MagGeo - Annotate geo-magnetic satellite data with GPS trajectories

Data-fusion tool to combine geo-magnetic data with GPS trajectories

Fernando Benitez-Paez

2025-07-27

Table of contents

Calculation of Magnetic Components   57

How MagGeo Works   59

III. Notebooks   63

MagGeo - Sequential Mode   65

Table of contents

What's MagGeo

MagGeo is a tool that helps ecologists or animal movement researchers to link earth's magnetic field data from satellite source to GPS trajectories. Inspired by the Environmental Data Automated Track Annotation System (Env-DATA) Service a tool from Movebank and help researcher to get a better understanding about the geomagnetic variations across the GPS trajectories.

MagGeo is entirely built-in python and includes a two Jupyter Notebooks that offer two ways to link GPS tracks with the geomagnetic components using the data from one of the up-to-date satellite sources - Swarm Constellation. MagGeo will create an enriched GPS track with the following components:

- **Latitude** from the GPS Track.
- **Longitude** from the GPS Track.
- **Timestamp** from the GPS Track.
- **Magnetic Field Intensity** mapped as Fgps in nanoTeslas (nT).
- **N (Northwards) component** mapped as N in nanoTeslas (nT).
- **E (Eastwards) component** mapped as E. in nanoteslas (nT).
- **C (Downwards or Center)** component mapped as C in nanoTeslas (nT).
- **Horizontal component** mapped as H in nanoTeslas (nT).
- **Magnetic Declination or dip angle** mapped as D in degrees
- **Magnetic Inclination** mapped as I in degrees
- **Kp Index** mapped as kp
- **Total Points** as the amount of Swarm measures included in the ST-IDW process from the trajectories requested in the three satellites.

- **Minimum Distance** mapped as MinDist, representing the minimum distance amount the set of identified point inside the Space Time cylinder and each GPS point location.
- **Average Distance** mapped as AvDist, representing the average distance amount the set of distances between the identified Swarm Point in the Space Time cylinder and the GPS Points location.

Researchers, particularly ecologists now can study the annotated table to analyze the geomagnetic Spatio-temporal variation across any GPS trajectory.

Suggestions?

MagGeo is work in progress and we are constantly making improvements that you can follow up with the commits made in the pubic GitHub repo. For general inquiries, scientific concepts, suggestions please email: Fernando.Benitez@st-andrews.ac.uk, ud2@st-andrews.ac.uk, jed.long@uwo.ca

For **errors**, or **improvements** please submit an issue in this repo, describing the problem.

About MagGeo

Inspired by The Environmental Data Automated Track Annotation System (Env-DATA) Service a tool on Movebank, where ecologists and animal movement researchers all over the world can link movement data with global environmental datasets. Including hundreds of variables from a diverse set of data sources including the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the US National Oceanic and Atmospheric Administration (NOAA), and others. EnvData allow researchers to annotate in space and time multiples environmental information to enrich their GPS tracks to analyze the influence of several environmental variables in the trajectory. Using the Env-DATA Track Annotation Service registered users on MoveBank are able to get environmental parameters—such as wind conditions, land use, vegetation, and snow cover—for the whole world. Using different interpolation methods users can include multiple environment variables selecting from a comprehensive list of datasets (you can browse the available dataset here).

The second element that inspired **MagGeo** is having a tool to help researcher to get a better understanding over how the earth's magnetic field is being used by birds as one of their navigational strategies. Despite of there are several approach in this regards we know still have little knowledge about how birds can use the influence of the magnetic field for their migration patterns, especially for those long-distance migratory animals. Other studies have been reflecting into the magnetic field influence based on magnetic field estimation models, or using some displacement experiments with particular species. The disadvantage of those previous studies is the magnetic field is a highly dynamic force that have different impact

around the earth every day. Having said that MagGeo wants to take advantage of what is considered best survey of the geomagnetic field and its temporal evolution - Swarm Constellation. Swarm is a ESA's magnetic field mission, launched on 22 November 2013, consists of the three identical **Swarm satellites** (**A**lpha, **B**ravo, and **C**harlie). Swarm A and C flying side-by-side (1.4° separation in longitude) at an altitude of 462 km (initial altitude) and Swarm B at higher orbit of 511 km (initial altitude) are equipped with the following set of identical instruments.

The data products available from Swarm are Level 1b and Swarm Level 2 products. These products include Swarm magnetic field models, ionospheric and thermospheric products, and others. MagGeo use the Swarm Level 1b data product as the corrected and formatted output from each of the three Swarm satellites. For more information about the Swarm Data Products click here.

MagGeo has been deployed using a set of Jupyter notebooks a powerful tool to run a python environment. Completely build in python 3.8 MagGeo is a well described program that will guide you through several steps to annotate your GPS trajectories with the geomagnetic field components reported by Swarm. You can access to Swarm Data products via HTTP or FTP using :

- via any HTTP browser at http://swarm-diss.eo.esa.int
- directly via an ftp client at ftp://swarm-diss.eo.esa.int

However **MagGeo** use **VirES** (Virtual environments for Earth Scientists) a platform for data & model access, analysis, and visualisation for ESA's magnetic mission **Swarm**. This is a powerful client with the viresclient API that provide several classes and methods defined in the vires client package. The `viresclient` Python package allows you to connect to the VirES server to download Swarm data and data calculated using magnetic models.

Part I.

How to use it

1. Installation

Installing MagGeo

MagGeo is a specialized tool that helps researchers analyze magnetic field data in relation to animal movement patterns. This guide will walk you through installing MagGeo on your computer, even if you're more familiar with R than Python.

1.1. What You'll Need

Before we begin, understand that **MagGeo** is currently built using Python (similar to how some R packages depend on specific R versions). We'll set up a contained environment so it won't interfere with any existing R installations or other software on your computer.

1.2. Step 1: Install Miniconda (Your Python Environment Manager)

What is Miniconda? Think of Miniconda as a package manager for Python, similar to how CRAN manages R packages. It creates isolated environments so different projects don't conflict with each other.

1.2.1. For Windows Users:
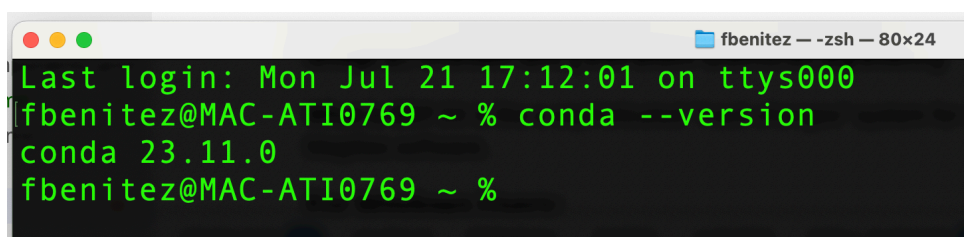
1. Visit https://docs.conda.io/en/latest/miniconda.html
2. Download the "Miniconda3 Windows 64-bit" installer
3. Run the downloaded `.exe` file as administrator
4. Follow the installation wizard:

   - Accept the license agreement
   - Choose "Just Me" installation
   - Use the default installation location
   - **Important:** Check "Add Miniconda3 to my PATH environment variable" (even if it shows a warning)

5. Click "Install" and wait for completion

1.2.2. For macOS Users:

1. Visit https://docs.conda.io/en/latest/miniconda.html
2. Download the "Miniconda3 macOS 64-bit pkg" installer
3. Double-click the downloaded `.pkg` file
4. Follow the installation wizard using default settings
5. The installer will automatically add conda to your PATH

Verification: After installation, open a new terminal (macOS) or Command Prompt (Windows) and type `conda --version`. You should see a version number.

1.3. Step 2: Clone MagGeo

You will need to clone the MagGeo repository in your local computer (we are currently working to make make MagGeo available in `pip` to an easier installation), for now simply clone it and you will have access to all MagGeo.

If you don't have **Git**, or you are not use to work with version control, you will need to install git to clone any repository from GitHub

> 💡 Installing Git
>
> **What is Git?** Git is like a sophisticated "track changes" system for code, similar to version control in collaborative documents. We need it to download the MagGeo software.
>
> ### 1.3.1. For Windows Users:
>
> 1. Visit https://git-scm.com/downloads
> 2. Download "Git for Windows"
> 3. Run the installer with these recommended settings:
>
>   - Select "Use Git from the Windows Command Prompt"
>   - Choose "Checkout Windows-style, commit Unix-style line endings"
>   - Use all other default settings
>
> ### 1.3.2. For macOS Users:
>
> **Option A (Recommended):**
> 1. Open Terminal (found in Applications > Utilities)
> 2. Type `git --version` and press Enter
> 3. If **Git** isn't installed, macOS will prompt you to install Xcode Command Line Tools

4. Click "Install" and follow the prompts
Option B:
1. Visit https://git-scm.com/downloads
2. Download "Git for macOS"
3. Run the installer with default settings

1.3.3. For Both Windows and macOS Users:

1. **Open your command line interface:**

   - **Windows:** Press `Windows + R`, type `cmd`, press Enter
   - **macOS:** Press `Cmd + Space`, type "Terminal", press Enter

2. **Navigate to your desired location** (optional but recommended):

```
# Windows users:
cd C:\Users\YourUsername\Documents

# macOS users:
cd ~/Documents
```

3. **Download MagGeo:**

```
git clone https://github.com/MagGeo/MagGeo-Annotation-Program.git
```

   This creates a folder called "**MagGeo-Annotation-Program**"
   with all the necessary files.

1.4. Step 4: Navigate to the MagGeo Folder

```
cd MagGeo-Annotation-Program
```

14

1.5. Step 5: Create the MagGeo Environment

This creates a specialized Python environment with all the specific packages MagGeo needs, similar to installing a comprehensive R package with all its dependencies.

```
conda env create --file environment.yml
```

> ⚠️ Warning
>
> **This process will take 5-10 minutes** as it downloads and installs all required components. You'll see progress messages - this is normal.

1.6. Step 6: Activate the MagGeo Environment

Every time you want to use MagGeo, you'll need to "activate" its environment first:

```
conda activate MagGeoEnv
```

What this does: Switches your command line to use the MagGeo-specific Python environment, ensuring all the right tools are available.

1.7. Step 7: Set Up VirES Access (Magnetic Field Data)

What is VirES? VirES provides access to satellite magnetic field data from ESA's Swarm mission. Think of it as a specialized database for mag-

15

netic field measurements that MagGeo uses to understand the magnetic environment your animals experienced.

MagGeo use **VirES** (Virtual environments for Earth Scientists) a platform for data & model access, analysis, and visualization for ESA's magnetic mission **Swarm**. This is a powerful client with the viresclient API that provide several classes and methods defined in the vires client package. The `viresclient` Python package allows you to connect to the VirES server to download Swarm data and data calculated using magnetic models.

1.7.1. Create Your VirES Account:

1. **Sign up:** Visit https://vires.services/oauth/accounts/signup/

   - Use your institutional email if possible
   - Create a secure password
   - Verify your email address

2. **Log in:** Go to https://vires.services/

3. **Get your access token:**

   - Follow the detailed instructions at https://viresclient. readthedocs.io/en/latest/access_token.html
   - Your token is like a password that allows MagGeo to download data for you
   - **Important:** Keep this token secure and don't share it

4. **Configure MagGeo with your token:**
   ```
   # Replace YOUR_TOKEN_HERE with your actual token
   viresclient set_token --token YOUR_TOKEN_HERE
   ```

1.8. Testing Your Installation

To verify everything is working:

1. Make sure you're in the MagGeo directory and the environment is activated
2. Try running a simple test (specific commands will depend on MagGeo's documentation)

1.9. Troubleshooting Common Issues

1.9.1. Windows-Specific Issues

Problem: "conda is not recognized as an internal or external command"

- **Solution:** Miniconda wasn't added to PATH during installation

- **Fix:** Reinstall Miniconda and ensure you check "Add Miniconda3 to my PATH environment variable"

Problem: "Permission denied" errors

- **Solution:** Run Command Prompt as Administrator

- **Fix:** Right-click Command Prompt and select "Run as administrator"

Problem: Long path names causing issues

- **Solution:** Install MagGeo in a shorter path like `C:\MagGeo`

1.9.2. macOS-Specific Issues

Problem: "command not found: conda"

- **Solution:** Terminal doesn't recognize conda
- **Fix:** Close and reopen Terminal, or run `source ~/.bash_profile`

Problem: "Permission denied" during Git clone

- **Solution:** You don't have write permissions in the current directory
-
- **Fix:** Navigate to your home directory first: `cd ~`

Problem: Xcode Command Line Tools installation fails

- **Solution:** Install Xcode from the App Store first, then try again

1.9.3. General Issues

Problem: Environment creation fails with package conflicts

- **Solution:** Clear conda cache and try again
- **Fix:** Clean your conda enviroment cache and reinstall the MagGeo environment.

`conda clean --all`

`conda env create --file environment.yml`

Problem: VirES token authentication fails

- **Solution:** Token might be incorrect or expired

- **Fix:** Generate a new token from the VirES website and reconfigure

Problem: Git clone fails with "Repository not found"

- **Solution:** Check your internet connection and the repository URL

- **Fix:** Try cloning again, or download the ZIP file directly from GitHub

1.10. Reporting an Issue

If you encounter issues not covered here:

1.10.1. Before Reporting an Issue:

1. Note your operating system (Windows 10/11, macOS version)
2. Copy any error messages exactly as they appear
3. Note which step in the installation process failed

1.10.2. How to Report an Issue:

1. Visit   https://github.com/MagGeo/MagGeo-Annotation-Program/ issues
2. Click "New Issue"
3. Choose "Bug report" or "Installation help"
4. Provide a clear title like "Installation fails at Step 5 on Windows 11"
5. Include:

   - Your operating system
   - The exact error message
   - What you were trying to do when the error occurred
   - Screenshots if helpful

19

1.11. Next Steps

Once installation is complete:

- Read the MagGeo user guide for analysis workflows

- Prepare your animal movement data according to the format requirements

> 💡 Tip
>
> Remember: You only need to install MagGeo once, but you'll need to activate the environment (`conda activate MagGeoEnv`) each time you want to use it.

Run MagGeo using the sample data

MagGeo can be executed using the same terminal you have been using in the previous steps. If you want to get familiar with MagGeo and get an annotated GPS trajectory using the data we have included as an example (data folder), run the following command (replace your virES token where is requiered):

```
python MagGeo_main.py -p parameters/default.yml --token YOUR_TOKEN_HERE
```

Now MagGeo will start to download the Swarm Data.

```
(base) fbenitez@MAC-ATI0769 MagGeo-Annotation-Program % poetry run python MagGeo_SA.py -p parameters/default.yml
Setting access token for https://vires.services/ows  ...
Generate a token at https://vires.services/accounts/tokens/
Enter token:
Token saved for https://vires.services/ows
--
Reading parameters file: parameters/default.yml
--
Getting Swarm Data: 100%|
Annotating the GPS Trayectory:  27%|
```

Once the data has been downloaded, MagGeo will process it to make the annotation process (for more information about how this is done, visit our methodological paper in Movement Ecology)

The last step MagGeo does is annotating the gathered data, that would take more time depending how big is your dataset. In our example it only takes 4 seconds.

Run MagGeo using the sample data

```
(base) fbenitez@MAC-ATI0769 MagGeo-Annotation-Program % poetry run python MagGeo_SA.py -p parameters/default.yml
Setting access token for https://vires.services/ows  ...
Generate a token at https://vires.services/accounts/tokens/
Enter token:
Token saved for https://vires.services/ows
--
Reading parameters file: parameters/default.yml
--
Getting Swarm Data:  40%|████████████████████████████████
```

And **Congrats you got annotated data**. The results will be stored in the folder results for your futher analysis. You will find a .csv file named like **GeoMagResult_+name_of_your_csv_file_trajectory.**

Run MagGeo using your data.

If you are ready to annotate your GPS trajectories. You need to update the parameters file in MagGeo to let the program know what are the correct values of your data.

- Copy the csv file with your trajectories into the data folder.
- Open and Update the following parameters in the file `default.yml` located in parameters folder:
- `gpsfilename: "name_of_your_trajectory.cvs"` Include the name of your Input data. The GPS trajectory you need to annotate with the geomagnetic satellite data.
- `Lat: "latitude_column_name_in_your_trajectory"`
- `Long: "longitude_column_name_in_your_trajectory"`
- `DateTime: "Date_Time_column_name_in_your_trajectory"` make sure you have one column that includes Date and Time values together.
- `altitude: "altitude_column_name_in_your_trajectory"` if you do not have any altitute column, you can leave that in blank, including only `""`

Save your changes, return to the Terminal and run:

```
python MagGeo_main.py -p parameters/default.yml --token YOUR_TOKEN_HERE
```

Run MagGeo step by step, using Jupyter Notebook.

