Attitude Control System Toolbox Project Report

ASTRIS
AEROSPACE

Astris Aerospace Inc

By: Khang Nguyen

October 14, 2022

Abstract

Satellites can be used for a variety of purposes; they're changing the way we live and many systems we work with daily. Satellites that do remote sensing, communication, and navigation all have current applications or future research. Regardless of the application, these satellites are able to accurately determine and control their attitude.

The Attitude Control System Toolbox (ACS Toolbox) is an open-source pure Python library developed by professional Guidance, Navigation and Control engineers for satellite Attitude Control System (ACS) design. The algorithms in this library are fundamental for ACS design. The toolbox provides a realistic simulation tool-set, which includes the space environment, time keeping, ephemerides, sensors and actuator selection, and hardware placement.

This document will present the theory that is used to create the ACS toolbox models. An example code will be present at the end of each section to show the ACS toolbox application.

Contents

Chapter 1

Introduction

1.1 Introduction

Satellite Attitude Control System play a fundamental role in every satellite mission.

Chapter 2

Coordinate

2.1 The Earth

The ACS toolbox is primarily for satellites that orbit Earth, as they have certain established parameters for the Earth. There is a set of fundamental parameters that can be used to specify the shape of any planet. These parameters can be used to specify the planet's location, shape, size, and field. The following are the four basic parameters for the Earth.

The first parameter for Earth is its radius. The modern acceptable value for the mean equatorial radius of the Earth ($R_\oplus$) is

$$R_\oplus = 6378.1363\ km$$

The semiminor axis of the Earth's ellipsoid can be used for calculating relative positions on the Earth's surface. The semiminor axis ($b_\oplus$), also called the polar axis, is

$$b_\oplus = 6356.7516005\ km$$

We can also calculate the earth's flattening (f) and eccentricity ($e_\oplus$).

$$f = 0.003352813178 \quad e_\oplus = 0.081819221456$$

The Earth's rotational velocity ($w_\oplus$) is mostly constant. With the aid of precise clocks, observers have found small variations in the value of the Earth's rotational velocity. The the adopted value of Earth's rotational velocity ($w_\oplus$) is

$$w_\oplus = 7.292115 \times 10^{-5}\ \pm 1.5 \times 10^{-12} rad/s$$

2.2 Location Parameter

Latitude and longitude are values that describe a locations on the Earth

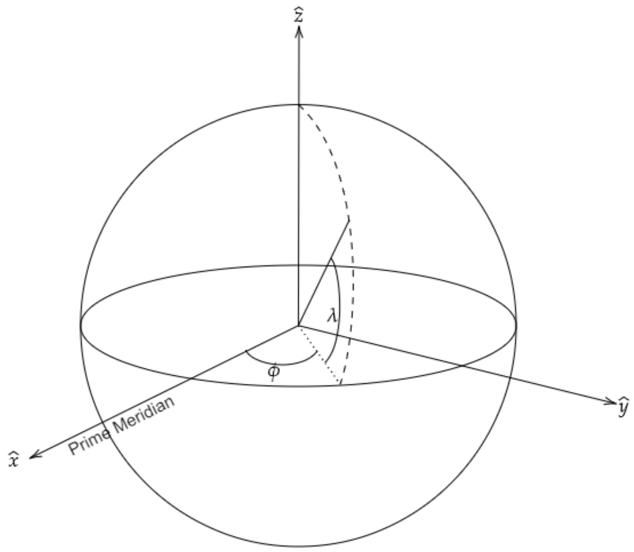

Figure 2-1: Using Latitude and Longitude to measure Location. Terrestrial latitude ϕ and longitude λ reference the Earth's equator. The Greenwich meridian is the zero point for longitude, and the Earth's equator is the reference point for latitude

Longitude is an east-west angular displacement measured positive to the east from a primary meridian. The prime meridian for the Earth is the meridian that the Royal Observatory at Greenwich lies on. Longitude can range from $0°$ to $360°$ when measured from east to west, or from $0°$ to $\pm180°$ when measured from east to west. The Earth's equatorial plane is the reference plane to measure latitude. Latitude is the north-south measurement from this reference plane, with values from $0°$ to $90°$. The latitude is positive in the Northern Hemisphere.

2.3 Earth Shape

Knowledge of the Earth's shape is essential for many applications such as location ground station, remote sensing, geodesy, oceanography, plate tectonics, viewing constraints and many other applications. The spherical Earth is the simplest model and sufficient for many studies, but satellites have shown that the Earth is roughly elliptical. The mean sea level is a good approximation to the ellipsoid model. The mean sea level is called the geoid. A plumb-bob will hang perpendicular to it at every point on earth.

This model does not affect the longitude, but it changes the latitude. There are some common latitudes that we can use in astrodynamics. Geocentric latitude (ϕ_{gc}) is the angle measured at the Earth's centre from the plane of the equator to the point of interest. Most maps use geodetic latitude (g_{gd}) as the angle between the equator and normal to the surface of the ellipsoid. Most maps use geodetic latitude.

We can use the site vector to convert between geocentric latitude and geodetic latitude.

2.4 Site Vector

Site locations, such as those found on standard maps, are usually given in geodetic coordinates. But we require geocentric values for calculations that involve gravitational potential operations. Therefore, we must be able to represent position vectors in geocentric values.

First, we need to determine two auxiliary quantities obtained from the geometrical properties of an ellipse.

$$\mathrm{C}_{\oplus} = \frac{R_{\oplus}}{\sqrt{1 - e_{\oplus}^{2} sin^{2}(\phi_{gd})}}$$

$$\mathrm{S}_{\oplus} = \frac{R_{\oplus}(1 - e_{\oplus})}{\sqrt{1 - e_{\oplus}^{2} sin^{2}(\phi_{gd})}}$$

Where the mean equatorial radius of the Earth $R_{\oplus} = 6378.1363$ km and the Earth's eccentricity $e_{\oplus} = 0.081\ 819\ 221\ 456$. $C_{\oplus}$ is commonly known as the radius of curvature of the meridian.

Next, we can determine the site vector at a location on Earth. We can assume that the height above sea level (h_{ellp}) is equal to the height above the ellipsoid.

$$r_{\vec{IJK}} = \begin{bmatrix} (C_{\oplus} + h_{ellp}) \cdot cos(\phi_{gd}) \cdot cos(\lambda) \\ (C_{\oplus} + h_{ellp}) \cdot cos(\phi_{gd}) \cdot sin(\lambda) \\ (S_{\oplus} + h_{ellp}) \cdot sin(\phi_{gd}) \end{bmatrix}$$

2.4.1 Find Site Vector Using ACStoolbox

We often require the precise location of a sensor or station. For general application, the geocentric values provide satisfactory results. We can use geocentric rectangular coordinates and reference the location of the site from the Earth's center.

