Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

BORE ANGLE CALC

Rev:	**Change Description**	**By**	**Date**
1.0	First Draft	R. Mamich	20170817
1.1	Validated	R. Mamich	20170822
1.2	Review Comments Incorporated	R. Mamich	20171018
2.0	Add Inertial Heading Option	J. Vaz Carneiro	20230112

Contents

---

1 Model Description

The Bore Angle Calculation model uses vector mathematics in order to calculate the miss angle from a boresight on the spacecraft to its desired target. The model also calculates the azimuth angle of the boresight.

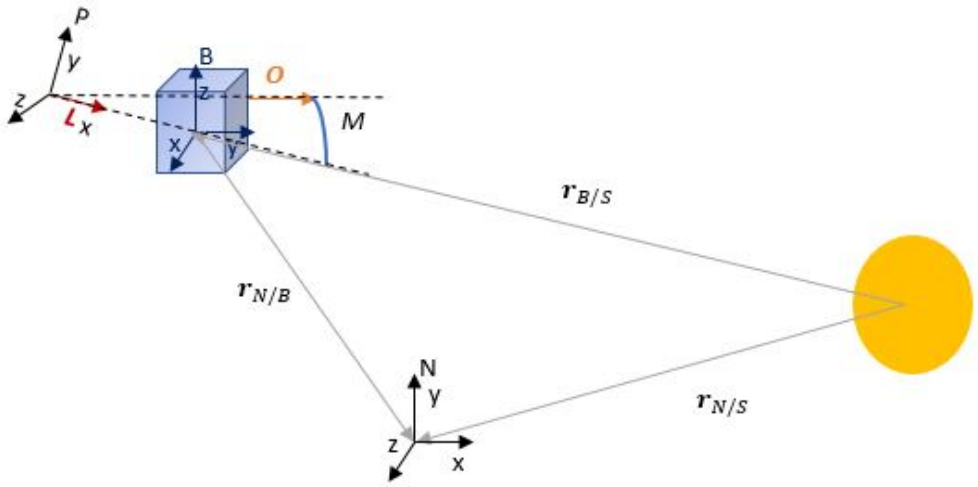

Fig. 1: All of the vectors and the miss angle are labeled here. $\boldsymbol{O}$ is the vector that represents the boresight of the instrument that the test is interested in. $\boldsymbol{L}$ is the direction that the spacecraft should be pointing its boresight in. $\boldsymbol{M}$ is the miss angle of the boresight to its desired direction. All of the $\boldsymbol{r}$ vectors are various position vectors designated by their subscripts.

The model begins by creating a direction cosine martix to represent the pointing frame from the inertial frame. Then the boresight vector is projected into this frame, and the miss angle and the azimuth angle are calculated using standard trigonometry. These calculations are performed in the following way:

First, the unit relative position vector of the spacecraft is calculated relative to the celestial body that the spacecraft is supposed to be pointing at. Using Equation 1. Where $\hat{\boldsymbol{r}}_{B/S}$ is the unit relative position of the spacecraft to the planet, $\boldsymbol{r}_{N/S}$ is the position of the celestial body in the inertial frame, and $\boldsymbol{r}_{N/B}$ is the position of the spacecraft in the inertial frame.

$$\hat{\boldsymbol{r}}_{B/S} = \frac{\boldsymbol{r}_{N/S} - \boldsymbol{r}_{N/B}}{|\boldsymbol{r}_{N/S} - \boldsymbol{r}_{N/B}|} \tag{1}$$

Next, the unit relative velocity of the spacecraft is calculated in the same manner.

$$\hat{\boldsymbol{v}}_{B/S} = \frac{\boldsymbol{v}_{N/S} - \boldsymbol{v}_{N/B}}{|\boldsymbol{v}_{N/S} - \boldsymbol{v}_{N/B}|} \tag{2}$$

Then, direction cosine matrix from the inertial frame to the pointing frame is constructed. The first row of the direction cosine matrix is set to be the unit relative position vector calculated in the first step. The last row of the direction cosine matrix is set to be the result of the cross product of the relative unit position vector with the cross product of the relative position vector with the relative velocity vector. Then, the second row of the direction cosine matrix is set to be the cross product of the first and last row of the direction cosine matrix, as show in Equation 3

$$[\boldsymbol{PN}] = \begin{bmatrix} \hat{\boldsymbol{r}}_{B/S}^T \\ ((\hat{\boldsymbol{r}}_{B/S} \times (\boldsymbol{r}_{B/S} \times \boldsymbol{v}_{B/S})) \times \hat{\boldsymbol{r}}_{B/S})^T \\ (\hat{\boldsymbol{r}}_{B/S} \times (\boldsymbol{r}_{B/S} \times \boldsymbol{v}_{B/S}))^T \end{bmatrix} \tag{3}$$

