$ 渦あり

・λ_1, λ_2 が実数 $\Rightarrow$ 渦なし

6.2 内藤 智さんの『ＢＺ反応の研究』

線形安定性解析を研究した内藤さんの卒業研究レポートより、線形安定性解析のプログラムと実験結果・考察を引用

6.2.1 線形安定性解析のプログラム

```
//<APPLET code="Anteisei.class"width=700 height=500></APPLET>

import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Anteisei extends Applet implements ActionListener {

    //スクリーン
    private Image img;
    private Graphics gra;
```

```
//ユーザーとのインターフェイス
private double eps,q;
private Label la1,la2;
private TextField tf1,tf2;
private Button bt1,bt2;
private TextArea ta;

//安定性解析の関数のためのパラメーター
private double f;
private double u,v;
private double a,b,c,d;
private double tr_A,det_A,D;//線形化して出来た行列

//二分法のためのパラメーター
final private double EPS =1e-15;//打ち切り誤差
final private int limit=50;//打ち切り回数
private double low,mid,high;
private int k=1;
private double lastf,sol;

//座標系の変換パラメーター
private double ratiox,ratioy,X0,Y0;
private int WX = 500, WY =500;

//描画範囲
private double x_max =   10;
private double x_min =   -1;
private double x_margin = (x_max-x_min)/10;
private double y_max =  120;
private double y_min =  -20;
private double y_margin = (y_max-y_min)/10;

private boolean IsIn(double x, double y) {
    return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
}

// 座標変換の準備
private void space(double x0, double y0, double x1, double y1) {
    X0 = x0; Y0 = y0;
    ratiox = WX / (x1 - x0);
    ratioy = WY / (y1 - y0);
}

// ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
private int wx(double x) {
    return (int)(ratiox * (x - X0));
}
// ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
private int wy(double y) {
    return WY - (int)(ratioy * (y - Y0));
}

//tf からの数値読み取り
private void ReadFields(){
    eps = Double.valueOf(tf1.getText()).doubleValue();
    q   = Double.valueOf(tf2.getText()).doubleValue();
}
```

```
//２乗の計算
private double sqr(double x){
    return(x * x);
}

//判別式の関数 g1
private double g1(double f){

    u      = (1-f-q+Math.sqrt(sqr(f+q-1)+4*q*(1+f)))/2;
    v      = u;
    a      = (-2*u*u*u + (1 - 4 * q)*u*u + 2*q*(1-q)*u
              + q*(q-2*f*v))/(eps*sqr(q+u));
    b      = f*(q-u)/(eps*(q+u));
    c      =  1;
    d      = -1;

    tr_A   = a + d;
    det_A  = a*d - b*c;
    D      = sqr(tr_A) - 4*det_A;

    return D;
}

//固有方程式の根の関数 g2
private double g2(double f){

    u      = (1-f-q+Math.sqrt(sqr(f+q-1)+4*q*(1+f)))/2;
    v      = u;
    a      = (-2*u*u*u + (1 - 4 * q)*u*u + 2*q*(1-q)*u
              + q*(q-2*f*v))/(eps*sqr(q+u));
    b      = f*(q-u)/(eps*(q+u));
    c      =  1;
    d      = -1;

    tr_A   = a + d;
    det_A  = a*d - b*c;
    D      = sqr(tr_A) - 4*det_A;

    if(D>0)
        return (tr_A+Math.sqrt(D))/2;

    else
        return (tr_A)/2;
}

public void init() {

    //tf la bt のレイアウト
    setLayout(null);

    add(la1 = new Label("εの値:"));
    la1.setBounds (510,20,90,30);
    add(la2 = new Label("qの値:"));
    la2.setBounds (510,70,90,30);
    add(tf1 = new TextField(""+1e-2));
    tf1.setBounds (610,20,80,30);
    add(tf2 = new TextField(""+8e-4));
```

```
    tf2.setBounds (610,70,80,30);

    //Start ボタン
    add(bt1  = new Button("Start"));
    bt1.addActionListener(this);
    bt1.setBounds (610,140,80,30);

    //Clear ボタン
    add(bt2  = new Button("Clear"));
    bt2.addActionListener(this);
    bt2.setBounds (610,190,80,30);

    //TextArea
    add(ta =new TextArea(5,7));
    ta.setBounds (510,280,180,220);

    //スクリーンの確保
    img=createImage(WX,WY);
    gra=img.getGraphics();
    gra.setColor(new Color(230,230,230));
    gra.fillRect(0,0,WX,WY);

    //座標軸とます目
    space(x_min-x_margin, y_min-y_margin
          , x_max+x_margin, y_max+y_margin);

    gra.setColor(Color.gray);
    for (double i=x_min; i<=x_max ; i=i+1){
        gra.drawLine(wx(i),wy(y_min),wx(i),wy(y_max));
    }
    for (double i=y_min; i<=y_max; i=i+10){
        gra.drawLine(wx(x_min),wy(i),wx(x_max),wy(i));
    }

    gra.setColor(Color.black);
    gra.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
    gra.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
    gra.drawString("0",wx(0.0),wy(0.0));
    gra.drawString("5",wx(5),wy(0.0));
    gra.drawString("10",wx(10),wy(0.0));

    gra.setColor(Color.blue);
    gra.drawString("判別式のグラフ",120,20);
    gra.drawString("5000",wx(-1),wy(50));
    gra.drawString("10000",wx(-1),wy(100));
    gra.setColor(Color.red);
    gra.drawString("固有値の実部の最大値のグラフ",220,20);
    gra.drawString("50",wx(0),wy(50));
    gra.drawString("100",wx(0),wy(100));

}

public void paint(Graphics g){
    update(g);
}
public void update(Graphics g){
    g.drawImage(img,0,0,this);
}
```

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == bt1) {

        //一つ前の点と今の点
        double fx1,fx2;
        double fy1,fy2,fy3,fy4;

        //点を結ぶときの幅
        double delta1 = 5e-5;

        space(x_min-x_margin, y_min-y_margin
              , x_max+x_margin, y_max+y_margin);
        ReadFields();

        for(f=0; f<=x_max; f=f+delta1) {

            fx1=f-delta1;
            fx2=f;
            fy1=g1(f-delta1)/100;
            fy2=g1(f)/100;
            fy3 = g2(f-delta1);
            fy4 = g2(f);

            //判別式のグラフの表示
            gra.setColor(Color.blue);
            if(IsIn(fx1,fy1) && IsIn(fx2,fy2)) {
                gra.drawLine(wx(fx1),wy(fy1),wx(fx2),wy(fy2));
                fx1=fx2;fy1=fy2;
            }

            //固有方程式の根のグラフの表示
            gra.setColor(Color.red);
            if(IsIn(fx1,fy3) && IsIn(fx2,fy4)) {
                gra.drawLine(wx(fx1),wy(fy3),wx(fx2),wy(fy4));
                fx1=fx2;fy3=fy4;
            }
        }

        repaint();

        double delta2 = 5e-2;

        //二分法による解の表示
        for(double x=0; x<=x_max; x=x+delta2){

            low=x;
            high=x+delta2;

            if( g1(low)*g1(high)<=0){
                for (k=1;k<=limit;k++){
                    mid=(low+high)/2;
                    if(g1(low)*g1(mid)<0)
                        high=mid;
                    else
                        low=mid;
                    if(g1(mid)==0 || Math.abs(high-low)<Math.abs(low)*EPS){
                        sol=mid;
```

```
            ta.setText(ta.getText()+lastf+"<f<"+sol+"\n");

            if(g1(low)<0){
                ta.setText(ta.getText()+"渦あり");
            }
            else{
                ta.setText(ta.getText()+"渦なし");
            }
            if(g2(low)<0){
                ta.setText(ta.getText()+"安定"+"\n");
            }
            else{
                ta.setText(ta.getText()+"不安定"+"\n");
            }

            break;
        }
        lastf=sol;

        if(k>limit){
            ta.setText(ta.getText()+"収束しない");
        }
    }
}

if ( g2(low)*g2(high)<=0){
    for (k=1;k<=limit;k++){
        mid=(low+high)/2;
        if(g2(low)*g2(mid)<0)
            high=mid;
        else
            low=mid;
        if(g2(mid)==0 || Math.abs(high-low)<Math.abs(low)*EPS){
            sol=mid;
            ta.setText(ta.getText()+lastf+"<f<"+sol+"\n");

            if(g1(low)<0){
                ta.setText(ta.getText()+"渦あり");
            }
            else{
                ta.setText(ta.getText()+"渦なし");
            }
            if(g2(low)<0){
                ta.setText(ta.getText()+"安定"+"\n");
            }
            else{
                ta.setText(ta.getText()+"不安定"+"\n");
            }

            break;
        }
        lastf=sol;

        if(k>limit){
            ta.setText(ta.getText()+"収束しない");
        }
    }
}
```

```
            }

            ta.setText(ta.getText()+sol+"<f"+"\n");

            if(g1(low)<0){
                ta.setText(ta.getText()+"渦あり");
            }
            else{
                ta.setText(ta.getText()+"渦なし");
            }
            if(g2(low)<0){
                ta.setText(ta.getText()+"安定"+"\n");
            }
            else{
                ta.setText(ta.getText()+"不安定"+"\n");
            }

