

# SOAR INTERFACE CONTROL DOCUMENT

SATTELITE FOR OPTIMAL CONTROL AND IMAGING

Taylor P. Reynolds\*

February 17, 2021

This document is the interface control document for the primary payload of the SOCi mission. The payload is SOCi's Optimal Attitude Reorientation (SOAR) that uses convex optimization techniques to compute guidance trajectories to steer the spacecraft to a desired orientation while satisfying various constraints. Specifically, trajectories are generated that respect angular rate bounds, torque bounds, and attitude inclusion and exclusion zone constraints. The problem is formulated as a nonlinear optimal control problem and solved as a sequence of convex optimization problems using the open source solver ECOS.

This document provides a summary of the system, technical details on how the controller works, the interfaces to other parts of the spacecraft, and the verification and validation procedure used to ensure proper operation. I have stashed all of the data tables at the back to keep the document readable.

## 1 Summary of System

The SOAR payload is a software component within the Guidance, Navigation, and Control (GNC) subsystem. The SOAR payload will produce feedforward reaction wheel torque commands and a reference state vector in order to control the orientation of the spacecraft. All reorientations will be performed starting from the spacecraft's current attitude, and will terminate at a desired attitude. The payload's execution is triggered by the primary GNC software by a change in the operating mode via the signal `gnc_mode`. The inputs to the SOAR function are detailed in §2.4.

A block diagram of the SOAR control system is provided in Figure 1. It must be coupled with a feedback controller (shown as  $C(s)$ ) to compensate for small errors that accumulate during the execution of a feedforward maneuver. Note that the “difference” junction used to compute the state errors is not a subtraction since quaternions are used; however the same symbol is overloaded for simplicity. The feedback controller  $C(s)$  is not designed as part of SOAR, and resides in the primary GNC software.

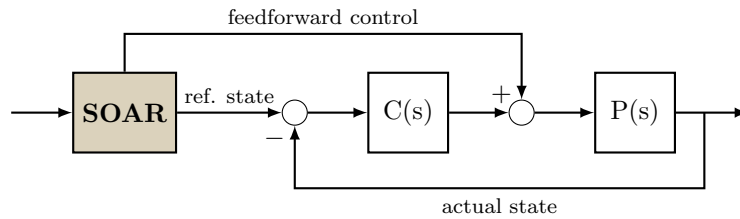


Figure 1: Abstract control loop for the SOAR payload. The controller represented by  $C(s)$  is not designed as part of the payload.

### 1.1 Problem Statement

The problem that is solved by the SOAR payload can be summarized as follows. Define the state and control vectors to be

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \mathbf{h}_B(t) \\ \mathbf{h}_w(t) \end{bmatrix}_{10 \times 1} \quad \text{and} \quad \boldsymbol{\tau}(t) = \begin{bmatrix} \tau_x(t) \\ \tau_y(t) \\ \tau_z(t) \end{bmatrix} \in \mathbb{R}^3 \quad (1)$$

---

\*Ph.D. Candidate, RAIN Lab, W.E. Boeing Department of Aeronautics & Astronautics, University of Washington: [tpr6@uw.edu](mailto:tpr6@uw.edu)

where  $\tau_x, \tau_y, \tau_z$  are the components of the net torque vector about the spacecraft's body axes,  $\mathbf{q} \in \mathbb{R}^4$  is the attitude quaternion,  $\mathbf{h}_B \in \mathbb{R}^3$  is the spacecraft bus angular momentum (in the body frame) and  $\mathbf{h}_w \in \mathbb{R}^3$  is the net reaction wheel angular momentum about the spacecraft body axes.

The equations of motion are assumed to be (suppressing the argument of time)

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{h}}_B \\ \dot{\mathbf{h}}_w \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{q} \otimes (J^{-1}\mathbf{h}_B) \\ -\boldsymbol{\tau} + (\mathbf{h}_B + \mathbf{h}_w)^\times (J^{-1}\mathbf{h}_B) \\ \boldsymbol{\tau} \end{bmatrix}. \quad (2)$$

The problem statement is given below. See the Appendix for an explanation of the constraints and how they are formulated. The cost function aims to minimize the power consumed by the reaction wheels to perform the maneuver. The  $\mathcal{L}_2$  norm of the net torque signal is used to model their power consumption over the maneuver time. The details and specific formulae used to compute solutions to this problem are provided in an Appendix to this document.

$$\begin{aligned} & \min_{t_f, \boldsymbol{\tau}(\cdot)} \int_{t_0}^{t_f} \boldsymbol{\tau}^\top \boldsymbol{\tau}(t) dt \\ \text{subject to: } & (2) \\ & \|J^{-1}\mathbf{h}_B\|_\infty \leq \omega_{\max} \\ & \|\boldsymbol{\tau}\|_\infty \leq \tau_{\max} \\ & \mathbf{q}^T \tilde{M}_e \mathbf{q} \leq 2 \\ & \mathbf{q}^T \tilde{M}_i \mathbf{q} \leq 2 \\ & t_{f,\min} \leq t_f \leq t_{f,\max} \\ & \mathbf{q}(t_0) = \mathbf{q}_{ic}, \quad \mathbf{h}_B(t_0) = \mathbf{h}_{B,ic}, \quad \mathbf{h}_w(t_0) = \mathbf{h}_{w,ic}, \\ & \mathbf{q}(t_f) = \mathbf{q}_f, \quad \mathbf{h}_B(t_f) = 0. \end{aligned} \quad (\text{Problem 1})$$

Note that the fourth and fifth constraints define the attitude exclusion zone (due to the camera) and the attitude inclusion zone (due to the sun sensor) respectively. Details on how this constraint is formulated can be found in [1]. They are each a second-order cone constraint due to their quadratic nature. All other inequality constraints are linear. All inequality constraints are therefore convex, as are the boundary conditions (the final two lines of constraints) and the cost function. The nonlinear dynamics are the only non-convexity in Problem 1.

## 2 System Interfaces

### 2.1 System Overview


A block diagram representation of the overall system is given in Figure 2. The SOAR library takes nine inputs and provides two outputs; details of each are provided in Table 2. The main function of the library is to use inputs 2-7 to define an instance of Problem 1, and then solves said problem for a trajectory (state, control and maneuver time) that will reorient the satellite while respecting each constraint.

The SOLVE function contains the custom-written parser and interface to the ECOS solver. When called upon, it will parse the data (inputs 2-7) into a sequence of convex optimization problems that approximate the non-convex Problem 1. The (discrete) solution that is computed then gets passed to the INTERP function so that it may be transformed to a continuous time reference state and control vectors using the same sampling time as the primary GNC flight software. Since the discretization scheme used in the SOLVE function will necessarily be a much coarser one, this is effectively a discrete-to-continuous transformation and uses a fixed step-size RK4 integration scheme.

The components of the `soar_telemetry` struct are given in Table 3. The first three are the converged solution from the SOLVE function. The first fifteen entries in `exitcode` are the ECOS solver return values for each of the (maximum) fifteen iterations. The 16th and 17th are signals that provide additional information about the computed solution. The possible values are given in Table 4. The `slv_time_s` is the ECOS-reported time taken to solve each iteration's convex optimization problem. This time is *setup and solve* time, and is computed in the `matlab_main.c` function. Finally, the `soar_count` is an integer that indicates how many times (in total) the SOLVE function has been called.

