Columbia International Publishing
Journal of Modeling, Simulation, Identification, and Control
(2014) Vol. 2 No. 1 pp. 31-44
doi:10.7726/jmsic.2014.1003
Research Article

Orbital Insertion Control of a Three-stage Solid Rocket Booster Modeled in Six Degrees-of-freedom

Peter H. Zipfel[1*]

Received 6 September 2014; Published online 11 October 2014

Abstract

How to model the orbital insertion of a payload in a high fidelity simulation is summarized in this paper. Though executed frequently in real time, actual flight software is inaccessible to a broader audience, and too complex for analytical studies. For exploratory trade studies of a three-stage solid rocket booster, the ascent guidance law is derived from first principles and embedded in an open-source six degrees-of-freedom simulation. This simulation, known by the acronym ROCKET6, is based on the C++ simulation framework CADAC++. It models, at high fidelity, aerodynamics, propulsion, autopilot, guidance, INS, GPS and star-tracker, while introducing real world atmospheric effects like wind and turbulence. The ascent of a typical booster from Vandenberg AFB, CA, is presented as test case and the results of 100 Monte Carlo runs are documented. The stochastic post-run analysis shows accurate insertion performance.

Keywords: Ascent Guidance; Orbital Insertion; Booster; Linear Tangent Guidance Law; Six Degrees-of-Freedom Simulation; Monte Carlo Methodology; Thrust Vector Control; Reaction Control System

1. Introduction

Inserting objects into space has become a routine undertaking by many nations. However, just recently we learned from The Wall Street Journal (2014) that the first two operational satellites of the European Galileo project were placed roughly two thousand miles off the intended positions. "An apparent problem with the *guidance system* of the upper stage caused both satellites to end up in what Arianespace described a noncompliant orbit."

The guidance system has the task to steer the booster autonomously from the launch pad to orbit. These guidance signals are translated by the autopilot into commands to the thrust vector control or the reaction control system, which, in turn, steers the main thrust vector towards the intercept.

*Corresponding e-mail: mastech.zipfel@cox.net
1* Modeling and Simulation Technologies

Already in the 1950', this guidance problem was the subject of extensive research, Perkins (1956). Ten years later Perkins (1966) remarked, "There are still many people engaged in trajectory, performance, and guidance work who do not understand its utility or validity." He is referring to the methodology as stated in the title of his report, "Derivation of Linear-Tangent Steering Laws."

The linear tangent guidance law (LTG) has been the foundation for all successive orbital insertion guidance. The first major application was for the Apollo program. Researchers at MIT Instrumentation Laboratory—now famous—gave lectures to NATO/AGARD on the application of LTG, see Draper, Wrigley, Hoag, Battin, Miller, Koso, Hopkins & Vander Velde (1965). The Space Shuttle ascent guidance is also based on LTG, Jaggers (1972), and even the NASA Orion program has the same linage, Thrasher and Fill (2011).

These papers, reports, and lectures expound on the analytical aspects of ascent guidance. The actual implementation is not accessible in the open literature (as far as I can determine.) However, there is a recent paper by Chavez and Lu (2009), which goes into some detail, but the implementation of the guidance law is limited to a three degrees-of-freedom simulation.

To create a realistic implementation, a full six degrees-of-freedom (6 DoF) simulation is required. This is my objective here, to derive the orbital insertion equations for a three-stage solid rocket booster and implement them in a high fidelity 6 DoF simulation. CADAC++, Zipfel (2012), is the framework in C++, and ROCKET6 is the name of the simulation. The code is 'open source' and contained in the software package CADAC4, which is bundled with the textbook, Zipfel (2014)—but ROCKET6 is not covered in the book itself.

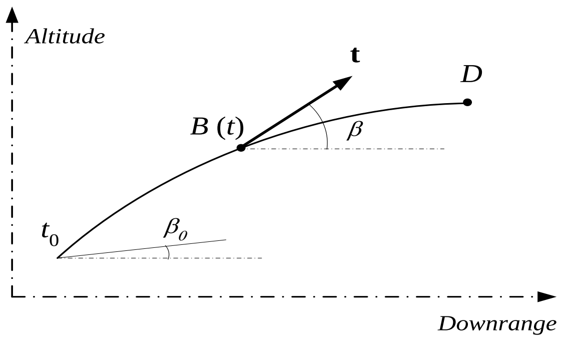
2. Ascent Guidance

The Wall Street Journal article points out that the orbit insertion of a spacecraft is still tricky business. The task is to shape the ascent trajectory such that the payload is released with the desired orbital parameters. It begins with the endo-atmospheric rise of the booster, aligned in the desired orbital plane. Once the booster leaves the atmosphere, the on-board guidance system controls the thrust vector such that the terminal conditions are met. Rather than using orbital elements, frequently, the terminal conditions are specified in terms of orbital position (distance from the center of earth), inertial velocity, and flight path angle. Note that the desired downrange is not part of the terminal conditions and cannot be specified.

For liquid rocket engines, which can be throttled, the thrust vector is regulated in both magnitude and direction. But for solid rockets, only the thrust direction and the *boost engine cut-off*, or BECO, can be controlled. Understandably therefore, the guidance task for solid boosters is more challenging.

The linear tangent guidance law is the basis for both liquid and solid boosters. It is obtained by solving the following two-point boundary value problem using the calculus of variations: given the initial conditions and the desired end conditions, minimize the time to intercept. As Bryson and Ho (1975) showed, the optimal solution requires that the thrust angle change linearly with time, under the assumptions of flat earth, constant gravity, constant thrust, and no aerodynamic forces.

With β the thrust angle, Eq. 1 of Fig. 1 shows the relationship of the tangent of β as a function of time *t*.



