EVE: THE FINAL FRONTIER

A Launch Vehicle Designer Tutorial

Arrowstar

March 16, 2019

Table of Contents

1 Purpose and Scope

Launch Vehicle Designer, often referred to as “LVD,” is the latest general-purpose trajectory design tool within the KSP Trajectory Optimization Tool (KSPTOT) suite of software. Originally released in KSPTOT v1.6.0 and conceived as a way to optimize launch vehicle and ascent trajectories, LVD has since expanded to be capable of performing trajectory design across all mission phases, including launch, cruise, landing, aerobraking, and everything in between.

The goal of the tutorial is to walk you, the mission analyst, through the process required to design a sample return mission to Eve. All phases of the mission will be covered: Kerbin ascent and departure, interplanetary coasts, Eve arrival and landing, and Eve ascent and departure. All of the high-fidelity mission design work will be done using the Launch Vehicle Designer tool.

This tutorial assumes that you have familiarity with the porkchop plotter application and “Compute Departure” application of KSPTOT. It also assumes a basic familiarity with the concepts of space mission design and planning. No familiarity with LVD is assumed.

All times shown will be in the “Earth” time system. If necessary, you can change this appropriately in the Edit -> Time System menu of the main KSPTOT user interface.

2 Early Mission Design

As is the case with all mission design, our analysis will start out with low fidelity modeling of the basic trajectories and then move to the higher fidelity LVD tool. We will look at two different areas.

First, we must understand roughly what our interplanetary trajectories will look like. This means that we must determine approximately when we will depart Kerbin, arrive at Eve, depart Eve, and arrive back at Kerbin. We also want to know roughly what our interplanetary orbits are and what our Kerbin and Eve departure hyperbolic excess velocity vectors are.

Second, we must size the vehicle required to carry out this mission. This means looking at roughly how much delta-v is required for each leg of the mission and using that knowledge to determine the size of fuel tanks and structure. Our level of analysis will not get down to picking individual components within KSP itself, but could be used to help determine those after the trajectory design has been completed.

2.1 Conceptual Interplanetary Trajectory

2.1.1 Launch Site Location

We will assume that we are launching from the planet Kerbin at the Kerbin Space Center (KSC). The coordinates of this facility are shown in the table below.

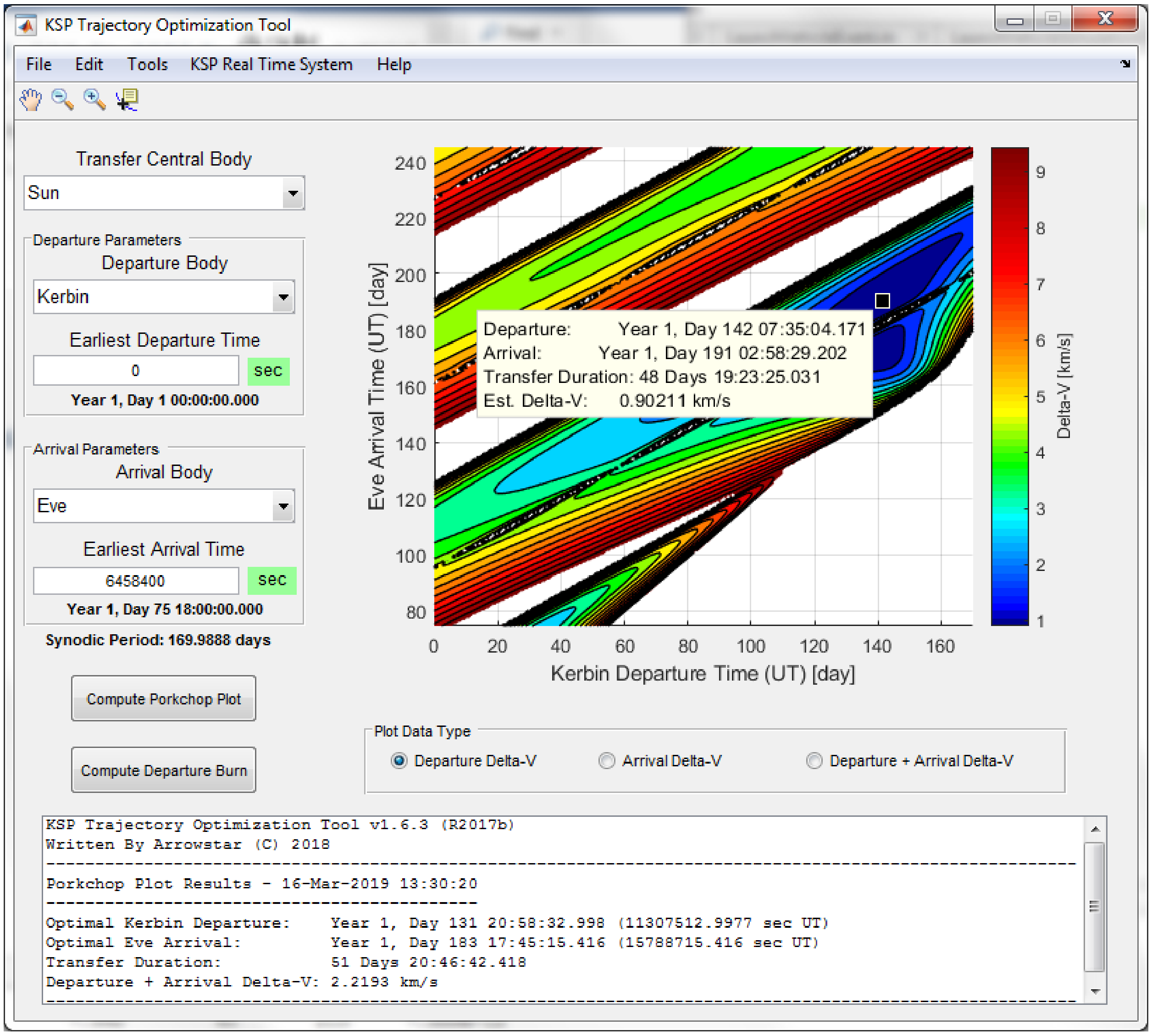
Table 1: Launch Site Coordinates

Latitude	-0.1025	degN
Longitude	285.4247	degE
Altitude	0.06841	km

2.1.2 Kerbin to Eve Trajectory

Using the porkchop plot utility within KSPTOT, we can generate a porkchop plot that characterizes the amount of effort required to depart Kerbin and arrive at Eve.

Set the earliest departure time to 0.0 seconds, the earliest arrival time to 6458400 seconds, and plot the “Departure Delta-V” only. Make sure Eve is selected as the arrival body. The result should resemble the figure below.


KSP Trajectory Optimization Tool
File
Edit
Tools
KSP Real Time System
Help
Transfer Central Body
Sun
Departure Parameters
Departure Body
Kerbin
Earliest Departure Time
0
sec
Year 1, Day 1 00:00:00.000
Arrival Parameters
Arrival Body
Eve
Earliest Arrival Time
6458400
sec
Year 1, Day 75 18:00:00.000
Synodic Period: 169.9888 days
Compute Porkchop Plot
Compute Departure Burn
Departure: Year 1, Day 142 07:35:04.171
Arrival: Year 1, Day 191 02:58:29.202
Transfer Duration: 48 Days 19:23:25.031
Est. Delta-V: 0.90211 km/s
Eve Arrival Time (UT) [day]
Kerbin Departure Time (UT) [day]
Delta-V [km/s]
Plot Data Type
Departure Delta-V
Arrival Delta-V
Departure + Arrival Delta-V

```
KSP Trajectory Optimization Tool v1.6.3 (R2017b)
Written By Arrowstar (C) 2018
--------------------------------------------------------------------------------
Porkchop Plot Results - 16-Mar-2019 13:30:20
---------------------------------------------
Optimal Kerbin Departure:    Year 1, Day 131 20:58:32.998 (11307512.9977 sec UT)
Optimal Eve Arrival:         Year 1, Day 183 17:45:15.416 (15788715.416 sec UT)
Transfer Duration:           51 Days 20:46:42.418
Departure + Arrival Delta-V: 2.2193 km/s
--------------------------------------------------------------------------------
```

Figure 1: Kerbin to Eve Porkchop Plot

Our plan will be to leave Kerbin somewhere around Year 1, Day 131 and arrive at Eve somewhere around Year 1, Day 183. The transfer duration is nearly 52 Earth days.

2.1.3 Kerbin Departure Orbit and Targets

Tap the Compute Departure button on the main KSPTOT UI after generating the above porkchop plot. Leaving the departure date as-is, right click the Eve Arrival Time textbox and select “Find Optimal Arrival Time for Input Departure Time.” This will optimize the arrival time “under the hood” in order to take out any of the error resulting from using a porkchop plot with a coarse grid. The resultant arrival date should be above 15788715.4169 seconds UT. The Kerbin departure time is 11307512.9977 seconds UT. Leave the initial orbit SMA at 700 km.

Tap the “Compute Departure Burn” button. The Kerbin Departure Information window will appear with a plot and data to the right. Note the departure hyperbolic orbit’s inclination, RAAN, and argument of periapsis. Place theses three quantities into their respective areas within the “Enter Transfer & Initial Orbit Information” box and re-tap the “Compute Departure Button.”

The critical information we must acquire from this screen is the departure orbit, the departure outbound hyperbolic velocity vector, and the departure delta-v. Your information should be very similar to the following.

Table 2: Approximate Kerbin Departure Orbit

	Value	Unit
Semi-major Axis	-3166.2538	km
Eccentricity	1.221081435	
Inclination	30.6159	deg
Right Ascension of AN	264.5773	deg
Argument of Periapse	151.2229	deg

Table 3: Kerbin Departure Targets

	Value	Unit
Outbound Hyperbolic Velocity Vector Right Ascension	-155.6607	deg
Outbound Hyperbolic Velocity Vector Declination	-27.1902	deg
Outbound Hyperbolic Velocity Vector Magnitude	1.056119094	km/s

Table 4: Approximate Kerbin Departure Delta-v

	Value	Unit
Total Delta-V	1.10135	km/s
Prograde Delta-V	1101.34786	m/s
Orbit Normal Delta-V	-0.00433	m/s
Radial Delta-V	1.15314	m/s

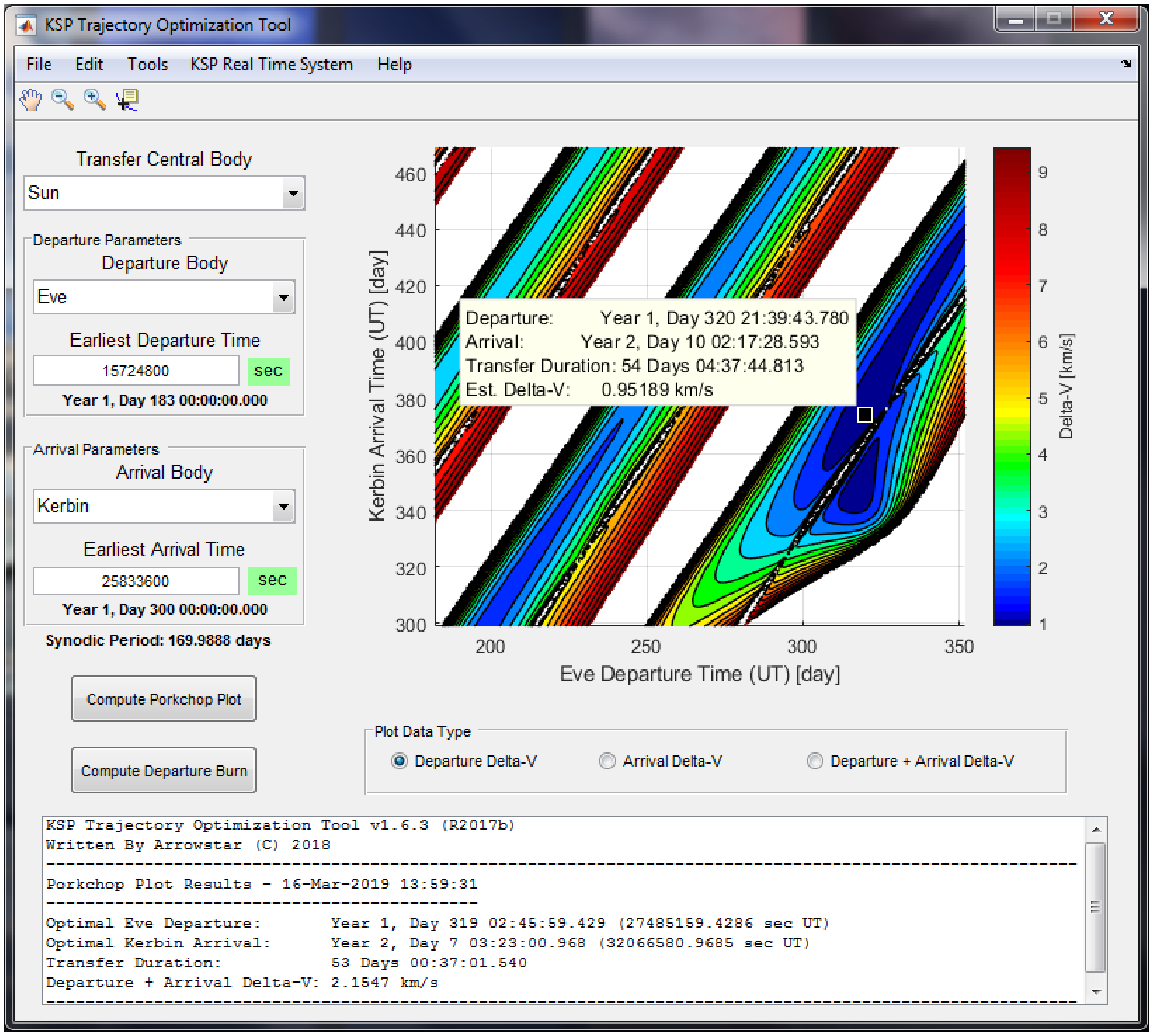
Note that only the total delta-v magnitude is really important for our future analysis. The components are only shown for completeness. However, as our departure profile will be a direct ascent to the Kerbin departure targets, we only look for the approximate delta-v here to estimate how much additional delta-v is required beyond what is required to merely make orbit.

2.1.4 Eve to Kerbin Trajectory

Returning to the porkchop plot utility, set the departure body to Eve and the earliest departure time to Year 1, Day 183 (15724800 seconds UT). This is approximately the time we are arriving at Eve and we do not want to depart sooner than we arrive.

Set the arrival body to Kerbin and the earliest arrival time to Year 1, Day 300 (25833600 sec UT).

Tap the Compute Porkchop Plot button. Your resultant porkchop plot should look like the one shown in the figure below.


KSP Trajectory Optimization Tool
File Edit Tools KSP Real Time System Help
460
440
420
400
380
360
340
320
300
Kerbin Arrival Time (UT) [day]
200
250
300
350
Eve Departure Time (UT) [day]
9
8
7
6
5
4
3
2
1
Delta-V [km/s]
Compute Porkchop Plot
Compute Departure Burn

Transfer Central Body
Sun

Departure Parameters
Departure Body
Eve
Earliest Departure Time
15724800 sec
Year 1, Day 183 00:00:00.000

Arrival Parameters
Arrival Body
Kerbin
Earliest Arrival Time
25833600 sec
Year 1, Day 300 00:00:00.000

Synodic Period: 169.9888 days

Departure: Year 1, Day 320 21:39:43.780
Arrival: Year 2, Day 10 02:17:28.593
Transfer Duration: 54 Days 04:37:44.813
Est. Delta-V: 0.95189 km/s

Plot Data Type
[✓ Departure Delta-V] Arrival Delta-V Departure + Arrival Delta-V

```
KSP Trajectory Optimization Tool v1.6.3 (R2017b)
Written By Arrowstar (C) 2018
-----------------------------------------------------------------------------------------
Porkchop Plot Results - 16-Mar-2019 13:59:31
-------------------------------------------
Optimal Eve Departure:        Year 1, Day 319 02:45:59.429 (27485159.4286 sec UT)
Optimal Kerbin Arrival:       Year 2, Day 7 03:23:00.968 (32066580.9685 sec UT)
Transfer Duration:            53 Days 00:37:01.540
Departure + Arrival Delta-V:  2.1547 km/s
-----------------------------------------------------------------------------------------
```

Figure 2: Eve to Kerbin Porkchop Plot

Our plan will be to leave Eve somewhere around Year 1, Day 319 and arrive back on Kerbin around Year 2, Day 7. Our total stay on Eve is about 136 Earth days. The return to Kerbin will take about 53 days.

2.1.5 Eve Landing Site

We need to select the location on Eve that we wish to land upon. In orbit to make departure easier, we want somewhere reasonable high. This will keep us out of the lowest and thickest part of the atmosphere. Let's look at a topographical map of Eve.

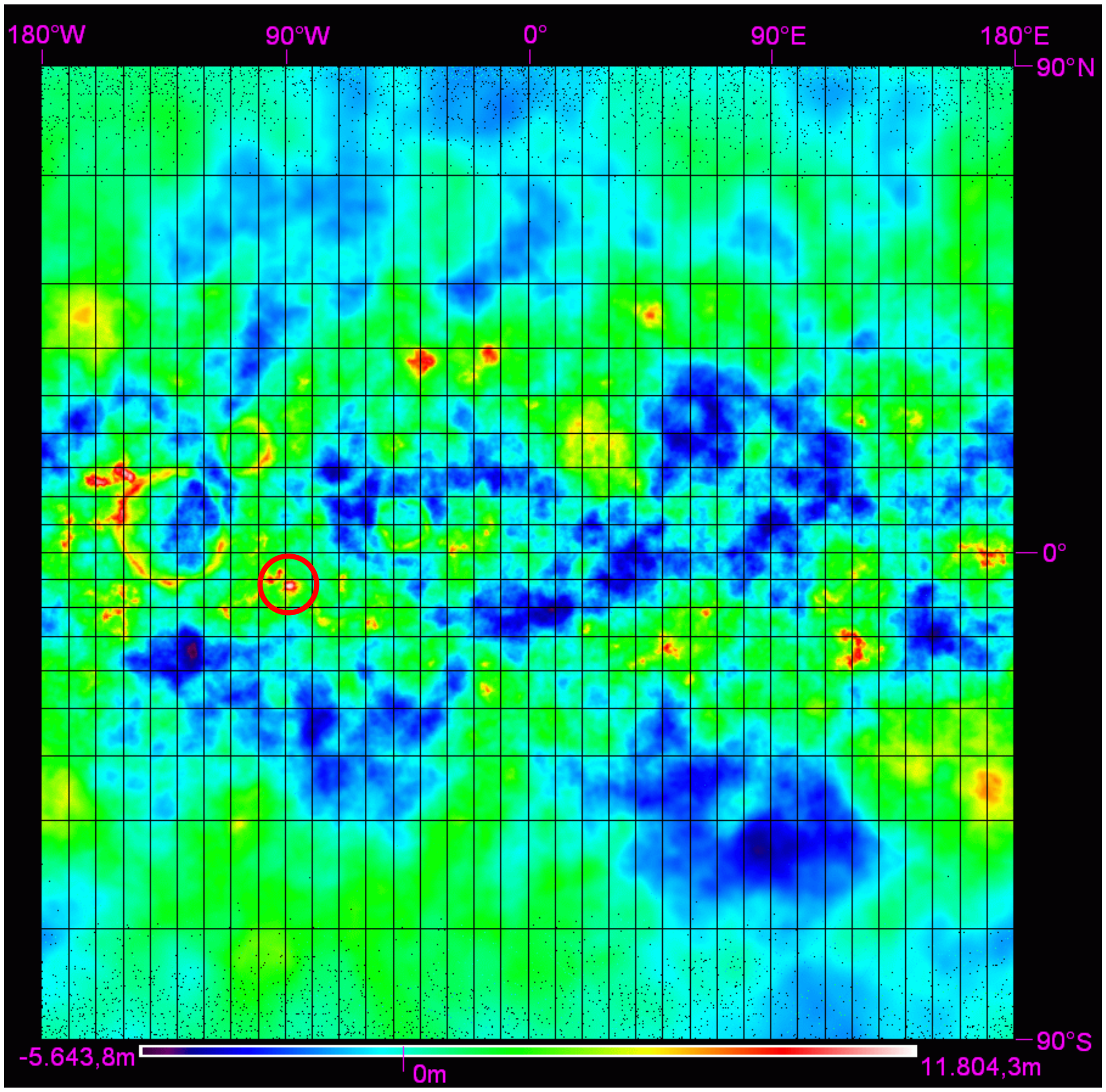


Figure 3: Topographical Map of Eve
Credit: KSP Wiki (https://wiki.kerbalspaceprogram.com/wiki/File:Eve_map_800.gif)

There is a mountain peak at a latitude of about -12 degN and a longitude of -88 degE. The elevation there appears to be about 11.8 km above sea level. We will use these values for our landing target.

Table 5: Eve Landing Site Coordinates

Latitude	-12.0	degN
Longitude	-88.0 / 272.0	degE
Altitude	11.8	km

2.1.6 Eve Departure Orbit and Targets

Repeat the process described in section 2.1.3 to find the Eve departure orbit and targets. Use an initial orbit semi-major axis of 800 km. After re-inserting the inclination, RAAN, and argument of periapsis as last time, your resultant departure information should be similar to the following.

Table 6: Approximate Eve Departure Orbit

	Value	Unit
Semi-major Axis	-8592.3301	km
Eccentricity	1.093106262	
Inclination	25.1057	deg
Right Ascension of AN	294.9042	deg
Argument of Periapsis	149.7824	deg

Table 7: Eve Departure Targets

	Value	Unit
Outbound Hyperbolic Velocity Vector Right Ascension	-116.3921	deg
Outbound Hyperbolic Velocity Vector Declination	-20.0852	deg
Outbound Hyperbolic Velocity Vector Magnitude	0.975217605	km/s

Table 8: Approximate Eve Departure Delta-v

	Value	Unit
Total Delta-V	1.42785	km/s
Prograde Delta-V	1427.85236	m/s
Orbit Normal Delta-V	-0.00490	m/s
Radial Delta-V	2.70123	m/s

2.1.7 Kerbin Splashdown Site

It is desirable for our Eve sample to be returned as quickly as possible back to the Kerbal Space Center, located at the coordinates shown in Table 1. Let us also assume we want a water landing for ease of recovery and landing. Given this, where should we land?

