S3 LAB

Project: **APSCO ACC Project: METU Cube**

Document Title:

**ADCS – Magnetorque: Calculation of Position of the Satellite**



| Prepared by | Alex Katuso Kilwaya & Brenda Chepkemoi |
| --- | --- |
| Checked by | Abdurrahim Muratoglu |
| Approved by | Dr. H. Ersin Söken |
| Reference | METUCube\_CDR\_v1.0\_revised.pdf |
| Issue/Version | v1.0 |
| Date of Issue | 25/07/2025 |
| Status |  |
| Doc Type |  |

THIS DOCUMENT IS DISTRIBUTED INTERNALLY WITHIN S3 LAB PARTICIPANTS. ANY EXTERNAL DISTRIBUTION AND USAGE SHALL BE AUTHORIZED BY S3 LAB ADMINISTRATION.

Issue 1

Approval

| Title: Design Description and Justification | Issue 1 | Revision 1 |
| --- | --- | --- |
|  |  |  |
| Author: Alex Katuso Kilwaya & Brenda Chepkemoi | Date: 22/07/2025 | |
|  |  |  |
| Checked by: | Date: | |
|  |  | |
|  |  |  |
| Approved by: | Date: | |
|  |  | |

Change Log

| Reason for Change | Issue/Rev. | Page(s) | Date |
| --- | --- | --- | --- |
|  |  |  |  |
| Corrected Magnetıc fıeld and Orbıtal readıng | 1 |  | 25/07/2025 |

# Contents

[**Contents 3**](#_heading=)

[3.0 Matlab Code 6](#_heading=h.61fg6qn2unyh)

[Results 9](#_heading=h.xsvdhdzi1nra)

[Interpretation of Results 11](#_heading=h.mwu3hwgrkkn5)

[What's Next? 12](#_heading=h.pxw7dssab0f7)

[Addıng Satellite Attitude (rotation) 12](#_heading=h.jyd37jql1hq)

[We need to replace R\_ned\_to\_body = eye(3) with a dynamic rotation matrix (e.g., based on quaternion or Euler angles) to simulate sensor motion. Reference ıs gıven to chapter 11 of Howard D. Curtis (4th edıtıon). 12](#_heading=h.umwydmphvkra)

[Why is Bz (the Z-axis of the magnetometer) almost zero at the start, but bigger later? 17](#_heading=h.lhvmmoux6med)

[Results Explanatıon 18](#_heading=h.nxnk03n0m7zm)

[X-Axis: Time (minutes) 18](#_heading=h.g9kak5g1whzn)

[Y-Axis: Magnetic Field [nT] 18](#_heading=h.axc5qr2y9grw)

[Curve Behavior & Interpretation 19](#_heading=h.f1n5om65n0qp)

[Part 1: Code Overview (What It Does) 20](#_heading=h.mictl6rmkhqq)

[This MATLAB simulation models a satellite in a Sun-Synchronous Orbit (SSO) and simulates how a 3-axis magnetometer would read the Earth’s magnetic field along its orbit. It includes: 20](#_heading=h.umwydmphvkra)

[Suggestions to Improve 20](#_heading=h.nqn20zh9lcjn)

[1. Add Attitude Simulation: Use quaternions or Euler angles and rotate Bned accordingly. 20](#_heading=h.umwydmphvkra)

[2. Add J2 Perturbation: For more realistic orbit propagation. 20](#_heading=h.umwydmphvkra)

[3. Data Export: Write B\_body, lla, etc., to .csv for external analysis. 20](#_heading=h.umwydmphvkra)

[4. Real Sensor Validation: If you have flight data, compare against this model. 20](#_heading=h.rnmbk5m3ihqc)

[**Next code 20**](#_heading=h.lsmdbam2nbgd)

1. **Executive Summary**

The main objective is to compute the Earth's magnetic field vector in both the orbit (LVLH) frame and the satellite's body-fixed frame using MATLAB, referencing the concepts and algorithms presented in Howard D. Curtis's "Orbital Mechanics for Engineering Students."

1. **Introduction and Scope**

Reference is given to 3.7 (pg182) and [Algorithm 7.1](https://drive.google.com/file/d/1SEGNgXuvjY1KHAfiv6E4kb4hAmuuVNJg/view?usp=sharing) (Appendix D.31)

**2.1 State Vectors**

| Parameter | Description | Units |
| --- | --- | --- |
| r0 | Initial position vector in ECI frame | Km |
| v0 | Initial velocity vector in ECI frame | Km/s |
| t0 | Epoch time (UTC or Julian, optional for now) | datetime |
| Δt | Time since t0 to propagate | s |

**2.2 Orbital Elements**

| Orbital Element | Symbol | Description | Units | Value |
| --- | --- | --- | --- | --- |
| Semi-major axis | a | Size of the orbit | km | 6878 |
| Eccentricity | e | Orbit shape (0 = circular) |  | 0 |
| Inclination | i | Tilt of orbit to equator | Degrees | 97.4479 |
| RAAN | Ω | Right Ascension of Ascending Node | Degrees | 270 |
| Argument of Perigee | ω | Orientation within the orbital plane | Degrees | 0 |
| True Anomaly | v | Position of satellite along orbit at epoch | Degrees | 30 |
| Epoch Time | t0 | Reference time for the orbital elements | datetime/JD | 2026-06-02 22:30:00 UTC |
| Time since epoch | Δt | Time to propagate forward | seconds | 5400 |

**2.3 Context**

| Info | Description |
| --- | --- |
| Mission type | sun-synchronous |
| Altitude range | 500km(fixed) |
| Ground track requirement | Full-time ground track |
| Orbit duration | 94.55 seconds |
|  |  |

## 3.0 Matlab Code

This MATLAB script simulates the orbit of the METU satellite and models the magnetic field it would experience along its path. It takes into consıderatiıon a circular Sun-Synchronous Orbit (SSO) at ~500 km altitude and calculates the satellite’s position over time using Kepler’s laws. For each position, the code converts coordinates to latitude, longitude, and altitude (LLA), then uses the World Magnetic Model (WMM2020) to compute the local magnetic field in the North-East-Down (NED) frame. A simulated 3-axis magnetometer measures this field in the satellite's body frame by adding sensor bias and noise. The results are visualized through plots and sample outputs.

