DxMT 東京科学大学拠点 D²MatE 2026年度チュートリアルシリーズ
東京科学大学 総合研究院 フロンティア材料研究所学術講演会

生成AIとToyプログラムで理解するバンド理論・データ解析

第１回第２部　生成AIでプログラムを作る

神谷利夫

東京科学大学
総合研究院
元素戦略MDX研究センター

Institute of SCIENCE TOKYO

本チュートリアルには、
Microsoft365 Copilot / Chat GPT / Gemini / Claude等
を使用した結果が含まれています

本講義の方針

- 計算や解析の原理は説明しません
- 生成AIでプログラムを作る手順がわかるように説明します
- 「失敗例」への対処法を説明するため、
  過去の作成例を中心に説明します
- 「作ったプログラムを動かす」までを目標にします
  プログラムの改善、動作検証などは 第2回で説明します

- Pythonとエディタをインストールしてある場合は、
  配布したプログラムで実習できます。
  ただし、詳しい実行手順の説明は省略します
  行頭の「>」は入力例の目印です。
  「>」を含めず、その後のコマンドをターミナルにコピーして実行してください。

AI時代の研究者:
コードを書く能力から、コードの正しさを保証する能力へ

AI時代のトップエンジニアは「自分でコードを書く人」から
「巨大なコード生成システムを指揮する人」へ変化する

昔　　: 1000行書ける人

今　　: 10000行をレビューできる人

AI時代: 100万行をAIに生成させ、その正しさを保証できる人

保証されていない「動いているように見える」プログラムはテロ行為

研究も・・・

昔　　: たくさん実験できる人

今　　: たくさんのデータを解析できる

AI時代: 大量の結果の中から、本当に正しいものを見抜ける人

目的に応じてAIコーディングの方法を選ぶ

1 小さく一般的な処理



- Web UI
- 軽量・高速モデル
- 実行しながら修正

2 科学計算・専門処理



- 高推論モデル
- 独立した基準で検算
- 結果の妥当性を確認

3 多数のファイル・ライブラリ



- AIエージェント
- 要件と受入条件を定義
- 計画・実装・テスト

4 既存コードの局所修正



- AI支援エディタ
- 関連コードを参照
- リポジトリを正本にする

規模・専門性・検証の重要度に応じて組み合わせる

この講義ではWebUIを使ってプログラムを開発します

理由

- インストール不要で使える (アカウント作成は必要)
- 簡単なプログラムであれば十分。
  AIエージェントでも、プロンプトの入れ方は同じ。
- AI支援コードエディタ (VSCodeなど) で局所的に修正するより、
  WebUIで全コードを出力しなおすほうが早い
  2000行程度のコードであれば、人間が手動で数行直すよりも
  早く出力してくれる
- AIエージェントの機能を十分に使うには、
  事前に、仕様やプロンプトを設計する必要がある。
- AIエージェントをいい加減な設計で使うと、
  想定外に大きな無駄課金や時間が発生する

簡単なプログラムの例: Excelファイルのマージ

事務処理プログラム: サンプルデータも生成AIで作成

Q: data1.xlsxとdata2.xlsxを読み込み、snをキーとしてデータを結合してmerged.xlsxに保存するプログラムを作って

A: merge_excel.py

Q: サンプルのdata1.xlsxとdata2.xlsxも作って

A: make_excel_data.py
を実行してdata1.xlsx, data2.xlsxを作製

data1.xlsx

sn	name	height
1001	Suzuki	170
1002	Tanaka	165
1003	Sato	180
1004	Kato	172

data2.xlsx

sn	weight	age
1001	65	21
1003	75	24
1004	68	22
1005	70	23

> python merge_excel.py

saved: merged.xlsx

rows : 3

merged.xlsx

sn	name	height	weight	age
1001	Suzuki	170	65	21
1003	Sato	180	75	24
1004	Kato	172	68	22

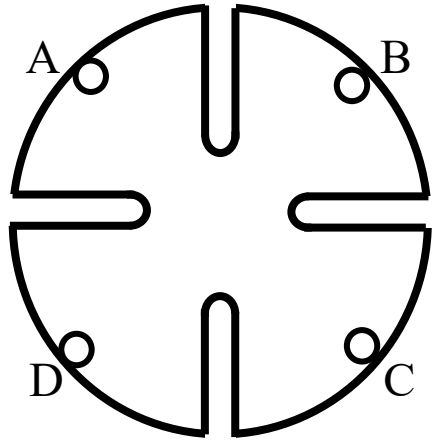
簡単なプログラムの例: 1変数方程式の解を求める

バイブコーディング

Van der Pauw法

L.J. van der Pauw, Philips. Res. Rep. **13** (1958) 1.

電極AB間に電流I_{AB}を流し、電極CD間の電圧V_{CD}を測定：$R_{AB,CD} = V_{CD} / I_{AB}$
電極BC間に電流I_{BC}を流し、電極DA間の電圧V_{DA}を測定：$R_{BC,DA} = V_{DA} / I_{BC}$



$$\rho = \frac{\pi d}{\ln 2} \cdot \frac{(R_{AB,CD} + R_{BC,DA})}{2} \cdot \boldsymbol{f(R)}$$

$f(R)$: 形状補正係数: $R = R_{AB,CD}/R_{BC,DA}$

$$\exp\left(-2\frac{\ln 2}{f(R)}\frac{1}{R+1}\right) + \exp\left(-2\frac{\ln 2}{f(R)}\frac{R}{R+1}\right) - 1 = 0$$

Van der Pauw 法における形状補正係数 f

$R_{AB,CD} / R_{BC,DA}$	f	$R_{AB,CD} / R_{BC,DA}$	f
1.0	1.0	1.4	0.9903
1.1	0.9992	1.5	0.9860
1.2	0.9971	2.0	0.9603
1.3	0.9941	3.0	0.9067

プロのプロンプト例: 仕様を与えてプログラムを作る

以下の仕様に基づいて、完成したプログラムのみを出力してください。
説明文、思考過程、擬似コード、段階的な構築は不要です。
最終的なコードだけを、1つのファイルとして成立する形で出力してください。

【目的】
（ここに作りたいプログラムの目的を書く）

【仕様】
- （仕様1）
- （仕様2）
- （仕様3）
  …

【要件】
- コードのみを出力すること
- コメントは必要な範囲で付けること（または付けない）
- 外部ライブラリの使用可否を指定（例：numpy, scipy, matplotlib 使用可）
- CLI 化の有無（argparse を使うかどうか）
- 入出力形式（CSV, 標準入力, 引数など）
- エラー処理の有無
- 再現性のための random seed の扱い（必要なら）

【禁止事項】
- 説明文を出力しない
- 擬似コードを出力しない
- ステップバイステップの解説を出力しない
- コード以外の文章を出力しない

でも、研究で、細かい仕様が最初に決まっていることは稀です。
プログラムを作りながら改善していきます

バイブコーディングでプログラムをつくる

バイブコーディング: 最近使われている言葉

例えるなら: ノリ (vibe) でコードを書き、生成AIの回答を受けてさばき、修正と追加を繰り返しながらコードを磨いていくやり方

ただし: 汚いコードができたらただの「行き当たりばったり」
良いコードを作るように心がけよう

バイブコーディングのサイクル:

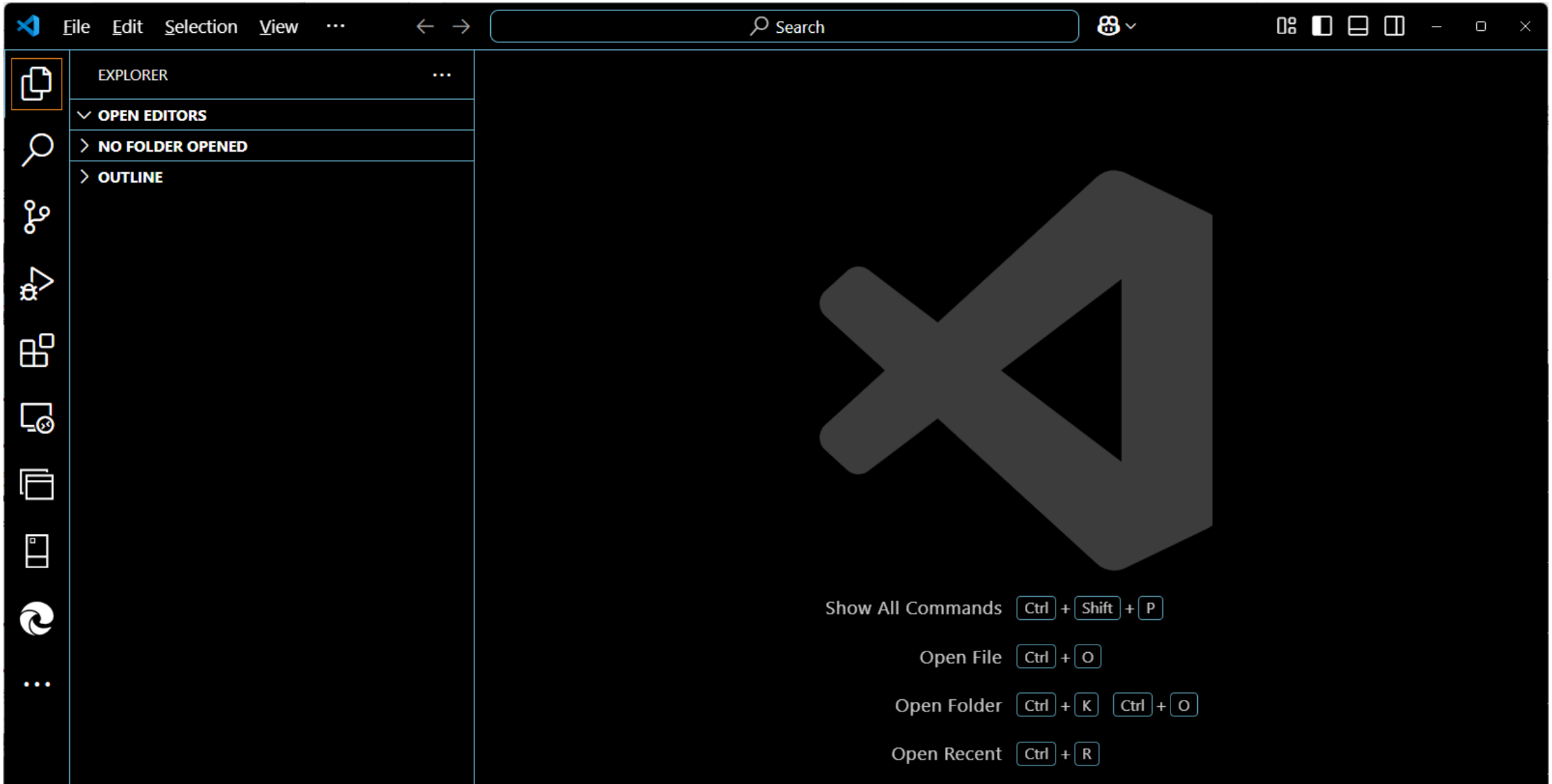
1. 「XXXのプログラムを作って」から始める
2. 回答のプログラムを動かしてみる。
   - エラーがでたら「エラーが出たので修正してください (エラーメッセージ)」と返す
   - うまく動いたら、使いやすくなるように機能追加
     「起動時引数で制御できるように」「パラメータを増やして」
     「グラフを描いて」「リファクタリングして」 など。
3. 2 を、満足いくコードができるまで繰り返す