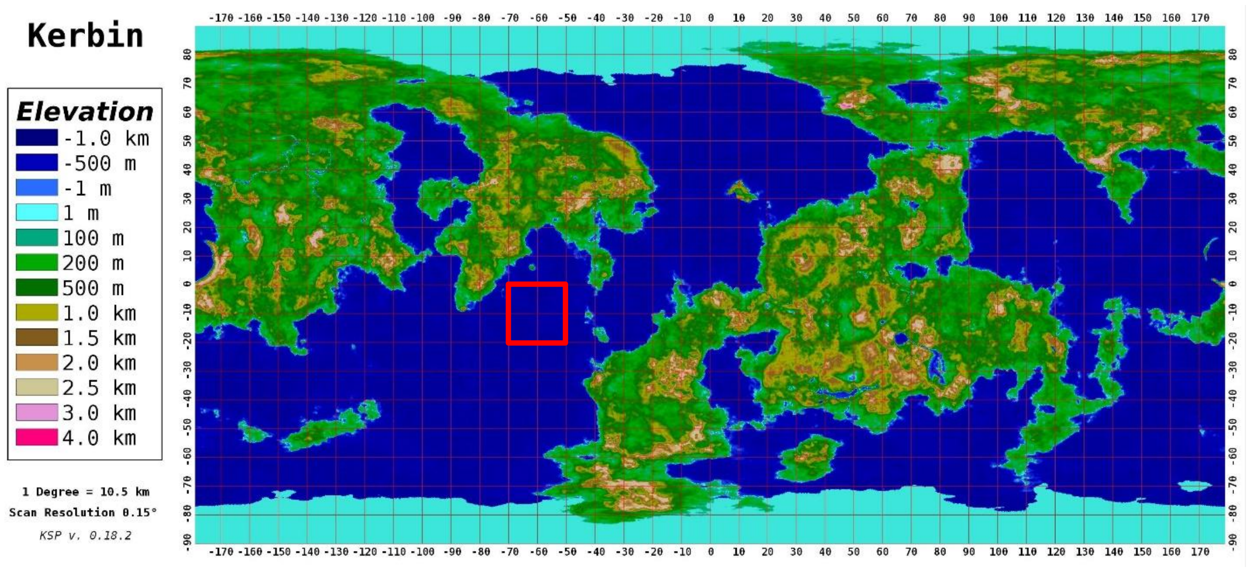


Figure 4: Map of Kerbin (Credit: Zeroignite)
Source: KSP Wiki (https://wiki.kerbalspaceprogram.com/images/a/a9/Kerbin_heightmap.jpg)

A red square has been marked on the map in Figure 4 that shows a suitable area. The coordinates that correspond with this region are:

Table 9: Kerbin Return Region Coordinates

Minimum Longitude	290.0	deg
Maximum Longitude	310.0	deg
Minimum Latitude	-20.0	deg
Maximum Latitude	0.0	deg

2.2 Delta-v Budget

2.2.1 Ascent Delta-v Estimates

Based on our previous analysis, we already know roughly what the delta-v required to leave the spheres of influence of Kerbin and Eve will be. However, we do not know what the Kerbin and Eve ascent delta-v will be. We can use a handy delta-v map, similar to the one from https://13375.de/KSPDeltaVMap/ in order to obtain these values.

Using this website, we see 3400 m/s required to achieve low Kerbin orbit from the surface of Kerbin and 8000 m/s to achieve low Eve orbit from the surface of Eve.

2.2.2 Delta-v Budget Summary

Let's summarize our delta-v budget, keeping in mind that we will use direct descent into the atmospheres of Kerbin and Eve to minimize the amount of total delta-v required for the mission. Parachutes will be used to slow down instead.

We will apply a safety factor to each of the base delta-v values in order to have some margin for error when we actually execute the mission. Our standard safety factor will be 15% of the base delta-v. The Eve ascent delta-v safety factor will be 10% because we are landing at a high elevation and the delta-v map provided ascent delta-v for a sea level launch.

Table 10: Mission Delta-v Estimate

Event	Base Delta-v [km/s]		Safety Factor		Total Delta-v [km/s]
Kerbin Ascent	3.40	×	1.15	=	3.910
Kerbin Departure	1.10	×	1.15	=	1.265
Eve Entry, Descent, and Landing	0.00	×	1.15	=	0.000
Eve Ascent	8.00	×	1.10	=	8.800
Eve Departure	1.43	×	1.15	=	1.645
Kerbin Entry, Descent, and Landing	0.00	×	1.15	=	0.000
Sum					**15.620**

2.2.3 Mission Phase Dry Mass and Propellent Mass Estimates

When we use Launch Vehicle Designer, we will need to provide estimates for the initial mass of the various stages. This includes both the dry mass and the propellant mass.

It is outside the scope of this tutorial to look at optimal stage sizing and optimal delta-v split per stage. However, there is a very rough estimation we can perform that merely uses the rocket equation to approximately determine how much mass each stage will be. Reworking the rocket equation to solve for initial mass prior to a delta-v maneuver, we get:

$$m_0 = m_1 e^{\left(\frac{\Delta V}{g_0 I_{sp}}\right)}$$

Using this expression and working backwards from the Kerbin Entry, Descent, and Landing (EDL), we can approximate roughly how big our rocket will be. The Excel file that was used to generate this table is included with this tutorial for your perusal as needed. For now, it is sufficient to simply accept the values here until you actually go to fly the mission in KSP.

The values in the table assume that 500 kg of Eve rocks and associated sample return vehicle arrive at Kerbin.

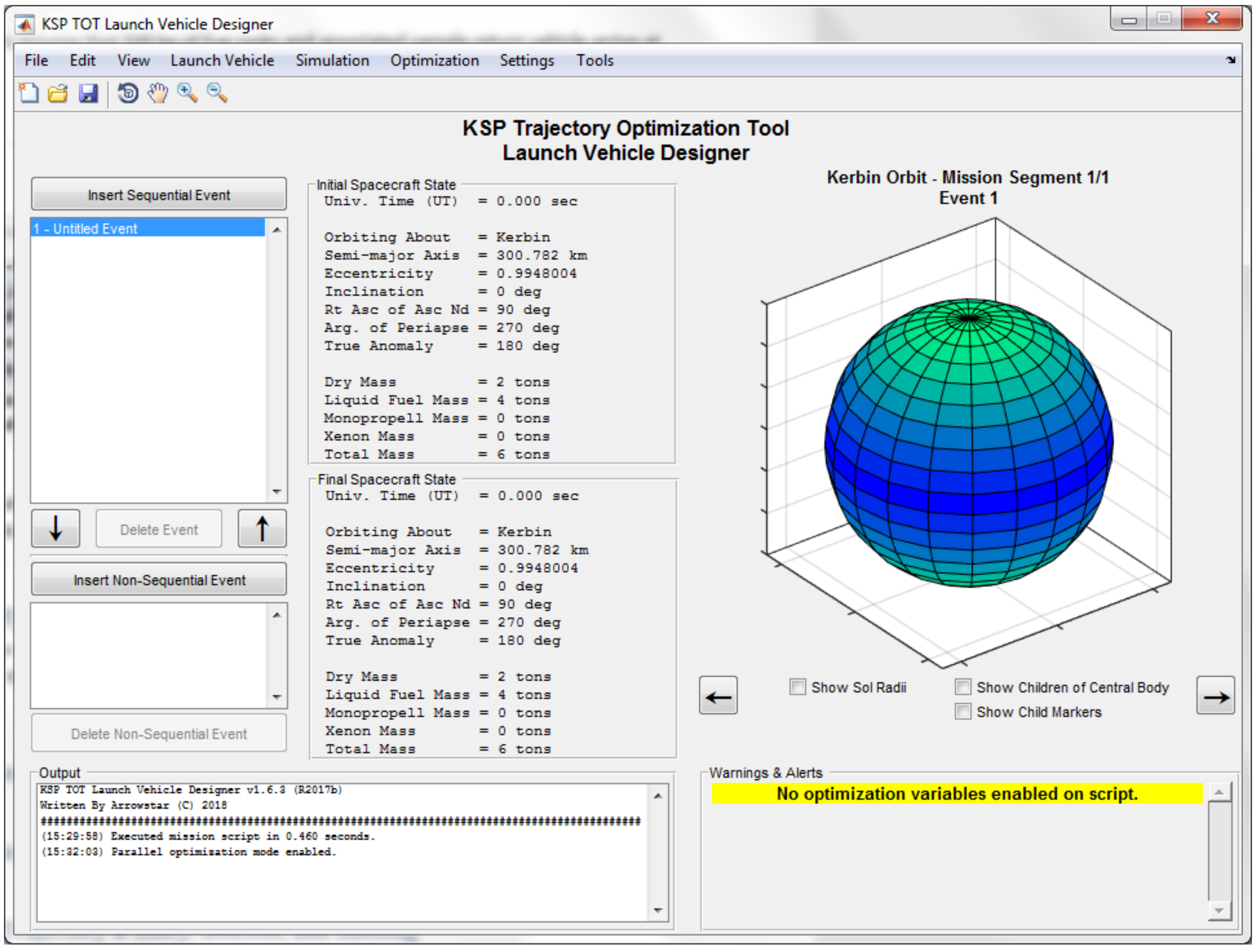
Table 11: Mission Phase Mass Estimates

Phase	Stage #	Delta-v [m/s]	Dry Mass [mT]	Prop Mass [mT]	Total Mass [mT]
Payload	▨	▨	0.5000	0.0000	0.5000
Phase 2	3	**4237.78**	0.6077	2.7685	3.3763
	2	**3226.36**	3.3430	15.2292	18.5723
	1	**2980.86**	18.1399	77.3334	95.4733
Phase 1	3	**2375.46**	41.3759	165.5036	206.8795
	2	**1462.83**	62.3454	249.3816	311.7270
	1	**1336.71**	109.5491	438.1963	547.7454

Remember that the values in this table are only coarse estimates designed to help us determine rough sizes for our vehicle. The work we do in Launch Vehicle Designer will undoubtably help us refine these numbers more.

3 High Fidelity Mission Planning with Launch Vehicle Designer

Having completed the early mission design phase, let's open LVD. Using the main KSPTOT GUI, open LVD with the Tools -> Mission Planning -> Launch Vehicle Designer menu option. Your user interface should look like below.


KSP TOT Launch Vehicle Designer
File Edit View Launch Vehicle Simulation Optimization Settings Tools
KSP Trajectory Optimization Tool
Launch Vehicle Designer
Insert Sequential Event
1 - Untitled Event
Delete Event
Insert Non-Sequential Event
Delete Non-Sequential Event
Kerbin Orbit - Mission Segment 1/1
Event 1
Show Sol Radii
Show Children of Central Body
Show Child Markers

Initial Spacecraft State

```
Univ. Time (UT)    = 0.000 sec

Orbiting About     = Kerbin
Semi-major Axis    = 300.782 km
Eccentricity       = 0.9948004
Inclination        = 0 deg
Rt Asc of Asc Nd   = 90 deg
Arg. of Periapse   = 270 deg
True Anomaly       = 180 deg

Dry Mass           = 2 tons
Liquid Fuel Mass   = 4 tons
Monopropell Mass   = 0 tons
Xenon Mass         = 0 tons
Total Mass         = 6 tons
```

Final Spacecraft State

```
Univ. Time (UT)    = 0.000 sec

Orbiting About     = Kerbin
Semi-major Axis    = 300.782 km
Eccentricity       = 0.9948004
Inclination        = 0 deg
Rt Asc of Asc Nd   = 90 deg
Arg. of Periapse   = 270 deg
True Anomaly       = 180 deg

Dry Mass           = 2 tons
Liquid Fuel Mass   = 4 tons
Monopropell Mass   = 0 tons
Xenon Mass         = 0 tons
Total Mass         = 6 tons
```

Output

```
KSP TOT Launch Vehicle Designer v1.6.3 (R2017b)
Written By Arrowstar (C) 2018
##########################################################################################
(15:29:58) Executed mission script in 0.460 seconds.
(15:32:03) Parallel optimization mode enabled.
```

Warnings & Alerts

No optimization variables enabled on script.

Figure 5: Clean LVD User Interface

We can now move on to setting up our mission.

3.1 Vehicle "Construction" and "Pad Rollout"

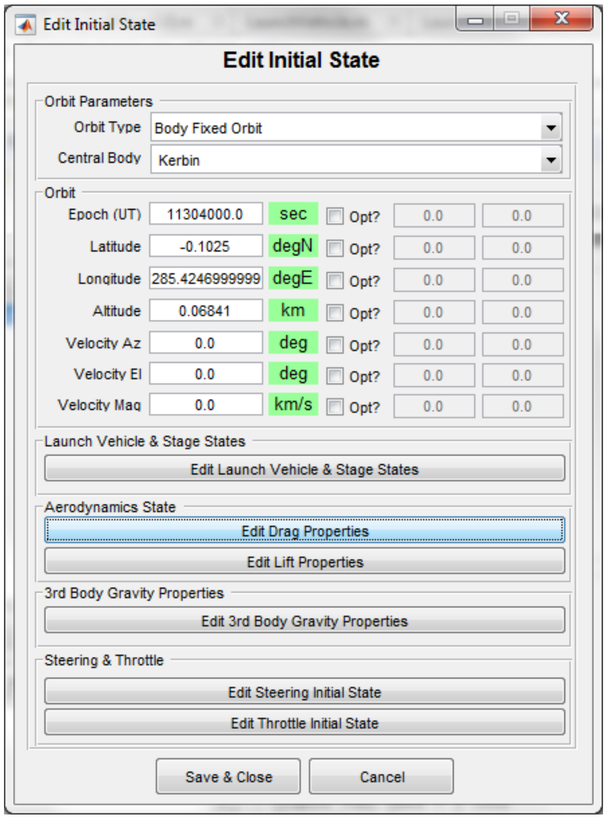
In this section, we will use LVD to define the parameters of our Eve sample return vehicle and define the initial state of that vehicle.

3.1.1 Setting Up the Initial State (Part 1)

The first thing we must do is set up our initial state. We will assume that we are stationary on the launch pad of KSC at the coordinates we tabulated in Table 1: Launch Site Coordinates. Use the Launch Vehicle -> Edit Initial State menu to do so.

In the dialog box that appears, edit the latitude, longitude, and altitude to what is shown in Table 1. Make sure that the orbit type is "Body Fixed" and the central body is Kerbin.

In the Epoch field, enter 11304000.0 seconds, corresponding to Year 1, Day 131 20:00:00. See Figure 1. Your initial state user interface should look like the following at this point.


Edit Initial State

Edit Initial State

Orbit Parameters

Orbit Type	Body Fixed Orbit
Central Body	Kerbin

Orbit

Epoch (UT)	11304000.0	sec	☐ Opt?	0.0	0.0
Latitude	-0.1025	degN	☐ Opt?	0.0	0.0
Longitude	285.4246999999	degE	☐ Opt?	0.0	0.0
Altitude	0.06841	km	☐ Opt?	0.0	0.0
Velocity Az	0.0	deg	☐ Opt?	0.0	0.0
Velocity El	0.0	deg	☐ Opt?	0.0	0.0
Velocity Mag	0.0	km/s	☐ Opt?	0.0	0.0

Launch Vehicle & Stage States

Edit Launch Vehicle & Stage States

Aerodynamics State

Edit Drag Properties

Edit Lift Properties

3rd Body Gravity Properties

Edit 3rd Body Gravity Properties

Steering & Throttle

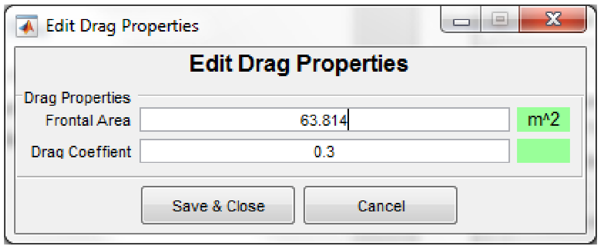
Edit Steering Initial State

Edit Throttle Initial State

Save & Close | Cancel

Figure 6: Initial State User Interface w/ Launch Site Location and Time

Now let's edit some other parameters. Open the "Edit Drag Parameters" box and change the drag coefficient to 0.3 and the frontal area to 63.814 m^2. This corresponds to a 3.75-meter tank with four 1.25 m^2 boosters strapped next to it with a reasonable coefficient of drag for a rocket. Save and close the drag dialog box.


Edit Drag Properties

Edit Drag Properties

Drag Properties

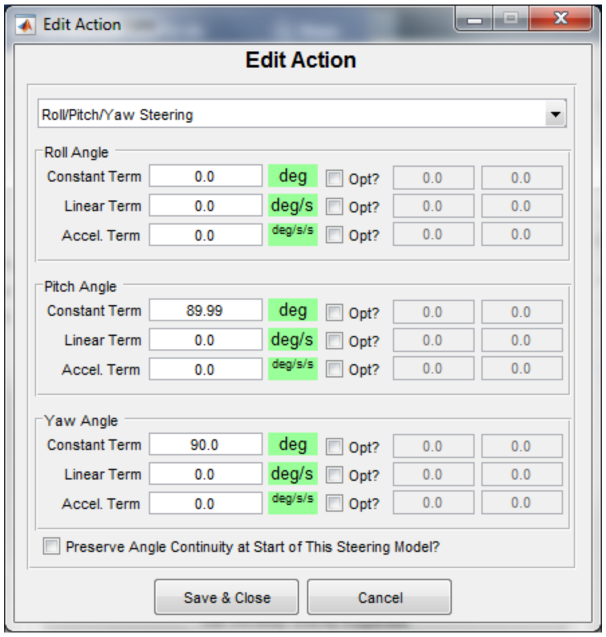
Frontal Area	63.814	m^2
Drag Coeffient	0.3	

Save & Close | Cancel

Figure 7: Initial State Drag Dialog Box

The KSPTOT wiki features a page that describes how one might obtain the drag frontal area through graphical techniques. While not the only way to obtain the information (simple geometry works well for cylindrical rockets), it is one of the more novel approaches to solving the problem. Here is a link to the wiki page: Getting Vehicle Area for LVD Drag Force Calculations

Next open the Steering Initial State dialog box. Make sure that Roll/Pitch/Yaw Steering is shown at the top of the dialog box in the menu. Set the Pitch Angle Constant Term to 89.99 deg and the Yaw Angle Constant Term to 90 deg. This points the rocket straight up initially and oriented to fly eastward. Save and close the dialog box.



Roll Angle

Constant Term	0.0	deg	☐ Opt?	0.0	0.0
Linear Term	0.0	deg/s	☐ Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	☐ Opt?	0.0	0.0

Pitch Angle

Constant Term	89.99	deg	☐ Opt?	0.0	0.0
Linear Term	0.0	deg/s	☐ Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	☐ Opt?	0.0	0.0

Yaw Angle

Constant Term	90.0	deg	☐ Opt?	0.0	0.0
Linear Term	0.0	deg/s	☐ Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	☐ Opt?	0.0	0.0

☐ Preserve Angle Continuity at Start of This Steering Model?

Figure 8: Initial State Steering Model Dialog Box

There is a singularity at pitch angles of 90 degrees that can cause the R/P/Y steering model to not work as intended. If possible, avoid setting the pitch angle to exactly 90 degrees by offsetting it slightly as we have done here.

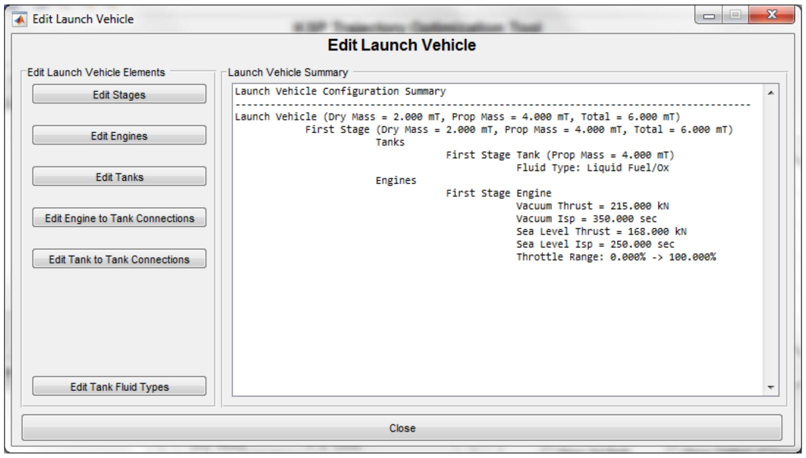
From here, you may now save and close the Edit Initial State dialog box.

3.1.2 Setting Up the Vehicle Model

Next, we need to put in our launch vehicle stage information. Some of this will come from Table 11, though our primary goal is simply to use that information as a guide to help us select realistic values.

In putting together this tutorial, I have created a vehicle model based loosely on parts in KSP. As we go through creating stages, tanks, and engines, some explanation will be provided to show roughly where certain mass, thrust, and specific impulse numbers came from.

Open up the vehicle editor by using the Launch Vehicle -> Edit Launch Vehicle menu.



```
Launch Vehicle Configuration Summary
--------------------------------------------------------------------------------
Launch Vehicle (Dry Mass = 2.000 mT, Prop Mass = 4.000 mT, Total = 6.000 mT)
          First Stage (Dry Mass = 2.000 mT, Prop Mass = 4.000 mT, Total = 6.000 mT)
                    Tanks
                              First Stage Tank (Prop Mass = 4.000 mT)
                                        Fluid Type: Liquid Fuel/Ox
                    Engines
                              First Stage Engine
                                        Vacuum Thrust = 215.000 kN
                                        Vacuum Isp = 350.000 sec
                                        Sea Level Thrust = 168.000 kN
                                        Sea Level Isp = 250.000 sec
                                        Throttle Range: 0.000% -> 100.000%
```

Figure 9: Clean Edit Launch Vehicle UI

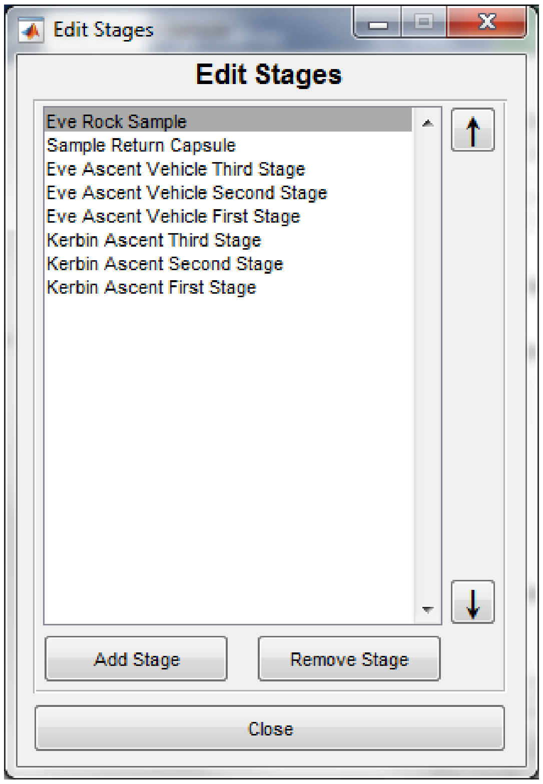
3.1.2.1 Creating the Vehicle Stages

Click the “Edit Stages” button to create the various stages of our vehicle. We will add the following stages, as shown in the table. To do so, click the “Add Stage” button. For each stage, enter the name and dry mass as shown. Order the stages in the order in the table using the up and down arrow buttons as needed. You may edit the pre-existing first stage by double clicking its entry in the list to create the first stage, if you’d like.

