

熱伝導方程式の数値的解法

森田 太郎

1 はじめに

熱伝導方程式に代表される偏微分方程式はコンピュータにおける数値計算によって計算されることが一般的である。ここでは比較的学習コストが低いと言われている Python を用いて熱伝導方程式を解くことを目的とする。また従来は Anaconda を使うのがベストプラクティスとされていたが、Python を web 開発などの他の業務で使う際はそうではない場合が多いため本稿では Anaconda を使わない¹。

2 開発環境の構築

2.1 Python のインストール

プログラミング言語 Python のインストール方法を示す。既に Anaconda をインストールしている場合は次セクション「はじめのコード」まで呼び飛ばした方が良いでしょう²

1. <https://www.python.org/downloads/> にアクセスし、「Download」ボタンから Python インストーラーをダウンロードする。
2. ダウンロード後にインストーラーを起動し Python をインストール。
3. インストール時に「Add to PATH」にチェックを入れること（重要）

以上でプログラミング言語 Python のインストールは完了した。

2.2 パッケージのインストール

Python だけでもプログラムを書くことはできるが、Python の素晴らしさは先人が作ってきたプログラムをライブラリとして使用できる点である。Python は科学技術計算に強いと言われているが、それは Python には科学技術計算に必要な「Numpy、Scipy」といったライブラリやデータプロットの「Matplotlib」が使用できるからである。ここでは科学技術計算に必要なライブラリをインストールする。

1. windows のスタートから「Windows Powershell」を開く（検索ウィンドウから探すと早い）。
2. 「pip3 install matplotlib numpy scipy」と打ち込み Enter キーを押す。このとき Python インストール時にパスを追加していないとエラーが出る。

2.3 Visual studio code のインストール

Python 自体のインストールはここまでで完了しているが、プログラムの内容（スクリプト）を編集するアプリケーションを用意するのが良い。数あるエディタから今回は比較的導入しやすい Visual Studio Code を紹介する。