#### 2.1.1 Constraint Boundary Values

The constraint boundaries have the following values, each of which are configurable parameters:

 sections/media/"aactLogo".png

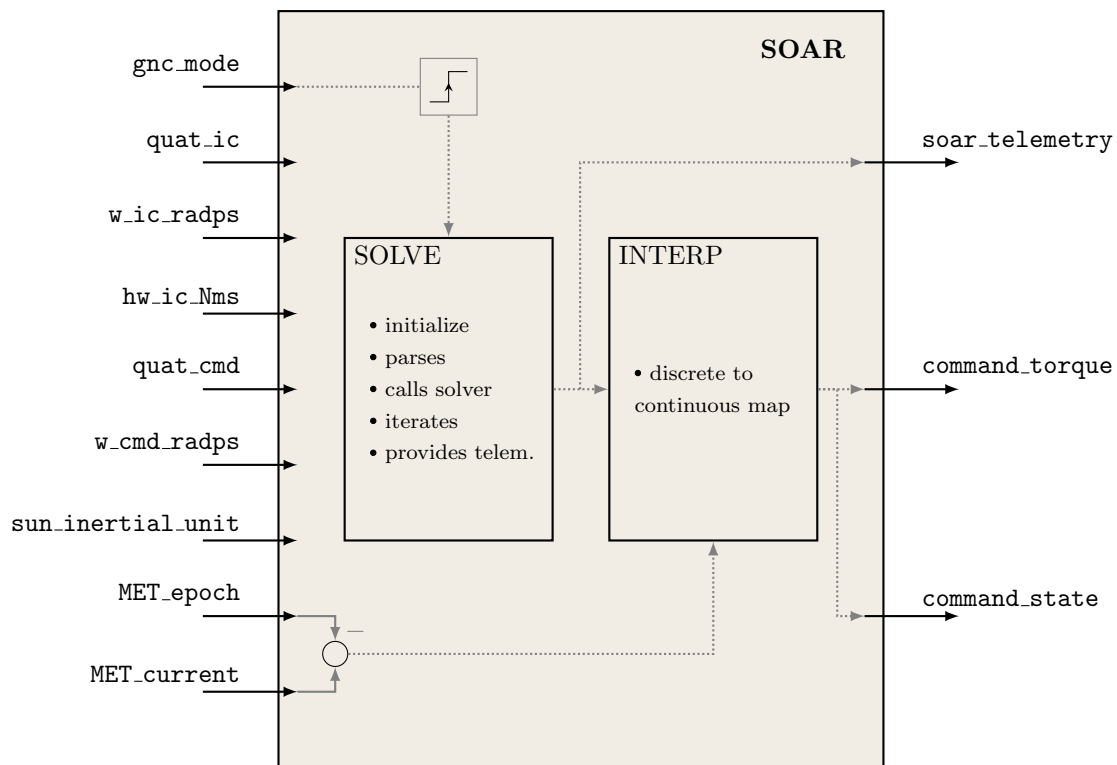


Figure 2: A representation of the input/output structure of the SOAR payload and its two primary functional components. The input `gnc_mode` triggers the execution of the *SOLVE* block which pulls in all unconnected inputs.

1. an angular rate limit of  $\omega_{\max} = 0.1$  rad/s,
2. a torque limit of  $\tau_{\max} = 3.2$  mNm,
3. a final time between  $t_{f,\min} = 15$  s and  $t_{f,\max} = 25$  s,
4. the inertial sun vector remains within a  $\theta_{\max} = 60^\circ$  degree cone (half-angle) centered around the sun sensor boresight direction,
5. the inertial sun vector remains outside of a  $\theta_{\min} = 60^\circ$  degree cone (half-angle) centered around the camera boresight direction.

The torque limit is below what can be produced by the reaction wheels, and was selected to equal the capability of a single wheel. This provides a buffer so that the addition of feedback control will not saturate the wheels, and also means that if a single wheel were to fail, the SOAR payload will not ask for a torque that exceeds the reduced capability with no need to change the code.

### 2.1.2 SOAR Operating Modes

There are effectively two operating modes of the SOAR payload: operational and idle. The trigger attached to the `gnc_mode` shown in Figure 2 is the arbiter between the two modes. The output of the trigger goes high when the following conditions are met:

1. The value of `gnc_mode` = 33,
2. The previous value of `gnc_mode` != 33.

The SOAR payload can be commanded to its operational mode by the primary GNC mode manager by switching the value of `gnc_mode` to 33 from any other value. The conditions upon which the primary GNC operating mode is set to 33 are provided in the [main GNC ICD](#). Maintaining the primary GNC mode at this value will not continuously trigger the payload due to the second clause. As a result, the primary GNC mode must be toggled back to some value that is not 33 prior to requesting another solution. **The primary GNC flight software is assumed to be in charge of requesting a re-solve for a given maneuver.** The current design is such that triggering a re-solve internally will create an algebraic loop.

A brief attempt was made to design internal re-solves into the library, but a decision was made that it would be simpler to manage this within the primary GNC mode manager.

### 2.1.3 Control Torque Allocation

The equations of motion used in the SOAR system are given in (2). Since we are using a four-wheel system, a note is needed on what assumptions are being made about how the torques computed by SOAR should be allocated to the wheels. The SOAR payload computes the **net** torque about the spacecraft body axes that should be applied to perform the maneuver, while accounting for the **net** momentum in the wheels about the same body axes.

Let  $\mathbf{H} \in \mathbb{R}^4$  be the vector whose components are the angular momenta stored in each wheel and let  $\mathbf{T} \in \mathbb{R}^4$  be the vector whose components are the torques delivered by each wheel. Assuming that the net angular momentum of the wheels,  $\mathbf{h}_w$ , is related to the momentum of each wheel,  $\mathbf{H}$ , according to some map  $g : \mathbb{R}^4 \rightarrow \mathbb{R}^3$ , we have

$$\mathbf{h}_w = g(\mathbf{H}). \quad (3)$$

It should be clear that the net torque,  $\boldsymbol{\tau}$ , is related to the torque given by each wheel,  $\mathbf{T}$ , according to the same relationship.

This section sets some ground rules for how the map  $g$  is chosen, and offers a possible choice. The equations of motion (2) should actually read

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{h}}_B \\ \dot{\mathbf{H}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\mathbf{q} \otimes (J^{-1}\mathbf{h}_B) \\ -g(\mathbf{T}) + (\mathbf{h}_B + g(\mathbf{H}))^\times (J^{-1}\mathbf{h}_B) \\ \mathbf{T} \end{bmatrix}, \quad (4)$$

for an *eleven-dimensional* state  $\mathbf{x} = [\mathbf{q}^\top \mathbf{h}_B^\top \mathbf{H}^\top]^\top$ . These equations of motion come from the conservation of momentum combined with the four-wheel geometry. The equations in (4) are *equivalent* to those in (2) if and only if

1. There exists a map  $g^\dagger : \mathbb{R}^4 \rightarrow \mathbb{R}^3$  such that  $\mathbf{z} = g(g^\dagger(\mathbf{z}))$  for any  $\mathbf{z} \in \mathbb{R}^3$ .
2.  $\frac{d}{dt}g(\mathbf{z}(t)) = g(\dot{\mathbf{z}}(t))$  for any  $\mathbf{z}(t) \in \mathbb{R}^4$ .