Table 12: Stage Parameters

Stage Name	**Dry Mass [mT]**	**Notes**
Eve Rock Sample	0.100	Represents the rocks we will pick up on the surface of Eve to return to Kerbin.
Sample Return Capsule	0.400	Represents the vehicle that will hold the Eve sample. Likely a HECS2 probe core and associated hardware.
Eve Ascent Vehicle Third Stage	0.875	An LV-909 engine, FL-T400 Fuel Tank, FL-T200 Fuel Tank.
Eve Ascent Vehicle Second Stage	3.250	RK-7 "Kodiak" engine, 4x FL-T800 Fuel Tank
Eve Ascent Vehicle First Stage	16.500	An RE-M3 Mailsail engine, Kerbodyne S3-14400 Tank, 2x FL-C1000 Fuel Tank
Kerbin Ascent Third Stage	30.500	An Kerbodyne KR-2L engine and Kerbodyne S4-256 Fuel Tank, Kerbodyne S4-164 Fuel Tank, 2x FL-C1000 Fuel Tank
Kerbin Ascent Second Stage	60.000	7x S3 KS-25 "Vector" Liquid Fuel Engine, 2x Kerbodyne S4-256 Fuel Tank
Kerbin Ascent First Stage	116.000	4x S3 KS-25x4 "Mammoth" Liquid Fuel Engine, Kerbodyne S4-512 Fuel Tank, Kerbodyne S4-128 Fuel Tank, Kerbodyne S4-128 Fuel Tank.

When you have completed adding all of the stages, your Stages dialog box should look like the following.



Eve Rock Sample
Sample Return Capsule
Eve Ascent Vehicle Third Stage
Eve Ascent Vehicle Second Stage
Eve Ascent Vehicle First Stage
Kerbin Ascent Third Stage
Kerbin Ascent Second Stage
Kerbin Ascent First Stage

Figure 10: Completed Edit Stages Dialog Box

Tap the Close button to return the Edit Launch Vehicle dialog box. Notice how the information text on the right side of the box populates with information regarding the vehicle.

3.1.2.2 Creating the Vehicle Engines

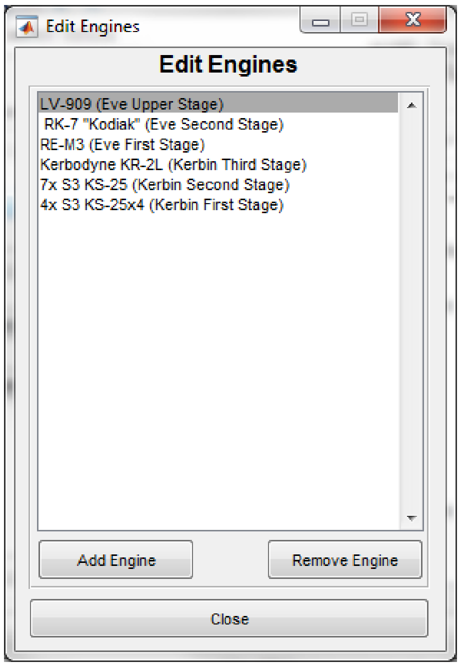
Tap the Edit Engines button to open up the user interface to add and edit engines. To add an engine, tap the Add Engine button. Do this for each engine in Table 13 below, populating the fields of the dialog box that appears with the data in the table.

Table 13: Engine Parameters

Name	Stage	Vac Thrust	Vac I_{sp}	S.L. Thrust	S.L. I_{sp}	Min Throttle	Max Throttle
LV-909 (Eve Third Stage)	Eve Ascent Vehicle Third Stage	60.0	345.0	14.78	85.0	0.0.	100.0
RK-7 "Kodiak" (Eve Second Stage)	Eve Ascent Vehicle Second Stage	442.0	300.0	419.9	285.0	0.0	100.0
RE-M3 (Eve First Stage)	Eve Ascent Vehicle First Stage	2550.0	310.0	2344.351	285.0	0.0	100.0
Kerbodyne KR-2L (Kerbin Third Stage)	Kerbin Ascent Third Stage	2000.0	340.0	1205.88	205.0	0.0	100.0
7x S3 KS-25 (Kerbin Second Stage)	Kerbin Ascent Second Stage	7000.0	315.0	6555.57	295.0	0.0	100.0
4x S3 KS-25x4 (Kerbin First Stage)	Kerbin Ascent First Stage	16000.0	315.0	14984.12	295.0	0.0	100.0

(NB: The Eve first and second stage engines have had their thrust values multiplied by 1.7 from the game default for the listed parts. This is due to an error on the author's part found late in writing this tutorial that resulted in insufficient thrust during Eve ascent. Because of the difficulty associated with reworking the entire Kerbin ascent trajectory once again, this small modification was implemented instead.)

When you have completed this, your Engines dialog box should look like the following.



LV-909 (Eve Upper Stage)
RK-7 "Kodiak" (Eve Second Stage)
RE-M3 (Eve First Stage)
Kerbodyne KR-2L (Kerbin Third Stage)
7x S3 KS-25 (Kerbin Second Stage)
4x S3 KS-25x4 (Kerbin First Stage)

Figure 11: Completed Engine Dialog Box

Tap the Close button to return to the Edit Launch Vehicle dialog box.

3.1.2.3 Creating the Vehicle Tanks

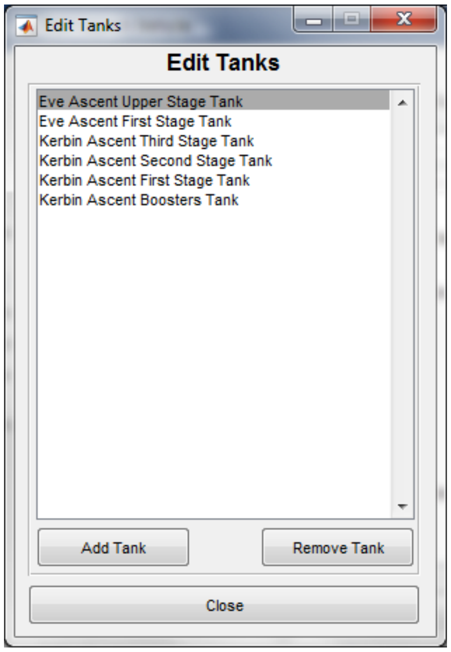
Tap the Edit Tanks button to open up the user interface to add and edit fuel tanks. To add a tank, tap the Add Tank button. Do this for each tank in Table 14 below, populating the fields of the dialog box that appears with the data in the table.

All tanks should use the “Liquid Fuel/Ox” fluid type.

Table 14: Tank Parameters

Name	Stage	Mass [mT]
Eve Ascent Third Stage Tank	Eve Ascent Vehicle Third Stage	3.0
Eve Ascent Second Stage Tank	Eve Ascent Vehicle Second Stage	16.0
Eve Ascent First Stage Tank	Eve Ascent Vehicle First Stage	84.0
Kerbin Ascent Third Stage Tank	Kerbin Ascent Third Stage	172.0
Kerbin Ascent Second Stage Tank	Kerbin Ascent Second Stage	256.0
Kerbin Ascent First Stage Tank	Kerbin Ascent First Stage	448.0

When you have completed this, your Tanks dialog box should look like the following.



Eve Ascent Upper Stage Tank
Eve Ascent First Stage Tank
Kerbin Ascent Third Stage Tank
Kerbin Ascent Second Stage Tank
Kerbin Ascent First Stage Tank
Kerbin Ascent Boosters Tank

Figure 12: Completed Edit Tanks Dialog Box

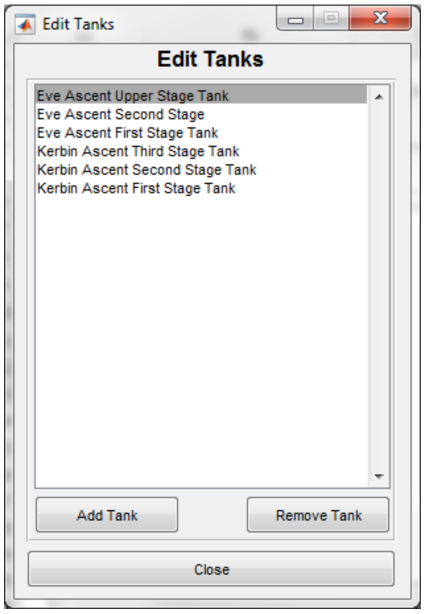
Tap the Close button to return to the Edit Launch Vehicle dialog box.

3.1.2.4 Setting Up the Engine to Tank Connections

In Launch Vehicle Designer, it is not enough to simply specify a set of fuel tanks and engines. You must also connect engines to tanks by creating "Engine to Tank Connections." You can think of these like the feed lines that flow fuel from a tank to a connected engine. These connections are very flexible. Multiple tanks may connect to a single engine and multiple engines may feed from a single tank.

Tap the "Edit Engine to Tank Connections" button to open the "Edit Engine to Tank Connections" dialog box. For each engine in Table 13, click the "Add Connection" button, select that engine, and select the corresponding tank. For example, for the "LV-909 (Eve Upper Stage)" engine, select the "Eve Ascent Upper Stage Tank" tank.

When you have done this for each engine, your dialog box should look like the following.



Eve Ascent Upper Stage Tank
Eve Ascent Second Stage
Eve Ascent First Stage Tank
Kerbin Ascent Third Stage Tank
Kerbin Ascent Second Stage Tank
Kerbin Ascent First Stage Tank

Figure 13: Completed Engine to Tank Connections Dialog Box

Tap the Close button to return to the Edit Launch Vehicle dialog box.

3.1.2.5 Vehicle Configuration Review

When you have completed setting up the launch vehicle, your Launch Vehicle Summary text area should look as follows. If it does not, correct the element of the vehicle that has errors.

```
Launch Vehicle Configuration Summary
-----------------------------------------------------------------------------------------
Launch Vehicle (Dry Mass = 227.625 mT, Prop Mass = 948.413 mT, Total = 1176.038 mT)
         Eve Rock Sample (Dry Mass = 0.100 mT, Prop Mass = 0.000 mT, Total = 0.100 mT)
                  Tanks
                  Engines
         Sample Return Capsule (Dry Mass = 0.400 mT, Prop Mass = 0.000 mT, Total = 0.400 mT)
                  Tanks
                  Engines
         Eve Ascent Vehicle Third Stage (Dry Mass = 0.875 mT, Prop Mass = 2.768 mT, Total = 3.643 mT)
                  Tanks
                           Eve Ascent Upper Stage Tank (Prop Mass = 2.768 mT)
                                    Fluid Type: Liquid Fuel/Ox
                  Engines
                           LV-909 (Eve Upper Stage)
                                    Vacuum Thrust = 60.000 kN
                                    Vacuum Isp = 345.000 sec
                                    Sea Level Thrust = 14.780 kN
                                    Sea Level Isp = 85.000 sec
                                    Throttle Range: 0.000% -> 100.000%
         Eve Ascent Vehicle Second Stage (Dry Mass = 3.250 mT, Prop Mass = 15.229 mT, Total = 18.479 mT)
                  Tanks
                           Eve Ascent Second Stage (Prop Mass = 15.229 mT)
                                    Fluid Type: Liquid Fuel/Ox
                  Engines
                            RK-7 "Kodiak" (Eve Second Stage)
                                    Vacuum Thrust = 442.000 kN
                                    Vacuum Isp = 300.000 sec
                                    Sea Level Thrust = 419.900 kN
                                    Sea Level Isp = 285.000 sec
                                    Throttle Range: 0.000% -> 100.000%
         Eve Ascent Vehicle First Stage (Dry Mass = 16.500 mT, Prop Mass = 77.333 mT, Total = 93.833 mT)
                  Tanks
                           Eve Ascent First Stage Tank (Prop Mass = 77.333 mT)
```

```
                                        Fluid Type: Liquid Fuel/Ox
                    Engines
                            RE-M3 (Eve First Stage)
                                        Vacuum Thrust = 2550.000 kN
                                        Vacuum Isp = 310.000 sec
                                        Sea Level Thrust = 2344.351 kN
                                        Sea Level Isp = 285.000 sec
                                        Throttle Range: 0.000% -> 100.000%
          Kerbin Ascent Third Stage (Dry Mass = 30.500 mT, Prop Mass = 165.504 mT, Total = 196.004 mT)
                    Tanks
                            Kerbin Ascent Third Stage Tank (Prop Mass = 165.504 mT)
                                        Fluid Type: Liquid Fuel/Ox
                    Engines
                            Kerbodyne KR-2L (Kerbin Third Stage)
                                        Vacuum Thrust = 2000.000 kN
                                        Vacuum Isp = 340.000 sec
                                        Sea Level Thrust = 1205.880 kN
                                        Sea Level Isp = 205.000 sec
                                        Throttle Range: 0.000% -> 100.000%
          Kerbin Ascent Second Stage (Dry Mass = 60.000 mT, Prop Mass = 249.382 mT, Total = 309.382 mT)
                    Tanks
                            Kerbin Ascent Second Stage Tank (Prop Mass = 249.382 mT)
                                        Fluid Type: Liquid Fuel/Ox
                    Engines
                            7x S3 KS-25 (Kerbin Second Stage)
                                        Vacuum Thrust = 7000.000 kN
                                        Vacuum Isp = 315.000 sec
                                        Sea Level Thrust = 6555.570 kN
                                        Sea Level Isp = 295.000 sec
                                        Throttle Range: 0.000% -> 100.000%
          Kerbin Ascent First Stage (Dry Mass = 116.000 mT, Prop Mass = 438.196 mT, Total = 554.196 mT)
                    Tanks
                            Kerbin Ascent First Stage Tank (Prop Mass = 438.196 mT)
                                        Fluid Type: Liquid Fuel/Ox
                    Engines
                            4x S3 KS-25x4 (Kerbin First Stage)
                                        Vacuum Thrust = 16000.000 kN
                                        Vacuum Isp = 315.000 sec
                                        Sea Level Thrust = 14984.120 kN
                                        Sea Level Isp = 295.000 sec
                                        Throttle Range: 0.000% -> 100.000%
```

3.1.3 Setting Up the Vehicle Initial State (Part 2)

3.1.3.1 Set Up the Launch Vehicle and Stage States

Now that we've set up the vehicle itself, we need to return to the initial state and set up a few more parameters. Open up the Edit Initial State dialog box once more (Launch Vehicle -> Edit Initial State).

Tap the Edit Launch Vehicle & Stage States button. This brings up a dialog that allows you to edit the initial state of stages, engines, and various connections.

Starting with the Stage States, click through each stage and ensure that it is active by checking the "Stage is Active?" check box with the stage selected. You may double click on a stage to toggle its state. The exception to our setup is the "Eve Rock Sample" stage, which should be **inactive** initially. We will activate it later, on Eve. This stage's checkbox should be unchecked.

When you are done, all stage checkboxes should be checked except the "Eve Rock Sample" stage. This represents the full vehicle sitting on the launch pad, waiting to lift off. (Except the rock sample, of course, which is still on Eve!)

An active stage has the following effects on the complete vehicle:

- The stage's dry mass is added to the total vehicle dry mass.
- Any engines on the stage may product thrust, if the engine itself is active.
- Any tanks on the stage may flow propellent into or out of that tank.

Deactivating a stage removes these behaviors from the complete vehicle.

Next, move on to the Engine States. Starting with the first engine, check to make sure all engine states are **unchecked**. This represents the engines not having been started yet.

An active engine has the following effects on the complete vehicle:

- The engine will draw propellent from any tanks that it is connected to.
- The engine will produce thrust according its propulsion model.

Note that rockets in LVD only have one throttle channel (just like in stock KSP), so if you want to have your throttle greater than 0.0% but do not want an engine to produce thrust, you must deactivate it.

Move on to the Engine to Tank Connection States. Starting with the first connection, check to make sure that all connection states are **checked**.

An active engine to tank connection has the following effects on the complete vehicle:

- A connected engine will be able to draw propellant from the connected tank so long as there is propellent in the tank to draw out.

Engines will only draw thrust from tanks that have active connections to that engine and that have propellent in them. If you have multiple tanks connected to an engine but want to only draw propellent from a subset of those tanks, deactivate the corresponding engine to tank connection.

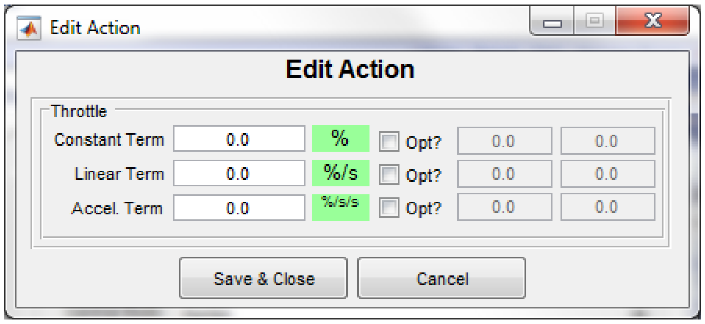
Finally, check the box labeled "Hold Down Clamps are Enabled?". This enables the modeling of rocket hold down clamps. When hold down clamps are active, the rocket will be fixed to its location relative to the surface of the body it is currently on or orbiting, and will rotate with the surface of that body.

When you are done, tap the Save and Close button to return to the Edit Initial State dialog box.

3.1.3.2 Set Up the Throttle Initial State

Click the "Edit Throttle Initial State" button to bring up the dialog box to edit the initial throttle state. Ensure that all entries in the throttle dialog box are 0.0. This corresponds to a throttle which is completely off and is not changing.

When you have done this, your dialog box should look like the following.



Throttle

Constant Term	0.0	%	Opt?	0.0	0.0
Linear Term	0.0	%/s	Opt?	0.0	0.0
Accel. Term	0.0	%/s/s	Opt?	0.0	0.0

Figure 14: Completed Throttle Initial State Dialog Box

When you are done, tap the Save and Close button to return to the Edit Initial State dialog box. Then tap the Save and Close button to close the Edit Initial State dialog box and return to the main LVD user interface.

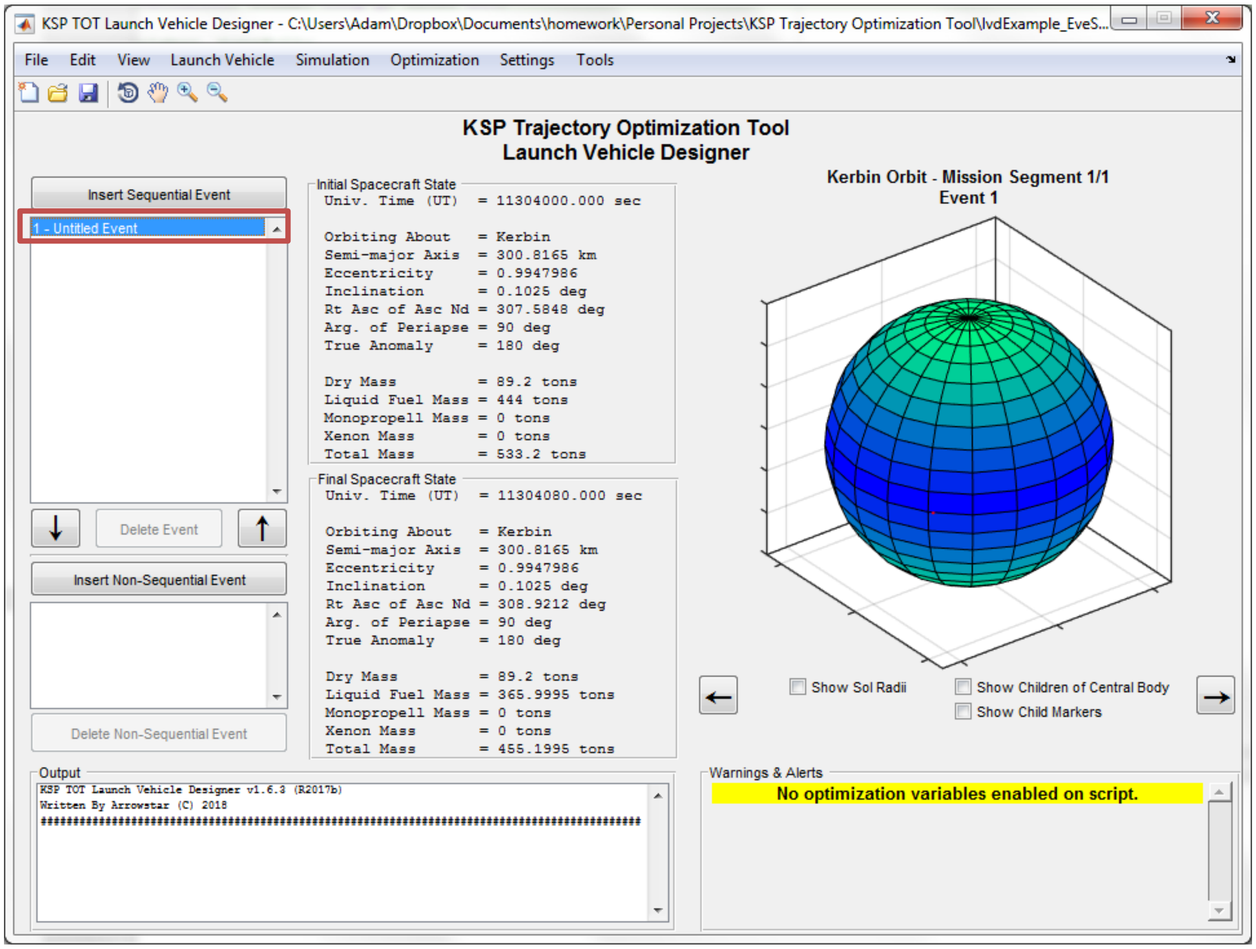
3.2 Kerbin Ascent and Departure Trajectories

3.2.1 Launch Vehicle Pre-Start to Tower Clear

Our goal for this phase of the mission is to create the events that will model starting up the first stage liquid rocket engine, throttling up to 100%, igniting the solid rocket boosters, and lifting off the launch pad to the point that the vehicle has cleared the tower.

3.2.1.1 Starting Up the First Stage Engines

To begin, notice that there is one event currently in the script. All LVD missions must have one event, so instead of deleting this one (which we cannot do), double click the "Untitled Event" in the upper event script list.


