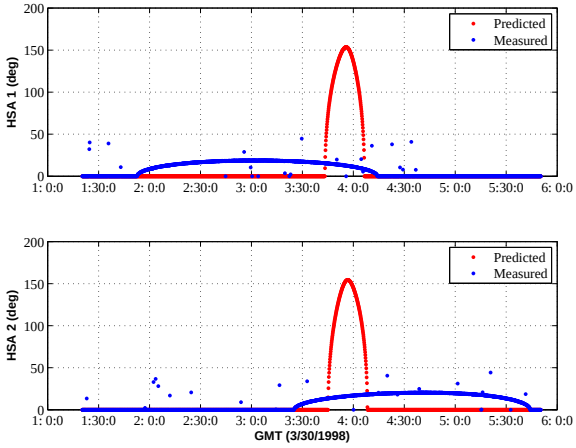
Princeton SATELLITE SYSTEMS

Spin Axis Attitude Determination Module



Spin Axis Attitude Determination Module

Printing History:

Princeton Satellite Systems, Inc.
33 Witherspoon Street
Princeton, New Jersey 08542

Technical Support/Sales/Info: http://www.psatellite.com

CONTENTS

CHAPTER 1

INTRODUCTION

The Spin Axis Attitude Determination Module (SAAD), for use with the Spacecraft Control Toolbox Professional Edition, provides you with a complete, integrated set of tools needed to perform spin-axis attitude determination using horizon sensor and sun sensor measurements during spacecraft transfer orbit.

The SAAD Module provides a comprehensive set of features and capabilities including:

- Differential-corrector, Conjugate Gradient and Nelder-Meade attitude determination algorithms
- Iterated extended Kalman Filter for real-time applications
- Cone intercept, and chordwidth/dihedral angle attitude determination methods
- Singular value decomposition least squares solver for ill-conditioned data
- Data quality evaluation tools
- Horizon sensor dynamics models
- Graphical user interface that frees you from coding

SAAD is available in GUI based and script based packages. The GUI based tools (with the GUIs shown on the following pages) are recommended for flight operations. The script based version is suitable for R&D use.

The SAAD Module can process any combination of data that is available on your system. It accepts horizon sensor inputs as either leading and trailing edge times or as dihedral angles. Midscan, leading or trailing dihedral angles are handled with equal ease. The data can even be processed without dihedral angles or without chordwidths. This feature was useful on a recent mission when a temporary glitch made the leading edge times, and therefore the chordwidths, unavailable. We were able to continue attitude determination using only the sun angle and the trailing edge dihedral angles.

1.1 Point and Click Interface

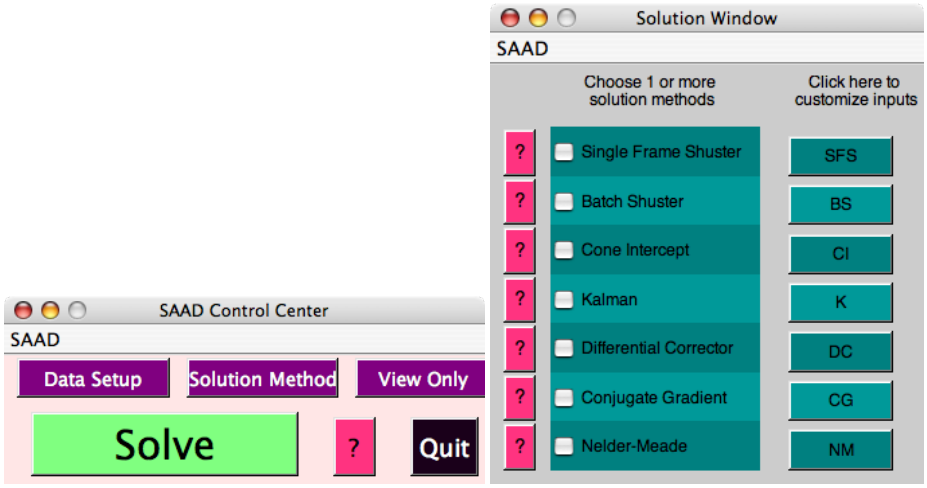
Setting up your data for spin-axis attitude determination has never been easier. The SAAD GUI interface makes it easy to:

- include multiple data files - there is no limit on the number of files that can be processed at once
- select and customize some or all of the data conditioning procedures
- include biases on measurements and sensor positions, including HSA time delays

- view measured and predicted data before solving
- select one or more methods for determining the attitude
- customize the solution methods
- save cases so that analysis can be duplicated later
- get help on any part of the package

Figure 1-1 shows the main window and solution method selection window from the graphical user interface. All of the solution methods are available from this window, along with help functions.

Figure 1-1. Interactive SAAD package



SAAD makes extensive use of the Matlab graphics capabilities, which makes analyzing your data an easy task. Plots of measurement data make evaluating data quality and effectiveness of data culling a snap. You can also see how well the measured data matches predictions before and after solutions, as in the plot below. Solutions are also presented numerically (as errors and residual statistics) for all of the solution methods.

1.2 Use During Mission Operations

There is no reason to hand code your own algorithms since you can easily import your spacecraft telemetry data directly into SAAD for processing. A data loading function that handles data-only ASCII files is included with the package. A detailed input-output specification is provided for creating loading functions for other file types. In addition, the tutorial provides complete information on how to manipulate binary files. We give examples of how to access formatted text-and-data ASCII files too!

We've used this package on four missions for two different companies and three different types of spacecraft with excellent results! On the most recent mission our predictions, from all of our algorithms, matched post apogee burn attitude predictions to within two tenths of a degree.

CHAPTER 2

TUTORIAL

2.1 Introduction

In this section we will review some concepts needed to understand use of the SAAD Module. Please see the *Spacecraft Attitude and Orbit Control* for additional theoretical information on spin axis attitude determination.

2.1.1 Coordinate frames

Attitude determination is the process for determining the orientation of a spacecraft with respect to a reference frame. Common reference frames are

- the earth-centered frame, with one axis aligned with the nadir vector and a second normal to the orbit plane. This frame rotates at orbit rate.
- the local vertical-local horizontal frame (LVLH), with one axis aligned with the local perpendicular to the earths surface. This frame rotates at orbit rate and is the same as the earth-centered frame when the spacecraft is in an equatorial orbit.
- the earth centered inertial frame (ECI), with the +x-axis along the J2000.0 mean vernal equinox, the +z-axis along the J2000.0 mean north pole and the +y-axis forming an orthogonal right-hand set. This frame is also known as the Equator of J2000. By using "mean" we imply that we are ignoring the nutational motion of the earths pole.

2.1.2 Spinning Spacecraft

Spacecraft are spun in transfer orbit to provide passive stability against external environmental torques such as solar pressure, aerodynamic drag, etc. and to provide stability against disturbances due to the delta-v engines needed to put the spacecraft in its final orbit. This is particularly important when firing large solid rockets that can produce large torques transverse to the spin axis.

A second advantage is that a spinning spacecraft can use the rotation of the spacecraft to scan for targets, such as the sun and the earth, for use in attitude determination. The sensors can be fixed to the body and as a consequence are simpler, less expensive and more reliable than scanning sensors or imaging sensors. The relatively low cost of such sensors is particularly important when one considers that transfer orbit only occupies a small fraction of a typical satellites life.

This module assumes the use of a single axis fixed sun sensor and a fixed horizon sensor. The sun sensor boresight is nominally perpendicular to the spin-axis of the satellite. It has two outputs. The first indicates when the sun passes

through its field-of-view which is aligned with the spin-axis. The sensor will output a signal known as the command eye pulse to indicate that this has happened. The second output gives the angle of the sun with respect to the boresight. When the sun projection passes through the boresight vector it will trigger the command eye pulse (CEP) as long as the sun vector is within the field of view of the sensor.

The sun vectors orientation in the inertial frame is known from the sun ephemeris. Consequently, once the angle is measured, the spin axis vector must lie on a cone with the sun vector along the cones axis and the apex of the cone at the origin of the reference frame. The earth is the most convenient second source.

The earth horizon sensor actually gives two measurements. One is a vector to the leading edge and the other to the trailing edge. If we combine this with the sun measurement it this should give three measurements and an unambiguous attitude. Clearly just a measurement of the earth pulse is inadequate for this purpose but the command eye pulse (CEP) gives us a reference that can be used for this purpose.

Unfortunately, this does not completely solve the problem. For one thing, the earth horizon is poorly defined. For another, the two earth edge vectors are only and earth angular width apart (about 17.5 deg at geosynchronous altitude) and the geometric uncertainty is large. The only way to deal with the measurement uncertainties is to take a lot of measurements at different points in the orbit thus providing new geometric information with each measurement. The measurements can be processed using either a batch algorithm or sequentially as they come are taken.

2.2 The attitude solution

The SAAD Module has several methods for solving for the attitude. They can be broken into two categories:

- Closed Form
- Numerical

2.2.1 Closed Form

Two methods of closed form solutions are provided. One is the cone intercept method. Given the angles between two known vectors and the spin axis it is possible to find one or two attitudes that fit those angles.