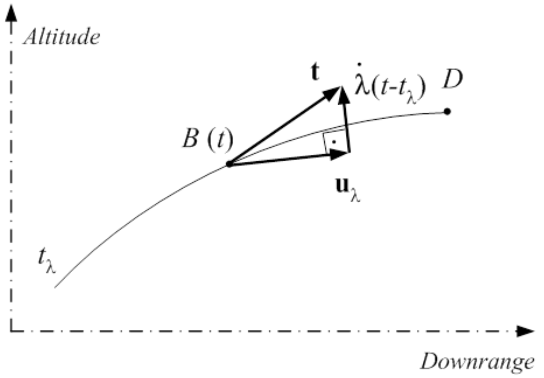
$$\tan\beta = \tan\beta_0 - c(t - t_0) \qquad (1)$$

β	thrust angle
c	constant
$\mathbf{t}$	thrust vector
B	current vehicle position

Fig.1. Linear tangent guidance law

The initial conditions at time t_0 are given by the location and the initial thrust angle β_0. Currently, the booster is at *B* with its thrust vector **t** and thrust angle β. It is to reach the desired end position at *D*.

To use the guidance law in a 6 DoF simulation, the planar LTG of Fig. 1 has to be extended to three dimensions. I replace the thrust angle β by the thrust vector **t** and formulate the LTG in three dimensions, as shown in Eq. 2 of Fig. 2.



$$\mathbf{t} = \mathbf{u}_\lambda + \dot{\boldsymbol{\lambda}}(t - t_\lambda) \qquad (2)$$

$\mathbf{u}_\lambda$	unit vector in direction of velocity-to-go
t_λ	reference time
D	desired end state
$\dot{\boldsymbol{\lambda}}$	turning rate of thrust vector; normal to $\mathbf{u}_\lambda$

Fig. 2. Three-dimensional linear tangent guidance law

The constant *c* of in Eq. 1 has been replaced by the turning rate vector lamda_dot in Eq. 2. This equation is the control law for placing the second and third stages of the booster into the desired end conditions. But because we are dealing with solid rockets, only the direction of **t** is of interest and I take therefore the unit vector of Eq. 2 as steering law

$$\mathbf{u}_t = unit\left\langle \mathbf{u}_\lambda + \dot{\boldsymbol{\lambda}}(t - t_\lambda)\right\rangle \qquad (3)$$

This steering law will be applied iteratively over the remaining trajectory.

3. Implementing the LTG Controller

The idealized assumptions for the LTG are never satisfied. The earth is not flat but modeled as the WGS84 rotating ellipsoid, gravity is decreasing by the inverse square law, and some residual aerodynamic forces are still present. To adjust for these changes, the LTG is solved iteratively, i.e., for every iteration, the starting condition is the current state, while the target state remains the same.

A predictor-corrector scheme is set up. Starting with the current state, the trajectory to the end-state is predicted, the miss recorded, and the trajectory corrected to eliminate the miss. The shape of the trajectory is assumed Keplerian and the direction of the thrust is given by the LTG steering law. Accurate time-to-go calculations have to be maintained to issue the BECO command at the right time. To describe the implementation of the LTG controller, I first have to introduce the salient coordinate systems.

3.1 Coordinate Systems

The six degrees of freedom equations of motion are solved in inertial coordinates, also called the J2000 system, Zipfel (2014). The first axis 1^I points at the Aries constellation, the third axis 3^I is parallel and in the direction of the earth's rotation vector, and the second axis 2^I completes the right-handed orthogonal coordinate system. Aerodynamic forces, thrust, and the moment of inertia tensor are expressed in body coordinates. The first body-axis 1^B is the tetragonal axis of symmetry of the booster, pointing through the nose. The second and third axes are normal to the first axis and complete the right-handed orthogonal coordinate system.

The formulation of the steering equations requires the definition of the flight–plane triad of base vectors $\mathbf{u}_d$, $\mathbf{u}_y$, $\mathbf{u}_z$, see Fig. 3.

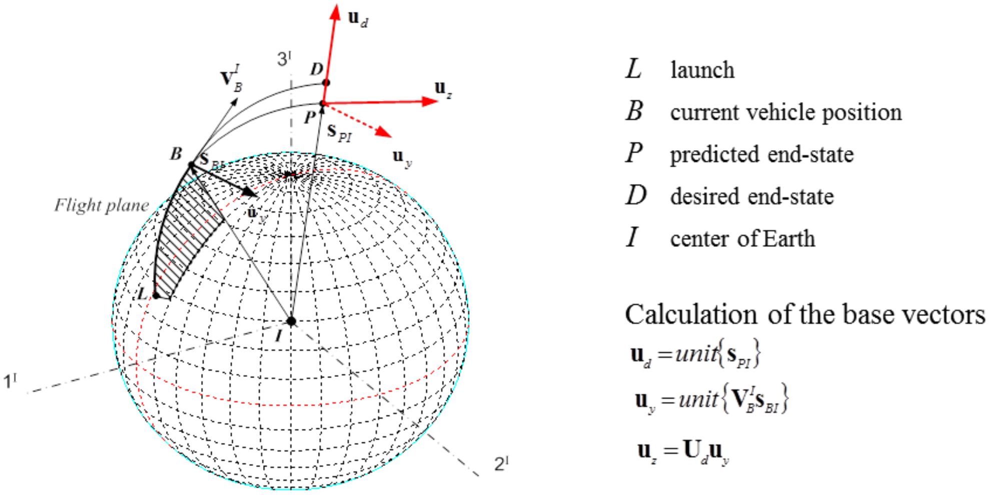


Fig. 3. Flight-plane base triad

The booster is currently at *B* with inertial velocity vector $\mathbf{v}^I_B$ and displacement vector $\mathbf{s}_{BI}$ from the center of the earth *I*. Forward projection of the trajectory predicts the target point to be at *P*, which is the base point of the base triad $\mathbf{u}_d$, $\mathbf{u}_y$, $\mathbf{u}_z$. The distance between *P* and *D* is the error that needs to be corrected for the next cycle of computation. (Note: $\mathbf{V}^I_B\mathbf{s}_{BI}$ is a vector product of the skew-symmetric form of $\mathbf{v}^I_B$ multiplied by $\mathbf{s}_{BI}$, similarly $\mathbf{U}_d\mathbf{u}_y$.) This triad will be used to express the velocity-to-be-gained and range-to-go vectors.

