AIAA 2016-1286

AIAA SciTech
4-8 January 2016, San Diego, California, USA
AIAA Atmospheric Flight Mechanics Conference

Trajectory Scripts for Aircraft and Spacecraft Flight Path Analysis

Daniel P. Raymer[1]
Conceptual Research Corporation
PO Box 5429, Playa del Rey, CA, USA, 90296-5429

This paper describes a control scheme used to quickly define a complicated trajectory for analysis of aircraft and spacecraft operations for which closed-form equations are not suitable. A "Trajectory Script" is a sequence of event-driven flight control commands that define a trajectory and "fly" the vehicle during simulation. A Trajectory Script can be developed in less than a minute, and can be changed and rerun in seconds. The method is implemented in the ROAST trajectory module of the RDS^{win} aircraft design software, but could be applied to any time-dependent trajectory analysis. The paper describes the creation and operation of Trajectory Scripts, and discusses specific coded algorithms essential to implementing the concept in a time-domain trajectory simulation program.

Nomenclature

AOA	= Angle of Attack
L/D	= Lift-to-Drag Ratio
M	= Mach Number
OTIS	= Optimal Trajectories by Implicit Simulation (trajectory code)
POST	= Program to Optimize Simulated Trajectories (trajectory code)
RDS^{win}	= Aircraft design software package *("Raymer's Design System")*
ROAST	= RDS Optimal AeroSpace Trajectories
T/W	= Thrust-to-weight ratio
W_o	= Aircraft Takeoff Gross Weight
W/S	= Wing loading (weight/area)

I. Introduction

There is a need for a rapid method of defining and modifying trajectories for time-dependent analysis of aircraft maneuvers and spacecraft launch and recovery. Normal aircraft missions are readily modeled in an analytical domain, as typified by the Breguet range equation and similar mathematical approaches. These aren't usable for complicated trajectories such as a pull-up, climb, parabolic pushover, and recovery to level flight as might be used by a carrier aircraft releasing a Pegasus-like launch vehicle. No equation can directly model this – it takes a simulation.

For launch vehicles, a simple vertical launch trajectory can be modeled by a time-dependent code using AOA or pitch rate optimization to attain a target flight path angle at engine cutoff. However, many launch vehicles trajectories will require complicated guidance logic that depends upon events determined "on the fly."

For example, the Pegasus-like launch vehicle might be dropped from its carrier aircraft at a certain speed and altitude, coast for 5 seconds while the carrier banks away, start its motors and accelerate in level flight until a certain speed is reached, pull up at 2 g's until a desired climb angle is reached, climb at that angle until free of the atmosphere, then arc over to a desired burnout angle. Again, this takes a simulation.

These sorts of trajectories can be modeled in a high-end code like POST or OTIS, but it takes an expert a considerable amount of time to define the trajectory and guidance logic. Alternatively, a complicated aircraft or

[1] President, Conceptual Research Corporation. AIAA Fellow.

spacecraft trajectory can be "flown" by the analyst in a real time simulation, but that isn't likely to be optimal nor is it exactly repeatable. Also, the results depend upon the piloting skills of the analyst.

So, there is a need for a scheme that lets you quickly build and run a complicated time-domain trajectory. The guidance logic should be defined in an intuitive, vehicle-related manner, and should be tied to an easy-to-build flight vehicle data file that can be readily changed for trade studies.

A method called "Trajectory Script" was developed to fill this need. It has been implemented in the ROAST trajectory module of the RDS^{win} aircraft design and analysis software, but could be applied to any time-dependent trajectory analysis. This methodology will be discussed in the remainder of this paper.

The RDS^{win}-Professional[12] aircraft design software, developed by this author and marketed through Conceptual Research Corporation, is an integrated design environment which includes a design layout module for concept development, analysis modules for aerodynamics, weights, propulsion, stability, cost, performance, range, sizing, and optimization. The technical methods employed in RDS are largely based on those described in this author's textbook *Aircraft Design: A Conceptual Approach*[3].

