VII. 常微分方程式と高精度解法

１６．ローレンツのカオス

現象を記述する方程式が決定論的で計算可能な場合でも未来が予測できないことがある。このことを最初に示したのがローレンツによるカオスのモデルである。このような現象は非線型な場合に起こりうる。

16.1 熱対流とローレンツの方程式

エドワード・ローレンツはカオスを発見した気象学者として知られる。ローレンツは大気の対流運動の単純化したモデルを考え, 対流の時間変動が次の３つの式で表されるとした。

$$\frac{dX(t)}{dt} = -\sigma X(t) + \sigma Y(t) \tag{16.1}$$

$$\frac{dY(t)}{dt} = -X(t)Z(t) + \gamma X(t) - Y(t) \tag{16.2}$$

$$\frac{dZ(t)}{dt} = -X(t)Y(t) - bZ(t) \tag{16.3}$$

これらの式は対流の速度や温度をフーリエ級数で表し, 低次のモードだけを考えることにより得られる。ここで,

$X(t)$: 流線関数の基本モード

$Y(t)$: 温度の基本モード

$Z(t)$: 温度の第 1 次高調波モード

σ: プラントル数(大きいほど粘性の影響が強い)

γ: レイリー数(大きいほど浮力が強い)

b: 対流の水平波長

に相当すると考えることができる。

$$X(\infty) = Y(\infty) = Z(\infty) = 0 \tag{16.4}$$

$$(X_{\pm}, Y_{\pm}, Z_{\pm}) = (\pm C, \pm C, \gamma - 1) \tag{16.5}$$

$$C = \sqrt{b(\gamma - 1)} \tag{16.6}$$

ローレンツの式はγによって次のような3種類の解を持つ。

(1) $\gamma < 1$ のとき: 対流が停止する

$$X(\infty) = Y(\infty) = Z(\infty) = 0 \tag{16.7}$$

(2) $1 < \gamma < \gamma_T$ (=24.7368)のとき: 対流が定常状態となる。

$$(X_{\pm}, Y_{\pm}, Z_{\pm}) = (\pm C, \pm C, \gamma - 1) \tag{16.8}$$

$$C = \sqrt{b(\gamma - 1)} \tag{16.9}$$

(3) $\gamma > \gamma_T$ のとき: 対流がカオスになる。解は定常解の周りで振動する。

このとき，定常解をアトラクターと呼ぶ。

ここでは，２つのことに注目する。まず，γの違いにより解が異なった様相をしめすことが数値計算で得られるのかという点である。もう1つは初期値をわずかに変化させるとどのようになるかという点である。

１７．常微分方程式の数値解法

ローレンツの式は非線型であるため, 定常状態のような特別の場合を除いて解析的に解くことができない。そのため，初期条件を与えて，時間発展問題としてコンピュータを使用して数値的に解くことになる。

17.1 オイラー法による解法

偏微分方程式の時と同様，３つの常微分方程式の微分を前進差分

$$\frac{df}{dt} = \frac{f^{n+1} - f^n}{\Delta t} \tag{17.1}$$

で置き換える。ここで，n はタイムステップ番号である。すなわち，

$$\frac{X^{n+1} - X^n}{\Delta t} = -\sigma X^n + \sigma Y^n \tag{17.2}$$

$$\frac{Y^{n+1} - Y^n}{\Delta t} = -X^n Z^n + \gamma X^n - Y^n \tag{17.3}$$

$$\frac{Z^{n+1} - Y^n}{\Delta t} = -X^n Y^n + bZ^n \tag{17.4}$$

である。ベクトル式で書くと，

$$\frac{\boldsymbol{X}^{n+1}-\boldsymbol{X}^{n}}{\Delta t}=\boldsymbol{F}\left(\boldsymbol{X}^{n}\right) \tag{17.5}$$

と表すことができる。ただし，

$$\boldsymbol{X}^{n}=\begin{bmatrix} X^{n} \\ Y^{n} \\ Z^{n} \end{bmatrix} \tag{17.6}$$

$$\boldsymbol{F}\left(\boldsymbol{X}^{n}\right)=\begin{bmatrix} -\sigma X^{n}+\sigma Y^{n} \\ -X^{n}Z^{n}+\gamma X^{n}-Y^{n} \\ -X^{n}Y^{n}+bZ^{n} \end{bmatrix} \tag{17.7}$$

である。この時間微分は1次の前進差分であるから，この方法はオイラー法である。ローレンツの式は1階常微分方程式であるから，初期における X, Y, Z の値を与えることにより，時間進行により解くことができる。