3.2 Integrating the Trajectory Equations

I follow the Long and McHenry (undated) NASA class notes on powered flight prediction to integrate the trajectory equations and thus obtain the LTG control parameters. According to Zipfel (2014) the translational equations are in tensor form

$$D^I D^I \mathbf{s}_{BI} = \frac{\mathbf{f}_p}{m} + \mathbf{g} \tag{4}$$

Reading from left to right: Twice the rotational derivative relative to the inertial frame *I* of the displacement vector of booster *B* relative to the center of the earth *I* equals the propulsive force vector divided by mass plus the gravitational acceleration. Eq. 2 gives the thrust vector, and is renamed here $\mathbf{f}_p$

$$\frac{\mathbf{f}_p}{m} = \frac{f_p}{m}\left(\mathbf{u}_\lambda + \dot{\boldsymbol{\lambda}}(t - t_\lambda)\right) \tag{5}$$

Expressed in inertial coordinates, the trajectory equations are now in matrix form

$$\left[\frac{d^2 s_{BI}}{dt^2}\right]^I = \frac{f_p}{m}\left([u_\lambda]^I + [\dot{\lambda}]^I (t - t_\lambda)\right) + [g]^I \tag{6}$$

Integrating from the current time to the end-time yields

$$\underbrace{\left[\frac{ds_{BI}}{dt}\right]^I_{\substack{Desired\\end\\state}}}_{[v^I_D]^I} - \underbrace{\left[\frac{ds_{BI}}{dt}\right]^I_{\substack{Current\\state}}}_{[v^I_B]^I} = \underbrace{\underbrace{\int_0^{t_{go}} \frac{f_p}{m} dt}_{L}\, [u_\lambda]^I + \left(\underbrace{\int_0^{t_{go}} \frac{f_p}{m} t\, dt}_{J} - t_\lambda \underbrace{\int_0^{t_{go}} \frac{f_p}{m} dt}_{L}\right)[\dot{\lambda}]^I}_{[v_{thrust}]^I} + \underbrace{\int_0^{t_{go}} [g]^I dt}_{[v_{grav}]^I} \tag{7}$$

And integrating again

$$\underbrace{[s_{BI}]^I_{\substack{Desired\\end\\state}}}_{[s_{DI}]^I} - \underbrace{[s_{BI}]^I_{\substack{Current\\state}}}_{[s_{BI}]^I} - [v^I_B]^I t_{go} = \underbrace{\underbrace{\int_0^{t_{go}}\!\!\int_0^{t} \frac{f_p}{m} dtdt}_{S}\, [u_\lambda]^I + \left(\underbrace{\int_0^{t_{go}}\!\!\int_0^{t} \frac{f_p}{m} t\, dtdt}_{Q} - t_\lambda \underbrace{\int_0^{t_{go}}\!\!\int_0^{t} \frac{f_p}{m} dtdt}_{S}\right)[\dot{\lambda}]^I}_{[r_{thrust}]^I} + \underbrace{\int_0^{t_{go}}\!\!\int_0^{t} [g]^I dtdt}_{[r_{grav}]^I} \tag{8}$$

$$[\dot{\lambda}]^I, [u_\lambda]^I, t_\lambda.$$

We have six equations, but seven unknowns. There are four thrust integrals of which the first-integrals are abbreviated by L and J, and the second-integrals by S and Q. They play an important part in the solution of the LTG steering law, and their accuracy depends on a good prediction of time-to-go.

To solve for the seven unknowns, I first introduce one more conditions, i.e., the turning rate of the thrust vector $[\dot{\lambda}]^I$ be zero at the end state. Then from Eq. 7 we get

$$t_\lambda = J / L \tag{9}$$

At the terminal time, the thrust velocity becomes the velocity-to-be-gained $[v_{thrust}]^I = [v_{go}]^I$. Combination with Eq. 9 yields the turning rate of the thrust vector

$$[u_\lambda]^I = unit\left\{[v_{go}]^I\right\} \tag{10}$$

To obtain a solution for the last three unknowns, we use Eq. 8 and set $[r_{thrust}]^I = [r_{go}]^I$ at the end-time

$$[\dot{\lambda}]^I = \frac{[r_{go}]^I - S[u_\lambda]^I}{Q - S\,t_\lambda} \tag{11}$$

Eqs. (9), (10), and (11) solve for the seven unknowns, given the thrust integrals J, L, S, Q and a good prediction of velocity-to-be gained and range-to-be-gained.

3.3 Program Flow

I chose CADAC++ as the simulation framework for implementing the LTG guidance law for the three-stage solid rocket booster. CADAC++ (in C++) succeeded in the year 2000 CADAC_FTN (in FORTRAN) of the previous two decades. Both codes have been used by the University of Florida, the U.S. Air Force, and others. Its architecture is documented by Zipfel (2012 and 2014).

CADAC++ consists of modules that mirror the components of the system. In our case—the ROCKET6 simulation—the LTG steering law is implemented in the *guidance* module. For greatest realism, I select the 6 DoF CADAC++ version that models the WGS84 earth, accommodates a special weather deck with winds and turbulence, and can be executed in the Monte Carlo mode.

