常微分方程式の解法

目次

1　はじめに

本レポートでは常微分方程式の数値解法やその安定性について述べる. なお, 実験で用いたプログラムは付録に記す.

2　課題 3

常微分方程式の初期値問題

$$u' = u, \;\; u(0) = 1 \tag{1}$$

を考える. まず, 時間変数を離散化し, ステップ幅 $\Delta t = \frac{1}{2^i}$, ステップ数 $N = 2^i$ とする. 後述する 5 種類のアルゴリズムを用いて, この方程式を $t = 1$ まで数値的に計算し, 厳密解との誤差を求める. ここで,$t = 1$ における各 Δt での数値解 $u_N^{(i)}(1)$ と厳密解 $\exp(1)$ との差を $E(\Delta t) = |u_N^{(i)}(1) - e|$ とする.
続いて,$E(\Delta t)$ の関数形を理論的に予測し, アルゴリズムごとに実験して確かめる.
用いるアルゴリズムを以下に記す.

- 前進オイラー法
- 2 次アダムス・バッシュフォース法
- 3 次アダムス・バッシュフォース法
- 2 次ルンゲ・クッタ法
- 4 次ルンゲ・クッタ法

2.1　各アルゴリズムの詳細

高次精度一段法であるルンゲ・クッタ法とそれ以外で構築過程が異なるため,2 節に分けて記す.
なお, 前述のとおりステップ幅 $\Delta t = \frac{1}{2^i}$, ステップ数 $N = 2^i$ で離散化している. なお, 実際の計算では必要な初期値には厳密解を用いる.

2.1.1　ルンゲ・クッタ法以外のアルゴリズム

ルンゲ・クッタ法以外のアルゴリズムについては, 構築の初期段階は共通している.
1 変数の常微分方程式を考える.

$$u' = f(t, u(t)), \;\; u(0) = u_0. \tag{2}$$

この式を $t \in [t_{n-1}, t_n]$ で積分する.

$$u_n - u_{n-1} = \int_{t_{n-1}}^{t_n} f(\tau, u(\tau)) d\tau. \tag{3}$$

本節内の 3 つのアルゴリズムでは, この右辺の被積分関数の近似方法が異なる.

■前進オイラー法　前進オイラー法では, 式 3 の被積分関数を $f(t_{n-1}, u_{n-1})$ に置き換える.
$f(t_{n-1}, u_{n-1})$ の値は τ に依らないので, 結果,

$$u_n \approx u_{n-1} + f(t_{n-1}, u_{n-1})\Delta t, \tag{4}$$

が得られる.

■2 次アダムス・バッシュフォース法　2 次アダムス・バッシュフォース法では, 式 3 の被積分関数を,τ に関する 1 次多項式 $p_1(\tau)$ で近似する.

$$\int_{t_{n-1}}^{t_n} f(\tau, u(\tau))d\tau \approx \int_{t_{n-1}}^{t_n} p_1(\tau)d\tau. \tag{5}$$

$f_n = f(t_n, u_n)$ として,$p_1(\tau)$ を $(t_{n-1}, f_{n-1}), (t_{n-2}, f_{n-2})$ を用いた Lagrange 補間で構成する. 結局,

$$u_n \approx u_{n-1} + \frac{\Delta t}{2}(3f_{n-1} - f_{n-2}), \tag{6}$$

が得られる.

■3 次アダムス・バッシュフォース法　3 次アダムス・バッシュフォース法では,2 次のときの $p_1(\tau)$ の代わりに $p_2(\tau)$ を用いる. 結局,

$$u_n \approx u_{n-1} + \frac{\Delta t}{12}(23f_{n-1} - 16f_{n-2} + 5f_{n-3}), \tag{7}$$

が得られる.

2.1.2　ルンゲ・クッタ法

ルンゲ・クッタ法は, 高次精度一段法に分類されるアルゴリズムである. 高次精度一段法は, 与えられていない初期値を計算する必要があるという多段法の問題を解消するため, 未知の値の近似をアルゴリズムに取り込んだ手法である.

■2 次ルンゲ・クッタ法　2 次ルンゲ・クッタ法では, 陰解法のクランク・ニコルソン法の式

$$u_n \approx u_{n-1} + \frac{\Delta t}{2}(f(t_{n-1}, u_{n-1}) + f(t_n, u_n)), \tag{8}$$

に登場する未知の値 u_n を,$u_n^* = u_{n-1} + f_{n-1}\Delta t$ で近似する. 結局,

$$\begin{cases} u_n \approx u_{n-1} + \dfrac{\Delta t}{2}(f(t_{n-1}, u_{n-1}) + f(t_n, u_n^*)), \\ u_n^* = u_{n-1} + f_{n-1}\Delta t, \end{cases} \tag{9}$$

となる.

■4 次ルンゲ・クッタ法　4 次ルンゲ・クッタ法は

$$u_n = u_{n-1} + \frac{\Delta t}{6}(F_1 + F_2 + F_3 + F_4), \tag{10}$$

$$\begin{cases} F_1 = f(t_{n-1}, u_{n-1}), \\ F_2 = f(t_{n-1} + \Delta t/2, u_{n-1} + F_1 \Delta t/2), \\ F_3 = f(t_{n-1} + \Delta t/2, u_{n-1} + F_2 \Delta t/2), \\ F_4 = f(t_n, u_{n-1} + F_3 \Delta t), \end{cases} \tag{11}$$

と書ける.

2.2　理論的予測と計算結果

2.2.1　理論的予測

n 次のアルゴリズムにおいて,1 ステップにおける誤差は $O((\Delta t)^{n+1})$ である. ステップ幅 Δt での $t = 1$ までのステップ数は $1/\Delta t$ であるから, $t = 1$ での誤差の次数は $(\Delta t)^{n+1} \cdot (\Delta t)^{-1} = (\Delta t)^n$ になることが予想される.

2.2.2　計算結果

計算結果を図 1 に示す. 両対数グラフで直線になっていることから,$E(dt)$ はべき関数であることがわかる. また, 傾きをべき関数と比較することにより, アルゴリズムごとの $E(dt)$ の次数がわかる.
アルゴリズムごとに指数をまとめて表 1 に示す.

表 1: アルゴリズムごとの $E(dt)$ の指数

アルゴリズム	指数
前進オイラー法	1
2 次アダムス・バッシュフォース法	2
3 次アダムス・バッシュフォース法	3
2 次ルンゲ・クッタ法	2
4 次ルンゲ・クッタ法	4

