Exploring AI-Based Tools for Scientific Software Development

Nazneen Ansar

Mentors: Dr. Fangfang Yu and Dr. Bikash Basnet

Objectives

- To explore the capabilities of AI-Based tools across different facets of scientific software development, encompassing areas like code generation, documentation support, code translation, and debugging.
- To identify the strength and limitations of using AI-Based tools for real-world context-based applications that can be easily integrated into existing frameworks of code.

Results

- ChatGPT was able to assist in low-level conversion of the example GOES-16 ABI spectral bands processing module. The chatbot was able to derive information about the input code on its own, understand at least in part what the code intends to do, and provide feedback, test cases, and improvements to the code if asked to do so. ChatGPT's abilities include efficient python library usage and transition, code optimization, detailed annotations, and context-aware code segmentation.
- In evaluating AI-Based tools for scientific software development, distinct strengths were observed: Bard excelled in preserving code integrity during conversion, while ChatGPT showcased proficiency in code optimization and generation. For code documentation within VSCode, GitHub Copilot emerged as the superior tool due to its predictive capabilities, albeit lacking an interactive chatbot feature.
- These findings emphasize the potential of AI tools in streamlining and enhancing the software development process.

Figure(s)

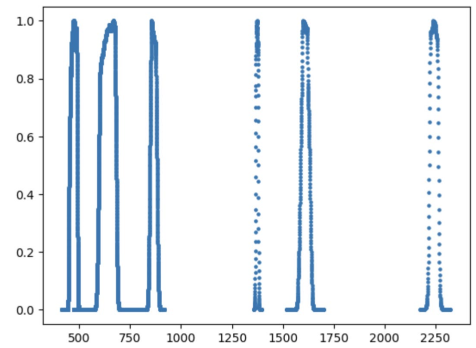
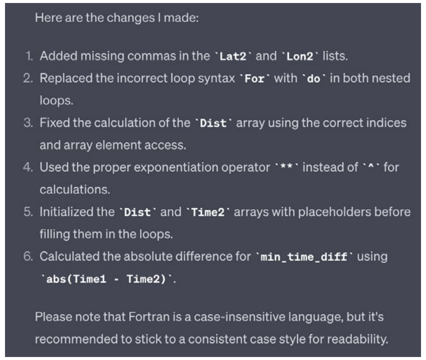


Fig 1: Plot of wavelength vs Relative Spectral Response of GOES-16 ABI. These spectral datasets were used during IDL to Python conversion test using ChatGPT.

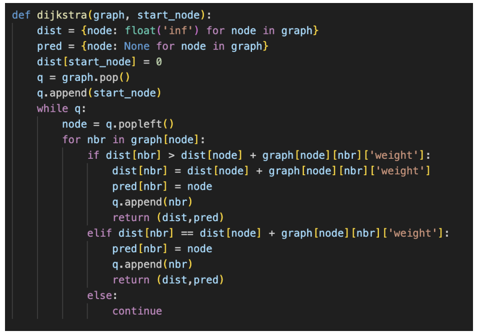


Here are the changes I made:

1. Added missing commas in the `Lat2` and `Lon2` lists.
2. Replaced the incorrect loop syntax `For` with `do` in both nested loops.
3. Fixed the calculation of the `Dist` array using the correct indices and array element access.
4. Used the proper exponentiation operator `**` instead of `^` for calculations.
5. Initialized the `Dist` and `Time2` arrays with placeholders before filling them in the loops.
6. Calculated the absolute difference for `min_time_diff` using `abs(Time1 - Time2)`.

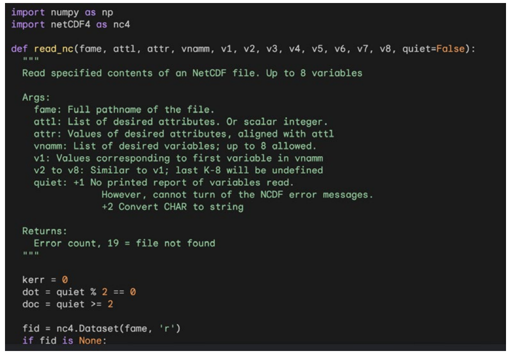
Please note that Fortran is a case-insensitive language, but it's recommended to stick to a consistent case style for readability.

Fig 2: ChatGPT suggestions for debugging the space-time collocation algorithm pseudocode.



```
def dijkstra(graph, start_node):
    dist = {node: float('inf') for node in graph}
    pred = {node: None for node in graph}
    dist[start_node] = 0
    q = graph.pop()
    q.append(start_node)
    while q:
        node = q.popleft()
        for nbr in graph[node]:
            if dist[nbr] > dist[node] + graph[node][nbr]['weight']:
                dist[nbr] = dist[node] + graph[node][nbr]['weight']
                pred[nbr] = node
                q.append(nbr)
                return (dist,pred)
            elif dist[nbr] == dist[node] + graph[node][nbr]['weight']:
                pred[nbr] = node
                q.append(nbr)
                return (dist,pred)
            else:
                continue
```

Fig 3: Weighted shortest path algorithm (Dijkstra's) generated by CoPilot



```
import numpy as np
import netCDF4 as nc4

def read_nc(fame, attl, attr, vnamm, v1, v2, v3, v4, v5, v6, v7, v8, quiet=False):
  """
  Read specified contents of an NetCDF file. Up to 8 variables

  Args:
    fame: Full pathname of the file.
    attl: List of desired attributes. Or scalar integer.
    attr: Values of desired attributes, aligned with attl
    vnamm: List of desired variables; up to 8 allowed.
    v1: Values corresponding to first variable in vnamm
    v2 to v8: Similar to v1; last K-8 will be undefined
    quiet: +1 No printed report of variables read.
            However, cannot turn of the NCDF error messages.
            +2 Convert CHAR to string

  Returns:
    Error count, 19 = file not found
  """

  kerr = 0
  dot = quiet % 2 == 0
  doc = quiet >= 2

  fid = nc4.Dataset(fame, 'r')
  if fid is None:
```

Fig 4. Documentation provided by Bard for the Spectral bands processing module.