Fig. 4 shows the top flow diagram of the guidance module. The unit vector of thrust is computed first in inertial coordinates, transformed into body coordinates, and then sent as command to the reaction control system (RCS). More details are provided in Fig. 5.

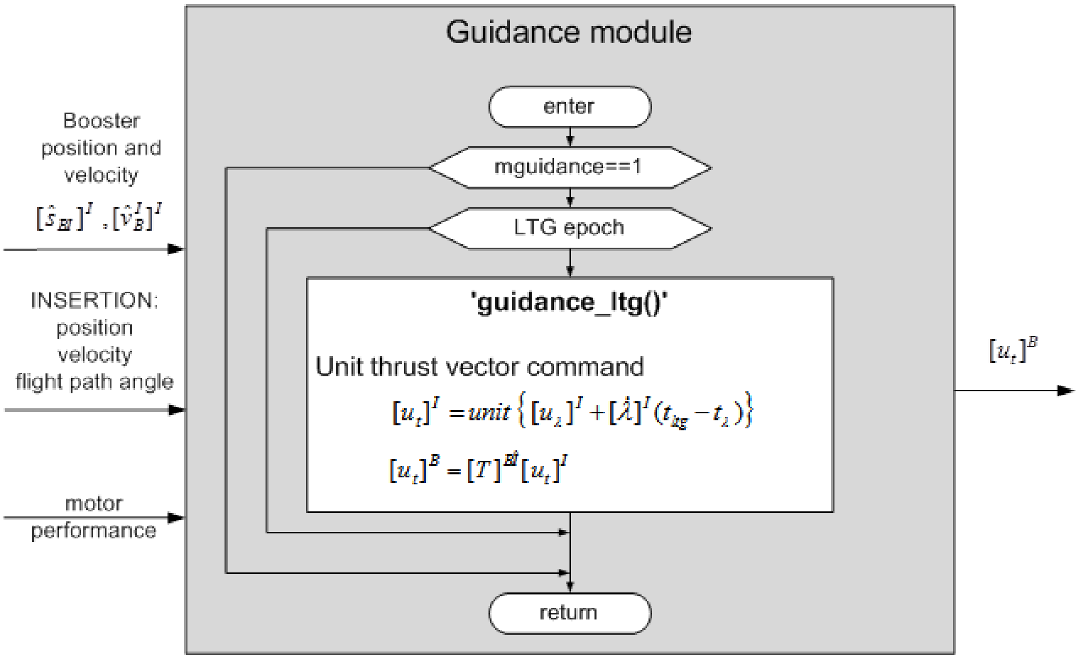

Guidance module
Booster position and velocity
$[\hat{s}_{BI}]^I, [\hat{v}_B^I]^I$
INSERTION: position velocity flight path angle
motor performance
enter
mguidance==1
LTG epoch
'guidance_ltg()'
Unit thrust vector command
$[u_t]^I = unit\{[u_\lambda]^I + [\dot{\lambda}]^I(t_{ltg} - t_\lambda)\}$
$[u_t]^B = [T]^{BI}[u_t]^I$
return
$[u_t]^B$

