AAS 21-565

DIRECTLY LEVERAGING THE STATE TRANSITION MATRIX IN STK ASTROGATOR

Nathaniel Kinzly[*], Sai Chikine[†], John Carrico[‡], Mike Loucks[§], and Cody Short[¶]

Trajectory design is, as the name suggests, a design problem at heart. Trajectory designers have a wide variety of tools at their disposal in order to design the best trajectory in the context of a mission. These tools include analytical frameworks such as the Circular Restricted 3-body Problem (CR3BP) that can yield a first-guess for a trajectory, objectives such as fuel usage that can be used to compare solutions, and numerical optimizers that can take a designer from a first guess to an optimal trajectory. The state transition matrix (STM) represents an analytical tool that allows trajectory designers to perform classical dynamical system stability analyses for an orbit. These analyses have applications in rendezvous and proximity operations (RPO) missions, missions in cislunar space, and a myriad of other astrodynamics contexts. This paper will highlight new STM-related capabilities and research areas within the Systems Tool Kit (STK) Astrogator capability set from AGI, an Ansys Company and give a motivating example for how these new capabilities can be useful for trajectory designers.

INTRODUCTION

The STM is not a novel concept in Astrogator.[1] It has existed for use in the tool for several years via user plugins. In 2018, the STM was made more directly available, and this addition was the first step to adding more support for the STM to be used natively in Astrogator. The recent additions for the STM, released in STK 12.2, are focused in two main areas: adding STM eigenvalues and eigenvectors and updating STM interpolation to use Hermite interpolation. Both of these additions present unique challenges to integrate into the STK architecture. There has also been research done into propagating the STM through both impulsive and finite maneuvers, and the propagation through finite maneuvers is a planned addition for STK 12.4. These new features all allow trajectory designers to do more with the STM and to perform analyses with the STM without writing their own code.

This paper presents the new STM-related additions and research areas within STK. First, a brief introduction to the STM is presented followed by a discussion of the implementation details of the STM eigenvalues and eigenvectors, including an analysis of the post-processing done on these values to ensure they are presentable in STK reports. Afterward, a comparison between Lagrange and Hermite interpolation for the STM is presented. Then, research pertaining to propagation of

[*]Astrodynamics Software Engineer, AGI, an Ansys Company, 220 Valley Creek Blvd., Exton, PA 19341
[†]Aerospace Engineer/Astrodynamicist, Advanced Space, 1400 W 122nd Ave, Westminster, CO 80234
[‡]CTO & Principle Astrodynamics Specialist, Space Exploration Engineering, 324 Main Street #173, Laurel, MD 20725
[§]CEO & Chief Astrodynamics Specialist, Space Exploration Engineering, 687 Chinook Way, Friday Harbor, WA 98250
[¶]STK Astrogator Product Owner and Technical Lead, AGI, an Ansys Company, 220 Valley Creek Blvd., Exton, PA 19341

the STM through finite and impulsive maneuvers is discussed. Next, the accuracy of the propagated STM is investigated by using the STM to approximate the propagation of states near a known propagation. Finally, an example of using the STM for trajectory design is described and discussed.

THE STATE TRANSITION MATRIX

Many analysis and design processes require information not only of a particular solution but also of the behavior about that solution in the design space. Frequently, the question of what happens if reality is slightly perturbed from the original design reference must be considered. Such perturbations may arise, for example, as a consequence of insufficiently modeled forces. High-fidelity models for spacecraft motion can never perfectly account for all forces acting on the vehicle—these models can yield extremely accurate and precise results, but ultimately, they remain *models*. It is often extremely valuable to have some mechanism for assessing and characterizing the behavior adjacent to the original solution. One option for such assessment is an examination of the linear flow associated with a given reference solution to the underlying model using the state transition matrix.

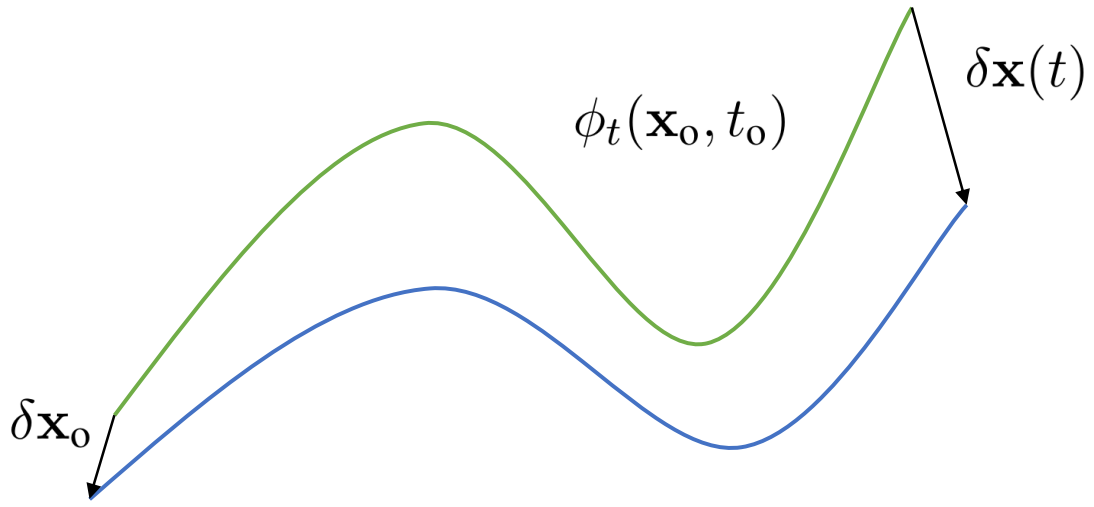


Figure 1. Nominal (green) and perturbed (blue) trajectories with final and initial state variations.

The STM captures the effects of an initial state variation $\delta\mathbf{x}_0$ along a solution $\phi_t(\mathbf{x}_0, t_0)$ resulting in a final state variation $\delta\mathbf{x}(t)$ after a duration t. This concept is captured in Figure 1 where the nominal path is drawn in green above a perturbed path in blue along with vectors indicating the initial and final variations along the path. The STM can be produced by numerical integration of the variational equations corresponding to the underlying model, frequently posed as a system of first-order differential equations.

$$\dot{\mathbf{x}} = f(\mathbf{x}, t), \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1}$$

The solution $\phi_t(\mathbf{x}_0, t_0)$ necessarily satisfies the differential equations such that

$$\dot{\phi}_t = f(\phi_t(\mathbf{x}_0, t_0), t), \qquad \phi_{t_0}(\mathbf{x}_0, t_0) = \mathbf{x}_0 \tag{2}$$

It follows from Equation (2) that the derivative of the solution $\phi_t(\mathbf{x}_0, t_0)$ with respect to the initial state, denoted $\Phi_t(x_0, t_0)$ will result in the variational equations[2]

$$\dot{\Phi}_t(\mathbf{x}_0, t_0) = A\Phi_t(\mathbf{x}_0, t_0), \quad \Phi_{t_0}(\mathbf{x}_0, t_0) = I, \quad A = \frac{\partial f(\phi_t(\mathbf{x}_0, t_0), t)}{\partial x} \tag{3}$$

Where $\Phi_t(\mathbf{x}_0, t_0)$ is the STM, I is the identity matrix, and A is the Jacobian of the equations of motion. Referring back to Figure 1, the associated depiction can be captured notationally in Equation (4)

$$\delta\mathbf{x}(t) = \Phi_t(\mathbf{x}_0, t_0)\delta\mathbf{x}_0 \tag{4}$$

In the context of astrodynamics, the STM is typically a 6x6 matrix to correspond with the 6 element position and velocity state. Additionally, there are examples of using the STM in astrodynamics areas such as RPO,[3] in cislunar space,[4,5] and others.[6]

IMPLEMENTATION OF STM EIGENVALUES AND EIGENVECTORS

As part of STK 12.2, the STM eigenvalues and eigenvectors were made available using new data providers and as calculation objects within Astrogator. This section presents some of the challenges and design choices that came with implementing these values sensibly in the STK architecture.

The STM Eigenvalue Problem

The eigenvalue problem for the STM is given in Equation 5 where λ_i is one of the 6 eigenvalues and $\mathbf{v}_i$ is its corresponding eigenvector.

$$\Phi_t(\mathbf{x}_0, t_0)\mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{5}$$

In the context of the STM, the eigenstructure determines which initial state perturbations lead to the largest and smallest changes on the final state. In STK, the eigenvalues and eigenvectors are found using the Eigen[7] library.