            sol=0;
        }

        //グラフの消去
        else if(e.getSource() == bt2){

            gra.setColor(new Color(230,230,230));
            gra.fillRect(0,0,WX,WY);

            space(x_min-x_margin, y_min-y_margin
                  , x_max+x_margin, y_max+y_margin);

            gra.setColor(Color.gray);
            for (double i=x_min; i<=x_max ; i=i+1){
                gra.drawLine(wx(i),wy(y_min),wx(i),wy(y_max));
            }
            for (double i=y_min; i<=y_max; i=i+10){
                gra.drawLine(wx(x_min),wy(i),wx(x_max),wy(i));
            }

            gra.setColor(Color.black);
            gra.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
            gra.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
            gra.drawString("0",wx(0.0),wy(0.0));
            gra.drawString("5",wx(5),wy(0.0));
            gra.drawString("10",wx(10),wy(0.0));

            gra.setColor(Color.blue);
            gra.drawString("判別式のグラフ",120,20);
            gra.drawString("5000",wx(-1),wy(50));
            gra.drawString("10000",wx(-1),wy(100));
            gra.setColor(Color.red);
            gra.drawString("固有値の実部の最大値のグラフ",220,20);
            gra.drawString("50",wx(0),wy(50));
            gra.drawString("100",wx(0),wy(100));

            repaint();

        }
    }
}
```

6.2.2 実験結果と考察

$\epsilon = 0.01, q = 8 \quad \times 10{-}4$ のとき、

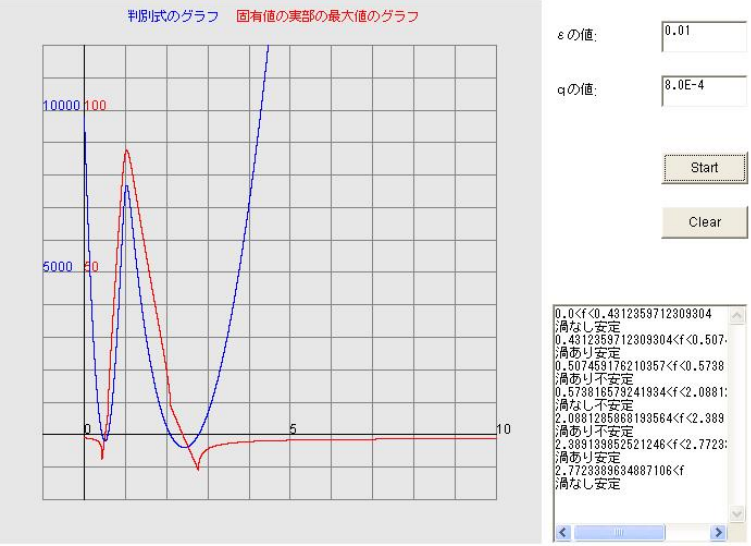


図 6.1: 内藤さんの Anteisei.java より

$0.01 < \epsilon < 0.48, 1 \times 10^{-12} < q < 8 \times 10^{-4}$ で数値実験をした結果、ϵ, q の値を変化させていくと、上記と同じようなパターンが現れる。したがって、ϵ, q の広い範囲で同じパターンが現れることが予想される。すなわち、

$\exists f_1, f_2, f_3, f_4, f_5, f_6 \qquad$ s.t.

$$
\begin{aligned}
f < f_1 &\qquad \text{渦なし安定,} \\
f_1 < f < f_2 &\qquad \text{渦あり安定,} \\
f_2 < f < f_3 &\qquad \text{渦あり不安定,} \\
f_3 < f < f_4 &\qquad \text{渦なし不安定,} \\
f_4 < f < f_5 &\qquad \text{渦あり不安定,} \\
f_5 < f < f_6 &\qquad \text{渦あり安定,} \\
f_6 < f &\qquad \text{渦なし安定}
\end{aligned}
$$

となる。

6.3 解曲線

6.3.1 数値解法 — Runge-Kutta(ルンゲ・クッタ) 法

ルンゲ・クッタ法は微分方程式の数値解法の一つである。常微分方程式の初期値問題の近似解法に関しても、近似式の次数を上げて精度の高い公式を作ることができる。そのような公式の中で、最も広く使われているのは (４次) の Runge-Kutta 法である。

常微分方程式の初期値問題

$$\frac{dx}{dt} = f(t,x) \qquad (t \in I), \tag{6.7}$$

$$x(t_0) = x_0 \tag{6.8}$$

を満たす $x = x(t)$ を求めることを考える。ここで I は $t_0 \in \boldsymbol{R}$ を含む $\boldsymbol{R}$ の区間で、

$$\begin{cases} f : \boldsymbol{R}^{n+1} \supset \Omega \to \boldsymbol{R}^n & \text{連続,} \\ (t_0, x_0) \in \Omega \end{cases}$$

は与えられているとする。

$I = [a,b]$ とする。$N \in \boldsymbol{N}$ に対し、I を N 個の小区間に分ける:

$$a = t_0 < t_1 < t_2 < \cdots < t_N = b$$

このとき、各 t_j における X の値 $x(t_j)$ の近似値 x_j を求める。$h_j := t_{j+1} - t_j \quad (j = 0, 1, \cdots, N-1)$ を刻み幅とする。

$$\begin{cases} k_1 = hf(t_j, x_j) \\ k_2 = hf(t_j + h/2, x_j + k_1/2) \\ k_3 = hf(t_j + h/2, x_j + k_2/2) \\ k_4 = hf(t_j + h, x_j + k_3) \end{cases},$$

$$x_{j+1} = x_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

で $\{x_j\}_{j=0}^{N}$ を計算する方法を Runge-Kutta 法と呼ぶ。

内藤さんの『ＢＺ反応の研究』の中では、この解法を用いている。(ベクトル場など多少の修正あり。)

＜Runge-Kutta 法によるプログラム＞

```
//内藤さんのプログラムを改良
//変更点:  ベクトル場  初期値入力  描画範囲

//<APPLET code="Oregonator_vector1.class"width=1050 height=650></APPLET>

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class Oregonator_vector1 extends Applet implements MouseListener,ActionListener {

    //スクリーン１
    private Image img1;
    private Graphics gra1;

    //スクリーン２
    private Image img2;
    private Graphics gra2;

    // Runge Kutta 法のための引数
    private double h;
    private double [] t;
    private double [] x;
    private double [] z;

    //ユーザーとのインターフェイス
    private double q,eps,xx0,zz0,tt,tt0,cf, x_max, x_min, y_max, y_min, x_margin, y_margin;
    private int n;
    private Label la1,la2,la3,la4,la5,la6,la7,la8,la9,la10,la11,la12,la13;
    private TextField tf1,tf2,tf3,tf4,tf5,tf6,tf7,tf8,tf9,tf10,tf11,tf12,tf13;
    private Button bt1,bt2,bt3,bt4,bt5;

    // x-z 座標系の変換パラメーター
    private double ratiox,ratioy,X0,Y0;
    private int WX = 650, WY = 650;

    private boolean IsIn1(double x, double y) {
return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
    }

    // x-z 座標変換の準備
    private void space1(double x0, double y0, double x1, double y1) {
X0 = x0; Y0 = y0;
ratiox = WX / (x1 - x0);
ratioy = WY / (y1 - y0);
    }

    // x-z ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wx(double x) {
      return (int)(ratiox * (x - X0));
    }

      //逆写像
    private double wxinv(int ix) {
return ix/ratiox + X0 ;
    }

    // x-z ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wy(double y) {
return WY - (int)(ratioy * (y - Y0));
    }

      //逆写像
    private double wyinv(int iy) {
return (WY - iy)/ratioy + Y0 ;
    }
```

```
    // x-t,z-t 座標系の変換パラメーター
    private double ratioa,ratiob,A0,B0;
    private int WA = 350, WB =330;

    // x-t,z-t 描写範囲
    private double b_max =   1.5;
    private double b_min =  -0.5;

    private boolean IsIn2(double a, double b) {
return (tt0 <= a && a <= tt && b_min <= b && b <= b_max);
    }

    // x-t,z-t 座標変換の準備
    private void space2(double b0,double b1) {
A0 = tt0; B0 = b0;
ratioa = WA / (tt - tt0);
ratiob = WB / (b1 - b0);
    }

    // x-t,z-t ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wa(double a) {
      return (int)(ratioa * (a - A0));
    }

    // x-t,z-t ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wb(double b) {
return WB - (int)(ratiob * (b - B0));
    }