Together these conditions should ensure that no matter what allocation scheme is used downstream of the SOAR payload, the intended behaviour will be recovered.

**An Example Allocation Method:** Suppose that the net momentum is related to the momentum in each wheel by

$$\mathbf{h}_w = \begin{bmatrix} \cos \beta & 0 & -\cos \beta & 0 \\ 0 & \cos \beta & 0 & -\cos \beta \\ \sin \beta & \sin \beta & \sin \beta & \sin \beta \end{bmatrix} \mathbf{H} = A_w \mathbf{H} \quad (5)$$

The  $3 \times 4$  matrix  $A_w$  is singular, but following [4, §7.3.4] the pseudo-inverse can be used to write

$$\mathbf{T} = A_w^\dagger \boldsymbol{\tau} \quad \text{and} \quad \mathbf{H} = A_w^\dagger \mathbf{h}_w \quad (6)$$

where  $A_w^\dagger = A_w^\top (A_w A_w^\top)^{-1} \in \mathbb{R}^{4 \times 3}$  is such that  $A_w A_w^\dagger = I_3$ . Thus the mapping  $g(\mathbf{T}) = A_w \mathbf{T}$  satisfies the first condition required above (existence of an “inverse” like function). Next, it is clear that the second condition is satisfied because the matrix  $A_w$  is constant. This allocation scheme will ensure that SOAR produces the expected behaviour, which has been verified in simulation by mapping an optimal set of controls  $\boldsymbol{\tau}(t)$  to  $\mathbf{T}(t)$  for  $t \in [0, t_f]$  and then integrating the equations of motion (4). The resulting state vector matched exactly the output of SOAR.

### 2.1.4 Building & Autocoding

You need to build the SOAR executable before you can run a unit test or use the SOAR software. The building/autocoding process that is used for SOAR was developed in [2]. First, make sure your current MATLAB directory is `ROOT/adcs/sw/components/adcs_oac/build`, where `ROOT` is whatever directory the repository is stored in. Assuming that you have correctly run `mex -setup` at least once on your machine, you can type `build_soar` at the MATLAB command line. This will take a few seconds and you should see lots of output as components get built. There will be one warning about an implicit function declaration but this is safe to ignore. This process will create three files in the `build` folder, and you may now run one of the unit tests and/or use the SOAR library in simulation.

**Note:** The `.gitignore` for the repository is set up to ignore the autogenerated mex files that this process creates. You will need to rebuild the files on any new machine that you use.

To autocode the SOAR library, or anything that contains it, you need to link the external solver files located in `ROOT/adcs/sw/components/adcs_oac/include/ecos/`. This must be done with the full path from your machine's actual root directory. You need to provide Simulink with the include directory, which is `ROOT/adcs/sw/components/adcs_oac/include/ecos/include` and the source files (\*.c) from the `ROOT/adcs/sw/components/adcs_oac/include/ecos/src/` folder. Figure 3 has an example of what the path should look like. The window shown in Figure 3 is found by clicking on

Code -> C/C++ Code -> Code Generation Options...

Once these files are linked, you should be able to use the Code Generation Quick Start feature to generate the C code that will be handed to the CDH team.

## 2.2 Mechanical Interface

The SOAR payload is a software function that will be run on the main flight computer. There is no separate mechanical interface.

## 2.3 Electrical Interface

The SOAR payload is a software function that will be run on the main flight computer. There is no separate electrical interface.

## 2.4 Command and Data Interface

All inputs listed in Table 2 will come from the main GNC flight software. Refer to [main GNC ICD](#).

The output `SOAR_solution` must be saved for future downlink to the ground station. This has been included in the COM system's measurement budget, and shall be maintained by CDH immediately once it is made available by SOAR. The two remaining outputs are to be provided as the GNC guidance solution and feed forward to the feedback controller that is used during primary GNC mode 33.

## 2.5 Thermal Interface

The SOAR payload is a software component that will be run on the main flight computer. There is no separate thermal interface. However, running the SOAR payload may cause an marked increase in the flight computer's computational load, and therefore it may produce more heat than nominal operating conditions. By calling SOAR only once (instead of repeatedly, see §2.1.2) it is hoped that SOAR will not create a large difference in thermal properties than what is expected for the main flight computer.

# 3 Verification and Validation

This section outlines the testing campaign that SOAR has so far passed during design, as well as a test case that can be used to assess correct functionality during a bench test. The following tests have been made:

1. A unit test of just the SOLVE function, found in the folder

`.../adcs_oac/test/soar_unit_test`

2. A unit test of the entire SOAR library, found in the folder

`.../adcs_oac/test/soar_interface_unit_test`

3. A Monte Carlo test of the entire SOAR library, found in the folder

`.../adcs_oac/test/soar_mc_test`

"figs/" "autocode\_include".png

"figs/" "autocode\_srcs".png

"sections/media/" "aactLogo".png

Table 1: Monte Carlo results on 2014 MacBook Pro with a 2.2 GHz Intel Core i7 processor and 16GB RAM. Note that this is **not** flight hardware, so the timing values are not representative of OBC performance.

Metric	Value
Total trials	1000
Successful trials	1000
Failed trials	0
Maximum degree error	2.22°
Median degree error	0.4672°
Maximum solver time	65.2 ms
Median solver time	19.3 ms
Maximum number of iterations	13
Median number of iterations	4

Each folder contains a Matlab script and a Simulink library. Running the Matlab script will run the test and plot some results. The other test folders are not unit tests of the SOAR library in its Simulink form and can be ignored at this point (they were mostly used for development).

A 1000 trial Monte Carlo test was run to assess the performance of the SOAR library at the preliminary design stage. The commanded attitude and angular rates were set to  $\mathbf{q}_{cmd} = (1, 0, 0, 0)$  and  $\boldsymbol{\omega}_{B,cmd} = (0, 0, 0)$  for each trial. The initial attitude was chosen by selecting a random direction on the unit sphere as an eigenaxis, and a error angle uniformly sampled from the interval  $[-90^\circ, 90^\circ]$ . The initial quaternion is then

$$\theta \sim \mathcal{U}(-90^\circ, 90^\circ), \quad \hat{\mathbf{n}} = \frac{(x, y, z)}{\|(x, y, z)\|}, \quad x, y, z \sim \mathcal{N}(0, 1), \quad \mathbf{q}(t_0) = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\hat{\mathbf{n}}) \quad (7)$$

The initial attitude rates were chosen from

$$\boldsymbol{\omega}_B(t_0) \sim \mathcal{N}(0_{3 \times 1}, \Sigma_\omega), \quad \Sigma_\omega = \mathbf{diag}\{10^{-3}, 10^{-3}, 10^{-2}\} \quad (8)$$

and the initial reaction wheel spin rates (in units of RPM) were chosen from

$$\Omega(t_0) \sim \mathcal{N}(\Omega_0, \Sigma_\Omega), \quad \Omega_0 = \begin{bmatrix} 1000 \\ -1000 \\ 1000 \\ -1000 \end{bmatrix}, \quad \Sigma_\Omega = 50I_4 \quad (9)$$

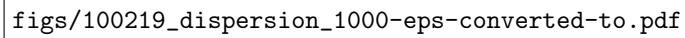
which was then mapped to  $\mathbf{h}_w(t_0)$  to serve as the initial condition.

