数値流体ゼミ（第 1 回・後半戦）

染矢真好

2021 年 4 月 12 日

このゼミで扱うのは偏微分方程式（以下，PDE）の『差分解法』である。物理で出てくる微分方程式では変数が連続値をとるが，コンピュータは離散的な値しか取れない。そこで，前半戦で見たように微分を差分で近似し，段階的に解を求めていくという手法が用いられる。これが微分方程式の差分解法である。しかしいきなり PDE をやるのもあれなので，少しだけ常微分方程式（以下，ODE）の差分解法に触れておくことにする。

目次

1 オイラー法 1

2 色々な差分 2

3 ODE の差分解法の実装 3
- 3.1 （前進）オイラー法の場合 3
- 3.2 後退オイラー法の場合 4
- 3.3 ホイン法の場合 4
- 3.4 ルンゲ・クッタ法の場合 4

4 差分の誤差 5

1 オイラー法

次のように，区間 $a \leqq t \leqq b$ で定義された 1 階の ODE を考える（簡単のため，微分可能性などは必要なだけ仮定する）。

$$\frac{dy}{dt} = f(t, y(t)), \quad y(a) = y_0$$

区間を N 等分し，刻み幅を $h = \dfrac{b-a}{N}$ とする。そして，n（$0 \leqq n \leqq N$）番目の分点 $t_n = a + nh$ における値 y_n の数値解 Y_n を求める方法を考えよう。$t = t_n$ 近傍で，左辺の微分を次のように近似する。

$$\frac{y_{n+1} - y_n}{h} \simeq f(t_n, y_n) \qquad \therefore \quad y_{n+1} \simeq y_n + hf(t_n, y_n)$$

これは，$t = t_n$ における接線を伸ばして $t = t_{n+1}$ における値 y_{n+1} を推定する方法であり，オイラー法（前進オイラー法，陽的オイラー法）という。（前進）オイラー法のスキームは次のようになる。

（前進）オイラー法

$$\begin{cases} Y_0 = y_0 \\ Y_{n+1} = Y_n + hf(t_n, Y_n) \end{cases}$$

2 色々な差分

よく考えると，微分 $\dfrac{dy}{dt}$ の定義は 1 通りではなく，複数考えることができる。次の定義はいずれも同じことである。

$$\frac{dy}{dt} = \lim_{h\to 0}\frac{y(t+h)-y(t)}{h}$$
$$\frac{dy}{dt} = \lim_{h\to 0}\frac{y(t)-y(t-h)}{h}$$
$$\frac{dy}{dt} = \lim_{h\to 0}\frac{y(t+h)-y(t-h)}{2h}$$

これを差分で近似した場合，当然のことながらそれぞれ異なった結果を与える。

$$\frac{dy}{dt} \simeq \frac{y(t+h)-y(t)}{h} = \frac{y_{n+1}-y_n}{h}$$
$$\frac{dy}{dt} \simeq \frac{y(t)-y(t-h)}{h} = \frac{y_n-y_{n-1}}{h}$$
$$\frac{dy}{dt} \simeq \frac{y(t+h)-y(t-h)}{2h} = \frac{y_{n+1}-y_{n-1}}{2h}$$

これらは順に，前進差分（forward difference），後退差分（backward difference），中心差分（central difference）とよばれている。

（前進）オイラー法は明らかに前進差分の考え方に基づいている。では後退差分で作るとどうなるだろうか？

$$\frac{y_n-y_{n-1}}{h} \simeq f(t_n, y_n)$$

添字を 1 個ずらして

$$\frac{y_{n+1}-y_n}{h} \simeq f(t_{n+1}, y_{n+1}) \qquad \therefore \quad y_{n+1} \simeq y_n + hf(t_{n+1}, y_{n+1})$$

つまり

$$Y_{n+1} = Y_n + hf(t_{n+1}, Y_{n+1})$$

となり，Y_{n+1} について陽には表せなくなる。このような解法は陰的解法と呼ばれ，特にこのスキームは陰的オイラー法（または後退オイラー法）と呼ばれる（陽に表せる場合は陽的解法であるという。前進オイラー法は陽的である）。

（後退／陰的）オイラー法

$$Y_{n+1} = Y_n + hf(t_{n+1}, Y_{n+1})$$

ここまできたら，中心差分でもやってみたくなる。

$$\frac{y_{n+1}-y_{n-1}}{2h} \simeq f(t_n, y_n) \qquad \therefore \quad y_{n+1} \simeq y_{n-1} + 2hf(t_n, y_n)$$

よって

$$Y_{n+1} \simeq Y_{n-1} + 2hf(t_n, Y_n)$$

これまでは 1 つ前の情報があればよかったが，この場合は 2 つ前の情報も必要になる。これを 2 段階法という[*1]。2 段階法の場合，Y_1 だけは別の方法（例えば普通のオイラー法）で求めておく必要がある。

最後に，ODE の 1 段階法のうち代表的なホイン法と（古典的）ルンゲ・クッタ法を紹介する。

[*1] なおオイラー法は 1 段階法である。

ホイン法（修正オイラー法）

$$Y_{n+1} = Y_n + \frac{h}{2}\left[f(t_n, Y_n) + f(t_{n+1}, Y_n + hf(t_n, Y_n))\right]$$

点 (t_n, Y_n) における傾きと，点 (t_n, Y_n) を起点としてオイラー法で決めた点 $(t_{n+1}, Y_n + hf(t_n, Y_n))$ における傾きの平均値を取り，その傾きを持つ直線を伸ばして次の点 (t_{n+1}, Y_{n+1}) を計算する。