    //数値の読み取り
    private void ReadFields(){
q   = Double.valueOf(tf1.getText()).doubleValue();
eps = Double.valueOf(tf2.getText()).doubleValue();
xx0 = Double.valueOf(tf3.getText()).doubleValue();
zz0 = Double.valueOf(tf4.getText()).doubleValue();
h   = Double.valueOf(tf5.getText()).doubleValue();
tt  = Double.valueOf(tf6.getText()).doubleValue();
cf  = Double.valueOf(tf7.getText()).doubleValue();
x_min = Double.valueOf(tf10.getText()).doubleValue();
x_max = Double.valueOf(tf11.getText()).doubleValue();
y_min = Double.valueOf(tf12.getText()).doubleValue();
y_max = Double.valueOf(tf13.getText()).doubleValue();
x_margin = (x_max-x_min)/20;
y_margin = (y_max-y_min)/20;

n   = (int)(10*(Math.abs(tt/h)+1));
    }

    //関数の定義, ベクトル場の定義
    private double f(double x, double z)
    {
     return(x*(1-x)-cf*z*(x-q)/(q+x))/eps;
    }
    private double g(double x, double z)
    {

     return(x-z);
    }
```

```
   //ベクトルの設定

     //ベクトルの長さを均一にする
    double norm(double x, double y) {
return Math.sqrt(x*x+y*y);
    }
    void arrow(double x0, double y0, double vx, double vy) {  /* ベクトルを引く関数をつくる */
double angle = 15 * Math.PI / 180.0;
double cos = Math.cos(angle), sin = Math.sin(angle);
double Norm = norm(vx, vy);
double fx = 0.04 * (x_max - x_min), fy = 0.04 * (y_max - y_min);
vx /= Norm; vy /= Norm;
vx *= fx; vy *= fy;
double xe = x0 + vx, ye = y0 + vy;
vx *= - 0.2; vy *= -0.2;
double vx1= vx*cos-vy*sin;
double vy1= vx*sin+vy*cos;
double vx2= vx*cos+vy*sin;
double vy2=-vx*sin+vy*cos;
if (IsIn1(x0, y0) && IsIn1(xe, ye)) {
    line1(x0, y0, xe, ye);
    line1(xe, ye, xe+vx1, ye+vy1);
    line1(xe, ye, xe+vx2, ye+vy2);
}
    }

   //線を引くメソッドを定義
    private void line1(double x0, double y0, double x1, double y1)
    {
gra1.drawLine(wx(x0),wy(y0),wx(x1),wy(y1));
    }
    private void line2(double x0, double y0, double x1, double y1)
    {
gra2.drawLine(wa(x0),wb(y0),wa(x1),wb(y1));
    }
   //ルンゲクッタ法
    private void runge() {

double t,k1,k2,k3,k4,l1,l2,l3,l4;

//
x   = new double [n+1];
        z   = new double [n+1];

h=(tt-tt0)/n;

x[0]=xx0;
z[0]=zz0;
for(int j=1; j<=n; j++){
t=tt0+h*(j-1);
k1=h*f(x[j-1],z[j-1]);
l1=h*g(x[j-1],z[j-1]);
k2=h*f(x[j-1]+k1/2.0,z[j-1]+l1/2.0);
l2=h*g(x[j-1]+k1/2.0,z[j-1]+l1/2.0);
k3=h*f(x[j-1]+k2/2.0,z[j-1]+l2/2.0);
l3=h*g(x[j-1]+k2/2.0,z[j-1]+l2/2.0);
k4=h*f(x[j-1]+k3,z[j-1]+l3);
```

```
l4=h*g(x[j-1]+k3,z[j-1]+l3);
x[j]=x[j-1]+(k1+2.0*k2+2.0*k3+k4)/6.0;
z[j]=z[j-1]+(l1+2.0*l2+2.0*l3+l4)/6.0;
}
    }

    //パラメーター表示
    public void init() {

setCursor(new Cursor(Cursor.HAND_CURSOR));

//tf la bt のレイアウト
setLayout(null);
add(la1 = new Label("ｑの値:"));
la1.setBounds (650,10,90,20);
add(la2 = new Label("εの値:"));
la2.setBounds (835,10,90,20);
add(la3 = new Label("初期値のｘ座標:"));
la3.setBounds (650,30,90,20);
add(la4 = new Label("初期値のｚ座標:"));
la4.setBounds (835,30,90,20);
add(la5 = new Label("時間刻み幅:"));
la5.setBounds (650,50,90,20);
add(la6 = new Label("ｔ座標の上限:"));
la6.setBounds (835,50,90,20);
add(la7 = new Label("ｆの値:"));
la7.setBounds (650,70,90,20);
add(la8 = new Label("区間の分割数:"));
la8.setBounds (835,70,90,20);
add(la9 = new Label("平衡点の座標:"));
la9.setBounds (650,160,80,20);
add(la10 = new Label(" x の範囲"));
la10.setBounds (650,185,50,20);
add(la11 = new Label("～"));
la11.setBounds (750,185,20,20);
add(la12 = new Label(" z の範囲"));
la12.setBounds (650,205,50,20);
add(la13 = new Label("～"));
la13.setBounds (750,205,20,20);

//初期値などの代入
add(tf1 = new TextField("" +8E-4));
tf1.setBounds (745,10,80,20);
add(tf2 = new TextField("" +1E-2));
tf2.setBounds (930,10,80,20);
add(tf3 = new TextField("" +0.01));
tf3.setBounds (745,30,80,20);
add(tf4 = new TextField("" +0.01));
tf4.setBounds (930,30,80,20);
add(tf5 = new TextField("" +0.001));
tf5.setBounds (745,50,80,20);
add(tf6 = new TextField("" +20.0));
tf6.setBounds (930,50,80,20);
add(tf7 = new TextField("" +0.6));
tf7.setBounds (745,70,80,20);
add(tf8 = new TextField("" +100000));
tf8.setBounds (930,70,80,20);
      add(tf9 = new TextField(""));
```

```
tf9.setBounds (730,160,350,20);
add(tf10 = new TextField("" +0.0));
tf10.setBounds (700,185,50,20);
add(tf11 = new TextField("" +1.0));
tf11.setBounds (770,185,50,20);
add(tf12 = new TextField("" +0.0));
tf12.setBounds (700,205,50,20);
add(tf13 = new TextField("" +1.0));
tf13.setBounds (770,205,50,20);

//ボタンの設定

     //スクリーン 1 Start ボタン
add(bt1  = new Button("Start 1"));
bt1.addActionListener(this);
bt1.setBounds (650,100,175,20);

  //ベクトル場 start ボタン
add(bt2  = new Button("ベクトル場表示"));
bt2.addActionListener(this);
bt2.setBounds (650,120,175,20);

  //スクリーン 1 Clear ボタン
add(bt3  = new Button("Clear 1"));
bt3.addActionListener(this);
bt3.setBounds (650,140,175,20);

  //スクリーン 2 Start ボタン
add(bt4  = new Button("Start 2"));
bt4.addActionListener(this);
bt4.setBounds (825,100,175,20);

  //スクリーン 2 Clear ボタン
add(bt5  = new Button("Clear 2"));
bt5.addActionListener(this);
bt5.setBounds (825,120,175,20);


//スクリーンの設定

           //スクリーン 1 の確保
img1=createImage(WX,WY);
gra1=img1.getGraphics();
gra1.setColor(new Color(230,255,230));
gra1.fillRect(0,0,WX,WY);

   // 座標変換
ReadFields();
space1(x_min-x_margin, y_min-y_margin,
     x_max+x_margin, y_max+y_margin);

   // ラベル
gra1.setColor(Color.black);
gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);

   //スクリーン 2 の確保
img2=createImage(WA,WB);
gra2=img2.getGraphics();
```

```
gra2.setColor(new Color(230,255,230));
gra2.fillRect(0,0,WA,WB);

           //ラベル
gra2.setColor(Color.red);
gra2.drawString("ｔ－ｘ",100,10);
gra2.setColor(Color.orange);
gra2.drawString("ｔ－ｚ",150,10);

gra2.setColor(Color.black);
gra2.drawString("のグラフ",200,10);


addMouseListener(this);

    }

    public void paint(Graphics g){
update(g);
    }
    public void update(Graphics g){
g.drawImage(img1,0,0,this);
g.drawImage(img2,655,320,this);
    }


    //ボタン操作
    public void actionPerformed(ActionEvent e) {

//ボタン 1
if(e.getSource() == bt1) {
             // 初期値などを取得
    ReadFields();

          // Runge Kutta 法で解を計算
    runge();

    // 解曲線を描く
    space1(x_min-x_margin, y_min-y_margin,
   x_max+x_margin, y_max+y_margin);
    gra1.setColor(Color.blue);
    for(int j=1; j<=n; j++) {
if (IsIn1(x[j-1], z[j-1]) && IsIn1(x[j], z[j])) {
    line1( x[j-1], z[j-1], x[j], z[j]);
}
    }

    gra1.setColor(Color.lightGray);
    for (double i=x_min; i<=x_max ; i=i+0.1){
line1( i, y_min, i, y_max);
    }
    for (double i=y_min; i<=y_max; i=i+0.1){
line1( x_min, i, x_max, i);
    }

    // ラベルを描く
    gra1.setColor(Color.black);
    line1( x_min, 0.0, x_max, 0.0);
```

```
    line1( 0.0, y_min, 0.0, y_max);
    gra1.drawString("0",wx(0.0),wy(0.0));
    gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);
    for(double i=0.5; i<=x_max; i=i+0.5){
gra1.drawString(""+i,wx(i),wy(0.0));
    }
    for(double i=0.5; i<=y_max; i=i+0.5){
gra1.drawString(""+i,wx(0.0),wy(i));
    }
    repaint();

