Re-Entry Point Targeting for LEO Spacecraft using Aerodynamic Drag

Aerospace Control and Guidance Systems Committee Meeting #118

October 19-21, 2016

Sanny Omar

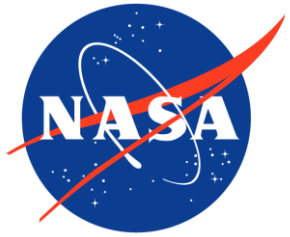
Riccardo Bevilacqua

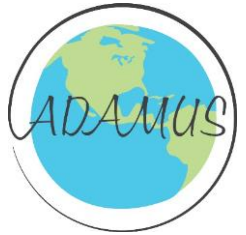
Laurence Fineberg

Justin Treptow

Yusef Johnson

Scott Clark





UF | UNIVERSITY of FLORIDA

Project Overview and Objectives

- Most Low Earth Orbit (LEO) spacecraft do not have thrusters and re-enter atmosphere in random locations at uncertain times
  - Objects pose a risk to persons, property, or other satellites
  - Has become a larger concern with the recent increase in small satellites
- Working on a NASA funded project to design a retractable drag device to expedite de-orbit and target a re-entry location through modulation of the drag area
- Will be discussing the re-entry point targeting algorithm here

Guidance Generation Algorithm Control Parameters

- Must control de-orbit latitude and longitude
- Control parameters are
  - t_{swap} = time until ballistic coefficient is changed
  - C_{b1} = ballistic coefficient from t_0 to t_{swap}
  - C_{b2} = ballistic coefficient from t_{swap} to t_{term}
    - Spacecraft maintains some predetermined drag profile after t_{term}
- Given enough time, variation of these parameters should be sufficient to target any point on the ground whose latitude is below the orbit inclination
  - This algorithm generates a guidance that the spacecraft must follow to re-enter in the desired location
- The first step is to establish a way of determining de-orbit location based on spacecraft initial conditions and control parameters

Assumptions for Analytical Solution

- Circular orbit around spherical Earth
- Density is a function of semi major axis
  - If density is a function of altitude in a circular orbit around a spherical Earth, density is also a function of semi major axis
- De-Orbit point is just before aerodynamic forces exceed gravitational forces
  - orbital elements still valid

Analytical Mapping from Initial to Final State

- If a satellite with C_{b1} takes time t_1 to achieve some change in semi major axis and experiences a change in true anomaly $\Delta\theta_1$ during this time, then for a satellite with the same initial conditions and C_{b2},

$$t_2 = \frac{C_{b1} t_1}{C_{b2}}$$

$$\Delta\theta_2 = \frac{C_{b1} \Delta\theta_1}{C_{b2}}$$

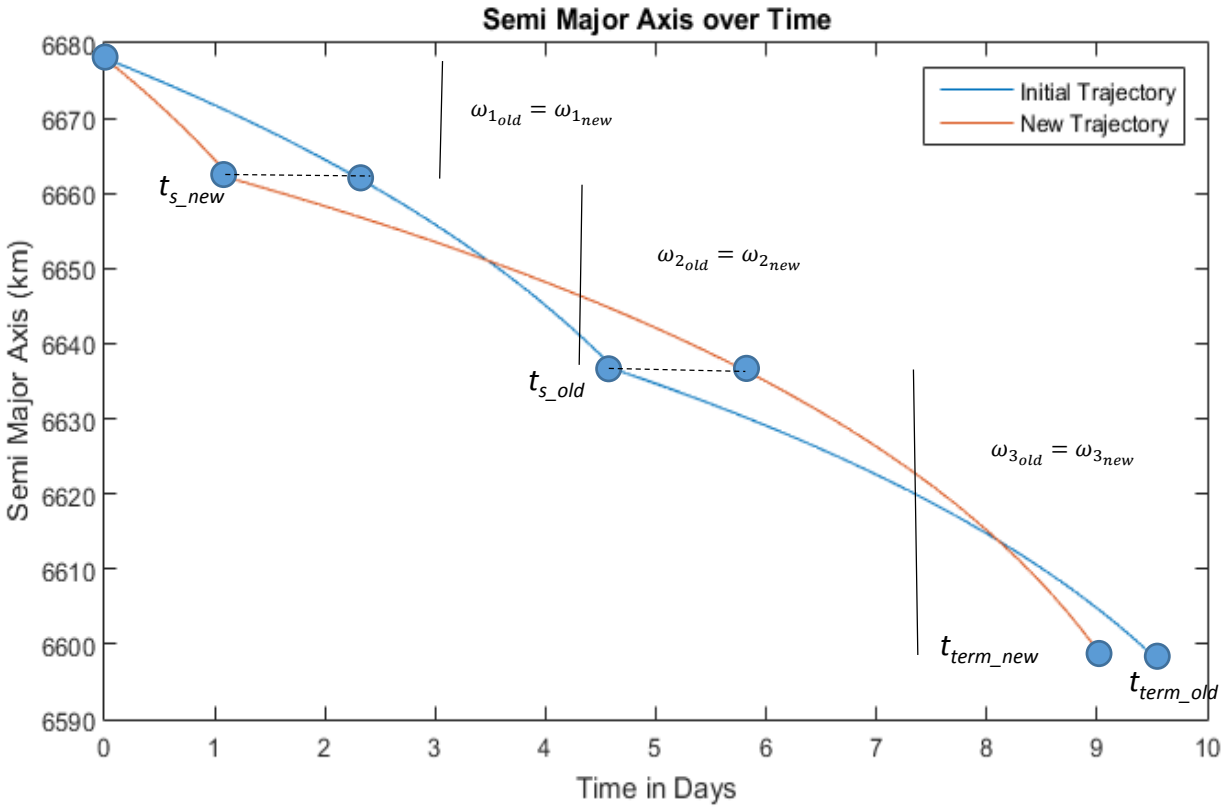
- It also proves that the average orbital angular velocity $\omega_{avg} = \frac{\Delta\theta}{\Delta t}$ for a given change in semi major axis is independent of ballistic coefficient

Analytical Mapping from Initial to Final State Cont.

- A trajectory is propagated from the initial conditions with some C_{b1}, C_{b2}, and t_{swap}
- The relations $\Delta\theta_2 = \frac{C_{b1}\Delta\theta_1}{C_{b2}}$ and $t_2 = \frac{C_{b1}t_1}{C_{b2}}$ are used to determine the de-orbit time and change in true anomaly of a spacecraft with the same initial conditions and a different C_{b1}, C_{b2}, and t_{swap}
- Average rate of change of right ascension of the new trajectory is the same as in the initial trajectory
  - RAAN change of new trajectory is old RAAN rate times new orbit lifetime
- From this, the orbital elements of the spacecraft at de-orbit can be calculated. These orbital elements and the de-orbit time can be used to calculate de-orbit latitude and longitude.

Determining Δt_t and $\Delta \theta_t$ for New Trajectory Based on Old Trajectory

- Divide trajectories into three phases
- Phases go from same initial to final semi major axes in old and new trajectories
  - C_b values are unchanging in each phase
- Average angular velocity in each phase constant between old and new trajectories
- Time and change in true anomaly associated with each phase in the new trajectory calculated based on corresponding phase in old trajectory and analytical relations
- Both spacecraft assumed to follow the same decay profile after t_{term} (terminal point)



Overview of Targeting Algorithm

- Control parameters are C_{b1}, C_{b2}, and t_{swap}
- Vary t_{swap} only such that the target latitude is achieved with minimum correctable longitude error
- Determine the change in orbit lifetime needed to hit desired longitude
  - Time for earth to rotate into desired position
  - Max longitude error is $\left(\frac{T}{2*t_{day}}\right) * 2\pi R_e \approx 1250\ km$ at equator
- Manipulate C_{b1}, C_{b2}, and t_{swap} such that orbit lifetime changes to correct longitude error as much as possible

Step 1: Targeting Latitude with t_{swap} Only

- Time required and change in true anomaly the same between old and new trajectories during phases one and three.
- During phase two:

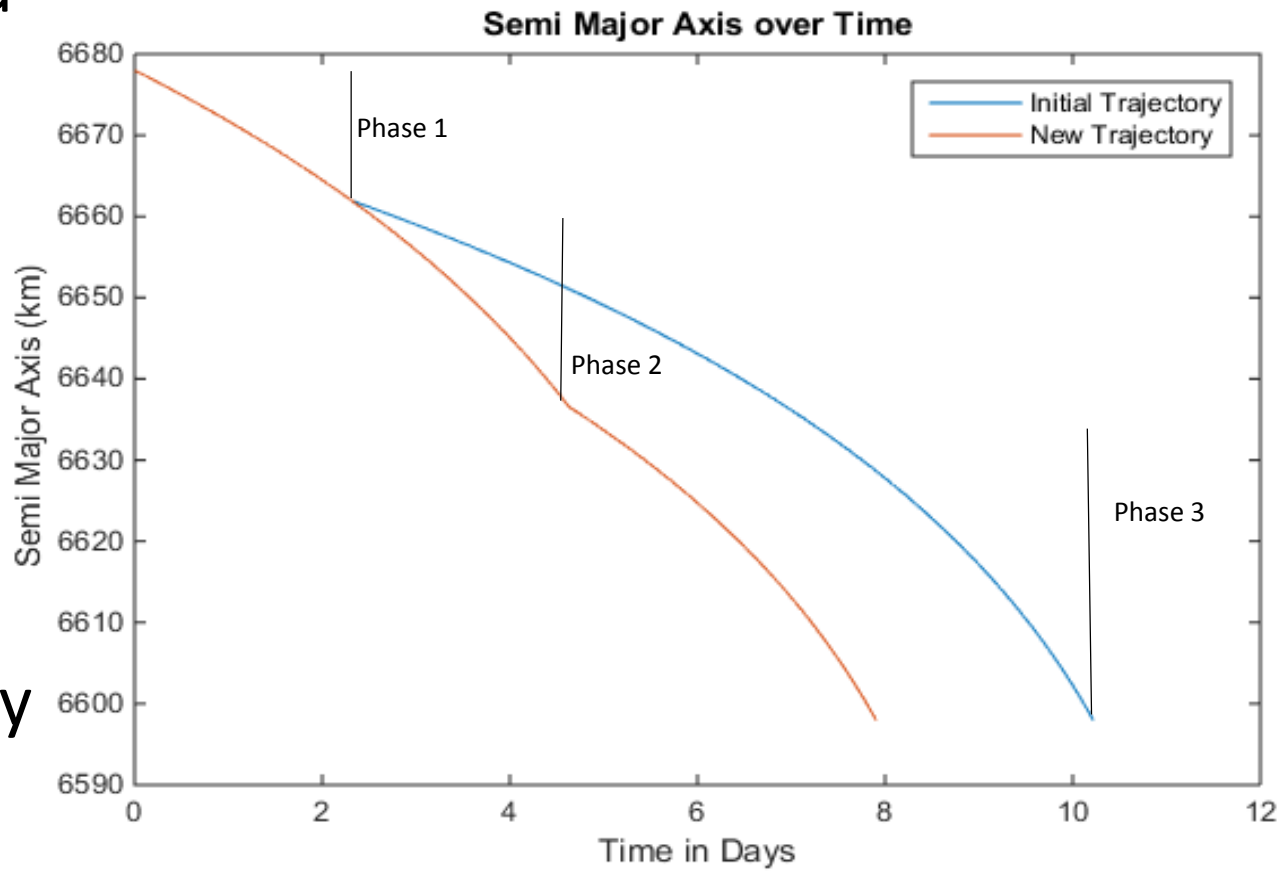
$$t_2 = \frac{C_{b2} t_{20}}{C_{b1}}$$

- Total increase in orbit lifetime given by

$$\Delta t_d = \Delta t_{swap} \left(1 - \frac{C_{b1}}{C_{b2}}\right)$$

- Increase in total change in true anomaly given by

$$\Delta\theta_d = \omega_{2\_avg} \Delta t_d$$



Latitude Targeting Cont.

