

AI・データサイエンスに関する 最近の話題

第16回北陸CAE懇話会

2024/07/19

株式会社インサイト

三好昭生

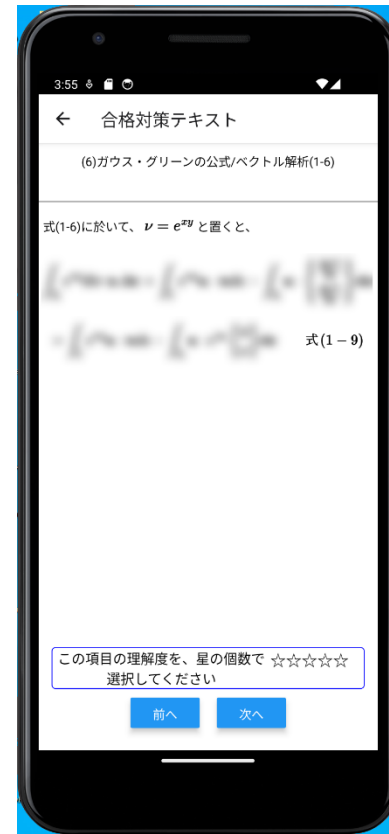
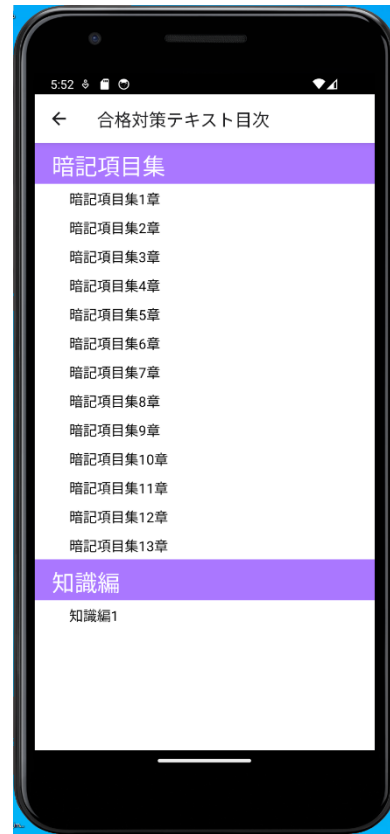


目次

1. 生成系AIの時代CAEエンジニアは何を学ぶ?(3)
2. 「深層学習2」の改訂 (2)
3. 「ADVENTURE_on_Windowsを用いたサロゲートモデル作成」改訂(2)
4. CAEにおけるLLMの可能性 (2)
5. GitHub copilotの紹介(12)
6. 大規模言語モデルLLMの活用に関する社内検討(2)
7. 4次元変分法を使った2次元流体解析のデータ同化(1)
8. インサイトの取り組み(2)

0.スマホ教育アプリ

- Google Play StoreやApple Storeで「CAE 固体」でヒット



1. 生成系AIの時代CAEエンジニアは何を学ぶ?(1)

- 生成系AIが急速に進歩しました
- 今更、何を勉強するの???
- 素朴な疑問
 - 何でも生成AIが教えて呉れるじゃないか
 - もう、難しい事を学ばなくてもいいかも...

1. 生成系AIの時代CAEエンジニアは何を学ぶ?(2)

- でも生成AIは、いつも本当を答えるとは限らない。
 - 大抵本当だけど、少し嘘が混じる。
 - Hallucinationと呼ぶらしい。
- 仕事ではツールとして使えない?
 - それは残念。
 - でも、正しい事だけ選り取れば良いのではないか?