ROAST is the flight simulation and launch trajectory analysis module of RDS^{win}-Pro. ROAST determines what an airplane or aerospace vehicle will do in response to various control inputs from an initial starting speed and altitude. For launch vehicles, it determines speeds and altitudes that can be reached with a given amount of fuel/propellant. The program outputs a large table (figure 1) detailing the simulation second-by-second, including speed, altitude, angle of attack, flight path angle, dynamic pressure, distance, weight, fuel burn, and more. ROAST also graphs the results including altitude vs. distance, velocity vs. time, and many more.

While deliberately not as sophisticated as the industry-standard POST and OTIS trajectory codes, ROAST has faired well in calibration comparisons to POST and is far simpler to set up and run. This makes it an ideal tool for initial evaluations and parametric trade studies, but it should not be relied upon for "final" answers.

II. Trajectory Script Overview

ROAST includes four automatic guidance schemes suitable for launch trajectories, namely Minimize AOA, Set AOA, Find Pitch Rate, and Set Pitch Rate. ROAST also allows the user to "fly" the vehicle in real time, using a joystick or arrow keys to control angle of attack. During real-time operation the user can also bring up a menu of autopilot guidance commands such as angle of attack hold or climb angle seek.

The Trajectory Script method allows specifying a sequence of those autopilot guidance commands. These are implemented in response to "triggers" based on what the vehicle is doing during its flight. Using a Trajectory Script, the guidance commands for the air-dropped Pegasus-like vehicle described above can be defined in just a few minutes, then run in ROAST in a few seconds more.

A Trajectory Script is just a text file. It can be created and modified with any text editor, including one as simple as MS NotePad. In the ROAST implementation of RDS^{win}, a new or existing Trajectory Script is opened in an editable pop-up box when beginning an analysis.

III. Trajectory Script Format

A Trajectory Script is a text file consisting of a series of "Event Triggers" paired with flight control commands. Triggers are greater-than or less-than tests applied to calculated flight parameters such as altitude, velocity, or load factor, or to vehicle status parameters such as fuel weight or drag. When a particular trigger is satisfied, its stated command becomes active. Examples include setting throttle or angle of attack, or tracking a commanded parameter such as climb angle or velocity.

A typical Trajectory Script is:

```
START SCRIPT: Units=fps
When Altitude>10000 Set AOA=-0.05
When Altitude>20000 Set throttle=80
When q-dynamic<100  Set throttle=999
When Thrust<0.01    Set AOA=0.0
When q-dynamic>500  Set S-turn=1.
END SCRIPT
```

A Trajectory Script begins with a line beginning "Start Script:" This also specifies the units of the parameters in the script ("Units=fps" or "Units=mks"). The vehicle's initial flight conditions can optionally be specified, such as:

```
START SCRIPT: Units=fps   Altitude=10k   FltPathGamma=20 Velocity=600
```

Note that RDS^{win} permits use of common aerospace abbreviations such as “k” and “counts” when inputting data.

IV. Event Triggers

Event Triggers are listed after the "Start Script" line and have the following format:

```
WHEN (event)<>(number)  SET (control)=(number)
```

Event triggers can be set for any of the items seen in the ROAST printout data columns, namely:

- Time
- Altitude
- Range
- Velocity
- M#
- AOA
- Gamma   (climb path angle)
- PitchRate
- V-hor
- V-vert
- Weight
- Fuel
- Thrust
- Drag
- Lift
- n-lift (load factor due to aero lift)
- nX-Accel (total acceleration load factor)
- nZ-Accel
- q-dynamic
- q-alpha (dynamic pressure times AOA)
- Delta-V
- EnergyHt
- Aero Heating (Chapman estimate)

An Event Trigger is active until it is satisfied, at which time the given Control is executed and the next Event Trigger is read and becomes active. Controls which have been set by an Event Trigger are continued to the end unless superseded by another Event Trigger.

Triggers are always greater-than or less-than. There are no equality triggers because they are unlikely to be exactly met. Instead, a desired trigger such as [When Altitude=20000] can be accomplished by using [When Altitude >19999].

To set a control that becomes active at the start of the simulation, a trigger such as [When Time>-1] can be used. For a command to take effect upon Burnout, a trigger such as [When Thrust<1] can be used.