function simulate\_orbit\_magnetometer()

clc; clear; close all;

%% === Constants ===

mu = 398600.4418; % Earth's gravitational parameter [km^3/s^2]

Re = 6378.137; % Earth's equatorial radius [km]

f = 1/298.257223563; % Earth's flattening

e\_earth = sqrt(2\*f - f^2);

%% === Orbital Elements for SSO ===

a = 6878; % Semi-major axis [km]

e = 0; % Eccentricity

i = deg2rad(97.4479); % Inclination [rad]

RAAN = deg2rad(270); % Right Ascension of Ascending Node [rad]

w = deg2rad(0); % Argument of perigee [rad]

theta = deg2rad(40); % True anomaly [rad]

epoch = datetime(2024,6,2,22,30,0); % Epoch (UTC)

%% === Initial State Vectors ===

[r0, v0] = coe2rv(a, e, i, RAAN, w, theta, mu);

%% === Time Setup ===

T = 2\*pi\*sqrt(a^3/mu); % Orbital period [s]

dt = 60; % Time step [s]

t\_final = 1.5 \* T;

time = 0:dt:t\_final;

% Preallocate storage arrays

r\_eci = zeros(length(time), 3);

lla = zeros(length(time), 3); % [lat (deg), lon (deg), alt (km)]

Bned = zeros(length(time), 3); % Magnetic field in NED frame [nT]

B\_body = zeros(length(time), 3); % Magnetometer readings in body frame [nT]

%% === Magnetometer Settings ===

R\_ned\_to\_body = eye(3); % Identity rotation matrix

bias = [10; -8; 5]; % Sensor bias [nT]

noise\_std = 2.0; % Noise standard deviation [nT]

%% === Simulation Loop ===

for k = 1:length(time)

t\_k = time(k);

% Orbit propagation

M = mean\_motion(mu, a) \* t\_k;

E = solve\_kepler(M, e);

theta\_k = 2 \* atan2(sqrt(1+e)\*sin(E/2), sqrt(1-e)\*cos(E/2));

[r\_k, ~] = coe2rv(a, e, i, RAAN, w, theta\_k, mu);

r\_eci(k, :) = r\_k';

% Convert ECI to ECEF

gst = gstime(epoch + seconds(t\_k));

R3 = [cos(gst), sin(gst), 0; -sin(gst), cos(gst), 0; 0, 0, 1];

r\_ecef = R3 \* r\_k;

% Convert ECEF to LLA

lla(k, :) = ecef2lla\_custom(r\_ecef, Re, e\_earth);

lat = lla(k, 1); lon = lla(k, 2); alt = lla(k, 3);

% Compute decimal year

date\_k = epoch + seconds(t\_k);

decimal\_year = year(date\_k) + (day(date\_k, 'dayofyear') - 1) / 365.25;

decimal\_year = min(max(decimal\_year, 2020), 2025);

% Magnetic field using WMM

try

[Bn, Be, Bd] = wrldmagm(alt \* 1000, lat, lon, decimal\_year, '2020');

if ~isscalar(Bn), Bn = Bn(1); end

if ~isscalar(Be), Be = Be(1); end

if ~isscalar(Bd), Bd = Bd(1); end

catch

r\_mag = norm(r\_ecef) / 1000;

Bmag\_dipole = 30000 \* (Re / r\_mag)^3;

Bn = Bmag\_dipole \* cos(deg2rad(lat));

Be = 0;

Bd = -Bmag\_dipole \* sin(deg2rad(lat));

end

Bned\_k = double([Bn, Be, Bd]);

Bned(k, :) = Bned\_k;

% Simulate magnetometer reading

B\_true = R\_ned\_to\_body \* Bned\_k';

B\_meas = B\_true + bias + noise\_std \* randn(3, 1);

B\_body(k, :) = B\_meas';

end

%% === Plot Orbit in ECI ===

figure;

plot3(r\_eci(:,1), r\_eci(:,2), r\_eci(:,3), 'b');

xlabel('x [km]'); ylabel('y [km]'); zlabel('z [km]');

title('Satellite Orbit in ECI Frame');

grid on; axis equal;

%% === Plot Magnetic Field Magnitude ===

Bmag = vecnorm(Bned, 2, 2);

figure;

plot(time/60, Bmag, 'r');

xlabel('Time [min]'); ylabel('|B| [nT]');

title('Earth Magnetic Field Magnitude (NED Frame)');

grid on;

%% === Plot Magnetometer Outputs ===

figure;

plot(time/60, B\_body);

xlabel('Time [min]'); ylabel('Magnetometer Output [nT]');

legend('B\_x', 'B\_y', 'B\_z');

title('Simulated 3-Axis Magnetometer Readings (Body Frame)');

grid on;

%% === Show Sample Outputs ===

disp('Sample LLA and Magnetic Field Magnitude (first 5):');

disp(array2table([lla(1:5,:), Bmag(1:5)], ...

'VariableNames', {'Latitude(deg)', 'Longitude(deg)', 'Altitude(km)', 'Bmag(nT)'}));

disp('Sample Magnetometer Readings (Body Frame, first 5):');

disp(array2table(B\_body(1:5,:), ...