Design Goals

The main priority for this new addition is presenting useful information using the built-in reporting mechanisms. Since eigenvalue algorithms do not guarantee any ordering of their output, the eigenvalues require special treatment so that they can be presented naturally in reports. Additionally, the eigenvalue computation should be unintrusive; users who do not care about the STM eigenvalues or eigenvectors should not experience the computational overhead required to compute these quantities. These two design goals led to the decision to support the STM eigenvalues and

eigenvectors as quantities primarily processed after propagation. The eigendecomposition of the STM is only performed once either the eigenvalues or eigenvectors are requested. At that time, the eigenvalues and eigenvectors are sorted for consistency and saved in case they are requested again. This workflow also avoids recomputing the eigendecomposition if multiple eigenvalues or eigenvectors are requested at once.

Eigenvalue Sorting

In order to sort the eigenvalues so that each eigenvalue is self consistent when reported on, several options were explored. The sorting problem can be formulated as a matching problem. Let an eigenvalue slot be a number from one to six. Then, the matching problem is one where at each time step, each eigenvalue needs to be matched with its most appropriate eigenvalue slot such that the values described by an eigenvalue slot are approximately smooth and continuous in time. A visual representation of a single step of eigenvalue sorting is given in Figure 2

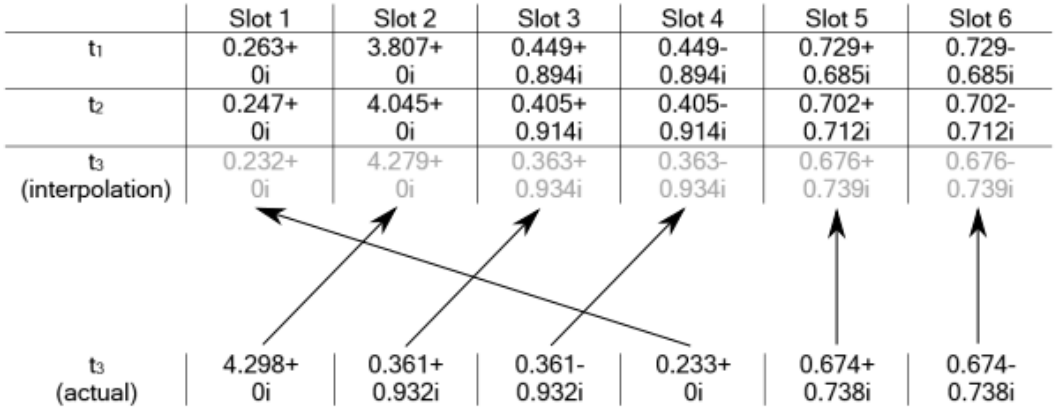


Figure 2. Depiction of a Step in Eigenvalue Sorting

Sorting Approaches The first and simplest option that was explored was one using linear interpolation, denoted the *Linear Interpolation Sorting* method. For each eigenvalue slot, a linear interpolation is performed using the previous two time points to find the value that is expected to be in the eigenvalue slot for the time point of interest. Then, the algorithm greedily chooses the eigenvalue that has a closest match to the first interpolated value and assigns that eigenvalue the first slot. Then, the algorithm looks among the remaining eigenvalues for the eigenvalue closest to the second interpolated value and assigns that eigenvalue the second slot. This process is repeated until all eigenvalues at the time step are assigned a slot. This algorithm was also tested using higher order interpolation methods, specifically, the *Second Order Lagrange Sorting* method, which uses a second order Lagrange interpolation instead of a linear interpolation.

The second option is a more complex variation of the first. This method, denoted as the *Minimum Difference Assignment* method, uses linear interpolation to find a guess for each slot like the Linear Interpolation Sorting method. However, when the algorithm needs to assign each eigenvalue to a slot, it chooses the set of assignments that minimizes the sum of the difference between the interpolated values for each slot and the assigned eigenvalue to each slot. For each time step, this method calculates the differences between each slot guess and each eigenvalue and then performs a brute force search over all possible assignments to find the one that minimizes the difference. Since this

approach searches over 720 different possible assignments, this method is significantly slower than the prior method.

The third option, denoted the *Stable Matching Assignment* method, formulates the matching problem as a stable matching problem.[8] In this formulation, elements of each set (in this case the set of eigenvalue slots and the set of eigenvalues) rank each of the elements of the opposing set according to their preferences. In this case, the preferences are chosen based on the difference between the interpolated eigenvalue for each slot and the true eigenvalues, with each slot preferring closer eigenvalues and each eigenvalue preferring closer interpolated values for that slot. The goal is to find a set of stable matches among the two sets. An unstable match between elements A and B is one where there exists another matching between C and D where A is higher on D's preference list than C, and D is higher on A's preference list than B. In other words, a matching is unstable if there are two elements (A and D) that would prefer to be matched with each other rather than their current matches. This problem can be solved using the Gale-Shapley algorithm.[8] The algorithm works by greedily matching based on one of the two set's preference lists, and it only breaks up previous matches if an unstable match is created. Therefore, the eigenvalue sorting is completed by finding the interpolated values for each slot at a time step, creating preference lists for each eigenvalue and slot, and running the Gale-Shapley algorithm to give the matching for that time step.

Approach Evaluation Evaluating these sorting techniques is difficult for one main reason: the actual eigenvalues of the STM are not smooth curves. These interpolation-based approaches rely on the first derivative (or higher order derivatives) to change slowly. It becomes harder to predict which eigenvalue should correspond to each slot when there are jumps in the eigenvalues, which can happen when the STM eigenvalue set passes through a bifurcation or collision. These events occur when an STM has two complex-conjugate eigenvalues at one time step when it did not at the previous time step or vice-versa. These points in time pose challenges for the sorting algorithms since the derivative of the eigenvalues with respect to time changes significantly at time steps where the STM gains or loses a pair of complex-conjugate eigenvalues.

Therefore, the evaluation of these sorting algorithms is performed qualitatively by inspection of various STM eigenvalue plots and observing how "natural" the eigenvalue correspondence appears. If there are large jumps or obvious switching between two values compared to another method, then that method is not preferred for that particular propagation.

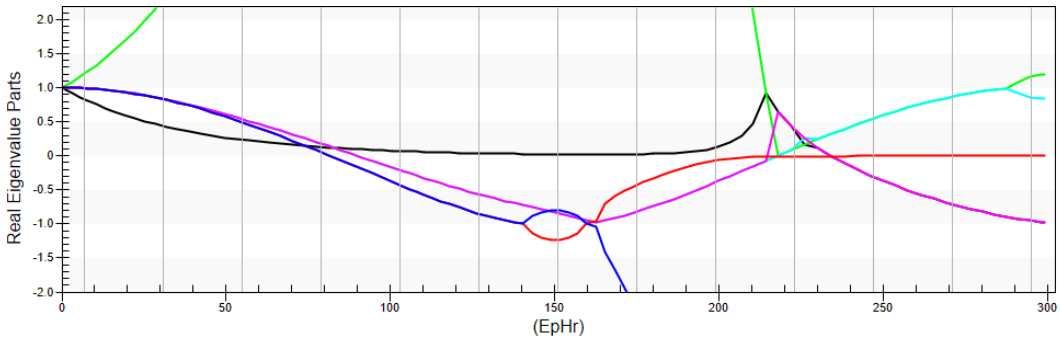


Figure 3. Real Parts of STM Eigenvalues Sorted by Linear Interpolation Method

Plots of sorted, real eigenvalues using each of the three implemented methods, Linear Interpolation Sorting, Second Order Lagrange Sorting, and Minimum Difference Assignment, are given in Figures 3, 4, and 5 respectively. The STM that produces these eigenvalues was propagated along a