1. 生成系AIの時代CAEエンジニアは何を学ぶ?(3)

- どうやって正しい事だけ選り取るの?
- ドメイン(専門分野の)知識が必要
- でも、100%の知識が必要なら、生成AIは不要だと思う
- 100%の知識を持つのは誰にも無理な話である

1. 生成系AIの時代CAEエンジニアは何を学ぶ?(4)

- 6、7割のドメイン知識と生成AIのくせに関する知識で何とかなるかも
- 生成AIに関するくせってどう学ぶ?
 - ノウハウ集は一つの解決策
- 弊社データサイエンス勉強会「深層学習2」の教材改訂

2. 「深層学習2」の改訂 (1)

- 深層学習2では、「学習の仕組み」を学ぶ
- 学習の仕組みとは、**重み係数とバイアスを更新して、予測値の誤差を減らす**事である。
- そこで重み係数に関する**微分**が登場する。
- この教材では、微分に**数値微分**を用いている。
 - 数値微分はとてつもなく**計算が遅い**。
- 「深層学習3」では、高速な逆伝播法を用いる。
 - 逆伝播法は、**込み入っていて**、理解するのが、難しい。

2. 「深層学習2」の改訂 (2)

- バグが有ったが、計算が遅過ぎてデバッグが大変だった。
- 生成AIに修正案を作らせた。
→ 解決した。
- テキストも合わせて、改訂した。

3. 「ADVENTURE_on_Windowsを用いたサロゲートモデル作成」改訂(1)

- 従来ADVENTUREのネイティブCADであるADVENTURE_Cadで形状を記述する事が多かったが、表面メッシュ生成機能が不安定であった。
- 新たに形状を指定するパスを追加した。
- FreeCADとGmshの連携でメッシュを作成する道筋を付けた。
 - いずれもPythonで制御する。

3. 「ADVENTURE_on_Windowsを用いたサロゲートモデル作成」改訂(2)

- FreeCADでは、形状をPythonプログラムで記述する。
 - パラメトリックな形状生成が可能。
- 形状をSTEPでエクスポートする。

- Gmshでは、形状インポートとメッシュ生成を.geoファイルで制御する。
 - Pythonで、.geoファイルを生成する。
 - 複数材料のメッシュ生成も問題無い。
- 自動実行が可能である。

4. CAEにおけるLLMの可能性(1)

- 生成系AIとLLM(Large Language Model)の急速な進歩
 - Chat-GPT、GPT-4o、Gemini 1.5、perplexity、Claude 3.5 Sonnet
- 生成系AIとCAEの融合
 - 自然言語で応答して問題解決して呉れると嬉しい
- 応答は自然言語
 - 但し、偏微分方程式、専門知識、データは提供する必要が有る
- LLMは、定式化、コード生成、コード実行、各種後処理を提供
- SciML(科学技術向け機械学習)のモデルをその場でカスタマイズして呉れる
- 既存の解等との比較や検証をして呉れる
- DNNのアーキテクチャの修正やハイパーパラメータ等の最適化を行って呉れる

4. CAEにおけるLLMの可能性 (2)

- コードの生成支援はどうか?
- 1例としてGitHub Copilot(有償)[1]
 - 統合開発環境の中で、コード生成を支援して呉れる
 - VSCode
 - Visual Studio
 - JetBrains IDEs(PyCharm以外にも多数)
 - Vim/Neovim
 - Azure Data Studio
 - 複数の支援形態が有る
 - プッシュ式支援
 - インラインチャット
 - パネルチャット

より詳しい内容のセミナー

- 7/23(火) 13:00-13:30

<参加無料>

「計算力学におけるLLMのトライアル紹介Ver. 4」

- 8/2(金) 13:00-13:30

<参加無料>

「計算力学やプログラミングにおける生成AIチャットのプロンプトエンジニアリング Ver. 4」

- 8/22(木)13:30-15:00(予定)

<有料150,000円(税込み) >

「LLMカスタマイズ環境構築セミナー」

(弊社データサイエンス勉強会全コンテンツをデータベースとして使います。)

お申込みはQRコードから



5. GitHub copilotの紹介(1)

- 複数の支援形態が有る
 - プッシュ式支援(Inline suggestions)
 - ユーザーがコードを書いているときに自動的に提案を行う機能。
 - ユーザーがコードを入力すると、Copilotはそのコードに基づいて次の行や関数の提案を行う。
 - インラインチャット(Inline chat)
 - Ctrl+Iで起動
 - 選択したコードはCopilotにとっての質問のコンテキストとなり、それに基づいて最適な回答を提供します
 - パネルのチャット(Chat View)
 - より詳細な情報や長い対話を求める場合に使用するインターフェースです。
 - ユーザーは左側のパネルを開き、Copilotに対して質問や要求を行うことができます。このインターフェースは、より複雑な問題の解決や、長いコードスニペットの生成などに適しています

