Assignment 5: Magnetic Field Modeling

Abstract

Through this assignment the student will learn about the IGRF magnetic field model, how to transform between different reference frames and how to create algorithms to map the magnetic field vector from the ENU frame to the orbit frame.

1 Objective

The objective with this assignment is to calculate the magnetic field vector in the orbit frame using the IGRF model. This assignment allows the calculation of the magnetic field vector in the orbit frame, which in turn will be used in week 6 for attitude determination together with the Sun Vector Model.

2 International Geomagnetic Reference Model (IGRF)

The International Geomagnetic Reference Model is a high accuracy model for Earth's magnetic field. The IGRF model is modeled as the gradient of the magnetic potential $\mathbf{B}(r,\phi,\theta,t) = -\nabla V(r,\phi,\theta,t)$ where the potential can be written as the sum

$$V(r,\phi,\theta,t) = a\sum_{l=1}^{L}\sum_{m=0}^{l}\left(\frac{a}{r}\right)^{l+1}(g_l^m(t)\cos(m\phi)+h_l^m(t)\sin(m\phi))P_l^m(\cos(\theta))$$

where $g_l^m(t)$ and $h_l^m(t)$ are Gauss-coefficients, ϕ is the longitude and θ is the colatitude, a is the radius of the Earth, r is the radial distance from the center of the Earth while $P_l^m(t)$ is the Schmidt normalized Legrendre polynomial. Note that this notation does not correspond with the rest of the subject. The model is of degree l and order m. Typically, the IGRF is of order 12, such that the model becomes rather complex, and takes much time to implement. Luckily, the model is readily available through the Github module ppigrf.

Install ppigrf using the command "pip install ppigrf" within your pyvista_env in Anaconda Prompt. Details about the "pure-Python" implementation of the IGRF model can be found here: `https://github.com/IAGA-VMOD/ppigrf`. From the example given on Github, the framework can be used to read out magnetic field vector in the East-North-Up (ENU) frame using the following command.

```
import ppigrf
from datetime import datetime

lon = 5.32415  # degrees east
lat = 60.39299 # degrees north
h   = 1000        # kilometers above sea level
date = datetime(2025, 3, 28)

Be, Bn, Bu = ppigrf.igrf(lon, lat, h, date) # returns east,
    north, up
```

```
print("East: ", Be, "nT")
print("North: ", Bn, "nT")
print("Up: ", Bu, "nT")
```

3 Transformations

Before we are able to use the output from the IGRF model to anything useful, we first need to establish the functions needed to transform the vectors from ENU Frame to orbit and back again. This means that we need to find the transformations Orbit→ECI→ECEF→LLA→NED→ENU; where ENU is East North Up, while NED is North East Down.

3.1 From Orbit to Inertial

The transformation from orbit frame to inertial frame can be constructed using the quaternion $\mathbf{q}_{i,o}$, which is defined in previous assignment as:

$$\mathbf{q}_\Omega = \begin{bmatrix}\cos(\frac{\Omega}{2}) & 0 & 0 & \sin(\frac{\Omega}{2})\end{bmatrix}^\top \tag{1}$$

$$\mathbf{q}_i = \begin{bmatrix}\cos(\frac{i}{2}) & \sin(\frac{i}{2}) & 0 & 0\end{bmatrix}^\top \tag{2}$$

$$\mathbf{q}_{\omega+\theta} = \begin{bmatrix}\cos(\frac{\omega+\theta}{2}) & 0 & 0 & \sin(\frac{\omega+\theta}{2})\end{bmatrix}^\top \tag{3}$$

where

$$\mathbf{q}_{i,o} = \mathbf{q}_{\omega+\theta} \otimes \mathbf{q}_i \otimes \mathbf{q}_\Omega \tag{4}$$

and Ω is the right ascention of the ascending node, ω is the argument of perigee, θ is the true anomaly, and i is the inclination. The rotation matrix can then be constructed as

$$\mathbf{R}_o^i = \mathbf{I} + 2\eta_{i,o}\mathbf{S}(\boldsymbol{\epsilon}_{i,o}) + 2\mathbf{S}^2(\boldsymbol{\epsilon}_{i,o}). \tag{5}$$