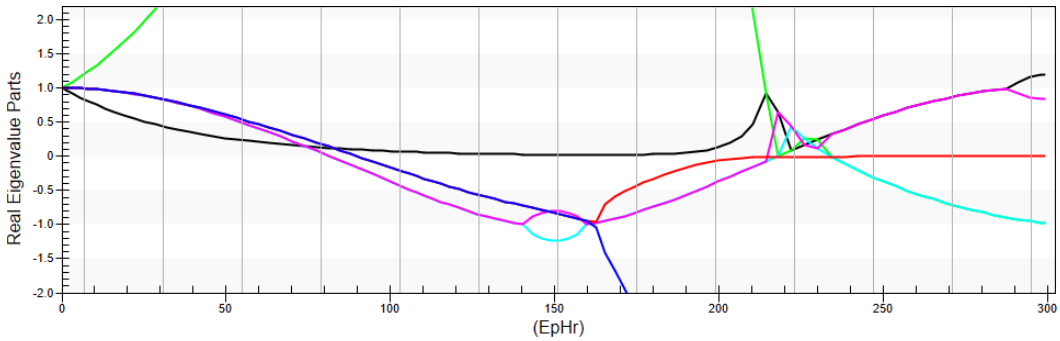


Figure 4. Real Parts of STM Eigenvalues Sorted by Second Order Lagrange Method

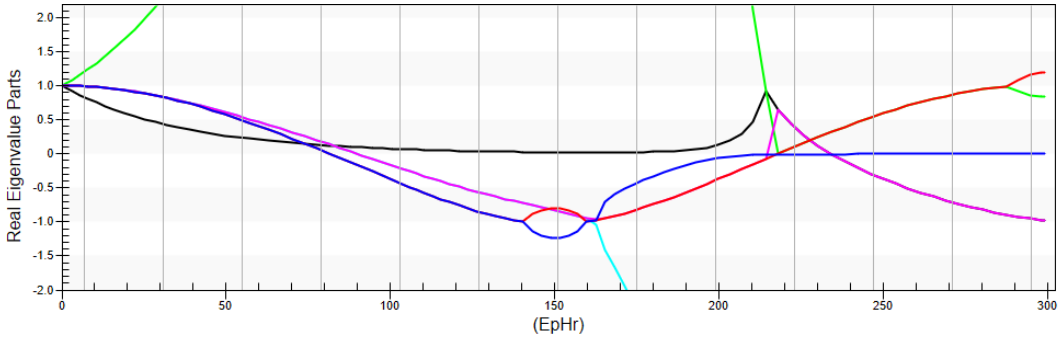


Figure 5. Real Parts of STM Eigenvalues Sorted by Minimum Difference Assignment Method

Lyapunov orbit using circular restricted 3 body problem (CR3BP) dynamics, and the orbit is shown in Figure 6. In each of the methods, between around 210 and 230 epoch hours, the eigenvalues jump unnaturally, suggesting that each of the sorting algorithms fails at these times. However, the Minimum Difference Assignment seemed to have no further swapping after the initial swap, while the other two methods have further swapping after the initial error. Additionally, the Second Order Lagrange Sorting has more jumps and therefore a worse sorting performance than the Linear Interpolation Sorting for this particular case.

The trajectory that was used in Figures 3 through 5 was propagated using a variable time step RKF78 integrator.[9] Figure 7 represents the same Linear Interpolation Sorting method depicted in Figure 3, but it is instead using the STM eigenvalues of a trajectory propagated with a fixed integration step of 1 minute. In this case, there are no obvious jumps between 210 and 230 epoch hours, suggesting that the sorting was successful in this case. This specific point gave the sorting algorithms trouble when using a variable time step due to the creation of a complex-conjugate pair of eigenvalues at this time. The discontinuity in the derivatives of the green and black eigenvalues made the interpolation prediction for each eigenvalue slot less accurate. By decreasing the time between ephemeris points, the interpolation within each eigenvalue slot was more likely to correctly predict the subsequent value for that slot, which resulted in better sorting. Due to this dependence on time step, the Stable Matching Assignment approach is not used. Since the matching preferences are reliant on the behavior of the previous time steps, they become unreliable when the integration time steps are too far apart. This lack of reliability clashes with the rigor of the stable matching approach.

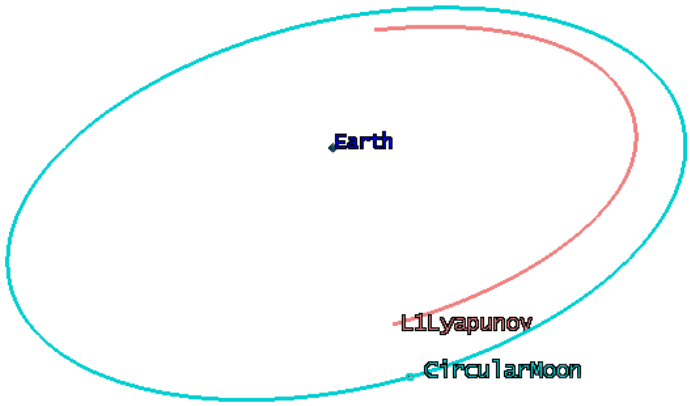


Figure 6. Inertial View of the Orbit Associated with Figures 3 - 5 and Figure 7

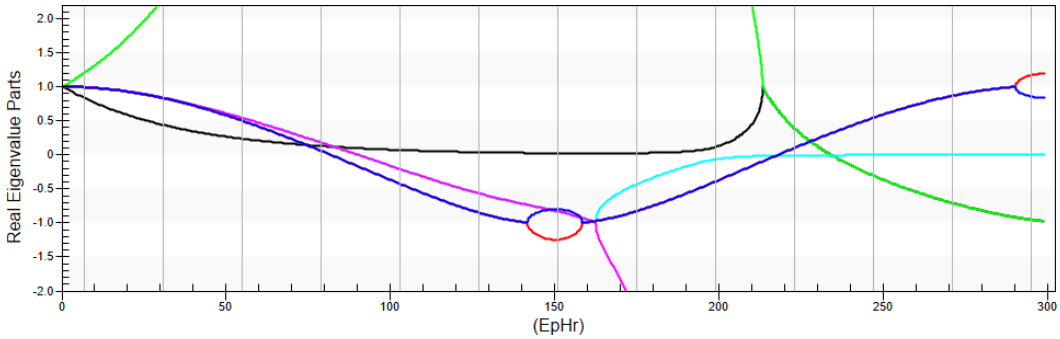


Figure 7. The real parts of the STM Eigenvalues sorted by the linear interpolation method. Unlike in Figure 3, the integration step size for this plot is fixed to one minute, which gives the interpolation more data points. This extra data lets it avoid jumping around the 215 epoch hours mark.

Table 1 gives the execution times of five different runs of each sorting algorithm as well as the average execution time of those five runs. These algorithms were timed on an Intel Xeon ES-2620 v3 @ 2.40 GHz CPU running Windows 10. The Linear Interpolation Sorting approach has the shortest execution time due to its simplicity while the Minimum Difference Assignment method has the longest due to the number of different comparisons during each time step. Since each sorting method uses a bounded number of previous steps to process the current step, each of these methods' run-times scale linearly with the number of integration points. As a result, examining the run-times with a different number of integration points leads to the same relative conclusions. Due to the fact that each sorting method has similar sorting performance when integration steps are large, the simplicity and speed of the linear interpolation sorting was chosen as the implemented sorting method. In the case when there are obvious sorting errors for a given trajectory, reducing the maximum variable time step size or changing the integration to a fixed time step should reduce or eliminate the sorting errors.

Table 1. Execution Times for Sorting Methods

Run Number	Run Time (microseconds)		
	Linear Interpolation	Second Order Lagrange	Minimum Difference
1	241	484	1677
2	336	404	1668
3	324	402	1677
4	333	306	1282
5	242	400	1299
Average	295.2	399.2	1520.6

Eigenvector Sorting

As with the eigenvalues, the STM eigenvectors are also sorted so that they can be presented consistently when reported. Since the eigenvectors are paired with eigenvalues, the eigenvectors are sorted to maintain these pairs as the eigenvalues are sorted. However, there is an additional component to the eigenvector post-processing. The signs of eigenvectors output from the eigendecomposition algorithm are arbitrary. Reporting on these eigenvector components without post-processing the eigenvector signs can show switching between positive and negative values at different time steps. To account for this, the signs of the eigenvectors are reconciled such that for the first three time steps, the x element of each eigenvector is set to be positive. After that, a linear interpolation is done using the previous two time steps to find an expected value for the x element of each eigenvector. If the difference between the interpolated value and the actual value is greater than the interpolated value, then the sign on the eigenvector for that time step is flipped.

INTERPOLATION

Interpolation of numerical results is a convenient way to obtain estimates at times between integration nodes without needing to re-propagate a system. As discussed in the previous section, the trends in eigenvalues can be fairly erratic, which can lead to large interpolation errors for the eigenvalues. Additionally, any errors in sorting can introduce more interpolation error. For these reasons, the interpolation of the eigenvalues is accomplished by first interpolating the STM and then finding the eigendecomposition from that interpolated result.

Prior to STK 12.2, Astrogator used third order Lagrange interpolation for the STM to be consistent with other interpolated quantities*. Since this interpolation was also going to be used for the newly introduced STM eigenvalues, an investigation was made into the performance of Lagrange interpolation for the STM compared to other interpolation methods.