Using the ACS toolbox, we can find the site location using the geodetics library. The following example shows how to find station coordinates at Toronto with 43.6532° geodetic latitude and −79.2832° longitude.

```python
"""ellipsoid.py example"""

# ACSToolbox packages.
from acstoolbox.geodetics import ellipsoid

# python library.
import numpy as np
import pytest as pytest

# Evaluate the site position vector at Toronto, Ontario ( Geodetic latitude = 43.6532  N
    , longitude = -79.2832 W, height = 0.076 M
if __name__ == "__main__":
    phi_d = 43.6532
    lamda = -79.2832
    height = 0.076
    # Calculate the site vector
    site_vector = ellipsoid.SitePositionVector3D(phi_d, lamda, height)
    print("Site Vector" = site_vector)
```

Code 2.1: ACS toolbox geodetics library example: Find site vector of Toronto

```
Site Vector = [859.52248524 -4541.5945528 4380.34474178]
```

Code 2.2: Site Vector expected output

The position vector for Toronto is [859.52248524, -4541.5945528, 4380.34474178] KM. We can use the position vector to find the geocentric and geodetic location of a site.

Chapter 3

Time

3.1 Introduction to Time

Time is the fundamental dimension in every branch of science. The main purpose of time is to define with precision the moment of a phenomenon. This moment is referred to as the epoch of the event. Consequently, the epoch is designated as a particular instant described as a date. We need the concept of a precise time interval to determine the epoch of an event. We need to agree on a fundamental epoch. We can determine other epochs by counting the number of intervals from the fundamental epoch.

Time is critical in our application because objects move with high velocity. Four time scales now provide timekeeping for scientific, engineering, and general purposes: sidereal time, solar (universal time), dynamic time, and atomic time. Sidereal time and solar time are based on the Earth's rotation and related to each other by mathematical relationships. Dynamical and atomic time are independent from the other forms.

We determine the period from observations of the apparent diurnal motions of the Sun and stars due to the Earth's rotation. This leads to the common way of dividing time into hours, minutes, and seconds. We can also describe time as an angle because the Earth rotates through one revolution (360°) every day. We can convert between hours (^{h}), minutes (^{m}) and seconds (^{s}) into degrees ($^\circ$), arcminutes ($'$) and arcseconds ($''$).

$$1^h = 60\,minutes(60^m) = 3600\,seconds(3600^s)$$

$$1^\circ = 60\,arcminutes(60') = 3600\,arcseconds(3600'')$$

$$1^h = 15^\circ \qquad 1^\circ = \frac{1^h}{15} = 4^m$$

$$1^m = 15^{'} \qquad 1^{'} = \frac{1^m}{15} = 4^s$$

$$1^s = 15^{''} \qquad 1^{''} = \frac{1^s}{15}$$

The Earth's diurnal rotation with respect to the stars and the Sun gives rise to the concepts of sidereal time and solar time, respectively. The lengths of these days differ mainly because of the Earth's annual orbital motion about the Sun. When compared to the sidereal day, Earth rotates nearly one degree more per solar day. This causes the sidereal day to be about four minutes shorter than the solar day.

3.2 Sidereal time

The Earth's diurnal rotation with respect to the stars gives rise to the concepts of sidereal time. We define it by the successive transits of the local meridian by a given star. Because the stars are several orders of magnitude farther away than the Sun, their location would not change significantly even during a year.

We define sidereal time as the hour angle of the vernal equinox relative to the local meridian. Because the vernal equinox is the reference point, the sidereal time associated with the Greenwich meridian is called Greenwich Mean Sidereal Time (θ_{GMST} or GMST). The sidereal time at a particular longitude is called Local Sidereal Time (θ_{LST} or LST)

We can convert GMST to LST at a particular longitude (λ) using this equation,

$$\theta_{LST} = \theta_{GMST} + \lambda$$

3.3 Solar Time

The Sun governs our daily activity, and because its motion is regular, we base solar time on that motion. Solar time is defined by successive transits of the sun over a local meridian. The Sun's apparent motion results from a combination of the Earth's rotation around its axis and its motion around the Sun. The Royal Observatory at Greenwich, England was adopted as the 0° longitude point. Earth rotates more than 360.986° during one solar day because of its motion around the sun.

According to Kepler's first law, Earth's orbit is slightly elliptical. This leads to the time intervals between successive transits of the sun not being identical. The Sun occasionally lags or leads the mean value of its position in the sky. This gives the concepts of apparent solar time and mean solar time.

3.3.1 Apparent Solar Time

The Earth's orbit about the Sun has a small eccentricity, causing the length of each day to differ by a small amount. Apparent solar time is the interval between successive transits which we observe from a particular longitude:

$$local\,apparent\,solar\,time = LHA_{\odot} + 12^h$$

Where LHA is the Sun Local Hour Angle for a local observer, this is the time equal to the elapsed time from when the Sun was overhead. The 12-hour increment ensures that 0^h corresponds to midnight and 12^h to noon.

The Greenwich apparent solar time is similar using the Greenwich Hour Angle (GHA) and the right ascension of the Sun ($\alpha_{\odot}$),

$$Greenwich\,apparent\,solar\,time = GHA_{\odot} - \alpha_{\odot} + 12^h$$

3.3.2 Mean Solar Time

Mean solar time is derived from measurements of the Earth orientation. These measurements allow Greenwich Mean Solar Time to be found, and then mean solar time is calculated through a numerical formula. The mean solar time at Greenwich as universal time (UT) and the difference between apparent and mean solar time as the equation of time (EQ_{time}),

$$\begin{aligned} EQ_{time} = & -1.91466471^{\circ} sin(M_{\odot}) - 0.019994643 sin(2M_{\odot}) \\ & + 2.466 sin(\lambda_{ecliptic}) - 0.0053 sin(4\lambda_{ecliptic}) \end{aligned}$$

The difference in apparent and mean solar time varies from -14 to +16 minutes.

Some irregularities in the Sun's apparent motion make it difficult to use for reckoning time. This lead to the concept of universal time (UT), and there are three variations of UT (UT0, UT1 and UT2). We must distinguish between UT0, UT1 and UT2 for precise application. The different between them are small.