KSP TOT Launch Vehicle Designer - C:\Users\Adam\Dropbox\Documents\homework\Personal Projects\KSP Trajectory Optimization Tool\lvdExample_EveS...
File Edit View Launch Vehicle Simulation Optimization Settings Tools
KSP Trajectory Optimization Tool
Launch Vehicle Designer
Insert Sequential Event
1 - Untitled Event
Delete Event
Insert Non-Sequential Event
Delete Non-Sequential Event
Kerbin Orbit - Mission Segment 1/1
Event 1
Show Sol Radii
Show Children of Central Body
Show Child Markers

Initial Spacecraft State
```
Univ. Time (UT)    = 11304000.000 sec

Orbiting About     = Kerbin
Semi-major Axis    = 300.8165 km
Eccentricity       = 0.9947986
Inclination        = 0.1025 deg
Rt Asc of Asc Nd   = 307.5848 deg
Arg. of Periapse   = 90 deg
True Anomaly       = 180 deg

Dry Mass           = 89.2 tons
Liquid Fuel Mass   = 444 tons
Monopropell Mass   = 0 tons
Xenon Mass         = 0 tons
Total Mass         = 533.2 tons
```

Final Spacecraft State
```
Univ. Time (UT)    = 11304080.000 sec

Orbiting About     = Kerbin
Semi-major Axis    = 300.8165 km
Eccentricity       = 0.9947986
Inclination        = 0.1025 deg
Rt Asc of Asc Nd   = 308.9212 deg
Arg. of Periapse   = 90 deg
True Anomaly       = 180 deg

Dry Mass           = 89.2 tons
Liquid Fuel Mass   = 365.9995 tons
Monopropell Mass   = 0 tons
Xenon Mass         = 0 tons
Total Mass         = 455.1995 tons
```

Output
```
KSP TOT Launch Vehicle Designer v1.6.3 (R2017b)
Written By Arrowstar (C) 2018
##########################################################################################
```

Warnings & Alerts

No optimization variables enabled on script.

Figure 15: Double-click the "Untitled Event."

Doing so will bring up a dialog box that allows you to edit the event.

First, let's rename the event to "First Stage Throttle Up".

Next, click the "Set Termination Condition Parameters" button. This will bring up a dialog box that will allow you to specify when the event ends. For now, leave the termination type as "Event Duration" and set the Event Duration to 0.0 seconds.

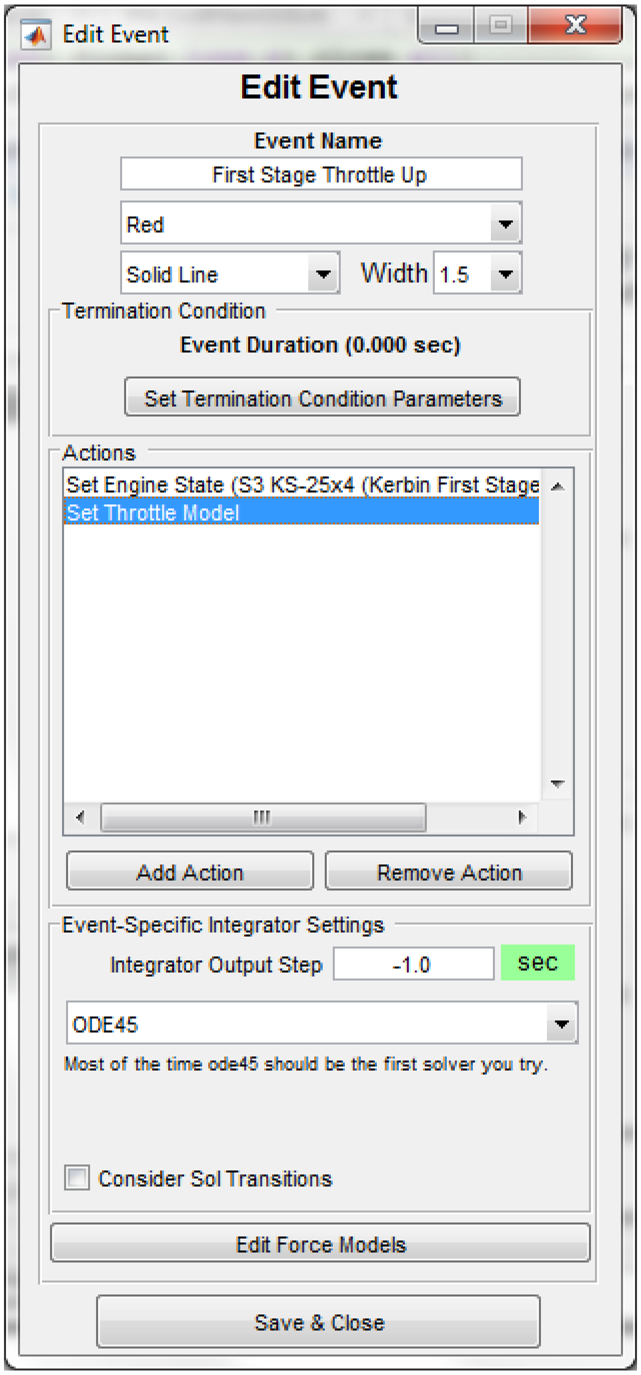
Finally, let's add an action that will activate the first stage engines and begin throttling up.

Click the "Add Action" button. A list dialog will appear. Find "Set Engine Active State" in the list and double click it. A dialog box will appear with a list of engines. Find the Kerbin First Stage engine, select it, and then set the engine state to "Active." Tap Save and Close when you are done.

Add another action, this time a "Set Throttle Model" action. A dialog box will appear asking you what type of model you wish to add. Select "Polynomial Model." In the "Edit Action" dialog box that appears, set the throttle Constant Term to 0.0 %, the Linear Term to 200 %/s, and the Accel. Term to 0.0 %/s/s. Save and Close when you are done.

Finally, uncheck the "Consider Sol Transitions" checkbox.

Save and Close the Edit Event dialog box when you are done.



Edit Event

Event Name: First Stage Throttle Up

Red

Solid Line — Width 1.5

Termination Condition: Event Duration (0.000 sec)

Actions:
- Set Engine State (S3 KS-25x4 (Kerbin First Stage
- Set Throttle Model

Event-Specific Integrator Settings

Integrator Output Step: -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider Sol Transitions

Figure 16: Edit Event dialog box for Event 1

Let's discussion for a moment what we just did.

First, we gave the event a name. This is the name that will show up in the script event list and other places in the software.

Second, we set the termination condition for the event to 0.0 seconds duration. This means that the event will last 0.0 seconds from the initial state or end of the last (in other words, it terminates immediately).

Third, we created two event actions. The first event action toggles the first stage engine state from inactive to active. It is now ready to produce thrust according to the throttle setting. The second action begins throttling up the engines at a rate of 200 % per second, meaning that in 0.5 seconds the engines will reach full thrust.

Finally, we unchecked the “Consider SoI Transitions” box. This is a CPU time saving measure. If this box is unchecked, LVD will not transition to a new sphere of influence during this event because the check required to see if an SoI change is needed is not run. Because we are at the surface of Kerbin and not expecting to need to change spheres of influence any time soon, we should uncheck this box to make the script execute a bit faster.

How do events work?

When an event is propagated, a number of things happen in a particular order:

1. The final state of the previous event (or the initial state if this is the first event) is inherited as the initial state of the event.
2. The spacecraft position, velocity, and mass are propagated according to the various force models active and equations of motion.
3. Event propagation is terminated when the termination condition is achieved.
4. The event actions are executed at the end of the event propagation in the order they are shown in the Actions list in the Edit Event dialog box.

Note that if an event propagation termination condition is not reached in a sufficient amount of simulation time, the event will stop propagating where it is. You can alter this amount of time in the Settings -> Integration Settings -> Max Simulation Time menu. This represents the total simulation time for the sum of all events.

Also note that if the propagation takes too much real-world clock time to propagate, propagation will be terminated. You can alter this amount of time using the Settings -> Integration Settings -> Max Script Propagation Time menu. A good value for this number is somewhere between 5 seconds (for short missions) and 30 seconds (for long missions).

3.2.1.2 Propagating to Full Throttle

Now let’s create another event by pushing the “Insert Sequential Event” button on the main LVD user interface. (The difference between sequential events and non-sequential events will be discussed later.)

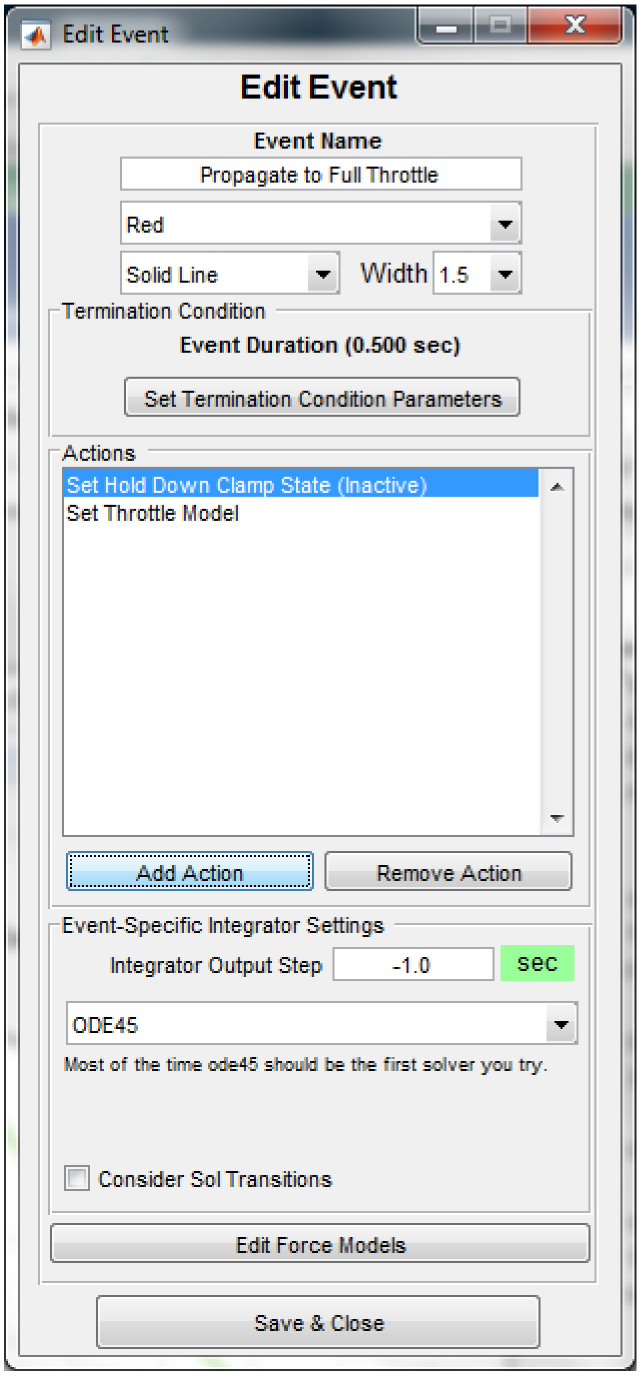
Notice the familiar Edit Event dialog box appears. Name this new event "Propagate to Full Throttle". In the Termination Condition area, set the event termination condition to a duration of 0.5 seconds, similar to what you did for the last event.

There are two things we want to do at the end of the 0.5 second period. First, we want to release the hold down that is keeping our rocket attached to Kerbin. Add another event, "Set Hold Down Clamp State." In the dialog box that appears, set the state to Inactive.

Second, we want to set the throttle model to be a constant 100% thrust. Create another "Set Throttle Model" action, select the polynomial type, and set the constant term to 100% and the linear rate and acceleration to 0.0.

Finally, as usual, uncheck the "Consider SoI Transitions" checkbox.

Save and Close the Edit Event dialog box. You should notice that the final spacecraft state is now 0.5 seconds later than the initial state. The actual position and velocity of the vehicle will not have changed (relative to the surface of Kerbin), though, because the hold down is released at the *end* of the event.


Edit Event

Edit Event

Event Name: Propagate to Full Throttle

Red

Solid Line, Width 1.5

Termination Condition

Event Duration (0.500 sec)

Actions:
- Set Hold Down Clamp State (Inactive)
- Set Throttle Model

Event-Specific Integrator Settings

Integrator Output Step: -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider SoI Transitions

Figure 17: Edit Event dialog box for Event 2

3.2.1.3 Propagate to Tower Clear

At this stage of our simulation, our rocket will begin lifting off the launch pad. In a few seconds, it will need to begin rolling to the correct heading to put it into space. However, for safety reasons, we want to rocket to clear the launch tower before it begins any maneuvers.

Let's say that the launch tower is 30 meters (0.03 km) tall. If our initial altitude above the sea level of Kerbin is 0.06841 km, then we know we will want to propagate to an altitude above sea level of 0.09841 km. Let's do this with another event.

Create a new event with the new "Propagate to Tower Clear".

In the Termination Condition area, push the "Set Termination Condition Parameters" button. Set the termination condition type to "Altitude" in the dialog box. The altitude target should be 0.09841 km. Save and Close the termination condition dialog box when you are done.

The only event action we will want to add here is one which adds some yaw rate in order to steer our vehicle onto the correct launch azimuth to make our orbit target later.

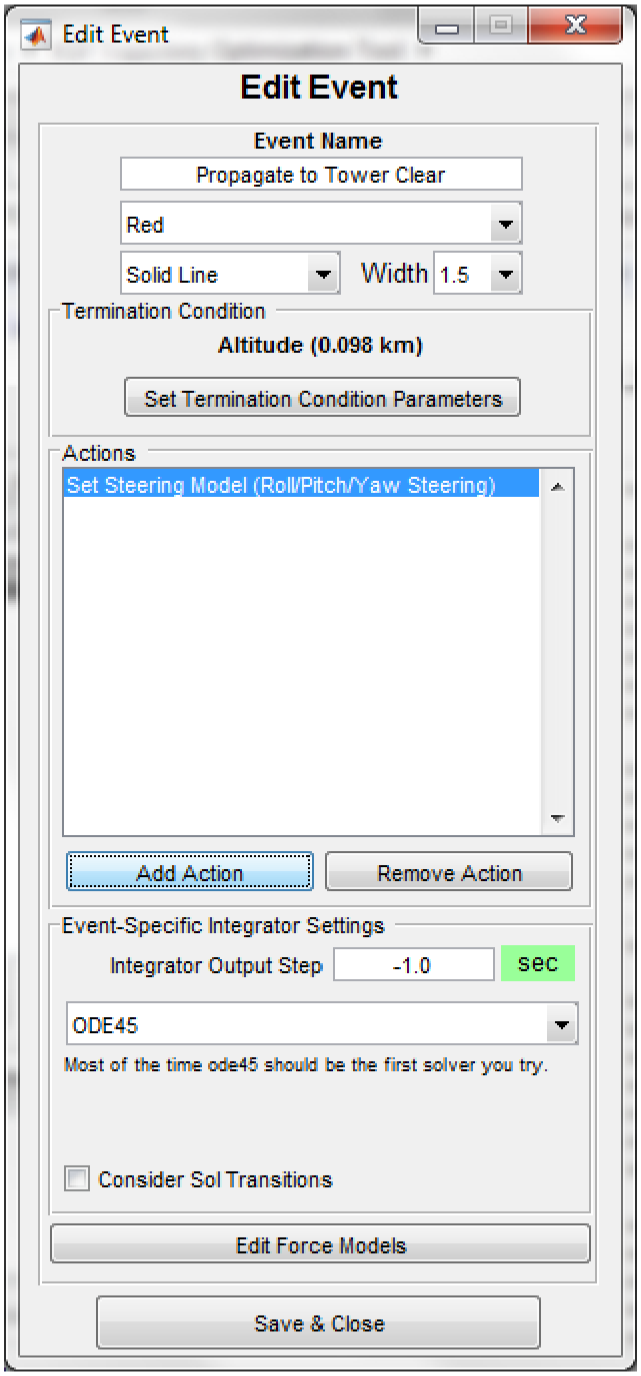
Add an action and select "Set Steering Model". Leave the Roll/Pitch/Yaw Steering as-is, and check the "Preserve Angle Continuity at Start of This Steering Model?" checkbox at the bottom. You'll notice that the constant terms of the various angles are greyed out. This is because this steering model will inherit those values from the attitude state of the vehicle.

Let's set the linear rate of the Yaw Angle to 30.0 deg/s.

Also, since at this point we are clear of the launch tower, we can begin pitching the vehicle downwards. Set the linear rate of the Pitch Angle to -1.0 deg/s. (NOTE: The sign on this angle is very important. It must be negative here.) Finally, check the "Opt?" checkbox next to the Linear Term and set the bounds on this variable to -2.5 to 0.0 deg/s.

Save and Close the Steering Model dialog box when you are done.

Finally, as usual, uncheck the "Consider SoI Transitions" checkbox.



Propagate to Tower Clear
Red
Solid Line
Width 1.5
Termination Condition
Altitude (0.098 km)

Set Steering Model (Roll/Pitch/Yaw Steering)

Integrator Output Step -1.0 sec
ODE45
Most of the time ode45 should be the first solver you try.
☐ Consider SoI Transitions

Figure 18: Edit Event dialog box for Event 3

When you Save and Close the Edit Event dialog box, notice that the final state of the vehicle is now about 6 seconds after the initial state.

How does the Roll/Pitch/Yaw steering model work?

This steering model is a three-rotation model in which each rotation angle is modeled independently in time using polynomials. When all angles are 0.0 degrees, the vehicle is oriented wings level, heads up, pointed *north.*

The order of rotation is as follows:

1. The vehicle is rotated around about the Yaw axis.
2. The vehicle is rotated about the Pitch axis.
3. The vehicle is rotated about the Roll axis.

The yaw axis is normally the negative radial vector. The pitch axis is horizontal to the surface of the body. The roll axis is generally down the long axis of the vehicle.

The other steering models use a similar definition for their order of rotation.

Note also that we created our first optimization variable in this event section. The pitch rate of this event will be optimized whenever we call the optimizer. You should notice that the yellow warning about their not being any optimization variables on the script has gone away and been replaced with a green “No Errors” box. You’ll also notice an asterisk (“*”) next to the Event 3 name in the script event list. This means that there is at least one active optimization variable present in that event.

3.2.2 Tower Clear to Orbit Insertion

In this section, we will create events that terminate the yaw rate of the vehicle and define a series of pitch events to allow the vehicle to reach orbit.

3.2.2.1 Terminating the Yaw to Desired Launch Azimuth

Recall that previously we set the initial yaw angle to 90 degrees (due East) and the yaw rate to 30 deg/s. The first thing we need to do after clearing the tower is terminate that yaw such that we end up on the correct launch azimuth. Let’s do so by creating a new event.

Title the new event "Terminate Roll to Launch Azimuth".

Leave the termination condition as "Event Duration" but set the time to 1 second. In the "Edit Termination Condition" dialog box, also check the "Opt?" box to optimize the duration of this event. Then, set the lower and upper bounds of the event duration to be 0.0 seconds and 3.0 seconds respectively. The initial guess of 1.0 seconds comes from the fact that we want a 30 degrees inclination (see Table 2), and so a yaw of 30 degrees off east is achieved after 1 second of 30 deg/sec of yawing.

The guess on the upper bound corresponds to a full 90 degrees of yaw rotation, which would actually put the vehicle pointing due south. We know this is too much for a 30 degrees inclination, and so it will serve as a good upper bound.

Add an action to set the steering model again and check the box to preserve angle continuity.

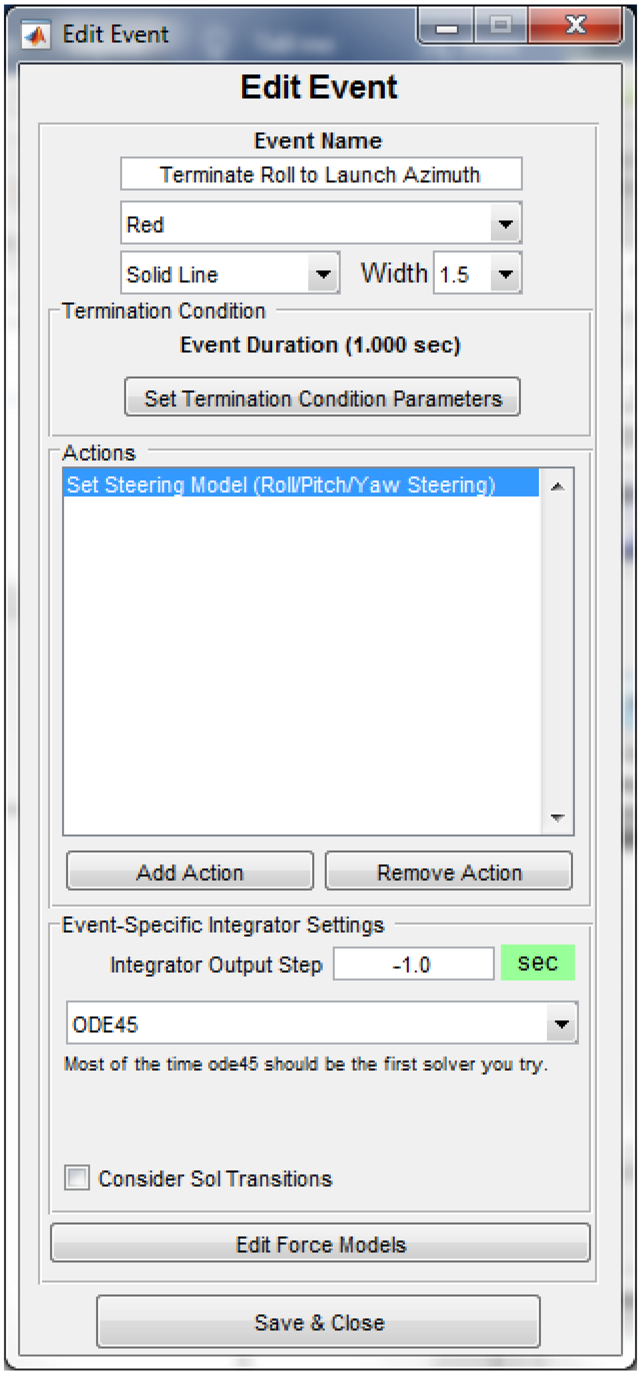
Set the Yaw Angle Linear Term to be 0.0 deg/s. This will terminate the roll.

Set the Pitch Angle Linear Term to be -0.1 deg/s. As before, tap the "Opt?" checkbox and set the bounds to -0.75 deg/s and 0.0 deg/s.

Save and Close the steering model dialog box.

Finally, as usual, uncheck the "Consider SoI Transitions" checkbox.

Save and Close the Edit Event dialog box.



Terminate Roll to Launch Azimuth

Red

Solid Line Width 1.5

Termination Condition

Event Duration (1.000 sec)

Set Steering Model (Roll/Pitch/Yaw Steering)

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider SoI Transitions

Figure 19: Edit Event dialog box for Event 4

Notice that about 7 seconds has elapsed in the final spacecraft state, which is 1.0 seconds more than what we had last time we created an event. This is the expected result, as our event duration is set to 1.0 seconds.

When you create a new event and go to save it, the propagation engine automatically propagates the mission script with the new event included. Because of the way MATLAB works, the first time a new event that uses a new function is propagated, there may be an excessive pause while the MATLAB engine "Just in Time" compiles the new code.