1. <https://code.visualstudio.com/download> からインストーラーをダウンロードする。
2. インストールする。

以上で Visual Studio Code のインストールは完了した。

¹数値計算しきしない場合はベスト。Anaconda の numpy は IntelCPU に最適化されている。

²Anaconda と純 Python の共存はなにかと面倒が多い気がする（やったことないのでわからない）

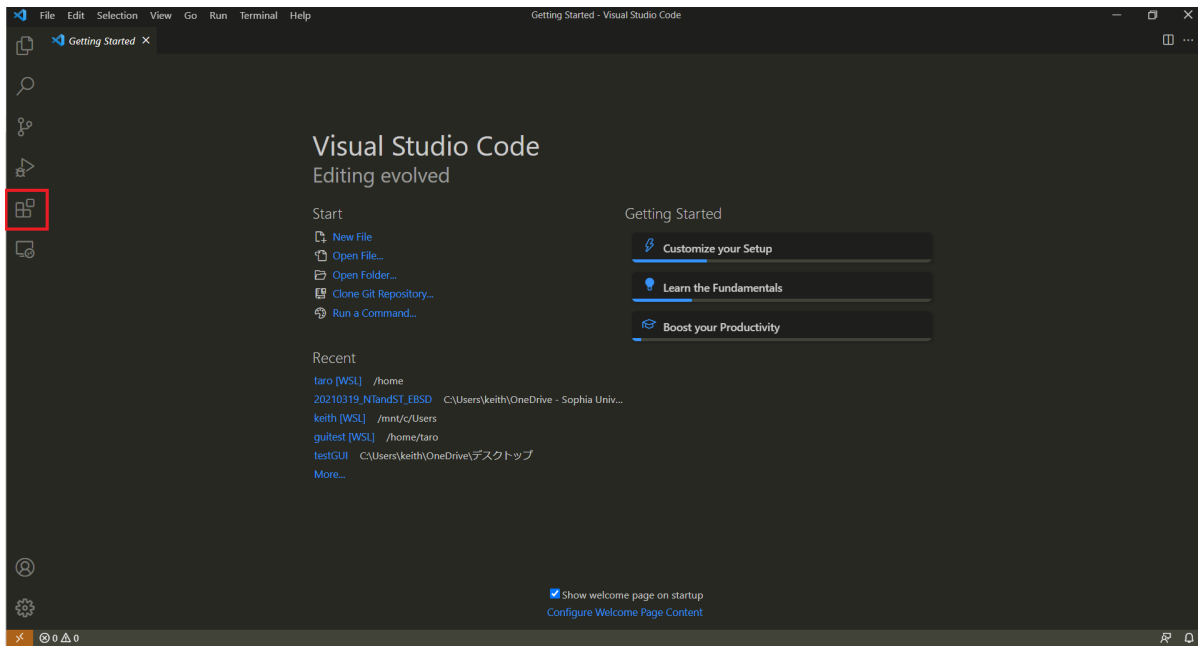


Figure 1: Visual studio code のメニュー画面

2.4 Visual studio code の設定

Visual Studio Code は標準アプリのメモ帳と比較して以下のようなメリットがある。

- コードがシンタックスハイライトされており、可読性が向上している
- 括弧やコードが自動で補完される（自動補完機能）
- Visual Studio Code 内でプログラムを実行できる
- デバック機能が付属している

また Visual Studio Code は Python に限らずほとんど全てのプログラミング言語を扱うことのできる汎用性に優れたアプリケーションである。

Visual studio code のインストールは以上で完了したが、Python を使うには若干の設定が必要である。

2.5 拡張機能のインストール

Visual studio code 上で Python を有効に使用するためには拡張機能をインストールする必要がある。

1. Visual studio code のメニュー画面左側の「Extensions」をクリックし、検索ウィンドウに「Python」と入力し Enter キーを押す。
2. 表一番上の拡張機能をインストールする（写真 2 参照）

これで Visual Studio Code 上で Python が使えるようになった。

3 はじめのコード

まず手始めに画面に「Hello world」と表示されるプログラムを書いてみる（プログラミング言語はまず Hello world から学習を始める）Visual Studio Code（以下 Vscode）の左上「File」－「New File」からファイルを新規作成する。新しくタブが開かれるのですぐに「File」－「Save」でファイル名をつけて保存する。このときファイル名を「hello.py」とすること。一般的にファイルの「.」以降を拡張子と呼び、取り扱うアプリケーションやプログラミング言語によって異なる。Python ファイルの拡張子は「.py」である。ファイルの拡張子を「py」としないと Python 側でファイルを認識しないので注意する。ここではファイル名を「hello.py」として以下のコードを書いてみよう。

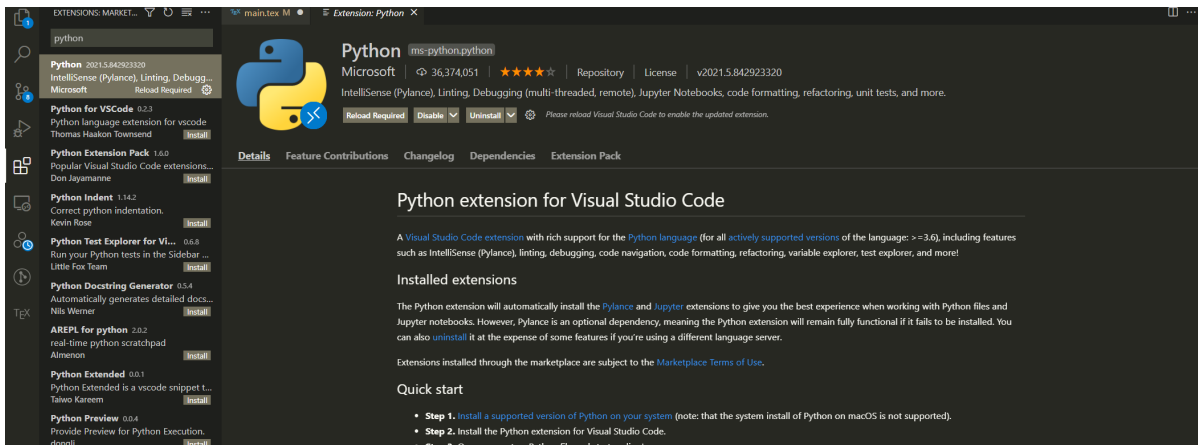


Figure 2: Vscod の拡張機能画面

Listing 1: hello.py

```
1 print("Hello world")
```

コードを書いたらファイルを保存し、VSCode 右上に表示されている緑色の矢印ボタン（再生ボタンみたいな）を押してコードを実行する。画面下に Powershell が開いてコードが実行されて、Hello world と表示されただろうか？ もしエラーが表示されている場合はエラー内容を読んで、コードをもう一度見直してみよう。直したら上記の手順で再度実行しよう。おめでとう！ これで君は Python を使えるようになった！

