Anaconda と Jupyter Notebook を用いた Python プログラミング（3）

※このドキュメントは相互参照型のハイパーテキストを意図しており，Web サイトおよびブックマーク，図表，数式，補足説明へのリンク（参照）を下線付き青文字で表し，（↩）の記号は初出の参照元へのリンクを表す。

3．NumPy と SciPy, Matplotlib の利用

3-1　NumPy と SciPy の利用

NumPy（ナンパイ）は Python（パイソン）から利用できる数値計算 (numerical calculation（ニュメリカル キャルキュレイション）) のライブラリであり，SciPy（サイパイ）は科学技術計算に用いられるライブラリである。大規模データを処理する場合には NumPy の利用が必須となる。また，複雑な理論モデルや，最適化アルゴリズムのライブラリを使用する場合など SciPy の利用が必要になる場合もある。

3-2　Matplotlib の利用

Matplotlib（マットプロットリブ）は Python を用いてグラフを作成したり，データを可視化するために標準的に用いられるライブラリである（Matplotlib の web サイト： https://matplotlib.org/ ）。

例えば Matplotlib を利用して双曲線正接函数（かんすう）(hyperbolic tangent（ハイパボリック タンジェント） function（ファンクション）; tanh x) 曲線の形状を確認したいときには，以下のようなコード (Code 3.2.1) を用いれば良い。

(Code 3.2.1) Matplotlib を利用した双曲線正接函数曲線の描画例 (↩)

```
1) import matplotlib.pyplot as plt # matplotlib.pyplot を plt としてインポートする
2) import numpy as np # numpy を np としてインポートする
3) x = np.linspace(-4,4,400) # x の範囲を -4 から 4 とし 400 分割する
4) y = np.tanh(x) # 双曲線正接関数値を y とする
5) fig, ax = plt.subplots()  # fig(figure),ax(axes) オブジェクトを使う
6) ax.plot(x,y) # x に対して y をプロットする
7) plt.show() # 画面にプロットを表示する
```

(↩)

(Code 3.2.1) を Jupyter Notebook のコードセルから実行すれば，Figure 3.2.1 のようなグラフが表示される。

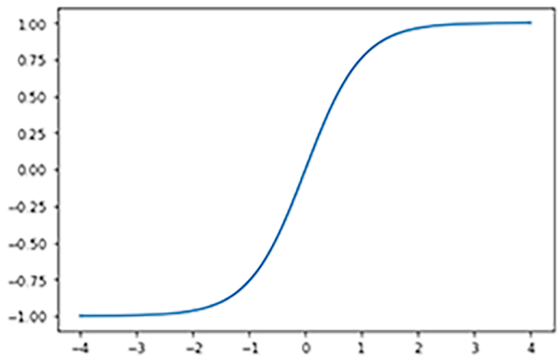

Figure 3.2.1 (Code 3.2.1) により得られる双曲線正接函数曲線の形状（↩）

Figure 3.2.1 に示したグラフの画像は，画面スクリーンショットにより作成したビットマップ画像であるが，レポートや論文中のグラフ・線図には，ベクタ画像を用いるべきである。（補足 3.2.A）Microsoft 365 のプレゼンテーション・ソフト PowerPoint（パワーポイント）では “*.svg” 形式のベクタ画像を挿入することができる。

しかし Microsoft 365 のワープロ・ソフト Word（ワード）ではベクタ画像を挿入できず，グラフを表示する場合にも，やむを得ずビットマップ画像を使う場合がある。（補足 3.2.B）

補足

（補足 3.2.A）ビットマップ画像とベクタ画像（↩）

ビットマップ画像 (bitmap image) は，19 世紀フランス人画家のジョルジュ・スーラ (Georges Seurat) の点描画法 (Figure 3.2.A.1) のように，色と明るさを変えた細かい点（ドット; dot）の集まりとして 2 次元の画像を表現する。**ラスタ**（ラスター）画像 (raster image) という語も，同じ意味で用いられる。２次元検出器を使って画像や動画を撮影したり，デジタル化された印刷機で印刷物を作成，液晶ディスプレイで画像・動画を表示するなど，視覚情報に関するデジタル入出力では普通に用いられる表現形式である。