Interpolation Methods

The two interpolation methods evaluated here are both forms of polynomial interpolation. The interpolation algorithm is given function values y_i at $n+1$ different input times t_i and tasked with finding a polynomial of degree n, $p(t)$, such that $p(t_i) = y_i$ for $i = 0$ to n. For Hermite interpolation, the algorithm is also given at least the first derivative of the function at each input time. The polynomial used for Lagrange interpolation is given in Equation (6) where p is given by a sum of Lagrange polynomials ℓ_i.[10]

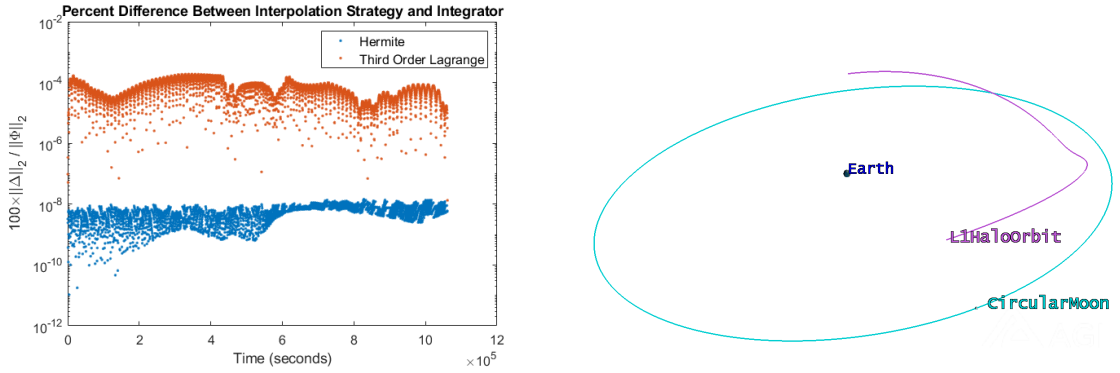
*Position and velocity are interpolated with 7th order Lagrange interpolation. Other state elements are interpolated with a 3rd order Lagrange interpolation. The STM is interpolated with a 3rd order Hermite Interpolation

$$p(t) = \sum_{i=0}^{n} y_i \ell_i(t) \quad \text{where} \quad \ell_i(t) = \prod_{0 \le j \le n} \frac{t - t_j}{t_i - t_j} \tag{6}$$

On the other hand, Hermite interpolation finds the interpolation polynomial $p(t)$ using generalized divided differences of similar form to Newton interpolation.[10]

Results

The two interpolation methods are tested by comparing an interpolated STM to to an integrated STM for the trajectory given in Figure 8(b). This trajectory is a halo orbit integrated in the circular restricted three-body problem.[11] For both methods, each element of the STM, Φ, was interpolated separately. Additionally, the derivatives of the STM, $\dot{\Phi}$ that are used in Hermite interpolation are found using the STM variational equations given in Equation (3). The interpolation result is found by interpolating a variable time-step integrator output to six minute intervals, and the integration result is found by using a fixed time-step of one minute.



(a) Interpolation Error

(b) Inertial View of the Orbit from **(a)**

Figure 8. **(a)** The matrix 2-norm of the difference, Δ, between the interpolation results and integration results normalized by the matrix 2-norm of the integrated STM, Φ, reported as a percent. **(b)** The inertial view of the halo orbit corresponding to the STM shown in **(a)**.

A plot of the interpolation errors for both Lagrange and Hermite interpolation of the STM is given in Figure 8(a). As shown, the Hermite interpolation gives results closer to those from the numerical integration compared to the Lagrange interpolation at almost all time steps. Specifically, the maximum interpolation error for the Hermite interpolation is approximately two orders of magnitude better than the Lagrange interpolation. From these results, Hermite interpolation was chosen to be used for STM interpolation in Astrogator.

Propagation Function Changes

When propagating a trajectory in STK, the propagator has several "Propagation Functions" that each contribute to the overall acceleration of the spacecraft.[12] These functions allow the user to customize the fidelity of the propagation by including things such as gravitational forces from other bodies or contributions from solar radiation pressure or drag. Each of these propagation functions contains intermediate quantities used in calculations for the spacecraft's acceleration. Within the

state transition matrix propagation function, the Jacobian of each of the other propagation functions is calculated and used to propagate the STM. In order to make use of the Jacobian for interpolation, a new mechanism was required within the system architecture to store these values during integration and recover them later for reporting purposes. As a result of this new mechanism, it is now easier to extract similar quantities from propagation functions in the future if there is a demonstrated interest from users, which has applicability beyond the STM.

STM PROPAGATION THROUGH MANEUVERS

As useful as it is to have a tool that allows one to easily see how perturbations affect ballistic trajectories, it may be even more so when considering maneuvers; burns must eventually be considered when designing virtually every real-world mission. Maneuvers can be modeled one of two ways: the **impulsive** method, wherein a maneuver is modeled as a discontinuous ΔV, and the **finite** method, where the maneuver is actually modeled over some finite time period using some kind of thruster force model, which can vary in fidelity. Thankfully, the STM can be propagated through maneuvers, both impulsively modeled burns and finite ones.

STM Propagation Through Impulsive Maneuvers

Impulsive maneuvers are a very popular method of modeling maneuvers for the initial stages of trajectory design. They are simple to conceptualize, and allow for easy quantification of requirements for a mission ("how much ΔV is this mission?"). They can also be mathematically simpler since one does not need to consider thruster dynamics. However, since impulsively modeled maneuvers represent something fundamentally nonphysical - a discontinuous trajectory - they preclude the use of the hitherto discussed method to obtain the STM via a simple numerical integration of Equation (3). Instead, properly computing the STM when a trajectory contains an impulse discontinuity requires splitting the solution of the ODE at the impulse time, and considering each part separately.

Transition Matrix Review Related to the STM dynamics defined in Equation (3), other sensitivity/transition matrices can be derived which inform how an ODE solution is affected by changes in system parameters $\mathbf{w}$ and changes in the initial time t_0. Consider the dynamical system $\dot{\mathbf{x}} = f(\mathbf{x}, t, \mathbf{w})$, with a solution $\mathbf{x}(t) = \mathbf{X}_t(\mathbf{x}_0, t_0, \mathbf{w})$. Then, we can examine the state transition matrix Φ, parameter transition matrix P and the initial time transition matrix Θ:

$$\Phi \triangleq \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0} \tag{7}$$

$$P \triangleq \frac{\partial \mathbf{x}(t)}{\partial \mathbf{w}} \tag{8}$$

$$\Theta \triangleq \frac{\partial \mathbf{x}(t)}{\partial t_0} \tag{9}$$

The associated dynamics can then be found by differentiating $\dot{\mathbf{x}} = f(\mathbf{x}, t, \mathbf{w})$ with respect to the parameters and initial time, respectively, and switching the order of differentiation since we assume continuity of second derivatives:

$$\dot{\Phi}^f = A^f \Phi^f, \qquad \Phi^f_{t_0} = \mathbf{I} \tag{10}$$

$$\dot{P}^f = A^f P^f + \frac{\partial f}{\partial \mathbf{w}}, \qquad P^f_{t_0} = \mathbf{0} \tag{11}$$

$$\dot{\Theta}^f = A^f \Theta^f, \qquad \Theta^f_{t_0} = -f(\mathbf{x}_0, t_0, \mathbf{w}) \tag{12}$$

where A^f is equal to A as in Equation (3). Note that P is the same as the sensitivity matrix used in consider covariance analysis in orbit determination, typically denoted θ.[13]

System with Discontinuity Now, consider a second dynamical system $\dot{\mathbf{y}} = g(\mathbf{y}, t, \mathbf{w})$, whose initial time is defined as $t = \tau$, so that $\mathbf{y}_0 = \mathbf{y}(t = \tau)$. Furthermore, the initial condition $\mathbf{y}_0$ represents a discontinuous jump (the impulse) from the system $\dot{\mathbf{x}} = f(\mathbf{x}, t, \mathbf{w})$ as follows:

$$\mathbf{y}_0 = \mathbf{x}(\tau) + \delta(\mathbf{x}(\tau), \tau, \mathbf{w}) \tag{13}$$

Note that the impulse function δ is a differentiable function of τ, the impulse time, the state $\mathbf{x}$ at τ, and the parameter vector $\mathbf{w}$. Similar to Equations (10)-(12), the transition matrices for this system are:

$$\dot{\Phi}^g = A^g \Phi^g, \qquad \Phi^g_{t_0} = \mathbf{I} \tag{14}$$

$$\dot{P}^g = A^g P^g + \frac{\partial g}{\partial \mathbf{w}}, \qquad P^g_{t_0} = \mathbf{0} \tag{15}$$

$$\dot{\Theta}^g = A^g \Theta^g, \qquad \Theta^g_{t_0} = -g(\mathbf{x}_0, \tau, \mathbf{w}) \tag{16}$$

Here, $A^g \triangleq \frac{\partial g}{\partial \mathbf{y}}$.