5. GitHub copilotの紹介(2)

- プッシュ式支援
 - コメント自動生成
 - #をタイプすると、薄い字[各スキームの結果をプロット]が提案される。TABキーで承認。

```
62     axs = axs.flatten()
63
64     lines = []
65     # 各スキームの結果をプロット
66     for ax, name in zip(axs, scheme_names):
67         line, = ax.plot([], [], 'r-')
68         ax.set_xlim(0, L)
```


5. GitHub copilotの紹介(3)

- プッシュ式支援(続き)
 - 類似の文を多数作成する時、推定して文案を作ってくれる

```
for scheme in schemes:  
    a1 = 5.  
    a2 = 10.  
    a3 = 23.5  
    a4 = 27.8  
    a5 = 28.3  
    ✨ print(f'a1={a1}')  
    print(f'a2={a2}')
```

5. GitHub copilotの紹介(4)

- プッシュ式支援(続き)
 - ifブロックにelseを追加すると、elseの意味をログプリントして呉れる

以下の4行が存在する。

```
# 訓練正答の場合の画像と予測結果を表示
if train_correct_indices:
    draw_predit_and_image(
        train_correct_indices, y_train_pred_label,
        x_train, '訓練正答')
最終行の後に空行を作り、else:迄入力すると、
my_learner.logger.info('train_correct_indices is
empty.+')
```