The result of a single trial was assessed first by the value of the `exitcode` from SOAR. If the 16th entry was zero, the trial was deemed a success; any other value was deemed a failure, see Table 4. Next, the resulting feedforward control/state were integrated through the spacecraft's nonlinear equations of motion using the dynamics that are being used to design the primary GNC flight software. A notional feedback controller was added to provide a sense of the closed-loop capabilities. The total degree error *at the end of the maneuver* (i.e., the optimal time  $s^* = t_f^*$ ) between  $\mathbf{q}(t_f^*)$  and  $\mathbf{q}_{cmd}$  was computed using  $\mathbf{q}_{err} = \mathbf{q}(t_f^*)^* \otimes \mathbf{q}_{cmd}$  and

$$\theta_{err} = 2 \arccos(q_{err,0}). \quad (10)$$

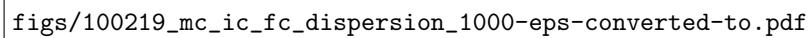
This measures the attitude error at the end of the maneuver *only*, and does not provide information as to how well the path was tracked for times  $t \in [t_0, t_f^*]$ . Small values of  $\theta_{err}$  are necessary but not sufficient to indicate satisfactory performance. If  $\theta_{err}$  is small and there is no constraint violation for any time  $t \in [t_0, t_f^*]$ , then the maneuver is satisfactory at the preliminary design stage.

Key metrics from the 1000 trials are given in Table 1, and are visualized in Figures 4 and 5. No constraint violations were observed during closed loop simulations, and all trials achieved a final degree error of less than  $3.5^\circ$ , the required control error bound per [GNC Requirement 4.2.10](#).



figs/100219\_dispersion\_1000-eps-converted-to.pdf

(a) Initial and final angle distributions



figs/100219\_mc\_ic\_fc\_dispersion\_1000-eps-converted-to.pdf

(b) Final versus initial angle errors

Figure 4: Closed loop results from 1000 randomized unit tests of SOAR. All final errors are within the required  $3.5^\circ$  control error bound.

"sections/media/"aactLogo".png



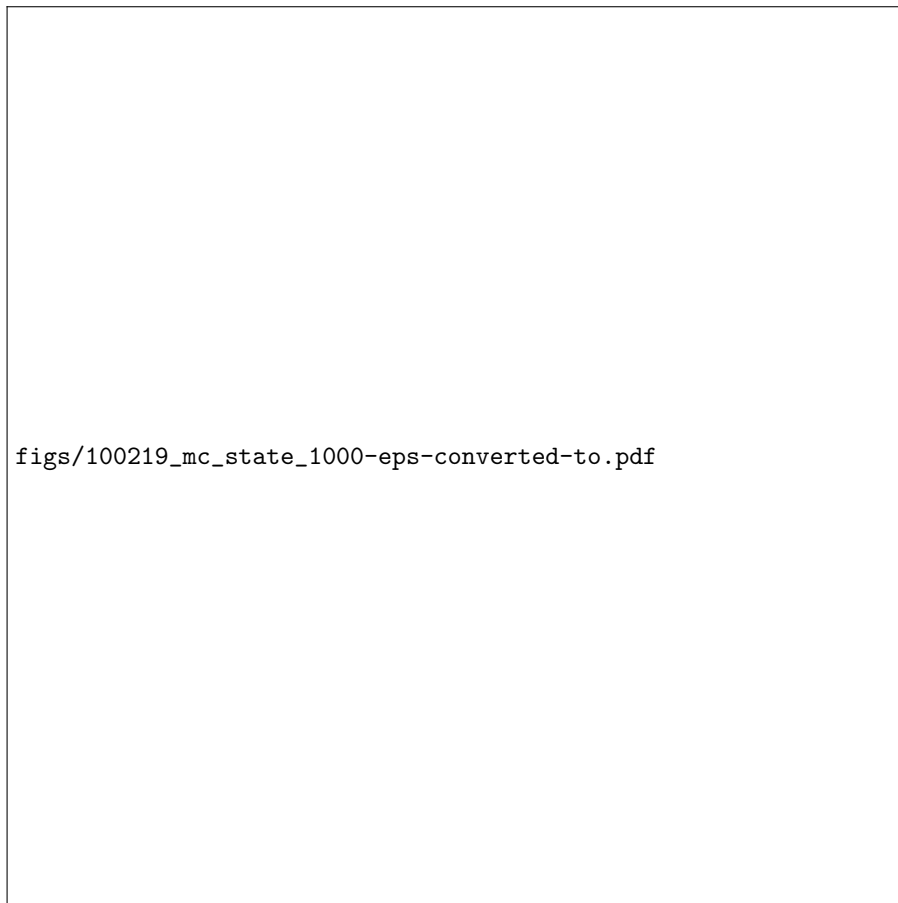


Figure 5: Closed loop results from 1000 randomized unit tests of SOAR. Dashed red lines indicate constraint boundaries.

## Reference Tables

Table 2: Description of the SOAR payload's inputs and outputs

<b>Inputs</b>				
	<b>Name</b>	<b>Size</b>	<b>Type</b>	<b>Description</b>
1	<code>gnc_mode</code>	<i>scalar</i>	<code>int8</code>	The current GNC system operating mode.
2	<code>quat_ic</code>	$4 \times 1$	<code>double</code>	The current estimated attitude of the s/c.
3	<code>w_ic_radps</code>	$3 \times 1$	<code>double</code>	The current estimated angular velocity of the s/c.
4	<code>hw_ic_Nms</code>	$3 \times 1$	<code>double</code>	The current momentum vector of the reaction wheels.
5	<code>quat_cmd</code>	$4 \times 1$	<code>double</code>	The desired attitude at the end of the maneuver
6	<code>w_cmd_radps</code>	$3 \times 1$	<code>double</code>	The desired angular velocity at the end of the maneuver
7	<code>sun_inertial_unit</code>	$3 \times 1$	<code>double</code>	The inertial sun direction at the current time. This is assumed to be constant over the duration of the maneuver.
8	<code>MET_epoch</code>	<i>scalar</i>	<code>double</code>	The maneuver epoch time in units of mission elapsed time (MET) provided via ground command prior to maneuver execution.
9	<code>MET_current</code>	<i>scalar</i>	<code>double</code>	The current value of MET.
<b>Outputs</b>				
	<b>Name</b>	<b>Size</b>	<b>Type</b>	<b>Description</b>
1	<code>soar_telemetry</code>	$153 \times 1$	<code>double</code>	The entire output from the SOAR iterations. This should all be routed to ground for post-processing. See Table 3.
2	<code>command_state</code>	$10 \times 1$	<code>double</code>	The interpolated state vector to be used as a reference for the feedback control system.
3	<code>command_torque</code>	$3 \times 1$	<code>double</code>	The interpolated control vector to be used as a feedforward actuator command.

