ローレンツ方程式

BV18076　森大樹

2020 年 11 月 15 日

1　勉強背景

ローレンツ方程式を大学の授業で扱い, 興味を持った.

2　ローレンツ方程式とは

ローレンツ方程式とは, $\alpha, \beta, \gamma \geq 0$ を定数として, $\mathbb{R}^3$ 上の次の微分方程式系のことをいう.

$$\begin{aligned}
\frac{dx}{dt} &= f(t,x,y,z) = -\alpha x + \alpha y \\
\frac{dy}{dt} &= g(t,x,y,z) = -zx + \beta x - y \\
\frac{dz}{dt} &= p(t,x,y,z) = xy - \gamma z
\end{aligned}$$

これは, 対流の振る舞いを記述する数理モデルとしてローレンツによって研究された方程式である. また, この方程式はカオスであることが知られている. 初期値の少しの誤差が時間がたつにつれ大きくなることや有界な範囲に軌道が収まることなどの特徴を持つ. ローレンツ方程式の軌道は係数を適当な範囲から選ぶとき, 長時間追跡したときに蝶が羽を広げたような図形になることが知られている. さらに, 初期値を変えても同じような図形が現れることが知られている. 特にローレンツの論文には $\alpha = 10,\ \beta = 28,\ \gamma = \frac{8}{3}$ が与えられていた.

有界な範囲に収まることを示す.

証明. $J(x,y,z) = \beta x^2 + \alpha y + \alpha(z - 2\beta)$ とおくと (楕円体の式),

$$\begin{aligned}
\frac{d}{dt}J(x,y,z) &= 2\beta x\frac{dx}{dt} + 2\alpha y\frac{dy}{dt} + 2\alpha(z-2\beta)\frac{d}{dt}(z-2\beta) \\
&= 2\beta x(-\alpha x + \alpha y) + 2\alpha y(-zx + \beta x - y) + 2\alpha(z-2\beta)(xy - \gamma z) \\
&= -2\alpha(\beta x^2 - \beta xy + xyz - \beta xy + y^2 - xyz + 2\beta xy + \gamma z^2 - 2\beta\gamma z) \\
&= -2\alpha(\beta x^2 + y^2 + \gamma z^2 - 2\beta\gamma z) \\
&= -2\alpha\{\beta x^2 + y^2 + \gamma(z^2 - 2\beta z + \beta^2 - \beta^2)\} \\
&= -2\alpha\{\beta x^2 + y^2 + \gamma(z-\beta)^2 - \gamma\beta^2\}
\end{aligned}$$

$\frac{d}{dt}J(x,y,z) \geq 0$ となるような (x,y,z) の集合を P とすると P は $\mathbb{R}^3$ 内の有界な閉区間になる ($\because \{\beta x^2 + y^2 + \gamma(z-\beta)^2 - \gamma\beta^2\} \leq 0$ より $beta x^2 + y^2 + \gamma(z-\beta)^2 \leq \gamma\beta^2$となるから. つまり, 楕円体の内部にあるから.).P 内での $J(x,y,z)$ の最大値を M と すると, $M < J(x,y,z)$ を満たす任意の (x,y,z) では, $\frac{d}{dt}J(x,y,z) < 0$ であることと $J(x,y,z) \geq 0$ であるから $J(x,y,z)$ は有限時間内に M 以下またはある値に収束すると考えられる. よって, $\lim_{t\to\infty} J(x,y,z) < R$ となる $R \in \mathbb{R}_{>0}$ となる R が存在するのでローレンツ方程式は有界な範囲に軌道が収まることがわかる. □

3　4 段 4 次ルンゲ・クッタ法のスキーム

4 段 4 次ルンゲ・クッタ法のスキーム：(h:刻み幅,X_0, Y_0, Z_0 は初期値)

$$
\begin{cases}
x_{n+1} = & x_n + h\Phi_1(t_n, x_n, y_n, z_n) \\
y_{n+1} = & y_n + h\Phi_2(t_n, x_n, y_n, z_n) \\
z_{n+1} = & z_n + h\Phi_3(t_n, x_n, y_n, z_n) \quad (n = 0, 1, 2, \cdots) \\
& \text{where } \Phi_1(t_n, x_n, y_n, z_n) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
& \qquad \Phi_2(t_n, x_n, y_n, z_n) = \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \\
& \qquad \Phi_3(t_n, x_n, y_n, z_n) = \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) \\
& k_1 = f(t_n, x_n, y_n, z_n) \\
& l_1 = g(t_n, x_n, y_n, z_n) \\
& m_1 = p(t_n, x_n, y_n, z_n) \\
& k_2 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1, y_n + \frac{h}{2}l_1, z_n + \frac{h}{2}m_1) \\
& l_2 = g(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1, y_n + \frac{h}{2}l_1, z_n + \frac{h}{2}m_1) \\
& m_2 = p(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1, y_n + \frac{h}{2}l_1, z_n + \frac{h}{2}m_1) \\
& k_3 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2, y_n + \frac{h}{2}l_2, z_n + \frac{h}{2}m_2) \\
& l_3 = g(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2, y_n + \frac{h}{2}l_2, z_n + \frac{h}{2}m_2) \\
& m_3 = p(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2, y_n + \frac{h}{2}l_2, z_n + \frac{h}{2}m_2) \\
& k_4 = f(t_n + h, x_n + hk_2, y_n + hl_2, z_n + hm_3) \\
& l_4 = g(t_n + h, x_n + hk_2, y_n + hl_2, z_n + hm_3) \\
& m_4 = p(t_n + h, x_n + hk_2, y_n + hl_2, z_n + hm_3) \\
\\
x_0 = X_0 \\
y_0 = Y_0 \\
z_0 = Z_0
\end{cases}
$$

