We did some more research for inertial measurement units. We found four more companies and e-mailed them about their product information.

- BAE Systems
- MicroSCIRAS
- KVH
- Summit Instruments

Their data sheets are posted in our notebooks. The IMU from Summit Instruments is about $25,000 because they deal mainly with the military, so they are out of our price range. The relevant drift information for these companies are posted on the interesting products link on our website.

We made a new function called earth_param, that has all the earth constant parameters listed. This function will return an array as shown in the code below:

```
function [earth]=earth_param()

% EARTH_PARAM  Returns the constant parameters of the earth for use
in different functions.
%
% [earth] = earth_param()
%
% Returns:
%         a - earth equitorial radius
%         b - earth polor radius
%         e - eccentricity of elipsoid
%         f - (a-b)/a
%         o - omega - earth turn rate
%
% Written By: Brian Bleeker
%             Rob MacMillan
earth.a = 6378137.0;
earth.b = 6356752.3142;
earth.f = (earth.a-earth.b)/earth.a;
earth.e = sqrt(earth.f*(2-earth.f));
earth.o = 7.292115E-5;
```

Then we had to change our original functions, ecef2ned and ned2ecef to accomidate this change. The new code is shown below:

```
function [NED] = ecef2ned(gtime, ecefpos)

% ECEF2NED  Convert ecef coordinates to NED coordinates and diplay.
%
% [NED] = ecef2ned(gtime, ECEF position)
%  ECEF Position Array = [x, y, z]
%
%  Example:
%     array.x
%     array.y
%     array.z
%  *** The array parameters must have *.x, *.y, and *.z.
```

```
%
% Enter the time, and the x,y,z positions in ecef coordinates.
% This function will display two figures:
%           1:  x,y,z position in ecef coordinates and
%           2:  latitude, longitude, and altitude
%
% NED is returned with 3 paramerters:
%           1:  NED.lat - latitude
%           2:  NED.lon - longitude
%           3:  NED.alt - altitude
%
% A negative longitude is the Western hemispere.
% A negative latitude is in the Southern hemisphere.
%
% Written  By: Brian Bleeker
%              Rob MacMillan

earth = earth_param;                               %get earth
parameters

gtime = gtime(:,1) - gtime(1,1);

figure(1), subplot(311), plot(gtime, ecefpos.x), grid
title('ECEF X Position')

subplot(312), plot(gtime, ecefpos.y), grid
title('ECEF Y Position')

subplot(313), plot(gtime, ecefpos.z), grid
title('ECEF Z Position')

len = length(ecefpos.x);                            %get length of
data

for i = 1:len
    lon(i) = atan2(ecefpos.y(i), ecefpos.x(i));     %long =
atan(y,x) - direct
    lon(i) = lon(i)*180/pi;                         %convert to
degrees

    h = 0;                                          %initialize
    N = earth.a;
    flag = 0;
    j = 0;
    p = sqrt(ecefpos.x(i)^2 + ecefpos.y(i)^2);

    sinlat = ecefpos.z(i)/(N*(1-earth.e^2)+h);      %First iteration
    lat(i) = atan((ecefpos.z(i)+earth.e^2*N*sinlat)/p);
    N = earth.a/(sqrt(1 - (earth.e^2)*(sinlat^2)));
    prevalt = (p/cos(lat(i)))-N;
    prevlat = lat(i)*180/pi;

    while (flag < 2)                                %do at least 100
iterations
        flag = 0;
        sinlat = ecefpos.z(i)/(N*(1-earth.e^2)+h);
        lat(i) = atan((ecefpos.z(i)+earth.e^2*N*sinlat)/p);
        N = earth.a/(sqrt(1 - (earth.e^2)*(sinlat^2)));
        alt(i) = (p/cos(lat(i)))-N;
        lat(i) = lat(i)*180/pi;
        if abs(prevalt-alt(i)) < .00000001
            flag = 1;
        end
        if abs(prevlat-lat(i)) < .00000001
            flag = flag + 1;
        end
        j = j+1;
        if j == 100
            flag = 2;
        end
        prevalt = alt(i);
        prevlat = lat(i);
    end
 end

 NED.lat = lat;
 NED.lon = lon;
 NED.alt = alt;

figure(2), subplot(311), plot(gtime, NED.lon), grid
title('NED Longitude')

subplot(312), plot(gtime, NED.lat), grid
title('NED Latitude')

   subplot(313), plot(gtime, NED.alt), grid
```

```
function [ecef_pos] = ned2ecef(ned)
% NED2ECEF2  Converts NED coordinates to ECEF coordinates.
%
% [ecef_pos] = ned2ecef(ned)
%
% Returns:
%         ecef_pos.x - x position
%         ecef_pos.y - y position
%         ecef_pos.z - z position
%
% Written By: Brian Bleeker
%             Rob MacMillan

earth = earth_param;
N = (earth.a)/(sqrt(1-(earth.e)^2*(sin((ned.lat*(pi/180))))^2));


coslat = cos(ned.lat*(pi/180));
sinlat = sin(ned.lat*(pi/180));
coslon = cos(ned.lon*(pi/180));
sinlon = sin(ned.lon*(pi/180));

ecef_pos.x = (N+ned.alt)*coslat*coslon;
ecef_pos.y = (N+ned.alt)*coslat*sinlon;  % compute the current
longitude

ecef_pos.z = (N*(1-earth.e^2)+ ned.alt)*sinlat;
```