Table 3: The components of the `soar_telemetry` struct.

<b>Name</b>	<b>Size</b>	<b>Type</b>
<code>opt_state</code>	100	<code>double</code>
<code>opt_ctrl_Nm</code>	30	<code>double</code>
<code>final_time_s</code>	1	<code>double</code>
<code>exitcode</code>	17	<code>double</code>
<code>slv_time_s</code>	15	<code>double</code>
<code>soar_count</code>	1	<code>uint32</code>

Table 4: Possible values and meanings of `exitcode`.**From ECOS: Entries 1-15**

Value	Description
0	Optimal solution found
1	Certificate of primal infeasibility found
2	Certificate of dual infeasibility found
10	Optimal solution found subject to reduced tolerances
11	Certificate of primal infeasibility found subject to reduced tolerances
12	Certificate of dual infeasibility found subject to reduced tolerances
-1	Maximum number of iterations reached
-2	Numerical problems: unreliable search direction
-3	Numerical problems: slacks or multipliers outside cone
-4	Interrupted by signal or CTRL-C
-7	Unknown problem in solver

**Custom: Entry 16**

Value	Description
1-15	Number of global iterations taken

**Custom: Entry 17**

Value	Description
0	All iterates found optimal solution and the terminal iterate meets the convergence criteria
-1	One of the iterates did not return a 0 or 10
3	Maximum number of iterations reached: virtual control too large
4	Maximum number of iterations reached: state change too large

## Appendix: Technical Details

A general convex optimization problem is expressed

$$\min_{\mathbf{z}} f_0(\mathbf{z}) \quad (11a)$$

$$\text{subject to: } h_i(\mathbf{z}) = 0, \quad i = 1, \dots, n_{eq} \quad (11b)$$

$$f_j(\mathbf{z}) \leq 0, \quad j = 1, \dots, n_{iq} \quad (11c)$$

where  $\mathbf{z} \in \mathbb{R}^{n_z}$  denotes the *solution variable*,  $f_0$  is the cost function, the  $h_i$  are the *equality constraints* and the  $f_j$  are the *inequality constraints*. The functions  $f_j$  for  $j = 0, 1, \dots, n_{iq}$  must be *convex functions* of the variable  $\mathbf{z}$ , while the functions  $h_i$  for  $i = 1, \dots, n_{eq}$  must be *affine functions* of the variable  $\mathbf{z}$ .

### Convex Cones

Each proper cone  $\mathcal{K}$  defines a *generalized inequality* such that we may write  $G\mathbf{z} \preceq_{\mathcal{K}} h$  if and only if  $h - G\mathbf{z} \in \mathcal{K}$ . In particular we shall care about two cones, the *linear cones* and the *quadratic cones*. Linear cones are denoted by the set

$$\mathbb{R}_+^n = \{\mathbf{z} \in \mathbb{R}^n \mid \mathbf{z} \geq 0\}, \quad (12)$$

where the inequality is understood elementwise. The other type of convex cones that we use are the second order cones, each of which are a set  $\mathcal{Q}^{n+1}$  defined by

$$\mathcal{Q}^{n+1} = \{(z_0, \mathbf{z}_n) \in \mathbb{R} \times \mathbb{R}^n \mid z_0 \geq \|\mathbf{z}_n\|_2\}. \quad (13)$$

To see how these are used to express various constraints as second order cone constraints, first note that a standard quadratic form is expressed as

$$\mathbf{x}^T A^T A \mathbf{x} + b^T \mathbf{x} + c \leq 0 \quad (14)$$

where  $(A, b, c) \in \mathbb{R}^{n \times n} \times \mathbb{R}^n \times \mathbb{R}$  are constant constraint parameters. In standard form, this constraint is expressed by

$$\left\| \begin{bmatrix} A\mathbf{x} \\ \frac{1}{\sqrt{2}}(1 + c + b^T \mathbf{x}) \end{bmatrix} \right\|_2 \leq \frac{1}{\sqrt{2}}(1 - b^T \mathbf{x} - c), \quad (15)$$

The proof of which follows from some straight-forward algebra. Now observe that according to the definition in (13) we would like

$$\begin{bmatrix} \frac{1}{\sqrt{2}}(1 - b^T \mathbf{x} - c) \\ A\mathbf{x} \\ \frac{1}{\sqrt{2}}(1 + b^T \mathbf{x} + c) \end{bmatrix} \in \mathcal{Q}^{n+2}$$

It should be clear that the first entry in the vector above is equivalent to  $z_0$  in (13), and the remaining rows are equivalent to  $\mathbf{z}_n$ . Now, ECOS would like the matrix-vector pair  $(G, h)$  such that  $h - G\mathbf{x} \in \mathcal{Q}^{n+2}$ , and so we set

$$h := \begin{bmatrix} \frac{1}{\sqrt{2}}(1 - c) \\ 0_{n \times 1} \\ \frac{1}{\sqrt{2}}(1 + c) \end{bmatrix} \quad G := \begin{bmatrix} \frac{b^T}{\sqrt{2}} \\ -A \\ -\frac{b^T}{\sqrt{2}} \end{bmatrix} \quad (16)$$

The pair  $(G, h)$  may be passed to ECOS to represent the quadratic constraint (14).

### Continuous Problem Formulation

The continuous time optimal control problem to be solved in 1 and is repeated here.

**Problem 1.** Find the time  $t_f^* \in \mathbb{R}_{++}$  and torque commands  $\boldsymbol{\tau} : [t_0, t_f^*] \rightarrow \mathbb{R}^3$  such that

$$\min_{t_f, \boldsymbol{\tau}(\cdot)} \int_{t_0}^{t_f^*} \boldsymbol{\tau}^\top \boldsymbol{\tau} dt \quad (17a)$$

$$\text{subject to } \dot{\mathbf{q}} = \frac{1}{2} \mathbf{q} \otimes \boldsymbol{\omega}_{\mathcal{B}} \quad (17b)$$

$$\dot{\mathbf{h}}_{\mathcal{B}} = \boldsymbol{\tau} + (\mathbf{h}_{\mathcal{B}} + \mathbf{h}_w)^\times (J^{-1} \mathbf{h}_{\mathcal{B}}) \quad (17c)$$

$$\dot{\mathbf{h}}_w = -\tau \quad (17d)$$

$$\|\boldsymbol{\omega}_B\|_\infty \leq \omega_{\max} \quad (17e)$$

$$\|\mathbf{u}_B\|_\infty \leq u_{\max} \quad (17f)$$

$$\mathbf{q}^T \tilde{M}_e \mathbf{q} < 2 \quad (17g)$$

$$\mathbf{q}^T \tilde{M}_i \mathbf{q} < 2 \quad (17h)$$

$$t_{f,\min} \leq t_f \leq t_{f,\max} \quad (17i)$$

$$\mathbf{q}(t_0) = \mathbf{q}_{ic}, \quad \boldsymbol{\omega}_B(t_0) = \boldsymbol{\omega}_{B,ic}, \quad \mathbf{h}_w(t_0) = \mathbf{h}_{w,ic}, \quad (17j)$$

$$\mathbf{q}(t_f) = \mathbf{q}_f, \quad \boldsymbol{\omega}_B(t_f) = \mathbf{0}. \quad (17k)$$

## Transcription to Second-Order Cone Program

This section details the steps necessary to obtain a convex second-order cone approximation of 1 that is solved repeatedly, until convergence, within the SOLVE block of Figure 2.