Fig. 4. Flow diagram of the guidance module

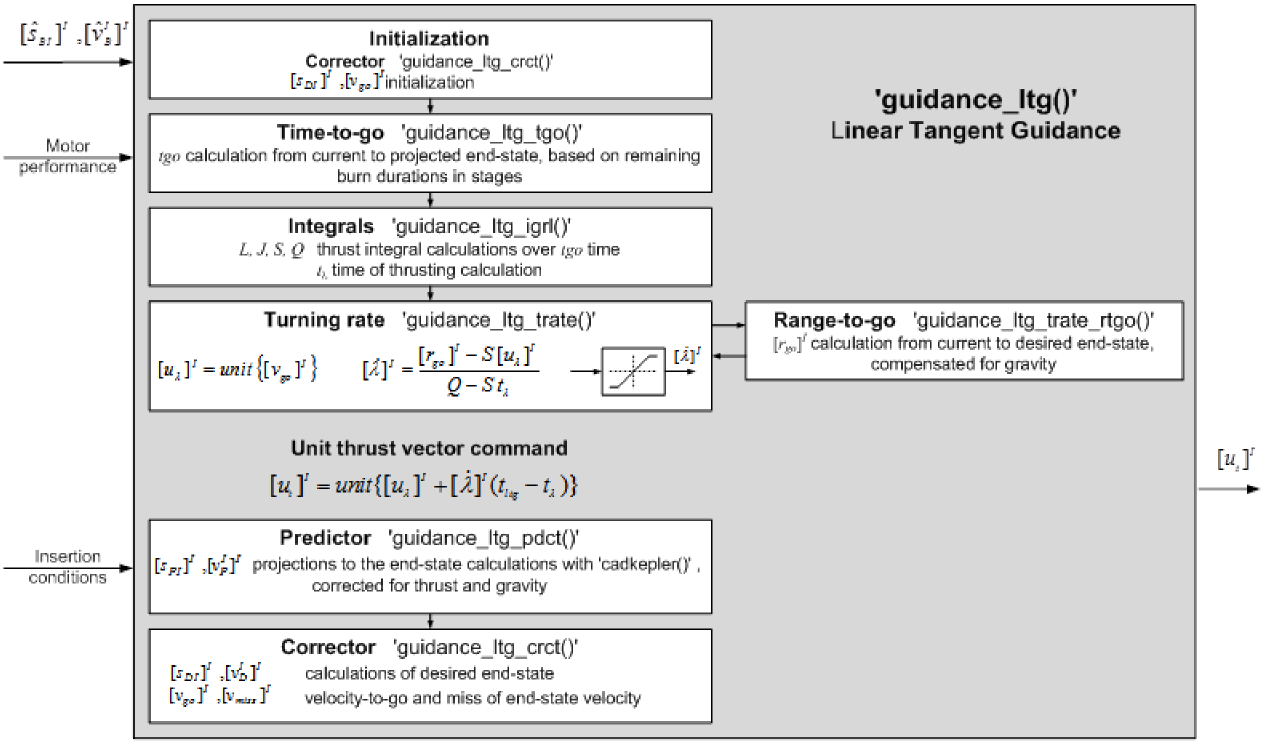

$[\hat{s}_{BI}]^I, [\hat{v}_B^I]^I$
Motor performance
Insertion conditions
Initialization
Corrector 'guidance_ltg_crct()'
$[s_{DI}]^I, [v_{go}]^I$ initialization
Time-to-go 'guidance_ltg_tgo()'
tgo calculation from current to projected end-state, based on remaining burn durations in stages
Integrals 'guidance_ltg_igrl()'
L, J, S, Q thrust integral calculations over tgo time
t_λ time of thrusting calculation
Turning rate 'guidance_ltg_trate()'
$[u_\lambda]^I = unit\{[v_{go}]^I\}$
$[\dot{\lambda}]^I = \frac{[r_{go}]^I - S[u_\lambda]^I}{Q - St_\lambda}$
$[\dot{\lambda}]^I$
Range-to-go 'guidance_ltg_trate_rtgo()'
$[r_{go}]^I$ calculation from current to desired end-state, compensated for gravity
'guidance_ltg()'
Linear Tangent Guidance
Unit thrust vector command
$[u_t]^I = unit\{[u_\lambda]^I + [\dot{\lambda}]^I(t_{ltg} - t_\lambda)\}$
$[u_t]^I$
Predictor 'guidance_ltg_pdct()'
$[s_{PI}]^I, [v_P^I]^I$ projections to the end-state calculations with 'cadkepler()' , corrected for thrust and gravity
Corrector 'guidance_ltg_crct()'
$[s_{DI}]^I, [v_D^I]^I$ calculations of desired end-state
$[v_{go}]^I, [v_{miss}]^I$ velocity-to-go and miss of end-state velocity

Fig. 5. Major C++ functions of the guidance module and their names in single quotes

After initialization, time-to-go and the corresponding thrust integrals are calculated at every guidance iteration. With range-to-go available, the turning rate of the thrust vector is determined. The predictor projects the trajectory to the final state, and the corrector corrects for the miss while calculating the velocity-to-go value. Then the unit thrust vector command $[u_t]^I$ can be issued to the RCS actuators.

3.4 Time-To-Go Calculation and BECO

The thrust integrals require an accurate prediction of the time-to-go t_{go}. Time-to-go from the current epoch to the end-state epoch is determined based on the total burn time of the remaining stages.

The required burn times of the individual stages and of the last stage (until BECO) is obtained by integrating over the thrust profiles (acceleration) until the value equals the required end-state velocity v_{go}. The integration time is t_{go}. Fig. 6 shows the three-stage thrust profiles.

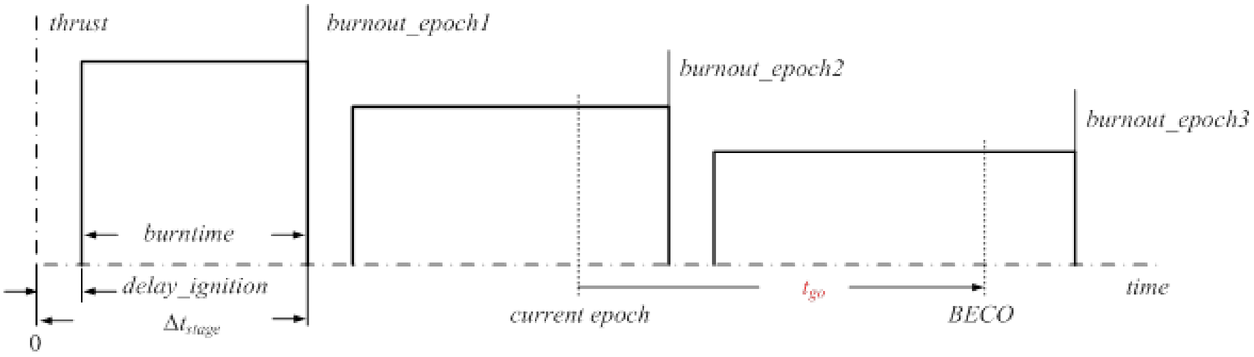


Fig. 6. Three-stage booster thrust profile

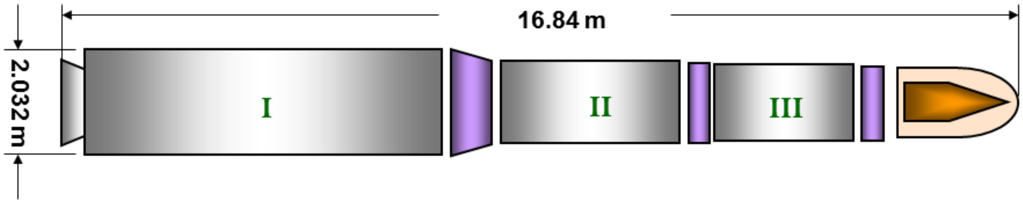
Each stage is characterized by fuel mass, specific impulse, and burn rate, from which thrust and burn-time is calculated. The separation times must be taken into account. When the last stage meets the end conditions, BECO is issued and the rocket engine shuts down.

4. Three-Stage Booster

Three-stage solid rocket boosters are relatively rare. They are primarily used for military purposes, like the Peacekeeper and the Minuteman II. These are ballistic missiles and not orbital delivery systems—the latter having mostly liquid fueled engines. However, just recently, Orbital Sciences started converting its Antares booster into an all solid propelled orbital launch system. Solid rockets are cheaper, easier to maintain, and ready to fire at the push of a button.