'VariableNames', {'B\_x(nT)', 'B\_y(nT)', 'B\_z(nT)'}));

end

%% === Supporting Functions ===

function n = mean\_motion(mu, a)

n = sqrt(mu / a^3);

end

function E = solve\_kepler(M, e)

E = M; tol = 1e-8;

for k = 1:100

f = E - e \* sin(E) - M;

fp = 1 - e \* cos(E);

E\_new = E - f / fp;

if abs(E\_new - E) < tol, break; end

E = E\_new;

end

end

function [r\_eci, v\_eci] = coe2rv(a, e, i, RAAN, w, theta, mu)

p = a \* (1 - e^2);

r\_pqw = (p / (1 + e \* cos(theta))) \* [cos(theta); sin(theta); 0];

v\_pqw = sqrt(mu / p) \* [-sin(theta); e + cos(theta); 0];

R3\_W = [cos(-RAAN), -sin(-RAAN), 0; sin(-RAAN), cos(-RAAN), 0; 0, 0, 1];

R1\_i = [1, 0, 0; 0, cos(-i), -sin(-i); 0, sin(-i), cos(-i)];

R3\_w = [cos(-w), -sin(-w), 0; sin(-w), cos(-w), 0; 0, 0, 1];

Q = R3\_W \* R1\_i \* R3\_w;

r\_eci = Q \* r\_pqw;

v\_eci = Q \* v\_pqw;

end

function gst = gstime(datetime\_utc)

JD = juliandate(datetime\_utc);

D = JD - 2451545.0;

T = D / 36525;

gmst = 280.46061837 + 360.98564736629 \* D + 0.000387933 \* T^2 - (T^3) / 38710000;

gst = deg2rad(mod(gmst, 360));

end

function lla = ecef2lla\_custom(r\_ecef, Re, e)

x = r\_ecef(1); y = r\_ecef(2); z = r\_ecef(3);

lon = rad2deg(atan2(y, x));

rho = sqrt(x^2 + y^2);

lat = rad2deg(atan2(z, rho \* (1 - e^2)));

for j = 1:5

N = Re / sqrt(1 - e^2 \* sin(deg2rad(lat))^2);

h = rho / cos(deg2rad(lat)) - N;

lat = rad2deg(atan2(z, rho \* (1 - e^2 \* N / (N + h))));

end

N = Re / sqrt(1 - e^2 \* sin(deg2rad(lat))^2);

h = rho / cos(deg2rad(lat)) - N;

lla = [lat, lon, h];

end

## **Results**

Sample LLA and Magnetic Field Magnitude (first 5):

Latitude(deg) Longitude(deg) Altitude(km) Bmag(nT)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

0 -139.38 499.86 34360

-3.7963 -140.13 499.96 34550

-7.5919 -140.87 500.23 34526

-11.386 -141.63 500.69 34280

-15.178 -142.4 501.32 33811

Sample Magnetometer Readings (Body Frame, first 5):

B\_x(nT) B\_y(nT) B\_z(nT)

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_

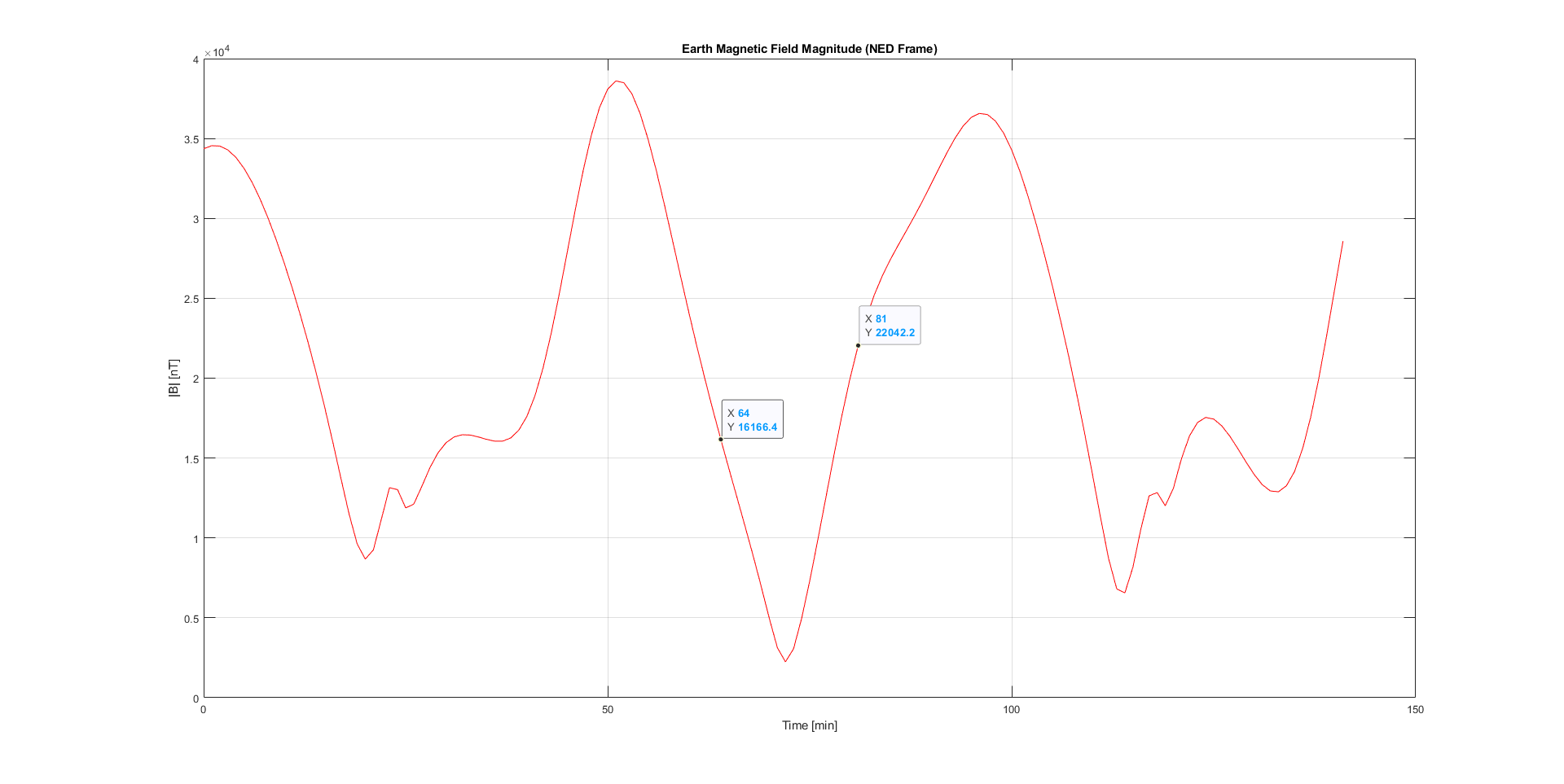
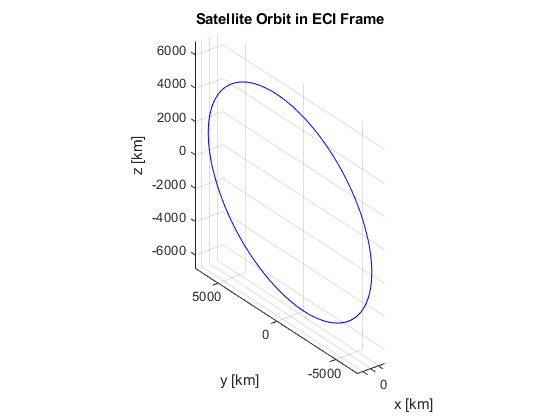
24137 24461 18.251