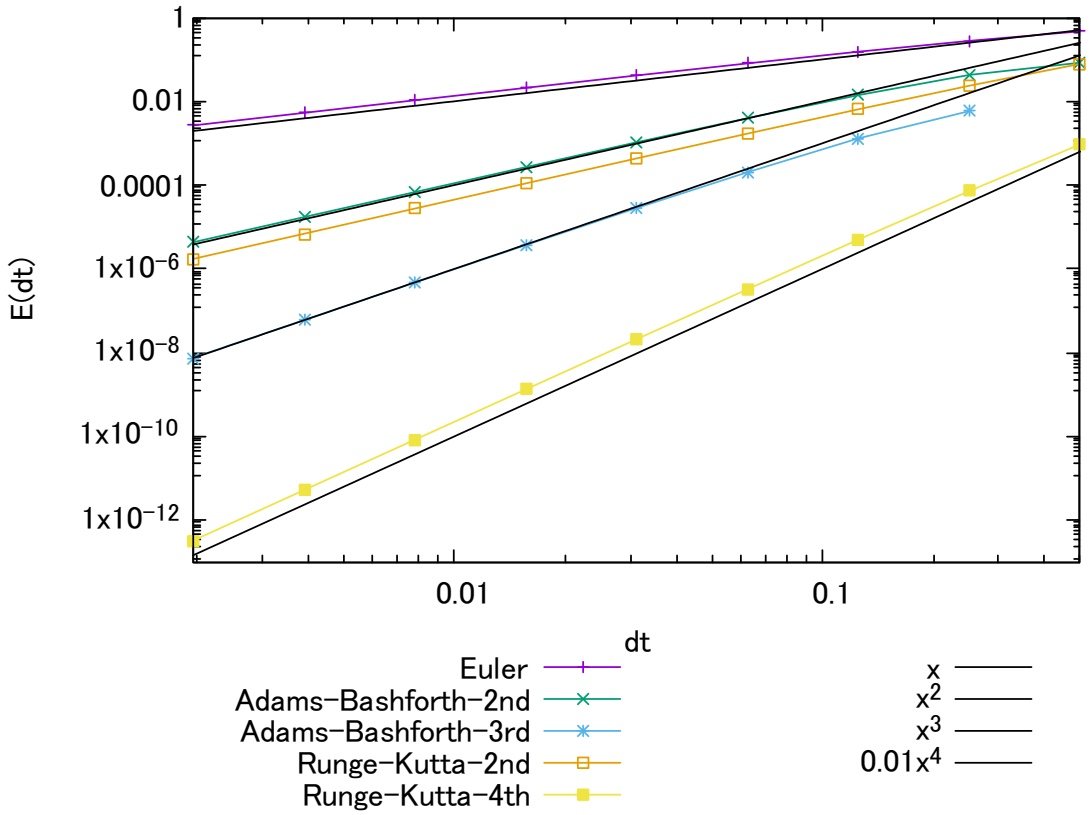


図 1: 各手法の $E(dt)$ の値

3　課題 4

3.1　アルゴリズムの安定性

常微分方程式

$$u' = -\alpha u + \beta, \quad u(0) = u_0, \quad \alpha > 0, \quad \beta \in \mathbb{R}, \tag{12}$$

を用いて, クランク・ニコルソン法 (CN) と 2 次ルンゲ・クッタ法 (RK2) の安定性を調べる.

3.1.1　理論的考察

まず, 差分式

$$u_n = a_1 u_{n-1} + a_0, \tag{13}$$

を満たす数列 $\{u_n\}$ を求める. 上の漸化式を解くと,

$$u_n = (u_0 - \frac{a_0}{1-a_1})a_1^n + \frac{a_0}{1-a_1}, \tag{14}$$

が得られる. よって, 式 13 が発散するための必要十分条件は $|a_1| > 1$ である.
以上を踏まえて, 各アルゴリズムの安定性を議論する.

■クランク・ニコルソン法　クランク・ニコルソン法の更新式は以下である.

$$u_n = u_{n-1} + \frac{\Delta t}{2}(f_{n-1} + f_n). \tag{15}$$

いま,$f_n = -\alpha u_n + \beta$ だから, 結局

$$u_n = \frac{2 - \alpha\Delta t}{2 + \alpha\Delta t} u_{n-1} + \frac{2\beta}{1 + \alpha\Delta t}\Delta t. \tag{16}$$

$\alpha > 0, \Delta t > 0$ より, 常に $\frac{2-\alpha\Delta t}{2+\alpha\Delta t} < 1$ となる. よって, クランク・ニコルソン法は任意の Δt に対して安定である.

■2 次ルンゲ・クッタ法　2 次ルンゲ・クッタ法の更新式は以下である.

$$u_n = u_{n-1} + \frac{\Delta t}{2}(f_{n-1} + f(t_n, u_{n-1} + f_{n-1}\Delta t)). \tag{17}$$

これに $f_{n-1} = -\alpha u_{n-1} + \beta, f(t_n, u_{n-1} + f_{n-1}\Delta t) = -\alpha(u_{n-1} + f_{n-1}\Delta t) + \beta$ を代入して整理すれば

$$u_n = \{1 - \alpha\Delta t + \frac{\alpha^2}{2}(\Delta t)^2\} u_{n-1} + \frac{\beta\Delta t}{2}(2 - \alpha\Delta t) \tag{18}$$

したがって,2 次ルンゲ・クッタ法は $|1 - \alpha\Delta t + \frac{\alpha^2}{2}(\Delta t)^2| < 1$ のとき, すなわち $0 < \Delta t \leq \frac{2}{\alpha}$ のとき安定で,$\Delta t > \frac{2}{\alpha}$ のとき発散する.

3.1.2　数値実験の結果

式 12 で $u_0 = 1, \alpha = 10, \beta = 1$ としたときの厳密解は $u = \frac{9}{10}e^{-10t} + \frac{1}{10}$ である. 先ほどの議論によれば,2 次ルンゲ・クッタ法では $\Delta t > 0.2$ で数値解が発散するはずである. 図 2,3 に各アルゴリズムによる数値解を示す.

CN 法ではどの dt に対しても同じような挙動を示すのに対し,2 次 RK 法では $dt = 0.205 > 2/\alpha$ で値が発散している.

