(Preprint) AAS 11-527

A GENERAL EVENT LOCATION ALGORITHM WITH APPLICATIONS TO ECLIPSE AND STATION LINE-OF-SIGHT

Joel J. K. Parker*and Steven P. Hughes†

A general-purpose algorithm for the detection and location of orbital events is developed. The proposed algorithm reduces the problem to a global root-finding problem by mapping events of interest (such as eclipses, station access events, etc.) to continuous, differentiable event functions. A stepping algorithm and a bracketing algorithm are used to detect and locate the roots. Examples of event functions and the stepping/bracketing algorithms are discussed, along with results indicating performance and accuracy in comparison to commercial tools across a variety of trajectories.

INTRODUCTION

One of the fundamental problems in spacecraft mission design is that of orbital event location; that is, the determination of the start and end times and durations of various events of interest along the trajectory of a spacecraft. While event location can be defined generally, in the context of spacecraft trajectories it includes such examples as the start and end times of an eclipse period, the start and end times of communications contact with a ground station, and the time of closest approach between two spacecraft. This is critically important to several disciplines, including spacecraft design and mission planning. Spacecraft design engineers must know the durations of eclipse events to properly perform power system design, and mission planners must know the event start and end times to schedule communication and data collection. Likewise, mission planners and operators must know the timing and duration of ground station line-of-sight events so that communication and tracking can be established.

There exist several tools that perform event location for spacecraft trajectories. These include NASA tools such as the General Maneuver Program (GMAN),[1] Swingby,[2] and the Acquisition Data Program (ACQSCAN),[3] as well as commercial off-the-shelf (COTS) tools such as AGI's STK and A.I. Solutions' FreeFlyer. The Navigation and Mission Design Branch (NMDB) at NASA Goddard Space Flight Center is currently leading development of the General Mission Analysis Tool (GMAT), a new open-source mission design tool that is designed for eventual operational use. However, GMAT lacks the capability to perform event location, a necessary operational requirement.

The purpose of this study was to develop an algorithm to perform event location for spacecraft trajectories and to demonstrate the approach with a prototype software tool. Based on this expe-

*Flight Dynamics Engineer, Navigation and Mission Design Branch, Code 595, NASA Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt, MD 20771.
†Flight Dynamics Engineer, Navigation and Mission Design Branch, Code 595, NASA Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt, MD 20771.

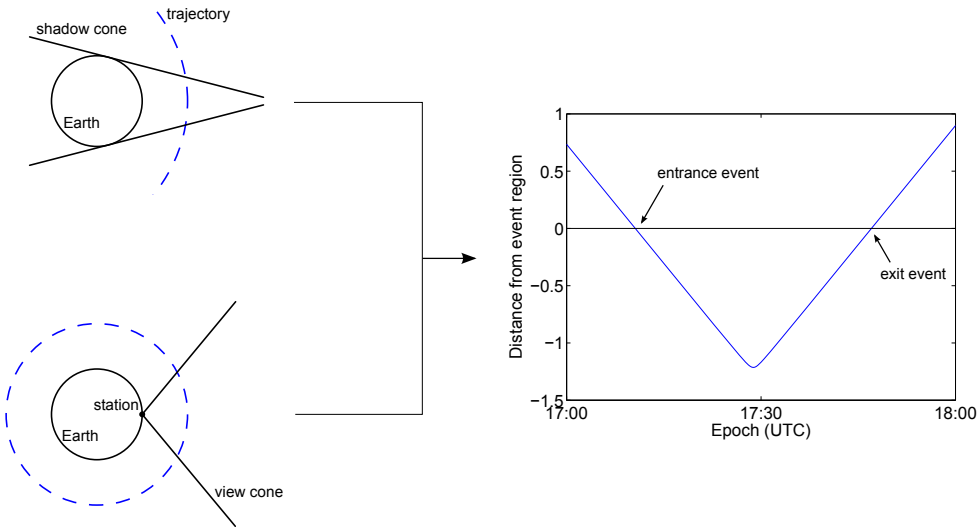


Figure 1: Eclipse (top) and station line-of-sight (bottom) physical and function representations

rience, parts of this algorithm will be implemented in GMAT to provide it with an event location capability.

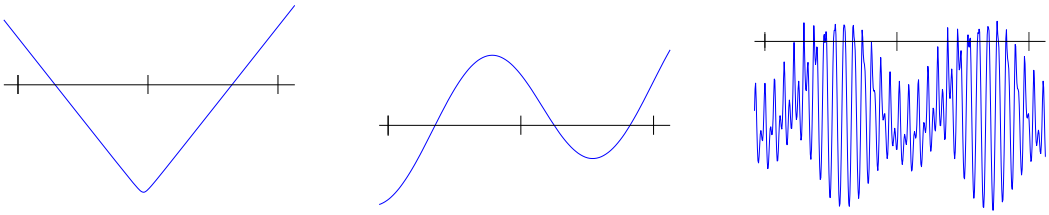
At its core, event location is a global root-finding problem. This can be seen graphically in Figure 1, which in the upper left shows the physical situation of a spacecraft trajectory passing through a shadow caused by a celestial body (such as Earth). This situation can also be represented in a functional form similar to that shown in the graph at the right of the figure. As the spacecraft travels along its trajectory, its "distance" from the event boundary (in this case, entrance to or exit from the shadow region) can be plotted as a function of time. This is referred to as the event function. The measure of distance can be chosen based on the application, but in all cases the instant of the event is represented by an ordinate value of zero. In Figure 1, while the spacecraft is outside the shadow region its distance value is positive, while inside the shadow region it is negative, though this is not a requirement. The diagram in the lower left shows a similar case involving a spacecraft travelling through the visibility cone of a ground station, which can be represented in functional form in an identical way.