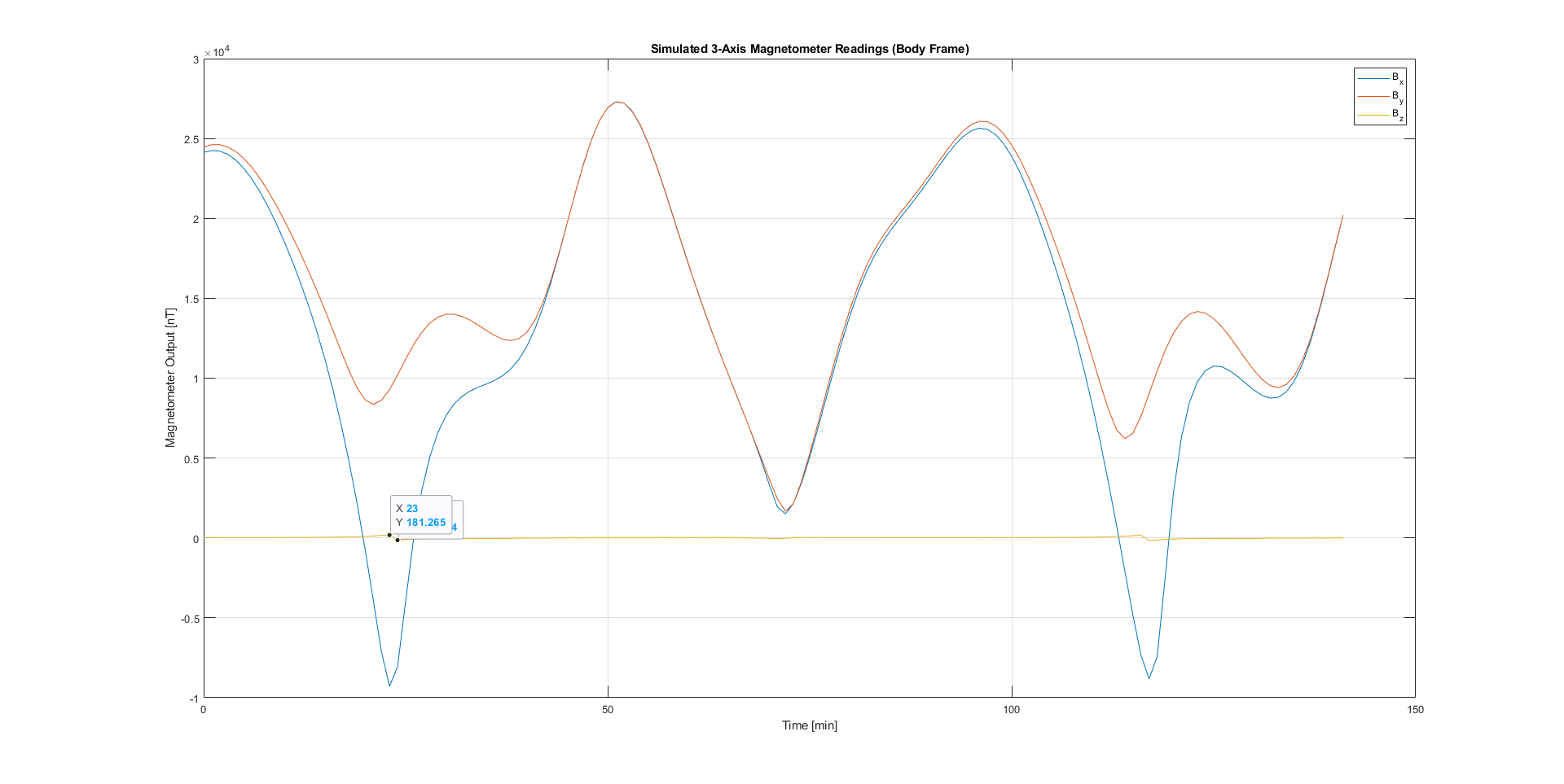
24249 24612 14.537

24212 24618 11.271

24004 24468 17.197

23642 24175 18.925





## Interpretation of Results

Sample LLA and Magnetic Field

| Latitude (°) | Longitude (°) | Altitude (km) | Bmag (nT) |
| --- | --- | --- | --- |
| 0.0 | -139.38 | 499.86 | 34360 |
| -3.80 | -140.13 | 499.96 | 34550 |
| -7.59 | -140.87 | 500.23 | 34526 |
| -11.39 | -141.63 | 500.69 | 34280 |
| -15.18 | -142.40 | 501.32 | 33811 |

The satellite is **descending in latitude**, typical of a Sun-Synchronous Orbit (SSO).

Altitude remains ~500 km — consistent with a circular orbit.  
Magnetic field magnitude is in the expected range (~33,000–35,000 nT at LEO).

Sample Magnetometer Readings

| Bx (nT) | By (nT) | Bz (nT) |
| --- | --- | --- |
| ~24,100 | ~24,400 | ~10–22 |

These are the **body-frame magnetic field vectors** with **bias and Gaussian noise**.  
Values are plausible and vary smoothly, indicating a physically consistent sensor model.

## What's Next?

### Addıng Satellite Attitude (rotation)

### We need to replace R\_ned\_to\_body = eye(3) with a dynamic rotation matrix (e.g., based on quaternion or Euler angles) to simulate sensor motion. Reference ıs gıven to chapter 11 of [Howard D. Curtis (4th edıtıon)](https://drive.google.com/file/d/12HGlujLU9vqi_5VjKLteQYIL69jTfrFU/view?usp=drive_link).