**Time Normalization** We define our *state* to be  $\mathbf{x} = [\mathbf{q} \ \boldsymbol{\omega}_B]^T \in \mathbb{R}^7$  and our *control* to be  $\mathbf{u} = \mathbf{u}_B \in \mathbb{R}^3$ . The dynamics in (17b) and (17c) can then be expressed in general using

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, \mathbf{u}), \quad t \in [t_0, t_f]. \quad (18)$$

Since we are solving for the maneuvers final time, we must map the free-final time problem in Problem 1 to a fixed-final time problem. We do so by defining a *normalized time*  $\tau \in [0, 1]$  and changing our dynamics (18) to be

$$\frac{d\mathbf{x}}{dt} = \frac{d\mathbf{x}}{d\tau} \frac{d\tau}{dt} = f(\mathbf{x}, \mathbf{u}) \quad \Rightarrow \quad \frac{d\mathbf{x}}{d\tau} = \frac{dt}{d\tau} f(\mathbf{x}, \mathbf{u}) \quad (19)$$

where  $s := \frac{dt}{d\tau}$  is called the *time-scaling factor*. By including  $s$  as a solution variable, we can recover the final time for the maneuver using the relation  $t_f = s\tau_f$ . Since  $\tau_f = 1$  by definition, we see that  $s = t_f$ . I use  $s$  to keep the equations looking nicer.

**Linearization** The dynamics are linearized about a reference trajectory so that they are an affine function of the solution variables. Let's assume that we have access to a reference trajectory  $\bar{\mathbf{z}}(\tau) := (\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau), \bar{s})$  for  $\tau \in [0, 1]$ . The first order Taylor series about  $\bar{\mathbf{z}}(\tau)$  is then

$$\frac{d\mathbf{x}}{d\tau} \approx \bar{s} f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + A(\bar{\mathbf{z}}(\tau))(\mathbf{x}(\tau) - \bar{\mathbf{x}}(\tau)) + B(\bar{\mathbf{z}}(\tau))(\mathbf{u}(\tau) - \bar{\mathbf{u}}(\tau)) + S(\bar{\mathbf{z}}(\tau))(s - \bar{s}) \quad (20a)$$

$$= A(\bar{\mathbf{z}})\mathbf{x} + B(\bar{\mathbf{z}})\mathbf{u} + S(\bar{\mathbf{z}})s + Z(\bar{\mathbf{z}}) \quad (20b)$$

where,

$$A(\bar{\mathbf{z}}) := \frac{d}{d\mathbf{x}} s f(\mathbf{x}, \mathbf{u}) = s \begin{bmatrix} A_{11} & A_{12} & 0_{4 \times 3} \\ 0_{3 \times 4} & A_{22} & A_{23} \\ 0_{3 \times 4} & 0_{3 \times 3} & 0_{3 \times 3} \end{bmatrix} \quad (21a)$$

$$B(\bar{\mathbf{z}}) := \frac{d}{d\mathbf{u}} s f(\mathbf{x}, \mathbf{u}) = s \begin{bmatrix} 0_{4 \times 3} \\ -I_3 \\ I_3 \end{bmatrix} \quad (21b)$$

$$S(\bar{\mathbf{z}}) := \frac{d}{ds} s f(\mathbf{x}, \mathbf{u}) = f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \quad (21c)$$

$$Z(\bar{\mathbf{z}}) := -A(\bar{\mathbf{z}})\bar{\mathbf{x}} - B(\bar{\mathbf{z}})\bar{\mathbf{u}} \quad (21d)$$

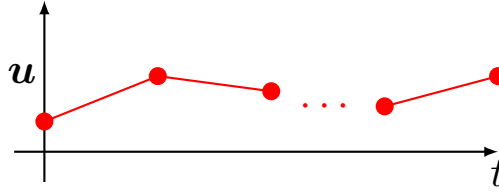
and,

$$A_{11} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}_B^\top \\ \boldsymbol{\omega}_B & \boldsymbol{\omega}_B^\times \end{bmatrix}, \quad A_{12} = \frac{1}{2} \begin{bmatrix} -\mathbf{q}_v^\top \\ \mathbf{q}_v^\times + q_4 I_3 \end{bmatrix} J^{-1}, \quad A_{22} = (\mathbf{h}_B^\times + \mathbf{h}_w^\times) J^{-1} - \boldsymbol{\omega}_B^\times, \quad A_{23} = -\boldsymbol{\omega}_B^\times.$$

**Discretization** I've chosen to parameterize our control with piecewise linear basis functions after some preliminary testing against other methods. To discretize the dynamics (projecting into a finite dimensional space), first discretize time into  $N$  evenly spaced intervals. For now we proceed with  $N = 10$ .

$$t_0 < t_1 < \dots < t_{N-1} < t_f \quad (22)$$

Figure 6: Depiction of the control signal parameterization. The red dots are the discrete control vectors  $\mathbf{u}_k$  for  $k = 1, \dots, N$  proceeding left to right.



and represent the discretized time points using  $k := t_k$ . Now, instead of the *continuous times*  $t \in (t_0, t_f]$ , we have the *discrete times*  $k \in \{t_0, t_1, \dots, t_{N-1}, t_f\}$ .

The control signal is parameterized over each time interval as

$$\mathbf{u}(\tau) = \frac{t_{k+1} - t}{t_{k+1} - t_k} \mathbf{u}_k + \frac{t - t_k}{t_{k+1} - t_k} \mathbf{u}_{k+1}, \quad k = 1, \dots, N-1. \quad (23)$$

This expression linearly interpolates between  $\mathbf{u}_k$  and  $\mathbf{u}_{k+1}$ . The continuous time input can now be represented using a finite number of vectors, namely the set  $\{\mathbf{u}_k\}_{k=1}^N$ . This parameterization is depicted in Figure 6.

I'm skipping a bunch of details, which can all be found in [5, 3], but the discretization method reformulates (20) into the form

$$\mathbf{x}_{k+1} = A_{d,k} \mathbf{x}_k + B_{d,k}^- \mathbf{u}_k + B_{d,k}^+ \mathbf{u}_{k+1} + S_{d,k} s + R_{d,k} \Rightarrow \mathbf{X} = A\mathbf{X} + B\mathbf{U} + Ss + \mathbf{R} \quad (24)$$

where the matrices  $\{A, B, S, R\}$  are computed by the function `foh.m` in the code. The equation to the right of the implication is simply a concatenation using stacked vectors  $\mathbf{X}, \mathbf{U}$  and  $s$ . The constraints in (17e)-(17h) are all “discretized” by simply applying them at the discrete temporal nodes only.