If this happens, you will likely get a warning that the maximum simulation propagation time has been exceeded. Just re-run the script by using Simulation -> Run Script (or cntrl-P). The script should execute much quicker.

If this does not resolve the issue, then you will need to increase this duration as discussed earlier.

At this point, the vehicle is configured to be on the correct launch azimuth (or it will be after optimization), pitching down from vertical begin gaining horizontal speed, and all engines are running at full throttle.

3.2.2.2 Defining the Non-Sequential Staging Events

At some point, we would anticipate that our first stage and second stage of the Kerbin ascent vehicle will run out of fuel and need to be discarded. It is possible to do this with normal events as we have been using so far, but this way lacks flexibility as it forces events to occur in a particular sequence. What if we do not know where in the script the first stage will need to be jettisoned? Or what if we do not want to be constrained by a particular script set up? This is what non-sequential events are for. They allow events to be executed at indeterminate times in parallel with the main mission script.

Let's set up an event that will jettison the first stage when it runs out of fuel. Tap the "Insert Non-Sequential Event" button. A dialog box appears for the new non-sequential event.

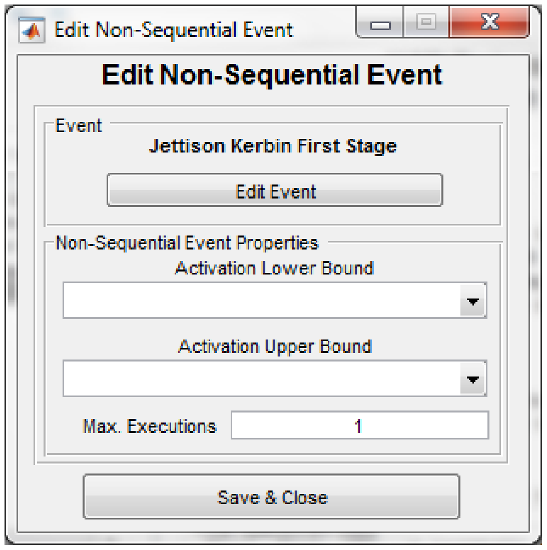

Edit Non-Sequential Event
Edit Non-Sequential Event
Event
Jettison Kerbin First Stage
Edit Event
Non-Sequential Event Properties
Activation Lower Bound
Activation Upper Bound
Max. Executions
1
Save & Close

Figure 20: Edit Non-Sequential Event dialog box

First, tap the "Edit Event" button. This will bring up the familiar Edit Event dialog box.

Title the event "Jettison Kerbin First Stage".

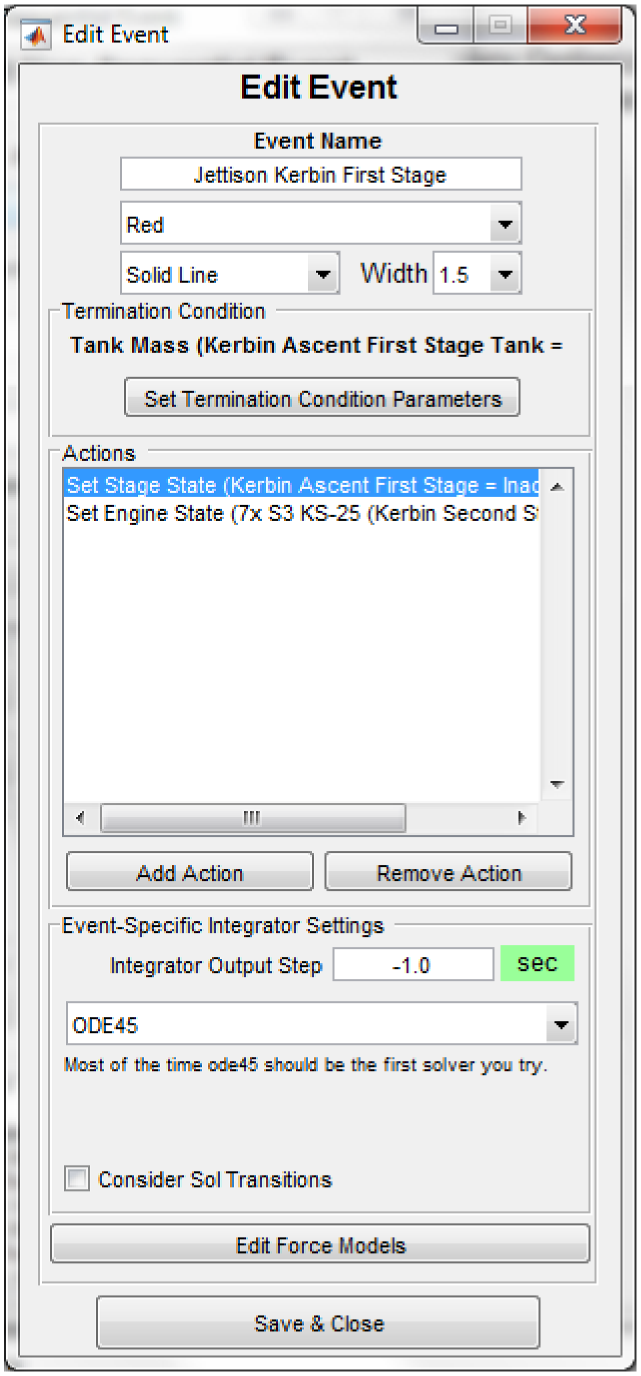
Set the Termination Condition to "Tank Mass", select the Kerbin Ascent First Stage tank, and set the tank mass to 0.0 mT.

Now we create an event action that will deactivate the booster stage. Add an action, select the "Set Stage Active State", select the Kerbin Ascent First Stage, and set the stage state to Inactive.

Next, we need to ignite the second stage engines here. Create a new action, "Set Engine State," select the "Kerbin Ascent Second Stage" engines and set the state to "Active."

Finally, as usual, uncheck the "Consider SoI Transitions" checkbox.

Save and Close the Edit Event dialog box.



Jettison Kerbin First Stage
Red
Solid Line
Width 1.5

Tank Mass (Kerbin Ascent First Stage Tank =

Set Stage State (Kerbin Ascent First Stage = Inac
Set Engine State (7x S3 KS-25 (Kerbin Second S

Integrator Output Step -1.0 sec
ODE45
Most of the time ode45 should be the first solver you try.
☐ Consider SoI Transitions

Figure 21: Jettison Kerbin Ascent First Stage event

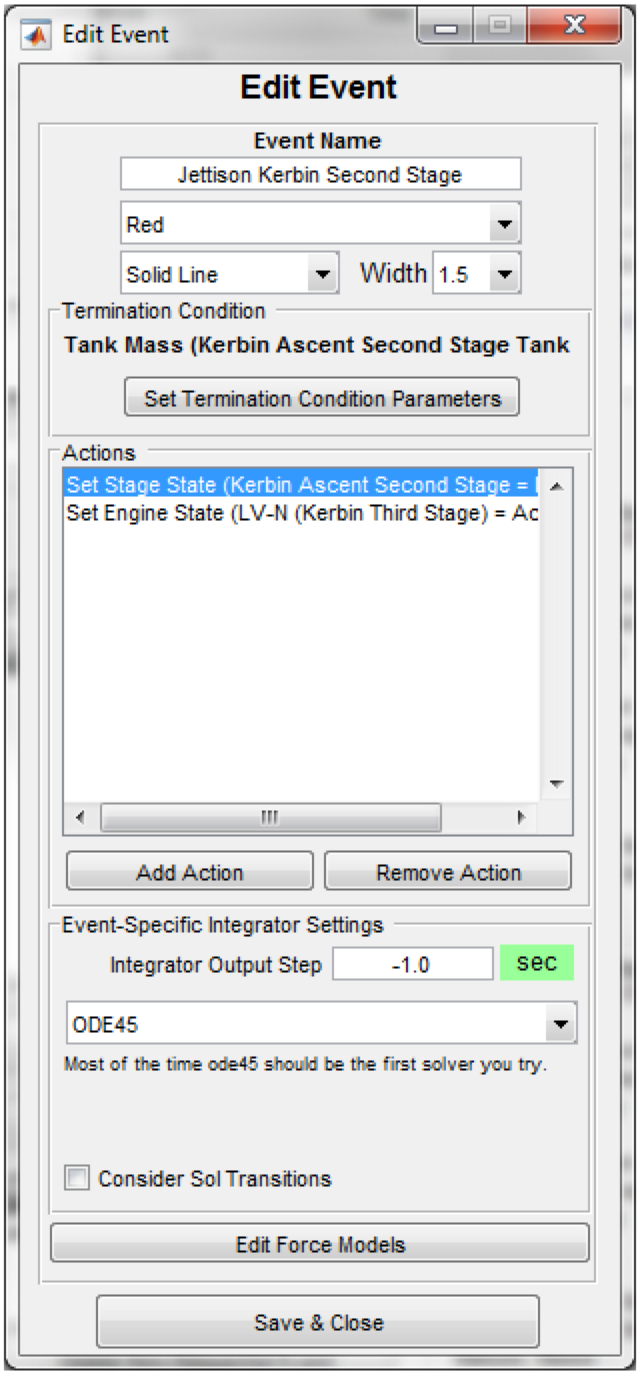
Within the Edit Non-Sequential Event dialog box, you will notice "activation" bounds. These define the events that the non-sequential event is allowed to execute between. A blank bound implies either no lower bound or no upper bound. For now, leave both bounds empty.

You will also notice a field labeled "Max. Executions." This is the number of times that this non-sequential event can execute over the course of the whole mission. Generally, you will want to leave this at 1, but there are cases in which allowing more executions may be desirable. For now, leave it at 1.

Tap the Save and Close button to return the main LVD user interface.

Repeat this process once more, setting up a jettison event for the second stage of the Kerbin ascent vehicle. Title it "Jettison Kerbin Second Stage". Use the second stage tank, triggering when that tank is at 0.0 mT, as before. Deactivate the respective stage as before. In addition, when jettisoning the second stage, activate the third stage engines.

Your new non-sequential event should look like that in Figure 22 when you are done defining it.



Jettison Kerbin Second Stage
Red
Solid Line
Width 1.5
Termination Condition
Tank Mass (Kerbin Ascent Second Stage Tank

Actions
Set Stage State (Kerbin Ascent Second Stage = I
Set Engine State (LV-N (Kerbin Third Stage) = Ac

Event-Specific Integrator Settings
Integrator Output Step -1.0 sec
ODE45
Most of the time ode45 should be the first solver you try.
Consider Sol Transitions

Figure 22: Edit Event Dialog Box for Second Stage Jettison Non-Seq Event

3.2.2.3 Defining the Pitch Profile to Reach Kerbin Orbit

In order to reach orbit, the ascent vehicle must continue pitching downwards to the horizon. (And, in fact, there are cases where small pitch up maneuvers may occur as well.) However, we cannot know ahead of time what the desired pitch rate will be at every moment. Therefore, we want to optimize the full Kerbin ascent pitch profile. We define this profile through the use of the linear pitch rates at each event termination.

To do so, we will be creating seven events after the "Terminate Roll to Launch Azimuth" event in the script event list. Each of these events will be set up identically to each other, and each will have an optimized pitch rate that the optimizer can adjust in order to maximize ascent performance.

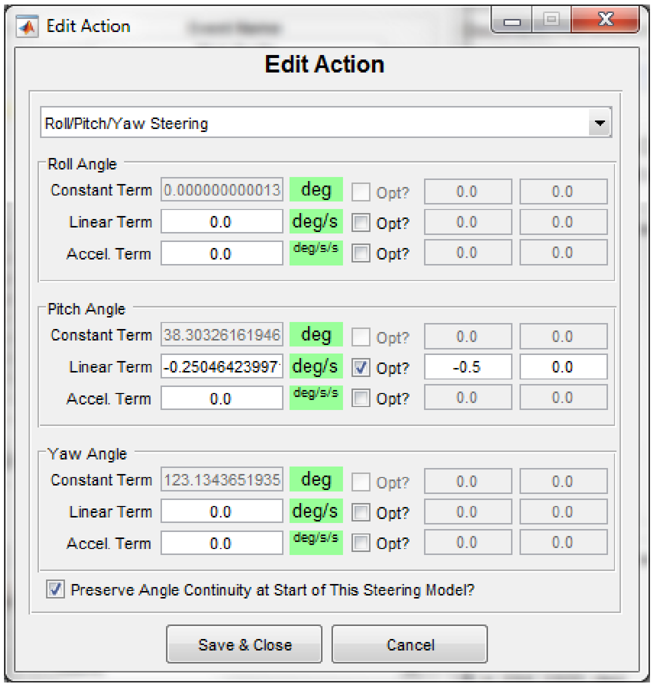
Create the following event seven times. It should show up in the script event list as Events 5-12.

Table 15: Pitch Profile Event Properties

Event Property	Value
Name	"Pitch Profile"
Termination Condition Type	Event Duration
Termination Condition Value	65 seconds
Termination Condition Bounds	5 to 200 seconds
Termination Condition Opt? Flag	Checked

Create a new "Set Steering Model" action for each of these events. In them, set the Pitch Angle Linear Term to -0.1 deg/s, check the Opt? box for that value, and set the bounds to -0.75 deg/s to 0 deg/s. Also make sure that the "Preserve Angle Continuity" checkbox is checked.

When you are done, your Steering Model dialog box will look something like this:



Edit Action

Roll/Pitch/Yaw Steering

Roll Angle

Constant Term	0.000000000013	deg	Opt?	0.0	0.0
Linear Term	0.0	deg/s	Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

Pitch Angle

Constant Term	38.30326161946	deg	Opt?	0.0	0.0
Linear Term	-0.25046423997	deg/s	✓ Opt?	-0.5	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

Yaw Angle

Constant Term	123.1343651935	deg	Opt?	0.0	0.0
Linear Term	0.0	deg/s	Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

✓ Preserve Angle Continuity at Start of This Steering Model?

Figure 23: Kerbin Ascent Pitch Profile Steering Model Setup

Save and Close all dialog boxes and return to the main LVD window.

Finally, we need to create one more event that will set the throttle to zero at the end of our ascent burn.

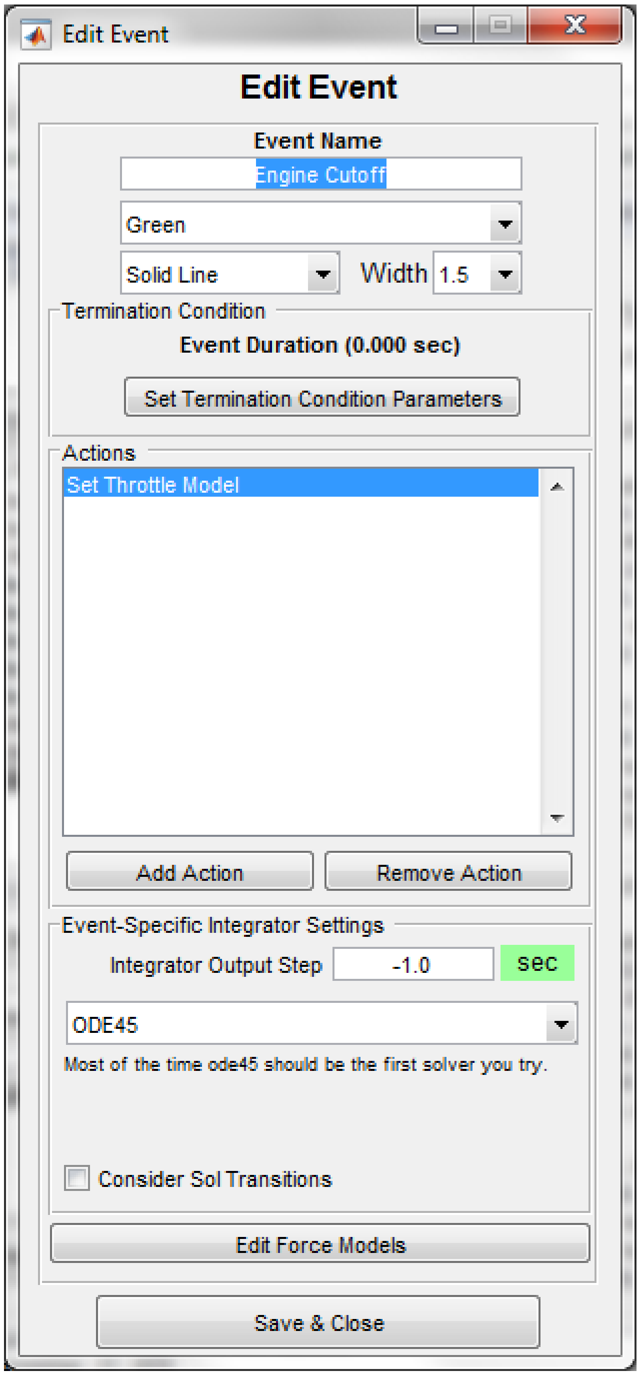
Title the event “Engine Cutoff”.

Set the Termination Condition to “Event Duration” and set the duration to 0.0 sec. This will not be optimized.

Now we create an event action that will set the throttle to zero. Create a “Set Throttle Model” action, select the polynominal type (as before), and set all values to 0.0. This represents an instantaneous change of throttle to 0.0.

Finally, as usual, uncheck the “Consider Sol Transitions” checkbox.

Save and Close the Edit Event dialog box.


Edit Event

Event Name: Engine Cutoff

Green

Solid Line

Width 1.5

Termination Condition: Event Duration (0.000 sec)

Actions: Set Throttle Model

Event-Specific Integrator Settings

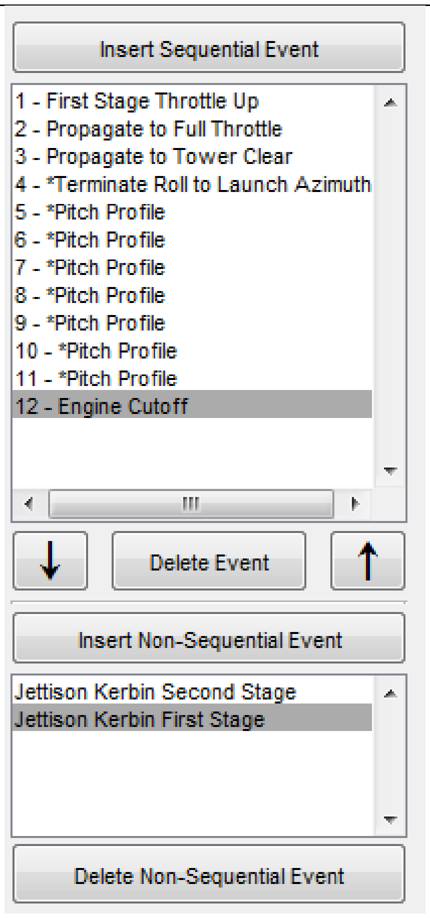
Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider Sol Transitions

Figure 24: Engine Cutoff event

When you are done, your script list should look like the one on the right.	 Insert Sequential Event 1 - First Stage Throttle Up 2 - Propagate to Full Throttle 3 - Propagate to Tower Clear 4 - *Terminate Roll to Launch Azimuth 5 - *Pitch Profile 6 - *Pitch Profile 7 - *Pitch Profile 8 - *Pitch Profile 9 - *Pitch Profile 10 - *Pitch Profile 11 - *Pitch Profile 12 - Engine Cutoff Delete Event Insert Non-Sequential Event Jettison Kerbin Second Stage Jettison Kerbin First Stage Delete Non-Sequential Event Figure 25: Script Event List for Kerbin Ascent

We are now ready to optimize the ascent to achieve our target orbit.

3.2.2.4 Optimizing the Ascent

The first step in optimizing the ascent is to provide the optimizer with constraints that define our orbit at the end of the ascent. Use the "Optimization" menu and select "Edit Constraints." A dialog box will appear.

Add four constraints with the following properties by tapping the Add Constraint button.

Table 16: Kerbin Ascent Target Constraint Defintions

Constraint Property	Constraint 1	Constraint 2	Constraint 3	Constraint 4
Constraint Type	*Altitude*	*C3 Energy*	*Hyp. Vel. Vect. R.A.*	*Hyp. Vel. Vect. Decl.*
Upper Bound	10000.0	1.11538754071	204.3393	-27.1902
Lower Bound	71.0	1.11538754071	204.3393	-27.1902
Scale Factor	100.0	1.0	100	10.0
Celestial Body	<blank>	<blank>	<blank>	<blank>
Applicable Event	Engine Cutoff	Engine Cutoff	Engine Cutoff	Engine Cutoff
Constraint Active?	Checked	Checked	Checked	Checked

Notice that the third and fourth constraint are straight from Table 3: Kerbin Departure Targets. The second constraint is actually C3 energy and not the magnitude of the hyperbolic velocity vector. Why? C3 energy is defined for all orbit types, elliptical and hyperbolic. The magnitude of that hyperbolic velocity vector is only defined for hyperbolic orbits. It is never good to use constraints that potentially change their behavior in the middle of an optimization, hence why we select C3 energy to use as a constraint instead.

Semi-major axis is another quantity that could be used to represent the magnitude of the hyperbolic velocity vector. It is also defined for elliptical and hyperbolic orbits. However, SMA is not smooth when transitioning from elliptical to hyperbolic values: it goes to positive infinity and then becomes negative. The optimizer will not like this behavior.

C3 energy, on the other hand, is smooth through the elliptical/hyperbolic transition. It is negative for elliptical orbits, zero for parabolic orbits, and positive for hyperbolic orbits.

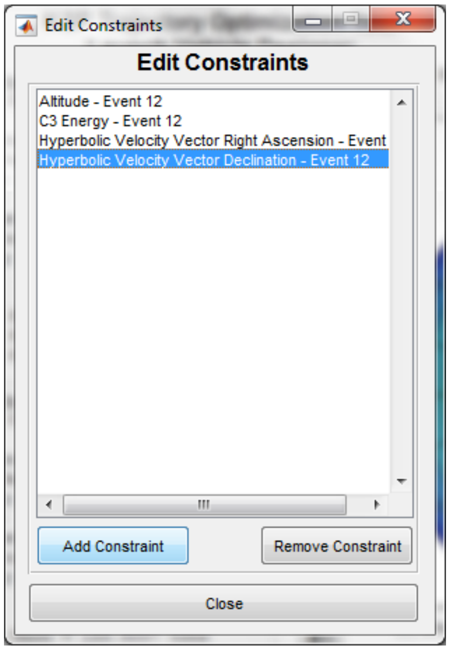
C3 energy is known as the "characteristic energy" of the orbit.

C3 energy is related to the magnitude of the hyperbolic excess velocity vector, v_∞, by the following relationship:

$$C_3 = v_\infty^2$$

So, we just square the value in Table 3 and make it our constraint.

Save and Close the Edit Constraint dialog box after entering each constraint. When you are done, your Constraint dialog box should look like the following.