There are two possible solutions which are where the cones intercept. The correct solution is found by taking multiple measurements and observing which solution doesnt change very much. The image solution will tend to jump around. This method only requires chordwidths and sun angles. The dihedral angle is not required.

This is mechanized by the routine `ConeInt`.

A second method takes advantage to the dihedral angle to measure the attitude. The dihedral angle eliminates the ambiguity inherent in the cone intercept method. It is implemented in `BShuster` and `SFShustr`.

The closed form solutions solve only for right ascension and declination and do not estimate unknown biases. Consequently they are best used to produce initial estimates of the attitude or to check the results from the numerical methods.

For a demo of the closed form methods see `TCF`. Sample attitude data is generated using `ADGen`.

2.2.2 Numerical

Three classes of numerical algorithms are provided. They are

1. Batch Least Squares

Differential Corrector - `DiffCorr`
Conjugate Gradient/Steepest Descent - `ConjGrad`

2. Recursive Least Squares
   Iterated Extended Kalman Filter - `KFSAAD`
3. Nonlinear Equation Solver
   Nelder-Meade - `fminsearch`

for a total of four distinct methods. The theory behind these methods is discussed in *Spacecraft Attitude and Orbit Control*. For a demo of the numerical (batch) methods see `TBAD`.

2.3 Reading in binary data

The following code reads in binary data into formatted numbers.

```
[fid,message] = fopen(file,'r');
if fid == -1,
  error(message);
end

a = fread(fid);

header = a([1 3 5 7 9])';
satID = header(1);
year = header(2);
month = header(5);
aveType = header(4);
```

By default fread returns 8 bit characters. Consequently, each element of a has 8 bits from the file.

2.4 Reading in ascii data

It is often necessary to extract attitude data from formatted ASCII text files. The most robust way to do this is to copy the file into an array of characters and search for keywords that show you where the attitude data is located. For example, assume the data file has the form:

```
SPACECRAFT ID: 7
DATE 11/7/93
ATTITUDE DATA: 123.4 123.3 122.2 123.6 122.9
ATTITUDE DATA: 122.4 123.3 123.1 123.6 122.7
```

The attitude data can then be read out using

```
[fid,message] = fopen(fileName,'r');

if fid == -1
  error(message);
end

buffer = fscanf(fid,'%1c');
j = findstr(buffer,'ATTITUDE␣DATA:␣');

for k = 1:length(j)
  q(k,:) = sscanf(buffer((j(k)+15):(j(k)+43)),'%f');
end
```

Each row of `q` will contain the numbers in each line. `LoadAtt` is a function template for reading ASCII text files.

2.5 Solving for the attitude

This section gives a methodical approach to attitude determination. The script `TKalAD` provides a complete attitude determination system that was used during actual missions.

2.5.1 Data collection

Determine whether the sun or the moon will be in the horizon sensor field of view by using `BObjIntF` with your current best estimate of the attitude.

Collect horizon sensor and sun sensor data for one complete apogee pass. The data collection should be as near to continuous as possible. Do not worry about getting short chordwidths at this point. You can edit them out later. The horizon sensor output should be in the form of the times for the rise and fall of the horizon pulse. Edit out scans corrupted by the sun or moon. Convert these to chordwidths and midscan dihedral angles.

It is important to collect HSA data when the chordwidths are changing rapidly as a function of time. This means that you should try and get chordwidths as small as seven degrees. This is very important. Chordwidth data taken near the peaks in a given pass gives very little useful information.

It is also necessary to remove data that does not fit the models in the attitude determination software. Small horizon sensor chords (less than seven degrees) should be eliminated for this reason.

2.5.2 Data Validation

Use `ADComp` to compare the measurements with values based on the current best estimate of attitude. `ADComp` will plot the measured data against the expected data. They should agree reasonably well. Check chordwidths, dihedral angle and sun angles. Look out for the following:

- Measured sun angles that do not follow the sun angle trend for the time of year. For example if the sun angle trend is negative and you observe the sun angle increasing in a file it indicates that the sun data is corrupted. This can happen when the sun is behind the earth and sun light is scattered off of the edge of the earth.
- Short chordwidths. This can be due to the sun reflecting off of the oceans or polar ice.

Corrupted data should be removed from the solution.

2.5.3 Orbit Data

Get a file with an orbit position vector for each measurement. Given an initial position vector you can use `RVOrbHF` for the same purpose. It is usually sufficient to get the orbital elements at apogee and propagate them using `RVOrbGen`.

2.5.4 Eliminate Outliers

Use the function `Outliers` to eliminate spurious data. Eliminate any horizon sensor data with chordwidths less than seven degrees. `ADCull` removes noisy data by comparing chordwidths with chordwidths generated by the model used in the toolbox and eliminates those that differ from the modeled chordwidths by more than a selected threshold. `ADCull` will only work properly if you have a good orbit solution and if your attitude solution is not too far off. If the difference between the modeled and measured chords is large on average, `ADCull` will not remove noisy data. Always check the data visually with both your initial attitude guess and with your new solution.

2.5.5 Correct for Sensor Dynamics and Radiance Variations

Your sensor manufacturer will give you corrections for the earths radiance variations and the sensor dynamics. The latter will be a function of the spin rate of your satellite. Apply these corrections to the dihedral angles and chordwidths. `Radiance` can be used to determine the peak radiance as a function of latitude.

2.5.6 Closed Form Solutions

Use `BShuster` and `ConeInt` to get closed form solutions of right ascension and declination. The results should be nearly the same. If they are not you either have large dihedral biases, or your data is corrupted.

2.5.7 Least Squares Solution

Once you have an approximate solution use the least squares methods to refine the solution. First, determine your apriori covariance matrix for all of the states. The uncertainties in the biases can be obtained from the alignment data and accuracies. The uncertainty in right ascension and declination can be obtained from the closed form solution or the results of attitude determination on your last pass.

Use `ADPartls` to determine the sensitivity of the measurements to the states. If you see that one of the states (such as the cant angle) has little influence on the measurement you should eliminate it from the state vector.

Generate the measurement matrix and determine the condition number and rank. If the rank is less than the number of states or the condition number is high you must reduce the number of states. Look for rows and columns that are similar or ones that have very small diagonal elements and eliminate those, starting with the worst. Your a priori covariance matrix will tend to reduce the illconditioning but you must be careful not to try and solve for states that the current geometry prevents you from finding.

You can solve for sensor biases. Sensor biases can be due to both physical misalignments and modeling errors. The sun sensor bias can be caused by sensor misalignments or spacecraft coning. The horizon sensor cant biases can be caused by misalignments or spacecraft coning. The dihedral biases can be caused by unmodeled delays in the sensor. The chordwidth biases can be caused by radiance variations and altitude errors. All of the horizon sensor biases may include components due to modeling errors. This is not true about the sun sensor biases and large sun sensor biases should be viewed with skepticism.

Use `KFSAAD` first. Observe the variation in the estimates as a function of measurement sample. The attitude should converge smoothly. Look for any sudden jumps. If you see any check the data at those points and eliminate any bad data.

Use `DiffCorr` and `ConjGrad` to come up with solutions. Check all the statistical outputs of those functions. They should agree pretty well. If they dont you should review your data.

Finally use `fminsearch`. `fminsearch` is slow to converge but does not use the measurement derivative functions and is a good check on the previous results.

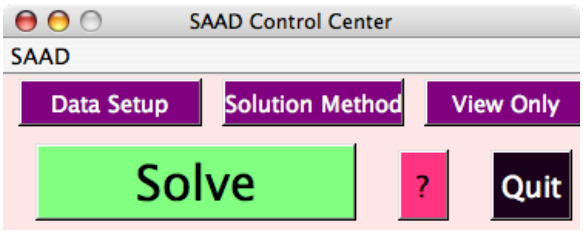
CHAPTER 3

SAAD GUI

3.1 Main window

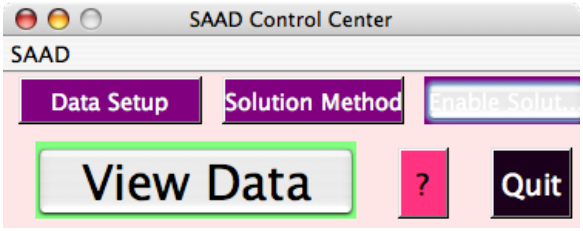
The main function for using the SAAD interface is `ISAAD`. This will open the main control window for the Interactive Spin Axis Attitude Determination (ISAAD) Package and populate it with default data.

Figure 3-1. SAAD Main GUI



Both Data Setup and Solution Window open new GUIs which will be discussed below. View Only is a toggle button which will change the Solve button to a View Data button, as seen in Figure 3-2. The toggle button now says Enable Solution. The Quit button quits the GUI and the ? button opens a help text window.

