連続系アルゴリズム
レポート課題3

連絡先：mail@myuuuuun.com

平成 28 年 2 月 16 日

レポート内で使用したプログラムは GitHub にアップロードしています。 https://github.com/myuuuuun/various/tree/master/ContinuousAlgorithm/HW3/

問題 1

前進オイラー法の絶対安定領域をもとめよ。

テスト問題 $\frac{dy}{dx} = \lambda x,\ \ y(0) = 1$ に対して、前進オイラー法 $y_{n+1} = y_n + \lambda y_n \Delta x$ が安定かどうかを考える。

$$\left| \frac{y_{n+1}}{y_n} \right| = |1 + \lambda \Delta x| < 1$$

をみたせばよいので、$1 + \lambda \Delta x = re^{i\theta}$ とおけば、

$$\lambda \Delta x = re^{i\theta} - 1$$

したがって前進オイラー法の絶対安定領域は、複素平面上で $(-1, 0)$ を中心とする半径 1 の円の内部となる。

問題 2

万有引力を及ぼし合う 2 つの質点の運動を適当なルンゲクッタ法で解け。

※問題設定が上手くできなかったため、もう少し単純な問題 $\frac{d^2x}{dt^2} = -x$ を解いています。
※ plot の都合上 python を使っています

ソースコード: hw3-1.py（主要部のみ）

```
#-*- encoding: utf-8 -*-
"""
solve ordinary differential equations

Copyright (c) 2016 @myuuuuun
Released under the MIT license.
"""
import math
import numpy as np
import pandas as pd
import functools
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.cm as cm
EPSIRON = 1.0e-8
np.set_printoptions(precision=3)
np.set_printoptions(linewidth=400)
np.set_printoptions(threshold=np.nan)
pd.set_option('display.max_columns', 130)
pd.set_option('display.width', 1400)
plt.rcParams['font.size'] = 14

# 日本語対応
mpl.rcParams['font.family'] = 'Osaka'

# Explicit RK4 Method
# 4 段4 次ルンゲ・クッタ
def runge_kutta(func, init, t_start, step, repeat):
    if not isinstance(func, list):
        func = [func]
    if not isinstance(init, list):
        init = [init]
    if len(init) != len(func):
        raise ValueError微分係数の数と初期値の数が一致しません("")
    dim = len(func)
    path = np.zeros((dim+1, repeat), dtype=float)
    path[:, 0] = [t_start] + init
    k1 = np.zeros(dim, dtype=float)
    k2 = np.zeros(dim, dtype=float)
    k3 = np.zeros(dim, dtype=float)
    k4 = np.zeros(dim, dtype=float)

    for i in range(1, repeat):
        current = path[1:, i-1]
        path[0, i] = t_start + i * step

        # k1
```

```
        for s in range(dim):
            k1[s] = func[s](current)

        # k2
        for s in range(dim):
            k2[s] = func[s](current + step * 0.5 * k1)

        # k3
        for s in range(dim):
            k3[s] = func[s](current + step * 0.5 * k2)

        # k4
        for s in range(dim):
            k4[s] = func[s](current + step * k3)

        path[1:, i] = current + step * (k1 + 2*k2 + 2*k3 + k4) / 6

    return path


if __name__ == '__main__':
    """
    Sample: solve x''(t) = -x, x(0) = 1, x'(0) = 0

    analytic solution is x(t) = cos(t)
    """
    x = lambda array: array[1]
    dx = lambda array: -1 * array[0]
    func = [x, dx]
    init = [1, 0]
    t_start = 0
    step = 0.01
    repeat = 10000

    ts = np.arange(t_start, step*repeat, step)
    true_path = np.cos(ts)
    euler_path = euler(func, init, t_start, step, repeat)
    modified_euler_path = modified_euler(func, init, t_start, step, repeat)
    rk4_path = runge_kutta(func, init, t_start, step, repeat)

    fig, ax = plt.subplots(figsize=(16, 8))
    plt.title(r'Initial value problem $\"x = -x(t)$')
    plt.xlabel("t")
    plt.ylabel("x")

    plt.plot(ts, true_path, color='orange',
         linewidth=3, label="true_path(x=cos(t))")
    plt.plot(euler_path[0], euler_path[1], color='blue',
         linewidth=1, label="Euler approx")
    plt.plot(modified_euler_path[0], modified_euler_path[1], color='green',
         linewidth=2, label="Modified Euler approx")
    plt.plot(rk4_path[0], rk4_path[1], color='red',
         linewidth=1, label="RK4 approx")

    plt.legend()
    plt.show()
```

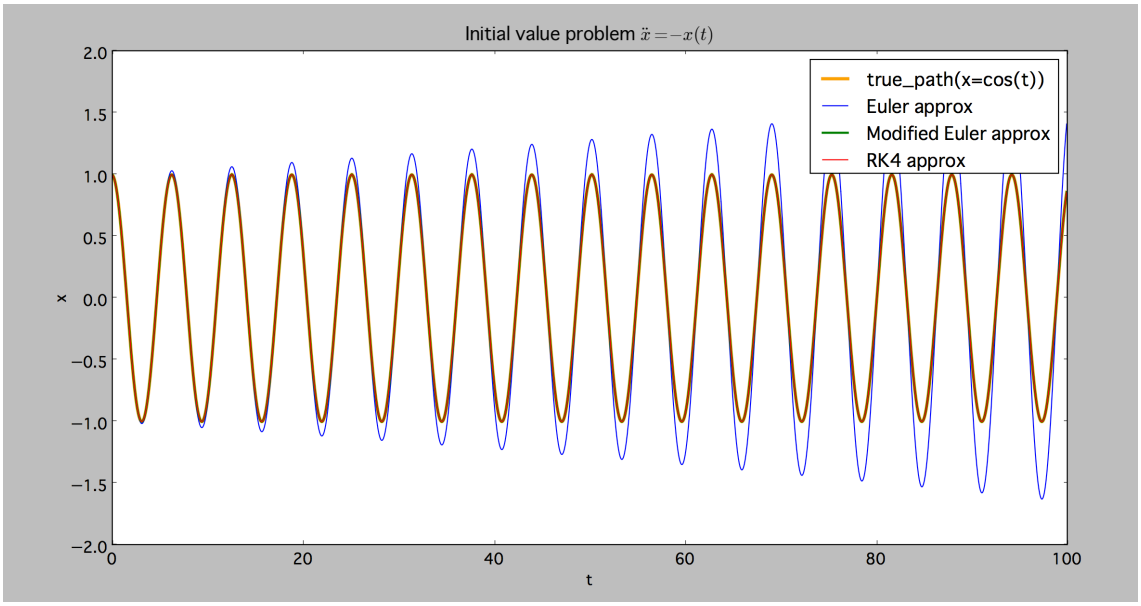

図 1: 解析解, 前進オイラー法, 修正オイラー法, 4 段 4 次陽的 RK 法を比較