MagGeo includes a set of Jupyter Notebooks, you will find four notebooks (.ipynp) in the Notebooks folder.

In a Terminal, make sure you are using `MagGeoEnv` environment, and run:

```
jupyter notebook
```

A Jupyter Notebook dashboard will comes out in your browser locally (e.g. http://localhost:8888) then you can explore MagGeo and its content. Go to Notebooks folder and open any of the following notebook for a step by step process. You can add cells to make your own test or analysis, but be aware that any change you do at the code migth affects the correct performance of the program.

Run MagGeo step by step, using Jupyter Notebook.

- Main Notebook : An initial and descriptive notebook where you can get detail information about MagGeo, the sample data used, background concepts and software requirements.
- Sequential Mode: Annotation Notebook applying a sequential mode. Using a traditional loop to going through the GPS track rows and process every row computing the magnetic components. Particularly useful for small datasets.
- Parallel Mode: If you have a "big" dataset (e.g. 1 million of records) you might want try the parallel mode. The parallel mode has some differences when you run the required libraries in a windows or Linux environment. We have tested **MagGeo** in a windows server environment.
- Notebook basics: If you are not familiar with Jupiter Notebooks and want to learn about the basics over how to run the notebooks before try the annotate tool. You can try this notebook to get the basics elements to run cells, read data, and plot some a basic chart.

The following image will help you to understand how the sequential and parallel mode differ, and how in parallel mode you should be able to use the full capacity of your machine. However it is quite important to identify

when we need to use a parallel mode. For small datasets running **MagGeo** in Parallel mode could be even slower than the sequential mode.

Data requirements

> **!** Important
>
> Your trajectory must be in a csv format:
> There are three columns that must be included in your GPS trajectory. Make sure your GPS trajectory includes **Latitude** , **Longitude** and **timestamp**. We suggest that the Timestamp column follow the day/month/year Hour:Minute (**dd/mm/yyyy HH:MM:SS**) format, Latitude and Longitude should be in decimal degrees (WGS84). If you have a **altitude** attribute, make sure that the units are in kilometers. Other columns will be ignored. Here it is an example of how your GPS track should looks like.
> For this example we are reading the **BirdGPSTrajectory.csv** file. If you want to run the method using your own csv file, make sure you store your the file in the `/data` folder. For more information about the dataset we used in this example go to the Main Notebook.

Dataset used as use-case

MagGeo comes with a dataset for the use case you see in the notebook outputs. This dataset comes from the MoveBank Data Repository (https://www.datarepository.movebank.org/)[1][2].

We annotated tracking data of greater white-fronted geese (Anser albifrons), which migrate between northern Germany and the Russian Arctic. We annotated data for fifteen individuals in a single autumn migration (42 days) a total of 973 GPS location.

[1] Kölzsch A, Müskens GJDM, Kruckenberg H, Glazov P, Weinzierl R, Nolet BA, Wikelski M (2016) Towards a new understanding of migration timing: slower spring than autumn migration in geese reflects different decision rules for stopover use and departure. Oikos. doi:10.1111/oik.03121

[2] Kölzsch A, Kruckenberg H, Glazov P, Müskens GJDM, Wikelski M (2016) Data from: Towards a new understanding of migration timing: slower spring than autumn migration in geese reflects different decision rules for stopover use and departure. Movebank Data Repository. doi:10.5441/001/1.31c2v92f

Part II.

The science behind

About MagGeo

Inspired by The Environmental Data Automated Track Annotation System (Env-DATA) Service a tool on Movebank, where ecologists and animal movement researchers all over the world can link movement data with global environmental datasets. Including hundreds of variables from a diverse set of data sources including the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the US National Oceanic and Atmospheric Administration (NOAA), and others. EnvData allow researchers to annotate in space and time multiples environmental information to enrich their GPS tracks to analyze the influence of several environmental variables in the trajectory. Using the Env-DATA Track Annotation Service registered users on MoveBank are able to get environmental parameters—such as wind conditions, land use, vegetation, and snow cover—for the whole world. Using different interpolation methods users can include multiple environment variables selecting from a comprehensive list of datasets (you can browse the available dataset here).

The second element that inspired **MagGeo** is having a tool to help researcher to get a better understanding over how the earth's magnetic field is being used by birds as one of their navigational strategies. Despite of there are several approach in this regards we know still have little knowledge about how birds can use the influence of the magnetic field for their migration patterns, especially for those long-distance migratory animals. Other studies have been reflecting into the magnetic field influence based on magnetic field estimation models, or using some displacement experiments with particular species. The disadvantage of those previous studies is the magnetic field is a highly dynamic force that have different impact

around the earth every day. Having said that MagGeo wants to take advantage of what is considered best survey of the geomagnetic field and its temporal evolution - Swarm Constellation. Swarm is a ESA's magnetic field mission, launched on 22 November 2013, consists of the three identical **Swarm satellites** (**A**lpha, **B**ravo, and **C**harlie). Swarm A and C flying side-by-side (1.4° separation in longitude) at an altitude of 462 km (initial altitude) and Swarm B at higher orbit of 511 km (initial altitude) are equipped with the following set of identical instruments.

The data products available from Swarm are Level 1b and Swarm Level 2 products. These products include Swarm magnetic field models, ionospheric and thermospheric products, and others. MagGeo use the Swarm Level 1b data product as the corrected and formatted output from each of the three Swarm satellites. For more information about the Swarm Data Products click here.

MagGeo has been deployed using a set of Jupyter notebooks a powerful tool to run a python environment. Completely build in python 3.8 MagGeo is a well described program that will guide you through several steps to annotate your GPS trajectories with the geomagnetic field components reported by Swarm. You can access to Swarm Data products via HTTP or FTP using :

- via any HTTP browser at http://swarm-diss.eo.esa.int
- directly via an ftp client at ftp://swarm-diss.eo.esa.int

However **MagGeo** use **VirES** (Virtual environments for Earth Scientists) a platform for data & model access, analysis, and visualisation for ESA's magnetic mission **Swarm**. This is a powerful client with the viresclient API that provide several classes and methods defined in the vires client package. The `viresclient` Python package allows you to connect to the VirES server to download Swarm data and data calculated using magnetic models.

2. Key Concepts

1. Animal Navigation Fundamentals

This is not a comprehensive book related to animal navigation fundamental, however there are certain concepts you might want to recall and understand that establish the foundations of what you have built **MagGeo**.

2.0.1. True Navigation

Long-distance migratory navigation represents one of nature's most remarkable phenomena, consisting of two primary components that work together to enable animals to travel vast distances with inexplicable precision. The first component, is **compass orientation**, involves determining the correct direction of movement relative to environmental cues (where to go). The second component, **geographic positioning**, requires animals to know their current location at any specific time during their journey (where am I)[1].

These mechanisms together support what Ecologists term "**true navigation**" - defined as the ability to find the way to a distant, unknown location using only cues that are available locally at the animal's current position[2].

2.0.2. Navigation Mechanisms

Migratory animals (e.g Birds) have evolved sophisticated sensory systems that can detect and interpret multiple environmental cues for navigation purposes [@demsar2025]. These cues fall into two primary categories based on their navigational function:

For Compass Orientation: Animals use several types of compass systems to maintain consistent directional movement:

- **Solar compass**: Utilizes the Sun's position, accounting for its apparent movement across the sky and seasonal variations

- **Stellar compass**: Relies on star patterns and celestial rotation, particularly useful during nighttime migration

- **Polarized light patterns**: Detects polarization patterns in skylight that remain consistent relative to the Sun's position

- **Earth's magnetic field**: Uses the planet's magnetic field as a reliable directional reference[3]

For Geographic Positioning: To determine their specific location, animals can access several types of positional cues:

- **Earth's magnetic field**: Utilizes spatial gradients and patterns in Earth's magnetic field intensity and inclination[3]

- **Olfactory cues**: Chemical signatures that vary geographically, creating olfactory landscapes

- **Visual landmarks**: Topographical features, coastlines, and other geographical markers

- **Infrasound**: Low-frequency sound waves that can travel long distances and vary by location[1]

2.1. 2. Geomagnetic Navigation

Geomagnetic navigation represents a strategy whereby animals use information derived from Earth's magnetic field for either compass orientation, geographic positioning, or both functions simultaneously[3]. This navigational mechanism has been documented across an taxonomic range, suggesting it evolved independently multiple times or represents an ancient, conserved ability.

Evidence for geomagnetic navigation exists across diverse taxa:

- **Birds**: From songbirds to seabirds, with extensive experimental evidence[1]

- **Fish**: Including both oceanic and freshwater species

- **Sea turtles**: Particularly well-documented in marine navigation [1]

- **Terrestrial mammals**: From small rodents to large ungulates[11] [12]

- **Marine mammals**: Including whales and dolphins[13] [1]

However the study of geomagnetic navigation employs two primary methodological approaches, each with distinct advantages and limitations:

Lab Experiments: Controlled studies where animals are placed in artificial magnetic field to examine how magnetic parameters influence the onset and direction of migratory behavior. Laboratory experiments provide exceptional precision and experimental control, allowing researchers to manipulate specific magnetic parameters while holding other variables constant[2] [1] [1]. However, the observed behaviors in such controlled environments differ significantly from what occurs in wild[1]. Additionally, these studies typically focus on small numbers of individuals from single species, which limits the results across multiple species[1].

Field Studies with Tracking: Modern migration research employs tracking technologies, particularly GPS tags combined with displacement experiments. These approaches provide greater ecological relevance by studying animals in their natural environments[1] [2] [21]. However, most tracking studies have been limited by their reliance on static representations of the geomagnetic field (space and temporal), failing to account for the dynamic temporal variations that may influence animal navigation[22].

2.2. 3. Earth's Magnetic Field

Earth's magnetic field is a complex, dynamic system that results from the superposition of magnetic fields originating from multiple sources. Understanding this complex environmental variable is crucial for comprehending how animals use magnetic data for navigation. The total magnetic field at any location and time results from the combination of three primary components, each with different characteristics, spatial patterns, and temporal behaviors[3] .

Core Field: The dominant contributor to Earth's magnetic field is the core field, generated by the geodynamo mechanism operating within Earth's outer liquid core. This field arises from the motion of electrically conductive molten iron as it circulates within the outer core, driven by thermal convection and the planet's rotation. The core field exhibits a broadly dipolar structure on large scales, resembling that of a giant bar magnet tilted approximately 11 degrees from Earth's rotational axis. However, this apparent simplicity masks considerable complexity in the detailed structure, with significant non-dipolar components that create regional variations in field strength and direction across the globe.

The core field dominates the total magnetic field at Earth's surface, typically contributing over 98% of the field strength at any location. This field undergoes gradual changes over time scales of years to decades, known as

secular variation, as the circulation patterns within the outer core evolve. These long-term changes can amount to tens of nanoTesla per year at any given location, representing a slow but persistent evolution of the magnetic environment that animals experience.

Lithospheric Field: The lithospheric or crustal field originates from the magnetic properties of rocks within Earth's crust. Certain minerals, particularly magnetite and other iron-bearing compounds, became magnetized when they formed and retain this magnetization over geological time scales. The lithospheric field creates local and regional magnetic anomalies - areas where the magnetic field strength or direction differs noticeably from what would be expected from the core field alone.

These magnetic anomalies can range from subtle variations of a few nanoTesla to dramatic variations of thousands of nanoTesla over distances of kilometers to hundreds of kilometers. The lithospheric field is considered essentially static over time periods relevant to animal lifespans, changing only over geological time scales of millennia or longer. For migratory animals, these crustal magnetic signatures provide potentially stable, location-specific magnetic features that could serve as magnetic landmarks or components of magnetic maps.

External Fields: External magnetic fields result from electrical current systems flowing in Earth's ionosphere and magnetosphere, driven by interactions between the solar wind and Earth's magnetic field. These fields exhibit the most dynamic behavior of the three components, varying on time scales from seconds to days. The external field contributions include both regular, predictable daily variations and irregular disturbances associated with space weather events.

The regular daily variations, known as Sq (solar quiet) currents, result from the regular heating of the ionosphere by solar radiation, which creates predictable current systems that vary with local time and season. These regular variations typically amount to 20-30 nanoTesla at mid-latitudes but can be significantly larger at high latitudes near the magnetic poles.
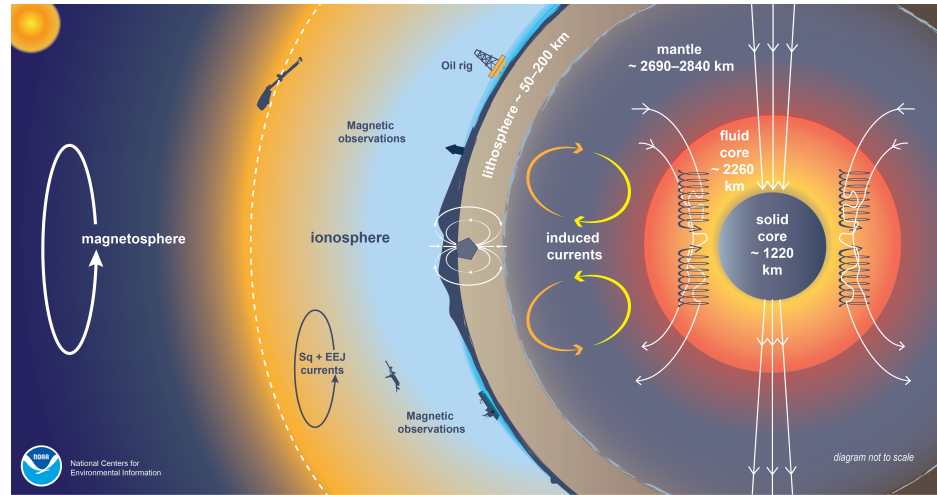
Figure 1 showing magnetic field components and coordinate system, NOOA, https://www.ncei.noaa.gov/news/HDGM

2.2.1. Magnetic Field Measurements and Coordinate Systems

Scientists measure Earth's magnetic field using a standardized coordinate system that allows for consistent description and comparison of magnetic field vectors worldwide. The magnetic field vector **B** is decomposed into components using the North-East-Centre (NEC) coordinate system, also known as the local geodetic coordinate system.

In this system, measurements are made relative to the local geographic reference frame at each point on Earth's surface. The North component points toward geographic north (true north), the East component points toward geographic east, and the Centre component points radially downward toward Earth's center. This coordinate system provides a consistent framework for describing magnetic field variations across the globe and through time.

From these three orthogonal components, several important magnetic field parameters are derived:

- **Intensity (F)**: The magnitude or total strength of the magnetic field vector B, representing the length of the vector in three-dimensional space

- **Inclination (I)**: The angle between the magnetic field vector B and its horizontal component H, measuring how steeply the field lines plunge into or emerge from Earth

- **Declination (D)**: The angle between the horizontal component H and geographic north, indicating how much magnetic north deviates from geographic north

- **Horizontal component (H)**: The horizontal projection of the magnetic field vector, representing the component parallel to Earth's surface

These parameters provide different types of information that animals might potentially use for navigation. Intensity varies systematically with latitude and can provide positional information, while inclination also varies predictably with latitude and could serve as a magnetic latitude indicator. Declination provides information about magnetic versus geographic north, potentially useful for compass orientation.

2.3. 4. Spatial and Temporal Variation

2.3.1. Global Spatial Patterns

Intensity: The total magnetic field intensity shows global patterns that reflects the underlying dipolar structure of the core field. At the magnetic equators, where the field lines run horizontally, the total intensity reaches its minimum values of approximately *23,000 nanoTesla*. Moving toward

higher magnetic latitudes, the field intensity gradually increases as the field lines become more vertical, reaching maximum values of approximately *62,000 nanoTesla* near the magnetic poles.

This variation in intensity with magnetic latitude creates what researchers term "isodynamic lines" - contours of equal magnetic intensity that generally run east-west but show considerable deviation from simple latitudinal bands due to non-dipolar components of the field. These intensity gradients are steep enough that animals traveling north-south distances of hundreds of kilometers would encounter measurable changes in field intensity.

Inclination: Magnetic inclination, the angle at which field lines plunge into Earth, provides perhaps the most systematic navigational cue available from the geomagnetic field. At the magnetic equator, field lines run horizontally, giving inclination values near zero degrees. Moving toward the magnetic poles, inclination increases progressively, reaching 90 degrees (vertical) at the magnetic poles themselves.