Derivative of the State with Respect to the Impulse Time As a first course of analysis, the derivative of the solution $\mathbf{y}(t)$ with respect to the discontinuity time τ will be derived here. First, note that the solution to Equation (15) is $\Theta^g = -\Phi g$. Then:

$$\begin{aligned} \frac{d\mathbf{y}(t)}{d\tau} &= \frac{\partial \mathbf{y}(t)}{\partial \tau} + \frac{\partial \mathbf{y}(t)}{\partial \mathbf{y}_0}\frac{d\mathbf{y}_0}{d\mathbf{x}_0} \\ \implies \frac{d\mathbf{y}(t)}{d\tau} &= \Phi^f_t \left(\frac{d\mathbf{y}_0}{d\mathbf{x}_0} - g(\mathbf{y}_0, \tau, \mathbf{w}) \right) \end{aligned} \tag{17}$$

Furthermore, applying the chain rule to the quantity $\frac{d\mathbf{y}_0}{d\mathbf{x}_0}$, and noting that $\mathbf{y}_0 = \mathbf{x}(\tau) + \delta(\mathbf{x}(\tau), \tau, \mathbf{w})$, we obtain:

$$\frac{d\mathbf{y}_0}{d\mathbf{x}_0} = \frac{\partial \mathbf{x}(\tau)}{\partial \tau} + \frac{d\delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{d\tau} \tag{18}$$

Applying the chain rule once again:

$$\begin{aligned} \frac{d\delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{d\tau} &= \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t} + \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}}\frac{\partial \mathbf{x}(\tau)}{\partial t} \\ &= \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t} + \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}} f(\mathbf{x}(\tau), \tau, \mathbf{w}) \end{aligned} \tag{19}$$

Therefore, we find that

$$\frac{d\mathbf{y}(t)}{d\tau} = \Phi^f_t \Big(f(\mathbf{x}(\tau), \tau, \mathbf{w}) - g(\mathbf{y}_0, \tau, \mathbf{w}) + \frac{d\delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{d\tau} \Big) \tag{20}$$

Derivative of the Final State with Respect to the Initial State With a necessary intermediate step complete, we can move on to deriving an expression for the quantity of interest: the state transition matrix relating the state of the second system - that is, after our impulse - to the initial state $\mathbf{x}_0$ of the first system. We denote this quantity:

$$\Phi^{gf}_t \triangleq \frac{d\mathbf{y}(t)}{d\mathbf{x}_0}, \quad t > \tau \tag{21}$$

We can begin by applying the chain rule to compute Φ^{gf}:

$$\begin{aligned}\frac{d\mathbf{y}(t)}{d\tau} &= \frac{\partial \mathbf{y}(t)}{\partial \tau}\frac{\partial \tau}{\partial \mathbf{x}_0} + \frac{\partial \mathbf{y}(t)}{\partial \mathbf{y}_0}\frac{d\mathbf{y}_0}{d\mathbf{x}_0} \\ &= \Phi_t^g\left(\frac{d\mathbf{y}_0}{d\mathbf{x}_0} - g(\mathbf{y}_0,\tau,\mathbf{w})\frac{\partial \tau}{\partial \mathbf{x}_0}\right)\end{aligned} \tag{22}$$

and

$$\frac{d\mathbf{y}_0}{d\mathbf{x}_0} = \frac{d\mathbf{x}(\tau)}{d\mathbf{x}_0} + \frac{d\delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{d\mathbf{x}_0} \tag{23}$$

Now, examining the first term in Equation (23), we find that:

$$\begin{aligned}\frac{d\mathbf{x}(\tau)}{d\mathbf{x}_0} &= \frac{\partial \mathbf{x}(\tau)}{\partial t}\frac{\partial \tau}{\partial \mathbf{x}_0} + \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}_0}\frac{\partial \mathbf{x}_0}{\partial \mathbf{x}_0} \\ &= f(\mathbf{x}(\tau),\tau,\mathbf{w})\frac{\partial \tau}{\partial \mathbf{x}_0} + \Phi_\tau^f\end{aligned} \tag{24}$$

Note that Equation (24) contains extra terms aside from the usual STM Φ_τ^f - this stems from the fact that the transition time τ may itself be a function of the initial state $\mathbf{x}_0$.

The second term in Equation (23) evaluates to:

$$\frac{d\delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{d\mathbf{x}_0} = \frac{\partial \delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{\partial t}\frac{\partial \tau}{\partial \mathbf{x}_0} + \frac{\partial \delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{\partial \mathbf{x}}\frac{d\mathbf{x}(\tau)}{d\mathbf{x}_0} \tag{25}$$

Finally, after making substitutions and simplifying, we find that the quantity of interest Φ_t^{gf} can be obtained as:

$$\frac{d\mathbf{y}(t)}{d\mathbf{x}_0} = \Phi_t^{gf} = \Phi_t^g\left(\left(\mathbf{I} + \frac{\partial \delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{\partial \mathbf{x}}\right)\Phi_\tau^f + \frac{d\mathbf{y}(t)}{d\tau}\bigg|_{t=\tau}\frac{\partial \tau}{\partial \mathbf{x}_0}\right) \tag{26}$$

Note that $\frac{d\mathbf{y}(t)}{d\tau}\Big|_{t=\tau}$ is obtained via Equation (20).

Derivative of the Final State with Respect to Parameters As with the STM, the parameter sensitivity matrix P^{fg} must take the impulse/discontinuity into account. To derive the proper expression, we begin by using the chain rule as usual:

$$\begin{aligned}\frac{d\mathrm{y}(t)}{d\mathbf{w}} &= \frac{\partial \mathbf{y}(t)}{\partial \tau}\frac{\partial \tau}{\partial \mathbf{w}} + \frac{\partial \mathbf{y}(t)}{\partial \mathbf{y}_0}\frac{d\mathbf{y}_0}{d\mathbf{w}} + \frac{\partial \mathbf{y}(t)}{\partial \mathbf{w}} \\ &= -\Phi_t^g g(\mathbf{y}_0,\tau,\mathbf{w})\frac{\partial \tau}{d\mathbf{w}} + \Phi_t^g\frac{d\mathbf{y}_0}{\mathbf{w}} + P_t^g \\ &= \Phi_t^g\left(-g(\mathbf{y}_0,\tau,\mathbf{w})\frac{\partial \tau}{d\mathbf{w}} + \frac{d\mathbf{y}_0}{d\mathbf{w}}\right) + P_t^g\end{aligned} \tag{27}$$

The derivative of the post-impulse state with respect to the parameters ($\frac{d\mathbf{y}_0}{d\mathbf{w}}$) is:

$$\frac{d\mathbf{y}_0}{d\mathbf{w}} = \frac{d\mathbf{x}(\tau))}{d\mathbf{w}} + \frac{d\delta(\mathbf{x}(\tau),\tau,\mathbf{w})}{d\mathbf{w}} \tag{28}$$

Next, the first term turns out to be:

$$\begin{aligned}\frac{d\mathbf{x}(\tau)}{d\mathbf{w}} &= \frac{\partial \mathbf{x}(\tau)}{\partial t}\frac{\partial \tau}{\partial \mathbf{w}} + \frac{\partial \mathbf{x}(\tau)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial \mathbf{w}} \\ &= f(\mathbf{x}(\tau), \tau, \mathbf{w})\frac{\partial \tau}{\partial \mathbf{w}} + P_\tau^g\end{aligned} \tag{29}$$

And the second term is:

$$\frac{d\delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\mathbf{w}} = \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t}\frac{\partial \tau}{\partial \mathbf{w}} + \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}}\frac{d\mathbf{x}(\tau)}{d\mathbf{w}} + \frac{\partial \delta(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{w}} \tag{30}$$