UT1 is found by reducing the observations of stars from many ground stations. UT1 depends on that apparent diurnal motion of the stars. The practice is to measure the locations of the radio galaxies and apply that information to determine solar time. For an observer at a known longitude λ, UT0 is sometimes calculated as 12 hours plus the Greenwich Hour Angle, GHA

$$UT0 = 12^h + GHA_{\odot} = 12^h + LHA_{\odot} - \lambda$$

UT1 is the correction of UT0 for polar motion so time is independent of station location. The correction can be calculate using the following equation,

$$UT1 = UT0 - (x_p sin(\lambda) + y_p cos(\lambda))tan(\phi_{gc})$$

The longitude, λ, and the x_p and y_p angle describe the instantaneous positions of the pole. The geocentric latitude (ϕ_{gc}) is the latitude of the observing site. The difference between UT1 and UT0 is about 30 milliseconds (0.030^s).

UT1 can be corrected for seasonal variations to yield UT2. UT2 is considered obsolete.

3.4 Coordinated Universal Time (UTC)

The most commonly used time system is Coordinated Universal Time (UTC), which is derived from atomic time. It is designed to follow within $\pm 0.9^s$ ($\Delta UT1 = UT1 - UTC$). UTC is the time zone that we commonly use in our everyday lives. Because UT1 varies irregularly due to variations in the Earth's rotation, we must periodically insert leap seconds into UTC to keep the two time scales in close agreement. This minimizes navigation errors when using UTC = UT1 and maximizes the time between leap second insertions. UTC always differs by an integer number of seconds from TAI. UTC and a $\Delta UT1$ estimate have been broadcast on WWV (WWV) since January 1, 1972.

We define a time zone for a particular region. The calculations will be more accurate because they will define the time for a specific longitude and not just a region. Many countries use Daylight Saving Time, but UTC does not change.

3.5 Julian Date

The Julian date (JD) is the interval of time measured in days from the epoch January 1, 4713 B.C., 12:00. JD is an essential concept in astrodynamics. Many relations for astronomical equations of motion use the number of Julian centuries from a particular epoch.

To find the Julian date from a known date and time within the period March 1, 1900 to February 28, 2100, we can use the following algorithm,

$$JD = 367(yr) - INT(\frac{7(yr+INT(\frac{mo+9}{12}))}{4}) + INT(\frac{275mo}{9}) + d + 172013.5 + \frac{\frac{\frac{s}{60}+min}{60}+h}{24}$$

The International Astronomical Union (IAU) recommends using a Modified Julian Date (MJD) to reduce the number of decimal places in the Julian Date. The MJD can be calculated from the JD using this equation,

$$MJD = JD - 2400000.5$$

This adjustment reduces the size of the date to about two significant digits. It also begins each day at midnight instead of noon.

Julian Centuries are often used as the time scale, and a shorthand notation such as J1900 or J2000 is common. Some commonly used epochs are,

$$J2000 = 2451545.0 = January\, 1^{st}, 2000\, 12:00\, TT$$

$$J1900 = 2414021.0 = January\, 1^{st}, 1900\, 12:00\, UT1$$

3.5.1 Find Julian Date using ACS Toolbox

Julian Date provides a continuous, simple, and concise method of preserving year-month-day-hour-minute-second information in one variable. This is good for computer applications. Therefore we want to convert Gregorian date UTC to Julian Date.

We can convert a known date with known year, month, day, hour, minute and second to Julian Date using ACS toolbox. The following example show how to convert January 1, 2000 at 12:00:00 to Julian Date using the ACS Toolbox's time library

```python
# ACS Toolbox library
from acstoolbox.time.clock import Clock

# Python library
import numpy as np

if __name__ == "__main__":
    # J2000 (January 1, 2000 12hh00mm00ss).
    gregorian_j2000 = [2000, 1, 1, 12, 0, 0]

    # Initialize clock and convert Gregorian date.
    clock = Clock()
```

```python
    print("Julian Date: ", clock.GregorianToJulianDate(gregorian_j2000))
```

Code 3.1: ACS toolbox's time library example: convert January 1

```
Julian Date:  2451545.0
```

Code 3.2: Julian Date expected output

January 1, 2000 at 12:00:00 is 2451545.0 in JD, this is also the J2000 epoch.

There are many other time conversion can be done using the ACS toolbox.

3.6 Atomic Time

We need a highly accurate time system which is independent of the average rotation of the Earth. Atomic time is the most precise time standard. The International Atomic Time (TAI) based on the specific quantum transition of electrons in a cesium-133 atom. The transition causes the emission of photons of a known frequency that we can count. We define the atomic second by a fixed number of cycles. One SI second equals the duration of 9,192,631,770 periods of the wavelength associated with the radiation emitted by the electron transition between two hyper fine levels of the ground state of cesium-133.

3.6.1 Find TAI Julian Date using ACS Toolbox

For precise time applications such as communication systems, we need to know the atomic time at a certain Gregorian UTC time.

We can find the TAI Julian Date from UTC Gregorian using the ACS Toolbox. The following example show how to convert March 24, 2022 12:02:00 UTC Gregorian.

```python
# ACS Toolbox library
from acstoolbox.time.clock import Clock

# Python library
import numpy as np

if __name__ == "__main__":
    # Initialize clock and set the epoch to March 24, 2022 12hh02mm00ss.
    clock = Clock()
    epoch_gregorian_utc = [2022, 3, 24, 12, 1, 60]

    print("TAI Julian Date:", clock.UTCGregoriantoTAIJD(
        epoch_gregorian_utc
    ))
```

Code 3.3: ACS toolbox's time library: convert January 1

```
TAI Julian Date: 2459663.0018171296
```

Code 3.4: Atomic Time expected output

January 1, 2000 at 12:02:00 in UTC Gregorian is 2459663.0018171296 days in TAI Julian date

3.7 Dynamical Time

We can also measure time by the motion of bodies, such as the Earth's motion around the sun, or the Moon's motion around the Earth. This is dynamical time, so time is the independent variable in the equation that describes an object's motion. The ephemeris time (ET) was created to provide a more stable time reference than those based on the Earth's variable rotation. But the relativistic effects were significant. Many astronomical equations of motion reference the barycenter of the solar system. Terrestrial Dynamical Time (TDT) and barycentric dynamic time (TDB) were created to replace ET and link to the barycentric-referenced equations of motion.

