Bringing Magics weather maps to Matplotlib

Alish Dipani

[1] Problem Statement & Summary

ECMWF's Magics [1] is a meteorologically oriented graphical library that can visualise data coded on GRIB and NetCDF on different projections of the globe. This library makes the visualisation of the most common meteorological parameters convenient by defining a set of predefined visual definitions. While in Python, the library does not fully take advantage of the most popular visualisation library, Matplotlib [2], and the syntax and conventions are not highly pythonic. Combining the library with Matplotlib would allow the users to combine meteorological data with various plots like scatter plots, heatmaps, etc., available in Matplotlib. And updating the syntax would make the library more user-friendly. ECMWF created a new library, Magpye [3], which is currently in the Beta phase to tackle these issues. This project aims to continue the development of the Magpye library and work towards the following goals:

1. Making the syntax and conventions more pythonic.
2. Combining the functionality of Matplotlib with Magics.
3. Making the plots more customisable.

[2] Approach

Magpye library acts as a python interface to Magics. It allows users to load meteorological data and plot it on predefined geographical areas (such as Africa, Europe, Global, etc.) using predefined styles for lines and shading. Magpye takes in the data and passes it to Magics *macro* module, which plots the data. This final plot is in Magics's own binary format and can be decoded into Matplotlib Axes using the *binary* module. The main components of the Magpye library are shown in Figure 1.

In Magpye, users can define a *GeoMap* object with one of the predefined geographical areas. Then, data is loaded and added to the map in the form of line contours or filled contours, which have various predefined styles available. Various features can then be added to the geographical area such as rivers, coastlines and gridlines. Finally, users can save this figure in, either a PDF format or PNG format, and can also display this in Jupyter Notebooks.

Now, this plot is saved in Magics's binary format and the *binary* module can be used to convert this into Matplotlib axes. We can then manipulate these axes to customise the plot, make it interactive or even add Matplotlib graphs on top of the map.

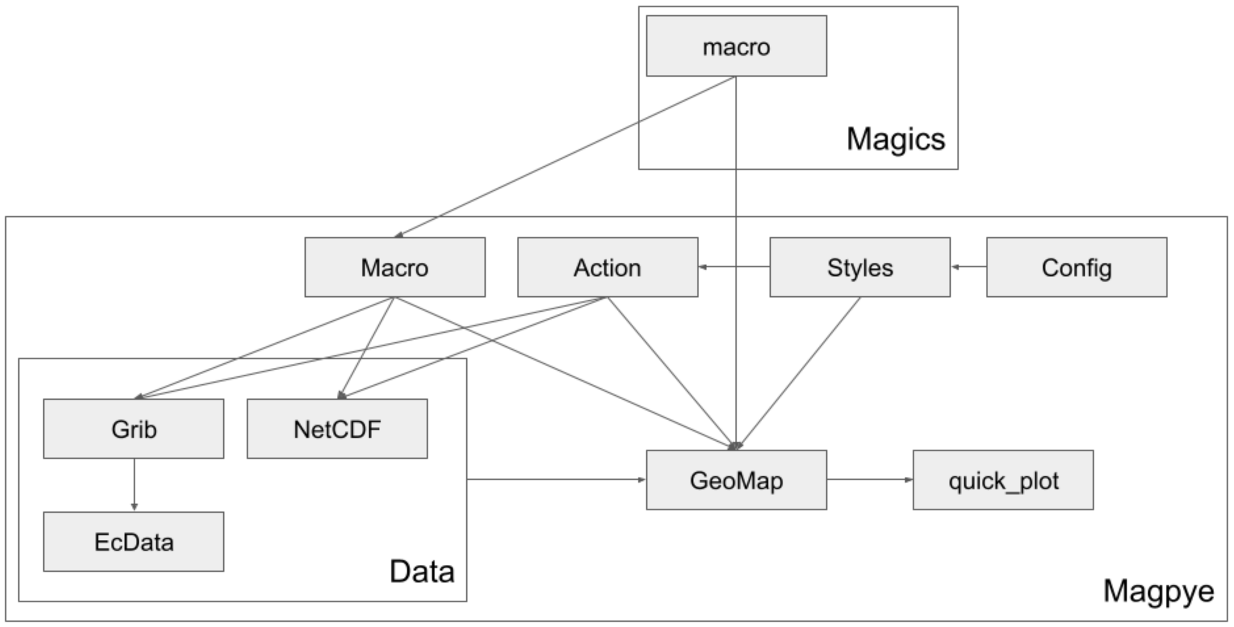


Figure 1. Major components of the library Magpye.