Impulse Time Determined via Constraint Equation The impulse time τ is frequently determined via some stopping condition instead of being set explicitly. For example, an impulsive burn may be set to be executed at apoapsis. In cases like this, all of the above analysis still applies. Consider the impulse time τ defined by some constraint equation ϕ as follows:

$$\phi(\mathbf{x}(\tau), \tau, \mathbf{w}) = 0 \tag{31}$$

The goal is then to find expressions for $\frac{\partial \tau}{\partial \mathbf{x}_0}$ and $\frac{\partial \tau}{\partial \mathbf{w}}$ to be used when needed in the above equations. To find these, we can differentiate Equation (31) with respect to $\mathbf{x}_0$:

$$\frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t}\frac{\partial \tau}{\partial \mathbf{x}_0} + \frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}}\frac{d\mathbf{x}(\tau)}{d\mathbf{x}_0} = 0 \tag{32}$$

From here, we can solve for the desired quantity, componentwise, as follows:

$$\frac{\partial \tau}{\partial x_{0,k}} = -\frac{\frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}}\Phi^f_{k,\tau}}{\frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t} + \frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}} f(\mathbf{x}(\tau), \tau, \mathbf{w})} \tag{33}$$

where the subscript k denotes the k^{th} column of the STM Φ^f. By differentiating with respect to the parameters $\mathbf{w}$ instead of $\mathbf{x}_0$, we can similarly obtain the derivative of the impulse time with respect to the parameters:

$$\frac{\partial \tau}{\partial w_k} = -\frac{\frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}}P^f_{k,\tau}}{\frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial t} + \frac{\partial \phi(\mathbf{x}(\tau), \tau, \mathbf{w})}{\partial \mathbf{x}} f(\mathbf{x}(\tau), \tau, \mathbf{w})} \tag{34}$$

Impulsive STM Computation Overview For a typical astrodynamics problem, the two systems $\dot{\mathbf{x}} = f(\mathbf{x})$ and $\dot{\mathbf{y}} = g(\mathbf{y})$ describe the same dynamics. The impulse time τ may be defined explicitly or through some constraint as in Equation (31). For example, an impulse burn may be applied at an orbit apsis or when some other state constraint is met. The impulse itself may also be defined through some function of the pre-impulse state; as long as the computation of the impulse is differentiable, the above equations hold. Furthermore, any relevant partial derivatives involving the parameters $\mathbf{w}$ must be computed.

Next, the set of 3 ODEs described by Equations (10)-(12) should be integrated in the typical fashion with any numerical integration scheme. This system should be propagated until the impulse time τ. At this point, any impulse is applied to the state, and another propagation is run from time $t = \tau$ to $t = t_f$, with the initial conditions set as defined in Equations (14)-(16). Once propagated to the desired time, Equation (26) can be used to compute the desired STM.

The computation of the transition matrices Φ_t^{gf}, P_t^{gf}, Θ_t^{gf} was tested on multiple test problems. One test problem was a Duffing oscillator problem with an impulse time defined through a discontinuity time and state dependent constraint equation $\phi(\mathbf{x}(\tau), \tau, \mathbf{w}) = 0$ and impulse determined through a similar discontinuity time and state dependent equation $\delta(\mathbf{x}(\tau), \tau, \mathbf{w})$. Another test problem was a 2-body orbit propagation with a fixed-magnitude ΔV impulse, but an impulse time τ determined through an apse stopping condition $\phi(\mathbf{x}(\tau), \tau, \mathbf{w}) = 0$. In both cases, the transition matrices accurately matched the same quantities computed numerically, and this result for the Duffing oscillator problem is given in Figure 9. Furthermore, the transition matrices produced via either method yielded accurate results when used to predict the effects of small perturbations in initial state, time, impulse time, and parameter vector, on the final state.

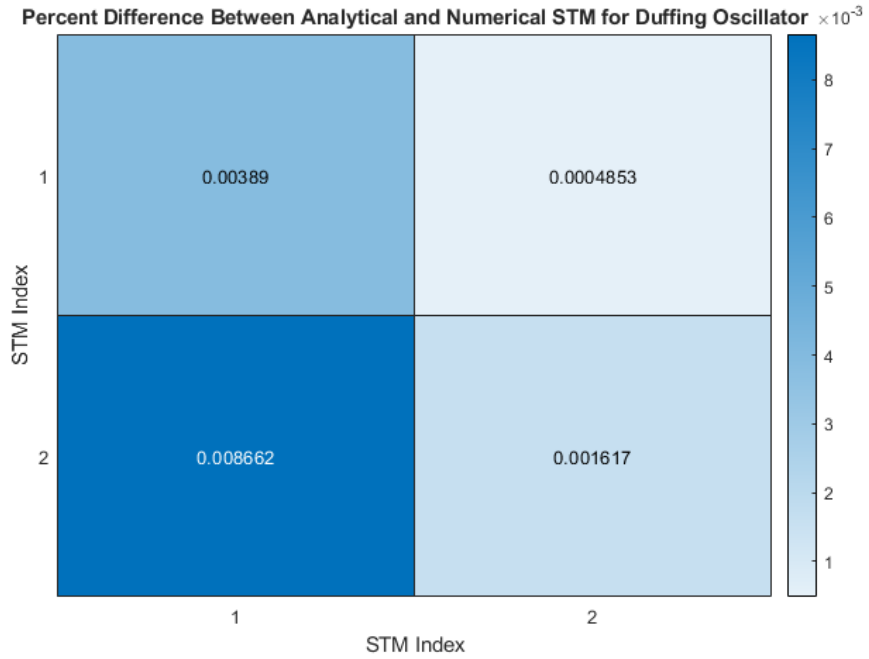


Figure 9. A heat map of the difference between the STM for a Duffing oscillator system computed numerically and analytically.

The use of the impulsive STM Φ^{gf} can be very powerful for initial mission design; designers can continue to use simple impulsive burn models while also being able to use Φ^{gf} and its analogues P^{gf} and Θ^{gf} to quickly and easily analyze the effect of small changes to initial conditions.

STM Propagation Through Finite Maneuvers

Though the finite maneuver model is, in the vast majority of cases, more complex, and more accurately captures the dynamics, it is actually generally easier to mathematically model the evolution of the STM through a finite burn, for the same reason that it is more realistic than an impulsive one: finite burn models are continuous. Therefore, the STM can be propagated through a finite maneuver model relatively simply by augmenting the state with thrust or thruster acceleration components, including the thruster dynamics in the equations of motion, and defining the thrust/acceleration dynamics to be zero (for constant thrust/acceleration), or via the derivative of the thruster/acceleration

model. For example, consider a simple thrust model given by:

$$\vec{a}_T = \begin{bmatrix} \frac{T_{\max}}{m} u_x \\ \frac{T_{\max}}{m} u_y \\ \frac{T_{\max}}{m} u_z \end{bmatrix} \tag{35}$$

where $\vec{u} \triangleq \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}$ is the thrust fraction vector ($\|\vec{u}\| \leq 1$), m is the spacecraft mass, and $T_{\max}$ is the maximum thrust. Mass dynamics are given by

$$\dot{m} = -\frac{T_{\max}\|\vec{u}\|}{V_e} \tag{36}$$

where $V_e = I_{\text{sp}} g_0$ is the exhaust velocity of the propellant. Then, the augmented state can be defined as

$$\mathbf{X} \triangleq \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & m & a_{T,x} & a_{T,y} & a_{T,z} \end{bmatrix}^T \tag{37}$$

Over a thrust arc where the thrust is constant, the derivatives of the thrust terms are zero. The STM can then be computed over this arc in the typical manner, using a 10x10 Jacobian matrix instead of a 6x6 one. Any thrust model can be substituted here in a similar manner, as long as the thruster acceleration dynamics are properly defined.

In practice, due to the wide variety of thrust models supported in STK, the STM, when required in the presence of finite maneuvers, is obtained via numerical methods - namely finite differencing. In the finite difference method of computing derivatives, multiple trajectories are propagated through the dynamics and are differenced to directly approximate the derivative. Recall that the STM $\Phi_t(\mathbf{x}_0, t_0)$ can be defined as:

$$\Phi_t(\mathbf{x}_0, t_0) = \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(t_0)} \tag{38}$$

Hence, this matrix can be approximated via finite differencing column-wise: since each column corresponds to a different state component, each state component can be perturbed in turn to fill in each column.

Many finite differencing schemes exist, starting from simple forward or backward difference formulas, which require 2 points, and increasing in order which require more points. Higher order methods can be more accurate for a given step size (the difference in initial conditions), at the cost of more propagations, which costs computation time. They can also be more stable in the sense that a larger step size can be used without worrying about the error blowing up.