**Virtual Control** To guide the convergence process, we introduce a new term to address the issue of *artificial infeasibility*. This stems from the fact that the linearization may produce an infeasible iteration if there does not exist some admissible control that satisfies (24). We augment the dynamics with a virtual control term that allows us to synthetically retain feasibility according to

$$\mathbf{X} = A\mathbf{X} + B\mathbf{U} + Ss + \mathbf{R} + \mathbf{V} \quad (25)$$

where  $\mathbf{V}$  can be thought of as an unconstrained input that needn't respect the physical dynamics of the problem. We heavily penalize the use of virtual control (so that we only use it when absolutely necessary) by augmenting the cost function (17a) with  $w_\nu \|\mathbf{V}\|_1$ . If  $\mathbf{V} = 0$ , then satisfaction of (25) implies satisfaction of (24).

After these modifications, we have a convex parameter optimization problem that can be solved iteratively on-board the spacecraft.

**Problem 2.** Find the final time  $s$  and the control  $\mathbf{U} \in \mathbb{R}^{3 \times N}$  such that

$$\min_{s, \gamma, \mathbf{U}, \mathbf{X}, \mathbf{V}} \quad \gamma + w_\nu \|\mathbf{V}\|_1 \quad (26a)$$

$$\text{subject to:} \quad \mathbf{X} = A\mathbf{X} + B\mathbf{U} + Ss + \mathbf{R} + \mathbf{V} \quad (26b)$$

$$\|H_h \mathbf{x}_k\|_\infty \leq \omega_{\max}, \quad k = 1, \dots, N \quad (26c)$$

$$\|\mathbf{u}_k\|_\infty \leq u_{\max}, \quad k = 1, \dots, N \quad (26d)$$

$$\mathbf{x}_k^\top H_q^\top \tilde{M}_e H_q \mathbf{x}_k < 2, \quad k = 1, \dots, N \quad (26e)$$

$$\mathbf{x}_k^\top H_q^\top \tilde{M}_i H_q \mathbf{x}_k < 2, \quad k = 1, \dots, N \quad (26f)$$

$$t_{f,\min} \leq s \leq t_{f,\max} \quad (26g)$$

$$\mathbf{U}^\top \mathbf{U} \leq \gamma \quad (26h)$$

$$\mathbf{x}_1 = [\mathbf{q}_{ic}^\top \mathbf{h}_{B,ic}^\top \mathbf{h}_{w,ic}^\top]^\top, \quad H_f \mathbf{x}_N = [\mathbf{q}_f^\top \mathbf{0}_{1 \times 3}^\top]^\top \quad (26i)$$

where  $\mathbf{X} = [\mathbf{x}_1^\top \mathbf{x}_1^\top \dots \mathbf{x}_N^\top]^\top \in \mathbb{R}^{10N}$  and similarly for  $\mathbf{U} \in \mathbb{R}^{3N}$ . Moreover,  $H_q = [I_4 \ 0_{4 \times 6}]$ ,  $H_h = [0_{3 \times 4} \ J^{-1} \ 0_{3 \times 3}]$  and  $H_f = [I_7 \ 0_{7 \times 3}]$ .

The solutions to Problem 2 are used to re-discretize the nonlinear dynamics, producing a new set of matrices  $\{A, B, S, R\}$ , which are then used to re-solve Problem 2.

**Scaling** The solution variables can be scaled for better numerical stability by using the diagonal matrices  $D_x$ ,  $D_u$ ,  $D_s$  that are defined in the `init_SOAR_params.m` file. These are defined so that, for example,  $\mathbf{x}_k = D_x \hat{\mathbf{x}}_k$ , and the solver is asked to solve for  $\hat{\mathbf{x}}_k$  instead. The entries in  $D_x$  can be chosen so that the expected range of  $\hat{\mathbf{x}}_k$  is roughly  $[-1, 1]$ , providing better numerical conditioning. Sufficient performance has been observed without any scaling (i.e., setting all of these matrices to the identity). For the preliminary design, each matrix has been set to the identity matrix. *If no scaling is required to successfully pass performance testing, these operations should be removed from the library as they would be useless algebraic operations that needlessly increase solution time.*

**Standard Form of Constraints** Our combined *solution vector* is

$$\mathbf{z} = [\mathbf{X}^\top \quad \mathbf{U}^\top \quad \gamma \quad s \quad \mathbf{V}^\top \quad \boldsymbol{\eta}_v^\top] \in \mathbb{R}^{n_z}$$

where  $n_z = N(3N_x + N_u) + 2$ . We will first map the cost function to standard form for second-order cone problems. The cost function must be linear, and so we will use the epigraph form to rewrite the cost function (26a) as

$$\mathbf{c}^\top \mathbf{z}, \quad \mathbf{c} = [0_{1 \times 10N} \quad 0_{1 \times 3N} \quad 1 \quad 0 \quad 0_{1 \times 10N} \quad w_\eta \mathbf{1}_{10N}] \quad (27)$$

and add the following linear constraint to Problem 2

$$-\boldsymbol{\eta}_v \leq \mathbf{V} \leq \boldsymbol{\eta}_v. \quad (28)$$

We can then write the equality constraints (26b) and (26i) in the form  $A\mathbf{z} = \mathbf{b}$  using

$$\begin{bmatrix} [I_{10} \quad 0_{10 \times 10(N-1)}] & 0_{10 \times 3N} & 0_{10 \times 1} & 0_{10 \times 1} & 0_{10 \times 10N} & 0_{10 \times 10N} \\ \mathbf{A} - I_{10N} & \mathbf{B} & 0_{10N \times 1} & \mathbf{S} & I_{10N} & 0_{10N \times 10N} \\ [0_{10 \times 10(N-1)} \quad I_{10}] & 0_{10 \times 3N} & 0_{10 \times 1} & 0_{10 \times 1} & 0_{10 \times 10N} & 0_{10 \times 10N} \end{bmatrix} \mathbf{z} = \begin{bmatrix} \mathbf{x}_0 \\ -\mathbf{R} \\ \mathbf{x}_f \end{bmatrix} \quad (29)$$