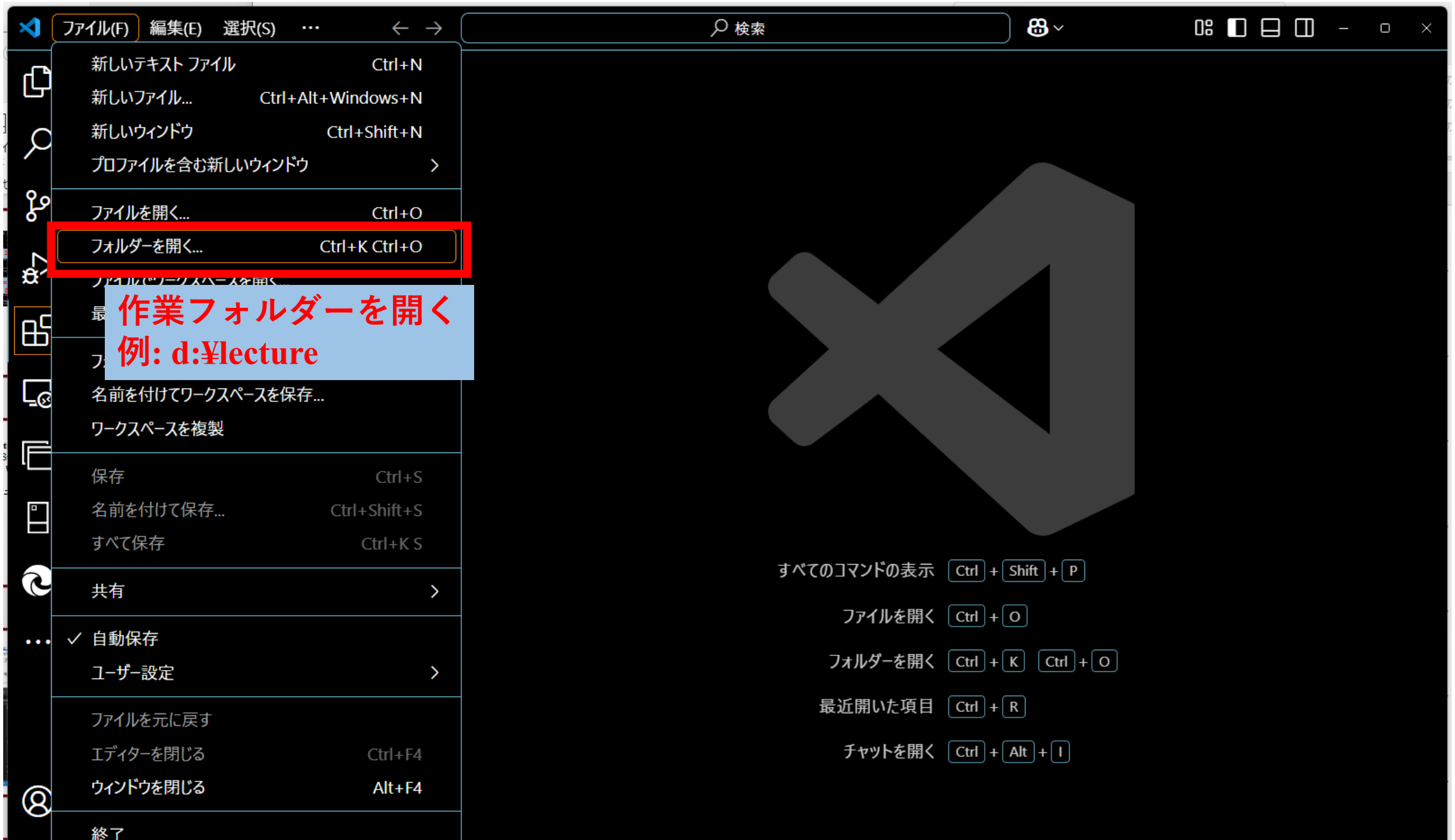
ChatGPT5Instantでのプロンプト:
「ふるえるぞハート！燃えつきるほどヒート！！ 刻むぞ血液のビート！」みたいな雰囲気で、バイブコーディングの雰囲気を伝える絵

Visual Studio Code (VSC)の起動

実行方法: Windowsのタスクバー “検索”ボックスやターミナルから
“code” と入力して実行



作業フォルダーに移動


ファイル(F)
編集(E)
選択(S)
検索
新しいテキスト ファイル Ctrl+N
新しいファイル... Ctrl+Alt+Windows+N
新しいウィンドウ Ctrl+Shift+N
プロファイルを含む新しいウィンドウ
ファイルを開く... Ctrl+O
フォルダーを開く... Ctrl+K Ctrl+O
作業フォルダーを開く
例: d:¥lecture
名前を付けてワークスペースを保存...
ワークスペースを複製
保存 Ctrl+S
名前を付けて保存... Ctrl+Shift+S
すべて保存 Ctrl+K S
共有
自動保存
ユーザー設定
ファイルを元に戻す
エディターを閉じる Ctrl+F4
ウィンドウを閉じる Alt+F4
終了
すべてのコマンドの表示 Ctrl + Shift + P
ファイルを開く Ctrl + O
フォルダーを開く Ctrl + K Ctrl + O
最近開いた項目 Ctrl + R
チャットを開く Ctrl + Alt + I

プロンプトとコード出力

Q: van der Pauw法で抵抗率を補正する形状補正因子f(Rab,cd/Rad,bc)を計算し、Rab,cd/Rad,bcに対しプロットするプログラムを作ってください

Gemini

チュートリアル資料用にコードを出力してください。私の他のチャットの内容やプロフィールは、このチャット内では忘れてください

van der Pauw法で抵抗率を補正する形状補正因子...

van der Pauw法において、試料の形状や電極配置の不完全さを補正するための係数 f を求めるプログラムですね。

この係数 f は、以下の超越方程式（ファン・デア・パウの方程式）を解くことで得られます。

$$\exp\left(-\frac{\pi R_a}{R_s}\right)+\exp\left(-\frac{\pi R_b}{R_s}\right)=1$$

ここで、抵抗の比を $Q=R_a/R_b$ と置くと、補正因子 f は以下の関係式を満たすように定義されます。

$$\frac{Q-1}{Q+1}=f\cdot\operatorname{arccosh}\left(\frac{\exp(\ln 2/f)}{2}\right)$$