A Proof of Concept (with outputs) of this is as follows:

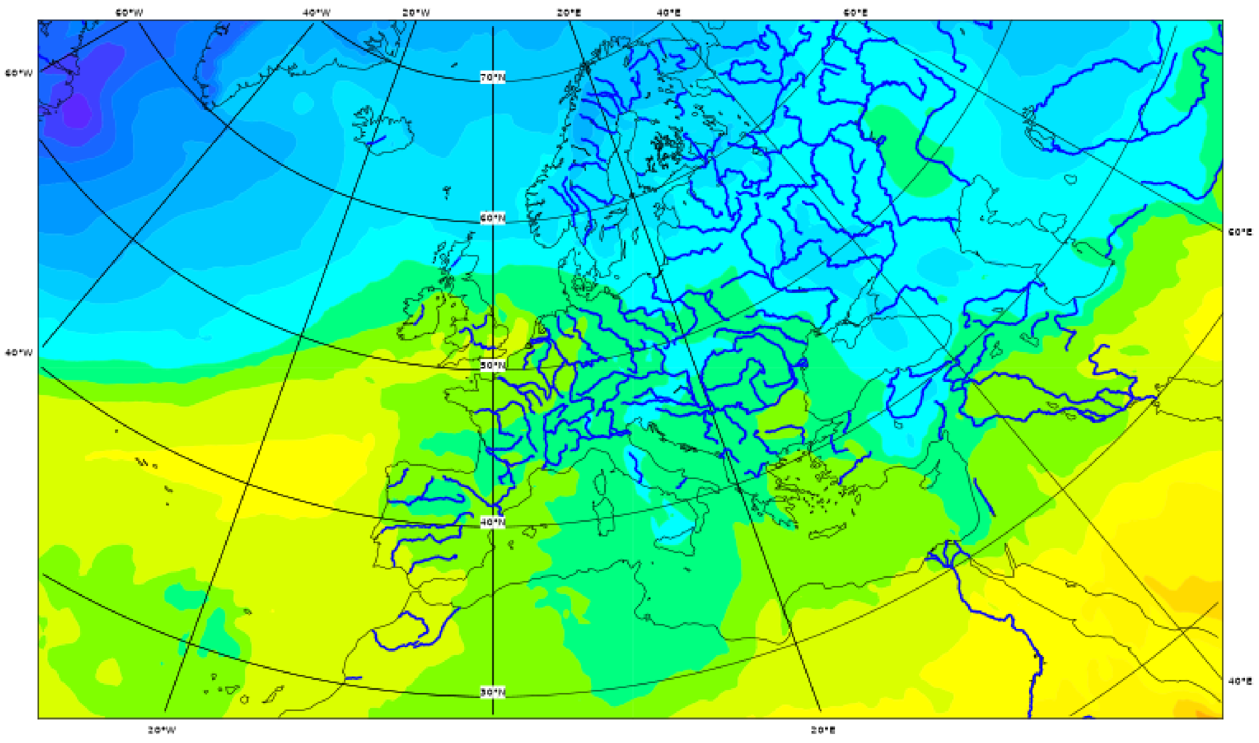
1. First, we create the map using Magpye:

```python
# Importing Libraries
from magpye import GeoMap

# Defining the GeoMap object
fig = GeoMap(area_name='europe')

# Loading the data as shaded contour and predefined style
fig.contour_shaded("t850.grib", style="rainbow_temperature1")

# Adding gridlines, coastlines and rivers to the map
fig.coastlines(resolution="medium")
fig.gridlines()
fig.rivers(line_thickness=3, resolution="high")
fig.show()
```