    //平衡点を計算する
    ReadFields();
    double x_h =(1-cf-q+Math.sqrt(Math.pow(cf+q-1,2)+4*q*(1+cf)))/2;

    //平衡点をプロット
    gra1.setColor(Color.red);
    double h1= x_h-(x_max-x_min)/100;
    double h2= x_h+(x_max-x_min)/100;
    double h3= x_h-(x_max-x_min)/100;
    double h4= x_h+(x_max-x_min)/100;
    line1( h1, x_h, h2, x_h);
    line1( x_h, h3, x_h, h4);

    //平衡点を表示させる
    String he=Double.toString(x_h);
    tf9.setText("("+ he + "," + he +")");

}

//ボタン 2
//ベクトル場表示
else if(e.getSource() == bt2){
    ReadFields();
    gra1.setColor(Color.green);
    double dx = (x_max - x_min) / 20, dy = (y_max - y_min) / 20;
  for(double x=x_min; x<=x_max; x += dx){   /* 矢印を書く */
      for(double y=y_min; y<=y_max; y += dy){
  arrow(x, y, f(x,y), g(x,y));
      }
  }
  repaint();
}

//ボタン 3
// 相曲線とベクトル場をクリア
  // ... 相図をクリアしてグリッドなど描き直し
else if(e.getSource() == bt3) {
    gra1.setColor(new Color(230,255,230));
    gra1.fillRect(0,0,WX,WY);

    // ラベルを描く
    gra1.setColor(Color.black);
    gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);

    repaint();

    //初期値をはじめの値にする
```

```
    tf3.setText("" +0.01);
    tf4.setText("" +0.01);

    //平衡点の値を消す
    tf9.setText("" );
}

//ボタン 4
   // 解曲線 (t,x(t)), (t,z(t)) を描く
else if(e.getSource() == bt4) {
    double TT,TT0;
    // 初期値などを取得
    ReadFields();
    space2(b_min, b_max);

    //座標軸
    gra2.setColor(Color.black);
    line2( tt0, 0.0, tt, 0.0);
    line2( tt0, b_min, tt0, b_max);
    gra2.drawString( ""+tt0 ,wa(tt0),wb(0.0));
    gra2.drawString( "1.0",wa(tt0),wb(1.0));
    gra2.drawString("t",wa(0.9*tt),wb(0.0));
    gra2.drawString(" x,z",wa(tt0),wb(1.4));

     runge();

    for (int j=1; j<=n; j++) {
TT =tt0+h*j;
TT0=tt0+h*(j-1);
if (IsIn2(TT0, x[j-1]) && IsIn2(TT, x[j])) {
    gra2.setColor(Color.red);
      line2(TT0,x[j-1],TT, x[j]);
      x[j-1]=x[j];
      TT0=TT;
}
if (IsIn2(TT0, z[j-1]) && IsIn2(TT, z[j])) {
    gra2.setColor(Color.orange);
    line2(TT0, z[j-1], TT, z[j]);
      z[j-1]=z[j];
      TT0=TT;
}
    }
    repaint();
}

//ボタン 5
   // 解曲線 (t,x(t)), (t,z(t)) 消去
else if(e.getSource() == bt5) {
    gra2.setColor(new Color(230,255,230));
    gra2.fillRect(0,0,WA,WB);

    gra2.setColor(Color.red);
    gra2.drawString("ｔ－ｘ",100,10);
    gra2.setColor(Color.orange);
    gra2.drawString("ｔ－ｚ",150,10);

    gra2.setColor(Color.black);
    gra2.drawString("のグラフ",200,10);
```

```
    repaint();
}

    }


    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
    public void mouseClicked(MouseEvent e){

//マウスによる初期値入力
// if(e.getModifiers()==){
    space1(x_min-x_margin,y_min-y_margin,x_max+x_margin,y_max+y_margin);
    int xx=e.getX();
    int zz=e.getY();
    double x_m=wxinv(xx);
    double z_m=wyinv(zz);
    String xx0=Double.toString(x_m);
    String zz0=Double.toString(z_m);
    tf3.setText(xx0);
    tf4.setText(zz0);

      // 初期値などを取得
    ReadFields();
              // Runge Kutta 法で解を計算
    runge();

    // 解曲線を描く
    gra1.setColor(new Color(0,0,200));
    for(int j=1; j<=n; j++) {
if (IsIn1(x[j-1], z[j-1]) && IsIn1(x[j], z[j])) {
    line1( x[j-1], z[j-1], x[j], z[j]);
}
    }

    repaint();
}
    public void mouseEntered(MouseEvent e){}
    public void mouseExited(MouseEvent e){}
}
```

これでは、$x \fallingdotseq 0$ で解曲線がうまく描けない。($x > 0$ のはずが、$x < 0$ になってしまう。)

6.3.2 数値解析 — RKF45

E.Fehlberg (1970) は次の公式 RKF45 を提案した。

$$x_{j+1} = x_j + h\left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right).$$

ただし、

$$
\begin{aligned}
k_1 &= f(t_j, x_j) \\
k_2 &= f\left(t_j + \frac{1}{4}h, x_j + \frac{1}{4}hk_1\right) \\
k_3 &= f\left(t_j + \frac{3}{8}h, x_j + \frac{1}{32}h(3k_1 + 9k_2)\right) \\
k_4 &= f\left(t_j + \frac{12}{13}h, x_j + \frac{1}{2197}h(1932k_1 - 7200k_2 + 7296k_3)\right) \\
k_5 &= f\left(t_j + h, x_j + h\left(\frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right)\right) \\
k_5 &= f\left(t_j + \frac{1}{2}h, x_j + h\left(-\frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right)\right).
\end{aligned}
$$

これは6段5次の公式であるが、それだけでなく

$$x^*_{j+1} = x_j + h\left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right)$$

という値を作ると、x^*_{j+1} は $x(t_{j+1})$ に対して４次の近似値となる。この次数の差を利用してステップ幅の自動調節をする方法を以下説明する。　このアイディアのキーは、s 段 m 次公式に、f の値を計算することなく $(m-1)$ 次公式を付随させるところにある。このような Runge-Kutta 型公式を、$(m-1)$ 次公式が m 次公式に埋め込まれているといい、**埋め込み型 Runge-Kutta 法**とよぶ。

＜原理の説明＞

$t = a$ から $t = b$ まで、誤差の大きさの見積もりが、こちらが指定した値 ϵ より小さくなるように解くことを目標にする。このとき ϵ のことを許容誤差限界とよぶ。$[a, b]$ を $a = t_0 < t_1 < \cdots < t_N = b$ と分割して解くことにする。$t = t_n$ における値 x_n まで定まったとする。t_{n+1} における５次公式、４次公式による近似値をそれぞれ x_{n+1}, x^*_{n+1} とすると、

$$x_{n+1} - x(t_{n+1}) = ch^6 + O(h^7), \quad x^*_{n+1} - x(t_{n+1}) = c'h^5 + O(h^6)$$

となる。ただし、$h := t_{n+1} - t_n$ とおいた。これから

$$x_{n+1} - x(t_{n+1}) - x^*_{n+1} - x(t_{n+1}) = -c'h^5 + O(h^6).$$

通常は、x_{n+1} は x^*_{n+1} よりも格段と精度がよいと期待できるので、この式の値が x^*_{n+1} の誤差の良い評価となっていると考えられる。

$$\delta_{n+1} := \| x_{n+1} - x^*_{n+1} \|$$

と置いておく。

さて、$[a, b]$ で解いたときの許容誤差限界を ϵ とするのであるから、$[t_n, t_{n+1}]$ で解いたときの許容誤差限界は

$$\epsilon \cdot \frac{h}{H}, \qquad H := b - a$$

とするのが妥当であろう。それゆえ

$$\delta_{n+1} \leq \frac{\epsilon h}{H}$$

であればよいが、そうでない場合は、h が大きすぎて十分な精度が得られていないと考え、もっと h を小さくすることにする。h の代わりに、$\bar{h} \ll h$ なる $\bar{h}$ を用いて

$$\bar{t}_{n+1} = t_n + \bar{h}$$

とおき、これに対応した $\bar{x}_{n+1}, \bar{x}^*_{n+1}$ を求め、$\bar{\delta}_{n+1} :=\| \bar{x}_{n+1} - \bar{x}^*_{n+1} \|$ とおく。

$$\delta_{n+1} \sim\| c' \|\| h \|^5, \quad \bar{\delta}_{n+1} \sim\| c' \|\| \bar{h} \|^5$$

となると期待できるから、

$$\bar{\delta}_{n+1} \sim \left(\frac{| \bar{h} |}{| h |} \right) \delta_{n+1}.$$

を $\bar{h}$ について解くと

$$\bar{h} \leq h \left(\frac{\epsilon h}{H \delta_{n+1}} \right)^{1/4}$$

安全のために

$$\bar{h} := 0.9h \left(\frac{\epsilon h}{H \delta_{n+1}} \right)^{1/4}$$

で $\bar{h}$ を定めることにする。

＜RKF45 による解法のプログラム＞