This relationship between inclination and magnetic latitude has led researchers to propose that inclination serves as a primary magnetic coordinate for animal navigation. The inclination gradient is steep enough that latitudinal movements of tens of kilometers produce detectable changes in inclination angle, potentially providing fine-scale positional information.

Declination: Magnetic declination on the other side, the deviation of magnetic north from geographic north, shows the most complex spatial patterns of the major magnetic field components. Unlike intensity and inclination, declination does not follow simple latitudinal patterns but instead shows regional variations that reflect the non-dipolar components of the core field and crustal magnetic anomalies.

In some regions, declination changes very gradually over large distances, while in other areas it can change by several degrees over distances of hundreds of kilometers. These declination patterns create distinctive regional

magnetic signatures that could potentially serve as magnetic landmarks for animal navigation systems.
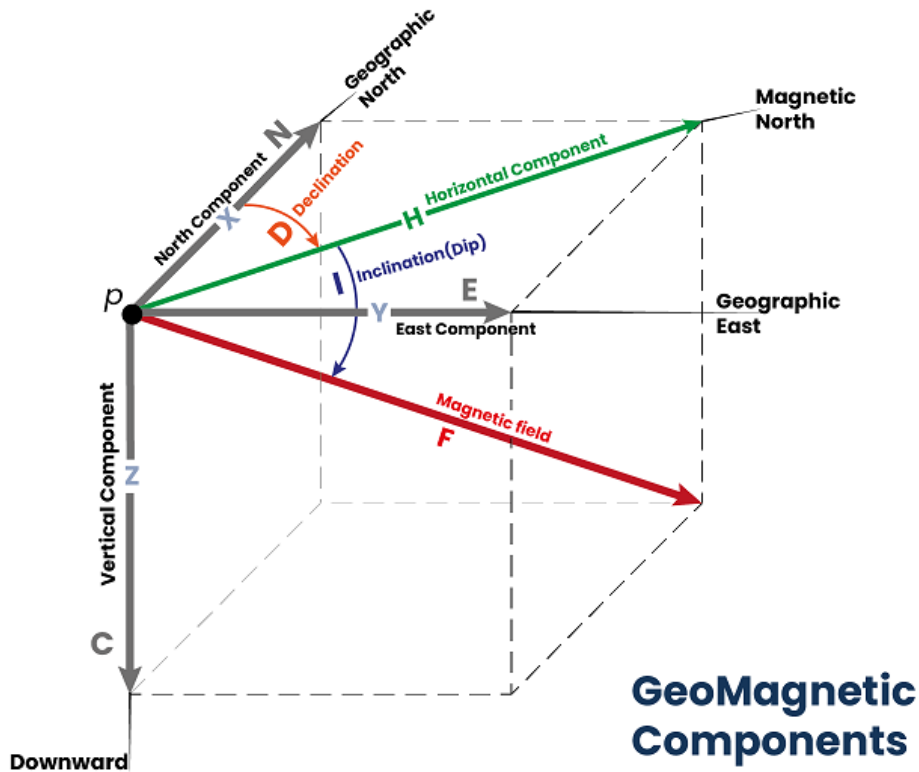
2.3.2. Temporal Variations

The temporal behavior of Earth's magnetic field spans an enormous range of time scales, from rapid variations lasting seconds to gradual changes occurring over geological time. Understanding these temporal variations is crucial for reading how animals use magnetic data for navigation.

Quiet-Time Variations: Even during geomagnetically quiet periods, Earth's magnetic field shows regular daily variations driven by solar heating of the ionosphere. During daylight hours, solar ultraviolet radiation ionizes atmospheric gases, creating electrical conductivity that allows current systems to flow in the ionosphere. These currents generate magnetic fields that add to the main field measured at Earth's surface.

The amplitude of these quiet-time variations depends strongly on latitude and local time. At mid-latitudes, typical daily variations amount to approximately 20 nanoTesla, with maximum variations occurring during local noon when ionospheric currents are strongest. At polar latitudes, where ionospheric current systems are more complex, daily variations can reach 100 nanoTesla or more.

These regular variations follow *predictable* patterns that repeat daily and show seasonal modulations. For animals using magnetic cues for navigation, these predictable variations might either be filtered out by their sensory systems or potentially used as additional timing information.

Geomagnetic Storm Disturbances: During periods of enhanced solar activity, Earth's magnetic field can experience big disturbances that far exceed normal quiet-time variations. Geomagnetic storms occur when disturbances in the solar wind create enhanced coupling between the solar wind's magnetic field and Earth's magnetosphere. These events drive intense electrical currents in the magnetosphere and ionosphere, creating magnetic field variations that can exceed *1000 nanoTesla i*n polar regions and *250 nanoTesla at mid-latitudes*[23].

The temporal characteristics of geomagnetic storms are particularly relevant for animal navigation. Storm-related magnetic field changes can occur within seconds to days, creating rapid alterations in the magnetic environment that animals experience. The magnitude of these changes often exceeds the established sensitivity thresholds of animal magnetoreceptors by factors of ten or more, suggesting that geomagnetic storms could significantly disrupt magnetic navigation systems.

Secular Variation: On longer time scales, Earth's magnetic field undergoes continuous slow changes known as secular variation, driven by evolving circulation patterns within the liquid outer core. These changes typically amount to tens of nanoTesla per year at any given location, representing a gradual evolution of the magnetic field over time scales of years to decades.

For long-lived migratory animals or populations with strong site fidelity, secular variation could potentially require periodic recalibration of magnetic navigation systems. However, the slow rate of secular variation means that it is unlikely to affect navigation during individual migratory journeys, which typically last weeks to months.

2.4. 5. Geomagnetic Storms and Solar Wind

2.4.1. Storm-Related Disruption

The sensitivity thresholds established by neurophysiological and behavioral research have profound implications when compared to the magnitude of magnetic field variations that occur during geomagnetic storms. Solar wind disturbances can create magnetic field changes exceeding 1000 nanoTesla within minutes during major geomagnetic storms[2], representing variations that are 5-65 times larger than established animal sensitivity thresholds.

Potential Disruption Mechanisms: Animals using magnetic intensity as a positional cue could experience significant navigation errors during geomagnetic storms. If birds or other animals rely on specific intensity values as location markers, sudden storm-related intensity changes could create the illusion that they have been displaced to entirely different geographic locations. This effect would be similar to the responses observed in virtual magnetic displacement experiments[1] [1] [1] [2], where animals exposed

to artificial magnetic fields characteristic of distant locations attempt to compensate by changing their flight direction.

Animals using magnetic inclination or declination for compass orientation might experience disruption of their directional sensing capabilities during storms. Rapid changes in field direction could cause temporary loss of magnetic compass function, potentially forcing animals to rely on alternative compass systems such as solar or stellar compasses .

Species-Specific Responses: Different animal species likely respond to geomagnetic disturbances in various ways, depending on their particular methods of using magnetic information for navigation[3]. Some species might temporarily cease migration during severe magnetic disturbances, while others might switch to alternative navigation systems. The behavioral responses probably depend on the availability of backup navigation cues and the flexibility of each species' navigation system.

2.5. 6. Geomagnetic Data Sources

2.5.1. Terrestrial Observatory Networks

The systematic measurement of Earth's magnetic field began in the 1830s and has evolved into a sophisticated global network of ground-based observatories that provide continuous, high-precision magnetic field measurements.

INTERMAGNET Network: The International Real-time Magnetic Observatory Network (INTERMAGNET) represents the primary source of ground-based geomagnetic data, currently comprising 152 observatories distributed across the globe[3] . These observatories employ standardized instrumentation and data collection procedures to ensure consistency and comparability of measurements worldwide.

Each INTERMAGNET observatory typically operates multiple magnetometers, including absolute instruments that measure the true magnetic field vector and variometers that detect changes in the field with high temporal resolution. The observatories record magnetic field measurements continuously, typically at one-second or one-minute intervals, creating detailed time series that capture both gradual changes and rapid fluctuations.

Advantages of Terrestrial Measurements: Ground-based observatories offer several significant advantages for magnetic field research. Their calibration accuracy is exceptionally high, with measurement uncertainties typically less than 1 nanoTesla for carefully maintained instruments. The continuous operation of these stations creates long-term data records, some extending back more than a century, that are invaluable for studying secular variation and long-term magnetic field changes.

The temporal resolution of ground-based measurements is also superior to satellite data, with one-second sampling rates that can capture rapid magnetic field fluctuations associated with geomagnetic storms and other space weather events. This high temporal resolution makes ground-based data particularly valuable for studying the detailed time evolution of magnetic disturbances.

Limitations of Ground-Based Networks: Despite their high accuracy and temporal resolution, terrestrial magnetic observatories have significant limitations for global magnetic field studies. The spatial coverage is highly irregular, with dense networks in some regions (particularly northern and western Europe) but sparse or absent coverage in others. Oceanic regions are particularly poorly covered, with very few magnetic observatories on islands and none on the open ocean.

The effective spatial coverage of each observatory is limited to approximately 1000 kilometers, beyond which local magnetic variations can differ significantly from those measured at the observatory. This limitation

means that only regions with dense observatory networks provide complete spatial coverage, excluding most animal migration pathways that cross oceanic or remote continental regions.

Temporal data availability also presents challenges for real-time applications. Observatories submit their data to the central INTERMAGNET network at different times, often with delays of months to years for the highest quality, fully processed data. Some observatories occasionally cease operation due to funding or technical issues, creating gaps in temporal coverage.

2.5.2. Satellite-Based Measurements

Satellite missions have revolutionized geomagnetic field research by providing consistent global coverage with standardized instrumentation. Over the past 60 years, a series of dedicated magnetic field missions has complemented terrestrial measurements and enabled global-scale studies of magnetic field structure and dynamics.

Historical Satellite Missions: The evolution of satellite-based geomagnetism began with early missions such as POGO (1965-1971), which provided the first global magnetic field measurements from space. Subsequent missions including Magsat (1979-80), Ørsted (1999-present), CHAMP (2000-2010), and SAC-C (2000-present) progressively improved measurement accuracy and global coverage[3] .

Each mission contributed unique capabilities and extended the temporal baseline of satellite magnetic measurements. Magsat provided the first high-precision global magnetic field model, while Ørsted began the era of continuous, long-term satellite magnetic monitoring. CHAMP delivered exceptionally precise measurements that advanced understanding of both the main field and its temporal variations.

The Swarm Mission: The European Space Agency's Swarm mission, launched in 2013, represents the current state-of-the-art in satellite-based

magnetic field measurement[3] . The mission employs three identical satellites arranged in a carefully designed constellation that provides enhanced spatial and temporal sampling of the global magnetic field.

The constellation consists of satellites Alpha and Charlie, which orbit in parallel at an altitude of 480 kilometers with a separation of approximately 150 kilometers at the equator. Satellite Bravo orbits at a higher altitude of 510 kilometers in a different orbital plane that, at present, is nearly perpendicular to the Alpha-Charlie pair. This configuration provides measurements at two different altitudes and enables separation of different magnetic field sources.

Swarm Instrumentation: Each Swarm satellite carries identical scientific instrumentation designed for precise magnetic field measurement. The primary instruments include an Absolute Scalar Magnetometer (ASM) that measures the total magnetic field intensity, a Vector Field Magnetometer (VFM) that measures the three-component magnetic field vector, and a GPS receiver for precise orbit determination.

The VFM provides magnetic field measurements at 1 Hz resolution in the North-East-Centre coordinate system, calibrated using measurements from the ASM and stellar cameras that determine satellite orientation. This combination of instruments enables precise determination of both the magnitude and direction of the magnetic field vector at satellite altitude.

Advantages of Satellite Measurements: Satellite measurements resolve many of the limitations of ground-based networks by providing consistent global coverage with standardized instrumentation. The polar-orbiting design of most magnetic field satellites ensures complete global coverage within a few days, including oceanic and remote continental regions that lack ground-based observatories.

Satellite data are typically available within days to weeks of collection, much faster than the fully processed data from many ground-based observatories. The standardized instrumentation and data processing proce-

dures also ensure consistency in measurements across different geographic regions and time periods.

The orbital perspective of satellites enables measurement of the magnetic field above most ionospheric current systems, providing information about magnetic field sources that cannot be obtained from ground-based measurements alone. This capability is particularly valuable for studying external field variations and their global patterns.

Satellite Data Access and Formats: Swarm data are made freely available through the European Space Agency's Earth Observation Policy, with data access provided through the VirES (Virtual workspace for Earth observation Scientists) web platform . The Level 1b data products provide corrected and calibrated magnetic field measurements in standard scientific formats.

However, satellite magnetic data are provided in specialized formats such as the Common Data Format (CDF) developed by NASA [3], which are not readily accessible using standard ecological data analysis software. This technical barrier has historically limited the use of satellite magnetic data outside the specialized geophysics community, despite the open data policies of space agencies.

2.5.3. Geomagnetic Indices and Activity Monitoring

Understanding and quantifying the level of geomagnetic activity is crucial for both space weather applications and studies of magnetic field effects on biological systems. Geomagnetic indices provide standardized measures of magnetic field disturbance that enable comparison of activity levels across different times and locations.

K-Index System: The K-index represents the fundamental unit for measuring local geomagnetic disturbance. Calculated every three hours at individual magnetic observatories, the K-index quantifies disturbances in the horizontal component of the magnetic field on a quasi-logarithmic scale

from 0 to 9^2 . A K-index of 0 indicates perfectly calm magnetic conditions, while values of 5 or higher indicate geomagnetic storm conditions.

The K-index calculation involves comparing the observed magnetic field variations during each three-hour period to the expected quiet-day variation for that location and season. This approach accounts for the regular daily variations in the magnetic field and highlights unusual disturbances that exceed normal quiet-time levels.

Kp Index: The planetary Kp index provides a global measure of geomagnetic activity by averaging K-indices from a standardized network of magnetic observatories distributed worldwide[3] . Calculated every three hours, the Kp index ranges from 0 to 9 and serves as a proxy for the energy input from the solar wind into Earth's magnetosphere.

The Kp index is widely used in space weather applications and research studies because it provides a single number that characterizes global magnetic activity levels. Kp values of 5 or higher indicate geomagnetic storm conditions, with values above 7 representing severe storms that can have significant technological and potentially biological impacts.

2.6. References

1. Deutschlander ME, Beason RC. Avian navigation and geographic positioning. J Field Ornithol. 2014;85(2):111–33.

2. Holland RA. True navigation in birds: from quantum physics to global migration. J Zool. 2014;293:1–15.

3. Mouritsen H. Long-distance navigation and magnetoreception in migratory animals. Nature. 2018;558:50–9.

4. Chernetsov N. Compass systems. J Comp Physiol A. 2017;203:447–53.

5. Gagliardo A. Forty years of olfactory navigation in birds. J Exp Biol. 2013;216:2165–71.

6. Bonadonna F, Gagliardo A. Not only pigeons: avian olfactory navigation studied by satellite telemetry. Ethol Ecol Evol. 2021.

7. Wiltschko R, Wiltschko W. Avian navigation: a combination of innate and learned mechanisms. Adv Study Behav. 2015;47:229–310.

8. Lohmann KJ, Lohmann CMF, Putman NF. Magnetic maps in animals: nature's GPS. J Exp Biol. 2007;210:3697–705.

9. Naisbett-Jones LC, Putman NF, Stephenson JF, Ladak S, Young KA. A magnetic map leads juvenile European eels to the Gulf Stream. Curr Biol. 2017;27:1236–40.

10. Brothers JR, Lohmann KJ. Evidence that magnetic navigation and geomagnetic imprinting shape spatial genetic variation in sea turtles. Curr Biol. 2018;28:1325–9.

11. Burda H, Begall S, Hart V, Malkemper EP, Painter MS, Phillips JB. Magnetoreception in mammals. In: Fritzsch B, editor. The senses: a comprehensive reference (second edition): Elsevier; 2020. p. 421–44.

12. Genzel D, Yovel Y, Yartsev MM. Neuroethology of bat navigation. Curr Biol. 2018;28(17):R997–R1004.

13. Granger J, Walkowicz L, Fitak R, Johnsen S. Gray whales strand more often on days with increased levels of atmospheric radio-frequency noise. Curr Biol. 2020;30(4):R155–6.

14. Vanselow H, Jacobsen S, Hall C, Garthe S. Solar storms may trigger sperm whale strandings: explanation approaches for multiple strandings in the North Sea in 2016. Int J Astrobiol. 2017;17(4):336–44.