Figure 3-2. SAAD Main GUI after selecting View Only



The steps to using the GUI are:

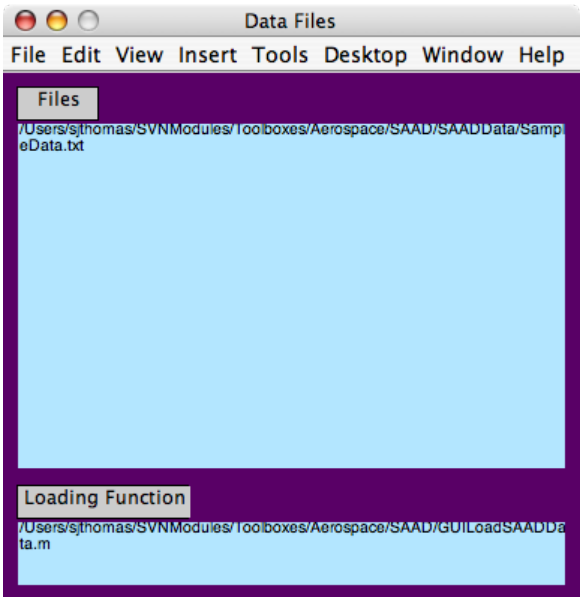
1. Click Data Setup and then enter the information in each of the blocks in the left-hand column of the Data Window. (The blocks on the right-hand side are optional.)
2. Click View Only and then View Data in the main control window.
3. If necessary, make changes to entries in the Data Window, and view the data again.
4. Click Solution Methods in main control window.

5. Choose one or more solution methods from the Solution Window.
6. Click Solve in the main control window.

Further help is available in each window by clicking the Help buttons, labeled with a question mark.

Sample data is included to allow you to run a demo. From the menu, choose Load and select `DemoSAADCase.mat` from the SAADData directory. This file utilizes the data file `SampleData.txt` and the data loading function `LoadSAADData`, further described below. When the data is loaded you will see the window in Figure 3-3 displaying the paths to the selected files, but the paths will be relative to your own toolbox installation.

Figure 3-3. Data files view



You can now view the data and solve using Nelder-Meade using the above instructions. When you view the data you should see the following plots.

Figure 3-4. Sample HSA chordwidths and residuals

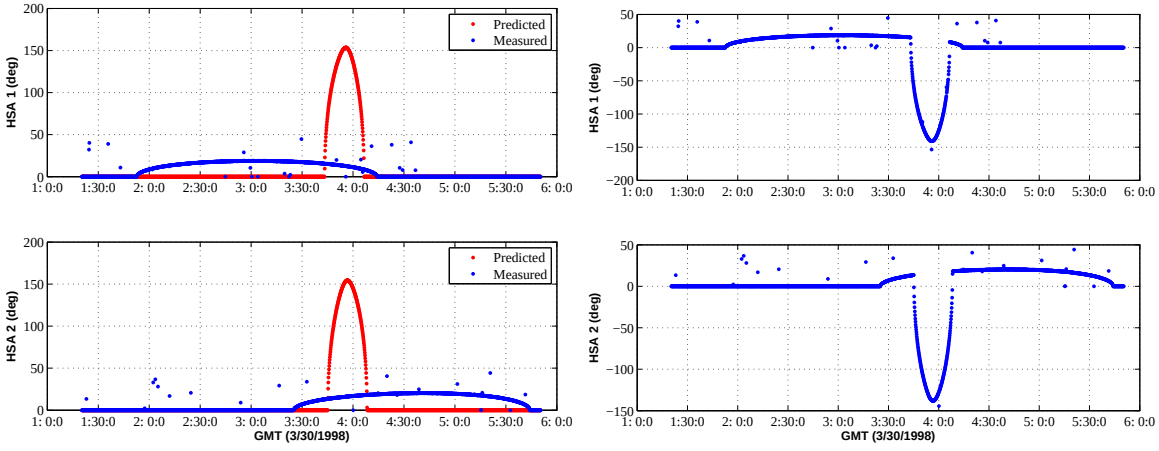


Figure 3-5. Sample HSA dihedral angles and residuals

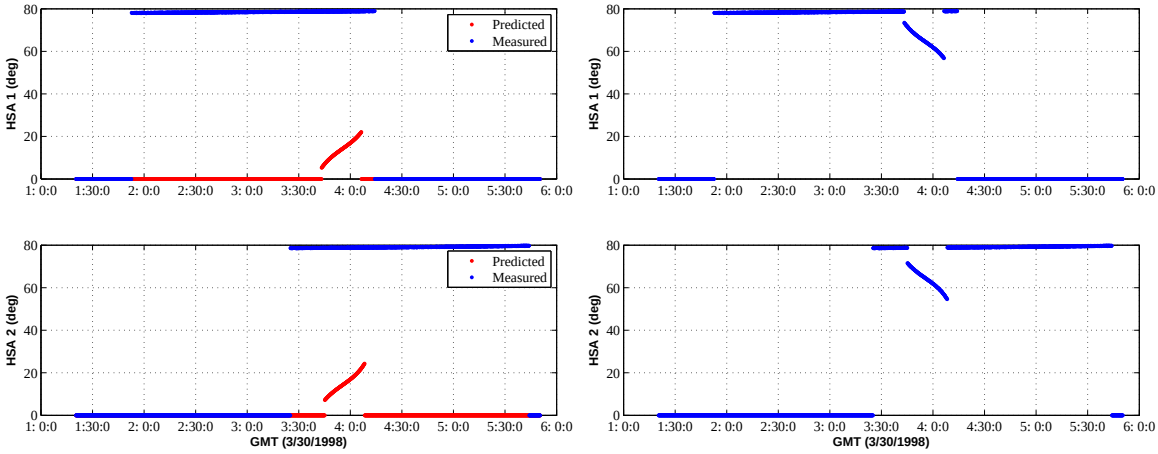
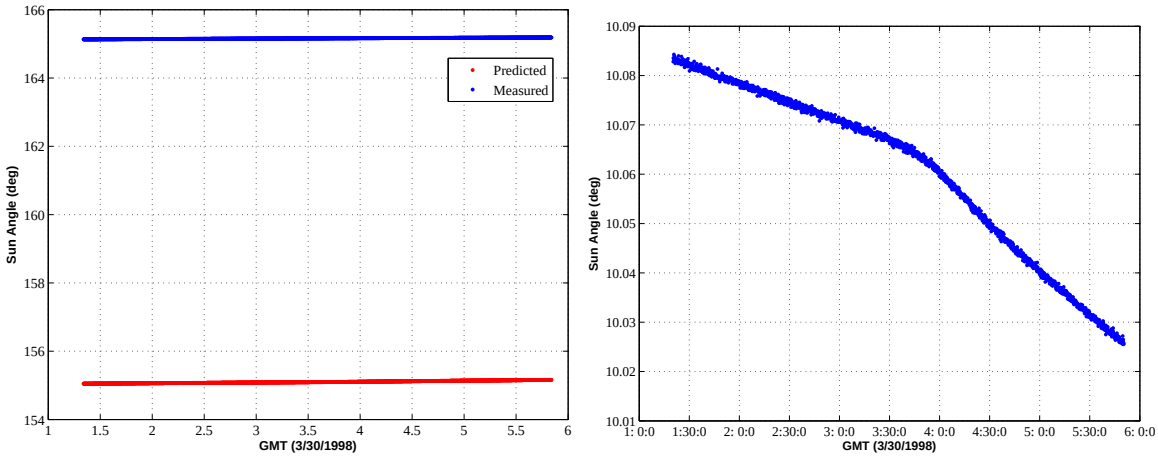


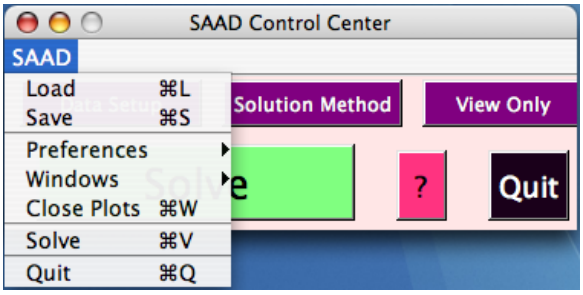
Figure 3-6. Sample sun angles and residuals



3.2 Menu

You can save and reload case data after filling in this GUI using the menu. For an example, see `DemoSAADCase.mat` in SAADData, mentioned in the previous section. You can use this example data for learning to use this GUI. There are several preferences you may set, including having a warning to save your data before performing solve actions. The menu also contains navigation between the ISAAD windows and an action to close the plots generated.