```
//Oregonator_vector_mk.java を修正


//<APPLET code="Oregonator_vector_mk.class"width=1050 height=650></APPLET>

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import corejava.*;

public class Oregonator_vector_mk extends Applet implements MouseListener,ActionListener {

    // 最終時刻 tt の初期値
    static final double init_tt = 20.0;
    private double eps_tol = 1e-5;
    private boolean verbose = false;
    //スクリーン 1
    private Image img1;
    private Graphics gra1;

    //スクリーン 2
    private Image img2;
    private Graphics gra2;

    // RKF45 のための引数
    private double h;
    private double hh;
```

```
    private double [] t;
    private double [] x;
    private double [] z;
    private double next_x, next_z, error, xn, zn;

    //ユーザーとのインターフェイス
    private double q,eps,xx0,zz0,tt,tt0,cf,
x_max, x_min, y_max, y_min, x_margin, y_margin;
    private int n, maxn;
    private Label la1,la2,la3,la4,la5,la6,la7,la8,la9,la10,la11,la12,la13;
    private TextField tf1,tf2,tf3,tf4,tf5,tf6,tf7,tf8,tf9,tf10,tf11,tf12,tf13;
    private Button bt1,bt2,bt3,bt4,bt5;
    private TextArea ta1, ta2;

    // x-z 座標系の変換パラメーター
    private double ratiox,ratioy,X0,Y0;
    private int WX = 650, WY = 650;

    private boolean IsIn1(double x, double y) {
return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
    }

    // x-z 座標変換の準備
    private void space1(double x0, double y0, double x1, double y1) {
X0 = x0; Y0 = y0;
ratiox = WX / (x1 - x0);
ratioy = WY / (y1 - y0);
    }

    // x-z ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wx(double x) {
      return (int)(ratiox * (x - X0));
    }

      //逆写像
    private double wxinv(int ix) {
return ix/ratiox + X0 ;
    }

    // x-z ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wy(double y) {
return WY - (int)(ratioy * (y - Y0));
    }

      //逆写像
    private double wyinv(int iy) {
return (WY - iy)/ratioy + Y0 ;
    }

    // x-t,z-t 座標系の変換パラメーター
    private double ratioa,ratiob,A0,B0;
    private int WA = 350, WB =330;

    // x-t,z-t 描写範囲
    private double b_max =   1.5;
    private double b_min =  -0.5;

    private boolean IsIn2(double a, double b) {
```

```
return (tt0 <= a && a <= tt && b_min <= b && b <= b_max);
    }

    // x-t,z-t 座標変換の準備
    private void space2(double b0,double b1) {
A0 = tt0; B0 = b0;
ratioa = WA / (tt - tt0);
ratiob = WB / (b1 - b0);
    }

    // x-t,z-t ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wa(double a) {
      return (int)(ratioa * (a - A0));
    }

    // x-t,z-t ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
    private int wb(double b) {
return WB - (int)(ratiob * (b - B0));
    }

    //数値の読み取り
    private void ReadFields(){
q   = Double.valueOf(tf1.getText()).doubleValue();
eps = Double.valueOf(tf2.getText()).doubleValue();
xx0 = Double.valueOf(tf3.getText()).doubleValue();
zz0 = Double.valueOf(tf4.getText()).doubleValue();
        cf  = Double.valueOf(tf5.getText()).doubleValue();
tt  = Double.valueOf(tf6.getText()).doubleValue();
x_min = Double.valueOf(tf8.getText()).doubleValue();
x_max = Double.valueOf(tf9.getText()).doubleValue();
y_min = Double.valueOf(tf10.getText()).doubleValue();
y_max = Double.valueOf(tf11.getText()).doubleValue();
x_margin = (x_max-x_min)/20;
y_margin = (y_max-y_min)/20;
maxn   = 1000;
t = new double [maxn+1];
x = new double [maxn+1];
z = new double [maxn+1];
x[0] = xx0;
z[0] = zz0;
    }

    //関数の定義, ベクトル場の定義
    private double f(double x, double z)
    {
     return(x*(1-x)-cf*z*(x-q)/(q+x))/eps;
    }
    private double g(double x, double z)
    {

     return(x-z);
    }

   //ベクトルの設定

     //ベクトルの長さを均一にする
    double norm(double x, double y) {
return Math.sqrt(x*x+y*y);
```

```
    }
    void arrow(double x0, double y0, double vx, double vy) {  /* ベクトルを引く
                                                                関数をつくる */
double angle = 15 * Math.PI / 180.0;
double cos = Math.cos(angle), sin = Math.sin(angle);
double Norm = norm(vx, vy);
double fx = 0.04 * (x_max - x_min), fy = 0.04 * (y_max - y_min);
vx /= Norm; vy /= Norm;
vx *= fx; vy *= fy;
double xe = x0 + vx, ye = y0 + vy;
vx *= - 0.2; vy *= -0.2;
double vx1= vx*cos-vy*sin;
double vy1= vx*sin+vy*cos;
double vx2= vx*cos+vy*sin;
double vy2=-vx*sin+vy*cos;
if (IsIn1(x0, y0) && IsIn1(xe, ye)) {
    line1(x0, y0, xe, ye);
    line1(xe, ye, xe+vx1, ye+vy1);
    line1(xe, ye, xe+vx2, ye+vy2);
}
    }

   //線を引くメソッドを定義
    private void line1(double x0, double y0, double x1, double y1)
    {
gra1.drawLine(wx(x0),wy(y0),wx(x1),wy(y1));
    }
    private void line2(double x0, double y0, double x1, double y1)
    {
gra2.drawLine(wa(x0),wb(y0),wa(x1),wb(y1));
    }

    private double max(double a, double b) {
if (a > b)
    return a;
else
    return b;
    }
    private void report(String s) {
String newStr = ta1.getText();
if (newStr.length() < 1000)
    ta1.setText(newStr + s);
else
    ta1.setText(s);
    }
    // RKF45 公式で t から t+h まで解を計算
    // 初期値 (x,z)
    // 結果 (next_x,next_z), 誤差見積もり *error
    private void rkf45_0(double t, double h, double x, double z) {
double kx1,kx2,kx3,kx4,kx5,kx6,ky1,ky2,ky3,ky4,ky5,ky6;
double X,Z,newx,newx2,newz,newz2;

kx1=f(x,z);
ky1=g(x,z);
X = x+h*kx1/4.0; Z = z+h*ky1/4.0;
kx2=f(X, Z);
ky2=g(X, Z);
X = x+h*(3.0*kx1+9.0*kx2)/32.0; Z = z+h*(3.0*ky1+9.0*ky2)/32.0;
```

```
kx3=f(X,Z);
ky3=g(X,Z);
X = x+h*(1932.0*kx1-7200.0*kx2+7296.0*kx3)/2197.0;
Z = z+h*(1932.0*ky1-7200.0*ky2+7296.0*ky3)/2197.0;
kx4=f(X,Z);
ky4=g(X,Z);
X = x+h*(439.0/216.0*kx1-8.0*kx2+3680.0/513.0*kx3-845.0/4104.0*kx4);
Z = z+h*(439.0/216.0*ky1-8.0*ky2+3680.0/513.0*ky3-845.0/4104.0*ky4);
kx5=f(X,Z);
ky5=g(X,Z);
X = x+h*(-8.0/27.0*kx1+2.0*kx2-3544.0/2565.0*kx3+1859.0/4104*kx4-11.0/40.0*kx5);
Z = z+h*(-8.0/27.0*ky1+2.0*ky2-3544.0/2565.0*ky3+1859.0/4104*ky4-11.0/40.0*ky5);
kx6=f(X,Z);
ky6=g(X,Z);
newx=x+h*(16.0/135.0*kx1+6656.0/12825.0*kx3
  +28561.0/56430.0*kx4-9.0/50.0*kx5+2.0/55.0*kx6);
newz=z+h*(16.0/135.0*ky1+6656.0/12825.0*ky3
  +28561.0/56430.0*ky4-9.0/50.0*ky5+2.0/55.0*ky6);
newx2=x+h*(25.0/216.0*kx1+1408.0/2565.0*kx3
   +2197.0/4104.0*kx4-1.0/5.0*kx5);
newz2=z+h*(25.0/216.0*ky1+1408.0/2565.0*ky3
   +2197.0/4104.0*ky4-1.0/5.0*ky5);
next_x = newx;
next_z = newz;
error = norm(newx-newx2,newz-newz2);
    }
    // 時刻 t0 から tn まで解く
    // 初期値は (x0, z0)
    // 結果は xn, zn に入れて返す
    // 許容誤差限界を eps とする
    private void rkf45_1(double t0, double tn,
 double x0, double z0,
 double eps) {
// 最大反復回数 maxiter
int maxiter = 1000;
// 計算機イプシロン程度の数
double epsmac = 1e-15;
int itend; // 最後のステップになるとき 1, そうでないとき 0
double epsv = max(eps, epsmac); // まっとうな許容誤差限界
double eb = Math.abs(epsv / (tn - t0)); // 単位時間あたりの許容誤差限界
Format f_f = new Format("%6.3f");
Format f_e = new Format("%6.3e");
Format f_g = new Format("%6.3g");
double t, h;