The linear inequality constraints in (26c), (26d), (26g) and (28) are expressed as

$$\begin{bmatrix} H_w^T \\ -H_w^T \\ H_u^T \\ -H_u^T \\ H_s^T \\ -H_s^T \\ H_v^T - H_{\eta_v}^T \\ -H_v^T - H_{\eta_v}^T \end{bmatrix} \mathbf{z} \leq \begin{bmatrix} w_{\max} \mathbf{1}_{3N} \\ w_{\max} \mathbf{1}_{3N} \\ u_{\max} \mathbf{1}_{3N} \\ u_{\max} \mathbf{1}_{3N} \\ t_{f,\max} \\ -t_{f,\min} \\ 0_{10N \times 1} \\ 0_{10N \times 1} \end{bmatrix} \iff G_{lin} \mathbf{z} \leq \mathbf{h}_{lin}, \quad (30)$$

where,

$$H_u = \begin{bmatrix} 0_{10N \times 3N} \\ I_{3N} \\ 0_{(20N+2) \times 3N} \end{bmatrix}, \quad H_s = \begin{bmatrix} 0_{13N \times 1} \\ 0 \\ 1 \\ 0_{20N \times 1} \end{bmatrix}, \quad H_\gamma = \begin{bmatrix} 0_{13N \times 1} \\ 1 \\ 0 \\ 0_{20N \times 1} \end{bmatrix}, \quad H_v = \begin{bmatrix} 0_{(13N+2) \times 10N} \\ I_{10N} \\ 0_{10N \times 10N} \end{bmatrix}$$

$$H_{\eta_v} = \begin{bmatrix} 0_{(23N+2) \times 10N} \\ I_{10N} \end{bmatrix}, \quad \bar{H}_w = I_N \otimes [0_{3 \times 4} \quad J^{-1} \quad 0_{3 \times 3}], \quad H_w = \begin{bmatrix} \bar{H}_w \\ 0_{(23N+2) \times 3N} \end{bmatrix}$$

If there are no attitude exclusion/inclusion zone constraints, then the second-order cone constraint(s) are written as a function of  $\mathbf{z}$  using

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} H_\gamma^\top \\ -H_u^\top \\ \frac{1}{\sqrt{2}} H_\gamma^\top \end{bmatrix} \mathbf{z} \leq \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0_{3N \times 1} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \iff G_{quad} \mathbf{z} \leq \mathbf{h}_{quad} \quad (31)$$

This cone has dimension 32.

For each  $k = 1, \dots, N$ , the attitude inclusion/exclusion constraints can be written using

$$P_{i,k} = \tilde{M}_i H_q H_k \quad \text{and} \quad P_{e,k} = \tilde{M}_e H_q H_k$$

$$G_{i,k} = \begin{bmatrix} 0_{1 \times n_z} \\ -P_{i,k} \\ 0_{1 \times n_z} \end{bmatrix} \quad \text{and} \quad G_{e,k} = \begin{bmatrix} 0_{1 \times n_z} \\ -P_{e,k} \\ 0_{1 \times n_z} \end{bmatrix}$$

Table 5: Maximum dimensions for the SOAR payload optimization problem. The function `convert_to_CCS` inside the `SOLVE` function maps a general matrix  $M$  to the compressed column storage matrices  $\{M_{jc}, M_{ir}, M_{pr}\}$  – see [2].

Variable	Size	Type	Value	Description
$n$	1	int32	332	number of primal variables
$m$	1	int32	474	number of generalized inequality constraints, rows of $G$ in (32)
$p$	1	int32	117	number of equality constraints, rows of $A$
$l$	1	int32	322	number of linear inequality constraints, rows of $h_{lin}$
$ncones$	1	int32	21	number of second order cone constraints, rows of $h_{quad}$
$\mathbf{c}$	332	double	see (27)	cost vector
$G_{jc}$	333	int32		generalized inequality constraint matrix (part of CCS format)
$G_{ir}$	834	int32		generalized inequality constraint matrix (part of CCS format)
$G_{pr}$	834	double		generalized inequality constraint matrix (part of CCS format)
$\mathbf{h}$	474	double	see (32)	generalized inequality constraint vector
$q$	21	int32		dimensions of each second order cone. Size $m$
$A_{jc}$	333	int32		equality constraint matrix (part of CCS format)
$A_{ir}$	1278	int32		equality constraint matrix (part of CCS format)
$A_{pr}$	1278	double		equality constraint matrix (part of CCS format)
$\mathbf{b}$	117	double	see (29)	equality constraint vector

$$\mathbf{h}_{i,k} = \begin{bmatrix} \frac{3}{\sqrt{2}} \\ 0_{4 \times 1} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad \mathbf{h}_{e,k} = \mathbf{h}_{i,k}$$

where,

$$H_{x_k} = \begin{bmatrix} 0_{10 \times 10} & \cdots & \underbrace{I_{10}}_{k\text{th block}} & \cdots & 0_{10 \times 10} & 0_{10 \times (23N+2)} \end{bmatrix}$$

Each of these cones have dimension 6. The quantities  $G_{\cdot,k}$  and  $\mathbf{h} = \mathbf{h}_{\cdot,k}$  are appended to the bottom of  $G_{quad}$  and  $\mathbf{h}_{quad}$  as they are computed. The problem sizes are summarized in Table 5.

The inequality constraints are then passed to ECOS using the data

$$G = \begin{bmatrix} G_{lin} \\ G_{quad} \end{bmatrix}, \quad \mathbf{h} \preceq_{\mathcal{K}} \begin{bmatrix} \mathbf{h}_{lin} \\ \mathbf{h}_{quad} \end{bmatrix} \quad \Rightarrow \quad G\mathbf{z} \leq \mathbf{h}, \quad \text{and} \quad q = \begin{bmatrix} 32 \\ 6 \mathbf{1}_{2N} \end{bmatrix}. \quad (32)$$

Note that  $q$  is the ECOS input that provides the second order cone dimensions, and has length  $ncones = 2N + 1$ . The standard form of Problem 2 is then simply

**Problem 3.** Find the vector  $\mathbf{z} \in \mathbb{R}^{n_z}$  such that

$$\min_{\mathbf{z}} \quad \mathbf{c}^T \mathbf{z} \quad (27)$$

$$\text{subject to:} \quad A\mathbf{z} = \mathbf{b} \quad (29)$$

$$G\mathbf{z} \preceq_{\mathcal{K}} \mathbf{h} \quad (32)$$

where the generalized inequality is either a linear cone as in (12) or a second order cone as in (13). Which one applies to which entries of  $G$  and  $\mathbf{h}$  is controlled by the value of  $l$  and  $ncones$ . Always the first  $l$  rows of  $G$  are linear cones, and the next  $sum(q)$  rows are second order cones of dimension  $q(k)$ .

## References

- [1] Unsik Lee and Mehran Mesbahi. Feedback Control for Spacecraft Reorientation Under Attitude Constraints via Convex Potentials. *Transactions on Aerospace and Electronic Systems*, 50(4):2578–2592, 2014.
- [2] Danylo Malyuta. *An Optimal Endurance Power Limiter for an Electric Race Car Developed for the AMZ Racing Team Semester Project*. M.S., Swiss Federal Institute of Technology (ETH) Zurich, 2016.
- [3] Taylor P. Reynolds, Michael Szmuk, Danylo Malyuta, Mehran Mesbahi, Behçet Açıkmeşe, and John M. Carson III. Dual Quaternion Based 6-DoF Powered Descent Guidance with State-Triggered Constraints. *arXiv e-prints*, 2019. arXiv:1904.09248.



- [4] Marcel J. Sidi. *Spacecraft Dynamics and Control*. Cambridge University Press, 1997.
- [5] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. Successive Convexification for Real-Time 6-DoF Powered Descent Guidance with State-Triggered Constraints. *arXiv e-prints*, 2018. arXiv:1811.10803.