Winter Interim
12/19/02 – 12/21/02

All individual functions for the attitude computer are written. Now the program for the attitude computer can be written. First, here is the code for the initial C_b^n. It looks a little messed up because there are a lot of sine and cosine equations involved in this matrix. To see the equation for the directional cosine matrix, see equation 9.

```
function [cbn] = cbninit(yaw, pitch, roll)
% CBNINIT(yaw, pitch, roll)    Derive the initial directional cosine matrix
%
% Written By: Brian  Bleeker
%             Rob MacMillan

yaw = yaw*pi/180;
pitch = pitch*pi/180;
roll = roll*pi/180;

cbn = [cos(pitch)*cos(yaw), -cos(roll)*sin(yaw) + sin(roll)*sin(pitch)*cos(yaw),
sin(roll)*sin(yaw) + cos(roll)*sin(pitch)*cos(yaw);
      cos(pitch)*sin(yaw), cos(roll)*cos(yaw) + sin(roll)*sin(pitch)*sin(yaw), -
sin(roll)*cos(yaw) + cos(roll)*sin(pitch)*sin(yaw);
      -sin(pitch), sin(roll)*cos(pitch), cos(roll)*cos(pitch)];
```

Just for reference of the new Euler angles, a function was written to derive the Euler angles from the directional cosine matrix. The equation for this was found on page 50. Note, the function has not been written yet for the provision that ? approaches +- 90°. This will have to be done in the future because the following equations will approach 0/0. Here is what has been written so far.

```
function euler(cbn, pos)
% euler(cbn, position)    Derive the euler angles from cbn
%
% Written By: Brian  Bleeker
%             Rob MacMillan

global angles;

angles.y(pos) = (atan(cbn(2,1)/cbn(1,1)))*180/pi;
angles.p(pos) = (asin(-cbn(3,1)))*180/pi;
angles.r(pos) = (atan(cbn(3,2)/cbn(3,3)))*180/pi;
```

These equations also look a little messy in the Matlab code, so here they are written out. It can be seen from equation 9 how these equations are derived. It can now be seen why new equations have to be written in the case that the denominator is 0. This will be done later.

Roll: $f = \arctan\left\{\frac{C_{32}}{C_{33}}\right\}$ (eq 1)

Pitch: $q = \arcsin\{-C_{31}\}$ (eq 2)

Yaw: $j = \arctan\left\{\frac{C_{21}}{C_{11}}\right\}$ (eq 3)

It was found that the equation for $?_{nb}^{b}$ was incorrect. When relating the turn rate of the navigation frame with respect to the earth in the navigation frame and the turn rate of the earth with respect to the inertial frame in the navigation frame, C_n^b should be used instead of C_b^n. C_b^n relates the body frame to the navigation frame. Since the turn rate in the body frame is desired, C_n^b should be used.
Not much had to be done to the overall attitude computer. Deriving C_n^b is done by transposing C_b^n. This was found on page 44.

$$C_n^b = C_3 C_2 C_1 \quad \text{(eq 4)}$$

$$C_b^n = C_n^{bT} = C_1^T C_2^T C_3^T \quad \text{(eq 5)}$$

Yaw – Rotation ? about the z axis, $C_1 = \begin{bmatrix} \cos y & \sin y & 0 \\ -\sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (eq. 6)

Pitch – Rotation ? about the y axis, $C_2 = \begin{bmatrix} \cos q & 0 & -\sin q \\ 0 & 1 & 0 \\ \sin q & 0 & \cos q \end{bmatrix}$ (eq. 7)

Roll – Rotation F about the x axis, $C_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos f & \sin f \\ 0 & -\sin f & \cos f \end{bmatrix}$ (eq. 8)

$$C_b^n = \begin{bmatrix} \cos q \cos j & -\cos f \sin j + \sin f \sin q \cos j & \sin f \sin j + \cos f \sin q \cos j \\ \cos q \sin j & \cos f \cos j + \sin f \sin q \sin j & -\sin f \cos j + \cos f \sin q \sin j \\ -\sin q & \sin f \cos q & \cos f \cos q \end{bmatrix} \quad \text{(eq. 9)}$$