15. Kishkinev D, Chernetsov N, Pakhomov A, Heyers D, Mouritsen H. Eurasian reed warblers compensate for virtual magnetic displacement. Curr Biol. 2015;25(19):822–4.

16. Kishkinev D, Packmor F, Zeichmeister T, Winkler H-C, Chernetsov N, Mourisen H, et al. Navigation by extrapolation of geomagnetic cues in a migratory songbird. Curr Biol. 2021;31(7):1563–9.

17. Pakhomov A, Anashina A, Heyers D, Kobylkov D, Mourtisen D, Chernetsov N. Magnetic map navigation in a migratory songbird requires trigeminal input. Nat Sci Rep. 2018;8:11975.

18. Wikelski M, Arriero E, Gagliardo A, Holland RA, Huttunen MJ, Juvaste R, et al. True navigation in migrating gulls requires intact olfactory nerves. Nat Sci Rep. 2015;5:17061.

19. Bowlin MS, Bisson I-A, Shamoun-Baranes J, Reichard JD, Sapir N, Marra PP, et al. Grand challenges in migration biology. Integr Comp Biol. 2010;50(3

Calculation of Magnetic Components

Swarm data provide information on the earth's magnetic field at the orbit level, which is above the ionosphere, where geomagnetic field is affected by the electrical currents induced by the interaction of the solar wind and magnetosphere (see the figure above to understand the vertical contributions of the earth' magnetic field). This means that to obtain the values of the magnetic field on the Earth's surface where animals are migrating, the raw measurements from Swarm need to be corrected removing the effects from the modeled values from core, crust and magnetosphere.

We do this in three steps:

1. We download the swarm residuals for the GPS date, getting essentially the unmodelled ionospheric field contribution that we haven't captured in any of the other models.

2. Because at the ground level the core and crust contributions are strongen than the ones at the satellite altitude, We use the CHAOS model to compute the core, crust and magnetosphere contributions for the particular, latitude, longitude, datetime and altitude of each GPS point.

3. Finally we add those values modeled by CHAOS values with the Swarm residuals, computing a comprehensive magnetic values than include the stronger core and crust contributions and the unmodellled values from the satellite altitude that capture all the inoshpere field contributions. For details of this correction see Supplementary Information 1 in our main paper.

> **!** Important
>
> **CHAOS**: It is a comprehensive field model, containing the modeled contributions of the time-varying core, the static crustal field, the average time-varying magnetosphere. We show calculation of the residuals (adding this parameter `residuals=True`) which means: `data - CHAOS(Core+Static+Magnetosphere)`

The **CHAOS** model is available on the server as model variables:

- CHAOS-Core: SH degree 1-20
- CHAOS-Static: SH degree 21-110
- CHAOS-MMA-Primary: Magnetosphere external field, SH degree 1-2
- CHAOS-MMA-Secondary: Magnetosphere internal induced field, SH degree 1-2

How MagGeo Works

MagGeo is a tool that contains a set of python functions to carry out a data manipulation and mathematical processes to transform the data from Swarm Constellation,include the modelled values from CHAOS model (last version in December 2019) and annotate the interpolated values into every GPS point in a given GPS trajectory.

The following process is a set of three steps, **1. Get the Swarm data**, particularly the magnetic residuals in the NEC components. **2. Run the Spatio-temporal kernel** where the script will filter the Swarm data based on our spatial-temporal cylinder, to interpolate the magnetic components in the NEC frame for the given GPS track. Finally using the enriched GPS track, the final step will be the **3. Calculation of Magnetic Components** which use the CHAOS model to compute the magnetic components at the ground level. With the interpolated magnetic values corrected by the ionospheric contribution. The script will compute the other magnetic components including F (magnetic intensity), H (Horizontal component), Declination and Inclination. For detailed information of this process go to the Main Notebook.

Interpolation (ST-IDW) and Annotation Process

Once we have requested the data we need for each point in the GPS Track and considering we have gathered the available data from the three satellites for one day (*24 hours, every 60 seconds around 1440 Swarm measures per satellite*). Now we need to `filter` in space and time the

available points to compute the magnetic values for each GPS point in its particular date and time. Therefore, before running the interpolation process we set **four** functions that will require the *latitude* and *longitude* and the *epoch* time of each GPS point to filter the Swarm points into what we called **Space-Time Cylinder**. The following picture can provide a better explanation about how the points inside the space-time cylinder are included or excluded. **Figure I**, shows how the R of the Space-Time cylinder is based on the GPS point latitude. **Figure II**, illustrate the geometric components behind the space-time cylinder where the points are filtered and included in the interpolation process. Figure B, will help you to understand how the following functions compute the required parameters **Figure III** shows the idea behind the interpolation process for each GPS point, requesting and processing the available Swarm measures by the three satellites. For more information about the time-space windows please read the full paper where we explain the details of it.

SwarmMagAnnotation function:

Working as the main function for the annotation process, this functions meets the previous functions running the space-time window filters, and the computing the ST- IDW process. This function will run a interpolation process for each GPS Point considering only the Swarm points inside the Space-Time cylinder computed by the four previous functions. The return value will be an array with the values the annotated magnetic values for the GPS point. The function will be executed inside a loop going through the GPS track. This function is the annotation process per se, and through this process at first we run the interpolation getting the magnetic values in NEC reference frame and then we compute the extra magnetic values that are useful to get a better understating of the earth's magnetic field at this particular location, date and time.

Therefore, before running the interpolation process we set **four** functions that will require the *latitude* and *longitude* and the *epoch* time of each GPS point to filter the Swarm points into what we called **Space-Time Cylinder**. The following picture can provide a better explanation about

how the points inside the space-time cylinder are included or excluded. **Figure I**, shows how the R of the Space-Time cylinder is based on the GPS point latitude. **Figure II**, illustrate the geometric components behind the space-time cylinder where the points are filtered and included in the interpolation process. Figure B, will help you to understand how the following functions compute the required parameters **Figure III** shows the idea behind the interpolation process for each GPS point, requesting and processing the available Swarm measures by the three satellites. For more information about the time-space windows please read the full paper where we explain the details of it.

---

i Note

Auxiliary Functions:The function ST_IDW_Process includes 4 auxiliary functions to run the spatial-temporal kernel
distance_to_GPS function: Is the function in charge to calculate the distance between each GPS Point and the Swarm Point.
Kradius function: Is the function in charge to compute the R (radius) value in the cylinder. The R value will be considered based on the latitude of each GPS Point.
DistJ function: This function will calculate the d value as the hypotenuse created in the triangle created amount the locations of the GPS point, the location of the Swarm points and the radius value.
DfTime_func function: This is a time function to selected the points in the range of a the DeltaTime - DT window. The Delta time window has been set as 4 hours for each satellite trajectory.

Part III.

Notebooks

MagGeo - Sequential Mode

Authors | Fernando Benitez-Paez, Urška Demšar, Jed Long, Ciaran Beggan

Contact | Fernando.Benitez@st-andrews.ac.uk, ud2@st-andrews.ac.uk, jed.long@uwo.ca, ciar@bgs.ac.uk

2.7. Overview

This Jupyter Notebook will guide you through the required steps to annotate your GPS tracking data with the earth's magnetic field data from Swarm (European Space Agency). This version is called Sequential Mode, alternatively you can use Parallel Mode to take advantage of parallelized computing if required. More information about the Swarm satellites can be found in the Main Document on the MagGeo github repository. This script will use a sequential loop to run the annotation process for each GPS Point (row) from your data.

To execute the code, you can go through each cell (pressing Crtl+Enter), you will also find inner comments **##** to describe each particular step. If you are not familiar with using Jupyter Notebooks, you might want to take some time to learn how first, for example take a look at the notebook-basics.ipynb Notebook inside MagGeo.

2.8. Data requirements

Your trajectory must be in a csv format:

There are three columns that must be included in your GPS trajectory. Make sure your GPS trajectory includes **Latitude** , **Longitude** and **timestamp**. We suggest that the Timestamp column follow the day/month/year Hour:Minute (**dd/mm/yyyy HH:MM:SS**) format, Latitude and Longitude should be in decimal degrees (WGS84). Optionally an altitude column can be used providing altitude (the altitude must be in **km**). Other Columns will be ignored. Here it is an example of how your GPS track should look:

For this example we are reading the BirdGPSTrajectory.csv file. If you want to run the method using your own csv file, make sure you store your the file in the ./data folder. For more information about the dataset we used in this example go to the Main Notebook.

2.9. Import the required python libraries

```python
import datetime as dt
from datetime import timedelta
import sys, os
import pandas as pd
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
from viresclient import set_token
sys.path.append("..")

import utilities
from utilities.MagGeoFunctions import getGPSData
```

```
from utilities.MagGeoFunctions import Get_Swarm_residuals
from utilities.MagGeoFunctions import ST_IDW_Process
from utilities.MagGeoFunctions import CHAOS_ground_values
```

2.10. Add your VirES web client Token

The **VirES client API**, requires a token. Before start you need to get your own VirES token. You can visit https://vires.services/ to get yours, and then add it into the next cell.

```
set_token("https://vires.services/ows", set_default=True)
```

2.11. Read the GPS track

The following steps will load the GPS track from a csv file, and set some requirements before downloading geomagnetic data from Swarm. If your csv track file doesnt not have any altitude attribute, MagGeo will use sea level as your altitude (i.e., 0 Km). **Altitude column units must be Km**

```
base_dir=os.path.dirname(os.getcwd())
temp_results_dir = os.path.join(base_dir, "temp_data")
results_dir = os.path.join(base_dir, "results")
data_dir = os.path.join(base_dir, "data")
utilities_dir = os.path.join(base_dir, "utilities")
```

```
# Make sure the csv file of your trackectory is stored in the Data folder.
# Enter the name of your GPS track csv file including the extension .csv  and press Enter (e
# Make sure you have a columnn that integrates date and time, before include in MagGeo.
```

67

```
# If your csv track file does not have any altitude attribute, MagGeo will us
# i.e height (Only in KM)
gpsfilename= "BirdGPSTrajectoryTest.csv"
Lat="location-lat"
Long="location-long"
DateTime="timestamp"
altitude = "height"
```

```
# Here MagGeo is reading your CSV file, taking the Lat, Long, Date&Time and A
# Setting the date and time attributes for the required format and computing
GPSData = getGPSData(data_dir,gpsfilename,Lat,Long,DateTime,altitude)
GPSData
```

2.12. Validate the correct amount of Swarm measures

The following loop is identifiying the time and validating if the time is less than 4:00 hours and more than 20:00 hours to bring one extra day of data. The result of this validation is written in an empty python list which will be later validated to get the unique dates. This avoids duplicate downloading of data for the same day and reduces overall computational time.

```
datestimeslist = []
for index, row in GPSData.iterrows():
    datetimerow  = row['gpsDateTime']
    daterow = row['dates']
    hourrow = row['times']
    hourrow = hourrow.strftime('%H:%M:%S')
    if hourrow < '04:00:00':
        date_bfr = daterow - (timedelta(days=1))
```

```
        datestimeslist.append(daterow)
        datestimeslist.append(date_bfr)
    if hourrow > '20:00:00':
        Date_aft = daterow + (timedelta(days=1))
        datestimeslist.append(daterow)
        datestimeslist.append(Date_aft)
    else:
        datestimeslist.append(daterow)
```

Getting a list of unique dates to download the Swarm Data

```
def uniquelistdates(list):
    x = np.array(list)
    uniquelist = np.unique(x)
    return uniquelist

uniquelist_dates = uniquelistdates(datestimeslist)
uniquelist_dates
```

2.13. Download Swarm residuals data

Once the date and time columns have been defined and the unique dates
are identified the script can start the download process. Usually the data
from Swarm is requested using only one satellite, however **MagGeo** will
use the magnetic measures from the three satellite of the Swarm Mission
(Alpha, Bravo, Charlie). Be aware satellite Charlie, got its AMS bro-
ken earlier in the mission, although the initial dates still have valid data
MagGeo can use.

Be aware: Due to the amount of dates in the demo GPS track (42 days),
the time to process the sample data will take approximately 10 minutes.

Unfortunatly the download process migth be a slow process, particulaly for the magnetic models data MagGeo requieres.

Set a connection to the VirES client and using the function Get_Swarm_residuals we will get the swarm residuals for the dates included in the previous list.

```
%%time

hours_t_day = 24 #MagGeo needs the entire Swarm data for each day of the ider
hours_added = dt.timedelta(hours = hours_t_day)

listdfa = []
listdfb = []
listdfc = []

for d in tqdm(uniquelist_dates, desc="Getting Swarm Data"):
    #print("Getting Swarm data for date:",d )
    startdate = dt.datetime.combine(d, dt.datetime.min.time())
    enddate = startdate + hours_added
    SwarmResidualsA,SwarmResidualsB,SwarmResidualsC = Get_Swarm_residuals(sta
    listdfa.append(SwarmResidualsA)
    listdfb.append(SwarmResidualsB)
    listdfc.append(SwarmResidualsC)
```

Concat the previous results and temporally save the requested data locally: Integrate the previous list for all dates, into pandas dataframes. We will temporally saved the previous results, in case you need to re-run MagGeo, with the following csv files you will not need to run the download process.

```
%%time

PdSwarmRes_A = pd.concat(listdfa, join='outer', axis=0)
```

```
PdSwarmRes_A.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_A.csv'), header=True)
PdSwarmRes_B = pd.concat(listdfb, join='outer', axis=0)
PdSwarmRes_B.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_B.csv'), header=True)
PdSwarmRes_C = pd.concat(listdfc, join='outer', axis=0)
PdSwarmRes_C.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_C.csv'), header=True)

TotalSwarmRes_A = pd.read_csv(os.path.join(temp_results_dir,"TotalSwarmRes_A.csv"),low_memor
TotalSwarmRes_A['timestamp'] = pd.to_datetime(TotalSwarmRes_A['timestamp'])
TotalSwarmRes_B = pd.read_csv(os.path.join(temp_results_dir,"TotalSwarmRes_B.csv"),low_memor
TotalSwarmRes_B['timestamp'] = pd.to_datetime(TotalSwarmRes_B['timestamp'])
TotalSwarmRes_C = pd.read_csv(os.path.join(temp_results_dir,"TotalSwarmRes_C.csv"),low_memor
TotalSwarmRes_C['timestamp'] = pd.to_datetime(TotalSwarmRes_C['timestamp'])

TotalSwarmRes_A.head(10) #If you need to take a look of the Swarm Data, you can print TotalS
```

2.14. Spatio-Temporal filter and interpolation process (ST-IDW)

Once we have requested the swarm data, now we need to `filter` in space and time the available points to compute the magnetic values (NEC frame) for each GPS point based on its particular date and time. The function ST_IDW_Process takes the GPS track and the downloaded data from swarm to filter in space and time based on the criteria defined in our method. With the swarm data filtered we interpolate (IDW) the NEC components for each GPS data point.

```
%%time
#Sequential mode, applying a traditional loop using iterrows.
if __name__ == '__main__':
    dn = [] ## List used to add all the GPS points with the annotated MAG Data. See the last
    for index, row in tqdm(GPSData.iterrows(), total=GPSData.shape[0], desc="Annotating the
```

71

```
        GPSLat = row['gpsLat']
        GPSLong = row['gpsLong']
        GPSDateTime = row['gpsDateTime']
        GPSTime = row['epoch']
        GPSAltitude = row['gpsAltitude']
        #print("Process for:", index,"DateTime:",GPSDateTime)
        try:
            result=ST_IDW_Process(GPSLat,GPSLong,GPSAltitude, GPSDateTime,GPS
            dn.append(result)
        except:
            #print("Ups!.That was a bad Swarm Point, let's keep working with
            result_badPoint= {'Latitude': GPSLat, 'Longitude': GPSLong, 'Alti
            dn.append(result_badPoint)
            continue
```

Temporally save the ST-IDW result locally. Still MagGeo needs to run the calculation of geomagnetic components, brining the magnetic values at the altitude provided for your GPS track.

```
GPS_ResInt = pd.DataFrame(dn)
GPS_ResInt.to_csv (os.path.join(temp_results_dir,"GPS_ResInt.csv"), header=Tr
GPS_ResInt
```

2.15. Compute the magnetic components at the trajectory altitute using CHAOS model

The function CHAOS_ground_values is used to run the calculation of magnetic components. This adjustment requeries the magnetic components at the trajectory altitude (or at the ground level) using CHAOS (theta, phi, radial). This process also further conducts the rotation and transformation between a geocentric earth-based reference system