3.7.1 Terrestrial time (TT)

Terrestrial time (TT) is another name for TDT. Terrestrial Time is the theoretical timescale of apparent geocentric ephemerides of bodies in the solar system. TT is independent of equation of motion theories and uses the SI second as the fundamental interval. It is related to other times,

$$UTC = UT1 - \Delta UT1$$

$$TAI = UTC + \Delta AT$$

$$TT = TAI + 32.184^s$$

The Terrestrial Time (TT) standard is a continuous running time scale, unaffected by the irregularities of the Earth's rotation and orbit around the Sun. One day in TT has precisely 86400 SI seconds.

Find Terrestrial Time Using ACS Toolbox

We use Terrestrial Time primarily for time-measurements of astronomical observations made from the surface of the Earth. Knowing the Terrestrial Time from a UTC Gregorian time is required for astronomical applications.

We can find the Terrestrial Time Using ACS Toolbox's time library. The following example shows the conversion from March 24, 2022 12:02:00 UTC Gregorian to TT.

```python
# ACS Toolbox library
from acstoolbox.time.clock import Clock

# Python library
import numpy as np

if __name__ == "__main__":
    # Initialize clock and set the epoch to March 24, 2022 12hh02mm00ss.

    clock = Clock()
    epoch_gregorian_utc = [2022, 3, 24, 12, 1, 60]

    TAI = clock.UTCGregoriantoTAIJD(epoch_gregorian_utc)
    TT = clock.TAIstoTTs(TAI)
    print("Terrestrial Time: ", TT)
```

Code 3.5: ACS toolbox convert January 1

```
Terrestrial Time:  2459695.1858171294
```

Code 3.6: ACS toolbox convert January 1

We have to convert UTC Gregorian to TAI then to TT. At 12:02:00 on January 1, 2000, the time is 2459695.1858171294 in Terrestrial Time.

3.7.2 Barycentric dynamical time (TDB)

Barycentric dynamical time (TDB) is defined as the independent variable of equations of motion with respect to the barycenter of the solar system. TDB includes relativistic effects and requires TT, UT1, gravitational constants and other parameters. We can convert between barycentric and terrestrial time using the following equation,

$$TDB = TT + \frac{2r_s}{a_e n_e} e_e sin(E) + other$$

Using the Schwarzs child radius ($R_s = 1.478km$), the Earth's semimajor axis about the Sun ($a_e = 149598023km$), the Earth's mean motion about the Sun ($n_e = 1.991 \times 10^{-7} rad/s$) and the eccentricity of the Earth's orbit ($e_e = 0.016708617$). "Other" includes small effects contributed by the Moon and planets, as well as the Earth's diurnal motion. The Earth's eccentric anomaly (E) is obtained from the mean anomaly, and we can compute it using this equation,

$$E \cong M_{\oplus} + E_e sin(M_{\oplus}) + \frac{e_e^2}{2} sin(2M_{\oplus})$$

Substituting everything in the Barycentric dynamical time equation, we get

$$TDB \cong TT + 0.001658^s sin(M_{\oplus}) + 0.00001385 sin(2M_{\oplus})$$

$$M_{\oplus} \cong 357.5277233^{\circ} + 35999.0.5034 T_{TT}$$

Chapter 4

Sun

4.1 Solar Phenomenal

We require position vectors for the Sun to analyse perturbation forces on satellites, solar-panel illumination and remote sensing. Sunrise or sunset conditions are needed when designing a mission for sensor viewing.

Very precise ephemerides of the Sun with respect to the Earth are available through the Jet Propulsion Laboratory (JPL). It's often convenient to place a mathematical algorithm onboard a satellite using a less precise formula from the Astronomical Almanac. Using equations used in the reduction of coordinates, we can produce methods to determine the local time of sunrise and sunset.

4.2 Sun Position Vector

The determination of the sun's position vector is particularly useful to satellites for power generation analysis (using solar cells) and attitude determination and control subsystems that use sun sensors. The Astronomical Almanac lists an algorithm which evaluates the vector from the Earth to the Sun in a Mean-Equator of Date (MOD) frame with $0.01°$ accuracy and valid from 1950 to 2050. The diagram on the next page shows the geometry necessary to visualise the problem.

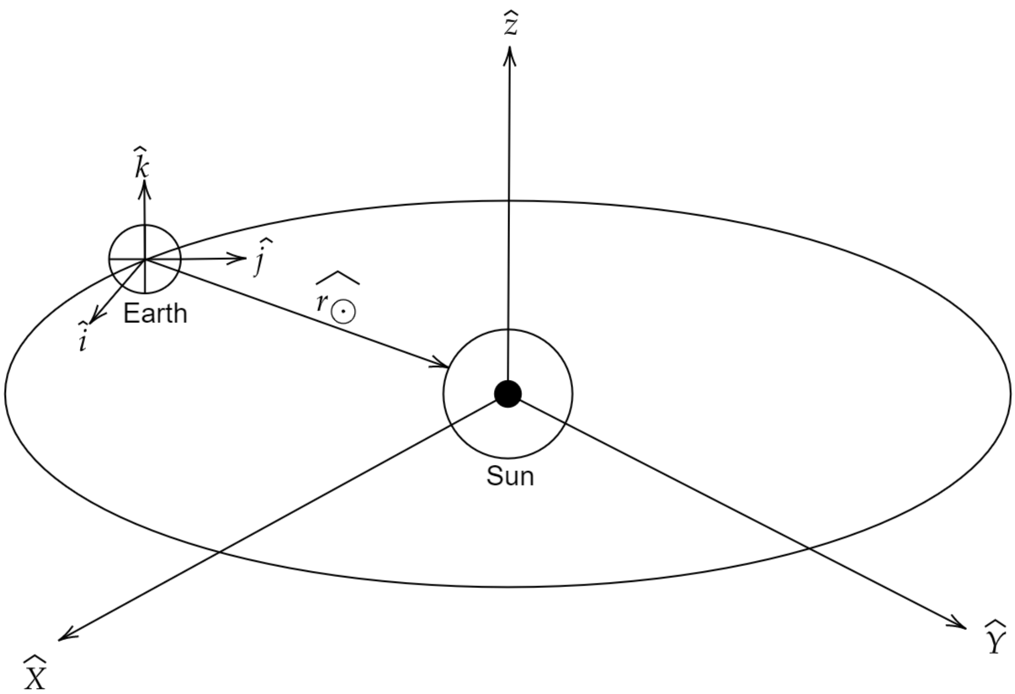

Figure 4-1: Geometry for the sun position vector.The position vector of the Sun rests on determining the ecliptic longitude (ecliptic longitude is 0°) and the range.