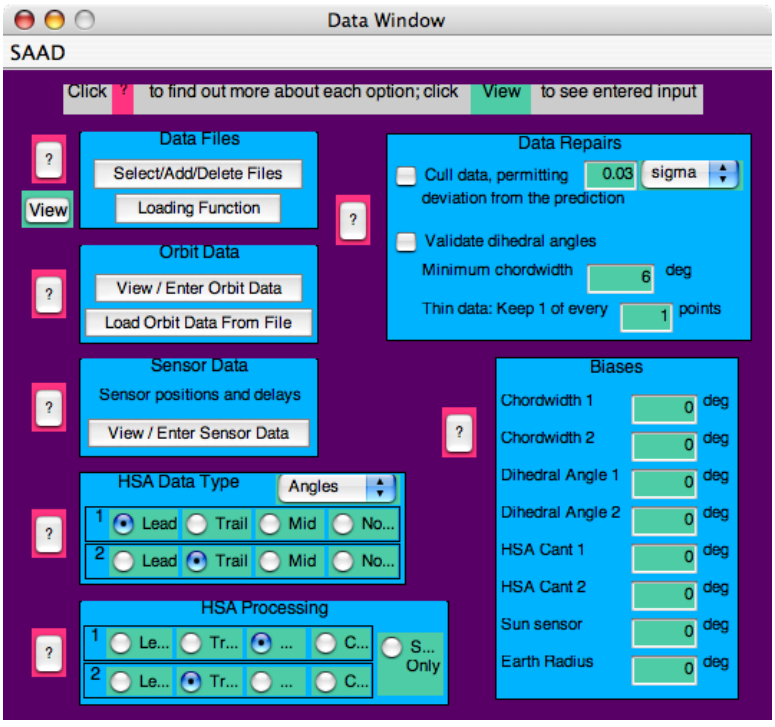
Figure 3-7. SAAD Menu



3.3 Data Setup

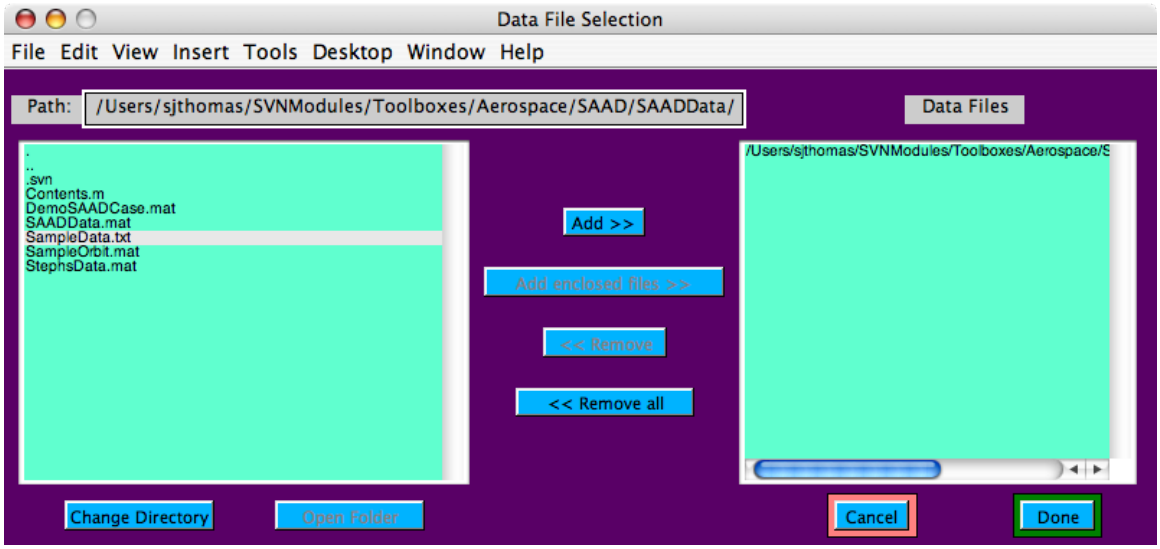
When you click Data Setup from the main window you will see the Data Window.

Figure 3-8. Data Window



The Select/Add/Delete File button allows you to add and/or delete files from a list of currently selected files. You negotiate to the files you want to add by typing in a path, or by using the pop-up dialog in Figure 3-9 on the next page.

Figure 3-9. Data File Selection



The Loading Function button brings up a standard dialog for you to choose a data loading function. The purpose of this function is to read the data files produced by your ground system, and translate them into the form used by the software. This function must have the following form:

```
measuredData = LoadFunctionName(files)
   ------
   Inputs
   ------
   files             (:)     Struct array of file names - Fields are
       .pathName    string
       .fileName    string

       -------
       Outputs
       -------
       measuredData    (1,1)    Struct array of measured data. Fields are:
           .jD           (:)     Time tags
           .cW1          (:)     HSA 1 Chordwidths
           .cW2          (:)     HSA 2 Chordwidths
           .dA1          (:)     HSA 1 Dihedral Angles
           .dA2          (:)     HSA 2 Dihedral Angles
           .sA           (:)     Sun angles'
```

Ideally, the Matlab working directory will be unchanged after calls to this function. A sample loading function is included with the Toolbox, called `LoadSAADData.m`. This function assumes that the data is stored in space or tab delimited ascii format, and that each row of data is in the following order:

```
    y m d h m s sA cW1 dA1 cW2 dA2'
```

The sun angles, chordwidths, and dihedral angles are assumed to be in degrees. If your ground system generates the data in a different format the sample loading function can be modified, or a custom function with the given input-output specification can be written.

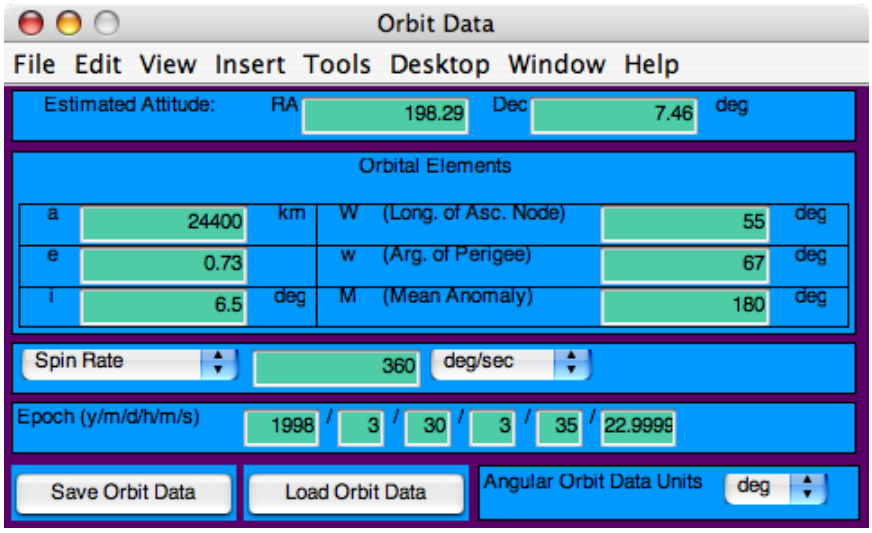
Orbit data can either be entered directly from the screen, or loaded from a file. The required entries are:

- Right Ascension (RA) and Declination (Dec)

- Orbital Elements: Semi-major axis (a), Eccentricity (e), Inclination (i), Longitude of the ascending node (W), Argument of perigee (w), and Mean anomaly (M)
- SpinRate
- Epoch

Figure 3-10 shows the orbit data window with the sample case data loaded.

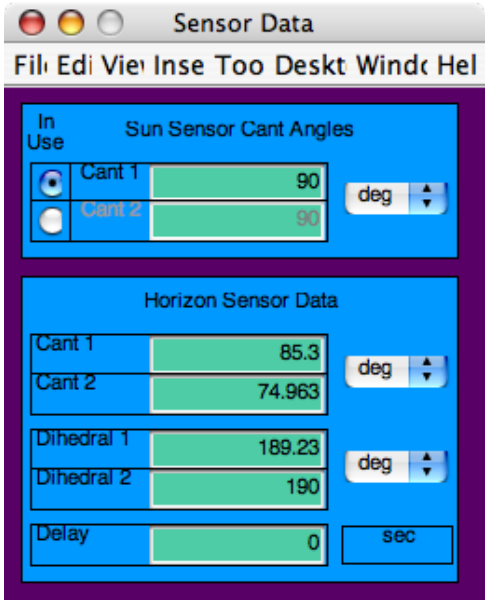
Figure 3-10. Orbit Data window



The spin rate can also be entered as a period. The units menu in the lower right hand corner of the orbit data display window allows the user to choose either degrees or radians for display and entry of all the angular data.

The sensor data consists of angles which define the positions of the sun and horizon sensors, and the time delay in the horizon sensor data.

Figure 3-11. Sensor Data window



Cant angles for both types of sensors are defined as the angles from the spin axis to the sensor boresights. The horizon sensor (HSA) dihedral angles are the angles from the sun sensor to the HSAs in the plane perpendicular to the spin axis. The HSA delay is the time delay in the system between when the horizon is actually detected, and the edge times sent to the processor. This delay results in a bias in the dihedral angle measurements that is dependent on the spin rate.

The HSA data may be in the form of chordwidth and dihedral angles, or leading and trailing edge times. Use the menu in the upper right-hand corner to select angles or times. If the data is angles, the dihedral angles may be leading, trailing, or midscan. If the data is times, choose midscan unless only 1 time (leading edge or trailing edge) is valid. In that case, your selection should indicate which is valid. If there is no data available from an HSA (for example, if it was off the earth during an entire data pass), then choose None for that HSA. This will prevent solution methods which require HSA data.

The data from each HSA may be processed as a leading edge, trailing edge, or midscan angle. In general, midscan processing should be used because this method makes use of all available HSA data. Leading and trailing processing methods assume that no chordwidths are available. These may be useful if some malfunction causes only leading or trailing angles (or edge times) to be available. CW is chordwidth only processing; dihedral angles are ignored if provided. SSA only processing uses the sun angle only, no HSA data. The accuracy of this method may be limited, and the data must cover a long enough time span for both RA and Dec to be observable. The first 4 options will be diabled if no HSA data is available (i.e. None is selected in the HSA Data Type block).