17.2 予測子・修正子法：2次ルンゲ・クッタ法

オイラー法は1次精度と打ち切り誤差が大きいので，より高次の打ち切り誤差を持つ方法が望ましい。ところで(16.1)〜(16.3)の右辺は非線型であり，陰解法を使用しても連立1次方程式とはならない。このため，陰解法で解くには反復法が必要になってしまい，計算の効率が悪い。そこで，未来の予測値を作ってそこから時間微分係数の値を計算し直し，これを用いて時間を進める方法が用いられる。つまり，行きつ戻りつ計算するのである。このような方法は**予測子・修正子法** (predictor-corrector method)とよばれる。

予測子ステップはオイラー法で計算する。導関数

$$\boldsymbol{k}_1=\boldsymbol{F}\left(\boldsymbol{X}^{n}\right) \tag{17.11}$$

より，未来の予測値

$$\tilde{\boldsymbol{X}}^{n+1}=\boldsymbol{X}^{n}+\Delta t\boldsymbol{k}_1 \tag{17.12}$$

を計算する。このあと，これを予測子として，未来の値から計算される勾配

$$\boldsymbol{k}_2=\boldsymbol{F}\left(\tilde{\boldsymbol{X}}^{n+1}\right) \tag{17.13}$$

を計算した後，未来の値は，修正子ステップ

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^{n} + \frac{\Delta t}{2}\left(\boldsymbol{k}_1 + \boldsymbol{k}_2\right) \tag{17.14}$$

から計算される。すなわち，

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^{n} + \frac{\Delta t}{2}\left[\boldsymbol{F}\left(\boldsymbol{X}^{n}\right) + \boldsymbol{F}\left(\tilde{\boldsymbol{X}}^{n+1}\right)\right] \tag{17.15}$$

と計算したことになる。つまり，クランク・ニコルソン法の陰的な項の代わりに予測子から計算した導関数を用いていることになる。この方法はクランク・ニコルソン法と同様，２次の打ち切り誤差を持つ。

17.3 ４次ルンゲ・クッタ法

１つのタイムステップの途中も使って修正を加えるとより高次の打ち切り誤差を持つ方法を作ることができる。このような方法を**ルンゲ・クッタ法** (Runge-Kutta method) とよぶ。予測子・修正子法は２次の精度を持つルンゲ・クッタ法ということができる。連立常微分方程式を解くのに最もよく用いられる４次ルンゲ・クッタ法では，途中，現在のステップの導関数を１回，中間ステップの導関数を２回，未来のステップの導関数を１回用いる。現在ステップの導関数

$$\boldsymbol{k}_1 = \boldsymbol{F}\left(\boldsymbol{X}^{n}\right) \tag{17.21}$$

より中間ステップの予測子を求める。

$$\tilde{\boldsymbol{X}}^{n+1/2} = \boldsymbol{X}^{n} + \frac{\Delta t}{2}\boldsymbol{k}_1 \tag{17.22}$$

これを用いて，さらに中間ステップの導関数を計算し，

$$\boldsymbol{k}_2 = \boldsymbol{F}\left(\tilde{\boldsymbol{X}}^{n+1/2}\right) \tag{17.23}$$

$$\hat{\boldsymbol{X}}^{n+1/2} = \boldsymbol{X}^{n} + \frac{\Delta t}{2}\boldsymbol{k}_2 \tag{17.24}$$

のように中間ステップの値を計算し直す。これから未来のステップの予測子を求める。

$$\boldsymbol{k}_3 = \boldsymbol{F}\left(\hat{\boldsymbol{X}}^{n+1/2}\right) \tag{17.25}$$

$$\widehat{\boldsymbol{X}}^{n+1} = \boldsymbol{X}^{n} + \Delta t\boldsymbol{k}_3 \tag{17.26}$$

未来ステップの予測子から，未来側の導関数

$$\boldsymbol{k}_4 = \boldsymbol{F}\left(\widehat{\boldsymbol{X}}^{n+1}\right) \tag{17.27}$$

を求める。導関数に重みをつけて平均し，未来の値を下式のように計算する。

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \frac{\Delta t}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right) \tag{17.28}$$

ここで，導関数の重みの割合 $1, 4\,(=2+2), 1$ がシンプソン法と同じになっている。ここでは，ルンゲ・クッタ法を利用して，ローレンツの式を解くことにする。