- If the $\Delta\theta_d$ needed to get from the de-orbit location to the target latitude is calculated, the increase in t_{swap} required can be calculated

$$\Delta\theta_d = \omega_{2_{avg}} \Delta t_{swap} \left(1 - \frac{C_{b1}}{C_{b2}}\right)$$

$$\Delta t_{swap} = \frac{\Delta\theta_d C_{b2}}{\omega_{2_{avg}} (C_{b2} - C_{b1})}$$

- $\omega_{2_{avg}}$ is dependent on t_{swap} so it is necessary to iterate to pick the proper t_{swap}
- In practice, only a few iterations needed until error is negligible

Finding the Optimal Swap Time

- Maximum and minimum t_{swap} values

$$t_s \in \left[0, \left(t_{s_{old}} + \frac{C_{b2}}{C_{b1}} t_{20}\right)\right]$$

- Find all feasible t_{swap} values that yield perfect latitude targeting
- Pick t_{swap} that also yields minimum correctable longitude error and is close to the t_{swap} from the initial numerically propagated trajectory
  - Calculating longitude controllability will be discussed soon

Step 2: Analytical Longitude Targeting

- Latitude targeting corresponds to a total change in true anomaly from the initial conditions
- We want to change orbit lifetime without varying the total change in true anomaly
  - This allows us to wait until the Earth rotates into the proper position for longitude targeting without disturbing the latitude targeting

$$\Delta t_d = \frac{\lambda_{err}}{\omega_e}$$

- We can do this by varying C_{b1}, C_{b2}, and t_{swap} in a special way

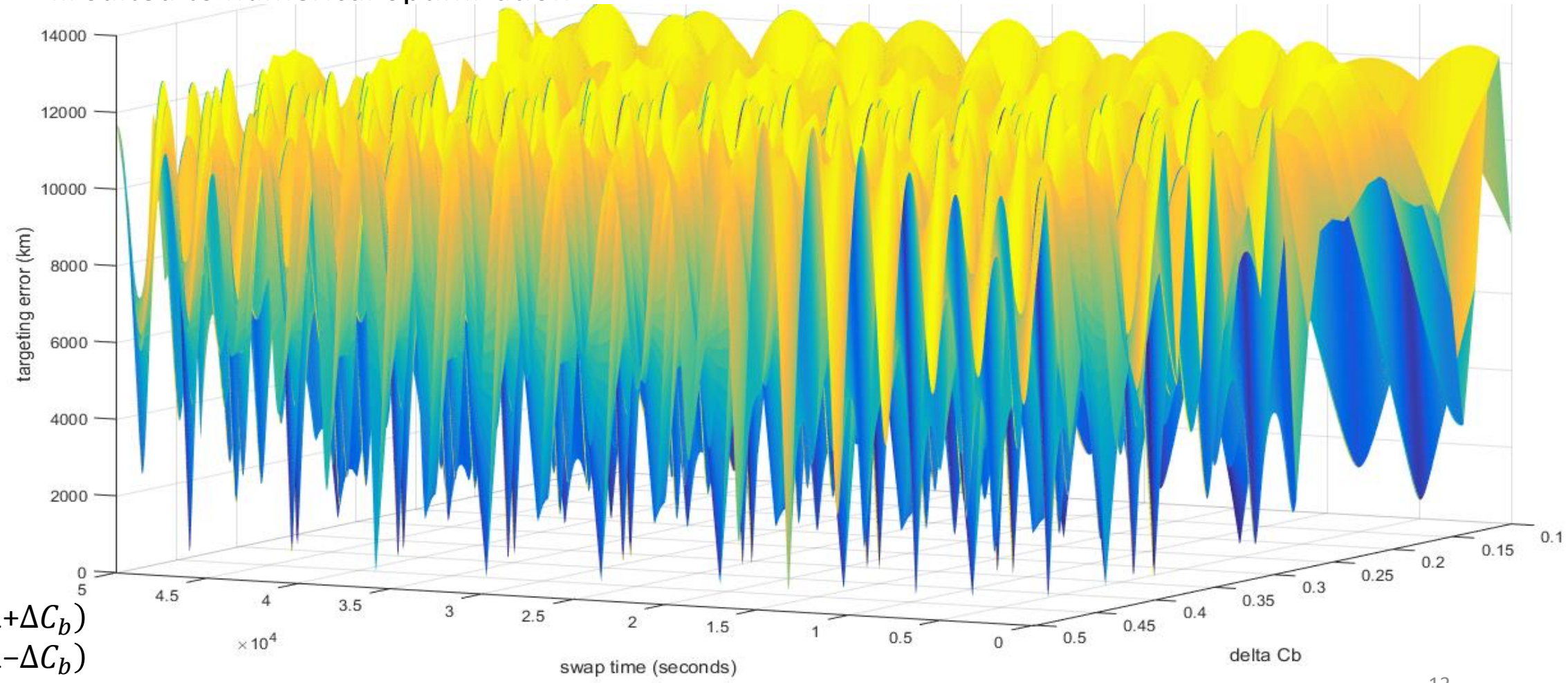
$$C_{b2} = \frac{C_{b20}\left(\Delta t_{20}(\omega_{10} - \omega_{20})\right)}{(\Delta t_{20})(\omega_{10} - \omega_{20}) + (\Delta t_d)\omega_{10}}$$

$$C_{b1} = \frac{\Delta\theta_{10} C_{b10} C_{b2}}{(\Delta\theta_{10} + \Delta\theta_{20}) C_{b2} - \Delta\theta_{20} C_{b20}}$$

$$t_{s_{new}} = \frac{t_{s_{old}} C_{b10}}{C_{b1}}$$

Controllability Analysis of Targeting Algorithm

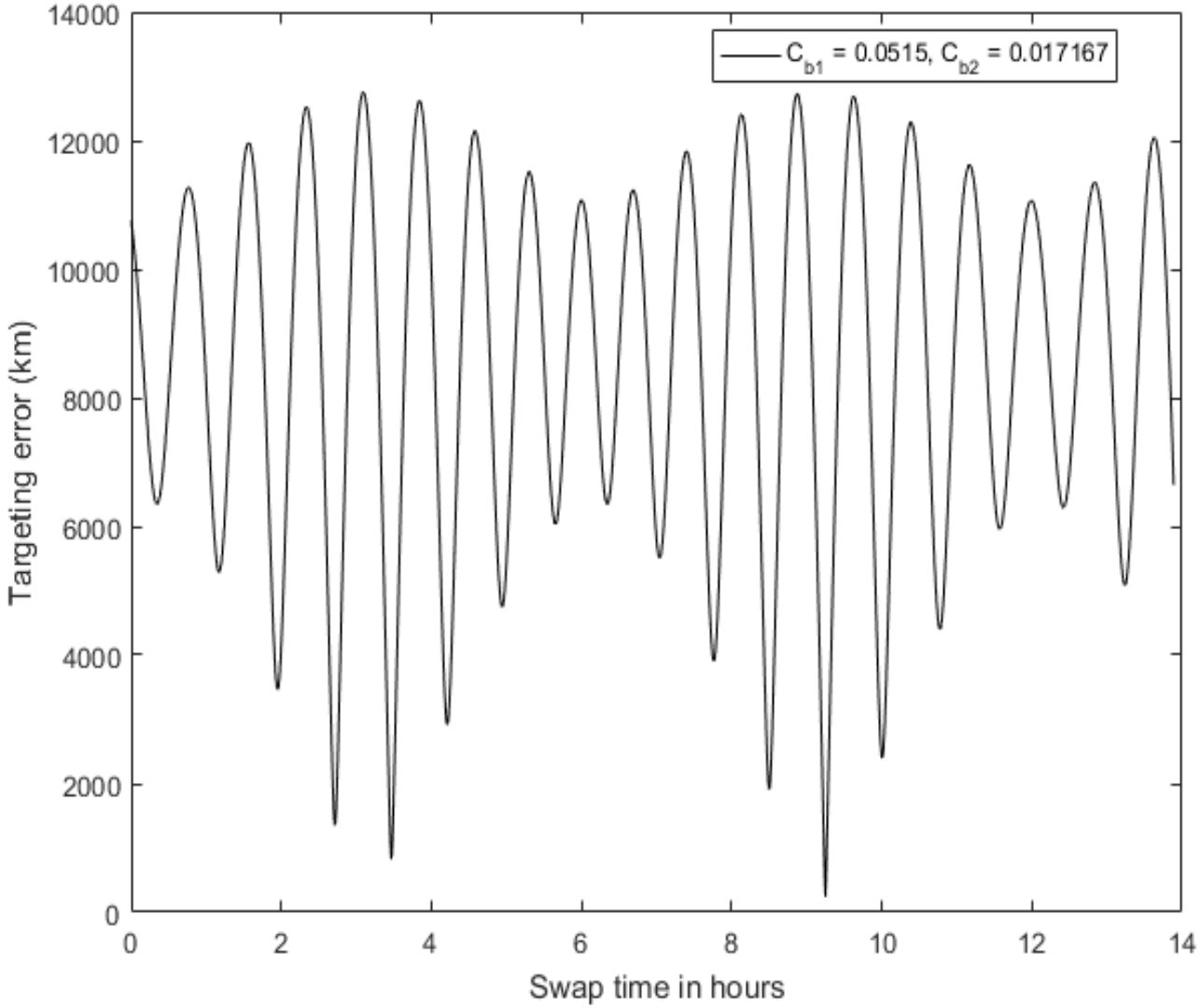
- Controllability is the ability to get from some initial state to any desired final state using the available controls
- System may be controllable but has numerous local minima
  - Ill suited to numerical optimization



$C_{b1} = .034(1+\Delta C_b)$
$C_{b2} = .034(1-\Delta C_b)$

Effects of Variations of Only t_{swap} Cont.

- With large difference between C_{b1} and C_{b2}, multiple t_{swap} values provide local error minimization
- Both latitude and longitude error can be made quite small with only variations of t_{swap} in this case



Min and Max Life Increase Without Changing Latitude Targeting

- Max life increase requires the maximum possible value of C_{b2} and minimum C_{b1}
- Solve for the required C_{b1} given max C_{b2}

$$C_{b1} = \frac{\Delta\theta_{10} C_{b10} C_{b2}}{(\Delta\theta_{10} + \Delta\theta_{20}) C_{b2} - \Delta\theta_{20} C_{b20}}$$