As a test case, I use a generic three-stage solid rocket booster, called Small Launch Vehicle (SLV). Its mass properties and thrust profiles were provided by the U. S. Air Force Research Laboratory and the aerodynamic properties were obtained from Missile Datcom.

Fig. 7 shows the basic dimensions, mass properties, and motor performance. The booster can place about 1000 kg into low earth orbit. Notice that for 6 DoF implementation, you need beside mass also moment of inertia and c.g. locations.



Mass Properties

MODE		Launch	Fire 2nd Stage	Fire 3rd Stage
Mass	kg	48,983.70	15,490.50	5,024.48
CgX	m	10.5265	5.9100	3.6510
CgY	m	-0.0001	-0.0002	-0.0006
CgZ	m	0.0000	-0.0001	-0.0002
Ixx	kg*m^2	21,943.38	5,043.47	1,519.04
Iyy	kg*m^2	671,626.02	51,912.18	5,157.65
Izz	kg*m^2	671,624.75	51,928.57	5,178.28

Motor Specifications

STAGE #		I	II	II
Thrust (vac)	N	1,407,610	527,903	124,907
Isp (vac)	sec	279.200	284.600	284.488
Exit Area	m^2	1.0601	0.5838	0.2318

Fig. 7. Three-stage solid booster test case

In the ROCKET6 simulation, the booster with its mass properties is modeled in the propulsion module. Fig. 8 shows all the modules that comprise the simulation. The colors indicate the class structure. The class hierarchy starts with an abstract base class, called *Cadac*, which has no modules, only bookkeeping functions. From it is derived the class *Round6* with its member functions shown in green. Here the 6 DoF equations of motions are solved, kinematic calculations executed, and the atmospheric environment provided with its atmosphere, winds, and turbulence.

Finally, the blue modules, which model the booster components, are part of the class *Hyper* that derives from the *Round6* class. In each block the interface arrays that communicate between the modules are indicated.

