課題２：レポート

著者名：村上裕亮

日付：2014.6.6

1　はじめに

このレポートでは、包絡線定理について説明し、次いで自分のコードについて説明し、工夫したところや今後の課題を説明する.

2　包絡線定理

2.1　一般的な包絡線定理

ある多変数関数（$n \geq 1$）$f(x_1, x_2, \cdots, x_n, t)$ に対して、t の値をある値 $\bar{t}$ で固定すると、$y = f(x_1, x_2, \cdots, x_n, \bar{t})$ という方程式は $n+1$ 次元空間の面になることが分かる. この面を c とする. そこで、t の値を 1 つ決めると面が 1 つ定まることが分かり、t の値を動かしてその面の通過領域の境界を C とすると、C が定まる場合がある. このときの C を表す関数を $F(x_1, x_2, \cdots, x_n)$ とすると、

$$F(x_1, x_2, \cdots, x_n) = \max_t f(x_1, x_2, \cdots, x_n, t) \tag{1}$$

ないし、

$$F(x_1, x_2, \cdots, x_n) = \min_t f(x_1, x_2, \cdots, x_n, t) \tag{2}$$

となるはずだ.

また、このとき、F 上の点を 1 点とり、その点を $(x_1^*, x_2^*, \cdots, x_n^*)$ とすると、この点に対応する t は (1) や (2) を解いた最大化（最小化）問題の解になるので一意に定まることが分かる. その解を $\bar{t}$ とおくと、面 C と面 c は $(x_1^*, x_2^*, \cdots, x_n^*, \bar{t})$ で接することが分かる. したがって、面 C の点 $(x_1^*, x_2^*, \cdots, x_n^*)$ での接面を求めたければ、面 c の点 $(x_1^*, x_2^*, \cdots, x_n^*, \bar{t})$ での接面を求めれば良いと分かる.F 上の全ての点で同じことが言えるので、全ての $i(1 \leq i \leq n)$ について以下が成立する.

$$\frac{\partial}{\partial x_i} F(x_1, x_2, \cdots, x_n) = \frac{\partial f}{\partial x_i}(x_1, x_2, \cdots, x_n, \bar{t}) \tag{3}$$

ここで、$F(x_1, x_2, \cdots, x_n) = f(x_1, x_2, \cdots, x_n, \bar{t})$ だから、(3) は

$$\frac{\partial}{\partial x_i} f(x_1, x_2, \cdots, x_n, \bar{t}) = \frac{\partial f}{\partial x_i}(x_1, x_2, \cdots, x_n, \bar{t}) \tag{4}$$

と変形できる.(4) の主張は、接点 (=最大化問題を解いた解) であれば、代入してから微分しても、微分してから代入しても結果は変わらないというところにあり、これが包絡線定理の主張となる（尾山・安田 [1]）.

2.2　具体例

$n = 1$ の場合で、多変数関数を

$$f(x,t) = xt - t^2 \tag{5}$$

とする.t を実数全体で動かしたときの包絡線を求めると、(5) は t について 2 次関数なので、

$$f(x,t) = -\left(t - \frac{x}{2}\right)^2 + \frac{x^2}{4}$$

となるので、

$$F(x) = \max_t f(x,t) = \frac{x^2}{4}$$

と分かる. このことを Python に図示させたのが以下の図 1 や図 2 だ.

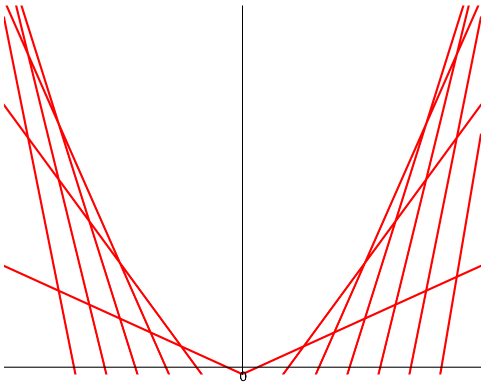

図 1: 包絡線（1）

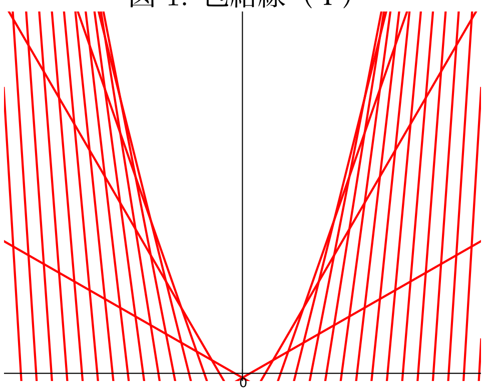

図 2: 包絡線（2）

これらの形を見ると、放物線の形が見えることが分かる.