To find the the position vector of the Sun, first we need to find the mean longitude of the Sun ($\lambda_{M_\bigodot}$), it is calculated in the MOD frame using the Julian Centuries in UT1 (T_{ut1}):

$$\lambda_{M_\bigodot} = 280.4606184° + 36000.77005362° T_{UT1}$$

Next the mean anomaly for the Sun $M_\odot$ is found using T_{ut1}:

$$M_\circ = 357.5277233° + 35999.05034 T_{UT1}$$

The ecliptic latitude (ϕ_ξ) and longitude (λ_ξ) of the Sun are determined with the aforementioned quantities. The ecliptic latitude is often approximated as 0°, while the ecliptic longitude can be expressed as:

$$\lambda_{ecliptic} = \lambda_{M_\bigodot} + 1.914666471° sin(M_\bigodot) + 0.019994643 sin(2M_\bigodot)$$

Before expressing the Sun's coordinates with respect to the Earth, we also determine the obliquity of the ecliptic:

$$\epsilon = 23.439291^\circ - 0.0130042T_{UT1}$$

We can find the position magnitude using values of the Earth:

$$r_\odot = 1.000140612 - 0.016708617cos(M_\odot) - 0.000139589cos(2M_\odot)$$

Finally, we can combine all of the above to evaluate the Sun's position in the MOD frame:

$$\vec{r_\odot} = \begin{bmatrix} r_\odot cos(\lambda_{ecliptic}) \\ r_\odot cos(\epsilon) sin(\lambda_{ecliptic}) \\ r_\odot sin(\epsilon) sin(_{ecliptic}) \end{bmatrix}$$

We can use third-party software such as Systems Tool Kit (STK) from AGI to validate the ACStoolbox model Sun vector. STK model is validated with extremely high accuracy.

To find the Sun vector using the STK software, we must add a facility object. Then we need to find the Sun Vector in the ICRF, which is a realization of the ICRS (system). The following graphs show the result of the ACStoolbox model with AGI's Systems Tool Kit. We will model the Sun Vector in 2018 using STK to check with ACStoolbox's model. The two value are very similar.

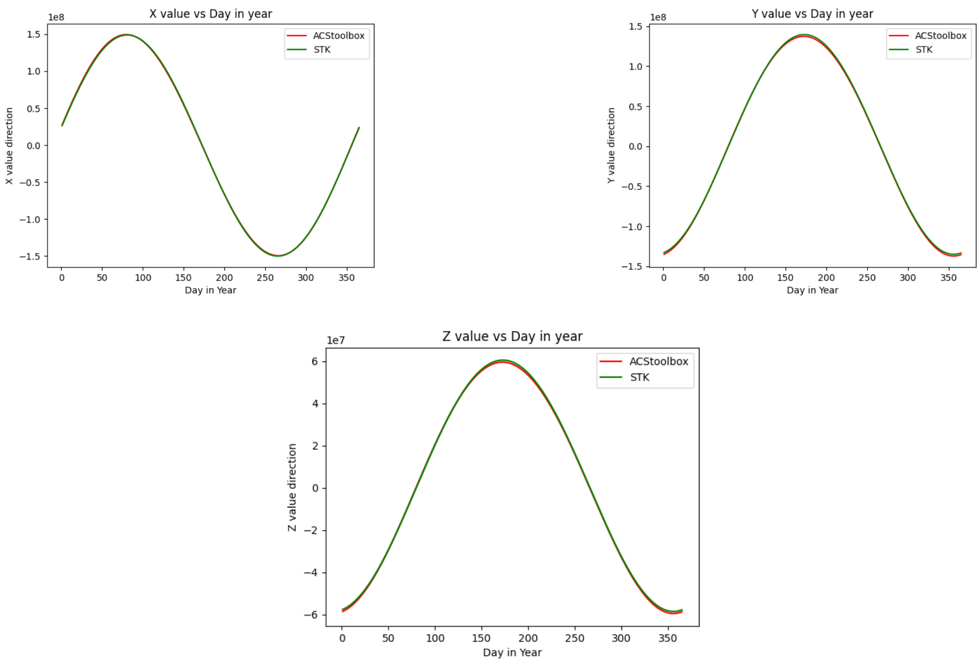

Figure 4.1: Comparing the ACStoolbox's Sun Vector Values to the STK Sun vector everyday in 2018

We can see the angle different between the STK Sun Vector and the ACStoolbox's Sun Vector.

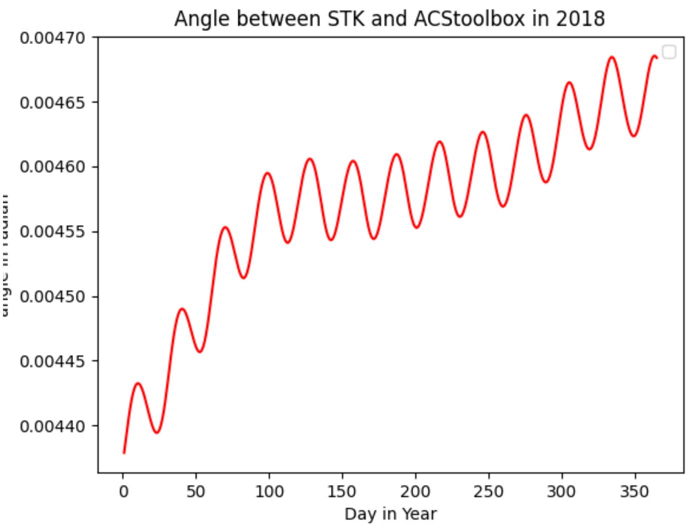

Figure 4.2: The angle difference between STK Sun Vector and ACStoolbox Sun Vector

4.3 Find Sun Position Vector using ACS toolbox

we need to know the Sun Position vector at a certain time for a lot of application such as sun tracking for solar panel.

We can find the Sun Position Vector using the ACStoolbox's emphemeris library. The following example will show the sun vector in the MOD frame on April 2^{nd}, 2006 at 0:00h UTC.

```
from acstoolbox.time.clock import Clock
from acstoolbox.ephemerides.sun import Sun


# Evaluate unit sun vector from UTC Gregorian date.
if __name__ == "__main__":
    clock = Clock()
    sun = Sun(clock)

    # Evaluate the sun vector in the MOD frame on 2 April 2006 00hh00mm00ss.
    epoch_gregorian_utc = [2006, 4, 2, 0, 0, 0]
    s_mod = sun.GetUnitMODPositionFromUTC(epoch_gregorian_utc)

    print("Sun Vector: ", s_mod)
```

Code 4.1: ACS Toolbox's emphemeris libary: Sun Position Vector on April 2^{nd}

```
Sun Vector: [0.9776782, 0.1911521, 0.0828717]
```

Code 4.2: Sun Position Vector expected output

The Sun Position vector on April 2 at 0:00h UTC is [0.9776782, 0.1911521, 0.0828717].