Altitude - Event 12
C3 Energy - Event 12
Hyperbolic Velocity Vector Right Ascension - Event
Hyperbolic Velocity Vector Declination - Event 12

Figure 26: Kerbin Ascent Constraints - Part 1

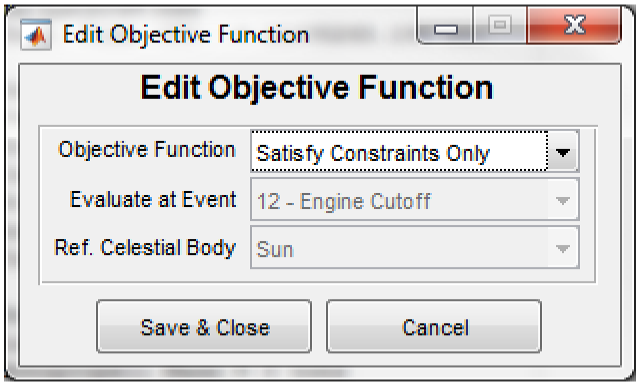
Close the Edit Constraints dialog box.

Note that this will only be the first time that we add constraints for our Kerbin ascent. These are only used to provide the initial ascent target. Later we will re-optimize the ascent to active target Eve directly.

Next, we need to set up the objective function. Use the Optimization -> Edit Objective Function menu to do so.

At first glance it might seem intuitive to set the objective function to "Maximize Vehicle Mass." However, this is actually in opposition to our C3 energy constraint, which demands a particularly high orbit energy. In cases when you start optimizing far from a good solution, maximizing the vehicle mass (or expending less propellant than needed) can actually run your vehicle into the ground in a way that is not recoverable. In fact, if you ever see your altitude hit the minimum altitude for the simulation (-1 km by default), then it is wise to stop the optimization, discard the results, and tweak the current mission state in some way to try to get the optimizer to avoid that solution.

In any event, let's make sure that the objective function is set to "Satisfy Constraints Only." Save and Close the dialog box.


Edit Objective Function

Edit Objective Function

Objective Function: Satisfy Constraints Only

Evaluate at Event: 12 - Engine Cutoff

Ref. Celestial Body: Sun

Figure 27: Kerbin Ascent Objective Function - Part 1

Finally, we need to head back to the initial state and modify the initial epoch to be an optimized quantity. This is akin to changing the time of the launch. Use the Launch Vehicle -> Edit Initial State menu item to do so.

Check the "Opt?" box next to Epoch (UT). Set the lower and upper bounds to 11293200.0 seconds and 11314800.0 respectively. Save and Close the dialog box.

⚠ It is usually necessary to optimize the time of launch to meet a particular hyperbolic velocity vector right ascension angle value.

It is now time to optimize the mission. Use the Optimization -> Optimize Mission menu item (or cntrl-O) to run the optimizer. This will take some time, usually a few minutes, so sit back and relax for a bit.

What you should see when you run the optimizer this first time is 1) the objective function is always 1, and 2) that the maximum constraint violation is ultimately trending towards 0.0. Initially the constraint violation might bounce around and this is fine. However, within 10-20 iterations, it should be settling down and trending towards 0.0.

When the constraint violation is small (<= 0.01) and the objective function value is not moving, feel free to stop the optimization using the "Stop" button. The optimizer may also quit if it believes it has satisfied all constraints and the objective function is as good as it's going to get, or if it can't find a way to satisfy all constraints. In the case of the latter, try running the optimizer a few times to see if it can't break out of that behavior.

Once you've successfully optimized the mission to have met our orbit target, we can move on to the next phase of the trajectory optimization: maximizing the ascent performance.

Change the objective function (Optimization -> Edit Objective Function) to "Maximize Vehicle Mass" at Event 12 – Engine Cutoff.

Now re-run the optimization (Optimization -> Optimize Mission) and wait for the optimizer to converge on a solution. As with before, this may take some time.

Notice here how we got a solution which met the constraints first and then tried to optimize it to improve some metric of the trajectory or vehicle? This is a very common technique for difficult problems and one that we will use often.

If everything has gone well, you should now have an ascent trajectory which has minimized the propellant mass used to make orbit and is on its way out of the Kerbin SoI.

My final solution had an initial state epoch of 11293770 seconds UT and a final liquid fuel mass of 138.5 mT (total vehicle mass at engine cutoff was 190 mT).

3.2.3 Orbit Insertion to Kerbin SoI Departure

We now need to create two more events. The first will jettison our Kerbin Ascent Third Stage and the second will send us out to the Sun sphere of influence.

Create a new event in the main mission script, as many times before.

Title the event "Jettison Kerbin Ascent Third Stage".

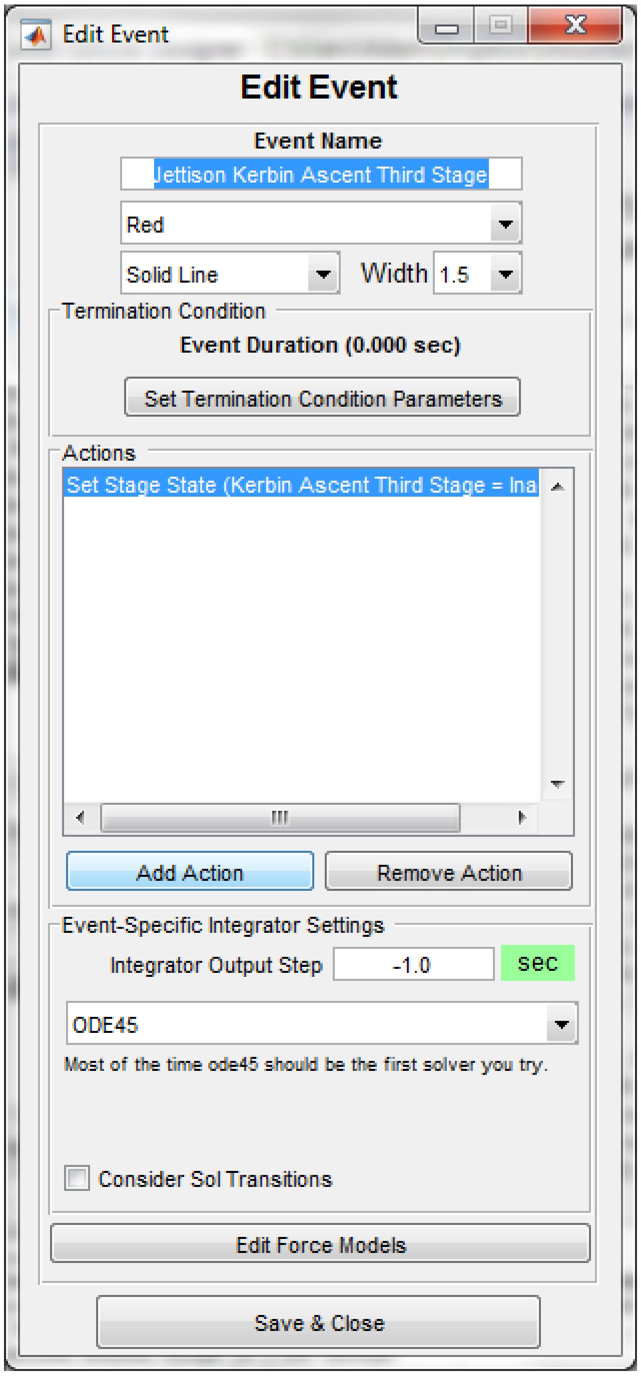
Set the Termination Condition to "Event Duration" and set the duration to 0.0 sec. This will not be optimized.

Now we create an event action that will set the third stage to be inactive. Create a "Set Stage State" action, select the "Kerbin Ascent Third Stage", and set the value to "Inactive".

Let's terminate any attitude motion. Create a new action of type "Set Steering Model." Check the "Preserve Angle Continuity" checkbox at the bottom and make sure all angular rates and accelerations are 0.0.

Finally, as usual, uncheck the "Consider Sol Transitions" checkbox.

Save and Close the Edit Event dialog box.


Edit Event

Edit Event

Event Name: Jettison Kerbin Ascent Third Stage

Red

Solid Line   Width 1.5

Termination Condition: Event Duration (0.000 sec)

Actions: Set Stage State (Kerbin Ascent Third Stage = Ina

Event-Specific Integrator Settings

Integrator Output Step: -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

Consider Sol Transitions

Figure 28: Jettison Kerbin Ascent Third Stage event

3.3 Interplanetary Coast Trajectory 1

Let's create another event, this time to optimize our trajectory to Eve. We will create an event with a duration of about 52 Earth days to begin.

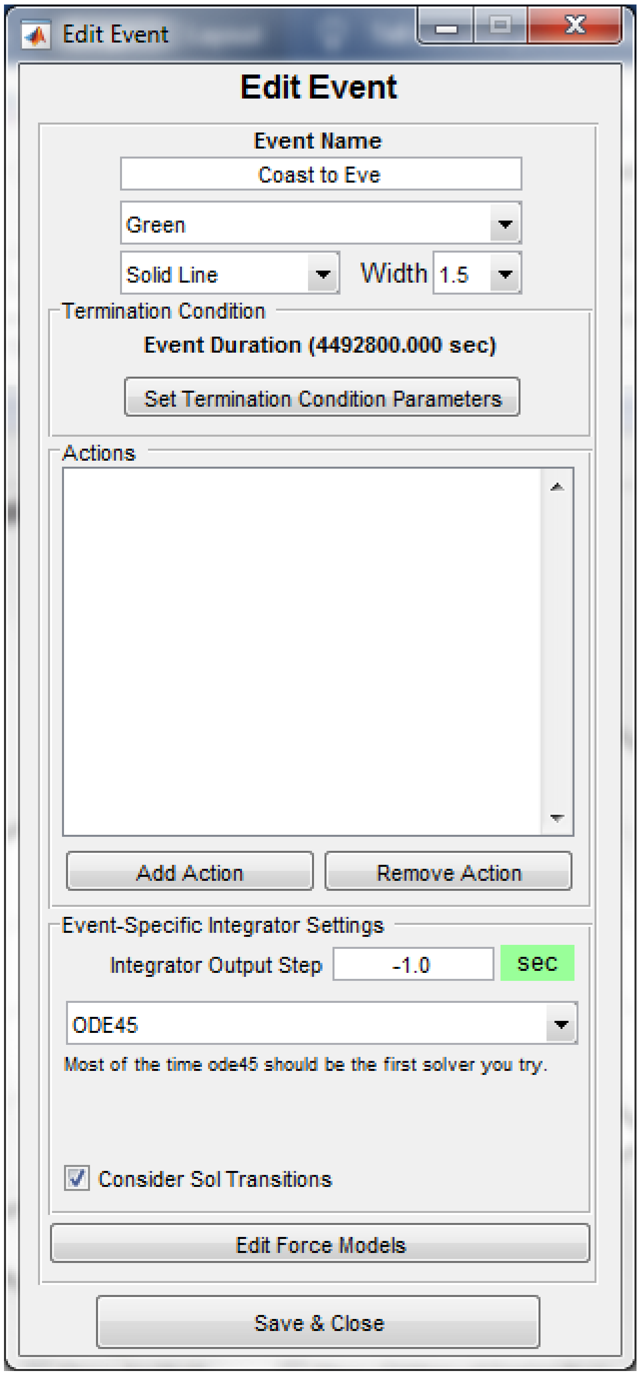
Title the event “Coast to Eve”.

Set the Termination Condition to “Event Duration”. Set the initial value to 52 days (or 4492800.0 seconds). Check the “Opt?” box and set the lower bound to 3888000.0 seconds (45 days) and the upper bound to 5184000.0 seconds (60 days).

Set the line color from Red to Green if you desire.

Unlike the usual procedure, *check* the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.



Event Name: Coast to Eve

Green

Solid Line   Width 1.5

Event Duration (4492800.000 sec)

Integrator Output Step -1.0 sec

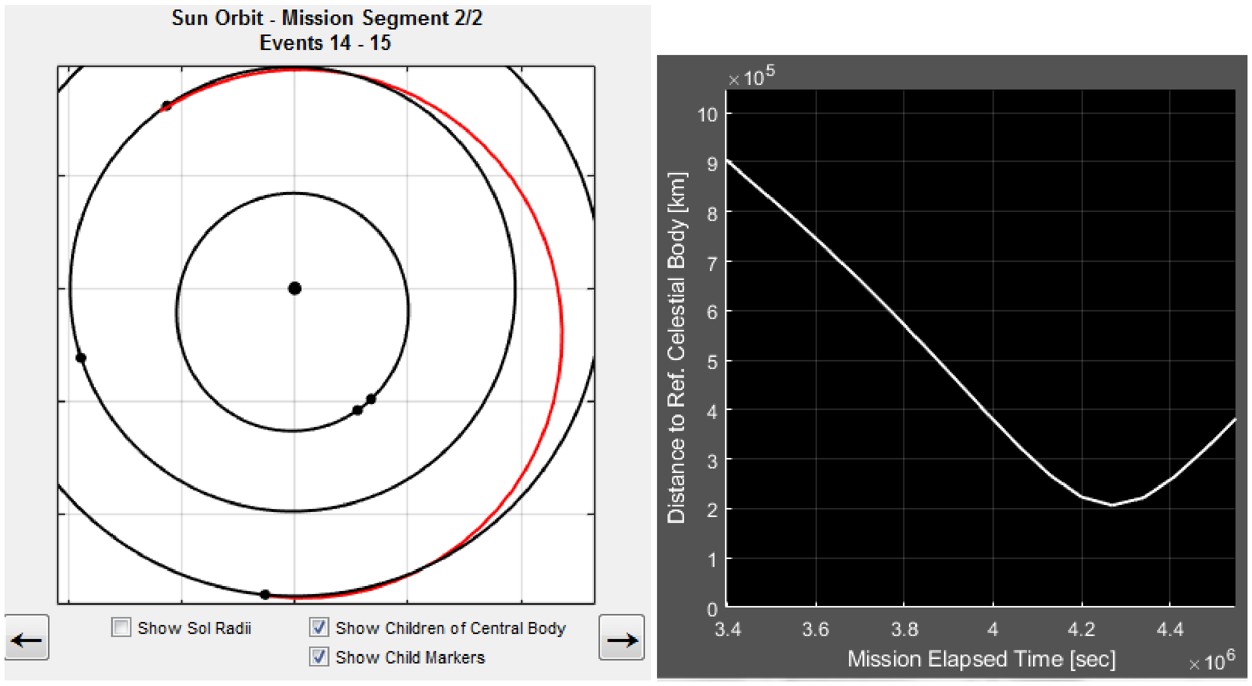
ODE45

Most of the time ode45 should be the first solver you try.

[✓] Consider SoI Transitions

Figure 29: Coast to Eve event

You should immediately notice that our trajectory gets very close to Eve without doing more than we’ve already done! We might not hit the SoI but we are very close. By the way, this should be a good indication as to how well the numbers that come out of the porkchop plot actually work! (Answer: pretty well!)


Sun Orbit - Mission Segment 2/2
Events 14 - 15
Distance to Ref. Celestial Body [km]
× 10^5
0
1
2
3
4
5
6
7
8
9
10
3.4
3.6
3.8
4
4.2
4.4
Mission Elapsed Time [sec]
× 10^6

☐ Show Sol Radii
☑ Show Children of Central Body
☑ Show Child Markers

Figure 30: Kerbin to Eve Trajectory prior to Ascent Optimization #3

It's time to prepare to optimize this trajectory.

Open up the Edit Constraint dialog box (Optimization -> Edit Constraints). Now open up the constraints for C3 energy, hyperbolic velocity vector right ascension, and hyperbolic velocity vector declination on the "Engine Cutoff" event. Uncheck the "active" checkbox and Save and Close all three of these events. This disables these constraints.

Add a new C3 energy constraint at this point, evaluated at the Engine Cutoff event, with a lower bound of 0.001 km^2/s^2 and an upper bound of 1000.000 km^2/s^2. This should help ensure that the ascent trajectory is still hyperbolic when it terminates above Kerbin.

In the event that we hit approach the surface of Eve in the course of our optimization (an event we hope happens!), we need to ensure that the simulation does not crawl to a halt when the surface is reached. This can happen in very thick atmospheres such as the one Eve has. This phenomenon is called "stiffness" and it is to be avoided when propagating the trajectories of spacecraft and rockets. In any event, use the Settings -> Integration Settings -> Minimum Altitude menu. Set the minimum altitude to 0.0 km.

Finally, let's set the objective function to "Minimize Distance to Body" (Optimization -> Edit Objective Function). Set the reference celestial body to Eve and the event to evaluate it at as "Coast to Eve."

In any event, rerun the optimizer (Optimization -> Optimize Mission). This optimization could take some time, so be patient with it. You may need to rerun the optimizer a few times in order to get a solution that provides a trajectory which hits Eve. It doesn't have to be perfect, it just needs to intersect the planet.

3.4 Eve Arrival Trajectory & Entry, Descent, and Landing

Recall from Table 5 that there is a particular landing site that we wish to achieve. Let's now create some constraints to do so. Open up the Edit Constraint dialog box and add the following constraints.

Table 17: Eve Arrival Target Constraint Definitions

Constraint Property	Constraint 1	Constraint 2	Constraint 3	Constraint 4
Constraint Type	*Altitude*	*Latitude*	*Longitude*	*Dist to Re.f Cel. Body*
Upper Bound	11.9	-11.5	272.5	800.0
Lower Bound	11.7	-12.5	271.5	699.0
Scale Factor	10.0	1.0	1.0	100.0
Celestial Body	Eve	Eve	Eve	Eve
Applicable Event	Engine Cutoff	Engine Cutoff	Engine Cutoff	Engine Cutoff
Constraint Active?	Unchecked	Checked	Checked	Checked

Note that we allowed for significant margin on each of the altitude/latitude/longitude constraints. The angles are within half a degree and the altitude is within 0.1 km. This is pretty broad. However, for an initial guess at a trajectory, these are sufficient and may be refined later.

Also notice that the altitude constraint is not active. We will add this constraint in later and optimize just the Eve coast event duration to achieve the altitude we want. For now, we are content to slam into the ground.

Change the Objective Function to be "Satisfy Constraints Only" and rerun the optimizer (Optimization -> Optimize Mission). As before, this optimization could take some time.

Warning: This will be the most challenging part of the tutorial. This optimization problem is hard.

When I went through this part of the tutorial, it took me 4-5 hours of fiddling and optimizing to achieve the approximate landing site desired on Eve. In the very likely event that you will have troubles too, you can try the following.

- Swap the optimizer algorithm.
    - Interior Point is the default and works well in many cases. However, it is known to bounce away from good solutions early in the iteration process.
    - SQP tends to handle satisfying constraints a bit better (in these cases) but converges more slowly and can take more time.
- Set the Eve coast duration to its actual propagated value by right clicking the event in the event list and selecting "Copy Duration of Event." Then, edit the Eve coast event and put the value you just copied into the Event Duration. Finally, subtract off 100-200 seconds so that you are still approaching the planet.
    - This allows the optimizer to see a gradient on the coast time. It may see a flat gradient (so no change in the objective function by changing the variable) if the script propagator gets caught up with stopping the script early at the minimum altitude.
- Swap the objective function between "Satisfy Constraints Only" and "Minimize Distance to Body" with Eve selected as the body and "Coast to Eve" as the event.
- Toggle the optimization "off" (so "**") on everything but the Coast to Eve duration and the final pitch profile event. This will probably only work if the solution is already in the vicinity of Eve and you just need to home it in.

If you still have trouble after some time, it is okay to be satisfied with wherever you landed and move on. Just make sure that you have a trajectory which intercepts Eve.

Having achieved our desired latitude and longitude on Eve, it's now time to achieve the desired altitude.

First, we need to turn off the optimization of all other variables except the Coast to Eve event duration. Open up the Edit Initial State dialog and uncheck the "Opt?" box next to the Epoch field. Save and Close the Edit Initial State dialog box.

Right click on Events 4-11 in the script event list. For each one, select the "Toggle Optimization for Selected Event." You will notice two starts ("**") next to the event. This means that the event optimization has been turned off. Events with two starts ** will not optimize their variables and constraints on those events will not be evaluated.

Events that have had their optimization disabled will not evaluate constraints on those events and will not optimize variables on those events. *However,* objective functions associated with optimization disabled events will still be evaluated as normal.

Go back into the Edit Constraints dialog box and open up the Altitude constraint on the "Coast to Eve" event. Activate it by checking the checkbox and then save and close the Edit Constraint and Edit Constraints dialog boxes.

Now rerun the optimizer. It should converge quickly on a solution that meets the desired altitude.

If you have difficulty with this, back off the Coast to Eve event duration until you are above the desired 11.8 km. Then rerun the optimizer.

There are two final steps we need to achieve before we move on. First, open up the Coast to Eve event. Add an action called "Set Hold Down Clamp State". Set the Hold Down State to "Active" and Save and Close the dialog box. This changes the equations of motion within LVD such that our Eve spacecraft will now be fixed the surface of Eve at the altitude, latitude, and longitude we ended up at.

Next, we need to make sure the rocket is upright after we land. Add another action called "Set Steering Model." Set the Pitch Angle constant term to 89.99 degrees and all other values to 0. Leave the "Continuity" checkbox unchecked here.

After this, create an action to modify the drag aerodynamics of the rocket. Set the frontal drag area to 44.1786467 m^2 and the drag coefficient to 0.3. This corresponds to our rocket dropping in radius from the largest 5.0 meter radius parts down to the slightly smaller 0.375 meter radius parts.

We are now ready to go for the surface stay! Right click on the Coast to Eve event and disable optimization on that event too, since we won't need to optimize that event any more after this.

Additional work could have been done here to model the parachute descent and landing. For example, after fixing the trajectory, we could have:

1. Backed off the Coast to Eve event by some period of time that would place the trajectory outside of the atmosphere;
2. Created another event which propagates to some "parachute deploy altitude" and added an action to change the aerodynamics to model increased drag parachutes; and
3. Created yet another event that propagates to the target altitude of 11.8 and then continues as above.

For the sake of brevity and ease of implementation, these steps were skipped. You are welcome to add them back in at any point if you desire!

3.5 Eve Surface Stay

Before we can begin modeling the Eve surface stay, two parameters that influence the integrator. First, set the maximum simulation time (Settings -> Integration Settings -> Max Simulation Time) to 30176018.0 seconds. Then set the maximum propagation time (Settings -> Integration Settings -> Max Script Propagation Time) to 20.0 seconds.

Next, add an event that represents the approximately 136-day surface stay.

Title the event “Eve Surface Stay”.

Set the Termination Condition to “Event Duration”. Set the initial value to 136 days (or 11750400.0 seconds).