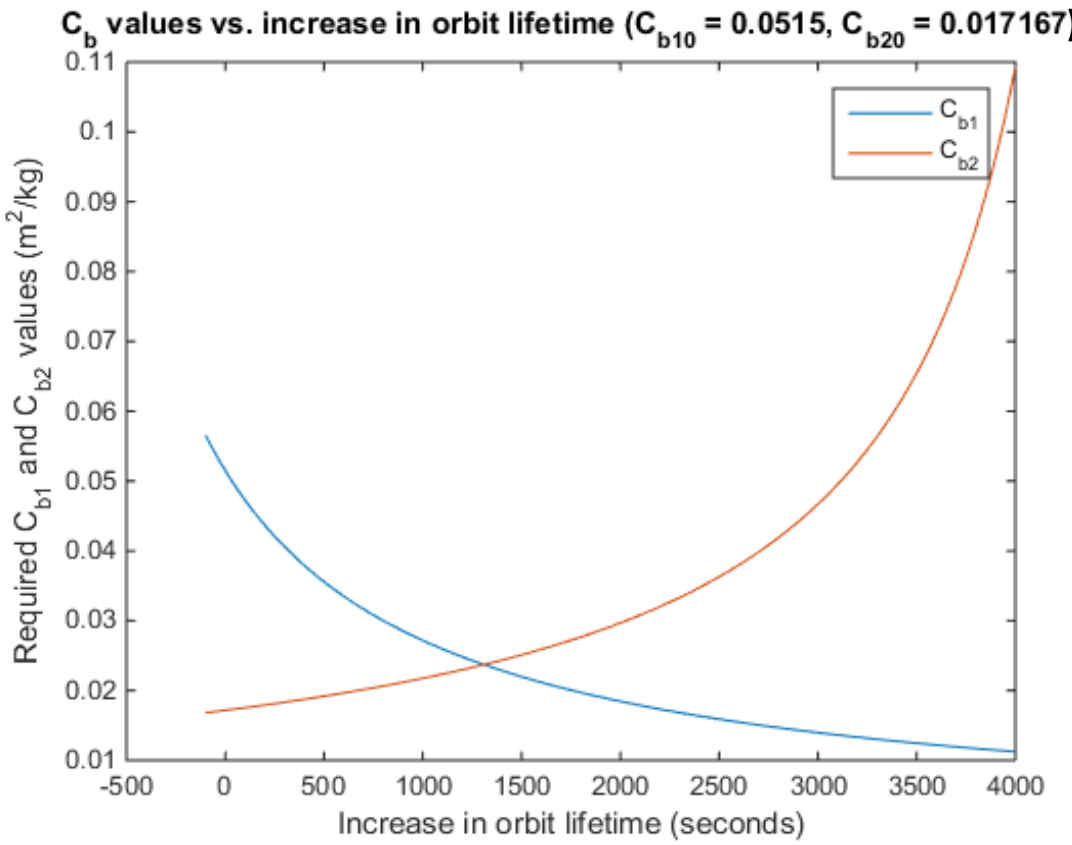
  - If this C_{b1} is attainable, the calculated configuration yields max life increase
  - Otherwise: solve for required C_{b2} given min C_{b1}

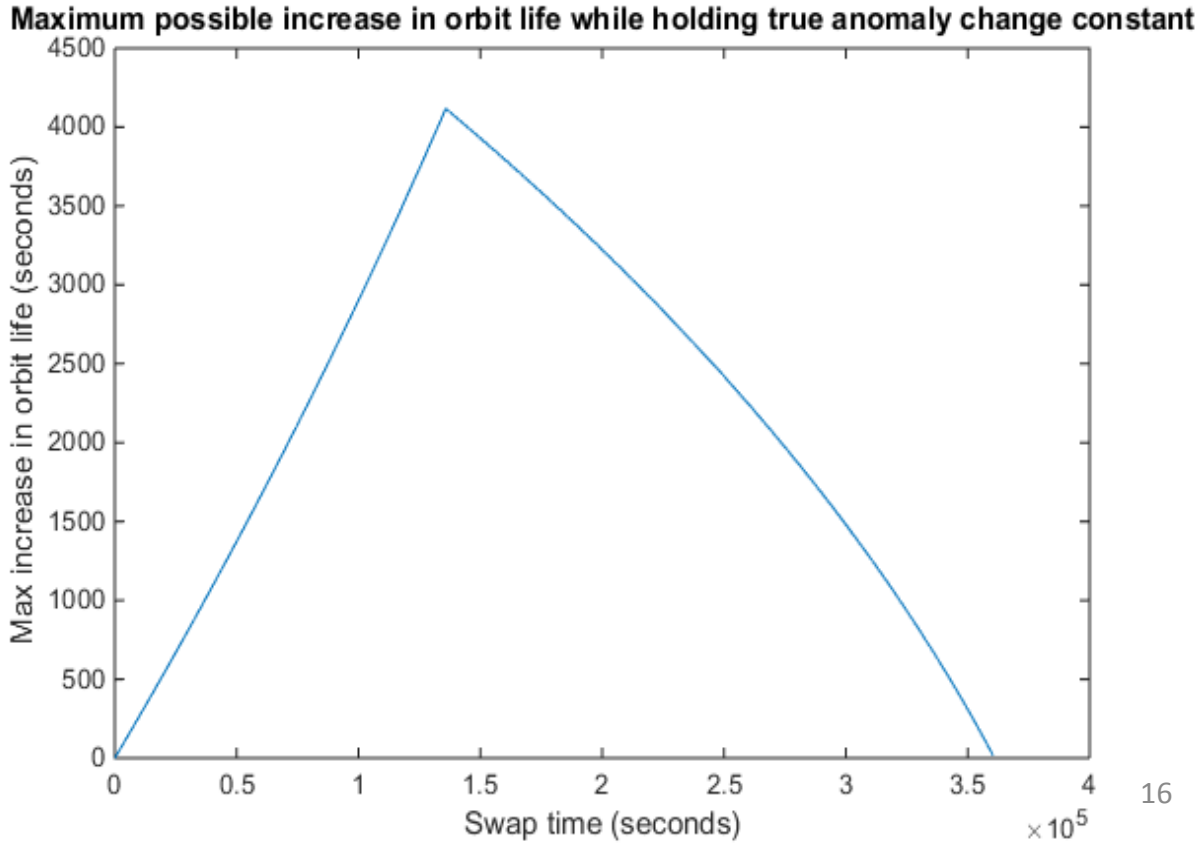
$$C_{b2} = \frac{\Delta\theta_{20} C_{b20} C_{b1}}{(\Delta\theta_{10} + \Delta\theta_{20}) C_{b1} - \Delta\theta_{10} C_{b10}}$$

    - This configuration gives max life increase
- For min life increase solve for required C_{b1} given min C_{b2} or required C_{b2} given max C_{b1}
  - Min life increase may be negative

Ensuring Maximum Controllability

- Max possible increase in orbit lifetime occurs when t_{swap} is near the middle of the decay trajectory and C_{b10} and C_{b20} are as far apart as possible.
- Graphs below for an initial 300 km circular orbit with $C_{b10} = .0515$ and $C_{b20} = .0172$ and maximum and minimum C_b values of .1 and .01 m²/kg respectively

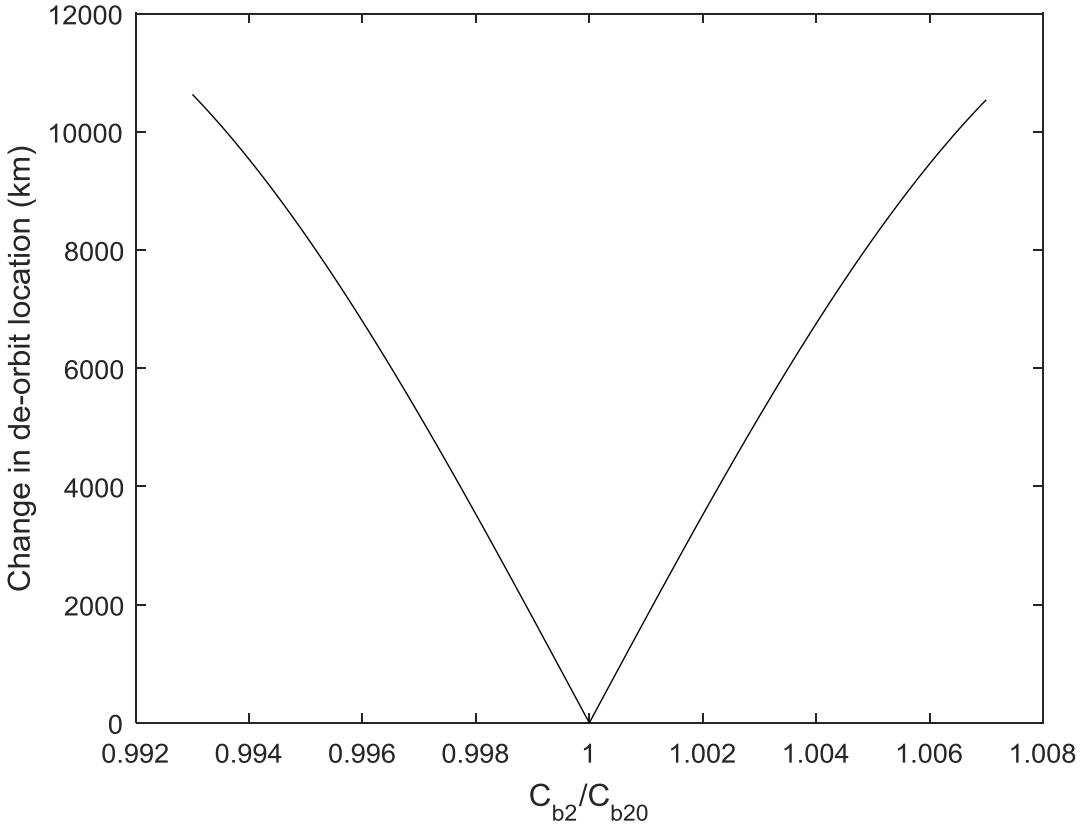




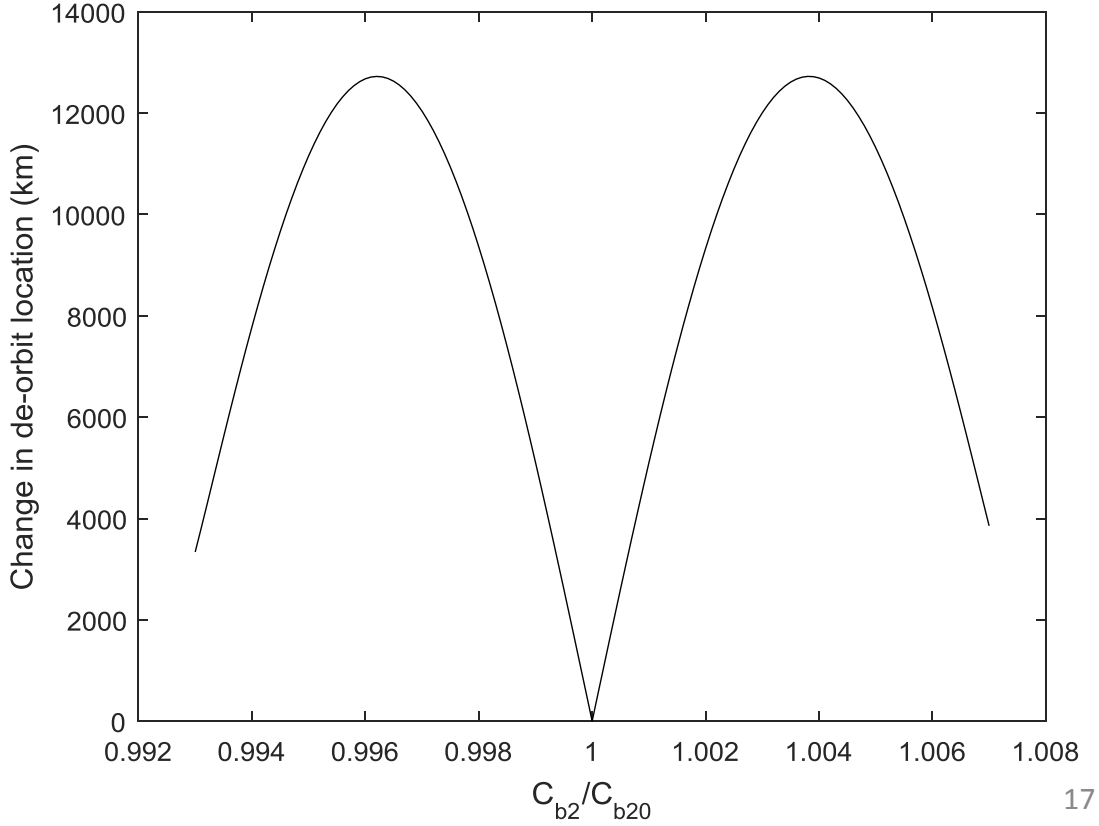
Sensitivity Analysis

- Investigates effects of changes in drag profile on de-orbit location
- An error in estimated drag force of .25% puts satellite on the other side of Earth
  - Closed loop control definitely necessary
  - Changes in density profile between numerical propagations can cause analytical and numerical solutions to diverge
    - System less sensitive when orbit life is shorter