以下に、Pythonを用いた計算とプロットのコードを記述します。

Pythonによる実装コード

このコードでは、`scipy.optimize` の `fsolve` を使用して数値的に f を算出します。

コードは “コードブロック形式” で出力される

Pythonによる実装コード

このコードでは、`scipy.optimize` の `fsolve` を使用して数値的に f を算出します。

Python

コピーアイコンをクリック
=>エディタへ貼り付け

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

def vdp_equation(f, Q):
    """
    van der Pauwの補正因子fを求めるための超越方程式
    f: 形状補正因子
    Q: 抵抗比 (Ra/Rb)
    """
    # 数値計算の安定性のた、f=0付近でのエラーを回避
    if f <= 0: return 1e10

    left_side = (Q - 1) / (Q + 1)
    # arccoshの引数が1未満にならないよう調整
    term = np.exp(np.log(2) / f) / 2.0
    right_side = f * np.arccosh(term)

    return left_side - right_side

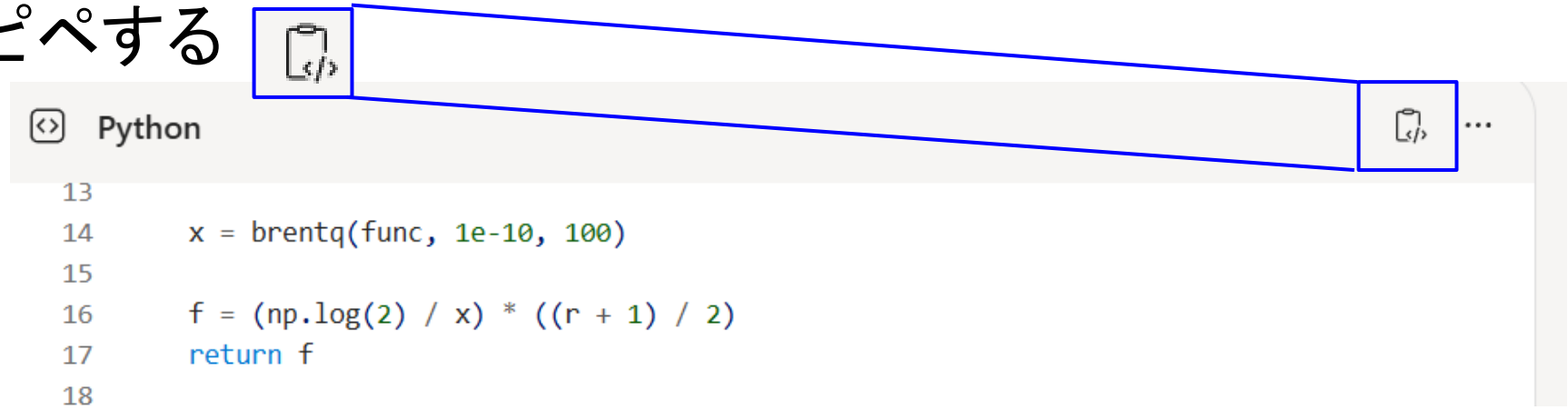
def calculate_f(Q):
    """抵抗比Qに対してfを解く"""
    # 初期値は通常1に近い値からスタート
    f_initial_guess = 1.0
    f_solution = fsolve(vdp_equation, f_initial_guess, args=(Q,))
    return f_solution[0]

# 1. 抵抗比 Q (Ra/Rb) のデータ範囲を作成 (1から100まで)
```

注意: コードの出力形式はころころ変わるので柔軟に対応

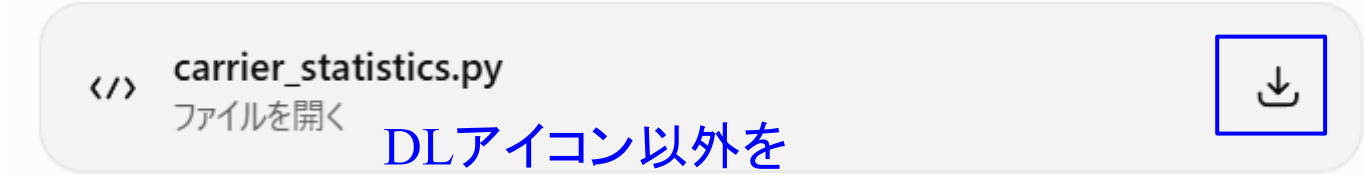
典型的な出力例

・コードブロック形式: コピペする



・ファイルリンク:

・Previewを開いてダウンロード／コピー



マウスカーソルを乗せると
DLアイコンがでる

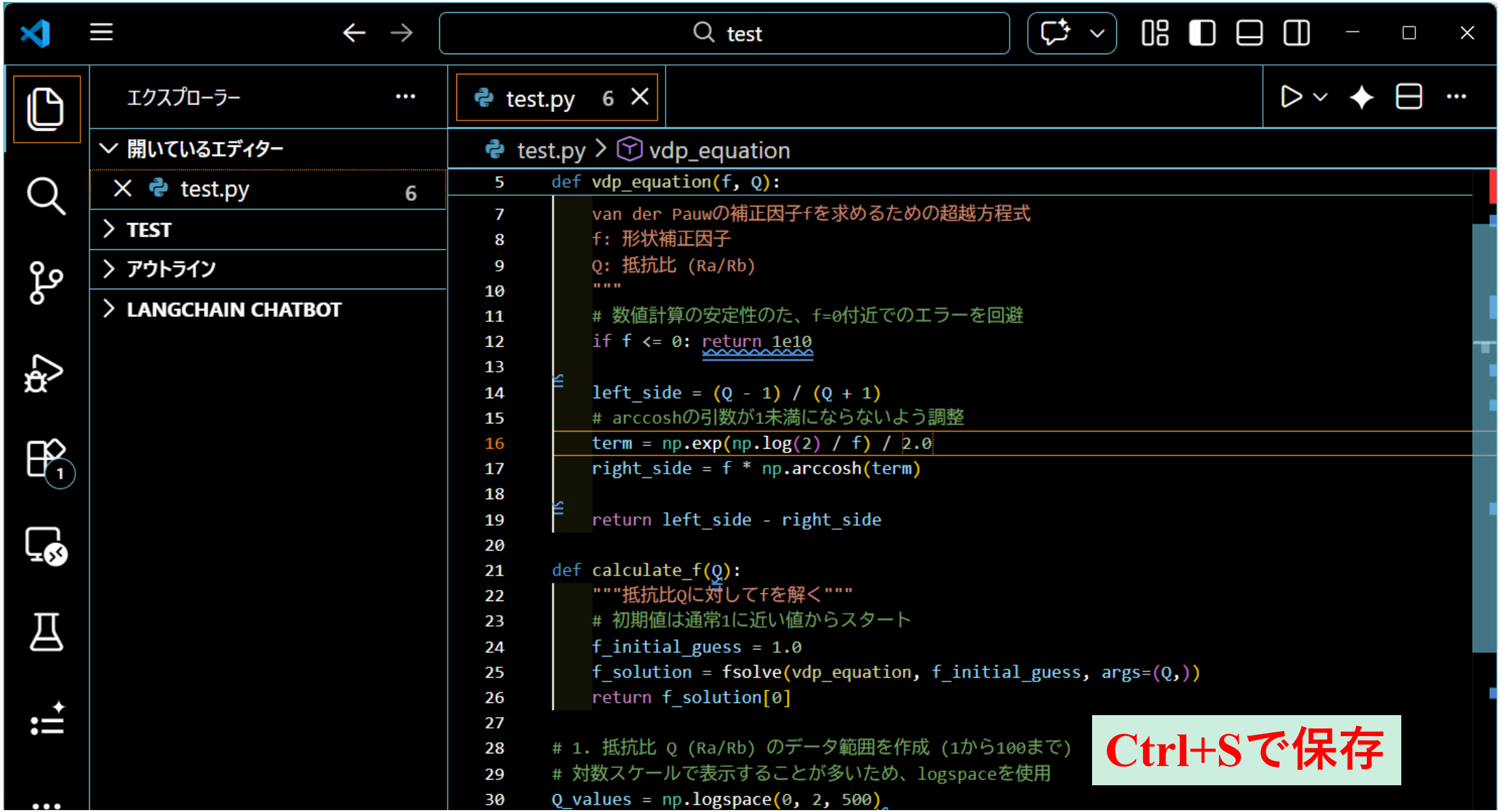
DLアイコン以外を
クリックするとPreview

・リンクからダウンロード

test.pyファイルを作る



VSCodeへ貼り付け



.pyファイルを実行する: python.exeの選択



実行してもうまくいかない

```python
def vdp_equation(f, Q):
    van der Pauwの補正因子fを求めるための超越方程式
    f: 形状補正因子
    Q: 抵抗比 (Ra/Rb)
    """
    # 数値計算の安定性のた、f=0付近でのエラーを回避
    if f <= 0: return 1e10