Chapter 5

Moon

5.1 Lunar Phenomena

Knowledge of the Moon's location and Illumination is required in many application. Lunar illumination is used to determine optimum observation times. Optical tracking device use software and hardware that prevent direct observation of the Moon.

5.2 Moon Position Vector

We often need to know the position vector from the Earth to the Moon. The ephemerides of the Jet Propulsion Laboratory (JPL) are the most accurate, but the Astronomical Almanac provided us with a less precise formula. Higher precision theory will show the complexity of the Moon motion. The following diagram is good to visualize the transformation.

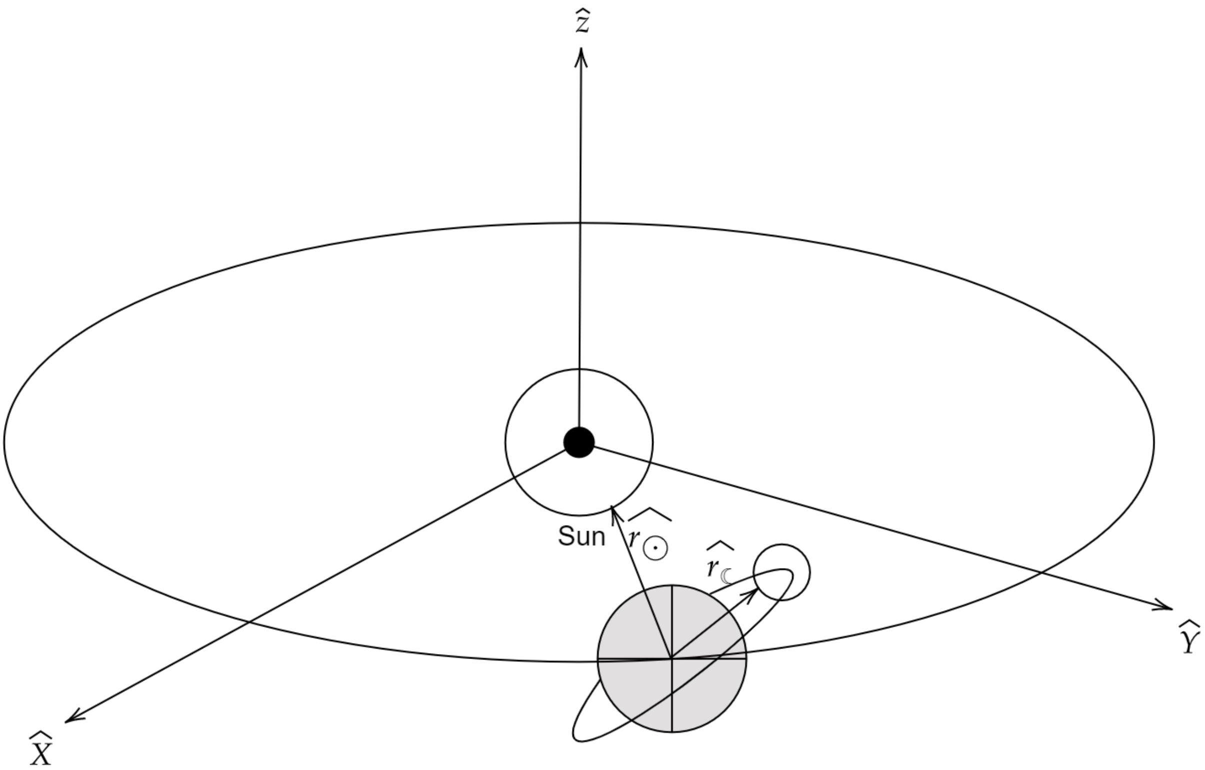

Figure 5-1: Geometry for the Moon position Vector. Notice the ecliptic latitude is not zero for the Moon. The ecliptic coordinates determine the position vector.

To find the Moon position vector, first, we need to determine the number of centuries elapsed from epoch J2000 using Julian Date in Barycentric dynamic time ($\mathcal{J}_{tdb}$)

$$\mathrm{T_{tdb}} = \frac{\mathcal{J}_{tdb} - 2\,451\,545.0}{36\,525}$$

Next, we need to find an expression for the mean anomalies ($M_{☾}$)

$$\mathrm{M}_{☾} = 134.9 + 477\,198.85 \cdot T_{tdb}$$

The mean argument of latitude ($u_{M_{☾}}$) can be found using (T_{tdb})

$$u_{M_{☾}} = 93.3 + 483\,202.03 \cdot T_{tdb}$$

The elongation ($D_{\odot}$) can also be found using T_{tdb}:

$$D_{\odot} = 117.85 + 445\,267.115 \cdot T_{tdb}$$

As with the sun, we can find the ecliptic longitude (λ_ξ), ecliptic latitude (ϕ_ξ) and the parallax (ϑ) using the following equations.

$$\begin{aligned}\lambda_\xi = \lambda_☾ + 6.29 \cdot sin(M_☾) - 1.27 \cdot \sin\left(M_☾ - 2D_\odot\right) + 0.66 \cdot sin(2D_\odot) \\ +0.21 \cdot sin(2M_☾) - 0.19 \cdot sin(M_\odot) - 0.11 \cdot sin(u_{M_☾})\end{aligned}$$

The ecliptic latitude (ϕ_ξ) isn't zero, so we can find it as

$$\begin{aligned}\phi_\xi = 5.13 \cdot sin(u_{M_☾}) + 0.28 \cdot sin(M_☾ + u_{M_☾}) - 0.28 \cdot sin(u_{M_☾} - M_☾) \\ -0.17 \cdot sin(u_{M_☾} - 2D_\odot)\end{aligned}$$

The horizontal parallax (ϑ) is

$$\begin{aligned}\vartheta = 0.9508 + 0.0518 \cdot cos(M_☾) + 0.0095 \cdot cos(M_☾ - 2D_\odot) \\ +0.0078 \cdot cos(2D_\odot) + 0.0028 \cdot cos(2M_☾)\end{aligned}$$

The obliquity of the ecliptic can be calculated using T_{tdb}

$$\epsilon = 23.439291 - 0.0130042 \cdot T_{tdb}$$

The next step is to find the magnitude of the moon position vector.

$$\mathrm{r}_☾ = \frac{1}{sin(\vartheta)}$$

Finally, we can combine all of the above to evaluate the Moon Position Vector expressed in the mean frame:

$$\mathbf{r}_\phi = r_☾ \cdot \begin{bmatrix} cos(\phi_\xi)cos(\lambda_\xi) \\ cos(\epsilon)cos(\phi_\xi)cos(\lambda_\xi) - sin(\epsilon)sin(\phi_\xi) \\ sin(\epsilon)cos(\phi_\xi)sin(\lambda_\xi) + cos(\epsilon)sin(\phi_\xi) \end{bmatrix}$$

Similar to validating th Sun Vector, we can use third-party software such as Systems Tool Kit (STK) from AGI to validate ACStoolbox model.

