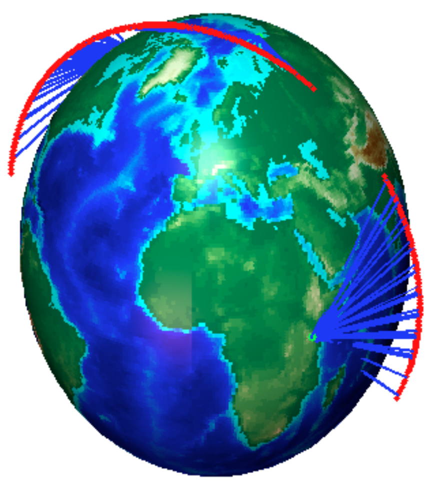
ASEN 5010 Course Project

Zach Dischner
University of Colorado at Boulder
Department of Aerospace Engineering

The essence of attitude determination is the estimation of a body's pointing relative to a known frame, given one or more observations taken in different frames. This paper will discuss the development and theoretical testing of attitude estimation algorithms for a satellite two such sensors. Numerical simulations will simulate measurements taken in various frames, and from them determine the spacecraft's attitude relative to the inertial frame. Various parameters such as measurement noise, and orbit characteristics will be examined in order to quantify their impact on performance on the attitude estimation algorithms.



Nomenclature

i Orbit Inclination Angle
Ω Longitude of the Ascending Node
θ Orbit Position Angle
r Orbit Radius
r_E Mean Radius of the Earth
μ Gravitational Constant
n Mean Orbit Rate
$[XY]$ DCM Mapping Y to X Frames
${}^{X}x$ x expressed in X frame components
Left Superscript
N Inertial Frame
B Body Frame
T Topographical {n,e,d} Frame

I. Introduction

The spacecraft in question for this examination is equipped only with sun and magnetic field direction sensors. Numerical simulations of measurements taken from both of these devices in the B frame, as well as the N frame. Relating measurements between frames, and ultimately providing a measure of the spacecraft's attitude, will be done with the Optimal Linear Attitude Estimation (OLAE) method. Through this numerical simulation, the affect of sensor noise, errors in the orbit parameters, and other sources of error will be examined and quantified.

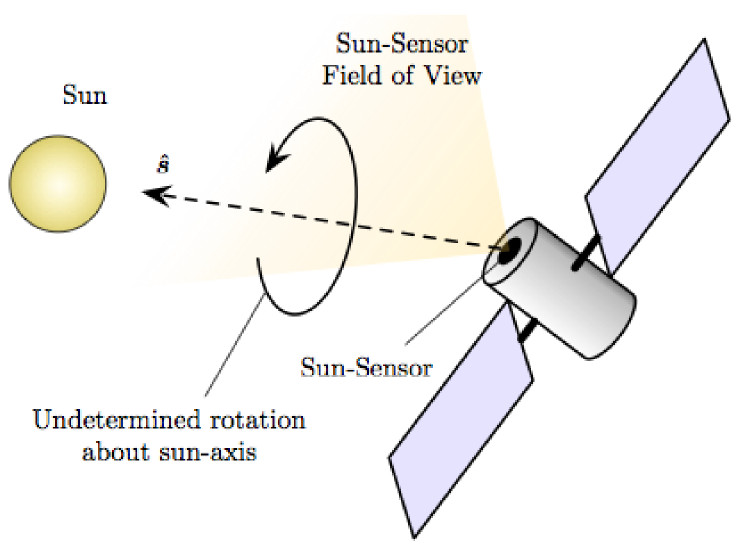


Figure 1. Satellite Sun Sensor Visual

The spacecraft itself is orbiting the Earth in a pure circular orbit, and is tumbling free of any active attitude control. The inertial frame position of the satellite is given by:

$$ {}^{N}r = \begin{pmatrix} \cos\Omega\cos\theta - \sin\Omega\sin\theta\cos i \\ \sin\Omega\cos\theta + \cos\Omega\sin\theta\cos i \\ \sin\theta\sin i \end{pmatrix} \tag{1} $$

where

$$\Omega = 2^o$$
$$i = 75^o$$

For this simulation, the longitude of the ascending node is Ω, the orbit inclination is i, and θ is the orbit position angle relative to the equatorial crossing point. As the orbit is circular, its governing equation is:

$$\theta(t) = \theta_0 + nt \tag{2}$$

where n is the mean orbit rate, r is the constant orbit radius, and mu is the Earth's gravitational constant.