By defining the appropriate measures of distance for each type of event, a functional representation of that event can be generated, and the times of those events can be determined by finding the roots of that function.

There are several complicating factors that make this problem non-trivial. The most basic information available about a spacecraft trajectory is its ephemeris, or time history of position and velocity states. These states occur at discrete times (evenly spaced or not) and represent individual points on the function plots represented in Figure 1. Given only the values of the event function at these times, three cases could occur, as shown in Figure 2.

Figure 2a shows the trivial case, in which there exists a single root (and therefore a single event) between each pair of ephemeris steps shown by vertical lines on the x-axis. Given these points, the roots can be found by applying a classical root-finding method such as bisection or Brent's.[4]

Figure 2b shows the case in which two roots exist between the final two ephemeris steps in the figure. In this case, the roots are not individually bracketed, so a bisection-like method will fail.



(a) Type I—One root (b) Type II—Two roots (c) Type III—Many roots

Figure 2: Types of event function behavior between ephemeris steps

However, by considering the derivative of the function (which has a single root between the steps), both roots can be found.

Figure 2c shows the final case, in which many roots (three or more) occur between ephemeris steps. This case cannot be handled in a general way by considering the derivatives of the function, and so must be prevented from occurring. This is the goal of the algorithm development in the next section.

ALGORITHMS

Three classes of algorithms were developed as a part of this study: event functions, function steppers, and an event detector.

Event Functions

An event function is a mathematical representation of a physical scenario, such as a spacecraft travelling through a region of shadow (see Figure 1). It can be defined arbitrarily, but for the purposes of this algorithm, it must behave according to certain rules. The event function must:

- be continuous,
- have a known first derivative, and
- have roots at the times of the events (and only at these times).

In general, the event function is a measure of distance of the spacecraft at any point in time from the event occurrence. This study focuses on detecting eclipse and ground station line-of-sight events, which combined require a total of five different event functions.

An eclipse event is the entrance to or exit from a region of shadow created by a celestial body blocking all or part of the sun's rays. The shadow region can be one of three types, as shown in Figure 3.

The planar geometry for the umbra function is shown in Figure 4. In this construction, the umbral cone created by the central body is shown by the solid tangent lines, and the spacecraft trajectory by the dashed curve. At any instant in time, the satellite position (with radius vector $\mathbf{r}$) defines a circle about the shadow body of radius R_b. This circle intersects the umbral cone at the point located by $\mathbf{e}$, which makes an angle η with the edge of the cone. The angle γ_u between the vectors $\mathbf{r}$ and $\mathbf{e}$ defines the event function, which depends only on the satellite radius vector $\mathbf{r}$, the vector $\mathbf{v}$ from the

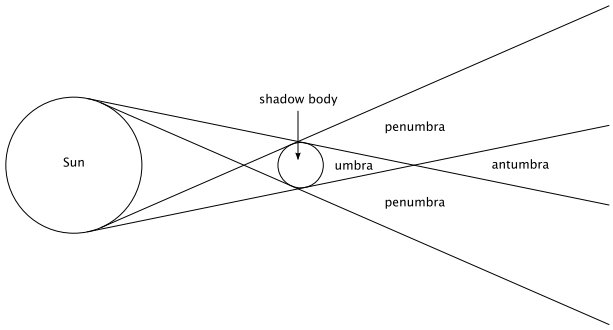


Figure 3: Shadow types

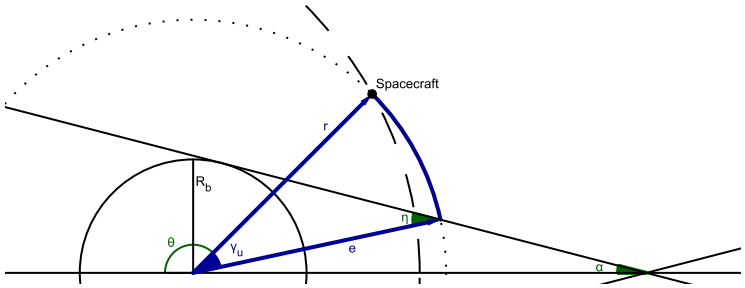


Figure 4: Umbra event function

shadow body to the sun, and the radii of the sun (R_s) and shadow body (R_b). The event function itself is:

$$\gamma_u = \pi - \theta + \alpha - \eta, \tag{1}$$

where

$$\theta = \cos^{-1}\left(\frac{\mathbf{v}\cdot\mathbf{r}}{|\mathbf{v}|\,|\mathbf{r}|}\right), \tag{2}$$

$$\alpha = \sin^{-1}\left(\frac{R_S - R_b}{|\mathbf{v}|}\right), \text{and} \tag{3}$$

$$\eta = \sin^{-1}\left(\frac{R_b}{|\mathbf{r}|}\cos(\alpha)\right). \tag{4}$$

The first derivative of the umbra function is found by differentiating the equations above with respect to time:

$$\dot{\gamma_u} = -\dot{\theta} + \dot{\alpha} - \dot{\eta}, \tag{5}$$

where

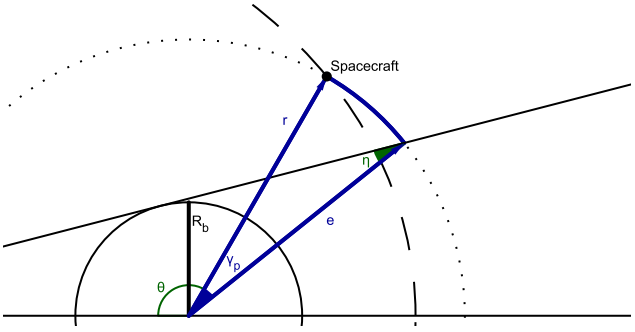


Figure 5: Penumbra event function

$$\dot{\theta} = -\frac{\frac{\dot{\mathbf{v}}\cdot\mathbf{r}+\mathbf{v}\cdot\dot{\mathbf{r}}}{|\mathbf{v}|\,|\mathbf{r}|} - \frac{(\mathbf{v}\cdot\mathbf{r})\,(\dot{\mathbf{v}}\cdot\mathbf{v})}{|\mathbf{v}|^3\,|\mathbf{r}|} - \frac{(\mathbf{v}\cdot\mathbf{r})\,(\dot{\mathbf{r}}\cdot\mathbf{r})}{|\mathbf{v}|\,|\mathbf{r}|^3}}{\sin(\theta)}, \tag{6}$$

$$\dot{\alpha} = -\frac{(R_S - R_b)\,(\dot{\mathbf{v}}\cdot\mathbf{v})}{|\mathbf{v}|^3\cos(\alpha)}, \text{ and} \tag{7}$$

$$\dot{\eta} = R_b\frac{\frac{-\dot{\alpha}\sin\alpha}{|\mathbf{r}|} - \frac{(\dot{\mathbf{r}}\cdot\mathbf{r})\cos(\alpha)}{|\mathbf{r}|^3}}{\cos(\eta)}. \tag{8}$$

A similar construction can be defined for the penumbra shadow event, as shown in Figure 5. Its equation is

$$\gamma_p = \pi - \theta - \phi - \delta, \tag{9}$$

where θ is defined above,

$$\phi = \sin^{-1}\left(\frac{R_S + R_b}{|\mathbf{v}|}\right), \text{ and} \tag{10}$$

$$\delta = \sin^{-1}\left(\frac{R_b}{|\mathbf{r}|}\cos(\phi)\right). \tag{11}$$

The derivative $\dot{\gamma_p}$ is found by differentiating the equations above in the same manner as $\dot{\gamma_u}$. Likewise, the equation for the antumbra function, shown geometrically in Figure 6, is

$$\gamma_a = \pi - \theta - 2\alpha + \eta, \tag{12}$$

where θ, α, and η are defined above.

Finding ground station line-of-sight events relies on the coupling of two different function types: the elevation function for the station's central body and the line-of-sight function for third bodies.