Multiple triggers can be defined using "OR" such as:

```
When Time>50 OR Altitude>20000 Set AOA=0.5
```

Event Triggers can be defined incrementally using "MORE", such as:

```
When time>60 More Set AOA=0
```

Once the previous event trigger has been satisfied, this event trigger will watch until 60 more seconds have passed, and then do its command.

V. Trajectory Script Controls

Once an Event Trigger has been satisfied, its Control is set to the value given. These either directly set a control such as throttle, or specify a target for the autopilot to attempt to implement. Once activated by satisfaction of an Event Trigger, a Control command is followed until superseded by another Control. Available Controls include:

- AOA
- PitchRate
- Throttle setting
- Climb Path Angle (gamma)
- Rate of Climb
- n-lift
- nZ-Accel
- Climb Speed
- Level Flight Speed
- Re-entry S-turn Load Factor

Most Controls work by calculating and setting the vehicle Angle of Attack, redoing it for every time step (4 times a second in ROAST). Pre-defined limitations on q, M, n-Axial, n-Lateral, and AOA are active, and may result in a reduction in commanded AOA. The AOA is also restricted by CLmax if dynamic pressure is non-trivial. Finally, the commanded AOA will be adjusted to comply with a pre-defined maximum pitch rate (normally 10 deg/sec).

The Level Flight Speed Control adjusts the Throttle Setting as required, and the Throttle Setting Control obviously sets it directly. Otherwise the commands leave the previous Throttle Setting unchanged.

VI. Control Logic And Coding

Most of the Trajectory Script Controls are fairly obvious and simple in their coding. It's the overall concept that makes the method so powerful. The actual algorithms are described below, along with some pseudocode snippets. The term "ScriptDoValActive" is the numerical value read from the Script Control:

Throttle: This Control directly changes the Throttle Setting as given in percent of maximum thrust available at that speed and altitude.

```
THROTTLE=ScriptDoValActive
```

AOA: This directly sets Angle of Attack, whereupon the code checks that it does not result in excess load factor, stall, or excessive angle versus the given maximum AOA. Also, change in AOA per unit time step is limited to an input maximum rate.

```
AOA = ScriptDoValActive * deg2rad
```

PitchRate: Target pitch rate is set to the given value. AOA is then adjusted each time step to make it so.

```
PitchRate = ScriptDoValActive * deg2rad
AOA = AOA - DeltaClimbAng  + PitchRate  * DeltaT
```

n-lift: This sets a target aerodynamic lift load factor in the direction perpendicular to the velocity direction. AOA is adjusted each time step to make it so.

```
CLift  = CLmxMARG * CLMAX
znPullup = ABS(ScriptDoValActive)
IF CLift > znPullup * WS/Q THEN CLift = znPullup * WS/Q
IF Q > 10 THEN AOA = CLift/CLALPHA  ELSE AOA = AOAscriptMax
IF AOA  > AOAscriptMax THEN  AOA = AOAscriptMax
```

nZ-Accel: This sets a target acceleration in the direction perpendicular to the velocity vector including aerodynamic lift, thrust, and the weight vector, and AOA is adjusted to make it so. Note that in wing-borne level flight, n-lift equals one (1g) whereas nZ-Accel=0 since the vehicle is not accelerating upwards.

```
IF Q > 10 THEN  AOA = CLmxMARG * CLMAX/CLALPHA
            ELSE   AOA = AOAscriptMax
IF AOA  > AOAscriptMax THEN  AOA = AOAscriptMax
AOA = AOA  * ibSign
GOSUB SetLiftCoef
Fz = T * SIN(AOA ) + LiftCoef * Q * Sref - WtEff * COS(ClimbAng )
Accelz = Fz * gConst/WtNow
  (then iterate to find AOA that matches set value)
```