Once the direction cosine matrix has been created, the direction cosine matrix from the body frame to the pointing frame is created by multiplying the direction cosine matrix from the body frame to the inertial frame and the inertial frame to the pointing frame together. Then the boresight vector, $\boldsymbol{O}$ (which is shown in Figure 1), is projected into the pointing frame by dotting the new direction cosine matrix with the boresight vector in the body frame. This is shown in Equation 4.

$${}^{\mathcal{P}}\boldsymbol{O} = [\boldsymbol{PB}] \cdot {}^{\mathcal{B}}\boldsymbol{O} \tag{4}$$

From there, the boresight vector is multiplied by the baseline vector in the pointing frame. Where the baseline vector in the pointing frame is defined as $\boldsymbol{L} = \hat{\boldsymbol{i}}$. Then the product of the boresight vector and the baseline vector is passed through the arccosine function to get the boresight miss angle. As shown in Equation 5.

$$M = \arccos(\boldsymbol{O} \cdot \boldsymbol{L}) \tag{5}$$

Alternatively, the basline vector can be defined in the inertial frame instead of using the pointing frame defined above. In this case, the basline vector is projected onto the body frame first, and then the same equation in 5 is used to compute the miss angle.

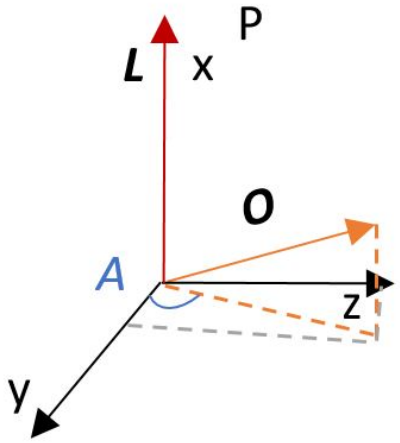

Fig. 2: The azimuth angle, A is shown here with respect to the pointing coordinate frame and the vectors used to construct the azimuth angle.

The azimuth angle is calculated by passing the product of the last two components of the borsight vector. This is shown in Equation 6 and in Figure 2.

$$A = \arctan\left(\frac{\boldsymbol{O}_{\hat{k}}}{\boldsymbol{O}_{\hat{j}}}\right) \tag{6}$$

For the case where the inertial heading is used, the azimuth angle is undefined and set to zero.

2 Model Functions

- **Compute Celestial Output Data**: This method computes the output structure for messaging. The miss angle is absolute distance between the desired body point and the specified structural vector. The azimuth angle is the angle between the y pointing axis and the desired pointing vector projected into the y/z plane.
- **Compute Inertial Output Data**: This method computes the output structure for messaging. The miss angle is absolute distance between the desired body pointing vector and the specified inertial heading vector.

3 Model Assumptions and Limitations

When the miss angle goes to 0° the azimuth angle is ill defined. The model will default to setting the azimuth angle to 0°. This also occurs when an inertial heading is used instead of the celestial body.

The module defaults to using the celestial body to compute the miss and azimuth angles. This means that when a celestial body message is provided along with an inertial heading, it will output the results using the celestial body data.

4 Test Description and Success Criteria

In order to test the model, the boresight vector on the spacecraft, and the orientation of the spacecraft were varied to be in each quadrant of the body frame. There were a total of 16 different tests performed in this manner. Eight different boresight vector placements (holding the spacecraft orientation constant), and eight different spacecraft orientations (holding the boresight vector constant). There was a 17th test performed in order to check the azimuth angle when the miss angle is zero. For every combination

of parameters, the pointing reference frame was calculated and compared to the simulated pointing frame, and using that, the miss angle and azimuth angle were calculated.

The boresight vector was calculated in the same manner that it was calculated in the model using numpy libraries to do so. It was then compared to the model's boresight vector. The test was deemed a success if the boresight vector was withing 1E-10 of the model's boresight vector using the unitTest-Support script. It should be noted that the boresight vector that the user passes to the model doesn't need to be a unit vector because the model will normalize the vector.