$$n = \sqrt{\frac{\mu}{r^3}}$$
$$r = 6878\text{km}$$
$$\mu = 398600\text{km}^3/\text{s}^2$$

The spacecraft is axially symmetric, and has the inertia tensor given in body frame components:

$$^{B}I = \begin{bmatrix} 25 & 2.5 & 0.5 \\ 2.5 & 20 & 0 \\ 0.5 & 0 & 15 \end{bmatrix} \text{kg m}^2$$

The spacecraft also is experiencing no external torques, and has initial 3-2-1 Euler angles and initial angular rates defined as:

$$\begin{bmatrix} \psi_0 \\ \theta_0 \\ \phi_0 \end{bmatrix} = \begin{bmatrix} 5^o \\ 10^o \\ -5^o \end{bmatrix} \Bigg| \; ^{B}\omega_0 = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \end{bmatrix} \text{deg/s}$$

II. Results

In this section, the development, testing, and simulation of various subprocess in the attitude determination process will be discussed. Unless otherwise mentioned, all numerical calculations, simulations, and visualizations were all performed in Matlab, and all code can be found in IV.

II.A. Part A

The first step in setting up a numerical simulation of the satellite was to obtain its position in orbit for an arbitrary time. Using EQ 1, solutions for the satellite positions were found over a ten minute period. Figure 2 shows the result of this simulation.

II.B. Part B

Next, a routine was written to take a body orientation as a set of MRPs $\sigma_{\mathbf{B/N}}$, and with the known inertial sun direction vector $^{N}\hat{s} = (0, -1, 0)^T$, compute the simulated sun direction measurement in body coordinates.

This problem essentially boils down to rotating a measurement in one frame into another frame. In general, this is done by multiplying one observation by the appropriate rotation matrix.

$$^{X}a = [XY]^{Y}x$$

Where in this case, the two frames are the body B and inertial N frames, and the vector quantity being rotated is the measured sun direction vector $\hat{s}$.

$$^{B}\hat{s} = [BN]^{N}\hat{s}$$

In the above equation, $[BN]$ is the 3x3 rotation matrix, or DCM, which describes the three dimensional rotation between the B and N frames. In this case, this rotation is given as a set of MRPs. From lecture notes, the 3x3 DCM can be extracted from a set of MRPs through the relationship in EQ 3.