4 次のステップへ

画面に Hello world を出力するだけでは退屈なので、もう一つステップアップしたプログラムを書こう。ここでは科学技術計算らしく、numpy と matplotlib を用いて Sin プロットを行うプログラムを書く。以下にコードを示す。

Listing 2: sin curve

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3
4 x = np.linspace(0, 10, 1000)
5 y = np.sin(x)
6
7 plt.plot(x, y)
8 plt.show()
```

Sin カーブは出力されただろうか？ 次にコードについて説明する。1 行目、2 行目ではライブラリのインポートを行っている。ここでは数値計算ライブラリの Numpy とデータプロットライブラリの Matplotlib をインポートしている。4 行目では 0 から 10 まで等間隔に 1000 分割した配列 (ndarray) を作成している。Numpy を使わずにこの機能を実装すると割と大変だが、Numpy によって 1 行のコードで実現できる。5 行目では作製した x を変数として Sin を適応した配列を y に代入している。7 行目では Matplotlib の plot 関数を用いて x vs y (Sin カーブ) を出力している。

このように数値計算ライブラリの Numpy を使うことで楽に Sin カーブをプロットすることができた。

5 Python で数値計算の基礎知識

5.1 変数

プログラミングでは「変数に値を代入する」という作業が多い。コードで書くと以下の通りである。

```
1 a = 1
```

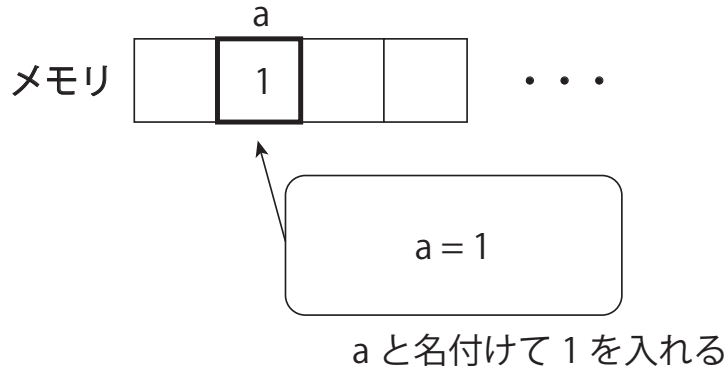


Figure 3: メモリと変数の関係の定性的な説明

これは「a という変数に 1 を代入する」を示す。我々は「a = 1」を目にすると「a は 1 に等しい!」と脊髄反射的に認識してしまうが、プログラミングの世界ではそうではなくあくまで「代入している」。言い換えれば「=」は「代入を示す演算子」である。

変数についてさらに詳しく記す。コンピュータ上にはメモリというデータを保管する装置が存在する。これは「横に連なるロッカー」のような構造をしており、ロッカーの一つ一つにデータを格納することができる。このロッカーのようなものをアドレスというが、変数とはこの「メモリ上のあるアドレス」についた名前を指す。つまり「変数 a に 1 を代入する」とは「メモリ上のあるアドレスに a という名前をつけてそこに 1 を格納する」ことである (図 3 参照)。

6 配列

どのプログラミング言語にも「配列」といったデータを一括に扱う方法が存在する。前述した変数では a = 1 のように 1 つの変数につき 1 つのデータしか格納することができない。この場合、例えば時刻のようなデータを格納する場合は不便であるが、配列を使うことによって多数のデータを 1 つの変数として使うことができる。以下に配列をつかったコードを示す。

Listing 3: array.py

```
1 x = [1, 2, 3]
2 print(x)
```

上記コードを実行すると「[1, 2, 3]」が出力される。このように配列を用いることで x に 1, 2, 3 の値を一括して格納することができた。

配列とメモリの関係を以下に示す。配列を作成するときはメモリ上で配列の大きさ分を確保し、その後に逐次 1, 2, 3 と代入する。つまり「a = [1, 2, 3]」ではまず配列の大きさである 3 マス分をメモリ上で確保しその領域を a と名付け、逐次 1, 2, 3 と代入する (図 4 参照)。

実際に配列を用いて計算するには配列から要素を取り出す必要がある。例えば「x から 1 をとりだして、それに 2 を加えたい」といった処理を行うためには x から 1 を取り出す必要がある。その処理を実現したコードを以下に示す。

Listing 4: array.py