270 km circular orbit with 1976 standard atmosphere, C_{b10}=.025, C_{b20}=.01, t_{s0}=150,000 s



300 km circular orbit with 1976 standard atmosphere, C_{b10}=.025, C_{b20}=.01, t_{s0}=150,000 s



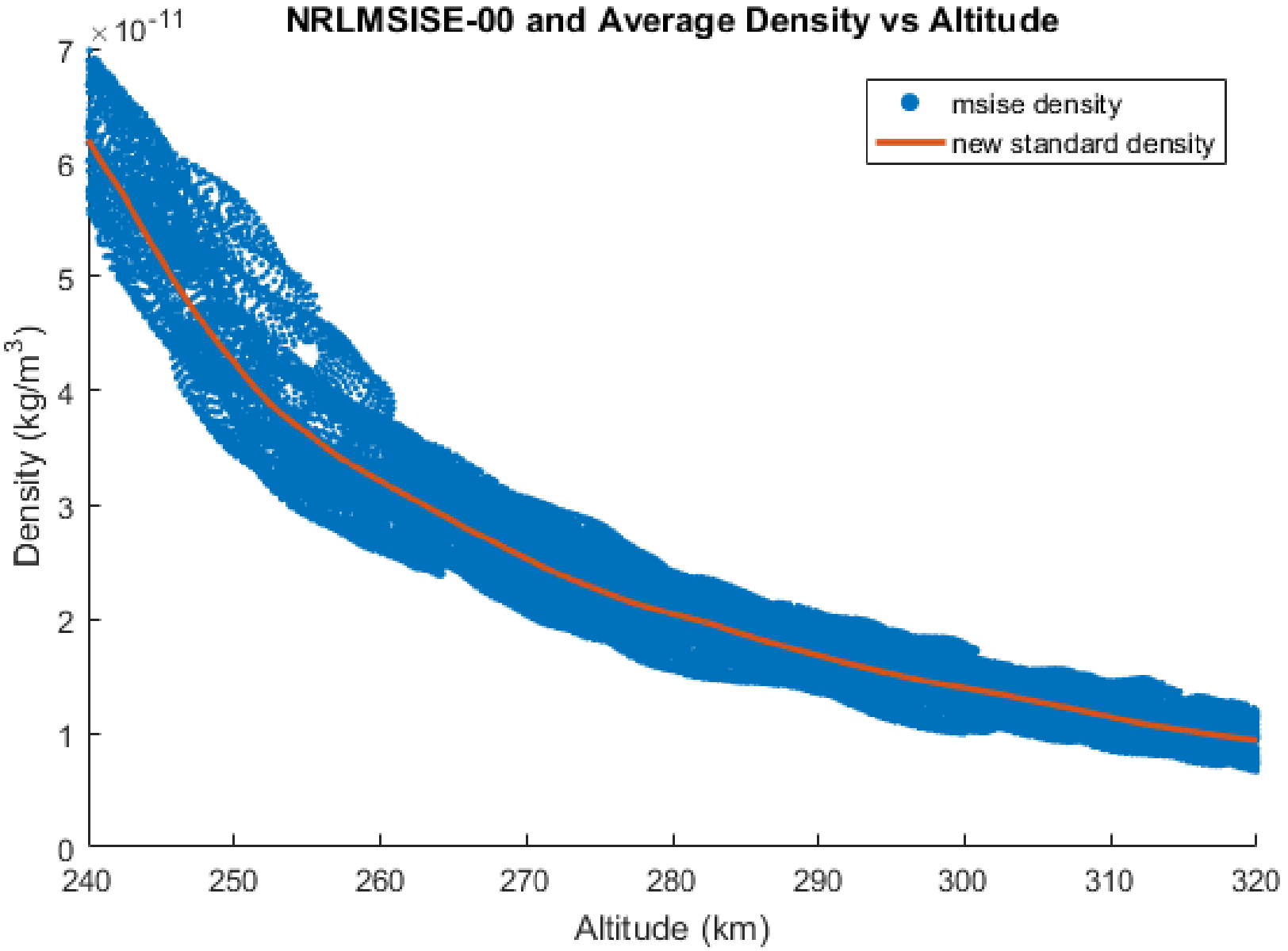
Creating a New “Standard” Atmosphere

- Procedure necessary when variations in NRLMSISE-00 density vs. altitude profile between iterations cause deviations between analytical and numerical solutions
- Latitude targeting performed using NRLMSISE-00 model until spacecraft close to desired target
- Atmosphere divided into bands where density in each band decays exponentially with altitude

$$\rho = \rho_o e^{\left[-\frac{h-h_o}{H}\right]}$$

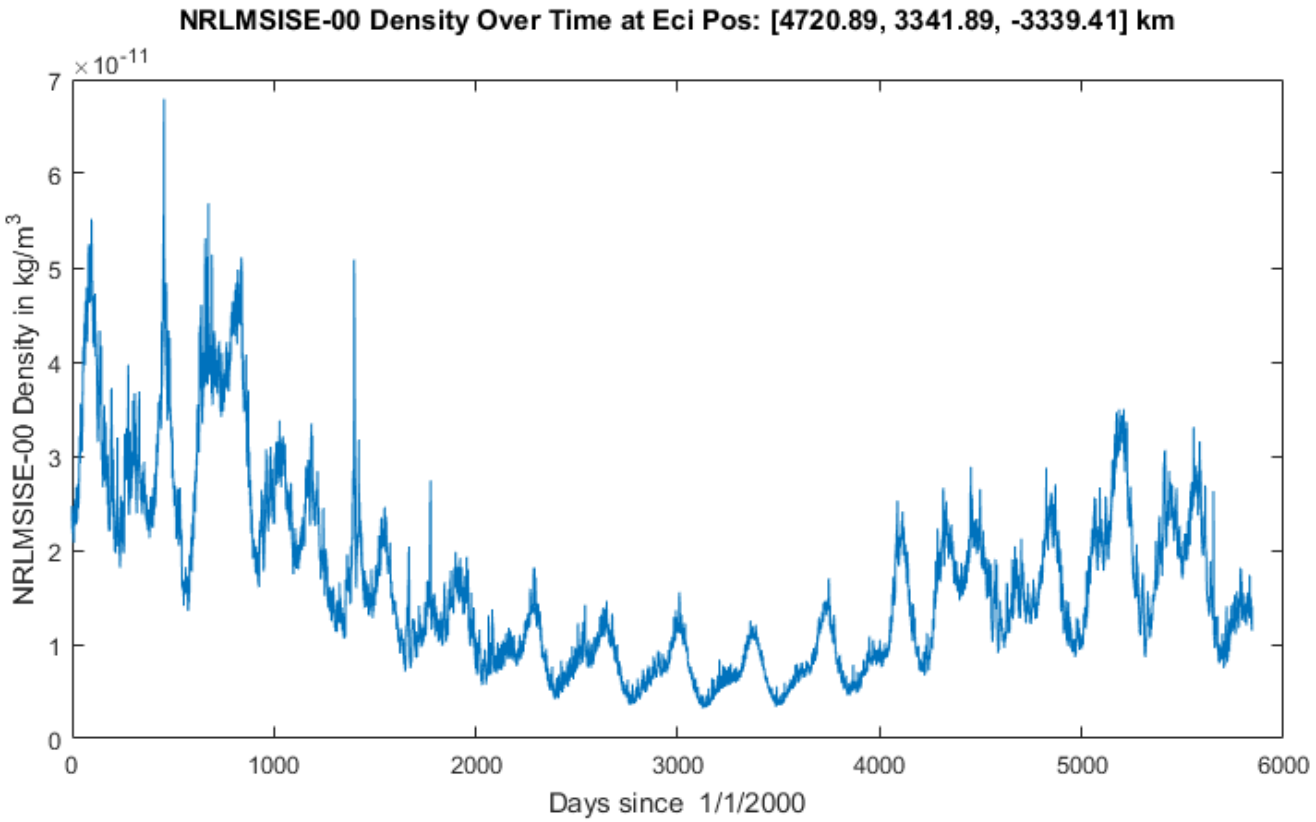
  - Density measurements from NRLMSISE-00 propagation used to create new standard atmosphere
- This model used for future targeting iterations
  - Results in shorter run times and lower final errors
  - Average difference in drag between NRLMSISE-00 and “new standard” densities generally under 10%
    - Errors in drag estimation corrected by inner loop guidance tracker