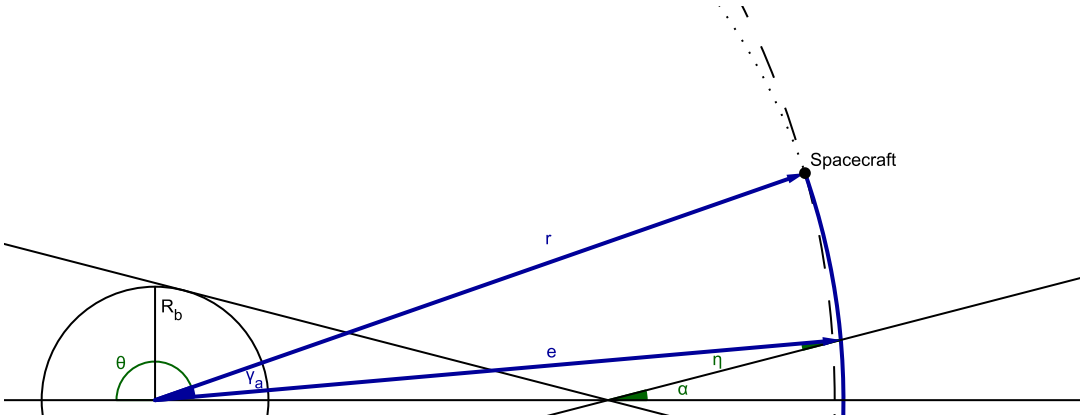


Figure 6: Antumbra event function

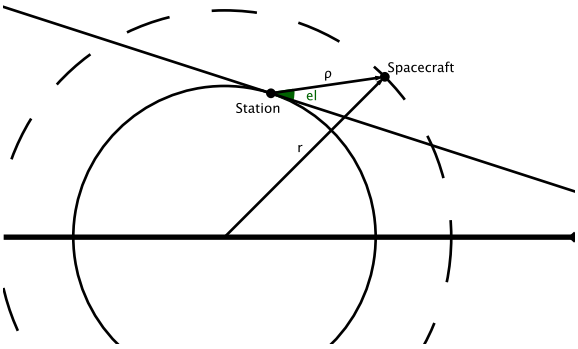


Figure 7: Elevation event function

The elevation function represents the visibility of the satellite from the ground station above its local horizon. This geometry is shown in Figure 7. The definition of the ground elevation angle is

$$el = \sin^{-1}\left(\frac{\rho_z}{\rho}\right), \tag{13}$$

where ρ is the spacecraft range and is equivalent to $|\mathbf{r}|$.[5] However, this formulation is not smooth through a single pass, which is not ideal. Since any arbitrary function can serve as the event function as long as it has identical roots to Eq. (13),

$$e_f = \sin\left(\frac{\rho_z}{\rho}\right) \tag{14}$$

and its derivative

$$\dot{e_f} = \cos\left(\frac{\rho_z}{\rho}\right)\left(\frac{\dot{\rho_z}}{\rho} - \frac{\rho_z\dot{\rho}}{\rho^2}\right). \tag{15}$$

can be substituted. In this formulation, the elevation angle is

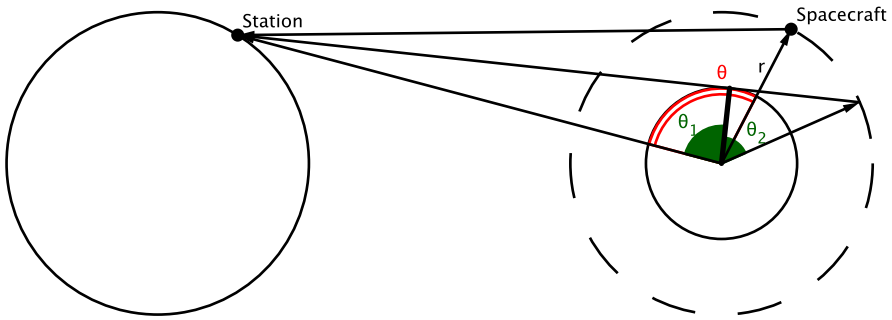


Figure 8: Line-of-sight event function

$$el = \sin^{-1}\left(\sin^{-1}\left(e_f\right)\right) \tag{16}$$

The station line-of-sight function geometry is shown in Figure 8. This function captures the obstruction of line of sight between the ground station and the spacecraft through any bodies other than the central body of the station. The formulation relies on the magnitudes of the angles θ_1 and θ_2, the sum of which represents the maximum angle at which line-of-sight is preserved, and θ, the actual angle between the spacecraft and ground station vectors from the center of the obstructing body. The equation is

$$l = \theta_1 + \theta_2 - \theta, \tag{17}$$

where

$$\theta_1 = \cos^{-1}\left(\frac{R_b}{|\mathbf{r}|}\right), \tag{18}$$

$$\theta_2 = \cos^{-1}\left(\frac{R_b}{|\mathbf{r_{gs}}|}\right), \text{ and} \tag{19}$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{r}\cdot\mathbf{r_{gs}}}{|\mathbf{r}|\,|\mathbf{r_{gs}}|}\right). \tag{20}$$

Stepping Algorithms

To locate the roots of these event functions (and thus locate the events), the functions can be integrated along with the trajectory, or they can be evaluated in a post-processing step. In the former case, the integrator will be able to adjust the step size based on the event function behavior using a method such as that developed by Shampine.[6] However, for simplicity for the purposes of this study, a postprocessing method was used, which requires a search technique for its roots. There are two classes of search methods that can be applied: fixed steppers and adaptive steppers.

A fixed stepper is the simplest of the search methods; it uses a fixed interval (or step size) and moves along the function by that interval, searching for roots between each step. In equation form, this is represented as

$$t_1 = t_0 + h, \tag{21}$$

where t_0 is the current step, t_1 is the next step, and h is the step size. However, this method has several drawbacks. With a step size chosen too small, many more steps are taken than are required, reducing performance. With a step size chosen too large, there exists the possibility of missing events, especially if more than two roots occur between adjacent steps. And if the function frequency changes within the ephemeris interval, it is possible that a well-chosen step size may be too small for one portion and too large for another.

In general, it is much more efficient to search for roots using an adaptive method, which adjusts its behavior using knowledge of the behavior of the function (through its derivatives). There are several potential methods available, including frequency analysis, polynomial approximation, and Taylor methods to automatically adjust the step size sized based on the local function behavior. Of these options, the Taylor method is straightforward to understand and implement, and is used widely as a part of numerical integration methods (such as Runge-Kutta methods).