Then we had a discussion with Dr. Ahn. We showed him our block diagram for the attitude computer. He gave us some suggestions on what we should change. The new block diagram for the attitude computer can be seen in FIG :

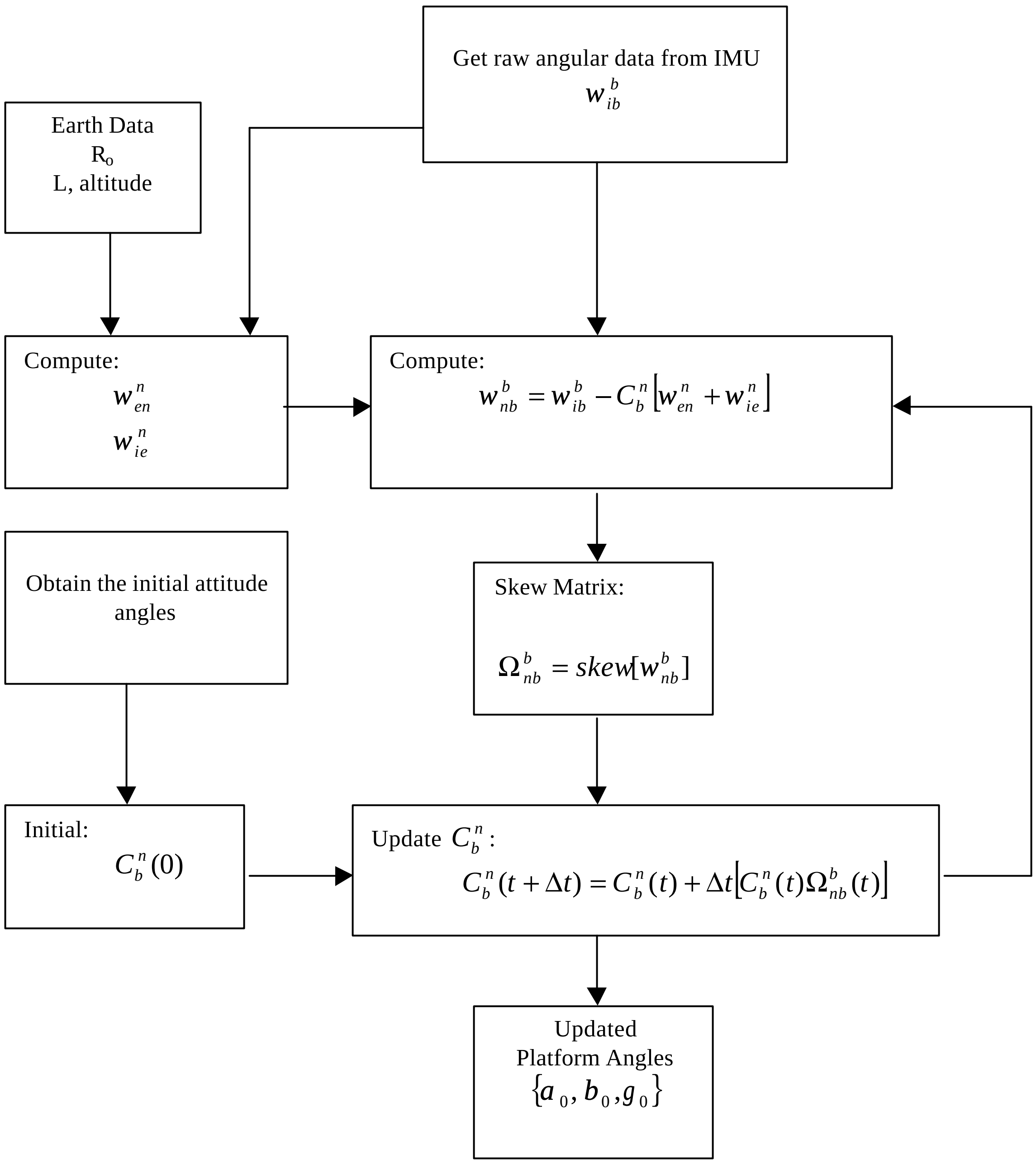
Get raw angular data from IMU
w_{ib}^{b}
Earth Data
R_o
L, altitude
Compute:
w_{en}^{n}
w_{ie}^{n}
Compute:
$w_{nb}^{b} = w_{ib}^{b} - C_b^n\left[w_{en}^{n} + w_{ie}^{n}\right]$
Obtain the initial attitude angles
Skew Matrix:
$\Omega_{nb}^{b} = skew[w_{nb}^{b}]$
Initial:
$C_b^n(0)$
Update C_b^n:
$C_b^n(t+\Delta t) = C_b^n(t) + \Delta t\left[C_b^n(t)\Omega_{nb}^{b}(t)\right]$
Updated
Platform Angles
$\{a_0, b_0, g_0\}$