S-turn: This gives several ways of controlling the vehicle during re-entry. If not set, re-entry skipping is permitted to occur. If set to a value greater than 1.0 the re-entry skipping is suppressed and the vehicle performs an S-turn at that load factor value. If set to exactly 1.0, skipping is suppressed and the vehicle reenters and bleeds off speed in level flight.

```
IF Q > 10. THEN  AOA = CLmxMARG * CLMAX/CLALPHA
            ELSE   AOA = AOAscriptMax
znPullup = 1.
IF ABS(ClimbAng  * rad2deg) > 5. THEN
   IF ClimbAng  * rad2deg  > 5.  THEN  znPullup = xNpushover
   IF ClimbAng  * rad2deg < -5.  THEN  znPullup = 1./xNpushover
    AOA =AOA  * znPullup
    IF AOA  > CLmxMARG * CLMAX/CLALPHA  THEN
            AOA = CLmxMARG * CLMAX/CLALPHA
            EXIT SELECT
            END IF
GOSUB SetLiftCoef
Accelz = LiftCoef * Q * Sref
  (then iterate to find AOA that matches set value)
```

The next four are too complicated to show here as pseudocode, but represent fairly straightforward programming. The resulting AOA controls are faded in over time to avoid flight path "bobbles."

Gamma: Target climb path angle is set, holding the current throttle setting. AOA is adjusted each time step to make it so.

ClimbRate: Sets target rate of climb, i.e., vertical velocity. Throttle is unchanged. AOA is adjusted each time step to make it so.

Climb Speed: This sets the target vehicle velocity during climb. Throttle is unchanged. AOA is adjusted to reach Gamma that makes it so.

Level Flight Speed: Sets target vehicle velocity in level flight. AOA and throttle are adjusted to make it so, namely horizontal speed with Gamma=0.

VII. Sample 1: Air-Launch Carrier Aircraft

Two related examples are provided based on a carrier aircraft doing a zoom climb to launch an orbital booster. An advanced transport concept, previously developed by the author for a NASA study[4], was assumed to be the platform for releasing a notional reusable launch vehicle. The study question of interest is, "Would a pullup-to-launch maneuver be a net improvement vs. a level flight launch as used form the OSC Pegasus launch from the L-1011?"

First, the advanced transport concept was modified with the weight and drag of the notional "StarCar" upper stage (**Error! Reference source not found.**). The Trajectory Script below was defined to start in level flight at 850 fps

(~M.82), pull up to a 20 degree climb angle, hold it until speed drops below 600 fps (~M.6), push over, descend to initial altitude, then level out:

```
START SCRIPT: Units=fps Altitude=20k  FltPathGamma=0  Velocity=850
when time>-1           Set aoa=5
when time> 2           Set n-lift=1.5
When Gamma>18     Set Gamma=20
When Velocity<600 Set n-lift=.5
When Gamma<-9      Set Gamma=-10
When altitude<20k   Set gamma=0
END SCRIPT
```

The results from this simple script are shown in figure 3. The vehicle quickly attains the target climb path angle of 20 degrees and holds it for about 30 seconds, as speed bleeds off. Given a reasonable time allowance for launch preparation, the launch would take place at 27,000 feet at 600 fps (356 kts).

When a velocity drops to 600 fps, the autopilot begins a pushover which continues until a 10 degree downward flight path angle is reached. Minimum speed during pushover is about 556 fps (330 kts) which is safely above stall speed at that altitude.

Descent continues until the vehicle returns to its initial altitude of 20kft, whereupon level flight is commanded and tracked. However, once in level flight the velocity continues to drop, indicating that the thrust of the transport isn't enough to maintain the initial velocity with the StarCar on top.

VIII. Sample 2: Air-Launched Launch Vehicle

The second example continues the first example, now following the launch trajectory of the StarCar as it comes off the carrier aircraft. Note that the analysis below is based upon an assumed mass fraction which is probably better than what could be obtained with today's technologies!

The Trajectory Script starts in a 20 degree climb at 27,000 feet and 356 kts. It sets initial angle of attack to 10 degrees, which was previously determined to produce a slight pull-up at launch speed.

When the vehicle has accelerated past Mach 0.9, a 1.5g pull-up is initiated which continues until a climb path angle of 50 degrees is reached. Note that to avoid an overshoot, the autopilot is instructed to begin tracking that angle before it is reached.

Once past 60kft, the angle of attack is set to a slight positive value of half a degree. Attempting to track a certain climb path angle once out of the atmosphere can result in extreme angles of attack. The half degree angle was originally set to one degree then iterated to find the angle that resulted in burnout at just over zero flight path angle, maximizing horizontal velocity for launch of an orbital upper stage.

