

# Verification of Pointing Constraints for the Dawn Spacecraft

C. Anthony Vanelli\*, Edward Swenka† and Brett Smith‡

*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109*

NASA’s Dawn spacecraft, an ion-thrust science mission to Vesta and Ceres, has numerous pointing constraints critical for safe operation. Onboard software automatically chooses target attitudes but enforces only a simplified constraint set at slew endpoints. Onboard fault-protection also uses simplified constraints, and violations can result in safing events that dramatically consume mission margins for missed thrust. Lastly, for funding reasons the operations team is lean, forcing the development of month-long command sequences. These factors place a premium on reliable maneuver design, prediction, and verification against pointing constraints. This paper presents Sleuth, a ground tool built to address these concerns.

## Nomenclature

$I_{sp}$	specific impulse
$\Delta v$	delta velocity
$r$	current spacecraft heliocentric range
$q$	current spacecraft attitude with respect to J2000
$q_{goal}$	intended goal attitude of spacecraft
$q_{cmd}$	instantaneous spacecraft attitude command (sent to attitude controller)
$v_a$	spacecraft aiming vector, in spacecraft frame
$v_g$	inertial target (goal) vector, in J2000 frame
$v_s$	constraint model sensitivity vector, in spacecraft frame
$v_t$	constraint model threat vector, in J2000 frame
$\alpha$	constraint model cone half-angle
$r_{min}$	minimum heliocentric range at which a constraint is active
$r_{max}$	maximum heliocentric range at which a constraint is active
$t_g$	time spent in geometric incursion of a constraint
$t_c$	time spent in outside (clear) of a geometric constraint
$t_{charge}$	allowable constraint exposure time until violation (“charge-up time”)
$t_{discharge}$	time required for a constraint violation to reset (“discharge time”)

## I. Introduction

NASA’s Dawn spacecraft, launched in September 2007, is a deep space mapping mission en route to asteroids Vesta and Ceres, using an enhanced version of the low-thrust ion propulsion system flown by Deep Space 1 (DS1). Built by Orbital Sciences Corporation and the Jet Propulsion Laboratory (JPL), Dawn will conduct science operations at Vesta from August 2001 through May 2012, then depart for Ceres, arriving in February 2015. If all goes well, Dawn will become the first interplanetary spacecraft to orbit two planetary bodies, a feat made possible by the capability of its ion-propulsion system (IPS).

---

\*Senior Engineer, Guidance and Control System Engineering, 4800 Oak Grove Drive, Mail Stop 264-828.

†Manager, Guidance and Control System Engineering, 4800 Oak Grove Drive, Mail Stop 264-828.

‡Staff Engineer, Guidance and Control System Engineering, 4800 Oak Grove Drive, Mail Stop 230-104.

Low thrust propulsion systems provide extraordinary flexibility in mission design. Compared to chemical-based systems with specific impulse ( $I_{sp}$ ) of 200–400 seconds, the Dawn engines can provide  $I_{sp}$  from 1900–3200 seconds depending on available solar power. This high efficiency reduces propellant mass while enabling extraordinarily large total deliverable  $\Delta v$ . In turn, large  $\Delta v$  grants flexibility in choosing mission objectives, plus a wide selection of launch dates and trajectories. Compared to the previous record of 4.3 km/s established by DS1, the Dawn IPS will deliver about 11 km/s total  $\Delta v$  over the life of its eight-year mission.<sup>1,2</sup>

However, this capability comes at a price. Compared to conventional “ballistic coast” interplanetary missions that use chemical propulsion to deliver  $\Delta v$  nearly impulsively, the Dawn spacecraft must maintain thrust nearly continuously for long periods to build up sufficient  $\Delta v$  to reach its objectives (see Figure 1). As Rayman et al relate, during certain parts of the mission the thrust duty cycle can approach 95%, while attempting to preserve a total “missed thrust budget” of approximately one month.<sup>1</sup> These facts result in a kind of “thrust schedule pressure”: an impetus to maintain thrusting as much as possible, particularly during the early phase of the mission when solar power is readily available, in order to “bank” as much thrust as possible against those days when thrust will be lost due to anomalies and other mishaps.

Continuous thrusting also means continuous operations costs. The mission requires the constant attention of a fully-staffed navigation team, since continuous thrusting means the spacecraft trajectory is constantly being re-shaped. For funding reasons the operations team is generally very lean, and this fact has forced the adoption of month-long activity sequences. Planning such long sequences is a recurring challenge.

As with any spacecraft, there are numerous pointing constraints that must be honored for celestial navigation, power and thermal management, safe IPS operation, and the safety of science instrumentation. While onboard attitude steering software uses a simplified heuristic for choosing target attitudes during maneuvers, this algorithm does not fully capture all the pointing constraints and can occasionally result in solutions that are not always desirable from an operations perspective. Additionally the onboard fault-protection constantly monitors a subset of constraints with the potential response of halting thrust, resulting in lost thrust margin.

The continuously changing nature of the trajectory, the length of the planning cycles, the abundance of spacecraft pointing constraints, and nuances in the software behaviors place a high value on having a maneuver design and prediction tool that can predict spacecraft maneuvers, verify them against pointing constraints, and provide assurances that a chosen month-long activity sequence will meet operations needs while satisfying all pointing constraints. Iterations are typically necessary, but lean staffing makes time precious, so short turnaround time is essential to success.

In April 2007, development of the Sleuth ground tool began in earnest to address these needs. Sleuth (whimsically called Sleutooth) runs a model of the flight software pointing logic, feeding in the spacecraft trajectory, initial conditions, and a time-indexed sequence of ground commands, and checks the resultant spacecraft behavior against a programmable list of constraints. Sleuth utilizes the JPL-standard navigation toolkit to provide knowledge of celestial bodies such as the Earth, sun, other planets, and asteroids; the locations of these bodies are used both to provide target pointing references to the flight software pointing model as well as the constraint checking logic. A record of constraint violations (and clearances) is produced for the sequence under test.

## II. Spacecraft Configuration and Pointing Constraints

The Dawn spacecraft is a hybrid design, drawing upon heritage from Orbital’s STAR-2 and LeoSTAR-2 designs and upon JPL’s experience from Deep Space 1, the first interplanetary low-thrust mission. The flight system is shown in Figure 2. We now briefly mention those elements that provide well-quantified pointing constraints.