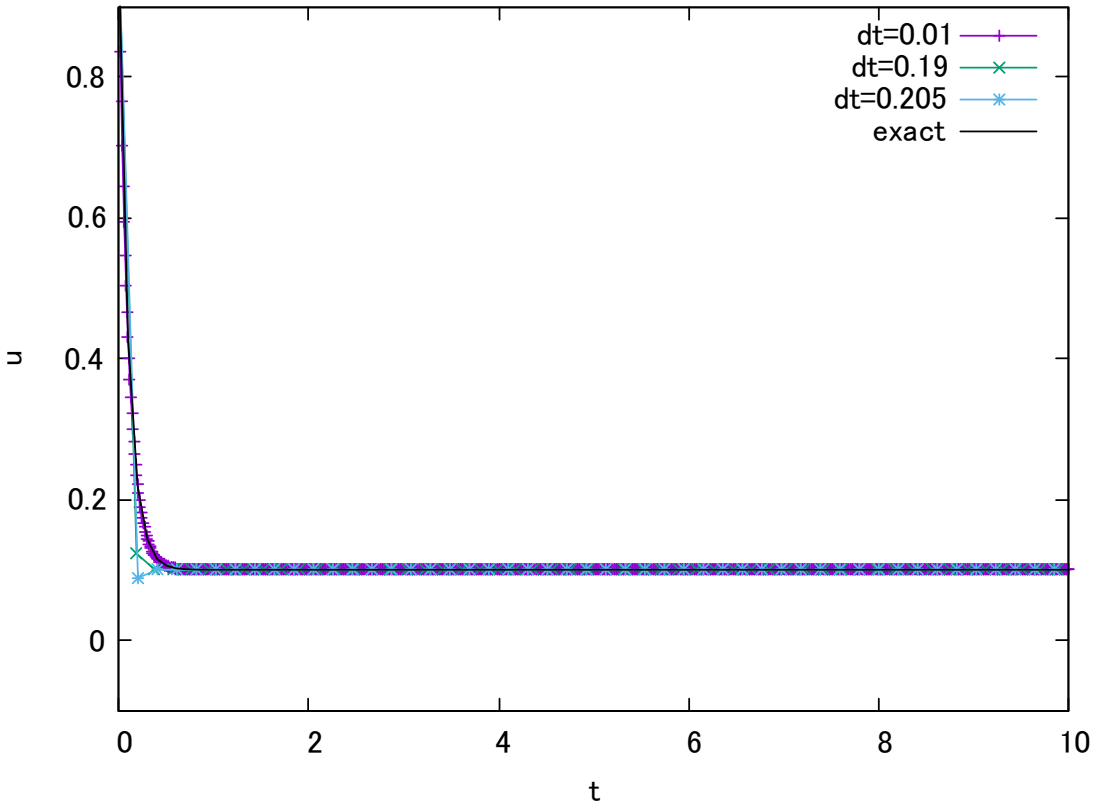


図 2: CN 法

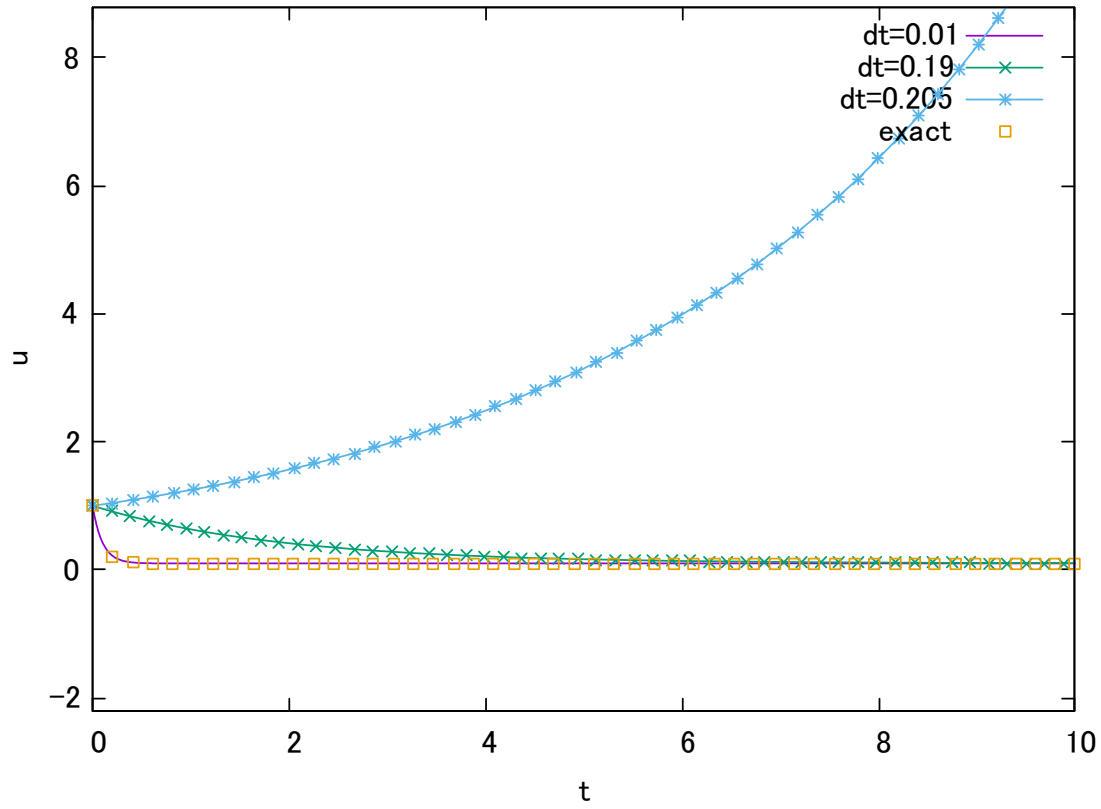


図 3: 2 次 RK 法

4 課題 5

微分方程式

$$u' = -2u + 1, \tag{19}$$

の厳密解は

$$u = (u_0 - \frac{1}{2})\exp(-2t) + \frac{1}{2}, \tag{20}$$

である. よって初期値に依らず $\lim_{t\to\infty} u = \frac{1}{2}$ となる.
アルゴリズム

$$u_n = u_{n-2} + 2\Delta t f(t_{n-1}, u_{n-1}) \tag{21}$$

の安定性について考え, 実際に数値計算を行う.

4.1 アルゴリズムの安定性

いま,$f(t_{n-1}, u_{n-1}) = -2u_{n-1} + 1$ だから, 式 21 は

$$u_n = u_{n-2} + 2(-2u_{n-1} + 1)\Delta t \tag{22}$$

$$u_n + 4\Delta t u_{n-1} - u_{n-2} = 2\Delta t \tag{23}$$

となる. この方程式の同次解を求める. 特性方程式

$$\lambda^2 + 4\Delta t\lambda - 1 = 0 \tag{24}$$

の解は

$$\lambda = -2\Delta t \pm \sqrt{4(\Delta t)^2 + 1} \tag{25}$$

になるから, もとの方程式の同次解は $A(-2\Delta t - \sqrt{4(\Delta t)^2 + 1})^n + B(-2\Delta t + \sqrt{4(\Delta t)^2 + 1})^n$ となるが, 任意の Δt に対して $|-2\Delta t - \sqrt{4(\Delta t)^2 + 1}| > 1$ となるので, 数値解は $n \to \infty$ で発散する.

4.2 数値実験の結果

数値解が発散することを実際に確かめる. 図 4 に計算結果を示す. 初期値には厳密解の値を用いた.

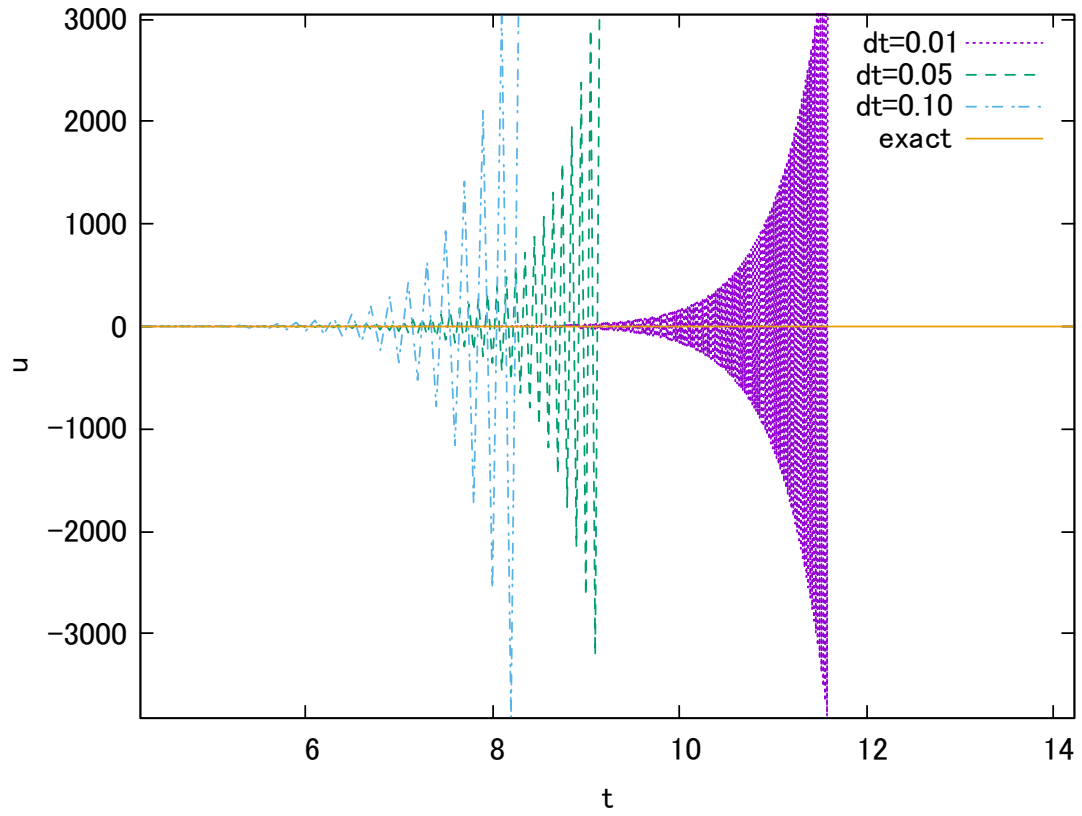


図 4: 発散の様子.±3000 を超えた部分は切り落としている.

Δt の値によって発散の速さが違うものの, どの Δt においても発散しているのがわかる.

5 課題 7

ローレンツ方程式

$$\begin{aligned} x' &= \sigma(y - x) \\ y' &= rx - y - xz \\ z' &= xy - bz \end{aligned} \tag{26}$$

を考える. 初期値を $x(0), y(0), z(0) = (1 + \epsilon, 0, 0)$ とし, 定数を $\sigma = 10, b = 8/3, r = 28$ とする. この方程式について, ステップ幅 $\Delta t = 0.01$ の 4 次ルンゲクッタ法を用い, 時間区間 $t \in [0, 100]$ における軌道を求める.