Culling eliminates data which differs from the prediction by more than the given percentage. Culling will increase the accuracy of your solution by eliminating bad data. Validating the dihedral angles ensures that all times with zero chordwidths also have zero dihedral angles. Minimum chordwidth: Data points with chordwidths below this value will be discarded. This feature is useful if your HSAs provide inaccurate chordwidths when the earth is small. Data thinning: If you run out of memory during solutions, or the processing time is very long, you can "thin" the data by keeping only 1 of every n points.

All known biases in the data should be entered; except for HSA time delays. Because the angular value of the delay will be dependent on the spin rate, the delay amount should be entered with the sensor data, and the correct angular value will be automatically calculated.

3.4 Solutions

The solution window appears when you click `Solution Method` from the main GUI (Figure 3-12 on the next page). Each method has a subwindow for specifying options which appears when you click the buttons on the right.

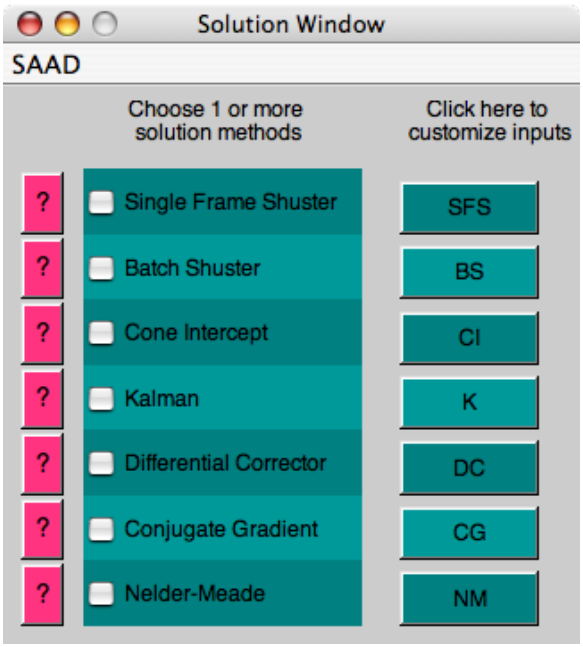
Single Frame Shuster is a closed form solution method which uses a vector formulation. It takes into account the chordwidths and dihedral angles. A separate solution is calculated for each HSA. Use it to get an initial attitude estimate or to check your final attitude and bias solutions.

Batch Shuster is a closed form solution method which uses a vector formulation. It takes into account the chordwidths and dihedral angles. A separate solution is calculated for each HSA. Use it to get an initial attitude estimate or to check your final attitude and bias solutions.

Cone Intercept is a closed form solution method. It takes into account only the chordwidths, an therefore finds two possible solutions. A bucket method is used to pick the correct solution. Use Cone Intercept to get an initial attitude estimate. A separate solution is calculated for each HSA.

Kalman implements an extended, iterated Kalman filter. Examining the attitude and covariance trends can locate bad data. The solution of this method should be viewed with caution if the covariances increase during the solution, if the residual statistics are not better for the solution than for the original data, or if the final plots of the measured and predicted data do not match up.

Figure 3-12. Solution window



These are advanced options that can be used to fine-tune the Kalman solution method. The forgetting factor can be used to prevent the covariance from becoming artificially low by "forgetting" some of the information gained from measurements. It should be a number between 0 and 1, where 1 means the filter doesn't forget at all, 0 means that the filter forgets each measurement right away (preventing the covariance from changing at all). The number of iterations determines how many iterations are performed at each measurement. The attitude reference is used to create an error plot, if it is entered.

Differential Corrector is a batch least squares estimator algorithm. It iterates over the set of measurements until the differential correction between estimates becomes very small, or the loss function no longer decreases. (Ref: Wertz, ed. (1976), "Spacecraft Attitude Detrmination and Control," Kluwer.) You can solve for some or all of the states listed below. The states not solved for will be assumed to be equal to the biases entered in the Data Window, except for the right ascension and declination, which are assumed equal to the estimated attitude entered with the orbit data.

- Right Ascension
- Declination
- HSA Cant biases - errors in the modeled cant angles
- HSA Dihedral biases - errors in the modeled sensor dihedral angles or biases in the measured dihedral angles. The latter could be caused by a time delay.
- Sun sensor cant bias
- Earth radius bias

There are advanced options that can be used to fine-tune the Differential Corrector solution method, as shown in Figure 3-14 on the facing page. The error tolerance is used to determine when to stop the iteration. The SVD (singular value decomposition) tolerance is used to determine the rank of the matrix for the least squares solution. The SVD tolerance is the lowest singular value permitted. Showing the initial value of HWH can help determine if you are attempting to solve for too many states. (Low HWH indicates a well-determined problem, higher HWH may mean you should solve for fewer states. However, it may be misleading if the first measurement does not contain data from both HSAs.

Figure 3-13. Differential Corrector states selection window

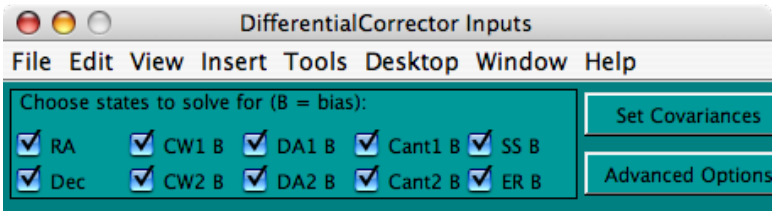
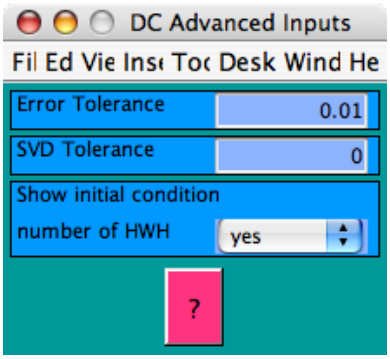


Figure 3-14. Differential Corrector advanced options



The Conjugate Gradient method is a batch least-squares estimator algorithm. It operates by improving the estimate at each step by moving in a direction that is not only "conjugate" or perpendicular, to the current gradient, but to all previous directions as well. You can solve for some or all of the states listed above under Differential Corrector.

Nelder-Meade finds the solution which has the least "cost" as measured by the deviations of the measurements from the solution, weighed by the uncertainty in the measurements. It converges more slowly than the other numerical methods, but often provides better results. You can solve for some or all of the states listed with Differential Corrector.

There are advanced options that can be used to fine-tune the Nelder-Meade solution method. The tolerances for x and cost are both used to determine when to stop the iteration. The max number of steps limits the iterations permitted. Showing the intermediate steps causes the state and cost, and number of iterations to be printed at each step.

When you are finished configuring your solutions, you click the Solve button. In the demo case, the following output is printed to the screen.

```
>> ISAAD

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Solving, please wait...
HSA1 Chordwidth Mean     =     -4.19 deg
HSA1 Chordwidth Std      =     46.33 deg
HSA2 Chordwidth Mean     =     -3.24 deg
HSA2 Chordwidth Std      =     45.95 deg
HSA1 Dihedral Angle Mean =     76.21 deg
HSA1 Dihedral Angle Std  =      5.29 deg
HSA2 Dihedral Angle Mean =     76.57 deg
HSA2 Dihedral Angle Std  =      6.08 deg
Sun Sensor Mean          =     10.06 deg
Sun Sensor Std           =      0.02 deg

--------------------------------------------------------------------------------

Nelder-Meade
```

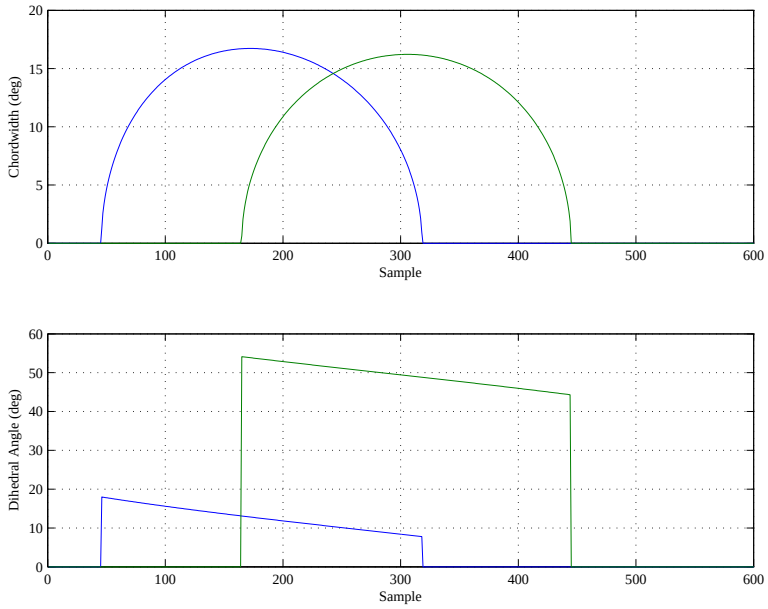
```
--------------------------------------------------------------------------------


x0 =

    3.6652
    0.1571
         0
         0
         0
         0
         0
         0
         0
         0

Right Ascension          =    225.12 deg
Declination              =      1.12 deg
HSA1 Chordwidth Bias     =     -7.15 deg
HSA2 Chordwidth Bias     =    -64.40 deg
HSA1 Dihedral Angle Bias =     -5.33 deg
HSA2 Dihedral Angle Bias =     13.76 deg
HSA1 Cant Angle Bias     =     14.73 deg
HSA2 Cant Angle Bias     =     43.10 deg
Sun Sensor Bias          =      6.04 deg
Earth Radius Bias        =    -13.49 deg

Right Ascension Error =        15.12 deg
Declination Error     =        -7.88 deg

-----Residual Statistics-------
HSA1 Chordwidth Mean     =      9.67 deg
HSA1 Chordwidth Std      =     34.82 deg
HSA2 Chordwidth Mean     =     65.26 deg
HSA2 Chordwidth Std      =     35.35 deg
HSA1 Dihedral Angle Mean =     78.26 deg
HSA1 Dihedral Angle Std  =     14.12 deg
HSA2 Dihedral Angle Mean =     58.81 deg
HSA2 Dihedral Angle Std  =     14.61 deg
Sun Sensor Mean          =     16.04 deg
Sun Sensor Std           =      0.03 deg
```