ビットマップ画像は，ヒトがペンや筆（ふで）を使って画像や文字を表現しようとするのとは異なり，やや無駄に大量の情報資源（メモリ）を消費する傾向がある。

ビットマップ画像を保存・再生するためのファイル形式の代表的なものに，ビットマップ (bitmap) 形式 (*.bmp または *.raw)，TIFF（ティフ）形式 (tagged image file file format) (*.tif あるいは *.tiff)，JPEG（ジェイペグ） (joint photographic experts group; JPEG) 形式 (*.jpg あるいは *.jpeg)，GIF（ジフ）形式 (general interchange format) (*.gif)，PNG（ピーエヌジー） (portable network graphics) 形式 (*.png)，RGBA (red-green-blue-alpha) 形式 (*.rgba) などがある。環境に依存する可能性はあるが，matplotlib.pyplot.savefig() メソッドでは，出力ファイルの名称を “*.raw”, “*.tif”, “*.tiff”, “*.jpeg”, “*.jpg”, “*.png”, “*.rgba” とすれば，それぞれ対応するファイル形式でビットマップ画像データファイルを出力させ，補助記憶装置に記録することができる。

音声付きの動画を保存するためのファイルの形式として国際標準化機構 (International Standardization Organization; ISO) の策定した MPEG-4（エムペグ） (*.mp4) がある。

また，Web ページやプレゼンテーション・ソフトなどで，GIF 動画（GIF（ジフ） アニメ） (*.gif) の用いられる場合もある。

ビットマップ画像・動画の作成，編集をするための市販ソフトウェアとしては， Adobe 社の Photoshop が代表的である。Adobe Photoshop を用いれば，複数のビットマップ画像ファイルから GIF 動画を作成することもできる。

Figure 3.2.A.1　ジョルジュ・スーラ「グランド・ジャット島の日曜日の午後」 (1884-1886) シカゴ美術館（元の画像は TIFF，1.2 MB，666 × 444 pixel^2, A4 版で解像度 167 dpi だが PDF 化により低画質化している）　(↩)

現時点で，動作環境によらず Matplotlib の matplotlib.pyplot.savefig() メソッドで出力が可能であり，Microsoft 365 のワードプロセッサ Word などで作成する文書に確実に挿入できるビットマップ画像の形式は PNG 形式である。Adobe Photoshop や他のビットマップ画像処理ソフトを用いて，PNG 形式とそれ以外のビットマップ画像ファイル形式を相互に変換・保存することも困難ではない。

ベクタ画像（ベクトル画像） (vector image) のデータは，ペンや筆を使って幾何学的な図形を表現しようとする「線画」(line art) に近い。線分を描画するためには始点と終点の位置の情報があれば良く，曲線を描画するためにも経由点の位置などを指定して滑らかな曲線（ベジェ曲線）を描く方法が用いられる。15-16 世紀ドイツ人画家アルブレヒト・デューラー (Albrecht Dürer) の水彩画 (Figure 3.2.A.2) では，筆の動かし方，筆圧の変化などで２次元の画像が表現される。

ベクトル画像には，ビットマップ画像と異なりスケーラブルな (scalable)（尺度を変えられる）特徴があり，線画の場合には，拡大表示しても線の輪郭がぼやけたりギザギザになることがない。

ベクタ画像を保存・再生するためのファイル形式として代表的なものに，ポストスクリプト (PostScript) 形式 (*.ps)，EPS (encapsulated PostScript) 形式 (*.eps)，PICT 形式 (*.pict または *.pic, *.pct)，PDF (portable document format) (*.pdf)（可搬性文書形式），SVG (scalable vector graphics; SVG) 形式 (*.svg)，圧縮 SVG (zipped SVG) 形式 (*.svgz)，ウィンドウズ・メタファイル形式 (Windows metafile format; WMF) (*.wmf)，拡張メタファイル形式 (enhanced metafile format; EMF) (*.emf) などがある。matplotlib.pyplot.savefig() メソッドでは，出力ファイルの名称を “*.ps”, “*.eps”, “*.pdf”, “*.svg”, “*.svgz”, “*.emf”とすれば，それぞれ対応するファイル形式でベクタ画像データが出力される。