5.1 $\epsilon = 0$ での軌道

$\epsilon = 0$ とする. このときの $x(t)$ と軌道 $(x(t), y(t), z(t))$ を求める. 計算結果を図 5,6 に示す.

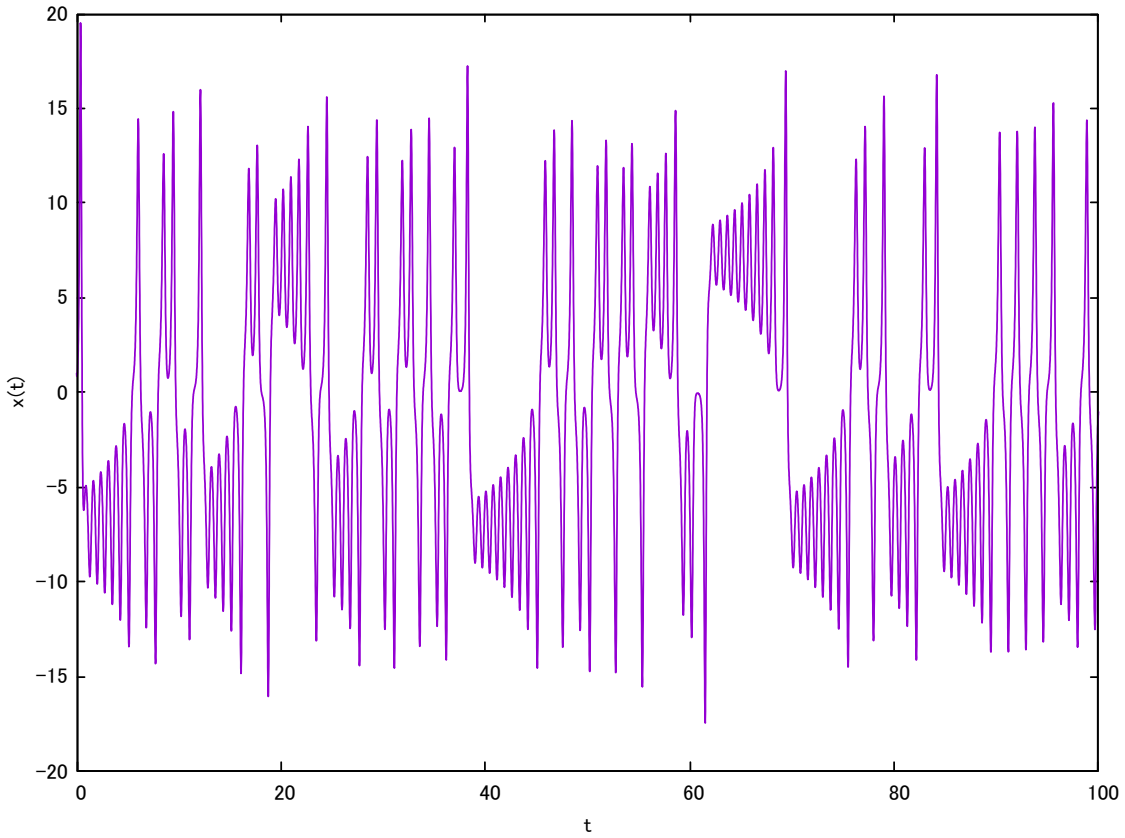


図 5: x の時間変化

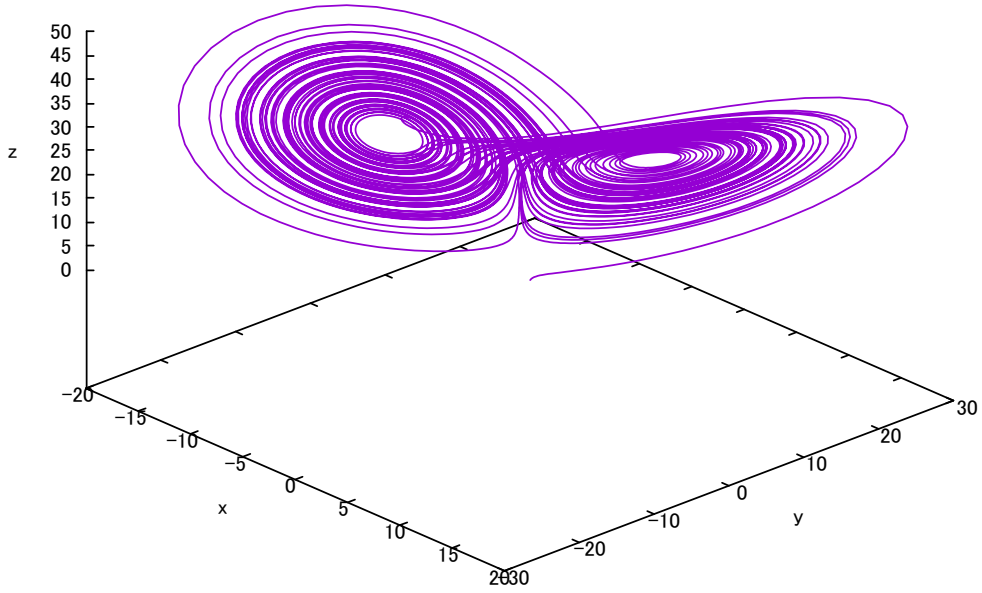


図 6: 点 (x, y, z) の軌道

図 6 から, 点 (x, y, z) は折れ曲がった 8 の字状の軌道を描くことがわかる.

5.2 初期値の変化による軌道の変化

n を整数とし,$\epsilon = 0$ での解と $\epsilon = 10^{-n}$ での解をそれぞれ

$$\boldsymbol{x}_0(t) = (x_0(t), y_0(t), z_0(t)) \tag{27}$$
$$\boldsymbol{x}_n(t) = (x_n(t), y_n(t), z_n(t)) \tag{28}$$

とする. また,$\boldsymbol{x}_0(t)$ と $\boldsymbol{x}_n(t)$ との距離を

$$\Delta_n(t) = \|\boldsymbol{x}_n(t) - \boldsymbol{x}_0(t)\| \tag{29}$$

で定める.$\|\cdot\|$ はユークリッド距離である. $n = 2, 4, 6$ について,$\Delta_n(t)$ を計算することで

- $\log_{10} \Delta_n(t)$ を t の関数としてプロットしたとき, 十分大きい t を取ると漸近値の近くで変動すること
- 漸近値に至る少し前の時間帯では $\Delta_n(t)$ の関数形と指数は n に依存しないこと

を確かめ,$\Delta_n(t)$ の関数形と指数を定める. 図 7 に計算結果を示す.