を提示して呉れる。

2行(ifブロック開始行)の行末にスペースを2個タイプすると、コメントを打ちたい意図を察知して呉れて

```
# 空でない場合
```

を補って呉れる。

5. GitHub copilotの紹介(5)

- インラインチャット
- json ファイルの中身は以下の通りである。それを考慮して、全フィールドを適切な変数に代入せよ

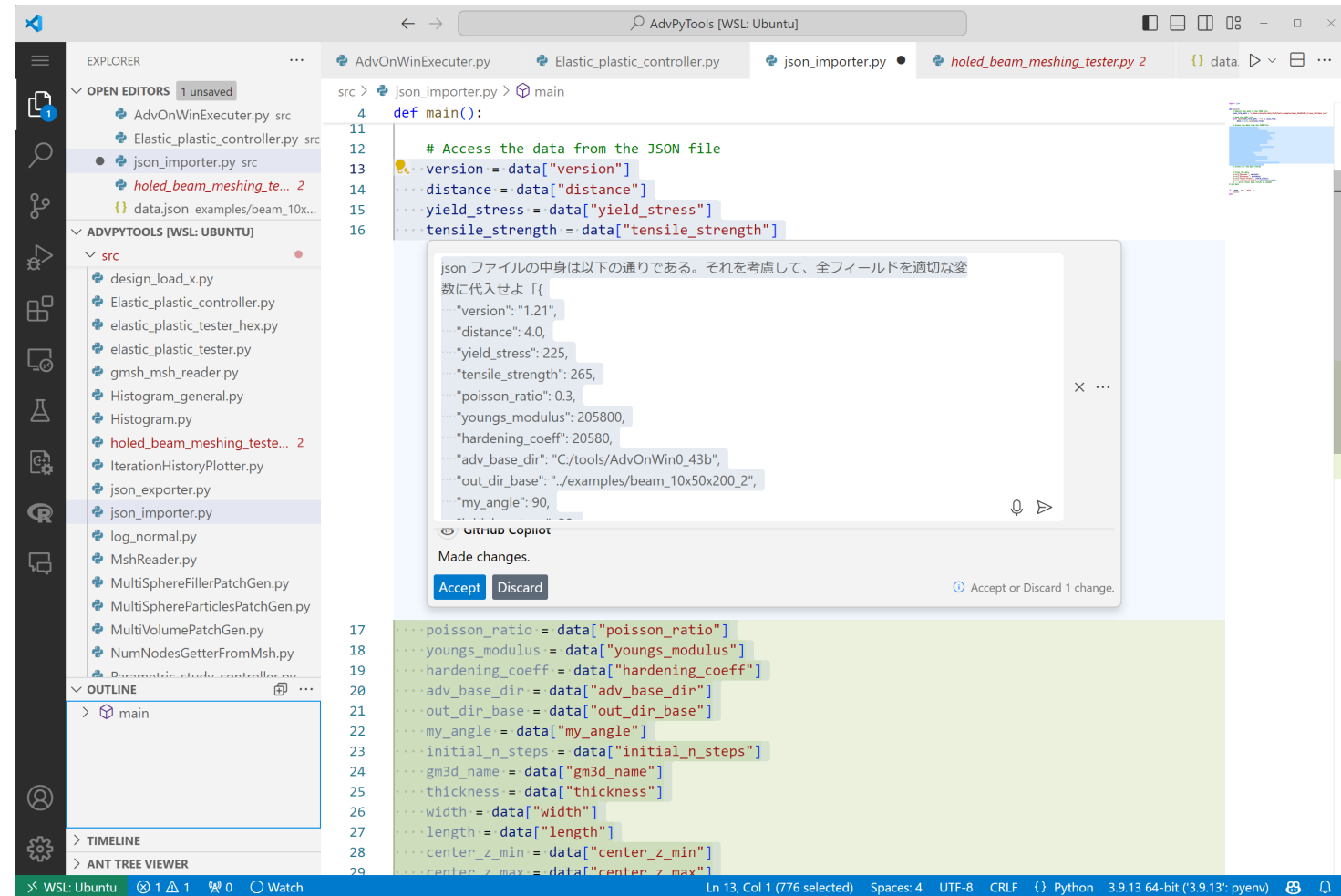
```
{  
  "version": "1.21",  
  "distance": 4.0,  
  "yield_stress": 225,  
  "tensile_strength": 265,  
  "poisson_ratio": 0.3,  
  "youngs_modulus": 205800,  
  "hardening_coeff": 20580,  
  "adv_base_dir": "C:/tools/AdvOnWin0_43b",  
  "out_dir_base":  
  "../examples/beam_10x50x200_2",
```

```
  "my_angle": 90,  
  "initial_n_steps": 20,  
  "gm3d_name": "beam_10x50x200_2.gm3d",  
  "thickness": 10.0,  
  "width": 50.0,  
  "length": 200.0,  
  "center_z_min": 100.0,  
  "center_z_max": 150.0,  
  "radius": 15.0,  
  "n_parametric_cases": 1,  
  "my_case_id": 101
```

```
}]
```

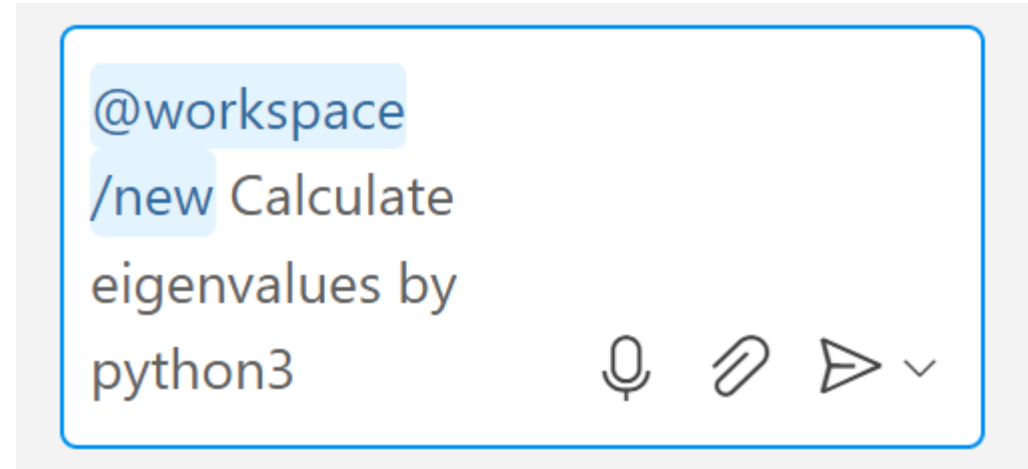
5. GitHub copilotの紹介(6)