CHAPTER 4

SAAD DEMOS

There are three script demos included in the Module, `TBAD.m`, `TCF.m`, and `TKalAD.m`. They are listed in entirety in this chapter. All of the demos use `ADGen` to generate example attitude determination data. The HSA data is shown in the plot.

Figure 4-1. HSA Data



4.1 Numerical Methods with `TBAD`

Listing 4.1. Script to demonstrate numerical methods *TBAD.m*

```
%--------------------------------------------------------------------------
%   Test attitude determination using the batch methods
%   See also ConjGrad, DiffCorr
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%   Copyright 1994-2001 Princeton Satellite Systems, Inc. All rights reserved.
%--------------------------------------------------------------------------
```

```
9
10 % Define global variables
11 %-----------------------
12 global uss_ us1_ us2_ ue1_ ue2_ er1_ er2_ zs_ zh1_ zh2_ zd1_ zd2_;
13 global Ws_ Wh1_ Wh2_ Wd1_ Wd2_ cant1 cant2;
14 global x0_ d_ S0_ xA_;
15
16 nsamples  = 400;
17
18 degToRad  = pi/180;
19 rPMToRPS  = 2*pi/60;
20
21 cant1     = 94*degToRad;
22 cant2     = 86*degToRad;
23 dihedral1 =  0*degToRad;
24 dihedral2 = 45*degToRad;
25
26 sigcw     = 1;
27 sigda     = 1;
28 sigss     = 1;
29
30 ssbias    = 0.01;
31 ssnoise   = 0.0*degToRad;
32 ssquant   = 0.00000000001*degToRad;
33
34 erbias    =  0.0;
35 cw1bias   =  0.0;
36 cw2bias   =  0.0;
37 da1bias   =  0.0;
38 da2bias   =  0.0;
39 cant1bias =  0.0;
40 cant2bias =  0.0;
41
42 c10      = cos(10*degToRad);
43 s10      = sin(10*degToRad);
44
45 sun      = [c10;0;s10];
46
47 usun     = [ c10*ones(1,nsamples);zeros(1,nsamples);s10*ones(1,nsamples)];
48
49 qtype    = 'round';
50 onesigma = 0.0;
51 spinrate = 10*rPMToRPS;
52 quant    = 1.e-12;
53 delay    = 0;
54
55 % The orbital elements
56 %--------------------
57 el(1)        = (42167 + 6800)/2;
58 el(2)        = 7 * pi/180;
59 el(3)        = 0;
60 el(4)        = 0;
61 el(5)        = 0.7;
62
63 ra           =  1.5;
64 dec          = -0.4;
65 uspin        = RaDec2U(ra,dec);
66
67 % Physical spacecraft data
68 %-------------------------
69 cantAngle    = [cant1;cant2];
70 cantBias     = [cant1bias;cant2bias];
71 dihedralBias = [dihedral1;dihedral2];
72 mA           = linspace(pi/2,3*pi/2,nsamples);
73
74 % Generate attitude data
75 %----------------------
76 [tLE,tTE,sa,unadir,eradius] = ADGen( el, usun, uspin, spinrate, mA, [delay;delay], ...
```

```
                                    [quant;quant], [qtype;qtype], [onesigma;onesigma], ...
                                    cantAngle, cantBias, dihedralBias, ssquant, ...
                                    'round', ssnoise, ssbias, erbias );

% Just a change of variable for convenience
%-------------------------------------------
tte1 = tTE(1,:);
tte2 = tTE(2,:);
tle1 = tLE(1,:);
tle2 = tLE(2,:);

% Extract the data with chordwidths above a threshold
%----------------------------------------------------
chordwidth1 = spinrate*(tte1-tle1);
chordwidth2 = spinrate*(tte2-tle2);

% Short chords have lots of information but tend to be unreliable because the model
% is inaccurate when chords are small
%-------------------------------------------------------------------------------------
kr  = find(chordwidth1 > 6*degToRad | chordwidth2 > 6*degToRad);

% This is the data used by the estimators
% The first three are ephemeris data the next two are chordwidths, the following
% two are dihedral angles and the last is the sun angle
%-------------------------------------------------------------------------------
us  =   usun(:,kr);
ue  = unadir(:,kr);
er  = eradius(kr);
cw1 = chordwidth1(kr) + cw1bias;
cw2 = chordwidth2(kr) + cw2bias;
da1 = 0.5*spinrate*(tte1(kr)+tle1(kr))-dihedral1 + da1bias;
da2 = 0.5*spinrate*(tte2(kr)+tle2(kr))-dihedral2 + da2bias;
sb  = sa(kr);

% The state vector:
%          1          2        3        4        5        6
% [       ra,       dec, cwbias1, cwbias2, dabias1, dabias2,
%          7          8        9      10
%  cantbias1, cantbias2, sunbias, erbias]

lkr = length(kr);

% Create the measurement vector. Must conform to definitions
% in hname and Hname

k1 = find(cw1 > 0);
k2 = find(cw2 > 0);

uss_ = us;
us1_ = us(:,k1);
us2_ = us(:,k2);
ue1_ = ue(:,k1);
ue2_ = ue(:,k2);
er1_ = er(k1);
er2_ = er(k2);
zs_  = sb';
zh1_ = cw1(k1)';
zh2_ = cw2(k2)';
zd1_ = da1(k1)';
zd2_ = da2(k2)';
Ws_  = ones(lkr,1)       /sigss^2;
Wh1_ = ones(length(k1),1)/sigcw^2;
Wh2_ = ones(length(k2),1)/sigcw^2;
Wd1_ = ones(length(k1),1)/sigda^2;
Wd2_ = ones(length(k2),1)/sigda^2;

length([zs_;zh1_;zh2_;zd1_;zd2_]);

```

```
x0      = [1.6 -0.3  0.01 0 0  0 0 0  0 0]';

P  = diag([100,100,1,1,1,1,1,1,1,1]);
S0 = inv(P);
%S0 = zeros(10,10);

% States to be solved for
%------------------------
kX = 1:2;

disp('------------------------------------------------------------------------')
disp('Nelder-Meade')
disp('------------------------------------------------------------------------')

% options(1)  is nonzero, intermediate steps in the solution are displayed;
% options(2)  is the termination tolerance for x; the default is 1.e-4.
% options(3)  is the termination tolerance for F(x); the default is 1.e-4.
% options(14) is the maximum number of steps; the default is 500.

options           = zeros(1,14);
options([2 3 14]) = [1.e-4 1.e-4 500];

if( IsVersionAfter( 5.3 ) )
  TolX    = options(2); TolFun = options(3); MaxFunEvals = options(14);
  Options = optimset('TolX',TolX,'TolFun',TolFun,'MaxFunEvals',MaxFunEvals);
  x       = fminsearch( 'CostNM',x0(kX), Options,   kX,S0,x0,10);
else
  x       = fmins     ( 'CostNM',x0(kX), options,[],kX,S0,x0,10);
end

x = rem(x,2*pi);
uSpinNM = RaDec2U( x(1), x(2) )

disp('------------------------------------------------------------------------')
disp('Conjugate Gradient')
disp('------------------------------------------------------------------------')

[x,k,P,wmr,sr,J,sig,nz] = ConjGrad( 'FX', 'CostF', S0, x0, kX, 0.00001, 1, 0 );

x = rem(x(:,end),2*pi);
uSpinCG = RaDec2U( x(1,end), x(2,end) );

disp('------------------------------------------------------------------------')
disp('Differential Corrector')
disp('------------------------------------------------------------------------')

[x,k,rsvd,cHWH,rank,P,wmr,sr,J,sig,nz] = DiffCorr( 'FX', S0, x0, kX, .001, 0 );

x = rem(x(:,end),2*pi);
uSpinDC = RaDec2U( x(1,end), x(2,end) );

disp(sprintf('\nSpin Vectors\n'))
disp(sprintf('%24s %24s %24s\n','Nelder Meade','Conjugate Gradient',...
             'Differential Corrector'))
ax = ['x' 'y' 'z'];
for k = 1:3
  disp(sprintf('%s %24.8f %24.8f %24.8f\n',ax(k),uSpinNM(k),uSpinCG(k),uSpinDC(k)));
end

%--------------------------------------
% PSS internal file version information
%--------------------------------------
% $Date: 2007-06-22 14:05:33 -0400 (Fri, 22 Jun 2007) $
% $Revision: 10037 $
```