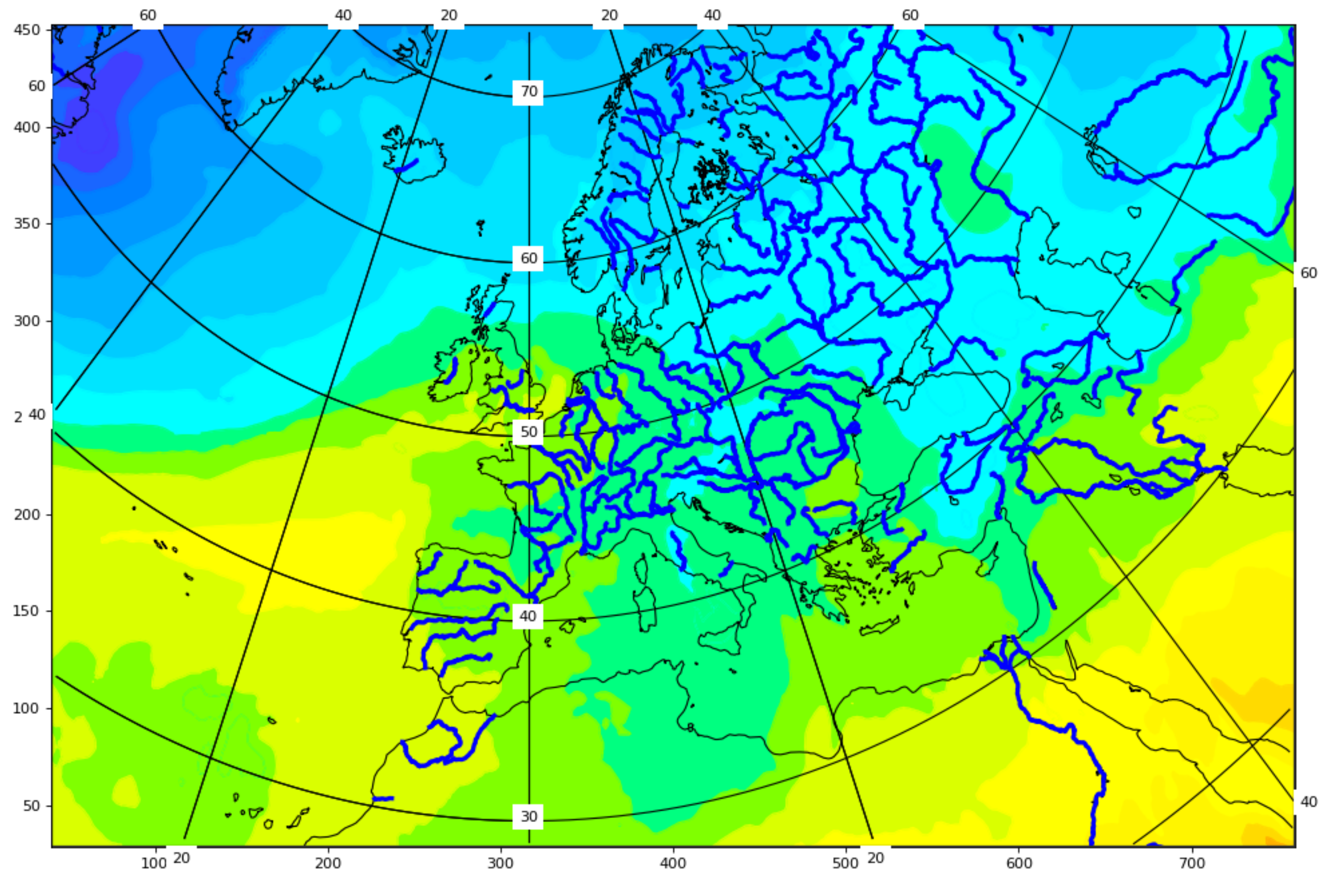


2. Then, we save the figure and convert it into matplotlib axes using Magics's *binary* module:

```python
# Save the figure
mgb = "magics-binary.mgb"
fig.save(mgb)

# Load the figure and decode using binary module
from Magics import binary
%matplotlib widget

# Plot the figure as matplotlib axes
from matplotlib.pyplot import figure
figure(figsize=(15, 10), dpi=80)
binary.plot_mgb(mgb)
```



