Orbital Simulation Tools for CubeSat Missions

Sydney Whilden[a], W. Conor McFerren[a], Dr. David Cotten[a,b]
sydney.whilden25@uga.edu, wcm73452@uga.edu, dcotte1@uga.edu

[a]Small Satellite Research Laboratory [b]Center for Geospatial Research

Problem

Scheduling small satellite operations involves balancing competing requirements. One of the most popular and robust tools available for orbital simulation is Systems Tool Kit (STK). While STK's graphic user interface makes many of its capabilities available even to users who cannot code, the GUI imposes some limits on its functions. Additionally, STK's Connect commands require extremely particular syntax, and can be tricky to work with. Creative solutions are needed to:

- Consistently reproduce complex scenarios while limiting difficulty for end user
- Efficiently handle Connect command syntax
- Output simulation data in maximally useful format

Methods

We handled the problem by creating the foundations of a GUI-free Python-based STK user interface. It builds on top of the provided STK Python API. Some of the first steps were:

- Determined reliable way to start an instance of STK using Python API
- Created simple ways to access object library
- Wrote test scripts for simple scenarios such as those involving surface temperature and Sun angle
- Created subroutines to expedite actions in test scripts and to reformat output data into simple CSVs
- Used subroutines to write new test scripts of more complex scenarios

JUSTFORSHOW.PY

```python
def FormatName(app, obj_class, raw_name, purpose, parent_name, parent_class):

    """
    Take a string form of a component name and reformat it to a usable format
    for a variety of purposes.

    Parameters:
        app (stkhelper): Current instance of STK.
        obj_class (str): The class the named object belongs to, e.g.
            'Satellite'.
        name (str): The object's given name.
        purpose (str): Can be one of several things.
            'Plain' : The plain name with no parent object included, but with
                spaces in the name replaced with underscores.
            'Path' : The object pathway, e.g. */Satellite/MOCI
            'Full' : The full name of the object including the parent, but
                not formatted to be a scenario pathway (i.e., no asterisk)
        parent (True or False): Whether or not the object has a parent object

    """
    # determine parent object, if any
    if parent_name=='None' and parent_class=='None':
        parent_name = ''
        parent_class = ''
    elif parent_name!='None' and parent_class!='None':
        parent_name = parent_name.replace(' ','_') + '/'
        parent_class = parent_class.replace(' ','_') + '/'
    else:
        print('''If the object has a parent, both the parent\'s name and class
              must be given. If the object has no parent, pass 'None' for the
              parent object's name and class.''')

    # replace punctuation, if any
    if '.' or '(' or ')' in raw_name:
        raw1 = raw_name.replace('.','') # remove periods
        raw2 = raw1.replace('(','') # remove left parenthesis
        name = raw2.replace(')','') # remove right parenthesis
    else:
        name = raw_name
```

Figure 1: Example of a function to format object names in STK Connect language using more compact Python code.

Case Study: Sun Exclusion

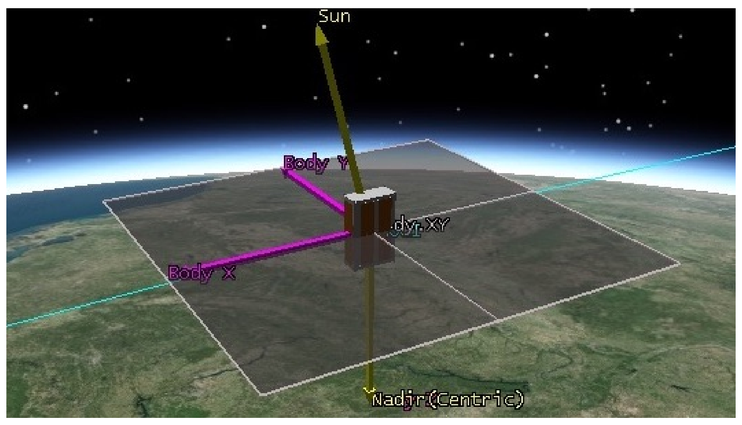

Figure 2: Satellite body axes and Sun-satellite vector.