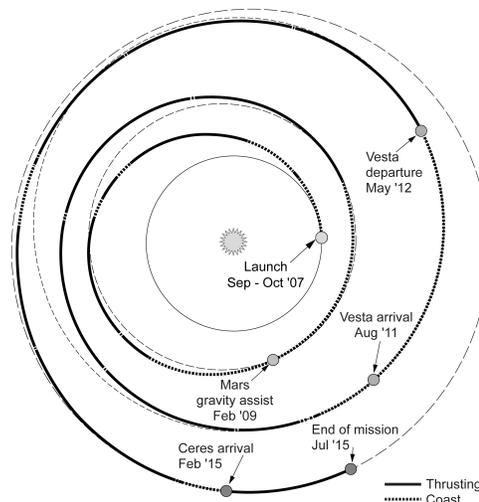
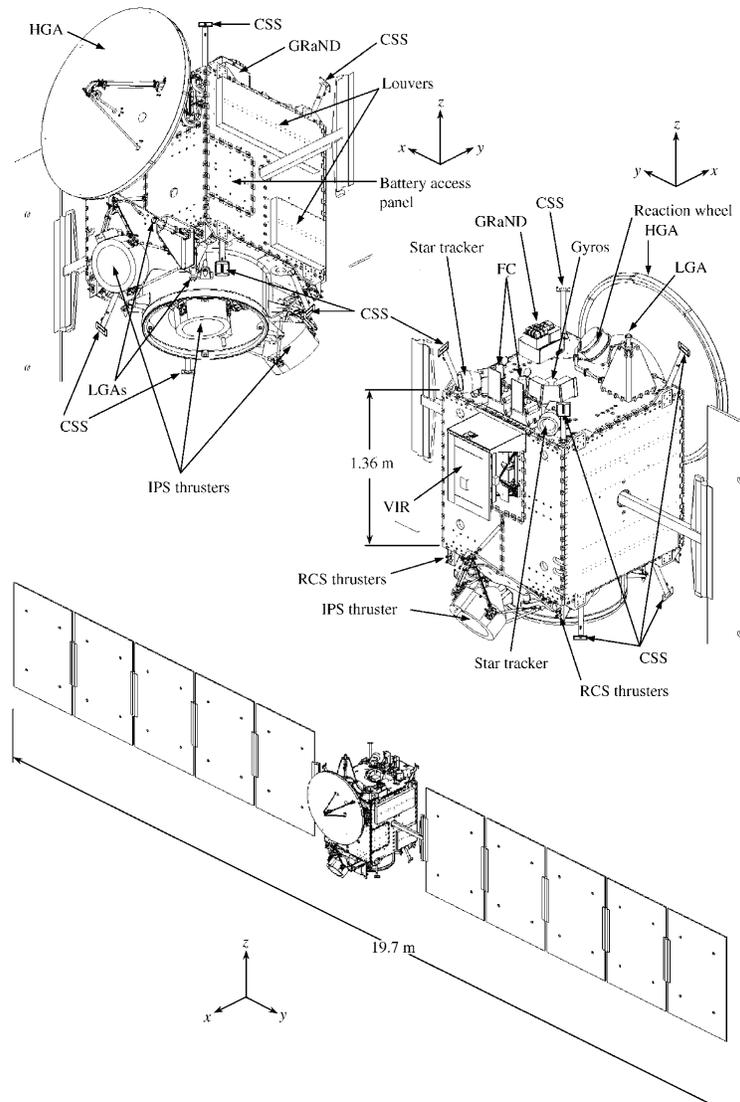


Figure 1. Dawn interplanetary trajectory as of September 2007. The solid portions indicate IPS thrusting periods; the dotted portions indicate coasting or science on-orbit operations.



**Figure 2. Dawn flight system.**

The flight system features three xenon ion-propulsion engines, compared to DS1's single engine. The engines are operated one-at-a-time. One engine thrusts along the spacecraft centerline, while the other two are canted at nearly  $50^\circ$  from the spacecraft  $+z$ -axis. All three engines are gimballed in two axes to allow the attitude control system (ACS) to control the spacecraft attitude using thrust vector control. (Reaction wheels or hydrazine thrusters provide the third axis.) The placement of the engines, as well as the internal xenon tank, imposes thermal restrictions on solar input to the lower portion of the vehicle. These restrictions vary according to whether the central IPS engine is operating or not.

The spacecraft draws power from two large ( $18 \text{ m}^2$  each) solar array panels, mounted along a motorized shaft providing  $360^\circ$  range of motion about the spacecraft  $y$  axis. These large arrays are required to provide sufficient power to operate an IPS engine even at 3 AU. Optimal power generation requires that the sun be kept within the spacecraft  $xz$  plane.

The science instrumentation is mounted atop the  $+z$  deck of the spacecraft and consists of two Framing Cameras (FC), a Gamma Ray and Neutron Detector (GRaND), and the Visible and Infrared spectrometer (VIR). These instruments each impose specific restrictions on  $+z$  deck exposure to the sun.

From a thermal standpoint, the spacecraft is generally flown with the  $+x$  face toward the sun and the  $-x$  face kept in shadow. The VIR instrument is mounted on the  $-x$  face, imposing additional restrictions

in that direction. Thermal louvers and the GRaND instrument also impose constraints on the  $+y$  face.

A fixed high gain antenna (HGA) mounted on the front side of the spacecraft, and three low gain antennae (LGA) mounted facing the  $+z$ ,  $+x$ , and  $-z$  faces of the spacecraft, provide communications. Only one LGA is active at any one time.

The ACS system uses two star trackers mounted at the rear of the  $+z$  deck, each aimed  $30^\circ$  to either side of the spacecraft  $xz$  plane. These trackers are sensitive to bright bodies such as the sun (and Vesta and Ceres, when in orbit there). Additionally, the onboard flight software maintains the ephemerides of up to 3 other celestial bodies (for science and HGA targeting). If any of these bodies should stray within a pre-set angle of a star tracker's boresight, the flight software flags that tracker suspect and will ignore the output from that star tracker. Unfortunately, this occurs regardless of range to the body or whether the star trackers maintain lock, and the ground must intervene to reset the software flag.

Table 1 provides a sampling of the constraints imposed on the Dawn spacecraft. The restrictions vary in size, direction, allowable exposure time, and also vary according to heliocentric range. Some constraints are imposed for flight system safety, while others are imposed to maintain a high standard of performance.

**Table 1. Some examples of Dawn pointing constraints**

System	Sensitive Axis	Threat	Exclusion Angle	Applicable Range	Allowable Exposure Time	Reason
VIR	$-x$	Sun	$90^\circ$ $42^\circ$	$\leq 2.1$ AU $2.1+$ AU	none none	performance
VIR	$-x$	Sun	$69^\circ$	$\leq 2.1$ AU	8 hr	safety
VIR	$+z$	Sun	$10^\circ$	any	none	safety
VIR	$+z$	Sun	$60^\circ$	any	none	performance
FC	$-x$	Sun	$90^\circ$ $40^\circ$	$\leq 2.1$ AU $2.6+$ AU	none none	performance
FC	$+z$	Sun	$45^\circ$	any	none	performance
FC	$+z$	Sun	$7.8^\circ$ $7.8^\circ$	$\leq 2.1$ AU $2.1+$ AU	10 s 240 s	performance
FC	$+z$	Sun	$45^\circ$	$\leq 2.0$ AU	none	safety
GRaND	$+y$	Sun	$35^\circ$	$\leq 1.2$ AU	none	safety
GRaND	$-x$	Sun	$90^\circ$ $60^\circ$	$\leq 1.4$ AU $\leq 1.8$ AU	none none	safety
GRaND	$+z$	Sun	$45^\circ$	$\leq 1.2$ AU	none	safety
GRaND	$+z$	Sun	$20^\circ$	$\leq 1.3$ AU	none	safety
IPS *(when on)	$-z$	Sun	$70^\circ$ $45^\circ$	$\leq 1.4$ AU $\leq 2.0$ AU	none none	safety
IPS *(when off)	$-z$	Sun	$45^\circ$	$\leq 1.5$ AU	none	safety
Power	$-y$ $+y$	Sun Sun	$45^\circ$ $45^\circ$	any any	none none	maintain power
Star Trackers	boresights boresights	Sun <i>body n</i>	$33^\circ$ <i>varies</i>	any any	none none	performance

\* center IPS engine only.