ベクタ画像の作成，編集をするための市販ソフトウェアとしては， Adobe 社の Illustrator が代表的である。

Figure 3.2.A.2　アルブレヒト・デューラー「若いうさぎ (young hare)」 (1502) アルベルティーナ美術館（元の画像は 3.2 MB，850 × 943 pixel^2，A4 版で解像度 294 dpi だが，PDF 変換で低画質化している）(↩)

(↩)

（補足 3.2.B）論文・レポート用のグラフを作成する場合に注意すべきこと（↩）

Figure 3.2.1 を再掲する。

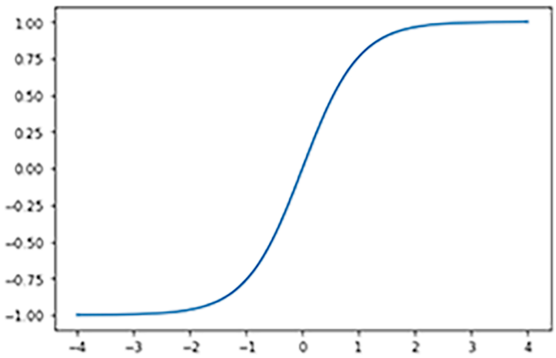

Figure 3.2.1 (Code 3.2.1) により得られるグラフ（↩）

Figure 3.2.1 には軸ラベル axis label（アクスィス ラベル）がついておらず，論文やレポートなどに掲載する図としては，そのままでは使えない。一般的に，大学の学生実験では，レポートを書く際の一般的な注意として，グラフには軸ラベルを必ず付けるように指示されるはずである。

Figure 3.2.1 では左軸 left axis と下軸 bottom axis に目盛 ticks が打たれ，目盛の表す数値が示されているが，右軸 right axis と 上軸 top axis にも，左軸と下軸の目盛と同じ位置に目盛を打った方が，読者はグラフから数値を読み取りやすくなる。

Figure 3.2.1 のグラフの例では，双曲線正接函数曲線の特徴は，原点を通る水平線 horizontal line（ホリゾンタル）と垂直線 vertical line（ヴァーティカル）を引くか，グリッド grid を表示する方がわかりやすい。グラフ内での曲線の上下の余白 margin（マージン）はあった方が良いが，左右の余白は無い方が良い。

Matplotlib でのデフォルトフォントはサンセリフ体 (Sans-serif) である。セリフ体 (Serif) と比べてサンセリフ体はフォントウェイト（線の太さ）を重めにしやすく，低解像度の表示デバイスで表示する場合の視認性に優れる面がある。プロジェクタを使ったプレゼンテーションのための資料では，低解像度表示デバイスを用いて閲覧されることが前提となり，サンセリフ体の用いられる例の多くなる傾向がある。一方で，サンセ

リフ体では，アラビア数字の 1 (いち) とアルファベット小文字の l (エル)，アルファベット大文字の I (アイ)，縦罫線記号 (vertical line) | がそれぞれ 1 (いち)，l (エル)，I (アイ)，| のように表示され，字形の違いを少し区別しにくくなる難点もある。

科学技術系の記事では，物理量を表す記号や変数をイタリック体（斜体） *a*, *b*, *c*, *d*, ..., *A*, *B*, *C*, ... のように表記し，単位を表す記号や定数はローマン体（立体） a, b, c, d, ..., A, B, C, ... で表記することが標準的である。セリフ系のフォント・ファミリーを用いれば，イタリック体とローマン体とで字形がかなり異なるので，物理量を表す記号と単位を表す記号との区別がしやすくなる。

日本の小学校から高校までの教科書や，新聞，多くの雑誌にも見られるように，印刷物では「見出し」にサンセリフ系の「ゴシック体」のボールド字（太字）を用い，本文にはセリフ系の「明朝体」非ボールド字を使うページデザインの用いられることが一般的である。