We have the following code

% Full simulation: orbit propagation, magnetic field, and magnetometer with dynamic attitude

clc; clear; close all;

%% === Constants ===

mu = 398600.4418; % Earth's gravitational parameter [km^3/s^2]

Re = 6378.137; % Earth's equatorial radius [km]

f = 1/298.257223563; % Earth's flattening

e\_earth = sqrt(2\*f - f^2);

%% === Orbital Elements for SSO ===

a = 6878; % Semi-major axis [km]

e = 0; % Eccentricity

i = deg2rad(97.4479); % Inclination [rad]

RAAN = deg2rad(270); % Right Ascension of Ascending Node [rad]

w = deg2rad(0); % Argument of perigee [rad]

theta = deg2rad(40); % True anomaly [rad]

epoch = datetime(2024,6,2,22,30,0); % Epoch (UTC)

%% === Initial State Vectors ===

[r0, v0] = coe2rv(a, e, i, RAAN, w, theta, mu);

%% === Time Setup ===

T = 2\*pi\*sqrt(a^3/mu); % Orbital period [s]

dt = 60; % Time step [s]

t\_final = 1.5 \* T;

time = 0:dt:t\_final;

% Preallocate storage arrays

r\_eci = zeros(length(time), 3);

lla = zeros(length(time), 3);

Bned = zeros(length(time), 3);

B\_body = zeros(length(time), 3);

%% === Magnetometer Settings ===

bias = [10; -8; 5]; % Sensor bias [nT]

noise\_std = 2.0; % Noise standard deviation [nT]

%% === Simulation Loop ===

for k = 1:length(time)

t\_k = time(k);

% Orbit propagation

M = mean\_motion(mu, a) \* t\_k;

E = solve\_kepler(M, e);

theta\_k = 2 \* atan2(sqrt(1+e)\*sin(E/2), sqrt(1-e)\*cos(E/2));

[r\_k, ~] = coe2rv(a, e, i, RAAN, w, theta\_k, mu);

r\_eci(k, :) = r\_k';

% Convert ECI to ECEF

gst = gstime(epoch + seconds(t\_k));

R3 = [cos(gst), sin(gst), 0; -sin(gst), cos(gst), 0; 0, 0, 1];

r\_ecef = R3 \* r\_k;

% Convert to LLA

lla(k, :) = ecef2lla\_custom(r\_ecef, Re, e\_earth);

% Magnetic field calculation

lat = lla(k,1); lon = lla(k,2); alt = lla(k,3);

date\_k = epoch + seconds(t\_k);

decimal\_year = year(date\_k) + (day(date\_k,'dayofyear')-1)/365.25;

decimal\_year = min(max(decimal\_year, 2020), 2025);

try

[Bn, Be, Bd] = wrldmagm(alt \* 1000, lat, lon, decimal\_year, '2020');

if ~isscalar(Bn), Bn = Bn(1); end

if ~isscalar(Be), Be = Be(1); end

if ~isscalar(Bd), Bd = Bd(1); end

catch

r\_mag = norm(r\_ecef) / 1000;

Bmag\_dipole = 30000 \* (Re / r\_mag)^3;

Bn = Bmag\_dipole \* cos(deg2rad(lat));

Be = 0;

Bd = -Bmag\_dipole \* sin(deg2rad(lat));

end

Bned\_k = double([Bn, Be, Bd]);

Bned(k,:) = Bned\_k;

%% === Dynamic Attitude (simulate slow rotation) ===

omega\_body = [0.01; 0.01; 0.005]; % rad/s

theta\_att = norm(omega\_body) \* t\_k;

axis = omega\_body / norm(omega\_body);

q = [cos(theta\_att/2);

axis(1)\*sin(theta\_att/2);

axis(2)\*sin(theta\_att/2);

axis(3)\*sin(theta\_att/2)];

R\_ned\_to\_body = quat\_to\_dcm(q);

%% === Simulate Magnetometer ===

B\_true = R\_ned\_to\_body \* Bned\_k';

B\_meas = B\_true + bias + noise\_std \* randn(3,1);

B\_body(k,:) = B\_meas';

end

%% === Plots ===

figure; plot3(r\_eci(:,1), r\_eci(:,2), r\_eci(:,3), 'b'); grid on; axis equal;

xlabel('x [km]'); ylabel('y [km]'); zlabel('z [km]');

title('Satellite Orbit in ECI Frame');

figure; plot(time/60, vecnorm(Bned, 2, 2), 'r'); grid on;

xlabel('Time [min]'); ylabel('|B| [nT]');

title('Earth Magnetic Field Magnitude (NED Frame)');

figure; plot(time/60, B\_body); grid on;

xlabel('Time [min]'); ylabel('Magnetometer Output [nT]');

legend('B\_x','B\_y','B\_z'); title('Simulated Magnetometer Readings (Body Frame)');

%% === Sample Outputs ===

disp('Sample LLA and Magnetic Field Magnitude (first 5):');

disp(array2table([lla(1:5,:), vecnorm(Bned(1:5,:),2,2)], ...

'VariableNames', {'Latitude(deg)', 'Longitude(deg)', 'Altitude(km)', 'Bmag(nT)'}));

disp('Sample Magnetometer Readings (Body Frame, first 5):');

disp(array2table(B\_body(1:5,:), ...