Problem Without a baffle, the star tracker has a Sun exclusion angle of 90°. If the star tracker encounters the Sun during a science or maintenance operation, important ADCS maneuvers could be hampered. It is necessary to find a method to repeatedly determine the likelihood of the star tracker's encounters with the Sun coinciding with other operations.

Approach

- Using STK, calculate times when sun intrudes on exclusion FOV
- Use STKHelper to make custom report comparing intrusion times with other variables, such as beta angle
- Engine to run scripts repeatedly and quickly, and automatically output usable data products

Python Package

STKHelper provides an easy interface for using STK with Python. STK's Python API's focus on usability with many programming languages results in the sacrifice of its object-oriented nature, complicating syntax and data return. STKHelper combats this by establishing classes and formatting cumbersome syntax behind the scenes. This greatly reduces the time input required of the user.

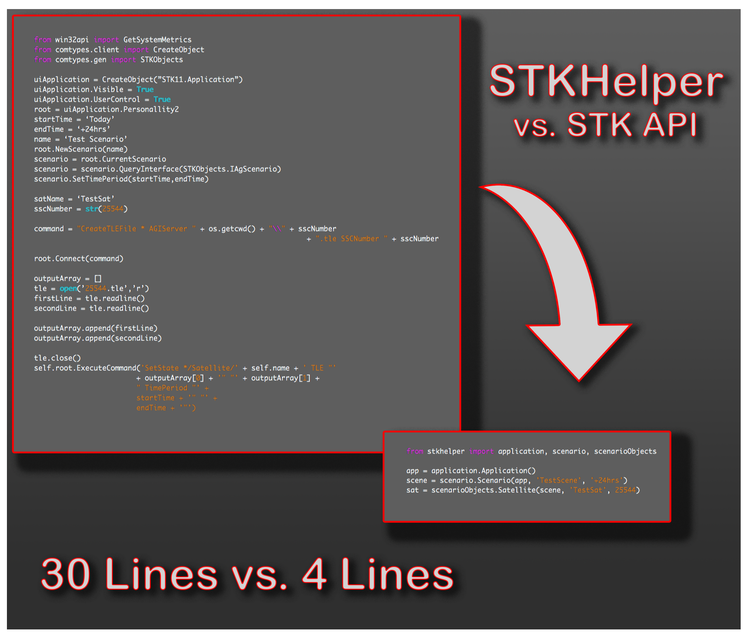

Figure 3: Example of STKHelper shortening lengthy code.

Case Study: Results

About 14% of sun encounters occur at a sun elevation angle greater than 30°, which is the lower limit for imaging. Thus, the chance of an encounter during an image attempt is no higher than (about) 14%.

Sun Elevation

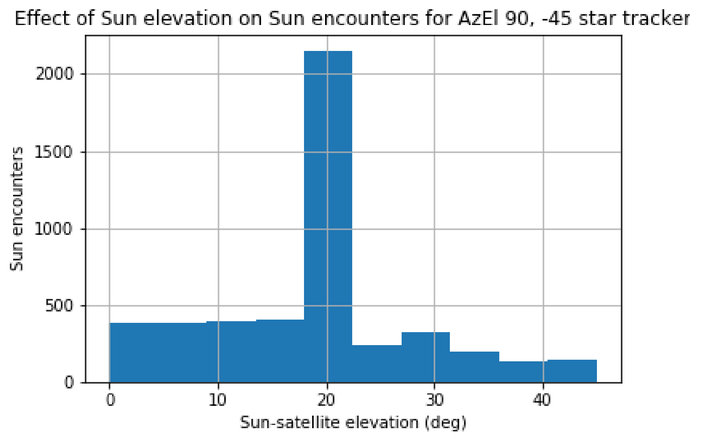

Figure 4: Sun elevation angle relative to the satellite during each encounter, for +X face.