//
t = t0;
h = tn - t0;
itend = 0;

ta1.setText(""+f_f.form(t0)+"から"+f_f.form(tn)+"まで解く\n"
   +"(x,z)=("+f_f.form(x0)+","+f_g.form(z0)+")\n");
for (int iter=1; iter<=maxiter; iter++) {
    rkf45_0(t, h, x0, z0);
    xn = next_x; zn = next_z;
    if (verbose)
report("rkf45_0() から戻る\n iter="+iter
       +",(x,z)=("+f_g.form(xn)+","+f_g.form(zn)+"),\n"
```

```
       +"error="+f_e.form(error)+"\n");
    if (error <= Math.abs(eb * h)) { // 計算がちゃんとできた
if (itend == 1) { // 最後の計算だったら
    report(" ちゃんと最後まで行った。\n");
    return;
}
itend = 0;
t += h;
h *= 0.9 * Math.sqrt(Math.sqrt(Math.abs(eb * h) / error));
if (verbose)
    report("t="+f_g.form(t)+"\n"+" 新しい h="+f_e.form(h)+"\n");
if (Math.abs(h) > Math.abs(tn-t)) {
    // もう残り時間は短くて、|h| 未満
    h = tn - t;
    itend = 1;
}
x0 = xn;
z0 = zn;
    }
    else {
// 推定誤差時間刻み幅 h あたりの許容誤差よりも大きい場合
if (error > epsv) {
    h *= 0.1;
    report("h が大きいので小さくした。"+"\n"
   +"新 h="+f_e.form(h)+"\n");
}
else {
    h *= 0.9 * Math.sqrt(Math.sqrt(Math.abs(eb * h) / error));
    if (verbose)
report("新しい h="+f_e.form(h)+"\n");
    if (Math.abs(h) < epsmac) {
// 非常に小さい時間刻み幅が出てきた。もう無理だ！
// 警告を発すべきである mk
report(" h="+f_e.form(h)+" は小さくなりすぎ！\n");
return;
    }
}
    }
} // for ループの終わり
report("rkf45_1(): 最後まで計算したけれど...\n");
    } // rkf45_1() の終わり

    private void rkf45(double ts, double te,
       double []t, double []x, double []z,
       double eps) {
int i;
n = 1000;
if (n > maxn)
    n = maxn;
double dt = (te - ts) / n;
for (i = 0; i <= n; i++)
    t[i] = ts + i * dt;
for (i = 0; i < n; i++) {
    rkf45_1(t[i], t[i+1], x[i], z[i], eps / n);
    x[i+1] = xn;
    z[i+1] = zn;
}
    }
```

```
    //パラメーター表示
    public void init() {
setCursor(new Cursor(Cursor.HAND_CURSOR));

//tf la bt ta のレイアウト
setLayout(null);
add(la1 = new Label("ｑの値:"));
la1.setBounds (650,10,90,20);
add(la2 = new Label("εの値:"));
la2.setBounds (835,10,90,20);
add(la3 = new Label("初期値のｘ座標:"));
la3.setBounds (650,30,90,20);
add(la4 = new Label("初期値のｚ座標:"));
la4.setBounds (835,30,90,20);
        add(la5 = new Label("ｆの値:"));
la5.setBounds (650,50,90,20);
add(la6 = new Label("ｔ座標の上限:"));
la6.setBounds (835,50,90,20);
//add(la7 = new Label("ｆの値:"));
//la7.setBounds (650,70,90,20);
// add(la8 = new Label("区間の分割数:"));
// la8.setBounds (835,70,90,20);
add(la7 = new Label("平衡点の座標:"));
la7.setBounds (650,130,80,20);
add(la8 = new Label(" x の範囲"));
la8.setBounds (650,155,50,20);
add(la9 = new Label("〜"));
la9.setBounds (750,155,20,20);
add(la10 = new Label(" z の範囲"));
la10.setBounds (650,175,50,20);
add(la11 = new Label("〜"));
la11.setBounds (750,175,20,20);
add(ta1 =new TextArea(5,7));        //TextArea
ta1.setBounds (650,200,175,100);

//初期値などの代入
add(tf1 = new TextField("" +8E-4));
tf1.setBounds (745,10,80,20);
add(tf2 = new TextField("" +1E-2));
tf2.setBounds (930,10,80,20);
add(tf3 = new TextField("" +0.01));
tf3.setBounds (745,30,80,20);
add(tf4 = new TextField("" +0.01));
tf4.setBounds (930,30,80,20);
add(tf5 = new TextField("" +0.6));
tf5.setBounds (745,50,80,20);
add(tf6 = new TextField("" +init_tt)); // tt
tf6.setBounds (930,50,80,20);
//add(tf7 = new TextField("" +0.6));
//tf7.setBounds (745,70,80,20);
//add(tf8 = new TextField("" +100000));
//tf8.setBounds (930,70,80,20);
      add(tf7 = new TextField(""));
tf7.setBounds (730,130,350,20);
add(tf8 = new TextField("" +0.0));
tf8.setBounds (700,155,50,20);
add(tf9 = new TextField("" +1.0));
```

```
tf9.setBounds (770,155,50,20);
add(tf10 = new TextField("" +0.0));
tf10.setBounds (700,175,50,20);
add(tf11 = new TextField("" +1.0));
tf11.setBounds (770,175,50,20);

//ボタンの設定

     //スクリーン 1 Start ボタン
add(bt1  = new Button("Start 1"));
bt1.addActionListener(this);
bt1.setBounds (650,70,175,20);

  //ベクトル場 start ボタン
add(bt2  = new Button("ベクトル場表示"));
bt2.addActionListener(this);
bt2.setBounds (650,90,175,20);

  //スクリーン 1 Clear ボタン
add(bt3  = new Button("Clear 1"));
bt3.addActionListener(this);
bt3.setBounds (650,110,175,20);

  //スクリーン 2 Start ボタン
add(bt4  = new Button("Start 2"));
bt4.addActionListener(this);
bt4.setBounds (825,70,175,20);

  //スクリーン 2 Clear ボタン
add(bt5  = new Button("Clear 2"));
bt5.addActionListener(this);
bt5.setBounds (825,90,175,20);

//スクリーンの設定

           //スクリーン 1 の確保
img1=createImage(WX,WY);
gra1=img1.getGraphics();
gra1.setColor(new Color(230,255,230));
gra1.fillRect(0,0,WX,WY);

   // 座標変換
ReadFields();
space1(x_min-x_margin, y_min-y_margin,
       x_max+x_margin, y_max+y_margin);

   // ラベル
gra1.setColor(Color.black);
gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);

   //スクリーン 2 の確保
img2=createImage(WA,WB);
gra2=img2.getGraphics();
gra2.setColor(new Color(230,255,230));
gra2.fillRect(0,0,WA,WB);

           //ラベル
gra2.setColor(Color.red);
```

```
gra2.drawString("ｔ－ｘ",100,10);
gra2.setColor(Color.orange);
gra2.drawString("ｔ－ｚ",150,10);

gra2.setColor(Color.black);
gra2.drawString("のグラフ",200,10);


addMouseListener(this);

    }

    public void paint(Graphics g){
update(g);
    }
    public void update(Graphics g){
g.drawImage(img1,0,0,this);
g.drawImage(img2,655,320,this);
    }

    //ボタン操作
    public void actionPerformed(ActionEvent e) {

//ボタン 1
if(e.getSource() == bt1) {
             // 初期値などを取得
    ReadFields();

    // RKF 45 で解を計算
    rkf45(0.0, tt, t, x, z, eps_tol);

    // 解曲線を描く
    space1(x_min-x_margin, y_min-y_margin,
   x_max+x_margin, y_max+y_margin);
    gra1.setColor(Color.blue);
    for(int j=1; j<=n; j++) {
if (IsIn1(x[j-1], z[j-1]) && IsIn1(x[j], z[j])) {
    line1( x[j-1], z[j-1], x[j], z[j]);
}
    }

    gra1.setColor(Color.lightGray);
    for (double i=x_min; i<=x_max ; i=i+0.1){
line1( i, y_min, i, y_max);
    }
    for (double i=y_min; i<=y_max; i=i+0.1){
line1( x_min, i, x_max, i);
    }

    // ラベルを描く
    gra1.setColor(Color.black);
    line1( x_min, 0.0, x_max, 0.0);
    line1( 0.0, y_min, 0.0, y_max);
    gra1.drawString("0",wx(0.0),wy(0.0));
    gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);
    for (double i=0.5; i<=x_max; i=i+0.5){
gra1.drawString(""+i,wx(i),wy(0.0));
    }
```

```
    for(double i=0.5; i<=y_max; i=i+0.5){
gra1.drawString(""+i,wx(0.0),wy(i));
    }
    repaint();

    //平衡点を計算する
    ReadFields();
    double x_h =(1-cf-q+Math.sqrt(Math.pow(cf+q-1,2)+4*q*(1+cf)))/2;

    //平衡点をプロット
    gra1.setColor(Color.red);
    double h1= x_h-(x_max-x_min)/100;
    double h2= x_h+(x_max-x_min)/100;
    double h3= x_h-(x_max-x_min)/100;
    double h4= x_h+(x_max-x_min)/100;
    line1( h1, x_h, h2, x_h);
    line1( x_h, h3, x_h, h4);