Once all propellant is exhausted, angle of attack is set to zero for the ballistic arc in space. On the way down, the autopilot is instructed to level out and bleed off speed, starting at 80kft.

```
START SCRIPT: Units=fps Altitude=27k  FltPathGamma=20 Velocity=600
When time>-1              Set AOA=10
When M#>.9                Set n-lift=1.5
When gamma>45          Set gamma=50
When Altitude>60000   Set AOA=.5
when thrust<1              Set AOA=0
when Altitude<80000    Set S-turn=1
END SCRIPT
```

Results are shown in **Error! Reference source not found.**. The vehicle quickly attains the target climb angle and accelerates nearly vertically, reaching a maximum speed of about 10,000 fps (the round number is just a coincidence). As mentioned above, the AOA of half a degree was iterated to obtain a nearly-horizontal flight path at burnout. Conditions at apogee are 9,969 fps at 217 kft.

Trajectory Scripts were invented specifically for rapidly defining trade studies. For this example, the interesting comparison is a horizontal release from the carrier aircraft rather than the zoom-climb release analyzed above.

Prior to the zoom-climb, the carrier aircraft is at 20,000 ft in level flight. When launched from this initial condition, the StarCar was found to reach an apogee of 9,855 fps at 236 kft.

For comparison, the zoom-climb release was iterated, changing AOA until a 236 kft apogee was reached. This resulted in a 9,919 fps apogee, slightly better than the level flight launch but probably not worth the extra trouble and risk of a zoom-climb launch.

It took literally 10 seconds to change the Trajectory Script for level launch and rerun it, and less than a minute to iterate to find the equivalent apogee.

IX. Summary & Conclusions

Trajectory Scripts use a sequence of event-driven flight control commands to define a trajectory and “fly” the vehicle. They are an easy, quick, and user-friendly way to define trajectories for which closed-form performance equations are not suitable, and are most-useful for quick analysis and conceptual trade studies.

Implemented in the RDS^{win} ROAST code, Trajectory Scripts could be applied to any time-domain simulation of aircraft flight path or rocket vertical launch.

Figures:

Time	Altitude	Range	V	M	AOA	Gamma	Weight	Fuel	Thrust	Drag	Lift	n-Lateral	n-Axial	Q	Delta-V
(sec)	(ft)	(nmi)	(ft/sec)		(deg)	(deg)	(lbs-m)	(lbs-m)	(lbs-f)	(lbs-f)	(lbs-f)			(lb-f/sqft)	(ft/sec)
1	7.1	0	11.4	0.01	0	90	61130.1	298.9	82905.5	4.2	0	0	0.36	0.1	32.8
2	25.8	0	23	0.02	0	90	60831.2	597.8	82911.1	5.8	0	0	0.36	0.5	63.8
3	56.2	0	34.9	0.03	0	90	60532.3	896.8	82920.8	14.6	0	0	0.37	1.2	95
4	98.6	0	46.9	0.04	0	90	60233.3	1195.7	82934.7	27.6	0	0	0.38	2.3	126.6
5	153.1	0	59.2	0.05	0	90	59934.4	1494.6	82953	45.1	0	0	0.38	3.7	158.5
6	220.1	0	71.7	0.06	0	90	59635.5	1793.5	82975.6	67.2	0	0	0.39	5.6	190.8