In STK, a 7 point central differencing scheme is used as a tradeoff between computation time and accuracy. For a vector valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, the 7 point central differencing scheme works as follows:

$$\left[\frac{\partial f}{\partial \mathbf{x}}\right]_i \approxeq \frac{f(\mathbf{x}+3h_i) - 9f(\mathbf{x}+2h_i) + 45f(\mathbf{x}+h_i) - 45f(\mathbf{x}-h_i) + 9f(\mathbf{x}-2h_i) - f(\mathbf{x}-3h_i)}{60h_i} \tag{39}$$

where $\left[\frac{\partial f}{\partial \mathbf{x}}\right]_i$ is the i^{th} column of the derivative matrix, and h_i denotes a perturbation in the i^{th} component of the $\mathbf{x}$ vector. Each column requires 6 propagations, so this method requires $6m$ numerical integrations to compute, where m is the dimension of the state vector.

Though computationally intensive, the finite-differencing method of obtaining the STM is very robust, and results in a reliably accurate STM through arbitrarily complex state and/or maneuver dynamics.

LINEAR APPROXIMATION WITH THE STM FOR FINAL STATE ESTIMATION

One of the values of the STM is that it allows for a linear perturbation analysis of a trajectory. In particular, after disturbing the initial state by $\delta\mathbf{x}_0$, the perturbed final state at the end of the propagation, $\mathbf{x}_{pf}$, can be estimated using $\mathbf{x}^*_{pf}$ found using Equation (40), which follows directly from Equation (4)

$$\mathbf{x}^*_{pf} \triangleq \mathbf{x}_f + \Phi_t(\mathbf{x}_0, t_0)\delta\mathbf{x}_0 \tag{40}$$

where $\mathbf{x}_f$ is the final state found by propagating the unperturbed initial state. If the STM accurately captures the dependence of the final state on the initial state, then for small values of $\delta\mathbf{x}_0$, $\mathbf{x}^*_{pf}$ will approximate $\mathbf{x}_{pf}$ well. Given interest in particular situations where the propagated STM accurately captures this dependence, the following series of tests were conducted.

Setup

Table 2. Initial State Vector Used to Produce STM Linear Approximation Errors

State Component	Value
x (km)	6678.14
y (km)	0
z (km)	0
V_x (km/s)	0
V_y (km/s)	6.78953
V_z (km/s)	3.68641

All tests were performed on a LEO orbit with an initial orbit state given in Table 2. The propagated trajectory consisted of a propagation for t seconds, a finite maneuver, and another propagation for t seconds. The finite maneuver was performed for 100 seconds with an engine with 500 N of thrust along the spacecraft's velocity vector and an Isp of 300 s. Each test was performed as follows:

1. Propagate the trajectory and STM with some value for t to get $\mathbf{x}_f$.
2. Perturb the initial state by $\delta\mathbf{x}_0$ and propagate the trajectory using the same values for t as in (1) to get the perturbed final state $\mathbf{x}_{pf}$
3. Use the STM to calculate $\mathbf{x}^*_{pf}$ with Equation (40)
4. Calculate the difference $\delta\mathbf{x}_f = \mathbf{x}^*_f - \mathbf{x}_f$.

The value of $\delta\mathbf{x}_f$ gives a metric to evaluate how well the STM is able to approximate the propagation of a perturbed state. Lower values of $||\delta\mathbf{x}_f||$ indicate that a linear perturbation analysis with the STM is a good approximation.

Several different parameter values for t and $\delta\mathbf{x}_0$ were tested In order to evaluate this type of analysis in a range of environments. For t, propagation times of 5500, 27500, 55000 seconds were

tested. Multiples of 5500 seconds were chosen since 5500 is the approximate orbital period. The value of $\delta\mathbf{x}_0$ was found using Equation (41)

$$\delta\mathbf{x}_0 = (\mathbf{x}_0 + \mathbf{1})\Delta \tag{41}$$

where Δ is $\pm 10^{-4}$, $\pm 10^{-5}$, or $\pm 10^{-6}$. The results of each of these tests with each combination of parameters are given and discussed in the Results section.

Another interesting point of comparison for the STM accuracy is when there are stopping conditions that depend on the initial state. One example of such a stopping condition is the periapsis stopping condition. In order to test how useful the STM is using a linear perturbation analysis, the above steps were repeated with small modifications: instead of propagating for t seconds, the first propagation was for n apoapsis crossings occurring after 10 seconds of propagation, and the second propagation was for n periapsis crossings occurring after 10 seconds of propagation. An additional modification was made due to the fact that the propagation time of the perturbed trajectory, t_{pf}, could be different from the propagation time of the original trajectory, t_f. The final state $\mathbf{x}_f$ was taken to be $\mathbf{x}(t_{min})$ where $t_{min} = \min(t_f, t_{pf})$ is the minimum duration between the original and perturbed trajectories. Additionally, the STM used to estimate $\mathbf{x}_f$ was taken to be $\Phi_{t_{min}}(\mathbf{x}_0, t_0)$. All other parts of the procedure were kept the same.

Results

Table 3. STM Linear Approximation Position Errors for Duration Propagation

Propagation Time, t (seconds)	$\|\|\delta\mathbf{x}_{f,1-3}\|\|$ (kilometers)					
	$\Delta = 10^{-4}$	$\Delta = -10^{-4}$	$\Delta = 10^{-5}$	$\Delta = -10^{-5}$	$\Delta = 10^{-6}$	$\Delta = -10^{-6}$
5500	0.2451	0.2424	0.002996	0.002758	0.0001693	0.0001412
27500	6.288	6.301	0.06620	0.06547	0.002154	0.001924
55000	26.95	26.95	0.3030	0.2830	0.01235	0.01151

The STM approximation position errors for varying propagation durations are given in Table 3. These results are generally unsurprising; the approximation error is larger with larger initial state perturbations and more propagation time. Additionally, the linearization errors for positive state perturbations are approximately equal to those for negative state perturbations. The approximation errors when the state perturbations are small are fairly small. Therefore, the linear approximation generally does well to estimate the full propagation when initial perturbations are sufficiently small.

Table 4. STM Linear Approximation Position Errors for Apse Crossing Propagation

Number of Crossings, n	$\|\|\delta\mathbf{x}_{f,1-3}\|\|$ (kilometers)					
	$\Delta = 10^{-4}$	$\Delta = -10^{-4}$	$\Delta = 10^{-5}$	$\Delta = -10^{-5}$	$\Delta = 10^{-6}$	$\Delta = -10^{-6}$
1	0.6096	0.4439	0.04613	0.04402	0.004533	0.004489
5	6.236	6.426	0.09188	0.1018	0.007276	0.006401
10	26.53	27.12	0.2728	0.2630	0.003470	0.002514

As with the duration propagation tests, the STM does a fairly good job at approximating the final state in the apse crossing tests. It does especially well as the initial state perturbations get smaller. The orbital period for each of these satellites is approximately 5500 seconds, so the total propagation times between the duration tests and apse crossing tests are comparable. One interesting comparison

between the two tests is that for the shortest propagation time in both tests, the approximation errors for the apse crossing test are all larger than the corresponding errors for the duration test. However, for the longest propagation times, the trend almost reverses, with all the approximation errors for the apse crossing test being lower than the corresponding errors for the duration tests except for the $\Delta = -10^{-4}$ case.

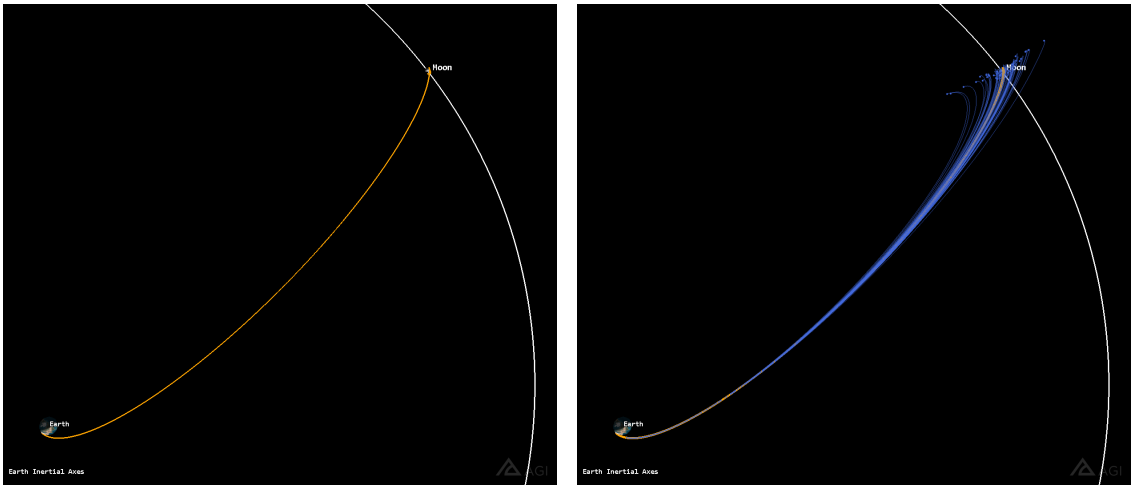
The presence of a minimum propagation time for the apse crossing tests significantly impacts the results for the tests. Without it, positive values of Δ trigger the first apoapsis crossing immediately, which results in the perturbed trajectory being vastly different from the original trajectory. Even though this analysis shows the STM captures the final state dependence on the initial state well, problem-specific dependencies can lead to unexpected changes in a trajectory due to small initial state perturbations that are not captured in the STM. However, these dependencies can be mitigated as long as the user is aware they exist.