欧米の学術雑誌出版社は，論文中の図の原稿として TIFF 形式のビットマップ画像ファイルあるいは EPS 形式のベクタ画像ファイルであれば受け付ける場合が多い。科学技術系の論文・レポートに掲載する図の中で用いる文字記号は，本文中で用いる文字記号とフォントファミリー・字形まで含めて一致させるのが望ましい。

例えば，ビットマップ画像ファイルを論文の図原稿として提出する場合に，線画（グラフやイラストレーション）であれば解像度 600 dpi (dots per inch) 以上，ハーフトーン画像（電子顕微鏡写真など）であれば解像度 300 dpi 以上と指定される場合がある。

学術雑誌に掲載される記事では，二段組のレイアウト (two-column layout) が取られ，米国以外の国では A4 版縦長 （210 mm × 298 mm），米国ではレターサイズ（216 mm × 279 mm）の用紙の用いられる場合が多い。ページ左右の余白は 10 mm – 15 mm 程度，二段（カラム）の間隔は 7 – 10 mm 程度である。図 (figure) (フィギュア) の幅が 80 mm 程度（約 3.1 inch，約 220 pt）であれば， 1 カラムに収めることができる。２カラム分を使う場合には，図の幅は 170 mm 程度（約 6.7 inch, 約 480 pt）まで使える。

コード (Code 3.2.1) に必要なことを書き加えて (Code 3.2.B.1) のようにして実行すれば，カレントディレクトリに “001.png” , “001.svg”, “001.pdf” という名称の PNG, SVG, PDF ファイルが作成される。

(Code 3.2.B.1) Matplotlib からレポート原稿用の図ファイルを作成する (↩)

```
1) import matplotlib.pyplot as plt # matplotlib.pyplot を plt としてインポートする
2) import numpy as np # numpy を np としてインポートする
3) plt.rcParams['font.family'] = 'Serif' # セリフ体のフォントを使う
4) plt.rcParams['mathtext.fontset'] = 'stix' # 数式には stix フォントを使う
5) plt.rcParams['font.size'] = 9 # フォントサイズは 9 pt
6) x = np.linspace(-4,4,400) # x の範囲を -4 から 4 とし 400 分割する
7) y = np.tanh(x) # 双曲線正接関数値を y とする
8) fig, ax = plt.subplots()  # fig(figure),ax(axes) オブジェクトを使う
9) fig.set_size_inches(3.1,1.8) # 幅 3.1 inch, 高さ 1.8 inch とする
10) ax.plot(x,y,label=r'$y = \tanh x$') # 凡例用ラベルを指定して凡例を設定する
11) ax.set_xlabel(r'$x$') # 横軸ラベルを設定する
12) ax.set_ylabel(r'$y$') # 縦軸ラベルを設定する
13) ax.tick_params(bottom=True,top=True,left=True,right=True) # 上下左右に目盛を
14) ax.tick_params(length=6) # 目盛線の長さを 6pt にする
15) ax.axhline(y=0,linewidth=0.5,c="black") # 原点を通る水平線を描画する
16) ax.axvline(x=0,linewidth=0.5,c="black") # 原点を通る垂直線を描画する
17) ax.legend() # 凡例を表示する
18) fig.tight_layout() # 図の余白を細めに自動設定する
19) plt.margins(x=0) # グラフの左右余白 (margin) は 0 にする
20) plt.savefig('fig01.png',dpi=600) # 図を 600 dpi の PNG 形式で保存
21) plt.savefig('fig01.svg') # 図を SVG 形式で保存する
```

```
22) plt.savefig('fig01.pdf') # 図を PDF 形式で保存する
23) plt.show() # 図を画面表示する
```

(↩)

(Code 3.2.B.1) では，Matplotlib に標準的に搭載されている数式組版(くみはん)システム TeX(テック)（テフ）の機能（の一部）を用いている。

Matplotlib では，軸ラベルや凡例(はんれい) legend(レジェンド) などに用いる文字列に `r'$...$'` の表現を用いれば「...」の部分は TeX での表現と解釈され，例えば `r'$\overline{x}$'` とすれば $\overline{x}$ のように表示される。

Matplotlib に標準搭載される TeX 機能は限られているが，LaTeX(ラテフ)（ラテフ）のインストールされている処理系では，使える機能を拡大させ，さらに多様な表現をできるようになる場合がある。