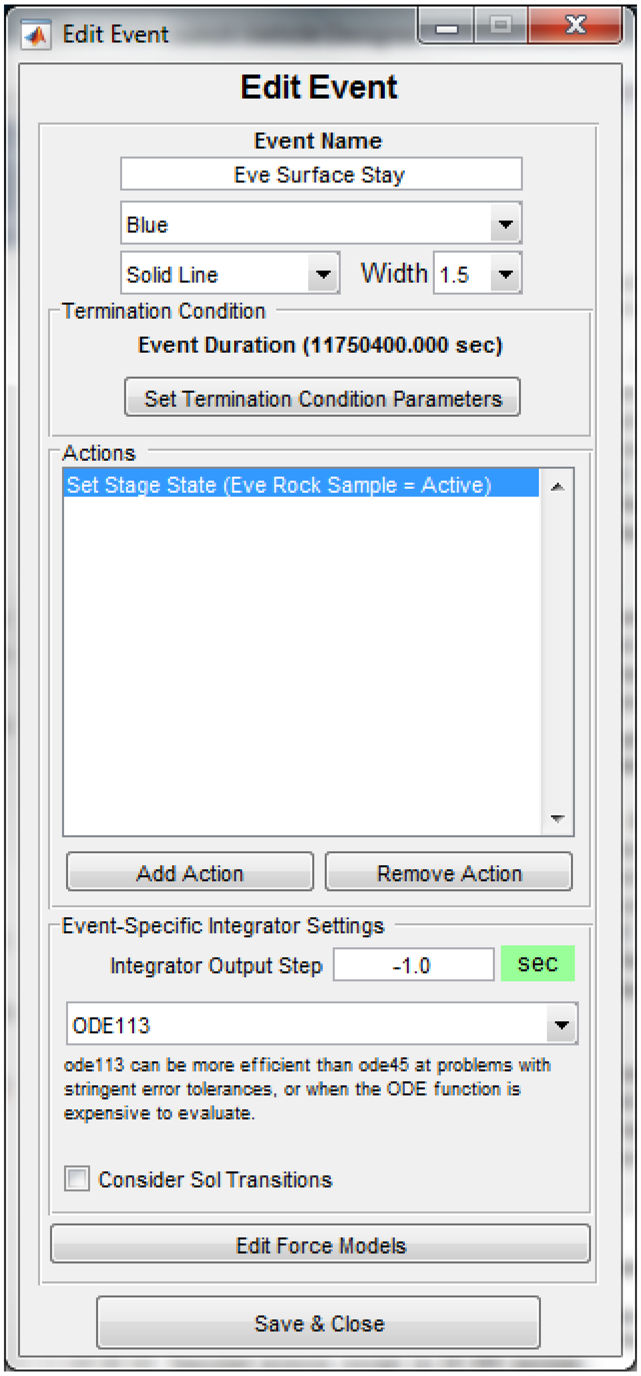
Set the line color from Red to Blue if you desire.

Add an action to set the active state of a stage (“Set Stage State”). Select the “Eve Rock Sample” stage and set it to “Active.” This represents us picking up and storing the 100 kg of Eve rock samples we want to return to Kerbin.

Change the integrator from ODE45 to ODE113.

Finally, as usual, uncheck the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.



Event Name: Eve Surface Stay
Blue
Solid Line — Width 1.5
Termination Condition: Event Duration (11750400.000 sec)
Actions: Set Stage State (Eve Rock Sample = Active)
Event-Specific Integrator Settings
Integrator Output Step -1.0 sec
ODE113
ode113 can be more efficient than ode45 at problems with stringent error tolerances, or when the ODE function is expensive to evaluate.
[ ] Consider SoI Transitions

Figure 31: Eve Surface Stay event

You can confirm that you have created the surface stay event correctly if your final spacecraft state is at around Year 1, Day 319 (see Figure 2).

Why is ODE113 used for the Eve surface stay event?

Our normal integrator of choice will be ODE45 for most mission events. However, for very long duration coasts where only gravity (or launch clamps) dictates the motion, ODE113 can be faster. This is because ODE113 can ultimately take much longer steps in time than ODE45, provided that the propagation is long enough for ODE113 to get a feel for the dynamics and confidently take those larger steps. ODE113 will still be slower than ODE45 if the events are too short.

Next, we will create an event that will actually optimize the Eve launch time. Insert another event after the Eve Surface Stay event.

Title the event “Optimize Eve Launch Time”.

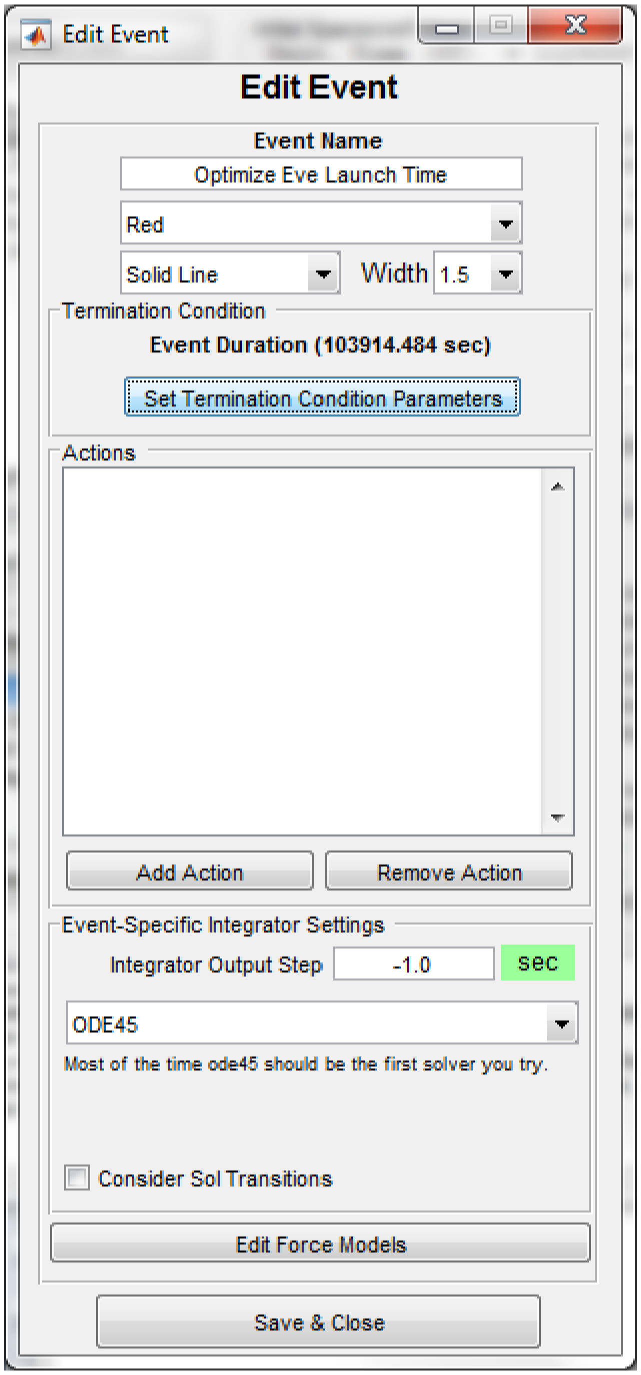
Set the Termination Condition to “Event Duration”. Set the initial value to 100000.0 seconds. Check the “Opt?” flag and set the lower bound to 0.0 seconds and the upper bound to 200000.0 seconds.

Set the line color from Red to Blue if you desire.

Change the integrator from ODE45 to ODE113. The ODE113 algorithm will be faster at propagating these long-duration surface stays.

Finally, as usual, uncheck the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.



Optimize Eve Launch Time

Red

Solid Line

Width 1.5

Event Duration (103914.484 sec)

Set Termination Condition Parameters
Actions
Add Action
Remove Action
Event-Specific Integrator Settings

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider SoI Transitions

Figure 32: Optimize Eve Launch Time event

We are setting up the Eve surface stay in two parts like this because Launch Vehicle Designer is smart enough to know that it should not propagate events that will not be changed when optimizing. If we only optimize this short stay, we can propagate the long 136 day stay only once and then not have to worry about it, thereby speeding up our optimization speed immensely.

This concludes the surface stay modeling.

Note that if the script takes an excessively long amount of time to execute after inserting the surface stay event, you can do one of two things:

1. Use the Settings -> Integration Settings -> Sparse Integrator Output option to only output data at the start and end of each event. This will improve script execution performance at the expense of the quality of the trajectory plot.
2. Start a new LVD mission case, transferring the final state over as the new initial state of the new mission case.

This tutorial will assume that neither of these of these modifications are required.

3.6 Eve Ascent and Departure Trajectories

Our ascent and departure trajectory will be modeled in a manner identical to that of the Kerbin ascent and departure.

Let us create four events. The first will begin throttling up the vehicle. The second will release the hold down clamps at full throttle. The third will propagate to 30 meters above our current altitude in order to clear terrain and begin our roll to launch azimuth. And the fourth will terminate the ascent roll and begin the pitch profile to orbit.

3.6.1 Throttle Up to Launch Azimuth Roll

Title the event "Eve First Stage Throttle Up".

Set the Termination Condition to "Event Duration". Set the initial value to 0.0 seconds.

Set the line color from Red to Blue if you desire.

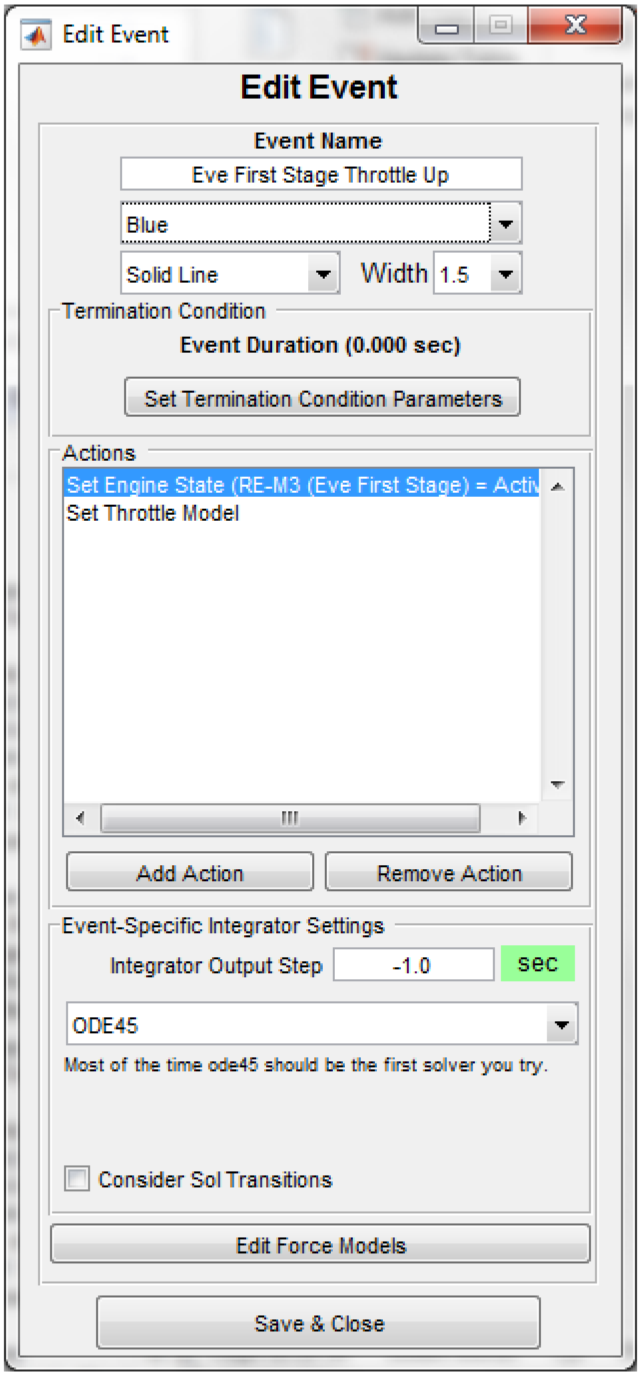
Finally, let's add an action that will activate the first stage engines and begin throttling up.

Click the "Add Action" button. A list dialog will appear. Find "Set Engine Active State" in the list and double click it. A dialog box will appear with a list of engines. Find the Eve First Stage engine, select it, and then set the engine state to "Active." Tap Save and Close when you are done.

Add another action, this time a "Set Throttle Model" action. A dialog box will appear asking you what type of model you wish to add. Select "Polynomial Model." In the "Edit Action" dialog box that appears, set the throttle Constant Term to 0.0 %, the Linear Term to 200 %/s, and the Accel. Term to 0.0 %/s/s. Save and Close when you are done.

Finally, as usual, uncheck the "Consider Sol Transitions" checkbox.

Save and Close the Edit Event dialog box.



Event Name: Eve First Stage Throttle Up
Blue
Solid Line
Width 1.5
Termination Condition
Event Duration (0.000 sec)
Actions
Set Engine State (RE-M3 (Eve First Stage) = Activ
Set Throttle Model
Event-Specific Integrator Settings
Integrator Output Step -1.0 sec
ODE45
Most of the time ode45 should be the first solver you try.
☐ Consider Sol Transitions

Figure 33: Eve First Stage Throttle Up event

Create the second event.

Title the event “Eve Propagate to Full Throttle”.

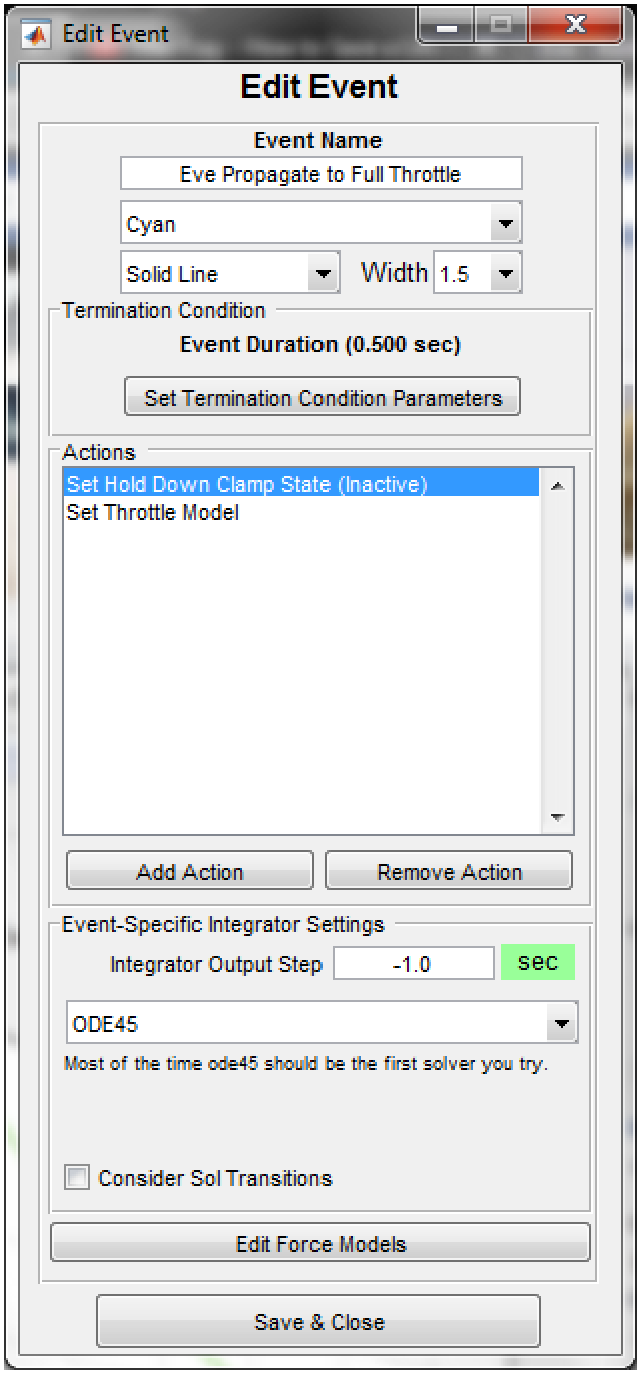
Set the Termination Condition to “Event Duration”. Set the initial value to 0.5 seconds.

There are two things we want to do at the end of the 0.5 second period. First, we want to release the hold down that is keeping our rocket attached to Eve. Add another event, “Set Hold Down Clamp State.” In the dialog box that appears, set the state to Inactive.

Second, we want to set the throttle model to be a constant 100% thrust. Create another “Set Throttle Model” action, select the polynomial type, and set the constant term to 100% and the linear rate and acceleration to 0.0.

Finally, as usual, uncheck the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.


Edit Event

Event Name: Eve Propagate to Full Throttle

Cyan

Solid Line — Width 1.5

Termination Condition: Event Duration (0.500 sec)

Actions:
Set Hold Down Clamp State (Inactive)
Set Throttle Model

Event-Specific Integrator Settings

Integrator Output Step: -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider SoI Transitions

Figure 34: Eve Propagate to Full Throttle event

You should notice that the final spacecraft state is now 0.5 seconds later than the initial state. The actual position and velocity of the vehicle will not have changed (relative to the surface of Eve), though, because the hold down is released at the *end* of the event.

Insert the third event.

Title the event "Eve Propagate to 30 meters".

In the Termination Condition area, push the "Set Termination Condition Parameters" button. Set the termination condition type to "Altitude" in the dialog box. The altitude target should be 30 meters above your current altitude. So if you are landed at 11.80 km, then the target will be 11.83 km. Save and Close the termination condition dialog box when you are done.

The only event action we will want to add here is one which adds some yaw rate in order to steer our vehicle onto the correct launch azimuth to make our orbit target later.

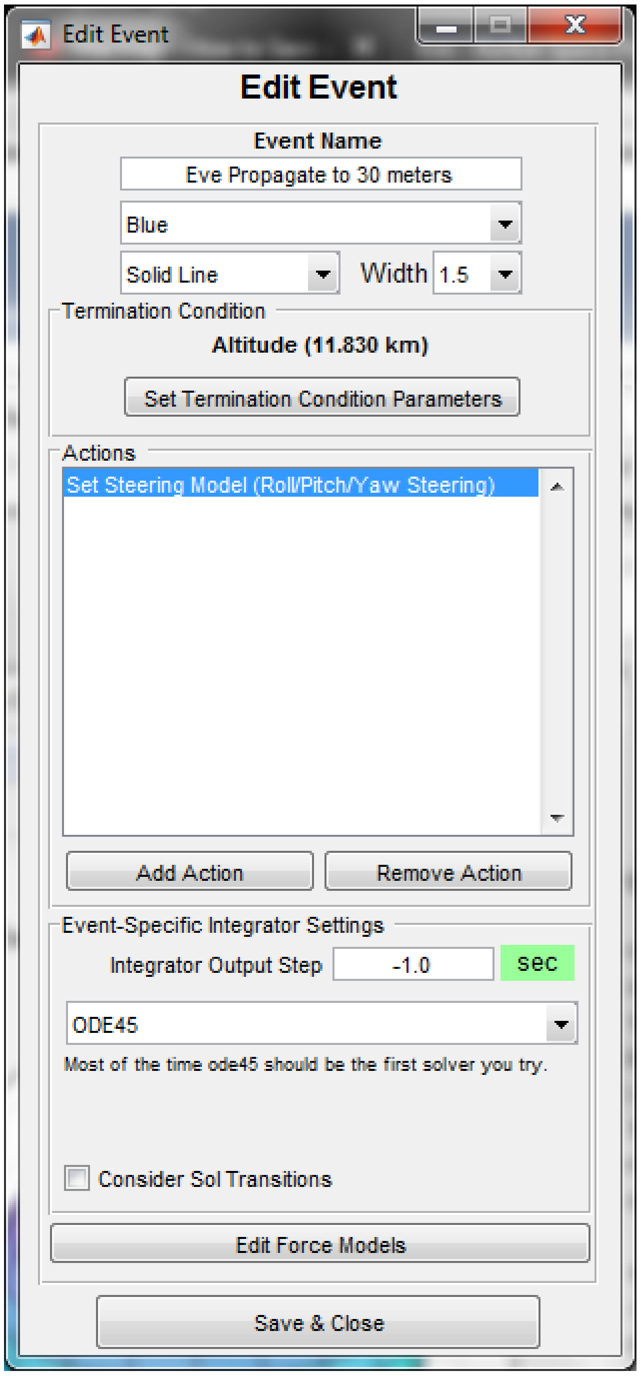
Add an action and select "Set Steering Model". Leave the Roll/Pitch/Yaw Steering as-is, and check the "Preserve Angle Continuity at Start of This Steering Model?" checkbox at the bottom. You'll notice that the constant terms of the various angles are greyed out. This is because this steering model will inherit those values from the attitude state of the vehicle.

Let's set the linear rate of the Yaw Angle to 30.0 deg/s.

Also, since at this point we are clear of the launch tower, we can begin pitching the vehicle downwards. Set the linear rate of the Pitch Angle to -1.0 deg/s. (NOTE: The sign on this angle is very important. It must be negative here.) Finally, check the "Opt?" checkbox next to the Linear Term and set the bounds on this variable to -2.5 to 0.0 deg/s.

Save and Close the Steering Model dialog box when you are done.

Finally, as usual, uncheck the "Consider Sol Transitions" checkbox.



Event Name: Eve Propagate to 30 meters
Blue
Solid Line   Width 1.5
Termination Condition
Altitude (11.830 km)
Actions
Set Steering Model (Roll/Pitch/Yaw Steering)
Event-Specific Integrator Settings
Integrator Output Step -1.0 sec
ODE45
Most of the time ode45 should be the first solver you try.
☐ Consider Sol Transitions

Figure 35: Eve Propagate to 30 Meters event

And finally, insert the fourth event of our Eve ascent start up sequence.

Title the new event "Eve Terminate Ascent Roll".

Leave the termination condition as "Event Duration" but set the time to 3 seconds. In the "Edit Termination Condition" dialog box, also check the "Opt?" box to optimize the duration of this event. Then, set the lower and upper bounds of the event duration to be 0.0 seconds and 6.0 seconds respectively. Remember that our initial yaw angle is 0.0 degrees, and so we are pointed north. A yaw at 30 deg/s for 3 seconds gets us down to 90 degrees, which is due east. The optimizer can figure out the actual desired launch azimuth from this initial guess.

Add an action to set the steering model again and check the box to preserve angle continuity.

