科学計算研究室 Python ゼミ

～ 10. 常微分方程式の数値解法 (Runge-Kutta 法)～

2021-04-07 福田　浩

1　原理

常微分方程式の解法として，Euler 法はシンプルで理解しやすいが，誤差が大きくなる場合がある．数値積分の解法には，台形公式と Simpson 公式があった．台形公式が微小区間を 1 次関数近似するのに対し，Simpson 公式では 2 次関数近似することで，誤差を大幅に低減することに成功した．Runge-Kutta 法は常微分方程式の解法の Simpson 公式版のような位置づけである．

$$\frac{dy}{dx} = f(x,y) \tag{1}$$

$$\int dy = \int f(x)\,dx \tag{2}$$

$$\int_{y_0}^{y_1} dy = \int_{x_0}^{x_0+h} f(x)\,dx \tag{3}$$

$$y_1 = y_0 + \int_{x_0}^{x_0+h} f(x)\,dx \tag{4}$$

Simpson 公式が以下の式で表すことが出来たことを思い出そう．

$$\int_{x_1}^{x_3} (\alpha x^2 + \beta x + \gamma)\,dx = \frac{\delta x}{3}(y_1 + 4y_2 + y_3) \tag{5}$$

ここで，$\delta x = h/2$ になるように h を設定すると

$$\int_{y_0}^{y_1} dy = \int_{x_0}^{x_0+h} f(x)\,dx \tag{6}$$

$$= \frac{h}{6}\left\{f(x_0) + 4f\left(x_0 + \frac{h}{2}\right) + f(x_0+h)\right\} \tag{7}$$

となるから，式 4 に式 7 から

$$y_1 = y_0 + \frac{h}{6}\left\{f(x_0) + 4f\left(x_0 + \frac{h}{2}\right) + f(x_0+h)\right\} \tag{8}$$

が得られる．これが Runge-Kutta の公式である．

2　課題

微分方程式の数値解法プログラムを Python で実装せよ

$\frac{dy}{dx} = f(x)$ **で** f **が** x **の** n **次多項式である場合**

- 多項式の次数 `n` と，スライス個数 `N` を標準入力から入力する機能を備えること
- スライス幅 `dx` を標準入力から入力する機能を備えること
- x, y の初期値を標準入力から入力する機能を備えること
- 各係数 `a[]` を標準入力から入力する機能を備えること

- Runge-Kutta 法で求めた y の値を標準出力に出力する機能を備えること
- $\dfrac{dy}{dx} = 2x$ で，初期値 x=0，y=0，スライス幅 dx=0.1，0.05，0.025 の時，x=10 の値を求め，理論値と比較せよ

$\dfrac{dy}{dx} = f(x)$ で $f(x)$ が $\dfrac{1}{\cos^2 x}$ の場合

- スライス個数 N を標準入力から入力する機能を備えること
- スライス幅 dx を標準入力から入力する機能を備えること
- x, y の初期値を標準入力から入力する機能を備えること
- Runge-Kutta 法で求めた y の値を標準出力に出力する機能を備えること
- 初期値 x=0，y=0，スライス幅 dx=0.1，0.05，0.025 の時，x=1.5 の値を求め，理論値と比較せよ

3 参考：C++ソースコード

$\dfrac{dy}{dx} = f$ で f が $2x$ の場合

```
#include <iostream>
#include <math.h>

using namespace std;

int main(void){
    double x, y, dx, a[10];
    int n, N;

    cin >> n >> N;
    cin >> dx;
    cin >> x  >> y;
    for (int i=0;i<n+1;i++){
        cin >> a[i];
    }

    for(int i=0;i<N+1;i++){
        cout << x << "\t" << y << endl;
        for(int j=0;j<n+1;j++){
            y += a[j]*pow(x,     j)/6*1*dx;
            y += a[j]*pow(x+dx/2,j)/6*4*dx;
            y += a[j]*pow(x+dx,  j)/6*1*dx;
        }
        x += dx;
    }

    return 0;
}
```

$\dfrac{dy}{dx} = f$ で f が $\dfrac{1}{\cos^2 x}$ の場合

```
#include <iostream>
#include <math.h>

using namespace std;

int main(void){
    double x, y, dx, a[10];
    int N;

    cin >> N;
    cin >> dx;
    cin >> x  >> y;

    for(int i=0;i<N+1;i++){
        cout << x << "\t" << y << endl;
        y += pow(cos(x     ),-2)/6*1*dx;
        y += pow(cos(x+dx/2),-2)/6*4*dx;
        y += pow(cos(x+dx  ),-2)/6*1*dx;
        x += dx;
    }

    return 0;
}
```

4 更なる検討

Runge-Kutta 法の計算誤差を考察せよ.

- $\dfrac{dy}{dx} = 2x$ の計算結果と理論値を比較せよ
- $\dfrac{dy}{dx} = \dfrac{1}{\cos^2 x}$ の計算結果と理論値を比較せよ