'VariableNames', {'B\_x(nT)', 'B\_y(nT)', 'B\_z(nT)'}));

%% === Supporting Functions ===

function n = mean\_motion(mu, a)

n = sqrt(mu / a^3);

end

function E = solve\_kepler(M, e)

E = M; tol = 1e-8;

for k = 1:100

f = E - e \* sin(E) - M;

fp = 1 - e \* cos(E);

E\_new = E - f / fp;

if abs(E\_new - E) < tol, break; end

E = E\_new;

end

end

function [r\_eci, v\_eci] = coe2rv(a, e, i, RAAN, w, theta, mu)

p = a \* (1 - e^2);

r\_pqw = (p / (1 + e \* cos(theta))) \* [cos(theta); sin(theta); 0];

v\_pqw = sqrt(mu / p) \* [-sin(theta); e + cos(theta); 0];

R3\_W = [cos(-RAAN), -sin(-RAAN), 0; sin(-RAAN), cos(-RAAN), 0; 0, 0, 1];

R1\_i = [1, 0, 0; 0, cos(-i), -sin(-i); 0, sin(-i), cos(-i)];

R3\_w = [cos(-w), -sin(-w), 0; sin(-w), cos(-w), 0; 0, 0, 1];

Q = R3\_W \* R1\_i \* R3\_w;

r\_eci = Q \* r\_pqw;

v\_eci = Q \* v\_pqw;

end

function gst = gstime(datetime\_utc)

JD = juliandate(datetime\_utc);

D = JD - 2451545.0;

T = D / 36525;

gmst = 280.46061837 + 360.98564736629 \* D + 0.000387933 \* T^2 - (T^3) / 38710000;

gst = deg2rad(mod(gmst, 360));

end

function lla = ecef2lla\_custom(r\_ecef, Re, e)

x = r\_ecef(1); y = r\_ecef(2); z = r\_ecef(3);

lon = rad2deg(atan2(y, x));

rho = sqrt(x^2 + y^2);

lat = rad2deg(atan2(z, rho \* (1 - e^2)));

for j = 1:5

N = Re / sqrt(1 - e^2 \* sin(deg2rad(lat))^2);

h = rho / cos(deg2rad(lat)) - N;

lat = rad2deg(atan2(z, rho \* (1 - e^2 \* N / (N + h))));

end

N = Re / sqrt(1 - e^2 \* sin(deg2rad(lat))^2);

h = rho / cos(deg2rad(lat)) - N;

lla = [lat, lon, h];

end

function R = quat\_to\_dcm(q)

q0 = q(1); q1 = q(2); q2 = q(3); q3 = q(4);

R = [1 - 2\*(q2^2 + q3^2), 2\*(q1\*q2 - q0\*q3), 2\*(q1\*q3 + q0\*q2);

2\*(q1\*q2 + q0\*q3), 1 - 2\*(q1^2 + q3^2), 2\*(q2\*q3 - q0\*q1);

2\*(q1\*q3 - q0\*q2), 2\*(q2\*q3 + q0\*q1), 1 - 2\*(q1^2 + q2^2)];

end

**Running this code gıves addıtonal results shown below**

Sample LLA and Magnetic Field Magnitude (first 5):

Latitude(deg) Longitude(deg) Altitude(km) Bmag(nT)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

0 -139.38 499.86 34360

-3.7963 -140.13 499.96 34550

-7.5919 -140.87 500.23 34526

-11.386 -141.63 500.69 34280

-15.178 -142.4 501.32 33811

Sample Magnetometer Readings (Body Frame, first 5):

B\_x(nT) B\_y(nT) B\_z(nT)

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_

24135 24462 12.079

16869 29836 4318.2

13160 28880 13592

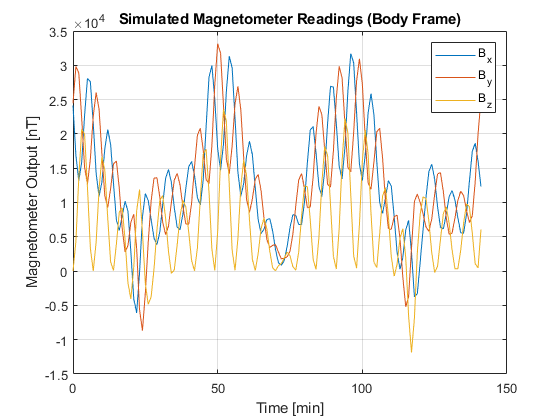
15860 22302 20644

22695 15132 19985

### **Why is Bz (the Z-axis of the magnetometer) almost zero at the start, but bigger later?**

At the beginning of the simulation, when time is zero, the satellite hasn't rotated yet. It is using the Earth's magnetic field exactly as it is, without any change in direction. Therefore, the part of the Earth's magnetic field that points straight down (this is what Bz measures) is very small at that location. So, the magnetometer sees almost no magnetic field in the Z direction. Bx and By show values because Earth’s magnetic field points mostly North and East.

Later in the simulation (time > 0), the satellite starts rotating. When it rotates, the magnetic field from Earth spreads out across the X, Y, and Z directions within the satellite's body. Now the Z-axis starts picking up stronger parts of the Earth's magnetic field. That’s why the Bz value becomes much larger as time goes on.



The ımage above can be explaıned as follows

## **Results Explanatıon**

This plot shows the **3-axis magnetometer measurements** in the **satellite's body frame** over time as it orbits Earth. Each colored curve represents one axis of the magnetic field vector measured by the magnetometer:

🔵 B\_x — Magnetic field along the satellite's body x-axis (North)

🔴 B\_y — Magnetic field along the satellite's body y-axis (East)

🟡 B\_z — Magnetic field along the satellite's body z-axis(Down)

### X-Axis: Time (minutes)

The horizontal axis spans about **150 minutes**, which corresponds to **1.5 orbits** of our satellite, since a circular orbit at ~500 km altitude has a period of ~96 minutes. So we are seeing the **magnetic field environment across one and a half Earth revolutions**.

### Y-Axis: Magnetic Field [nT]

The vertical axis is in **nanoTesla (nT)**, whereas the Earth's magnetic field magnitude typically ranges from **~25,000 to ~65,000 nT** depending on altitude and latitude. In this plot, the field magnitudes mostly range between **~-1.5×10⁴ nT to ~+3.5×10⁴ nT**, which is **expected at orbital altitudes,** and noise and sensor bias are also included, so we are seeing **simulated "realistic" readings**.

### Curve Behavior & Interpretation

#### Oscillations / Variations

The field components **oscillate periodically** due to the Earth’s magnetic field varying with **latitude** and **longitude** and satellite **attitude change** (modeled by quaternion-based body rotation). This causes the magnetic vector in the body frame to shift continuously, even if the satellite follows a stable orbit.