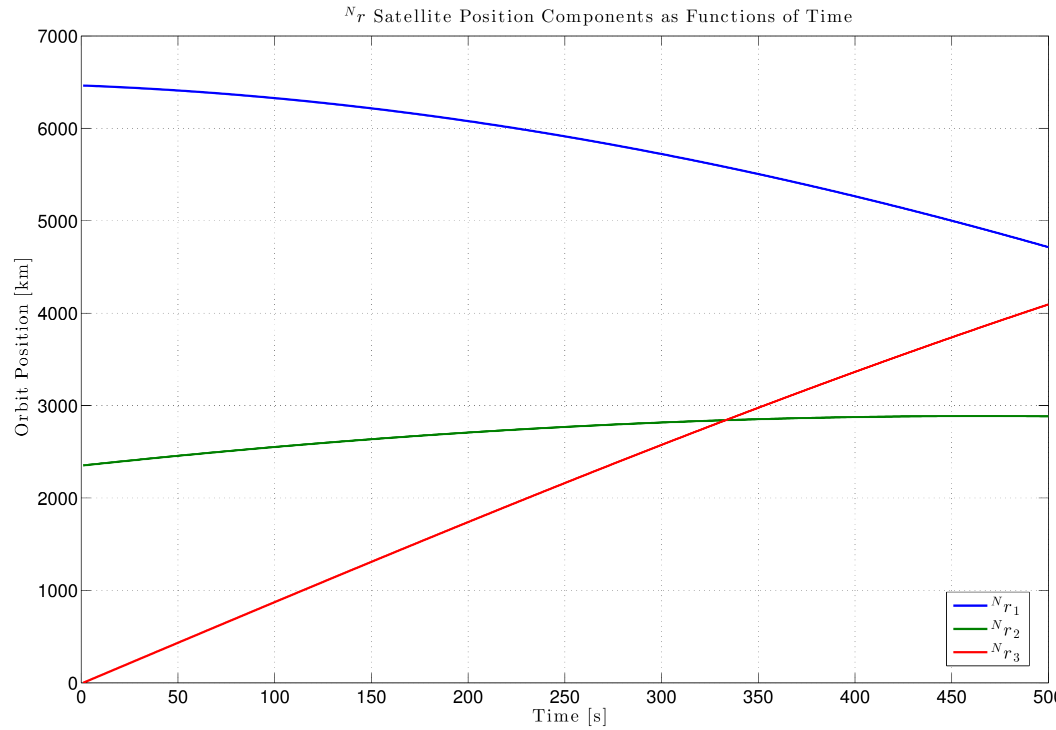


Figure 2. Orbit Position in Time

$$[BN] = [I_{3x3}] + \frac{8[\tilde{\sigma}]^2 - 4(1-\sigma^2)[\tilde{\sigma}]}{(1+\sigma^2)^2} \tag{3}$$

This conversion was provided with the course's computational toolbox, and was implemented in Matlab here. The code for this routine is found in Appendix IV.B.

II.C. Part C

The frame for the satellite to be used in this simulation is a North-East-Down (NED) topographic frame τ. Another main frame of interest is the Earth-fixed frame ϵ.

$$\tau : \{\hat{n}, \hat{e}, \hat{d}\}$$

$$\epsilon : \{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$$

This frame is illustrated in

In order to describe the arbitrary rotation between the two frames, a DCM was constructed. Starting at the ϵ frame, I first rotated λ about the third axis, $\hat{e}_3 = \hat{n}_3$.

$$[M_3(\lambda)] = \begin{bmatrix} \cos\lambda & \sin\lambda & 0 \\ -\sin\lambda & \cos\lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Now, $\hat{e}'_2 = \hat{e}$. The next rotation is about $\hat{e}$ needs to be a negative rotation of magnitude θ. However, this rotation produces a frame in which $\hat{e}''_3 = \hat{n}$. By adding another $\pi/2$ rotation to the previous, all three axes of the ϵ'' frame align with the new NED τ frame.