### III. Relevant Flight Software Behaviors

The majority of the flight software, including the pointing logic to be discussed herein, is heritage from Orbital's LEO missions and has been adapted for deep space use. Portions of the flight software relating to onboard ephemeris knowledge are heritage from JPL. The onboard Fault Protection (FP) software represents

a combination of Orbital and JPL experience. The FP software enforces a reduced set of pointing constraints and reacts to violations. The ACS pointing software has attitude steering logic (called “power steering”) that attempts to maintain pointing within general safe bounds, but is unaware of specific constraint rules (and of FP). Each of these software elements are described in turn.

### III.A. Fault Protection Constraint Checking

Fault Protection watches over a small subset of constraints, listed in Table 2. Each of these constraints can be enabled or disabled within the FP software.

Table 2. Some examples of Dawn Fault Protection constraints.

System Protected	Sensitive Axis	Threat	Exclusion Angle	Tolerated Exposure	When Applicable
IPS	$-z$	Sun	$70^\circ$	60 s	applies if center IPS on
IPS	$-z$	Sun	$45^\circ$	300 s	applies if center IPS off
GRaND	$+y$	Sun	$35^\circ$	60 s	applies if GRaND on
GRaND	$-x$	Sun	$90^\circ$	60 s	applies if GRaND on
GRaND & FC	$+z$	Sun	$45^\circ$	25 s	applies if GRaND or FC on
VIR & FC	$+z$	Sun	$25^\circ$	10 s	applies if VIR or FC on
VIR	$-x$	Sun	$69^\circ$	8 hr	always applicable

Fault Protection has only one effective reaction to constraint violation: it commands an “Earthpoint” response wherein it first cancels all ongoing activities, including shutting down IPS thrust, next commands ACS to point the high gain antenna at Earth, and then awaits instructions from the ground. In certain dire situations (usually indicating a loss of attitude reference or control failure), FP may command a “safe mode” response (a.k.a. “Sunpoint”), wherein it cancels all ongoing activities, and orders ACS to give up 3-axis attitude knowledge and hold the  $+x$  axis on the sun using the sun sensors and gyros, while awaiting ground instruction through the LGA antenna. The  $\sim 10$ bps data rates available through the LGA make diagnosis and recovery arduous, resulting in lost thrust margin. Significantly, this most dire response can be triggered when the ACS software rejects a pointing command, as discussed below.

### III.B. ACS Attitude Commanding

Under nominal operations, the ACS flight software offers five flavors of three-axis control modes, shown in Figure 3. The first of these, known as TargetInertial mode, is a generic 3-axis pointing mode. The pointing command system accepts an aiming vector in spacecraft body coordinates and a corresponding inertial target vector. The target vector may be inertially fixed, but most often is retrieved from a list of known targets propagated on board by the flight software.

Of the other four modes, two are used for thrusting and two are used for science activities. With regard to pointing commands, these modes can be considered special cases of the TargetInertial mode: the aiming vector is pre-defined by the modes, and the target vector is evaluated from pre-selected onboard ephemeris. This capability allows the thrust vector or the science target vector to vary with time.

The pointing system is depicted in Figure 4. Observe that the pointing system does not specify a full 3-axis attitude. It accepts only a single aiming vector  $v_a$  and a corresponding inertial target (or goal) vector  $v_g$ , which describe a *family* of candidate attitudes with a single degree of freedom. An onboard algorithm called “Power Steering” augments this one-axis attitude in order to devise the full 3-axis attitude.

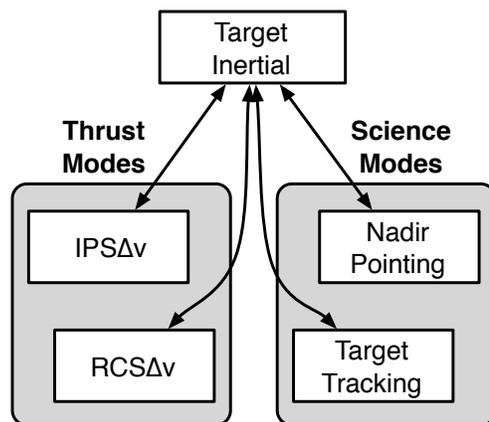


Figure 3. Dawn ACS modes.

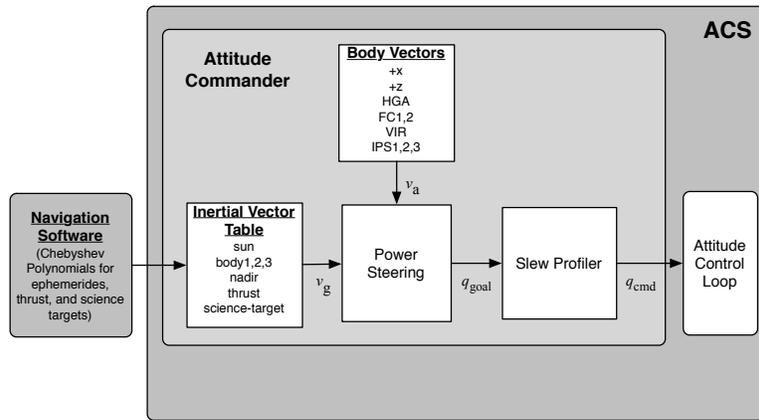


Figure 4. Dawn attitude commanding.