#### B\_x and B\_y Magnitudes Are High

B\_x and B\_y are **dominant**, showing strong changes between 0 and 3.5×10⁴ nT. This suggests that the satellite’s body x and y axes often **align closely with horizontal components** of the magnetic field (North and East).

#### B\_z is Smaller and Periodic

The B\_z component (yellow) is **weaker and more periodic**, possibly peaking around ±1×10⁴ nT  
This is expected because **vertical (Down) components** of Earth’s field are smaller at equatorial latitudes and satellites rarely point straight down

#### Noise & Bias Effects

* Small jitter or fluctuation is due to **sensor noise** (added as Gaussian noise)
* A constant offset (bias) is also present, simulating real hardware imperfections
* These make your simulation more realistic for hardware-in-the-loop or Kalman filter testing

#### Why It Changes Over Time

The satellite:

* Moves through Earth’s **non-uniform magnetic field.**
* Changes its orientation via a **simulated rotating quaternion**.

So, the **magnetic vector thıs cubesat body frame is dynamic**, just like in a real satellite.

Wıth reference to chapter 10 (INTRODUCTION TO ORBITAL PERTURBATIONS ) of the book,

## **Part 1: Code Overview (What It Does)**

### This MATLAB simulation models a **satellite in a Sun-Synchronous Orbit (SSO)** and simulates how a **3-axis magnetometer** would read the **Earth’s magnetic field** along its orbit. It includes:

## **Suggestions to Improve**

### **Add Attitude Simulation**: Use quaternions or Euler angles and rotate Bned accordingly.

### **Add J2 Perturbation**: For more realistic orbit propagation.

### **Data Export**: Write B\_body, lla, etc., to .csv for external analysis.

### **Real Sensor Validation**: If you have flight data, compare against this model.

### 

# Next code

This below matlab code ıs a buıld up of the prevıos codes and performs a comprehensive simulation of a satellite’s orbit propagation, magnetic field measurement, and magnetometer output. It starts by defining Earth’s physical constants and the orbital parameters for a Sun-synchronous orbit, then converts these orbital elements into position and velocity vectors in an Earth-centered inertial frame. The satellite’s position is propagated over time, transformed into Earth-centered Earth-fixed coordinates, and then converted to latitude, longitude, and altitude. At each time step, the Earth’s magnetic field is calculated at the satellite’s location using a geomagnetic model, with a fallback approximation if needed. The magnetic field vector is then rotated into the satellite’s body frame using quaternion-based attitude representation, and sensor biases and noise are added to simulate realistic magnetometer readings. The code visualizes the satellite orbit, magnetic field magnitude, and magnetometer outputs over the simulated period, and provides sample data for position and magnetometer readings. Overall, it simulates the interaction between orbital dynamics, Earth’s magnetic environment, and sensor measurements for a satellite.