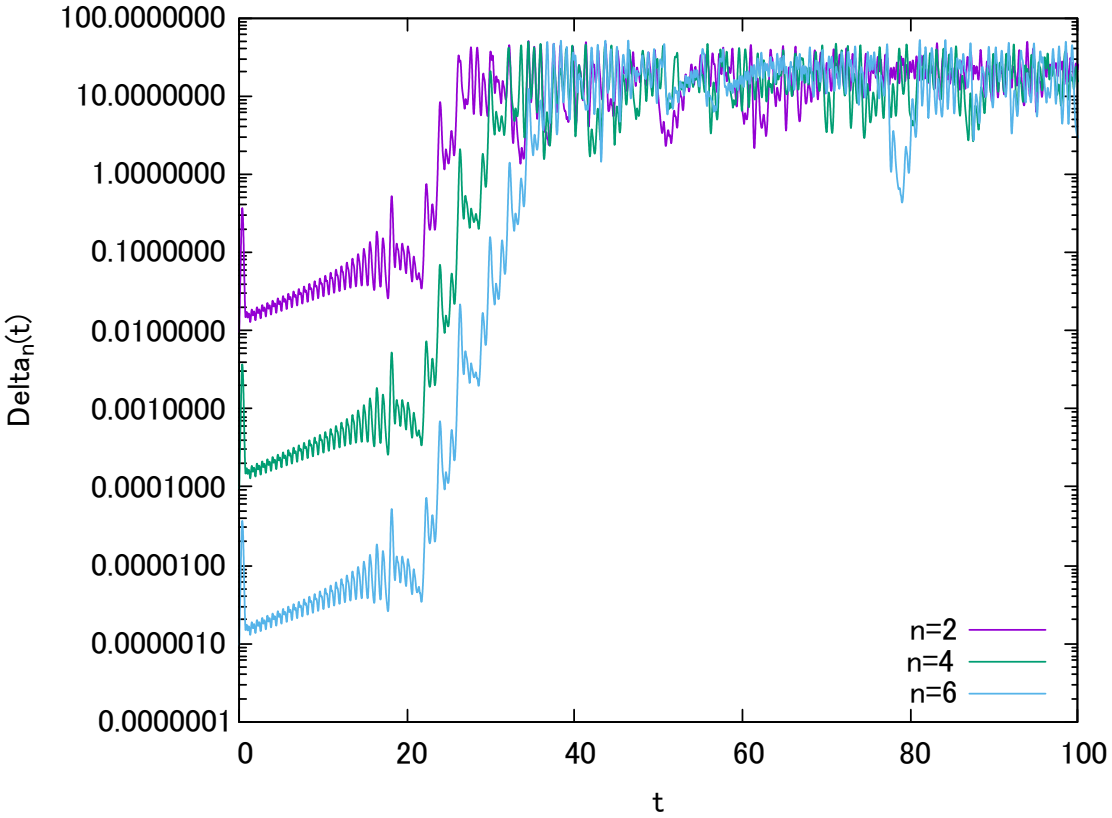


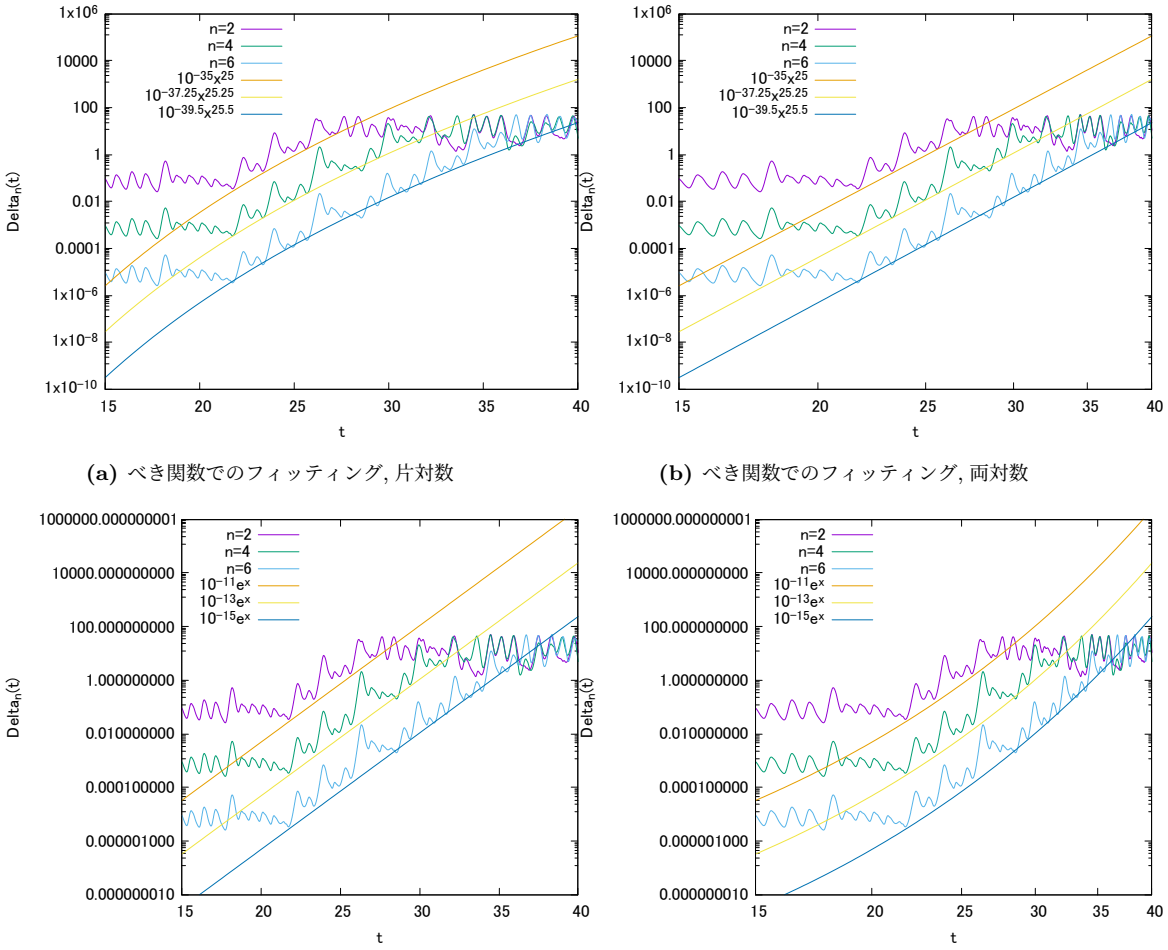
図 7: 漸近の様子

5.2.1 グラフの漸近

図 7 を見れば, 確かに十分大きい t において $\Delta_n(t)$ がある値の近くで変動している様子が見て取れる.

5.2.2 漸近直前の関数形

次に, 漸近する直前の $\Delta_n(t)$ の関数形を調べる. 図 8 にフィッティングの結果を示す. 見やすくするため, 表示する範囲を $t \in [15 : 40]$ にしている.



(a) べき関数でのフィッティング, 片対数

(b) べき関数でのフィッティング, 両対数

(c) 指数関数でのフィッティング, 片対数

(d) 指数関数でのフィッティング, 両対数

図 8: フィッティングの結果

指数関数では n の値に依らずグラフの傾きが一定だが, べき関数では n の値でグラフの傾きが変化している. したがって, 漸近直前の関数形は指数関数であり, 傾きに影響する指数は 1 である.

6 課題 8

次の微分方程式を考える.

$$\frac{dx_i}{dt} = \omega_i - \frac{K}{N}\sum_{j=1}^{N}\sin(x_i - x_j), \quad (i = 1, \cdots, N). \tag{30}$$

ただし,x_i は単位円上の位置を表す.K,ω_i は定数であり,ω_i は以下で与えられる.

$$\omega_i = \tan\left[\pi(\frac{i}{N+1}) - \frac{1}{2}\right], \quad (i = 1, \cdots, N). \tag{31}$$

また,x の初期値は以下で与えられる.

$$x_i(0) = y_i + 0.01 \sin y_i, \;\; y_i := \frac{2\pi}{N}(i-1), \;\; (i = 1, \cdots, N). \tag{32}$$

このとき, オーダーパラメータ $R(t)$ を以下で定める.

$$R = \sqrt{R_x^2 + R_y^2}, \;\; R_x = \frac{1}{N}\sum_{j=1}^{N} \cos x_j, \;\; R_y = \frac{1}{N}\sum_{j=1}^{N} \sin x_j. \tag{33}$$

6.1 計算量の削減

式 30 をそのままコーディングすると, 計算量は $O(N^2)$ になるが, 計算の手順を工夫することで $O(N)$ で計算可能である. 以下にその理由を示す. 式 30 の右辺第二項について, 加法定理を用いて変形する.

$$-\frac{K}{N}\sum_{j=1}^{N} \sin(x_i - x_j) = -\frac{K}{N}\sum_{j=1}^{N} (\sin x_i \cos x_j - \cos x_i \sin x_j) \tag{34}$$

$$= -K\sum_{j=1}^{N} (R_x \sin x_i - R_y \cos x_i). \tag{35}$$

したがって, 時刻 t ごとに予め R_x, R_y を計算しておくことで式 30 を $O(N)$ で計算できる.