Fig - (Block Diagram for attitude computer)

Now we need to create the rest of the functions to complete the block diagram for the attitude computer. We want to create a separate function for each block.

First, we created a function for the turn rate of the earth with respect to the inertial frame in the NED frame.

```
function [win] = trEARTH_IF_NED(lat)

% TREARTH_IF_LGF  Find the turn rate of the earth with respect to
the inertial
%                 frame in the local NED frame.
%
% [win] = trEARTH_IF_NED(Latitude)
%
% Latitude is in decimal degrees.
%
% Written By: Brian Bleeker
%             Rob MacMillan

earth=earth_param;

win(1) = earth.o * cos(lat*pi/180);
win(2) = 0;
win(3) = -earth.o * sin(lat*pi/180);

win = transpose(win);
```

The following pieces of code were created:

- The turn rate of the NED frame with respect to the earth.
- The turn rate of the body with respect to the NED frame in the body frame.
- The skew matrix operation.
- Update directional cosine matrix function.

```
function [wen] = trLGF_earth(ve, vn, lat, h)

% TRLGF_EARTH  Find the turn rate of the local geo frame with
respect to the earth.
%
% [wen] = trLGF_earth(Velocity east, Velocity north, Latitude,
altitude)
%
% Latitude is in decimal degrees.
%
% Written By: Brian Bleeker
%             Rob MacMillan

earth=earth_param;
Rn = (earth.a(1-earth.e^2))/((1-
earth.e^2*(sin(lat*pi/180))^2)^(3/2));
Re =  earth.a/((1-earth.e^2*(sin(lat*pi/180))^2)^(1/2));

wen(1) = ve/(Re+h);
wen(2) = -vn/(Rn+h);
wen(3) = (-ve*tan(lat*pi/180))/(Re+h);
```

```
wen = transpose(wen);
```

```
function [wnb]=trBODY_NED_BODY(win,wen,wib,cbn)

% TRBODY_NED_BODY   Find the turn rate of the vehicle with respect
to the navigaton
%                   frame in the body frame.
%
% [wnb] = trBODY_NED_BODY(turn rate of the earth w.r.t inertial
frame in local geo frame,
%                         turn rate of local geo frame w.r.t the
earth,
%                         raw angular data from IMU, Directional
cosine matrix)
%
%
% Written By: Brian Bleeker
%             Rob MacMillan

wnb=wib-cbn*(wen+win);
```

```
function [omeganb] = skew(wnb)
% SKEW(WNB)    Derive the skew matrix of wnb
%     WNB = [wnb.x,wnb,y,wnd.z]
% skew(wnb)
%             0     -wnb.z     wnb.y
%             wnb.z     0     -wnb.x
%            -wnb.y  wnb.x       0
%
% Written By: Brian Bleeker
%             Rob MacMillan
omeganb = [0,-wnb.z,wnb.y;wnb.z,0,-wnb.x;-wnb.y,wnb.x,0];
```

```
function [cbn2]=update_cbn(cbn, delt, omeganb)

% UPDATE_CBN    Update the directional cosine matrix to compute
%               new attitude angles.
%
% [cbn] = update_cbn(cbn - old values, delta time, skew(wnb))
%
% Written By: Brian Bleeker
%             Rob MacMillan
%
cbn2 = cbn + delt[cbn*omeganb];
```

We also created a help file called navigation to show what functions can be used for IMU data manipulation.

```
function navigation

%earth_param     - Returns the constant earth parameters to be
used in other functions.
%ECEF2NED        - Convert ecef coordinates to NED coordinates.
%NED2ECEF        - Convert NED coordinates to ecef coordinates.
%skew            - Derive the skew matrix of wnb
```

```
%trLGF_EARTH     - Find the turn rate of the local geo frame with
respect to earth.
%trBODY_NED_BODY - Find the turn rate of the vehicle in the
navigation frame with respect
%                  to the body frame.
%trEARTH_IF_NED  - Find the turn rate of the earth with respect
to the inertial frame
%                  in the local NED frame.
%update_cbn      - Update the directional cosine matrix.
%
%Help function   - To find more information about these
functions.
```