3　Python プログラム

3.1　用いたコード

今回表示するのに用いた Python のコードは以下だ.

```
from mpl_toolkits.axes_grid.axislines import SubplotZero
import matplotlib.pyplot as plt
import numpy as np
import pylab as pl

FIGNUM = 0

if FIGNUM == 0:
    n = 13
    area = 1
if FIGNUM == 1:
    n = 31
    area = 1.5


def f(x,a):
    return area*(-x**2+a*x)

def subplots():

    fig = plt.figure(1)
    ax = SubplotZero(fig, 111)
    fig.add_subplot(ax)

    for direction in ["xzero", "yzero"]:
        ax.axis[direction].set_axisline_style("-|>")
        ax.axis[direction].set_visible(True)

    for direction in ["left", "right", "bottom", "top"]:
        ax.axis[direction].set_visible(False)

    return fig, ax

fig, ax = subplots()

plt.axis([-n**0.8, n**0.8, -n*0.02, n])
```

```
plt.xticks([0])
plt.yticks([])

a = np.linspace(-2*n, 2*n, 100*n)

for x in range(n):
    y = f(x-n/2+0.5, a)
    ax.plot(a, y, 'r-', linewidth=2)
plt.savefig('envelope' + str(FIGNUM) + '.png',
transparent=True, bbox_inches='tight', pad_inches=0)
plt.savefig('envelope' + str(FIGNUM) + '.pdf',
bbox_inches='tight', pad_inches=0)
plt.show()
```

以降、このコードについての説明を施す.

```
from mpl_toolkits.axes_grid.axislines import SubplotZero
import matplotlib.pyplot as plt
import numpy as np
import pylab as pl
```

では、必要なモジュールを Python に読み込んだ.

```
FIGNUM = 0

if FIGNUM == 0:
    n = 13
    area = 1
if FIGNUM == 1:
    n = 31
    area = 1.5
```

では、`FIGNUM` を 0 と 1 で切り替えることで、2 種類のパラメータの切り替えを行った. ここで、`n` は直線の本数、`area` はその密度（値が大きいほど疎、小さいほど密）を表している.

```
def f(x,a):
    return area*(-x**2+a*x)
```

では、今回用いる関数を定義した. ここで式全体に `area` がかかっているのは、密度を変化させるためだ.

```
def subplots():
```

```
    fig = plt.figure(1)
    ax = SubplotZero(fig, 111)
    fig.add_subplot(ax)

    for direction in ["xzero", "yzero"]:
        ax.axis[direction].set_axisline_style("-|>")
        ax.axis[direction].set_visible(True)

    for direction in ["left", "right", "bottom", "top"]:
        ax.axis[direction].set_visible(False)

    return fig, ax

fig, ax = subplots()
```

では、グラフの軸などを変更した（ヒントは Web[2] から得た）. 具体的には、軸を表示させ、デフォルトでは周囲に値が表示されるがそれを消した.

```
plt.axis([-n**0.8, n**0.8, -n*0.02, n])

plt.xticks([0])
plt.yticks([])
```

では、グラフの表示される領域と、軸上に表示させる数値を入力している. ここで、グラフの表示領域に n が含まれているのは、直線の本数に応じて表示領域を変化させるためだ. 横軸上に 0 をプロットし、それ以外の数字は表示させていない.

```
a = np.linspace(-2*n, 2*n, 100*n)
```

では、a で直線の幅を設定した（ここでは-2*n から 2*n）.

```
for x in range(n):
    y = f(x-n/2+0.5, a)
    ax.plot(a, y, 'r-', linewidth=2)
```

では、x をパラメーターに for 文で繰り返し直線を引いた.ax.plot は、a の間隔ごとに y をとり、その間は直線で補完して線を描く指示をしている.

```
plt.savefig('envelope' + str(FIGNUM) + '.png',
transparent=True, bbox_inches='tight', pad_inches=0)
plt.savefig('envelope' + str(FIGNUM) + '.pdf',
bbox_inches='tight', pad_inches=0)
plt.show()
```

では、完成した図を png 形式と pdf 形式のファイルにそれぞれ保存し、最後にその図を表示させている.

3.2　このコードの課題

グラフの描画範囲は今回は恣意的に `plt.axis([-n**0.8, n**0.8, -n*0.02, n])` としたが、包絡線の形によって描画範囲はその都度変えなければならず、多変数方程式の形が変わると対応できないことになってしまう. 望ましいのは包絡線全体が映り、かつ、それ以外の部分はあまり映らないことで、包絡線と直線（場合によっては曲線）の接点の座標を取得して、その範囲というように設定するのが本来は望ましいと思われる.

参考文献

[1] 尾山大輔・安田洋祐「経済学で出る包絡線定理」『経済セミナー』2011 年 10・11 月号.

[2] http://matplotlib.org/examples/axes_grid/demo_axisline_style.html