72

(CHAOS) and geodetic earth-based reference system (GPS track). Once the corrected values are calculated the non-necesary columns are removed. For more information about this process go to the Main Notebook.

```
%%time
X_obs, Y_obs, Z_obs, X_obs_internal, Y_obs_internal, Z_obs_internal =CHAOS_ground_values(uti
GPS_ResInt['N'] =pd.Series(X_obs)
GPS_ResInt['E'] =pd.Series(Y_obs)
GPS_ResInt['C'] =pd.Series(Z_obs)
GPS_ResInt['N_Obs'] =pd.Series(X_obs_internal)
GPS_ResInt['E_Obs'] =pd.Series(Y_obs_internal)
GPS_ResInt['C_Obs'] =pd.Series(Z_obs_internal)


GPS_ResInt.drop(columns=['N_res', 'E_res','C_res'], inplace=True)
GPS_ResInt
```

2.16. The final result

With the NEC components for each GPS Track point, it is possible to compute the aditional magnetic components. For more information about the magnetic components and their relevance go to the main paper or notebook.

```
<strong> The annotated dataframe will include the following attributes:</strong> If you nee
<ul>
  <li><strong>Latitude</strong> from the GPS Track.</li>
  <li><strong>Longitude</strong> from the GPS Track.</li>
  <li><strong>Timestamp</strong> from the GPS Track.</li>
  <li><strong>Magnetic Field Intensity</strong>  mapped as Fgps in nanoTeslas (nT).</li>
  <li><strong>N (Northwards) component</strong> mapped as N in nanoTeslas (nT).</li>
  <li><strong>E (Eastwards) component</strong> mapped as E. in nanoteslas (nT).</li>
  <li><strong>C (Downwards or Center)</strong> component mapped as C in nanoTeslas (nT).</li
```

73

```
  <li><strong>Horizontal component</strong> mapped as H in nanoTeslas (nT).<
  <li><strong>Magnetic Inclination </strong> mapped as I in degrees.</li>
  <li><strong>Magnetic Declination or dip angle</strong> mapped as D in degr
  <li><strong>Kp Index</strong> mapped as kp</li>
  <li><strong>Total Points</strong> as the amount of Swarm points included i
  <li><strong>Minimum Distance</strong> mapped as MinDist, representing the
  <li><strong>Average Distance</strong> mapped as AvDist, representing the a
</ul>
```

```
%%time
# Having Intepolated and weighted the magnetic values, we can compute the ot
GPS_ResInt['H'] = np.sqrt((GPS_ResInt['N']**2)+(GPS_ResInt['E']**2))
#check the arcgtan in python., From arctan2 is saver.
DgpsRad = np.arctan2(GPS_ResInt['E'],GPS_ResInt['N'])
GPS_ResInt['D'] = np.degrees(DgpsRad)
IgpsRad = np.arctan2(GPS_ResInt['C'],GPS_ResInt['H'])
GPS_ResInt['I'] = np.degrees(IgpsRad)
GPS_ResInt['F'] = np.sqrt((GPS_ResInt['N']**2)+(GPS_ResInt['E']**2)+(GPS_Res
GPS_ResInt
```

The previous dataframe (GPS_ResInt), MagGeo has computed the geo-magnetic components for each locations and time of your CSV trajectory. Now we will finish up combining the original atributes from your CSV with the annotated results from MagGeo.

```
%%time
originalGPSTrack=pd.read_csv(os.path.join(data_dir,gpsfilename))
MagGeoResult = pd.concat([originalGPSTrack, GPS_ResInt], axis=1)
#Drop duplicated columns. Latitude, Longitued, and DateTime will not be part
MagGeoResult.drop(columns=['Latitude', 'Longitude', 'DateTime'], inplace=Tru
MagGeoResult
```

74

2.17. Export the final results to a CSV file

```
%%time
#Exporting the CSV file
outputfile ="GeoMagResult_"+gpsfilename
export_csv = MagGeoResult.to_csv (os.path.join(results_dir,outputfile), index = None, header
```

2.18. Validate the results (Optional)

To validate the results we plot the Fgpscolumn.

```
## Creating a copy of the results and setting the Datetime Column as dataframe index.
ValidateDF = GPS_ResInt.copy()
ValidateDF.set_index("DateTime", inplace=True)
## Plotting the F column.
hist = ValidateDF.hist(column='F')
plt.title('F distribution')
plt.xlabel('F in nT')
plt.ylabel('# of measurements')
```

2.19. Map the GPS Track using the annotated Magnetic Values (Optional)

Now we are going to plot the annotated GPS track stored into the Mag-DataFinal dataframe to see the different magnetic components in a map to have a better prespective of the impact of the earth magnetic field.

75

```python
ValidateDF.plot(kind="scatter", x="Latitude", y="Longitude",
    label="Magnetic Intensity in nT",
    c="F", cmap=plt.get_cmap("gist_rainbow"),
    colorbar=True, alpha=0.4, figsize=(10,7),
    sharex=False #This is only needed to get the x-axis label working due to

)

plt.ylabel("Longitude", fontsize=12)
plt.xlabel("Latitude", fontsize=12)
plt.legend(fontsize=12)
plt.show()
```

```python
import geopandas
import geoplot
import hvplot.pandas
gdf = geopandas.GeoDataFrame(ValidateDF, geometry=geopandas.points_from_xy(Va
gdf.head()
```

```python
gdf.hvplot(title=f'Annotated trajectory using MagGeo - F GeoMag Intensity',
        geo=True,
        c='F',
        tiles='CartoLight',
        frame_width=700,
        frame_height=500)
```

```python
gdf.hvplot(title=f'Annotated trajectory using MagGeo - I Inclination',
        geo=True,
        tiles='CartoLight',
        c='I',
        cmap='Viridis',
        frame_width=700,
        frame_height=500)
```

76

```python
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres'))

ax = world.plot(color='white', edgecolor='gray', figsize = (18,8))

minx, miny, maxx, maxy = gdf.total_bounds
ax.set_xlim(minx, maxx)
ax.set_ylim(miny, maxy)

gdf.plot(ax=ax, column='F', legend=True,
         legend_kwds={'label': "Magnetic Intensity in nT",
                      'orientation': "horizontal"})
plt.ylabel("Longitude", fontsize=12)
plt.xlabel("Latitude", fontsize=12)

plt.show()
```

```python
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize = (18,8))

ax1 = world.plot(ax=ax1, color='white', edgecolor='black')
xlim = ([gdf.total_bounds[0],  gdf.total_bounds[2]])
ylim = ([gdf.total_bounds[1],  gdf.total_bounds[3]])
ax1.set_xlim(xlim)
ax1.set_ylim(ylim)


gdf.plot(ax=ax1, column='F', legend=True,
         legend_kwds={'label': "Magnetic Intensity in nT",
                      'orientation': "horizontal"})
plt.ylabel("Longitude", fontsize=9)
plt.xlabel("Latitude", fontsize=9)
ax1.set_title('Magnetic Intensity - F')
```

```python
ax1.set_xlabel('Latitude')
ax1.set_ylabel('Longitude')


ax2 = world.plot( ax=ax2, color='white', edgecolor='black')
xlim = ([gdf.total_bounds[0],  gdf.total_bounds[2]])
ylim = ([gdf.total_bounds[1],  gdf.total_bounds[3]])
ax2.set_xlim(xlim)
ax2.set_ylim(ylim)


# We can now plot our ``GeoDataFrame``.
gdf.plot(ax=ax2, column='I', legend=True, cmap='Spectral',
        legend_kwds={'label': " Inclination in Degrees",
                     'orientation': "horizontal"})
ax2.set_title('Inclination - I')
ax2.set_xlabel('Latitude')
ax2.set_ylabel('Longitude')
```

MagGeo - Parallel Mode

Authors | Fernando Benitez-Paez, Urška Demšar, Jed Long, Ciaran Beggan

Contact | Fernando.Benitez@st-andrews.ac.uk, ud2@st-andrews.ac.uk, jed.long@uwo.ca, ciar@bgs.ac.uk

2.20. Overview

This Jupyter Notebook will guide you through the required steps to annotate your GPS tracking data with the earth's magnetic field data from Swarm (European Space Agency). This version is called Parallel Mode to take advantage of parallelized computing to process big datasets.

To execute the code, you can go through each cell (pressing Crtl+Enter), you will also find inner comments `##` to describe each particular step. If you are not familiar with Jupyter Notebook, you migth want to take some time to learn how to use it first, for example take a look at the notebook-basics.ipynb Notebook inside MagGeo.

For parallel processing, there are some considerations to make:

1. Linux and Windows environments have some differences. In windows we need to separate the functions and store them separately, then import them into a `main` function.

2. Defining what part of the process is CPU bound and what part is I/O bound: Identify what parts of the program are I/O bound (writing or reading from the disk or network) and what part par CPU bound (Processing capacity). To take advantage of our CPU capacity we need to identify the process where the CPU is actually doing the main Tasks.

2.21. Data requirements

Your trajectory must be in a csv format:

There are three columns that must be included in your GPS trajectory. Make sure your GPS trajectory includes **Latitude** , **Longitude** and **timestamp**. We suggest that the Timestamp column follow the day/month/year Hour:Minute (**dd/mm/yyyy HH:MM:SS**) format, Latitude and Longitude should be in decimal degrees (WGS84). Optionally an altitude column can be used providing altitude (the altitude must be in **km**). Other Columns will be ignored. Here it is an example of how your GPS track should look:

For this example we are reading the BirdGPSTrajectory.csv file. If you want to run the method using your own csv file, make sure you store your the file in the ./data folder. For more information about the dataset we used in this example go to the Main Notebook.

2.22. Import the requeried libraries

```python
import datetime as dt
from datetime import timedelta
import sys, os
import pandas as pd
```

```
import numpy as np
from tqdm import tqdm
from pathlib import Path
import matplotlib.pyplot as plt

from viresclient import set_token
sys.path.append("..")
import utilities
from utilities.MagGeoFunctions import getGPSData
from utilities.MagGeoFunctions import Get_Swarm_residuals
```

2.23. Add your VirES web client Token

The **VirES client API**, requires a token. Before start you need to get your own VirES token. You can visit https://vires.services/ to get yours, and then add it into the next cell.

```
set_token("https://vires.services/ows", set_default=True)
```

2.24. Reading the GPS track

The following steps will load the GPS track from a csv file, and set some requirements before download the data from Swarm. Importing the GPS track. You can note that there is a folder to store the CSV file. Using `os.getcwd()` you can validate where the file is located.

```
base_dir=os.path.dirname(os.getcwd())
temp_results_dir = os.path.join(base_dir, "temp_data")
results_dir = os.path.join(base_dir, "results")
data_dir = os.path.join(base_dir, "data")
```

```
#Make sure the csv file of your trajectory is stored in the Data folder.
#Enter the name of your GPS track csv file including the extension .csv  and
# Make sure you have a column that integrates date and time, before include
#If your csv track file does not have any altitude attribute, MagGeo will us
# i.e height (Only in KM)

gpsfilename= "BirdGPSTrajectoryTest.csv"
Lat="location-lat"
Long="location-long"
DateTime="timestamp"
altitude = "height"
```

```
# Here MagGeo is reading your CSV file, taking the Lat, Long, Date&Time and
# Setting the date and time attributes for the required format and computing
GPSData = getGPSData(data_dir,gpsfilename,Lat,Long,DateTime,altitude)
GPSData
```

	gpsDateTime	gpsLong	gpsLat	gpsAltitude	epoch	dates
0	2014-09-08 05:54:00	68.307333	70.854717	0.000	1410155640	2014-09-08
1	2014-09-08 06:10:00	67.975050	70.830300	0.406	1410156600	2014-09-08
2	2014-09-08 06:26:00	67.752417	70.761717	0.498	1410157560	2014-09-08
3	2014-09-08 06:42:00	67.561983	70.686517	0.787	1410158520	2014-09-08
4	2014-09-08 07:14:00	67.548317	70.685450	0.337	1410160440	2014-09-08
...	...	...	...	...	...	...
194	2014-09-27 11:09:00	49.503800	67.735100	0.098	1411816140	2014-09-27
195	2014-09-27 11:25:00	49.503767	67.735100	0.099	1411817100	2014-09-27
196	2014-09-27 11:40:00	49.503667	67.735100	0.100	1411818000	2014-09-27
197	2014-09-27 11:56:00	49.503650	67.735100	0.100	1411818960	2014-09-27
198	2014-09-27 12:11:00	49.503617	67.735000	0.092	1411819860	2014-09-27

Setting the date and time attributes for the requerided format and computing the epoch column. Values like Maximum and Minimun Date and

time are also calculated.

2.25. Validate the right amount of Swarm measures

The following loop is identifiying the time and validating if the time is less than 4:00 hours and more than 20:00 hours to bring one extra day of data. The result of this validation is written in a empty python list which will be later validated to get the unique dates avoing to download data for the same day and reducing the the downloand time process.

```
%%time
datestimeslist = []
for index, row in GPSData.iterrows():
    datetimerow  = row['gpsDateTime']
    daterow = row['dates']
    hourrow = row['times']
    hourrow = hourrow.strftime('%H:%M:%S')
    if hourrow < '04:00:00':
        date_bfr = daterow - (timedelta(days=1))
        datestimeslist.append(daterow)
        datestimeslist.append(date_bfr)
    if hourrow > '20:00:00':
        Date_aft = daterow + (timedelta(days=1))
        datestimeslist.append(daterow)
        datestimeslist.append(Date_aft)
    else:
        datestimeslist.append(daterow)
```

Getting a list of unique dates, to being used to download the Swarm Data

```
%%time
def uniquelistdates(list):
    x = np.array(list)
    uniquelist = np.unique(x)
    return uniquelist

uniquelist_dates = uniquelistdates(datestimeslist)
uniquelist_dates
```

2.26. Download Swarm residuals data

Once the date and time columns have been defined, and the unique dates were identified the script can start the download process. Usually the data from Swarm is requested using only one satellite, however **MagGeo** will use the magnetic measures from the three satellite of the Swarm Mission.

Be aware: Due to the amount of dates the GPS track has (42 days) to request and compute the residuals, the time to process the sample data will take approximately 10 minutes.

Set a connection to the VirES client and using the function Get_Swarm_residuals we will get the swarm residuals for the dates included in the previous list.

```
%%time

hours_t_day = 24
hours_added = dt.timedelta(hours = hours_t_day)

listdfa = []
listdfb = []
listdfc = []
```

```
for d in tqdm(uniquelist_dates, desc="Getting Swarm Data"):
    #print("Getting Swarm data for date:",d )
    startdate = dt.datetime.combine(d, dt.datetime.min.time())
    enddate = startdate + hours_added
    SwarmResidualsA,SwarmResidualsB,SwarmResidualsC = Get_Swarm_residuals(startdate, enddate
    listdfa.append(SwarmResidualsA)
    listdfb.append(SwarmResidualsB)
    listdfc.append(SwarmResidualsC)
```

Concat the previous results and temporally save the requested data locally: Integrate the previous list for all dates, into pandas dataframes. We will temporally saved the previous results, in case you need to re-run MagGeo, with the following csv files you will not need to run the download process.

```
%%time
TotalSwarmRes_A = pd.concat(listdfa, join='outer', axis=0)
TotalSwarmRes_A.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_A.csv'), header=True)
TotalSwarmRes_B = pd.concat(listdfb, join='outer', axis=0)
TotalSwarmRes_B.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_B.csv'), header=True)
TotalSwarmRes_C = pd.concat(listdfc, join='outer', axis=0)
TotalSwarmRes_C.to_csv (os.path.join(temp_results_dir,'TotalSwarmRes_C.csv'), header=True)
TotalSwarmRes_A #If you need to take a look of the Swarm Data, you can print TotalSwarmRes_E
```

2.27. Set the number of processes, and split the dataframe (GPSData) into chunks

We can set the number or processess we need to dedicate for the multiprocessing mode, of course that also depends on the number of cores the machine you are using to run **MagGeo**. You can use