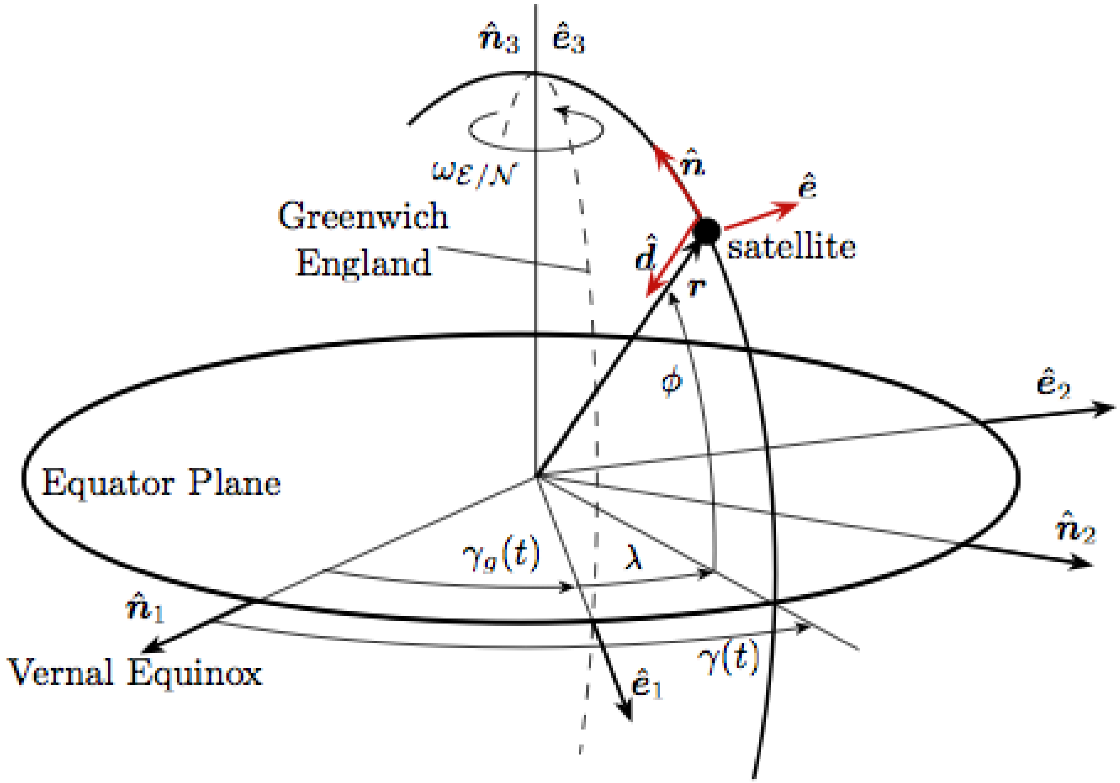


Figure 3. North East Down Frame

$$[M_2(-\phi\pi/2)] = \begin{bmatrix} \cos(-\phi\pi/2) & 0 & -\sin(-\phi\pi/2) \\ 0 & 1 & 0 \\ \sin(-\phi-\pi/2) & 0 & \cos(-\phi\pi/2) \end{bmatrix} \tag{5}$$

With no orbit inclination, these two orbit inclinations are all that are necessary to rotate the τ frame into the ϵ frame. The resulting DCM is computed by multiplying EQ 5 by EQ 4.

$$[TE] = [M_2][M_3] \tag{6}$$

II.D. Part D

Like in the previous section, another frame of interest is the earth-centered-inertial (ECI) frame. For the purpose of this exercise, the ECI frame differs from the ECEF frame by the angle γ, the Greenwich local sidereal time. It is assumed that this angle is a constant function of time, and only has components in the $\hat{e}_3$ direction.

$$\gamma(t) = \gamma_{t_0} + \omega_{E/N} \tag{7}$$

$$\omega_{E/N} = 361\text{deg/day}$$

Now, to rotate between the ECI and ECEF frames, a rotation matrix DCM can be built in the same manner as before in section II.C. But now, the rotation angle γ is a function of time.

$$[EN] = [M_3(\gamma(t))] = \begin{bmatrix} \cos\gamma(t) & \sin\gamma(t) & 0 \\ -\sin\gamma(t) & \cos\gamma(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The Matlab code for this computation is included in Appendix IV.D.

II.E. Part E

Combining many of the above sections, now a routine was written in which the magnetic field unit direction vector in inertial frame components ${}^{N}M$ was solved for, given inputs ${}^{N}r$ and t.

The magnetic field vector was given in NED frame components in EQ 9.

$$\begin{bmatrix} M_{north} \\ M_{east} \\ M_{down} \end{bmatrix} = [\because] \begin{bmatrix} 29900 \\ 1900 \\ -5530 \end{bmatrix} \text{nT} \tag{9}$$

r_{eq} is the earth's mean equatorial radius, r is the mean orbit radius, λ is the satellite longitude relative to ECEF, and ϕ is the satellite latitude.

Since EQ 9 is given, and from the orbit position vector, both λ and ϕ can be calculated, the task now is to rotate the computed magnetic field components from the NED (T) frame to the N frame. In the previous sections, I discussed how the [TE] and [EN] frames were constructed, as functions of λ, ϕ, and t. In this routine, those routines were used to get a final inertial representation of the earth's magnetic field at a point in time along the orbit, as shown in EQ 10.

$$\begin{bmatrix} M_{\hat{n}_1} \\ M_{\hat{n}_2} \\ M_{\hat{n}_3} \end{bmatrix} = [EN]^T[TE]^T \begin{bmatrix} M_{north} \\ M_{east} \\ M_{down} \end{bmatrix} \tag{10}$$

A simulation of this calculation was made for ten minutes of an orbit, and the resulting plot is shown in Figure 4. The code for this computation is included in Appendix IV.E.