3.2 From Inertial to ECEF

The rotation matrix from Inertial to ECEF can be constructed as

$$\mathbf{R}_e^i = \begin{bmatrix}\cos(\omega_{i,e}t) & -\sin(\omega_{i,e}t) & 0 \\ \sin(\omega_{i,e}t) & \cos(\omega_{i,e}t) & 0 \\ 0 & 0 & 1\end{bmatrix}$$

where $\omega_{i,e}$ is the angular speed of the Earth, and where it can be transposed to go from Inertial to ECEF such that

$$\mathbf{R}_i^e = (\mathbf{R}_e^i)^\top \tag{6}$$

Table 1: Parameters used for the ECEF to LLA algorithm

Parameters	Description
$a_e = 6378137$ m	Semi-major axis
$b_e = 6356725$ m	Semi-minor axis
$\omega_{ie} = 7.292115 \cdot 10^{-5}$	Angular speed of Earth
$e_e = 0.0818$	Eccentricity of reference ellipsoid

3.3 From ECEF to LLA

To transform a position in ECEF frame to Latitude (μ), Longtude (l), Altitude (h) (LLA) is not straight forward. To that end, (Fossen, 2011, pp. 36-37), shows how to find the LLA using an interative algorithm. The approach considers the WGS-84 reference ellipsoid to represent the Earth, where several parameters are needed as shown in Table 1. Let $\mathbf{r}^e = \begin{bmatrix} x & y & z \end{bmatrix}^\top$, then algorithm is as follows:

1. Compute $p = \sqrt{x^2 + y^2}$
2. Compute the approximate value μ from
$$\tan(\mu) = \frac{z}{p}(1 - e_e^2)^{-1}$$
3. Compute an approximate value N (radius of curvature) from
$$N = \frac{a_e^2}{\sqrt{a_e^2 \cos^2(\mu) + b_e^2 \sin^2(\mu)}}$$
4. Compute the ellipsoidal height by
$$h = \frac{p}{\cos(\mu)} - N$$
5. Compute an improved value for the latitude by
$$\tan(\mu) = \frac{z}{p}\left(1 - e_e^2 \frac{N}{N + h}\right)^{-1}$$
6. If $|\mu_k - \mu_{k-1}| < 1 \cdot 10^{-6}$, end the algorithm, otherwise continue with step 3.

After convergence, the longitude can be found as $l = \text{atan2}(y, x)$. For convention, this can be implemented as

```python
def calculate_lla_from_ecef(r_e):
    # Implementing the algorithm from Fossen, 2011.

    # Earth WGS-84 parameters
    a_e = 6378137
    b_e = 6356725
    w_ie = 7.292115e-5
    e_e = 0.0818
```

```
    x = r_e[0]
    y = r_e[1]
    z = r_e[2]

    # 1. Compute p = sqrt(x^2+y^2)
    p = np.sqrt(x**2+y**2)

    # 2. Compute the approximate value of mu
    mu = np.arctan((z/p)*(1-e_e**2)**-1)

    mu_old = 10
    while abs(mu - mu_old) > 1e-6:
        # Storing old value
        mu_old = mu

        # 3. Compute an approximate value N (radius of
curvature)
        N = a_e**2/np.sqrt(a_e**2*np.cos(mu)**2+b_e**2*np.
sin(mu)**2)

        # 4. Compute the ellipsoidal height
        h = p/np.cos(mu) - N

        # 5. Compute an improved value for the latitude by
        mu = np.arctan((z/p)*(1-e_e**2*(N/(N+h)))**-1)

    # Calculate longitude
    l = np.arctan2(y, x)

    latitude = mu
    longitude = l
    altitude = h

    return latitude, longitude, altitude
```