85

`multiprocessing.cpu_count()` to set the number of processes as the the number of cores your machine has. Beside that we will also to split the GPS track into chucks to dedicate each core for each chuck. For more information take a look at the Home Notebook.

```python
import multiprocessing
import sklearn
from multiprocessing import Pool

NumCores = multiprocessing.cpu_count()
df_chunks = np.array_split(GPSData,NumCores)
df_chunks
```

```
[             gpsDateTime      gpsLong      gpsLat  gpsAltitude        epoch  \
 0   2014-09-08 05:54:00    68.307333   70.854717        0.000   1410155640
 1   2014-09-08 06:10:00    67.975050   70.830300        0.406   1410156600
 2   2014-09-08 06:26:00    67.752417   70.761717        0.498   1410157560
 3   2014-09-08 06:42:00    67.561983   70.686517        0.787   1410158520
 4   2014-09-08 07:14:00    67.548317   70.685450        0.337   1410160440
 5   2014-09-08 07:30:00    67.549433   70.685750        0.026   1410161400
 6   2014-09-08 07:46:00    67.530983   70.690333        0.026   1410162360
 7   2014-09-08 08:03:00    67.506233   70.692683        0.023   1410163380
 8   2014-09-08 08:34:00    67.506167   70.692533        0.022   1410165240
 9   2014-09-08 08:50:00    67.506383   70.692583        0.023   1410166200
 10  2014-09-08 09:37:00    67.501633   70.695017        0.026   1410169020
 11  2014-09-08 09:54:00    67.498917   70.693850        0.026   1410170040
 12  2014-09-08 13:53:00    67.505800   70.692667        0.000   1410184380

          dates      times
 0   2014-09-08   05:54:00
 1   2014-09-08   06:10:00
 2   2014-09-08   06:26:00
 3   2014-09-08   06:42:00
```

```
4    2014-09-08   07:14:00
5    2014-09-08   07:30:00
6    2014-09-08   07:46:00
7    2014-09-08   08:03:00
8    2014-09-08   08:34:00
9    2014-09-08   08:50:00
10   2014-09-08   09:37:00
11   2014-09-08   09:54:00
12   2014-09-08   13:53:00   ,
          gpsDateTime     gpsLong      gpsLat  gpsAltitude        epoch  \
13 2014-09-08 14:09:00   67.505883   70.692667        0.034   1410185340
14 2014-09-08 14:25:00   67.506000   70.692617        0.034   1410186300
15 2014-09-08 14:41:00   67.505883   70.692750        0.033   1410187260
16 2014-09-08 15:13:00   67.511617   70.694350        0.036   1410189180
17 2014-09-08 15:29:00   67.509050   70.693650        0.034   1410190140
18 2014-09-08 15:45:00   67.511250   70.693467        0.032   1410191100
19 2014-09-08 16:01:00   67.510000   70.693450        0.033   1410192060
20 2014-09-08 16:33:00   67.510767   70.693633        0.034   1410193980
21 2014-09-08 16:49:00   67.509500   70.693583        0.033   1410194940
22 2014-09-08 17:05:00   67.509750   70.693600        0.033   1410195900
23 2014-09-08 17:38:00   67.509700   70.693633        0.035   1410197880
24 2014-09-08 17:53:00   67.509650   70.693650        0.017   1410198780
25 2014-09-21 12:23:00   49.996717   66.897217        0.019   1411302180


        dates       times
13   2014-09-08   14:09:00
14   2014-09-08   14:25:00
15   2014-09-08   14:41:00
16   2014-09-08   15:13:00
17   2014-09-08   15:29:00
18   2014-09-08   15:45:00
19   2014-09-08   16:01:00
20   2014-09-08   16:33:00
21   2014-09-08   16:49:00
```

```
22   2014-09-08   17:05:00
23   2014-09-08   17:38:00
24   2014-09-08   17:53:00
25   2014-09-21   12:23:00   ,
         gpsDateTime     gpsLong     gpsLat  gpsAltitude       epoch  \
26  2014-09-21 12:55:00  49.997550  66.897567        0.030  1411304100
27  2014-09-21 13:11:00  49.998683  66.898600        0.000  1411305060
28  2014-09-21 13:43:00  49.997133  66.898300        0.000  1411306980
29  2014-09-21 14:00:00  49.997383  66.897600        0.000  1411308000
30  2014-09-21 14:17:00  49.997467  66.897517        0.000  1411309020
31  2014-09-21 14:31:00  50.005250  66.895483        0.044  1411309860
32  2014-09-21 15:03:00  50.040800  66.890583        0.000  1411311780
33  2014-09-21 15:19:00  50.047867  66.890167        0.000  1411312740
34  2014-09-21 15:35:00  50.047600  66.890150        0.000  1411313700
35  2014-09-21 16:07:00  50.047800  66.890183        0.000  1411315620
36  2014-09-21 16:23:00  49.961467  66.980783        0.000  1411316580
37  2014-09-22 05:52:00  49.847017  66.960383        0.000  1411365120
38  2014-09-22 06:08:00  49.846967  66.960450        0.000  1411366080


         dates      times
26   2014-09-21   12:55:00
27   2014-09-21   13:11:00
28   2014-09-21   13:43:00
29   2014-09-21   14:00:00
30   2014-09-21   14:17:00
31   2014-09-21   14:31:00
32   2014-09-21   15:03:00
33   2014-09-21   15:19:00
34   2014-09-21   15:35:00
35   2014-09-21   16:07:00
36   2014-09-21   16:23:00
37   2014-09-22   05:52:00
38   2014-09-22   06:08:00   ,
         gpsDateTime     gpsLong     gpsLat  gpsAltitude       epoch  \
```

```
39 2014-09-22 06:24:00   49.846950  66.960367       0.000  1411367040
40 2014-09-22 06:40:00   49.847350  66.960383       0.000  1411368000
41 2014-09-22 07:13:00   49.846850  66.960400       0.000  1411369980
42 2014-09-22 08:02:00   49.712750  67.096717       0.000  1411372920
43 2014-09-30 06:35:00   47.120217  66.684717       0.006  1412058900
44 2014-09-30 06:51:00   46.696433  66.529533       0.000  1412059860
45 2014-09-30 07:07:00   46.267167  66.378733       0.000  1412060820
46 2014-09-30 07:23:00   45.813017  66.230733       0.000  1412061780
47 2014-09-30 08:04:00   44.659750  65.770600       0.000  1412064240
48 2014-09-30 08:46:00   43.495867  65.287350       0.000  1412066760
49 2014-09-30 10:10:00   41.314500  64.129950       0.000  1412071800
50 2014-09-30 10:26:00   40.957017  63.896433       0.000  1412072760
51 2014-09-30 11:04:00   40.313267  63.317133       0.000  1412075040


         dates      times
39   2014-09-22   06:24:00
40   2014-09-22   06:40:00
41   2014-09-22   07:13:00
42   2014-09-22   08:02:00
43   2014-09-30   06:35:00
44   2014-09-30   06:51:00
45   2014-09-30   07:07:00
46   2014-09-30   07:23:00
47   2014-09-30   08:04:00
48   2014-09-30   08:46:00
49   2014-09-30   10:10:00
50   2014-09-30   10:26:00
51   2014-09-30   11:04:00   ,
          gpsDateTime     gpsLong     gpsLat  gpsAltitude       epoch  \
52 2014-09-30 11:21:00   40.012267  63.049433          0.0  1412076060
53 2014-09-30 11:53:00   39.520267  62.547750          0.0  1412077980
54 2014-09-30 12:09:00   39.286400  62.327483          0.0  1412078940
55 2014-09-30 16:27:00   35.165983  59.422467          0.0  1412094420
56 2014-09-30 16:59:00   34.759200  58.980250          0.0  1412096340
```

```
57  2014-09-30  17:15:00   34.471817   58.817917          0.0   1412097300
58  2014-09-30  17:47:00   34.448300   58.827933          0.0   1412099220
59  2014-09-30  18:04:00   34.448467   58.826017          0.0   1412100240
60  2014-09-30  18:20:00   34.435233   58.819850          0.0   1412101200
61  2014-09-30  18:35:00   34.201167   58.727250          0.0   1412102100
62  2014-09-30  19:08:00   33.569233   58.405500          0.0   1412104080
63  2014-09-30  19:24:00   33.236700   58.241183          0.0   1412105040
64  2014-09-30  19:40:00   33.014350   58.041567          0.0   1412106000


          dates     times
52   2014-09-30  11:21:00
53   2014-09-30  11:53:00
54   2014-09-30  12:09:00
55   2014-09-30  16:27:00
56   2014-09-30  16:59:00
57   2014-09-30  17:15:00
58   2014-09-30  17:47:00
59   2014-09-30  18:04:00
60   2014-09-30  18:20:00
61   2014-09-30  18:35:00
62   2014-09-30  19:08:00
63   2014-09-30  19:24:00
64   2014-09-30  19:40:00  ,
            gpsDateTime     gpsLong      gpsLat  gpsAltitude        epoch  \
65  2014-09-30  19:56:00   32.698767   57.913150        0.000   1412106960
66  2014-09-30  20:12:00   32.470350   57.709567        0.003   1412107920
67  2014-09-30  20:28:00   32.223617   57.567683        0.002   1412108880
68  2014-10-01  00:29:00   31.970950   57.454033        0.000   1412123340
69  2014-10-01  00:44:00   31.971250   57.453767        0.000   1412124240
70  2014-10-01  01:00:00   31.971667   57.453667        0.037   1412125200
71  2014-10-01  02:42:00   31.894067   57.427383        0.038   1412131320
72  2014-10-01  02:59:00   31.669200   57.321483        0.128   1412132340
73  2014-10-01  03:14:00   31.446667   57.195767        0.135   1412133240
74  2014-10-01  05:05:00   29.696600   56.703217        0.131   1412139900
```

90

```
75 2014-10-01 06:19:00  28.411983  56.252417          0.127  1412144340
76 2014-10-01 06:35:00  28.150767  56.085417          0.132  1412145300
77 2014-10-01 07:23:00  27.264867  55.628233          0.132  1412148180


        dates       times
65  2014-09-30   19:56:00
66  2014-09-30   20:12:00
67  2014-09-30   20:28:00
68  2014-10-01   00:29:00
69  2014-10-01   00:44:00
70  2014-10-01   01:00:00
71  2014-10-01   02:42:00
72  2014-10-01   02:59:00
73  2014-10-01   03:14:00
74  2014-10-01   05:05:00
75  2014-10-01   06:19:00
76  2014-10-01   06:35:00
77  2014-10-01   07:23:00   ,
          gpsDateTime     gpsLong     gpsLat  gpsAltitude       epoch  \
78 2014-10-01 07:40:00  26.967033  55.513917        0.132  1412149200
79 2014-10-01 07:56:00  26.703400  55.396383        0.131  1412150160
80 2014-10-01 11:42:00  22.913617  53.929300        0.129  1412163720
81 2014-10-01 12:13:00  22.347700  53.795100        0.131  1412165580
82 2014-10-01 12:29:00  22.068817  53.726433        0.129  1412166540
83 2014-10-01 13:02:00  21.519500  53.613633        0.000  1412168520
84 2014-10-01 13:18:00  21.238517  53.580000        0.142  1412169480
85 2014-10-01 13:34:00  20.974200  53.522383        0.137  1412170440
86 2014-10-01 13:50:00  20.682200  53.486600        0.136  1412171400
87 2014-10-01 14:22:00  20.148067  53.381400        0.000  1412173320
88 2014-10-01 14:38:00  19.858717  53.353933        0.134  1412174280
89 2014-10-01 14:54:00  19.568867  53.324533        0.149  1412175240
90 2014-10-01 15:26:00  18.964567  53.259633        0.145  1412177160


        dates       times
```

```
78   2014-10-01  07:40:00
79   2014-10-01  07:56:00
80   2014-10-01  11:42:00
81   2014-10-01  12:13:00
82   2014-10-01  12:29:00
83   2014-10-01  13:02:00
84   2014-10-01  13:18:00
85   2014-10-01  13:34:00
86   2014-10-01  13:50:00
87   2014-10-01  14:22:00
88   2014-10-01  14:38:00
89   2014-10-01  14:54:00
90   2014-10-01  15:26:00  ,
              gpsDateTime     gpsLong     gpsLat  gpsAltitude       epoch  \
91   2014-10-01 15:42:00  18.664850  53.268217        0.145  1412178120
92   2014-10-01 19:42:00  16.177583  53.797867        0.143  1412192520
93   2014-10-01 19:57:00  16.178717  53.798633        0.136  1412193420
94   2014-10-01 20:13:00  16.179067  53.798667        0.135  1412194380
95   2014-10-01 20:30:00  16.179150  53.798850        0.133  1412195400
96   2014-10-01 21:02:00  16.178950  53.798350        0.134  1412197320
97   2014-10-01 21:18:00  16.178817  53.798350        0.146  1412198280
98   2014-10-01 21:34:00  16.179200  53.798467        0.124  1412199240
99   2014-10-01 21:50:00  16.178483  53.798617        0.117  1412200200
100  2014-10-01 22:22:00  16.179033  53.798367        0.118  1412202120
101  2014-10-01 22:38:00  16.178983  53.797933        0.119  1412203080
102  2014-10-01 23:10:00  16.178733  53.798333        0.118  1412205000

          dates      times
91   2014-10-01  15:42:00
92   2014-10-01  19:42:00
93   2014-10-01  19:57:00
94   2014-10-01  20:13:00
95   2014-10-01  20:30:00
96   2014-10-01  21:02:00
```