The Taylor adaptive stepping method is based on the error control method used by adaptive Runge-Kutta numerical integration algorithms.[7] In this method, two approximations are made of the event function value at the next time step. As an example, the value of the function at time t_1 may be approximated using a second-order Taylor approximation,

$$f_1(t_1) = f(t_0) + hf'(t_0) + \frac{h^2}{2} f''(t_0) + O\left(h^3\right), \tag{22}$$

while a third-order approximation of the same function value is

$$f_2(t_1) = f(t_0) + hf'(t_0) + \frac{h^2}{2} f''(t_0) + \frac{h^3}{6} f'''(t_0) + O\left(h^4\right). \tag{23}$$

The difference between the two approximations can be interpreted as an error value using the formula

$$err = \frac{1}{h} \left| f_1(t_1) - f_2(t_1) \right|, \tag{24}$$

where h, again, is the step size. This error value can then be compared against a user-supplied tolerance. If the error is less than tolerance, the step size may be too small, while if it is greater than tolerance, the step size may be too large.

The step size itself is controlled by using a quality factor defined by

$$q = s_f \left(\frac{tol}{err} \right)^{\frac{1}{n}}, \tag{25}$$

where tol is the error tolerance, n is the order of the lower-order approximation (in this example $n = 2$), and s_f is a safety factor, usually 0.9 or 0.95 (though in this analysis s_f was left at 1). The quality factor is defined such that if the error is greater than the tolerance, $q < 1$. If the error is less than the tolerance, $q > 1$.

In Eq. (22)–(23), $f(t_0)$ and $f'(t_0)$ are known (these are the values of the event function and the event function derivative, respectively). However, the higher derivatives of f are not known, and approximations must be used.

The higher derivatives of f can be approximated by finite differencing. For example, the classic central difference formula for the first derivative of f is

$$f'(t_i) = \frac{1}{h}\left[-\frac{1}{2}f(t_{i-1}) + \frac{1}{2}f(t_{i+1})\right] + O\left(h^2\right). \tag{26}$$

This formula uses the two surrounding time steps (that is, $t_i - h$ and $t_i + h$) to approximate the value of f' at the current step with an error on the order of h^2. Similar formulas exist for forward differencing (using the current and next time steps) and backward differencing (using the current and previous time steps). Eq. (26) can be easily adapted to find the second derivative of f based on its first derivative, as given by

$$f''(t_i) = \frac{1}{h}\left[-\frac{1}{2}f'(t_{i-1}) + \frac{1}{2}f'(t_{i+1})\right] + O\left(h^2\right). \tag{27}$$

Formulas with identical error terms also exist for higher derivatives, such as

$$f'''(t_i) = \frac{1}{h^2}\left[f'(t_{i-1}) - 2f'(t_i) + f'(t_{i+1})\right] + O\left(h^2\right) \tag{28}$$

and

$$f^{(4)}(t_i) = \frac{1}{h^3}\left[-\frac{1}{2}f'(t_{i-2}) + f'(t_{i-1}) - f'(t_{i+1}) + \frac{1}{2}f'(t_{i+2})\right] + O\left(h^2\right). \tag{29}$$

In this way, Taylor methods up to the fourth order can be derived using finite difference approximations through the fifth derivative. This method can easily be extended further, with the consideration that more time steps, and therefore more function evaluations, will be included in the calculation.

The overall adaptive stepping algorithm is illustrated in Figure 9, where the entire sequence is repeated at each step. First, Eqs. (22)–(23) (or their higher-order equivalents) are evaluated using the current step size to approximate the function value at the next step. The two approximations are compared and the error value and quality factor are calculated. The error value is compared to a user-defined tolerance, and if the tolerance is met, the step is accepted and applied. The quality factor is multiplied by the current step size to increase it, and the procedure moves to the next step. However, if the tolerance comparison fails, the quality factor is used to decrease the step size, and the process is repeated for the same step until tolerance is met.