```
1 x = [1, 2, 3]
2 ans = x[0] + 2
```

配列の要素を取り出すときは a[0] のようにして数字で要素を指定する。この数字をインデックスというが、配列の n 番目にある数値を取り出したい場合はインデックスとして n - 1 を指定する。ここで注意すべきなのはインデックスの開始番号は 0 からであるということである³。つまり 1 番目の要素を取り出したい場合はインデックスとして 0 を指定する。10 番目の要素を取り出したい場合はインデックスとして 9 を指定する。

³多くのプログラミング言語ではこの仕様。Matlab は異なる。

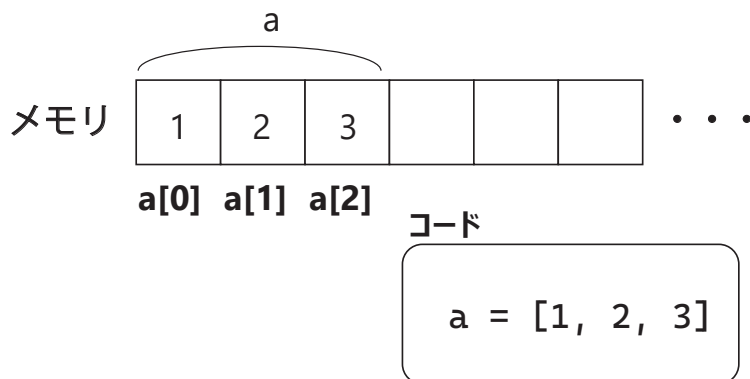


Figure 4: 配列とメモリの関係の定性的な説明

また Python では配列に似た構造としてタプルやリストが存在するが、これらは混同しやすく混乱の元となるため本稿では `numpy` の `ndarray` を配列とする。`numpy` の配列 `ndarray` は以下のようにして定義することができる。

Listing 5: `numpyarray.py`

```
1 import numpy as np
2
3 x = np.array([1, 2, 3])
```

7 Numpy による科学技術計算の初歩

熱伝導方程式の数値計算を始める前に、まず数値計算ライブラリの `Numpy` の基本的な使い方について記す。`Numpy` は Python 上で科学技術計算を容易に行うことを実現したライブラリである。行列計算や統計計算などが簡単なコマンドによって実行することができる。動作原理の知識がなくても扱うことができるが、ブラックボックス化されているためそれがかえってエラーやバグの元にもなる。最悪の場合は論文のデータねつ造につながる可能性がある。以下に配列の平均値を求めるコード比較を行う。 n つの要素からなる $x_1, x_2, x_3 \dots x_n$ の平均値 x_{ave} は

$$x_{\text{ave}} = \frac{\sum_{k=1}^n x_k}{n} \quad (1)$$

コードは以下のようになる。

Listing 6: `average.py`

```
1 import numpy as np
2
3 x = np.array([1, 2, 3])
4 xave = (x[0] + x[1] + x[2]) / len(x)
```

4 行目の `len()` は配列の大きさを得るコマンドである。このコードでは `Numpy` は配列の定義のみに使用している。このコードでは `x` の要素数が 3 つであるため、それぞれを足し合わせることで平均値を求めることができるが、要素数が 100 の場合はこのコードで平均値を求めることはできない。このように上記のコードは `x` の要素数が変わるだけで平均値を求めることができなくなり、普遍的なコードではない⁴。

次に `Numpy` を活用した平均値算出のコードを示す。

⁴このようなスパゲッティコードは書くべきではない（ゴミのようなコードのことをスパゲッティコードという）。

Listing 7: averagenumpy.py

```
1      import numpy as np
2
3      x = np.array([1, 2, 3])
4      xave = np.average(x)
```

ぱっと見でわかりやすいコードになったと思う。さらにこのコードでは x の要素数が変化しても平均値を計算できるコードである。このように Numpy ライブラリを用いることで科学技術計算が楽になる。しかし、平均値の算出方法を知らないのに `average()` を使うのは良くないことがなんとなく想像できると思う。原理を理解した上で使うように心がけよう。

7.1 行列計算