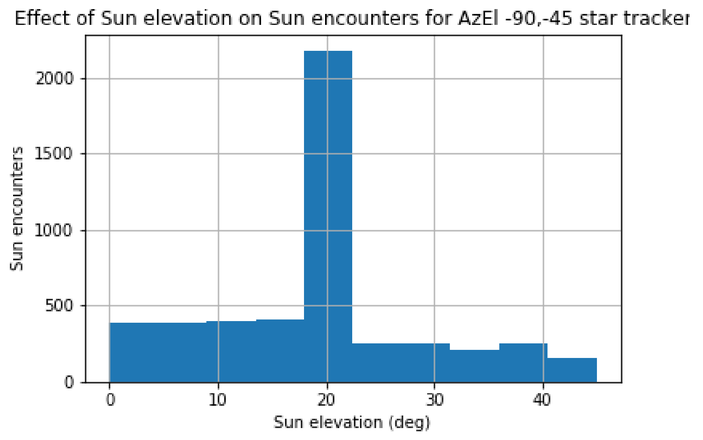

Figure 5: Sun elevation angles at time of encounter for the -X face.

Beta Angle

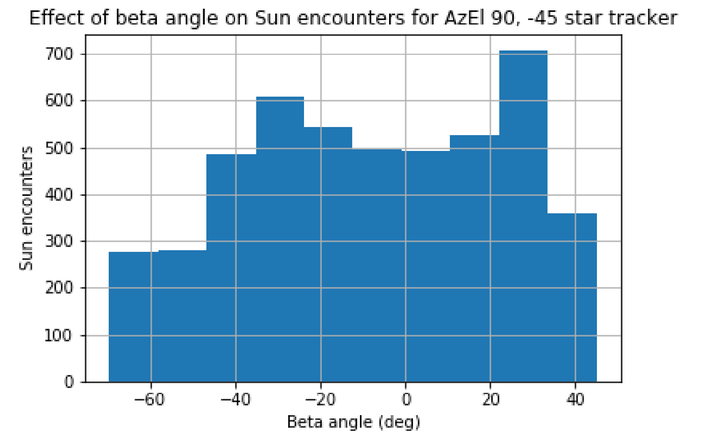

Figure 6: Relationship between beta angle and frequency of sun encounters for the +X face.

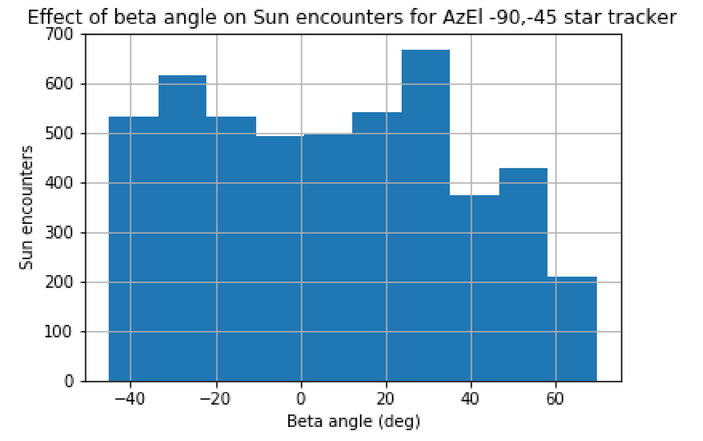

Figure 7: Relationship between beta angle and sun encounter frequency for the -X face.

Acknowledgements

Thanks to S. Godfrey Hendrix for fixing STK licensing issues, Michael Ely for solar panel data and help with COLLADA files, Kaelyn Deal for help with COLLADA files, Megan Arogeti, Jackson Parker, Caleb Adams, and the people who answered Conor's millions of emails to AGI support

References

1. Leicher, Grable, Adams, King, Copenhaver, Hendrix, Parker, Whilden (2019). MOCI Concept of Operations (v6.0.1). Internal document.
2. Whilden, S. (2019) Star Tracker Sun Encounter Notes (v0.0.5). Internal document.