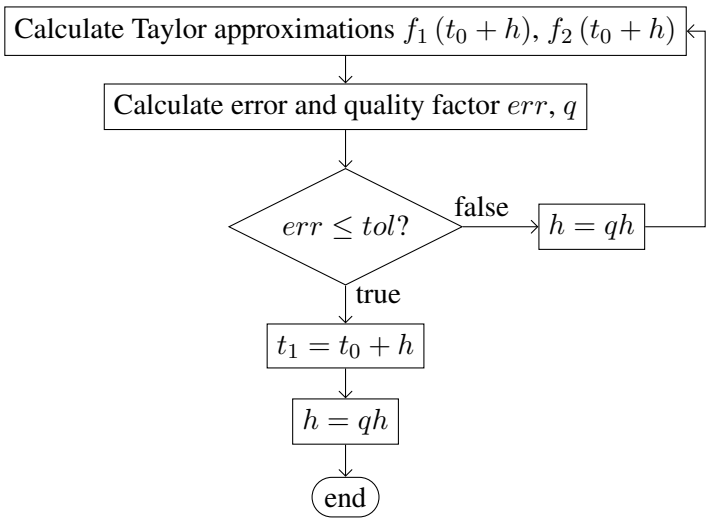


Figure 9: Adaptive stepping algorithm

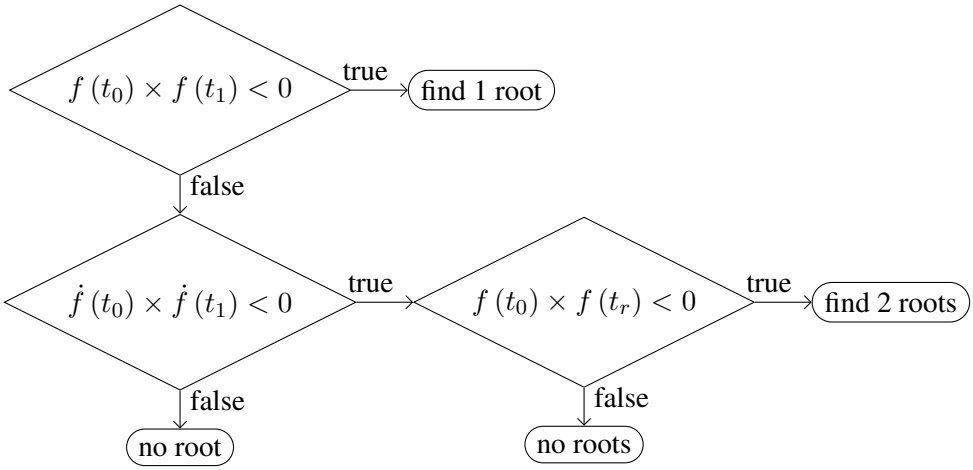


Figure 10: Event detection algorithm

The effect of this is that in intervals where the function is slowly changing, the step size is increased to improve performance, and where the function is changing rapidly, the step size is decreased to preserve accuracy.

Event Detector

To find the roots of the event function (and therefore find the event times), the system applies the event detection algorithm, shown in Figure 10, at each time step produced by the stepping algorithm. The detection algorithm performs a series of checks to determine if a root has occurred between the current and next steps. It begins by checking the signs of the function values at the two steps; if they are different a single root exists between the steps and a local root finding method (such as Brent's method) is used to find it. If the function values have the same sign, but the signs of their derivative values are different, then there may exist a maximum of two roots between the steps. In this case, the local root finding method is used to find the root of the derivative (and thus the extremum of the

original function). Once the time value of the extremum is found, the signs of the original function values at that point and at one of the endpoints are compared to determine the existence of any roots between. If the signs are opposite, there exists two roots, one between the extremum point and each endpoint. In the case of a discrete event (one without a beginning and ending event), the extremum itself will be zero, and a single root will be found.

IMPLEMENTATION

The algorithms discussed in the previous section were implemented in a standalone prototype event locator tool as a proof of concept. The tool was written in object-oriented MATLAB and operates in a post-processing mode on the ephemeris for a single spacecraft.

The event locator uses as input an ephemeris file in the format of a Consultative Committee for Space Data Systems (CCSDS) Orbit Ephemeris Message[8] and reports event data either to the screen or to a file. A project file in the text-based YAML format[9] is used to specify options. For eclipse events, the system supports umbra, penumbra, and antumbra events for any of the major celestial bodies. The MATLAB version of the SPICE Toolkit (version N0064)[10] is used for celestial body ephemeris data, and custom bodies can be used by supplying an additional SPK file and indicating its path. Ground station line-of-sight can be calculated for any user-defined station. Stations are specified in the system's project file by supplying a name, a central body, and planetodetic coordinates. In addition, multiple algorithmic settings can be set by the user including the stepping algorithm to use (fixed or adaptive and adaptive orders 2–4), the stepping algorithm tolerance, and the local root-finding method (Brent's, bisection, secant) and its tolerance.

For each event type requested by the user (e.g. umbra eclipse, penumbra eclipse, ground station line-of-sight), the system processes a series of individual event functions using the desired stepping method. The number of functions varies according to the type of event requested and the number of celestial bodies involved: if only umbral eclipse events at Earth are requested, a single event function is processed. However, if all three eclipse types are requested for both Earth and Moon, as well as ground station line-of-sight for an Earth-based station, a total of 13 event functions are processed: umbra, penumbra and antumbra for each eclipse body, the elevation function at the Earth ground station, and line-of-sight through all other celestial bodies defined by default (Mercury, Venus, Moon, Mars, Phobos, Deimos). This can be reduced by future enhancements to only consider line of sight through user-requested bodies. Further, if the ephemeris file contains two segments, the number of event functions processed increases to 26, because each segment must be processed individually due to potential discontinuities caused by impulsive maneuvers or dynamics model changes.

