離散化

1　離散化された x 軸上での関数

　コンピューターでは連続した x 軸を扱うことが出来ないため、離散化して扱う。

Jupyter notebook で新しくファイルを作成し、series と名付けよう。

```
import numpy as np
```

　まずは離散化された世界での x 軸を生成する。

```
num = 10
x = np.linspace(0, 2 * np.pi, num, endpoint=False)
x
```

　上のコードを実行することで、x は$[0,2\pi)$の区間を 10 等分した値が格納された配列となる。

　linspace は線形に区切ったグリッドを生成する関数だ。類似関数として、対数グリッドを生成する logspace が存在する。

　endpoint は端点を含むかどうかを決めるオプションであり、True にすると、区間が$[0,2\pi]$になる。

　次に、この軸上で sin 関数を表現してみよう。

```
y = np.sin(x)
```

　ここで numpy では sin 関数は要素ごとに作用することを利用した。すなわち、上の式は

```
y = np.array([np.sin(x[i]) for i in range(num)])
```

　と等価である。また、任意のiに対して

$$y[i] == \sin(x[i])$$

　が成立することを確認して欲しい。

2 グラフに表示する

さて、せっかく sin 関数を生成したので、よく見える形で確認したい。そのために matplotlib というツールを使う。

```
import matplotlib.pyplot as plt
```

上のような形で matplotlib.pyplot を plt という名前で import して欲しい。この plt は描画のための強力なツールとなっている。ためしに、横軸を「x」、縦軸を「y」にしたグラフを描いてみよう。

```
plt.plot(x, y)
```

上のコードを実行すると、非常に荒い sin カーブが表示されるはずである。荒い原因はもちろん x が区間を 10 等分したものだからである。区間を 50 等分したらもちろんもっとましなグラフが書けるようになる。

2.1 課題

x を 50 等分で作り直し、滑らかなグラフを描け。

3 複数のグラフの描画

複数のグラフを描画することももちろん可能である。

```
y1 = x ** 1
y2 = x ** 2
plt.plot(x, y1)
plt.plot(x, y2)
```

上のコードを実行すると$y = x$, $y = x^2$の両方のグラフが同時に描画されていることがわかるだろう。ここではさらにグラフにラベルを付けることも出来る。

```
y1 = x ** 1
y2 = x ** 2
plt.plot(x, y1, label="x")
plt.plot(x, y2, label="x ** 2")
plt.legend()
```

このようにグラフを表示することは時折有効である。

4 級数

さて、今回の課題では級数を取り扱う。級数の最も代表的な例として、exp 関数のマクローリン展開を試してみよう。exp 関数の n 回目の微分は

$$\frac{d^n}{dx^n}\exp(x) = \exp(x)$$

であることを利用すると、exp 関数のマクローリン展開は

$$\exp(x) = 1 + x + \frac{x^2}{2} + \cdots + \frac{x^n}{n!} + \cdots = \sum_i \frac{x^i}{i!}$$

となる。では、このマクローリン展開を有限項で打ち切った時の関数の形をプロットしてみよう。また、今回は x の範囲を少し変えて$[-3,3)$の範囲でプロットしてみよう。まずは通常の exp をプロットしてみる。

```
x = np.linspace(-3, 3, num=num, endpoint=False)
y = np.exp(x)
plt.plot(x, y)
```

次に、4 次までの級数展開をプロットし、比較してみよう。

```
x = np.linspace(-3, 3, num=num, endpoint=False)
y = np.exp(x)
y_series = 1 + x + x ** 2 / 2 + x ** 3 / 6 + x ** 4 / 24
plt.plot(x, y, label="exact")
plt.plot(x, y_series, label="series")
plt.legend()
```

最後に、任意の次数のマクローリン展開を計算する関数を記述しよう。ただし、exp の表現には階乗関数が必要だが、これは numpy には実装されていない。自分で実装しても良いが、scipy というライブラリに実装されているので使ってみよう。階乗は factorial という名前の特殊関数として scipy に存在する。

```
from scipy.special import factorial
```

次に、任意の次数の exp のマクローリン展開を計算する関数を記述しよう。

```
def maclaurin_exp(x, n):
    y = np.zeros_like(x)
    for i in range(n):
        y += x ** i / factorial(i)
    return y
```

4.1 課題

maclaurin_exp を使って[−7,7)の範囲で 10 次の exp のマクローリン展開をプロットしてみよう。

5 その他の関数のマクローリン展開

exp 以外の関数、例えば、cos のマクローリン展開を考えよう。

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{24} - \cdots$$

と表現される。exp と違い、cos 関数の場合、奇数次の項は存在せず、偶数 i に対して第 i 項は

$$\frac{(-1)^{i/2}}{i!} x^i$$

となる。Range を上手に使えば、このような一つ飛ばしの場合も簡単に実装できる。

```
def maclaurin_cos(x, n):
    y = np.zeros_like(x)
    for i in range(0, n, 2):
        y += x ** i / factorial(i) * (-1) ** (i / 2)
    return y
```

```
x = np.linspace(-4, 4, num=num, endpoint=False)
y = np.cos(x)
y_series = maclaurin_cos(x, 10)
plt.plot(x, y, label="exact")
plt.plot(x, y_series, label="series")
plt.legend()
```

5.1 課題

$$\sin x = x - \frac{x^3}{6} + \frac{x^5}{120} - \cdots$$

であり、sin のマクローリン展開は奇数 i に対して第 i 項は

$$\frac{(-1)^{(i-1)/2}}{i!} x^i$$

となることを利用し、sin のマクローリン展開を実装せよ。また、実際の sin と比較せよ。

6 保存しておこう

せっかくいろいろな関数のマクローリン展開を実装したので、これらを一枚の Python ファイルに書き込んでおいていつでも import 出来るようにしておこう。

macraurin.py

```
import numpy as np
from scipy.special import factorial


def maclaurin_exp(x, n):
    y = np.zeros_like(x)
    for i in range(n):
        y += x ** i / factorial(i)
    return y
```

```
def maclaurin_cos(x, n):
    y = np.zeros_like(x)
    for i in range(0, n, 2):
        y += x ** i / factorial(i) * (-1) ** (i / 2)
    return y


def maclaurin_sin(x, n):
    ...
```

こうすることで、いつでもこれらの関数を import して使うことが出来る。

```
import macraurin
plt.plot(x, macraurin.maclaurin_cos(x, 10))
```

6.1 課題

scipy を使わずに factorial を実装してみよう。

Arctan のマクローリン展開を実装し、プロットを比較してみよう。