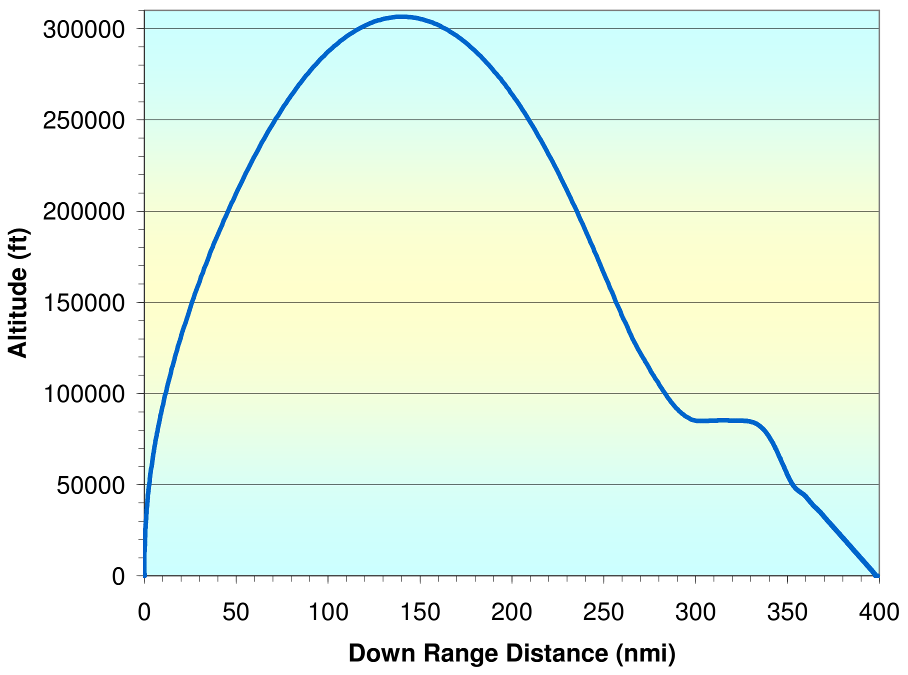


figure 1.   Typical ROAST Output Data and Graph