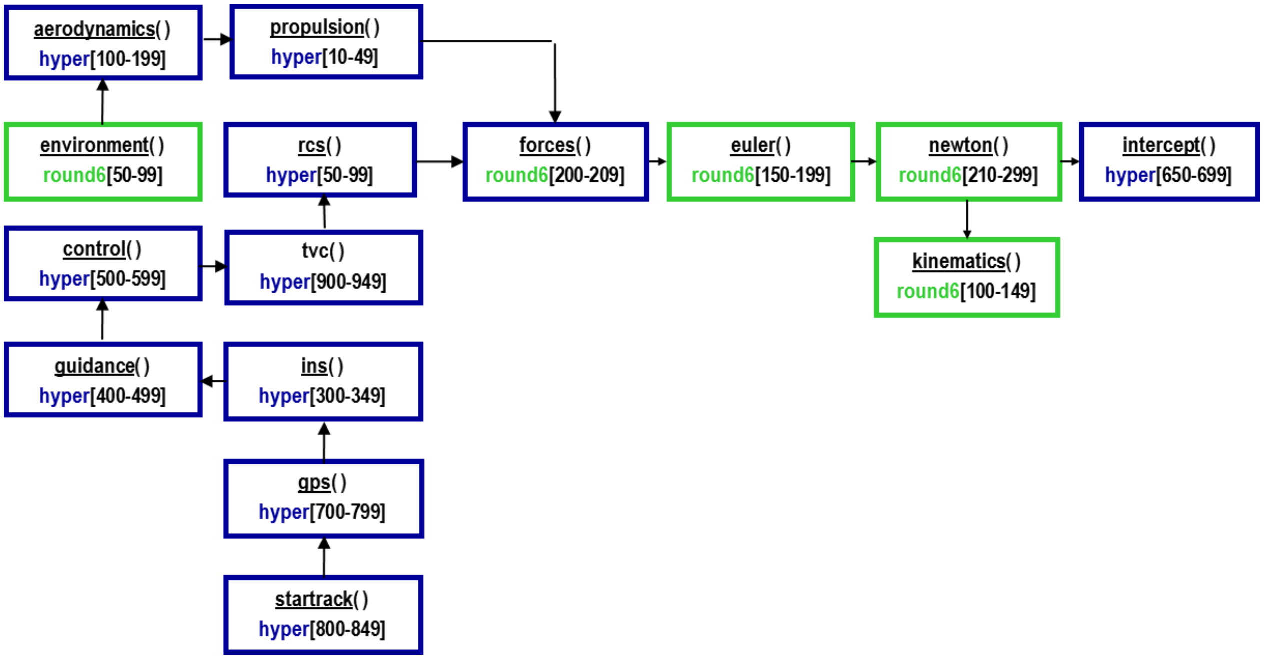


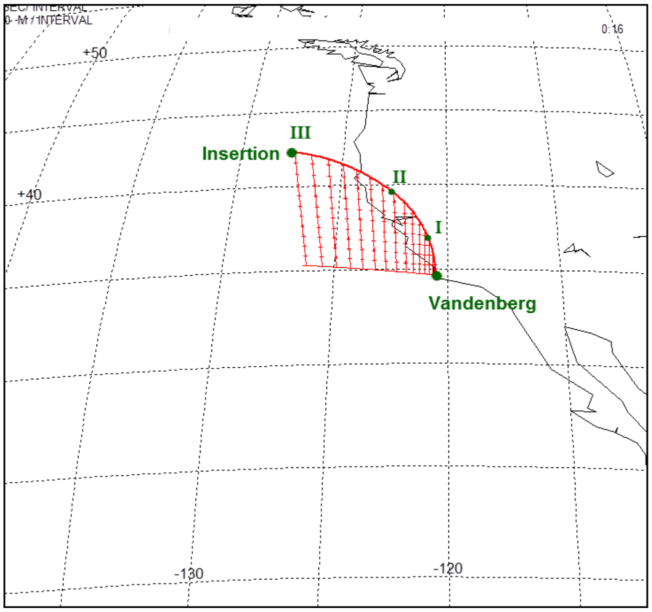
Fig. 8. ROCKET6 modular architecture

We are already familiar with the guidance, propulsion, and aerodynamic modules. The remaining modules complete the high fidelity ROCKET6 simulation. 'startrack', 'GPS', and 'INS' provide the guidance module with realistic navigation information. The guidance module operates in two modes. During endo-atmospheric ascent, it commands the pitch program to line up the booster for the exo-atmopheric second mode in which the LTG steers the booster to its orbital end phase. The guidance commands are converted in the control module to signals that either command the thrust vectoring nozzles of the first stage or the reaction control systems of stages two and three.

The simulation loops through these modules each integration step, occuring every five milliseconds. At booster staging, new propulsion, mass properties, and guidance parameters are read in. Eventually, the third booster will reach its terminal state and the intercept module will stop the run and print out the achieved insertion values with their deviations from the desired end-conditions.

5. Six DoF Simulation Results

As test case, a launch from Vandenberg AFB, CA is shown Fig. 9. At this launch complex, the booster is launched in a westerly direction and cannot take advantage of the easterly earth rotation. To the contrary, the booster has to overcome the initial inertial velocity of 382 m/s at Vandenberg. The desired insertion conditions are 6,470,000 m from the earth center (about 100 km altitude), 6600 m/s inertial speed, and 1 deg flight path angle.



Navigation
GPS/INS

Guidance
I Pitch Program
II & III LTG

Control
RCS lift-off
TVC endo
RCS exo

Fig. 9. Test trajectory

First, I make a run without any system errors to assess the LTG performance under ideal conditions. ROCKET6 prints out these end conditions:

```
input_test.asc  Three-stage rocket ascent   Sep  3 2014 09:56:59
***Boost engine cut-off time = 182.255 sec ***
   Orbital position = 6.47e+006 m  Inertial speed = 6600.6 m/s  Flight path angle  = 0.964598 deg
   Position error    = -1.083 m         Speed error   = -0.600766 m/s     Angle error   = 0.0354019 deg
```

As you can see, the LTG does an excellent job meeting the end conditions. However, it does not reflect the actual world. Major uncertainties are introduced by the INS performance and wind/turbulence effects. I choose a strap-down IMU updated by GPS and a fairly severe wind profile combined with a low altitude Dryden turbulence model, see Zipfel (2014). Now, because stochastic errors corrupt the solution, ROCKET6 must be run in the Monte Carlo mode.

Fig. 10 shows the results of 100 Monte Carlo runs. The scatter plot is a bivariate representation of the velocity error versus the range error (distance from center of the earth). The respective mean values are marked by the center of the 50% error ellipse. They are as follows together with their standard deviations: range error 0.9484±5.0576m; velocity error 0.1228±1.854m/s. This represents an excellent insertion performance, which is further supported by the narrow spread of the flight path angle mean and standard deviation of 0.0168±0.0158 deg. At insertion, the trajectory is slightly biased upward, a desirable condition.

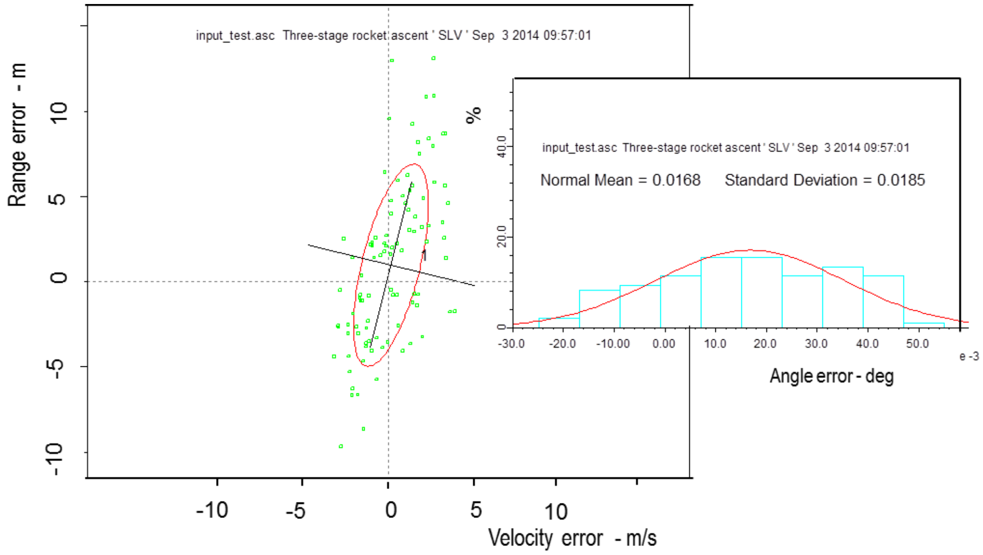


Fig. 10. Orbital insertion accuracy. Scatter plot for velocity vs. range error and histogram for flight path angle error (100 Monte Carlo runs)

To complete the presentation of the orbital insertion of a three-stage solid booster, I provide in Fig. 11 the salient ascent trajectory traces of the test case.