### III.B.1. Power Steering

Given the desired one-axis pointing command, the flight software’s Power Steering logic attempts to construct a full 3-axis attitude solution  $q_{\text{goal}}$  by finding an attitude that keeps the array yokes orthogonal to the sun vector (or as close to orthogonal as possible), while also satisfying the primary pointing requirement. Although it is not actually computed in this manner, one may visualize the Power Steering solution per the following recipe:

1. Rotate the spacecraft to align the spacecraft aiming vector  $v_a$  with the inertial target vector  $v_g$ .
2. Spin the spacecraft about the newly-aligned vectors so as to make the  $y$  axis maximally orthogonal to the sun vector. Note that there may be multiple candidate solutions, but in general there are two.
3. Resolve this ambiguity by choosing for  $q_{\text{goal}}$  whichever candidate solution minimizes sun exposure on the  $-x$  face of the spacecraft.
4. In the event that all candidates provide the same  $-x$  sun exposure, set  $q_{\text{goal}}$  to whichever solution minimizes sun exposure on the  $+z$  face.
5. In the event that the preceding step still does not determine a unique solution, set  $q_{\text{goal}}$  to the candidate solution requiring the least Euler-angle displacement from the *current* attitude.

Note that the pointing command may specify a mathematically degenerate case, such as “point  $v_a$  at the sun” (i.e.,  $v_g$  is the sun). In this case the primary pointing requirement and the augmented array yoke pointing requirement are not independent: both reference the sun. Observe that no matter how the spacecraft rotates about the aiming vector, the sun remains fixed in spacecraft coordinates. One cannot improve the array pointing nor the sun exposure on various parts of the spacecraft. In this case, the power steering algorithm simply selects the least-displacement slew from the current attitude.

### III.B.2. Onboard Vector Tables

To compute the final attitude, the Power Steering algorithm uses two short “vector tables” maintained aboard the spacecraft (see Figure 4). The Body Vector Table is a collection of static aiming vectors in spacecraft body coordinates, such as the  $+x$  axis or the VIR boresight. A special command option allows the ground operator manually specify any chosen direction vector in the spacecraft frame.

The Inertial Vector Table carries up to seven time-varying inertial target vectors (in J2000 coordinates). The first five are derived from ephemeris data: these are the vectors from the spacecraft to the Sun, to each of up to three programmable targets of interest (such as Earth, Mars, and Vesta), and to the central body of the spacecraft’s orbit (currently also the Sun). The sixth and seventh vectors are the thrust direction vector and the science target vector, used in thrust and science modes.

The Navigation flight software computes these vectors using onboard ephemerides (represented as Chebyshev polynomials) of the spacecraft orbit about its central body and the solar orbits of the three planetary targets. Chebyshev polynomials also provide the two vectors used for thrust pointing and science pointing. The Navigation software evaluates these polynomials and publishes the resultant state vectors to the ACS software. Though these state vector updates are quite frequent, the ACS software interpolates the states between updates to assure smoothness.

In TargetInertial mode only, a special pointing option allows the operator to specify a static inertial vector by providing Right Ascension and Declination angles in J2000 coordinates. This option is not available while in the thrusting or science modes.

### III.B.3. Tracking and Slewing

Under nominal operation the power steering logic always executes, continuously recomputing the target attitude  $q_{\text{goal}}$  from the most recently received pointing command, the latest information from the inertial vector tables, and the current attitude. An exception to this rule occurs when a new pointing command is received. Pointing commands can arrive in several ways: explicitly as sequenced commands to change the current one-axis pointing specification, or implicitly as a change in ACS's operating mode (i.e., a shift from TargetInertial mode to a thrust pointing mode or a science pointing mode, or vice versa).

When a new pointing command is received, the power steering logic computes the new goal attitude, which can be a long ways off—and hence can produce a large discontinuous jump in commands sent to the attitude controller. To handle this scenario, an intermediate “slew profiler” function steps in at the moment a new pointing instruction is received. This function intercepts the output from the power steering algorithm and smoothly slews the spacecraft from the *a priori* attitude to the new target attitude. The slew profile consists of a time series of attitude and rate commands that take the spacecraft along an Euler-axis slew trajectory using either a triangular or trapezoidal rate profile (ramp-ramp or ramp-coast-ramp). The slew profiler does not consider array yoke orthogonality nor sun exposure on the  $-x$  and  $+z$  faces during the slew; these considerations are only in effect at slew endpoints (when the slew profiler is inactive).

### III.B.4. Caveats

Although the ACS software produces a power-optimal attitude at all times (except while in mid-slew) and attempts to minimize sun exposure of the  $-x$  and  $+z$  sides of the spacecraft, there are some caveats.

With no commandable option to specify a full 3-axis attitude, operators have no way to lock the spacecraft inertially: the ACS is free to rotate the spacecraft about the aiming vector to optimize array pointing. Moreover, if the geometry changes over time so that the “alternate” solution is now superior, the spacecraft may “flip”—either fast or slow, depending on how fast geometry changes. Under most thrusting and cruise activities, these effects are both tolerable and beneficial. But they can lead to difficulty for science and optical navigation activities: one does not usually desire a camera or other instrument to also rotate about its own boresight when aiming it at a target, arranging mosaics, and so forth. This is particularly important for VIR, whose field-of-view is a slit.

The ACS software also rejects pointing commands that place the sun too close to either the  $+y$  or  $-y$  axes. (The software originally rejected commands that place the sun too close to the  $+z$  axis as well, but this behavior has been disabled.) Critically, ACS rejects any new pointing command that is received while a slew is in progress. *These pointing command rejections are a serious matter*: when the system rejects a pointing command, it triggers a Fault Protection response that shuts off IPS thrusting and commands the spacecraft to safe mode. Together these behaviors mean that command sequences must ensure valid pointing targets, and must carefully leave adequate time for a slew to complete before commanding another one.

Power Steering's “fallback” behavior that resorts to the least-displacement slew from the current attitude can also cause difficulty: it encodes a dependency on attitude history into activity sequences. This dependency can render a sequence ineffective unless care is taken to make it robust, or to assert a specific initial attitude at the start (by deliberately choosing a starting attitude for which Power Steering is known to prefer a particular solution).

Lastly, because FP is watching even during slews, it is possible that a slew which “nicks” an FP constraint risks an FP response even if the goal attitude is ultimately safe.

## IV. Slewth

Given the number and complexity of the constraints, the behavioral features of the attitude commanding software, and the small number of flight team members, a tool was needed that could reliably predict both spacecraft attitude behavior as well as warn of constraint violations. Development began about six months before launch, presenting the virtual certainty that Slewth’s development would proceed through launch and beyond. The need for at least partial functionality soonest, plus the shortness of development time dictated an architectural and design philosophy that encouraged rapid prototyping and testing to retire risk as early as feasible, yet could gracefully accomodate expansion without becoming cumbersome.

### IV.A. Design Approach

The fundamental premise behind the Slewth software is to model only the pointing command and their kinematics, not to simulate the entirety of the spacecraft dynamical environment (control torques, environmental disturbance, and the like). Without denying the usefulness of full dynamic simulation, we argue that for our purposes this premise is valid: the properly designed attitude controller will maintain the attitude within a control tolerance of the attitude command, whereas the properly designed attitude commander delivers commands that the controller can follow.