EXAMPLE USE CASE

In preparation for operations, the orbit analysts identify risks, characterize the effects, and design methods to mitigate them. A major risk in trajectory design is that the uncertainties from Orbit Determination (OD) and maneuver execution performance errors will cause the trajectory to deviate to an extent that the mission requirements will not be met. The common mitigation technique is to alter subsequent maneuvers to compensate for the OD and maneuver uncertainties, and if needed, to insert extra Trajectory Correction Maneuvers (TCMs). These mitigation strategies, of course, can increase the required propellant, and the extent of the increase must be examined. A numerical method often used to analyze this risk is to model the uncertainties statistically with a Monte Carlo simulation. This simulation makes statistically significant random draws from the orbit state error covariance (calculated from OD) and random draws from the expected maneuver pointing and magnitude uncertainties and re-plans the subsequent trajectory maneuvers, including the TCMs, to calculate a statistical distribution on the likely propellant requirements. To obtain statistically significant information from the Monte Carlo, several thousand simulated runs are often needed. Using numerical integration to propagate all the deviations in the state and the maneuver can take many hours and even days. A depiction of a cislunar transfer nominal trajectory side by side with 50 Monte Carlo simulations is given in Figure 10.

Instead of running the high-fidelity orbit propagation for each deviated OD and maneuver state, the deviations can be propagated using the STM methods described above. For instance, in the case of a cislunar trajectory transferring a spacecraft from the Earth to the Moon, the deviations in the orbit state at perigee and deviations on a maneuver (e.g., at perigee) cause the trajectory to deviate as it approaches the Moon. In the Monte Carlo simulation thousands of deviations are created using random draws and propagated using a high-fidelity force model to the time of the next maneuver, which could be a TCM or the Lunar Orbit Insertion (LOI) capture at the Moon. However, instead of propagating the thousands of deviated orbits using numerical integration, the deviations can be propagated more quickly with a single matrix multiply of the STM and a single addition to the non-perturbed final orbit state.

A typical cislunar transfer trajectory can have an eccentricity of more than 0.96, and takes usually takes from 4.5 to 5.5 days to transfer from the perigee at Earth to the lunar orbit distance.[14–17] The STM tested for this case approximated the numerically integrated trajectories to better than a tenth of a percent, which is accurate enough to characterize the extra propellant needed to compensate for the OD and maneuver execution errors.



(a) Nominal Cislunar Transfer (b) 50 Monte Carlo Simulations

Figure 10. (a) A nominal 5-day cislunar transfer trajectory **(b)** Fifty runs of a Monte Carlo simulation with perturbations on the initial state sampled from uncertainty (such as from orbit determination or maneuver execution uncertainty).

Monte Carlo simulations are also used to estimate the probability in achieving other mission goals in addition to propellant limits. Continuing the example of a cislunar transfer to the Moon, such critical goals typically are the expected altitude at LOI, the lunar orbit inclination, the right ascension of the ascending node, and the argument of periselene. By propagating the deviations with the STM, the deviations in the mission goals can be calculated more quickly than using numerical integration to propagate the trajectory under the influence of the Earth, Sun, and Moon gravity fields.

This same Monte Carlo simulation technique is, of course, used for other missions besides lunar transfers. For example, heliocentric trajectories encountering a planet for a gravity assist, or to capture, require the same analysis as for encountering the Moon, and will benefit from the speed increase by using the STM. And similar statistical analysis with multiple spacecraft (such as formation flying or constellation station-keeping) can be performed substituting STM propagation for numerical orbit integration of every perturbation.

CONCLUSION

The STM is a powerful tool available to trajectory designers. It can be useful by itself, but in a complex environment like STK, it becomes very powerful. By adding more capabilities relating to the STM directly into Astrogator, trajectory designers not only have more individual tools to work with, but they also have new ways to combine tools to execute novel and interesting missions. Additionally, software like Astrogator reduces the code implementation burden on designers, so that they can focus more of their time on design and analysis.

ACKNOWLEDGMENT

The authors would like to thank Jim Woodburn and Vince Coppola for their guidance and assistance throughout these analyses. Their suggestions and feedback were essential to getting something that works well into the final product. Vince was also very involved in working out the math for the

STM after an impulse.

REFERENCES

[1] J. Carrico and E. Fletcher, "Software Architecture and Use of Satellite Tool Kit's Astrogator Module for Libration Point Orbit Missions," *Libration Point Orbits and Applications*, Girona, Spain, June, 2002.

[2] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. Springer–Verlag, 1989.

[3] T. E. Carter, "State Transition Matrices for Terminal Rendezvous Studies: Brief Survey and New Example," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 1, 1998, pp. 148–155, https://doi.org/10.2514/2.4211.

[4] V. Muralidharan and K. Howell, "Stationkeeping in Earth-Moon Near Rectilinear Halo Orbits," *AIAA/AAS Astrodynamics Specialist Conference*, South Lake Tahoe, California, August 2020.

[5] E. Zimovan, K. Howell, and D. Davis, "Near Rectilinear Halo Orbits and their Application in Cis-Lunar Space," *3rd IAA Conference on Dynamics and Controls of Space Systems*, Moscow, Russia, May 2017.

[6] B. Marchand, "Temporary Satellite Capture of Short-Period Jupiter Family Comets from the Perspective of Dynamical Systems," Master's thesis, School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana, 2000.

[7] Gaël Guennebaud and Benoît Jacob and others, "Eigen Documentation," April 2021. Online: `https://eigen.tuxfamily.org/index.php?title=Main_Page`, Accessed: 20 July 2021.

[8] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson, 2006.

[9] AGI Documentation Team and Subject Matter Experts, "Astrogator Documentation," July 2021. Online: `https://help.agi.com/stk/#gator/ab-prop.htm#numint`, Accessed: 16 July 2021.

[10] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis, Third Edition*. Springer-Verlag, 2002.

[11] C. Short, A. Haapala, and N. Bosanac, "Technical Implementation of the Circular Restricted Three-Body Model in STK Astrogator," *AIAA/AAS Astrodynamics Specialist Conference*, South Lake Tahoe, California, August 2020.

[12] C. Short, P. Ghosh, and A. Claybrook, "Revisiting Trajectory Design with STK Astrogator, Part 1," *AAS/AIAA Astrodynamics Specialist Conference*, Portland, Maine, August, 2019.

[13] B. D. Tapley, B. E. Schutz, and G. H. Born, *Statistical Orbit Determination*. Academic Press, 2004.

[14] H. Shyldkrot, E. Schmidt, D. Geron, J. Kronenfeld, J. Carrico, M. Loucks, L. Policastri, and J. Taylor, "The First Commercial Lunar Lander Mission: Beresheet," *AAS/AIAA Astrodynamics Specialist Conference*, Portland, Maine, August 2019.

[15] M. Loucks, L. Plice, D. Cheke, C. Maunder, and B. Reich, "The Ladee Trajectory as Flown," *25th AAS/AIAA Spaceflight Mechanics Meeting*, Williamsburg, Virginia, January 2015.

[16] M. Mesarch, M. Beckman, D. Folta, R. Lamb, and K. Richon, "Maneuver Operations Results from the Lunar Reconnaissance Orbiter (LRO) Mission," *SpaceOps 2010 Conference Delivering on the Dream*, HuntsVille, Alabama, April 2010.

[17] D. Carrington, J. Carrico, C. Roberts, A. Seacord, P. Sharer, L. Newman, K. Richon, B. Kaufman, J. Middour, *et al.*, "Trajectory Design for the Deep Space Program Science Experiment (DSPSE) Mission," *Proceedings of the AAS/NASA International Symposium*, Greenbelt, Maryland, January 1993.