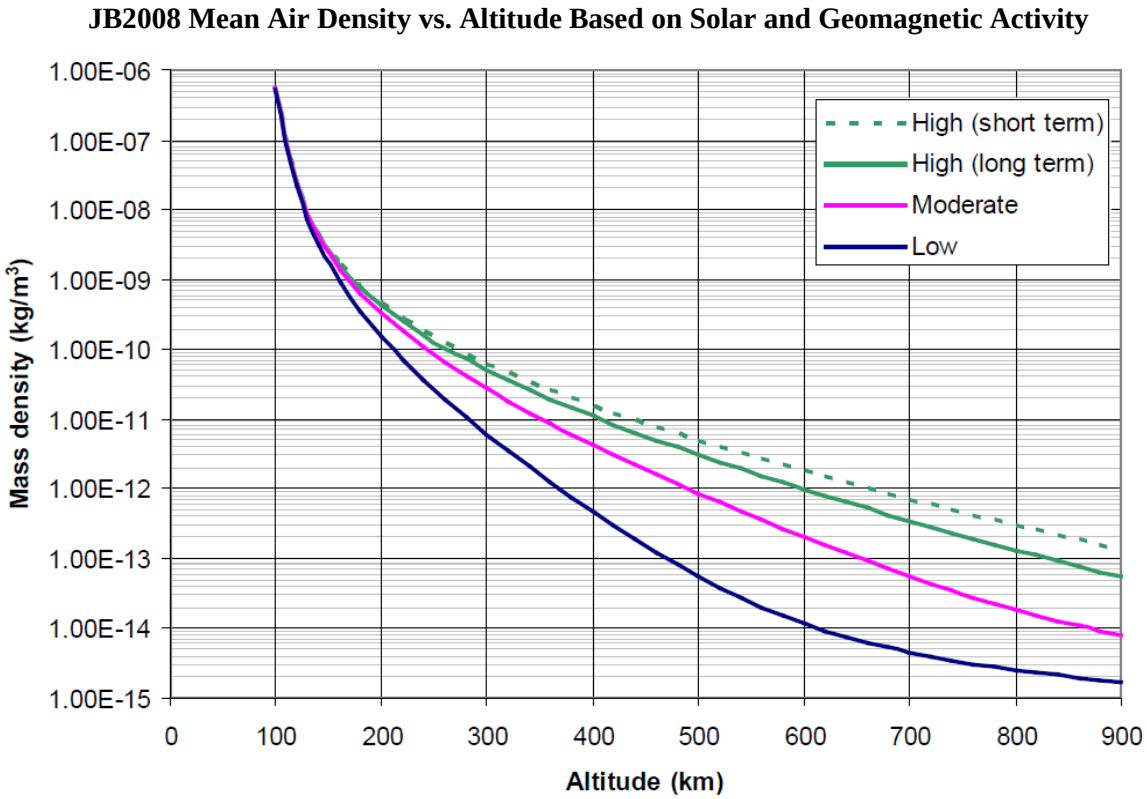
Example of New Atmosphere



Density Variations Over Time

These variations necessitate the use of advanced models like NRLMSISE-00

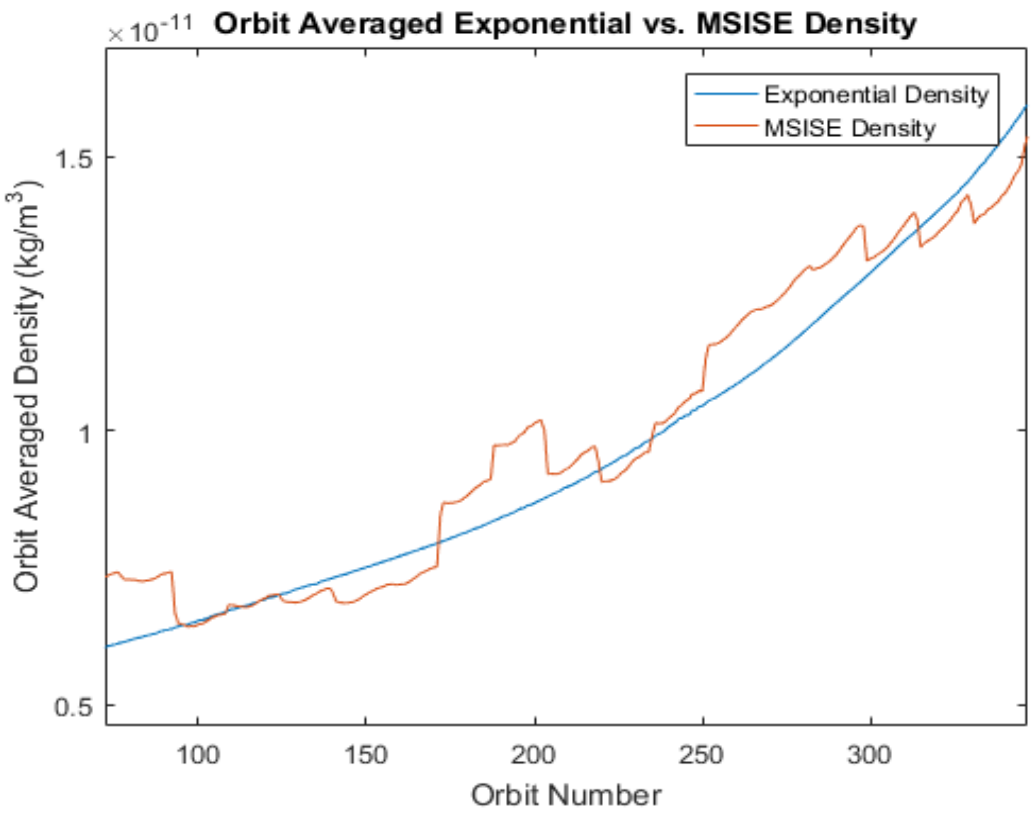


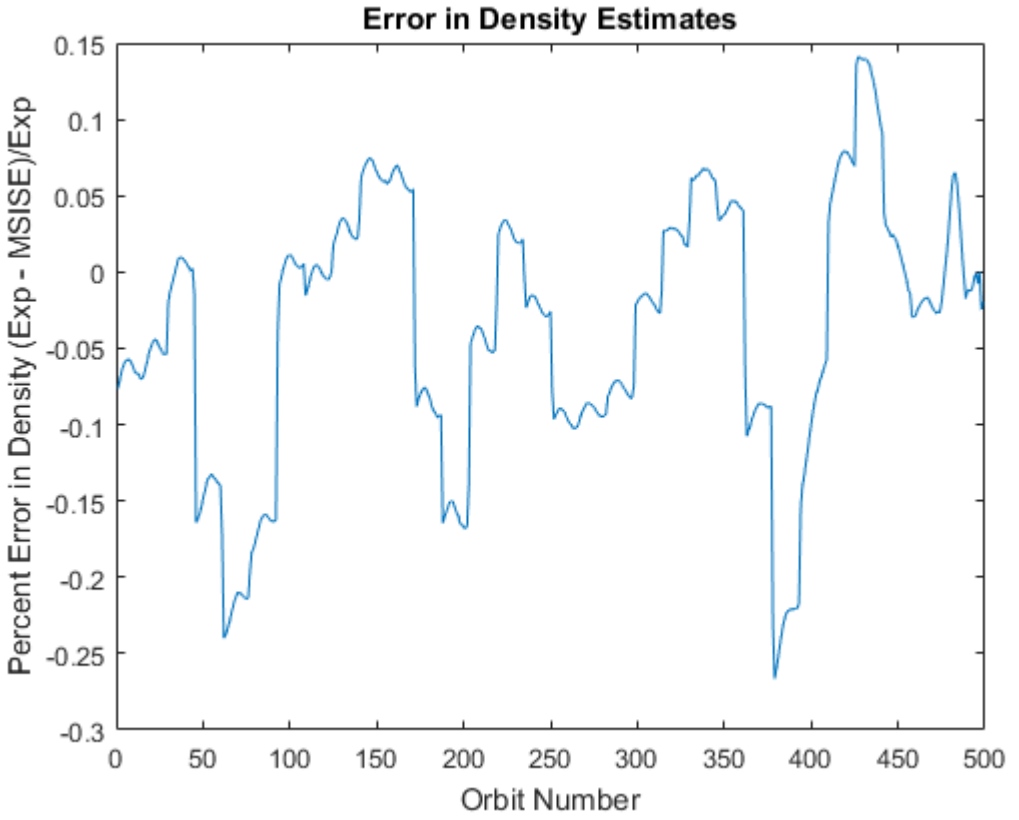


International Organization for Standardization, "ISO/DIS 14222, Space Environment (natural and artificial) - Earth Upper Atmosphere," Sep. 2013

Errors between exponential and MSISE densities

- init_mean_elems = struct('semi_major', 6720, 'ecc', .0037, 'right_asc', 4.843, 'arg_per', .4653, 'true_anom', 2.0547, 'inclination', .7001)
- epoch = [2006, 10, 12, 8, 15, 14.2]; % Simulation epoch [year month day hour minute second] utc time
- targ_latlong = [.2939 2.5943]; % radians
- Cb_min = 0.0232, Cb_max = 0.0359
- Guidance generated using density from “New” exponential atmosphere
- Goal is less than factor of two error between estimated and actual drag force





Affects of Atmospheric Rotation

- Atmosphere rotates at approximately the same rate as Earth due to viscous forces
- Velocity to use in drag equation is not orbital velocity but velocity relative to airstream

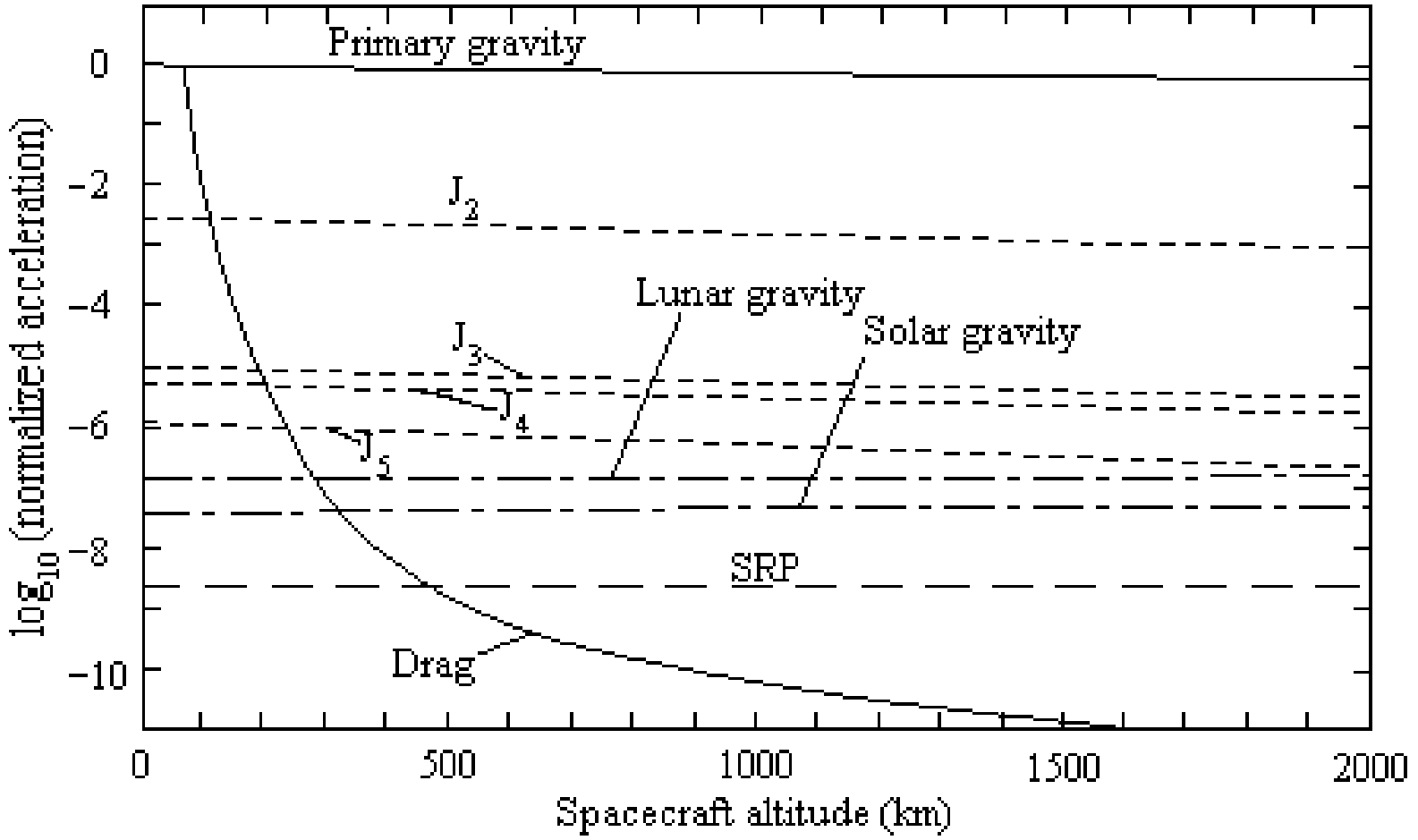
$$\vec{v}_{\infty} = \vec{v}_{orb} - \vec{\omega}_{\oplus} \times \vec{r}$$

$$F_d = \frac{1}{2} C_d \rho A \vec{v}_{\infty}^2$$

- Makes a significant difference in drag force and increases convergence time
- Must target de-orbit point (where drag force exceeds gravitational force) instead of ground
- De-orbit point usually around 120 km (satellite melted by this point)
  - Relative wind blows satellite hundreds of km off course when targeting ground