    left_side = (Q - 1) / (Q + 1)
    # arccoshの引数が1未満にならないよう調整
    term = np.exp(np.log(2) / f) / 2.0
    right_side = f * np.arccosh(term)
```

問題 12　デバッグ コンソール　ターミナル　ポート　クエリ結果

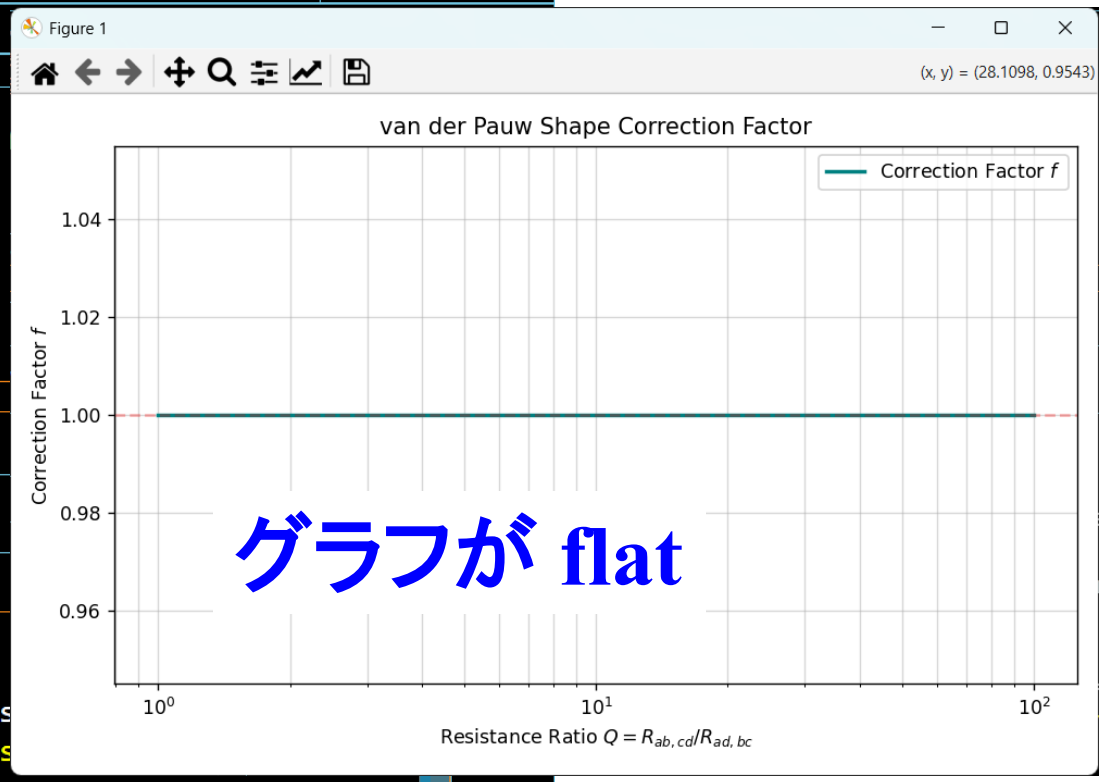
出力

フィルター

ターミナル

```
  making good progress, as measured by the
    improvement from the last ten iterations.
    f_solution = fsolve(vdp_equation, f_initial_guess
PS G:\test> & C:\Users\tkami\AppData\Local\Programs
.exe g:/test/test.py
g:\test\test.py:17: RuntimeWarning: invalid value encountered in arccosh
  right_side = f * np.arccosh(term)
g:\test\test.py:25: RuntimeWarning: The iteration is not making good progre
ss, as measured by the
  improvement from the last ten iterations.
  f_solution = fsolve(vdp_equation, f_initial_guess, args=(Q,))
PS G:\test>
```

Warningがでてる

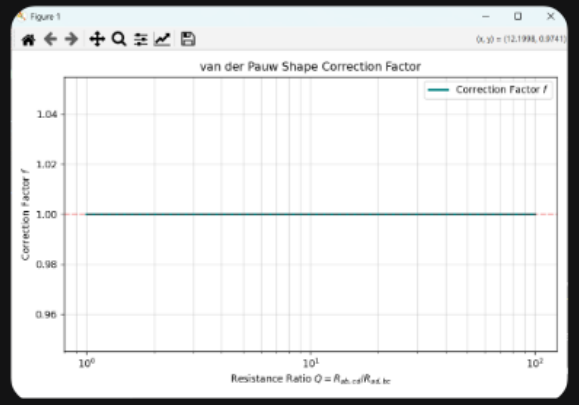


グラフが flat

行 16、列 39　スペース: 4　UTF-8　CRLF　{ } Python　Finish Setup　Go Live

実行してもうまくいかない

グラフとメッセージを張り付けて修正を依頼



Warniが出たうえ、添付の画像のように、f(R)=1になってしまい、計算結果がおかしいです

g:\test\test.py:17: RuntimeWarning: invalid value encountered in arccosh...

分析

ご提示いただいた画像とエラーメッセージを確認しました。

`RuntimeWarning: invalid value encountered in arccosh` が出ていることから、程で `arccosh` の引数が **1未満** になってしまい、実数解が得られず数値計算が停止（期値の 1.0 に固定）してしまっています。

van der Pauwの方程式は f が小さくなるほど数値的に不安定になりやすいため、`fso` 索範囲や初期値、あるいは方程式の記述形式を少し工夫する必要があります。

修正版のコードでは、より安定して解を導けるように以下の点を変更しました。

修正版：van der Pauw 補正因子計算プログラム

`fsolve` の代わりに、よりロバストな（頑健な）境界指定解法である `bisect`（二分法）を使用するか、方程式の形を `arccosh` を使わない形式に書き換えるのが定石です。

Python

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import bisect

def vdp_target(f, Q):
    """
    van der Pauwの方程式を exp を用いた基本形式で定義
```

このように、

1. プログラム生成
2. 実行
3. エラー修正

の2～3を繰り返しながら、正常に動くプログラムを作っていきます

この後は、
既に作ってあったプログラムを見ながら進めていきます

プロンプト

チュートリアルの都合により、以下のプロンプトを使いました

Q: van der Pauw法で抵抗率を補正する形状補正因子f(Rab,cd/Rad,bc)をNewton法で計算し、Rab,cd/Rad,bcに対しプロットするプログラムを作ってください

便利ですが、禁じ手です
- **危険すぎるので、数式は与えましょう**
- **数式は画像で与えてもOK**
- **教科書等と数式・結果を確認しましょう**

A: 数式は勝手に推測して vdPauw_f.py を作った

```
from scipy.optimize import newton

def vdp_residual(f, r):
    ln2 = np.log(2.0)
    term1 = np.exp(-(ln2 / f) * (2 * r / (r + 1)))
    term2 = np.exp(-(ln2 / f) * (2 / (r + 1)))
    return term1 + term2 - 1.0

def solve_f_sequence(r_list):
--cut--
        sol = newton(vdp_residual, x0=current_f, fprime=vdp_derivative,
                     args=(r,), tol=1e-12, maxiter=100)
```

$$\exp\left(-2\frac{\ln 2}{f}\frac{1}{R+1}\right)+\exp\left(-2\frac{\ln 2}{f}\frac{R}{R+1}\right)-1=0$$

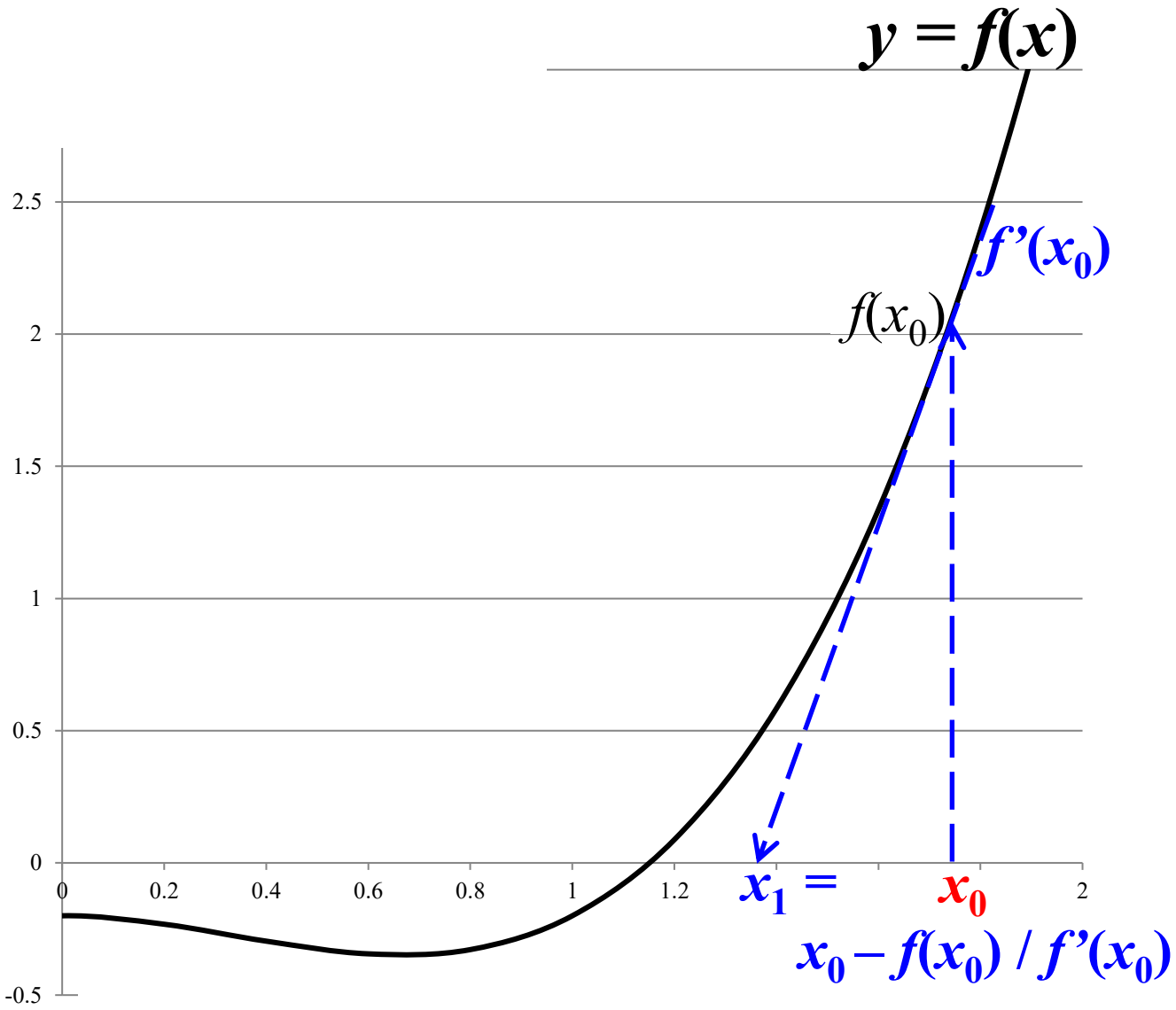
超越方程式の根: Newton法

$f(x) = 0$ を解きたい

1. 根の推定値 x_0
2. x_0+dx が根であるとする
   $f(x_0+dx) = f(x_0) + dx\, f'(x_0) \sim 0$
3. $x_1 = x_0 + dx = x_0 - f(x_0) / f'(x_0)$
   がより良い根であろう
4. x_0 をx_1 に置き換え、
   x_1 が x_0 に十分近くなるまで
   2~3を繰り返す