This premise allows Slewth to do away with modeling environmental disturbances, spacecraft dynamics, actuators, sensors, and indeed much of the flight software behavior. It dramatically focuses the development effort onto just the core pieces of the problem: what the ground commands the spacecraft to do, how the spacecraft interprets those commands, and whether the resultant behavior violates any pointing constraints.

#### IV.A.1. Architecture

Figure 5 presents the heart of the Slewth architecture. It consists of three functional groupings, together with an over-arching command interface (not shown). The middle group is the ACS Pointing Model, which models how the ACS flight software generates the attitude commands. The Constraint Checker follows this group, evaluating the commanded attitude against the programmed constraints. At the left, the Inertial Vector Manager provides both the ACS Pointing Model and the Constraint Checker with the correct inertial vectors. Slewth runs these functional groups completely at each simulated time step (200 ms).

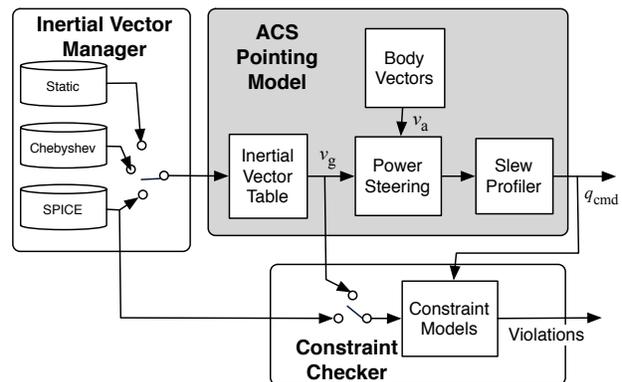


Figure 5. Slewth core architecture.

#### IV.A.2. Inputs

The Slewth command layer interfaces with all the functional groups, and controls execution. The command layer is a TCL-driven command-line interface, through which the operator selects the constraints to be checked, loads the activity’s ephemeris into the Inertial Vector Manager, sends commands to the ACS pointing model, and instructs Slewth to execute over a range of time. This interface was chosen because it is interactive, scriptable, and programmable. Interactive behavior confers the flexibility needed when the ground team seeks to answer the “what if?” questions that typically occur during sequence design; input scripts provide the convenience, recordkeeping, and repeatability needed when the team wishes to run (and re-run) a pre-designed sequence of commands through Slewth and check the results. The programmable features of the TCL interface have yet to be employed in operations, but are available if needed.

#### IV.A.3. Outputs

Data is collected and recorded from all three functional groups on each time step. From this data, Slewth produces constraint violation logs, time histories of spacecraft attitudes and rates, summaries of all slews

executed and how long they took, and time histories of all the vectors of interest (sun, Earth, and any other vectors loaded into the Inertial Vector Manager). The data format produced by Slewth is compatible with the data format used by JPL’s standard multi-mission queries tools, for compatibility with pre-existing software for generating plots and rendering visualizations.

#### **IV.B. Inertial Vector Manager**

The Inertial Vector Manager (IVM) provides generalized inertial vector services to both the ACS Pointing Model and the Constraint Checker. Recall that the Navigation flight software uses Chebyshev polynomials to represent up to seven inertial targets: five ephemeris targets and two directional targets. These polynomials are generated from SPICE<sup>a</sup> files, a standard format in use at JPL to represent ephemeris and vector data. The SPICE files represent the best available “truth” knowledge of the spacecraft’s current trajectory.

The IVM emulates the behavior of the Navigation flight software, providing the ACS Pointing Model with the necessary vector updates. It has three specialized modes: Static Mode, SPICE Mode, and Chebyshev Mode.

In Static Mode, the user may assert constant values for each of the five ephemeris vectors (spacecraft to sun, to the three planets, to the central body), as well as the thrust direction and science target direction vectors. These vectors remain fixed until the user next commands them. Historically, Static Mode was the first capability provided in Slewth, and remains useful in testing “special case” scenarios.

In SPICE Mode, Slewth utilizes user-provided SPICE files for the ephemerides. From this data it automatically computes the five ephemeris state vectors. The thrust and science target vectors are still taken as constants provided by the operator, just as they are in Static Mode. Thus computed, all seven inertial target vectors are fed into the ACS Pointing Model at a programmable update interval, chosen to match the current flight update rate on the spacecraft. Note that in this mode, Slewth is effectively feeding the “truth” trajectory into the ACS Pointing Model.

Chebyshev Mode is the most flight-like, using the same Chebyshev files uplinked to the flight system itself. The IVM computes all seven inertial target vectors from the provided Chebyshev files, and provides state updates to the ACS Pointing Model at the proper update interval. Chebyshev Mode thus becomes a mechanism by which Slewth can validate not only the pointing sequence, but the Chebyshev uplink products as well.

#### **IV.C. ACS Pointing Model**

The ACS Pointing Model is functionally equivalent to how the ACS pointing flight software itself works, as detailed earlier. Slewth’s ACS Pointing Model accepts all the same commands that drive the ACS flight software and influence pointing behavior. These commands include the pointing commands, mode changes (which carry implicit pointing commands), parameter updates that change entries in the body vector table, and slew profiler tuning commands.

Commands which are irrelevant to Slewth (such as sun sensor threshold settings, for example) are benignly ignored. This feature simplifies the process of testing flight sequences with Slewth.

#### **IV.D. Constraint Checker**

The Slewth Constraint Checker checks all pointing constraints for violations, and emits warnings for each violation encountered.

Constraints are not hard-coded but are defined at run-time, affording both versatility and resilience to changing flight experience. This flexibility is especially important in light of the difficulty in establishing well-characterized thermal models of the spacecraft and its hardware. As better understanding of spacecraft behavior is gained, it becomes a simple matter to update the constraint table and re-test existing sequences in light of the new data.

Slewth builds upon the constraint model used in Refs. 3 and 4 to define constraints that consist of a combination of shape specifications, range specifications, and exposure time specifications. These specifications may be combined to create constraints that are large or small, active only at certain ranges or at any range, trigger immediately or slowly over time, or any combination. In the sequel, we break these specifications

---

<sup>a</sup>Spacecraft Planet Instrument C-Matrix Event

into *geometric* and *temporal* components. Note that the use of the term *geometric* connotes both shape *and* range.

#### IV.D.1. Geometric Constraint Component

Constraint geometry consists of both *shape* and *range*. For simplicity, we define the shape of a constraint as a simple cone of half-angle  $\alpha$  centered on a spacecraft sensitive vector  $v_s$ , and keyed against a specified inertial threat or “danger” vector  $v_t$ .

Constraints are also defined to have ranges  $[r_{\min}, r_{\max}]$  over which they are active. If the spacecraft is outside the this range, the constraint is inactive. For simplicity all ranges are heliocentric.

For a chosen constraint, if the spacecraft attitude  $q$  and heliocentric range  $r$  satisfy *both*

$$r_{\min} \leq r \leq r_{\max}, \quad (1)$$

$$v_s \cdot v_t - \cos \alpha \geq 0, \quad (2)$$

(where  $v_t$  has been transformed into the spacecraft frame using  $q$ ) then we say that the chosen constraint is under a *geometric incursion*. If either of these conditions are not met, then we say that the constraint is in geometrically *clear*.