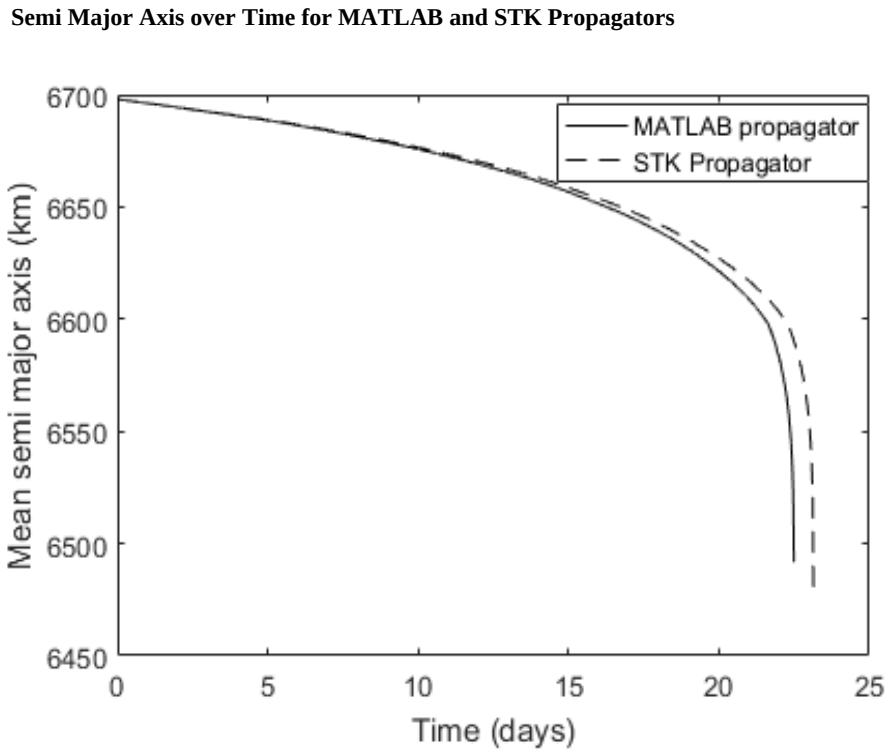
Other perturbations

- Image from astro.cornell.edu
- Simulation includes drag, primary gravity, and J2-J4
  - Other perturbations insignificant in LEO



STK Validation

- Validated MATLAB propagator using STK HPOP propagator with historic NRLMSISE-00 densities
  - Agreed within 3%
  - Main difference due to method of altitude calculation
    - MATLAB sim used altitude above ellipsoid while STK used geodetic altitude
- Created module to use STK propagator for guidance generation
  - All tested cases converged (see table below)
  - Half of run time was MATLAB overhead
    - Could get very fast algorithm with c++ implementation



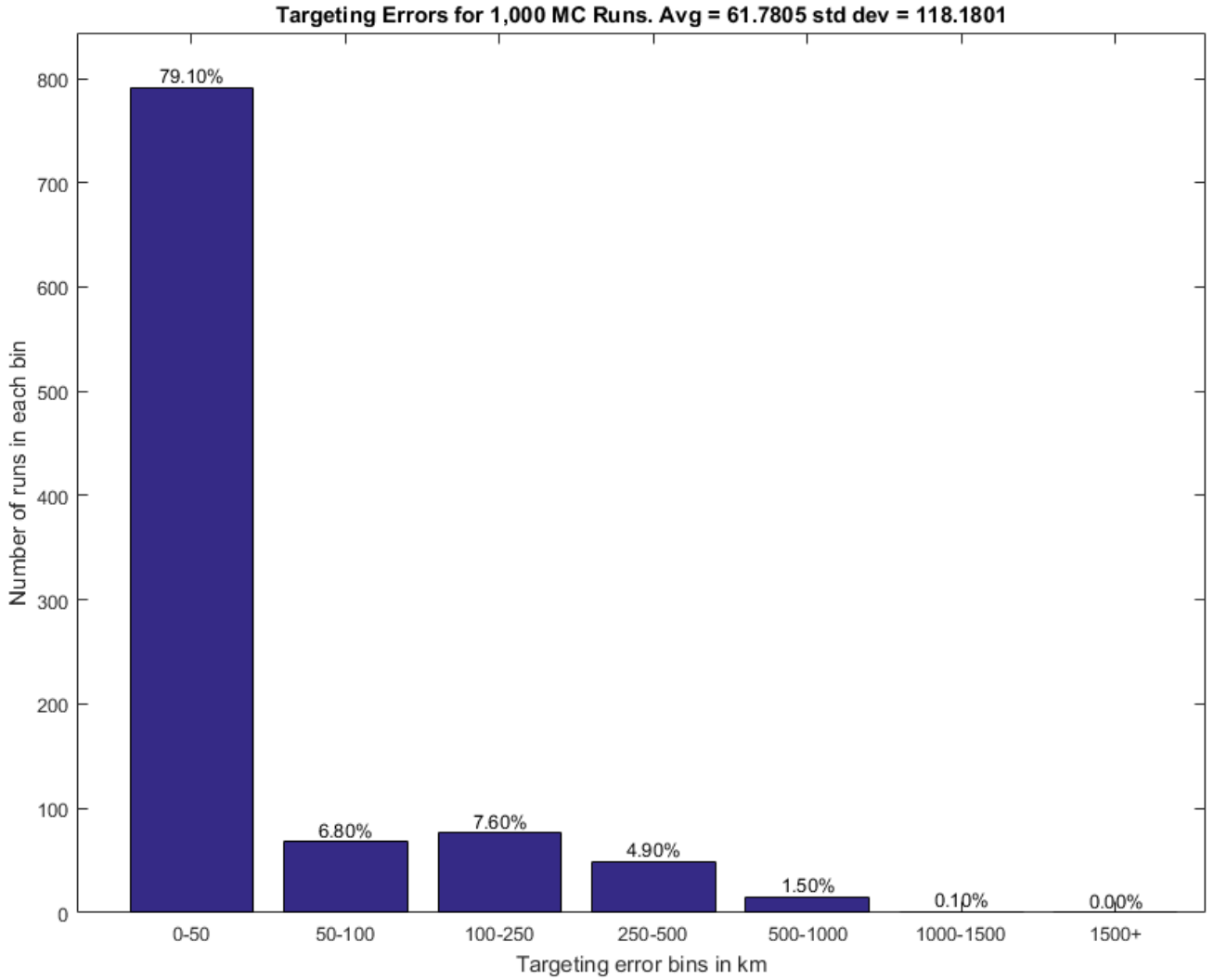
Initial Orbital Elements (a[km], e, Ω[deg], ω[deg], θ[deg], i[deg])	Epoch [y m d h m s]	Target (lat[deg], long[deg])	(C_{b1}, C_{b2}, C_{b_term}, t_{swap})	Number Numerically Propagated Trajectories	Orbit Life (hours)	Total targeting error (km)	Simulation Run Time (s)
(6708, 0, 0, 0, 90°, 45°)	[2015 3 1 0 0 0]	(20, 60)	(.0225, .0106, .0175, 143.5)	25	393.3	19.6	699
(6688, .004, 0, 0, 45°, 60°)	[2015 4 1 0 0 0]	(-30, 40)	(.0128, .0120, .0175, 140.1)	36	456.7	30	1174
(6678, 0, 180, 0, 60°, 90°)	[2015 5 1 0 0 0]	(10, 200)	(.0250, .0100, .0175, 73.6)	14	426.9	94	426
(6698, 0, 0, 0, 30°, 90°)	[2015 6 1 0 0 0]	(85, 100)	(.025, .01, .0175, 96.4)	10	518.1	83.2	391
(6698, 0, 90°, 0, 45°, 90°)	[2015 7 1 0 0 0]	(-60, 0)	(.0127, .0148, .0175, 215.8)	51	506	323.7	1796

Monte Carlo Validation

- 1000 Monte Carlo runs with randomly generated initial conditions and simulation parameters
- MATLAB simulator used with historical NRLMSISE-00 densities, J2-J4 zonal harmonics, rotating atmosphere, and non-spherical Earth for altitude calculations
  - Trajectory re-propagated between analytical solutions until convergence with numerical solutions

Variable	Range	Probability Distribution
Semi Major Axis	[6668, 6778] km	Uniform
True Anomaly	[0, 360] degrees	Uniform
Eccentricity	[0, .004]	Uniform
Right Ascension	[0, 360] degrees	Uniform
Argument of the Periapsis	[0, 360] degrees	Uniform
Inclination	[1, 97] degrees	Uniform
Impact Latitude	[0, inclination-.001] degrees	Uniform
Impact Longitude	[-180, 180] degrees	Uniform
Cb_{max}	[.033, .067]	Uniform
Cb_{min}	[.0053, .027]	Uniform
epoch	[11/1/2003, 11/1/2014]	Uniform

Targeting Error Histogram and Average Results

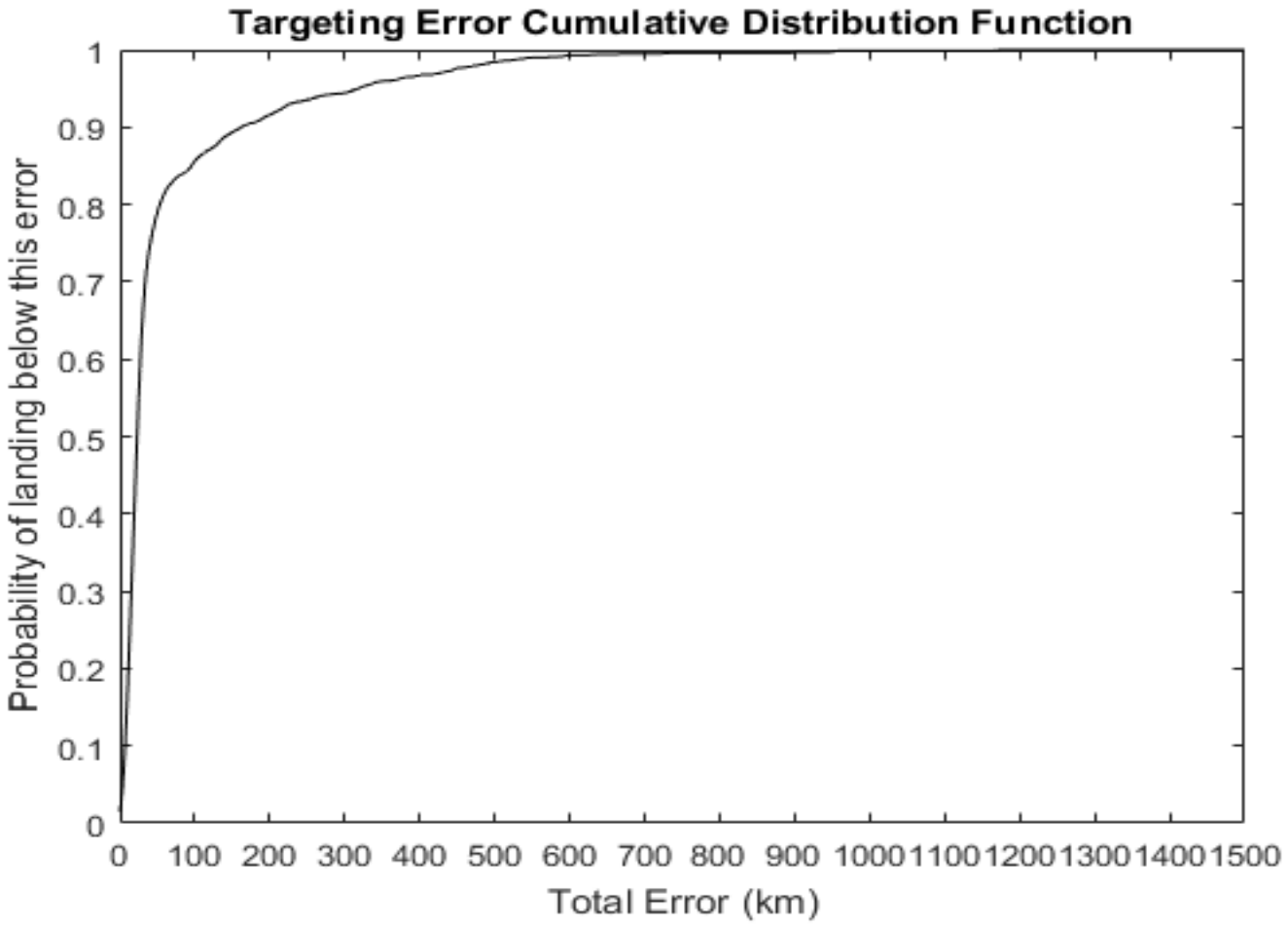


Average Simulation Results

Total Error (km)	Longitude Error (km)	Latitude Error (km)	Orbit Lifetime (days)	Sim. Run Time (mins)
61.8	53.4	16.0	16.8	59