問題 VII-1

(1) ローレンツの式を解くプログラムをコンパイルして実行せよ。

(2) γの値を変えると，上記のような 3 種類の解が得られることを示せ。

(3) カオスとなるような条件のとき，初期値を極端に大きくしたらどのような解が得られるか調べよ。

(4) カオスとなるような条件のとき，初期値を 0.01 程度変化させて実行するとどのようになるか調べよ。

(5) (3)(4)はどのような意味を持つか考察せよ。

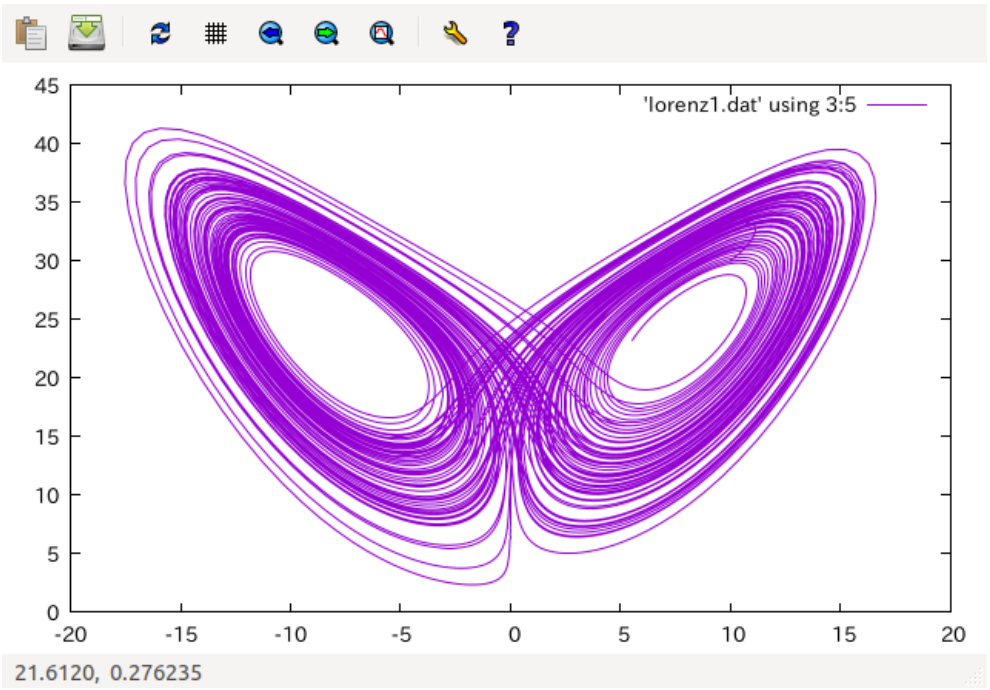

図 17.1　$X(t)$と $Z(t)$をプロット

問題 7-1　ローレンツ方程式のルンゲ・クッタ法による解法プログラム

ローレンツ方程式をルンゲ・クッタ法により解くプログラムである。プログラムはメインルーチン lo_main.f90, サブルーチン lo_cal_RHS.f90, モジュールプログラム lo_mod_para_l.f90 の 3 つからなる。パラメータの設定はデータファイル lo_para.dat にある。コンパイルはモジュールファイルを先に作っておく必要があるので,

```
f95 −c lo_mod_para_l.f90
f95 −c lo_main.f90 lo_cal_RHS.f90
f95 −o lo.out lo*.o
```