Slewth provides each constraint model with the threat vector identified with that constraint. Typically the threat vector data is provided by SPICE files loaded into Slewth. It is occasionally useful to check the constraints against the vectors used by the ACS Pointing Model.

#### IV.D.2. Temporal Constraint Component

Some spacecraft constraints allow for hardware to tolerate exposure to a threat for a finite amount of time, whereas others trigger instantly. After having been exposed, such hardware may have a recovery period that must be satisfied before the effects of such exposure have diminished. We parameterize this behavior with two simple values: how much time  $t_{\text{charge}}$  must elapse before the constraint will trigger a violation, and how much time  $t_{\text{discharge}}$  must elapse before the constraint will be fully “reset.”

The time  $t_g$  spent under geometric incursion is tallied; if  $t_g \geq t_{\text{charge}}$  then the constraint triggers a violation warning. Setting  $t_{\text{charge}}$  to zero triggers warnings immediately upon geometric incursion.

If spacecraft later clears the geometric constraint, then the time  $t_c$  spent outside the constraint shape is tallied; when  $t_c \geq t_{\text{discharge}}$  the constraint violation warning is cancelled and the constraint is reset. Setting  $t_{\text{discharge}}$  for a constraint immediately resets the constraint whenever the geometry clears.

#### IV.D.3. Constraint States

From the above description, we can derive four basic states for a constraint. Figure 6 depicts these states and their transitions. We define these states as follows:

- **COMPLIANCE.** The spacecraft attitude and range lie outside the constraint geometry, and the constraint has not been violated for at least  $t_{\text{discharge}}$  time.
- **CHARGING.** The spacecraft attitude and range intrude upon the constraint geometry, but the total time spent within the constraint bounds has not yet exceeded  $t_{\text{charge}}$ . (The constraint is “warming up.”)
- **VIOLATION.** The spacecraft attitude and range have intruded upon the constraint geometry for at least  $t_{\text{charge}}$  time. The constraint is violated.
- **DISCHARGING.** The spacecraft attitude and range no longer intrude upon the constraint geometry, but the time spent outside the constraint bounds has not yet exceeded  $t_{\text{discharge}}$ . (The constraint is “still warm.”)

Slewth emits advisories with every transition. These advisories are collected in a violation report for the activity being tested. The operations team uses these violation reports to assess whether or not the activity is safe to execute on board the spacecraft.

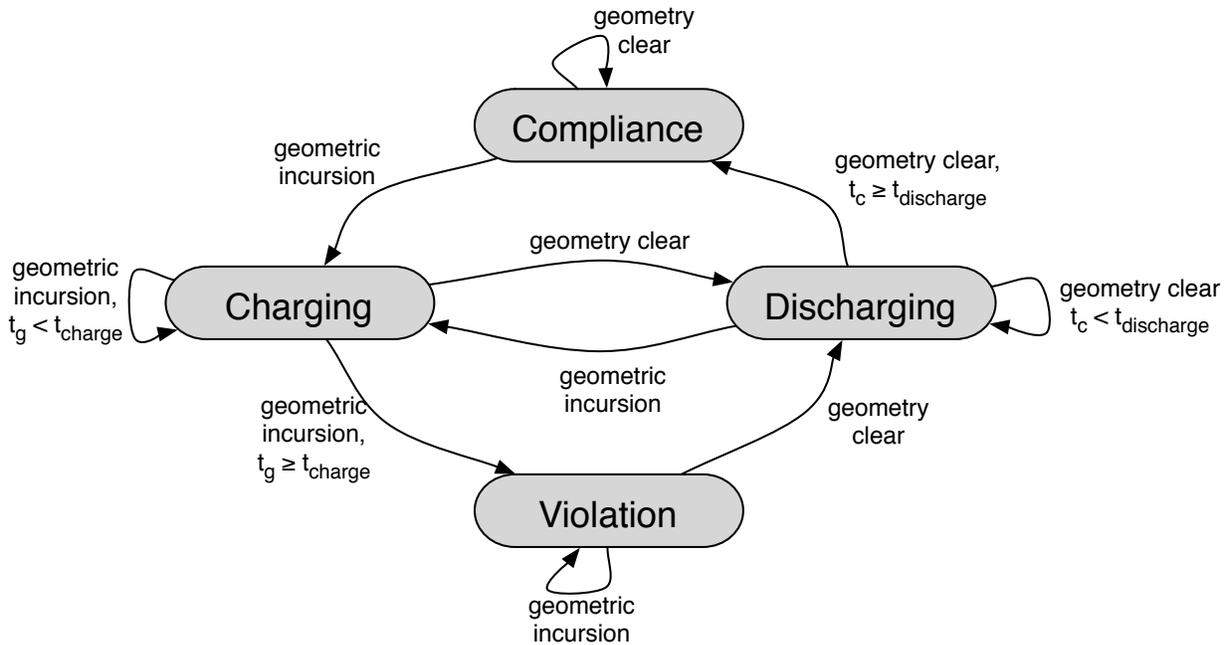


Figure 6. Slewth constraint states transitions

#### IV.E. Performance

Slewth successfully completes simulation runs spanning 100 days of mission activities in about 2 hours’ execution time on a Sparc Ultra 45 workstation. The original requirement on its performance was to complete 30 simulated days in 8 hours.

Slewth was written in portable C-code, and runs on Linux and Mac OS X. On a MacBook Pro laptop, it completes the same 100 day simulation in about 35 minutes; on a MacPro desktop it takes about 18 minutes. Because of its fast performance, we recently ran several mission-length simulations, spanning 6.5 years of simulated time. The mission-length simulation finishes in about 6 hours on the MacPro desktop. These test cases are not merely a *tour de force*: among other uses, they enable studies to optimize the strategy for switching between the three IPS engines.

#### IV.F. Verification of Results

Confidence in Slewth results from its strong correlation with benchmark test cases run on Dawn’s realtime testbed facility. This facility is the benchmark facility used to validate the legitimacy of all uplink products radiated to the spacecraft. All Slewth results were matched well within acceptance limits (to within the repeatability of the testbed runs themselves).

Flight experience with Slewth results continues to boost confidence, validating both the tool and the “lean is better” design approach.

### V. Slewth Inflight Experience

Slewth has been used in support of flight activities beginning with pre-launch preparations for Dawn’s initial on-orbit checkout operations and continuing to the present day. We first summarize a brief overview of how Slewth is used in support of flight activities, then relate specific experiences.