This is the newest attitude computer written thus far.

```
%ATTITUDE COMPUTER
%-------------------------------------------------------------------------
-----------------------------------

%GLOBAL VARIABLES
global angles;

%Get IMU data from file
%----------------------
%wib
wib = [0,0,0]';
v_e_n.n = 0;
v_e_n.e = 0;
v_e_n.d = 0;

%Enter the roll, pitch, yaw
yaw = 0;
pitch = 0;
roll = 0;
lat = 0;
delt = 24;
h = 0;
i = 1;
time = 0;
len = 3600;

earth = earth_param;                          %obtain earth data

prevcbn = cbninit(yaw, pitch, roll);        %inital directional cosine
matrix
cnb = prevcbn';                               %get cnb to relate NED to
body for wnb
euler(prevcbn, i);                      %obtain the euler angles
time(i) = 0;

for i = 1:len
    win = trEARTH_IF_NED(lat);                    %turn rate of the earth
    w.r.t inertial frame in the NED frame
    wen = trLGF_earth(v_e_n, lat, h);           %turn rate of the NED
frame w.r.t earth in the NED frame
    wnb = trBODY_NED_BODY(win,wen,wib,cnb); %turn rate of the body
w.r.t NED in the body frame
    skewwnb = skew(wnb);                          %skew wnb
    cnb = update_cbn(cnb, delt, skewwnb);   %update the directional
cosine matrix
    cbn = cnb';
    euler(cbn, (i+1))                     %obtain the euler angles
    time(i+1) = time(i)+delt;
end

figure(1), subplot(311), plot(time, angles.r), grid
title('Roll - Time in seconds')

subplot(312), plot(time, angles.p), grid
```

```
title('Pitch - Time in seconds')

subplot(313), plot(time, angles.y), grid
title('Yaw - Time in seconds')
```

This program was run using initial values of 0 for everything. It also assumed that there was no velocity or angular rates. This was done just to see what would happen to the Euler angles. The plots of these angles can be seen below.

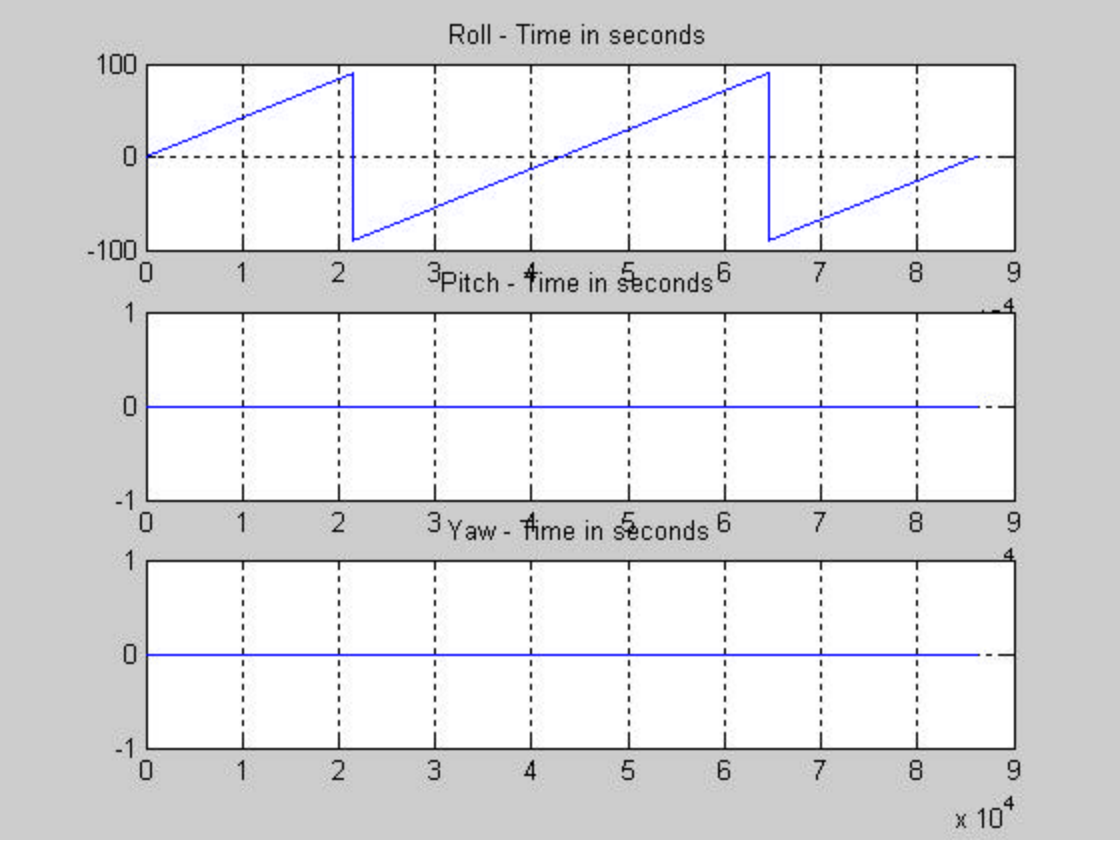

Figure 1 – Plots of Roll, Pitch and Yaw in seconds

This program was run at a rate of 24 seconds, 3600 times. This is equal to one day. It can be seen in figure 1 that the yaw and pitch stayed at zero. The roll however, went from 0° to 90°, -90° to 90°, and -90° to 0°, or a 360 degree rotation. This is due to the rotation of the earth. To show the accuracy of the equations, this is the data for one iteration, or 24 seconds.