    //平衡点を表示させる
    String he=Double.toString(x_h);
    tf7.setText("("+ he + "," + he +")");

} // end if

//ボタン 2
  //ベクトル場表示
else if(e.getSource() == bt2){
  ReadFields();
  gra1.setColor(Color.green);
  double dx = (x_max - x_min) / 20, dy = (y_max - y_min) / 20;
  for(double x=x_min; x<=x_max; x += dx){   /* 矢印を書く */
      for(double y=y_min; y<=y_max; y += dy){
  arrow(x, y, f(x,y), g(x,y));
      }
  }
  repaint();
}

//ボタン 3
  // 相曲線とベクトル場をクリア
  // ... 相図をクリアしてグリッドなど描き直し
else if(e.getSource() == bt3) {
    gra1.setColor(new Color(230,255,230));
    gra1.fillRect(0,0,WX,WY);

    // ラベルを描く
    gra1.setColor(Color.black);
    gra1.drawString("オレゴネータ（２変数版 ）の解曲線",200,30);

    repaint();

    //初期値をはじめの値にする
    tf3.setText("" +0.01);
    tf4.setText("" +0.01);

    //平衡点の値を消す
    tf7.setText("" );
```

```
    ta1.setText("");
}

//ボタン 4
   // 解曲線 (t,x(t)), (t,z(t)) を描く
else if(e.getSource() == bt4) {
    double TT,TT0;
    // 初期値などを取得
    ReadFields();
    space2(b_min, b_max);

    //座標軸
    gra2.setColor(Color.black);
    line2( tt0, 0.0, tt, 0.0);
    line2( tt0, b_min, tt0, b_max);
    gra2.drawString( ""+tt0 ,wa(tt0),wb(0.0));
    gra2.drawString( "1.0",wa(tt0),wb(1.0));
    gra2.drawString(""+tt,wa(0.9*tt),wb(0.0));
    gra2.drawString(" x,z",wa(tt0),wb(1.4));

    // RKF45 で解を計算
    rkf45(0.0, tt, t, x, z, eps_tol);

    for (int j=1; j<=n; j++) {
  if (IsIn2(t[j-1], x[j-1]) && IsIn2(t[j], x[j])) {
      gra2.setColor(Color.red);
      line2(t[j-1],x[j-1],t[j], x[j]);
  }
  if (IsIn2(t[j-1], z[j-1]) && IsIn2(t[j], z[j])) {
      gra2.setColor(Color.orange);
      line2(t[j-1], z[j-1], t[j], z[j]);
  }
  if (Math.abs(t[j]) >= Math.abs(tt))
      break;

    }
    repaint();
}  // end if

//ボタン 5
   // 解曲線 (t,x(t)), (t,z(t)) 消去
else if(e.getSource() == bt5) {
    gra2.setColor(new Color(230,255,230));
    gra2.fillRect(0,0,WA,WB);

 gra2.setColor(Color.red);
      gra2.drawString("ｔ－ｘ",100,10);
      gra2.setColor(Color.orange);
      gra2.drawString("ｔ－ｚ",150,10);

      gra2.setColor(Color.black);
      gra2.drawString("のグラフ",200,10);

      repaint();
}

    }
```

```
    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
    public void mouseClicked(MouseEvent e){

//マウスによる初期値入力
// if(e.getModifiers()==){
    space1(x_min-x_margin,y_min-y_margin,x_max+x_margin,y_max+y_margin);
    int xx=e.getX();
    int zz=e.getY();
    double x_m=wxinv(xx);
    double z_m=wyinv(zz);
    String xx0=Double.toString(x_m);
    String zz0=Double.toString(z_m);
    tf3.setText(xx0);
    tf4.setText(zz0);

      // 初期値などを取得
    ReadFields();
    // RKF45 で解を計算
    rkf45(0.0, tt, t, x, z, eps_tol);
    // 解曲線を描く
    gra1.setColor(new Color(0,0,200));
    for (int j=1; j<=n; j++) {
if (IsIn1(x[j-1], z[j-1]) && IsIn1(x[j], z[j])) {
    line1( x[j-1], z[j-1], x[j], z[j]);
}
    }

    repaint();
    }  // end MouseClick
    public void mouseEntered(MouseEvent e){}
    public void mouseExited(MouseEvent e){}