92

```
97    2014-10-01  21:18:00
98    2014-10-01  21:34:00
99    2014-10-01  21:50:00
100   2014-10-01  22:22:00
101   2014-10-01  22:38:00
102   2014-10-01  23:10:00   ,
            gpsDateTime    gpsLong     gpsLat  gpsAltitude        epoch  \
103 2014-10-01 23:26:00  16.178650  53.798483        0.126  1412205960
104 2014-10-01 23:42:00  16.179417  53.798400        0.129  1412206920
105 2014-10-01 23:58:00  16.179233  53.798633        0.107  1412207880
106 2014-10-02 03:42:00  16.179567  53.795533        0.108  1412221320
107 2014-10-02 04:14:00  16.161283  53.889767        0.109  1412223240
108 2014-10-02 05:02:00  15.977883  54.208517        0.913  1412226120
109 2014-10-02 05:18:00  15.791683  54.210117        0.823  1412227080
110 2014-10-02 05:35:00  15.640317  54.180867        0.010  1412228100
111 2014-10-02 05:51:00  15.502350  54.150750        0.005  1412229060
112 2014-10-02 06:22:00  15.502000  54.150600        0.009  1412230920
113 2014-10-02 06:38:00  15.500733  54.145917        0.000  1412231880
114 2014-10-02 07:26:00  15.500717  54.146000        0.000  1412234760


          dates      times
103   2014-10-01  23:26:00
104   2014-10-01  23:42:00
105   2014-10-01  23:58:00
106   2014-10-02  03:42:00
107   2014-10-02  04:14:00
108   2014-10-02  05:02:00
109   2014-10-02  05:18:00
110   2014-10-02  05:35:00
111   2014-10-02  05:51:00
112   2014-10-02  06:22:00
113   2014-10-02  06:38:00
114   2014-10-02  07:26:00   ,
            gpsDateTime    gpsLong     gpsLat  gpsAltitude        epoch  \
```

```
115 2014-10-02 07:43:00   15.499767  54.145750          0.000  1412235780
116 2014-10-02 11:43:00   15.236950  54.089233          0.000  1412250180
117 2014-10-02 11:59:00   15.236767  54.089383          0.003  1412251140
118 2014-10-02 12:15:00   15.236850  54.089300          0.005  1412252100
119 2014-10-02 12:31:00   15.236900  54.089283          0.381  1412253060
120 2014-10-02 13:04:00   15.236850  54.089267          0.928  1412255040
121 2014-10-02 13:20:00   15.236750  54.089433          0.337  1412256000
122 2014-10-02 13:36:00   15.236067  54.090017          0.267  1412256960
123 2014-10-02 13:52:00   15.236983  54.089300          0.357  1412257920
124 2014-10-02 14:24:00   15.237417  54.089350          0.140  1412259840
125 2014-10-02 14:41:00   15.236900  54.089200          0.315  1412260860
126 2014-10-02 14:57:00   15.237067  54.089250          0.132  1412261820


            dates     times
115    2014-10-02  07:43:00
116    2014-10-02  11:43:00
117    2014-10-02  11:59:00
118    2014-10-02  12:15:00
119    2014-10-02  12:31:00
120    2014-10-02  13:04:00
121    2014-10-02  13:20:00
122    2014-10-02  13:36:00
123    2014-10-02  13:52:00
124    2014-10-02  14:24:00
125    2014-10-02  14:41:00
126    2014-10-02  14:57:00    ,
            gpsDateTime    gpsLong     gpsLat  gpsAltitude      epoch  \
127 2014-10-02 15:29:00   15.236800  54.088833        0.136  1412263740
128 2014-10-02 15:45:00   15.237250  54.090200        0.133  1412264700
129 2014-10-02 19:45:00   15.111483  54.130650        0.133  1412279100
130 2014-10-02 20:01:00   15.111733  54.131350        0.130  1412280060
131 2014-10-02 20:17:00   15.111050  54.132650        0.133  1412281020
132 2014-10-02 20:34:00   15.109267  54.131633        0.133  1412282040
133 2014-10-02 21:06:00   15.107017  54.133217        0.132  1412283960
```

```
134  2014-10-02 21:22:00  15.106033  54.133567      0.128  1412284920
135  2014-10-02 21:38:00  15.103083  54.134050      0.124  1412285880
136  2014-10-02 21:54:00  15.101467  54.133400      0.129  1412286840
137  2014-10-02 22:26:00  15.096867  54.132917      0.132  1412288760
138  2014-10-02 22:42:00  15.093067  54.133083      0.130  1412289720


          dates      times
127  2014-10-02   15:29:00
128  2014-10-02   15:45:00
129  2014-10-02   19:45:00
130  2014-10-02   20:01:00
131  2014-10-02   20:17:00
132  2014-10-02   20:34:00
133  2014-10-02   21:06:00
134  2014-10-02   21:22:00
135  2014-10-02   21:38:00
136  2014-10-02   21:54:00
137  2014-10-02   22:26:00
138  2014-10-02   22:42:00   ,
            gpsDateTime     gpsLong      gpsLat  gpsAltitude       epoch  \
139  2014-10-02 23:30:00  15.093517  54.138450      0.130  1412292600
140  2014-10-02 23:46:00  15.094350  54.139350      0.133  1412293560
141  2014-10-03 03:47:00  15.093983  54.146717      0.128  1412308020
142  2014-10-03 04:03:00  15.095067  54.147683      0.134  1412308980
143  2014-10-03 04:19:00  15.098000  54.149517      0.128  1412309940
144  2014-10-03 04:35:00  15.104433  54.147617      0.119  1412310900
145  2014-10-03 05:23:00  15.152317  54.096800      0.121  1412313780
146  2014-10-03 05:39:00  15.190917  54.093067      0.132  1412314740
147  2014-10-03 05:55:00  15.197550  54.096967      0.133  1412315700
148  2014-10-03 06:27:00  15.243367  54.089450      0.135  1412317620
149  2014-10-03 06:43:00  15.243600  54.089300      0.135  1412318580
150  2014-10-03 06:59:00  15.244167  54.089067      0.133  1412319540


          dates      times
```

```
139   2014-10-02   23:30:00
140   2014-10-02   23:46:00
141   2014-10-03   03:47:00
142   2014-10-03   04:03:00
143   2014-10-03   04:19:00
144   2014-10-03   04:35:00
145   2014-10-03   05:23:00
146   2014-10-03   05:39:00
147   2014-10-03   05:55:00
148   2014-10-03   06:27:00
149   2014-10-03   06:43:00
150   2014-10-03   06:59:00   ,
            gpsDateTime     gpsLong      gpsLat  gpsAltitude       epoch  \
151 2014-10-03 07:15:00   15.244133   54.089417        0.133  1412320500
152 2014-10-03 07:33:00   15.244200   54.089733        0.133  1412321580
153 2014-10-03 07:47:00   15.243783   54.089867        0.130  1412322420
154 2014-10-03 11:49:00   15.238017   54.089583        0.136  1412336940
155 2014-10-03 12:05:00   15.238167   54.089050        0.158  1412337900
156 2014-10-03 12:22:00   15.238100   54.088700        0.158  1412338920
157 2014-10-03 12:37:00   15.236933   54.088900        0.160  1412339820
158 2014-10-03 13:10:00   15.237350   54.089000        0.162  1412341800
159 2014-10-03 13:26:00   15.237433   54.089017        0.109  1412342760
160 2014-10-03 13:42:00   15.237433   54.088983        0.099  1412343720
161 2014-10-03 13:58:00   15.237433   54.088983        0.106  1412344680
162 2014-10-03 14:30:00   15.236800   54.088517        0.101  1412346600


           dates      times
151   2014-10-03   07:15:00
152   2014-10-03   07:33:00
153   2014-10-03   07:47:00
154   2014-10-03   11:49:00
155   2014-10-03   12:05:00
156   2014-10-03   12:22:00
157   2014-10-03   12:37:00
```

96

```
158   2014-10-03   13:10:00
159   2014-10-03   13:26:00
160   2014-10-03   13:42:00
161   2014-10-03   13:58:00
162   2014-10-03   14:30:00   ,
            gpsDateTime     gpsLong      gpsLat  gpsAltitude        epoch  \
163 2014-10-03 14:47:00  15.236733   54.088233        0.105   1412347620
164 2014-10-03 15:35:00  15.237633   54.088950        0.101   1412350500
165 2014-10-03 15:51:00  15.243483   54.091083        0.108   1412351460
166 2014-10-03 19:52:00  14.787900   54.018417        0.103   1412365920
167 2014-10-03 20:08:00  14.787250   54.018517        0.104   1412366880
168 2014-10-03 20:24:00  14.786450   54.018783        0.101   1412367840
169 2014-10-03 20:40:00  14.785433   54.019250        0.113   1412368800
170 2014-10-03 21:28:00  14.784967   54.019617        0.127   1412371680
171 2014-10-03 21:44:00  14.786017   54.018650        0.123   1412372640
172 2014-10-03 22:00:00  14.785600   54.018983        0.123   1412373600
173 2014-10-03 22:32:00  14.784983   54.020017        0.104   1412375520
174 2014-10-03 22:48:00  14.785433   54.020367        0.099   1412376480


          dates      times
163   2014-10-03   14:47:00
164   2014-10-03   15:35:00
165   2014-10-03   15:51:00
166   2014-10-03   19:52:00
167   2014-10-03   20:08:00
168   2014-10-03   20:24:00
169   2014-10-03   20:40:00
170   2014-10-03   21:28:00
171   2014-10-03   21:44:00
172   2014-10-03   22:00:00
173   2014-10-03   22:32:00
174   2014-10-03   22:48:00   ,
            gpsDateTime     gpsLong      gpsLat  gpsAltitude        epoch  \
175 2014-10-03 23:04:00  14.785533   54.020600        0.099   1412377440
```

97

```
176  2014-10-03 23:20:00   14.785650   54.021083        0.098   1412378400
177  2014-10-03 23:36:00   14.785433   54.021517        0.098   1412379360
178  2014-10-03 23:52:00   14.785767   54.021817        0.103   1412380320
179  2014-10-04 03:53:00   14.778733   54.025167        0.103   1412394780
180  2014-10-04 04:09:00   14.778000   54.025550        0.104   1412395740
181  2014-10-04 04:25:00   14.777683   54.025383        0.096   1412396700
182  2014-10-04 04:41:00   14.778167   54.025033        0.104   1412397660
183  2014-10-04 05:29:00   14.780733   53.994583        0.099   1412400540
184  2014-10-04 05:45:00   14.780900   53.994617        0.098   1412401500
185  2014-10-04 06:01:00   14.780883   53.994533        0.099   1412402460
186  2014-10-04 06:33:00   14.780950   53.994300        0.098   1412404380


              dates      times
175      2014-10-03   23:04:00
176      2014-10-03   23:20:00
177      2014-10-03   23:36:00
178      2014-10-03   23:52:00
179      2014-10-04   03:53:00
180      2014-10-04   04:09:00
181      2014-10-04   04:25:00
182      2014-10-04   04:41:00
183      2014-10-04   05:29:00
184      2014-10-04   05:45:00
185      2014-10-04   06:01:00
186      2014-10-04   06:33:00  ,
             gpsDateTime     gpsLong      gpsLat  gpsAltitude        epoch  \
187  2014-10-04 06:49:00   14.780833   53.994033        0.103   1412405340
188  2014-10-04 07:38:00   14.778800   53.993117        0.103   1412408280
189  2014-10-04 07:53:00   14.718333   53.959217        0.103   1412409180
190  2014-08-30 12:39:00   75.312300   71.454100        0.101   1409402340
191  2014-08-30 14:42:00   75.347050   71.463450        0.102   1409409720
192  2014-08-30 18:49:00   75.379600   71.457583        0.103   1409424540
193  2014-09-27 10:54:00   49.504100   67.734967        0.100   1411815240
194  2014-09-27 11:09:00   49.503800   67.735100        0.098   1411816140
```

98

```
195 2014-09-27 11:25:00  49.503767  67.735100        0.099  1411817100
196 2014-09-27 11:40:00  49.503667  67.735100        0.100  1411818000
197 2014-09-27 11:56:00  49.503650  67.735100        0.100  1411818960
198 2014-09-27 12:11:00  49.503617  67.735000        0.092  1411819860


          dates     times
187  2014-10-04  06:49:00
188  2014-10-04  07:38:00
189  2014-10-04  07:53:00
190  2014-08-30  12:39:00
191  2014-08-30  14:42:00
192  2014-08-30  18:49:00
193  2014-09-27  10:54:00
194  2014-09-27  11:09:00
195  2014-09-27  11:25:00
196  2014-09-27  11:40:00
197  2014-09-27  11:56:00
198  2014-09-27  12:11:00  ]
```

2.28. Spatio-Temporal filter and Interpolation process (ST-IDW)

Once we have requested the swarm data, now we need to `filter` in space and time the available points to compute the magnetic values (NEC frame) for each GPS point based on its particular date and time. The function ST_IDW_Process imported in the row_handler, takes the GPS track and the downloaded data from swarm to filter in space and time based on the criteria defined in our method. With the swarm data filtered we interpolated (IDW) the NEC components for each GPS data point, based on the latitude, date, time and number of Swarm points filtered.

The function CHAOS_ground_values, inside the MagGeoFunctions file,

is used to run the **Calculation of magnetic components**. This calculation requeries the magnetic components at the trajectory altitude (or at the ground level) using CHAOS (theta, phi, radial). This process include a rotation and transformation between a geocentric frame (CHAOS) and geodetic frame (GPS track). Once the corrected values are calculated, are included in the GPS track, and the non-necesary columns are removed. For more information about this process go to the Main Notebook.

2.28.1. Run the (ST-IDW) process in parallel mode

Although the next cell seems to run a small `main` function. What is happening is a call for several functions running at same time for several cores. Initially we set a pool of processes. Using the `pool` class we will distribute the assigned function among the data chucks we created. Every data chunk will be like a subset of the entire GPS track. So we need to iterate among data chunk. And inside every data chunk we need to identify the `datetime`, `epoch`, `altitude`, `latitude` and `longitude` of each row to run the interpolation & annotation process using the Swarm data we have filtered and stored in the previous steps.

The function in charge to distribute the required function (`row_handler`) among the data chunks is the map function from the `pool` class.

`row_handler.py` is an interows iteration to get the required parameter for the `ST_IDW_Process` function.

Auxiliary Functions:

ST_IDW_Process function: This is the main function in charge to read the Swarm Data already filtered, and then import DfTime_func, distance_to_GPS, Kradius , DistJ functions to compute the spatial-time cylinder and the annotation process. The return of this function is a row (dictionary) that will be appended into a python list where all the results

100

from the different cores. The python list from every process is concatenated into a pandas dataframe in the main function having there the whole chain of the parallel process.

distance_to_GPS function: Is the function in charge to calculate the distance between each GPS Point and the Swarm Point.

Kradius function: Is the function in charge to compute the R (radius) value in the cylinder. The R value will be considered based on the latitude of each GPS Point.

DistJ function: This function will calculate the d value as the hypotenuse created in the triangle created amount the locations of the GPS point, the location of the Swarm points and the radius value.

DfTime_func function: This is a time function to selected the points in the range of a the DeltaTime - DT window. The Delta time window has been set as 4 hours for each satellite trajectory.

CHAOS_ground_values function: This is the calculation of geomagnetic components function to get the CHAOS magnetic values and process the Nres,Eres,Cres values and transform them into the N,E,C values at the GPS altitude.

```python
%%time
from functools import partial
from utilities.row_handler import row_handler


if __name__ == '__main__':
    with multiprocessing.Pool(NumCores) as pool:
        GeoMagParallelResult = pd.concat(pool.map(partial(row_handler),df_chunks), ignore_in
```

Annotating the GPS Trayectory: 100%| | 12/12 [00:00<00:00, 18.64it/s]

```
CPU times: user 245 ms, sys: 265 ms, total: 510 ms
Wall time: 17.5 s
```

With the Parallel mode the Annotation process takes about 12 seconds to complete (We had tested the parallel process in a windows server machine with 12 cores, see the image bellow). With the same GPS track in the sequetial mode the process is complete in about 2 minutes. In the image bellow you can see how the machine create several python processes and all cores (full CPU capacity) is taken.

Multiprocessing:

is even more powerfull when you have to process a big amount of data (e.g. 2 millons of points). Although here is making a notable improvement if you have to process a big dataset the parallelization makes even more sense.

Be aware that there is no output cell in here, you can follow the parallelization progress in the Anaconda Prompt.

2.29. The final result

With the NEC components for each GPS Track point, it is possible to compute the aditional magnetic components. For more information about the magnetic components and their relevance go to the main paper or notebook.

```
<strong> The annotated dataframe will include the following attributes:</strong> If you nee
<ul>
  <li><strong>Latitude</strong> from the GPS Track.</li>
  <li><strong>Longitude</strong> from the GPS Track.</li>
  <li><strong>Timestamp</strong> from the GPS Track.</li>
  <li><strong>Magnetic Field Intensity</strong>  mapped as Fgps in nanoTeslas (nT).</li>
  <li><strong>N (Northwards) component</strong> mapped as N in nanoTeslas (nT).</li>
  <li><strong>E (Eastwards) component</strong> mapped as E. in nanoteslas (nT).</li>
  <li><strong>C (Downwards or Center)</strong> component mapped as C in nanoTeslas (nT).</li
  <li><strong>Horizontal component</strong> mapped as H in nanoTeslas (nT).</li>
  <li><strong>Magnetic Inclination </strong> mapped as I in degrees.</li>
  <li><strong>Magnetic Declination or dip angle</strong> mapped as D in degrees</li>
  <li><strong>Kp Index</strong> mapped as kp</li>
  <li><strong>Total Points</strong> as the amount of Swarm messuares included in the ST-IDW
  <li><strong>Minimum Distance</strong> mapped as MinDist, representing the minimum distance
  <li><strong>Average Distance</strong> mapped as AvDist, representing the average distance
</ul>
```

```
#14. Having Intepolated and weigth magnetic values, we can compute the other magnectic compo
GeoMagParallelResult['H'] = np.sqrt((GeoMagParallelResult['N']**2)+(GeoMagParallelResult['E'
#check the arcgtan in python., From arctan2 is saver.
DgpsRad = np.arctan2(GeoMagParallelResult['E'],GeoMagParallelResult['N'])
GeoMagParallelResult['D'] = np.degrees(DgpsRad)
IgpsRad = np.arctan2(GeoMagParallelResult['C'],GeoMagParallelResult['H'])
GeoMagParallelResult['I'] = np.degrees(IgpsRad)
```

```
GeoMagParallelResult['F'] = np.sqrt((GeoMagParallelResult['N']**2)+(GeoMagPar
GeoMagParallelResult
```

	Latitude	Longitude	Altitude	DateTime	TotalPoints	Minimum_Di
0	70.854717	68.307333	0.000	2014-09-08 05:54:00	46	327.950987
1	70.830300	67.975050	0.406	2014-09-08 06:10:00	46	340.038476
2	70.761717	67.752417	0.498	2014-09-08 06:26:00	55	348.223318
3	70.686517	67.561983	0.787	2014-09-08 06:42:00	55	355.472899
4	70.685450	67.548317	0.337	2014-09-08 07:14:00	55	355.980432
...	...	...	...	...	...	...
194	67.735100	49.503800	0.098	2014-09-27 11:09:00	58	314.462546
195	67.735100	49.503767	0.099	2014-09-27 11:25:00	39	314.463935
196	67.735100	49.503667	0.100	2014-09-27 11:40:00	39	314.468143
197	67.735100	49.503650	0.100	2014-09-27 11:56:00	39	314.468858
198	67.735000	49.503617	0.092	2014-09-27 12:11:00	39	314.470867

The previous dataframe (GPS_ResInt), MagGeo has computed the geomagnetic components for each locations and time of your CSV trajectory. Now we will finish up combining the original atributes from your CSV with the annotated results from MagGeo.

```
%%time

originalGPSTrack=pd.read_csv(os.path.join(data_dir,gpsfilename))
MagGeoResult = pd.concat([originalGPSTrack, GeoMagParallelResult], axis=1)
#Drop duplicated columns. Latitude, Longitued, and DateTime will not be part
MagGeoResult.drop(columns=['Latitude', 'Longitude', 'DateTime'], inplace=True
MagGeoResult
```

```
CPU times: user 3.91 ms, sys: 1.05 ms, total: 4.95 ms
Wall time: 15.9 ms
```

104

	timestamp	location-long	location-lat	height	individual_id	Altitude	TotalPoints	Minir
0	08/09/2014 05:54	68.307333	70.854717	0.000	1	0.000	46	327.9
1	08/09/2014 06:10	67.975050	70.830300	0.406	1	0.406	46	340.0
2	08/09/2014 06:26	67.752417	70.761717	0.498	1	0.498	55	348.2
3	08/09/2014 06:42	67.561983	70.686517	0.787	1	0.787	55	355.4
4	08/09/2014 07:14	67.548317	70.685450	0.337	1	0.337	55	355.9
...	...	...	...	...	...	...	...	...
194	27/09/2014 11:09	49.503800	67.735100	0.098	2	0.098	58	314.4
195	27/09/2014 11:25	49.503767	67.735100	0.099	2	0.099	39	314.4
196	27/09/2014 11:40	49.503667	67.735100	0.100	2	0.100	39	314.4
197	27/09/2014 11:56	49.503650	67.735100	0.100	2	0.100	39	314.4
198	27/09/2014 12:11	49.503617	67.735000	0.092	2	0.092	39	314.4

2.30. Export the final results to a CSV file

```
%%time
#Exporting the CSV file
outputfile ="GeoMagResult_"+gpsfilename
export_csv = MagGeoResult.to_csv (os.path.join(results_dir,outputfile), index = None, header
```

```
CPU times: user 6.95 ms, sys: 2.86 ms, total: 9.8 ms
Wall time: 53.7 ms
```

2.31. Validate the results (optional)

To validate the results we plot the Fcolumn.