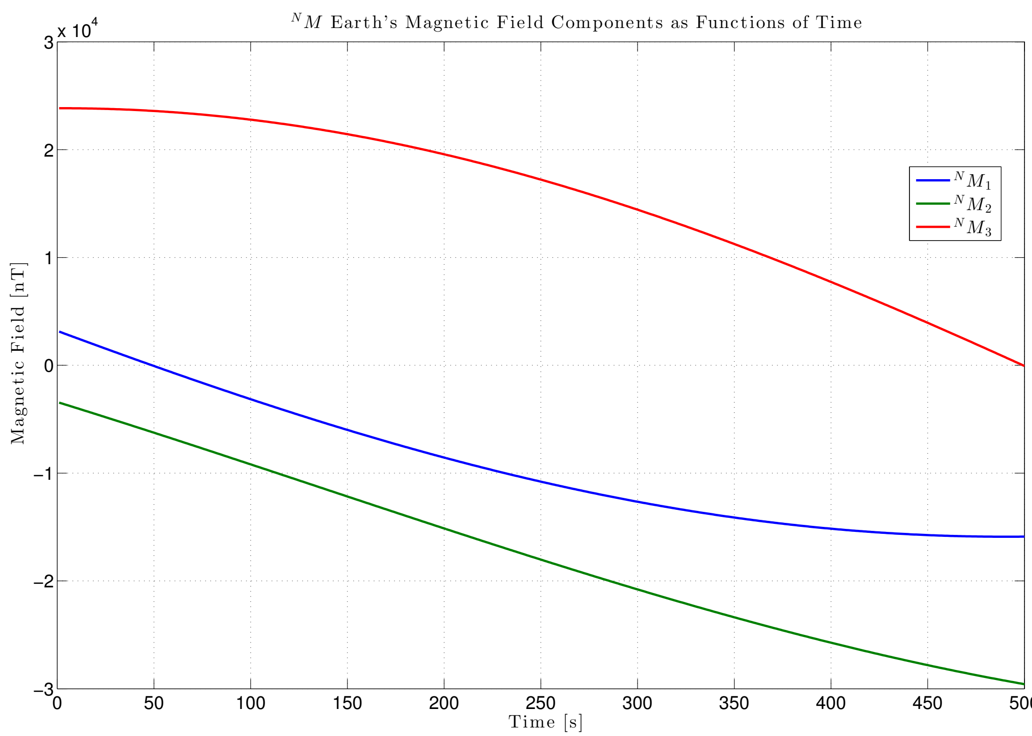


Figure 4. Manetic Field Elements

III. Conclusion

Here I shall conclude!

IV. Appendix

IV.A. Appendix A: Code for part A

```
function r_N = computeR_N(t)
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                           computeR_N.m
% Author:   Zach Dischner
% Date:     April 1, 2013
%
% Usage:
%   r_N = computeR_N(t)
%
% Description:  Computes orbit position vector in inertial frame components
%               for ASEN 5010 project circular orbit
%
% Inputs:  t    ==> Time value array [s]. Either row or column vector
%
% Outputs: r_N  ==> Orbit position in N frame components
%                  | r1(t1)    r2(t1)  r3(t1) |
%                  | r1(t2)    r2(t2)  r3(t2) |
%                  | r1(t3)    r2(t3)  r3(t3) |
%                               ...
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

%% Orbit Parameters
Omega   = deg2rad(20);      % RAAN?                        [rad]
i       = deg2rad(75);      % Orbit Inclination Angle      [rad]
mu      = 398600;           % Gravatational Parameter      [km^3/s^2]
r       = 6878;             % Circular Orbit Radius        [km]
n       = sqrt(mu/r^3);     % Mean Orbit Rate              [s]
theta_0 = 0;                % Initial Angle                [rad]

%% Compute r_N(t)
if ~iscolumn(t)
    t=t';
end

theta   = theta_0 + n*t;

r_N = r*[ cos(Omega).*cos(theta)-sin(Omega).*sin(theta).*cos(i) , ...
        sin(Omega).*cos(theta)+cos(Omega).*sin(theta).*cos(i) , ...
                            sin(theta).*sin(i)                  ];
```

IV.B. Appendix B: Code for part B

```
    function s_B = computeSunVec_B(sigma_BN)
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                           computeSunVec_B.m
% Author:   Zach Dischner
% Date:     April 4, 2013
%
% Usage:
%   s_B = computeSunVec_B(sigma_BN)
%
% Description:  Computes the sun attitude attitude as seen by the satellite
%               body. It does so with a constant inertial sun attitude, and
%               the MRP set sigma_B/N.
%
%               Given s_N, and BN, get s_B
%
```

```
% Inputs:  sigma_BN    ==> MRP set describing rotation between B and N
%                          frames
%
% Outputs: s_B  ==> Sun direction vector in B frame components
%
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

%% Data Masseuse
if ~iscolumn(sigma_BN)
    sigma_BN = sigma_BN';
end

%% Define Sun Position Vector
s_N = [ 0 ; -1 ; 0 ];

%% Convert MRPs into [BN] DCM
BN  = MRP2C(sigma_BN);

%% Get the Rotation
s_B = BN*s_N;
```