のように行うこと。プログラムは

https://home.hiroshima-u.ac.jp/nakakuki/Lectures/enshu/lorenz_rk.tar

からダウンロード可能である。

メインプログラム　lo_main.f

```
!
!  ****  Lorenz Chaos model  ****
!
   program lorenzrk

      use mod_para_l

      implicit none

      character(60):: apara
      real(8):: dt,   t
      integer:: nt,   it,   nft,  i

      real(8),allocatable:: X(:), Y(:), Z(:)

      real(8):: X_o(3), X_m(3), X_f(3)
      real(8):: k1(3),  k2(3),  k3(3),  k4(3)
```

```
      open ( 10, file='lo_para.dat', status='old' )

!  ****  read parameters  ****

      read ( 10,* )
      read ( 10,* ) apara
      read ( 10,* ) sig
      write ( 6,* ) apara
      write ( 6,* ) sig

      read ( 10,* ) apara
      read ( 10,* ) gamm
      write ( 6,* ) apara
      write ( 6,* ) gamm

      read ( 10,* ) apara
      read ( 10,* ) b
      write ( 6,* ) apara
      write ( 6,* ) b

      read ( 10,* ) apara
      read ( 10,* ) dt
      write ( 6,* ) apara
      write ( 6,* ) dt

      read ( 10,* ) apara
      read ( 10,* ) nt
      write ( 6,* ) apara
      write ( 6,* ) nt

      read ( 10,* ) apara
      read ( 10,* ) nft
      write ( 6,* ) apara
```

```
      write ( 6,* ) nft

!  ****  Set dimension of X, Y, Z  ****

      allocate ( X(0:nt),Y(0:nt),Z(0:nt) )

!  ****  Inital condition  ****

      it = 0
      t  = 0.0d0

      read ( 10,* ) apara
      read ( 10,* ) X(0)
      write ( 6,* ) apara
      write ( 6,* ) X(0)

      read ( 10,* ) apara
      read ( 10,* ) Y(0)
      write ( 6,* ) apara
      write ( 6,* ) Y(0)

      read ( 10,* ) apara
      read ( 10,* ) Z(0)
      write ( 6,* ) apara
      write ( 6,* ) Z(0)

!  ****  open output file and write inital condition  ****

      open ( 20, file='lorenz1.dat' )

      write ( 20,* ) it, t, X(0), Y(0), Z(0)

!  ****  Time marching loop  ***************************************************
```

```
      do it = 1,nt
        i = it - 1
        t = dfloat(it) * dt

!   Runge-Kutta method

        X_o(1) = X(i)
        X_o(2) = Y(i)
        X_o(3) = Z(i)

        call cal_RHS( X_o, k1 )
        X_m(:) = X_o(:) + 0.5d0 * dt * k1(:)

        call  cal_RHS( X_m, k2 )
        X_m(:) = X_o(:) + 0.5d0 * dt * k2(:)

        call  cal_RHS( X_m, k3 )
        X_f(:) = X_o(:) + dt * k3(:)

        call  cal_RHS( X_f, k4 )
        X_f(:) = X_o(:) + dt / 6.0d0 * ( k1(:) + 2.0d0*k2(:) + 2.0d0*k3(:) + k4(:) )

        X(i+1) = X_f(1)
        Y(i+1) = X_f(2)
        Z(i+1) = X_f(3)

!  ****  write results in each nft steps  ****

        if ( mod(it,nft) == 0 ) write (20,*) it, t, X(it), Y(it), Z(it)

      end do
```

```
!  ****  End of Time marching loop  ************************************************

      close (10)
      close (20)

      stop

    end program lorenzrk
```

ローレンツ方程式をルンゲ・クッタ法により解くプログラムで，σやγなどの物理パラメータの宣言をモジュール，導関数，すなわち，右辺の計算をサブルーチンとしている。
use mod_para_l：モジュール mod_para_l を呼び出す。モジュール内で宣言された変数はグローバル変数として扱われる。モジュールをコンパイルすると lo_mod_para_1.mod というファイルと lo_mod_para_1.o の両方ができる。lo_mod_para_1.mod ファイルは他の Fortran90 ファイルをコンパイルするときに必要になるので，モジュールプログラムは先にコンパイルしなければならない。lo_mod_para_1.o は実行形式ファイルを作るときに使用される。

サブルーチンプログラム　lo_cal_RHS.f

```
!
!    calculate RHS of Lorenz Equation
!
   subroutine cal_RHS( X, k )

      use mod_para_l

      implicit none

      real(8):: X(3), k(3)

!  Calculate RHS

      k(1) = -sig * X(1) + sig * X(2)
```

```
      k(2) = -X(1) * X(3) + gamm * X(1) - X(2)
      k(3) = X(1) * X(2) - b * X(3)

      return

   end subroutine cal_RHS
```

ローレンツ方程式の右辺を計算する。モジュールが use により呼び出されている。

モジュールプログラム　lo_mod_para_l.f

```
!
!   Definition of parameters
!
   module mod_para_l

      implicit none
      real(8):: sig,  gamm, b

   end module mod_para_l
```

共通の物理パラメータを格納するを変数を宣言している。モジュールで宣言された変数はグローバル変数として扱われる。サブルーチンが多く，サブルーチン間にまたがって使用される変数が多数あるときには便利な機能である。

パラメータファイル　lo_para.dat

```
# ****  Parameters for Lorenz model calculation  ****
'# (1) Sigma (Prandtl number) '
10.0d0
'# (2) Gamma (Rayleigh number): critical value = 24.7368 '
25.0d0
'# (3) b (wave length) '
```

```
2.666666666666666667d0
'# (4) dt (Time step increment) '
2.0d-3
'# (5) nt (Number of time steps) '
50000
'# (6) nft (Time-step interval writing results) '
10
'# (7) X(0) (initial value of X) '
10.0d0
'# (8) Y(0) (initial value of Y) '
15.0d0
'# (9) Z(0) (initial value of Z) '
30.0d0
```

物理パラメータの値や初期値などを与える。γの値や初期値を変化させたいときには，値を変更する。このファイルの値を変更した場合でも再コンパイルは必要ない。