#### V.A. Use in Activity Sequence Development

The very first versions of Slewth were as an aid in maneuver design for ACS activities: the ACS team had already identified a need for a software tool that quickly and reliably predicted power steering’s chosen attitude as well as predicting slew durations. Slewth’s interactive capability and fast performance are invaluable

in the design process: the fast turnaround enables the ACS operations team to try various strategies in designing maneuvers, make strong first assessments, and select promising candidates for deeper analysis. Also critical is Slewth's portability to several platforms: all the authors have utilized Slewth to perform predictions while on travel, in meetings, and for on-the-spot assessments when time is short.

These features quickly led to Slewth's being used in a more official capacity for activity development: Slewth enabled the quick assessment of a range of candidate attitude strategies for the full spectrum of planned initial checkout activities, including science and ACS sensor calibrations and the challenging series of three weeklong IPS thruster checkouts.

The IPS checkouts would have been especially challenging without software aid during development. Not only was each activity approximately a week in length, but the IPS checkouts require Earth-based doppler observations along 3 different "sightlines" into the spacecraft (as viewed from the spacecraft perspective). Simply specifying three different inertial thrust directions is not sufficient: recall that the ACS Power Steering software is free to rotate the spacecraft about the aiming vector (in this case the inertial thrust direction) in order to optimize power. Such optimizations would often contrive to present virtually the same spacecraft aspect to the Earth, or result in attitudes at which comm could not be achieved (and hence no doppler data). Slewth was used to identify attitudes that would suffice to provide good doppler observability, good communications, and comply with all known pointing constraints.

With Dawn's launch slip from July to September, all the designs for these activities were rendered obsolete at a stroke, requiring re-design for the new sun-earth geometry pertaining to the later launch date. Slewth was once again put to use.

Currently Slewth is again being used in a major design role, supporting the development of the upcoming Mars Gravity Assist activity in February 2009.

## V.B. Use in Activity Sequence Verification

Slewth also sees use as part of the official verification process for every major activity undertaken on the spacecraft since launch. This includes the initial checkout activities cited above, as well as recurring cruise thrust activities. Slewth has also been used to verify pointing for several special activities the Dawn mission has undertaken to better understand the spacecraft's thermal behavior and its solar array performance (the latter being crucial to maximizing thrust and so also preserving thrust margin).

With the help of additional front-end and back-end software written by the authors, existing uplink review products are converted into a series of Slewth commands and run through the Slewth software. Slewth's outputs are similarly parsed and converted into output products that adhere to standard formats required by the Dawn navigation team, science team, the thermal management team, and the attitude control team. These products are then fed into other operations tools to refine navigation predictions, predict momentum accrual due to solar pressure, design the momentum desat strategy for the next month's activities and drive visualization tools that render the next month's activities in form easily interpreted by the ground team. A sample of some of the output products for recent inflight activity is shown in Figure 7.

## V.C. Real world examples of Slewth in action

### **I would much rather choose a simpler example.**

Slewth's quick turnaround time coupled with the ability to define new pointing constraints "on the fly" have been particularly valuable in dealing with the usual problems experienced in initial checkout, as new inflight experience suggests that pre-launch constraints may have been too strict or (in some cases) not strict enough. This capability has been particularly useful in dealing with uncertainty in the thermal modeling of the Dawn spacecraft. We here present the authors' experience with the last of the IPS checkout activities as a telling case in point.

As described earlier the IPS checkout activities required three different spacecraft-Earth relative attitudes for good doppler residual observability. This amounts to placing the spacecraft-Earth vector in three different locations in the spacecraft frame. The first of positions, dubbed "Earthline" is such that the IPS thrust vector is aligned primarily along the line-of-sight from the spacecraft to the Earth. The other two thrust attitudes (dubbed "N1" and "N2") are chosen so that the thrust vector is as nearly orthogonal to the earthline as possible while also preserving communication and honoring all pointing constraints. Under the normal checkout plan, while on the Earthline attitude the IPS engine is slowly "baked out" for about 8 hours and then thrust is ramped up from low thrust to maximum thrust (91 mN) over a period of 20 hours.

```

2008-175T17:09:16.318:
  turn #3: commencing TargInertial mode slew to targetspec: bvec = HGA; ivec = iBody1 (Earth)
  Aiming bvec          aim_b = [ 1.000000 -0.000219 -0.000391]
  Slew displacement    Theta = 3.000788 rads (171.932473 degs)
  Slew duration        TurnTime = 2303.40 secs
  SlewMode 1 RateProf 2 Slew.Euler_b = [ 0.976657 0.068966 -0.203433]
  targeted final quat  = [-0.536021 0.189038 -0.117666 0.814310]
  targeted final sun position sun_bf = [ 0.842492 0.000000 0.538710]
  targeted final earth position earth_bf = [ 1.000000 -0.000219 -0.000391]
  targeted final angles:
    +ZLGA_Earth = 90.02 degs   +X_Sun = 32.60 degs
    +XLGA_Earth =  0.03 degs   +Y_Sun = 90.00 degs
    -ZLGA_Earth = 89.98 degs   +Z_Sun = 57.40 degs

```

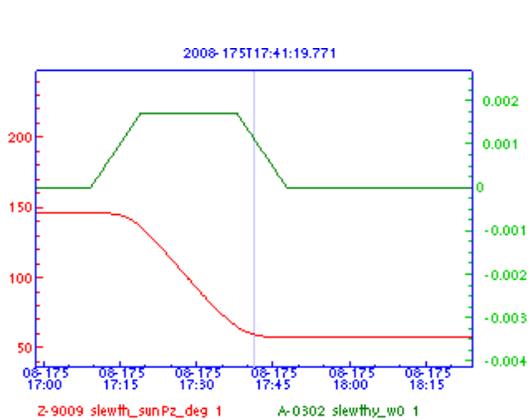
(a)

```

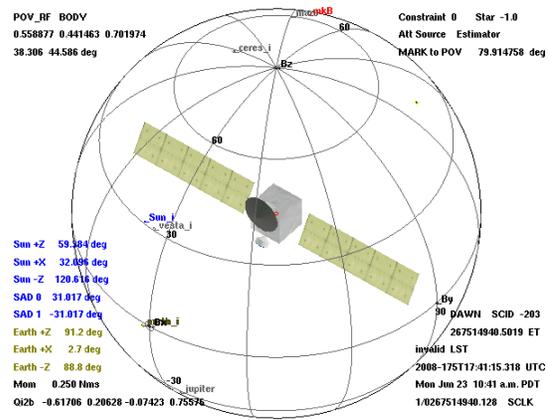
2008-175T17:40:40.118: Constraint (8:VirBorePerf) changed status: now in VIOLATION.
2008-175T22:56:25.123: Constraint (8:VirBorePerf) changed status: now in COMPLIANCE.
                          Violation duration 18945.000 seconds.
2008-183T19:36:16.285: Constraint (8:VirBorePerf) changed status: now in VIOLATION.