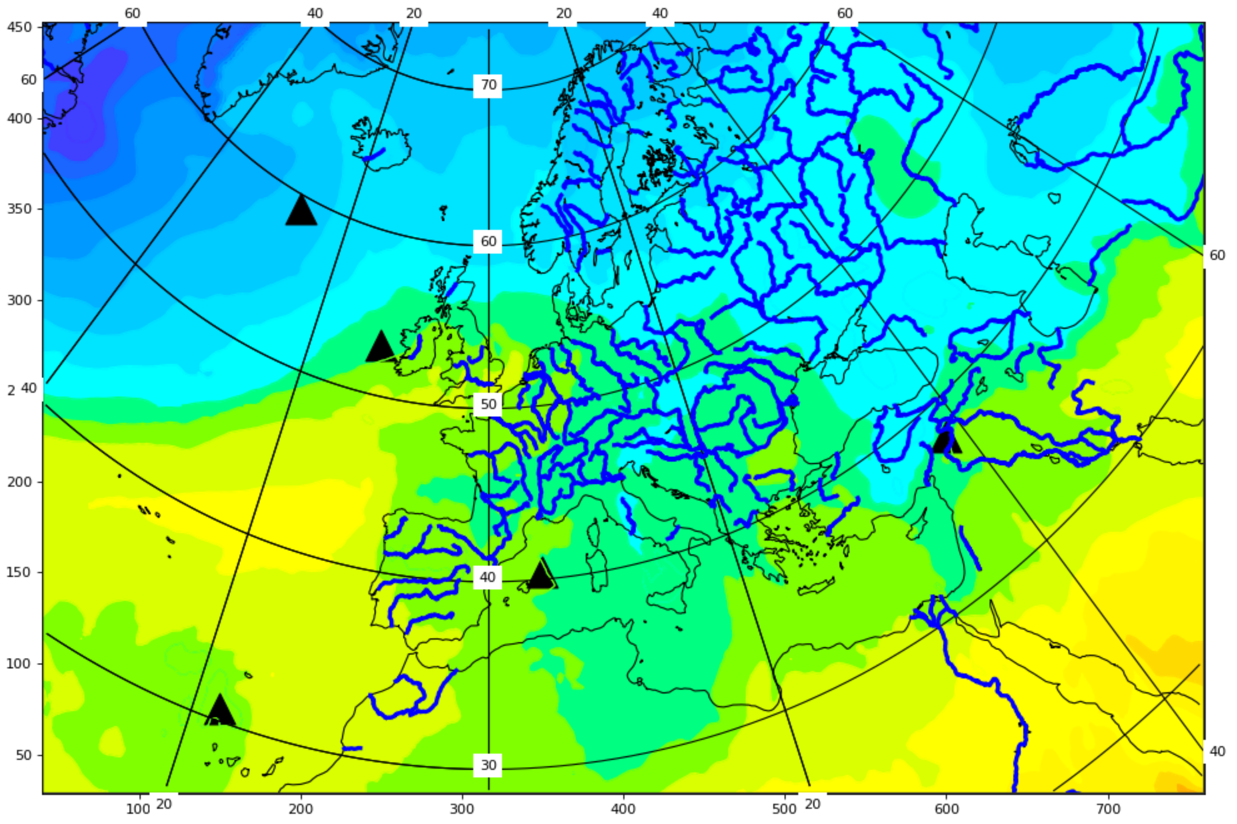
3. Finally, we add a matplotlib scatter plot over the map to mark various points:

```
# Importing matplotlib
import matplotlib.pyplot as plt

# Define data to be plotted
# This data can be python arrays or numpy arrays
x = [150, 200, 250, 350, 600]
y = [75, 350, 275, 150, 225]

# Create a scatter plot with the following properties:
# - upper filled triangle markers
# - marker size of 500 points
# - Black marker color
plt.scatter(x, y, s=500, marker='^', c='k')

# Show the plot
plt.show()
```



This proof of concept shows that the plot can be converted to matplotlib axes and can be manipulated and customised. But, the interface for this customisation and adding graphs are missing. A solution to this would be introducing a new module in Magpye, which would allow such manipulations and pretty plots.

[3] Key Milestones and Deliverables

I plan a four-phase timeline for achieving the objectives of this project:

- **Phase A:** Improving upon the current codebase by documenting the Magics and Magpye libraries and planning the features to add to the Magpye library.
- **Phase B**: Improving the plots by modifying and adding several features to customise the plots, such as handling of axes, fonts, font sizes, aspect ratio, etc.
- **Phase C**: Adding the feature of plotting matplotlib graphs upon the maps.
- **Phase D**: Documenting the Magpye library and creating example notebooks and tutorials, and quick-start guides for users.

Each phase will be followed by a code/documentation review.

Phase-wise deliverables are as follows:

- **Phase A**: Documentation of Magics and Magpye libraries
- **Phase B**: Improvement in plots and adding features to customise the plots
- **Phase C**: Adding matplotlib graphs upon the plots
- **Phase D**: Documentation of Magpye library and tutorials for Magpye library.

The plan and timeline for each phase are given in the next section.