3.4 From ECEF to NED

With the angle for the longitude and latitude, the rotation from NED to ECEF can be constructed as (Fossen, 2011, p34)

$$\mathbf{R}_n^e = \begin{bmatrix} -\cos(l)\sin(\mu) & -\sin(l) & -\cos(l)\cos(\mu) \\ -\sin(l)\sin(\mu) & \cos(l) & -\sin(l)\cos(\mu) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{7}$$

where l is the longitude and μ is the latitude. This allows the transformation matrix from ECEF to NED to be found as

$$\mathbf{R}_e^n = (\mathbf{R}_n^e)^\top. \tag{8}$$

3.5 From NED to ENU

To transform the magnetic field model from the North-East-Down (NED) frame to the East-North-Up (ENU) frame, the rotation matrix that rotates from NED

to ENU can be constructed as

$$\mathbf{R}_n^u = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{9}$$

which changes the order between North and East, and changes the direction of the z-component. Consider a vector $\mathbf{v}^n = \begin{bmatrix} 1 & 2 & -3 \end{bmatrix}^\top$, then it can be found in ENU as

$$\mathbf{v}^u = \mathbf{R}_n^u \mathbf{v}^n = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}. \tag{10}$$

Transformations

1. Create the following functions
   - calculate_rotation_matrix_from_inertial_to_ecef()
   - calculate_lla_from_ecef()
   - calculate_rotation_matrix_from_ecef_to_ned()
   - calculate_rotation_matrix_from_ned_to_enu()
2. The function calculate_rotation_matrix_from_orbit_to_inertial() should be available from earlier assignments.
3. Let $\Omega = 0$, $\omega = 0$, $i = 75\frac{\pi}{180}$, $\theta = 30\frac{\pi}{180}$ and use that to calculate the rotation matrix from orbit to inertial.
4. Let $t = 30s$ and use that to calculate the rotation matrix from ECI to ECEF.
5. Let a position vector be defined as $\mathbf{r}^o = \begin{bmatrix}6420652 & 5236678 & 1111957\end{bmatrix}^\top$, and show that the vector is in the following frames together with the latitude, longitude and altitude:
   - $\mathbf{r}^i = \begin{bmatrix}2942109 & 930595 & 7769299\end{bmatrix}^\top$
   - $\mathbf{r}^e = \begin{bmatrix}2944132 & 924174 & 7769299\end{bmatrix}^\top$
   - $\mathbf{r}^n = \begin{bmatrix}-14629 & 0 & -8359653\end{bmatrix}^\top$
   - $\mathbf{r}^u = \begin{bmatrix}0 & -14629 & 8359653\end{bmatrix}^\top$
   - Latitude = 68.4385°, Longitude = 17.4273°, Altitude = 1999968
6. Create a new function called calculate_magnetic_field_in_orbit_frame(r_i, R_o_i, date, t) that reads out the magnetic field vector in ENU frame, and then converts it all the way to the orbit frame using the different rotation matrices developed through this assignment. Remember to convert the altitude from meters to km, and radians to degrees, and then return the magnetic field in Teslas (multiply by $1 \cdot 10^{-9}$).
   - Given a radius vector $\mathbf{r}^i = \begin{bmatrix}2938363 & 942355 & 7769299\end{bmatrix}^\top$ and a date 2025-01-10, and similar orbit as defined above, the magnetic field vector should be in ENU and orbit frame respectively:

     $$\mathbf{b}^u = \begin{bmatrix}207.364 & 5409.098 & -24245.019\end{bmatrix}^\top \cdot 10^{-9}$$
     $$\mathbf{b}^o = \begin{bmatrix}-22006.422 & -11440.268 & -1399.984\end{bmatrix}^\top \cdot 10^{-9}$$

   - Confirm that you obtain similar results.
7. Document your results in the report and explain what you have done.