**% Full simulation: orbit propagation, magnetic field, and magnetometer**

**clc; clear; close all;**

**%% === Constants ===**

**mu = 398600.4418; % Earth's gravitational parameter [km^3/s^2]**

**Re = 6378.137; % Earth's equatorial radius [km]**

**f = 1/298.257223563; % Earth's flattening**

**e\_earth = sqrt(2\*f - f^2);**

**%% === Orbital Elements for SSO ===**

**a = 6878; % Semi-major axis [km]**

**e = 0; % Eccentricity**

**i = deg2rad(97.4479); % Inclination [rad]**

**RAAN = deg2rad(270); % Right Ascension of Ascending Node [rad]**

**w = deg2rad(0); % Argument of perigee [rad]**

**theta = deg2rad(40); % True anomaly [rad]**

**epoch = datetime(2024,6,2,22,30,0); % Epoch (UTC)**

**%% === Initial State Vectors ===**

**[r0, v0] = coe2rv(a, e, i, RAAN, w, theta, mu);**

**%% === Time Setup ===**

**T = 2\*pi\*sqrt(a^3/mu); % Orbital period [s]**

**dt = 60; % Time step [s]**

**t\_final = 1.5 \* T;**

**time = 0:dt:t\_final;**

**% Preallocate storage arrays**

**r\_eci = zeros(length(time), 3);**

**lla = zeros(length(time), 3); % [lat (deg), lon (deg), alt (km)]**

**Bned = zeros(length(time), 3); % Magnetic field in NED frame [nT]**

**B\_body = zeros(length(time), 3); % Magnetometer readings in body frame [nT]**

**%% === Magnetometer Settings ===**

**bias = [10; -8; 5]; % Sensor bias [nT]**

**noise\_std = 2.0; % Noise standard deviation [nT]**

**%% === Simulation Loop ===**

**for k = 1:length(time)**

**t\_k = time(k);**

**% Orbit propagation**

**M = mean\_motion(mu, a) \* t\_k;**

**E = solve\_kepler(M, e);**

**theta\_k = 2 \* atan2(sqrt(1+e)\*sin(E/2), sqrt(1-e)\*cos(E/2));**

**[r\_k, ~] = coe2rv(a, e, i, RAAN, w, theta\_k, mu);**

**r\_eci(k, :) = r\_k';**

**% Convert to ECEF**

**gst = gstime(epoch + seconds(t\_k));**

**R3 = [cos(gst), sin(gst), 0; -sin(gst), cos(gst), 0; 0, 0, 1];**

**r\_ecef = R3 \* r\_k;**

**% ECEF to LLA**

**lla(k, :) = ecef2lla\_custom(r\_ecef, Re, e\_earth);**

**lat = lla(k, 1); lon = lla(k, 2); alt = lla(k, 3);**

**% Decimal year for magnetic model**

**date\_k = epoch + seconds(t\_k);**

**decimal\_year = year(date\_k) + (day(date\_k, 'dayofyear') - 1)/365.25;**

**decimal\_year = min(max(decimal\_year, 2020), 2025);**

**% Magnetic field calculation with fallback**

**try**

**[Bn, Be, Bd] = wrldmagm(alt \* 1000, lat, lon, decimal\_year, '2020');**

**% Ensure scalars for concatenation**

**if ~isscalar(Bn), Bn = Bn(1); end**

**if ~isscalar(Be), Be = Be(1); end**

**if ~isscalar(Bd), Bd = Bd(1); end**

**catch**

**Bmag = 30000 \* (Re / (Re + alt))^3;**

**Bn = Bmag \* cosd(lat);**

**Be = 0;**

**Bd = -Bmag \* sind(lat);**

**end**

**Bned\_k = double([Bn, Be, Bd]);**

**Bned(k, :) = Bned\_k;**

**% Dynamic rotation using quaternion**

**omega = [0.01; 0.02; 0.015];**

**q = euler\_to\_quat(omega \* t\_k);**

**R\_ned\_to\_body = quat\_to\_dcm(q);**

**B\_true = R\_ned\_to\_body \* Bned\_k';**

**B\_meas = B\_true + bias + noise\_std \* randn(3, 1);**

**B\_body(k, :) = B\_meas';**

**end**

**%% === Plotting ===**

**figure;**

**plot3(r\_eci(:,1), r\_eci(:,2), r\_eci(:,3), 'b');**

**xlabel('x [km]'); ylabel('y [km]'); zlabel('z [km]');**

**title('Satellite Orbit in ECI Frame');**

**grid on; axis equal;**

**figure;**

**Bmag = vecnorm(Bned, 2, 2);**

**plot(time/60, Bmag, 'r');**

**xlabel('Time [min]'); ylabel('|B| [nT]');**

**title('Earth Magnetic Field Magnitude (NED Frame)');**

**grid on;**

**figure;**

**plot(time/60, B\_body);**

**xlabel('Time [min]'); ylabel('Magnetometer Output [nT]');**

**legend('B\_x', 'B\_y', 'B\_z');**

**title('Simulated 3-Axis Magnetometer Readings (Body Frame)');**

**grid on;**

**%% === Display Sample Outputs ===**

**disp('Sample LLA and Magnetic Field Magnitude (first 5):');**

**disp(array2table([lla(1:5,:), Bmag(1:5)], 'VariableNames', {'Latitude(deg)', 'Longitude(deg)', 'Altitude(km)', 'Bmag(nT)'}));**

**disp('Sample Magnetometer Readings (Body Frame, first 5):');**

**disp(array2table(B\_body(1:5,:), 'VariableNames', {'B\_x(nT)', 'B\_y(nT)', 'B\_z(nT)'}));**

**%% === Supporting Functions ===**

**function n = mean\_motion(mu, a)**

**n = sqrt(mu / a^3);**

**end**

**function E = solve\_kepler(M, e)**

**E = M; tol = 1e-8;**

**for k = 1:100**

**f = E - e \* sin(E) - M;**

**fp = 1 - e \* cos(E);**

**E\_new = E - f / fp;**

**if abs(E\_new - E) < tol, break; end**

**E = E\_new;**

**end**

**end**

**function [r\_eci, v\_eci] = coe2rv(a, e, i, RAAN, w, theta, mu)**

**p = a \* (1 - e^2);**

**r\_pqw = (p / (1 + e \* cos(theta))) \* [cos(theta); sin(theta); 0];**

**v\_pqw = sqrt(mu / p) \* [-sin(theta); e + cos(theta); 0];**

**R3\_W = [cos(-RAAN), -sin(-RAAN), 0; sin(-RAAN), cos(-RAAN), 0; 0, 0, 1];**

**R1\_i = [1, 0, 0; 0, cos(-i), -sin(-i); 0, sin(-i), cos(-i)];**

**R3\_w = [cos(-w), -sin(-w), 0; sin(-w), cos(-w), 0; 0, 0, 1];**

**Q = R3\_W \* R1\_i \* R3\_w;**

**r\_eci = Q \* r\_pqw;**

**v\_eci = Q \* v\_pqw;**

**end**

**function gst = gstime(datetime\_utc)**

**JD = juliandate(datetime\_utc);**

**D = JD - 2451545.0;**

**T = D / 36525;**

**gmst = 280.46061837 + 360.98564736629 \* D + 0.000387933 \* T^2 - (T^3) / 38710000;**

**gst = deg2rad(mod(gmst, 360));**

**end**

**function lla = ecef2lla\_custom(r\_ecef, Re, e)**

**x = r\_ecef(1); y = r\_ecef(2); z = r\_ecef(3);**

**lon = rad2deg(atan2(y, x));**

**rho = sqrt(x^2 + y^2);**

**lat = rad2deg(atan2(z, rho \* (1 - e^2)));**

**for j = 1:5**

**N = Re / sqrt(1 - e^2 \* sind(lat)^2);**

**h = rho / cosd(lat) - N;**

**lat = rad2deg(atan2(z, rho \* (1 - e^2 \* N / (N + h))));**

**end**

**N = Re / sqrt(1 - e^2 \* sind(lat)^2);**

**h = rho / cosd(lat) - N;**

**lla = [lat, lon, h];**

**end**

**function q = euler\_to\_quat(euler\_angles)**

**phi = euler\_angles(1); theta = euler\_angles(2); psi = euler\_angles(3);**

**c1 = cos(psi/2); s1 = sin(psi/2);**

**c2 = cos(theta/2); s2 = sin(theta/2);**

**c3 = cos(phi/2); s3 = sin(phi/2);**

**q1 = c1\*c2\*c3 + s1\*s2\*s3;**

**q2 = c1\*c2\*s3 - s1\*s2\*c3;**

**q3 = c1\*s2\*c3 + s1\*c2\*s3;**

**q4 = s1\*c2\*c3 - c1\*s2\*s3;**

**q = [q1; q2; q3; q4];**

**end**

**function R = quat\_to\_dcm(q)**

**q1 = q(1); q2 = q(2); q3 = q(3); q4 = q(4);**

**R = [1 - 2\*(q3^2 + q4^2), 2\*(q2\*q3 - q1\*q4), 2\*(q2\*q4 + q1\*q3);**

**2\*(q2\*q3 + q1\*q4), 1 - 2\*(q2^2 + q4^2), 2\*(q3\*q4 - q1\*q2);**

**2\*(q2\*q4 - q1\*q3), 2\*(q3\*q4 + q1\*q2), 1 - 2\*(q2^2 + q3^2)];**

**end**

**Discussion of the results**