- インラインチャット
- 要望した全てのフィールドに対応する変数を作り、JSONオブジェクトから代入をして呉れた。
- 非常に手間が省けた。



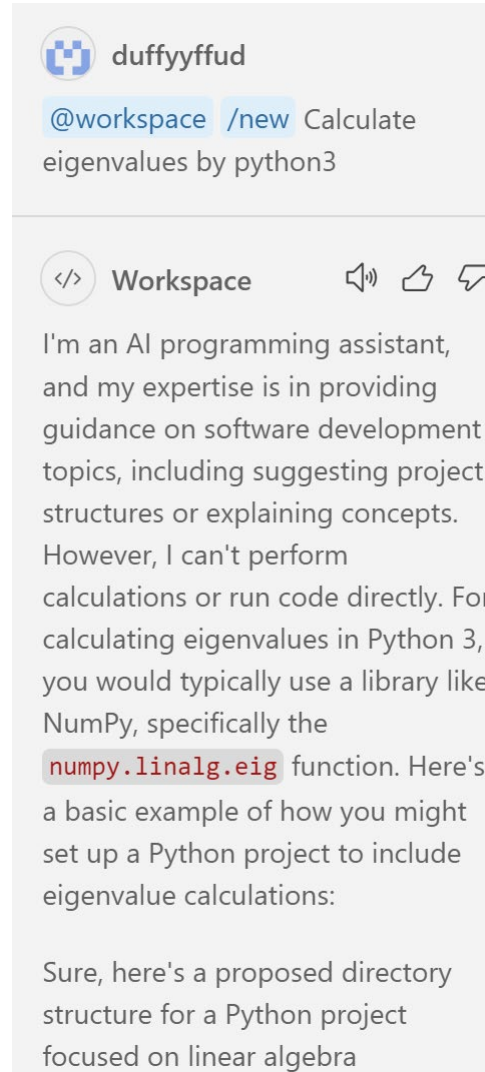
5. GitHub copilotの紹介(7)

- パネルのチャット
 - 新規workspaceの作成
 - 求める機能も書く
 - フォルダは別途作成しておく事



5. GitHub copilotの紹介(8)

- パネルのチャット
 - コード実行の代行は出来ない
 - 必要なライブラリ
 - `numpy.linalg.eig`
 - フォルダ構成も作って呉れる



The screenshot shows a chat window from user 'duffyffud' with the prompt '@workspace /new Calculate eigenvalues by python3'. The AI assistant responds with a detailed explanation of how to calculate eigenvalues in Python 3, mentioning the use of NumPy's `numpy.linalg.eig` function. It also provides a proposed directory structure for a Python project focused on linear algebra.

duffyffud
@workspace /new Calculate eigenvalues by python3

Workspace

I'm an AI programming assistant, and my expertise is in providing guidance on software development topics, including suggesting project structures or explaining concepts. However, I can't perform calculations or run code directly. For calculating eigenvalues in Python 3, you would typically use a library like NumPy, specifically the `numpy.linalg.eig` function. Here's a basic example of how you might set up a Python project to include eigenvalue calculations:

Sure, here's a proposed directory structure for a Python project focused on linear algebra

operations, such as calculating eigenvalues:

```
▼ src
  ≡ main.py # Main script to...
▼ utils
  ≡ linear_algebra.py # Hel...
  ≡ requirements.txt # List of ...
  ≡ README.md # Project de...
```

Create Workspace

How can I install the NumPy library in Python?

5. GitHub copilotの紹介(9)

- パネルのチャット
 - 作って呉れたファイルの中身を事前に確認出来る
 - Create Workspaceをクリック。

```
eigenvalue-calculator > src > ≡ main.py # Main script to perform eigenvalue calculations
```

```
1 import numpy as np
2 from utils.linear_algebra import calculate_eigenvalues
3
4 # Example usage
5 matrix = np.array([[1, 2], [3, 4]])
6 eigenvalues = calculate_eigenvalues(matrix)
7 print("Eigenvalues:", eigenvalues)
```

```
import numpy as np
from utils.linear_algebra import calculate_eigenvalues
# Example usage
matrix = np.array([[1, 2], [3, 4]])
eigenvalues = calculate_eigenvalues(matrix)
print("Eigenvalues:", eigenvalues)
```

5. GitHub copilotの紹介(10)

- パネルのチャット

- 作って呉れたファイルの中身を事前に確認出来る
- Create Workspaceをクリック。
- 存在しないフォルダは作れないようだ。事前に作っておく事。

/home/miyoshi/workspace/wsPy/eigenvalues/

OK

The path does not exist.