解を安定させるために
Damping Factor λ を使うことが一般的

$$x_{k+1} = x_k - f(x_k) / f'(x_k) / (1 + \lambda)$$



実行してみる

実行
> python vdPauw_f.py

出力: エラー発生

```
Traceback (most recent call last):
  File "vdPauw_f.py", line 30, in <module>
    f_values = [solve_f(r) for r in r_values]
                ^^^^^^^^^^
  File "dPauw_f.py", line 25, in solve_f
    return newton(vdp_f_func, x0=0.9, args=(r,))
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "(cut)AppData¥(cut)¥Python312¥Lib¥site-packages¥scipy¥optimize¥_zeros_py.py", line 391, in newton
    raise RuntimeError(msg)
RuntimeError: Failed to converge after 50 iterations, value is 0.9001915296267972.
```

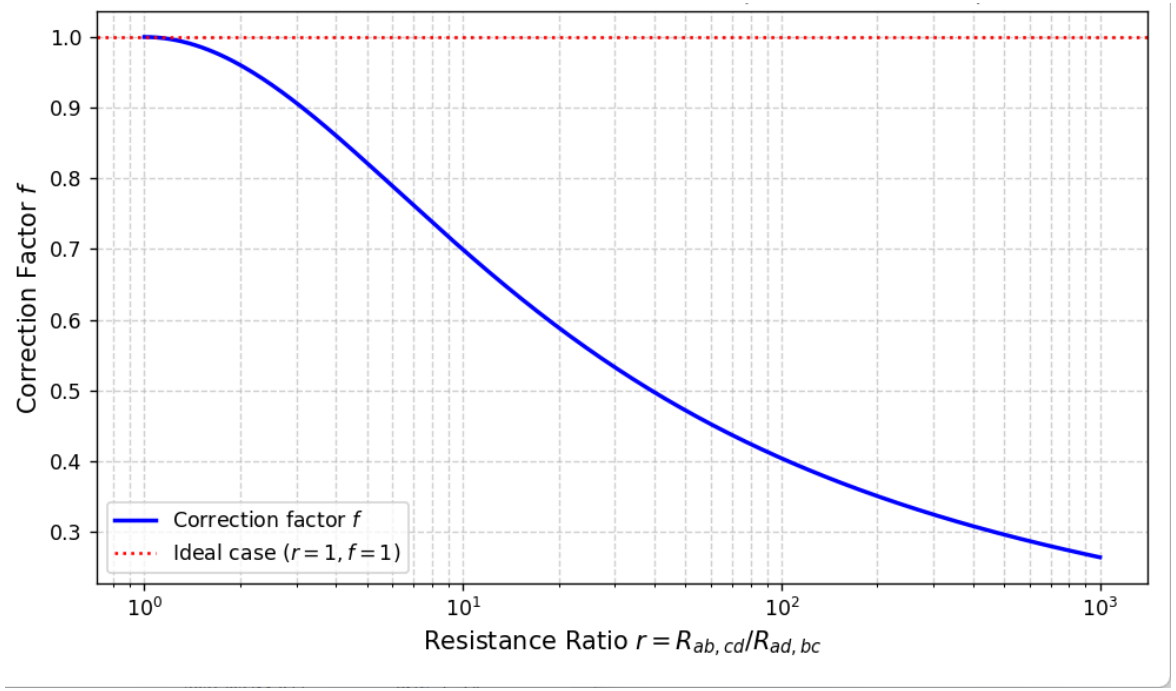
修正プロンプトと動作確認

Q: エラーです。

Traceback (most recent call last):
（Tracebackメッセージを張り付け）

あるいはもっと具体的に。

Q: 収束エラーです。R=1でf=1が解なので、
これを初期値に順次Rを変えると安定して解けませんか
（Tracebackメッセージを張り付け）



A: vdPauw_f.py

```
from scipy.optimize import newton
def solve_f_sequence(r_list):
--cut--
    sol = newton(vdp_residual, x0=current_f,
                 fprime=vdp_derivative,
                 args=(r,), tol=1e-12, maxiter=100)
```

実行結果

> python vdPauw_f.py

✅ 計算結果を 'vdp_f_results.csv' に保存しました。
✅ プロット図を 'vdp_f_plot.png' に保存しました。

Rとfの値が合っているか、Excelで確認

vdPauw_check.xlsx

R	f	exp(-2ln(2)/f/(R+1))+exp(-2ln(2)/f*R/(R+1))-1
1	1	0
1.1	0.9992	-9.45694E-06
10	0.7	0.000473989
100	0.4	-0.001391231
1000	0.27	0.000804576

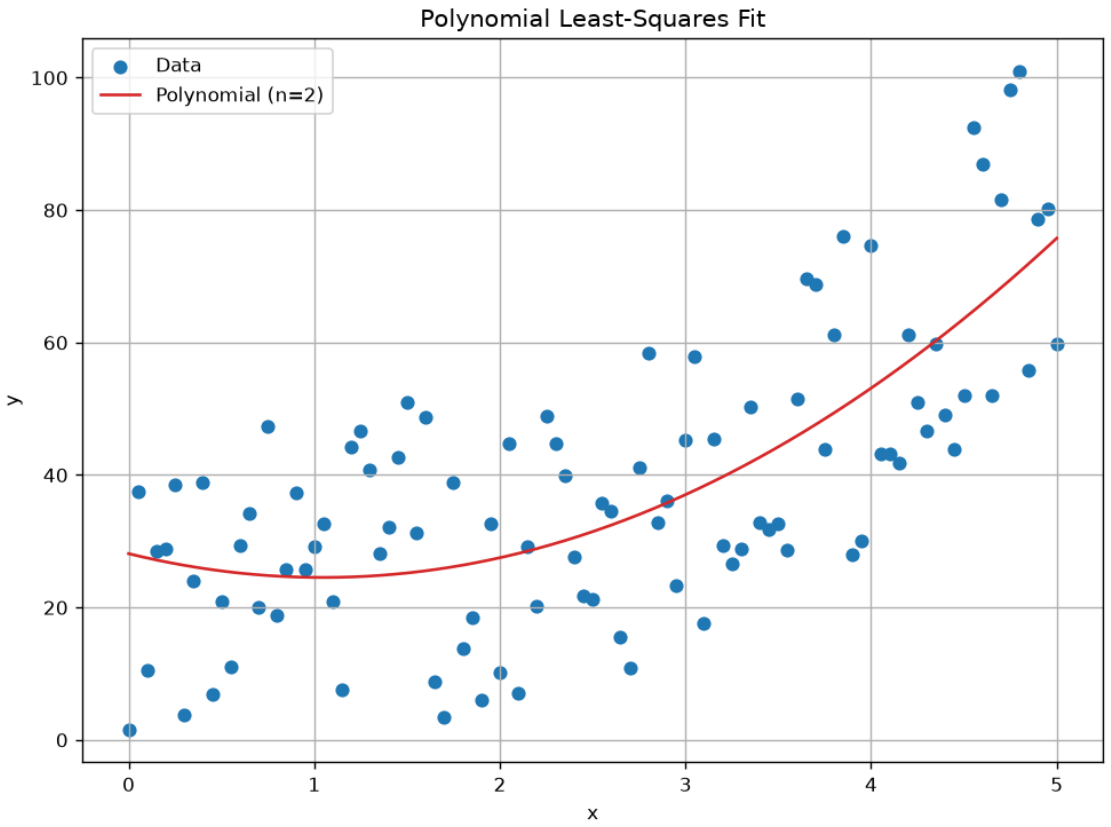
$$\exp\left(-2\frac{\ln 2}{f}\frac{1}{R+1}\right)+\exp\left(-2\frac{\ln 2}{f}\frac{R}{R+1}\right)-1=0$$