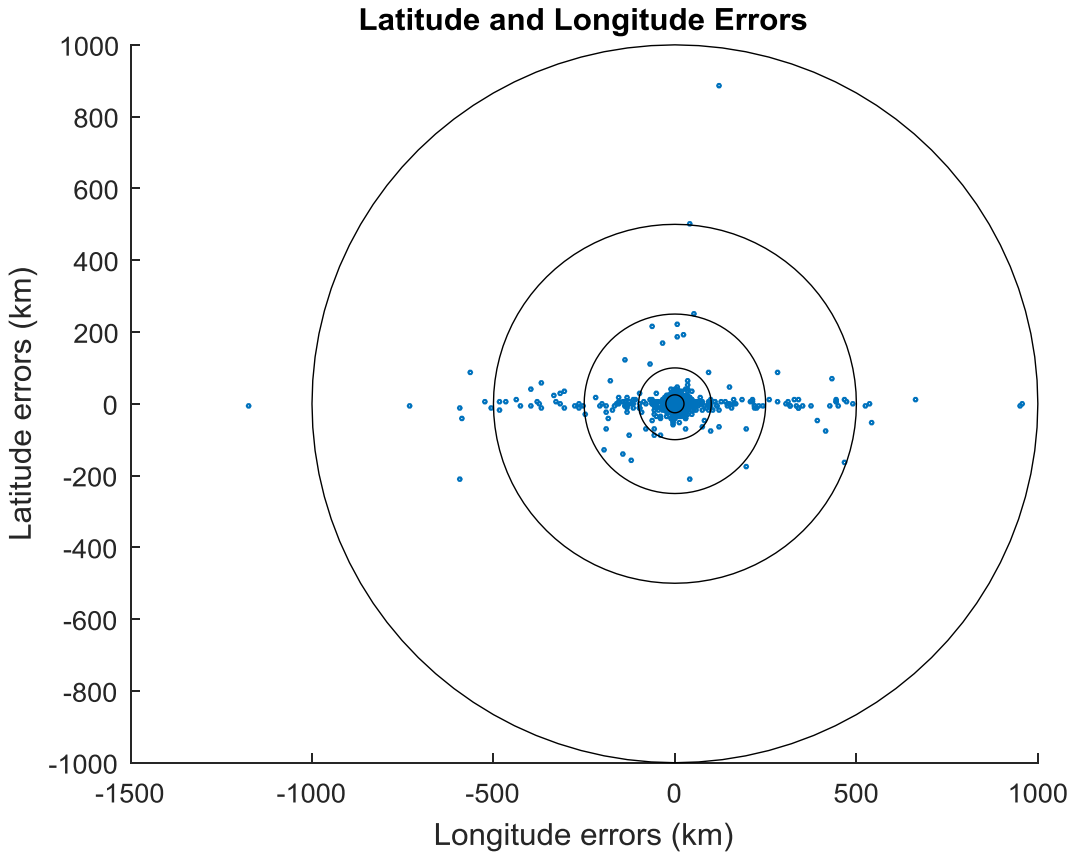
Error Cumulative Distribution Function

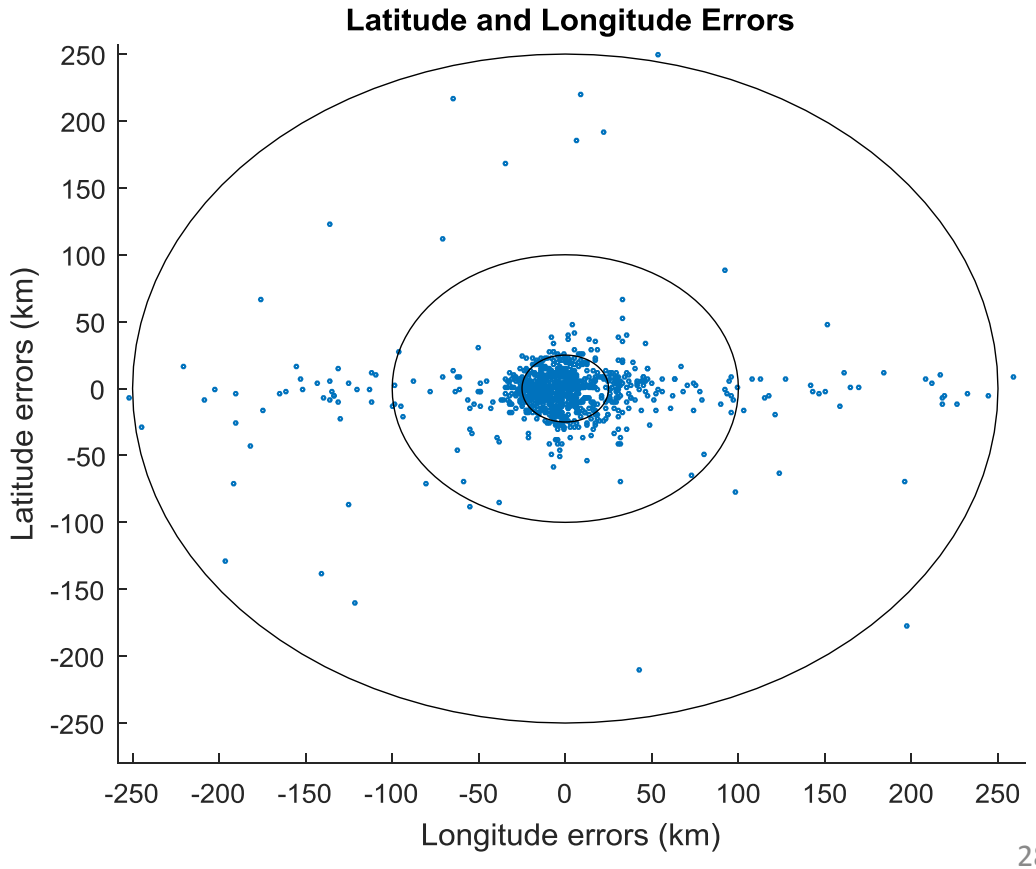
- Matlab fitdist function used with ‘kernel’ argument to get estimate of targeting error probability distribution function (pdf)
- Pdf integrated to get cumulative distribution function (cdf)
- 98.4% change of less than 500 km error



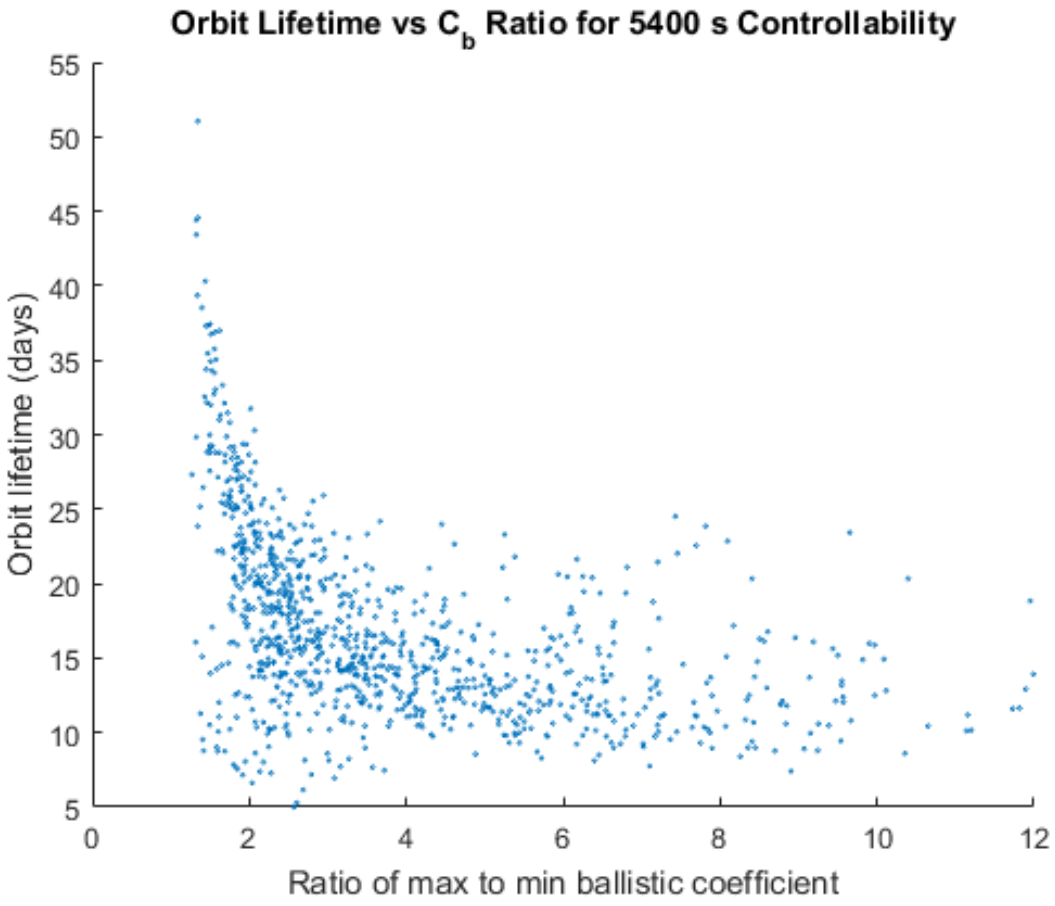
Latitude and Longitude Errors from MC Runs

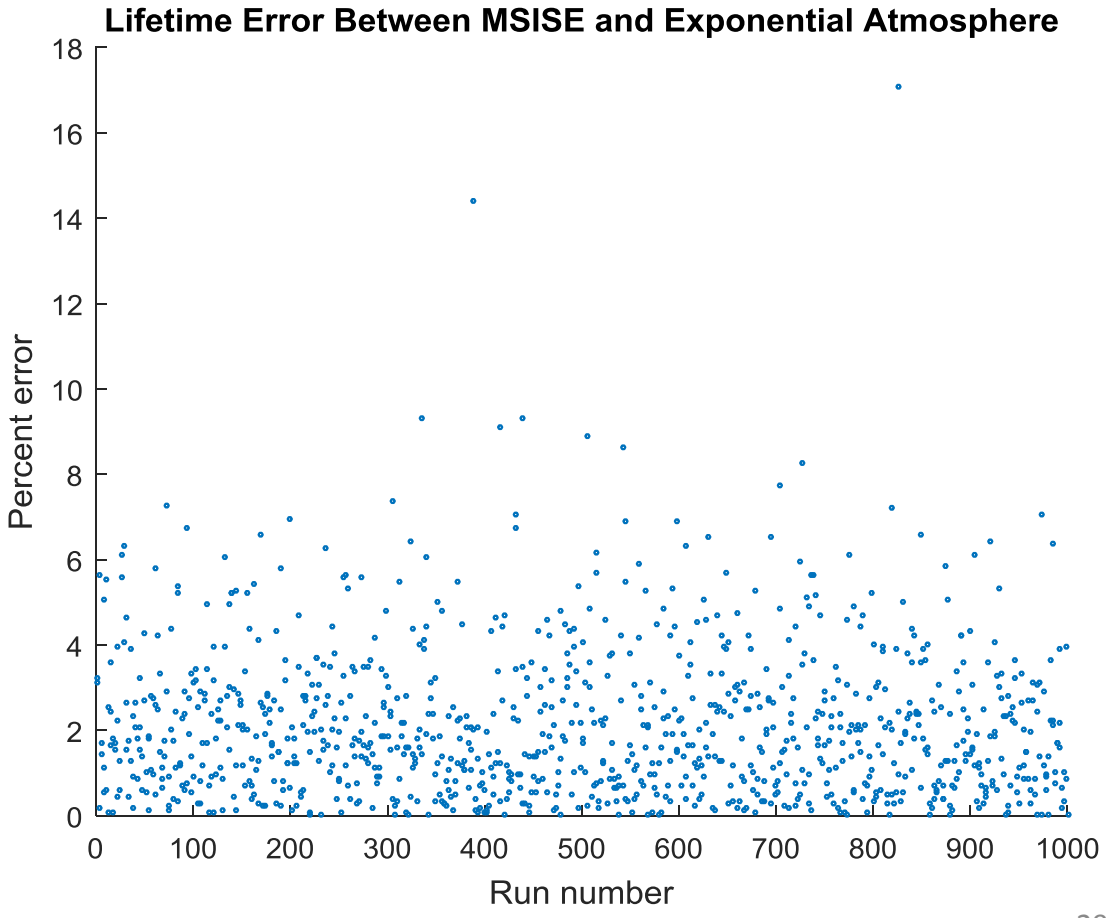
- Circles represents 25, 100, 250, 500, and 1000 km errors





