Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-coarseSunSensor

MODULE TO APPLY A PRESCRIBED FORCE OR TORQUE ONTO A RIGID BODY

Prepared by	H. Schaub

Status: First Version
Scope/Contents
This module defines both individual Coarse Sun Sensor (CSS) modules, as well as an array or constellation of CSS devices. The CSS modules determine an ideal cosine response behavior, and can be corrupted through a Kelly-curve and gaussian noise. The CSS response can also be reduced due to being partially or fully in a planet's shadow.

Rev:	**Change Description**	**By**
v1.0	Initial document	H. Schaub

Contents

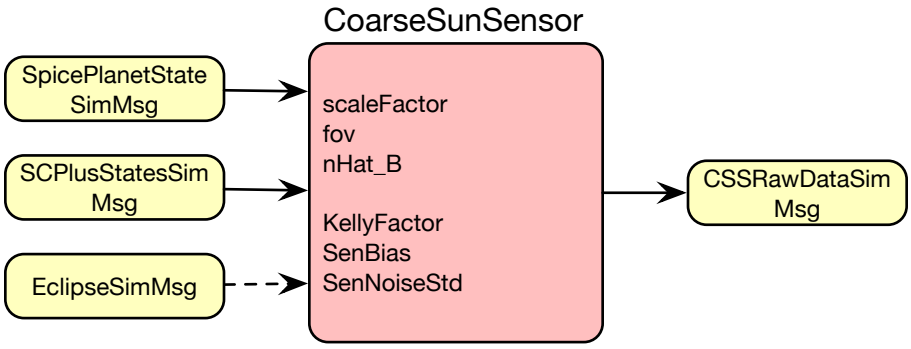

Fig. 1: Illustration of the CoarseSunSensor() modulej

1 Introduction

This document describes how Coarse Sun Sensor (CSS) devices are modeled in the Basilisk software. Each CSS is modeled through a nominal cosine response relative to the sunlight incidence angle. This response can then be corrupted, or reduced due to being in a planet's shadow. It is possible to add individual CSS sensors to the BSK evaluation stack. However, it is typically simpler to use the `CSSConstellation()` class to store a list of CSS units which are updated as a group, along with a unique CSS array output message.

2 Single CSS module

2.1 I/O Messages

First, let us discuss the input and output messages of the individual CSS sensor module. The two required input messages are of the type `SpicePlanetStateSimMsg` and `SCPlusStatesSimMsg`. The first message is used to determine the sun's location relative to the inertial frame ${}^{\mathcal{N}}\boldsymbol{r}_{\odot/\mathcal{N}}$. The second message is used to read in the spacecraft inertial position vector relative to the same inertial frame ${}^{\mathcal{N}}\boldsymbol{r}_{\mathcal{B}/\mathcal{N}}$. Finally, the last message is optional. If it is connected, it provides the sun shadow parameter indicating if the spacecraft is in a planet's shadow.

The output message of an individual CSS unit creates a message containing the simulated CSS sensor.

2.2 CSS Signal Simulation

To evaluate the sun heading vector $\boldsymbol{s}$, the satellite and sun position vectors are used.

$$^{\mathcal{N}}\boldsymbol{s} = {}^{\mathcal{N}}\boldsymbol{r}_{\odot/\mathcal{N}} - {}^{\mathcal{N}}\boldsymbol{r}_{\mathcal{B}/\mathcal{N}} \tag{1}$$

After normalizing this vector to $\hat{\boldsymbol{s}}$ and mapping $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ to $[BN]$, it is mapped into body frame components through

$$^{\mathcal{B}}\hat{\boldsymbol{s}} = [BN]\,{}^{\mathcal{N}}\hat{\boldsymbol{s}} \tag{2}$$

The CSS sensor unit normal vector is given by $^{\mathcal{B}}\hat{\boldsymbol{n}}$ in body frame components. The normalized cosine sensor signal $\hat{\gamma}$ is thus determined through

$$\hat{\gamma} = \hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{s}} = \cos\phi \tag{3}$$

where ϕ is the CSS sunlight incidence angle. This is the normalized CSS signal where 1 is returned if the sensor is looking straight at the sun. If the sensor axis $\hat{\boldsymbol{n}}$ is more the field of view half-angle (set through `fov`) from the sun axis, then a 0 signal is simulated. This `fov` variable is the angle from $\hat{\boldsymbol{n}}$ beyond which the CSS signal is set to zero.