4　ソースコード (C 言語)

ローレンツ方程式を 4 段 4 次ルンゲ・クッタ法により, 近似する解を求めるソースコードである. このとき,t を 0 から 20 までとし,x, y, z の初期値をそれぞれ 20,20,20 とする. また, 刻み幅を 20 を 1000 で割ったもの (ソースコード内では刻み幅は dt で表されている) とする.

ソースコード 1　4 段 4 次ルンゲ・クッタ法

```
1  #include <stdio.h>
2  #include <math.h>
3
4  /*--- 時刻からまで計算T0T1 ---*/
5  #define T0    0.0
6  #define T1    20.0
7
8  /*--- 初期値の設定 ---*/
9  #define X0    1.0
10 #define Y0    0.0
11 #define Z0    0.0
12
13 /*--- (T0, T1)の分割数 ---*/
14 #define N     1000
15
```

```
/*--- 方程式右辺 f(t,x,y,z), g(t,x,y,z), h(t,x,y,z) の設定 ---*/
double f(double t, double x, double y, double z) { return -10 * x+10*y; }
double g(double t, double x, double y, double z) { return -z*x+28*x-y; }
double h(double t, double x, double y, double z) { return x*y-(8/3)*z; }

/*--- メイン関数 ---*/
int main(void)
{
    int i;
    double dt;       // 刻み幅
    double x, y, z;  // 近似解
    double t;       // 時刻
    double k_1, k_2, k_3, k_4,l_1,l_2,l_3,l_4,j_1,j_2,j_3,j_4;

    dt = (T1 - T0) / N;

    /* 初期値の設定  */
    x = X0;
    y = Y0;
    z = Z0;
    t = 0.0;

    printf("%6.4f %16.14f %16.14f %16.14f \n", t, x, y, z);
    for (i = 1; i <= N; i++)
    {
        t = T0 + i * dt;

        k_1 = f(t, x, y, z);
        l_1 = g(t, x, y, z);
        j_1 = h(t, x, y, z);
        k_2 = f(t + dt / 2.0, x + dt * k_1 / 2.0, y + dt * l_1 / 2.0, z + dt * j_1 / 2.0);
        l_2 = g(t + dt / 2.0, x + dt * k_1 / 2.0, y + dt * l_1 / 2.0, z + dt * j_1 / 2.0);
        j_2 = h(t + dt / 2.0, x + dt * k_1 / 2.0, y + dt * l_1 / 2.0, z + dt * j_1 / 2.0);
        k_3 = f(t + dt / 2.0, x + dt * k_2 / 2.0, y + dt * l_2 / 2.0, z + dt * j_2 / 2.0);
        l_3 = g(t + dt / 2.0, x + dt * k_2 / 2.0, y + dt * l_2 / 2.0, z + dt * j_2 / 2.0);
        j_3 = h(t + dt / 2.0, x + dt * k_2 / 2.0, y + dt * l_2 / 2.0, z + dt * j_2 / 2.0);
        k_4 = f(t + dt, x + dt * k_3, y + dt * l_3, z + dt * j_3);
        l_4 = g(t + dt, x + dt * k_3, y + dt * l_3, z + dt * j_3);
        j_4 = h(t + dt, x + dt * k_3, y + dt * l_3, z + dt * j_3);

        x = x + dt / 6 * (k_1 + 2.0 * k_2 + 2.0 * k_3 + k_4);
        y = y + dt / 6 * (l_1 + 2.0 * l_2 + 2.0 * l_3 + l_4);
        z = z + dt / 6 * (j_1 + 2.0 * j_2 + 2.0 * j_3 + j_4);

        /* 各ステップでの時刻、近似値(成分x,成分y)の表示 */
        printf("%6.4f %16.14f %16.14f %16.14f \n", t, x, y, z);
    }
}
```

その結果が下にある図 1 である. このように蝶々の羽を広げたような形になっている (gnuplot で描写).

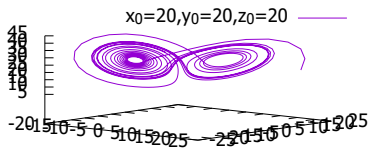

図 1　初期値 (x=20,y=20,z=20)

5　まとめ

今回はローレンツ方程式についてまとめたが他のカオスについて知りたいと思った.

参考文献

[1] ”雑感等”＜ https://kazmus.hatenablog.jp/entry/2017/03/02/204925 ＞,2020 年 11 月 8 日.

[2] ”ベクトル場のホモクリニック分岐と Lorenz-type attractor” ＜ http://www.kurims.kyoto-u.ac.jp/ kyodo/kokyuroku/contents/pdf/0806-09.pdf ＞,2020 年 11 月 20 日

[3] 俣野博 (2007) 『微分方程式入門-基礎から抱擁へ』 岩波書店.