現行の Microsoft 365 のワープロ・ソフト Word(ワード) と表計算ソフト Excel(エクセル) では文書中にベクタ画像を挿入することはできないが，プレゼンテーション・ソフト PowerPoint(パワーポイント)（パワポ）の場合，［挿入］→［画像］→［このデバイス...］から該当する SVG ファイルを選択すれば，Figure 3.2.B.1 に示すようなベクタ画像を挿入することができる。

Adobe Illustrator は SVG 形式と PDF 形式のファイルを読み込むことができ，線分の末端を丸く加工したり，線分どうしの接続部を丸い形に加工するような追加の編集をし，EPS 形式を含む他のファイル形式で保存をすることもできる。

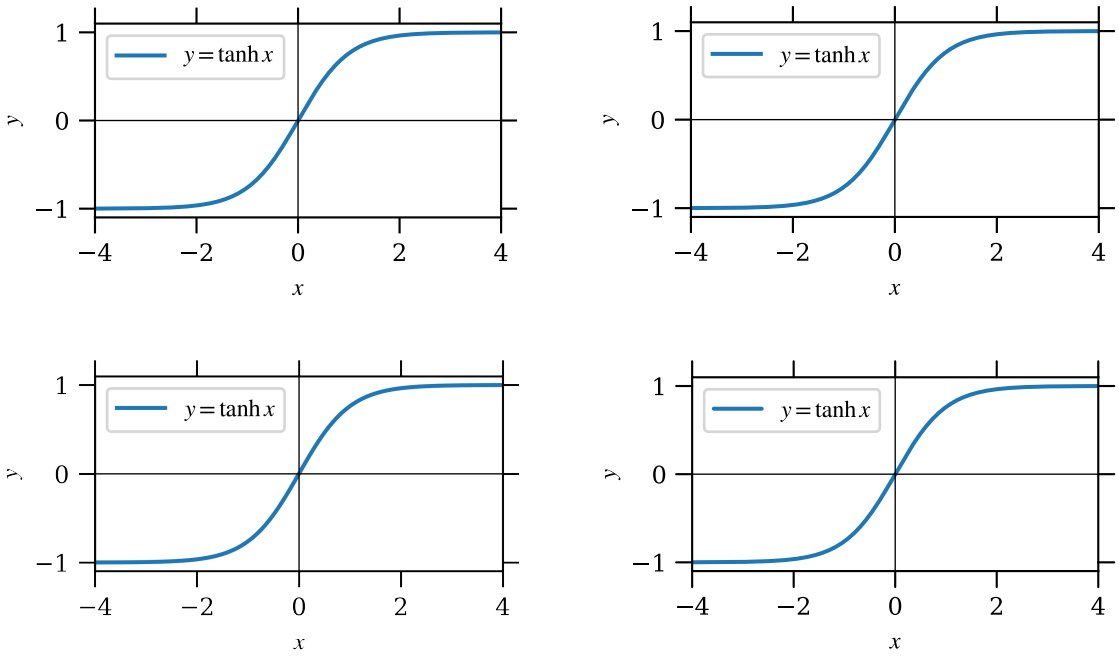

Figure 3.2.B.1 (Code 3.2.B.1) により得られるグラフ（左上：PDF 12 kB, 右上：SVG 16 kB, 左下：600 dpi の PNG 46 kB）と，Adobe Illustrator により線分の末端を丸型に加工したイラスト（右下）(↩)

PDF 形式のファイルは，Windows 11 や macOS を含めた多くのオペレーティング・システムの標準的なファイル管理システム（File Explorer(エクスプローラー) や Finder(ファインダ)）でプレビューを正確に表示できるなど，扱いやすい場面が多い。Apple macOS の標準ワードプロセッサである Pages では文書中に PDF 画像を挿入することも LaTeX での表現を挿入することも容易である。Microsoft Word でも「買い切り型」の LTSC (long term servicing channel)

版ではベクタ画像挿入をできるらしいが，「サブスクリプション型」の Microsoft 365 では，現時点では Word 文書原稿に PDF のようなベクタ画像を挿入できない。（↩）