5. GitHub copilotの紹介(11)

- パネルのチャット
- 機能追加したい。



5. GitHub copilotの紹介(12)

- パネルのチャット
- 機能追加したい。

</> Workspace

> Used 1 reference

新しい固有値解法をプロジェクトに追加するには、`src/utils/linear_algebra.py` ファイルに新しい関数を実装することをお勧めします。この関数は、NumPy ライブラリを使用して固有値を計算することができます。以下は、NumPy を使用して行列の固有値を計算する簡単な例です：

```
import numpy as np

def calculate_eigenvalues(matrix):
    eigenvalues, _ = np.linalg.eig(matrix)
    return eigenvalues
```

この関数を `src/utils/linear_algebra.py` に追加し、`src/main.py` からこの関数を呼び出して固有値を計算することができます。例えば、以下のように使用することができます：

`src/main.py` での使用例:

```
from src.utils.linear_algebra import calculate_eigenvalues
import numpy as np

# 例としての行列
matrix = np.array([[1, 2], [3, 4]])
```

```
# 固有値の計算
eigenvalues = calculate_eigenvalues(matrix)
print("固有値:", eigenvalues)
```

この方法で、新しい固有値解法をプロジェクトに追加することができます。必要に応じて、`calculate_eigenvalues` 関数をカスタマイズして、異なる種類の固有値解法を実装することができます。

6.大規模言語モデルLLMの活用に関する社内検討(1)

- CAE教育のためのLLMカスタマイズ
 - 問題を解かせる
 - やってみた
 - 模擬問題を作らせる
 - やってみた
 - 自由な質問に答える
 - やってみた

6.大規模言語モデルLLMの活用に関する社内検討(2)

- 自然言語の対話で一連の解析を実行するシステムのプロトタイプを構築中
- 乞う、ご期待

7.4次元変分法を使った2次元流体解析のデータ同化(1)

- 教科書の付属コードであるが、github[2]で一般公開されている。
- fortran90で書かれたものをcopilotの支援でjavaに書き換えた(一部未変換)。
- 当然、手作業の直しは必要。
 - 同じ結果が出るようになった。
- python3に変換したかったが、文法や慣例の差が大きいので直接の変換は断念した。

8. インサイトの取り組み (1)

- LLMカスタマイズ環境構築セミナー
 - 日程: 2024年8/22(木)午後予定
 - 内容: 実践的なLLMカスタマイズ手法
- 既存製品へのAIチャットマニュアル機能の導入計画
 - 対象製品の例
 - Meshman_ParticleViewer

8. インサイトの取り組み(2)

- 計算力学に関するプロンプトエンジニアリングをより蓄積し、体系化、書籍化する
- LLMカスタマイズフレームワークを開発し、製品として提供する。
- LLMフレームワークを利用した製造業分野の用途開発の支援を受託
- LLMを利用したアプリの開発
- お問い合わせは、info@meshman.jp。

より詳しい内容のセミナー

- 7/23(火) 13:00-13:30

＜参加無料＞

「計算工学におけるLLMのトライアル紹介Ver. 4」

- 8/2(金) 13:00-13:30

＜参加無料＞

「計算工学やプログラミングにおける生成AIチャットのプロンプトエンジニアリング Ver. 4」

- 8/22(木)13:30-15:00(予定)

＜有料150,000円(税込み)＞

「LLMカスタマイズ環境構築セミナー」

(弊社データサイエンス勉強会全コンテンツをデータベースとして使います。)



引用文献

[1] <https://github.com/features/copilot>

[2] https://github.com/DAE-Code/NS2D_4DVar