6.2 オーダーパラメータと定数 K の関係

以下ではステップ幅 $\Delta t = 0.01$ の 4 次ルンゲ・クッタ法を用いる.$N = 10^2, 10^3$ に対して, 区間 $t \in [50, 100]$ における $R(t)$ の時間平均を $\overline{R}$ で表す.

$$\overline{R} = \frac{1}{50}\int_{50}^{100} R(t)dt. \tag{36}$$

これを $K \in [1, 3]$ について求める.K の刻み幅は 0.1 とする.
計算結果を図 9 に示す.

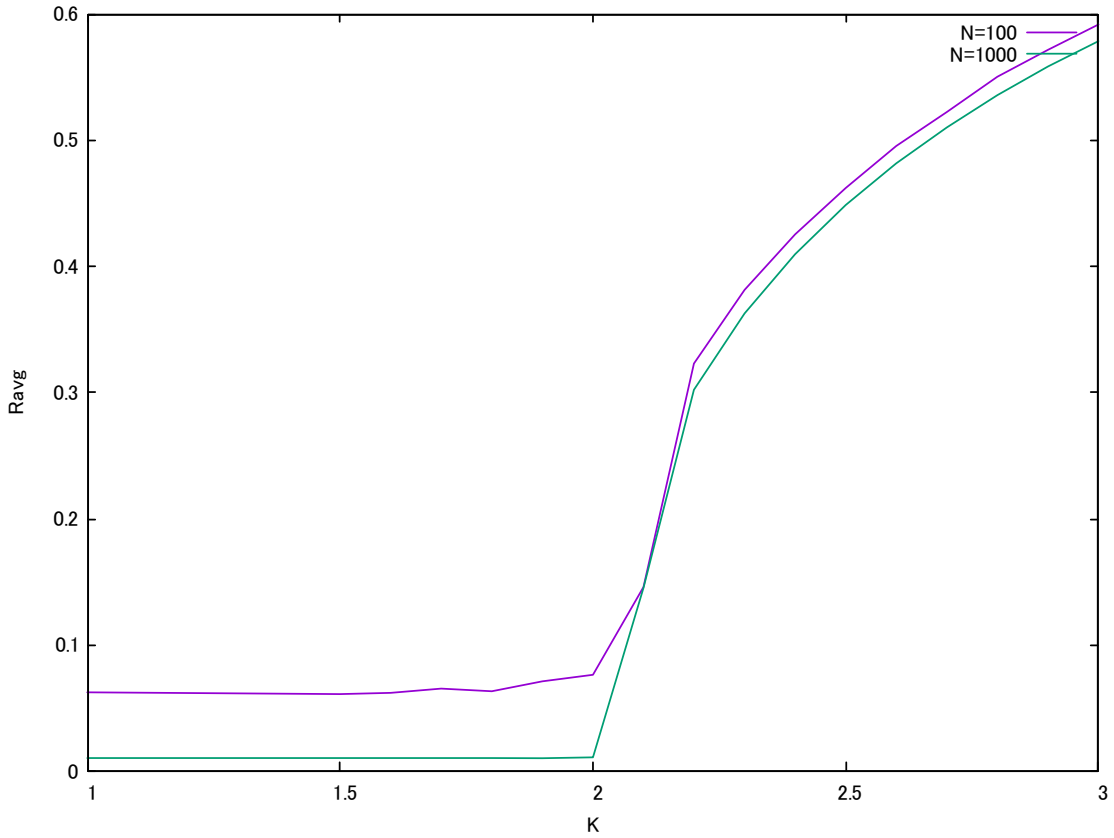


図 9: K に対する R の時間平均

6.3 臨界点

$N \to \infty$ としたとき,$K = 2$ は臨界点と呼ばれている. 以下ではその理由を議論する.

■オーダーパラメータ オーダーパラメータ R は各点の位相の揃い具合を示している. $R = 0$ のときは全体が無秩序に動いており,$R > 0$ であれば一定の秩序が存在している. このことから, オーダーパラメータの値を見ることでその時刻における点のまとまり具合を知ることができる. たとえば,$K < 2$ の部分に注目すると,$N = 100$ に比べて $N = 1000$ のほうが値が小さくなっている. これは, 点の数が多い分, 自然にできるまとまりが見られにくくなるためだと考えられる.

■臨界点 図 9 を確認すると,$N = 100, 1000$ のどちらにおいても,$K = 2$ 付近で 2 階微分係数が 0 から不連続に変化している. このことから, 式 30 で表されるモデルでは,N の値に依らず $K = 2$ で相転移が起こるのではないかと推測される.

■結論 以上のことから,$N \to \infty$ としたときに $K = 2$ が臨界点と呼ばれるのは,$N \to \infty$ のときに相転移が起こるのが $K = 2$ であるからだと考えられる.

7 付録

コード 1: 前進オイラー法

```
#include<iostream>
#include<cmath>
#include<vector>
#define imax 10

void euler(double dt){
    double t = 0;
    double step = 1/dt;
    double u0 = 1;
    double u_old = u0;
    double u_new;
    for (double i=0;i<step;i++){
        u_new = u_old + u_old*dt;
        u_old = u_new;
    }
    std::cout << fabs(exp(1) - u_new) << std::endl;
}

int main(void){
    std::vector<double> iset(imax);
    iset[0] = 1;
    for (int i=1; i<imax; i++){
        iset[i] = iset[i-1]/2;
    }
    for (int i=1;i<(int)iset.size();i++){
        std::cout << iset[i] << "␣";
        euler(iset[i]);
    }
    return 0;
}
```

コード 2: 2 次 AB 法

```
#include<iostream>
#include<cmath>
#include<vector>
#define imax 10

void ab2(double dt){
    double t = 0;
    double step = 1/dt;
    double u0 = 1;
    double u1 = exp(dt);
    double u_old_old = u0;
    double u_old = u1;
    double u_new;
    for (double i=0;i<step-1;i++){
        u_new = u_old + dt*(3*u_old-u_old_old)/2;
        u_old_old = u_old;
        u_old = u_new;
    }
```

```
    std::cout << fabs(exp(1) - u_new) << std::endl;
}

int main(void){
    std::vector<double> iset(imax);
    iset[0] = 1;
    for (int i=1; i<imax; i++){
        iset[i] = iset[i-1]/2;
    }

    for (int i=1;i<(int)iset.size();i++){
        std::cout << iset[i] << "␣";
        ab2(iset[i]);
    }
    return 0;
}
```

コード 3: 3 次 AB 法

```
#include<iostream>
#include<cmath>
#include<vector>
#define imax 10

void ab3(double dt){
    double t = 0;
    double step = 1/dt;
    double u0 = 1;
    double u1 = exp(dt);
    double u2 = exp(2*dt);
    double u_3old = u0;
    double u_2old = u1;
    double u_old = u2;
    double u_new = exp(1);
    for (double i=0;i<step-2;i++){
        u_new = u_old + dt*(23*u_old-16*u_2old+5*u_3old)/12;
        u_3old = u_2old;
        u_2old = u_old;
        u_old = u_new;
    }
    std::cout << fabs(exp(1) - u_new) << std::endl;
}

int main(void){
    std::vector<double> iset(imax);
    iset[0] = 1;
    for (int i=1; i<imax; i++){
        iset[i] = iset[i-1]/2;
    }

    for (int i=1;i<(int)iset.size();i++){
        std::cout << iset[i] << "␣";
        ab3(iset[i]);
    }
    return 0;
}
```