The miss angle was calculated in two separate ways. The first method mirrored the module's method. Just as in the module, a direction cosine martix was created to represent the pointing frame from the inertial frame. Then the boresight vector was projected into this frame, and the miss angle was calculated using standard trigonometry. The key difference in the first method of validation is that the validation used the python numpy library primarily rather than the RigidBodyKinematics.py file.

The second method used the existing inertial reference frame in order to calculate the miss angle of the boresight. In this method, the baseline vector was projected into the inertial frame. Then just as the first, the miss angle was calculated using standard trigonometry.

The second method relied on the direction cosine matrices created in the first method. That being said, the first operation in the second method was to calculate the position vector of the spacecraft in the inertial reference frame. Then the baseline vector was projected into the inertial frame using the direction cosine matrix from the inertial frame to the pointing frame. Then the position vector in the inertial frame was dotted with the baseline projection. Just as in the first method, the product of the two vectors was passed through the arccosine function in order to calculate the miss angle of the boresight.

Once the miss angle calculations were complete, the calculated miss angles were compared to the miss angle pulled from the mission simulation. If the calculated miss angles were withing 1E-10 of the simulation miss angle, the test passed.

Then the azimuth angle was calculated using the same method as in the model. Again, it should be noted that the standard python numpy library was used in the validation calculation. The test passed when the calculated azimuth angle was within 1E-10 of the simulation azimuth angle.

Table 2: Tolerance Table (Absolute)

Parameter	Tolerance
Boresight Vector	1E-10
Miss Angle	1E-10
Azimuth Angle	1E-10

As for the case where the inertial heading is used, a similar procedure is applied. In this case, only the miss angle is verified. First, the inertial heading is computed in the body frame, and then the miss angle is computed using the body heading frame as well. Only the miss angle is checked, since the azimuth is ill-defined.

Similarly to the celestial body test case, multiple different inputs are used to parameterize the test. The different inputs consist of different inertial headings, along with different spacecraft attitudes. The body heading vector is assumed to always be pointing in the body-frame x direction.

5 Test Parameters

It should be noted that the boresight vector passed to the model does not need to be a unit vector because the model will normalize the vector.

Table 3: Table of test parameters

Test	Boresight Vector	$\sigma_{B/N}$	Test	Boresight Vector	$\sigma_{B/N}$
1	$\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	9	0.0, 0.0, 1.0	-0.079, 0.191, 0.191
2	$-\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	10	0.0, 0.0, 1.0	-0.261, 0.108, 0.631
3	$\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	11	0.0, 0.0, 1.0	0.261, 0.108, -0.631
4	$-\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	12	0.0, 0.0, 1.0	0.079, 0.191, -0.191
5	$\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	13	0.0, 0.0, 1.0	0.079, -0.191, 0.191
6	$-\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	14	0.0, 0.0, 1.0	0.261, -0.108, 0.631
7	$\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	15	0.0, 0.0, 1.0	-0.261, -0.108, -0.631
8	$-\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}$	0.0, 0.0, 0.0	16	0.0, 0.0, 1.0	-0.079, -0.191, -0.191
			17	1.0, 0.0, 0.0	0.0, 0.0, 0.0

6 Test Results

Table 4: Pass or Fail Table

Test	Pass/Fail	Test	Pass/Fail
1	PASSED	9	PASSED
2	PASSED	10	PASSED
3	PASSED	11	PASSED
4	PASSED	12	PASSED
5	PASSED	13	PASSED
6	PASSED	14	PASSED
7	PASSED	15	PASSED
8	PASSED	16	PASSED
		17	PASSED

7 User Guide

The module can use either a celestial body message or an inertial heading set by the user. It defaults to using the celestial body message if both the message and the heading are configured.

A common setup for both cases contains:

- `BACObject = boreAngCalc.BoreAngCalc()`: Construct the boreAngCalc module
- `BACObject.ModelTag = "solarArrayBoresight"`: Set the model tag
- `BACObject.boreVec_B = boreVec_B`: Set the body frame boresight vector assuming it is assigned to variable `boreVec_B`
- `BACObject.scStateInMsg.subscribeTo(scMsg)`: Attach the spacecraft's state message from `scMsg`
- `TotalSim.AddModelToTask(unitTaskName, BACObject)`: Attach the module to a task

If using the celestial body message, then:

- `BACObject.celBodyInMsg.subscribeTo(celBodyMsg)`: Attach the celestial body message of type `SpicePlanetStateMsg()`

If using the inertial heading, then:

- `BACObject.inertialHeadingVec_N = inertialHeading`: Define the inertial heading