The system is currently capable of locating events along arbitrary trajectories, considering any celestial body that can be supplied in a SPICE-compatible format, and considering any ground station that can be specified by planetodetic coordinates on any of those celestial bodies. The results are collected from all requested event types, combined if necessary, and reported in a user-friendly text format. As it was developed as a proof of concept, the system assumes spherical celestial bodies and does not include light-time corrections.

TESTING AND RESULTS

To test the event locator a series of nine test cases was created involving different types of trajectories and multiple classes and combinations of events. The cases are listed in Table 1.

Test Case	Description	Eclipse events	Station LOS events
Simple	Quick LEO test	Earth umbra/penumbra	Equatorial station
LEO	Earth orbit with Moon umbra	Earth umbra/penumbra Moon umbra/penumbra	NASA GSFC
HEO1	Highly-elliptical Earth orbit with long eclipse	Earth umbra/penumbra	Canberra
HEO2	Highly-elliptical Earth orbit with short eclipse	Earth umbra/penumbra	Madrid
Lunar	Moon orbit with Earth shadows and Earth station access	Moon umbra/penumbra Earth umbra/penumbra	Moon station Plesetsk
Martian	Mars orbit with Phobos shadows and Earth station access	Mars umbra/penumbra Phobos penumbra	Mars station Kwajalein
Earth Transit	Transit of Earth/Moon from Mars orbit with Earth/Moon station access	Earth penumbra/antumbra Moon penumbra/antumbra Mars umbra/penumbra Phobos penumbra	Moon station Wallops
Maven	MAVEN escape trajectory with Earth eclipse	Earth umbra/penumbra	none

Table 1: Test Cases

		Simple		LEO		HEO 1		HEO 2		Lunar		Martian		Earth Transit		Maven
		Eclipse	Station	Eclipse	Station	Eclipse	Station	Eclipse	Station	Eclipse	Station	Eclipse	Station	Eclipse	Station	Eclipse
Fixed 60s	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.02	1.73	4.49	1.59	2.96	0.04	883.24	91.69	271.05	0.06
	Run Time	0.81	0.85	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Fixed 300s	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.03	1.73	4.49	1.59	2.96	0.04	883.24	91.69	271.05	0.06
	Run Time	0.18	0.22	0.23	0.25	0.21	0.21	0.20	0.21	0.21	0.21	0.22	0.23	0.21	0.21	0.20
Fixed 600s	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.02	1.72	4.49	1.59	2.96	0.04	883.24	91.69	271.05	0.06
	Run Time	0.11	0.14	0.14	0.15	0.11	0.11	0.10	0.11	0.11	0.12	0.13	0.14	0.11	0.11	0.10
Fixed 1200s	Max Diff.	0.06	0.01		0.01	1.43	1.02	1.73	4.49	1.59	2.96	0.03	883.24	91.68	271.05	0.06
	Run Time	0.08	0.10		0.11	0.06	0.06	0.06	0.06	0.06	0.07	0.08	0.09	0.06	0.06	0.05
Fixed 2400s	Max Diff.	0.06	0.01		0.01	1.43	1.03	1.73	4.49	1.60	2.96			91.68	271.05	0.06
	Run Time	0.05	0.06		0.06	0.03	0.04	0.03	0.04	0.04	0.04			0.04	0.04	0.03
Adaptive 2/3	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.03	1.73	4.49	1.59	2.97	0.03	883.24	91.68	271.05	0.06
	Run Time	1.00	1.00	0.95	0.55	0.22	0.13	0.17	0.10	0.22	0.13	0.77	0.78	0.29	0.26	0.10
Adaptive 3/4	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.03	1.72	4.49	1.59	2.96	0.04	883.24	91.68	271.05	0.06
	Run Time	0.84	0.55	0.51	0.26	0.15	0.10	0.12	0.08	0.17	0.11	0.47	0.49	0.22	0.21	0.08
Adaptive 4/5	Max Diff.	0.06	0.01	1.47	0.01	1.43	1.03	1.73	4.49	1.59	2.96	0.04	883.24	91.68	271.05	0.06
	Run Time	0.74	0.58	0.42	0.24	0.14	0.10	0.12	0.08	0.15	0.11	0.39	0.41	0.21	0.20	0.09

Table 2: Test results

All cases, with the exception of Maven, contain both eclipse and ground station line-of-sight events. The Maven case is intended to test a specific eclipse case and thus contains no ground stations. Each case was implemented as a MATLAB function using the object model in STK 9.2.1. STK was used to generate an ephemeris file, an eclipse report, and a station access report for each ground station defined in the test case. Then a series of event locator project files was created for each case, each testing a different stepping algorithm. The event locator was run using each project file in an automated way, and its eclipse and station line-of-sight reports were compared to those generated from STK. The test results (defined as a pass if all events were detected and their times and durations were within tolerance, nominally set at 5 seconds), the actual maximum time difference for all events in the test, and the number of function evaluations executed during the run were all recorded.