To find the moon vector using the STK software, you must add a facility object. You then need to find the Moon Vector in the ICRF, which is a realization of the ICRS (system). The following graphs show the result of the ACStoolbox model with AGI's Systems Tool Kit. We will model the Moon Vector in 2018 using STK to check with ACStoolbox's model. The two value are very similar.

We can see the angle different between the STK Moon Vector and the ACStoolbox's Moon Vector.

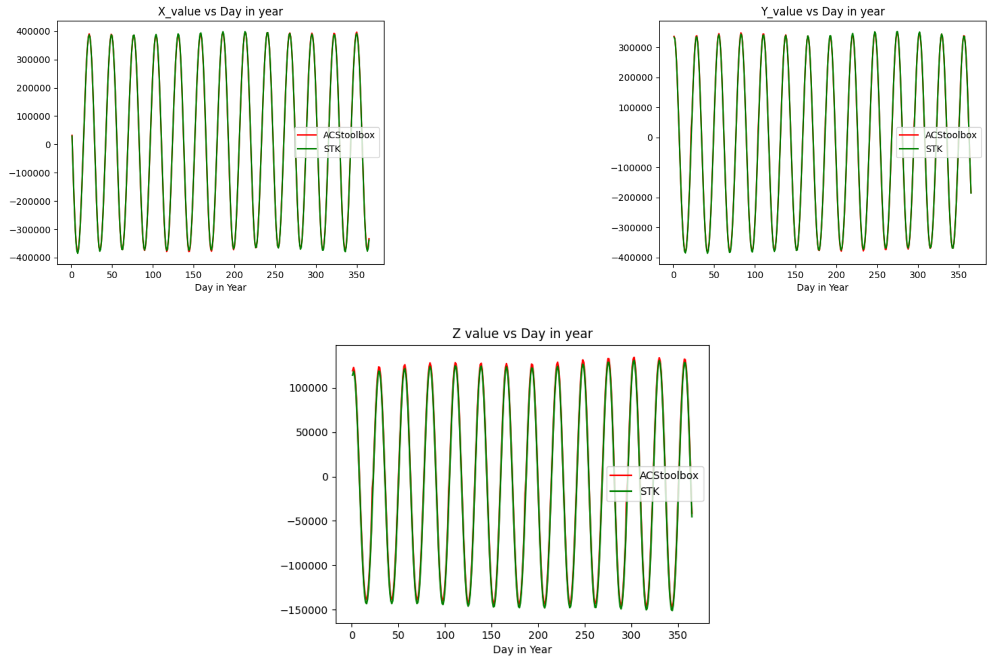

Figure 5.1: Comparing the ACStoolbox's Moon Vector Values to the STK moon vector everyday in 2018

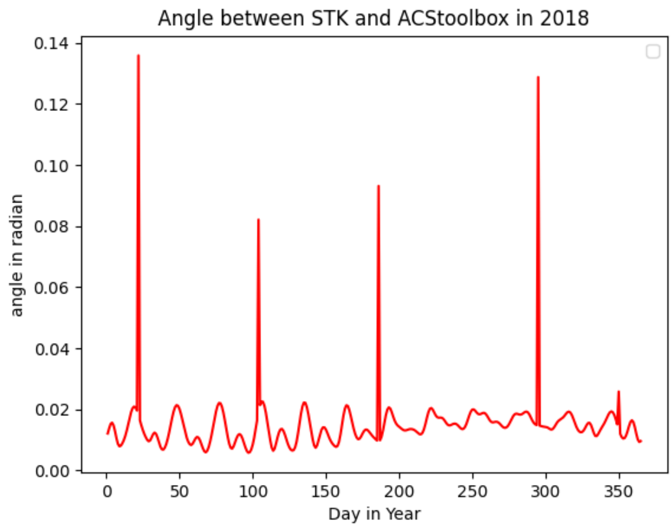

Figure 5.2: The angle difference between STK Moon Vector and ACStoolbox Moon Vector

5.2.1 Find Moon Vector using ACS Toolbox

Moon Position Vector is needed for many optical tracking hardware application. We can find the Moon Position Vector using the ephemerides library in the ACS toolbox using Coordinated Universal Time (UTC). The ephemerides library can give the Moon Position vector in Earth Radius(ER), Kilometer(KM), and normalize moon vector.

The following example show how to find the Moon Position Vector from April 28, 1994, 0:00 UTC in Kilometer(KM).

```python
# ACSToolbox packages.
from acstoolbox.time.clock import Clock
from acstoolbox.ephemerides.moon import Moon

# Standard packages.
import numpy as np
import pytest as pytest

# Evaluate moon vector from UTC in KM
if __name__ == "__main__":
    clock = Clock()
    moon = Moon(clock)

    # Evaluate the moon vector from April 28, 1994, 0:00 UTC
    gregorian_utc = [1994, 4, 28, 0, 0, 0]
    m_vector = moon.VectorFromUTCinKM(gregorian_utc)
    print("Moon Position Vector = ", m_vector)
```

Code 5.1: ACS toolbox's ephemerides library example: Find the Moon Position Vector.

```
Moon Position Vector = [-134241.192, -311571.349, -126693.681]
```

Code 5.2: example output

The Moon Position vector on April 28, 1994, 0:00 UTC is [-134241.192, -311571.349, -126693.681] KM.