コード 4: 2 次 RK 法

```
#include<iostream>
#include<cmath>
#include<vector>
#define imax 10

void rk2(double dt){
    double t = 0;
    double step = 1/dt;
    double u0 = 1;
    double u_old = u0;
    double u_new = exp(1);
    for (double i=0;i<step;i++){
        u_new = u_old + dt*(u_old+u_old+dt*u_old)/2;
        u_old = u_new;
    }
    std::cout << fabs(exp(1) - u_new) << std::endl;
}

int main(void){
    std::vector<double> iset(imax);
    iset[0] = 1;
    for (int i=1; i<imax; i++){
        iset[i] = iset[i-1]/2;
    }

    for (int i=1;i<(int)iset.size();i++){
        std::cout << iset[i] << "␣";
        rk2(iset[i]);
    }
    return 0;
}
```

コード 5: 4 次 RK 法

```
#include<iostream>
#include<cmath>
#include<vector>
#define imax 10

void rk4(double dt){
    double t = 0;
    double step = 1/dt;
    double u0 = 1;
    double u_old = u0;
    double u_new;
    for (double i=0;i<step;i++){
        double f1 = u_old;
        double f2 = u_old + dt*f1/2;
        double f3 = u_old + dt*f2/2;
        double f4 = u_old + dt*f3;
        u_new = u_old + dt*(f1+2*f2+2*f3+f4)/6;
        u_old = u_new;
    }
    std::cout << fabs(exp(1) - u_new) << std::endl;
}

```

```
23 int main(void){
24     std::vector<double> iset(imax);
25     iset[0] = 1;
26     for (int i=1; i<imax; i++){
27         iset[i] = iset[i-1]/2;
28     }
29 
30     for (int i=1;i<(int)iset.size();i++){
31         std::cout << iset[i] << "␣";
32         rk4(iset[i]);
33     }
34     return 0;
35 }
```

コード 6: 課題 4 のコード

```
1  #include<iostream>
2  #include<cmath>
3  #include<vector>
4  #include<fstream>
5  
6  using namespace std;
7  
8  void cn(double dt, double start, double end, double alpha, double beta, double u0, string
       FILE_NAME){
9      double t = start;
10     double step = (end - start)/dt;
11     double u_old = u0;
12     double u_new = u_old;
13 
14     ofstream outputfile(FILE_NAME);
15 
16     outputfile << t << "␣" << u_new << endl;
17 
18     for (double i=0;i<step;i++){
19         t += dt;
20         u_new = (2 - alpha*dt)/(2 + alpha*dt)*u_old + 2*beta*dt/(2 + alpha*dt);
21         u_old = u_new;
22         outputfile << t << "␣" << u_new << endl;
23     }
24 }
25 
26 void rk2(double dt, double start, double end, double alpha, double beta, double u0, string
       FILE_NAME){
27     double t = start;
28     double step = (end - start)/dt;
29     double u_old = u0;
30     double u_new = u_old;
31     double u_star;
32     ofstream outputfile(FILE_NAME);
33 
34     outputfile << t << "␣" << u_new << endl;
35 
36     for (double i=0;i<step;i++){
37         t += dt;
38         u_star = u_old + (-alpha*u_old + beta)*dt;
39         u_new = u_old + (-alpha*u_old + beta - alpha*u_star + beta)*dt/2;
```

```
        u_old = u_new;
        outputfile << t << "␣" << u_new << endl;
    }
}

int main(void){
    cn(0.01,0,10,10,1,1,"cn_0.01.txt");
    cn(0.19,0,10,10,1,1,"cn_0.19.txt");
    cn(0.205,0,10,10,1,1,"cn_0.205.txt");
    rk2(0.01,0,10,10,1,1,"rk2_0.01.txt");
    rk2(0.19,0,10,10,1,1,"rk2_0.19.txt");
    rk2(0.205,0,10,10,1,1,"rk2_0.205.txt");
    return(0);
}
```

コード 7: 課題 5 のコード

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>

using namespace std;

void calc(double dt, string FILE_NAME){
    double t = 0;
    double u0 = 1;
    double u1 = 0.5*exp(-2*dt)+0.5;
    double step = 20/dt;
    double u_old_old = u0;
    double u_old = u1;
    double u_new = u0;

    ofstream outputfile(FILE_NAME);

    for (double i=0;i<step;i++){
        t += dt;
        u_new = u_old_old + 2*dt*(-2*u_old+1);
        u_old_old = u_old;
        u_old = u_new;
        outputfile << t << "␣" << u_new << endl;
    }
}

int main(void){
    calc(0.01,"0.01.txt");
    calc(0.05,"0.05.txt");
    calc(0.10,"0.10.txt");
    return(0);
}
```

コード 8: 課題 7 のコード 1

```
#include<iostream>
#include<cmath>
#include<vector>
#include<fstream>
```

```
5
6  const double sigma = 10;
7  const double b = 8/3;
8  const double r = 28;
9
10 using namespace std;
11
12 double fx(double x,double y, double z){
13     return sigma*(y - x);
14 }
15
16 double fy(double x, double y, double z){
17     return r*x - y - x*z;
18 }
19
20 double fz(double x, double y, double z){
21     return x*y - b*z;
22 }
23
24 void rk4(double x, double y, double z, string FILE_NAME){
25     ofstream outputfile(FILE_NAME);
26     double t = 0;
27     int step = 10000;
28     double dt = 0.01;
29     double f1x,f2x,f3x,f4x,f1y,f2y,f3y,f4y,f1z,f2z,f3z,f4z;
30
31     outputfile << t << "␣" << x << "␣" << y << "␣" << z << endl;
32
33     for (int i=0; i<step; i++){
34     t += dt;
35
36     f1x = fx(x, y, z);
37     f1y = fy(x, y, z);
38     f1z = fz(x, y, z);
39
40     f2x = fx(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
41     f2y = fy(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
42     f2z = fz(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
43
44     f3x = fx(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
45     f3y = fy(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
46     f3z = fz(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
47
48     f4x = fx(x + f3x*dt, y + f3y*dt, z + f3z*dt);
49     f4y = fy(x + f3x*dt, y + f3y*dt, z + f3z*dt);
50     f4z = fz(x + f3x*dt, y + f3y*dt, z + f3z*dt);
51
52     x += (f1x + 2*f2x + 2*f3x + f4x)*dt/6.0;
53     y += (f1y + 2*f2y + 2*f3y + f4y)*dt/6.0;
54     z += (f1z + 2*f2z + 2*f3z + f4z)*dt/6.0;
55
56     outputfile << t << "␣" << x << "␣" << y << "␣" << z << endl;
57     }
58 }
59
60 int main(void){
61     rk4(1,0,0,"1.txt");
```

```
62     return 0;
63 }
```

コード 9: 課題 7 のコード 2

```
1  #include<iostream>
2  #include<cmath>
3  #include<vector>
4  #include<fstream>
5
6  const double sigma = 10.0;
7  const double b = 8.0/3.0;
8  const double r = 28.0;
9
10 using namespace std;
11
12 double fx(double x,double y, double z){
13     return sigma*(y - x);
14 }
15
16 double fy(double x, double y, double z){
17     return r*x - y - x*z;
18 }
19
20 double fz(double x, double y, double z){
21     return x*y - b*z;
22 }
23
24 void rk4(double epsilon, double x, double y, double z, string FILE_NAME){
25     ofstream outputfile(FILE_NAME);
26     double t = 0;
27     int step = 10000;
28     double dt = 0.01;
29     double f1x,f2x,f3x,f4x,f1y,f2y,f3y,f4y,f1z,f2z,f3z,f4z;
30     double ef1x,ef2x,ef3x,ef4x,ef1y,ef2y,ef3y,ef4y,ef1z,ef2z,ef3z,ef4z;
31     double ex = x + epsilon;
32     double ey = y;
33     double ez = z;
34     double delta;
35
36     delta = sqrt((ex-x)*(ex-x) + (ey-y)*(ey-y) + (ez-z)*(ez-z));
37     outputfile << t << "␣" << delta << endl;
38
39     for (int i=0; i<step; i++){
40     t += dt;
41
42     f1x = fx(x, y, z);
43     f1y = fy(x, y, z);
44     f1z = fz(x, y, z);
45
46     f2x = fx(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
47     f2y = fy(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
48     f2z = fz(x + f1x*dt/2.0, y + f1y*dt/2.0, z + f1z*dt/2.0);
49
50     f3x = fx(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
51     f3y = fy(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
52     f3z = fz(x + f2x*dt/2.0, y + f2y*dt/2.0, z + f2z*dt/2.0);
```