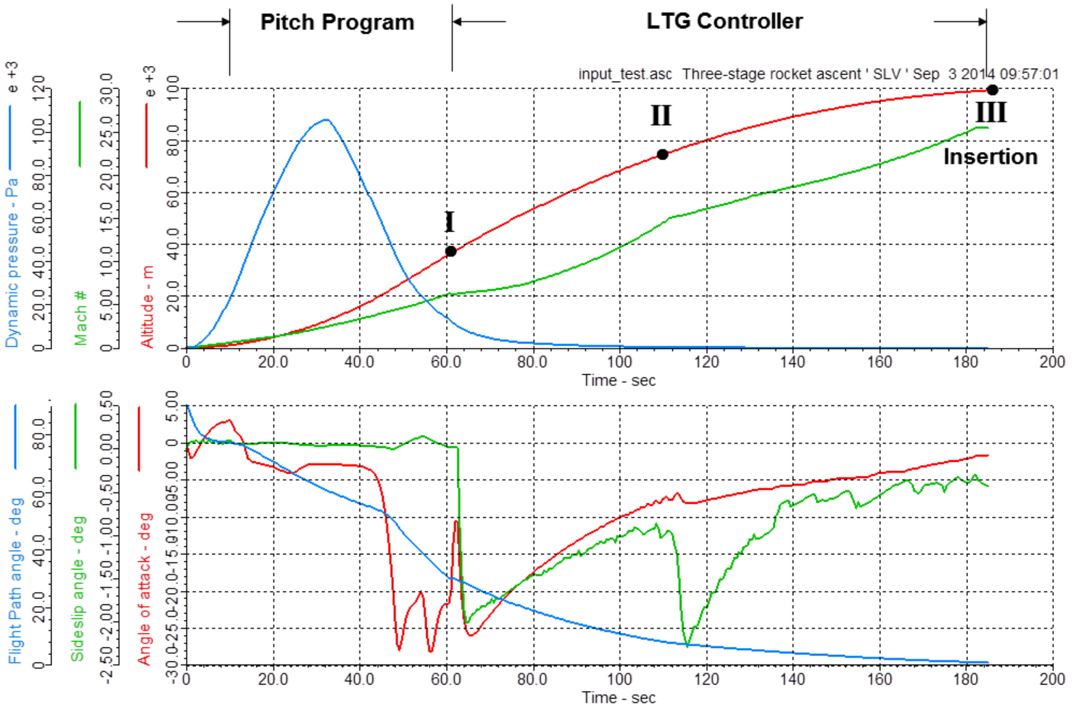


Fig. 11. Sample trajectory traces of the test case

The booster lifts off the launch pad under RCS control. At 10 sec, the pitch program takes over with the first stage using TVC to control pitch and yaw, while RCS continues to maintain zero roll angle. During this phase, while the dynamic pressure reaches its maximum, it is mandatory to keep the angle of attack and sideslip angle under control in order not to exceed the structural load limits of the booster. As the first stage is jettisoned, the LGT controller takes over to guide the remaining stages to the insertion conditions. Because the nozzles of the rockets are fixed, the whole booster has to be turned to direct the thrust. The excursion of the angle of attack and sideslip angle evidence this behavior. Eventually, at insertion, these angles are near zero.

6. Summary and Conclusions

This was a short exposition of a complex project. The guidance law that inserts a space vehicle into orbit is based on the linear tangent guidance law. That is the case for the Apollo program and the Space Shuttle and will also be so for the Orion program. Here, I applied it to a three-stage solid booster that inserts a 1000 kg payload into low earth orbit. I tried to establish the connection between the derivation of the guidance equations and their implementation in a high fidelity six degrees-of-freedom simulation, called ROCKET6. To assess the performance under real-world conditions, the Monte Carlo methodology was employed with wind and gust effects, as well as uncertainties in INS and GPS performance. The results of 100 Monte Carlo runs show that the orbital insertion conditions are accurately met.

This presentation is unique, because no other high fidelity, open source simulation of this kind is known to be available. ROCKET6, in conjunction with this paper, is well suited for graduate level research, and engineering trade-off studies.

References

Bryson, A. E., & Ho, Y. (1975). Applied optimal control. Hemisphere Publishing Co.

Chavez, F. & Lu, P. (2009). Rapid ascent trajectory planning and closed-loop guidance for responsive launch. In AIAA 7th Responsive Space Conference, April 27–30, 2009, Los Angeles, CA.

Draper, C. S.; Wrigley, W.; Hoag, D. G.; Battin, R. H.; Miller, J. E.; Koso, D. A.; Hopkins, A. L.; Vander Velde, W. E. (1965). Apollo guidance and navigation. Instrumentation Laboratory, R500, Vol. 1. Guidance, Navigation, and Control Conference, August 2011, Portland, OR.

Jaggers, R.F. (1972). Multi-stage linear tangent guidance as baseline for the space shuttle vehicle. NASA MSC-07458, Internal Note MSC-EG-72-39.

Long, A. D. & McHenry, H. L. (undated). Powered flight prediction, NASA class notes.

Perkins, F. M. (1956). Flight mechanics of ascending satellite vehicles. Aerospace Corporation. Jet Propulsion Vol. 25 (5), Part 1.

Perkins, F. M. (1966). Derivation of linear-tangent steering laws. The Aerospace Corporation. AD 643209.

Teren, F. (1966). Explicit guidance equations for multistage boost trajectories. NASA TN D-3189.

The Wall Street Journal (24 August 2014). Galileo Satellites Launched into Wrong Orbits.

Thrasher, S. W. & Fill, T. J. (2011). Orion's exoatmospheric burn guidance architecture and algorithm. In AIAA Guidance, Navigation, and Control Conference, August 2011, Portland, OR.

Zipfel, P.H. (2012). CADAC: Multi-use architecture for constructive simulations. Journal of Defense Modeling and Simulation, (Vol. 9, Num. 2, pp. 129-145).

http://dx.doi.org/10.1177/1548512910395641

Zipfel, P.H. (2014). Modeling and simulation of aerospace vehicle dynamics. American Institute of Aeronautics and Astronautics, 3rd Edition.
http://dx.doi.org/10.2514/4.102509