```

(b)



(c)



(d)

Figure 7. Sample of output of Slewth products, with visualizations. (a) Turn report excerpt showing sample turn starting at 2008-175T17:09:16 and useful facts. The turn lasts 2303 seconds. Note the final sun angle reported at 57.40°. (b) Violation report excerpt, filtered for the VIR boresight performance constraint (maintain sun  $\geq 60^\circ$  from +z axis). The first violation occurs during the listed sample turn. It clears some 5.25 hours later, and immediately reports compliance (since  $t_{\text{discharge}} = 0$  for this constraint). (c) Plot of spacecraft rate (top, right scale) and sun angle from +z axis (bottom, left scale). (d) Rendering of Dawn spacecraft shown *inside* celestial sphere. Note the sun vector sitting just above the 30° line (i.e., within 60° from +z), which violates the VIR constraint.

After shutting down, the spacecraft returns to a “background” attitude chosen for its favorable thermal and comm characteristics to relay telemetry. The spacecraft then slews to the N1 attitude and begins thrusting at 91mN for a period of about 4 hours. This is repeated at the N2 attitude, and then data is returned to Earth. The attitudes depend significantly on the time chosen for the activity, since the location of the spacecraft-Earth vector is crucial.

Design for last IPS checkout, dubbed “flight thruster Charlie” (FTC) began on September 6, 2007 in the final weeks prior to launch. The targeted execution time was 53 days away, on October 29. This particular IPS checkout activity also had a special goal: to characterize Dawn’s thrust vector control when combined with the hydrazine reaction control system (RCS) thrusters as opposed to the reaction wheel. Consequently the activity required a second burn at the N2 attitude to test this control mode. We used Slewth to generate predicts per this original design.

However, after the spacecraft launch on September 27, numerous issues realized during early spacecraft checkout forced a series of delays upon the FTC activity. The most serious of these were the delays imposed after a incident the week of October 14, during the preceding checkout of the *second* IPS engine. This incident resulted in a two week delay to FTC—too long a delay for the existing design to withstand.

We began redesign on October 22, with a target execution date 21 days later on November 12. This execution date presented a much more challenging geometry: though it was comparatively easy to find safe

thrusting attitudes it was difficult to find a trio of safe attitudes that would meet doppler observation requirements. As the re-design continued, concerns developed that the then-in-use (and planned-for) background attitude might not be thermally safe due to excessive solar heat input into the bond material used to mount the RCS thrusters.

Those concerns and another schedule slip compelled a third design effort, begun October 25, with a target date of November 14 (20 days away). This design effort also included selecting a new background attitude. Slewth was used to identify the new background attitude that would be thermally safe. Furthermore, many believed that this late execution date rendered the doppler observation problem intractable. We used Slewth to identify and demonstrate a suite of attitude solutions that, while not optimal, were acceptable for doppler observation. These attitudes tended to place the Sun almost along the spacecraft  $+z$  axis, but the existing flight constraints in effect at the time indicated this was a safe posture provided the science instrumentation remained powered off.

On October 30, the science team's assessment of their instruments' thermal behavior during prior activities raised fresh concerns about our proposed attitude strategy. On the same day, the thermal team raised concerns about the gyros overheating at these attitudes: the thermal model uncertainty was such that it was unclear the gyros would be safe even if powered off.

Consequently, we began a new design on November 3, with the activity now scheduled for execution on November 13 (10 days away). Slewth revealed that under the new effective pointing constraints, only earthline attitudes could be found. Once again thermal concerns began to crop up: new IPS temperature data suggested that the earthline attitude we deemed best (which placed the sun where it could shine directly onto the IPS gimbal mechanism) might now be thermally risky at high thrust levels. Low thrust levels were considered fine.

On November 5, with the activity only 8 days away, we decided to split the problem in three parts: since the only N1 and N2 attitudes that could be found were considered thermally unsafe, we deferred these "off-thrust-axis" observations until the Spring of 2008. As for the earthline observations, we split them into two components: the low-thrust portion would be conducted at the original earthline attitude with good comm and good visibility; for the high-thrust portion we would turn the spacecraft around and thrust towards Earth, placing the sun  $\sim 17^\circ$  off the  $+z$  axis. This latter attitude still required the gyros to be powered off, but no longer threatened the science instruments. We also planned the almost-cancelled TVC/RCS control functionality test at this latter attitude.

This final plan was run through Slewth, and approved by management on November 9th. We completed development of final uplink products on the 11th. We began execution of the modified FTC activity on the spacecraft itself on November 13, successfully concluding with the TVC/RCS functionality test on November 16. Slewth was instrumental in keeping this activity within the reach of possibility.

## VI. Conclusions

We have presented Slewth, an ACS ground tool that has been successfully used to predict and verify the pointing of the Dawn spacecraft in support of pre-Launch planning, initial on-orbit checkout, and ongoing mission activities. It continues to remain the baseline verification tool for future activities.

Future versions of Slewth will provide improved modeling of flight software behaviors in support of science operations during the science-return phases of the Dawn mission (when the spacecraft orbits Vesta and subsequently Ceres). Some of the constraints for Dawn are more accurately modeled as compound constraints composed of wedges and melds,<sup>5</sup> but the current version of the Slewth does not as yet handle compound constraints. Slewth also has no awareness of the power states of the science instrument or the IPS engines. Future versions of Slewth may address these factors.

With substitution of the Dawn pointing model for an alternative spacecraft pointing model, Slewth can be extended to provide ground operations constraint checking for other missions.

## Acknowledgments

This work could not have been completed without the assistance of Alex Manka, Ruyi Sun, and Dominic Bruno of Orbital Sciences Corporation; Steve Collins of JPL's guidance and control section, and Ian Roundhill and Don Han of JPL's navigation section. The visualization of the Dawn spacecraft attitude in Figure 7 is courtesy the superb TBALL visualization tool developed by Steve Collins. Their support is gratefully

acknowledged.

This work was prepared at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

## References

<sup>1</sup>Rayman, M. D., Fraschetti, T. C., Raymond, C. A., and Russell, C. T., "Dawn: A mission in development for exploration of main belt asteroids Vesta and Ceres," *Acta Astronautica*, Vol. 58, 2006, pp. 605–616.

<sup>2</sup>Rayman, M. D., "The successful conclusion of the Deep Space 1 mission: important results without a flashy title," *Space Technology*, Vol. 23, 2003, pp. 185–196.

<sup>3</sup>Vanelli, C. A., *Autonomous Reorientation of a Maneuver-Limited Spacecraft Under Simple Pointing Constraints*, Eng. thesis, California Institute of Technology, May 1997.

<sup>4</sup>Rasmussen, R. D., Singh, G., Rathbun, D. B., and Macala, G. A., "Behavioral model pointing on Cassini using target vectors," *Advances in Astronautical Sciences*, Vol. 88, 1995.

<sup>5</sup>Vanelli, C. A. and Ali, K. S., "High gain antenna pointing on the Mars exploration rovers," *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, IEEE, Oct. 2005, pp. 28–33.