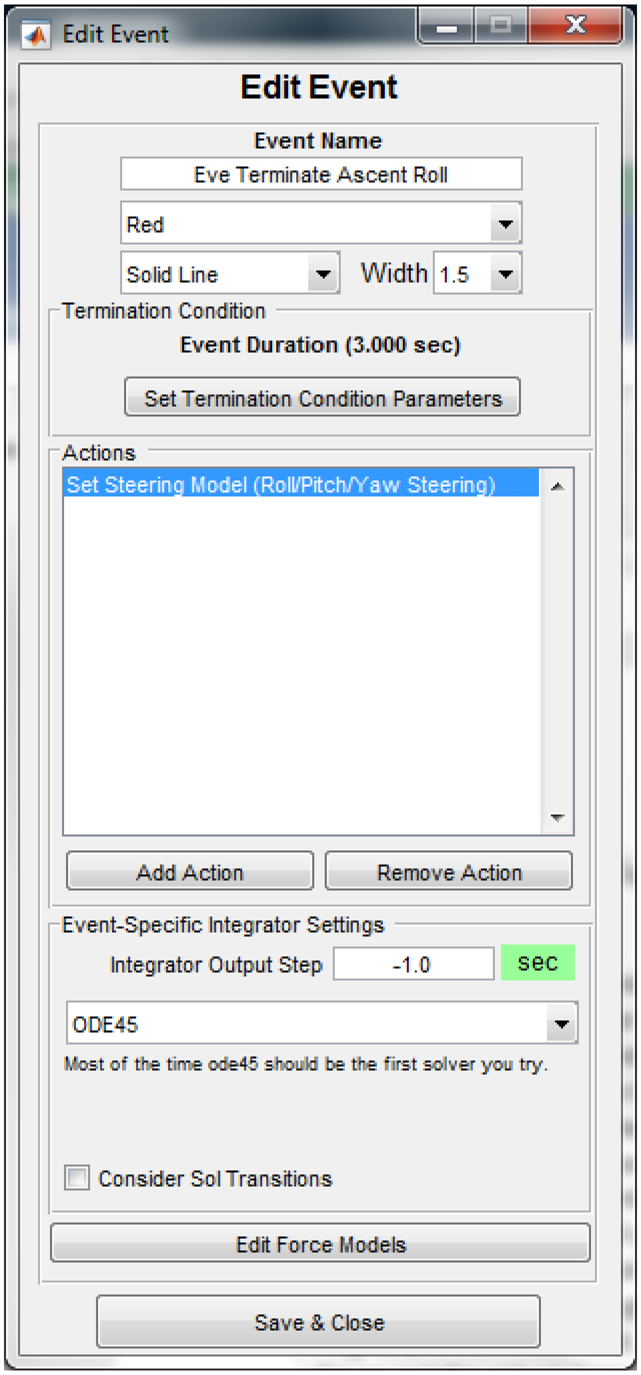
Set the Yaw Angle Linear Term to be 0.0 deg/s. This will terminate the roll.

Set the Pitch Angle Linear Term to be -0.1 deg/s. As before, tap the "Opt?" checkbox and set the bounds to -0.75 deg/s and 0.0 deg/s.

Save and Close the steering model dialog box.

Finally, as usual, uncheck the "Consider Sol Transitions" checkbox.

Save and Close the Edit Event dialog box.



Eve Terminate Ascent Roll

Red

Solid Line   Width 1.5

Event Duration (3.000 sec)

Set Steering Model (Roll/Pitch/Yaw Steering)

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☐ Consider Sol Transitions

Figure 36: Eve Terminate Ascent Roll event

3.6.2 Launch Azimuth Roll to Orbit Insertion

3.6.2.1 Setting up the Pitch Profile

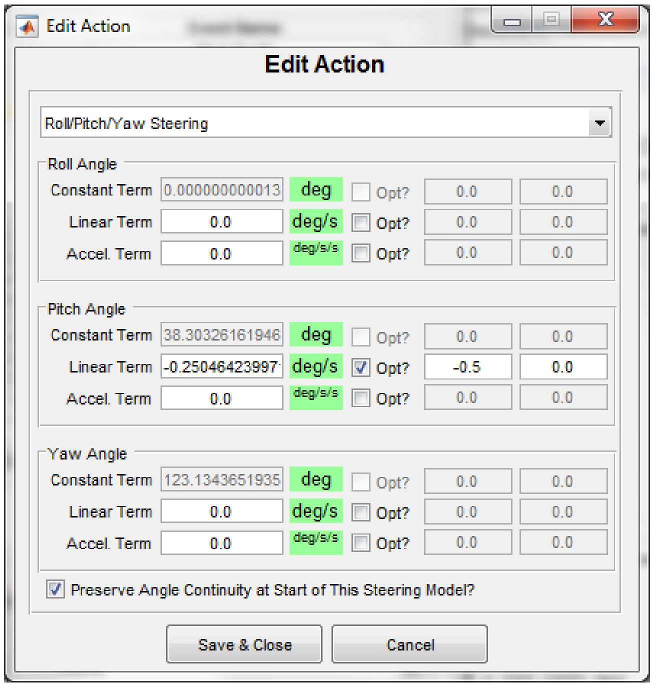
As with the Kerbin ascent, we will create seven events that represent the pitch profile to be optimized. Create the following event seven times. It should show up in the script event list as Events 20-26.

Table 18: Eve Pitch Profile Event Properties

Event Property	Value
Name	“Pitch Profile”
Termination Condition Type	Event Duration
Termination Condition Value	43 seconds
Termination Condition Bounds	5 to 150 seconds
Termination Condition Opt? Flag	Checked

Create a new “Set Steering Model” action for each of these events. In them, set the Pitch Angle Linear Term to -0.1 deg/s, check the Opt? box for that value, and set the bounds to -0.75 deg/s to 0 deg/s. Also make sure that the “Preserve Angle Continuity” checkbox is checked.

When you are done, your Steering Model dialog box will look something like this:



Roll/Pitch/Yaw Steering

Roll Angle

Constant Term	0.000000000013	deg	Opt?	0.0	0.0
Linear Term	0.0	deg/s	Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

Pitch Angle

Constant Term	38.30326161946	deg	Opt?	0.0	0.0
Linear Term	-0.25046423997	deg/s	✓ Opt?	-0.5	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

Yaw Angle

Constant Term	123.1343651935	deg	Opt?	0.0	0.0
Linear Term	0.0	deg/s	Opt?	0.0	0.0
Accel. Term	0.0	deg/s/s	Opt?	0.0	0.0

✓ Preserve Angle Continuity at Start of This Steering Model?

Figure 37: Eve Ascent Pitch Profile Steering Model Setup

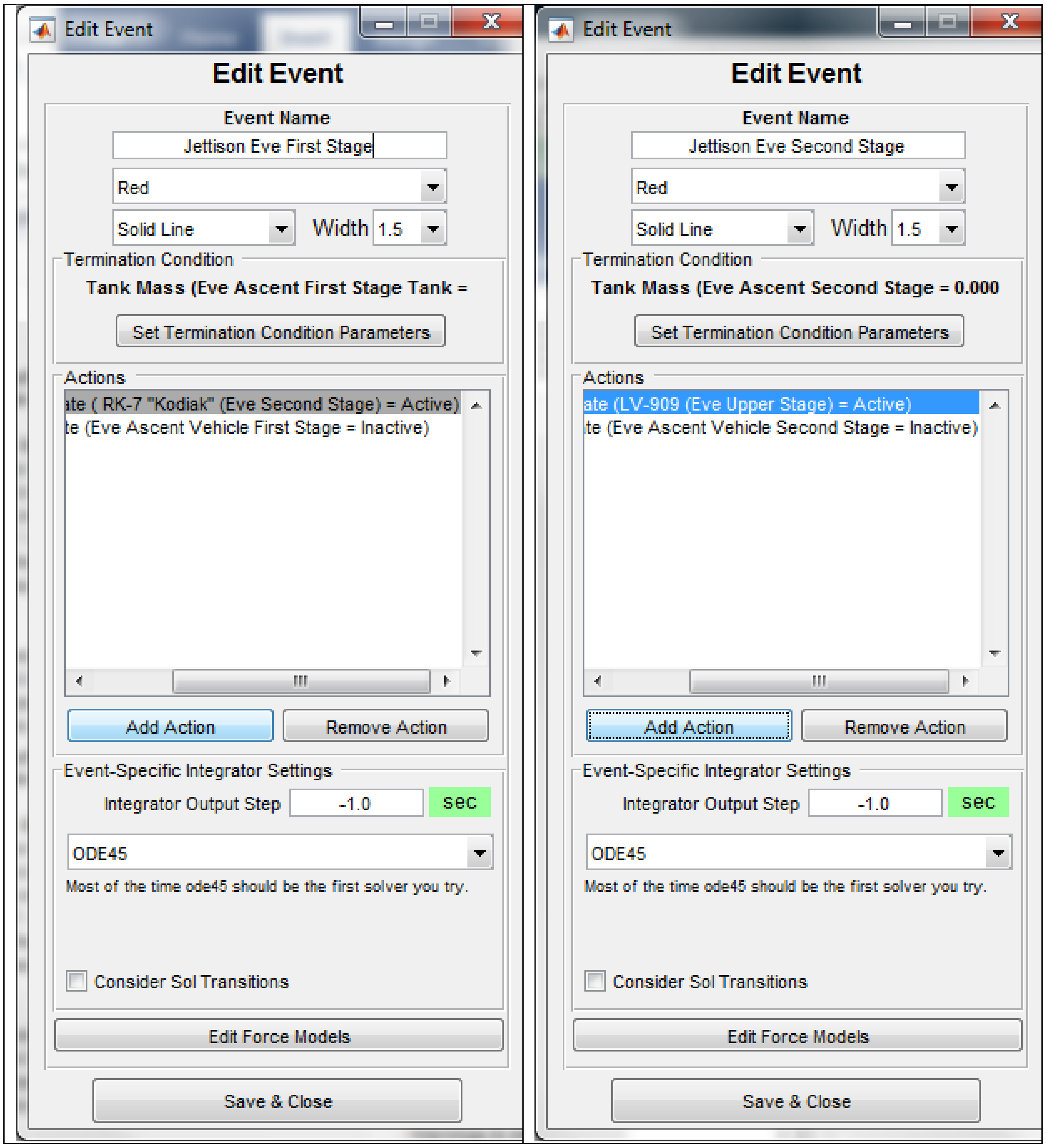
Save and Close all dialog boxes and return to the main LVD window.

3.6.2.2 Eve Ascent Staging

As with previously, we need to create two new non-sequential events to jettison the Eve ascent first and second stages. Do this now. The first new non-sequential event should deactivate the Eve first stage and activate the Eve second stage engines. The second new non-sequential event should deactivate the Eve second stage and activate the Eve third stage engines. Trigger both events off of their respective Eve first and second stage tank masses, triggering when they equal 0.0 mT remaining.

Both new non-sequential events should only activate once and you may leave the upper and lower event bounds for each empty.

Here is what your new non-sequential events should look like when you are finished.



Edit Event

Event Name: Jettison Eve First Stage

Red

Solid Line — Width 1.5

Termination Condition: Tank Mass (Eve Ascent First Stage Tank =

Actions:
- ate (RK-7 "Kodiak" (Eve Second Stage) = Active)
- te (Eve Ascent Vehicle First Stage = Inactive)

Event-Specific Integrator Settings

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

Consider Sol Transitions

Edit Event

Event Name: Jettison Eve Second Stage

Red

Solid Line — Width 1.5

Termination Condition: Tank Mass (Eve Ascent Second Stage = 0.000

Actions:
- ate (LV-909 (Eve Upper Stage) = Active)
- te (Eve Ascent Vehicle Second Stage = Inactive)

Event-Specific Integrator Settings

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

Consider Sol Transitions

3.6.2.3 Eve Ascent Engine Cutoff

Finally, we need to create one more event that will set the throttle to zero at the end of our ascent burn.

Title the event “Eve Ascent Engine Cutoff”.

Set the Termination Condition to “Event Duration” and set the duration to 0.0 sec. This will not be optimized.

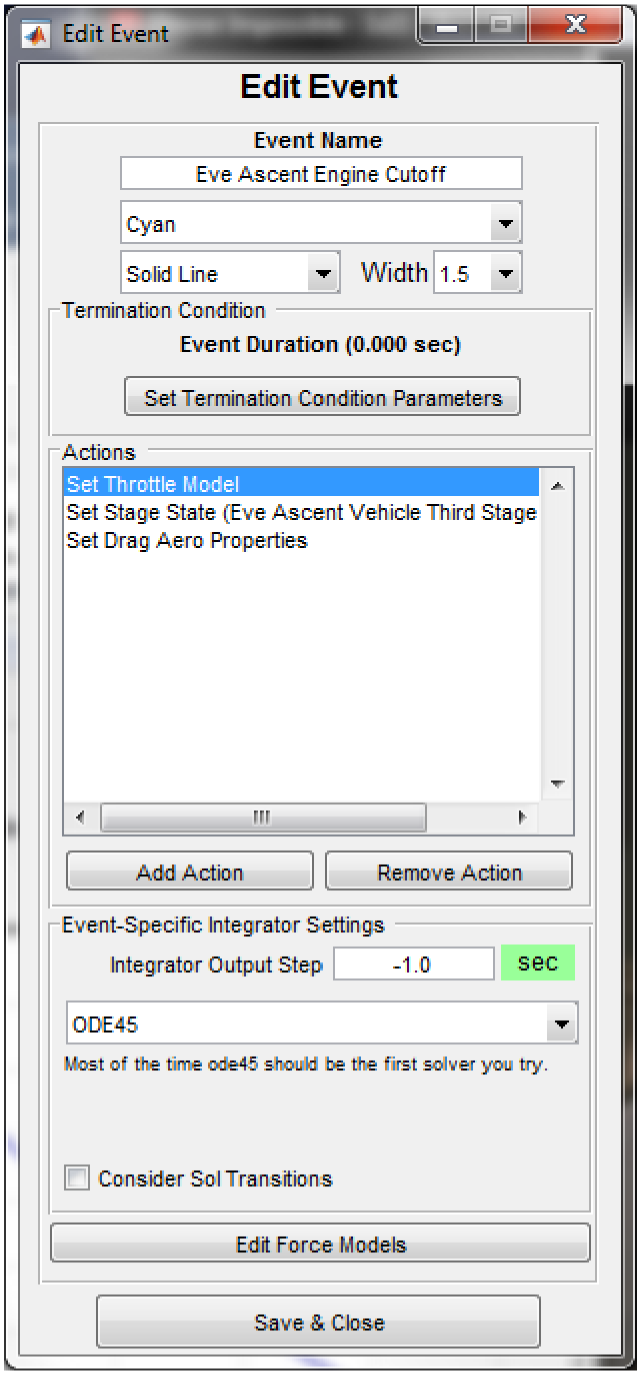
Now we create an event action that will set the throttle to zero. Create a “Set Throttle Model” action, select the polynomial type (as before), and set all values to 0.0. This represents an instantaneous change of throttle to 0.0.

Create another action to jettison the Eve Ascent third stage. Create a “Set Stage State” action, select the appropriate stage, and set it to “Inactive.”

Create a final action to update the drag aerodynamics to reflect the fact that we are now a 0.625-meter radius body. Set the drag coefficient to 0.2 and the area to 1.2271846 m^2.

Finally, as usual, uncheck the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.



Edit Event

Event Name: Eve Ascent Engine Cutoff

Cyan

Solid Line   Width 1.5

Termination Condition: Event Duration (0.000 sec)

Actions:
Set Throttle Model
Set Stage State (Eve Ascent Vehicle Third Stage
Set Drag Aero Properties

Event-Specific Integrator Settings

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

Consider SoI Transitions

Figure 38: Eve Ascent Engine Cutoff event

3.6.2.4 Optimizing the Eve Ascent

The first step in optimizing the ascent is to provide the optimizer with constraints that define our orbit at the end of the ascent. Use the "Optimization" menu and select "Edit Constraints."

First, open up each constraint that currently exists and deactivate it by unchecking the "Constraint Active?" checkbox.

Add four constraints with the following properties by tapping the Add Constraint button.

Table 19: Eve Ascent Target Constraint Definitions

Constraint Property	Constraint 1	Constraint 2	Constraint 3	Constraint 4
Constraint Type	*Altitude*	*C3 Energy*	*Hyp. Vel. Vect. R.A.*	*Hyp. Vel. Vect. Decl.*
Upper Bound	10000.0	0.951049377	243.6079	-20.0852
Lower Bound	91.0	0.951049377	243.6079	-20.0852
Scale Factor	100.0	1.0	100	10
Celestial Body	<blank>	<blank>	<blank>	<blank>
Applicable Event	Eve Ascent Engine Cutoff	Eve Ascent Engine Cutoff	Eve Ascent Engine Cutoff	Eve Ascent Engine Cutoff
Constraint Active?	Checked	Checked	Checked	Checked

As before, run the optimizer (Optimization -> Optimization Mission). This optimization will also take some time. Use the skills you acquired in optimizing the Kerbin ascent trajectory to successfully meet all of the constraints of the Eve ascent trajectory while minimizing fuel usage.

3.7 Interplanetary Coast Trajectory 2

As with our outbound trajectory from Kerbin to Eve, now that we've achieved the approximate ascent insertion targets, we must modify the trajectory slightly in order to actually hit Kerbin.

Let's create the event that will actually propagate us to Kerbin.

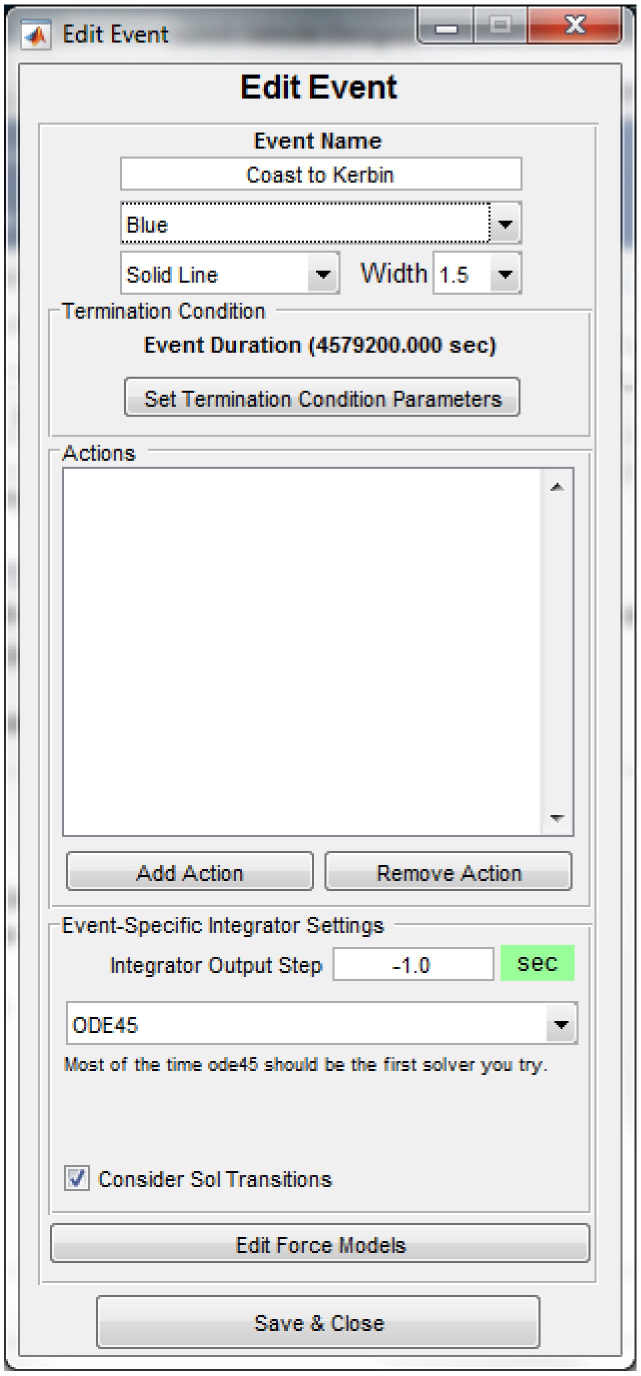
Title the event “Coast to Kerbin”.

Set the Termination Condition to “Event Duration”. Set the initial value to 53 days (or 4579200.0 seconds). Check the “Opt?” box and set the lower bound to 2160000.0 seconds (25 days) and the upper bound to 5184000.0 seconds (60 days).

Set the line color from Red to Blue if you desire.

Unlike the usual procedure, *check* the “Consider SoI Transitions” checkbox.

Save and Close the Edit Event dialog box.



Edit Event

Event Name: Coast to Kerbin

Blue

Solid Line   Width 1.5

Termination Condition

Event Duration (4579200.000 sec)

Actions

Event-Specific Integrator Settings

Integrator Output Step -1.0 sec

ODE45

Most of the time ode45 should be the first solver you try.

☑ Consider SoI Transitions

Figure 39: Coast to Eve event

Disable the constraints you created in Table 19, excepting the Altitude constraint which should be left active. Add another C3 energy constraint to be evaluated at the event “Eve Ascent Engine Cutoff,” and set its upper and lower bounds to 1000 and 0.001 km^2/s^2.

Finally, add a second constraint, “Distance to Ref. Celestial Body”. Set the body to “Kerbin” and the upper bound to 600.001 and lower bound to 599 km.

Set the objective function to be “Satisfy Constraints Only” on the “Coast to Kerbin” event.

Run the optimizer. You may need to use some of the tricks and tips discussed when targeting Eve earlier in order to get this to converge. Continue trying until you get it.

3.8 Kerbin Arrival Trajectory

Now that you have a trajectory that intersects Kerbin, add the following constraints. This correspond to the values shown in Table 9.

Table 20: Kerbin Arrival Constraints

Constraint Property	Constraint 1	Constraint 2
Constraint Type	*Latitude*	*Longitude*
Upper Bound	0.0	310
Lower Bound	-20.0	290
Scale Factor	1.0	1.0
Celestial Body	Kerbin	Kerbin
Applicable Event	Coast to Kerbin	Coast to Kerbin
Constraint Active?	Checked	Checked

Re-optimize the trajectory once more. As with the similar trajectory to arrive at Eve, this optimization may be difficult and require a number of runs. Use the tips and tricks discussed previously to get the trajectory to converge and satisfy the constraints.

Congratulations! If you’ve made it this far, you’ve successfully put together an end-to-end Eve sample return trajectory.

4 Additional Challenges

After you've completed the first part of this tutorial, here are some additional things you can add to your Eve sample return mission trajectory that will increase the modeling fidelity.

4.1 Payload Fairings

Encapsulate the Sample Return Capsule in a payload fairing for Kerbin ascent.

1. Add a stage called "payload fairing". Set the dry mass to some reasonable value that can be obtained from KSP.
2. Create a non-sequential event to "jettison" the payload fairing at a dynamic pressure of 0.05 kPa.

4.2 Eve Parachute Deploy

Deploy parachutes during Eve ascent to slow the Eve ascent vehicle down prior to touchdown on Eve.

1. Insert an event after the "Coast to Eve" event. Call it "Coast to Eve Surface."
2. Alter the "Coast to Eve" event to propagate to an altitude of 13 km above the surface of Eve. You may need to actually create a few new events to do this, a few "Next SoI transition" events to get to Eve's SoI and then the Go to Altitude event.
    a. Rename the event to "Coast to Eve Parachute Deployment".
    b. Add an action to change the drag aerodynamics.
        i. Change the drag coefficient to 1.5;
        ii. Change the drag area to 5 m^2 (or pick a number).
    c. Eliminate the "Hold Down" clamp state action.
3. Alter the new "Coast to Eve Surface" event.
    a. Add a "Set Hold Down" action and turn on the hold down clamps.