Let p_s be the local solar shadow parameter. If the spacecraft is outside of a planet's shadow, this value is 1. If it is within the shadow, then it is $0 \leqslant p_s < 1$. The shadow adjusted CSS signal is thus computed as

$$\hat{\gamma}_s = \hat{\gamma} p_s \tag{4}$$

To simulate CSS signal corruptions, then a Kelly curve corruption, a bias and a gaussian noise can be included. The normalized bias is set through `SenBias`, while the normalized noise is set through `SenNoiseStd`. Let $\boldsymbol{p}_n$ be the normalized sensor noise. Note that this noise levels is non-dimensional relative to the unit cosine curve.

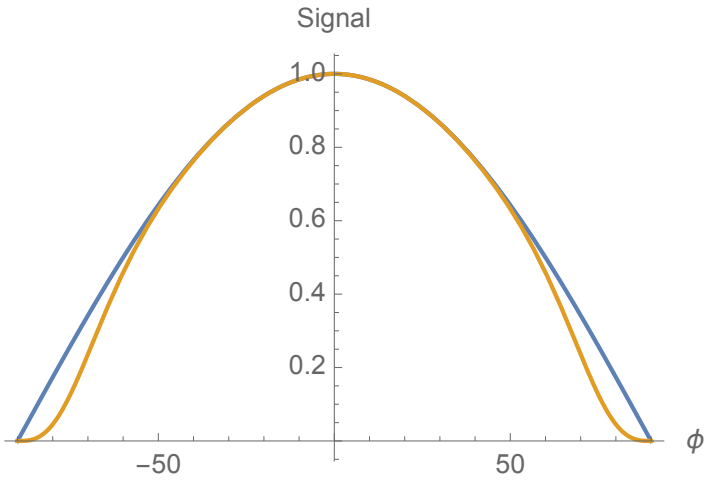

Fig. 2: Kelly distortion illustration with $\kappa = 0.1$

The Kelly corruption parameter allows for the CSS signal to pinch towards zero for larger incidence angles as illustrated in Figure 2. The amount of signal distortion is set through κ, where the Kelly factor p_κ is then computed as

$$p_\kappa = 1 - e^{-\hat{\gamma}_s^2/\kappa} \tag{5}$$

The corrupted normalized CSS signal is then

$$\hat{\gamma}_c = (\hat{\gamma}_s + p_a)p_\kappa + p_n \tag{6}$$

At this point the signal shape is correct, but not the scale. The final step is to scale. Using the `scaleFactor` $= \alpha$ parameter, the CSS signal is now scaled to a dimensional value where the peak is not 1, but rather `scaleFactor`.

$$\gamma = \hat{\gamma}_c \alpha \tag{7}$$

2.3 Setting the CSS Unit Direction Vector

The unit vector of each CSS device is set through the $\mathcal{B}$ frame vector representation

$$\texttt{nHat\_B} \equiv {}^{\mathcal{B}}\hat{\boldsymbol{n}} \tag{8}$$

It is possible to set these vectors directly. However, there are some convenience functions that make this process easier.

Multiple CSS devices are often arranged together on a single CSS platform. The orientation of the body-fixed platform frame $\mathcal{P}$ is given through $[PB]$. In the CSS module, the DCM is specified through the variable `dcm_PB`.

2.4 Module Parameters

This section describes the various module parameters that can be set via Python to configure the CSS unit. The platform frame is

2.4.1 `theta` Parameter

This is the CSS azimuth angle relative to the platform frame.