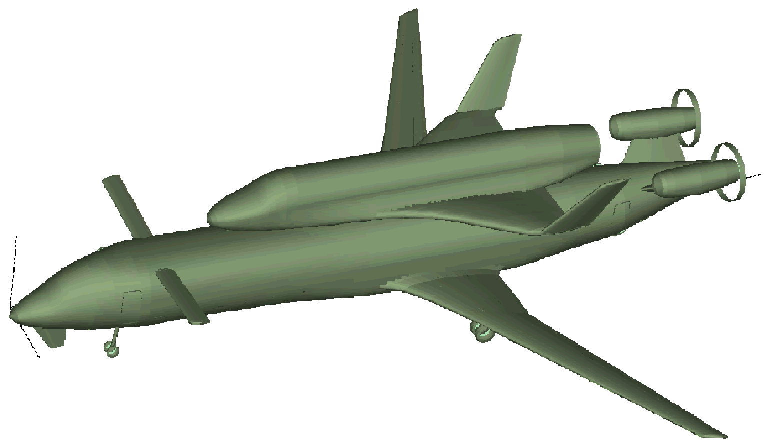
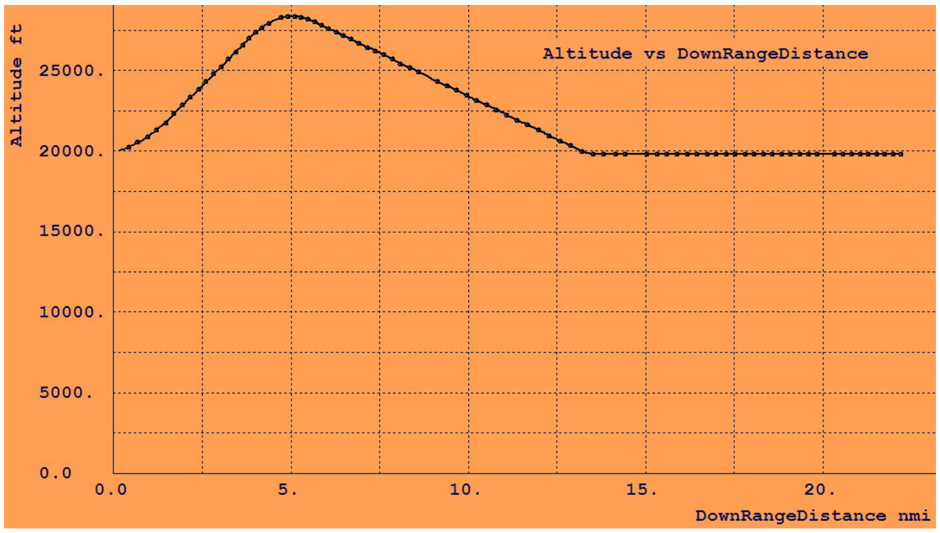
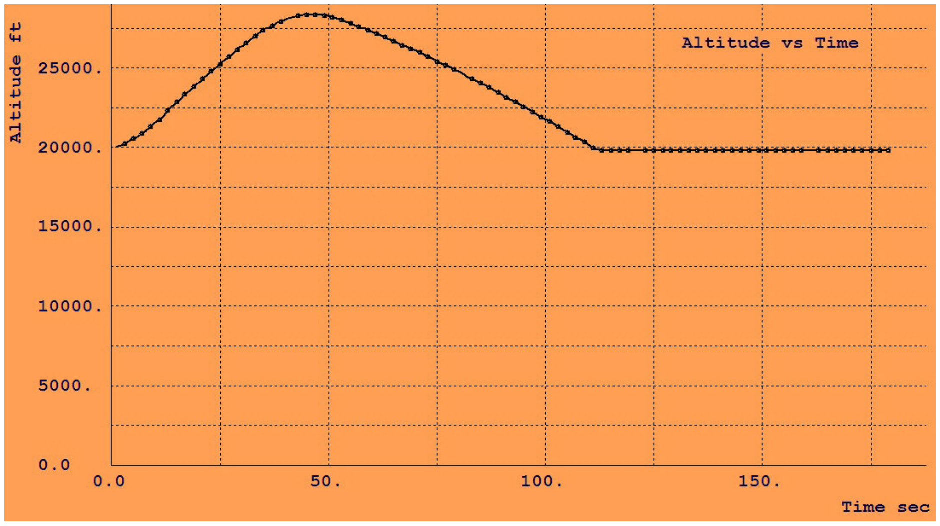