105

```
## Creating a copy of the results and setting the Datetime Column as datafram
ValidateDF = GeoMagParallelResult.copy()
ValidateDF.set_index("DateTime", inplace=True)
## Plotting the F column.
hist = ValidateDF.hist(column='F')
plt.title('F distribution')
plt.xlabel('F in nT')
plt.ylabel('# of measurements')
```

```
Text(0, 0.5, '# of measurements')
```
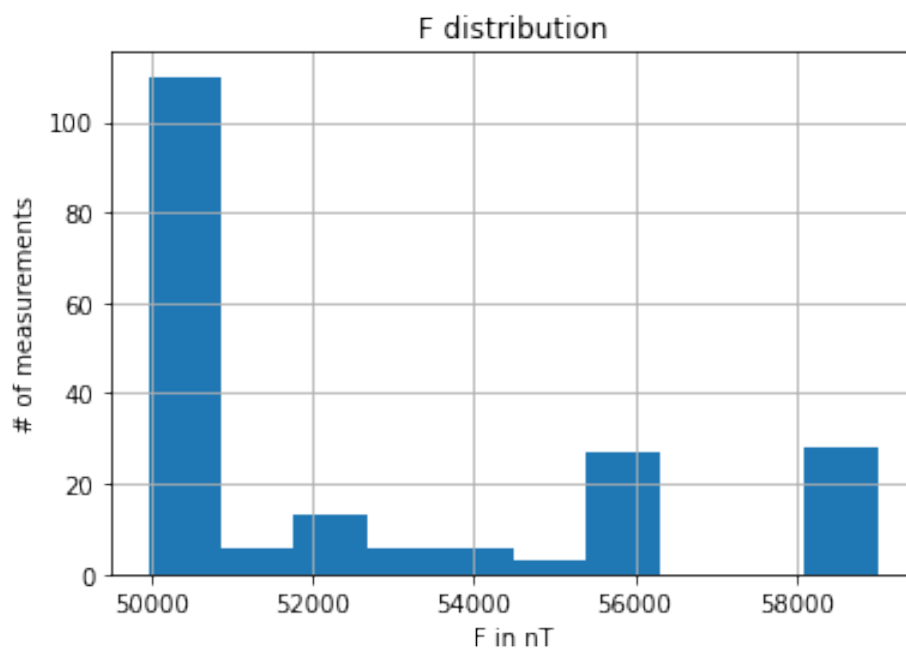
2.32. Mapping the GPS Track using the annotated Magnetic Values (optional)

Now we are going to plot the annotated GPS track stored into the MagDataFinal dataframe to see how the different magnetic components in a map to have a better prespective of the impact of the earth magnetic field.

```python
ValidateDF.plot(kind="scatter", x="Latitude", y="Longitude",
    label="Magnetic Intensity in nT",
    c="F", cmap=plt.get_cmap("gist_rainbow"),
    colorbar=True, alpha=0.4, figsize=(10,7),
    sharex=False #This is only needed to get the x-axis label working due to a current bug i
)

plt.ylabel("Longitude", fontsize=12)
plt.xlabel("Latitude", fontsize=10)
plt.legend(fontsize=12)
plt.show()
```
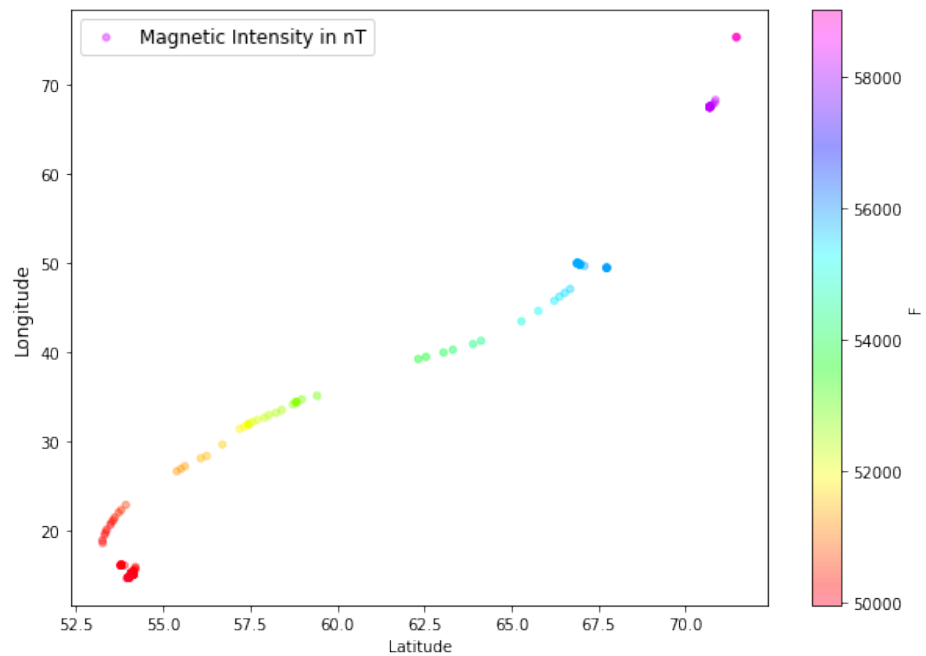
```
import geopandas
import geoplot
import hvplot.pandas
gdf = geopandas.GeoDataFrame(ValidateDF, geometry=geopandas.points_from_xy(Va
gdf.head()
```

Unable to display output for mime type(s): application/javascript, applicatic

Unable to display output for mime type(s): application/javascript, applicatic

Unable to display output for mime type(s): text/html

108

2.32. Mapping the GPS Track using the annotated Magnetic Values (optional)

DateTime	Latitude	Longitude	Altitude	TotalPoints	Minimum_Distance	Average_Distan
2014-09-08 05:54:00	70.854717	68.307333	0.000	46	327.950987	665.008368
2014-09-08 06:10:00	70.830300	67.975050	0.406	46	340.038476	667.146029
2014-09-08 06:26:00	70.761717	67.752417	0.498	55	348.223318	678.815409
2014-09-08 06:42:00	70.686517	67.561983	0.787	55	355.472899	680.040733
2014-09-08 07:14:00	70.685450	67.548317	0.337	55	355.980432	680.062802

```
gdf.hvplot(title=f'Annotated trajectory using MagGeo - F GeoMag Intensity',
          geo=True,
          c='F',
          tiles='CartoLight',
          frame_width=700,
          frame_height=500)
```

```
:Overlay
   .Tiles.I  :Tiles    [x,y]
   .Points.I :Points    [Longitude,Latitude]    (F)
```

```
gdf.hvplot(title=f'Annotated trajectory using MagGeo - I Inclination',
          geo=True,
          tiles='CartoLight',
          c='I',
          cmap='Viridis',
          frame_width=700,
          frame_height=500)
```

```
:Overlay
   .Tiles.I  :Tiles    [x,y]
   .Points.I :Points    [Longitude,Latitude]    (I)
```

109

```python
world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowres
ax = world.plot(color='white', edgecolor='black', figsize = (12,6))

minx, miny, maxx, maxy = gdf.total_bounds
ax.set_xlim(minx, maxx)
ax.set_ylim(miny, maxy)

# We can now plot our ``GeoDataFrame``.
gdf.plot(ax=ax, column='F', legend=True,
         legend_kwds={'label': "Magnetic Intensity in nT",
                      'orientation': "horizontal"})
plt.ylabel("Longitude", fontsize=9)
plt.xlabel("Latitude", fontsize=9)

plt.show()
```
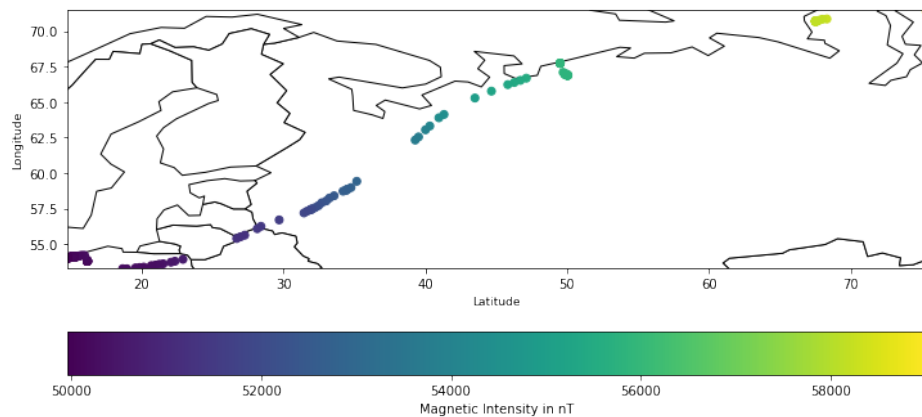


```python
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize = (15,6))

ax1 = world.plot(ax=ax1, color='white', edgecolor='black')
xlim = ([gdf.total_bounds[0],  gdf.total_bounds[2]])
```

```
ylim = ([gdf.total_bounds[1],  gdf.total_bounds[3]])
ax1.set_xlim(xlim)
ax1.set_ylim(ylim)


gdf.plot(ax=ax1, column='F', legend=True,
        legend_kwds={'label': "Magnetic Intensity in nT",
                      'orientation': "horizontal"})
plt.ylabel("Longitude", fontsize=9)
plt.xlabel("Latitude", fontsize=9)
ax1.set_title('Magnetic Intensity - F')
ax1.set_xlabel('Latitude')
ax1.set_ylabel('Longitude')


ax2 = world.plot( ax=ax2, color='white', edgecolor='black')
xlim = ([gdf.total_bounds[0],  gdf.total_bounds[2]])
ylim = ([gdf.total_bounds[1],  gdf.total_bounds[3]])
ax2.set_xlim(xlim)
ax2.set_ylim(ylim)


# We can now plot our ``GeoDataFrame``.
gdf.plot(ax=ax2, column='D', legend=True, cmap='Spectral',
        legend_kwds={'label': " Declination in Degrees",
                      'orientation': "horizontal"})
ax2.set_title('Declination - D')
ax2.set_xlabel('Latitude')
ax2.set_ylabel('Longitude')
```
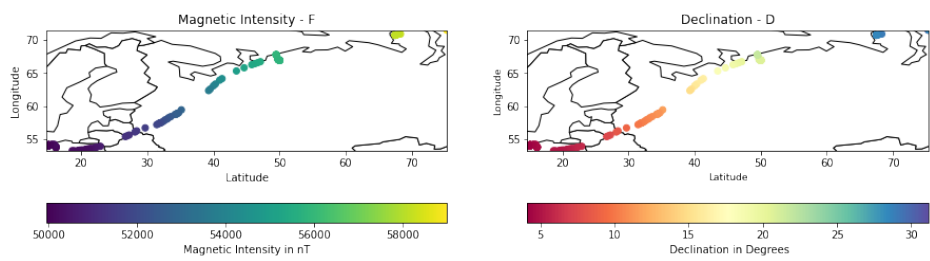
```
Text(567.7954545454544, 0.5, 'Longitude')
```

MagGeo - Parallel Mode

Magnetic Intensity - F

Magnetic Intensity in nT

Declination - D

Declination in Degrees

Troubleshooting Guide

> **i** Note
>
> With Python and in particular with many new ways to use your computer, create folder and scripts things can get messy and very confusing, this is very normal and it is part of the process. The first thing is do not get frustrated or trying all sort of things without understanding what is the root of the issue. So you need to think systematically and then find out the solution. Here I have listed a few potential and common issues you might find. So first take all of this before escalating your issue to the lecturer or the IT support team.

Before Seeking Help

You should complete this checklist:

☐ Miniconda is installed and `conda --version` works
☐ Environment was created without errors
☐ Environment shows `(maggeo)` when activated
☐ Verification script runs successfully
☐ Jupyter Lab starts without errors

Standardized Error Reporting

When reporting problems, always include:

```
# Run these commands and include output
conda info
conda list geopandas
python --version
jupyter --version
```

Issue 1: "conda: command not found"

This is the most common issue for beginners.

Windows Solution:

```
# Option 1: Use the correct command prompt
# Search for "Anaconda Prompt" in Start Menu if available
# Or reinstall Miniconda ensuring PATH is added

# Option 2: Manually add to PATH
set PATH=%PATH%;C:\Users\%USERNAME%\miniconda3\Scripts
set PATH=%PATH%;C:\Users\%USERNAME%\miniconda3
```

macOS Solution:

```
# Add to PATH temporarily
export PATH="$HOME/miniconda3/bin:$PATH"

# Add to PATH permanently
```

```
echo 'export PATH="$HOME/miniconda3/bin:$PATH"' >> ~/.bash_profile
source ~/.bash_profile

# For zsh users (macOS Catalina and later)
echo 'export PATH="$HOME/miniconda3/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Issue 2: Environment Creation Fails

Common causes and solutions:

```
# Solution 1: Clean conda cache
conda clean --all

# Solution 2: Update conda first
conda update conda

# Solution 3: Try creating environment with explicit solver
conda env create -f environment.yml --solver=classic

# Solution 4: Check internet connection and try again
# Large downloads may timeout on slow connections
```

Issue 3: Different Python Versions

All students must have Python 3.10 for consistency.

```
# Check your Python version
python --version

# If incorrect, remove environment and recreate
```

```
conda env remove --name maggeo
conda env create -f environment.yml
```

Issue 4: Package Conflicts During Installation

This indicates environment file issues:

```
# Solution: Use mamba for faster, more reliable solving
conda install mamba -n base -c conda-forge
mamba env create -f environment.yml
```

Issue 5: Jupyter Lab Won't Start

Consistency check:

```
# Ensure environment is activated
conda activate maggeo

# Verify Jupyter installation
jupyter --version

# If missing, reinstall
conda install jupyter jupyterlab -c conda-forge

# Start Jupyter Lab
jupyter lab
```

Issue 6: Import Errors Despite Successful Installation

Environment activation problem:

```
# Always activate environment first
conda activate maggeo

# Check which Python you're using
which python     # macOS/Linux
where python     # Windows

# Should show path to conda environment, not system Python
```

Emergency Reinstallation

If all else fails, sometimes it is better and quicker run a complete clean installation:

```
# Remove environment
conda env remove --name maggeo

# Clean all caches
conda clean --all

# Recreate environment
conda env create -f environment.yml
```

Additional Resources

Use the following instructions as a guide for extra resources and better familiarity with working with Python. In case you want to manage your python environment, export your outcomes to PDF, and get extra learning resources.

Environment Management - Useful Commands

```
# List all environments
conda env list

# Activate environment
conda activate maggeo

# Deactivate environment
conda deactivate

# Update all packages in environment
conda update --all

# Install additional package
conda install package-name

# Remove environment
conda env remove --name maggeo
```

Updating the Environment

```
# Update environment from file
conda env update -f environment.yml --prune
```

Exporting Your Environment

```
# Export current environment
conda env export > my-environment.yml
```