簡単なプログラムの例: m次多項式最小二乗法

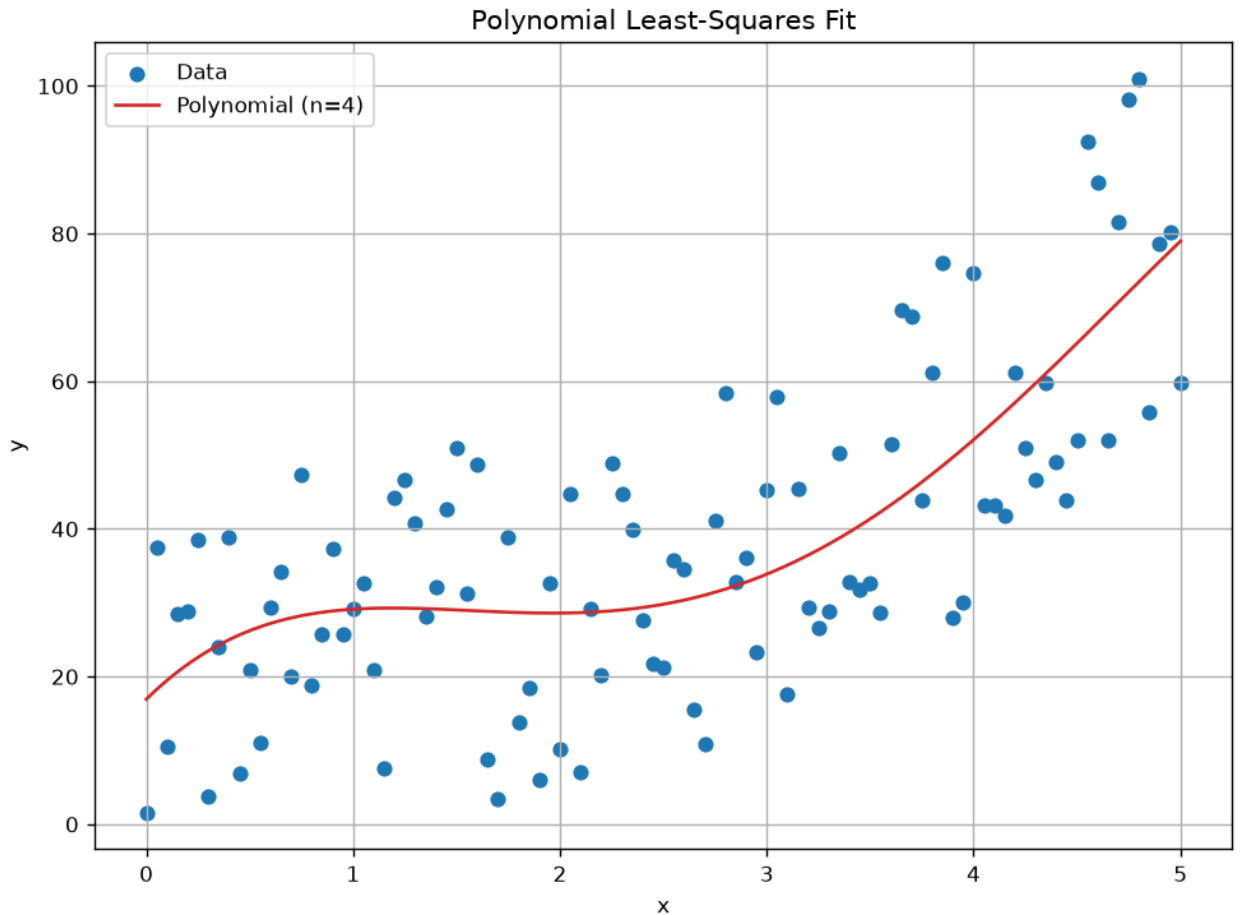
プログラム作成: m次多項式最小二乗法

プロンプト: python lsq_polynomial.py random-poly.xlsx 2
のように、データファイル名と多項式の次数を与えて多項式最小二乗法をするプログラムを作って

python lsq_polynomial.py random-poly.xlsx 2



python lsq_polynomial.py random-poly.xlsx 4



プログラムの説明を生成AIに依頼

lsq_polynomial.py ファイルをプロンプトに投入するか、コピペして
プロンプト: 初心者にもこのプログラムの動作がわかるように詳しく説明して
（コードを張り付け）

回答:

実行方法

例えば

Shell

```
python lsq_polynomial.py random-poly.xlsx 2
```

と実行すると、

- random-poly.xlsx を読む
- 2次多項式

$$y = c_0 + c_1 x + c_2 x^2$$

でフィットする

という意味になります。

全体の流れ

Plain Text

```
Excel読込
    ↓
x, y データ取得
    ↓
polyfitで最小二乗法
```

4. polyfitによる最小二乗法

このプログラムの核心です。

Python

```
coeffs = np.polyfit(x, y, degree)

```

NumPyに

> xとyをdegree次多項式で最小二乗フィットしてください

とお願いしています。

例えば

Python

```
degree = 2
```

なら

$$y = c_2 x^2 + c_1 x + c_0$$

の係数を返します。

ピークフィットプログラムの作成例

Q1:ピークフィットのプログラムを作ってください
A1: peakfit01.py
Q2:
#初期パラメータをコンソールに出力して
#最適化パラメータをコンソールに出力して
#入力データ、初期パラメータのグラフ、最適化パラメータのグラフを比較して
A2: peakfit02.py
Q3: # 初期値をargparseでうけとれるようにして
A3: peakfit03.py
Q4: # プログラム終了時に usageを表示して
A4: peakfit04.py
Q5:# フィッティング結果の誤差を計算して表示して
A5: peakfit05.py

実行例

Q6: 初期値が悪いと、フィッティング結果も悪いです。もっとよいアルゴリズムは使えますか
A6: 初期値探索を使うと回答してきたので、ここでは却下
Q7: 広域安定なアルゴリズムを使えませんか
A7: 修正コードしか表示してくれなかった
Q8: 全コードを出力してください
A8: peakfit8.py => エラー
Q9: エラーです

```
  File "D:¥peakfit08.py", line 88, in peak_fit
    print(f"b0_0   = {p0[3]:.5f}")
                      ^^^^^^^^^^^
TypeError: unsupported format string passed to NoneType.__format__
```

A9: 回答に従って peakfit8.py を修正

実行例

Q10: フィッティング過程を可視化できますか。パラメータの変化、誤差の変化をコンソールに出して、繰り返しcycleごとにグラフもupdate
A10: .curvefit()ではなく、.minimize()を使うことを提案
peakfit10.py => 動きました

Q11:このプログラムでは、どのようなアルゴリズムが使えますか
A11: .minimize()の method で指定できるアルゴリズム

アルゴリズム	Method	収束範囲	速度	初期値依存
Nelder–Mead	nelder-mead	△	△	中程度
BFGS / L-BFGS-B	bfgs	○	○	中程度
Conjugate gradient	cg	○	○	中程度

他の例: テキスト読み上げプログラム

テキスト読み上げプログラム: Edge TTS (Text-To-Speech)

状況説明: Microsoft Edge使っているTTSは、クラウドサービスとして無料で使えます。

Q: Edge TTSを使って読み上げるプログラムを作って。引数で与えた文字列が存在するファイル名ならそれを読み込み、そうでないならその文字列を読み上げて

A:

ライブラリのインストール: > pip install edge-tts

プログラム: edge_tts.py

実行したらエラーが出ました。

```
>python edge_tts.py こんにちは
Read text from command line
Traceback (most recent call last):
(略)
  File "edge_tts.py", line 22, in speak
    communicate = edge_tts.Communicate(
                  ^^^^^^^^^^^^^^^^^^^^
AttributeError: module 'edge_tts' has no attribute 'Communicate'
```

エラー修正のプロンプト

Q: エラーが出ました。

```
>python edge_tts.py こんにちは
Read text from command line
Traceback (most recent call last):
  File "edge_tts.py", line 76, in <module>
    main()
    ~~~~^^
  File "edge_tts.py", line 72, in main
    asyncio.run(speak(text))
    ~~~~~~~~~~~^^^^^^^^^^^^^
  File "C:¥Users¥tkami¥AppData¥Local¥Programs¥Python¥Python313¥Lib¥asyncio¥runners.py", line 196, in run
    return runner.run(main)
           ~~~~~~~~~~^^^^^^
(略)
  File "edge_tts.py", line 22, in speak
    communicate = edge_tts.Communicate(
                  ^^^^^^^^^^^^^^^^^^^^^
AttributeError: module 'edge_tts' has no attribute 'Communicate'
```

A: プログラム名が edge_tts.py と、ライブラリ名 edge_tts と同じために、このプログラムをimportしてしまったため。

対策: プログラム名を edge_tts_sample.py に変更

> python edge_tts_sample.py こんにちは

Read text from command line

メディアプレイヤーが立ち上がって音声がでました

注: 引数無しで実行すると Usage を表示しますが、プログラムファイル名がハードコーディングされているため、”edge_read.py” となっています

> python edge_tts_sample.py

Usage:

python edge_read.py "こんにちは"

python edge_read.py sample.txt

注意: 読み上げる文章は外部サーバーに送信されます

より実用的なプログラムの作成例

チュートリアル: http://d2mate.mdxes.iir.isct.ac.jp/D2MatE/?page=tutorial&id=programming

source HTML | display HTML

- 2026材料データ × 計算材料科学 ハンズオン
- 2026チュートリアルコース

- 学会等講演
- プレゼンテーション
- 生成AI
- プログラミング
- 数値解析
- 結晶・回折・干渉
- 結晶・回折・干渉(内部限定)
- 統計力学
- 第一原理計算
- 物理・科学
- 半導体デバイス
- 半導体デバイス(内部限定)
- データ科学
- 科学史

▶ pythonチュートリアル: Debug ー printデバッグ

▶ pythonチュートリアル: Debug ー デバッガ

▶ Vibeコーディング (難易度★): van der Pauw法の形状補正因子を例に

▶ Vibeコーディング (難易度★): 非線形最小二乗法

▶ Vibeコーディング(難易度★★★): 散乱モデルによる移動度の温度依存性のフィッティング (線形最小二乗法、非線形最小二乗法)

▶ Vibeコーディング (難易度★★★★): 半導体のキャリア密度の温度依存性のフィッティング (非線形最小二乗法)