IV.C. Appendix C: Code for part C

```
function TE = Earth2TopoDCM( lambda, phi)
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                           Earth2TopoDCM.m
% Author:   Zach Dischner
% Date:     April 4, 2013
%
% Usage:
%   TE = Earth2TopoDCM( lambda, phi, inclination )
%
% Description:  Maps the earth-fixed frame [EN] to a topographic frame
%               [TE] ==> {n e d}
%
%               Lamens ==> Rotation matrix between earth-fixed and
%                          topographic frame
%
% Inputs:  lambda ==> Angle between grenwich and equator orbit crossing
%          phi    ==> Along-track angle between satellite and equator plane
%                     (satellite latitude)
% ???      inclination ==> Orbit inclination?
%
% Outputs: TE     ==> DCM from earth fixed frame to the topographic frame
%
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


%% Define Individual Rotations
% First: Rotate LAMBDA about e^3
rot1    = Euler3(lambda);

% Second: Rotate (-PHI  - pi/2)about e^2' (Can be done in just one
% rotation.
rot2    = Euler2(-phi - pi/2);

% Third: Inclination angle?
inclination = 0;
rot3    = Euler1( inclination );

%% Assemble the Rotation Matrix
TE = rot3*rot2*rot1;
```

IV.D. Appendix D: Code for part D

```
function EN = ECI2ECF( t )
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
%                           ECI2ECF.m
% Author:   Zach Dischner
% Date:     April 6, 2013
%
% Usage:
%   EN = ECI2ECF( t )
%
% Description:  Maps the earth-centered inerital frame [N] to an
%               earth-centered earth-fixed frame [E]. Assumes constant
%               rotational rate
%
%               Lamens ==> Rotates ECI to ECF by [EN]
%
% Inputs:  t  ==> Time that has passed since grenwich 0:00:00 ???
%
% Outputs: EN ==> DCM from ECI to ECEF.
%
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

%% Obtain Rotation Rate
w_EN = 361 *(pi/180)*(1/24)*(1/3600); %rotation [deg/day][rad/deg][day/hour][hour/sec]

%% Obtain Rotation Angle
gamma_t0 = deg2rad(20);

gamma    = gamma_t0 + w_EN*t;

%% Assemble DCM
EN       = Euler3(gamma);
```

IV.E. Appendix E: Code for part E

```
function M_N = r2MagVec( r_N, t )
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                           r2MagVec.m
% Author:   Zach Dischner
% Date:     April 6, 2013
%
% Usage:
%   M_N = r2MagVec( r_N, t )
%
% Description:  Compute the Magnetic field vector for a given satellite
%               position and time
%
%               Lamens ==> Find the magnetic field vector at a point in
%                          space and time
%
% Inputs:   r_N  ==> Inertial satellite position vector [r1 r2 r3]
%           t    ==> Time that has passed since grenwich 0:00:00 ???
%
% Outputs:  M_N  ==> Magnetic field vector in N frame components
%
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

%% Defines and Constants
r_E     = 6378; % Earth's Equatorial Radius [km]
r_orbit = norm(r_N);
% lam     = 0.001*sin(t);
% phi     = 0.1*cos(t);
lam     = atan2(r_N(2), r_N(1)); % *look at it again
phi     = asin(r_N(3)/r_orbit);

%% Calculate Magnetic Field Vector
M_T     = -(r_E/r_orbit)^3.*...
                        [ -cos(phi)      sin(phi)*cos(lam)      sin(phi)*sin(lam);...
                            0              sin(lam)              -cos(lam)    ;...
                         -2*sin(phi)  -2*cos(phi)*cos(lam)   -2*cos(phi)*sin(lam)]...
                         *[29900; 1900; -5530];        % nT
```

```
EN      = ECI2ECF(t);
TE      = Earth2TopoDCM(lam,phi);

M_N     = EN'*TE'*M_T;
```

An appendix, if needed, should appear before the acknowledgments. Use the 'starred' version of the `\section` commands to avoid section numbering.

Acknowledgments

A place to recognize others.

References

[1]Rebek, A., *Fickle Rocks*, Fink Publishing, Chesapeake, 1982.