#043★★★★ 2022.5.23 / 2023.7.29

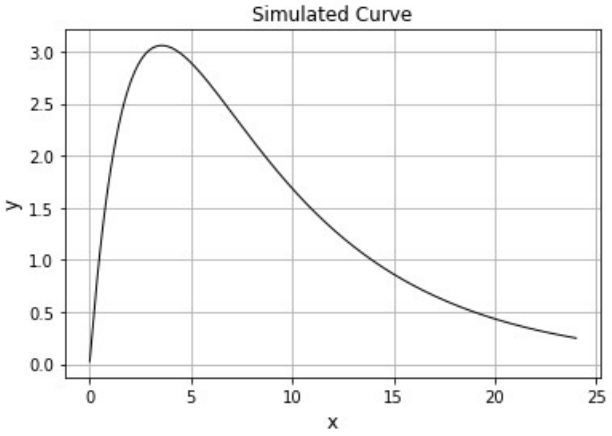
高速ラプラス逆変換（FILT）（１）～薬物速度論への応用。

電気工学領域などで「ラプラス変換」という数学的手法が用いられ、大学生や一般向けテキストが本屋に並んでいる。ラプラス変換とは、例えば複雑な微分方程式を解くためにラプラス変換形という「複素数次元の式」を利用する方法である（詳細は成書を参照）が、式が複雑な場合には解析的な解が得られない場合がある。その場合「ラプラス逆変換」を数値的に行う方法として細野敏夫氏（工学博士）が 1984 年に「BASIC による高速ラプラス変換」という著書を発刊し、今でも工学系領域で引用されている。その原点は、論文 'T. Hosono, Numerical inversion of Laplace transform and some applications to wave optics, Radio Sci., 16(6), 1015-1019 (1981).' である。ラプラス変換した（複素数次元の）数式を数値的に元に戻す（時間次元の数値解を得る）操作は高速逆ラプラス変換（Fast Inverse Laplace transform, FILT）と称される。その定義は次のとおり。

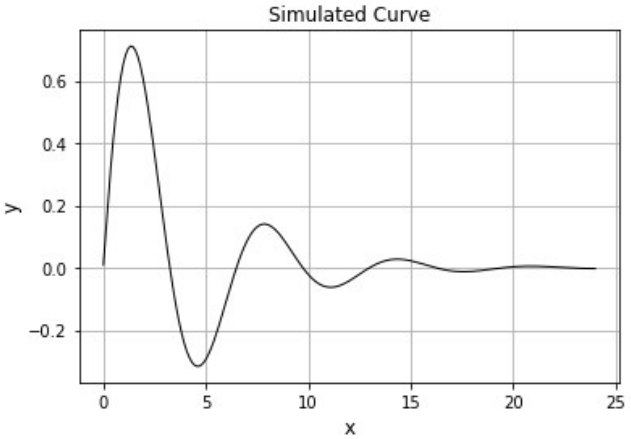
$$f(t) \cong \frac{exp(a)}{t} \sum_{n=1}^{\infty} (-1)^n \cdot g_n$$

$$g_n \cong Im\left[\tilde{f}\left(\frac{a + j(n-0.5)\pi}{t}\right)\right]$$

ここで、$f(t)$ は時間次元の解、$\tilde{f}(s)$ はラプラス次元の解、Im は複素数のうちの虚数部（を取り出す操作）を示す。当時の書籍では BASIC 言語が用いられ、虚数を扱えずに数式の工夫をしたこと、無限級数の近似を精度よく短時間で求めるためにオイラー級数を利用したことなど、高度な工夫がなされていた。いまでこそ簡単に計算が行える環境になっているが、当時の数学者の工夫と発想に感服する。



Python でプログラミングしてみた。上記オイラー級数を利用していないのでやや計算時間が必要な場合もある。薬物動態経口投与 1-コンパートメントモデル（上図）、さらに薬学系ではないが少し複雑なモデル式を試してみた（下図）。それぞれの図に対応するラプラス変換形の式は次のとおり。



$$\tilde{f}(s) = \frac{Dose \cdot ka}{Vd} \cdot \frac{1}{(s + ka)(s + ke)}$$

$$\tilde{f}(s) = \frac{1}{(s^2 + 0.5 \cdot s + 1)}$$

各数値は後出のリスト内を参照。`def fs(s1)`内の `#` はコメント行を意味し、どちらかの `#` を削除してその式をグラフ描画に用いる。プログラム中末尾の `x_max` はx軸の最大値を示し、この数値を変更することで時間軸の幅（最大値）を変更できる。リストを次のページに示す。定数 a は 6 とした。

```
# FILT
import math
import cmath
import matplotlib.pyplot as plt
def fs(s1):
    #return 10.0 / 2.0 * 0.5 / (s1 + 0.5) / (s1 + 0.1386)
    return 1 / ( s1 * s1 + 0.5 * s1 + 1)
def ft(t, aa):
    sum1 = 0
    for n in range(1, 10000):
      # ここでは強引な有限項反復計算なので、この最大数が小さいと計算精度が悪い。n = 100000 などが必要な場合がある。
        if (t == 0):
            t = 0.01
        s1 = complex(aa / t, (n - 0.5) * 3.141592 / t)
        imfs = fs(s1).imag # Imaginary part of f(s)
        sum1 = sum1 + (-1) ** n * imfs
    sum1 = sum1 * math.exp(aa) / t
    return sum1
def plots(data_x, data_y):
    plt.title("Simulated Curve", fontsize = 12)
    plt.xlabel("x", fontsize = 12)
    plt.ylabel("y", fontsize = 12)
    plt.grid()
    plt.plot(data_x, data_y, linestyle = "solid", linewidth = "1", color = "black")
    plt.show()
if __name__ == '__main__':
    data_x = []
    data_y = []
    x_min = 0
    x_max = 24
    n_div = 200
    for x in range(n_div + 1):
        xx = (x_max - x_min) * x / n_div
        data_x.append(xx)
        data_y.append(ft(xx , 6))
    plots(data_x, data_y)
```