▶ Vibeコーディング (難易度★★★★★): 薄膜XRDの非線形最小二乗法 (粒子群最適化)

▶ Pythonチュートリアル: コードレビュー (生成AIが作ったコードの確認)

Vibeコーディング (難易度★★★★): 半導体のキャリア密度の温度依存性のフィッティング (非線形最小二乗法) の one-shotプロンプト

半導体のキャリア統計をシミュレートする、以下の仕様を満たすPythonプログラムを作成してください。

1.【物理モデル】
- 伝導帯、価電子帯、ドナー/アクセプター準位を考慮し、電荷中性条件 (p - n + ND+ - NA- = 0) を満たすフェルミ準位 Ef を決定する。
- キャリア密度 (n, p) は、ボルツマン近似ではなく「フェルミ・ディラック積分 (オーダー1/2)」を用いて計算すること。
- 不純物のイオン化 (ND+, NA-) は、占有関数と退化因子 (gD=2, gA=4) を用いて計算すること。

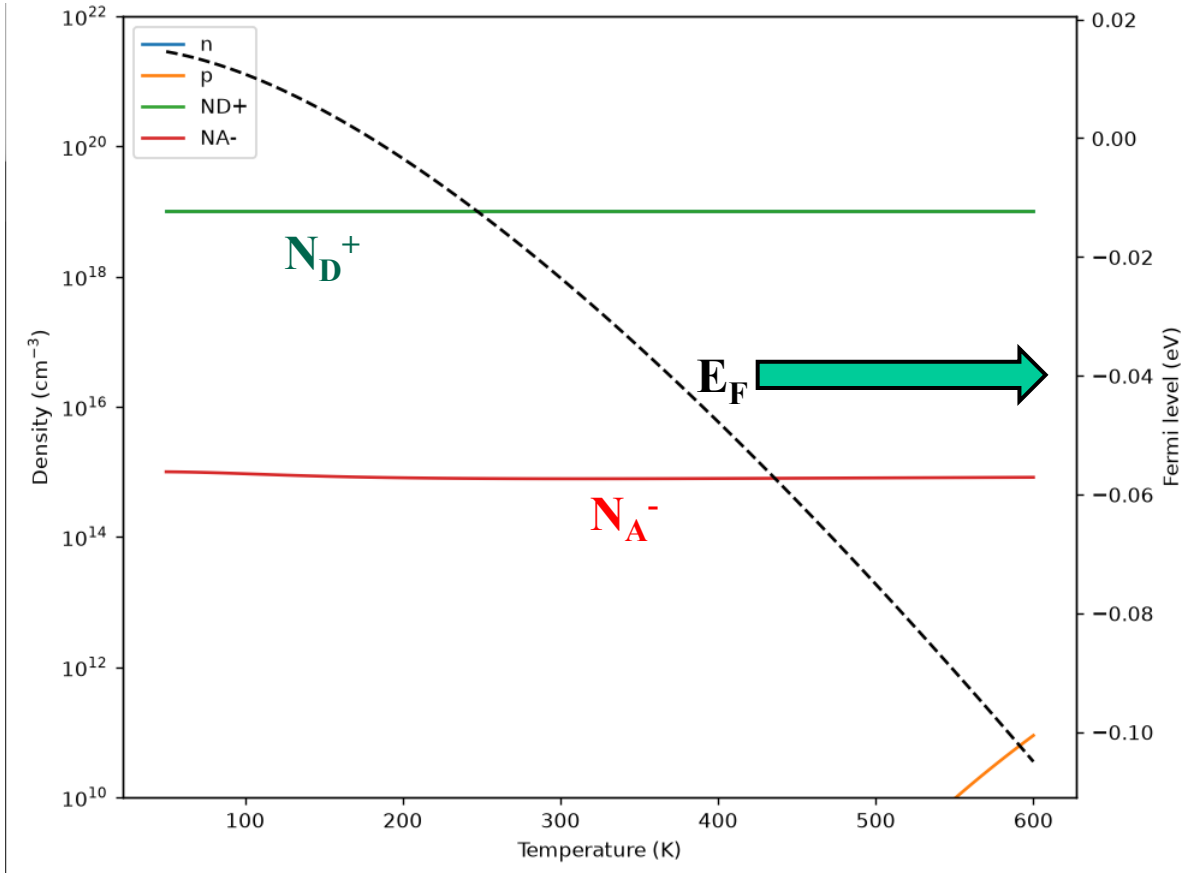
2.【数値計算の安定化】
- 強い縮退（Efがバンド内にある状態）でも計算できるよう、指数関数のオーバーフローを避けるために `scipy.special.expit` を使用した数値積分を行うこと。
- Efの決定には `scipy.optimize.brentq` (Brent法) を使用し、解を囲む範囲 (bracket) が不明な場合は自動で符号反転を確認するまで探索範囲を広げるロジックを実装すること。

3.【プログラム構造】
- `argparse` を使用して、バンドギャップ(Eg)、有効質量(me, mh)、不純物密度(ND, NA)などのパラメータを外部から与えられるようにする。
- `initialize()` 関数でパラメータの初期設定を行い、`main()` で計算とプロットを行う。
- `if __name__ == "__main__":` ギミックを使用すること。
- 各関数名（m2Nc, m2Nv, fe, fh, Fj, Ne, Nh, NDp, NAm, deltaQ）は短く、物理的な意味を反映させること。

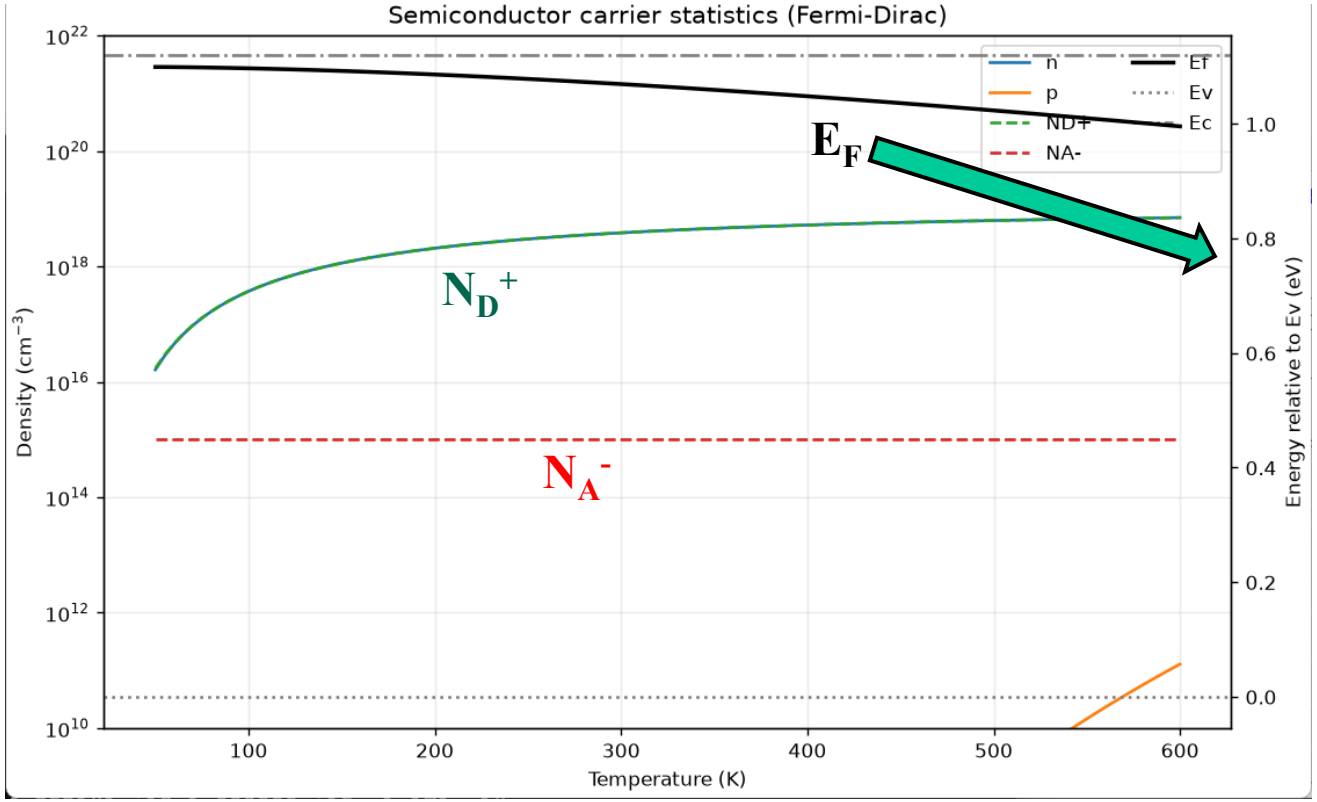
4.【出力とプロット】
- 指定された温度範囲（例: 50K～600K）でスイープを行い、結果をコンソールに表示する。
- グラフ表示の密度範囲は、Nmin, Nmaxパラメータで指定できるようにする (デフォルト Nmin=1e10, Nmax=1e22)
- グラフは2軸プロットとし、左軸に「キャリア密度・イオン化不純物密度 (ログスケール)」、右軸に「フェルミ準位 Ef (リニアスケール)」を描画すること。