5.3 Phases of the Moon

The percentage of the Moon's surface that reflects light back to Earth is constantly change. There are four phases of the Moon (new, full, last, and first quarter phases). The Moon's phase can be calculated using the Sun's and Moon's ecliptic longitudes.

$$\text{phase}_{☾} = \lambda_{ecliptic_{\odot}} - \lambda_{ecliptic_{☾}}$$

The values for each phase are:

- 0°: new moon - 90°: first quarter - 180°: full - 270°: last quarter

The phase is close to the angle between the Sun and the Moon. The percentage of the Moon's surface that is illuminated can be approximated as

$$\%disk = \frac{100\%}{2} \cdot (1 - cos(phase_{☾}))$$

The Earth blocks solar light from reaching the Moon (lunar eclipse) does occur on occasion. The lunar phases are due to the relative position of the Moon with respect to an Earth observer.

The phases of the Moon also have other names. Waxing and waning Moons describe if the apparent Moon illumination is increasing or decreasing, respectively. Gibbous is when more than 50 % of the Moon appears illuminated.

Each season (winter, etc.) has 3 full Moons. The blue moon occurs whenever there are 4 full Moons in a season, the third full moon is the blue Moon.

5.3.1 Find Percent of Moon's Surface Illuminated using ACS Toolbox

We can find the percent of the Moon's surface is illuminated using the ACS toolbox's ephemeris library. The following example show the percents of Moon surface that is Illuminated on April 3, 2020 at 0:00 UTC

```python

from acstoolbox.time.clock import Clock
from acstoolbox.ephemerides.moon import Moon


if __name__ == "__main__":
    clock = Clock()
    moon = Moon(clock)
    # Evaluate the moon phase on April 3, 2020 at 0:00 UTC
    gregorian_utc = [2020, 4, 3, 0, 0, 0]
    phase = moon.PercentageMoonIlluminated(gregorian_utc)
    print("Surface of the moon that is illuminated: "phase)
```

Code 5.3: ACS toolbox Moon surface that is Illuminated example

```
Surface of the moon that is illuminated: 66.7277
```

Code 5.4: Moon surface that is Illuminated expected output

The expected Moon's surface that is illuminated on April 3, 2020 at 0:00 UTC is 66.7277%

Chapter 6

International Geomagnetic Reference Field (IGRF)

6.1 IGRF model

The Spherical Harmonic model presents some computational aspects of the geomagnetic field models. The Earth's magnetic field (B), can be represented as the gradient of a scalar potential function (V),

$$B = -\nabla V$$

V can be represented by a series of spherical harmonics,

$$V(R,\theta,\phi) = a\sum_{n=1}^{k}(\frac{a}{r})^{(n-1)}\sum_{m=0}^{k}(g_m^n cos(m\phi) + h_m^n sin(m\phi))P_n^m(\theta)$$

Where a is the Earth's equatorial radius (6371.2 km), g_n^m and h_n^m are Gaussian coefficients, and R, θ, ϕ are geodetic coefficients describing location on Earth.

The Gaussian coefficients are determined empirically by a least-squares fit to measurements of the field. A set of these coefficients constitutes a model of the field. The International Association of Geomagnetism and Aeronomy (IAGA) releases the Generation International Geomagnetic Reference Field every 5 years and can be used for the next 5 years.

The Legendre function (P_n^m) is related to the Schmidt functions ($S_{n,m}$) and the Gauss function ($P^{n,m}$) as follow,

$$P_n^m = S_{n,m}P^{n,m}$$

The Schimdt factors are best combined with the Gaussian coefficients because they are independent of R, θ, ϕ and must be calculated only once during a computer run. Thus, we define

$$g^{n,m} = S_{n,m} g_n^m$$
$$h^{n,m} = S_{n,m} h_n^m$$

The following recursion relation can be used to calculate the Schimdt factors ($S_{n,m}$):

$$S_{0,0} = 1$$
$$S_{n,0} = S_{n-1,0}[\frac{2n-1}{n}] \qquad n \geq 1$$
$$S_{n,m} = S_{n,m-1}\sqrt{\frac{(n-m+1)(\delta_m^1+1)}{n+m}} \qquad m \geq 1$$

where the Kronecker delta, $\delta_j^i = 1$ is $i = j$ and 0 otherwise.

The $P^{n,m}$ can be similarly obtained from the following recursion relations:

$$P^{0,0} = 1$$
$$P^{n,n} = sin(\theta)P^{n-1,n-1}$$
$$P^{n,m} = cos(\theta)P^{n-1,m} - K^{n,m}P^{n-2,m}$$

where

$$K^{n,m} = \frac{(n-1)^2 - m^2}{(2n-1)(2n-3)}$$
$$K^{n,m} = 0$$

The gradient in the magnetic field equation leads to partial derivatives of the $P^{n,m}$. We need

$$\frac{\partial P^{0,0}}{\partial\theta} = 0$$
$$\frac{\partial P^{n,n}}{\partial\theta} = sin(\theta)\frac{\partial P^{n-1,n-1}}{\partial\theta} + cos(\theta)P^{n-1,n-1} \qquad n \geq 1$$
$$\frac{\partial P^{n,m}}{\partial\theta} = cos(\theta)\frac{\partial P^{n-1,m}}{\partial\theta} - sin(\theta)P^{n-1,m} - K^{n,m}\frac{\partial P^{n-2,m}}{\partial\theta}$$

We can calculate the magnetic field (**B**) from the gradient of V using the coefficients $g^{n,m}$ and $h^{n,m}$, and the recursion relation. Specifically,

$$B_r = \frac{-\partial V}{\partial r} = \sum_{n=1}^{k}(\frac{a}{r})^{(n+2)}(n+1)\sum_{m=0}^{n}(g^{n,m}cos(m\phi) + h^{n,m}sin(m\phi))P^{n,m}(\theta)$$

$$B_\theta = \frac{-1}{r}\frac{\partial V}{\partial \theta} = -\sum_{n=1}^{k}(\frac{a}{r})^{(n+2)}\sum_{m=0}^{n}(g^{n,m}cos(m\phi) + h^{n,m}sin(m\phi))\frac{\partial P^{n,m}(\theta)}{\partial \theta}$$

$$B_\phi = \frac{-1}{rsin(\theta)}\frac{\partial V}{\partial \theta} = \frac{-1}{sin(\theta)}\sum_{n=1}^{k}(\frac{a}{r})^{(n+2)}\sum_{m=0}^{n}(-g^{n,m}cos(m\phi)+h^{n,m}sin(m\phi))P^{n,m}(\theta)$$

6.1.1 Find the Earth Magnetic Field at a location Using ACS Toolbox

Appendix A

Appendix A

[1] D. A. Vallado and W. D. McClain, Fundamentals of astrodynamics and applications, 2nd ed. Dordrecht;: Microcosm Press, 2001.