To ensure that both tools were using identical input values, certain STK data files were edited to match the values used by the SPICE kernels in the event locator. The changes included:

- Setting each central body radius to match the values given by SPICE
- Setting each central body to spherical shape
- Adding the appropriate ephemeris kernels to STK's SPICE directory
- Setting each central body to use SPICE as its ephemeris source

Table 2 shows the results for each test case (column groups) and each stepping method (row groups). The cases that pass by matching all events within tolerance (5 seconds) are marked green; cases that detect all events but with a larger time difference are marked yellow. There are several cases that do not detect all events used the fixed step method with large step sizes; these are considered as expected failures and are marked with a crosshatch pattern.

In general, the large majority of cases detect all events with very small time differences (many on the order of hundredths of seconds). The Earth Transit case falls outside of the 5-second tolerance in all instances, while the Martian case falls outside of tolerance only for station line-of-sight events. These cases all include trajectories at Martian distances, which exacerbates certain known modeling differences between the event locator and STK. For line-of-site events, the event locator does not consider light-time delays in its calculations. This by itself accounts for the vast majority of the timing errors in these line-of-sight cases. In the case of Earth Transit eclipse events, there is a

difference in Sun/Earth position between SPICE and STK on the order of hundreds of kilometers. This is by far the largest discrepancy between the two systems for the eclipse functions, and probably accounts for the approximately 90-second event time differences. More research into the issue is necessary to confirm this.

CONCLUSIONS

The objective of this study was to develop a general algorithm for event location and apply it to eclipse and ground station line-of-sight events. This was accomplished by developing the necessary event functions for the three eclipse types and the two station line-of-sight criteria, a fixed-step and several adaptive-step search methods, and an algorithm for event detection between steps. These algorithms were implemented in MATLAB as a prototype event locator tool and tested by comparing to existing software tools.

There are several avenues of continued development. There exists the possibility of improving the event detection algorithm by implementing a local polynomial approximation scheme to increase efficiency. This scheme would use an approximating polynomial between the two steps to check for the possibility of roots between them; if the polynomial has no real roots, it can be assumed that the original function also does not and the function itself does not need to be evaluated. There are also potential efficiency improvements available by implementing higher-order adaptive stepping methods, using the framework described previously.

From a software perspective, the addition of a graphical user interface would improve overall usability. Because the rationale for this study was the lack of an event location capability in GMAT, it is expected that these algorithms will be implemented in GMAT as native code, using the event locator tool as a prototype. Finally, there is also interest in releasing this tool as a standalone utility, an option that remains to be explored.

NOTATION

$\mathbf{e}$	vector from shadow body center to intersection of spacecraft auxiliary circle and umbral cone
e_f	elevation function value
el	spacecraft elevation measured from ground station
err	Taylor approximation error
h	step size
$\mathbf{r}$	spacecraft position vector
q	adaptive stepper quality factor
R_b	shadow body radius
R_S	Sun radius
s_f	adaptive stepper safety factor
t	time
$\mathbf{v}$	vector from center of shadow body center to Sun
α	umbral half-cone angle
γ_a	antumbra event function value
γ_p	penumbra event function value
γ_u	umbra event function value
η	angle between $\mathbf{e}$ and the umbral cone

θ angle between $\mathbf{r}$ and the sunward side of the shadow body-Sun line
θ angle between spacecraft and ground station position vectors, measured from the center of an external celestia
θ_1, θ_2 auxiliary angles for line-of-sight function
ϕ penumbral half-cone angle
ρ spacecraft range

REFERENCES

[1] "General Maneuver Program (GMAN) Mathematical Specifications, Revision 1, Update 7," tech. rep., Mission Operations and Data Systems Directorate, NASA Goddard Space Flight Center, Greenbelt, MD, November 1993.

[2] "Mission Analysis and Design Tool (Swingby) User's Guide, Revision 3," tech. rep., Flight Dynamics Division, NASA Goddard Space Flight Center, Greenbelt, MD, September 1995.

[3] "Acquisition Data Program (ACQSCAN) Functional Description and User's Guide," tech. rep., Flight Dynamics Division, NASA Goddard Space Flight Center, Greenbelt, MD, 1995.

[4] R. Brent, *Algorithms for Minimization Without Derivatives*. Prentice-Hall Series in Automatic Computation, Prentice-Hall, 1972.

[5] D. A. Vallado, *Fundamentals of Astrodynamics and Applications*. El Segundo, CA: Microcosm Press, 2nd ed., 2001.

[6] L. F. Shampine, I. Gladwell, and R. W. Brankin, "Reliable Solution of Special Event Location Problems for ODEs," *ACM Transactions on Mathematical Software*, Vol. 17, March 1991, pp. 11–25.

[7] R. L. Burden and J. D. Faires, *Numerical Analysis*. Belmont, CA: Thomson, 2005.

[8] "Orbit Data Messages," Tech. Rep. CCSDS 502.0-B-2, The Consultative Committee for Space Data Systems, November 2009.

[9] O. Ben-Kiki, C. Evans, and I. d. Net, "YAML Ain't Markup Language (YAML) Version 1.2," Tech. Rep. 3rd Edition, October 2009.

[10] "The SPICE Toolkit, version N0064," June 2010.