MS365 Copilot (無料版) とChatGPT-6.1 Solで作ってみた

> python Ne-T_copilot.py --ND 1e19 --NA 1e15 --Eg 1.12



> python Ne-T_sol.py --ND 1e19 --NA 1e15 --Eg 1.12



今朝つくったので、コードの検証ができていません

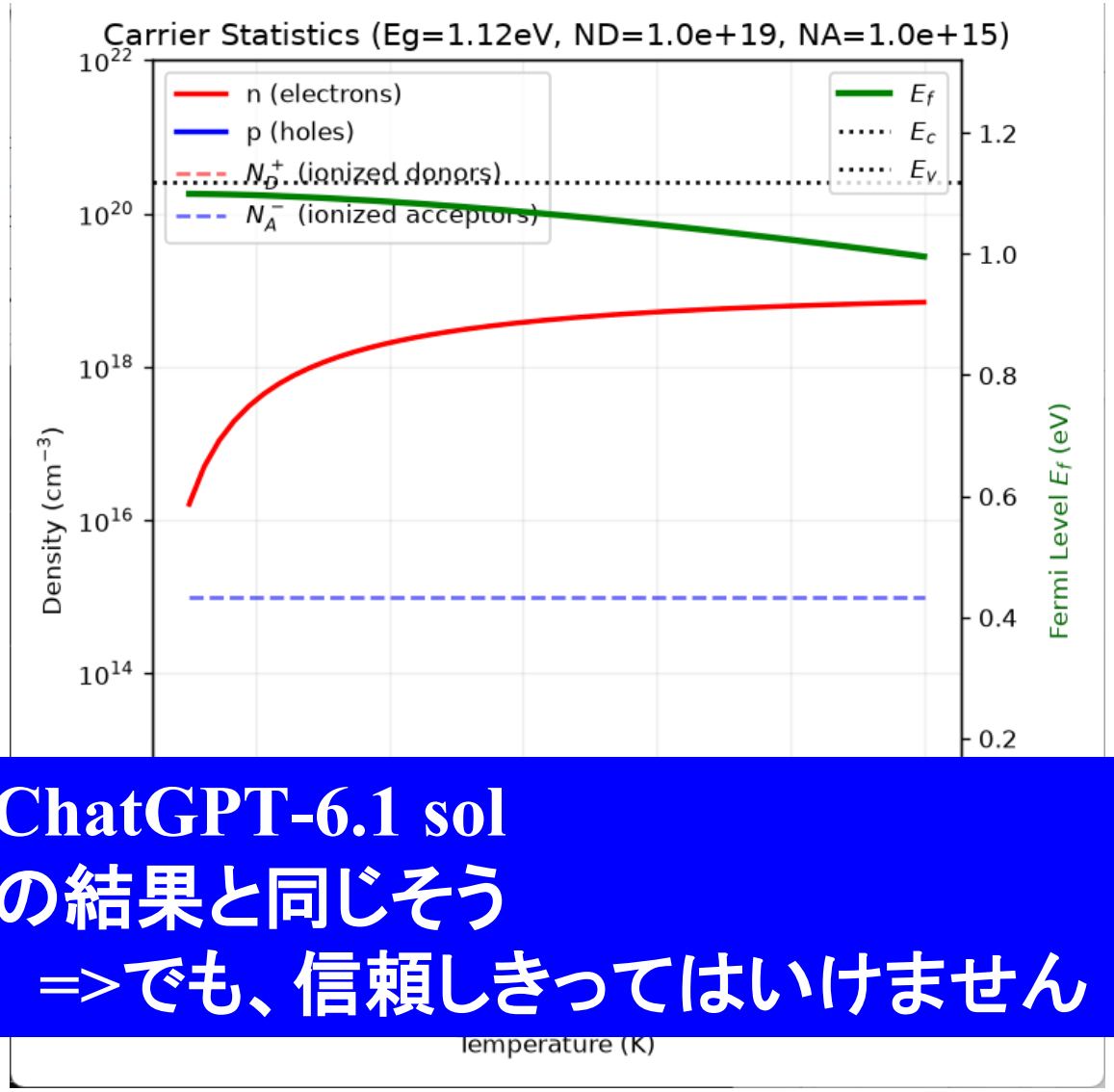
N_D^+をみると、Copilotでは、デフォルト値が違うか、

低温 (<200K)の計算精度が出ていない可能性がある。

原因を調べたうえで使う必要があります

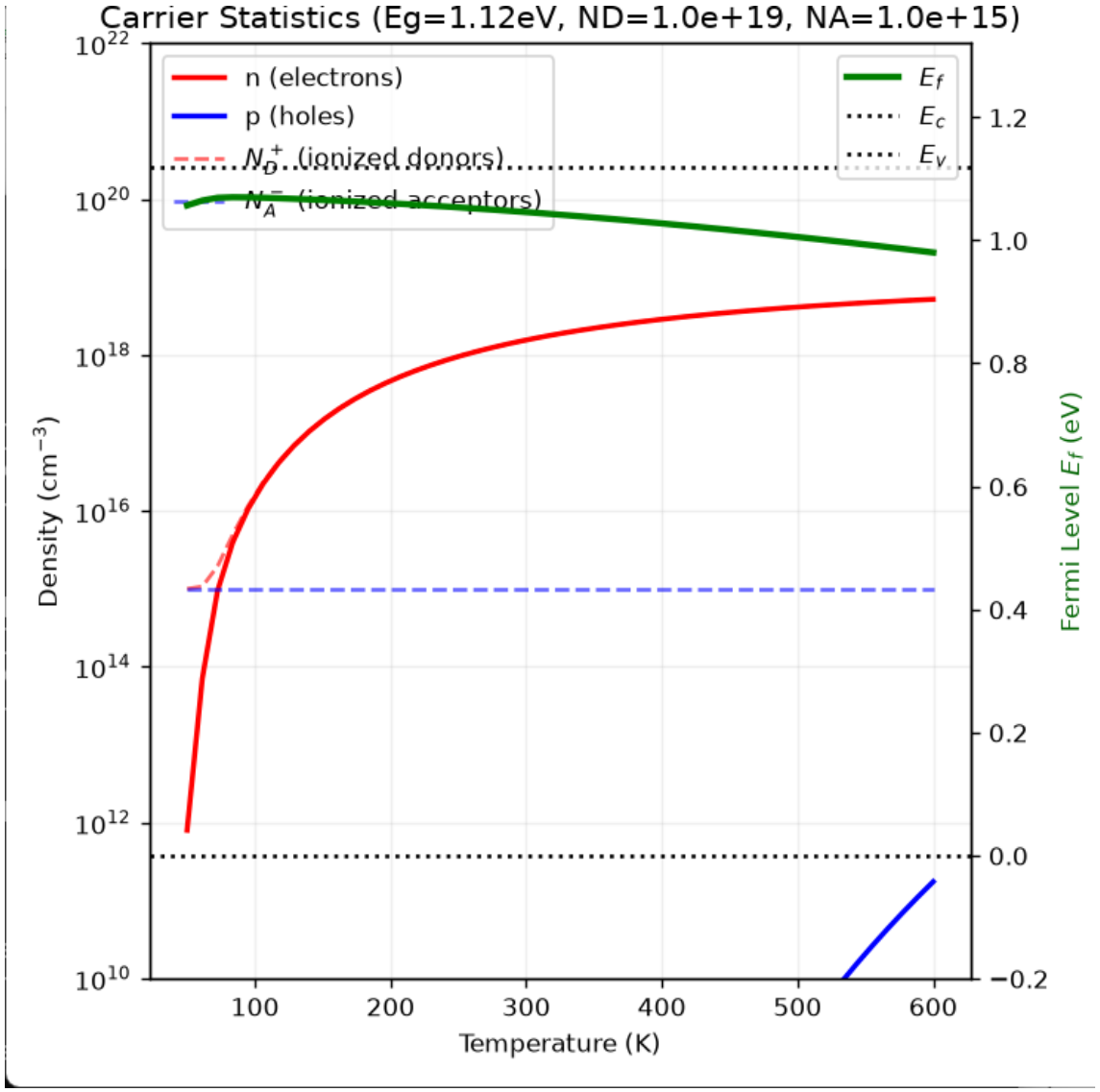
公開動画のプログラム：Gemini 3Flash: 031-Ne-T_oneshot_gemini3flash.py

> python 031-Ne-T_oneshot_gemini3flash.py **--ND 1e19 --NA 1e15 --Eg 1.12**



ChatGPT-6.1 sol の結果と同じそう
=>でも、信頼しきってはいけません

> python 031-Ne-T_oneshot_gemini3flash.py **--ND 1e19 --NA 1e15 --Eg 1.12 --ED 0.1**



まとめ

生成AIでプログラムを作るのは簡単です

- とりあえず、何をしたいかの指示をして、「プログラムを作って」とすると、
  たいていはpythonプログラムを作り、使い方を説明してくれます
- 解析したいデータ (Excelファイルなど) がある場合、それを投入し、
  「添付.xlsxファイルのデータをArrheniusプロットで最小二乗回帰し、活性化エネルギーを出力するプログラムを作って」
  などとすると、データファイル形式の説明も不要です。

ただし、数式などを入れないと、とんでもない理論のプログラムを作りかねません

- 数式、フィッティング変数など
- データファイルの構造と変数の説明
- どこに出力してほしいか (コンソール、ファイル、グラフ)
- 引数で条件や入力ファイル、出力ファイル名を変えられるように依頼
- 計算条件が後で確認できるよう、「入力情報のrepeat出力」を出力するように依頼

は必ず明記しましょう