figure 2.   Advanced Technology Transport with “StarCar” Launch Vehicle





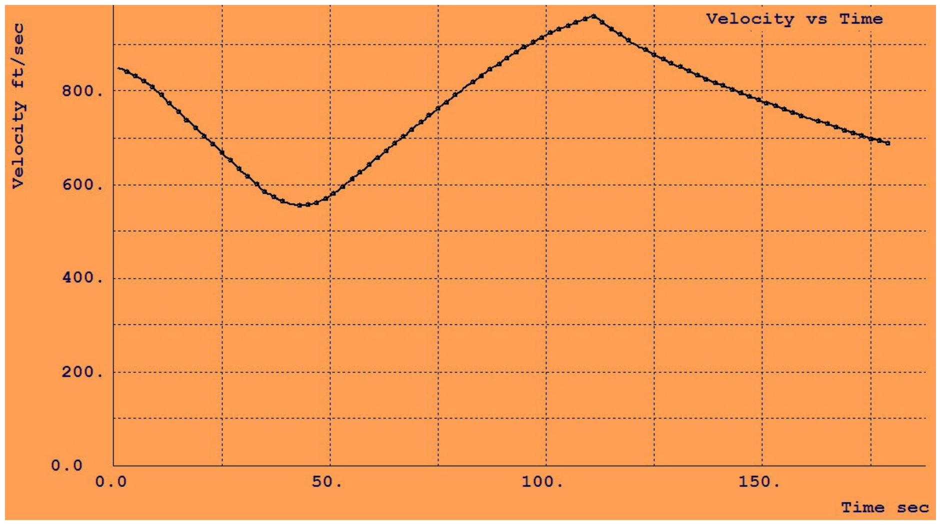


figure 3. ATA+StarCar Trajectory Results

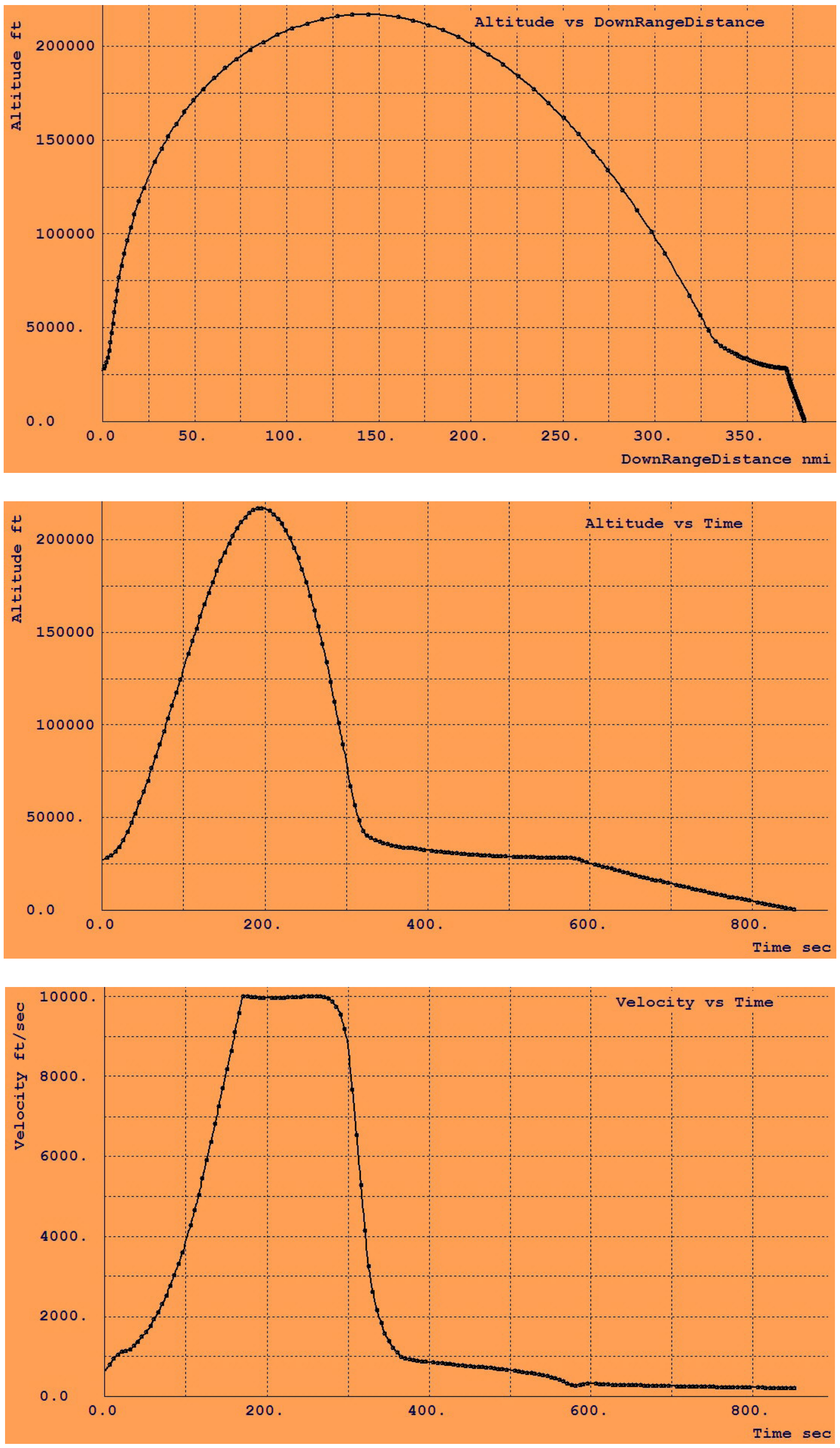


figure 4.   StarCar Trajectory Results

References

[1] Raymer, D., **"RDSwin: Seamlessly-Integrated Aircraft Conceptual Design for Students & Professionals"**, AIAA Paper 2016-(tbd), AIAA Aerospace Sciences Meeting, San Diego, CA, 2016 (submitted for publication)

[2] Raymer, D., **"RDS: A PC-Based Aircraft Design, Sizing, and Performance System,"** AIAA Paper 92-4226, Aug. 1992

[3] Raymer, D., ***Aircraft Design: A Conceptual Approach***, American Institute of Aeronautics and Astronautics, Washington, D.C., 1989 (5th Edition 2012)

[4] Raymer, D., et. al., ***Advanced Technology Subsonic Transport Study: N+3 Technologies and Design Concepts***, NASA/TM—2011-217130, NASA-GRC, Nov. 2011