```
53
54      f4x = fx(x + f3x*dt, y + f3y*dt, z + f3z*dt);
55      f4y = fy(x + f3x*dt, y + f3y*dt, z + f3z*dt);
56      f4z = fz(x + f3x*dt, y + f3y*dt, z + f3z*dt);
57
58      x += (f1x + 2.0*f2x + 2.0*f3x + f4x)*dt/6.0;
59      y += (f1y + 2.0*f2y + 2.0*f3y + f4y)*dt/6.0;
60      z += (f1z + 2.0*f2z + 2.0*f3z + f4z)*dt/6.0;
61
62
63      ef1x = fx(ex, ey, ez);
64      ef1y = fy(ex, ey, ez);
65      ef1z = fz(ex, ey, ez);
66
67      ef2x = fx(ex + ef1x*dt/2.0, ey + ef1y*dt/2.0, ez + ef1z*dt/2.0);
68      ef2y = fy(ex + ef1x*dt/2.0, ey + ef1y*dt/2.0, ez + ef1z*dt/2.0);
69      ef2z = fz(ex + ef1x*dt/2.0, ey + ef1y*dt/2.0, ez + ef1z*dt/2.0);
70
71      ef3x = fx(ex + ef2x*dt/2.0, ey + ef2y*dt/2.0, ez + ef2z*dt/2.0);
72      ef3y = fy(ex + ef2x*dt/2.0, ey + ef2y*dt/2.0, ez + ef2z*dt/2.0);
73      ef3z = fz(ex + ef2x*dt/2.0, ey + ef2y*dt/2.0, ez + ef2z*dt/2.0);
74
75      ef4x = fx(ex + ef3x*dt, ey + ef3y*dt, ez + ef3z*dt);
76      ef4y = fy(ex + ef3x*dt, ey + ef3y*dt, ez + ef3z*dt);
77      ef4z = fz(ex + ef3x*dt, ey + ef3y*dt, ez + ef3z*dt);
78
79
80      ex += (ef1x + 2.0*ef2x + 2.0*ef3x + ef4x)*dt/6.0;
81      ey += (ef1y + 2.0*ef2y + 2.0*ef3y + ef4y)*dt/6.0;
82      ez += (ef1z + 2.0*ef2z + 2.0*ef3z + ef4z)*dt/6.0;
83
84
85      delta = sqrt((ex-x)*(ex-x) + (ey-y)*(ey-y) + (ez-z)*(ez-z));
86      outputfile << t << "␣" << delta << endl;
87      }
88  }
89
90  int main(void){
91      rk4(0.01,1.0,0,0,"n2.txt");
92      rk4(0.0001,1.0,0,0,"n4.txt");
93      rk4(0.000001,1.0,0,0,"n6.txt");
94      return 0;
95  }
```

コード 10: 課題 8 のコード

```
1  #include<iostream>
2  #include<cmath>
3  #include<vector>
4  #include<fstream>
5
6  #define dt 0.01
7
8  using namespace std;
9
10 double omega(int i, int N){
11     return tan(M_PI*((double)i/(N+1)-0.5));
```

```
12 }
13
14 vector<double> f1(vector<double> x, double K, int N){
15     double Rx, Ry;
16     vector<double> g(N+1);
17     Rx = Ry = 0;
18
19     for (int i=1;i<N+1;i++){
20     Rx += cos(x[i])/N;
21     Ry += sin(x[i])/N;
22     }
23     for (int i=1; i<N+1; i++){
24         g[i] = omega(i, N) - K* (sin(x[i])*Rx - cos(x[i])*Ry);
25     }
26     return g;
27 }
28
29 vector<double> f2(vector<double> x, vector<double>k1, double K, int N){
30     double Rx, Ry;
31     vector<double> g(N+1), h(N+1);
32     Rx = Ry = 0;
33     for (int i=1;i<N+1;i++){
34         g[i] = x[i] + k1[i]*dt/2.0;
35         Rx += cos(g[i])/N;
36         Ry += sin(g[i])/N;
37     }
38     for (int i=1; i<N+1; i++){
39         h[i] = omega(i,N) - K* (sin(g[i])*Rx - cos(g[i])*Ry);
40     }
41     return h;
42 }
43
44 vector<double> f3(vector<double> x, vector<double>k2, double K, int N){
45     double Rx, Ry;
46     vector<double> g(N+1), h(N+1);
47     Rx = Ry = 0;
48     for (int i=1;i<N+1;i++){
49         g[i] = x[i] + k2[i]*dt/2.0;
50         Rx += cos(g[i])/N;
51         Ry += sin(g[i])/N;
52     }
53     for (int i=1; i<N+1; i++){
54         h[i] = omega(i,N) - K* (sin(g[i])*Rx - cos(g[i])*Ry);
55     }
56     return h;
57 }
58
59 vector<double> f4(vector<double> x, vector<double>k3, double K, int N){
60     double Rx, Ry;
61     vector<double> g(N+1), h(N+1);
62     Rx = Ry = 0;
63     for (int i=1;i<N+1;i++){
64         g[i] = x[i] + k3[i]*dt;
65         Rx += cos(g[i])/N;
66         Ry += sin(g[i])/N;
67     }
68     for (int i=1; i<N+1; i++){
```

```
69          h[i] = omega(i,N) - K* (sin(g[i])*Rx - cos(g[i])*Ry);
70      }
71      return h;
72  }
73
74  double rk4(vector<double> x, int N, double K){
75      double t, Rx, Ry, Rsum, Ravg;
76      vector<double> k1, k2, k3, k4;
77      Rsum = 0;
78
79
80      for (t=0; t<=100; t+=dt){
81          k1 = f1(x, K, N);
82          k2 = f2(x, k1, K, N);
83          k3 = f3(x, k2, K, N);
84          k4 = f4(x, k3, K, N);
85
86          Rx = Ry = 0;
87          for (int i=1;i<N+1;i++){
88          x[i] += dt*(k1[i] + 2*k2[i] + 2*k3[i] + k4[i])/6.0;
89          Rx += cos(x[i])/N;
90          Ry += sin(x[i])/N;
91          }
92          if (t >= 50){
93          Rsum += sqrt(Rx*Rx + Ry*Ry) * dt;
94          }
95      }
96      Ravg = Rsum/50.0;
97      return Ravg;
98  }
99
100 void calc(int N, string FILE_NAME){
101     ofstream outputfile(FILE_NAME);
102
103     vector<double> x(N+1), y(N+1);
104
105     for (int i=1; i<N+1; i++){
106         y[i] = 2*M_PI*(i-1)/N;
107         x[i] = y[i] + 0.01*sin(y[i]);
108     }
109
110     for (double K=1; K<=3.01; K+=0.1){
111     double res = rk4(x,N,K);
112     outputfile << K << "␣" << res << endl;
113     }
114 }
115
116 int main(void){
117     int N = 100;
118     int M = 1000;
119     calc(N, "kadai8_100.txt");
120     calc(M, "kadai8_1000.txt");
121     return 0;
122 }
```

参考文献

[1] 実験演習ワーキンググループ (2022),「数理工学実験テキスト_2022 年版」