[4] Timeline

A tentative timeline is as follows:

Week	Objective
Phase A	
Week 1 - 2 [2nd May - 15th May]	- Discussing specific features to implement - Documenting Magics library
Week 3 - 4 [16th May - 29th May]	- Planning feature implementations and improving Magpye to accommodate those features - Documenting Magpye library
Code Review & Documentation Review	
Phase B	
Week 5 - 7 [30th May - 19th June]	- Implementing and improving the handling of the fonts and axes, - Implementing and improving the thickness of the lines - Understanding the warnings - Implementing and improving the aspect ratio and georeferencement
Week 8 - 10 [20th June - 10th July]	- Implementing and improving more features - Improving Documentation of Magpye library
Code Review & Documentation Review	
Phase C	

Week 11 - 14 [11th July - 7th August]	- Implementing adding matplotlib plots to Magpye plots
Code Review & Documentation Review	
Phase D	
Week 15 - 17 [8th August - 31st August]	- Improving the documentation of Magpye library - Creating example notebooks and tutorials - Creating installation and quick start guides
Code Review & Documentation Review	

Changes in timelines are possible upon recommendations of the mentors.

[5] Progress Updates and Contact with mentors

I plan to communicate with the mentors regularly (in weekly / bi-weekly meetings) to discuss the project's progress. These meetings can be held over mail or video-conferencing.

[6] Maintainability, Extensibility & Reproducibility

I plan to implement the libraries (Magics and Magpye) in Python with standard libraries like Numpy and Matplotlib. I plan to make the code fairly easy to use and extend through code reviews and documentation.

I plan to follow the standard Yahoo Reproducibility Guidelines [4] and the practical guidelines specified in the Facebook Reproducibility Checklist [5] to ensure reproducibility in the implementations.

[7] About Me

I am an incoming PhD (starting September 2022) student at the Department of Psychology at Northeastern University, Boston, USA. I currently work (based in India) as a Research Assistant at Northeastern University and as a Project Assistant at National Brain Research Centre, India. My work is on the intersection of Vision Science, Neuroscience and Machine Learning with various scientific and clinical applications. Before these positions, I was a Machine Learning Engineer at Baylor College of Medicine, USA and a Research Intern at TCS Research, India and Institute of Molecular Genetics at Montpellier, CNRS, France.

I have a Bachelor's of Engineering in Computer Science from BITS Pilani, Goa Campus (Graduated in December 2019). Through these academic and industrial experiences and my formal background, I am highly trained in programming in various languages, including C++, Python and Ruby. I have also published multiple papers involving Neuroscience and Deep Learning. One of them, titled "AvaTr: One-Shot Speaker Extraction with Transformers", was published at InterSpeech 2021.

I am also highly interested in Open Source Development, especially building libraries for data visualisation. I was the lead developer for the library Rubyplot [6] (https://github.com/alishdipani/rubyplot/wiki), which aimed to be the defacto plotting library for the programming language Ruby, inspired by Matplotlib. My efforts in the same were supported by Google (through Google Summer of Code 2019 [7]) and The Ruby Association, Japan (through the Ruby Association Grant 2019 [8]). Through this experience, I have a high familiarity with the internal workings of Matplotlib as I have successfully replicated various parts of it to Rubyplot.

More details are as follows:
- Name: Alish Dipani
- Email: alish.dipani@gmail.com
- Website: https://alishdipani.github.io/
- CV: Alish_Dipani_CV.pdf
- Github: https://github.com/alishdipani
- Google Scholar: https://scholar.google.com/citations?user=i028n20AAAAJ&hl=en
- LinkedIn: https://www.linkedin.com/in/alish-dipani/
- Twitter: https://twitter.com/alu57202

[8] References

1. Magics: https://github.com/ecmwf/magics-python
2. Matplotlib: https://github.com/matplotlib/matplotlib
3. Magpye: https://github.com/ecmwf/magpye
4. Yahoo Reproducibility Guidelines: https://github.com/yahoo/ml-reproducibility-guidelines
5. Facebook Reproducibility Checklist: https://ai.facebook.com/blog/how-the-ai-community-can-get-serious-about-reproducibility
6. Rubyplot: https://github.com/alishdipani/rubyplot
7. GSoC 2019 project: https://summerofcode.withgoogle.com/archive/2019/projects/6622714041729024
8. The Ruby Association Grant 2019 announcement: https://www.ruby.or.jp/en/news/20191031