}
```

6.3.3 実験結果と考察

定義 limit cycle(リミットサイクル)

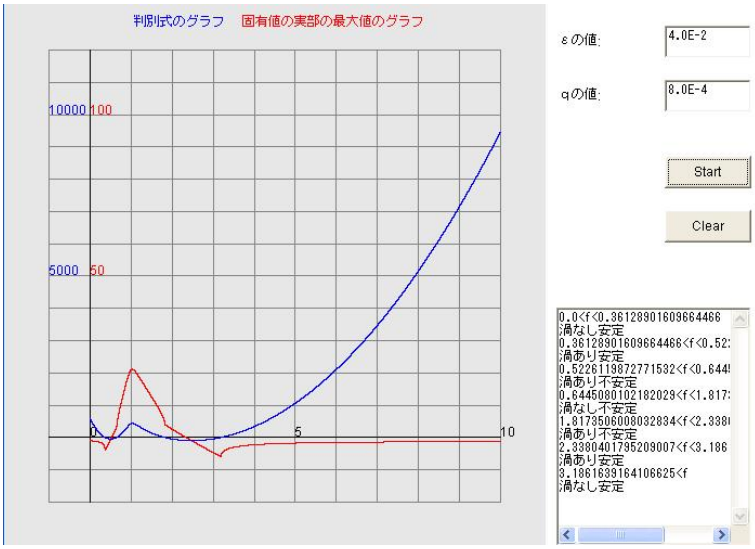
一般に、周期軌道 Γ についてそのある近傍から出発したすべての解が周期軌道 Γ に $t \to \infty$ で巻きついて行くとき、Γ を **limit cycle(極限周期軌道)** という。

limit cycle は、あるパラメーター値を境に出現することがある。

＜結果＞ $\epsilon = 4 \times 10^{-2}, q = 8 \times 10^{-4}$ のときの安定性解析をもとに解曲線を表示

＊安定性解析結果

＊解曲線



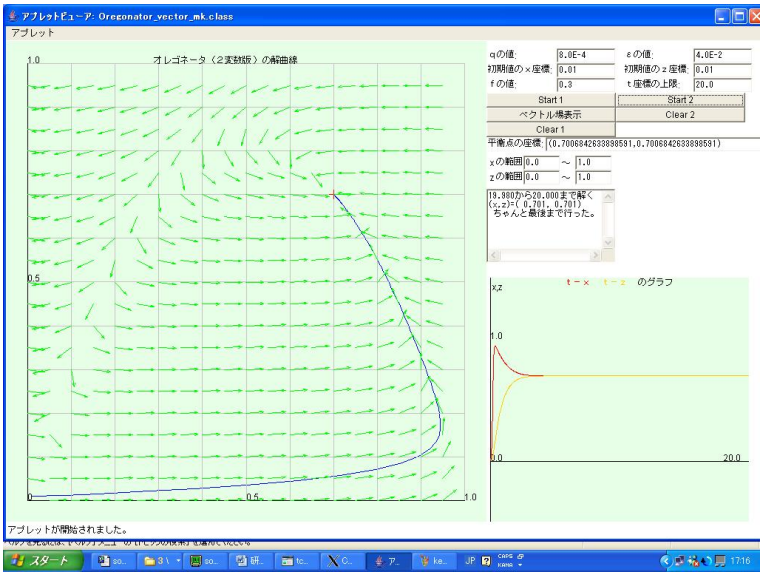


図 6.2: $f = 0.3$ のとき、平衡点に落ち込むことから、渦なし安定

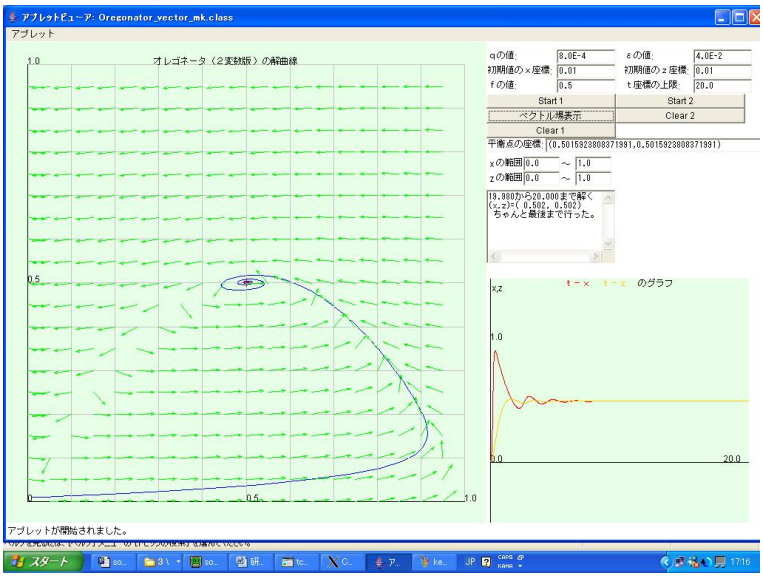


図 6.3: $f = 0.5$ のとき、平衡点の近くで渦を巻きながら平衡点に落ち込んでいく様子がわかる。したがって、渦あり安定

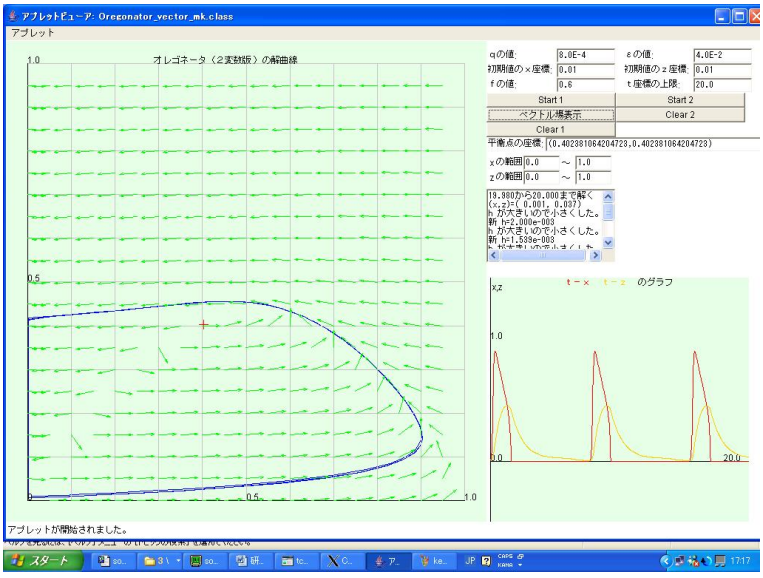


図 6.4: $f = 0.6$ のとき、リミットサイクルになる様子がわかる。渦を見ることはできなかったが、不安定

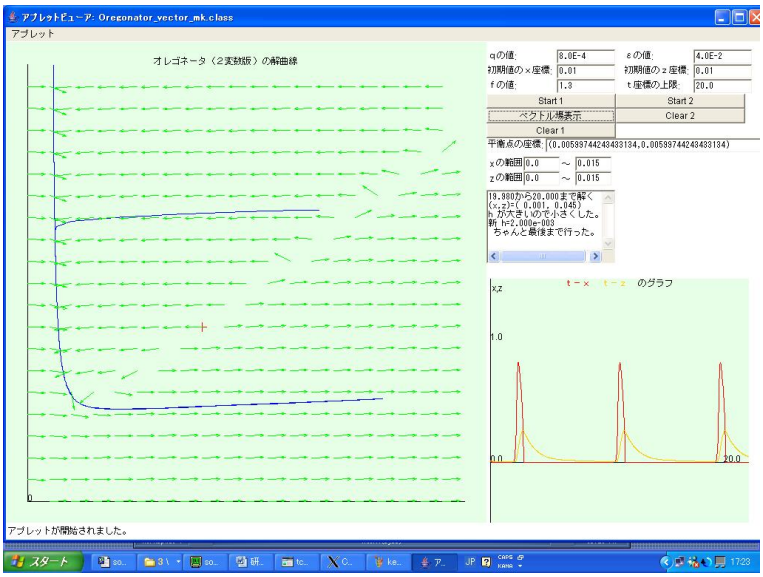


図 6.5: $f = 1.9$ のとき、リミットサイクルになる様子がわかる　渦なし不安定

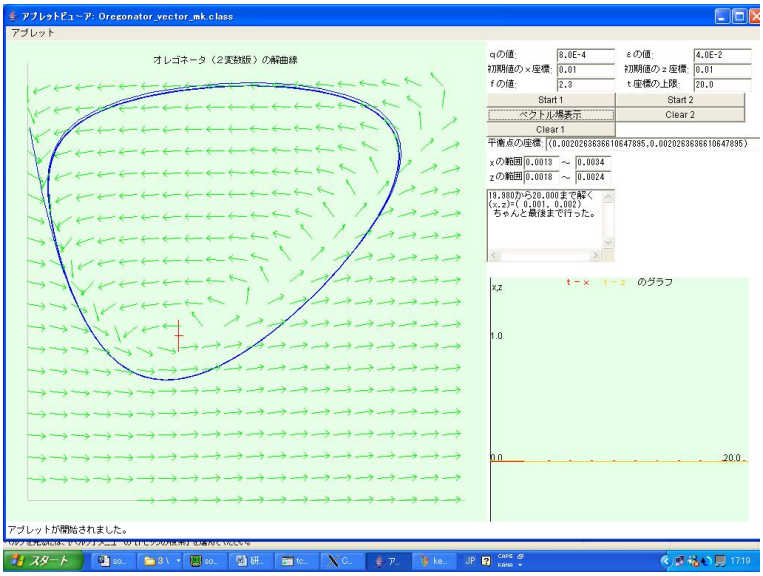


図 6.6: $f = 2.3$ のとき、$f = 0.6$ と同様ではあるが、リミットサイクルがより小さいものになる。こちらは、渦を見ることができ、渦あり不安定

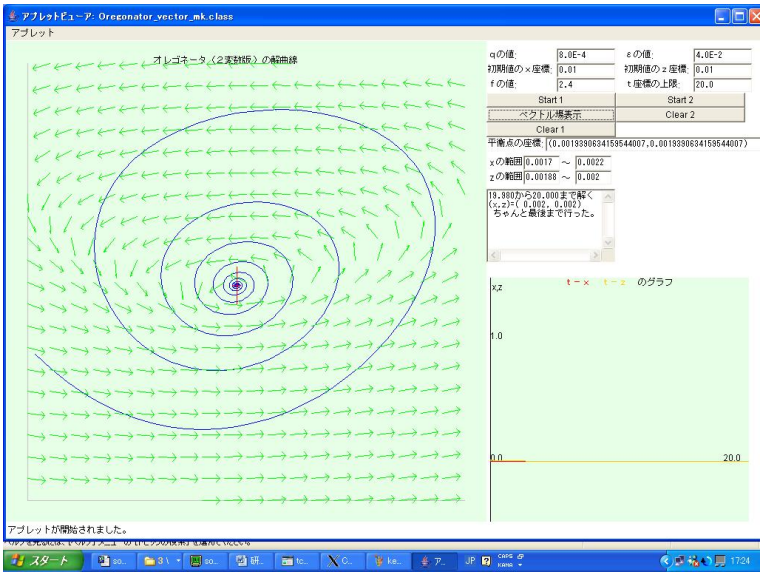


図 6.7: $f = 2.4$ のとき、渦を巻いて平衡点に落ち込む様子がわかる。したがって、渦あり安定

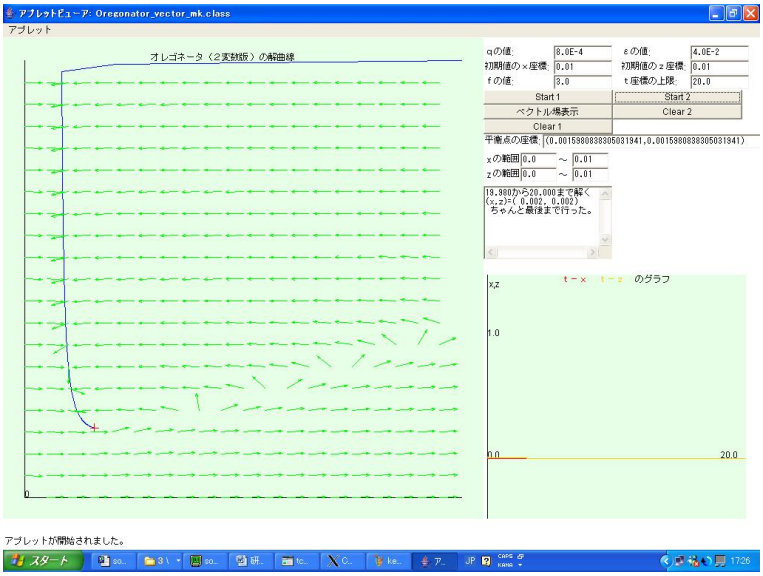


図 6.8: $f = 3.0$ のとき、そのまま、平衡点に落ち込む。したがって、渦なし安定

関連図書

[1] 小畠 陽之助・相沢 洋二 訳
「G. ニコリス, I. プリゴジーヌ　散逸構造—自己秩序形成の物理学的基礎—」、岩波書店 (1999)

[2] 太田 隆夫　「非平衡系の物理学」、裳華房 (2000).

[3] 三池 秀敏・森 義人・山口 智彦　「非平衡系の科学　反応拡散系ダイナミクス」、講談社 (1997).

[4] 笠原 皓司　「数理化学ライブラリー　微分方程式の基礎」、朝倉書店 (1994).

[5] 桂田 祐史　「ＢＺ反応メモ」(2002).

[6] 桂田 祐史　「常微分方程式の初期値問題の数値解法」(2003).

[7] 原野 賢視　「パターン形成のシミュレーション」(2001 年度卒業研究).

[8] 内藤 智　「ＢＺ反応の研究」(2002 卒業研究).