TBAD.m

The output of the demo is shown below.

```
>> TBAD
------------------------------------------------------------------------
Nelder-Meade
------------------------------------------------------------------------

uSpinNM =

    0.0637
    0.9119
   -0.4054

------------------------------------------------------------------------
Conjugate Gradient
------------------------------------------------------------------------
Using conjugate gradients
Initial cost   4.9760e+00

Cost and number of measurements used   6.3084e-01 863

Cost and number of measurements used   2.6663e-01 958

Cost and number of measurements used   2.1131e-01 971

Cost and number of measurements used   2.1082e-01 975

Cost and number of measurements used   2.1081e-01 975

Cost and number of measurements used   2.1081e-01 975

------------------------------------------------------------------------
Differential Corrector
------------------------------------------------------------------------

Spin Vectors

            Nelder Meade       Conjugate Gradient   Differential Corrector

x               0.06367329              0.06406641               0.06400678

y               0.91190515              0.91482650               0.91484554

z              -0.40543151             -0.39873296              -0.39869884
```

4.2 Closed Form Methods with TCF

Listing 4.2. Script to demonstrate closed form methods *TCF.m*

```
%--------------------------------------------------------------------------
%   Test attitude determination using closed form methods
%   See also SFShustr, BShuster, ConeInt
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%   Copyright 1994-1995 Princeton Satellite Systems, Inc. All rights reserved.
%--------------------------------------------------------------------------

SetFont('Times',9)

```

```
nsamples  = 600;

degToRad  = pi/180;
radToDeg  = 180/pi;
rPMToRPS  = 2*pi/60;

cant1     = 94*degToRad;
cant2     = 86*degToRad;
dihedral1 =  0*degToRad;
dihedral2 = 45*degToRad;

ssbias    = 0.0;
ssnoise   = 0.0*degToRad;
ssquant   = 0.000001*degToRad;

erbias    =  0.0;
cw1bias   =  0.01;
cw2bias   =  0.0;
da1bias   =  0.0;
da2bias   =  0.0;
cant1bias =  0.0;
cant2bias =  0.0;

c10       = cos(10*degToRad);
s10       = sin(10*degToRad);

sun       = [c10;0;s10];

usun      = [ c10*ones(1,nsamples);zeros(1,nsamples);s10*ones(1,nsamples)];

qtype     = 'round';
onesigma  = 0.0;
sunbias   = 0.0;
spinrate  = 10*rPMToRPS;
quant     = 1.e-6;
delay     = 0;

% The orbital elements
%---------------------
el(1)         = (42167 + 6800)/2;
el(2)         = 7 * pi/180;
el(3)         = 0;
el(4)         = 0;
el(5)         = 0.7;

ra            =  1.5;
dec           = -0.4;
uspin         = RaDec2U(ra,dec);

cantAngle     = [cant1;cant2];
cantBias      = [cant1bias;cant2bias];
dihedralBias  = [dihedral1;dihedral2];
mA            = linspace(pi/2,3*pi/2,nsamples);

[tLE,tTE,sa,unadir,eradius] = ADGen( el, usun, uspin, spinrate, mA, [delay;delay],...
                                     [quant;quant], [qtype;qtype], [onesigma;onesigma],...
                                     cantAngle, cantBias, dihedralBias, ssquant,...
                                     'round', ssnoise, ssbias, erbias );

tte1 = tTE(1,:);
tte2 = tTE(2,:);
tle1 = tLE(1,:);
tle2 = tLE(2,:);

% Single frame Shuster method
%----------------------------
[rAE,decE,rd] = SFShustr( usun, unadir, sa, tle1, tte1, cant1, eradius, spinrate, 6*degToRad,
     dihedral1 );
```

```
79
80 disp(sprintf('Single␣Frame␣Schuster:␣Right␣ascension␣error␣=␣%12.4f␣deg',(ra - rAE)*radToDeg)
       )
81 disp(sprintf('␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Declination␣error␣␣␣␣=␣%12.4f␣deg',(dec-decE)*radToDeg)
       )
82
83 % Single frame Shuster method
84 %---------------------------
85 sigmaS = 0.0001*degToRad;
86 sigmaE = 0.0001*degToRad;
87 sigmaN = 0.0001*degToRad;
88 [rAE,decE,rd] = BShuster( usun, unadir, sa, tle1, tte1, cant1, eradius, spinrate,...
89                           6*degToRad, dihedral1, sigmaS, sigmaE, sigmaN );
90
91 disp(sprintf('Batch␣Schuster:␣Right␣ascension␣error␣=␣%12.4f␣deg',(ra  -  rAE)*radToDeg))
92 disp(sprintf('␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Declination␣error␣␣␣␣=␣%12.4f␣deg',(dec - decE)*radToDeg))
93
94 % Cone Intercept Method
95 %---------------------
96 [ra,dec,rd] = ConeInt( usun, unadir, sa, tle1, tte1, cant1, eradius, spinrate, 6*degToRad );
97
98 disp(sprintf('Cone␣Intercept:␣Right␣ascension␣error␣=␣%12.4f␣deg',(ra  -  rAE)*radToDeg))
99 disp(sprintf('␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Declination␣error␣␣␣␣=␣%12.4f␣deg',(dec - decE)*radToDeg))
100
101 %--------------------------------------
102 % PSS internal file version information
103 %--------------------------------------
104 % $Date: 2007-06-22 14:05:33 -0400 (Fri, 22 Jun 2007) $
105 % $Revision: 10037 $
```

TCF.m

The demo output is below.

```
>> TCF
Single Frame Schuster: Right ascension error =      -0.0001 deg
                       Declination error     =      -0.0004 deg
Batch Schuster: Right ascension error =      -0.0001 deg
                Declination error     =      -0.0004 deg
Cone Intercept: Right ascension error =      -0.0005 deg
                Declination error     =      -0.0024 deg
```

The demo also produces the plots in Figure 4-2.

Figure 4-2. Single Frame Schuster

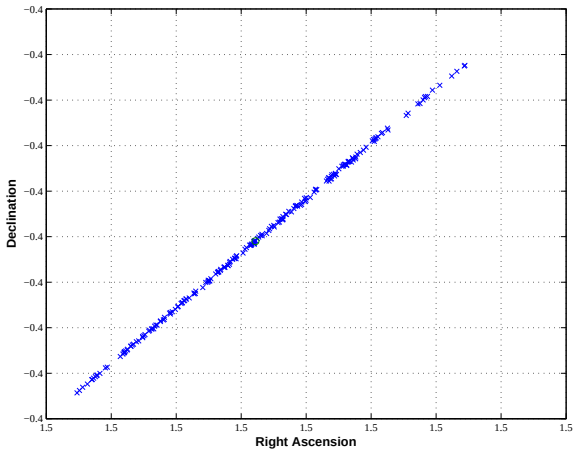


Figure 4-3. Cone Intercept right ascension and declination

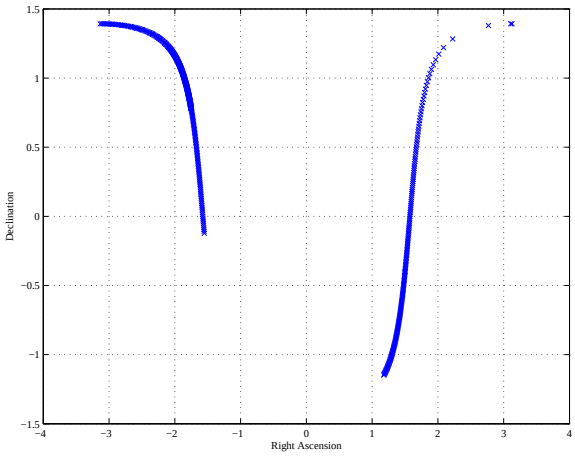
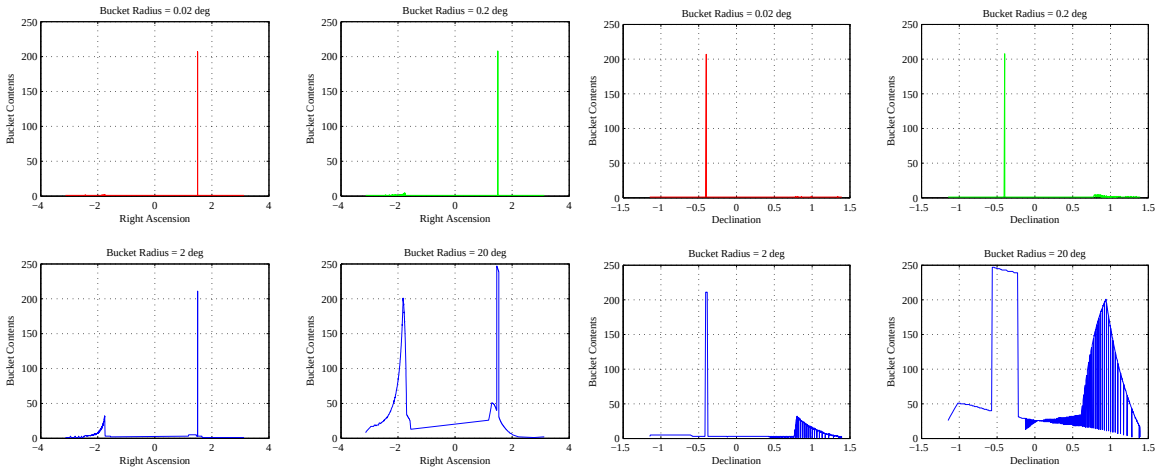