The earth rotates about 15.041067°/hour. In 24 seconds, the earth rotates 0.10027378°. When running the program, the roll after 24 seconds = 0.100273676798553°.

$$\% \text{ Error} = \left| \frac{Theoretical - Experimental}{Theoretical} \right| \times 100 = 0.000103\%$$

When running the program 3600 times, the roll after 1 day = 360.985236474792547°
This also is an error of 0.000103%.

When this is done at an increment of 0.001 seconds, and starting at 0°, the % error is:

$$\left|\frac{0.015041067 - 0.015041066876065}{0.015041067}\right| \times 100 = 8.2394 \times 10^{-7}\%$$

This error is very minimal over 3600 iterations. This is also the same value over 36000 iterations, a step of 0.0001 seconds. This was tried with different time step values and different number of iterations. Every time it was done, there was very minimal error.

1/28/03

Testing Platform

The original plan for today was to finish the development of the IMU testing platform. We haven't been able to find a gearbox for the rotating portion of the testing platform y et, therefore we are restricted from building anything but the base of the platform. We need to know the dimensions of the gearbox to be able to construct the uprights that hold the gearbox and rotating rod. We did purchase the wood for the base and uprights and the casters. We will have to wait two lab periods for Dave Miller, lab shop supervisor, to return from vacation. He will be constructing the rotating rod made of metal that needs to be flattened where it will be attached to the metal plate that will hold the IMU.

IMU User's Manual

We picked up the IMU that was sent to Bradley over winter interim. We read the 'DMU User's Manual' which is the generic manual from Crossbow for their inertial sensors. We had to read over it carefully so that we don't damage the $3,000.00 product. Most importantly, for now, the power supply must be connected properly.

Red = (+)
Black = (GND)

The Gyro-View software that came with the IMU was installed on the laboratory computer. Set up for the IMU is pretty simple. Just connect the power supply as explained above and connect the 9-pin connector to the serial port of the computer and the 15-pin connector to the IMU. Upon startup, the software recognizes when the IMU is connected to the computer and power supply properly.

Data Collection

The IMU has three modes with which data can be collected.

Voltage Mode: The analog sensors are sampled and converted to digital data with 1 mV resolution. The data is 12-bit, unsigned. The data is just voltages, which would need to be converted to numerical angular rates and accelerations before we could analyze the data.

Scaled Sensor Mode: The analog sensors are sampled, converted to digital data, temperature compensated, and scaled to engineering units. The digital data represents the actual value of the quantities

measured. The acceleration data is measured in g's and therefore must be converted to force to be useful data. The angular rate data is measured in degrees per second.

Angle Mode: This mode acts the same as the scaled sensor mode, however it outputs the yaw, pitch, and rolls angles with the other data.

We will be using the scaled sensor mode. The data is outputted to a text file in a spreadsheet format, which can be edited nicely in Matlab. We didn't use the angle mode because it gives the angles of the yaw, pitch, and roll, which is data we don't need.

IMU Coordinate System

X-axis: From face with connector through the IMU
Y-axis: Along the face with connector from left to right.
Z-axis: Along the face with the connector from top to bottom.

We didn't have much time, but were able to connect the IMU to the computer and gather some data. The default sample rate was 207 samples per second. Gyro-View has a real time graph, which displays all 6 measured values: 3 angular rates, and 3 accelerations. We tested moving the IMU in all directions and were able to see the lines move as expected. We took about a couple minutes of data with the IMU just sitting on the table to see what kind of errors we would get. Each trial requires a lot of data to be stored in the text file. Here is a sample of some data.

Time (s)	Roll rate (deg/s)	Pitch rate (deg/s)	Yaw rate (deg/s)	X Accel (g)	Y Accel (g)	Z Accel (g)	Temperature (°C)	Timer counts
406.436	-0.42114	-0.88348	0.109863	-0.00467	-0.00458	0.997284	27.721	42381
406.4409	-0.37537	-0.86975	0.247192	-0.00559	-0.00339	0.998016	27.721	36281
406.4457	0.521851	-0.40283	-0.32959	-0.00522	-0.00403	0.997284	27.721	30185
406.4506	-0.06409	-0.27008	0.672913	-0.00559	-0.00485	0.997284	27.721	24086
406.4554	-0.13733	-0.0412	0.169373	-0.00559	-0.00339	0.997375	27.721	17987

We ran three trials and looked at the average roll, pitch, and yaw rates. See data below.

Trial01.txt

	Roll	Pitch	Yaw	Data Points
	25.47457	-1307.31	254.0772	3208
Avg Error	0.007941	-0.40756	0.079201	

Trial02.txt

	Roll	Pitch	Yaw	Data Points
	-28.299	-592.328	58.75848	1437
Avg Error	-0.01969	-0.4122	0.040889	

Trial03.txt

	Roll	Pitch	Yaw	Data Points
	320.8099	-3854.71	1460.88	15150
Avg Error	0.021176	-0.25444	0.096428	