ルンゲ・クッタ法

$$\begin{cases} Y_{n+1} = Y_n + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right] \\ k_1 = f(t_n, Y_n) \\ k_2 = f\left(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_1\right) \\ k_3 = f\left(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_2\right) \\ k_4 = f\left(t_n + h, Y_n + hk_3\right) \end{cases}$$

3　ODE の差分解法の実装

これまで出てきた

- （前進）オイラー法
- 後退オイラー法
- ホイン法
- ルンゲ・クッタ法

を用いて，次の初期値問題を解かせる Python のコードを書いてみよう。

例題

$t \in [0, 1]$ で定義された ODE

$$\frac{dy}{dt} = y + t, \quad y(0) = 0$$

を解き，$0 \leqq t \leqq 1$ における $Y(t)$ のグラフを描く。また解析解が $y = e^t - t - 1$ であるので，$y(1) = e - 2$ となるはずだが，各スキームでどれくらい誤差（相対誤差）が出ているのかを調べる。

なお刻み幅は適当に $dt = 1/100 = 0.01$ とでもしておく（さっきまで h だったのにいきなり dt になっているが特に深い意味はない）。

3.1　（前進）オイラー法の場合

相対誤差は $e = 1.88 \times 10^{-2}$ となった。

3.2 後退オイラー法の場合

$Y_{n+1} = Y_n + dt(t_{n+1} + Y_{n+1})$ を Y_{n+1} について解くと

$$Y_{n+1} = \frac{Y_n + t_{n+1}}{1 - dt}$$

となる。

相対誤差は $e = 1.91 \times 10^{-2}$ となった。

3.3 ホイン法の場合

相対誤差は $e = 6.26 \times 10^{-5}$ となった。

3.4 ルンゲ・クッタ法の場合

相対誤差は $e = 3.12 \times 10^{-10}$ となった。

最後にコードの例を示す。

```
import numpy as np
import matplotlib.pyplot as plt
import japanize_matplotlib
%matplotlib inline

N = 100
dt = 1/N
y_0 = 0


def func(t,y):
    return t+y


t = 0
y = y_0

t_list = [0]
y_list = [y_0]

for i in range(N):
    # オイラー法の場合

    y += func(t,y)*dt
    t += dt


    # 後退オイラー法の場合
    """
    t += dt
    y = (y+t*dt)/(1-dt)
    """

    # ホイン法の場合
    """
    y += ( func(t,y) + func(t+dt, y+dt*func(t,y)) )*dt/2
    t += dt
    """

    # ルンゲ・クッタ法の場合
    """
    k1 = func(t,y)
    k2 = func(t+dt/2, y+k1*dt/2)
    k3 = func(t+dt/2, y+k2*dt/2)
    k4 = func(t+dt, y+k3*dt)
```

```
    y += (k1 + 2*k2 + 2*k3 + k4)*dt/6
    t += dt
    """


    t_list.append(t)
    y_list.append(y)


plt.plot(t_list, y_list)
plt.xlabel('t')
plt.ylabel('y')
plt.show()


y_1_real = np.e-2
e_rel = np.abs(( y_list[-1] - y_1_real ) / y_1_real )
print(e_rel)
```

4 差分の誤差

まず前進差分の誤差を考える。Taylor 展開

$$y(t+h) = y(t) + y'(t)h + O(h^2) \tag{1}$$

より

$$\frac{y(t+h) - y(t)}{h} = y'(t) + O(h)$$

このように差分と微分の差（打ち切り誤差）は h の 1 乗のオーダーを持つ。これを 1 次の精度を持つと言ったりする。

後退差分の場合は同様で，上の式で h を $-h$ に置き換えると

$$y(t-h) = y(t) - y'(t)h + O(h^2) \tag{2}$$

よって

$$\frac{y(t) - y(t-h)}{h} = y'(t) + O(h)$$

となり，やはり 1 次である。

では中心差分はどうか？ 式 (1) と式 (2) で省略されていた項を書き出してみる。

$$y(t+h) = y(t) + y'(t)h + \frac{y''(t)}{2}h^2 + \frac{y'''(t)}{6}h^3 + O(h^4) \tag{3}$$

$$y(t-h) = y(t) - y'(t)h + \frac{y''(t)}{2}h^2 - \frac{y'''(t)}{6}h^3 + O(h^4) \tag{4}$$

辺々引くと

$$y(t+h) - y(t-h) = 2y'(t)h + \frac{y'''(t)}{3}h^3 + O(h^4)$$
$$\therefore \quad \frac{y(t+h) - y(t-h)}{2h} = y'(t) + O(h^2)$$

となり，2 次精度であることがわかる。

中心差分の例では，（$(-h)^2 = h^2$ なので）$y(t+h)$ と $y(t-h)$ の h^2 の項が打ち消しあってくれて，2 次の精度になっている。このようにマイナスの出てくるところに注意し，うまく係数を調整して引き算することによって，高い精度を引き出すことができる。

この考え方を発展させれば，2 階微分に対する差分を作ることができる。式 (3) と式 (4) を加えると

$$y(t+h) + y(t-h) = 2y(t) + y''(t)h^2 + O(h^4)$$
$$\therefore \quad \frac{y(t+h) - 2y(t) + y(t-h)}{h^2} = y''(t) + O(h^2)$$

となり，左辺は 2 階微分に対する 2 次精度の差分であることがわかる。