Figure 4-4. Cone Intercept Buckets and Sorted Buckets



4.3 Attitude Determination with TKalAD

Listing 4.3. Script to demonstrate a Kalman Filter for attitude determination *TKalAD.m*

```
%--------------------------------------------------------------------------------
%   Attitude determination using a Kalman Filter
%   See also KFSAAD
%--------------------------------------------------------------------------------

%--------------------------------------------------------------------------------
%   Copyright (c) 1994-1995 Princeton Satellite Systems, Inc.
%   All rights reserved.
%--------------------------------------------------------------------------------

nsamples  = 600;

degToRad  = pi/180;
rPMToRPS  = 2*pi/60;

cant1     = 94*degToRad;
cant2     = 86*degToRad;
dihedral1 =  0*degToRad;
dihedral2 = 45*degToRad;

ssbias    = 0.0;
ssnoise   = 0.0*degToRad;
ssquant   = 0.000001*degToRad;

erbias    =  0.0;
cw1bias   =  0.0;
cw2bias   =  0.0;
da1bias   =  0.0;
da2bias   =  0.0;
cant1bias =  0.0;
cant2bias =  0.0;

% Sun vector is constant in this demo
%------------------------------------
c10      = cos(10*degToRad);
s10      = sin(10*degToRad);
usun     = DupVect([c10;0;s10],nsamples);

qtype    = 'round';
onesigma = 0.0;
sunbias  = 0.0;
spinrate = 10*rPMToRPS;
quant    = 1.e-6;
delay    = 0;

% The orbital elements
%---------------------
el(1)        = (42167 + 6800)/2;
el(2)        = 7 * pi/180;
el(3)        = 0;
el(4)        = 0;
el(5)        = 0.7;

% The true attitude
%------------------
ra           =  1.5;
dec          = -0.4;
uspin        = RaDec2U(ra,dec);

% Get the horizon sensor data and sun angles
%-------------------------------------------
cantAngle    = [cant1;cant2];
cantBias     = [cant1bias;cant2bias];
```

```
dihedralBias = [dihedral1;dihedral2];
mA           = linspace(pi/2,3*pi/2,nsamples);

[tLE,tTE,sa,unadir,eradius] = ADGen( el, usun, uspin, spinrate, mA, [delay;delay],...
                                     [quant;quant], [qtype;qtype], [onesigma;onesigma],...
                                     cantAngle, cantBias, dihedralBias, ssquant,...
                                     'round', ssnoise, ssbias, erbias );

% Reformat the data
%------------------
tte1 = tTE(1,:);
tte2 = tTE(2,:);
tle1 = tLE(1,:);
tle2 = tLE(2,:);

% Extract the data with chordwidths above a threshold
%----------------------------------------------------
chordwidth1 = spinrate*(tte1-tle1);
chordwidth2 = spinrate*(tte2-tle2);

kr          = find(chordwidth1 > 6*degToRad | chordwidth2 > 6*degToRad);

us          =   usun(:,kr);
ue          = unadir(:,kr);
er          = eradius(kr);
cw1         = chordwidth1(kr) + cw1bias;
cw2         = chordwidth2(kr) + cw2bias;
da1         = 0.5*spinrate*(tte1(kr)+tle1(kr))-dihedral1 + da1bias;
da2         = 0.5*spinrate*(tte2(kr)+tle2(kr))-dihedral2 + da2bias;
sb          = sa(kr);

% The state vector
%-----------------
%          1          2        3        4        5        6
% [       ra,       dec, cwbias1, cwbias2, dabias1, dabias2,
%           7          8        9       10
%  cantbias1, cantbias2, sunbias, erbias]

% Measurement functions
% Sun:            harc, dhdxarc
% Chordwidth:     hcw,  dhdxcw
% Dihedral angle: hda,  dhdxda

H    = zeros(1,10);

% Initial guess
%--------------
x    = [1.5 -0.3  0.01 0 0  0 0 0  0 0]';
ff   = 0.1;
nits = 1;
Rsb  = 0.01;
Rcw  = 0.1;
Rda  = 0.1;
P    = diag([100,100,100,1,1,1,1,1,1,1]);

% Call the Kalman filter. It will process all measurements and plot the results
%------------------------------------------------------------------------------
KFSAAD(x,P,Rcw,Rda,Rsb,cw1,cw2,da1,da2,sb,ue,us,er,cant1,cant2,ff,[],[ra,dec])

%-------------------------------------
% PSS internal file version information
%-------------------------------------
% $Date: 2007-06-22 14:05:33 -0400 (Fri, 22 Jun 2007) $
% $Revision: 10037 $
```

TKalAD.m

This demo creates the following plots.

Figure 4-5. Spin Axis

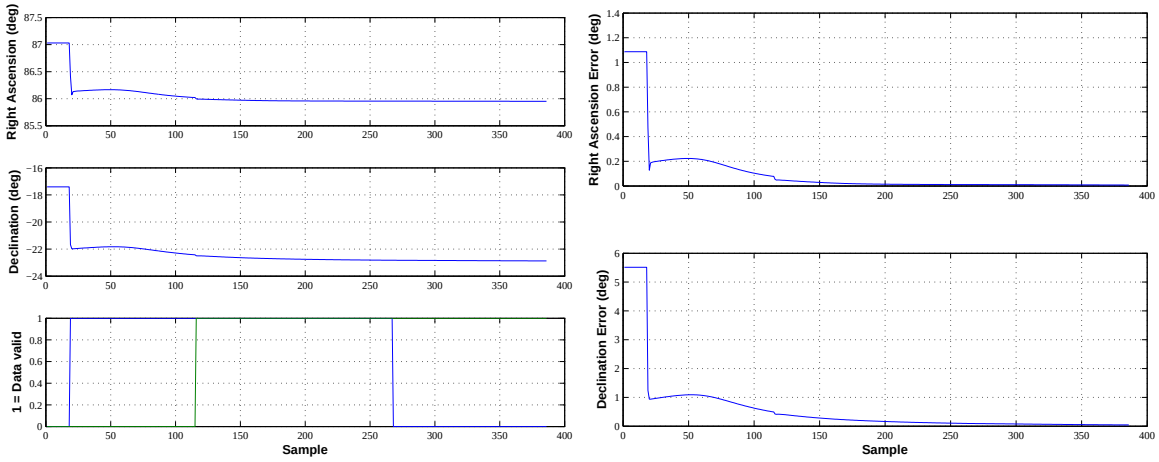


Figure 4-6. Biases

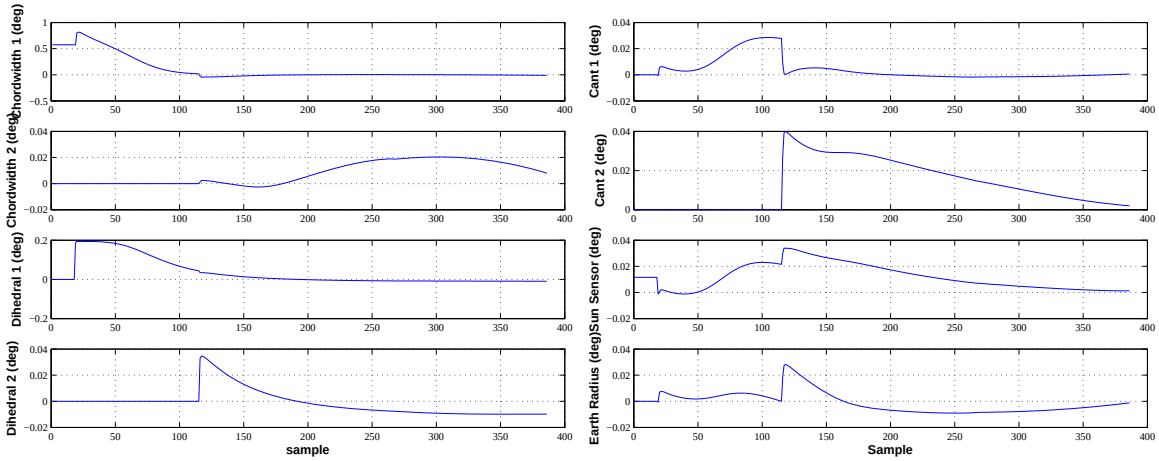


Figure 4-7. Covariances