Orbit Life and Density Errors





NASA Re-Entry Debris Mitigation Requirements

- NASA-STD-8719.14A section 4.7.2 part *b* mandates that “no surviving debris impact with a kinetic energy greater than 15 joules is closer than 370 km from foreign landmasses, or is within 50 km from the continental U.S., territories of the U.S., and the permanent ice pack of Antarctica”
- Part c of this section mandates that the product of the probability of re-entry targeting failure times the probability of human casualty from the uncontrolled re-entry not exceed .0001
- All 1,000 cases had well under 1500 km total error
- Many points in the ocean have with no land within 1500 km
  - Targeting algorithm good enough even for launch vehicle upper stage disposal (assuming the guidance tracking works)
  - Could meet error requirements with only latitude targeting component

Conclusions and Practical Considerations

- Error always under 1250 km
- Longitude controllability more limited than latitude controllability
- In current MC runs, targeting starts as 5400s (1 orbital period) longitude controllability
  - System can vary orbit life by 5400s without varying total change in true anomaly
- Starting later has several effects
  - Propagation time shorter due to shorter orbit life
  - Less deviation between numerical and analytical solutions
    - Fewer iterations before convergence
    - Lesser likelihood of convergence failure
    - Can call NRLMSISE-00 model directly instead of using exponential density
  - Controllability more limited
- Better to start early and periodically regenerate guidance
  - Benefits of the early and late starts reaped
- For keeping spacecraft debris away from people, latitude targeting only may be sufficient
  - Max longitude error 1250 km
  - Much simpler and faster
- System exceeds NASA debris mitigation requirements

Future Work

- Write inner loop guidance tracking algorithm
  - Desired true anomaly over time specified by numerically propagating trajectory with control parameters from guidance generation algorithm
  - Based on errors in true anomaly and rate of change of true anomaly, drag device deployed or retracted to track guidance
  - Adaptive control likely necessary to account for changing environmental conditions
- Additional Monte Carlo testing of algorithms
  - System should be able to handle expected errors and uncertainties in model
- Launch CubeSat to test the hardware and algorithms

Backup Slides and Detailed Derivations

Nomenclature

$t_{s_{old}}$ = swap time in the initial trajectory

$t_{s_{new}}$ = swap time in new trajectory

t_{term} = time until terminal point

C_{b10} = ballistic coefficient during initial trajectory from t_0 to $t_{s_{old}}$

C_{b20} = ballistic coefficient from $t_{s_{old}}$ until terminal point in initial trajectory

C_{b1} = ballistic coefficient during new trajectory from t_0 to $t_{s_{new}}$

C_{b2} = ballistic coefficient from $t_{s_{new}}$ until terminal point in new trajectory

$\Delta\theta_{10}$ = change in true anomaly from t_0 until $t_{s_{old}}$ in initial trajectory

$\Delta\theta_{20}$ = change in true anomaly from $t_{s_{old}}$ until terminal point in initial trajectory

$\Delta\theta_1$ = change in true anomaly from t_0 until $t_{s_{new}}$ in new trajectory

$\Delta\theta_2$ = change in true anomaly from $t_{s_{new}}$ until terminal point in new trajectory

$\Delta\theta_t$ = desired total change in true anomaly of new trajectory

Δt_{10} = time until swap point in initial trajectory

Δt_{20} = time from swap point until terminal point in initial trajectory

Δt_1 = time until swap point in new trajectory

Δt_2 = time from swap point until terminal point in new trajectory

Δt_t = total desired orbit lifetime of new trajectory

$\Delta\theta_d$ = difference in total change in true anomaly between the new trajectory and the initial trajectory

Δt_d = difference in total orbit lifetime between the new trajectory and the initial trajectory

Mapping from initial to final state cont.

- $\frac{da}{dt} = \frac{2a_d}{n}$ in a circular orbit (from Gauss Variation of Parameters)
- $n = \sqrt{\frac{\mu}{a^3}} = \frac{d\theta}{dt}$ in a circular orbit
- $\frac{d\theta}{da} = \frac{n^2}{2a_d}$
- $a_d = -C_b \rho V^2$ (drag acts opposite velocity)
- Rearranging these equations, we get
  - $\Delta t C_b = \int_{a_0}^{a_f} -\frac{da}{2\sqrt{\mu a}\rho}$
  - $(\Delta\theta) C_b = \int_{a_0}^{a_f} -\frac{da}{2a^2\rho}$
- If density is assumed to be a function of only semi major axis, then the time required for the satellite to fall from one semi major axis to another is dependent only on the values of final and initial semi major axes
  - The same holds true for the change in true anomaly during this time

Calculating Necessary $\Delta\theta_d$

- ϕ_i = true anomaly + argument of periapsis at de-orbit
- ϕ_d = true anomaly + argument of periapsis at desired latitude
  - Angle from ascending node line to impact point along orbit track
- Calculate z component of eci position vector at target latitude for a given set of orbital elements

$$R_z = |r| \sin(lat)$$

- Radius vector in the perifocal frame given by

$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \frac{h^2}{\mu(1 + e\cos\theta)} \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$$

- R_z calculated using the bottom row of the DCM from perifocal to ECI frame (given in "Orbital Mechanics for Engineering Student", Figure 4.16)

$$R_z = \frac{h^2}{\mu(1 + e\cos\theta)} [\sin(\omega)\sin(i) \quad \cos(w)\sin(i) \quad \cos(i)] \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$$

- R_z and all orbital elems known except θ (true anomaly)
- Use bisection method (matlab fzero) to solve for θ

$$\phi_d = mod(\theta + \omega, 2\pi)$$

- Two possible values for ϕ_d

$$\phi_{d2} = \pi - \phi_{d1}$$

$$\Delta\theta_d = mod(\phi_d - \phi_i, 2\pi)$$

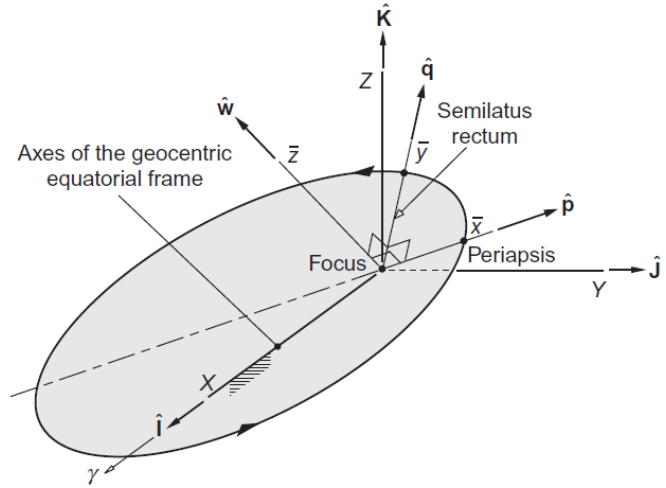
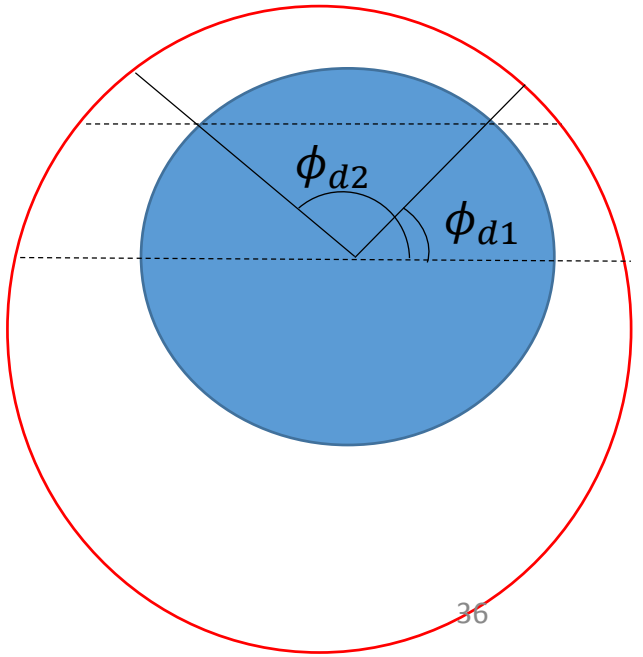


FIGURE 4.16

Perifocal ($\bar{x}\,\bar{y}\,\bar{z}$) and geocentric equatorial (XYZ) frames.



Initial Relations

- From the relations derived earlier for circular orbits and the assumption that the drag configurations are swapped at the same semi major axis in the new and initial trajectories:

$$\Delta\theta_1 + \Delta\theta_2 = \Delta\theta_t$$

$$\Delta t_1 + \Delta t_2 = \Delta t_t$$

$$\Delta\theta_1 = \frac{\Delta\theta_{10} C_{b10}}{C_{b1}}$$

$$\Delta\theta_2 = \frac{\Delta\theta_{20} C_{b20}}{C_{b2}}$$

$$\Delta t_1 = \frac{\Delta t_{10} C_{b10}}{C_{b1}}$$

$$\Delta t_2 = \frac{\Delta t_{20} C_{b20}}{C_{b2}}$$

Calculating the Control Parameters for Longitude Targeting

- From the previous relations, we can solve algebraically for the C_{b1} and C_{b2} required to achieve a desired $\Delta\theta_t$ and Δt_t

$$\Delta\theta_t = \Delta\theta_1 + \Delta\theta_2 = \frac{\Delta\theta_{10} C_{b10}}{C_{b1}} + \frac{\Delta\theta_{20} C_{b20}}{C_{b2}}$$

$$C_{b1} = \frac{\Delta\theta_{10} C_{b10} C_{b2}}{\Delta\theta_t C_{b2} - \Delta\theta_{20} C_{b20}}$$

$$\Delta t_t = \frac{\Delta t_{10} C_{b10}}{C_{b1}} + \frac{\Delta t_{20} C_{b20}}{C_{b2}} = \frac{\Delta t_{10}(C_{b10})(\Delta\theta_t C_{b2} - \Delta\theta_{20} C_{b20})}{\Delta\theta_{10} C_{b10} C_{b2}} + \frac{\Delta t_{20} C_{b20}}{C_{b2}}$$

$$C_{b2} = \frac{C_{b20}(\Delta t_{20}\Delta\theta_{10} - \Delta t_{10}\Delta\theta_{20})}{(\Delta t_t)(\Delta\theta_{10}) - (\Delta t_{10})(\Delta\theta_t)}$$

$$\Delta\theta_{t_{new}} = \Delta\theta_{t_{old}}, \Delta t_{t_{new}} = \Delta t_{t_{old}} + \Delta t_d$$

- Longitude targeting algorithm assumes that the swap point occurs at the same semi major axis for the new and initial trajectories
  - Must update $t_{s_{new}}$ to ensure this

$$t_{s_{new}} = \frac{t_{s_{old}} C_{b10}}{C_{b1}}$$